学校的理想装备

电子图书·学校专集

校园网上的最佳资源

# Linux操作系统（二）

# Networking

返回总目录

# 第三部分 Networking

# 目　　录

# 第 8 章电 子 邮 件

你的 Linux 系统拥有电子邮件程序，它能够让你给本地系统或者是其他在互联网中的系统上的其他用户发送消息。你能够通过多种途径来发送和接收消息，当然这要取决于你所用的电子邮件程序。这本书提供了 Linux 系统中两种最流行的电子邮件的使用方法：Mail 和 Elm.。它们都定义了一种不同类型的界面。虽然所有的电子邮件程序都同样能够执行发送和接收消息的基本任务，它们却会拥有不同的界面。Mail 程序使用一个基本命令行的界面，并在它自己的外壳程序中执行这些命令。在大多数的 Linux 系统上都能找得到 Mail 程序，并且它被公认为是一种标准。Elm 程序则采取一种全屏工作的界面，就像在 vi 编辑器中那样只使用一些单个键的命令。在附上这本书的 OpenLinux 系统中，上面所说的两种邮件程序都会提供。

此外，用电子邮件来发送消息，你就能不仅仅是依靠 Write 和 Talk 这两种实用程序来和目前在线上的用户通讯。这些工具为你和其他用户建立起一种直接的联系，使你能够就像用电话或是无线电一样进行通讯。

## 8.1 本地与互联网地址

每一个 Linux 系统上的用户都拥有一个邮件地址，并且无论何时，你若想

要发送邮件，你就必须要提供你想要发送消息的用户的邮件地址。对于那些就在本地 Linux 系统上的用户，地址只要包含该用户的注册名字就足够了。可是当你想要给其他系统上的用户发送消息时，你不仅仅要知道该用户的注册名，而且还要知道他们所在系统的地址。互联网络上的地址要求一个系统的地址必须是唯一可识别的。

大多数的系统都有让你发送邮件的互联网地址。这些互联网的地址采用一种叫做域名寻址的寻址方式。一个系统被分派一个域名，这个域名与系统名字结合起来，产生一个属于该系统的特有地址。域名由一个句号与系统名分开，也许还会用一些附加的域名来进一步限制。这是域名地址的语法：

login-name@system-name.domain-name

那些做为一个局域网的部分的系统，通常都被赋予同样的域名。比如说 U.C.Berkeley 中的 garnet 系统和 violet 系统的域名都叫做 berkeley.edu.假如要给 garnet 系统上的用户 chris 发送消息，你就只需要包括这个域名就可以了：

chris@garnet.berkeley.edu.

在下一个例子中，将使用域名寻址来把一条消息发送给在 garnet 系统上的用户 chris.$ mail chris@ garnet.berkeley.edu < mydata

早期的域名反映了互联网是首先在美国发展起来的这一事实。他们将互联网地址划分为若干类，比如商业，军事，或是教育系统。域名.com 象征着一个商业组织，同样的，.edu 则用来表示教育机构。随着互联网发展成为一个全球的互联网络，一套国际性的域名的命名标准建立起来。这些域名能够指明系统所在的国家。比如.fr 代表法国，.jp 代表日本，而 .us 则代表美国。表 8-1 列出了几种一般常见的国际域名。

## 8.2 邮件传送媒介：deliver，sendmail，smail

用邮件传输媒介将邮件发送给目标单元或从目标单元接收到邮件。Deliver 程序会处理你自己本地系统上的用户之间的邮件交换，sendmail 和 smail 则向网络上的目标系统或网络上的站点发送和接收邮件。为了在网络上发送邮件，它们采用了 Simpe Mail Transport Protocol(SMTP)。为了建立更加直接的 uucp 通讯，它们还使用了 uucp 协议。Sendmail 是一种比较小且易于配置的邮件传送媒介，而 smail 则相对要复杂些，并且拥有更加丰富的功能 。你的 Caldera 网络桌面管理程序会自动为你安装并配置好 sendmail 程序。只要你一启动你的 Caldera 系统，你就能够在互联网络上发送和接收邮件。如果你正在使用的并不是这种 Linux 系统，那么你也许只能自己亲手安装和配置这些邮件传送媒介了。当然你也可以从 Linux 的 ftp 站点上下载这些程序的升级版本，并在你的系统上安装它们。

在你系统上的用户能够在本地系统上发送和接收邮件之前，你必须要设置好一个返回界面。这将让你的系统能够确定地址，从而在系统内部发送和接收邮件。返回界面将使用 localhost 的主机名以及一个为本地系统使用而保留的特殊的 ip 地址：127.0.0.1.Caldera 系统的网络桌面管理程序的软件包中便已经安装了这样一个返回界面。

你可以查看你的/etc/hosts 文件来确认你的返回地址是否已经被设置为本地的主机。你会看到 localhost 127.0.0.1 被列在第一条条目上。如果并没有“localhost"这一条目，你就只能向下面这样使用 ifconfig 和 route 命令来自己设置返回地址了。其中 lo 是表示返回的术语。

Inconfig lo 127.0.0.1
Route addnet 127.0.0.0

检查在目录/etc/rc.d/下的网络初始化文件 rc.inet1，看看是否有以上这些命令。如果没有，你应该加上它们。一旦这些命令被加上去，在你每次启动系统的时候，返回界面就会自动被创建。在 Redhat 版本的 Linux 系统中，这个初始化文件则叫做 inet，放在/etc/rc.d/init/目录下。可以在本书的第 20 章网络管理上查阅到有关对这些命令和文件更详尽的阐述。

## 8.3 从远程 pop 邮件服务器上获得邮件：popclient

无论你是通过直接联网还是拨号到一个网络服务提供商（ISP）上，你的 linux 系统都被装备得能够在互联网上发送任何的消息。然而，如果你使用的是一个 ISP，你很有可能是用这个 ISP 的邮件服务器来接收邮件。你用来在该 ISP 上登录的用户名，通常被用做你在该 ISP 邮件服务器上的电子邮件名。比如说，如果你用 Larisa 这个用户名登录到一个 ISP 上，那么你的电子邮件名也叫做 Larisa。用这个电子邮件名发送给你的邮件会被放在该 ISP 的邮件服务器上，当你登录到服务器上时，能够在那儿获得这些邮件。这种用来接收邮件的邮件服务器通常是一个 pop 服务器。Pop 表示邮局协议邮件（Post Office Protocol）.

为了获得你的邮件你就必须登录到你的 ISP 的邮件服务器上。

象 mail 或是 elm 这样的程序会读取由你的 Linux 系统发送和接收到的邮件。目前的这些程序还没有从你的 ISP 邮件服务器上获得邮件的能力，因为那是分离开来的远程系统。然而不管怎样，你总能够使用 popclient 程序将邮件从你的

ISP 的 POP 服务器上下载到你自己的系统上，然后再使用 mail 或是 elm 来阅读它。

为了使用 popclient 你还必须知道你的 ISP 的 POP 服务器的网址。通常这些地址会以 popd 开头，例如 NETCOM 的 POP 服务器的网址就是 popd.ix.netcom.com。查看你的 ISP 来获得正确的地址名。你还需要确定这个 pop 服务器使用的是 POP2 协议还是 POP3 协议。绝大多数的系统都使用 POP3 协议。你可以用-3 来指明是 POP3 协议，同样的，用-2 来指明是 POP2 协议。此外，你还要提供你在该 ISP 的邮件服务器上的电子邮件名。因此你要加上一个后跟着电子邮件名的-u 选项。下面是一个对 pop3 服务器的 popclient 命令语法：

popclient -3 – u username pop-server

一旦你执行了上面这条命令，你会被提示输入密码。这将成为你在该 pop 服务器上电子邮件记录的密码。通常这与你用来登录到 ISP 上的密码是一样的，尽管有些系统允许你的密码不一样。如果你想跳过这个提醒输入密码的提示，你可以在执行 popclient 时，在命令行中包含-P 选项以及密码：

popclient -3 – u username – p password pop-server

使用 popclient 联接到你的 ISP 上，然后在命令行上输入带有选项和 pop 服务器名的 popclient 命令。你将会看到有消息告诉你是否有邮件，如果有的话，则显示有多少消息将会被下载。你能够用 mail，elm 或者其他任何一种邮件程序来读这些消息。

#popclient -3 – u mylogin – p mypass popd.ix.netcom.com

netcom（version: 1.6 Rel（SB））at ix10 starting:built n Jun 25 1997

16:07:34.

2 messages in folder

reading message 1.

Reading message 2.

［root@turtle/toot］#mail

Mail version 5.5 -kw 5/30/95. Type ?for help.

"/var/spool/mail/root":2 messages 2 new

>N 1 root@turtle.trek.com fri Aug 29 00:05 15/555 "birthday"

N 2 root@turtle.trek.com Fri Aug 29 00:05 15/552 "party"

&q

Held 2 messages in /var/spool/mail/root

下载下来的邮件会放在用户执行 popclient 命令的信箱中。如果你以 Chris 登录到到你的 Linux 系统上，那么这些邮件就会放在 Chris 的 mailbox 中。如果你是以 Larisa 来登录，那么邮件就会放到 Larisa 的 mailbox 中。你能够使用 -o 选项来指定一个特别的 mailbox，把邮件加到 mailbox 这个文件中，而不是靠系统来维护。

把 popclient 程序和 mail 程序命令结合起来使用是很方便的。这样的话，你就只需要执行一道命令让 popclient 来下载邮件，并启动一个邮件程序就可以读取它们。Popclient 有返回值可供你用来检查那儿是否还有邮件（popclient 的邮件页面上）。如果那儿有邮件的话，那么 popclient 将返回 0，如果没有邮件，popclient 则会返回 1，如果是发生了一个错误，则返回其他值。下面这个叫做 popelm 的程序用 popclient 来检查是否有邮件并下载它们。如果有邮件，那么 elm 程序就会被自动调用来显示它们。如果没有邮件，那么消息“No Mail”就

会显示出来（记得要用 chmod 755 script-name 来使这个程序可执行。）

```
Popelm
#!/usr/bin/bash
if popclent -3  – u mylogin  – p
mypass my-ISP-POP-server
then
  elm
  else
  echo "No mail"
fi
```

## 8.4 邮件实用工具

使用这些邮件实用工具，你能够很容易向其他用户发送和接收消息。发送消息就只是简单地敲入一个注册名，以及跟着敲入消息正文。接收消息则仅仅是从一个接收邮件列表上选择一条消息而已。当你用 Mail 发送消息时，有命令可以修改消息的正文以及消息头。接收到邮件后，你可以回复它们，把它们保存到文件中，或者仅仅是简单地删除它们。.mailrc 这个专门的初始化文件能够给你的 Mail 工具添加上如别名等特征。

现在为大家所知的 Mail 工具最初是为 BSD Unix 系统创建的，就简单地命

名为 mail.后来的 Unix 系统 V 也采纳了 BSD 的 mail 工具，并重新命名为 mailx. 大多数的 unix 系统，包括 Redhat Linux，都使用这个 mailx 工具，且直接用 Mail 来表示它。

### 8.4.1 发送邮件

你利用 Mail 来发送的邮件可以是从键盘上敲入的东西，也可以是某个文件的内容。当你从键盘上输入一条消息时，你可以用专门的符号命令来编辑它。这些符号命令可以让你保存消息，重新显示你所写的内容，或是启动一个编辑器来编辑它。

为了发送邮件，可以敲入一个单独的 mail 命令，并在后面加上你所要发给消息的人的邮件地址。按 ENTER 键后，那么会出现提示要你输入一个信件的主题。输入该邮件的主题，然后再一次按下 ENTER 键。这时候，你已经处于输入模式，任何东西输入都被做为信件的内容。按 ENTER 键将在信件正文加入新的一行。当你完成所有的输入时，在新的一行行首敲入 Ctrl+d 来结束这条消息，并把它发送出去。当你敲了 Ctrl+d 后，你会看到屏幕上显示 EOT（end-of-transmission）的字样。

在下面的例子中，用户把邮件发送给一个地址为 robert 的用户，邮件的主题是“ Birthday".在输完消息正文之后，他按了 Ctrl+d.

```
$ mail robert
Subject:Birthday
Your present is in the mail
Really.
```

^D
EOT
$

## 标准输入与重定向

Mail 工具能够从标准输入中接收输入。在缺省情况下，标准输入就是用户在键盘上的输入内容。然而利用重定向，标准输入则可以从一个文件中读取输入。因此，通过重定向，你就可以在 Mail 程序中使用文件的内容来作为要发送的邮件。你可以用 vi 编辑器来创建并编辑一个文本文件，然后把这个文件作为 Mail 程序的重定向输入。在下面的这个例子中，文件 mydata 被重定向为 Mial 程序的输入，并发送给用户 robert.

$ mail robert<mydata

注意到当你利用重定向来发送一个文件时，你并没有机会来给这封邮件输入一个题目。然而，mail 命令的 -s 选项允许你在命令行中给邮件指定一个主题。这个主题将被显示在该文件接收者的邮件开始部分。表 8-2 列举了 Mail 程序的其他几个选项，我们将在本章的后面部分对这些选项做进一步的阐释。下面的这个例子中，Robert 将文件 mydata 发送出去，并指定邮件的主题为“ party ”。

$ mail – s party chris<mydata

## 向多个用户发送邮件

你可以同时将一条消息发送给多个用户，只要在 mail 命令后的命令行参数上列出那些用户的地址。在下面的例子中，该用户将一封邮件同时发送给 chris 和 aleina.

```
$ mail chris aleina
Subject : Birthday
Your present is in the mail
Really.
^D
EOT
$
```

你也可以利用重定向把某个文件的内容同时发送给几个用户。下面的例子中，文件 mydata 的内容就同时被送给了 robert 和 aleina.

```
$mail robert aleina <mydata
```

## 将邮件拷贝到文件中

你可能还想为自己保存一份所发送消息的拷贝。在命令行上的地址后指定一个文件名，就可以将消息备份到你帐号下的文件中。这个文件名必须是一个包含有斜杠的相对或绝对路径名。路径名确定了一个文件名参数，作为正在发送的邮件要保存到的文件名。在下面的例子中，该用户将邮件的一个拷贝备份到名为 birthnote 的文件中，它采用的是一个相对路径名，以及用来表示当前工作目录的句号：./birthnote

```
$ mail robert ./birthnote
Subject: Birthday
Your present is in the mail
Really.
^D
EOT
```

```
$ cat birthnote
Subject: Birthday
Your present is in the mail
Really.
$
```

**当使用重定向或是同时给多个用户发信时，技巧也是一样的**

```
$ mail robert aleina < mydata ./birthnote
```

### 8.4.2 编辑 Mail 邮件：转义命令

**一封邮件由两部分组成：消息头和消息正文。消息头中包含了该邮件的信息，比如邮件要发送到的地址以及邮件的主题等内容。地址是作为 mail 命令的参数指定的。然后 Mail 工具会提示用户输入该邮件的主题。输完主题之后，用户就开始输入邮件的正文。**

**那一系列消息处理指令，即人们所知的转义命令，允许你在消息头或是正文中进行编辑操作。一个转义命令由波浪号~后加一个单独的字母组成，波浪号要放在一行的行首。波浪号（充当一种特殊字符的角色）及其后的命令并不做为消息的一部分。如果你需要在消息中输入波浪号的字符，那么你可以连续地输入两个波浪号（~~）来获得这种效果。这些使用波浪号的转义命令在表 8-2 中列了出来。你还可以通过~？命令来获得这写命令的列表。**

**当你发送邮件时，有三种基本的操作：输入消息头的信息，输入邮件的正文，以及输入转义命令来对消息头或正文进行操作。对消息头操作的转义命令允许你修改地址以及主题等信息，而对正文进行操作的转义命令则允许你重新显示、保存或是用编辑器来修改正文。**

## 编辑正文的转义命令

在输入一封邮件的正文时，并不像一个编辑器那样有命令模式。一旦你输入了主题并按回车，你就进入了该邮件的正文输入模式。Mail 工具的输入模式受到所有在 vi 编辑器的输入模式下的各种限制。你只是在输入一串字符流，唯一能够做的改正工作就是使用 BACKSPACE 键，来抹去该行上光标左边的字符。你不能够移动光标去执行其他任何的编辑操作。

通过~v 命令来启动 vi 编辑器，你能够不受所有 Mail 输入模式下的限制。你可以在一行中单独敲入~v 命令，然后回车。一旦你进入了 vi 编辑器，你的邮件就能够象其他任何的文本一样编辑了。实际上，你是在用 vi 编辑器来编写你的邮件，而不是依靠 Mail 的输入模式了。在 vi 编辑器中，你所输入的的消息会被显示成为一个可编辑的文本。

Vi 编辑器的保存命令将编辑好的正文保存到邮件中，而不是保存到一个文件中。当你使用命令 ZZ 来退出 vi 编辑器时，你将文本保存为 Mail 的消息正文，并返回 Mail 工具的输入模式。无论如何，该消息的正文都不会被重新显示一遍，而是在屏幕上出现一个用括弧括起来的单词“ continue"。你可以接着输入更多的内容，执行其他的转义命令，或是用 ctrl+d 来结束编辑并发送邮件。

虽然启用 vi 编辑器是最有用的转义命令之一，你却可以对正文做一些基本的操作，比如重新显示正文，将正文内容保存到文件中，或是从另一个文件中读入信件正文。在任何时候，你都可以敲入~p 转义命令来重新显示你的邮件内容，它将会将在此之前所输入消息都打印出来。当你在一行中单独敲入 ctrl+d 并回车，你所输入的一切都会被显示出来。通常都会在结束调用 vi 编辑后，敲入~p 命令来确认一下编辑的结果。如果你这样做了，你将会发现在 vi 编辑器中

编辑及输入的内容都被显示成消息的正文。

```
$ mail aleina
Subject: Files
This is a list of all the
Students in my class.
~p
Message contains:
To: aleina
Subject: Files
This is a list of all the
Students in my class.
（continue）
^D
EOT
$
```

还有其他一些转义命令允许你将你的邮件保存到文件中，或是将某个文件的内容读到你的邮件中。转义命令~w 保存你的邮件，而命令~r 则从另一个文件读入文本，成为邮件的一部分。敲入命令~w 和要把邮件写入的文件名。命令~r 则多与要读的文件一同敲入。在下面的这个例子中，指令~w mydata 把输入的消息保存到文件 mydata 中。然后用~r 命令来读取文件 mynames 的内容，并把它添加到 Mail 的消息正文内。

```
$mail aleina
Subject: Files
This is a list of all the
```

```
students in my class.
~w mydata
"mydata" 2/48
~r mynames
"mynames"   3/15
~p
Message contains:
To: aleina
Subject: Files
This is a list f all the
students in my class.
Mary
Joe
Harold
 (continue)
^D
EOT
$
```

如果你突然改变了主意，不想再把这封信息发送出去，你可以用~x 或是~q 转义命令来退出 Mail 程序。这时刚才所写的文件就会丢失，回到外壳程序 shell 中。如果你使用的是~q 命令，那么刚才所编辑的消息文本就会保存到一个名为 dead.letter 的文件中。

```
 $ mail aleina
Subject: Files
```

```
This is a list of all the
students in my class.
~x
$
```

你还可以通过过滤器来对消息的正文进行加工。把当前的消息正文当做过滤器的输入，就可以得到一个可以取代原来消息正文的文本。比如说，如果你的消息包含一个列表，你可以将这个列表通过管道送到 sort 过滤程序中，然后得到被排序后输出代替的消息内容。转义命令~|允许你将你的文本用管道输送给一个过滤程序。~|把该过滤程序当作它的命令参数。编辑的消息文本被当作标准输出送到管道，再输送到过滤器中。从过滤器得到的输出会取代原来的消息内容。在下面的例子中，用户利用~| sort 命令来对消息中的列表排序。

```
$ mail george
Subject: Names
Mary
Joe
Harold
~| sort
~p
Message contains:
To:george
Subject: Names
Harold
Joe
```

```
Mary
(continue)
^D
EOT
$
```

一个非常有用的过滤程序是 fmt，它是为格式化 Mail 消息而设计的。通常当你输入一条消息的时候，文本的每一行并不具有相同的长度，这是因为他们是在一个词处理程序中。而过滤程序 fmt 则可以格式化你的文本，使每一行都有大约 72 个字符的标准长度。以空格或是制表符开头的行会被当作一段的开头。下面的例子中，该用户利用 fmt 来格式化消息文本。

```
$ mail george
Subject: Title
George,
Have you thought of a new
Title for the
Project we were discussing
Last month?
It should have a new theme based on
The realistic expectations of
our target audience.
~| fmt
(continue)
~p
```

```
Message contains:
To: george
Subject: Title
George,
Have you thought of a new title for the project we were discussing last
month? It should have a new theme based on the realistic expectations of our
target  audience.
(continue)
^D
EOT
$
```

## 编辑消息头的转义命令

其他的一些转义命令允许你改变消息头的各个部分。在一个消息头中，可能有以下四个部分组成：接受者的地址列表，主题，拷贝清单以及隐蔽的拷贝清单.这两个拷贝清单是可要可不要的。但你的消息头里却一定要有地址列表，而且虽然你可以跳过主题这一栏，但通常情况下，系统会提示你在主题栏里输入些东西。

通常启动 Mail 的时候，你最初在命令行上输入的是那些信件要发送到的地址列表。但假如你是在编辑你的信件正文时临时想添加其他地址，那么你只要通过使用转义命令~t，就完全可以做到这一点。

而~s 命令则允许你为你的信件输入一个新的主题。为了更改主题这一项内

容，你可以在~s 命令后跟上新的主题。下面的例子对主题和地址列表都做了更改。Larisa 这个地址被加到地址列表中，现在，邮件不但会象原来那样发送给aleina 而且还会发送一份给 Larisa。然后又把主题“ Files ”改成“ Class Roster ”。

```
$ mail aleina
Subject: Files
This is a list of all the
students in my class.
~t larisa
~s Class Roster
Message contains:
To: aleina larisa
Subject: Class Roster
This is a list of all the
students in my class.
（continue）
^D
EOT
$
```

也许你还想把这个消息的副本抄送给其他人。那么这个副本中将会包含有它被抄送的地址。实际上，你正在给某个人发送的此邮件副本并非只是邮件的内容，而包括了邮件的消息头部分。为了发送邮件的副本，你可以通过~c 命令来建立一个拷贝发送的清单。输入~c 并在后面加上接收这些消息拷贝的地址列表就可以了。

每一个收到消息副本或是那些收到消息原件的人都会在邮件的尾部看到一

个拷贝发送清单。如果你不希望某个人的地址被显示出来，但又希望他能够收到这个消息的一个副本。那么你可以用隐蔽的拷贝清单来达到这一目的。输入~b命令，并在其后写入那些你不希望在其他接受者的拷贝清单中显示的地址。只有那些在普通拷贝发送清单中的地址才会出现在邮件中。在下面的这个例子中，larisa 和 marylou 会收到该消息的拷贝，并且在每一份发送的邮件拷贝上都能看到他们的地址被打印在拷贝清单中。另一方面，象 valerie 那样则也会收到一份该消息的拷贝，但在任何一份拷贝中都不会看到她的地址被列出来。

```
$ mail aleina
Subject: Files
This is a list of all the
students in my class.
~c larisa marylou
~b valerie
~p
Message contains:
To: aleina
Subject: Files
Cc: larisa marylou
Bcc: valerie
This is a list of all the
students in my class.
（continue）
^D
EOT
```

```
$
```

如果你想更改消息头中的全部内容，使用转义命令~h 就可以一下子对它们全部进行修改。系统首先会提示输入一个新的地址列表，然后是主题，最后是一个新的拷贝发送清单。

### 8.4.3 接收邮件：Mail 外壳程序

当你有新信件到达时，它们会被放在你的邮箱中。所谓的邮箱其实也就是一个文件，里面装着最近收到的所有信件，并且会一直保存这些信件直到你重新取出它们。只要在命令行上敲入一个单独的 mail 命令，就可以启动 Mail 工具来取出这些保存着的信件。Mail 工具是一个很复杂的程序，可以用来管理收到的邮件，它有着自己的外壳程序以及一套专有的命令和提示。一旦你启动了 Mail 工具，你就进入了 Mail 的外壳程序，在那里你可以接收邮件，回复邮件，甚至是发送一封新的邮件。在表 8-3 中列出了最常用的几种 Mail 命令。

如果你的邮箱中并没有信件，屏幕上就会出现一个提示告诉你没有邮件。只有当邮箱中有信件存在时，你才能够进入 Mail 的外壳程序。在下面的这个例子中，该用户试图进入 Mail 的外壳程序，但是因为邮箱里并没有邮件，所以屏幕上只是出现了一个简短的提示，用户依然留在他登陆的 shell 中。

```
$ mail
Sorry,  no mail
$
```

当你刚进入 Mail 的外壳程序时，一个包含有关于每条消息的扼要信息的列表会打印在屏幕上。这些扼要信息前面还有用来表示该消息状态的状态指示符以及消息序号。状态指示符是用一个单独的大写字母来表示的，通常都会用一个 N 来表示“新邮件”，用 U 来表示“未读过的邮件”。接着状态指示符一栏的是消息的数目，用来方便用户指明是哪一条消息。接下来的几栏则是发信者的地址，收到信件的日期、时间，然后是该消息的行数和字数，最后一栏则是发信者为这封信起的主题。在显示完这些扼要信息之后，Mail 的外壳程序就会显示它的提示符——一个问号“?”。在这个提示的后面，输入命令就可以对消息进行操作。这里是一个 Mail的消息头信息显示出来的例子。

```
$mail
Mail version 5.5-kw 5/30/95. Type ? for help
"/var/spool/mail/chris": 3 message 3 new
>N 1 valerieTue Feb 1110:14:325/44    "Budget"
N 2 aleinaWed Feb 1212:30:1728/537   "Birthday"
N 3 robertFri Feb 14 8:15:2416/293     "Homework"
```

## 消息列表和当前消息指示符

Mail 可以通过消息列表或是一个当前消息指示符（>）来指定某一条消息。大于号被放在某条消息的前面，表示这是当前的消息。当 Mail 命令中不指定消息的数目时，命令就会缺省地对当前消息做操作。比如说，在显示消息头信息时，如果命令中并没有给出信件的顺序号，那么消息一就会首先显示出来，因为它就是当前所指向的消息。如果在命令中给出了消息序号，那么这条消息就

自动成为当前消息，而当前消息指示符也会自动移到它的前面。如果你想要显示消息二，那么消息二也就成了当前的消息。

你还可以用消息列表来指定某条消息。许多 Mail 命令都能够同时对若干条消息进行操作。在一个 Mail 命令中，通过一个包含有若干个消息序号的消息列表，就可以指定若干条消息。你还可以输入首末两篇消息的序号来确定一个范围，两个序号之间用破折号隔开。在前面给出消息头信息的例子中，你可以用 1-3 来指定这三篇消息。

你还可以用专门的字符来指定当前的消息。^符号表示的是第一条消息，比如说，^-3 就表示从第一篇到第三篇的这么一个范围。而符号$则表示最后一条消息，所以 4-$就表示了从第四条消息到最后一条消息。单独的一个$就表示是最后一条消息，而单独的一个句号.则表示当前的消息。星号*则表示邮箱中的全部邮件。比如说，如果你想要显示所有消息，只要输入命令 p *就可以了。

你还可以根据发信者的地址或是信件的主题来选择一组信件。一个单独的地址就包含了一个消息列表，指向来自这个地址的所有信件。robert 就是一个指向所有来自 Robert 消息的列表。在一个斜杠后加上一个模式名，则指代所有主题中包含有此模式的消息。比如说，/birthday 就是一个指代所有主题为 birthday 的消息的列表。

你还可以输入一个冒号，并在后面跟上一个字符来代表你所想要看的消息的状态，这样子也能够指定一条消息。这些指示符是消息头中状态指示符的小写形式，比如说 n 表示那些在消息头中用 N 来表示的新邮件。:n 是一个包括所有新邮件的邮件列表。命令 p:n 将会显示所有的新邮件；:u 则指代所有未读的消息，同样这些消息的状态指示符为 U.

显示邮件

另外一套命令则是用来显示邮件的。单独地输入一个消息的序号，就可以显示该消息。那么消息就会一屏一屏地在屏幕上打印出来。按空格键或是回车则可以继续显示下一屏。假如你在 Mail 的提示符后输入数字 1，那么第一封信件就会被显示在屏幕上。

如果你想要连续地看若干条消息，你必须在 Mail 的提示符后输入它们的序号。其他的命令则允许你根据消息相对于当前消息的位置，来指定和显示消息。利用命令-就可以打印在当前消息之前的消息。如果你用-命令加一个数字，就可以指定一条在当前消息前若干条处的消息。假设当前消息的序号为 6，那么-4 就可以指定第二条消息。你还可以指定在当前消息之后的那些消息。用命令+加上一个数字，然后回车，就可以将当前消息之后的消息显示出来。

如果你想要显示一个范围内的消息，你必须使用命令 p 和 t。就像 Mail 中大多数其他命令一样，这些命令以一个消息的列表作为它们的参数。就像指定一条单独的消息一样，在消息列表中你可以指定一组消息或是某个范围内的消息。你可以通过一个接一个地列出消息的序号来指定一组消息，消息序号之间以空格分开，比如说 1 3 指的就是第一和第三封邮件，p 1 3 则会将第一和第三封邮件显示出来。你还可以用一个破折号来指定一个范围，比如说 1-3，指定了一、二、三这三封消息，同样地，p 1-3 则会将这三封邮件都显示出来。

如果命令 p 或 t 不带一个消息列表的参数时，系统则会显示当前的消息。你也可以用+或-和 p，t 中的一个命令连用，则可以显示前面或后面的邮件。在下面的例子中，一个不带消息序号的命令 p，将当前消息——消息一给打印出来。然后用命令 p 加上一个消息序号来显示消息二。

```
Mail version 5.5-kw 5/30/95. Type ? for help
"/var/spool/mail/chris": 3 message 3 new
>N 1 valerieTue Feb 1110:14:325/44    "Budget"
  N 2 aleina       Wed Feb 1212:30:1728/537    "Birthday"
  N 3 robert       Fri Feb 143:15:2416/293   "Homework"
& p
From valerie Wed Feb 11 10:14:17 PST 1996
To: chris
Subject:Budget
Status: R
You are way under budget
so far.
Congratulations
Val

& p 2
From aleina Wed Feb 11 10:14:17 PST 1996
To: chris
Subject: Birthday
Status: R

Yes,  I did remember your present
 Aleina

&
```

下面的这个表列举了若干个例子，是命令 p 加上各种不同组成的消息列表的功能。要记住特殊字符的功能，比如说^，$以及*分别表示第一个，最后一个和全部消息，还有用来指定某种类型邮件的:。你也可以用地址或主题来指定消息。

| 命令 | 作用 |
| --- | --- |
| p $ | 显示最后一条消息 |
| p * | 显示所有的消息 |
| p ^-3 | 显示第一至第三条消息 |
| p .-3 | 显示从当前消息到第三条消息 |
| p n | 显示下一条消息，而不显示当前消息 |
| p +2 | 显示从当前消息算起的第二条消息 |
| p /budget | 显示所有主题一栏包含有"budget"的消息 |
| p dylan | 显示所有由地址为 dylan 发来的消息 |
| p :n | 显示所有收到的新消息 |
| p :u | 显示所有还未读过的消息 |
| p :r | 显示所有你已经答复的消息 |

当一条消息显示完之后，你又进入了 Mail 的提示状态&。但这时消息头的信息并不会自动地又为你重新显示一遍。使用命令 h，你就能够在任何时候重新显示一遍消息头。在下面的这个例子中，用户输入命令 h 来重新显示消息头的列表。

```
& h
```

```
>N 1 valerieTueFeb 1110:14:325/44    "Budget"
 N 2 aleina      Wed Feb 1212:30:1728/537   "Birthday"
 N 3 robert      Fri Feb 143:15:2416/293   "Homework"
&
```

有时候，这个消息头会很长，以致于一屏并不能显示完。在这种情况下，命令 h 就只显示消息头的第一屏信息。而命令 z+和 z-就能够向前或向后翻屏。如果你知道你想看的信件的确切序号，你可以在命令 h 后加上这个序号来显示消息头。比如说，h12 就会显示第十二条消息的消息头，当然也包括在它前后的一些消息的消息头。

## 删除或恢复消息

除非你输入命令让 Mail 程序删去一条消息，否则当你离开 Mail 程序时，所有已经读过的消息都会自动保存起来。你可以用删除命令 d 来删去一条消息。输入命令 d 并跟上某条消息的序号，就可以删去这条消息。命令 d2 将会删去第二条消息。你也可以输入信件序号指定的范围，来同时删去若干条消息：d 2-4 将删去消息二、三和四。如果你输入一个不带任何消息序号的命令 d，那么 Mail 将会删去当前消息。在下面的例子中，用户删去了第三条消息。

```
Mail version 5.5-kw 5/30/95. Type ? for help
"/var/spool/mail/chris": 3 message 3 new
>N 1 valerieTueFeb 1110:14:325/44    "Budget"
 N 2 aleina      Wed Feb 12    12:30:1728/537    "Birthday"
 N 3 robert      Fri Feb 14    3:15:2416/293    "Homework"
& d 3
```

```
& h
>N 1 valerie   Tue Feb 1110:14:325/44 "Budget"
 N 2 aleina    Wed Feb 1212:30:1728/537 "Birthday"
 &
```

在离开 Mail 程序之前，如果你改变了先前的主意，想要把一些已经删掉的文件重新保存起来，你就可以使用命令 u 来做到这一点。命令 u 让你恢复一个在本次对 Mail 操作中删除的消息。删除命令其实并不是马上就把消息给删去，所有的消息都一直保存着，直到你退出 Mail 程序。同样的，你也可以指定若干消息或是一个消息的范围来同时对若干消息进行操作。命令 u 3 能够恢复消息三，而 u 2-4 则恢复消息二、三和四。比如说：

```
& h
>N 1 valerieTueFeb 1110:14:325/44    "Budget"
 N 2 aleina       Wed Feb 1212:30:1728/537   "Birthday"
& u 3
& h
>N 1 valerie   Tue Feb 1110:14:325/44 "Budget"
 N 2 aleina    Wed Feb 1212:30:1728/537 "Birthday"
 N 3 robert    Fri Feb 143:15:2416/293 "Homework"
&
```

## 答复和发送邮件新邮件：R，r，m 和 v

收到邮件的时候，你还可以发送你自己的邮件。你可以对收到的消息进行答复，或者就只是象你使用 mail 命令加上收信人地址那样，直接发送一封新的

邮件。在答复邮件时，Mail 允许你自动地使用接收到邮件中消息头部分的信息。你只要指定你想要答复的邮件序号，然后输入你要答复的内容就可以了。

使用命令 R 和 r，可以来答复收到的邮件。命令 R 后跟着一个消息序号，会自动产生一个发送邮件的消息头，并进入了可以敲如入邮件正文的输入模式。这个消息头中会包含有发信者的地址以及发信者指定的主题。但这个主题会加上一个 Re:，来表示这是一封回复某条消息的邮件。然后仅仅是输入你想要输入的内容，并在一行开头单独输入一个 CTRL-d 来结束输入。这封回信马上就被送给原信件的发信者。

```
Mail version 5.5-kw 5/30/95. Type ? for help
"/var/spool/mail/chris": 3 message 3 new
>N 1 valerieTueFeb 1110:14:325/44    "Budget"
 N 2 aleina    WedFeb 1212:30:1728/537  "Birthday"
 N 3 robert    FriFeb 143:15:2416/293 "Homework"
& R 2
To: aleina
Subject : RE: Birthday
Is it a really big present?
^D
EOT
&
```

假设把发信给你的人，还把这封信发给了其他几个用户。使用命令 r 就可以把你的回信不仅仅是发送给原信的发信者，还发给了那几个也收到这封信的用户。要特别注意命令 r 的使用。你也许并不想让你的回信发送给收到原信的所

有人。如果你只想把你的答复发送给发信者一个人，你就需要使用命令 R。

通过使用命令 m，你可以创建并发送一个新的邮件。邮件就会象你在登陆的 shell 中那样被发送出去。除非命令 m 用来取代了 mail。在下面的这个例子中，用户发送一封新邮件给 cecelia。

```
Mail version 5.5-kw 5/30/95. Type ? for help
"/var/spool/mail/chris": 3 message 3 new
>N 1 valerieTueFeb 1110:14:325/44    "Budget"
 N 2 aleina    WedFeb 1212:30:1728/537  "Birthday"
 N 3 robert    FriFeb 143:15:2416/293 "Homework"
& m cecelia
Subject: Birthday
Did you remember my present?
oops

^D
EOT
&
```

就像你在用 mail 命令发送信件时做的一样，在 Mail 的外壳程序下发送一封新邮件或者是直接答复某封邮件时，你都可以使用转义命令。转义命令~v 让你能够在 vi 中编辑你的回信，然后再回到答复信件的输入模式。一旦按下 CTRL-d 将信件发送出去，你又回到了 Mail 的提示符下。

假设在发送邮件时，你想要把你曾经收到的某封邮件内容包括进来。比如说当你给某个人写回信时，你也许想把他给你的信件内容也加进来。或者也许你想把在此之前某个人给你的信添加到给另外一个人的回信中去。使用转义命

令~m 和~f 你就能够把某封信件的内容读入你正在编辑的信件中。~m 和~f 转义命令会把一个消息的列表作为命令的参数，通常都是某个邮件序号。就好像命令~m 2 会把第二封邮件的内容读到你正在写的新信件中来。命令~m 和~f 的不同之处在于，~m 命令会对读进来的每一行信件内容进行缩排，以区别于你正在编辑的信件正文，而~f 命令则不对它进行缩排，仅仅是将这些内容象消息的正文一样添加进来。

你也可以用命令 v 来直接编辑某条你收到的信件。比如说，也许在保存某封信件之前，你想要在里面加上注解来说明你的看法评论。或者你想把这些评论直接添加到邮件上，然后使用命令~m 来把它添加到你给其他人的回信中去。为了编辑某条指定的消息，你只要在命令 v 后加上该消息的序号就可以了：v 3 命令会编辑第三条消息。

## 退出 Mail

输入命令 q 便可以退出 Mail 工具，然后返回你所登陆进来的 shell 命令行中。你读过的邮件被放在你的主目录下，一个名为 mbox 的文件中。Mail 程序会告诉你在你的 mbox 文件中保存着多少封邮件。缺省情况下，这些信件会自动从你的收信邮箱中移去。如果出于某个原因你并不想把某封邮件从邮箱中移开，你就要在离开 Mail 之前，输入一个 pre 命令，后面跟上你不想移开的信件序号。那么这封邮件就会被保存在你的 mailbox 文件中。

如果在你读某封信件之前，你退出了 Mail 程序，那么这封信就会保留在你的 mailbox 文件中，等待你下次在启动 Mail 程序时来阅读它。当你再一次进入 Mail 的 shell 时，这些上次没有看的信件也会在消息头里显示出来，并且它们

前面的状态指示符为 U，来标明这是些以前收到的旧信但并未阅读过。

```
Mail version 5.5-kw 5/30/95. Type ? for help
"/var/spool/mail/chris": 3 message 3 new
>N 1 valerieTueFeb 1110:14:325/44    "Budget"
 N 2 aleinaWedFeb 1212:30:1728/537  "Birthday"
&q
Save 1 message in mbox
$
```

你还可以用命令 x 来退出 Mail。命令 x 就好比是一个笼统的撤消操作的命令。在你本次操作 Mail 期间，任何被删去的信件都会被恢复过来。

保存和获得在 Mailbox 中的邮件：s 和 S 如果并不想把邮件保存在文件 mbox 中，那么你可以用命令 s 来明确地将信件保存到你选定的文件中。不管怎样，s 命令都会把邮件连同它的消息头部分一同存如文件中，实际上，也就是重新创建了另一个 mailbox 文件。然后就好像是在用 mail 来获得 mailbox 中的邮件一样，你可以使用 Mail 工具来获得这个新的 mailbox 文件中的信件。

要保存某封邮件，你就要在命令 s 后输入这封邮件的序号，然后再跟上你想要将它保存到的文件名。如果这个文件已经存在了，那么这封邮件就会被自动添加到文件的末尾。在下面的例子中，命令 s 2 family_msgs 将第二封邮件保存到文件 family_msgs 中。通过指定一组消息
序号或是指定一个序号范围，就能够同时将若干邮件保存到一个文件中：s 1-3 family_msgs 将信件一、二、三都保存到文件 family_msgs 中。

```
Mail version 5.5-kw 5/30/95. Type ? for help
"/var/spool/mail/chris": 3 message 3 new
```

```
>N  1 valerieTueFeb 1110:14:325/44"Budget"
 N  2 aleina    WedFeb 1212:30:1728/537"Birthday"
 N  3 robert    FriFeb 143:15:2416/293"Homework"
& s 2 family_msgs
```

你可以将邮件保存到任何你指定的文件中。在把某个发信人发来的信件组织到一个文件中时，这种功能是很有帮助的，通常会以这个发信人的名字来给此文件命名。比如说所有 robert 发来的邮件都可以保存到一个名为 robert 的文件中。用命令 S 而不用命令 s 就可以自动地为你做这样的保存工作。命令 S 加上一个消息的列表就可以将这些指定邮件保存到一个以发信人名字为文件名的文件中。如果这个文件不存在，那么命令 S 就会自动地创建一个。在下面的例子中，用户保存三封邮件到一个以发信人名为文件名的文件中，在这个例子中，文件名就为 robert.

```
Mail version 5.5-kw 5/30/95. Type ? for help
"/var/spool/mail/chris": 3 message 3 new
>N  1 valerieTueFeb 1110:14:325/44"Budget"
 N  2 aleina    WedFeb 1212:30:1728/537"Birthday"
 N  3 robert    FriFeb 143:15:2416/293"Homework"
& S 3
& q
$ ls
mbox robert
$
```

将邮件连同它的消息头部分一起保存到一个文件中就会创建一个新的

mailbox 文件，通过 Mail 程序你就可以读取和管理这个 mailbox 中的邮件。消息头部分给出了邮件序号，以便 Mail 程序拥有足够的信息来判断是哪一封邮件，显示消息头列表，删除某封信件，以及执行其他的 Mail 操作。要获得某个 mailbox 中的邮件，你可以在 mail 命令后加上 -f 选项以及该 mailbox 的文件名。如果这时你已经是在 Mail 的 shell 中，那么你也可以执行 folder 命令来切换到一个指定的 mailbox 文件。比如说命令 mail -f family_msgs 能够进入 mailbox 文件 family_msgs 中。然后，会显示一个消息列表，在 family_msgs 文件中的每一封邮件的信息都会包含在内。Mail 中的命令诸如 d 和 p 等都能够用来对这些邮件进行管理。

```
$ mail -f family_msgs
Mail version 5.5. Type ? for help
"family_msgs": 1 message
 >1 aleinawed Feb 1212:30:1728/537"Budget"
&
```

Mail 程序是为你对任何指定的 mailbox 文件进行操作而设计的。系统会把新到的邮件放在你的 incoming mailbox 中，在缺省情况下，当你启动 Mail 时，你就自动地进入了 incoming mailbox 中，并对其中的邮件进行操作。你可以切换到其他的 mailbox 文件，或者只要你喜欢就可以切换回你的 incoming mailbox。为了切换到另一个 mailbox 文件，你输入一个 folder 命令，后面跟上这个 mailbox 的文件名。于是这个 mailbox 中的邮件的消息头等扼要信息就会被显示出来，你就可以开始对这些消息进行操作。要想切换回到你的 incoming mailbox 中，你可以在 folder 命令后跟上一个 % 符号，% 代表的就是你的 incoming

mailbox。你也可以用符号#来代替 mailbox 文件名来在当前 mailbox 文件和上一个 mailbox 之间切换。#表示的是上一次操作的 mailbox 文件。

在下面的例子中，用户启动 Mail 并进入他的 incoming mailbox，然后使用 folder 命令来切换到 family_msgs 这个 mailbox 文件中。然后该用户又用命令 folder %来切换回到开始的 incoming mailbox 中。

```
$ mail
Mail version 5.5-kw 5/30/95. Type ? for help
"/var/spool/mail/chris": 2 message 2 new
>N 1 valerieTueFeb 1110:14:325/44"Budget"
 N 2 robert    FriFeb 143:15:2416/293"Homework"
& folder family_msgs
Held 2 message in  /var/spool/mail/chris
"family_msgs" : 1 message
>1 aleinaWedFeb 12 12:30:1728/537"Birthday"
&folder %
 "/var/spool/mail/chris": 2 message 2 new
>N 1 valerieTueFeb 1110:14:325/44"Budge"
 N 2 robertFriFeb 143:15:2416/293"Homework"
```

就像前面提到的那样，当你退出 Mail 程序时，你已经读过的邮件会连同它们的消息头部分保存在文件 mbox 中。如果文件 mbox 已经存在，那么新邮件就会被添加到文件的末尾。也许你想要重新获得一封阅读过、现在保存在 mbox 文件中的邮件。既然文件 mbox 也包含着消息头，它也是一个可以用 Mail 程序来进行管理操作的 mailbox 文件。你可以在上&符号来切换到 mbox 中(即 folder &)，其中符号&代表的是保存你已读信件的文件名。屏幕上会显示出一个包含

有所有你已经阅读过的邮件的消息头列表。Mail 命令诸如 p 和 d 之类就能够显示和删除这些邮件。你甚至还可以在这里用 r 或者是 R 命令来回复信件。在下面的例子中，用户启动 Mail 程序来获得已经阅读过的程序。

```
$ mail -f  mboxMail version 5.5-kw 5/30/95. Type ? for help
"/var/spool/mail/chris": 2 message 2 new
>1 marylouMonFeb 1110:14:325/44“ Trip ”
   2 dylan     FriFeb 143:15:2416/293“ Food ”
&
```

## 保存信件正文到文件中：发送和接受文件

假设你只是想保存某封信件的内容而不想连同它的消息头部分的信息。在这种情况下，你可以使用命令 w 来将一封信件的正文保存到某个文件中，而不必包括它的消息头部分。命令 w 的语法和命令 s 是一样的。w 1 newbudget 把信件一的正文保存到文件 newbudget 中。newbudget 是一个标准的文本文件，而不是一个 mailbox 文件。它并不能用 Mail 程序来获得里面的信件内容。

```
Mail version 5.5-kw 5/30/95. Type ? for help
"/var/spool/mail/chris": 3 message 3 new
>N 1 valerieTueFeb 1110:14:325/44"Budget"
 N 2 aleina      WedFeb 1212:30:1728/537"Birthday"
 N 3 robert      FriFeb 143:15:2416/293"Homework"
& w 1 newbudget
```

你可以利用这一特性来从其他用户那儿获得文件而不仅仅是接收消息。利

用重定向，任何的用户都能够通过 Mail 程序来个你传送一个文本文件。然后你就能够利用命令 w 来接收和保存这个文本文件。反过来，你也能够把某个文本文件发送给其他用户，然后该用户就同样用命令 w 来获取这个文本文件。

为了利用 Mail 来发送一个文本文件，你就必须使用重定向。Mail 程序的输入模式能够接收标准输入。反过来标准输入也能够直接从一个文本文件中获取输入。在下面的例子中，用户把文件 complist 发送给了 chris.

```
$ mail chris < complist
```

当你收到这个文本文件时，该文件上附有一个消息头。为了将这份文件不带消息头地保存下来，你可以使用 w 命令。假设文件 complist 是接收到的第一条消息，那么命令 w 1 complist 就会把这封邮件不带消息头的正文部分保存下来。

```
$ mail
Mail version 5.5-kw 5/30/95. Type ? for help
"/var/spool/mail/chris": 1 message 1 new
>N 1 aleina   Wed   Feb   12   12:30:17   5/44
& w 1 complist
```

用 Mail 程序来传送文件有一个主要的限制，那就是你只能够发送字符文件而不能发送二进制文件。在传送过程中，Mail 程序会把这个文件当作是消息头的输入而破坏了文件。不过，有其他的一些专门的程序用来传送二进制文件，比如 ftp 等。在第 11 章中会对二进制文件的传输做详细的探讨。这些程序在传送非常大的文件以及二进制文件时要比 Mail 程序要可信赖得多。

### 8.4.4Mail 别名，选项以及 Mail Shell 初始配置文件：.mailrc

Mail 程序有着它自己的初始配置文件.mailrc，每次发送或是接收消息启动 Mail 程序时，它都会自动执行。在这个文件里，你可以定义 Mail 的选项和为 Mail 建立各种别名。你可以设立选项来给 Mail 增加新的特性，比如说，你可以更改提示或是保存你所发出信件的拷贝。使用 Mail 的别名功能，你就能够很容易地把信件同时发送给几个用户。你会发现在把不同的消息散发给同一组用户时，Mail 的这个特性是很有用的。

#### Mail 别名

你也许经常需要发送或是散发消息给同一组用户。举例说来，假设你是一个研究小组的成员。你想要给这个研究小组中每一个成员发送同一条消息，比如说你们下次开会的时间。通常，每次你给这一组成员发送消息时，你都不得不重新输入一遍所有人的地址。然而，你若是在初始配置文件中为这一组用户的地址起一个别名，那么以后当你要给他们同时散发某个文件时，就不需要再把他们的地址一一列出，而只是写上你定义的别名就可以了。这个别名可以代替在命令行上的地址列表。

为了定义一个别名，你可以输入关键字 alias，然后输入你所选的别名，再跟上这个别名所代表的地址列表。这里有一个解决方法是：为地址列表定义的别名，要在 Mail 命令行中使用的话，就必须在 Mail Shell 中对这个别名进行定义。每次当你离开了 Mail 再重新进入 Mail Shell 的时候，你都必须重新定义一遍这个别名。但如果你是在 Mail Shell 的初始配置文件.mailrc 中放入这个别名的定义，这个定义的工作可以在你进入 Mail Shell 时自动地进行。为了在.mailrc

文件中加入别名，首先要用一个文本编辑器诸如 vi 等来编辑.mailrc 文件。然后输入 alias 命令，然后跟上别名以及别名代表的地址列表。要确保在地址列表中没有一换行符来分开这些地址。下面的例子在.malric 文件中定义了别名 myclass..mailrc

```
alias myclass chris dylan aleina justin larisa
```

当 Mail 启动时，.mailrc 文件就会自动地执行，定义别名代表的地址。当你使用 Mail 来发送一封邮件时，你可以用别名来取代命令行上的地址列表。你也可以在 Mail 的 Shell 内发送信件时使用别名。这两种情况下，文件.mailrc 都会被执行，为地址定义别名。在下面的这个例子中，别名 myclass 取代了地址列表。文件 homework 的内容会发送到那些地址包含在 myclass 别名定义中的用户那里。

```
$ mail myclass < homework
```

Mail 选项

Mail 还提供了一组选项，你可以在文件.mailrc 中对它们进行定义，那么在你每次使用 Mail 时它们都会是有效的。比如说，你可以把 Mail 提示符定义成其他更多的一些字符而不是&。

表 8-4 列举了几种比较常见的 Mail 选项。在关键字 set 后跟上选项名，如果需要的话还要加上一个合适的符号和字符串，你就可以设置一个 Mail 的选项。譬如，set prompt="*"会把 Mail 的提示符设置成一个星号"*"。一旦文件.mailrc 被执行，那么它就会对在文件中指定的选项进行设置。

sign 是一个非常有用的的选项，它能够指定一个签名档，当你使用转义命令~a 时，就可以把这个签名添加到你的邮件中去。通常 sign 选项都被设置为你的名字或者是加入一些其他信息，比如说你的电话号码、你的 ip 地址等等。

下面的这个例子把签名设置为“ Robert and Valerie ”

```
set sign=“ Robert and Valerie ”
```

你可以使用转义命令~a来将你的签名添加到任何一封邮件上去。

```
$ mail aleina
Subject : Dinner
Lets have ice cream for dessert
OK?
~a
Robert and Valeire
^D
EOT
$
```

另一个非常有用的选项则是 record。这个选项会引导 Mail 来自动保存你所创建和发送的所有消息。在设置 record 选项的时候，你需要指定 Mail 应该把你要发送的邮件保存到哪一个文件中。在下面的例子中，用户设置了 record 选项，并且把已经发送出去的邮件保存在一个叫做 outbox 的文件中。如果没有指定一个绝对路径名，那么这个文件就会放在你的主目录下。

```
set record=“ outbox ”
```

## 组织你的 Mailbox 文件：folder，MBOX 以及 outfolder

你会注意到那些你用 s 或 S 命令创建的 mailbox 文件，以及你的 mbox 文件会散布在你系统的各个不同目录中。你用命令 s 创建的 mailbox 文件会被放

置在你当前的工作目录中，无论你启动 Mail 程序时处于的是什么目录。而你用命令 S 创建的 mailbox 文件，还有 mbox 文件则放在你的主目录下。事实上，你可以指定一个专门的目录，来把你所有的 mailbox 文件都放在里面，而不必是你的主目录或是你的工作目录。要达到这个目的，你就必须使用到以下这三个专门的选项：folder，MBOX，和 outfolder。

首先你要建立一个新的目录来存放你的 mailbox 文件。然后在你的.mailrc 文件中设置 folder 选项，把该目录的路径名字赋给它。通常用 folder 目录来代表这个目录。folder 选项共有两个作用。一是将所有你用 S 命令创建的 mailbox 文件保存到赋值给 folder 选项的目录中，二是激活了+这个符号，使它在 Mail 程序中成为一个专用字符，代表 folder 这个目录。当使用命令 s 时，你可以在 mailbox 文件名之前加上一个+号，来把这个 mailbox 文件保存到 folder 指定的目录中，和其他的 mailbox 文件放在一起。当你使用 folder 命令来切换到另一个 mailbox 文件的时候，你可以在这个 mailbox 文件名之前加上一个符号+，Mail 才会在 folder 目录中寻找该文件。

在下面的例子中，用户已经创建了一个叫做/home/chris/mail 的目录，并且把这个路径名赋给了 folder 变量。现在用户使用命令 S 创建的任何 mailbox 文件都会被放在这个目录中。你可以任意地给这个目录起名字，但在本例子中，它就只是简单地叫 mail.

set folder=“ /home/chris/mail”

在接下来的例子中，用户把消息二保存到一个名为 family_msgs 的文件中。文件名的前面有一个代表 folder 选项指定目录的+号。给定 folder 选项的设置如上面的例子所述，文件名+family_msgs 会把 family_msgs 这个 mailbox 文件

保存到目录/home/chris/mail 下，和其他的 mailbox 文件放在一起。然后该用户切换到 folder 目录下的其他 mailbox 文件中，注意一定要在 mailbox 文件名前 加上+号，以便让 Mail 在 folder 目录而不是在当前工作目录中寻找它。

```
Mail version 5.5-kw 5/30/95. Type ? for help
"/var/spool/mail/chris": 3 message 3 new
>N 1 valerieTueFeb 1110:14:325/44"Budget"
  N 2 aleina    WedFeb 1212:30:1728/537"Birthday"
  N 3 robert    FriFeb 143:15:2416/293"Homework"
& s 2 +family_msgs
"/usr/mail/chris/mail/family_msgs" [Appended]
& folder +family_msgs
Held 2 messages in /usr/mail/chris
"+family_msgs": 1 message 1 new
>1 aleinaWed Feb 1212:30:1728/537"Birthday"
```

然而，你的 mbox 文件无论如何都不会被自动地放入这个 folder 目录中。mbox 文件的名字信息被保存在一个叫做 MBOX 的专门的变量中。你可以在 mbox 的名字前加上一个符号+，然后赋给变量 MBOX，那么你就能够把 mbox 文件放到你的 folder 目录下。就像刚才设置 folder 选项时那样，你同样地在你的.mailrc 文件中给 MBOX 赋值就可以了。

```
set MBOX = +mbox
```

你还可以指定那些用来保存你发送邮件的特定文件，来放到你的 folder 目录里。但你必须要先对你的 record 选项进行定义，把你想要用来保存发送邮件的文件名字赋给它。然后再把这个文件放到你的 folder 目录下，你可以象这里

示范的那样来设置 outfolder 选项。

```
set outfolder
```

你也可以仅仅是给 record 选项赋值，在那个保存发送邮件的文件名前加上一个加号“+”。

```
set record=+outbox
```

当你设置完这三个选项之后，你所有的 mailbox 文件都会被保存在一个特定的目录下。所有的这些设定都应该放在你的.mailrc 文件中。下面是一个.mailrc 文件的样本，包含有 Mail 别名和许多其他的变量或选项的赋值。

```
.mailrc
alias myclass chris dylan aleina justin larisa
set sign="Robet and Valerie"
set folder="/home/chris/mail"
set MBOX=+mbox
set record=outbox
set outfolder
set prompt="*"
```

## 8.5Elm 工具

另一个流行的邮件工具则是有 Dave Taylor 开发的 Elm。虽然它并不是一个正式的标准邮件工具，却取得了越来越广泛的应用。Elm 并不像 Mail 那样使

用命令行，而是采取了全屏的工作方式，用户友好界面使得处理邮件的任务变得十分容易执行。消息每次都会显示一整屏，而且你可以不断翻屏来在消息的前后移动。你不再需要在命令行中输入命令，取而代之的是你键盘上的键，这些键作为单字符命令来对邮件进行操作，就好像你是在使用单字符的 vi 命令来进行编辑操作那样。Elm 使用的命令中，有很多与其他工具使用的命令一致，比如说 more 和 vi。表 8-5 列举了和其他工具不一样的一些 Elm 命令。

### 8.5.1 用 Elm 发送邮件

在 Elm 中，通过它的命令，你可以把信件发送到任何一位用户手中。信件的内容可以是你从键盘上输入的正文或者是一个文件。当你从键盘上输入信件内容时，你就像是在使用一个例如 vi 之类的标准编辑器。

为了发送一封邮件，你只要敲入 elm 然后跟上你要将信件发送给的用户地址。当你按了回车之后，Elm 就会显示把这个用户名字显示出来，然后提示你输入主题。注意这里 Elm 显示的是这个用户的确切名字而不是用户的地址。它会从在线信息上获得用户的姓名，就和使用 finger 命令来获得用户姓名是一样的。

```
$ elm robert
Send only mode  [ELM 2.4 PL20]
To: Robert Petersen
Subject of message : Birthday
Copies to:

Invoking editor...
```

在主题提示栏，你可以输入给这封信输入一个主题。然后 Elm 就会提示你输入一个信件拷贝的发送清单。如果你想要把这封信件的拷贝发送给其他的用户，那么你就在这里输入他们的地址列表，如果你并不想这么做，那么你只需要按回车，跳过这一部分就行了。这时候一旦你按了回车键，那么你就进入了一个标准编辑器，通常而言都是 vi 编辑器，那么你就可以开始用这个编辑器去输入你的信件内容。一定要记住，你使用的是 vi 编辑器，如果你真的要输入邮件内容，那么你必须先按 a 或是 i 命令来进入 vi 的输入模式。当你输入信件后，你可以按 Esc 键来返回到 vi 的命令模式。最后当你完成了信件编辑工作时，用命令 zz 就能够保存信件文本并退出 vi 编辑器。

```
your present is in the mail
really
~
~
~
~
~
~
```

在完成上述步骤之后，Elm 会提示你输入一个动作命令，这时候你可以发送、不发送而直接退出、重新编辑这封信件，或是编辑它的消息头。每一个功能都只是通过一个单字符命令来实现的。你所要做的只是简单地按一下键盘，就好像你要按的是一个 vi 编辑器的命令那样，并没有什么命令行来让你在里面输入这个字母并按回车。

```
Please choose one of the following options by parenthesized
```

letter:

e）dit message， edit h）eaders， s）end it， or f）orget it

当你运行 Elm 程序时，你可以在命令行使用-s 选项来指定一个主题。在下面的例子中，主题“ Tonight's celebration ”就是在命令行中进行指定的。这样的话，Elm 将不再提示用户输入一个主题。

$ elm george - s " Tonight\'s celebration"

而在 Elm 消息菜单中的 h 选项则是用来编辑消息头的。使用这个选项你就可以改变你的消息头中任何一部分，或是给这封信的消息头加上其他的一些信息，比如一个抄本传送的地址列表。当你输入一个 h 的时候，一个编辑消息头的屏幕就会被显示出来，为消息头的每一栏都列举了提示。要想更改或是添加这些栏的值，只需要按一下消息头提示部分的第一个字母就够了。在屏幕的底端会有提示告诉你每一栏的名字，你就可以根据这些提示来输入一个新的内容。按下回车键，这些新内容就会在消息头部分显示出来。

比如说，倘若要修改主题，就可以先按一个 s，然后系统就会提示你输入一个新的主题。在输入了这个新主题并按回车之后，主题栏的新内容就会在消息头中显示出来。你还可以同样地编辑消息头的其他栏内容。如果要添加一个抄件传送的地址清单，那么就要按 b 键，然后跟着输入要想传送的地址并回车。一旦按了回车键，你就离开了消息头编辑屏幕，这时候你看到的并不是消息头提示部分的首字母，而是回到了开始的消息菜单的界面下。图 8-1 显示了消息头编辑屏幕的一个样本。

Message Header Edit Screen

T）o: robert （Robert Petersen）

C）c:

B） cc:
S） ubject Birthday

R） eply to:
A） ction:E)xpires：
P） riority:Precede(n)ce:
I） n-reply to:

Choose header， u） ser defined header， d） omainize
!） shell， or {enter}

Choise:

**图** 8-1Elm **的消息编辑屏幕**

Mailbox is '/var/spool/mail/chris's with 3 messages
N1 Gabriel Matoza Feb 11(5)"Budget"
N2 Aleina Petersen Feb 12 （28） "Birthday"
N3 Marylou Carrion Feb 14 （16） "Homework"

You can use any of the following commands by pressing the first character:d） elete or u） ndelete mail， m） ail a message， r） eply or f） orward mail， q） uit To read a message， press {enter}. j=move down， k=move up， ?=help

Command:

图 8-2Elm 的消息头

### 8.5.2 使用 Elm 来接受邮件

如果想要用 Elm 来接受邮件，你只要和用 Elm 来发送邮件时做一样的工作——在命令行上输入单独的一个 elm 就够了。于是 Elm 程序就会显示一个消息头的列表，在这些消息头中对你已收到邮件的信息进行描述。消息头是从屏幕的顶端开始列写的，在屏幕的底端则是一个信息菜单，里面列举了在这个消息头屏幕中你可以执行的操作命令。在 Elm 中用 index 来指代这个消息头列表。这些消息头的信息显示在屏幕上就好像是一篇用 vi 编辑器编辑的文本。如果邮件的消息头部分内容多于一屏，你可以用+键来翻到下一屏。当然你也可以用-键来翻回到上一屏。

Elm 的消息头看起来和 Mail 中是一样的。一个 Elm 的消息头中包括有：状态指示符、邮件序号、日期、发信者姓名、信件的行数以及主题。和 Mail 一样，Elm 用一个字符来表示信件的状态。N 表示这是一封新收到的信件，O 则表示是一封仍未读过的旧邮件。而邮件序号则是用来在 Elm 命令中指代对应的邮件。图 8-2 显示了一个 Elm 消息头的样本。里面的邮件都是新收到的。第一封信件是二月 11 日 Gabriel Matoza 寄来的，该邮件共有 5 行，主题为 Budget。注意到在消息头中显示出来的是发信者的全名而不是发信者的地址。

当前邮件的前面会有一个箭头->来指示，或者是背景被置为高亮度。如果要对某封邮件执行特定的操作，你就必须使这封待操作的邮件成为当前邮件。

你可以通过好几种方法来达到这个目的：或者是输入这封邮件的序号然后回车；或者是通过使用移动命令 j 和 k，来在屏幕上直接移动当前邮件的指示符，其中，j 命令是将表示当前邮件的->指示符移动到上一封邮件，k 命令则使->移到下一封邮件。你还可以使用上下两个方向键来达到同样的目的。当你按 k 或 j 命令时，那些表示当前邮件位置的->或是高亮度条会随着你的命令而移动到上一封或下一封邮件。如果当前邮件是第一封邮件的话，那么按 j 命令两次则最后一封信件会成为当前邮件。

若想要显示当前邮件的内容，你只要简单地按一下回车就可以了。如果是要显示其他的信件，你则可以采取按 k、j 命令来使之成为新的当前邮件或是输入该信件的序号方法，然后回车。一个显示有信件内容的新屏幕会出现在你的面前。如果该信件内容长于一屏，你可以用和 more 程序一样的命令来不断翻屏。敲击空格键你就会进入下一屏，而按 b 则回到上一屏。你甚至能够在这封信件中搜寻某个指定的模式。

Message 1/3 From Aleina PetersenFeb 11，9604:13:56 am -0300

Subject: Birthday
To: robert （Robert Peterson）
Date: Mon，12 Feb 1996 10:14:17 -0700 （PDT）

Yes， I did remember your present

Aleina

Command ('i' to return to index) :

当你检查完你的信件后，就可以使用 i 命令来返回到显示消息头的界面下。也就是说只要简单地按一下 i 键就可以了，i 代表的就是 Elm 用来指代消息头列表的索引（index）。按 p 键则可以打印邮件。如果是单独的一个 p 那么打印的就是当前消息头所指向的信件内容，而若是在 p 后面跟上一个信件序号则可以打印一个指定的邮件。

如果你想同时对若干信件进行操作，那么你首先要使用命令 t 来标记它们，然后再输入一个操作命令，则此命令会对这几封标记了的信件同时起作用。标记邮件只要将当前邮件指示符移到该邮件上并按 t 键就可以了，然后你就会看到一个+符号出现在该消息头之前。假设你要同时打印若干邮件，你就可以使用命令 t 来对这些想要打印的邮件的消息头做标记，然后按 p 键来打印它们。在下面的这个例子中，用户给前两封邮件设了标记。如果他接着按了 p 键，那么消息一及消息二的内容都会被打印出来。

```
+N 1 Gabriel MatozaFeb 11 (5) "Budget"
+N 2 Aleina Petersen    Feb 12 (28)    "Birthday"
 N 3 Marylou Carrion    Feb 14 (16)    "Homework"
```

此外 Elm 还允许你通过模式来选择一个当前消息头。Elm 中有若干条这样的命令，能够根据指定的模式来搜寻邮件的不同部分。命令/能够在一个消息头的地址和主题栏中寻找指定的模式，而双斜杠命令//则能够在邮件的正文部分搜寻指定模式。如果你输入了一个/命令，那么 Elm 将提示你输入一个要寻找的模式。

一旦你按了回车键，那么 Elm 就开始在每一个消息头的地址与主题栏中寻找你输入的这个模式。然后它会在找到的第一个与指定模式相匹配的消息头处停下来，并且使它成为当前消息。以前面所举的例子为例，如果你搜寻一个包含有模式“ birth ”的消息头，那么你就能够使第二条消息成为当前消息头。

命令//能够在你的信件正文中搜寻所指定的模式。输入一个//命令，然后 Elm 就会提示你输入一个模式。Elm 将会在你所有信件的正文中寻找，并在第一个正文中含有相匹配模式的消息头处停下来，同时使这条消息成为当前消息。比如说，如果用户输入了一个//以及模式 congratulations，然后 Elm 在信件正文中寻找，在第一条消息正文中得到一个 congratulations 的匹配，那么第一条消息也就成为了当前消息

->N 1Gabriel MatozaFeb 11（5） "Budget"
N 2Aleina Petersen Feb 12（28） "Birthday"
N 3Marylou Carrion Feb 14（16） "Homework"

其他的寻找命令则执行一些专门的操作，比如命令 CTRL-t 能够根据某个模式来对消息头做标记，或者是象命令 CTRL-d 那样对包含有某个模式的消息进行删除操作。你可以用命令 CTRL-t 来标记来自某个用户的所有信件或者是关于某个题目的所有信件。然后你就可以对这些标记好的信件进行诸如打印等操作。举例说来，假设你要打印所有关于“ birthday ”这一主题的信件，首先要做的是输入 CTRL-t 以及模式 birthday，这样就能够把所有关于生日的信件都做上标记，然后再按 p 键，于是这些标记了的邮件就都会被打印出来。

### 8.5.3 退出 Elm 程序

按 q 键，你就可以退出了 Elm 程序。在你离开 Elm 之前，Elm 会问你是否想把你所读的邮件保存到你的 received mailbox 文件中。received mailbox 文件的名字包含在一个叫做 received 的 Elm 变量中。此外 Elm 还会问你是否要将未读的邮件保存到你的收件箱中，以便于你下一次再阅读它们。如果选择了不保存，那么这些未读过的信件就会被删除掉。当然在你这次使用 Elm 期间删除过信件，那么 Elm 也会让你再一次确认是否真的删除它们。在下面的例子中，用户确定要删除信件，把已读过的邮件保存到 received mailbox 文件中，并且将未阅读的邮件保留下来以供下一次启动 Elm 时阅读。

```
Dlee message? (y/n)      n
Move read messages to "received" folder? (y/n)  y
Keep unread messages in incoming mailbox? (y/n)  y
Keeping 2 messages and storing 1
```

另外还有几个命令可以用来退出 Elm。命令 Q 会直接退出 Elm 而不给出任何关于移动已读信件或是保留未读邮件的提示。根据事先设置好的缺省情况来决定是否保存信件。命令 x 和 CTRL-q 命令也能够退出 Elm 程序，并使收件箱完好如初，所有执行的删除操作或是阅读都被忽略掉。

### 8.5.4 删除和反删除信件:d 和 u

如果要删掉当前消息，只要简单地按一下 d 键就够了。这封信件前面的状态指示符就会变成一个 D，并且在你离开 Elm 时删除掉这封信件。如果要删除

的信件并不是当前信件，那么你就要使用 j 或是 k 键来将当前消息标志移动到这封信件前，或者是输入这封信件的序号，然后再按 d 键。要删除第五封信件就只要先按 5 然后按 d。如果想要同时删除若干封信，则要先用 t 来标记它们，然后再按 d 命令。在下面的例子中，用户已经删除了第二封信件，于是第二封信的状态指示符变成了一个 D。

```
  N 1Gabriel MatozaFeb 11 (5) "Budget"
->D 2 Aleina Petersen  Feb 12 (28)  "Birthday"
  N 3 Marylou Carrion  Feb 14 (16)  "Homework"
```

你也可以用命令 CTRL-d 来选择一组在地址或标题栏中包含有某个指定模式的消息，然后再删除。利用这个特性，你就可以删除来自某个用户的所有信件或是关于某个主题的所有信件。在下面的例子中，用户决定删除所有来自 Aleina 的信件，于是他按 CTRL-d，并在输入一个模式的提示后输入“ Aleina”。第二封和第四封邮件的地址栏和此模式相匹配，因此 Elm 就为它们做上删除标记。

```
  N 1 Gabriel MatozaFeb 11 (5) "Budget"
  D 2 Aleina Petersen  Feb 12 (28)  "Birthday"
  N 3 Marylou Carrion  Feb 14 (16)  "Homework"
->D 4 Aleina Petersen  Feb 16 (32)  "party"
```

要恢复一个消息，移动到它的头部或者输入它的消息号，然后按下 u 键。该消息的状态就由 D 变成 U，表明一个恢复操作。相应于用 u 键来恢复删除的消息或者是加标签的消息，CTRL-u 命令将恢复删除的在它们的地址或者主题

区域中带有某个模式的所有消息。CTRL-u 和 CTRL-d 使用方法一样。

### 8.5.5 回复消息

在 Elm 使用工具中，你可以给你收到的任何一个消息发一个回复。Elm 用一个消息的头信息来判定发送者的地址。r 命令发送一个消息的回复。你通过首先把一个消息头变成当前的消息头或者通过输入该消息的序号，然后按 r 键来回复一条消息。要使用 r 键，Elm 首先询问你是否在你的回复中包含该消息的一个拷贝。你回答 y 或者 n。Elm 然后打开一个屏幕把发送者的名字和主题显示在顶部。光标处于主题位置，如果你愿意可以让你改变主题。按回车键继续。然后你被提示你是否希望发送其他副本。通过输入回车键，你就可以进入一个象 Vi 这样的标准编辑器，在里面你可以输入你想回复的文字。就像和一个普通的消息一样，要输入文字，首先你要通过一个 a 命令进入输入方式。你用 ESC 退出输入模式。然后是 zz 命令保存文本返回到 Elm 中。发送消息的菜单就被显示出来。然后你就可以按 s 键来发送这条消息。

### 8.5.6 从 Elm 中发送消息

当你在 Elm 实用工具中你可以用 m 命令来发送消息。通过按下 m 键，你就被提示输入你想发给消息的用户的地址。你输入地址然后就被提示输入主题。在主题之后，你被询问你是否想发送其他副本。如果不想，按下回车键。然后你就在一个 Vi 编辑器中，你可以编辑和输入你的消息。在推出编辑器之后，你

可以用 s 命令来发送这条消息。

### 8.5.7 在 Elm 中保存消息

你可以用 s 命令把一条消息保存到一个邮箱中。或者是把当前的头变成你想保存的消息头，或者是输入消息的序号，然后按下 s 键。在屏幕的底部，Elm 给你提供保存提示一个缺省的邮箱的文件名称。这个名称就是发送者的消息的名称。Elm 假设你也许希望根据谁发送的消息把它们保存到不同的信箱文件中来组织消息。要在发送者的名字显示在邮箱上提示保存的时候保存你的消息，就按下回车键。反之，如果你想在提示保存的时候用自己选的名字来保存该消息，输入一个=符号紧跟着是文件的名字。例如，要把你的消息保存到文件 birthdays，输入=birthdays。

如果你想把几个消息保存到同一个信箱文件中，你首先给该消息头加上标签，再按下 s 键。你接下来指明的邮箱文件就会把所有加标签的消息保存在里面。

Elm 还提供两个特殊的邮箱文件，它们的名字在 Elm 变量中引用为 received 和 sent。received 邮箱文件和邮件中的一个 mbox 文件类似。你读过的邮件在你退出 Elm 的时候会自动保存在这个文件中。sent 文件通常是用来放你已经发送的消息。你可以通过在提示保存的时候输入一个>来显式的把你的消息存放在 received 邮箱文件中。一个<将把消息保存到 sent 邮箱中。

### 8.5.8 用 Elm 来读取邮箱文件

在 Elm 中，你可以从阅读收到的消息切换到读取一个特定的邮箱文件。c 命令将提示你一个邮箱文件的名称。只要按下 c 键提示就出现了。提示将显示用来保存当前的消息头的发送者所发送的消息的邮箱文件的名称。你可以通过在一个=符号之后输入文件的名字来打开一个你自己的邮箱文件。=birthdays 将指明 birthday 邮箱文件。在下一个例子中，当前的头是一个叫做 aleina 的用户发送的消息。当用户按下 c 命令的时候，“改变文件夹”的提示就显示在命令提示上，作为缺省，发送这个消息的用户也显示在“改变到哪个文件夹”提示中。在这个例子中，用户的名字是 aleina。

```
Command: Change folder
Change to which folder:=aleina
The user then enters the new folder name at the prompt,
overwriting =aleina:

Command: Change folder
Change to which folder: =birthdays
```

一旦你改变到另一个文件夹，在这个邮箱文件中所有消息的消息头都将被显示，你就可以显示这些消息，把它们从文件中删除，或者是发送一个回复。文件夹的名字将在屏幕上方被显示。

你可以在文件夹提示中通过输入一个名字来打开其他任何一个文件夹。输

入一个!将改回到到来文件箱。按下>将改到 received 文件夹，<将回到 sent 文件夹。

### 8.5.9Elm 别名：别名菜单和 aliases.text

你可以在使用工具中的 Elm 别名菜单中，或者是在一个叫作 aliases.text 的特殊 Elm 初始化文件中输入别名来创建一个别名。当然别名菜单是管理 Elm 别名最简单和有效的方法。但是，如果你想加入一组别名，象 Mail 用的那样，你就需要知道怎样在 aliases.texe 文件中直接输入一个别名。

### Elm 别名：别名菜单

要在 Elm 使用工具中创建一个别名，你使用 a 命令，它将显示别名菜单。你可以在菜单中输入命令来创建一个别名，删除一个别名，或者是列出所有的别名或者特定的别名。要创建一个新的别名，你通过按下 n 键来选择新别名选项。然后 Elm 就提示你输入别名，用户名，和用户的地址。然后它自动把这个别名安装和添加到 aliases.text 文件中。

Aliases [ELM 2.4 PL20]

You can use any of the following commands by pressing the first character;

a）lias current message， n）ew alias， d）elete or u）ndelete an alias，m）ail ro alias， or r）eturn to main memu. To view an alias press <enter>.

j = move down， k = move up， ? = help

Alias:n
Add a new alias to database..
Enter alias name:mark

下面的例子将一个接一个的显示在一行上。在输入该别名的名称之后，那个提示和名称将被一个要你输入你要做别名的那个人的姓氏的提示代替。提示你输入姓氏的提示又被输入姓的提示的符号代替,然后继续,被地址提示符代替。最后，你被询问你是否接受一个新的提示符号。你可以输入 y 或者 n.

Enter last name for mark: Petersen
Enter first name for mark: Mark
Enter optional comment for mark:
Enter address for mark: mark@ violet.berkeley.edu
Messages addressed as: mark@violet.berkeley.edu (mark petersen)
New alias: mark is 'Mark Petersen'. Accept new alias? (y/n) y

你也可以为当前消息头所指明的消息发送人创建一个别名。在这种情况下，按动 Elm 别名菜单里的 a 命令。Elm 将指示你填入一个别名，但将从消息头得到用户的名字和地址。别名将自动安置并被加到 aliases.text。以上可能是创建别名最简单的方法了。

通过按下 d 键，你可以删除一个别名。Elm 提示你输入别名，然后将之从 aliases.text 文件中删掉。你也可以用 l 命令列出别名清单，或用 p 命令列出包

括名字和地址在内的详细别名清单。

### Elm 中的别名: .elm 和 aliases.text

Elm 在你的根目录下保留着一个名为.elm 的目录，通过它可以对你对 Elm 的使用进行设置。每次你激活 Elm，都将为你的应用生成一个 shell，在其中你可以定义自己的别名和设置变量。.elm 目录里有一些特别的初始化文件，你可以将别名和设置变量放在这些文件里。这些初始化文件操作起来就像.mailrc 文件，象用.mailrc 设置 Mail 一样，可以用它们设置你的 Elm 功能。

你可以在.elm 目录下的 aliases.text 文件中定义 Elm 别名。Elm 别名的命名规则是：别名，一个等号，用户名，另一个等号，用户地址。

alias = user name = user address

在下一个例子中，用户为 Gabriel Matoza 创建了一个别名，地址为地址为 gabriel@garnet.berkeley.edu 的 gabriel@ garnet.berkeley.edu。

Sgabe = Gabriel Matoza = gabriel@ garnet.berkeley.edu

你可以让一个人拥有许多别名。只需列出它们，用冒号将两两开。在下一个例子中，用户为 robert@violet.fallon.edu 的 Petersen 创建了两个别名。

robert, bob = Robert Petersen = robert@violet.fallon.edu

你每创建一个独立的的别名，都可以用它们来创建别名组。一个别名组和在 Mail 的.mailrc 文件中使用的别名的操作方式相同。当你向一个别名组发送一个消息，它将被送到组中每一个用户那里。你可以用下面的方法定义别名组。alias-list 中是已定义的被逗号隔开的独立用户别名。

group alias = group name = alias-list

在下一个例子中，用户使用从前定义的“gabe”和“robert”创建了一个别名组。别名组称作myclass，组名是“photography class”。如果用户向myclass发送一个消息，它将送往robert@violet.fallon.edu和gabriel@garnet.berkeley.edu。

myclass = photography class =robert， gabe

下面的例子是alias.text文件。注意先定义独立的别名，下面的别名组使用了这些别名。

alias.text

gabe = gabriel matoza = valerie@garnet.berkeley.edu

robert，bob = robert petersen = robert@violet.edgene.edu

myclass = photography class = robert， val

每当你在alias.text文件中加入了新别名，都必须通过执行newalias命令来将它们安置到Elm系统中。

### 8.5.10Elm 选项：.elmrc

Elm有许多不同的选项来设置你的Elmshell。这些选项是些变量，可以开关或赋值。一些Elm选项是很简单的开关量，你可以使用特定值YES和NO来打开或关掉它。例如，语句alwaysstore=YES就是将选项设置为随时把接到的邮件存储到received信箱文件中。其它的Elm选项使用字符串来设置。例如，语句receivedmail=mybox将选项receivedmail设为“mybox”。这个选项装有乘放接受邮件的文件的文件名。在这个例子中，收到的邮件将被存入mybox

文件中。

通过在你的.elm 目录下的.elmrc 文件输入语句，或者使用 Elm 的选项菜单，你可以对选项进行设置。在 Elm 中，命令 o 激活选项菜单。选项菜单只显示比较常用的一些选项。当改变一个选项时，按下该选项的第一个字母，然后输入一个新值。

## 8.6Pine

Pine 是一个邮件程序,同时有新闻浏览功能。Pine 与 Internet 新闻组和 Email 程序相兼容。使用 Pine 的 email 功能，你可以轻松地发送消息、文本和图片。它有一个丰富的选项序列，有兼容性好的 Internet 连接性能，使你既能接收信件接收 Usenet 新闻。Pine 也为你提供了一个地址手册，可以把常用的 Email 地址放在里面。Pine 从命令行启动，界面简捷，并使用光标。输入命令 pine 即可启动 Pine。

Pine 支持全屏幕光标控制。图 18-3 显示了它所使用的菜单，通过方向键可以移动光标指向不同的条目，回车键确认。每一个条目用一个大写字母标出；可以通过输入这个字母来选定条目。命令 O 会列出你可能使用的其它 Pine 命令。

要想送出消息，选择 Compose Message 条目。这会激活一个屏幕，你可以输入你的消息。首先，提示你填入消息头的各个条目，包括 email 地址和标题。你还可以附上文件一起传送。

接着你输入消息的正文。列在屏幕底部的一系列命令可以完成不同的任务。你可以用 CTRL-R 浏览文件，用 CTRL-C 退出。使用 CTRL-X 来发出消息，如图 18-4 所示。

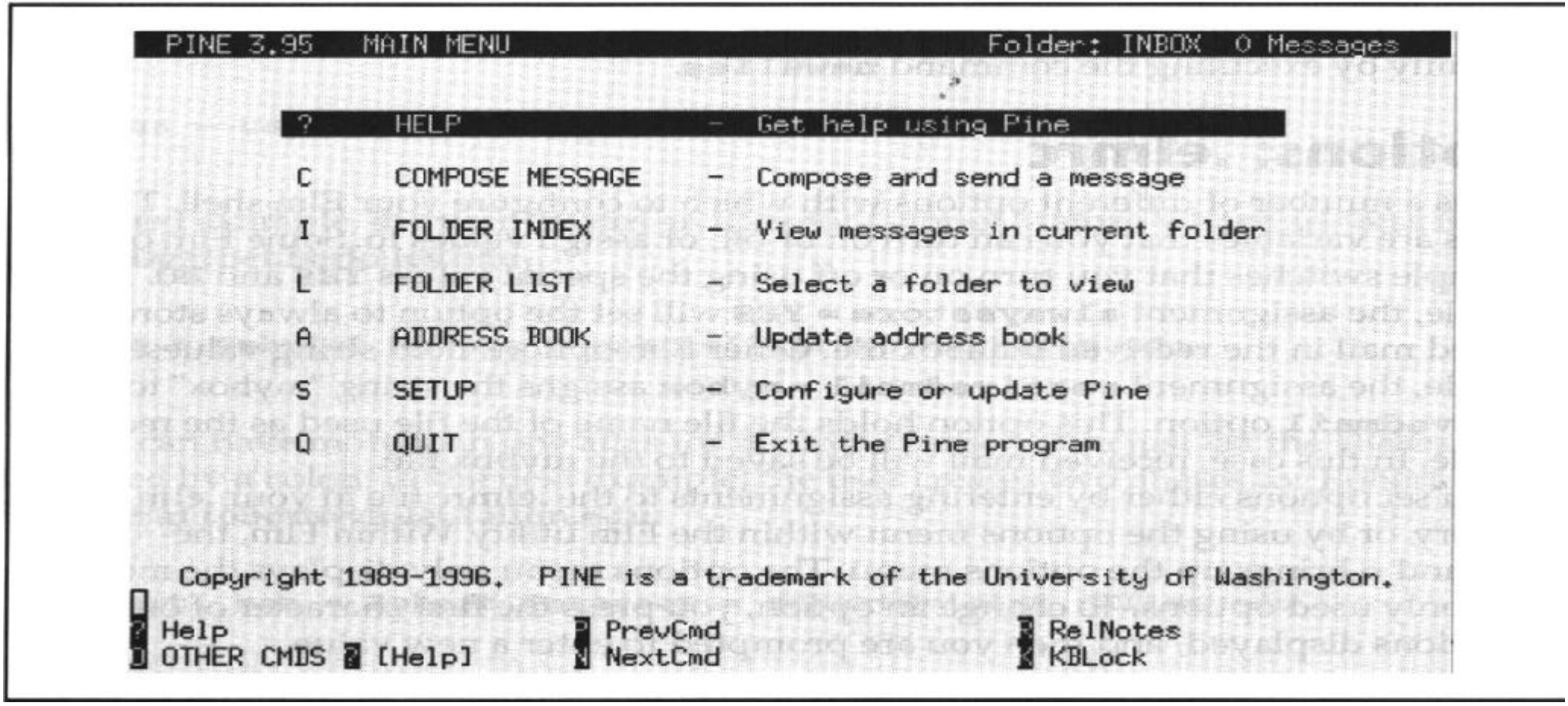


图 8-3Pine 主菜单

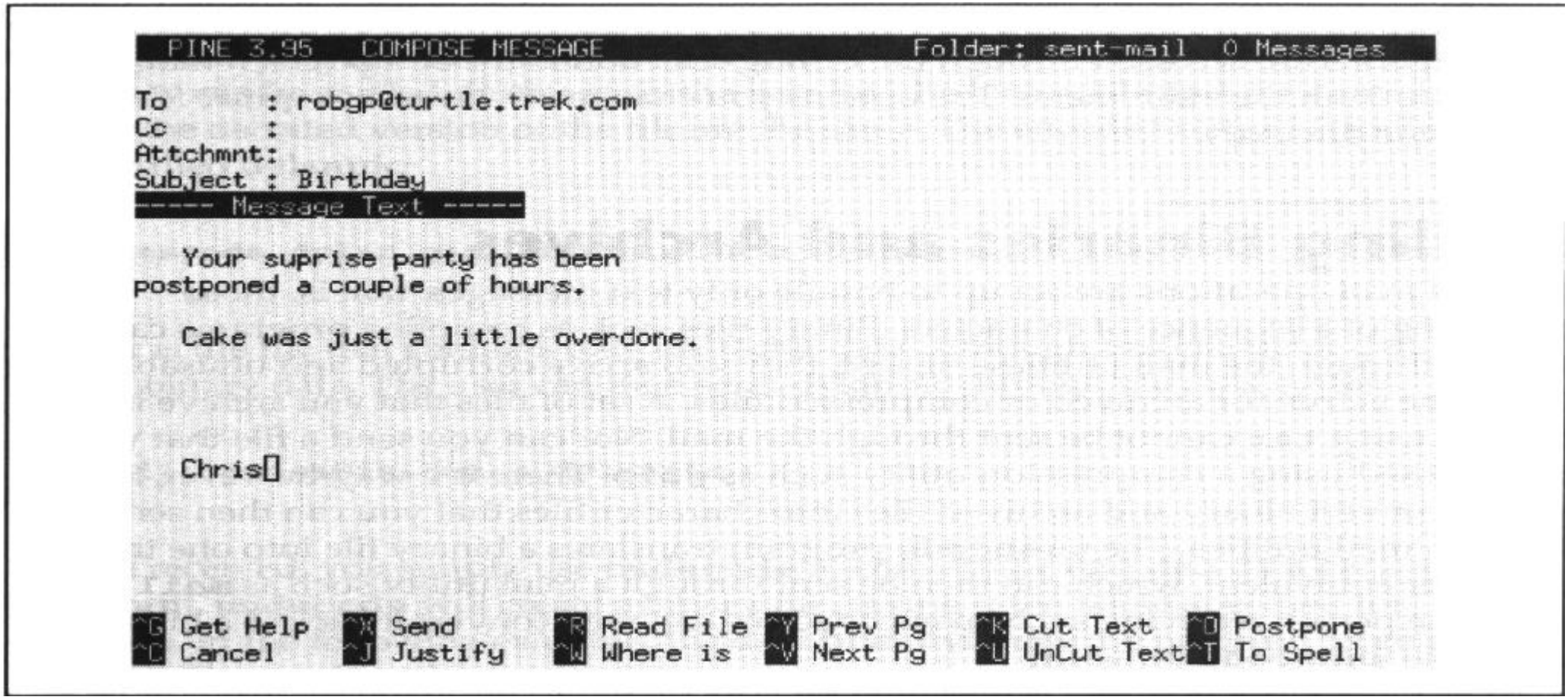


图 8-4 用 Pine 发出一个消息

Pine 将发出和接受到的信息整理到指定文件夹里去，你可以在主菜单的 Folder List 条目中指定这些文件夹。可以使用的的文件夹从左到右被列出来。三个文件夹会自动为你建立起来；INBOX，sent-mail，和 saved-messages。INBOX 文件夹乘放你已接到但尚未阅读的邮件。Sent-mail 文件夹为了你发给他人的消息，saved-message 文件夹里有你已经读过希望保存的消息。使用左右方向键选择一个文件夹，回车。选择了 INBOX 文件夹会列出你已经接收到的

消息。这些消息的头部会显示出来，可以选择任一头部来察看这条消息。你所选择的文件夹会成为缺省文件夹。通过在主菜单里选择 Folder Index 条目可以回到这个文件夹。

Pine 是为 Internet 设计的，它可以阅读 Internet 上的新闻组。在 Setup 画面中的 Config 条目能够打开一个选项列表，你可以在其中填入你的 Internet Email 服务器或者新闻服务器。你的邮件可以通过 Internet 发送出去。

Pine 将新闻组看成一个邮件文件夹。选择 Folder List，使用 A 命令加入新闻组的名字，可以列出这个新闻组。它会被当作另一个文件夹列出。当你选择了它，新闻组头部取代了邮件头而被列出。你也可以使用 Pine 寄出新闻，就像送出一个消息一样。

Pine 拥有大量的选项，通过它们你可以定制 Pine 的操作。通过选择主菜单里的 Setup，再选择 Config，可以进入这些选项。你所修改的选项被存在根目录下的.pinrc 文件中。

## 8.7Mailing Binaries and Archives

Internet 邮件操作是针对文本而建立的，所谓文本即字符串的集合。二进制文件，例如编译过的程序，不能被邮寄出去——如果邮寄，文件会被破坏使得不能使用。对于归档和压缩文件也是如此。你用 tar 软件归档为一个文件的一系列文件在通过邮寄后都将不可使用。同样，用 gzip 这样的压缩工具进行压缩的文件也不能邮寄。但是，可以将二进制文件和压缩文件转化成字符文件，以

进行邮寄。uuencode 软件可将一个二进制文件同等地转化为字符文件，后者可以通过象 mail 或者 elm 这样的邮件系统邮寄出去。接到序号后的文件的人可以再通过 uudecode 软件将它再解码复原为一个二进制文件。

uuencode 软件既可以用标准输入，也可以指定文件输入。无论哪种情况，你都必须为解码后的所创建的二进制文件提供一个文件名。uuencode 将序号后的数据输出至标准输出。uuencode 使用方法如下：

uuencode file name

name 是序号后的文件名，file 是被序号的二进制文件的名字。记住，因为 uuencode 将序号后的数据送往标准输出，你应当令这个输出指向一个文件；接下来你就可以它邮寄出去了。uudecode 软件 takes as its argument the file that holds the encoded data.它会使用你在 uuencode 命令中所提供的文件名生成一个二进制文件。基本步骤如下：

uuencode file name > datafile
mail adress < datafile
& s msg-num datafile
uudecode datafile

文件被序号然后象一个消息一样被寄出。当这条消息被接到后，它被存为一个文件，随后被解码，生成的文件被称作 name。

在下一个例子中，用户将一个名为 dylan.gif 的图像文件进行序号。象 gif 和 jpeg 这样的图像文件是二进制文件，在它们被邮寄之前必须先要转化成字符格式。在此例中，二进制文件的名字和将它序号后的文件名相同。序号输出被重置，指向了一个名为 dylanpic 的文件。

```
$ uuencode dylan.gif dylan.gif > dylanpic
```

dylanpic 文件中只载有字符型数据，但是这些字符型数据是被序号了的二进制数据。用户接下来将 dylanpic 邮寄出去。

```
$ mail larisa@ix.com < dylanpic
```

接到邮件后，你只需很简单地使用 uudecode 软件来将序号数据转化为它原来的二进制格式。uudecode 将生成一个二进制文件，这个文件使用在 uuencode 中给出的名字。在下面的例子中，dylanpic 文件中的数据被作为一个消息接收下来。接信人将这个消息存为 dylanpic。随后 uudecode 将这个消息转化成原来的二进制形式，存成文件 dylan.gif。接信人存储消息所用的名字并不要求与发信人使用的相同。你可以使用任何名字，但是必须与 uudecode 使用的相同。

```
$ mail
Mail version 5.5-kw 5/30/95. Type ? for help.
"/var/spool/mail/chris":1 message 1 unread
>U 1 robertMon Apr 8 00:06 236/14104
& s 1 dylanpic
"dylanpic"  [New file]
& q
$ uudecode dylanpic
$ ls
dylan.gif
```

对于归档和压缩文件你可以如法炮制。例如，为了利用一个消息将数个 gif 图像文件邮寄出去，首先，你可以将它们合并到一个文件中，压缩它。然后使

用 uuencode 将该文件序号，作为消息邮寄出去。收到消息的人做相反的事情：解码，解压缩，得到 gif 图像。你同样可以对全体目录和它们的子目录做相同的事情。在接下来的两个例子里，birthday 目录全体被 tar 归档 ，被 gzip 压缩。压缩文件被序号为字符型数据存在名为 birthday 的文件中。二进制数据的名字是 birthday.tar.gz。文件通过 mail 作为消息邮寄。接信人将消息存为文件 birthd，用 uudecode 将 birthd 解码，生成 birthday.tar.gz 文件，再经解压缩就可以重新生成全部 birthday 目录。

```
$ tar cvf birthday.tar birthday
$ gzip birthday.tar
$ uuencode birthday.tar.gz birthday.tar.gz > birthdaydir
$ mail aleina@pango1.com < birthdaydir
$ mail
Mail version 5.5-kw 5/30/95.  Type ? for help
"/var/spool/mail/chris":1 message 1 new
>N 1 robert                  Mon Apr      8 00:12 236/14162
& s 1 birthd
"birthd"  [New file]
& q
$ uudecode birthd
```

因为 uuencode 可以从标准输入接收二进制数据，你可以将 ARCHIVE、压缩、序号、和邮寄操作合并为一个管道操作，如下所示：

```
$ tar cf - birthday | gzip | uuencode birthday.tar.gz | mail aleina@pango1.com
```

tar 操作里的 - 指定了标准输出并通知 tar 将它的输出从文件输出转向标准输出。注意到了在 uuencode 的参数里含有解码二进制文件名，可以知道 uudecode 在对消息进行处理时会生成一个名为 birthday.tar.gz 的二进制文件。

uudecode 和 uuencode 程序在你的 Caldera 光盘里的 sharutils.4.1-2 软件包里。如果你的 Linux 系统是最小安装，里面不会有这两个程序。你需要使用 glint 或者 rpi -i 来安装软件包。表格 8-6 展示了一些邮寄和通讯工具。

## 8.8 收到信件的通知：from and biff

当你接到消息时，它们被自动放置在你的邮箱文件中，但并不会自动通知你接到了消息。为了能了解是否有消息还未读，你可以使用 Mail 工具来察看消息，或者使用 From and biff 工具来告诉你是否有未读的消息。

From 工具告诉你已经接到了的消息和等待被阅读的消息。对于每一个等待的消息，它都会列出发送者的地址和接收到的时间。启动 From，只需键入 from 并回车。

```
$from
1 From valerie Sun Feb 11 10:14:32 1996
   Subject: Budget
2 From aleina Mon Feb 12 12:30:17 1996
   Subject: Birthday
3 From robert Wed Feb 14 8:15:24 1996
```

```
    Subject: Homework
$
```

当你接受到一条消息时，biff 工具会及时通知你。当你正在盼望一条消息、希望它一到达就能知道的时候，它是很有用的。biff 会自动显示一条刚接到的消息的头部及其前几行。在命令行键入 biff y 可以打开 biff。键入 biff n 来关掉它。键入 biff 可以查询 biff 当前是开还是关。biff 一接到一条消息就会显示一个通知，不管当时你在干什么。你可能正在编辑一个文本，biff 就会跳出来在屏幕上显示一条通知。你可以接下来继续你的编辑。在下例中，用户首先打开了 biff。biff 随后通知用户接到了一条消息。用户接着检查了一下 biff 是否仍然打开。

```
$ biff y
$
New mail for chris has arrived:
--Data: Sun Feb 11 12:30:21
From: dylan
To: chris
Subject: Food
Chris,
Have you tried the chocolate
...more...
$
$ biff
is y
$
```

使用 mesg n 命令可以暂时将 biff 屏蔽掉，阻止消息在屏幕上出现。mesg n

不只屏蔽所有 Write 和 Talk 消息，它也屏蔽 biff 和 Notify 消息。使用 mesg y 命令可以撤销屏蔽。当你在做不想被打扰的工作时，mesg n 命令会很有用。

## 8.9 与其它登录用户通讯：Write 和 Talk

有时，你可能想同你的 Linux 系统上的其它用户进行即时通讯，而等不及他们浏览邮件。当你的 Linux 系统有其它用户登录时，通过使用 Write 和 Talk 工具你可以做到这点。Write 工作起来就像是无线电通讯，你可以联系上一个登录的人并在他的屏幕上显示消息。Talk 工具工作起来象电话，你可以与另一位用户进行直通双向交谈。

### 8.9.1 直接连接：Write 工具

Write 工具让你向另一用户发送即时消息。发送方键入的消息会马上显示在接受方的屏幕上。通过这种方式，Write 可以确保立即吸引一个人注意力。但是，Write 有一个明显的局限。它只能联系上已经登录的用户。如果你不能确定某人是否已登录，可以使用 who 命令来察看。

Write 不像标准邮寄操作。它不发送将放入邮箱文件的消息；它在另一位用户的屏幕上直接显示消息。当你键入 Write 命令、后面跟着一个用户的名字时会建立一个连接。你下面可以输入文本，它将会在对方的屏幕上显示出来。输入 CTRL-d 可以中断连接。接受方会首先在他或她的屏幕上得到一个通知，说明你发出了消息，并给出日期和时间。紧跟着这个通知，将显示消息。

在下例中，一个用户给 cecelia 写了条消息。输入完消息后，用户输入 CTRL-d 切断了连接。

```
$ write cecelia
How are you today?
^D
```

cecelia 接到了消息头和消息本身。发送人输入的 CTRL-d 显示为 EOT。

```
Message from gabriel  [Tues July 5 10:31]
How are you today?
EOT
```

你也可以用 Write 实现你和另一位用户间的双向交流。你在 Write 命令后附上接受方名字。接收到消息的用户使用他或她自己的 Write 命令，附上发送方的名字。你和另一用户相互发出的消息会在彼此的屏幕上出现。

交互式的 Write 通讯就好像你们在通过电台交谈。第一个用户发送一段消息，随后说明消息结束，接下来另一位用户进行响应。在许多 Linux 和 Unix 用户都接受的交谈中，一段消息的结束“over”通常用 o 表示，结束和发送“over and out”用 oo 表示，它表明你将结束通讯。你需要使用 CTRL-d 来从根本上切断与他人的连接。但是，对话双方都需要输入 CTRL-d。你的 CTRL-d 切断你与他人的连接，他人的 CTRL-d 切断他（她）与你的连接。

Write 通信将显示在你的屏幕上，无论你当前的工作是什么。如果你不想被 Write 的消息所打扰，可以使用 mesg 命令来屏蔽掉它。mesg 命令有两个选项，y 或 n。mesg -n 屏蔽 Write 信息；mesg -y 允许接收 Write 信息。

### 8.9.2 交互式通讯：Talk 工具

你可以通过 Talk 工具来实现你与其它用户间的双向交互式通讯。与 Write 不同，Talk 工作起来更像一个电话——你和其它用户可以同时输入消息。电话中两个人频繁地彼此交谈，Talk 的工作和它很相似。

输入 Talk，后跟另一个用户的称呼（通常是登录名），就可以启动一个通讯过程。这会在另一用户的屏幕上显示一条消息，给出你的称呼的同时询问他或她是否愿意交谈。他或她会使用自己的 Talk 命令做出回答。于是你和对方的屏幕都会被分割为两个区域。位于上方的区域显示你键入的东西，下方的显示对方键入的东西。任意方都可以通过 CTRL-c 来中断交谈过程。

## 8.10 总结：电子邮件

使用电子邮件工具，你能够在你的系统上向他人发送消息。一个用户的登录名确定了他或她的地址。在本章里介绍了两种界面迥然不同的常用电子邮件工具：Mail 和 Elm。

Mail 是一种在大多数 Linux 和 Unix 系统中常见的标准电子邮件工具。它拥有一个简单的命令行界面，拥有自己的在特定的 Mail shell 下操作的一整套命令。可以认为 Mail 中消息收发的命令与所有电子邮件工具中所定义的命令相同。当使用 Mail 发送一条消息时，你可以进行像重新显示消息、将消息存为文件、或激活 Vi 编辑器来编辑消息等等操作。你也可以在编辑器中创建消息，存为文件，然后将之发送出去。对于 Mail，这牵涉到用重定向把文件的内容作为邮件

命令的输入。当接收消息时，通过命令 mail 来激活 Mail 工具。你可以看到一个消息头的列表。每个消息头提供了对应消息的信息，包括发送人和消息的标题。各种 Mail 使得你能够浏览、打印、存储、或删除一条消息。你甚至能使用消息来即时回复他人。

在命令行的 mail 命令后列出用户们的称呼，可以向不止一个用户发送消息。或者，不用每次列出所有的用户称呼，取而代之，可以在.mailrc 文件中创建一个别名。.mailrc 文件中的命令在每次执行 mail 命令时都会执行一遍。当你向那些用户发送消息时，你可以使用别名来代替一大堆称呼。

Elm 工具使用一个全屏幕界面和单键命令来完成诸如发送、接收、存储消息、为多个称呼创建别名等操作。全屏幕界面常常使得 Elm 比 Mail 易于使用。

Write 和 talk 命令使人可以与其他的登录用户进行直接通讯。通讯中不使用消息。取而代之的是你所输入的东西被即时显示在给其他用户的终端上。Write 命令象无线电通讯，一个人先说，然后等待他人的回答。Talk 命令象电话通讯，两个用户可以同时说话。

表 8-1Internet 域

| Internet | 地址说明 |
|---|---|
| login-name@system.domain | Internet 邮件地址 |
| | chris@ garnet.rose.edu |
| 标准域 | |
| com | 商业组织 |
| edu | 教育院所 |
| gov | 政府机构 |

| | |
|---|---|
| int | 国际组织 |
| mil | 军队 |
| net | 网络组织 |
| org | 非盈利组织 |
| **国际域** | |
| at | 奥地利 |
| au | 澳大利亚 |
| ca | 加拿大 |
| ch | 瑞士 |
| de | 德国 |
| dk | 丹麦 |
| es | 西班牙 |
| fr | 法国 |
| gr | 希腊 |
| ie | 爱尔兰 |
| jp | 日本 |
| nz | 新西兰 |
| uk | 联合王国（英国） |
| us | 美国 |

表 8-2 用于发送消息的 Mail 命令

| Mail 命令行参数 | 说明 |
| --- | --- |
| -f 信箱-文件名 | 激活 Mail 工具，浏览你的目录下信箱文件中的消息 |
| -H | 只显示消息头 |
| -s 标题 | 发送消息时，指定消息标题 |
| -v | 显示用于发送消息的命令的执行结果 |
| 消息头上的~命令 | |
| ~h | 提示用户输入地址、标题、和副本列表 |
| ~s 标题 | 输入新标题 |
| ~t 地址 | 向地址列表中添加地址 |
| ~c 地址 | 向副本列表中添加地址 |
| ~b 地址 | 向盲副本列表中添加地址 |
| 消息正文中的~命令 | |
| ~v | 激活 Vi 编辑器；保存消息文本中的变化 |
| ~p | 重新显示消息正文 |
| ~q | 结束消息并退出 Mail 工具 |
| ~w 文件名 | 将消息储存为文件 |
| ~r 文件名 | 将一个文件的内容读入消息正文 |
| ~e | 激活缺省文本编辑器 |

| | |
|---|---|
| ~\|filter | 将消息内容通过管道送入过滤器，并使用过滤器的输出代替消息 |
| ~m | 消息-列表当发送消息或者回复接到的消息时，插入收到消息的内容；内容是 intended；接到消息的时候使用 |
| ~f | 消息-列表当发送消息或者回复接到的消息时，插入收到消息的内容；与 ~m 的差别是，内容不是 intended；接到消息的时候 |
| **使用常用~命令** | |
| ~? | 显示所有命令列表 |
| ~~ | 向文本中输入一个~符号 |
| ~! Command | 当输入一个消息时执行一个 shell 命令 |

**表 8-3 用于接收消息的 Mail 命令**

| 状态码 | 说明 |
|---|---|
| N | 新到消息 |
| U | 先前未阅览的消息 |
| R | 在当前过程中阅读消息 |
| P | 保存消息，浏览并存于接收邮箱 |
| D | 删除消息；消息标为删除 |

| | |
|---|---|
| O | 旧消息 |
| * | 已存往别的邮箱文件的消息 |
| h | 重新显示消息头 |
| z+ z- | 如果消息头列表多于一屏，将之上下滚动 |
| t 消息-列表 | 显示被消息-列表所指定的一条消息；如果没有任消息列表在使用，则显示当前消息 |
| p 消息-列表 | 显示被消息-列表所指定的一条消息；如果没有任消息列表在使用，则显示当前消息 |
| n 或 + | 显示下一个消息 |
| - | 显示前一个消息 |
| top | 消息-列表显示被消息-列表所指定的一条消息的头几行；如果没有任何消息列表在使用，则显示当前消息 |
| **消息列表** | |
| message-number | 用消息序号进行查询 |
| num1-num2 | 查询序号始于 num1 终于 num2 的消息. |
| . | 当前消息 |
| ^ | 第一篇消息 |
| $ | 最后一篇消息 |
| * | 所有等待在邮箱里的消息 |
| /pattern | 所有在主题区域带有该模式的消息 |
| address | 所有从 address 发来的消息 |

| | |
|---|---|
| :c | n 新收消息<br>o 老消息<br>r 已读消息<br>u 未读消息<br>d 删除消息 |
| **删除和恢复消息** | |
| d message-list | 使用 message-list 检索，从邮箱中删除一条消息 |
| u message-list | 使用 message-list 检索，恢复先前曾被删除的一条消息 |
| q | 退出 Mail 工具，将所有已读消息存于 mbox 文件 |
| x | 退出 Mail 工具，不删除任何已标为删除的消息；等效于在退出前对所有删除消息执行 u 命令 |
| pre message-list | 将消息保存于待读邮箱，无论它是否读过 |
| **发送和编辑消息** | |
| r | 所有接到消息的人发送回复 |
| R | 向发消息给你的人发送回复 |
| m address | 向一个使用 Mail 工具的人发送消息 |
| v message-list | 使用 Vi 编辑消息 |

| | |
|---|---|
| **保存消息** | |
| s message-list filename | 将使用 message-list 检索得到的消息存为文件，包括消息头 |
| S message-list | 将使用 message-list 检索得到的消息存为文件，以发送人的名字命名 |
| **保存消息** | |
| w message-list filename | 将使用 message-list 检索得到的消息存为文件，不包括消息头；只存储正文部分 |
| folder mailbox-filename | 切换到另一个邮箱文件 |
| % | 指定接收邮箱文件的文件名 folder % 切换到接收邮箱文件 |
| # | 指定 name of previously acessed mailbox file % 将切换到先前的邮箱文件 |
| & | 指定自动存放已读文件的邮箱文件名；常称为 mboxfolder % 切换到 mbox 文件 |
| **常用命令** | |
| ? | 显示所有 Mail 命令 |
| ! command | 在 Mailshell 内执行用户的 shell 命令 |

**表 8-4Mail 选项**

| 选项 | 说明 |
|---|---|
| alias name address-list | 为一个地址列表创建一个别名 alias |

| | |
|---|---|
| | myclass chris aleina larisa$ mail myclass |
| asksub | 输入题目 set asksub |
| askcc | 输入副本地址 set askcc |
| prompt=string | 重定义 Mail 提示符 set prompt="&" |
| sign=string | 定义字符串，以便随后使用~a 命令将之插入到你正入的消息中去 set sign="Robert and Valerie" |
| folder=directory | 将所有由命令 s 或 S 产生的邮箱文件存于指定目录 set folder=$HOME/mail |
| record=filename | 自动为你所创建和发送的消息保存一个副本；消息存于在 record 选项中所设定的文件中 set record=$HOME/outbox |
| outfolder | 将 record 文件放于文件夹目录中；如下例，outbox 是一个 folder 指定目录中的文件 set record=outboxset outfolder |
| MBOX=filename | 放有 mbox 文件的名字，已读消息将自动存入该文件中；在缺省的情况下，mbox 放在你的根目录下；将+放置在 mbox 之前可以将文件放到 folder 目录中 set MBOX=+mbox |

表 8-5Elm 命令

| 选项 | 说明发送消息 |
|---|---|
| elm address | 使用 Elm 发送消息 |

| | |
|---|---|
| s | 发送消息 |
| e | 编辑消息 |
| f | 退出，不发送消息 |
| h | 编辑消息头 |
| elm | 激活 Elm 工具 |
| ? | 激活帮助；键入命令，察看该命令的信息 |
| ? | 全部命令列表. 返回 Elm |
| q | 退出 Elm 工具，提示已读和未读消息，删除标为删息 |
| Q | 退出 Elm 工具，无提示 |
| x 和 CTRL –q | 退出 Elm 工具，不做删除和消息存储，所有消息保原样 |
| + | 显示下一屏消息头 |
| - | 显示上一屏消息头 |
| 选择消息 | |
| j | 移动到下一个消息头，设它为当前消息 |
| k | 移动到上一个消息头，设它为当前消息 |
| message-number | 将指定消息序号号的消息设为当前消息 |
| /pattern | 在主题或地址头中检索该模式，设第一个搜寻到为当前消息 |
| //pattern | 在消息正文中检索该模式，设第一个搜寻到的消为当前消息 |

| | |
|---|---|
| t | 给当前消息设标志；+会出现在消息之前；你可以消息设标志，然后对它们统一进行某种操作 |
| CTRL-t | 在主体和地址头中检索模式，将所有匹配消息标记 |
| **对消息的操作** | |
| **回车** | 显示当前消息 |
| i | 回到头部（索引） |
| p | 打印当前消息 |
| d | 删除当前消息；消息头被标为 D，并在退出 Elm 时删 |
| CTRL-d | 在主体和地址头中检索模式，将所有匹配消息除 |
| U | 恢复当前消息 |
| CTRL-u | 在主体和地址头中检索模式，将所有匹配消息复 |
| r | 回复当前消息；地址和主题取自当前信息头；你下面可以象处理任何其他消息一样对回复进行编辑和发送 |
| s | 存储当前消息，或在邮箱文件中所有被标志的消息；缺省时，被存储的消息的地址就是发送方的地址；你可以预先使用=来指定你自己的邮箱文件 =mailbox-filename 你也可以使用如下命令将消息存入接收或发送邮箱文件> 将消息存入接收邮箱文件< 将消息存入发送邮箱文件 |
| **操作** | |
| m | 将消息从 Elm 工具中发送出去；你编辑和发送一条 |

| | |
|---|---|
| | 消息 |
| c | 使用 Elm 对一个指定的邮箱文件进行操作；命令将操作对象从接收邮件文件切换到任意别的邮箱文件 |
| a | 管理 Elm 别名；键入 a 命令，别名菜单会弹出，有如下选项： a 使用当前消息的名字和地址创建一个别名<br>M 使用你输入的名字和地址创建一个别名<br>d 删除一个别名<br>l 列出所有别名<br>p 列出特定别名的名字和地址<br>s 显示所有系统别名<br>r 返回 Elm 主菜单 |

表 8-6 电子邮件和通讯工具

| 工具 | 说明 |
|---|---|
| mail | 电子邮件工具，用于收发邮件 |
| elm | 电子邮件工具，用于收发邮件 |
| uuencode filename | 将二进制文件序号为字符文件，使其可以通过电子邮件系统发送出去；重指向一个文件并发送该文件 uuencode filename > datafile |
| uudecode file | 将序号文件解码，生成一个二进制文件，名字取自 uuencode |

| | |
|---|---|
| from | 告诉你已接受到哪些消息 |
| biff | 通知你消息的到来 |
| write address | 在已登录用户的终端上显示消息 |
| talk address | 与已登录用户进行双向通讯 |

# 第 9 章 Usenet 和 Newsreaders

Usenet 是一种开放的邮件系统，在此系统上，用户可以发布新闻和自己的某些观点。它操作起来象一个全系统的邮箱，使得您的 Linux 系统上的任何用户都能够从此邮箱阅读和发送消息。用户的消息被表现为 Usenet 的文件，这些文件分布在能接受和识别它们的系统中。每个接受 Usenet 文件的系统被认为是一个站点。某些站点执行 Usenet 的组织和发布的操作，从一些站点接收到消息，然后把它们组织成 Usenet 文件，并以广播的形式发送给别的用户。这些站点叫做主干站点，它们担当发行人的任务，收到文章并把它们组织分配到不同的组中。

要访问 Usenet 新闻，您首先要能访问一个新闻服务器。新闻服务器收到每天的新闻素材，并使得它们能被别的系统访问。在您的网络环境中可以有一个系统充当新闻服务器。如果连接到某个 Internet 服务提供商（ISP），则 ISP 会提供一个新闻服务器供您使用。您还需要用一个新闻阅读器来阅读 Usenet 上的文章。新闻阅读器是指连接到新闻服务器并能存取文章的客户端程序。在 Internet 和基于 TCP/IP 的网络中，新闻服务器和新闻阅读器之间的信息传输使用的是网络新闻传输协议(NNTP)，新闻服务器也被认为是 nntp 新闻服务器。

另外，还可以在 Linux 上创建自己的新闻服务器，用来运行一个本地的 Usenet 新闻服务，或是用来下载和管理 Usenet 的所有文章。Linux 上新闻传

输代理程序(News Transport Agents)，便能用来创建这样的新闻服务器。

## 9.1Usenet 新闻

Usenet 文件夹被设计具有刊物的功能，包含在文件夹中的消息被认为是刊物中的文章。用户可以在 Usenet 上发表文章，而且使它立刻发布到世界各地的 Usenet 站点。别人可以通过 Usenet 马上阅读文章，而不必等待书面刊物的出版。因为每种刊物都属于不同的类别，所以对 Usenet 文件夹也进行分类，每类叫做一个新闻组。当用户发表文章时，此文章便被指定到相应的新闻组中。别的用户可以在此新闻组中找到此文章来阅读。您可以把每个新闻组看作是一本不断更新的杂志。例如，您想读关于计算机的文章，您可以到 Usenet 的计算机新闻组中找到此类文章。

近来，Usenet 文件夹也被用做为电子公告牌，大家在上进行一些讨论。这些文件夹被划分为不同的新闻组，尽管文件夹中的文章不像是杂志刊物中的文章，而更像是在谈话。

每个新闻组都有一个名字。为了分类，通常每个名字包括三部分：概括的主题，副标题和特定的主题。每部分之间用圆点分隔。例如，可能有一些关于娱乐的新闻组，叫做 rec。

其中有一些是关于食物的，叫 food。这些组中可能有一个组，用于专门讨论一个特定主题，假如是烹饪，叫 recipes。那么这个新闻组的名字为 rec.food.recipes。

大多数的电子公告牌只是用来讨论的，其中没有象刊物中发表的文章。这些组是用 alt 或 talk 作为概括的主题。例如，talk.food.chocolate 组中可能包括讨论巧克力多么好或坏的谈话，而 alt.food.cholate 中可能包括一些非正式的思索，如巧克力在人类文明中的重要性。下面的例子是一些 Usenet 新闻组的名字：

comp.ai.neural-nets
comp.lang.pascal
sci.physics.fusion
rec.ares.movies
rec.food.recipes
talk.politics.theory

Linux 有关于各种各样主题的新闻组。一些是讨论组，还有一些组中包括 Linux 的最新开发信息。如果您有一些问题，可以在这些组中寻求帮助。下面列出的是最近的一些很流行的 Linux 新闻组。从第 1 章中，您能找到更详细的列表。

comp.os.linux.announce 关于 Linux 开发的公告
comp.os.linux.admin 关于系统管理的问题
comp.os.linux.misc 关于特殊的问题和观点
comp.os.linux.setup 关于安装中的问题
comp.os.linux.help 问题和特殊难题的答案

阅读 Usenet 文章需要一个新闻阅读器，象 trn 或 tin。使用这些新闻阅读器时，首先要指定一个新闻组，然后阅读其中的文章。新闻阅读器有一个友好的用户界面，允许浏览、选择文章来阅读、保存或打印文章。trn 可能是当今最广

泛使用的新闻阅读器，它是早期的新闻阅读器 rn 的一个最新的，也是功能强大的版本。它使用了一种高级的检索机制-线索，用它来找出同一主题的文章。

您能写一篇文章，并把它加到一个新闻组中以供别人阅读。往一个新闻组中加一篇文章被称为发布文章。可以用一个特殊的功能-Pnews 来发表文章。

## 9.2 安装 trn 和 tin

在能使用 trn 和 tin 之前，您需要安装它们。在 Caldera 快速安装程序中，tin 并不能自动安装。您首先要以超级用户注册进入系统。在开始安装之前，要确信已用下面的命令把 CD-ROM 安装到 Linux 系统中：

```
$mount /mnt/cdrom
```

要安装 trn 和 tin，可以用您的 Caldera Desktop 上的 glint 安装程序，或在命令行上直接用 rpm。用 glint 前，首先需要打开一个窗口，接着用鼠标单击 Application 图标，然后单击 News 图标。将看到 trn 和 tin 软件。选择一个并单击它的图标，然后选择 Install 开始安装。

另外，还可以在命令行上进行操作。先转到 /mnt/cdrom/packages/RPMS 目录下，接着输入如下命令：

```
$rpm - i trn-3.6.1.i386.rpm
$rpm -I tin-1.22-2.i386.rpm
```

用 tin 和 trn 都能阅读远程新闻服务器上提供的 Usenet 新闻。这些新闻服务器使用网络新闻传输协议(NNTP)，而且可以通过 Internet 访问它们。为了连接到远端的新闻服务器，首先把新闻服务器的 Internet 地址赋给一个 Shell 变量

NNTPSERVER，接着把这个 Shell 变量传给当前环境。Shell 变量 NNTPSERVER 的赋值和传送工作可以定义在初始化文件.bash_profile 中，当注册时自动对此变量进行赋值和传送。

```
$NNTPSERVER=news.servername.com
$export NNTPSERVER
```

使用 tin 连接到远程新闻服务器时，要在 tin 命令行中使用-r 选项或用 rtin 命令。

```
$tin -r
```

## 9.3 新闻传输代理

Internet 上的 Usenet 新闻被作为每日新闻素材提供给超过 10，000 个新闻组。这些素材提供给一些站点，这些站点允许其他的系统通过新闻阅读器来访问其中的新闻。这些站点便是新闻服务器，访问它们的新闻阅读器作为它们的客户端。新闻服务器软件叫做新闻传输代理程序，它能给新闻阅读器提供新闻，并使您能阅读新闻组和在新闻组上发表文章。Linux 上三个最流行的新闻传输代理程序为：INN，nntp 和 Cnews。Cnews 和 nntp 都是小型的、简单的软件，它们是专为小型的网络设计的。INN 是一种功能强大的软件，专为大型的网络设计。

Usenet 上的每日新闻素材通常数量巨大，需要消耗服务器的大量时间和内存。所以您可能不希望用自己的 Linux 系统来接收这些素材。如果您的系统运行在一个网络环境中，可以用网络中的一个系统作为新闻服务器，在此系统中

装上新闻传输代理软件。此服务器便可以用来接收和管理新闻素材。网络中的其他系统上的用户可以通过新闻阅读器来访问此新闻服务器。

如果在网络环境中已有一个新闻服务器，就不需要再安装新闻传输代理程序。只须用您的新闻阅读器来访问此服务器（参见前面章节中关于NNTPSERVER）。如果使用 Internet 服务提供商(ISP)，这些 ISP 都有自己的新闻服务器，您能用新闻阅读器 trn 或 tin 连接到这些远程服务器。但是要记住 trn 和 tin 将花费一些时间去下载选定的新闻组上的所有文章，和下载新闻组上所有的更新信息。

您可以用新闻传输代理软件来建立一个本地网络新闻组，以供您的系统用户和本地网络上的其他用户来访问。您能安装 nntp 或 Cnews 来管理本地新闻组。这样，您的系统用户就可以发表文章和阅读本地新闻。虽然 nntp 和 Cnews 更适合局域网络，但是您也可以使用 INN。

### 9.3.1INN

在您的 Caldera CD-ROM 中的 InterNetNews(INN)新闻传输代理软件，提供了新闻服务器的所有功能。您能使用 glint 或 rpm 从 Caldera CD-ROM 中安装它，或从 Linux 的 ftp 站点上下载，这样的 ftp 站点很多，象 sunsite.unc.edu 等。当您用 rpm 安装时，记住要选用 INN 软件的 Redhat 版本。下面是一个用 rpm 安装 INN 的例子。

```
$rpm -i inn-3.6.1.i386.rpm
```

通过此命令可以安装 INN 程序，同时也在相应的目录下安装了配置文件。

在您的/usr/doc 目录下，可以看到一个关于 HOW-TO 文件的目录。INN 被

配置为启动系统时自动启动。INN 的主要程序是 innd。d 代表一个守护进程，当您的系统启动时，此进程便被启动，并在您工作时始终在后台运行。大部分 INN 配置文件放在/etc/news 下。在此目录下，文件 inn.conf 用来为 INN 设置选项，host.nntp 用来控制对您的系统中新闻的访问。

可以按您的需要编辑任何的配置文件。man 页的帮助中有关于所有的 INN 程序和大多数配置文件，象 innd，inn.conf，rnews 和 host.nntp。正确配置 INN 是一个复杂且费时的过程。

一定要仔细的阅读参考手册和您的 CD-ROM 上/docs/HTML/news.html 页面下的 HOW-TO 文档。

## 9.4trn 新闻阅读器

用流行的 trn 新闻阅读器，可以浏览一列新闻组，并从中选择一个以阅读其中的文章。trn 的界面有许多强大的功能，例如可以在一组文章中进行模式搜索。trn 新闻阅读器是 rn 的增强版本，它允许您通过主题、文章或线索来显示和搜索文章。trn 中的 t 代表线索。线索是文章之间的连接，例如文章和前面发表的文章之间是同一主题，或此文章是前文的后续文章。您仍然可以使用标准的 rn 命令从一篇文章移动到下一篇。然而，trn 具有一种特殊界面，被称为选择器。在其中可以通过线索来移动文章。例如，如果您读一篇关于特定主题的文章，当输入 n 命令时，便移动到线索中此主题的下一篇文章，而不像通常的移到下一篇顺序发表的文章。除了按发表文章的顺序在文章中移动外，还可以

按不同的线索或不同主题来移到不同的文章。对于文章和它的后续文章也是如此。一篇文章和它的后续文章之间用线索连接，当您阅读一篇文章时，用 n 命令将从此文章中移动到它的第一篇后续文章，而不是移到它的下一篇顺序发表的文章。这样，用线索和 n 命令就可以阅读一篇文章和它的所有后续文章，而不用再一一搜索。在表 9-1 中列出了 trn 新闻阅读器的命令。

trn 在两个级别中操作：新闻组列表和文章列表。当您第一次执行 trn 时，可以从新闻组列表中选择一个。也可以使用命令在新闻组列表中上下移动，一旦找到您想看的，选中它，就可以选择和阅读其中的文章。当读完后，还可以离开此新闻组，从新闻组列表中再选择您感兴趣的其他新闻组来阅读。可以用 trn 的选择器来显示、组织和移动文章。也可以使用标准的 rn 命令来管理文章列表。

trn 新闻阅读器通过是否包含有未读的新闻来区分新闻组。您能用 trn 的特殊命令来寻找那些只包含未读新闻的新闻组。未读新闻是指新闻组中从未读过的文章。trn 通过放在用户的帐号下的.newsrc 文件来跟踪文章的已读或未读状态。每个用户都有自己的.newsrc 文件，它包含了 Usenet 服务器提供的所有的新闻组。每一项用来跟踪其中是否包含未读或已读的新闻。

trn 新闻阅读器允许您订阅一个新闻组或撤消订阅。当第一次运行 trn 时，您将获得 Usenet 服务器上所有新闻组的访问权，也自动地给您订阅所有的新闻组。然而，您可能对好多新闻组并不感兴趣，为了不使那些您不感兴趣的新闻组扰乱您的界面，可以撤消对它们的订阅。如果以后您又对某些新闻组感兴趣，还可以再订阅。在.newsrc 文件中记录着您订阅的新闻组。

### 9.4.1 新闻组列表

在 Linux 系统提示符后面输入 rn，则启动 rn 新闻阅读器。rn 将首先显示新闻组标题的一个简短列表。而当启动 trn 时，在显示列表前，trn 将首先把新增的新闻组列表和您的.newsrc 文件中的列表相对照。如果有新增的新闻组未列在文件中，trn 将逐个问您是否要订阅此新闻组。在提示下，输入 y 表示订阅，否则输入 n。

也许会有大量新的新闻组需要您添加。如果您不想在启动后进行此项工作，使用 trn 命令时加上-q 选项即可跳过此过程。trn 将直接显示新闻组标题的一个简短列表，跳过增加新闻组的订阅询问过程。

```
$trn -q
```

此后，trn 将检查您的.newsrc 文件中列出的新闻组中是否有未读的新闻。如果有，将显示这些新闻组标题。新闻组标题中会提示此新闻组中有多少未读新闻。接着 trn 会提示是否要阅读这些文章。如果您不想阅读，输入 n 命令则移动到下一个新闻组。p 命令使您移回前一个新闻组。

在提示符后输入+命令，则可以进入 trn 选择器，其中列出新闻组中的文章。您能从中选择您想显示的文章。如果输入 y，将显示新闻组中的第一篇文章。接着会提示您是阅读下一篇文章，还是退回到新闻组列表。在提示符后输入 q，即可离开此新闻组，并返回到新闻组列表。

下面的例子中，使用者首先进入 trn 界面，显示一列新闻组标题。提示第一个新闻组，用 n 命令跳过此新闻组。在下一行提示中，使用者输入一个+命令，则显示新闻组 comp.os.linux.misc 中的文章。

```
$trn
```

```
comp.a.language3articles
comp.os.linux.misc                1article
rec.arts.movies                   7articles
rec.fool.recipes                  245articles
sci.physics.fusion                32 articles
talk.politics.theory              126 articles
  etc.
===== 3 unread articles in comp.ai.language -- read now ?  [+ynq]  n
===== 1 unread articles in comp.os.linux.misc -- read now ?  [ynq]  +
```

trn 有大量的命令操作在新闻组列表中的移动。您能移动到第一个或最后一个新闻组，移动到上一个或下一个新闻组，或是直接移动到特定名字的新闻组。例如，$将使您直接移到新闻组列表的末尾。许多命令用来区分已读和未读的新闻组。^将使您移到第一个包含未读新闻的新闻组，而数字 1 将使您移到所有新闻组列的第一个。小写字母 n 和 p 将使您移到下一个和上一个包含未读新闻的新闻组。大写的 N 和 P 将使您移到所有新闻组列表中的下一个和上一个新闻组。

当第一次运行 trn 时，您的新闻组中将包括大量未读的文章。您可以使用一个简单的命令把它们标为已读，而不用去一一阅读。在新闻组的提示符下输入命令 c，则把该新闻组中所有的文章标为已读。此时，用于选择未读新闻组的命令 n 和 p 则不能选择，直到有新的文章加到新闻组中。

通常您都知道要访问的新闻组的名字。可以不必使用 n 和 p 命令逐个移动，而使用模式查找直接移到您想访问的新闻组。模式查找命令使得 trn 在定位新闻组方面的功能更强大。在提示符后，键入/和要查找的模式。/执行在新闻组列表中向前搜索，?执行向后搜索。下面的例子中，用户搜索新闻组 food recipes。

```
$trn
comp.a.language3 articles
comp.os.linux.misc                1 article
rec.arts.movies                   7 articles
rec.fool.recipes                245 articles
sci.physics.fusion               32 articles
talk.politics.theory            126 articles
etc.

====== 3 unread articles in comp.ai.language - read now ?  [+ynq]  n
/food.recipes
Searching...
====== 245 unread articles in rec.food.recipes - read now?  [+ynq]  +
```

**您还可以通过一个新闻组的全名来定位它。命令** g **加上新闻组的名字将定位到此新闻组。**

```
====== 3 unread articles in comp.ai.language - read now ?  [+ynq]
g rec.food.recipes
Searching...
====== 245 unread articles in rec.food.recipes - read now?  [+ynq]
```

trn **的列出和查找的命令只是针对与您已订阅的新闻组。用命令** l **您能列出和查找未订阅的新闻组。单独使用命令** l **可以列出所有的未订阅的新闻组。在** l **后加上一个模式，则查找包含此模式的所有未订阅的新闻组。例如，**ltrek **将查找未订阅的新闻组中名字包括“**trek**”的新闻组。**

**使用命令** a **可以订阅一个新闻组。只须用** a **加上要订阅的新闻组名字。还**

可以使用命令 u 来取消订阅。例如，u rec.foods.recipes 将取消此新闻组的订阅。取消后，您就不能再用查找命令/和 g 来选定它。当然，您可以用命令 l 来定位它：l rec.foods.recipes。用命令：a rec.foods.recipes 可以再次订阅此新闻组。

### 9.4.2trn 选择器

前面已经提到，在 trn 的提示符后输入+，则可以进入选择器。在选择器中可以使用线索。选择器的界面中包括每个文章的作者、关联的线索数和主题。任何后续文章的前面加有>符号。文章是按照它所属的线索来分类的。每一个线索的第一篇文章前加有一个 ID，ID 包括一个小写的字母或单个数字，从字母 a 开始。下面是一个 trn 选择器的例子：

```
rec.food.recipes258 articles(moderated)
a Dylan Chris1Fruit Salad
b       Cecelia Petersen        1       Fudge Cake
d       Richard Leland          2       Chocolate News
        Larisa@atlash
        Aleina Petersen         1       >White chocolate
        Maryann Price           1       >Chocolate Fudge
        mark@pacific            1       >
        Justin G.               1       >Chocolate
e       George Petersen         1       Apple Muffins
f       Marylou Carrion         1       REQUEST: romantic dinners
g       Valerie Fuller          1       REQUEST: Dehydrated Goodies
```

```
i       Carolyn Blacklock          1       REQUEST: Devonshire Cream
         Bill Bode                 1       >
j       Bonnie Matoza              1        Sauces
l       Gabriel Matoza             1       Passion Fruit
o        Ken Blacklock             1        REQUEST: blackened (red)fish
         augie@ napa                  3       >blackened (red)fish
         John Carrion
         Anntoinnete
 -- Select threads (date order) - 24%
```

**上面例子是进入选择器后显示的第一屏，第一篇文章前加有字母** a。**您按下空格或**>**键即可显示文章的下一屏。按一下**<**键即可显示前一屏。从上面例子开始显示的下一屏中，线索仍然从** a **开始显示。每一个线索前的** ID **在一屏中是唯一的；它们只是用来标识屏幕中的文章线索。**

**为了阅读一篇文章，首先应选择文章的线索，然后在选择器中显示此文章。您可以按下和** ID **相应的键来选择一个线索。例如，按下** d **即可选择第一行前加有** d **的线索。在上面例子中，如果选定一个线索，则在其** ID **之前会显示一个符号**+**。一旦选定一个线索，按下** ENTER **键或大写字母** Z **即可显示该线索的文章。首先显示的是此线索的第一篇文章。按** n **键可以移到该线索的下一篇文章。一旦找到您需要的文章，用标准的** trn **命令就能显示以阅读此文章。如果显示的文章超过一屏，按** SPACEBAR **键可显示下一屏。**

**任何时候您都可以通过按**+**键返回到选择器。当您不想阅读一个选定线索中的文章，需要删除对此线索的选定。通过按两次和** ID **所对应的字母键，您可以**

删除选定。当您第一次按下相应的字母键，在 ID 前将显示一个符号+。再按一次，则符号+消失，代表删除选定。

您也可以用光标命令来选定一个线索。当初次进入选择器屏幕时，光标在第一个线索的 a 的前面。用光标命令 n 和 p 或箭头键，可以移动到下一个或退回到前一个线索。用 n 命令或下箭头移动到下一个线索，p 命令或上箭头退回到前一个线索。为了显示一个特定线索中的文章，您只须把光标移到此线索的 ID，然后按 ENTER 或 Z 键即可。

如果您想一次阅读多个线索，通过按下和每个线索的 ID 相应的键，就可以给每个您想阅读的线索做一个标记。例如若想选定 ID 为 b 的线索，按 b 键即可。然后移动到下一个您需要选择的线索，重复同一动作。在每个您选择的线索前都显示一个+。也可以从一屏移到下一屏，选择您需要的线索。全部选定后就可以显示这些线索的文章。

### 9.4.3 选择器的显示模式：文章，主题和线索

选择器可以按三种显示模式来显示文章，这三种模式为：文章，主题和线索。输入命令 S 后，输入 a 则选定文章模式，输入 s 则选定主题模式，输入 t 则选定线索模式。在不同模式下，按=键可以向后和向前转换。

每种模式也可以被认为是一种线索。当使用主题模式是，文章按照主题项进行分组。主题是用户在文章标题的主题区域输入的内容。有相同主题项的文章被放在同一线索下。主题分组只限于那些有相同主题项的文章。文章的主题稍有不同便不会被放在一起。

当使用线索模式时，文章和它的后继文章以及有相同主题的文章被放在一

起。后继文章前带有符号>。线索模式和主题模式的不同在于它把文章的所有后续文章放在一起，尽管一些后续文章可能有不同的主题。文章模式不显示线索。在此模式下，文章只是按发表的顺序排列，每篇文章带有自己的 ID。

根据您选用的显示模式不同，选择器的屏幕显示也将不同。在主题模式下，选择器显示每篇文章的作者，并把它们按照主题分类。首先是主题下的第一位作者，然后是此主题下的文章数，接着是主题。主题只在第一篇文章后显示一次。在主题模式下，ID 代表一个主题，不是某一篇文章。在屏幕上，ID 只被放在不同主题分组的开始。按下和 ID 相应的字母键，即可选定一个主题。主题模式提供一种简单的方法来访问不同主题的文章。同时主题标题的列表也是对新闻组中讨论的主题的总结。

下面的例子中，选择器是在主题模式下。注意 d 项指定的主题为 Chocolate News。在此主题下有两篇文章，一篇的作者是 Richard Leland，另一篇的作者是 Larisa@atlash。文章数为两篇。项 i 也有两篇文章，第二篇是一篇的后续文章。用符号>说明。项 u 说明主题下有三篇文章，都是后续文章。注意它实际是项 o 的后续文章，虽然项 o 有不同的主题。

```
    rec.food.recipes258 articles(moderated)
    a       Dylan Chris                 1       Fruit Salad
    b       Cecelia Petersen            1Fudge Cake
    d       Richard Leland              2Chocolate News
            Larisa@atlash
    e       George Petersen              1Apple Muffins
    f      Marylou Carrion             1REQUEST: romantic dinners
    g       Valerie Fuller            1REQUEST: Dehydrated Goodies
```

```
i       Carolyn Blacklock          1REQUEST: Devonshire Cream
        Bill Bode                  1>
j       Bonnie Matoza               1Sauces
l       Gabriel Matoza             1Passion Fruit
o        Ken Blacklock              1REQUEST: blackened (red)fish
r       dylan@sf                    1REQUEST: Cheese Toast
s        Penny Bode                  1REQUEST: Sausage Recipes
t       gloria@stlake              1Biscuit Recipe
u        augie@napa                  3>blackened (red)fish entree
        John Carrion
        Anntoinnete
v       John Gunther
-- Select threads (date order) - 24%
Selector mode: threads, Subjects, Articles?  [tsa]  s
```

**在线索模式中，选择器显示的文章是通过后续连接或是通过主题连接的。也即是说，在线索模式下，无论通过何种方式互相连接的文章均被分在一组。后续文章被列在原始文章的后面，且后续文章带有符号>。分在一组中的文章便在同一个线索中。用线索模式您可以方便的访问一篇文章和它的所有后续文章，以查看所有的讨论和意见。通过主题连接的文章也处在同一线索中，并和它们的后续文章放在一起。每个线索有自己的 ID。按下和 ID 相应的字母键即可选择一个线索。**

**下面的例子中，选择器是在线索模式下。注意屏幕的显示和主题模式下的显示有所不同。项 d 包括同主题的文章，也包括后续文章，共有六篇。前两篇有相同的主题，后四篇是它们的后续文章，带有符号>。大多数的后续文章的主**

题是不同的，其中的两个，作者是 Maryann Price 和 mark@pacific 的文章有同一主题。项 o 代表一个线索，它的第一篇是 Ken Blacklock 的文章，还包括它的三篇后续文章：augie@ napa，John Carrion，Anntoinette。

虽然这三篇文章回复自线索中不同的文章，但它们有相同的主题。

rec.food.recipes258 articles(moderated)

a Dylan Chris1Fruit Salad

b Cecelia Petersen 1Fudge Cake

d Richard Leland 2Chocolate News

Larisa@atlash

Aleina Petersen 1>W hite chocolate

Maryann Price 2>Chocolate Fudge

mark@pacific

Justin G. 1>Chocolate

e George Petersen 1Apple Muffins

f Marylou Carrion 1REQUEST: romantic dinners

g Valerie Fuller 1REQUEST: Dehydrated Goodies

i Carolyn Blacklock 1REQUEST: Devonshire Cream

Bill Bode 1>

j Bonnie Matoza 1Sauces

l Gabriel Matoza 1Passion Fruit

o Ken Blacklock 1REQUEST: blackened (red)fish

augie@ napa 3>blackened (red)fish

John Carrion

Anntoinnete

-- Select threads (date order) - 24%

Selector mode: threads, Subjects, Articles? [tsa] t

下一个例子中，文章是按照发表的顺序排列的，每篇文章有自己的 ID。注意项 d 和项 e 有同一主题。在文章模式中没有线索，文章和它们的后续文章是分开排列的。例如，虽然 augie@napa 的文章是 Ken Blacklock 的文章的后续，但它们分布在不同的部分，且有各自的 ID：r 和 v。

rec.food.recipes258 articles(moderated)

a Dylan ChrisFruit Salad
b Cecelia Petersen Fudge Cake
d Richard Leland Chocolate News
e Larisa@atlash Chocolate News
f George Petersen Apple Muffins
g Marylou Carrion REQUEST: romantic dinners
i Valerie Fuller REQUEST: Dehydrated Goodies
j Carolyn Blacklock REQUEST: Devonshire Cream
l Bonnie Matoza Sauces
o Gabriel Matoza Passion Fruit
r Ken Blacklock REQUEST: blackened (red)fish
s dylan@ sf REQUEST: Cheese Toast
t Penny Bode REQUEST: Sausage Recipes
u gloria@stlake Biscuit Recipe
v augie@ napa >blackened (red)fish
w John Gunther Oatmeal Cookies
x Margaret REQUEST: Potato Salad

```
y       Frank Moitoza          REQUEST: Sesame Chicken
z       maryann@sebast           >Summer desserts

 -- Select threads (date order) - 24%
Selector mode: threads, Subjects, Articles?  [tsa]  a
```

### 9.4.4 显示文章：trn 线索树

当您选定一篇文章时，便显示文章标题，文章的第一屏和一个 more 提示行。文章被一屏一屏地显示，就像用 more 命令显示文件。按空格键即可显示下一屏。您可以按 b 键退回到前一页。用命令 q 中止显示并退出。还可以在文章中进行模式查找。用命令 g 加上要查找的模式即可以查到此模式在文章中第一次出现的地方。用命令 G 可以重复此查找。

```
rec.food.rcipes(moderated) #7155 (229 more)
From : richpete@ garnet.berkeley.edu (Richard Petersen)
[1]  Spelling Cookies
Followup -To: poster
Date: Mon Dec 20 04:48:01  PST  1995
Organization: University of California, Berkeley
Lines: 43
Spelling Cookies
INGREDIENTS
1 cup flour
2 treaspoons single acting baking powder, or 1 teaspoon double acting
```

baking powder,or 1 teaspoon baking soda
1/2 teaspoon nutmeg
1/4 teaspoon cinnamon
3/4 cup butter ( or 1 1/2 sticks)
1 cup brown or dark brown sugar
1/2 cup regular sugar
1 egg
1 teaspoon vanilla extract
1/4 cup milk
--More-(44%)
**在上面显示的状态下按空格键可以显示下面的信息。**
rec.food.rcipes(moderated) #7155 (229 more)
［1］ Spelling Cookies
12 ounces semisweet chocolate chips
pecan or walnut bits
3/4 cup wheat germ
2 3/4 cups rolled oats(Old Fashioned Quaker Oats)
STEPS
1. Mix brown and regular sugars together
2. cream butter, then mix in sugars. Then mix in egg, vanilla extract, and milk until creamy.
3. Separately mix flower, cinnamon, nutmeg,baking powder(or baking soda),salt(ifwanted)
4. Mix in flower concoction into butter/sugar concoction until creamy.
5. With spatula add in wheat germ, then nuts, then chocolate chips, then

oats.

6. Preheat oven to about 300 degrees. Spoon out onto cookie sheets. Cook for about 20 minutes or so. Watch carefully. When top of cookies begin to brown they are done.

7. Eat immediately or refrigerate. Aging only improves taste.

8. These are called spelling cookies because when I gave the recipe to a friend in my class he daughter found so many spelling mistakes that she called them spelling cookies.

9. Eat at you own risk. Gook luck.

End of article 7155( of 7158) - what next ? [npq]

**在方括号中，按 n 将显示新闻组中的下一篇文章，按 p 显示前一篇文章。按 q 将退回到新闻组列表中。如下例，按+可以回到选择器中。**

End of article 7155( of 7158) - what next ? [npq]

**当显示一个线索的第一篇文章时，屏幕的右上角会显示一个线索树。线索树代表一个线索中文章之间的联系。每一篇未读的文章被一个数字代表，该数字从 1 开始，位于一个方括号中。每个数字之间用线连接。当前您正在显示的文章的代表数字会增亮显示。一旦您读完一篇文章，移到下一篇文章时，读完的文章的代表数字会用圆括号括起来，而移到的下一篇文章的数字会增亮。**

**一个线索能标明文章之间的关系。线索中的第一篇文章位于线索树的左上角。树的分支位于它的右下方，下一列是后续文章。所有的后续文章用线连接起来。例如，第一篇文章的后续文章的代表数用括号括起来，并放在第一篇文章的代表数的右边作为新的一列。文章数和后续文章数之间用线连接。关于此文章的其余的后续文章被放在第一篇后续文章所在列的下面，每篇文章之间也**

用线连接。最后一篇后续文章用斜线连接。图 9-1 显示一个线索树。

大家可能针对某人的回答来展开一个对话和讨论。后续文章也可能有自己的后续文章。二级后续文章的代表数字被放在后续文章数字的右边。二级后续文章可能还有后续文章，它的后续文章的代表数字仍是放在它的右边。如图 9-2 所示，后续文章和它的后续文章放在一行中。

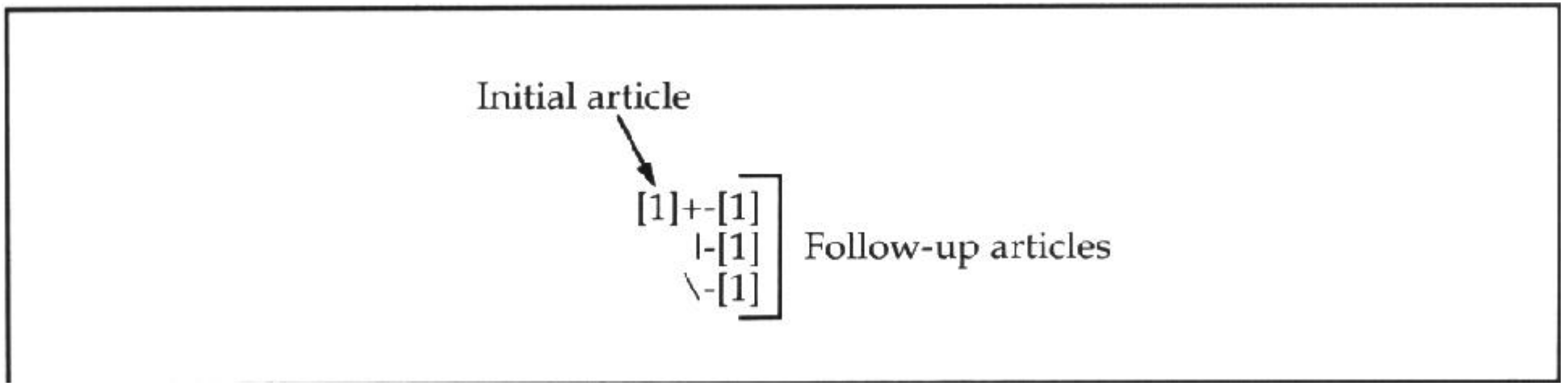


图 9-1 显示初始文章和后续文章的线索树

任何一篇文章可能有不止一篇的后续文章。一篇文章的所有的后续文章被放在同一列中，因此，一个线索树可以向水平和垂直方向扩展。水平方向是后续文章和它的后续文章。垂直方向是一篇文章的所有后续文章。

同第一篇文章有相同主题的文章被列在第一篇文章的下面。它们之间并没有用线连接。如图 9-3 所示，最左面一列文章代表数是主题的分类，右边的列代表后续文章。有相同主题的文章在同一列中，但之间没有连接线。

线索树中的文章的代表数用来表明是否有文章共享主题。第一篇文章用数字 1 来代表，表明是此线索的初始文章。有相同主题的文章有相同的代表数字。

别的代表数字也为 1 的文章表示和第一篇文章有相同的主题。在同一线索中的有不同主题的文章有不同的代表数字。后面的文章如果和前面文章有相同的主题，则使用前面文章的代表数字。第一篇和代表数字为 1 的文章有不同主题的文章用数字 2 来代表，别的和它有相同的主题的文章也使用代表数字 2。在线索中遇到一个不同的主题，则代表数字加 1。这种记数方式使您能识别线索中有不同主题的部分。这也正象检测谈话过程中转向不同的话题的部分。另外，线索中可能包含您不感兴趣的主题，可以很容易的识别出来，并不去阅读它们。图 9-4 显示一个有不同主题的线索树。第一篇后继文章的代表数字为 2，表明此文章于第一篇文章有不同的主题。此文章下面的文章的代表数字为 3，表明它有另一个主题。

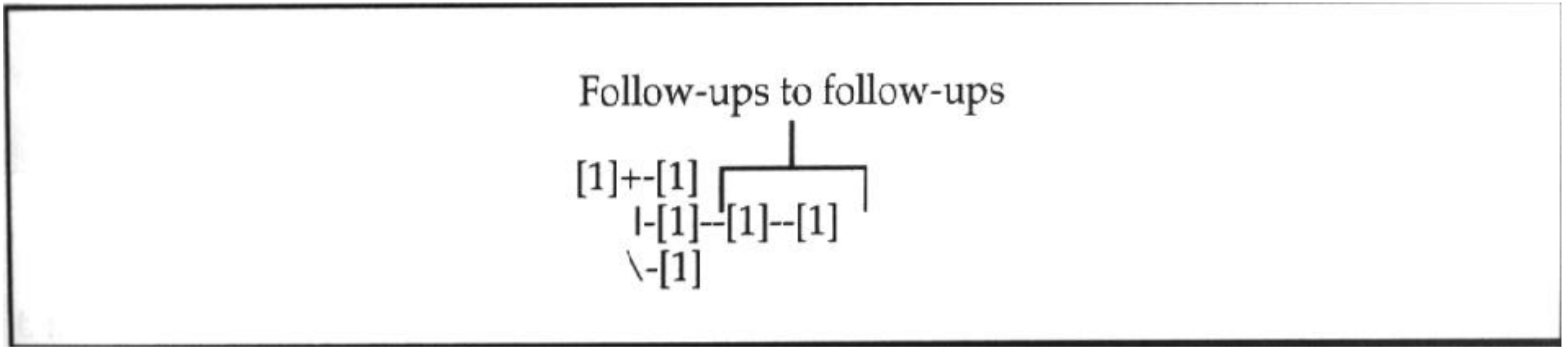


图 9-2 显示后续文章和它的后续文章的线索树

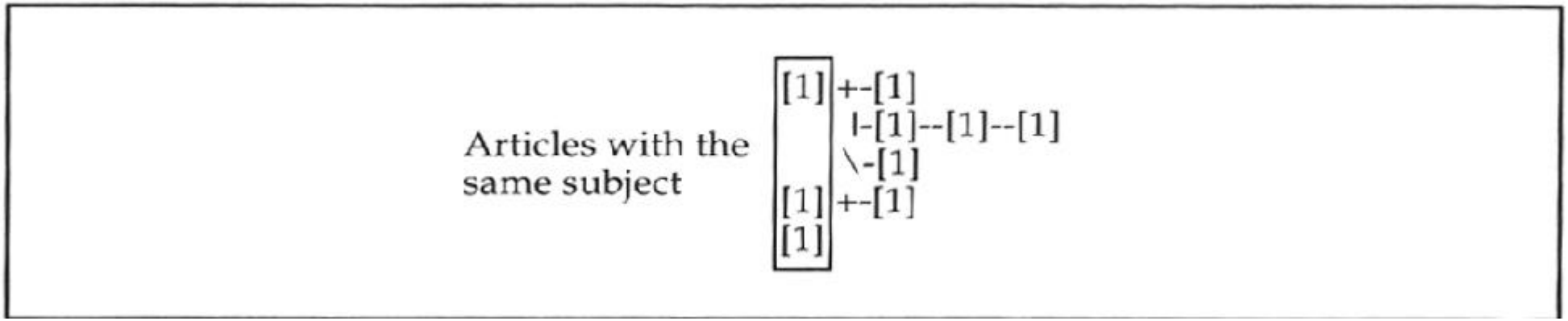


图 9-3 显示主题文章的 trn 线索树

您能在线索树中从一篇文章移到另一篇，直接显示想看的文章，而不用显示中间的文章。

还可以在线索树中向后移动。线索树对于在其中向前或向后移动是一种很方便的方法。您能用箭头在线索树中上下移动，移到的数字便会增亮。例如，用下箭头从当前的文章移到下一篇，下一篇的数字增亮。按右箭头移到右边的数字。按左箭头移回左边的数字，按上箭头能向上移动。当前增亮的数字所代表的文章被显示出来。当您在线索树中移动到不同的数字时，显示不同的文章。

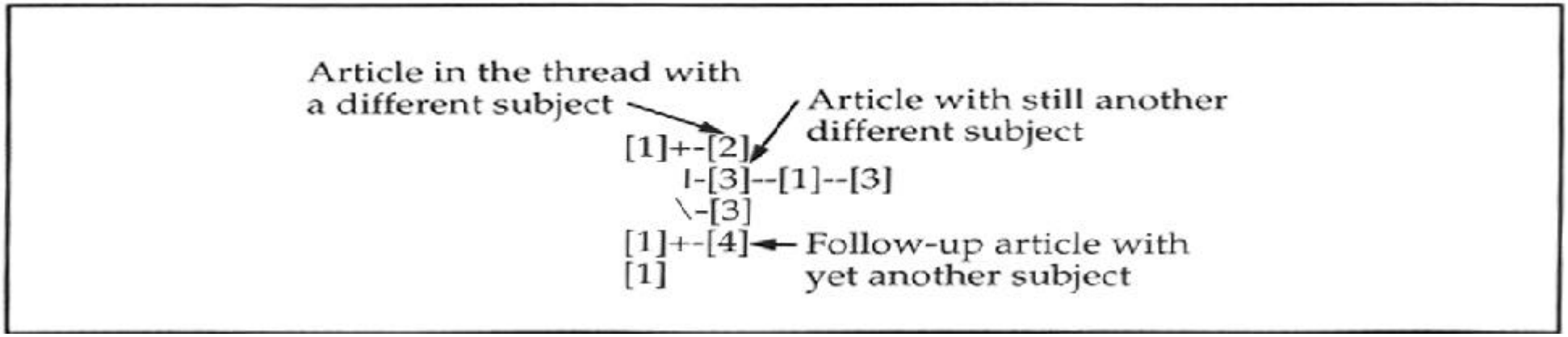


图 9-4 显示不同主题的文章的 trn 线索树

### 9.4.5 保存文章

您能保存正在阅读的文章。显示文章时，输入命令 a 和文件名即可保存此文件。如果文件还未保存，系统将提示您是否想以邮件格式保存文件。您输入 y，文件将保存为邮件格式，文章将作为邮箱中的一个消息。能用 mailx 命令和 -f 选项来读取存在文件中的文章。输入 Enter 键，则文件被保存为文本格式。

```
rec.food.rcipes(moderated) #7155 (229 more)
From : richpete@ garnet.berkeley.edu (Richard Petersen)
 [2]   Spelling Cookies
Followup -To: poster
Date: Mon Dec 20 04:48:01   PST   1995
Organization: University of California, Berkeley
Lines: 43
Spelling Cookies
                                        INGREDIENTS
```

1 cup flour
2 treaspoons single acting baking powder, or 1 teaspoon double acting baking powder,or 1 teaspoon baking soda
1/2 teaspoon nutmeg
1/4 teaspoon cinnamon
3/4 cup butter ( or 1 1/2 sticks)
1 cup brown or dark brown sugar
1/2 cup regular sugar
1 egg
1 teaspoon vanilla extract
1/4 cup milk
End of article 7155( of 7158) - what next ? [npq] s spellcookies
File /h/garnet_d/richpete/News/spellcookies doesn't exist --
use mailbox format ? [ynq] y
Saved to mailbox /h/garnet_d/richpete/News/spellcookies
End of article 7155(of 7158) - what next ? [npq]

**如果您把文章保存到一个已存在的文件中，则此文章将被加到文件的后面。当存到一个邮件格式的文件中，新加的文章将成为一个新的消息。当多篇文章保存于一个文件时，把此文件存为邮件格式便有诸多优点。您可以通过邮件方便的存取特定的文章。也可以用 R 命令答复文章的作者，还可以很容易地把文章合并到给别的用户的消息中。**

### 9.4.6 在 trn 中回复文章：后续文章和消息

您能通过两种方式回复一篇特定的文章：发表一篇后续文章或通过邮件送给作者一个消息。后续文章是发表在 Usenet 上的，用来回复您读过的文章，Usenet 上的其他用户都可以看到此文章。而邮件则不同，邮件是通过 Linux mail 发送的一个私人的消息。命令 f 和 F 将发表一篇文章，而 r 和 R 命令发送一个回复消息。

#### 发表后续文章

您能用 trn 新闻阅读器来发表后继文章。当阅读一篇文章时，您能使用命令 f 和 F 来发表后继文章。这两个命令都是调用 Pnews，用它输入和发表文章。F 命令将在您的后续文章中包括您回应的文章的正文。显示的正文的每行前有一个 >。

通过定位您要回复的文章并按下 f 命令，可以发表一篇后续文章。您首先会进入带有后续文章标题的编辑器中，可以在此输入文章的正文或编辑文章标题。在退出编辑器之前，将提示您是发送、忽略、编辑还是显示后续文章。用 send 命令能发表文章。

您可以看到在所在的新闻组中，后续文章如何把您带入关于一篇文章的讨论中。不仅能阅读文章，还读到别人在后续文章中对它的一些观点。正如前面所讲的，您可以对一篇后续文章发表后续，来讨论别的用户对文章的观点。发表后续文章时，只须定位到要回复的后续文章，按 f 并输入您的回复。

当您回复一篇文章时，在回复中包含文章的原文对理解会有帮助。用 F 命令可以做到这点。被包含的原文的每行前加有 >。您不用在回复文章中包含所有

的原文。假设您只对文章的一部分进行回复，您可以在编辑器中删除原文的其余部分，只保留原文中要回复的部分和回复一起发表。甚至可以用编辑器把您的回复插入到原文中，使您的回复看起来更像是注释。用命令 F 将使您的回复文章更清晰易懂且避免重复输入别人的注释。

用 F 命令来对后续文章发表回复时十分方便。它能自动加入您要回复的用户的评论，而不用您痛苦地亲自输入。

### 寄回复给作者

您可以直接给文章的作者寄一封邮件，以代替发表一篇后续文章。寄邮件时用命令 r 和 R。首先定位到您要回信的文章，按 r。显示文章标题，启动邮件程序。将进入一个编辑器中，您能输入消息。文章标题也在编辑器中，您可以修改主题或总结行。当离开编辑器时，系统会提示您是发送、中止、编辑或显示消息。send 命令将把此邮件寄给作者。

用 R 命令可以把原文包含在您的消息中。象 F 命令一样，R 命令也将把原文包含在您的消息中，其每行原文前加有>。在编辑器中，您能按需要对原文删除、复制或移动。

## 9.4.7 标记文章

一旦您阅读过一篇文章，trn 将不在文章列表中显示此文章标题，以后您也不能用 trn 来访问它。如果想以后能回到此文章，您可以把它标记为未读，使得 trn 在文章列表中继续显示此文章标题。当您阅读文章时，用命令 m 可以标记文章为未读。如果下次您想阅读此文章，只须用命令 M 即可。

### 9.4.8 文章的选择

您能选择一组文章来进行模式查找或引用序号。一个模式查找命令加一个冒号和一个 trn 命令将使得此命令作用于所有的文章，而不仅只是下一篇文章。一个模式查找加一个冒号将有效的选定一组文章，并在其中进行模式查找。例如，如果您想保存所有的主题涉及“cookies”的文章，可以输入命令/cookies/:s cookfile，如下所示。

```
End of article 6914(of 7158) - what next ? [npq] /cookies/:s
cookfile
Searching ... 7094
File /h/garnet_d/richpete/News/cookfile doesn't exist -
use mailbox format? [ynq] y
Saved to mailbox /h/garnet_d/richpete/News/cookfile
7128Appended to mailbox /h/garnet_d/richpete/News/cookfile
7155Appended to mailbox /h/garnet_d/richpete/News/cookfile
done
End of article 6914 ( of 7158) - what next ? [npq]
```

符号^是一个特殊的模式，它代表所有的文章。您能用下面的命令把所有的文章保存到一个文件中：

```
/^/:s myfile
```

用一个模式和=命令，则可以显示一列关于某一特定主题的文章。= 命令可以能列出每篇文章的序号和主题。当 = 命令和模式查找连用时，它能列出所匹配的文章的序号和主题。下面的例子将列出关于 cookies 的所有未读文章的序

号和主题。

/cookies/:=

如果您想查找已读和未读的所有文章，需要用命令 r 来限定模式。下面的命令将列出所有的关于 cookies 的文章，其中包括已读过的文章。

/cookies/r:=

缺省状态下，模式查找只搜索文章的主题。当使用 a 限定模式后，将搜索文章的全文。下例中将搜索出所有用到 chocolate 的食谱。这种查找的结果通常是太长不便于打印出来。

/chocolate/a:=

您可以通过列出文章相应的序号来引用一组文章，各序号之间用逗号分隔，最后用冒号加上一个应用到这些文章的命令。当引用的是连续的文章序号时，可以用一个范围来表示。例如，3-5 代表文章 3，4 和 5。要保存文章 34，17 和 9-12，您可以用如下命令：

34,17,9-12:s myfile

下例中，用户把文章 7155 和 7128 存入文件 goodcook 中：

7155,7128:s goodcook.
End of article 6919 (of 7158) -- what next? [npq] 7155,7128:s goodcook
7155Appended to mailbox /h/garnet_d/richpete/News/goodcook
7128Appended to mailbox /h/garnet_d/richpete/News/goodcook
End of article 6919 (of 7158) -- what next? [npq]

在模式和序号引用中，您可以扩展使用另外的命令。每个命令之间用冒号分隔。下面的例子中，文章 34，17 和 9-12 分别被保存和打印。

```
34,17,9-12:s myfile: | lp
```

### 9.4.9trn 选项

使用 trn 时有一些选项。用-n 选项可以指定您希望阅读的新闻组或新闻组的类型。用-l 选项可以列出新闻组的标题。下面的例子中,用户列出 rec.food.recipes 新闻组的标题。

```
$ trn -l -n rec.food.recipes
```

用-q 选项，可以跳过订阅阶段而直接阅读新闻组。

```
$ trn -q rec.food.recipes
```

### 9.4.10 文章列表：rn 新闻阅读器

在您的文章列表中，除了使用 trn 选择器外，还可以用 rn 命令来显示和选择项目。rn 的在新闻组列表中移动的命令和在文章列表中移动的命令是相同的。根据操作的列表的类型，您可以在新闻组或文章中移动。例如，n 命令可以使您移到下一个新闻组，同样 n 命令也可以使您移到下一篇文章。表 9-2 列出 rn 新闻阅读器的命令。

一旦您定位一个要阅读的新闻组，便可以进入文章级并选择文章来阅读。在新闻组的提示中输入 y，则进入文章级，显示新闻组的第一篇文章的文章标题。您可以阅读第一篇文章，或用文章级命令移到新闻组中的另一篇文章。文章级命令和前面描述的新闻组列表中的命令是相同的。n 命令移到下一篇文章，p 命令移到前一篇文章，^命令移到第一篇未读的文章，$命令移到最后一篇文章。

您能通过输入文章的序号而直接移到此文章。新闻组中的文章是按顺序编号的。用符号=可以得到一列未读文章的题目和序号。每行列出文章的序号、题目和主题。输入文章序号即可直接移到它。

End of article 7155( of 7158) - what next ? [npq] 7155
rec.food.rcipes(moderated) #7155 (229 more)
From : richpete@ garnet.berkeley.edu (Richard Petersen)

在文章级，您也可以使用 rn 的模式查找来定位一篇文章。命令“ /”加上一个模式，就可以向前查找文章列表中所有文章的主题域，命令？向后查找。下面的例子中，使用者在文章的主题域中查找“ cookies”。

End of article 7050( of 7158) - what next ? [npq] /cookies
Searching ...

给模式查找加一些限制，就可以指定是否要查找文章的正文、整个文章标题或您已读过的文章。用 a 限定符查找文章的正文，h 限定符查找整个文章标题，r限定符查找所有已读和未读的文章。下面的例子中，使用者在所有的已读和未读的文章中，在文章标题中查找 “ richpete ”。

End of article 6914( of 7158) - what next ? [npq] /richpete/ht
Searching ...

您还能搜索有相同主题的文章。首先定位到一篇有您想查找的主题的文章。按 CTRL-n 可以定位到下一篇有相同主题的文章。再按 CTRL-n 可以查找下一篇有相同主题的文章。Ctrl-p 能退回到前一篇有相同主题的文章。

## 9.5 发表文章：Pnews

您可以在选择的新闻组中发表文章。可能最常用的发表文章的方法是用Pnews。它会提示您关于文章标题的信息，带您进入编辑器，在编辑器中可以输入您的文章，然后会提示您是发送、编辑、保存或退出文章。

首先，在 Linux 的提示符后输入 Pnews，将提示您首先进入一个新闻组。在新闻组的提示后输入一个？，就可以列出所有的新闻组。如果您决定选择哪个新闻组，那么用命令？将便于选择。从您的系统中的 news 目录下的新闻组文件中，能随时取得关于新闻组的列表。

选定新闻组后，您将被要求指定一个分布。分布可以是存在的任意区域。您可以使所发表的文章仅供本地的用户阅读，或让文章能使全世界的用户阅读。分布还有许多中间级别，如北美、美国、某个州或城市。Pnews 将首先列出所有的称谓，在提示后输入您所需要的分布。对于美国，您可以输入 usa。

接着会要求您输入文章的题目或主题。它将用于文章的分类或模式查找。您还被问是否要发表文章。按 y，继续。接着，您将被问是否要加一个事先准备好的文件到您的文章中。通常可以事先用标准编辑器把您的文章写好并存在一个文件里。如有事先准备好的文件，就可以把此文件加到文章中，则文件的内容被读入要发表的文章中。如果您在此不输入文件名，Pnews 将自动带您进入一个编辑器，在此可以输入您的文章。

最后，Pnews 将提示您是发送、编辑、保存或显示文章。只须输入命令的第一个字符即可执行该命令。例如，如果您改变主意决定不发表此文章，可以在提示后输入 a，即不发表文章并退出 Pnews。另一方面，您输入 e 可以重新

编辑文件，以找出错误或加入新内容。输入 e 后，您将进入一个标准的编辑器，文章的内容显示在其中。修改后退出此编辑器（用 ZZ 退出 vi）即可回到 Pnews 提示。若想发表文章，只须输入 s，文章被送到 Usenet 管理器，并发表在合适的新闻组中。

如果想用标准的编辑器来输入您的文章，在提示您是否需要加入文章时，按 Enter 键。您将被要求输入一种编辑器。在括号里 Pnews 给出它使用的缺省编辑器-通常是 Vi-按 Enter 键即可使用它。接着 Pnews 会带您进入缺省的编辑器中，并在其中显示文章标题。您可以更改文章标题，更改您的主题和所在的新闻组。还可以用标准编辑命令来编辑您的文章。完成编辑后，退出编辑器（如您在 Vi 中，用 ZZ 退出）。Pnews 将提示您是发送、编辑或显示文章。您当然也可以在提示后输入 e 以重新编辑文件。完成编辑后，按 s 即可发送文章。

### 9.5.1Pnews 签名：.signature 文件

通常结束一篇文章时，您要附上相同的标准签名信息，象您的名字、Internet 地址或地址、和一个礼貌的结束语。当您的文章发表的越多时，自动地附加签名便会显地很有用。要做到这一点，您需要在主目录下创建一个名为.signature，并在其中包含您的签名信息。Pnews 将自动地把文件的内容附加在文章的结尾。您可以使用任何标准的编辑器来创建.signature 文件。

## 9.6tin 新闻阅读器

tin 新闻阅读器的操作和用 tin 很相似。然而，tin 有一个可以用于新闻阅读器和文章的屏幕选择器。当输入 tin 后，选择器将在屏幕上列出您的新闻组。您能从中选择需要的新闻器并显示其中的文章。

一组屏幕移动命令可以用于所有的屏幕，无论是新闻组、文章列表或文章正文。Ctrl-d，Ctrl-f 和空格键使您移到下一屏。Ctrl-u，Ctrl-b 和 b 使您向后移到前一屏。上箭头和 k 键能使您在屏幕中向上移一行。下箭头和 j 键使您向下移一行。命令 q 使您从一个选择器中退回到前一个。例如，如果您在文章选择器中，按 q 将退回到新闻组选择器。按 Q 将从 tin 新闻阅读器中完全退出。

tin 还有一组编辑和历史命令，可以用它们来编辑您输入的命令。Esc 将抹去一个已输入的命令,并让您重新输入。编辑命令是 Emacs 命令的子集。CTRL-d 删除一个字符，CTRL-f 和右箭头能向右移动一个字符。CTRL-b 和左箭头能向左移动一个字符。若想插入正文，只须把光标移到您想插入正文的位置，即可输入正文。tin 总保存有您输入的命令的历史记录。用 CTRL-p 可以调出您以前输入的命令，在历史的命令列中，用 CTRL-p 可以向回一个一个地移动，用 CTRL-n 可以向下移动。用 man tin，可以调出 tin 的命令帮助页，在其中，可以找到所有的 tin 命令。tin 新闻阅读器的命令都列在表 9-3 中。

当启动 tin 时，首先显示一屏新闻阅读器。在屏幕的顶端将显示“Group Selection”。在它的右边将有“h=help”。按下 h 键将调出一个帮助菜单。列出新闻组的包括一个可以识别新闻组的选择数，后面是未读的文章数和新闻组的名字。

要选择一个新闻组，必须首先移到此新闻组。若要移到此新闻组，有很多种方法。如果您要选择的新闻组显示在当前的屏幕，可以用上箭头和下箭头移到它。CTRL-d 将使您移到下一屏，CTRL-u 使您移到前一屏。也可以不用箭头键，直接输入要选新闻组的索引号。还可以用模式查找来定位一个新闻组。另外，您能用/加上一个模式以向下查找，用？可以向后查找。

一旦您定位到需要的新闻组，按 ENTER 即可显示其中的文章。一列常用的命令将显示在屏幕的底端。命令 s 将订阅一个新的新闻组，u 命令将撤消订阅。

```
Group Selection (agate.berkeley.edu 3230)h=help
        1       3       comp.ai.language
        2       1       comp.os.linux.misc
        3       7       rec.arts.movies
        4       24      rec.food.recipes
        5       32      sci.physics.fusion
        6       126     talk.politics.theory
<n>=set current to n, TAB=next unread, /=search pattern,
c)atchup,
g)oto, j=line down, k=line up, h)elp, m)ove, q)uit, r=toggle
all/unread,
s)ubscribe, S)ub pattern, u)nsubscribe, U)nsub pattern,
y)ank in/out
search forwards > rec.food
```

tin 新闻阅读器将显示新闻组中的主题和作者，前面带有一个索引号，另外如果是未读的文章，还有一个+。符号+代表所有的未读的文章。从选择器中，

您能选择想显示的文章。移到一篇文章并按 ENTER 即可选定一篇文章。新闻组和文章选择器使用许多相同的命令。上箭头和下箭头将从一篇文章移到下一篇。如果一篇文章超过一屏，您能用 CTRL-u 向后移动或 CTRL-d 向下移动一屏。还可以通过输入文章的索引号来移到一篇文章。下面便是 tin 文章选择器屏幕。

```
    rec.food.recipes(119T 124A 0K 0H R) h=help
    65     +      Fruit Salad                  Dylan Chris
    66     +      Fudge Cake                     Cecelia Petersen
    67     +      Chocolate News               Richard Leland68
           +      Apple Muffins               George Petersen
    70     +      REQUEST: romantic dinners Marylou Carrion
    71     +      REQUEST: Dehydrated Goodies Valerie Fuller
    72     +      REQUEST: Devonshire Cream     Carolyn Blacklock
    73     +      Sauces                         Bonnie Matoza
    74     +      Passion Fruit                Gabriel Matoza
    75     +      REQUEST: blackened (red)fish Ken Blacklock
    76     +      REQUEST: Cheese Toast          dylan@sf
    77     +      REQUEST: Sausage Recipes       Penny Bode
    78     +      Biscuit Recipe               gloria@ stlake
    79     +      >blackened (red)fish         augie@napa
    80     +      Oatmeal Cookies                John Gunther
    81     +      REQUEST: Potato Salad          Margaret
    82     +      REQUEST: Sesame Chicken          Frank Moitoza
    83     +      >Summer desserts             maryann@sebast
```

<n>=set current to n, TAB=next unread, /=search pattern,
^K)ill/select,
a) uther search, c)atchup, j=line down, k=line up, K=mark read,
l)ist thread,
| = pipe, m)ail, o=print, q)uit, r=toggle all/unread, s)ave,
t)ag, w=post

**常用的访问文章的命令显示在屏幕的底端。您能用/命令加一个特定的模式来查找一篇文章。a 命令允许您通过一个特定的作者来查找文章。s 命令用来保存文章。用 w 命令能在新闻组中发表您的文章。您将被提示一个标题信息，接着输入文章的正文。一旦您选择一篇文章，就可以对它发表一篇后继或回复文章。**

**当选择一篇文章后，它会被显示。如果文章显示超过一屏，按空格键能向前滚动一屏，按 b 键向后滚动。用 B 命令您能按一个特定的模式寻找文章，用 s 命令能保存文章，用 f 命令您能发表一篇后继文章，用 r 命令您能给文章的作者送个消息。**

Mon, 20 Dec 1995 04:48:01rec.food.recipesThread33 of19
Lines 43 Spelling Cookies No responses
richpete@garnet.berkeley.edu. Richard Petersen at University of California

Spelling Cookies

INGREDIENTS

1 cup flour
2 treaspoons single acting baking powder, or 1 teaspoon double acting baking powder,or 1 teaspoon baking soda

1/2 teaspoon nutmeg
1/4 teaspoon cinnamon
3/4 cup butter ( or 1 1/2 sticks)
1 cup brown or dark brown sugar
1/2 cup regular sugar
1 egg
1 teaspoon vanilla extract
1/4 cup milk
<n>=set current to n, TAB=next unread, /=search pattern,^K)ill/select,
a) uther search, c)atchup, j=line down, k=line up, K=mark read,
| = pipe, m)ail, o=print, q)uit, r=toggle all/unread, s)ave, t)ag, w=post

-- More - (44%) [46/100]

通过移动到文章或新闻组的项并按+，您可以自动的选择它们。在自动地选择这些项之前将会出现一个*号。这就是所谓的热点。在大多数的 tin 命令象 same 或 mail 命令中，都涉及到自动选择项。

## 9.7 二进制编码：uuencode 和 uudecode

某些新闻组指定包含二进制编码的文件。在这些新闻组中，一些文章包含二进制编码的文件，象 jpeg 图象。新闻组和邮件系统并不能处理二进制文件，它们只能处理字符文件。然而，您可以把二进制文件转换成字符格式并把它们发表。为了做到这点，就需要用到 uuencode 和 uudecode 程序。uuencode 用

来把二进制文件转换成字符文件。接着，您可以把此字符文件发表到新闻组中，或通过电子邮件寄给其他用户。uudecode 程序用来翻译用 uuencode 编码的文件。您能在新闻组中保存了一个 uuencode 文件后，然后用 uudecode 把它再翻译二进制文件。参见第 8 章对这两个程序的讨论。

指定包含二进制文件的新闻组的名字中包含“ binaries ”。例如，alt.binaries.pictures 便包含用 uuencode 编码的 jpeg 或 gif 图片。alt.binaries.multimedia 中包含用 uuencode 编码的 mpeg 或 mov 文件。二进制的文件被编码发表后，经常分成几个部分。若要把它们组成二进制文件，您需要下载所有的部分，用 uudecode 去解码混合成二进制文件。

## 9.8 总结：Usenet 和新闻阅读器

Usenet 被认为是一种在线的电子新闻服务，它包括期刊文章、最新的电子公告牌和对不同主题的讨论。Usenet 按主题的不同分成不同的新闻组。您可以访问新闻组并阅读其中的文章。也可以在一个特定的新闻组中组成和发表您的文章。还可以通过两种方式回复一篇文章，在新闻组发表您的回复以供大家阅读，或直接发送您的回复消息给文章的作者。

为了访问 Usenet 中的文章，您需要使用一种新闻阅读器程序。两个最流行的新闻阅读器为 trn 和 tin。trn 允许你使用模式查找搜索新闻组和文章，也允许您复制文章和发表自己的文章。trn 能够区分已读和未读的文章，使您很容易的访问新闻组中新发表的文章。trn 使用一种称为是选择器的界面来列出按线索分

组的文章，线索使您能通过主题或相关的后续文章来引用文章。您可以选择一组想检查的文章，接着移到它们相关的文章。当用线索显示文章时，您能用到线索树。线索树是用来标明文章和后续文章之间的关系。用线索树，可以在树中从一篇文章移到别的文章。

tin 对新闻组和新闻组中的文章都使用选择器。它和 trn 共享许多相同特征。通过选择器可以很容易的访问和引用新闻组。

表 9-1trn 命令

| 选项 | 功能/描述 |
| --- | --- |
| -c | 检查所有的未读新闻组 |
| -r | 重新启动前一个新闻组 |
| -q | 启动时跳过新闻组选择 |
| -0 | mode sort-order |
| Mode | |
| a | 文章模式 |
| s | 主题模式 |
| t | 线索模式 |
| Sort-order | |
| d | 日期 |
| s | 主题 |
| a | 作者 |
| c | 文章数 |

| | |
|---|---|
| g | 主题-日期分组 |
| 选择新闻组 | |
| y | 选择当前的新闻组 |
| n | 移到下一个带有未读文章的新闻组 |
| N | 移到下一个新闻组 |
| p | 移到前一个带有未读文章的新闻组 |
| P | 移到前一个新闻组 |
| - | 移到前一个被选择的新闻组 |
| ^ | 移到第一个带有未读文章的新闻组 |
| num | 移到序号为 num 的新闻组 |
| $ | 移到最后一个新闻组 |
| g 新闻组名 | 移到为此名的新闻组 |
| /模式 | 向前查找有此模式的新闻组 |
| ?模式 | 向后查找有此模式的新闻组 |
| L | 列出订阅的新闻组 |
| l 模式 | 列出未订阅的新闻组 |
| u 新闻组名 | 取消新闻组的订阅 |
| a 新闻组名 | 订阅一个新的新闻组 |
| c | 标记新闻组中的文章为已读 |
| 显示选择器 | |
| + | 从 trn 提示行中进入选择器或离开选择器回到 trn 提示行 |

| | |
|---|---|
| | **选择选择器的模式：主题，线索或文章** |
| | Selector Mode: Threads, Subjects, Articles? [tsa] |
| S | s **主题模式，通过主题显示文章** |
| | a **文章模式，每篇文章单独显示** |
| | t **线索模式，通过线索显示文章** |
| = | **在文章选择器和主题/线索选择器中切换** |
| O | Subject Order by Date, Subject, or Count? [dscDSC] |
| | Thread Order by Date, Subject, Author, subject-date Groups? |
| L | **选择器中的项以短格式，中等格式或长格式显示** |
| E | **高级模式；只显示选中的文章** |
| k | **从选择器的显示中删除一篇文章或主题** |
| U | **显示未读的文章** |
| **在选择器中移动** | |
| SPACEBAR | **显示文章的下一屏** |
| > | **显示线索的下一屏** |
| < | **显示文章的上一屏** |
| $ | **显示文章的最后一屏** |
| ^ | **显示文章的第一屏** |

**文章**

| | |
|---|---|
| id | 选定/撤消选定一篇文章的线索 |
| id* | 选定/撤消选定带有相同主题 ID 的文章 |
| n | 移到下一个线索 ID |
| p | 移到前一个线索 ID |
| z | 开始显示选中的文章，显示结束后返回到新闻组 |
| x | 开始显示选中的文章，显示结束后移到下一个新闻组向下查找每篇文章的主题域中是否有此模式 |
| /模式 | 修改参数<br>h 向下查找文章标题中是否包含模式<br>/模式/b<br>a 向下查找文章标题或文章正文中是否包含模式<br>/模式/a<br>r 查找中包括已读文章<br>/模式/r<br>c 让查找更敏感<br>/模式/c |
| ?模式 | 向上查找每篇文章的主题域中是否有此模式修改参数 h 向上查找文章标题中是否包含模式?模式?h a 向上查找文章标题或文章正文中是否包含模式 r 查找中包括已读文章 c 让查找更敏感 |
| / | 重复前一个向下查找 |
| ? | 重复前一个向上查找 |

| | |
|---|---|
| /模式:命令 | 选择一组匹配模式的文章并把命令用于它们选项功能/描述 |
| id,id:命令 | 选择一组被序号引用的文章并把命令用于它们显示文章 |
| SPACEBAR | 显示文章的下一屏 |
| ENTER | 滚动到文章的下一行 |
| d | 滚动到文章的下半屏 |
| b | 显示文章的上一屏 |
| v | 从开头从新显示文章 |
| q | 显示文章的最后一屏 |
| g 模式 | 在文章中查找模式 |
| g | 在文章中重复模式查找 |
| 保存文章 | |
| :w | 保存选中的文章 |
| :s | 把选中的文章保存到一个邮箱文件中回复文章 |
| r | 回复当前的文章 |
| R | 回复当前的文章并在回文中包括原文 |
| f | 对当前的文章发表一篇后继文章 |
| F | 对当前的文章发表一篇后继文章并在文章中包括原文 |
| 标记文章 | |
| m | 把当前文章标为已读 |
| n | 把当前文章标为已读并移到下一篇文章 |
| j | 把当前文章标为已读并显示文章的结尾 |

| | |
|---|---|
| c | 在当前的新闻组中把所有的文章标为已读 |
| trn 选项参数 | |
| EDITOR | 组织回复文章的编辑器 |
| MAILPOSTER | 发送回复的邮件功能 |
| PAGER | 阅读文章的分页功能 |
| SAVEDIR | 保存文章的目录 |
| NAME | 您发表的文章的标题中您的全名 |
| ORGANIZATION | 您发表的文章的标题中您的组织名 |
| NNTPSERVER | NNTP 远程新闻服务器地址 |

表 9-2 可用于 trn 的 rn 文章列表命令

| 选择文章 | 功能 |
|---|---|
| y | 显示当前的文章 |
| n | 移到下一篇未读的文章 |
| N | 移到下一篇文章 |
| p | 移到上一篇未读的文章 |
| P | 移到上一篇文章 |
| - | 移到上一篇选中的文章 |
| ^ | 移到第一篇未读的文章 |
| num | 移到序号为 num 的文章 |
| $ | 移到最后一篇文章 |
| CTRL-n | 移到下一篇和当前文章有相同主题的文章 |

| | |
|---|---|
| CTRL-p | 移到上一篇和当前文章有相同主题的文章 |

表 9-3tin 新闻阅读器命令

| 屏幕移动命令 | 功效/描述 |
|---|---|
| DOWN 箭头, j | 向下移动一行 |
| UP 箭头, k | 向上移动一行 |
| $ | 到最后一行 |
| CTRL-U,CTRL-B, b,PAGE UP | 到上一屏 |
| CTRL-D,CTRL-F, SPACEBAR,PAGE DOWN | 到下一屏 |
| CTRL-L | 刷新屏幕 |
| q | 退回到上一级 |
| Q | 退出 tin |
| **编辑命令和历史命令** | |
| CTRL-f, RIGHT 箭头 | 移到下一个字符 |
| CTRL-b, LEFT | 移到上一个字符 |
| CTRL-D, BACKSPACE,DEL | 删除字符 |
| CTRL-p | 历史记录中上一个命令 |
| CTRL-N | 历史记录中下一个命令 |
| ESC | 抹去输入的命令 |
| **新闻组选择器** | |

| | |
|---|---|
| num | 移到索引号为 num 的新闻组 |
| ENTER | 选中当前的新闻组 |
| TAB | 移到下一个未读的新闻组 |
| / | 向下查找 |
| ? | 向上查找 |
| g | 按名字选择一个新的组 |
| K | 把文章/线索标为已读并移到下一篇未读文章/线索 |
| l | 按当前的线索列出文章 |
| C | 把所有的文章标为已读并移到下一个未读的组中 |
| c | 把所有的文章标为已读并移到选择组的菜单 |
| s | 订阅一个新闻组 |
| u | 撤消对一个新闻组的订阅 |
| S 模式 | 订阅包含模式的新闻组 |
| U 模式 | 撤消对包含模式的新闻组的订阅 |
| M | 显示配置选项菜单 |
| v | 显示版本信息 |
| h | 帮助命令 |
| **屏幕移动命令功效/描述** | |
| CTRL-K | 删除/自动选择当前新闻组 |
| H | 切换到迷你帮助菜单 |

| | |
|---|---|
| l | 切换为反转的视频信号 |
| **文章选择器** | |
| num | 移到索引号为 num 的文章 |
| $ | 移到最后一篇文章 |
| ENTER | 选择当前的文章 |
| TAB | 移到下一篇未读的文章 |
| a | 向下查找作者 |
| A | 向上查找作者 |
| / | 向下查找主题 |
| ? | 向上查找主题 |
| n | 移到下一组 |
| p | 移到上一组 |
| N | 移到下一篇未读的文章 |
| P | 移到上一篇未读的文章 |
| d | 切换到显示主题或主题和作者 |
| r | 切换显示所有文章/只是未读文章 |
| u | 切换显示非线索文章/线索文章 |
| z | 把文章标为未读 |
| X | 标记所有未被选中阅读的未读文章 |
| t | 标识文章未正交叉发表/邮寄/用管道传送/打印/保存 |

| | |
|---|---|
| U | 撤消所有文章的标识 |
| s | 保存文章/线索/热点/模式/标识的文章到文件中 |
| m | 把文章/线索/热点/模式/标识的文章邮寄给某人 |
| o | 把文章/线索/热点/模式/标识的文章输出到打印机 |
| w | 在当前的组中发表一篇文章 |
| W | 列出某用户发表的文章 |
| x | 交叉发表当前的文章到另外的组中 |
| * | 选择线索 |
| . | 切换线索的选择 |
| @ | 反向所有的选择，即把选中的改为未选，未选的改为 |
| 选中（对所有的文章） | |
| ~ | 撤消选择（对所有文章） |
| + | 在组或文章中执行自动选择，以创造热点；在选择的项前面将有一个* |
| = | 如果至少有一个未读的文章被选中，则标记选择的线索 |
| ! | 退出 shell |
| - | 显示最后一则消息 |

| | |
|---|---|
| | 用管道输送文章/线索/热点/模式/标识的文章到命令中 |
| 显示文章 | |
| b | 向上移动一页 |
| SPACEBAR | 向下移动一页 |
| 屏幕移动命令功效/描述 | |
| B | 查找文章正文体 |
| ; | 如果至少有一个未读的文章被选中，则标记选择的线索 |
| s | 保存文章/线索/热点/模式/标识的文章到文件中 |
| m | 把文章/线索/热点/模式/标识的文章邮寄给某人 |
| o | 把文章/线索/热点/模式/标识的文章输出到打印机 |
| w | 在当前的组中发表文章 |
| f | 在当前的组中发表后继文章 |
| r | 给文章的作者发送一篇回复 |
| tin 选项参数 | |
| VISUAL | 组织回复文章的编辑器 |
| NNTPSERVER | NNTP 远程新闻服务器地址 |

# 第 10 章 Internet 工具

Internet 是一个遍布全球的计算机网络，您可以通过 Internet 地址和一组 Internet 工具来访问它。在 Internet 上许多计算机被配置为服务器，用来给用户提供信息。允许访问和复制的信息都存放在文件里。每个服务器，通常被称为站点，都有自己的 Internet 地址，通过地址您能够在 Internet 上找到它们。Linux 提供一组 Internet 工具，通过它们您能够访问 Internet 上的站点，并从站点上找到和下载所需要的信息。

为了访问 Internet 站点，您的计算机必须连接 Internet 上。您的计算机可能已经连接到一个 Internet 的网络中。如果您使用的是一台单独的计算机，例如个人电脑，您必须从 Internet 服务提供商（ISP）那里连接到 Internet。一旦拥有一个自己的 Internet 地址，就可以把您的 Linux 系统配置连接到 Internet 并使用大量的 Internet 工具访问 Internet 站点。第 12 章描述了如何配置您的 Linux 系统以建立一个连接。

Internet 工具 telnet 和 ftp 允许您连接到另外的 Internet 站点。telnet 执行一个远程注册，它可以注册到连接在 Internet 上的另一台计算机。例如，您可以用它来搜索国会图书馆的在线目录。ftp 能连接到一个站点，并允许您的系统和已连接的站点之间进行文件传输-下载或上传。用 ftp 您可以连接到一个有 Linux 软件的站点并直接把软件下载到您的计算机中。

找出什么是有用的信息，有用的信息位于什么系统中，以及存放在系统的什么地方是压倒一切的任务。通常您需要事先知道文件的位置。然而，有两个

Internet 工具，Archie 和 Gopher 使您能在网络上搜索文件。用 Archie 就像是在在线目录中查找，如同您用关键字在书中查找一个标题，使用 Archie 还可以用模式查找来搜索一个文件名。Gopher 的操作更像一个图书管理员。它提供给您一系列不同主题的菜单。您可以从一个菜单移到另一个，逐渐缩小您的查找范围直到找到所需的信息。用 Gopher，您能直接传输信息而不用再借助于 ftp。

最近几年，Web 浏览器已经变成访问 Internet 信息的最主要的工具。然而，Web 浏览器仍是依靠基本的 Internet 工具来寻找和传输信息。telnet，ftp，Archie 和 Gopher 都是用来定位和访问 Internet 站点的工具，用它们还可以检索这些站点上的信息。Web 浏览器使用这些工具来取得 Internet 上的信息，但是 Web 浏览器对用户隐蔽了这些工具，它使得交互界面更友好。通过 Web 浏览器，在 Internet 上执行任务更容易。您可能需要用本章中描述的 Internet 工具来完成大多数的任务，像从一个特定的 ftp 站点上下载文件。例如，如果您需要从一个 Linux 的 ftp 站点上下载一个 Linux 软件程序，您就可以使用 ftp 来完成它。

## 10.1Internet 地址

Internet 使用一组网络协议，称为 TCP/IP-传输控制协议/Internet 协议。在一个 TCP/IP 网络中，消息被分成一些小的组件-数据报，然后通过大量的路由器把数据报传输和交付到它们的目的计算机。一旦收到这些数据报，它们就又被重组成原始信息。数据报也被称为数据包。已经证明，把数据分成小组件来发送的方式，比把数据作为整体进行一个巨大的传输更有效也更快捷。用小组

件发送时，如果其中任何一个组件丢失或被破坏，只须把此组件重传。而作为整体进行传输时，任何一部分被破坏或丢失，整个信息都要重传。

在像 Internet 这样的 TCP/IP 的网络中，每台计算机都被给定一个特有的地址叫做 IP 地址。IP 地址是用来标识和定位一个网络中的特定的主机-一台连在网络上的计算机。一个 IP 地址包含四个段，每段之间用一个圆点分隔。每段包含一个从 0 到 255 的数字，某些值被保留做特殊的用途。IP 地址被分成两个部分，一部分用做标识网络，另一部分标识特定的主机。每段的数值是由网络的分类来决定的。在 Internet 上，网络按大小分成三类-A 类，B 类和 C 类。一个 A 类的网络地址中，只有第一段用做 IP 地址，其余三段均用做主机地址，所以它允许大量的计算机连接到此类网络中。大多数的 IP 地址都是容量要小的多的 C 类地址。

C 类地址中的前三段用来标识网络，只有最后一段用来标识主机。它的地址的语法为：

net.net.net.host

在一个 C 类网络地址中，前三个数是 IP 地址中的网络标识部分。此部分包含三个网络数，每一个用来标识一个子网。在 Internet 上无论是最大的网络或最小的网络都被组成一个子网。最后一个数用来标识一个特定的计算机-也被称为主机。您可以把 Internet 看成是包含一系列子网的网络，它的子网还包含自己的子网。网络地址中最右边的数标识宿主计算机，它前面的数标识计算机所属的子网。此数左面的数标识子网所属的网络，依次类推。Internet 地址 192.18.187.4 代表第四个连接到以 187 标识的网络中的计算机。网络 187 是一个标识为 18 的网络的子网。同样网络 18 是网络 192 的子网。下面表示如何分

解 IP 地址：

192.18.187.4 IP 地址

192.18.187 网络标识

4 主机标识

IP 地址是由管理 Internet 的网络信息中心（NIC）提供的。您可以从 NIC 那里得到一个 Internet 地址，或者如果您在一个已连接到 Internet 的网络中，网络管理员可以给您分配一个地址。如果使用 Internet 服务提供商，ISP 可能给您得到一个地址，或每次您连接时，临时从它们现有的地址池中取得一个分配给您。

某些数字是保留的。127，0，255 便不能作为正式的 IP 地址的一部分。地址 127.0.0.0 是一个回送地址，用于您的计算机上的用户之间互相通信。数 255 是特殊的广播标识，用它可以给网络中所有站点广播消息。在 IP 地址中使用 255 则涉及到连接到同一级别的所有站点。例如，192.18.255.255 将广播一则消息给网络 192.18 中的所有计算机，给它的子网和子网中所有的主机。地址 192.18.187.255 则广播给本地网络中的每台计算机。如果在 IP 地址中的网络部分使用 0，则主机数将涉及您本地网中的计算机。例如，0.0.0.6 涉及您本地网中的第六台计算机。如果您想给本地网中所有计算机广播消息，您可以使用地址 0.0.0.255。Internet 上的所有主机通过 IP 地址来标识。当给 Internet 上的一台主机发送消息时，您必须提供它的 IP 地址。然而，使用四个数字的 IP 地址很不方便，它不便于记忆，而且输入时容易出错。为了更容易的标识 Internet 上的计算机，出现了一种域名服务（DNS）。DNS 为每个 IP 地址建立一个域名地址。域名地址是一系列用圆点分隔的名字。当您使用域名地址时，它都被

自动转换成标识 Internet 主机的 IP 地址。域名地址远比 IP 地址易于记忆和使用。

一个域名地址需要在 NIC 登记注册，以保证 Internet 上的每台计算机都有一个唯一的名字。创建域名时必须遵从一个特定的命名习惯，正如第 8 章中讨论的那样。域名地址中包含您给计算机起的主机名，一个标识您的网络的域名，和一个标识您所在的网络的扩展名。下面是一个域名地址的语法：

host-name.domain-name.extension

下面的例子是一个叫 sunsite 的计算机的域名，它处在一个叫做 unc 的网络中，它又是教育机构的一部分，所以加有扩展名 edu。

sunsite.unc.edu

过去从域名地址到 IP 地址的转换通常是由个人主机完成的。当这种转换只限于为数不多的已知 IP 地址到域名地址的转换时，个人主机还可以承担此项工作。然而，当连接在 Internet 上的主机越来越多时，地址转换的工作便需要由特定的服务器来完成，这种服务器就是大家所熟知的域名服务器或被称为名字服务器。一个名字服务器拥有一个的数据库，数据库中包含一些域名地址和它们相应的 IP 地址。本地局域网中有时也有自己的名字服务器。如果一个名字服务器找不到一个地址，它会通过别的名字服务器来执行查找转换。您的计算机上的一个叫做分解器的程序从名字服务器中取得 IP 地址，并把地址用到那些使用域名地址的应用中。

用 whois 和 nslookup 命令，您能取得连接到 Internet 上的不同的网络和主机的域名服务器的信息。输入 whois 和主机或网络的域名，whois 将显示关于主机的信息，象联系人的通信地址和电话号码。

```
$whois  domain-address
```

用 nslookup 命令加上域名地址，将会找到域名相对应的 IP 地址。

$nslookup domain-address

nslookup 有一种交互的模式，通过这种模式，在输入时，您可以不必指定域名。用 nslookup，还可以查找一个主机的其他信息。例如，用 HINFO 选项可以找到主机所使用的操作系统。在 nslookup 的帮助页中列出各种选项及响应的用法。

## 10.2 远程注册：telnet

您可以使用 telnet 远程注册到网络中的另一个系统中。此系统可以是位于您的局域网中或是通过 Internet 可以连接的。telnet 的操作就像从一个远程终端注册到另一个系统中。您将被要求一个注册名，某些情况下还要输入口令。然后，您就通过一个帐号注册进入另一个系统。如果您拥有一个系统中的有效帐号，就可以通过 telnet 注册进入该系统。

通常，telnet 是用来连接 Internet 上提供公共信息服务的站点，例如国会图书馆和它的的在线目录，提供 Archie 索引的站点等。这些站点允许客户机注册，也就是它们说不需要特殊的注册名和口令，任何人都可以注册。然而，这些站点对公共的访问做了一些专门的设计，提供给您一些可以访问的选项菜单，也控制您对系统的访问。

您可以通过关键字 telnet 来直接启动 telnet。如果您知道要连接的站点名，可以在 Linux 命令行中输入 telnet 后，再直接输入您要连接的站点名。下面的

例子中，用户指定连接到 garnet 系统。

```
$telnet garnet.berkeley.edu
Connected to garnet
login:
```

telnet 有一个命令模式，在此模式下，可以使用一系列命令来配置您的连接。直接输入 telnet 来调用 telnet，或是在对话期间按下 CTRL-]，就可以进入 telnet 命令模式。telnet 中的 help 命令将列出所有可用的 telnet 命令。在表 10-1 中列出常用的 telnet 命令，在帮助页（man telnet）将更详细所有命令。在下面的例子中，用户首先调用 telnet。接着出现一个提示符：telnet>，代表进入命令模式。用命令 open 即可连接到另一个系统。

```
$telnet
telnet> open garnet.berkeley.edu
Connected to garnet.berkeley.edu
login:
```

一旦建立连接后，您需要进行一个注册的过程。如果注册到一个常规的系统，还需要提供注册名和口令。完成注册后，将会出现一个操作系统提示符，若是 Linux 或 Unix 系统，将出现$或%。现在您已直接连接到此系统中，可以输入您需要的操作命令。

完成您的工作后，退出系统。将中断连接并返回您自己系统中的 telnet 提示符。用 quit 命令可以退出 telnet。

```
telnet>quit
```

当您使用 telnet 连接到一个提供公共访问的站点时，就不需要提供注册名和口令。但是这种访问被一系列菜单控制，这些菜单也限制您在系统中的操作。

图 10-1 给出用 telnet 访问国会图书馆的在线目录站点 locis.loc.gov 时出现的对话窗口。

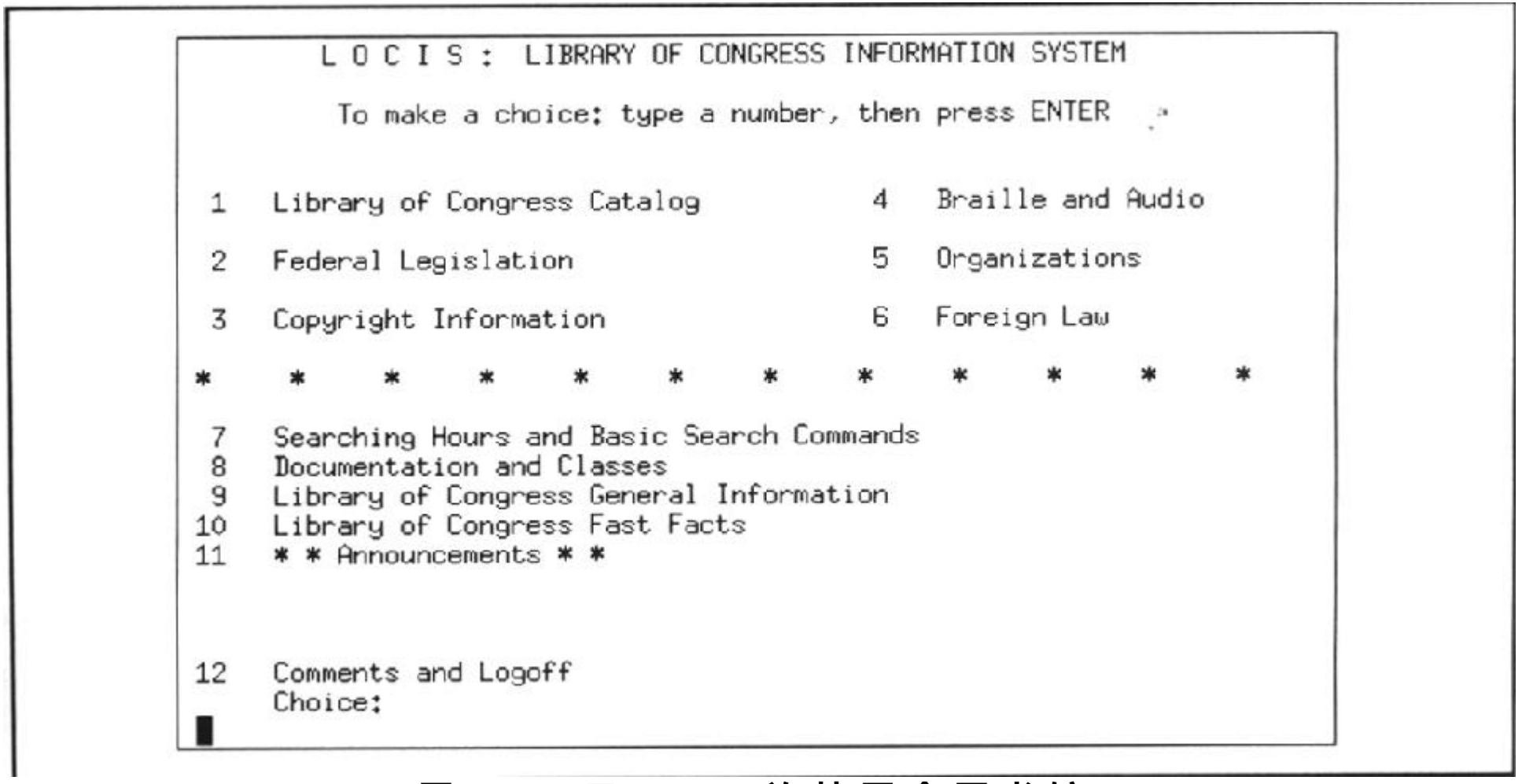


图 10-1 用 telnet 连接国会图书馆

$telnet locis.loc.gov

如果您用一个特定的帐号注册到一个系统，您可以用 -l 选项来指定此注册

名。它将使您跳过 login 的提示符。您可以在 telnet 命令行中或在 open 命令中使用 -l 选项，如下例所示。在此，使用者用一个叫做 richpete 的帐号注册进入系统 rose.berkeley.edu 中。

```
$ telnet rose.berkeley.edu -l richpete
telnet> open rose.berkeley.edu -l richpete
```

使用某些在线软件，象 Gopher 或 Web 浏览器，连接到一些公共站点时将自动调用 telnet。有些时候，连接到一个站点时，您会发现出现的是一些简单的菜单选项，而不是通常的 Web 浏览器的图形界面或 Gopher 的显示页。此时，您已经通过 telnet 远程连接到另一站点，显示的是站点界面，此站点用此界面来控制对系统资源的访问。您只须按照菜单提示进行操作。当离开此站点后，您将返回到您的 Web 浏览器或 Gopher 页。

## 10.3 网络文件传输：ftp

一个最常用的 Internet 工具是 ftp。您可以使用 ftp 从一个站点直接传输一个很大的文件到另一个站点。它可以操作文本和二进制的文件。ftp 的全称为文件传输协议。它可以在基于 TCP/IP 的网络系统中执行，象 Internet。在表 10-2 中列出 ftp 的一组命令，您可以使用这些命令来管理文件的传输。

ftp 首先用一个帐号注册到连接在网络或 Internet 中的另一个系统。一旦注册入另一个系统，您可以上传或下载文件。要想注册到系统中，您需要知道远程系统的一个帐号的注册名和口令。例如，如果您有 Internet 上两个站点的帐

号，您就可以使用 ftp 从一个站点向另一个站点传输文件。然而，在 Internet 上有许多站点允许使用 ftp 进行公共访问。许多站点中存放很多大的文件以供任何人访问和下载。这些站点被称为 ftp 站点，它们的 Internet 地址通常是以 ftp 开头。sunsite.unc.edu 便是一个存放 Linux 文件的 ftp 站点。这些 ftp 站点允许任何用户使用匿名的 ftp 注册。匿名注册时使用“anonymous”作为注册名，用您的 Internet 地址作为口令，就可以注册到这些系统中并从系统传输文件到您自己的系统。

您可以通过命令 ftp 来直接启动 ftp。如果要连接到一个特定的的站点，您可以在命令行中的 ftp 后直接输入站点名。否则，要使用 ftp 的 open 命令来连接远程的站点。您会被提示输入远程站点名。输入站点名后，ftp 连接到此系统，提示输入注册名。在提示中包括“Name”和包含在括号中的系统名和您的本地注册名。有时远程系统中的注册名和您的本地注册名一致，您可以在提示注册名时按 ENTER 键。如果注册名不一致，输入在远程系统中的注册名。输入后，将提示您输入口令。下面的例子是以 robert 帐号注册到远程系统 garnet 中。

```
$ ftp
ftp> open
(to) garnet
Connected to garnet.berkeley.edu.
220 garnet.berkeley.edu  FTP server (ULTRIX Version 4.1  Sun May 16
10:23:46  EDT 1996) ready.
Name (garnet.berkeley.edu:root): robert
password required
Password:
```

user robert logged in

ftp>

为了节省一些步骤，可以在调用 ftp 时，直接在命令行中指定远程系统。这样的话便不需要用 open 命令，直接开始注册过程。

$ftp garnet.berkeley.edu

Connected to garnet.berkeley.edu.

220 garnet.berkeley.edu FTP server (ULTRIX Version 4.1 Sun May 16 10:23:46 EDT 1996) ready.

Name (garnet.berkeley.edu:root):

为了连接到一个公共的 ftp 站点，您要执行一个匿名注册。在提示注册名时，输入 anonymous。在提示输入口令时，输入您的 Internet 地址。一旦出现 ftp 提示符，就可以开始进行文件传输。您首先要切换到所需的目录下或是转换成二进制模式。下面的例子是一个完整的 ftp 过程，用户用匿名注册，然后下载一个 Archie 软件包。访问的 ftp 站点是 sunsite.unc.edu。在切换到目录 /pub/Linux/system/Network/info-systems 后，使用 get 命令下载 Archie 软件。

$ftp sunsite.unc.edu

Connected to fddisunsite.oit.unc.edu.

220 helios FTP server (Version wu-2.4(39) Tue May 16 01:34:21 EDT 1996) ready.

Name (sunsite.unc.edu:root): anonymous

331 Guest login ok,send your complete e-mail address as password.

password:

230 Guest login ok,access restrictions apply.

Remote system type is UNIX.

```
Using binary mode to transfer files.
ftp> cd /pub/Linux/system/network/info-systems
250 CWD command successful.
ftp> 1s
200 PORT command successful.
150 Opening ASCII mode data connection for /bin/1s.
total 1130
drwxrwxr-x4 671002512 Jan16      02：54 .
drwxr-xr-x20 6710021024 Sep5     1995 ..
drwxr-xr-x2 671002512 Jul21      1994 .cap
-rw-r--r--1 671002570 Jan16      02：55 INDEX
-rw-r--r--1 671002586 Apr8       1995
 archie-1.4.1.1inux.1sm
-rw-r--r--1 67100224335 Apr9     1995
 archie-1.4.1.1inux.tar.gz
-rw-r--r--1 671002109147 Sep25   1995
 archie-1.4.1.src.tar.gz
226 Transfer complete.
ftp> binary
200 Type set to I.
ftp> get archie-1.4.1.1inux.tar.gz
200 PORT command successful.
150 Opening BINZRY mode data connection for
 archie-1.4.1.1inux.tar.gz (24335 bytes).
226 Transfer complete.
```

24335 bytes received in 7.87 secs (3 Kbytes/sec)
ftp> close
221 Good-bye.
ftp> quit

一旦您注册进入，便可以在远程系统或本地系统中执行 Linux 命令。若想在 ftp 中对本地系统执行命令，您可以在命令前加一个感叹号。前面没有感叹号的命令将在远程执行。在下面的例子中，第一个命令列出远程系统中的文件，第二个命令列出本地系统中的文件。

ftp>ls
ftp>!ls

此规则有一个例外，当在远程系统中切换目录时用命令 cd，而在本地系统中切换目录时，您需要使用一个特殊的命令 lcd。在下面的例子中，第一个命令在远程系统中切换目录，第二个命令在本地系统中的切换目录。

ftp>cd
ftp>lcd

您可以用 close 命令关闭对一个系统的连接。如果需要，可以再打开另一个连接。用命令 quit 或 bye 可以结束 ftp 对话窗口。

ftp>close
ftp>bye
Good-bye
$

从远程系统中下载文件，或者上传文件到远程系统中，可以使用命令 get 和 put。get 命令能从远程系统下载文件到您本地系统中。put 命令可以把文件

从您的系统上传到远程系统。从某种意义上说，是您的本地系统从远程系统 get 文件和 put 文件到远程系统中。下面的例子中，便是用 put 命令把本地系统中的 weather 文件上传到远程系统中。

```
ftp> put weather
PORT command successful.
ASCII data connection
ASCII Transfer complete
ftp>
```

传输文件时，可以使用字符模式或二进制模式。缺省的模式时字符模式。命令 ascii 可以设置到字符模式，命令 binary 可以设置到二进制模式。如果您要传输程序或压缩文件，首先要确保传输模式是二进制模式。若程序是一个二进制的文件，则必须在二进制模式下传输它。在 Internet 站点上大多数的软件包都是存档文件和压缩文件。您需要在二进制模式下下载这些文件。下例中，首先把传输模式设置成二进制模式，接着用 get 命令把存档的软件包 life.tar.gz 从远程系统下载到您的本地系统。

```
ftp>binary
ftp>get life.tar.gz
PORT command successful
Binary data connection
Binary Transfer complete
ftp>
```

您通常需要传送几个文件，并用特定的字符指定文件名。然而，put 和 get

命令一次只能操作一个文件并且不能使用特定的字符。为了一次传送多个文件，您需要使用另外两个命令,mput 和 mget。当使用 mput 和 mget 时，您将被提示输入一列文件。您可以输入一列文件名或用特定字符指定一个文件列。例如，*.c 将代表所有扩展名为.c 的文件，*将代表当前目录下的所有文件。使用 mget 时，文件将被逐个的从远程系统传送到您的本地系统。每传送一个文件，系统将提示将被传送的文件名。您输入 y 来传送文件或输入 n 来取消传送。接着您将被提示下一篇文章。使用 mput 时也一样，只不过是把文件从您的本地系统传送到远程系统。下面的例子中，用 mget 把所有扩展名为.c 的文件传送到您的本地系统。

```
ftp> mget
(remote-file) *.c
mget calc.c ? y
PORT command successful
ASCII data connection
ASCII transfer complete
mget main.c ? y
PORT command successful
ASCII data connection
ASCII transfer complete
ftp>
```

# 10.4Archie

在 Internet 上有大量的站点对公共访问开放。包含在它们上的文件对公众开放，任何人都可以使用诸如 ftp 这样的文件传输程序来取得这些文件。然而，除非您已得知文件的位置，否则从无数个站点中找到文件将是十分困难的事情。Archie 便是专为查找文件而设计的，它可以告诉您查到的文件的位置。一旦您知道站点，便可以用 ftp 来访问它并从中下载文件。您可以把 Archie 看成是不同 Internet 站点上所有可用文件的在线索引。

Archie 中有一个数据库，库中存放所有的文件名和它们所在的站点，此数据库每月更新一次。在不同的 Archie 站点上有此数据库的备份，这些站点都作为 Archie 服务器。您可以查询这些站点以寻找某一文件名，Archie 服务器将给出查找结果，列出不同的文件和它们存放的站点。

您可以通过 telnet 对话窗口，或是通过安装在您的系统中的 Archie 客户程序来访问 Archie 服务器。一个 Archie 客户程序将自动地访问 Archie 服务器，执行您的查询并得到结果。因为对 Archie 潜在的需求，所以鼓励用户使用 Archie 客户程序，而不是用交互的注册使用。然而，如果您没有 Archie 客户程序，就不得不使用 telnet 注册到 Archie 服务器并直接在服务器上执行您的查询。根据需要，Archie 服务器可以限制同时访问或查询服务器的用户数。表 10-3 列出了 Archie 客户程序的选项和 Archie 服务器可以使用的命令。

Archie 客户程序不包括在您的 Caldera Network Desktop 中。然而，您可以从 ftp 站点 ftp.mcgill.ca 或 sunsite.unc.edu 和它的镜像站点上下载。在站点 sunsite.unc.edu 上，Archie 位于目录/pub/Linux/system/Network/info-systems

中。当前的 Archie 软件包叫做 aarchie-1.4.1.linux.tar.gz。下载后，使用 gunzip 和 tar xvf 命令展开它。展开过程中将新建一个包含源代码的目录/archie-1.4.1（前面的 ftp 例子演示了如何下载一个 Archie 客户程序）。转换到此目录下，输入命令 make，您就可以编译您的 Archie 客户程序。然后产生一个 Archie 客户程序叫做 archie。接着您可以把此程序复制到目录/bin 或/usr/local/bin 下。在文件 INSTALL 里有关于如何创建 Archie 客户程序的更详细的说明。

### 10.4.1Archie 客户程序

如果您的系统中有 Archie 客户程序，通过输入命令 archie 加上选项和一个模式，您可以执行 Archie。模式是用来寻找和匹配文件名。这有一个 Archie 命令的语法

```
$archie - option pattern
```

如果您不输入选项，而直接输入一种模式，Archie 将把模式作为文件名的全称。下面的例子是用户查找一个名为 recipe 的文件。

```
$archie recipe
```

不同的选项允许您创造更强有力的查询。选项不能合用。如果您合用选项，只有最后一个起作用。-c 选项将把模式看作不完全的模式，查找它是否在文件名中出现。下面的例子是使用者查找文件名中包含 recipe 的文件。它将匹配 recipe，cookie_recipe 和 oldrecipes。

```
$archie -c recipe
```

-s 选项执行的操作和-c 选项执行的操作属于同一类，但又有所不同。它将匹配模式的大写或小写的所有情况。下面例子中，将匹配象 recipe，Recipe 和

oldRecipes 这样的文件名。

$archie -s recipe

-r 选项允许您使用规则表达式。这意味着您能使用特殊字符来匹配多种文件名。例如，假设您想做一个精确的查询，查找的文件名中第一个字母可能大写也可能小写。则规则表达式 [Rr] ecipe 用来查找以大写或小写 r 为开头的文件名。另外，s*放在末尾时将查找以 s 结尾或不以 s 结尾的文件名：recipes 或 recipe。

$archie -r [Rr] ecipe

别的选项能控制 Archie 的输出。-m 选项加上一个数字用来限制输出的匹配项数。如果您只想看前十项，可以用-m10。-t 选项将输出的项按日期排序，开始的为最近的项。

$archie -n10 -t java
Host sun.rediris.ed
Location:/docs/faq/comp/lang
DIRECTORY drwxr-xr-x512Mar3002:18java
Host sunsite.rediris.ed
Location:/software/linux/distributions/jurix/source/networking/www
DIRECTORY drwxr-xr-x 512Mar402:52java
Location:/software/linux/networking/net-sources/www
DIRECTORY drwxr-xr-x 512Mar308:13java
Host kobra.efd.lth.se
Location:/pub/language
DIRECTORY drwxr-xr-x 512Jan2413:51java
Host sunsite.rediris.ed

Location:/software
DIRECTORY drwxr-xr-x 512Mar1317:36java

为了保存您的结果，可以用-o 选项加上一个文件名把输出改向到此文件中。下面的例子便是把 Archie 查找的结果放在名为 javares 的文件中。

```
$archie -m50 -t java -o javares
```

### 10.4.2Archie 服务器

在 Internet 上有一些 Archie 公共服务器。它们的地址通常以 archie 打头。例如，archie.sura.net，archie.internic.net 和 archie.doc.ic.ac.uk 都是流行的 Archie 服务器。Archie 服务器位于世界各地，有一些在美国。如果由于某些原因，您不能访问 Archie 客户程序，那么您只能使用 Archie 服务器。您应该在系统中安装和使用 Archie 客户程序。但是如果您使用的是没有 Archie 客户程序的系统，Archie 服务器对于执行查找也是十分有用的。

为了使用一个 Archie 服务器，您首先需要使用 telnet 注册。当提示输入注册名时，输入 archie。一旦注册成功，将显示一个 Archie 提示符：archie>。在提示符后您可以执行查找和输入参数。执行查找时使用命令 prog 加上要查找的字符串。下面的例子是查找名为 Linux 的文件。

```
archie> prog Linux
```

当您使用 Archie 客户程序时，可以使用常规表达式或模式匹配来限定查找。为了做到这点，您可以使用 set 命令和一个特定的选项来设置一个特征查找。

```
archie> set search option
```

rgex 选项表示将使用常规表达式来限定查找，sub 选项则表示将使用模式

匹配来限定查找。下面的例子中，使用者搜索包含模式“Linux”的文件名。

```
archie> set search sub
archie> prog Linux
```

下面的例子是使用常规表达式来执行查找。常规表达式 [Rr] ecipes*将搜索有如下特征的文件名：以大写或小写字母 r 开头和结尾包括或不包括一个 s，象 recipes 或 Recipe。

```
archie> set search regex
archie> prog  [Rr] ecipes*
```

另外，您可以设置控制输出的选项。例如，您可以通过设定 maxhits 来限制查询结果的输出数目。变量 sortby 是用来控制输出项按照一个特定的字段排序。可以使用 set 加上一个变量名和变量的值来对变量赋值。下面的例子中，使用者限定查询项只输出到第十项，并把输出的结果用主机名来排序。

```
archie> set maxhits 10
archie> set sortby hostname
```

一旦您完成操作，可以使用 quit 命令退出注册。

```
archie> quit
```

### 10.4.3Xarchie

Xarchie 是一个 X-Windows 程序。它让您可以使用菜单和列表框来执行 Archie 查找和显示结果。您甚至可以使用 Xarchie 为选中的文件执行一个 ftp 操作，并下载到您的系统中。Xarchie 有三个菜单，File、Settings 和 Query。菜单栏中的第四项是一个用来停止 Archie 查找的 Archie 按钮。最后一项是用来

启动窗口并显示大量 Xarchie 帮助文件的 Help 按钮。您可以按需要选择不同的按钮。Xarchie 是通过访问 Archie 服务器来执行一个查找，并显示查找的结果。选中 Settings 菜单中的 Archie Host，可以显示一列 Archie 服务器。Settings 菜单中还包含一些可以用来配置您的查找的项，可以让您决定排序的顺序和模式匹配的方法。

为了执行一个查找，在标记为 Search Term 的方框中输入查找项，接着选中 Query 菜单的 Item 项。显示的窗口中包含几个列表框，第一个框中列出找到的 Internet 站点。当您单击一个站点时，被查找的文件的路径名将被显示在第二个框中（显示的路径名可能不止一个）。第三个框中显示找到的项，可能时文件或是目录。如果是目录，双击它便可以显示其中的文件。如果您想下载一个文件，只须单击文件名并在 File 菜单中选中 Get 项。关于 Archie 项目的更多的信息显示在在列表框下面的框中。图 10-2 显示了 Xarchie 程序查找项目“java”的结果。

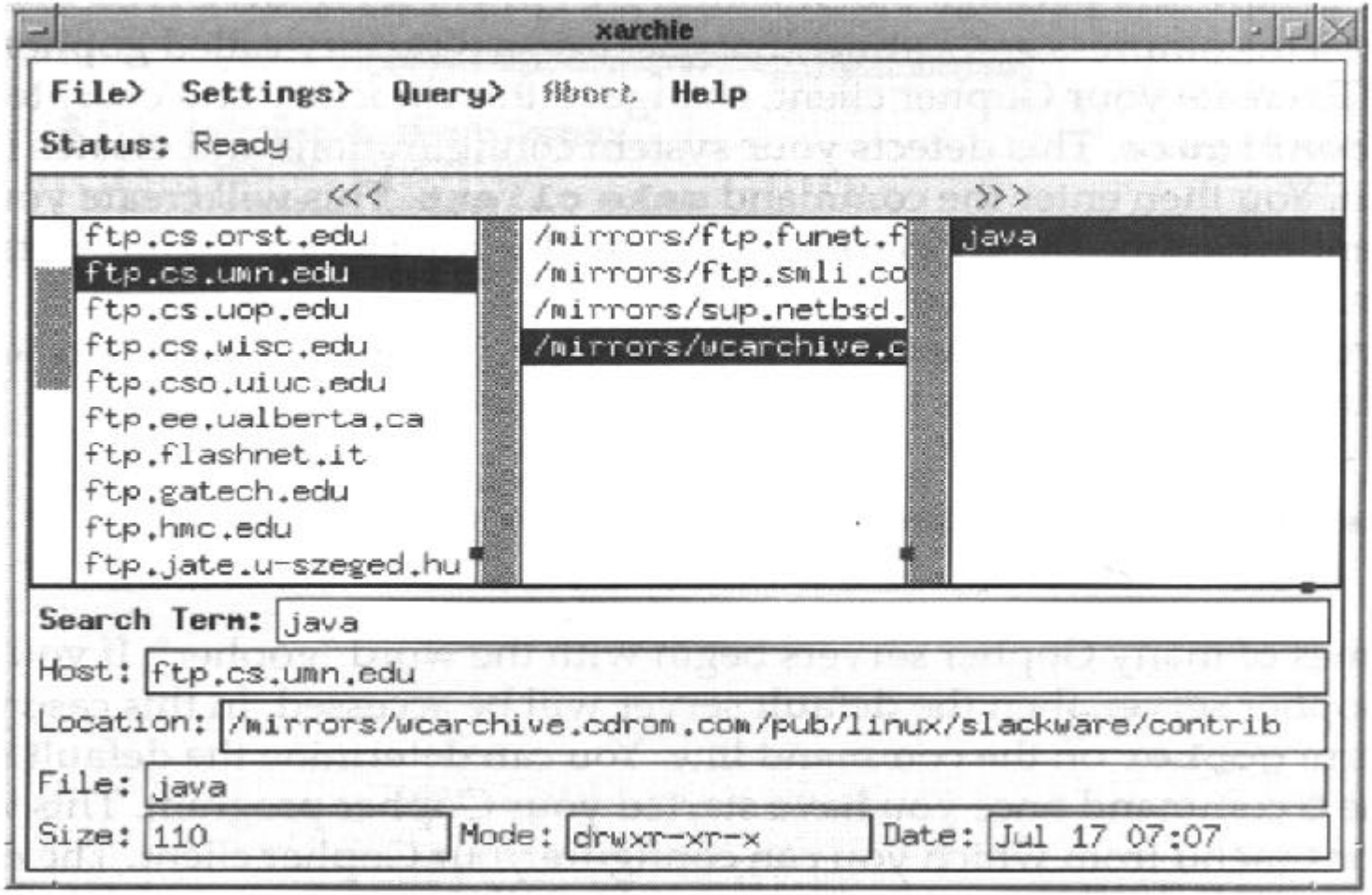


**图 10-2 显示查找结果的 Xarchie 程序**

## 10.5 一个 Internet 的用户界面：Gopher

Gopher 是关于 Internet 服务和资源的，对用户友好的，并使用菜单驱动的目录。它可以做为 Internet 服务的用户界面来操作。它综合了 telnet,ftp 和 Archie 的功能，允许您浏览和选择不同的数据库、文件和在线的信息服务。Gopher 的界面使用自顶向下的菜单系统，菜单中包括普通的和特殊的主题。您可以在菜单中随意移动以选择需要的项。

Gopher 的设计思想是把分布在网络上的信息都放在您的指尖。它起初是由 University of Minnesota 开发设计的，主要是用来连接大学内的各个系并提供全校的分布式的信息服务。每个系都有自己的 Gopher 信息服务器，任何用户都可以通过 Gopher 客户程序来访问它。这种分布模型很快就被 Internet 采用，使得其中大量的信息更容易被访问。许多大学的 Gopher 服务器都即可在校内的网络中使用又可通过 Internet 访问。

为了访问 Gopher 服务器，您需要使用 Gopher 客户程序。Gopher 客户程序提供一个菜单界面，通过它您可以发出请求并执行请求，连接到另外的服务中或传送一个文件。通常有一些 Gopher 客户程序可以使用。在您的 Caldera CD-ROM 中有一个 xgopher，它是一个基于 X-Windows 的 Gopher 客户程序，它包含一些按钮和下拉式菜单，允许您在 Gopher 菜单中移动。您可以从 ftp 站点 boombox.micro.umn.edu 上的 /pub/gopher 目录中下载 University of Minnesota 的行式 Gopher 客户程序。也可以从 Linux 的 ftp 站点或它的镜像站点上的目录 /pub/packages/info-systems/gopher/boombox/unix 中下载。现在此软件包名为 gopher2_3.tar.gz。运行它时不需要启动 X-Windows，而且它的运

行速度也越来越快。此客户程序是一个包含在 Gopher 服务器程序的软件包的一部分，在第 12 章中会详细介绍。当您解压和展开此软件包时，将会创建一个 gopher2_3 的目录。为了生成您的 Gopher 客户程序，首先切换到此目录下，执行命令 configure。它将决定您的系统配置，并专门生成一个 makefile 文件。输入命令 make client,将生成您的 Gopher 客户程序。在/doc 目录下的 INSTALL 文件中有关于安装过程的更详细的说明。

您可以在启动 Gopher 客户程序时指定要访问的 Gopher 服务器。下面的例子便是访问 Gopher 服务器 gopher.tc.umn.edu。

```
$gopher gopher.tc.umn.edu
```

许多 Gopher 服务器的名字都是以“gopher”开头的。如果您没有指定一个 Gopher 服务器，则去访问默认的服务器。这种情况下，您只须在命令行中输入 gopher。一旦启动您的 Gopher 程序，按下命令 o 就启动一个选项菜单。从此菜单您可以设置默认 Gopher 服务器，或者配置您的 Gopher 客户端。选项菜单把它的信息保存在一个名为.gopherrc 的配置文件中，此文件存放在您的 home 目录下。

您也可以用 telnet 来访问一些 Gopher 服务器,象 gopher.uicu.edu。用 telnet 访问服务器的地址，并使用地址的第一个名字注册，第一个名字通常是 gopher。象 Archie 一样，再次强烈推荐您使用 Linux 系统中的 Gopher 客户程序。

### 10.5.1Gopher 菜单

Gopher 菜单中包括一列菜单项，其中有代表文件、菜单、数据库和 telnet 连接的项。每个菜单项的类型由放置在项末尾的限定词说明。表示文件的项是

以一个圆点限定符来结尾的。表示菜单的项的结尾是一个斜杠。数据库项的结尾是一个符号<?>。限定符<CSO>是用来指定一个查找用户地址和信息的 CSO 名字服务器。限定符<TEL>指定一个 telnet 连接。限定符<Picture>，<Movie>和<)分别说明图象，视频和声音文件。图 10-3 显示一个 Gopher 菜单。

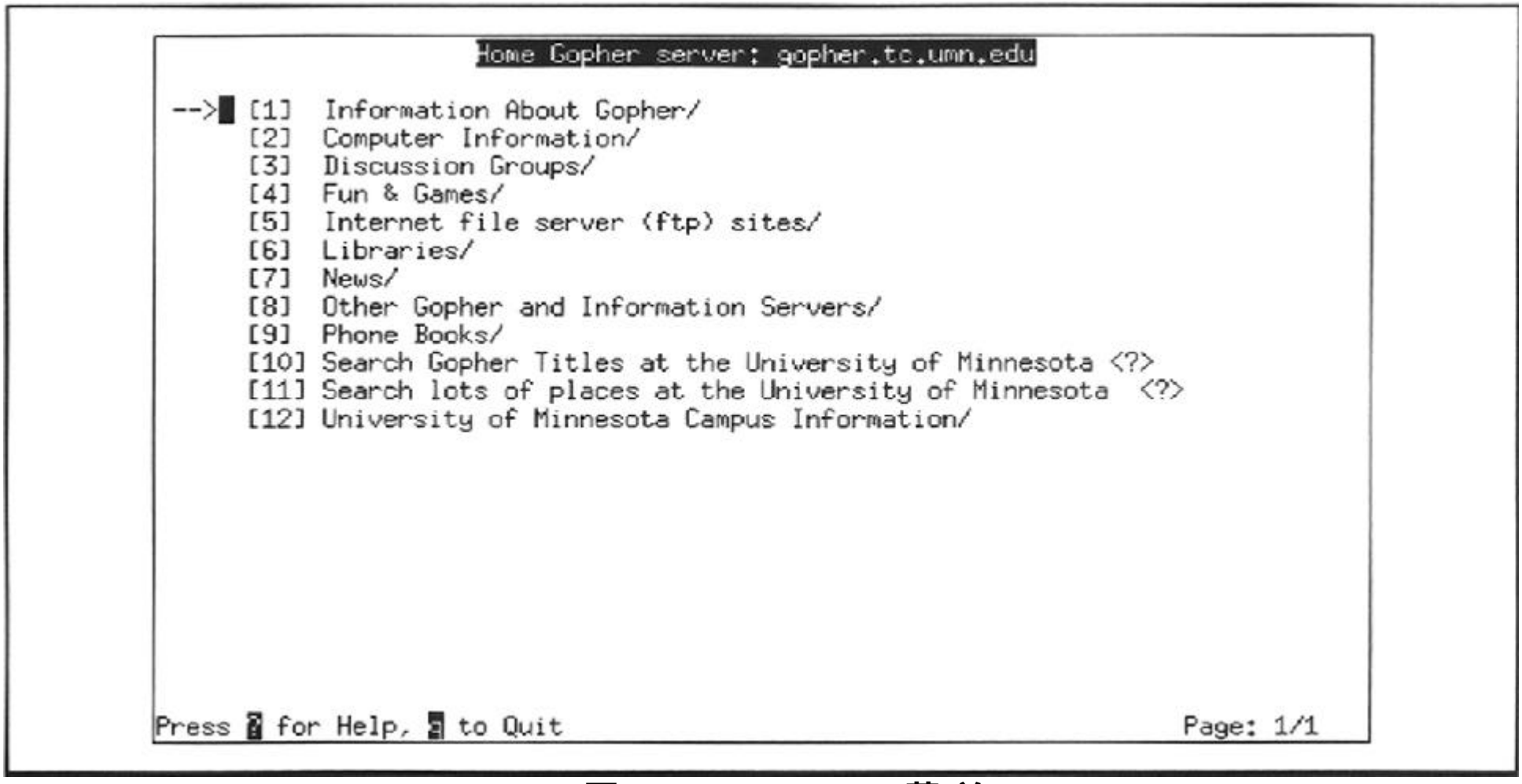


图 10-3Gopher 菜单

刚启动 Gopher 菜单时，在第一个菜单项前显示一个箭头。您可以用此箭头来选择需要的项。您也可以用键盘上的箭头键把屏幕上的箭头从一个菜单移到另一个。用 UP ARROW 可以移到上一项，用 DOWN ARROW 移到下一项。下面的例子显示 Gopher 的第一层菜单。前三项都有一个/限定符，标明它们能引出其他的菜单。第一项用一个圆点结束，表明它是一个文本文件。注意箭头处在第一个菜单项前。

一旦您把箭头移到需要的项，按 ENTER 可以选中和显示该项。另外，您还可以输入菜单项的序号后按 ENTER。您可以在菜单中连续移动到需要的项。还可以用 u 键退回到前一个菜单项。任何时候，按 q 即可退出 Gopher。图 10-4 列出 Gopher 命令。

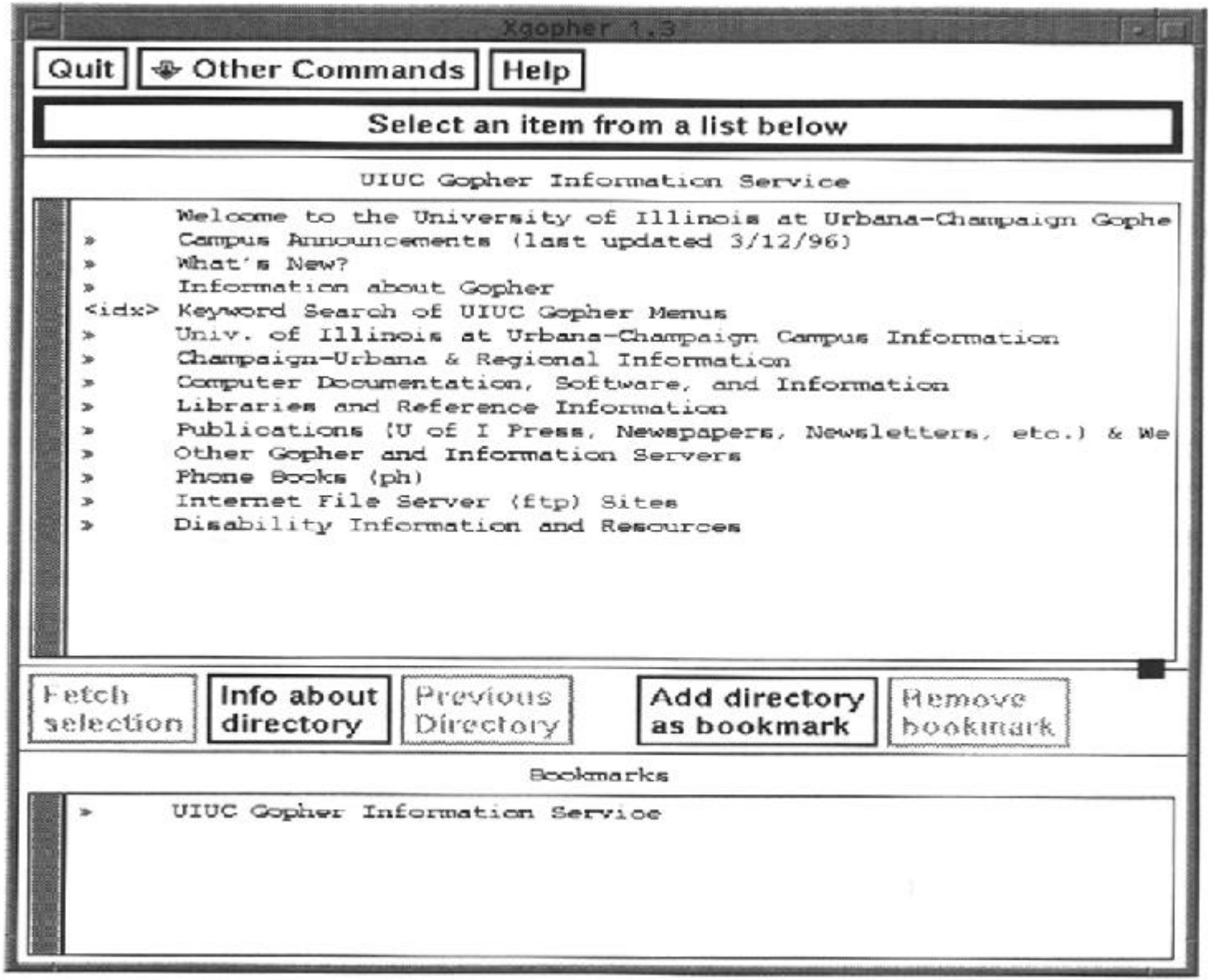


**图 10-4 显示 xgopher 客户程序**

一个 Gopher 菜单可能非常长，经常包括好几页。可以按 SPACEBAR 移到下一屏。命令 b 可以退回到菜单的前一屏。当前所在的屏数和菜单的总屏数将显示在屏幕的右下角。

为了定位一个菜单项，您可以在屏幕中一个一个地查找，或是可以用 Gopher 菜单项的查找功能。Gopher 能在每个菜单项中执行一个模式查找。为执行模式查找，首先按下斜杠键/，在屏幕中将打开一个方框，提示您输入一个查找模式。输入模式后，按 ENTER 键开始执行查找。系统将在每个菜单项的正文中查找该模式。在任何时候，按 CTRL-g 即可取消查找。查找的结果将显示在屏幕上。还可以继续执行查找直到寻找到您所需的所有项目。在查找结果中选中您想显示的项并按 ENTER 即可显示它。显示一项后，Gopher 将提示您是否想保存、邮寄或打印信息，或是退回到前一个 Gopher 菜单。

Press <RETURN> to continue, <m> to mail, <s> to save, or <p> to print:

保存选项所做的工作和 ftp 是一致的。使用此选项则把显示出来的包含信息的文件传输到您的系统。另外，还可以把此信息邮寄给您自己或寄给那些能接受邮件信息的用户。您还可以只打印此信息而不保存。

xgopher 客户程序的操作略有不同。xgopher 菜单中不是使用箭头来定位，您可以用鼠标单击菜单项。双击某项即选中它。用按钮和下拉式菜单在 Gopher 菜单中向前或向后移动。

### 10.5.2 用 Gopher 访问服务信息

用 Gopher 可以很容易地访问服务的信息，象查找在线的图书目录。这样的

服务信息在菜单项中以<TEL>限定符结尾。当您选择这样的项，Gopher将为您telnet到提供服务的服务器上。例如，当您使用Gopher注册到国会图书馆的站点。首先从Gopher菜单中选择图书馆菜单。将进入另一层Gopher菜单，其中包括您可以访问的在线的图书馆。选择国会图书馆的菜单项，显示国会图书馆的Gopher菜单。从此菜单中，您能选择菜单项以列出更进一层的Gopher菜单。注意此菜单中有一些项是文件，还有一些是Gopher菜单，其中还有两个项是在线信息服务，它们带有<TEL>限定符。您选择它们以连接到国会图书馆的在线目录。Gopher客户程序就建立一个telnet连接，并警告您将离开Internet和Gopher而进入一个在线服务。

一旦注册入一个在线信息服务，您将用服务提供的界面和命令进行操作。在访问国会图书馆时，系统给您一个菜单并限定您的查找。可以使用关键字执行查找并显示结果。结束telnet时，只须简单的退出注册。

### 10.5.3Gopher书签菜单

当通过Gopher查找项目时，您可能需要保存一些菜单或菜单项以供以后访问。例如，在今后的几天内您需要频繁地访问NASA。通常，每次您都需要访问Gopher菜单的关于NASA的项，首先从第一级菜单开始，一步步的执行下去。您可以用Gopher的书签功能来替代每次执行的操作，把NASA项放在一个特殊的书签菜单中，这样您就可以直接访问NASA项。从某种意义上说，您创建了自己的Gopher菜单。

用命令A和a可以把一个菜单或菜单项加入书签列表中。命令A把当前显示的整个菜单加入您的书签列表中。小写的命令a则是把当前选中的菜单项加

入书签列表中。把菜单或菜单项加入书签列表中后，您可以从书签菜单中直接访问它们。书签列表用来产生书签菜单，并把其内容加入书签菜单中。用命令 v 可以访问书签菜单。

在如图 10-5 所示的 NASA 的例子中，使用者只须用 Gopher 菜单定位它一次，并用命令 A 或 a 把它加到书签列表中。无论何时使用者若想再次访问 NASA 项，他只须显示书签菜单并从中选择 NASA 菜单项即可。

您可以按需要把任何菜单项加入书签列表中。在图 10-6 中，使用者把 San Francisco 的天气预报加入书签菜单。使用者首先要找到 San Francisco 的天气预报。从 News 菜单开始，使用者移到 National Weather Forecast 菜单，其中列出各个州名。从州名列表中选中 California 并移到 California 菜单，其中列出 California 的各个城市。最后，使用者把箭头移到 San Francisco。接着按 a 键把 San Francisco 菜单项加入书签菜单中。在屏幕的中间将打开一个对话窗口，提示此菜单项的名字将加入到书签菜单中。在此显示的名字是它缺省的名字，即 San Francisco。使用者在名字后加一个单词“Weather”，使得书签菜单中新的菜单项名为“San Francisco Weather”。现在，要查找 San Francisco 的天气，使用者只须按 v 键调出书签菜单并从中选择菜单项 San Francisco Weather 即可。不需要再查找 News，Forecasts，State 和 City 等一系列菜单。

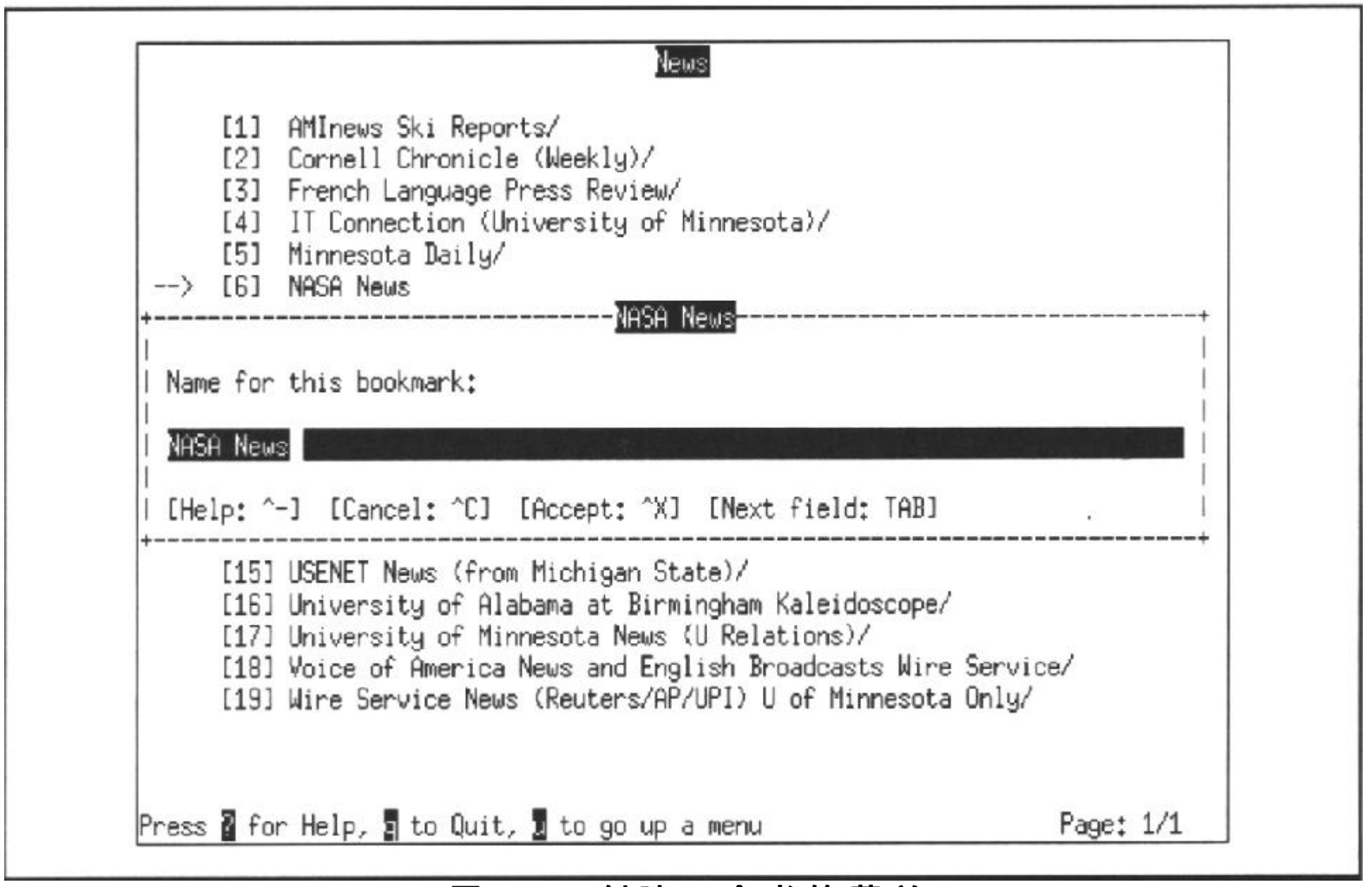


**图 10-5 创建一个书签菜单**

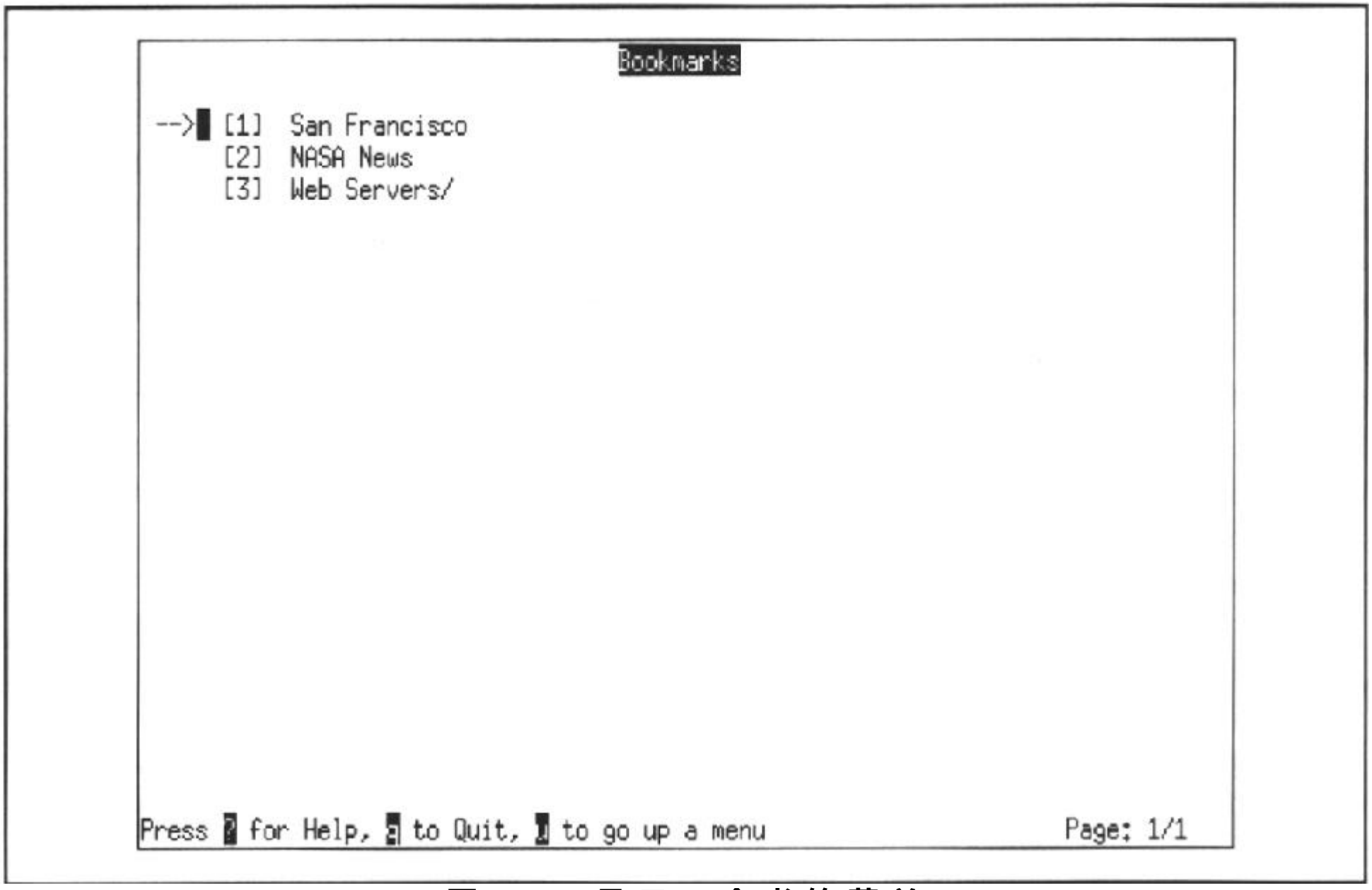


**图 10-6 显示一个书签菜单**

当您不需要书签列表中的一项时，用命令 d 可以删除它。例如，一旦使用者不再需要 NASA 菜单项，他可以在书签菜单中选中此项并按 d 键即把它删除。

正如您保存菜单项一样，也可以在书签列表中保存一个完整的菜单。用 A 命令可以把一个菜单加入您的书签列表，它将把当前显示的菜单加入书签菜单中。例如，如果您在下几个星期中要经常参考 California 的各个城市的天气预报。可以使用 Gopher 的书签功能把 California 的天气预报菜单作为一项加入您的书签菜单中。无论何时您需要访问 California 的天气预报菜单，只须显示您的书签菜单并从中选择 California 的天气预报菜单即可直接访问。

此功能对保存使用 Gopher 进行 Archie 查找所得结果十分有用。在 Gopher 得主菜单中，ftp 查找项允许您使用 Archie 查找来搜索一个文件并把它下载到您的系统中。查找的结果以 Gopher 菜单的形式列出。每个菜单项都是存放在一个匿名 ftp 服务器上的文件。选择一个菜单项将使 Gopher 去访问 ftp 服务器并把文件传输到您的系统中。

从 Archie 查找得到的 Gopher 菜单只是临时的菜单。如果想保存它留做以后使用，您需要把它放在书签菜单中。当显示此菜单时，按命令 A 将把它加入您的书签菜单。以后就可以从书签菜单中访问您以前的 Archie 查找结果。

### 10.5.4Veronica

关于一个主题的信息资源是广泛地分布在 Internet 上。每次您要查找一个主题的资源时，都要通过不同的 Gopher 菜单一个一个的查找。Veronica 的功能正是用来把您的上述工作流程化。Veronica 将用使用者提供的关键字搜索所有的 Gopher 菜单，然后产生一个包含所有查找结果的菜单。所以使用 Veronica

将有效地产生一个关于特定主题的临时的 Gopher 菜单。您可以象使用常规的 Gopher 菜单一样，在此临时菜单中选择所需要的项目。

为了使用 Veronica，首先您需要访问 Gopher 中的 Veronica 菜单。Veronica 菜单中列出不同的 Veronica 服务器，您可以用它们来查找 Gopher 菜单。第二和第三个菜单项提供关于 Veronica 的更多的信息。第一个菜单总是空的。每个服务器菜单项的结尾带有限定符<?>，它表明一个可以用关键字来查询的数据库。选择一个服务器并按“ENTER“。在屏幕中间将打开一个对话方块，提示您输入查找的关键字。输入后，再按“ENTER”， Veronica 将执行查找。所有 Veronica 能查到的结果被定制到一个 Gopher 菜单中。您可以在此菜单中选择需要的项。如果您想知道某个菜单项的出处，可以移到此菜单项并按=以显示它的信息。

通过 Veronica 产生的菜单也是临时的，您可以把它加入到书签列表中。当显示您的 Veronica 菜单时，按命令 A 就可把它加到您的书签列表中。以后通过按命令 v 显示书签菜单，并从中选择 Veronica 菜单便可以直接访问它。您可以在书签菜单中保存您所需要的任何 Veronica 查找结果。还可以在 Veronica 产生的菜单中选定一项后，按命令 a 把此项保存。

## 10.6WAIS

WAIS（广泛的信息服务）是用来查找 Internet 上可用数据库的信息服务。在 Internet 上有许多 WAIS 数据库，其中包含关于不同主题的文章，象关于电

影或编程等方面的文章。一个 WAIS 数据库中收集大量的文档,这些文档用 WAIS 索引软件做了索引。这些索引可以用来查找文件。

为了搜索 WAIS 数据库,您可以使用 WAIS 客户程序 waisq,swais 或 xwais,或是 Web 浏览器象 Netscape。waisq 使用命令行的用户界面。swais 提供全屏用户界面。xwais 使用 X-Windows 界面。swais 的命令和选项列在表 10-5 中。一个 WAIS 客户程序允许您选择一个 WAIS 数据库并使用复杂的逻辑查询来执行查找。然后列出结果并对结果编排序号。您可以选择列表中需要的文章的序号,系统将显示文章。也可以保存或打印它。您还可以使用 Web 浏览器来访问 WAIS 站点,象 www.wais.com。在此站点您可以执行查找或移到别的站点。

不像查找别的信息服务,WAIS 可以执行全文查找并对结果进行排列。WAIS 将查找每篇文章的全文,而不仅仅只查找文章的标题或索引字的预定列。在这种情况下,每篇文章的内容都被检测,提供了更全面的查找。WAIS 查找的结果将被从 0 到 1000 进行排序,开始的文章是有最佳匹配查找项的文章。您可以在结果中进一步查找。

如果想使用 WAIS 客户程序,首先要下载它并安装在系统中。有一个免费版本的 WAIS 叫 freeWAIS,是由 Clearinghouse for Network Information Discovery and Retrieval(CNIDR)开发的。它是一个完整的软件包,包括 WAIS 服务器、一个索引软件,swais 和 xwais。在它的 Web 站点 ftp.cnidr.org 上有一个 Linux 的版本,它已经编译过,可以直接运行。您也可以下载源代码,自己编译。在站点 sunsite.unc.edu 和它的镜像站点的 /pub/packages/info-system/wais 目录中也有 freeWAIS。

为了直接访问 WAIS 数据库,您的系统中需要装有 WAIS 源文件。一个 WAIS

源文件中包含一个 WAIS 数据库的 Internet 地址和数据库中的信息。源文件的扩展名为.src，应该放在您的/usr/lib/wais-sources 目录中。您可以自己创建此目录，并从 sunsite.unc.edu 中取得 WAIS 源文件以供通常访问 WAIS 数据库。

在目录/pub/packages/info-systems/wais 中，首先选中并下载文件 wais-servers.tar.Z，接着把此文件放到目录/usr/lib/wais-sources 中。用 gunzip 和 tar 解压和扩展文件。将会有大量的 WAIS 源文件被展开。其中有一个特殊文件名为 directory-of-servers.src，它将访问在 quake.think.com 上的 WAIS 服务器，此服务器中含有 WAIS 服务器的主目录。下一次启动 swais 时，将显示一个长的数据库列表以供您从中选择。每个数据库都有一个自己的 WAIS 源文件，存放在/usr/lib/wais-sources 目录中。按/来查找数据库名或用箭头键在列表中上下移动，当找到您需要的项，按 ENTER 选中它。您能从中找到任何需要的东西，艺术品乃至菜谱。

用 freeWAIS 软件，可以创建自己的 WASI 数据库。您可以收集一些文档，对它们进行排序，使它们能被 Internet 上其他用户查找访问。在第十二中将详细描述 freeWAIS 服务器软件。

## 10.7 总结：访问 Internet

Linux 有一组 Internet 工具，您可以使用它们来查找和访问 Internet 上的信息站点。用 telnet 可以远程注册到连接在 Internet 上的其它计算机。例如，您可以直接连接到国会图书馆的在线目录并在上面执行查询。用 ftp 您可以从 ftp

站点上下载文件到您的计算机中。用 Archie 您可以找到某一个文件的所在位置。Gopher 混合了上述三种工具的功能，是一个功能强大、易于使用的用于浏览 Internet 的工具。它把信息组织到一系列菜单中，您可以在菜单中移动并找到您需要的项。最后，您能用 WAIS 在多种数据库中执行全面的查找。

表 10-1telnet 选项和命令

| telnet 命令模式 | 作用 |
| --- | --- |
| -d | 把调试触发器设为 TRUE |
| -a | 用 USER 变量自动注册，USER 变量由 ENVIRON 选项设定 |
| -n tracefile | 为记录轨迹信息打开 tracefile |
| -l user | 用 user 作为注册名连接到一个远程系统 |
| -e escape char | 设置 telnet 的换码符 |
| close | 关闭一个 telnet 并退回到命令模式 |
| open host [ [-l] user] [-port] | 打开到一个特定主机的连接 |
| quit | 关闭所有的 telnet 连接并退出 telnet |
| !command | 执行一个 Linux 命令 |
| Status | 显示当前 telnet 的状态 |
| ?command | 显示一个不带参数的帮助总结；如果指定一个命令，显示此命令的信息 |
| set variable valueunset variable | 设定不同的 telnet 变量；参见 man 页 |

| | |
|---|---|
| value | 中更详细的列表 |
| toggle arguments | 触发器的不同标记，为 true 或 false；参见 man 页中更详细的列表 |

表 10-2ftp 命令

| 命令 | 作用 |
|---|---|
| ftp | 调用 ftp 命令 |
| Open | 打开到另一个系统的连接 |
| Close | 关闭一个连接 |
| quit 或 bye | 退出 ftp |
| get filename | 从远程系统中发送文件到本地 |
| put filename | 从本地发送文件到远程系统 |
| mget regular-expression | 允许您从远程系统中一次下载多个文件；您可以使用特定的字符来指明文件；在文件的传输过程中，将逐个提示您 |
| mput regular-expression | 允许您从一次上传多个文件到远程系统中；您可以使用特定的字符来指明文件；在文件的传输过程中，将逐个提示您 |
| Binary | 以二进制模式传输文件 |
| ascii 以 ascii | 模式传输文件 |

| | |
|---|---|
| cd directory | 改变远程系统中的目录 |
| lcd directory | 改变本地系统中的目录 |
| help 或？ | 列出 ftp 命令 |

表 10-3Archie 命令

| Archie 客户程序选项 | 描述/作用 |
|---|---|
| archie options search-string | |
| -e | 精确的模式匹配（缺省） |
| -c | 寻找包含模式的文件名 |
| -s | 查找忽略某种情况的文件名 |
| -r | 模式是一个常规表达式 |
| -t | 按照日期对结果排序 |
| -l | 把输出的结果作为一个记录，能用程序对其进行语法分析 |
| -o filename | 把查找结果放在 filename 中 |
| -h hostname | 查询主机名为 hostname 的 Archie 服务器 |
| -m num | 限制结果的最大数（匹配后） |
| -N num | 估计查询结果的数目 |
| -L | 列出已知的 Archie 服务器 |
| -V | 详细选项；Archie 将输出一个长列表 |

| 公共服务器命令 | |
|---|---|
| Prog | 查找出现模式的文件名 |
| List | 列出已知的 Archie 服务器 |
| Site | 列出在一个特定主机上的文件 |
| Mail | 邮寄查找的结果 |
| Quit | 退出在 Archie 服务器上的注册 |
| Help | 显示帮助 |
| set variable value | 设置 Archie 变量 |
| Show | 显示 Archie 变量的当前值 |
| Unset | 删除变量 |
| **服务器变量** | |
| autologout num | 在自动退出对 Archie 服务器的注册前，Archie 的空闲等待的分钟数，缺省的为 15 分钟 set autologout 10 |
| mailto address | 结果发送的地址 set mailto richpete@garnet.ber keley.edu |
| maxhits num | 限制结果的最大数目（匹配） set maxhits 10 |
| Pager | 用缺省的分页功能，象 More，来显示您的结果 set pagerun set pager |
| serch option | 查找类型 set search subcase search options |

| | |
|---|---|
| sortby option | sub 包含文件名的查找模式<br>subcase 区分大小写的查找模式<br>exact 精确的查找模式<br>regex 使用常规表达式<br>输出的排序类型<br>set sortby filename sortby optins<br>none 不排序<br>filename 用文件名的字母顺序对输出进行排序<br>hostname 用主机名对输出进行排序<br>time 按最近日期对输出进行排序<br>size 按最大尺寸对输出进行排序<br>rfilename 反转用文件名的字母顺序的排序<br>rhostname 反转用主机名的字母顺序的排序<br>rtime 按最远日期对输出进行排序<br>rsize 按最小尺寸 |
| 对输出进行排序<br>Status | 当执行查找时，发表一个状态报告 set statusunset s tatus |

| | |
|---|---|
| term terminal-id | 您使用的终端类型 set term vt100 |

表 10-4Gopher 命令

| 选项 | 作用 |
|---|---|
| gopher [-sb] [-t title] [-p path] [hostname port] | |
| -p string | 启动时把此字符串发送给根服务器 |
| -t string | 为 Gopher 客户程序设置初始屏幕的标题菜单项限制符.文件 |
| / | 菜单 |
| <CSO> | CSO 名字服务器 |
| <TEL> | telnet 连接 |
| <Picture> | 图象，象 gif 或 jpeg |
| <) | 声音文件 |
| <Move> | 视频，象 mov 或 avi |
| <HTML> | 超文本文件 |
| <Bin> | 二进制文件 |
| <HCX> | Machintosh BinHexed 文件 |
| <PC Bin> | DOS 二进制文件 |
| <MIME> | 多用途的 Internet Mail 扩展文件 |

| | |
|---|---|
| <?> | 带关键字查找的数据库 |
| **移动和选择菜单项** | |
| k 和 UP ARROW | 移到上一个菜单项 |
| j 和 DOWN ARROW | 移到下一个菜单项 |
| num | 移到菜单中第 num 个菜单项 |
| l，RIGHT ARROW，ENTER | 按任意一个选择当前的项 |
| **在菜单中查找一项** | |
| /pattern | 在菜单中查找某模式并移到第一个有此模式的项 |
| n | 重复前一个查找 |
| **对菜单项执行的操作** | |
| = | 显示关于菜单项的信息 |
| s | 保存当前的项到一个文件中 |
| m | 把当前项寄给一个用户 |
| p | 打印当前项 |
| **在菜单屏幕中移动** | |
| >，+，SPACEBAR | 移到下一个菜单屏 |
| <，-，b | 移到上一个菜单屏 |
| **返回到前一个菜单** | |
| u， | .移到上一个菜单 |
| m | 返回到主菜单 |
| **书签命令** | |

| | |
|---|---|
| a | 加一个选中的项到书签列表中 |
| A | 加当前的菜单到书签列表中 |
| D | 从书签列表中除去一项 |
| V | 显示书签列表中选项，退出和帮助命令 |
| Q | 退出 Gopher |
| Q | 不提示直接退出 Gopher |
| ? | 帮助 |
| o | 显示和改变 Gopher 的选项环境变量 |
| PAGER | 客户端用色将文件显示给用户 |
| GOPHER_MAIL | 用来显示文件的客户程序 |
| GOPHER_PLAY | 发送邮件的程序 |
| GOPHER_TELNET | 播放声音的程序 |
| GOPHER_HTML | 连接 telnet 服务的程序 |
| GOPHER_PRINTER | 从一个管道中打印的程序 |

表 10-5WAIS 客户程序命令

| | |
|---|---|
| -s sourcename | 选择 sourcename 来搜索；第 12 章中将讨论资源 |
| -S sourcedir | 指定一个源目录；缺省的是:~/wais-sources |
| -C sourcedir | 指定一个公用的源目录；缺省的是 |

| | |
|---|---|
| | /usr/lib/wais-sources |
| -h | 帮助信息 |
| **命令** | |
| j，DOWN ARROW，^N | 向下移动一个 |
| k，UP，ARROW，P | 向上移动一个 |
| J，V，D | 移到下一屏 |
| K，ESC v，^U | 移到上一屏 |
| ### | 资源号为###的位置 |
| /sss | 查找资源 sss |
| SPACEBAR，.(period) | 选择当前资源 |
| = | 撤消对所有资源的选择 |
| v，,(comma) | 查看当前的资源信息 |
| <ret> | 执行查找 |
| s | 选择新的资源（刷新资源列表） |
| w | 选择一个新的关键词 |
| X， | -永久地移走当前资源 |
| o | 设置和显示 swais 选项 |
| h，？ | 显示帮助 |
| H | 显示历史程序 |
| q | 离开程序 |

# 第 11 章 World Wide Web

World Wide Web 是一种包含不同类型信息的超文本的数据库，其中的信息分布在 Internet 上的许多不同的站点。数据库中包含许多项，相关项之间互相链接。所以在寻找一项时，您能从此项寻找到它相关的项。例如，您能够找到一篇关于 Amazon 雨林的文章，接着从此文章您能找到雨林的地图或是一个图片。从这方面讲，一个超文本数据库更像一个互相连接的数据的环球网，您能从网中的一个数据追踪到其他的数据。信息被显示在众所周知的 Web 页中。在 Web 页中，有些关键字是高亮度的，它们链接到另一个 Web 页或是别的诸如图片、文章或文件的项。

World Wide Web 连接遍布全球的 Internet 上不同站点上的数据。它常被称为 WWW 或简称为 Web。World Wide Web 起源于欧洲的 CERN 研究实验室。CERN 中仍保留最原始的 WWW 服务器。一个操作起来象 Web 服务器的 Internet 站点被称为 Web 站点。这些 Web 站点经常是关于某一特定主题或机构，例如，Smithsonian Web 站点或 NASA Web 站点。这些 Web 站点的 Internet 地址通常以 www 开头，像 www.caldera.com 便是 Caldera 的 Web 站点的地址。一旦您连接到一个 Web 站点，您能使用超文本连接从一个 Web 页移到另一页。

您可以使用被称为浏览器的客户程序来访问 Web。现在有许多不同的浏览器可以使用。这些浏览器分别可以用在 Unix，Windows，Mac 和 Linux 等系统

上。某些浏览器，象 Netscape 和 Mosaic 有运行在所有系统上的版本。在您的 Linux 系统上，可以从几个浏览器中选择一个来使用。您的 OpenLinux 系统中带有 Arena 浏览器。除了 Arena 之外，还有一个 Lynx 浏览器。这是一个命令行模式的浏览器，它只能显示文本。您可以免费下载别的浏览器到您的系统中。例如 Mosaic 和 Netscape 是基于 X-Windows 的浏览器，它们有显示图片、播放声音和视频的功能。Mosaic 可以使用公共的序列号。从 Netscape 站点上您可以下载 Netscape 的一个试用版。

## 11.1URL 地址

使用如图 11-1 所示的通用资源定位器（URL）来访问 Internet 资源。一个 URL 由三部分组成：传输协议，主机名和路径名。传输协议和主机名用冒号和两个斜杠“ :// ”分开。路径名总是以一个斜杠开始。

传输协议通常是 http（超文本传输协议），表明是一个 Web 页。别的传输协议是 gopher，ftp 和 file。正如它们名字所示，gopher 和 ftp 分别表示 Gopher 和 ftp 连接。file 显示一个文本或目录文件，也可以是一个 HTML 文件。表 11-1 列出大量的传输协议。

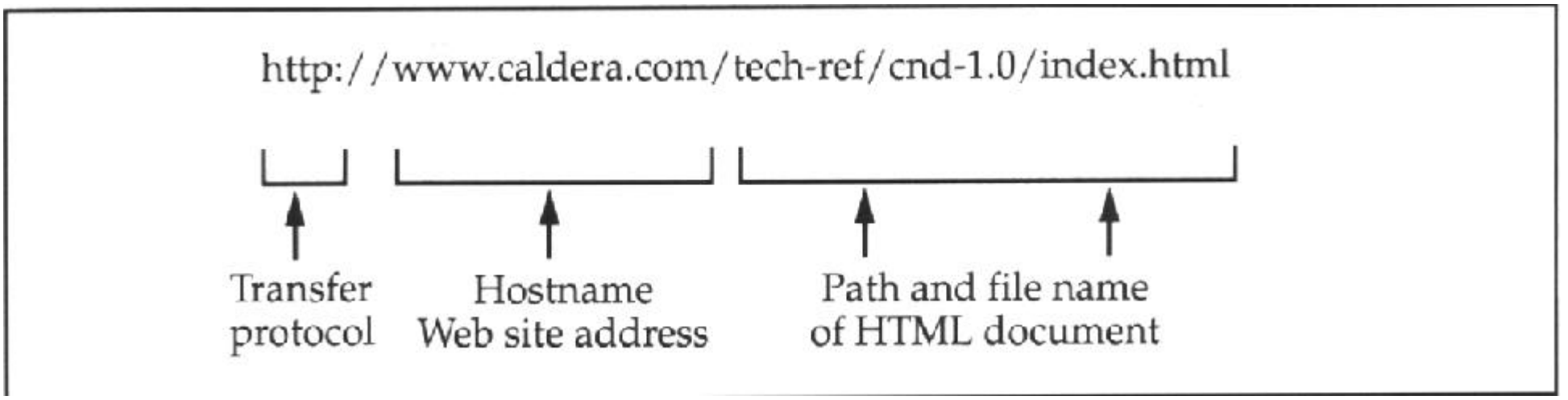


图 11-1 通用资源定位器（URL）

主机名是位于特定 Web 站点的计算机。您可以认为是 Web 站点的地址。通常大多数主机名是以 www 开头。下面的例子中，URL 定位在站点 home.netscape.com 上的 Web 页 toc.html。

http://home.netscape.com/toc.html

如果您不想访问一个 Web 页，可以退出它。当您访问一个 Web 站点时，将自动的访问它的主页。为了能直接访问一个 Web 站点，您能使用它的主机名来访问。一个 Web 站点的缺省的主页名为 index.html，位于站点的顶级目录中。一个 Web 站点能重新指定一个特殊的文件作为缺省主页。然而，如果没有指定特定的文件，文件 index.html 将被作为主页。下面的例子中，用户将调用 Caldera 主页。

http://www.caldera.com

路径名指定在主机系统中源文件存放的目录和文件名。例如，

/pub/Linux/newdat.html 便指明一个名为 newdat 的 HTML 文件，存放在 /pub/Linux 目录中。当您移到站点上另外的 Web 页中，您也移动到更深层的目录树中。下面的例子中，用户访问目录/tech-ref/cnd-1.0/faq 中的 faq.html文件。

http://www.caldera.com/ tech-ref/cnd-1.0/faq/faq.html

如同上面所讲的，您可以只指定目录名，而不指定一个特定的 Web 页文件，您的 Web 浏览器将寻找此目录下一个叫 index.html的文件，并把此文件作为目录的缺省 Web 页。下面的例子将显示目录/tech-ref/cnd-1.0 中的 index.html页。

http://www.caldera.com/tech-ref/cnd-1.0/index.html

您可以使用此技术来访问本地系统中的 Web 页。例如，一旦安装后，一个 Java 的演示 Web 页将存放在/usr/local/java/中。因为是在您的本地系统，所以不需要写主机名。当您指定路径名时，将自动显示 index.html 页。同样还可以显示您的系统文件，它存放在/usr/doc/HTML/ldp 中，并且是 Web-page 格式。

file:/usr/local/java

file:/usr/cod/HTML/ldp

如果您引用一个没有 index.html 的目录，Web 服务器将为您创建一个，并在您的浏览器中显示。此索引文件只简单列出该目录中的文件和目录。您可以单击一项来显示文件或进入另一个目录。第一项指的是父目录。从源文件的扩展名中可以得知要对它所进行的操作。一个图片将有扩展名.gif 或.jpeg，并将被转换以显示。声音文件有扩展名.au 或.wav 将被播放。下面的 URL 地址便是引用一个 gif 文件。您的浏览器将调用一个图形阅读器来显示它。表 11-2 中提供一列不同的文件扩展名。

http://www.train.com/engine/engine1.gif

# 11.2Web 页

一个 Web 页是一种特定格式的文件，它能被 Web 浏览器显示。您可以认为 Web 页是一个包含文本和图片的字处理文件。在 Web 页中，能够嵌入调用其他 Internet 资源的链接。一个 Internet 资源可以是一个图片、文件、telnet 连接或是一个 Web 页。Web 页能作为访问不同 Internet 工具的一个界面，可以访问的工具象用 ftp 下载文件，用 telnet 连接到远程站点的在线目录中。

当您第一次启动桌面时，将看到如下所示的一个 Caldera Info 图标。此图标是您的桌面用来显示 HTML 页。

Web 页中包含文本和图形。文本被分成段落并被组织成不同的主题。各种大小的图片可以放在页中的任何位置。页中通常包括一些能调用另外的 Internet 资源的定位点。每个定位点指定一个特定的 Internet 资源。一个定位点可以引用一个图片、另一个文件；有些定位点可以引出一个 Web 页或 Web 站点。这些定位点通常是增亮的文本或图形，且和页中的其他文本有不同的颜色。通常的文本是黑色的，文本中的定位点可以是绿色、兰色或红色。移动鼠标指针到定位点上并单击它，便选中一个定位点。同时也调出此定位点指向的 Internet 资源。如果指向的是一个图片，则显示此图片，若是一个 Web 页，显示此页。若指向的是另一个 Web 站点，则访问此站点。从颜色中您能了解到定位点的状态和使用的浏览器。Mosaic 和 Netscape 都使用兰色表示您未访问的定位点。对于访问过的定位点，Netscape 使用紫色表示，而 Mosaic 使用红色。一个 Web 页可以指定自己特定的颜色。

您的 Web 浏览器将保留一列您访问过的 Web 页。您可以在列表中很容易

的向前或向后移动。调用一个 Web 页后，您可以使用浏览器移回您以前曾调用的 Web 页。Web 浏览器按您的调用顺序组织 Web 页。它将跟踪您访问过的 Web 页的踪迹。然而，在一些站点中，Web 页被按一个特定顺序链接起来，就像书中的章节。这些页中，通常在页底部有一些引用下一页或前一页的按钮。按 Next 钮将显示站点中的下一页。按 Home 钮将会到第一页。

## 11.3Web 浏览器

大多数的浏览器是设计用来访问不同的信息。它能访问远程系统中的一个 Web 页或是本地系统中的一个文件。一些浏览器还可以访问一个远程的新闻服务器或一个 ftp 站点。站点的信息类型通常由关键字指定，http 表明一个 Web 站点，nntp 表明新闻服务器，ftp 表明是 ftp 站点，file 表明您系统中的文件。

为了访问一个 Web 站点，您可以输入 http://加上该站点的 Internet 地址。如果您想访问站点上的一个特定的 Web 页，在 Internet 地址后加上此页的路径名，然后按 ENTER，浏览器将为您连接此 Web 站点并显示它的主页或是您指定的 Web 页。

您可以用 Web 浏览器访问您系统中的 Web 页，只须输入 file 和一个冒号，file:，再加上您要显示的 Web 页的路径名，并不需要指定 Internet 站点。记住，每个 Web 页都有扩展名.html。在您的系统中的 Web 页能连接到您的系统中的其他 Web 页或是远程系统中的 Web 页。当第一次启动您的 Caldera Network Desktop 时，您的浏览器将显示一个本地的 Web 页，通过此页可以连接到

Caldera 的 Web 站点，从此站点您能获得在线的支持。您可以创建自己的页，并可以使其中的一个做为您的缺省主页。

在 Web 站点上的 Web 页中经常包含到另外的 Web 页的连接，被连接的 Web 页可能在同一站点上，也可能在不同的站点。通过这些连接，您可以从一页移到另一页。当您使用定位点或按钮从一个 Web 页移到另一个 Web 页时，您的浏览器中将显示当前页的 URL。您的浏览器中将保存有在此浏览器中访问过的一列 Web 页。大多数的浏览器都有一些按钮，它们可以使您在访问过的按钮页列表中向前或向后移动。您能一个一个的退回到您前一个访问的 Web 页，同样您也可以移到下一个访问过的 Web 页。

为了得到某一特定的页，您可能要在一系列互相连接的页中移动，直至找到您需要的页。访问任何 Web 页都需要它的 URL 地址。您可以输入它的 URL 地址，这样就可以直接访问此页，而不用向上面所述的那样一个一个的查找。为了省去您记录和输入 URL 地址，大多数的 Web 浏览器都有一个活动表，在表中记录您想直接访问的 Web 页。当您访问一个以后还想继续访问的 Web 页时，把它加到您的浏览器的活动表中。在活动表中列出的是 Web 页的标题，而不是它的 URL 地址。直接选中活动表中的项，即可访问它的 Web 页。

大多数的 Web 浏览器也能访问 ftp 和 Gopher 站点。您可以发现用浏览器访问一个 ftp 站点比直接用 ftp 访问要方便容易。目录和文件都将被直接显示，只须单击名字即可选择它们。访问 ftp 站点时，首先要输入 ftp://和 ftp 站点的 Internet 地址。将显示一个目录的内容，包括文件和子目录。单击目录名即可移到此目录中，单击文件名即可下载该文件。您可以看到一项“..”，代表父目录。您能移到一个目录的子目录中，并通过选择“..”而退回到此目录中。返回

的您自己的主页，即退出 ftp 站点。您还可以用浏览器访问 Gopher 站点。输入 gopher://和 Gopher 站点的 Internet 地址。您的 Web 浏览器将显示此站点的主 Gopher 菜单。接着就可以从一个 Gopher 菜单移到另一个。

大多数的浏览器可以连接到您的新闻服务器中来访问特定的新闻组或文章。这是一个本地操作，访问您已连接的新闻服务器。您只须输入 nntp 加一个冒号和新闻组或新闻文章。一些浏览器，象 Netscape，提供一个附加的新闻阅读器浏览器，用它可以访问远程的新闻服务器。如先前所述，在 Linux 中有几个流行的浏览器可以使用。在此将介绍四个颇有特色的浏览器：Netscape Navigator，Mosaic，Arena 和 Lynx。Netscape 和 Mosaic 是基于 X-Windows 的 Web 浏览器，它们能显示图形、视频和声音，也可以操作新闻阅读器和邮件程序。Arena 是设计用来支持 HTML3 的新特征的浏览器，它支持用 Markup 语言开发的 Web 页。Lynx 是一个基于命令行的浏览器，不能显示图形。除此之外，它们每一个都是全功能的 Web 浏览器。您的 Caldera Lite Network Desktop 中有 Arena 和 Lynx。您可以很容易的下载 Netscape 和 Mosaic，以及 Arena 和 Lynx 的最新版本。Netscape 是一个商业产品，所以它有 90 天免费试用版，Mosaic，Arena 和 Lynx 都是免费的。

### 11.3.1Netscape Navigator

超文本数据库是设计用来访问任何类型的数据，无论是文本、图形、声音或视频。然而，您是否能访问这些数据要依靠您使用的浏览器的类型。一个最常用的浏览器使 Netscape Navigator。不同版本的 Netscape 可以使用在不同

的操作系统之上，象 X-Windows，Microsoft Windows 和 Macintosh。用在 X-Windows 之上的 Netscape 浏览器能显示图形、声音、视频和基于 Java 的程序（本章的后面会讲到 Java）。您能从站点 www.netscape.com 上得到更多的关于 Netscape 的信息。

从 Caldera 的 ftp 站点 ftp.caldera.com 上您能取得一个 Netscape Navigator。第三章中讲述了如何下载和安装 Netscape Navigator 浏览器的 RPM 软件包。从 Netscape 的 ftp 站点 ftp8.netscape.com 上您能得到一个压缩的版本。您可以把压缩的文件放到一个安装目录中，象/usr/local，用命令 gunzip 和 tar xvf 解压和展开。

一旦您有了 Netscape 浏览器，便开始准备使用它。Netscape Navigator 是一个基于 X-Windows 的应用，您能从桌面中操作它。如果您还未启动 X-Windows，用 startx 命令启动它。接着用一个目录窗口打开您放置 Netscape Navigator 的目录，找到 Netscape 的图标并双击它即可启动 Netscape Navigator。您可以把 Netscape 的图标移到桌面上以易于访问。Netscape Navigator 的图标如下所示：

Netscape Navigator 中显示一个窗口（见图 11-2），在窗口的上方有一系列操作 Web 页的按钮和一个用于输入 URL 地址的方框。最上方的下拉菜单能访问 Netscape 的特性。为了访问一个 Web 站点，您需要在 URL 框中输入它的地址并按 ENTER。

图标工具栏，如图 11-3 所示，在浏览器的上方，有一些按钮可以用于移动 Web 页和进行其他的操作。Back 和 Forward 按钮使您在一列您访问过的 Web 页中先后或向前移动。Home 按钮（有个房子的图标）将退出一个 Web 站点使

您回到您的系统中。这还有一个带有停止标志的 Stop 钮，当您连接和显示一个新的 Web 页时，它将被击活。如果 Web 页太长，您可以单击 Stop 钮终止此过程。

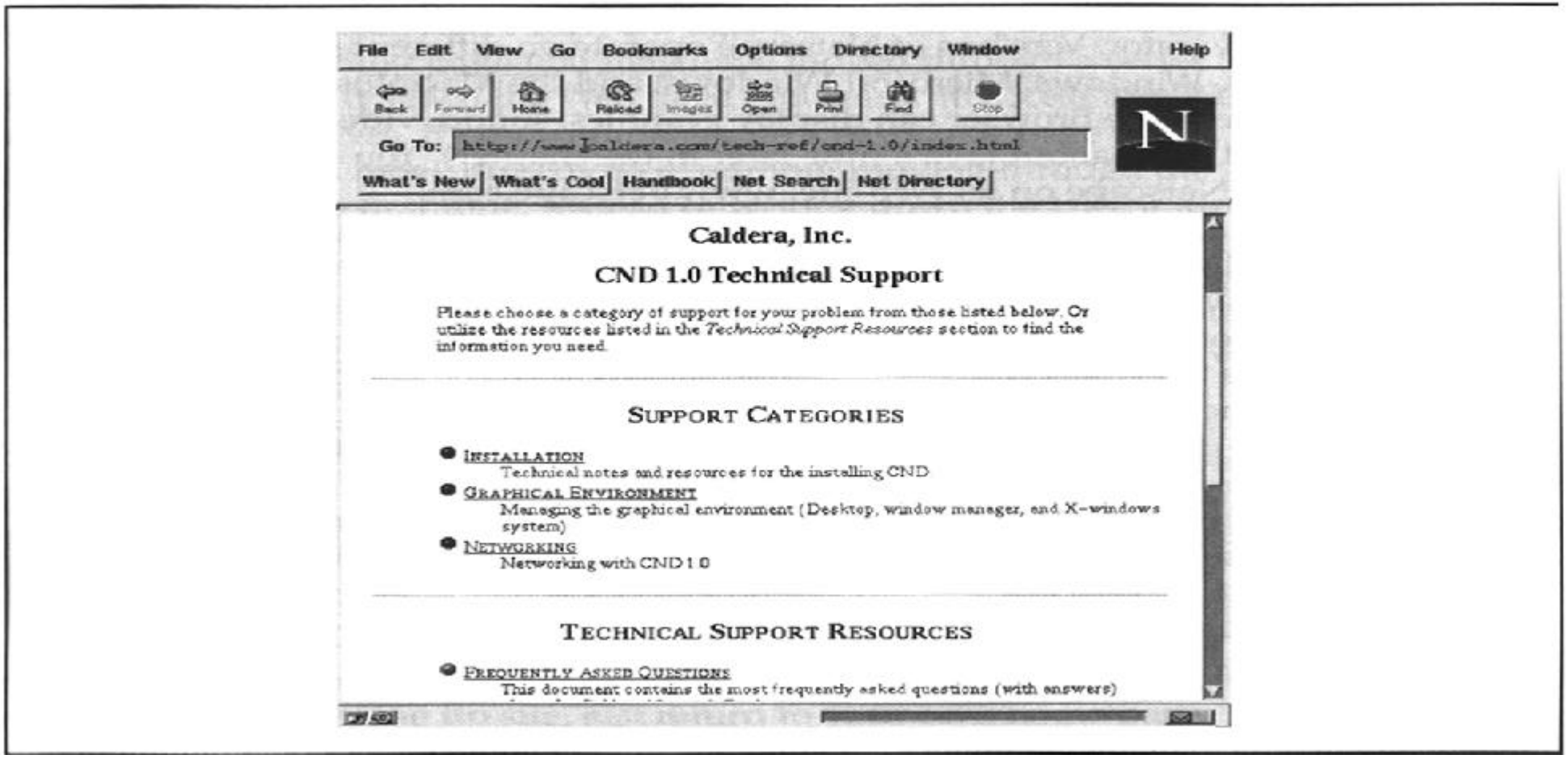


图 11-2Netscape Navigator 浏览器


图 11-3Netscape Navigator 图标栏

Netscape 用书签保存放在活动表中的 Web 页的 URL，它将标记您想直接访问的页。书签菜单允许您加入最喜爱的 Web 页到活动表中，然后您可以调用您的书签并从中选择所需要的 Web 页。

通过 Windows 菜单中的 Mail 项，您可以打开一个全功能的邮件客户程序，用它可以在 Internet 上发送和接收邮件。同样在 Windows 菜单中的 News 项，使您打开一个全功能的新闻阅读器，通过它可以在 Usenet 新闻组中阅读和发表文章。从这也可以说明您的 Netscape Navigator 不仅是一个 Web 浏览器，而且是一个邮件程序和新闻阅读器。

在 Windows 菜单中的其余项用来增强对 Web 浏览器的操作。在地址簿里您可以保存一列 Web 站点的 URL。书签项让您能编辑书签列表，加一项或删除一项。History 项中有一列您已访问过的 URL。如果您想退回到一个未被加到书签中的已访问过的 Web 页，您可以从历史列表中找到它。您还可以用 Netscape 收发邮件和访问 Usenet 新闻组。

在 Netscape Navigator 的 Option 菜单中，可以为您的浏览器设定不同种类的优先选项。您可以设置邮件、新闻、网络和安全的选项，还可以设置别的普通选项。在普通选项中，可以决定您的主页和您希望显示的工具栏。对邮件和

新闻选项，可以输入您使用的 Internet 上的邮件和新闻服务器。Netscape 能被设置用 NNTP 协议访问您订阅的新闻服务器。您可以在新闻服务器之间任意切换。如果您通过防火墙连接到 Internet 上，需要在 Proxies 屏幕中输入您的网络中作为防火墙网关的计算机的地址。一个防火墙是指一台计算机，用它做为 Internet 和您的网络的控制网关。防火墙有很多中类型。一种对操作有着严格限制的防火墙是 proxies，它收到用户的访问 Internet 请求，作为自己的请求执行访问。用户并没有直接连接到 Internet。配置它要从 Option 菜单中选择 Network，接着选择 Proxies 屏幕。在此输入网络中作为防火墙网关的计算机地址。

为了保存一个 Web 页，选择 File 菜单中的 Save as 项。它将打开一个带有缺省目录的对话方块。其中包含三个框，最上端的框是一个过滤器。最底端的是文件名。中间是当前目录中的不同目录。您可以输入您的路径和文件名，或单击显示在中间框中的“..项”移回到目录树中。您可以单击目录名以移到此目录中，当您到达您希望的目录，就可以保存您的 Web 页。

### 11.3.2Mosaic

Mosaic 是第一个为 Web 设计的基于图形的浏览器，它能显示图形、声音和视频数据。不像 Netscape，它是免费供用户使用。基于不同系统象 X-Windows，Microsoft Windows，Macintosh 的 Mosaic 有不同的用户界面。它是由 University of Illinois at Urbana-Champaign 的 Center for Supercomputing Applications(NCSA)开发的。关于 Mosaic 的更多的信息可以在 NCSA 的 www.ncsa.uiuc.edu 站点中找到。从 Mosaic 的 ftp 站点 ftp.ncsa.uiuc.edu 中的

/Web/Mosaic/Unix/binaries 目录中可以下载一个 Mosaic。此目录中显示了包含了不同的版本的目录。那些目录名中有一个 b 的代表当前正在测试的一个新的 beta 版本。当前可用的 Mosaic 的版本是 2.6，位于目录 2.6 中。在目录 2.7b 中也有一个 2.7 版的测试版本。切换到一个目录并用 get 命令下载此软件。

```
# ftp ftp.ncsa.uiuc.edu
Connected to ftp.ncsa.uiuc.edu.
220 curley FTP server (Version wu-2.4(25) Thu Aug 25 13:14:21 CDT 1994) ready.
Name (ftp.ncsa.uiuc.edu:root): anonymous
331 Guest login ok, send your complete e-mail address as password.
Password:
230-
230-Welcome to NCSA's new anonymous FTP server! I hope you find what you are
230-  looking for. If you have any technical problems with the server,
230-  please e-mail to ftpadmin@ncsa.uiuc.edu. For other questions regarding
230-  NCSA software tools, please e-mail softdev@ncsa.uiuc.edu.
230 Guest login ok, access restrictions apply.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> cd /Web/Mosaic/Unix/binaries
250 CWD command successful.
ftp> cd 2.6
250-Please read the file README-2.6
250-  it was last modified on Fri Jul  7 14:31:14 1995 - 267 days ago
250 CWD command successful.
ftp> ls
200 PORT command successful.
150 Opening ASCII mode data connection for /bin/ls.
total 22596
drwx------   2 101      10           2048 Jul  7  1995 .
drwxr-xr-x   6 12873    wheel        2048 Oct 25 17:51 ..
-rw-r--r--   1 101      10         797705 Jul  7  1995 Mosaic-ibm-2.6.Z
```

```
rw-r--r--   1 101      10         915718 Jul  7  1995 Mosaic-indy-2.6.Z
-rw-r--r--   1 101      10         903973 Jul  7  1995 Mosaic-linux-2.6.Z
-rw-r--r--   1 101      10         648431 Jul  7  1995 Mosaic-sgi-2.6.Z
-rw-r--r--   1 101      10        1708074 Jul  7  1995
   Mosaic-solaris-23-2.6.Z
-rw-r--r--   1 101      10           1835 Jul  7  1995 README-2.6
ftp> binary
200 Type set to I.
ftp> get Mosaic-linux-2.6.Z
200 PORT command successful.
150 Opening BINARY mode data connection for Mosaic-linux-2.6.Z (903973 bytes).
226 Transfer complete.
903973 bytes received in 276 secs (3.2 Kbytes/sec)
ftp> get README-2.6
200 PORT command successful.
150 Opening BINARY mode data connection for README-2.6 (1835 bytes).
226 Transfer complete.
1835 bytes received in 0.74 secs (2.4 Kbytes/sec)
ftp> close
221 Good-bye.
ftp> quit
# exit
```

此软件是一个.Z 压缩的文件，不像 Netscape 文件那样是一个存档的 gzip 的压缩文件（.tar.gz）。用 gunzip 解压它，而不是用 tar。这将产生一个叫做 Mosaic-linux-2.6 的文件。用命令 chmod 把文件改变成为可执行的，755。接

着把此文件放在您想访问的目录中。下面的例子中，使用者首先解压文件，然后把它变为可执行，并把 Mosaic 浏览器移到/usr/local目录下。

```
$gunzip Mosaic-linux-2.6.Z
$chmod 755 Mosaic-linux-2.6.Z
$mv Mosaic-linux-2.6.Z /usr/local
```

象 Netscape 一样，Mosaic 是一个 X-Windows 的应用。在您运行它之前，必须启动您的桌面。在桌面中，打开您放置 Mosaic 浏览器的目录，找到它的图标并单击它。为了方便使用，可以把 Mosaic 图标拖拉到您的桌面上。Mosaic 窗口将打开、显示您的 Web 主页(参见图 11-4)。

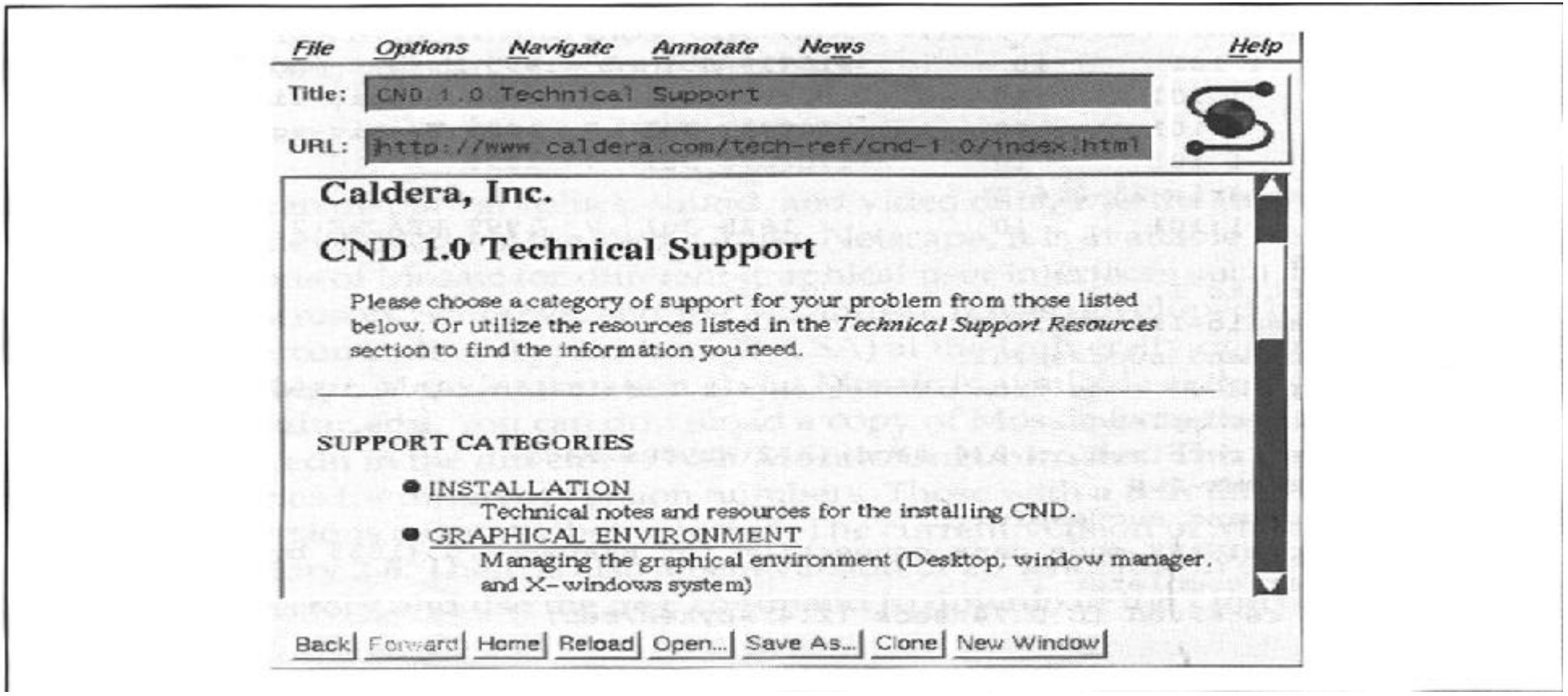


图 11-4Mosaic Web 浏览器

在 Mosaic 窗口中有两个方框。第一个显示当前 Web 页的标题。第二个用来输入 URL。然后显示 Web 页。在窗口的底部是一系列按钮，用它们来控制对 Web 页的访问。Back 和 Forward 按钮能使您在一列访问过的 Web 页中向后或向前移动。Home 按钮能使您返回您的主页。Save as 按钮能把当前页保存到您的系统中。

当您开始访问一个 Web 页时，屏幕右上角的球体图象将开始转动。当完成 Web 页的显示后，球体将停止转动。当球体转动时，您若是不能决定是否访问

此 Web 页，可以单击此球体以停止访问。从这方面讲，球体图标充当停止按钮的功能。有时，访问某一个 Web 页需要大量的时间，您可以单击转动的球体以停止访问。

在 Mosaic 窗口上端的菜单允许您管理您的 Web 查找。在 Navigate 菜单中，您管理一列最喜欢的 Web 站点的活动列表。Options 菜单中有几个项用来配置您的 Mosaic 浏览器。可以设置您的主页或者指定邮件或新闻服务器。也能设置缺省的浏览器的背景颜色和 URL 连接的颜色。News 菜单使您能用 Web 浏览器来访问 Usenet 新闻组，显示和保存文章。Mosaic 也带有一些安全的特征用来保护您的系统。

### 11.3.3Arena

Arena 是一种实验性的浏览器,它是设计用来测试新的 HTML 3 语言和 WC3 函数库。Arena 支持表格、数学公式和样式表。它不具有 Web 浏览器的所有功能，然而，它将被用做开发基于 HTML3 的 Web 页的工具。Arena 被安装在您的 Caldera Lite 桌面中并作为您的缺省浏览器。它的显示屏幕的顶端是一些命令按钮，往下是一个 URL 输入区域，接着便是 Web 页的显示区域。

在 Arena 浏览器的上端的一系列按钮用来控制移动和执行命令（见图 11-5）。为了输入一个新的 URL，您应首先按一下 Open 按钮。它将清除 URL 区域使您能输入新的 URL。用 CTRL-h 退格。用 Abort 按钮可以离开 URL 区域并将取消一个 Web 访问的操作。Forward 按钮使您移到前一页，Back 按钮使您退回到先前的显示页。Home 按钮将返回到您主页。别的按钮的功能和其他的 Windows 应用中按钮的功能。

一个对开发 HTML3 页十分有用的命令是 View 按钮。当您按 View 按钮，Web 页区域将显示一个 HTML 文档，其中包括 ASCII 文本和 HTML 标识符。通过它能看到您的 Web 页是如何组成的。如果您想了解一个特殊功能是如何实现的，可以单击 view 来看 HTML 标识符是如何使用的。

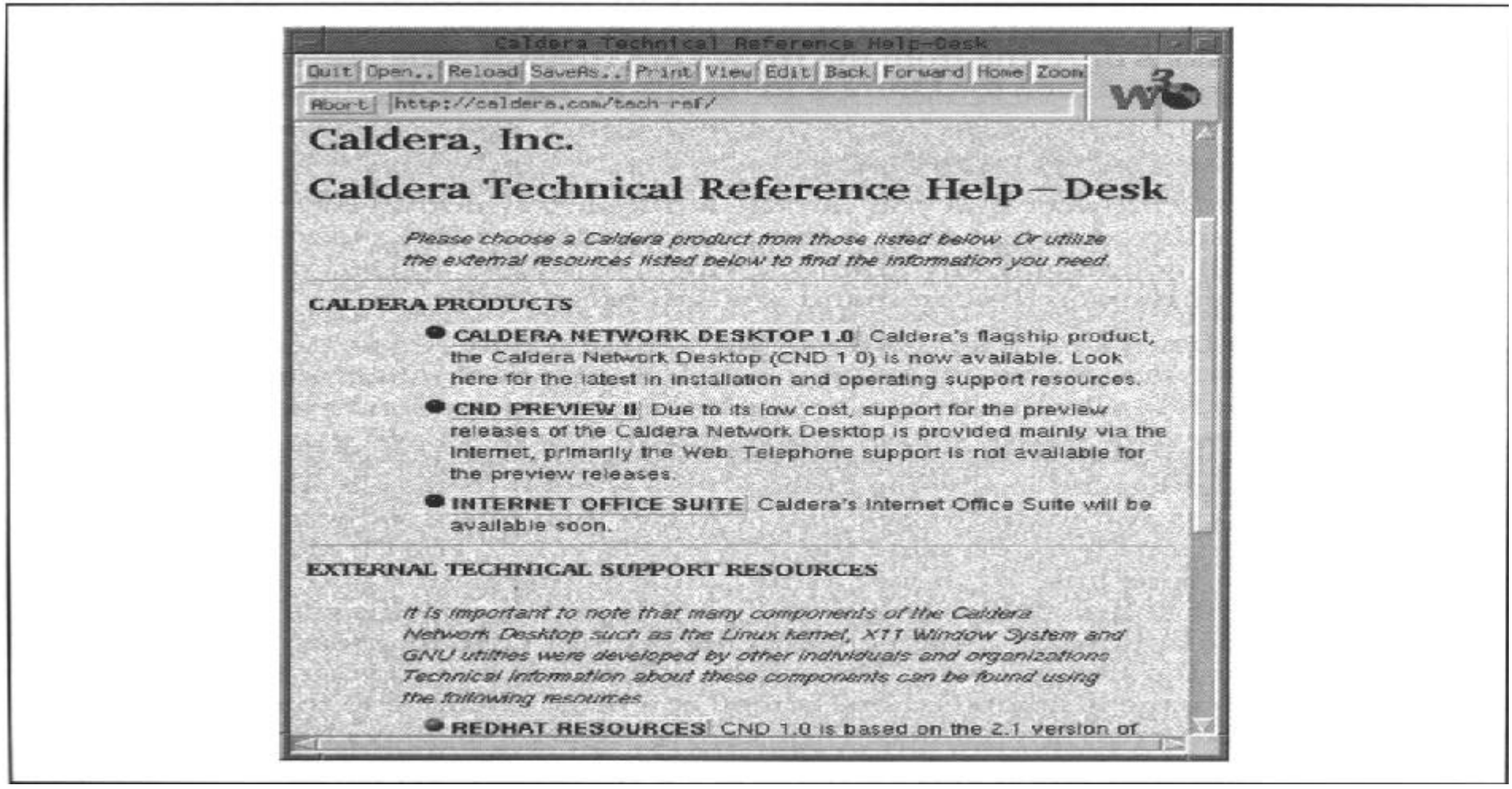


图 11-5Arena 浏览器

通过相应的 shell 环境变量，可以设置您的主页，缺省的编辑器和任何的 proxies。您可以把变量 WWW_HOME 设置成您需要的主页的 URL。为了设置缺省的编辑器，您可以把变量 EDITOR 设置成您想使用的编辑器的名字。如果需要指定一个 proxy，您可以把变量 http_proxy 设成 proxy 的 URL。Shell 环境变量通常设置在您的.bash_profile 或.bshrc file 中。在 BASH shell 中，您需要使这些环境变量发生作用，下面显示一些例子。如果您用 C-shell，需要使用命令 setenv。

```
WWW_HOME=file://home/httpd/html/myfile.html
EDITOR=vi
http_proxy=pango1.train.com
export WWW_HOME EDITOR http_proxy
```

为了了解关于 Arena 项目的更多信息，单击 Help 按钮。它将使您连接到 Arena 的主页 www.w3.org/hypertext/WWW/Arena/。在此您可以找到关于此项目的讨论和 HTML3 的指南。您也可以从 ftp 站点 ftp.w3.org 中的/pub/arena 目录中下载关于 Arena 的当前版本。当前

Arena 的 Linux 版本在文件 arena-beta-2b-linux.tar.gz。

### 11.3.4Lynx：行模式的浏览器

Lynx 是一种行模式的浏览器，您可以不使用 X-Windows 而直接使用该浏览器（见图 11-6）。一个 Web 页仅以文本方式显示。文本页可以包含到其他 Internet的链接,但不能显示图形、视频或声音。在命令行上输入 lynx 并按 ENTER 即可启动 Lynx 浏览器。

$lynx

文本中的链接用黑体显示并分散在 Web 页中。被选中的链接以反白增亮显示，并且在链接文本的周围有一个阴影框。第一个链接被自动选中。按 DOWN ARROW 键将顺序移到页中的下一个链接。UP ARROW 退回到上一个链接。为了选中一个链接，您首先使其增亮，并按 ENTER 或 RIGHT ARROW 键。如果您想到一个特定的站点，按 g。将在屏幕的底部打开一行，并提示 URL to open：。在此能输入您想去的站点的 URL 地址。按 m 将返回到您主页。Web 页的文本分屏显示，按 SPACEBAR 或 PAGE DOWN 即可移到下一屏。PAGE UP 显示文本的上一屏。您可以按 DOWN 或 UP ARROW 移到文本的下一行或上一行，以显示链接周围的文本。按=键可以显示对当前 Web 页和它的 URL 的描述。

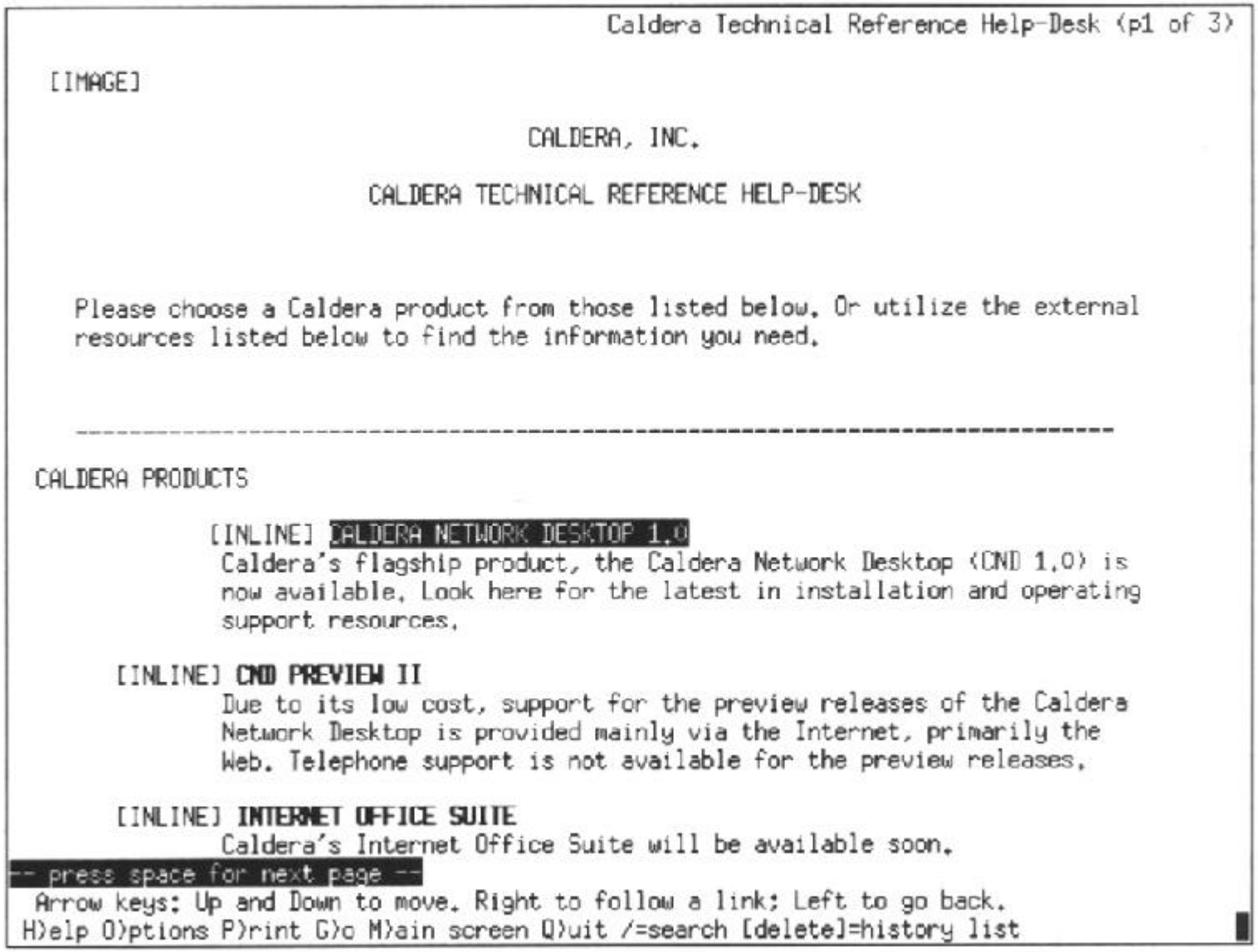


图 11-6Lynx 浏览器

Lynx保存有您在一个对话期内访问过的所有Web页。LEFT ARROW将使您移到上一个显示的页。RIGHT ARROW将移到列表中下一页。Lynx把此列表称为历史列表。按DEL可以直接显示此历史列表。您能在此列表中用UP和DOWN ARROW上下移动以选择一个特定的Web页的连接并用RIGHT ARROW或ENTER来访问它。Lynx也支持书签。按a，将自动把当前Web页加到一个书签文件中。按v将显示一列书签。象在历史列表中的操作一样，您能用UP和DOWN ARROW来选择一个书签连接。按RIGHT ARROW或ENTER将移到并显示此Web页。

Lynx用一组单字符命令来执行大量的浏览器的功能。按下？键，能显示命令列表。例如按d键将下载一个文件。按h键将显示帮助页。为了查找您当前的Web页文本，按/键，它将在屏幕底部打开一个输入框，您可以输入查找模式。Lynx将增亮此模式在文本中出现的下一处。按n，Lynx将查找此模式下一次出现。用\键将在当前此模式页的原始版本和着色后的版本之间切换，为您显示HTML标识符或格式过的文本。

除了显示限制外，Lynx可以称为使全功能的Web浏览器。您能用Lynx下载一个文件或建立一个telnet连接。能访问Web上的所有信息。因为它不需要执行一些图形浏览器要执行的额外操作，所以它执行起来更快，能快速的显示Web页的文本。此特征使它成为一个较理想的公共Web客户程序。您能够telnet到一个公共的Web客户端，象info.cern.ch，用它的行模式浏览器来访问Web，而不是从您的系统访问。当然，因为您是在操作远程的系统，所以不能下载文件。如果在您当前的系统中没有浏览器来访问Web时，这是一个简单的访问Web

的方法。

## 11.4HotJava

HotJava 现在已有 Linux 的版本。从 Blackdown Web 站点上可以下载它。

http://www.blackdown.org/java-linux.html

选择和单击 Java Products 选项。浏览器是一个压缩文件.tat.gz。首先把此文件移到您想安装 HotJava 的目录中，比如/usr/local。接着用 gunzip 和 tar 解压和展开它，或用下面所示的 tar xvzf。

```
#mv hjb1-0sd-linux-nort.tar.gz  /usr/local
#cd /usr/local
#tar xvzf hjb1_0sd-linux-nort.tar.gz
```

然后将创建一个名为 HotJava1.0 的目录，其中包括 bin 和 lib 两个子目录。启动 HotJav

a 浏览器的命令在目录 Hotjava1.0/bin 中，名为 hotjava。您应该把路径名/usr/local/HotJava1.0/bin 放到您的/etc/profile 文件或.profile 脚本中的 PATH 变量里。然后可以用如下所示的命令启动 HotJava。

```
#hotjava
```

HotJava 使用 JDK1.1.1。您应首先安装 JDK1.1.1，并把变量 JDK_HOME 设置为 JDK1.1.1

在您的系统中的路径，例如，/usr/local/jdk1.1.1。

图 11-7 显示一个 HotJava 浏览器。HotJava 浏览器能按您的需要进行设置。

它的 Places 菜单允许您收集 Web 页地址或使用 Netscape 书签。它也能显示图文框和表格。HotJava 支持一些安全特性，象带符号的 applet。在读取 HTML 代码时更灵活，允许浏览器更正一些错误。HotJava 也支持 Unicode2.0 字符集，它允许显示拉丁或非拉丁字符，象中文、日文和韩文。

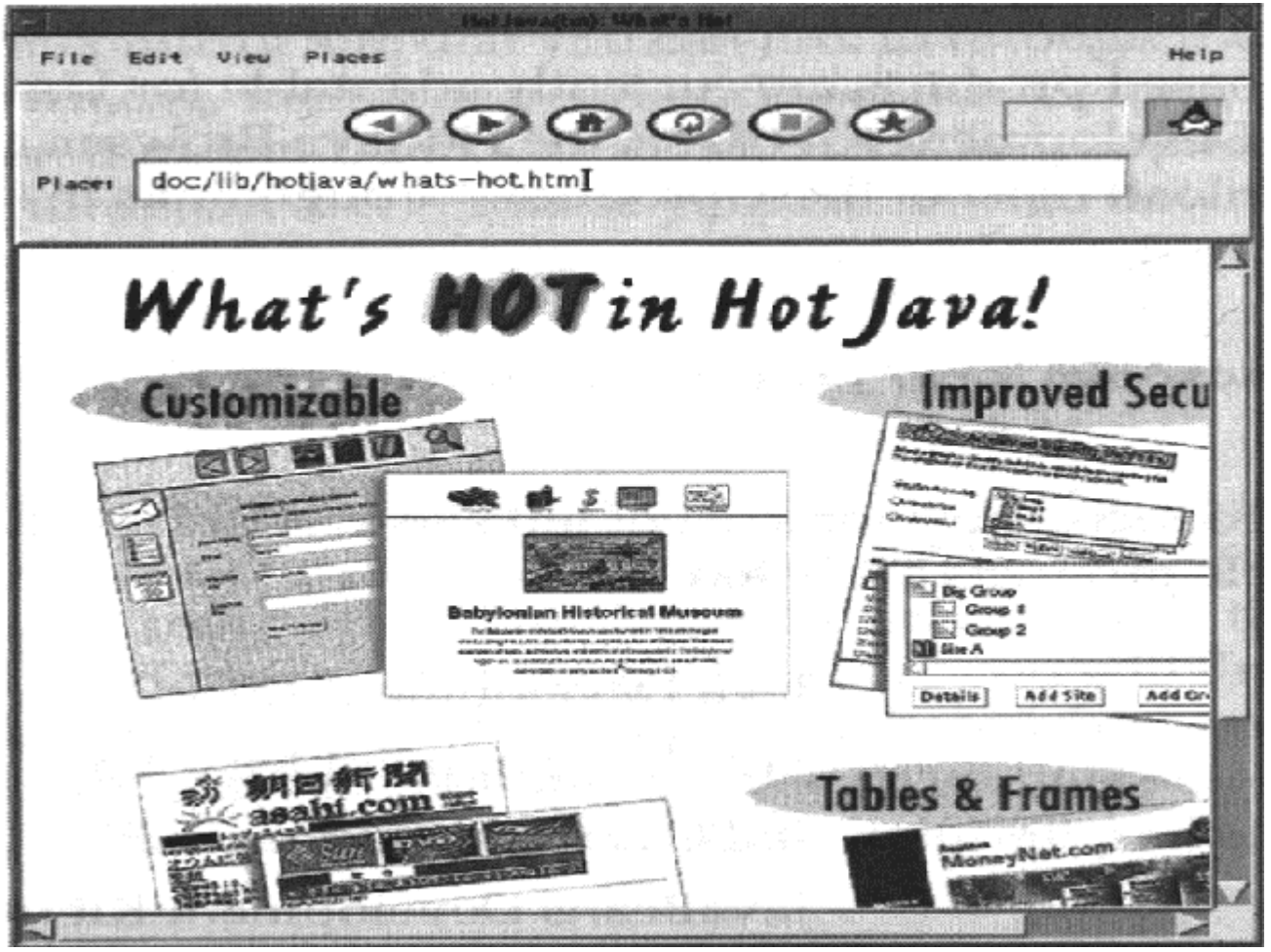


图 11-7HotJava Web 浏览器

### 11.4.1Web 查找功能

在 Internet 上查到您想要的特定的 Web 页将是一个很慢的过程。通常，您不得不在浏览器中从一个站点连接到另一个站点直到查到您需要的一个 Web 页。在这方面，Web 页的连接是非常受限制的。为了查找一个 Web 页，您可以使用一个索引或一个程序来帮助查找。一个索引通常被作为 Internet 的黄页，它允许您使用复合查询来查找 Web 页。一个最流行的索引是 Yahoo 站点，http://www.yahoo.com。您能在此查找关于任何主题的 Web 页。

您也可以用一个已知的程序 spiders 来逐个站点的查找 Web 页。一些流行的 spiders 是 WebCrawler 和 World Wide Web Worm（WWWW）。一个 spider 不仅能显示一个给定主题的页，而且列出从此页到相关主题的连接。另外，一个 spider 不但显示您查找的页，还列出到 Web 上其他页的连接。您能从 www.cs.colorado.edu/home/mcbryan/WWWW.html 站点上取得取得 WWWW spider。WebCrawler 在站点 www.biotech.washington.edu/WebCrawler/WebCrawler.html。

## 11.5Linux 的 Java

现在已有大量的基于 Java 的产品可用于 Linux 中。有 Linux 版本的 Java 开发软件包（JDK），HotJava 浏览器和 Java 的 Web 服务器。您能从 Blackdown 的 Web 站点上下载这些产品，Blackdown 的 Web 页为：

http://www.blackdown.org/java-linux.html

您可以从一列可用的 Java 产品中选择需要的项。下面列出一些产品：

JDK1.0.2Java 开发软件包 1.0

JDK1.1.3 Java 开发软件包 1.1.3

JDBC JDBC 数据库访问 API

BDK JavaBeans 开发软件包(要求 Java 开发软件包 1.1.1)

Java Web Server

JSDK Java Servlet 开发软件包

JWS20 工作组 2.0 Build 125

HotJava HotJava 1.0 Web 浏览器（要求 Java 开发软件包 1.1.1）

BISS-AWT Java Framework 一个免费的 Framework/AWT 代替品

Castanet Tuner 1.1

## 11.6Java 开发软件包：JDK

Java 开发软件包是一种工具，用它来可以创建和调试您的 Java 程序，并对 Java 应用提供支持，象 HotJava 浏览器。软件包中包含演示的 Java 程序和源代码。从 Sun 的 Web 站点您能取得关于 JDK 的详细的文件：

http://java.sun.com/products/jdk/

现在有两个版本的 JDK 可用，1.0 和 1.1。1.0 是一个老的版本，它被大多数的浏览器支持。1.1 是一个较新的版本，有许多老版本的浏览器不支持它。

JDK1.1 包括一些先进的功能，比如 JavaBean 和数据库连接。两个 JDK 版本都可用于 Linux。您能从一个 Java-Linux 的镜像站点下载它们，从前面提到的 Blackdown Web 页能连接到此镜像站点。选择“Geting the JDK”项即可。在 OpenLinux CD-ROM 上也有 JDK1.0。许多的 Java 的应用象 HotJava 浏览器需要 JDK1.1。为了使用它们，您需要下载和安装 JDK1.1。JDK1.1 也有两个版本 JDK1.1.1 和最新的 JDK1.1.3。

### 11.6.1JDK1.0.2

Java 开发软件包的 1.0.2 版本在您的 OpenLinux CD-ROM 上的 contrib 目录中，作为一个 RPM 软件包。它也是 JDK1.0 的最后一个版本。安装之前，首先要安装上您的 CD-ROM 并切换到它的 contrib 目录下，进入 RPMS 子目录，您将发现 JDK 的静态的和共享的两个版本。共享的版本是为那些有 Motif 共享库的用户使用的。对于 OpenLinux Lite，安装静态的版本。用命令 rpm -i 安装 JDK 软件包（在目录 contrib 中您不能用 Lisa 安装软件）。

```
#mount /mnt/cdrom
#cd /mnt/cdrom/OpenLinux/contrib/RPMS
#ls JDK*
JDK_shared-1.0.2.p12-3.i386.rpm    JDK_static-1.0.2.p12-3.i386.rpm
#rpm -i JDK_static--1.0.2.p12-3.i386.rpm
```

JDK 软件包在目录/usr/local/java 中安装 Java 应用软件、库函数和演示程序。Java 应用软件包括一个 Java 编译器，javac，一个 Java 调试程序，jdb，和一个 applet 查看程序，appletviewer。为了使用这些应用程序，您需要把目录/usr/local/java/bin 加到您的变量 PATH 中，变量 PATH 在文件/etc/profile

或.profile 中（参见第 15 章）。appletviewer 位于目录/usr/local/java 中，您要把此目录也加到 PATH 中。被包含的应用软件列在表 11-1 中。

### 11.6.2JDK1.1.1

JDK1.1 提供了比 JDK1.0 更多的功能。JDK1.1 支持国际化、带符号的 applet、JAR 文件格式、增强的 AWT（window 工具集）、JavaBeans 组件模式、增强的网络功能、针对大数的数学软件包、远程方法调用（RMI）、反射、数据库的连接（JDBC）、新的 Java 界面、串行的对象、内部的类和增强的执行功能。关于这些特征的具体描述可以在 JDK 的文档中找到。

当前有两个版本的 JDK1.1：JDK1.1.1 和最新的 JDK1.1.3。下面将介绍如何安装 JDK1.1.1。JDK1.1.3 的安装过程也很相似。当前 JDK1.1.1 的 Linux 版本是一个压缩文件。您需要下载、解压和展开.tar.gz 文件。一旦您下载此文件，把它放在您想安装 JDK1.1.1 的目录中，通常是/usr/local。然后用 gunzip 解压，并用 tar xvf 展开，您也可以把两个操作合在一起用，如下所示的 tar xvzf 解压并展开。

```
#mv jdk.1.1.1-linux-v3.tar.gz  /usr/local
#cd /usr/local
#tar xvzf jdk.1.1.1-linux-v3.tar.gz
```

展开后将创建一个名为 jdk1.1.1 的目录，其中有包含 Java 应用软件和函数库的两个子目录 bin 和 lib。您需要把目录/usr/local/jdk1.1.1/bin 加到您的变量 PATH 中，变量 PATH 在文件/etc/profile 或.profile 脚本中（TCSH shell 是.login）。另外在/etc/profile 或.profile 中，您需要加一项，把目录/usr/local/jdk1.1.1/bin

指定给变量 JAVA_HOME，如下所示。此变量被许多 Java 应用程序象 HotJava 使用来定位 Java 解释器。

JDK_HOME=/usr/local/jdk1.1.1/bin

表 11-1JDK 工具和目录

| JDK1.0 和 1.1 工具 | 描述 |
|---|---|
| Java | Java 解释器 |
| javac | Java 编译器 |
| jdb | Java 调试器 |
| javadoc | 以 HTML 格式产生 API 文档 |
| javah | 为一个 Java 类创建 C 头文件和 C 存根文件 |
| javap | 对编译过的 Java 文件进行反编译 |
| appletviewer | 允许您在没有 Web 浏览器时运行 applets |
| | |
| JDK1.1 附加的工具 | |
| Jar | ava 存档工具 |
| JDK1.0 和 1.1 | 工具描述 |
| Javakey | 数字符号工具 |
| native2ascii | Native-To-ASCII 转换器 |
| rmic | Java RMI 转换器 |
| rmiregistry | Java 远程对象寄存器 |
| serialver | 串行版本的命令 |

| | |
|---|---|
| updateAWT | AWT1.1 转换工具 |
| Java 目录 | |
| Bin | Java 应用软件 |
| Lib | Java 类库 |
| Include | Java 包含文件 |
| Demo | Java applet 的例子 |

## 11.7Java Applets

您可以象使用标准编程语言编写程序一样来创建 Java Applets。首先用文本编辑器来编写源代码。源代码被保存在扩展名为.java 的文件中。接着可以用编译器 javac 来编译源文件，产生一个 Java Applet。这个 Applet 文件将有一个扩展名.class。例如在目录/usr/local/java/demo/blink 中有一个名为 Blink.java 的 java 源文件。您能切换到此目录中并编译文件 Blink.java，产生文件 Blink.class。

#javac Blink.java

在目录中的文件 example1.html 将调用 Blink.class applet。启动您的浏览器并调用此文件即可运行 Blink applet。

在 Web 页中使用 HTML 标识符<applet>调用 applet。此标识符中包含一些属性，其中一个是：code。您可以给 code 分配编译过的 applet 的名字。这里还有一些可选的属性，您能通过它们设置一些特性，象 applet 的显示区域和它

的排列。您甚至可以访问远程 Web 站点上的 applet。一个关于不同 applet 特性的完整列表在文件/usr/local/java/README.jdk 中。下面的例子中，一个名为 Blink.class 的 applet 将显示在 Web 浏览器的中间的方框中，此方框高 140 象素，宽 100 象素。

```
< applet code="Blink.class" width=100 height=140
align=center></applet>
```

为了调用调试器,您可以使用命令 appletviewer 加-debug 选项和运行 applet 的 HTML 文件名。

```
appletviewer -debug mypage.html
```

## 11.8 用 Linux 作为 Web 服务器

当您创建自己的 Web 站点时,Linux 是一个理想的系统。必须先把您的 Linux 系统配置成一个服务器（在第 12 章中描述）。您还需要创建 Web 页和您想放在 Web 站点上的资源。Web 页的制作并不困难。从一页连接到另一页即可以让用户在您的站点上移动。您还可以从您的站点建立连接连到其他站点上的资源中。

### 11.8.1 做为一个 Web 服务器的网络配置

您的 Caldera Network Desktop 已经配置成一个 Web 服务器。用来管理 Web 服务器的 httpd 软件也以安装运行。您要做的只是执行必须的网络服务器的配

置和指定文件和目录对远程用户开放。如果您使用另外的 Linux 版本，需要取得和安装一个 Web 服务器。表 11-3 中列出几个流行的 Web 服务器。

### 11.8.2 创建您自己的 Web 站点

为了创建自己的 Web 站点，您首先要取得一个 Internet 地址并把您的系统连接到 Internet。然后，要启动您的 Web 服务器守护进程并为您的 Web 站点的用户设置正确的使用目录。您的 Caldera Network Desktop 已经安装了 Web 服务器守护进程并为它设置了合适的目录。您只须连接到 Internet 并创建自己的 Web 页。您的 Caldera Web 服务器为您的 Web 站点页设置的目录为 /home/httpd/html。把您创建的 Web 页放到此目录中。您还可以创建别的子目录以存放连接的别的 HTML 页。在 Rich Stout 的 The World Wide Web: The Complete Reference 中有一些优秀的文本关于 Web 页的创建和管理。另外，此书还对 Web 页的结构进行了详细的描述。您能从下面的站点得到当前的关于 HTML 开发的最新信息。

http://www.w3.org/hypertext/WWW/MarkUp/MarkUp.html

### 11.8.3 用 HTML 创建 Web 页

Web 页是由 HTML 创建的，HTML 是一种超文本的链接标示语言，它是 SGML（通用标识语言）的一个子集。创建 HTML 文档就是把 HTML 标识符插入文本文件中。从这一方面讲，创建一个 Web 页就和使用一个基于标识符的字处理过程一样简单。您用 HTML 标识符格式化文本，并使其显示为一个 Web 页。Web 页本身就是一个文本文件，可以用象 vi 这样的编辑器来创建它。对于

熟悉 Unix 中的基于标识符的字处理过程的用户，您会发现它和 nroff 概念上是相似的。HTML 标识符指定标题、列表和段落，也涉及基于标识符的字处理资源。表 11-4 中列出许多常用的 HTML 标识符。

这还有另一种创建 Web 页的方法。Linux 版本的 WordPerfect 能从一个 WordPerfect 文档自动地产生一个 Web 页。您可用 WordPerfect 所有字处理的特征来创建 Web 页。还有特殊的创建 Web 页的程序象 tkWWW，它能很简单地帮您创建复杂的 Web 页，而不用输入任何 HTML 标识符。每项操作都可以通过 window 界面中的对象来完成（见表 11-3）。记住无论您使用什么工具来创建 Web 页，它本身都是一个 HTML 文档。

## 分页和文本格式

一个 HTML 标识符包含一个括在尖括号中的关键字，例如，分段的标识符是<P>。标识符通常成对使用，结束的标识符和开始的一样，只不过前面加有一个斜杠。所以您可以用<P>开始一段，并用</P>结束它。整个 Web 页应包含在<HTML>标识符中。在您的 Web 页的第一项应是<HTML>，最后一项是</HTML>。您可以把您的 Web 页分成标题和正文体两部分。标题部分包含您的 Web 页的题目，正文部分包含内容。

## 格式化文本

在 Web 页中您可用格式标识符来组织您的正文标题、段落或列表。您可用标识符<Hn>来输入一个标题，n 代表标题的级别。<H1>是顶级标题，<H2>是下一级，依次类推。依据使用的浏览器，子标题将被缩进或描述为更小的或不

同的字体。每个标题用它的结束标识符来结束，<H1>对应</H1>，<H2>对应</H2>。您可用 UL 和 LI 来显示一列项。在此列的开始和结束插入<UL>和</UL>。列中的每一项加上<LI>和</LI>。UL 显示符号列表，数字列表用 OL。

您可用 TOP、BOT 和 MID 标识符把一些组件象图片或段落放在 Web 页的顶部、底部或中间。

## 引用 Internet 资源：HREF

可以指定您的 Web 页中的某些文本或图形链接到其他的 Web 页或是其他项，比如图片或文档。这些项可以在您的系统中或在别的 Web 站点上。当用户选定一个链接的文本时，将查找它链接的 Web 页或项。为了创建链接，必须把您 Web 页中的文本或图形和一个 Internet 资源的 URL 相关联。Internet 资源可以是另一个 Web 页、一个项比如一个图片或文件，或甚至是另一个 Internet 工具，象 ftp 或 telnet。您可用 HREF 锚号标识符来创建此链接。HREF 锚号标识符开始于<A HREF=。在等号后输入用双引号括起来的 URL，用>结束此 HREF 锚号标识符。在 HREF 后，输入显示在 Web 页上用于链接此 URL 的文本，用结束锚号标识符</A>结束文本的输入。当 Web 页显示在浏览器中时，您将看到此文本被设置成特殊的颜色以指定它们引用一个 URL。您也可以包括一个图片引用，在这种情况下，单击图片便可引用 URL。它的语法如下：

```
<A HREF="URL">text</A>
```

在下面的例子中，HREF 锚号标识符将链接到在 www.caldera.com 系统上的一个 Web 页。文本“Caldera's decumentation on the Internet”将显示为特定的颜色，通常为蓝色。当用户单击此文本时，将访问 URL www.caldera.com。

传输协议是 http，将显示一个 Web 页。

<A HREF="http://www.caldera.com/doc/">Caldera's documentation on the Internet</A>如果您在文本部分包括了一个图片引用，则用户可以单击图片以引用 URL。下面将讨论图片引用。在下面的例子中，一个图片文件 books.gif 将显示在 Web 页上，使用者单击 book 图片将引用 URL。图片引用是<IMG SRC="book2.gif">。<H3><A HREF= “ http://www.caldera.com/doc/gs/gs.html"><IMG SRC="book2.gif">

Getting Started with the Network Desktop(Internet)</A></H3>

为了访问您的本地系统中的 Web 页，不需要传输协议和主机名。您只须指定路径名。路径名可以是您当前的 Web 页所在路径的相对路径。在下面的例子中，Web 页 desktop.html位于您系统的 lg 子目录中。

<H3><A HREF = "lg/desktop.html"> Desktop User's Guide </A></H3>

除了引用一个 Web 页外，您还能引用在相同的页中通过特定名字指定的标号文本。它也成为超文本目标或命名元素。一个 HREF 能访问此目标，直接跳到它的文本。为了在 HREF 中引用此目标，您要在它的名字之前加一个#号。在下面的 HREF 中，文本“ The newest engine ”将显示在 Web 页上。当用户单击此文本时，在 Web 页上标号名为 engine1 的目标将显示。
此目标可以是标题、图片或任何正文部分。

<A HREF="#engine1">The newest engine</A>

这里有几种方法可以创建一个超文本目标。您可用一个 NAME 锚号标识符和一个目标名再加上目标文本。下面的例子中，目标名为 engine1，文本正文是“ This is the newestengine on the block ”。此名字能被用在 HREF 中来指定特定的超文本目标。

<A NAME="engine1">The newest engine on the block</A>

在上述定义了 HREF 锚号和 NAME 锚号后，您可用 HREF 锚号来引用在 NAME 锚号中指定的文本。当用户单击在 HREF 锚号中指定的文本“The newest engine”时，将移到 Web 页中以 NAME 锚号定义的文本中的“This is the newest engine on the block”部分。此技术经常被用在使用者移到不同的标题，每个标题开始于页的不同段落。为了命名一个标题，您需要在 NAME 锚号标识符中包含完整的标题。

<A NAME=“Heading-name”>
<Hn>Heading text<＼Hn>
</A>

用您定义的标题的名字，能跳到 Web 页中标题文本所在处。

<A HREF="#Heading-name">some text</A>

在下面的例子中，engine1 现在引用的是一个标题，而不是文本。

<A NAME=“engine1”>
<H2>This is the newest engine on the block<＼H2>
</A>

在最新的 HTML 的版本，HTML3 中，您可用 ID 标识符来创建目标。您可设置任何东西为目标，包括图片。

<Hn ID="name"> Heading text<＼Hn>

## 图象和声音

能使用 HTML 命令在您的 Web 页中显示图片。图片是一个 gif 或 jpeg 文件，通常和 Web 页在同一个目录下。<IMG SRC=filename>标识符将显示一个指定

的图片。例如，为了显示一个叫做 books.gif 的图片，您把它放在 IMG 标识符中。

<IMG SRC="books.gif">

IMG 标识符将把图片作为 Web 页的一部分显示。另外，您还可以用一个单独的图片显示程序显示图片。为了做到这一点，您要用一个 HREF 标识符来引用图片文件。应用程序可以有效地链接到此图片文件并显示它。

<HREF = "engine.gif"> The greatest engine in the world </A>

您可以用相同的方法来播放声音和视频文件。这些文件依靠外部的应用来运行它们。一个 HREF 标识符将链接到声音和视频文件并用附加的应用运行它们。

<HREF = "whistle.au"> Stream melody</A>

## Web 页的例子

这儿显示的文件 cald1.html 是您的系统中 /usr/doc/HTML/calderadoc/caldera.html 文件的缩短形式。它显示讨论过的 Web 页的许多的结构特征。整页都在标识符 <HTML> 和 </HTML> 中。页标题被定义在标识符 <TITLE> 中，它放在标识符 <HEAD> 中。<BODY> 标识符将标记显示在 Web 页中的文本。<H1> 将以大字体显示一个标题。<P> 标识符将把后面的文本格式化成段落。<H2> 和 <H3> 将分级显示更小的标题。<HR> 描述一条分栏线。引用 URL 的标识符 <A HREF> 将嵌入段落和标题中间。在标题中的包含图象的标识符能被用来引用 URL。大多数的 URL 引用本地文件，另一些引用远程站点上的 Web 页。例如，<A HREF = "crisp/crisp.html> 引用一个本地的在子目录 crisp 中的 Web 页 crisp.html 。<A HREF =

"http://home.netscape.com/eng/mozilla/2.0/handbook/"> 将引用在 home.netscape.com Web 站点上的 handbook 目录。图 11-8 显示 cald1.html 如何被显示。

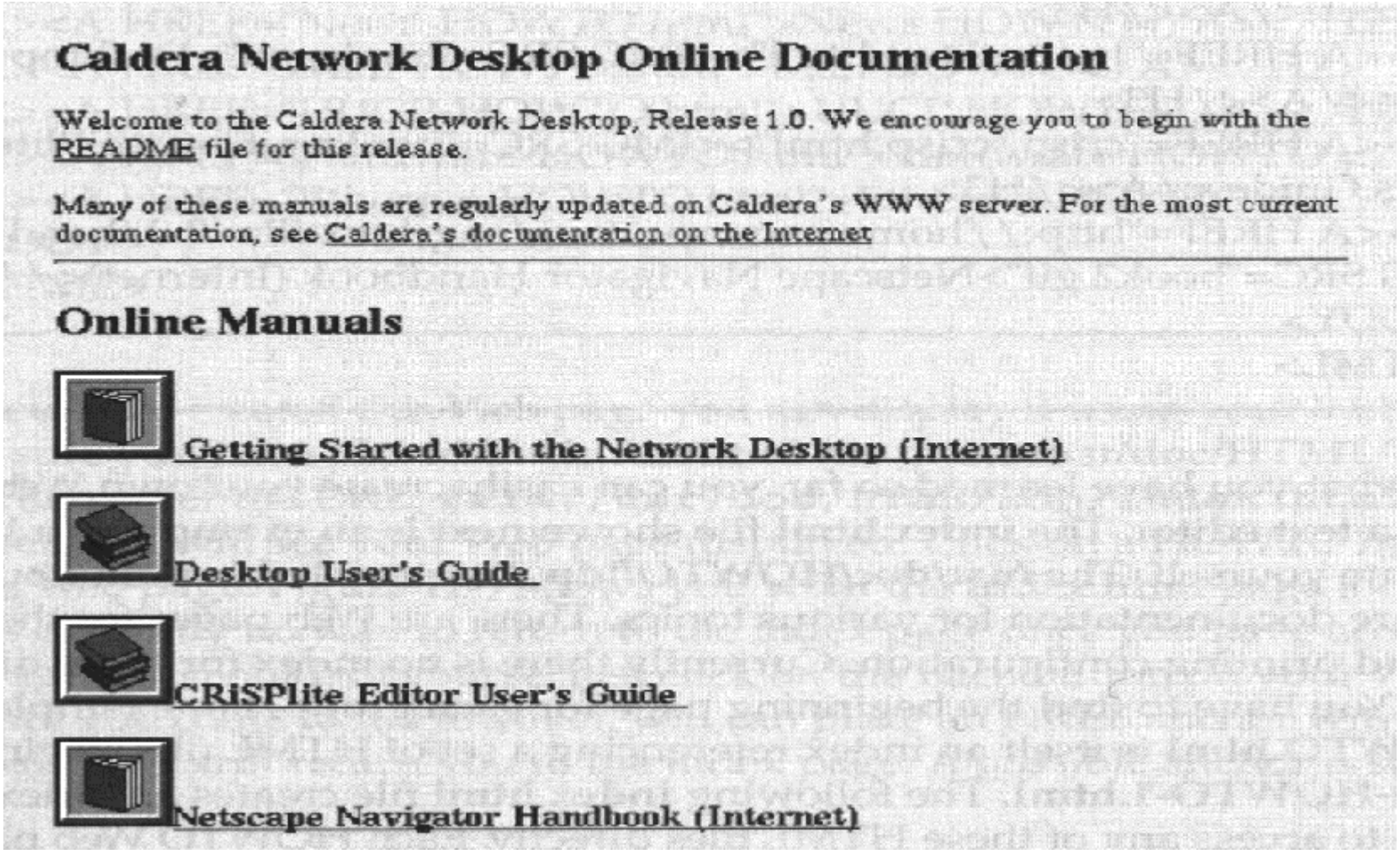


图 11-8 关于在线文档的 cald1.html Web 页

cald1.html

```
<HTML>
<HEAD>
<TITLE>Caldera Network Desktop Online Documentation</TITLE>
</HEAD>
<BODY>
<H1>Caldera Network Desktop Online Documentation</H1>
<P>
Welcome to the Caldera Network Desktop,Release 1.0. We encourage you to begin with the <A HREF="file:/etc/README">README</A>file for this release.
<P>
Many of these manuals are regularly updated on Caldera's WWW server.For the most current documentation,see<A HREF="http;//www.caldera.com/doc/">Caldera's documentation on the Internet</A>
<HR>
<H2>Online Manuals</H2>
<H3><A HREF="http://www.caldera.com/doc/gs/gs.html">
<IMG SRC="book2.gif">Getting Started with the Network Desktop (Internet)</A></H3>
<H3><A HREF="lg/desktop.html"><IMG SRC="books.gif">Desktop User's Guide</A></H3>
<H3><A HREF="crisp/crisp.html"><IMG SRC="books.gif">CRiSPlite Editor User's</A></H3>
<H3><A HREF="http:/home.netscape.com/eng/mozilla/2.0/handbook/">
```

<IMG SRC="book2.gif">Netscape Navigator Handbook(Internet)</A></H3>
</BODY>
</HTML>

现在利用您所学的知识，可以很容易地创建您自己的 Web 页。您需要的只是一个文本编辑器。下面显示的 index.html 文件便是一个您可以自己创建的 Web 页。在目录 /usr/doc/HOWTO/ldp 中有大量的 Web 页，它们是关于不同主题的文档。这些 Web 页是关于 PPP 协议和打印配置的。现在没有关于这些不同的 Web 页的索引。您不得不寻找每个组的开始页。例如，文件 Sound-HOWTO.html 是一个以它开始的一组 HTML 文件的索引。下面的 index.html 文件创建了一个索引，用它您可以直接访问其余的 HTML 文件。每个 HOWTO Web 页都有一个引用标题。例如，<H2><A HREF="PPP-HOWTO.html">PPP HOWTO</A><H2>引用文件 PPP-HOWTO.html。在 Web 页上显示文本 PPP HOWTO。单击此文本，将显示 Web 页 PPP-HOWTO。

/usr/doc/HOWTO/ldp/index.html

```
<HTML>
<HEAD>
<TITLE>Index of HOWTO Documents</TITLE>
</HEAD>
<BODY>
<H1>List of HOWTO Documents</H1>
<H2>by Dylan Petersen</H2>
<P>
```

```
v.1.0,5 April 1996
<P>
<H2><A HREF="Printing-Usage-HOWTO.html">Printing
HOWTO</A></H2>
<H2><A HREF="Sound-HOWTO.html">Sound HOWTO</A></H2>
<H2><A HREF="NET-2-HOWTO.html">Network HOWTO</A></H2>
<H2><A HREF="PPP-HOWTO.html">PPP HOWTO</A></H2>
<H2><A HREF="News-HOWTO.html">News HOWTO</A></H2>
<H2><A HREF="Mail-HOWTO.html">Mail HOWTO</A></H2>
<H2><A HREF="UUCP-HOWTO.html">UUCP HOWTO</A></H2>
<H2><A HREF="Hardware-HOWTO.html">Hardware HOWTO</A></H2>
<H2><A HREF="Ethernet-HOWTO.html">Etherenet HOWTO</A></H2>
</BODY>
</HTML>
```

一旦您创建这个用 Web 浏览器能显示的 Web 页。首先必须确定把此 index.html 文件放在目录 /usr/ldo/HMTL/ldp 中。接着在您的浏览器中输入 URL，file:/usr/doc/HTML/ldp/index.html。您将看到您的 Web 页。按 HOWTO 标题引用它的 Web 页。图 11-9 显示了 index.html 页。

现在一旦您访问一个 HOWTO 页，怎么能回到您的索引页呢？当然，可以通过浏览器中的 Back 按钮往后退，直到回到索引页。另外，还可以在 HOWTO Web 页中加一个 HREF 锚号，用它返回您的索引页。下面是一个可用的 HREF：

```
<A HREF="index.html">Return to Main Index</A>
```

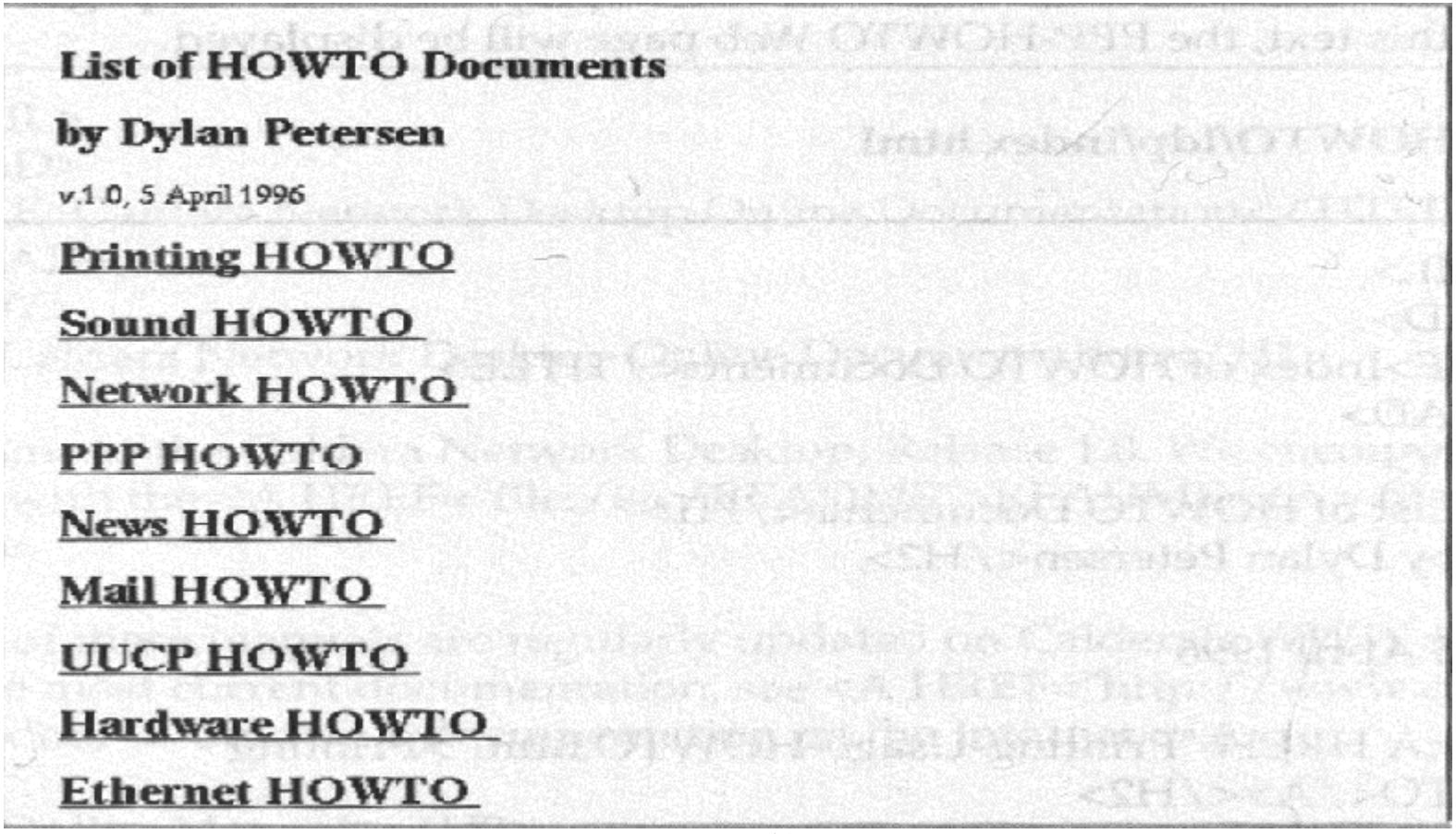


图 11-9HOWTO 页的 index.html Web 页

### 11.8.4 公共网关接口

一个公共网关接口（CGI）是一个程序，利用它在 Web 站点上的 Web 服务器可以和 Web 浏览器进行交互。当用浏览器显示在某一 Web 站点上的一个 Web 页时，Web 页可能调用 CGI 程序来为您提供一定的实时的信息或从您那得到信息。例如，一个 Web 页无论何时被访问，都可以执行服务器的 date 命令来显示当前日期。一个 CGI 脚本可以是一个 Linux shell 脚本或程序。对于 shell

脚本，您可以使用在第 15 和 16 章中描述的 BASH 或 TASH shell 命令。对于程序，您可以使用象 C 语言等编程语言来开发。还有两个特殊的 HTML 操作能被做为 CGI 脚本：查询文本和表格。它们两个都可以从特定的用户接受和执行交互相应。

您已经了解用户如何使用浏览器来显示一个特定 Web 站点上的 Web 页。事实上，用户从 Web 站点上以 Web 页的形式接收信息。用户还可以在有限的范围内发送信息给 Web 站点。这些信息通常在您的浏览器显示的 Web 页中有提示。Web 服务器能使用 CGI 程序来接收和处理这些信息。

在 Web 页上，您可使用一个单独的区域给用户进行在线的回应，或用一个有此区域的表格。<ISINDEX>标识符显示一个单独的编辑框，以供用户输入文本。它们经常用来得到一个关键字以进行一个基于文本的查询。

一个表格是一个有多个不同类型的输入区域的 Web 页。这些输入区域可以是用于输入文本的输入框，或是只须单击的 chesk 框或 radio 框。文本框可以是定义好结构的，允许您按一定的结构输入字符，象电话号码。也可以是无结构的，允许用户顺序输入，象它们的一个建议和注释。表格可以涉及基于表格的查询。

在表格中输入信息后，用户通过单击 Submit 按钮把信息送回到服务器。服务器收到表格和它的一些指令，运行特定的 CGI 程序来处理此表格。

## 11.9 总结：WWW

World Wide Web（WWW）连接了Internet上分布在全球的不同类型的信息，并把这些信息以一种易用的形式，Web页，表现出来。Web页是由超文本标记语言（HTML）编写的。一个Web页可以连接到另外的Web页，同样也可以被另外的Web页连接。这些Web页可以在同一个站点或在不同的站点上。最重要的是，每个Web页都可以显示您访问的多种信息，象文本、图片、视频和声音。在Internet上的信息资源是由URL来标识的。URL由三部分组成：传输协议，Web站点的主机名和资源的文件名。传输协议决定一个资源是如何被访问的：http访问一个Web页，而ftp进行一个文件传输。URL能被用来访问Web页或其他站点上的资源。您可以使用Web浏览器来访问遍布Internet的Web站点上的Web页。Web浏览器将保存一个对话期内访问过的不同的Web页，允许您在这些页中向前或向后移动。您也能保存一个您喜欢的Web页活动列表，用浏览器直接访问它们。许多浏览器有新闻阅读器和邮件程序的功能。Linux上可以使用一些流行的浏览器象Netscape Navigator和Mosaic。

您的Linux系统也能作为一个Web服务器，详细的关于配置Web服务器的方法将在12章中讨论（Caldera Network Desktop已经安装和配置一个Web服务器）。用Web服务器软件，您能设置您自己Web站点。Web页是一种包含HTML标识符的文本文件。您能用HTML标识符来格式化您的Web页。所以组成一个Web页只是输入文本和合适的HTML标识符。HREF锚号标识符用URL来引用Internet资源。Web页可以分布在不同的子目录中，且每个Web页都可以连接到您系统中或其他Web站点上的Web页。在一个目录中，一个

Web 页（通常是 index.html）能作为此目录的一个内容列表，并能连接到目录中其他的 Web 页。

表 11-2Web 传输协议

| 协议 | 描述 |
| --- | --- |
| http | 访问 Web 站点的超文本传输协议 |
| Gopher | 访问 Gopher 站点 |
| ftp | 匿名 ftp 连接的文件传输协议 |
| telnet | 创建一个 telnet 连接 |
| wais | 访问 WAIS 站点 |
| news | 读取 Usenet 新闻；用网络新闻传输协议（NNTP） |

表 11-3Web 资源的文件类型

| 文件类型 | 描述 |
| --- | --- |
| .html | 用 HTML（超文本标记语言）格式化的 Web 文档页 |
| 图象文件 | |
| .gif | 用 gif 压缩的图象 |
| .jpeg | 用 jpeg 压缩的图象 |
| 声音文件 | |

| | |
|---|---|
| .au | Unix 的声音文件 |
| .wav | Microsoft Windows 的声音文件 |
| .aiff | Machintosh 的声音文件 |
| **视频文件** | |
| .QT | 多平台的 Quicktime 视频文件 |
| .mpeg | 视频文件 |
| .avi | Microsoft Windows 的视频文件 |

**表 11-4 使用 Web 的工具**

| Web 浏览器 | 位置 |
|---|---|
| Netscape Navigator | ftp3.netscape.com/pub/navigator/n.n/unix<br>ftp3.netscape.com 通过 ftp8.netscape.com 的其他可用的镜像站点 |
| NCSA Mosaic | ftp.ncsa.uiuc.edu/Web/Mosaic/unix |
| Arena | http://www.w3.org/hypertext/WWW/Arena/Status.html<br>ftp.w3.org/pub/arena |
| Lynx | ftp://ftp2.cc.ukans.edu/pub/lynx(包括最大的分布) |
| HTML 编辑器 | |
| Phoenix | http://www.bsd/uchicago.edu/ftp/pub/phoenix |
| TkWWW | http://www.w3.org/hypertext/WWW/tkWWW |

| | |
|---|---|
| | ftp.aud.alcatel.com/pub/tcl/extensions |
| **Web 服务器** | |
| Apache | http://www.apache.org |
| CERN httpd | ftp://info.cern.ch/pub/www.bin |
| NCSA httpd | ftp.ncsa.uiuc eduhttp://boohoo.ncsa.uiuc.edu |
| Plexus | http://www.bsdi.com/2.2.1/dist/Plexus.html |
| Netscape Server | ftp3.netscape.com/pub/server/n.n/unix |
| **Web 实用程序** | |
| Java | ftp.blackdown.org/pub/Java/linux |
| WebCrawler | http://www.biotech.washington.edu/WebCrawler/WebCrawler.html |
| World Wide Web Worm | http://www.cs.colorado.edu/home/mcbryan/WWWW.html<br>http://info.cern.ch/WWW/Tools |

**表 11-5Web 页中 HTML 代码**

| **基本标识符** | **描述** |
|---|---|
| <HTML>Web **页**</HTML> | **把**<HTML>**作为您的** Web **页的第一项，**</HTML>**作为最后一项** |
| </HEAD> | Web **页的标题部分，包括所有的配置和题目** |
| **Web 页标题** | |
| </HEAD> | Web **页的题目，将被用在活动列表** |

| | |
|---|---|
| <TITLE>title text</TITLE> | 中以指明 Web 页 |
| <BODY><br>Web 页文本 | </BODY>Web 页的正文，即显示在 Web 页中的内容 |
| <ADDRESS><br>Web 页创建者的地址 | Web 页创作者的 Internet 地址 |
| <BASE= Href“ Web 页路径名 ”> | Web 页的路径名，它作为和此页相关的路径名的基本路径名 |
| 格式 | |
| <Hn>标题题目<＼Hn> | 标题；n 是子标题的等级，<H1>是一级标题，<H2>是二级标题 |
| <P>段落文本</P> | 段落 |
| <CENTER> | 中间的文本 |
| <BLINK> | |
| <BR> | 折行 |
| <PRE><br>Prefotmatted 文本<br></PRE> | 显示先面的不带格式的文本 |
| <HR>Horizontal rule </HR> | 显示水平的分栏线 |
| <CLEAR> | |
| 图象 | |
| <IMAGE SRC"file.gif"<br>ALLGN "position" | 显示在 Web 页中的图象<br>在 Web 页中图象或文本的位置； |

| | |
|---|---|
| | position 可以只 Bottom、Top、Left 和 Right |
| WIDTH | 设置显示图象的宽 |
| HEIGHT | 设置高 |
| 锚号 | |
| <A 锚号标识符 /A> | 锚号标识符，象 URL 引用 |
| HREF“URL 地址”> | URL 引用，联系特定的文本到 URL 地址 |
| 引用文本 | |
| <A NAME "text">锚号文本</A> | 创建一个 Web 页中的文本的锚号引用 |
| </A> | |
| <Hn>标题文本<＼Hn></A> | |
| <A NAME=“标题文本”> | 为 Web 页中的锚号引用作一个标题 |
| </A> | |
| ID“锚号文本” | 在 Web 页中用 ID 代替 NAME 来创建一个锚号文本 |
| <Hn ID=“锚号文本”>标题文本<＼Hn> | 在 Web 页中引用锚号文本 |
| <A HREF“#锚号文本”>显示在页 | 创建连接到站点上的其他 Web 页； |

| | |
|---|---|
| 中的文本</A><LINK REL=关系 | 在 Web 页的上方或下方显示具有 REL 描述的关系的按钮 |
| HREF=“ URL 地址 ”> | REL=关系<br>当前 Web 页和其他页的关系<br>Previous 前一个 Web 页的 HRL<br>Next 下一个 Web 页的 HRL<br>Home 主页的 HRL<br>Banner 所有 Web 页都显示的标题的 HRL 列表 |
| <LH>列出标题</LH> | 列表的名字 |
| <LI>列出项的文本</LI> | 列出项 |
| <UL>列出项中的实体</UL> | 未排序的列 |
| <OL>排序后列出实体</OL> | 排序列表，通常是用数字 |
| <DL>列出实体</DL> | 定义列表；一列项目和每一个的解释，叫做定义；term 使您指定的一个单词 |
| <DT>定义一个项目</DT> | 表示一个被定义的列表项的项目 |
| <DD>定义</DD> | 伴随一个定义项目的文本 |
| 表格 | |
| <TABLE>表项</TABLE> | 显示一个表 |
| <TC></TC> | |
| <TR></TR> | |

| | |
|---|---|
| <TH></TH> | |
| <TD></TD> | |
| **配置** | |
| BGCLOLR= rrggbb | **背景颜色；十六进制的数来代表颜色；**rr **红，**gg=**绿，**bb=**蓝；全为** 0=**黑，全为** 1=**白** FFFFFF**；它被设置在** BODY **标识符中** <BODY BGCOLOR=137HF2> |
| TEXT=rrggbb | **文本的颜色，也被设置在** BODY **标识符中** |
| BACKGOUND=file.gif | **被用来作为** Web **页背景的图片，它被设置在** BODY **标识符中项** |
| &lt | < |
| &gt | > |
| &amp | & |
| &quot | " |

# 第 12 章 Internet 服务器

为了反映出 Unix 和 Internet 的开发之间的紧密关系，Linux 提供了很好的 Internet 服务，像 Web，WAIS，ftp 和 Gopher。除了访问其他的站点之外，您可设置自己的 Linux 系统作为一个 Web 站点或 ftp 站点。别人可以用您创建的 Web 页来访问您的系统或下载您提供的文件。这样的系统便被称为服务器，并能提供服务。您可设置您的系统成为一个 Web 服务器或 ftp 服务器，把它连接到 Internet 中使其他用户可以访问。一个单个的 Linux 系统可以提供几种不同的服务。您的 Linux 系统可以是一个 Web 服务器和一个 ftp 服务器，也可以同时是一个 Gopher 服务器和 WAIS 服务器。一个用户可以用您的 ftp 服务下载文件，同时另一个用户可以阅读您的 Web 页。为了做到上述功能，您要为每个服务安装和运行合适的服务器软件。每项服务操作时都运行一个守护进程，用它来查找远程用户对它的操作请求。

OpenLinux 系统已经是一个 Web、ftp 和 Gopher 服务器。在设计此系统时已经把它设计为 Internet 服务器。通常您需要自己取得和安装服务器软件，但当您安装 OpenLinux 时，这些服务的服务器软件将自动安装并配置。每次您启动 Caldera Network Desktop，就启动了 Web 和 ftp 服务器守护进程。为了使您的 Linux 系统成为一个 Web 服务器，只须创建 Web 页。另外，只须把您要使之可用的文件放在 ftp 目录下，即成为 ftp 服务器。一个 Gopher 服务器也被

自动安装。

为了使您的 Linux 系统作为一个 Internet 服务器，您必须取得一个到 Internet 的连接并提供远程用户到您系统的访问。访问通常是指允许匿名注册到服务器保留的目录中。您的 OpenLinux 系统已配置允许 Web 和 ftp 用户进行访问。建立一个能容纳服务器操作的 Internet 连接并不简单。您需要一个专门的连接，或由 Internet 服务提供商提供一种连接。您不仅是自己连接到 Internet，您还要允许许多的用户通过 Internet 和您建立一个很大流量的连接。

如果您只想给本地网络提供服务，就不需要专门的连接。另外，还可以允许用户通过调制解调器连接和注册到您的系统，以提供信息服务。用户可以拨号到您的系统中使用您的 Web 页或用 ftp 下载文件。无论您想提供什么样的服务，您都需要安装合适的服务器软件并运行它。

本章将考察 Internet 的四部分服务：Web、ftp、WAIS 和 Gopher。对于 Web、ftp 和 GN Gopher 服务已经被安装，您不用按照描述的执行安装的步骤。另外，此部分将向您讲述服务被设置在什么目录中，以及如何把文件象 Web 页放到这些目录中。在您的 Caldera Network Desktop 上没有设置 WAIS，所以您要取得此服务器软件并按照本章的描述对它进行安装。

## 12.1 启动服务器

一个服务器就是一个和其他程序同时运行的守护进程，它不断的查找是否有对服务的请求，此请求可以来自您的系统的其他用户或通过网络连接到您的

系统中的远程的用户。当它收到一个来自用户的服务请求，它便启动一个会话以提供服务。例如，如果一个用户想从您的系统中下载一个文件，他能使用他自己的 ftp 客户程序来请求您的 ftp 服务器为他启动一个会话。在此会话期间，他能访问您的系统并从中下载文件。

为了使用户能访问您的服务器，您的服务器必须运行。如果您在您的系统中设置一个带有 HTML 文件的 Web 站点，在用户能访问您的 Web 站点和显示这些文件之前，您必须运行 httpd Web 服务器程序。

有一些方法可以启动一个服务器。一种方法是从命令行手工输入服务器程序的名字和它的参数选项。然后按 ENTER，服务器将被启动并且继续显示命令行提示。服务器将在您完成其他任务时同时运行。为了查看您的服务器是否运行，您可以输入下面的命令以查看当前运行的所有进程。您将能看到您为服务器程序启动的进程。

```
#ps -ax
```

您的系统可以自动地启动这些服务器，而不用每次开机后都要自己手工执行这些服务器程序。依照您想如何使用一个服务器，这有两种方法来配置自动启动。您可以让服务器自开机后便启动，并一直运行直到您关闭它；另外还可以让服务器收到用户的第一个对它的服务请求时再启动。如果一个服务器经常被使用，您可能想让它始终在运行。如果它不常被使用，您可能希望当有服务请求时再启动它。例如，如果您运行一个 Web 站点，您的 Web 服务器将随时可能收到来自 Internet 上的请求，而对于 ftp 站点来说，您可能不常收到请求，在这种情况下，您可能想让您的 ftp 服务器收到请求时再启动。当然有些 ftp 站点会收到频繁的请求，它们将被设置为不断运行的 ftp 服务器。

为了启动系统时自动启动一个服务器，需要用到目录/etc/rc.d/init.d 中的启动脚本。您可以通过配置 inetd 守护进程来指定当收到一个服务请求时再启动一个服务器。ined 将查找服务请求并在收到请求时启动服务器。您的 OpenLinux 系统已经配置自动启动并不间断运行 Web 服务器。它的配置脚本叫做 httpd.init，存放在目录/etc/rc.d/init.d 中。ftp 服务器在 inetd 下配置运行。当有人初始化一个 ftp 会话时才启动它。在配置文件 inetd.conf 中您能找到一个关于 ftp 服务器的项，而不像 Web 服务器那样，在目录/etc/rc.d/init.d 中没有关于 ftp 的脚本。

## 12.2 服务器初始化脚本

OpenLinux 用了一种补充的初始化文件，此文件在其他的系统中并不存在。对于其他系统，您可能不得不把服务器命令放在系统初始化文件 rc.local 中。对于 OpenLinux，您能使用目录/etc/rc.d/init.d 中的初始化脚本。

目录/etc/rc.d/init.d 中有一些当您启动系统时自动执行的 shell 脚本。这些脚本中有些命令可以执行特殊的程序象服务器程序。在这些程序中，您能找到一个关于 Web 服务器的脚本叫做 httpd。小心不要更改其他的脚本程序。这些都是关于您的网络接口或打印机守护进程必须的启动程序。在目录中有一个简单的启动脚本叫做 skeleton。为了给服务器创建一个新的脚本，您可以首先复制一个 skeleton 脚本。接着编辑新脚本，把服务器路径名和命令放在合适的位置。作为一个例子，在下面显示 httpd 脚本。Web 服务器程序的名字为 httpd，

$NAME=httpd。注意检查此程序是否存在的行［-f$DAEMON］，$DAEMON 设置在程序/usr/sbin/httpd 中。

```
/etc/rc.d/init.d/httpd
#!/bin/sh
#httpdThis shell script takes care of starting and stopping httpd.
PATH=/bin:/usr/bin:/sbin:/usr/sbin
NAME=httpd
DAEMON=/usr/sbin/$NAME

#Source function library.
./etc/rc.d/init.d/functions
#Source networking configuration.
./etc/sysconfig/network
#Check that networking is up.
 [${NETWORKING}="no"] && exit0
 [!-r/etc/sysconfig/daemons/$NAME] && exit0
./etc/sysconfig/daemons/$NAME
 ["$ONBOOT"="no"-a"$PROBABLY"="booting"] && exit 0
DAEMON=$DAEMON.$VARIANT
 [-f $DAEMON] || exit 0
#See how we were called.
case"$1"in
start)
#Start daemons.
```

```
if [-f/etc/httpd/$VARIANT/conf/httpd.conf] ;then
echo-n"Starting $IDENT services($VARIANT):"
start-stop-daemon -S -n $NAME -x $DAEMON— -f
/etc/httpd/$VARIANT/conf/httpd.conf
touch/var/lock/subsys/httpd
echo"."
elif ["$PROBABLY"="goofing"] ;then
echo"$VARIANT httpd not configured:Skipped!"
fi
;;
stop)
#Stop daemons.
 [-f /var/lock/subsys/httpd] ||exit 0
if [-f /var/run/$NAME.pid] ;then
echo -n"Shutting down$VARIANT httpd:"
start-stop-daemon -K -p/var／run/$NAME.pid -n $NAME
echo"."
fi
rm-f/var/lock/subsys/httpd
;;
*)
echo "Usage:httpd{start|stop}"
exit 1
esac
```

exit 0

httpd 脚本首先执行一个定义脚本中使用的函数的脚本，然后执行一个网络脚本以决定网络配置。接着它检查这个守护进程的配置文件是否存在，如存在，将执行此文件。对于 http 守护进程，配置文件名为/set/ssyconfig.daemons/httpd。这个文件中设置了一个在以后命令中将使用的变量$VARIANT 的值，在此值为 Apache。

./etc/rc.d/init.d/functions
./etc/sysconfig/network
[ ! -r /etc/sysconfig/daemons/$NAME ] && exit 0
./etc/sysconfig/daemons/$NAME

Web 服务器程序将执行脚本中下面的命令行。变量$NAME 被设置为 httpd，$DAEMON 被设置为 Web 服务器程序，/usr/sbin/httpd。-f 选项指定此 Web 服务器的配置文件，在配置文件中$VARIANT 指定一个使用的 Web 服务器，在此值为 Apache。start-stop-daemon 函数加上-S 选项将启动此守护进程。

start-stop-daemon-S-n $NAME -x $DAEMON -- -f
/etc/httpd/$VARIANT/conf/httpd.conf

start-stop-daemon 函数加上-K 选项将关闭此守护进程，用-p 选项指定进程号和-n 选项指定名字。

start-stop-daemon-K-p/var/run/$NAME.pid -n $NAME

系统中为每个运行级都设置一个子目录，这些初始化的文件能被这些子目录中的连接访问。在目录/etc/rc.d 中，有一组命名规则为 rcN.d 的子目录，在

此 N 是用来指定运行级别的数字。rc 脚本将在系统启动时决定运行级别，并只执行为此运行级别设置的子目录中的脚本文件。缺省的运行级别为 3，它是多用户级。当您启动系统时，rc 脚本将执行指定在 rc3.d 目录中的启动脚本。rc3.d 目录中有符号连接到/etc/rc.d/init.d 目录中的启动脚本（参见第 6 章关于符号连接）。所以在目录/etc/rc.d/init.d 中的 httpd 脚本实际是通过 rc3.d 目录中的符号连接并被调用。在 rc3.d 目录中连接到/etc/rc.d/httpd 脚本的是 S85httpd。为了使得服务器能自动启动，首先要在目录/etc/rc.d/init.d 中创建一个启动脚本，然后在目录/etc/rc.d/rc3.d 中创建一个符号连接到此脚本。

假设您已在系统中安装了一个 Gopher 服务器，现在想自动启动它。最简单的创建一个启动脚本 gopherd，University of Minnesota's Gopher 服务器程序的方法是把 httpd 复制一份命名为 gopherd。httpd 和 gopherd 都是网络服务器，用了许多相同的脚本。

```
#cp httpd gopherd
```

接着您要编辑 gopherd 脚本并替换 httpd 和 gopherd 中不同的地方。对于变量$DAEMON，

您应该分配 gopherd 服务器程序的路径名，如/usr/sbin/gopherd。

```
$NAME=gopherd
$DAEMON=/usr/sbin/gopherd
```

下面的行将执行 gopherd 服务器。注意命令中包括一些参数，象 Gopher 文件的目录。服务器的不同参数也有所不同。http Web 服务器没有参数。您能把引用移到变量$VARIANT 中。

```
start-stop-daemon  -S  -n $NAME -x $DAEMON /usr/lib/gopher-data
```

现在切换到目录/etc/rc.d/rc3.d 中，创建一个符号连接到

/etc/rc.d/init.d/gopherd 脚本。命令 ln 加-s 选项将创建此连接。

#ln -s /etc/rc.d/init.d/gopherd S94gopherd

以后再启动您的系统时，gopherd 服务器将自动启动，不断地运行等待的请求。

除了复制脚本 httpd 外，您还能复制脚本 skeleton。然后把 skeleton 的行替换成 gopherd 服务器的检查、启动和关闭。还可以加一些命令行来执行 /etc/sysconfig/network 脚本并检查网络是否正常连接。

您会注意到，这里没有 ftp 服务器的脚本。在 OpenLinux 中，ftp 服务器是由 inetd 程序管理的，在下一部分中会有详细的描述。如果您希望 ftp 服务器时刻运行，您可以象在操作 gopherd 服务器那样创建一个 init 脚本和一个符号连接。ftp 服务器程序的名为 in.ftpd。

您还需要给文件 inetd.conf 中 inetd 项加注释，使它不再管理 ftp 服务器。

### 12.2.1netd 服务器管理

如果您的系统中的某一服务平均只有很少的请求，就不需要此服务的服务器时刻运行。可以在远程用户访问它时再运行。inetd 守护进程管理 Internet 服务器，并只在系统收到对服务的请求时才调用它们。inetd 将不停地检查远程用户对某一特定服务的请求，当收到请求时便启动相应的服务器守护进程。例如，当 inetd 收到一个用户访问 ftp 的请求，它启动 ftpd，ftp 守护进程。ftpd 将处理此请求，允许用户下载文件。

要使得 inetd 调用合适的服务器守护进程，您必须配置此服务。要在文件 /etc/services 和/etc.inetd.conf 中放置服务器的项。/etc/services 文件中列出您

的系统中可用的服务。文件/etc/services 中的项包含服务的名字和它的端口名以及用斜杠分隔开的协议。出现在您的 Redhat 文件/etc/services 中 ftp 项如下所示。别的系统可能只要求一项。

ftp-data20/tcp
ftp 21/tcp

/etc/inetd.conf 文件是 inetd 配置文件。您需要为文件中的项指定服务、协议和调用的服务器程序。下面显示的是一个 ftp 的项。服务器路径和参数根据不同的 Linux 也有所不同。

#<service> <sock_type> <proto> <flags> <user> <server_path><args>
ftpstreamtcpnowaitroot/usr/sbin/ftpdftpd

在别的 Linux 中，配置行可能存在，但前面加有一个#号把它注释掉。您只须除去#号。如没有此配置项，就需要加上它。

tcpd

您可以用 tcpd 守护进程给 inetd 管理服务器加上一个安全级。您能设置 tcpd 来监视服务器通过 inetd 的连接。tcpd 将检查远端的用户认证以确认他们进行一个有效的请求。用 tcpd 还能限制远端用户访问您的系统。一列主机名存放在文件 hosts.allow 和 hosts.deny 中。文件中的项有如下格式：service:hostname:domain。domain 是可选项。对于 service 您能指定一个特定的服务象 ftp 或输入 ALL 指定所有的服务。对于 hostname 您能指定一个特定主机名，或用 ALL 指定所有的主机。下面的例子中，第一项允许所有的主机访问 Web 服务，http。第二项允许主机 pango1.train.com 访问所有的服务。第三和第四项允许主机 rose.berkeley.edu 和 caldera.com 进行 ftp 访问。

http:ALL

ALL:pango1.train.com
ftp:rose.berkeley.edu
ftp:caldera.com

文件 hosts.allow 中包含您允许访问的主机。如果您只是不允许一些特定的主机访问，您可以在文件 hosts.allow 中对服务指定 ALL，而在文件 hosts.deny 中列出您禁止的主机名。tcpd 的帮助页（man tcpd）提供关于 tcpd 更多的信息。

为了使 tcpd 监视一个服务器，您需要在文件 inetd.conf 中的服务器项的路径名区域中放置 tcpd 的路径名。您的 Caldera Network Desktop 已经设定了 ftpd 服务器项。用 tcpd 守护进程的路径名，/usr/sbin/tcpd 可以代替 ftpd 程序的路径名，/usr/sbin/in.ftpd。参数域中列出 in.ftpd 服务器程序。

#<service> <sock_type> <proto> <flags> <user> <server_path><args>
ftp stream tcp nowait root /usr/sbin/tcpdin.ftpd

当 inetd 收到一个 ftp 服务的请求，它调用 tcpd 守护进程，用它来接管和监视连接。接着它启动 in.ftpd 服务器程序。缺省情况下，tcpd 将允许所有的请求。为了允许所有的对 ftp 服务的请求，您需在您的文件/etc/hosts.allow 中输入如下所示的项，项 ALL:ALL 将使您的系统对所有的用户开放所有的服务。

ftp:ALL

## 12.3ftp 服务器

OpenLinux 已经安装一个 ftp 服务器并创建一个您放置供 ftp 访问的文件的

目录。目录已经被配置控制远程用户的访问，限制用户只能访问 ftp 目录和它的子目录。留给 ftp 的目录为/home/ftp。您要把允许 ftp 访问的文件放在目录/home/ftp/pub 中。还可以在其中创建子目录来放文件。一旦连接到网络中，一个远程的用户可以 ftp 到您的系统中，并从目录/hotm/ftp/pub 或它的子目录中下载文件。

虽然 OpenLinux 已经配置成一个 ftp 站点，但是理解它是如何配置的也是十分有用的。当您安装另一个系统时，您就需要知道下面所描述的如何安装 ftp 服务器和创建它的数据目录。然而在 OpenLinux 中已经完成这些设置，您不需要再一一设置。

在不同的 Linux 的 ftp 站点上都有 ftp 服务器软件。在 sunsite.unc.edu 和它的镜像站点上，ftp 服务器软件位于/pub/Linux/systems/Network/file-transfer 目录中。The Washington University 的 ftp 服务器是一个文件叫做 wu-ftpd-2.4-fixed.tar.gz。将来您可能想从这些站点中下载您的 ftp 服务器软件的更新版本并用它们来更新您的服务器。如果您从一个 ftp 站点下载 ftp 服务器软件，您将首先解压和展开它们。将会自动创建一些目录存放文档和源代码。服务器软件包中将有安装说明，可以用他们指导创建您的服务器目录和编译软件。

为了允许远程用户访问，必须运行服务器软件，ftpd（表 12-1 中有关于 ftpd 的选项）。和其他的服务器一样，您可以在启动机器是开始 ftp 服务，可以当收到一个请求时通过 inetd 启动 ftp 服务器，或直接从命令行中启动它。关于使用 inetd 启动 ftp 服务器，在前面“启动服务器”中已详细描述。为了在初启系统时便启动 ftp 服务器，您需要象在 Web 服务器中描述的那样，在目录/etc/rec./inin.d 中创建一个 init 脚本。要想从命令行中启动 ftp 服务器，您要输

入 ftpd 命令和参数或选项。ftpd 服务器能通过选项被调用。通常要带允许注册的-l 选项。选项-t 和-T 用来给用户设定中断时间，一段时间内没有动作，就将切断那些连接。选项-d 显示调试信息，-u 为上传的文件设置 umask 值。安装在您的 Caldera Desktop 中的 ftp 服务器的名字是 in.ftpd，用它来直接调用 ftp 服务器。

表 12-1Web 服务器目录和 http 选项

| 选项 | 作用 |
| --- | --- |
| -d | 为系统日志写调试信息 |
| -l | 在系统日志中注册每个会话 |
| -t seconds | 设置不活动的超时秒数（缺省的是 15 分钟） |
| -T seconds | 允许用户设定的最大的超时时间（缺省的是两小时） |
| -a | 使用 ftpaccess（5）配置文件 |
| -A | 禁用 ftpaccess（5）配置文件 |
| -L | 把设置到 ftpd 服务器的命令注册到系统日志中 |
| -l | 把 ftpd 收到的文件注册到 xferlog |
| -o | 把 ftpd 传输的文件注册到系统日志 |

## 12.4Ftp 服务器配置文件

这有一组配置文件您可用它们来管理您的 ftp 站点。在您的 OpenLinux 系

统中，这些文件位于目录/etc 下，并以 ftp 作为名字的开头。ftpaccess 文件将决定访问您的 ftp 站点的用户的权能。访问、信息、权限、记录和其余的杂项功能将被指定。例如，您可给某一目录创建一个别名，当一个 ftp 用户注册时显示一个信息或不允许任何匿名用户删除文件。文件中的 loginfails 项定义一个用户在断出之前试着登录的次数，email 项指定 ftp 管理员的 email 地址。ftpaccess 的帮助页中列出可能的选项。当安装您的 OpenLinux 系统，ftp 服务器将被自动安装和配置。下一部分显示的是 ftpaccess 文件和它的配置。

权能是为三种不同类型的用户设置的，三种用户为：匿名、客户和真正的用户。匿名用户是指用匿名注册的用户。客户用户是指用客户帐号注册或访问的用户。一个真正的用户是指一个在系统中有帐号的用户并用 ftp 连接访问系统。用类选项可定义您自己的类。在这里显示的 ftpaccess 文件中，创建一个叫做 all 的类，类中包括所有的匿名、客户和真正类型的用户。

当显示一个信息时，信息项指定一个带有此信息的文件。当用户注册时显示一个信息，当进入某一个目录时显示另一个信息。例如，下面的项是用户注册时显示在文件/welcome.msg 中的信息。

```
message  /welcome.msglogin
```

为了设定权限，您用命令加上一个 yes 或 no，接着列出用户类型或类。在此显示的文件 ftpaccess，所有的用户能执行 tar 和 compress 操作，但是匿名和客户用户将被禁止使用 chmod、delete、overwrite 和 rename 操作。他们不能删除文件、改变它们、更改它们的名字或权限。

```
/etc/ftpaccess
class all real,guest,anonymous *
email root@localhost
```

```
loginfails 5
readme README* login
readme README* cwd=*
message /welcome.msglogin
message.messagecwd=*
compress yesall
taryesall
chmodnoguest,anonymous
deletenoguest,anonymous
overwritenoguest,anonymous
renamenoguest,anonymous
log transfers anonymous,real inbound,outbound
shutdown/etc/shutmsg
passwd-check rfc822 warn
```

您能用 ftphosts 文件来允许或禁止别的主机访问您的 ftp 站点。ftpusers 文件列出不能通过 ftp 访问的用户。例如，即使您知道口令，也不能通过 ftp 连接注册成为一个超级用户。ftpgroups 是一个组存取文件，它允许 ftp 用户成为您系统中一个特定组的成员。此文件中列出特定的组口令。在 ftpaccess 文件中 private 项必须被设置成 yes。表 12-2 中列出可能的 ftpconversions，象压缩和存档的选项。关于 ftpaccess 项，参见表 12-3。

表 12-2ftp 配置文件

| | |
|---|---|
| ftpaccess | ftp 用户的访问、信息、权限、记录和其余的杂项功能 |
| ftpconversions | 为压缩和归档操作的 Ftp 转换表 |

| | |
|---|---|
| ftpusers | 不允许通过 ftp 访问的用户列表，象 root |
| groupsftp | ftp 用户用特定的口令能访问的组。格式是 access-group:encrypted:real-group<br>access-group 是一个任意的字符串。real-group 是列在 /etc/group 中的一个有效的组名 |
| ftphosts | 允许或禁止不同的主机访问您的 Ftp 站点<br>allow username addrglob [addrglob...]<br>只允许匹配 addrglob 的主机名以 username 注册<br>deny username addrglob [addrglob...]<br>将拒绝匹配 addrglob 的主机名以 username 注册 |

表 12-3Ftpaccess 项（以“glob”结尾的项目可以使用文件名匹配符；*、？和[]）；类型列表是用逗号分隔的一列包含关键字：“anonymous”、“guset”和“real”的列表

| 访问能力 | | 描述 |
|---|---|---|
| autogroup | group classglob [classglob...] | 允许通过一个匿名用户的特定类别访问一个组的只读的文件和目录 |
| class | class typelist addrglob [addrglob...] | 定义用户的类和表 addrglob 的原地址。typelist 是用逗号分隔的一列包含关键字：“anonymous”、“guset”和“real”的列表 |
| deny | host-addrglob | 总是拒绝名字和和 host-addglob 匹配的 |

| | |
|---|---|
| message_file | 主机的访问，并显示 message-file 文件 |
| guestgroup groupname [groupname...] | 当用户是一个特定组的成员时，允许客户通过一个真正的用户进行访问。客户用户的口令项将指定一个在 ftp 目录中的一个用户主目录 |
| limit class n times message_file | 在时间 times，限制类别为 class 的用户数不超过 n，并显示 message_file 文件 |
| noretrieve file-list | 不检索这些文件 |
| loginfails number | 在 number 次注册失败后，中断 ftp 连接。缺省值为 5。 |
| private yes\|no | 用户成为指定在组访问文件 ftpgroups 中的一个组的成员信息功能 |
| banner file | 注册之前显示标题。文件要求全路径名 |
| email email-address | 定义 ftp 管理员的 email 地址 |
| message file{when{class...}} | 在 ftp 注册时或切换目录时显示文件的内容。“ when ”参数可以是“ LOGIN ”或“ CWD=dir ”；dir 指定当进入该目录时显示信息。在信息文件中可能是“ magic cookies”，此文件将引起 ftp 服务器替换“ cookie ”并带 d 上一些特定的字符串，象日期或用户名 |
| raadme file{when{class}} | 当用户注册或使用切换目录命令时，提示 |

| | |
|---|---|
| | 文件 file 存在和此文件的修改日期记录功能 |
| log commands typelist | 能够记录用户使用的命令 |
| log transfers typelist directions | 记录文件的传输。directions 是逗号分隔的一列项，包括“ inbound ”和“ outbound ”，它们将记录文件上传到服务器或从服务器下载 |
| **杂项功能** | |
| alias string dir | 为一个目录定义一个别名字符串 |
| cdpath dir | 载 cdpath 中定义一项，它定义一个在切换目录时用到的搜索路径 |
| compress yes\|no classglob [classglobtar yes\|no classglob [ classglob...] | 使能够 compress 或 tar 任何匹配 classglob 的类。实际的转换定义在外部文件 ftco nversion 中 |
| shutdown path | 如果 path 指定的文件存在，服务器将检查此文件已决定是否关闭服务器 |
| virtual address root\|banner\|logfile path | 打开虚拟服务器功能权限功能允许或不允许执行特殊功能的能力。缺省情况下所有的用户都允许 |
| chmod yes\|no typelist | 允许或不允许改变用户权限 |
| delete yes\|no typelist | 允许或不允许删除文件，rm |
| overwrite yes\|no typelist | 允许或不允许更改文件 |

| | |
|---|---|
| rename yes\|no typelist | 允许或不允许给文件重命名，mv |
| umask yes\|no typelist | 允许或不允许文件创建权限 |
| passwd-check none\|trivial\|rfc822(enforce\|warn) | 定义服务器对匿名 ftp 用户进行口令检查的级别是否强制执行 |
| path-filter typelist mesg allowed_charset{disallowed regexp...} | 对于在 typelist 中的用户，path-filter 定义常规表达式来控制文件名的格式。这里可能有多个不允许的 regexp |
| upload root-dir dirglob yes\|no ownergroup mode ["dirs"\|"nodirs"] | 定义一个带有 dirglob 的目录允许或禁止上传 |

### 12.4.1ftp 用户帐号

为了允许别的用户用匿名 ftp 访问您的系统，您必须创建一个名为 ftp 的帐号。您能给帐号 ftp 设置一些限制，使得任何远程的 ftp 用户不能访问您系统的其他部分。您必须改变此帐号在文件/etc/passwd 中的项，使一般的用户不能访问它。在您的 Caldera Network Desktop 已经设置允许匿名用户进行 ftp 注册，下面的项在您的/etc/passwd 文件中。

ftp:*:14:50:FTP User:/home/ftp:

在口令区域中的星号用来保护此帐号，它将阻止其他的用户以此帐号注册，并因此控制它的文件或访问您的系统的其他部分。用户 ID，14，是一个独特的 ID。注释域是“FTP User”。注册目录是/home/ftp。当 ftp 用户注册到您的系统时，他将处于此系统中。如果没有设置主目录，需创建一个，并用命令 chown

为 ftp 用户改变它的权限。

组 ID 是 ftp 组的 ID，是专门为匿名 ftp 用户设置的。您能通过为 ftp 组设置限制来限制匿名的 ftp 用户。下面是一个您能在/etc/group 文件中找到的关于 ftp 组的项。对于别的 Linux，如果您没有此项，应该加上它。

ftp::50

目录/home/ftp 的权限中应该否定写权限。如果您不希望 ftp 用户创建和删除目录，您可以用 chmod 命令设置权限 555 来禁止写访问：chmod 555 /home/ftp。

目录/home/ftp 和它的权限、口令和组项已经设置在您的 Caldera Network Desktop。如果您在另外的 Linux 上设置一个 ftp 服务器，要自己设置/home/ftp。

### 12.4.2ftp 服务器目录

为了防止您的系统遭到 ftp 用户的一些意外的访问，您应在 ftp 目录中，如/home/ftp 中，创建一组有限制的目录。在表 12-4 中提供一列目录。保护的一个重要部分是阻止远程用户使用不再限制目录中的命令或程序。例如，因为 ls 命令位于您的/bin 目录中，您可能不希望用户使用 ls 列出文件名。同时，您又希望 ftp 用户使用 ls 命令。为了做到这一点，您需要在目录/home/ftp 中创建一个新的目录 bin，接着复制一份命令 ls 放到/home/ftp/bin 中。此目录将限制 ftp 用户的使用，他们使用的命令 ls 是目录/home/ftp/bin 中的命令，而不是您自己用的/bin 中的 ls 命令。通过同样的方法，您可以让 ftp 用户使用其他命令。您需要的其他命令是 cd，它允许用户切换目录，命令 more，它让用户显示文本文件。您的 Caldera Network Desktop 已经创建了/home/ftp/bin 目录并在其中

安装一些基本命令。您能加入一些希望使用的命令。

表 12-4ftp 目录

| | |
|---|---|
| /home/ftp | ftp 服务器目录，所有者是 root（在一些系统中，此目录可能是/usr/local/ftp）；所有的目录和子目录有权限 555 或 755 以限制其它用户或组只有读或执行的权限 |
| /home/ftp/bin | 此目录中含有 ftp 远程用户能执行的命令，象 ls 和 cd |
| /home/ftp/etc | 此目录中含有配置文件象它自己的 passwd 文件 |
| /home/ftp/pub | 您提供的可下载的文件放置在此目录中；您可以在其中设置子目录 |
| /home/ftp/lib | 如果 ls 命令需要 ld.so.1 文件，就需要此目录来存放此文件 |

您还需要一个/home/ftp/etc 目录，存放您的 passwd 和 group 文件的副本。另外它也阻止 ftp 用户访问/etc 目录下的原文件。编辑/home/ftp/etc/passwd 文件，删除您的系统的一般用户的项。剩余的项的口令应被设置为*，以保护访问。对于 group 文件，除去所有的用户组并设置所有的口令为*。

```
#cat  /home/ftp/etc/passwd
root:*:0:0:::
bin:*:1:1:::
operator:*:11:0:::
ftp:*:14:50:::
nobody:*:99:99:::
```

```
#cat  /homt/ftp/etc/group
root::0:
bin::1:
daemon::2:
sys::3:
adm::4:
ftp::50:
```

目录/home/ftp/pub 中放有您想让远程 ftp 用户下载的文件。当 ftp 用户注册到系统时，它将处于目录/home/ftp 中，并能切换到目录/home/ftp/pub 中开始访问其中的文件。在/home/ftp/pub 中能加入任何您希望的目录和文件。您甚至可以指定一些目录为上传目录，允许 ftp 用户上传文件到您的系统中。

一些 Linux 系统要求，ls 命令工作时要访问 libc.so.1 和 rld 文件。它们通常存放在您的/lib 目录中。因为您不希望 ftp 用户间接访问您的系统，您要创建一个/home/ftp/lib 目录，并复制这些文件，把复制的文件放在此目录中。另外，因为 rld 使用/dev/zero 文件，您还要创建一个/home/ftp/dev 目录并用 mknod 复制设备文件/dev/zero，然后把它放到此目录中。

### 12.4.3 权限

为了限制 ftp 用户只能访问目录/homt/ftp 和它的子目录，您需要对 ftp 用户隐藏文件结构的其余部分。您要让目录/homt/ftp 呈现为 ftp 用户的主目录。实际的主目录/，和其他的目录结构则对 ftp 用户隐藏。您可以用命令 chroot 加上参数 ftp，使得目录/home/ftp 呈现为主目录。

#chroot ftp

现在，当 ftp 用户用 cd / 命令来切换目录时，他总是切换到目录/home/ftp 中。

作为进一步的限制，在/home/ftp 中所有的具有命令的目录和命令本身都是为 root 所有，而不是被 ftp 用户所有。简而言之，ftp 用户不能控制这些目录。您能用命令 chown 来改变目录的所有权。下面的例子便是把目录/home/ftp/bin 和命令/home/ftp/bin/ls 的所有权设置成 root。root 应该拥有/homt/ftp/bin，/homt/ftp/etc 以及它们包含的所有文件。Caldera Network Desktop 已经为您作到这点。对于别的系统，您可能要自己设置所有权。

#chown root  /home/ftp/bin

ftp 目录的权限应该设置为允许 ftp 用户访问。对于所有者、组和另外的用户，有三组权限：读、写和执行。为了允许 ftp 用户访问，组和目录的其他权限应设置为可读和执行。执行权限允许 ftp 用户访问目录，读权限则允许列出目录中的内容。目录不允许 ftp 用户具有写权限。没人想让 ftp 用户能删除或添加一个目录。例如，目录/homt/ftp/bin 需要读和执行权限，因为 ftp 用户要访问和执行它的命令。对于拥有可以下载的文件的目录/homt/ftp/pub 来说，它必须拥有读和执行的权限。

作为目录的所有者，您需要写权限以便能添加新文件或子目录。当然，只有当您做改变时才需要写权限。为了进一步的安全，当您不需要做改动时，能设置这些目录对所有的用户包括所有者都只开放读和执行的权限。用命令 chmod 加上数字 555 和目录名将设置对所有的用户为读和执行权限。当前在您的 Caldera Network Desktop 中，ftp 目录的权限设置的为 755，它给了所有者写

权限。

#chmod 555 /home/ftp/bin

对于目录/homt/ftp/bin 中文件的权限和其他指定的 ftp 目录的权限能有更多的限制。一些文件需要执行，而另一些文件只要被读。目录/home/ftp/bin 或/home/ftp/lib 中的文件 ls 和 rld 需要执行，可以设置权限为 555。在目录/home/ftp/etc 中的文件象 passwd 和 group 可以设置权限为 111，即只读的权限。如下面的例子显示的，能用命令 chmod 来设置文件的权限。在您的 Caldera Network Desktop 系统中已经为您设置了这些权限。对于别的系统，您可能要自己设置。

#chmod 111 /home/ftp/etc/passwd

### 12.4.4ftp 文件

在目录/homt/ftp/pub 下的目录中有 ftp 文件，作为礼貌，您应为 ftp 用户创建一个 readme 文件和一个 index 文件。readme 文件应包含对此目录中的文件的一个简单描述。index 文件应包含一列文件和对文件中的内容进行简单描述。

## 12.5Web 服务器

在安装 OpenLinux 期间，将自动地在您的系统中安装 Apache Web 服务器和所有的必须的目录及配置文件。您的 OpenLinux 系统已经是一个全功能的 Web 站点。每次当您启动系统，Web 服务器也同时启动，并不断地运行。您的

Web 站点的数据文件的目录为/home/httpd/html。把您的 Web 页放在此目录或它的子目录下。在 Apache 的站点 http://www.apache.org 中提供了它的 Web 服务器在线的支持和手册。在 Caldere 的站点 http://www.caldera.com 中也有在线的支持。您能得到关于开发您的 Web 站点和您遇到的问题的更多的信息。

您不需要做更多的事情。一旦连接到一个网络中，远程的用户将能访问您的 Web 站点。但是，了解一个 Web 站点是如何建立的将是十分有好处的。如果您使用另外的系统，可能需要知道安装 Web 服务器和创建目录的过程。下面将对此过程进行一个描述。

安装在您的 OpenLinux 系统中的 Web 服务器把 Web 站点设置在目录/homt/httpd 中。为了管理此站点，还要设置一些目录，如表 12-5 所示。cgi-bin 有网关接口，文件 icons 拥有您的主页所用的图标。您的 Web 页被放在目录/home/http/html 中。把您的主页的 index.html 文件也放在此目录下。配置文件在不同的目录，/etc/httpd/conf 中。

表 12-5Web 服务器目录和 httpd 选项

| Web 服务器目录 | 描述 |
|---|---|
| /home/httpd | 您的 Web 服务器的目录 |
| /home/httpd/cgi-bin | 公共网关接口和脚本 |
| /home/httpd/icons | 主页的图标 |
| /home/httpd/html | 您的 Web 站点的 Web 页 |
| /etc/httpd/variant/conf | 存放您的 Web 服务器的配置文件，variant 是一个特定的 Web 服务器；例如， |

| | |
|---|---|
| | /etc/httpd/apache/conf 中的配置文件是安装在 OpenLinux 上的 Apache Web 服务器的 |
| httpd 选项 | |
| -d | 如果不用缺省的目录，用此选项指定一个目录 |
| -f | 允许您指定一个不同于 httpd.conf 的配置文件 |
| -v | 显示版本 |

还有一些免费的 Web 服务器软件可供您使用。NCSA httpd Web 服务器是最早开发的服务器之一。Apache 和它非常相似并改正了 NCSA httpd 服务器软件中的一些问题。您能从多数的 Linux ftp 站点上下载服务器软件。在 Linux ftp 站点和它的镜像站点上，Web 服务器软件位于目录 /pub/Linux/systems/Networks/info-systems/www/server 中。您能用 ftp 或 Netscape 来访问此目录。将来能从此目录中下载您的 Web 服务器的更新版本并升级您的服务器。

如果从一个 ftp 站点上下载一个 Web 服务器软件，您需要解压文件并提取档案文件。此过程中将用到许多以前的目录，不过在目录中新增一些文档和源代码。服务器软件包中将包括关于您的服务器目录的创建说明和编译您的软件的说明。

### 12.5.1 配置您的 Web 服务器

用目录/etc/httpd/conf 目录中的一些配置文件能配置您的 Web 服务器软件。根据需要不同配置也有所不同，可以让您的 Web 服务器作为一个守护进程不间断地运行，或当 inetd 需要时调用它。如果您的服务器将有大量的使用，应该让它作为一个守护进程直接运行。

httpd.conf 文件用来配置 Web 服务器守护进程。它列出了一列变量和它们的值。每一项包含一个单独的变量名和一个值，中间用一个空格分开。这些变量为您的 Web 服务器设置不同的特征。一些变量需要路径名，另外的一些只用设置关键字“on”或“off”。在您的 Caldera Network Desktop 中已经设置了这些变量。您能增加一个或改变一项。表 12-6 提供您放在 httpd.conf 文件中的不同变量的一个完整列表。下面是一个例子，用变量 ServerAdmin 设定用户发送关于管理的问题的邮件地址。用您想接收关于系统管理的邮件的地址来代替项 you@ your.address。

#ServerAdmin: Your address, where problems should be e-mailed.
ServerAdmin you@ your.address

在此文件中只设置一些变量，您不需要每个都进行设置。一些需要您的系统的特殊信息。例如，ServerName 为您的 Web 服务器指定一个另外的主机名。它必须是有效的主机名。假设您的系统的主机名是 richlp.ix.com。还有一个另外的主机名 www.ix.com。注意此项前有一个#号。去掉#号，并输入您的 Web 服务器的主机名以取代 new.host.name。

#ServerName allows you to set a hostname which is sent back to clinets for

#your server if it's different than the one the program would get(i.e. use
#"www" instead of the host's real name).
#ServerName new.host.name

文件 srm.conf 配置 Web 服务器的资源。它列出一组变量和它们的值。项的前面通常带有一个注释来解释此项。一些项已经被赋值，另外的一些要您自己输入值。您能用标准的文本编辑器来改变项。表 12-7 中提供一列这些变量。下面的例子显示项 DocumentRoot。此项前有一个注释解释。项本身包含一个变量名和它的值，在此值为一个目录的路径名。

表 12-6httpd.conf 的变量

| 变量 | 描述 |
| --- | --- |
| AccessConfig | 文件 access.conf 的位置(缺省的是/conf/access.conf) |
| AgentLog | 动作的记录文件（缺省的是 logs/agent_log） |
| ErrorLog | 记录错误的文件的位置；如果不是以/开头，则指相对于服务器主目录的地址（缺省的是 log/error_log） |
| Group | 服务器作为一个守护进程运行的组 ID |
| IdentityCheck | 对远程用户进行识别检查 |
| MaxClients | 限定服务器运行的最大总数，例如，限定能同时连接的客户数；超过此数，客户将不能注册（缺省值为 150） |
| MaxRequestsPerChild | （缺省数是 30） |

| | |
|---|---|
| PidFile | 服务器将记录它的 pid 到此文件（缺省的是 /logs/httpd.pid） |
| Port | 等待请求的端口 |
| ResourceConfig | 文件 srm.conf 的位置（缺省是 conf/srm.conf） |
| ServerAdmin | 管理员的 e-mail 地址 |
| ServerName | 另外的服务器的主机名 |
| ServerRootWeb | 服务器的用户的主目录；也是服务器的配置、错误和记录文件所放置的目录（缺省的是 /usr/locla/etc/httpd） |
| ServerType | 或是单机或是 inetd |
| StartServers | 启动的服务器数（缺省的是 5） |
| TimeOut | 等待用户请求的秒数；如果在此时间内没有收到请求，用户将退出注册(缺省的是 400) |
| TransferLog | 记录的路径（缺省的是 log/access_log） |
| TypesConfigMIME | 配置文件的位置（缺省的是 conf/mime.conf） |
| User | 服务器的用户 ID |

表 12-7srm.conf 的变量

| 变量 | 描述 |
| --- | --- |
| AccessFileName | 在每个目录中拥有访问控制信息的文件 |
| AddDescription | 放置在服务器产生的索引中的文件的简短描述 |
| AddEncoding extensions-list | 当浏览器查到压缩文件时，对它们进行解压；extensions 列表中包括不同的扩展名，用来指定压缩的类型，象.gz 指定 gzip。AddEncoding x-gzip gz |
| AddIcon image file file - extensions | 件和一列文件扩展名，象.mpg， .bin 或.ps AddIcon /icons/movie. gif .mpg .qt |
| AddIconbyEncoding | 指定图标并加一些解码的信息 |
| AddIconByType | 用 MIME 类型来确定图标的使用 |
| AddLanguage language Extension | 允许您指定一个文档的语言；如果一个文件使用浏览器能识别的语言，您可以使用浏览器的内容可变的功能，指定语言和扩展名，使浏览器能显示此文件 AddLanguage en .en |
| AddType type/subtypeextension | 允许您不用编辑即可覆盖 MIME 类型，并指定特定的文件为某一种类型；在 Apache |

| | |
|---|---|
| | 的文件 srm.conf-dist 中有一些项能使用象 map 文件，这些项被注释掉 |
| Alias alias-namepath-name | 为不同的路径名创建别名 Alias /icons/ /usr/local/etc/httpd/icons/ |
| DefaultType | 文档使用的缺省的 MIME 类型，服务器不能从文件的扩展名中找到的此类型（缺省是 text/html） |
| DefaultIcon | 不包含图标的文件所显示的图标 |
| DirectoryIndex | 您的 Web 站点索引的文件名；这些文件被做为 HTML 目录的预写索引；多个项之间用空格分隔（缺省是 index.html） |
| DocumentRoot | 您提供的文档的目录；缺省下，所有的请求都从此目录中获取文档，但是符号连接和别名可能指向其他位置（缺省是 /usr/local/etc/httpd/htdocs |
| FancyIndexing | 为了做索引，加文件名和图标到文件列中；可以设置为 on 或 off |
| HeaderName | 做为目录索引扩展的文件（缺省是 HEADER） |
| IndexIgnore file-list | 被目录索引忽视的文件集；包括一些文件，象 README 或 HEADER |
| IndexOptions | 索引选项 |

| | |
|---|---|
| LanguageProiority Language-list | 允许您给定一些语言的一个优先顺序；忽略内容具有多义性 LanguagePriority en fr de |
| OldScriptAlias | 同 Alias |
| Readmename | 服务器缺省情况下查找的 README 文件的名字（缺省是 README） |
| Redirect | 告诉用户在那能找到不在您的服务器上的文件 |
| ScriptAlias alias-name path-name | 控制包含服务器脚本的目录<br>ScriptAlias /cgi-bin/ /usr/local/etc/httpd/cgi-bin/ |
| UserDir | 如果收到一个用户请求，附加到用户主目录中的目录名(缺省是/public_html) |

DocumentRoot /usr/local/etc/httpd/htdocs

文件 access.conf 决定一些服务是否允许用户访问，这些服务位于服务器的某一目录中。此文件中包括一系列的指令，每个指令都包括在一组目录标识符中。开始的标识符是单词“ Directory”加上一个目录路径名，并用尖括号括起来，<Directory pathname>。结束的标识符用相同的<>符号，只是在“ Directory”前加有一个斜杠，</Directory>。在标识符中能加入一些指令。

在指令 Options 中，您能授权一些特征，象使用符号连接。用指令 AllowOverride，您能决定文件.htaccess 能覆盖那些被指令设定的特征。用 Limit 指令，您能改变 access.conf 文件以控制 Web 用户访问您的系统。Limit 指令用了一组标识符象 Directory 标识符。用<Limit>开始一个 Limit 指令并用</Limit>

结束它。Limit 指令指明谁能访问您的 Web 服务器。还有另外的一些可用选项。例如，allow 选项加上一列主机名，将限制只能这些主机访问。deny 选项加上一列主机名将拒绝这些系统的访问。表 12-8 列出了不同的指令和它们的选项。

在您的文件 access.conf 中，将找到两个目录项。第一个目录项意味着把此目录作为您的 Web 站点的主目录。您的 Web 页也位于此目录中。确认在第一个目录项中使用的路径名是否就是您的 Web 站点使用的根路径名。如果不是，改正它。下面是一个 Directory 项的例子。

```
#This should be changed to whatever you set DocumentRoot to.
<Directory /home/httpd/htdocs>
Options Indexes FollowSymLinks
AllowOverride All
# Controls who can get stuff from this server.
<Limit GET>
order allow, deny
allow from all
</Limit>
</Directory>
#Place any directories you wnat access information for after this one.
```

表 12-8access.conf 的指令

| 目录标识符 | 描述 |
| --- | --- |
| <Directory path-name></Directony> | 指定设置控制的目录；以</Directory>结束指令 |

| | |
|---|---|
| Options feature-list | 指定目录的服务器选项；放置在目录和限制指令中<br>All 使用所有的特征<br>ExecCGI 使 CGI 脚本可执行<br>FollowSymLinks 使用符号连接<br>Includes 允许使用 include 文件<br>IncludeNoExec 允许 include 文件但不允许 exec 操作<br>Indexes 允许用户查询索引<br>None 禁用所有特征<br>SymLinksIfOwnerMatch 使用符号连接前检查用户的 ID |
| AllowOverridefeature-list | 目录中.htaccess 文件能覆盖的控制选项<br>All 不限制访问<br>AuthConfig<br>AuthName 目录的授权名<br>AuthType 目录的授权类型<br>AuthUserFile 包含用户名和口令的文件 |

| | |
|---|---|
| <Limit></Limit> | AuthGroupFile 包含容许的组名的文件<br>FileInfo 使用 AddType 和 AddEncoding 指令<br>Limit 使用限制指令<br>Options 使用 Options 指令<br>用下列指定控制对您的 Web 服务器的访问；项 all 指定所有的主机<br>allow host-list -host-list 允许在 host-list 中指定的主机访问<br>denies host-list host-list 拒绝在 host-list 中指定的主机访问<br>orders options 对 deny 和 allow 列表进行排序 order deny,allow<br>requires host-list host-list 请求使用用文件 AuthUserFile 进行认证 |

### 12.5.2 启动 Web 服务器

您能用命令 httpd 手工启动您的 Web 服务器。此命令有一些选项。选项-d 允许您为 http 程序指定一个不同于缺省的目录。用选项-f，您能指定一个不是

httpd.conf的配置文件。选项-v显示版本号。

在前面描述init脚本时提到,您的OpenLinux系统启动时将自动地启动Web服务器守护进程。Web服务器的启动脚本叫做httpd，它位于目录/etc/rc.d/init.d中。一个叫做S85httpd的符号连接并运行目录/etc/rc.d/rc3.d中的rc程序。对于别的Linux系统，您能把Web服务器的命令放在系统启动脚本象rc.locla或rc.sysinit中。

如果您希望inetd守护进程调用httpd,在文件/etc/services和/etc/inetd.conf中放置一个关于httpd的项。/etc/services中列出可用于您的系统中的不同的服务。对于Web服务器，您可以输入http和指定一个端口/tcp。

http80/tcp

在文件/etc/inetd.conf中的Web服务器项和ftp项很相似。安装在您的系统Caldera Network Desktop中的Web服务器的路径名是/usr/sbin/httpd.httpd，它没有参数。

http stream tcp nowait nobody /usr/sbin/httpd httpd

为了使tcpd能监视和控制Web服务器的请求，需要用路径名/usr/sbin/tcpd代替路径名/usr/sbin/httpd。

http stream tcp nowait nobody /usr/sbin/tcpd httpd

还要为您的httpd.config-dist文件中的ServerType变量指定值inetd。

#ServerType is either inetd, or standalone.

ServerType inetd

OpenLinux没有配置从inetd运行您的Web服务器。如果您希望这样，需要从自动启动的列表的启动守护进程列中除去您的Web服务器，并在文件/etc/services和/etc/inetd.conf中加入合适的项。为了配置自动启动列表,从Lisa

主菜单中选择“ System Configuration ”，从下一级菜单中选择“ System Configuration ”。接着选择“ Configure daemon/server autostart ”项。显示一列守护进程和服务器。项后带有一个`x`，表示在启动您的系统时将启动的项。移到 Web 服务器的项，按 SPACEBAR 删除它和`x`。接着按 ENTER（参见第 19 章关于 Startup Daemon 菜单的讨论）。

如果安装另一个 Web 服务器，您要确保配置文件被正确设置和安装。如果您用另一个版本，并且自己安装 Apache，您将注意到会出现带有扩展名.conf-dist 的配置文件。您需要把这些文件复制到有相同前缀，后缀为.conf 的文件中。Web 服务器将只从带有后缀.conf 的文件中读取配置信息。然后可以进一步创建您的配置项。

为了检查您的 Web 服务器，启动 Web 浏览器并输入您的系统的 Internet 域名。对于系统 turtle.trek.com，用户输入 http://turtle.trek.com。将显示放在 Web 服务器上主目录中的主页。最简单的方法是用 lynx，命令行浏览器。启动 lynx 并按 g 以打开一行，在此能输入您自己系统的 URL。lynx 将显示您的 Web 站点的主页。不过首先要确认把文件 index.html 放在/home/httpd/html 目录中。

您也能用 telnet 来检查您的 Web 服务器的操作。用 telnet、您的系统主机名和您的 Web 服务器操作使用的端口

```
telnet turtle.trek.com 80
```

# 12.6Gopher 服务器

Gopher 服务器用一种高度组织的方法来访问 Internet 资源，诸如数据文件或图形。不同于 ftp，用 Gopher 能呈现给用户一个菜单以供它们选择。每项菜单能引出另一个菜单或另一个 Gopher 站点。从这个方面讲，Gopher 象 Web 那样，允许您从一个站点移到另一个站点以查找资源，但是它又象 ftp 那样只列出资源，没有文本或图形来进行解释。

Gopher 用一个 TCP/IP 协议叫做 Gopher 协议。它能使得 Gopher 菜单文件进行高速地传输。Gopher 信息包含在 Gopher 文件中，这些文件包含一列项，这些项在某些站点上是可访问的。每一项被组织成五个区域来指定区域中的信息和这些项的地址。每个区域之间用一个 tab 分开：类型、显示名字、选择器字符串、主机名和端口。

类型能是一些可能的 Gopher 代码中的一个，正如表 12-9 中所列出的。显示名是对显示在 Gopher 菜单中的项目的一个描述。选择器字符串是项目的独特的标识符。主机名是项目所处的系统的主机名，端口是访问此主机系统时所用的端口（通常是 70）。

表 12-9Gopher 文件类型

| 文件类型 | 描述 |
|---|---|
| 0 | 文本文件 |
| 1 | Gopher 目录 |
| 2 | CSO 电话号码服务器 |

| | |
|---|---|
| 3 | 错误 |
| 4 | BinHex Machintosh 文件，HQX |
| 5 | 二进制的 DOS 文件 |
| 6 | Unix UUencoded 文件 |
| 7 | 全文本索引（Gopher 菜单文件） |
| 8 | Telnet 连接，包括远程主机的地址 |
| 9 | 二进制文件 |
| g | GIF 图象文件 |
| h | HTML 文件 |
| I | Graphic 图象文件（不是 GIF） |
| M | MIME 多部分混合信息 |
| P | Adobe PDF 文件 |
| S | 声音文件 |
| T | TN3270 telnet 连接 |

Gopher 是由 University of Minnesota 开发的，而且还在不断地开发新的版本。您能从 University of Minnesota 的 Gopher ftp 站点 boombox.micro.unm.edu 中目录/pub/gopher/unix 下取得一个 Gopher。您也能从大多数的 Linux ftp 站点取得 Gopher 服务器软件。例如，在站点 sunsite.unc.edu 和它的镜像站点的 /pub/Linux/systems/Network/info-systems/gopher 目录或 /pub/packages/info-systems/gopher/boombox-mirror/unix 目录中能取得此软件。还有一个 GNU 免费的 Gopher 服务器软件，叫做 GN Gopher。它就是您的 Caldera CD-ROM 中的名为 gn-2.22-1.i386.rpm 的软件包。用 rpm 或 glint 安装和配置您的 GN Gopher

服务器。University of Minnesota 为任何教育机构或非商业应用提供免费的 Gopher 软件。但是对于商业用户需要一个授权的费用或任何人要通过 Gopher 服务器访问信息需要付费。GN Gopher 服务器软件对任何人，商业或非商业的均免费。University of Minnesota 也有一个更高的版本叫做 Gopher+，它当前是一个商业产品。

安装 GN 和 University of Minnesota 的 Gopher 服务器有所不同。本章中的例子用的是 University of Minnesota 的 Gopher 的 2.3 版和 2.20 版的 GN Gopher，它们能从大多数的 Linux ftp 站点上取得。您还可能得到更新的版本。University of Minnesota 的 Gopher 软件包中包括 Gopher 客户程序和服务器软件。GN Gopher 软件包只包括服务器软件。

gopher 客户程序是一种速度很快的行模式的客户程序。另外还有一些别的客户程序可以使用，象 xgopher，它在您的 Caldera CD-ROM 中。

### 12.6.1Gopher 的用户帐号和数据目录

您需要一个数据目录来存放 Gopher 数据文件，还要有能进行 Gopher 访问的用户帐号和组。虽然 Caldera Network Desktop 已经创建了 Gopher 用户，但它没有创建数据目录。如果您使用另外的版本，需要自己创建 Gopher 用户。

#### Gopher 的用户帐号

为了能更好地控制其他 Gopher 用户访问您的系统，您应该有一个名为 gopher 的用户帐号。您的 Caldera Network Desktop 已经创建了 Gopher 用户帐号并对它进行了配置。在您的 Caldera Network Desktop 的文件 /etc/passwd

中您将能发现下面的关于 Gopher 的项。

gopher:*:13:30:gopher:/home/gopher

口令域中的星号用来保护此帐号，以防止其他的用户通过此帐号注册，并因此控制它的文件或访问您的系统的其他部分。用户 ID，13，是一个独特的 ID。注释域为“ gopher”。注册目录为/home/gopher。当 Gopher 用户注册到您的系统，他将进入此目录。

对于别的系统，您要自己创建 Gopher 帐号。您能给此帐号加上一些限制以阻止远程的 Gopher 用户访问您系统中的其他部分。您能改变/etc/passwd 文件中关于此帐号的项，通过设置口令域为一个星号来阻止其他一般的用户访问它。组 ID 是指 gopher 组的 ID，它专为 gopher 用户设置。您能在 gopher 组上设置一些限制用来限制所有的 Gopher 用户。下面是一个在文件/etc/group 中的关于 gopher 组的项。对于其他的 Linux 系统中如果没有此项，可以加上它。

gopher::30:

## Gopher 数据目录

Caldera Network Desktip 为它的 gn Gopher 服务器指定/home/gopher 目录作为 Gopher 数据目录。您的 Gopher 文件的数据目录应该和 Gopher 用户的主目录相同。如果您想使用不同的目录，您要用命令 chown 把目录的所有权改为 Gopher 用户。当配置您的 Gopher 服务器软件时，确信指定的目录是您的 Gopher 数据目录。否则，服务器将找不到您的 Gopher 文件。

### 12.6.2University of Minnesota Gopher

在下面的例子中，用户下载了 Linux 版本的 University of Minnesota Gopher 软件包，名为 gopher2_3.tar.gz。首先用 gunzip 解压此软件包，然后用 tar 展开文件和目录。将产生一个名为 gopher2_3 的目录。此目录中有包含有文档和应用的不同的子目录。gopherd 目录中含有 Gopher 服务器的源代码，gopher 目录中有 Gopher 客户程序的源代码。doc 目录中包含文档，包括您的帮助文档。

```
#gunzip gopher2_3.tar.gz
#ls
gopher2_3.tar
#tar xvf gopher2_3.tar
#ls
gopher2_3  gopher2_3.tar
#cd gopher2_3
#ls -F
Copyright  MANIFEST  Makefile.config.in  Makefile.in  README
conf.h
config.guess  config.h.in  config.sub  configure  configure.in
copyright  doc/  gopher/  gophfilt/  install-sh  make.com
object/  patchlevel.h
```

为了安装 Gopher 服务器，首先要在 gopherd 目录中的一个配置文件中指定选项。您要提供一些信息，象在什么目录中放置您的 Gopher 菜单文件。接着编译 Gopher 软件。编译软件只不过是 Gopher 在源代码目录下输入命令

make。make 将使用 Makefile 文件正确地为您编译 Gopher 程序。University of Minnesota 的 Gopher 2.3 有配置功能，它能自动决定如何配置您的系统并为系统创建合适的 Makefile 文件。任何系统的特定的信息都被精确地设定在配置文件中。

## 配置 University of Minnesota 的 Gopher 服务器

在创建 Gopher 服务器之前，您要用文件 gopherd.conf 和 gopherdlocal.conf 中的项来配置它。在 University of Minnesota 的 Gopher 服务器中，这些配置文件在子目录 gopherd 中。gopherd.conf 是用来配置系统的一些特定特征，象限制的连接数。gopherdlocal.conf 为您的 Gopher 服务器提供信息，象管理员的名字和控制指定的远程系统的访问。

gopherd.conf gopherd.conf 和 gopherdlocal.conf 文件中包含 Gopher 服务器的配置说明。文件中已列出一组命令的缺省规范。您只须删除每行开始的#，以除去它的注释功能，并按您的需要改动任何值。

您能设置选项象允许的最大用户数或传输文件的压缩方法。输入选项时，首先输入选项的说明符和一个冒号，加上一个空格和选项的值。下面的例子是设定最大的用户数：

MaxConnections: 15

您必须为系统中的 Gopher 服务指定一个别名。用 hostalias：项来指定。通常此项指的是您的系统的主机名的全称，有的系统可以是主机名的一部分，比如是 gopher。下面的例子中 hostalias 表示主机名的全称的别名，garnet.train.com

hostalias: garnet.train.com

您还需要为文件 gopherdlocal.conf 指定全路径名。用命令 include 加上文件 gopherdlocal.conf 的路径名即可。在您的 Caldera Network Desktop 中，gopherd 服务器和文件 gopherd.conf、gopherdlocal.conf 应该安装在目录 /usr/sbin 中。当 gopherd 服务器运行时，它首先读取文件 gopherd.conf 的配置信息，接着读取文件 gopherdlocal.conf 的特定的配置信息。在文件 gopherd.conf 中的 include 操作需要 gopherdlocal.conf 的全路径名，在此为 /usr/sbin/gopherdlocal.conf。

include: /usr/sbin/gopherdlocal.conf

用 ignore 和 ignore_patt 选项，您能限制 Gopher 用户可访问的文件。ignore 将拒绝访问带某一扩展名的文件，例如，ifnore: conf 将拒绝任何对扩展名为 .conf 文件的访问。ignore_patt 限制有特定模式的文件，例如，ignore_patt bin 将限制任何对名字中包含模式“ bin ”文件的访问。

gopherdlocal.conf gopherdlocal.conf 文件中拥有本地的用户定制。此文件中的项将覆盖文件 gopherd.conf 中可与之相比较的项。在文件 gopherdlocal.conf 中您指定一些管理信息象系统管理员的名字和您的 Gopher 服务的描述。在此文件中有两项是关于系统管理员的：Admin:和 AdminEmail:。用 Admin:项来加一个系统管理员的名字。您还可以加一些其他的信息，象一个电话号码。用 AdminEmail:项来指定系统管理员的 email 地址。

Admin: Richard Petersen

AdminEmail: rp@richlp.com

一系列的项提供关于您的 Gopher 服务器的信息。项 Abstract:将显示给系统中的用户一个布告。它简明的描述您的 Gopher 服务器提供的服务。项 Language 将告诉用户它的大多数的文件使用什么语言。您应该设定这两项。

别的信息项都是可选的。在项 Org:中您可以输入您的组织名，对于 Loc:项您能输入您的地址。项 BummerMsg:指定一个信息，当您的站点中有太多的用户使用而不能再被用户访问时，产生错误并显示此信息。

用 access:项您能限制其他用户访问。一个 access:项下面的格式：

access: hostname permission-list num

hostname 是主机名或是网络或远程系统的 IP 地址。num 指定从远程网络或系统中同时允许多少用户访问。permission-list 设定远程网络或系统中的用户的访问权限。有四种可能的权限：浏览，ftp，读和查找。为了禁止某权限，您可以在它前面加上一个感叹号（！）。浏览权限允许用户列出目录中的文件，ftp 允许您的系统作为一个 ftp 网关，读权限允许访问文件，查找权限允许访问索引。

对于 hostname，您能指定一个网络地址，一个特定系统的地址或用缺省项以使得任何用户都可以访问。缺省项是指用关键字“default”。下面的例子中，Gopher 服务器对任何用户开放读和查找权限，当时关闭浏览和 ftp 权限。没有网络或系统能同时有 15 个以上的用户访问您的 Gopher 服务器。

access: default !ftp read search !browse 15

您能用 access:项加上网络的域名来控制一个网络的访问。下面的例子中，网络 train.com 中最多能有 5 用户访问您的 Gopher 服务器，但是没有浏览功能。

access: train.com ftp read search !browse 5

您能用网络或系统的 IP 地址代替域名。对于网络，您只能用 IP 地址中的网络部分。不要忘记在结束时用一个句号。在下面的例子中，网络地址为 199.189. 的网络被拒绝使用您的系统作为一个 ftp 网关。同时只能有 7 个用户访问。

access: 199.189. !ftp read search browse 7

为了给特定的系统提供访问，您能输入主机的系统名。在下面的例子中，IP 地址为 204.166.189.21 的系统能完全访问您的 Gopher 系统。如果您的系统是单人使用的单机，可以指定此系统只能有一个用户访问。

access:204.166.189.21 ftp read search browse 1

Makefile.config 和 conf.h 在编译 Gopher 服务器之前，应该检查您的 Makefile.config 和 conf.h 文件中的某些配置。在文件 Makefile.config 文件中首先检查要操作的 Gopher 程序的路径名。它被指定给变量 PREFIX。如果您想在不同的目录中操作 Gopher，改变赋给变量 PREFIX 的缺省路径名。Galdera Network Desktop 将从目录/usr/sbin 中查找服务器程序。您应给变量 PREFIX 赋值为目录/usr/sbin。还能改变其他的目录变量。

PREFIX=/usr/sbin

您还应该检查 DOMAIN，SERVERPORT，SERVERDATA 和 SERVEROPTS 变量的项。DOMAIN 变量指定您的域名的网络部分。例如 richlp.ix.com 的网络部分应是.ix.com。一定要带有前一个圆点。如果您的域名命令能显示您的全域名，您可以让此项为空。

DOMAIN=.ix.com

SERVERPORT 变量被设置为 70，为通常的访问您应该保留此值。变量 SERVERDATA 有 Gopher 数据文件所在的目录，缺省值为/gopher-data。如果您想用另一个目录，能把此目录赋给 SERVERDATA。在 Caldera Network Desktop 中您应该设置为/home/gopher。变量 SERVEROPTS 被赋以一列选项以控制您的 Gopher 服务。您能在标题中加入日期和时间或设置的最大用户数。这些选项列在表 12-10 中。

SERVERDATA=/home/gopher

在文件 conf.h 中，您能设置一些特征的值，象超时的时间，和某一特定操作，象看一个图形文件时使用的程序。在文件 conf.h 中含有一列定义项，它们和 C 编程中的定义项很相似。每项以#define 开头，加上一个大写的项目和一个值。为了改变一个特定项目的值，用文本编辑器编辑此文件，删除此值并输入一个新值。缺省的值已经输入。下面的例子设定打印选项为值：lpr 命令。

#define PRINTER_COMMAND "lpr"

对于 Gopher 客户程序您能用 CLIENT1_HOST 项来指定它将连接的缺省的 Gopher 服务器。下面的例子连接到一个 Gopher Internet 站点：gopher.tc.umn.ecu，

#define CLIENT1_HOST "gopher.tc.umn.edu"

文件 conf.h 容易被混淆。它是一个 C 程序，不是一个 shell 脚本文件。它和别的配置文件不同。在大多数的配置文件中，一个#是一个注释。而在文件 conf.h 中，它是一个定义指令的开始。在 conf.h 中，一个注释是用/*开头并以*/结尾的。加注释的项便失去作用，要使它起作用只须删除/*和*/符号。

虽然文件 conf.h 的缺省的项工作的很好，您能按需要对它们做改变。然而，改变用于 Linux 系统的项时要小心。Linux 将使用那些不是专为其他系统设计的项。如果您看到一个项前有一个#if defined(system)，system 是一个操作系统的名字，然后下面的项将用于此系统，直到下一个#endif。因此#if defined（sun）将只用于 Sun 系统。conf.h 文件中有一大部分是为 VMS 操作系统定义的，在文件的开头和结尾有些项是用于 Linux 的。

您现在可以开始编译您的 Gopher 软件。能用下面的命令编译 Gopher 客户程序和 Gopher 服务器：

#make install

您可以分别编译客户程序或服务器，用命令“make”加上项目“client”或“server”。

#make client

#make server

它将创建一个名为 gopherd 的服务器程序。您能从 inetd 直接调用 gopherd。

表 12-10University of Minnesota Gopher 的 Makefile.config 和 conf.h

| Makefile.config | 描述 |
|---|---|
| PREFIX | 安装软件的基本的路径名（缺省的是/usr/local） |
| CLIENTDIRGopher | 客户程序被安装的目录（缺省的是/usr/local/bin） |
| CLIENTLIBGopher | 客户程序的帮助文件 |
| MAN1DIRGopher | 客户程序的帮助文件被安装的目录 |
| MAN8DIRgopher | 服务器的帮助文件被安装的目录 |
| SERVERDIR | Gopher 服务器和配置文件被安装的目录（gopherd 和 gopherd.conf- 缺省的是/usr/local/etc） |
| CLIENTOPTS | Gopher 客户程序的选项<br>-DONOMAIL 远程用户不能邮寄文件<br>-DAUTOEXITONU 同时处理 q 和 u，并自动地从主菜单中退出（如果 Gopher 被其他应用程序调用时，此选项很有用） |

| | |
|---|---|
| DOMAIN | 如果您的系统不能返回有效的全域名，您要指定系统的网络地址；否则，保持为 null |
| SERVERPORT= num | 设置 Gopher 服务器用来等待请求的端口，通常为 70 |
| SERVERDATA= pathname | 对于 Gopher 服务器可用的 Gopher 数据文件的所在目录；对于 Caldera Network Desktop 此目录为 /usr/lib/gopher-data |
| SERVEROPTS= options-list | -ADD_DATE_AND_TIME 往 Gopher 标题中增加日期和时间 |
| Gopher 服务器的选项 | -DDL 设置支持 dl 数据库功能<br>-DCAPFILES 提供和 .cap 目录兼容<br>-DLOADRESTRICT 限制用户访问<br>-DSETPROCTITLE 设置 ps 显示的进程标题 |
| DLPATH | 到 dl 数据库的路径 |
| conf.h | |
| CLIENT1_HOST | 首先连接的主机（列在这的是缺省的主机）<br>#define CLIENT1_HOST "gopher.tc.umn.edu" |
| CLIENT2_HOST | 连接的主机 #define CLIENT2_HOST "gopher2.tc.umn.edu" |
| CLIENT1_PORT | 首先连接的端口（缺省的是 70）<br>#define CLIENT1_PORT 70 |
| CLIENT2_PORT | 连接的端口（缺省的是 70） |

| | |
|---|---|
| PAGER_COMMAND | 文本的分页命令 |
| MAIL_COMMAND | Gopher 用户发送邮件的命令（缺省的是 /bin/mail） |
| TELNET_COMMAND | telnet通话使用的命令（缺省的是 telnet） |
| PAINTER_COMMAND | 打印的命令（缺省的是 lpr） |
| PLAY_COMMAND | 播放声音的命令（缺省的是/bin/false） |
| IMAGE_COMMAND | 显示图象的命令（缺省的是 xloadimage） |
| HTML_COMMAND | 显示 Web 页的命令（没有缺省值，一般用 lynx） |
| MAXLOAD | 限制连接平均负载（缺省是 10.0） |
| READTIMEOUT | 从网络中读取数据的超时时间（缺省的是 1*60） |
| WAISMAXHITS | 从一个 WAIS 查询中返回的最大的查找数目（缺省为 40） |
| NOMAIL | 如果定义此项，则限制用户邮寄和下载文件（当前用/* */注释掉，除去注释便使此项可用） |
| GOPHERHELP | 在线的 Gopher 帮助文件 |
| TN3270_COMMNAD | TN3270 通话的命令 |
| conf.h | |
| MIME_COMMAND | MIMI操作的命令 |
| AFTP_HOST | 设置 ftp 网关（当前被设置为 gopher-gw.micro.umn.edu） |
| CONF_FILE | 设置 gopherd.conf 文件所在的位置；缺省的是 /usr/local/etc/gopherd.conf；对于 Caldera |

| | |
|---|---|
| | Network Desktop 您要把它改成/usr/sbin<br>#define CONF_FILE /usr/sbin |
| DELETE_BOOKMARKS_ONLY | 限制用 delete 命令删除书签（当前被注释掉） |

## 启动 University of Minnesota 的 Gopher

在启动您的 Gopher 服务器之前，首先应该创建一些 Gopher 菜单文件，在下一部分中将对此进行描述。正如前一部分“启动服务器”中描述的那样，您能从命令行中直接启动 Gopher 服务器，或在使用 init 脚本引导系统时启动，或是当收到一个 Gopher 服务请求时通过 inetd 来启动。University of Minnesota Gopher 服务器选项将列在表 12-11 中。

表 12-11University of Minnesota Gopher 服务器选项

| 选项 | 描述 |
|---|---|
| -C | 取消目录请求的高速缓存 |
| -c | 不受 chroot的限制运行 ;允许通过符号连接访问 Gopher 数据主目录外的文件，象系统的帮助文件；有潜在的安全问题，应该和-u username 选项一起使用 |
| -D | 启动调试功能 |
| -I | 用 inetd 调用 Gopher |
| -L num | 指定最大的平均负载 |
| -l logfile | 记录到日志文件的连接 |

| | |
|---|---|
| -o conf-file | 指定备用 gopherd.conf 配置文件 |
| -u username | 在指定的用户名下运行 gopherd；为安全目的提供附加的限制 |
| -U userid | 在 userid 下运行 gopherd；为安全目的提供附加的限制（同 -u） |

Caldera Network Desktop 已经配置您的系统，使之通过 inetd 来运行 Gopher。此配置将把 Gopher 数据目录定为 /home/gopher。在 /etc/services 文件中您将找到一个关于 Gopher 的项。它通过端口号和协议指定服务名。在别的系统中，您要自己输入它。

gopher70/tcp

在文件 /etc/inetd.conf 中也有一个关于 Gopher 的项。当前此项引用 GN Gopher 服务器，gn。您能把它改为引用 gopherd 服务器。在项中 gn 的地方替换为 gopherd，并加一个 -l 选项和 Gopher 数据目录的路径名。选项 -l 指定 gopherd 被 inetd 调用。您能加一个端口号，在此为 70。此项还被设置用 tcpd 来监视和控制远程用户的访问。下面的例子列出此项。

Gopher 数据文件在目录 /home/gopher 中，端口号为 70。gopherd 程序和参数为 gopherd -l

/home/gopher 70。

gopher stream tcp nowait root/usr/sbin/tcpd gopherd -l /home/gopher 70

如果您不想使用 tcpd，能用 gopherd 服务器程序的路径名取代它。

gopher stream tcp nowait root/usr/sbin/gopherd gopherd -l

/home/gopherd 70

如果您想在命令行启动 gopherd 服务器，可输入 gopherd 和它的参数，Gopher 数据文件目录的路径名和端口号。gopherd 有一些可能的选项，列在表 12-12 中。作为防范措施，您能用-u 选项指定一个非 root 的所有者来运行 gopherd。您需要首先创建一个用户，并在它的 passwd 项中输入一个*。用户的主目录将是 gopher 数据目录。Caldera Network Desktop 已经创建一个名为 gopher 的用户。下面的例子中，gopherd 服务器程序的全路径名为/usr/sbin/gopherd，Gopher 数据文件在/home/gopher 中，端口号为 70。

/usr/sbin/gopherd -u gopher/home/gopher 70

为使 Gopher 服务器在启动系统时自动启动，您需要在目录/etc/rc.d/init.d 中创建一个 init 脚本。本章的开始部分对 init 脚本和如何创建 gopherd 服务器的 init 脚本有一个详细的描述。

表 12-12gopherd.conf 和 gopherdlocal.conf 项

| gopherd.conf | 描述 |
| --- | --- |
| hostalias: DNS-alias-name | 用此主机名来代替系统的主机名，必须有一个 DNS，用来引用一个系统，比如 gopher.ix.com 上 Gopher 服务器 hostalias: gopher.turnip.com |
| cachetime: seconds | 一个缓冲文件中保存的 Gopher 目录的时间 |
| viewext: extension Gophertype Prefix Gopher+type | 变换文件名的扩展名成为一个特定的 |

| | |
|---|---|
| [Language] | Gopher 类型；大多数的名字已经在文件 gopherd.conf 中设置了。第一个参数是一个扩展名，象.gif；第二个参数是单个字符的 Gopher 类型（1，0，I 等）；第三个参数是要加到一般文件名路径前的前缀；第四个参数是 Gopher+视图特征或 Internet Media 类型（正规叫做 MIME 内容类型），象 image/gif；可选的第五个参数是指定文件使用的语言以替换缺省的语言 viewext; .jpg I 9 image/JPEGviewext: .html h 0 text/html |
| ignore: extension | 忽略带有指定扩展名的文件；这些文件不提示给 Gopher 的用户 ignore:bin |
| ignore_patt: regular-expression | 忽略匹配 regular expression 的文件；这些文件不提示给 Gopher 的用户 |
| blockext: extension | 映射带有指定扩展名的文件到属性块中 |
| decoder: extension program | 当检索到文件时，用指定的程序对带有指定扩展名的文件进行操作；和压缩文件一起使用 decoder: .gz /usr/gnu/bin/zcatpids_directory: path-name 一个存放 pid 文件的临时目录 |
| maxconnections: num | Gopher 服务器能同时处理的并发连接的数 |

| | |
|---|---|
| | 目 |
| gopherdlocal.conf | |
| admin:administrator-name-and-info | 管理员的名字和附加信息，象电话号码 |
| adminemail:email-address | Gopher 服务器的管理员的 Email 地址 |
| site:site-description | 站点的描述名 |
| loc:address | 站点的地址：街道，城市等 |
| geog: | 站点的经度和纬度 |
| language:default-language | 站点使用的缺省语言 |
| secureusers:filename | 列出经过认证的主机和网络的文件 |
| bummermsg: | 当拒绝一个客户访问时显示的信息 |
| access:domain name access-list | 允许您决定谁能浏览目录、阅读文件和在搜索您的系统，第一个参数时域名、IP 地址或缺省；第二个参数是一列逗号分隔的单词用来控制访问，这些单词是：browse，read，search 和 ftp。每个单词前加一个！就可拒绝此种访问；象 default !browse,read,search,!ftp 设置缺省为拒绝浏览和 ftp，但允许读和查找。如果您首先设定缺省值，以后的项中将继承这些属性，您只需要指定不同的项。另外，您还可以通过增加一个可选的参数来指定 |

并发事务的访问数 access: default browse,read,search,ftp 5

### 12.6.3GN Gopher服务器

您的Caldera Linux系统已经配置了GN Gopher服务器。如果您选择完全安装的选项，GN Gopher将自动安装，如果选用推荐的安装选项，您要用rpm或glint来安装gn-2.22-l.i386.rpm。无论选择哪种安装方式，现在都可以使用您的Gopher服务器了。您可以开始把Gopher文件和菜单加到/home/gopher目录中。要运行mkcache来使得gopher客户程序能访问您的gopher文件。然而，如果用的是Linux的另外版本。您要能从一个Linux ftp站点中下载标准的GN Gopher服务器软件包，gn-2.20.tar.gz。然后配置并安装它。解压和展开此文件将创建一个名为gn-2.20的目录，其中放有GN Gopher服务器的源程序和文档。

GN Gopher软件包中包含一些在编译之前需要为Linux配置的源代码。还有一个配置文件，config.h和一个Makefile文件中包含编译指令。在doc子目录中的文件INSTALL有关于如何创建您的GN服务器和设置Linux站点的详细说明。在目录/doc/examples中有GN Gopher菜单的大量的例子。

### 配置GN Gopher服务器

为了配置GN Gopher服务器，您要改变文件config.h和Makefile中的项。有些项是必须配置的,您需要指定它们。在文件的开头的标题“Compulsory items tofill in”中指定了这些项。必须输入您的主机名和Gopher数据目录的路径名，

以及指定您使用的 Linux 系统。还可以按照需要进行其他的定制。在文件 conf.h 中每项的格式为：一个#define 加上一个项目和它的值。您能改变此值。下面列出的项对 Caldera Network Desktip 版本的 Linux 是有效的。实际应用中，应该用您的系统主机名取代 gannet.train.com。

```
#define GN_HOSTNAME    "garnet.train.com"
#define ROOT_DIR    "/home/gopher"
#define LINUX
#define MAINTAINER      "mailto: justin@garnet.train.com"
#define ROOT_MENU_NAME  "GN  --  A Gopher/HTTP Server"
#define GN_LOGFILE  "/var/log/gn.log"    /* "/path/to/gn.log" */
#define MIME_TYPE_FILE  "/usr/local/gn_mime.types"
#define WAISGN  "/usr/sbin/waisgn"
```

在必须配置的项后有一些其他的项，它们在缺省状态下能很好地工作，但您也可以按需要对它进行改变。比如您能通过指定 DEFAULTPORT 项为另一个端口号，从而使用此端口。您能设置 TIMEOUT 项为超时的时间，设置 MAXDEPTH 项为菜单查找的最大深度。在 Makefile 中，可以指定您使用的 C 编译器的类型和您的服务器程序放置的目录。您将能找到两个关于 C 编译程序 CC 的类型的项。一个指定为 gcc，前面加有一个#把它注释掉，而另一项指定为 cc，前面没有#号。您可以在 cc 前加一个#，而去掉 gcc 前的#。这样将设置 CC 项为 gcc 编译器。另外在 Makefile 中您能设置 SERVERBINDIR，它是您放置的服务器程序的目录路径。BINDER 是 mkcache 和 uncache 程序的目录。将一些目录设置成放置程序的目录，象/usr/sbin。在您的 Caldera Network Desktip 中用 Redhat 版本，目录 SERVERFINDER 应被设置为/usr/sbin，此目

录中有守护进程。虽然 mkcache 和 uncache 程序可以放在任何目录中，最好还是把它们和 GN 服务器放在一起，只须把 BINDER 也设定为/usr/sbin 即可以做到这点。

在这些变量的下面，将是关于 include 目录的项。如下所示，除去关于 Linux 的项前面的#，以使它们能起作用。

```
# INCLUDES= -I..   -I../gn
# For Linux use
INCLUDES= -I .. -I../gn -I/usr/include/bsd
```

在文件 Makefile 中，再往下便是一个关于库函数的空项，如下所示。

```
#Libraries to be included.
LIBS=
```

下面的项是为不同的系统指定不同的库函数，这些项被注释掉。关于 Linux 的项如下所示：

```
#For Linux use
#LIBS = -lbsd
```

您可以除去 LIBS 项前的#号，或是在上面的空 LIBS 后加上-lbsd 的值。

```
LIBS= -lbsd
```

一旦配置完成后，您能输入 make 来创建一个服务器。它将产生两个可执行的服务器程序叫做 sgn 和 gn。gn 是为 inetd 使用的，而 sgn 是能直接使用的一个独立的守护进程。还将产生两个应用程序，mkcache 和 uncache。然后输入 make install 在您的系统中安装这些程序。

## 启动 GN Gopher 服务器

现在可以启动服务器了。在启动 Gopher 服务器之前，您应该首先创建一些如下一部分所描述的 Gopher 菜单文件。您可以启动服务器做为一个独立的守护进程或通过 inetd 来启动服务器。见表 12-13 和 12-14

Caldera Network Desktop 已经把您的系统配置为通过 inetd 运行 GN Gopher 服务器。创建和安装 GN Gopher 服务器后，为了有一个全功能的 Gopher 服务器，必须创建您的 Gopher 数据文件并把它们放在目录 /home/gopher 中。

为了使用 inetd，在文件 /etc/services 和 /etc/inetd.conf 中必须有关于 Gopher 服务器的项。Caldera Network Desktop 已经放置此项。您能在文件 /etc/services 中找到下面的项：

```
gopher 70/tcp
```

在文件 /etc/inetd.conf 中，您将能找到关于 GN Gopher 服务器的项。注意 GN Gopher 服务器是被 gn 调用的。此版本的 GN 服务器是通过 inetd 工作。另外，tcpd 用来监视和控制 Gopher 访问。

```
gopher stream tcp nowait root /usr/sbin/tcpd  gn
```

您也可以在引导系统时用 init 脚本或直接从命令行启动 GN Gopher。在这两种情况下，服务器使用 GN Gopher 的 sgn 版本来启动，并以一个独立的守护进程运行。如果您想在引导时启动 Gopher，您需如前面的“启动服务器”中所讲述的创建一个目录 /etc/rc.d/init.d 中的 init 脚本。输入 sgn 和参数，用来指定 Gopher 数据文件的目录的路径名和端口号（通常为 70）。端口号用 -p 选项来指定。作为一种预防措施，sgn 将按照用户在文件 config.h 中的 USERID 项的指定来自动运行。

sgn -p 70 /home/gopher

表 12-13GN 服务器配置文件

| | |
|---|---|
| GN_HOSTNAME | 显示您的服务器的主机名；在此是您的系统的主机名 |
| ROOT_DIR | 设置 Gopher 数据目录；对于 Caldera Network Desktop 它是/home/gopher |
| LINUX | 定义 Linux 做为您的操作系统；缺省的是 SUN_OS；您要用 LINUX 代替它 |
| MAINTAINER | 管理员的邮件地址 |
| ROOT_MENU_NAME | 您想显示的做为 Gopher 菜单标题的名字 |
| GN_LOGFILEGN | 记录文件的位置 |
| MIME_TYPE_FILE | MIME 配置文件的位置；缺省的是/usr/local/gn-mime.types；您能把文件放在任何地方，但是确认设置正确的目录 |
| WAISGN | 控制 WAIS 索引/usr/sibn/waisgn 的 waisgn 程序的路径名 |
| DEFAULTPORT | 设置 Gopher 访问的缺省端口，当前是 70 |
| TIMEOUT | 设置等待请求的超时时间 |
| MAXDEPTH | 设置查询菜单的最大长度 |
| USERID | 为了安全，运行 GN 服务器所使用的用户 ID |
| GROUPID | 为了安全，运行 GN 服务器所使用的组 ID |

| | |
|---|---|
| DECOMPRESS | 解压文件所用的程序；缺省的时/usr/local/bin/zcat |
| MENUFNAME 设 | 置菜单文件所使用的名字；缺省的是 menu |
| TEMPDIR | 设置临时目录；缺省的是/tmp |

表 12-14GN 服务器的 Makefile 文件

| | |
|---|---|
| CC | 设置 C 编译器；缺省的设置是 cc；您能把它改成 gcc；CC=gcc |
| INCLUDE | 设置 Include 目录 ;应该被设置使用/usr/include/bsd |
| SERVBINDIR | 您的 GN 和 SGN 服务器的程序；缺省的是../bin；应该设置为/usr/sbin |
| BINDIR | 缓冲支持程序的目录；缺省的是../b；应该设置为/usr/bin |
| LIBS= | 设置使用的库；除去关于 Linux 的那一项前的#号；应该设置为-lbsd |

### 12.6.4 测试 Gopher 服务器

当您的 Gopher 服务器运行时，您能用 telnet 或 Gopher 客户程序来测试它。用 telnet 时，首先 telnet 到您自己的系统中并指定 Gopher 使用的端口。下面的命令将测试在系统 garnet.train.com 中的一个 Gopher 服务器。将显示启动信息。

```
telnet garnet.train.com 70
```

如果按 ENTER，将显示您的服务器上的 Gopher 主目录中的菜单项，并以

下面的格式输出：

type display-name selector hostname port

下面是一个测试 Gopher 服务器的例子：

```
#telnet garnet.train.com 70
Trying 127.0.0.1 ...
Connected to garnet.train.com
Escape character is '^] '.

0About My Weather Site 0/intro    garnet.train.com    70    +
1California Weather Information    1/calif    garnet.train.com    70    +
1New york Weather this week    1/newyork    garnet.train.com    70    +
1The Weather in Hawaii    1/weather/hawaii/
garnet.train.com    70
```

还可以用 Gopher 客户程序来访问您的 Gopher 服务器。下面的例子中，用 garnet.train.com 中的 Gopher 客户程序访问本系统中的 Gopher 服务器。将显示 Gopher 服务器的主菜单，用户能从中选择或访问。

```
gopher garnet.train.com
```

### 12.6.5Gopher 目录

当您访问一个 Gopher 站点时，将显示一个 Gopher 菜单。此菜单是用包含在一个 Gopher 目录中的特定的专用文件生成的。Gopher 菜单被设计为通过目录进行操作。目录将列出其中可用的文件或引出另一个目录。每个目录中专用 Gopher 菜单配置文件提供一些信息，这些信息是关于不同的可用数据文件以及

如何访问它们。University of Minnesota Gopher服务器用.cap目录和连接文件来组织Gopher菜单。GN Gopher服务器用一个menu和.cache文件。然而，连接文件和GN menu文件的项基本是相同的。

## Gopher cap和连接文件

缺省情况下，一个Gopher目录中的所有文件和子目录将自动地以字母顺序显示在一个Gopher菜单中。数据文件的类型为0，目录为类型1。每个菜单项的名字是一个文件名或目录名。用cap文件能重写此列表。

您能给Gopher数据文件改名。而通常用一个描述语句在Gopher菜单中对Gopher文件进行说明。通过选中菜单项，也就选中伴随着此语句的文件。或者通过一个.cap目录中专用文件或者通过扩展的连接文件中的项来实现描述语句和Gopher数据文件的关联。

Gopher数据文件的每个目录中都有自己的.cap目录。.cap目录中的文件和在Gopher文件数据目录中的文件有相同名字。如果您有一个Gopher数据文件名为engine.1，在.cap目录中也有一个文件名为engine.1。在.cap目录中的文件包含三项：名字、类型和Numb。名字被赋值为引用的Gopher数据文件的菜单项的描述语句。类型项指定Gopher资源的类型。1代表一个目录，0代表一个文本文件。Numb项被赋值为Gopher菜单中的您的Gopher项的序号。例如，Numb=3表示这是Gopher菜单中的第三项。

```
.cap/engine.1
Name=The best engine in the world
Type=0
Numb=1
```

当在 Gopher 菜单中显示此项时，它将被显示在第一项。选中它将选中 engine.1 数据文件。

1. The best engine in the world.

虽然.cap 文件能被用来引出您的目录中的数据文件，但它们不能引出其他目录或其 Gopher 站点中的文件。为了补充这点，您需要使用连接文件中的连接项。一个目录中的连接文件可以有许多连接项以连接不同的 Gopher 资源。一个连接文件是任何以圆点开头的文件。一般的连接文件的名字为：.links。一个连接的每一项都有五个变量集：名字、类型、端口、路径和主机名。您还可以按菜单顺序加一项，NUMB。名字项是用在菜单项中的描述语句。端口是用来连接远程系统的端口，它通常是 70。类型是菜单项引用的资源类型。一个资源能是一个文件，它也能是一个 telnet 通话或一个图形文件。Gopher 不能引用数据文件。路径变量含有菜单项引用的资源的路径名。在此的路径名是一个以 Gopher 数据目录开始的路径名。目录/home/gopher/calif 将有一个路径名/calif，在此/home/gopher 是 Gopher 数据目录。路径名前可以加上类型。calif 目录可以是 1/callif，1 代表目录的类型。主机名是资源所在的主机名。从您的系统引用的资源将是您的系统的主机名。如果资源位于别的系统，它将是那个系统的主机名。用一个+赋给端口和主机名项将表明使用当前的端口和主机名。对于您自己系统中的文件和目录，最好省去端口和主机名项。

```
.links
Name=The best engine in the world
Type=1
Port=70
Path=1/engines
```

Host=richlp.ix.com

您可以用连接来设定用 ftp 或 WAIS 连接到其他系统中以访问它们的文件和信息。这时您使用的服务和它的参数将指定在路径项中。主机名和端口都是一个+。一个 ftp 连接的格式如下所示：

Name=ftp-fiile-or-directory
Type=1
Path=ftp:hostname/path/
Host=+
Port=+

例如，为了设定一个 ftp 连接以访问 chris.train.com 中的文件 caboose1，您应如下所示设置路径。当前的端口和主机名都是+。

Name=The last caboose
Type=0
Port=+
Path=ftp:pango1.train.com/usr/lib/gopher-data/caboose1
Host=+

从一个 WAIS 连接，能访问您的系统或远程系统中的 WASI 资源。对于您的系统，用 waisrc:加上一个 WAIS 资源的路径。对于远程系统，在 waisrc:后还要加上主机名。例如：

waisrc:pango1.train.com/wais/data

您还可以设置一个连接以执行 shell 脚本，而不仅仅只是访问资源。在这种情况下，路径变量被设置为 exec:加上脚本参数和脚本名。参数用双引号括起来，如果没有参数，用一对空的双引号。参数和脚本之间用一个冒号分隔。

path=exec:"arguments":script

除了用每个目录中各自的连接文件来管理各自的.cap 文件外，您能用一个扩展的连接文件来包含本地文件项和连接项。这样的连接文件一般名为.names。.names 将列出名字项和 Numb 项，并有一个指明文件路径的路径项。文件.names 还有一个摘要项，您能为它输入一个文件内容的简明描述。在下面显示的文件.names 的第一项引用一个本地文件，而第二项引用一个远程 Gopher 站点。

```
.names
Path=/engines
Name=The best engine in the world
Numb=1
Abstract=A discussion of the best steam engine ever built.

Name=The oldest train running
Type=1
Port=70
Path=1/museums
Host=pang01.train.com
```

GN Gopher 目录

GN Gopher 目录放置菜单项在一个名为 menu 的文件中。每个 Gopher 目录有它自己的 menu 文件，文件中有关于每个菜单项的项目。GN Gopher 使用和 University of Minnesota Gopher 相同的项目集，仅有一些例外。在文件 menu 中，您为每个菜单项列出路径、名字、Numb 和摘要。然而，在路径项的路径

名之前，您必须指定 Gopher 类型。它通常是一个数字项，0 表示文本文件，1 表示目录。

```
Name=About My Weather Site
Path=0/intro
Type=0
Numb=1
Abstract=Important Weather Information
```

GN Gopher 要求为每个目录创建一个.cache 文件。在目录中执行 mkcache 程序即可生成一个.cache 文件。您能在每个 gopher 目录中用此方法创建.cache 文件，或从 gopher-data 主目录中执行 mkcache 加上-r 选项。命令 mkcache -r 将为当前目录和它的所有子目录产生.cache 文件。在您的 Caldera Network Desktop 中,您能从/home/gopher 目录中运行 mkcache -r 来为您的所有 Gopher 目录创建.cache 文件。

GN Gopher 也完全支持 Web 页。它能显示 Web 页,并通过 Web 页的 HREF 引用它的菜单项。

参见 GN 文档中关于这些功能的详细描述。

### 12.6.6Gopher 索引：gopherindex

用 gopherindex 命令能为您的 Gopher 数据文档创建全文索引。gopherindex 在 University of Minnesota Gopher 软件包中，它用 waisindex 来执行它的索引，所以您必须安装有 WAIS。它使用参数的几个可能选项和一个 Gopher 数据目录。在数据目录中的所有文档将被做一个索引。-N 选项指定用在一个 Gopher

菜单中的索引文件的一个描述。下面的例子将对目录/home/gopher/baseball 中的文件进行索引。

```
/usr/sbin/gopherindex -v -N "Search CIS Services Short Courses"
/home/gopher/baseball
```

除了使用 gopherindex，您还能直接使用 waisindex 并对索引文件创建一个连接。

```
waisindex -r /home/gopher/baseball
```

在目录连接或 menu 文件中您能为索引文件创建连接项。类型 7 是一个文档 WAIS 索引类型。

```
Type=7
Name=Baseball Index
Host=+
Port=+
Paht=7/.index/index
```

### 12.6.7 一个用 Gopher 例子

下面是一个简单的 Gopher 例子。它是一个 Gopher 站点，站点上的文件是关于一些州的天气信息。每个州的文件放在它自己的目录中。开始的 Gopher 菜单在 Gopher 数据目录中。此菜单项被定义在.name 文件中（对于 GN Gopher 将是名为 menu 文件）。大多数的菜单项引用其他的 Gopher 目录。有一个菜单项是对 Gopher 站点的介绍信息的文件。在主 Gopher 目录中只有这个介绍文件和.name 文件。下面的两项连接到本地 Gopher 目录，calif 和 newyork。最后的一项连接到另一个 Gopher 站点 garnet.train.com。

.names

Path=/intro
Name=About My Weather Site
Numb=1
Type=0
Abstract=Important Weather Information.

Name=California Weather Information
Numb=2
Type=1
Path=/calif

Name=New York Weather this week
Numb=3
Type=1
Path=/newyork

Name=The Weather in Hawaii
Numb=4
Type=1
Port=70
Path=/usr/lib/weather/hawaii/
Host=garnet.train.com
California 菜单项引用 calif 子目录。此目录下有一些关于 California 天气信

息的文件。目录 calif 中列出两个数据文件，一个连接文件和一个.caps 目录。在.cap 目录中有两个文件，这两个文件和目录 calif 中两个数据文件有相同的名字。对于 calif 目录中 surf.data 和 storm.data 文件，在.caps 目录中也有两个名为 surf.data 和 storm.data 的文件。下面显示的是.caps 目录中的文件。

```
surf.data
Name=Surfing Conditions
Numb=1
Type=0
storm.data
Name=Storm Advisory
Numb=2
Type=0
```

在 calif 目录中，.link 文件列出一个访问远程系统中文件的项。

```
.links

Name=The Weekend Snowpack
Numb=3
Type=0
Port=+
Path=ftp:rose.net.com/usr/lib/weather/snow/current.txt
Host=+
```

对于 GN gopher，menu 文件如下面所示，每个路径名前都有一个文件类型（0 代表文本文件，1 代表目录）。

```
menu
```

```
Name=About My Weather Site
Path=0/intro

Name=California Weather Information
Path=1/calif

Name=New York Weather this week
Path=1/newyork

Name=The Weather in Hawaii
Path=0/usr/lib/weather/hawaii/
```

## 12.7WAIS 服务器

WAIS（广域信息服务）用关键字查找一个文档数据库，并按照它们的重要程度排列和显示结果。这是一种非常有效的方法使信息能在网络上使用。WAIS 是由 Thinking Machines 开发的，现在由 WAIS Inc 管理。WAIS 的免费版本称为 freeWAIS，它能从 Clearinghouse for Networking Inofrmation Discovery and Retrieval(CNIDR)获得。您还能从 CNIDR 和 Linuxftp 站点上取得 freeWAIS 的 Linux 版本。

freeWAIS 软件包中包括客户程序、一个服务器程序和一个索引程序（参见表 12-15 和 12-16）。客户程序被称为 swais、xwais 和 waissearch。它们是用

来输入请求和显示结果。索引软件被称为 waisindex。您能用它为您的 WAIS 文档创建一个关键字索引，以提供快速有效的访问能力。服务器软件被称为 waisserver。用它您能创建自己的 WAIS 站点并允许其他的用户在您的 WAIS 文档上执行查找。

表 12-15WAIS 服务器：z3950

| 选项 | 描述 |
|---|---|
| -p portnum | 监听端口，如果提供 portnum，则用端口号为 portnum 的端口 |
| -s | 监听标准的输入/输出 |
| -d directory | 用此目录做为索引的缺省位置 |
| -e path-name | 把错误输出重定向到 path-name，如果为指定此目录，则重定向到/dev/null |
| -l log_level | 设置记录的级别 0、1、5 和 10<br>1 只记录错误和警告信息<br>5 记录 MEDIUM 优先级的信息<br>10 记录所有的信息 |
| -u user | 已指定的用户运行服务器 |
| -v | 显示当前的版本和数据 |

表 12-16waisindex:选项和文件类型

| Waisindex | 文件描述 |
|---|---|
| index-name.doc | 关于文档的信息，包括大小和名字 |
| index-name.dct | 是一个字典文件，其中包含交叉索引 |

| | |
|---|---|
| | 和倒置文件的所有的关键词列表 |
| index-name.fn | 列出为索引创建的文件 |
| index-name.hl | 所有标题的表；标题是题目并当它们在查找结果中时显示它们 |
| index-name.inv | 倒排文件，其中包括一个单词表，它们重要程度的排序，它们到索引文档的连接 |
| index-name.src | 包含索引信息的源描述文件，包括它位于什么系统，它涉及的主题，谁来管理等信息 |
| index-name.status | 包含用户定义的信息 |
| waisindex | 选项 |
| -a | 给存在的一行附加索引 |
| -contents | 给文件的内容做索引（缺省） |
| -d path-name | 为索引文件指定路径名；路径名将被加到索引文件名的前面 |
| -e logfile | 重定向错误信息到 logfiile |
| -export | 为了允许 Internet 访问，加主机名和 TCP 端口到源描述文件中；否则将不包括连接信息，文件只能被本地访问 |
| -l num | 设置记录的级别 0、1、5 和 10<br>0 不记录 |

| | |
|---|---|
| | 1 只记录错误和警告信息<br>5 记录 MEDIUM 优先级的信息<br>10 记录所有的信息 |
| -mem | 索引时使用的内存数 |
| -M | 连接不同类型的文件 |
| -contents | 决定是否对文件内容做索引；-contents 对整个文件做索引； |
| -nocontents-nocontents | 只对文件名和标题做索引，不对内容 |
| -pairs | 如何对待大写单词；-pairs（缺省）把大写的单词做为一项； |
| -nopairs-nopairs | 把大写单词中字母分别做为一项 |
| -pos | 是否在索引中包括单词的位置信息；-pos 将包括此信息，并 |
| -nopos | 允许近似匹配搜索，但索引的大小将增加；-nopos 不包括此信息 |
| -r | 递归的对子目录做索引 |
| -register | 登记一个索引到 WAIS 服务目录中 |
| -t | 指定文档文件的类型 |
| -T | 设置文档的类型 |
| 文档文件的类型 | |
| filename | 用文件名做为标题行的文本类型 |

| | |
|---|---|
| first_line | 用文件中的第一行做为标题行的文本类型 |
| one_line | 对每个句子做索引的文本类型 |
| text | 对文档做索引的文本类型，标题行是路径名 |
| ftp | 包含访问系统的 ftp 代码 |
| GIPGIF | 图象文件 |
| PICTPICT | 图象文件 |
| TIFFTIFF | 图象文件 |
| MPEGMPEG | 文件 |
| MIDIMIDI | 文件 |
| HTML | 用于 Web 页的 HTML 文件 |
| mail_or_rmail | 索引邮箱文件 |
| mail_digest | 用标题行中的主题对 email 做索引 |
| netnets | 对 USENET 新闻做索引 |
| ps | Postscript 文件 |

您能从站点 ftp.cnidr.org 或大多数的 Linux ftp 站点上取得 freeWAIS。在 ftp 站点 ftp.cnidr.org 上有一个 freeWAIS 的 Linux 版本，并且是已经编译过的，可以马上使用的版本。当前用于 Linux 的软件包是 freeWAIS-0.5-linux.tar.gz，位于目录/pub/CNDIR.tools/freewais 中。您也可以下载源代码自己编译。源代码软件包叫做 freeWAIS-0.5.,tar.Z，它能按照不同的系统进行不同的配置。freeWAIS 也可以从 sunsite.unc.edu 或它的镜像站点上的/pub/packages/info-

systems/wais 目录中下载。此目录中有一些不同的版本。

创建一个您想放置 freeWAIS 的目录，通常是/home/wais。建议您下载预编译过的用于 Linux 的二进制的软件包 freeWAIS-0.5-linux.tar.gz。接着用 gunzip 解压，用 tar xvf 展开文件。将创建一个名为 freeWAIS-0.5-linux 的目录。此目录中包含那些二进制文件。您需要把它们安装到合适的目录中，象/usr/bin。如果您想下载源代码，并自己编译 freeWAIS，您首先要配置软件，然后再编译创建它的二进制文件。

### 12.7.1 配置和安装 freeWAIS 源代码：Makefile 和 ir.h

如果您下载源代码，它的目录中将有一些子目录：doc 中含有文档，src 中含有源代码。还有一个特殊的子目录，wais-test，中含有您的服务器的测试文件。

您首先要在 Makefile 文件中设定变量 TOP。TOP 需要赋值为 freeWAIS 源代码所在的目录的路径名。输入您的 WAIS 源代码所在目录的路径名。

除了 Makefile 之外，还有一些特定的 Makefile，每个文件都有一个扩展名，用来指定它是特定操作系统的 Makefile。Makefile.linux 文件便是用来创建 Linux 版本的 freeWAIS。此文件已经为 Linux 作了一些配置。然而，您能指定或改动另一些选项。下面将对这些选项进行详细的描述。这些选项被赋值给变量 CFLAGS。开始已经设置了缺省的选项。下面是一个项 CFLAGS 的例子。记住您只能修改此行。

```
CFLAG = -Wall -m486 -fwritable-strings -Who-unuesd -I$(INCLUDE)
-DTELL_USER -DUSG -SDECURE_SERVER -
```

DRELEVANCE_FEEDBACK -DBOOLEANS
-DPARTIALWORD -DLITERAL -DSOUND -DBIBDS -DLINUX

下面是一列其他的有用的选项：

Makefile 选项描述

-DBIO 允许为生物符号做索引

-DBOOLEANS 允许用 AND，OR 和 NOT 进行布尔查找

-DBINGINDER 为文档的最大集合做索引

-DLITERAL 文字串查找

-DPARTIALWORD 允许在模式匹配中使用*以匹配模式的任何变化。例如，用 hum*匹配 human、hummingbird。

-DRELEVANCE_FEEDBACK 允许您从查找中选择相关的文档并用在它们之上进行新的查找

-DSECURE_SERVER 提供最好的服务器安全

-DTELL_USER 告诉服务器谁正在连接

-DUSE_SYSLOG 使用系统日志进入系统

如果需要，可以限制您的 WAIS 服务只允许特定的网络或系统访问。为了做到这点，创建一个 SERV_SEC 文件，并输入允许访问的网络和系统的域名和 IP 地址。文件 SERV_SEC 被定义在文件 ir.h 中的 include 目录的一项，如下所示：

#define SERVSECURITYFILE "SERV_SEC"

SERV_SEC 文件中的项包含域名和 IP 地址，其中 IP 地址是可选项，例如：

pango1.train.com 204.166.189.21

您能给进一步限制到一个特定的数据库的访问，只允许某些指定的网络和

系统访问某些数据库。您需要建立一个 DATA_SEC 文件。此文件也被定义在文件 ir.h 中的 include 目录的一项，如下所示：

```
#define DATASECURITYFILE "DATA_SEC"
```

文件 DATA_SEC 中的每一项首先列出数据库，接着是域名和可选的 IP 地址。在域名的区域用一个*则表示所有的用户均可访问此数据库。下面的例子显示文件 DATA_SEC 中的项，第二项为所有的用户打开一个 oldata 数据库。

```
mydata  pango1.train.com  204.166.189.21
oldata        *                *
```

一旦您准备编译 freeWAIS，要使用下面的 make 命令和项“linux”。它将为您的 Linux 系统创建 WAIS 客户程序、索引器和服务器程序。接着首先要运行目录 src/x 中的 xmkmf 来为 xwais 创建 Makefile。

```
#make linux
```

### 12.7.2 创建索引

为了使用 WAIS，必须为可用的文档创建索引。索引的过程是由 waisindex 命令完成的，它将创建一个特定的 WAIS 数据库。您能为单个文件、一组文件或文件的整个目录和子目录创建索引。数据文件和它的索引一起组成一个 WAIS 数据库。您能分别为不同的文件或文件组建立索引，在您的服务器上建立不同的 WAIS 数据库。WAIS 数据库应该位于当

WAIS

服务器被调用时指定的 WAIS 数据目录中。

waisindex 能创建一个倒排文件索引，它将涉及指定的文件中的每个单词。

这样就可以在文档中进行全文的关键字搜索。waisindex 命令有一些选项，在选项的最后是被索引的文件、文件组或目录。用-d 选项能指定索引的名字。waisindex 将为文档创建几个索引文件。每个索引文件有自己的扩展名来说明自己的功能，但它们名字的前缀是用-d 选项指定的。如果您没在此选项中指定名字，“index”将作为前缀名。另外，如果您希望您的数据库能 Internet 上用户访问，您还要加一个-export 选项。没有此选项，您的数据库只能被您的系统中的其他用户访问。下一部分将讨论-export 选项。waisindex 的选项和索引文件列在表 12-14 中。

waisindex -d index-file -export file-list

如果您对列出的多个文件进行索引，则对这些文件将做成一个索引。如果您想对一个子目录中的所有文件做索引，只须在命令中用-r 选项加上目录名。

waisindex -d index-file -export -r directory-name

若要加一个文件或目录到一个已存在的数据库中，您要在对它做索引时用选项-a。还要用选项-d 和数据库名以把此索引文件加到指定的数据库中。您能在命令行中列出几个文件以把它们加入数据库中。如果您想加一个目录，要使用-r 选项加上目录名。

waisindex -d index-file -a -export file-list

下面的例子中，用户首先对文件 cookies 和 cakes 做索引，为它们创建一个索引名为 recipes。从 recipes 中查询将搜索 cookies 和 cakes 两个文件。下面的例子中,用户将对文件 pies 做索引并把它加到索引 recipes 中。WAIS recipes 数据库中现在包含文件 cookies、cakes 和 pies。然后对目录 snack，包括它的所有文件和它的子目录中的所有文件，做索引，索引名为 junkfood。最后一个例子中，又一次对 sancks 做索引，但这一次索引将被加到 recipes 数据库中。

junkfood 数据库仍然存在并引用 snacks 目录。

```
#waisindex -d recipes -export cookies cakes
#waisindex -d recipes -export -a pies
#waisindex -d junkfood -export  -r snacks
#waisindex -d recipes -export -a -r snacks
```

用-t 选项，您将能对不同类型的文件做索引。您能索引图片、邮箱文件、HTML 页和标准的文本文件。不同的文本的类型列在表 12-14 中。对于文本文档，您能进一步指定对每行做索引。如果您按行索引，当用关键字查找时，WAIS 将指出关键字所在的行。下面的例子中，使用者对文档 breads 的每行做索引，产生索引文件叫做 cereals。

```
#waisindex -d cereals -t one_line breads
```

waisindex 命令还能把不同类型的文件关联到一个文件中。例如，如果您有一些图象、视频或声音文件，想把它们和一个指定的文本文档相关联，您可以用 waisindex 把这些文件连接在一起。当一个用户检索一个文本文档时，同时将检索关联的图象、视频或声音文件。当用户阅读文本时，它也能显示一个图片或播放声音。关联文件必须和它们所关联的文件有相同的前缀。例如，如果有一个文档名为 train.txt，有一个图形文件 trani.gif 和声音文件 train.midi。用-M 选项和一列文件类型，加上-export 选项把一组文件连接到一个索引文件。

```
#waisindex -d train -M text, tiff, mpeg, midi –export
/user/waisdata/train/*
```

为了让 WAIS 和您的 Web 资源很好地结合，您要为您的 Web 页创建 WAIS 索引。用 waisindex 和-T HTML 选项，用它来指定被索引的文件类型是 HTML 文档。索引名可以是类似 myweb 之类的名字。此索引将允许 WAIS 搜索 Web

HTML 文档。在下面的例子中，使用者将对目录/home/httpd/html 中的 Web 页做索引。索引名为 myweb，类型是 HTML。通过指定-contents 选项对每个 Web 页的全部内容做索引。选项-export 将包括 Internet 访问的主机信息。

```
#waisindex -d myweb -T HTML -contents  -export -r
/home/httpd/html/*.html
```

### 12.7.3 您的 WAIS 资源

当您为文件做索引以便数据库可用时，waisindex 将为数据库创建一个源文件，通过它别的用户能访问您的数据库；它将提供一些信息，象数据库的名称等。一些 WAIS 数据库访问时要收费，用源文件能显示收费信息。您还能从中找到管理员的地址以发送一些建议。在源文件的结束是 WAIS 数据库的一个简短描述。

当用命令 waisindex 创建数据库时，如果您指定-export 选项，为了使别的系统中的用户访问您的数据库,将加入一些字段。其中有两个字段是关于 Internet 信息的，一个是 IP 名，另一个是您的主机的 IP 地址。被加入的另一个字段是用来指定访问您的计算机上的 WAIS 数据库的端口。如果您没有用-export 选项，这些字段将是空的。这种情况下，只有您的系统中的用户才能访问数据库。

如果需要，您能改变源文件中的任何字段。完整的源文件是用括号括起来的。每行中的字段是以一个开始的冒号加上字段名。您能编辑源文件并加入更多的描述。注意描述是用双引号括起来的。第一个引号是在“ description ”的后面，然后是描述的文本，最后是结束的引号，它单独占一行。下面的例子是一个 zipcodes 的源文件，名为 zipcodes.src。

```
(:source
:versiton 3
:ip-address "192.31.181.1"
:ip-name "quake.think.com"
:tcp-port 210
:database-name "/proj/wais/db/sources/zipcodes"
:cost 0.00
:cost-unit  :free
:maintainer  "wais@quake.think.com"
:descriptin "
WAIS index of USA Zip Code database.
The full Zipcodes file may be obtained via FTP using the URL:
<ftp://obi.std.com/obi/ZIPCODES/zipcode.txt>
"
)
```

别的用户用源文件来访问 WAIS 数据库。一个源文件将告诉用户它位于哪个主机以及它调用的是什么。您可以把它看做是 WAIS 数据库的 URL。远程的用户为了访问数据库必须首先拥有源文件。您可以把源文件发送给用户，他可以把源文件插入他的主机的 wais-sources 目录中，或者把源文件登记注册到一个象 quake.think.com 和 cnidr.org 的 WAIS 服务器中，这些服务器中有一个服务器的目录。您的源文件和其他的源文件都放在这些服务器中。用一个 WAIS 客户程序象 swais，使用者能访问服务器的此目录并找到列在其中的 WAIS 数据库。接着它们可以选择和查询您的数据库。

当您创建一个数据库时，在对文件做索引时用-register 选项就可以注册数

据库。您也可以等到以后再注册，可能您需要先测试它。注册时只须用命令 waisindex 和选项 -d 加索引名，然后再加上 -register 选项。

```
waisindex -d recipes -register
```

### 12.7.4 测试您的 WAIS 服务器

在 freeWAIS-0.5 源代码包中，有一个目录名为 test-wais，此目录中含有一组测试文件，用它们您可以测试索引或服务器的访问。在此目录中有一个 shell 脚本为 test.waisindex。如果您查看此 shell 脚本，您将看到几个 waisindex 命令，用测试文件创建几个不同的数据库。它将创建四个测试索引。索引 test-Bool 测试布尔查找能力。test-Comp 索引测试压缩文件的操作。test-Docs 索引对使用目录 docs 中的文档进行递归查找进行测试。test-Multi 检查对不同类型的文件象 GIF 图形文件的操作（这些命令名前都有一个路径 ../bin/，如果您已经下载 WAIS 二进制文件并已进行安装，应该删除前面的 ../bin/）。

您还可以注意到为每个测试数据库都创建一个源文件：一个 boolean.src 和 doc.src。为了在本地测试您的服务器，把这些源文件复制到您的 WAIS 客户程序用来访问 WAIS 资源的目录。例如，如果 swais 用 /usr/lib/wais-sources 做为它的源目录，然后复制 test-boolean.src（源）文件到此目录中。现在或者通过 inetd，或直接启动您的 WAIS 服务器（如果您直接启动，在命令的最后加上一个 & 号）。用 test-wais 目录的路径名作为 WAIS 数据目录。当您启动 swais，它将列出在它的源目录中所有资源，包括 test-boolean 资源和其它测试资源。您能选择和查询您的测试资源，它将访问您的 WAIS 服务器并返回结果。可以试着查找关键字“ boolean ”。也能把目录 wais-test 移到您的 WAIS 数据目录

中。为了创建一个总体的数据库，您可以对 wais-test 用 -r 选项做索引。

### 12.7.5 启动 freeWAIS

您可以让 WAIS 不间断地运行或当需要时通过 inetd 调用它。用 waisserver 命令和一些选项能启动 WAIS 服务器。您能用 -d 选项指定 WAIS 索引的缺省位置。还能用 -p 设置端口，用 -u 设置用户名。在命令的最后加上一个 & 号。当直接调用时，waisserver 将在后台运行。为了使 WAIS 以一个持续运行的守护进程运行，您应在一个 rc.d/init 启动文件象 wais.init 中输入 waisserver 命令。下面的例子是一个 waisserver 命令。

```
waisserver -d /usr/wais/wais_index &
```

为了更安全地运行您的 WAIS 服务，以一个普通用户而不是以 root 来运行 waisserver。新建一个用户并在它的 passwd 项中输入一个 *。然后启动 waisserver 时能用 -u 选项加上用户名。在下面的例子中，waisserver 以用户 sports 运行。

```
waisserver -u sports -d /usr/wais/wais_index &
```

为了使 inetd 启动 WAIS 服务器，您必须在文件 /etc/services 和 /etc/inetd.conf 中放置合适的项。对于文件 /etc/services，您可以放置如下的项。

```
z3590210/tcp#z39_50 protocol for WAIS
```

接着在文件 inetd.conf 中，放置调用 WAIS 服务器的项。当 waisserver 被 waisserver.d 调用，它将知道是通过 inetd 来运行。由于这个原因，参数列中的第一个参数是 waisserver.d。

```
z3590 stream tcp nowait root /usr/sbin/waisserver
waisserver.d /home/wais -e server.log
```

## 12.8 总结：Internet 服务器

您能设置您的 Linux 系统做为一个服务器以提供大量的 Internet 服务。您需要的是合适的服务器软件和安全地组织的目录。设置您的 Linux 系统做为一个 ftp 服务器，一个 Web 服务器，一个 Gopher 服务器或 WAIS 服务器的服务器软件是免费的。OpenLinux 在安装系统时自动地安装 Web 和 ftp 服务器软件。在系统的桌面上，您可以操作 Linux 系统，使它做为一个 Web 站点或一个 ftp 站点。

同时，还可以用不同的 Internet 服务器运行。它们以守护进程运行，等待远程用户的请求。当收到对某一服务的请求时，相应的服务器处理此请求。一个远程的用户能连接到您的 ftp 服务器并下载文件，同时另一个用户可以连接到您的 Web 服务器并浏览您的 Web 页。依照服务器的运行频率，您可以让它不间断地运行或只有当需要时用 inetd 守护进程调用它。为了使一个服务器不间断地运行，您只用简单调用它的服务器程序。用 inetd 调用服务器时，您必须在文件/etc/services 和/etc/inetd.conf 中放置合适的项，然后运行 inetd 守护进程。

Internet 服务器软件可以从不同的 Linux 站点上免费下载。参见第 1 章中关于 ftp 站点。在站点 sunsite.unc.edu 和它的镜像站点中，大多数的服务器软件位于目录/pur/linux/systems/info-systems 中。以后可以从这些站点中下载更新的软件版本。Caldera Network Desktop 当前已安装 Apache Web 服务器和 wa-ftpd ftp 服务器。

# 第 13 章远 程 访 问

Linux 提供远程访问其他 Linux 或 Unix 系统的能力。您能在远程系统上复制文件或执行 Linux 命令，也可以远程注册到系统中的一个帐号中。您可以不是在像 ftp 或 Gopher 那样在一个特定的界面上操作，而是在自己的 shell 中访问远程的命令，并将在远程系统中执行操作。

远程访问的命令通过网络连接进行操作。Unix 和 Linux 系统之间有两种类型的连接，每种连接使用不同的网络协议。用于 Internet 的 TCP/IP 协议（参见第 10 章）也可以用在局域网中。使用 TCP/IP 的网络一般都是专用连接，象以太网连接。UUCP 协议能提供另一种 Linux 和 Unix 系统之间的网络交换。UUCP 是一种早期出现的协议，它是为两个没连在网络中的系统之间操作而设计的。用 UUCP，一个系统能在预定的时间通过电话线连接到另一个系统，并立刻发送一组交换信息。通过 UUCP，可以很方便地直接连接到一个特定的系统、在两个系统间传输数据并切断连接。UUCP 允许您通过调制解调器连接并和另一个系统交换信息。

TCP/IP 和 UUCP 有各自的远程访问命令集，从命令集能反映出每个协议的优缺点。TCP/IP 的远程访问命令被做为远程或简单的 r 命令。常用的命令前加上一个 r 表示它的操作是远程的。例如，rcp 是一个命令，它可以远程的复制文件从一个系统到另一个系统。r 命令的优点是执行实时的操作。对于您能访问的网络中的系统，可以复制文件和执行命令，这些操作马上可以执行。用 r 命令能很容易地从一个系统复制一个目录到另一个系统。您能访问支持 TCP/IP 的

连接，象以太网、CSLIP 或 PPP，中的系统。

用 UUCP，您能通过普通的电话线拨号到另一个允许您访问的系统。UUCP 的操作是批量的。一个系统中用户提交他们的请求，象在远程系统中复制文件或执行命令。这些请求汇集在一起，等到一旦建立连接，马上把他们发送出去。远程系统收到请求，执行它们，接着建立另一个连接到您的系统并送回响应。一些请求可能是从远程系统复制文件到您的系统，当远程系统响应时，这些文件将发送到您的系统。由此可见，用 UUCP 的远程的操作将是非常费时。用户要等待系统发送请求，并等待远程系统响应。

## 13.1TCP/IP 远程访问操作：rwho，rlogin，rcp 和 rsh

TCP/IP 网络通信软件包使用远程访问的命令，这些命令首先是由 UC Berkely 为 Arpanet 开发的。它允许您远程注册到另一个系统中，并从一个系统复制文件到另一个系统。您能取得关于另一个系统的信息，比如当前谁正在注册使用。当调用一个系统的地址时，这些远程命令使用域名或 IP 地址。和 TCP/IP 远程访问命令一样，域名地址开始也是为在 Arpanet 上使用而设计的。TCP/IP 远程访问命令列在表 13-1 中。

许多 TCP/IP 命令可以和用在 Internet 上的网络通信功能相比较。例如，用 TCP/IP 命令 rlogin 可以远程注册到一个系统，它和 telnet 相似。rcp 命令能远程复制文件，它执行和 ftp 相同的功能。TCP/IP 命令的不同之处是它们提供给用户的易用和易控制性。您能很容易地访问在不同的 Unix 或 Linux 系统中的帐

号，并且能控制访问这些帐号但没有提供口令的用户。事实上，您能提供给不同的用户提供给关于您的帐号的一种组权限。

### 13.1.1TCP/IP 网络系统信息：rwho、uptime 和 ping

这些命令是一些 TCP/IP 命令，通过它们，您能从网络中的不同系统上取得信息。您能找到谁正在注册，得到另一个系统中用户的信息，或查询一个系统是否正在运行。例如，rwho 命令和 who 命令的功能很相似。它显示网络中的每个系统的当前注册的用户

```
$rwho
violetrobert:tty1Sept 10 10:34
garnetchris:tty2          Sept 10 09:22
```

命令 ruptime 可以显示网络中的每个系统的信息。此信息能显示出每个系统是如何执行。ruptime 显示系统是否运行，它运行了多久，系统中的用户数和系统在最后 5、10 和 15 分钟内的系统负荷。

```
$ruptime
violetup11+04:10,8 users,load 1.201.10
garnetup11+04:10,20 users, load 1.501.30
```

命令 ping 能检测出系统是否启动和运行。ping 命令加上您想检测的系统名做为参数。

下面的例子将检测 violet 是否启动并连接在网络中。

```
$ping violet
violet is alive
$
```

如果您想检测的系统已经关机，将得到一个如下的响应。这种情况下，garnet是关闭并没有连接到网络中。

```
$ping garnet
no answer from garnet
$
```

### 13.1.2 远程访问权限：.rhosts

您能用.rhosts 文件控制使用 TCP/IP 命令对您帐号的访问。用户能用标准的编辑器象 Vi 来创建他们帐号中的.rhosts 文件。它必须位于用户的主目录。下面的例子中。使用者显示文件.rhosts 文件的内容。

```
$ cat .rhosts
garnet chris
ciolet robert
```

使用.rhosts 文件是一种允许用户不提供口令而访问您的系统的简单方法。如果需要禁止此用户访问，只须简单地从文件.rhost 中删除系统名和用户注册名。如果一个用户的注册名和系统名在文件.rhost 中，那么此用户即可不提供口令而直接访问系统。并不是所有的远程注册操作都需要这种访问形式（您能用输入口令的方式来替代）；但一些远程命令要求有.rhosts 文件，象远程复制文件或远程执行 Linux 命令。如果您想在远程系统的帐号中执行这些命令，此帐号的.rhosts 文件中必须有您的注册名和系统名。

通过.rhosts 对某一系统进行访问时，也允许您使用 TCP/IP 命令直接访问此系统中您的其他帐号。您不需要先注册到这些帐号中。可以把系统中您的其他帐号做为当前注册帐号的扩展。不管文件处于您的哪个帐号下，都可以用 rcp

命令从一个目录复制到另一个目录。用命令 rsh，可以在您的其他帐号中执行任何 Linux 命令。

### 13.1.3 远程注册：rlogin

您可能在网络中的不同系统上都有自己的帐号，或者可以访问别人在另一个系统上的帐号。要访问别的系统中的帐号，首先要注册到您的系统中，接着通过网络远程注册到帐号所在的系统中。用命令 rlogin 可以远程注册到别的系统。命令的参数应是一个系统名。命令将把您连接到另一个系统中并开始注册的过程。

用 rlogin 的注册过程和一般的注册过程有所不同，用 rlogin 时用户不被提示输入注册名。rlogin 假设您的本地系统中的注册名和远程系统中的一致。所以象上面执行 rlogin 命令时，您将马上被提示输入口令。输入口令后，您即可进入远程系统中的帐号。

rlogin 假设注册名是相同的，因为大多数的人用 rlogin 访问别的系统中的注册名一般和本地的注册名是相同的。然而，当远程系统中的注册名和本地系统的不同时，选项 -l 允许您输入远程系统帐户的不同的注册名。语法如下所示：

```
$rlogin system-name  -l  login-name
```

在下面的例子中，用户使用注册名 robert 注册到 violet 的系统中。

```
$rlogin violet -l robert
password
$
```

一旦注册到远程系统中，您能执行任何命令。可以用 exit、CTRL-d 或 logout

（TCSH 或 C-shell）结束连接。

### 13.1.4 远程复制文件：rcp

您能用命令 rcp 从远程系统复制文件到本地系统中。rcp 执行文件传输的功能，它的操作和 cp 命令很相似，但它是通过网络连接到另一系统。执行命令 rcp 时要求远程系统的.rhosts 文件中有您的本地系统名和注册名。命令 rcp 用关键字 rcp 开头，参数为源文件名和复制的目标文件名。为了指定文件在远程系统中，您需要在文件名前放置一个系统名，两者之间用冒号分隔，如下所示：

```
$rcp system-name:source-file  system-name:copy-file
```

当复制一个文件到远程系统中时，复制的目标文件是远程文件，它要求带有系统名。而源文件在您的本机系统中，不要求系统名：

```
$rcp source-file  remote-system-name:copy-file
```

在下面的例子中，用户从自己的系统中复制文件 weather 到远程系统 violet 并重命名为 monday。

```
$rcp weaher  violet:Monday
```

当从远程系统中复制一个文件到本地时，源文件是远程文件，它要求带有系统名。而复制的目标文件在您的本机系统中，不要求系统名：

```
$rcp remote-system-name:source-file  copy-file
```

在下面的例子中，用户从远程系统 violet 复制文件 wednesday 到自己的系统中并重命名为 today。

```
$rcp wiolet:wednesday  today
```

您能用 rcp 命令在远程系统和本地系统之间复制整个目录。rcp 命令加上-r

选项将从一个系统复制一个目录和它的子目录到另一个系统。象 cp 命令一样，rcp 要求一个源目录和复制目录。在远程系统中的目录要求系统名和一个以分隔系统名和目录名的冒号，以及目录名。当从您的系统复制目录到一个远程系统，则在远程系统中的复制目录需要远程系统名

```
$rcp -r source-directory  remote-system-name:copy-directory
```

在下面的例子中，使用者把目录 letters 复制到远程系统 violet 中的目录 oldnotes 中。

```
$rcp -r letters  violet:oldnotes
```

当从您的系统复制一个远程系统中的目录到本地时，在远程系统中的源目录需要远程系统名。

```
$rcp -r remote-system-name:source-directory  copy-directory
```

在下面的例子中，使用者把远程系统 violet 中的目录 birthdays 复制到本地的目录 party 中。

```
$rcp -r violet:birthdays  party
```

同时，您可以用星号指定名字，或用圆点引用当前目录。对于 Shell 的特殊字符，是由您的本地系统进行解释转换，而不是远程系统。为了使远程系统解释转换一个特定字符，您必须通过某种方式引用它。为了复制远程系统种所有带扩展名.c 的文件到您的系统中，您需要用特殊字符-星号：*.c 来指定所有的带扩展名.c 的文件。您必须注意引用星号的方式。下面的例子中，在系统 violet 中的带.c 扩展名的文件被复制到使用者的系统中。注意，星号是通过一个反斜杠引用的。而最后的圆点，表示当前的目录，并没被引用。它是由您的本地系统解释并转换的。

```
$rcp  violet:＼*.c  .
```

下面的例子中，目录 report 将从使用者的本地系统复制到远程系统的当前目录中。注意圆点被引用，它将被远程系统解释转换。

```
$rcp -r reports  violet:＼.
```

### 13.1.5 远程执行： rsh

您可能需要在远程系统中执行一个命令。rsh 命令将在远程 Linux 系统上执行一个命令并把结果显示到您的系统中。当然，您的系统名和注册名必须在远程系统的.rhosts 文件中。命令 rsh 有两个一般的参数，一个系统名和一个 Linux 命令。语法如下所示：

```
$rsh  remote-system-name  Linux-command
```

在下面的例子中，rsh 命令将在远程系统 violet 中执行一个 ls 命令以列出在 violet 中目录/home/robert 中的文件。

```
$rsh violet la /home/robert
```

除非是引用特定字符，否则它将被本地系统解释转换。对于控制标准输出的特殊字符更是如此，象重定向或管道字符。下面的例子中列出远程系统上的文件，并把它们送到本系统中的标准输出。重定向操作符由本地系统解释，并把输出改向到本地系统中的文件 myfiles 中。

```
$rsh violet ls /home/robert > myfiles
```

如果您引用一个特定字符，它将成为 Linux 命令的一部分被远程系统解释。引用重定向操作符将允许您在远程系统中执行重定向操作。下面的例子中，引用一个重定向操作符。它变成 Linux 命令的一部分，包括命令的参数，文件名 myfile。命令 ls 产生一列文件名并把它们重定向到远程系统中的一个文件 myfile

中。

```
$rsh violet ls /home/robert '>' myfiles
```

对于管道操作也是如此。下面例子中第一个命令输出一列文件到本地的打印机中。标准的输出通过管道输出到您的在线打印机中。第二个命令中，一列文件将输出远程系统的打印机上。管道线被远程系统解释，输送标准输出到远程系统的打印机中。

```
$rsh violet ls /home/robert | lpr
$rsh violet ls /home/robert '|' lpt
```

## 13.2 从 Unix 到 Unix 的复制：uucp

您可以使用 UUCP 的一组远程命令来执行在其他系统上的操作。例如，uucp 命令能从一个系统复制文件到另一个系统。正如您能访问本地系统中的文件一样，您可以访问其他系统中的文件。然而，UUCP 命令和您本地的命令一样受到权限的限制。被保护的文件和目录不能被访问。只有权限设置为 other 的文件和目录才可以访问。

您可以把 UUCP 命令看作是通过邮件系统引用其他系统中的文件。这些命令被设计使用点到点的通信来操作。就像使用不同系统之间的邮件能力来实现网络连接。当您对一个系统执行一个 UUCP 命令时，则系统会收集对同一系统的所有命令并排序。接着把命令发送到目标系统中去执行。一旦系统收到命令就开始执行它们，并把结果发送回来。一些系统之间可以进行接收和发送命令，并组成一个 UUCP 网络。整个过程依靠网络中的每个系统发送命令到另一个系

统和从另外的系统中接收命令。从这方面来讲，只需最简单的连接即可实现强大的网络功能。另外，它不要求特殊的结构，只发送和接收必要的信息。

有四个主要的 UUCP 命令：uuto，uupick，uucp 和 uux。uuto 命令发送文件到另外的系统中，uupick 命令从另外的系统中接收邮件。这两个命令用于发送和接收大文件。uucp 命令从一个系统中复制文件到另一个系统。许多 UUCP 命令对应于 TCP/IP 的远程访问命令。uucp 的操作象 rcp，而 uux 更像 rsh。

### 13.2.1 安装和配置 UUCP

UUCP 软件包不能自动安装到您的 Linux 系统中。对于 Caldera Network Desktop 您需要首先要安装您的 Caldera CD-ROM，接着或是从根用户的桌面上使用 glint命令，或是用 rpm 命令来安装 UUCP 软件包。另外，您可以从 Linux ftp 站点上下载此软件。用 gunzip 解压并用 tar 安装这些文件。有不同版本的 Linux UUCP 可以下载。不同的版本有不同的配置，所以要仔细阅读任何安装信息。两个流行的软件是 Taylor UUCP 和 HDP UUCP。这儿介绍的是 Taylor UUCP。

安装完成后，您必须配置 UUCP。这将是一个非常复杂的过程。仔细阅读 UUCP 的文档，象 HOW-TO 文档，甚至是关于主题的文本。在目录/usr/lib/uucp 中有一个包含配置文件的简单配置。此目录下有三个配置文件：Permissions，Devices 和 Systems。Permissions 文件列出能访问您的系统的不同的系统和访问的类型，还包括您能访问的系统。Devices 文件列出您能用于 UUCP 通信的调制解调器和初始化的信息，象调制解调器的速度。Systems 系统列出您能访问的不同 Linux 或 Unix 系统的拨号和注册信息，包括电话号码、注册名和口令。

在文件 Permissions 中，列出您想与之交互的系统和允许它具有的权限。

同样，此系统也应有相同的项列在它的 Permissions 文件中以允许您访问。通过给变量赋值来设定权限。对于每个系统都有一组变量，它们在同一行中输入（如果您想在新的一行中输入它们，可以在一个“＼”来扩展此新行）。下面列出这些变量。

MACHINE　　您希望访问的和允许访问您的系统的远程系统
MACHINE=rose

LOGNAME　　当远程系统访问本地系统时，此注册名的权限将发挥作用
LOGNIME=uucp

COMMANDS　　远程系统能在您的系统中执行的命令
COMMANDS=uucp：uux

READ　　存储传输的目录 spool 将被发送到远程系统
READ=/usr/spool/uucppublic

WRITE　　从远程系统收到存储传输的目录 spool
WRITE=/usr/spool/uucppublic

SENDFILES　　指定您是否能发送文件到远程系统
SENDFILES=yes

REQUEST　　指定远程系统是否能访问您的系统中的文件
SENDFILES=no

下面显示的是一个简单的 Permissions 文件。调用的远程系统是 rose。当访问 rose 时，使用的注册名是 uucp。rose 能在本地系统中执行的命令是 uucp 和 uux。从本地系统到 rose 的传输包含在目录/usr/spool/uucppublic 中。从 rose 到本地系统的传输也被包含在/usr/spool/ucppublic 中。本地系统能发送文件到 rose，当 rose 不能从本地请求文件。

/usr/lib/uucp/Permissions
LOGNAME=uucp MACHINE=rose\
READ=/usr/tmp:/usr/spool/uucp/uucppublic\
WRITE=/usr/tmp:/usr/spool/uucp/uucppublic\
SENDFILES=yes REQUEST=yes\
COMMANDS=rmail:rnews:uucp

在文件 Systems 中，每个系统的拨号和注册信息在不同的行中。每行的开头是远程系统名。您能指定使用的调制解调器。也可以输入 Any，让 UUCP 来选择可用的调制解调器。您能指定调制解调器的类型，通常是 ACU，它代表自动呼叫设备。接着您可以输入调制解调器的速度，您能指定一个范围，然后是电话号码。接着指定 login 提示的最后一些字符，象 ogin：和注册名。同样输入 password 的最后一些字符，象 word：和口令。下面是/usr/lib/uucp/Systems 文件的一个简单项：

rose Any ACU 19200 5555555 "" \r ogin: richlp word: mypass

在 Devices 文件中列出了您的调制解调器的类型、使用的端口、速度和一个驱动文件。您可以为同一个调制解调器设定多个项，以指定不同的速度和驱动程序。调制解调器的类型通常为 ACU（自动呼叫设备），它是指能自己拨号的调制解调器的类型。现在所有的调制解调器均为此类型。下面列出 Devices 文件的一个简单项，端口是 cua4，第四个串口；速度是 38400。

ACU cua4 - 38400 dialfast

## 13.2.2UUCP 地址

一个 UUCP 网络通常使用寻址的路径形式，它反映 UUCP 的点对点通信方

式。本地系统能连接到另一个系统，此系统可以位于国内的不同地域，还可以连接到位于世界其它地方的系统中。所有的系统之间并没有直接连接而是一个系统连接到另一个，另一个再连接到其他系统，依次类推。通过中转连接系统发送信息，您能达到此网络中的远端系统。例如，如果 garnet 系统连接到 stan 系统，stan 系统又连接到 bell 系统，那么在 garnet 上的用户能和在 bell 系统中用户通信。然而，这种通信不是实时的。一个信息做为一组批量信息的一部分从一个系统发送到另一个系统，当它们到达目标地址时会有所耽搁。

在寻址的路径形式中，系统地址放在用户的注册名前，并用一个感叹号分隔。下面是地址的语法：

system!login-name

在下面的例子中，mailx 的功能是发送一个信息给名为 garnet 的 Linux 系统上的用户 chris。Chris 的地址用地址格式表示。

$mailx garnet!chris < mydata

在 C-shell 中，寻址的路径形式要求在感叹号前加上一个斜杠。因为感叹号在 C-shell 中表示一个历史命令。斜杠将对感叹号转义，只把它做为一个感叹号，而不是一个历史命令。C-shell 路径地址的语法如下所示，还有一个用在命令 mailx 中的 C-shell 地址，

system＼!login-name

>mailx garnet＼!chris < mydata

在寻址的路径形式中，另一个系统的用户地址中包含您到达目的系统所要经过的中转系统名。每个中转系统被分别写在用户系统名前并用感叹号逐个分隔开。如果您在 garnet 系统上，并想发送一个信息给 bell 系统中的用户 robert，

您要指定信息发送过程中要经历的中转系统。在下面的例子中，中转系统是stan，给出地址为 stan!bell!robert。可能有许多的中转系统，如果要发送信息给 aleina 上的 rost，您需要经过三个系统，在地址中应一一指出这些系统。下面的例子中，信息将最终被送到目的地。第一个命令，一个信息首先被送到 stan 系统，接着送到 robert 所在的 bell 系统。第二个命令中，信息首先发送到 lilac，它再把信息送到 sf，最后 sf 把信息送到 aleina 所在的 rose。

```
$mailx stan!bell!tobert < mydata
$mailx lilac!sf!rose!aleina < mydata
```

您经常可以指定连接系统的一些不同的中转路径。一个网络可以通过不同的方式连接。两个系统之间有一个最短的连接。找到并定位一个用户的正确连接的路径顺序将很复杂，但通过正确的连接也将很快达到目的地。下面两个命令显示两个到达同一系统的不同路径。第一个例子中，到达 rose 之前经过三个系统。第二个例子只经过一个系统，sf。

```
$mailx lilac!mac!violet!rose!aleina < mydata
$mailx sf!rose!aleina < mydata
```

### 13.2.3 连接的系统：uname

在一个 UUCP 网络中，您能连接到许多系统中。命令 uname 将列出用户能远程连接的，或在其上执行象 uucp 等远程命令的系统。在下面的例子中，命令 uname 列出所有连接的系统。

```
$uname
garnet
rose
```

```
lilac
$
```

uname 命令加上-l 选项将显示您的本地系统名。

```
$uname -l
violet
$
```

uname 命令产生一个系统名，并把它们送到标准输出。这列名字可能非常长，所以您可以把它们保存到一个文件中，或用打印机打印出来，而不是只在屏幕上显示。您能重定向这列系统名到一个文件中并保存，通过管道输送到打印机中以打印出来，或通过一个查找过滤器过滤并得到一个特定的系统名。在下面的例子中，第一个命令保存一列系统名到一个文件中，第二个命令打印此列系统名，最后一个命令用 grep 查看在此列中是否有一个指定的系统名。

```
$uname > syslist
$uname | lpr
$uname | grep garnet
garnet
$
```

### 13.2.4 创建 UUCP 连接：uucico 和 uuxqt

在 Linux 系统中，uucico 程序将处理所有的 UUCP 通信。uucico 代表 UUCP 调用调出（UUCP Call-In Call-Out）。它是一个等待任何 UUCP 传输的守护进程进程，并把传输保存到目录/usr/lib/uucp/uucppublic 中。一个跟踪的操作 uuxqt 将解释和执行在传输信息中指定的操作。命令 uucico 和 uuxqt 是系统管理员操

作，只能由根用户执行。

uucico 程序也把传输发送到其他系统。UUCP 请求被汇集在一起并通过 uucico 发送到它所在的 UUCP 网络中的下一个系统中。此传输将继续从一个系统发送到另一个系统，直到信息到达它的目的系统。

做为根用户，您能用 uucico 程序拨号到另一个等待连接的系统中。程序将建立连接，传输 uucp 命令请求，然后收到来自其他系统的回应和另外的 uucp 请求。使用 uucico 的语法如下所示：

```
uucico  -options  remote-name
```

两个有效的选项是-r 和-x。-r 选项能禁用自动重拨号的等待时间，-x 选项加上数字 9 能设定调试状态，使您能看到 uucico 所进行的动作。下面的例子中，根用户和 rose 系统之间建立一个连接。

```
$uucico  -r  -x 9 rose
```

### 13.2.5 邮件文件传输：uuto 和 uupick

UUCP 提供了一个邮件功能，用来发送大文件。可以用命令 uuto 发送文件，用命令 uupick 接收文件。这些命令的操作更像命令 mailx。uuto 命令在一种批处理的模式下操作。您的请求和系统的其他 uuto 请求被排在一个队列中。当您的请求到达队列的顶端，您的文件即被发送。如果在这期间改变了您的文件，发送的不是原始文件，而是改变后的文件。命令 uuto 有一个选项允许您来避免这种情况。选项-p 将马上复制您的文件到一个虚拟目录中，当发送时，把复制的文件发送出去。您能随意修改原始的文件。uuto 的选项-m 能通知您文件是何时被发送的。表 13-2 中列出 uuto 的各选项。uuto 命令的语法和选项如下所示：

```
$uuto filename address
$uuto -n -p filename address
```

在下面的例子中，文件 mydata 被送到 violet 上的地址 marylou 中。

```
$uuto mydata violet!marylou
```

您能用命令 uupick 来接收由 uuto 命令发送的文件。为了接收文件，首先在命令行中输入 uupick，并且不加参数。接收的发自其他系统 uuto 命令的文件将被顺序显示。首先提示您的是第一个接收的文件名。提示的末尾有一个问号，等待您的回应。可以在回应中指定您想如何处理接收到的文件。常见的回应是 m，它把文件移到您当前的目录中。为了移动文件到指定的目录中，您能在 m 后指定目录名。然后按 ENTER，您将被提示下一个接收的文件名。输入您的回应，按 ENTER，将提示下一个文件名。继续操作直到完成用 uuto 命令发送的整个文件队列。如果您只按 ENTER，而不输入回应，文件将保持未接收的状态，并在下次执行 uupick 命令时再次提示该文件。不同的 uupick 命令被列在表 13-2 中。

下例中，命令 uupick 提示用户接收到三个文件。第一个文件，mydata，被移到当前目录中,并显示文件块的大小。第二个文件,party,被移到目录 birthdays 中。对文件 project 没进行操作。下次使用者再执行 uupick 时将会再次提示它。

```
$uupick
from system violet: file mydata ? m
10 blocks
from system garnet: file party ? m /home/chris/birthdays
2 blocks
from system violet: file project ?
```

$

您可能想检查是否收到某个特定系统上的用户发送的文件。命令 uupick 的-s 选项加上一个系统名将只提示从此系统中发送来的文件。下例中，uupick 将只提示从 violet 接收来的文件。

$uupick -s violet

### 13.2.6 直接复制文件：uucp 和 uustat

尽管 uuto 命令从一个帐号发送文件到另一个帐号，命令 uucp 能直接从一个用户的目录中复制文件到另一个用户的目录。使用 uucp 时，您访问的系统中的不同的帐号被看成是不同的目录。象使用命令 cp 一样，uucp 命令可以带有两个选项：源文件名和目标文件名。

$uucp  source-file copy-file

您能用 uucp 命令把您的目录中的文件复制到另外系统中，或从另外的系统中复制文件到本地。某些情况下，复制的目标文件名或源文件名前应有系统名和文件的路径名。在下例中，文件 mydata 被复制到 violet 系统中的目录 george 中。

$uucp  mydata  violet!/home/george/mydata

uucp 命令的操作是一种后台的批处理模式。您的 uucp 命令被收入一个队列中，当它到达队列的顶部时，将开始复制您的文件。如果您在这期间改动了此文件，那么复制的文件将是改动后的版本。用选项-C 能解决此冲突。使用 uucp 命令时加上-C 选项，则要复制的文件首先被复制的系统的一个虚拟目录中。当实际进行复制工作时，系统将使用存放在虚拟目录中的文件。表 13-2 中列出 uucp

的不同选项。在下面的例子中，文件 mydata 将首先复制到虚拟目录中，然后对虚拟目录中文件进行 uucp 的操作。

```
$uucp -C mydata violet!/home/george/mydata
```

如果在命令 uucp 中，您指定的目标文件所在的目录在远程系统的文件系统中不存在，命令 uucp 将在创建此目录。然而，远程系统可能不希望您在他们的系统中创建目录。这种情况下，您能用-f 选项来指示 uucp 不创建不存在的目录。在下面的例子中，使用者复制文件到 george 的 home 目录下的子目录 birthday 中。在第一个 uucp 命令中，如果 birthday 目录不存在，命令将创建它。在第二个 uucp 命令中，目录 birthday 将不被创建。

```
$uucp  party violet!/home/george/birthday/party
$uucp -f party violet!/home/george/birthday/party
```

可能有时您只知道远程用户名，而不知道远程用户的从根开始的全路径名。事实上，在使用 uucp 命令时，您需要指定远程文件的全路径名，以使得 uucp 命令能找到此文件。当您不知道路径名时，可以用一个波浪线操作符来查找用户的全路径名。波浪线操作符，~，加上它的参数一个用户名，能被转换成用户的主目录的全路径名。例如~george 被转换为/home/george。您能用波浪线操作符和一个用户名来指定一个文件的全路径名。下面的例子中，波浪线操作符用来指定一个全路径名，第一个是为文件 mydata，第二个是为 party。第二个命令把 violet 系统中的文件 party 复制到使用者自己的目录中。

```
$uucp mydata violet!~george/mydata
$uucp violet!~george/party  party
```

用命令 uucp，能从一个远程系统中把文件复制到另一个远程系统。下面的例子便是把 violet 中的一个目录下的 mydata 复制到 garnet 的一个目录中。

```
$uucp violet!~george/mydata  garnet!~robert/mydata
```

记住 uucp 命令是以批处理模式执行的。用它执行任务可能需要一定的时间。命令 uustat 可以列出关于当前 uucp 操作的信息。用-u 选项，您能显示对指定用户的 uucp 操作。用-s 选项，能显示对指定系统的操作。下面的例子中显示系统 garnet 中用户 robert 的 uucp 的操作。

```
$uustat -urobert -sgarnet
```

您能用 uustat 来结束 uucp 操作。uustat 将列出每个执行中的 uucp 的工作号。加一个-k 选项和工作 ID 便能结束此 uucp 工作。

```
$uustat  -k  795
```

### 13.2.7 远程执行：uux

用命令 uux，您能对其他系统中文件执行远程命令。在一个 uux 命令中，通过文件和命令路径来引用文件和命令。例如，violet!~robert/filmdata 引用 violet 系统中 robert 主目录中的文件 filmdata。您的系统中的命令和文件通过一个前置的感叹号来引用，不用加系统名。!mydata 表示您自己系统中的文件 mydata。执行命令时也一样。为了执行您自己系统的命令，在命令前加一个感叹号。下面的例子中，robert 的主目录下的文件 filmdata 被显示在使用者自己的终端上。

```
$uux  !cat  violet!~robert/filmdata
```

在一个 uux 命令中，为了避免您的 shell 解释一些特殊字符，象重定向和管道（>，<，|），您需要对它们加引号。您可以只对这些符号加引号或对整个命令都加引号。在下面的例子中，命令 cat 复制文件并通过管道把它们输送到打印机。

$uux '!cat violet!~george/party garnet!~robert/food | lpr'

如果您想使用远程系统上的命令，需要在命令前加上它的路径名。例如，假设您想在远程系统的打印机上打印文件，而不是在自己的打印机上。您能在命令 lpr 前加上一个系统路径。在下面的例子中，用户在远程系统的打印机上打印文件 filmdata。

$uux "!cat violet!~robert/filmdata | violet!lpr"

象 uucp 命令一样，uux 命令不能被马上执行。它们被放在一个队列里，当某一命令到达队列头时才执行它。在此期间，您可以对操作的文件进行更改。被操作的文件将是更改后的文件。象 uucp 命令一样，用 uux 命令的-C 选项能避免这种情况。此选项将对操作的文件建立一个临时的副本，操作是对其副本进行。表 13-2 中列出 uux 的选项。

每个特定的 Linux 系统对 uux 带有的可在本系统上执行的命令都有一些限制。象 rm 这种删除文件的命令就不允许执行。在一个给定的系统中，能被 uux 执行的命令列在权限文件

/usr/lib/uucp/Permissions 中。

## 13.3 总结：远程访问

通过网络，您能使用 TCP/IP 或 UUCP 网络协议的远程命令访问远程系统。TCP/IP 的远程命令允许您注册到远程系统的帐号中。您能在其他系统中复制文件和执行 Linux 命令。然而，要想在远程系统中执行命令，远程系统必须提供被访问的权限。远程系统为了能给您提供访问，系统中必须有一个.rhosts 文件，

在文件中要列出您的系统名和注册名。

除了使用 TCP/IP 外，您还可以使用 UUCP 协议。它们两个的设计和使用很相似。UUCP 没有 TCP/IP 的功能强大，但它更易于操作。UUCP 使用路径形式进行寻址。一个系统可以通过许多中转系统连接到另一个系统中。UUCP 的远程命令可以发送文件、在本地系统和远程系统之间复制文件、远程执行命令和远程注册到其他系统中。

表 13-1TCP/IP 远程访问命令和它们的选项

| 命令 | 作用 |
|---|---|
| rwho | 显示网络中所有注册到系统的用户 |
| ruptime | 显示网络中每个系统的信息 |
| ping | 检测网络中系统是否正在运行 |
| 远程命令 | |
| rlogin system-name | 允许您远程注册到另一个系统的一个帐号中$ rlogin violet |
| -l | 允许您指定要注册的帐号注册名$rlogin violet -l robert |
| -x | 对所有传输的数据使用 DES 加密 |
| -d | 启动套接字调试 |
| -e | 为 rlogin 通话设置转义符；缺省的转义符是 ~ 符。 |
| -E | 使得所有的字符不被解释为转义符 |
| -8 | 允许 8 位数据地址，以使得允许传输特定代码 |

| | |
|---|---|
| -k realm | 为一个指定领域内的远程主机取得一个 Kerberos tckets，而不是为所有的远程主机 |
| rcp sys-name:file1 | 允许您从一个系统的帐号中复制文件到另一系统中；如果没有 |
| sys-name:file2 | 给定系统名，则表示是当前系统 $rcp mydata violet:newdata |
| -r | 用 -r 选项，允许您复制目录 $rcp -r newdocs violet:edition |
| -p | 保留源文件的更改时间和模式 |
| -d | 启动套接字调试 |
| -x | 对所有传输的数据使用 DES 加密 |
| -k realm | 为一个指定领域内的远程主机取得一个 Kerberos tickets，而不是为所有的远程主机 |

表 13-2UUCP 命令和选项

| 命令 | 作用/描述 |
|---|---|
| Uucico optionsremote-system | 拨号并连接到远程系统；它是一个系统管理员的操作，只能由根用户来执行 $ uucico -r -x rose |
| -r | 以主模式启动（调到一个系统中）；隐含选项 |
| -s system | 如果没有指定系统，可以调用任何处于等待的系统 |
| -r0 -s slave | 以从属模式启动 |

| | | |
|---|---|---|
| -f | | 忽略任何等待调用系统的请求 |
| -l,-p | | 用 login：提示注册名，用 password：提示输入口令 |
| -p port | | 指定调出或监听的端口 |
| -c | | 仅当有对某一系统的操作时，调用此系统 |
| -x type | | 启动调试类型；用数字 9 启动所有的类型。这些类型是：bnormal，chat，handshake，uucp-proto，proto，port，config，spooldir，execute，incoming，ougoing。调试的信息放在/usr/spool/uucp/Debug |
| uuxqt | | 被 uucico 调用来执行 uux 请求的程序 |
| uname | | 列出您连接的系统 |
| uuto | filename | 发送大文件到另一个系统的邮件命令$uuto mydata |
| address | | violet!aleina |
| -m | | 当文件发送后通知发送者 |
| -p | | 复制文件到一个虚拟目录中并发送此副本 |
| uupick | | 接收用 uuto 发送的邮件文件的邮件命令；您将被分别提示每个文件 |
| uupick | | |
| m dir | | 把接收到的文件移动到您的目录中 |
| a dir | | 把所有收到的文件移动到您的目录中 |
| d | | 删除收到的文件 |
| p | | 显示收到的文件 |
| ENTER | | 让文件保持等待 |

| | |
|---|---|
| Q | 退出 uupick |
| * | 列出 uupick 的命令 |
| !cmd | 执行一个 Linux 命令，转义到您的 shell |
| uucp sys-name !filenamesys-name !filename | 从一个系统中复制文件到另一个系统$uucp mydata violet!robert/newdata |
| -m | 当一个 uucp 工作完成时通知使用者 |
| -n user | 当执行一个 uucp 工作时，通知远程用户 |
| -C | 复制文件到虚拟目录中，并发送此副本 |
| -c | 不把文件复制到虚拟目录中（缺省） |
| -f | 不在目标系统中创建目录 |
| -g | 指定服务的级别（高，中，低） |
| uustat | 列出当前的 uucp 工作；用-k 选项和工作号，您可以删除此工作 |
| -a | 列出所有用户的所有工作 |
| -u user | 列出指定用户的所有工作 |
| -s system | 列出指定系统的所有工作 |
| -k jobid | 删除一个 uucp 工作 |
| -c | 一个工作在队列中的时间 |
| uux | 远程地访问另一系统中的命令；文件名和命令名前应加有一个感叹号 |
| -z | 当工作完成时通知用户 |

| | | |
|---|---|---|
| | -n | 禁止工作成功的通知 |
| | -C | 复制文件到虚拟目录中，并发送此副本 |
| | -c | 不把文件复制到虚拟目录中（缺省） |
| | -g | 指定服务的级别（高，中，低） |
| cu | | 远程注册到另一系统的一个帐号中（调用 Unix） |
| | -s | 指定波特率（传输速度），象 1200,2400,4800,9600,38400,等 |
| | -c | 选择使用的本地局域网 |
| | -l | 选择使用的通信线路 |
| | -e | 设置为偶校验 |
| | -o | 设置为奇校验 |
| | -h | 设置为半双工 |
| | -n | 提示电话号码而不用在命令行上输入 |
| | ~! | 临时转到本地系统中 |
| | exit | 结束本地系统的使用退回到远程系统中 |
| | ~% | 把一个命令转义到本地系统中 |
| | ~% take | 从远程系统中复制文件到本地系统 |
| remote-file | | |
| | ~% put | 从本地系统中复制文件到远程系统 |
| remote-file | | |
| ct | | 过一个自动响应的调制解调器从您系统连接到远程系统中，用远程终端的电话号码做为参数，还可以用别 |

| | |
|---|---|
| | 的选项来指定传输特征，象波特率和奇偶校验。$ ct 6427400 |
| -s | 波特率$ct -s 1200 6427400 |

# 第四部分 Shells

# 第 14 章过滤器与正则表达式

Unix 系统的一个非常突出的特性是其过滤器命令，而 Linux 系统也具有这一特性。过滤器是一些读取数据的命令，并能对这些读取的数据执行适当的操作，然后把操作的结果送到标准输出。这些过滤器根据不同的任务可输出不同的结果。一些过滤器输出一些输入信息，而有一些过滤器仅输出输入信息的部分内容，还有一些过滤器把输入的信息进行修改，尔后再进行必要的输出。但有些过滤器有不同的选项，根据不同的选项，你可以进行部分或全部输入信息的输出，或进行必要的修改后再输出。你可以认为过滤器就是读取数据流，并对读取的数据流进行必要的修改操作，最后输出这些修改后的数据流。

输入至过滤器的数据流可以是一系列的输入数据，包括从文件、设备及命令或其它过滤器命令的输出等。过滤器仅仅对输入的数据流进行操作，它并不修改原数据。如果过滤器从文件中接收数据，其原文件是不会被修改的，因为过滤器只是从文件中读取数据。

过滤器的输出经常被输出到标准输出，也可以被重定向到另外一个文件中或设备上，或者用管道输入给另外一个实用程序或过滤器。所有管道与重定向的特性同样适用于过滤器。通常过滤器读取数据，进行必要的修改，然后通过管道输入至另外一个过滤器。但是，请注意：输入的原文件数据是不会被修改的，这一点非常重要。

许多实用程序与过滤器使用模式（pattern）来在文件中定位、选择指定的文本。但有时候，你必须使用方式更灵活、功能更强大的模式来在文件中同时

搜索不同的字符或字符串。此时，你可以在模式中使用一组特殊的字符来完成这一方式灵活、功能强大的搜索。如果一个模式中包含一些特殊字符，这我们称该模式为正则表达式。大多数模式及实用程序都可以使用正则表达式来进行模式搜索，例如 egrp、grep、awk、Ed 及 sed 等。尽管在正则表达式中的许多特殊字符与在 Shell 下的特殊字符相同，但它们的使用方式、代表的意义有所不同。Shell 命令下的特殊字符主要用在文件名中，而正则表达式主要用来搜索文本。

本书把过滤器分为常见的三类：文件过滤器、编辑过滤器及数据过滤器。本章将主要介绍文件及编辑过滤器。大多数有关 Unix 系统的参考书上都介绍过数据过滤器。本章的附表部分详细列出了所有的过滤器，包括数据过滤器。本章将首先向你介绍怎样在过滤器中使用重定向及管道，并比较过滤器中使用重定向及管道产生的各种不同的输出，然后详细介绍编辑过滤器及正则表达式。

## 14.1 在过滤器中使用重定向及管道：cat，tee，head 及 tail 命令

过滤器可以把它们的输出送到标准输出上，其缺省的标准输出是显示屏。一个最简单的过滤器是输出文件的内容。本书已经向你介绍了 cat 及 tee 命令，你可能没意识到的是：cat 及 tee 命令就是过滤器实用命令。它们接收多行数据，并把它们输出至屏幕。cat 过滤器命令接收来自文件的输入，并把接收的数据拷贝到标准输出上，缺省的标准输出就是屏幕。过滤器 tee 命令接收输入的数据，

并把它们同时输出到标准输出及指定的文件。除了 cat 命令及 tee 命令以外，还有另外两个过滤器命令用来输出文件：head 及 tail 命令。head 过滤器命令输出文件的首部，而 tail 过滤器命令输出文件的尾部。

你可以保存过滤器的输出至文件或输出至打印机。你可以使用重定向或管道操作来达到上述目的。如果要把过滤器的输出保存到文件中，你可以使用重定向操作符“>”；如果你想把重定向的输出输出至打印机，你可以使用管道来把它输出至 lpr 实用程序（lpr 实用用来打印文件）。在下面的命令中，cat 命令用管道将其输出的结果输出到 lpr 命令，然后，lpr 命令将它在打印机上输出。

```
$ cat complist | lpr
```

另外一些用来显示文件的命令如 more 命令，看起来似乎是一个过滤器命令，而实际上不是。你必须注意过滤器命令与面向设备（device oriented）的实用命令之间的区别，例如 lpr 命令与 more 命令之间的区别。过滤器命令是把它们的输出结果输出到标准输出上，而面向（device oriented）的实用命令例如 lpr 命令，尽管可以从标准输入设备上接收数据的输入，但它们只能把数据输出到标准输出设备上，more 命令也是如此，只不过其输出设备是终端。这些面向设备（device oriented）的实用命令也可以接收来自过滤器命令的输入，但它们仅能将它们输出至设备。

所有的过滤器命令都可以接收来自标准输入的数据输入。事实上，通过管道，你也可以把过滤器的输出作为另一个过滤器的输入。此外，也有许多过滤器可以直接接收来自文件的输入，这些过滤器可以用文件名作为它们的参数，并直接从这些文件中读取数据。例如过滤器命令 cat 及过滤器命令 sort 就是如此，它们或者使用文件名参数来直接从文件中读取数据，或者从标准输入上读

取数据。

如果你在使用 cat 命令的时候没有指定文件名，那么，cat 命令将从标准输入上读取数据。此时，cat 命令将等待你从键盘上输入数据，并把你键入的数据作为其输入。你可以使用 CTRL-D 命令（文件结束字符）来结束你的本次标准输入。在下面的例子中，cat 命令从标准输入上读取数据（此时标准输入是键盘），并用重定向把标准输入输出到名为 mydata 的文件中去。

```
$ cat > mydata
Hello Marylou
 How are you
today
^D
$
```

cat 命令的一个更强的特性与功能是它能把几个文件的内容合并到输出流。你可以用管道把这一输出流输入到一个实用程序，甚至另一个过滤器，并允许实用程序或过滤器把这些文件合并后的内容作为一个数据流进行操作。例如，如果你想一次逐屏浏览多个文件的内容，可以首先用 cat 过滤器命令把多个文件合并，然后用管道把合并后的数据输入到 more 过滤器命令上，more 命令将从标准输入上接收这些数据的输入。对于下面的一组命令，cat 过滤器命令将把文件 preface 及 intro 的内容拷贝至一个输出中。第一个例子是用管道把合并后的输出输入到 more 命令，于是，你可以逐屏浏览合并后的文本。第二个例子是用管道把合并后的输出输入到打印机。第三个命令是用重定向把合并后的输出保存到名为 frontdata 的文件中。

```
$ cat preface intro | more
```

```
$ cat preface intro | lpr
$ cat preface intro > frontdata
```

过滤器可以接收来自管道的输入，也可以把它们的输出数据输入至管道。同时，你也可以用管道把过滤器或实用程序的输出数据输入到过滤器，而此过滤器的输出数据又可以用管道输出至另一过滤器。你甚至可以用一系列的过滤器命令把前一个过滤器的输出数据作为下一个过滤器的输入。例如，你可以用管道操作把 cat 过滤器的输出数据作为另一个过滤器的输入。必须注意，此时过滤器将通过管道操作来接收标准输入的输入，而不是从文件。例如，假设你想打印 cat 命令的输出，同时你又想把其输出保存到另一个文件，这时，你可以首先用 tee 过滤器命令将其输出保存到一个文件，然后用管道命令把其输出数据输出至 lpr 命令实用程序，以打印出该文件。tee 命令是一个过滤器命令，该命令把标准输入拷贝到一个文件中，同时把它这一标准输入输出至标准输出上。必须注意，通过管道操作，命令 tee 接收来至过滤器命令 cat 的输出。在下面的例子中，tee 过滤器命令首先把 cat 命令的输出保存到文件 frontcopy 中，然后通过管道把它输出到 lpr 实用程序上，从而打印出该文件。

```
$ cat preface intro | tee frontcopy | lpr
```

### 14.1.1 输出文件的头部与尾部命令：head 及 tail 命令

假设，如果你只想查看一个文件究竟保存的是什么内容。为此，你可以只查看文件的头几行，而不必浏览整个文件。过滤器命令 head 可以达到上述目的。head 命令可以只显示文件或标准输入的头几行。与许多其它过滤器命令一样，head 过滤器有各种选项，通过这些选项，你可以控制文件或标准输入的输

出。缺省情况下，head 过滤器仅显示文件的头十行。你可以通过选项（在破折号后面跟一个数字）在命令行上指定你将要显示的行数。在下面的例子中，首先用 cat 命令输出 preface 文件的所有内容，然后用 head 命令显示 preface 文件的头三行。

```
$ cat preface
a text file in Unix
consists of a stream of
characters. An editor can
be used to create such
text files, changing or
adding to the character
data in the file.

$ head -3 preface
a text file in Unix
consists of a stream of
characters. An editor can
```

同样，如果你只想查看文件的尾部，你可以使用另一个过滤器命令 tail。缺省情况下，tail 命令显示文件的最后十行。与在 head 过滤器命令一样，你可以在破折号后面指定你想显示的行数。在下面的例子中，preface 文件的最后三行被显示在屏幕上。

```
$ tail -3 preface
text files, changing or
adding to the character
```

data in the file.

过滤器命令 tail 首先读取数据，然后输出其“过滤”后的数据。对于 tail 命令是数据的最后几行。与 cat 命令一样，你可以通过管道操作把 tail 及 head 命令输出至打印机或者用重定向操作输出至另一文件。在下面的例子中，tail 过滤器通过管道操作把 preface 文件的最后五行输出至打印机。

```
$ tail -5 preface | lpr
```

## 14.2 输出类过滤器：wc，spell 及 sort

过滤器命令的输出可以是经过修改后的输入数据的拷贝，或者是输入数据的一部分，甚至是输入数据的一些简单信息。有些过滤器命令的输出可能是上述的一种，而有些过滤器命令可以通过选项来指定其输出。wc 命令、spell 命令及 sort 命令可以用来说明上述三种不同的输出。过滤器命令 wc 仅显示出文件的行数、单词数及字符数，过滤器命令 spell 仅选择那些有拼写错误的单词，然后把它们输出。过滤器命令 sort 将按顺序输出所有的输入数据。这三个过滤器命令及它们的常用选项在表 14-1 中列出。

### 14.2.1 计数命令：wc 命令

过滤器命令 wc 把它的输入数据看作数据流（这些数据通常是来自于读取的文件），然后该命令统计文件中的行数、单词数及字符数（包括行尾的换行字符），并把它统计的结果输出。在下面的例子中，wc 命令统计并输出了 preface

文件中的行数、单词数及字符数。

```
$ wc preface
6    27   142    preface
```

wc 命令有三个选项，这些选项可以用来输出任意一个指定的统计数字，如行号、单词数或字符数，其中，-w 选项仅统计文件中的单词数，-c 选项仅统计文件中的字符数，而-l选项仅统计文件中的行数。

### 14.2.2 拼写检查命令：spell 命令

过滤器命令 spell 将检查其输入数据中的拼写错误，并同时输出这些拼写出错的单词。

```
$ spell foodlistsp
soop
vegetebels
```

通过重定向操作，你可以把这些拼写出错的单词保存到指定的文件中去，而通过管道操作，你可以打印出这些出错的单词。在下面的例子中，用户把拼写出错的单词保存到名为 misspell 的文件中。

```
$ spell foodlistsp > misspell
```

记住，你可以通过管道操作把一个过滤器命令的输出数据输入到另一个过滤器命令。例如，如果你仅想知道一个文件中究竟出现了多少次拼写错误，你可以通过管道把过滤器命令 spell 的输出数据输入到过滤器命令 wc 来统计文件中的拼写错误。在下面的例子中，文件 foodlistsp 的拼写错误被输出，而该列出的拼写错误通过管道操作输入给过滤器命令 wc，最后，用带-w 选项的 wc 命令统计并输出出错的单词数。

```
$ spell foodlistsp | wc -w
2
```

### 14.2.3 文件排序命令：sort 命令

过滤器 sort 命令将对文件中的各行进行排序。sort 命令有许多非常实用的选项，这些选项最初是用来对数据库格式文件的内容进行各种排序操作的。实际上，sort 命令可以被认为是一个非常强大的数据管理工具，用来管理有类似于数据库文件记录的文件。本小节将介绍怎样使用 sort 命令把文件中的文本行按字母顺序进行排序。

过滤器命令 sort 将逐行对文件中的内容进行排序，如果两行的首字符相同，该命令将继续比较这两行的下一字符，如果还相同，将继续进行比较。在下面的例子中，过滤器命令 sort 将输出 foodlist 文件中文本行排序后的结果。请注意，在原文件的第二、三行上的第一个单词完全相同，该命令将从它们的第二个单词 vegetables 与 fruit 的首字符处继续进行比较。

```
$ cat foodlist
vegetable soup
fresh vegetables
fresh fruit
lowfat milk

$ sort   foodlist
fresh fruit
fresh vegetables
```

lowfat milk
vegetable soup

当然，你可以保存排序后的文件内容，或这把排序后的文件内容输出至打印机。在下面的例子中，用户把排序后的文件内容保存到名为 slist 的文件中。

```
$ sort foodlist > slist
```

过滤器命令也可以对标准输入进行操作。例如，如果你想把几个文件文本行合并，并对合并后的文本行进行排序，你可以首先用过滤器命令 cat 把多个文件合并，然后用管道操作把合并后的文本行输入给过滤器命令 sort，sort 命令将输出这些合并及排序后的文本行。在下面的例子中，文件 veglist 与文件 fruitlist 的文本行经过合并与排序后被保存到文件 clist 中。

```
$ cat veglist fruitlist | sort > clist
```

## 14.3 搜索文件命令：grep 命令与 fgrep 命令

过滤器命令 grep 与 fgrep 用来搜索与文件中的字符串相匹配的模式，然后它通知用户在什么文件中搜索到与指定的模式匹配的字符串，并打印出所有包含该字符串的文本行，在该文本行的最前面是该行所在的文件名。grep 命令一次只能搜索一个指定的模式，而 fgrep 命令却可以一次搜索多个模式。过滤器命令 grep 与 fgrep 的使用及其选项见附表 14-1。

过滤器命令 grep 与 fgrep 搜索与定位文件中特定的主题是非常有用。要搜索的模式可以被认为是一些关键词，你可以用它们来搜索文件中包含的这些关键词。对于 grep 命令，它的搜索功能比 fgrep 要强大，因为 grep 命令的搜索

模式可以是正则表达式，而 fgrep 却不能。有关正则表达式（在 Ed 行编辑器中常使用正则表达式），本章的后半部分将给予介绍。

### 14.3.1grep 命令

过滤器命令 grep 有两个参数，其中，第一个参数是要搜索的模式，而第二个参数是要搜索的文件名列表。其用法如下：在命令行的命令后键入搜索的模式，再键入要搜索的文件。其中，文件名列表中也可以使用特殊字符，如星号“*”等，以用来生成文件名列表。

```
$ grep pattern filenames-list
```

在下面的例子中，grep 命令搜索 preface 文件中包含模式“stream”的文本行。

```
$ grep stream preface
consists of  a  stream of
```

如果你想在搜索的模式中包含多个单词，你可以用单引号把要搜索的模式括起来，以用来表明搜索的模式是由包含空格的多个单词组成。否则，Shell 将把空格认为是命令行参数的定界符，而 grep 命令将把搜索模式中的单词解释为文件名列表中的一部分。在下面的例子中，grep 命令用来搜索模式“text file”。

```
$ gep 'text file' preface
A text file in Unix
text files, changing or
```

如果你在文件名列表中使用了多个文件，grep 命令将在匹配的文本行之前输出相应的文件名。在下面的例子中，grep 命令搜索文件 preface 及 intro 中的模式“data”，在输出每个匹配的文本行之前，列出了其对应的文件名。

```
$ grep data preface intro
preface: data in the file.
intro: new data
```

前面已经提到过，你可以在命令行上用 Shell 特殊字符来生成将要搜索的文件名列表。在下面的例子中，特殊字符星号“*”用来生成一个文件名列表，该列表包含当前命令下所有的文件。该命令将非常简单方便地搜索出当前目录下所有文件中包含与模式匹配的字符串。

```
$ grep data *
```

特殊字符在搜索一组指定的文件是非常有用。例如，如果你想搜索所有的 C 程序源文件中特定的模式，你可以用“*.c” 来指定文件名列表。假设你的 C 程序中包含一个不必要的循环语句（如 while 语句），并要找到这些语句，你可以用如下的命令来搜索并显示所有包含 while 语句的代码行：

```
$ grep while *.c
```

过滤器命令 grep 也有一组选项，这些选项可以改变其输出方式。例如，你可以在搜索到的文本行上加入行号，或者只输出文本行的行号，或者输出所有与搜索模式不匹配的文本行，或只简单地输出已搜索到指定模式的文件名。同时，你也可以指定 grep 命令忽略大小写。这些选项在附表 14-1 中列出。

### 14.3.2fgrep 命令

搜索文件命令 fgrep 的搜索速度要快于 grep 命令，且 fgrep 命令可以一次迅速地搜索多个模式。但是，与 grep 不同，fgrep 命令不能搜索正则表达式。此外，搜索文件命令 fgrep 不能使用特殊字符，只能搜索确定的模式。

你可以在命令行上键入搜索的模式，或者使用-f 选项来从文件中读取要搜索的模式。在命令行上键入搜索的模式时，每个模式必须用换行字符隔开（从文件中读取的搜索模式也是如此），且整个搜索模式列表必须用双引号括起来；此外，每个搜索模式本身还必须用反斜扛字符“＼”隔开。在下面的例子中，用户搜索 preface 文件中包含字符串“editor”及“create”的文本行。必须注意的是，字符串“editor”及“create”用换行字符隔开，且换行字符的前一字符是反斜扛字符“＼”。

```
$ fgrep "editor＼
create" perishables
characters. An editor can
be used to create such
```

用-f 选项，fgrep 命令将从文件中读取搜索模式列表。该文件中必须包含一个搜索模式列表，且每个搜索模式必须独占一行，fgrep 命令在进行搜索时将同时搜索这些模式。当你经常要搜索一组常见字符串时，fgrep 命令的该功能非常有用。在下面的例子中，用户把要搜索的模式放置在文件 mypats 中，然后，fgrep 命令从文件 mypats 中读取搜索的模式。

```
$ cat mypats
editor
create
$ fgrep -f  mypats  preface
```

## 14.4 过滤器编辑程序

与 Unix 系统一样，在 Linux 系统中，文本文件由一系列的文本行组成。因此，许多编辑器及过滤器可以逐行对文件中的文本行进行操作。Unix 系统下的第一个编辑器 Ed 就是一个行编辑器，它一次只能在一行文本上进行操作。另外一些编辑实用程序及过滤器命令对文本行的操作与 Ed 编辑器类似。事实上，Ed 编辑器与其它编辑器基本上采用了一组相同的核心行编辑命令。一些过滤器编辑命令如 sed 及 diff 等使用相同的行编辑命令来对过滤器输入进行编辑。

过滤器编辑命令用来对过滤器的输入进行编辑操作，这些过滤器的输入可以是读取于文件的数据或者是接收来自于标准输入的数据。与前面介绍的过滤器命令一样，过滤器编辑命令接收输入的数据，并对输入的数据进行各种操作，然后把修改后的输入数据输出。从这种意义上说，数据是被“过滤”了。一个过滤器编辑命令接收文本行作为其输入，并可以在文本行上执行行编辑命令，再输出修改后的文本。

有三个主要的过滤器编辑命令：tr 命令，用来转换字符；diff 命令，用来逐行比较两个文件的不同之处，并输出它们的不同之处；sed 命令，用来对输入的数据执行行编辑操作。作为一个过滤器命令，sed 命令不修改任何原文件，相反，它只产生原文件的一个修改版。过滤器 diff 命令以行编辑操作的形式提供编辑信息，它告诉用户需要执行哪些行编辑命令可以把第一个文件修改至与第二个文件完全相同。这些行编辑操作本身就告诉你比较的两个文件之间怎样不同。

### 14.4.1 流式编辑器：sed

过滤器命令 sed 对其输入的数据执行行编辑操作。其中，输入的数据可以从文件中读取，也可以接收来自于标准输入的数据。sed 命令名来自于“stream editor”的缩写。sed 命令有两个参数，一个是行编辑操作参数，另一个是文件名列表参数。通过行编辑命令参数，sed 命令将生成文件名列表中所指定的文件的相应修改版，然后把这些修改版输出至标准输出，而所有的原文件本身并不被修改。无论是否被修改过，原文件中的文本行都将被输出。从这种意义上说，sed 命令生成所有输入文件的一个完整拷贝，尽管这个拷贝被编辑、修改过。

```
$ sed "edit-command' file-list
```

set 编辑器有一组编辑命令，这组编辑命令与 Ed 行编辑器的编辑命令相同。sed 命令中的命令参数必须用单引号括起来，以防止编辑命令中的特殊字符被 Shell 所解释或分割。sed 编辑命令在附表 14-2 中列出。在下面的例子中，sed 命令生成了 preface 文件的一个修改版。其中，编辑命令“3 d”与行编辑命令完全相同，即删除文件的第三行。因此，sed 命令将显示出 preface 文件的修改版（删除了第三行文本）。

```
$ sed '3 d' preface
A text of file in Unix
consists of a stream of
be used to create such
text files, changing or
adding to the character
```

data in the file.

与 Ed 编辑器编辑命令一样，sed 编辑器的编辑命令也是单个字符。例如，字符 d 表示用来删除一文本行；a 命令用来在当前行之后插入文本；n 命令将使编辑器带行号输出文本行；i 命令用来在当前行之前插入文本；c 命令将用新的文本替换一行或多行；s 命令用来用指定的替换模式替换文本行上指定的字符串。所有这些编辑操作与对应的 Ed 编辑器中的编辑命令完全相同。

一个非常常见的行编辑操作是模式替换操作。与 Ed 编辑器中的替换命令一样，sed 命令的替换命令也是在命令 s 后键入搜索的模式（被替换的文本）与要替换文本。替换命令将用要替换的文本替换与搜索模式匹配的字符串。

```
$ sed 's /create/generate/' preface
A text file in Unix
consists of a stream of
be used to generate such
text files, changing or
adding to the character
data in the file.
```

sed 命令可以同时执行几个编辑命令。你可以在命令行上输入编辑命令，也可以把编辑命令放置在一个文件中，当执行 sed 命令时，系统将读取该文件。sed 命令的-e 选项可以使用户在命令行上输入多个编辑命令，且在每个编辑命令之前必须输入-e 选项。sed 命令的-f 选项可以让用户从一个文件中读取一组行编辑命令，其用法是：sed 命令后键入-f 选项，然后输入保存了一组行编辑命令的文件名，最后输入文件名列表。运行 sed 命令时，sed 命令将从-f 选项后指定的文件中读取行编辑命令。在下面的例子中，行编辑命令是从文件 myed

中读取的。

```
$ sed -f  myed preface
$ cat myed
ld
s/create/generate/g
```

尽管 sed 命令的许多编辑命令与 Ed 行编辑器中的编辑相同，但是，那些由多行组成的命令须从文件中读取。例如 a 命令将在当前行之后添加文本，此时，你添加的文本将不止一行。因此，你可以在文件中保存这些编辑命令：在文件的第一行上输入 a 命令，然后在余下的行上输入添加的文本，且在每行的行尾，你必须键入换行符。这样做可以防止 Shell 把换行特殊字符解释为 Linux 命令的结束。

sed 命令把接收的输入看作是流式数据，它可以从标准输入中读取这一流式数据，也可以从文件中读取这一流式数据。如果在 sed 命令的命令行上没有指定文件名，那么，它将从标准输入上读取数据。在输入流式数据的过程中，无论是从文件中读取数据还是从标准输入上读取数据，你都可以用行号来定位文本行。通过标准输入，并使用管道，sed 命令可以接收来自其它 Linux 命令的输出数据。同时，sed 命令还可以接收从键盘（标准输入设备）输入的数据。此外，数据还可以是来自于文件或设备的重定向数据。总之，sed 编辑器命令是在编辑流式数据，这就是 sed 命令的命令名由来。

你可以把 sed 命令看作是具有编辑功能的 cat 命令。与 cat 命令一样，sed 命令可以接收来自于标准输入的数据，也可以是来自于文件名列表中文件的内容。如果在 sed 命令的命令行上把文件名列表作为其输入，它将从这些文件中读取数据作为其输入，如果没有指定文件名，那么，sed 命令将从标准输入上

读取输入数据。与 cat 命令一样，sed 命令会把输入数据的拷贝输出到标准输出上，但是，与 cat 不同，sed 命令将对其拷贝进行编辑操作，这些编辑操作由命令行上的第一个参数指定。这些行编辑操作将对将要输出的数据进行相应操作，然后生成原输入的一个修改版，且其命令行上指定的原文件（包括文件名列表中的所有文件）将不作任何修改，只是标准输出中的内容被修改。标准输出还可以通过重定向操作输处到指定的文件中去，以创建一个新的文件（原文件的修改版）。在下面的例子中，sed 命令的修改输出通过重定向操作后，被保存到 pfile 文件中，即 pfile 文件就是 preface 文件的一个修改版。

```
$ sed '3 d' preface > pfile
```

### 14.4.3 文件差别与修改命令：diff 命令

过滤器命令 diff 用来比较两个文件，并输出那些内容不同的文本行。diff 命令不仅能显示两个文件是否相同，而且能详细显示比较的两个文件在哪些地方不同。过滤器命令 diff 还输出一些行编辑信息，这些信息可以帮助你怎样将第一个文件的内容修改到与第二个文件完全相同。对第一个文件将要进行的编辑修改信息详细说明了它怎样与第二个文件不同。从这种意义上说，diff 命令的输出信息实际上是编辑信息。

当你在编辑一个文件，且你想知道你所编辑的文件是怎样被修改的时候，diff 命令输出的编辑信息就会对你非常有帮助。你可以保存修改前的原文件，同时保存一个经过修改后的文件。此时，如果想查看修改后的文件是怎样被修改的，你可以使用 diff 命令来比较原文件及当前修改后的文件。

从两个文件的第一行开始，diff 命令将逐行比较第一个文件与第二个文件相

应行之间的差别。该命令将只输出两文件那些不相同的文本行，换句话说，将输出两个文件各自独有的行文本行。第一个文件中独有的行在被列出时将在每行之前加一个小于符号“<”，而第二个文件中独有的行在被列出时将在每行之前加一个大于符号“>”。两个文件中完全相同的行将不被列出。

```
$ diff file1 file2
file1-linenums edit-command file2-linenums
<Differing line in file1
>Differing line in file2
```

在每行之前，diff 输出编辑信息，这些信息表明你怎样修改第一个文件，就可以使第一个文件与第二个文件完全相同（第一个文件可能需要在文件的不同地方经过多个不同的编辑修改）。每个修改都有一个编辑提示，其后跟着两个文件中对应的不同文本行。如果要使比较的两个文件完全相同，第一个文件中有的行可能需要被删除，或者需要被添加，或者需要被修改。行修改提示以行号开头，其后跟着可能的三种编辑命令提示字符：或 a 字符，或 d 字符，或 c 字符。a 字符表明哪些文本行必须从第二个文件添加至第一个文件中。其意义与 Ed 行编辑器中的编辑命令 a 相同，即需添加文本行。在 a 编辑命令提示符之后，diff 命令将列出那些需要从文件 file2 中插入到文件 file1 中的文本行。

d 编辑命令提示符表明要使第一个文件与第二个文件相同，第一个文件中的哪些文本行需要被删除。其意义与 Ed 行编辑器中的编辑命令 d 相同，即需删除文本行。在 d 编辑命令提示符之后，diff 命令将列出那些需要从文件 file1 中删除的文本行。c 编辑命令提示符表明要使第一个文件与第二个文件相同，第一个文件中的哪些文本行需要被修改。其意义与 Ed 行编辑器中的编辑命令 c 相同，即需修改文本行。在 c 编辑命令提示符之后，diff 命令将列出两文件中的

文本行，以帮助用户要使第一个文件与第二个文件完全相同，第一个文件中的哪些文本将要被修改，且两文本行之间用破折线隔开。下面的例子列出了 diff 命令输出的行提示符句法。

f1-linenum a f2-line1, f2-line2Append lines from file2 to after f1-linenum in file1.> file2 lines

f1-line1, f1-line2 d f2-linenum　Delete the lines in file1.

< file1 lines

f1-line1 f1-line2 c f2-line1, f2-line2　Replace lines in file1 with lines in file2.

< file1 lines

---

> file2 lines

在编辑命令提示符的前后都有一个行号或一系列的行号（行号范围）。在编辑命令提示符前的行号用来定位第一个文件，而编辑提示符后的行号用来定位第二个文件。diff 命令将从左至右读取编辑提示行，从而指定你如何根据第二个文件来修改第一个文件。例如，对于 diff 命令编辑提示行从左至右为“5a10,14”，则该提示行表示需把第二个文件中的第 10 至 14 行插入到第一个文件的第 5 行之后。经过上述修改之后，第一个文件将与第二个文件完全相同。编辑提示行“3d”表示要使第一个文件与第二个文件完全相同，需删除第三行。编辑提示行“4,6c9”表示需对第一个文件的第 4 至 6 行进行修改（用第二个文件的第 9 行进行替换）。

如果需要，你可以把 diff 命令的输出通过重定向操作保存到文件中去。在下面的例子中，用户把 doc.v1 文件与 doc.v5 文件之间的差别保存到文件 change5

中去。如果你正在对原文件进行修改，你可以把修改后的文件保存下来，然后再比较原文件与修改后的文件之间的差别，并把它们的差别保存下来，此时被保存的文件将是你到目前为止的修改记录。在上例中，你可以把 doc.v1 文件理解为原文件，而把 doc.v5 文件看作为原文件的最新版本，则文件 change5 包含文件 doc.v5 相对于文件 doc.v1 的所有修改。当你继续进行修改的时候，你可以再用 diff 命令来记录文件的修改。总之，你可以使用该命令来记录你修改文件的过程。

```
$ diff  -b  doc.v1 doc.v5 > change5
```

## 14.5 正则表达式

正则表达式可以允许你对各种可能的模式进行匹配，你也可以用这些可能的模式定位于文本的不同位置。你可以在文本中搜索以不同的字符开头或结尾的模式，或者搜索与行尾或行首文本匹配的模式。正则表达式中的特殊字符有如下几个：“^”、“$”、“*”、“.”及“[]”等。符号“^”及符号“$”分别表示与文本行的行尾与行首匹配；星号“*”表示重复（包括 0 次与 1 次）出现的字符；而句点表示与任意的单个字符匹配；方括号“[]”表示与一类字符匹配。

### 14.5.1 与行尾及行首字符匹配的特殊字符：“^”与“$”

如果你想与行首的模式匹配，你可以在模式中首先键入特殊字符“^”，然后输入搜索的模式。在下面的例子中，正则表达式“^consists”与行首的模式“consists”匹配。

```
^consists
consists of a stream of
```

如下的例子使用特殊字符“$”来与行尾的模式匹配。

```
such$
be used to create such
```

### 14.5.2 与任何单个字符都匹配的特殊字符：“.”

句号“.”可以匹配任意的单个字符。句号“.”可以与你搜索模式中的任何字符都匹配。例如，模式“b.d”将搜索到由三个字符组成的字符串，该字符串的首字符为字符“b”，第三个字符为字符“d”，而第二个字符可以是任何字符。例如，该模式将与“bid”，“bad”，“bed”，“b+d”及“b d”等字符串匹配（注意空格也是一个有效字符，tab键也是如此）。

对于单个特殊字符“.”，必须通过其前后的字符或字符串才能达到应有的效果，即通过上下文关系。模式“b.d”中特殊字符“.”的上下文关系为：其前面的字符为字符“b”，其后面的字符为字符“d”。如果在模式“b.”后面没有字符“d”，那么，任何以字符“b”开头，不少于两个字符的字符串都与

上述模式匹配，如“bid”，“bath”，“bedroom”，“bump”，“submit”，“habit”及“harbor” 等。

你可以在正则表达式中使用任意多个特殊字符“.”。你可以在模式中连续键入多个特殊字符“.”来与前缀或后缀匹配。例如，模式“box..”将与字符串“box”开头、以任意两个字符结尾的字符串匹配。例如，它可以与“boxes”、“boxer”及“boxed”等字符串匹配。

你也可以在正则表达式中把特殊字符“.”与特殊字符“^”及“$”混合使用，以便搜索与行首或行尾的字符串相匹配的模式。假设你有一个文件，该文件中有些文本行以文件名为结尾，而你想在文件中搜索这些以文件名“week”为结尾的字符串，且该文件只有 5 个字符，例如 week1，week2，weeka 等，但不是 weekend，这时，你可以使用模式“week.$”来搜索与之匹配的字符串。

```
week.$
 report on week4
 report on week15 不匹配
 week1 weather        不匹配
```

### 14.5.3 重复匹配特殊字符：“*”

星号特殊字符“*”将多次重复匹配一个字符（包括 0 次与 1 次）。要匹配的字符是模式中星号“*”前面的字符。你可以把星号“*”理解为一个操作算子，而把它前面的字符作为其操作数。星号将重复匹配其前面的字符。下面是特殊字符“*”的句法：

c* 该正则表达式将与任意个 c 相匹配，例如“c”、“cc”、“ccc”、

“cccc”等。

当你需要替换一个字符的多次重复时，用特殊字符“*”号是非常方便的。下面的例子中，与字符“b”开头，其后跟着一系列连续的字符“o”相匹配字符串可以用正则表达式“bo*”来匹配。因此，与该正则表达式相匹配的字符或字符串有“boooo”、“bo”、“boo”及“b”等。

bo*
book
born
booom
zoom 不匹配

有必要给特殊字符“*”号提供一个上下文关系字符，而特殊字符“*”将与重复 0 次或任意次其前面的字符匹配。假如你想在文本中搜索一个或多个连续字符“b”，例如字符或字符窜“b”、“bb”或“bbb”等。你可能认为模式“b*”可以达到上述目的，但实际上是达不到的，因为模式“b*”也可以与出现 0 次的字符“b”匹配。因此文本行上的任何字符都与模式“b*”匹配。在下面的例子中，模式“b*”将匹配任何字符，因为尽管下面的文本行中的首字符都不是字符“b”，但都是出现了 0 次“b”字符的文本行。

b*
aaaaa
abb
aabbb

为了避免上述问题，你可以在特殊字符“*”前输入上下文关系字符。对于上述问题，你可以在特殊字符“*”输入字符“b”，此时，模式“bb*”将与一

个或任意个字符（窜）匹配。实际上，在进行模式匹配时，系统将首先搜索与字符“b”匹配的字符然后与重复任意次的字符“b”匹配（包括 0 次）。下面的例子中，模式“bb*”将与上例第二、三行文本中的字符串“bb”及“bbb”匹配。

bb*

aaaa 不匹配

abb

aabbb

模式“.*”与文本行上的任意字符都匹配。事实上，模式“.*”将与整行文本匹配。如果你在字符串“.*”前输入上下文关系字符或字符串，那么，你可以用该模式来匹配文本行中的各种文本块。例如，如果在模式“.*”之前有一字符串，那么，该字符串与模式“.*”合并成的模式将匹配该字符串至文本行行尾的所有文本；如果在模式“.*”之后有一字符串，那么，该模式“.*”与该字符串合并成的模式将匹配该文本行行首至该字符串的所有文本；如果在模式“.*”的前后都有字符或字符串，那么，由这些字符串与模式“.*”组成的文本将匹配文本行上前后字符串之间的所有文本。看下面的例子：模式“.*and”将匹配文本行行首至字符串“and”之间（包括字符串“and”）的所有文本；模式“and*.”将匹配从字符窜“and”（包括“and”）至文本行行尾之间的所有文本；另外，模式“o.*F”将与字符“o”与“F”之间（包括字符“o”与“F”）的所有的文本匹配。

.*andHello to you and to them Farewell

and.*Hello to you and to them Farewell

o.*F Hello to you and to them Farewell

由于特殊字符“*”可以与 0 次重复或任意次重复的字符串匹配，例如，模式“I.*t”与“Intelligent”匹配，同时它还与字符串“It”匹配，因此，你可以在进行模式搜索时给该特殊字符提供一个上下文关系字符。

### 14.5.4 字符类特殊字符：“[]”

如果你需要去匹配一组设定的字符或字符串，而不是匹配一个指定的字符或字符串，或者不是去匹配任意字符，那么，你可以使用特殊字符“[]”来定义一组字符。例如，你可能想去匹配与字符“A”或者“H”结尾的单词，如“seriesA”或“seriesH”，而不是单词“seriesB”或“seriesK”。此时，如果你输入特殊字符“.”去匹配，那么，你将与所有的字符都匹配。但是，你想指定你要匹配的字符只可能是“A”或者是“H”，这时，用特殊字符“[]”可以达到上述目的。

你可以在方括号特殊字符“[]”来匹配一组可能的字符，这组可能的字符依次被放置在特殊字符“[]”内（它们的顺序如何并不重要）。你可以把这组可能的字符认为是你定义的一个字符类。对于任意字符，如果字符集中定义了该字符，那么，我们可以认为该字符与这一字符集匹配。你可能已经意识到，正则表达式中的特殊字符“[]”与 Shell 下特殊字符“[”、“]”的用法非常相似。看下面的例子：用户用正则表达式“doc[agN]”来搜索以“doc”开头、以字符“a”、“g”或“N”结尾的字符串。该正则表达式与字符串“doca”，“docg”及“docN”等匹配，而不与字符串“docP”匹配。

```
doc [agN]
 List of documents
```

doca docb
docg docN docP

方括号特殊字符在匹配各种模式的前后缀时非常有用。例如，如果你想匹配文件中以前缀“week”开头的几个不同模式，如“week1”，“week2”等，你可以使用方括号特殊字符。在下面的例子中，模式“week[245]”将与字符串“week2”，“week4”等字符串匹配，而不与字符串“week1”匹配。

week[245]
week2 weather
report on week4
week1 reports 不匹配

方括号特殊字符也可以非常方便地去匹配以大写或者小写字母开头的模式。由于Linux系统是区分大小写的，因此，模式“computer”与模式“Computer”是不同的。字符串“computer”将不会与以大小字母“C”开头的单词“computer”匹配。为了同时搜索文件中以大写字母“C”及小写字母“c”开头的单词“computer”，你可以在方括号特殊字符中指定字符“c”或者“C”为搜索模式的可能首字符。例如，模式“[cC]omputer”将匹配文件中以字母“c”（不管是否是大小写）开头的单词“computer”。

有时，你可能想去搜索除了几种指定模式之外的所有其它可能模式（你可以把这些例外的模式认为是某类模式的补集）。为了搜索除了几种指定的模式之外的所有其它可能模式，你可以使用特殊字符“^”，该特殊字符被放置在方括号之内，而被放置在一些指定的字符之前。与前面所述的匹配相反，方括号特殊字符代表任何与不在方括号内指定的字符匹配。例如，为了匹配所有以字符串“week”开头而不以字符“5”和“7”结尾的字符串，你可以使用如下的

模式：

week [^57]

week7 weather 不匹配

reports on week4

week5 reports 不匹配

你可以用破折号“-”在方括号内指定一个字符范围。在 ASCII 字符集中，小写字母是一组连续的 ASCII 码字符，你可以用 [a-z] 来代表所有的小写字母。看下面的两个例子：在第一个例子中，模式“doc [a-z] ”将和字符串“doc”与任何小写字母组成的字符串匹配；你也可以在方括号内指定多个字符范围，第二个例子中的模式“doc [A-Za-z] ”将和字符串“doc”与所有英文字母组成的字符串匹配。

doc [a-z] doca docg docN docP doc1

doc [A-Za-z] doca docg docN docP doc1

通过指定范围，模式“week [14] [^12] ”可以被更精确地重新表示为模式“week [14] [3-9] ”。对于前一个特殊模式“ [^12] ”将与除字符“1”与“2”以外的任何字符（包括字母字符）匹配，而后一个特殊模式使用范围“ [3-9] ”，此时，特殊模式“ [3-9] ”将仅与字符“3”与字符“9”之间（包括字符“3”及“9”）的数字字符匹配。

week [14] [3-9]

week43 weather

reports on week15

week41 reports 不匹配

week4g reports 不匹配

你可以使用方括号特殊字符及其它特殊字符来创建一些非常高效的匹配模式。如果你使用方括号特殊字符与星号特殊字符的组合，那么，可以用它们来匹配一组任意重复的字符。例如，模式“doc[123]*”将搜索任意以字符串“doc”开头、以字符“1”、“2”与“3”的任意组合为结尾的字符串，如模式“doc221”及模式“doc3321311”等。要搜索字符串“doc”与任意的数字字符的组合，你可以用模式[0-9]来指定。因此，模式“doc[0-9]*”将匹配任何与“doc”开头，并以任何、任意个数字字符为结尾的字符串，如“doc582”、“doc7834103”等。你也可以使用方括号特殊字符与星号特殊字符的组合来搜索以指定的数字开头，或以指定的数字为结尾的整数字符，例如，模式“23[0-9]*”将搜索所有以“23”开头的数，如“23”、“235”及“2378945”等。模式“[0-9]＼..50”将匹配任何与“0.50”为结尾的数，如“7.50”、“1000.50”等。

使用指定的字符范围与星号特殊字符的组合，你可以搜索一些有一定特性的模式，例如，你可以搜索只含有小写字母的模式，或者只含有数字字符的模式。对于上述情况，你必须注意要给星号特殊字符提供用来搜索的上下文关系字符，同时需记住特殊符号星号“*”可以与一个字符的0次或任意次重复匹配。如果要把指定的字符范围作为星号字符的上下文关系字符，你只需在星号特殊字符“*”之前再键入一次该字符范围即可。例如，模式“[0-9][0-9]*”将与任何数字匹配，而模式“[A-Z][A-Z]*”将与任何大小写字母匹配。

[A-Z][A-Z]*we sold 9645 IBM and DEC components

[0-9][0-9]*we bought 9645 oranges today

有时，你可以在模式中同时使用多种特殊字符（见下表）。例如，模式“^

[^0-9] *”将选择从文本行行首至该行第一个数字字符串之前的所有文本，模式“ [^0-9] *$”将选择文本行上第一个数字字符串至该行行尾之间的所有文本。

```
^ [^0-9] *      we bought 9645 oranges today
 [^0-9] *$      we bought 9645 oranges today
```

| 特殊字符 | 匹配方式 | 操作 |
| --- | --- | --- |
| ^ | 从文本行行首开始 | 定位于文本行行首 |
| $ | 至文本行行尾 | 定位于文本行行尾 |
| . | 任意字符 | 匹配模式中的任意字符 |
| * | 重复匹配字符 | 匹配模式中的重复字符 |
| [] | 字符类 | 匹配模式中的字符类（字符集） |

### 14.5.5grep 实用程序与正则表达式

Shell 下的特殊字符可以让你搜索、匹配文件名，而正则表达式允许你搜索、匹配文件中的文本。在 grep 命令中使用正则表达式，你可以用指定的模式来搜索文件或匹配、定位文件中的文本行。你可以在 grep 实用程序的搜索模式中使用特殊字符，此时的模式即称为正则表达式。grep 正则表达式中可以使用的特殊字符有“*”、“.”、“[]”、“^”及“$”等。

例如，如果你想使用列出文件详细信息格式的命令 ls 显示你当前目录下的所有目录（仅仅是目录，而不包括文件），你可以首先用长格式生成一个所有目录及文件的列表，然后通过管道把该列表输入给命令 grep，通过该命令我们

可以输出目录名列表。为只输出目录名列表，你可以使用特殊字符“^”来搜索指定输出文本行的行首字符。前面已经介绍过，在以长格式方式命令 ls 输出时，第一个字符用来指定文件的类型。其中，字符“d”代表一个目录，，“l”代表一个符号链，而“a”代表一个普通文件。通过模式“^d”，grep 命令将匹配那些仅以字符“d”开始的文本行。

```
$ ls -l | grep '^d'
drwxr-x---   2  chris  512  Feb  10  04:30  reports
drwxr-x---   2  chris  512  Jan   6  01:30  letters
```

如果你只想列出那些有符号链的文件，你可以使用模式“^l”。

```
$ ls -l | grep '^l'
lrw-rw-r--   1  chris  group  4  Feb  14  10:30  lunch
```

必须分清 Shell 特殊字符与搜索模式中使用的特殊字符之间的区别。当你在 grep 模式中使用特殊字符时，你必须用引号把搜索模式括起来。必须注意，正则表达式及 Shell 下都使用的特殊字符有：星号“*”、句点号“.”及方括号“[]”。如果你不用引号把正则表达式括起来，那么，模式中使用的任何特殊字符将都将被解释为 Shell 特殊字符。如果不用引号括起来，星号“*”将被用来生成文件名，而不是被 grep 实用程序用来解释为要搜索的重复字符，而用引号括起来后，系统将确保 grep 实用程序把特殊字符看作为正则表达式的一部分。 在下面的例子中，模式中使用的星号“*”特殊字符是正则表达式特殊字符，而在文件列表参数中使用的星号“*”特殊字符为 Shell 特殊字符。此命令将搜索当前目录下的所有文件，看这些文件中是否包含字符串“report”或其后面有任意个重复的“s”字符的字符串。

```
$ grep 'reports*' *
```

mydata: The report was sitting on his desk.
weather:The weather repotrs were totally accurate.

特殊字符“ []”将指定一组匹配的字符，或者指定一组一系列的匹配字符，或者指定一系列非匹配的字符。例如，模式“doc[abc]”将与字符串“doca”、“docb”及“docc”匹配，而不与字符串“docd”匹配。该模式同样可以由一个字符串范围“ [a-c]”来指定，即由正则表达式“doc [a-z] ”来表示。对于模式“doc [^ab] ”，它将匹配任何以字符串“doc”开头，而不以字符“a”或“b”为结尾的字符串，因此字符串“docc”将与上述模式匹配，而字符串“doca”、“docb”不与其匹配。

$ grep 'doc [abc] ' myletter
File letter doca and docb.
We need to redo docc.

### 14.5.6 完全正则表达式与扩展特殊字符：“|”、“ ()”、“+”及“?”

有些 Linux 实用程序，例如 egrep 命令及 awk 命令可以在它们的模式中使用一组扩展特殊字符。这些扩展特殊字符有：“|”、“ ()”、“+”及“?”等。它们的作用见下表。

| 扩展特殊字符 | 作用 |
| --- | --- |
| pattern \| pattern | 逻辑或，用来搜索两个可能的指定模式 |
| (pattern) | 用来搜索可能的模式 |

| | |
|---|---|
| char + | 与其前一字符的 1 次或多次重复匹配 |
| char ? | 与其前一字符的 0 次或 1 次匹配 |

其中，特殊字符“+”及“?”是星号“*”特殊字符的一个变种，而特殊字符“|”及“()”提供了新的功能。使用了上述扩展特殊字符的模式称为完全正则表达式。在标准行编辑器 Ed 及 Ex 中不能使用这些扩展特殊字符，本节讨论的实用程序中，只有 egrep 及 awk 可以使用扩展特殊字符。

特殊字符“+”将与单个字符的一个或该字符的任意次重复匹配。例如，对于模式“t+”将至少与一个以上（包括一个）的字符“t”匹配，其功能与模式“tt*”完全相同。因此，上述模式将与字符串“sitting”或字符串“bitting”匹配，但不与字符串“ziing”匹配。扩展特殊字符“?”将与指定字符的 0 次或一次字符匹配。例如，由于“t?”将与一个或 0 个字符“t”匹配，表达式“it?i”将与字符串“ziing”或“biting”匹配，但不与字符串“sitting”匹配。看下面的例子，要搜索的模式是：字符“a”后任意次重复的字符“n”，然后跟一字符“e”。通过使用扩展特殊字符“+”，正则表达式“an+e”将匹配由字符“a”、字符“n”的任意次重复及字符“e”组成的字符串。因此，正则表达式与字符串“anew”及“canned”等匹配。在下面的第二个例子中，正则表达式“an?e”将搜索由字符“a”、0 个或一个字符“n”及字符“e”组成的字符串，因此，该表达式将与“ane”与“anew”匹配，而不与字符窜“canned”匹配。

```
an+e
anew
canned
an?e
anew
```

canned 不匹配

扩展特殊字符“|”与“（）”主要用来对模式块起作用，而不是只对字符起作用。特殊字符“|”是一个逻辑或特殊字符，用来在一个正则表达式中指定多个要搜索的字符串。尽管这些字符串是这个正则表达式的一部分，但它们是被作为相对独立的字符串来搜索的。例如正则表达式“create | stream”可以用来搜索字符串“create”或字符串“stream”。

```
create | stream
 consists of a stream of
 be used to create such
```

egrep 实用程序综合了实用程序 grep 与 fgrep 的功能。与 fgrep 命令一样，egrep 命令可以同时搜索多个模式；同时，与 grep 命令一样，egrep 命令可以在搜索时使用特殊字符及使用正则表达式进行搜索。但是，与 grep 命令又有所不同，在用 egrep 命令进行搜索时，可以在搜索模式中使用扩展特殊字符，如逻辑或操作等。因此，可以这么说，egrep 命令是上述三个过滤器搜索实用程序中功能最强大的一个。

为了一次搜索多个模式，你可以在命令行上输入要搜索的模式，并用换行字符把它们分割开（fgrep 命令就是如此），也可以在一个正则表达式中使用逻辑或“|”特殊字符来指定多个搜索的模式。尽管这些搜索的模式实际上只是一个正则表达式的一部分，但它们在搜索时都是相互独立的模式。例如，对于模式“create | stream”，egrep 将搜索模式“create”或“stream”。

```
$ egrep 'create | stream' preface
consists of a stream of
be used to create such
```

特殊字符组可以用来帮助你精确指定你所要搜索的模式。例如，如果你要列出在 6:00a.m.与 12:00p.m.之间更新的所有文件，如果你只是使用特殊字符类指定的模式“ [01] [6-90-2] ”来进行模式匹配，那么，你将不仅匹配到字符串“ 12”，而且匹配到字符串“ 02”；不仅匹配到字符串“ 06”，而且匹配到字符串“ 16”。但是，使用字符组与特殊字符“ |”，你可以解决上述问题。正则表达式“（0 [6-9] |1 [0-2] ）”将成功地匹配到你在 6:00a.m.与 12:00p.m. 之间更新的所有文件（详见下面的例子）。

```
$ ls -l | egrep '(0 [6-9] | 1 [0-2] ):.*$'
-rw-r--r--1chris  weather  207Jan 2710:55forecast
-rw-rw-r--   1chris  weather  308  Feb 1712:40monday
-rw-r--r-x   1chris  weather  789  Feb 0606:45roster
-rw-rw-r-x    1chris  weather  942  Feb 1208:20strom
```

## 14.6 本章小结：过滤器

过滤器接收标准输入输入的数据，经过相应的操作，然后再输出经过“过滤”后的数据（原始数据并不会被修改），有三种常见的过滤器：文件过滤器、编辑过滤器及数据过滤器。文件过滤器主要用来对文件执行一些基本的操作，如搜索、显示文件中匹配的模式。编辑过滤器主要用来执行编辑操作。数据过滤器主要用来修改、处理文件中的数据域。所有这些

过滤器可以使用正则表达式来执行功能强大的模式匹配操作。尽管有很多过滤器把文件名作为它们的参数，但所有的过滤器都以标准输入作为其输入数

据。这些过滤器也允许你使用管道把一个过滤器的输出数据输入到另一个过滤器。在命令行上，你可以键入一系列的过滤器命令，并把其前一个过滤器输出的数据作为下一个过滤器命令的输入。从这种意义上说，一个输入的数据可以被传递给几个过滤器。在传递这一数据的过程中，每一个过滤器在输出数据、传递给下一个过滤器时都会作必要的修改。

根据过滤器的功能不同，它们可以生成各种各样的输出。有些过滤器仅生成一些简单的统计数据，如 wc 命令仅输出文件中的字符数、单词或文本行数。另外一些过滤器有选择性的输出部分输入的文本。例如，spell 命令将仅输出有拼写错误的单词；head 命令仅输出文件的头部的部分文本行，而 tail 命令仅输出文件尾部的部分文本行；命令 diff 命令将输出两文件的文本行之间的差别。还有另外一些过滤器输出所有其输入的文本，但输出的是经过部分修改后的文本。pr 命令将以页面格式在文件中标明页眉与页号，而 sort 命令将输出经过排序后的文本。

大多数的过滤器都有一组选项，这些选项可以使你进一步选择、修改过滤器输出数据的方式。例如，带 -n 选项的命令 pr 可以输出带行号的文本，而带 -num 选项（num 为一个数字类型的参数）的 tail 命令将允许你输出至文件结尾处的 num 行文本。

### 14.6.1 文件过滤器

文件过滤器用来对文件执行一些基本的操作，例如用来显示、编辑文件内容、搜索文件中的模式、生成格式化文本及备份文件等。例如，命令 cat、head 及 tail 过滤器命令用来显示文件。其中，cat 命令用来显示这个文件的内容，而

head 命令用来列出文件的头几行，tail 命令用来列出文件的最后几行。tail 命令有一些选项，通过这些选项，你可以选择要显示多少行至文件尾的文本及怎样显示这些文本。

过滤器命令 cmp 及 comm 命令用来比较文件。cmp 过滤器用来对字符进行逐个比较，并输出行号及第一个不相同的字符。comm 过滤器用来对文本行进行逐行比较，并输出两个文件中相同的文本行及不相同的文本。grep 命令用来搜索文件中特定的模式。你可以用该命令同时搜索多个文件，该命令将分别输出文件名及该文件中与模式匹配的文本行或行号。grep 命令也有几个选项，例如，-n 选项将在输出匹配的文本行的同时输出行号。

pr 命令用来输出一个或多个格式化的文本。pr 命令也有多个选项，例如，使用-h 选项，你可以给输出的文件指定一个页眉；使用-w 选项，你可以在文本中添加页码。pr 命令的一个非常有用的选项是可以用-n 选项来输出文本行行号。该选项与-t 选项（禁止添加页眉）一起可以用来生成一个简单的、带行号的格式化文本。

你可以用 cpio 命令来管理备份的文件。通过该命令，你可以把一个文件拷贝到档案（archive）格式文件中去，必要时，你可以再展开该文件。cpio 命令并不直接存取该文件，只是通过重定向操作来读取或保存一个档案。cpio 命令有两个重要的选项，其中，-i 选项用来保存档案，而-o 选项用来展开文件。你也可以把目录及其文件保存到一个档案文件中，但你首先必须用 find 命令搜索你当前目录下所有文件的全路径名，然后通过管道操作把这些文件备份到档案文件中。

### 14.6.2 编辑过滤器

Linux 系统中的文本文件是以行为单位来组织文件的。许多编辑实用程序，如行编辑器及编辑过滤器等，都把文件看作是一系列的文本行来进行定位与编辑的。所有的编辑器都有一组核心行编辑命令，并通过这组命令对文本行上的文本进行各种编辑操作。文件中的文本是通过行来定位的，因此，我们可以用行号或通过模式匹配来定位文本行（即通过定位文本行中的搜索文本来定位文本行）。对于一些特定的文本行，我们还可以使用特殊行定位符来定位它们，例如，字符“$”用来定位于文件的最后一行，字符“.”用来定位当前行。在一些行编辑器中，如 Ed 编辑器，字符“+”用来定位当前行的下一行，字符“-”用来定位当前行的上一行。同时，你还可以在 k 命令后跟单个字符来标识当前文本行，然后你可以在该标识的字符之前键入单引号来定位上述标识的文本。

当你定位到文本行后，你可以用行编辑命令来在文本行上输入文本、删除文本或替换文本，同时，你还可以移动或拷贝文本行。如果你想在文本行上修改指定的文本，那么，你可以使用文本替换命令，该命令可以让你用一个指定的文本去替换另一个指定的文本。需注意的是，该指定的文本既是被用来搜索的文本，也是将被替换的文本。

tr、diff 及 sed 命令用来对输入数据进行编辑操作，而这些输入数据是从标准输入上读取的。它们用来对输入的文本文件进行编辑，并生成相应文件的修改版。这些过滤器编辑命令可以从文件或标准输入上接收数据的输入，经过一定的修改后，再输出编辑后的输入。你可以用重定向操作把将要输出的数据保存到一个文件或输出到一个设备上如打印机等。

sed 命令实际上是一个流式行编辑器，该编辑器通过行编辑命令对输入的数据进行编辑，并产生输入数据的修改版。但是，与大多数编辑器不同，sed 编辑器的比较命令在缺省条件下是全局命令，但你可以使用模式或行号来严格限制编辑命令的操作范围。例如，在 sed 编辑命令之前输入搜索模式这一选项，则这些命令将被严格限制在与搜索模式相匹配的文本行上。

diff 命令用来比较两个文件，然后输出两文件中有差别的文本行。该命令可以详细地输出文件的编辑信息，以帮助用户经过怎样的修改可以把第一个文件变成与第二个文件完全相同的文件。带-e 选项的 diff 命令可以输出行编辑命令，通过这些行编辑命令，你可以把第一个文件修改成第二个文件的一个精确拷贝，即把第一个文件修改成与第二个文件完全相同的文件。

tr 命令可以转换输入数据流中的指定字符。它可以用两个指定的字符列表特殊执行几个字符转换操作。执行该命令时，第一个字符列表中的字符被转换成第二个字符列表中对应的字符。tr 命令也有一些选项，这些选项可以使你删除或替换输入数据中重复的字符。tr 命令一个常见的应用是对文件进行简单的加密。

### 14.6.3 正则表达式

在进行文本搜索的时候，有时你需要功能更强大、使用更灵活的模式来进行模式搜索。许多有编辑能力的实用程序在进行模式搜索的时候都可以使用一组标准的特殊字符，如果一个搜索模式中包含这些特殊字符，那么，我们称该搜索模式为正则表达式。尽管有些特殊字符与 Shell 下的一些特殊字符相同，但它们的功能是不同的。正则表达式中的特殊字符是用来搜索文本的，而 Shell

下的搜索匹配的文件名的。

用正则表达式特殊字符，你可以定位文本行的行首或行尾（字符“$”及字符“^”）；可以匹配一个字符的任意次重复（字符“*”）；可以匹配任意的字符或字符集中任意可能的字符（字符“.”及字符“[”、“]”）。你会发现，很多具有编辑能力的实用程序都可以使用正则表达式。还有一些实用程序如 sed 命令及 awk 命令可以在正则表达式中使用扩展特殊字符。

特殊字符只有在模式中才具有特殊的意义，而在模式之外，它们可能代表不同的意义，或者仅仅是一个普通的字符。例如，符号“$”在行编辑器中用来定位文件的最后一行，在模式中，特殊字符“$”用来定位文本行行尾，而在替换命令中，符号“$”仅仅是一个普通的字符。

对于替换命令，替换文本有其自己的一组特殊字符，你可以用这组特殊字符来“构造”替换的文本。如果想在搜索模式或替换文本中使用特殊字符，你可以用反斜扛“＼”来引用一个特殊字符。例如，如果要搜索一个包含特殊字符“.”的模式，你必须用反斜扛符号“＼”来引用该字符，即用符号“＼.”来表示。

命令 grep 用来在文件中执行模式搜索，并输出那些包含搜索模式的文本行。grep 命令有很多选项，你可以用这些选项来决定是否输出文本行行号，或者是输出文件中匹配的文本行还是非匹配的文本行。grep 命令有两个变种命令：fgrep 及 egrep 命令。正如在第 8 章中指出的那样，fgrep 命令可以同时搜索多个模式，但该命令不允许在搜索的模式中使用特殊字符。egrep 命令也可以同时搜索多个模式，但它允许在搜索的模式中使用特殊字符。事实上，egrep 命令也允许使用扩展特殊字符，如“|”、“+”及“?”等。

### 14.6.4 数据过滤器

Linux 系统中还有一组过滤器命令，它们可以对输入数据流执行各种数据操作。这组数据过滤器命令与其它过滤器一样，首先接收数据输入，然后对输入的数据流执行各种操作，最后输出经过各种操作（如修改）后的数据。有些文件的文本类似于数据库文件，它们以数据域的形式组织文本：文件中的每个文本行都是一个记录，而文件行上的每个单词都是该记录的数据域。数据过滤器命令就是针对这类文件设计的。数据过滤器命令把包含这种记录的文件作为其输入，然后用给定的评判标准输出选择的记录。

有五个数据过滤器命令，它们是 sort 命令、cut 命令、paste 命令、join 命令及 uniq 命令（见表 14-3）。sort 命令用来对文件中的所有记录进行排序（如对指定的字段按字母顺序进行排序），并生成记录排序后的文件。sort 命令当然也可以对任意的文本文件的文本行进行排序。

cut 命令输出数据文件中所有选择的字段。paste 命令能够合并几个数据文件中的记录，并输出合并后的记录。join 命令首先比较两个文件中指定字段的值，再合并这两个文件中相应的记录，最后输出这些合并后的记录。uniq 命令用来确定那些值相同的字段，该命令可以用来统计文件中有多少字段的值相同，也可以在输出数据时用该命令来删除文件中所有重复的字段。

尽管上述数据过滤器命令不能执行专业数据库管理软件中那些复杂的数据库操作，但是你会发现它们还是可以执行专业数据库管理软件中一些常见的数据库操作。例如，你可以进行数据排序、选择显示的字段，你也可以有选择性的恢复不同文件中匹配的记录，你甚至还可以用多个数据过滤器命令来完成复杂的数据查询、排序操作。例如，你可以用 join 命令合并从不同文件中选择的

记录，然后通过管道把合并后的记录输出至 sort 命令，以对合并后的记录进行排序。

表 14-1 文件过滤器命令

| 命令 | 功能 |
| --- | --- |
| cat filename | 显示一个文件的内容，它用文件名作为其参数，并直接把文件的内容输出到标准输出上，其缺省的标准输出是屏幕 |
| tee filename | 在把标准输入输出到标准输出的同时把它拷贝到一个或多个文件中。该命令通常与另一个过滤器命令一起使用，从而在把输出数据送到另一个过滤器或实用程序的同时保存该输出数据 |
| head filename | 显示一个文件的头几行，缺省值是后十行，但是，你可以指定你要显示的文本行行数 |
| tail [ +/-num ] [options] filename | 显示一个文件的最后几行 ，缺省值是前十行，但是，你可以指定你要显示的文本行行数 |
| 选项： | |
| -num | 用 num 参数来显示指定要显示的文本行行数（从文件尾部向前计算）。 |
| +num | 显示页号为 num 以后的所以文本 |
| -c | 以字符数来显示文本。该选项与-num 或者+num 选项同时使用时，num 表示要显示的字符数 |
| -l | 以文本行行数来显示文本。该选项与-num 或者+num |

| | |
|---|---|
| | 选项同时使用时，num 表示要显示的文本行行数，为缺省选项 |
| -r | 以反序显示文本行。该选项与-num 或者+num 选项同时使用时，num 表示要以反序显示的文本行行数。+lr 表示以反序显示所有的文本行 |
| wc filename | 统计并输出文件中包含的行数、单词数及字符数 |
| 选项： | |
| c | 只统计文件中的字符数 |
| l | 只统计文件中的行数 |
| w | 只统计文件中的单词数 |
| spell filename | 检查文件中每个单词的拼写情况，并只输出那些有拼写错误的单词 |
| +filename | 该选项可以用你自己所定义的词汇来检查文件中的拼写错误 |
| sort filename | 输出经过排序后的文件内容 |
| comm filename filename | 逐行比较两个文件的内容，并根据两文件文本行的内容输出它们之间的差别 |
| grep [ options ] pattern filenames | 搜索一个或多个文件中与指定模式匹配的字符串，并列出该字符串所在的文本行 |
| 选项： | |
| l | 忽略大小写 |
| C | 只输出每个文件中包含匹配模式的行的数目 |

| | |
|---|---|
| L | 显示每个包含有一处或多处与搜索模式匹配的文件名 |
| N | 在每个匹配的文本行前面加上该行在文件中的行号 |
| V | 输出那些所有不包含搜索模式的文本行 |
| fgrep [ options ] patterns flle-list | 同时用多个模式去搜索文件名列表中所有文件。该命令的运行速度要比 grep 及 egrep 命令要快。但是，不能在搜索模式中使用特殊字符，也就是不能搜索正则表达式 |
| 选项： | |
| -f filename | 如果有该选项，fgrep 命令将从文件名为 filename 的文件中读取所有要搜索的模式。 |
| egrep [ options ] patterns file-list | 用模式搜索文件列表中的所有文件。与 fgrep 命令一样，它也可以从文件中读取要搜索的模式列表，与 grep 命令一样，在搜索模式中可以使用正则表达式（即可以使用特殊字符），与 grep 命令的不同之处在于 egrep 命令可以使用扩展特殊字符，如：?、\|、+等。 |
| Pr | 输出格式化后的文本，包括添加页码、页眉及其它指定的格式化方式。 |
| cpio -o > | archive-file |
| cpio -i > | filenames |
| < archive-file | 把文件拷贝到档案文件中，或从档案文件中释放已 |

| | |
|---|---|
| | 备份的文件。该命令有两种操作方式：一种是使用-o 选项把文件拷贝到档案文件中去，另一个选项是使用-i 选项从档案文件中释放已经备份了的文件。当要把文件拷贝到档案文件中时，你可以首先用 ls 命令或 find 命令生成文件名列表。 |

表 14-2 编辑过滤器命令

| 命令 | 功能 |
|---|---|
| sed editing-command file-list | 输出经过编辑修改后的文件。sed 命令以编辑命令及文件名列表作为其参数。编辑命令对从文件名列表中读取的数据进行编辑操作，然后输出经过修改后的文件拷贝。命令行上的编辑命令参数与 Ed 行编辑器中使用的命令相同。 |
| 选项： | |
| -n | 用该选项，sed 命令不会自动输出文本行。该选项通常与打印命令一起使用，以便仅输出选择的文本行。 |
| f filename | 如果有该选项，sed 命令将从指定的文件中读取编辑命令。 |
| A | 在下一文本行后添加文本 |
| I | 在前一文本行上添加文本 |
| C | 修改文本 |

| | |
|---|---|
| D | 删除文本 |
| P | 打印文本 |
| W | 把文本行写到一个文件中去 |
| R | 从文件中读取文本行 |
| Q | 修改后退出 sed 编辑器 |
| N | 跳至下一行 |
| s/pattern/replacement | 用 replacement 文本去搜索替换文件中匹配的模式 |
| g s/pat/rep/g | 在文本行上进行全局替换 |
| p s/pat/rep/p | 输出修改后的文本行 |
| w s/pat/rep/w fname | 把修改后的文本行写入文件，要写入的文本行可以 |
| /pattern/ | 用一个模式去搜索与定位 |
| diff filename filename | 逐行比较两个文件之间的差异，并输出有差异的文本行及怎样修改这些差异的指令，通过这些信息，你可以将第一个文件修改为与第二个文件完全相同的文件 |
| f1-linenum a f2-line1, f2-line2 | 把文件 f2 中的 f2-line1 与 f2-line2 之间的所有文本添加到文件 f1 的第 f1-linenum 行之后 f1-line1,f1-line2 d f1-linenum 删除文件 f1 中 f1-line1 与 f1-line2 之间的所有文本 |
| f1-line1,f1-line2 c f2-line1,f2-line2 | 用 f2 文件中 f2-line1 与 f2-line2 之间的文本去替换文件 f1 中 f1-line1 与 f1-line2 之间的所有文本 |
| 选项： | |

| | |
|---|---|
| b | 忽略空格及制表符 |
| c | 输出不同文本行的上下行文本。其缺省值是输出不同文本行的上下三行文本 |
| e | 输出一组 Ed 编辑器比较命令列表，通过这些命令，你可以把第一个文件修改成与第二个文件完全相同的文件 |
| tr first-character-listsecond-character-list | 把输入文件中所有出现的第一个字符列表中的字符用第二个字符列表中的字符代替 |
| **选项：** | |
| [] | 指定一系列字符 |
| d | 用该选项，tr 将删除输入数据中任何包含在字符列表中的任何字符 |
| c | 替换那些不在字符列表中的字符 |
| s | 删除文件中重复的字符 |

表 14-3 数据过滤器命令

| 命令 | 功能 |
|---|---|
| sort -option file-list | 对输入的数据以行为单位进行排序，以生成一个经过排序后的文件。你可以以字母为序进行排序（顺序或反序排序），或执行数字排序。 |
| **选项：** | |

| | |
|---|---|
| -o filename | 把 sort 命令的输出保存到文件 filename 中。你可以使用该选项来安全地覆盖原文件，生成一个经过排序后的文件。 |
| C | 检查该文件是否是经过排序后的文件。如果该文件是没排序的文件，sort 命令将显示一个错误信息，否则，它不显示任何信息。 |
| M | 合并已经排序后的文件。 |
| U | 对于重复的行，该选项将仅输出一次 |
| D | 该选项不理睬所有非字母、数字符或空白字符，即以字母表顺序进行排序 |
| F | 忽略大小写 |
| I | 忽略非打印字符 |
| M | 按月名排序，该选项将排序月份字段 |
| N | 按字段的算术值进行排序，而不是按字符顺序 |
| R | 以反序进行排序 |
| B | 忽略字段前的空白字符 |
| +num | 文本行上需要忽略的字段，即对所有行的第 num 个字段的下一字段进行排序，如+2 将忽略前两个字段，而对所有文本行的第三个字段进行排序 |
| -num | 使 sort 命令在第 num 个字段前停止排序 |
| -tc | 指定一个新的分割符，如字符“c”，缺省的字符是空格 |

| | |
|---|---|
| paste -option file-list | 把不同文件中的文本行合并，然后输出它们 |
| -d delimiter-list | 你可以用你自己指定的字符作为定界符，以分离合并后的文本行 |
| cut -option file-list | 拷贝文件中指定的字段或指定的字段列。在使用 cut 命令的过程中，你必须使用-f 选项或-c 选项 |
| 选项： | |
| -fnum | 指定你要从文件中拷贝出的字段，字段从 1 开始计算 |
| -fnum1,num2 | 指定要拷贝的字段 |
| -fnum1-num2 | 指定从 num1 开始至 num2 的一系列字段 |
| -cnum-num | 指定要拷贝的一列字符 |
| -ddelimiter-list | 指定自己的定界符 |
| join -option file-list | 排除输入数据中重复的文本行。你也可以比较选定字段来排除选定域中相同的文本行（此时，如果选定的字段相同，那么，该命令认为这些文本行是重复行）。 |
| uniq option input-file output-file | |
| 选项： | |
| c | 使用该选项，uniq 命令将输出文本中重复的文本行，同时在该行的文本行的前面显示重复的次数 |
| d | 使用该选项，uniq 命令将仅输出重复的文本行 |
| u | 使用该选项，uniq 命令将仅输出非重复的文本行 |
| -num | 在进行比较时需要忽略的字段数，只比较剩下的字 |

| | |
|---|---|
| +num | 段<br>在进行比较时需要忽略的字符数，只比较剩下的字符 |

# 第 15 章 BASH Shell

Linux 下主要有三种不同的 Shell：Bourne Again Shell（BASH），Public Korn Shell（PDKSH）及 TCSH Shell。这些 Shell 都是相应 UNIX 下 Shell 的增强版。BASH Shell 是 UNIX 下的 Borune Shell 的增强版，它包括了 Korn Shell 和 C Shell 的大多数高级特性。TCSH Shell 是 C Shell 的增强版，而 C Shell 最初是 BSD 版 UNIX 系统下开发的一种 Shell。PDKSH 是 UNIX 系统下 Korn Shell 的子集。尽管这些 UNIX 系统下的 Shell 有较大的区别，但它们仍然有许多共同的特性。虽然 UNIX 系统中 Bourne Shell 缺乏其它 UNIX 版本下 Shell 的许多功能，但在 Linux 中，BASH Shell 同时具有 Korn Shell、C Shell 及 TCSH Shell 的大多数高级特性。

在 Lunix 系统中，所有以上三种 Shell 你都可以使用，但其缺省 Shell 是 BASH Shell。本章中所有的例子都是使用的 BASH Shell。当你登陆进入 Lunix 系统后，系统将采用缺省 Shell，但你可以转换到另一种 Shell 下：键入 tcsh，系统将使用 TCSH Shell；键入 bash，系统将使用 BASH Shell。你可以用 CTRL＋d 或者用 exit 命令退出一种 Shell。

在使用 Lunix 系统的过程中，你可以只使用一种 Shell 完成你所要做的工作。本书第 5 章讨论了所有 Shell 的一般特性，而本章及下一章将讨论 BASH Shell 及 TCSH Shell。本书不讨论 PDKSH Shell，因为 PDKSH Shell 是 Korn Shell

的一个子集，且 Korn Shell 的大多数高级特性都可以在 BASH Shell 中找到。

## 15.1 命令和文件名扩展特性

BASH Shell 命令行有一个与 TCSH Shell 非常相似的内在特性：命令与文件名扩展特性。当你输入一个还没完成的命令或文件名时，你只需键入 TAB 键就能激活命令与文件名扩展特性，帮你完成该命令剩余的输入。如果有多个命令或文件的前缀相同，BASH Shell 将响

铃并等待你输入足够的字符，以便选择唯一的命令或文件名。在下面的例子中，用户键入 cat 命令和一个还没有输全的文件名，通过键入 TAB 键，系统开始搜索与之匹配的命令或文件名，如果找到，系统将自动输入搜索到的命令或文件名。用户敲 ENTER 键后，系统将执行这条命令。

```
$ cat pre TAB
$ cat preface
```

BASH Shell 也能列出你当前目录下的部分匹配的文件名来完成文件名扩展。如果你键入“ESC”键，然后键入“？”，Shell 将列出所有与你输入的字符串相匹配的文件名。我们看下面的例子：在没有完成的输入后键入“ESC？”，Shell 将列出所有与输入的字符串相匹配的文件名，然后 Shell 回显命令行，根据列出的文件名，你可以键入你要输入的文件名或按“TAB”键来完成文件名的字符扩展。

```
$ ls
document docudrama
```

```
$cat doc ESC ?
document
docudrama
$ cat docudrama
```

## 15.2 命令行编辑

BASH Shell 本身具有命令行编辑功能，能使你在执行所键入的命令之前方便地修改你所键入的命令。如果你在键入命令时犯了一个拼写错误，你只需在运行你所键入的命令之前，使用编辑命令来纠正你的拼写错误，然后执行它，而不用再重新输入整行命令。这对那些要键入长路径文件名作参数的命令特别有用。

命令行编辑的操作是 Emacs 编辑命令的子集。你可用 CTRL-f 或者右方向键向前移向一字符，CTRL-b 或者左方向键向后移向一字符。CTRL-d 或 DEL 键将删除光标所在位置的字符。要增加字符，只需把光标移到你所希望插入字符的地方，插入新的字符。在任何时候你可以键入“ENTER”键去执行一个命令。在下一节，你将用命令行编辑操作去修改你先前所键入的命令。

## 15.3 实 用 命 令

在 BASH Shell 中，history 实用命令能够保存你最近所执行的一些命令。这些最近你所执行的命令在被 history 实用命令保存时，其历史记录号从 1 开始，但只有有限个命令可以被保存下来，其缺省值是 500。history 实用命令用一小段内存来记录有限个最近你所执行的命令。要查看最近你所执行的一组命令，只用键入 history 命令，然后键入 ENTER（回车键），你最近所执行的一组命令即按执行的先后顺序被显示出来（命令前的数字为历史记录号）。

```
$ history
1 cp mydata today
2 vi mydata
3 mv mydata reports
4 cd reports
5 ls
...
```

所有这些命令都被称之为事件（event）。一个事件表示一个操作已经发生，也即一个命令被执行。这些事件根据它们被执行的先后顺序用数字给以标识，这一标识称为历史事件号。最后执行的历史事件的事件号最大。每个事件都可被它的历史事件号或命令的初始字符或字符串等确定。

histroy 实用命令能够使你查询以前的事件，并可把它们显示到命令行上或执行它。最简便的办法是通过上下光标键，把先前的一个个事件逐次显示到命令行。你不必要先运行 history 命令，就可执行上述操作。按一下上光标键，你

上一次执行的一个事件将出现在命令行上，再按一下，上一次的前一事件又会出现在命令行上；按一下下光标键，将会使当前事件的下一事件出现在光标行上。BASH Shell 也可以通过键入“ESC”、“TAB”键来完成对历史事件的字符扩展。正如标准的命令行扩展特性，你键入历史事件的部分字符串，然后键入“ESC”，再键入“TAB”键，与你刚才键入的字符串相匹配的历史事件将自动扩展并回显到命令行处。如果不止一个事件与你输入的字符串相匹配，你将会听到一声响铃，继续键入字符或字符串，Shell 将会唯一确定你所要键入的历史事件。然后，你可以用命令行编辑操作来编辑显示在命令行上的事件。通过左右键在命令行上移动，你可以把光标停留在任何地方，插入你所需要的字符或字符串，如用 BACKSPACE 键和 DEL 键，你可以删除字符或字符串。按回键后，这一事件将被执行。你也可以通过！命令来查询和执行历史事件。在！命令最后键入能够确定历史事件的关联字符，这关联字符可以是历史事件的历史事件号，或者该事件的前几个字符串。在下面的例子中，历史事件号为 3 的事件被查询到。同时，又用其开头几个字符串去匹配，也被查询到。

```
$ !3
mv mydata  reports
$ !mv
mv mydata reports
```

你也可以用一个偏移量（相对于历史事件列表中最后一个事件）来查询历史事件。负的偏移量将通从历史事件列表表尾向前偏移，从而查询到所要的事件。在下面的例子中，历史事件号为 2 的事件“vi mydata”就是用一个负的偏移量查询到，并执行的。必须注意的是，这个偏移量是相对于相对于历史事件列表中的最后一个事件的。在本例中，历史事件列表中最后一个事件是事件 5，

历史事件列表中第一个事件为 1。从历史事件号为 5 的事件，往前偏移 4，即是历史事件号为 2 的事件。

```
$ !-4
vi mydata
```

如果“!”命令后没有任何参数，则系统将默认为上一事件。在下面的例子中，用户在命令行上键入无参数的“!”命令，系统将执行上一事件。本例中，上一事件为“ls”命令。

```
$ !ls
mydata today reports
```

你也可以用模式去搜索一个历史事件。该搜索的模式必须用符号“?”括起来。在下面的例子中，模式“?myd?”用来搜索历史事件号为 3 的历史事件 vi mydata。

```
$ !?myd?
vi mydata
```

### 15.3.1 历史事件编辑

你也可以编辑历史事件列表中你先前执行的任何事件。在 BASH Shell 中，有两种方法来实现。你可以用命令行编辑功能去查询和编辑历史事件列表中的任何事件。同时，你也可以用 history fc 命令选项来查询一个事件并用全屏幕编辑器 vi 来编辑它。两种方法涉及到两种不同的编辑器。前一种命令的局限性在于它只能用命令行行编辑器进行单行编辑（其编辑命令是 Emacs 编辑器命令的子集）。但是，它能使你非常方便地查询历史事件列表中的事件。第二种方法

用功能强大的标准全屏幕编辑器 vi 来完成编辑，但它仅针对某些特定的历史事件。

用命令行编辑器，你不仅可以编辑当前的命令，而且可以编辑历史事件列表中的事件并执行之。你可以用 CTRL-P 命令转移到历史事件列表中当前的事件的前一事件，用 CTRL-n 命令转移到历史事件列表中当前的事件的下一事件，用 ESC <命令转移到历史事件列表中最初的一事件，用 ESC>命令转移到历史事件列表中的最后一事件。你甚至可以用一种方式去搜索指定的事件，例如，在反斜“＼”后键入搜索模式将在历史文件列表中向后(backward)搜索历史文件，而在问号“？”后键入搜索模式将向前(forward)搜索历史文件。键入“n”命令重复进行上次搜索。

一旦你找到编辑的事件，你可以使用 Emacs 命令行编辑命令对该文件进行行编辑。按 CTRL-d 进行字符删除，CTRL-f 或光标右键前移一个字符串，CRTL-b 或光标左键后移一个字符。如果要增加一个字符，把光标移到你需要输入字符的地方键入字符即可。表 15-1 列出了查询历史事件列表的各种命令。

如果想用标准的编辑器对一事件进行编辑，你可以用 fc 命令及一个指定的事件关联（如历史事件号）来查询该事件，此时，系统所使用的编辑器是 Shell 为 fc 命令指定的缺省编辑器。下面的例子将用标准编辑器编辑历史事件号为 4 的事件 cd reports，然后执行之。

```
$ fc 4
```

你可以也可以一次选择多个命令进行编辑并执行。通过先输入第一个命令的标识符，然后输入最后一个命令的标符来选择多个命令。这一标识符可以是命令的历史事件号，也可以是命令的初始字符或字符串。在下面的例子中，用

历史事件号来选择在事件号在 2 与 4 之间的一系列命令，然后编辑并运行它们。另一例子采用历史事件列表中初始字符或字符串来作标识符。

```
$ fc 2 4
$ fc vi c
```

fc 使用缺省的编辑器，这一缺省编辑器被 FCEDIT 某些特殊变量所指定。通常这一编辑器为 vi 编辑器。但是，如果你想使用 Emacs 编辑器， 在你使用 fc 命令时，你可以用-e 选项和 emacs 选项。下面的例子将用 Emacs 编辑器对事件号为 4 的事件 cd reports 进行编辑，然后运行。

```
$ fc -e emacs 4
```

### 15.3.2 配置 History: HISTFILE 及 HISTSIZE

你系统所保存的事件数被保存在一个特定的系统变量中，这个变量就是 HISTSIZE。这个变量的缺省值通常被设置为 500。你可以非常简单方便地改变它的值，并赋之以新的值。在下面的例子中，用户重新设置变量 HISTSIZE 的值为 10。

```
$ HISTSIZE=10
```

实际上，历史事件被保存在一个文件中，而这个文件名由变量 HISTFILE 指定。通常这个文件的缺省名是.bash_history。但是，你可以通过给变量 HISTFILE 赋值，来指定新的文件名。在下面的例子中，显示出了变量 HISTFILE 的值，然后，新的文件名 newhist 被赋值给它，所有历史事件将被保存在 newhist 文件中。

```
$ echo $HISTFILE
.bash_history
```

```
$ HISTFILE=“ newhist”
$ echo $HISTFILE
newhist
```

## 15.4 别名

通过 alias 命令，你可以给一个命令创建一个新的名字。alias 命令象一个宏，对其代表的命令进行替换。它不是逐字对命令名进行替换，而是仅仅对命令取另外一个名字。

alias 命令由关键词 alias 开头，然后是要替换命令的新的名字，其后是等号（=）及将要替换的命令，在等号之间不能有空格。下面的例子中，list 就是 ls 命令的一个别名。

```
$ alias list=ls
$ ls
mydata today
$ list
mydata today
$
```

你也可以用一个别名对一个命令或其选项进行替换。但是，你必须用一对单引号把它们括起来。所有包括空格的别名必须用单引号把它们括起来。在下面的例子中，lss 是带-s 选项 ls 命令的别名，lsa 是带-F 选项的 ls 命令的别名。带-s 选项的 ls 命令将列出文件及它们的文件占的块大小；带-F 选项的 ls 命令

将在目录名前加一“/”。需要主要的是，对命令和其选项要单引号把它们括起来。

```
$ alias lss='ls -l'
$ lss
mydata 14   today 6   reports 1
$ alias lsa='ls -F'
$ lsa
mydata today /reports
$
```

你也许经常对一个带参数选项的命令取别名。如果你经常执行带有复杂参数组合的命令，你可能想用简单的别名来代替。例如：假如你常常想列出所有你的源程序和目标代码文件，假设这些文件分别以.c 和.o 为扩展名，那么你需要用带参数的 ls 命令（此时这个参数是特定的字符串*.[co]），于是，你可以用别名给这个命令取别名。下面给出的例子中，用 ls c 给带参数*.[co] 的 ls 命令取别名。

```
$ alias lsc='ls *.[co] '
$ lsc
main.c   main.o   lib.c   lib.o
```

你也可以用一个 Shell 命令名给一个命令行取别名。这在你要执行带特定选项的 Shell 命令时非常有用。例如带-i 选项的 rm、cp 及 mv 命令通常用来保证不覆盖已存在的同名的文件。如果你在使用上述命令的过程中，希望系统给予相应的提示来确认你是否想覆盖已有的文件，你可以给它们取别名去包含这一选项。例如：下面的命令就是用带-i 选项的 rm、cp 及 mv 命令取别名为 rm、cp

及 mv 的。

```
$ alias rm=' rm -i'
$ alias mv=' mv -i'
$ alias cp=' cp -i'
```

不带任何参数的 alias 命令将列出所有当前已经起作用的别名及它们所代表的命令。你可以用 unalias 命令去删除一个别名。在下面的例子中，用户列出了当前所有的别名，然后用 unalias 命令删除别名 lsa。

```
$ alias
lsa=ls -F
list=ls
rm=rm -i
$ unalias lsa
```

## 15.5 控制 Shell 的运行方式

BASH Shell 有一些特殊变量，能让你控制 Shell 以不同的方式工作。例如：变量 noclobber 能防止在重定向输出时意外地覆盖一个文件。你可以通过 set 命令来使 noclobber 特殊变量有效或无效。set 命令有两个参数：一个选项来指定变量开（on）或关（off），另一变量是特殊变量的变量名。要使一特殊变量开（有效），用-o 选项，要使其关（无效），用+o 选项。

例如：

```
$ set -o feature// 使 feature 变量开
```

```
$ set +o feature          // 使 feature 变量关
```

三个最常用的 Shell 特殊变量变量有：ignoreeof、noclobber 及 noglob。表 15-2 列出了 set 命令的用法及几个 Shell 特殊变量的作用。

## 15.5.1ignoreeof

设置了 ignoreeof 变量后，你将不能使用 CTRL-d 来退出 Shell。CTRL-d 不仅被用来退出 Shell，而且可以用来终止用户直接输往标准输出上的输入。该操作经常在 MailX 程序和一些 Shell 实用命令中使用，例如 cat 实用命令。你可能在上述实用程序操作中，非常容易误操作而意外地退出 Shell。ignoreeof 特殊变量正是用来防止这些意外的退出。在下面的例子中，用带-o 选项的 set 命令来使 ignoreeof 特殊变量有效(on)，这时，用户只能用 logout 或 exit 命令退出 Shell。

```
$ set -o ignoreeof
$ ctrl-d
Use exit to logout
$
```

## 15.5.2noclobber

设置了 noclobber 变量后，可以使在重定向输出时保护已存在的文件。如果你用重定向把输出保存到一个已经存在的文件中时，noclobber 这一特殊变量可以防止文件在进行标准输出时被意外覆盖，原文件将被受到保护。这种情况有可能发生在把重定向输出的内容保存到你输入的文件名中，而你给出的这

个文件名可能已经存在了。因此，noclobber 特殊变量可以防止你意外地覆盖一个你已经存在的原文件。下面的例子中，用户设置 noclobber 特殊变量为有效（on），在重定向时，用户试图去覆盖系统已经存在的文件 myfile，此时，系统将返回一错误信息。

```
$ set -o noclobber
$ cat preface>myfile
myfile: file exists
$
```

你可能觉得你的确是想在重定向输出时覆盖这个文件，系统的这一提示将浪费你的时间，这时，你可以在重定向操作算子后面加一惊叹号（!）。于是，系统将不考虑 noclobber 变量的有效性而直接把重定向输出的内容输出至指定的文件。

```
$ cat preface >! myfile
```

### 15.5.3noglob

设置了 noglob 变量后，Shell 将不扩展文件名中一些特殊的字符或字符串。例如：字符“*”、“?”、“[”、“]”及“~”等字符将不再展开多义性文件名与一文件名相匹配。如果你在文件名中使用了这些特殊字符，这个特殊变量或许就对你有用。下面的例子中，如果用户希望列出结尾为“?”的文件名 answer?，可通过如下步骤：首先，用户使 noglob 变量为无效（关），然后再列出文件名。可以看到，目前命令行上的问号（“?”）被认为是文件名中的一个字符，而不是被看作一个特殊的字符（代表任意一字符）。

```
$ set -o noglob
```

```
$ ls answer?
answer?
```

## 15.6 环境变量与子 Shell：export 命令

当你登陆到你的帐号后，Linux 系统将创建你的用户 Shell。在此 shell 中，你可以使用 Shell 命令或申明变量。你也可以创建并运行 Shell 脚本程序。然而，当你运行一个 Shell 脚本程序时，系统将创建一个子 Shell。此时，系统中将有两个 Shell，一个是你登陆时系统的缺省 Shell，另一个是系统为运行脚本程序创建的 Shell。在脚本程序 Shell 中，你可以执行有自己 Shell 的另一脚本 Shell。当一个脚本程序运行完毕，它的脚本 Shell 将终止，你可以返回到你执行该脚本之前的 Shell。从这种意义上来说，你可以有许多 Shell，每个 Shell 都“寄生”在某 Shell 之上（称之为父 Shell）。

在一个子 Shell 内定义的变量只在该子 Shell 内有效。如果你在一 Shell 脚本程序中定义了一变量，当该脚本程序运行时，你定义的变量只是该脚本 Shell 内的一局部变量。其它的 Shell 不能引用它。从这种意义上来说，变量是隐藏在它的 Shell 之内的。

你可以在如下三种主要的 Shell 中定义环境变量，即 BASH Shell、PDKSH Shell 及 TCSH Shell。但是，BASH Shell 中实现环境变量的机制与 TCSH Shel 有很大的不同。在 BASH Shell 中，环境变量被输出的。也就是说。环境变量在每个子 Shell 中都有一个拷贝。例如，如果 myfile 这个变量被输出，那么这个变量在每个子 Shell 中都会为你自动定义。而在 TCSH Shell 中，环境变量只

被定义一次，一旦被定义，在任何子 Shell 中都可以直接引用。

在 BASH Shell 中，一个环境变量可以被认为是一个 Added capabilities 的常规变量。要使其变量值可以改变，你可以使用 export 命令对已定义的变量进行变量输出。export 命令将使系统在创建每一个新的 Shell 时定义那个变量的一个拷贝。每个新的 Shell 将有其自己的环境变量的拷贝。这个过程常称之为变量输出。

在下面的例子中，变量 myfile 是在 dispfile 脚本程序中定义的。然后用 export 命令使其成为一环境变量。变量 myfile 将可以输出至任何子 Shell，例如当执行 printfile 脚本程序时产生的子 Shell。

dispfile 脚本程序清单：

```
/************** begin dispfile ***********/
myfile="List"
export myfile
echo "Displaying $myfile"
pr -t -n $myfile
printfile
/************** end dispfile ***********/
```

printfile 脚本程序清单：

```
/************** begin printfile ***********/
echo "Printing $myfile"
lpr $myfile&
```

```
/************** end printfile ***********/

$ dispfile
Displaying List
1 screen
2 modem
3 paper
Printing List
$
```

**把环境变量的输出理解为全局变量是一种误解。一个新的** Shell **绝不会改变其** Shell **以外的变量的值。但是，这个变量的值将被传递给这个新创建的** Shell**。你可以将输出变量理解为将变量的值输出到一个** Shell**，而不是将它们本身。对那些比较熟悉结构化编程的读者来说，可以把输出变量理解为“值引用”。**

## 15.7 用特殊 Shell 变量设置登陆 Shell

**前面已经指出，在你登陆进入你的帐号，**Linux **系统将为你创建** Shell**。这个** Shell **被称为你的登陆** Shell **或用户** Shell**。当你执行** Shell **程序脚本时，系统将创建用户** Shell **的子** Shell**。你可以在你的用户** Shell **下定义变量，你也可以定义环境变量，而这些环境变量可以被在运行** Shell **程序脚本时创建的任何子** Shell **所引用。**

Linux **有一些特殊** Shell **变量，这些变量允许你配置你的用户** Shell**。很多这**

些特殊 Shell 变量在你登陆时系统已经为你定义好，但是，有些变量你可以自己再定义。

这些特殊变量名是系统的一些保留关键字。你不能用这些关键字作为你自己的变量名。为使这些特殊变量容易识别，这些特殊 Shell 变量都是由一些大写字母组成的字符串。例如：关键字 HOME 是系统用来定义名为 HOME 的特殊变量的，它是一个特殊环境变量，用来保存用户的主目录路径名。另一方面，前面的章节中介绍的关键字 noclobber 用来定义 noclobber 环境变量，这个特殊变量可以防止在重定向输出覆盖已有的文件。

许多特殊变量在你登陆你的用户时被系统自动赋了初始值，但是，如果需要，用户可以改变这些特殊变量的值。但是，有些特殊变量的值是不能被重新赋值的。例如，保存用户主目录路径名的特殊环境变量 HOME 是用户不能给重新赋值的。一些命令，例如 cd 命令就是通过保存用户主目录的特殊变量 HOME 来进行目录定位的。一些常用的特殊环境变量将在本节中详细介绍。

另外一些系统定义并赋以一定的初始值的特殊变量是用户可以任意改变的。你可以重新给它们定义一个新的值。例如，变量 PATH 是系统定义的，在登陆时被赋以一初始值，这一初始值包括了一些命令所在的目录路径。当你执行某个命令时，Shell 将自动搜索这些路径。你可以在 PATH 变量中增加新的路径，在执行命令时将对这些新的路径进行搜索

还有一些特殊变量，用户登陆系统后并不给它们赋值。这些变量通常是一些可选择项。例如，变量 EXINIT 允许你为 vi 编辑器进行设置选项。在每次登陆系统后，你必须重新为这些变量定义值。从这种意义上说，这些变量可称之为“用户级定义”（user-defined）特殊变量。

这三类特殊变量没有严格、权威的分类与定义。本书根据它们特点，分别称这类特殊变量为：用户不能改变其值的“系统自赋值”（system-determined）特殊变量、用户可以重新定义一新值的“可重定义”（redefinable）变量及用户必须自己赋值的“用户级定义”（user-defined）特殊变量。

你可以用 env 命令列出当前定义的特殊变量及其值。env 命令与 set 命令很相似，但 env 命令仅列出系统特殊变量。

```
$ env
USERNAME=chris
ENV=/home/chris/.bashrc
HISTSIZE=1000
HISTFILE=/home/chris/.bash_history
HISTFILESIZE=1000
HOSTNAME=garnet
LOGNAME=chris
HISTFILESIZE=1000
CDPATH=:$HOME/letters:$HOME/oldletters
MAIL=/var/spool/mail/chris
WWW_HOME=file:/usr/doc/calderadoc-0.80-1/Caldera_Info
TERM=linux
HOSTTYPE=i386
PATH=/sbin:/bin:/usr/sbin:/usr/bin:/usr/X11R6/bin:/home/chris/bin:
HOME=/home/chris
SHELL=/bin/bash
PS1= [＼u@＼h　＼w] ＼$
```

```
PS2=>
MAILCHECK=10000
MAILPATH=/home/mail/chris:/home/chris/projmsgs
CRPATH=/usr/lib/CRISPlite/macros
OSTYPE=Linux
NNTPSERVER=nntp.ix.netcom.com
EXINIT='set nu ai'
TZ=PST5PDT
SHLVL=1
_=/usr/bin/env
```

你可以用称为初始化文件的特殊脚本语言去对“可重定义”（redefinable）变量及“用户级定义”（user-defined）变量进行自动定义。初始化文件是一个特殊的 Shell 脚本程序，只要用户进入这个特定的 Shell，它将自动运行。你可以对这个初始化文件进行编辑，给一些特殊变量进行定义与赋值。当你进入 Shell，初始化文件将被执行，并对一些特殊变量进行定义与赋值，从而有效地给这些特殊变量赋予你所定义的值。例如，在 BASH Shell 中，其初始化文件是.profile，当你每次登陆进入 Linux 时，该文件都被自动运行。该文件包含了一些特殊变量的定义与赋值。因为.profile 文件仅仅是一种 Shell 脚本程序，因此，你可以用任何一种编辑器对其进行编辑，例如 vi 编辑器。如果需要，你可以改变那些特殊变量的值。

在 BASH Shell 中，所有“可重定义变量”（redefinable）及“用户级定义变量”（user-defined）是为环境变量而设置的。当你定义或重定义一特殊变量时，你必须对其进行输出（export），以使其成为环境变量。这意味着你对任

何特殊变量的改变都必须用 export 变量进行输出。在登陆初始化文件.profile 文件的结尾，你会看到通常有一 export 命令来对该文件中定义的所有特殊变量进行输出。

### 15.7.1“系统自赋值”（system-determined）特殊变量

在这里，将介绍三个常见的系统“系统自赋值”（system-determined）特殊变量，它们是：HOME、LOGNAME 及 TZ。当你一登陆，系统就自动定义它们，你可以访问它们的值，但不能改变它们。它们都是环境变量，任何的子 Shell 都可以访问它们的值。

HOME

HOME 变量保存你主目录的路径名。这个主路径是系统管理员在创建你的用户时由系统管理员指定的。当你登陆进入你的帐号时，系统将自动把主目录路径名保存到 HOME 变量中。下面的例子中，用 echo 命令列出变量 HOME 的值。

```
$echo $HOME
/home/chris
```

当你需要指定绝对路径时，我们经常用到 HOME 变量。在下面的例子中，用保存了主目录名的 HOME 变量来指定 reports 的绝对路径名。

```
$ ls $HOME/reports
```

LOGNAME

LOGNAME 变量保存你的登陆名，而不是路径名。如果用户以用户名 chris 登陆，系统将以字符串“chris”作为变量 LOGNAME 的值。在某些情况下，你

需要使用你的登陆名，这时，你可以使用 LOGNAME 变量。下面的例子将介绍 LOGNAME 变量的使用。我们知道，用户的 mail 信箱的目录名与其登陆用户名相同，用户可以直接使用他的登陆名或简单方便地引用 LOGNAME 变量的值定位 mail 目录。如果有时用户决定要改变其登陆名，那么，他们更愿意使用 LOGNAME 变量。

```
$ ls /usr/mail/$LOGNAME
TZ
```

TZ 变量用来指定系统所使用的时区。它是在你登陆进入系统时由系统设置的。在 TZ 变量中，用三个字段来显示其值。前三个字母显示当地时区，下一个字母是当地时区与格尼威制标准时间（Greenwich mean time）的时差数，最后的三个字母是当地的夏令时（Daylight Saving）时区。下面的例子中，TZ 变量保存的是太平洋标准时间（Pacific standard time），它和格尼威制标准时间的时差是八个小时，使用的是太平洋夏令时。

```
$ echo $TZ
PST8PDT
```

### 15.7.2BASH Shell 可重定义特殊变量

可重定义变量保存如下一些信息：Linux 命令所在的路径名、mail 信箱中文件的路径名以及你的命令提示符等。你可以通过简单的赋值操作来修改上述任何变量的值。另外一些常用的可重定义的特殊变量有：SHELL、PATH、PS1、PS2、及 MAIL 等。SHELL 变量保存你所登陆后使用的 Shell 类别（包括路径的程序名）。PATH 变量保存执行 Linux 命令时要搜索的目录表。PS1 和 PS2

变量保存命令提示符的值。MAIL 变量用来保存你的邮件信箱中文件的路径名。

SHELL

在 Linux 系统中，当你登陆时，系统将启动 BASH Shell、PDKSH Shell 及 TCSH Shell 这三种 Shell 中的任何一种。而 SHELL 变量将保存系统启动的 Shell 名（包括目录路径名），也就是你的缺省 Shell。Shell 程序放置在/bin 目录下。下面是不同 Shell 程序所在的目录路径列表：

BASH Shell：/bin/bash

PDKSH Shell：/bin/pdksh

TCSH Shell：/bin/tcsh

在下面的例子显示出 SHELL 变量的值。

```
$ echo $SHELL
/bin/bash
```

PATH

PATH 变量保存一系列由“：”号分开的目录路径。每次要执行一个命令时，系统将对在 PATH 变量中列出的路径名逐一进行搜索，以找到要执行的命令。例如，cp 命令放置在 linux 系统中的/usr/bin 路径下，这个目录路径是 PATH 变量中列出的目录路径之一。在每次执行 cp 命令之前，系统将搜索这一路径，从而找到 cp 应用程序。系统会给 PATH 变量定义一初始值。在 Linux 系统中，PATH 变量的初始路径名为 usr/bin 和/usr/sbin。

Shell 可执行任何可执行文件，包括你开发或创建的应用程序或脚本程序。因此，PATH 变量也可以搜索你的工作目录。如果你想在你的工作目录下运行你自己的脚本程序或应用程序，Shell 也可以搜索到它们。

在 PATH 变量保存的字符串中，路径和路径之间不能有空格。如果该字符串中仅仅只有一个“：”号而没有指定特定的路径，那么，系统将仅搜索你的工作目录。在整个字符串的结尾加一个“：”号表示包括你的工作目录。例如：字符串/usr/bin:/usr/sbin:表示将要搜索三个目录，即目录/usr/bin、/usr/sbin 及你当前的工作目录。

```
$ echo $PATH
/usr/bin:usr/sbin:
```

如果需要，你可以任意加入新的路径至 PATH 变量中。如果你用 Shell 脚本程序创建了几个你自己的 Linux 命令，那么，上述特性就非常实用。你可以创建一个目录，并把上述新的 Shell 脚本命令放到这个目录，然后把该目录加到 PATH 变量中，于是，无论你在何目录下，你都可以执行上述 Shell 脚本命令。因为在 PATH 变量中已包含了脚本命令所在目录字符串，当你发出执行一个脚本命令的时候，系统每次都对该目录进行搜索。

你可以用变量赋值的方式往 PATH 变量的路径列表中增加一个新的路径，并且可以在你的 Shell 下直接执行这一赋值过程。看下面的例子：用户 chris 向 PATH 变量中增加了一新的名为 mybin 的路径。尽管你可以仔细地往 PATH 变量中键入所有路径名列表及新的路径名来完成这一赋值过程，但你可以用 $PATH 给 PATH 变量赋值，然后加入新的路径名。下面的例子中，增加的新的路径名中的 HOME 的值用来代表用户的主目录。必须注意的是在两个“：”号之间没有任何字符串，表示要搜索工作目录。

```
$PATH=$PATH:$HOME/mybin
$export PATH
$echo $PATH
```

/usr/bin:/usr/sbin::/home/chris/mybin

如果你仅在登陆后才往 PATH 变量中增加一个目录，那么，这个目录仅仅是在你的登陆期间被增加。当你退出用户，然后重新登陆，登陆初始化文件.bash_profile 将重新用该文件原来设置的 PATH 变量的值来初始化 PATH 变量（稍后，我们在本章将详细介绍.bash_profile 文件）。如果你想保存你新增加的路径，你必须编辑你的.bash_profile 文件，找到给 PATH 变量赋值的地方，然后只需往 PATH 变量中插入“：”号及新的路径至路径列表中即可。

PS1 和 PS2

PS1 和 PS2 变量分别用来设置系统首选和辅助命令提示符。BASH Shell 的首选命令提示符是符号“$”，你可以用新的字符（窜）给 PS1 变量赋新的值来改变命令提示符。下面的例子中，Shell 命令提示符由符号“$”变成了符号“->”。

```
$ PS1="->"
->export PS1
->
```

你可以把命令提示符设置成任何你想设置的符号，包括字符串。如下例：

```
$ PS1="Please enter a command:  "
Please enter a command: export PS1
Please enter a command: ls
mydata /reports
Please enter a command:
```

PS2 变量用来设置辅助命令提示符，用来当一个命令较长，一行无法完成或没有完成时，系统在下一行给用户的命令提示符。缺省的辅助命令提示符是

“>”。当一行无法完成或没有完成时，后续增加的命令行将用辅助命令提示符而不是首选命令提示符。你可以通过上述改变首选命令提示符的方法来改变辅助命令提示符的值。例如：

```
$ PS2="@"
```

与 TCSH Shell 一样，BASH Shell 提供一组预定义符，以便配置你的命令提示符。你可以用时间、你的用户名或者你的当前目录路径名，甚至可以用你当前将要输入命令的历史事件号作为你的命令提示符或提示符的一部分。每个预定义符以“\”符号开头。“\w”代表当前的工作目录，“\t”代表当前的时间，“\u”代表你的用户名。“\!”将显示下一命令的历史事件号。下面的例子中，用户在命令提示符中增加了当前工作目录。

```
$ PS1="\w $"
/home/dylan $
```

这些预定义符必须是有双引号括起来的一字符串。如果没有双引号，那么这些预定义符将不会被赋值给 PS1 和 PS2 变量，而是把字符串本身的值作为命令提示符。例如命令行 PS1=\w 将把字符串“\w”设置为命令提示符，而不是把当前工作目录设置为命令提示符。下面的例子将时间和历史命令号一起作为新的命令提示符。

```
$ PS1="\t \! ->"
```

下面是一些用来配置系统命令提示符的一组预定义符列表：

\!当前历史事件号

\$ 除超级用户外，所有的其它用户都用“$”作命令提示符，而超级用户用“#”作命令提示符

\d　　当前日期
\s　　当前 Shell 类别
\t　　当前时间
\u　　当前用户名
\w　　当前工作目录

MAIL、MAILCHECK 及 MAILPATH

Linux 系统把发送给你的邮件保存在用户邮件信箱文件里，MAIL 变量就是用来保存用户邮件信箱文件名的。当你执行 Mailx 实用程序时，等待你阅读的邮件就从这个文件中取出。邮件信箱文件及邮件实用程序在本书第 8 章有详细介绍。尽管你可以改变 MAIL 变量的值，但一般很少或没有必要改变它。Linux 系统需要这个目录路径名去寻找用户邮件信箱文件。

```
$ echo $MAIL
/var/mail/chris
```

MATLCHECK 变量用来设置时间间隔。每隔这个设定的时间间隔，如果你有新邮件，系统将会通知你已经有新邮件。如果你希望系统一收新邮件就通知你，你可以减少这一时间间隔。如果你不想被它干扰，你可以延长这一时间间隔。缺省的时间间隔是 10 分钟（600s）。下面的例子中，时间间隔被设置成 20 分钟（1200s）。

```
$ MAILCHECK=1200
$ export  MAILCHECK
```

如果你想在检查另外一些邮件信箱文件的来件情况，你可以设置 MAILPATH 变量，用来保存另外一些邮件信箱文件的目录路径表。与 MAIL 和 MAILCHECK 变量不同，MAILPATH 是“用户级定义”（user-defined）特殊变量。要使用

该变量，你必须给它赋一值。你可以给它赋值任何你想核查的邮件信箱文件的目录路径名。在下面的例子中，除在 MAIL 变量中指定的邮件目录路径外，用户还指定了另外一需要检查来件情况的邮件文件目录路径名。

```
$ MAILPATH=/home/mail/$LOGNAME
$ export MAILPATH
```

### 15.7.3BASH Shell “用户级定义”（user-defined）特殊变量

BASH Shell 中用户级特殊变量保存当前终端类型、缺省的 vi 编辑器配置等信息。很多变量如 TERM 和 CDPATH 等增强了 Shell 的可操作性。另外一些是为某些特定的实用程序设定的。例如 EXINIT 变量就是用来配置 vi 和 Ex 编辑器的。BASH Shell 环境下的一些可重定义变量和用户级定义变量在表 15-2 中列出。

用户级定义特殊变量不是系统定义的。如果你要使用它们，你自己必须定义并赋值给它们。三个常见的用户级定义特殊变量有：CDPATH、TERM 和 EXINIT。CDPATH 变量保存目录路径名，以便 cd 命令方便地寻找、改变目录。TERM 变量保存当前使用的终端类型。EXINIT 变量用来配置 vi 和 Ex 编辑器命令。

CDPATH

如果 CDPATH 变量没有定义，当我们在 cd 命令后给出目录名作参数时，它仅在当前工作的目录下搜索所给出的目录名。但是，如果定义了 CDPATH 变量，执行上述命令时，将在 CDPATH 变量中所有列出的目录中去搜索。如果该目录被搜索到，运行 cd 命令后将转至该目录下。如果你正在进行一项工作，而

该工作必须不断地在不同的目录下转来转去，设置 CDPATH 变量后就会对上述工作非常有用。要转至和你现在目录差别很大的目录路径下，你必须知道该目录的绝对路径。但是，你只需简单地将要转向的目录路径的父路径增加到 CDPATH 变量中，cd 命令将在其父路径下自动搜索，为你找到你所要转向的目录，并转至该目录下。需要指出的是，你应该把你要转向的目录的父路径赋值给 CDPATH 变量，并不是该目录本身。

在下面的例子中，CDPATH 被增加了搜索路径 /home/chris/letters。/home/chris/letters 是 thankyou 的父目录。当执行带参数 thankyou 的 cd 命令时，CDPATH 中的目录将自动搜索，包括 letters 目录，于是，thankyou 目录将被搜索到。

```
$ CDPATH=$CDPATH:/home/chris/letters
$ export CDPATH
$ echo $CDPATH
:/home/chris/letters
$ cd thankyou
$ pwd
/home/chris/letters/thankyou
$
```

当然，你可以编辑 .bash_profile 文件，以便永久地往 CDPATH 变量中增加一些目录路径名。本书建议在向 CDPATH 变量中增加新的目录路径名的过程中，用 HOME 变量去指定各目录路径的主目录路径部分。这是因为系统管理员在对文件系统进行重新组织的过程中，你的主目录路径有可能改变。HOME 变量始终保存你目前的主目录路径名。在下面的例子中，目录路径名

/home/chris/letters 是用 $HOME/letters 指定的。

```
$ CDPATH=$CDPATH:$HOME/letters
$ export CDPATH
$ echo $CDPATH
:/home/chris/letters
```

TERM

TERM 变量保存目前你使用的终端的终端名。如果你是从一个终端登陆的，系统将询问你的终端名，你所输入的终端名将被保存在变量 TERM 中。一些应用程序如标准编辑器，将通过 TERM 变量来查明你所使用的终端类别，这样，系统才正确映射键盘输入的命令与屏幕显示。下面的命令将显示你的终端类别：

```
$ echo $TERM
tvi925
```

如果你想改变你的终端类别，你可以给 TERM 变量赋值另一终端名来改变它。在下面的例子中，TERM 变量被赋值以 vt100 终端类型，然后，用输出变量 TERM，以使在使用 Shell 的过程中都能访问它。

```
$ TERM=vt100
$ export TERM
$
```

EXINIT

EXINIT 变量可用来保存一些编辑器的命令选项，这些命令选项可以用来配置 vi 和 Ex 编辑器的编辑环境与编辑命令。当你启动这些编辑器时，EXINIT 变量中保存、设置的命令将被执行。EXINIT 变量中保存的命令通常用来指定编辑器的一些编辑特性，如是否显示行号、是否进行行缩进等。这些都会在第 17 章

中详细讨论。

在下面的例子中，EXINIT 变量被赋值给一个 set 命令，该 set 命令用来设置编辑器显示行号及自动缩进特性。请注意，两个命令可以缩写并合并为一个字符串。

```
$ EXINIT='set nu ai'
$ export EXINIT
```

### 15.7.4BASH Shell 登陆初始化文件：.profile

.profile 文件是 BASH Shell 的登陆初始化文件，该文件名也可以是.bash_profile。它是一个脚本程序，任何一个用户登陆进入 BASH Shell 后，它都将被自动执行。该程序用一些常用 Shell 命令去定义一些用来管理 BASH Shell 的特殊环境变量。它们可能是一些系统定义类特殊环境变量的重定义或一些用户级定义类特殊变量。例如，当你登陆进入系统，用户 Shell 必须知道 Linux 系统的一些系统命令所在的目录。Linux 系统将根据 PATH 变量所指定的目录路径来进行搜索。但是，PATH 变量必须首先被赋与指定的目录路径名。在.bash_profile 中，有一个赋值过程，它正是用来完成上述赋值操作的。由于它是在.bash_profile 中，这一赋值过程在每次用户登陆时都被自动执行。

也有一些特殊变量需要用 export 命令进行输出。以使它们在你创建的任何子 Shell 中都能访问其值。你可以用一个 export 命令以要输出的变量列表作为参数同时输出几个变量。通常在.bash_profile 文件的结尾用一个 export 命令输出文件中定义的所有变量。如果在这个列表中有一个变量遗漏，那么你将无法访问该变量。请大家注意下面.profile 文件结尾处的 export 命令。

下面列出的是一个标准的.bash_profile 文件的拷贝，这个文件会在你的用户创建时自动生成。请注意 PATH 变量的赋值情况（通过 HOME 变量给其赋值）。PATH 变量和 HOME 变量都是系统定义的特殊变量，它们已被系统所定义。PATH 变量中保存目录的路径名，在执行任何输入的命令时都要对这些目录进行搜索，而 HOME 变量保存你的主目录路径名。赋值语句 PATH=$PATH:$HOME/bin 起到了在重定义的过程中包含主目录中 bin 的目录的作用。因此，在执行任何命令的过程中，也将搜索你的 bin 目录，包括你自己定义的脚本程序或应用程序等。必须注意的是，PATH 必须用 export 命令进行输出，以便你创建的任何子 Shell 都能访问。如果你想对你的主目录也进行搜索，你可以在用 vi 或 Emacs 编辑器修改.bash_profile 文件中的如下行：PATH=$PATH:$HOME＼bin:$HOME，在上述行的末尾增加:$HOME。事实上，你可以在该行中增加任何你需要搜索的目录名。

.profile 文件清单

```
#.bash_profile

#Get the aliases and functions
if [-f~/.bashrc] ; then
   .~/.bashrc
fi

#User-specific environment and startup programs
```

```
PATH=$PATH:$HOME/bin
ENV=$HOME/.bashrc
USERNAME=""

export USERNAME ENV PATH
```

Linux 系也有其自己的 profile 文件，任何用户登陆进入用户时，它都会执行。这个系统初始化文件就是 profile，在/etc 目录下可以找到该文件（/etc/profile），里面包含了系统提供给每个用户的一些特殊变量的定义，该文件的清单在下面列出。请注意，PATH 变量重定义时包含了/usr/X11R6/bin 目录。该目录路径包含了 X-Winodw 中用户在使用 Caldera Desktop 时要执行的的一些命令。HISTFILE 也被重定义，它被赋值给一个较大的历史事件数。

/etc/profile 文件清单

```
# /etc/profile

# Systemwide environment and startup programs
# Functions and aliases go in /etc/bashrc

PATH="$PATH:/usr/X11R6/bin"
PS1="[\u@\h\W] \\$"

ulimit -c 1000000
umask 002
```

```
HOSTNAME='/bin/hostname'
HISTSIZE=1000
HISTFILESIZE=1000
#Default page for the arena browser
WWW_HOME=file:/usr/doc/Calderadoc-0.80-1/Caldera_Info
#Default path for CRiSplite
CRPATH=/usr/lib/ CRiSplite/Macros

export PATH PS1 HOSTNMAE HISTSIZE HISTFILESIZE WWW_HOME
CRPATH
if ["$TERM"=consol]
then
MIMICOM="-l -m -con -tmc"; export MINICOM
fi
```

**初始化文件，**profile **是一个文本文件，你可以用任何一个文本编辑器编辑该文件。通过编辑**.bash_profile **文件，你可以用编辑命令把一个新的目录路径名插入到目录路径名列表中去，从而非常方便地给** PATH **变量增加新的搜索路径。你也可以增加新的变量定义。但是，如果你定义了变量，必须记住：要在**export**命令的参数列表中增加你所定义的变量名。例如，如果你的**.bash_profile**文件中没有变量** EXINIT **的任何定义，那么，你可以编辑该文件，并增加新的赋值语句给** EXINIT **变量赋值。赋值语句** EXINIT='set nu ai'**将设置** vi **编辑器带行号及自动缩进**(indentation)**。然后，你需将** EXINIT **变量增加到** export **命令的变量参数列表中。当**.bash_profile **再一次被运行时，变量** EXINIT **将被赋值于命令行** set nu ai**。当** vi **编辑器被执行时，变量** EXINIT **被赋值的命令行将被执**

行，从而将编辑器 vi 自动设置成带行号，并进行自动缩进。

在下面的例子中，用户的.bash_ profile 文件已经被修改，包含了变量 EXINIT 的定义，并重新定义了变量 CDPATH、PS1 及 HISTSIZE。变量 HISTSIZE 在重定义时，其保存的历史事件号减少了，由系统.profile 文件中定义的 1000 减少到 30；对特殊变量 PS1 的重定义，使系统提示符包括了当前的工作目录。你对文件.bash_profile 中特殊变量的任何改变都将覆盖先前已在.profile 系统文件中定义的特殊变量的值。然后，这些特殊变量需用 export 命令进行变量输出。

.profile 文件清单

```
# .bash_profile

# Get the aliases and functions
if  [-f ~/.bashrc] ; then
    .~ /.bashrc
fi

#User-specific environment and startup programs

PATH=$PATH:$HOME/bin:$HOME
ENV=$HOME/.bashrc
USERNAME=""
CDPATH=$CDPATH:$HOME/bin:$HOME
HISTSIZE=30
EXINIT='set nu ai'
```

PS1="＼w　＼$"

export USERNAME ENV PATH CDPATH HISTSIZE EXINIT PS1

尽管.profile 文件在你每次登陆进入系统时都会自动执行，但它不会在你每次修改就自动运行。因此，登陆初始化文件.profile 仅仅在你每次登陆时才会自动运行。如果你想使你的修改起作用而又不想在每次修改后退出系统进行再登陆，那么，你可以用“.”命令运行.profile 文件。.profile 文件是一个 Shell 脚本程序，和其它 Shell 脚本程序一样，在“.”命令后键入.profile 就可以运行该文件。

$ .profile

## 15.7.5BASH Shell 初始化文件：.bashrc

.bashrc 文件是一个初始化文件，它在你每次进入 BASH Shell 或创建一新的子 Shell 时都会被执行。如果 BASH Shell 是你的登陆 Shell，那么，在你登陆时，.bashrc 文件会就会与文件.bash_login 文件同时运行。如果你在另一种 Shell 下进入 BASH Shell，则.bashrc 文件会自动运行，文件中的一些变量与别名就会被定义。

实际上，.bashrc Shell 初始化文件在你每次创建一个 Shell 时，都会被执行。例如，当你运行一个脚本 Shell 程序时。换句话说，每次创建一个子 Shell，.bashrc 文件就会被执行。这时，在.bashrc Shell 初始化文件中任何定义的局部变量或别名，在用 export 命令进行变量输出后，都会起作用。.bashrc 文件中通常是一些别名与特征变量的定义，而这些特征变量是用来开启某些 Shell 特性的。

别名与特征变量是在 Shell 内局部定义的，但.bashrc 文件将在创建一新的子 Shell 中定义它们。因此，.bashrc 文件中通常是给 rm、cp 及 mv 等命令定义的别名。在下面的例子是一个具有许多标准定义的.bashrc 例子文件。

.bashrc 文件清单

```
# Source global definitions
if [ -f /etc/bashrc ] ; then
./etc/bashrc
fi

set -o ignoreeof
set -o noclobber

alias rm 'rm -i'
alias mv 'mv -i'
alias cp 'cp -i'
```

Linux 系统通常有一个名为.bashrc 的系统文件，所有的用户在进入 Shell 时都要运行该文件。文件中有一些特定的全局别名与特征变量，这些别名与变量对任何进入 BASH Shell 的用户都是必不可少的。该文件位于/etc 目录下（/etc/.bashrc）。用户自己的.bashrc 文件位于用户主目录下，该文件中包含很多必要的初始化命令。上面例子中的./etc/bashrc 文件正是用来做该项工作的。你可以在你自己的.bashrc 文件中定义你自己的命令或定义。如果你修改

了.bashrc 文件，且希望它们能在当前的登陆环境下起作用，你可以用“.”命令或 source 命令重新执行该文件。

```
$ source .bashrc
```

### 15.7.6BASH Shell 退出初始化文件：.bash_logout

.bash_logout 也是一个初始化文件，当用户退出系统时，将执行该文件。当你退出时，你可以指定系统执行你想做的操作。.bash_logout 文件不是用来定义变量或别名的，而是通常由一些 Shell 命令组成的关机程序，也就是关机时你想要系统做的一些操作。通常希望系统退出时的操作是清屏，然后向用户发出祝福或结束信息。

和.bash_profile 文件一样，你可以在.bash_logout 文件中增加你自己的命令。实际上，在你的用户被创建时，.bash_logout 并不能自动建立。你需用 vi 或 Emacs 编辑器自己去创建该文件。你可以在该文件中加入退出祝福语或其它操作。在下面的例子中，用户在.bash_logout 文件中加入 clear 命令和 echo 命令。当用户退出系统时，clear 命令将清屏，然后显示“Good-bye for now”信息。

.bash_logout 文件清单

```
clear
echo "Good-bye for now"
```

## 15.8BASH Shell 程序设计

BASH Shell 具有和其它程序设计语言一样的编程能力，你可以用它设计出复杂的 Shell 程序。Shell 程序由一些 Linux 命令组成，用来完成一些特定的任务。Linux Shell 能提供许多编程工具给用户,用户可以利用这些工具来创建 Shell 脚本程序。在 Shell 程序脚本中，你可以定义变量并赋值给它们，你也可以在脚本程序中定义变量。在执行该脚本程序时，用户还可以通过交互式方式给这些变量赋值；Linux Shell 还提供了循环、条件等控制结构。其中，循环结构用来多次执行某些 Linux 命令；条件结构用来控制、选择你要执行的命令；

你还可以建立算术或比较操作表达式。所有这些程序设计语句及控制结构与其它程序设计语言中相应的语句及控制结构类似。你可以用 Shell 变量、控制结构、表达式及 Linux 命令来进行 Shell 程序设计。通常由这些组成 shell 程序的指令编写成可以单独运行的脚本程序。你可以用任何标准编辑器来创建这些脚本文件。要运行该 Shell 程序，你只需执行该脚本文件即可。你甚至可以拆分到几个脚本文件中，每个文件都可以调用其它 Shell 脚本程序。可以把变量、表达式及控制结构看作是工具，通过这些工具把多个 Linux 命令组合成一个命令。从这种意义上说，一个 Shell 程序是用户创建的新的、复杂的 Linux 命令。

### 15.8.1Shell 脚本程序：命令行及注释行

一个 Shell 脚本程序是一个由 Linux 命令组成的文本文件，你可以用任何文本编辑器来编辑它。你可以以 Shell 脚本程序的文件名作参数，用 sh 命令或“ .”

命令来执行脚本程序中的命令。你也可以设置文件的属性为“可执行性”（executable），然后在命令提示符后直接键入任何要运行的文件名去运行该脚本程序。用 chmod 命令来设置脚本程序的可执行性属性，可以把脚本程序设置成可执行文件。在用 chmod 命令修改文件的可执行性属性时，可以用绝对方式或符号方式。用符号方式“u+x”可修改文件的可执行性属性，例如命令行“chmod u+x hello”将把脚本程序 hello 设置成可执行性。

然后，你可以直接运行脚本程序 hello，如同使用一个 Linux 命令一样。你只需设置文件的可执行性属性一次。一旦你设置了它，该属性将保持，直至你改变其这一属性为止。

hello 脚本程序清单

```
echo "Hello, How are you"
$ chmod u+x hello
$ hello
Hello, how are you
```

在用绝对方式设置文件的可读、可写权限时将同时设置文件的可执行性属性。请参考本书第 7 章关于绝对方式与符号方式的详细解释与说明。简要地说，700 将把文件设置成可写、可读及可执行权限，500 将仅设置成可执行和可读权限，300 设置成可执行和可写权限，400 将只具有可执行权限。通常用户将权限设置成 700 或 500。在下面的例子中，用户用绝对方式将文件 hello 设置成可执行权限。

```
$ chmod  750 hello
$ hello
hello, how are you
```

$

通常，在脚本文件中用简短的解释来说明文件的功能、特定命令与变量的作用和目的是非常有用的。你可以用注释行来进行简短的说明。注释行可以自成一行或是文本行的一部分，注释内容前必须用符号“#”注明，但第一行除外。注释行以换行符为结束，Shell 将忽略任意行上符号“#”后的字符。Shell 脚本程序的第一行是保留行，用来识别创建的 Shell 类型(下面将会讨论到)。在下面的例子中，描述脚本程序的文件名及其功能的注释行放置在该文件的文件头。

hello 脚本程序清单

```
# The hello script says hello
echo "Hello, how are you"
```

有时希望在一种 Linux Shell 下运行为另一种 Shell 编写设计的脚本程序。假设你目前在 TCSH Shell 下，而想运行一个为 BASH Shell 编写设计的脚本程序（该脚本程序包含了一些 BASH Shell 命令），首先，你必须用 sh 命令把该 Shell 脚本程序转换成 BASH Shell 脚本程序，然后运行该脚本程序，最后在转换成 TCSH Shell。然而，你可以自动进行这一过程：在你的脚本程序的首行键入字符串“#!”，然后键入你系统的 Shell 程序的路径名。Shell 始终首先检查脚本程序的最初一个字符，以辨认该脚本程序是 BASH Shell 脚本程序还是 PDKSH Shell 脚本程序亦或 TCSH Shell 脚本程序。如果最初字符是空格，那么，该脚本程序便被假定为 BASH Shell 脚本程序或 PDKSH Shell 脚本程序；如果只是单独的“#”号（无“!”号），该脚本将被假定为 TCSH Shell 脚本程序。但是，如果“#”号后面跟着字符“!”，那么，Shell 将读取其后的 Shell 类型路径名。字符串“#!”后面始终是 Shell 类型的路径名，用来确定该 Shell 脚本程序运行的 Shell 类型。如果你目前在与某脚本程序运行环境不同的 Shell

下，当前 Shell 将读取 Shell 脚本程序中 Shell 的路径名，并转换至该 Shell 下执行此脚本程序。但此时如果其初始字符是空格或“ # ”号，当前 Shell 将无法确定该脚本程序是 BASH Shell 脚本程序还是 TCSH Shell 脚本程序。这时，该 Shell 脚本程序将仅能在它的所在的 Shell 环境下才能运行。如果你想让 Shell 辨认你的脚本程序的 Shell 类别,你必须在字符串“ #! ”后面键入你所运行的 Shell 类型。

例如，如果你在脚本程序 hello 的首行键入字符串“ #!/bin/sh ”，你就可以在 TCSH Shell 中直接运行该脚本程序。运行时，系统首先将改变到 BASH Shell 下，然后执行该脚本程序的命令，然后返回 TCSH Shell 或任意执行该脚本时的 Shell。在下面的例子中，hello 脚本程序输入了“ #!/bin/sh ”命令，然后用户可以在 TCSH Shell 中执行该脚本程序。

hello 脚本程序清单

```
#!/bin/sh

# The hello script says hello

echo "Hello, how are you"

> hello
Hello, how are you
```

## 15.9 变量与脚本程序

在 Shell 中，你可以用变量及脚本语句来创建 Shell 程序。在 Shell 脚本程序中，你可以定义变量并给它们赋值。变量定义在本书第 5 章已详细讨论过。由于变量定义在 Shell 程序设计过程中非常有用，作者在这将会简要介绍。

在 Shell 脚本程序的输入、输出操作中，变量使用的非常多。read 命令允许用户交互式地给一个变量赋值。当需要用户进行数据输入时，程序设计者通常会给用户一个提示去通知用户。另一种脚本程序数据输入方式称之为 Here 文件方式，它允许你为程序中的命令提供多行输入。该数据输入方式克服了总需要通过读取外部文件等方式来进行多行数据输入。

### 15.9.1 变量的定义与赋值：=、$、set 及 unset

在 Shell 中，当你首次使用变量的名字的时候，需要对一变量进行声明。变量的名字可以是任何一组字母组成的字符或字符串，包括下划线。变量名也可以包含数字，但数字不能作为变量的第一个字母。变量名中不能有如下字符，如“!”号、“&”字符及空格等。有些变量是 Shell 保留的，只能被 Shell 自己使用。变量名也不能包含几个英文单词，因为系统要使用空格去对命令进行句法分析，并用它分隔命令名及其参数。

你可以用赋值操作给一个变量赋值：键入变量的名，然后键入赋值操作“=”，再键入相关数据，变量将被赋值。需指出的是，赋值操作语句中不能有任何空格。任何一组字符串都可以赋值给一个变量。在下面的例子中，变量 greeting 被赋以字符串“Hello”。

```
$ greeting="Hello"
```

一旦你给一个变量赋值，你就可以使用该变量，并引用它的值。通常我们引用变量的值作为一个命令的参数。你可以在变量前面加美元字符“$”来引用变量的值。“$”字符是一个特殊操作符，它用变量的名来引用变量的值。该引用将导致命令行上的变量名被一组字符窜所代替。这样，任何在变量名前有“$”字符的地方，变量名将由变量的值所代替。

```
$ echo $greeting
Hello
```

你可以用 set 命令来列出所有已定义的变量的列表。如果你认为不再需要某变量，可用 unset 命令来删除该变量。

说明：表 15-3 列出了一些常用的 BASH Shell 命令及其命令参数与功能，如 echo、read 及 break 等。

### 15.9.2 变量值：字符串

你赋值给变量的值可能是由一组字符组成的字符串。这些字符可能是你直接输入的，也有可能是你执行一个 Linux 命令后的得到的结果。大多数情况下，你需要用单引号、双引号、反斜扛号及回引号等去引用这些变量的值。单引号、双引号、反斜扛及回引号能让用户以不同的方式引用字符串的值。回引号“`”的特殊功能在于：当执行一个 Linux 命令时，它同时把执行的结果作为命令行上的一个命令行参数。

## 字符串引用：双引号、单引号和反斜杠

尽管字符变量的值可以由任何字符串组成，但当你在字符串中包含了一些特殊字符，而这些特殊字符又是被 Shell 作为操作算子的，这时脚本程序就可能会出现问题。Shell 有一类特殊的字符，它们可以被用在命令行赋值中。星号“*”、问号“?”、括号“（”及“）”等都是一组特殊字符，这些特殊字符可以实现文件名扩展。此外，特殊字符“.”代表当前的目录，“$”代表变量的值，大于符“>”及小于符“<”是重定向操作算子，符号“&”用来执行后台命令，而符号“|”表示通过管道输出执行的结果。如果你想使用上述特殊字符作为你变量值的一部分，你必须首先引用它们。在命令行上引用一个特殊变量字符时，它不会被 Shell 解释为一个特殊字符。

单引号及双引号允许你同时引用几个特殊变量，任何在单引号及双引号内的特殊字符都会被引用。一个反斜杠只引用单个字符（其前面的字符）。如果你想把多个单词赋值给一个变量，你需用单引号去把那些用空格定界的单词括起来（你也可以用双引号），认为是把一个字符串赋值给了一个变量。当然，任何在双引号内的特殊字符也会被引用。

下面的例子用三种方法去给字符类型变量赋值。在第一个例子中，用把双引号内用空格定界的单词赋值给变量 notice。由于空格是在双引号内的，因此，它们被认为是普通的字符，而不是 Shell 用来分割命令行参数的定界符。在第二个例子中，单引号内的字符串也被赋值给变量 message，特殊字符“.”被认为是一个普通的字符。在第三个例子中，星号“*”特殊字符被括在双引号内，该字符也被认为是字符串内的一个普通字符，而 Shell 不会认为它是一个通配符。

```
$ notice="The meeting will be tomorrow"
$ echo $notice
The meeting will be tomorrow

$ message='The project is on time.'
$ echo $ message
The project is on time.

$ notice="You can get a list of files with ls *.c"
$ echo $notice
You can get a list of files with ls *.c
```

但是，双引号中的特殊字符“$”不会被赋值给变量，该特殊字符用来引用变量的值。在双引号内的“$”字符后的变量仍然将代表该变量的值，而该变量的值将成为字符串的一部分，而不是变量名成为字符串的一部分。在双引号内你可以使用该方法来引用多个变量的值。在下面的例子中，变量 winner 的值将被 notice 变量引用。

```
$ winner=dylan
$ notice="The person who won is $ winner"
$ echo $notice
The person who won is dylan
```

另一方面，如果你也不想在双引号内引用一个变量的值，你可以用单引号来把赋值的字符窜括起来。此时，单引号将忽略字符串内的任何引用，而把特

殊字符“$”仅看作为一个普通的字符。在下面的例子中，单引号忽略了其内的变量 winner 的引用。

```
$ winner=dylan
$ result='The name is the $winner variable'
$ echo $result
The name is the $winner variable
```

此时，如果用的是双引号，那么，变量 winner 的值将会被变量 result 引用。看下面的例子：字符串“$winner”被变量 winner 的值替换。

```
$ winner=dylan
$ result="The name is the $winner variable"
$ echo $result
The name is the dylan variable
```

你可以用反斜扛特殊字符“＼”来在字符串中使用任何特殊字符，包括特殊字符“$”。当你在赋值的字符串中同时包括变量值引用及普通字符“$”时，你必须用到反斜扛字符“＼”。在下面的例子中，为了把特殊字符“$”看作一个普通的美元字符，你可以在其前面输入一反斜扛字符，及用字符串“＼$”表示。同时，由于字符串是用双引号括起来的，因此，$winner 将引用变量 winner 的值。

```
$ winner=dylan
$ result="$winner won ＼$100.00"
$ echo $reuslt
dylan won $100.00
```

### 15.9.3Linux 命令的值：回引号“`”（back qutoes）

尽管你可以通过定义字符或字符串变量来引用变量的值，你还可以直接把 Linux 命令的运行结果赋值给变量。但是，如果你要把 Linux 命令的运行结果赋值给变量，你必须首先运行该命令。如果你在命令行上输入 Linux 命令，该命令首先将会被执行，它的执行结果将成为命令行的一个参数。如果把将要首先运行的 Linux 命令放置在回引号内，命令的运行结果将被赋值给变量。我们也可以把回引号看作是一个表达式，而该表达式是包含了一个将要运行的 Linux 命令及其运行的结果。

在下面的例子中，命令 ls *.c 首先将被执行，而其运行结果将被赋值给变量 listc。命令 ls *.c 用来列出所有扩展名为.c 的文件，然后，这个文件名列表将被赋值给变量 listc。

```
$ listc='ls *.c'
$ echo $listc
main.c prog.c lib.c
```

必须注意回引号与单引号之间的区别：单引号把一个 Linux 命令看作是一组字符串，而回引号将表示它是一个 Linux 命令，并会执行这个 Linux 命令。有时，你可能在想要使用回引号时无意间输入了单引号。下面的两个例子说明了两者之间的区别。对于第一个例子，赋值给变量 lscc 的值是用单引号而不是回引号，此时，“ls *.c”仅被看作是一个字符串赋值给变量 lscc。对于第二个例子，字符串“ls *.c”是用回引号括起来的，因此，该命令将会被执行。于是，所有那些以.c 为扩展名的文件名生成一个文件名列表，然后该文件名列表被赋

值给变量 lscc。

```
$ lscc='ls *.c'
$ echo $lscc
ls *.c

$ lscc='ls *.c'
$ echo $lscc
main.c prog.c
```

### 15.9.4 引用命令：单引号

有时，你可能会在一个 Linux 命令上使用单引号。在 Linux 命令上使用单引号可以使你把一个 Linux 命令赋值给一个变量，此时，你可以把该变量作为该 Linux 命令的另外一个名字。如果在命令行上输入该变量的名之前键入操作算子“$”，该变量所代表的 Linux 命令将被执行。在下面的例子中，Shell 变量被赋值以 Linux 命令 ls -F，必须注意命令上的单引号。当 Shell 变量在命令行上被赋值的时候，包含该命令的变量将成为一个命令行参数，且其包含的命令将会被执行。

```
$ lsf='ls -F'
$ $lsf
mydata /report/letters
$
```

实际上，与别名命令类似，该变量是你为 Linux 命令创建的另外一个名字。在余下的章节中可以看到，在特定的 Shell 环境下，你可以用一个特别的别名

命令来完成上述操作。

### 15.9.5 脚本输入与输出命令：echo、read 及 <<

在一个脚本程序中，你可以使用 echo 命令来输出数据，而用 read 命令来读取输入数据给变量。此外，你还可以使用 Here 文件来在脚本中指定数据，并把它们重定向给一个命令。

在脚本程序中，echo 命令可以把数据送到标准输出，这些数据可能是以字符串形式输出的字符。你将会发现，echo 命令可以输出变量的值，也可以输出常量字符串。

read 命令可以从终端上读取一行输入数据，并赋值给一个或多个变量。该命令可以使用户交互式地给变量赋值。的直至换行符的任何字符都将被读入，并把这些读入的字符赋值给变量。在 Shell 程序中，你可以一起使用 echo 命令与 read 命令，以便提示用户输入数据，输入数据后，read 命令将把读入的数据赋值给变量。在下面的 greetvar 脚本程序中，用户被提示输入数据给变量 greeting，然后，read 命令读取用户键入的数据，并赋值给变量 greeting。

greetvar 脚本程序清单：

```
echo Pease enter a greeting:
read greeting

echo "The greeting you enter was $greeting"

$ greetvar
```

```
Please enter a greeting:
Hi
The greeting you enter was Hi
$
```

当处理用户的输入时，你必须考虑到用户有可能输入 Shell 特殊字符。Linux 命令中的任何特殊字符，无论它是在脚本程序还是在其它地方，该特殊字符都将被引用，除非它们被引号括起来。如果一个变量的值是一个特殊字符，且变量的值是用$来引用的，那么，Shell 将认为是一个特殊字符，并把它用变量的值代替。在脚本程序 greetvar 程序中，$greeting 被放置在一个被引号括起来的字符串中，从而可以使用特殊字符“$”来求变量 greeting 的值。否则，如果 $greeting 所在的字符串没有被用括号括起来，那么，特殊字符的意义将被扩展。有时，你也希望那些特殊字符被扩展，例如，你想列出所有以用户输入的字符为前缀的文件时，此时，用户输入的任何特殊字符的意义都需被扩展。在下面的例子中，脚本程序中的特殊字符“*”被扩展，以显示以扩展名为.c 的所有文件。请注意$fref 没有被双引号括起来。

listfile 脚本程序清单：

```
echo Please enter a file reference:
read fref

echo The files you request are: $fref

$listfiles
```

```
Please enter a file reference:
*.c
The files you request are: calc.c lib.c main.c
```

通常情况下，Shell 脚本程序包含一系列的 Linux Shell 命令，但有时，你可能需要把输入的数据作为一个命令的参数，例如，如果你想把多行数据输入给 Shell 脚本程序中的某个命令。Here 操作可以达到上述目的。Here 操作是一个重定向操作，用来把 Shell 脚本中读入的数据重定向给一个命令。它之所以称为 Here 操作是因为重定向数据是来自脚本程序，而不是来自其它的文件。Here 操作用两个小于号“<<”来表示。“<<”操作可以认为是一个重定向操作，它用来把脚本程序中的数据输入给一个命令。“<<”操作算子放置在它要被重定向的命令之后，“<<”后的输入数据将被认为是输入给命令的数据。你可以用 CTRL-d 操作来结束数据输入。你也可以不使用 CTRL-d 而用自己指定的一个结束符来结束本次数据输入，这时，在同一行上、“<<”操作算子之后的单词将被认为是本次数据输入的结束符。该结束符可以是任何符号。所有至结束符之前的数据行都将被作为其重定向命令的输入。

在下面的例子中，一些信息被发送给用户 mark，其中，发送的信息来自于 Here 操作。本次 Here 操作使用的结束符是单词“myend”。

mailmark 脚本程序清单：

```
mail mark << myend
Did you remember the meeting
Robert
myend
```

### 15.9.6 脚本程序命令行参数

与 Linux 命令一样，一个 Shell 脚本程序也可以有其运行参数。当你启动一个脚本程序时，你可以在命令行上键入脚本程序名，然后输入运行该脚本程序的一些运行参数。你可以在脚本程序中用“$”操作算子及其在命令行上的参数序号来引用这些参数。命令行上的参数按顺序从 1 开始编号，第一个参数可以用“$1”来引用，而其第二个参数可以用“$2”来引用，依此类推。“$0”将引用该命令行上的第一个字符，即 Shell 脚本程序名本身。

所有对这些参数的引用都可以被认为是对只读变量的值的引用。如果你比较熟悉程序设计语言中的术语，那么，你可以把命令行上的参数认为是系统赋值给参数变量$1 及$9 等的值，而参数变量在脚本程序运行的过程中是只读的，你不能再给它们赋值。一旦系统给它们赋初值后，你不能再改变这些参数变量的值。从这种意义上说，参数变量更像常数，这些常数是由命令行参数确定的。除非被引号括起来，命令行上的每一个单词将会被认为是命令行上一个独立的参数。如果你在命令行上输入了多个参数，那么，可以用相应的参数号来引用它们。在下面的例子中，命令行上输入了四个参数。

greetargs 脚本程序清单：

```
echo "The first argument is: $1"
echo "The second argument is: $2"
echo "The third argument is: $3"
echo "The fourth argument is: $4"
```

```
$ greetargs Hello Hi Salutation "How are you"
The first argument is:Hello
The second argument is: Hi
The third argument is:Salutation
The fourth argument is:How are you
```

你可以用一组特殊字符以各种方式去引用命令行上的命令参数，例如“$*”、“$@”及“$#”等。其中，“$#”代表命令行上的参数个数，这在为一个脚本程序指定固定的参数数目时很有用。“$*”用来代表命令行上的所有参数（在一个命令行上可以有多达 9 个输入参数）。“$@”也代表命令行上的所有参数，但它可以使你单独引用所有命令行参数中的每个参数。“$*”与“$@”命令之间的区别不是很明显，但当你使用 for-in 控制结构引用命令行参数时，你就可以很好地理解它们之间的差别。因此，本节将对这些特殊参数只作简要介绍，在本章后面的控制结构中将有更详细的介绍。

在下面的例子中，命令行参数先后被用特殊变量“$*”及“$@”显示在屏幕上。命令行上的参数数量被特殊变量“$#”显示在屏幕上。

sargs 脚本程序清单：

```
echo $*
echo $@
echo "There are $# arguments"
```

```
$ sargs Hello Hi W elcome
Hello Hi W elcome
Hello Hi W elcome
There are 3 arguments
```

### 15.9.7 输出变量与脚本 Shell

当你执行一个脚本程序的时候，你将初始化一个新的进程，而该进程有其自己的 Shell。在该进程的 Shell 中，你可以自己定义变量、执行 Linux 命令，甚至执行其它的脚本程序。如果你在当前正在运行的脚本程序中执行另一个脚本程序，那么，当前正在被执行的脚本程序将被挂起，系统控制被转换到执行的另一个脚本程序上，在返回到被挂起的脚本程序之前，此脚本程序中的所有的命令都将首先被执行，然后再返回至被挂起的脚本程序中继续执行。

从一个脚本程序中执行另一个脚本程序的过程与程序设计语言中的函数或过程调用很相似。你可以认为是一个脚本程序在调用另一个脚本程序，直到被调用的脚本程序执行结束后，调用该程序的脚本程序才开始继续执行其余下的命令。

任何在你脚本程序中定义的变量都会在它对应的脚本 Shell 中定义，这些定义的变量只是局部变量，它只在它自己的 Shell 中起作用。从这种意义上说，有些变量是被隐藏在它们自己的 Shell 之中。但是，有时你希望在脚本程序中定义的变量可以在其调用的任何脚本程序中起作用。虽然你不能直接定义这样的变量，但你可以用 exort 命令把你定义的变量输出到另外一个脚本程序中。当你用 export 命令输出一个变量时，该命令将告诉系统在每个新创建的子 Shell 中都定义一个该变量的拷贝。于是每个新创建的子 Shell 都可以有该输出变量的拷贝。在下面的例子中，变量 myname 被定义，特殊也被用 export 命令输出。

```
$ myname="Charles"
$ export myname
```

通常有的读者把输出变量理解为全局变量，这种理解实际上是错误的。一

个 Shell 决不会引用一个在其外部定义的变量。如果在一个 Shell 内输出了一个变量，在创建新 Shell 时，系统只是在其子 Shell 中生成了该输出变量的值的一个拷贝，即输出变量只输出它们的值，而不是该变量本身。从某种程度上说，一个输出变量是一个限定范围内的全局参数（a scoped global parameter），系统会把该变量的值拷贝至在该 Shell 上创建的任何子 Shell 中。任何在输出变量后的 Shell 下直接或间接调用的 Shell 脚本程序都会得到输出变量初始值的一个拷贝。

## 15.10Shell 算术赋值操作：let 命令

let 命令是 BASH Shell 下对数字型变量进行操作的操作算子。通过该命令，你可以比较两个值或者执行算术操作如加法、乘法等。这些操作算子通常被用在 Shell 程序中，以管理控制结构或执行必要的计算。let 命令用关键词 let 来标识。该命令的句法如下：关键词 let 后跟着两个数字，而数字之间由一个算术或关系操作算子把它们分隔开。

```
$ let value1 operator value2
```

你可以使用表 15-4 列出的任何操作算子。let 自动假定操作算子是算术算子或关系算子。你不必象在 Shell 用变量求值符“$”去求变量的值，因为 let 命令会自动求出变量的值，并把它们的值转换成数字。这也意味着你可以在你的数字算子中使用简单的数字表达式。在下面的例子中，let 命令把 2 和 7 相乘，相乘后的结果输出到标准输出。

```
$ let 2*7
```

```
14
```

如果你想在算术表达式的操作数之间输入空格，你必须用双引号把整个算术表达式括起来。此时，let表达式如下：

```
$ let "2 * 7 "
```

你也可以在 let 表达式中包含赋值操作算子。在下面的例子中，乘法的计算结果被赋值给变量 res。

```
$ let "res = 2 * 7 "
$ echo $res
14
$
```

你也可以在数字之间使用任何关系操作算子来进行比较操作，例如，可以用关系算子来检查一个数字是否大于另一个数字。关系操作算子通常用来管理控制结构，例如循环控制结构或条件控制结构。在下面的例子中，脚本程序 helloprg 显示字符串“ Hello ”三次。该程序用 let 命令小于或等于操作算子来管理控制循环（let "again <= 3"），并用 let 命令使变量 again 逐次加 1（let "again = again +1"）。必须注意，当变量 again 被加 1 时，变量前不必加入求值符号“ $ ”，let 命令将在表达式中自动求出变量 again 的值。

helloprg 脚本程序清单：

```
again=1
while let "again <= 3"
 do
 echo $again   Hello
 let "again = again +1 "
```

```
done

$ Helloprg
1 Hello
2 Hello
3 Hello
```

## 15.11 控 制 结 构

你可以在 Shell 程序脚本中使用 Shell 控制结构来控制 Linux 命令的执行。控制结构可以使你重复执行某个命令或者在选择执行一些命令中特定的一个或几个命令。控制结构主要由两部分组成：测试条件命令与要执行的命令。如果测试条件返回条件为真，则其后的命令将被执行。因此通过使用控制结构这种方式，你可以在可能将要执行的命令之间作出选择。

主要有两类不同的控制结构：循环控制结构及条件控制结构。循环控制结构用来重复执行一些命令，而条件控制结构只有在满足某种条件时才执行相应的命令。BASH Shell 有三种主要的循环控制结构：While 循环、for 循环及 for-in 循环；有两种条件控制结构：if 条件控制结构及 case 条件控制结构。

while 及 if 控制结构是更常见的控制结构，用它们可以实现多种功能，如执行重复操作及在各种不同的测试条件下作出选择等。case 及 for 控制结构是更专门化的控制结构。case 控制结构是 if 条件控制结构的一种受限形式，它常用

来实现菜单；for 控制结构是一种有限次循环结构，它在执行每个循环的过程中会把列表中一系列的赋值给变量，直至这些列表中的值被赋值完毕。

if 及 while 控制结构以运行 Linux 命令作为它们的测试条件。所有的 Linux 命令在它们这些完毕后都会返回一个退出状态（exit status），如果该命令被成功执行，那么，该命令返回的退出状态为 0，如果该命令由于某种原因运行失败，那么，该命令返回的退出状态将会是一个正整数，这个数具体与它们的失败原因有关。if 及 while 控制结构将检查执行的 Linux 命令的返回值，看这一返回值是 0 还是其它的数字。对于 if 及 while 控制结构，如果测试命令的退出状态是 0，这测试条件命令被成功执行，然后开始执行控制结构中的命令。

### 15.11.1 测试命令

通常，你可能需执行一个比较测试操作，通过该操作来比较两个值的大小，而在控制结构中使用的测试操作是一个 Linux 命令，不是一个关系表达式。但如果要在控制结构中比较两个值，你可以用 Linux 系统中的 test 命令来执行这一比较操作。test 命令将比较两个值，并返回该命令的退出状态，如果该命令被成功执行，其返回状态将为 0。

通过 test 命令，你可以比较整数、字符串，甚至执行一个逻辑操作。该命令由关键词 test 开头，其后是要比较的值，这两个比较的值之间被一个选项分隔开（该选项用来指定要进行比较的类别）。这个选项可以认为是一个操作算子，只是在输入该选项的时候用一个负号“-”及一个字母表示，例如，-eq 选项表示比较两个值之间是否相等。但是，对于两个字符窜比较操作实际上是一个操作算子，而不是一个选项。当你比较两个字符串是否相等的时候，你要使

用等号“=”，对于不等号，用“!=”表示。表 15-5 中列出了 test 命令中使用的所有选项与操作算子。test 命令的句法如下：

```
test value -option value
test string = string
```

在下面的一个例子中，用户比较两个整数是否相等。此时，你需使用相等选项-eq，该命令通过 test 命令的退出状态来检查测试操作的测试结果。Shell 特殊变量“$?”用来保存最近一个 Linux 命令的退出状态（返回值）。

```
$ num=5
$ test  $num -eq 10
$ echo $?
1
```

你也可以使用方括号而不用 test 命令中的 test 关键词。例如，对于 test 命令 test $greeting="hi"，你可以些写成：

```
$ [  $greeting  =  "hi"  ]
```

同样，test 命令 test $num -eq 10 也可以写成：

```
$ [  $num  -eq  10  ]
```

方括号本身与其它字符之间必须用空白字符隔开，这些空白字符可以是空格、TAB 键或回车符等，如果没有空白字符，那么，该命令是无效的。

test 命令是一个 Linux 系统命令，该命令在测试控制结构中被大量使用。必须注意，该命令对字符串及整数等类型有不同的选项。同时，不要混淆字符串比较与整数比较选项之间的区别，对于比较两个字符串是否相等，用选项“=”，

而对两个整数的比较，需用选项“-eq”。

### 15.11.2 条件控制结构：if、if-else、elif 及 case

BASH Shell 中有一组条件控制结构，这组控制结构允许你选择要执行的命令。这些控制结构基本上与其它程序设计语言中的条件控制结构基本相同，但仍然有一些差别。if 条件控制结构用来测试一个 Linux 命令的执行后的退出状态（返回值），而不是一个表达式。此外，if-then 条件控制结构必须以关键词 fi 结尾，而 case 命令必须用关键词 esac 结尾。条件控制结构的详细句法见表 15-6。

#### if-then 条件控制结构

if 条件控制结构用来在命令执行之前设置一个条件选项，该条件选项是一个指定的 Linux 命令的退出状态（返回值）。如果该命令被成功运行，其返回值将为 0，然后，if 控制结构中的命令将被执行；如果该命令的返回值为一个任何非 0 值，这表明该命令运行失败，则 if 控制结构内的命令将不会被执行。

if 控制结构以关键词 if 开头，其后跟一个 Linux 命令，该命令的退出状态（返回值）将被引用，且该命令始终将会被执行；该命令之后是另起一行的关键词 then，再其后可以是任何一组 Linux 命令。最后，由关键词 fi 来结束该条件控制结构。通常你需根据一个 Linux 命令是否成功执行来在两组可能的命令之间作出选择，其中，else 关键词正是用来使 if 控制结构在这两个可能的选项之间作出选择。如果 Linux 命令成功运行，那么，关键词 then 后的命令将被执行；反之，如果该命令运行失败，那么，关键词 else 命令之后的命令将被执行。if-

then-else 控制结构的句法如下：

```
if Linux Command
then
commands
else
commands
fi
```

下面的脚本例子程序 elsels 用来执行带不同选项的列文件命令-ls 命令，这两个可能的选项是或者列出文件的大小，或者列出文件的所有信息。

elsels 脚本程序清单

```
echo Enter s to list file sizes,
echo otherwise all file information is listed.
echo -n "Please enter option:"
read choice

if  [ "$choice" = s ]
 then
 ls -s
else
ls -l
fi
```

```
echo Good-bye

$ elsels
Enter s to list file sizes,
otherwise all file informaton is listed.
Please enter option: s
total 2
 1   monday   2   today
Good-bye
$
```

在 Shell 脚本程序中，if 控制结构也通常被用来检查用户是否输入了合适的参数数。特殊 Shell 变量“#”将保存用户输入的参数数量。在测试命令中使用“$#”，可以检查用户是否输入了正确的参数数。

如果用户在命令行上输入了不正确的参数个数，你可以结束该 Shell 脚本程序的执行。你可以用 exit 命令来结束 Shell 脚本程序，并返回退出状态。exit 命令可以返回一个参数值，

例如，返回值为 0 表明该脚本程序成功运行后结束，而其它返回值，如 1，表明脚本程序运行时发生了错误。在下面的例子中，ifarg 脚本程序只需输入一个参数。如果用户在输入参数的过程中少输入了一个命令行参数，或者用户多输入了一个命令行参数，if 条件控制结构将为真，错误的消息将被打印出，脚本程序也将退出，并返回一个表示程序运行出错的一个返回值。

ifarg 脚本程序清单：

```
if  [ $# -ne 1 ]
 then
 echo Invalid number of arguments
 exit 1
fi

exho $1
```

```
$ ifarg

Invalid number of argumnets
```

elif 控制结构允许你在该控制结构中嵌入 if-then-else 控制结构。elif 结构表示“ else if”，该控制结构可以使你在几个可能的选项之间作出选择。第一个选项用 if 结构指定，其后跟另一个可能选项，每个可能选项都由其自己的 elif 控制结构指定。最后一个 elif 控制结构的可能选项用 else 来指定。如果第一个 if 控制结构的条件测试失败，那么，控制结构将把程序控制流程传递给其后的 elif 控制结构，如果又失败，那么，程序控制流程将传递给下一个 elif 控制结构。该过程直至到满足条件的测试命令，然后，该 elif 控制结构中的命令将被执行。最后，程序的控制流程传递给以 fi 关键词结尾的 if 控制结构之外。

## 逻辑命令：&&及||

逻辑命令用来对两个 Linux 命令执行逻辑操作。该命令的句法如下：

```
command && command
```

command || command

对于逻辑与“&&”，如果两个命令都成功运行，那么，该逻辑命令的退出状态（返回值）为 0。对于逻辑或“||”，如果任何一个命令成功运行，那么，逻辑或命令成功运行，并该返回该命令的退出状态为 0。逻辑命令可以使你在控制结构的测试命令中使用逻辑操作。

case

case 控制结构可以使你在几个可能的选项之间作出选择。它通过一个字符串与其它几个可能的模式之间的比较结果来作出相应选择。每个可能的模式都与一组操作相关联。如果找到与字符串匹配的模式，那么，与该模式相关的一组操作将被执行。case 控制结构以关键词 case 开始，其后跟一个字符串变量，然后是关键词 in，再其后是一组可能的模式,每个模式可以是一个正则表达式，并以回括符结束。在回括符之后是列出的与该模式想关联的一组命令，再在一个独立的行上输入两个分号“;”，表明与该模式关联的一组命令已经结束。最后是以关键词 esac 结尾的 case 命令。该控制结构的句法如下：

```
case string in
  pattern)
  commands
;;
  pattern)
  commands
;;
  *)
  default   commands
```

```
;;
esac
```

在上面的模式中可以包含任何 Shell 特殊字符。这些 Shell 特殊可以是“*”、“[]”、“?”及“|”等。你可以用单个特殊字符“*”来指定一个缺省的模式。该特殊字符“*”可以与任何模式匹配，因此，它可以作为一个非常有用的缺省项。如果字符串与所有的模式都不匹配，则它必与特殊字符“*”匹配。这样，如果没有与指定的字符串匹配的模式，缺省项将被执行。缺省项是可选的，你可以不输入该项。

一个 case 结构通常用来实现菜单。在下面的脚本程序 ischoice 中，程序问用户以何种形式列出当前目录下的所有文件。注意，本程序中的缺省项用来警告用户输入无效。

lschoice 脚本程序清单：

```
#Program to allow the user to select different ways of
# listing files

echo s. List Sizes
echo l. List all File Information
echo c. List C Files

echo -n "Please enter choice: "
read choice

case  $choice  in
```

```
  s)
ls -s
;;
  s)
ls -l
;;
  c)
ls *.c
;;
*)
  echo Invalid Option
esac

$ lschoice
s. List Sizes
l. List all File Information
c. List C Files
Please enter choice: c
main.c   lib.c   file.c
$
```

### 15.11.3 循环控制结构：while，for-in 及 for

BASH Shell 有一组循环控制结构，这组循环控制结构允许你重复执行一个

Linux 命令。这些循环控制命令是：while、for-in 及 for 控制结构。与 BASH Shell 中的 if 控制结构一样，while 及 until 控制结构也需测试一个 Linux 命令的执行结果。但是，for 及 for-in 控制结构不用执行条件测试命令。它们只是简单地把列表中的一系列值依次赋值给指定的变量来重复执行循环命令。此外，while 及 until 控制结构与在其它程序设计语言中的相应结构类似，但 for 及 for-in 控制结构却有很大的不同。循环控制结构的用法见表 15-6。

## While 循环控制结构

while 循环控制结构用来重复执行命令。while 循环控制结构以关键词 while 开头，其后是一个 Linux 命令，然后下一行是 do 关键词。该结构用关键词 done 表示循环控制的结束。

下面是 whle 循环控制结构的句法：

```
while Linux command
 do
commands
 done
```

while 控制结构中的 Linux 命令通常是一个用方括号括起来的测试命令。在下面的脚本程序 myname 中，程序要求用户输入用户名，然后，该程序打印出用户的名字。该循环控制结构通过用 test 命令（以用方括号括起来的方式）来测试变量的值。

myname 脚本程序清单：

```
again=yes
```

```
while  [  "$again" = yes ]
do
echo -n "Please enter a name:"
read name
echo "The name you entered is $name"

echo -n "Do you wish to continute ?"
read again
done

echo Good-bye
```

```
$ myname
Please enter a name: George
The name you entered is George
Do you wish to continute? yes
Please enter a name: Robert
The name you entered is Robert
Do you wish to continute? no
Good-bye
$
```

for-in **循环控制结构**

for-in **循环控制结构可以用来依次引用一列表中的值。该控制结构有两个操**

作数：一个变量及一个值列表。在 for-in 循环控制结构中，列表中的每一个值被依次赋值给变量。与 While 循环控制结构一样，for-in 控制结构也是一种循环控制结构，每执行一次该循环后，列表中的下一个值将被赋值给变量。当到达值列表的表尾后，for-in 循环控制结构将会停止。与 while 循环一样，for-in 控制结构的循环体以关键词 do 开头，以关键词 done 为结束。

for-in 控制结构的句法如下：

```
for variable in list-of-values
 do
 commands
 done
```

在下面的脚本程序 mylistfor 中，用户简单地输出列表中的每个项，并在列表项后加上当天的日期。该列表中的每个项被依次赋值给变量 grocery。

mylistfor 脚本程序清单：

```
tdate='date+D'

for grocery in milk cookies apples cheese
do
 echo "$grocery    $tdate"
done

$ mylistfor
milk12/23/93
cookies 12/23/93
apples     12/23/93
```

```
cheese      12/23/93
$
```

for-in 循环控制结构在管理文件时非常方便。你可以使用特殊字符来生成一个文件名列表，然后在 for-in 循环控制结构中使用这个文件名列表。例如，特殊字符“*”将列出当前目录下的所有文件，而“*.c”将列出当前目录下所有扩展名为.c 的文件。如果在 for-in 控制结构中使用特殊字符“*”（见下例）将生成由你当前目录下的所有文件名组成的列表。

```
for myfiles in *
do
commands
done
```

下面的脚本程序 cbackup 用来把当前目录下的所有扩展名为.c 的文件拷贝到名为 sourcebak
的目录下。请注意，该例子中特殊字符“*”的使用。

cbackup 脚本程序清单：

```
for backfile in *.c
do
 cp $backfile sourcebak/$backfile
 echo $backfile
done
```

```
$ cbackup
io.c
lib.c
main.c
$
```

## for 循环控制结构

for 循环控制结构中没有如 for-in 控制结构中使用的值列表（for-in 控制结构中用该值列表来作为命令行参数）。对于 for 循环控制结构，它把运行脚本程序时输入的命令行参数作为其循环值列表，而 for 命令将自动依次把这些值赋值给 for 循环中的变量。

在执行第一次循环时，命令行的第一个参数将被赋值给 for 循环中变量，而在执行第二次循环时，命令行的第二个参数将被赋值给 for 循环中变量，依此类推。for 循环控制结构没有指定值列表，但你必须在执行脚本程序的命令行上输入必要的参数去指定列表中的项。“$@”是一个特殊参数变量，它的值就是命令行上的所有参数组成的一个列表。看下面的例子：当运行例子脚本程序 cbackuparg 时，命令行上的 C 程序文件名将组成一个值列表，而该列表中的每个文件名将被自动依次赋值给 for 循环控制结构中的变量 backfile。在执行 for 控制的第一个循环时，变量 backfile 的值为第一个命令行参数，即$1，而在执行第二个循环时，变量 backfile 的值即为第二个命令行参数，即为$2。

cbackuparg 脚本程序清单：

```
for backfile
do
  cp $backfile source/$backfile
  echo "$backfile"
done

$ cbackuparg main.c lib.c io.c
main.c
lib.c
io.c
```

## 15.12BASH Shell 小结

Linux 系统下开发出了三种不同的 Shell,它们是 :BASH Shell、PDKSH Shell 及 TCSH Shell。BASH Shell 同时具有 PDKSH Shell 及 TCSH Shell 的大多数高级特性，包括命令行编辑、历史实用程序及别名等。历史实用命令能列出或查询你先前所执行的命令，你甚至还可以编辑并运行这些编辑后的命令。BASH Shell 下的 alias 实用命令可以使你给一个 Linux 命令取别名，你甚至还可以给带参数的命令取别名。

BASH Shell 下还有一组 Shell 特征开关变量。三个常见的特征开关变量是：ignoreeof、noclobber 及 noglob。ignoreeof 开关变量可以防止你意外的退出，

noclobber 开关变量可以防止你在使用重定向时意外地覆盖已经存在的文件，而 noglob 开关变量可以把 Shell 特殊字符看作普通的文本字符。

BASH Shell 下也有一组特殊的变量，通过这些变量，你可以配置你的用户 Shell。在这些特殊变量中，有的变量是由系统定义的，你不能再重新定义或赋值给它们，还有一些是用户定义变量，你必须明确地定义并赋值给它们。通过使用这些特殊变量，你可以配置你的用户环境。例如，PATH 特殊变量用来指定你运行的 Linux 命令或实用程序所在的目录；EXINI 变量用来保存编辑器 Ex 及 Vi 的缺省配置。

有一些特殊变量是在登陆初始化文件中定义与赋值的。登陆初始化文件是一个 Shell 脚本程序，该脚本程序在 Shell 被创建时自动执行。在 BASH Shell 中，登陆初始化文件是.bash_profile。在 C Shell 中，登陆初始化文件是.login。在 BASH Shell 中还有一个名为.bash_logout 的文件，该文件在你退出 Shell 时被自动执行。所有的 Shell 都有其 Shell 初始化文件，这些文件在你进入 Shell 时自动执行。BASH Shell 下的 Shell 初始化文件是.bashrc。

BASH Shell 有象其它程序设计语言一样的程序设计能力。你可以在 BASH Shell 脚本程序中定义变量，并给这些变量赋值。你也可以交互式地给一个变量赋值。BASH Shell 还有一组控制结构，包括条件控制结构及循环控制结构等。与其它程序设计语言一样，你可以在 Shell 脚本程序中设计程序语句，并执行该脚本程序。

你可以在 Shell 中定义变量，并给这些变量赋值。你可以在变量的前面键入特殊字符“$”来引用该变量的值。你也可以把变量作为命令行参数，该变量的值可以是目录路径，甚至还可以是将要运行的 Linux 命令。

你可以用文本编辑器来创建包含 Shell 命令及变量定义的文件，这些文件就是我们所说的 Shell 脚本程序。一个脚本程序甚至还可以有参数变量，这些参数变量用来接收你在命令行上输入的各种参数。通过设置 Shell 脚本程序的可执行权限，你可以把该 Shell 脚本程序看作是另一个新的 Linux 命令。

echo、read 命令及 Here 文件等可以用来控制脚本程序的输入与输出。echo 命令用来输出数据至标准输出，而 read 命令可以使你以交互式方式读取输入数据至变量。这些数据是来自标准输入输入的数据，它可以是来自键盘的输入数据，或者是来自于一个文件的重定向。Here 文件使你直接输入文本数据至脚本程序。Here 操作是在符号“<<”之后输入用户定义的结束定界符，定界符后输入的文本将被赋值给一个命令。

BASH Shell 脚本程序中常使用的变量是字符变量，即它们以字符串作为其变量值。除了等号之外，字符变量再没有其它操作算子，如果你想对变量执行算术、关系操作等，你必须使用 let 命令。在 BASH Shell 中，let 命令可以使你象使用数字变量一样使用定义的变量，例如，你可以执行加、减、乘、除等操作。

BASH Shell 中使用的循环控制结构及条件控制结构与其它程序设计语言使用的控制结构类似。BASH Shell 中使用的循环控制结构有：while、for-in 及 for 循环。条件控制结构是 if 及 case 结构。while 循环与其它程序设计语言中的 while 循环类似，该循环将永远执行，直至该控制结构中的测试条件是假为止。但是，在 BASH Shell 控制结构中的测试条件是一个 Linux 命令。Linux 系统中的所有命令执行后都会返回一个退出状态（返回值），以表明该该命令是否被成功运行。如果一个 Linux 命令的退出状态是 0，则表明该命令被成功运行；如果一

个 Linux 命令的退出状态是非 0 值，则表明该命令没能成功运行。对于 while 循环，其测试条件也是一个 Linux 命令。如果该 Linux 命令成功运行，那么，while 循环将继续；如果该 Linux 命令运行失败，那么，while 循环将结束。

BASH Shell 下的特殊命令 test 可以用来实现与其它程序设计语言中类似的测试条件。test 命令可以对两个变量值进行比较操作或关系操作。如果比较操作的结果为真，那么，test 命令的退出状态将为 0，表明该命令成功运行；如果比较操作的结果为假，那么，test 命令的退出状态将为非 0 值，表明该命令运行失败。在 test 命令中，你可以使用关系操作算子（即比较两个操作数）来控制一个循环结构。

if 条件控制结构根据一个测试条件的运行结果来决定是否运行该控制结构内的命令。与 while 控制结构一样，这个测试条件也是一个 Linux 命令。如果该 Linux 命令成功运行，那么，if 控制结构内的命令将被执行。if 条件控制结构也使用 test 命令来实现类似于程序设计语言中条件结构。test 命令可以对两个变量执行关系操作，如果关系操作的结果为真，那么，if 控制结构将继续执行其后的命令。你可以使用 elif 及 else 结构在 if 控制结构中嵌入多个条件控制。与 if 结构一样，elif 控制结构内有它自己的 Linux 测试命令。

case 条件控制结构类似于一个受限的 if 结构，它把一个变量的值与多个可能的模式进行比较，如果找到与变量值匹配的字符串，与该匹配的模式相关联的一组命令将被执行。case 条件控制结构非常适合实现菜单，即它可以让用户在多个可能的菜单选项之间作出选择。

for-in 循环控制结构用来把列表中的一系列值依次赋值给一个指定的变量。该结构中没有测试条件。for-in 控制结构中的一系列值可以由任何一个 Linux 命

令生成。例如，ls 命令可以生成一个文件名列表。这个列表中的值也可以是一组指定的单词，该列表中的每个单词都将会被依次赋值给 for-in 结构中指定的变量。该循环将一直进行下去，直至列表中的所有项都赋值给指定的变量为止。对于 for 循环控制结构，脚本程序不会指定其列表项中的每一项，而在运行其脚本程序时把该脚本程序的命令行参数作为其列表项。

表 15-1 命令行编辑、history 命令及查询历史事件

| 命令行编辑命令 | 功能 |
| --- | --- |
| CTRL-b 或左光标键 | 左移一个字符（移至前一个字符） |
| CTRL-f 或右光标键 | 右移一个字符（移至后一个字符） |
| CTRL-a | 移至行首 |
| CTRL-e | 移至行尾 |
| ESC f | 后移（forward）一个单词 |
| ESC b | 前移（backward）一个单词 |
| DEL | 删除光标所在的字符 |
| BACKSPACE 或 CTRL-h | 删除光标前的字符 |
| CTRL-d | 删除光标后的字符 |
| CTRL-k | 删除至行尾 |
| CTRL-n | 或向下光标键移至历史事件列表中当前事件的下一历史事件 |
| CTRL-p | 或上光标键移至历史事件列表中当前事件的前一历史事件 |

| | |
|---|---|
| ESC < | 移至历史事件列表表首 |
| ESC > | 移至历史事件列表表尾 |
| EASC TAB | 历史事件名自动匹配与自动扩展 |
| fc event-reference | 用标准编辑器编辑一个事件，然后执行之选项：<br>-l 列出最近的历史事件，与 history 命令相同<br>-e 编辑 event-reference，即启动一个指定的编辑器去编辑指定的历史事件 |
| !event num | 用历史事件号来查询一个历史事件 |
| !characters | 用历史事件的字符前缀来查询历史事件 |
| !?pattern? | 用模式来查询历史事件列表中的事件 |
| !-event num | 通过偏移量（相对于事件表中的第一个事件）来查询历史事件 |
| !num1-num2 | 引用历史事件号在 num1 与 num2 之间的事件 |

表 15-2BASH Shell 特殊变量与 Shell 特征开关变量

| Shell 特殊变量 | 功能 |
|---|---|
| **系统定义变量** | |
| HOME | 用户的主路径名 |
| LOGNAME | 用户登陆名 |
| USER | 用户登陆名 |
| TZ | 系统使用的时区 |
| **可重定义变量** | |

| | |
|---|---|
| SHELL | 用户正在使用的 Shell 类别及其应用程序所在路径 |
| PATH | 在执行命令时系统要搜索的目录路径列表 |
| PS1 | 主 Shell 命令提示符 |
| PS2 | 次选 Shell 命令提示符 |
| IFS | 定界符 |
| MAIL | 在接收电子邮件时 mail 实用程序需要检查保存邮件的文件名 |
| MAILCHECK | 检查是否收到新邮件的时间间隔 |
| 用户定义特殊变量 | |
| MAILPATH | 在接收电子邮件时 mail 实用程序需要检查的邮件文件路径名列表 |
| TERM | 终端名 |
| CDPATH | 执行 cd 命令时要搜索的子目录路径名 |
| EXINITEx/Vi | 编辑器的初始化命令 |
| BASH Shell | 特性开关变量 |
| set -+o feature | 可以用 set 命令来设置 BASH Shell 特性开关变量，-o 选项打开开关变量，而+o 选项关闭开关变量<br>$ set -o noclobber 打开开关变量 noclobber<br>$ set +o noclobber 关闭开关变量 noclobber |
| ingoreeof | 禁止使用 CTRL-D 退出 |

| | |
|---|---|
| noclobber | 重定向操作时不覆盖已存在文件 |
| noglob | 禁止文件名中特殊字符扩展（如*、?、~、及 []） |

表 15-3BASH Shell 命令与参数

| BASH Shell 命令 | 功能 |
|---|---|
| break | 从 for、while 及 loop 循环中退出 |
| Continue | 跳过循环中余下的命令，并继续下一次循环 |
| Echo | 显示变量值 |
| | -n 选项排除输出时的换行符 |
| Eval | 执行命令行 |
| Exec | 在当前进程中执行命令，运行时不创建新的子 Shell 而使用当前的 Shell |
| Exit | 从当前 Shell 中退出 |
| export var | 在每个子 Shell 中生成变量 var 的一个拷贝（值引用） |
| History | 列出历史事件列表中的事件 |
| let "expression" | 用表 15-4 中的操作算子给数字、关系表达式赋值（表达式必须用双引号括起来） |
| Read | 从标准输入上读取一行 |
| Return | 从函数中退出 |
| Set | 当在命令行上执行且带参数时，表示给变量赋新的值，当不带任何参数时，表示将列出当前定义 |

| | |
|---|---|
| | 的所有变量 |
| Shift | 把所有的命令行参数左移，于是引用该参数时的参数号将比先前小 1，也就是参数$3 将赋值给参数$2，依次类推。先前的参数$1 将丢失 |
| test value option value<br>[value option value] | 比较两个参数，用在 Linux 命令的控制结构中 test 2 -eq $count [ 2 -eq $count] |
| unset | 取消一个变量的定义 |
| **命令行参数** | |
| $0 | Linux 命令名 |
| $n | 从 1 开始的第 n 个命令行参数，你可以用 set 命令去改变它们的值 |
| $* | 从 1 开始的所有命令行参数，比可以用 set 命令去改变它们的值 |
| $@ | 可以被逐个单独引用的命令行参数 |
| $# | 命令行参数的个数 |
| $$ | 当前进程的进程 ID 号及 PID 号 |
| $! | 当前后台进程的 PID 号 |
| $? | 最后执行的一个 Linux 命令的退出状态（exit status），即返回结果 |

表 15-4let 表达式操作算子

| 算术操作算子 | 功能 |
| --- | --- |
| + | 加 |
| - | 减 |
| * | 乘 |
| / | 除 |
| % | 取模-除法的余数 |
| 关子操作算子 | |
| > | 大于 |
| < | 小于 |
| >= | 大于等于 |
| <= | 小于等于 |
| != | 不等于 |
| == | 等于 |
| & | 逻辑与 |
| \| | 逻辑或 |
| ! | 逻辑非 |

表 15-5Test 命令操作选项

| 整数比较 | 功能 |
| --- | --- |
| -gt | 大于 |
| -lt | 小于 |

| | |
|---|---|
| -ge | 大于等于 |
| -le | 小于等于 |
| -eq | 等于 |
| -ne | 不等于 |
| **字符串比较功能** | |
| -z | 测试空窜 |
| -n | 测试字符串的值 |
| = | 字符串相等 |
| != | 字符串不等 |
| str | 测试一个字符串是否不是空窜 |
| **逻辑操作算子功能** | |
| -a | 逻辑与 |
| -o | 逻辑或 |
| ! | 逻辑非文件测试操作算子功能 |
| -f | 文件存在，且是一常规文件 |
| -s | 文件非空 |
| -r | 文件是可读文件 |
| -w | 文件可以被修改（写入） |
| -x | 文件可以被执行 |
| -d | 文件是一个目录名 |
| -h | 文件是符号链 |
| -c | 文件与一特定设备关联 |

| -b | 文件与一块文件关联 |
|---|---|

表 15-6BASH Shell 控制结构与循环

| if, else,elif,case | 功能 |
|---|---|
| if ( command ) then commands fi | 如果其测试命令被成功运行，那么，if 控制结构将执行 then 后的一个或一组命令 |
| if ( command ) then commands esle commands fi | 如果其测试命令被成功运行，那么，if-else 控制结构将执行 then 后的一个或一组命令，否则，else 后的一个或一组命令将被执行 |
| if ( command ) then<br>command<br>elif ( comamnd ) then<br>command<br>else<br>command<br>fi | elif 可以允许你在该控制结构中嵌入一个 if 控制结构，从而可以使你在几个可能选项之间进行选择，如果 if 后的测试命令的返回结果为真，其后的命令将被执行，而控制结构将退出整个 elif 控制结构 |
| case string in<br>pattern)<br>command;;<br>esac | case 控制结构中的 string 与多个可能的模式进行匹配，如果找到匹配的模式，那么，与该匹配模式相关联的命令将被执行 |
| command && command | 如果两个命令的返回值都是 0，那么， |

| | |
|---|---|
| | 逻辑与的返回结果将为 0；如果其中的一个命令的返回结果为非 0 值，则逻辑与的返回结果为非 0，逻辑与的返回值为假 |
| command \|\| command | 如果两命令中的一个命令返回值为 0，那么，逻辑 OR 条件命令返回的结果将为 0，逻辑或将返回 0；如果两个命令都返回值一个非 0 值，那么，逻辑或条件命令的返回值将为假，并返回一个非 0 值!command 逻辑非将反置命令的返回值 |
| **循环控制结构：功能** | |
| while，until，for，for-in select | 只有当 while 控制结构的测试命令为真时，do 后面的命令才被执行 |
| while (command)<br>do<br>command<br>done | |
| until command<br>do<br>command<br>done | 只有当 until 控制结构的测试命令为假时，do 后面的命令才被执行 |
| for variable in list-values<br>do command | for-in 控制结构用来引用列表中的一系列值，列表中的值被逐个赋值给变 |

| | |
|---|---|
| done | 量 variable |
| for variable do command<br>done | for 循环控制结构用来引用执行脚本程序时输入的命令行参数。命令行参数上的每个参数将被逐个赋值给变量 variable |
| select string in item-list<br>do<br>command<br>done | select 控制结构可以根据 item-list 中的项创建一个菜单，然后，执行相应的命令，该命令通常是一个 case 控制结构 |

# 第 16 章 TCSH Shell

TCSH Shell 实质上是 C-Shell 的另一版本，它在 C-Shell 的基础上增加了许多新的特性。它与标准的 C-Shell 完全兼容，并且具有 C-Shell 的所有功能与特性，包括 Shell 程序语言及 history 实用程序等。C-Shell 本身最初是为 BSD Unix 系统开发的，它包括了其最初版 BourneShell 的所有核心命令，但与 Bourne Shell 又有很大的不同，特别是前者具有更多更复杂的特性，如 Shell 程序设计。C-Shell 是在 Bourne Shell 之后才开发出来的，因此，它最先引入了一些新的特性，如命令行编辑、history 应用程序及别名等。后来的 Korn Shell 具有上述这些特性外，还增加了更多更实用的命令行和 history 通用编辑命令。同样，这些功能也增加到 BASH 及 TCSH Shell 中去了。因此，在 TCSH Shell 中，你会发现它比最初的 C-Shell 具有更多的高级命令行与 history 编辑特性。

在第 5 章中讨论过，TCSH Shell、BASH Shell 及 PDKSH Shell 都具有一些基本的 Shell 命令。TCSH Shell 与其它 Shell 的一个显著不同是它以“>”符号作为缺省的命令提示符，而不是“$”。本章将重点讨论 TCSH Shell 与其它 Shell 的不同之处。具体地说，主要是讨论命令行编辑、history 实用程序及 Shell 环境变量与系统初始化文件等。本章还将介绍 TCSH Shell 下的 Shell 编程。表 16-1 中列出了 TCSH Shell 下 history 命令的使用说明。

## 16.1 命令行扩展

命令行有一个内在特征，那就是命令行与文件名扩展特性。如果你把一个输入还没有完成的字符串作为一个文件名参数，你可以键入 TAB 键来激活这一特性，该特性可以完成文件名自动扩展。如果你键入 CTRL-D，那么，系统将列出所有与你键入的字符或字符串相匹配的文件名。你可以在命令行上键入文件名的部分字符，然后键入 TAB 键就可以利用该特性了。这时，Shell 将寻找与你所键入的部分前缀相匹配的文件名，并自动将文件名扩展在命令行上。

在下面的例子中，用户键入 cat 命令，并在其后键入一个还未完成的文件名，当用户键入 TAB 键后，Shell 搜索与之相匹配的文件名，如果找到，将扩展该文件名至命令行上。

```
> cat pre TAB
  > cat preface
```

如果不止一个文件具有与你键入的前缀相匹配，系统将尽可能多的扩展与文件相匹配的字符，然后响铃，等待用户键入更多的字符，以便选定一个文件，例如：

```
> ls
document docudrama
> cat doc TAB
  >   cat docu beep
```

如果你想列出所有与你未完成的文件名相匹配的文件，可以在命令行上键入 CTRL-D。下面的例子中，在没完成的文件名后键入了 CTRL-D，Shell 列出了所有用户可能想输入的文件名。

> cat doc CTRL-D
document docudrama
> cat docu

系统又重新回显命令行，你可以键入你所想键入的文件名的剩下部分，或继续输入字符，再键入 TAB 键来完成文件名的自动扩展。

> cat docudrama

## 16.2history 命令

和 BASH Shell 及 PDKSH Shell 一样，TCSH Shell 的 history 实用命令保存最近你所执行的一些命令。history 实用命令用一小段内存来记录有限个最近你所执行的命令。如果历史实用命令没有自动启动，你首先必须用 set 命令定义它，并赋一 t 值给它，用来确定你想保存最近执行的命令个数。该步骤通常是在你的系统配置文件中完成的，这在本书第 14 章中已经讨论过。下面的例子中，设置 history 实用程序保存最近的 5 个命令。

> set history=5

这些被保存的命令在 Linux 系统中通常被称为事件（event）。要想查看最近你执行的一些命令，在命令行上键入 history 命令，然后键入 ENTER 键（回车键），那么，你最近执行的一些命令的列表将回在屏幕上（命令前的数字为历史记录号）。

> history
1 ls

```
2 vi mydata
3 mv mydata reports
4 cd reports
5 ls -F
>
```

history 实用命令能让你查询以前的历史事件，并可把它们显示到命令行上或执行它。但是，你不必先用 history 将它们列出。最简便的办法是通过上下光标键，把先前的一个个事件逐次显示到命令行。按一下上光标键，你上一次执行的一个事件将出现在命令行上。再按一下，上一次的前一事件又会出现在命令行上。按一下下光标键，将会使当前事件的下一事件出现在光标行上。

你也可以对显示在命令行上的事件进行编辑。通过左右光标键在命令行上移动光标。你可以在任何你想输入字符的地方插入字符，用 BASKSPACE 键或 DEL 键删除你想要删除的字符，用 CTRL-A 命令将光标移动至命令行的行首，用 CTRL-E 命令将光标移动至命令行行尾，用 CTRL-K 命令删除光标后所有的字符，用 CTRL-U 命令删除整行。

显示的历史事件列表是用户在对历史事件进行搜索时强有力的手段。所有这些历史事件可以通过历史事件的事件号、事件的初始字符或字符串及字符匹配等方式查询到。一种查询方式是在字符串中嵌入“?”号。 你可以用 history 应用程序的“!”号命令重新执行历史事件。在“！”号命令后键入能够确定先前事件的关联字符，这关联字符可以是先前事件的历史事件号，或者该事件的前几个字符串。在下面的例子中，历史事件列表中的第二个事件（历史事件号为 2）的事件被查询到。同时，又用其开头几个字符串去匹配，也被查询到，最后用事件正文字符匹配方式也被查询到。

```
> !2
vi mydata
> !vi
vi mydata
> !?myd?
vi mydata
```

你也可以用一个偏移量（相对于历史事件列表中最后一个事件）来查询历史事件。负的偏移量将通从历史事件列表表尾向前偏移，从而查询到所要的事件。在下面的例子中，历史事件号为 2 的事件“vi mydata”就是用一个负的偏移量查询到的。

```
> !-4
vi mydata
```

惊叹号“!”也可以查询用户所执行的最后一个命令，它相当于偏移量为-1。在下面的例子中，惊叹号“!”及偏移量为-1 都能查询到用户最后执行的命令：ls -F。

```
> !!
ls -F
mydata /reports

> !-1
ls -F
mydata /reports
```

### 16.2.1 历史事件替换

一个事件关联（用来指定某事件）可以认为它代表了组成该事件的字符串。例如，“!1”实际上代表字符串“ls”。因此，你可以把一个事件关联作为另一个命令的一部分。历史事件的运行基本上就是一个替换操作，即用组成历史事件的字符去代替“!”号及其事件关联。

在下面的例子中，首先列出历史事件列表，然后，第一个历史事件的关联被替换，作为了新命令的一部分，即：“!”号与事件关联（此处为历史事件号1）被替换成“ls”，成为命令：ls > myfiles 的一部分。

```
> history
1 ls
2 vi m ydata
3 mv mydata reports
4 cd reports
5 ls -F

> !1 > myfiles
ls > myfiles
```

你也可以引用某历史事件中特定的单词。一个事件可被分解为几个单词，我们可以从 0 开始依次给每个单词编号。在一个事件关联后键入“：”号，然后键入历史事件中单词的编号就可唯一确定一个单词。例如，“!3”代表历史事件号为 3 的事件 mv mydata reports,在该事件中，其第二个单词为 mydata，因此，“!3:1”就代表单词 mydata。下面的例子中，“2:0”代表事件 2 中编号为 0 的单词 vi，于是，该命令就是：vi preface。

```
> !2:0 preface
```

vi preface

用一系列的数字，你可以从一个事件中引用几个单词。在第一个单词与最后一个单词的编号之间用“-”分开。下面的例子中，3:0-1 将引用历史事件号为 3 的事件中前两个单词 mv 和 mydata。

> !3:0-1 oldletters
mv mydata oldletters

特殊字符“^”和“$”分别代表一个事件中第二个单词及最后一个单词，它们用来引用事件中的参数。如果你只想引用某事件的第一个参数，符号“^”将代表该参数，而符号“$”代表该事件的最后一个参数。“^-$”将引用事件中所有的参数（只有事件的第一个单词即命令名将不被引用）。下面的例子中，前一历史事件的参数将被引用为当前事件的参数。首先，历史事件号为 2 的事件的第一个参数（第二个单词）mydata 被作为 lpr 命令的参数，用来打印该文件；随后，历史事件号为 3 的事件的最后一个参数 reports 被作为 ls 命令的参数，用来列出 reports 目录中的所有文件名； 最后，历史事件号为 3 的事件的所有参数 mydata、reports 被用来作 cp 命令的参数。

> lpr !2:^
lpr mydata
>ls !3:$
ls reports
> cp !3:^-$
cp mydata reports

符号“*”也是一个特殊的符号，用来代表历史事件中的所有参数，它与“^-$”的功能是相同的。上例中最后一个例子可以符号“*”代替如下：

```
> cp !3:*
cp mydata reports
```

C-Shell 及 TCSH Shell 中，在命令的任何地方都可以使用惊叹号“!”，它会被解释为对历史事件的引用。如果由于某些原因，你要使用惊叹号“!”，例如在电子邮件地址中，必须用在惊叹号前

放置反斜杠“＼”来防止惊叹号引用历史事件。

```
> mail garnet＼!chris < mydata
```

## 16.3 别名

通过 alias 命令，你可以给一个命令创建一个新的名字。alias 命令象一个宏，对其代表的命令进行替换。它不是逐字对命令名进行替换，而是仅仅对命令取另外一个名字。alias 命令由关键词 alias 开头，然后是要替换命令的新的名字，其后是空格及将要替换的命令。下面的例子中，list 就是 ls 命令的一个别名。

```
> alias list ls
> ls
mydata intro
> list
mydata intro
>
```

如果你要改名的命令有选择项，那么，你必须用单引号把该命令及其选择

项括起来。如果改名后的命令有空格，在改名时，你同样必须用单引号括起来。下面的例子中，带-l选项的ls命令被改名为long1：

```
> alias long1 'ls -l'
ls -l
-rw-r--r--   1   chris   weather 207 Feb 20   11:55   mydata
> long1
-rw-r--r--   1   chris   weather 207 Feb 20   11:55   mydata
>
```

你也可以用一个Shell命令名给一个命令行取别名。例如带-i选项的rm、cp及mv命令通常用来保证不覆盖已存在的同名的文件。如果你在使用上述命令的过程中，希望系统给予相应的提示来确认你是否想覆盖已有的文件，你可以给它们取别名去包含这一选项。例如：

下面的命令就是给带-i选项的rm、cp及mv命令取别名为rm、cp及mv。

```
> alias rm=' rm -i'
> alias mv=' mv -i'
> alias cp=' cp -i'
```

不带任何参数的alias命令将列出所有当前已经起作用的别名及它们所代表的命令。你可以用unalias命令去取消一个别名。在下面的例子中，用户列出了当前所有的别名，然后用unalias命令取消别名lss。

```
> alias
lss ls -s
list ls
rm rm -i
> unalias lss
```

## 16.4TCSH Shell 特征变量：Shell 特性

TCSH Shell 有几个特征变量，这些特征变量可以控制 Shell 在不同的方式下工作。TCSH Shell 特征变量包括 PDKSH Shell 中定义的一些变量以及 TCSH Shell 中专有的一些变量。例如在 TCSH Shell 中有一开关变量 noclobber，可以防止你重定向时意外地覆盖一个已经存在的文件。另外一些常见的特征变量有：echo、noclobber、ignoreeof 及 noglob 等（见表 16-2）。
通过定义或不定义这些变量能关闭或打开与该特征变量相关的 TCSH Shell 特性。每个变量都是对应于某特定特性。例如：通过定义 noclobber 特征变量，能打开 noclobber 变量所对应的 TCSH Shell 特性。你可以通过 set 命令来定义一个特性变量，用 unset 命令来取消某个特征变量的定义。要打开 noclobber 变量所对应的特性，你可以键入命令：set noclobber，要关闭这一特性，键入命令：unset noclobber。

> set feature-variable
> unset feature-variable

通常这些特征变量也被称为按钮（toggle）类变量，因为它们用来打开或关闭 Shell 的某些特性。

### 16.4.1echo

如果设置 echo 特征变量，将使 TCSH Shell 具有如下特性：在每次执行一命令前，TCSH Shell 将首先显示该命令。set echo 命令将打开上述特性，unset echo 将关闭上述特性。

### 16.4.2ignoreeof

设置了 ignoreeof 变量后，TCSH Shell 将阻止用户使用 CTRL-D 命令来退出用户 Shell，它可以用来防止用户意外退出系统。当该特性开关被关闭时，你可以按 CTRL-D 键来退出系统。但是，CTRL-D 不仅仅被用来退出 Shell，而且可以用来终止用户直接输往标准输出上的输入。该操作经常在 mail 程序和一些 Shell 实用命令中使用，例如 cat 实用命令。你可能在上述实用程序操作中，非常容易误操作而意外地退出 Shell。ignoreeof 特殊变量正是用来防止这些意外的退出。当该变量被设置时，你只能用 logout 命令非常明确地告诉 Shell 你想退出系统。

```
> set ignoreeof
> Ctrl-D
Use logout to logout
>
```

### 16.4.3noclobber

设置了 noclobber 变量后，可以使在重定向输出时保护已存在的文件。如果你在使用重定向操作把数据输出到一个已经存在的文件中时，用 noclobber 这一特殊变量可以防止文件在进行标准输出时被覆盖，原文件将被受到保护。这种情况有可能发生在把重定向输出的内容保存到你输入的文件名中，而你给出的这个文件名可能已经存在了。因此，noclobber 特殊变量可以防止你意外地覆盖一个已经存在的原文件。

```
> set  noclobber
```

```
> cat preface > myfile
myfile: file excists
>
```

你可能觉得当时你的确是想在重定向输出时覆盖一个文件，系统的这一提示将浪费你的时间，这时，你可以在重定向操作算子后面加一惊叹号（!）。这时，系统将不考虑 noclobber 变量的有效性而直接把重定向输出的内容输出至指定的文件。

```
> cat preface >! myfile
```

### 16.4.4noglob

设置了 noglob 变量后，Shell 将不扩展文件名中一些特殊的字符或字符串。例如：字符“*”、“?”、“[”、“]”及“~”等字符将不再展开使多义性文件名与一文件名相匹配。如果由于某些原因，你在文件名中使用了这些特殊字符，这个特殊变量或许就对你有用。下面的例子中，如果用户希望列出结尾为“?”的文件名 answer?，可通过如下步骤：首先，用户设置 noglob 变量来关闭上述特殊字符扩展特性，然后再列出文件名。可以看到，目前命令行上的问号（“?”）被认为是文件名中的一个字符，而不是被看作一个特殊的字符（代表任意一字符）。

```
> set  noglob
> ls answer?
answer?
```

## 16.5 用 TCSH Shell 特殊设置用户系统

与 BASH Shell 一样，你可以在 TCSH Shell 中使用特殊 Shell 变量去配置你的 Linux 系统。有些变量是你的系统最初就定义了的，你可以重新给它们定义一个新的值；也有另外一些变量，你必须首先定义它们。其中，一个最常见的特殊 Shell 变量就是 prompt 变量，这个变量用来任意设置

你想设置成的命令行提示符。还有一个常见特殊 Shell 变量就是 history 变量，它用来保存你想保存的历史事件数。

在 TCSH Shell 中，许多特殊变量与 BASH Shell 及 PDKSH Shell 中的特殊变量名相同，且功能相似。有些特殊变量由大写字母组成，但更多的是由小写字母组成。变量 EXINIT 及 TERM 仍然是由大写字母组成，但变量 history 及 cdpath 由小写字母组成。另外一些特殊变量完成相同的功能，但使用时却有很大的不同。例如变量 mail 同时具有 BASH Shell 中的 MAIL、MAILPATH、MAILCHECK 这三个变量的功能。这些变量在表 16-3 中列出。

### 16.5.1 变量 prompt、prompt2 及 prompt3

变量 prompt、prompt2 及 prompt3 设置你的命令行提示符。你可以把命令提示符设置成任何符号或字符串。要把命令提示符设置成不同的符号，只需用 set 命令把所要设置的符号赋值给 prompt 变量即可。在下面的例子中，用户把符号“+”赋值给 prompt 变量，使其成为新的命令提示符。

```
> set prompt ="+"
```

+

Shell 提供一组预定义符，用这些预定义符可以更容易地配置你的命令提示符。你可以用当前时间、你的用户名及当前目录名等作为命令提示符的一部分，甚至可以用将要输入的命令的历史事件号来作为命令提示符。每个预定义符由符号“%”开头，例如“%/”代表当前的工作目录，“%t”代表当前时间，“%n”代表你的用户名，而“%!”将显示出将要输入命令的历史事件号。在下面的例子中，用户在命令提示符上增加了当前的工作目录。

```
> set prompt = "%/   >"
/home/dylan >
```

下一例子同时使用了当前时间及下一命令的历史事件号来作为命令提示符。

```
> set prompt = "%t %! $"
```

这里列出 TCSH Shell 提供的一些预定义符：

| 预定义符 | 作用 |
| --- | --- |
| %/ | 当前目录 |
| %h、%!、! | 当前将要输入命令的历史事件号 |
| %t | 当前时间 |
| %n | 当前用户名 |
| %d | 当前日期（用星期几表示） |
| %w | 当前月份 |
| %y | 当前年份 |

变量 prompt2 使用在如下特殊情况：当一个命令需用几行才能完成输入时，增加的行将用 prompt2 作为命令提示符。如果检查拼写特性被激活，命令提示符将采用变量 prompt3 的值。

### 16.5.2cdpath

变量 cdpath 保存一些目录路径名，当执行 cd 命令时，shell 将对变量 cdpath 中指定的目录路径名进行搜索。这些目录路径名形成一个列表，正如 TCSH Shell 中赋值给变量 path 的目录路径名列表一样。请注意目录路径名间空格。

```
> set cdpath=(/usr/chris/reports /usr/chris/letters)
```

### 16.5.3history 和 savelist

前面已经介绍过，变量 history 用来保存用户想要保存的历史事件数。你只需赋值给 history 变量一个最大数（最大的历史事件号），当达到最大历史事件号后，系统将重新从 1 开始计数。而 savelist 变量保存当你退出系统时，将要保存到.history 文件中的事件数。当你重新登陆系统时，这些事件将成为你系统的初始历史事件列表。

```
> set history=20
> set savelist=5
```

### 16.5.4mail

在 TCSH Shell 中，mail 变量同时具有 BASH Shell 或 PDKSH Shell 中三个变量 MAIL、MAILCHECK 及 MAILPATH 的功能与特性。TCSH Shell 中的

变量 mail 在被赋值时，需给其赋值一个变量列表，该列表中同时包括检查邮件的时间间隔及检查的邮件信箱文件所在的目录路径名。要确定检查邮件的时间间隔及检查的邮件信箱文件所在的目录路径名，需给 mail 变量赋值一个变量列表，该列表由圆括号“（）”括起来的一系列字符串组成，字符串与字符串之间由空格分隔开。表中第一项是一个数字，它用来设定检查邮件的时间间隔（用秒表示），该项相当于 BASH Shell 中变量 MAILCHECK 中保存的值。余下的项为邮件信箱文件的目录路径名，以便系统检查该邮件信箱文件是否收到新的邮件。需要注意的是，该部分的值相当于同时保存了 BASH Shell 或 Korn Shell 中变量 MAIL、MAILPATH 的值。

下面的例子中，mail 变量被设置成：每 20 分钟（1200 秒）检查一次信箱，要检查的邮件信箱文件在目录/usr/mail/chris 下。列表中的第一项为 1200，用来赋值给 mail 变量，以决定检查邮件时间间隔，第二项为将要检查的邮件信箱文件的路径名。

> set mail (1200 /usr/mail/chris)

你也可以非常方便地给 mail 赋值多个邮件信箱文件目录路径。在下面的例子中，两个邮件信箱文件目录路径被赋值给变量 mail。必须注意列表中各项之间的空格。

> Set mail (1200/usr/mail/chris/home/mail/chris)

16.6TCSH Shell 初始化文件：.login、.tcshrc 及.logout

TCSH Shell 有三个初始化文件：.login、.tcshrc 及.logout。这些文件根据它们所执行的功能来命名。.login 是登陆初始化文件，在你每次登陆时都自动运行；.logout 文件在你每次退出系统时都自动运行；.tcshrc 文件是 Shell 初始

化文件,在你每次进入 TCSH Shell 时都自动运行,可以是在直接登陆进入 TCSH Shell，或是在另外一种 Shell 下，用 tcsh 命令转换至 TCSH Shell 下。

### 16.6.1.login

TCSH Shell 有其自己的登陆初始化文件.login，该文件中包含一些 Shell 命令及一些特殊环境变量的定义,它们用来配置用户的 Shell。该文件对应于 BASH Shell 及 PDKSH Shell 中的.profile 文件。

.login 文件通过命令 setenv 给一些特殊环境变量赋值，例如变量 TERM。你可以用任何编辑器编辑.login 文件来改变这些特殊环境变量的值，也可以在该文件中增加新的环境变量，但是，必须注意：在 TCSH Shell 中，用来给特殊环境变量赋值的命令是 setenv。下面的例子中，变量 EXINIT 被赋值，从而使 vi 编辑器具有显示行号及自动缩进特性。

```
> setenv EXINIT 'set nu ai'
```

在编辑.login 文件的时候必须非常小心，稍不注意可能导致变量赋值不正确或根本没被赋值。因此，在对.login 文件进行编辑之前，备份.login 文件是明智而必要的。

如果你已经更改了.login 文件，并希望你的更改在当前登陆下生效，你需要重新运行该文件。此时，你需用 source 命令。source 实际上可以运行任何初始化文件,包括.tcshrc 及.logout 文件等。在下面的例子中,用户重新运行了.login 文件。

```
> source .login
```

如果你也想在你的 Linux 系统上使用 PDKSH Shell，你需在.login 文件中定

义一名为 ENV 的变量，并把 PDKSH Shell 初始化文件赋值给它。在登陆进入 TCSH Shell 之后，如果你想从 TCSH Shell 进入 PDKSH Shell，系统将会找到 PDKSH Shell 初始化文件，并执行该初始化文件。下面列出了.login 文件的清单，可以看到，文件中最后一命令 setenv ENV $HOME/.kshrc 就是把 PDKSH Shell 初始化文件赋值给变量 ENV。

.login 文件清单

```
setenv term vt100
setenv EXINIT 'set nu ai'

setenv ENV $HOME/.kshrc
```

### 16.6.2.tcshrc

在你每次进入 TCSH Shell 或在 TCSH Shell 创建任何子 Shell 时，系统都会运行.tcshrc 初始化文件。如果 TCSH Shell 就是你登陆后的 Shell，在登陆及.login 文件运行后，.tcshrc 文件就会运行。如果你是从另外一个 Shell 进入 TCSH Shell，.tcshrc 文件就会被自动运行，该文件中的一些变量或别名就会被定义并生效。

实际上，初始化文件.tcshrc 在你每次创建 Shell 时（例如运行 Shell 脚本程序时），都会自动执行。换句话说，每创建一个子 Shell，.tcshrc 文件就会被自动运行。这意味着，在初始化文件.tcshrc 中定义的局部变量会被输出至任何子 Shell。尽管一些“用户级定义”（user-defined）特殊变量，例如 history，是局部变量，但是，它们会在每个子 Shellz 中被定义。因此，从这个意义上说，

变量 history 是为每个子 Shell 设置的，尽管每个子 Shell 都有其自己的局部变量。你甚至可以在一个子 Shell 中改变局部变量 history 的值，但它不会影响其它子 Shell 的值。与 BASH Shell 中一样，Shell 初始化文件中定义的变量可以用 export 命令进行变量输出。在第 15 章中已经讨论过，在 BASH Shell 及 PDKSH Shell 中输出的变量仅传递该变量的一个拷贝至其子 Shell，改变这一拷贝的值并不改变该输出变量本身的值。

.tcshrc 文件中也包括一些别名的定义及一些用来改变 Shell 特性的特殊开关变量。别名和特性变量是在 Shell 内局部定义的，但 Shell 会在每个子 Shell 中定义它们的一个拷贝。因此，.tcshrc 文件通常定义一些为 rm、cp 及 mv 命令等定义的别名。下面的 .tcshrc 文件中包括了一些标准的定义。

.tcshrc 文件清单

```
set shell=/usr/bin/csh
set path=$PATH(/bin /usr/bin .)
set cdpath=(/home/chris/reports /home/chris/letters)

set prompt = "!$cwd >"
set history =20

set ignoreeof
set noclobber

alias rm "rm -i"
alias mv "mv -i"
```

alais cp "cp -i"

局部变量与环境变量不同，它们是由 set 命令定义的。任何在.tcshrc 中定义的局部变量必须用 set命令；任何定义的环境变量必须用 setenv命令去定义，且必须在文件.login 文件中定义，例如变量 TERM。下面是另外一些在.tcshrc 文件中定义的局部变量。请注意变量 history 及 noclobber 是用 set命令定义的。

set history=20
set noclobber

你可以编辑上述任何变量，并给它们重新赋值。但是，当给 path 及 cdpath 变量增加目录路径时，必须记住：这些目录路径要用圆括号“()”括起来，且每给目录路径之间要用空格分隔开。

如果你增加了新的目录路径名，你必须确保目录路径与目录路径之间已用空格分隔开。如果你对.tcshrc 文件作了任何更改，并想使这些更改在当前 Shell 下生效，记住必须用

source 命令重新运行该文件。

> source .tcshrc

### 16.6.3.logout

.logout 文件也是一初始化文件，但该文件是在用户退出系统时执行的。.bash_logout 也是一个初始化文件，当用户退出系统时，将执行该文件。当你退出时，你可以指定系统执行你想做的操作。.logout 文件不是用来定义变量或别名的，而是通常由一些 Shell 命令组成的关机程序，也就是关机时你想

要系统做的一些操作。例如，一个常见的操作是退出时检查后台运行的进程；另一常见操作是希望系统退出时清屏，然后向用户发出祝福或结束信息。

与.login文件一样，你可以在.logout文件中增加自己的Shell命令语句，用vi编辑器你可以改变退出时系统向用户发出的信息或增加其它操作与功能。在下面的例子中，用户的.logout文件中有clear及echo两条命令，当用户退出时，clear命令将首先进行清屏，然后echo命令向用户显示“Good-bye for now”信息。

.logout文件清单

```
clear
echo "Good-bye for now"
```

## 16.7TCSH Shell程序设计

与BASH Shell及PDKSH Shell一样，TCSH Shell也具有和其它程序设计语言一样的编程能力。你可以定义变量并赋值给它们，你也可以在脚本程序中定义变量，键入Linux命令并执行这些脚本程序。TCSH Shell还提供了循环、条件等控制结构。其中，循环结构用来多次执行某些Linux命令；条件结构用来控制、选择你要执行的命令。你也可以在程序中截获并处理中断。

TCSH Shell与其它Shell的不同之处在于它的控制结构更接近程序设计语言，并符合程序设计的规范。例如，TCSH Shell的条件控制结构是被赋值以true

或 false 的一个表达式，不是一个 Linux 命令。TCSH Shell 中的表达式使用的算子与 C 语言程序设计中的表达式使用的算子相同。你可以进行各种不同的赋值操作、算术运算、比较运算及逻辑运算等。你可以声明数字型变量，然后在上述操作、运算中使用它们。

### 16.7.1TCSH Shell 变量、脚本程序及参数

你已经知道，在 TCSH Shell 中，可以象在 BASH Shell 及 PDKSH Shell 中那样使用 Shell 变量。你可以在 Shell 中定义变量，并赋值给它们，同时，也可以引用脚本程序命令行参数。与 BASH Shell 进行变量输出一样，你也可以在程序中定义环境变量。但 TCSH Shell 的不同之处在于在定义变量时必须定义你所定义的变量类型。在 TCSH Shell 中，TCSH Shell 用 set、@及 setenv 命令来定义变量。TCSH Shell 还允许你定义数字型变量及数组。@命令用来定义数字型变量，用它们可以完成算术运算。圆括号“（）”用来定义并引用数组。

进行 TCSH Shell 脚本程序设计基本上与 BASH Shell 差不多，但也有一些重要差别。例如，TCSH Shell 脚本程序的第一行第一列必须是“#”号；尽管 echo 命令可以用来进行显示、输出，但没有用来处理输入的 read 命令，而需用标准输入重定向操作来给变量赋值。

#### TCSH Shell 变量

TCSH Shell 中，在使用一个变量之前，你必须首先声明该变量。你可以用 set 命令，然后是一个变量名来声明一个变量。变量名可以是由任何一组字母组

成的字符或字符串，包括下划线。变量名也可以包含数字，但数字不能是变量名的第一个字母。变量名中不能有如下字符，如“!”号、“&”字符及空格等。这些变量是 Shell 保留的，只能被 Shell 自己使用。变量名也不能包含几个英文单词，因为系统要在空格处分解命令行，因此空格是命名行中的各项之间的分隔符。下面的例子声明了变量 greeting，你可以用 unset 命令取消该变量的定义。

```
> set greeting
```

你这样用 set 命令给一个变量赋值：首先键入 set 命令，然后键入变量名及赋值操作符“=”，最后是变量要赋与的值。任何字符或字符串都可以赋值给变量。下面的例子中，字符串“hello”被赋值给变量 greeting。

```
> set greeting="hello"
```

在 TCSH Shell 的赋值操作中，你要么在赋值操作符“=”的两边同时加上空格或根本不用加人空格。赋值操作：

```
> set greeting ="hello"
```

将失败，因为在赋值操作“=”之前键入了空格，而在其后却没有。

你可以键入不带任何参数的 set 命令来得到所有已定义的变量列表。在下面的例子中，用户用 set 命令列出了已定义的所有变量及它们的值。

```
> set
greeting    hello
poet Virgil
```

与 BASH Shell 一样，符号“$”是一个特殊操作符，它用来引用一个变量的值，该引用返回一个变量的值，通常是一组字符，然后用这组字符代替变量名。这样，任何在变量名前有“$”字符的地方，变量名将由变量的值所代替。

在下面的例子中，Shell 变量 greeting 被赋以值“ hello ”，然后该值被用作 echo 命令的参数。于是，echo 命令将把这组字符在屏幕上显示出来。

```
> echo $greeting
hello
```

TCSH Shell 脚本程序：输入与输出

你可以在脚本程序中方便地定义和使用各种变量。正如下面的例子一样，你可以用文本编辑器在脚本程序中输入 Linux 命令及进行赋值操作。你可以设置文件的属性为“可执行性”（executable），然后，如同执行一个 Shell 命令一样，你可以在命令提示符后直接键入文件名去运行该文件。记住要设置文件的可执行权限，可以用 chmod u+x 命令或绝对方式 chmod 700 命令去设置它。在脚本程序中，你可以使用 echo 命令进行数据输出，但是，变量的数据读取是通过重定向标准输入来读入的。在 TCSH Shell 中，没有与 BASH Shell 中 read 命令相类似的命令。

TCSH Shell 将检查脚本程序的第一个字符，以决定该文件是否是 TCSH Shell 下的脚本程序。必须记住，所有的 TCSH Shell 脚本程序必须以符号“ # ”为程序的第一行第一个字符，系统将依此来判断该脚本程序是否是一个 TCSH Shell 脚本程序。注意下面脚本程序 greet 的首字符“ # ”。同时，符号“ # ”不只是放在文件的第一行第一字符，它可以和一个普通字符一样，放在文件的任何地方。

greet 脚本程序清单：

```
#
#Script to output hello greeting
```

```
set greeting="hello"
echo The value of greeting is $greeting

>chmod u+x greet
>greet
The value of greeting is hello
```

set 命令与重定向操作符“ $< ”组合将读取用户从标准输入设备输入的任何字符。下面的例子中，greeting 变量读取用户输入的数据：

```
> set greeting = $<
```

在用 echo 命令进行数据输入时，你可以把你的输入字符串与输入提示字符串放置在同行上。

TCSH Shell 在 echo 命令中使用特殊选项-n 来消除输出字符串窜尾的换行符，光标将仍然停留在输出字符串窜尾上。

```
> echo -n Please enter a greeting:
```

如果希望在你的输入提示符字符尾包含一个空格字符，你需在输入特殊符后键入格，然后用双引号括起来。例如：

```
> echo -n "Please enter a  greeting:  "
```

下面的脚本程序 greetpt 中，在输入数据时，TCSH Shell 将使输入提示符与输入数据在同一行上。

greetpt脚本清单：

```
#

echo -n "Please enter a greeting:  "
```

```
set greeting=$<

echo "The greeting you entered was $greeing"

> greetpt
Please enter a greeting:    hello
The greeting you entered was hello
>
```

**参数数组：**argv

**当一个脚本程序被运行时，所有命令行的单词将由空格符分隔为不同的项，这些项将被保存在一个称之为** argv **的数组元素之中，元素** argv [0] **保存命令名，从** argv [1] **开始，每个元素将保存命令行上的每个参数。对所有的** Shell **脚本程序，**argv [0] **将始终保存** Shell **脚本程序名。和其它一些数组一样，你可以用操作算子“**$**”去存取数组** argv **中的所有元素（命令行参数），**$argv [1] **将存取** argv **数组中的第二个元素，即命令行第一个参数。如果不止输入一个参数，你可以通过数组** argv **存取命令行中的每个参数。在下面的例子中，**myargs **脚本程序在屏幕上打印出命令行上的四个参数。**

myargs **脚本程序清单：**

```
#

echo "The first argument is: $argv [1] "
echo "The second argument is: $argv [21] "
echo "The third argument is: $argv [3] "
```

```
echo "The fourth argument is: $argv [4] "

> myargs Hello Hi yo "How are you"
The first argument is: Hello
The second argument is: Hi
The third argument is: yo
The fourth argument is: How are you
>
```

argv 数组中的元素可以被简写为符号“$”加元素的序号。于是$argv [1] 就可以简写为“$1”。因此，TCSH Shell 脚本程序对命令行参数的引用与 BASH Shell 及 PDKSH Shell 对参数的引用类似。特殊参数变量 argv [*] 将引用命令行上所有的参数，$argv [*] 可以简写为“$*”。我们可以注意到，这和 BASH Shell 中引用所有变量的用法相同。

参数变量#argv 保存命令行上所有输入参数的个数。这在为一个脚本程序指定固定的参数数量时很有用。例如：这个数可以用来核实用户是否正确地输入了需要的参数数。下面的脚本程序 arglist 演示了特殊参数变量 argv [*] 及#argv 的使用。程序首先显示命令行上输入的参数的个数，然后在屏幕上显示出输入的这些参数。

arglist 脚本程序清单：

```
#

echo "The number of arguments entered is $# argv"
```

```
echo "The list of arguments is: $argv [*] "

> arglist Hello Hi yo
The number of arguments entered is: 3
The list of arguments is: Hello Hi yo
>
```

## 数字型变量：@

在 TCSH Shell 中，你可以用@命令而不是 set 命令去声明数字型变量，然后，你可以用这些变量进行算术运算、比较运算、逻辑运算等。在这方面，TCSH Shell 程序设计与其它程序设计语言类似。数字变量与字符串变量是两种差别较大的变量，它们的处理方式也有很大的不同。你不能用 set 命令去给一个数字型变量赋值。@命令由关键字“@”、变量名、赋值操作符“=”及算术表达式组成。下面的例子声明了一个数字型变量 num 并把 10 赋值给该变量。

```
> @ num = 10
```

你可以使用各种操作算子，如“+”等算术操作算子。它们的用法与前面介绍的 awk 命令及 C 程序设计中的操作算子的用法相同。表达式中可以是任何算术、比较或逻辑表达式。你也可以用圆括号去创建更复杂的表达式。表达式中的操作数必须用空格把它们与操作算子分隔开，例如 10*5 是一个无效的表达式，必须加入空格，写成 10 * 5。你也可以在表达式中使用数字型变量作为操作数。下面的例子中，count 被声明为一数字型变量，然后表达式中使用了该变量。请注意：变量 count 前使用了“$”操作符，以使表达式使用的是变量的值 3 而不是变量名。表 16-4 是数字型操作算子列表。

```
> @ count = 3
> @ num = 2 * ($count + 10)
> echo $num
26
>
```

## 环境变量：setenv

TCSH Shell 中有两类变量：局部变量及环境变量。局部变量仅仅局限于它所声明的 Shell 的内部，而环境变量类似于全局变量，它的作用范围包括 Shell 本身及其创建的任何子 Shell，但不包括其父 Shell。环境变量有 setenv 命令定义，你可以用如下方式给一个环境变量赋值：setenv 命令，然后是变量名，最后是变量将要赋予的值，其中没有任何操作算子。下面的例子中，greeting 环境变量被赋值以“Hello”。

```
> setenv greeting Hello
```

无论什么时候调用一个 Shell 脚本程序，该 Shell 脚本程序会创建它自己的 Shell。如果一个 Shell 脚本程序是从另一个脚本程序中被调用并执行的，前者有它自己的 Shell，且该 Shell 有独立于后者的 Shell。此时有两个 Shell，第一个脚本程序的 Shell（称为父 Shell）及在调用第二个脚本程序时创建的新 Shell（称为子 Shell）。当一个脚本程序是被另一个脚本程序调用并执行的，那么，我们称前者创建的 Shell 是后者的子 Shell，后者称为父 Shell。

每个 Shell 都有它自己的一组变量，子 Shell 不能存取父 Shell 中的局部变量，但它可以引用父 Shell 中的环境变量。所有父 Shell 中声明的环境变量都可

以被它们的子 Shell 引用。

### 16.7.2 控制结构

和 BASH Shell 及 PDKSH Shell 一样，TCSH Shell 也有一组控制结构，这组控制结构能让你控制脚本程序中命令的执行，例如循环和条件控制结构。其中，循环结构用来多次执行某些 Linux 命令，条件结构用来控制、选择你要执行的命令；while 及 if 控制结构是更常用的控制结构，while 控制结构用来重复执行命令，而 if 控制结构根据表达式的值的真假来选择执行相应的命令；switch 和 foreach 控制结构更是为适应特殊目的而提供的两种控制结构，switch 控制结构把表达式的值与一组该表达式可能的值相比较，然后再执行相应的命令。foreach 结构是一种有限次循环结构，每次循环时，它把一系列值（有限个）中的一个值赋值给变量，然后执行循环体内的命令，直到这些值全被赋值完为止。

TCSH Shell 与其它 Shell 的不同之处在于它的控制结构提供了比其它 Shell 程序设计语言更多的格式。TCSH Shell 控制结构的测试条件是一个被赋值以真（True）或假（False）的表达式，而不是一个 Linux 命令。TCSH Shell 与 BASH Shell 的一个最大不同之处在于 TCSH Shell 结构不能用进行重定向与管道输出，它们是更严谨的控制结构，以控制命令的执行。

#### 测试表达式

if 和 while 控制结构用表达式作为它们的测试条件。当表达式的值为非 0 值时测试条件为真（true），当表达式的值为 0 时为假（false）。在 TCSH Shell 中，比较及等号等符号通常被方便地用来在测试表达式中，因为这种表达式的

结果通常是 1（true）或 0（false）。你可以在表达式中使用各种操作算子。测试表达式也可以是算术或字符串比较，但字符串比较仅可以用等号或不等号。

与 BASH Shell 及 PDKSH Shell 不一样，在 TCSH Shell 中，你必须用圆括号“（”和“）”把 if 及 while 控制结构中的测试表达式括起来。下面的例子中的测试表达式用来判断两个字符串是否相等。

```
if ( $greeting  ==  "hi" ) then
echo Informal Greeting
endif
```

TCSH Shell 中有一些测试算子用来测试字符串是否与字符串或正则表达式（Regular expression）相等或不等。算子“==”与“!=”分别用来判断两字符串相等或不等。算子“=~”与“!~”分别用来判断字符串与正则表达式是否相匹配或判断两者之间是否不匹配。这里的正则表达式可以包括任何 Shell 特殊字符。在下面的例子中，如果变量 greeting 的值是任何以大写字母 H 或小写字母 h 开头的字符串，那么它将和正则表达式 [Hh] *相匹配。

```
if ( $greeting  =~  [Hh] * ) then
echo Informal Greeting
endif
```

与 BASH Shell 一样，TCSH Shell 也有几个特殊算子来测试文件的存取状况。许多这些操作算子是相同的。在下面的例子中，if 命令用来测试文件 mydata 是否是可读的。表 16-5 中文件存取状况的一些特殊算子。表 16-6 列出了 TCSH Shell 中的一些命令与变量。

```
if (  -r mydata  ) then
```

```
echo Informal Greeting
endif
```

## Shell 条件控制结构：if-then、if-then-else 及 switch

TCSH Shell 有一组条件控制结构，这些条件结构用来控制、选择你要执行的 Linux 命令。这些条件控制结构的用法大部分与在 BASH Shell 中的条件控制结构的用法相类似，但有一些有重大差别。例如，TCSH Shell 中的 if 条件控制结构以关键词 endif 为结束；switch 控制结构中使用 case 关键词，并且它以关键词 endsw 为结束，而在各 case 之间用 breaksw 来表明一个 case 段的结束。此外，有两种 if（条件）控制结构：一种是简单型条件控制结构，它仅用来执行一条命令；另一种是复杂型条件控制结构，它可以执行好几个命令或可选命令。简单型条

件控制结构由关键词 if、然后是测试条件及单个 Linux 命令组成。复杂型的条件控制结构以

关键词 endif 为结束。TCSH Shell 的条件控制结构的用法在表 16-7 中列出。

### if-then

if-then 控制结构可以为几个 Linux 命令设置条件，该条件是一个表达式。如果表达式的值不是 0，则表达式的值为真（true），于是 if 控制结构内的命令将被执行；如果表达式的值是 0，则表达式的值为假（false），if 控制结构内的命令将不被执行；if-then 控制结构由关键词 if 开头，关键词后是由圆括号“（”、“）”括起来的一个表达式，紧接其后是关键词 then，然后，你可以在此后的行上加入任何条 Linux 命令。关键词 end if 将结束 if 控制结构。需要注意的是：

在 BASH Shell 中，关键词 then 是单独另起一行，而在
TCSH Shell 中，then 和表达式在同一行上。if-then 控制结构的句法如下：

```
if （Expression） then
Commands
endif
```

下面的 ifls 脚本程序将列出文件及其大小。如果你在程序提示符后键入字符“s”，当前目录下的每个文件及该文件所占用的块数将会被列出，如果用户输入字符“s”以外的任何字符，if 测试条件将不成立，脚本程序将不执行任何命令。

ifls 脚本程序清单：

```
#
echo -n "Please enter option: "
set option=$<

if ( $option == "s" ) then
  echo Listing files by size
  ls -s
endif
```

```
> ifls
Please enter option: s
Listing files by size
```

```
total 2
 1 monday   2 today
>
```

if-then-else

通常，你需要在一个表达式值的真、假这两者之间作出选择，而关键词 else 允许 if 控制结构在上述两者之间作出选择。如果表达式的值为真（true），在紧跟测试表达式及关键词 then 之后的命令将立即被执行，如果表达式的值为假（false），在紧跟关键词 else 之后的命令将被执行。if-then-else 控制结构的句法如下：

```
if ( Expression)then
Commands
else
Commands
endif
```

下面的脚本程序 elsels 在执行 ls 命令的时候将有两种可能的选项，在列出当前目录下的文件时，或者是同时列出文件的大小，或者是列出文件的所有信息。如果用户在程序提示符后键入字符“s”，将同时列出文件的大小，否则列出文件的所有信息。需要注意的是，请读者注意该脚本程序与前一章 BASH Shell 中介绍的 elsels 脚本程序的句法及结构有什么不同。

elsels 脚本程序清单：

```
#
echo Enter s to list file sizes,
```

```
echo otherwise all file information is listed.
echo -n "Please enter option:    "
echo option=$<

if ($option == "s" ) then
ls -s
  else
  ls -l
endif
echo Good-bye

> elsels
Enter s to list file sizes,
otherwise all file information is listed.
Please enter option:    s
total 2
1    monday     2 today
Good-bye
>
```

switch

switch 控制结构从可能的几个结果中选择相应要执行的命令。它和 BASH Shell 中的 case 控制结构非常类似，它根据一个字符串与几个可能的模式（pattern）之间的比较结果作出选择。每个可能的模式都对应于一组命令，当

找到与字符串匹配的模式的时候，与该模式对应的命令将被执行。switch 控制结构以关键词 switch 开头，紧接其后是用圆括号括起来的字符串，该字符串通常是来自赋值了的变量。此后是一组模式，每一个模式之前是关键词 case，其后以符号“:”结束。与模式对应的命令在符号“:”后列出，且这些命令以关键词 breaksw 结束。最后以关键词 endsw 结束 switch 控制结构。switch 控制结构的句法如下：

```
switch  ( test-string )
case pattern:
commands
breaksw
case pattern:
commands
breaksw
default:
commands
breaksw
endsw
```

每一个模式将与 test-string 相匹配，直至找到与 test-string 相匹配的模式为止。如果没找到与之匹配的模式，缺省的项将被执行，该缺省项由关键词开头，且是可选的，你也可以没有该项，但是，告诉用户没有找到与 test-string 匹配的模式是有用的。

switch 控制结构通常被用来实现菜单。下面的例子程序 lschoice 要求用户

输入以不同方式列出当前目录下的文件。必须注意，default 项的作用在于对无法找到匹配的字符串输入提出警告。

lschoice 脚本程序清单：

```
#
echo s. List Sizes
echo l. List All File Inforamtion
echo c. List C Files

echo -n "Please enter choice:   "
set choice=$<

switch ($choice)
case s:
ls -s
breaksw
case l:
ls -l
breaksw
case c:
ls *.c
breaksw
default:
echo Invaid Option
```

```
breaksw
endsw

> lschoice
s. List Sizes
l. List All File Inforamtion
c. List C Files
Please enter choice: c
io.c   lib.c   main.c
>
```

## 循环：while 和 foreach

TCSH Shell 中有一组循环控制结构，这些循环控制结构允许你多次重复执行 Linux 命令，它们是：while、foreach 及 repeat。TCSH Shell 的控制结构在表 16-7 中列出。while 控制结构与 C 语言中的 switch 语句类似，与 TCSH Shell 中的 if 控制结构一样，while 控制结构首先检验表达式的值；TCSH Shell 中的 foreach 控制结构与 BASH Shell 中的 for 和 for-in 控制结构一样，不进行任何检验，它仅仅逐步把一系列值依次赋值给指定的变量。在这方面，foreach 控制结构与其它程序设计语言中的相关语句有很大不同。

### while

while 控制结构用来重复执行 Linux 命令。一个 while 循环以关键词 while 开头，然后紧接其后是由圆括号括起来的表达式，以关键词 end 结束 while 循

环。while 循环控制结构的句法如下：

```
while ( expression )
commands
end
```

我们可以用 while 循环控制结构及 switch 控制结构的组合编制菜单驱动程序。注意下面的例子包含菜单 quit 选项，该菜单将赋值 no 给变量 again，并退出循环。

lschoice 脚本程序清单：

```
#
set again=yes

while ($again == yes )
echo "1.List Sizes"
echo "2.List All File Information"
echo "3.List C Files"
echo "4.Quit"
echo -n "Please enter choice:   "
set choice = $<

switch ($choice)
case 1:
ls -s
```

```
breaksw
case 2:
ls -l
breaksw
case 3:
ls *.c
breaksw
case 4:
set again = no
echo Good-bye
breaksw
default:
echo Invalid Option
endsw
end

>lschoice
1.List Sizes
2.List All File Information
3.List C Files
4.Quit
Please enter choice: 3
main.c    lib.c    file.c
1.List Sizes
2.List All File Information
```

```
3.List C Files
4.Quit
Please enter choice: 4
Good-bye
>
```

## foreach

foreach 控制结构用来把一系列的值逐步赋值给某变量。它与 BASH Shell 中的 for-in 控制结构非常类似。foreach 结构有两个操作数：变量及由圆括号括起来的一系列值（组成一列表），该列表中的每项都将被赋值给 foreach 结构里的变量。与 while 控制结构一样，foreach 控制结构是一种循环结构，每循环一次，列表中下一项将赋值给变量，直至最后一项，然后终止循环。与 while 循环控制结构一样，foreach 控制结构的循环体以关键词 end 结尾。switch 控制结构的句法如下：

```
foreach variable ( list of values )
commands
end
```

在下面的例子脚本程序 mylist 中，用户要求简单地输出列表项中的每一项与当前的日期。

mylist 脚本程序清单：

```
#
set tdate='date'+%D"
```

```
foreach grocery ( milk cookies apples cheese )
echo "$grocery $tdate"
end

$ mylist
milk 12/23/96
cookies      12/23/96
apples       12/23/96
cheese        12/23/96
$
```

foreach 循环控制结构在管理文件时非常有用。在 foreach 控制结构中，你可以使用 Shell 特殊字符来生成由一系列文件名组成的列表，该列表中的每一个文件名将被赋值给 foreach 控制结构中的变量。例如星号（“*”）将生成由所有文件名及目录组成的列表；*.c 将生成由所有的以.c 为扩展名的文件组成的列表，这些文件通常是 C 程序源文件。下面的例子将把每一个文件都拷贝一备份，并把这些备份放置到名为 sourcebak 的目录下：模式*.c 生成由一系列文件名组成的列表，列表中的每一项被赋值给 foreach 循环控制结构体中的变量。

cbackup 脚本文件清单：

```
#

foreach backfile (*.c)
```

```
 cp $backfile sourcebak/$backfile
 echo $backfile
end

>cbackup
io.c
lib.c
main.c
```

**没有指定列表项的值的** foreach **控制结构将把命令行参数作为其列表项。此时，在运行** shell **文件时在命令行上指定的参数将成为列表中的项，并将把这每个命令行参数值按次序依次赋值给** foreach **控制结构中的变量。第一次循环时把第一个参数赋值给** foreach **循环控制结构变量，第二次循环时把第二个参数赋值给** foreach **循环控制结构变量，依次类推。**

**在下面的脚本程序** mylistarg **中，**foreach **循环控制结构列表中的值没有指定，但** foreach **控制结构将连续把命令行中的参数赋值给** grocery **变量，直至最后一个参数，然后结束循环。**

mylistarg **脚本程序清单：**

```
#
set date='date'+%D"

foreach grocery ( )
echo "$grocery   $tdate"
end
```

```
> mylistarg milk cookies apples cheese
milk12/23/96
cookies  12/23/96
apples   12/23/96
cheese   12/23/96
>
```

你也可以用特殊参数变量 argv [*] 来明确地把命令行参数作为 foreach 循环控制结构中的列表项。下面的例子中，在运行 cbackuparg 时，在命令行上加入了一组 C 程序文件名，在 foreach 循环控制结构中，argv [*] 将代表命令行上所有的输入参数，每个参数将被依次赋值给变量 backfile。第一次循环时 $backfile 等同于$argv [1] ，第二次循环时$backfile 等同于$argv [2] 。

变量 argnum 用来引用每个参数。文件运行的结果显示表明：foreach 循环控制结构中的变量 backfile 的值与输入命令行参数是一致的。

cbackuparg 脚本程序清单：

```
#

@argnum = 1
foreach backfile ( $argv [*]  )
 cp $backfile sourcebak/$backfile
 echo "$backfile $argv [$argnum] "
 @argnum = @argnum + 1
end
```

```
> cbackuparg main.c lib.c io.c
main.c main.c
lib.c lib.c
io.c io.c
```

## 16.8 小结

你可以用 TCSH Shell 中的 history 实用程序列出、查找你先前运行过的命令，你甚至可以编辑这些命令，然后运行这些编辑后的命令。用 TCSH Shell 中的 alias 命令，你可以给一个命令取别名，这个别名可以是一个命令甚至是带参数的该命令本身。在 TCSH Shell 中，有一组可开启、关闭的特征开关变量。常见的特征开关变量有：ignoreeof、noclobber、noglob。特征变量 ignoreeof 可以防止意外退出；noclobber 特征变量可以防止在进行重定向操作时覆盖已经存在的文件；noglob 特征变量可把一些特殊字符仅当作普通字符处理。

与 BASH Shell 及 PDKSH Shell 一样，TCSH Shell 也具有和其它编程语言一样的编程能力。你可以定义变量并给这些变量赋值，你也可以使用控制结构去实现循环与条件控制。在脚本程序中，你可以定义变量、嵌入 shell 命令、设置条件控制结构，也可以象运行其它程序一样去运行它们。

定义的变量可以是字符型变量或数字型变量。声明并给字符型变量赋值时要用 set 命令，而声明数字型变量并给它们赋值时要用@命令。@命令允许你

用复杂的数字、关系及位运算表达式的值来给一数字型变量赋值。你也可以定义环境变量。在该 Shell 下创建的所有子 Shell 中，环境变量都是可访问的。环境变量的定义与赋值需使用 setenv 命令，但该命令更常用来给特殊环境变量赋值。

在 TCSH Shell 中，循环与条件控制结构与其它程序设计语言的循环与条件结构类似。TCSH Shell 中的循环控制结构有：while 与 foreach 控制结构，条件控制结构有 if与 switch。if及 while 控制结构与其它程序设计语言中的 if与 while 结构完全一样，它们以测试表达式的值为假来结束循环。如果测试表达式的值为真，if 条件控制结构将执行其控制结构内的命令。测试表达式是一个 TCSH Shell 表达式，该表达式中的操作算子与 C 程序设计语言中的表达式操作算子类似。在 TCSH Shell 的表达式中可以有许多赋值、算术、比较及位运算算子，而 BASH Shell 及 PDKSH Shell 的表达式中，有些算子是不能用的。与 C 程序设计一样，当表达式的值为非 0 时，TCSH Shell 表达式为真，反之，当表达式的值为 0 时，TCSH Shell 表达式为假。

switch 控制结构是 if 结构的一种受限形式。switch 控制结构把字符串与一组可能的模式（pattern）相比较，如果找到与字符串匹配的模式，则与该模式相关联的一组操作将被执行。switch 控制结构在要实现菜单（即要用户在可能的几个选项中作出选择）时特别有用。foreach 控制结构用来把一列表中的所有项依次赋值给指定的变量，该控制结构它没有测试条件。列表中的项可以由 Shell 特殊字符来生成或直接指定。例如星号（“*”）将生成一个由所有文件与目录名组成的列表；同时，列表中的项也可以由一组字符串直接指定，然后这些指定的字符串将依次赋值给 foreach 控制结构中的变量。foreach 控制结构将一直

循环下去，直到列表中的最后一项也被赋值完为止。

附表：

表 16-1TCSH Shell history 实用程序命令

| 事件引用 | 功能 |
| --- | --- |
| ! event num | 通过历史事件号引用历史事件 |
| ! characters | 通过历史事件的字符串前缀引用历史事件 |
| ! ?pattern? | 通过模式来查询历史事件列表中的事件 |
| ! -event num | 通过历史事件号偏移来引用历史事件 |
| ! num1-num2 | 引用历史事件号在 num1 与 num2 之间的事件 |
| **事件字符串引用** | |
| ! event num:word num | 引用事件号为 num 的历史事件中的特定单词（word） |
| ! event num: ^ | 引用事件号为 num 的历史事件中的第一个参数（第二个单词） |
| ! event num:$ | 引用事件号为 num 的历史事件中的最后一个参数 |
| ! event num:^-$P | 引用事件号为 num 的历史事件中所有的参数 |
| ! event num:* | 引用事件号为 num 的历史事件中所有的参数 |
| **事件编辑与替换** | |
| ! event num:s/pattern/newtext/ | 用模式替换来编辑一历史事件。用来定位历史事件中特定的单词 |

| | |
|---|---|
| ! event num:sg/pattern/newtext/ | 用模式来替换历史事件中所有匹配 |
| ! event num:s/pattern/newtextp | 禁止编辑后历史事件的执行 |

表 16-2TCSH Shell 特征变量

| 命令及特征开关变量 | 功能 |
|---|---|
| set 与 unset | 命令用 set 或 unset 命令来打开或关闭 Shell 特征变量<br>$ set feature-variable<br>$ set noclobber 打开 noclobber 特征变量<br>$ unset noclobber 关闭 noclobber 特征变量 |
| echo | 在执行该命令之前回显该命令行 |
| ingoreeof | 禁止使用 CTRL-D 退出 |
| noclobber | 重定向操作时不覆盖已存在文件 |
| noglob | 禁止文件名中特殊字符扩展（如*、?、~、及 [ ]） |
| notify | 后台任务结束后立即通知用户 |
| verbose | 历史命令引用后回显该命令 |

表 16-3TCSH Shell 特殊变量

| 系统自定义变量 | 简要说明 |
|---|---|
| home | HOME 用户主目录 |

| | | |
|---|---|---|
| | LOGNAME | 用户登陆名 |
| Cwd | 当前工作目录 | |
| 可重定义特殊变量简要说明 | | |
| shell | SHELL | 登陆 Shell 目录路径及程序名 |
| path | PATH | 执行命令时要搜索的目录路径名列表 |
| prompt | 首选命令提示符 | |
| mail | mail 实用程序在收到消息时要检查的邮件文件名 | |
| 用户级定义变量简要说明 | | |
| cdpath | 执行 cd 命令时要搜索的子目录路径名 | |
| history | 历史事件列表中要保存的历史事件数 | |
| savelist | 为下次登陆时保存的历史事件数，这些历史事件被保存在文件.history 中 | |
| EXINIT | Ex/Vi 编辑器的初始化命令 | |
| TERM | 终端名 | |

表 16-4 数字算子

| 赋值算子 | 功能/说明 |
|---|---|
| = | 赋值符 |
| += | 与表达式相加，然后赋值 |

| | |
|---|---|
| -= | 与表达式相减，然后赋值 |
| *= | 与表达式相乘，然后赋值 |
| /= | 与表达式相除，然后赋值 |
| %= | 表达式取模，然后赋值 |
| ++ | 变量加一 |
| -- | 变量减一 |
| - | 负号算子 |
| + | 加 |
| - | 减 |
| * | 乘 |
| / | 除 |
| % | 取模 |
| **关系操作算子** | |
| > | 大于 |
| < | 小于 |
| >= | 大于等于 |
| <= | 小于等于 |
| != | 不等于 |
| == | 等于 |

表 16-5 表达式操作算子

| 字符串比较 | 功能/简要说明 |
| --- | --- |
| = = | 字符串相等 |
| != | 字符串不等 |
| =~ | 字符串与模式比较，测试两者是否相等，模式可以是任何常规表达式 |
| !~ | 字符串与模式比较，测试两者是否不相等，模式可以是任何常规表达式 |
| **逻辑操作算子** | |
| && | 逻辑与 |
| \|\| | 逻辑或 |
| ! | 逻辑非 |
| **文件测试** | |
| -e | 文件存在 |
| -r | 文件是只读 |
| -w | 文件可以被修改（写入） |
| -x | 文件可以被执行 |
| -d | 文件是一个目录名 |
| -f | 文件是一个二进制文件 |
| -o | 文件为用户所有 |
| -z | 文件为空 |

表 16-6TCSH Shell 命令与变量

| Shell 命令 | 功能 |
| --- | --- |
| echo | 显示变量之值，-n 选项禁止输出至新的一行 |
| eval | 执行命令行 |
| exec | 在当前进程中执行命令(使用当前 Shell 而不生成新的子 Shell) |
| exit | 从当前的 Shell 中退出 |
| setenv va | r 使变量 var 可以被每个新创建的子 Shell 存取 |
| printenv | 显示所有环境变量的值 |
| set | 给变量赋新的值，不带任何参数的 set 命令将列出当前所有定义的变量 |
| @ | 赋值与数字表达式 |
| shift | 把所有的命令行参数左移，于是引用该参数时的参数号将比先前小 1，也就是参数$3 将赋值给参数$2，依次类推。先前的参数$1 将丢失 |
| unset | 取消一个变量的定义 |
| unsetenv | 取消一个环境变量的定义 |
| **命令行参数** | |
| $argv［0］ $0 | Linux 命令名 |

| | |
|---|---|
| $argv [n] $n | 从 1 开始的第 n 个命令行参数，你可以用 set 命令去改变它们的值 |
| $argv [*] $* | 从 1 开始的所有命令行参数，比可以用 set 命令去改变它们的值 |
| $#argv $# | 命令行参数的个数 |

表 16-7TCSH Shell 控制结构

| 条件控制结构： | |
|---|---|
| if-then，esle，switch<br>if ( expression ) then<br>commands<br>endif | 如果 expression 为真，其后的命令将被执行。可以输入多个 Linux 命令 |
| if ( expression ) then<br>command<br>esle<br>command<br>endif | 如果 expression 为真，then 后的命令将被执行。如果 expression 为假，else 后的命令将被执行 |
| switch (string)<br>case pattern:<br>command<br>breaksw<br>default:<br>command | 可以使你在多个命令之中作出选择的循环控制结构 while 及 foreach |

| | |
|---|---|
| endsw | |
| while (expression)<br>command<br>end | 只要 expression 的值为真，expression 后的命令将永远被执行 |
| foreach variable (arg-list)<br>command<br>end | arg-list 列表中有多少项就执行多少次该循环。每执行一次循环，列表中一项将被赋值给变量 variable。与 Bourne Shell 中的 for-in 控制结构类似 |