微机上的 UNIX 操作系统
Linux 使用指南

王小鸽
周桂林
孟　杰　编著
孙　彤
余　翔

清华大学出版社

(京)新登字 158 号

内 容 简 介

Linux 操作系统是一个可从 Internet 上免费获取的可在微机上安装运行的 UNIX 操作系统的完整实现。本书分两部分介绍这一操作系统。在第 1 部分中,首先简要介绍了 Linux 的发展历史、Linux 的功能、安装运行 Linux 所需要的硬件配置,然后系统介绍 Linux 的获取、安装和使用操作,最后介绍 Linux 的多用户系统管理。第 2 部分较详细地介绍 Linux 的网络功能,包括网络软件的一些基本概念、系统的网络配置以及一些常用的网络软件。这部分内容将为那些准备把微机配置成网络站点机的读者提供非常有用的参考。为了适应 Linux 操作系统不断发展的特点,在本书的附录里为读者提供了从 Internet 上获取更多有关 Linux 资料的方法以及 FTP 站点和 BBS 站点的目录。

本书适用于科技人员、教师、学生及广大计算机开发或使用人员。

版权所有,翻印必究。
本书封面贴有清华大学出版社激光防伪标签,无标签者不得销售。

图书在版编目(CIP)数据

微机上的 UNIX 操作系统 Linux 使用指南/王小鸽等编著 .—北京: 清华大学出版社, 1997.4

ISBN 7-302-02496-0

Ⅰ.微… Ⅱ.王… Ⅲ.UNIX 操作系统-分时操作系统-指南 Ⅳ.TP316.1-62

中国版本图书馆 CIP 数据核字(97)第 07604 号

出版者: 清华大学出版社 (北京清华大学校内,邮编 100084)
印刷者: 北京丰华印刷厂
发行者: 新华书店总店北京科技发行所
开 本: 787× 1092 1/16 印张: 15.25 字数: 362 千字
版 次: 1997 年 8 月 第 1 版 1997 年 8 月 第 1 次印刷
书 号: ISBN 7-302-02496-0/TP·1263
印 数: 0001—5000
定 价: 19.50 元

第 1 部分　入　門　篇

第 1 章　Linux 操作系统简介

1.1　Linux 的简要历史

由于它的庞大的支持基础和发行系统,UNIX 成为世界范围内最有影响(最广泛使用)的操作系统之一。起初 UNIX 是作为小型机和大型机上的多任务系统而开发的,尽管它有一些含糊不清的接口和缺少标准化等缺点,但是它仍然很快地发展成为广泛使用的操作系统。

许多计算机爱好者感到 UNIX 正是他们所想要的东西——一个真正的操作系统。于是,一群 UNIX 的爱好者要亲自在他们自己的机器上实现 UNIX,从而开始了 Linux 的开发。

UNIX 在许多机器上有各种版本——从个人计算机至超级计算机,如 Cray Y-MP。大部分个人计算机上的 UNIX 版本是很昂贵的。大约两年前,一个 386 上的 AT&T 系统 V 单机版本大约要花 1 500 美元。

Linux 操作系统是一个免费发行的 UNIX 操作系统在微机上的实现,它是由芬兰赫尔辛基大学的 Linus Torvalds 先生首先于 1991 年开始开发的。Linux 的开发得到了遍布于 Internet 上的许多 UNIX 程序员和爱好者的帮助。借助于 Internet,任何只要是懂得如何去作并有能力和兴趣的人都可以参与 Linux 的开发。Linux 的核心部分没有使用任何来源于 AT&T 或其他有专利的代码。

大部分 Linux 上能用到的软件均来源于著名的位于美国麻省 Cambridge 的 GNU 工程及免费软件基金会。总之,遍及全世界的程序开发人员都为这一不断成长的 Linux 软件库作出了贡献。

早期的 Linux 是 Linus Torvalds 出于个人兴趣爱好于 1991 年开始着手进行开发的,后来受到由 Andy Tanenbaum 开发的名为 Minix 的小型 UNIX 系统的启发和鼓励而进一步发展起来。首次关于 Linux 的讨论是在 USENET 的 comp. os. minix 新闻组上展开的,讨论主要是关于开发一个小型的,为学术界所用的,称为 Minix 的 UNIX 操作系统的变体。参与讨论的 Minix 用户希望有一个比 Minix 功能更多的为学术界用的真正的 UNIX的实现。

最早期 Linux 的开发是从 80386 的保护模式界面上的任务转换功能开始的。全部用汇编语言编写。从那以后的工作,Linus Torvalds 在 Comp. os. minix 新闻组上有这样一段叙述:“从那以后仍然是很繁锁的编程工作,但是我有一些设备使得调试能够容易地进行。从这时起,我开始用 C 语言编程,这显然加速了编程工作。我开始认真考虑开发一个比 Minix 更好的 Minix 这一大胆想法。我希望有朝一日能在 Linux 下编译 gcc…。我用了两个月的时间作了一些基本的准备工作,然后又用了稍长一点的时间,我有了第一个硬盘驱动程序(尽管它还有许多毛病,但它当时在我的机器上能正常工作了)和一个小型文件

管理系统。我完成0.01版本的时间大约在1991年8月下旬,当时这个系统并不漂亮,它还没有软盘驱动器,而且干不了太多事。我想大概从来就没有人编译并安装过那个版本。尽管这样,那时我已经陷进去而无法自拔了,我觉得不到我能彻底抛弃Minix时我是不会停止这项工作的。”

Linux 0.01版本从来没正式宣布出版过。事实上这个版本无法独立运行。它仅仅是一个核心程序的雏型,需要Minix的支持才能进行编译和运行。

1991年10月5日,Linus Torvalds正式宣布了Linux的第一个正式版本——0.02版本。这个版本能够运行bash和gcc。他在comp.os.minix新闻组上宣布他的Linux 0.02版本时写道:“你是否期待着Linux 1.1到来的美好日子?期待着那种由使用者自己写设备驱动程序的时光?你是否尚未找到一个在计算机上干的有趣的项目?你是否正为在操作系统上找个题目能使你对其进行修改而为你所用?你是否正为未能在Minix上找到这样的题目而感到烦恼?那么,下面这个布告也许正是你要寻找的:正如我一个月之前提到的,我正在着力编写一个为AT-386计算机所用的看上去像Minix的免费的操作系统,它终于达到了在某种程度上可以使用的阶段。为了更广的发行我愿意献出源程序代码。这仅仅是0.02版本……但是我已经成功地在它下面运行了bash, gnu-make, gnu-sed, compress等等。”

由此可见,Linux操作系统从一开始就是一个编程爱好者的系统。它的出发点在于核心程序的开发,而不是对用户系统的支持。至今,在Linux世界里似乎仍然保持了这种把对用户的支持服务当成次要问题,而核心程序的开发才是真正的中心任务的风气。

在Linux的0.03版本之后,越来越多的人陆续参与了该系统的开发工作。很快Linus Torvalds就把版本升到0.10,在陆续几个版本之后,他又把版本升成0.95。这反映了他对这个操作系统的期望。他希望很快就能有正式的发行版。一般来说,软件要到理论上已经完备或者已经没有毛病的时候才给予1.0版本号。这是在1992年的3月间。又过了一年多,于1993年12月下旬,Linux的核心程序仍只是0.99 p/14版,正逐步地趋于1.0。现在,在我们着手编写此书的时候,Linux已经达到了2.2.0版本了。现今,Linux已经完全是UNIX的一个完整的实现了。它能够运行X窗口系统,TCP/IP网络协议,Emacs, UUCP,电子邮件和新闻组等许多软件。几乎所有免费的软件都已移植到Linux上,且一些商业软件也正在推出。越来越多的硬件得到了系统的支持。许多人在80486的Linux上运行Benchmark,并发现它们与中等规模的工作站(从SUN到DEC)不相上下。谁能猜想到,几年前一个小小的操作系统而今能成长到如此规模并且在个人计算机世界占领了相当的地位。

1.2 Linux的功能

Linux操作系统支持几乎所有在其他UNIX操作系统的实现上所能找到的功能,另外还包括一些在UNIX系统的其他实现版本上没有的功能。这里简单地浏览一下Linux核心程序的一些主要功能。

正如所有其他版本的UNIX一样,Linux是一个完全多任务、多用户的操作系统,也

就是说,允许多用户同时登录到一台机器上同时运行多道程序。

Linux 在源代码级几乎与一些 UNIX 标准兼容,其中包括 IEEE POSIX.1、AT&T 系统 V 和 BSD。在它的开发过程中一直以源代码的可移植性为原则。因此,你很容易发现在一些不同的实现上共享的一些共同的特性。

大量的在 Internet 上或其它地方可以获取的 UNIX 上的免费软件同样可以在 Linux 上编译运行。此外,所有的源程序,包括核心程序、设备驱动程序、软件库用户程序和开发工具等的源程序等都是免费可取的。

其他一些 Linux 系统内部特有的功能,如 POSIX 任务控制(用于如 csh 和 bash 等的 shell)、虚拟终端(pty 设备)及对使用动态加载的一些特定键盘的支持。Linux 还支持虚拟控制台。这种虚拟控制台可使用户在多个登录上进行转换。DOS 的用户会发现这种 Linux 虚拟控制台功能与 screen 的功能相似。

Linux 的核心能仿真 387-FPU 指令,所以,即使没有数学辅助处理器,系统仍然可以运行需要浮点运算的数学指令的程序。

Linux 可以支持各种类型的文件系统。各种文件系统,如 ext2fs 文件系统已经被设计为 Linux 专用。其它文件系统,如 Minix-1 和 Xenix 文件系统也都为 Linux 所支持。Linux 不仅支持 MS-DOS 文件系统,同时还允许用户从硬盘或软盘上直接存取 MS-DOS 的文件。可以读各种标准的 CD-ROM 的 ISO-9660-CD-ROM 文件系统类型也为 Linux 所支持。我们将在第 2 章和第 4 章介绍关于 Linux 的文件系统。

Linux 提供了 TCP/IP 网络协议的完备实现。包括了许多广为应用的以太网卡、SLIP,PLIP,NFS 等设备驱动程序。同时也支持完备的 TCP/IP 客户与服务器功能,诸如 FTP,telnet,NNTP 和 SMTP 等。在本书的第 2 部分将详细介绍 Linux 上的网络软件和管理。

Linux 的核心的开发利用了 Intel 80386 和 80486 处理器的保护模式。特别是利用了保护模式基于描述词的存储器管理模式以及这两类处理器所具有的其他许多先进的功能。了解如何在 80386 保护模式下编程的人都知道这种芯片的设计是考虑到了诸如 UNIX或 Multics 这样的多任务系统的实现的。Linux 充分利用了芯片的这一功能。

Linux 核心还支持了只把可执行代码的需要执行的部分页面调入内存的管理策略。也就是说,只有真正被用到的程序部分才被从硬盘上读进内存。此外,有些页面是被多个可执行文件共享的。也就是说在物理存储器上如果有一段可执行代码同时被几个任务执行,这些任务将共享一份代码,这样就减少了内存的使用。

为了增加可用内存空间,Linux 同时也实现了硬盘的分页管理:也就是说在硬盘上可分配多达 256MB 的交换空间。当系统申请更大的物理存储空间时,它会把暂时不用的页面交换至硬盘。这样就能运行较大的程序和同时支持多用户。然而,要记住“交换空间”并不能取代物理随机存储器,因为随机存储器的存取速度要快得多。

核心程序也实现了用户程序和硬盘高速缓存的统一的存储池管理。用这种管理方式,所有的未被使用的内存空间全部用于进行高速缓存,而当需要运行大的程序时再减小高速缓存使用的内存空间。

可执行代码利用了共享程序库的动态连接方式,也就是说允许多个可执行代码共用

一份可共享的程序库文件里的程序库代码,正像 SunOS 的共享程序库机制。用这种链接方式可以使可执行文件占用少得多的硬盘空间,特别是那些调用许多库程序的大型程序。还有一种静态连接程序库。对于那些希望能用目标调试(object debugging)或者要保留一份完整的可执行代码而不需要有共享程序库同时存在的用户来说,可以用静态连接程序库。Linux 共享程序库是在运行时动态连接的,可允许程序员用他们自己的程序库来取代共享程序库的模块。

1.3 Linux 上的常用软件

这一节将介绍 Linux 上的许多应用程序和一些常用的计算功能。一个操作系统最重要的部分就是它所支持的各种界面及广泛的应用软件,而这些软件是免费的,这一条更是吸引人。

1.3.1 基本命令和工具

在标准的 UNIX 系统上可以找到的应用软件几乎都已移植到 Linux 上了,包括了一些基本命令,如 ls, awk, tr, sed, bc, more, 等。因此,毫不奇怪,一个 UNIX 的用户会觉得整个 UNIX 工作环境已经在 Linux 上重现了——所有的标准命令和应用程序都已经在那儿了。(不熟悉 UNIX 的用户请参阅第 3 章有关 Linux 的基本用法介绍。)

Linux 上有许多文本编辑程序,包括 vi, ex, pico, jove 以及 GNU Emacs 和它的其他版本,如 Lucid Emacs 和 Joe。不论你熟悉的是什么样的文本编辑程序,几乎都已移植到 Linux 上了。

文本编辑工具的选择是一个很有趣的事。许多 UNIX 的老用户仍然在用着最简单的如 vi 这样的编辑工具。由于 vi 有许多局限性,使得如 Emacs 这样的功能更为复杂的编辑程序正在日益被广泛接受。Emacs 可以支持一个完备的 LISP 为基础的宏语言和解释器,一个功能很强的命令语法以及其他一些有用的扩充。利用 Emacs 的宏可以阅读电子邮件及新闻,编辑目录内容。Emacs 甚至参与了一个人工智能的心理治疗过程。

在这里还有一点值得一提:那就是大部分 Linux 的应用程序都是 GNU 软件。这些 GNU 应用软件所支持的先进的功能在 UNIX 的 AT&T 或 BSD 标准版本上不一定能找到。例如,GNU 版本的 vi 编辑程序,elvis,包括了一个结构宏语言,这个宏语言与 AT&T 原版的实现不一样。但是 GNU 的应用软件仍然保持了与 BSD 或 AT&T 系统 V 相兼容的性质。许多人认为 GNU 的这些版本要比原版的软件更先进。

对许多用户来说最重要的应用软件应是 Shell 了。Shell 程序读入用户命令并加以执行,此外,许多 Shell 程序提供了诸如任务控制的机制。允许用户同时管理几个运行中的进程,输入输出重新定向,以及一个用于 Shell 任务描述的命令语言。一个任务描述文件是一个用命令语言写的程序,类似于 MS-DOS 下的批处理文件(batch file)。

在 Linux 上可运行多种 Shell。这些不同的 Shell 之间的最大差别就在于命令语言。例如,C Shell 使用的命令语言看上去很像 C 语言。而 Bourne Shell 使用的是另一种不用的命令语言 。对 Shell 的选择多是基于对命令语言的选择。而选用的 Shell 在某种程度上确

定了你的在 Linux 下的工作环境。

不论你熟悉和使用的是哪一种 Shell, 它的某种版本大概已经移植到 Linux 上了。最广泛采用的是 Bash (GNU Bourne Again Shell), 这是一个 Bourne Shell 的变体, 它包括了诸如任务控制、命令历史、命令与文件名的完整化操作。Emacs 式的用于编辑命令行的界面(接口)以及其他一些功能很强的标准 Bourne Shell 的命令语言的扩充等一些比较先进的功能。另一个被广为采用的 Shell 是 Tcsh, 它是一个带有许多扩充功能的 C Shell, 其扩充的部分类似于 bash 扩充部分。Linux 上还有其他诸如 zsh (一个类似于 Bourne Shell 的小型 shell), Korn shell (ksh), BSD 的 ash 和 Plan9 shell (rc) 等。为什么这些基本的应用程序会这么重要呢? Linux 为我们提供了独特的机会使我们能根据自己的需要去选择和裁剪使用的系统。例如, 如果你是唯一的使用你的系统的人而且你喜欢用 vi 编辑程序, 喜欢用 bash 作为 shell, 那么你就可以不在系统上安装其他的编辑程序和 shell 了。Linux 系统为喜欢自己动手设计环境的计算机人员提供了方便。

1.3.2 文本与文字处理程序

几乎所有的计算机用户都在某种程度上需要用到文档处理程序。在个人计算机用户中, 文字处理程序是一个用得很频繁的程序, 包括了编辑和整理文本, 图形, 表格及其它, 并且打印文本。

在 UNIX 用户中, 文本处理程序使用得较为普遍。文本处理与经典的文字处理有所不同。使用文本处理程序时, 首先由作者用“格式描述语言”描述文本的格式, 然后将带有格式描述的文本用任何一种编辑程序如 vi 或 Emacs 进行输入和修改, 而不是在特定的文字输入环境中输入。当文本输入完成后, 用户用文本处理程序对文本进行处理, 转化成适合打印的某种格式。这个过程类似于用一个如 C 这样的语言进行编程。先可用任何编辑程序输入按一定语法要求编写的文件, 然后把文件“编译”成可打印的形式。

Linux 可以支持多种文本处理程序。一为 groff。这是一个 GNU 版的格式文本处理程序 nroff。nroff 程序是由贝尔实验室首先开发且仍在世界范围内被许多 UNIX 用户所使用。另一个比较现代化的文本处理程序为 TEX 。是由计算机科学的泰斗 Donald Knuth 先生开发的。TEX 的一种“方言”LATEX 也可以在 Linux 上使用。TEX 和 groff 的不同在于它们的格式语言的语法不同。至于选用哪一种语言则取决于个人的爱好与兴趣, 以及相应的处理程序是否提供一些所需的其他辅助功能。例如, 许多人认为 groff 格式语言比较别扭, 而 TEX 的语法要接近于人类自然语言而易读, 所以选用 TEX。但是 groff 可以生成全 ASCII 码输出, 可以在普通显示器上读, 而 TEX 的输出则是为了在打印设备上输出, 直接在普通显示器上读就较困难。当然, 也有许多程序可以帮助人们进行从 TEX 到 groff 的转换或是把 TEX 的格式文件输出成全 ASCII 码的文件。

另一文本处理系统为 Texinfo, 它是 TEX 的扩充。免费软件基金会专门用它来编写软件程序说明书。Texinfo 可以从一个源文件产生一个打印的程序说明书或可在显示器上直读的超文本(hypertext)“Info”说明书。这种“Info”文件是 GNU 软件如 Emacs 的说明书的主要格式。

一个格式语言看上去是个什么样子呢?一般来说, 一个用格式语言写的源文件的组成

由文本为主,外加附有说明字形、字体,边框尺寸,列表等特殊功能要求的"控制码"。

作为一个例子,请看以下文本:

Mr. Torvalds:

We are very upset with your current plans to implement post-hypnotic suggestion in the Linux terminal drive code. We feel this way for three reasons:

1. planting subliminal messages in the terminal driver is not only immoral, it is a wast of time;

2. It has been proven that "post-hypnotic suggestions" are ineffective when used upon unsuspecting UNIX hackers;

3. We have already implemented high-voltage electric shorks, as a security measure, in the code for login.

We hope you will reconsider.

若用 LATEX 格式语言来写,看上去就是:

```
\begin{quote}
Mr. Torvalds:
We are very upset with your current plans to implement {\post-hypnotic suggestion
\}
in the {\bf Linux } terminal drive code. We feel this way for three reasons:
\begin{enumerate}
\item Planting subliminal messages in the terminal driver is not only immoral, it is
a wast of time;
\item It has been proven that "post-hypnotic suggestions" are ineffective when used
upon unsuspecting UNIX hackers;
\ item We have already implemented high-voltage electric shorks, as a security
measure, in the code for login.
\end{enumerate}
We hope you will reconsider.
\end{quote}
```

作者以任何编辑方式输入以上用 LATEX 格式语言写的源文本文件,然后运行 LATEX 程序产生一个格式化的输出。对于初次使用者来说,格式语言看上去似乎挺别扭,但实际上很容易掌握。用这样的文本处理程序可以加强作者在写作时的排版标准的意识。例如,所有的计数列表在文章的各部分都将以同样的形式出现,除非作者修改了这个计数列表环境。其基本目的是使得作者可以更集中精力于写作而不必用过多的精力来考虑文章的版面。

另一方面,那种"所见即所得"的文字处理程序也是颇具魅力的。其主要原因是这些文字处理程序为作者进行编辑提供了非常有效的可视界面。但是,这些界面也有一些局限性。例如,许多文字处理程序也需要提供一个特殊的"格式语言"来产生复杂的表达式,如一些数学公式等。而这正是文本处理程序的功能,只不过被缩小到一个局部范围内罢了。

文本处理程序有一个好处,就在于可以让用户说明自己的确切要求。此外,文本处理程序允许用户用任何一种编辑器输入文本文件,且文本文件也较容易由一种格式转换成另一种格式。当然为这种灵活性付出的代价就是"所见即所得"这样的图形界面。

许多文字处理程序的使用者习惯于在编辑时就预览文本的版面。而另一方面,使用文本处理程序写作的作者在写作和编辑的时候是不太用眼睛去看排版后的文本会是如何,而是逐渐习惯于从格式命令上去想象出文本排版以后的样子。

有一些程序可以帮助用户在打印之前预览一下排版后的文本的样品。例如,xdvi 程序可以显示在 X 窗口下 TEX 产生的与设备无关的文件。其他应用软件如 xfig 则为画图或流程图等提供了一个"所见即所得"的图形界面。该界面把图形转化成文本处理程序语言供插入文本中去。

应当指出的是,尽管文本处理程序如 groff 早在文字处理程序之前就已经存在,但许多人仍然倾向于使用这些文本处理程序。这是由于使用文本处理程序受图形环境的约束较小且功能较多。不管怎样,Linux 上可支持 idoc 文字处理程序。且在不远的将来我们将会看到一些商业化的文字处理程序在 Linux 上运行。当然如果你不愿意放弃使用文字处理程序进行文本处理的话,你除了使用 Linux 之外总还可以使用 MS-DOS 或其他操作系统。

有许多与文本处理程序有关的应用程序。Linux 的 TEX 版本就包括了功能极强的 METAFONT 系统。该系统可用于字形设计。其他如 ispell,一个交互式拼写检查和纠正程序: makeindex,一个用于为 LATEX 文件产生目录的工具: 以及 groff 和 TEX 的用于生成各种格式的文件和数学文本的宏程序包。还有一些用于在 groff、TEX 和大量的其他格式之间进行转换的翻译程序。

1.3.3 程序设计语言和辅助软件

Linux 提供了一个完整的 UNIX 编程环境,包括在 UNIX 上可以找到的所有的标准程序库、编程工具、编译器及调试器。在 UNIX 软件开发世界里,应用程序及系统程序的设计通常用 C 或 C+ + 语言。在 Linux 上的标准 C 或 C+ + 编译器是 GNU 的 gcc。这是一个先进的、具有许多选项的编译器。它也能编译包括 AT&T3.0 版在内的 C+ + 语言及一种面向目标的 C 语言。

除了 C 和 C+ + 语言之外,许多其它的语言编译器和解释器已经都移植到 Linux 上了。诸如 Smalltalk, FORTRAN, Pascal, LISP, Scheme 和 Ada。此外,还支持各种用于编写 80386 保护模式的汇编语言。还有一些诸如 perl, tcl/ tk 这样的 UNIX 上的编程工具和语言在 Linux 上也同样可以使用。

先进的调试器 gdb 也已经移植到 Linux 上。这个工具帮助用户一步一步地运行自己的程序以寻找到出错的地方。gprof 是一个用于监视程序运行的工具,可以统计用户程序的运行性能,使用户知道自己的程序的各部分运行的时间; Emacs 文本编辑器为各种程序设计语言提供了交互式编辑及编译的环境; 一些用于编译和组织管理一些较大的应用程序的工具诸如 GNU make 和 imake; 以及用于源程序的封锁及版本修改控制的程序 rcs,等等,也都能在 Linux 上运行。

Linux 实现了共享程序库的动态链接机制,简称 DLL。由于这种机制是将子程序代码在运行的时候才动态地进行链接的,就使得程序的二进制代码的长度大为减少。这些 DLL 也允许应用程序用自己编写的程序代码取代系统内部原有的函数定义。例如,如果应用程序希望用自己的定义的 malloc()函数而不用系统所提供的该函数,链接程序将用程序员自己编的 malloc 而不用系统提供的。

Linux 还是一个开发 UNIX 应用程序的理想的系统。它提供了各种先进的环境及工具。它支持各种标准如 POSIX. 1,使得在 Linux 上编写的软件能容易地移植到其他 UNIX 系统上去。专业的 UNIX 程序员及系统管理员可以在家里的个人计算机上的 Linux 上开发软件,然后搬到 UNIX 系统上。这不仅可以省下大量的时间与金钱,而且如果允许的话,你可以就在家里工作。计算机系的学生们可以用 Linux 来学习 UNIX 程序设计并且去发现和利用系统的各个不同的方面如核心程序的结构。有了 Linux,用户不仅可以使用全部的程序库及编程环境,更有意义的是用户还能看到全部的核心程序及库程序的源代码。

1.3.4 X 窗口系统

X 窗口系统是 UNIX 的机器上的标准的图形界面。它是一个有效的支持多种应用的环境。使用 X 窗口,用户可以同时在一个屏幕上开多个窗口,每一个窗口可以有一个用户登录。X 界面通常带有一个诸如鼠标器这样的指示装置。

已经有许多 X 窗口专用程序,如游戏、图形应用软件、程序设计环境及文字说明工具等。有了 Linux 及 X 窗口,你的机器就像一台工作站了。再加上 TCP/IP 网络软件,你甚至可以在你自己的机器上显示一些在其他机器上运行的 X 图形应用程序,就像使用其他支持 X 窗口的系统一样。

X 窗口系统首先是 MIT 开发的,并是免费可取和使用的。但是,许多商业软件发行人已经在发行一些对原有 X 窗口系统的增强版软件了。Linux 上现在使用的是 XFree86 版,这是一个免费的专为以 80386 为基础的 UNIX 系统如 Linux 这样的系统移植的版本。XFree86 支持多种显示器硬件,包括 VGA,Super VGA,及一些 带加速器的显示器适配器。它是一套完整的 X 窗口软件,包括 X 服务器及其他多种应用程序,程序库及文档资料。

标准的 X 应用程序包括了用于 X 窗口的以文本方式为主的应用程序的终端仿真器 xterm,管理用户登录的 X 管理程序 xdm,时钟显示器 xclock,用于 X 窗口的联机用户手册 xman,等等许多。不胜枚举。XFree86 基本版带有最初 MIT 发行版的那些标准应用程序。许多其他应用程序可以在不同的地方获取。理论上说,任一为 X 窗口编写的应用程序都能在 Linux 下进行编译和运行。

X 窗口图形界面的外观和使用在很大程度上是由窗口管理程序控制的。这个程序掌握着窗口的位置,提供一个具有用于改变窗口大小,设窗口标志(iconifying),窗口的移动和窗口框架外形等功能的用户接口。标准的 XFree86 版本包括 twm,一个颇具代表性的经典的 MIT 窗口管理程序以及其他一些更先进的窗口管理程序如 olvwm (Open Look Virtual Window Mamager) 。被 Linux 用户普遍使用的窗口管理程序是 fvwm,这是一个

小型的窗口管理程序。它所需要的内存大约只是 twm 所需的内存的一半,并为窗口提供了一个三维的外型和一个虚拟的工作台——如果用户把鼠标移到屏幕的边缘,则整个工作台随之移动,就好像这个工作台的台面比实际的屏幕要大一样。它可以按用户的需要作各种"剪裁",并且可以使所有的鼠标器的功能都可以由键盘来完成。

XFree86 版本给希望开发 X 应用程序的人员提供了编程程序库和 include 文件。它支持各种 widget sets ,如 Athena, Open Look 及 Xaw3D 等。它还提供了所有的标准的字型, bitmaps 程序员手册及资料文档。它还支持一个三维图形编程接口 PEX。

许多 X 应用程序员使用有专利的 Motif 的 widget set 来进行程序开发。有一些软件零售商出售一些在 Linux 上运行的 Motif 二进制代码的单用户或多用户使用权。因为 Motif 本身比较昂贵,没有多少 Linux 用户拥有它。但是静态地链连 Motif 的程序的二进制代码是可以免费发行的。因此,如果你用 Motif 编写了一个应用程序而且打算免费供他人使用的话,你可以提供该程序的二进制代码。那么,没有 Motif 的用户也可以使用这个应用程序了。

安装和使用 X 窗口系统的唯一的条件就是对硬件和内存的要求。一台 4M 内存的 386 就能运行 X 窗口系统。但是如果能有 8M 或 8M 以上的内存,使用起来就能更舒适一些。处理器速度能更高一些自然更好,但是有足够的内存则更重要一些。此外,要得到更好的显示效果,推荐使用一个加速显示卡(如 Local bus S3-chipset 卡)。如有足够的硬件支持的话,用户会发现在 Linux 上运行 X 窗口程序的速度可以达到或超过在其他 UNIX 工作站上运行 X 程序的速度。

在第 2 章中我们将介绍如何在系统上安装 X 系统, X 系统的使用将在第 3 章中介绍。

1.3.5 网络设置

有兴趣与外面的世界进行通信和交流吗? Linux 支持了两种基本的 UNIX 上的网络协议: TCP/IP 和 UUCP。TCP/IP 是 Internet 上使用的网络协议。有了 Linux, TCP/IP 以及与网络的连接,就能与世界各地的 Internet 上的用户通过电子邮件, USENET 新闻组, FTP 等手段进行通信交流了。目前 Internet 网上就有许多的结点机使用 Linux 系统。

大部分的 TCP/IP 网络使用以太网(Ethernet)作为网络的物理连接方式。Linux 支持多种以太网卡及与个人电脑的接口,包括手提式电脑上的袖珍以太网适配器。

但是,并非每个人家中都有以太网端口, Linux 也支持 SLIP 串行线网间协议(serial line internet protocal),它允许用户经调制解调器连接到 Internet 上去。要使用 SLIP, 用户必须能进入 SLIP 服务器。所谓 SLIP 服务器就是一台用户可经电话拨号方式进入的与网络联接的机器。许多公司或学校都有这样的服务器。事实上,如果你的 Linux 系统有 Ethernet 联接及调制解调器,你可以把它装配成一个为其他机器服务的 SLIP 服务器。

Linux 支持的网络功能有:网络文件系统 (Network File System, 缩写为 NFS),它允许用户自己的系统与网络上其他机器共享文件;文件传送协议(File Transfer Protocal, 缩写为 FTP),它允许用户在机器间传送文件;用 SMTP 协议进行电子邮件的传送和接收的 sendmail; NNTP 为基础的电子新闻系统如 C-News 和 INN; finger 命令,允许用户获

取在其他 Internet 网上用户的一些简单信息; 还有其他诸如 telnet, rlogin 及 rsh 等命令, 这些命令能使用户在网络其他机器上登录和执行命令。理论上说, Linux 可以支持各种 TCP/IP 为基础的应用程序和协议。

Linux 上有各种邮件和新闻阅览程序, 诸如 elm, pine, rn, nn 及 tin。不论你喜欢用哪一种, 都可以把你的系统设置成能够读取和发送世界各地的电子邮件和新闻。

如果你已经有了在其他 UNIX 系统上使用 TCP/IP 应用程序的经验, 那么可以说你已经知道如何在 Linux 上使用这些应用程序。Linux 提供的是标准的 socket 编程界面, 因此, 可以说任何使用 TCP/IP 协议的应用程序都可以移植到 Linux 上。Linux 的 X 服务器也支持 TCP/IP 协议, 允许用户在自己的机器上显示在其他机器上运行的应用程序。

UUCP (UNIX to UNIX Copy) 是一个比较老的用于在 UNIX 机器之间进行文件、邮件、新闻传送的机制。原先, UUCP 是在用调制解调器和电话线相连接的机器之间进行工作的, 但是后来 UUCP 也可用于 TCP/IP 网络上进行传送。如果你没有 TCP/IP 网络或 SLIP 服务器的网络连接, 你可以在系统上设置 UUCP 来进行文件和电子邮件的发送和接收。

本书的第 2 部分将详细介绍 Linux 上的网络安装, 配置和应用。

1.3.6 电子通信及 BBS 软件

如果你有一个调制解调器, 你可以用一个 Linux 提供的电子通信程序包与其他机器进行通信。在美国有许多人通过电子通信软件与 BBS 及其他一些商业化网络服务(如 Prodigg, CompuServe 及 American-On-Line) 相连。有些个人用户用调制解调器连到工作单位或学校。你甚至可以用调制解调器和 Linux 系统来接收和发送传真。Linux 下的电子通信软件非常类似于 MS-DOS 或其他操作系统下的通信软件。任何使用过通信软件包的人都会发现使用 Linux 的通信软件与使用其他系统上的通信软件几乎完全相同。

Linux 上使用最广泛的通信包是 Seyon。这是一个界面可由用户按照自己的要求进行设置和修改的 X 应用程序。它支持多种文件传送协议如 Kermit, Zmodem 等。其他 Linux 上的通信程序还有 C-Kermit, pcomm 和 minicom 等。这些通信程序很类似于在其它操作系统上相应的通信软件, 它们的使用也很方便。

如果你没有与 SLIP 服务器的连接, 你可以用 term 命令来拓宽你所使用的串行线路。term 允许你经调制解调器在远程机器上拥有多份登录。term 也可以让用户把远程机器上的 X 客户(client)的连接重新导向到经串联线联接的本机的 X 服务器上, 这样你就能在本机的 Linux 系统上显示远程机上的 X 应用程序。

运行 BBS(广告牌) 对许多人来说已经成为一种爱好。Linux 支持各种 BBS 软件, 其中许多软件的功能比其他操作系统上的相应的软件所提供的功能要强。只要有一条电话线, 一个调制解调器和 Linux, 你就能把你的系统变成一个 BBS。你的系统就通过拨号方式与全世界的 BBS 读者相联。Linux 的 BBS 软件包括了 XBBS 及 UniBoard BBS 软件包。

大部份的 BBS 软件把用户限制于菜单为基础的系统中, 这样的系统只为用户提供一些特定的功能和应用。一个不同于 BBS 的让用户进入系统的方式是让用户进入一个完全的 UNIX 环境。这种方式允许用户以拨号方式进入系统, 并且可以登录成为系统的一般

用户。虽然这一功能要求系统管理员付出一定量的用于管理的工作,但是在Linux 系统上为公共用户提供UNIX 环境并不困难。与TCP/IP 网络软件的配合,可以为你的系统上的用户提供电子邮件和新闻的服务。

如果你没有TCP/IP 网络的连接或UUCP 服务,Linux 仍然允许你与一些BBS 网进行通信。如使用FidoNet,可以通过电话线交换电子新闻和邮件。更多的关于电子通信和BBS 软件等网络软件的介绍详见第2 部分。

1.3.7 与MS-DOS 的界面

已经有许多实用软件提供了与MS-DOS 的界面。最为用户所熟悉的是Linux 上的MS-DOS 仿真器。这一仿真器允许用户从Linux 上直接运行MS-DOS 上的应用程序。尽管Linux 与MS-DOS 是两个截然不同的操作系统,80386 的保护模式环境允许一些任务的运行就像MS-DOS 应用程序在80386-emulation 模式下运行一样。

尽管一些广泛被使用的MS-DOS 应用程序已经可以在MS-DOS 仿真器上运行了,但是这个仿真器还处于发展阶段。因此,如果一个MS-DOS 的应用程序用了一些系统的很特殊的功能的话就可能无法在仿真器上运行,因为毕竟只是一个仿真器。例如,你不能期望能运行使用80386 保护模式特性的所有程序,如微软窗口的386 增强型。

在Linux 的MS-DOS 仿真器上成功运行的应用程序有:4DOS 命令解释程序,Foxpro 2.0,Harvard Graphics,MathCad,Stacker 3.1,Turbo Assembler,Trubo C/C++,Turbo Pascal,微软窗口3.1(realmode)及WordPerfact 5.1。标准MS-DOS 命令及应用程序如pkzip 等也能在仿真器上工作。

MS-DOS 仿真器的主要目的只是为那些主要运行Linux,但有时也需要运行几个MS-DOS 应用程序的用户提供一些方便,而不是要提供一个完备的MS-DOS。当然,如果这样的仿真器不能满足你的要求,你可以在一个机器上同时安装MS-DOS 和Linux 两个操作系统。使用一个称为LILO 的启动程序,你可以在启动的时候说明要使用哪一个操作系统。Linux 也可以与其他操作系统诸如OS/2 等并存。

Linux 提供了一个在Linux 和MS-DOS 之间传送文件的完善的界面,你可以把MS-DOS 分区或软盘安装(mount)到Linux 上,然后直接存取MS-DOS 文件。当前正在开发的一项称为WINE 的工程是开发一个Linux X 窗口的微软窗口仿真器。一旦这个项目完成,用户就可以直接在Linux 下运行MS 窗口的应用程序。这与Sun Microsystem 上的WABI 窗口仿真器类似。虽然WINE 还在开发阶段,但已经可以看出其结构是很好的。

1.3.8 其他应用程序

Linux 操作系统有许多形形色色的应用程序。Linux 的基本中心是一个个人UNIX 操作系统,但是这一点正在逐步地改变着。Linux 上的商业及科学软件正在日益扩充,连一些商业软件厂商也加入了Linux 应用程序的贡献者队伍。

在Linux 上有几个关系数据库,包括Postgres,Imgres 和Mbase。这些数据库具有完备的功能,它们的专业化的客户/服务器数据库应用程序与其他在UNIX 平台上可找到的数据库相似。商品化的数据库rdb 也可以在Linux 上运行。

科学计算应用程序,包括有限元分析工具 FELT;用于画出函数图形和数据分析的应用程序 gunplot;与 MATLAB 类似的符号数学程序包 Octave,计算器 xspread,基于 X 窗口的 fractint fractal 生成器 xfractint,数理统计软件包 xlispstat,等等。其他应用程序如电路模拟程序 Spice;图象/数字信号处理及可视化系统 Khoros 等也可以运行。当然,还有许多这样的应用程序已经或正在移植到 Linux 上。不论你是哪一领域,把 UNIX 上的应用程序移植到 Linux 上是不难的。Linux 提供了一个完整的 UNIX 编程界面,足以作为任何科学计算方面的应用的基础。

像任何其他操作系统一样,Linux 也有它自己的一些游戏程序。包括经典的文本为基础的地牢游戏,如 Nethacker, Moria 和 MUDs;多用户地牢游戏可以由多个用户同时交互地进行,如 DikuMUD 和 TinyMUD;以及一系列 X 游戏,如 xtetris, netrek, xboard, 等等。

对于高保真音响爱好者来说,Linux 支持各种声卡和有关软件。如一个可以控制光盘驱动器使之成为普通的光盘机的软件 CDplayer; 一个可以用来为合成器或其他 MIDI 控制的乐器进行音乐作曲的 MIDI 序列发生器(sequencer)和编辑器,以及数字化声音编辑器等。

我们在本书里无法一一介绍 Linux 上的丰富的应用软件。附录 A 中介绍了一些有关 Linux 的其他信息来源,包括了一个已经移植到 Linux 上的许多软件的清单。尽管这个清单远非完全,但它包括了相当数量的软件。另外一条寻找应用程序的途径是通过 Internet 查看 Linux FTP 站点上的 INDEX 文件。简要地浏览一下,你将发现有不少软件你会想去试一试,玩一玩。

如果你仍然找不到你需要的应用程序,你总是可以考虑把在其他平台上运行的软件移植到 Linux 上去。大多数在 UNIX 平台上运行的免费的软件在 Linux 上编译不会遇到什么困难的。最坏的情况下,你总是还可以自己动手编制。

1.4 关于 Linux 的版权

Linux 的发行与使用遵循 GPL 版权声明。GPL 是 GNU 计划的产物。它制定了一些“免费软件”的发行和修改的规则条款。这里的“免费”一词来源于英文里的“free”,它在这里不仅仅是免费的意思,而且还有自由的含义。由于 GPL 经常被曲解,我们在这里作一些说明,以帮助读者理解 GPL 的目标以及它对 Linux 的影响。

初始,Linus Torvalds 在发布 Linux 时使用的是一个比 GPL 发行和使用有更严格限制的版权声明。它允许该软件自由地发行和修改,但是禁止在发行和使用上用钱进行交易。而 GPL 允许人们从买卖 GPL 版权的软件中获利,但不允许任何人限制其他人在发布和修改方面的任何权利。首先需要声明的是在 GPL 保护下的免费软件并非是 public domain 的。public domain 的软件是没有任何版权的,是属于公众的。而由 GPL 保护的软件,是有版权的。它表示版权属于其作者,受到标准国际版权法的保护。如果认为免费自由发行就等于没有版权,那就是错误的。

GPL 的软件也不是“shareware”。一般地说,“shareware”软件归作者所有,作者享有

版权,发行时要求用户支付一定的费用。而 GPL 版权的软件可以免费地发行和使用。

GPL 允许人们修改这些免费软件,并发行他们自己的版本,但是,任何由 GPL 软件派生出来的软件必须仍然由 GPL 保护。换句话说,任何公司或个人不得把 Linux 修改后在有限制的版权下注册和销售。任何从 Linux 演变或修改来的软件仍然应属 GPL 的范畴。

GPL 条款允许软件免费自由地发行和使用,但是它也允许团体或个人在发行时收取一定的费用,甚至赢利。但是在销售 GPL 软件时,发行人没有权利从购买者中解除其权利。即如果你买了这样的软件后,你有权自由、免费地发行或者也进行销售。

这乍看起来好像自相矛盾。为什么要卖软件赢利而另方面 GPL 则允许任何人自由免费获取呢? 其实这并不难理解。例如,某个公司决定把大量的免费软件装到 CD-ROM 上进行发行,这个公司需要收一定的费用来支付生产和发行 CD-ROM 的销售。如果该公司决定从中获利,是 GPL 允许的。

销售免费软件的组织机构必须遵循 GPL 规定的条款,首先,它不能约束了购买者的权利。这就是说如果你买了 GPL 软件的 CD-ROM,你可以自由地复制并发行,收不收费自行决定。第二,发行者必须明确告知用户这些软件是 GPL 条款保护的。第三,发行者必须提供完整的所有软件的源代码,以使任何人都可以在软件上进行修改。

允许公司发行和销售免费软件是一件很好的事。因为并非所有的人都有 Internet 的上网权来免费自由地获取如 Linux 这样的 GPL 软件。GPL 条款允许公司销售 GPL 软件,从而扩大了这些软件的发行范围。在免费软件世界,重要的不是金钱,自由免费软件的宗旨是开发和发行软件并允许任何人获取和使用。例如,许多机购以邮购方式销售 Linux 软盘、磁带、或 CD-ROM 从中获利。而 Linux 的开发人员可能永远不会看到这些利润。这一点在开发者销售者之间的共识是该软件在 GPL 上注册时起就达成的。在下一节中我们将谈到这一 GPL 的宗旨是如何影响 Linux 开发的。

1.5 Linux 的设计宗旨和方法

Linux 是一个很独特的操作系统。理解它的设计宗旨和设计方法对有效地使用它是很有帮助的。

在商业性的 UNIX 开发机构中,整个系统的设计是按着质量管理、源程序的修改控制、程序说明、错误问题报告和纠正等一系列的步骤进行的。开发人员不能随意增加功能或改变代码的关键部分。他们只有在对问题报告中提出的问题进行相应的修改时才能按规定修改程序,且必须把这些修改存入源程序控制系统,以便在必要时可以把程序重新恢复到修改之前的状态。每个开发人员只负责系统程序的其中一部分或几部分的工作。只有负责某一部分程序的这个人可以取出该部分的源程序然后进行修改。在开发机构内部,质量管理部门在操作系统的每一条新的逻辑路径上进行各种测试并报告出现的问题。而修改和解决这些问题是开发部门的责任。在一个新的版本发行之前还需要用一个复杂的统计分析方法来保证在新的版本的发行之前有一定百分比的问题已经解决,然后整个操作系统作为一个整体才能通过发行的要求。总的来说,商业性的 UNIX 开发者用于维护

和支持他们的代码所采用的过程是非常复杂的。公司必须有定量的方法证明它们的下一个版本可以发行，包括关于操作系统的性能的集取和统计。开发 UNIX 是一个很大的项目，大到需要雇用几百甚至上千的程序员、测试员、程序说明书编写员及管理人员等。当然各个开发公司都不完全一样，但大致上是如此。

而 Linux 系统的开发则完全不同。根本没有类似的有组织地开发的东西，它的开发过程没有使用源程序控制系统，没有结构问题报告，没有统计分析。Linux 现在是，而且很可能永远是一个计算机编程迷的操作系统。这里的“计算机编程迷”是指那些全身心地，狂热地投入编程的程序员，那些热衷于“玩”计算机，并在上面捣鼓各种有趣的事情的人。

Linux 基本上可以说是一群遍布于全世界的 Internet 上的自愿人员开发出来的操作系统。在 Internet 上，乃至 Internet 之外，任何人都有机会帮助 Linux 的开发、查错、移植新的软件、编写程序说明、或帮助新的用户。没有一个组织来监查该系统的开发，而主要靠经过各种邮件通信及 USENET 新闻组进行联系和配合工作的。在 Linux 的开发过程中已经形成了一些共识。例如，任何人如果想把他编写的代码放进正式的核心程序中去的话，需把代码寄给 Linus Torvalds。由他进行测试，并决定是否加入核心程序中去。只要该程序不会破坏整个系统的整体设计，他一般会很乐意把它加进去的。

整个系统的设计是开放式的和功能式的。尽管最近一些系统的新功能和重要改变的数量已经越来越少。但是一般的规律是这样的，每过几个月就会发布一个核心程序的新版本。只要系统没有什么严重的或显而易见的问题，它就会被认为是稳定的，于是就可以发行。这里没有严格的质量管理和控制。Linux 开发的宗旨并不在于发行一套完美的代码，而在于开发一个免费的，自由的，为开发者而存在和发展的 UNIX。

欲给系统增加新功能或新应用软件的人一般先推出 alpha 版。这是为那些勇敢的无顾虑且喜欢在原稿上挑毛病的人提供的试用版本。由于 Linux 的用户主要遍布于 Internet 上，alpha 版的软件一般是先置放到 Linux 的 FTP 站点上去，然后在 USENET 广告牌上刊出有关如何获取和测试的说明。有兴趣的人可以从网上获取，然后把试用的情况、建议、问题报告、使用中出现的问题等与原作者进行交流。

在 alpha 版的初试中发现的问题解决后，代码可升为 beta 版，这时一般可以认为它已经基本稳定。但它也许尚未完善，有些功能可能还没实现。再进一步发展后将进入最终版，这时可以认为该软件达到了功能完全和可使用阶段。对用于核心程序的软件来说，它一旦完成，作者就可以请求 Linus Torvalds 先生把该软件加入到核心程序中去，或作为核心程序的可选功能 。

要指出的是以上所说的只是一些在 Linux 开发中俗成的惯例而不是规定。完全由开发者自己决定是否发行、何时发行何种测试版本。有些人对于自己的程序非常地自信，根本不发行 alpha 或其他测试版本。

你也许会产生这样的疑问：像这样一群无组织的自愿人员进行无组织无系统地编程和调试一个完全的操作系统怎么会成功呢？事实是，它是一个最有效率的最有目的性的系统开发工程。整个 Linux 核心程序从草稿开始，没有一个程序是有专人控制的。大量的工作是由志愿者完成的。他们移植和编写了许多程序库，开发了文件系统，编了许多硬件的驱动程序。

对大部分用户来说,要从头开始安装一个完整的系统,即从核心开始然后逐步加上各种应用程序及其软件,是有一定的困难的。Linux 的发行考虑到了这一点。因此,有一些发行版包括了一些帮助安装和运行一个完整系统所需的程序。需要重申的是:Linux 的发行不是只用一个标准的发行版而是有好几种不同的发行版,各有千秋。在第 2 章将介绍各种 Linux 的发行版和 Linux 的安装。

由于上述的 Linux 的特点,我们觉得有必要对初学者和有经验的 UNIX 或 Linux 用户分别说几句。

给初学者:首先,安装和运行自己的 Linux 系统并不要求很多的预备知识。事实上,许多初学者都已成功地在他们自己的系统上安装了 Linux。自己动手安装系统是一个有意义的实验活动,当然也是一个繁琐的,困难的过程。幸运的话,你可以不需要任何帮助就能成功地安装 Linux。在 Linux 上运行程序的过程中你可以逐步掌握一些 Linux 的基本知识。这本书里提供了许多可以帮助用户起步的内容。第 3 章包括了主要的 Linux 的基本操作。第 4 章介绍了基本的 Linux 的系统管理。在你开始安装系统之前最好也能先阅读一遍这两章,这对你在安装过程中遇到问题时会很有帮助。另一方面,任何人都不可能在一夜之间从一个初学者变成一个系统管理员。你必须要有充分的思想准备,准备走一段崎岖的路。

给 UNIX 或 Linux 老用户:即使是有好几年的 UNIX 编程及系统管理经验的人也有可能在安装使用 Linux 时遇到困难。从 Linux 不是一个商品化的系统这一点来看,就不难理解了。因为 Linux 没有按照许多你可能熟悉的 UNIX 系统那样的标准来开发的。更准确地讲,系统的稳定性是一个重要的因素之一。另一个因素是功能问题。在一些情况下,有些新的代码在尚有毛病或功能尚未完全时就被放进了核心程序。大多数 Linux 的开发者是这样的考虑的:更重要的是让用户尽早地使用和测试一个程序而不是发布一个完美的程序,这也是 Linux 的宗旨。例如,WINE 程序在完成测试之前就先公布 alpha 版本。这样,整个 Linux 世界就有机会涉足这个程序,进行测试、试用、帮助开发。如果一些用户觉得 alpha 版本已经能满足他们的要求了,就可以开始使用了。而商业化的 UNIX 厂家几乎从未以这种方式发行过软件。另一方面,Linux 系统差不多几个月就发布一个新的核心程序。它支持的新的软件也在层出不穷。正是用于这种动态性,即使你是一个有十几年经验的 UNIX 系统管理员,且使用过各种 UNIX 版本,对 Linux 可能也要一段熟悉过程。我们希望 Linux 能满足或超出你的期望。但由于 Linux 的宗旨是免费自由软件及持续不断的开发和生长,在 Linux 的圈子里把发展放在更重于稳定的地位,这对于长期涉足于 UNIX 世界中的不少用户来说是一个不太容易接受的概念。我们相信,只要不过于苛求完美,Linux 确实是像其他 UNIX 的实现一样的一个完整的可用的操作系统。

1.6 Linux 与其他操作系统之不同

首先必须指出的是 Linux 可以与其他系统“和平同处”于同一机器上。在一台机器上同时安装 MS-DOS 和 Linux 是没有问题的。甚至还可以在两个不同的操作系统之间交互运行。但是了解 Linux 与其他操作系统如 MS-DOS, OS/2 和其他个人电脑上实现的

UNIX等操作系统之不同仍然是重要的。

1.6.1 为什么使用 Linux?

为什么要使用 Linux 而不选用其他的操作系统呢?我们自然可以列出许多种理由来。如果你是 UNIX 的开发人员或使用 UNIX 的程序员,为何不考虑在自己家里的机器上也设置一个 UNIX 环境呢? Linux 能使你在自己的 PC 上调试开发 UNIX 软件,包括数据库和 X 窗口应用程序。如果你是个学生,使所在的学校的计算机有不少是 UNIX 环境。用 Linux 你就能用你自己的 UNIX 系统并按照你自己的需要进行配置。此外,如果你没有其他 UNIX 机器供使用的话,安装与运行 Linux 也是学习 UNIX 的操作的绝好方式。另外还有一点不能忽视,那就是 Linux 不仅仅是为了个人计算机上的 UNIX 用户。它的完整性及强壮性足以胜任一些大型任务或分布式计算的需要。在国外的许多公司,特别是小公司正在从以 UNIX 为基础的工作站环境往 Linux 上转移,以 Linux 环境替代原有的 UNIX工作站环境。许多大学也正在发现 Linux 作为操作系统教学来说是再合适不过了。一些大的商业软件公司也开始看到了自由免费软件系统可以给他们提供的机会。在下面的内容中,将说明 Linux 与其他操作系统之间的最重要的区别。我们希望 Linux 可以满足你的计算工作的需要或者至少能改善你现有的计算环境。要记住: 要想知道 Linux 的滋味,最好的办法就是亲口尝一尝,哪怕只是安装部分系统。

1.6.2 Linux 与 MS-DOS 之比较

常常可以看见在同一机器上运行 Linux 和 MS-DOS 情形。尽管 Linux 也提供了类似的应用程序如 TEX 等,许多 Linux 用户仍习惯于使用 MS-DOS 上的一些应用程序如文字处理程序。总有各式各样的原因使得许多人愿意使用两个系统上的各种不同的应用程序。如果你的整篇论文用的是 MS-DOS 上的 WordPerfect,要转换成 TEX 或其他别的格式也许不容易。当然,还有许多 MS-DOS 上的应用程序在 Linux 上没有,所以同时使用两种操作系统是不难理解的。

也许你已经知道,MS-DOS 并没有充分地利用 80386 和 80486 处理器的全部功能。而另一方面,Linux 又是在处理器的保护模式下运行的并且利用了处理器的全部功能。你可以直接访问所有的内存,甚至虚拟存储器。Linux 还提供了 MS-DOS 所不具备的完整的 UNIX 界面。所以在 Linux 下开发和移植 UNIX 的应用程序是件很容易的事。而在 MS-DOS 下,你会被限制在一个 UNIX 程序设计功能的一个小子集上。因为 Linux 是一个 UNIX系统的实现,你就不会有这样的局限性。

我们可以用几大张纸的篇幅对 MS-DOS 和 Linux 的不同进行讨论,也可以只用“它们是两个不同的操作系统”一句话简单概括。MS-DOS 价格比起其他商品化的操作系统来说不算昂贵,而且在 PC 世界中站住了脚。没有哪个操作系统在 PC 机上占有这么高的比例。其主要原因是因为价格合适。而 Linux 是免费的,在这方面哪个操作系统更占优势是一目了然的了。

你可以根据自己的需要和期望来评判 Linux 和 MS-DOS 的长短。Linux 并不是“老少咸宜”式的。如果你希望在家里运行一个完整的 UNIX 系统,并且不打算高价去购买

UNIX的其他实现,Linux 就该会是你的必然选择了。有一些工具可以让你在 Linux 和 MS-DOS 之间交互式运行。例如,从 Linux 上很容易读取 MS-DOS 的文件。还有一个称为 MS-DOS 的仿真器的软件,一个微软窗口仿真器也正在开发之中。

1.6.3 Linux 与其他操作系统的比较

有一些更先进的操作系统,例如 IBM 的 OS/2 和微软公司的 windows NT 正在被越来越多的人使用。OS/2 和 windows NT 都是像 Linux 一样的多任务的操作系统。从技术上讲这三者有些类似,他们粗略地说都支持相同的用户界面、网络功能、安全性等。但是,Linux 与其他操作系统的真正的不同在于 Linux 是 UNIX 的一个实现,所以它从 UNIX 用户那里受益匪浅。

是什么使得 UNIX 如此有影响呢? 一方面是由于它是最广泛使用的多用户操作系统,另一方面还因为它是大部分自由免费软件的基础。如果你有幸能上 Internet 你就会发现几乎所有的从网上可获取的免费软件都是为 UNIX 系统编写的。连 Internet 自身也主要是以 UNIX 为基础的。

UNIX 有着许多来自不同商家的不同的实现,但是没有一个专门的机构负责发行。UNIX 的界面总在不断地发展和变化。尽管许多人要求对系统进行标准化,但是都没有一个机构来组织这一设计。因此,任何软件商或任何有兴趣的人都可以按任一标准来实现任一版本。Linux 主要采用 POSIX.1 标准。随着时间的迁移,它可能会选择其他的标准。而在另一方面,其他许多操作系统,如 OS/2 和 Windows NT 是独家的产品,它的界面及设计都由独家控制,只有它所属的那家公司实现这些设计。从某种意义上讲,这种组织方式是有好处的。它为用户界面,编程建立了严格的标准。不论你走到哪里,OS/2 就是一种 OS/2,Windows NT 就是一种 Windows NT。总之,尽管标准化并不是 Linux 开发的主要话题,但是却是 Linux 与其他操作系统的一个明显的不同之处。

1.6.4 Linux 与 UNIX 的其他实现的比较

市面上也有一些在 80386 和 80486 上的 UNIX 的实现。80386 体系结构的设计是为 UNIX 的实现考虑的,这一点已经被一些软件开发机构利用到 UNIX 的实现上。从功能上看,其他微机上的 UNIX 的实现与 Linux 很类似。你可以看到大多数的商业 UNIX 版本支持的软件,编程环境和网络功能几乎与 Linux 相同。但是在其他一些地方,Linux 与这些商业化 UNIX 还是有一些不同之处。

首先,Linux 支持的硬件与商品的 UNIX 版本的支持的范围略有不同。一般来说,Linux 支持的硬件设备都是为大多数计算机用户所知的硬件设备,而商业 UNIX 商家则支持的硬件设备的面一般要更广一些。这是由于 Linux 开发者所能接触到的硬件的局限性。

其次,商品 UNIX 版本通常附带了一套完整的说明书和用户手册。而大部分 Linux 的程序说明却比较有限。在 1.8 节中,我们将介绍一些有关 Linux 程序说明及一些其他信息的来源。

至于稳定性和强壮性,不少用户根据他们的经验认为 Linux 与其他商品 UNIX 系统

基本相当。Linux 还在发展阶段,有一些功能,如 TCP/IP 网络协议等的稳定性将会随着时间的推移进一步得到完善。一些实际应用的例子已经证明 Linux 可以像其他 UNIX 的实现版本一样实用和可靠。例如,Linux 被用于一只海洋学考察船上的电信管理和数据分析的工作。一些国家的南极考察站也使用了 Linux。在美国,一些医院已经或正在准备使用 Linux 来管理病历。

对于许多用户来说,价格是选择操作系统的一个很重要的因素。如果你可以连到 Internet上并下载所需的文件的话,那么 Linux 软件对你来说就是免费的。如果你没有办法自己上 Internet 去取,那么你可以经过邮购的手段取得以磁盘、磁带或光盘形式的整套软件。当然,你可以从朋友处拷贝一份或是同几个人一道共同分担邮购的手续费用。另外,如果你要在许多机器上安装 Linux,只需获取一套发行版,因为 Linux 软件的发行没有诸如"单机版权"这样的限制。

也有其他一些免费或廉价的在 386 或 486 上实现的 UNIX 系统。最为人们熟悉的大概要数 386 BSD 了。它是把 BSD UNIX 移植到 386 上的版本。在许多方面 386 BSD 与 Linux 差不多。至于哪一个更如人意,那就看个人的爱好和需要了。它们的唯一一个显著的不同之处在于 Linux 的开发是开放式的,任何人可以自愿加入开发过程,而 386 BSD 是有组织地、封闭式地进行开发和维护的。正因为如此,在设计思想和方法上这两个系统存在着极大的不同。这两个系统的目标也不完全一样:Linux 的目标是从零开始设计一个完整的 UNIX,而 386 BSD 的目标只是把已有的 BSD UNIX 编码移植到 386 上。NetBSD 是另外一个 BSD NET/2 版本在许多机器上的移植,包括 386。NetBSD 的开发略为开放一些。

商品 UNIX 的实现版本的价值不应该受到贬低。它的价格不仅包括了软件,还包括了说明书,系统维护及质量的保证。尽管这些对于一般个人计算机用户来说不显得那么重要和必需,但是对于一些机构来说有时是很重要的。

1.7 安装 Linux 的硬件要求

至此,你可能已经接受了这样的看法:即 Linux 是一个很好的系统,它可以为你作各种事情。在你兴冲冲地准备安装它之前,你还需要知道了解一下它对硬件的要求及它所支持的硬件设备的一些局限性。

首先要重申的是:由于 Linux 是由它的用户开发的。这就意味着 Linux 所支持的硬件局限于那些由这些参与开发的用户所使用的或能接触到的设备 。虽然大部分的硬件和 80386/ 80486 系统的外部设备都能支持,但是一些更专业化的或鲜为人知的设备则尚未得到支持。随着时间的推移,更多的硬件将得到支持。

另一个硬件支持方面的局限性来自于许多公司把硬件的接口私有化。因为这么一来,Linux 开发的自愿人员就不可能为这样的设备写驱动程序了。因为如果他编写了这样的驱动程序,那么该驱动程序将属于拥有这个接口的公司。这就违反了 GPL 版权的条款。这些公司为它们的硬件接口编制在 MS-DOS 或 MS Windows 上的驱动器,而使用这些硬件的用户则不需要知道该接口的结构。这就造成了 Linux 没有这类设备驱动器的原因之一。

对于这种情况,Linux 开发人员几乎是毫无办法。在一些情况下,程序设计员试图凭据对这些接口的猜想来尝试着写驱动程序。在另一些情况下,开发人员以提问的方式与设备厂家合作以图得到一些有关信息。成功的程度则各不相同。

在下面的内容中我们将介绍一下 Linux 上对硬件的要求和支持。这里要说明的是许多硬件驱动程序尚在开发中,以下所列的主要包括了已经被使用了一段时间并已知是稳定的硬件的驱动程序。如果有什么疑问可以查阅你所使用的 Linux 发行版随带的的说明。

1.7.1 对主板和 CPU 的要求

Linux 目前支持的 CPU 有 Intel 80386、80486 及奔腾系列,包括了 386SX、486SX、486DX、486DX2 等。其它非 Intel 的机器如 AMD 和 Cyrix 处理器等也可以运行 Linux。如果你用的是一台 80386 或 80486SX,你最好要有一个数学辅助处理器。当然这并不是非有不可的,因为如果机器上没有数学辅助处理器的话,Linux 核心程序可以实现浮点运算器的仿真。所有标准的浮点运算器配置 Linux 都支持,如 IIT,Cyrix Fasmath 及 Intel 辅助处理器。

系统主板必须用 ISA 或 EISA 总线结构。由此决定系统与外部设备和主总线上的其他部件连接的界面。当今大部分市面上出售的系统使用的都是 ISA 或 EISA 总线。IBM PS/2 上使用的 Microchannel (MCA) 总线目前尚未支持。使用局部总线以加快显示器和磁盘的存取速度的系统 Linux 也支持。不过,建议你最好选用如 VESA 这样的标准局部总线的结构。

1.7.2 对内存的要求

与其他类似的操作系统比较,Linux 需要的内存非常小。只要有 2MB 以上的内存就可以运行。但是我们建议至少用 4MB 内存。内存越大,系统运行越快。Linux 能支持 386/486 的 32 位地址,换句话说,它能自动地利用你的机器上的所有的内存。

Linux 在 4MB 内存的配置下就能运行得不错,包括运行 X 窗口和 Emacs 等很多有用的应用程序。但是,配置更多一些内存几乎与选用一个更快的处理器一样重要。8MB 内存一般来说对个人计算机用户是足够了。如果你估计系统将会有比较重的多用户负荷,16MB 内存或更大一些也许是必要的。

大部分 Linux 用户都把硬盘上一部分分区指定为内存交换空间。这部分空间被用作虚拟内存。尽管交换空间不是物理内存的替代,它可以允许你的系统在运行一个很大的应用程序时把一些暂时没用的代码暂时从内存交换到交换空间去。至于交换空间应设置多大,取决于若干因素,我们将在 2.2 节中讨论。

1.7.3 对硬盘的要求

你并不一定非要有一个硬盘才能运行 Linux,完全可以用软盘来运行一个最小的系统配置。这不但会很慢,并且很受限制。况且大部分用户的机器上都已经配有硬盘。你必须有一个 AT-标准(16 位)的控制器,虽然 Linux 核心程序也支持 XF-标准的(8 位)控制器,但是大部分机器上现在使用的是 AT-标准。Linux 支持所有 MFM,RLL 和 IDE 控制

器。而只有那些实现ST506硬件仿真器的ESDI控制器Linux才支持。一般来说,对于非SCSI硬盘和软盘控制器来说,如果你能在MS-DOS或其他操作系统上使用,那么你也就能在Linux上使用。

尽管Linux对SCSI控制器的支持是有一定限度的,因为这类控制器接口的标准范围太宽,它还是提供了对一些比较普遍使用的SCSI硬盘控制器的支持。Linux支持的SCSI控制器有: Adaptex AHA1542B, AHA1542B, AHA1742A, AHA1522, AHA1740, Future Domain 1680, TMC-850, TMC-950, Seagate ST-02, UltraStor SCSI, Western Digital WD7000FASST等。

1.7.4 对硬盘空间的要求

当然,要安装Linux,你需要在硬盘上有一定的自由空间。Linux支持一台机器上多个硬盘。如果必要的话,你可以给Linux分配多个硬盘上的空间。需要的硬盘空间大小主要取决于需要和想在系统上安放的软件的数量。与UNIX的其他实现版本相比,Linux需要的空间是比较小的。可以在10～20MB的硬盘上安装运行一个完整的系统。但是如果你考虑以后还要扩充或安装像X窗口这样的大型软件包,那你则最好有更大一点的空间。如果你的系统打算支持多用户,那么你还得考虑为每个用户都提供存放文件的空间。此外,除非你有很大的物理内存(10MB以上),你还得留出一个交换空间作为虚拟内存。关于交换空间的安装及使用将在"2.2.3 Linux格式的分区"中详细讨论。

每个Linux的发行版本通常都随带一些说明,帮助你根据你所选择安装的软件来准确地计算所需要的空间的大小。你可以用不到20MB的空间运行一个最小的系统。而一个包括了各种软件的完整的系统需要80MB左右。一个包括了为多用户留的文件空间及未来扩充用的空间的大的系统则可能需要100～150MB。这些都是一个粗略的估计,你需要根据你自己的要求和目标具体确定存储空间的大小。

1.7.5 对显示器和Video适配器的要求

Linux支持所有标准的Hercules, CGA, VGA, IBM monochrome及super VGA卡和文本显示器。一般来说,如果一个显示卡和显示器在诸如MS-DOS等其他操作系统上可以配合工作,它也就能在Linux下工作。

诸如X窗口系统这样的图形环境有它自己对显示硬件的要求。我们将把这些要求放在第2章中介绍。

1.7.6 其他硬件

以上列出了运行Linux所需的一些基本硬件。下面介绍一些可选装置(如磁带、CD-ROM、声卡等)的情况。

1. 鼠标器和其他指示器

在大多数情况下,只有在类似于X窗口系统这样的图形环境下才需要用鼠标器或其他指示器。但是也有一些Linux应用程序虽然与图形环境无关但仍然也使用鼠标器。

Linux支持所有标准的串行鼠标器,包括logitech, MM系列,Mouseman, Microsoft

(两个键)及 Mouse System (三个键)。Linux 同时也支持 Microsoft, logitech 和 ATIXL 总线鼠标器(bus mice)和 PS/2 鼠标器。其他指示器,如 trackball, 作为上述鼠标器的仿真器也能在 Linux 下工作。

2. 光盘存储器

几乎所有的光盘存储器都使用 SCSI 接口。只要你有一个 Linux 支持的 SCSI 适配器,你的光盘存储器就能工作。在 Linux 上能使用的光盘驱动器有 NEC CDR-74、Sony CDU-541、Texel DM-3024、Sony CDU-31a 和 Mistsumi。Linux 也支持标准的光盘存储器的 ISO-9660 文件系统。

3. 磁带驱动器

有几种不同的磁带驱动器在市面上有售。这些驱动器大多使用 SCSI 界面并为 Linux 所支持。已经证实的 Linux 支持的磁带驱动器有: Sankyo CP150SE, Tandberg3600, Wangtek5525ES, 5150ES,带有 DC36 适配器的 5099EN 和 QIC-02 驱动器。支持其他磁带机的驱动程序正在开发之中。

4. 打印机

Linux 支持所有的并行打印机。如果你能在 MS-DOS 或其他操作系统下控制并行端口来驱动打印机的话,在 Linux 下边同样能控制。Linux 的打印软件是由 UNIX 标准的 lp 和 lpr 软件组成的,它允许用户通过网络在远程的打印机上打印。

5. 调制解调器

像打印机一样, Linux 支持所有的串行调制解调器,包括内部的和外部的调制解调器。在 Linux 上有大量的通信软件,包括 kermit, pcomm, minicom 和 seyon。如果你的调制解调器可以在其他操作系统下使用,那么它一定也能在 Linux 下使用。

1.7.7 以太网卡

Linux 支持许多种以太网卡和局域网适配器,包括:

3com 3c503, 3c503/16; Novell NE1000, NE2000; Western Digital WD8003, WD8013; Hewlett Packard HP27245, HP27247, HP27250; D-Link DE-600。

下列的网卡也可以在 Linux 上工作:

LANNET LEC-45, Alta Combo, Artisoft LANtastic AE-2, Asante Etherpak 2001/2003, D-Link Ethernet II, LTC E-NET/16 P/N 8300-200-002, Network Solutions HE-203, SVEC 4 Dimension Ethernet, 4-Dimension FD0490 EtherBorad 16。

与以上任何一种网卡兼容的网卡在 Linux 也应该能工作。

1.8 有关 Linux 的资料来源

除了本书之外还有许多关于 Linux 的资料。特别要指出的是,对那些没有用过 UNIX 的用户来说,有一些书虽然不是专门介绍 Linux 的,而是介绍 UNIX 的一般资料,也是很有帮助的。如果你对 UNIX 不熟悉,除本书外,我们还建议你花点时间读一些有关 UNIX 的书。

下面介绍其他一些 Linux 资料的来源,这些资料多为电子媒体形式,需要有网络访问的权限,诸如访问 Internet, USENET, 或其他网络的权限。如果你自己没有访问网络的权限,也许你可以通过你的有访问权限的朋友帮助你得到一份资料的拷贝。

1.8.1 从网络上获取资料

如果你有了 Internet 的访问权限,通过不记名 FTP 访问可以从世界各地的文件库中获取有关 Linux 的许多资料。如果你没有直接访问 Internet 的权限,你仍然有可能获取其中的一部分资料。许多以光盘方式发行的 Linux 版本包括了这里谈到的资料。此外,在其他网络如 Fidonet 和 Compuserve 上也可获取。如果你能发送电子邮件到 Internet 上,可以试用 ftpmail 服务器来获取所需资料。ftpmail 服务器会从 FTP 站点把文件以电子邮件方式把资料邮寄到你的电子信箱。有关 ftpmail 的使用在附录 B 中有进一步介绍。

许多 FTP 站点存有关于 Linux 的资料。在附录 B 中将把一些比较常用的站点地址列出以供读者参考。为了减轻网络上的通信负载,建议读者优先选用那些距离较近的站点地址。

附录 A 包括了一些可以从不记名 FTP 站点上获取的资料目录。其文件可能会因地而异。大多数的文件区把相关的资料存放在 Linux 目录区的 docs 子目录下。例如,在 sunsite.unc.edu 地址上,Linux 文件存放在 /pub/Linux 目录下,有关的资料可以在 /pub/Linux/docs/ 子目录下找到。大部分资料会定期地贴到一个或多个与 Linux 有关的 USENET 新闻组上,详见“1.8.4 USENET 新闻组”。

1.8.2 Linux 在 WWW 网上

Linux 的资料的 Homepage 可以从 WWW 网的 URL 地址:

http://sunsite.unc.edu/mdw/linux.html

上看到。这个 homepage 包括了许多有关如何使用 Linux 的资料。同时也提供了一些可以找到资料的其他地址的目录。

1.8.3 书及其他出版物

至此,关于 Linux 的正式出版物几乎没有什么。比较有价值的是由 Linux 资料工程推出的书,这项在 Internet 上进行的工程为 Linux 的使用编写和发行手册。这些手册从形式上看类似于商品化的 UNIX 资料,它们包括了从安装 Linux 到使用和运行这个系统,编程,网络和核心程序的开发,等等,内容非常丰富。这些手册可以通过 FTP 获取,或者通过邮购。附录 A 中作了详细的介绍。

关于 Linux 的书正式出版虽然没有多少,但是市面上可以看见不少的 UNIX 的书。而这些 UNIX 的书有许多内容同样适合于 Linux。特别是关于如何操作以及编程方面的资料。在这些方面 Linux 与 UNIX 没有什么差别。简单地说,几乎所有关于 Linux 的使用和编程的一般资料都可以从关于 UNIX 的书中查到。事实上,本书也可以看成是对众多的 UNIX 书籍的一个补充。我们在这里提供的是关于 Linux 的重要的、特殊的和详细的资料。读者可以从 UNIX 的书上找到有用的一般性资料。

有几本较好的介绍 UNIX 的书作为基础,加上本书在手,你完全可以开始在 Linux 上作各种事情了。附录 A 提供了一些值得推荐的 UNIX 书,适合于从初学者到精通UNIX 的人。

有一份关于 Linux 的月刊名叫《Linux Magazine》,在世界范围内发行。阅读这种杂志是保持对 Linux 用户世界有最新了解的一种有效途径,特别是对无法访问 USENET 新闻组的人来说更是如此。

1.8.4 USENET 新闻组

USENET 是一个世界范围内的电子新闻和电子论坛,简称新闻组。各个新闻组有各自的论题和中心内容。许多 Linux 的开发工作是在 Internet 和 USENET 的推动和帮助下完成的。在 USENET 上有一些新闻组是专门讨论与 Linux 有关的题目的。

最初的 Linux 新闻组是 alt.os.linux。它是为了把许多关于 Linux 的讨论从 comp.os.minix 新闻组中分离出去而建立的。很快,这个新闻组的负荷就大到足以在 comp 层上开辟新的新闻组的规模了。于是在 1992 年 2 月由读者投票表决,comp.os.linux 正式成立。

comp.os.linux 很快地成长为 USENET 上最热门的新闻组之一。比其他 comp.os 下的新闻组更热闹。到 1992 年 12 月,投票表决关于把该新闻组分成几个子新闻组。只有 comp.os.linux.announce 得到需要的票数。在 1993 年 7 月这个新闻组终于分出了一个新的层。几乎有 2 000 人参加了 comp.os.linux 重组的投票表决,成为 USENET 上规模最大的投票表决事件之一。

如果你无法直接访问 USENET,但却能从 Internet 上接收和发送电子邮件,可以使用“邮件到新闻”的转换的网关(Gateway)方式。

comp.os.linux.announce:该组是一个有主持人负责的新闻组,专门用于发布公告和一些关于 Linux 的比较重要的消息(如问题报告、软件的重要 patch 等)。如果你打算看 Linux 新闻组,这是一个首选的组。这个组上的一些重要新闻经常不在其他组中重现。这个组还包括了一些定期发布的有关 Linux 的资料的消息。

由于这个组是有主持人管理的,因此往上张贴公告必须经过主持人的批准。Matt Welsh 先生和 Lars Wirzenius 先生是这个组的二位主持人。你先把准备张贴的文章送主持人批阅。通常你也只需像往其他无主持人的新闻组上贴布告一样操作,新闻管理的软件会自动地把文章转送该组主持人待批。但是如果你使用的系统配置不合适的话,你也可能需要先把文章送到主持人处,其地址为:linux-announce@tc.cornell.edu。

以下所列的是无主持人的新闻组。

comp.os.linux.help:这是一个最热闹的新闻组,它是专为人们提出一些关于 Linux 使用、设置,或运行其他在 Linux 上的软件时碰到的问题和解决方法的地方。如果你在使用中遇到了问题,你可以试着往这个新闻组上去寻求答案。可能你会收到某个人的回信,帮你解决问题。但是,我们建议大家在往新闻组上寻求帮助时,能首先自己从手头的资料上询找答案。

comp.os.linux.admin:这个新闻组专门讨论 Linux 系统管理的问题,特别是多用户

环境下的系统管理。关于 Linux 的系统管理,如软件打包,作备份文件,用户管理等问题的讨论都可以在这里进行。

comp. os. linux. development:这是专门讨论 Linux 系统开发的地方。所有关于核心程序开发及其他系统开发的问题都可在这个新闻组中讨论。例如,你要写一个驱动程序,需要一些编程方面的帮助,这个新闻组是最合适的寻求帮助的地方。这个新闻组同时也是讨论 Linux 发展方向及目标的地方。

值得指出的是,该新闻组从技术上讲不是讨论 Linux 上的软件的地方,而是讨论 Linux 自身的开发,如核心程序、系统程序库等的开发问题。其他关于在 Linux 下开发应用程序的问题应该放到其他新闻组中去讨论。

comp. os. linux. misc: 该新闻组囊括了所有那些不属于任何其他 Linux 专门新闻组的问题的讨论。特别是一些技术性问题的讨论,比如"是用 Linux 还是用 Windows NT?"这样的问题。

值得一提的是,原有的新闻组 comp. os. linux 已经被分成几个分组了。如果你的系统原先可访问 comp. os. linux 而不是新的如所列的新闻组,应请系统管理员创立这些新的新闻组。

1.8.5 Internet 通信录

如果你有 Internet 上的电子邮件的使用权限,那么你可以通过参加一些通信录而得到 USENET 的一些新闻广告。即使你没有直接和 Internet 连接,只要你能访问任何一个可以与 Internet 进行邮件交换的网络,就可以参加这些通信录。

"Linux—— 活动家"通信录主要是为 Linux 开发者及有兴趣参与开发的人提供一个相互联系的方式。这是一个称为"多渠道"的通信录。你可以根据兴趣和需要参加一个或多个"渠道"。一些可选择的"渠道"有,关于一般性问题讨论的 NORMAL 频道; 关于核心程序讨论的 KERNEL 频道; 关于 gcc 编译器和程序库开发的 GCC 频道; 关于 TCP/IP 网络问题讨论的 NET 频道; 关于 Linux 资料的编写与开发的 DOC 频道,等等。

读者可以给 linux-activists@niksula. hut. fl 送电子邮件索取关于通信录的有关资料,包括可选频道的目录以及如何订阅和退定某个频道。

还有不少关于 Linux 的专用通信录。查找这些通信录的最好途径是注意浏览 USENET 新闻组上的公告,以及读一些公开的定期在 USENET 的 news. answers 新闻组上发布的通信录。

1.9 获取帮助

当你在 Linux 世界里探索和漫游的时候,毫无疑问地希望得某种程度的帮助。哪怕是很熟悉 UNIX 的人有时也仍然会被 Linux 的一些操作和功能所困扰而不知所措。因此,知道从哪里寻找帮助自然就是很重要的了。

最基本和直接的途径要数 Internet 上的通讯录和 USENET 上的新闻组了。如果你没有上这两个网络的权限,你也可以从其他类似的网络寻找关于 Linux 的讨论的地方,如

BBS 等。一些商业机构也提供一些Linux 的商业化支持。它们提供的服务需要付咨询费。

下面是几点建议。这些建议有助于你提高使用Linux 的能力,同时也能帮助你成功地找到问题的解决办法。

首先,查询所有能找到的文件和说明书。当你遇到问题时第一件事就是查阅所有参考书和资料。附录A 列出了部分参考资料。因为所有的资料都是为用户使用Linux 提供的。一些关于UNIX 的资料常常也是可以用来对付Linux 的问题的。本书当然是一个很好的资料。很可能你的问题在查阅本书之后就能得到解决。

如果你有上网的权限和条件去访问与Linux 有关的新闻组或有电子通讯录,请你在贴你的求助信或求助条之前,首先读一下版上已有的新闻和其他内容。很多问题其他人可能已经遇到了并且有人已经作了解答。你若没有先浏览一下就求助往往会把自己带进尴尬的境地。

第二,学会自己动手。在许多情况下,更好的办法是在求助之前自己动手。在许多情况下更好的办法是自己先独自地作一些研究和尝试。不管怎样,当你选用Linux 这个操作系统时你就应该有这种思想准备了。因为Linux 是由于对计算机的业余爱好和在上面的大量的自发的研究和尝试而形成的系统,它不是一套商业化的系统。事实上,通过一些尝试你能学会许多有关系统的知识进而能自己动手解决问题。也许有一天你会成为Linux 的专家呢。不要小看了自己摸索出来的经验和解决问题的能力。也不要以为在运行Linux 时不会碰到一些自己可以解决的小麻烦。

第三,保持冷静。遇到问题时要保持冷静,克服急躁。急是没有用的。随着Linux 的日益成熟和越来越多的人加入使用的行列,许多问题会逐步得到解决的。但是不论如何,即使是商业化的UNIX 系统有时也会出一些小毛病。当各种尝试失败时,先放一会儿,等你完全从急躁情绪中摆脱出来之后再重新开始,这时你的脑子往往会好使得多,思路会清晰得多。

第四,如果你最终决定到网络上贴求助告示,务必把告示内容写清楚明了。要知道在网络上能看到你的告示的人虽然很多,但是并不是人人都喜欢回答别人的问题,特别是没有说清楚的问题。要想到别人不是你的咨询顾问,没有义务回答你的问题。因此,务必要尽可能地写得礼貌,简明切题,内容文字清晰。

如何才能做到这样呢?首先,要尽可能多的提供有关你所使用的机器和你遇到的问题的详细情况。如果只是简单地问:“我怎样才能让电子邮件工作?”这样的问题,你可能得不到任何有效的帮助。你至少应该告知你的机器型号,使用的软件,以及你是如何运行,结果怎样,出的问题是什么样的,等等。当你提供技术性信息时,最好把有关的软件的版本以及硬件的简要情况包括进去。当然,也没有必要过分地详细了。例如,当你在说明你使用网络软件的问题时,关于机器的商标和牌子,显示器的型号就是无关的了。其次,记住自己动手努力一番之后再贴求助告示。这样你就能把问题说得更清楚。

另外,在网上读告示的人,不论他多乐于助人,也经常会对一遍又一遍地被问的相同的问题感到厌倦的。在许多时候,如果你首先浏览一下你打算贴告示的新闻组的话,可能会发现你的问题已经有人提出,或者已经有答案了。

最后,请读者记住,网络是一个寻求帮助的好地方,但是其资源是有限的,大家都有义务来更有效地利用它。

第 2 章　Linux 软件的获取与安装

本章将介绍如何从众多的 Linux 软件套装版本中选择并获取一套 Linux 软件，并将其安装在微机上。前面已经讲过，Linux 软件并没有正式的发行版本，它的每一种发行版本都是为不同的目标而组合成的。这些不同的发行版本也有不同的发行方式，如通过 Internet 上的 FTP 服务，通过遍布全世界的 BBS 系统，或使用磁盘、磁带及 CD-ROM 邮寄发行。

每个 Linux 发行版本有其自身的安装方法。本章将以 Linux 软件的 Slackware 发行版本为例，详细介绍 Linux 软件的安装方法。本章所介绍的基本安装过程适用于各发行版本。如果用户在安装过程中遇到了麻烦，可查阅附录 A 所列的资料或寻求其他形式的帮助。

2.1　Linux 的发行版本及获取方式

Linux 是一套免费软件，没有一个特定的组织或实体对它的发行负责，因此每个人都可以自由地组装并发行一套自己的 Linux 软件。这种发行方式使得当前存在许多 Linux 软件的发行版本，用户必须从中选择一套适合自己的发行版本。许多发行版本包含了运行一个完整系统所需的各种软件包，要占用大量的磁盘空间；而另一些较小的发行版本只包含一些常用的软件包，以节省用户的磁盘空间；还有一些发行版本只包含 Linux 系统内核，用户必须自行安装诸如 X-Window 系统之类的软件包。

那么怎样从诸多的 Linux 发行版本中选择一个呢？用户可以在 USENET 新闻组上得到已经安装过 Linux 系统人的帮助，但最好是身边有一个安装过 Linux 系统的朋友。每个人的需要和观点都不同，影响选择的因素也很多。但实际上大部分流行的 Linux 发行版本都包含基本相同的软件包，这也多少降低了用户选择的盲目性。作为本章例子的 Slackware 的 Linux 发行版本就很流行。

2.1.1　从 Internet 上获取 Linux 软件

获取 Linux 软件最方便的方式是在 Internet 上使用 FTP 程序。附录 B 列出了一组存放有 Linux 软件的 FTP 站点地址。其中之一是 sunsite.unc.edu，该地址的/pub/Linux/distributions 目录下有一些不同的 Linux 发行版本。

大部分这类 FTP 站点地址中的发行版本包含的是一组磁盘映象文件，其中每个文件对应于一张软盘。在 MS-DOS 中执行 RAWRITE.EXE 程序可将每个映象文件的内容写到一张软盘上，从而形成一组 Linux 系统的安装盘。大部分 FTP 站点地址中都有 RAWRITE.EXE 程序，其中在 sunsite.unc.edu 处的完整路径为/pub/Linux/system/Install/rawrite。

取得一组 Linux 系统安装盘之后，用其中一张名为“boot diskette”的安装启动盘启动微机就可以开始安装了。Linux 系统通常是直接从软盘安装的，但有些发行版本允许从硬盘上的 MS-DOS 分区安装，甚至还可以通过局域网来安装。每个发行版本都应该有描述各种有效的安装方式的文档。

每个 FTP 站点地址处提供的 Linux 发行版本，都用 README 文件来描述如何下载文件并为安装系统制作软盘。在获取 Linux 之前用户应仔细阅读该文件，以避免遇到不必要的麻烦。最后别忘了在用 FTP 程序下载 Linux 软件前先使用 binary 命令，将传输设置为二进制方式。有关 FTP 程序的使用请参考附录 B 或相关的 Internet 书籍。

用户也可以通过邮购获得 Linux 软件的软盘、磁带和 CD-ROM 版本。

2.1.2 Slackware 的 Linux 发行版本的获取

1．Slackware 版本的组成。Slackware 是由 Patrick Volkerding 维护的较流行的 Linux 发行版本。它安装方便且十分完整，并可以从 Internet 上免费获取或通过邮购 CD-ROM 取得。Slackware 发行的 Linux 套装软件包含了一系列磁盘组，每组磁盘包含一些特定类型的软件。用户可以选择安装其中某些组，等以后需要时再对其他组单独安装。每个磁盘组有一个英文标识，见表 2.1 中的介绍。

表 2.1 Slackware 发行版的磁盘组

标识	组 成 简 介
A	基本系统，可以满足系统的基本运行。
AP	各种附加信息和应用程序，如手册页，groff，ispell，term，joe，jove，ghostscript，sc，bc 和其他小程序。
D	程序开发工具，包含 GCC/G++/Objective C，make，byacc，GNU 的 bison、flex、gdb 及 C 语言库，SVGAlib，ncurses，clisp，f2c，p2c，m4，perl 和 rcs。
E	文本编辑器 GNU Emacs 19.25。
F	一组用户常见的问题的解答和其他文档。
I	GNU 的 Info 软件和 Info 页以及能被 info 或 Emacs 读出的各种软件的文档。
N	网络应用程序，包括 TCP/IP，UUCP，mailx，dip，deliver，elm，pine，smail，cnews，nn，tin 和 trn。
OOP	面向对象的程序设计工具 GNU Smalltalk 和 Smalltalk 与 X(STIX)的接口。
Q	Linux 内核的源程序及映象。
TCL	图形界面接口工具 Tcl，Tk，TclX，blt 和 itcl。
Y	一组游戏程序。
X	XFree86 的基本系统及 libXpm，fvwm 和 xlock。
XAP	X 应用软件，包括 X11 ghostscript，libgr13，seyon，workman，xfilemanager，xv，GNU chess 和 xboard，xfm，ghostview 及一些 X 游戏。
XD	X11 程序开发工具，X11 库，服务器连接器和 PEX 支持。
XV	Xview，Xview 库和 Open Look 的窗口管理器。
IV	Interviews 库，包括文件及 doc 和 idraw 程序。
OI	ParcPlace 的 Object Builder 和 Object Interface Library。
T	文章排版系统 TeX 和 LaTeX。

其中 A 组各磁盘能装入 1.2M 的软盘,而其他各组需要 1.44M 的软盘。只有 1.2M 软盘驱动器的用户可先安装基本系统,然后将其他各组下载到硬盘上并从硬盘继续安装。安装时 A 组磁盘是必需的,其他各组是可选的。作者推荐用户至少应安装 A,AP,D 和 N 组,另外若用户要使用 X Window 系统还须安装 X 组。

2. Slackware 版本的下载。许多 FTP 站点都有 Slackware 版本的 Linux 软件,较常用的有:

- ftp.pkv.edu.cn:/pub/slackware/slackware3.1 (中国)
- sunsite.unc.edu:/pub/Linux/distributions/slackware
- tsx-11.mit.edu:/publ/linux/packages/slackware
- ftp.cdrom.com:/pub/linux/slackware

  其中 ftp.cdrom.com 是 Slackware 版本的最初发行地址。

  安装前,下列文件是必须下载的。

- README 和 SLACKWARE-FAQ 文件。用户在安装前应仔细阅读这些说明文件。
- 安装启动盘的磁盘映象文件。使用 RAWRITE.EXE 可将这些文件写入一张软盘,使之成为 Linux 系统的安装启动盘。bootdsks.144 目录下放的是一系列 3.5 英寸 1.44M 软盘的映象文件,而 bootdsks.12 目录下放的是 5 英寸 1.2M 软盘的映象文件。用户只需从这些文件中选出合适的建立启动盘。不同的映象文件包含了启动时不同的系统配置,具体说明如下:
  - bare.gz:只包含 IDE 硬盘驱动程序,针对用户若只有 IDE 硬盘控制器且不从 CD-ROM 或网络安装的用户。
  - cdu31a.gz:包含 IDE,SCSI 和 Sony SDU31A/33AF 驱动程序。
  - mitsumi.gz:包含 IDE,SCSI 和 Mitsumi CD-ROM 驱动程序。
  - net.gz:包含 IDE 和网络驱动程序。
  - sbpcd.gz:包含 IDE,SCSI 和 SoundBlaster Pro/Panasonic CD-ROM 驱动程序。
  - scsi.gz:包含 IDE,SCSI 和 SCSI CD-ROM 驱动程序。
  - scsinet.gz:包含 IDE,SCSI,SCSI CD-ROM 和网络驱动程序。
  - sony535.gz:包含 IDE,SCSI 和 Sony 525/531 CD-ROM 驱动程序。
  - xt.gz:包含 IDE 和 XT 硬盘驱动程序。

若用户需要从 SCSI 硬盘,CD-ROM 或网络上安装 Linux 系统,就须相应的 SCSI,CD-ROM 或网络的驱动程序。由于不同的硬件设备可能存在冲突,应尽量选择需要的驱动程序。

- 安装基础盘的磁盘映象文件。使用 RAWRITE.EXE 可将这些文件写入一张软盘,使之成为 Linux 系统的安装基础盘。与安装启动盘类似,rootdsks.144 和 rootdsks.12 目录下分别存放了一系列 3.5 英寸 1.44M 和 5 英寸 1.2M 软盘的映象文件,用户只需使用其中合适的映象文件。每个映象文件具体说明如下:
  - color144.gz:适用于 1.44M 软盘的彩色菜单式的安装程序。多数用户应选择这

一文件。

- umsds144.gz: 适用于 1.44M 软盘的彩色菜单式的安装程序,用于安装 UMSDOS 文件系统。这将允许用户将 Linux 软件安装在 MS-DOS 系统下以避免对硬盘重新分区。以后将对此详细讨论。
- tty144.gz: 适用于 1.44M 软盘的命令行终端方式的安装程序。如果 color144.gz 文件不工作,用户应尝试使用此文件。
- colrlite.gz: 适用于 1.44M 软盘的彩色菜单式的安装程序。
- umsds12.gz: 类似 umsds144.gz,只是适用于 1.2M 软盘。
- tty12.gz: 适用于 1.44M 软盘的命令行终端方式的安装程序。

- GZIP.EXE。它在 install 子目录下是 gzip 压缩程序的 MS-DOS 版本,可用来压缩安装启动盘和基础盘上的文件。
- RAWRITE.EXE。它在 install 子目录下是 MS-DOS 程序,用来建立安装启动盘和基础盘。
- 目录 a1,a2 和 a3 下的文件。它们构成了安装 Linux 所必需的 A 磁盘组。用户将这些文件从网络下载后,可用 MS-DOS 的 copy 命令放在不同的软盘上或硬盘上的不同子目录下。下载文件时应注意使用"mget * "命令,而不是"mget *.* "命令。
- ap1,ap2 等目录下的文件。用户可以根据自己的选择下载这些文件,同样将它们放在不同的软盘上或硬盘上的不同子目录下。

3. 使用 Slackware 版本的 CD-ROM。用户也可使用含有 Slackware 版本 Linux 软件的 CD-ROM 来安装 Linux 系统。多数 Slackware 的 CD-ROM 只是 FTP 地址中 Slackware 版本的复制。因此用户不必从 Internet 上下载 Linux 软件的各目录和文件,只需从 CD-ROM 上得到这些文件即可。

4. 制作 Linux 安装启动盘和基础盘。在 MS-DOS 系统下,启动盘和基础盘的磁盘映象文件应先用 GZIP.EXE 程序还原后,再使用 RAWRITE.EXE 程序将其写入已格式化且完好无损的软盘上。以 bare.gz 和 color144.gz 为例,具体过程如下。

```
C:\> GZIP -D BARE.GZ                    (产生 BARE 文件)
C:\> GZIP -D COLOR144.GZ                (产生 COLOR144 文件)
C:\> RAWRITE
```

这时插入空的软盘,并按提示键入要写入的文件名(如 BARE 或 COLOR144)及软盘驱动器符号,RAWRITE.EXE 程序就会将相应的磁盘映象写入软盘。用这种方法制作出的 Linux 安装启动盘和基础盘已不再是 MS-DOS 格式的软盘,而是 Linux 格式的软盘。

用户也可在 Unix 系统下制作 Linux 安装启动盘和基础盘。具体过程如下。

(假设 Unix 系统中软盘驱动器对应的设备文件名为/dev/rfd0)

```
$ gunzip bare.gz                        (产生 bare 文件)
$ gunzip color144.gz                    (产生 color144 文件)
(用户插入一张软盘)
$ dd if= bare of= /dev/rfd0 obs= 18k
```

(用户插入另一张软盘)

$ dd if= color144 of= / dev/ rfd0 obs= 18k

5. 准备所需的 Slackware 各磁盘组。Slackware 版本的 Linux 软件既可以从软盘安装,也可以从硬盘的 MS-DOS 分区安装。当然,从硬盘安装需占用不少硬盘空间,但这比从软盘安装更快也更可靠。由于 Linux 系统无法从压缩后的 MS-DOS 分区上读取文件,因此这些分区不能使用诸如 Double Space, Stacker 等 MS-DOS 系统下的磁盘实时压缩工具。

若用户打算从硬盘安装 Linux 系统,就只需建立一个主目录(如 SLACKWARE),再在主目录下为各磁盘建立子目录(如 A1, A2, AP1, AP2 等),并将各磁盘的文件下载到相应的子目录下。

若用户打算从软盘安装 Linux 系统,就应先准备好足够多的 MS-DOS 系统下已格式化且完好无损的软盘,然后将不同目录下的文件下载到不同的软盘上,同时别忘了在每张盘的标签上记下该盘的编号。

2.2 Linux 系统的安装准备

2.2.1 系统安装的总过程

尽管不同版本的 Linux 软件各有不同,但总体的安装过程是相同的,大概分为以下几步:

1. 对硬盘重新分区。如果硬盘上已经安装了其他操作系统,用户应对硬盘重新分区,从而空出 Linux 操作系统所需的硬盘空间。

2. 启动 Linux 系统。

3. 建立 Linux 分区。

4. 建立 Linux 交换区和文件系统。

5. 在文件系统上安装 Linux 软件。

6. 安装用于从硬盘启动 Linux 系统的 LILO 启动程序,或制作一张启动 Linux 系统的软盘。

7. 安装完毕,重新启动 Linux 系统。

多数 Linux 的发行版本都有一个安装程序带领用户一步步地安装 Linux 系统,同时也可以自动进行上述过程的某些步。安装 Slackware 的 Linux 版本时,用户只需对硬盘重新分区,然后启动 setup 命令进入安装过程的其他各步。

安装 Linux 系统并不困难,但需记住许多细节。因此最好的方法就是记录下整个安装过程,记下所有输入和用户认为异常的输出。遇到麻烦时,用户就可以追溯以前的操作,找到原因。此外,这样的一份记录在向其他人求助或为别人提供帮助时也是很有用的。

2.2.2 硬盘分区简介

一个硬盘可被分为几个分区,每个分区安装一个独立的操作系统。比如,一个硬盘的

三个分区分别装入 MS-DOS, OS/ 2 和 Linux 操作系统。如果硬盘已经被其他操作系统占满了,用户又要安装 Linux 操作系统,就需要改变其他分区的大小,即重新分区,以留下足够的空间安装 Linux 系统。用户可以建立多个 Linux 使用的分区,这些分区可以存放 Linux 软件,也可以作为 Linux 系统的交换区。

硬盘上的第一个扇区(0 柱,0 头,1 号扇区)是硬盘的主引导扇区。它是用来启动系统的,里面有一个分区表记录了该硬盘上各分区的位置和大小。

MS-DOS 中关于分区有三个概念:基本分区,扩展分区和逻辑分区。基本分区是最常用的,但由于分区表尺寸的限制,一个硬盘只能有四个基本分区。为了突破这一限制,就需要扩展分区。扩展分区作为四个基本分区之一而存在,但它并不存放用户数据,相反,它可被划分为多个真正存放数据的逻辑分区。MS-DOS 系统下,每个硬盘只能有一个扩展分区,但它可以覆盖整个硬盘,并产生多个逻辑分区。MS-DOS 将它在各硬盘中的基本分区依次命名为 C:, D: 等,若有扩展分区,则再对各扩展分区中的各逻辑分区向后依次命名。

2.2.3 Linux 格式的分区

在对硬盘重新分区前,用户应先了解 Linux 系统需要多少空间和 Linux 系统怎样使用这些空间。

在 Unix 系统中,文件被存放在文件系统中。所谓文件系统就是硬盘(或其他存储介质如 CD-ROM 和软盘等)中符合特定格式的用来存放文件的一个区域。每个文件系统都覆盖了目录树的一个部分。举例来说,系统中可有一个文件系统存放/ usr 目录下的所有文件,同时另有一个文件系统存放/ tmp 目录下的所有文件。根文件系统是最高层的文件系统,它对应了系统的根目录/ 。

在 Linux 系统中,每个硬盘上的文件系统都有独自的分区。比如,若“ / ”与“ / usr ”是两个文件系统,硬盘上就必须有两个分区分别存放它们。

安装 Linux 系统前,用户必须准备好存储 Linux 软件的文件系统。任何 Linux 系统都至少有一个文件系统(作为根文件系统),因此也至少需要一个分配给 Linux 系统的硬盘分区。许多 Linux 用户倾向于将所有文件存放在根文件系统上,这样只需一个文件系统,通常比多个文件系统更容易管理。当然,用户可自由选择安装多个文件系统。使用多个文件系统通常是为了安全。万一一个文件系统损坏了,通常其他文件系统还是完好的。相反,若只使用根文件系统,那么一旦它损坏了,用户的所有文件都将丢失。另外,由于当前一个文件系统不能分布在多个硬盘上,多个文件系统还可使用户使用多个硬盘上的空间。

另一个需要考虑的问题是为 Linux 系统交换区准备硬盘分区。系统交换区是作为系统虚拟存储器的一块区域。用户有两种方式设置系统交换区:一种是使用 Linux 文件系统上的一个交换文件,这类文件可在安装完 Linux 软件后建立;另一种是建立一个或多个独立的作为系统交换区用的交换分区,多数用户选择这种方式。

一个交换文件或交换分区最大可有 16MB。若需要更大的系统交换空间,用户可以建立多达 8 个这类交换区。尽管使用交换区不是必须的,但若系统的物理内存小于 16M,建立交换区是十分必要的。

总之,Linux 系统一般需要两类分区:文件系统的分区和交换分区。文件系统的大小

与用户要安装的软件数量及使用的 Linux 发行版本有密切的关系。一个小的 Linux 系统只需 20M 或更少的空间,而一个大的系统至少需要 80M 空间。此外,用户还应考虑为以后安装其他软件留出适当的空闲空间。交换区的大小取决于系统虚存的需求量,一般可设置为系统物理内存大小的两倍,即在 4M 物理内存的系统上 8M 的交换区应是足够了。其实,交换空间的大小取决于用户使用的软件的需求量,一个内存极大的系统甚至根本不需要交换区。

2.2.4 对硬盘重新分区

如果用户使用空的硬盘来安装 Linux 系统,就可以跳过这一节。

通常改变一个分区大小的方法是删除它后再重新建立一个分区。这样会删掉分区中的所有文件,因此用户应先备份这些文件,重新分区后再将它们装回去,但分区缩小后,原来的文件不一定完全能装回新的分区,也许需要删除一些软件。现在已有了一些不破坏 MS-DOS 分区中原来的文件并对硬盘重新分区的程序,其中之一是 FIPS 程序,许多 Linux 的 FTP 站点都带有这一程序。

fdisk 是对硬盘重新分区时常用的程序。每个操作系统都有类似的程序。MS-DOS 下该程序用 FDISK 命令启动。本章介绍的 MS-DOS 下用 FDISK 程序对硬盘重新分区的方法也基本适用于其他操作系统。用户应注意每个操作系统的 fdisk 程序只能建立和修改本操作系统的分区,因此,应不要使用 MS-DOS 下的 FDISK 命令建立 Linux 分区,反之也一样。有关 fdisk 程序使用的具体细节请参考各操作系统的使用手册。

本章的例子是一个单硬盘系统,MS-DOS 占用了整个硬盘空间,并且只有一个分区,相应的盘符是 C:。用户应先将分区中的文件备份,并且保存一张可启动的 MS-DOS 系统盘,以将备份的文件重新装回新的 MS-DOS 分区中。这时,从 MS-DOS 的系统软盘启动,并键入命令"FDISK C:"启动 FDISK 程序。

然后选择 FDISK 菜单项"Display partition table"来显示分区表,并记录下显示的分区信息。要删除一个分区,选择菜单项"Delete an MS-DOS Partition or Logical DOS Drive",然后输入要删除的分区类型(基本分区,扩展分区或逻辑分区)及分区号并注意检查所有的警告信息。要建立一个分区,选择菜单项"Create an MS-DOS Partition or Logical DOS Drive",然后输入要建立的分区类型及大小(以 1MB 为单位)。

这时,退出 FDISK 程序,并使用命令"FORMAT C: /S"对新的分区格式化。然后将备份的文件装回新的 MS-DOS 分区。

2.3 Linux 系统的安装

2.3.1 启动 Linux 系统

将 Linux 的安装启动盘插入软盘驱动器,重新启动系统。这张软盘包含了一个很小的 Linux 系统。多数 Linux 发行版本允许用户在启动时输入一些硬件参数迫使系统对某些硬件进行识别。Slackware 的发行版本会出现提示符"boot:",提示用户输入硬件参数,而

其他某些版本需在启动时按下 shift 和 ctrl 键进入这种状态。同时,系统还会给出一些信息提示用户如何操作。若用户没有特殊的参数输入,按下回车键即可。

系统启动 Linux 内核后会提示用户登录,用户可用“root”或“install”进行登录,然后开始建立 Linux 分区等安装过程。

2.3.2 Linux 系统中的驱动器和硬盘分区

Linux 系统为系统中的每个驱动器及硬盘分区设立了专门的设备文件名。在 MS-DOS 下,软盘驱动器的名字为 A: 和 B:,硬盘的分区分别以 C:,D: 等依次命名。Linux 下设备的命名则很不同。

Linux 系统的/dev 目录下包含了所有与系统硬件设备对应的设备文件名。比如,/dev/mouse 代表了系统中的鼠标器。同时,硬盘上的不同分区也在 Linux 系统中有相应的设备文件名。表 2.2 中列出了本章要用到的系统中各种磁盘设备和分区的名字。

表 2.2 设备与文件名对照表

设　　备	文件名
第一个软盘驱动器(A:)	/dev/fd0
第二个软盘驱动器(B:)	/dev/fd1
第一个 IDE 硬盘(整个硬盘)	/dev/hda
第一个 IDE 硬盘,1 号基本分区	/dev/hda1
第一个 IDE 硬盘,2 号基本分区	/dev/hda2
第一个 IDE 硬盘,3 号基本分区	/dev/hda3
第一个 IDE 硬盘,4 号基本分区	/dev/hda4
第一个 IDE 硬盘,1 号逻辑分区	/dev/hda5
第一个 IDE 硬盘,2 号逻辑分区	/dev/hda6
……	
第二个 IDE 硬盘(整个硬盘)	/dev/hdb
第二个 IDE 硬盘,1 号基本分区	/dev/hdb1
……	
第一个 SCSI 硬盘(整个硬盘)	/dev/sda
第一个 SCSI 硬盘,1 号基本分区	/dev/sda1
……	
第二个 SCSI 硬盘(整个硬盘)	/dev/sdb
第二个 SCSI 硬盘,1 号基本分区	/dev/sdb1
……	

若系统中不存在表 2.2 中的某个设备,则/dev 目录下也没有该设备对应的文件名。比如,若第一个 IDE 硬盘中只有两个基本分区,则/dev/hda3 和/dev/hda4 将不存在,硬盘上各逻辑分区从/dev/hda5 开始依次向后编号。在以后的安装过程中,用户必须使用这些设备文件名来标识硬盘和各个分区。

2.3.3 建立 Linux 系统的硬盘分区

用户登录进系统后,系统出现提示符“# ”,然后用户可键入 fdisk 命令开始建立

Linux 分区, 格式为:

fdisk < drive>

其中< drive> 为要增加 Linux 分区的硬盘设备名, 缺省值为/dev/hda。用户需对每个要建立 Linux 分区的硬盘执行 fdisk 命令。

启动 fdisk 命令后, 系统提示输入命令,

Command (m for help):

键入 m, 可得到 fdisk 中所有命令, 见表 2.3 中的功能简介。

表 2.3 fdisk 命令

显示器输出	说明
# fdisk /dev/hda	启动 fdisk 程序
Command (m for help): m	fdisk 提示用户输入命令(m 为帮助)
Command action a toggle a bootable flag	切换分区可启动标志
d delete a partition	删除一个分区
l list known partition types	列出已知的分区类型
m print this menu	输出本菜单
n add a new partition	建立一个分区
p print the partition table	列出当前分区表
q quit without saving changes	忽略已做的修改并退出 fdisk
t change a partition's system id	改变分区的系统标识号
u change display/entry units	改变显示/入口单元
v verify the partition table	检验分区表
w write table to disk and exit	保存已做的修改并退出 fdisk
x extra functionality (experts only)	其他功能(专家使用)

首先, 用户应使用 p 命令显示硬盘当前的分区表, 并对此作记录, 以备以后参考。下面是帮助使用的例子:

```
Command (m for help): p
Disk /dev/hda: 16 heads, 38 sectors, 683 cylinders
Units = cylinders of 608* 512 bytes

   Device    Boot  Begin  Start   End   Blocks   Id        System
/dev/hda1     *      1      1     203   61693    6   DOS 16-bit > = 32M

Command (m for help):
```

这个例子中, 硬盘上只有一个 MS-DOS 分区/dev/hda1, 是位置从 1 号柱面起始到 203 号柱面结束, 大小 61693KB(约 60MB)的一块区域。该硬盘共有 683 个柱面, 因此其余 480 个柱面可留给 Linux 系统的分区使用。

这时, 使用 n 命令建立新的 Linux 分区。例子中将建立两个 Linux 分区, 它们均为基本分区, 在 Linux 系统中的名字将为/dev/hda2 和/dev/hda3。

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

```
Command action
e extended
p primary partition (1-4)
p
Partition number (1-4): 2
First cylinder (204-683): 204
Last cylinder or + size or + sizeM or + sizeK (204-683): + 80M
Warning: Linux cannot currently use 33090 sectors of this partition
Command (m for help): n
Command action
e extended
p primary partition (1-4)
p
Partition number (1-4): 3
First cylinder (474-683): 474
Last cylinder or + size or + sizeM or + sizeK (204-683): + 10M
```

fdisk 先询问是建立基本分区(p)还是扩展分区(e)。本例选择建立基本分区。然后，fdisk 询问要建立的分区号及分区的起始柱面和大小。由于 1 号分区已是 MS-DOS 分区，因此，本例建立 2 号分区，起始柱面为 204，分区大小为 80M，以后用来存放文件系统。同样，再建立起始柱面为 474，大小为 10M 的 3 号分区作为系统交换区。

随后，用户需将 3 号分区标记为 Linux 的交换分区(代号为 82)。具体做法是使用 l 命令显示各种已知分区类型的编号，再使用 t 命令将 3 号分区的类型该为“Linux Swap”所对应的编号。这时，再键入 p 命令得到新的分区表，用户同样应对此作详细记录，并检查分区的位置数据以保证各分区没有重叠的柱面。可以看到系统输出中，82080 和 10336(单位为 kB)分别与 80M 和 10M 相近似。

```
Command (m for help): p
Disk /dev/hda: 16 heads, 38 sectors, 683 cylinders
Units = cylinders of 608* 512 bytes

   Device    Boot  Begin   Start   End    Blocks   Id   System
/dev/hda1     *       1       1   203     61693    6   DOS 16-bit > = 32M
/dev/hda2           204     204   473     82080   81   Linux/MINIX
/dev/hda3           474     474   507     10336   82   Linux swap
```

本例中硬盘的其余柱面(柱面 508-683)没有用到，以后用户可以再建立新的分区。最后，用户应使用 w 命令将新的分区表写回硬盘并退出 fdisk 程序。

为了稳妥起见，用户应该重新从软盘启动系统以使系统读入新的分区表，然后继续安装过程。

2.3.4 对系统的交换分区格式化

建立了硬盘的交换分区之后，用户还需使用 mkswap 命令对每个交换分区进行格式

化,以备 Linux 系统使用。格式化完成后,系统命令 swapon 可打开某交换分区,将其投入系统的实际使用中。mkswap 和 swapon 命令的格式为:

```
mkswap -c < partition>  < size>
swapon < partition>
```

其中< partition> 为交换分区对应的设备文件名;< size> 为分区的大小,实际值可使用分区表中的值(本例中为 10336);参数"-c"表示 mkswap 程序检查分区中有无坏块。下面是本例实际使用的命令行。

```
# mkswap -c /dev/hda3 10336
# swapon /dev/hda3
```

2.3.5 建立文件系统

使用 Linux 分区存储文件前,用户应先在每个 Linux 分区上建立文件系统。这一过程类似于 MS-DOS 中分区使用前必须对其进行的格式化操作。Linux 系统同时支持多种文件系统,其中包括诸如 MS-DOS 等其他操作系统使用的文件系统。每种文件系统都有自身的特性(如文件名长度和最大文件尺寸等的限制)。

Linux 系统本身最常用的是二级扩展文件系统(Second Extended Filesystem),简称 ext2fs。ext2fs 是一种效率和灵活性都很高的文件系统,它支持长度不超过 256 字节的文件名和大小不超过 4T(4× 10^{12})字节的文件。本书第 4 章还将介绍 Linux 系统支持的其他文件系统,但在此用户最好使用 ext2fs 文件系统。

mke2fs 命令在指定的 Linux 分区上建立 ext2fs 文件系统,具体格式为:

```
mke2fs -c < partition>  < size>
```

其中各参数的意义同 mkswap 命令相同。本例使用

```
mke2fs -c /dev/hda2 82080
```

建立 ext2fs 文件系统。

2.3.6 安装 Linux 软件

完成前面各项准备工作后,就可以安装 Linux 系统的软件了。多数 Linux 发行版本都有一个安装程序引导整个安装过程。某些版本还允许多种安装方式,如从软盘安装,从硬盘的 MS-DOS 分区安装,从 CD-ROM 安装,甚至还可以通过 TCP/IP 局域网使用 FTP 或 NFS(网络文件系统)安装。某些 CD-ROM 的安装程序还允许用户选择是否将大部分软件保留在 CD-ROM 上。

Slackware 的 Linux 发行版本只需用户使用 fdisk 命令建立 Linux 分区(如果需要的话,再用 mkswap 和 swapon 命令建立系统交换区),然后运行 setup 程序向硬盘安装 Linux 系统。

setup 命令启动后,系统出现主菜单,整个安装过程基本上由这个主菜单来控制。用户可以用方向键在菜单项中移动,按回车键或空格键后运行该菜单项的功能。主菜单中的命令,见表 2.4。

表 2.4　Setup 主菜单及说明

菜单项	使用说明
Help	显示 setup 的帮助文件。
Keymap	确定系统使用的键盘类型及各键的映象。
Quick	允许用户从快速安装方式和详细安装方式中选择。缺省为详细安装方式,这也是系统推荐的安装方式。
Make tags	允许安装专家建立一个专门的标记文件以记录预先选择的软件包。初级用户不必关心这项的使用。
Addswap	这是多数用户需选择的第一个菜单项。进入此项后,系统显示已有的 Linux 交换分区(fdisk 中标记为"Linux swap"的分区)。用户可以指定系统使用那些交换分区,以及是否对这些分区执行 mkswap 命令。如果用户在"2.3.4 对系统的交换分区格式化"中已经执行了这个命令,就不必再执行了。即使这样,使用此菜单项也是必要的,它可以保证这些交换分区在系统安装后的有效性。
Target	允许用户选择在哪个分区上安装 Linux 系统的软件。进入此项后,系统显示已有的 Linux 分区(fdisk 中标记为"Linux native"的分区)。用户可以选择在哪个分区上记录根文件系统和建立何种类型的文件系统。ext2fs 是理想的文件系统类型。用户还可以选择其他 Linux 系统可使用的分区及该分区的文件系统在目录树中的位置。比如,用户为/usr 建立了一个单独的分区,就可以安装这个分区并指定它安装在目录/usr 上。如果用户在"2.3.5 建立文件系统"中已经建立了文件系统,也应选择此菜单项来配置 Linux 系统的目录结构。
Source	允许用户选择从硬盘,软盘或光盘上安装 Linux。用户在选择了其中一种设备之后会被提示输入 Slackware 文件所在的分区,分区的类型,文件所在的目录名。如果 Slackware 报告出错,这可能是文件读取有问题。可参阅"2.6.3 安装 Linux 系统软件时出现的问题"。
Disk Sets	允许用户选择所需安装的磁盘组,其中 A 磁盘组是必需的。
Install	选择 Install 菜单项后,系统开始安装所选择的软件。用户会被提示选择系统输出的信息量。多数用户选择 normal(普通)方式。对于每个要安装的磁盘组,系统先安装必要的部分,然后提示用户是否安装系统推荐安装的部分和其他可任意选择安装的部分。若是从软盘安装,系统还会提示用户更换软盘。每个软件包被安装时,系统都会给出该软件包的简要信息。用户可记录一下系统都安装了哪些软件包。
Configure	允许用户在安装完 Linux 系统的软件后需配置一下系统。

2.3.7　建立系统启动软盘或安装 LILO

安装完 Linux 系统软件后,用户应该考虑如何启动 Linux 系统。每个 Linux 的发行版本都提供了一种或多种启动方式。通常,安装程序会为用户建立一张系统引导软盘,用户可以从这张软盘引导系统后,再使用硬盘进行工作。许多 Linux 发行版本还提供了另一种通过安装 LILO 的启动方式。LILO 将被安装在硬盘的主引导扇区上,它可以用来引导硬盘上各分区中的不同操作系统。

Slackware 的发行版本中 setup 程序主菜单的 Configure 菜单项就允许用户建立一张系统引导软盘和在硬盘上安装 LILO。Configure 菜单项还允许用户设置调制解调器,鼠标器,时区等与系统相关的信息。安装 LILO 需要许多硬盘的分区信息,而且 LILO 程序还有可能和硬盘上其他操作系统设置的引导程序相冲突,因此本节推荐使用系统软盘来引导 Linux 操作系统。本书第四章将详细介绍如何安装 LILO 程序。

2.3.8 Linux 系统的其他部分的配置与安装

如果在前面的安装过程中,用户没有遇到任何麻烦,那么 Linux 系统就安装成功了。如果用户遇到了某些问题,2.6 节也许能给予一些帮助。

虽然现在系统已经可以运行 Linux 操作系统了,但要使用 Linux 的网络和与 MS-DOS 下的 Windows 类似的图形环境,还需要安装和配置 TCP/IP 网络软件和 X Window 图形软件包。本书第 2 部分将详细介绍如何配置网络,2.5 节将介绍如何安装和配置 X Window 图形系统。

2.4 Linux 系统的启动与关闭

系统安装完成后,用户需重新启动 Linux 系统,用 root 进行登录,然后就可以使用该系统了。与 MS-DOS 不同的是,用户绝对不应直接按系统机箱上的 Reset 键来重新启动系统,也绝对不应直接关掉计算机电源来关闭系统。这是因为 Linux 系统与其他 UNIX 系统一样使用内存作为硬盘的缓冲区,简单的关机或冷启动会导致内存中还未写回硬盘的信息丢失,以致渐渐地造成系统的崩溃。

Linux 提供了 shutdown 命令来关闭系统。shutdown 命令的格式为:

shutdown -r now。

键入该命令后,等待系统提示“system is halted”,然后用户可按 Reset 键重新启动系统或关掉计算机的电源。

2.5 配置 X Window 系统软件包 XFree86

2.5.1 XFree86 简介

X Window 系统是 UNIX 操作系统下功能强大的图形环境。它最初是由 MIT 设计的,以后逐渐成为 UNIX 平台上的工业标准。实际上,世界上各种 UNIX 系统都可运行 X Window 系统的某个变种。

XFree86 是由 David Wexelblat 领导的一个小组将 MIT 的 X Window 系统的 X11R6 版移植到 80386/80486/Pentium 系列 CPU 下运行的 UNIX 平台(如 System V/386、386BSD、Linux 等)上而形成的免费软件包,其中包含了可执行程序,支持文件,函数库及其他工具。

下面将简要介绍如何在 Linux 系统上安装和配置 XFree86 软件包。Xfree86 随带的

文档中有更详细的说明。使用和掌握 X Window 系统则不在本书的讨论范围之内，用户可参考其他关于 X Window 的书。

2.5.2 安装 XFree86 的硬件要求

每个 XFree86 的版本都支持一些特定的显示芯片组。用户可以从显示卡的手册中查到该卡使用的显示芯片组。用户也可以执行 XFree86 软件包中的 SuperProbe 程序来确定系统显示卡使用的显示芯片组。1995 年 2 月发行的 XFree86 3.11 版支持以下各显示芯片组。

- SVGA 类的显示芯片组
  - Tseng ET3000，ET4000AX，ET4000/W32
  - Western Digital/Paradise PBGA1
  - Western Digital WD90C00，WD90C10，WD90C11，WD90C24，WD90C30，WD90C31，WD90C33
  - Genoa GVGA
  - Trident TVGA8800CS，TVGA8900B，TVGA8900C，TVGA8900CL，TVGA9000，TVGA9000i，TVGA9100B，TVGA9200CX，TVGA9400CX，TVGA9420
  - ATI 18800，18800-1，28800-2，28800-4，28800-5，28800-6，68800-6，68800AX，68800LX，88800
  - NCR 77C22，77C22E，77C22E+
  - Cirrus Logic CLGD5420，CLGD5422，CLGD5424，CLGD5426，CLGD5428，CLGD5429，CLGD5430，CLGD5434，CLGD6205，CLGD6215，CLGD6225，CLGD6235，CLGD6420
  - Compaq AVGA
  - OAK OTI067，OTI077
  - Avance Logic AL2101
  - MX MX68000，MX 680010
  - Video 7/Headland Technologies HT216-32
- 带图形加速器的 SVGA 类的显示芯片组
  - 8514/A
  - ATI Mach8，Mach32
  - Cirrus Logic CLGD5420，CLGD5422，CLGD5424，CLGD5426，CLGD5428，CLGD5429，CLGD5430，CLGD5434，CLGD6205，CLGD6215，CLGD6225，CLGD6235
  - S3 86C911，86C924，86C801，86C805，86C805i，86C928，86C864，86C964
  - Western Digital WD90C31，WD90C33
  - Weitek P9000
  - HT AGX-014，AGX-015，AGX-016

· Tseng ET4000/W32, ET4000/W32i, ET4000/W32p

另外系统的总线类型(如 VESA、PCI 等)与 XFree86 是否支持某一显示芯片组的显示卡无关。

除 Avance Logic, MX 和 Video 7 只支持 256 色模式外,其他各显示芯片组均同时支持 256 色和单色模式。如果显示卡配置了足够大的显示存储器(VRAM),有些显示芯片组(如 Mach32, P9000, S3 和 Cirrus 等)还支持每象素 16 位(64K 色)或 32 位(4G 色)的模式。XFree86 中的单色模式还支持如下的一些通用的 VGA 显示卡。

· Hercules 单色显示卡
· Hyundai HGC1280
· Sigma LaserView
· Appoll 单色显示卡
· Compag AVGA(64K VRAM)
· Genoa GVGA(不多于 64K VRAM)

XFree86 所支持的显示芯片组的范围将随着版本的更新而不断扩大。因此,用户应参考手中 XFree86 软件包的发行说明来确定该版本的支持范围。

本书推荐的可安装 XFree86 软件包的系统应配有 486 级或更高档的 CPU,不少于 8M 的内存和一块使用 XFree86 所支持的显示芯片组的显示卡,最好是带有图形加速器的显示卡。尽管 XFree86 可以安装在 4M 内存 16M 虚存的系统上,但那样会使系统运行得极慢,有时甚至比 8M 内存的系统慢十倍以上。

2.5.3 XFree86 系统的服务器

XFree86 系统实际运行时,有一个称为 X 服务器的模块一直在 Linux 系统中运行,它为其他(客户)程序提供图形显示方面的服务。不同的显示卡(或显示芯片组)有不同的功能和使用方法,因此 X 服务器通过与一个对应与某一类显示芯片组的显示底层服务器相互配合共同完成图形显示的功能。表 2.5 是 XFree86 软件包所提供的各种显示底层服务器。

表 2.5 显示底层服务器

显示底层服务器	对应的显示芯片组/显示卡
XF86_8514	8514/A
XF86_AGX	IIT AGX
XF86_Mach32	ATI Mach32
XF86_Mach8	ATI Mach8
XF86_Mono	单色显示模式
XF86_P9000	Weitek P9000
XF86_S3	S3
XF86_SVGA	普通的 Super VGA 显示芯片组
XF86_VGA16	普通的 EGA/VGA 显示卡
XF86_W32	Tseng ET4000/W32

2.5.4 XFree86 系统的配置

配置 XFree86 系统并不困难,若希望使带图形加速器的显示卡取得最佳的性能,就需花较多的时间了。XFree86 系统通过为每类显示芯片组提供不同的底层服务器和相同的上层软件从而为不同的显示卡的用户提供相同的功能。XFree86 底层的服务器配置文件是 XF86Config 文件,也是 XFree86 系统的主要配置文件。它包含了系统的显示卡及鼠标、显示器等输入输出设备的参数。XF86Config.eg 文件是 XF86Config 的一个例子。用户可将其复制到 XF86Config 文件作为配置的出发点。XF86Config 的联机手册详细说明了该文件的格式。这里将举例介绍 XF86Config 文件的每一个部分。用户应根据自己的系统配置对各部分进行相应的填写。

多数情况下,应该从所有显示卡都支持的较低的分辨率(如 640x480)的基本模式开始配置 XFree86 系统。将基本模式配置好以确定系统没有安装上的问题后,可再进行高级模式的配置来充分发挥显示硬件的能力。

本章只提供简要的配置方法,详细的内容请参考下列几份文件资料:

- 目录/usr/X11R6/lib/X11/doc 下的 XFree86 文件,尤其是作为 XFree86 配置教程的 README.Config 文件。
- 上述目录中不同显示芯片组对应的说明文件 README.* ,如 README.S3,README.Cirrus 等。
- XFree86 的联机手册。
- XFree86Config 的联机手册。
- 用户使用的底层服务器的相应的联机手册 XF86_* ,如 XF86_SVGA 等。

XF86Config 文件的每个部分称为一节,由一些文本行组成,每节的起始和终止分别由“Section <节名>”和“EndSection”标识。文件中以“#”开头的行为注释行,没有实际意义。

1. Files 节

```
Section "Files"
    RgbPath     "/usr/X11R6/lib/X11/rgb"
    FontPath    "/usr/X11R6/lib/X11/fonts/misc/"
    FontPath    "/usr/X11R6/lib/X11/fonts/75dpi/"
EndSection
```

这一节指示了一些 XFree86 系统文件所在的目录。其中 RgbPath 行指定 X11R6 的 RGB 颜色数据库的位置,每个 FontPath 行指定了一个放有 X11 字体文件的目录。通常,不需改变这些行,但应检查是否用户安装的所有字体都得到了说明,也就是/usr/X11R6/lib/X11/fonts 下的每个目录都应在 Files 节中有对应的一行。

2. ServerFlags 节

```
Section "ServerFlags"
#  Uncomment this to cause a core dump at the spot where a signal
#  is received.  This may leave the console in an unusable state,
```

```
#  but may provide a better stack trace in the core dump to aid in
#  debugging.
#     NoTrapSignals

#  Uncomment this to disable the < Ctrl> < Alt> < BS>  server abort
#  sequence
#     DontZap
EndSection
```

这一节指定了为 XFree86 底层服务器提供的全局标志。本节通常是空的,例子中的各行都是注释行。

3. Keyboard 节

```
Section“Keyboard”
     Protocol     “Standard”
     AutoRepeat  500 5
     ServerNumLock
EndSection
```

这一节指定了键盘的设置。例子中的设置适用于大多数系统。

4. Pointer 节

```
Section“Pointer”
     Protocol     “MouseSystem”
     Device       “/dev/mouse”
#  Baudrate and SampleRate are only for some Logitech mice
#     BaudRate    9600
#     SampleRate 150

#  Emulate3Buttons is an option for 2-button Microsoft mice
#     Emulate3Buttons

#  ChordMiddle is an option for some 3-button Logitech mice
#     ChordMiddle
EndSection
```

这一节指定了鼠标器的配置。用户只需设置 Protocol 和 Device 两项。Protocol 项指定了鼠标器使用的协议(不是鼠标器的商标或生产厂家)。Linux 系统下的 XFree86 支持的协议有:

- BusMouse
- Logitech
- Microsoft
- MMSeries
- Mouseman

· MouseSystems

· PS/2

· MMHitTab

Device 项指定了鼠标器对应的设备文件名,通常是/dev/mouse。/dev/mouse 通常是某个串行口设备文件(串行鼠标器)或总线鼠标器的设备文件的逻辑连接。用户应保证此项中指定的设备文件存在。

5. Monitor 节

```
Section“Monitor”
    Identifier “CTX 5468 NI”
#  These values are for a CTX 5648NI only!  Don't attempt to use
#  them with your monitor (unless you have this model)
    Bandwidth      60
    HorizSync      30-38, 47-50
    VertRefresh    50-90

# Modes:  Name          dotclock   horiz                    vert
ModeLine “640x480”      25         640    664    760    800  480  491  493  525
ModeLine “800x600”      36         800    824    896   1024  600  601  603  625
ModeLine “1024x768”     65        1024   1088   1200   1328  768  783  789  818
EndSection
```

这一节指定了系统显示器的设置。XF86Config 文件中可有多个 Monitor 节分别对应系统中不同的显示器或同一显示器的不同设置。通常用户只需一个 Monitor 节。

Identifier 行说明了此 Monitor 节的名字,可以是任意字符串。它用于以后标识此 Monitor 节。

HorizSync 和 VertRefresh 行分别指定了系统显示器的水平同步频率(单位是 kHz)和垂直刷新频率(垂直同步频率)(单位是 Hz)。对于多同步显示器,这两行里是一些频率的范围,而对于固定频率的显示器这两行里是一些以逗号隔开的离散频率。这两行的值可以从显示器手册的技术说明中查得,它们的作用是对 ModeLine 行中指定的各模式进行检查,以防某模式需要的频率超出范围而损坏显示器。

每个 ModeLine 行指定了显示器的一种分辨率模式的定时参数,格式为:

ModeLine < 名称> < 时钟频率> < 水平值组> < 垂直值组>

其中< 名称> 可以是任意字符串,以备以后引用该模式; < 时钟频率> 是该模式所需的点时钟频率(必须是显示卡可以产生的频率),通常是以 MHz 为单位; < 水平值组> 和 < 垂直值组> 分别包括 4 个值,它们指定了显示器的电子枪应何时向显示屏发射电子及水平和垂直同步脉冲应何时发出以开始水平和垂直回扫。

Linux 系统/usr/X11R6/lib/X11/doc 下的 modeDB.txt 和 Monitors 两个文件包含了一些显示器的 ModeLine 行,其中 modeDB.txt 文件包括 VESA 标准分辨率的定时参数。用户应先为大多数显示器都支持的这些 VESA 标准模式填写 ModeLine 行。下例是

该文件的一个部分,它描述了 640x480 显示模式的 VESA 标准的定时参数,其中点时钟为 25.175MHz。用户的显示卡必须支持 25.175MHz 的频率才能使用这一模式。

```
#  640x480@60Hz Non-Interlaced mode
#  Horizontal Sync =  31.5kHz
#  Timing:  H= (0.95us,  3.81us,  1.59us),  V= (0.35ms,  0.064ms,  1.02ms)
#
#  name        clock   horizontal  timing      vertical  timing        flags
   "640x480"  25.175  640   664   760   800   480   491 493   525
```

用户只需将上面最后一行前面加上"ModeLine"就可以将其放入 XF86Config 文件的 Monitor 节中了。这两个文件中还包含了其他显示器支持的模式的定时参数,用户一定要根据自身显示器的类型来选择适当的行,否则可能会损坏系统的显示器。若用户还不能成功地设置 ModeLine 行,可参考 VideoMode.doc 文件。该文件详细地描述了怎样根据显示器手册的技术说明来确定所支持的各分辨率描述的定时参数。

若用户觉得所设置的模式基本可以工作了,就可以稍微改动一下这些参数来尝试达到预期的结果。VideoModes.doc 文件对此也提供了一些帮助。同时用户还可调节显示器上的一些调节钮,这也许能省去调整 ModeLine 行的时间。

6. Device 节

```
Section "Device"
        Identifier  "# 9 GXE 64"
        #  Nothing yet;  we fill in these values later
EndSection
```

这一节指定了系统显示卡的参数,其中 Identifier 行可以是标识该卡的任意字符串。系统的/usr/X11R6/lib/X11/doc 目录下的 modeDB.txt, AccelCards 和 Devices 文件列出了各种显示卡的参数。同时,每种显示芯片组还有对应的 README.* 文件可供参考。用户也可以从这些文件中取得显示卡的参数并填入 Device 节中。

不过用户在开始时不必在这些文件中查找参数,而只需在 Device 节中填写 Identifier 行,系统的 X 服务器以后会自动查询系统显示卡的参数,如显示芯片组、时钟和卡上的显示内存量等。这里使用 S3 显示芯片组的# 9 GXE 64 显示卡为例来介绍如何配置显示卡。在系统中无其他任务运行时,执行命令

```
X -probeonly > /tmp/x.out 2> &1
```

(若用户使用的是 bash 作为命令处理器)

或

```
X -probeonly &> /tmp/x.out
```

(若用户使用的是 csh 作为命令处理器)

得到系统显示卡的参数。由于这段输出较长,命令中将输出记录在/tmp/x.out 文件中。查看/tmp/x.out 文件,它应包含下述一些行。

```
Xfree86 Version 3.1 / X Window System
(protocol Version 11, revision 0, vendor release 6000)
Operating System: Linux
Configured drivers:
```

S3: accelerated server for S3 graphics adaptors (Patchlevel 0)

mmio-928, s3-generic

... ...

(--) S3: card type: 386/486 localbus

(--) S3: chipset: 864 rev. 0

(--) S3: chipset driver: mmio-928

... ...

(--) S3: clocks: 25.18 28.32 38.02 36.15 40.33 45.32 32.00 00.00

这说明 XF86-S3 服务器可使用两种显示芯片组: mmio-928 和 s3-generic, 而 X 服务器找到了一个使用 mmio-928 显示芯片组的显示卡。若在执行 X -probeonly 命令前 Device 节中还没有 clocks 行, 命令输出中 S3: clocks 行还将提供了显示卡所能产生的各种时钟频率。因此用户可将下面两行加入 Device 节。用户应注意不要改变 clocks 行中各数值的顺序或删掉其中重复的数值。

Chipset "mmio-928"

Clocks: 25.18 28.32 38.02 36.15 40.33 45.32 32.00 00.00

有些带加速器的显示卡使用的是可编程的时钟芯片。这种芯片允许 X 服务器选择显示卡的时钟。此时, 输出中的 clocks 行列出的数值就不再有意义了。用户也应使用 ClockChip 行来代替 Clocks 行, ClockChip 行指定了显示卡所使用的时钟芯片的类型。用户可以查看各显示底层服务器的联机手册以取得时钟芯片的类型。同样, 有些带加速器的显示卡还要求 Device 节中的 Ramdac 行指定显示卡上 RAMDAC 芯片的类型, 通常 X 服务器可以准确地查询出 RAMDAC 芯片。

某些显示卡可能还要求用户填入一些额外的 Option 行来进一步说明显示卡的特性, 这些行通常只是为了显示卡能取得最佳的性能。各种服务器的手册页和/usr/X11R6/lib/X11/doc 目录下的各文件对此有详细的介绍。

最后, 例子中得到的 Device 节大致如下。

```
Section "Device"
    Identifier "# 9 GXE 64"
    Chipset "mmio-928"
    ClockChip "icd2061a"
    Option "number-nine"
    Option "dac-8-bit"
EndSection
```

7. Screen 节

```
Section "Screen"
    Driver      "Accel"
    Device      "# 9 GXE 64"
    Monitor     "CTX 5468 NI"
    Subsection  "Display"
```

```
        Depth       16
        Modes       "1024x768" "800x600" "640x480"
        ViewPort    0   0
        Virtual     1024   768
    EndSubsection
EndSection
```

这一节为特定的 XFree86 底层服务器指定了一套显示器和显示卡的组合。

Drive 行指定了用户要使用的 X 服务器,不同的显示底层服务器使用不同的 X 服务器,其对应关系见表 2.6。用户必须保证/usr/X11R6/bin/X 是所要使用的 X 服务器的逻辑连接。

表 2.6 X 服务器与显示底层服务器对照表

X 服务器	显示底层服务器
Accel	XF86_S3, XF86_Mach32, XF86_Mach8, XF86_8514, XF86_P9000, XF86_AGX 和 XF86_W32
SVGA	XF86_SVGA
VGA16	XF86_VGA16
VGA2	XF86_Mono
Mono	XF86_Mono 和 XF86_VGA16 中的非 VGA 的单色模式

Device 行指定了在某个 Device 节中说明的显示卡的标识。同样,Monitor 行指定在某个 Monitor 节中说明的显示器的标识。

Display 子节定义了对应于显示卡/显示器组合的 XFree86 底层服务器的一些性质。其中一些行的意义如下。

- Depth 行指定了每象素占用几个存储位,通常的值 8 表示 256 色。VGA16 服务器的值为 4,而 VGA2 单色服务器的值为 1。有些带图形加速器的显示卡还支持 16,24 和 32 这几个值。
- Modes 行指定了用户希望使用的一些分辨率模式的标识,即是在 Monitor 节中 ModeLine 行中的标识。XFree86 系统启动时先使用第一个模式。用户随后可使用 Ctrl-Alt-"+"(数字键盘)和 Ctrl-Alt-"-"(数字键盘)来切换实际显示模式。一般来说,用户应先尝试较低分辨率的模式,然后再配置更高分辨率的模式。
- Virtual 行指定了虚拟屏幕的大小。XFree86 系统能使用显示卡上的所有存储器来扩展实际屏幕的尺寸。当用户在屏幕的边缘向外移动鼠标器时,屏幕就会滚动,出现超出屏幕大小的那部分平面的图象。通常,1M 字节的显示卡可以支持 1024x768,256 色(每象素 8 位)的虚拟屏幕。

  虚拟屏幕可以充分使用显示卡上的存储空间,但也受到其限制。fvwm 程序是一个类似的屏幕管理器,它使用了不同的技术可提供更大的虚拟屏幕。大多数 Linux 系统都使用 fvwm。fvwm 的联机手册提供了更详细的信息。
- ViewPort 行指定了 XFree86 系统启动时屏幕左上角在虚拟屏幕中的位置,一般

设为(0, 0)。若不写这一行,虚拟屏幕的正中一块将被实际显示出来,这也许并不是用户所希望的。

另一些行对于使 XFree86 系统开始工作不是必要的,详细的说明可参考 XF86Config 的联机手册。

2.5.5 XFree86 系统的启动

当用户填写完 XF86Config 文件后,就可以使用/usr/x11R6/bin 目录下的 startx 命令来启动 XFree86 系统了。startx 使用 xinit 程序来启动 X 服务器,而 xinit 初始化图形系统后执行用户主目录下.xinitrc 中的命令。若.xinitrc 文件不存在,则系统将执行缺省的文件/usr/X11R6/lib/X11/xinit/xinitrc 中的命令。如下是一个较标准的.xinitrc 文件的内容。

```
#!/bin/sh
xterm -fn 7x13bold -geometry 80x32+ 10+ 50 &
xterm -fn 9x15bold -geometry 80x34+ 30-10 &
oclock -geometry 70x70-7+ 7 &
xsetroot -solid midnightblue &
exec twm
```

此文件将启动两个 xterm 任务,一个 oclock 任务,并且将屏幕的背景颜色设为 midnightblue,即一种深蓝色,最后它将程序控制权传递给窗口管理器 twm。注意,前四条命令都是后台执行的(命令的最后有字符 &),而 twm 是用 exec 语句启动的。这表明 twm 程序将替代 xinit 程序继续执行下去。一旦 twm 程序执行完毕,X 服务器也将自动地被关闭。用户也可以使用可提供虚拟屏幕的 fvwm 程序代替 twm 程序作为窗口管理器。用户可任意修改.xinitrc 文件,但应保证此文件最后一条命令是用 exec 语句在前台执行的,否则 X 服务器将在启动后被立即关闭。用户也可以同时按下 Ctrl、Alt 和退格键来关闭 X 服务器,从而退出 XFree86 窗口系统。

启动 XFree86 系统后,用户就可以自由地使用 Linux 系统下的 X Window 环境了。有关 X Window 的使用请参考其他讲述 X Window 的书籍。

2.6 疑难解答

几乎所有第一次安装 Linux 系统的人都会遇到某些问题,其中多数问题是对安装过程操作的理解出现了偏差,极少的可能是 Linux 系统本身的缺陷。这一节将分类讲解通常用户会遇到的问题。

2.6.1 启动 Linux 安装启动盘时出现的问题

第一次启动 Linux 的安装启动盘时,用户可能遇到下面几个问题。

· 启动时出现磁盘介质错误

最常见的原因是软盘坏了。用户只需取一张完好的软盘重新制作一张安装启动盘

即可。

- 在启动时或启动后系统出现“死机”现象

  安装启动盘启动后,系统内核将显示一系列信息,指出被 Linux 系统所识别的硬件设备,然后系统显示登录提示符允许用户登录或系统直接进入安装程序。由于启动过程中的某几步较为费时,有时系统从表面上看是死机了,但实际上并没有,用户只需耐心地等待一会儿即可。若用户等上几分钟还不见系统或其他设备的活动,系统才是真正地死机了。启动过程中,下列各步是比较费时的:

  - 从软盘装入 Linux 系统内核。
  - 系统内核启动时可能会花至少 15 秒查找 SCSI 设备。这时系统通常会显示:lp- init: lp1 exists(0), using polling driver。
  - 系统启动后,用户使用 root 或 install 登录后,系统会花 20 秒左右从软盘读入安装程序或 Linux 命令解释器。

  系统真正死机的原因可能有以下几种:

  - 系统没有足够多的内存来启动 Linux 系统。
  - 系统中的硬件与 Linux 不兼容,或硬件是兼容的但没有配置正确。

  对由于硬件的原因出现的死机问题将在“2.6.2 与硬件有关的问题”中解答,这里介绍由于内存太小或其他的原因导至死机的处理办法。

- 启动时系统报告内存不够(“out of memory”)

  如果用户的内存为 4M 或更小时,很可能会出现此现象,有时还会导致系统死机。Linux 系统本身至少需要 2M 内存来启动,有些版本则要求至少 4M 内存。有些 Linux 版本的安装启动盘会在内存中建立一个至少 1M 的虚拟盘并在上面建立文件系统及读入启动程序。因此用户可尝试使系统在启动时不安装虚拟盘。Slackware 发行版本的用户可以在启动安装盘 A1 后系统出现 LILO 提示符时输入“floppy”即可。有时系统在内存不够时并不提示“out of memory”,而是出现死机或启动失败。此时用户应判断是否是本节前面所述的硬件上的原因。若不是,就应尝试不安装虚拟盘。

- 系统启动时报告“permission denied”或“file not found”

  这主要是因为软盘上有坏的区域,用户应换一张好的软盘重新建立安装启动盘再试。

- 系统启动时报告“VFS: Unable to mount root”

  这表明系统找不到根文件系统。若用户从软盘安装,则可能时软盘坏了;若是从 CD-ROM 上安装,则可能是 CD-ROM 坏了或系统没有找到用户的 CD-ROM 驱动器。用户可参考“2.6.2 与硬件有关的问题”中的内容。

2.6.2 与硬件有关的问题

硬件的不兼容性是安装 Linux 系统时最常见的问题。即使是某些 Linux 系统支持的硬件,若配置不对或与其他硬件发生冲突,也会使系统启动上死机或识别不出该硬件设备。

1．确定硬件故障的原因

计算机系统中有许多硬件设备，因此硬件出现故障时确定故障的位置是很重要的。最简单的方法是逐一去掉系统中不必要的硬件，确定到底是哪个硬件设备与 Linux 系统配合不好。除了主机外，软盘控制器(可能在多功能卡上)和软盘驱动器，显示控制器和显示器及键盘是系统所必需的，而其他诸如硬盘驱动器、光盘驱动器、鼠标器、网络适配卡及声音卡等则是可以加上或去掉的硬件设备。

若去掉某一硬件后系统可正常启动，而重新安装上该硬件后系统不能正常启动，那么该硬件就是问题所在了。硬件故障原因一般有两类：

(1) Linux 系统不支持该硬件。查找 Linux 系统支持的硬件的列表就可得到答案。

(2) 该硬件使用的系统资源，如 I/O 地址空间，中断请求号，DMA 通道等需硬件独占的资源，与系统中已有的硬件发生冲突。通常这种冲突可通过查看该硬件的使用手册，并根据手册调整该硬件的一些跳线开关来改变其使用的系统资源，从而避免资源的冲突。每个硬件设备的手册都应说明该硬件使用的系统资源以及如何改变使用的资源号。有时 Linux 安装盘的系统内核预先设置了一些设备的中断号，比如预置 TMC-950 SCSI 控制器，Mitsumi CD-ROM 控制器和总线式鼠标器均使用 5 号中断。这时，用户如果同时使用两个或多个这类设备就必须先安装其中一个设备，然后在安装好 Linux 系统后再修改并重新编译 Linux 系统的内核以改变冲突设备的预置中断号。第 4 章对此有详细介绍。表 2.7 中给出了一些常用设备对系统资源的占用情况。

表 2.7　常用设备占用的系统资源

设备名	I/O 地址	中断请求号	DMA 通道号
ttyS0(COM1)	3f8	4	不用
ttyS1(COM2)	2f8	3	不用
ttyS2(COM3)	3e8	4	不用
ttyS3(COM4)	2e8	3	不用
lp0(LPT1)	378-37f	7	不用
lp1(LPT2)	278-27f	5	不用
fd0, fd1(软盘 1 和 2)	3f0-3f7	6	2
fd2, fd3(软盘 3 和 4)	370-377	10	3

2．识别硬盘驱动器和控制器的问题

Linux 系统启动时，系统应显示一系列信息表明系统内核检测到的设备，如：

```
Console: colour EGA+ 80x25, 8 virtual consoles
Serial driver version 3.96 with no serial options enabled
tty00 at 0x03f8 (irq = 4) is a 16450
tty03 at 0x02e8 (irq = 3) is a 16550A
lp- init: lp1 exists (0), using polling driver
...
```

在某一时刻，系统会显示：

```
Partition check:
```

hda: hda1 hda2

hdb: hdb1 hdb2 hdb3

来表明系统所检测到的硬盘分区。

但有时 Linux 系统并没有检测到硬盘分区,也就使用不了这些分区。这种情况的出现大致有以下几种原因:

- Linux 系统不支持该硬盘驱动器或硬盘控制器
- 硬盘驱动器或控制器没有设置正确

  这种情况也常发生于系统带有两个 IDE 硬盘驱动器时。用户应将两个驱动器上的跳线一个设置成主硬盘(master)方式并将另一个设置成从硬盘(slave)方式。这种情况也常发生于 SCSI 控制器。用户需要参考该硬件手册对其进行重新配置。通常若 MS-DOS 等其他操作系统能正确识别系统的硬盘分区时,故障就不出在硬盘设备上了。
- 硬盘控制器配置正确但仍没有被系统识别

  有些 SCSI 控制器需要用户在启动时向系统提交该设备的信息。下面‘3.SCSI 控制器和设备的问题’中节会有详细的解释。
- 硬盘的格式参数(如柱面数、头数及扇区数)没有被系统所识别

  通常 Linux 系统从 CMOS 中读出这些信息。但有些系统的 CMOS 中没有存放这些信息,这时 Linux 系统需要从以下几种渠道之一取得这些信息。

(1) 在启动安装盘时指定硬盘的格式参数。具体格式如下:

boot: linux hd= < 柱面数> , < 头数> , < 每道扇区数>

(2) 在安装 LILO 时指定这些参数。这样,用户就不必在每次启动时都要输入这些参数了。第四章对 LILO 的安装有进一步的介绍。

3. SCSI 控制器和设备的问题

这里列出了 CD-ROM,硬盘和磁带等 SCSI 控制器和设备的常见问题。如果 Linux 系统不能识别你的控制器或设备,请参阅下面的有关出错信息。

- 在所有的可能的设备号上都查出 SCSI 设备: 这是由于设备和控制器使用了同样的地址造成的。需要通过跳线设置使设备和控制器本身使用不同的地址。
- 虽然设备肯定是完好的但是 Linux 系统却报告错误: 这种问题可能是由于坏的电缆或终端造成的。如果 SCSI 总线没有两端都终结,访问 SCSI 设备就可能有问题。当有任何疑问时,不妨先检查一下电缆。
- SCSI 设备报告超时错误(timeout error): 这种出错通常是由于 IRQ,DMA 或设备地址冲突造成的。也要检查控制器上的中断是否设置正确。
- 使用 BIOS 的 SCSI 控制器未找到: 如果 BIOS 取消了,或者核心不能识别控制器的信号,使用 BIOS 去检测控制器会失败。参见 Linux SCSI HOWTO 中有关的更多信息。
- 使用存储映射的 I/O 端口的控制器不工作: 当存储映射的 I/O 端口没有被正确缓存时会导致这种问题。在 XCMOS 设置中将卡地址标记为费缓存的,或者取消所有的缓存可以解决这种问题。

- 在对磁盘分区时如果得到如下信息“cylinders > 1024”，或者用户无法从一个号数 1023 的磁盘分区上启动：因为 BIOS 中柱面(cylinders)的限制为 1024，任何使用超过此柱面数的分区将都不能被 BIOS 访问。就 Linux 而言，这种情况只会影响到系统的启动；一旦系统启动以后，就可以访问这些分区。你可以选择或者用启动软盘启动 Linux，或者从使用柱面数小于 1024 的分区中启动。2.3.7 部分中有有关创建启动软盘和安装 LILO 的有关信息。
- CD-ROM 和其它可装卸的设备在系统启动时没有被识别：可以尝试启动时在驱动器中放入 CD-ROM(或磁盘)。对有些设备来说，这是必须的。
- 如果系统没有识别到 SCSI 控制器，就需要在系统启动是进行强制的硬件检测。对于没有 BIOS(BIOS-less)的 SCSI 控制器来说这一点尤为重要。Linux 决大多数版本允许在启动安装介质时指定控制器的 IRQ 和共享内存地址。例如，如果你使用的是 TMC-8xx 控制器，可以 LILO 启动显示时输入：

  boot: linux tmx8xx= < interrupt> , < memory-address>

  这里< interrupt> 为控制器的 IRQ，< memory-address> 为共享内存地址。是否可以进行这种操作依赖于你所使用的 Linux 版本；可以参阅其文档获取详细信息。

除了这里列出的外，Linux SCSI HOWTO(参见附录 A)中包含了大多数 SCSI 设备的有用信息。

2.6.3 安装 Linux 系统软件时出现的问题

通常，Linux 系统软件的安装会很顺利。多数问题的原因是用户的安装盘坏了或用户没有足够的硬盘空间。以下是几种常见的现象及解释。

- 系统报告“Read error”(读错误)

  这表明用户的安装盘有问题。若从软盘安装，用户应保证除了安装启动盘是 Linux 格式外，其他软盘都是 MS-DOS 格式。其他格式的软盘在安装 Linux 系统时都不能使用。若软盘的格式是正确的，那么一定是出现读错误的软盘坏了。用户只需重新制作一张相应的好的软盘就可以重新开始安装了。同样，用户若使用 CD-ROM 安装 Linux 系统，就应保证 CD-ROM 的数据面没有损伤。
- 系统报告“tar: read error”或“gzip: not in gzip format”

  这通常是由于磁盘上的文件不对。有可能是磁盘本身坏了，也有可能是文件下载时使用的是文本(ASCII)方式而不是二进制(Binary)方式。用户需要重新制作或下载相应的软盘或文件。
- 系统报告“Device full”(设备满)

  这是由于安装时硬盘空间不够了。解决的方法是重新建立文件系统并选择更少量的软件进行安装。
- 系统报告“read- intr: 0x10”

  这通常表明硬盘上有坏区域。但若该信息出现在 mkswap 或 mke2fs 命令执行过程中，则表明是系统硬件的故障，或是 CMOS 中的硬盘格式参数不对或用户在系

统启动时输入的硬盘格式参数不对。用户应纠正这样的错误设置。

- 系统报告“file not found”(文件找不到)
  这表明安装盘上缺少某些需要的文件。用户应检查用 FTP 下载安装盘上的文件时是否使用了命令“mget * .* ”。这条命令只下载文件名中带有“.”的文件,而不是所有的文件。正确的命令应是“mget * ”。

2.6.4 Linux 安装后系统启动遇到的问题

“2.6.1 启动 Linux 安装启动盘时出现的问题”中介绍的一些有关安装盘启动时可能遇到的问题及解决方法通常也适用于这里。另外用户还可能遇到以下几种情况。

1. 从软盘启动 Linux 系统。如果用户从软盘启动 Linux 系统,尤其是使用安装启动盘而不是安装程序制作的 Linux 系统启动盘时,系统可能提示用户输入 Linux 根文件系统所在的分区。此时,用户可以在提示符“boot:”后键入“linux hd= < 分区设备名> ”。

2. 从硬盘启动 Linux 系统。如果用户在硬盘上安装了 LILO,就可以从硬盘直接启动系统了。但有时由于 LILO 没有配置正确,即使硬盘上各操作系统都是正确的,系统也不能正常启动。通常只需使用软盘启动系统后重新配置 LILO。第 4 章对 LILO 的安装有详细的介绍。

3. 登录 Linux 系统。通常用户使用“root”用户名就可以登录进系统了。如果系统提示输入密码,用户可同样输入“root”。如果这时还不能登录进系统,用户应参考安装手册,登录时应该使用的用户名和密码可能就在上面。

第 3 章　Linux 使用入门

本章并不涉及大量的细节,也不涉及其他高级论题,只是使 Linux 的新用户掌握最基本的知识。由于 Linux 是 UNIX 的一种实现,它的用法中有 95% 与其他 UNIX 系统的用法完全相同。对于熟悉 UNIX 的读者来说,本章的内容也能帮助他了解其他 UNIX 系统和 Linux 的一些区别。

3.1　Linux 的基本概念

Linux 是一个多任务、多用户的操作系统。也就是说,在 Linux 系统中同时可以有多个用户使用同一台计算机,运行多个不同的应用。这与 MS-DOS 是不同的。在 MS-DOS 中,同一时间只能有一个用户使用 MS-DOS。在 Linux 系统上,用户为了使系统能够识别自己,必须进行登录。登录分为两步:输入用户的登录名,系统将根据该登录名来识别用户;输入用户的口令,该口令是用户自己选择的一个字符串,对其他用户是保密的,在登录时系统用来辨别真假用户的关键字。系统认为只有用户自己和用户授权的人知道这个口令,所以只允许输入正确口令的用户登录,其他用户则不能登录到这个帐户下。

在传统的 UNIX 系统中,系统管理员会在为用户建立新帐户时赋给用户一个用户名和一个初始的口令。在下面的讨论中,我们使用一个假想的用户名“zhang”。

另外,每个 UNIX 系统都被赋予一个主机名(hostname)。这个主机名是赋给计算机的名字、标识。主机名用来在网络上识别独立的计算机,但是,即使用户的计算机没有被联网,它也应该有一个主机名。在“4.10.2 设置主机名”中,我们会谈到为系统设置主机名的问题。在下面的例子中,系统的主机名假设是“mousehouse”。

3.1.1　建立普通用户帐户

用户在使用系统之前,一定要请系统管理员为自己建立一个普通用户帐户。即使用户自己就是系统管理员,也应该为自己建立一个普通用户帐户,这是因为使用如前一章提到的特殊帐户,诸如 root 或 install 等帐户进行一般的操作和使用是不合适的。root 帐户一般用来运行特权命令,维护系统,有关信息将在“4.1 普通用户和超级用户”一节中讨论。

系统管理员要登录到 root 帐户,使用 useradd 或 adduser 命令给用户建立一个帐户。参看“4.4 管理用户”一节中有关该过程的描述。

3.1.2　登录

用户的普通用户帐户建立之后,就可以进行登录了。在登录时,用户会在屏幕上看到类似下面的提示:

```
mousehouse login:
```

这时,输入用户名,然后键入回车键。例如用户名为 zhang 的用户输入“zhang”及回车键。此时,用户会在屏幕上看到输入口令的提示:

```
mousehouse login: zhang
Password:
```

这时,需要输入口令。键入时,口令不会在屏幕上显示出来,所以要小心键入。如果用户输入了错误的口令后,就会在屏幕上看到下列信息:

```
Login incorrect
```

这时需要重新输入。

当用户正确地输入用户名和口令后,就会合法地进入系统,畅游其间。

3.1.3 虚拟控制台

系统的控制台是与系统直接相联的监视器和键盘。由于 Linux 是一个多用户的操作系统,用户可以通过系统的串行端口联接其他的终端,但这些不是控制台。Linux 和许多版本的 UNIX 系统一样,提供了虚拟控制台(VC)的访问方式,这就允许用户在同一时间从控制台进行多次登录。

用户可以登录到自己的系统来证明这一点。登录后,按一下 Alt-F2 键。这时用户又可以看到“login:”提示符。用户看到的是第二个虚拟控制台。这时只需按一下 Alt-F1 键,就可以回到第一个虚拟控制台。

一个新安装的 Linux 系统可能允许用户使用 Alt-F1 到 Alt-F4 键来访问前四个虚拟控制台。有些系统可以启用 12 个虚拟控制台——每一个虚拟控制台对应用户键盘上的一个功能键。正如用户所看到的,虚拟控制台的使用会使系统更加强大——用户可以同时在多个控制台上工作。

尽管虚拟控制台的使用仍有一些局限性,如用户一次只能看到一个虚拟控制台,但它可以使用户感受到 Linux 系统的多用户、多任务的能力。用户可以在某一虚拟控制台上进行的工作尚未结束时,切换到另一虚拟控制台开始另一项工作。

3.1.4 Shell 和命令

用户使用 Linux 系统时,大多数时间是在通过 shell 同系统进行对话。shell 是一个接收用户输入(如用户键入的命令)并将其转换成指令的程序。可以把它比作 MS-DOS 中的 COMMAND.COM。shell 是用户和 Linux 系统之间的界面。Linux 还有许多其他界面,如 X 窗口系统。它允许用户同时使用鼠标和键盘进行命令操作。

用户登录之后,系统就启动一个 shell,这时用户就可以开始键入命令。下面是一个例子。zhang 登录后,便可以看到 shell 的提示符。

```
mousehouse login: zhang
Password: zhang's password
Welcome to Mousehouse!

/home/zhang $
```

“/home/zhang$”是 shell 的提示符,它表明 shell 已经做好了接收命令的准备(该提示符自身所表明的意思将在后面讲解)。现在我们来让系统作一些有趣的事情:

```
/home/zhang$ make work
make: *** No way to make target ‘work’. Stop.
/home/zhang$
```

make 是系统中一个命令的名字,后面跟着的是命令的参数。在接收到这个命令后,shell 便执行它。由于输入的命令的参数不正确,系统返回信息后停止该命令的执行。这个例子有趣的是从返回信息看来,系统回答用户时的“口吻”并不是很友好。

这个例子给我们提出了一个问题:命令是什么?当用户键入 make work 之后,发生了什么事情?命令行的第一个字“make”是命令的名字。命令行的其他字是这个命令的参数。例如:

```
/home/zhang$ cp foo bar
```

这里,命令的名字是“cp”,参数是“foo”和“bar”。

当用户键入命令后,shell 做了几件事情。首先,它要检查该命令是不是 shell 的内部命令。也就是说,该命令是否由 shell 自身执行。Linux 中有许多这样的命令,我们将在以后讲述。shell 还要检查该命令是否是其他命令的别名或替代名。如果都不是,shell 就会在硬盘上查找以该命令名为名字的程序。如果找到了这样的程序,shell 就会将命令行的参数传递给它,并执行它。

在例子中,shell 会寻找名为 make 的程序,并以 work 为参数执行它。make 是一个经常被用来编译大程序的程序,它以参数作为目标来进行编译。在“make work”中,make 编译的目标是 work。因为 make 找不到以 work 为名字的目标,它便给出错误信息以表示运行失败,用户又回到系统提示符下。

用户键入有关命令行后,如果 shell 找不到以其中的命令名为名字的程序,shell 就会给出错误信息。例如,如果用户键入:

```
/home/zhang$ eat dirt
eat: command not found
/home/zhang$
```

可以看到,用户得到了一个没有找到该命令的错误信息。当用户敲错命令后,系统都会给出这样的错误信息。

3.1.5 退出系统

在我们对 Linux 系统进行进一步阐述之前,我们先介绍一下如何退出系统。

在 shell 提示符下,键入下列命令退出系统:

```
/home/zhang$ exit
```

还有其他退出系统的方法,但上面一种是最安全的。

3.1.6 改变口令

为了使用户帐户的安全得到更好的保护,用户还应该知道如何修改口令。修改口令的

命令是 passwd, 它将提示用户输入旧口令和新口令, 之后还要求用户再次确认新口令, 以提醒用户不要忘记自己的口令-如果忘记了, 用户可以请系统管理员为自己重新设置一个。(有关系统管理员的操作, 请参阅第 4 章。)

3.1.7 文件和目录

在大多数操作系统中, 包括 Linux, 都有文件的概念。文件是命了名(称为文件名)的一组信息。例如用户的一些文章、电子邮件信息、源程序和一些可以执行的代码, 都是以文件的方式保存的。

文件以文件名作标识。例如, 用户的历史方面的文章可以用 history-paper 作为文件名。一般来说, 用户所选择的文件名都是有意义的, 它能够体现出文件的内容。与 MS-DOS 以及其他一些操作系统不同的是, 在 Linux 中, 文件名没有标准格式。通常, Linux 的文件名中可以包含任意字符(“/”字符除外, 且“～”字符不能作为文件名的第一个字符——请参阅‘3.1.10 引用主目录’), 文件名的长度限制在 256 个字符以内。

目录是一组文件的集合。它可以被理解成存放许多文件的‘文件夹’。目录也有一个用作标识的目录名。另外, 目录被组织在一个树状结构中; 也就是说, 目录中还可以包含其他目录。

文件可以通过路径名来引用。路径名由文件名及其前面包含该文件的目录名组成。例如, 用户 zhang 有一个名为 papers 的目录, 该目录中有三个文件: history-paper、math-paper 和 computer-paper。这三个文件保存了用户 zhang 的三篇文章。用户 zhang 可以用路径名来引用文件 math-paper:

```
papers/math-paper
```

可以看到, 目录名和文件名之间用一个“/”分开。正是因为这个原因, 文件名自身不能包含“/”。

目录是可以相互嵌套的。例如, 我们假设用户 zhang 在 papers 目录中还有一个名为 art 的目录, 我们称它为 papers 目录的子目录。在这个子目录中包含有两个文件: chap1 和 chap2。文件 chap1 的路径名就是:

```
papers/art/chap1
```

可见, 路径名实际上就是用户查找一个特定文件的‘路径’。子目录上一层的目录称为父目录。在这里, papers 目录就是 art 目录的父目录。

3.1.8 树型目录

为使系统资源及程序容易查找, Linux 对文件进行了合理的布局。这个布局就是从根目录“/”开始的树型目录。根目录下有一些非常重要的子目录, 如/bin、/etc、/dev、/usr, 等等。这些目录依次又包含其他的子目录, 子目录中又包含系统的配置文件、程序等。

特别要指出的是, 每个用户都有自己的主目录, 它是在系统管理员为用户建立帐户时确立的, 用户登录后马上就进入自己的主目录。用户可以在他的主目录中保存自己的文件和目录。上述例子中, 用户 zhang 的所有文件及目录(如 papers、history-paper、art 等)都保存在用户 zhang 的主目录中。通常, 用户的主目录在目录/home 下, 以用户自身的用户

名命名。用户 zhang 的主目录为: /home/zhang。

一个典型的 Linux 系统的树型目录结构如图 3.1 所示。

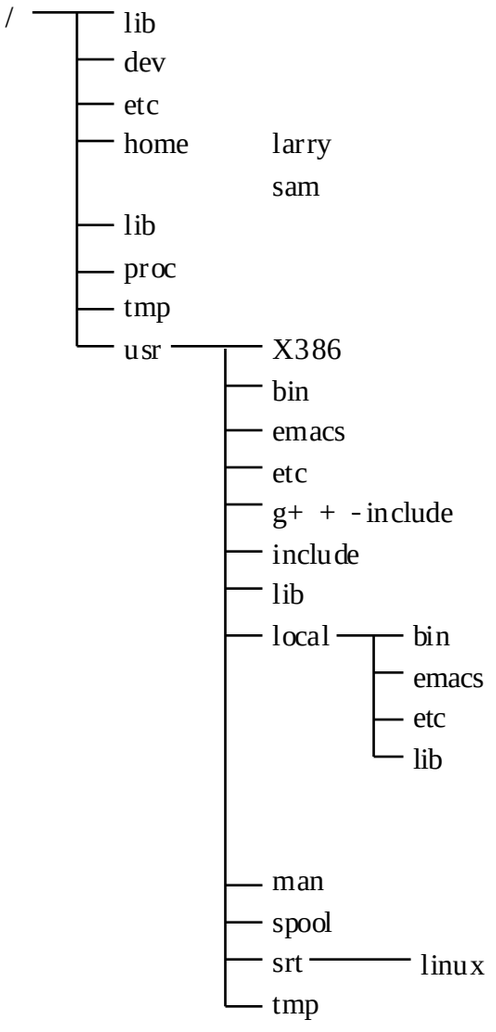


图 3.1　Linux系统的树型目录结构

3.1.9　当前工作目录

使用 Linux 系统时,用户都是在某个目录下向 shell 键入命令的,可以将当前工作目录理解为当前所在的目录。当用户刚刚登录到系统中时,当前工作目录为用户的主目录——在上面的例子中是/home/zhang。当用户访问一个文件时,可以根据该文件与当前工作目录的相对位置来引用它,而不需要列出这个文件的完整的路径名。

下面举一个例子。用户 zhang 的 papers 目录中有一个名为 history-paper 的文件。如果用户 zhang 想显示这个文件,可以使用下列命令:

/home/zhang $ cat　/home/zhang/papers/history-paper

cat 命令的功能是在屏幕上显示文件内容。用户也可以根据文件 history-paper 与当前工作目录的相对位置来引用该文件。这时,命令为:

/home/zhang $ cat papers/history-paper

如果用户以“/”以外的字符(如 papers/history-paper)作为文件引用的第一个字母,系统就认为用户相对于当前工作目录来引用文件。这就是相对路径名。

如果用户以“/”字符(如/home/zhang/papers/history-paper)作为文件引用的第一个

字母,系统就将它解释成全路径——这就是说,路径名包括了从根目录到这个文件的完整的路径。这就是绝对路径名,或称为全路径名。

3.1.10 引用主目录

用户可以通过一个波浪字符"~"来引用自己的主目录。例如,命令

```
/home/zhang$ cat ~/papers/history-paper
```

和下面的命令

```
/home/zhang$ cat /home/zhang/papers/history-paper
```

的意思是相同的。shell 将用用户主目录名来替换"~"字符。

另外,用户也可以用"~"来引用其他用户的主目录。如果 /home/wang 是用户 wang 的主目录,shell 会自动把路径名~wang/letters 解释成 /home/wang/letters。)。"~"只是引用主目录的一种方法。实际上系统中并没有以"~"命名的目录——它只是 shell 提供的一个小窍门。

3.2 Linux 系统使用初步

我们开始介绍本节内容之前,要请用户注意:在 Linux 系统中,文件和命令的名字是区分大小写的。这与 MS-DOS 等操作系统是不同的。例如,make 与 Make 或 MAKE 是完全不同的命令。文件名和目录名也是如此。

3.2.1 遍历目录

到目前为止,我们已经知道如何进行登录和退出了。但如何使用路径名来引用文件呢? 又如何改变我们的当前工作目录呢?

在目录结构中进行遍历的命令是 cd,它是英文的"改变目录"(change directory)的缩写。在 Linux 系统中,常用的命令都是两个到三个字符的。cd 命令的用法是:

```
cd <目录名>
```

在这里,<目录名> 是用户希望到达的目录的名字。

如前面所说的,当用户登录后,是从自己的主目录开始工作的。如果用户 zhang 想进入到 papers 子目录,他需要使用下列命令:

```
/home/zhang$ cd papers
/home/zhang/papers$
```

这时,用户 zhang 的当前目录可以从提示符中反映出来。这时,用户 zhang 的当前工作目录是 papers,他可以使用下列命令来看他的 history-paper:

```
/home/zhang/papers$ cat history-paper
```

用户 zhang 可以使用下列命令返回父目录:

```
/home/zhang/papers$ cd ..
/home/zhang$
```

每个目录中都有一个以".."(注意,"cd"和".."之间有一个空格)为名字的入口来表

示该目录的父目录。同样,每个目录中都有一个以“.”为名字的入口表示该目录自身。所以,下面的命令:

/home/zhang/papers$ cd .

一般不用。

用户也可以在 cd 命令中使用全路径。若 zhang 要想进入 wang 的主目录,可以使用下面的命令:

/home/zhang/papers$ cd /home/wang

/home/wang$

另外,用户也可以使用不带参数的 cd 命令返回自己的主目录。

/home/wang$ cd

/home/zhang$

3.2.2 查看目录中的内容

下面介绍一个新命令:ls。ls 命令能打印出当前目录下所有文件和目录的清单。类似于 MS-DOS 下的 dir 命令。例如:

/home/zhang$ ls

Mail

letters

papers

/home/zhang$

我们可以看到,在用户 zhang 的当前目录下,有三个名字:Mail、letters 和 papers。但它们是目录还是文件呢? ls 命令的 -F 选项可以告诉我们更多的信息。例如,

/home/zhang$ ls -F

Mail/

letters/

papers/

/home/zhang$

上面名字带有“/”后缀的表示它是一个子目录。我们可以看到,这三个都是子目录名。

使用“ls -F”列出的文件名也可能用“*”作为后缀,这表示该文件是一个可执行的,或者说是一个可以被运行的代码;如果被“ls -F”列出的文件名没有任何后缀,则该文件只是一个普通的文件,既不是目录,也不是可执行的程序。

一般来说,Linux 命令除参数外,还可以有许多选项。选项一般都以“-”开始,如命令“ls -F”。-F 选项使 ls 命令给出更多的有关文件类型的信息。

如果在 ls 命令中以目录名为参数,那么它将打印出指定目录中的内容。例如,

/home/zhang$ ls -F papers

computer-paper

history-paper

math-paper

art/

/home/zhang$

上面的例子中,ls-F paper 命令给出了 papers 目录下的文件和子目录的清单。

下面,我们给出 /etc 目录下的内容:

/home/zhang$ ls /etc

Images	ftpusers	lpc	rc.new	shells
adm	getty	magic	rc0.d	startcons
bcheckrc	gettydefs	motd	rc1.d	swapoff
brc	group	mount	rc2.d	swapon
brc~	inet	mtab	rc3.d	syslog.conf
csh.cshrc	init	mtools	rc4.d	syslog.pid
csh.login	init.d	pac	rc5.d	syslogd.reload
default	initrunlvl	passwd	rmt	termcap
disktab	inittab	printcap	rpc	umount
fdprm	inittab.old	profile	rpcinfo	update
fstab	issue	psdatabase	securetty	utmp
ftpaccess	lilo	rc	services	wtmp

/home/zhang$

其中有些文件是大家感兴趣的系统的可执行程序。

与 MS-DOS 不同,Linux 的文件名可以长于 8 个字符,并且在文件名中的任何位置都可以出现句点,并且在一个文件名中可以出现多个句点。

可以使用"cd . ."命令回到当前目录的父目录上,然后再进入其他目录:

/home/zhang$ cd . .

/home$ cd . .

/$ cd usr

/usr$ cd bin

/usr/bin$

也可以使用 cd 直接进入要去的目录,如使用"cd /usr/bin"。

请用户使用 ls 和 cd 尝试遍历各个目录来熟悉 Linux 系统。在进行遍历时,用户可能会遇到"Permission denied"的错误信息,这只是 Linux 系统的一种保护机制。当用户访问一个目录时,他必须具有对该目录的访问权限,我们将在"3.7 文件权限"一节中讨论。

3.2.3 建立新目录

用户可以使用 mkdir 命令来为自己建立一个新目录。mkdir 的意思是"make directory",即"建立目录"。

mkdir <目录名>

下面举一个例子进行说明。例如,

/home/zhang$ mkdir reports

```
/home/zhang$ ls -F
Mail/
letters/
papers/
reports/
/home/zhang$ cd reports
/home/zhang/reports$
```

mkdir 命令可以有多个参数。其参数可以是全路径名,也可以是相对路径名。上例中的 reports 是相对路径名。请看下面的例子:

```
/home/zhang/reports$ mkdir /home/zhang/reports/chap1 ~/reports/chap2
/home/zhang/reports$ ls -F
chap1/ chap2/
/home/zhang/reports$
```

该例子中有两个参数,建立了两个目录。一个用了全路径名,另一个用的是相对路径名。

3.2.4 复制文件

cp 命令可以用来复制文件,它有两个以上的参数。cp 命令的最后一个参数既可以是一个文件名,也可以是目录名。当这个参数是文件名时,cp 命令行只能有两个参数,第一个参数指定要被复制的文件。这时,cp 命令的功能是将第一个参数指定的文件复制到第二个参数指定的文件中。当最后一个参数是目录名时,cp 命令行中该目录名前可以有一个以上的参数,分别指定要被复制的文件。这时,cp 命令的功能是将指定的文件复制到指定目录下,并保持文件名不变。请看下面的例子:

```
/home/zhang/reports$ cp /etc/rc .
/home/zhang/reports$ cp /etc/shells .
/home/zhang/reports$ ls -F
chap1/ chap2/ rc shells
/home/zhang/reports$ cp /etc/rc rctmp
/home/zhang/reports$ ls -F
chap1/ chap2/ rc rctmp shells
/home/zhang/reports$
```

3.2.5 移动文件

mv 命令用来把文件从一个地方移到另一个地方。同 cp 命令一样,mv 命令可以有两个以上的参数。最后一个参数既可以是一个文件名,也可以是目录名。当这个参数是文件名时,mv 的命令行只能有两个参数,第一个参数代表要被移动的文件。这时,mv 命令的功能是将第一个文件移动到第二个文件中,也就是文件的更名。当最后一个参数是目录名时,mv 的命令行中该目录名前可以有一个以上的参数,分别指定要被移动的文件。这时,

mv 命令的功能是将指定的文件移动到指定目录下,并保持文件名不变。请看下面的例子:

```
/home/zhang/reports$ mv  rctmp  rcnew
/home/zhang/reports$ ls  -F
chap1/   chap2/   rc   rcnew   shells
/home/zhang/reports$ mv  rcnew  chap1
/home/zhang/reports$ls  -F
chap1/   chap2/   rc   shells
/home/zhang/reports$ cd chap1
/home/zhang/reports/chap1$ ls  -F
rcnew
/home/zhang/reports/chap1$
```

注意:当目的文件已经存在时,通常 mv 和 cp 命令会将其改写,并不对用户进行任何提示。这有时会造成用户信息的丢失。所以,当用户将一个文件移到另一个目录时,要小心:该目录中可能已经存在同名文件,而你的操作会将其改写!用户可以通过在命令行中增加相应的选项(-i)来使系统在覆盖相应文件前等待用户确认。

3.2.6 删除文件和目录

下面我们来学习怎样删除一个已经存在的文件或目录。

rm 命令用来删除一个已经存在的文件。rm 的意思是"remove"即删除。如下面的例子:

```
/home/zhang/reports$ rm rc frog
rm: frog: No such file or directory
/home/zhang/reports$ ls  -F
chap1/   chap2/   shells
/home/zhang/reports$
```

正如用户所看到的,rm 命令并不十分友好。它在不需要我们作任何确认的情况下,就进行删除操作(可以通过命令选项-i 来改变这一点),并且,即使在命令行出现错误的情况下,它也毫不迟疑地执行删除操作,这有时确实很危险。所以,用户删除文件时,在敲入回车键之前一定要仔细检查命令行!

与 rm 命令相关的一个命令是 rmdir,它是用来删除一个空目录。当被删除的目录中包含文件或子目录时,它会给出错误信息。

3.2.7 浏览文件

more 和 cat 命令用来浏览文件内容。more 在显示文件内容时,每次只显示一屏;而 cat 命令则是一次显示所有内容。

可以用下列命令浏览文件 shells 的内容:

```
/home/zhang/reports$ more  shells
```

使用 more 命令进行浏览时,按空格键显示下一页,例如,按 b 键显示前一页。在 more 中还有其他的命令 ,如 s、h、f、n、v、f 等,但上述的空格键和 b 键是基本操作。按 q 键会退出 more。

使用 cat 命令(如 cat shells),文本会很快滚过屏幕,用户根本来不及阅读。“cat”的意思是“concatanate”(连接),这才是这个命令的真正用途。cat 命令可以用来把几个文件的内容连接起来,并把结果存入另一个文件。关于 cat 命令的用法在以后会讲到。

3.2.8 获取联机帮助

几乎所有的 UNIX 系统,包括 Linux,都提供了联机手册。这些联机手册包括各种系统命令、资源、配置文件等的在线文档。

访问联机手册的命令是 man。例如,用户若想找到有关 ls 命令的其他选项的信息,可以使用下列命令:

```
/home/zhang $ man ls
```

这时,有关 ls 命令的所有信息便会显示在屏幕上。

man 命令还有关键字功能。例如,如果用户对有关 Postscript(一种打印机控制语言)的命令感兴趣,可以键入 man -k ps 或 man -k Postscript,屏幕便显示出 Linux 中所有包含单词“ps”或“Postscript”的命令、系统调用以及其他文档信息。这一功能在用户寻找有关不知道其名字的工具时,是非常有用的。

联机手册除了会告诉用户命令的语法外,还会告诉用户有关该命令的其他大量信息。建议用户查看一下我们已经讲过的命令的联机手册。当我们在以后的章节中介绍新命令时,建议用户也去查看它们的联机手册。

由于 Linux 提供的联机手册并不完全,有些命令的联机手册还不能被查到。必要时用户可以参考其他关于 UNIX 的书或手册。

3.3 Linux 的文件系统

Linux 的文件系统是文件和目录层次的集合。

在 3.1 节中,已经介绍了文件、目录以及树型目录的概念,并在图 3.1 中给出了 Linux 系统的树型目录结构。正是这些文件、目录按照树型结构组织在一起,构成了 Linux 的文件系统。

Linux 系统中的文件除了 3.1 节中的所述的普通文件外,还包括一些特别文件,如设备文件、管道文件等。Linux 系统能象存取普通文件那样存取特别文件,以实现对 I/O 设备、管道等的访问。

下面对 Linux 的文件系统中各目录的内容及功能进行介绍。了解文件系统的结构将有助于用户更好地理解和使用 Linux。

在 Linux 系统的根目录下,有如下目录:bin、dev、etc、home、install、lib、mnt、proc、root、tmp、user、usr 和 var。

(注:对不同的 Linux 版本,这些目录会略有所不同。这里只是作为一个例子。)

下面,我们来介绍这些目录:

/bin　bin 是“binaries”(二进制)的简称,也就是二进制代码或可执行代码的意思。这里是存放许多基本的系统程序的地方。用户可以用“ls -F /bin”命令来列出其中的文件。在所列出的文件中,有许多是用户已经熟悉的命令,如 cp、ls 和 mv。所列出的文件正是这些命令对应的可执行代码。当用户使用 cp 命令时,shell 所执行的程序就是“/bin/cp”。

/dev　dev 是“device”(设备)的简称。/dev 目录中存放的文件是设备驱动程序——这些程序被用来访问系统设备和资源,如磁盘设备(disk drive)、调制解调器(modem)、内存(memory)等。例如,用户可以通过访问/dev/mouse 来访问鼠标(mouse)输入,就像从文件中读取数据一样。

以 fd 开头的文件名代表软盘设备。fd0 代表第一个软盘驱动器,fd1 代表第二个。聪明的用户会注意到该目录下除这两个外还有其他的软盘驱动器。它们代表特殊类型的软盘。如通过 fd1H1440 能访问驱动器 1 中的高密度的、3.5 英寸的软盘。

下面我们列出一些经常会用到的设备文件。注意,即使用户没有下面列出的某些设备,/dev 中也为你提供了访问这些设备的接口。

- /dev/console 指系统控制台,也就是直接连接到系统的监视器。
- 各种不同的/dev/ttys 和/dev/cua 设备用来存取串行端口。例如,/dev/ttys0 相当于 MS-DOS 下的“COM1”。/dev/cua 用来连接调制解调器。
- 以“hd”开头的设备名代表硬盘设备。/dev/hda 代表整个第一个硬盘。/dev/hda1 代表/dev/hda 的第一个分区。
- 以“sd”开头的设备名代表 SCSI 设备。如果用户有 SCSI 的硬盘,要用/dev/sda 而不是/dev/hda 来访问它。SCSI 的磁带要用 st 设备访问,而 SCSI 的 CD-ROM 要通过 sr 设备访问。
- 以“lp”开头的设备名代表并行端口。/dev/lp0 相当于 MS-DOS 中的“LPT1”。
- /dev/null 就像一个黑洞——所有发给这个设备的数据都会永久消失。这个命令也是非常有用的。当用户不想让命令的输出显示在屏幕上时,可以将这些输出送给/dev/null。我们在后面会用到这个设备。
- 以“/dev/tty”开头的设备名代表用户系统上的虚拟控制台(可以通过按 Alt-F1, Alt-F2 等键来访问)。/dev/tty1 代表第一个虚拟控制台,/dev/tty2 代表第二个虚拟控制台,等等。
- 以“/dev/pty”开头的设备名代表“伪终端”。它们用于为远程登录提供“终端”。例如,如果你的计算机是联网的,远程计算机可以通过 telnet 登录到你的计算机上,他们使用的就是/dev/pty 设备。

/etc　/etc 目录中存放了各种系统配置文件。其中包括用户信息文件 /etc/passwd、系统初始化文件 /etc/rc 等。

/sbin　/sbin 用来保存系统管理员使用的基本的系统程序。

/home　/home 中存放了用户的主目录。例如,/home/zhang 是用户“zhang”的主目录。在一个新装的系统中,这个目录中可能还没有用户。

/lib　/lib 中存放的是共享程序库映象。这些文件中包含了被许多程序所共享的代码。在 Linux 系统中,被许多程序共享的子程序保存在一个公共的地方——/lib 中,这样就避免在每个程序中都包含一份相同的子程序的副本。这将使得可执行文件变得更小些,从而节省用户系统上的空间。

/proc　/proc 是一个"虚拟的文件系统",在/proc 中的文件是保存在内存的,而不是保存在磁盘上。它们代表系统中正在运行的各种进程,并且允许用户在任一时刻获取有关正在运行的程序和进程的信息。

/tmp　许多程序在运行中需要产生或保存一些暂时的信息或数据。这些信息或数据被存储在/tmp 目录中。

/usr　/usr 是一个很重要的目录。它包含了大量的子目录。这些子目录中依次又包含了一些系统使用的非常重要的有用程序和配置文件。

前面所述的各个目录是保证系统能够运转的基本条件。而 /usr 目录下的许多内容对系统的运转来说则是可选的。然而,这些功能会使用户的系统更加有用和有趣。没有/usr 这个目录,用户会感到系统有些单调无聊,只有一些类似于 cp 和 ls 的命令。/usr 目录中包含了许多大的软件包以及相应的配置文件。

/usr/X386　如果用户安装了 X 窗口系统,这个系统将被保存在/usr/X386 中。X 窗口系统一个功能强大的图形环境,它提供了大量的图形工具和程序。如果用户对 Microsoft Windows 或 Machintosh 环境很熟悉,就不会对 X 窗口系统感到陌生。/usr/X386 目录中包含了 X 窗口系统的所有可执行程序、配置文件和支持文件。

/usr/bin　/usr/bin 是 Linux 系统上的真正的软件仓库。它包含了其他目录如/bin 中没有找到的大多数可执行程序。

/usr/etc　与/etc 一样,/usr/etc 中包含了各种系统程序和配置文件。相比之下,/usr/etc 中包含的文件更多些。一般来说,/etc 中包含的文件对系统来说更基本些。

/usr/include　/usr/include 目录中包含了 C 语言头文件。这些文件大多以.h 结尾,用来描述 C 语言程序中所用到的数据结构名称、子过程和常量。

/usr/g++-include　/usr/g++-include 目录中包含了 c++ 编译器的头文件(与/usr/include 很类似)。

/usr/lib　/usr/lib 目录中包含了与/lib 中文件相对应的"存根"以及静态库。当编译一个程序时,程序会与/usr/lib 中的库连接。当程序需要库中的实际编码时,/usr/lib 会引导程序查找/lib。另外,许多程序还把配置文件存入/usr/lib 中。

/usr/local　/usr/local 目录与/usr 很相似。它包含了各种程序和文件。这些程序和文件对系统来说不是基本的,但却可以使系统功能更强、更生动有趣。一般来说,/usr/local 中的程序是为系统特设的——也就是说,/usr/local 与 Linux 系统本身是无关的。

用户会在这个目录下找到一些大的软件包,如 TEX(文档编排系统)和 Emacs(一个功能强大的编辑器)等(如果用户安装了这些软件)。

/usr/man　联机用户手册就存放在/usr/man 目录中。每个联机手册的"节"都有两个子

目录(用户可以使用 man 命令得到详细信息)。例如,/usr/man/man1 中包含联机手册第一节的源码(也就是没有经过编排的内容);而/usr/man/cat1 中包含第一节已编排的内容。Linux 的联机手册分为九节,各节分别对应的内容如下:用户命令(内部命令)、系统调用、库函数、设备(如 hd、sd 等)、文件格式(如 wtmp、/etc/passwd 等)、游戏(指内部游戏)、宏软件包(如 nroff、ascii)、系统管理和核心程序。

/usr/src　/usr/src 中包含了系统上各种程序的源代码(未经编译的程序)。该目录中,最重要的子目录是/usr/src/linux,其中包含了 Linux 核心程序的源代码。

/var　/var 目录中存放着那些不断在扩充着的目录。这个目录中的有些内容原来存放在/usr 目录中。为了保持/usr 目录的相对稳定,那些经常被修改的目录就移放在/var 中。这些目录包括:

/var/adm　/var/adm 目录中包含了系统管理员感兴趣的各种文件,特别是系统日志(system log)。在这个文件中记录了系统的错误和问题。其他文件用来记录系统的登录信息,包括失败的登录。

/var/spool　/var/spool 目录中包含那些将被"转入"到其他程序中的文件。例如,如果用户的计算机是联网的,他接收到的电子邮件将被存储到/var/spool/mail 直到用户阅读或删除它。发出或收到的新闻会存到/var/spool/news 中,等等。

3.4　shell 的类型

我们在前面曾讲过,Linux 是一个多任务、多用户的操作系统。shell 是用户与 Linux 系统之间的界面。用户是通过 shell 与系统打交道的。在本节中,我们将介绍 Linux 系统中 shell 的特点和类型。

请用户不要把 Linux(操作系统自身)和 shell 相混淆。shell 只是操作系统的一个界面。shell 在 Linux 之上提供了大量的功能。

用户在提示符下敲入的命令是通过 shell 解释执行的。但 shell 不只是用户交互命令的解释器,它同时也是一种功能强大的编程语言,用户可以使用这种语言编写 shell 程序。shell 对写在 shell 程序中的 shell 命令进行批处理。shell 编程语言是一个功能强大的工具。用户利用这种语言可以使得 Linux 系统变得更加方便有效、功能更加强大。请参看 3.13节的有关内容。

UNIX 世界中的 shell 有多种类型。其中最主要的两种是"Bourne shell"和"C shell"。Bourne shell 使用早期 UNIX 系统,如系统 III 上原始 shell 的命令语法。大多数 UNIX 系统上的 Bourne shell 的名称是/bin/sh(这里,"sh"代表"shell")。C shell 采用另一种命令语法,有点类似于 C 程序设计语言。大多数 UNIX 系统上的 C shell 的名称是/bin/csh。

在 Linux 系统中,提供了多种可用的 shell。最常用到的两种是 Bourne Again Shell,或"Bash"(/bin/bash)和 Tcsh(/bin/tcsh)。Bash 是包含了 C shell 中的一些优点的 Bourne shell 的一种形式。因为 Bourne shell 语法的集合只是 Bash 支持的语法的子集,用 Bourne shell 写的任何 shell 程序都能在 Bash 下运行。Linux 为喜欢使用 C shell 语法的人提供了 Tcsh。Tcsh 是 C shell 的一个扩展版本。

用户可以根据自己的兴趣选择 shell 的类型,所选用的 shell 类型将不会影响系统的普通命令,如 cp、ls 等的使用。这些命令的语法都是相同的。只有当用户书写 shell 程序或使用 shell 的一些高级特性时,不同的 shell 类型所引起的不同才会显示出来。

在以后的章节中,当涉及到 shell 的一些特性时,我们会指出 Bourne shell 和 C shell 之间的差别。当然,这些差别大多是很细微的。

3.5 通 配 符

大多数 Linux 命令的参数是文件名。大多数的 UNIX 操作系统,包括 Linux 上的各类型的 shell 的一个主要特点:能够使用特殊字符来同时引用多个文件名。这种特殊的字符称为通配符。shell 根据由普通字符和特殊字符组成的模式来引用文件名。

通配符"*"可用来代表文件名中的任一字符或字符串。例如,当用户在文件名中使用通配符"*"时,shell 将把"*"用指定的目录中的所有文件名中的字符或字符串替代后进行匹配,并将匹配的文件名传给相应命令作为参数。

我们来看一个简单的例子。假设用户 zhang 在当前目录中有 5 个文件:ch1.c、ch2.c、ch3.f、test 和 result。使用下列命令可以列出所有以.c 为后缀的文件名:

```
/home/zhang$ ls *.c
ch1.c    ch2.c
/home/zhang$
```

正如我们所看到的那样,命令"ls *.c"中的通配符"*"被当前目录下的文件名中所有的字符或字符串替代后进行匹配,得到两个匹配的文件名 ch1.c 和 ch2.c。

单独的"*"可以与所有文件名相匹配。

```
/home/zhang$ ls *
ch1.c  ch2.c  ch3.f  result  test
/home/zhang$
```

我们再来看下面的例子:

```
/home/zhang$ ls c*
ch1.c  ch2.c  ch3.f
/home/zhang$ ls *t
result  test
/home/zhang$ ls t*t
test
/home/zhang$
```

将通配符"*"转换成文件名的过程称为通配符的扩展。这一过程是由 shell 来完成的。命令自身是没有识别参数表中"*"的能力的。shell 将通配符扩展为所有能与之匹配的文件名。

需要说明的是:通配符"*"不能与单句点(.)开头的文件名相匹配。以单句点开头的文件被看成是"隐藏"文件——这种隐藏不表示真正意义上的隐藏,而只是不能被一般的 ls 命令列出。这些隐藏文件的文件名中的单句点必须显式匹配。例如,我们用带有-a 选项

的 ls 命令,可以看到包括以“.”开头的所有文件:

```
/home/zhang$ ls -a
.  ..  .csh-profile  .cshrc  ch1.c  ch2.c  ch3.f  result  test
/home/zhang$
```

我们看到,当前目录中有两个隐藏文件——.csh-profile 和.cshrc。我们进行下列操作:

```
/home/zhang$ ls *
ch1.c  ch2.c  ch3.f  result  test
/home/zhang$ ls .cs*
.csh-profile  .cshrc
/home/zhang$
```

除通配符“*”外,还有其他的通配符,例如:

?	匹配任意单一字符
[	引入一个字符组
]	结束一个字符组
-	表示一种字符范围

“?”匹配任一单一字符。因此,命令“ls ?”将列出所有单字符文件名,“ls tes?”能列出文件名“test”,但不能列出文件名“test.bak”。

方括号和“-”用于构成字符组模式。在字符组中的字符由方括号括起来。模式“abc[aeiou]”将匹配任何以字符“abc”开头而以任一元音字母结尾的文件名。“-”可以用在一对方括号内部,以表明字符的范围。模式“def[0-9]”将匹配任何前三个字符为“def”,第四个字符为数字的文件名。请看下面的例子:

```
/home/zhang$ ls ch[0-9].c
ch1.c ch2.c
/home/zhang$
```

字符“-”在方括号外面就失去了通配符的作用。而字符“*”和“?”在方括号内时,就失去了作为通配符的能力。另外,若要使某一个通配符失去作为通配符的能力,可以在它前面加一个反斜杠“\”。因此,ls * * 命令将列出当前目录中名字以“*”结尾的文件名。

取消通配符能力的另一途径是引号。如:用命令 ls *“*”可以列出所有以“*”为结尾的文件名。

综前所述,用户可以使用通配符一次引用多个文件,这样可以使一些命令大大简化。如:

```
/home/zhang$ cp /etc/s* .
```

将把/etc 目录下的所有以“s”开头的文件都复制到当前目录/home/zhang 中。

3.6 Linux 系统的输入和输出

3.6.1 标准输入和标准输出

一般情况下,Linux 从标准输入中读取输入信息,并将输出信息送到标准输出。标准输入和标准输出通常缩写成“stdin”和“stdout”。用户的 shell 通常将标准输入设置成键盘,而将标准输出设置成显示器。

我们以 cat 命令为例。cat 命令的功能是从命令行的所有文件中读取数据,并把这些数据直接送到标准输出。因此,使用命令

```
/home/zhang/papers$ cat history-paper math-paper
```

将会把文件 history-paper 和 math-paper 的内容依次显示到屏幕上。

但是,如果 cat 的命令行中没有参数,它就会从标准输入中读取数据,并将其送到标准输出。例如:

```
/home/zhang/papers$ cat
Hello, world.
Hello, world.
Bye.
Bye.
ctrl-D
/home/zhang/papers$
```

正如上面所显示的那样,用户输入的每一行都立刻被 cat 命令输出到屏幕上。当从标准输入中读取数据时,命令在接收到 EOT(end-of-text)后便知道输入已经结束。通常,EOT 由键入 ctrl-D 产生。

下面请看另一个例子。命令 sort 按行读入文件正文(当命令行中没有给出文件名时,表示从标准输入读入),将其排序,并将结果送到标准输出。下面的例子是从标准输入读入一个采购单,并将其进行排序。

```
/home/zhang/papers$ sort
bananas
carrots
apples
ctrl-D
apples
bananas
carrots
/home/zhang/papers$
```

现在我们在屏幕上得到了已排序的采购单。

3.6.2 输入、输出重新定向

由于标准输入和标准输出是由 shell 建立的，因此，可以由 shell 重新指定标准输入和标准输出。输入和输出的重新定向，是 Linux 以及其他 UNIX 操作系统的重要特征之一。

我们希望将上例中的采购单保存在一个文件中。在命令行使用“ > ”时，shell 将把标准输出重新定向到一个文件中。如下例所示：

```
/home/zhang/papers$ sort > shopping-list
bananas
carrots
apples
ctrl-D
/home/zhang/papers$
```

这次，sort 命令的结果没有被显示在屏幕上，而是被存到文件 shopping-list 中。我们来看一下这个文件的内容。

```
/home/zhang/papers$ cat shopping-list
apples
bananas
carrots
/home/zhang/papers$
```

现在，我们假设原始的、没有经过排序的采购单是保存在文件 terms 中，而不是从键盘读入。我们在 sort 命令的命令行中给出该文件的文件名，来对采购信息进行排序，并对标准输出进行重新定向。

```
/home/zhang/papers$ sort items > shopping-list
/home/zhang/papers$ cat shopping-list
apples
bananas
carrots
/home/zhang/papers$
```

还有另一种方法来完成上述功能。我们不仅可以对标准输出进行重新定向，还可以对标准输入进行重新定向。这可以通过在命令行中使用“ < ”符合来实现。

```
/home/zhang/papers$ sort < items
apples
bananas
carrots
/home/zhang/papers$
```

命令 sort < items 和命令 sort items 的结果是相同的，但两者的实现机制是不同的。前者使我们有机会验证了标准输入的重新定向。在 sort < items 命令执行时，sort 命令把文件 item 作为标准输入，并从中读取数据。输入的重新定向是由 shell 完成的，sort 命令

自身并没有得到要读入的文件的名称。sort 命令仍然从标准输入读入数据，就像用户从键盘键入数据一样。

以上例子引进了过滤器的概念。过滤器是一种从标准输入读入数据，对它进行某种处理，然后将处理的结果传送到标准输出的程序。用户可以通过重新定向将标准输入和标准输出设置成文件。sort 命令是一个简单的过滤器的例子：它将输入的数据进行排序，并将结果送到标准输出。cat 命令则是一个更简单的过滤器，它不对输入的数据进行任何操作，而只是简单地将其输出。

使用“ > ”将标准输出重新定向到一个文件有时是有破坏性的。例如，命令

```
/home/zhang/papers$ ls  > file-list
```

将会重新改写文件 file-list 中的内容。如果用户改用符号“ > > ”进行重新定向，当指定文件已经存在时，输出将会被附加在指定文件原有内容的后面，而不是改写它。

例如，当文件 file-list 已经存在时，命令

```
/home/zhang/papers$ ls  > > file-list
```

将把 ls 命令的输出流写到文件 file-list 的原有内容的后面。

3.6.3 管道的使用

在上节中，我们已经讲述了如何将 sort 命令作为一个过滤器。但是，如果需要进行排序的数据来自另一个命令的输出，如 ls 时，我们将如何操作呢？例如，sort 命令的 -r 选项可以将数按反字母顺序进行排序。如果我们要将当前目录下的文件以反字母顺序列出，一种方法是：

```
/home/zhang/papers$ ls
computer-paper
history-paper
math-paper
reports
/home/zhang/papers$ ls  > file-list
/home/zhang/papers$ sort -r  file-list
reports
math-paper
history-paper
computer-paper
/home/zhang/papers$
```

这里，我们使用的方法是将 ls 命令的输出存入到一个文件中，然后对这个文件执行 sort -r 命令。这种方法比较笨拙，并且使用了临时文件来保存 ls 命令产生的数据。

解决这个问题的方法是使用流水线的操作方式。流水线是 shell 的另一个特性，它允许用户将一串命令连成一个“管道”：第一个命令的标准输出被直接设置成第二个命令的标准输入，依次类推。在下面的例子中，我们将 ls 命令的标准输出送到 sort 命令的标准输入。符号“©¦”用来建立管道。

```
/home/zhang/papers$ ls ©¦sort -r
reports
math-paper
history-paper
computer-paper
/home/zhang/papers$
```

这个命令形式就非常简洁了。

下面我们介绍一个非常有用的使用管道的例子。如果用下面的命令显示/usr/bin 下的一长串文件名,大部分文件名会在屏幕上一闪而过,用户根本来不及阅读它们。

```
/home/zhang/papers$ ls /usr/bin
```

现在,可以加上命令 more,并用管道把两个命令连起来,如下所示:

```
/home/zhang/papers$ ls /usr/bin ©¦more
```

这样,我们就可以轻松地、一页一页地浏览 /usr/bin 中的文件名了。

还不止如此! 我们还可以将两个以上的命令串接到一起组成一个管道。命令 head 的功能是显示标准输入的头几行,显示的行数由命令后面的参数指定。如果我们希望显示出当前目录中字母顺序为最大的文件名,可以使用下面的命令:

```
/home/zhang/papers$ ls ©¦sort -r ©¦head -1
reports
/home/zhang/papers$
```

在这个管道中,命令 head -1 的作用是显示出它所得到的输入流的头一行。它的输入流是经过 sort -r 命令按反字母顺序排序后的 ls 命令输出的文件名。

请用户注意:输入、输出的重新定向以及管道的使用都是 shell 提供的特殊功能——是 shell 提供了“>”、“>>”、“<”和“©¦”的使用形式。命令自身没有对此作出任何贡献,功劳是属于 shell 的。

3.7 文件权限

3.7.1 文件权限的概念

在一个 Linux 系统中有多个用户。为了保护用户个人的文件不受其他用户的侵犯,Linux 系统提供了文件权限的机制。这种机制允许文件和目录归一个特定的用户所有。这个用户有权对他所拥有的文件或目录进行存取或其他操作,也有权限制其他用户在这些文件上的存取和操作权限。例如,用户 zhang 在自己的主目录中建立了一些文件,那么用户 zhang 就是这些文件的所有者,有权对这些文件进行存取,也有权限制其他用户对这些文件进行存取。

Linux 还允许用户之间以及用户组之间共享文件。如果用户 zhang 愿意,他可以切断所有到他的文件的通路,使得任何人都不能存取这些文件。在大多数系统中,缺省的情况是允许其他用户读自己的文件,但不能对它们进行修改和删除。

如上所述,每个文件都被一个特定用户所拥有,但同时文件还被系统指定的用户组所拥有。每个用户在被建立用户目录时都被放到至少一个用户组中。但是,系统管理员还可以将用户编进多个用户组里。

用户组通常是根据使用计算机的用户的种类来划分的。例如,在一个学校的 UNIX 系统上,用户可以被划分成 student、staff、faculty 或 guest 等组。除以上的用户组外,还有几个系统定义的组(如 bin 和 admin),系统使用这些组来控制对资源的访问——普通的用户很少属于这些系统组。

文件的权限有三部分:读、写和执行。这些权限可以分别赋予三种类型的用户:文件的所有者、文件所属的用户组和组外的其他所有用户。

读权限允许用户阅读文件的内容,或对目录来说,允许用户列出目录中的内容(使用 ls 命令)。写权限允许用户写和修改文件,或对目录来说,写权限允许用户在这个目录中建立新文件或删除文件。最后,执行权限允许用户运行文件即程序或 shell 文件(如果这个文件是程序或 shell 文件),或对目录来说,执行权限允许用户进入和退出该目录。

3.7.2 文件权限的解释

让我们用一个例子来解释上面所说的文件权限。使用带有 -l 选项的 ls 命令来显示包括文件权限的长格式信息。

```
/home/zhang/papers $ ls -l history-paper
-rw-r--r-- 1 zhang users 505 Mar 13 19: 05 history-paper
/home/zhang/papers $
```

在上面所列出的文件的长格式信息中,第一列代表文件的类型和权限;第二列代表文件的链接数(我们将在 3.8 节介绍文件的链接);第三列是文件的所有者(zhang);第四列是文件所属的用户组(users);第五列代表以字节为单位的文件的长度;接着的三列代表文件的最后修改的日期和时间;最后一列是文件的名(history-paper)。上面的例子中,文件 history-paper 的所有者是 zhang,属于 users 组,文件的长度为 505 个字节,最后修改的日期和时间为 3 月 13 日 19: 05。下面我们来看一下该文件的权限。

上面列出的“-rw-r--r--”串表示文件“history-paper”的类型和权限。其中的第一个字母表示文件的类型。“-”表明该文件是一个普通文件。“d”代表目录,“b”代表块设备文件,“c”代表字符设备文件。后面九个字符每三个一组依次代表文件的所有者、文件所在的用户组以及组外其他用户对该文件的访问权限。

表示文件权限的三个字符,依次代表读、写和执行权限。系统用“r”代表读权限、“w”代表写权限、“x”代表执行权限。当用户没有相应的权限时,该权限对应的位置用短线“-”表示。如上面的例子中,第一个短线后面的三个字符“rw-”表示用户 zhang 自己对该文件有读、写权限,但没有执行权限。接下去的三个字符“r--”表示与用户 zhang 同属于 users 组的用户对该文件有读权限,但没有写权限和执行权限。也就是说,属于 users 组的其它用户只能读该文件。最后的三个字符“r--”表示组外的其他用户也只有该文件的读权限,即只能阅读该文件,但不能对该文件进行写或执行操作。

下面,我们来看一下目录的权限:

/home/zhang$ ls -l reports

drwxr-xr-x 1 zhang users 1024 Mar 13 14:00 reports

/home/zhang$

在上例中,字符串"drwxr-xr-x"表明 reports 是一个目录,其所有者 zhang 对该目录具有读、写和执行权限,即用户 zhang 可以进入该目录,可以对其中的文件进行查看、修改,并可以在其中建立新文件。而其他的用户只能进入该目录,并列出其中的内容。

3.7.3 文件权限的依赖性

需要特别指出的是,用户对文件所拥有的权限还要依赖于他对文件所属的目录拥有的权限。例如,即使一个文件的权限为"-rwxrwxrwx",其他用户也只有在具有该文件所属目录的读和执行权限时,才能够对该文件进行读、写和执行。

例如,如果用户 zhang 希望限制他人对他所有文件的访问,他只需将他的主目录/home/zhang的权限设置成"-rwx------"。这样,其他任何用户都不能访问他的目录,包括其中的所有文件和子目录。用户 zhang 不需要一个个地修改他的每一个文件的权限。

换一句话说,为了访问一个文件,用户必须具有对该文件路径名中所有目录的执行权限以及对该文件自身的相应的访问权限。

通常,UNIX 系统上的用户都是非常开放的,文件的权限一般都设置成"-rw-r--r--",这种权限允许其他用户读文件,但不能以任何方式改变它。目录的权限一般都设置成"-rwxr-xr-x",这种权限允许其他用户查看目录,但不能在其中建立或删除文件。

但是,许多用户都不希望其他用户访问自己的文件。这时,用户可以把文件的权限设置成"-rw------"。这样,其他用户将不能访问该文件(包括读)。同样,用户可以把目录的权限设置成"-rwx------"以防止其他用户访问该目录。

3.7.4 改变权限

命令 chmod 用来设置或改变文件的权限。只有文件的所有者和超级用户可以改变文件的权限。命令 chmod 的语法是:

chmod {a,u,g,o} {+,-} {r,w,x} <文件名>

该命令首先要指明权限所要赋予的用户类,可以是 a、u、g、o 中的一个或多个,其中 a 代表所有的用户,u 代表文件的所有者自身,g 代表同组的用户,o 组外的其他用户。接着,要指明是增加权限还是删除权限,用"+"选项表示增加权限,用"-"选项表示删除权限。最后,要指明所要增加或删除的权限,这些权限可以是 r、w 和 x 中的一个或多个,其中 r 代表读权限,w 代表写权限,x 代表执行权限。

另外,用户还可以用一个八进制数来设置权限,语法如下:

chmod nnn <文件名>

其中,nnn 代表三个八进制的数字,分别对应于用户自身、同组用户和组外其他用户的权限,每个八进制数字所对应的三位二进制数字分别对应于读、写和执行权限。"1"表示赋予相应的权限,"0"表示不赋予相应的权限。

下面举几个例子:

chmod a+ r history-paper	将该文件的读权限赋予所有的用户
chmod + r history-paper	与上面命令的功能一样——当没有指明用户类(augo)时,缺省为 a
chmod og-x history-paper	删除文件所有者以外的所有用户的执行权限
chmod g+ rw history-paper	允许该文件的同组用户读、写该文件
chmod 666 history-paper	将该文件的读、写权限赋予所有的用户

3.8 文件链接

文件的链接可以使 UNIX 系统中的一个文件有多个名字。系统是用文件的 i 节点号来识别文件的,i 节点号是文件系统中文件的唯一标识符。目录实际上是 i 节点号和其相应文件名的一个列表。目录中的每一个文件都是一个与特定的 i 节点的链接。

3.8.1 硬链接

命令 ln 用来为一个文件建立多个链接。例如,假设用户在当前的目录中有一个文件 foo。使用命令 ls -i,我们可以看到这个文件的 i 节点号。

```
$ls  -i  foo
22192  foo
$
```

可见,文件 foo 在文件系统中的 i 节点号是 22192。用户可以用以下命令建立一个称作 bar 的到文件 foo 的链接:

```
$ ln  foo  bar
```

使用 ls -i ,我们可以看到这两个文件有相同的 i 节点号。

```
$ ls  -i  foo  bar
22192  bar  22192  foo
$
```

现在,不论访问 foo 还是 bar,实际上都是访问一个文件。如果对 foo 做了修改,实际上也是对 bar 作修改。从任一角度来看,foo 和 bar 都是同一个文件。

上面的链接称为硬链接,因为它们直接链接到 i 节点上。注意,用户只能在同一个文件系统中对文件建立硬链接,符号链接(在“3.8.2 符号链接”中会讲到)就不会有这个限制。

当用户用 rm 命令删除文件时,实际上是删除了文件的一个链接。例如,如果用户使用命令

```
$ rm foo
```

则只是名字为 foo 的链接被删除了,文件 bar 依然存在。只有当文件没有其他链接时,才被真正地从文件系统中删除。通常,文件只有一个链接,所以使用 rm 命令会删除文件自身。但是,如果一个文件有多个链接,那么使用 rm 命令将只删除该文件的一个链接。为了删除文件,用户必须首先删除该文件的所有链接。

命令 ls -l 可以列出文件的链接数以及其他信息。

```
$ ls -l foo bar
-rw-r--r-- 2 root root 12 Aug 5 16: 51 bar
-rw-r--r-- 2 root root 12 Aug 5 16: 50 foo
```

上面信息的第 2 列代表文件的链接数。

正如上面所说,目录实际上只是一个包含了从链接到 i 节点号的转换信息的文件。每个目录中至少有 2 个硬链接: 指向目录自身的链接‘. ’和指向其父目录的链接‘. . ’。根目录(/)中的‘. . ’链接也指向其自身。

3.8.2 符号链接

符号链接与硬链接不同,它是另一种类型的链接。符号链接允许用户为文件取另个名字,但却不将其与 i 节点相链接。

命令 ln -s 将建立一个文件的符号链接。例如,如果我们使用命令

```
$ ln -s foo bar
```

就会建立一个指向文件 foo 的符号链接 bar。如果我们使用命令-i,便会看到这两个文件实际上有不同的 i 节点。

```
$ ln -i foo bar
22195 bar 22192 foo
$
```

使用命令 ls -l,我们会看到文件 bar 是指向 foo 的一个符号链接。

```
$ ls -l foo bar
lrwxrwxrwx 1 root root 3 Aug 5 16: 51 bar→foo
-rw-rw-r-- 1 root root 12 Aug 5 16: 50 foo
$
```

符号链接的权限位是没有意义的。它们总是为 rwxrwxrwx。实际上,符号链接的权限是由符号链接的目标(在我们的例子中,是文件 foo)的权限决定的。

从功能上讲,硬链接和符号链接是相似的,但是它们也有一些区别。例如,用户可以对一个不存在的文件建立符号链接;但硬链接却不允许这样。系统核心对符号链接和硬链接的操作是不同的。虽然这只是技术上的不同,但这一点有时却很重要。符号链接只是指明它们所指向的文件,这对用户是很有帮助的;而硬链接的问题是,没有一个简单的办法用来确定有哪些文件是链接到相同的 i 节点上。

Linux 系统中很多地方都用到了链接。对/lib 中的共享程序库映象来说,符号链接显得更为重要。有关程序库的管理请参阅第四章。

3.9 作业控制

3.9.1 作业和进程

作业控制是许多 shell(包括 Bash 和 Tcsh)的一个特性,它允许用户在同一个时刻控

制多个正在运行的命令,或作业。在进一步论述之前,我们先谈一谈进程的概念。

每当运行一个程序,我们都启动了一个进程。我们把正在运行的程序称为进程,命令 ps 将显示正在执行的进程的信息,请看下面的例子:

```
/home/zhang$ ps
PID   TTY   STAT   TIME   COMMAND
24    3     S      0:03   (bash)
161   3     R      0:00   ps
/home/zhang$
```

第一列的 PID 称为进程的标识号,是系统赋予每个正在执行进程的唯一编号。最后一列 COMMAND,是正在执行的进程的名称。上面的命令中我们只看到了用户 zhang 当前正在运行的进程,它们是 bash(用户 zhang 的 shell)和 ps 命令自身。正如用户所看到的,bash 与 ps 命令同时运行。当 zhang 敲入命令 ps 后,bash 执行该命令。ps 完成运行后(即进程表显示完后),控制返回 bash 进程,bash 显示提示符,等待另一个命令。

一个正在执行的进程对 shell 来说称为一个作业。术语进程和作业是可互换的。但是,一般来论,进程与作业控制相关联时,才被称为作业。作业控制是 shell 提供的使用户能够在多个独立作业间进行切换的功能。

在大多数情况下,用户在同一时间只运行一个作业,即他们最后向 shell 键入的命令。但是,使用作业控制,用户可以同时运行多个作业,并在需要时在这些作业间进行切换。这会有什么用途呢?例如,当用户编辑一个文本文件,并需要中止编辑做其他事情时,利用作业控制,用户可以让编辑器暂时挂起,返回 shell 提示符开始做其他的事情。当用户做完之后,可以重新启动挂起的编辑器,返回到刚才中止的地方,就像用户从来没有离开编辑器一样。这只是一个例子。作业控制还有许多其他实际的用途。

3.9.2 前台和后台

作业可以在前台,也可以在后台运行,在同一时间只能有一个作业在前台运行。前台作业是指正在对它进行交互操作的作业,它从键盘接收输入,并将输出送往屏幕(当然,用户可以对输入输出进行重新定向)。另一方面,后台的作业不接收终端的输入。一般来说,它们在运行时不需要交互操作,而只是默默地运行。

有一些作业需要很长时间才能结束,并且运行起来很单调乏味。编译程序就是一个这样的作业。我们没有必要坐等这些作业的完成,我们只需在后台运行它们就可以了。当作业在后台运行时,用户可以自由地运行其他程序。

作业还可以被挂起,被挂起的作业是指当前不在运行、但只是被暂时中止的作业。用户挂起一个作业后,可以在需要时在前台或后台恢复它的运行。恢复被挂起的作业将不会改变作业的状态——作业将从被中止处开始继续运行。

注意:挂起作业并不等于中断作业。当用户中断一个正在执行的进程时(通过按中断键,通常是 Ctrl-C),该进程将被永远撤消。当一个作业被撤消后,就再也不可能被恢复;用户只能重新运行这个命令。还要注意:有的程序对中断设置了陷井,当敲入 Ctrl-C 键后,并不马上撤消该进程。目的是允许程序在终止前进行必要的清理操作。实际上,有些程序

不允许用户用中断来终止运行。

3.9.3 后台运行作业和撤消作业

让我们先举一个简单的后台运行作业的例子。命令 yes 表面上看起来是一个没有什么用途的命令,它只是向标准输出发送无穷尽的一串 y(这实际上是很有用的的一个命令。如果用户将 yes 的输出通过管道送到另一个提问一串 yes 或 no 问题的命令,y 串将会确认所有这些问题)。

```
/home/zhang$ yes
y
y
y
 ┆
```

该命令将打印出一连串 y。用户可以通过输入中断键(通常为 Ctrl-C)来中断这个进程。我们可以将 yes 的标准输出重定向到/dev/null。前面曾讲过,/dev/null 就像一个黑洞,任何被送入这个黑洞的数据都会消失。

```
/home/zhang$ yes > /dev/null
```

这时,我们看到,屏幕上没有任何信息,并且 shell 提示符也不存在了。这是因为 yes 命令还在运行,并且将它所产生的 y 串送往/dev/null。我们用中断键 Ctrl-C 来终止 yes 的运行。

假设我们希望 yes 命令继续运行,但同时也希望 shell 提示符仍回到屏幕上去做其他的事情。这时可以将 yes 放到后台运行。这样,就能够在不需要任何交互操作的情况下使它继续运行。

将一个进程放到后台运行的一种方法是在命令后面加“&”字符。

```
/home/zhang$ yes > /dev/null &
[1] 164
/home/zhang$
```

这时,用户可以看到,shell 的提示符又回到屏幕上。屏幕上显示的[1]代表 yes 进程的作业号。shell 为每一个正在运行的作业都赋予了一个作业号。因为 yes 是当前运行的唯一的一个作业,所以它被赋予作业号 1。“164”是进程标识符,或称“PID”,它是系统赋予这个进程的标号。在下面看到这两个号码都可以用来引用作业。

现在,yes 进程正在后台运行,并不断地把一连串 y 字符送给/dev/null。要查看这个进程的状态,可以使用 shell 内部命令 jobs:

```
/home/zhang$ jobs
[1]+  Running yes > /dev/null
/home/zhang$
```

还可以用 ps 命令来查看作业的状态。

可以使用 kill 命令来撤消这个作业。kill 命令用作业号或 PID 作为参数来指定要撤销的作业。当使用作业号来标识作业时,在作业号前面一定要加百分号“%”。该作业的作

业号为 1,所以用下列命令

/home/zhang$ kill %1

将会撤消这个作业。

现在,我们已经撤消了这个作业,可以再用 jobs 命令来检查一下。

/home/zhang$ jobs

[1]+ Terminated yes > /dev/null

/home/zhang$

结果证实了这个作业确实已被撤消。如果再次使用 jobs 命令,屏幕上将不会显示任何信息。

还可以使用进程标识号(PID)来撤消作业。在这个例子中,PID 为 164,

所以命令

/home/zhang$ kill 164

与命令

/home/zhang$ kill %1

是等价的。

当使用进程标识号来指定作业时,进程标识号之前不加"%"。

3.9.4 中止作业和重新启动作业

还有另外一种方法用来将作业放到后台运行。首先正常地在前台启动一个作业,然后暂时中止它,最后再在后台重新启动它。

例如,像平常一样,在前台启动 yes 进程:

/home/zhang$ yes > /dev/null

当然,由于 yes 是在前台运行的,用户不能在屏幕上看到 shell 的提示符了。现在,将这个作业挂起,而不是将其用 Ctrl-C 中断。挂起一个作业将不会撤消这个作业,而只是将这个作业暂时中止了,直到用户重新启动它。用户可通过键入挂起键(通常是 Ctrl-Z)来挂起这个作业。

/home/zhang$ yes > /dev/null

Ctrl-Z

[1]+ stopped yes > /dev/null

/home/zhang$

当一个作业挂起后,它只是不运行了。这时,这个作业不再占用 CPU 时间。用户可以重新启动这个作业,使这个作业重新运行,就像什么都没有发生过一样。作业将在被中止的地方继续运行。

使用 fg 命令,可以在前台重新启动一个作业。

/home/zhang$ fg

yes > /dev/null

shell 将把命令的名字重新显示到屏幕上,使用户知道目前哪个作业放在前台执行。用户可以再次使用 Ctrl-Z 中止这个作业,改成用命令 bg 将这个作业放入后台执行。这

时,命令将在后台运行,就像在上节中用“&”来启动这个命令一样。

/home/zhang$ bg

[1]+ yes > /dev/unll &

/home/zhang$

这时,我们又在屏幕上得到了 shell 提示符。jobs 命令会告诉用户 yes 命令确实在运行。我们可以像前面那样用 kill 命令来撤消这个命令。

如何再次中止这个作业呢? 因为作业是在后台运行,使用 Ctrl-Z 已经没有什么用了。要中止该作业,应先用 fg 命令将这个作业放到前台,然后再中止它。可用 fg 命令将已中止的作业或在后台运行的作业放到前台运行。

后台作业和被中止的作业有很大不同。被中止的作业没有运行——它不再使用任何 CPU 时间,并且它也不做任何工作。这个作业仍然占用系统内存,但是这些内存可能被交换到硬盘上。而后台作业则是在运行、使用内存、并且与用户的其他作业同时进行的一些任务。这里要提醒用户的是,如果一个后台作业有向显示器终端的标准输出,那么该作业在运行时可能会向用户终端显示一些信息。这样的显示有时会干扰用户的前台作业。例如,如果用户使用命令

/heme/zhang$ yes &

而没有将输出重新定向到/dev/null, y 串会被显示到用户的屏幕上。为了终止这无穷无尽的“y”,用户只能使用 fg 命令把作业放到前台,并使用 Ctrl-C 终止它的运行。另外要注意,fg 和 bg 命令的缺省作业是最近被中止的作业。当用户用 jobs 命令列出时,作业号后的“+”号表明该作业是最近被中止的作业。如果用户同时运行多个作业,fg 或 bg 命令可以使用作业 ID 作为参数来将指定的作业放到前台或后台,如:

/home/zhang$ bg %2

(作业 2 被放到前台),或

/home/zhang$ bg %3

(作业 3 被放到后台)

在 fg 或 bg 命令中不能使用进程标识符。另外,使用单独的作业号也可以将作业放到前台。例如命令

/home/zhang$ %2

与命令

/home/zhang$ fg %2

是等价的。

读者要注意的是,作业控制是 shell 的一个特性。fg、bg 和 jobs 命令是 shell 的内部命令。如果由于某种原因用户所使用的 shell 不支持作业控制,那么这些命令就不会起作用。

3.9.5 作业的顺序执行

用户可以在一个命令行同时输入两个或更多的命令来使作业顺序执行。这时,作业之间要用分号(;)或与号(&)分开。Linux 系统将按照命令行上的顺序来执行这些作业,并将所有的输出在屏幕上显示出来。

下面我们举一个例子:

```
/home/zhang/report $ cd; pwd; ls -F
/home/zhang
Mail/ letters/ papers/ reports/
/home/zhang $
```

该命令将顺序执行以下三个命令:

(1) cd 命令将使当前工作目录变为用户的主目录;

(2) pwd 命令显示当前工作目录的全路径;

(3) 列出当前工作目录中的内容。

另外, Bash 和 Tcsh 在作业控制方面有些地方是不同的。不过, 这种差别是很细微的。也有些 shell 根本就不支持作业控制。但是, 大多数 Linux 系统上的 shell 是支持作业控制的。

3.10 X 窗口系统

3.10.1 X 窗口系统介绍

X 窗口系统是在 1984 年由美国麻省理工学院(MIT)在 DEC 的支持下开发的一个分布式的图形工作环境。X 是 MIT 的计算机科学实验室为雅典娜(Attena)计划而设计的, 它最初主要在 MIT 及 DEC 内部使用。随着版本的发行, 很多厂商开始将 X 作为一个商业产品。X 窗口系统的版本 11 是由一个硬件和软件制造商联合会(即 X 联合会)支持开发的。X 联合会将 X 作为它们产品系列的用户界面, 支持并控制 X 窗口系统的标准规范。X 窗口系统已经在大多数 UNIX 系统、DEC 的 VAX/VMS 操作系统、Linux 操作系统以及许多计算机系统上实现。

X 窗口系统允许应用程序员开发可移植的图形用户界面。它最重要的特性之一就是它的设备独立型体系结构。X 允许应用程序在任何支持 X 协议的硬件上显示包含文本及图形的窗口, 而不必改写应用程序。这种设备的独立性, 加上 X 作为工业标准的位置, 使基于 X 的应用程序可以在一个由大型机工作站以及个人计算机组成的异构环境下工作。

X 窗口系统(以后我们简称为 X)每几年就有一个新的版本, 我们称为发行版。到写本书时为止, 最后的版本是 X11R6, 也就是发行版 6。X11 中的 11 是正式的版本号。

X 的体系结构是建立在客户-服务器模型基础上的, X 的应用程序称为客户(client)。例如, xterm 就是一个客户程序, 在用户登录后显示用户的 shell。对所有客户程序提供服务的程序称为服务器(server)。服务器创建并操纵屏幕上的窗口, 产生文本及图形, 并处理像键盘、鼠标这样的输入设备。服务器在所有应用程序及显示器硬件之间提供了一个可移植层。例如, 服务器为 xterm 建立了窗口, 并且与用户进行通信。

客户和服务器是两个分离的程序。客户使用异步字节流协议, 通过一个网络互联, 与服务器进行通信。X 支持很多网络协议, 包括 TCP/IP、DECnet 等。多个用户可以同时与一个服务器相联; 同样, 一个客户也可以同多个服务器互联。这样, 客户和服务器就可以在

两个物理上分离的机器上分别运行了,这正是 X 的一大优点。这样,只要提供了运行图形的标准,X 用户就可以在远程的机器上运行程序,并且让该程序的输出显示在用户面前的工作站上。

还有一个用户可能已经熟悉的术语是窗口管理程序。窗口管理程序是一个特殊的客户程序,它告诉服务器放置各种窗口的位置,并为用户提供了移动这些窗口的方法。服务器自身并没有为用户作任何事情,它只是提供了一个用户和客户程序之间的缓冲区。

3.10.2 启动或退出 X 窗口系统

1. 启动 X。如果用户登录后没有自动启动 X,可以在通常的文本 shell 提示符下启动它。启动 X 的命令可能是 startx、xinit 或 openwin。用户可以试着用这些命令来启动,请参阅用户所用版本的文档。如果 X 没有被启动,则可能使用的系统上并没有安装 X,或者是 X 的配置不正确。

2. 退出 X。X 的菜单中包括了退出 X 的菜单项——Exit Window Manager 或 Exit X 或其他类似的菜单项。请用户试着找到这个菜单项(记住,X 系统中可能有不止一个菜单,请用户试一下其他鼠标键)并选择它。如果 X 是在用户登录时自动启动的,这个操作会使用户退出 Linux;如果 X 是被用户手动启动的,这个操作会使用户返回文本状态的提示符下。

3.10.3 X 的启动

当用户第一次启动 X 时,同时会启动几个程序。首先,服务器被启动。然后,一些常用的客户程序被启动。被启动的客户程序在不同机器上可能有的不同。通常是窗口管理程序即 fvwm 或 twm、shell 提示符即 xterm 和时钟即 xclock。用户也可以自己启动一些需要的客户程序:按鼠标键,屏幕上会出现一个菜单(在下一节我们会介绍 X 的菜单)。可以移动鼠标来选择相应的客户程序。

1. Xclock

首先介绍一个最简单的客户程序:xclock。正像用户所希望的那样,xclock 的功能是在一个小窗口中向用户显示时间。

在 xclock 窗口中所进行的一切动作,如键入或击按鼠标,都不会影响它的工作。它的功能只是指示时间。用户可以为 xclock 程序选择不同的选项来使它在不同的方式下工作。例如,xclock -digital 可以建立一个数字钟。xclock -update 1 将建立一个每秒移动一步的秒针,xclock -update 5 可以建立一个每 5 秒移动一步的秒针。

有关 xclock 选项的信息,可参阅它的联机手册——man xclock.

2. Xterm

带有 shell 提示符(例如/home/zhang $ 等)的窗口是由称为 xterm 的客户程序来控制的。xterm 通常在 X 启动时被启动,但有时也需要用户自己通过菜单启动。xterm 是一个复杂的程序。第一眼看来,它好像没有做什么事情,但实际上它做了大量的工作。xterm 仿效终端以使通常文本格式的 UNIX 应用程序能正确地运行。

在本章前面的章节中,我们讲述了许多有关 Linux 的命令行的知识,这些都可以在

xterm 窗口中使用。在使用 xterm 窗口之前,用户应将鼠标光标(一般是 X 或箭头的形状)移到 xterm 窗口中将其激活。这一动作是与窗口管理程序有关的。

3.10.4 窗口管理程序

在 Linux 系统上,通常有两种不同的窗口管理程序。一个称为 twm,它是 Tab Window Manager 的缩写。另一个是经常使用的 fvwm。twm 比 fvwm 要大。twm 或 fvwm 都是可配置的,用户可以自己设置各个键所对应的功能。

1. 建立新窗口

当一个新的窗口被建立时,窗口管理程序可能做三件事情。用户可能通过窗口管理程序使得在新窗口的轮廓显示出来后,可以在屏幕上安排它的位置,这称为手工放置。

窗口管理程序也可以将新窗口放置到屏幕上的任一位置,这称为任意放置。

最后,有时一个应用会要求被放置到屏幕上的特定位置,或者说,窗口管理程序可以被设置成将特定的应用一直显示在屏幕上的同一位置。例如,用户可以设定将 xclock 一直显示在屏幕的右上角。

2. 当前窗口

窗口管理程序控制着许多重要的事情。用户感兴趣的第一件事情就是"当前窗口"。服务器的当前窗口是指接收用户键盘输入的那个窗口。通常,X 的当前窗口是由鼠标光标的位置来确定的。如果鼠标光标在一个 xterm 窗口中(用户可以同时运行多个 xterm 客户程序),那么这个 xterm 就会接收用户的键盘输入。注意,这与许多其他的窗口系统是不同的,如 MS Windows、OS/2 或 Macintosh,在这些窗口中,用户必须在窗口中击按鼠标,以使该窗口成为当前窗口。另外,通常在 X 窗口中,如果用户的鼠标光标离开了窗口,该窗口便不再是当前窗口了。

但是,请注意,用户可以配置 twm 或 fvwm,以使得用户在使一个窗口成为当前窗口时必须在窗口中击按鼠标,并且当在其他地方击按鼠标时,会使该窗口不再是当前窗口。

3. 移动窗口

在 X 中,用户还可以通过窗口管理程序来配置移动窗口的方法,在 twm 的配置中,有几种不同的移动窗口的方法。最显而易见的方法是将鼠标光标移到窗口的标题区(title bar),并在屏幕上拖动窗口。拖动时,要将鼠标放在标题区,在移动鼠标时要按住鼠标键。

另一个移动窗口的方法是在拖动鼠标时按住一个特定的键,例如,用户可以按住 Alt 键,将鼠标移到一个窗口上,来拖动这个窗口。

4. 窗口的深度

由于 X 系统允许窗口重叠,便出现了窗口深度的概念。尽管窗口和屏幕都是二维的,X 允许一个窗口在另一个窗口的前面,部分或全部地遮盖另一个窗口。

下面是几个有关窗口深度的操作:

- 提升窗口,即将一个窗口放到前面。这个功能通常是由在窗口标题区击按一个鼠标键来完成的。在配置窗口管理程序时指定所需击按的鼠标键。(也可以配置成多个鼠标键)
- 降低窗口,即将窗口放到后面。这通常是由在窗口标题区击按另一个不同的鼠标

键来完成的。也可以将有些窗口管理程序配置成:当有其他窗口遮盖该窗口时,击按鼠标键会将其放到前面;而当该窗口在前面时,击按鼠标键会将其放到后面。

· 在窗口间循环是许多窗口管理程序允许的另一个操作。这个操作将把循环地将每个窗口放到前面。

5. 图标化和最大化

还可以通过其他的访问来完全掩藏窗口。第一个就是图标化。不同的窗口管理程序有不同的方法来完成图标化。在 twm 中,许多用户都配置一个图标管理程序。这是一个特定的窗口,其中包含了屏幕上所有其他的窗口的一个列表。如果用户在屏幕上的窗口的名字上击按鼠标键,窗口便消失了——它被图标化了。但窗口仍然是活动的,只是用户看不到它。在图标管理程序上击按鼠标键,这个窗口会被还原到屏幕上。

这个操作将是很有用的。例如,用户可能拥有许多不同计算机上的远程 xterm。但是,因为用户在一定时间内不会使用所有这些 xterm,用户可以使用这个操作让大多数 xterm 窗口图标化,而只使用其中很少几个。

一些窗口管理程序可能在屏幕的底部建立真正的图标,也可能将图标分散在根窗口中。

窗口管理程序必须支持的另一个操作是最大化,例如,在 twm 中,用户可以将窗口的高度、宽度或两者最大化。不同的应用程序在窗口大小改变时会有不同的反应。例如,xterm 会扩大工作区,但字体并不扩大。

6. 菜单

窗口管理程序的另一个任务是对用户提供菜单,能够快速地执行任务。例如,用户可以建立一个能自动运行 ghostview(一个能阅读 ps 文件的应用程序)的 menu 选择项,这样,用户就不需要在 xterm 的 shell 提示符下键入该命令了。这一点在没有 xterm 运行的情况下更为重要。例如,如果 X 启动时没有启动 xterm,则用户可以通过菜单来启动它。同样,用户也可以通过菜单启动其他程序。

通常,可以在根窗口上击按鼠标键来使用不同的命令。根窗口是所有其他窗口后面的一个不动的窗口。缺省情况下,根窗口是灰色的,但可以被设置成任何其他颜色。用户可以试着看一个菜单,在根窗口上击按并按住鼠标键,一个菜单便会弹出。用户可以将鼠标光标移到一个菜单项上(移动时不要松开鼠标键)并松开鼠标键,这样,该菜单项对应的应用便被投入运行。

3.10.5 X 程序

有很多应用程序都从 X 中得到很大的益处。有些系统,如 emacs,可以在文本方式下运行,也可以在 X 中运行。在 X 中运行时,它会创建自己的 X 窗口。但是,大多数 X 程序只能在 X 下运行。通常使用的 X 程序除前面说过的 xclock、xterm 外,还有 ghostview(ps 文件阅读器),xpaint(图形编辑器)等。

X 还为程序员提供了大量的开发工具。这些工具包括最底层的程序库 Xlib 以及其上的各类开发工具,如 X Toolkit、Openview、Motif,等等。用户可以使用这些工具开发自己的 X 程序。

3.11 屏幕文本编辑器 vi

文本编辑器用于编辑文本文件,如信件、C 语言程序或系统配置文件。Linux 系统提供了许多这样的编辑器,vi 是其中使用最普遍的一种。所有的 UNIX 系统的实现都提供了这种编辑器。vi 是英文“visual editor”的缩写,我们称它为屏幕文本编辑器。vi 屏幕文本编辑器是建立和编辑文件的强有力的工具,其设计目标是用于视频终端,这种终端被用作一个窗口,可以看到文件的正文。用户可以用一些简单的命令对正文进行修改,所作的修改会很快在屏幕上反映出来。

除 vi 外,Linux 还提供了其他的文本编辑器,如 Emacs。Emacs 是一种功能更强的自解释的编辑器。提供适当配置,Emacs 几乎可以仿真任何正文编辑器的基本功能。但由于 Emacs 以及它的支撑文件相对较大,所以并不是任何系统中都配备这个文本编辑器。

在所有的文本编辑器中,vi 不是最易使用的一种,也不是功能最强大的一种。但由于它在 UNIX 世界中使用得最普遍,所以用户应该学会使用它。这就是本节讲述 vi 的原因。

本节只讨论 vi 的基本使用方法,而不对它的一些特殊功能进行过多的讨论。有兴趣的读者可以查阅 vi 的联机手册或专门介绍 vi 的书。

3.11.1 vi 的基本概念

vi 有三种操作方式,称为命令方式、插入方式和命令行方式。

当用户启动 vi 后,便进入命令方式。这种方式允许用户使用特定的命令编辑文件或切换到其他方式。例如,用户在命令方式下键入“x”会删除光标所在处的字符。箭头键可以使光标在用户正在编辑的文件中移动。通常,命令方式下使用的命令有 1-2 个字符长。

用户可以在插入方式下插入或编辑文本。使用 vi 时,用户的大部分时间是在这种方式下工作。通过使用一些命令(例如 i)从命令方式切换到插入方式。在插入方式,用户从当前光标所在的位置向文件插入正文。按 Esc 键可以使用户从插入方式返回命令方式。

命令行方式是一种特殊方式。在这种方式下,可以向 vi 输入特定的扩充命令。用户可以通过在命令方式下键入“:”来进入命令行方式。当进入命令行方式后,输入的命令会在屏幕的最后一行显示出来。例如,可以在命令行方式使用诸如“wq”(写文件并从 vi 中退出)或“q!”(从 vi 中退出,并不保存所作的修改)这样的命令。命令行方式通常用来执行 vi 的那些多于一个字符的命令。在命令行方式中,用户可以输入单行命令,并按 enter 键来执行它。

3.11.2 启动 vi

理解这些概念、学习使用 vi 的最好方法是真正地启动 vi 并进行文件编辑。

vi 命令的语法是:

vi < 文件名>

这里的文件名是用户所要编辑的文件的名称。这个文件可以是一个新文件,也可以是一个已经存在的文件。作为例子,我们用下面的命令启动 vi 来编辑一个名为 test 的新文

件:

/home/zhang $ vi test

这时,屏幕显示如下(我们用方框代表屏幕,假设屏幕中只有 6 行,以节省空间,实际上,vi 可以使用任意大小的屏幕):

```
-
~
~
~
~
~
~
"test" [New file]
```

带有字符"~"的行不是文件中的部分,这些行只在屏幕上显示,而非文件中的行。"-"代表当前光标位置。屏幕的最后一行显示了文件名,并指出这是个新文件。

3.11.3 插入正文

现在我们在命令方式下。键入"i"可以切换到插入方式。在插入方式下,就可以开始插入正文了。

```
Now is the time for all good men to come to the aid of the party.
~
~
~
~
~
```

当插入正文时,可以插入任意多行(当然,在每行的结尾要键入 Return 键)。用户可以使用 Backspace 键擦去刚才键入的错误。

按 Esc 键可以从插入方式返回到命令方式。

在命令方式,可以使用箭头键在所编辑的文件中移动光标。当光标已经被移到文件的边界上而无法继续移动时,vi 会发出嘟嘟声来提醒用户。因为现文件中只有一行,所以当使用向上、向下箭头键时,vi 会发出嘟嘟声。

除了使用 i 命令外,还有其他的方式用来插入正文。例如,a 命令可以从当前光标位置之后插入正文,而不是在当前光标位置插入。如,用户用向左箭头键将光标移到"good"和"men"之间。

```
Now is the time for all good- men to come to the aid of the party.
~
~
~
```

```
~
~
```

按“a”键,开始插入方式,键入“wo”,然后键入Esc键返回到命令方式。

```
Now is the time for all good women to come to the aid of the party.
~
~
~
~
~
```

使用o命令,可以从当前行的下一行开始插入正文。例如,用户按“o”键插入另外一行:

```
Now is the time for all good women to come to the aid of the party.
Afterwards, we'll go out for pizza and beer .
~
~
~
~
```

要记住,在任一时刻,用户都是在三种方式之一下操作:命令方式(在这种方式下,命令i, a, o等是可用的);插入方式(在这种方式下,用户可以插入正文,并用Esc键返回命令方式):命令行方式(在这种方式下,用户可以运行一些扩充命令,有关内容将在下面阐述)。

3.11.4 删除正文

在命令方式下,可以使用x命令删除光标所在处的字符。如果用户按五次x键,便可以得到下面的结果:

```
Now is the time for all good women to come to the aid of the party.
Afterwards, we'll go out for pizza and
~
~
~
~
```

现在按“a”,插入一些正文,再按Esc键,屏幕显示如下:

```
Now is the time for all good women to come to the aid of the party.
Afterwards, we'll go out for pizza and Diet Coke.
~
~
```

```
~
~
```

用户可以用 dd 命令删除一整行(也就是在一行按两次“d”键)。如果光标在第二行,用户连续按两次“d”键,屏幕上会显示:

```
Now is the time for all good women to come to the aid of the party.
~
~
~
~
~
```

使用 dw 命令,可以删除光标所在的单词。将光标放在单词“good”下,键入“dw”,屏幕上会显示:

```
Now is the time for all women to come to the aid of the party.
~
~
~
~
~
```

3.11.5 修改正文

用户可以使用命令 R 来修改正文中的单词。将光标放在“party”的第一个字母上,键入“R”,然后键入单词“hungry”,则屏幕显示如下:

```
Now is the time for all good women to come to the aid of the hungry.
~
~
~
~
~
```

使用命令 R 来编辑正文与 i 和 a 是相类似的,只是 R 命令重写正文,而不是插入正文。

r 命令改写光标所在处的单个字符。例如,将光标移到单词“Now”的开头,键入命令 r 后键入字母 C,可以得到:

```
Cow is the time for all good women to come to the aid of the hungry.
~
~
~
```

```
~
~
```

命令“～”用来改变光标所在字符的大小写。例如,将光标放到‘Cow’单词中的字母“o”上,连续地键入“～”,就会得到:

```
COW IS THE TIME FOR ALL GOOD WOMEN TO COME TO THE AID OF THE HUNGRY.
~
~
~
~
~
```

3.11.6 移动光标

我们已经知道如何用箭头键来在文档中移动光标。除箭头键之外,以下命令也可以用来移动光标:

h	将光标左移一个字符	k	将光标上移一个字符
j	将光标右移一个字符	l	将光标下移一个字符

记住:以上命令只在命令方式下工作。

其他可以用来移动光标的命令是:

H	将光标移到屏幕的左上角	w	将光标向右移动一个字
L	将光标移到屏幕的最后一行	b	将光标向左移动一个字
G	将光标移到文件的最后一行	0	将光标移到当前行的第一个字符
nG	将光标移到文件的第 n 行	$	将光标移到当前行的最后一个字符

当编辑一个大的文件时,可以用下列命令快速地滚动多行:

Ctrl-u光标上滚半屏	Ctrl-f光标上滚一整屏
Ctrl-d光标下滚半屏	Ctrl-b光标下滚一整屏

另外,用户也可以将移动光标的命令和其他命令一起使用。例如,删除命令,命令 d$ 删除从光标所在位置到该行结尾处的所有内容;dG 将删除从光标所在位置到文件结尾处的所有内容,等等。

3.11.7 保存文件并退出 vi

用户可以使用 :q! 命令退出 vi,但这个命令不保存对文件的修改。当用户键入: 后,光标将移到屏幕的最后一行,用户此时处在命令行方式。如下所示:

```
COW IS THE TIME FOR ALL GOOD WOMEN TO COME TO THE AID OF THE HUNGRY.
~
~
~
```

```
~
:_
```

在命令行方式,可以用一些扩展的命令。q! 命令就是其中之一,该命令退出 vi 但不保存文件。命令 wq 可以在保存文件之后退出 vi。在命令方式下使用命令 ZZ(注意,这里没有用)与在命令行方式下使用命令 wq(这里要先键入:进入命令行方式)是等价的。要记住,用户在命令行方式中输入命令之后一定要键入回车(enter)键。

使用: w,可以保存当前编辑的文件,但不会退出 vi。

3.11.8 编辑另一个文件

使用:e 命令可以编辑另一个文件。例如,要停止编辑文件 test,而编辑文件 foo,可使用下面的命令:

```
COW IS THE TIME FOR ALL GOOD WOMEN TO COME TO THE AID OF THE HUNGRY.
~
~
~
~
~
: e foo_
```

如果用户在没有保存第一个文件的情况下使用了:e 命令,会得到下面的错误信息:

```
No write since last chance ("edit!" overrides)
```

上面的信息表明 vi 在用户保存第一个文件之前一般不能编辑另一个文件。这时,用户要用:w 命令来保存第一个文件,然后使用命令:e。用户也可以使用如下的命令:

```
COW IS THE TIME FOR ALL GOOD WOMEN TO COME TO THE AID OF THE HUNGRY.
~
~
~
~
~
: e! foo_
```

! 告诉 vi 用户希望在不保存第一个文件的情况下,对另一个文件进行编辑。

3.11.9 插入其他文件的正文

用户使用: r 命令可以将其他文件的正文插入到当前正在编辑的文件中。例如,命令

```
: r  foo.txt
```

可以将文件 foo.txt 的正文插入到当前文件光标所在的位置。

3.11.10 运行 shell 命令

用户可以在 vi 中运行 shell 命令。命令: r! 和命令: r 的作用是相似的。不同在于: r 命令后面的参数是个文件名, 它把文件的正文插入到光标所在的位置, 而: r! 命令后面的参数是个 shell 命令, 它把该 shell 命令的输出插入到光标所在的位置。例如, 如果用户使用下列的命令:

: r! ls -F

用户会得到下面的结果:

```
COW IS THE TIME FOR ALL GOOD WOMEN TO COME TO THE AID OF THE HUNGRY.
letters/
papers/
reports/
~
~
~
```

用户也可以在 vi 中执行 shell 命令, 并在命令完成后返回 vi 文本编辑器。例如, 如果用户使用命令

: ! ls -F

命令 ls -F 将被执行, 并且将结果显示在屏幕上。但命令的输出并不被插入到用户编辑的文件中。如果用户使用命令

: shell

vi 将启动一个子 shell 而暂时将 vi“挂起”, 然后在该 shell 下执行其他命令。用户可以使用 exit 命令从 shell 中退出并返回 vi。

3.11.11 获得帮助

vi 里并没有提供交互式的帮助功能(大多数 UNIX 系统的程序都没有提供这种功能), 但是用户可以阅读 vi 的联机用户手册来得到有关 vi 的用法。由于 vi 是 ex 文本编辑器的可视化前端, 是 ex 命令处理了 vi 下的命令行方式的许多命令, 所以在阅读 vi 的联机用户手册时, 也可以查阅一下 ex 命令联机的用户手册。

3.12 设置用户工作环境

shell 提供了许多机制来设置用户的工作环境。正如我们在前面讲到的, shell 不仅仅是一个命令解释器, 它还是一个功能强大的程序设计语言, 用户可以使用 shell 建立适合自己需要的专用工具和环境, 使其在 Linux 下的工作变得既简单又迅速。在概念上, shell 程序与 MS-DOS 中的批处理命令类似, 但 shell 语言更完备, 功能更强。

我们在前面说过, 不同的 shell 使用不同的语法。例如, Tcsh 使用类似于 C 的语法, 而 Bourne shell 使用另一种语法。在本节中, 我们不去讨论这两种 shell 的不同之处。我们的

目的是以一种 shell 为例,介绍如何使用 shell 来设置用户工作环境。所以,这里我们假设使用的是 Bourne shell。

3.12.1 shell 程序

假设下面是经常要使用的一组命令:

```
/home/zhang $ cat chapterl chapter2 chapter3 > book
/home/zhang $ wc -l book
```

这组命令完成的工作是将文件 chapterl, chapter 2 和 chapter3 连接到一起,并将结果放入文件 book 中,然后,在屏幕上显示出文件 book 的行数。如果我们希望将这组命令缩减成一个简短命令以减少键盘的输入,可以通过把这组命令组织到一个 shell 程序中来达到这个目的。

首先用前面讲过的 vi 文本编辑器来建立一个 shell 程序文件,并将其存到文件 makebook 中。

```
/home/zhang $ cat makebook
$ A shell stript to create the book
cat chapter1 chapter 2 chapter3 > book
wc -l book
/home/zhang $
```

文件 makebook 就是一个简单的 shell 程序.

在这个 shell 程序中,第一行是注释行,以字符“$”开头。shell 在执行时会忽略注释行,注释行主要为程序阅读者提供注释信息。其他行是要执行的命令。

在执行 shell 程序时,shell 将文件安排为标准输入,shell 从文件而不是从终端读取输入。

shell 程序有两种执行方法。一种执行方法是将 shell 程序的文件名作为 sh 命令的参数,由子 shell(sh)运行。如:

```
/home/zhang $ sh makebook
```

执行 shell 程序的另外一种方法是先把 shell 程序的文件的权限设置成可执行的,然后在系统提示符下直接运行该可执行 shell 程序文件。即:

```
/home/zhang $ chmod u+ x makebook
```

将使用用户获得 shell 程序 makebook 的执行权限。这时,用户可以直接使用下列命令运行该程序:

```
/home/zhang $ makebook
```

在对 shell 程序命名时,一定要注意:shell 程序的文件名不能与系统命令的名字相同。否则,系统将不再执行系统命令,而执行相应的 shell 程序,这将极大地干扰用户的工作。

3.12.2 shell 变量和环境

像大多数程序设计所用的语言一样,shell 允许用户定义变量。变量只是一个指定名

字的数据。

需要注意的,我们假设这里用的 shell 是 Bourne shell, 如 Bash。Tcsh 以及其他类似 C 的 shell 有另一套设置变量的机制,感兴趣的读者请查阅有关的联机手册。

用户可以使用赋值命令为变量赋值。变量赋值后,用户可以用变量名加前缀“$”来访问这个变量。例如以下命令:

```
/home/zhang$ foo="hello there"
```

为名为 foo 的变量赋值字符串“hello there”。这时,用户便可以用“$foo”引用变量 foo 的值。“$”字符会通知 shell,后面引用的是变量的值而不是变量的名或一般字符串。例如

```
/home/zhang$ echo $foo
hello there
/home/zhang$
```

与下列命令的结果是相同的:

```
/home/zhang$ echo "hello there"
hello there
/home/zhang$
```

当一个变量名字的后面紧接一些字符,而这些字符又不是这个名字的一部分时,应该用花括号或双引号将变量名括起来,使变量名与其后面的字符分开。如:

```
/home/zhang$ echo ${foo}!!
hello there!!
/home/zhang$
```

shell 的变量是属于 shell 内部的。也就是说,只有 shell 能够访问这些变量。shell 变量在书写 shell 程序时是非常有用的。例如,如果用户需要保存一个文件名,就可以把这个文件名存储到一个变量中。使用命令 set 会列出所有已定义的 shell 变量。

不过,shell 允许用户将 shell 的变量输出给环境。所谓环境是用户执行的所有的命令都能够访问的一组变量。当在 shell 中定义了一个变量后,可以将它输出到环境中以使其同时也成为环境的一部分,或称环境变量。export 命令的作用就是将 shell 变量输出到环境中去。(注意与 Bash 不同,Tcsh 使用 setenv 命令来设置环境变量,请参阅 Tcsh 的联机手册。)

环境变量对 UNIX 系统是非常重要的。它允许用户通过设置一个命令所需要的环境变量来设置这个命令。

下面举一个例子。命令 man 使用了环境变量 PAGER。这个变量的缺省值为 more。它指定 man 命令按分页方式显示手册内容。如果用户将 PAGER 设置为其他命令名,它就会使用设置的命令来显示手册内容,而不使用 more 命令。

如果将 PAGER 设置成“cat”,这会使命令 man 一次将所有输出都显示到屏幕上,而不进行分页。例如,

```
/home/zhang$ PAGER="cat"
```

然后,将 PAGER 传播到环境中。

/home/zhang$ export PAGER

这时,使用命令 man ls,联机手册内容会在屏幕上一闪而过。

现在,如果将 PAGER 设置成"more",more 命令就会被用来显示手册。

/home/zhang$ PAGER="more"

这次在修改 PAGER 的值后就不再需要使用 export 命令了。我们只需要对每个变量进行一次输出;之后对这个变量所进行的所有修改都会自动传播到环境中。

每个命令的联机手册会告诉用户该命令是否使用了环境变量。例如 man 命令的联机手册就指出该命令使用变量 PAGER 来指定分页命令。有一些命令共享环境变量。例如许多命令都使用环境变量 EDITOR 来指定使用的缺省的文本编辑器。

环境的另一个作用是用来保存用户每个登录的一些重要信息。如环境变量 HOME 保存着用户主目录的名字。

/home/zhang/report$ echo $HOME

/home/zhang

/home/zhang/reports$

最重要的环境变量之一是 PATH。当用户通过全路径名给出一条命令时,如下列命令:

/home/zhang$ /bin/ls

shell 按全路径名找到该命令的可执行文件并执行之。但当用户没有给出全路径名时,如命令:

/home/zhang$ ls

shell 是如何找到要执行的命令 ls 的呢? shell 是使用环境变量 PATH 来查找用户键入命令对应的可执行文件的。PATH 变量提供了一个目录清单,shell 依次到该清单所列的各目录中搜索未用全路径名给出命令。用户可以使用 echo 来查看 PATH 变量。

/home/zhang$ echo $PATH

/bin:/usr/bin:/usr/local/bin:.

可以看到,PATH 变量的值是一个目录清单,每个目录用冒号":"分隔开。用户未给出全路径命令名时,shell 将按从左到右的顺序对这些目录依次进行搜索,直到发现指定命令为止。如搜索其中的所有目录(包括当前目录)仍未发现给出的命令,shell 将给出错误信息。在上例中,当用户运行 ls 命令时,shell 会首先查找/bin/ls,依次类推。

PATH 的使用会使用户节省许多时间。这样就不需要记住所有可执行命令的驻存地。在许多系统中,可执行程序分散在许多不同的目录中,如/usr/bin,/bin 或 /usr/local/bin。用户可以将任何要求 shell 自动搜索的目录加到 PATH 变量的目录表中去。

这里需要指出的是 PATH 与查找一般文件无关,只用于命令的查找。例如,执行命令

/home/zhang$ cp foo bar

时,shell 只在当前目录中寻找文件 foo 和 bar。shell 已经假定 cp 命令给出的是文件的全名。

注意,在 PATH 变量中,往往包含了一个句点".",它代表当前工作目录。这就允许用户在当前工作目录中建立一个程序(包括 shell 程序),并像运行一个命令那样运行它,而

不需要指明路径(./makebook)。在 PATH 中也可以用额外的冒号来表示当前工作目录,如下面的两个例子:

PATH=:/bin:/usr/bin:/usr/local/bin

PATH=:/bin:/usr/bin:/usr/local/bin

对于不在 PATH 的目录清单中出现的目录,shell 将不会对其进行搜索,包括当前工作目录。

当用户登录到系统后,PATH 变量通常已被设置。只有当用户希望改变缺省设置时,才需要对其进行重新设置。

常用的 shell 环境变量有:

HOME: 定义用户的主目录的名字。

PATH: shell 查找目录时使用的查找路径。

IFS: 内部字段的分隔符,通常是空格、制表符或换行符。这个内部字段的分隔符将目录按字段分开。

PS1 和 PS2: shell 使用的基本提示符。其中 PS1 是 shell 使用的正常提示符,PS2 要求额外的输入,以便完成那些明显是不完整的命令。

TERM: 终端的型号,有些命令,如 vi,需要知道所使用的终端型号,以便产生正确的输出。

3.12.3 shell 初始化文件

除了用户建立的 shell 文件外,还有一些 shell 自己使用的特殊功能的 shell 文件。其中最重要的是用户的初始化文件,这些文件由 shell 在用户登录时自动投入运行。

初始化文件一般是一些简单的 shell 文件,但是却是非常有用的。在用户登录时,通过 shell 自动地执行这些初始化文件来设置用户的工作环境和自动启动一些在登录后总要运行的一些命令。例如,如果用户在登录后总要使用 mail 来查看信件,用户就可以将这个 mail 命令放置到初始化文件中以使其自动启动。初始化文件使用户可以根据自己的意愿设置用户的环境变量。

不论是 Bash 还是 Tcsh,登录时执行的 shell 称为登录 shell。登录 shell 与其他后产生的 shell 是有区别的。登录 shell 是用户在登录时调用的 shell。通常情况下,用户只使用这个 shell。但是,当用户从另外一个程序转入到 shell 命令状态,如从 vi 中执行 shell 命令,就启动了一个不同于登录 shell 的新的 shell 例程。另外,当用户启动一个 shell 程序时,一个新的 shell 例程就被自动启动来运行这个 shell 程序。

Bash 使用的初始化文件包括:/etc/profile(这是一个由系统管理员建立,所有的 Bash 用户在登录时都会执行的文件)、$HOME/.bash_profile(由 Bash 的登录 shell 执行的文件)和 $HOME/.bashrc(由 Bash 的所有非登录 shell 例程执行的文件)。如果.bash_profile不存在,将由.profile 来代替它。

Tcsh 使用下列初始化文件:/etc/csh.login(由所有 Tcsh 用户在登录时执行的文件)、$HOME/.tcshrc(在登录时被执行,而且所有 Tcsh 的新的 shell 例程也执行这个文件)和 $HOME/.login(登录时,在.tcshrc 之后执行)。如果.tcshrc 不存在,将由.cshrc

代替它。

3.12.4 X 窗口系统初始化文件

大多数用户喜欢在图形环境中进行工作。对 Linux 系统来说,图形环境就意味着使用 X。既使用户已经对 MS Windows 或 Macintosh 很熟悉,对 X 窗口系统也还需进一步熟悉一下,特别是对它的配置。

对 MS Window 或 Macintosh 系统,可以在图形环境内来配置环境。例如,如果希望改变背景,只需在一些特定的图形设置程序中用鼠标键选择新颜色。但在 X 中,系统的缺省设置是由一些文本文件控制的,这些文件需用户直接进行编辑。换句话说,用户要设置背景颜色,只能将颜色的名字输入到一个文件中。用户要想自如、准确地配置自己的 X 系统,就需要掌握这些文本文件。

配置 X 的一些重要文件是:

.xinitrc　　在 X 启动时运行的文件

.twmrc　　由 X 窗口管理程序 twm 执行

.fvwmrc　　由 X 窗口管理程序 fvwm 执行

如果这些文件存在,它们应存放在用户的主目录下。

.xinitrc 是 X 启动时投入运行的一个简单的 shell 文件,它能够做任何其他 shell 文件可以做的工作。当然,它主要是用来启动各种 X 程序并设置窗口系统参数。

通常.xinitrc 文件中包含有下列内容:一些对 xsetroot 程序的调用,这些调用用来设置用户的根窗口(背景)的颜色和鼠标光标的形状;一些对 xmodmap 的调用,这些调用告诉服务器如何解释键盘输入的信号;另外还有用户希望每次启动 X 时运行的程序(如,xclock)。.xinitrc 文件中最后一个命令通常是要运行的窗口管理程序的名称,如/usr/bin/X11/twm.。

X 启动后,首先要启动窗口管理程序。窗口管理程序启动后,它将自动读自己的初始化文件,该文件所控制的内容是:菜单如何启动;各窗口的位置;图标控制和其他一些重要问题。如果使用的窗口管理程序是 twm,则其初始化文件是用户主目录下的.twmrc;如果使用 fvwm 作为窗口管理程序,则初始化文件为.fvwmrc。

用户可以在/usr/lib/X11/twm 或/usr/X11/lib/X11/twm 目录下,找到.twmrc 的一些例子.

同样,用户也可以在/usr/lib/X11/fvwm 或/usr/X11/lib/X11/fvwm 目录下,找到.fvwmrc的一些例子。

3.12.5 其他的初始化文件

在 Linux 中,还可以使用下列初始化文件来配置用户的工作环境:

.netrc　　为 ftp 提供缺省的登录用户名和口令

.rhosts　　使用户的帐户可以被远程访问

.forward　自动收转 mail

1. FTP 缺省值

.netrc 文件允许用户在运行 ftp 前特定的 ftp 的缺省值。举一个简单的.netrc 文件的例子:

```
machine   intelp01. tsinghua. edu. cn   login   zhang   password   zhang
machine   intelp02. tsinghua. edu. cn   login   zhang   password   zhang
machine   intelp03. tsinghua. edu. cn   login   zhang   password   zhang

machine   clone. mcs. anl. gov          login   fogel   password   zhang@
machine   osprey. mcs. anl. gov         login   fogel   password   zhang@

machine   sunsite. unc. edu   login   anonymous   password   zhang@tirc. dcs. tsinghua. edu. cn
```

.netrc 文件中的每一行都指出了一个机器名,该机器的一个缺省登录帐户和口令。当用户需要经常使用 ftp 时,这个文件将提供很大的便利。它使得用户不需每次都敲入不同站点上的用户名和口令。如果用户 ftp 到一个.netrc 中存在的机器,ftp 程序将使用在.netrc文件中找到的信息自动地进行登录.

当用户需要忽略.netrc 文件而不进行自动登录时,需在 ftp 命令中使用-n 选项,即“ftp -n”。

使用.netrc 文件时,一定要确保.netrc 文件只对用户自身是可读的,以防止其他用户盗走自己的口令。用户要使用 chmod 命令来设置文件的只读权限。当.netrc 的权限设置不正确时,ftp 以及其他使用.netrc 的程序将拒绝工作。

有关.netrc 的更多用法,请参阅 ftp 的联机手册。

2. 远程访问用户帐户

如果在用户的主目录中有.rhosts 文件,就可以远程运行该机器上的程序。当用户在 intelp01. tsinghua. edu. cn 的机器上有一个正确配置的.rhosts 文件时,就可以在另一台机器上如 intelp02. tsinghua. edu. cn 上运行 intelp0l. tsinghua. edu. cn 机器上的程序,而不需进行登录和输入口令。这时,输出将被送到当地的 intelp02. tsinghua. edu. cn 上。

例如,intelp0l 机器上用户 zhang 的.rhosts 文件如下:

```
intelp02. tsinghua. edu. cn      zhang
intelp03. tsinghua. edu. cn      zjy
intelp04. tsinghua. edu. ch      zhang
```

.rhosts 的格式是很清晰的:每一行是一个机器名及其后面的用户名。有了上面的在 intelp01. tsinghua. edu. cn 的机器上有了上面的.rhosts 文件后,当用户在文件中所列的机器上用指定的用户名登录后,就可以远程运行 intelpo1 机器上的程序,而将结果输出到登录使用的机器上。

通常用户使用 rsh 程序来运行远程程序。rsh 的意思是“远程 shell”(remote shell)。它能在远程的机器上启动一个 shell 并执行指定的命令。例如:

```
intelp02$ whoami
zhang
```

intelp02$ rsh intelp01.tsinghua.edu.cn“ls ～”

Mails/ letters/ papers/ reports/

intelp02$

intelp01.tsinghua.edu.cn 上的用户 zhang 允许 intelp02.tsinghua.edu.cn 上的用户 zhang 在 intelp01 上远程运行自己的程序。

当各计算机上的用户名不同时,.rhosts 也能正确地工作。这时,需要在 rsh 命令的命令行中使用“-l”选项来指明远程机器上所使用的用户的名字。如果用户希望运行远程机器上不同用户名的用户的程序,当远程机器中存在这个用户,并且远程用户在自己的.rhosts 文件中包含了用户当前所使用的机器名和用户名,那么就可以用带有“-l”选项的 rsh 来远程运行其上的程序。

如 intelp03 机器上的用户 zjy 可以执行 intelp01 机器上用户 zhang 的程序:

intelp03$ whoami

zjy

intelp03$ rsh -l zhang intelp01.tsinghua.edu.en "ls ～"

Mails/ letters/ papers/ reports/

intelp03$

.rhosts 文件中还可以有其他格式。如,用户可以在.rhosts 文件的远程机器的名后面不指定用户名,这时相应的远程机器上的所有用户都可以运行该用户的程序了!这当然是很危险的:其他用户可能会远程地删除你的文件。所以,如果用户在.rhosts 文件中没有指定用户名,那么用户应该保证.rhosts 文件只对用户自己是可读的,其他用户都没有对该文件的读权限。

3. 收转 mail

用户在自己的主目录下可以建立.forward 文件来自动收转 mail。.forward 文件不是严格意义上的“初始化文件”。如果该文件中包含了 mail 地址,那么所有发送到该用户的 e-mail 都将被转发到这个 e-mail 地址中,这一点是非常有用的。

我们在上面所列出的只是一些经常用到的其他初始化文件。初始化文件是根据系统的不同而不同的,而且依赖于系统上所安装的软件。用户可以查看主目录下以句点“.”开头的文件来了解更多的初始化文件。当然,这些文件中有些不是初始化文件,但大多数是。

3.13 与其他用户进行通信

Linux 是个功能强大的系统。两个不同的 Linux 系统的机器之间可以通过不同的方法交换信息。在本节中我们将向读者介绍如何使用这些网络功能来进行用户之间的联系。

3.13.1 电子邮件

Linux 最受欢迎的一个特性就是电子邮件。在 Linux 中有许多程序都具备电子邮件的功能。在所有这些程序中,最常用的当推 mail 命令。这是一个既可以发送又可以接收的交互式实用程序。Linux 的邮件系统在配置上虽不尽相同,当 mail 命令几乎总能运转。

1．发送电子邮件

要发送电子邮件，需要在 mail 命令中使用要发给邮件的用户名作为参数。然后用户要输入要发送的信息。用户所输入的信息要以 EOT（通常以 Ctrl-D 开头的行表示）为结尾。

下面举一个例子。假如用户 zhang 希望给同一系统的用户 wang 发送一个电子邮件。

```
/home/zhang$ mail  wang
Subject: The user documentation
Just testing out the mail system.
EOT
/home/zhang$
```

当用户向不在同一系统上的用户发送电子邮件时，在 mail 命令行的用户名后面要指定该用户所在机器的主机名或 IP 地址，如“用户名@主机名”后“用户名@IP 地址”。

mail 程序很简单。它就像 cat 一样，从标准输入一行一行地接收输入，一直到文件结束符 EOT 为止。

像上例那样输入电子邮件内容的一个缺点是：用户在键入回车符后，如果发现所输入的内容有错，修改起来很麻烦。可以使用管道或标准输入重新定向的方法来解决这个问题。例如，可以事先使用文本编辑程序编辑所要发出的电子邮件，如文件 report1 中包含了用户 zhang 要发送给用户 wang 的信件，这时，用户 zhang 可以使用命令“mail wang < report1”或命令“cat report1©¦mail wang”来发送这个信件。

2．读电子邮件

当 mail 命令行中没有指定参数时，它的功能是阅读电子邮件的内容。这时，该命令会检查信箱，如果信箱中有电子邮件，便列出每一个电子邮件的邮件头的一行；否则，会给出提示信息。我们看下面的例子：

```
/home/zhang$ mail
No mail for zhang
/home/zhang$
```

这说明当前信箱中没有电子邮件。用户可以试着给自己发两个电子邮件：

```
/home/zhang$ mail zhang
Subject: test
this is only a test for e-mail.
EOT
/home/zhang$ echo "hello"©¦mail zhang
/home/zhang$ mail
Mail version 5.5 6/1/90. Type ? for help.
"/usr/spool/mail/zhang": 2 message 2 new
>N  1  zhang     Tue  Apr  16  14:00  10/211  "test"
 N  2  zhang     Tue  Apr  16  14:02  9/191
&
```

mail 程序中的提示符是“ & ”。在这个提示符下可以运行一些简单命令。用户可以使用“ ? ”得到有关的帮助信息。

mail 中的基本命令是:

t < 信件标识> 在屏幕上显示信件内容。

d < 信件标识> 删除信件。

s < 信件标识> 将信件保存到文件中。

r < 信件标识> 回信——也就是说,给发出相应信件的人写回信。

q < 信件标识> 从 mail 中退出,并将用户已经阅读过且没有被删除的信件保存到用户主目录中的 mbox 文件中。

x < 信件标识> 从 mail 中退出,并不对信箱中的信件进行附加操作。

在上面的命令中,< 信件标识> 可以是以空格分离的数字列表或类似 2-4(其中包含 2、3、4) 的序列。此外,用户也可以使用发信者的用户名来指定信件。如果没有在命令中给出< 信件标识> ,则缺省为被显示的最后一个信件。

除 mail 命令外,用户还可以使用 pine、elm、emacs 等程序来发送和接收电子邮件。这里就不一一介绍了。

3.13.2 用户之间的对话

用户可以使用 talk 命令与其他用户进行对话。talk 命令是一个可视的程序,它将用户终端上的信息行复制到与之进行通话的用户的终端上。

talk 命令的用法是:

talk < 用户名> [终端号]

如果用户希望与同一机器上的用户进行对话,则上面命令行中的< 用户名> 就是指用户进行登录的用户名。一个用户希望与其他机器上的用户进行对话,则< 用户名> 用下面的形式给出:“ 用户名@主机名 ”或“ 用户名@IP 地址 ”。

当用户希望与之进行对话的用户同时登录多次时,可以使用终端号来指明所要进行对话的终端。

intelp01 机器上的用户 zhang 要与同一机器上的用户 wang 进行对话,可以使用下列命令:

```
/home/zhang $ talk wang
```

talk 会首先将下列信息送到对方的屏幕上:

```
Message from TalkDeamon@intelp01 at 17: 33...
talk: connection requested by zhang@intelp01
talk: respond with: talk zhang@intelp01
```

这时,被呼叫的用户 wang 要使用下列命令进行应答:

```
/home/wang $ talk zhang@intelp01
```

当对话通路被建立后,两个用户就可以同时进行输入,双方的所输入的信息会同时在双方的屏幕上两个分开的窗口内显示出来。可以使用中断键退出 talk; 这时,talk 会将光标移到屏幕的底部,并使终端恢复它的原来状态。

用户可以使用 mesg 命令来拒绝或恢复对话。

3.14 Linux 系统的基本命令

本节中我们介绍一些常用的 Linux 的基本命令,其中包括前几节中已经介绍过的命令。

在我们给出的命令语法格式中,尖括号内是命令行的参数,方括号内是命令行的可选参数。

注意:命令中的选项以“-”开始。一般情况下,多个单字符的选项用一个“-”合并到一起。如命令 ls -l -F 用 ls -lF 代替。

我们没有一一列出这些命令的所有选项,只讨论那些非常有用并且重要的选项。实际上,这些命令中多数有大量的选项,其中许多选项几乎从来没有用到过。用户可以使用 man 命令来看每个命令的联机手册,联机手册中会列出命令的所有选项。

还要说明的是,这些命令中有许多命令以文件或目录的列表(即多个文件或目录)为参数,如文件的列表用 < 文件 1> …< 文件 n> 表示,这个列表代表一个或一个以上的文件。例如,cp 命令的第一个参数是一个被复制的文件的列表,第二个参数是目录或文件。当复制多个文件时,第二个参数一定要是目录。

cat

功能:cat 一般用来串接文件,有时也被用来没有停顿地显示文件的所有内容。

语法:cat < 文件 1> < 文件 2> …< 文件 n>

说明:这里< 文件 1> 到< 文件 n> 是指要被显示或串接的文件。当对几个文件进行串接时,要使用标准输出重定向符将指定的文件串接到一个新文件中。

举例:cat letters/from-wang

该命令将显示出文件 letters/from-wang 的内容。

cat papers/history-paper papers/math-paper > newpaper

该命令将把文件 papers/history-paper 和 papers/math-paper 串接到一起,形成新文件 newpaper。

cd

功能:改变当前工作目录。

语法:cd [目录名]

说明:[目录名]是要进入的目录。“.”表示当前目录,“..表示父目录。当该参数缺省时,表示回到用户的主目录。

举例:cd ../reports 将当前目录移至其父目录的 reports 子目录。

chgrp

功能:改变用户组对文件的所有权。

语法:chgrp [选项] < 用户组> < 文件 1> < 文件 2> . . . < 文件 n>

说明:该命令将用来改变用户组对所有指定文件的所有权。< 用户组> 可以为用户组的名称,也可以为组号。

常用的选项有:

-R　递归地修改目录及其所属内容的所有权。

-f　当文件的所有权不能被修改时,该命令将不列出错误信息。

chmod

功能: 修改文件的访问权限。

语法: chgrp [选项] < 权限格式> < 文件 1> < 文件 2> . . . < 文件 n>

说明: 该命令根据所指定的权限格式来改变指定的文件序列的访问权限。< 权限格式> 可以是符号形式,也可以是新权限的八进制串。

符号形式为: [ugoa...][[+ -=][rwxXstugo...]...]。用户可以在一个命令中给出多个这种符号表达式,其间用逗号隔开。

[ugoa...]指明修改权限的用户。其中 u 代表文件的所有者; g 代表同组用户; o 代表其他用户; a 代表所有用户。

[rwxXstugo...]代表所修改的内容。其中 r 代表读权限; w 代表写权限; x 代表执行权限; X 代表只有当指定文件是目录时或某些用户已经具有它们的可执行权限时是可执行的; s 代表在执行时设置用户或用户组的标识号; t 代表在交换设备上保存程序的正文; u 代表用户当前所拥有的权限; g 代表同组用户当前所拥有的权限; o 代表其他用户当前所拥有的权限。

chown

功能: 改变用户和用户组对文件的所有权。

语法: chown [选项] < 用户> [: .][用户组] < 文件 1> < 文件 2> …< 文件 n>

说明: 该命令将用来改变用户和/或用户组对所有指定文件的所有权。如果除选项外的第一个参数只指明了< 用户> ,则该命令将所列出的各个文件的所有者改成这个用户,而文件的用户组不变; 如果< 用户> 后面紧跟着冒号(或句点)和用户组,则同时还将这些文件的用户组改变为指定的用户组; 如果< 用户> 后面紧跟着冒号(或句点)而没有用户组,则将这些文件的用户组改变为指定用户所在的登录用户组。

常用的选项用:

-R　递归地修改目录及其所属内容的所有权。

-f　当文件的所有权不能被修改时,该命令将不列出错误信息。

cp

功能: 将文件复制到另一个文件或目录。

语法: cp < 文件 1> < 文件 2>

cp < 文件 1> < 文件 2> …< 文件 n> < 目录名>

说明: 第一种格式是将以< 文件 1> 为名的文件复制到以< 文件 2> 为名的文件中。第二种格式中,< 文件 1> 到< 文件 n> 是要进行复制的文件名列表,cp 命令将这些文件复制到以< 目录名> 为名字的目录中,并保持文件名不变。

举例: cp shells test

当 test 是一个文件名时，该命令将当前目录下的文件 shells 复制到文件 test 中；但当 test 是当前目录中的一个目录名时，该命令将在 test 目录中建立一个 shells 文件的副本，并仍以 shells 命名。

echo

功能：复制命令行的参数。

语法：echo < 参数 1> < 参数 2> . . . < 参数 n>

说明：这里< 参数 1> 到< 参数 n> 是指要被复制的参数。

举例：echo“Hello world”

该命令将显示字符串“Hello world”。

grep

功能：显示出指定的文件中与指定的模式匹配的所有行。

语法：grep [选项] < 模式> < 文件 1> < 文件 2> …< 文件 n>

说明：这里 < 模式> 是指正则表达模式，而< 文件 1> 到< 文件 n> 是指要被搜索的文件。

该命令经常用到的选项有：

-c 输出行数，而不是行本身。

-l 列出包含与正则表达模式相匹配的正文内容的文件的名称。

-v 输出不与正则表达模式相匹配的所有行。

-n 在输出行前加上行号。

-y 正则表达模式中的小写字母将匹配文件中的大写或小写字母。

举例：grep loomer /etc/hosts

该命令将显示出文件/etc/hosts 中所有包含“loomer”的行。

ln

功能：为文件和目录建立链接。

语法：ln [选项] < 文件名> [链接名]

ln [选项] < 文件 1> < 文件 2> …< 文件 n> < 目录名>

说明：链接是一个用于参照另一个文件或目录的目录项。

该命令的第一种格式是为一个文件建立一个链接。

该命令的第二种格式是为< 文件 1> 到< 文件 n> 在以< 目录名> 为名的目录中建立链接，且仍然以原文件名为链接名。

所建立的链接的缺省形式是硬链接。且在建立链接时缺省情况是不删除已经存在的文件。

经常用到的选项是：

-s 为文件建立符号链接。

-v 在建立链接前打印出每一个文件的名字。

-I 在建立链接时向用户提问是否删除已经存在的文件。

-f 在建立链接时删除已经存在的文件。

ls

功能: 列出指定的目录中的文件或指定的文件的信息。

语法: ls [选项] [名字 1] [名字 2]. . .[名字 n]

说明: 这里[名字 1]到[名字 n]是指要列出的目录名或文件名。当 ls 命令行中没有指定所列出的目录或文件时,该命令将列出当前目录中的内容。

[选项]代表 ls 目录的选项,这些选项可以用于控制每个文件所要列出的信息,以及所列文件清单的顺序。

最常用到的选项是:

-F 列出有关文件类型的信息。

-l 产生长格式的列表,可以列出文件的大小、所有者、文件的访问权限等。

-a 列出指定目录下的全部文件,包括一句点“.”开头的隐藏文件。

-t 按文件的修改日期而不是按字母顺序来列出文件清单。

-r 将输出的顺序倒过来。

举例: ls -F /home/zhang

该命令会列出目录/home/zhang 目录下的所有内容及相应的类型。

man

功能: 打印出指定命令或资源(资源是指系统中除命令以外的其他工具,如库函数等)的联机手册条文。

语法: man [选项] [节] < 命令>

说明: 这里< 命令> 是指欲查询的命令或资源的名称。语法形式中的[节]指定在哪一节查找指定的资源。有关命令的手册信息一般在第一节。若没有指定节,则 man 命令会查找手册中的所有节。

该命令常用到的选项有:

-k 按关键字查询相应命令或资源

-w 列出指定的命令或资源对应的手册所在的路径名。

-t 使用 troff 正文格式加工程序,产生适合于照相排版的输出。

-n 使用 nroff 正文格式加工程序在标准输出上产生输出。

举例: man ls

该命令将列出 ls 命令的所有手册信息。

man -k ps

该命令将列出 Linux 中所有包含单词“ ps”的命令、系统调用以及资源的手册信息。

mkdir

功能: 建立新目录。

语法: mkdir < 目录 1> < 目录 2> . . . < 目录 n>

说明: 这里< 目录 1> 到< 目录 n> 指要建立的目录。

举例: mkdir /home/zhang/test

该命令将在/home/zhang 目录下建立新目录 test。

more

功能: 显示出指定文件的内容, 每次显示一屏。

语法: more < 文件 1> < 文件 2> . . . < 文件 n>

说明: 这里< 文件 1> 到< 文件 n> 是指要被显示的文件。

举例: more papers/history-paper

该命令将分页显示 papers/history-paper 文件的内容。

mv

功能: 将文件移到另一个文件或目录。这个命令与先复制一个文件, 再将源文件删除的操作是等效的。

语法: mv < 文件 1> < 文件 2>

mv < 文件 1> < 文件 2> …< 文件 n> < 目录名>

说明: mv 用来管理文件, 在 mv 命令的第一种格式中, 以< 文件 1> 为名字的文件被重新命名为< 文件 2> 。

在 mv 命令的第二种格式中, < 文件 1> 到. < 文件 n> 是要被移动的文件名列表, mv 命令将这些文件移动到以< 目录名> 为名字的目录中, 并保持文件名不变。

举例: mv shells test

当 test 是一个文件名时, 该命令将当前目录下的文件 shells 更名为 test;

当 test 是当前目录中的一个目录名时, 该命令将文件 shells 移到 test 目录下。

rm

功能: 删除文件。注意: 在 UNIX(或 Linux)系统中, 用户只能删除那些具有写权限的文件, 且一个文件被删除后, 便不可能被恢复(这一点与 MS-DOS 不同。在 MS-DOS 中, 被删除的文件可以用“undelete ”命令恢复。

语法: rm [选项] < 文件 1> < 文件 2> …< 文件 n>

说明: 这里< 文件 1> 到< 文件 n> 代表要被删除的文件列表。

最常用到的选项是:

-i 这是一个用于交互的选项。这个选项使系统在删除指定文件之前, 向用户发出确认提示。

-r 这个递归选项用于删除文件系统中的一棵子树。这时文件名参数应该是目录名。这个命令的全部文件、子目录等等都将被删除。

-f 这个选项将使 rm 命令在不管用户是否具有这个文件的写权限的情况下, 强行删除文件。

举例: rm -i /home/zhang/reports/shells

该命令将交互地删除目录/home/zhang/reports 目录下的文件 shells。

rm -r /home/zhang/papers

该命令将删除/home/zhang/papers 子树。

rmdir

功能: 删除空目录。使用 rmdir 删除目录时, 用户的当前工作目录一定不能包含在

要被删除的目录中。用户必须具有要删除目录的父目录的写权限才能进行删除。

语法: rmdir < 目录 1> < 目录 2> …< 目录 n>

说明: 这里< 目录 1> 到< 目录 n> 指要被删除的目录。

举例: rkdir /home/zhang/test

当/home/zhang 下的目录 test 为空时,该命令将删除该目录。

passwd

功能: 改变用户的登录口令。

语法: passwd [用户名]

说明: 普通用户可以使用 passwd 命令来修改自己的口令,这种情况下,该命令没有参数;超级用户可以使用这个命令修改普通用户的口令,这种情况下,该命令用[用户名]来指定所修改的用户。

普通用户使用时,该命令会要求用户输入旧口令,以证实修改口令的用户的合法性;然后会提醒用户输入新的口令,并要求用户重新输入新密码,以保证新口令的正确输入。为增强安全性,口令在输入时不被显示在屏幕上。

ps

功能: 列出进程的状态信息。

语法: ps [选项] [进程标识符]

说明: ps 命令用于列出有关活动进程的信息。当 ps 命令没有选项时,它将列出下列信息:终端设备名字、进程号、累计执行时间、以及命令行的有关缩写。

ps 命令经常用到的选项有:

-l 列出进程的长格式信息。

-a 列出与控制终端有关的所有进程的信息,包括其他用户的信息。

-u 列出进程的用户格式信息:包括用户名和起始时间。

-x 列出与控制终端没有关系的进程的信息。

举例: ps -ax

该命令列出系统上所有进程的信息。

ps

该命令列出用户的当前进程。

pwd

功能: 列出当前工作目录。

语法: pwd

说明: pwd 命令将列出当前目录的绝对路径名。

tee

功能: 将标准输入的信息写入标准输出以及指定的文件中。

语法: tee [选项] [文件 1] [文件 2]. . .[文件 n]

说明: 当该命令行中没有指定文件时,该命令只将标准输入写入标准输出;否则,该命令同时将标准输入写入指定的文件。tee 命令经常被用在管道线中作为

T 型接口，这样，用户既可以在终端上看到一个程序的输出，又可以将这个程序的输出保存在一个文件中。

该命令可以有下列选项：

-a 当将标准输出写入文件中时，若该文件已存在，所写入的内容附加在原文件的末尾，而不是将原文件覆盖。

-i 该选项可使命令 tee 不理会中断。

举例：ls ©¦tee flist

该命令将把 ls 命令的输出保存在文件 flist 中，并同时将其显示在屏幕上。

tty

功能：列出终端的特别文件名。

语法：tty

说明：该命令列出当前连到标准输入的终端的文件名。如果标准输入不是终端，则该命令会给出类似于“not a tty”的信息。

wc

功能：对文件中的字、行、字符进行计数。

语法：wc [选项] < 文件 1> < 文件 2> . . . < 文件 n>

说明：该命令用于在正文文件中统计正文的单元。通常，它报告指定文件的行、字以及字符的数目。如果命令行中没有指定文件，则指标准输入。

该命令常用的选项是：

-w 对字数进行统计。

-l 对行数进行统计。

-c 对字符数进行统计。

who

功能：列出当前正在使用系统的用户名。

语法：who

说明：who 命令将列出当前系统中已经登录全部用户清单，其中包括用户名、登录时间以及某个用户的终端号。

专门命令“who am i”（也可用作“whoami”）通常产生用户自己的登录用户名。

第4章　系统管理

本章是Linux系统管理的概述。系统管理非常重要,而且常常是耗时的工作,即便你是系统中唯一的用户。这里我们只介绍能使你的系统正常运转的最重要和最基本的操作,而忽略一些技术上的细节,因而可使你很容易地将Linux运转起来。假如你希望它运行得更好,就需要进一步阅读附录A中列出的参考资料,它们会有助于你更好地理解系统的运转及组织机制。

4.1　普通用户和超级用户

Linux是多用户操作系统,每个普通用户有一个帐户,包括用户名、home目录等一些信息。除此之外还有一些系统指定的特殊用户,它们有一些特权,其中最重要的是超级用户,在Linux中称为root用户。下面分别作详细的介绍。

4.1.1　root用户

从系统安全性考虑,Linux操作系统对普通用户在使用系统的权限上有一定的限制,特别是对系统的操作通常都有限制,以防止它对系统或对其他用户造成损害。用给文件加访问权限的办法可以使一个用户不能在不被允许的情况下读、写或删除其他用户私有的文件和目录以及为所有用户所共享的文件和目录(例如在/bin和/usr/bin目录下的文件)。每个用户可以对自己的文件赋予适当的权限使得其他用户不能访问或修改这些文件。

但是对于root用户来说没有这样的限制,root用户不仅可以读、写或删除系统中的任何文件,而且可以修改文件的权限及所有者。此外,root用户还可以运行一些特殊的程序,如给磁盘分区、建立文件系统等。

系统管理员以root帐户登录到系统中,可以完成一些普通用户不能完成的任务。由于root能够做任何事情,因而在这个帐户下,很容易由于操作失误而造成灾难性的后果。例如,作为一般的用户,如果你不小心使用了删除/etc目录中的所有文件的命令,系统将不允许你这样做。但如果你以root帐户登录,系统将对你的操作不加任何限制,因此使用root帐户时,稍有不慎就很容易会使你的系统崩溃,避免事故的最好的方法如下:

在多用户系统下不让普通用户通过root帐户登录,只允许系统管理员用root登录。这一限制可以通过设置口令来实现。

- 在命令行状态下,对于输入的每个可能导致系统损坏的命令,在最后键入return键之前,仔细重读一下整个命令,确认它是正确的。
- 不要习惯于使用root帐户。如果你越习惯于在root帐户中操作,就越可能把你的权力和普通用户的权力混淆。比如当你自认为是以larry帐户登录时,可能实际上

是以 root 帐户登录的。

- 为 root 帐户提供一个不同于普通用户帐户的提示符。这时需要修改 root 帐户下的 .bashrc 或 .login 文件,把 shell 提示符设置成不同于你的普通帐户的提示符。例如,普通用户使用"$"提示符,root 用户使用"#"提示符。
- 只有在完全必要的情况下才使用 root 帐户登录。一旦你作为 root 用户完成所需工作,就要及时退出。你使用 root 帐户越少,那么你对系统造成损害的可能性就越小。

实际上,也有一些人使用 root 帐户做每一件事,但是他们每个人都曾经犯过愚蠢的错误,以致使系统瓦解。一般的规则是,在你能够熟练地在 root 帐户下准确无误地操作之前,尽量少地登录进 root 帐户。

当然每一个人都可能犯错误,Linus Torvalds 自己也曾不小心删除了他的系统中的核心目录树,几个小时的工作永远丢失了。但幸运的是,由于他在文件系统的实现代码方面的知识,使他能够重新启动系统,并手工重建硬盘上的目录树。

4.1.2 处理误用系统的用户

Linux 安全性设计得并不严格。系统的安全性是后来加上去的,因为在最初开发系统时并没考虑到一个用户会侵扰其他用户。即使有安全性措施,普通用户还是有危害系统的可能。

系统管理员可以采取两种态度来对付误用系统的用户:或偏执或信任。而偏执通常只会给系统管理员带来更多的麻烦。实际上,对于大多数用户来说,他们并没有足够的能力或知识有意对系统造成危害。当一个用户给系统带来麻烦时(例如,用大文件占满用户的分区或运行一个大程序的多个实例),90% 的情况是由于他们并没有意识到自己的所作所为会造成什么样的后果。

当你处理引起潜在麻烦的用户的时候,不要谴责他们,最好是和他们心平气和地交谈,讨论挽回损失的方法,而不要引起对抗。最后你应做的是处在用户的位置上替他们考虑一下,如果用户确信你不信任或不喜欢他们,他们也许会通过在系统上删除文件或突破各种防线来难为你,这当然不是你愿意看到的。

如果你确实发现一个用户试图破坏系统或正在有意破坏系统,不要恶狠狠地对待他们。相反,只需采用灵活的方式警告一下他们,不要再让同样的事情发生。但是,如果你第二次又抓到他们在搞破坏,那么毫无疑问他们是故意的。

4.2 启 动 系 统

有以下几种方法可以从软盘或硬盘启动系统。

4.2.1 使用 boot 软盘启动

boot 软盘包含 Linux 系统内核代码的一个拷贝,它可用来启动 Linux。Linux 的 root 分区被编码进内核中,所以它能知道到哪里去找 root 文件系统所在的硬盘(rdev 命令可

用于在内核映象中设置 root 分区)。

为建立你自己的 boot 软盘,首先确定内核映象在你的硬盘中的位置,它应该在文件 /Image、/ect/Image 或 /vmlinux 中。

你也可以使用一个压缩的内核映象。在启动时,压缩的内核将它自己解压缩到存储器中,压缩内核可以减少占用的硬盘空间。如果你有一个压缩内核,它应在文件 /zImage 或 /ect/zImage 中。

当你知道内核映象文件的位置时,可以用 redev 命令将内核映象中的 root 设备设置成 Linux root 分区的名字,命令的格式是

redev < kernel-name> < root-device>

< kernel-name> 是内核映象的名字,< root-device> 是 Linux 的 root 分区的名字。例如,如下的命令是将内核 /etc/Image 中的 root 设备设置成 /dev/hda2:

rdev /ect/Image /dev/hda2

rdev 还能够设置内核中的其他选项。例如设置在启动的时候使用默认的 SVGA 模式。使用"rdev -h"可以得到 rdev 命令的帮助信息,设置完 root 设备之后,你可以将内核映象拷贝到软盘上。每次要把数据拷贝到软盘时,最好先用 MS-DOS 格式化软盘,从而可以将扇区和磁道信息记录到软盘上,这样它可以被检测出是高密度的还是低密度的。

例如,如下命令可将文件 /ect/Image 中的内核映象拷贝到软盘 /etc/fd0 中:

cp /ect/Image /dev/fd0

现在这个软盘就可启动 Linux 了。

4.2.2 使用 LILO 启动

另一种启动系统的方法是使用 LILO。它是一个可以驻留在你的硬盘的引导(boot)扇区的程序,当系统从硬盘启动时,将执行这个程序。它能够自动使用存储在硬盘上的内核映象来启动 Linux。LILO 同时也可引导启动其他几种操作系统。它可允许你在启动系统时,选择要启动的操作系统(如 Linux 或 MS-DOS),当你使用 LILO 启动时,默认的操作系统将被启动,除非你在启动过程中按一下 Ctrl、Alt 或 Shift 键。如果你按下上述三个键中的任何一个,系统将打印出一个启动提示,提示你输入要启动的操作系统名(例如输入"Linux"启动 Linux 或输入"msdos"启动 MS-DOS)。如果这时你按下 Tab 键,可以启动的操作系统将被列出。

安装 LILO 的简单方法是编辑配置文件 /etc/lilo.conf,然后运行如下命令。

/sbin/lilo

你想启动的每一种操作系统在 LILO 配置文件中都有与之相对应的一段描述。作为例子,下面列出的配置文件描述了一个在 /dev/hda1 上的 Linux 分区,以及 /dev/hda2 上的 DOS 分区的 LILO 启动信息。

```
# Tell LILO to modify the boot record on /dev/hda (the first
# non-SCSI hard drive). If you boot from a drive other than
# /dev/hda,
# change the following line.
```

```
boot = /dev/hda

# Name of the boot loader. No reason to mondify this unless you're
# doing some serious hacking on LILO
install = /boot/boot.b

# Have LILO perform some optimization.
compact

# Stanza for Linux root partition on /dev/hda1.
image = /etc/Image          # Location of kernel
  label = linux             # Name of OS (for the LILO boot menu)
  root = /dev/hda1          # Location of root partition
  vga = ask                 # Tell kernel to ask for SVGA modes at
                            # boot time

# Stanza for MSDOS partition on /dev/hda2.
other = /dev/hda2           # Location of partition
  table = /dev/hda1         # Location of partition table for
                            # /dev/hda2
  lable = msdos             # Name of OS (for boot menu)
```

配置文件中的第一个操作系统段将是 LILO 要默认启动的操作系统。可以按如前所述的方法在 LILO 提示行下,选择另外一种操作系统。记住每次当你更新盘上的内核映象时,应该重新运行 /sbin/lilo 以使变化反映到你的硬盘的启动扇区。

另一方面还要注意,如果你在上面的文件中使用"root = "行,则没有必要使用 rdev 命令来设置内核中的 root 分区。因为 LILO 在启动时已经设置了它。Linux FAQ(见附录 A)提供了有关 LILO 的更多的信息,包括怎样使用 LILO 来引导启动 OS/2 的启动管理器。

4.3 关闭系统

关闭 Linux 系统要复杂一些。记住当系统正在运行时不要直接关上电源,或按 reset 键。这是因为,内核将最近访问的磁盘数据块保存在内存缓冲区中,如果你不通知内核将它的缓冲区中的数据写到磁盘上,就重新启动系统,那么将可能损坏你的文件系统。

在关机时也要采取其它一些预防措施:发送给所有的进程一个信号,使得它们正常地退出(即执行写和关闭所有文件等操作之后,再退出);为了安全性,文件系统被拆卸掉;也可以提示用户系统将被关闭的消息,给他们退出的时间。

最简单的关机方法是用 shutdown 命令。命令格式是:

shutdown < time> < warning-message>

< time> 说明关闭系统的时间(按照 hh: mm: ss 的格式), < warning-message> 说明关闭之前,在所有用户终端上显示的一个消息。

你也可将(time(说明为"now",也就是立刻关闭系统。-r 选项可以加到 shutdown 命令上,使得 shutdown 之后重新启动系统。例如,在下午 8: 00 关闭系统可以使用如下的命令:

shutdown -r 20: 00

halt 命令可用于立刻关闭系统而不给出任何警告信息或用户正常退出的时间。当你是使用系统的唯一的用户,而且要关闭系统时,可以用 halt 命令。在你看到如下信息之前不要关闭电源或重新启动系统。

The system is halted

使用 shutdown 或 halt 命令来关闭系统是非常重要的。在有些系统上,同时按下 Ctrl + Alt+ Del 键,将启动一个 shutdown 操作,而在另一些系统中,使用这样的操作将立即重新启动系统,就可能会造成损害。

4.4 管理用户

不管在你的系统中是不是有许多用户,了解 Linux 下的用户管理方面的知识还是很重要的。即使你是唯一的用户,也应该有一个自己的普通用户帐户(即除了 root 帐户之外的帐户,用来做你的大部分工作)。

使用系统的每一个人都应该有自己的帐户,许多人共享一个帐户并非明智之举。一方面安全性是个问题,另一方面你也需要跟踪记录谁在做什么。

4.4.1 用户管理的概念

系统针对每个用户保存如下信息:

用户名(username)	用户名是系统中每个用户的唯一标识,比如 larry、karl 及 mdw 都是用户名。它可以用字母、数字或字符构成的串表示,长度限制在 8 个字符以内。
用户 ID(user ID)	用户 ID 或 UID 是系统中每个用户的标识号,系统在保存用户信息时,通常用 UID 而不用用户名来标识一个用户。
组 ID(group ID)	组 ID 或 GID 是用户的默认组的 ID,在 3.9 节中我们讨论了组权限,一个用户属于哪个组是由系统管理员规定的,后面我们还要进一步讨论它。
口令(password)	系统保存用户的加密口令,passwd 命令用于设置和改变用户口令。
全名(full name)	用户的真名或全名是和用户名(username) 存在一起的,例如用户 schomj 实际的名字可以是"Joe Schom"。
home 目录	home 目录是用户在登录后最初所在的地方。每个用户都应有他们自己的 home 目录,它通常位于 /home 目录下。

登录 shell　　用户的登录 shell 是在登录时启动的 shell, 例如 /bin/bash 和 /bin/tcsh。

文件 /ect/passwd 包含用户的以上信息, 文件中的每一行对应一个用户的所有信息, 它的格式如下所示:

username: encrypted password: UID: GID: full name: home directory: login shell

下面是一个例子

kiwi: Xv8q981g71oKK: 102: 100: Laura Poole: /home/kiwi: /bin/bash

这里第一个域"kiwi"是用户名(username), 下一个域"Xv8q981g71oKK"是加密的口令, 口令不以任何人们可读的格式存放在系统中, 口令使用它自己本身作为密钥编成密码。换句话说, 就是只有知道口令, 才能解密。这种形式的加密是很可靠的。

有些系统使用遮蔽口令(shadow password), 就是将口令信息转移到文件 /etc/shadow 中, /etc/passwd 是任何人都可读的, 而 /etc/shadow 不是任何人都可读的, 因而也就提供了某种程度的额外的安全性。此外遮蔽口令还提供其它一些特性, 如口令终止等, 这里我们将不再详述。

第三个域"102"是用户 ID(UID), 这个符号对于每个用户来说必须是唯一的。第四个域"100"是组 ID(GID), 它表示这个用户属于编号 100 的组。组信息与用户信息类似, 保存在文件 /ect/group 中, 详见"4.4.5 组(groups)"。第五个域是用户的全名(full name)"Laura Poole"。后两个域分别是用户的 home 目录"/home/kiwi"和登录时的 shell (/bin/bash)。home 目录中的名字可以不同于用户名(username), 但是采用相同的名字有助于确定目录。

4.4.2　增加用户

如果要增加一个用户, 可以采取以下步骤: 首先在文件 /ect/passwd 中给用户设置一项, 填入用户信息: 指定唯一的用户名和用户 ID(UID), 规定用户的组 ID(GID)、全名(full name)和其它信息。然后建立用户的 home 目录, 规定目录的权限, 使得这个目录归用户所有。在新的 home 目录里可以提供 shell 初始化文件, 完成一些系统范围内的配置(例如, 为一个新用户的输入 e-mail 建立一个 spool)。

虽然手工增加用户并不困难, 但当你运行一个有许多用户的系统时, 很容易忘掉一些事情。加入用户的最简单的方法是使用一个交互式的程序, 它向你提问所需的信息, 然后自动更新所有相关的系统文件。这个程序的名字是 useradd 或 adduser, 它取决于你安装的是哪一套发行版本的软件, 参看联机手册可知如何使用它们。

4.4.3　删除用户

类似地, 删除用户可以使用命令 userdel 或 deluser, 它取决于系统上安装的是哪一套发行版本的软件。

如果你想使一个用户暂时不能登录到系统中, 但不删除用户的帐户, 那么只需在文件 /etc/passwd 中该用户项的口令域前面加一个"*"。例如, 改变 kiwi 在文件/etc/passwd

中的用户项将限制 kiwi 用户登录。

kiwi: * Xv8Q981g71oKK: 102: 100: Laura Poole: /home/kiwi: /bin/bash

4.4.4 设置用户属性

当你为一个用户建立了帐户之后,可能需要改变其属性。例如,改变 home 目录或口令,最简单的办法是直接修改 /etc/passwd 文件的对应值。设置用户的口令,可使用 passwd 命令。例如下面命令将改变 larry 的口令。

passwd larry

只有 root 用户可以通过这种方式改变其他用户的口令,普通用户只能使用这个命令改变自己的口令。

在有些系统中,可以允许用户使用命令 chfn 和 chsn 设置他们自己的全名(fullname)和登录 shell 属性。如果没有这样的命令,他们只能求助于系统管理员来改变这些属性。

4.4.5 组(groups)

如前所述,每个用户属于一个或多个组。组关系的实际的重要性在于文件的权限。你不妨回顾一下 3.9 节的内容,在那里我们提到过每个文件都有组所有权和一套组的权限,它规定了在一个组里的用户可以怎样访问文件。

有几个特殊的组是由系统定义的,例如 bin、mail 和 sys。普通用户不应属于这些组中的任何一个;它们只被用于规定系统文件的权限。普通用户可以属于另外一个单独的组,例如 users。你可以保持几个用户组,例如 student、staff 和 faculty。

文件 /ect/group 包含了有关组的信息,它的每一行的格式是这样的:

group name: password: GID: other members

以下是一些组的例子:

root: *: 0:

users: *: 100: mdw,larry

guest: *: 200:

other: *: 250: kiwi

第一个组 root,是一个为 root 帐户保留的特殊的组。第二个组 users,是个普通用户组,它的 GID 是 100,用户 mdw 和 larry 是属于这一组的两个用户。也许你还记得在文件 /etc/passwd 中每个用户都被赋予一个默认的 GID,但是用户也可以同时属于多个组,只需在文件 /etc/group 中的其他组的那一行加入你的用户名。group 命令将显示你属于哪些组。

第三个组是 guest 用户组;第四个组是 other 用户组,用户 kiwi 属于这一组。

你会发现 /ect/group 中的“password”域很少被用到,它只是有时被用来设置组访问的口令,它不是非常重要的。为了防止用户用 newgroup 命令进入特权组,将“passwd”域设置成“*”。

命令 addgroup 或 groupadd 可用来在你的系统中增加用户组。通常,你只需在 /etc/group 中简单地加入一项,而不必做其它设置,就可增加一个组。同样删除组只需删除

/etc/group中对应的项。

4.5 归档和压缩文件

这一节将先介绍一下Linux系统中用于文件和软件的归档的一些工具。下一节我们将讨论制作备份的问题。

4.5.1 使用tar命令

tar命令经常被用于制作归档文件,它的格式如下所示:

tar <options> <file1> <file2> ... <fileN>

这里<options>项列出了tar的选项,<file1>到<fileN>列出了从归档文件中加入或提取的文件。例如,下列命令是将/etc中的所有文件打包到名为backup.tar的归档文件中。

```
# tar cvf backup.tar /etc
```

第一个参数"cvf"是tar的选项。其中"c"选项告诉tar建立一个新归档文件;"v"选项通知tar进入verbose模式,即在归档过程中打印已被归档的每一个文件名;"f"选项告诉tar下一个参数baskup.tar是要建立的归档文件名。其余的参数是加到归档文件上的文件和目录名。

作为另一个例子,下面这个命令将提取归档文件backup.tar中的所有文件或目录到当前目录。

```
# tar xvf backup.tar
```

"x"选项告诉tar作提取文件的操作,其余选项的含义与上一个例子相同。

从tar文件中提取文件有时会很危险,因为如果目录中原来的文件如果和提取出的文件的名字相同,则原有的文件会被新提取出的文件覆盖掉。因此,在提取tar文件之前最好先弄清楚是否有同名的文件。

另外,在提取tar文件之前最好先弄清楚提取出来的文件将被放置到哪里。例如,假设你归档文件/etc/hosts,/etc/group和/etc/passwd时使用下列命令

```
# tar cvf backup.tar /etc/hosts /etc/group /etc/passwd
```

注意,这里文件名用的是带路径名/etc的文件全名。为了提取文件到准确的位置,你需要先进入根目录,然后再作提取文件的操作。即使用下列命令:

```
# cd /
# tar xvf backup.tar
```

这是因为文件是用保存在归档文件中的路径名来提取的。

但是如果你使用下列命令归档文件

```
# cd /etc
# tar cvf hosts group passwd
```

路径名将不会保存在归档文件中。这样,你就需要在提取文件之前执行"cd /etc"命令进入\etc目录。正像你所看到的,tar文件提取到哪里是与它的建立的方法有关的。下面

命令可用于在打包之前显示归档文件中的文件的索引。

```
# tar tvf backup.tar
```

“t”选项指示 tar 命令列出归档文件中的文件的索引。

用这种方法你可以看到文档中的文件名是什么方式的,因而可将需要的文件提取到准确的位置。

4.5.2 gzip 和 compress

不同于 MS-DOS 的归档程序,tar 命令不能一边归档一边同时压缩文件。因此,如果你想压缩两个 1M 字节的文件,那么 tar 文件就按 2M 字节的空间归档。gzip 命令可用于压缩一个文件(要压缩的文件可以不是 tar 文件)。例如下面的命令可将 backup.tar 文件压缩成名为 backup.tar.gz 的压缩版本:

```
# gzip -9 backup.tar
```

其中开关-9 的意思是告诉 gzip 使用最高的压缩因子。

gunzip 命令可用于解压缩一个 gzip 压缩的文件,同样你也可以使用 gzip -d 命令。

在 Linux (以及 Unix)系统中 gzip 是较新的一种工具。很多年来,人们一直使用 compress 命令来压缩文件,但是现在由于其他的一些因素,compress 正在逐步被淘汰。

用 compress 压缩的文件的名字一般以.Z 结尾。例如,backup.tar.Z 是 backup.tar 的 compress 压缩版本,而 backup.tar.gz 是它的 gzip 压缩版本。可以用 uncompress 命令展开一个用 compress 压缩的文件;而 gunzip 命令也可用于展开用 compress 压缩的文件。

4.5.3 组合使用压缩和归档命令

如果要归档并压缩一组文件,可以使用如下命令:

```
# tar cvf backup.tar /etc
# gzip - 9 backup.tar
```

最后得到的结果将是 backup.tar.gz 文件。如果倒过来使用这些命令,可将文件解包:

```
# gunzip backup.tar.gz
# tar xvf backup.tar
```

当然,在解包一个 tar 文件之前一定要先确认你是在正确的工作目录下。

在 Unix 系统中可以只使用一个命令行来执行以上任务,如下所示:

```
# tar cvf - /etc ©¦gzip - 9c > backup.tar.gz
```

这里我们将 tar 文件发送给“-”,它代表 tar 命令的标准输出。它通过流水线作业把控制传给 gzip,gzip 对输入的 tar 文件进行压缩,结果保存在 backup.tar.gz 文件中。gzip 的-c 选项告诉 gzip 将它的输出传送给标准输出,而标准输出重定向到文件 backup.tar.gz 中。

类似地,也可用如下一个命令行解包:

```
# gunzip -c backup.tar.gz ©¦tar xvf -
```

这里, gunzip 解压缩 backup.tar.gz 的文件内容, 然后将得到的 tar 文件传送到标准输出。接着将标准输出通过管道传给 tar, tar 从“-”读入要处理的文件,“-”代表 tar 的标准输入。

tar 命令也包括 z 选项, 它指示 tar 用 gzip 或 compress 压缩算法来自动压缩/解压缩文件。例如命令

```
# tar cvfz backup.tar.gz /etc
```

等价于:

```
# tar cvf backup.tar /etc
# gzip backup.tar
```

同样, 命令

```
# tar xvfz backup.tar.Z
```

可取代如下命令

```
# uncompress backup.tar.Z
# tar xvf backup.tar
```

如果你想了解 tar 和 gzip 的更多信息, 请参看联机手册。

4.6 使用软盘及做备份

软盘通常被用作保存备份的媒体。如果你的系统上没有连接磁带驱动器, 就可以使用软盘来作备份(虽然它与磁带相比速度较慢并且可靠性差些)。

你也可以使用软盘来保存独立的文件系统。在这种情况下, 可以通过安装软盘上的文件系统的方法来访问其上的数据。

4.6.1 使用软盘做备份

使用软盘做备份的最简单的方法是使用 tar 命令, 命令的格式如下:

```
# tar cvfzM /dev/fd0
```

这个命令将使用软盘驱动器 /dev/fd0/ 上的软盘来做系统的完整备份。tar 的“M”选项允许做多卷备份; 也就是说当一个磁盘已写满, tar 会提示用户插入另一张盘。下面的命令可用来完整地恢复备份。

```
# tar xvfzM /dev/fd0
```

使用磁带做备份时也可使用这种方法。

还有其它一些程序也可做多卷备份, 例如 backflops 程序, 它可以在 FTP 的 tsx-11.mit.edu 站点取到。

做系统的完整备份是即费时又耗费资源的一项工作。绝大多数系统管理员采用逐步备份的方法, 每月做一次完整备份, 而每一周只对上一周被改动的文件做备份。这样, 即使你在月中毁掉系统, 还可以恢复上一个月的完整备份, 以及上一周的备份。

find 命令可以用来确定自某一天开始到现在为止哪些文件被修改。在 FTP 的 sunsite.unc.edu 站点可以找到管理逐步备份的 shell 程序。

4.6.2 用软盘存放文件系统

就像在硬盘上作分区一样你可以在软盘上建立一个文件系统。例如下面命令可以在驱动器 /dev/fd0 中的软盘上建立一个文件系统:

```
# mke2fs /dev/fd0 1440
```

文件系统的大小必须和软盘的容量相符。高密 3.5 英寸盘有 1.44M 字节(1 440 块)的容量,高密 5.25 英寸盘有 1.2M 字节(1 200 块)的容量。

为了访问软盘,你必须首先安装软盘上的文件系统。它的命令格式如下:

```
# mount -t ext2 /dev/fd0 /mnt
```

这个命令将/dev/fd0 中的软盘安装在/mnt 目录下。这样,软盘中的所有文件都将在/mnt 目录中出现。“-t ext2”说明被安装的是一个 ext2fs 类型的文件系统。如果你在软盘上建立的是其它类型的文件系统,你就需要在 mount 命令中特别说明它的类型。

当你用 mount 命令时,“安装点”(安装文件系统的目录)必须确实存在,如果不存在,可以先用 mkdir 命令建立它。

有关文件系统、安装方法和安装点的更多信息,请参看 4.8 节。

值得注意的一个问题是,对软盘的任何输入/输出数据都将经过内存缓冲区,这方面与硬盘的输入/输出一样。如果你试一下改变软盘上的数据,你就会看到软盘驱动器的指示灯可能不会闪亮。也就是说,你所作改变的那部分数据在 I/O 缓冲区内,只有到核心程序重写它的 I/O 缓冲区时,数据才会被真正写到软盘上。因此在你没有拆卸文件系统之前,不要像在 MS-DOS 系统上那样,简单地取出软盘。那样可能会导致信息的丢失。可以用下面命令来拆卸软盘上的文件系统:

```
# umount /dev/fd
```

当你想更换软盘时,要先使用命令 umount,再安装(mount)另一个。

4.7 升级和安装新软件

系统管理员的另一项职责就是升级和安装新软件。

Linux 软件版本的升级是很快的,每几个星期就会有新的核心程序发行版发表,而且其它软件也经常升级,因此 Linux 的许多用户总是感到需要经常升级他们的系统,以跟上快速发展的步伐。其实这样做既没必要,又浪费时间。如果你总是在忙于升级系统,那么就根本没有时间使用它了。

那么什么时候升级比较合适呢? 有些人看到一个新的发行版发表,比如说 Slackware 出了一套新版本,就觉得应该升级。许多 Linux 的用户每次都用最新的 Slackware 发行版重新安装他们的系统,这样做很花时间。一般来说,每次新的 Slackware 发行版的改动只是很小的一部分。每次都要重新安装好几十张盘,而实际上只有 10% 的软件是被修改的,这样做是很不值得的。

升级系统的最好方法是手工操作,只升级那些你认为应该升级的软件包。这也许会吓着一些人,他们需要知道升级什么,怎样升级,以及如果不升级会有那些方面落伍。其实为

了更好地使用 Linux,克服害怕“自己做”的心理是很重要的,Linux 可使你尽情发挥你的才能。实际上,当你已经将你的系统运行起来,并正确配置了所有的软件的时候,如果用最新发行版的软件重新安装系统,毫无疑问会清除你以前所做的配置。但是如果你只升级系统的一个组成部分,那么其它部分就不会被破坏。仅仅为了赶时髦而升级是完全没有必要的。Linux 不像 MS-DOS 或 Microsoft Windows,没有必要运行所有软件的最新版本。只有当你发现你需要新版本中新增加的功能时再升级。换句话说,就是在必要的时候升级需要升级的软件,不要为了升级而升级。

在你的系统中最需要升级的部分是核心程序,程序库和 gcc 编译器。它们是系统中必不可少的三部分,它们是相互依赖的,因而它们的版本必须保持一致才能正常工作。除了它们之外的软件没有必要经常升级。

4.7.1 核心程序的升级

升级核心程序时,只需取来源代码,然后你自己编译它们。之所以要你自己编译核心程序,一方面是为了能够让你按自己的需要决定核心程序的各种功能的取舍,另一方面是为了能够针对你的系统的结构和配置情况对核心程序进行优化,使其能够更高效地运行。

核心程序的源代码可以在任何 Linux 的 FTP 站点上找到(站点的列表见附录 B)。例如在 sunsite.unc.edu 站点,核心程序源代码位于/pub/linux/kernel 目录中。其版本用一个版本号和一个修补级 patchlevel 来编码。例如核心程序版本 0.99 的修补级 11 通常写作 0.99.p11 或 0.99.11。

核心程序的源代码通常以一个 gzip 压缩的归档文件的形式发行。例如,包含 0.99.p11 版本的核心程序源代码的文件名是 linux-0.99.11.tar.gz

在目录 usr/src 中打开这个压缩的归档文件,包括对压缩文件解压和对归档文件解包,将建立一个 usr/src/linux 目录,它里面包含所有的核心程序源代码。在打开新版本之前,你应该删除或重新命名已有的 usr/src/linux 目录。

源代码打开后,第一件事就是要确定 usr/include 目录中的两个符号链接是否正确。可以使用如下命令建立这些链接。

```
# ln -sf /usr/src/linux/include/linux /usr/include/linux
# ln -sf /usr/src/linux/include/asm /usr/include/asm
```

一旦你已经建立了这些链接,以后当你每次安装新版本的核心程序源代码时,就不用重新建立它们了(有关符号链接的内容见 3.8 节)。

注意为了编译核心程序,你必须先在系统中安装 gcc 和 g++ 编译器。你可能会需要这两个编译器的最新版本,这方面更多的内容将在后面的“4.7.3 gcc 升级”中介绍。

为了编译核心程序,首先要进入/usr/src/linux 目录,然后运行 make config 命令。这个命令将提示你一些配置选项,比如你想让新的核心程序支持什么样的文件系统类型等。

下一步就是要编辑/usr/src/linux/Makefile 文件,确认在文件中 ROOT_DEV 的定义是否正确。它定义了在系统启动时用作 root 文件系统的设备。它的通常定义是:

```
ROOT_DEV = CURRENT
```

除非你改变了文件系统设备,否则不需改变它。

再下一步就是运行 make dep 命令来确定所有源代码之间的依赖关系。这是非常重要的一步。

最后一步就是编译核心程序了。使用 make Image 命令进行编译并将新的核心程序映象存在文件/usr/src/linux/Image 中。而如果使用 make zImage 命令则编译后得到的核心程序映象是经过压缩的,以节省设备空间。压缩的核心程序的映象在启动的时候将自动解压缩。

得到核心程序的映象后,可以用 cp Image /dev/fd0 命令将其拷贝到一个软盘上,该盘可作为系统启动盘。如果用 LILO 安装它,则可以从硬盘启动系统。有关 LILO 的细节已经在"4.2.2 使用 LILO 启动"中作了介绍。

4.7.2 程序库的升级

如上所述,系统中的大多数软件都要使用共享程序库。当试图运行一个程序时出现如下消息:

Incompatible library version

就表明该程序所需的程序库的版本与要求的不符。那么就需要将程序库升级到该程序所需的版本。程序库是向后兼容的,使用旧版本的程序库的程序可以在新版本的库之下工作,反过来却不行。

新版本的程序库可以从 Linux 的任何 FTP 站点取到。在 sunsite.unc.edu 站点,它们位于/pub/Linux/GCC 目录中。release 文件说明了所有需要取的文件以及安装它们的方法。简单地说,你只需取 image-version.tar.gz 和 inc-version.tar.gz 两个文件,version 表示要安装的程序库的版本,例如 4.1.1。这两个文件是用 gzip 压缩的,image 文件包含将要安装在/lib 和/usr/lib 目录下的程序库的映象。inc 文件包含将要安装在/usr/include 目录下的 include 文件。

release-version.tar.gz 文件详细解释了安装过程(每一个发行版的安装过程都有不同之处)。一般来说需要将 .a 和 .sa 类型的库文件安装到/usr/lib 目录中,它们是在编译时要用到的程序库。

此外,共享程序库映象文件 libc.so.version 应安装在目录/lib 中。它们是使用库的程序在运行时调出的共享程序库映象。每一个程序库都有一个使用/lib 目录中的库的主版本号的符号链接。

例如 libc 程序库的版本 4.4.1 的主版本号为 4,包含该库的文件是 libc.so.4.4.1,/lib 中有一个名为 libc.so.4 的符号链接,它指向这个文件。当升级库的时候也需要相应地改变这个链接。例如,当你从 lib.so.4.4 升级到 lib.so.4.4.1 时,需要将这个链接指向新的版本。

使用如下命令可以将符号链接 libc.so.4 更新为指向文件 libc.so.4.4.1:

```
# ln -sf /lib/libc.so.4.4.1 /lib/libc.so.4
```

像这样在一次命令操作中改变链接是很重要的。如果你删除了符号链接 libc.so.4,那么依赖于该链接的程序(包括像 ls, cat 和 ln 这样的基本功能)将不能工作。

符号链接 libm.so.version 也同样需要改变。

4.7.3 gcc 的升级

gcc 编译器用来编译系统中的各种软件,包括最重要的核心程序。gcc 的最新版本可以从 Linux 的 FTP 站点取到。在 sunsite.unc.edu 站点,gcc 编译器及其程序库位于/pub/Linux/GCC目录中。目录中的 release 文件说明了安装它们的方法。

4.7.4 其它软件的升级

其它软件的升级通常是比较简单的,只需取来相应的文件然后安装它们。Linux 的软件可能是以二进制形式、源代码的形式或是同时以两种形式发行的。大部分的软件都是以 gzip 压缩的归档文件的形式分发的。如果发行版中不包含二进制代码,用户就需要自己编译它们。编译的过程很简单,通常只需在源代码所在的目录下键入 make 命令。

发现新软件的最简单的方法是阅读 USENET 新闻组 comp.os.linux.announce 中的最新软件发行版的通告。当你要在一个 FTP 站点中寻找软件时,首先取来名为 ls-lR 的索引文件,然后用 grep 命令寻找所需的文件。如果你可以使用 archie 的话,也可以使用它来完成这一工作(如果没有 archie,你可以远程登录到一个 archie 服务器,例如 archie.rutgers.edu。你可以 archie 帐户名登录,然后键入 help 命令查看帮助信息)。更多的细节请参见附录 A 所列的资料。

同样,采取重新安装 Slackware 或其它发行版本的方法进行升级也不是很好的办法。这样作无疑会破坏你当前的安装,包括用户目录和你定制的所有配置。最好的升级方法是一部分一部分地进行,也就是说每当你常用的一个软件有了一个新的版本时,单独升级它自己。

4.8 文件系统的管理

系统管理员的另一个任务是管理文件系统。这一任务中的最大量的工作是定期检查和处理文件系统中被破坏的文件。许多系统在启动时自动检查文件系统。

4.8.1 安装文件系统

让我们先来看一些有关文件系统的概念。一个文件系统必须安装在某个目录上,才能被访问。例如一个软盘上有一个文件系统,要访问它必须先将其安装在某个目录上,比如说 /mnt 目录。将文件系统安装到一个目录上之后,该文件系统中的所有的文件将出现在这个目录之下。拆卸了这个文件系统之后,目录将变成空的。

这对硬盘上的文件系统也是一样。系统在启动时自动安装硬盘上的文件系统。root 文件系统被安装在根目录上。

命令 mount 用来安装文件系统。系统启动时执行的系统初始化文件/etc/rc 中执行 mount -av 命令,它从文件/etc/fstab 中获得文件系统及其安装点的信息。从下面的例子(核心程序版本号为 1.1.37)中可以看出 fstab 文件的格式。

```
# device            directory       type        options
```

```
/dev/hda2       /          ext2       defaults
/dev/hda3       /usr       ext2       defaults
/dev/hda4       none       swap       sw
/proc           /proc      proc       none
```

其中第一个域是要安装的分区名(或设备名)。第二个域是安装点。第三个域是文件系统类型标识,比如 ext2 代表 ext2fs 文件系统,minix 代表 Minix 文件系统。表 4.1 列出了所有 Linux 可用的文件系统类型。在编译系统核心程序时,可以任意选择支持这些类型。fstab 文件中的最后一个域是安装选项,通常被设置为 defaults。

你可以看到 swap 分区也包含在这个文件中。它的 directory 域为 none,type 域为 swap。swapon -a 命令在/etc/rc 中也被执行,它使/etc/fstab 中的所有交换设备投入运行。

表 4.1 Linux 支持的文件系统类型

文件系统名	类型	主要功能
第二扩展文件系统	ext2	最普通的 Linux 文件系统。
扩展文件系统	ext	已由 ext2 替代。
Minix 文件系统	minix	最初的 Minix 的文件系统,很少被使用。
Xia 文件系统	xia	与 ext2 类似,但很少被使用。
UMSDOS 文件系统	umsdos	用于在 MS-DOS 分区上安装 Linux
MS-DOS 文件系统	msdos	用于访问 MS-DOS 的文件
/proc 文件系统	proc	为 ps 命令提供进程信息
ISO 9660 文件系统	iso9660	为大多数 CD-ROM 所使用的文件系统格式
Xenix 文件系统	xenix	用于访问 Xenix 的文件系统
系统 V 文件系统	sysv	用于访问 Unix 系统 V 在 x86 系列机器上的变种系统的文件系统
Coherent 的文件系统	coherent	用于访问 Unix 系统 V 的文件系统
HPFS 文件系统	hpfs	用于只读访问 HPFS 分区(双倍空间)

fstab 文件包含一个有关/proc 文件系统的特殊项。在 3.3 节我们提到 /proc 文件系统是用于存储系统状态信息的,如果 /proc 目录没有安装,像 ps 这样的命令将不能工作。

mount 命令只能由超级用户 root 使用。这是为了保证安全性,你当然不希望普通用户随便安装和拆卸文件系统。有几种软件包可使普通用户在不影响安全性的前提下安装和拆卸文件系统(特别是软盘)。

实际上,mount -av 命令安装除了 root 文件系统(在上面的例子中是/dev/hda2)之外的所有文件系统。root 文件系统是在系统启动时由系统内核自动安装的。

用下面的命令也可以手工安装一个文件系统:

```
# mount -t ext2 /dev/hda3 /usr
```

它等价于在上面的 fstab 文件实例中用/dev/hda3 项来安装这个文件系统。

一般情况下,你不需要手工安装和拆卸文件系统,/etc/rc 文件中的 mount -av 命令

将在系统启动时自动安装所有文件系统。在关闭系统之前 shutdown 或 halt 命令将自动拆卸所有的文件系统。

4.8.2 检查文件系统

时常检查文件系统,查看是否有被损坏的文件,是一种好的做法。一些系统在启动时执行/etc/rc 中加入的检查文件系统的命令自动完成检查的工作。

检查一个文件系统的命令依赖于被检查的文件系统的类型。对于最常使用的 ext2fs 类型的文件系统,检查它们的命令是 e2fsck。例如,如下的命令将检查/dev/hda2 上的 ext2fs 文件系统并自动修正发现的错误。

```
# e2fsck -av /dev/hda2
```

在检查一个文件系统之前最好先将其拆卸下来。例如,用下面的命令可拆卸/dev/hda2 上的文件系统。

```
# umount /dev/hda2
```

一个例外的情况是 root 文件系统是不能拆卸的。为了检查 root 文件系统,需要使用一个维护系统用的 boot/root 盘。当一个文件系统中某个文件是忙时,也就是说有一个运行的进程正在使用它时,这个文件系统也是不能拆卸的。例如,如果某个用户的当前工作目录在一个文件系统中,那么这个文件系统就不能被拆卸。如果你试图拆卸一个正在被使用的文件系统,那么你将看到错误信息“Device busy”。

其它类型的文件系统使用不同形式的 e2fsck 命令,比如 efsck 和 xfsck。在有些系统上,你可以简单地使用 fsck 命令,系统将自动确定文件系统类型,然后执行合适的命令。

在检查完一个安装的文件系统后,如果对它进行了某些修正,那么就应该立即重新启动系统(当然一般情况下)。例如,如果 e2fsck 命令报告它修正了文件系统中的某些错误,那么你就应该立刻用 shutdown -r 命令重新启动系统。这是为了让系统重新同步有关该文件系统的信息。

/proc 文件系统不需要按这种方式检查。它是由内核直接管理的一个存储器文件系统。

4.9 使用交换文件

交换(swap)空间除了可以使用一个交换分区之外,也可以使用一个文件。但是为了能这样作,你必须在建立交换文件之前,先安装好 Linux 的软件。

如果你已经安装好了一个 Linux 系统,则可以用如下的步骤来建立一个交换文件。假设我们想建立一个尺寸为 8202 块(大约 8M)的交换文件。首先用下面的命令建立文件本身。

```
# dd if=/dev/zero of=/swap bs=1024 count=8208
```

然后用下面的命令初始化这个交换文件,将其写到盘上,并使其投入运行。

```
# mkswap /swap 8202
# /etc/sync
```

```
# swapon /swap
```

像这样使用交换文件作为交换空间的做法的主要缺点是,对交换文件的所有访问都要通过文件系统。这意味着构成交换文件的磁盘块不是连续的。这样会造成交换操作性能的下降,因为当使用交换分区时磁盘块总是连续的,I/O 请求是直接针对设备进行的。

这样做的另一个缺点是它有可能会破坏文件系统。当使用大的交换文件时,如果出现某些错误,就可能会损坏文件系统。将文件系统和交换空间分开可避免这种情况的发生。

当你临时需要更多的交换空间时,使用交换文件是很方便的。例如当你要编译一个很大的程序,而想让编译过程执行得更快时,就可以采用这种方法。

当你不想使用一个交换文件时,首先使用 swapoff 命令:

```
# swapoff /swap
```

然后删除/swap 文件。

注意,每个交换文件或分区最大只能是 16M 字节,但在一个系统中最多可使用 8 个交换文件或分区。

4.10 杂项任务

4.10.1 系统启动文件

当系统启动时,在任何用户登录之前,一些文件将被系统自动执行。下面将说明这一过程。

系统启动时内核派生出/etc/init 进程。init 程序将读它的配置文件/etc/inittab,并按照该文件的内容派生对应的进程。从 inittab 启动的重要的进程之一就是在每个虚控制台启动的 etc/getty 进程。getty 进程为虚控制台启动一个登录(login)进程,这就使你可以在每个虚控制台上进行登录。

在/etc/inittab 中被执行的另一个进程是/etc/rc,主系统初始化文件。这个文件是一个简单的 shell 文本,它执行系统启动时所需的任何初始化命令,例如,安装文件系统和初始化交换空间。

你的系统还可能执行其它的初始化文本,例如/etc/rc.local。/etc/rc.local 通常包含你自己系统上特殊的初始化命令,例如设置 hostname。rc.local 可以从/etc/rc 或者/etc/inittab 中启动。

4.10.2 设置主机名(hostname)

在一个网络环境中,主机名用来唯一地标识一个特定的机器。设置系统的主机名很简单,只需使用命令 hostname。如果你的系统在一个网络上,那么它应该是你的机器的全名,比如 goober.norelco.com。如果系统不在任何个网络上,那么你可以取任意的主机名和域名。

当设置主机名时,它必须出现在文件/etc/hosts 中,该文件说明主机名和 IP 地址之间的对应关系。即使你的机器不在网络上,也应该将其主机名包含在/etc/hosts 文件中。

例如,如果你的机器不在一个 TCP/IP 网络上,而你的主机名是 floof.org,那么只需

将下面一行写到文件/etc/hosts 中。

```
127.0.0.1          floof.org localhost
```

它将你的主机名:floof.org 指定到 loopback 地址 127.0.0.1(当系统不在网络上时使用的地址)。别名 localhost 也被指定到这个地址。

如果你的系统在一个 TCP/IP 网络上,那么它的 IP 地址和主机名应出现在/etc/hosts 文件中。例如,如果你的主机名是 goober.norelco.com,IP 地址是 128.253.154.32,那么将下面一行加入到/etc/hosts 文件中。

```
128.253.154.32     goober.norelco.com
```

如果你的主机名没有出现在/etc/hosts 文件中,那么你将不能设置它。

使用 hostname 命令可以设置主机名。例如:

```
# hostname -S goober.norelco.com
```

在大多数情况下,hostname 命令是在一个系统启动文件中被执行的,比如在/etc/rc 或/etc/rc.local 中。编辑这两个文件,找到这个命令并将主机名改成你所要设置的名字,然后重新启动系统,那么主机名将设置成新的名字。

4.11 紧急情况的处理

在有些情况下,系统管理员将面对恢复严重事故造成的损失的问题。例如忘记了 root 帐户的口令或破坏了文件系统。遇到这种情况时不要惊慌,谁都有可能犯愚蠢的错误,这时寻找挽回损失的方法的过程,可能是学习系统管理的最佳途径。

解决任何问题的第一步是试图自己处理它。在大多数情况下,你会发现你自己处理它是很容易的。

只有很少的情况需要重新安装系统。很多新用户偶然删除一些基本的系统文件后,立即想到重新安装系统,这不是一个好的办法。在采取这样的万不得已的措施之前,最好先探查一下问题的所在,并请其他人帮助你解决它。对几乎所有的情况,你都可以使用一个维护盘来恢复你的系统。

4.11.1 使用维护盘进行恢复

系统管理员的必不可少的工具是一个 boot/root 盘,用它可以独立于硬盘来启动和运行一个完整的 Linux 系统。boot/root 实际上很简单,首先在软盘上建立一个 root 文件系统,将必要文件和目录拷贝到这一文件系统中,然后在这个软盘上安装 LILO 和可启动的系统内核。另一种方法是用一个软盘存放内核,另一个软盘用来存放 root 文件系统。两种方法的结果都是一样的,就是完全从软盘启动和运行一个 Linux 系统。

boot/root 盘的一个典型例子是 Slackware 的启动盘,包括一个 boot 盘和一个 root 盘。它们本来是用来安装 Slackware 分发版的 Linux 的,但用它们来进行系统维护的工作也是很方便的。

H.J Lu 的 boot/root 盘是这样的系统维护盘的另一个实例,你可以从 sunsite.unc.edu 站点的/pub/Linux/GCC/rootdisk 目录中找到它。当然你可以自己制作一个维护盘,

但在大多数情况下,使用一个现成的 boot/root 盘要更方便,而且可能比你自己制作的盘更完备。

使用 boot/root 盘的方法很简单,只需在你的机器上启动它并以 root 帐户登录进系统。为了能访问你的硬盘上的文件,需要手工安装你的文件系统。例如,用下面的命令可安装/dev/hda2 盘上的 ext2fs 型文件系统到/mnt 目录上:

```
# mount -t ext2 /dev/hda2 /mnt
```

注意,现在的根目录 / 是 boot/root 盘的根目录。为了访问你的硬盘文件系统中的文件,你必须先把它安装到某个目录下。这样如果你将硬盘上的文件系统安装到/mnt 目录下,则硬盘上原来的/etc/passwd 文件现在的路径是/mnt/etc/passwd。

4.11.2 忘记 root 帐户的口令时的处理方法

当你忘记 root 帐户的口令时,不要着急,你可以用 boot/root 盘启动系统,将你的 root 文件系统安装到/mnt 目录上,然后编辑/mnt/etc/passwd 文件,将其中对应于 root 帐户的一行中的口令域置空,如下所示:

```
root::0:0:root:/:/bin/sh
```

这样 root 帐户就没有口令了。当重新从硬盘上启动系统时,你就可以用 root 帐户登录了。登录之后,你可以用 passwd 命令设置新的口令。

4.11.3 文件系统被破坏时的处理方法

当你的文件系统被破坏时,如果使用的是 ext2fs 型的文件系统,你可以从软盘上运行 e2fsck 命令来修正文件系统中被损坏的数据。对于其他类型的文件系统,可以使用相应的 fsck 命令。

当你从软盘上检查文件系统时,最好不要安装它。

文件系统被破坏的常见的原因是超级块被损坏,超级块是文件系统的"头",它包含文件系统的状态,尺寸和自由块等信息。如果你损坏了一个文件系统的超级块(例如不小心直接将数据写到了文件系统的分区中),那么系统可能会完全不识别该文件系统,这样也就不能安装它了,而且 ext2fs 也不能处理这个问题。

幸运的是,ext2fs 型的文件系统将超级块的备份存放在驱动程序的块组(block group)边界。你可以用如下的命令告诉 e2fsck 使用超级块的拷贝。

```
# e2fsck -b 8193 < partition>
```

这里< partition> 指出文件系统所在的分区,-b 8193 选项告诉 e2fsck 使用存放在文件系统的 8193 块的超级块的拷贝。

4.11.4 恢复丢失的文件

如果你不小心删除了重要的文件,那么没有办法直接恢复。但是你可以将相应的文件从软盘上拷贝到硬盘上。例如,如果你删掉了文件 /bin/login,可以用 boot/root 盘启动系统,将 root 文件系统安装到/mnt 目录下,然后使用如下的命令:

```
# cp -a /bin/login /mnt/bin/login
```

-a 选项用于告诉 cp 在拷贝时保持文件的访问权限。

当然如果被删掉的基本文件不在 boot/root 盘中，也就不能用这种方法了。如果你作了备份的话，那么就可以用它们来恢复。

4.11.5 库被破坏时的处理方法

如果你不小心破坏了库或者/lib 目录中的符号链接，那么依赖于这些库的命令将不能执行。最简单的解决方法是用 boot/root 盘启动系统，安装你的 root 文件系统，然后修复/mnt/lib 目录中的库。

第 2 部分　网　络　篇

第 5 章　TCP/IP 网络简介

在介绍 Linux 中的网络配置和管理之前，首先在本章中简单介绍一下 TCP/IP 网络的有关基本知识。如果读者已经对 TCP/IP 比较熟悉，而只想了解 Linux 中具体的网络配置与管理问题，就可以跳过本章，直接阅读后面的内容。

5.1　TCP/IP 网络简介

TCP/IP(Transmission Control Protocol/Internet Protocol)是传输控制协议/网间网协议的英文缩写，最初是 1969 年开始由美国国防高级研究计划署(DARPA)资助的一个研究项目，用于研究网络互联技术。人们首先利用 TCP/IP 建立了一个试验性的网络，称为 ARPANET，在证明其获得成功后，ARPANET 于 1975 年开始转入正式的操作运行。

1983 年新的 TCP/IP 协议集被作为一个标准确定下来，并且要求 ARPANET 网络上的所有主机都使用此协议集。当 ARPANET 最后发展成为 Internet 后(ARPANET 于 1990 年后不复存在)，TCP/IP 的使用已经远远超过了 Internet 网络本身。最显著的是，基于 UNIX 的局域网都使用了 TCP/IP 协议。

在开始介绍 TCP/IP 网络的相关知识以前，我们先通过几个具体的例子，对 TCP/IP 产生一些感性的认识。在下面的讨论中，我们均以 Groucho Marx 大学(GMU)为例子进行介绍。学校中绝大多数系都有自己的局域网，有的是几个系共用一个，有的系则有几个局域网。这些网都通过校园的 FDDI 主干网互联，并且通过一个条高速线路与 Internet 连接。

假设有一台 Linux 主机连接在数学系的一个局域网上，名字叫 erdos。为了访问物理系的主机 quark，在 erdos 上使用下列命令:

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

在此提示下可以输入自己在 quark 上的登录名，如 andres 及相应的密码，系统就会为你提供一个 quark 上的 shell，这样在 erdos 上就可以像坐在 quark 的终端前面一样运行各种命令。当退出此 shell 后，就又返回到 erdos 主机的命令提示符下。这就是 TCP/IP 提供的一个交互式应用的一个实例:远程登录(remote login)。

当你登录到 quark 上时，可能希望运行基于 X11 的应用程序，如一个绘图程序，或者是 PostScript 的浏览器。为了告诉此应用程序将窗口显示到你的主机的屏幕上，而不是 quark 的屏幕上，就必须设置 DISPLAY 环境变量:

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

当启动应用程序后，在 erdos 上运行了 X11 的前提条件下，它将与 erdos 的而不是

quark 的 X 服务器进行联系,并将所有的窗口都显示到 erdos 的屏幕上。这种服务是由 TCP/IP 在 erdos 和 quark 之间来回传递 X11 分组(packet)完成的,让你产生现在是在一个系统中运行应用程序,而不是在两个系统的印象。也就是说在这里网络是透明的。

TCP/IP 网络中另一个非常重要的应用是网络文件系统(NFS)。它以另外一种形式使网络对用户透明,比如在本地主机上安装其它主机的一些目录结构,使得自己可以像访问本地文件系统一样访问其它主机的这些目录。

上面只是一些可以在 TCP/IP 网络上运行的几个例子,实际上 TCP/IP 的应用几乎是无限的。

下面首先简单的回顾一下网络硬件的有关知识,然后较为详细的介绍 TCP/IP 的工作方式,以便理解为什么和怎么配置你的 Linux 机器。

5.1.1 以太网

在所有的局域网中使用最为广泛的硬件类型就是通常所说的以太网(ethernet)。它由一根电缆组成,主机通过连接器(connector)或收发器(transceiver)连在电缆上。简单以太网的安装十分便宜,并且可以达到每秒 10M 位的传输速率,因此使用十分广泛。

以太网有三种组成形式,即粗缆、细缆和双绞线。细缆和粗缆使用的都是同轴电缆,区别在于电缆的粗细和主机与电缆连接的方式。细缆使用的是 T 型 BNC 连接器,安装时需将电缆切开,将 T 型连接器嵌入其中。而粗缆以太网的安装是在电缆上钻一个孔,将收发器通过一个特殊的螺钉附着在上面。一个收发器可以和一个或多个主机连接。细缆和粗缆以太网的电缆分别可以延伸 200m 和 500m,因此分别称为 10Base-2 和 10Base-5。双绞线是由两根铜线组成的,另外也还需要一些其它的附加设备,常被称之为 10Base-T。

尽管把一台主机连接到粗缆以太网上相比之下稍微有些麻烦,但它不需要使整个网络的运行都停止下来。而在细缆以太网上加入主机时,需要将电缆切开以插入连接器,因此必须使整个网络传输停止一段时间。大多数人喜欢使用细缆以太网,因为它十分便宜。然而在大型的网络安装中粗缆更为合适一些,这样在每次增加主机时不会影响到整个网络的正常运行。

以太网技术的一个缺陷是它的长度限制,使得它只能用在局域网中。但是多个以太网段可以通过中继器(repeater)、网桥(bridge)、路由器(router)等互相连接起来。中继器只是简单的将信号在两个或多个以太网段之间进行转发,因此通过中继器连接的多个以太网的行为与一个以太网类似。由于时间要求,任何两个主机间的中继器不能超过 4 个。网桥和路由器相比之下要复杂得多,它们需要对收到的数据进行分析,只有在数据的目的地址不在本局域网时才进行转发。

以太网的工作与总线系统类似,一个主机可以向本地网络上的其它主机发送最大为 1 500 字节的分组(packet)(或者称为帧)。每个主机的地址由以太网卡上的 6 个字节的物理地址标识。这些地址的书写形式通常是以冒号隔开的两位十六进制数字的序列,每两位十六进制数与一个字节对应,如 aa:bb:cc:dd:ee:ff。

一个主机发送的数据帧可以被局域网上所有相连的主机监听到,但只有目标主机进行接收和处理。如果两台主机试图同时发送数据,就会发生冲突,其解决办法是两台主机

都放弃发送,在等待一段随机长度的时间后再重新发送。

5.1.2 其它类型的硬件

在比较大型的安装中,通常使用的不仅仅是以太网一种设备。比如 GMU 的校园主干网就是使用光纤的 FDDI(Fiber Distributed Data Interface, 光纤分布式数据接口)。FDDI 使用一种完全不同的规则进行数据的传输:即在各主机间循环传递一个令牌(token),只有获得令牌的主机才能发送数据。FDDI 最大的优点是其传输速率可以达到 100M,光缆的最大长度可以达到 200km。

对于长距离的网络连接,经常使用的是基于 X.25 标准的另外一种不同类型的设备,称为公用数据网(PDN, Public Data Network)。X.25 需要特殊的硬件设备,即 PAD(Packet Assembler/Disassembler)。X.25 定义了自己的一套网络协议,但还是通常用它来连接使用 TCP/IP 和其它协议的一些网络。由于 IP 分组数据不能简单的映射到 X.25 上(反之也不行),因此一般是将其简单的封装在 X.25 的分组中在网络上传输。

其它技术包括使用低速、便宜的串行线来进行拨号访问。这需要另外一种传输协议,如 SLIP, PPP 等,我们将在后面加以介绍。

5.1.3 网间网协议(IP)

一般来说用户并不希望自己的主机只限于在一个以太网上连网。而是希望自己所在的局域网可以和其它网络相连,不用考虑其它网络的硬件类型以及其各部分的组成,就能够使用这些资源。例如在 GMU,就需要将各个系的局域网连接起来。

这种网络之间的连接是通过一个专门的称为网关(gateway)的主机来实现的。它通过在连接的网络之间进行数据的拷贝来处理输入和输出的数据。例如,如果你在数学系,想从 Linux 主机 erdos 上访问物理系局域网上的 quark,因为它们不在同一个以太网上,网络软件不能够直接给 quark 发送数据,因此需要依赖网关将数据进行转发。网关(假设叫 sophus)在接收到这些数据后通过校园主干网就将它们转发到物理系的网关 niels,由 niels 将数据传输到目的主机。erdos 和 quark 上的数据流动,如图 5.1 所示。

图 5.1 从 erdos 发送数据到 qurak 的三个步骤

这种将分组(packet)向远程主机传递的方式称为路由(routing),并且分组在这样的上下文中称为数据报(datagrams)。数据报的交换是由一个独立于硬件的协议网间网协议(IP)控制的。我们将在后面对 IP 和路由作详细的介绍。

IP 的主要优点在于它将物理上不相似的网络联接起来,构成一个同构的网络,称为网间网。

IP 也需要一个独立于硬件的寻址计划,每个主机都设置了一个 32 位的地址,称为 IP 地址。IP 地址通常写成 4 个十进制数,每个十进制数对应于 32 位中的一个字节,用“.”号隔开。如 quark 可能具有的 IP 地址是 0x954c0c04,常写为 149.76.12.4。这种格式也称为点分十进制表示(dotted decimal notation)。

你可能会注意到现在已经介绍了三种地址:首先是一个机器的主机名字,如 quark,然后是 IP 地址,最后每个主机都有一个硬件地址(如 6 个字节的以太网卡的地址)。所有的这些地址之间必须进行匹配,这样当你输入 rlogin quark.physics 时,网络软件可以获得 quark 的 IP 地址,当 IP 将数据传输到物理系的以太网时,它必须找到与此 IP 地址匹配的以太网卡的地址。

此处不再进一步讨论,后面将会对地址的映射进行较为详细的介绍。现在需要记住的是将主机名映射到 IP 地址称为域名解析(resolve),将 IP 地址映射到硬件地址称为地址解析。

5.1.4 串行线上的 IP

在串行线路(serial line)上,经常使用的是称为 SLIP(Serial Line IP)的标准。SLIP 的一个变体称为 CSLIP,或者称为压缩的 SLIP。它对 IP 的数据报头部进行压缩以更好的利用串行线路相对比较低的传输速率。另一个不同的串行协议为 PPP,称为点到点的协议(Point-to-Point Protocol)。PPP 比 SLIP 增加了一些特点,它包含了一个连接建立的握手过程。与 SLIP 相比最突出的优点是其不仅仅限于传输 IP 数据报,它被设计用来传输任何类型的数据报。

5.1.5 传输控制协议(TCP, Transmission Control Protocol)

现在,通过前面介绍的 IP,可以实现从一台主机向另一台主机进行数据报的发送。但只有这些显然是不够的。例如,你登录到 erdos,希望在 quark 上的 rlogin 进程同 erdos 上的 shell 进程建立比较稳定的连接。这样,来回传递的信息必须在发送端分解成一些分组(packet),由接收方重新将它们组装成字符流。这个过程看起来很简单,实际上涉及到许多工作。

对于 IP 来说,需要特别指出的是它不是十分可靠的。假设你所在的以太网上的 10 个人同时从某个 FTP 站点下载(download)最新版本的 XFree86,它所产生的通信量可能太大,使得网关处理不了。此时如果你碰巧向 quark 发送一个数据,而路由器的缓冲区刚好在此刻溢出,因此不能对此数据进行转发。IP 在遇到这种问题时只是简单的放弃此分组。这样分组就不可恢复地丢失了。因此就需要由负责通信的主机对数据传输的完整性和正确性进行检验,在发生错误后进行数据的重新发送。

这个工作是由另外一个称为传输控制协议(TCP, Transmission Control Protocol)来完成的,它是在 IP 之上建立可靠的服务。TCP 最主要的性质是它利用 IP 使你产生一种印象,就是在你的主机和远程主机的两个进程之间有一个简单的连接(connection)关系

存在,这样就不需要担心数据是怎样和沿什么路径进行传输的。一个 TCP 连接的工作与一个双向的管道十分类似,两个进程都可以从它读,也都可以向里写。可以把它想象为电话交谈。

TCP 通过两个主机的 IP 地址以及两个主机上的端口来识别连接的端点。端口可以认为是网络连接的附着点。如果进一步拿电话来作比较的话,IP 地址与区号类似,端口号相当与本地电话号码。

在本章开始 rlogin 的例子中,客户程序(rlogin)在 erdos 上开一个端口,与 quark 上 rlogin 服务程序 rlogind 监听的 513 端口相连,这样就建立了一个 TCP 连接。通过使用这个连接,rlogind 首先进行权限检查,然后派生出一个 shell。shell 的输入和输出被重定向到 TCP 连接,这样你在本机器上对 rlogin 的任何输入都将通过 TCP 连接传递给 shell 作为标准输入。

5.1.6 用户数据报协议(UDP, User Datagram Protocol)

当然,TCP 不是 TCP/IP 网络中唯一的用户协议。TCP 对于一些服务如 rlogin 来说还是比较合适的,但对于一些服务如 NFS 来说开销就太大了。NFS 使用一种用户数据报协议(UDP)来代替 TCP。与 TCP 一样,UDP 也允许一个应用程序与远程主机的一个端口联系服务,但它并不为此建立一个连接,而是向目标服务发送单个的数据报。

假设你从系里的 NFS 服务器(nfsserver)安装了一个 TEX 的目录结构,并想看一个描述如何使用 LATEX 的文档。在启动编辑器后,它首先读入整个文件。如果与 nfsserver 建立一个 TCP 连接,发送文件,然后再释放是很花费时间的。而向 nfsserver 发送一个请求,将文件包含在几个数据报中就快的多。然而 UDP 并不处理数据报的丢失和冲突。在这种情况下是由应用程序 NFS 自行解决的。

5.1.7 端口

端口可以看成是网络连接的附着点。如果一个应用程序希望提供某种服务,它将自己附着在端口上等待客户请求的到来(也称为对端口进行监听)。希望使用此服务的客户在本地主机分配一个端口,与远程主机的服务端口连接。

端口的一个重要性质是在客户和服务器之间一旦建立连接,必须生成服务器的另外一个拷贝来对服务端口进行监听,以等待更多客户的到来。这样就允许多个用户可以同时登录到同一台主机,并且都使用同一个端口 513。TCP 可以对这些连接进行区分,因为它们来自不同的端口或主机。例如你两次从 erdos 登录到 quark 上,那么第一个 rlogin 客户程序将使用本地的 1023 端口,第二个使用 1022 端口,但是它们两个都连接到 quark 的 513 端口。

这个例子表明可以把端口作为一个集结点,客户通过联系这些特殊的端口来获取特殊的服务。为了使客户知道正确的端口号,双方系统管理员对这些端口的分配必须达成一致的协议。对于一些广泛使用的服务如 rlogin 等,必须进行集中的管理。这个工作是由 IETF(Internet Engineering Task Force)进行的,他们经常发行名字为“Assigned Numbers”的 RFC。它描述了许多东西,其中包含分配给一些著名服务的端口号。Linux 中

是通过一个文件/etc/services 文件进行服务名称到端口号的映射,我们将在“7.1.3 services 和 protocols 文件”中对其进行详细的介绍。

值得注意的是TCP 和 UDP 连接都依赖于端口,但这些端口是不冲突的。这意味着 TCP 的 513 端口与 UDP 的 513 端口不同。实际上,这些端口是两种不同服务 rlogin (TCP)和 rwho(UDP)的访问点。

5.1.8 Socket 库

在UNIX 操作系统中,用来处理上述所有任务和协议的软件通常是作为操作系统核心的一部分,Linux 也不例外。在 UNIX 系统中最常用的编程接口是 Berkeley Socket 库。它提供了确定远程主机(bind(2))、传输协议(connect(2))、应用程序连接或监听(listen(2)或 accept(2))的服务。socket 库在一般程度上更通用,它不仅提供一类基于 TCP/IP 的套接字(AF- INET 套接字),还提供了一类处理局部与本机器的连接的套接字(AF-UNIX类)。有些实现还提供了处理其它的类型,如 XNS(Xerox Nnetworking System)协议,或者 X.25。

在 Linux 中,socket 库是作为 libc 标准 C 语言库函数的一部分。它现在只支持 AF-INET 和 AF- UNIX 两类套接字。现在有许多人正在努力提供对 Novell 网络协议的支持,最终还会有许多类型的套接字加入。

5.2 网络接口

为了屏蔽网络环境中网络设备的多样性,TCP/IP 定义了一个抽象的的接口,所有硬件的访问都是通过接口进行的。接口提供了一个对所有类型的硬件都相同的操作集合来处理基本数据的发送和接收。

对于连网所使用的每一个外部设备,都必须在操作系统核心中提供一个相应的接口。例如在 Linux 中以太网的接口称为 eth0,eth1 等,SLIP 接口称为 sl0,sl1。这些接口的名称主要用于配置目的,除此之外没有其它意义。

为了可以被 TCP/IP 网络使用,每个接口必须设置一个 IP 地址,以便在与其它主机进行通信时作为自己的标识。这个地址与上面提到的接口名称不同,如果把接口比喻成一扇门,那么 IP 地址就好比它上面的门牌号。

当然还需要设置其它设备参数,其中的一个是该设备可以处理的数据报长度的最大值,也称为最大传输单元(MTU,maximum transfer unit),其它的属性将在后面陆续介绍。

5.3 IP 地址

前面提到IP 网络协议理解的地址是一个 32 位数值。在网络环境中每个主机都必须分配一个地址。如果你使用的是一个没有和其它网络进行TCP/IP 传输的局域网,就可以根据个人的爱好随意进行分配。然而对于 Internet 上站点的 IP 地址是由一个称为网络信

息中心(NIC, Network Information Center) 的中心机构制定的。

为了便于阅读, IP 地址被分成四个 8 位数字, 称为“8 位位组”。例如 quark 的 IP 地址是 0x954C0C04, 通常写为 149. 76. 12. 4。这种格式也称为点分十进制表示(dotted decimal notation)。

采用这种标记的另一个原因是 IP 地址被分成网络号和主机号两个部分, 也就是说 IP 地址由两个部分组成: 前面的一部分是网络号, 后面一部分是主机号。当向 NIC 申请 IP 地址时, 它并不是为你想使用的每一台主机都分配一个 IP 地址, 而是给你一个网络号, 你可以根据自己的爱好将此范围类的有效 IP 地址随意指定给自己网络上的主机。

根据网络规模的大小, 主机号所占用的部分也有大小之分。为了满足不同的需要, 有几种不同类型的网络, 它们分别定义了对 IP 地址的不同划分方式, 见表 5.1。

表 5.1

网络类型	对 IP 地址划分方式
A 类	A 类网络地址中, 网络号仅占用第一个 8 位位组, 即其包含从 1. 0. 0. 0 到 127. 0. 0. 0 网络。它为主机号部分提供了 24 位, 可以允许每个网络上有 160 多万主机。
B 类	B 类包含的网络是从 128. 0. 0. 0 到 191. 255. 0. 0; 网络号是前面两个字节, 这样总共有 16 320 个网络, 每个网络可以包含 65 024 台主机。
C 类	C 类网络是从 192. 0. 0. 0 到 223. 255. 255. 0, 前三个字节是网络号, 其总共包含近 200 万个网络, 每个网络可以最多包含 254 台主机。
D, E, F 类	从 224. 0. 0. 0 到 254. 0. 0. 0 的地址或者是试验用的, 或者是为了将来的应用而保留的, 没有指定任何网络。

例如, IP 地址为 149. 76. 12. 4 的 quark, 就表明其为 B 类网络 149. 76. 0. 0 上的 12. 4 号主机。

你可能已经注意到在上述的各类网络的列表中, 主机部分的各位并不是可以取任何值的。这是因为主机部分的各位全为 0 或者全为 1 都是保留的号, 用于特殊的目的。当主机部分的所有位全为 0 时指整个网络的地址, 主机部分全为 1 是广播地址, 利用它可以同时访问该网络上的所有主机。因此 149. 76. 255. 255 并不是一个有效的主机的地址, 而是指网络 149. 76. 0. 0 上的所有主机。

另外还有两个网络地址是保留的, 它们是 0. 0. 0. 0 和 127. 0. 0. 0。第一个地址称为缺省路径, 后者称为回送地址(loopback address)。缺省路径与 IP 对分组进行路径选择有关, 下面将会详细介绍。

网络 127. 0. 0. 0 保留用来在你的主机内部进行 IP 传输。通常地址 127. 0. 0. 1 指定一个称为回送接口(loopback interface)的特殊接口, 它的行为同一个循环的线路类似。任何从 TCP 或 UDP 传递给它的数据马上返回, 就像它们刚刚从网络到达一样。这就允许你在没有实际联网的情况下进行网络软件的开发和测试。

5.4 地址解析

现在知道了IP地址是如何组成的,你可能会对如何在以太网上用它来区分不同的主机感到疑惑,因为以太网协议通过6个字节的地址来区分各主机,而此地址与IP地址是没有任何关系的。因此必须要有一种机制将IP地址映射到以太网的地址,这就是所谓的地址解析协议(ARP, Address Resolution Protocol)。实际上ARP并不仅仅在以太网中使用,在其它类型的网络中也是需要的。

当ARP希望找到与给定的IP地址相应的以太网地址时,它使用以太网的广播特性,将一个分组同时发送给网络上的所有站点。ARP发送的分组包含此IP地址的请求信息。每个接收到此分组的主机都将此IP地址与其本身的IP地址进行比较,如果地址匹配,就向查询的主机返回一个ARP应答。查询的主机可以从应答中分解出发送者的以太网地址。

你可能会感到迷惑,怎么知道在全世界那么多的网络中此主机是位于那个网络,此网络是否是以太网呢? 这个问题涉及到所谓的路由问题,即找出网络中一个主机的物理位置。我们将在5.5节"IP路由"中进行详细介绍。

现在再对ARP进行一些讨论。一个主机一旦发现了一个以太网的地址后,就把它存储在ARP缓冲区中,这样下一次向此主机发送分组时就不再需要进行查询了。然而把这些信息永久保存是不明智的,例如远程主机的以太网卡因为技术缘故进行了更换,这样ARP中保留的信息就失效了。为了可以进行下一次查询,ARP高速缓存中的项一般在过一定时间后就自动放弃掉。

有时也需要找出给定的以太网卡地址主机的IP地址。例如一个无盘机器希望从网络上的一个服务器启动时就会遇到这种情况,这种情形在局域网中是很常见的。一个无盘的客户实际上除了自己以太网卡的地址外不知道自己的其它任何信息。因此其最初需要做的是广播一个消息,让其启动服务器告知自己的IP地址。这里使用的是另外一个协议,称为逆向地址解析协议(RARP, Reverse Address Resolution Protocol),同时还需使用BOOTP协议,它定义了网络上无盘客户的启动过程。

5.5 IP 路 由

5.5.1 IP网络

在给别人写信的时候,需要在信封上注明收信人的详细地址,包括其所在的国家,省,县等。当把信投入信箱以后,邮局会将其投递到目的地:它首先被送到指定的国家,然后由该国家的邮局再将它投递到正确的省,县等。这种分层次的方式好处是非常明显的:当寄信的时候,当地邮局只需要大体上知道此封信往哪儿转发即可,而不必要知道信在目的国家内是如何投递的。

IP网络的构成与此类似。整个Internet由一些称为自治系统(autonomous system)的网络组成。每个自治系统进行其内部主机间的路由工作,这样发送一个数据报的工作就

简化为寻找目的主机所在的网络。这就是说,当数据报传递到其目的主机所在的网络上的任何主机后,进一步的处理完全由该网络在内部进行。

5.5.2 子网

子网的结构是通过将 IP 地址分成主机号和网络号两个部分来反映的。缺省情况下,目的网络的地址是从 IP 地址的网络部分获取的。这样 IP 地址中具有相同网络号的主机应该处于同一个网络,反过来,处于同一个网络的主机的 IP 地址中网络号部分应该相同。

由于一个网络本身又可能是有数百个小的网络如以太网组成,因此在此网络内部也可以采用同样的机制。即 IP 网络允许将一个 IP 网络再分成多个子网。

一个子网负责向其所在的 IP 网络的一部分 IP 地址发送数据报。对于 A,B 或 C 类网络来说是通过 IP 地址的网络部分来识别的。然而现在是将网络部分进行扩展,占用主机号部分的部分数据位。IP 地址中用来表示子网号码的数据的位数是通过子网屏蔽字,或称为网络屏蔽字给出的。它也是一个 32 位的数,规定了 IP 地址中网络部分的位屏蔽。

还是以 GMU 的校园网为例。它有一个 B 类网络地址 149.76.0.0,因此其网络屏蔽字为 255.255.0.0。

在 GMU 内部有许多子网,例如各个系的局域网。因此,在此范围的 IP 地址被分成 254 个子网,从 149.76.1.0 到 149.76.254.0。例如分配给理论物理系的网络号为 149.76.12.0。校园的主干网本身又是一个网络,其地址为 149.76.1.0。这些子网都具有相同的 IP 网络号,但是可以通过其 IP 地址的第三个字节区分开来。它们应该使用的网络屏蔽字为 255.255.255.0。

图 5.2 表明了对地址 149.76.12.4 的不同解释方式,当把它作为一般的 B 类网络和具有子网的解释是不同的。

图 5.2 一个 B 类网络的子网划分

需要注意的是子网的划分只是网络内部的划分。子网是由网络的所有者(或系统管理员)创建的。一般来说,子网是为了反映实际存在的边界而生成的,比如物理上分开的网络(如两个以太网),行政上分开的网络(比如两个系的网络)和地理上分开的网络,这些子网的管理都可以下发到一些具体的人。这种结构只对网络内部的行为有影响,对于网络外面的世界是完全屏蔽的。

5.5.3 网关

对网络进行子网的划分不但具有组织上的优点,而且还是硬件边界的自然结果。一个物理网络如以太网上的主机的访问范围是非常有限的:它只能和同一网络的主机进行直接的对话,对其它主机的访问只能通过所谓的网关(gateway)进行。网关是一个同时与两个或多个物理网络连接的主机,在这些网络之间进行数据报的交换。

为了使 IP 可以简单的识别出一个主机是否在本地的物理网络上,不同物理网络必须属于不同的 IP 网络。例如,假设数学系主机的网络号是 149.76.4.0,当向 quark 发送一个数据报时,erdos 上的网络软件根据其网络地址 149.76.12.4 判断出此目的主机属于其它

的物理网络,因此必须通过一个网关(缺省为 sophus)才能到达。

sophus 本身连接在两个不同的子网上:数学系和校园主干网。它通过两个不同的接口 eth0 和 fddi0 来分别对两个网络进行访问。那么现在的问题是给它设置什么 IP 地址呢?是给其子网 149.76.1.0 还是 149.76.4.0 上的地址呢?

回答是两个地址都要设置。它与数学系的局域网上的主机进行交谈时使用 IP 地址 149.76.4.1,当与主干网上的主机进行交谈是使用 IP 地址 149.76.1.4。

对每个子网都需给网关设置一个 IP 地址。这些地址和相应的网络屏蔽字,与访问子网的接口联系在一起。这样 sophus 接口和地址的映射,如表 5.2 所示:

表 5.2

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

最后一项描述的是回送接口。

图 5.3 是 Groucho Marx 大学校园网络的部分拓扑结构,同时处于两个网络的主机设定了两个 IP 地址。

图 5.3 Groucho Marx 大学的部分网络拓扑结构

一般来说,可以忽略将一个地址与主机或接口相关联的细微差别。对于仅在一个网络上的主机如 erdos,就可以称此主机具有此 IP 地址,尽管从严格意义上来说是它的以太网接口具有此 IP 地址。然而,对于一个网关来说这种区别就显得重要了。

5.5.4 路由表

现在把注意力集中到当把一个数据报向远程网络发送时 IP 如何选择使用哪一个网关。

前面已经提到,当给 quark 发送一个分组时,erdos 对目的地址进行检查,发现其不在本地网络上。因此把分组发送给缺省的网关 sophus,而网关也面临着同样的问题。sophus 识别到 quark 不在自己直接相连的任何网络上,因此必须寻找另外一个网关对数据进行转发。正确的选择应该是 niels,即到物理系的网关。sophus 因此需要一些信息确定合适的网关与目的网络联系。

IP 使用的路由信息基本上是一个将网络和到达它们的网关相连的表。同时还必须提供一个缺省的项,这个网关与网络 0.0.0.0 相关联。所有不知道目的网络的分组都由此路径发送。在 sophus 上,如表 5.3 所示:

表 5.3

Network	Gateway	Interface
149.76.1.0	-	fddi0
149.76.2.0	149.76.1.2	fddi0
149.76.3.0	149.76.1.3	fddi0
149.76.4.0	-	eth0
149.76.5.0	149.76.1.5	fddi0
...	...	...
0.0.0.0	149.76.1.2	fddi0

到 sophus 直接相连的网络不需要网关,因此其相应的网关项是“ - ”。

路由表可以通过各种方式建立。对于小的局域网来说,通常最有效的办法是手工建立,并使用 IP 命令在系统启动时为其填充 IP 地址。对于大型网络来说,可以通过路由监控进程在运行过程中进行建立和调整;这些监控程序运行在网络的中心主机上,并相互交换路由信息以计算出网络成员间的最佳路径。

根据网络规模的大小,通常使用不同的路由协议。在一个自治系统内部的路由使用的是内部路由协议(internal routing protocols)。其最主要的部分是 RIP,即所谓的 Routing Information Protocol(路由信息协议),它是由 BSD 的 routed 监控程序实现的。在自治系统之间的路由使用的是外部路由协议(External Routing Protocols),如 EGP(External Gateway Protocol), BGP(Border Gateway Protocol),这些(包括 RIP)都在 Cornell 大学的 gated 监控进程中进行了实现。

5.5.5 度量值

基于 RIP 的动态路由根据步长(hop)数来确定到目的主机或网络的最佳路径,此步长是指数据报在传输过程中所经过的网关的个数。路径越短,RIP 就选用它。非常长的路径,如步长为 16 或更多,就认为此路径不可用而放弃它。

如果希望使用 RIP 来管理网络内部的信息,就必须在所有的主机上都运行 gated。gated 在系统启动的时候检查所有活动的网络接口。如果有一个以上的活动接口(不包括循环返回接口),它假设此主机在数个网络之间进行分组的交换,就会动态的交换和广播路由信息。否则,只是被动的接收 RIP 更新信息对本地的路由表进行更新。

当从本地路由表开始广播信息时,gated 根据与路由表中的每一项相关联的所谓的度量值(metric value)来计算路径的长度。度量值是由系统管理员在配置此路径时设置的,应该反映使用此路径的开销。因此到此主机直接相连的子网的路径的度量值应该为 0,而通过两个网关的路径的度量值为 2。如果没有使用 RIP 或者 gated,就无需考虑所谓的度量值。

5.6 网间网互连控制消息协议

与 IP 相伴的还有另外一个协议,这就是网间网互连控制消息协议 ICMP(Internet Control Message Protocol),核心的网络代码使用此协议与其它主机进行出错消息的通信。例如,你在主机 erdos 上,希望 telnet 到 quark 主机的 12345 端口,但是却没有进程监听此端口。当第一个发向此端口的 TCP 分组到达 quark 时,网络层就会识别出来,并立刻向 erdos 返回一个 ICMP 消息,声明此端口不可访问。

ICMP 中的消息数量很多,许多是用来处理出错情况的。然而其中有一个非常有趣的称为重定向(Redirect)的消息,它是由路由模块在探测到另外一个作为网关的主机时生成的消息,而不管是否还存在更短的路径。例如,系统启动以后 sophus 路由表中的信息可能是不完全的,它只包含到数学系网络,到 FDDI 主干网的路径,以及缺省的指向 Groucho Computing Center 网关(gcc1)的路径。因此任何发向 quark 的分组都将送往 gcc1,而不是通往物理系的网关 niels。当收到这种分组后,gcc1 将会注意到这种路径选择很不合理,就会将数据转发到 niels,同时向 sophus 返回一个 ICMP 重定向消息告诉它更合理的路径。

现在看来这是一个非常聪明的方法,可以避免手工设置基本的路由信息。然而需要提醒读者的是完全依赖于动态路由机制,无论是 RIP,或者是 ICMP 的重定向消息,并不一定总是一种好的想法。ICMP 重定向和 RIP 给你很少或者没有机会对一些确实授权的路由信息进行验证。这就允许一些怀有恶意的人破坏整个网络的交通,或者做一些更坏的事情。由于这个原因,有一些版本的 Linux 网络代码仅处理影响网络路由的重定向消息,就好像其只能进行主机路径的重定向。

5.7 域名系统

5.7.1 主机名字解析

如上所述,TCP/IP 网络中的寻址主要围绕的是 32 位的 IP 地址。然而要想记住多个这样的地址是非常困难的事。因此主机还是使用通常的名字,如 gauss,strange 等。从这种名字到 IP 地址的转换是网络应用程序的责任。这个过程称为主机名字解析(host name resolution)。

希望查出给定名字主机 IP 地址的应用程序并不需要自己提供进行 IP 地址和主机查找的例程,而是使用一些库函数,如 gethostbyname(3),gethostbyaddr(3)等来透明的进行这些工作。传统上,这些库函数以及一些相关的库例程都组装在另外一个库中,称为解析库(resolver library)。在 Linux 上,它是作为标准 libc 的一部分。口语上就把这些函数的集合称为解析器(resolver)。

在一个小的网络如以太网,或者是这些网络的一个集合中,保存一个所有主机名字到 IP 地址的映射表是不太困难的事。这些信息通常保存在一个名叫/etc/hosts 的文件中。当增加或去掉主机,或者重新分配 IP 地址时,需要做的工作就是更新所有主机上的此文件。很明显,当网络由大量主机构成时,这种方法就显得非常笨拙。

一种解决办法是采用 Sun Microsystem 的网络信息系统 NIS(network information system),它也常称为黄页 YP(yellow pages)。NIS 将所有的 hosts 文件以及其它信息都存储在一个中央控制主机的数据库中,客户在需要的时候可以向它进行查询。然而这种方法也只适用于中型规模的网络,因为它需要保持一个所有 hosts 的中心数据库,并将其分发到所有服务器。

在 Internet 上,最初的地址信息也是存储在一个 HOSTS.TXT 数据库文件中。这个文件由网络信息中心 NIC(Network Information Center)维护,并且所有参与的站点都需要保存和安装此文件。随着网络规模的增长,这种方案带来了几个问题:除了经常安装 HOSTS.TXT 文件的管理开销以外,负责分发它的服务器负担越来越重。更严重的是所有的名字都必须到 NIC 进行登记,并且必须保证没有重名。

这就是为什么在 1984 年采用了一种称为域名系统 DNS(Domain Name System)的新名字解析机制。DNS 由 Paul Mockapetris 设计,可以同时解决上述的一些问题。

5.7.2 进入 DNS

DNS 按分层次的域来组织主机名。一个域是一些在某种意义上相关联的站点的集合:或者它们构成一个实际的网络(如一个校园的所有机器,或者 CERNET 上的所有主机);或者它们都属于一个特定的组织(如美国政府组织);或者是它们在地理位置上比较靠近。例如,大学都组织在 edu 域中,每个大学(或学院)在一个子域下组织它们的主机。GMU 的域名可能是 groucho.edu,数学系局域网的域名是 maths.groucho.edu。各系网络上主机的域名是将其主机名附加在系的域名上。如 erdos 的域名是 erdos.maths.groucho.edu。这就称为全称域名 FQDN(Full Qualifed Domain Name),它在全世界范围

内唯一确定了此主机。

图 5.4 显示了域名空间的一部分。此树型结构的根节点,由一个“.”标记的,称为根域(root domain),它包含其它所有的域。为了指示一个主机的名称是全称域名,而不是一个相对与局部域的名称,有时在它的名字后面以“.”结尾,以表明此名称的最后一部分是根域。

图 5.4 部分域名空间示意图

根据在域名层次中的位置,一个域可以称为是高层,第二层,或者第三层。还可以划分更多的层次,但是一般很少见。表 5.4 是一些经常看到的高层域。

表 5.4

高层域名	功能
edu	(决大多数美国)教育机构如大学等。
com	商业组织,公司等。
org	非商业性组织。通常私人的 UUCP 网络都属于此域。
net	网关和网络上其它管理主机。
mil	美国军事机构。
gov	美国政府机构。
uucp	官方的,所有以前使用 UUCP 名字的站点的名字都移到此域下。

从技术上来说,前四个是属于 Internet 的美国部分,但是也可以看到一些非美国站点也在这些域中,如 net 域。但 mil 和 gov 域只有美国使用。

美国以外的每个国家使用的高层域名是 ISO-3166 中定义的两个字母的国家编号。例如,芬兰使用的域名是 fi,法国是 fr,德国是 de,中国是 cn,澳大利亚是 au 等。在此高层的域名之下,每个国家的 NIC 可以根据它喜欢的任何方式来组织主机名。例如,澳大利亚使用的是与国际上一级域名类似的二级域名,如 com.au, edu.au,等。其他国家如德国,没有使用此额外的一级,而是使用稍长的直接指向运行一个特殊域的组织。例如经常可以看到类似与 ftp.informatic.unierlangen.de 的名称。

当然,这些国家域名并不意味着在此域名下的主机就一定位于该国家;它只是表明此主机是通过此国家的 NIC 进行登记注册的。一个瑞典制造厂可能在澳大利亚有一个分厂,而此分厂的主机还是利用一级域 se 进行登记注册。

除此之外,DNS 还作了更多的事情:它允许将一个子域的管理授权给它的系统管理员。例如,Groucho 计算中心的维护人员可能为每个系都分配一个子域,我们在前面已经遇到了 maths 和 phycics 子域。当他们发现物理系的网络太大,而从外面管理非常混乱时,就可以简单的将 physics. groucho. edu 域的控制权转交给该网络的系统管理员。这样物理系内部的人员就可以任意使用所喜欢的主机名,任意从他们的网络中分配 IP 地址,而不会对外部造成影响。

为了此目的,名字空间被分成许多区(zone),每个区以一个域作为根。请注意区和域的细微差别:groucho. edu 域包含 GMU 的所有主机,而 groucho. edu 区只包含计算中心直接管理的主机,如数学系的主机。物理系的主机属于另外一个区,即 physics. groucho. edu。在图 5. 4 中,每个区的开始都在其域名的右面标记了一个小圆。

5. 7. 3　DNS 中的名字查询

乍一看来,所有的这些域或区使得名字解析变成非常复杂。并且没有集中的机构控制名字如何分配到各个主机。

现在我们讨论 DNS 最巧妙的部分。当希望查出 erdos 的 IP 地址的时候,DNS 会告诉你去问管理它的人,由它们告诉你正确的结果。

实际上,DNS 是一个巨大的分布式数据库。它是通过由所谓的名字服务器(name server)提供一个指定的域或一些域的信息来实现的。在每一个区,至少有两个以上的保存其所在区的所有主机授权信息(authoritative information)的名字服务器。为了获得 erdos 的 IP 地址,就必须与 groucho. edu 区的名字服务器联系,它将会为你返回想要的数据。

说起来容易做起来难,你可能会这么想。例如,如何才能访问到 GMU 的名字服务器呢? 当主机没有安装地址解析预言(oracle)时,DNS 也提供了此功能。当应用程序想查找 erdos 的信息,它就与本地的域名服务器联系,进行所谓的重复查询。它通过向根域的名字服务器发送一个请求,查询 erdos. maths. groucho. edu 的地址。根名字服务器认识到此名称不属于自己管理的区,而是属于 edu 下的一个域。就告诉你去联系一个 edu 区的名字服务器以获得更多的信息,并给你发送一个所有 edu 名字服务器地址的列表。你的本地名字服务器会继续向这些列表中的某个名字服务器进行查询,比如 a. isi. edu。通过与根名字服务器类似的方式,a. isi. edu 知道了 groucho. edu 运行的是它们自己的一个区,就把它们的名字服务器指给你。本地名字服务器然后就向这些服务器中间的一个查询 erdos 的地址,该名字服务其识别出此名称属于本区,就返回其相应的 IP 地址。

现在看来查找一个 IP 地址产生了很大的传输量。但是与我们坚持使用 HOST. TXT 相比,此数据传输量是微不足道的。并且这种方式还有改进的余地。

为了进一步提高查询的响应速度,名字服务器会将其获得的信息存储在本地 cache 中。这样当你的本地网络希望查询地址属于 groucho. edu 域的主机的地址时,名字服务器

就不需要重复上面的整个过程,而是直接与 groucho. edu 的名字服务器联系。

名字服务器当然不会永远保存这些信息,而是在一定时间之后自动抛弃掉。这个更新的时间间隔称为生存时间 TTL(Time To Live)。DNS 数据库中的每一项数据都由此区的系统管理员设置了相应的 TTL。

5.7.4 域名服务器

保存一个区内所有主机信息的名字服务器称为是这个区的权威(authoritative),有时也称为主要名字服务器(master name servers)。对此区内所有主机的查询最终会在这些主要名字服务器中得到答案。

为了提供一个区的一致的映象,它的主要名字服务器之间必须有很好的同步。实现方法是将其中的一个设置成为主(primary)服务器,由它从数据文件加载本区的信息,其它后备的服务器定时地从主要服务器获取信息。

拥有数个名字服务器的一个目的是分散工作的负载,另一个目的是容错。当一个名字服务器因为系统崩溃或失去网络连接而停止工作时,所有的查询请求将反馈到其它服务器。此机制不能避免对所有 DNS 请求都返回错误信息的情况,例如,服务器程序本身具有的缺陷。

也可以考虑运行一个不管理任何域的名字服务器。这种服务器是有用的,它只为运行在本地网络的应用程序进行 DNS 查询,并缓存这些信息。这种服务器称为只缓存(caching-only)服务器。

5.7.5 DNS 数据库

在上面已经看到 DNS 不仅处理有关主机 IP 地址的事情,而且在名字服务器之间交换其它信息。实际的 DNS 数据库有一整套不同类型的数据项。

DNS 中的一条信息称为一个资源记录 RR(Resource Record)。每一个记录有一个类型与其相关联,描述了它代表的数据的类型和适用的网络类型。后者是为了适应不同编址机制的需要,例如 IP 地址(IN 类),Hesiod 网络的地址(MIT 使用),以及其它一些。最基本的资源记录类型是 A 记录,它将一个全称的域名与一个 IP 地址相关联。

当然,一个主机可能有一个以上的名字。然而其中必须有一个官方的名称,或者称为正规主机名,而其它只是指代前者的别名。区别在于正规的主机名有一个 A 记录与之相关联,而其它的只有一个指向正规主机名的 CNAME 记录类型。

这里不打算介绍所有的记录类型,只是给出一个简单的例子。下面是装入到 physics. groucho. edu 区的名字服务器的域数据库的一部分。

```
;
; Authoritative Information on physics. groucho. edu
@               IN   SOA     {
                niels. physics. grouchoedu.
                hostmaster. niels. physics. groucho. edu.
                1034                ; serial no.
```

```
                360000          ; refresh
                3600            ; retry
                3600000         ; expire
                3600            ; default ttl
                }
;
; Name servers
                IN   NS     niels
                IN   NS     gauss.maths.groucho.edu.
gauss.maths.groucho.edu.      IN   A      149.76.4.23
;
; Therretical Physics ( subnet 12)
niels           IN   A          149.76.12.1
                IN   A          149.76.1.12
nameserver      IN   CNAME niels
otto            IN   A          149.76.12.2
quark           IN   A          149.76.12.4
down            IN   A          149.76.12.5
strange         IN   A          149.76.12.6
...
; Collider Lab (subnet 14)
boson           IN   A          149.76.14.1
muon            IN   A          149.76.14.7
bogon           IN   A          149.76.14.12
```

除了 A 和 CNAME 记录以外，可以看到在此文件的头部有一个特殊的记录，延伸了好几行。这是 SOA 资源记录，表示权威初始（Start Of Authority），保存的是服务器管理区的一般信息。例如其中包含了所有记录缺省的生存时间。

注意到在此例文中所有没有以圆点结尾的名字都被被解释成相对于 groucho.edu 域。SOA 记录中的特殊名字@是指此域本身。

还可以看到 groucho.edu 域的名字服务器必须知道 physics 区的一些信息，以便将查询指到它们的名字服务器。这是通过一对记录实现的：NS 记录给出服务器的 FQDN，A 记录将一个地址与此名字相关联。由于这些记录把名字空间都保存在一块，经常称为线索记录（glue records）。它们是父区实际保存子区主机上信息的唯一记录。指向 physics.groucho.edu 的名字服务器的线索记录如下所示

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

```
                hostmaster.vax12.gcc.groucho.edu
                233         ; serial no
                360000      ; refresh
                3600        ; retry
                3600000     ; expire
                3600        ; default ttl
            }
...
;
; Glue records for the physics.groucho.edu zone
physics             IN      NS      niels.physics.groucho.edu
                    IN      NS      gauss.maths.groucho.edu
niels.physics       IN      A       149.76.12.1
gauss.maths         IN      A       149.76.4.23
.......
```

5.7.6 逆向查询

除了查询一个主机的IP地址外,有时还需要查询与一个地址相对应的标准主机名。这个过程称为逆向映射,有一些网络应用程序使用它来验证一个客户的身份。当使用单个hosts文件时,逆向查找只是简单的查找此文件,找到与此地址相应的主机名。

逆向域名解析的一个明显的问题是同一IP地址可能对应于若干域名,这给逆向解析带来一定的困难。另外,没有任何方法可以不经搜索整个服务器组而找到解析逆向查询的服务器。原因很简单,IP地址与域名服务器树形结构没有任何关系,要找到合适的映射,只有逐个搜索服务器。这也是虽然域名系统一产生就有逆向查询,但逆向查询并未得到广泛应用的原因。

为了解决上述问题,域名系统提供了一种特殊形式的逆向解析,为此专门构造一个特别的域及一种特别报文,这种报文叫作"指针查询"(pointer query)。在指针查询中,欲解析的IP地址被表达成一种像域名一样的可显示串形式,后缀为一个特殊的逆向解析域,域名为"in-addr.arpa"。比如一个以直观的点分十进制形式表示的IP地址:

aaa.bbb.ccc.ddd

其指针查询表达方式为:

ddd.ccc.bbb.aaa.in-addr.arpa

两种表达方式中IP地址部分顺序恰好相反,因为域名结构是自底向上(从子域到域),而IP地址结构是自顶向下(从网络到主机)。实质上逆向解析是将IP地址表达成一个域名,这样逆向解析的很大部分(包括服务器通信端口、算法甚至软件)可以纳入正向解析中。由于本地服务器不一定是in-addr.arpa或arpa域服务器,所以逆向解析可能要求助于其它服务器。为此,在Internet的根服务器中专门提供了一个数据库,内含可逆向解析的有效IP地址及有关域名服务器的信息。

例如,IP 地址 149. 76. 12. 4 相应于 4. 12. 76. 149. in-addr. arpa。将这些名字和它们的正规主机名相连的资源类型是 PTR。

生成一个权威的区通常意味着它的系统管理员被授予了如何将地址分配给名字的全部控制。由于其手下通常有一个或多个 IP 网络或子网,在 DNS 区和 IP 网络间有一到多的映射。例如物理系由子网 149. 76. 8. 0, 149. 76. 12. 0 和 149. 76. 14. 0 组成。

作为相应的结果, in-addr. arpa 域的新区必须和 physics 区一同创建,并授权给该系的网络系统管理员: 8. 76. 149. in-addr. arpa, 12. 76. 149. in-addr. arpa 和 14. 76. 149. in-addr. arpa。否则,在 Collider 试验室安装一个新的主机需要联系它们的父域,使得新地址进入它们的 in-addr. arpa 区文件。

第 12 号子网的区数据库如下所示:

```
;
; the 12. 76. 149. in-addr. arpa domain
@           IN      SOA     {
                    niels. physics. groucho. edu
                    hostsmaster. niels. physics. groucho. edu
                    233 360000 3600 36000000 3600
                    }
2           IN      PTR         otto. physics. groucho. edu
4           IN      PTR         quark. physics. groucho. edu
5           IN      PTR         down. physics. groucho. edu
6           IN      PTR         strange. physics. groucho. edu
```

下面是 12 号子网的父域数据库中相应的线索记录:

```
;
; the 76. 149. in-addr. arpa domain.
@               IN      SOA     {
                vax12. gcc. groucho. edu.
                hostmaster. vax12. gcc. groucho. edu.
                233 360000 3600 3600000 3600
                }
. . . . . .
; subnet 4:  Mathematics Dept.
1. 4            IN      PTR     sophus. maths. groucho. edu.
17. 4           IN      PTR     erdos. maths. groucho. edu.
23. 4           IN      PTR     gauss. maths. groucho. edu.
. . .
; subnet 12:  Physics Dept, separate zone
12              IN      NS      niels. physics. groucho. edu
                IN      NS      gauss. maths. groucho. edu
```

niels. physics. groucho. edu.　　IN　　A　　149. 76. 12. 1

gauss. maths. groucho. edu.　　IN　　A　　149. 76. 4. 23

这种方法的一个重要后果是所有的区只能作为 IP 网络的一个超集生成，更严重的是网络屏蔽字必须在字节边界上。GMU 的所有子网的网络屏蔽字都是 255. 255. 255. 0，因此对每个子网可以生成 in-addr. arpa 区。然而，如果网络屏蔽字是 255. 255. 255. 128，则为子网 149. 76. 12. 128 生成一个这样的区是不可能的，因为此时 12. 76. 149. in-addr. arpa 域被分成两个权威的区，其主机名字分别从 1 到 127 和 128 到 255，而没有办法将此信息告诉给 DNS。

第六章　网络的安装与配置

本章主要介绍 Linux 系统中 TCP/IP 网络以及 SLIP, PPP 等的配置问题。

6.1　网络软件及工具的获取

对 Linux 网络软件进行配置之前, 必须获得它的各个组成部分, 包括现有版本的操作系统核心代码(版本 2.0 或更新版本)、正确的系统函数库(libc)、TCP/IP 配置程序和文件(如/etc/ifconfig, /etc/hosts 等)以及一些网络应用程序(如 telnet, ftp, rlogin 等)。

Linux 独特的产生过程导致了其与一些商品化的操作系统在推广上的差别。在 Internet 上有许多站点都有 Linux 操作系统的相关内容、应用程序等。操作系统的不同部分可能分布在不同的站点。但也有些站点收集了比较全的内容, 同时提供了良好的安装工具。例如 sunsite.unc.edu 和 tsx-11.mit.edu 等站点的 slackware。在国内的一些 ftp 站点也有 slackware 供用户免费获取, 如 ftp.cernet.edu.cn, ftp.pku.edu.cn, ftp.xanet.edu.cn 等。如果你是从这些站点或类似的站点获得的 Linux, 其中可能已经包含了所有相关的网络部分代码和程序。不妨先检查一下, 如果已经有了, 就可以跳过本节, 直接阅读与网络相关的核心配置部分。

6.1.1　核心源码

Linux 的核心版本在不断地升级, 新的版本与老版本相比, 不仅更正了一些错误, 最主要的是提供了更多的驱动程序。一般来说, Linux 内核版本号中第二位数为偶数的是正式发行版本, 如 1.2.* , 而第二位数为奇数的版本是测试版, 如 1.1.* , 1.3.* 等。

下面的 FTP 站点有较新的正式核心版本:

sunsite.unc.edu

/pub/Linux/kernel/v2.0/linux-2.0.tar.gz

这是一个利用 gzip 进行压缩的文件, 其展开和安装步骤如下:

```
# cd /usr/src
# mv linux linux.old
# gzip -dc linux-1.2.13.tar.gz ©¦tar xvf -
```

该站点在此目录下可能会有一些类似于名叫 patch-2.0.1.gz 的文件。这是一些升级软件。例如, 你现在如果使用的是 2.0.1 版本的内核, 这就意味着你将 2.0 的核心版本进行了一次升级。如果该目录下有许多比你现在使用的核心版本高的升级文件, 就需要将这些文件都取过来, 按顺序用下面的命令形式对现有版本进行升级:

```
# cd /usr/src
# for patchfile in ../patch*
```

```
> do
> gzip -dc $patchfile ©¦patch -p0 2> > patch.errs
> done
...
```

在命令完成之后应该检查输出文件 patch.errs,看有没有出错信息。如果有出错信息,就表明至少有一个升级文件没有正确地运行。如果出现了这种情况,就需要逐个手工运行升级文件以找到出错的地方,检查出错的原因并进行修改。

6.1.2 函数库

函数库 libc 的版本最少应该是 4.4.2,因为在此以前的版本有一些影响网络屏蔽的问题。现在的 a.out 库(libc-4.6.27)可以在下面的地址查到:

```
sunsite.unc.edu
    /pub/Linux/GCC/
```

最少需要下列文件:

* image-4.6.27.tar.gz
* inc-4.6.27.tar.gz
* extra-4.6.27.tar.gz
* release.libc-4.6.27.tar.gz

在安装 libc-4.6.27 之前最好阅读 release.libc-4.6.27。需要注意的是使用 libc-4.5.26 及以后的版本,GCC 的版本也必须在 2.6.2 以后,Linux 内核的版本也必须在 1.1.52 以后。

6.1.3 网络配置的工具

进行网络配置需相应的工具,现在的 NET-2 工具包可以在下面的地址找到:

```
ftp.linux.org.uk
    /pub/linux/Networking/PROGRAMS/NetTools/net-tools-1.2.0.tar.gz
```

由于核心部分的网络代码在不断的修改,相应的工具软件也必须进行相应的修改,因此必须选择与你当前使用版本相配套的网络工具软件包。文件名反映了其支持的最早的核心版本。请选择小于或等于自己系统核心版本号的工具包。

网络工具软件的生成和安装如下:

```
# cd /usr/src
# mkdir net-tools
# gzip -dc net-tools-1.2.0.tar.gz ©¦tar xvf -
# make
```

make 命令将自动运行 Config.sh 文件,如果一切顺利,接着运行:

```
# make install
```

如果你当前的内核是 1.1.26 或更早的版本,则需要查找下列地点:

```
ftp.linux.org.uk
```

/pub/linux/Networking/PROGRAMS/Other/net032

在此目录中有三个版本的网络工具。下面列出了 net032 软件包和其对应的核心版本:

net-0.32d-net3.tar.gz　　1.1.12+

net-0.32b.tar.gz　　1.1.4+

net-0.32.old.tar.gz　　1.1.4 以前的核心

这些软件包中包含了必须的网络配置工具,如 ifconfig, route, netstat 等。

6.1.4 网络应用程序

通常还需要一些网络应用程序,如 telnet, ftp, finger 及它们所必须的监控进程。Florian La Roche 收集了比较完整的网络应用程序,包括二进制的代码和源代码。TCP/IP 应用程序和一些配置文件的例子存于下列 FTP 站点:

ftp.funet.fi

/pub/OS/Linux/PEOPLE/Linux/net-source/base/NetKit-A-0.08.bin.tar.gz

/pub/OS/Linux/PEOPLE/Linux/net-source/base/NetKit-B-0.0.6.tar.gz

如果该站点已经有了新的版本,可以选用较新的版本。

6.2 与网络相关的系统核心配置

在使用任何网络应用程序或者配置任何网络设备之前,必须保证操作系统核心程序提供了必要的网络支持。Linux 为支持比较广泛的设备配置,在预先编译好的核心程序中提供了对多种类型设备的支持。对一个具体的操作系统来说许多配置是多余的,而且占用了大量的内存空间,降低了系统的效率。因此最好的办法是根据自己系统的具体配置,重新编译新的适合本系统的内核程序。本节主要介绍在进行核心程序编译时如何选择与网络相关的选项,以构造一个支持网络功能及自己设备环境的最佳内核。

在重新编译核心程序之前要对核心进行配置。核心的配置是通过运行一个配置程序完成的,命令如下:

```
# cd /usr/src/linux
# make config
```

此程序提出一系列的配置选项,通过回答 y 或 n 来进行选择或放弃。其中有四个部分与网络配置有关。它们是一般设置、网络选项、网络设备支持和文件系统四个部分。其中最难配置的是网络设备支持部分,它的功能是选择你希望配置的物理设备的类型。一般来说,除了此部分外,其他部分都可以采用缺省设置。我们先对整个配置过程作大概的了解,然后再解释各选项的具体含义:

```
*
*  General setup
*
...
```

...

Networking support(CONFIG- NET)[y]y

...

...

在一般设置部分只需要选择是否支持网络,这里自然要选择支持,即回答 y。

*

* Networking options

*

TCP/IP networking(CONFIG- INET)[y]

IP forwarding/gatewaying (CONFIG- IP- FORWARD)[n]

IP multicasting (CONFIG- IP- MULTICAST)[n]

IP firewalling (CONFIG- IP- FIREWALL)[n]

IP accounting (CONFIG- IP- ACCT)[n]

*

* (It is safe to leave these untouched)

*

PC/TCP compatibility mode (CONNETC- INET- PCTCP)[n]

Reverse ARP (CONFIG- INET- RARP)[n]

Assume subnets are local (CONFIG- INET- SNARL)[y]

Disable NAGLE algorithm (normally enabled) (CONFIG- TCP- NAGLE- OFF)[n]

The IPX protocol (CONFIG- IPX) [n]

*

*

第二部分允许你选择或取消一些有趣的特性,一般来说选择缺省的设置就完全可以。如果你确实有兴趣了解它们的作用,可以进行自己的选择。后面我们将对各选项的具体含义作简要介绍。

*

* Network device support

*

Network device support? (CONFIG- NETDEVICES)[y]

Dummy net driver support (CONFIG- DUMMY) [n]

SLIP (serial line) support (CONFIG- SLIP) [y]

CSLIP compressed headers (CONFIG- SLIP- COMPRESSED) [y]

16 channels instead of 4 (SL- SLIP- LOTS) [n]

PPP (point-to-point) support (CONFIG- PPP) [y]

PLIP (parallel port) support (CONFIG- PLIP) [n]

Do you want to be offered ALPHA test drivers(CONFIG- NET- ALPHA) [n]

```
Western Digital/SMC cards (CONFIG_NET_VENDOR_SMC) [y]
WD80*3 support (CONFIG_WD8x3) [y]
SMC Ultra support (CONFIG_ULTRA) [n]
AMD LANCE and Pcnet (AT1500 and NE2100) support (CONFIG_LANCE) [n]
3COM cards ( CONFIG_NET_VENDOR_3COM) [n]
Other ISA cards (CONFIG_NET_ISA) [n]
EISA, VLB, PCI and on board controllers (CONFIG_NET_EISA) [n]
Pocket and portable adaptors (CONFIG_NET_POCKET) [n]
*
```

这一部分是最重要的部分,它要求你选择希望支持的硬件设备。

```
*
*  Filesystems
*
...
...
/proc filesystem support (CONFIG_PROC_FS) [y]
NFS filesystem support (CONFIG_NFS_FS) [y]
...
...
```

NFS 文件系统用于支持 NFS 客户和服务器程序。/proc 文件系统是必须的,因为许多网络应用程序和工具都要使用到它。

当完成系统的配置后就可以进行实际的核心编译了:

```
#  'make dep ; make clean'
#  make zImage ( zlilo / zdisk )
```

在重新编译内核后不要忘了进行检验,亦即使用新的内核,重新启动系统,通过运行一些应用程序来检查新的核心能否满足自己的要求并能够正常工作。

需要补充一点的是,在当前的 Linux 核心版本中,如 2.0 以及以后的版本中,对核心的配置除了上述 make config 命令外,还增加了字符状态下以菜单形式对核心进行配置的命令 make colormenu 以及 X 窗口系统中运行的图形配置界面命令 make xconfig。在后两种配置方式中,可以在一个界面中完成系统的各种配置选项,然后保存配置结果,进行相同的编译步骤。而不会像使用 make config 命令一样,如果某个选项选错了,则前功尽弃,必须从头开始再来一遍。

另外,随着 Linux 核心版本的不断更新,新版本的内核配置中会增加更多的选项,尤其是其支持的设备会有所增加。如果不能确定某些选项的取舍,最好是保留其缺省的设置。

6.2.1 部分网络选项的含义

在进行核心配置时有许多网络选项的问题。一般来说不需要进行修改,选择缺省的值

即可,但有些选项在特殊情况下是需要的。

TCP/IP networking　这一项的意义是十分明显的,它决定核心是否支持 TCP/IP 协议。

Dummy networking device　它允许 SLIP 和 PPP 用户在 Linux 机器上配置一个不依赖于已经建立的串行连接的地址。这是给 Linux 机器配置两个地址的简便方法。

IP forwarding/gatewaying　它决定当收到一个目的地址不是本地设备的分组时该怎么处理。如果你希望核心能够像一个 IP 路由器一样工作,就必须选择此项。大多数 SLIP 和 PPP 服务器都需要选择此项。

IP multicasting　这是支持 IP 广播的测试代码,例如"Internet Talk Radio"和 live video 等服务。提供这种服务还需要附加的程序支持,这里只是提供核心支持。

IP firewalling　这个选项可以为你的机器提供灵活的安全性选项。使你可以有选择的允许或禁止一定范围的 TCP/IP 端口。它也需要附加的程序支持。

IP accounting　这个选项适用与那些将他们的 Linux 机器提供商业性质的互连网络连接服务的用户。它可以统计每个端口或每个地址输入或输出的字节数。在此基础之上利用合适的软件工具就可以为每个使用你网络功能的用户进行记帐。

PC/TCP compatibility mode　在使用 PC/TCP 软件与 Linux 机器连接时有一个缺陷,选择此选项会弥补这个缺陷,但有时会降低网络的使用效率。

Reverse ARP　该选项允许将 RARP 协议配置到内核。它是为了允许启动 Sun 3 系统而加入的。除此之外没有其它的用处。

Assume subnets are local　此选项选择你是否假设整个子网都与自己的主机直接相连,或者通过网桥相连,或者在下一级又被分成多个子网。实际上让其保持缺省值几乎没有什么差异。

Disable NAGLE algorithm　这是一个决定什么时候传送一个分组的时间选项。缺省的选择在多数情况下可以最大限度的提供吞吐率,因此最好是选用缺省选择,不要修改,否则会降低系统的性能。

IPX protocol　此选项决定核心是否对 IPX 协议提供支持。IPX 是一个与 IP 功能相似的网际互连协议。大多数 Novell 系统都使用此协议。

Amateur Radio AX.25 Level 2　此选项决定是否在核心中编译入 Amateru Radio AX.25 协议。如果选择了此项,就会为编程提供一类新的网络套接字。

6.3　网络硬件配置

6.3.1　设备、驱动程序和接口

前面我们已经介绍了有关网络接口和 TCP/IP 问题的一些基本知识,但没有涉及当核心的网络代码访问一个硬件时究竟会发生一些什么事情这类的问题。为此,我们简单的讨论一下接口和设备驱动程序的有关概念。

为了使用某个具体的物理设备,如以太网卡,Linux 核心中必须有一些特殊的、知道如何对此设备进行访问的函数。这些函数称为设备驱动程序(device driver)。例如 Linux

中有几种商标的以太网卡的设备驱动程序,它们在功能上都非常相似。如以作者 Donald Becker 命名的“Becker 系列驱动程序”。

下面我们以以太网卡为例介绍设备驱动程序处理一个设备时都作了些什么。驱动程序必须能够和外设网卡上的逻辑器件进行交流:向网卡发送命令和数据,而网卡应该将接收到的数据传递给驱动程序。

在个人计算机上,这种通信是通过将网卡上的寄存器映射到内存的 I/O 区域实现的。核心发给网卡的命令和数据都必须通过这些寄存器进行。I/O 区域通常是通过给出其基地址描述的。例如,以太网卡典型的基地址是 0x300 或 0x320。

通常情况下是不用考虑这些硬件问题,如基地址等,因为核心在启动时会试图自动检测网卡的位置。其原理是核心从多个内存区域中读取一些数据,并将这些数据与某些网卡安装后这些内存区域应该具有的数据进行比较以判别网卡的类型。然而有些网卡是不能通过自动检测识别出来,另外,Linux 在启动时核心只检测一个网卡。因此如果使用了一个以上的网卡就必须通过一定的方法明确的将此网卡的信息告诉给核心程序。

另一个需要告诉核心程序的参数是中断向量的数值。硬件一般都是通过中断来请求核心对它的请求进行处理,如数据到达,缓冲区溢出等。在个人计算机中中断向量号一般是从 1 到 15。设置给硬件的中断号称为此设备的中断请求号码(Interrupt Request Number)。

同一种设备有许多种型号,为了便于核心对设备的统一存取,内核对设备的访问进行了抽象。即核心对设备的访问都是通过“接口”进行的。接口提供了对各种硬件都相同的一个抽象访问接口,例如发送和接收一个数据报的函数。

接口之间通过名字来区分。这些名字是在核心内部定义的,而不是 /dev 目录中的设备文件。通常以太网的接口是 eth0, eth1 等。接口到设备的分配通常是根据设备的配置顺序而定的,如第一个配置的以太网卡的接口名称为 eth0,第二个配置的为 eth1,等等。一个例外是 SLIP 接口,它的分配是动态的。也就是说无论何时建立一个 SLIP 连接,都为串行端口设置一个接口。

图 6.1 表明了硬件,设备驱动程序和接口三者之间的关系。

在系统启动时,核心会显示其检测到的设备和其安装的接口。下面是启动显示的一个例子:

```
...
This processor honours the WP bit even when in supervisor mode. Good.
Floppy driver(s): fd0 is 1.44M
Swansea University computer Society NET3.010
IP Protocols: ICMP, UDP, TCP
PPP line discipline registered.
SLIP: version 0.7.5 ( 4 chanels)
```

图 6.1　设备驱动程序、接口和硬件之间的关系

```
CSLIP: code copyright 1989 Regents of the University of California
dl0: D-Link DE-600 pocket adapter, Ethernet Address: 00: 80: C8: 71: 76: 95
checking 386/387 coupling ... Ok, fpu using exception 16 error reporting.
Linux version 1.1.11 (okir@monad) # 3 Sat May 7 14: 57: 18 MET DST 1994
```

它表明核心在编译时选择了 TCP/IP 支持，包含了 SLIP，CSLIP，PPP 的驱动程序。倒数第三行说明了其检测到了一个 D-Link DE-600 袖珍适配器(pocket adapter)，并将其安装在 dl0 接口。如果你使用的是另一种以太网卡，核心通常会显示一行以 eth0 开始的信息，后面是其检测到的网卡的类型。如果你安装了以太网卡但是在启动过程中没有显示类似的信息，表明核心没有正确的检测到你的网卡。下面将会在“6.3.2 Linux 网络设备”中介绍怎样处理这种情况。

6.3.2　Linux 网络设备

Linux 核心支持多种设备的多种硬件驱动程序。这里简要介绍其支持的几类设备驱动程序和它们使用的接口名称。

下面列出了 Linux 下各种接口的标准名称。许多驱动程序支持多个端口，这种情况下其命名按数字进行排列，如 eth0，eth1 等。

lo　回送接口。它用于测试目的，也有一些网络程序使用到它。它的运作向一个封闭的线路，任何写到它的分组立刻返回到本地主机的网络层。通常核心提供一个这样的端口，再少或再多这样的端口是没有什么意义的。

ethn　第 n 个以太网卡。这是绝大多数以太网卡的通用接口名称。

dln　这些端口用来访问另一类以太网设备 D-Link DE-600 袖珍适配器。特殊的是 DE-600 是通过并口驱动的。

sln　第 n 个 SLIP 接口。SLIP 接口根据分配给 SLIP 的顺序与相应的串行线路相联系，即为 SLIP 配置的第一条串行线路成为 sl0，依次类推。核心最多支持 4 个 SLIP 接口。

pppn　第 n 个 PPP 接口。与 SLIP 接口类似,PPP 接口与转换成 PPP 方式的串行线路相联系。核心最多支持 4 个 PPP 接口。

plipn　第 n 个 PLIP 接口。PLIP 协议通过并口传送分组。最多可同时支持三个 PLIP 接口。它们由 PLIP 驱动程序在系统启动是分配,并映射到并口上。

在下面的讨论中我们将重点介绍以上列出的一些驱动程序。

6.3.3　以太网安装

1. 支持的网卡

下面列出了 Linux 支持的部分常见类型的网卡。

3c503 3c503/16 3c509 / 3c509B 3c562 3c579 3c590 / 3c595

3c592 / 3c597 3c900 / 3c905

Accton MPX Accton EN1203, EN1207, EtherDuo-PCI

AT1500 AT1700 AT2450

AMD LANCE (7990, 79C960, PCnet-ISA) AMD 79C961 (PCnet-ISA+)

AMD 79C965 (PCnet-32) AMD 79C970 (PCnet-PCI) AMD 79C974 (PCnet-SCSI)

DE-100, DE-200, DE-220-T DE-520 DE-530 DE-600 DE-620 DFI

DEPCA, DE100/1, DE200/1/2, DE210, DE422

Digital EtherWorks 3 (DE203, DE204, DE205)

DE425 (EISA), DE434, DE435, DE500

DEC 21040, 21041, 21140, Tulip

27245A HP PC Lan+ (27247, 27252A) HP-J2405A

HP-Vectra On Board Ethernet

HP 10/100 VG Any Lan Cards (27248B, J2573, J2577, J2585)

Ether Express Ether Express PRO/10 Ether Express PRO 10/100B

NE1000, NE2000 NE2000-PCI (RealTek-8029) NE1500, NE2100

WD8003, SMC Elite

WD8013, SMC Elite16

SMC Elite Ultra

SMC EtherEZ (8416)

SMC EtherPower PCI (8432)

SMC-9000 / SMC 91c92/4

如果使用以上的网卡,你可以使用 Linux 发行的预先编译好的核心程序,它包含了这些常用设备的所有驱动程序。从长远的观点来看,最好是重新编译自己真正需要的核心程序。

2. 以太网卡的自动检测

在系统启动时,以太网部分的代码会检测以太网卡的存在及其类型。检测的地址和顺序,如表 6.1 所示:

表 6.1

网卡	检测地址
3c501	0x280, 0x300
3c503	0x300, 0x310, 0x330, 0x350, 0x250, 0x280, 0x2a0, 0x2e0
3c505	0x300, 0x280, 0x310
3c507	0x300, 0x320, 0x340, 0x280
3c509	Special ID Port probe
apricot	0x300
at1700	0x300, 0x280, 0x380, 0x320, 0x340, 0x260, 0x2a0, 0x240
atp	0x378, 0x278, 0x3bc
depca	0x300, 0x200
de600	0x378
de620	0x378
eexpress	0x300, 0x270, 0x320, 0x340
hp	0x300, 0x320, 0x340, 0x280, 0x2C0, 0x200, 0x240
hp-plus	0x200, 0x240, 0x280, 0x2C0, 0x300, 0x320, 0x340
lance	0x300, 0x320, 0x340, 0x360
ne	0x300, 0x280, 0x320, 0x340, 0x360
ni52	0x300, 0x280, 0x360, 0x320, 0x340
ni65	0x300, 0x320, 0x340, 0x360
smc-ultra	0x200, 0x220, 0x240, 0x280, 0x300, 0x340, 0x380
wd :	0x300, 0x280, 0x380, 0x240

自动检测程序有两个缺陷。一是它不能正确检测所有的网卡,特别是一些比较廉价的网卡和 WD80x3 的一些网卡。二是核心程序不会自动检测一个以上的网卡。这点是为了使用户可以控制将网卡设置到那个端口。

如果使用一个以上的网卡,或者自动检测网卡失败,都需要明确地把网卡的基地址和名字告诉核心程序。

在 Net-3 中有两种方法可以完成这一步骤。一种方法是在核心程序的源代码的/drivers/net/Space.c 文件中修改或添加信息,然后重新编译内核。只有在非常熟悉网络部分的代码时才推荐使用这种方法。另一种较好的方法是在系统启动过程中将这些信息提供给内核程序。如果是使用 lilo 启动系统,可以在 lilo.conf 文件中通过 append 选项说明,将参数传递给内核。为通知内核以太网设备的信息,将下列参数传递给内核,传递形式为:

append= "ether= irq, base- addr, param1, param2, name"

前三个参数是数字,最后一个是设备名称。所有的数字值都是可选的;如果它们被省略了或者设置为 0,核心将通过检测来确定他们的值,或者使用缺省的值。

第一个参数设置分配给此设备的 IRQ。对缺省的,核心将自动检测此设备的 IRQ 通道。3x503 驱动程序有一个特殊的地方,即它从表 5,9,3,4 中选择一个空闲的 IRQ 通道。

base- addr 参数给出此卡的 I/O 基地址;0 指示核心检测上面列出的地址。

剩余的两个地址不同的驱动程序使用的方式也有不同。对共享存储的卡,如WD80x3,用它来规定共享存储区的起始和结尾地址。其它网卡一般使用 param1 来设置显示调试信息的级别。从值 1 到 7 显示信息的冗余级别逐步提高,而 8 是将其全部关闭,0 是缺省的。3c503 驱动程序使用 param2 来选择使用内部收发器(缺省选择)还是外部收发器(值为 1)。前者使用卡上的 BNC 连接器,后者使用 AUI 端口。

如果具有两个网卡,可以让 Linux 自动检测一个,再通过 lilo 将第二块卡的参数传递给核心程序。然而,必须保证驱动程序不会偶然先检测到第二个网卡,否则另一个网卡就不会被注册。可以通过在 lilo 中加入 reserve 选项,明确告诉内核不要检测第二块网卡占用的 I/O 区域。

例如,为了使 Linux 在 0x300 处安装第二个以太网卡,作为 eth1,需要将下列参数传递给核心:

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

reserve 选项保证其他驱动程序不会访问该 I/O 区域。也可以使用核心参数来避免自动检测 eth0:

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

如果希望关闭所有的自动检测,只要将 base-addr 参数设置成 -1 即可:

ether= 0,-1,eth0.

6.3.4 串行硬件安装

1. 串行设备简介

UNIX 核心提供的访问串行设备的设备文件一般称为 ttys,这是 Teletype™ 的缩写。它是早期 UNIX 终端的主要生产厂家,现在一般用它来表示基于字符的数字终端,在此用它指代核心设备。

Linux 区分三类 ttys:(虚拟)控制台,伪终端(类似于一个双向的管道,由 X11 等程序使用)和串行设备。后者也称为 ttys,因为其允许通过一个串行的连接进行交互式的对话;它或者是一个用线连接的终端,或者是一个通过电话线连接的远程计算机。

ttys 有许多配置参数,可以通过 ioctl 系统调用来设置。许多系统调用都是针对串行设备的,因为他们需要更大的灵活性来处理各种不同方式的连接。

在串行线路的各种参数中最重要的是传输速率和奇偶位。也有一些标志用来在大小写字符和回车及换行间进行转换。tty 驱动程序也可能支持各种“线规则”(line discipline),使得设备驱动程序的行为完全不同。例如 Linux 中的 SLIP 驱动程序是按照一种特殊的线规则实现的。

2. 串行设备的访问

同 UNIX 系统中的其它设备一样,串行端口是通过/etc/目录中的特殊设备文件进行访问的。与串行设备相关的设备文件有两种类型,对每一个端口,两种类型的设备文件中都各有一个与之相对应。通过不同类型的文件进行访问,设备的操作也就不同。

一种类型的设备文件是当端口用来作为拨号进入时使用;它的主设备号是 4,文件名为 ttyS0,ttyS1 等,它的次设备号是 COM 端口号加 63。

假设调制解调器安装在 COM2 端口，则它的次设备号为 65，并且其拨号输出时的主设备号为 5。系统中应该有一个设备 cua1 具有这样的设备号。下面列出的是/dev 目录中的串行设备，其中第 5 和第 6 栏分别为其主设备号和次设备号：

```
$ls -l /dev/cua*
crw-rw-rw-   1      root root          5,      64   Nov 30   19:  31 /dev/cua0
crw-rw-rw-   1      root root          5,      65   Nov 30   22:  01 /dev/cua0
crw-rw-rw-   1      root root          5,      66   Nov 28   11:  32 /dev/cua0
crw-rw-rw-   1      root root          5,      67   Nov 19   19:  31 /dev/cua0
```

如果没有这些设备，就必须自己创建：在超级用户帐号下用下列命令

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

3．串行硬件

Linux 现在支持比较广泛的使用 RS-232 标准的串行卡。RS-232 是现在 PC 串行通信中最通用的标准。它使用几条线路来传输比特信息和同步信息。其他的线路用来进行通知传输设备的存在和握手。

在缺省的配置中，内核检查 4 个标准的串行端口 COM1 到 COM4。它们的次设备号如上面所述被指定为 64 到 67。

如果想正确安装串行端口，应该安装 setserial 命令和其脚本文件 rc.serial。此脚本文件应该在系统启动时从/etc/rc 中调用。它使用 setserial 命令来配置串行设备。一个比较典型的 rc.serial 文件如下所示：

```
#  /etc/rc.serial - serial line configuration script
#
#  Do wild interrupt detection
/sbin/setserial -W /dev/cua*

# configure serial devices
/sbin/setserial /dev/cua0 auto_ irq skip_ test autoconfig
/sbin/setserial /dev/cua1 auto_ irq skip_ test autoconfig
/sbin/setserial /dev/cua2 auto_ irq skip_ test autoconfig
/sbin/setserial /dev/cua3 auto_ irq skip_ test autoconfig

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

各参数的具体含义请参见与 setserial 随同的文档。

如果系统不能够检测到串卡，或者 setserial -bg 命令显示出不正确的设置，就必须通过明确给出正确的参数来进行强行配置。

如果调制解调器支持硬件握手，应该保证它使用了此功能。奇怪的是许多通信软件并不缺省地使用这种功能，因此必须手工的进行设置。这最好是在 rc.serial 文件中进行，使用 stty 命令：

$stty crtscts < /dev/cua1

为检查硬件握手是否已经有效,使用下列命令:

$stty -a < /dev/cua1

此命令将显示此设备的所有标志,标志前面如果带有负号则表明此标志已经被关闭了。

6.4 TCP/IP 网络配置

本节我们将依次介绍配置 TCP/IP 网络所必须的步骤:首先从设置 IP 地址开始,到 TCP/IP 网络接口的配置,以及一些网络安装和检错所常用的一些工具。

在本节中,我们通过一个比 GMU 更为简单的例子进行介绍,这个例子可能与读者实际情况更接近一些。假设一个名为 Virtual 的啤酒厂,为了更有效的管理业务,希望将他们的计算机进行连网,这里他们的计算机碰巧都是安装了 Linux 的 PC 机。

在同一层楼上,假设还有一个葡萄酒厂,其工作关系与啤酒厂非常密切,并且有自己的以太网。希望将两个厂的网络连接起来是非常自然的想法。首先,希望在两个子网之间设置一个网关进行网络之间的数据转发。其次,希望通过 SLIP 连接将他们的网络连接到 Internet 上。

6.4.1 设置主机名

绝大多数网络应用程序的运行依赖于主机名字的正确设置。主机名字的设置一般是在系统启动过程中通过运行 hostname 命令进行的。它的调用形式是:

hostname name

实际中通常使用的主机名字是没有域名的非正式的名字。例如 Virtual 啤酒厂的主机的名字可能是 vale.vbrew.com、vlager.vbrew.com 等。这些都是正规的,全称的域名。它们的本地主机名字可能仅是这些名字的第一部分,如 vale。由于本地主机经常需要查找主机的 IP 地址,因此必须保证解析库能够找到主机的 IP 地址,这就意味着必须将这些名字写入/etc/host 文件,我们将在“6.4.3 编辑 hosts 和 networks 文件”中对此文件进行介绍。

6.4.2 设置 IP 地址

如果只是在一台主机上配置网络软件,可以放心地略过本节,因为在这种情况下你仅需要设置回送(loopback)接口的 IP 地址,也就是 127.0.0.1。

对一个实际的网络如以太网来说,事情就稍微复杂一些。如果是将主机连接到一个现存的网络上,需要向该网络的系统管理员申请此网络上的 IP 地址。如果是自己进行全部网络的安装,就需要按下面描述的方法设置 IP 地址。

一个局域网中的所有主机通常都归为一个逻辑的 IP 网络,因此必须设置 IP 网络地址。如果有多个物理的网络,可以给每个网络都分配不同的网络地址,或者通过子网划分经 IP 网络分成几个子网。

如果网络没有连接到 Internet 上,你可以自由的选择任何有效的网络地址。只需要注

意的是必须从 A、B 或 C 类网络地址中选取，否则网络的运行可能不正常。然而，如果希望在不久的将来能够将网络连接到 Internet 上，就必须现在向 NIC 申请一个正规的 IP 地址。最好的方法是向你的网络服务提供者请求帮助。如果需要申请一个网络地址，应该与 hostmaster@internic.net 联系，领取网络地址申请单。

如果需要运行数个以太网(或者其它网络)，就必须将网络分成几个子网。需要注意的是仅在有不止一个的广播网络(broadcast network)上才需要进行子网的划分，而点到点的连接则不需要。例如一个以太网，一个或多个 SLIP 与外面的连接，这种情况下就不需要对子网进行划分。其原因在阅读后面的高级配置部分后就明白了。

作为例子，假设 Virtual 啤酒厂向 NIC 申请了一个 B 类网络地址 191.72.0.0。为了适应两个网络的情形，决定将主机号部分的前 8 位作为附加的子网位，而剩余的 8 位作为主机号，这样每个子网上允许有 254 台主机。然后将子网号 1 分配给啤酒厂，将子网号 2 分配给葡萄酒厂。这样两个网络的地址分别为 191.72.1.0 和 191.72.2.0。网络屏蔽字为 255.255.255.0。

两个网络之间的网关 vlager，在每个网络上的主机号都为 1，因此其 IP 地址分别为 191.72.1.1 和 191.72.2.1。图 6.2 显示了两个子网和网关。

图 6.2　啤酒厂与葡萄酒厂两个子网示意图

请注意，在这个例子中为了方便起见，使用了一个 B 类网络地址，而事实上一个 C 类网络地址可能更实际一些。实际上对子网的划分并不局限与字节边界，因此一个 C 类的网络也可以划分为多个子网。例如可以取主机部分的 2 位作为网络屏蔽字，每个子网上可以容纳 64 台主机。

6.4.3　编辑 hosts 和 networks 文件

当网络进行子网划分完成以后，需要利用/etc/hosts 文件进行一些简单的域名解析。如果不打算使用 DNS 或者 NIS 进行地址解析，就必须将所有的主机名字都放入 hosts 文件中。

即使你希望在一般的操作时都运行 DNS 或者 NIS，也需要将一些子网中的所有主机的名字都放入/etc/hosts 文件中。例如，希望在系统启动过程中，网络接口还没有运行的情况下进行一些名字解析。这不仅仅是一个方便的问题，还可以使你在脚本文件 rc.inet 中使用符号主机名。这样在 IP 地址改变以后，你只需要拷贝和更新所有主机上的/etc/

hosts 文件并重新启动机器即可。一般来说,经常将所有本地主机的名字和地址都放入 hosts 文件中,如果有的话,网关和 NIS 服务器的地址也应该加进去。

此外,在最初的测试阶段,应该保证让解析器只使用 hosts 文件中包含的信息。DNS 和 NIS 软件可能会携带有一些样本文件,在使用的时候会出现一些奇怪的结果。为了使所有的应用程序完全使用/etc/hosts 文件进行主机 IP 地址的查找,必须编辑/etc/host.conf 文件。方法是将其中任何以关键字 order 开始的行都注释掉,注释的方法是在新行的开始插入一个"# "号,同时在此文件中加入一行。

```
order hosts
```

对于解析库的配置我们将在 6.5 节进一步进行详细介绍。

hosts 文件中每一行为一项,由 IP 地址,主机名称,和主机的一些别名组成,别名是可选的。各域之间用空格或制表键分离,并且地址域必须从第一列开始。所有# 后面的内容都作为注释解释。

主机名称可以是全称,也可以是相对于本地域名的。例如 vale 主机,既可以使用名称 vale.vbrew.com,也可以使用 vale,这样即可以知道正式的名称,也可以知道它本地的名称。

下面是 Virtual 啤酒厂中 hosts 文件的例子。其中包含两个特殊的名字 vlager-if1 和 vlager-if2,他们给出了 vlager 使用的两个接口的地址。

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

就像使用符号名字来代替主机的 IP 地址一样,有时也可以用符号名字来代替网络号。与 hosts 文件伴随的还有一个/etc/networks 文件,它在网络的名字和网络号之间建立映射。在 Virtual 啤酒厂,我们可以设置如下的 networks 文件:

```
# /etc/networks for the Virtual brewery

brew-net            191.72.1.0
wine-net            191.72.2.0
```

6.4.4 为 IP 进行接口配置

按照前面的介绍设置完硬件之后,必须使这些硬件对核心网络软件来说是可见的。有一对命令用来配置网络接口,初始化路由表。这些任务一般都是在系统启动时在 rc.inet1 中做。最基本的工具是 ifconfig 和 route。

ifconfig 用来使核心网络层可以访问接口。它包括设置 IP 地址和其他参数,激活接口。处于激活状态是指核心可以通过它发送或接收 IP 数据报。ifconfig 最简单的调用方法是:

```
ifconfig interface ip-address
```

它将 ip-address 设置给 interface 并激活它。其他的参数值都取缺省的值。例如缺省的网络屏蔽字是由其 IP 地址的网络部分获取的,比如一个 B 类地址的屏蔽字是255.255.0.0。

route 用来在核心路由表中增加或删除路径,它的调用形式是:

```
route [add©del] target
```

这里 add 和 del 参数分别表示增加或删除到 target 的路径。

1. 循环往返接口

第一个需要激活的接口是循环往返接口:

```
# ifconfig lo 127.0.0.1
```

有时候你可能会看到使用 localhost 来代替 IP 地址。此时 ifconfig 会查找 hosts 文件,其中应该有为 127.0.0.1 地址定义主机名的项:

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

如果想查看接口的配置情况,只要用接口名称作为参数运行 ifconfig 命令即可:

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

正如同你所看到的一样,循环返回接口的网络屏蔽字为 255.0.0.0,因为 127.0.0.1 是一个 A 类网络地址。此接口也没有广播地址,因为广播地址对于循环返回接口通常是没有用的。然而如果在你的主机上运行有 rwhod 监控程序,为保证其能够正确运行就必须设置循环返回设备的广播地址。广播地址的设置我们将在“6.4.5 ifconfig”中进行介绍。

现在,已经可以利用最小的网络进行工作。还缺少的一步是在路由表中添加一项,告诉 IP 将此接口作为 127.0.0.1 的目的地。这是通过下列命令完成的:

```
# route add 127.0.0.1
```

这里还可以使用 localhost 来代替 IP 地址。

接下来的步骤检查是否一切都工作正常,例如,使用命令 ping。ping 是一个用来检查一个给定的地址是否可到达有效地址的网络工具程序,并测量发送到该地址一个数据报

往返所用的时间。这个时间常称为往返时间:

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

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

当按照这种方法调用 ping 时,它将不停的发送数据分组,只到被强行终止为止。上面的^ C 表示同时按下 Ctrl 和 C 键,其作用是强行终止正在运行命令。

上面的例子表明发送到 127.0.0.1 的分组传送正常,并且几乎立即就返回 ping 的调用。这表明已经正确安装了第一个网络接口。

如果 ping 的输出与上面的形式不同,这表明出了什么问题,就需要进行检查错误,看他们是否指示一些文件没有正确安装。首先检查 ifconfig 和 route 运行程序和当前使用的核心版本是否兼容,并且最重要的是核心在编译时是否选择了网络支持。可以通过查看/proc/net 目录是否存在来检查核心是否提供网络支持,如果存在,则表明核心提供对网络的支持。如果出现“Network unreachable”的错误信息,很可能是你的 route 命令出现了问题。要保证所用的 IP 地址与 ifconfig 中使用的一致。

上面的步骤对在单机上运行网络程序已经足够了。在将上面的命令加入 rc.inet1 后要确保它在/etc/rc 中获得执行。可以重新启动机器,检验各种应用程序。例如,使用“telnet localhost”命令将建立一个到自己机器的 telnet 连接,它给你显示一个登录的进程。

循环返回接口不仅在网络教科书中作为例子以及网络程序开发的测试过程中有用,实际上一些网络程序在通常的操作中也使用到了它。因此,无论机器是否连网,这个接口一般都需要设置。

2. 以太网接口

配置以太网接口与配置循环返回接口十分相似,区别在于如果进行子网划分,要多加几个参数而已。

例如,我们将一个 B 类的网络分成几个 C 类的网络,要想使接口认识到这一点,应该用下列的形式使用 ifconfig 命令:

```
# ifconfig eth0 191.72.1.2 netmask 255.255.255.0
```

此命令将 IP 地址 191.72.1.2 分配给 eth0 接口。如果省略了网络屏蔽字这一项,ifconfig 就会从 IP 网络类中推导出网络屏蔽字,这里是 255.255.0.0。现在进行以下检查:

```
$ ifconfig eth0
eth0        Link encap 10Mps Ethernet Hwaddr 00: 00: A0: 80: 4B: 42
```

```
          inet addr 191.72.1.2 Bcast 191.72.1.255 Mask 255.255.255.0
          UP BROADCAST RUNNING MTU 1500 Metric 1
          RX packets 0 errors 0 dropped 0 overrun 0
          TX packets 0 errors 0 dropped 0 overrun 0
```

可以看到 ifconfig 根据通常的值自动设置了广播地址(上面的 Bcast 域),即将主机域都设置为 1。消息的最大传输单位(核心为此接口生成的 Ethernet 帧的最大长度)也设置成了 1 500 字节。所有的这些值都可以利用特殊的选项进行重新设置,将在“6.4.5 ifconfig”中进一步进行详细的介绍。

与循环返回接口类似,必须在路由表中增加一项以通知核心可以通过接口 eth0 访问到此网络:

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

乍一看有点模糊,因为此命令并没有明确告诉 route 如何将数据转到那个接口。实际原理非常简单:核心程序检查所有已经配置的接口,将数据的目的地址(这里是 191.72.1.0)与接口地址的网络部分(即接口地址和网络屏蔽字的与)相比较,从而找到匹配的网络接口。这里匹配的唯一接口是 eth0。

我们现在来看-net 选项是什么含义。因为 route 既可以处理到网络的路由,又可以处理到单个主机的路由(如上面的 localhost)。当以点分形式(dotted quad notation)给出地址时,它将通过查看主机部分的数据位猜测此地址代表一个主机还是一个网络。如果地址的主机部分全为 0,route 认为其代表一个网络,否则认为其是一个主机的地址。因此,由于 route 不知道我们进行了子网划分,就会认为此地址表示的是一个主机。所以要明确的通过-net 选项告诉它此地址代表的是一个网络。

当然上面的 route 命令的使用方法比较麻烦,很容易出现拼写错误。一个更方便的方法是使用/etc/networks 中定义的网络名字。这样使得命令更可读,同时还可以省略-net 标识符,因为 route 现在知道 191.72.1.0 代表一个网络。

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

现在已完成了基本的配置步骤,需要验证 Ethernet 接口能够正确运行。方法是选择 Ethernet 上的一个主机,如 vlager,运行下列命令:

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

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

如果没有看到类似的输出结果,那么就是什么地方出了毛病。如果遇到了不寻常的分

组(packet)丢失率,表明硬件存在问题,如坏的或丢失的终端等。如果没有接收到任何分组,应该使用 netstat 对接口配置进行检查。ifconfig 显示的分组统计数字会告诉你分组是否从此接口送出。如果你能够访问远程主机,也应该登录到远程主机上对接口统计数据进行同样的检查。这样可以精确地确定分组丢失在什么地方。另外还可以使用 route 显示路由信息,以检查主机双方的路径表中的项是否正确。当不带任何参数运行 route 命令时,显示的是核心程序的全部路由表。-n 选项迫使其以点分形式(dotted quad)而不是主机名字来显示地址:

```
# route -n
Kernel routing table
Destination     Gateway         Genmask         Flags   Metric  Ref    Use Iface
127.0.0.1       *               255.255.255.255 UH      1       0      112 lo
191.72.1.0      *               255.255.255.0   U       1       0       10 eth0
```

每个域的详细含义将在“6.4.6 netsat”中进行介绍。Flag 栏包含每个接口的设置标识。U 对每个激活的接口都是设置的,H 标明目的地址是一个主机。如果对于一个希望是网络路径的接口设置了 H 标志,就必须在 route 命令中使用-net 选项。为了知道某个路径是否被使用过,可以检查两次运行 ping 后此路径的 Use 一栏的数值是否增加了 2。

3. 通过网关寻径

前面仅涉及到了在一个以太网上设置主机。然而经常会遇到网络通过网关与其它网络连接的情形。这些网关可以简单的连接到两个或更多的以太网,也可以连接到外面的世界,如 Internet。为了能够使用网关的服务,必须为网络层提供附加的路由信息。

例如,Virtual 啤酒厂和 Virtual 葡萄酒厂的以太网通过一台为名为 vlager 的主机作为网关进行连接。假设 vlager 已经配置好了,只需要在 vstout 的路由表中增加一项,告诉它可以通过 vlager 访问葡萄酒厂网络的所有主机。其设置如下:

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

关键词 gw 表示下一个参数代表一个网关。

当然,如果需要与葡萄酒厂网络上的主机进行对话,该主机上也必须有到啤酒厂的相应的路由项,否则只能够将数据从 vstout 发送到 vbardolino,而后者返回的任何相应信息都将丢失。

这个例子仅描述了连接两个以太网的网关。现在假设 vlager 还有一个 Internet 连接(假设是一个 SLIP 连接),那么我们将任何目的网络不是啤酒厂的数据报都传递给 vlager。这可以通过将其设置成为 vstout 的缺省网关:

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

网络名字 default 是 0.0.0.0 的简写,指示缺省的路径。并不需要将这个名字加入到/etc/networks文件中,因为它是在 route 中构造的。

当通过 ping 发现相隔一个或多个网关的主机间的分组丢失率比较高时,表明网络比较拥塞。分组丢失在这种情况下通常是由于网络负载暂时过重,而不一定是技术缺陷。

4. 网关配置

配置一台机器使其能够在两个以太网之间进行数据交换是相当简单的。假设我们的

主机叫 tirc, 它上面插有两个网卡, 每个网卡与一个以太网相连。所需要作的工作就是对每个端口进行分别配置, 分别给它设置 IP 地址即可。

在 hosts 文件中加入两个接口的信息是十分有用的, 如下所示:

```
166. 111. 16. 16        tirc
166. 111. 16. 16        tirc-if1
166. 111. 166. 129      tirc-if2
```

配置这两个接口的命令序列如下:

```
# ifconfig eth0 tirc-if1
# ifconfig eth1 tirc-if2
# route add -net 166. 111. 16. 0
# route add -net 166. 111. 166. 129
```

6. 4. 5 ifconfig

ifconfig 有许多调用参数, 它的一般运行格式是:

ifconfig interface [[-net©¦hosts] address [parameters]]

interface 是接口的名称, address 是设置给此接口的 IP 地址。IP 地址或者以点分四元组形式(dotted quad notation)给出, 或者以名字的形式给出。当以名字的形式给出时, ifconfig 将查找/etc/hosts 文件和/etc/networks 找到对应的 IP 地址。-net 和-host 选项强制 ifconfig 将该地址作为网络地址或主机地址。

如果仅用接口名称作为参数调用 ifconfig, 它将显示此接口的配置信息。如果不带任何参数运行, 它将显示已经配置的所有接口, 选项-a 同时显示没有激活的接口。例如, 对以太网接口 eth0 的运行显示如下:

```
# ifconfig eth0
eth     Link encap 10Mbps Ethernet  Hwaddr 00: 00: C0: 90: B3: 42
        inet addr 191. 72. 1. 2  Bcast 191. 72. 1. 255  Mask 255. 255. 255. 0
        UP BROADCAST RUNNING  MTU 1500  Metric 0
        RX packets 3136 errors 217  dropped 7  overrun 26
        TX packets 1752 errors 25  dropped 0  overrun 0
```

MTU 和 Metric 域分别表示此接口的 MTU(最大传输单元)和度量值。一些操作系统使用度量值来计算此接口的路由开销。Linux 目前没有使用此值, 定义它是为了考虑网络软件的兼容。

RX 和 TX 行显示的是已经接收和传送了多少没有出错的分组(packet); 发生了多少次错误; 丢失了多少分组; 由于内存不足丢弃了多少分组; 由于过载丢失了多少分组。接收过载通常是因为分组到达的速度超过了核心处理中断的速度。ifconfig 显示的标志符的值与命令行的给出的接口名字没有多大关系。

下面列出了 ifconfig 可以识别的与括号中给出的标志名相应的参数。已经打开的选项可以通过简单的在其前面加一“-”取消。

up 它表明此接口对 IP 层来说是可见的。当在命令行中给出了 address 后此选项隐

含处于设置状态。它也可以用来使用 down 选项而暂时不可见的接口重新可见。

此选项相对与标志 UP RUNNING。

down　它表明此接口对 IP 层来说是不可见。这样有效的阻止了通过此接口的任何 IP 通信。需要注意的是它不自动删除路径表中使用此接口的项。如果永久性的关闭了此接口,就需要删除这些路径项,如果可能的话可以提供一些替换路径。

netmask mask　此选项设置接口使用的子网屏蔽字。它或者以 0x 开始的 32 位十六进制形式给出,或者以点分四元组十进制数字给出。

pointtopoint address　此选项适用于仅涉及两台主机的点到点连接。在配置如 SLIP, PPP 接口时要用到此选项。

broadcast address　广播地址通常是将地址中的主机部分的比特位都设置成 1 构成的。有的 IP 使用不同的策略,此选项就是为了适合特殊的环境。当广播地址设置后,ifconfig 将显示 BROADCASTB 标志符。

metric number　此选项可以用来为此接口生成的路由表项设置度量值。RIP(路由信息协议, Routing Information Protocol)使用此度量值来为网络建立路由表。ifconfig 使用的缺省的度量值为 0。如果你没有运行 RIP 监控程序,就根本不需要此选项;如果运行了 RIP 监控程序,也很少需要修改此度量值。

mtu bytes　设置最大传输单位 MTU(Maximum Transmission Unit),即此接口一次能处理的数据包的最大字节数。对于以太网来说,MTU 的缺省值是 1500; SLIP 接口为 296。

arp　这个选项仅适用于广播网络,如以太网。该选项设置以后可以使用 ARP (Address Resolution Protocol)来探测与网络相连的主机的物理地址。对于广播网络来说,此项缺省处于设置状态。

-arp　禁止在此接口上使用 ARP。

promisc　使接口处于混杂模式(promiscuous mode)。在一个广播网络上,将使此接口接收所有的分组,而不管他们的目的地址是本机还是其他主机。这就使得可以通过分组过滤器,或者称为以太网侦听来分析网络的通信情况。(此选项与标志符 PROMISC 相对应。)

-promisc　关闭混杂模式。

allmulti　多发地址是指在一组主机中进行的广播,此主机组不需要一定属于一个子网。现在的核心还不支持这种地址。此选项同 ALLMULTI 标志相对应。

-allmuti　关闭多发(multicast)地址。

6.4.6　netstat

netstat 是一个检查网络配置和活动的非常有用的工具。下面分别对它的各个功能加以介绍。

1. 显示路由表

当用-r 作为参数运行 netstat 时,它显示当前的路由表。在 vstout 上的显示内容如下所示:

```
# netstat -nr
Kernel routing table
Destination     Gateway         Genmask         Flags   Metric Ref    Use Iface
127.0.0.1       *               255.255.255.255 UH        1      0       50 lo
191.72.1.0      *               255.255.255.0   U         1      0      478 eth0
191.72.2.0      191.72.1.1      255.255.255.0   UGN       1      0      250 eth0
```

-n 选项使得 netstat 按点分四元组的形式显示 IP 地址，而不是主机或网络的符号名称。

netstat 输出的第二栏显示的是路由项指向的网关。如果没有使用网关，那么就显示一个星号。第三栏显示的是路径“生成器”(generality)。当为一个 IP 地址寻找合适的路径时，核心查找路由表中的所有项，在与目的路径比较之前将地址与 Genmask 进行逐位的与操作。

第四栏显示描述路径的各种标志符：

G　此路径使用网关。

U　此接口处于“up”状态，即对 IP 层可见。

H　通过此路径只能访问到一个主机，例如循环返回接口的入口 127.0.0.1。

D　如果此表项是由 ICMP 重定向消息生成的则设置此标志。

M　如果此表项被 ICMP 重定向消息修改过则设置此标志。

Ref 一栏显示访问此路径的次数，也就是多少其它的路径依赖于此路径。最后两栏分别表示此路径使用的时间和分组要发送的接口。

2. 显示接口统计数字

当以-i 为参数运行 netstat 程序时，显示当前配置的网络接口的统计信息。如果同时附加了-a 选项，将显示核心中所有接口的信息，而不仅仅是当前配置的接口。其输出的形式如下所示：

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

MTU 和 Met 域分别显示了接口当前的 MTU 和度量值。RX 和 TX 栏分别显示正确接收和传输了多少分组(RX-OK/TX-OK)，RX-ERRR/TX-ERR 显示有多少损害的分组，RX-DRP/TX-DRP 显示丢弃了多少分组，RX-OVR/TX-OVR 显示由于过载丢失了多少分组。

最后一栏显示的是为此接口设置的标识符。它以一个字母的形式给出此接口的设置情况，字母与 ifconfig 命令显示的接口配置的长标识符名称的对应关系如下：

B　设置了广播地址。

L　此接口是回送接口。

M　　所有的包都已收到(混杂模式)。

N　　忽略尾部信息。

O　　此接口的 ARP 被关闭。

P　　点到点连接。

R　　接口正处于运行状态。

U　　接口处于“up”状态。

3. 显示网络连接信息

netstat 支持一些选项来显示主动的或被动的套接字。选项-t, -u, -w 和-x 分别显示活动的 TCP, UDP, RAW 和 UNIX 套接字连接。如果另外还使用了-a 选项, 等待连接(即侦听)的套接字也会显示出来。这样就给你一个当前系统中运行的服务程序的表列。

在 vlager 上运行 netstat -ta 的结果如下:

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

显示表明大多数服务器正简单的等待外来的连接请求。而第四行显示有一个从 vstout 来的 SMTP 连接, 第六行显示有一个到 vbardolino 的 telnet 连接。

如果只使用-a 选项, 将显示所有类型的套接字。

6.4.7　检查 ARP 表

在一些情况下, 查看甚至修改核心 ARP 表的内容是非常有用的, 例如, 当你怀疑有重复 Internet 地址而出现间歇性的网络问题, arp 工具就是处理这类情况的。它的命令行选项是:

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

所有的 hostname 实参即可以是主机的符号名字, 也可以是以点分十进制形式给出

的 IP 地址。

第一个调用显示指定的 IP 地址或主机的 ARP 项,如果没有给出 hostname,这表示是所有主机。例如在 vlager 上运行 arp 可能产生的结果是:

```
# arp -a
IP address      HW type              HW address
191.72.1.3      10Mbps Ethernet      00:00:C0:5A:42:C1
191.72.1.2      10Mbps Ethernet      00:00:C0:90:B3:42
191.72.1.4      10Mbps Ethernet      00:00:C0:04:69:AA
```

它显示了 vlager, vstout 和 vale 的以太网的地址。

使用-t 选项可以只显示指定的硬件类型。这些硬件类型可能是 ether, ax25 或者 pronet,分别代表 10Mbps 以太网,AMPR AX.25 和 IEEE802.5 令牌环网设备。

-s 选项用来将 hostname 的以太网地址永久性的加入 ARP 表。hwaddr 参数给出硬件地址,缺省为以太网地址。也可以利用-t 选项为其它类型的硬件设置硬件地址。

某些情况下可能需要手工将 IP 地址加入到 ARP 表中,例如,由于某些原因对远程主机的 ARP 查询失败的情形。

以-d 为参数调用 arp 将删除与给定主机相关的所有 ARP 项。可以用来迫使接口重新获取查询的 IP 地址相应的以太网地址。这在一个没有正确配置的系统广播了错误的 ARP 信息时是很有用的。

-s 选项还可以用来实现代理 ARP(proxy ARP)。这是一项特殊的技术,假设有一个主机 gate 是另一个主机 fnord 的网关,这两个主机(gate 和 fnord)的地址都是指向同一个主机 gate。其实现是为 fnord 生成一个 ARP 项,将它指向自己的以太网接口。现在当其它主机发出对 fnord 的 ARP 查询时,gate 就返回一个包含自己以太网地址的应答。查询的主机将把所有的数据报都发送给 gate,而由 gate 将它们转发给 fnord。

这种曲折的连接方式有时是需要的。例如,当你从实现了部分 TCP 而不能很好支持路由的 DOS 机器访问 fnord 时,通过使用代理 ARP,对于 DOS 机器看来就好像 fnord 位于本地子网上,这样它就不需要知道如何通过网关进行路由选择。

代理 ARP 的另一个非常有应用的是你的主机中的一个通过一个拨号,连接暂时的作为其它一些主机的网关。需要注意的是提供代理 ARP 的主机与网关要处于同一子网。比如 vstout 可以作为啤酒厂子网上任何主机的代理 ARP,而不能作为葡萄酒厂子网(191.72.1.0)任何主机的代理 ARP(191.72.2.0)。

下面是为 fnord 提供代理 ARP 的正确命令,其中给出的地址是 gate 的以太网的地址:

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

代理 ARP 可以通过下列命令取消:

```
# arp -d fnord
```

6.4.8 网络配置文件

前面详细的介绍了如何手工对网络进行配置,一个很自然的想法是希望能够将这些

配置命令存储下来,在系统启动的过程中自动进行,而不再需要手工逐个命令进行。在 Linux 中是通过/etc/rc.d/rc.inet1 和/etc/rc.d/rc.inet2 两个文件实现的。这两个文件类似于 DOS 系统中的 autoexec.bat 文件,在系统启动过程中由初始化程序执行其中的命令。因此可以将网络配置命令都写入这两个文件中,在系统启动过程中利用这些命令自动的完成网络的配置。

rc.inet1 和 rc.inet2 两个文件的侧重点不同。rc.inet1 文件中主要是通过 ifconfig 和 route 命令进行基本的 TCP/IP 接口的配置;而 rc.inet2 主要是用来启动一些网络监控进程如 inetd,portmapper 等。我们将在后面对这些监控进程进行介绍。下面首先看以下 rc.inet1 和 rc.inet2 的两个简单的例子,以便了解这两个文件的使用。

下面是 rc.inet1 的一个简单的例子:

```
# ! /bin/sh
#
# rc.inet1 --- Configures network devices.
#
# Attach the loopback device.
/sbin/ifconfig lo 127.0.0.2
# Add a route to point to the loopback device.
/sbin/route add 127.0.0.1
# End loopback
#
# Attach an ethernet device
#
# configure the IP address, netmask and broadcast address.
/sbin/ifconfig eth0 IPA.IPA.IPA.IPA
/sbin/ifconfig eth0 netmask NMK.NMK.NMK.NMK
/sbin/ifconfig eth0 broadcast BCA.BCA.BCA.BCA
# add a network route to point to it:
/sbin/route add -net NWA.NWA.NWA.NWA device eth0
#
# End ethernet
#
```

此 rc.inet1 文件主要由两部分组成,第一部分是对回送接口进行配置,第二部分是对一个以太网接口进行配置。在第二部分中:IPA.IPA.IPA.IPA 是指主机的 IP 地址;NMK.NMK.NMK.NMK 是指网络屏蔽字;BCA.BCA.BCA.BCA 为广播地址;NWA.NWA.NWA.NWA 为网络地址。

在实际配置过程中可以根据网络的实际情况将其替换为自己所需的地址。

rc.inet2 文件的建立比较简单,主要用来启动网络监控进程。下面是此文件的一个简单的例子:

```
# ! /bin/sh
#
# rc.inet2 This shell script boots up the entire INET system.
#          Note, that when this script is used to also fire
#          up any important remote NFS disks (like the /usr
#          distribution), care must be taken to actually
#          have all the needed binaries online - now- ...
#
# Author: Fred N. van Kempen, < waltje@uwalt.nl.mugnet.org>
#

# Constants.
NET= "/usr/sbin"
IN- SERV= "lpd"
LPSPOOL= "/var/spool/lpd"

# At this point, we are ready to talk to The World...
echo "Mounting remote file systems..."
/sbin/mount -a -t nfs        # This may be our /usr runtime!!!

echo -n "Starting daemons:"

# Start the SYSLOGD/Klogd daemons. These must come first.
if [ -f ${NET}/syslogd ]; then
    echo -n " syslogd"
    ${NET}/syslogd
    echo -n " klogd"
${NET}/klogd
fi

# Start the SUN RPC Portmapper.
if [ -f ${NET}/rpc.portmap ]; then
echo -n " portmap"
${NET}/rpc.portmap
fi

# Start the INET SuperServer
if [ -f ${NET}/inetd ]; then
```

```
  echo -n " inetd"
${NET}/inetd
else
  echo "no INETD found. INET cancelled!"
  exit 1
fi

# Start the NAMED/BIND name server.
if [ -f ${NET}/named ]; then
  echo -n " named"
  ${NET}/named
fi

# # Start the ROUTEd server.
# if [ -f ${NET}/routed ]; then
#     echo -n " routed"
#     ${NET}/routed -g -s
# fi

# # Start the various SUN RPC servers.
if [-f ${NET}/rpc.portmap ]; then
   # Start the NFS server daemons.
   if [ -f ${NET}/rpc.mountd ]; then
     echo -n " mountd"
     ${NET}/rpc.mountd
   fi
   if [ -f ${NET}/rpc.nfsd ]; then
echo -n " nfsd"
     ${NET}/rpc.nfsd
   fi
fi # Done starting various SUN RPC servers.

# The 'echo 'below will put a carriage return at the end
# of the list of started servers.
echo
# Done!
```

6.5 名字服务和解析器配置

前面已经讨论到 TCP/IP 网络可以依赖不同的机制将主机名字转换成地址,最简单的方法是将主机名字及相应的 IP 地址都放在文件/etc/hosts 中。这种方法仅适用于小型的局域网,由一个系统管理员进行管理,并且与外面的世界没有 IP 通信。hosts 文件的格式我们已经在前面进行了介绍。

作为另一种解决办法,可以使用 BIND(Berkeley Internet Name Domain Service)进行主机名字到 IP 地址的解析。配置 BIND 是一件很琐碎的活,但是一旦配置完成以后,要改变网络拓扑结构就非常容易了。同其它 UNIX 系统一样在 Linux 中也是通过一个叫 named 的程序来提供名字服务。named 在初始启动的时候,它将一些主文件(master files)装载到其 cache 中,然后等待远程或本地用户进程的查询。BIND 的设置有多种方式,并不需要在每个主机上都运行域名服务器。

6.5.1 解析库

当谈到解析器(resolver)的时候,我们并不是特指任何程序,而是指解析库(reslover library),它是标准 C 库中的一些函数集。其中最主要的例程是 gethostbyname(2)和 gethostsbyaddr(2),它们的功能分别为从指定的主机名字查找其 IP 地址和从指定的 IP 地址查找其主机名字。根据不同的配置,它们进行查找的方式可以不同。它们可以简单的查找 hosts 文件中的信息,或者查询一些名字服务器,或者使用 NIS 中的 hosts 数据库来完成名字和地址间的转换。其他应用,如 smail,可能为此而包含了不同的驱动程序,需要特殊的处理。

1. hosts.conf 文件

控制解析器配置的最主要的文件是 host.conf。它位于/etc 目录中,规定解析器使用那些服务,按什么顺序进行。

host.conf 中的选项必须分别处于不同的行中。各域之间可以通过空格键或制表键分开。以"# "打头的行为注释行。

它有下列一些选项:

order

它规定解析服务的顺序。有效的可选值有 bind,表示向名字服务器进行查询;hosts,表示在/etc/host 文件中查找;nis,表示在 NIS 中查找。可以同时设置它们中的一 个或多个。其在各行中出现的顺序决定了使用哪种方式进行查找的顺序。

multi

具有 on 和 off 两个可选值。它决定/etc/hosts 中的主机是否可以具有多个 IP 地址,具有多个 IP 地址的主机通常被称为是多宿主(multihomed)的。这个标志对 DNS 和 NIS 查询没有任何影响。

nospoof

DNS 允许通过使用 in-addr.arpa 域来查找一个 IP 地址的主机名称。名字服务

器试图提供一个错误的主机名称为“欺骗”(spoofing)。为预防这种情况,可以配置解析器对 IP 地址是否真的和返回的主机名相匹配进行检查。如果不匹配,就拒绝接受此主机名,并返回错误信息。通过设置 nospoof on 可以打开此行为。

alert

此选项以 on 或者 off 作为可选值,如果它打开了,当任何试图进行“欺骗”的行为出现时,解析器都会在系统日志中登记一条信息。

trim

此选项用域名作为参数,在进行查找之前将它从主机名中删除。这对于 hosts 中不带本地域名的主机名字的项来说是有用的。对具有本地域名的主机名字进行查找时,将会被首先将其中的域名删除,这样就使得/etc/hosts 的查找取得成功。

vlager 中的一个样本文件如下:

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

2. 解析器环境变量

hosts.conf 文件中的设置可以被一些环境变量的值所覆盖,它们是:

RESOLV-HOST-CONF

指定一个替换/etc/host.conf 的文件。

RESOLV-SERV-ORDER

覆盖 host.conf 中的 order 选项。服务器分别是 hosts,bind,和 nis,用空格、逗号、句号或分号隔开。

RESOLV-SPOOF-CHECK

定义预防欺骗的措施。其取值为 off 时这种检查被完全取消。当取值为 warn 和 warn off 时,都将进行欺骗行为的检查,分别记录或不记录信息。值“* ”打开欺骗性检查,但保持 host.conf 中定义的记录机制。

RESOLV-OVERRIDE-TRIM-DOMAINS

此环境变量定义一些裁剪域,它们覆盖 host.conf 中给出的域。

RESOLV-ADD-TRIM-DOMAINS

此环境变量定义一些裁剪域,这些裁剪域添加到 host.conf 文件中给出的那些域。

3. 配置名字服务器的查找 - resolv.conf

当配置解析库使用 BIND 名字服务进行主机的查找时,必须告诉它使用哪个名字服

务器。这是由一个单独的文件 resolv. conf 来指定的。如果文件不存在,解析器就假设名字服务器位于本地主机,如果存在,则由文件中指出。

如果是在本地主机上运行名字服务器,必须对其进行分别设置,下面将会对此进行解释。如果你是在一个局域网上,并且有机会使用一个现成的名字服务器,采用这种方法是比较好的。

resolv. conf 中最重要的选项是 nameserver,它给出所使用的名字服务器的 IP 地址。如果用 nameserver 选项给出了多个名字服务器,就会按顺序对它们进行尝试,直到查找成功为止。因此应该将最可靠的名字服务器放在最前面。目前最多支持三个名字服务器。

如果没有给出 nameserver 选项,解析器试图与本地主机的名字服务器联系。

另外两个选项 domain 和 search 与 BIND 第一次查询失败后附加到主机名字上的缺省的域相关。search 选项给出一些可能的域名,各域名间用空格或制表键隔开。

如果没有给出 search 选项,就通过本地的域名构造一列域名,方法是使用本身的域名,加上从父域一直到根的域。本地域名通过 domain 声明给出,如果没有给出,解析器通过系统调用 getdomainname(2) 获得。

如果你感到有些迷惑,请看 Virtual 啤酒厂的 resolv. conf 文件的一个样本:

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

当解析名字 vale 时,解析器将首先查找 vale,失败后查找 vale. vbrew. com 和 vale. com。

4. 解析器的强壮性(Robustness)

如果是在一个大型的网络中运行一个 LAN(局域网),可能的话,毫无疑问应该使用中心名字服务器。这种作法的好处是它可以建立非常丰富的 cache,因为所有的查询都被转向它们。然而这种方法也有一个缺陷:如果校园主干网的电缆由于某种原因突然断开了,这样本系 LAN 上的服务器也不能访问了,因为解析器不能和任何名字服务器取得联系。这样就不再能通过 X 终端进行登录,不能进行打印等操作。

尽管校园主干网不太容易损坏,但还是应该注意这种情况。

一种方法是设置本地的名字服务器对本域的主机名字进行解析,而将对其它主机名字的解析都转向主服务器。当然,这种方法仅在你运行有自己的域时才能使用。

另外,还可以在/etc/hosts 文件中保存你的域或 LAN 中的主机表列的一个备份。同时在/etc/hsot. conf 中应该包含“order bind hosts”,使得在中心名字服务器关闭时,解析器使用 hosts 文件进行域名解析。

6.5.2 运行 named

决大多数 UNIX 机器上提供域名服务的程序叫 named。它是一个服务器程序,最初

由 BSD 开发,用来向客户或其他名字服务器提供域名服务。当前 Linux 中使用的版本是 BIND-4.8.3。新的版本 BIND-4.9.3,现在还处于测试阶段,很快就能得到。

本部分需要对 DNS 系统的工作原理有一定的了解,如果你对本部分将介绍的内容有些迷惑,应该重新阅读 5.7 节的内容,那里对 DNS 的基本知识进行了介绍。

named 通常是在系统启动时开始运行,直到关机为止。它从配置文件/etc/named.boot 中获取信息,以及各种包含域名到地址映射的数据文件,后者称为"区文件"(zone file)。这些文件的格式和语义将会在后面进行介绍。

要运行 named,只需简单的运行:

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

named 就会启动,读取 named.boot 文件和此文件中指定的区文件。它将自己的进程号以 ASCII 的形式写入/var/run/named.pid,从主要服务器取下所有区文件,如果需要,开始在端口 53 上对 DNS 查询进行监听。

1. named.boot 文件

named.boot 文件一般来说非常小,它只包含一些指向包含区信息的主文件,以及指向其他名字服务器的信息。以";"开始的内容直到下一行开始为注释部分。在我们对 named.boot 文件的格式进行详细讨论之前,首先来看以下 vlager 中此文件的一个例子:

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

此文件中的 cache 和 primary 命令将信息装载到 named 中。信息是由第二个参数所指定的主要文件中获取的。它们主要包含文本形式的 DNS 资源记录,我们下面将会看到。

在这个例子中,如文件末尾的 primary 声明所示,我们把 named 配置成三个域的主要名字服务器。比如这些行中的第一行,告诉 named 作为 vbrew.com 的主要名字服务器,从文件 named.hosts 文件中获取区数据。关键字 directory 告诉 named 所有的区文件都位于/var/named 目录。

cache 项非常特殊,应该在所有运行名字服务器的机器上出现。它的功能是两方面的:首先告诉 named 使用 cache,并从指定的文件(这里是 named.ca)中读取有关根名字服务器暗示,我们将在下面对此作出解释。

下面列出了在 name.boot 中可以使用的一些重要选项:

directory　它指出区文件所处的目录。文件名应该相对于此目录给出。可以重复使用 directory 来指定多个目录。根据 Linux 文件系统标准,此目录应该是/var/named。

primary　它带一个域名和一个文件名作为参数,声明此域的本地服务器的权限。作为主要服务器,named 从指定的主要文件中获取区信息。一般来说在每个初始化文件中最少有一个 primary 项,及网络 127. 0. 0. 0(回送网络)的逆向映射。

secondary　它带一个域名,一个地址列表和一个文件名作为参数。它为本地服务器声明一个指定域的二级主服务器。

一个二级服务器也保存此域的所有权威的信息,但是它不是由文件中收集的,而是试图从主要服务器获取。在地址表中最少要有一个主服务器的 IP 地址提供给 named。本地服务器将依次跟它们取得联系,直到成功的获得了区数据库,然后把存入到第三个参数指定的备份文件中。如果没有主要的名字服务器作出反应,就从备份文件中获取区数据。

然后 named 周期性地对区数据进行更新。我们在下面介绍 SOA 资源记录类型时一同进行介绍。

cache　它带一个域和一个文件名作为参数。该文件包含根服务器信息的暗示,为一个指向根名字服务器的记录表。它仅识别 A 和 NS 记录类型。域名参数一般是根域名“.”。

这个信息对 named 来说是非常重要的。如果在启动文件中没有 cache 声明,named 将不会使用本地的 cache。这样当下一次查询不在本地网络的主机时会严重降低性能,增加网络的负载。并且 named 还不能访问到任何根名字服务器,这样它就解析不到其授权以外的任何地址。这种规则的一个例外是使用转发服务器(下面的 forwarders 选项)。

forwarders　此声明的参数是一个地址表。此表中给出的是一些名字服务器的 IP 地址,当 named 从它的本地 cache 没有解析到一个查询时,它就向这些名字服务器依次进行查询,直到成功为止。

slave　声明名字服务器为一个从名字服务器。也就是说它本身不进行任何查询,而是将查询转发到 forwarders 声明中指定的服务器。

另外还有两个选项 sortlist 和 domain,我们在这里没有介绍;还有两个指令 $INCLUDE 和 $ORIGIN 可能在区数据库文件中出现。由于它们都很少被使用到,所以在此都不进行介绍。

2. DNS 数据库文件

named 包含的主文件,如 named. hosts,通常有一个域与之相关联,称为起始点(origin)。这个域是由 cache 和 primary 命令说明的。在一个主文件内部,可以以相对于此域的形式给出域和主机名称。在配置文件中,如果一个名字以一个“.”结尾,则认为其是绝对域名,否则认为其是相对于起始点的。起始点本身可以通过“@”访问。

一个主文件中的所有数据都被分解成资源记录 RR(Resource Recorder),它们组成 DNS 中的最小信息单位。每个资源记录都有一个类型。例如 A 记录是将主机名称映射到它们的 IP 地址,CNAME 记录是为一个主机的正式名称起一些别名。作为一个例子,可以参看下一节“编写主要文件”中 Virtual 啤酒厂的 named. hosts 主文件。

主文件中的资源记录表示方法具有共同的格式:

[domain] [ttl] [class] type rdata

各域之间用空格键或制表键分开。只要在新行开始之前和最后一行之后用括号括起来,一个记录也可以延伸到几行。在分号和新一行开始之间的内容为注释。

[domain] 此记录适用的域。如果没有给出域,RR 就假设它适用的域与前一个 RR 相同。

[ttl] 为了迫使解析器在一段时间之后放弃信息,每一个资源记录都有一个“生存时间”ttl(time to live)。ttl 域以秒为单位给出了信息从服务器取得后的有效时间。它是一个十进制数,但最多不能超过 8 位数。

[class] 这是地址类型,如 IP 地址为 IN,Hesiod 类的对象为 HS。对于 TCP/IP 网络,必须使用 IN。

如果没有给出 class 域,就假设其与前一个 RR 的类型相同。

type 它说明 RR 的类型。最常见的类型是 A,SOA,PTR 和 NS。下面将对 RR 的这些类型进行介绍。

rdata 保存与 RR 相关的数据。此域的格式依赖于 RR 的类型;

下面列出了 DNS 主文件中部分 RR 的类型。还有一些 RR 的类型由于是试验性的,一般来说很少使用,因此不作介绍。

SOA 描述一个权威的区。它表明在 SOA RR 之后的记录包含的是此区的管理信息。每个包含 PRIMARY 声明的主文件都必须包含此区的一个 SOA 记录。其资源分组含下列域:

origin 这是此域的主要名字服务器的正式主机名字。它通常以绝对名字的形式给出。

contact 这是维护此域的人的 email 地址,其中“@”由“.”替代。例如 Virtual 啤酒厂的负责人是 janet,那么此域就是 janet.vbrew.com。

serial 这是由一个十进制数表示的区文件的版本号。无论什么时候区文件中的数据修改了,此版本号数就要加一。

二级名字服务器使用此系列号来识别区信息是否进行了修改。为了数据的更新,次要服务器定时查询主要服务器的 SOA 记录,将其系列号与本地缓存的 SOA 记录相比。如果系列号进行了改变,次要服务器就从主要服务器上将整个区数据库复制过来。

refresh 以秒为单位给出次要服务器对主要服务器进行两次 SOA 记录检查之间的时间间隔。它也是十进制数,且不能超过 8 位数。

一般来说网络的拓扑结构不会经常改变,因此对大型网络来说,此时间 间隔一般为一天,对于小的网络来说,此时间可以更长一些。

retry 此数字给出当次要服务器向主要服务器进行查询或区数据更新失败后隔多长时间在重新进行尝试。此时间不能太短,否则服务器或网络暂时性的出错会使次要服务器浪费许多网络资源。一个小时,或一个半小时可能是比较好的选择。

expire 以秒为单位给出服务器在没有能够和主要服务器取得联系的情况下,过多长时间以后必须放弃所有的区数据。它应该是比较大的,如 42 天。

minium 这是没有明确给出的资源记录的缺省 ttl 值。它要求其他名字服务器在一定时间以后放弃 RR。它与次要服务器经过多长时间以后对区数据进行更

新没有任何关系。

minium　应该取比较大的值,特别是对一些网络拓扑结构几乎不改变的 LAN。一个星期或者一个月是比较好的选择。在个别 RR 经常改变的情况下,你还可以对它们分别设置 ttl 的值。

A　它将一个 IP 地址和一个主机名相关联。资源数据域是以点分十进制形式给出的地址。

对每一个主机来说,只能有一个 A 记录。A 记录中的主机名因该是正式的主机名称。此主机的其它名字都是别名,必须通过 CNAME 记录将它们映射到正式名称上。

NS　它指向子区的主要名字服务器。其资源数据区包含名字服务器的主机名字。为了解析此主机名,需要一个附加的 A 记录,称为粘接记录(glue record),它给出名字服务器的 IP 地址。

CNAME　它将一个主机的别名与其正规主机名相关联。正规主机名是由主要文件中的一个 A 记录提供的,别名只是简单的通过 CNAME 记录与其相连,此外没有其它任何自己的记录。

PTR　这个类型的记录用来将 in-addr.arpa 域中的名字同主机名相关联。它用来将 IP 地址到主机名进行逆向的映射。主机名必须是正式名称。

MX　此类型的 RR 为一个域声明一个邮件交换器(mail exchanger)。MX 记录的语法是:

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

host 标明 domain 域的邮件交换器。每一个邮件交换器有一个整型的 preference 与其相关联。一个希望将邮件发送到 domain 域的邮件传输机构将对此域的 MX 记录进行逐个尝试,只到成功为止。尝试的顺序是从最小的 preference 开始,然后根据数值大小按递增的顺序进行尝试。

HINFO　此记录提供系统的硬件和软件信息。其语法是:

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

hardware 域给出此主机的硬件类型,硬件类型是用特殊的标记区别的。一些有效名字在“Assigned Numbers”(RFC 1340)中给出。如果此域为空,就必须用双引号括起来。software 域给出此系统的操作系统类型。有效的名字也在 RFC 的“Assigned Numbers”中规定。

3. 编写主要文件

下面给出了位于 valger 上的啤酒厂的名字服务器上的一些例子文件。由于所讨论的网络(一个 LAN)的性质,这个例子非常一目了然。

下面的 named.ca 文件给出了根名字服务器暗示记录的一个例子。一个典型的 named.ca 文件应该描述一打左右的名字服务器。可以通过下面将要介绍的 nslookup 工具获得根域的名字服务器的列表。

```
;
; /var/named/name.ca          Cache file for the brewery
;                  We'r not on the Internet, so we don't need
```

```
;               any root servers.  To activate these records, remove the semicolons.
;
; .                         99999999      IN    NS    NS.NIC.DDN.MIL
; NS.NIC.DDN.MIL            99999999      IN    A     26.3.0.103
; .                         999999999     IN    NS    NS.NASA.GOV
; NS.NASA.GOV               99999999      IN    A     128.102.16.10
```

文件 named.ca

```
;
; /var/named/named.hosts        Local hosts at the brewery
;                               Origin is vbrew.com
;
@               IN   SOA   vlager.vbrew.com. (
                        janet.vbrew.com
                        16          ; serial
                        86400       ; refresh: once per day
                        3600        ; retry:   one hour
                        3600000     ; expire:   42 days
                        604800      ; minimum: 1 week
                        )
                IN   NS      vlager.vbrew.com.
;
; local mail is distributed on vlager
                IN   MX      10 vlager
;
; loopback address
localhost.      IN   A       127.0.0.1
; brewery Ethernet
vlager          IN   A       129.72.1.1
vlager-if1      IN   CNAME vlager
; vlager is also news server
news            IN   CNAME vlager
vstout          IN   A       129.72.1.2
vale            IN   A       129.72.1.3
; winery Ethernet
vlager-if2      IN   A       129.72.2.1
vbardolino      IN   A       129.72.2.2
```

```
vchianti        IN   A        129.72.2.3
vbeaujolais     IN   A        129.72.2.4
```

named.hosts 文件

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

named.local 文件

```
;
; /var/named/named.rev        Reverse mapping of our IP address
                Origin is 72.191.in-addr.arpa.
;
@               IN   SOAvlager.vbrew.com. (
                                joe.vbrew.com . (
                                1               ; serial
                                360000          ; refresh: 100 hrs
                                3600            ; retry:   one hour
                                3600000         ; expire:   42 days
                                360000          ; minimum: 100 hrs
                                )
                IN   NS       vlager.vbrew.com.
; brewery
1.1             IN   PRT        vlager.vbrew.com.
2.1             IN   PRT        vstout.vbrew.com.
```

```
3.1             IN   PRT      vale.vbrew.com.
; winery
1.2             IN   PRT      vlager-if1.vbrew.com
2.2             IN   PRT      vbardolino.vbrew.com
3.2             IN   PRT      vchianti.vbrew.com
4.2             IN   PRT      vbeaujolais.vbrew.com
```

named.rev 文件

4. 检验名字服务器的设置

有一个很好的工具可以用来检查名字服务器的设置情况,这就是 nslookup,它既可以交互式的使用,也可以在命令行使用。后者可简单的用下列方式调用:

```
$ nslookup hostname
```

它将对 resolv.conf 中指定的名字服务器进行 hostname 的查询,(如果此文件包含的名字服务器不止一个,nslookup 将随机的抽取一个)。

交互式模式更有趣一些。除了可以查询一个单独的主机外,还可以查询任何类型的 DNS 记录,并传输一个域的所有区信息。

当不用任何参数进行调用时,nslookup 将显示其使用的名字服务器并进入交互式模式。在“>”提示符下,你可以输入任何想查询的域名。缺省的,它请求包含与此域相关的 IP 地址的 A 类记录。

你可以通过使用“set type= type”来改变这种类型,这里 type 可以是上面讨论的资源记录的名字,或者是 ANY。

下面是一个交互式调用 nslookup 的例子:

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

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

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

如果查询的名字没有 IP 地址与其相关联,但是在 DNS 数据库中发现了其它记录,nslookup 将返回一个出错信息“No type A records found”。此时可以通过“set type”命令来查询出 A 类记录以外的其他记录。例如,想得到 unc.edu 的 SOA 记录,可以使用下列命令:

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

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

Non-authoritative answer:
unc.edu
        origin= ns.unc.edu
        mail addr= shava.ns.unc.edu
        serial= 930408
        refresh= 28800 ( 8 hours)
        retry= 3600 ( 1 hour)
        expire= 1209600 (14 days)
        minimum ttl= 86400 ( 1 day)
        Authoritative answers can be found from:
        UNC.EDU nameserver= SAMBA.ACS.UNC.EDU
        SAMBA.ACS.UNC.EDU internet address= 128.109.158.30
```

通过类似的方式可以查询 MX **等记录类型。使用** ANY **类型将会返回与此名相关的所有资源类型。**

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

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

除了用于调试以外, nslookup **的一个实际应用是为** named.ca **文件获取当前的根名字服务器表列。它是通过查询与根域相关的所有** NS **记录进行的:**

```
Default Server:  ns2.net.edu.cn
Address:  202.112.0.33
```

> > Server: ns2.net.edu.cn
Address: 202.112.0.33

Non-authoritative answer:
(root) nameserver = A.ROOT-SERVERS.NET
(root) nameserver = H.ROOT-SERVERS.NET
(root) nameserver = B.ROOT-SERVERS.NET
(root) nameserver = C.ROOT-SERVERS.NET
(root) nameserver = D.ROOT-SERVERS.NET
(root) nameserver = E.ROOT-SERVERS.NET
(root) nameserver = I.ROOT-SERVERS.NET
(root) nameserver = F.ROOT-SERVERS.NET
(root) nameserver = G.ROOT-SERVERS.NET

Authoritative answers can be found from:
(root) nameserver = A.ROOT-SERVERS.NET
(root) nameserver = H.ROOT-SERVERS.NET
(root) nameserver = B.ROOT-SERVERS.NET
(root) nameserver = C.ROOT-SERVERS.NET
(root) nameserver = D.ROOT-SERVERS.NET
(root) nameserver = E.ROOT-SERVERS.NET
(root) nameserver = I.ROOT-SERVERS.NET
(root) nameserver = F.ROOT-SERVERS.NET
(root) nameserver = G.ROOT-SERVERS.NET
A.ROOT-SERVERS.NET internet address = 198.41.0.4
H.ROOT-SERVERS.NET internet address = 128.63.2.53
B.ROOT-SERVERS.NET internet address = 128.9.0.107
C.ROOT-SERVERS.NET internet address = 192.33.4.12
D.ROOT-SERVERS.NET internet address = 128.8.10.90
E.ROOT-SERVERS.NET internet address = 192.203.230.10
I.ROOT-SERVERS.NET internet address = 192.36.148.17
F.ROOT-SERVERS.NET internet address = 192.5.5.241
F.ROOT-SERVERS.NET internet address = 39.13.229.241
G.ROOT-SERVERS.NET internet address = 192.112.36.4
> Server: ns2.net.edu.cn
Address: 202.112.0.33

nslookup 的全部命令集可以通过在 nslookup 中的 help 命令获得。

5. 其它有用的工具

另外还有些工具可以帮助你成为一个 BIND 系统管理员。下面简单的介绍其中的两个,请参阅这些工具携带的文档,掌握它们的使用。

hostcvt 工具通过将/etc/hosts 文件转换成为 named 的主文件来帮助你进行初始的 BIND 配置。它同时生成 A, PTR, CNAME 等记录。当然它不能完成全部的工作,比如你可能需要修改 SOA 记录中的超时数据的值,增加 MX 记录等。hostcvt 是 BIND 源代码的一部分,但也可以在一些 Linux 的 FTP 服务器上单独获得。

在配置完名字服务器后,可能希望对配置情况进行检查。一个进行检查的理想的工具是 dnswalk。它是一个基于 perl 的软件包,它查阅整个 DNS 数据库,寻找一般性的错误,同时验证信息的一致性。dnswalk 可以在 ftp.uu.net 站点获得。

6.6 高级配置

6.6.1 SLIP 配置

SLIP(Serial Line Internet Protocol) 使用户可以通过串行线:如通过调制解调器连接的电话线,或者类似的租用线路来使用 TCP/IP。要使用 SLIP,需要你所在的地区有 SLIP 的服务器。全世界各地的许多大学和商业机构都提供了 SLIP 服务。

SLIP 利用主机上的串口来传送 IP 数据报。为此主机必须能够控制串行设备。SLIP 设备的名字为 sl0, sl1 等。它与你实际的串行设备是什么关系呢? 这里是由网络代码使用 ioctl 系统调用将串行设备改变成 SLIP 设备。现在一般有两个程序可以实现此功能: dip 和 slattach。

1. dip

dip(Dialup IP) 是一个智能程序,它可以设置串行设备的传输速率,命令调制解调器向远程主机拨号,自动将你登录到远程主机上,搜索服务器发送过来的消息并取出其中的信息,进行必须的 ioctl 调用以将你的串行设备转换成 SLIP 设备。dip 具有比较强大的功能,可以使用它来实现自动的登录过程。

dip 过去是和网络工具一块提供的,但现在它是分开开发的,因此必须分开获取。现在已经有各种不同版本的 dip,各有不同的特点。其中 dip-uri 版本的使用尤为广泛,下面介绍都以它为例。

可以在下列 FTP 站点地址获得此软件:

sunsite.unc.edu

/pub/Linux/system/Network/serial/dip337j-uri.tgz

其安装过程如下:

```
#
# cd /usr/src
# gzip -dc dip337j-uri.tgz ©¦tar xvf -
# cd dip.3.3.7j
< 编辑 Makefile 文件 >
# make install
```

#

原 Makefile 假设系统存在一个叫 uucp 的组,你可以根据自己机器的具体配置将其改为 dip 或者 SLIP。

2. slattach

slattach 与 dip 相比是个非常简单的程序,易于使用,没有 dip 那么复杂。它不具有描述功能,只是将串行设备转换为 SLIP 设备。它假设你已经知道了所有的相关信息,并且在运行它之前串行线路已经建立。当你的主机与服务器之间具有永久性的连接,如通过电缆或租借的线路时使用 slattach 是比较理想的。

3. 应该使用哪一个?

当你的 SLIP 服务器是一个拨号调制解调器或其他暂时性的连接时需要使用 dip。如果你的主机是通过租用的线路或电缆与服务器相连,不需要特殊的操作线路就可以工作,则使用 slattach 是比较合适的。

SLIP 的配置与以太网的接口十分相似,但也有一些关键的差别。

首先与以太网不同,SLIP 连接只涉及到两台主机,连接的一端各一个。另一方面以太网只要连接上以后就可以直接使用,而 SLIP 则不同,在使用之前必须根据连接类型通过特殊的方式进行网络连接的初始化。

如果使用的是 dip,这些初始化工作不需要在系统启动时进行,而是在以后准备使用连接时进行。可以使整个的初始化过程自动进行。如果使用的是 slattach,可能需要在 rc.inet1 文件中加入部分内容,我们稍后在“7. 使用租用线路的永久 SLIP 连接和 slattach”中详细介绍。

有两种主要类型的 SLIP 服务器:动态 IP 地址服务器和静态 IP 地址服务器。几乎所有的 SLIP 服务器都会在拨号进入时要用户提供帐号和密码进行登录。dip 可以自动的处理整个登录过程。

4. 拨号线的静态 SLIP 服务器和 DIP

静态 SLIP 服务器可以提供一个完全属于你的 IP 地址。每次连接到此 SLIP 服务器时都将使用该 IP 地址来配置你的 SLIP 端口。静态 SLIP 服务器将响应你的调制解调器的请求,显示输入用户名称和密码的信息,然后通过连接将任何目的地址是你的 IP 地址的数据传送给你。如果你具有一个静态 SLIP 服务器,就可以将你的 IP 地址和机器名字放入自己的/etc/hosts 文件中。同时可能也需要配置其他相关的一些文件,如:rc.inet2,host.conf,rc.local 等。请记住,在配置 rc.inet1 时,不需要为你的 SLIP 连接增加任何特殊的命令,而是由 dip 来完成所有配置接口的工作。只要给 dip 适当的信息,当命令调制解调器建立连接向 SLIP 服务器登录,它会为你自动配置端口。

如果你的 SLIP 服务器是按上述方式工作,就可以转到“6.dip 的使用”中了解如何正确的配置 dip。

5. 拨号线的动态 SLIP 服务器和 DIP

动态 SLIP 服务器是指在每次登录时从一个范围的地址中随机地为你分配一个 IP 地址。这意味着不能保证你每次登录时都是使用的同一个 IP 地址,所使用的 IP 地址在你退出后可能又分配给其他用户使用。网络系统管理员在配置此种类型的 SLIP 服务器时

会指定一个 IP 地址的范围归此服务器使用。当此服务器收到一个新的连接请求时,它寻找第一个未使用的地址,然后指导连接请求通过登录进程,并打印一个表示欢迎的消息,该消息中包含了分配给此连接的 IP 地址,在此连接持续期间都将使用此 IP 地址。

对这种类型服务器的配置与静态 SLIP 服务器的配置相似,不同的是必须增加一个步骤来获取服务器分配给你的 IP 地址以配置 SLIP 设备。

在新版本的 dip 中这些工作都可以有 dip 来做。它不但可以帮助登录,还可以自动读取欢迎消息中的 IP 地址并存储下来,以用来配置 SLIP 设备。

6. dip 的使用

前面已经介绍到 dip 是一个功能强大的程序,可以简化并自动执行拨号进入 SLIP 服务器,进行登录,开始连接,并使用合适的 ifconfig 和 route 命令配置 SLIP 设备。

使用 dip 必须写一个命令脚本文件(scripts),它由一些 dip 可以理解的命令构成,规定 dip 执行的操作。dip 是一个功能十分强大的程序,具有许多选项。这里不准备对各种选项进行介绍,可以通过阅读随 dip 软件所携带的联机手册,README 文件以及脚本文件的例子了解更多的细节。

可能会注意到作为样本的脚本文件 sample. dip,假设你使用的是静态 SLIP 服务器,因此事先知道自己的 IP 地址。对于动态的 SLIP 服务器,新版本的 dip 提供了命令,可以用来自动读取动态 SLIP 服务器分配给你的 IP 地址并配置 SLIP 设备。下面的脚本文件样本是 dip337j-uri. tag 中 sample. dip 的一个修改版本,对你来说可以作为一个比较好的起点。可以将其存储到/etc/dipscript 文件中,并对其进行编辑修改以满足你的具体的配置要求:

```
#

# sample.dip Dialup IP connection support program
#
#  This file ( should show) shows howto use the DIP.  This file should work for Annex type
#  dynamic servers, if you use a static address server then use the sample.dip file that comes
#  as part of the dip337-rui.tgz package.
#

main
# Next, setup the other side's name and address.
# My dialin machine is called "xs4all,hacktic.nl"(= = 193.78.33.42)
get $remote xs4all.hacktic.nl
# Set netmask on sl0 to 255.255.255.0
netmask 255.255.255.0
# Set the disired serial port and speed
```

```
port cua02
speed 38400
# Note! "Standard" pre-defined "errlevel" values:
# 0-OK
# 1-CONNECT
# 2-ERROR
#

# Prepare for dialing.
send ATQ0V1E1X4\r
wait OK 2
if $errlvl != 0 goto modem-trouble
dial 555-1234567
if $errlvl != 1 goto modem-trouble
# We are connected. Login to the system.
login:
sleep 2
wait ogin:20
if $errlvl != 0 goto login-trouble
send MYLOGIN\n
wait ord:20
if $errlvl != 0 goto password-error
send MYPASSWD\n
loggedin:
# We are now logged in.
wait SOMEPROMPT 30
if $errlvl != 0 toto prompt-error
# Command the server into SLIP mode
send SLIP\n
wait SLIP 30
if $errlvl != 0 goto prompt-error
# Get and Set you IP address from the server.
# Here we assume that after commanding the SLIP server into SLIP
# mode that it prints you IP address
get $locip remote 30
if $errlvl != 0 goto prompt-error

# Setup the SLIP operating parameters
```

```
get $mtu 296
# Ensure "route add -net default xs4all.hacktic.nl" will be done
default
# Say hello and fire up!
done:
print CONNECTED $locip ---> $rmtip
mode CSLIP
goto exit
prompt_error:
print TIME-OUT waitingfor sliplogin to fire up...
goto error

password:error:
print Trouble waiting for the Password: prompt...
goto error

modem_trouble:
print Trouble occurred with the modem...
error:
print CONNECT FAILED to $remote
quit

exit:
exit
```

上面的例子是假设你呼叫一个动态 SLIP 服务器,呼叫的是静态服务器,随同 dip337j-uri.tgz 的脚本文件样本 sample.dip 将会很有用处。

当给 dip 发出 get $locol 命令后,它在远端发送过来的文本信息中寻找一个像 IP 地址的字符串,即一由“.”分开的字符串。这样可以自动检测出动态 SLIP 服务器授予的 IP 地址。在上面的例子中通过你的 SLIP 连接自动建立一个缺省的路由。如果不希望这样做,比如你的机器同时还有一个以太网连接,是目前的缺省路径,可以从上面的脚本文件中将 default 命令删除掉。当此脚本文件都运行完成以后,如果运行 ifconfig 命令,就会看到一个 sl0 设备。这就是你的 SLIP 设备。如果需要的话,可以在 dip 命令执行以后使用 ifconfig和 route 命令对其进行配置。

注意到 dip 可以允许你用 mode 命令选择几种不同的协议,最通用的是选择 CSLIP,即所谓带数据压缩的 SLIP。需要注意的一点是连接两端的设置必须一致,因此必须保证自己的选择同服务器上的选择一样。

上面的例子相当强壮,可以处理绝大多数错误。参考 dip 的联机手册可以了解更多的信息。当然你可以在脚本文件中加入其他控制,比如在拨号失败后隔一定的时间后再使用

redial 命令重新拨号,如果同时访问多个 SLIP 服务器的话,还可以试着拨其他的服务器的号。

7. 使用租用线路的永久 SLIP 连接和 slattach

如果在两台机器之间有直接的电缆相连,或者通过租用的线路相连,或者具有其他永久性的串行连接,那么就不需要通过 dip 来设置串行连接。使用一个非常简单的 slattach 工具就可以配置这种类型的连接。

由于这种连接是永久性的,因此需要在 rc.inet1 文件中添加一些内容。实际上对此永久性连接所需要作的工作就是给串行设备设定合适的速度并将其转换成 SLIP 方式。slattach 允许你通过一个命令来实现这些。可以把下列部分加入 rc.inet1 文件:

```
#
#  Attach a leased line static SLIP connetion
#
#  config /dev/cua0 for 19.2kbps and cslip
/sbin/slattach -p cslip -s 19200 /dev/cua0&
/sbin/ifconfig sl0 IPA.IPA.IPA.IPA pointtopoint IPR.IPR.IPR.IPR up
#
#  End static SLIP
```

这里的 IPA.IPA.IPA.IPA 代表你的 IP 地址; IPR.IPR.IPR.IPR 代表远程主机的 IP 地址。

slattach 从 sl0 开始,将第一个未分配的 SLIP 设备分配给指定的串行设备。即 slattach 第一次将 sl0 分配给指定的串行设备,下一次是 sl1。

slattach 允许通过-p 参数配置几种不同的协议。你可以根据是否进行数据压缩而选用 CSLIP 和 SLIP。需要注意的是两端的设置必须一致。

6.6.2 把 Linux 配置成一个 SLIP 服务器

如果你的主机连在网络上,并且希望其他用户能够通过拨号访问并提供网络服务,就需要将其配置成一个 SLIP 服务器。如果使用 SLIP 作为串行线路的协议,有许多方法可以将你的主机配置成 SLIP 服务器,下面对其中的两种作详细的介绍。

1. 使用 sliplogin 的 SLIP 服务器

sliplogin 是一个应用程序,可以用来代替 SLIP 用户的登录 shell,将终端线转换成 SLIP 线。使用它可以将主机配置成动态服务器,也可以配置成静态服务器。

呼叫者像通常的登录过程一样进行登录:输入用户名和相应的密码。不同的是在用户登录之后并不是为其提供一个 shell 环境,而是执行 sliplogin 程序,该程序搜索它的配置文件/etc/slip.hosts 寻找与呼叫者登录名相同的项。如果找到相应的项,它将线路配置成 8 比特位的线路,然后运用 ioctl 调用将线路规则转换成 SLIP。当这个过程完成以后,就进行最后一步的配置。这里由 sliplogin 调用执行一个 shell 脚本文件(scripts),利用相关的 IP 地址和网络屏蔽字配置 SLIP 接口,并设置适当的路由信息。这个脚本文件通常为

/etc/sliplogin, 与 getty 类似, 如果有一些个别的用户需要特殊的初始化操作, 可以为其生成一个特殊的脚本文件, 其命名为/etc/slip.login.loginname, 就会运行此文件而不是缺省的/etc/sliplogin 文件。

为使 sliplogin 工作, 需要配置三到四个文件, 这些文件是:

- /etc/passwd, 用于建立拨号用户的帐号。
- /etc/slip.hosts, 包含每个拨号用户的独有的信息。
- /etc/slip.login, 它管理需要对用户进行的路由配置。
- /etc/slip.tty, 仅在将服务器配置成动态分配地址时才需要。
- /etc/slip.logout, 包含用户挂起或退出后的清除命令。

(1) sliplogin 软件的获取

可以从下列 FTP 站点获取 sliplogin 软件:

sunsite.unc.edu

/pub/Linux/system/Network/serial/sliplogin-2.0.tar.gz

该文件包含了源代码, 预先编译的二进制文件和联机手册的文档。

为保证只有授权的用户才可以运行 sliplogin 程序, 应该在/etc/group 中增加类似于下面所列的项:

```
..
SLIP::13:radio,fred
..
```

当安装 sliplogin 软件包时, Makefile 会自动将 sliplogin 程序的拥有组改为 SLIP, 这就意味着只有属于这一组的用户才能执行它。上面的例子中只允许用户 radio 和 fred 执行 sliplogin 程序。

此软件包的安装过程如下:

```
# cd /usr/src
# gzip -dc ../sliplogin-2.0.tar.gz ©¦tar xvf -
#  < edit the Makefile if you don't use shwdow passwords >
#  make install
```

如果希望在安装之前重新编译二进制文件, 在 make install 之前要先运行 make clean。如果希望将二进制文件安装到其他地方, 需要修改 Makefile 中的安装规则。

请参阅其 README 文件, 以获得更多的信息。

(2) 为 SLIP 主机配置/etc/passwd 文件

通常需要在/etc/passwd 文件中为 SLIP 客户创建一些特殊的登录帐号。一个传统的命名方法是在呼叫的主机前面加一个大写的‘S’作为前缀。例如, 如果有一个主机名为 radio的客户, 需要在/etc/passwd 文件中为其创建类似下面的项:

```
Sradio:Fvdkf71:123:34:radio SLIP login:/tmp:/sbin/sliplogin
```

实际上无论帐号名如何是无关紧要的, 只要对你有某种意义。

注: 呼叫者不需要任何特殊的主目录, 因为不会从此机器为其提供一个 shell, 因此选用

/tmp是合适的。可以看到sliplogin代替了一般的登录shell。

(3) 配置/etc/slip.hosts文件

sliplogin通过搜索/etc/slip.hosts文件找出与呼叫者帐户匹配的项,以获得此用户的详细的配置信息。呼叫者的IP地址和网络屏蔽字都是在这个文件中设定的。我们举一个两个主机的例子,其中的一个主机radio是静态配置,另一个主机albert是动态配置。具体配置情况如下所示:

```
#
Sradio    44.136.8.99    44.136.8.100    0xffffff00    normal
Salbert   44.136.8.99    DYNAMIC         0xffffff00    compressed
#
```

其中每一项的每一部分的含义如下:

① 呼叫者的登录名称。

② 服务器主机的IP地址,也就是本主机的IP地址。

③ 分配给呼叫者的IP地址。如果此部分标记为DYNAMIC,则根据/etc/slip.tty文件包含的信息给其分配IP地址,后面将对此文件进行介绍。需要注意的是只有在1.3以后版本的sliplogin才支持此功能。

④ 以十六进制的形式设置给呼叫机器的网络屏蔽字,如oxffffff00是一个C类网络的屏蔽字。

⑤ 可选的参数,用于使用或取消数据压缩及其他选项。

最常用的可选参数有:

normal

使用通常的不压缩的SLIP。

compressed

使用van Jacobsen压缩的数据头(CSLIP)。

二者之间是互斥的,只能使用二者之一。

(4) 配置/etc/slip.login文件

当sliplogin程序在/etc/slip.hosts文件中查找到匹配的项后,就试图执行脚本/etc/slip.login以使用IP地址和网络屏蔽字对SLIP接口进行实际的配置。

同sliplogin软件包一同发行的例子/etc/slip.login脚本文件的内容具有下列形式:

```
# ! /bin/sh -
#
#  generic login file for a SLIP line . sliplogin invokes this with the parameters:
#       $1      $2      $3          $4          $5          $6          $7-n
#  SLIPunit ttyspeed loginname local-addr remote-addr mask      opt-args
#
/sbin/ifconfig $1 $4 pointtopoint $5 mtu 1500 -trailers up
/sbin/route add $5
```

```
arp -s $5 <hw_addr> pub
exit 0
#
```

此脚本文件仅是简单的使用 ifconfig 和 route 命令来配置 SLIP 设备的 IP 地址, 远程 IP 地址, 网络屏蔽字, 并创建一个通过 SLIP 设备到远程地址的路径。

(5) 配置/etc/slip.logout 文件

当呼叫结束退出后, 要保证串行设备恢复到正常的状态以便能够顺利的进行下一次的登录。这项任务是通过调用/etc/slip.logout 脚本文件完成的。它的一个简单的样本如下:

```
# ! /bin/sh -
#
# slip.logout
#
/sbin/ifconfig $1 down
/sbin/route del $5
exit 0
#
```

所作的工作就是关闭接口, 删除开始建立的路径。

(6) 配置/etc/slip.tty 文件

如果使用动态分配 IP 地址的机制, 就必须配置/etc/slip.tty 文件以列出将什么地址设置给那个端口。如果希望你的服务器能够为用户动态的分配地址就需配置此文件。

文件由一个表组成, 每一项列出了支持通过 SLIP 连接拨号进入的 tty 设备, 以及用户从此端口拨号进入时分配给他的 IP 地址。

该文件的格式如下:

```
# slip.tty   tty-> IP address mappings for dynamic SLIP
# format:  /dev/tty?? xxx.xxx.xxx.xxx
#
/dev/ttyS0          166.111.166.100
/dev/ttyS1          166.111.166.101
#
```

这个表表明, 当呼叫者通过端口/etc/ttyS0 拨号进入时, 如果它本地主机上的/etc/slip.hosts 文件的远程主机地址域标明的是 DYNAMIC, 那么就将地址 166.111.166.100 分配给该呼叫者。

在这种方式下对所有不需要特定地址的用户在每个端口只需分配一个地址。这样就可以少用一些 IP 地址, 避免浪费。

2. 使用 dip 的 SLIP 服务器

下面介绍的内容是针对 dip337j-uri.tag 软件包的。对于其它版本的 dip 可能有一些不同, 请参见 dip 随同的文档以获得具体的信息。

dip 有一个输入操作方式,它自动为调用它的用户分配一项,并根据在/etc/diphosts 文件中找到的信息将串行线路配置成一个 SLIP 连接。这种输入操作方式是通过将 dip 作为 diplogin 调用激活的。因此使用 dip 配置 SLIP 服务器也就是创建一些特殊的帐户,用 diplogin 作为这些特殊帐户的登录 shell。

需要作的第一件事情就是建立下列符号连接:

ln -sf /usr/sbin/dip /usr/sbin/diplogin

然后需要在/etc/passwd 和/etc/diphosts 文件中各加入一项。这些项的格式如下:

通过 dip 将机器配置成 SLIP 服务器需要为用户创建一些特殊的 SLIP 帐户,用 diplogin 作为他们的登录 shell。推荐在所有 SLIP 帐户的名字前加一个大写的"S"作为起始字符,如 Sfredm。

/etc/passwd 文件中 SLIP 用户的帐号的一个例子如下所示:

Sfredm:;akdjf;a:121:43:Fred:/tmp:/usr/sbin/diplogin

此项中各项的含义依次为:

用户的登录名;加密的用户密码;用户的 ID 号;用户组 ID 号;用户的全名;用户的主目录;diplogin 作为用户的登录 shell。

用户登录以后,登录进程 login 在找到用户的帐户并进行密码检查之后,将执行 diplogin 命令。当用 diplogin 的方式调用 dip 时,它将自动把自己作为一个登录 shell。当它作为 diplogin 启动后所作的第一件事情就是使用 getuid()函数查明是谁调用的;然后搜索/etc/diphosts 文件第一个与用户的名匹配,或者是与呼叫进入的 tty 设备匹配的项,并对其进行正确的配置。通过或者在 diphosts 文件中为用户建立一项,或者给用户缺省的配置,可以将 SLIP 服务其配置成既有静态分配地址的用户,又有动态分配地址的用户。

以 diplogin 方式调用时 dip 将自动加入"Proxy-ARP"一项,因此不需要手工的加入。

(1) 配置/etc/diphosts 文件

dip 使用/etc/diphosts 文件对远程主机进行预配置。

/etc/diphosts 文件的一般格式如下:

...

Suwalt:: 166.111.166.133: 166.111.166.129: 255.255.255.0: SLIP uwalt: CSLIP, 1006

ttyS1:: 166.111.166.133: 166.111.166.129: 255.255.255.0: Dynamic ttyS1: CSLIP, 296

...

这些项中各域的含义分别如下:

1. 登录名称:由 getpwuid(getuid())返回的登录名或者是 tty 名称。

2. 没有使用:用于加密的密码。

3. 远端地址:请求呼叫主机的 IP 地址,以数字或名字的形式给出。

4. 本地地址:本机的 IP 地址,以数字或名字的形式给出。

5. 网络屏蔽字:以点分十进制形式给出。

6．注释域:可以放入任何想放入的东西。

7．协议:SLIP,CSLIP 等。

8．MTU:以十进制的形式给出。

因此,对于所有允许静态分配的 IP 地址访问的应该在/etc/diphosts 文件中有一项,如果允许用户访问一个特殊的端口以获得他们的动态分配的细节,就必须为此 tty 设备在 diphosts 文件中建立一项,而不是建立一项基于用户帐户的项。必须为每个 tty 设备配置一项,以使拨号用户无论通过什么样的调制解调器进行呼叫,都可以得到适当的配置。

当用户登录时,会向他们显示通常的登录和密码过程,在此他们输入自己的 SLIP 帐户和密码。如果权限检查通过,用户将看不到特殊的消息,在他们一端应该转成 SLIP 方式,然后连接好,并根据 diphosts 文件中的参数进行配置。

6.6.3 PPP(Point to Point Protocol)配置

PPP 是一个比较先进的,有效的协议,用来通过串行线路传送多种协议,如 tcp/ip。许多人用它来替代 SLIP。它提供了许多增强的功能,如错误检测和安全选项等。它改进了 SLIP 中已经发现的许多缺陷,对于同步链路和异步链路都提供支持。

PPP 协议的一个重要的特点是其具有协商一些参数如 IP 地址的动态分配的能力。这种特点允许 PPP 客户向服务器发送一个特殊格式的数据祯请求自己的地址。这样其配置就比 SLIP 容易。

1. 为什么要使用 PPP 而不是 SLIP?

使用 PPP 来替代 SLIP 的理由中最主要的是以下几点:

联网服务提供者只支持 PPP。

所使用的串行线路的噪音比较大。

PPP 对传输的每一帧都进行顺序检查,而 SLIP 不然。如果你所使用的串行线路的噪音较大,而又使用的是 SLIP,那么错误的纠正是在两端,即你的主机和服务器间进行的,而 PPP 的错误检测是局部进行的,在你的机器和服务器之间进行的。因此其错误纠正比较快。

需要使用 PPP 提供的其他功能。

PPP 提供了 SLIP 不具备的许多其他功能。例如通过串行线路传送除 IP 分组以外的其他数据,如 DECNET 或者 AppleTalk 数据帧等。

2. PPP 软件的获取

PPP 软件可以从下面的 FTP 站点取到:

sunsite.unc.edu

/pub/Linux/system/Networking/serial/ppp-2.1.2d.tar.gz

这个文件包含核心的源码,pppd 的源码和二进制文件。版本 1.0.0 和核心 1.0.x 配套使用,版本 2.1.2 和核心 1.2.x 配套使用,最新的版本 2.2.0 适用于核心版本 1.3.x。

3. PPP 软件的安装

PPP 软件的安装相当简单,介绍如下。

(1) 核心驱动程序

最新的核心版本提供了对 PPP 的支持。如果你使用的核心中在编译时没有选择对 PPP 的支持,不妨按照前面介绍的核心配置方法重新编译内核。当使用对 PPP 提供支持的内核时,在启动时应该看到类似下面的信息:

```
PPP: version 0.2.7(4 channels) NEW_TTY_DRIVERS OPTIMIZE_FLAGS
TCP compression code copyright 1989 Regents of the University of California
PPP line discipline registered
```

然后再使用下列命令检查/proc/net/dev 的内容:

```
# cat /proc/net/dev
```

其显示结果如下所示:

```
Inter-|   Receive                                                |  Transmit
 face |bytes    packets errs drop fifo frame compressed multicast|bytes    packets errs drop fifo colls carrier compressed
    lo:       0       0    0    0    0     0          0         0        0       0    0    0    0     0       0          0
  ppp0:       0       0    0    0    0     0          0         0        0       0    0    0    0     0       0          0
  ppp1:       0       0    0    0    0     0          0         0        0       0    0    0    0     0       0          0
  ppp2:       0       0    0    0    0     0          0         0        0       0    0    0    0     0       0          0
  ppp3:       0       0    0    0    0     0          0         0        0       0    0    0    0     0       0          0
```

这表明已经正确安装了核心驱动程序。

(2) pppd

PPP 软件和相关工具的文件的展开命令如下:

```
# cd /usr/src
# gzip -dc ppp-2.1.2d.tar.gz |tar xvf -
```

展开后得到若干个与安装有关的文件。

如果希望重新编译 ppp,直接在其安装的目录下运行 make 命令即可。在编译 lcp.c, upap.c, chap.c 等几个文件的时候可能会出现警告信息。这些警告信息可以忽略。

如果想重新编译 chat,可参阅 chat 目录下的 README.linux 文件。

安装只需要在 chat 和 pppd 目录下运行 make install 命令即可。此命令将会把 chat 和 pppd 的二进制文件放入/usr/sbin 目录,同时把联机手册 ppd.8 放入/usr/man/man8 中。

pppd 在超级用户的帐号下才能使用。

4. PPP 软件的配置和使用

PPP 软件既可以配置成一个客户,也可以配置成一个服务器。chat 程序执行与 dip 相似的功能,如自动拨号并登录到远程主机,与 dip 的区别在于它不进行 ioctl 调用将串行线路转换成 PPP 线,而是由 pppd 完成这个工作。pppd 既可以作为客户,又可以作为服务器。当它作为一个客户使用时,它调用 chat 程序进行连接的建立和登录,然后通过调用 ioctl 将线路转化为 PPP 线路,执行一些配置命令。

请参阅 pppd 和 chat 的联机手册以获取更多的信息。同 PPP 软件一同的 REAMDE 文件也值得一读。

(1) 通过拨号调制解调器配置 PPP 客户

当你有一个通过拨号调制解调器的网络提供这支持 PPP 时需要进行这种配置。当需要建立连接时只需简单的运行不带参数的 pppd 程序。这种配置假设你的 PPP 服务器为你动态的指定 IP 地址。稍后将介绍具有静态地址的问题。

pppd 程序有几个控制器行为的命令行参数。这些参数可以放在一个配置文件里面,这样当系统中的其他用户使用 ps 命令时这些参数就不会显示出来。这一点在 pppd 调用 chat 程序时尤为重要,因为此时需要密码登录到服务器,当然不希望系统中的其他用户看到你的密码。

pppd 使用的文件为/etc/ppp/options。

一个典型的 PPP 安装如下所示:

```
connect /etc/ppp/ppp-connect
/dev/ttyS1
19200
crtscts
modem
lock
asyncmap 0
defaultroute
```

他们分别说明了:

① pppd 应该使用/etc/ppp/ppp-connect 脚本文件来通过调制解调器进行拨号和登录。

② pppd 因该使用 ttyS1 设备,传输速率为 19200bps,通过 crtscts 硬件握手来检测调制解调器是否还连在线上,或呼叫是否结束。

③ pppd 应该为 tty 设备创建一个上锁文件,以保证在 pppd 使用的过程中其他进程不会使用它。

④ 线路是“8 位清除”的,也就是说所有的字符都可以通过线路自由的进行传输。asyncmap 选项决定 pppd 应该丢弃那些字符,此选项的缺省表示丢弃所有的控制字符。大多数 PPP 服务器都是 8 位清除的,并将 asymcmap 设成 0 以告诉 pppd 不要丢弃任何字符。

⑤ 当连接顺利建立以后 pppd 应该创建一个通过 ppp 设备的缺省路由。

下一 步是配置/etc/ppp/ppp-connect 脚本文件。一般来说在此脚本文件中应该使用 chat 程序,因为它的使用非常简单。

在开始时应该清除你的 PPP 服务器的登录顺序。下面是一个简单的例子:

```
CONNECT 14400
Welcome to XYZ PPP server!
login:        < you enter your username>
password    < you enter your password>
Now entering PPP:
```

. . . . . .

(2) 配置使用租借线路的 PPP 客户

为租借的线路配置 PPP 客户的过程非常简单。你还需要使用 pppd 程序，不过由于不需要建立调制解调器连接，因此传递给 chat 程序的参数非常简单。这里给出的例子中假设 PPP 服务器不需要特殊的登录过程。这样作是因为每个登录过程可能有一些差别。如果你简单的运行一个局部连接，可以进行如下的设置：

```
pppd defaultroute noipdefault debug \
kdebug 2 /dev/cua0 9600
```

它将会打开串行设备，产生 ioctl 调用将其转换成 ppp 设备，通过 ppp 接口设置缺省的路径。noipdefault 参数指示 pppd 程序从服务器请求所使用的地址。调试信息将放入系统日志中。kdebug 2 参数使得调试信息设置在 2 级上，这就可以获得更多的调试信息。它将以 9600bps 的速率使用/dev/cua0。

如果你的服务器不需要某种登录过程，就可以简单的像拨号服务器例子一样使用 chat 程序来为你完成这些操作。

(3) 配置 PPP 服务器

配置 PPP 服务器与建立一个 SLIP 服务器相似。你可以开一个特殊的 ppp 帐户，让该账户使用可执行的脚本文件作为登录的 shell。/etc/passwd 文件中的账户记录如下所示：

```
ppp:EncPasswd:102:50:PPP client login:/tmp:/etc/ppp/pplogin
```

而 ppplogin 文件的内容如下所示：

```
#!/bin/sh
exec /usr/sbin/pppd passive:192.1.2.23
```

这里的地址 192.1.2.23 是将分配给呼叫主机的地址。

如果希望同时为多个用户提供服务，就必须建立多个脚本文件，即为每个用户各建立一个，因为在每个脚本文件中只能指定一个地址。

第 7 章　网络应用程序及网络文件系统

7.1　网络应用程序

在配置好网络以后,我们现在转入介绍网络上提供的一些服务程序。本节讨论一些简单网络应用程序,包括 inetd 服务器,以及远程执行命令的配置问题。

7.1.1　inetd 超级服务器

一般来说,网络服务是通过所谓的监控进程(daemon)实现的。一个监控进程就是一个在后台运行的程序,它打开一个端口,在该端口等待外来的连接请求。如果来了一个连接请求,它就派生(spawn)一个进程接收这个连接,然后继续监视其它的连接请求。这种工作方式有一个缺点,即对提供的任何一项网络服务,必须有一个相应的监控进程对端口进行监听,以建立连接。这就意味着,当网络服务种类较多时,这些监控进程会占用大量的系统资源,如交换空间等。

因此,几乎所有的 UNIX 系统都运行一个超级服务器(super server),它为许多服务建立一个套接字(socket),通过 select(2)系统调用来同时监听所有的连接请求。当远程的主机请求其中的某一种服务时,超级服务器就为其派生相应的服务程序进行处理。

通常所用的超级服务器为 inetd。它在系统启动时开始运行,并从初始化文件/etc/inetd.conf 中确定它要管理的服务项目。除了文件中规定的服务类型外,inetd 还进行一些内部服务,例如,chargen 简单地产生一个字符串,daytime 返回系统时间等。

/etc/inetd.conf 中的每一项由一行组成,它包括下列域:

service type protocol wait user server cmdline

各域的含义如下:

service　服务名称。服务名称必须通过查找/etc/service 文件转换成相应的端口号。稍后将介绍 services 和 protocols 文件。

type　确定一个套接字(socket)类型,是 stream(面向连接的协议)或是 dgram(数据报协议)。基于 TCP 的服务应该使用 stream 类型,而基于 UDP 的服务应该选择 dgram 类型。

protocol　定义服务使用的传输协议。此协议名必须是 protocols 文件中的一个有效的协议名字。

wait　此选项仅适用于 dgram 型的套接字。其取值为 wait 或 nowait。如果说明为 wait,inetd 在任何时间为一个特定的端口仅执行一个服务程序。否则,它在执行一个服务程序后会立即继续监听该端口。stream 类型的套接字一般使用 nowait 选项。

user　此项为进程执行下的用户登录名。通常为 root 帐户,但有些服务也使用

其它的帐户。在这里最好使用最小权限规则，也就是说如果不影响程序的正确运行，最好不要在高权限的帐户下运行。

server　给出服务程序的全路径名。内部服务标记为 internal。

cmdline　传递给服务程序的命令行参数。通常为服务程序名，除非程序以不同的名称被调用时的行为不同。对内部服务来说，此域为空。

下面是/etc/inetd.conf 文件的一个例子。考虑到安全原因，其中 finger 服务被注释掉了，目的是防止他人通过它获得你系统中的用户情况。

```
#
#  inetd services
ftp          stream tcp nowait root /usr/sbin/ftpd in.ftpd -l
telnet       stream tcp nowait root /usr/sbin/telnetd     in.telnetd -b/etc/issue
# finger     stream tcp nowait bin /usr/sbin/fingerd      in.fingerd
login stream tcp nowait root /usr/sbin/rlogind in.rlogind
shell        stream tcp nowait root /usr/sbin/rshd in.rshd
exec         stream tcp nowait root /usr/sbin/rexecd in.rexecd
#
#       inetd internal services
#
daytime stream tcp nowait root internal
daytime dgram udp nowait root internal
time stream tcp nowait root internal
time dgram udp nowait root internal
echo stream tcp nowait root internal
echo dgram udp nowait root internal
discard stream tcp nowait root internal
discard dgram udp nowait root internal
chargen stream tcp nowait root internal
chargen dgram udp nowait root internal
```

7.1.2　tcpd 访问控制机制

将机器开放，允许别人通过网络进行访问时要考虑到系统安全的问题。有一些工具可以用来保护几种类型的入侵。但这些工具中的一部分，或者留有许多漏洞，或者不能有区别的接受或拒绝来自不同主机的服务请求，也就是说无法将一些服务仅提供给可以信任的主机。而一般的设置不能够做到这一点，inetd 或者将服务提供给所有客户，或者对所有用户都不提供服务。

一个有用的工具是称为监控进程限制器(daemon wrapper)的 tcpd。可以通过调用此程序而不是服务进程对 TCP 服务进行监视或保护。tcpd 将服务请求传递给 syslog 监控进程，检查是否允许远程主机使用该服务，只在允许时才执行真正的服务器程序。需要注

意的是它对于基于 UDP 的服务不起作用。

例如要限制 finger 监控进程,必须将 inetd.conf 中的相应行改为:

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

如果不增加任何访问控制,除了请求被传递到 syslog 授权机制的服务以外,对客户来说与一个通常的 finger 设置一样。(读者可对比上一个例子中被注释掉的 finger 服务。)

访问控制是通过两个文件/etc/hosts.allow 和/etc/hosts.deny 来实现的。它们包含一些表项,分别表示允许或拒绝一些主机对特定服务的访问。当 tcpd 处理一个来自名叫 p12.tirc.cs 的客户机的 finger 服务请求时,它按 hosts.allow 和 hosts.deny 的顺序查找这两个文件的内容,寻找其中服务和客户主机名字都匹配的项。如果在 hosts.allow 中找到了匹配的项,无论 hosts.deny 的内容如何,都对此访问进行响应。如果在 hosts.allow 中没有找到了匹配的项,而在 hosts.deny 中找到了匹配的项,则此请求被拒绝,并关掉连接。如果没有任何匹配,则接收服务请求。

访问控制文件的每一项具有下列格式:

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

servicelist 是/etc/services 中服务名的一个表列,或者是关键字 ALL。ALL 表示所有服务。如果想表示除 finger 和 tftp 之外的所有服务,可以使用"ALL EXCEPT finger,tftp"。

hostlist 是一串主机名或者是 IP 地址,或者是关键字 ALL,LOCAL 和 UNKNOWN。ALL 匹配所有的主机,LOCAL 匹配不带"."的主机名,UNKNOWN 匹配所有名字和 IP 地址查找失败的主机。一个以"."开始的名字匹配所有具有同样域名的主机。例如 .cs.tsinghua.eud.cn 可以匹配 tirc.cs.tsinghua.edu.cn。还可以以 IP 地址和网络屏蔽字的形式定义匹配的主机,更详细的资料请参阅 hosts_access(5) 的联机手册。

如果要拒绝除本地主机以外的所有主机访问 finger 和 tftp 服务,应将下列行放入/etc/hosts.deny,而/etc/hosts.allow 为空:

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

可选的 shellcmd 域可以包含一个 shell 命令,在此项匹配后被调用执行。可以利用此项功能来设置陷阱以揭露潜伏的攻击者:

```
in.ftpd: ALL EXCPET LOCAL .cs.tsinghua.edu.cn: \
    echo "resquest from %d@%h" >> /var/log/finger.log; \
    if [ %h != "tirc.cs.tsinghua.eud.cn" ]; then \
        finger -l @%h >> /var/log/finger.log \
    fi
```

%h 和%d 分别被 tcpd 扩展为客户主机名和服务名称。

7.1.3 services 和 protocols 文件

一些标准服务的端口号由 RFC 在"Assigned Numbers"中定义。要想使服务器和客户程序将服务名转换成为这些端口号,每台机器最少要有"Assigned Numbers"中的一部分。它保存在/etc/services 文件中。此文件的每一项具有下列形式:

service port/protocol [aliases]

这里 service 定义服务的名称, port 定义提供此服务的端口号, protocol 定义使用何种传输协议。通常为 udp 或 tcp。一个服务可以通过多种传输协议提供, 同一端口也可以提供多种服务, 前提是使用的协议不同。aliases 域可以用来定义服务的别名。

通常是不需要改变 Linux 系统中与网络软件一同带来的此文件。下面是这个文件的一个样本:

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

可以看到, 在上面这个样本文件中 echo 服务在端口 7 上同时向 TCP 和 UDP 提供, 而端口 512 提供了两种不同的服务。

同 services 文件类似, 网络库函数需要一种方式来转换协议名字, 例如将 services 文件中的协议名转换成为其它主机 IP 层可以理解的 IP 号码。这种转换通过查找/etc/protocols 文件完成的。此文件中每一项占用一行, 包含一个协议名和相关的数字。与/etc/

services 文件相比,此文件更不需要自己动手进行修改。下面是此文件的一个样本,供读者了解:

```
#
#  Internet (IP) protocols
#
ip          0       IP      # internet protocol, pseudo protocol number
icmp        1       ICMP    # internet control mesage protocol
igmp        2       IGMP    # internet group multicast protocol
tcp         6       TCP     # transimission control protocol
udp         17      UDP     # user datagram protocol
raw         255     RAW     # RAW IP interface
```

7.1.4 远程过程调用(RPC)

RPC(Remote Procedure Call),即远程过程调用,是一个非常通用的客户-服务器(client-server)应用机制。RPC 是由 Sun Microsystems 公司开发的一个工具和程序库软件包。在 RPC 之上最重要的应用程序有 NFS(网络文件系统)和 NIS(网络信息系统),我们将详细介绍其中的网络文件系统。

RPC 服务器由一组应用程序组成,客户通过向服务器发送 RPC 请求和进程调用参数来调用服务进程。服务器根据客户的请求启动相应的应用程序,并将可能的结果返回给客户。为了独立于具体的机器结构,在客户和服务器间的所有数据交换都通过外部数据表示(XDR, eXtern Data Representation)进行。

RPC 服务器和客户之间的网络通信比较特殊。RPC 服务器提供一个或多个进程构成的集合,每个集合成为一个 program,由一个唯一的 program 号标示。服务名到 program 号的映射表放在/etc/rpc 文件中,下面是此文件的一个样本:

```
#
#  /etc/rpc - miscellaenous RPC-based services
#
portmapper      100000      portmap sunrpc
rstatd          100001      rstat rstat_svc rup perfmeter
rusersd         100002      rusers
nfs             100003      nfsprog
ypserv          100004      ypprog
mountd          100005      mount showmounts
ypbind          100007
walld           100008      rwall shutdown
yppasswdd       100009      yppasswd
bootparam       100026
ypupdate        100028      ypupdate
```

在 TCP/IP 网络中,RPC 作者面临着将程序号映射到一般网络服务的问题。它们选择对服务器的每个程序和每个版本同时提供 TCP 和 UDP 端口。一般来说,RPC 应用程序都使用 UDP 来传送数据,只有当数据不能同时放在一个 UDP 分组时才使用 TCP 传输协议。

当然,客户程序必须有办法找出程序号被映射到哪个端口。如果使用配置文件来解决这个问题有些太不灵活,因为 RPC 应用程序不使用保留的端口,没有机制保证我们程序使用的端口不再被其它进程使用。因此 RPC 应用程序选用它们可以获得的端口,并将其向一个叫 portmapper 的监控进程进行注册。该监控进程做为该机器上所有 RPC 服务器的一个阶段:客户如果需要联系一个给定程序号的服务,首先向服务器主机上的该监控进程进行查询,由其返回该服务的 TCP 或 UDP 端口号。

这种方法一个缺点,即一旦 portmapper 死掉,所有的 RPC 端口信息都会丢失;这就意味着你必须逐个重新启动所有的 RPC 服务器,或者重新启动整个机器。

在 Linux 上,此 portmapper 程序叫着 rpc.portmap,位于/usr/sbin 目录下。要保证其从 rc.inet2 中启动,这个程序不需要任何配置。

7.1.5 配置远程命令

有许多命令用来在远程主机上执行命令。如 rlogin, rsh, rcp 和 rcmd 等,它们都在远程主机上生成一个 shell,允许用户执行命令。当然用户需要在执行命令的主机上有自己的帐号。通常用户需要告诉服务器自己的帐号和密码。

有时希望对一些特定的用户放松权限检查。例如,你经常要登录到局域网上的其它机器,可能希望不需要每次登录时都输入密码。这里仅建议对少部分密码数据库相同的少数主机,或者对少数因系统管理原因访问此主机的特殊用户放松权限检查。无论何时,如果允许其它用户可以不输入登录名和密码就可以进入你的主机,需要其保证不要偶然将访问权限再授给其他人。

有两种方式可以取消对远程命令的权限检查。一种方法是超级用户可以允许特定或全部主机上的特定或全部用户不输入密码就可以登录。这种访问是通过/etc/hosts.equiv 文件控制的。它包含一个主机和用户名的表列,这些用户同本地的用户处于同一地位。另一个方法是用户可以授权给其它主机上的其他用户访问自己的帐户,这些用户信息包含在该用户主目录下的.rhosts 文件中。为安全起见,这个文件必须归该用户或超级用户所有,并且不能是符号连接,否则该文件就不起作用。

当一个客户请求远程服务时,提供服务的主机在/etc/hosts.equiv 文件中搜索其主机名和用户名,然后搜索它希望登录的用户主目录下的.rhosts 文件。例如,假设 gauss 主机上的用户 janet 希望登录到 euler 主机上的 joe 帐号,在下面的叙述中我们把 Janet 作为客户用户,而 Joe 是本地用户。当 Janet 在主机 gauss 上敲入下列命令:

```
$rlogin -l joe euler
```

服务器将首先查看 hosts.equiv 文件,检查是否允许其自由访问,如果不允许,则继续查看 joe 主目录下的.rhosts 文件。

euler 主机上的 hosts.equiv 文件具有下列形式:

gauss

euler

-public

quark.physics.groucho.edu andres

每一项由一个主机名和可选的用户名组成。如果主机名后面没有任何用户名,则该主机上的任何用户都可以自用访问他们的本地帐号。在上面的例子中,Janet 将被允许从 gauss 上登录到 euler 主机上的 janet 帐户,euler 主机上除超级用户以外的其他用户也都可以同样登录到 gauss 主机上他们的帐户。然而,如果 Janet 想以 joe 的帐户登录,就会像通常一样输入登录名和密码。

如果主机的名字后面有用户名,如上面的最后一行所示,这个用户将可以不用输入密码访问此主机上除 root 之外的所有帐户。

主机名的前面也可以加一个负号,如“-public”所示。它表明对于 public 主机上的所有用户,无论.rhosts 文件中授予了个别用户什么权利,都需要进行权限检查。

文件.rhosts 的格式同 hosts.equiv 相同,但表示的含义有些微小的差别。我们考虑 euler 主机上 Joe 的.rhosts 文件:

chomp.cs.groucho.edu

gauss janet

第一项允许 Joe 可以从 chomp.cs.groucho.edu 主机上自由登录访问,但并不影响 euler 或 chomp 主机上的其他用户的访问权限。第二项允许 janet 自由的从 gauss 上访问 joe 的帐户。

7.2 网络文件系统 (NFS)

网络文件系统 (NFS, Network File System) 可能是使用 RPC 的最主要的网络服务。它使得用户可以像访问本地文件一样访问远程主机上的文件。NFS 是由客户和服务器共同合作实现的:在客户一边,通过一些核心函数调用来使用远程文件系统;在服务器一边,由 NFS 服务进程来提供文件数据。文件的访问对客户来说是完全透明的,并且 NFS 可以跨越各种服务器和主机平台运行。

NFS 具有以下优点:

- 对于所有用户都需访问的数据,可以将其保存在一个中心主机上,客户只需在系统启动时将其安装到本地目录,即可对其进行访问。例如,将所有用户的帐号都放在一个中心主机上,网络上的其它主机都从该机安装 /home 目录。如果同时还安装了网络信息系统(NIS, Network Information System),用户就可以登录到网络上的任何系统,而实际上还是在一个文件集合上工作。
- 可以将占用大量磁盘空间的数据只保存在一个主机上。例如,于 LATEX 和 METAFONT 相关的文件和程序都可以只保存在一个地方,其它主机通过 NFS 将其安装到本地目录进行透明的访问。
- 系统管理数据可以只放在一个主机上,而不需要再使用 rcp 命令等命令将同一文件安

装到多个不同的机器上。

我们现在来了解一下 NFS 的工作过程: 首先, 客户请求将远程主机的一个目录安装到本地目录, 此请求就像安装一个本地物理设备一样, 只是定义远程目录与定义本地物理设备的语法有所不同。例如, 如果希望将主机 host1 上的/home 目录安装到 host2 上的/users目录, host2 上的系统管理员运行下列命令:

```
# mount -t nfs host1: /home /users
```

mount 将试图通过 RPC 与 host1 上的安装监控进程 mountd 建立连接。服务监控进程将检查 host2 是否可以安装此目录, 如果允许, 就向 host2 返回一个文件句柄。以后对/users目录下文件的访问都通过此文件句柄进行。

当通过 NFS 对文件进行访问时, 核心向服务器主机上的 nfsd (NFS 监控进程) 发出 RPC 调用。该调用携带文件句柄、要访问的文件名以及用户的用户号和组号等参数。这些参数被用来检查是否有权限访问指定的文件。为防止未被授权的用户阅读和修改文件, 两个主机上的用户号和组号应该一致。

7.2.1 准备网络文件系统

在使用 NFS 之前, 无论是客户机还是服务器主机, 都需保证操作系统内核在编译时提供了对 NFS 的支持。可以通过下列方法来检查核心是否支持 NFS:

```
$cat /proc/filesystem
minix
ext2
msdos
nodev proc
nodev nfs
```

如果此命令显示的各文件系统中没有 nfs 这一项, 就必须对内核进行重新编译, 使它提供对 NFS 的支持。

另外一种检测办法是:

```
# mkdir /tmp/test
# mount localhost: /etc /tmp/test
```

如果此命令失败, 出现如下的出错信息:

```
fs type nfs not supported by kernel
```

则说明当前的核心程序不支持 NFS, 需要重新对内核进行编译。如果出现的是其它出错信息, 则无所谓, 因为还没有在主机上配置 NFS 监控进程。

7.2.2 安装 NFS 卷

安装 NFS 文件卷(volume, 这里不称之为文件系统是因为它与标准的文件系统有一定区别)和安装通常的文件系统十分相似。使用 mount 命令的语法如下:

```
# mount -t nfs nfs-volume local-dir options
```

nfs-volume 以 remote-host:remote-dir 的形式给出。由于这种形式的描述仅有 NFS

文件系统使用,因此可以将 -t nfs 选项省略。

在安装 NFS 文件卷时还可以附加一些选项。这些选项或者通过-o 开关直接在命令行使用,或者出现在/etc/fstab 中该文件卷相关行的选项域。命令行的选项可以覆盖 fstab 文件中的选项。

/etc/fstab 文件中 NFS 文件卷说明形式如下:

```
# volume                mount point          type      options
news:  /usr/spool/news  /usr/spool/news      nfs       timeo= 14, intr
```

然后就可以使用下列命令进行文件卷的安装:

```
# mount news:  /usr/spool/news
```

如果 fstab 文件中没有相应的说明项,NFS 的 mount 命令的使用就稍微复杂了。例如,假设你从名叫 host 的主机上安装用户目录,该机使用缺省的 4K 大小的块读写操作,你就需要通过下列办法将其块大小减到 2K 以适应 Linux 数据报大小的限制:

```
# mount host:  /home /home -o rsize= 2048, wsize= 2048
```

下面介绍常见的几个选项:

rsize= n, wsize= n。分别规定 NFS 客户进行读写请求的数据报的大小。它们现在的缺省值是 1024。

timeo= n。它以十分之一秒为单位规定 NFS 客户等待请求完成的时间。缺省的是 0.7秒。

hard。明确标明此文件卷是硬安装(hard-mount)。这个选项缺省处于设置状态。

soft。软安装驱动程序(与硬安装相对)。

intr。允许 NFS 调用被信号(signal)中断。此选项设置以后如果服务器没有反应可以放弃操作。

除了 rsize 和 wsize 选项外,其它的选项都用于服务器暂时不能访问时对客户的行为进行控制。它们按下列的方式协同工作:无论客户什么时候向 NFS 服务器发送一个请求,它期待请求的操作可以在一个给定的时间段内可以完成(此时间段的长度由 timeout 选项规定)。如果在此时间段内没有收到应答信息,就认为发生了"次超时"(minor timeout)。这时重新进行前一次的操作,并将超时时间延长一倍。当超时时间达到最大值 60 秒时,称发生了"主超时"(major timeout)。

默认情况下,当主超时发生后客户会在控制台打印一条消息,然后从头开始前次的操作,不同的是初始的超时时间段的长度是前一轮的两倍。这种操作有可能永远进行下去。对于一个 NFS 卷来说,如果对它的操作不停的进行重试,直到访问到服务器为止,则称此卷是"硬安装"的。相反,软安装的卷在发生主超时后会为调用进程产生一个 I/O 错误。由于 Linux 中采用了 UNIX 系统都采用的磁盘缓存技术,对磁盘的写操作并不一定立即完成,上述出错信息在调用进程下一次调用 write(2)以前并不传递给该进程本身,因此应用程序永远不能保证它对一个软安装卷的写操作是否成功。

对一个 NFS 卷进行软安装还是硬安装不仅仅是个人的口味问题,还与你希望从安装过程中得到的信息类型有关。例如,如果通过 NFS 安装 X 应用程序,当然不希望因为网络的暂时故障而中断 X 对话。通过硬安装,在这种情况下计算机就会等待,直到重新与

NFS 服务器建立联系。另一方面,对于一些非关键性的数据,例如,用 NFS 安装一个新的分区,或者是 FTP 文档,可以进行软安装,因为它们在远程主机暂时不能访问时不会挂起你的对话。如果你与服务器之间的网络连接不太可靠或者需通过一个负载比较重的路由器,就需要或者利用 timeout 选项增减超时时间,或者进行硬安装,但应该允许信号对 NFS 调用的中断,这样仍然可以中断挂起的文件访问。

7.2.3 NFS 监控进程

如果希望向其它主机提供 NFS 服务,就必须在机器上运行 nfsd 和 mountd 监控程序。像基于 RPC 的程序一样,它们不由 inetd 管理,而是在系统启动时开始执行,并通过 portmapper 对自己进行注册。因此必须保证在 rpc.portmap 运行后再启动它们。通常,可以在 rc.inet2 中包含以下几行的内容:

```
if [ -x /usr/sbin/rpc.mountd ]; then
    /usr/sbin/rpc.mountd; echo -n "mountd"
fi
if [ -x /usr/sbin/rpc.nfsd ]; then
    /usr/sbin/rpc.nfsd; echo -n "nfsd"
fi
```

NFS 监控进程提供给客户的文件归属信息一般都只是其用户和所属用户组的数值标识号。如果客户和服务器上用户和组名的数值标识都相同,就称他们共享同一个 uid/gid 空间。例如,利用 NIS 将 passwd 信息分布到局域网上的所有主机就属于这种情形。

然而在有些情况下不同主机上相同用户名和组名的数值标识符是不相同的。一种解决办法是将所有客户机上用户及其组号都根据服务器上的数值进行调整,另一种是通过 ugidd 映射监控进程来解决这个问题。通过使用下面将要介绍的 map-daemon 选项,可以告诉 nfsd 在客户机的 ugidd 的帮助下将服务器的 uid/gid 空间映射到客户机的 uid/gid 空间。

ugidd 是一个基于 RPC 的服务程序,像 nfsd 和 mountd 一样从 rc.inet2 中启动:

```
if [ -x /usr/sbin/rpc.ugidd ]; then
    /usr/sbin/rpc.ugidd; echo -n "ugidd"
fi
```

7.2.4 exports 文件

上面讨论的是客户的 NFS 配置,在服务器这边,也可以通过一些选项来配置客户的行为。这些选项都设置在/etc/exports 文件中。

就默认设置来说,mountd 并不允许任何人从本地主机安装其目录,从安全角度来说这是一个比较明智的态度。要想允许一个或多个用户可以以 NFS 形式安装其目录,就必须通过在 exports 文件中进行说明,以将其公开出去。下面是 exports 文件的一个例子:

```
# exports file for host1
/home        vale(rw)      vsout(rw)      vlight(rw)
```

```
/usr/X386    vale(ro)        vstout(ro)      vlight(ro)
/usr/TeX     vale(ro)        vstout(ro)      vlight(ro)
/            vale(rw,no_root_squash)
/home/ftp    (ro)
```

每一行定义了一个目录和可以安装它的主机。主机名通常是一个正规的域名,但有的包含了"*"和"?"通配符。例如lab*.dcs.tsinghua.edu.cn可以匹配lab01.dcs.tsinghua.edu.nc,也可以匹配laber.dcs.tsinghua.edu.cn。如果没有给出主机名,如上述文件中的/home/ftp,则表明任何主机都可以安装此目录。

当利用exports文件对客户主机进行权限检查时,mountd使用gethostbyaddr(2)系统调用来查找客户的主机名字。在配有DNS的系统中,此系统调用将返回客户的正规主机名字,因此必须保证在exports文件中不能使用别名。在没有DNS的系统中,此调用将返回hosts文件中与客户地址匹配的第一个主机名字。

主机名后面可以跟一些括起来的用逗号分开的可选标志符。这些标志符可以取下列值:

insecure。允许从此机器进行未授权的访问。

unix-rpc。向此机器请求UNIX域的RPC授权。即要求请求必须从一个保留的网间网端口发出(亦即端口号必须小于1024)。此选项缺省是设置的。

secure-rpc。向此机器请求安全的授权RPC。目前这一部分还没有实现。可以参见SUN公司发行的有关安全RPC的文献。

kerberos。向此机器请求Kerberos授权。目前这一部分还没有实现,可参阅MIT发表的关于kerberos授权系统的文献。

root_squash。这是一个安全特性,通过将客户机上uid为0的请求映射到服务器主机上的uid 65534(-2)来取消特定主机上的超级用户的特殊访问权限。uid 65534应该与用户nobody相关联。

no_root_squash。不对uid为0的请求进行映射,此选项缺省处于设置状态。

ro。安装的文件系统为只读的。为缺省选项。

rw。安装的文件系统为可读写的。

link_relative。通过适当的"../"等前缀将绝对符号链接(以/开始的链接)转换成为相对链接。此选项在映射一个主机的所有文件系统时才有意义,否则一些链接将什么地方也不指向,有时更糟糕的是指向一些其它的文件。此选项缺省情况下处于设置状态。

link_absolute。保持所有的符号连接(Sun提供的NFS服务器的通常行为)。

map_identity。此选项告诉服务程序客户机中使用的uid和gid与服务器主机相同,此选项缺省情况下处于设置状态。

map_daemon。此选项告诉NFS服务程序假设客户机和服务器主机没有共享同一个uid/gid空间。

nfsd将通过向客户机的ugidd监控进程查询建立一个客户机和服务器主机间的标识号映射。

附录 A　Linux 的信息来源

本附录列举了 Linux 信息的各种来源，诸如在机器上的文献、书籍等。许多文献是存放在 Internet 或 BBS 电子媒介上的。许多 Linux 发行版本身也带有部分文献。因此，当你安装上 Linux 之后，可能就会发现这里所列出的参考文献已经在你的系统里了。

A.1　联 机 文 献

以下所列的联机文献可以从任何一个 Linux 的 FTP 站点上取得(附录 B 列出了部分站点)如果你不与 Internet 联接，则可以从 BBS 或其他网络服务方式上寻找。如果你可以使用 Internet 电子邮件服务，就可以用 ftpmail 来获取这些文献。用户还可以用系统提供的 Internet 浏览程序，如 gopher 或 netscape 等，来交互式阅览这些联机文献。

The Linux Frequently Asked Questions List

这是一份关于 Linux 的常见问题及答案的总汇文献。从这些常见问题和答案中，读者可以了解到不少关于 Linux 的知识。也可从中得到有关 Linux 信息的其他来源。每一位新用户若有机会读一读这一文献，对学习 Linux 将会很有帮助。

The Linux META-FAQ

这是一份关于 Linux 的系统及一般讨论的情况的总汇。从这里读者可以获得更多的关于 Linux 系统的资料。

The Linux INFO-SHEET

这是一份对于 Linux 系统作技术性介绍的文献。它概述了系统的功能、特性以及可在 Linux 上运行的软件，也提供了一些 Linux 的一般的资料。

The Linux Software Map

这份文献的中文名可以译成“Linux 软件分布图”。故名思义，这是一份介绍 Linux 系统上众多的应用程序的资源分布的文献。它包括了各种软件从何处获取、由何人维护等重要信息。当然，这一份资料远不是 Linux 上应用程序的完整的清单。正如第 1 章中所说，Linux 的开发是开放式的，没有一个严密的组织机构，几乎每天都有新的软件移植或开发到 Linux 系统上。因此要列出一份完整的名单来几乎是不可能的。这里所提供的是一些比较常用的应用程序。如果你想寻找在 Linux 上的软件，可以先从这一份资料着手。

另外有一套介绍如何使用 Linux 的系列文献，分别较详细地介绍了具体的 Linux 各

个方面。它们是:

The Linux HOWTO Index: 本套系列文献的目录。

The Linux Installation HOWTO: 介绍如何安装 Linux。

The Linux XFree86 HOWTO: 介绍如何安装和配置 X 窗口系统软件。

The Linux Mail, News, and UUCP HOWTOs: 这三份文献分别介绍了配置和安装电子邮件, 电子新闻及 UUCP 通信的方法。由于这三方面有较紧密的联系, 所以读者们可将此三份文献一起阅读参考。

The Linux Hardware HOWTO: 详细介绍了 Linux 支持的主要硬件。这里并非包括了所有的硬件, 但是读者可从中看出一个清楚的轮廓。

The Linux SCSI HOWTO: 介绍了诸如磁盘、磁带、光盘等设备的配置及使用方法。

The Linux NET-2-HOWTO: 介绍了 TCP/IP 软件, 包括 SLIP 的安装、配置方法。

The Linux Ethernet HOWTO: 介绍 Linux 支持的各种的以太网设备, 阐述了如何配置和使用的方法。

The Linux Printing HOWTO: 介绍了 Linux 支持的打印软件, 如 lpr。同时也介绍了打印机和打印软件的配置。

A.2 Linux 手册

有一项称为 Linux Documentation Project(简称 LDP)的计划正着力于 Linux 手册及程序说明的编写。这些手册的编写工作各处于不同阶段。有几本已经可以从 FTP 站点取到或从 Linux 发行商处买到。下面所列的是几本已经完成的手册:

《Linux Installation and Getting Started》, Matt Welsh 著。这是一本 Linux 的使用指南。

《The Linux System Administrators' Guide》, Lars Wirzenius 著。这是一本运行、配置 Linux 系统的较完全的指南。有许多 Linux 系统特有的与系统管理方面的问题, 诸如文件系统管理, 作文件后备, 多用户的支持与管理等在本书中都有叙述。

《The Linux Network Adminiatrators' Guide》, Olaf Kirch 著。这是一本关于 Linux 上的网络软件的较为详细和全面的指南。

《The Linux Kernel Hackers' Guide》, Michael Johnson 著。这本指南介绍了 Linux 系统的核心程序的概念和内部结构, 是指导和帮助用户在 Linux 核心程序中添加或修改功能的很好的资料。

A.3 参考书及其他出版物

正如前面已经说过的, 关于 Linux 的参考书寥寥无几。但是, 由于 Linux 与 Unix 的特殊关系, 我们在这里列出一些可供读者参考的 Unix 的参考书。

A.3.1 关于 Linux 的使用

书名: Learning the Unix Operating System
著者: Grace Todino & John Strang
出版社: O'Reilly and Associates, 1987
ISBN: 0-937175-16-1

书名: Learning the vi Editor
著者: Linda Lamb
出版社: O'Reilly and Associates, 1990
ISBN: 0-937175-67-6

A.3.2 系统管理

书名: Essential system Administration
著者: Eleen Frisch
出版社: O'Reilly and Associates, 1991
ISBN: 0-937175-80-3

书名: TCP/IP Network Administration
著者: Craig Hunt
出版社: O'Reilly and Associates, 1990
ISBN: 0-937175-82-x

书名: Managing UUCP and Usenet
著者: Tim O'Reilly and Grace Todino
出版社: O'Reilly and Associates, 1991
ISBN: 0-937175-93-5

A.3.3 X 窗口系统

书名: The X Window System: A User's Guide
著者: Niall Mansfield
出版社: Addison-Wesley
ISBN: 0-201-5134102

A.3.4 程序设计

书名: The C Programming Language
著者: Brian Kernighan and Dennis Ritchie
出版社: Prentice-Hall, 1988

ISBN: 0-13-110362-8

书名: The Unix Programming Environment
著者: Brian Kernighan and Bob Pike
出版社: Prentice-Hall, 1984
ISBN: 0-13-937681-X

书名: Advanced Programming in the Unix Environment
著者: W. Richard Stevens
出版社: Addison-Wesley
ISBN: 0-201-56317-7

A.3.5 核心程序

书名: The Design of the Unix Operating System
著者: Maurice J. Bach
出版社: Prentice-Hall, 1986
ISBN: 0-13-201799-7

书名: The Magic Garden Explained
著者: Berny Goodheart and James Cox
出版社: Prentice-Hall, 1994
ISBN: 0-13-098138-9

附录 B　FTP 使用简介及站点表

文件传送协议, 简称 FTP, 是 Internet 在机器系统之间进行文件传输的一些程序的集合。大部分 Intenet 上的系统, 无论是 UNIX 或是 MS-DOS, 都有一个名为 ftp 的程序用于文件的传输。如前面章节中提到的, 如果你可以上 Internet, 获取 Linux 的最好办法就是用 ftp 下载(接收)Linux 软件。本附录只列出一部分 ftp 的功能和操作方法, ftp 的功能远不止于此。

本附录的末尾列出了存放 Linux 软件的 FTP 站点。如果你不能直接上 Internet, 但是能与 Internet 交换电子邮件, 则可以用 ftpmail 命令来获取 Linux 软件, ftpmail 的用法也在这里作简单介绍。

如果你使用的是 MS-DOS, UNIX 或 VMS 系统从 Internet 上下载文件的话, ftp 是一个命令驱动程序。但是, 也有一些不同的 ftp 的实现, 如 Macintosh 版的 ftp, 也称为 fetch, 是一个具有菜单驱动界面的程序, 很容易掌握。即使你用的不是命令驱动的 ftp, 这里介绍的 ftp 用法也会对你学会使用其他方式的 ftp 有一定的帮助。

ftp 可以用来从 Internet 站点上上载(传送)或下载(接收)文件。在大多数情况下, 是用它来下载软件。在 Internet 上, 有很大量的任何人都能取的 FTP 文件站点。这些站点是在网络上的允许任何人随意下载其上的软件和文件的机器。存放 Linux 软件的主要 FTP 站点是 sunsite. unc. edu。

B. 1　启 动 ftp

在介绍如何使用 ftp 之前, 首先要说明一点, 即在介绍用法的例子中给出的"屏幕"并不一定与你在机器上看到的屏幕显示的内容完全一致。我们省略了一些无关紧要的显示内容。另外, 用斜体字(italics)表示用户键入的内容, 其余的则是系统显示的内容。

要启动 ftp 并连接到某一站点, 只要简单地键入

ftp< hostname>

这里< hostname> 是你要连接的站点的名称。例如, 你要连接到名为 sunsite. unc. edu 的站点上, 只要键入 ftp sunsite. unc. edu 即可。

B. 2　登　　录

当启动了 ftp 后, 我们将从屏幕上看到如下显示内容:

connected to sunsite. unc. edu

220 sunsite. unc. edu FTPD ready at 15 Dec 1995 08: 20: 42 EDT

Name (sunsite. unc. edu: mxx):

这里 ftp 显示了站点已经准备就绪以及当地的时间,并且寻问要登录到该机器上的用户名。如果没有用户名输入,ftp 默认为是 mxx,即用户在启动 ftp 的机器上的名字。一般都用 anonymous 来登录到可供任何人下载软件的 FTP 站点上。ftp 在接收到用户输入的用户名之后会要求输入登录口令,一般用 anonymous 登录的话,就用用户的电子邮件地址作为口令。这时屏幕显示的内容为:

```
Name (sunsite.unc.edu: mxx): anonymous
331-Guest login ok, send e-mail address as password.
Password: mxx@here.tsinghua.edu
230-Welcome to sunsite.unc.edu
ftp>
```

这里 mxx@here.tsinghua.edu 是用户的电子邮件地址。最后一行是 ftp 的输入提示符,表示 ftp 正在等待用户的输入命令。

B.3 确定文件位置

在见到 ftp 输入提示符后,用户就可以输入 ftp 命令了。下面是几个最基本的 ftp 命令:ls< file> 和 dir< file> 是二个列文件名的命令。这里< file> 指定了要 ftp 列的文件名称。若省略该文件名,ftp 将列出当前目录下的所有文件。ls 与 dir 的不用在于 ls 上给出短格式的列表,而 dir 则给出长格式的列表,包括了文件的大小及修改日期等信息。

cd< directory>

是用于在目录间移动的命令,< directory> 指定了要去的目录名称。作用与用法与 Linux 或 MS-DOS 下的 cd 命令相同。cdup 命令用于从当前目录移至其上层目录。help< command> 命令用于查找一个命令的用法。< command> 指定了该命令的名称。如果该项缺省的话,ftp 将列出 ftp 中的所有命令的名称。用以上几个命令,用户就能在一个站点的文件目录中寻找所要的文件以便进行下载。

B.4 下 载 文 件

在进行文件下载之前,要作完以下几件事:

1. 打开 hash mark 显示开关。hash mark 显示是在文件的传输过程中给出了完成的传输量的信息。一般来说,一个 hash mark 是一个井字号(#),每传输 1024(或 8192)个字节就是显示一个这样的符号。用命令 hash 来打开这个开关。

2. 确定要下载的文件类型。对于 FTP 程序而言,文件可分成二类:二进制代码和文本文件。要下载的文件大多为二进制码类型,包括程序,压缩后的文件,档案文件等。当然也有一些文本类型的文件。为什么要确定文件类型呢?这是因为在某些系统,如 MS-DOS 系统,文本文件中的一些字符需要进行转换后才是可读的。而二进制码文件的传输则不需要任何转换,直接按字节进行传输。命令 ascii 和 bin 分别设置了文本类型和二进制码类型文件传输模式。如果不确定文件是何种类型时应选用二进制模式进行传输。一旦你用

文本模式传输二进制码类型文件,文件会被破坏而无法使用。这是使用 FTP 最常见的错误。

3. 确定本地目录位置。所谓本地目录就是用户打算放置下载的文件的位置。cd 命令是用于确定远程目录位置,lcd 命令用于确定本地目录位置。前面的字母 l 是英文中 local 的简写,即本地的意思。

作完以上几件事之后就可以开始下载文件了。命令

get< remote-name> < local-name>

用于下载操作。这里< remote-name> 是远程系统上的文件名,local-name 是本地机器上的文件名。本地文件名是任选项,如果缺省的话,则与远程文件同名。如果本地已经有一同名文件存在,那么就需要给出本地文件名,否则的话原有文件在下载操作完成之后就已经被刚下载的远程文件所取代。

B.5 退出 FTP

要结束 FTP 工作,只要用 quit 命令就能实现。close 命令用于关闭与一个站点的连接,但并不退出 FTP,用 open 命令可以再建立与另一个站点的连接。

B.6 使用 ftpmail

ftpmail 是系统为用户提供的一种经 Internet 上的电子邮件方式获取 FTP 站点上档案文件的服务。如果用户的系统没有直接与 Internet 相连但能够间接地往 Internet 上发送或接收电子邮件,那么可以使用 ftpmail 来获取 FTP 站点上的文件。当然,ftpmail 会很慢,特别是当文件很大的时候。还有一点要注意的是,在使用 ftpmail 下载大量的文件之前,先确定一下你的电子邮箱是否能应付大量的来函。有不少系统对来函的数量有限额,如果一个用户收到来函超过限额的话,这个用户的账号将会被取消。

sunsite.unc.edu 是 Linux 软件的主要存放站点也是 ftpmail 的服务器站点。只要送一条内容为

help

的电子邮件到

ftpmail@sunsite.unc.edu

你将会收到一份关于 ftpmail 命令表和简单的使用说明的回函。如果你往上述地址发一封内容为

open sunsite.unc.edu

cd /pub/Linux

dir

quit

的电子邮件,就能收到一份在 sunsite.unc.edu 上/pub/Linux 目录下的 Linux 文件清单的回函。当然 ftpmail 并不限于从 sunsite.unc.edu 上获取文件,它可以用来连接任何

FTP 档案站点。

B.7 Linux 的 FTP 站点表

表 B1 列出了存放 Linux 软件的部分比较知名的 FTP 站点。

表 B1 Linux FTP 站点

站点名	IP 地址	文件目录
tsx-11.mit.edu	18.172.1.2	/pub/linux
sunsite.unc.edu	152.2.22.81	/pub/Linux
nic.funet.fi	128.214.6.100	/pub/OS/Linux
ftp.mcc.ac.uk	130.88.200.7	/pub/linux
fgb1.fgb.mw.tu-muenchen.de	129.187.200.1	/pub/linux
ftp.informatik.tu-muenchen.de	131.159.0.110	/pub/Linux
ftp.dfv.rwth-aachen.de	137.226.4.105	/pub/linux
ftp.informatik.rwth-aachen.de	137.226.112.172	/pub/Linux
ftp.ibp.fr	132.227.60.2	/pub/linux
kirk.bu.oz.au	131.244.1.1	/pub/OS/Linux
ftp.uu.net	137.39.1.9	/systems/unix/linux
wuarchive.wustl.edu	128.252.135.4	/systems/linux
ftp.win.tue.n	l131.155.70.100	/pub/linux
ftp.ibr.cs.tu-bs.de	134.169.34.15	/pub/os/linux
ftp.denet.dk	129.142.6.74	/pub/OS/linux

其中 tsx-11.mit.edu, sunsite.unc.edu 和 nic.funet.fi 是 Linux 软件的主站点,大部分新软件都上载到这几个站点。其它站点大多是这几个站点的内容的“镜象”,即它们的内容是跟随几个主站点变化的。

附录 C　BBS　目　录

这里所列的是带有 Linux 软件的 BBS。Linux 用户不仅仅是 Internet 上活跃的一群。有人估计 Linux 的大多数用户并不直接与 Internet 连接。因此, BBS 仍然是一个为世界范围内的 Linux 用户提供软件和其他支持的重要途经。

C.1　美　　国

Citrus Grove Public Access, 916-381-5822. ZyXEL 16.8/14.4 Sacramento, CA.
Internet: citrus.sac.ca.us
Higher Powered BBS, 408-737-7040. ? CA. RIME -> HIGHER
hip-hop, 408-773-0768. 19.2k Sunnyvale, CA. USENET access
hip-hop, 408-773-0768. 38.4k Sunnyvale, CA.
UNIX Online, 707-765-4631. 9600 Petaluma, CA. USENET access
The Outer Rim, 805-252-6342. Santa Clarita, CA.
Programmer's Exchange, 818-444-3507. El Monte, CA. Fidonet
Programmer's Exchange, 818-579-9711. El Monte, CA.
Micro Oasis, 510-895-5985. 14.4k San Leandro, CA.
Test Engineering, 916-928-0504. Sacramento, CA.
Slut Club, 813-975-2603. USR/DS 16.8k HST/14.4K Tampa, FL. Fidonet 1: 377/42
Lost City Atlantis, 904-727-9334. 14.4k Jacksonville, FL. FidoNet
Aquired Knowledge, 305-720-3669. 14.4k v.32bis Ft. Lauderdale, FL.
Internet, UUCP
The Computer Mechanic, 813-544-9345. 14.4k v.32bis St. Petersburg, FL.
Fidonet, Sailnet, MXBBSnet
AVSync, 404-320-6202. Atlanta, GA.
Information Overload, 404-471-1549. 19.2k ZyXEL Atlanta, GA. Fidonet
1: 133/308
Atlanta Radio Club, 404-850-0546. 9600 Atlanta, GA.
Rebel BBS, 208-887-3937. 9600 Boise, ID.
Rocky Mountain HUB, 208-232-3405. 38.4k Pocatello, ID. Fionet, SLNet, CinemaNet
EchoMania, 618-233-1659. 14.4k HST Belleville, IL. Fidonet 1: 2250/1, f'req LINUX
UNIX USER, 708-879-8633. 14.4k Batavia, IL. USENET, Internet mail

PBS BBS, 309-663-7675. 2400 Bloomington, IL.

Third World, 217-356-9512. 9600 v.32 IL.

Digital Underground, 812-941-9427. 14.4k v.32bis IN. USENET

The OA Southern Star, 504-885-5928. New Orleans, LA. Fidonet 1: 396/1

Channel One, 617-354-8873. Boston, MA. RIME -> CHANNEL

VWIS Linux Support BBS, 508-793-1570. 9600 Worcester, MA.

WayStar BBS, 508-481-7147. 14.4k V.32bis USR/HST Marlborough, MA. Fidonet 1: 333/14

WayStar BBS, 508-481-7293. 14.4k V.32bis USR/HST Marlborough, MA. Fidonet 1: 333/15

WayStar BBS, 508-480-8371. 9600 V.32bis or 14.4k USR/HST Marlborough, MA. Fidonet 1: 333/16

Programmer's Center, 301-596-1180. 9600 Columbia, MD. RIME

Brodmann's Place, 301-843-5732. 14.4k Waldorf, MD. RIME -> BRODMANN, Fidonet

Main Frame, 301-654-2554. 9600 Gaithersburg, MD. RIME -> MAINFRAME

1 Zero Cybernet BBS, 301-589-4064. MD.

WaterDeep BBS, 410-614-2190. 9600 v.32 Baltimore, MD.

Harbor Heights BBS, 207-663-0391. 14.4k Boothbay Harbor, ME.

Part-Time BBS, 612-544-5552. 14.4k v.32bis Plymouth, MN.

The Sole Survivor, 314-846-2702. 14.4k v.32bis St. Louis, MO. WWIVnet, WWIVlink, etc

MAC's Place, 919-891-1111. 16.8k, DS modem Dunn, NC. RIME -> MAC

Digital Designs, 919-423-4216. 14.4k, 2400 Hope Mills, NC.

Flite Line, 402-421-2434. Lincoln, NE. RIME -> FLITE, DS modem

Legend, 402-438-2433. Lincoln, NE. DS modem

MegaByte Mansion, 402-551-8681. 14.4 V,32bis Omaha, NE.

Mycroft QNX, 201-858-3429. 14.4k NJ.

Steve Leon's, 201-886-8041. 14.4k Cliffside Park, NJ.

Dwight-Englewood BBS, 201-569-3543. 9600 v.42 Englewood, NJ. USENET

The Mothership Cnection, 908-940-1012. 38.4k Franklin Park, NJ.

The Laboratory, 212-927-4980. 16.8k HST, 14.4k v.32bis NY. FidoNet 1: 278/707

Valhalla, 516-321-6819. 14.4k HST v.32 Babylon, NY. Fidonet (1: 107/255), UseNet (die.linet.org)

Intermittent Connection, 503-344-9838. 14.4k HST v.32bis Eugene, OR. 1: 152/35

Horizon Systems, 216-899-1086. USR v.32 Westlake, OH.

Horizon Systems, 216-899-1293. 2400 Westlake, OH.

Centre Programmers Unit, 814-353-0566. 14.4k V.32bis/HST Bellefonte, PA.

Allentown Technical, 215-432-5699. 9600 v.32/v.42bis Allentown, PA. WWIVNet 2578

Tactical-Operations, 814-861-7637. 14.4k V32bis/V42bis State College, PA. Fidonet 1: 129/226, tac_ ops.UUCP

North Shore BBS, 713-251-9757. Houston, TX.

The Annex, 512-575-1188. 9600 HST TX. Fidonet 1: 3802/217

The Annex, 512-575-0667. 2400 TX. Fidonet 1: 3802/216

Walt Fairs, 713-947-9866. Houston, TX. FidoNet 1: 106/18

CyberVille, 817-249-6261. 9600 TX. FidoNet 1: 130/78

splat-ooh, 512-578-2720. 14.4k Victoria, TX.

splat-ooh, 512-578-5436. 14.4k Victoria, TX.

alaree, 512-575-5554. 14.4k Victoria, TX.

Ronin BBS, 214-938-2840. 14.4 HST/DS Waxahachie (Dallas), TX. RIME, Intelec, Smartnet, etc.

VTBBS, 703-231-7498. Blacksburg, VA.

MBT, 703-953-0640. Blacksburg, VA.

NOVA, 703-323-3321. 9600 Annandale, VA. Fidonet 1: 109/305

Rem-Jem, 703-503-9410. 9600 Fairfax, VA.

Enlightend, 703-370-9528. 14.4k Alexandria, VA. Fidonet 1: 109/615

My UnKnown BBS, 703-690-0669. 14.4k V.32bis VA. Fidonet 1: 109/370

Georgia Peach BBS, 804-727-0399. 14.4k Newport News, VA.

Top Hat BBS, 206-244-9661. 14.4k WA. Fidonet 1: 343/40

victrola.sea.wa.us, 206-838-7456. 19.2k Federal Way, WA. USENET

C.2 其他地区

Galaktische Archive, 0043-2228303804. 16.8 ZYX Wien, Austria. Fidonet 2: 310/77 (19: 00-7: 00)

Linux-Support-Oz, + 61-2-418-8750. v.32bis 14.4k Sydney, NSW, Austrailia. Internet/Usenet, E-Mail/News

500cc Formula 1 BBS, + 61-2-550-4317. V.32bis Sydney, NSW, Australia.

Magic BBS, 403-569-2882. 14.4k HST/Telebit/MNP Calgary, AB, Canada. Internet/Usenet

Logical Solutions, 299-9900 through 9911. 2400 AB, Canada.

Logical Solutions, 299-9912, 299-9913. 14.4k Canada.

Logical Solutions, 299-9914 through 9917. 16.8k v.32bis Canada.

V.A.L.I.S., 403-478-1281. 14.4k v.32bis Edmonton, AB, Canada. USENET
The Windsor Download, (519)-973-9330. v32bis 14.4 ON, Canada.
r-node, 416-249-5366. 2400 Toronto, ON, Canada. USENET
Synapse, 819-246-2344. 819-561-5268 Gatineau, QC, Canada. RIME -> SYNAPSE
Radio Free Nyongwa, 514-524-0829. v.32bis ZyXEL Montreal, QC, Canada.
USENET, Fidonet
DataComm1, + 49.531.132-16. 14.4 HST Braunschweig, NDS, Germany. Fido
2: 240/550, LinuxNet
DataComm2, + 49.531.132-17. 14.4 HST Braunschweig, NDS, Germany. Fido
2: 240/551, LinuxNet
Linux Server/Braukmann, + 49.441.592-963. 16.8 ZYX Oldenburg, NDS, Germany.
Fido 2: 241/2012, LinuxNet
MM's Spielebox, + 49.5323.3515. 14.4 ZYX Clausthal-Zfd., NDS, Germany. Fido
2: 241/3420
MM's Spielebox, + 49.5323.3516. 16.8 ZYX Clausthal-Zfd., NDS, Germany. Fido
2: 241/3421
MM's Spielebox, + 49.5323.3540. 9600 Clausthal-Zfd., NDS, Germany. Fido
2: 241/3422
Bit-Company / J. Bartz, + 49.5323.2539. 16.8 ZYX MO Clausthal-Zfd., NDS,
Germany. Fido 2: 241/3430
Fractal Zone BBS/Maass, + 49.721.863-066. 16.8 ZYX Karlsruhe, BW, Germany.
Fido 2: 241/7462
Hipposoft/M. Junius, + 49.241.875-090. 14.4 HST Aachen, NRW, Germany. Fido
2: 242/6, 4: 30-7,8-23: 30
UB-HOFF/A. Hoffmann, + 49.203.584-155. 19.2 ZYX+ Duisburg, Germany. Fido
2: 242/37
FORMEL-Box, + 49.4191.2846. 16.8 ZYX Kaltenkirchen, SHL, Germany. Fido
2: 242/329, LinuxNet (6: 00-20: 00)
BOX/2, + 49.89.601-96-77. 16.8 ZYX Muenchen, BAY, Germany. Fido 2: 246/147,
info magic: LINUX (22-24,0: 30-2,5-8)
Die Box Passau 2+ 1, + 49.851.555-96. 14.4 V32b Passau, BAY, Germany. Fido
2: 246/200 (8: 00-3: 30)
Die Box Passau Line 1, + 49.851.753-789. 16.8 ZYX Passau, BAY, Germany. Fido
2: 246/2000 (8: 00-3: 30)
Die Box Passau Line 3, + 49.851.732-73. 14.4 HST Passau, BAY, Germany. Fido
2: 246/202 (5: 00-3: 30)
Die Box Passau ISDN, + 49.851.950-464. 38.4/64k V.110/X.75 Passau, BAY,
Germany. Fido 2: 246/201 (8: 00-24: 00,1: 00-3: 30)

Public Domain Kiste, + 49. 30. 686-62-50. 16. 8 ZYX BLN, Germany. Fido 2: 2403/ 17

CS-Port / C. Schmidt, + 49. 30. 491-34-18. 19. 2 Z19 Berlin, BLN, Germany. Fido 2: 2403/ 13

BigBrother / R. Gmelch, + 49. 30. 335-63-28. 16. 8 Z16 Berlin, BLN, Germany. Fido 2: 2403/ 36. 4 (16-23: 00)

CRYSTAL BBS, + 49. 7152. 240-86. 14. 4 HST Leonberg, BW, Germany. Fido 2: 2407/ 3, LinuxNet

Echoblaster BBS # 1, + 49. 7142. 213-92. HST/ V32b Bietigheim, BW, Germany. Fido 2: 2407/ 4, LinuxNet (7-19, 23-01h)

Echoblaster BBS . 7142. 212-35, + 49. 7142. 212-35. V32b Bietigheim, BW, Germany. Fido 2: 2407/ 40, LinuxNet (20h-6h)

LinuxServer / P. Berger, + 49. 711. 756-275. 16. 8 HST Stuttgart, BW, Germany. Fido 2: 2407/ 34, LinuxNet (8: 3-17: 5, 19-2)

Rising Sun BBS, + 49. 7147. 3845. 16. 8 ZYX Sachsenheim, BW, Germany. Fido 2: 2407/ 41, LinuxNet (5: 30-2: 30)

bakunin. north. de, + 49. 421. 870-532. 14. 4 D 2800 Bremen, HB, Germany. kraehe@ bakunin. north. de

oytix. north. de, + 49. 421. 396-57-62. ZYX HB, Germany. mike@ oytix. north. de, login as gast

Fiffis Inn BBS, + 49-89-5701353. 14. 4-19. 2 Munich, Germany. FidoNet 2: 246/ 69, Internet, USENET, LinuxNet

The Field of Inverse Chaos, + 358 0 506 1836. 14. 4k v32bis/ HST Helsinki, Finland. USENET; ichaos. nullnet. fi

Modula BBS, + 33-1 4043 0124. HST 14. 4 v. 32bis Paris, France.

Modula BBS, + 33-1 4530 1248. HST 14. 4 V. 32bis Paris, France.

STDIN BBS, + 33-72375139. v. 32bis Lyon, Laurent Cas, France. FidoNet 2: 323/ 8

Le Lien, + 33-72089879. HST 14. 4/ V32bis Lyon, Pascal Valette, France. FidoNet 2: 323/ 5

Basil, + 33-1-44670844. v. 32bis Paris, Laurent Chemla, France.

Cafard Naum, + 33-51701632. v. 32bis Nantes, Yann Dupont, France.

DUBBS, + 353-1-6789000. 19. 2 ZyXEL Dublin, Ireland. Fidonet 2: 263/ 167

Galway Online, + 353-91-27454. 14. 4k v32b Galway, Ireland. RIME, @ iol. ie

Nemesis' Dungeon, + 353-1-324755 or 326900. 14. 4k v32bis Dublin, Ireland. Fidonet 2: 263/ 150

nonsolosoftware, + 39 51 6140772. v. 32bis, v. 42bis Italy. Fidonet 2: 332/ 407

nonsolosoftware, + 39 51 432904. ZyXEL 19. 2k Italy. Fidonet 2: 332/ 417

Advanced Systems, + 64-9-379-3365. ZyXEL 16. 8k Auckland, New Zealand.

Singet, INTLnet, Fidonet

Thunderball Cave, 472567018. Norway. RIME -> CAVE

DownTown BBS Lelystad, + 31-3200-48852. 14.4k Lelystad, Netherlands. Fido 2: 512/155, UUCP

MUGNET Intl-Cistron BBS, + 31-1720-42580. 38.4k Alphen a/d Rijn, Netherlands. UUCP

The Controversy, (65)560-6040. 14.4k V.32bis/HST Singapore. Fidonet 6: 600/201

Pats System, + 27-12-333-2049. 14.4k v.32bis/HST Pretoria, South Africa. Fidonet 5: 71-1/36

Gunship BBS, + 46-31-693306. 14.4k HST DS Gothenburg Sweden.

Baboon BBS, + 41-62-511726. 19.2k Switzerland. Fido 2: 301/580 and /581

The Purple Tentacle, + 44-734-590990. HST/V32bis Reading, UK. Fidonet 2: 252/305

A6 BBS, + 44-582-460273. 14.4k Herts, UK. Fidonet 2: 440/111

On the Beach, + 444-273-600996. 14.4k/16.8k Brighton, UK. Fidonet 2: 441/122