KH 科海图书
25年·IT技术出版专家

专家技术讲解，370分钟多媒体视频技术录像

Ubuntu

Linux从入门到精通

1DVD 整合Ubuntu 8.10安装光盘 + 370分钟多媒体视频技术录像

郝铃 李晓 编著

- 从系统、驱动、常用软件的安装讲起，快速掌握Linux基础知识
- 轻松向网络管理、Shell、Vi、X-Windows、进程管理进阶
- 深入到Apache、vsftpd、Postfix、Samba、DNS等服务器的配置
- 全面掌握企业所需的Linux应用技能

Ubuntu 8.10安装光盘
370分钟多媒体视频技术录像

科学出版社
北京科海电子出版社
www.khp.com.cn

Ubuntu
Linux从入门到精通

本书可以满足哪些人？

- 已经学过Windows XP，想了解Linux操作系统的初学者
- 想成为网络管理员，却苦于缺乏系统、网络、服务器管理经验的自学者
- 转向Linux嵌入式开发，但对Linux命令、环境配置不熟悉的软件开发人员

本书与一般Linux图书的不同点！

- 传统Linux图书主要以命令行的方式进行介绍，学习起来非常枯燥乏味，而且不便于记忆。本书针对Ubuntu的特点，使用图形界面进行讲解，并辅助命令行对照说明。既方便快速记忆和学习，又能掌握Linux命令行的应用。
- 内容全面，由浅入深。从最基础的系统、驱动、常用软件的安装讲起，深入到Apache、vsftpd、Postfix、Samba、DNS等服务器的配置。
- 为了便于学习，我们把书中的一些重要安装、配置实例制作成多媒体教学视频技术录像，操作步骤一览无遗。

科海图书服务信息
www.khp.com.cn

责任编辑：王海霞
封面设计：林　陶
技术电话：（010）82896445/46转8407
销售电话：（010）82896442 62630320

网上购书：www.huachu.com.cn

上架建议　计算机/Linux/Ubuntu

ISBN 978-7-03-023772-9
9 787030 237729

定价：59.00元（含1DVD价格）

前　言

目前越来越多的计算机用户已开始使用 Windows 系统的替代品，其中 Linux 就获得了不可思议的成功和流行。本书就是面向那些希望迁移到 Linux 系统平台的用户，它将介绍如何建立并使用好一个能够代替 Windows 的操作系统——Ubunbu。

在 Ubuntu 这个操作系统上，用户能找到自己所需要的功能，安装自己需要的应用软件，完成在 Windows 桌面系统上所能够完成的一切工作或娱乐活动。通过 Ubuntu，用户还能够在计算机的硬件配置上，获得一个先进的、没有约束的、安全的操作系统。在这个系统上，同时还能提供各种优质的网络服务，足可满足服务级需求。最让大家心动的是：在 Ubuntu 这个系统之中，所有的软件都是免费的。够吸引人的吧？还犹豫什么，下面就随着本书，进入 Ubuntu 的学习之旅吧！

如何学习 Linux

学习 Ubuntu Linux 之前，先简单地说说 Linux 的学习方法。Linux 的学习比较忌讳的是一开始就用自上而下的方法来学习，如为了修改文件才去学习 Vi 文本编辑器；为了配置 Web 服务器，而去学习 Apache。这样类似头疼医头，脚痛医脚，是一种治标不治本的学习方式。在学习本书之前，作者对大家学习 Ubuntu Linux 提出以下几点建议：

1. 尽可能系统化学习Ubuntu的各个知识点，以形成整体概念。
2. Linux的学习是一个循序渐进的过程，需要经常实践，多做练习，并适当记录学习心得，这是学习历程的见证，同时也是是坚定学习信念的法宝。
3. Linux的最大特点就是其开放性、开源性。Linux的共享资料非常丰富，读者应充分利用这些广泛的资源。如学习中遇到疑难问题，可以去搜索相关信息，或上Linux官方论坛、社区论坛等交流平台询问。

本书特点

本书是为想系统学习 Linux 的初学者准备的，从系统、驱动、常用软件的安装开始讲起，让读者快速掌握 Linux 的基础知识，并轻松向网络管理、Shell、Vi/Vim、X-Window、进程管理进阶，然后深入到 Apache、VSFTPD、Postfix、SAMBA、DNS 等服务器的配置，全面掌握企业所需的 Linux 应用技能。

传统的 Linux 图书主要以命令行的方式进行介绍，学习起来非常枯燥乏味，而且不便于记忆。本书针对 Ubuntu 的特点，使用图形界面进行讲解，并辅助命令行对照说明。既方便快速记忆和学习，又能掌握 Linux 命令行的应用。为了便于读者的实践和练习，本书有针对性地安排了配置实例和练习题。

本书内容

本书总体划分为 4 部分，读者可以顺序阅读本书，也可以根据个人兴趣选读部分章节。每章后面都附有课后习题，读者可以尝试回答，并进行实际操作，以加深对该章节内容的理解。

第 1 部分是 Ubuntu 的概述篇，包括第 1~3 章。第 1 章介绍 Linux/Ubuntu 系统的起源、发展，及发展前景等基本知识，让大家对 Ubuntu 有一个纵向了解。第 2 章系统地介绍 Ubuntu 系统的安装、配置。第 3 章是 Ubuntu 初体验，使读者对 Ubuntu 有一个直观的感觉。

第 2 部分是对 Ubuntu 进阶应用的整体介绍，包括第 4~9 章。主要介绍 Ubuntu 的 X-Window 环境、桌面的个性化设置、Ubuntu 的常用软件。另外还介绍 Shell 环境、最常用的命令行编辑器 Vi/Vim，以及软件包的管理。

第 3 部分介绍 Ubuntu 的日常管理，包括第 10~19 章。通过本篇的学习，读者基本可以实现 Linux 系统的管理工作。具体内容包括文件系统管理、用户管理、系统硬件设备管理、用户磁盘配额管理、Linux 网络与进程管理，还有 Shell 的高级应用。这部分内容相对比较多，也比较复杂，是用户进行日常管理必须掌握的内容。

第 4 部分讲述 Ubuntu 系统服务的管理，包括第 20~25 章。本篇先整体介绍 Ubuntu 系统服务的知识，然后针对 Ubuntu 的服务应用，逐章介绍 Ubuntu 中的 Web 服务器、FTP 服务器、邮件服务器、SAMBA、DNS 服务器等的安装、配置和管理。通过学习，读者可以在 Ubuntu 系统中实现自己的各种服务级应用。

光盘内容

本书光盘中内附带最新的 Ubuntu 8.10 操作系统，供读者学习使用。我们还把书中的一些重要安装、配置实例制作成多媒体教学视频，操作步骤一览无遗，学习起来更轻松。

本书是作者长期教学和 Linux 使用经验的总结，同时也得到了一些网络管理行业朋友的大力帮助，在此深表感谢。在本书的编写过程中，我们力求精益求精，但难免存在一些不足之处，敬请广大读者批评指正。联系方式：pcbook@263.net。

编 者

2008 年 10 月

目录 Contents

Part01 Ubuntu入门

Chapter01

Linux与Ubuntu 3

Chapter02

Ubuntu的安装与配置 27

配套多媒体教学视频索引

Contents 目录

Contents 目录

Contents 目录

Part03 Ubuntu日常管理

Chapter10

文件与目录管理 235

Chapter11

Ubuntu文件的属性与权限 259

Contents 目录

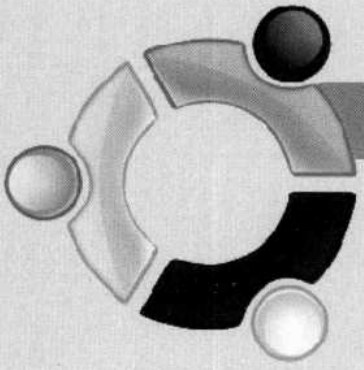
目录

Contents

Contents 目录

目录

Contents



Part 01

Ubuntu 入门

在本篇中，我们着重介绍以下几方面的内容：Linux及Ubuntu的起源、发展及应用前景等基本知识，让大家对Linux有一个直观的了解；对Ubuntu系统发布有个初步的认识，并通过一个具体的Ubuntu安装实例介绍安装的各步骤及在安装过程中的注意事项，让读者对Ubuntu有一个直观的体验。

Chapter01 Linux 与 Ubuntu

欢迎进入 Linux 世界。Linux 是一种应用于 PC（Personal Computer，个人计算机）和工作站的操作系统，而 Ubuntu 是 Linux 的一个分支发行套件。开始进入 Ubuntu 学习之旅之前，让我们对 Linux 及 Ubuntu 有个整体的了解。本章中我们将从 Unix 开始讲述，然后从 Unix 中引出 Linux，再从 Linux 细化到 Debian，最终到 Ubuntu，旨在追根溯源，让大家对 Linux 的发展有个整体全面的了解。另外，本章中我们还将对 Ubuntu 的几个功能应用进行了介绍，便于大家实际使用时体会 Ubuntu 的功用；同时，在第 1 章的结尾，我们简单地介绍了学习 Ubuntu 的一些心得和体会，旨在让大家学习时事半功倍。

1.1 Unix 概述

早在 Linux 诞生之前，就有一个相当稳定且成熟的操作系统即 Unix 存在了，可以说 Unix 是 Linux 的老前辈。Linux 在诞生及后续的发展过程中，继承和发扬了 Unix 系统的许多优点，进而也成为一套稳定而且优良的系统。如果你之前学过 Unix 的话，那么学习 Linux 一定会很轻松。

Unix 是一个强大的多用户、多任务操作系统，支持多种处理器架构。由于 Unix 具有技术成熟、可靠性高、网络和数据库功能强、伸缩性突出和开放性好等特色，可满足各行各业的实际需要，特别能满足企业重要业务的需要，可以说它已经成为主要的工作站平台和重要的企业操作平台。现在的 Unix 已经发展出多个分支，包括 SCO、SUN Solaris、BSD、FREEBSD、MINIX 等，它们都可以称作是 Unix，其中 SCO、SUN Solaris 等是收费的，MINIX 是用于教学目的，FREEBSD 对个人用户是免费的。

1.1.1 Unix 的诞生

Unix 最早由 Ken Thompson、Dennis Ritchie 和 Douglas McIlroy（前两者为图 1.1 中人物）于 1969 年在 AT&T 的贝尔实验室开发。

Unix 的诞生和 Multics（Multiplexed Information and Computing System）是有一定渊源的。Multics 是由麻省理工学院、AT&T 贝尔实验室和通用电气合作进行的操作系统项目，被设计运行在 GE-645 大型主机上。但是由于整个目标过于庞大，糅合了太多的特性，Multics 虽然发布了一些产品，但是性能都很低，最终以失败而告终。

图 1.1 Unix 的两位创建者 Ken Thompson & Dennis Ritchie

AT&T 后来撤出了投入到 Multics 项目的资源，其中一个开发者 Ken Thompson 则继续为 GE-645 开发软件，并最终编写了一个太空旅行游戏。经过实际运行后，他发现游戏速度很慢而且耗费昂贵——每次运行会花费 75 美元。在 Dennis Ritchie（没错，他就是 C 语言之父）的帮助下，Thompson 用 PDP-7 的汇编语言重写了这个游戏，并使其在 DEC PDP-7 上运行起来。这次经历加上 Multics 项目的经验，促使 Thompson 开始了一个 DEC PDP-7 上的新操作系统项目。Thompson 和 Ritchie 领导一组开发者，开发了一个新的多任务操作系统。这个系统包括命令解释器和一些实用程序，这个项目被称为 Unics（Uniplexed Information and Computing System），因为它可以支持多用户并发操作，后来这个名字被改为 Unix。

1.1.2 Unix 的发展

最初，Unix 是用汇编语言编写的，一些应用则是由一种叫做 B 语言的解释型语言和汇编语言混合编写的。B 语言在进行系统编程时不够强大，所以 Thompson 和 Ritchie 对其进行了改造，并于 1971 年共同发明了 C 语言。1973 年，Thompson 和 Ritchie 用 C 语言重写了 Unix。当时，为了实现最高效率，系统程序都是由汇编语言编写，所以 Thompson 和 Ritchie 此举是极具大胆创新和革命意义的；用 C 语言编写的 Unix 代码简洁、紧凑、易移植、易读、易修改，为此后 Unix 的发展奠定了坚实基础。

1974 年，Thompson 和 Ritchie 合作在 ACM 通信上发表了一片关于 Unix 的文章，这是 Unix 第一次出现在贝尔实验室以外。此后，Unix 被政府机关、研究机构、企业和大学所注意，并逐步流行开来。1975 年，Unix 发布了 4、5、6 三个版本。1978 年，已经有大约 600 台计算机在运行 Unix。1979 年，版本 7 发布，这是最后一个广泛发布的研究型 Unix 版本。20 世纪 80 年代相继发布的 8、9、10 版本只授权给了少数大学。此后这个方向上的研究导致了 Plan 9 的出现，这是一个新的分布式操作系统。

1982 年，AT&T 基于版本 7 开发了 Unix System III的第一个版本，这是一个商业版本，仅供出售。为了解决混乱的 Unix 版本情况，AT&T 综合了其他大学和公司开发的各种 Unix，开发了 Unix System V Release 1。这个新的 Unix 商业发行版本不再包含源代码，所以加州大学 Berkeley 分校继续开发 BSD Unix，作为 Unix System III 和 V 的替代选择。BSD 对 Unix 最重要的贡献之一是 TCP/IP。BSD 有 8 个主要的发行版中包含了 TCP/IP：4.1c、4.2、4.3、4.3-Tahoe、4.3-Reno、Net2、4.4 以及 4.4-lite。这些发行版中的 TCP/IP 代码几乎是现在所有系统中 TCP/IP 实现的前辈，包括 AT&T

System V Unix 和 Microsoft Windows。

其他一些公司也开始为其自己的小型机或工作站提供商业版本的 Unix 系统，有些选择 System V 作为基础版本，有些则选择了 BSD。BSD 的一名主要开发者，Bill Joy，在 BSD 基础上开发了 SunOS，并最终创办了 Sun Microsystems。

1991 年，一群 BSD 开发者（Donn Seeley、Mike Karels、Bill Jolitz 和 Trent Hein）离开了加州大学，创办了 BSDI （Berkeley Software Design，Inc）。BSDI 是第一个在 Intel 平台上提供全功能商业 BSD Unix 的厂商。后来 Bill Jolitz 离开了 BSDI，开始了 386BSD 的工作。

在 1994 年以后，BSD Unix 走上了复兴的道路。BSD 的开发也走向了几个不同的方向，并最终导致了 FreeBSD、OpenBSD 和 NetBSD 的出现。386BSD 被认为是 FreeBSD、OpenBSD 和 NetBSD 的先辈。

AT&T 继续为 Unix System V 增加了文件锁定、系统管理、作业控制、流和远程文件系统。1987 年到 1989 年，AT&T 决定将 Xenix（微软开发的一个 x86-pc 上的 Unix 版本）、BSD、SunOS 和 System V 融合为 System V Release 4 （SVR4）。这个新发行版将多种特性融为一体，结束了混乱的竞争局面。1993 年以后，大多数商业 Unix 发行商都基于 SVR4 开发自己的 Unix 变体了。

1.1.3 Unix 现况

Unix System V Release 4 发布后不久，AT&T 就将其所有 Unix 权利出售给了 Novell。Novell 期望以此来对抗微软的 Windows NT，但其核心市场受到了严重伤害，最终 Novell 将 SVR4 的权利出售给了 X/OPEN Consortium，后者是定义 Unix 标准的产业团体。最后 X/OPEN 和 OSF/1 合并，创建了 Open Group。Open Group 定义的多个标准定义着什么是 Unix 以及什么不是 Unix。实际的 Unix 代码则辗转到了 Santa Cruz Operation，这家公司后来出售给了 Caldera Systems。Caldera 原来也出售 Linux 系统，交易完成后，新公司又被重命名为 SCO Group。

根据最近的一个报道，当年负责研发 Unix 与后续维护工作的贝尔实验室 1127 部门已于 2005 年 8 月正式宣告解散。Ken Thompson 已退休，现居加州；Dennis Ritchie 调到别的部门；Douglas McIlroy 在达特茅斯学院担任教授。

1.1.4 Unix 多版本协调：POSIX

POSIX （Portable Operating System Interface，可移植操作系统接口）是基于 Unix 的，这一标准意在期望获得源代码级的软件可移植性。换句话说，作为一个 POSIX 兼容的操作系统编写的程序，应该可以在其他任何 POSIX 操作系统（即使是来自另一个厂商）上编译执行。

POSIX 的诞生和 Unix 的发展是密不可分的。前面提到，Unix 大概于 80 年代向美国各大高校分发 V7 版的源码以做研究。UC Berkeley 在 V7 的基础上开发了 BSD Unix。后来有很多商业厂家意识到 Unix 的价值，也纷纷以贝尔实验室的 System V 或 BSD 为基础来开发自己的 Unix，如 Sun OS，AIX，VMS。由于各厂家对 Unix 的开发各自为政，造成了 Unix 的版本相当混乱，给软件的可移植性带来很大困难，这对 Unix 的发展极为不利。所以，为结束这种局面，从 20 世纪 80 年代开始，POSIX，一个开放的操作系统标准开始制定。

POSIX 标准定义了操作系统应该为应用程序提供的接口：系统调用集。POSIX 是由 IEEE (Institute

of Electrical and Electronic Engineering）开发的，并由 ANSI（American National Standards Institute）和 ISO（International Standards Organisation）标准化。大多数操作系统（包括 Windows NT）都倾向于开发它们的变体版本与 POSIX 兼容。IEEE 制定的 POSIX 标准现在是 Unix 系统的基础部分。

1.2 从 Unix 到 Linux

图 1.2 Linux 的漂亮图标

Linux 是参考 Unix 开发并发展起来的一个类 Unix 操作系统。提到 Linux，相信大家都不陌生，Linux 操作系统是自由软件和开放源代码发展中最著名的例子。Linux 可指 Linux 操作系统，这时，Linux 是指一种计算机操作系统；Linux 也可指操作系统的内核；其实，严格来讲 Linux 这个词本身只表示 Linux 内核，但在实际应用上人们已经习惯了用 Linux 来形容整个基于 Linux 内核，并且搭配了各种人机界面、应用和服务软件的操作系统（也被称为 GNU/Linux）。基于这些组件的 Linux 软件被称为 Linux 发行版。一般来讲，一个 Linux 发行套件包含大量的软件，比如软件开发工具、数据库、Web 服务器（例如 Apache）、X-Window 桌面环境（比如 GNOME 和 KDE）和办公套件（比如 OpenOffice.org）等。如图 1.2 所示为 Linux 的图标。

1.2.1 GNU：非 Unix 项目计划

1983 年，理查·马修·斯托曼（Richard Stallman）创立了 GNU 计划（GNU Project）。这个计划有一个目标是为了发展一个完全免费、自由的 Unix-like 操作系统。GNU 是 GNU's Not Unix 的递归缩写。Richard Stallman 最早是在 net.Unix-wizards 新闻组上公布该消息，并附带一份《GNU 宣言》解释为何发起该计划，其中一个理由就是要"重现当年软件界合作互助的团结精神"。图 1.3 中人物便是 Richard Stallman 博士。

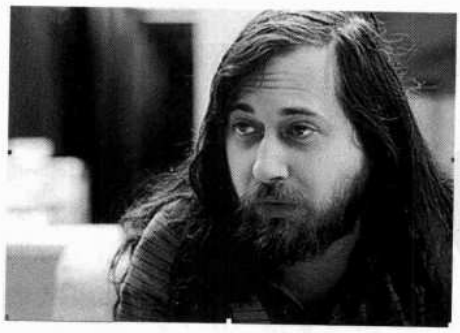


图 1.3 Richard Stallman 博士

Stallman 宣布 GNU 应当发音为 Guh-NOO 以避免与 new 这个单词混淆（注：Gnu 在英文中原意为非洲牛羚，发音与 new 相同）。我们知道 Unix 是一种广泛使用的商业操作系统，因此 GNU 要实现一系列类似 Unix 系统的接口标准，分别开发不同的操作系统部件。GNU 计划曾采用了部分当时已

经可自由使用的软件，如 TeX 排版系统和 X-Window 窗口系统等；同时 GNU 计划也开发了大批其他的自由软件。

为保证 GNU 软件可以自由地“使用、复制、修改和发布”，所有 GNU 软件都遵守一份在禁止其他人添加任何限制的情况下授权所有权利给任何人的协议条款，即 GNU 通用公共许可证（GNU General Public License，GPL），这就是被称为“反版权”（或称 Copyleft）的概念。

1985 年，Richard Stallman 又创立了 FSF（Free Software Foundation，自由软件基金会）来为 GNU 计划提供技术、法律以及财政支持。尽管 GNU 计划大部分时候是由个人自愿无偿贡献，但 FSF 有时还是会聘请程序员帮助编写。当 GNU 计划开始逐渐获得成功时，一些商业公司开始介入开发和技术支持，当中最著名的就是之后被 Red Hat 兼并的 Cygnus Solutions。

自 20 世纪 90 年代发起这个计划以来，GNU 开始大量地生产或收集各种系统所必备的组件，如函数库（Libraries）、编译器（Compilers）、调试工具（Debuggers）、文字编辑器（Text editors）、网页服务器（Web server），以及 Unix 的用户接口（Unix Shell）。另外，1990 年，GNU 计划开始在 Mach microkernel 的架构之上开发系统核心，也就是所谓的 GNU Hurd，但是这个基于 Mach 的设计异常复杂，发展进度也相对缓慢。

20 年来，这个项目不断发展壮大，包含了越来越多的内容。现在，GNU 项目开发的产品，如功能强大的文字编辑器 Emacs、C 语言编译器 GCC 等已经成为其他各种自由发布的类 Unix 产品中的核心角色。唯一依然没有完成的重要组件就是操作系统的内核（称为 HURD）。

1.2.2 Linux 内核的诞生

1990 年，芬兰黑客 Linus Torvalds 为尝试在英特尔 x86 架构上提供自由免费的类 Unix 操作系统内核，自己动手写了一个“类 Minix”的操作系统。整个故事从两个在终端打印 AAAA...和 BBBB...的进程开始，当时最初的内核版本是 0.02。Linus Torvalds 将它发到了 Minix 新闻组，很快就得到了关注。图 1.4 中人物便是 Linux 之父 Linus Torvalds。

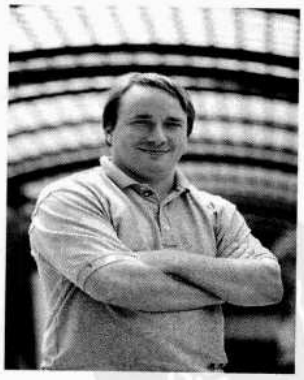


图 1.4 Linux 之父 Linus Torvalds

这里有一份 Linus Torvalds 当时在 Usenet 新闻组 comp.os.minix 所登载的帖子，这份著名的帖子标志着 Linux 计划的正式开始：

Hello, everybody out there using minix,

I'm doing a (free) operation system (just a hobby,

won't be big and professional like GNU) for 386 (486) AT clones.

Linus Torvalds 写的这个系统内核，初名为 Linus' Minix，意为 Linus 的 Minix 内核，后来改名为 Linux。Linus Torvalds 在这种简单的任务切换机制上进行扩展，并在很多热心支持者的帮助下开发和推出了 Linux 的第一个稳定的工作版本。1991 年 11 月，Linux 0.10 版本推出，0.11 版本随后在 1991 年 12 月推出，当时将它发布在 Internet 上，免费供人们使用。当 Linux 非常接近于一种可靠的、稳定的系统时，Linus 决定将 0.13 版本称为 0.95 版本。1994 年 3 月，正式的 Linux 1.0 出现了，这差不多是一种正式的独立宣言。截至那

时为止，它的用户基数已经发展得很大，而且 Linux 的核心开发队伍也建立起来了。

核心的开发和规范一直是由 Linux 社区控制着，版本也是唯一的。实际上，操作系统的内核版本指的是在 Linus 本人领导下的开发小组开发出的系统内核的版本号。自 1994 年 3 月 14 日发布了第一个正式版本 Linux 1.0 以来，每隔一段时间就有新的版本或其修订版公布。

最初，Linux 只能运行在 i386 系统上，实质上是个独立编写的 Unix 内核之克隆，旨在充分利用当时全新的 i386 架构。现如今 Linux 几乎能运行在所有现代架构之上，这要归功于来自世界各地的人们所做的大量开发工作。Linux 内核不仅在技术上占有一席之地，还在意识形态上占有重要位置。很多人相信自由软件的理念，并花费大量时间帮助开源技术，使之臻于完美。正是他们，促成了规范 Internet 发展的众多标准委员会、促成了一些组织如 Mozilla 基金会（负责创建了 Mozilla Firefox）的出现，还促成了无数其他使您受益匪浅的软件项目。开源精神，通常归因于 Linux，这正在深刻影响着世界各地的软件开发者和用户，他们驱使着各个社区朝着共同的目标前进。

另外，Linus 将标准的 GNU 许可协议改称为 Copyleft，以便与 Copyright 相对应。通用的公共许可（GPL）允许用户销售、复制和改变具有 Copyleft 的应用程序。当然这些程序也可以是 Copyright 的，但是你必须允许进一步的销售、复制和对其代码进行改变，同时也必须使他人可以免费得到修改后的源代码。事实证明，GPL 对于 Linux 的成功起到了极大的作用。它开辟了一个十分繁荣的商用 Linux 阶段，还为编程人员提供了一种凝聚力，诱使大家加入这个充满了慈善精神的 Linux 运动。

1992 年，Linux 与其他 GNU 软件结合（GNU 软件构成了这个 POSIX 兼容操作系统 GNU/Linux 的基础），自此完全自由的操作系统正式诞生。今天，GNU/Linux 已经成为发展最为活跃的自由、开放源码的类 Unix 操作系统。

1.2.3 Linux 内核的理解

一个计算机系统是一个硬件和软件的共生体，它们互相依赖，不可分割。计算机的硬件是由外围设备、处理器、内存、硬盘和其他的电子设备组成的计算机发动机。但若没有软件来操作和控制，硬件自身是不能工作的，完成这个控制工作的软件就称为操作系统，在 Linux 的术语中被称为“内核”。内核是操作系统的重要组成部分，它是硬件和软件之间进行通信的桥梁，是提供硬件抽象层、磁盘及文件系统控制、虚拟内存、设备 I/O、多任务等功能的系统软件。

今天，Linux 是一个一体化内核（monolithic kernel）系统。设备驱动程序可以完全访问硬件。Linux 内的设备驱动程序可以方便地以模块化（modularize）的形式进行设置，并在系统运行期间直接装载或卸载。尽管 Linus Torvalds 的初衷不是使 Linux 成为一个可移植的操作系统，今天 Linux 却是全球被最广泛移植的操作系统内核。从掌上电脑 iPa 到巨型电脑 IBM S/390，甚至于微软出品的游戏机 XBOX 都可以看到 Linux 内核的踪迹。Linux 也是 IBM 超级计算机 Blue Gene 的操作系统。

一般的，可以从 Linux 内核版本号来区分系统是 Linux 稳定版还是测试版。以版本 2.4.0 为例，2 代表主版本号，4 代表次版本号，0 代表改动较小的末版本号。在版本号中，序号的第二位为偶数的版本表明这是一个可以使用的稳定版本，如 2.2.5；而序号的第二位为奇数的版本一般有一些新的东西加入，是个不一定很稳定的测试版本，如 2.3.1。这样，稳定版本来源于上一个测试版升级版本，而一个稳定版本发展完全成熟后就不再发展。

在 Linux 内核中，主要内核模块包括：

- 进程管理（process management）
- 定时器（timer）
- 中断管理（interrupt management）
- 内存管理（memory management）
- 模块管理（module management）
- 虚拟文件系统接口（VFS layer）
- 文件系统（file system）
- 设备驱动程序（device driver）
- 进程间通信（inter-process communication）
- 网络管理（network management）
- 系统启动（system init）等

Linux 目前可以在以下结构上运行：

- Acorn：Archimedes，A5000 和 RiscPC 系列
- 康柏：Alpha
- 惠普：PA-RISC
- IA64：英特尔 Itanium 个人计算机
- IBM：S/390 和 AS/400
- 英特尔 80386 及之后的兼容产品：80386，80486 和整个奔腾系列；AMD Athlon，Duron，Thunderbird；Cyrix 系列。对英特尔 8086，8088，80186，80188 和 80280 芯片的支持正在开发中
- Mips 摩托罗拉 68020 及以上
- 新的 Amigas，一些苹果计算机
- PowerPC：所有较新的苹果计算机
- SPARC 和 UltraSPARC：升阳微系统的工作站
- Hitachi SuperH：SEGA Dreamcast
- 索尼公司：PlayStation 2
- 微软公司：Xbox
- ARM 系列

1.2.4 GNU/Linux 操作系统

通过上面两小节的介绍，我们来对 Linux 系统进行一个实质性的归纳理解：从技术上说，Linux 是一个系统内核，一个内核不是一套完整的操作系统，因此，我们通常所说的 Linux 操作系统是一套基于 Linux 内核的完整操作系统，科学来讲应是 GNU/Linux，即采用 Linux 内核的 GNU 操作系统。现在，一个典型的 GNU/Linux 发行版包括 Linux 内核、一些 GNU 程序库和工具、命令行 Shell、图形界面 X-Window 系统和相应的桌面环境，如 KDE 或 GNOME，并包含数千种从办公套件、编译器、文本编辑器到科学工具的应用软件。另外，大多数系统还包括了像提供 GUI 界面的 XFree86 之类的曾经运行于 BSD 的程序。

现在使用 Linux 内核的 GNU 操作系统变种已被广泛使用，尽管这些系统常冠以 Linux，更准确地说，它们应称为 GNU/Linux 系统。绝大多数基于 Linux 内核的操作系统使用了大量的 GNU 软件，包括 Shell 程序、工具、程序库、编译器及工具，还有许多其他程序，例如 Emacs。GNU 项目和自由软件的理念紧密相连，它也是衍生自 GNU 的、诸如 Ubuntu 等项目的关键所在。KDE 和 GNOME 等桌面系统使 Linux 更像是一个 Mac 或 Windows 之类的操作系统，提供完善的图形用户界面，而不同于其他使用命令行界面（Command Line Interface，CLI）的类 Unix 操作系统。正因为如此，GNU 计划的开创者理查德·马修·斯托曼博士提议将 Linux 操作系统改名为 GNU/Linux。

Linux 的基本思想有两点：第一，一切都是文件；第二，每个软件都有确定的用途，同时它们都尽可能被编写得更好。其中，第一点详细来讲就是系统中的所有内容都归结为一个文件，命令、硬件和软件设备、操作系统、进程等对于操作系统内核而言，都被视为拥有各自特性或类型的文件。至于说 Linux 是基于 Unix 的，很大程度上也是因为这两者的基本思想十分相近。

1.2.5 Linux 与 Unix 的区别

大部分 PC 机的 Unix 和 Linux 在实现方面类似，但同时 Linux 和 Unix 的商业版本依然存在许多差别。Linux 支持的硬件范围和商业 Unix 不一样：一般来说，商业 Unix 支持的硬件多一些，Linux 同期支持的硬件要少一些，可是 Linux 支持的硬件一直在扩大。Linux 几乎和商用 Unix 一样稳定；但是 Linux 是免费软件，用户可以从 Internet 上下载，如果上网不方便还可以以很低的价格通过邮购得到 Linux 的磁盘或 CD-ROM，也可以直接从朋友那里得到；商业 Unix 的价值则不然，除了软件本身的价格外，用户还需支付文档、售后支持和质保费，对于较大的机构，这些都很重要，商业 Unix 版本在这方面做得比较好；但是对于很多 PC 用户来说，这些并不重要，这时 Linux 足以满足他们的需求。

也有一些针对 PC 的便宜的 Unix，其中最有名的是 386BSD。在许多方面，386BSD 软件包和 Linux 兼容，但 Linux 更适合用户的需求。386BSD 和 Linux 最显著的区别是：Linux 的开发是开放的，任何志愿者都可以对开发过程作出贡献；而相比之下，386BSD 是由封闭的团队开发的。正是这样，这两种产品存在着严重的概念和设计上的差别：Linux 的目标是从头开始开发一个完整的 Unix 系统；386BSD 的目标则是对现有的 BSD 做些修改，以适合 80386 系统。

总的来说，LINUX 是由广大的黑客、软件开发者遵循 GNU 原则开发的模仿 Unix 的操作系统，两者的根本差异在于 Unix 的核心是有版权的，乱用不得，而 LINUX 的核心是免费的。

1.3 Linux 的发展、应用趋势及当前套件

上一节中提到 Linus 开发了 Linux 内核，并得到很多黑客的帮助，从而使 Linux 越来越完善。本节我们先来简单地了解一下 Linux 内核的发展；另外，GNU/Linux 操作系统在其内核不断完善和发展的过程中，GNU/Linux 也得到了快速成长，并形成了诸如 Red Hat、Debian 等不同发行版本，本节我们也将对这些方面进行介绍。

1.3.1 Linux 的发展

Linux 的历史是和 GNU 紧密联系在一起的。从 1983 年开始的 GNU 计划致力于开发一个自由并且完整的类 Unix 操作系统（包括软件开发工具和各种应用程序），到 1991 年 Linux 内核发布的时候，GNU 几乎已经完成了除系统内核之外的各种必备软件的开发。在 Linus Torvalds 和其他开发人员的努力下，GNU 组件可以运行于 Linux 内核之上。整个内核是基于 GPL（GNU General Public License，GNU 通用公共许可证）的，但是 Linux 内核并不是 GNU 计划的一部分。1994 年 3 月，Linux 1.0 版正式发布，Marc Ewing 成立了 Red Hat 软件公司，成为最著名的 Linux 分销商之一。

早期 Linux 的开机管理程序（Boot Loader）是使用 LILO（Linux Loader），存在着一些令人难以容忍的缺陷，例如无法识别 8GB 以外的硬盘，后来新增 GRUB（GRand Unified Bootloader）克服了这些缺点，具有"动态搜寻核心档案"的功能，可以让用户在开机的时候自行编辑开机设定系统文件，通过 ext2 或 ext3 文件系统载入 Linux Kernel。

Linux 的标志和吉祥物是一只叫做 Tux 的企鹅，标志的由来是这样的：因为 Linus 在澳洲时曾被动物园里的一只企鹅咬了一口，便选择了企鹅作为 Linux 的标志。Linux 的注册商标是 Linus Torvalds 所有的。这是由于在 1996 年，一个名叫 William R. Della Croce 的律师开始向各个 Linux 发布商发信，声明他拥有 Linux 商标的所有权，并且要求各个发布商支付版税，这些发行商集体进行上诉，要求将该注册商标重新分配给 Linus Torvalds。Linus Torvalds 一再声明 Linux 是免费的，他本人可以卖掉，但 Linux 绝不能卖。Linux 发行版的某些版本是不需要安装、只需通过 CD 或者可启动的 USB 存储设备就能使用的版本，它们称为 LiveCD。

总的来说，在开始的时候，Linux 只是个人狂热爱好的一种产物。但是现在，Linux 已经成为一种受到广泛关注和支持的操作系统。包括 IBM 和惠普在内的一些计算机业巨头也开始支持 Linux。很多人认为，和其他的商用 Unix 系统以及微软 Windows 相比，作为自由软件的 Linux 具有成本低、安全性高和更加可信赖的优势。

1.3.2 Linux 的应用及趋势

过去，Linux 主要被用作服务器的操作系统，但因它的廉价、灵活性及其 Unix 背景，使得它很适合更广泛的应用。传统上有以 Linux 为基础的 LAMP（Linux，Apache，MySQL，Perl/PHP/Python 的组合）经典技术组合，提供了包括操作系统、数据库、网站服务器、动态网页的一整套网站架设支持。而面向更大规模级别的领域，如数据库中的 Oracle、DB2、PostgreSQL，以及用于 Apache 的 Tomcat JSP 等都已经在 Linux 上有了很好的应用样本。除了已在开发者群体中广泛流行，它现在亦是网站服务供应商最常使用的平台。

随着 Linux 越来越流行，越来越多的原厂委托制造（OEM）开始在其销售的计算机上预装 Linux，Linux 的用户中也有了普通计算机用户，Linux 系统也开始慢慢抢占桌面计算机操作系统市场。同时，Linux 也是最受欢迎的服务器操作系统之一。Linux 在嵌入式计算机市场上也拥有优势，低成本的特性使 Linux 深受用户欢迎。使用 Linux 的主要成本为移植、培训和学习的费用，早期由于会使用 Linux 的人较少，这方面费用较高，但这方面的费用已经随着 Linux 的日益普及和 Linux 上的软件越来越多、越来越方便而降低。

基于其低廉成本与高度可设定性，Linux 常常被应用于嵌入式系统，例如机顶盒、移动电话及行动装置等。在移动电话上，Linux 已经成为与 Symbian OS、Windows Mobile 系统并列的三大智能手机操作系统之一；而在移动装置上，则成为 Windows CE 与 Palm OS 之外另一个选择。目前流行的 TiVo 数码摄像机使用了经过定制后的 Linux。此外，有不少硬件式的网络防火墙及路由器，例如 LinkSys 的部分产品，其内部都是使用 Linux 来驱动，并采用了操作系统提供的防火墙及路由功能。

采用 Linux 的超级计算机也愈来愈多，最近的 TOP 500 超级计算机列表表明，目前世界上最快速的两组超级计算机都是使用 Linux 作为其操作系统的。而在列表的 500 组系统里，采用 Linux 作为操作系统的占 371 组（即 74.2%），其中的前 10 位中，有 7 组是使用 Linux 的。

2006 年开始发售的 SONY PlayStation 3 也使用 Linux 的操作系统。之前，Sony 也曾为他们的 PlayStation 2 推出过一套名为 PS2 Linux 的 DIY 组件。至于游戏开发商雅达利及 id Software，都曾为其旗下的游戏推出过 Linux 桌面版本。此外，Linux Game Publishing 也曾专门为 Linux 平台撰写游戏，并致力于把其他在 Windows 平台编撰的游戏程序移植至 Linux 平台，及为移植游戏提供使用授权。而一个打算对所有生活在发展中国家孩子提供手提电脑的名为“每孩子皆有一部手提电脑 (OLPC)”的项目，正是使用 Linux 作为默认的操作系统。

总之，Linux 作为较早的源代码开放操作系统，将引领未来软件发展的方向。基于 Linux 开放源码的特性，越来越多大中型企业及政府投入更多的资源来开发 Linux。现今世界上，很多国家逐渐把政府机构内部门的计算机转移到 Linux 上，这种情况还会一直持续。Linux 的广泛使用为政府机构节省了不少经费，也降低了对封闭源码软件潜在的安全性的忧虑。

1.3.3 Linux 发行套件

正如上节所述，我们所说的 Linux 其实应该叫做 GNU/Linux 才对，即 Linux 内核加上 GNU 的外围软件。在 Linux 内核的发展过程中，一些组织或厂家，将 Linux 系统的内核与外围实用程序(Utilities)软件和文件包装起来，并提供一些系统安装界面和系统配置、设定与管理工具，就构成了一种发行版本（distribution），Linux 的发行版本其实就是 Linux 核心再加上外围的实用程序组成的一个大软件包而已。

相对于 Linux 操作系统内核版本，发行版本的版本号随发布者的不同而不同，它与 Linux 系统内核的版本号是相对独立的。因此把 SUSE、RedHat、Ubuntu、Slackware 等直接说成是 Linux 是不确切的，它们是 Linux 的发行版本，更确切地说，应该叫做“以 Linux 为核心的操作系统软件包”。根据 GPL 准则，这些发行版本虽然都源自一个内核，并且都有各自的贡献，但都没有自己的版权。Linux 的各个发行版本（distribution）都是使用 Linus 主导开发并发布的同一个 Linux 内核，因此在内核层不存在什么兼容性问题。每个版本都给人不一样的感觉，只是在发行版本的最外层才有所体现，而绝不是 Linux 本身特别是内核不统一或是不兼容。

20 世纪 90 年代初期，Linux 开始出现的时候仅仅是以源代码形式出现，用户需要在其他操作系统下进行编译才能使用。后来出现了一些正式版本，包括 Red Hat 公司的 Red Hat 系列，如 redhat9、Fedora Core7，RED HAT ES 等；以及社区（community）组织的 Debian 系列、Ubuntu。另外，还有 suse，Gentoo；此外，国内好一点的版本有两个：红旗 Linux 和小红帽 Linux。

总的来说，以 Linux 发行版在 Linux 用户群体中的广泛使用度作为衡量的标准，汇集了 10 大 Linux

发行版，再加上 FreeBSD，总计 11 套系统。如下：

- Ubuntu
- openSUSE
- Fedora
- Debian GNU/Linux
- Mandriva Linux
- PCLinuxOS
- MEPIS Linux
- KNOPPIX
- Slackware Linux
- Gentoo Linux
- FreeBSD

1.4 Ubuntu 的诞生

下面我们介绍本书的主角 Ubuntu。Ubuntu 是 Linux 诸多发行版本中的一个比较成熟且应用非常广泛的发行版本，Ubuntu 的产生离不开 Debian 的发展，更离不开一个人物，下面我们将详细地进行介绍。

1.4.1 Ubuntu 的开发蓝本：Debian

Debian 是一套为计算机设计的自由操作系统（OS）。操作系统是使计算机运行的基本程序和工具的集合。Debian 使用 Linux 核心，但大部分基本工具则来自 GNU 计划，因此我们称为 GNU/Linux。Debian GNU/Linux 不单是个操作系统，它也包含超过 18 733 个软件包，它们是一些已经编译过的软件，并包装成一个容易安装的格式。

Debian 的诞生与发展

德国的君斯坦市的 Ian Murdock（见图 1.5）是 Debian GNU/Linux 发行版的创始人，也是商用 Linux 发行商 Progeny 公司的创始人。他目前就职于 Sun Microsystems，Inc，负责 Sun 公司的操作系统平台发展战略。在加入 Sun 公司之前，Ian Murdock 是 Linux 基金会（Linux Foundation）的首席技术官（CTO），以及 Linux 平台交互标准 LSB（Linux Standard Base）的主席。

Debian 由当时还在美国普渡大学念书的 Ian Murdock 于 1993 年 8 月 16 日首次发表。Ian Murdock 最初把他的系统称为 Debian Linux Release。在定义文件 Debian Manifesto 中，Ian Murdock 宣布将以开源的方式，本着 Linux 及 GNU 的精神发行一套 GNU/Linux 发行版。Debian 的名称是由 Ian Murdock 的女友（现在为其妻子）Debra 和他自己的名字合并而成的。

图 1.5 Debian GNU/Linux 发行版创始人 Ian Murdock

Debian 计划最初发展缓慢，在 1994 年和 1995 年分别发布了 0.9x 系列版本；1.x 版本则在 1996 年发布。1996 年，Bruce Perens 接替 Ian Murdock 成为了 Debian 计划的领导者。同年，另一个开发者 Ean Schuessler 提议 Debian 应在其计划与用户之间建立一份社会契约。经过讨论，Bruce Perens 发表了 Debian 社会契约（Debian Social Contract）及 Debian 自由软件指导方针（Debian Free Software Guidelines），定义了开发 Debian 的基本承诺。如图 1.6 所示为 Debian 的图标。

debian

图 1.6 Debian 的图标

1998 年，在建基于 GNU C 运行期库的 Debian 2.0 发布之前，Bruce Perens 离开了 Debian 的开发工作。Debian 开始选择新的领导者，并发布了另外两个 2.x 版本，其中包含了更多接口和软件包。APT 和第一个非 Linux 接口——Debian GNU/Hurd 的开发也展开。第一个建基于 Debian 的 Linux 发行版 Corel Linux 和 Stormix 的 Storm Linux 在 1999 年开始开发。尽管未能成功开发，这两个发行版成为了 Debian 的 Linux 发行版的先驱。

2000 年下半年，Debian 对数据库和发布的管理作出了重大的改变，它重组了收集软件的过程，并创造了“测试”（testing）版本作为较稳定的对下一个发布的演示。同年，Debian 的开发者开始举办名为 Debconf 的年会，为其开发者和技术用户提供讲座。

Debian 的最新发行版本是 4.0，已于 2007 年 04 月 08 日 正式发行。如果想了解 Debian 4.0 的新增功能、升级注意事项或者是查阅新用户的安装手册，请参考发行信息。

Debian 的版本

Debian 可以算是迄今为止最遵循 GNU 规范的 Linux 系统。Debian 系统分为三个版本分支(branch)：稳定版本（stable）、测试版本（testing）和不稳定版本（unstable）。截至 2005 年 5 月，这三个版本分支分别对应的具体版本为：Woody，Sarge 和 Sid。其中，unstable 为最新的测试版本，其中包括最新的软件包，但是也有相对较多的 bug，适合桌面用户。testing 版本都经过 unstable 中的测试，相对较为稳定，也支持了不少新技术（比如 SMP 等）。而 Woody 一般只用于服务器，上面的软件包大部分都比较过时，但是稳定和安全性都非常高。目前的稳定版本为 Debian etch，目前的测试版本为 Debian lenny，不稳定版本永远为 Debian sid。具体如表 1.1 所示。

表 1.1 Debian 的历史发行版本

代号	发布日期	玩具总动员的对应角色	备注
0.01~0.91	1993 年 8 月—1994 年 1 月		
0.93R5	1995 年 3 月		
0.93R6	1995 年 11 月		
1.1 Buzz	1996 年 6 月 17 日	巴斯光年，电影主角之一的航天员	使用 Linux 内核 2.0
1.2 Rex	1996 年 12 月 12 日	暴龙	
1.3 Bo	1997 年 6 月 2 日	放羊的女孩	
2.0 Hamm	1998 年 7 月 24 日	小猪扑满	
2.1 Slink	1999 年 3 月 9 日	玩具狗	APT 面世
2.2 Potato	2000 年 8 月 15 日	Potato Head 先生	
3.0 Woody	2002 年 7 月 19 日	胡迪，电影主角之一的牛仔	
3.1 Sarge	2005 年 6 月 6 日	绿色塑胶玩具士兵的首领	
4.0 Etch	2007 年 4 月 8 日	玩具黑板	目前的稳定版本
Lenny	未定	望远镜	目前的测试版本
Sid	永远的不稳定版本	隔壁的男孩，玩具终结者 Sid 也是英语 Still In Development 的缩写	

Debian 的突出特点

（1）自由灵活的升级方式

为何有如此多的用户痴迷于 Debian 呢？apt-get/dpkg 是原因之一。dpkg 是 Debian 系列特有的软件包管理工具，它被誉为所有 Linux 软件包管理工具（比如 RPM）中最强大的。配合 apt-get，在 Debian 上安装、升级、删除和管理软件变得异常容易。

由于有这样的包装系统，升级到新的 Debian 版本非常轻松，只需要运行 apt-get update 或者 apt-get dist-upgrade，然后就可以在几分钟内由光盘升级，或者将 apt 指向超过一百五十个 Debian 镜像站点中的一个，接着由网络来升级。

（2）无与伦比的支持

发送到邮件列表的邮件通常会在十五分钟（或更短）之内得到开发人员的免费解答。与典型的电话支持服务相比，开发人员的解答更加专业和直接。

（3）简单方便的安装过程

如果您听说过 Linux 的安装很困难，那是您没有试过最新的 Debian。开发者一直坚持不懈地优化安装过程，您可以通过光盘、DOS、软盘，甚至是网络来安装 Linux。

（4）软件包的高度集成

Debian 能凌驾其他发行版本之上在于其软件包的良好集成度。因为所有的软件都是由同一个团体所包装，如此不仅可以在一个站点找到所有的软件，也可以确信开发者已经解决了所有复杂的集成性问题。他们认为，deb 格式具有某些超越 rpm 格式的优点，正是这种软件包之间的集成性让 Debian 成为更稳定、更强健的系统。

（5）源代码

如果您是一个软件开发人员，将会了解并欣赏 Debian 中附带的数百种开发工具和语言，以及附加于底部系统的数百万行源代码。所有主发行版中所包含的软件都符合 Debian 自由软件指导方针（DFSG）的标准。这意味着您可以直接利用这些代码来学习或研究，或把它们合并到新的自由软件方案中。当然，也有丰富的工具和代码适合在私有的开发方案中使用。

（6）错误跟踪系统

Debian 的错误跟踪系统采取公开的运作模式，不会试图隐瞒软件无法如用户希望般正常运转的事实，用户们可以提交 bug 报告并被通知该 bug 何时和为何被取消了。Debian 错误跟踪系统让 Debian 快速且诚实地回应问题。

（7）稳定性及更快、更容易的内存管理

与其他那些一天要崩溃几次的系统相比，Debian 有许多运行了多年没有重新启动机器的实例，即使有过，也是由于电源故障或硬件升级。

（8）良好的系统安全

Windows 95 根本就没有安全性可言，NT 的表现也非常差，而经过数年的发展以后，GNU/Linux 已经变得十分安全，Debian 也因此而受益。同样，Debian 非常注意在软件发布中快速修复安全的问题（通常没几天就会有修复过的软件包被上传）。

从历史上来看，“越隐蔽越安全”的观念是错误的。因为开放源码，Debian 的安全性会在开放的情况下被评估，因此，可以避免制定出不良的安全模式。许多人并不知道，任何机器都可以看到您在网络上发送的任何信息。Debian 有著名的 GPG（和 PGP）软件，允许邮件在用户之间秘密地被发送。另外，ssh 允许您和其他安装了 ssh 的机器创建安全的连接。

（9）缺乏流行的商业软件

Linux 下确实缺乏某些流行的商业软件，然而，绝大多数还是有替代的软件可用，它们模仿了商业软件中最好的特点，而同时具有作为自由软件的附加价值。

缺乏 Word 或 Excel 之类的办公软件应该不再是个问题，因为 Debian 已经包含了两个办公软件包，并且是完全的自由软件，KOffice 和 GNOME Office。OpenOffice 也将能够在下一个 Debian 发行版中获得。

1.4.2 Ubuntu 的诞生

Ubuntu 由马克·舍特尔沃斯（见图 1.7）创立，其首个版本于 2004 年 10 月 20 日发布，并以 Debian 为开发蓝本。它以每六个月发布一次新版本为目标，使得人们能更频繁地获取新软件，而其开发目的是为了使个人计算机变得简单易用，但它也提供服务器版本。如图 1.8 所示为 Ubuntu 的图标。

图 1.7 Ubuntu 始创者——马克·舍特尔沃斯

图 1.8 Ubuntu 的图标

Ubuntu 的每个新版本都会包含最新版本的 GNOME 桌面环境，并且会在 GNOME 发布新版本后一个月内发行。与以往基于 Debian 的 Linux 发行版，如 MEPIS、Xandros、Linspire、Progeny 与 Libranet 等比较起来，Ubuntu 更接近 Debian 的开发理念，因为其主要使用自由与开源软件，而其他的发行版则会附带很多闭源的插件。

Ubuntu 的软件套件主要是建基于 Debian 的稳定分支：不论是其软件套件格式（deb）还是软件管理与安装系统（Debian Apt/Synaptic）。Ubuntu 会将所有对软件套件的修改及时向 Debian 作出回馈，而不是在发布新版时才宣布这些修改，而事实上，很多 Ubuntu 的开发者均是 Debian 的主要软件套件的维护者，但是 Debian 与 Ubuntu 的软件套件并不一定与对方兼容。换言之，将 Debian 的软件包安装在 Ubuntu 上可能会出现兼容性问题，反之亦然。

Ubuntu 的运作主要依靠 Canonical 有限公司的支持，但也有来自 Linux 社区的热心人士提供协助，Ubuntu 的开发人员多称马克·舍特尔沃斯为 SABDFL（self-appointed benevolent dictator for life，自封的仁慈大君）。而在 2005 年 7 月 8 日，马克·舍特尔沃斯与 Canonical 有限公司宣布成立 Ubuntu 基金会，并对其提供 1 千万美元作为起始营运资金。成立基金会的目的是为了确保将来 Ubuntu 能持续开发并获得支持，但直至 2006 年，此基金会仍未投入运行。马克·舍特尔沃斯形容此基金会是在 Canonical 有限公司出现财务危机时的紧急营运资金。

1.4.3 Ubuntu 的理念

Ubuntu 是一个南非的民族观念，着眼于人们之间的忠诚和联系。该词来自于祖鲁语和科萨语。Ubuntu（发音“oo-BOON-too”－“乌邦图”）被视为非洲人的传统理念，也是建立新南非共和国的基本原则之一，与非洲复兴的理想密切相关。Ubuntu 精神的大意是“待人以仁”（人性对待他人）。另一种翻译可以是：“天下共享的信念，连接起每个人”。“具有 Ubuntu 精神的人心胸开阔，乐于助人，见贤思齐而不忌妒贤能，因为他/她拥有适度的自信，而这源自如下认知：自己乃是属于一个更大的整体，当他人受到伤害或死去时，当他人受到折磨或压迫时，这个整体就会消失。”大

主教 Desmond Tutu 作为一个基于 GNU/Linux 的平台，Ubuntu 操作系统将 Ubuntu 精神带给了软件世界。如图 1.9 所示为 Ubuntu 默认登录界面。

图 1.9 Ubuntu 默认登录界面

Ubuntu 项目完全遵从开源软件开发的原则，并且鼓励人们使用、完善并传播开源软件。也就是说，Ubuntu 目前是并将永远是免费的。然而，这并不仅仅意味着零成本，自由软件的理念是人们应该以所有“对社会有用”的方式自由地使用软件。“自由软件”并不只意味着不需要为其支付费用，它也意味着可以以自己想要的方式使用软件：任何人可以以任意方式下载、修改、修正和使用组成自由软件的代码。因此，除去自由软件常以免费方式提供这一事实外，这种自由也有着技术上的优势：进行程序开发时，就可以使用其他人的成果或以此为基础进行开发。对于非自由软件而言，这点就无法实现，进行程序开发时，人们总得白手起家。基于上述原因，自由软件的开发是迅捷、高效和激动人心的！

1.5 Ubuntu 的发展及版本控制

Ubuntu 问世以来，就在 Distrowatch.com 网站上创造了浏览率最高的佳绩。许多人认为，Ubuntu 成功的主要原因就在于其庞大的用户群，而且，Ubuntu 并不像其他 Linux 发行版，动辄需要用五张光盘才能安装，它只需要一张光盘既可启动（不需要安装）；如果想安装到硬盘上，只需要单击桌面上的安装图标，按照提示完成安装，具有很好的操作性。

1.5.1 Ubuntu 特色

接下来我将从不同的角度来诠释 Ubuntu 的不同特色，这些是 Ubuntu 在众多 Linux 发行版中能够脱颖而出的原因，也是在学习 Ubuntu 前首先要了解的内容。

系统管理

由于 Ubuntu 的开发者与 Debian 和 GNOME 开源社区互相协作，因此其桌面环境采用了 GNOME 的最新版本，并且与 GNOME 项目同步发布。Ubuntu 不仅仅使用与 Debian 相同的 deb -{zh-cn:软件包;zh-tw:软件套件}-格式，还和 Debian 社区有着密切联系，它直接地、实时地向 Debian 社区作出贡献，而不是只在发布时宣布一下。许多 Ubuntu 的开发者也负责为 Debian 的关键软件包进行维护。

Ubuntu 十分注重系统的安全性，它采用 Sudo 工具，所有与系统相关的任务均须使用此指令，并输入密码，这比传统的以输入系统管理员账号进行管理工作有更佳的安全性。

发行理念

Ubuntu 计划强调国际化，以便能为尽可能多的人所用。自 5.04 版本开始，使用万国码（UTF-8 Unicode）作为系统预设编码，使得来自不同国家的使用者可以看到对方的文字，而不会出现乱码。

此外，Ubuntu 的所有发行版本都免费提供。除了光盘映像文件（CD Image）可提供下载外，使用者亦可通过邮寄服务来获取免费的光盘；与其他大型 Linux 操作系统厂商不同，Ubuntu 并不对企业版收取升级订购费（即没有所谓的企业版本，人人所使用的版本皆一样，使用者只有在购买官方技术支持服务时才要付钱）。

Ubuntu 的发行理念强调尽量使用自由软件，并为所有用户提供从某个版本升级到下一个版本的方便的途径。

安装设定

一直以来，Ubuntu 均支持主流的 i386、AMD64 与 PowerPC 平台，因此大多数使用者皆可在其个人计算机上安装相应的 Ubuntu 版本。而在 2006 年 6 月，Ubuntu 新增了对 Sun 计算机的 UltraSPARC 与 UltraSPARC T1 平台的支持，使用者可下载相应版本进行安装。

自初始发行起，Ubuntu 即提供一张安装光盘与一张用来预览的 Live CD。在 Ubuntu 6.06 LTS 发布时，将原来只用作预览的 Live CD 更改为不只可以用来预览，并且可以使用图形接口进行安装的光盘（即 Desktop CD），而原来只提供文字安装接口的安装光盘则保留并改名（即 Alternate install CD）。

在硬件要求上，Ubuntu 需要 256MB 的内存，并需要 3GB 的硬盘空间用来安装。在文件系统格式方面，Ubuntu 默认采用 ext3 文件系统格式，但也可选择其他文件系统格式。而在读取微软窗口分区方面，它可以自由读取和写入 FAT32 文件系统格式的分区，但是对 NTFS 文件系统格式的分区则只可进行读取，而不能对其写入数据（但也可通过安装 ntfs-3g 软件包实现读写 NTFS 分区的功能）。而在 Ubuntu 里，可以通过 Samba 这个软件来与其他操作系统进行信息和文件交换，功能类似 Windows 平台上的网络邻居。

1.5.2 Ubuntu 软件组件

Ubuntu 的软件管理系统与 Debian 的相同，都使用 apt-get 这个指令，且都有图形界面的 Synaptic（新立得软件包管理器），而 Ubuntu 将所有软件分为 4 类，称为“组件（component）”，以反映不同的许可证和可用的支持级别。

- main 组件只包含符合 Ubuntu 许可证要求并可以从 Ubuntu 团队中获得支持的软件包，它力图使日常使用 Linux 系统时所需的任何东西都包括在内。这个组件内的包可以确保得到技术支持和及时的安全升级。在此组件内的软件必定是符合 Ubuntu 版权要求（Ubuntu license requirements）的自由软件，而 Ubuntu 版权要求大致上与 Debian 自由软件指导方针（Debian Free Software Guidelines）相同。
- restricted 组件包含了由于其重要性而被 Ubuntu 开发者支持的软件，但是它们并不具有合适的自由许可证，因此不能列入 main，其中包括仅能以二进制形式获得的显卡驱动程序。因为 Ubuntu 开发者无法获得源代码，其支持的水平与 main 相比是有限的。
- universe（“社区维护”）组件里包含的软件范围广泛，它们也许是受限于许可证，不为 Ubuntu 团队所支持。这样，用户可以使用 Ubuntu 的软件包管理系统安装各种程序，同时又与 main 和 restricted 中被支持的软件包相隔离。
- 最后是 multiverse（“非自由”）组件，其中包括了不符合自由软件要求而且不被支持的软件包。

在通常情况下，来自 main（官方支持）的软件会被安装，以满足大多数计算机用户的基本要求，而同样被安装的还有来自 restricted（版权限制）的软件，它是对系统可用性具有重要作用的软件包。

而因为 Ubuntu 的新版本在发行后，该版本的套件库便会被冻结，只提供安全性更新，因此官方推出了一个名为 Ubuntu Backports 的后续支持计划，让使用者可以获得最新版本的软件。

简而言之，在 Ubuntu 中，安装软件的操作大多是通过 apt-get 指令或 Synaptic 工具来完成的，在其中可找到所有官方提供的软件（不一定被支持），这与微软视窗操作系统的情况大不相同。在视窗操作系统中，用户要安装软件，便要购买该软件的安装文件或执行文件（文件扩展名为.exe），或在网络上下载，且要逐一寻找；而在 Ubuntu，则可一次完成大量软件安装，因为不论 apt-get 指令或 Synaptic，均可一次性搜寻并安装大量软件。

默认套件

以下的列表包含了在 Ubuntu 桌面中默认安装的部分软件，这些软件（如 GIMP、 OpenOffice.org 和 Firefox）是自由桌面的标准软件。

- GNOME：桌面环境与附属应用程序。
- GIMP：绘图程序。
- Firefox：网页浏览器（Web Browser）。
- Gaim：即时通信软件。
- Evolution：电子邮件（E-Mail）与个人资讯管理软件（PIM）。
- OpenOffice.org：办公软件（Office Software）。
- SCIM 输入法平台：支持东亚三国（中、日、韩）的文字输入，并有多种输入法供选择（只有在安装系统时选择东亚三国语系，才会在默认情况下被安装）。
- Synaptic：新立得软件包管理器。
- Totem：多媒体播放器。
- Rhythmbox：音乐播放器。

众多其他自由软件也可以通过 Synaptic 或 apt-get 命令连接至套件库下载并进行安装。默认情况下，Ubuntu 并没有安装防毒软件，这是因为 Linux 受到病毒的威胁不大，但用户可自行安装 ClamAV 这套防毒软件，其主要是用来扫描和清除可能感染微软视窗操作系统的病毒，这在作为电子邮件服务器的计算机上作用较大。而 Ubuntu 则默认安装了 iptables 这套防火墙软件，但却没有提供相关的图形设置接口，用户可自行安装 firestarter 这套图形接口的防火墙设定程序来进行设置。

私有版权软件的采用

Ubuntu 为第三方软件设立了认证程序。虽然其主要采用自由软件，但也接纳部分私有版权软件，只要该私有版权软件可以自由散发，Ubuntu 便会将其放于 multiverse 组件里。

不随 Ubuntu 发行的软件包括：

- 破解 DVD 加密的解密软件 DeCSS。
- 多媒体编码与解码程序库，如 Windows Media。
- 部分广受欢迎的以私有版权形式发布的浏览器插件：如 Adobe（合并前为 Macromedia）出版的 Shockwave（其没有 Linux 版本）与 Flash——其授权（EULA）禁止 Linux 发行版将其附于光盘或映像文件内发布，但 Ubuntu 将其置于 multiverse 的套件包（flashplugin-nonfree）内，其会自动从 Adobe 的网站下载 Linux 版 Flash 插件，并进行安装。

1.5.3 Ubuntu 的发布周期及衍生版本

Ubuntu 的发布周期

Ubuntu 每 6 个月发布一个新版本，且每个版本都有代号和版本号。如图 1.10 所示为 Ubuntu 发行套件的图标。

图 1.10 Ubuntu 发行套件的图标

版本号基于发布日期，例如第一个版本 4.10，代表是在 2004 年 10 月发行的。表 1.2 列出了 Ubuntu 的历史发行版本。

表 1.2 Ubuntu 的历史发行版本

版本	发布日期	代号
4.10	2004 年 10 月 20 日	Warty Warthog
5.04	2005 年 4 月 8 日	Hoary Hedgehog
5.10	2005 年 10 月 13 日	Breezy Badger
6.06 LTS	2006 年 6 月 1 日	Dapper Drake
6.10	2006 年 10 月 26 日	Edgy Eft
7.04	2007 年 4 月 19 日	Feisty Fawn

（续表）

版本	发布日期	代号
7.10	2007 年 10 月 18 日	Gutsy Gibbon
8.04	2008 年 4 月 21 日	Hardy Heron
8.10	2008 年 10 月 30 日	Intrepid Ibex

另外还有一个代号为 Grumpy Groundhog 的分支，它直接从 Ubuntu 套件库里的软件版本控制系统中取出软件的源代码，以进行永久性的测试和开发，因此它为不稳定的分支，且不会对公众开放。

正式衍生版本

目前，Ubuntu 正式支持的衍生版本包括：

（1）KUbuntu

KUbuntu 使用和 Ubuntu 一样的软件库，但不采用 GNOME，而使用更为美观的 KDE 为其默认桌面环境，其 logo 见图 1.11。

图 1.11 KUbuntu 的 logo

（2）EdUbuntu

EdUbuntu 是 Ubuntu 的教育发行版。这是为了使教育工作者可以在一小时内设计计算机教室或建立网上学习环境，并且可即时控制该环境而不用在家学习，其 logo 见图 1.12。

图 1.12 EdUbuntu 的 logo

（3）XUbuntu

XUbuntu 属于轻量级的发行版，它使用 Xfce4 作为桌面环境，采用与 Ubuntu 一样的软件库，其 logo 见图 1.13。

图 1.13 XUbuntu 的 logo

(4) Ubuntu Server Edition

Ubuntu Server Edition 提供了服务器的应用程序，如一个电子邮件服务器、一个 LAMP 网页服务器平台、DNS 设定工具、文件服务器与数据库管理。与原来的桌面版本相比，服务器版的镜像安装文件更小，并且其对硬件的要求更低。若要运行服务器版，只需要有 500MB 的硬盘空间与 64MB 的内存便可。然而它没有提供任何桌面环境，用户在默认环境下只能使用文字接口。

另外，马克·舍特尔沃斯也承诺将制作 Ubuntu-libre 发行版，其中只使用自由软件基金会认证过的自由软件。

非正式衍生版本

目前，Ubuntu 非正式支持的衍生版本包括：

(1) nUbuntu：专注于安全工具的版本。
(2) Ubuntu Lite：为旧计算机而设的版本。
(3) zUbuntu：为 IBM zSeries 主机移植的版本。
(4) Ebuntu：基于 Enlightenment 0.17 桌面环境并附有窗口管理器的 Ubuntu 修改版。
(5) Fluxbuntu：基于 Fluxbox 桌面环境的修改版。
(6) Gnoppix：这是基于 Ubuntu Live CD 而研制的以 GNOME 为默认桌面环境的 Live CD。
(7) DUbuntu：这是中国内地 Ubuntu 爱好者改进的 Ubuntu 版本，能更好地支持中文，并且添加了更多软件的 Live CD。

1.6 如何学习 Ubuntu

1.6.1 Ubuntu 主要应用

讨论如何学习 Linux 之前，我们要就 Linux 目前的应用情况来说明一下，让读者确定自己需要什么样的学习方式。

桌面计算机

所谓桌面计算机，就是一般用户在显示器前面工作的时候常见的操作系统，我们称之为 Desktop 系统，就是简称的桌面计算机。说到桌面计算机，就不能不提一下 X-Window System。至于我们所看到的美观的窗口画面，则是使用 X Server 提供的显示相关硬件的功能，来达到图形显示的窗口管理员（Window Manager，WM）所发挥的能力。这也就是说，WM 是挂在 X Server 上面来运行的一套显示窗口接口的软件，例如我们常见的 KDE、GNOME 等都是 WM。

那么，桌面计算机平时都在干些什么？简单地说，桌面计算机的日常工作大致包括：

- 上网浏览
- 文字处理
- 网络接口之公文处理

- 办公软件（Office Software）处理数据
- 收发电子邮件

在这些工作中，Linux 有美丽的图形接口 X-Window System 提供良好的图形用户接口（GUI），此外，目前发展中的 WM 也都具有中文版本！至于 Office 软件，则有类似 Open Office 的软件支持，所以，基本上在桌型计算机的使用中，Linux 已经足以应付大部分上班族的工作了。

工作站计算机

工作站计算机与桌面计算机不太一样的地方，在于工作站通常要应付比较重要的应用，例如工程流体力学的数值模型运算、多媒体的特效功能处理、软件开发者的工作平台等。Linux 有强大的运算能力，以及支持度相当广泛的 GCC 编译软件，因此在工作站当中也是相当好的一个操作系统选择。

网络服务器

承袭了 Unix 的良好传统，Linux 的网络功能非常的强大。此外，由于 GNU 计划，Linux 的服务器软件几乎都是免费的，因此，作为一部网络服务器，例如 WWW，Mail Server，File Server，FTP Server 等，Linux 绝对是上上之选。

嵌入式系统

近年来，电子相关产业蓬勃发展，其中，小型微计算机的发展尤为重要，例如家电产品、PDA，以及其他微型的计算机配件。这些计算机配件通常是直接嵌入产品当中的，例如，PDA 本身就是一个小型的计算机操作系统，这些系统就称为嵌入式系统。而要让这些嵌入式系统能够运作，自然就需要开发一套简单的操作系统，这个时候，可修改核心、让功能变简单的 Linux 则是很好的选择。因此，近年来有相当多的嵌入式系统选择 Linux 作为开发的平台。

1.6.2 如何学习 Ubuntu

扎扎实实从基础学起

不论学什么系统，“从头学起”是很重要的。这里建议大家：

（1）先理解一下基础的硬件及网络基础知识，硬件如磁盘、声卡、显卡等；网络知识如 IP 概念、路由概念、TCP/IP 等。

（2）先了解一下 Linux 的基础知识，如用户、群组的概念，权限的概念，程序的定义等。

（3）至少学会使用一种文本编辑器，如最好会用通用版的 Vi。实际操作 Linux 时，一定要学习 Shell，所以最好也能够了解 Shell Scripts。

在实践中探索进步

要学习 Linux，千万不能只限于书本上的理解，一定要为自己创造一个学习 Linux 的环境，即在计算机上装一个 Linux 或 Unix。其中 Ubuntu 就是一个特别适合 Linux 初学者的版本，建议大家使用。Ubuntu 安装方便，且网上学习资源丰富，强大的桌面系统、稳定的网络性能，一定能给你带来不一

样的感受。

安装过程中，一定要亲自动手把 Linux 装到硬盘上。对于现在的 Ubuntu 版本来说，其实跟安装 Windows XP 一样简单。但是从现在开始，请不要以 Windows 的工作方式来考虑 Linux 问题，应该尝试挖掘 Linux 身上的“天才 Unix”的气质。当然，Linux 的学习不能局限于只会安装上或只会在桌面上进行简单的操作，建议大家至少掌握 50 个以上的常用命令。

逐步积累，长期学习，最终精通

Linux 的普通操作与真正的系统管理是不能相提并论的。Linux 的学习是一个长期的过程，而要完全掌握它，并能熟练地进行系统管理，则需要多方面努力，特别要注意下面三个方面。

英语 其实计算机语言就是英文和字符，所谓多国语言只是外部包装，所以我们必须能够无障碍地阅读大量的英文技术文件，在搜索引擎找到的英文网站中熟练地检索。当然，若能很好地用英文直接交流，那么对 Linux 的学习和理解就能快很多。

Shell 它是内核与用户界面之间交流的通道，Shell 下的小脚本有点类似于 Windows 下的.bat，但 Shell 比.bat 功能强大得多。Shell 不只是解释命令，更是一种编程语言，有时候几百行的 C 语言用 Shell 几十行就能完成工作，这是因为 Shell 的工作方式是建立在系统已有的众多应用程序之上。

掌握 Linux 系统服务、系统管理的知识 学习 Apache，Ssh，Sendmail/Qmail，Proftp/Vsftp，Samba，Squid，MySQL/PostgreSQL/Oracle，Bind 等各种应用服务器的构架及电子商务的应用；熟悉 TCP/IP 协议簇；学习诸如 Apache+Php+Proftp+Mysql+Quota 的实现以及大型局域网、分布式集群等各种企业级应用解决方案；熟悉多用户管理、数据库管理、文件系统、逻辑存储管理、日志分析、备份与灾难数据恢复系统补丁、内核升级，以及在此基础上的防火墙构架等以保障系统安全为目的的各种系统管理技能。另外，对 Linux 内核原理要有一定的了解。

1.6.3 获取 Ubuntu 的免费学习资料

在 Linux/Ubuntu 的学习过程中，要掌握这个操作系统，建议大家除了要系统地学习本教程的各个知识点，做到融会贯通外，还要根据学习中遇到的实际问题，多上网检索查阅资料。下面是一些对学习 Linux/Ubuntu 非常有帮助的网站。

www.Linuxforum.net	国内最高水平的 GNU 站点
www.chinaUnix.net	Unix 系统管理的知识论坛
www.Linuxeden.com	下载软件不错，资源比较丰富
http://www.gnu.org/	GNU 官方网站
http://www.debian.org/intl/zh/	Debian 中文网站
http://www.ubuntu.com/	Ubuntu 官方网站
http://www.ubuntu.org.cn/	Ubuntu 中文官网及论坛
http://wiki.ubuntu.org.cn	中文 wiki（快速入门）

1.7 课后练习

1. 简述 Unix 的发展及演变过程。
2. 简述 Linux 的产生过程及 Linux 系统和 GNU、Linux 内核之间的关系。
3. Debian 和 Ubuntu 有什么关系？Ubuntu 的含义是什么？Ubuntu 的开发理念是什么？
4. Ubuntu 的发布周期为多长？当前的最新版本是什么？
5. 简述 Linux 在当前的主要应用。
6. 请根据自己的基础和目标制定 Linux 的学习计划。

Chapter02 Ubuntu 的安装与配置

Ubuntu 可以运行在主流 32 位 PC、AMD 64 位 PC 与 PowerPC 平台之上。在实际安装 Ubuntu 之前，须先做些准备工作，同时，因为 Ubuntu 对于硬件的配置要求较高，读者也须先了解一些硬件的概念，这样才比较容易了解 Ubuntu 的安装要求。举例来说，从安装 Ubuntu 的用途看，若是一般的桌面类型的系统，那么需要较好的显卡，以更好地运行 GNOME 或 KDE 桌面环境；如果是服务器类型的系统，那么最好有较大的内存和硬盘空间；当然，如果是用于学习 Ubuntu 的话，那就把所有的套件都安装上，以方便我们的学习。

2.1 计算机硬件准备

Ubuntu 对诸如内存、声卡、显卡的要求不比 Windows 系统低，但是由于硬件厂商通常直接与微软合作来保证兼容性，而让开源社区自行解决 Ubuntu 上的硬件支持，因此和 Windows 相比，Ubuntu 支持总有些滞后。特别是在笔记本上，对 Ubuntu 用户来说更具挑战。但正如前言所说，随着 Ubuntu 的发展，越来越多的公司和群体加入 Ubuntu 中来，对 Ubuntu 的硬件支持也在不断改观。接下来让我们先认识一下计算机的硬件系统。

2.1.1 计算机系统基础

任何计算机系统都是由硬件体系和软件体系两大部分组成的，其中软件体系依附于硬件体系，并在其上层完成具体的逻辑功能。下面我们简要介绍一下计算机硬件系统，而 Ubuntu 操作系统及其上运行的应用软件则是本书的主题，后续将全面介绍。计算机硬件系统由运算器、控制器、存储器、输入设备和输出设备五大部件组成。

运算器

运算器是一个"信息加工厂"。数据的运算和处理工作就是在运算器中进行的。这里的"运算"，不仅是加、减、乘、除等基本算术运算，还包括若干基本逻辑运算。

控制器

控制器是整个计算机的指挥中心，它取出程序中的控制信息，经分析后，便按要求发出操作控制信号，使各部分协调一致地工作。

存储器

存储器是存放程序和数据的地方，它还可以根据命令读取已保存的数据。

（1）存储器的主要技术参数

存储器的主要技术参数有存储容量、存取速度和位价格（即一个二进制位的价格）。

（2）存储器容量

存储器容量表示计算机存储信息的能力，并以字节（Byte）为单位。1 个字节为 8 个二进制位（bit）。由于存储器的容量一般都比较大，尤其是外存储器的容量提高得非常快，因此又以 2^{10}（1024）为倍数不断扩展单位名称。这些单位的关系如下：

1Byte＝ 8bit　1KB＝1024Byte　1MB＝1024KB　　1GB＝1024MB

（3）存储器系统的组成

存储器系统包括主存储器（内存储器）、辅助存储器（外存储器）和高速缓冲存储器（Cache）。三者按存取速度、存储容量、位价格的优劣组成层次结构，以提高 CPU 越来越高的速度要求，并较好地解决三个技术参数的矛盾。

（4）主存储器

主存储器用来存放当前参与运行的程序、数据和中间信息。它与运算器、控制器进行信息交换，其特点是：存储容量小，存取速度快，位价格适当。存储信息遇到特殊情况就不能保留（断电即丢失）。

输入设备

最常见的有键盘和鼠标，我们可以通过键盘的输入和鼠标的操作把一些基本的信息传输到计算机中；还有计算机中的硬盘和软盘，可以将事先存放在磁盘中的信息通过操作传送到计算机中去；此外还有扫描仪、数码照相机、数码摄像机等，可以把一些拍好的照片和录像传输到计算机中；麦克风也可以作为输入设备，它可以结合计算机中的软件操作把声音传输到计算机中去。输入设备中还有电子触摸屏，在邮局我们可以直接在触摸屏上进行操作，查询到全国各地的邮政编码。

输出设备

输出设备是人与计算机交互的一种部件，用于数据的输出。它把各种计算结果数据或信息以数字、字符、图像、声音等形式表示出来。常见的有显示器、打印机、绘图仪、影像输出系统、语音输出系统、磁记录设备等。

2.1.2 定位安装主机

在第一章提到 Linux 的常用功能和主要应用，如用于桌面型个人计算机或简单工作站，或用于网络服务器等。根据不同应用，我们安装 Ubuntu 系统时，对硬件系统的要求也不一样，因此我们需要根据实际需求定制不同的硬件配置。

个人应用计算机

硬件系统方面，可能对 CPU、主板的要求不是特别高，但是对多媒体的要求比较高，要求配置较好的显卡、声卡，另外还有音箱、麦克风等。操作系统上，Ubuntu X-Window 当前已经非常成熟，完全可以满足一般用户的多媒体应用。

（1）个人桌面应用

如果你是 Linux 世界的新手，并想尝试使用这个系统，个人桌面安装是你最恰当的选择。该类

安装会为你的家用、便携计算机，或桌面使用创建一种带有图形化环境的系统。个人桌面安装最适用于新用户。它会安装一种图形化的桌面环境（X 窗口系统），并为家庭和桌面使用创建一种理想的系统。下面是为只安装一种语言（如英语）的个人桌面安装所推荐的磁盘空间需求的最小值。

- 个人桌面：1.7GB。
- 兼选 GNOME 和 KDE 的个人桌面：1.8 GB。
- 如果你计划选择所有软件包组（例如，办公/生产率应用程序是一个软件包组），并且还选择了额外的单个软件包，你可能至少需要 5.0 GB 的磁盘空间。

（2）开发应用工作站

如果除了图形化桌面环境外，你还需要软件开发工具，工作站安装类型是你最恰当的选择。工作站安装会安装一个图形化桌面环境和 X 窗口系统，以及软件开发工具。下面是为只安装一种语言（如英语）的工作站安装所推荐的磁盘空间需求的最小值。

- 工作站：2.1GB。
- 兼选 GNOME 和 KDE 的工作站：2.2GB。
  如果你计划选择所有软件包组（例如，办公/生产率应用程序是一个软件包组），并且还选择了额外的单个软件包，你可能至少需要 5.0 GB 磁盘空间。如果你提供了多余的空间，那么就可以在需要的时候安装额外的数据了。

企业组织服务器

如果用户希望系统具有多种基于 Linux 的服务器功能，对 Ubuntu 的图形界面要求不高，但是对 Samba、Mail、TCP 等服务的要求很高。硬件系统方面，可能对 CPU、主板、性能和稳定性要求很高；同时要求硬盘的空间比较大，且相应速度快，性能稳定。下面是为只安装一种语言（如英语）的服务器安装所推荐的磁盘空间需求的最小值。

- 服务器（至少，无图形化界面）：850MB
- 服务器（全部选择，无图形化界面）：1.5GB
- 服务器（全部选择，包括图形化界面）：5.0GB

如果你计划选择所有软件包组，并且还选择了额外的单个软件包，你可能至少需要 5.0 GB 磁盘空间。在服务器安装中，除非你在软件包选择过程中安装适当的软件包，否则当系统引导时，X 窗口系统不会被配置，因而不会载入 GUI。

特殊应用系统

如果以上三种方式都不符合你的要求，那么就可以考虑使用定制安装。定制安装在安装中给予你最大的灵活性，你可以选择你的引导装载程序、想要的软件包等。对于那些熟悉 Ubuntu 安装的用户以及那些想要灵活安装的用户而言，定制安装是最恰当的选择。在定制安装中，你对将要在你的系统上安装的软件包有完全的控制。为定制安装推荐的磁盘空间需求如下所示。

- 定制（至少）：475MB
- 定制（全部选择）：5.0GB

2.2 Ubuntu 的安装文件

有许多途径可以获得 Ubuntu 的安装文件，所有这些途径都在 Ubuntu 官方网站的下载页面中进行了说明。我们可以从离自己最近的镜像服务器下载 ISO 映像文件，然后刻录 CD 安装 Ubuntu。自初始发行起，Ubuntu 即提供一片安装光碟与一片用来预览的 Live CD。在 Ubuntu 6.06 LTS 发布后，将原来只用作预览的 Live CD 更改为不只可以用来预览，并且可以使用图形接口进行安装的光碟（即 Desktop CD），而原来只提供文字安装接口的安装光碟则保留并改名（即 Alternate install CD）。另外，获取 Ubuntu 安装文件的其他途径是加入本地的 Linux 用户组（LUG），询问是否有人可以为您制作一份拷贝。一般来说，您可能需要支付 CD 和邮寄的成本。如图 2.1 所示为 Ubuntu 发行版光盘。

图 2.1 Ubuntu 发行版光盘（免费索取）

2.2.1 Ubuntu 的版本号

Ubuntu 的版本号是根据其发布日期而定，由该次发布的年份和月份组成，但并未反映其实际版本。Ubuntu 首次发布是在 2004 年 10 月，因此该版本为 4.10，同时该版本还有个代号：Warty Warthog。作者在编写本书时，Ubuntu 的最新版本是 8.04，也就是 2008 年 4 月发行的，它还有个代号：Hardy Heron。具体版本号请参考 http://wiki.Ubuntu.org.cn 首页下方的当前版本部分。如：

8.10 . Intrepid Ibex . 2008–10
8.04 . Hardy Heron . 2008–04
7.10 . Gutsy Gibbon . 2007–10
7.04 . Feisty Fawn . 2007–04
6.06 . Dapper Drake . 2006–06

2.2.2 Ubuntu 的发布形式

Ubuntu 的发布形式有两种：Desktop CD 和 Alternate CD。

Desktop CD 图形界面安装光盘

Desktop CD 可以让你无需改变计算机就能尝试 Ubuntu 系统，以后是否永久安装由你决定。这种 CD 是多数人想要使用的。要安装这个版本，主机至少需要 256MB 的内存。

Ubuntu 发布的 Linux 里面有一个非常具有迷惑性的版本–Desktop。因为有一套 Server，所以 Desktop 很容易从字面上被理解成桌面版，而这样理解的人通常会弄不明白剩下那个 Alternate 是什么东西。

其实 Desktop 是 Live CD 的名字，也就是刻录在光盘上运行的 Linux，是一套已经装好的系统。把它刻录到光盘上放进光驱就可以直接运行这套完整的 Linux。当然，你也可以在运行之后把它“安装”到硬盘上，而这样的安装更像是一种复制或者还原，类似 ghost。

Alternate CD 文字界面安装光盘

Alternate CD 可以使你执行一定的专家级的 Ubuntu 安装。它用于以下情形：

- 创建预配置 OEM 系统；
- 设置自动布置；
- 在无网络链接情况下从旧系统升级；
- LVM 和 RAID 分区；
- 在内存小于 256MB 的系统上安装（但是注意相应的低内存系统可能无法运行完整的桌面环境）。

Ubuntu 真正意义上正统的安装版本则是 Alternate，它是由许多 deb 包组成的。用户可以详细地制定安装内容，因此 Alternate CD 是正宗的硬盘安装盘。

Alternate CD 和 Desktop CD 的比较

Alternate 和 Desktop 两个版本在速度上也有较大区别。因为 Desktop 类似系统还原，其安装速度较快，而 Alternate 正式的安装相比之下会慢一些。而运行的时候恰好相反：Desktop 因为是既成的，可能会缺少对应软件环境的适应性，所以和灵活指定及安装的 Alternate 相比会慢一些。

Desktop CD 安装前可以看到基本界面和试用（LiveCD 功能），Alternate 面向安装个性化需求更高的更专业的用户。Desktop 和 Alternate 都有基本的软件包，要更多的软件包只要修改合适的源下载也是很快的。Desktop 是一个 Live CD，只用于桌面系统，也可以在 LiveCD 状态下安装。

Alternate 是标准的安装 CD，它包含了一些桌面程序，可以用高级安装模式安装，在安装时可以划分分区，也可以当作 Server 来用，不过一些 Server 程序要自己通过网络来安装 Alternate。安装过程会有高级分区以及 grub 安装选项，Desktop 则没有。

在桌面环境里面，以向导的方式引导安装 Live CD 就是把安装好的软件放到光盘中，安装时就把其上的内容复制到硬盘再简单配置就行了。

Alternate CD 上放的 DEB 的软件包，安装时要解压、配置，只有在计算机硬件配置很低时用 live CD 安装较慢，其余都比 Alternate 要快。

2.2.3 Ubuntu 安装文件的选择

当前 Ubuntu 发布版支持 Intel x86（IBM–compatible PC），AMD64（Hammer）和 PowerPC（Apple iBook、Powerbook，G4 和 G5）架构。其实现在也支持 Sun 的 Sparc 架构了，不过这些对我们意义不大，普通用户用的基本都是 Intel 或 AMD 的桌面或笔记本 CPU，只要从前两者（Intel x86 和 AMD64）

中挑选就行了。

Desktop CD

其中，Desktop CD 有以下两种形式。

（1）PC（Intel x86）Desktop CD

PC（Intel x86）Desktop CD 用于绝大多数 PC 机，包括大多数使用 Intel、AMD 等处理器的机器和能运行 MS Windows 系统的绝大多数机器，基于 Intel 处理器的新型 Apple Macintosh 机也可以使用该系统。如果你不确定，就选择这个版本。

（2）64-bit PC（AMD64）Desktop CD

选择这个版本可以充分发挥基于 AMD64 或 EM64T 架构（比如，Athlon64、Opteron、EM64T Xeon）的计算机的性能。如果你用的是 AMD 的非 64 位处理器，或者你需要充分支持 32 位代码，请选择 Intel x86 映像文件。

Server Install CD 服务器安装光盘

Server Install CD 允许用户在计算机上永久安装 Ubuntu 作为服务器使用。它不会安装图形用户界面。这其中包括三种映像可用，分别用于不同类型的计算机：

（1）PC（Intel x86）Server Install CD

用于绝大多数 PC。包括大多数使用 Intel、AMD 等处理器的机器和能运行 MS Windows 系统的绝大多数机器，基于 Intel 处理器的新型 Apple Macintosh 机也可以使用该系统。如果你不确定就选择这个版本。

（2）64-bit PC（AMD64）Server Install CD

选择这个版本可以充分发挥基于 AMD64 或 EM64T 架构的计算机的性能。如果你用的是 AMD 的非 64 位处理器，或者你需要充分支持 32 位代码，请选择 Intel X86 映像文件。

（3）SPARC Server Install CD

用于 Sun UltraSPARC 计算机，包括基于多核 UltraSPARC T1（"Niagara"）处理器的机器。

Alternate install CD

Alternate install CD 有如下形式的版本，分别用于不同类型的计算机。

（1）PC（Intel x86）Alternate Install CD

用于绝大多数 PC。包括大多数使用 Intel、AMD 等处理器的机器和能运行 MS Windows 系统的绝大多数机器，基于 Intel 处理器的新型 Apple Macintosh 机也可以使用该系统。如果你不确定就选择这个版本。

（2）64-bit PC（AMD64）Alternate Install CD

选择这个版本可以充分发挥基于 AMD64 或 EM64T 架构（比如，Athlon64、Opteron、EM64T Xeon）的计算机的性能。如果你用的是 AMD 的非 64 位处理器，或者你需要充分支持 32 位代码，请选择 Intel x86 映像文件。

总而言之，没有什么特殊要求就选择 32 位版（x86）的，如果想体验 AMD64 64 位系统，你需要做好有少数软件无法运行的心理准备。不过对刚入门的新手来说，建议还是使用 x86 Ubuntu。

对于桌面用户来说，Desktop 和 Alternate 这两种 CD 如何选择呢？一般选择 Desktop 可以让你通过它的 Live CD 功能先了解一下 Ubuntu 是什么样子，选择 Alternate 可以让安装时的兼容性更好些。当然如果实在为难，你又有 DVD 光驱的话，DVD 是个不错的选择，它相当于 Desktop 和 Alternate 的合集，还自带了大量的软件包，对于没有联网的朋友很有帮助。

另外，Ubuntu 根据使用桌面环境的不同，还分为 Ubuntu（采用 Gnome 桌面环境，界面比较简洁、用户最多）、KUbuntu（采用 KDE 桌面环境，界面比较华丽、有些软件特别优秀）和 XUbuntu（采用 Xfce 桌面环境，比较小巧、速度较快），大家可以分别到“Gnome 桌面环境”、“KDE 桌面环境”、“XFCE 桌面环境”了解一下各个桌面环境的特点再选择。如果你着急体验 Ubuntu，那作者推荐你使用 Gnome 桌面，相信不会让你失望。

2.2.4 Ubuntu 镜像文件的获取

读者可以通过以下两种途径获取安装的镜像文件。

下载安装文件

从网络上下载 Ubuntu Linux 的镜像安装文件，然后使用光盘刻录软件将镜像文件刻录成光盘。本书讨论的是使用 Ubuntu 8.04 进行硬盘安装，所以推荐下载 Alternate 版，另外最近下载的人比较多，再推荐使用 bt 下载。

下载地址：http://Ubuntu.csie.nctu.edu.tw/Ubuntu-releases/8.04/。

申请安装光盘

向 Ubuntu 社区申请免费的安装光盘。在 Ubuntu 网站的 Shipit 页面上填写个人信息后，申请者可在一个月左右收到邮局递送的 CD 光盘。

申请地址：http://shipit.ubuntu.com。

2.3 试用 Desktop CD

桌面版光盘（Desktop CD）可以让你无须安装或动用你计算机上的硬盘就可以直接在 CD 上试用 Ubuntu 桌面。要从这张光盘上安装 Ubuntu，你最少需要 192MB 的内存。需要注意的是，这样的光盘系统不能提供系统升级的功能。

Desktop 版本启动之后可以进入一个 Live CD 模式，你可以在里面先体验一下，操作如下：先将你计算机的 BIOS 设置成首先使用光盘启动，如图 2.2 所示，再将 Ubuntu 桌面（Desktop）光盘放入光驱并重新启动计算机。

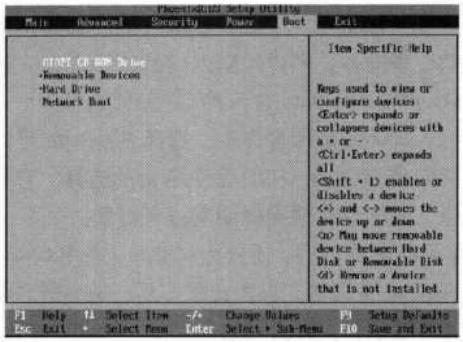


图 2.2 光盘启动的 BIOS 设置

主机由光盘启动后，如无意外，Ubuntu 的安装程序首先进入启动界面，如图 2.3 所示。

图 2.3 Desktop 光盘的启动界面

在启动界面中，最突出的就是 Ubuntu 的 Logo。Logo 的下方有 7 个功能选项：

(1) Start or install Ubuntu（启动 Ubuntu）：启动 Live Ubuntu 系统或直接开始安装 Ubuntu。

(2) Start Ubuntu in safe graphics mode（以安全图形模式启动 Ubuntu）：若第一个选项启动有问题，请改用这个选项开机。

(3) Install with driver update CD（安装时加载驱动程序）

(4) OEM install (for manufacturers)（为厂商提供的安装版本）

(5) Check CD for defects（检查 CD 是否有问题）

(6) Memory test（内存测试）：可以用这个项目检查计算机的内存有没有问题。

(7) Boot from first hard disk（从第一个硬盘启动）：你可以用这个选项启动硬盘中的操作系统。

在界面的下方，显示了 6 个功能键的功能。

F1 Help（求助）

第一次进入 Ubuntu 的安装启动界面时，可能会无所适从，这时候你就可以通过 F1 功能键获取 Ubuntu 的安装帮助信息，包括安装的前提条件、光盘启动模式、启动参数等，如图 2.4 所示。

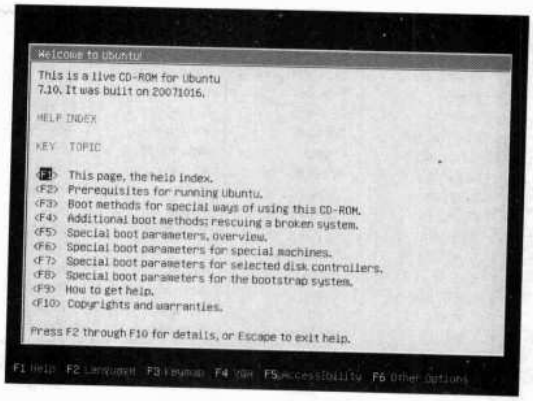


图 2.4 获取安装帮助

F2 Language（语言）

按 F2 进入语言选择窗口，选择安装 Ubuntu 的语言，默认情况下系统安装英文界面，这里我们选择熟悉的【中文（简体）】，如图 2.5 所示。需要注意的是，虽然我们选择了中文字体，但是由于 Ubuntu 对中文的支持度目前还不是很完善，因此在接下来的介绍中，读者将看到部分介绍的软件使用的都是英文资源文件。

图 2.5 选择语言

F3 Keymap（键盘映射）

按 F3 键打开键盘映射选择窗口，设置安装的键盘类型，默认情况下是标准的英文键盘，如图 2.6 所示。

F4 VGA

按 F4 键打开显示器分辨率选择窗口，设置安装过程中显示器的分辨率，如图 2.7 所示。如果不能确认显示器分辨率的话，可以选择 VGA 选项，安装程序将自动为你选择合适的分辨率。

图 2.6 选择键盘类型界面

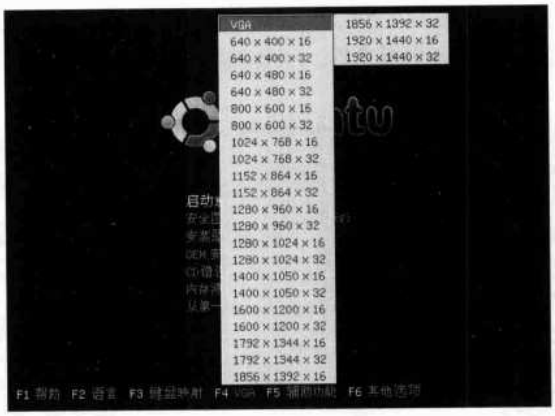


图 2.7 设置显示器分辨率

F5 Accessibility（辅助功能）

Ubuntu 在安装的过程中，也考虑到了残障人员的特殊情况，提供了一些特殊的辅助功能以帮助这些人员来完成系统的安装。按 F5 键打开辅助功能菜单，如图 2.8 所示。辅助功能包括：高对比度显示（High Contrast）、放大显示（Magnifier）和屏幕阅读器（Screen Reader）、盲文终端（Braille Terminal）、键盘修改（Keyboard Modifiers）和屏幕显示键盘（On-Screen Keyboard）。

F6 Other Options（其他启动选项）

如果在安装启动引擎中需要设置启动参数，按 F6 键输入 Boot Options 提示命令行，以便用户设置高级启动参数，如图 2.9 所示。

完成上面的设置后，选择启动或安装 Ubuntu，按回车键启动 Ubuntu，系统很快加载 Linux 内核，启动 Ubuntu LiveCD，这个过程大概需要 10 秒钟左右，如图 2.10 所示。

图 2.8 辅助功能设置

图 2.9 高级启动参数设置

图 2.10 启动 LiveCD

需要注意的是，从 LiveCD 启动到进入 Ubuntu 的桌面过程中，都是在光盘上完成的，而对主机的硬盘资源不会有影响。当你看到如图 2.11 所示的界面的时候，就表示 Ubuntu 已被启动。注意，此时启动的 Ubuntu 是光盘启动的 Ubuntu，它并没有被安装到计算机硬盘上。

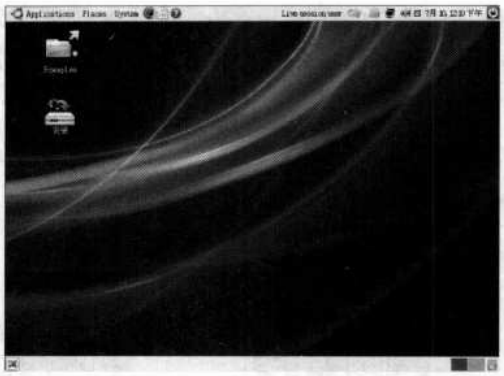


图 2.11 Ubuntu LiveCD 试用桌面

现在你可以体验一下这个 Ubuntu。桌面左上角的 Examples 文件夹中有一些多媒体及文件供用户测试之用，其中一个叫 Experience Ubuntu.ogg 的文件是由南非首位民选总统曼德拉 (Nelson Mandela) 解释 Ubuntu 意义的语音文件，如图 2.12 所示。

图 2.12 Ubuntu 试用目录

在图 2.11 所示的 Ubuntu 桌面上，除了 Examples 目录，还有一个“安装”图标，单击这个图标，可以进入桌面安装程序，将 Ubuntu 安装到硬盘。具体安装流程和 Alternate CD 安装一样，我们集中在后面介绍。

2.4 图形界面安装

在系统的安装过程中，无论系统采用的是什么引导方式，但安装的过程都是相同的。为了方便描述 Ubuntu 的安装过程，我们以在主机上安装单独的 Ubuntu 系统为例子，并且紧接上面使用 LiveCD 试用的介绍，使用“安装”图标完成 Ubuntu 系统的图形界面下硬盘安装进行介绍。

双击如图 2.11 所示的“安装”图标，启动【安装向导】对话框，这个安装过程分为 7 个步骤，每步骤完成特定的设置，然后在剩余的安装过程中就不再需要用户的参与了。

Step 01 **语言选择。**安装向导的第一步是询问用户希望安装过程中和最终完成的系统使用的默认语言，如图2.13所示，在这里我们选择【中文（简体）】作为我们安装的语言，然后单击【前进】按钮，进入下一步设置。

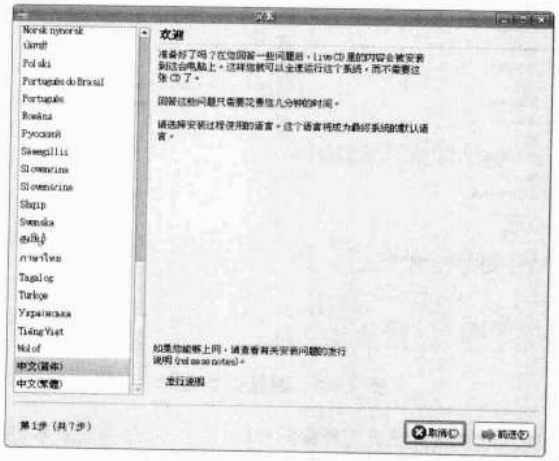


图 2.13　语言选择

Step 02 **国家地区选择。**安装向导的第二步为选择用户所在的国家或地区，由于我们在第一步中选择了【中文（简体）】，安装向导自动推荐了【所选城市】为Shanghai，同时，安装向导还自动选择相应的时区和时间，如图2.14所示，单击【前进】按钮，进入下一步设置。

图 2.14　国家地区选择

Step 03 **键盘类型。**安装向导的第三步为提示用户选择键盘类型，如图2.15所示。由于不同语言使用的键盘不一样，所以在这个设置上要谨慎选择。如果实在是不清楚键盘的类型，可以直接选择安装向导中提供的默认键盘类型。最后单击【前进】按钮，进入下一步设置。

图 2.15　键盘类型选择

Step 04 **硬盘分区。**安装向导的第四步是进行硬盘的分区，这一步是系统安装过程中最关键的一步，这是给新系统确认安装位置的地方，安装向导首先启动分区工具，对硬盘情况进行分析，如图2.16所示。因为我们讨论的是单操作系统安装，无论硬盘是否已被使用，安装程序将会对硬盘进行格式化，这个操作是具有破坏性的，会删除硬盘上原有的一切数据信息，所以在确认这个操作前，最好做好硬盘的数据备份工作。最后单击【前进】按钮，进入下一步设置。

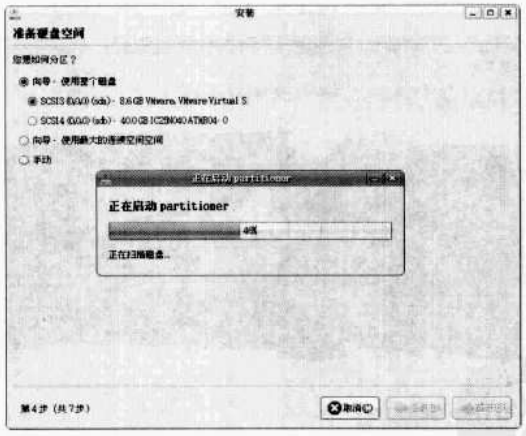


图 2.16　硬盘分区方式

Step 05 **导入文档和设置。**安装向导的第五步是导入文档和设置，也就是说把用户之前所用系统的文档和相关的配置文件导入到新安装的Ubuntu系统中，比如：IE的收藏夹、Firefox书签、聊天记录等可以直接导入到新的操作系统中，方便用户从另一个操作系统顺利过渡到Ubuntu系统中。不过，因为我们现在安装的是单系统，先前没有任何的用户和系统记录。安装向导将自动跳转到第6步设置。

Step 06 **用户账号和主机名设置。**安装向导的第六步是配置系统的初始用户名和主机名。我们都知道Linux操作系统中权限最高的是root用户，出于系统安全方面的考虑，安装向导要求创建一个用户来取代root，用于执行非管理任务的普通用户账号，如图2.17所示。除了创建用户外，第六步还包括了设置主机

名的任务。这个主机名用于标识主机接入网络后的身份。最后单击【前进】按钮，进入下一步设置。

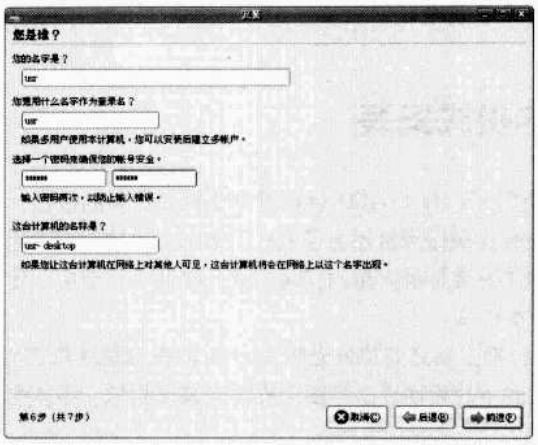


图 2.17　用户账号和主机名设置

Step 07 **安装信息提示。**这是安装向导的最后一步，它将用户在这次安装过程中所做的配置信息归纳总结并显示在如图2.18所示的窗口内，以便用户检查。

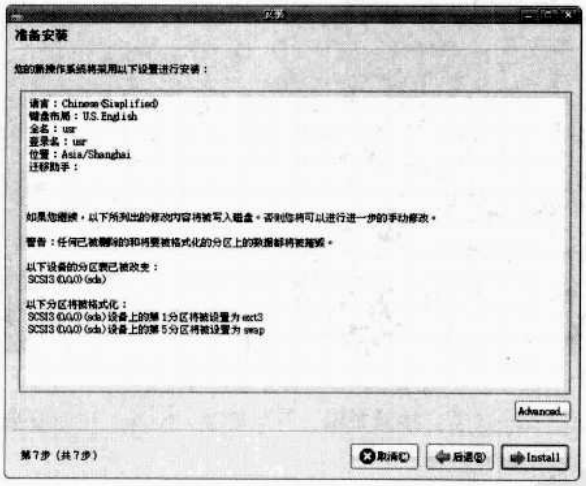


图 2.18　安装信息提示

单击Install按钮，开始正式安装Ubuntu系统，安装的过程可能会耗费较长时间，如图2.19所示。

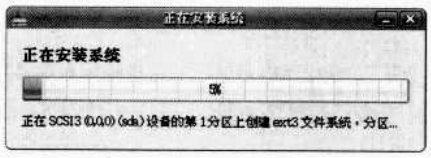


图 2.19　安装 Ubuntu 系统

系统安装完毕后，安装向导提示用户从光盘中取出安装光盘，然后主机将重新启动，这时候就可以进入全新安装的 Ubuntu 系统中。

2.5 文本模式安装

上一节我们讲述了使用 Ubuntu LiveCD 进行图形化界面安装的过程，不过对于很多主机来说，使用图形化界面来进行安装是缓慢而且不方便的，因为图形化的窗口占用了很多主机资源，因而在实际安装过程中，文本模式安装是最常用的安装方式。在这一小节中，我们将介绍在文本模式下安装 Ubuntu Linux 系统的基本步骤。

设置主机从光盘启动，把安装光盘放进光驱，Ubuntu 的安装程序将首先进入启动界面，如图 2.20 所示。可以看出，文本模式下的启动界面与图形界面安装很相似，具体的功能描述可参看 2.4 节的描述。在这里就不做介绍。

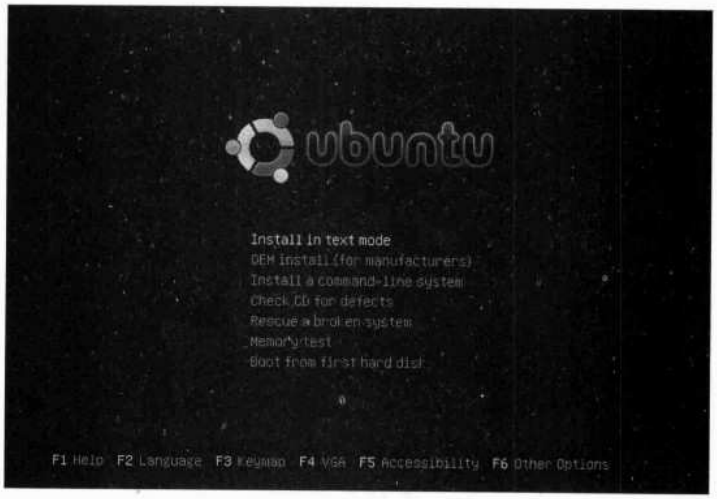


图 2.20 Ubuntu 安装启动界面

选择 Install in text mode 选项，按回车键，开始安装 Ubuntu。这时安装程序自动装载 Linux 内核，并进入 Ubuntu 安装前的设置准备。

Step 01 **语言选择。**加载Linux内核结束后，安装程序提示用户选择安装过程中和最终系统的语言，如图2.21所示。使用上下方向键选择需要的语言，这里我们选择【中文（简体）】。

Step 02 **国家地区选择。**根据Step 1选择的语言，安装程序将自动给出候选的国家列表，如图2.22所示，我们选择【中国】。

Step 03 **键盘类型选择。**在这一步中，安装程序提示用户是否检查键盘的类型，如图2.23所示。如果选择【否】，将使用默认的键盘类型，也就是标准的美国键盘。一般情况下，选择【否】选项就可以了。

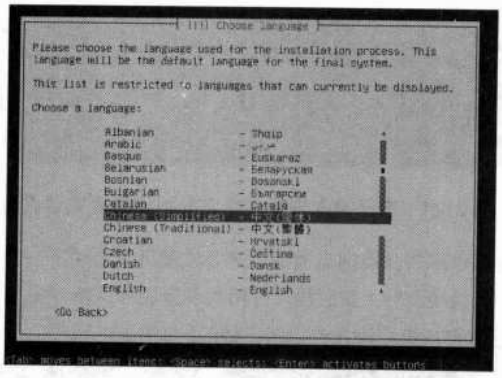


图 2.21　语言选择

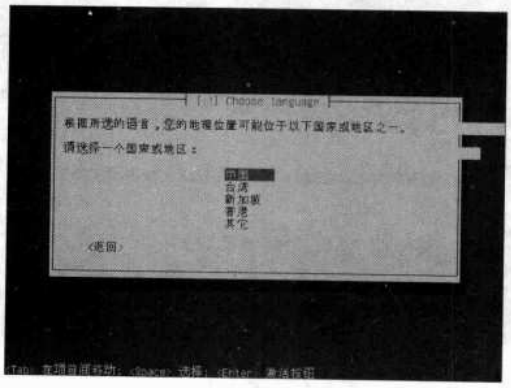


图 2.22　国家地区选择

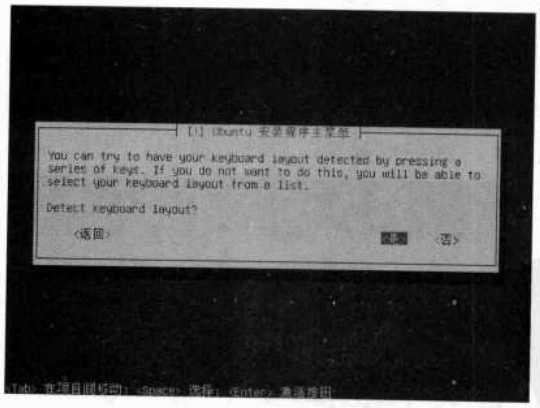


图 2.23　是否检测键盘类型

如果用户使用的确实不是标准键盘，没关系，那就选择【是】选项，安装程序将弹出如图2.24的窗口，提示用户用键盘输入窗口中提示的任意一个字符，以便安装程序能够正确地检查出到底使用的是什么键盘类型。

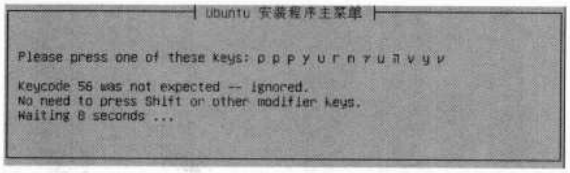


图 2.24　选择输入任意一个字符以便检测键盘类型

按照提示继续，安装程序将获取键盘类型，这时候就可以继续下一步的安装设置。

Step 04 **主机名。**完成键盘类型检测后，安装程序开始检测光驱和网络，安装程序将检测主机所在的网络是否支持DHCP，然后再提示用户输入主机名，如图2.25所示。

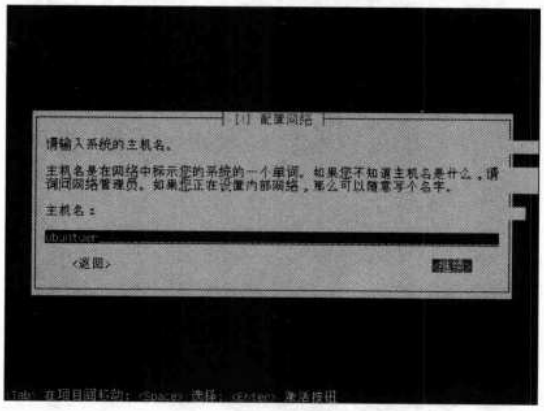


图 2.25　输入主机名称

Step 05 **硬盘分区。**就像我们前面图形界面安装中提到的一样，硬盘分区是系统安装的重要环节，需要小心谨慎地处理。在进行分区前，安装程序将对整个硬盘的状况进行检测。对于单操作系统来说，无论硬盘是否已被使用，安装程序将会对硬盘进行格式化，这个操作是具有破坏性的，会删除硬盘上原有的一切数据信息，所以在确认这个操作前，最好做好硬盘的数据备份工作。开始进入硬盘分区操作，我们选择【向导 – 使用整个磁盘】，如图2.26所示。

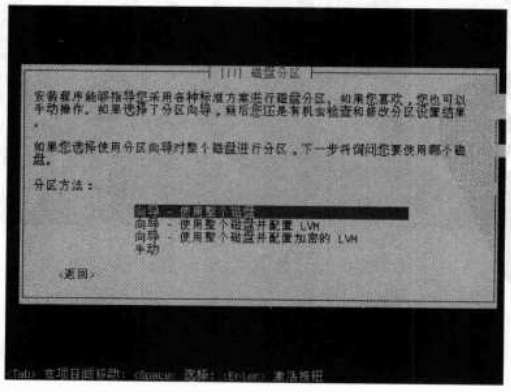


图 2.26　选择分区方式

按回车键，进入如图2.27所示的界面，选择硬盘SCSI3。

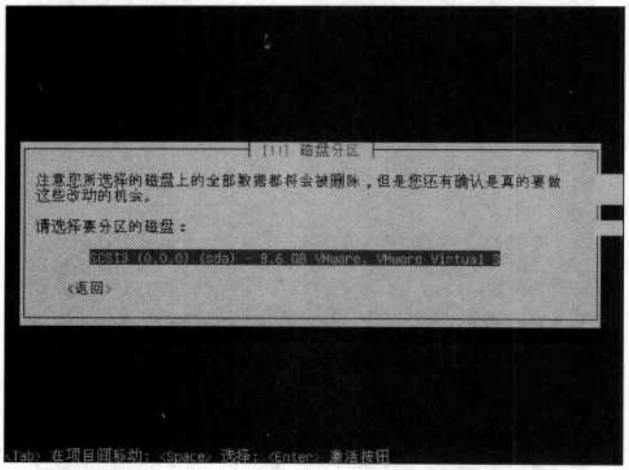


图 2.27　选择分区磁盘

按回车键，进入如图2.28所示的分区信息窗口，选择【是】，安装程序开始格式化硬盘。

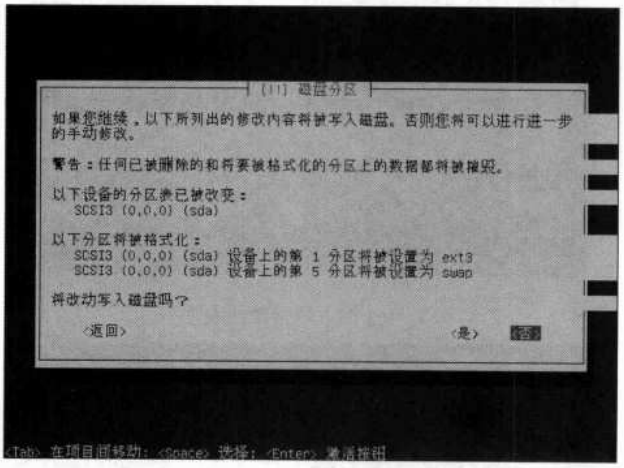


图 2.28　分区信息

Step 06 **时钟设置。**在完成硬盘分区后，进入时区和时间设置，如图2.29所示。系统时间使用的是协调世界时（Universal Time Coordinated，UTC），安装程序根据你选择的国家地区选择对应的时区，并转换成当地时间。

Step 07 **创建初始用户账号。**首先输入账户使用者姓名，如图2.30所示。按回车键进入下一个设置，安装程序要求输入账号的用户名称，如图2.31所示。

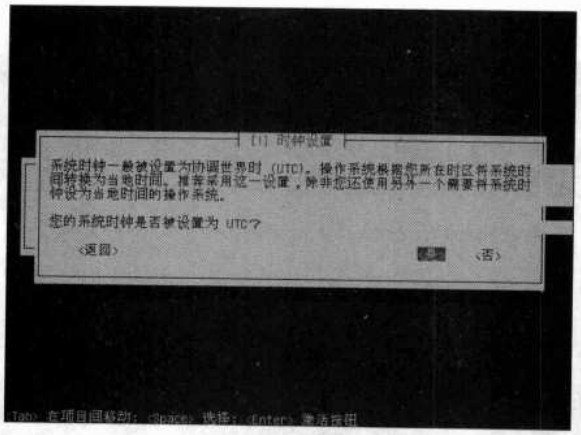


图 2.29 时钟设置

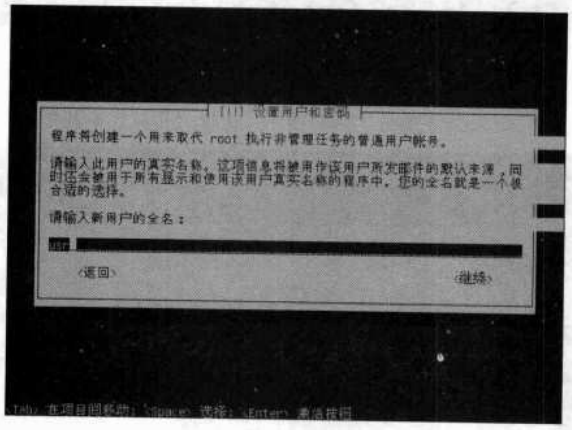


图 2.30 输入账号使用者姓名

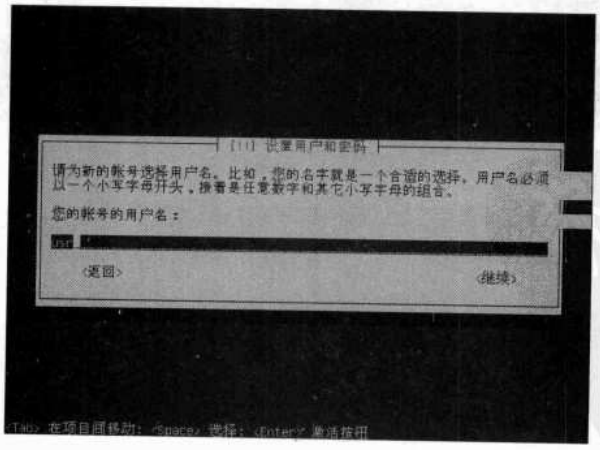


图 2.31 输入账号的用户名

按回车键进入密码设置，同时要求重复输入用户的密码，如图2.32和图2.33所示：

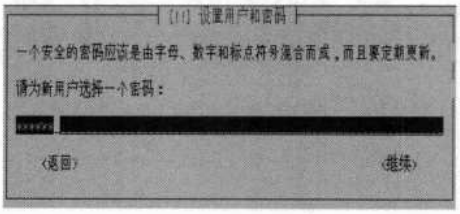


图 2.32　输入密码

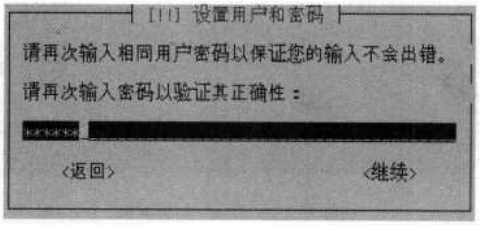


图 2.33　重复输入密码

Step 08 **安装基本系统。**完成账户的创建后，开始进入Ubuntu系统的基本安装，这一步将花费不少时间，如图2.34所示。

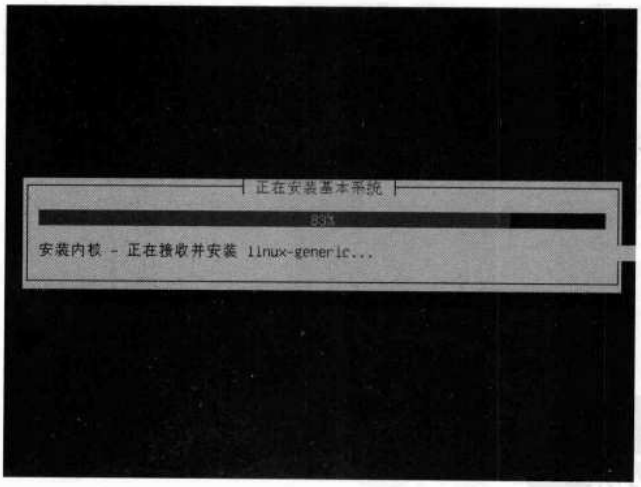


图 2.34　基本系统安装

在安装基本系统过程中，要设置apt软件包管理器和配置软件源的地址，如果主机所在的网络需要使用HTTP代理，可以在图2.35中输入代理HTTP地址。

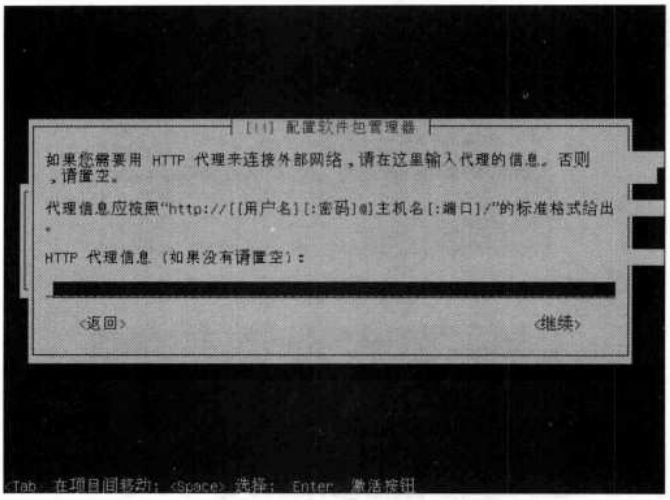


图 2.35 基本系统安装

Step 09 **分辨率设置。**紧接着，安装程序提示用户设置显示器的分辨率，如图2.36所示。当然，用户可以选择多个分辨率，不过最好不要选择显示器不支持的分辨率。因为系统重启后将使用分辨率最高的那个，这样有可能在显示器上不能完全显示窗口，造成不必要的麻烦，推荐使用1024 × 768的分辨率。

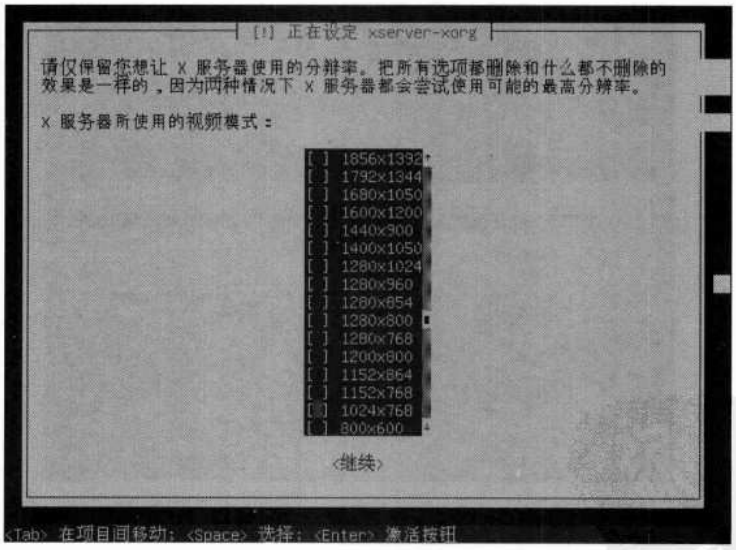


图 2.36 设置显示分辨率

Step 10 **安装结束。**完成显示设置后，系统将继续执行安装程序，大概30分钟后，提示整个系统安装结束，如图2.37所示。选择【继续】选项，完成整个安装过程，系统自动重启。

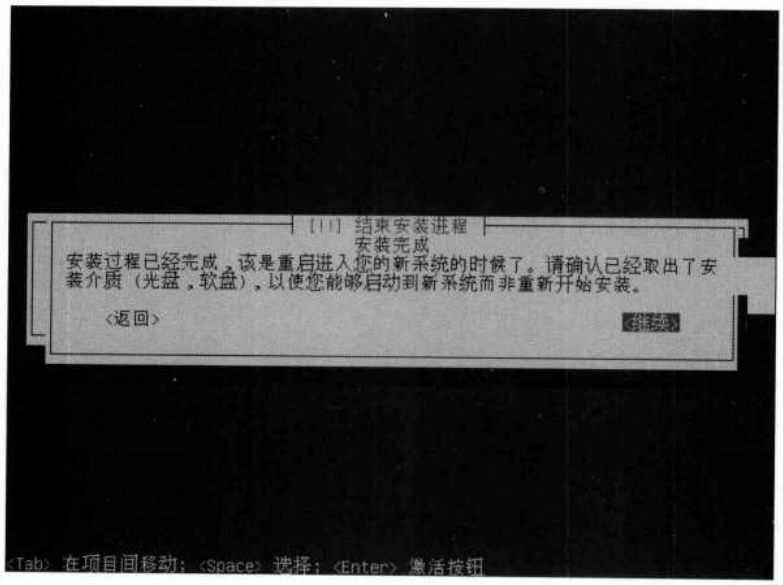


图 2.37　结束安装

系统重启后，就进入全新的 Ubuntu 系统，这时候就能看见登录窗口，使用安装过程中设置的用户账户（usr）登录，将看到崭新的 Ubuntu 系统桌面。

2.6　课后练习

1．计算机系统是否由硬件体系和软件系统两部分组成？硬件系统由哪五大部分组成？

2．想想自己将要安装的 Ubuntu 主要用于什么用途，准备以哪种形式安装。

3．Ubuntu 安装文件发布有哪两种模式？如何通过邮件的方式获取 Ubuntu 安装盘？

4．请尝试到 Ubuntu 网站中查到最新的 Ubuntu 安装版本和最近的镜像文件，下载一个 CD 镜像文件。

5．Desktop/Alternate 两种安装模式各自的用途及安装方式是什么？

6．请在实体机器或者虚拟机上尝试安装 Ubuntu，体验安装过程中各个步骤的操作。

Chapter03 Ubuntu 初体验

经过前面两章对 Ubuntu 的概要介绍和安装配置，下面就先轻装进入 Ubuntu 世界，一睹为快，体验一把它的华美界面与强大功能。本章首先会对 Ubuntu 开机过程及多系统引导管理器 GRUB 进行简单介绍；然后在紧接的后面两节中，针对 Ubuntu 的两个工作环境——桌面模式和 Shell 工作模式，进行简单体验式的介绍，旨在让大家对 Ubuntu 有个初步的认识；当然，第一次使用 Ubuntu 少不了关机，因此还会简单地介绍 Ubuntu 的关机模式；另外，Linux 帮助系统是 Linux 持续学习中不可多得的一个好工具，第五节中会专门介绍 Ubuntu 中的在线帮助系统及其他一些帮助途径；最后一节对用户在使用过程中最常遇到的问题进行专题排解。

3.1 第一次进入系统

每次开机时 Ubuntu 系统都将加载到机器上，本节中我们主要通过分析 Linux 的启动过程和 GRUB 引导程序来介绍一下这个过程。

3.1.1 Ubuntu 的启动过程

当我们按下已经安装好 Ubuntu 系统的计算机开机按钮时，计算机启动后运行的第一个程序是 GRUB。GRUB 是一个多重操作系统启动管理器，简单地说，它既能引导 Linux，同时也能引导 Windows。GRUB 引导加载界面如图 3.1 所示。

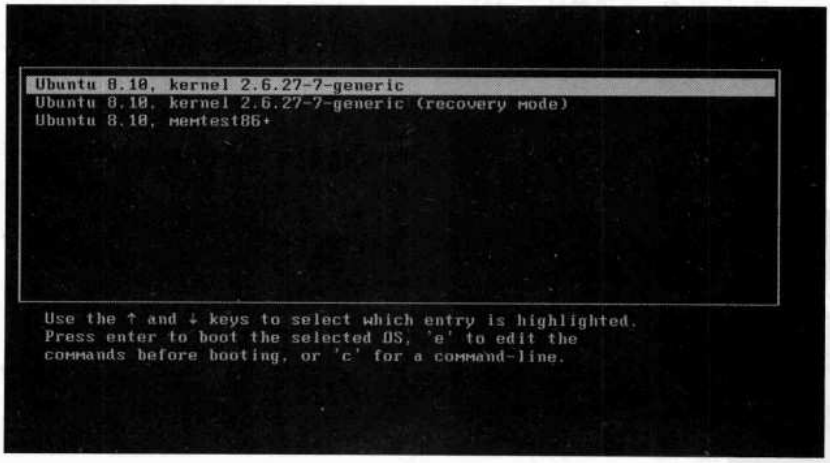


图 3.1 GRUB 加载界面

当 GRUB 把 Ubuntu 内核挂载后，它就会自动退出，计算机系统就开始进入 Ubuntu 自启动过程。

对于安装了 X-Window 的 Ubuntu 系统（一般为默认安装），默认进入 X-Window 图形模式。Ubuntu 启动的界面如图 3.2 所示。

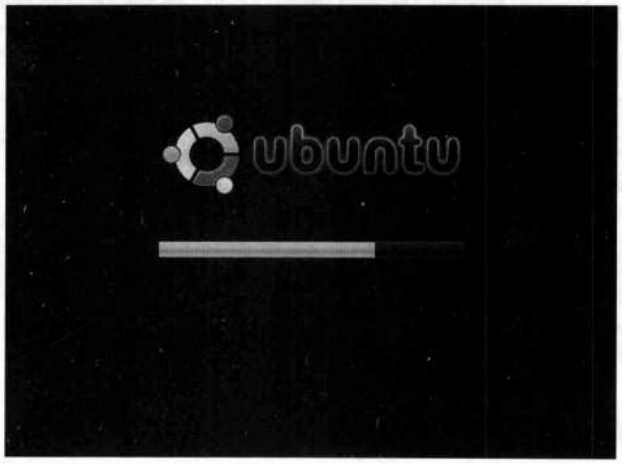


图 3.2 启动界面

系统启动完成，并进行用户登录后，就能进入 Ubuntu 桌面系统了，在这里，有一个华丽而丰富的 Ubuntu 世界等着我们。

另外，如果用户没有安装 X-Window 环境或系统设置默认启动为 Shell 环境时，我们登录后的界面是黑屏文字环境，在这里我们可以通过 Ubuntu Shell 环境进行 Linux 几乎所有的功能设置。

3.1.2 多系统引导管理器 GRUB

前面我们提到过 GRUB，下面重点介绍 GRUB 这个系统启动引导管理器。系统启动引导管理器是在计算机启动后运行的第一个程序，它负责将控制加载、传输到操作系统的内核，一旦把内核挂载，系统引导管理器的任务就算完成。系统引导的其他部分，比如系统的初始化及启动过程则完全由内核来控制完成。

在 x86 架构的机器中，Linux、BSD 或其他 Unix 类的操作系统中，最为常用的系统引导管理器有 GRUB 和 LILO 两个。从目前看来，GRUB 有逐渐取代 LILO 之势，GRUB 2.0 正在开发之中。GRUB 启动系统有两种方式，一种是直接通过命令行方式调用 Ubuntu 启动程序，一种是通过配置文件 menu.list 启动，下面我们分别对这两种方式进行一下简单介绍。

通过 GRUB 命令行来启动 Ubuntu

当我们把 GRUB 的 menu.lst 写错，或者丢掉了 menu.lst 的时候，比如在开机的时候，GRUB 会出现类似 grub>的命令提示符，这时需要我们用命令行启动系统。当然，您可以不用定义 GRUB 的菜单，直接用命令行来启动系统。另外，GRUB 的命令行才是王道，如果知道怎么用命令行来启动操作系统，那理解 menu.lst 的写法也不难，因为 menu.lst 的内容就是 GRUB 的一个一个的指令集合。

通过命令行来引导操作系统的流程，即手动把指令输入到 grub>提示符的后面，一步一步达到启动系统的目的，在这个过程中，如果您不知道有哪些命令，可以输入 help。另外，当我们在输入

命令时不太确定命令名或文件名时，可以适当地应用 Tab 键的自动补全功能

GRUB 命令行模式界面如图 3.3 所示。

图 3.3　GRUB 命令行模式界面

GRUB 命令行模式启动系统步骤大致如下：

Step 01　按Ctrl+C组合键进入GRUB的命令行模式，会出现grub> 提示符；

Step 02　通过cat (hd0, 6) /etc/fstab指令查看分区信息；

Step 03　通过root (hd[0—n], y) 指令指定/boot所在的分区；

Step 04　通过kernel 指令指定Linux的内核及所在的分区；

Step 05　通过initrd命令行来指定initrd文件；

Step 06　通过boot指令引导系统。

下面是一个 GRUB 引导 Linux 系统的实例（“/boot”和 Linux 的根“/”处于同一个硬盘分区的情况）：

```
grub> cat (hd0, 6) /etc/fstab
# This file is edited by fstab-sync - see 'man fstab-sync' for details
LABEL=/ / ext3 defaults 1 1
/dev/devpts /dev/pts devpts gid=5, mode=620 0 0
/dev/shm /dev/shm tmpfs defaults 0 0
/dev/proc /proc proc defaults 0 0
/dev/sys /sys sysfs defaults 0 0
LABEL=SWAP-hda1 swap swap defaults 0 0
/dev/hdc /media/cdrecorder auto pamconsole, exec, noauto, managed 0 0

grub> root (hd0, 6)
 Filesystem type is ext2fs, partition type 0x83

grub> kernel /boot/vmlinuz-2.6.11-1.1369_FC4 ro root=/dev/hda7 注：输入
 [Linux-bzImage, setup=0x1e00, size=0x18e473]

grub> initrd /boot/initrd-2.6.11-1.1369_FC4.img 注：输入 intrd 文件名的全名
 [Linux-initrd @ 0x2e1000, 0x10e685 bytes]
```

```
grub> boot
```

GRUB 的配置文件的 menu.lst

上面我们提到 menu.lst 内容其实就是 GRUB 的一个指令集合。一般来说 menu.lst 位于/boot/grub 目录中，也就是/boot/grub/menu.lst 文件；我们可以用 vi 或您喜欢的编辑器进行编辑，如果您不会用 vi，可跳到相应章节简单参看 vi 的使用。

/boot/grub/menu.lst 的内容示例：

```
default=0
timeout=5
#splashimage= (hd0, 6) /boot/grub/splash.xpm.gz
hiddenmenu
title Fedora Core (2.6.11-1.1369_FC4)
 root (hd0, 6)
 kernel /boot/vmlinuz-2.6.11-1.1369_FC4 ro root=LABEL=/
 initrd /boot/initrd-2.6.11-1.1369_FC4.img
title WinXp
 rootnoverify (hd0, 0)
 chainloader +1
```

下面对上面的示例进行简单介绍。

(1) default=0 是默认启动哪个系统，从 0 开始；每个操作系统启动的定义都是从 title 开始的，第一个 title 在 GRUB 的启动菜单上显示为 0，第二个启动为 1，依此类推。

(2) timeout=5 表示开机后 GRUB 画面出现 5 秒后开始以默认方式启动；如果在启动时移动上下键，则解除这一规则。

(3) #splashimage= (hd0, 6) /boot/grub/splash.xpm.gz 表示 GRUB 的背景画面，这个是可选项，加#号表示注释掉不喜欢 GRUB 的背景画面，也可以删除#号。

(4) hiddenmenu 表示隐藏 GRUB 的启动菜单，这项也是可选的，也可以用#号注释掉。

后面的内容是对 Linux 操作系统启动的实质内容，一般要包括 4 行：title 行、root 行、kernel 行和 initrd 行。

(5) title XXXXX：title 后面加一个空格，title 是小写的，后面的 XXXXX 可以自己定义，如 FC4。

(6) root (hd[0–n], y)：在本例中，我们看到的是 root (hd0, 6)，root (hd[0–n], y) 表示的是/boot 所在的分区。有时我们安装 Linux 的时候，大多是不设置/boot 的，这时/boot 和/在同一个分区。这个 root (hd[0–n], y) 很重要，因为/boot 目录中虽然有 grub 目录，最为重要的是还有 kernel 和 initrd 文件，这是 Linux 能启动起来最为重要的元素。

(7) kernel /boot/vmlinuz–2.6.11–1.1369_FC4 ro root=LABEL=/：kernel 行是指定内核及 Linux 的/分区所在位置。在这里以 kernel 起始，指定 Linux 的内核的文件所处的绝对路径。因为内核是处在/boot 目录中的，如果/boot 是独立的一个分区，则需要把 boot 省略，这行要写成 kernel /vmlinuz–2.6.11–1.1369_FC4 ro root=LABEL=/。

(8) initrd /boot/initrd-2.6.11-1.1369_FC4.img：如果/boot 是独立的一个分区，initrd 一行要把/boot 省略；如果/boot 不是独立的一个分区，而是和 Linux 的/分区处于同一分区，则不应该省略。

3.2 Ubuntu 桌面体验

Ubuntu 默认的桌面环境采用 GNOME X-Window，一个 Unix 和 Linux 主流桌面套件和开发平台。下面我们从桌面整体介绍、桌面环境设置，以及应用小程序的使用三个角度对 Ubuntu 环境进行体验式介绍。

3.2.1 桌面简介

Ubuntu 8.04 经过两年多的升级，桌面系统已经日趋完美，功能越来越丰富，使用也越来越便捷。下面我们就来对 Ubuntu X-Window 桌面进行简单介绍。

总体来说，Ubuntu 桌面的工作区域可以分为如下几个大块：

(1) **桌面** 最主体部分，上面有快捷方式，同时也是弹出窗口的主显示区。
(2) **任务栏面板** 默认在显示区顶端，包括菜单入口时钟等。
(3) **状态栏面板** 显示区最低端，可以显示当前打开窗口等。

桌面环境如图 3.4 所示。

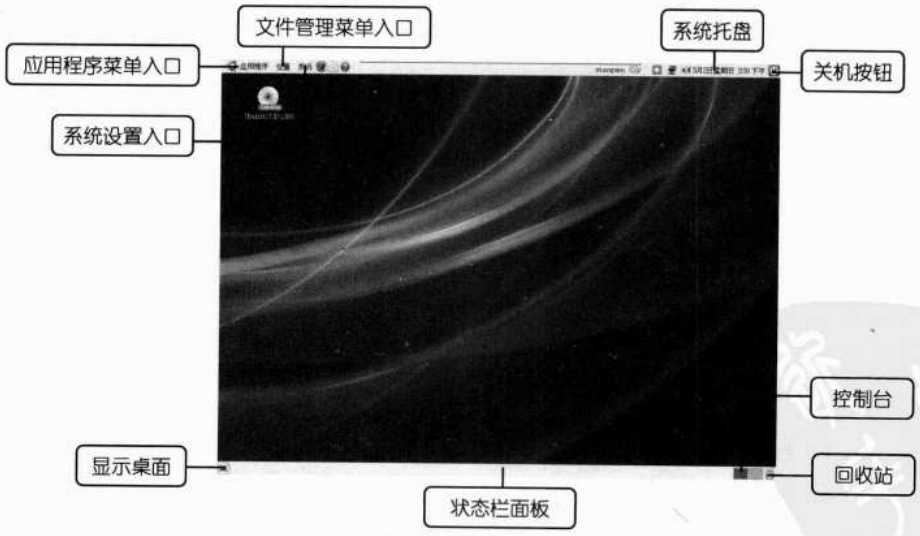


图 3.4 Ubuntu 桌面介绍

3.2.2 桌面应用的体验

Ubuntu 8.04 给人很大的视觉冲击，在诸如 NTFS 的写入、办公软件的更新和桌面搜索等方面也有不小的突破。

体验音乐播放器

当您放入一个音频 CD 时，Sound Juicer CD 播放器和提取器会自动打开，这时用户可以选择喜欢的歌曲进行播放。如果要提取音频 CD，需要选中提取的歌曲，然后单击 Extract 按钮或按 Ctrl+Enter 组合键。如果已经连接了 Internet，那么 Sound Juicer 将从 MusicBrainz.org 这个由社区维护的、超过 360 000 套专辑的数据库中检索 CD 艺术者、标题以及音轨的数据。

手工启动 Sound Juicer，可以通过选择【应用程序】|【影音】| Sound Juicer CD Extractor 进入窗口界面。

您可以使用首选项窗口来选择在您计算机上提取的音频文件的保存位置、提取的音频文件名以及这些文件的格式和编码。在主窗口选择【编辑】|【首选项】，Sound Juicer 能够把声音文件解压成各种格式，如 Ogg Vorbis，FLAC 等。CD 播放器的界面如图 3.5 所示。

图 3.5 CD 播放器

玩玩 ubuntu 自带的小游戏

Ubuntu 默认安装了许多游戏，包括 Aisleriot 纸牌、Gnome 方块和扫雷等，读者可以在【应用程序】|【游戏】中打开 Ubuntu 的游戏菜单，如图 3.6 所示。

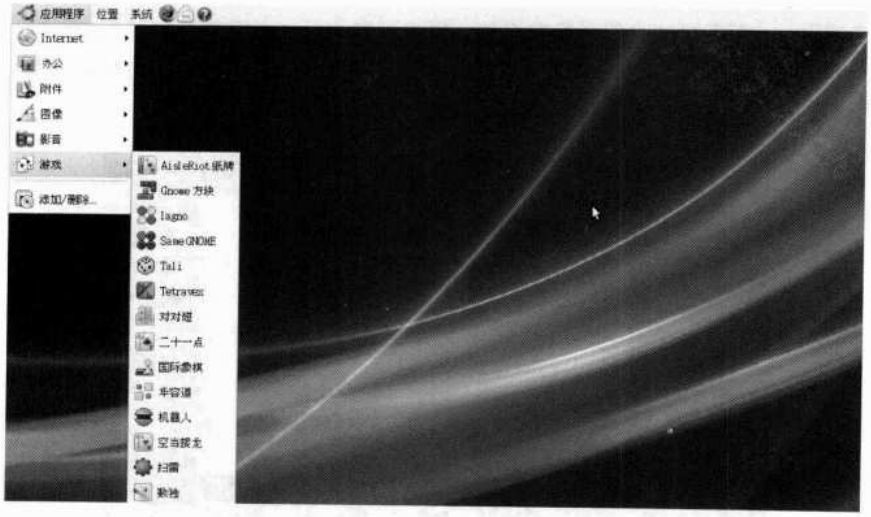


图 3.6 Ubuntu 游戏菜单

3.3 Ubuntu 终端体验

Linux 终端最初用于文件浏览器，即使是现在，当图形环境失效时，它仍被用作文件浏览器，所以学习中可以将终端理解为一个文件浏览器，用来浏览自己的文件和撤销曾做过的改动。在 Linux 发展初期，用户界面只有终端，这时终端也称为 Linux Shell 命令行。Linux 的几乎所有功能都可通过 Shell 命令来控制，所以在过去，这就是人机交互的主要方式。尽管目前 Linux 特别是 Ubuntu 中，大部分程序都有相应的图形工具，但有时这些图形工具还是会捉襟见肘，不够用，此时便是命令行大显身手的时候，因此即便现在，许多 Linux 用户仍觉得 Shell 命令比图形方式更快，更有优势。

因此，作为 Linux 用户，我们很有必要了解终端工作模式，并掌握常用的那些 Shell 命令。终端的使用并不像广大初学者想象的那么困难，使用 Shell 命令并不需要专门知识，和其他软件一样，它也仅仅是一个程序。下面就让我们开始 Ubuntu 终端的体验之旅吧。

3.3.1 如何进入 Ubuntu 终端

默认 Shell 模式

如果安装 Ubuntu 时没有选择 X-Window，系统就只有 Shell 工作界面，这时用户就直接工作在 Shell 环境下了。另外对于一些 Linux 的老用户，若在安装 Ubuntu 时特意指定系统默认启动模式为 Shell 时，系统也可直接进入 Shell 工作环境。关于 Shell 工作环境与 X-Window 窗口工作环境的切换我们将在 3.3.2 进行详细说明

窗口模式下启动终端

当然对于广大初学者来说，一般都习惯于在窗口模式下工作学习或娱乐。Ubuntu X-Window 为

了便于大家使用，一般包含一个终端窗口组件，用户可以调用这个组件。

在窗口模式下，我们可以通过选择【应用程序】|【附件】|【终端】打开终端窗口，终端窗口如图 3.7 所示。

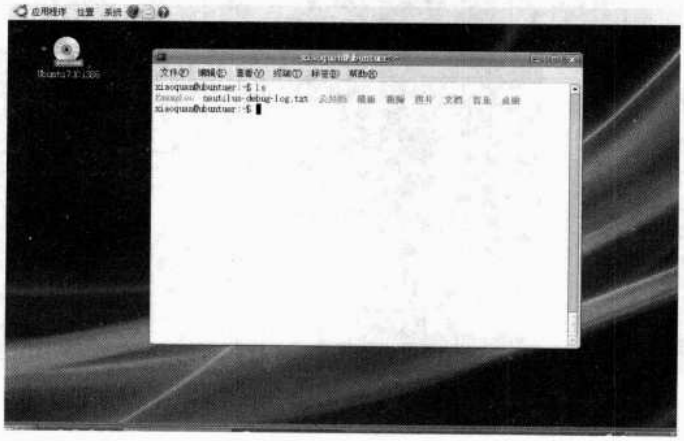


图 3.7 Ubuntu Term 终端

3.3.2 X-Window 与全 Shell 环境切换

在 Ubuntu 中使用命令行的常见方法是启动一个终端，但有些时候还是需要切换到真正的控制台下。

通常，Ubuntu 的图形模式与字符模式可以进行自由切换。

（1）在 Ubuntu 系统中，当选择图形模式启动，屏幕上开始出现鼠标的时候，就可以按 Ctrl+Alt+F2 直接进入命令行模式。

（2）以后也可以按 Ctrl+Alt+F7 切换到图形模式。

一共可以使用 6 个控制台，分别用快捷键 Ctrl+Alt+F1 到 Ctrl+Alt+F6 进行切换。

对于一些 Linux 的老用户，他们比较习惯与在命令行下工作，如果需要设置 Ubuntu 开机进入 Shell 字符模式，可以进行如下操作：

```
sudo apt-get install sysv-rc-conf
sysv-rc-conf 把 gdm 去除
```

操作界面十分简洁，你可以用鼠标单击，也可以用键盘方向键定位，用空格键选择，用 Ctrl+N 翻下一页，用 Ctrl+P 翻上一页，用 Q 退出。

3.3.3 基本输入与指令格式

Linux Bash Shell 的内部命令有 40 个，主要包括 exit、less、lp、kill、cd、pwd、fc、fg 等，包含了文件和目录操作、用户和组的配置与管理、挂载外部存储设备命令等。

Linux 命令通用格式

Linux 命令的通用格式为：

```
命令字 【命令选项】 【命令参数】
```

注：下列命令解释均是在命令行界面执行回车后的结果。

常用命令体验

（1）查看目录：- ls

ls（LiSt）用不同颜色、经过排列的文本列出目录下的文件。

（2）创建目录：- mkdir（目录名）

mkdir（MaKeDIRectory）命令可以创建目录。

（3）切换目录：- cd（/directory/location）

cd（ChangeDirectory）命令可以从您的当前目录切换到您指定的任意目录。

（4）复制文件/目录：- cp（源文件或目录名）（目标目录或文件名）

cp（CoPy）命令会复制您指定的任意文件，cp –r 命令则可以复制您指定的任意目录（注：包括该目录里的文件和子目录）。

（5）删除文件/目录：- rm （文件或目录名）

rm（ReMove）命令可以删除您指定的任意文件，rm –rf 命令则可以删除您指定的任意目录（注：包括该目录里的文件和子目录）。

（6）重命名文件/目录：- mv （文件或目录名）

mv（MoVe）命令可以重命名/移动您指定的任意文件或目录。

（7）查找文件/目录：- locate （文件或目录名）

locate 命令会在您的计算机里搜索指定的任意文件。它使用系统中的文件索引以便进行快速查找，运行命令 updatedb 可以更新该索引。每次开机，该命令便会（在合适的时机）自动运行。运行该命令需要具备管理员权限（参见第 3.3.4 节–root 用户和 sudo 命令）。

还可以使用通配符来匹配一个或多个文件，如*（匹配所有文件）或?（匹配一个字符）。

常用示例

```
[ubuntuer@localhost usr]# ls -ahl
total 188K
drwxr-xr-x  14 root root 4.0K Oct 18 15:09 .
drwxr-xr-x  24 root root 4.0K Feb 25 08:37 ..
drwxr-xr-x 2 root root  36K Dec 21 04:03 bin
drwxr-xr-x 2 root root 4.0K Feb 12  2006 etc
drwxr-xr-x 2 root root 4.0K Feb 12  2006 games
drwxr-xr-x  35 root root 4.0K Oct 18 15:09 include

[ubuntuer @localhost usr]# date
Tue Mar  4 21:57:13 CST 2008

[ubuntuer @localhost usr]# cal
```

```
    March 2008
Su Mo Tu We Th Fr Sa
    1
 2  3  4  5  6  7  8
 9 10 11 12 13 14 15
16 17 18 19 20 21 22
23 24 25 26 27 28 29
30 31
```

3.3.4 root 用户与 sudo 命令

GNU/Linux 系统的 root 用户具有系统的管理权限。出于安全考虑，普通用户并不具备这一权限。sudo 为 superuser do 的简写，即超级用户的工作，在 Ubuntu 的默认环境下，root（即管理员）账号是停用的，所有与系统相关的工作指令均需在指令前输入 sudo，并输入密码确认，这样做是为了防止因一时失误对系统造成破坏。

默认情况下，sudo 工具的默认密码是第一个账户的密码。因此在系统安装过程中创建的第一个用户账号具有使用 sudo 的权限。您可以通过用户和组来限制和赋予用户运行 sudo 的权限。当运行一个要求 root 权限的应用程序时，sudo 会要求用户输入自己的普通用户密码，这样可以防止恶意程序损害系统，还可以提醒用户应该小心谨慎地对待自己将要执行的管理操作。

在命令行里使用 sudo，只需直接在想执行的命令前加上 sudo 即可。随后 sudo 会提示用户输入自己的密码。在短时间内 sudo 会记住用户输入过的密码。设计这一特性的目的是为了避免用户在执行多个管理任务时重复输入密码。执行管理任务时请务必小心，以免损坏系统！下面是其他一些使用 sudo 的技巧。

（1）欲使用 root 终端，请在命令行里输入“sudo –i”。

（2）Ubuntu 里所有图形界面配置工具（启动时）默认已使用 sudo，因此如有必要它们会提示您输入自己的密码。

例如在命令行（可以是真的命令行模式，也可以是图形模式中附件菜单下的模拟终端），如果我们真的要安装 grub，要输入的命令是：

```
#sudo grub
grub> boot (hdx, y) //确定把硬盘 x 含有 GRUB 的 boot 目录安装到硬盘 y。
grub>setup (hdz) //安装 GRUB 的引导信息到硬盘 z。
grub>quit
```

其中，关于硬盘 x，y，z，可以按 Tab 键得到更多的提示信息以帮助用户作决定。如果按了 Tab 键后没反应，很可能是因为你用的只是命令 grub，而不是 sudo grub。

3.4 Ubuntu 的关机方式

计算机关机有硬关机和软关机两种方式，硬关机即按机器电源开关强制关闭系统，而软关机，则是系统前先行关闭正在运行的应用程序，同时进行内存回收、文件保存等善后处理，最后才正常

退出。硬关机对操作系统及硬盘等计算机硬件有很大损害，因此我们不提倡使用硬关机方式。

Ubuntu 作为 Linux 的一个成熟版本，也有着完善的关机功能。其 Shell 内核有着丰富的关机命令，而 X-Window 模式下也有关机按钮入口和关机模式选择窗口。一般来说，X-Window 的关机窗口底层也是调用 Shell 内核的相应命令。因此在本节我们先详细介绍 Ubuntu 常用的几个关机命令，然后在此基础上介绍一下可视化窗口模式下的关机方式。

3.4.1 Ubuntu 的 Shell 关机命令

很多教程也都有提到 Linux 关机命令，但大多只是简单地把命令罗列一下，而没进行区别介绍，因此有些让人难以理解，甚至让人产生诸如“其实 halt 就是调用 shutdown – h”这样的误解。其实 Linux 的关机命令有三个：halt，shutdown 和 poweroff，另外还有重启命令 reboot。shutdown 是专门负责关机的命令，并是独立的程序；halt 和 poweroff 是链接到 shutdown 的命令，没有独立的程序，其中 poweroff 直接调用 halt 的默认命令参数。下面从命令的功能原理、输入格式及参数含义等几个角度对 Ubuntu 的几个重要关机命令进行逐个介绍。

shutdown 命令

shutdown 命令可以安全地关闭或重启 Linux 系统，它在系统关闭之前给系统上的所有登录用户提示一条警告信息。该命令还允许用户指定一个时间参数，可以是一个精确的时间，也可以是从现在开始的一个时间段。精确时间的格式是 hh:mm，表示小时和分钟；时间段由“+”和分钟数表示。系统执行该命令后，会自动进行数据同步的工作。

shutdown 命令的工作原理是：在由命令 shutdown 发起的关机过程中，init 会试着运行 /etc/rc.shutdown 脚本，给所有进程发送 TERM 信号，最后给不按时停止的进程发送 KILL 信号。需要特别说明的是，要执行 shutdown 命令必须是 root 用户或 operator 组的成员。

```
[root@acnis root]# shutdown -help
shutdown: invalid option -- -
Usage: shutdown [-akrhfnc] [-t secs] time [warning message]
   -a:   use /etc/shutdown.allow
   -k:   don't really shutdown, only warn.
   -r:   reboot after shutdown.
   -h:   halt after shutdown.
   -f:   do a 'fast' reboot (skip fsck).
   -F:   Force fsck on reboot.
   -n:   do not go through "init" but go down real fast.
   -c:   cancel a running shutdown.
   -t secs: delay between warning and kill signal.
   ** the "time" argument is mandatory! (try "now") **
```

⚙语法

shutdown [–cfFhknr（参数名称）] [–t 秒数] [时间] [警告信息]

⚙参数

- –t seco：设定在几秒钟之后运行关机程序。
- –k：并不真正关机，只是发出警告信息给所有用户。

- -r：关机后立即重新启动。
- -h：关机后不重新启动。
- -f<秒数>：快速关机，重新启动时跳过 fsck。发出警告信息和关机信号之间要延迟多少秒。警告信息将提醒用户保存当前进行的工作。
- -F：关机时，强迫进行 fsck 动作。
- -n：快速关机，不经过 init 程序。
  一般的关机程序是由 shutdown 调用 init 来实现关机动作，使用此参数将加快关机速度，但是不建议用户使用这种关机方式。
- -c：取消一个已经运行的 shutdown。
  值得注意的是，当执行一个类似 shutdown -h 11:10 的命令时，只要按 Ctrl+C 键就可以中断关机的命令。若是执行类似 shutdown -h 11:10 &的命令将 shutdown 转到后台时，则需要使用 shutdown -c 将前一个 shutdown 命令取消。
- 时间：设定关机的时间。
  设置多长时间后执行 shutdown 命令。时间参数有 hh：mm 或+m 两种模式。hh：mm 格式表示在几点几分执行 shutdown 命令。例如"shutdown 10:45"表示将在 10:45 执行 shutdown。+m 表示 m 分钟后执行 shutdown。比较特别的用法是以 now 表示立即执行 shutdown。值得注意的是，这部分参数不能省略。
- 警告信息：要传送给所有登录用户的信息。

⚙应用举例

指定现在立即关机：# shutdown -h now

指定 5 分钟后关机，同时送出警告信息给登录用户：# shutdown +5

```
"System will shutdown after 5 minutes"
```

另外，MAN shutdown 得到以下内容，可以作为对照。

```
NAME
   shutdown - bring the system down
SYNOPSIS
   /sbin/shutdown [-t sec] [-arkhncfF] time [warning-message]
OPTIONS
   -a  Use /etc/shutdown.allow.
   -t  sec Tell  init (8) to wait sec seconds between sending processes
the  warning and the kill signal  before  changing  to
another  run-level.
   -k  Don't  really shutdown; only send the warning messages to every-body.
   -r  Reboot after shutdown.
   -h  Halt after shutdown.
   -n  [DEPRECATED] Don't call init (8) to do the  shutdown  but  do  it
ourself.  The use of this option is discouraged, and its results are not always what
you'd expect.
   -f  Skip fsck on reboot.
   -F  Force fsck on reboot.
   -c  Cancel an already running shutdown. With this option  it  is  of course
not possible to give the time argument, but you can enter a explanatory message on
the command line that will be  sent  to  all users.
```

halt 命令

Halt 命令用来关闭系统。其实 halt 命令就是调用 shutdown –h 命令。执行 halt 命令时，“杀死”应用进程，执行 sync 系统调用，文件系统写操作完成后就会停止内核。halt 会先检测系统的 runlevel，若 runlevel 为 0 或 6，则关闭系统，否则即调用 shutdown 来关闭系统。

```
[root@acnis root]# halt --help
usage: halt [-n] [-w] [-d] [-f] [-i] [-p]
   -n: don't sync before halting the system
   -w: only write a wtmp reboot record and exit.
   -d: don't write a wtmp record.
   -f: force halt/reboot, don't call shutdown.
   -p: power down the system (if possible, otherwise halt)
```

语法

```
halt [-dfinpw]
```

参数

- –d：不要在 wtmp 中记录。
- –f：不论目前的 runlevel 为何，不调用 shutdown 即强制关闭系统。
- –i：在执行 halt 命令之前，关闭全部的网络界面。
- –n：在执行 halt 命令之前，不用先执行 sync 命令。
- –p：执行 halt 命令之后，执行 poweroff。
- –w：仅在 wtmp 中记录，而不实际结束系统。

poweroff 命令

很多人不知道 poweroff 命令，这也是在/sbin 下面的命令，是一个连接到 halt –p 的命令上的 link。

```
[root@acnis sbin]# ls -ahl | grep pow
lrwxrwxrwx  1 root   root 4 2002-01-13  poweroff -> halt
```

为了进行说明，MAN 一下 poweroff 得到以下内容，可以作为对照。对于 halt –h 同时包含了 halt 命令和 reboot 命令，因为 poweroff 本就是 halt 的快捷方式而已。

```
NAME
   halt, reboot, poweroff - stop the system.
SYNOPSIS
   /sbin/halt [-n] [-w] [-d] [-f] [-i] [-p] [-h]
   /sbin/reboot [-n] [-w] [-d] [-f] [-i]
   /sbin/poweroff [-n] [-w] [-d] [-f] [-i] [-h]
OPTIONS
   -n   Don't sync before reboot or halt.
   -w   Don't actually reboot or halt but only write the wtmp record (in  the
/var/log/wtmp file) .
   -d   Don't write the wtmp record. The -n flag implies -d.
   -f   Force halt or reboot, don't call shutdown (8) .
   -i   Shut down all network interfaces just before halt or reboot.
   -h   Put all harddrives on the system in  standby  mode  just  before  halt
or poweroff.
```

```
      -p   When halting the system, do a poweroff. This is the default when halt
is called as poweroff.
```

reboot 命令

Reboot 命令可以让系统停止运作，并重新开机，其工作过程跟 halt 差不多，不过它是引发主机重启，而 halt 是关机，所以它的参数与 halt 相差不大。

语法

```
reboot [-dfinw]
```

参数

- -d：重新开机时不把数据写入记录文件/var/tmp/wtmp。本参数具有"-n"参数的效果。
- -f：强制重新开机，不调用 shutdown 指令的功能。
- -i：在重新开机之前，先关闭所有网络界面。
- -n：重新开机之前不检查是否有未结束的程序。
- -w：仅做测试，并不真的将系统重新开机，只会把重开机的数据写入/var/log 目录下的 wtmp 记录文件。

3.4.2 桌面环境下关机与注销

类似 Microsoft 的 Windows 桌面环境，在 Ubuntu 桌面环境下，也有专门的关机界面。单击 Ubuntu 界面上的相应按钮，调用底层的关机命令。

单击桌面左上角的关机按钮图标，即会弹出如图 3.8 所示的窗口，用户根据需要单击相应的按钮即可实现关机或重启及其他功能。

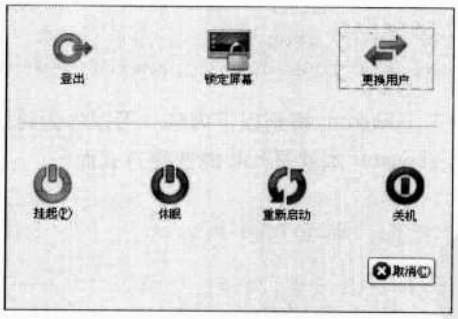


图 3.8 关机选项窗口

3.5 Ubuntu 在线帮助

许多初学者安装完一个 Linux 操作系统后不知从哪里入手学习 Linux，感觉没有帮助信息。其实当你安装了一个完整的 Linux 系统后其中已经包含了一个强大的帮助体系，只是可能你还没有发现

它或掌握使用它的技巧。在 Ubuntu 的学习与使用中，同样有着多种帮助途径。在本节中，我们简单地介绍一下 Ubuntu 的帮助体系——man 信息和在线帮助文档。另外，还介绍一下广泛的网络开放资源，如官网论坛、社区论坛等交流平台。

3.5.1 使用 help 查看系统命令帮助

前面提到过 Bash 的内部命令有 40 个，可以通过在命令行输入 help 获得 Bash 内置的命令列表。这些内部命令没有独立的命令程序（即无法搜索到这些命令）和帮助文件，help 命令提供这些命令的在线帮助。有趣的是，help 命令本身也是内部命令，所以使用的第一个 help 命令是：

```
#help help
```

上面是 help 命令把自己作为参数来获得自己的帮助。使用 help 命令提供某命令的帮助方法非常简单，在 help 命令后空一格输入命令名称即可，如：

```
#help kill
```

Help 命令提供的 kill 命令参数很详细，当你对 Linux 有一定了解后，往往只需要主要的语法，那么可以使用“-s”参数：该参数用于只显示被查询命令的简短语法描述，如：

```
#help -s kill
kill: kill [-s sigspec | -n signum | -sigspec] [pid | job]... or kill -l [sigspec]
```

3.5.2 man / info 应用

man 介绍

使用 man 命令获得手册页帮助，不仅可以获得命令的帮助信息，还可以获得配置新闻公报、设备文件、协议等多种类型的信息。

通常，用户只要在命令 man 后输入想要获取的命令的名称（例如 ls），man 就会列出一份完整的说明，其内容包括命令语法、各选项的意义以及相关命令等。

man 命令的一般形式为：man[选项]命令名称。

man 命令的常用选项和章节常用选项分别见表 3.1 和表 3.2。

表 3.1 man 命令的常用选项

常用选项	说明
-S	根据章节显示，由于一个命令名称可能会有很多类别，其类别说明如表 3.2 所示
-f	只显示出命令的功能而不显示其中详细的说明文件
-w	不显示手册页，只显示将被格式化和显示的文件所在位置
-a	显示所有的手册页，而不是只显示第一个
-E	在每行的末尾显示$符号

表 3.2 man 命令的章节常用选项

章节	说明
man1	一般用户的命令
man2	系统调用的命令，内核函数的说明
man3	C 语言函数库的命令
man4	有关驱动程序和系统设备的解释
man5	配置文件的解释，大多为“/etc”目录下各种配置文件的格式描述
man6	游戏和趣味小程序的命令
man7	其他软件或程序的命令和有关系统维护的命令
man8	系统管理工具手册，这些命令只有超级用户才可以执行
man9	Linux 系统例程手册

说明：

（1）手册页按照不同的类型放在不同的目录下，如：

```
#ls -d /usr/share/man/man?
```

（2）每个目录中都存放着对应类型的手册文件，手册文件大多为.gz 格式的压缩文件，命名规则为“手册名称.手册类型.gz”（如 hd.4.gz），如：

```
#ls /usr/share/man/man4
```

（3）使用 man N intro 命令可查看某类型手册页的说明，其中 N 为手册页的类型，如：

```
#man 4 intro
```

（4）如各类型中有同名的手册页，使用 man 命令时应先指定手册类型，再指定手册名称，如：

```
#man 1 passwd
#man 5 passwd
```

info 介绍

textinfo 是 Linux 系统提供的另外一种格式的帮助信息。和 man 相比，textinfo 具有更好交互功能。它支持链接跳转功能。通常使用 info 和 pinfo 命令来阅读 textinfo 文档。

Linux 中的大多数软件开发工具都来自自由软件基金会的 GNU 项目，这些软件的在线文档都以 textinfo 文件的形式存在。info 程序是 GNU 的超文本帮助系统。

info 文档存放在/usr/share/info 目录中。因为该文档通常提供整个软件项目的帮助文档，而不是某命令或配置文件的帮助，所以不是所有的软件包都带有该文档。

可以在 shell 提示符后输入 info（不带参数），它将列出一个文档的清单。如果您没有发现所需要的，那是因为没有安装包含那个文档的软件包，用 RPM 安装后再试，info 帮助系统的初始屏幕显示了一个主题目录，你可以将光标移动到带有*的主题菜单上面，然后按回车键进入该主题，也可以输入 m，后跟主题菜单的名称以进入该主题。例如，输入 m，然后再输入 gcc 就会进入 gcc 主题中。

info 系统是一个超文本系统，任何高亮度显示的文字都有一个链接导向更多的信息。使用 Tab 键将光标移动到链接，并按 Enter 键进入链接，按 p 键返回上一页，按 n 键翻到下一页，按 u 键回到

文档的上一层，输入 C-h 获得帮助，按 m 键进入选单界面。

如果要在主题之间频繁跳转，请记住如下的几个命令键：

- n：跳转到该节点的下一个节点；
- p：跳转到该节点的上一个节点；
- m：指定菜单名而选择另外一个节点；
- f：进入交叉引用主题；
- l：进入该窗口中的最后一个节点；
- TAB：跳转到该窗口的下一个超文本链接；
- RET：进入光标处的超文本链接；
- u：转到上一级主题；
- d：回到 info 的初始节点目录；
- h：调出 info 教程；
- q：退出 info；
- #info：直接用 info 命令可获得系统中 info 文档的分类列表；
- #info set：指定文档名称作为 info 命令的参数可直接查看相应的文档，可使用方向键进行查看。

3.5.3 在线文档 /usr/share/doc

许多程序带有附加的文档，如 Readme、FAQs (Frequently Asked Questions，常见问题）。这些文档可以在/usr/share/doc 目录下找到（版本 7.2 以前是"/usr/doc"）。

通常，这些文档是安装包的一部分。当然，也有例外，如 Linux 内核。如果安装内核源代码（kernel-sources），该文档在/usr/src/Linux/Documention 下；也可以安装单独的内核文档 (kernel-docs) 包，但这样的话，目录在/usr/（share）/doc/kernel-docs-[version]下。

HOWTOs、FAQs、E-Books 文档由 LDP（the Linux Documnet Project）成员维护，其中一部分为非英语版本。

HOWTOs 由经验丰富的 Linux 用户编写，现在已有 300 多个主题，比如配置特殊硬件，设置和使用软件，从其他系统的转换，问题处理等。如果您想了解某个专题，可以先看看 HOWTOs。这些 HOWTOs 的风格与您现在看的这些页面类似。

如果您已经有了离线版本，还是建议您到 Online index 看看，因为许多 HOWTOs 都会定期更新。

LDP 也有其他文档，比如和书相似的 Guides。这些文档涉及的主题很广泛，如安全、常规管理、网络及 Linux 内核等，其中一些是从 Linux 初级用户的角度写的。

LDP 还有一部分包含 FAQ documents，其中还有非常值得一读的 Linux FAQ。很多 Linux 的一般性问题都可以在这里找到答案。

软件包项目文档

Linux 中的大多数软件开发工具都来自自由软件基金会的 GNU 项目，这些软件包除了提供手册

页和 textinfo 外，还提供项目文档。软件包的存放目录是/usr/share/doc。每个软件包都拥有与其同名的文档目录，目录命名格式为“软件包名称，版本号”。tin 包版本 1.22 的文档就放在/usr/doc/tin-1.22 目录下。为了方便阅读，有些项目文档还提供了多种格式的文件。HTML 格式文件用浏览器阅读，ps 格式文件用 postscript Viwer 查看或打印输出，txt 格式文件用文本编辑器查看或在命令行下阅读。软件包的项目文档都包含特定的文件。

- AUTHOR：作者信息
- BUGS：系统存在的 Bug
- CHANGES：修改记录
- COPYING：版权声明
- INSTALL：安装方法
- LANGUAGES：语言类型
- NEWS：新版本的特点
- ONLINEHELP：在线帮助
- TODO：项目发展计划

HOWTO

HOWTO 文件是可供用户参考的联机文档的另一种形式，HOWTO 文件的文件名都有一个 HOWTO 后缀，并且都是文本文件。这些文件位于系统的/usr/share/doc/HOWTO 目录下。每一个 HOWTO 文件包含 Linux 某一方面的信息，例如它支持的硬件或如何建立一个引导盘。要想查看这些文件，进入 HOWTO 所在的目录，使用 more 命令：

```
#cd /usr/share/doc/HOWTO; more HOWTO
```

KDE Help Center

在 X-Window 下，使用浏览器来显示整个系统的帮助。一般来说，KDE 帮助通过把对应的图标放在 KDE 控制板上来实现。GNOME 帮助系统可以使用对应的 gnome-help-browser 命令，如图 3.9 所示。

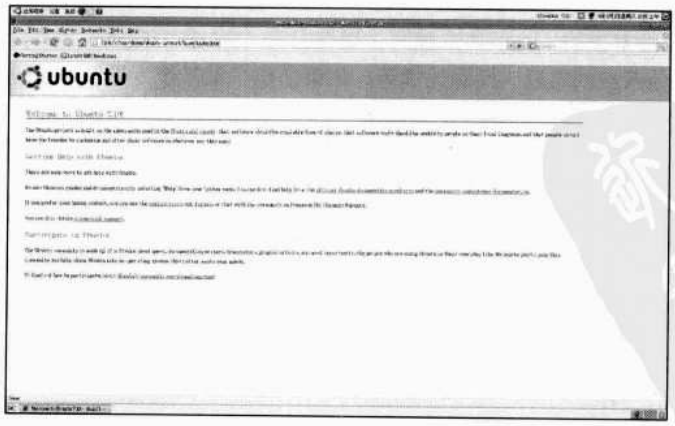


图 3.9 Ubuntu Document 查看窗口

3.5.4 Ubuntu 帮助社区

Ubuntu 官方中文论坛：http://forum.ubuntu.org.cn/。

Ubuntu 官方中文 wiki：http://wiki.ubuntu.org.cn/。

Ubuntu 官方英文 wiki：https://wiki.ubuntu.com/。

Ubuntu 官方英文论坛：http://www.ubuntuforums.org/。

Ubuntu magazine，英文杂志：http://www.fullcirclemagazine.org/。

Ubuntu geek，英文站点，有不少 Ubuntu 的学习、设置资料，其网址为：http://www.ubuntugeek.com/。

Ubuntu video，各种 Ubuntu 的视频，包括安装设置，其网址为：http://www.ubuntuvideo.com/。

Ubuntu screencast，Ubuntu 视频教程，含字幕下载，其网址为：http://screencasts.ubuntu.com/。

Ubuntu Document Storage Facility，英文 Ubuntu 文档，其网址为：http://doc.gwos.org/index.php/Main_Page。

3.6 本章问题排解

经过前面的 Unbuntu 之旅，相信大家一定被 Ubuntu 绚丽多彩的界面、丰富强大的功能所吸引，直到现在还流连于其中。当然有些经常使用 Windows 操作系统或 Linux 其他版本操作系统的用户，在自己的 Ubuntu 体验过程中会进行一些对比，从而产生一些疑问；还有些用户在 Ubuntu 使用过程中会遇到一些不解的小意外。本节就针对本章大家可能遇到的问题组织几个小专题进行集中分析，希望帮助大家更深入地体验一下 Ubuntu。

3.6.1 桌面图标哪儿去了

许多用过 Micrsoft Windows 操作系统的用户，一开始接触 Ubuntu 桌面时，他们常常感觉不习惯，常问道："桌面上的那些诸如'我的电脑'、'回收站'图标都到哪儿去了？"

不用着急，大家只是对 Ubuntu 界面暂时不习惯而已，Ubuntu X-Window 设计者考虑到：大部分用户在工作时，桌面上的图标几乎总是被窗口遮住，把窗口移开或切换到另一个工作区来获得图标常常十分痛苦，所以他们停用了所有特殊的桌面图标，而改在计算机菜单中提供了入口，这样当您只想打开文件管理器时，无需再做窗口清理工作了。另外，设计实现中采用了 trashapplet，它会在面板上提供一个可用的回收站图标，因而总能被访问到，它能够删除任何拖放进去的东西，而不仅仅是文件。

所以 Ubuntu 是为了更加方便大家的日常使用，而采用了另外一种风格的"我的电脑"和"回收站"的入口。如果用户想找回一些特殊的桌面图标，执行【程序】|【系统工具】|【配置编辑器】命令，浏览至 /apps/nautilus/desktop，选择想要显示的图标。

3.6.2 Ubuntu 开机黑屏问题

有些朋友提到："安装了 Ubuntu 7.10，但是自己安装完显卡驱动后开机再操作时，就会黑屏，以致不能进入到登录界面。"

关键看黑屏多久。如果超过 5 分钟就有问题了。Unbuntu 系统启动后，在进入 X-Window 时，短时间黑屏一下一般来说是正常的，因为等硬盘转完了就会出现登录界面的。

其实，Ubuntu 3D 桌面可以不用手动安装驱动，只需要多加两个文件，用自带的驱动完全可以运行得很好。用户只要做两件事：一是在软件源设置好之后，单击【系统】|【系统管理】|【受限驱动管理】，选中【显卡】；二是打开新立得，搜索 compiz，安装 compizconfig setting manager，搜索 xgl，安装 xserver-xgl。这样，已经安装好的驱动就是官方提供的版本，虽然版本低了点，但比冒险装非正式驱动好多了。然后，再搜索一个 start up manager 并进行安装，在【系统】|【系统管理】中找到，单击打开，改分辨率为 1024×768，色位 24bit，再重新启动计算机问题就解决了。

一般来说，不用自己去下载驱动程序的，那些是提供给用户试验用的，不稳定。

3.6.3 加速 Ubuntu 的开机过程

有些用户提到："我的 Ubuntu 系统开机过程有点慢，能不能像 Microsoft Windows 系统那样，对系统启动过程进行一下优化，从而达到更快启动 Ubuntu 系统的目的？"

回答是可以做到。对于已经比较熟悉 Ubuntu 的用户来说，通过下面 4 种方法可以加快 Ubuntu 7.10 的开机启动速度。

将关机选项中的 splash 改成 nosplash

```
sudo gedit /boot/grub/menu.lst

## ## End Default Options ##
title Ubuntu 7.10, kernel 2.6.22-14-generic
root (hd0, 3)
kernel /boot/vmlinuz-2.6.22-14-generic root=UUID=db4a6d44-20ef-4d59-b41b-
90c6dda1ab54 ro quiet nosplash locale=zh_TW
initrd /boot/initrd.img-2.6.22-14-generic
quiet
### END DEBIAN AUTOMAGIC KERNELS LIST
```

检查并关闭其他磁盘分区

```
sudo gedit /etc/fstab

# /dev/sda1
UUID=50BC-5067 /media/sda1 vfat defaults, utf8, umask=007, gid=46 0 0
# /dev/sda5
UUID=8A881E7D881E6849 /media/sda5 ntfs defaults, umask=007, gid=46 0 0
```

关闭一些省电的功能

```
sudo gedit /etc/rc.local

echo 10 >/sys/bus/usb/devices/usb1/power/autosuspend
echo auto >/sys/bus/usb/devices/usb1/power/level
echo 1 > /sys/module/snd_ac97_codec/parameters/power_save
echo 1500 > /proc/sys/vm/dirty_writeback_centisecs
```

修正 bug，使 compiz fusion 正常跑甚至加快一点

在 Section "Device"中加入以下代码：

```
sudo gedit /etc/X11/xorg.conf

Option "XAANoOffscreenPixmaps" "true"
Option "AGPSize" "32"
```

3.7 课后练习

1．体验开机过程。

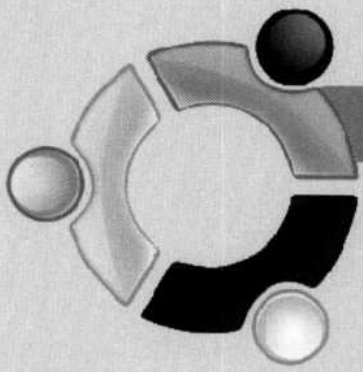
2．进入 Ubuntu 系统后，尝试换个自己最喜欢的桌面背景，设置屏保；尝试用 Ubuntu 自带的 firefox 上网浏览信息；尝试玩玩 Ubuntu 自带的一些小程序。

3．请尝试在 Ubuntu 系统上进行 X-Window 和 Shell 命令行的界面切换。

4．体验 Shell 命令 date 和 cal 的使用。

5．请在 Shell 环境下，尝试使用 shutdown、halt、poweroff 来关闭 Ubuntu；另外请分别用 shutdown 与 reboot 实现 Ubuntu 的重启，体会不同参数的妙处。

6．浏览 Ubuntu 帮助社区，从中找到适合自己的学习策略。



Part

02

Ubuntu 进阶应用

前一篇我们已经介绍了Linux基础、安装及体验等入门知识，下面我们将逐步深入地介绍Linux。本篇我们将重点从应用的角度，分别针对窗口化应用及命令行应用两个方向进行Ubuntu的介绍。具体包括X-Window的系统介绍、Ubuntu桌面体验及个性化设置、Ubuntu桌面应用、Ubuntu添加删除程序及软件包管理、Ubuntu Shell环境基础及入门，以及命令行编辑器Vim。

Chapter04 X-Window 介绍

X-Window 是类 Unix 系统中图形操作界面标准用语。因为微软抢先注册了 Windows 这个标准，所以所有类 Unix 系统中只能用 X-Window 称呼。GNOME 和 KDE 只是 Linux 系统中提供图形操作界面的两种解决方案，它们都使用 X-Window 这个标准来显示图形界面。在 Ubuntu 系统，默认的是 Gnome 桌面环境。在本章中，我们重点对以上概念进行介绍。

4.1 X-Window 基础

X–Window 是一个非常出色的图形系统，虽然它不是第一个为 Unix 而开发的视窗系统，但它却最流行。X–Window 设计非常巧妙，很多时候它在概念上比其他窗口系统更先进，因此经过这么多年，它仍然是工作站上的工业标准，其他窗口系统的许多概念都是从 X–Window 继承过来的。下面我们将逐步从 X–Window 的概念、X–Window 发展历程、X–Window 工作模型及实现原理等几个方面来剖析 X–Window。

4.1.1 什么是 X-Window

X–Window 是美国麻省理工学院(MIT)计算机科学研究室开发出来的一套计算机视窗系统，主要运行在 Unix 主机上（现在也有许多可以在其他操作系统上运行的 X–Window）。X–Window 通常也被叫做 X、“X 窗口”、“X Window 系统”和 X11 等。

X–Window 是一种用于 Unix 系统的标准 GUI (Graphical Device Interface，图形化用户界面)。GUI 是用鼠标和键盘控制的，具有下拉菜单、按钮、滚动条和为运行不同应用程序的重叠窗口界面；其他 GUI 环境的例子包括 Apple 的 Macintosh、Microsoft 的 Windows 和 IBM 的 O3/2 Presentation Manager。同时，X–Window 还是基于远程连接的客户机/服务器模式。

对于开发人员，X–Window 为开发基于图形的分布式应用程序提供软件工具和标准应用程序编程接口，完成的应用是与硬件无关的，这意味着它们可以在支持 X–Window 环境的任何系统上运行。因此，可以说 X–Window 也是一种控制用户怎样使用图形界面的协议簇（protocal）。

4.1.2 X 的发展历程

1984 年，由于研究项目需要，美国麻省理工学院 MIT 的 Bob Scheifler 与美国英文设备公司 DEC 的 Jim Gettys 一起合作要开发一套在 Unix 系统上运行的优良窗口系统。他们从斯坦福大学得到了一个名叫 W 的实验性窗口系统，于是便以该系统为基础开始发展，当进展到了足以和原来的系统有显著区别后，他们就把这个新系统称为 X。随后，新的版本不断诞生。从 1985 年开始，MIT 以外的其他人和组织开始加入 X 的发展。第一套商业 X 系统是 DEC (1986 年) 推出的 VAX Station–II/GPX，

之后是一连串版本的演进和完整协议的重新设计。1987 年 9 月，11 版推出。该版本历经变化，到目前为止，X11R6 被广泛使用。

严格地说，X 并不是特指某一个具体软件，而只是一种控制用户怎样使用图形界面的协议簇（protocals）。它规定了一个满足这种协议的产品应该具有什么样的功能（就如同 TCP/IP，DECnet 和 IBM 的 SNA，这些也都是协议，定义软件所应具备的功能）。任何软件系统只要能满足这些协议及 X 协会的其他相关规范，都可以被称为 X-Window 系统而无需理会其编码具体是如何实现的。实现 X 协议簇的软件很多，有商业软件，也有免费自由软件。在 Linux 的许多发布版中，主要采用 Xfree86，从名字可以知道这是一套自由软件，使用较多的版本是 3.3.2。除此以外，也有许多人喜欢使用 Accelerated X，但这是一套商业软件，目前使用较多的版本是 4.1。另外，在 Windows 95 和 Windows NT 下也有一些 X 软件，例如 MIX95 等。

1985 年，X-Window 组织制定了任何人只要付版权费便可使用 X 的授权许可。以下为一些重要的记事：

第 10 版发行：1985 年底。直到此时，MIT 以外的人和组织才开始对 X 有实质性的贡献。

DEC 于 1986 年 1 月推出第一套商业化的 X 产品 VAXstation-II/GPX。

第 10 版第 3 次发行：1986 年 2 月。从此时起，X 开始流传于世，人们把它移植到许多新的系统上。

第 10 版第 4 次发行：1986 年 11 月。

1987 年 1 月，在 MIT 举办第一次 X 技术会议。

在 1986 年，第 10 版 X 无法满足所有的需求已非常明显。MIT 和 DEC 开始从事于完整协议（protocal）的重新设计。这就是 X 第 11 版（所谓的 X11）。

第 11 版第 1 次发行：1987 年 9 月。

X 协会成立：MIT X 协会成立的目的是为了研究发展及控制标准。

第二次 X 技术会议：1988 年 1 月。

第 11 版第 2 次发行：1988 年 3 月。

第 11 版第 3 次发行：1988 年 10 月。

第 11 版第 4 次发行：1989 年 12 月。

第 11 版第 5 次发行：1991 年 9 月。

第 11 版第 6 次发行：1994 年 5 月。

第 11 版第 7 次发行：2005 年 12 月。

4.1.3 X 客户机/服务器模型

无论是哪一种 X 软件体系，一般由两大部分构成：即 X 服务器（X Server）和 X 客户机（X Client）。X 服务器是 X 系统的心脏，它直接控制系统内的显示设备，负责生成、删除视窗，可以控制键盘、鼠标以接收输入，能根据 X 应用软件的要求在窗口中做出写、画的动作。X 客户机就是那些显示窗口的程序。

客户机和服务器端不一定要运行在同一种操作系统上，它们甚至无需在同一种类型的计算机上运行。在 Microsoft 公司的 Windows 或 Apple 公司的 Mac OS 上运行 X Server 也是可以的，在它们

上面也有很多免费的和商业化的应用程序。

总之，X 服务器是 X 应用程序的基础，只有先运行 X 服务器，才能运行 X 应用程序。在这样的环境中，X 服务器和 X 客户机之间的通信就可以通过网络来进行。

Server（服务器）

Server 是控制显示器和输入设备（键盘和鼠标）的软件。Server 可以创建窗口，在窗口中画图形和写文字，回应 Client 程序的“请求”（requests）。但它不会自己动作，而是采用被动的方式，只有在 Client 程序提出需求后才开始响应动作。

每一套显示设备只对应一个唯一的 Server，而且 Server 一般由系统的供应商提供，用户通常无法修改。对于操作系统而言，Server 只是一个普通的用户程序而已，因此很容易更换新的版本，甚至是第三方提供的原始程序。

Client（客户机）

Client 是使用系统视窗功能的一些应用程序。在 X 下的应用程序称作 Client，原因是它是 Server 的客户，要求 Server 回应它的需求并完成特定的动作。

Client 无法直接影响视窗或显示，它们只能发送一个请求（request）给 Server，由 Server 来完成它们的请求。典型的请求通常是“在某个视窗中写‘Hello World’的字符串”，或者从 A 到 B 划一条直线。

Client 的功能大致可分为两部分：向 Server 发送“请求”和为用户执行程序，例如输入文字信息、计算等。通常，Client 程序执行用户请求的这一部分是和 X 独立的，它对于 X 几乎不需要知道什么。通常，应用程序（特别是大型的标准绘图软件、统计软件等）对许多输出设备具有输出的能力，而在 X-Window 中的显示只是 Client 程序许多输出格式中的一种，所以，Client 程序中和 X 相关的功能在整个程序中只占非常小的一部分。

用户可以通过不同的途径使用 Client 程序：通过系统提供的程序来使用；使用来自第三方的软件；或者是用户为了某种特殊应用而自己编写的 Client 程序。

第一次接触 X-Window 系统的用户很容易混淆 X-Window 系统中的客户机/服务器的概念，他们会认为 X-Window 下的客户机/服务器的概念与普通网络中的客户机/服务器的概念相同。在通常的观念中，用户在客户工作站上，使用远程服务器提供的文件或显示服务；而在 X-Window 下，用户使用 X 服务器进行操作，而客户程序可以运行在本地或者远程计算机上。

在 X-Window 下，服务器资源为 X 服务器的显示提供处理能力；X 客户端要显示图形图像，不能直接控制显示硬件，只能使用用户面前的 X 服务器提供的显示资源。同样，它也不能接收用户输入，也只能使用 X 服务器控制的键盘或鼠标资源来接收输入。在这里，X 服务器是硬件的控制者，X 客户机只是单纯地执行程序，它只能使用 X 服务器提供的服务进行输入输出。

X 服务器（X Server）是一个管理显示的进程，必须运行在一个有图形显示能力的计算机上。理论上，一台电脑上可以同时运行多个 X 服务器，每个 X 服务器能管理多个与之相连的显示设备。

X 客户机（X Client）是一个使用 X 服务器显示其数据的程序，它可以运行在与 X 服务器不同的电脑上。

X 协议（X Protocol）是 X 客户机和服务器进行通信的一套协定。X 协议支持网络，能在本地系统中和网络实现这个协议，支持的网络协议有 TCP/IP 和 DECnet 等。

X 的这种任务划分有几个优点：

- 客户端程序可以在远程计算机上执行计算任务，而 X 服务器仅负责复杂的图形显示，充分发挥 X 服务器在显示上的优势。
- 只有 X 服务器与硬件打交道，所有的客户端程序都与硬件无关，这很容易在不同的平台上移植。
- 客户端程序可以在不同的计算机上运行，从巨型计算机到个人计算机，从而充分发挥网络计算的优越性。
- 尽管每个 X 客户程序都可以对整个屏幕范围进行显示操作，但标准的做法是先创建一个显示窗口，此后客户端程序的所有显示都相对于这个窗口进行操作。这样，在同一屏幕内就能同时显示多个独立的客户端程序，通过对窗口的管理，可在不同的程序之间进行切换。每个窗口应该位于屏幕的哪个位置、何时显示、何时隐藏，以及视窗的标题、四周如何显示等，这些都不应该由产生这个窗口的客户端程序自己控制和维护，否则就不能达到简化设计的目的。X-Window 并没有自己实现这些任务，在 X 的设计原则中，这些代表 GUI 风格的任务仍然是客户端程序的任务，因此 X 将管理窗口的任务交给一个特殊的客户端程序——窗口管理器，使用不同的窗口管理器会使 X-Window 的外观看起来截然不同。
- X 系统只负责显示图形，并不限制显示和操作的风格，因此不同 X-Window 的风格并不相同，用户可以根据自己的喜好进行选择。
- 在 X-Window 上，所有窗口形成一个树状结构。X-Window 的窗口管理器运行在根窗口上，其他所有窗口为根窗口的子窗口；而其他窗口上也有相应的按钮、对话框等组件，这些又是它的子窗口。
- 由于 X-Window 系统只提供了最基本的系统调用，而具体的窗口都有很多共性，因此要开发 X 应用程序，应该首先使用开发工具包，而没有必要直接使用最基本的 X-Window 的系统调用，以简化编写程序的工作。不同的公司或组织开发了各种工具包来提供创建和管理具体窗口的组件，例如 Motif 套件、OpenLook 套件等。每种套件都提供了菜单、按钮、对话框等图形接口的标准组件，还提供基本的窗口管理器。使用不同套件开发的程序，其显示风格也不相同，因此就在不同程序之间形成了不同的 GUI 风格。Motif 和 OpenLook 就属于两种不同的图形接口风格，当前 Motif 成为商业 X-Window 的一个标准，很多商业软件均基于 Motif 进行开发，而 OpenLook 则没有获得更普遍的支持。

通信通道

有了 Server 和 Client，它们之间就要传输一些信息，这种传输信息的媒介就是我们所要介绍的 X 的第三个组成部件：通信通道。凭借这个通道，Client 发送“请求”给 Server，而 Server 回传状态（status）及其他一些信息给 Client。

Client 是通过函数库来使用通信通道的。在系统或网络上，支持通信形态需求的是内建于系统的基本的 X-Window 函数库（library）。只要 Client 程序利用了函数库，自然就有能力使用所有可用的通信方法。这时通道本身就变得不再重要了，而只是一个概念而已。

Server 和 Client 通信的方法大致有两类，对应于 X 系统的两种基本操作模式。

第一类，Server 和 Client 在同一台机器上执行。它们可以共同使用机器上任何可用的通信方法做互动式信息处理。在这种模式下，X 可以同其他传统的窗口系统一样高效工作。

第二类，Client 在一部机器上运行，而显示器和 Server 则在另一部机器上运行。因此两者的信息交换就必须通过彼此都遵守的网络协议进行，最常用的协议为 TCP/IP。这种通信方式一般被称为网络透明性，这几乎也是 X 独一无二的特性。

4.1.4 X 的窗口管理器及其原理

X 不是内置于操作系统的，它只是比用户层次稍高一些，在系统中也是一个相对独立的组件。这样做有如下优点：

(1) 易于安装和改版，甚至删除。这种工作不需要重启系统，也不会对其他应用程序造成干扰。

(2) 第三方很容易支持并加强它的功能。例如，假设制造厂商提供的系统不够好，用户可以向别人买更好或更快的版本。

(3) X 不会制定操作系统，因此 X 成为一种标准，这也是第三方发展软件的原动力。

(4) 在 Server 上进行工作时，如果程序异常中断，只会影响到窗口系统，不会造成机器的损坏或操作系统内核的破坏。

X 的设计目标之一就是能创建许多不同形式的用户接口。其他窗口系统提供具体的交互方法，而 X 只提供一般的架构，让系统创建者确定所需的交互风格。这种特性使得开发者可以在 X 的基础上创建全新的接口，并且可以在任何时刻根据自己的需要选用适当的接口。

一般来说，用户接口可以分为两部分：管理接口和应用接口。管理接口也就是窗口管理器，是命令的最高层，它负责在屏幕上建构或重建窗口，改变窗口的大小、位置，或者将窗口改变成图标等。应用接口确定了用户和应用程序之间的交互风格，即用户如何利用窗口系统的设备程序来控制应用程序并向它传输数据，例如，如何用鼠标来选定一个选项。

X 窗口管理器

X 的设计理念很像 Unix 的设计理念——tools，not policy。这就意味着 X 不会试图去规定应该如何去完成任务，而是只给用户提供一些工具，至于如何使用这些工具是用户自己的事情。

这套理念扩展了 X，它不会规定窗口在屏幕上应该是什么样子，要如何移动鼠标，应该用什么键来切换窗体（比如，Alt+Tab 键在 Microsoft Windows 环境中的作用），每个窗口的工具条应该看起来像什么，它们是否应该有关闭按钮等。

实际上，X 行使了一种叫做“窗口管理器”的应用程序的职责。有很多这样的程序可用：AfterStep，Blackbox，ctwm，Enlightenment，fvwm，Sawfish，twm，Window Maker 等。每一个窗口管理器都提供了不同的界面和观感，其中一些还支持“虚拟桌面”；有一些允许用户定制一些键来管理桌面；一些有“开始”按钮，或者其他类似的设计；一些是“可定制主题的（themeable）”，通过安装新的主题，可以完全改变外观。这些以及很多其他的窗口管理器，都可以在 Ports Collection 的 x11-wm 分类目录里找到。

每个窗口管理器也有不同的配置机制，有些需要手工配置文件，而另外一些则可以使用 GUI 工具来完成大部分的配置任务，举例而言，Sawfish 就使用 Lisp 语言书写配置文件。

（1）焦点策略

窗口管理器的另一个特性是鼠标的焦点策略（focus policy）。每个窗口系统都需要有一个选择窗口的方法来接收键盘的输入信息，以及当前哪个窗口处于可用状态。

通常比较熟悉的是一个叫做 click-to-focus 的焦点策略。这是 Microsoft Windows 使用的典型焦点策略，也就是在一个窗口上单击一下鼠标，这个窗口就处于当前可用的状态。

X 不支持一些特殊的焦点策略。确切地说，窗口管理器控制着在什么时候哪个窗口拥有焦点。不同的窗口管理器支持不同的焦点方案，它们都支持单击即获得焦点，而且它们中的大多数都支持好几种方案。

最流行的焦点策略有以下几种：

- focus-follows-mouse

  鼠标所在的窗口就是获得焦点的窗口。这个窗口不一定位于其他窗口之上。通过将鼠标移到另一个窗口上就可以改变焦点，而不需要在它上面单击。
- sloppy-focus

  这种方式是对 focus-follows-mouse 策略的一个小小扩展。对于 focus-follows-mouse，如果把鼠标移到了根窗口（或桌面背景）上，则其他所有窗口都会失去焦点，而相关的全部键盘输入也会丢失。如果选择了 sloppy-focus，则只有当指针进入新窗口时，窗口焦点才会发生变化，退出当前窗口时是不会变化的。
- click-to-focus

  当前窗口由鼠标单击来选择。窗口被“突出显示”，出现在所有其他窗口的前面。即使指针被移向了另一个窗口，所有的键盘输入仍会被这个窗口接收。

各窗口管理器支持不同的策略，您可以查看具体窗口管理器的文档。

（2）窗口部件

Microsoft 公司的 Windows 和苹果公司的 Mac OS 都有一个严格的窗口部件策略。应用程序开发者被建议确保他们的应用程序共享一个普通的所见即所得的用户界面。对于 X，它并不要求一个特殊的图形风格或一套完整的窗口部件集。

这样的结果是您不能期望 X 应用程序只拥有一个普通的所见即所得的界面。有很多的流行的窗口部件集设置，包括来自于 MIT 的 Athena、Motif®（模仿 Microsoft Windows 的窗口风格，所有部件都具有斜边和三种灰度）和 OpenLook 等。

如今，绝大多数比较新的 X 应用程序采用一组新式的窗口设计，这包括 KDE 所使用的 Qt，以及 GNOME 所使用的 GTK+。在这样一种窗口系统下，Unix 桌面的窗口部件统一起来，以使初学者感到更容易一些。

X 的实现原理

一个 X-Window 系统中可以包含多个 Screen，而 Screen 则是一个实际的 Monitor 或是 Device，每个 Display 则可以包含多个 Screen。Display 指的是一群 Screen 和一个 Pointer（一般是鼠标）加上一个键盘的集合。

（1）Event 和 Request

当 Client 需要在显示幕上打开一个 window 或是显示一段文字或图形时，Client（即 X 下的应用程序）向 Server 提出一个 Request，Server 收到 Request 之后，则依据 Request 的内容提供相对应的服务。当用户在属于某一 Client 的 window 内时，单击鼠标右键，或是敲了键盘上的一个键，则 Server 会发送一个 Event 给该 Client。而 Client 只要依据 Server 送回来的 Event，即可判断有何事件发生，然后依据所发生的 Event 进行相对应的动作。

（2）X-Window 的网络特性

Client 和 Server 不一定要在同一部电脑上执行，它们可以是接在网络上的两台电脑。当 Client 要向 Server 发出 Request 时，可以通过网络向运行 Server 的机器发出 Request。同样的，当 Server 要向 Client 通知 Event 的发生时，也可以通过网络向 Client 传送。因此你可以在 A 地的电脑上执行某一 X-Window 应用程序，而在 B 地通过 X Server 观察程序执行的结果。

一个 X-Window 程序，也就是 Client，在同一时间可以和一个以上的 Server 沟通，也可以同时和多个 Server 建立连接，在多个 Server 上显示，并接收多个 Server 的信息。同样，X Server 也可以在同一时间内，接收多个 Client 送来的信息，并做对应的处理。

（3）Window 的层级关系

X-Window 内的 window 是有层级关系的，这种层级关系可以画成一个树状图。在树状图的最上层是 rootwindow，这个 window 是 Server 所特有的。除了 rootwindow 之外，每个 window 都有一个 parentwindow，即树状图上的上一层 window。在树状图的下一层 window 称为 childwindow。每个 window 可以有多个 child，但不一定每个 window 都有 child。childwindow 下面还可以有 child。

在显示幕上，childwindow 永远在 parentwindow 上面（即 parent 被 child 盖住）。而 childwindow 可以比 parentwindow 大，但超过 parent 的部分，会被截掉而不显示在显示幕上。

当显示幕上有多个 window 存在时，可能会发生 window 重叠的现象。下面的 window 会被上面的 window 盖掉，当上面的 window 移离目前位置，而使原本被盖住的部分重新露出来，这时就必须重画露出来的部分。但 X 并不保证会保存原本显示在 window 上的数据。因此这时 Client 就必须负起重画的责任了。当 X Server 没保存重新露出来的部分的数据时，X Server 会送出 ExposeEvent 给 Client，通知 Client 哪些部分需要重画。然后 Client 就必须决定是否要重画，或是采取其他的处理方式。

（4）X-Window 的外观

在很多的 GUI 中，如 OS/2 等，都会提供 Button、ScrollBar、Menu 等基本组件，以方便建构程序的外观。但在 X 中并不提供这样的组件，相反的，X-Window 提供的是一套建立外观风格的机制。利用这套机制，用户可以建立各种不同风格的组件。这样用户就不必限制在系统内定的风格中而无法改变。

（5）Window Manager

在 X 中有一种极为特殊的程序，称为 Window Manager，其作用是管理 Desktop 上的各应用程序的 window，并提供特殊风格的 window 外观。不同的 Window Manager 有不同的外观风格。因此，我们可以更换不同的 Window Manager，以得到不同风格的 window 外观。而一般的 Client 则必须和 Window Manager 合作，以得到良好而风格一致的 window。Window Manager 也是一种 Client 程序，

只是 Window Manager 负有特别的任务——管理 Desktop 和 window 外观。

为了得到良好而风格一致的 window，Client 必须和 Window Manager 合作。其合作方式是通过由 Client 传送提示给 Window Manager，Window Manager 则会根据提示的内容给予应用程序 window 适当的外观，但 Window Manager 不一定会依据提示进行处理，仅仅参考。

4.1.5 X11R7 介绍

上一节从原理上介绍了 X-Window 的基础，下面我们对当前应用最广泛的版本 X11 的安装、应用等方面进行简单介绍。其实，目前 Ubuntu 使用的 X Server 就是 X11R7，它于 2005 年 12 月发布，与之前的 X11R6.9 实际上具有相同的源代码（Source Code）。不过 X11R7 的模块化设计可提高开发效率。

X11 包括 Xorg 和 XFree86™(以及一些其他这里没有讨论的软件包)。XFree86 是一种由 XFree86 Project，Inc.发布的 X11 服务。从 FreeBSD 5.3-RELEASE 开始，Xorg 成为官方所支持的 X11 的默认实现，它是由 X.Org 基金会开发的 X11 服务，采用与 FreeBSD 类似的授权。此外，也有一些用于 FreeBSD 的商业 X 服务器。

与 X11R7 有关的软件，大多放在/usr 及其子目录中。以下是较为重要的目录的说明。

/usr/bin：存放 X Server 和不同的 X Client。
/usr/include：开发 X Client 和图形所需的文件路径。
/usr/lib：X Server 和 X Clients 所需的函数库目录。
/usr/lib/X11：保存多项资源，如字体和文件等。
/usr/lib/xorg/modules：包含驱动程序与多种 X Server 模块。
/usr/X11/man：保存 X11 程序编写时的说明手册。
/etc/X11/xorg.conf：保存主要配置文件。

在安装时，如果没有设置 X-Window 系统，之后必须先设置鼠标、键盘、显示器以及显卡等，这样才能成功启用 X-Window 系统，而这些设置都记录在/etc/X11/xorg.conf 文件中，因此这个文件的重要性可见一斑。

/etc/X11/xorg.conf 由数个 Section/EndSection 的区块组成，而每个区块的格式如下：

```
Section "Section 名称"
    选项名称    "选项值"
    选项名称    "选项值"
    选项名称    "选项值"
    …
EndSection
```

下面将说明/etc/X11/xorg.conf 文件中使用的 Section 类型及每个类型可用的选项名称和选项值。

（1）ServerLayout

ServerLayout Section 主要用于建立 X Server 启动时的外观，如果文件中包含多个 ServerLayout Section，则默认使用第一个 ServerLayout Section 的设置。

以下是此区块的系统默认值，以及可供使用的选项说明：

```
Section "ServerLayout"
    Identifier  "Default Layout"
    Screen "Default Screen"
    InputDevice  "Generic Keyboard"
    InputDevice  "Configured Mouse"
    InputDevice  "stylus" "SendCoreEvents"
    InputDevice  "cursor" "SendCoreEvents"
    InputDevice  "eraser" "SendCoreEvents"
EndSection
```

- Identifier：该项是 ServerLayout Section 的唯一名称。
- Screen：Section 指定名称。
- InputDevice：在 X Server 中的 InputDevice Section 名称。通常在此仅有两行设置，也就是系统中的第一个鼠标和第一个键盘，而其他的设备大多可以忽略。

（2）Files

Files Section 用于设置 X Server 服务的路径，如字体和颜色。以下是此区块的系统默认值，以及可供使用的选项说明：

```
Section "Files"
    FontPath "/usr/share/X11/fonts/misc"
    FontPath "/usr/share/X11/fonts/cyrillic"
    FontPath "/usr/share/X11/fonts/100dpi/:unscaled"
    FontPath "/usr/share/X11/fonts/75dpi/:unscaled"
    FontPath "/usr/share/X11/fonts/Type1"
    FontPath "/usr/share/X11/fonts/100dpi"
    FontPath "/usr/share/X11/fonts/75dpi"
    FontPath "/usr/share/fonts/X11/misc"
    # path to defoma fonts
    FontPath "/var/lib/defoma/x-ttcidfont-conf.d/dirs/TrueType"
EndSection
```

- RgbPath：RGB 数据库的路径。这个文件定义 X 中所有有效颜色的名称，并且指定数值。
- FontPath：设置 X Server 寻找字体时的路径。可以同时使用多个路径，但需用逗号隔开。

（3）Module

Module Section 主要用来告诉 X Server 应加载哪些模块。这些模块可以提供额外的服务功能，一般并不需要更改此处的值。此处使用的唯一选项为 Load，它可用来加载模块。以下是此区块的系统默认值：

```
Section "Module"
    Load "i2c"
    Load "bitmap"
    Load "ddc"
    Load "dri"
    Load "extmod"
    Load "freetype"
    Load "glx"
```

```
    Load "int10"
    Load "type1"
    Load "vbe"
EndSection
```

(4) InputDevice

InputDevice Section 用于设置鼠标或键盘等输入设备，以便通过 X Server 提供信息给 Linux 系统，多数系统都存在至少两个 InputDevice Section（鼠标和键盘）。以下是此区块的系统默认值，以及可供使用的选项说明：

```
Section "InputDevice"
    Identifier "Generic Keyboard"
    Driver "kbd"
    Option  "CoreKeyboard"
    Option  "XkbRules" "xorg"
    Option  "XkbModel" "pc105"
    Option  "XkbLayout" "us"
    Option  "XkbOptions" "lv3: ralt_switch"
EndSection
Section "InputDevice"
    Identifier "Configured Mouse"
    Driver "mouse"
    Option  "CorePointer"
    Option  "Device"  "/dev/input/mice"
    Option  "Protocol"  "ExplorerPS/2"
    Option  "ZAxisMapping"  "4 5"
    Option  "Emulate3Buttons" "true"
EndSection
Section "InputDevice"
    Driver "wacom"
    Identifier  "stylus"
    Option "Device" "/dev/wacom"  # Change to
    # /dev/input/event
    # for USB
    Option "Type" "stylus"
    Option "ForceDevice" "ISDV4"   # Tablet PC ONLY
EndSection
Section "InputDevice"
    Driver "wacom"
    Identifier  "eraser"
    Option "Device" "/dev/wacom"  # Change to
    # /dev/input/event
    # for USB
    Option "Type" "eraser"
    Option "ForceDevice" "ISDV4" # Tablet PC ONLY
EndSection
Section "InputDevice"
    Driver "wacom"
    Identifier  "cursor"
    Option "Device" "/dev/wacom"  # Change to
```

```
    # /dev/input/event
    # for USB
    Option "Type" "cursor"
    Option "ForceDevice" "ISDV4"   # Tablet PC ONLY
EndSection
```

- Identifier：该项用于设置设备的名称。通常，这些名称的后面都会加上一个数字，且第一个设备的数字为 0。例如，第一个键盘的 Identifier 为 Keyboard0。
- Driver：该项用于告诉 X Server 应该从哪里加载驱动程序。
- 在大多数 InputDevice Section 中，尚有为数不多以 Option 为首的选项，并且包含特定的选项值。如果要启用这些选项功能，只要将每行开头的注释符号“#”去除即可。

（5）Monitor

Monitor Section 用于设置系统使用的显示器类型，设置此处选项时应特别留意，因为不适当的设置可能会给显示器造成损害。以下是此区块的系统默认值，以及可供使用的选项说明：

```
Section "Monitor"
    Identifier "Generic Monitor"
    Option "DPMS"
    HorizSync   28-51
    VertRefresh   43-60
EndSection
```

- Identifier：该项是显示器的唯一名称。在这些名称后面都会加上一个数字，第一个显示器的代表数字为 0（如 Monitor0）。
- VendorName：该项是显示器制造商名称。
- ModelName：该项是显示器类型名称。
- HorizSync：该项是与显示器兼容的水平刷新频率范围，其单位为 kHz。这个设置值会同时指出是否在此显示器中使用特定的 Modeline 值。
- VertRefresh：与显示器兼容的垂直刷新频率范围，其单位为 kHz。这个设置值会同时指出是否在此显示器中使用特定的 Modeline 值。

（6）Device

Device Section 用于设置显卡的信息内容，在此文件中至少需要包含一个以上的 Device Section。如果系统中包含多张显卡，或一张显卡上有多种设置值，则可以使用多个 Device Section 设置。以下是此区块的系统默认值，以及可供使用的选项说明：

```
Section "Device"
    Identifier  "VMWare Inc [VMware SVGA II] PCI Display Adapter"
    Driver "vmware"
    BusID "PCI: 0: 15: 0"
EndSection
```

- Identifier：该项是显卡的唯一名称。
- Driver：该项用来告诉 X Server 应从何处加载显卡的驱动程序。
- VendorName：该项是显卡制造商名称。
- BoardName：该项是显卡类型名称。

- BusID：该项是显卡的总线位置，这个选项适用于多显卡环境。

（7）Screen

Screen Section 合并了 Device 和 Monitor 的部分设置，以便能够形成成对的设置内容。在此文件中至少需要包含一个以上的 Screen Section。以下是此区块的系统默认值，以及可供使用的选项说明：

```
Section "Screen"
    Identifier  "Default Screen"
    Device "VMWare Inc [VMware SVGA II] PCI Display Adapter"
    Monitor  "Generic Monitor"
    DefaultDepth  24
    SubSection "Display"
        Depth  1
        Modes  "1024x768" "800x600" "640x480"
    EndSubSection
    SubSection  "Display"
        Depth  4
        Modes   "1024x768" "800x600" "640x480"
        EndSubSection
        SubSection "  Display"
        Depth  8
        Modes   "1024x768" "800x600" "640x480"
    EndSubSection
    SubSection  "Display"
        Depth  15
        Modes   "1024x768" "800x600" "640x480"
    EndSubSection
    SubSection  "Display"
        Depth  16
        Modes   "1024x768" "800x600" "640x480"
    EndSubSection
    SubSection  "Display"
        Depth  24
        Modes   "1024x768" "800x600" "640x480"
        EndSubSection
EndSection
```

- Identifier：该项定义一个 Screen 名称，以便在 ServerLayout Section 中进行参照。
- Device：该项指定 Device Section 中的名称。
- Monitor：该项指定 Monitor Section 中的名称。
- DefaultDepth：该项是默认的色深（Color Depth）位数。

（8）DRI

DRI（Direct Rendering Infrastructure）是一种接口，它让三维软件可以使用新型显示设备的三维硬件加速功能。除此之外，DRI 还能改善二维硬件加速的性能。但通常并不使用这个选项功能，除非在 Module Section 中打开 DRI 设置。以下是此区块的系统默认值。

```
Section "DRI"
    Mode 0666
EndSection
```

4.2 集成式桌面环境 KDE

习惯使用微软操作系统的用户可能没有想过，为什么每个人的快捷菜单、开始菜单、所有程序以及其他的系统设置都是如出一辙呢？能不能随着用户的喜好进行修改呢？的确，微软系统使用单一的图形界面，用户并无法更改它。而 Linux 在这方面就具有较大弹性，用户可以依其喜好随时改变图形界面。这也就是所谓的“集成式桌面环境”。当前主要两个“集成式桌面环境”是 GNOME 和 KDE。

4.2.1 集成式桌面环境简介

顾名思义，集成式桌面环境也是与窗口管理有关的软件，但是和窗口管理程序最大的不同之处是它的适用范围。X-Window 管理程序仅能负责桌面管理的工作，例如打开窗口、菜单、更改窗口大小或移动窗口等，而这些功能只是集成式桌面环境的一部分。

集成式桌面环境除了可以启动 X-Window 管理程序，更重要的作用是充当应用程序的统一界面。通常，在集成式桌面环境中都会包含许多工具程序，包括系统设置、文字处理、多媒体或网络的相关软件。

因为所有的窗口管理程序都包含在集成式桌面环境中，所以在更改了不同的窗口管理程序后，起初会看到窗口管理程序的桌面内容，但是在加载了集成式桌面环境后，桌面又会回到原来集成式桌面环境设置的桌面内容。也就是说，无论在集成式桌面环境中使用哪一种窗口管理程序，已设置的桌面环境都不会更改。

4.2.2 KDE 概述与发展历程

KDE（K Desktop Environment）是最早用于 Linux 系统的 GUI，最初由于商业化运作，没有广泛推广。自 1996 年 10 月 Matthias Ettrich 开发完成到现在，KDE 已成为 Linux 两大集成式桌面环境之一。

KDE 是一个用于 Unix 工作站的网络透明的现代化桌面环境。KDE 为满足在 Unix 工作站上的易用桌面的需求而不断探索，期望有 Mac OS 和微软的 Windows 那样的桌面环境。我们相信 Unix 操作系统是当今可用的最好的操作系统。实际上，在这些年来 Unix 已经成为信息技术专业人员无可争议的选择，当提到稳定性、可扩展性和开放性，Unix 目前是无与伦比的。但无论如何，在 Unix 上缺乏易于使用的现代化桌面环境，这已成为 Unix 成为办公和家用普通计算机用户的桌面系统的重大阻碍。Unix 在服务器市场占有优势，并且是计算机专业人士和科学领域中的首选计算平台，没有 Unix，就没有互联网。但是 Unix 也从事于满足普通计算机用户的需求。自从大量的类 Unix（如 Debian GNU/Linux、FreeBSD 和 NetBSD 等，这几个平台都具有非凡的品质和稳定性）在互联网上自由可用的时候，这种情况更加令人遗憾。

KDE 原本是使用 QT library（商业版权）发展，而 GNOME 却是使用自行开发的 GTK（开放源代码），因此在刚开始时，KDE 无法大量推广。直到 QT library 决定将其版权变为 QPL（Q Public License）之后，KDE 才被用于一般的非商业领域。

下面是KDE的发展历程。

- KDE项目始建于1996年10月，确切的公布日期是1996年10月14日。
- 1997年8月15日，KDE第一次代表会议在德国阿恩斯伯格市召开，共15人参加。
- 1997年12月，KDE协会创建，这是一个为在法律和财政上保护核心成员，避免相关纠纷而设立的组织。
- 1998年4月8日，KDE Free Qt基金会成立。
- 1998年4月19日，KDE 1.0发布。
- 2000年10月23日，KDE 2.0发布。
- 2002年4月3日，KDE 3.0发布。
- 2003年1月28日，KDE 3.1发布。
- 2004年2月3日，KDE 3.2发布。
- 2004年8月19日，KDE 3.3发布。
- 2005年3月16日，KDE 3.4发布。
- 2005年11月29日，KDE 3.5发布。
- 2008年1月11日，革命版本——KDE 4.0发布。

KDE是一个规模宏大的项目，我们很难用数字去量化它的实质，不过您可以从下面的数据中看看KDE的现状。

- 现在的KDE SVN代码仓库已经存储了超过400万行的代码（作为比较，Linux内核2.5.17版的代码量是370万行左右）。
- 超过800名贡献者在协助进行KDE的开发。
- 独立的翻译小组大约有300人。
- 仅在2002年5月间，据统计就有11 014次CVS代码提交。
- KDE在12个国家有17个以上的官方WWW镜像。
- KDE在39个国家有106个以上的官方FTP镜像。

4.2.3 KDE桌面环境及框架

KDE现在是Linux上可用的、易于使用的现代桌面环境。Unix/KDE和一些如GNU/Linux这样的自由的类Unix一起，组成了一个对于任何人都可用的完全自由和开放的计算平台，而且完全免费，任何人都可以修改它的源代码。当然它总是有可以改进的空间，相信开发者已经发布了一些当今可用的、能和商业操作系统/桌面整合的、合适的替代品。希望Unix/KDE组合最终能为普通计算机用户带来一个同样开放、可靠、稳定和专利自由的计算机环境。

KDE应用程序开发框架

以前在Unix/X11下开发应用程序是一个非常困难并且单调乏味的过程。KDE认识到了在一个计算平台上，平台和对于这个特定平台用户可用的一流应用程序的集合是同等重要的。从这些观点出发，KDE开发者已经开发了一流的复合文档应用程序框架，实现了最先进的框架技术，并且因此能与诸如微软的MFC/COM/ActiveX技术等流行开发框架相竞争。KDE的KParts复合文档技术，使

得开发人员可以快速地创建一流的应用程序，以实现最尖端的技术。

KDE 办公应用套件

因为 KDE 应用程序开发框架的优势，已经有大量的应用程序存在于 KDE 桌面环境下了。KDE 的基本发行版中可以选择使用这些程序。现在 KDE 也拥有了一个基于 KDE 的 KParts 技术的，由电子表格、幻灯片制作程序、组织者、新闻客户端和更多应用组成的办公应用套件。KDE 的幻灯片制作程序 KPresenter，已被成功用于多次演示。

4.2.4 KDE 的优缺点

传统的 X11 桌面对用户来说有以下不足之处。

- 没有一个基于桌面配置的简易的人机对话机制。
- 没有统一的应用程序帮助系统。
- 没有通用的应用程序开发框架。
- 没有一个复合文档框架。
- 缺少应用程序级的网络透明性。
- 编写 X11 程序非常困难、枯燥。

而对用户来说，KDE 会给您带来以下好处。

- 一个美观的现代化桌面。
- 一个具有完整网络透明性的桌面。
- 一个方便的集成帮助系统，它可以对 KDE 桌面及其应用程序帮助提供相同的访问途径。
- 所有的 KDE 应用程序都具有统一的视觉观感。
- 标准化的菜单、工具栏、键盘绑定、颜色样式等。
- 国际化支持，KDE 已拥有 60 余种语言的翻译版本。
- 统一的对话框系统，由具体的桌面配置来操作。
- 大量优秀的 KDE 应用程序。

4.2.5 KDE 组件说明

现在的官方 KDE 发行版包含了以下组件包。

- aRts：实时模拟音频合成器与声音服务器。
- KDE-Libs：一组必需的基本运行库。
- KDE-Base：KDE 的基本组件（窗口管理器、桌面、面板、文件管理器与网络浏览器 Konqueror 等）。
- KDE-Network：新闻组阅读器 KNode、新闻采集器 KNewsticker、拨号工具 Kppp 等。
- KDE-Pim：电子邮件客户端 KMail、地址簿管理器 KAddressbook、日程管理器 KOrganizer、Palm 同步前端 KPilot 等。

- KDE—Graphics：一组图形图像相关程序，如 DVI 文档查看器 KDVI、PostScript 查看器 KGhostView、绘图程序 KolourPaint、传真查看器 KFax 等。
- KDE—Multimedia：音频播放器 Noatun、MIDI 演奏器 KMidi、CD 播放器 KSCD 等。
- KDE—Accessibility：为生理上有残疾的用户设计的辅助工具。
- KDE—Utilities：文本编辑器 KEdit、计算器 KCalc、十六进制编辑器 KHexEdit、笔记工具 KJots 等。
- KDE—Edu：一组教学相关的程序。
- KDE—Games：空间射击游戏 KAsteroids、纸牌系列合集 KPat、俄罗斯方块 KTetris 等。
- KDE—Toys：娱乐小配件。
- KDE—Addons：提供给 Konqueror、Kate、Kicker、Noatun 等程序的插件合集。
- KDE—Artwork：附赠的图标、样式、壁纸、屏幕保护以及窗口装饰的集合。
- KDE—Admin：一些用于系统管理的工具。
- KDE—SDK：一组用于简单的 KDE 程序开发的脚本和工具包。
- KOffice：集成化办公套件。
- KDevelop：适用于 C/C++的集成化开发环境。
- KDE—Bindings：提供对若干种编程语言（Python、Ruby、Perl、Java 等）的绑定。
- KDEWebdev：Web 开发工具。

另外，有两个名义上的软件包并不属于官方 KDE 发行版的一部分，但它们也隶属于整个 KDE 项目之下。

- KDE—Extragear：所谓的 Extragear，是一系列和 KDE 项目有关的 KDE 软件集合，它们出于某种理由而不属于核心 KDE 发行版，但依然属于 KDE 项目。对翻译者和文档撰写者来说，其公示效应比其他第三方软件都更高。
- KDE—Playground：Playground 和 Extragear 非常接近，都是不属于核心 KDE 发行版但属于 KDE 项目。

最后，还有数以千计的优秀 KDE 软件留存于世，尽管它们不属于 KDE 项目官方管辖，但用户还是可以在 KDE 的软件中心里找到它们。

4.3 集成式桌面环境 GNOME

GNOME（The GNU Network Object Model Environment，GNU 网络对象模型环境），属于 GNU（GNU's Not Unix）计划的一部分，是开放源码运动的一个重要组成部分，是一种让使用者容易操作和设定计算机环境的工具。GNOME 的目标是基于自由软件，为 Unix 或者类 Unix 操作系统构造一个功能完善、操作简单、界面友好的桌面环境，它是 GNU 计划的正式桌面。

4.3.1 GNOME起源及目标

起源

GNOME计划是1997年8月由墨西哥的Miguel de Icaza和Federico Mena发起，是作为KDE的替代品。当时，为了克服KDE所遇到的QT许可协议和单一C++依赖的困难，以Miguel de Icaza为首的250名程序员就开始了一个新项目，这就是GNOME。

经过14个月的共同努力，终于完成了这个工程。现在GNOME已得到了占Linux市场份额最大的发行商Red Hat的支持，拥有了大量应用软件，包括文字处理软件Go、电子表格软件Gnumeric、日历程序GNOMEcal、堪与PhotoShop媲美的图形图像处理软件Gimp等。

现在GNOME与KDE成为两大竞争阵营，它们必将使得Linux更加易于使用。

在GNOME变得实用和普及之后，2000年9月Trolltech在GNU GPL和QPL（去掉了大多数争论多年的内容）双重许可证下发布了GNU/Linux版的QT库。但是QT的许可证还是有争议，因为GPL用于库时，对与之链接的代码——例如的KDE框架和任何为其编写的程序——都施加了许可证限制。

后来，GIMP Toolkit（GTK+）被选中作为QT Toolkit的替代，担当GNOME桌面的基石。GTK+使用GNU宽通用公共许可证（LGPL，一个自由软件许可证），允许链接到它的软件，例如GNOME的应用程序，使用任意的许可证。此时，GNOME桌面的库使用LGPL，而GNOME计划内的应用程序使用GPL许可证。

GNOME桌面系统使用C语言编程，但也存在一些其他语言的绑定，使得能够使用其他语言编写GNOME应用程序，例如C++、Java、Ruby、C#、ython和Perl等。

目标计划

GNOME计划提供了两个东西：一是GNOME桌面环境，一个对最终用户来说符合习惯并十分吸引人的桌面；二是GNOME开发平台，一个能使开发的应用程序与桌面其他部分集成的可扩展框架。

GNOME桌面主张简单、易用和恰到好处，因此GNOME开发时有两点很突出：一是可用性——设计和建立为所有人使用的桌面和应用程序，不论其技术技巧和是否身体残疾；二是国际化——保证桌面和应用程序可以用于很多语言。

组织方式

和大多数自由软件类似，GNOME组织也很松散，其关于开发的讨论遍布于众多向任何人开发的邮件列表。为了便于管理、扩大影响以及与对开发GNOME软件有兴趣的公司联系，2000年8月成立了GNOME基金会。基金会并不直接参与技术决策，而是协调发布和决定哪些对象应该成为GNOME的组成部分。基金会网站将其成员资格定义为："按照GNOME基金会章程，任何对GNOME有贡献者都可能是合格的成员。尽管很难精确定义，贡献者一般必须对GNOME计划有不小帮助。其贡献形式包括代码、文档、翻译、计划范围的资源维护或者其他对GNOME计划有意义的重要活动。"

基金会成员每年11月选举董事会，其候选人必须也是对GNOME有贡献者。

4.3.2 GNOME 平台及架构

尽管最初是 GNU/Linux 的桌面，GNOME 已经运行在大多数类 Unix 系统（如*BSD 变体、AIX、IRIX、HP-UX）上，并被 Sun Microsystems 公司采纳为 Solaris 平台的标准桌面，取代了过时的 CDE。Sun Microsystems 公司也以 Java Desktop System 名义发布了一个商业版的桌面——一个被 SUSELinux 系统使用的基于 GNOME 的桌面。GNOME 也移植到 Cygwin，使其能运行于 Microsoft Windows。GNOME 还被众多 LiveCDLinux 发行版使用，如 Gnoppix，Morphix 和 Ubuntu。LiveCD 能使计算机直接从 CD 引导，无需删除或者改变现有操作系统，如 Microsoft Windows。

重要 GNOME 项目

GNOME 桌面由许多不同的项目构成，部分重要的项目如下所示：

- ATK——辅助工具包
- Bonobo——复合文档技术
- GObject——用于 C 语言的面向对象框架
- GConf——保存应用软件设置
- GNOME VFS——虚拟文件系统
- GNOME Keyring——安全系统
- GNOME Print——GNOME 软件打印文档
- GStreamer——GNOME 软件的多媒体框架
- GTK+——构件工具包
- Cairo——复杂的二维图形库
- Human Interface Guidelines——Sun 微系统公司提供的使得 GNOME 应用软件易于使用的研究和文档
- LibXML——为 GNOME 设计的 XML 库
- ORBit——使软件组件化的 CORBAORB
- Pango——I18N 文本排列和变换库
- Metacity——窗口管理器

主要 GNOME 应用软件

英文 Wiki 上有更加完整的 GNOME 应用软件列表，其中主要包括：

- Abiword——文字处理器
- Evolution——联系/安排和 E-Mail 管理
- Gaim——即时通信软件
- gedit——文本编辑器
- The Gimp——高级图像编辑器
- Gnumeric——电子表格软件
- GnomeMeeting——IP 电话或者电话软件
- Inkscape—— 矢量绘图软件

- Nautilus——文件管理器
- Rhythmbox——类型 Apple iTunes 的音乐管理软件
- Totem——媒体播放器

4.3.3 GNOME 未来期待

在 GNOME 之下还有很多子计划，现在它们并不是都包含在 GNOME 发布版里。一些基于概念的纯粹试验性质的子计划有朝一日或许会加入到稳定的 GNOME 软件，其他还有一些子计划正在完善以便直接加入，例如 GNOME 存储和 D—BUS。

尽管 GNOME 应用软件可以使用很多编程语言，但是作为 GNOME 发行版一部分的 GNOME 桌面是单纯用 C 语言编写成的。关于是否使用其他高级语言如 C#，Python 和 Java 等，正在深入讨论。这些语言都已经用于开发 GNOME 应用程序，但是如果用于 GNOME 核心应用的开发，就必须在所有 GNOME 安装程序中加入相应的虚拟机，这会抬高能运行 GNOME 桌面的计算机的最低配置要求。

4.4 Ubuntu 中的桌面环境

Ubuntu 默认桌面环境采用 GNOME，而 Kubuntu 项目为 Ubuntu 用户提供了一个默认 GNOME 桌面之外的选择。由于 Kubuntu 团队的努力，Ubuntu 用户现在也可以在自己的系统上轻松安装和使用 KDE 桌面。

4.4.1 默认环境 GNOME

如果是刚刚完成 Ubuntu 的安装并第一次登录 Ubuntu 系统，用户将看到如图 4.1 所示的 GNOME 桌面环境。

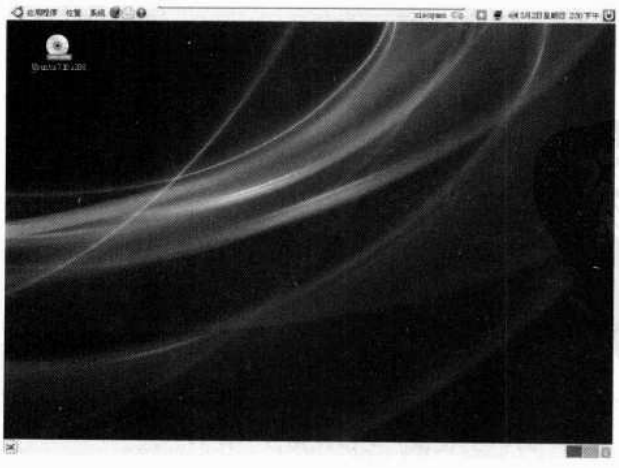


图 4.1 Ubuntu 下 GNOME 默认桌面

GNOME 控制中心

GNOME 中有一个功能很强的管理工具——控制中心，它用来设置系统方面的内容，是 GNOME 桌面环境中很重要的部分。打开这个程序很简单，只要在终端窗口下直接输入 gnome-control-center 命令即可，【控制中心】界面如图 4.2 所示。当然也可以执行主菜单的【系统】|【首选项】命令，然后直接选择其中对应的工具，如屏幕分辨率、屏幕保护等。

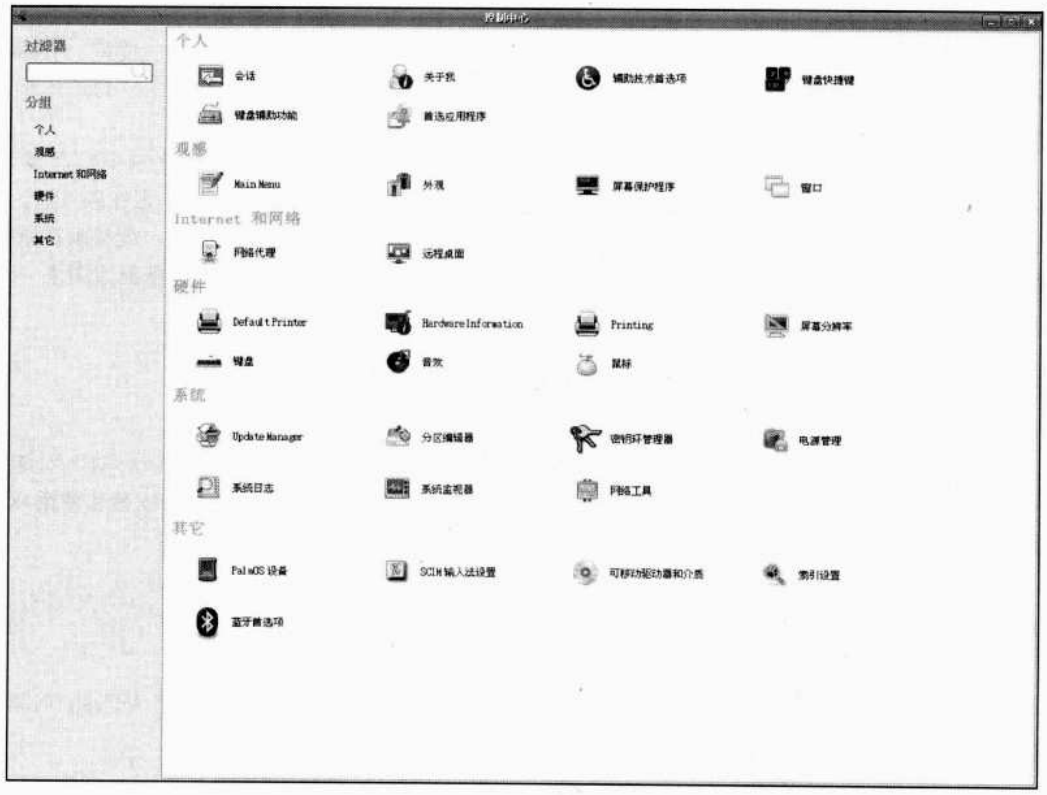


图 4.2 GNOME 控制中心

在【控制中心】窗口中，用户可以按照个人习惯和喜好，设计自己的桌面样式、风格、外观和行为。

Applet 的使用

Applet 是在自定义桌面环境时最常被加入控制面板的对象。它们其实是应用程序的一种，只不过它们的程序较为简单，一般都在面板的区域内执行。

如果要在面板中新增 Applet，只要在控制面板上单击鼠标右键，然后选择【添加到面板】子菜单中的任何 Applet，就可将它添加到 GNOME 面板。将 Applet 从控制面板上删除也很简单，在这个 Applet 上单击鼠标右键，并选择【从面板上删除】命令即可。

如果要在面板中移动 Applet 的位置，可以使用以下两种方法。

如果使用的是三键鼠标，可以在此 Applet 上按住鼠标的中间键，然后将该 Applet 拖曳至新的

位置，再放开鼠标即可。

如果使用的是双键鼠标，则需要在此 Applet 上单击鼠标右键，并选择快捷菜单中的【移动】命令，待鼠标的指针变为手形，便可移动 Applet 的位置。移动完成后，再次单击鼠标的左键或右键即可。

4.4.2 体验 KDE

安装 KD

在默认情况下，系统并不会安装 KDE，因此需要另行安装。安装的方法很简单，只要输入以下命令即可。

```
user@ubuntuer:~# sudo apt-get install kubuntu-desktop
```

输入以上命令后，系统就开始下载软件。由于需要的软件包很多，所以需要花费几十分钟来下载。所需文件下载完成后，系统会出现如图 4.3 所示的信息窗口，该窗口会提示不论系统中安装的显示管理器（Display Manager）有多少，在同一时间只能选择其中之一。在此以选择 kdm 为例。

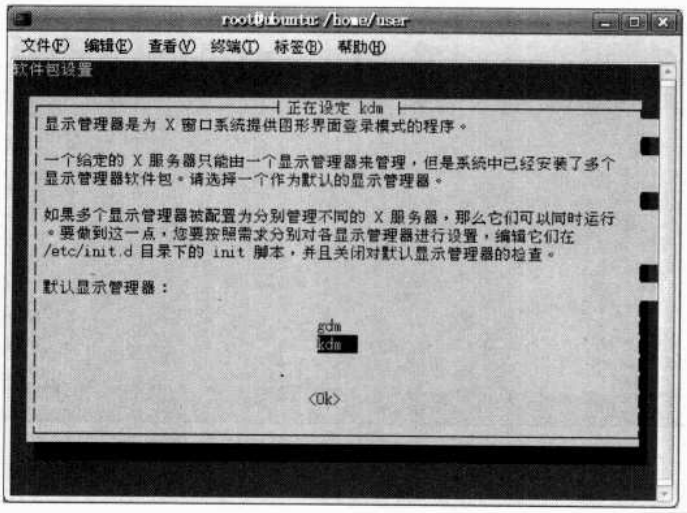


图 4.3 提示选择默认显示管理器

指定显示管理器后，系统会继续安装 KDE，直到完成为止。接下来，单击桌面右上角的关机按钮，在弹出的如图 4.4 所示的窗口中，单击【注销】按钮。

注销用户后，系统进入登录界面，由于默认的集成式桌面环境为 GNOME，所以要改用 KDE，在图形界面的左下角选择【选项】|【选择会话】，如图 4.5 所示。

这时系统弹出如图 4.6 所示的更改会话的窗口，选择第 3 项 KDE，然后单击【更改会话】按钮。

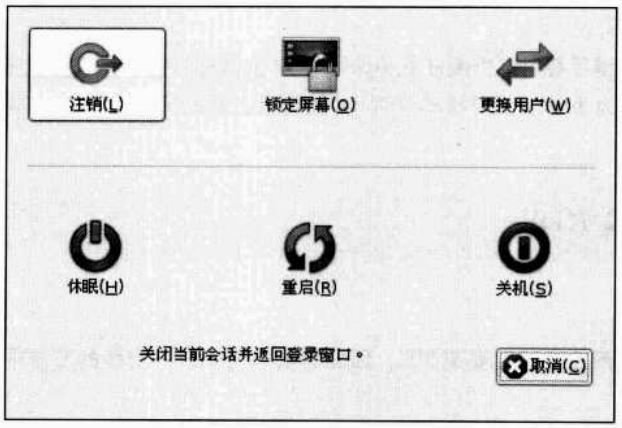


图 4.4 用户注销

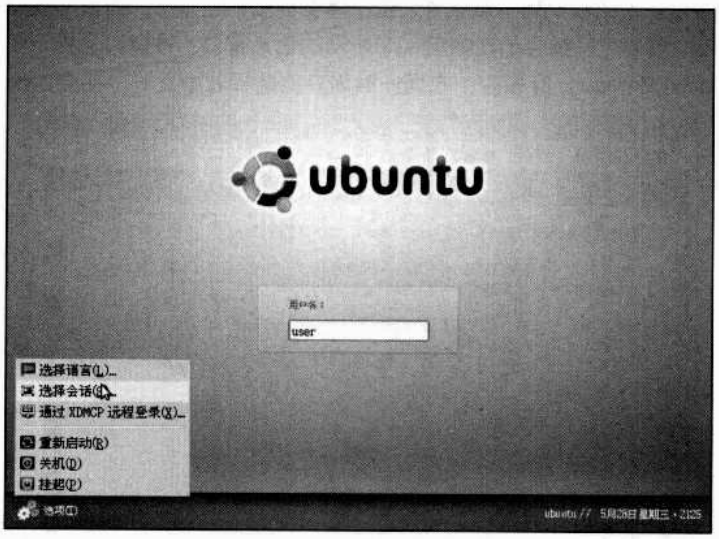


图 4.5 选择会话

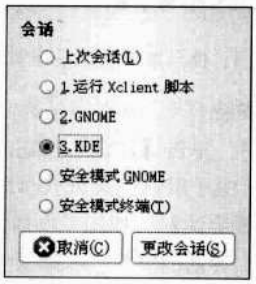


图 4.6 选择 KDE 会话

接着输入登录系统的用户名和密码，系统会询问是否把 KDE 作为会话的默认设置，如图 4.7 所示。

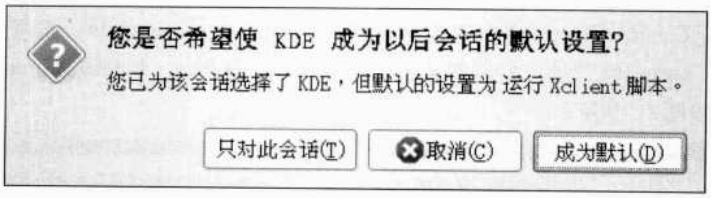


图 4.7 提示用户是否把 KDE 作为默认会话

单击【只对此会话】按钮，如果一切无误，系统就会开启 KDE 桌面环境，如图 4.8 所示。其实它与默认的 GNOME 很相似，操作的方式也差不多，唯一较大的差异就在于内置的软件包数量和种类。

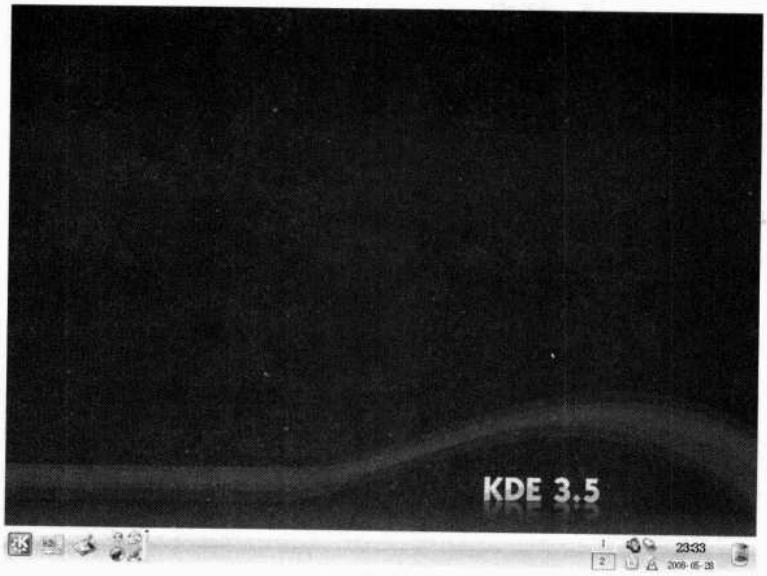


图 4.8 KDE 桌面环境

KDE 桌面默认包括两个部分：

- 屏幕下方的控制栏是用来启动应用程序以及在各个桌面环境间切换程序。
- 开始应用程序，也就是左下角的 K 图标。单击它会出现应用程序菜单，跟 Windows 操作系统下的【开始】菜单功能很相近。

设置 KDE

从 KDE 的桌面可以知道，它包含许多组件，不仅如此，KDE 桌面的功能也是相当齐全的。接下来将介绍 KDE 桌面中的组件的功能，这也是用户平时最常用到的部分。

要设置 KDE，首先需要在终端窗口中输入 kcontrol 命令，然后系统会弹出 Control Center 窗口，如图 4.9 所示。KDE 的大部分内容可以在此窗口中进行设置，本书在此介绍几个较为常用的设置。

（1）国家及语言

KDE 支持多国语言的功能，所以为了使用上的方便，可以对国家及语言的设置进行更改。要更

改此内容，需在 Control Center 窗口中左侧展开 Regional & Accessibility 选项，并单击其中的 Country/Region & Language 选项，右侧就会出现如 Locale、Numbers、Money、Time & Dates 和 Other 标签以供设置，如图 4.10 所示。

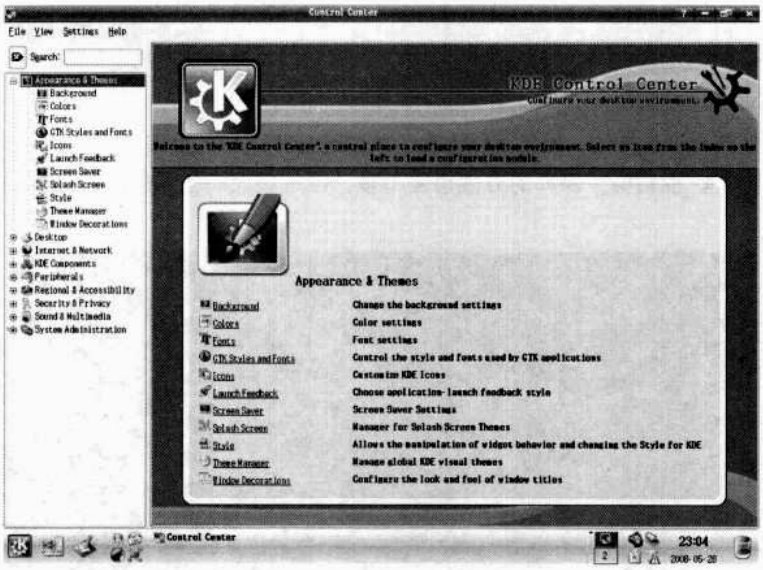


图 4.9 KDE 设置中心

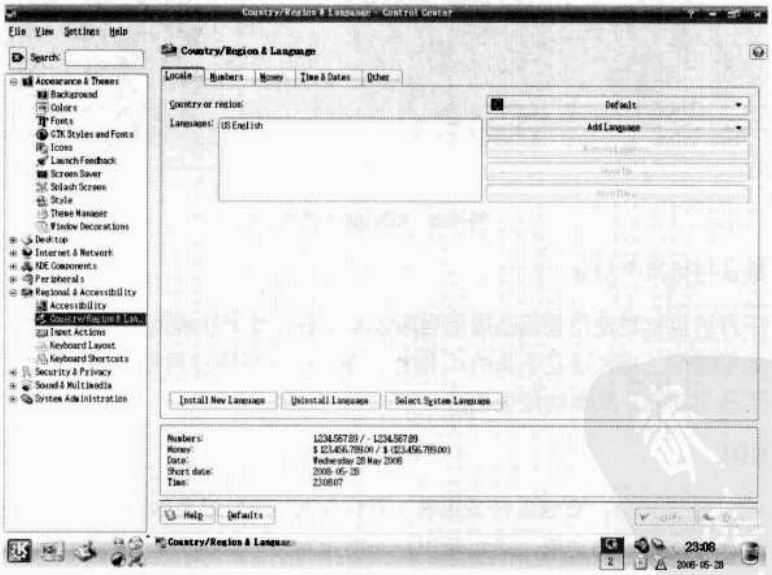


图 4.10 更改国家及语言选项

（2）窗口管理程序

KDE 中内置了多种窗口管理程序。如果要更改窗口管理程序的类型，可在 Control Center 窗口

中左侧展开 Appearance & Themes 选项，并单击其中的 Theme Manager 选项，右侧便会出现可供选择的窗口管理程序类型，如图 4.11 所示。

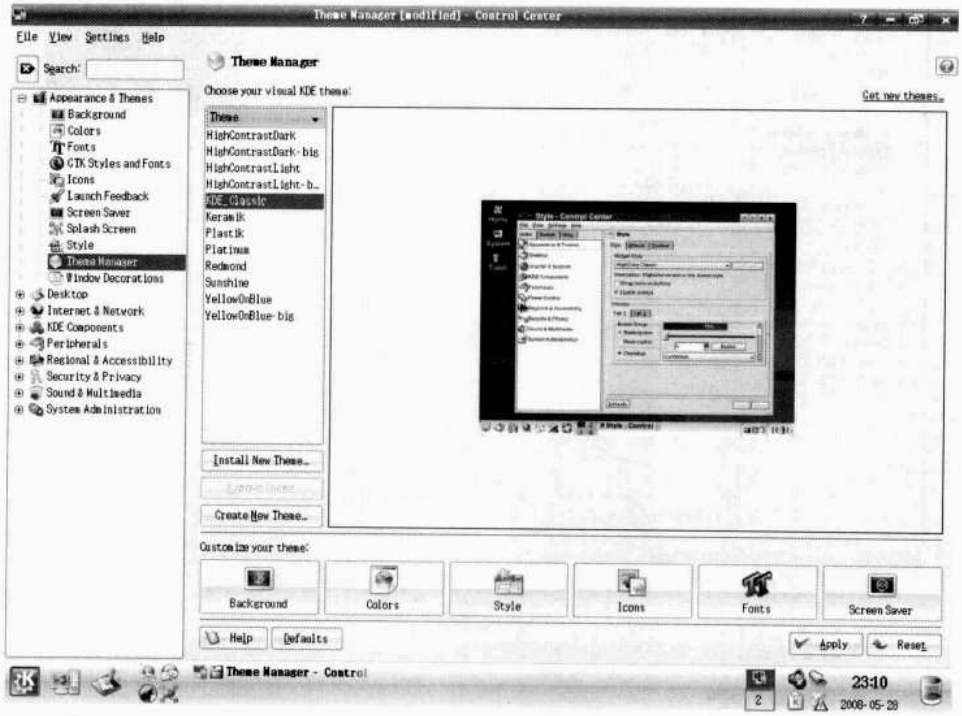


图 4.11　更改窗口管理程序类型

选择适合的窗口管理程序后，别忘了单击 Apply 按钮使设置生效。

（3）设置屏幕保护程序

为了保护屏幕并确保离开座位时计算机屏幕能够锁定，可以使用屏幕保护程序。在 KDE 中内置了很多不同类型的屏幕保护程序可供设置。

如果要使用屏幕保护程序，需在 Control Center 窗口中左侧展开 Appearance & Themes 选项，并单击其中的 Screen Saver 选项，右侧便会出现可供选择的屏幕保护程序类型，如图 4.12 所示。

（4）KDE 文件管理

Konqueror 是 KDE 家族重要的成员之一，因为它除了担任浏览器的工作外，同时也是全能的文件管理员。接下来将介绍 Konqueror 中几项常用的功能。

要打开 Konqueror，可以单击窗口左下角的 KDE 图标| Internet | Konqueror Web Browser 选项，系统即会打开 Konqueror 浏览器，如图 4.13 所示。

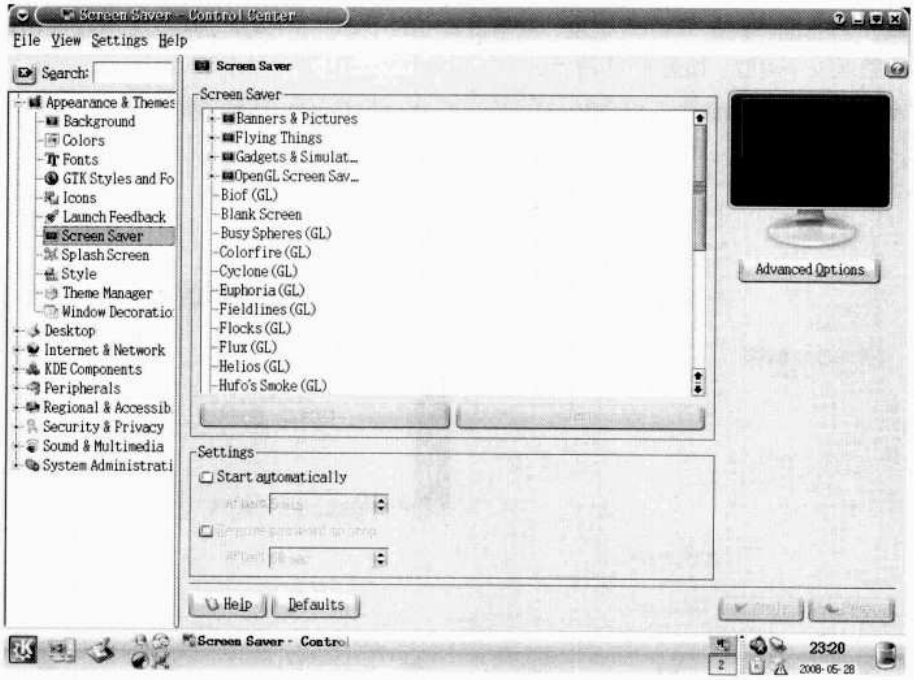


图 4.12 设置屏幕保护程序及密码

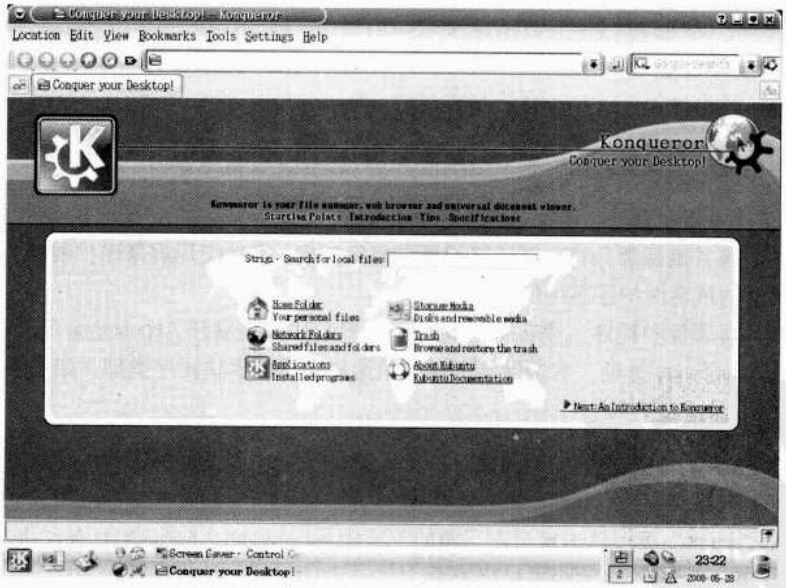


图 4.13 Konqueror 浏览器画面

直接单击工具栏上的 Home 图标，或按 Ctrl+Home 组合键，就可以进入文件管理模式，如图 4.14 所示。

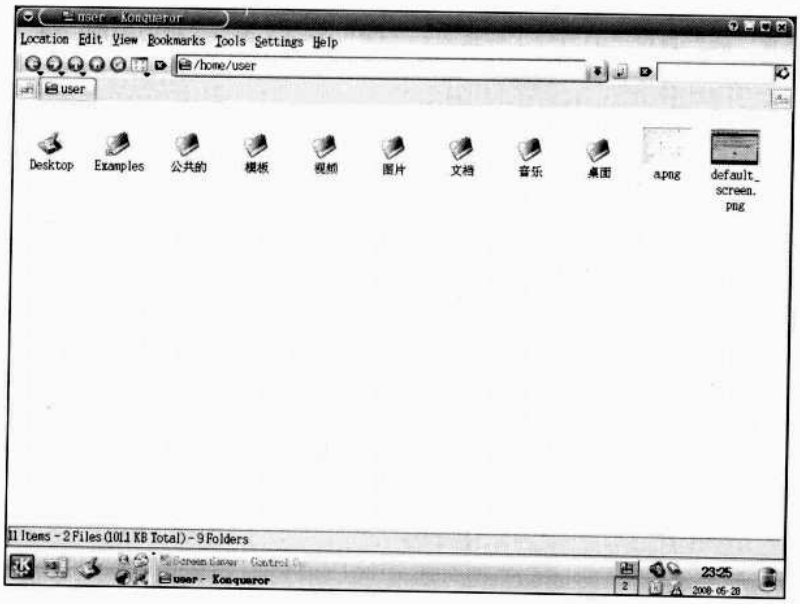


图 4.14 Konqueror 文件管理模式

Konqueror 文件管理模式中的操作方法与 Windows 操作系统很类似，单击目录或选择文件快捷菜单中的 Properties 命令，系统会出现该目录或文件的属性窗口，如图 4.15 所示。

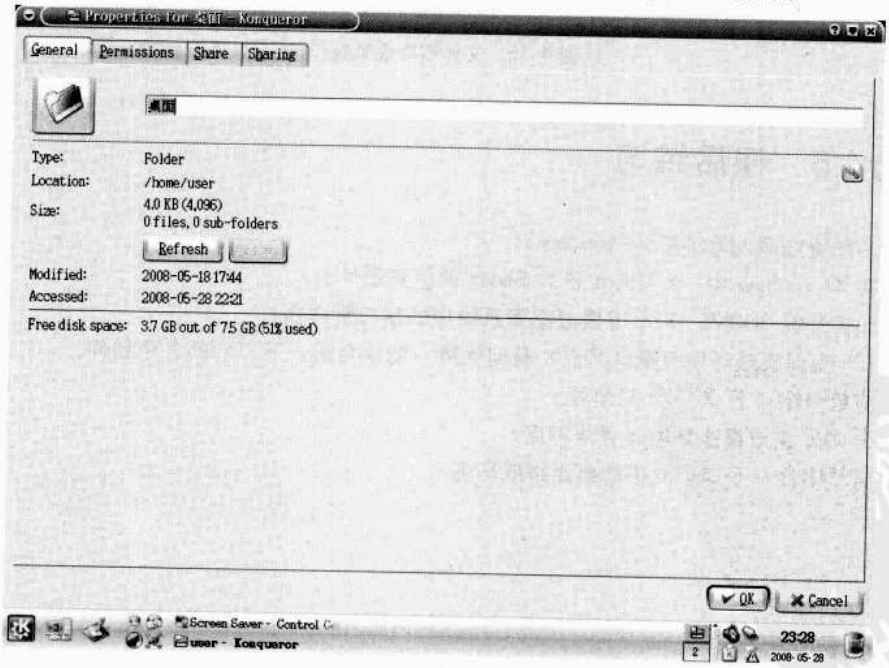


图 4.15 文件属性窗口

文件的属性窗口可以显示该文件的基本信息，如文件类型、位置或大小等，而此处唯一可供设

置的是文件名称。属性窗口中的另一个标签页——Permissions 是一个很方便的工具，如图 4.16 所示。可以通过单击的方式设置访问权限。

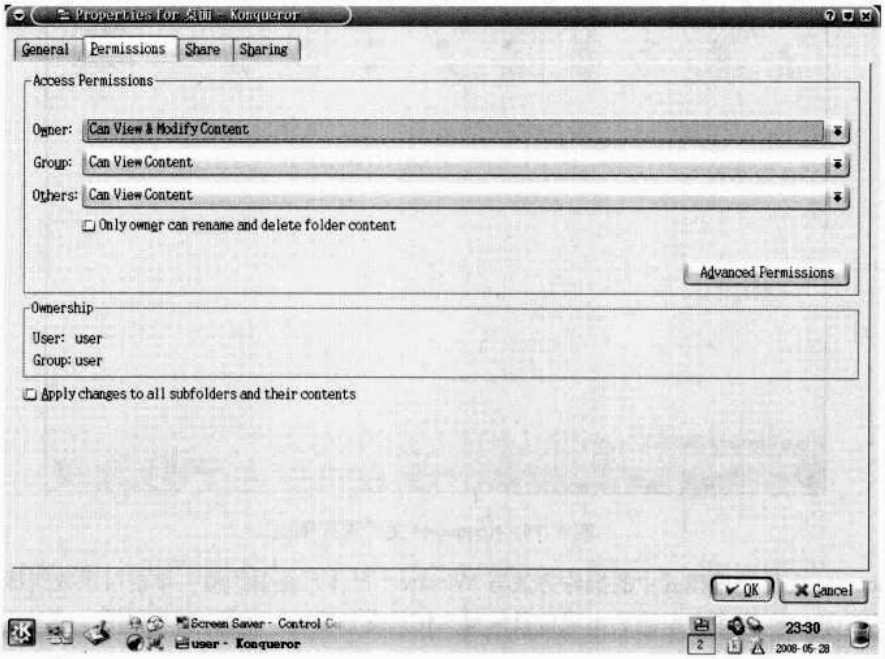


图 4.16 文件权限设置窗口

4.5 课后练习

1. 怎样更加深入地了解 X-window？
2. 在 X-window 中，X Client 和 X Server 的区别是什么？
3. 如何理解 GNOME、KDE 等集成桌面系统中的窗口管理器？
4. KDE 桌面集成环境有哪些特点？有何优势？简单介绍一下 KDE 的各个组件。
5. 简单介绍一下 GNOME 的起源。
6. GNOME 集成桌面环境有哪些特点？
7. 简单体验一下 Ubuntu 中的桌面集成环境。

Chapter05 Ubuntu 桌面环境及设置

在 GNU/Linux 系统的各发行套件中，Ubuntu 的最大亮点就是它的桌面友好性、易用性了，Unbuntu 集成了强大的桌面环境，这让习惯 Microsoft Windows 的用户能很快进行各种系统及应用操作。现在，Ubuntu 8.04 又整合了最新的 GNOME 2.22 图形桌面环境，桌面应用更上一层台阶。

记得在前面 Ubuntu 体验中已经对 Ubuntu 桌面环境进行了直观化、体验性的简单介绍，同时上一章对 X-Window 及 GNOME 集成桌面环境进行了概要介绍。下面，本章中我们就再上一个台阶，全面深入地介绍一下 Ubuntu 的 GNOME 图形桌面环境各个组成部分，以及如何进行桌面及系统设置，以让大家对该 Ubuntu 桌面环境有个系统全面的了解，并能为自己设置一个个性化的 Ubuntu 桌面环境。

5.1 Ubuntu 桌面环境组件概述

Ubuntu 系统启动环境会话后，即进入一个默认的启动画面，其中带有面板、窗口和各种图标。Ubuntu 的 Gnome 桌面环境和 Microsoft 的 Window 非常类似，因此如果用户熟悉 Microsoft 的 Window 操作的话，初次切入到 Ubuntu 这个环境，几乎感觉不出使用上的差别来，因此使用起来并不费劲。图 5.1 显示了一个典型的 Ubuntu 桌面环境。

图 5.1 Ubuntu 的桌面

Gnome 桌面环境的主要组件如下。

面板

面板就是桌面环境上的四周（主要是上下）条状区域，通过这些区域可以访问所有的系统应用

程序和菜单。默认情况下有上、下面板，面板可自由配置。面板上又包括菜单区、快捷方式图标区域、提示区、窗口切换区、任务栏、回收站等部分，其中格外重要的面板就是菜单面板。

菜单

通过“GNOME 菜单”用户访问几乎所有的标准应用程序、命令和配置选项，可以从“GNOME 菜单”访问“应用程序”菜单和“操作”菜单中的各个子菜单命令，也可以将“GNOME 菜单”作为按钮添加到面板上。

窗口

Ubuntu 桌面环境可以同时显示多个窗口，每个窗口中都可以运行不同的应用程序。窗口管理器为窗口提供框架和按钮。窗口管理器使用户可以执行诸如移动、关闭和改变窗口大小等标准操作。

工作区

用户可以将桌面环境分为几个独立的工作区。工作区是桌面环境中用户可以使用的单独区域。用户可以指定桌面环境中的工作区数量，也可以切换到不同的工作区，但是每次只能显示一个工作区。

Nautilus 文件管理器

Nautilus 文件管理器提供了一个集成的访问点，可以访问文件和应用程序。用户可以在 Nautilus 窗口中显示文件的内容，或者通过 Nautilus 窗口用相应的应用程序打开文件。用户可以使用文件管理器管理文件和文件夹。

桌面

桌面位于桌面环境中的所有其他组件之后，它是用户界面的活动组件。将对象放在桌面上可以快速访问文件和目录，或启动常用的应用程序，也可以在桌面背景上右击打开一个快捷菜单。

桌面环境最强大的功能就是能够自由配置，并且以多种方式执行任务。同时桌面环境提供桌面环境组件的互操作性。通常，可以用几种不同的方式执行相同的操作，如既可以从面板启动应用程序，也可以从菜单启动应用程序，还可以从桌面启动。系统管理员可以根据需要更改配置，所以桌面环境可能并非和本章所说的一模一样。但是，本章提供了许多有关如何使用桌面环境的快速指南。

5.2 Ubuntu 面板

当首次启动一个会话时，桌面环境通常至少包括如下两个面板：菜单面板和位于屏幕底部的边缘面板。

5.2.1 桌面面板概述

Ubuntu 桌面面板即横贯屏幕周围的长条，在 Ubuntu 系统默认的 GNOME 桌面中一般是屏幕上下两个长条，如图 5.2 和图 5.3 所示。

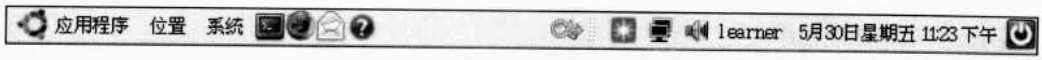


图 5.2 Ubuntu 桌面上侧面板

图 5.3 Ubuntu 桌面下侧面板

桌面面板是我们操作和使用 Ubuntu 系统的入口，默认的桌面面板包含了如下部分。

面板菜单区

面板菜单区包含了到所有应用程序的菜单项目的快捷途径。主菜单区默认包括三项：应用程序，位置和系统。

面板快捷区

面板快捷区类似 Windows 系统的快速启动栏，它上面是用来启动应用程序的各个快捷方式图标。快捷方式图标是便于你使用系统的图标和小型程序，可以说，面板快捷区和菜单一起构成了应用程序的启动入口，通过他们，用户才可以找到启动并使用 Ubuntu 的各个程序。Ubuntu 的菜单将在下一节集中介绍，现在先体验一下面板快捷区的快捷方式，例如，单击如图 5.3 所示的面板快捷区域的 Firefox 图标，即可打开 Firefox 浏览器应用程序以浏览互联网网页等。

面板通知区

面板通知区域是在上侧面板的右侧部分，嵌入在面板中的小程序，这些小程序在不妨碍用户工作的同时，允许用户运行指定任务或者监控系统或服务，包括：最新系统升级按钮、网络状态按钮、声音按钮、日期及时间以及开关机图标等。单击这些小图标，弹出相应的窗口，进入相应的设置。

关机按钮

单击最右侧的系统关机按钮，可以注销、从命令行运行应用程序、寻找文件或锁住屏幕（这会运行用口令保护的屏幕保护程序）。

关于各按钮的具体功能，我们将在后面相应章节进行详细介绍（有的已在前边相应章节进行详细介绍），在此不再重复介绍。

切回桌面按钮

位于下侧面板最左边，通过这个按钮可以最小化所有窗口，回到桌面区域。

桌面任务栏区

工作区切换器旁边的小程序是任务栏。任务栏用于向用户显示任意虚拟桌面上运行的应用程序名称。它在用户最小化应用程序的时候很有用，因为该程序会似乎从桌面“消失”。一旦程序“消失”了，用户可以单击它在任务栏上的名称来使其重现在桌面上。

工作区切换器

图形化桌面给用户提供了使用多个工作区的能力，因此用户不必把所有运行着的应用程序都堆积在一个可视桌面区域中。工作区切换器把每个工作区（或桌面）都显示为一个小方块，然后在上

面显示运行着的应用程序。

回收站

双击回收站图标，可以打开回收站文件浏览器。

其实，桌面面板也是通过 Ubuntu 的一个程序来实现的，这个程序就是 GNOME 面板控制程序。它和所有应用程序一样，只不过它是专门控制 GNOME 面板的程序，并在启动 X-window 时随系统一起启动的，图 5.4 是 GNOME 面板的基本信息。

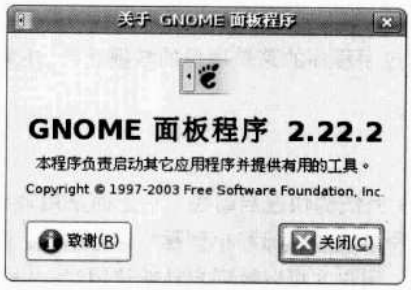


图 5.4 关于面板管理程序

桌面面板控制程序的功能很强大，如果大家对它感兴趣并想进一步了解更多的使用方法或实现原理的话可以查看面板手册（Using the Panels），右键单击上侧面板，在弹出的菜单中选择【帮助】，即可看见如何使用面板的介绍窗口，如图 5.5 所示。

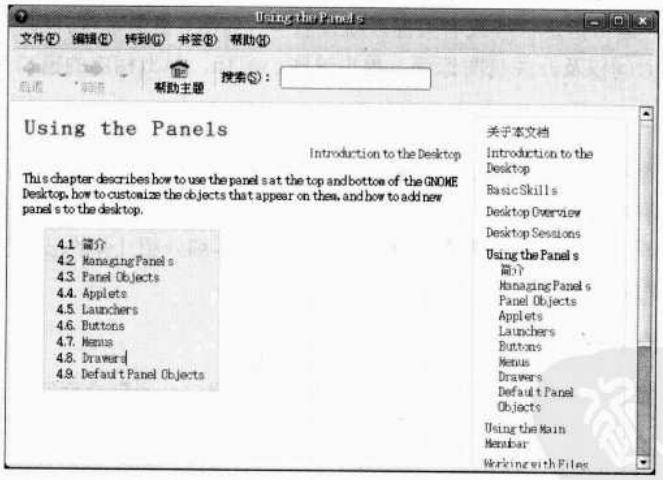


图 5.5 面板使用帮助

5.2.2 面板中对象的管理

上节面板概述中我们提到的面板中的各元素对象是可以进行操作的，这些对象通过面板程序进行管理，本节中就来了解一下面板中各对象的管理操作。

添加面板对象

一个面板可以容纳各种类型的对象，用户可以根据个人需要添加。一般来说，面板快捷区域建议主要添加用户常用的程序。在如图 5.6 所示的面板中除了有默认的菜单、窗口列表等默认面板对象外，还添加了一些其他对象，如小抽屉、天气小程序等。

图 5.6 添加小抽屉和天气小程序后的面板

在 GNOME 桌面集成环境中，可以添加以下任意对象到所有类型的面板中。

（1）小程序

小程序就是面板上的小型交互式应用程序，每个小程序都具有相似的用户界面，用户可以用鼠标或键盘进行操作。以下是面板中默认的小程序。

- 窗口列表：系统为每一个打开的窗口显示一个按钮。单击窗口列表按钮，可以最小化窗口或恢复窗口。默认情况下，窗口列表位于屏幕底部的边缘面板中。
- 工作区切换器：显示当前系统所有的工作区，并高亮显示当前工作区。使用工作区切换器可以在工作区之间切换。默认情况下，工作区切换器位于屏幕底部的边缘面板中。

要在面板上添加小程序，右键单击面板上的未用区域，选择【添加到面板】命令，打开如图 5.7 所示的对话框，用户可以从程序列表中选择要添加的小程序，选定了小程序后，如气象小程序，它就会出现在用户的面板上。

（2）启动器

启动器，也叫快捷方式，用于启动特定的应用程序、执行命令或打开文件，单击启动程序，系统就会执行与该启动程序关联的操作。启动程序可以位于面板上，也可以位于菜单中。

可以为应用程序创建自己的启动程序。启动器的添加分为两类，一种是添加应用程序启动器，另一种是添加自定义程序启动器。下面分别进行介绍。

要在面板中添加新的应用程序启动器，请右键单击该面板的空白区域，然后选择【添加到面板】命令，弹出如图 5.7 所示的对话框，在该对话框中，选择【应用程序启动器】后，再单击【添加】按钮，即可弹出如图 5.8 所示的对话框。

图 5.7 添加到面板窗口

图 5.8 添加应用程序启动器

选择你需要的应用程序就可以了，如选择【附件】，再单击【添加】按钮即可。

要添加一个自定义应用程序启动器，开始和上面流程一样，弹出图 5.7 所示的对话框后，选择【自定义应用程序启动器】，这样会打开如图 5.9 所示的对话框。用户可以在该对话框中输入应用程序的名称、位置和启动它的命令（如/usr/bin/foo），甚至为这个应用程序选择一个图标。单击【确定】按钮，这个新启动器的图标就会出现在面板上。

图 5.9 添加自定义应用程序启动器

（3）菜单

菜单是面板上的一个特殊对象，通过菜单可以访问所有桌面环境功能。面板默认包含菜单，所以可以使用菜单和面板的组合来执行任务。要从面板中打开菜单，请单击代表该菜单的图标；要从“菜单面板”中打开菜单，请单击代表该菜单的文本。

添加到面板的菜单是由一个带箭头的图标表示，箭头表明该图标代表的是菜单。用户也可以再次将“GNOME 菜单”添加到任何一个面板中。要将“GNOME 菜单”添加到面板中，请右击该面板的空白区域，然后选择【添加到面板】命令，然后在弹出的图 5.7 所示的对话框中选择【GNOME 菜单】，单击【添加】按钮即可。

（4）抽屉

抽屉是可以用抽屉图标打开或关闭的滑动扩展面板。在同时运行许多应用程序时，抽屉能帮助用户组织工作。用户可以将所有功能相同的元素放入一个抽屉，该抽屉可以放入任何其他类型的面板。

要向面板中添加抽屉，方法和添加应用小程序完全一样，请右击该面板的空白区域，然后选择【添加到面板】命令，然后在弹出的图 5.7 所示的对话框中的小程序列表中选择【抽屉】。

要打开抽屉，请单击该抽屉；要关闭抽屉，再次单击该抽屉。

（5）退出按钮

退出按钮在 Ubuntu 8.04 系统上是默认的面板按钮，位于上侧面板的最右侧。若没有，用户可以自定义添加。

要向面板中添加按钮，和添加应用小程序完全一样，请右击该面板的空白区域，然后选择【添加到面板】命令，然后在弹出的图 5.7 所示的对话框中，选择所需的【退出】按钮。

处理面板对象

面板中各个对象的位置是可以调整的，可以通过以下两种方式处理面板对象。

（1）在面板内或面板间移动对象

可以将任何对象移动到面板的任意位置，也可以将对象从一个面板移动到另一个面板。操作很简单，就是使用鼠标左键将面板对象拖动到新位置。

（2）将菜单项复制到面板

将菜单项从菜单拖到面板上。或者，如果该菜单项是一个启动程序，请右击该菜单项，然后选择【将此启动程序添加到面板上】命令。

5.2.3 面板的添加、删除及配置

Ubuntu 的面板具有很大的灵活性，可以进行相应的设置，以适应不同用户的使用习惯；另外还支持用户对 Ubuntu 桌面再次添加面板，如添加桌面右侧面板。下面我们就对面板的添加、删除及配置操作进行简单介绍。

添加面板

将鼠标移动到面板任何区域，然后单击右键，弹出快捷菜单，如图 5.10 所示。

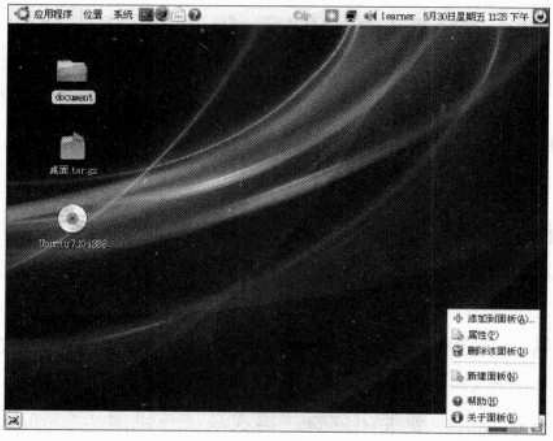


图 5.10 添加面板快捷菜单

选择【新建面板】，即在右侧添加了一个新的纯白面板，添加新面板后的效果如图 5.11 所示。

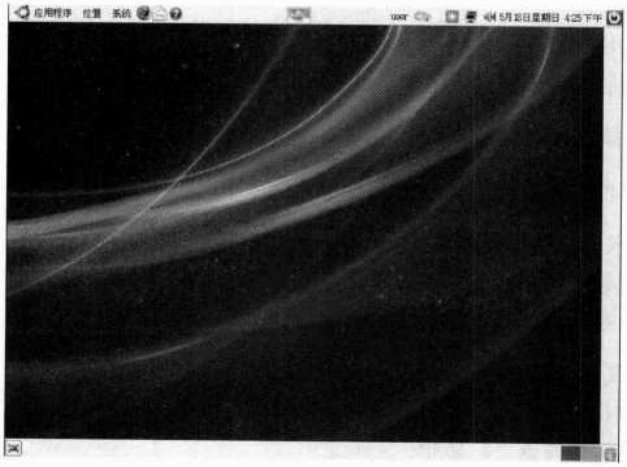


图 5.11 添加新面板后的桌面

删除面板

要删除面板，请右击该面板，然后在弹出的快捷菜单中选择【删除该面板】。

设置面板属性

用户可以自动或手动隐藏面板，把它放置在桌面上的任一边上，改变它的大小和颜色，改变它的行为方式。要改变默认的面板设置，右键单击面板上的未用区域，选择【属性】，弹出如图 5.12 所示的对话框。用户可以设置面板的大小、在桌面上的位置，以及是否想在不使用时自动隐藏它（【自动隐藏】复选框）。如果选择了要自动隐藏面板，除非把鼠标移到面板上（叫做徘徊，hovering），它就不会出现在桌面上。

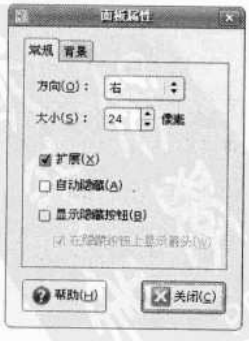


图 5.12 面板属性

5.3 Ubuntu 主菜单

在 Ubuntu GNOME 桌面的上侧面板中，有三个主菜单，分别是【应用程序】、【位置】、【系统】。用户可以单击面板上的【应用程序】、【位置】、【系统】按钮来把它扩展成一个大型菜单集合，该集合允许用户进入系统内的应用程序。

5.3.1 菜单功能说明

通过 Ubuntu 的菜单入口，可以启动几乎所有 Ubuntu 中的应用程序。注意，除了推荐的应用程序以外，还可以启动每个子菜单中的附加程序。这些子菜单使用户能够使用系统上大量的应用程序。

【应用程序】菜单

【应用程序】菜单中主要是 Ubuntu 系统所安装的普通应用程序的快捷入口。该菜单包含两级，第一级是对应用程序的分类，包括【办公】、【附件】、【互联网】、【图像】、【影音】和【游戏】，第二级就是具体的应用程序，例如：在【互联网】子菜单中包含 BitTorrent Download Client、【Ekiga 软电话】、【Evolution 邮件】、Firefox Web Browser、【Pidgin 互联网通讯程序】、Transmission BitTorrent Client、【XChat–GNOME 聊天】、【远程桌面查看器】和【终端服务客户端】，如图 5.13 所示。

图 5.13 【应用程序】菜单

【位置】菜单

【位置】菜单则主要是 Ubuntu 的文件系统以及光驱、打印机终端硬件设备入口，如图 5.14 所示，菜单中主要包括下面的选项。

- 主文件夹：打开当前用户的个人文件夹。
- 桌面：以文件夹的方式打开桌面的内容。
- 计算机：浏览本地的文件系统和远程资源。
- CD/DVD 刻录机：拖动文件到该目录后，可以单击【写入光盘】来刻录光盘。
- 软盘驱动器：浏览或操作软盘的文件夹。
- 网络：查看网络邻居的目录。
- 连接到服务器：通过 SSH，FTP，SAMBA 等访问远程资源。

除了上面描述的菜单项外，还有一个本地文件搜索的选项【搜索文件】，以及最近打开的文档链接【最近的文档】。

【系统】菜单

【系统】菜单主要是系统的外观及首选项设置及系统性能设置的功能入口，如图 5.15 所示。【首选项】与【系统管理】子菜单中包含了 GNOME 控制中心的绝大多数设置工具。在本章后面会重点介绍一部分内容。

图 5.14 【位置】菜单

图 5.15 【系统】菜单

5.3.2 菜单编辑管理

通过浏览【应用程序】菜单，你可以大概了解到有哪些应用软件已经被安装到了系统当中。但是不幸的是，它并没有将全部的已安装软件显示出来。比如某特定的“CD 播放器”其实就已经安装到了系统当中，却并没有在这个菜单中显示出来，若要设法在菜单中也把它显示出来，这时就要用到 Ubuntu 的菜单编辑器，即 Alacarte 菜单编辑器，如图 5.16 所示，通过它我们可以定制自己的菜单，并为那些安装后未自动出现菜单项的程序添加菜单项。

图 5.16 菜单编辑窗口

下面是添加一个新的菜单项的具体步骤。

Step 01 可以通过执行【应用程序】|【附件】|【Alacarte菜单编辑器】命令或在任何顶级菜单上右键单击并选择【编辑菜单】来打开 Alacarte菜单编辑器。

Step 02 在Alacarte的左侧面板中，选择要添加新菜单项的子菜单。

Step 03 单击右侧的New Item按钮，在弹出的对话框中输入“类型”，“名称”，“命令”和“注释”项。“命令”的内容通常是软件包名，“名称”的内容将出现在菜单上，“注释”的内容将出现在菜单

项附近显示的工具提示中。

另外，要改变菜单项的顺序，请使用 Alacarte 窗口右侧的 Move up 和 Move Down 按钮。要阻止菜单项的显示，可以使用每个菜单项旁边的复选框。这并没有删除该菜单项，因此您也可以用相同的方式恢复它。

5.4 Ubuntu 桌面、窗口和工作区

Ubuntu 为用户提供了 GNOME 和 KDE 两个可选的桌面环境，默认的 GNOME 桌面环境给用户带来不断增强的桌面使用体验，用户可以按照个人的偏好和习惯，定制个性化桌面、窗口及工作区的桌面环境。

5.4.1 Ubuntu 桌面

桌面位于桌面环境中其他所有组件的后面。桌面是用户界面的活动组件。用户可以在桌面启动应用程序，并且打开文件和文件夹；也可以添加桌面对象，从而更方便地访问经常使用的文件、文件夹和应用程序。例如，可以在桌面上添加应用程序的启动程序；也可以创建一个经常使用的文件的符号链接，然后将该链接添加到桌面上；还可以在桌面上存储文件和文件夹。

用户还可以通过右击桌面，打开【桌面】菜单，通过使用【桌面】菜单在桌面上执行操作。

桌面对象

桌面对象就是桌面上的图标，可以用这些图标来打开文件、文件夹和应用程序。桌面上的所有对象都位于桌面目录中。在将对象移到桌面上后，这些对象就会移动到此目录中。用户也可以使用主目录作为桌面目录。

默认情况下，桌面包含三个对象，用户还可以将对象添加到桌面上，从而更方便地访问经常使用的文件、文件夹和应用程序。例如，可以在桌面上添加一个启动程序，以便可以打开经常使用的特定应用程序。如表 5.1 所示为可添加到桌面上的对象类型。

表 5.1 可添加到桌面上的对象类型

对象类型	说明
符号链接	符号链接是指向其他文件或文件夹的对象。当从桌面上选择一个符号链接时，系统就会打开该符号链接指向的文件或文件夹。可以将符号链接移动或复制到桌面上 可以通过默认的箭头标志来识别符号链接，该标志出现在所有的符号链接中
启动程序	可以在桌面上添加如下两种类型的启动程序：应用程序和链接
文件	可以将文件添加到桌面上，桌面上的文件位于桌面目录中
文件夹	可以将文件夹移动到桌面上，而且可以在桌面上创建文件夹。桌面上的文件夹位于桌面目录中

另外，在桌面上，可以通过以下方法修改桌面对象。

（1）选择桌面上的对象

要选择桌面上的某个对象，请单击该对象。要选择多个对象，请按住 Ctrl 键，然后单击要选择的多个对象。

也可以选择桌面上的一个区域，从而选择该区域中的所有对象。在桌面上按住鼠标左键，然后拖过包含要选择的对象的区域。当按住鼠标左键然后拖动时，会有一个灰色的矩形标记出选择的区域。要选择多个区域，请按住 Ctrl 键，然后拖过要选择的多个区域。

（2）打开桌面上的对象

要打开桌面上的对象，请双击该对象。或者右击该对象，然后选择【打开】。在打开对象时，系统会对该对象执行默认操作。例如，如果该对象是一个文本文件，那么系统就会在 Nautilus 窗口中打开该文本文件。

文件类型相应的默认操作是在【文件类型和程序】首选项工具中指定的。要对某个对象执行非默认的操作，请右击该对象，然后选择【打开方式】，从【打开方式】子菜单中选择一个操作。

（3）向桌面上添加启动程序

桌面启动程序可以启动一个应用程序或链接到特定文件、文件夹或 FTP 站点。要在桌面上添加启动程序，请执行以下步骤。

Step 01 右击桌面，然后选择【新建启动程序】。即可显示【创建启动程序】对话框。有关如何在【创建启动程序】对话框中输入启动程序属性的信息，具体请参见5.2 Ubuntu面板一节。

Step 02 为该启动程序输入的命令就是在使用该桌面对象时执行的命令。下面介绍一些示例命令，以及这些命令执行的操作。

```
gedit：启动 gedit 应用程序。
gedit/user123/loremipsum.txt:在gedit 应用程序中打开文件/user123/loremipsum.txt。
nautilus /user123/Projects：在 Nautilus 窗口中打开 /user123/Projects 文件夹。
```

（4）在桌面上添加符号链接

用户可以在桌面上创建符号链接，以便执行以下操作：

- 在特定的应用程序中打开特定的文件。
- 在 Nautilus 窗口中打开特定的文件夹。
- 运行二进制文件或脚本。

要在桌面上创建符号链接，请执行以下步骤：

Step 01 在Nautilus 窗口中显示要为其创建符号链接的文件或文件夹。

Step 02 创建指向文件或文件夹的符号链接。要创建指向某个文件或文件夹的符号链接，请选择要为其创建链接的文件或文件夹，然后在菜单栏中执行【编辑】|【创建链接】命令。系统会在当前文件夹中添加一个指向该文件或文件夹的链接。可以通过默认的箭头标志来识别符号链接，该标志出现在所有的符号链接中。如图5.17所示为指向某个文件的符号链接。

图 5.17　链接文件示例

将符号链接拖到桌面上，系统就会将该对象的图标移动到桌面上。

（5）将文件或文件夹移动到桌面上

用户可以将文件或文件夹从 Nautilus 窗口中移到桌面上。要将文件或文件夹移动到桌面上，请执行以下操作：

Step 01　打开Nautilus 窗口。

Step 02　在视图窗格中，显示要移动的文件或文件夹。

Step 03　将该文件或文件夹拖到桌面上，系统就会将该文件或文件夹的图标移动到桌面上。或者，选择该文件或文件夹，然后在菜单栏中执行【编辑】|【剪切】命令，然后在快捷菜单中选择【粘贴】命令。

（6）将文件或文件夹复制到桌面上

用户可以将文件或文件夹从 Nautilus 窗口中复制到桌面上。要将文件或文件夹复制到桌面上，请执行以下操作：

Step 01　打开Nautilus 窗口。

Step 02　在视图窗口中，显示要移动的文件或文件夹。

Step 03　按住Ctrl键，然后将文件或文件夹拖到桌面上，系统会在桌面上添加该文件或文件夹的图标，并且系统也会将该文件或文件夹复制到桌面目录中。或者选择该文件或文件夹，然后在菜单栏中执行【编辑】|【复制】命令，然后在快捷菜单中选择【粘贴】命令。

（7）在桌面上创建文件夹对象

要创建文件夹对象，在快捷菜单中选择【创建文件夹】命令，桌面上就会添加一个【未命名文件夹】。输入新文件夹的名称，然后按下回车键，系统就会显示该文件夹的新名称。该新文件夹位于桌面目录中。

（8）移除或删除桌面上的对象

要从桌面上移除对象，在快捷菜单中选择【移动到回收站】命令，或者将该对象拖到【回收站】中。

当您从桌面中删除一个对象时，系统不会将该对象移动到回收站，而是立即将其从桌面上删除。只有您在 Nautilus 的【首选项】对话框中选择了【包括绕过回收站的删除命令】选项的情况下，【删除】菜单项才可用。要从桌面上删除一个对象，在右键快捷菜单中选择【删除】命令。

桌面菜单

要打开桌面菜单，请右击桌面上空白区域，弹出如图 5.18 所示的桌面菜单。

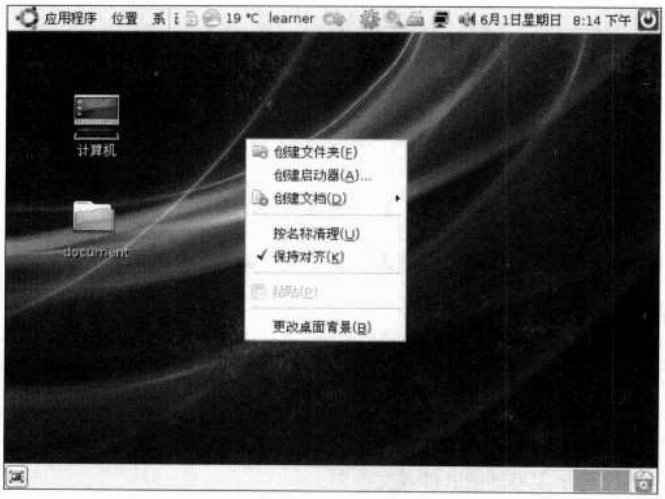


图 5.18 桌面菜单

关于这个菜单项功能的说明如表 5.2 所示。

表 5.2 桌面菜单上的菜单项

菜单项	功能
创建文件夹	在桌面上创建一个新的文件夹对象。该文件夹是在桌面目录中创建的
创建启动器	在桌面上创建启动程序
创建文档	创建新的文档
按名称清理	按照名称的字母顺序排列桌面上的对象
保持对齐	按照一定的顺序对齐桌面上的对象图标
粘贴	将缓冲区中的文件放置到桌面中
更改桌面背景	启动【背景】首选项工具，使用户可以更改桌面背景

5.4.2 Ubuntu 窗口

在桌面环境中可以同时显示许多窗口，每个窗口都有窗口框（即窗口的边框）。窗口框包含活动控制元素，可以利用这些元素对窗口进行操作。

窗口类型

桌面环境中具有以下两种类型的窗口。

（1）应用程序窗口

运行应用程序时，在应用程序窗口周围通常都有边框。在应用程序窗口的顶沿包含标题栏。标题栏包含可以用来对窗口进行操作的按钮（见图 5.19）。通过应用程序窗口框中的按钮可以执行诸如打开窗口菜单或关闭该窗口的操作。窗口菜单包含可以在窗口上执行的命令。

图 5.19　窗口顶沿

（2）对话框窗口

对话框窗口与交互式进程相互关联。对话框窗口是由窗口框、一个提供信息的交互窗口和用户控件组成。对话框窗口的边框内包含用于打开窗口菜单或关闭对话框窗口的按钮。

处理窗口

使用应用程序窗口或对话框窗口的边框能够执行多种多样的窗口操作。大多数控制元素位于窗口框的顶沿。

以下是窗口框的活动控制元素（见表 5.3）。

表 5.3　窗口框的控制元素及说明

控制元素	说明
窗口菜单按钮	单击窗口菜单按钮可以打开窗口菜单
标题栏	使用标题栏可以移动窗口
最小化按钮	单击最小化按钮可以最小化该窗口
最大化按钮	使用最大化按钮可以最大化和恢复窗口。要最大化窗口，请单击最大化按钮。要恢复窗口，请再次单击最大化按钮
关闭窗口按钮	单击关闭窗口按钮可以关闭窗口
边框	右击边框可以打开窗口菜单

要改变窗口大小，请抓取窗口的边框（注意不要抓取标题栏），拖动边框，直到窗口大小达到要求为止。

聚焦窗口

聚焦的窗口可以接收鼠标和键盘的输入。每次只能聚焦一个窗口。聚焦的窗口具有和其他窗口不同的外观。

可以使用表 5.4 所示的元素聚焦窗口。

表 5.4　聚集窗口的方法

元素	操作
鼠标	如果该窗口是可见的，则单击该窗口
快捷键	使用快捷键可以在打开的窗口之间切换。要聚焦窗口，请放开快捷键。在窗口之间进行切换的默认快捷键是 Alt + Tab 键
窗口列表	单击窗口列表中代表该窗口的按钮
工作区切换器	单击工作区切换器中要聚焦的窗口

5.4.3 Ubuntu 工作区

在桌面环境中可以同时显示多个窗口。窗口显示在桌面环境的分区中，这些分区被称为工作区。工作区是指在其中工作的离散区域。

每个工作区都包含同样的桌面、同样的面板和同样的菜单，但是，用户可以在每个工作区中运行不同的应用程序或打开不同的窗口。在桌面环境中一次只能显示一个工作区，但是可以在其他工作区中打开窗口。

当同时运行许多应用程序时，可以使用工作区来组织桌面环境。如果当前工作区内挤满了窗口，可以将工作转移到另一个工作区，也可以切换到另一个工作区，然后启动更多的应用程序。

工作区显示在工作区切换器小程序中。在图 5.20 中，工作区切换器包含两个工作区。第一个工作区内没有当前活动的窗口。第二个工作区包含打开的窗口。

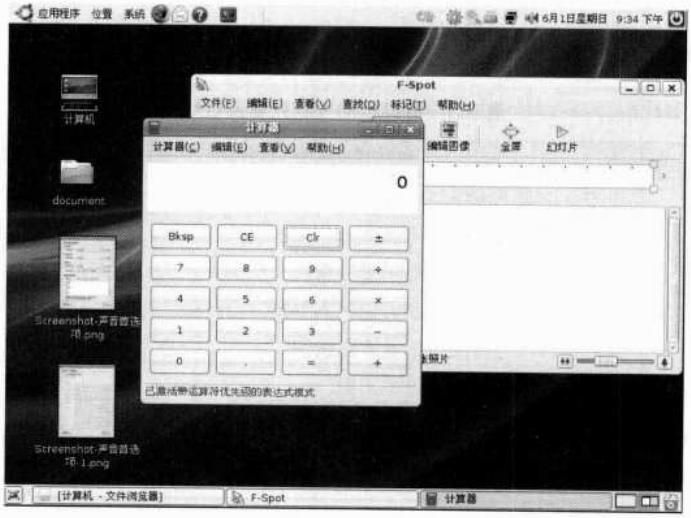


图 5.20 工作区示意图

在工作区之间切换

可以通过以下方法在工作区之间切换：

(1) 在工作区切换器中，单击想要工作的工作区。
(2) 按住 Ctrl+Alt+右箭头键可以切换到当前工作区右侧的工作区。
(3) 按住 Ctrl+Alt+左箭头键可以切换到当前工作区左侧的工作区。

添加工作区

要向桌面环境添加工作区，请右击工作区切换器小程序，然后选择【首选项】。即可显示【工作区切换器首选项】对话框。使用【工作区数量】文本框旁的微调按钮可以指定所需的工作区数量。

5.5 Ubuntu 桌面环境设置

对于 Ubuntu 系统来说，用户用得最多的还是 X-Window 桌面环境，而其中桌面是 X-Window 的脸——桌面是展示个性化最直接的地方，可以突出功能方便性，也可以突出美观性，一切尽可“我的地盘，我做主”。

5.5.1 外观首选项设置

Ubuntu 系统提供了一个统一的外观设置工具，也就是外观首选项，可以执行【系统】|【首选项】|【外观】命令打开如图 5.21 所示的窗口。在这个设置窗口中，用户可以进行的设置包括主题、背景、字体、界面和视觉效果。

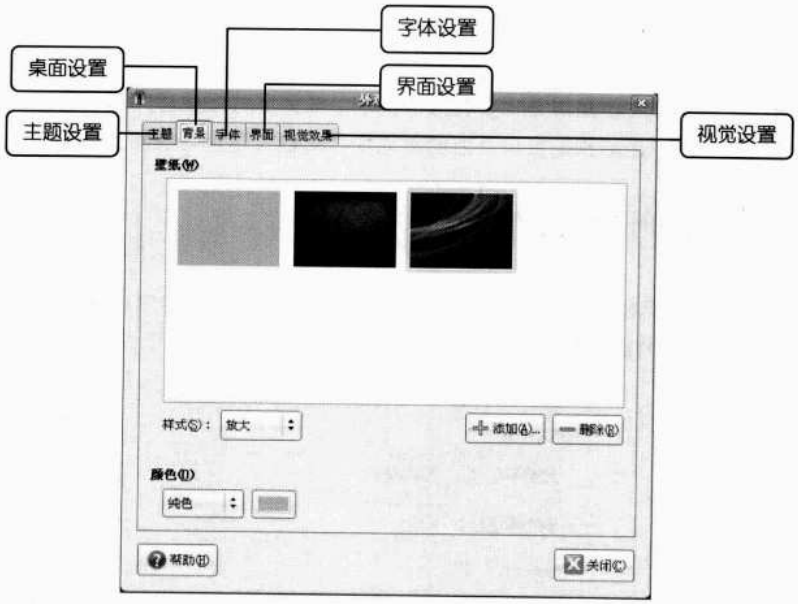


图 5.21 桌面外观设置

5.5.2 屏幕保护设置

屏幕保护程序是一种可以在不使用屏幕时用来替换屏幕图像的应用程序。可以通过以下方式使用屏幕保护程序。

- 在指定的一段空闲时间后激活。
- 在锁定屏幕后激活。

Ubuntu 提供了丰富的屏幕保护程序，用户可以通过执行【系统】|【首选项】|【屏幕保护】命令进入屏幕保护设置窗口，如图 5.22 所示。

图 5.22 屏幕保护程序设置

在屏幕保护程序中，左侧是屏幕保护程序的列表。选择屏幕保护程序后，可以在右侧窗口中预览选中的屏幕保护程序；拖动预览窗口下方的滑动块，可以设置屏幕保护的空闲时间间隔。

5.5.3 屏幕分辨率设置

当更换显示器时，需要选择适当的屏幕分辨率，以显示合适的图案比例。用户可以执行【系统】|【首选项】|【屏幕分辨率】命令进入屏幕分辨率设置窗口，如图 5.23 所示。

图 5.23 屏幕分辨率设置

登录并进入桌面后，首选项里面只能设置当前用户的桌面分辨率，登录界面的分辨率并不受影响。编辑/etc/X11/xorg.conf 文件，把不需要的分辨率都删除掉就可以了。

5.5.4 开机画面设置

Ubuntu 的开机画面管理程序叫 Usplash，用户可以执行【系统】|【系统管理】| Startupmanager 打开登录窗口首选项，如图 5.24 所示就是开机画面的管理界面，默认只有一个开机画面主题，这时用户可以安装喜欢的主题，比如 Ubuntustudio，安装完毕后在 startupmanager 中进行选择。

图 5.24 开机画面设置

需要注意的是，不同的开机画面有不同的分辨率，为了迎合显示器的尺寸，需要修改分辨率，在命令行中使用如下命令编辑分辨率：sudo gedit /etc/usplash.conf。

5.6 系统配置

Ubuntu 提供了系统各个方面的配置和管理，用户通过设置系统配置以满足特定的系统要求，下面介绍一下常用系统配置的选项。

5.6.1 本地语言设置

Ubuntu 使用 Language Support （语言支持）工具配置系统的语言环境，在语言支持工具启动的时候，会立即检查是否具备足够的语言包。在完成系统的安装后，初次打开语言支持工具的操作如下。

Step 01 执行【系统】|【系统管理】| language support命令打开语言设置窗口，启动程序的过程中将自动检查默认的语言包是否安装完整，如果没有完全安装，程序将弹出警告对话框，提示语言包没有完全安装，如图5.25所示。

图 5.25 启动语言设置并提示语言软件包没有完全安装

Step 02 单击Install按钮，会提示插入系统安装盘，确定后，进入升级状态，下载并安装默认语言的剩余软件包，如图5.26所示。

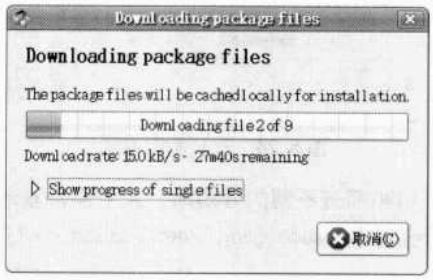


图 5.26 安装剩余语言包

Step 03 下载安装完成后，弹出图5.27所示的界面，提醒用户语言软件包更新成功。

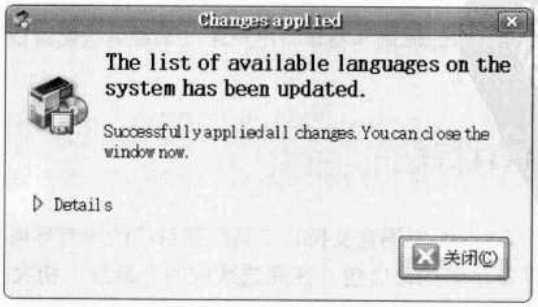


图 5.27 语言包安装完毕

Step 04 单击【关闭】按钮，关闭弹出的成功安装提示窗口。选择Chinese作为默认语言，单击【确定】按钮，如图5.28所示。这时系统提示系统更新完成，需要重新启动。

图 5.28 更改默认语言

重新启动后，Ubuntu 的登录界面、桌面系统、窗口等的字体都显示为简体中文。

5.6.2 日期和时间、及时区配置

Ubuntu 使用时间管理工具来设置系统的时间、日期和时区，也可以设置与时间服务器进行同步。设置时间、日期和时区的步骤如下：

Step 01 执行【系统】|【系统管理】|【时间和日期】命令，或者右键单击桌面右上角的时间显示栏，在弹出的窗口中选择【调整日期和时间】来【打开时间和日期设置】对话框，如图5.29所示。需要注意的是这个操作需要具有管理员的权限。

图 5.29 时间和日期设置

Step 02 要配置系统时区，单击图5.29中的【时区】栏，弹出如图5.30所示的时区选择窗口，单击代表你想要时区的城市，一个闪烁的红色“+”号显示在所选择的城市上，地图下面的【时区】微调框中的时区选择也会相应地改变。单击【关闭】按钮来选定时区并退出程序。

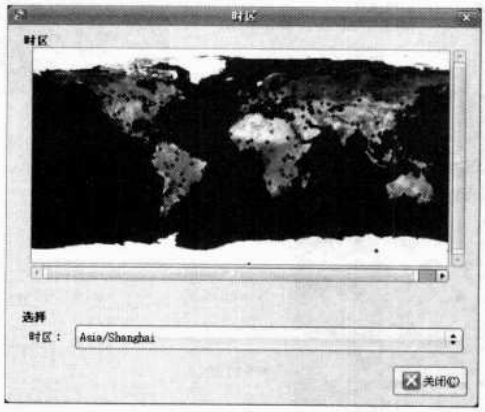


图 5.30 时区选择窗口

Step 03 单击【配置】后面的微调按钮，选择【手动】。

Step 04 要改变时间，在时间栏中，按照（时：分：秒）的顺序设置时间，直接在输入框中输入时间，或者通过旁边的上下微调按钮来调整。

Step 05 要改变日期，在日期栏中，选择需要的日期，可以使用微调按钮来选择年份和月份。

Step 06 单击【关闭】按钮，完成系统时间的设置，这时桌面右上角时间显示栏也随即改变。

5.6.3 输入法设置

Ubuntu 默认的输入法是 SCIM，若用户需要对输入法进行相应的设置，或要添加新的输入法，可以通过执行【系统】|【首选项】|【SCIM 输入法设置】命令进入设置窗口，如图 5.31 所示。

图 5.31 输入法设置窗口

5.6.4 用户自动登录系统

计算机启动时我们可以让一个用户自动登录。但并不推荐这么做，因为对大多数计算机而言这么做不安全，还可能导致其他用户访问您的信息。下面我们来介绍一下自动登录系统的操作步骤。

Step 01 执行【系统】|【系统管理】|【登录窗口】命令打开【登录窗口首选项】窗口，如图5.32所示。

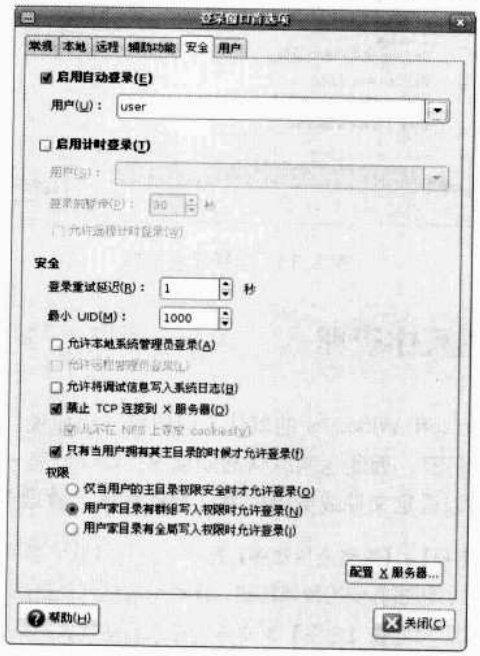


图 5.32 【登录窗口首选项】设置窗口

Step 02 选择【安全】标签页，选中【启用自动登录】复选框，然后单击右侧的下三角按钮选择要自动登录的用户。

Step 03 单击【关闭】按钮，完成设置。

5.6.5 随机自启动应用程序

Ubuntu 系统启动结束后，可以设定自动启动一些常用的程序，以做好使用准备。要设定自动启动的程序，可以进行如下操作：

Step 01 执行【系统】|【首选项】|【会话】命令打开【会话】窗口，如图5.33所示。

Step 02 单击【启动程序】标签页。

Step 03 用【添加】、【编辑】和【删除】按钮来管理启动程序。有些程序在进行配置时会自动添加到这个列表中。

Step 04 单击【关闭】按钮，完成设置。

图 5.33 会话设置窗口

5.6.6 首选应用程序管理

首选应用程序类似 Microsoft Windows 的默认启动程序，比如说，连接到网络，单击一个链接时，自动由默认的那个程序打开，因此也叫默认启动程序。Ubuntu 有一个专门管理默认启动程序的工具，用户通过这个工具可以设定文件或资源的默认启动程序。管理首选应用程序的步骤如下：

Step 01 执行【系统】|【首选项】|【首选应用程序】命令，打开【首选应用程序】对话框，如图5.34所示。

Step 02 在相应的软件分类中，单击右侧的微调按钮，选择相应的默认启动程序。

Step 03 如果选择自定义方式，需要在【命令】文本框中输入启动软件的命令。

Step 04 单击【关闭】按钮，完成设置。

图 5.34 首选应用程序设置

5.7 习题

1. Ubuntu 的桌面由哪几部分组成？
2. 体验一下 Ubuntu 的面板，尝试在面板中添加自己需要的对象或对象快捷方式。
3. 在 Ubuntu 中，主菜单包括哪些项？分别体验各分类子项的程序。
4. 如何在 Ubuntu 中编辑菜单并添加自己需要的应用程序快捷方式？
5. 体验 Ubuntu 的桌面环境，体验 Ubuntu 中弹出的窗口以及应用程序的工作区。
6. 桌面外观设置包括哪些项？
7. 请尝试设置 Ubuntu 系统桌面的分辨率，添加一个自己喜欢的屏幕保护程序。
8. 请尝试对 Ubuntu 进行一些系统设置，比如时间和日期的设置，输入法的设置等。

Chapter06 Ubuntu 的桌面应用软件

Ubuntu 的【应用程序】菜单包括【办公】、【图像】、【影音】、【互联网】、【游戏】、【附件】等几大部分，并以此为二级菜单，其下又包含不同的实际应用软件。在本章中，我们将从广大 Ubuntu 用户日常应用的角度，对大家经常用到的一些应用软件进行系统的介绍。

6.1 办公应用软件

Ubuntu 默认安装了应用广泛的办公套件——OpenOffice.org。OpenOffice.org 项目是世界上最成功的开源项目之一，它提供一个强大的字处理软件、电子表格、演示设计软件以及数据库管理软件，并提供大多数语言的支持。该套件与其他主流办公套件（如 Microsoft Office）完全兼容。由于篇幅有限，在这里我们将简单介绍 OpenOffice 的主要套件的功能，详细的操作读者可以参考 OpenOffice 的相关文档。

6.1.1 OpenOffice 软件简介

OpenOffice.org 是一套跨平台的办公室软件套件，它能在 Windows、Linux、MacOS X（X11）和 Solaris 等操作系统上运行，且与各个主要的办公室软件套件兼容。OpenOffice.org 是自由软件，任何人都可以免费下载、使用及推广它。表 6.1 是 OpenOffice.org 的主要模块。

表 6.1 OpenOffice.org 主要模块

组件	说明
Writer （文本文档）	用于实现日常文本的编写、格式排版和转换、插入图片等功能，它与 Microsoft Office 下的 Word 软件类似
Calc （电子表格）	用于制作电子表格，并提供数据统计、数据分析的功能，同时还能通过现有的数据生成图表，它与 Microsoft Office 下的 Excel 软件类似
Base （数据库）	用于在 OpenOffice.org 相关软件间无缝地操作数据库，包括创建和修改数据库表，查询和制作报表等操作，与 Microsoft Office 下的 Access 软件类似
Impress （演示文稿）	用于制作多媒体演讲稿，与 Microsoft Office 下的 PowerPoint 类似
Draw （绘图）	用于绘制简单的图表，通常由 OpenOffice 其他软件启动
Math （公式）	用于文档中创建和编辑数学公式，一般都是由 OpenOffice 的其他软件启动

在 Ubuntu 中，进入 OpenOffice.org 的方法是执行【应用程序】|【办公】| OpenOffice 命令。图 6.1 是启动 OpenOffice 的界面图。

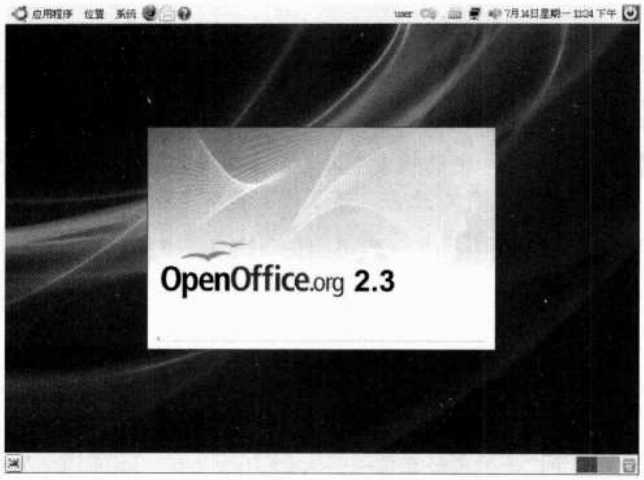


图 6.1 OpenOffice.org 启动画面

6.1.2 文字处理组件 Writer

Openoffice Writer 组件是一个现代的、功能完备的文字处理和桌面发布软件，它既可以方便地制作一份备忘录，也可以制作出包含目录、图表、索引等的一部完整的书籍。用户可以专心制作文件的内容，而 Writer 可以让您非常顺利地把文件修饰得美轮美奂。

通过执行【应用程序】|【办公】|【OpenOffice.org 文字处理】命令即可启动 Writer 组件，进入 OpenOffice Writer 的主界面，如图 6.2 所示。

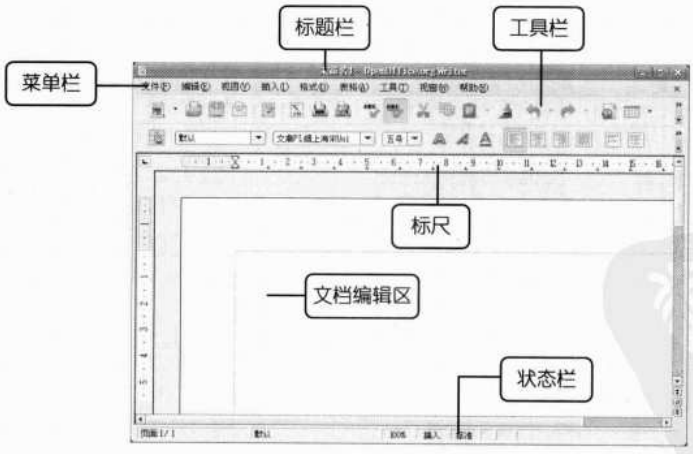


图 6.2 OpenOffice Writer 的主界面

默认情况下，Writer 的主界面包括标题栏、菜单栏、工具栏、标尺、状态栏以及文档编辑区。在窗口顶部的工具栏中包括了各类常用的工具，用于控制字体样式、纸张大小、版面调整等。

在启动 Writer 后，Writer 将自动创建一个新的空白文档，也可以通过 Writer 的向导来使用信函、

传真、会议议程和备忘录等标准文件的模板执行邮件合并等较复杂的任务。当然，也可以随意创建自己的模板。

如果用户经常需要编排文档，使用 Writer 定义的快捷键会使用户的操作更加流畅。表 6.2 列出了 Writer 下常用的快捷键。

表 6.2 Writer 常用快捷键

快捷键	说明
Ctrl + O	打开一个文档
Ctrl + S	保存当前文档
Ctrl + N	创建新文档
Shift+Ctrl+N	打开模板和文档对话框
Ctrl + P	打印文档
Ctrl+Q	退出应用程序
Ctrl + X	剪切选定元素
Ctrl + C	复制选定元素
Ctrl + V	从剪贴板粘贴
Shift + Ctrl + V	打开选择性粘贴对话框
Ctrl + A	全选
Ctrl+ Z	撤销上一次操作
Ctrl +Y	重复上一次操作
Ctrl+F	调用【查找和替换】对话框
Shift + Ctrl + F	搜索上一次输入的搜索项
Shift + Ctrl + J	在全屏模式和正常模式之间切换视图
Shift + Ctrl + R	重绘文档视图
Shift+ Ctrl +I	在只读文本中启用或禁用选择光标
Ctrl +I	所选择的区域具有斜体属性。如果把光标放到一个字上，则这个字同样也用斜体显示
Ctrl+B	所选择的区域具有粗体属性。如果把光标放到一个字上，则这个字同样也用粗体显示
Ctrl+U	所选择的区域具有下划线属性。如果把光标放到一个字上，则这个字同样也用带下划线格式显示

6.1.3 电子表格组件 Calc

OpenOffice.org Calc 是一个用户一直想要的表格处理工具——初学者觉得它直观易学；专业从事数据挖掘或分析的人欣赏它全面的高级功能。它可以帮助用户实现制作电子表格、创建可视化图表、进行数据统计处理等工作。如果用户需要，也可以与 Microsoft Excel 进行数据交互。

通过执行【应用程序】|【办公】|【OpenOffice.org 电子表格】命令即可启动 Calc 组件，进入 Calc 的主界面图，同时会自动创建一个新的电子表格文件，其中包含三个电子表格，如图 6.3 所示。

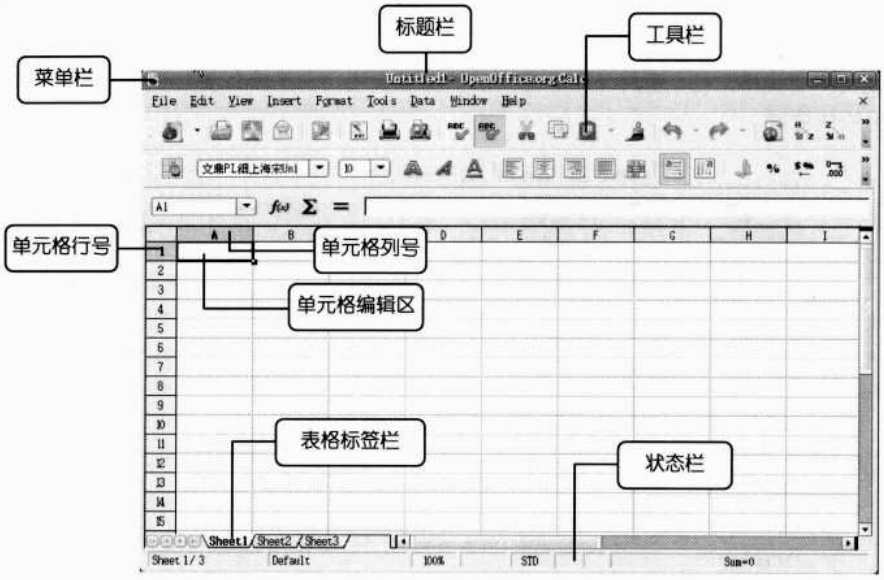


图 6.3 Calc 的主界面

默认情况下，Calc 的主界面包括标题栏、菜单栏、工具栏、单元格行列号、单元格行号、状态栏、单元格编辑区以及表格标签栏。在窗口顶部的工具栏中包括了各类常用的工具，用于控制字体样式、纸张大小、版面调整等。

当然，用户还可以自由地使用以往的 Microsoft Excel 文件，或将在 Calc 内编写的文件存储为 Excel 格式发送给仍然使用微软产品的用户。不过需要注意的是，Calc 和 Excel 并不完全兼容，为确保所有读者都能阅读您的表格处理文件，用户可在 Calc 内把文件导出成 PDF 格式——不用再买其他的软件。

6.1.4 演讲稿组件 Impress

OpenOffice.org Impress 是一个演讲稿制作应用程序，它可以为用户制作各种风格的演示文稿，提供了丰富的表现手段，包括图表、文字、音视频和动画效果，使演示人员能够轻松表达观点和展现研究成果。

通过执行【应用程序】|【办公】|【OpenOffice.org 演示】命令即可启动 Impress 组件，进入 Impress 的编辑界面图，如图 6.4 所示。

默认情况下，Impress 的主界面包含标题栏、菜单栏、工具栏、幻灯片浏览窗口、幻灯片编辑窗口、任务窗口、绘图工具栏、状态栏和视图切换栏。视图切换栏包括 5 种类型：Normal（普通视图）、Outline（大纲视图）、Notes（批注视图）、Handout（讲义视图）和 Slide Sorter（幻灯片浏览）。用户可以单击视图标签，进入到相应的视图模式进行编辑查看。而任务窗口中包含了常用的幻灯片制作功能，包括 Master Pages（样板）、Layouts（版式）、Custom Animation（自定义动画）和 Slide Transition（幻灯片切换）4 个视图，用户根据幻灯片制作需要，进入相应视图进行设置。

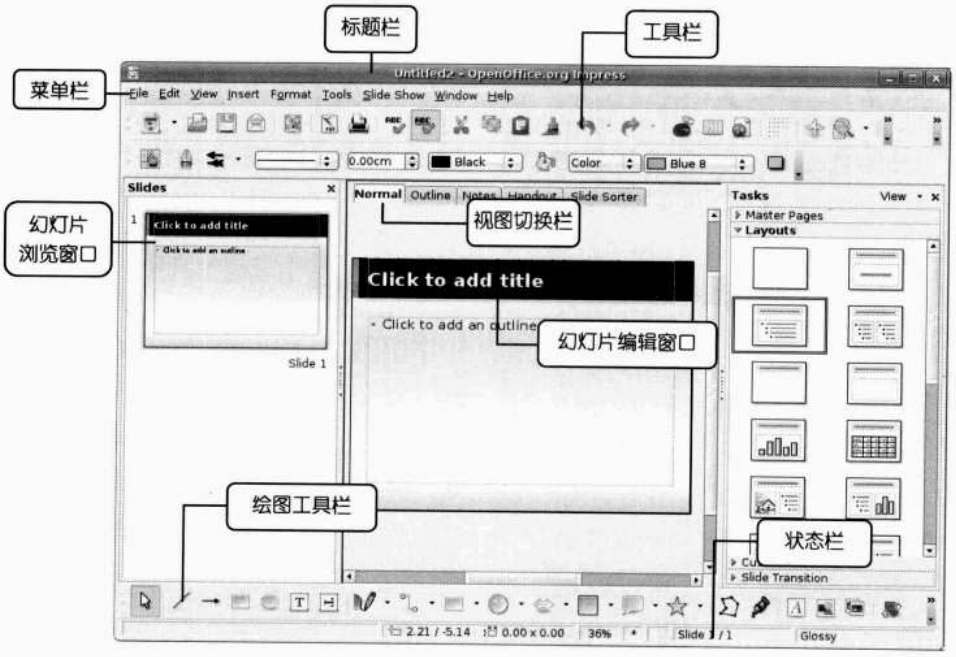


图 6.4 Impress 的主界面

6.1.5 数据库处理组件 Base

OpenOffice.org Base 是一个数据库应用程序。新版本的 Base 允许用户在 OpenOffice.org 的内部操作数据库中的数据。用户可以使用自己的数据库软件或者 Base 自己的基于 HSQL 的数据库引擎来创建和修改表格、表单、查询、报告。Base 为初、中、高级用户提供了向导、设计视图和 SQL 视图作为选择来设计数据库。Base 具有如下维护数据的功能：

- 为数据创建新的表格，也可以根据需要改变它们
- 表的索引维护，以加快数据访问
- 在可编辑表格内浏览表，增加、更改、删除记录
- 从数据中用报表向导产生令人信服的报表
- 用表单向导产生数据库应用实例

通过 Base 除了能浏览数据，而且还可以完成以下操作：

- 完成简单（单行）或者复杂（多行）数据的排序
- 通过简单（单选）或者复杂（逻辑查询）过滤器浏览到数据的子集
- 创建复杂的查询，以新途径（包括概括和多表视图）显示数据
- 用“报告自动生成”功能产生多种形式的报告

通过执行【应用程序】|【办公】|【OpenOffice 数据库】命令即可启动 Base 组件，进入如图 6.5 所示的界面。

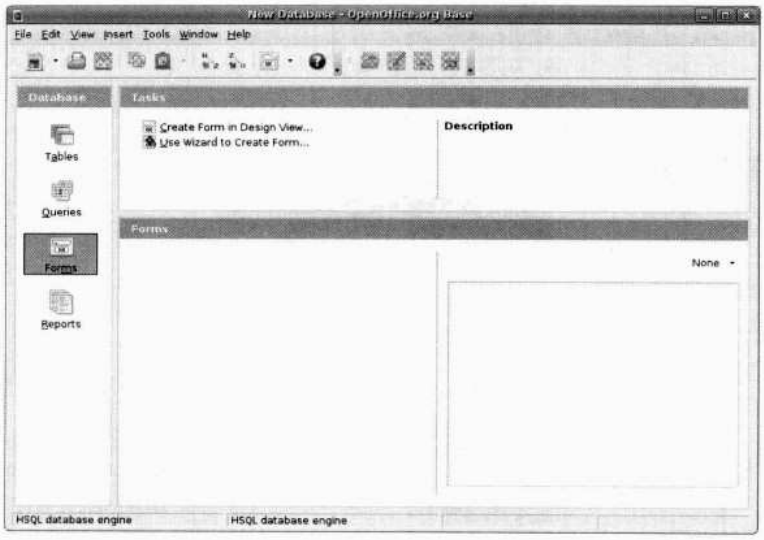


图 6.5 Openoffice Base 主界面

另外，Base 包含一个完整的 HSQL 数据库引擎版本，以 XML 文件存储数据。对于简单的数据库工作，它也可以直接访问 DBASE 文件。为了满足更多专业需求，Base 支持多种流行的数据库（如 Adabas D，ADO，Microsoft Access 和 MySQL）和通过工业标准的 ODBC 和 JDBC 驱动的数据库。它也支持 LDAP 兼容的地址簿，如 Microsoft Outlook，Microsoft Windows 和 Mozilla 等普通地址簿。

6.2 电子邮件

电子邮件已经成为人们相互联系的重要方式之一。用户主机在接入网络后，便可以通过电子邮件客户端收发和管理邮件。Ubuntu Linux 系统默认安装了 Evolution 邮件客户端。接下来，我们对 Evolution 邮件管理器和它的日历管理功能进行讲述。

6.2.1 Evolution 概述

Evolution 套件可以处理所有用户需要的邮件、联系人、任务以及日历。它也可以充当一个新闻阅读器，也可以和 GNOME 面板时钟集成在一起，以便用户可以单击访问其任务列表。

可以通过在桌面菜单系统中执行【应用程序】| Internet |【Evolution 邮件】命令来启动 Evolution，初次进入或者未曾配置过 Evolution 时，Evolution 会提示进行设置，如果已经设置好的话，就直接进入 Evolution 邮件管理器的主界面，如图 6.6 所示。

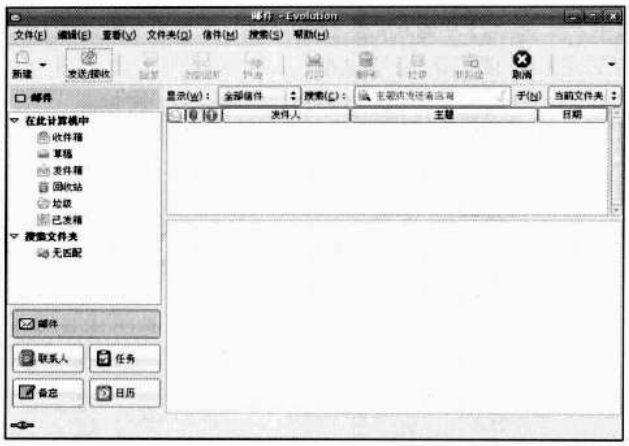


图 6.6 Evolution 邮件管理器的主界面

6.2.2 添加邮件用户

在初次启动 Evolution 时，对于没有设置过账号的 Evolution 的用户，Evolution 会自动弹出添加邮件账户的向导，开始添加邮件账户。当然用户也可以跳过这个安装向导，在运行 Evolution 后再添加账号。

通过账号助手的引导，用户可以轻松完成邮件账户的初始化。下面我们以建立 luntbuild@163.com 账号为例，介绍具体的操作步骤。

Step 01 启动Evolution邮件管理器，打开【Evolution账户助手】窗口，进入安装向导的第一步，如图6.7所示。

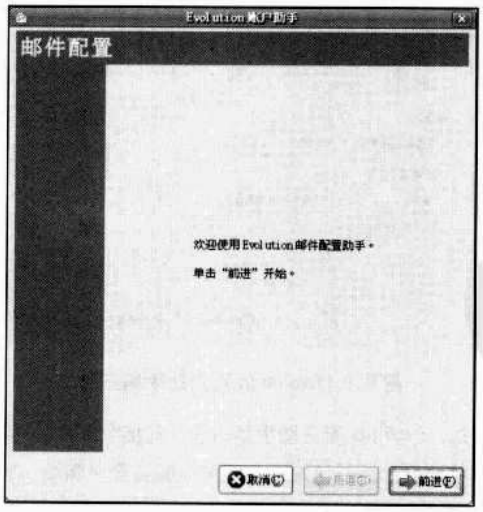


图 6.7 Evolution 账户助手第一步

Step 02 单击【前进】按钮，进入Evolution账户助手第二步，在【全名】文本框中输入希望对方看到的发件

人姓名，在【电子邮件地址】中输入完整的邮件地址，如果用户希望把这个邮件地址作为默认账号，那就勾选【使它成为我的默认账户】复选框，如图6.8所示。

图 6.8 Evolution 账户助手第二步

Step 03 单击【前进】按钮，进入Evolution账户助手第三步，如图6.9所示。单击【服务器类型】右侧的微调按钮，选择POP选项，然后输入服务器的地址，建议勾选【记住密码】复选框。

图 6.9 Evolution 账户助手第三步

Step 04 单击【前进】按钮，进入Evolution账户助手第四步，对接收邮件的功能进行配置。因为使用POP收邮件，所以使用客户端接收完邮件，该邮件立即被从服务器中删除。如果用户希望在服务器中保留邮件备份，那就得勾选【在服务器上保留信件】复选框，如图6.10所示。

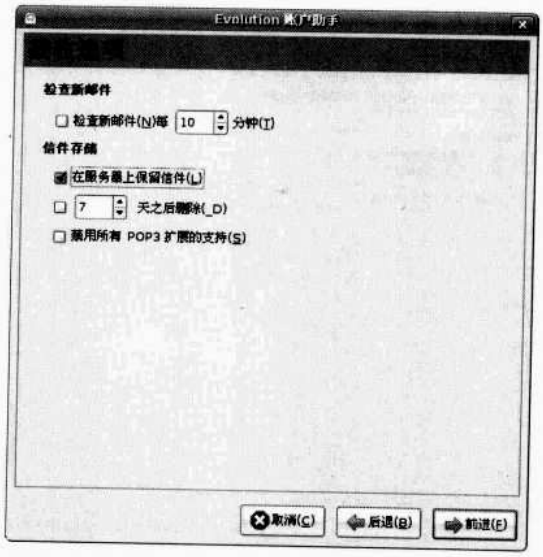


图 6.10 Evolution 账户助手第四步

Step 05 单击【前进】按钮，进入Evolution账户助手第五步，单击【服务器类型】右侧的微调按钮，选择SMTP选项，在【服务器】文本框中输入发送邮件的服务器地址，如图6.11所示。

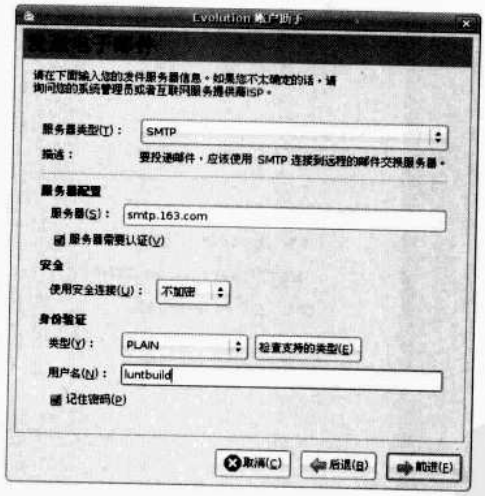


图 6.11 Evolution 账户助手第五步

Step 06 单击【前进】按钮，进入Evolution账户助手第六步，提示填写账户名称信息，如图6.12所示。

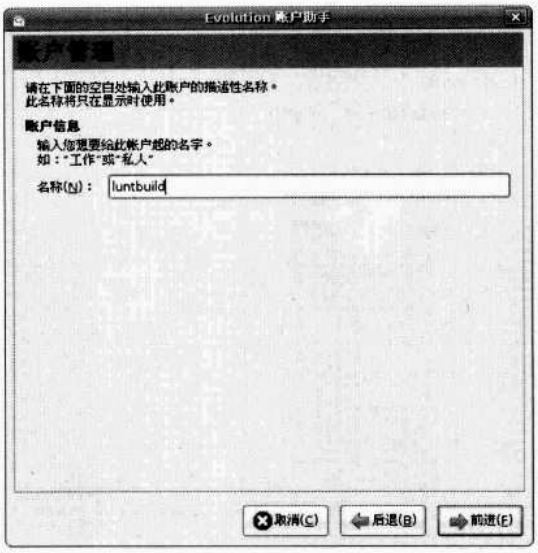


图 6.12 Evolution 账户助手第六步

Step 07 单击【前进】按钮，即可进入完成添加账号的提示窗口，如图6.13所示。

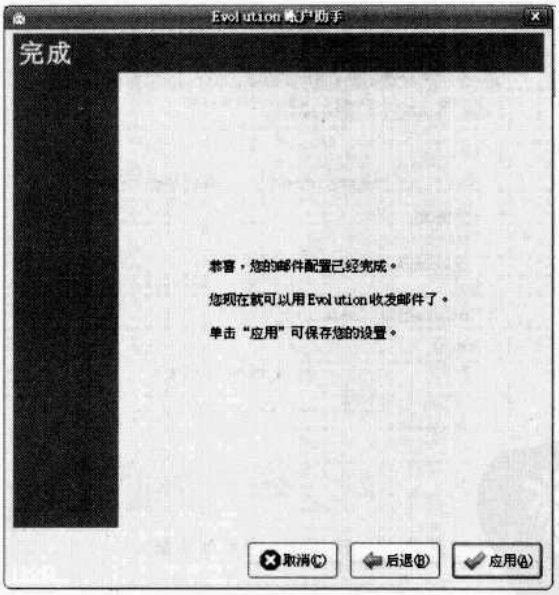


图 6.13 账户设置完成

添加账户结束后，单击【发送/接收】按钮，或使用快捷键F9，在弹出的【发送和接收邮件】提示对话框以及【为luntbuild输入密码】对话框（如图6.14所示），等待用户第一次输入密码，然后勾选【记住此密码】复选框，以后收发邮件时将不再要求输入密码了。

图 6.14　第一次收发邮件要求输入密码

6.2.3　收发邮件

在 Evolution 邮件管理中，收发电子邮件非常方便。只需要用户单击工具栏中的【发送/接收】按钮，邮件管理器将打开【发送和接收邮件】窗口，如图 6.15 所示，然后自动连接到邮件服务器下载新邮件。如果发件箱里有未发送的邮件，也将同时发送出去。

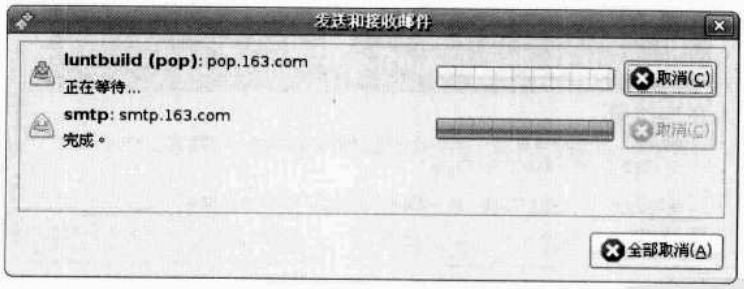


图 6.15　Evolution 邮件接收和发送

Evolution 支持用户脱机发送和接收邮件，也就是说，Evolution 在用户不连接互联网的情况下运行，完成邮件的发送和接收操作。实际上就是 Evolution 会在等待联通网络后，自动完成脱机时发出的操作命令。

收完邮件后，Evolution 将把邮件放在收件箱中，如果这个邮件没有阅读过，那就显示成加粗的字体，如图 6.16 所示。

图 6.16 Evolution 邮件列表

需要查看邮件的详细内容时，双击要阅读的邮件即可以查看具体的邮件内容，如图 6.17 所示。

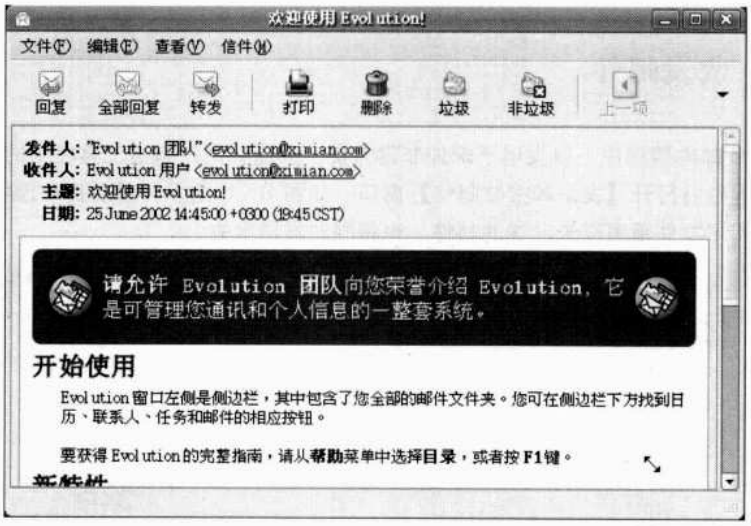


图 6.17 查看邮件

如果 HTML 邮件里包含有动画、图片或者 Flash 等内容，或许会出现无法显示的提示，这是因为目前很多 HTML 邮件里嵌入了恶意代码，只要打开邮件，恶意代码就会在用户主机上运行，对系统造成破坏。出于安全考虑，Evolution 邮件管理器默认使用纯文本的显示方式。当然，如果用户确认该 HTML 邮件是安全的，同样可以恢复 HTML 邮件的正常显示。在浏览邮件的状态下，执行【查看】|【装入图像】命令即可恢复 HTML 邮件的正常显示。

如果用户接收的邮件大多是 HTML 格式的，而且确认邮件很安全的情况下，可以对 Evolution 进行设置，以方便显示 HTML 邮件的内容。具体方法如下：

Step 01 执行【编辑】|【首选项】命令，打开【Evolution首选项】对话框。

Step 02 单击左侧【邮件首选项】。

Step 03 在右侧窗口选择【HTML信息】标签页，看到如图6.18所示的界面。

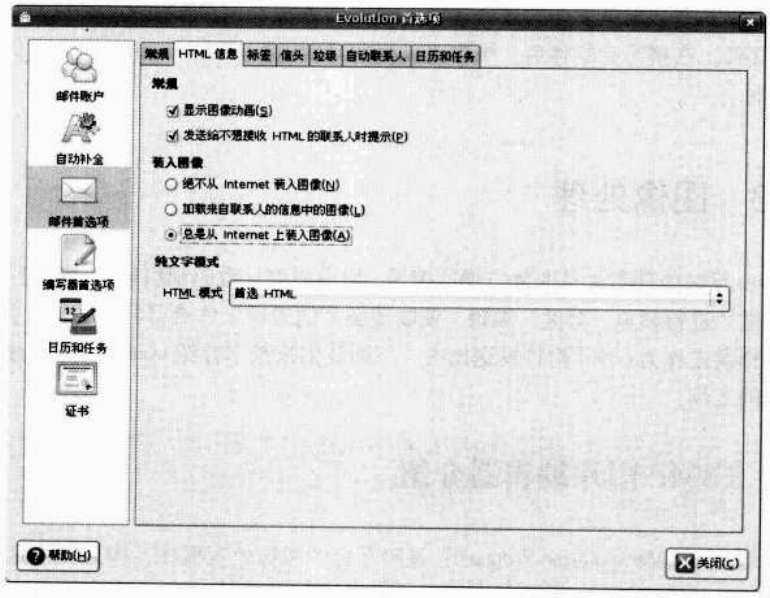


图 6.18 允许加载 HTML 邮件的图像

Step 04 选择【总是从Internet上装入图像】单选按钮，然后单击【关闭】按钮完成设置。

如果需要撰写电子邮件，Evolution 为用户提供了丰富的邮件撰写功能，通过执行【新建】|【邮件】命令新建邮件，或使用快捷键 Ctrl+Shift+M 打开【撰写新信件】窗口，如图 6.19 所示。

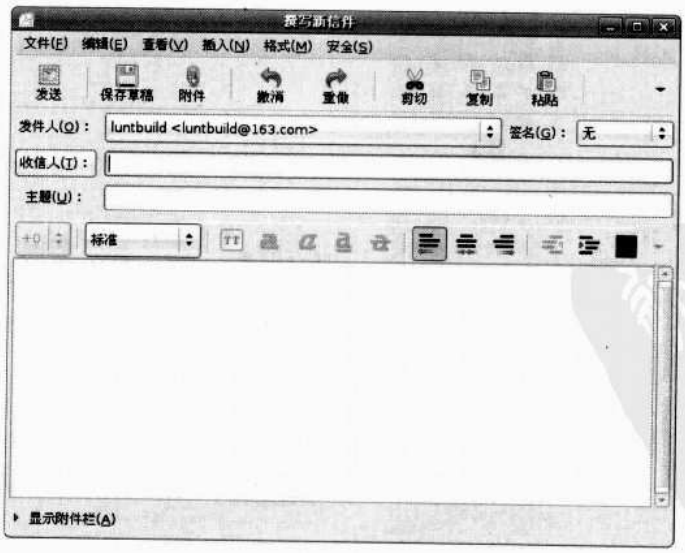


图 6.19 撰写新信件

默认情况下创建的是纯文本邮件，如果希望使用更多的文件编辑功能，可以采用 HTML 格式。如果想在邮件中追加附件，执行【插入】|【附件】命令，打开【插入附件】窗口，等待用户选择附件内容。需要注意的是，出于安全方面的考虑 Evolution 不允许用户添加某些特殊格式的附件，比如 exe 等可执行文件。在撰写完邮件后，单击工具栏的【发送】按钮，Evolution 将关闭新邮件窗口，并开始发送邮件。

6.3 图像处理

使用 Ubuntu 系统处理图片是非常方便的事情，用户能够从数码相机或移动硬盘上导入图片，能够很方便地对图片进行归总、分类、编辑，使收藏图片的管理工作变得异常轻松，并且随时都可以把图片打印、刻录或作为 Email 附件发送出去。下面我们来简单介绍 Ubuntu 下实现图片浏览与编辑以及屏幕抓图的工具。

6.3.1 GIMP 图片编辑器介绍

GIMP（GUN Image Manipulation Program）是跨平台的图像处理程序，作为一个专业的绘图软件，它具有丰富的图片处理功能，包括照片渲染、图像合成以及图像转换。

GIMP 在 Ubuntu 中默认安装，可以通过执行【应用程序】|【图像】|【GIMP 图片编辑器】来打开 GIMP 窗口，如图 6.20 所示。

图 6.20 GIMP 的主窗口

从图 6.20 看出，与大多数应用程序主界面风格明显不同，GIMP 并不是将所有功能选项都放置在一个窗口里。GIMP 开发人员一直认为，将所有的功能选项都放置在一个窗口下是一种非常低效的方法，因为它强迫程序启用大量的无用功能。GIMP 采取了专门的窗口实现特定功能。不过为了避免过多窗口带来的不便，它采用了灵活的组织功能对话框方式。

GIMP 对图像文件的操作是非常灵活的，可以通过很多方式打开图像文件，例如，GIMP 支持从 Naulius 文件管理器中将图像文件直接拖到 GIMP 中来打开并进行编辑。下面我们介绍一下如何使用 GIMP 打开一个图像文件。

Step 01 执行【文件】|【打开】命令，打开【打开图像】对话框。

Step 02 在左侧窗口中选择打开文件的存储路径，在中间的窗口选择要打开的文件，这样就能在右侧的窗口中预览图像文件，如图6.21所示。

图 6.21 打开图像

Step 03 单击【打开】按钮，打开一个新的图像窗口，显示所选择的图像文件。

GIMP 还提供了多种保存图像文件的格式。不过需要注意的是，从一种文件格式保存为另一种格式，可能会造成部分图像信息丢失。XCF 是 GIMP 内建的文件格式，除了修改过程的撤销信息不能保存外，几乎能保存一个图像所有的信息。但是 XCF 并不能被其他流行的图片处理软件识别，所以完成图像修改后，还是找一个流行的图像格式进行保存吧，如 JPEG、PNG 等。下面介绍保存并转换图像格式的具体操作方法。

Step 01 在图像窗口中，执行【文件】|【保存】命令，打开【保存图像】对话框，如图6.22所示。

Step 02 在【名称】文本框中输入图像文件名，在【保存于文件夹】中选择保存路径。

Step 03 单击【选择文件类型】下拉按钮，在展开的文件类型中选择需要转换的图像格式，在这里选择JPEG图像。

Step 04 单击【保存】按钮，完成图像文件的保存及格式转换。

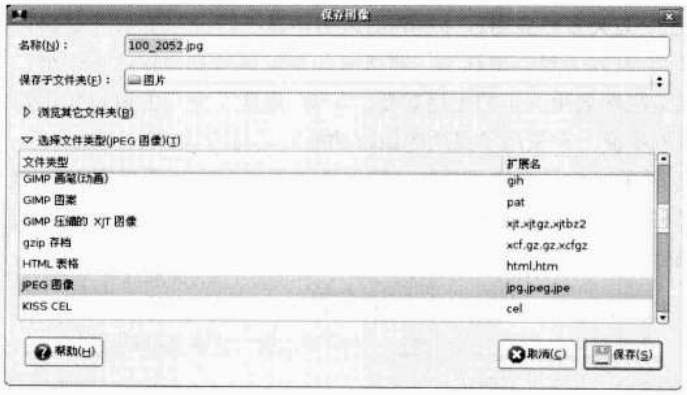


图 6.22 保存图像

6.3.2 使用 gThumb 图像查看器

gThumb 是一个高级图像查看器和浏览器。它有许多有用的功能，如浏览文件系统、幻灯片显示、图像编目、创建 Web 相册、照相机导入、图像 CD 刻录、文件批处理以及诸如透明和着色处理等图像快速编辑功能。

gThumb 默认包括在 Ubuntu 中。要启动它，执行【应用程序】|【图像】| gThumb 命令打开图像查看器 gThumb，如图 6.23 所示。需要获得更详细的帮助可以执行【帮助】|【内容】命令查找。

图 6.23 gThumb 图像查看器

6.3.3 屏幕抓图

用户在浏览网页、聊天或者系统遇到一些棘手的问题时，总是希望把屏幕上显示的图像保存下来，以便后期可以进行分析诊断。Ubuntu 默认安装了 Gnome-Screenshot 屏幕抓图工具，该工具使用方法非常简单，启动方式也很灵活：执行【应用程序】|【附件】|【抓图】命令打开抓图窗口，如

图 6.24 所示。

图 6.24 Gnome—Screenshot 抓图软件

如果用户经常需要抓取屏幕图像，使用快捷键非常方便，抓图快捷键如表 6.3 所示。

表 6.3 屏幕抓图快捷键

功能	快捷键
抓取整个屏幕	Print Screen
抓取当前活动的窗口	Alt+Print Screen

使用快捷键抓图很方便，不过在某些特殊情况下，快捷键可能会无能为力，比如抓取菜单的时候，由于菜单本身就处于激活状态，就无法使用键盘快捷键了。这时候，我们就可以使用图 6.24 中的选项【抓取前的延迟】，比如，设置为 5 秒，在这 5 秒钟内，我们确认进入目标菜单后，然后 Gnome—Screenshot 在 5 秒自动截取菜单画面。下面是抓取桌面菜单的具体例子。

Step 01 在【抓图】窗口中，选择【抓取整个桌面】单选按钮。

Step 02 在【抓取前的延迟】微调框中，输入5，并单击【抓图】按钮。

Step 03 进入到目标菜单中，注意动作要保证在5秒钟内完成。等待延迟时间结束，系统将自动弹出【保存抓图】对话框，如图6.25所示。

Step 04 在【名称】和【保存于文件夹】中输入文件的名称和保存路径。单击【保存】按钮，完成对菜单的截图。

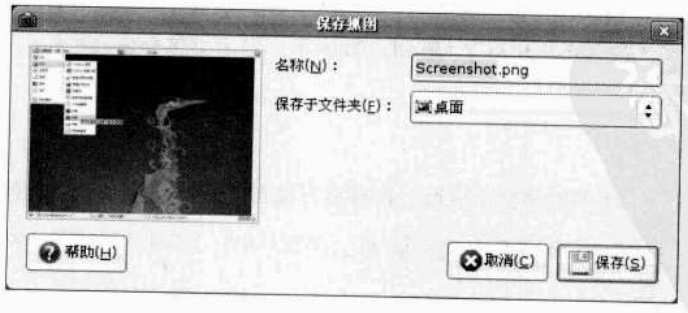


图 6.25 保存抓图

6.4 浏览器

使用浏览器查看网页已经成为现代网民的普遍习惯了，Linux 下有各种版本的 Web 浏览器，包括 Galeon、Mozilla、Opera、Konqueror 和 Netscape 等。Ubuntu Linux 默认安装了 Mozilla Firefox 浏览器，因而我们在这一节中将主要介绍 Firefox 浏览器。

6.4.1 Firefox 简介

Mozilla Firefox 俗称火狐，是由 Mozilla 基金会与开源团体共同开发的网页浏览器。Firefox 是从 Mozilla Application Suite 派生出来的网页浏览器，从 2005 年开始，每年都被媒体 PC Magazine 选为年度最佳浏览器。与 Opera，IE 浏览器相比，Firefox 非常精简，安装程序小于 10M，为下载和安装带来便捷。而同时 Firefox 还提供了迁移工具，方便用户把收藏夹和其他信息从其他浏览器导入到 Firefox 中。除此之外，Firefox 还具有以下特性：

标签页浏览

Firefox 支持标签页浏览，是指可以在一个窗口中开启多个页面，这个功能继承自 Mozilla Application Suite，也成为 Firefox 的著名特色。Firefox 也允许使用者在首页中使用“|”作为分隔符号，在启动时自动在多个分页中开启设置的首页，让使用者不只可以设置一个首页。

插件

Firefox 的扩展性非常好，用户可以选择安装各种各样的插件为 Firefox 添加各种新功能。插件的种类包罗万象——包括主题背景、鼠标样式、广告窗口阻挡、增强的分页浏览等。

及时查找

Firefox 提供增强的查找功能，包含了快速的“即打即找”功能，使用者只需要按 **F3 键**后在查找框输入要寻找的字符串，就可以自动标示出要寻找的字符串。

实时书签

通过实时书签，使用者可以以书签的方式来阅读 RSS 或 Atom 项目，这个功能第一次出现在 Firefox 1.0 的预览版，随后也移植到了 Mozilla Suite 中。即时书签会自动更新，方便用户订阅最新头条及为网站更新信息。

跨平台支持

Firefox 可以在多种不同的平台下执行，目前官方发布的版本支持了以下几种平台。

- 多种版本的 Windows 操作系统，从 98、98SE、Me、NT 4.0、2000、XP、Server 2003 到 Vista。
- 苹果计算机的 Mac OS X。
- 以 Linux 为基础的操作系统，系统中必须使用 X.Org Server 或 XFree86。

除了上述操作系统之外，由于 Firefox 是开放源代码的软件，加上代码是与操作系统独立的，因此 Firefox 可以在多种平台和操作系统上编译，包括 OS/2、AIX、FreeBSD、Windows XP Professional x64 版本都有可执行的 Firefox 编译文件。

支持多种网络标准

Firefox 支持非常多的网络标准，包括 HTML、XML、XHTML、SVG 1.1（部分）、CSS（除了标准之外，还有扩充的支持）、ECMAScript（JavaScript）、DOM、MathML、DTD、XSLT、XPath 和 PNG 图片格式（包含透明度支持）。

安全性

Firefox 使用了“沙盒安全模块”（Sandbox Security Model），限制了网页脚本语言对用户本地数据的访问，保护用户数据不受恶意脚本语言的攻击。对于网页数据的传输，则使用 SSL/TLS 的加密方式来保障使用者和网站之间传输数据的隐秘性。同时 Mozilla 基金会提供了“臭虫奖金”来奖励发现 Firefox 漏洞的研究者。Mozilla 官方希望安全弱点可以在被恶意利用之前被发现，进而去修正它，避免用户遭受攻击。

完备的开发工具

对于网页开发者，Firefox 也提供一个良好的开发平台。网页开发者可以通过内置的工具来进行开发工作，例如错误控制台、DOM 观察器，此外还可通过安装插件来延伸开发功能，如 firebug。

Ubuntu Linux 默认安装了 Mozilla Firefox 浏览器，并为 Firefox 定制了快速启动器，放在顶层面板中。正常启动后，用户只需要单击启动器中的图标，即可打开 Firefox 浏览器。此外还可以通过在桌面菜单系统中执行【应用程序】| Internet | Firefox 命令来启动。Firefox 界面如图 6.26 所示。

图 6.26 Firefox 窗口界面

图 6.26 是 Firefox 的默认风格，用户可以通过设置来隐藏/显示工具栏、定制工具栏的内容，或将安装的常用插件放置在工具栏中。因而，Firefox 能充分满足用户个性化定制的需求。

6.4.2 分页浏览网页

使用过 IE 浏览器的用户都知道，如果打开的网页比较多，浏览器的窗口就重叠地放在任务栏中，使浏览网页变得很繁琐，常常要花费一点时间来切换窗口。Firefox 具有分页浏览网页的功能，用户不必再打开新的窗口来打开网页，而是提供一个标签页显示一个网页的内容。

使用标签页浏览网页的方法很简单，操作步骤如下：

Step 01 执行【文件】|【新建标签页】命令，在Firefox浏览器中打开一个空白的标签页，也可以使用快捷键Ctrl+T。

Step 02 在地址栏中输入访问的URL，按回车键，浏览器开始访问该站点。

在浏览网页的过程，需要打开新链接时，右键单击页面的链接，在弹出的菜单中选择【在新标签页中打开】命令，Firefox 将立即新建一个标签页打开选中的链接地址；或者单击鼠标的中键达到同样的效果，如图 6.27 所示。

图 6.27 分页浏览网页

6.4.3 使用书签

在浏览网页的过程中，用户或许会发现比较有价值或有趣的网站，希望能把这个网址记录下来，以便下一次访问。使用 Firefox 浏览器的书签功能，能够为用户妥善收藏这些网址。具体操作如下：

Step 01 执行Bookmarks | Bookmarks This Page命令，或按Ctrl+D快捷键，或者在网页上单击右键，在弹出的菜单中选择Bookmarks This Page命令，Firefox将弹出Page Bookmarked对话框。

Step 02 在弹出的对话框中输入书签名，选择书签的位置，如图6.28所示。

Step 03 单击Done按钮，关闭Page Bookmarked对话框，完成一个页面的收藏。

用户在长期使用 Firefox 的书签后，或许收藏了很多网址，而这个时候，用户可能不再需要某些网址，下面就来整理一下，删除那些不需要的书签。

Step 01 执行Bookmarks | Organize Bookmarks命令，打开Library窗口，如图6.29所示。

Step 02 在Library窗口下方书签列表中选中需要删除的地址，然后单击窗口上方的Organize下拉菜单，在弹出的菜单中选择Delete命令，即可将选择的书签删除。

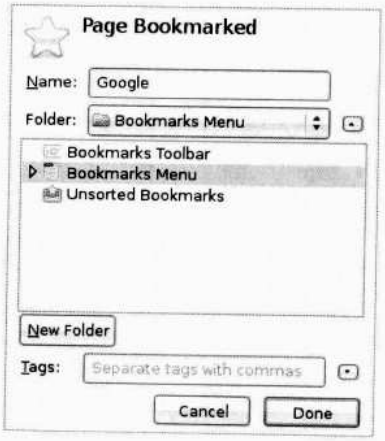


图 6.28 添加书签

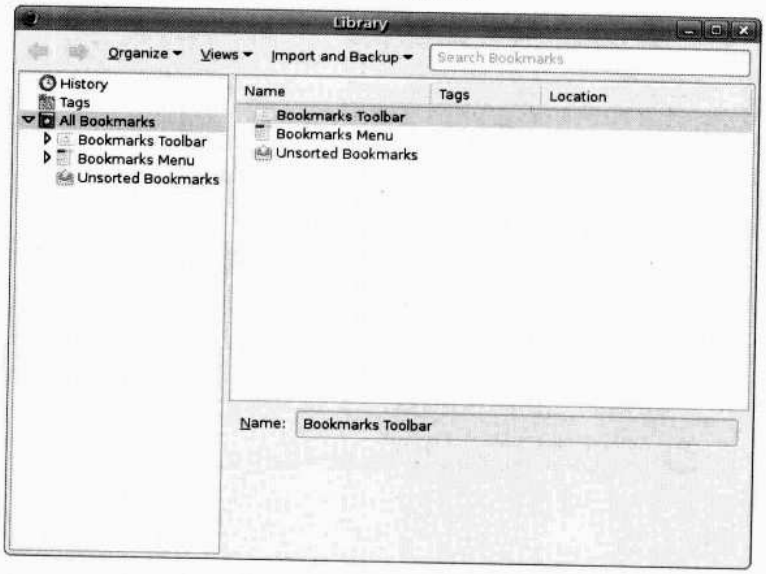


图 6.29 书签管理器（Library）

6.4.4 使用插件

Firefox 可以说是最容易定制的浏览器，用户可以定制工具栏、安装扩展软件、安装个性化主题等，这些都是通过插件来实现的。获取 Firefox 插件的途径有很多种，可以从 Mozilla 官方网站（http://addons.mozilla.org）中获取，除此之外，还可以从第三方机构的网站上获取。例如，开发工具 firebug 就是由 http://getfirebug.com/提供的。下面我们以 Adblock 插件为例，讲述一下如何使用插件管理器（Add-ons）下载并安装插件。

Step 01 在Firefox浏览器中，执行Tools | Add-ons命令，打开Add-ons窗口，默认打开Get Add-ons标签页。

Step 02 在查找的输入框内输入adblock，按回车键，Firefox开始通过网络查找adblock的相关插件，查找结果如图6.30所示。

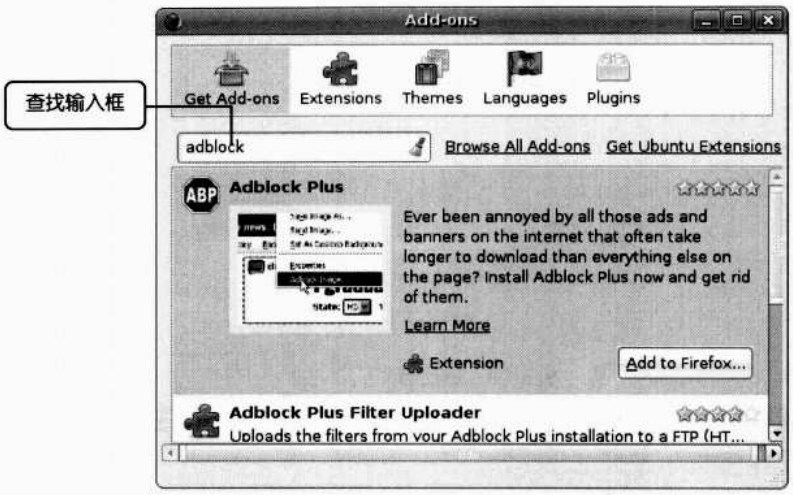


图 6.30 查找 adblock 插件

Step 03 单击查找结果中的Add to Firefox按钮，弹出Software Installation窗口，如图6.31所示。单击Install Now按钮，开始安装，直到安装结束。

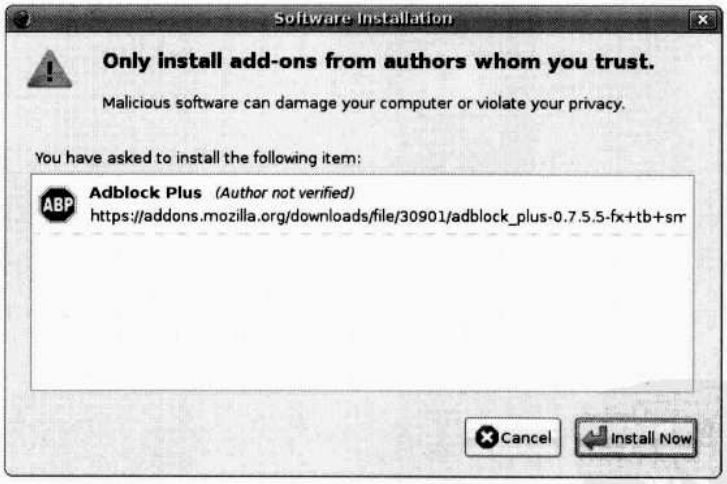


图 6.31 安装 Adblock 提示

Step 04 单击Restart Firefox按钮，将关闭系统中所有打开的Firefox浏览器。重启后，自动恢复原有访问的网页，并打开Add-ons窗口，提示用户安装了adblock新插件，如图6.32所示。

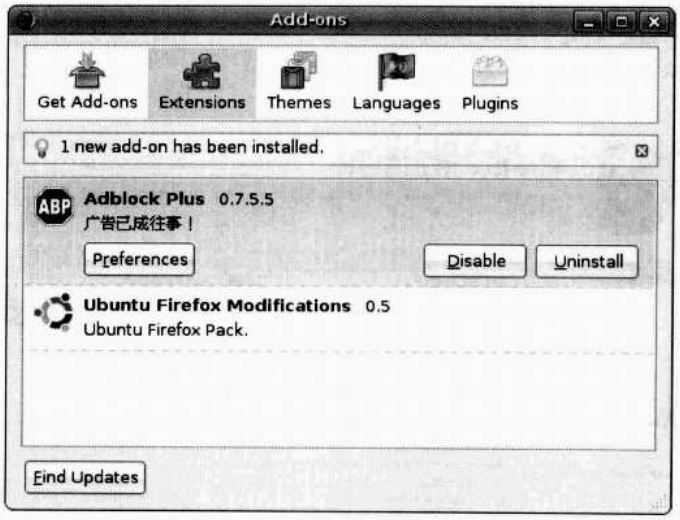


图 6.32　adblock 安装成功提示

Step 05 单击图6.32中adblock的Preferences按钮，打开【Adblock Plus首选项】窗口，便可对该插件进行相应的设置，如图6.33所示。

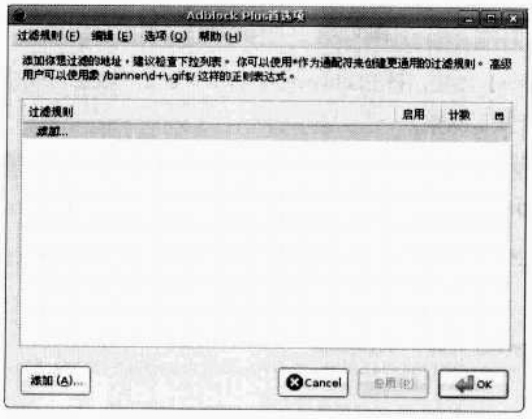


图 6.33　adblock 插件首选项

除了使用插件管理器下载安装插件外，用户还可以通过浏览器访问插件下载地址，通过网页浏览并下载插件，操作步骤和从插件管理器上下载插件类似，以上面所说的 firebug 插件为例，安装操作如下。

Step 01 在Firefox浏览器地址栏中输入http://getfirebug.com，按回车键，开始访问该网站，在网页中找到下载安装的按钮，如图6.34所示。

图 6.34 Firebug 插件页面

Step 02 单击【安装到Firefox】按钮，弹出Software Installation窗口，如图6.35所示。单击Install Now按钮，开始下载安装，直到安装结束。

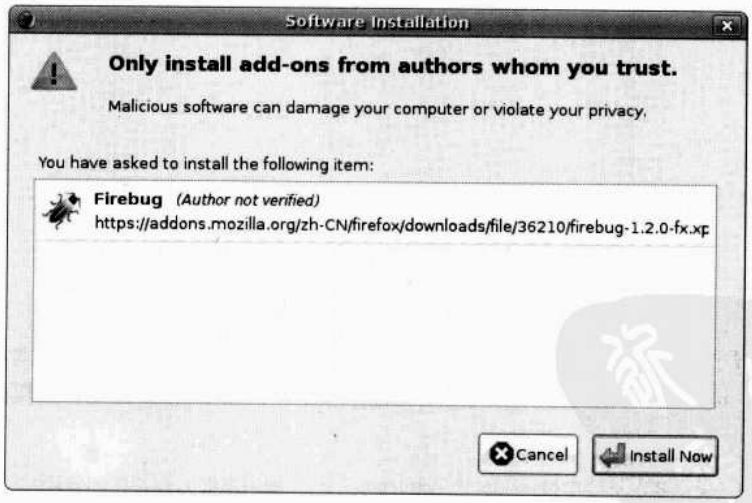


图 6.35 安装 firebug 提示

Step 03 单击Restart Firefox按钮，将关闭系统中所有打开的Firefox浏览器。重启后，自动恢复原有访问的网页，并打开Add-ons窗口，提示用户安装了firebug新插件，如图6.36所示。

图 6.36 firebug 安装成功提示

6.5 即时通讯

即时通讯（Instant Messaging，IM）是一个终端服务，允许两人或多人使用网络即时地传递文字信息、档案、语音或视频交流。即时通讯软件一般都具有友好的界面和简洁的操作，是与亲朋好友保持联系的好工具。Ubuntu 支持的即时通讯软件很多，包括 MSN、QQ、ICQ 等，在这一节中我们将着重说明 Ubuntu 下默认的通用即时通讯工具 Pidgin（Gaim）。

Pidgin 的前称是 Gaim（GTK AOL Instant Messenger），由于 AOL 公司的投诉，于是改名为 Pidgin。不过由于历史的原因，虽然这两个名称都指的是同一个软件，但 Gaim 的名声要比 Pidgin 大得多。

Pidgin 是一个可以在 Windows、Linux、BSD 和 Unixes 下运行的多协议即时通讯客户端，可以让用户用其所有的即时通讯账号一次登录。Pidgin 支持的即时通讯软件有 AIM、Bonjour、Gadu-Gadu、Google Talk、Groupwise、ICQ、IRC、MSN、MySpaceIM、QQ、SILC、SIMPLE、Sametime、XMPP、Yahoo! 和 Zephyr 等。同时 Pidgin 是免费软件，并以 GNU 通用公共许可协议发行。这意味着用户可以自由使用并修改它，但如果用户进行了修改，必须也开放修改过的源代码。

6.5.1 在 Pidgin 下使用 MSN

MSN Messenger 是微软公司推出的即时通讯软件，使用 MSN Messenger 可以与他人进行文字聊天、语音对话、视频会议等即时交流，还可以通过此软件来查看联系人是否联机，它在信息沟通方面的快捷性、便利性备受用户推崇，目前已有越来越多的人开始通过 MSN 进行在线即时沟通。

要在 Pidgin 下使用 MSN，必须正确设置账号信息。Pidgin 为用户提供了多种途径创建、配置即时通信账号的方法。在初次启动 Pidgin 的时候，会启动安装向导，引导用户一步步新建一个即时通信账号。当然，用户也可以通过账号管理器来管理账号信息。

添加账号

下面以建立 ubuntuer@live.com 的 MSN 账号为例，介绍具体的操作步骤。

Step 01 执行【应用程序】|【互联网】|【Pidgin互联网通信程序】命令，打开Pidgin主窗口，并启动安装向导的第一步，如图6.37所示。

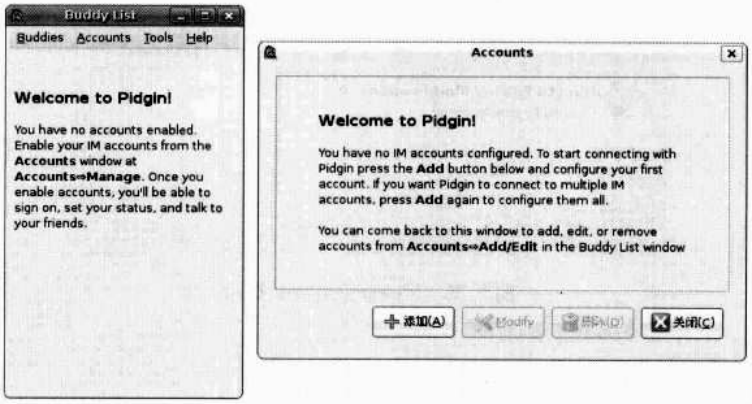


图 6.37 首次启动 Pidgin

Step 02 单击Accounts窗口中的【添加】按钮，弹出Add Account对话框，如图6.38所示。在Basic标签页设置账号的基本信息，本例中，单击Protocol微调按钮，选择MSN选项，在Screen name文本框中输入MSN的账号（ubuntuer@live.com），在Password文本框中输入账号的登录密码，在Local alias文本框中输入用户希望对方看到的名字。除此之外，如果用户希望每次登录的时候不必再次输入密码，那就需要勾选Remember password复选框；如果该账号还用于收发邮件，最好勾选New mail notifications复选框，以便能及时得到新邮件的信息。

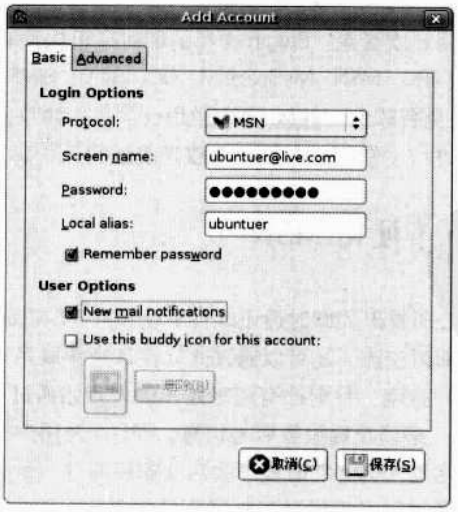


图 6.38 添加 MSN 账号的基本信息

Step 03 单击Advanced标签页，进入高级设置。Pigdin会根据用户选择的协议自动地配置MSN相关信息，包括服务器名称，使用的端口号，在这里我们使用默认值就可以了，如图6.39所示。

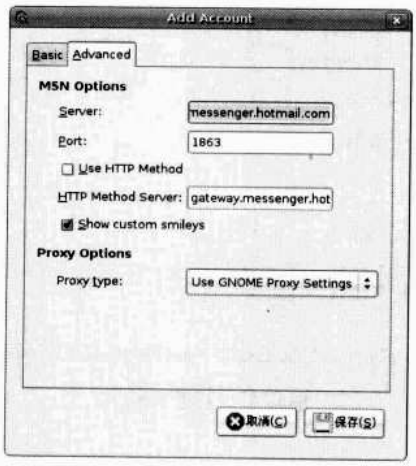


图 6.39 MSN 账号服务器的参数

Step 04 单击【保存】按钮，关闭对话框。在弹出的Accounts窗口中显示刚刚添加的MSN账号。同时Pidgin会自动连接到MSN服务器尝试登录，登录成功后，将看到好友的列表，如图6.40所示。

图 6.40 完成 MSN 账号添加

与好友聊天

成功登录 MSN 账号后，用户就可以通过 MSN 与好友进行聊天了。Pidgin 支持聊天窗口标签页的功能，用户可以在一个窗口中与多个好友聊天，这样避免出现太多的聊天小窗口占满整个屏幕。在 Pidgin 的好友列表中，单击希望与其聊天的一个或多个联系人，打开如图 6.41 所示的对话框。

图 6.41 标签页聊天

添加好友

Pigdin 提供了一个简单的添加好友的方法，执行 Buddies | Add Buddy 命令，弹出 Add Buddy 窗口，如图 6.42 所示。在 Buddy's screen name 文本框中输入好友的 MSN 账号，在 (Optional) Alias 文本框中输入好友的别名，也就是你看到的名字，在 Add buddy to group 文本框中直接输入或单击微调按钮指定好友所属的组群。最后单击【添加】按钮完成添加好友的设置，这时候你就能在好友列表中看到新加的好友了。

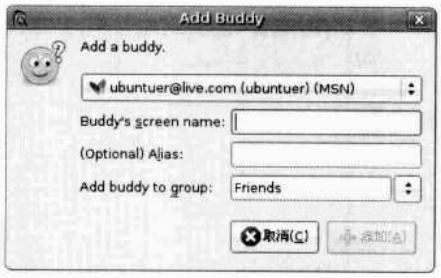


图 6.42 添加好友

6.5.2 在 Pidgin 下使用 QQ

QQ 是国内使用最广泛的互联网即时通讯工具，在微软 Windows 系统下，腾讯公司免费提供 QQ 的安装程序。在 Linux 下，QQ 的客户端软件有 Pidgin、LumaQQ 等。下面我们将着重介绍在 Pidgin 中使用 QQ 的具体操作过程。

添加 QQ 账号

添加 QQ 账号的方法与前面介绍的添加 MSN 账号的方法基本一样。

Step 01 在Pidgin主窗口中，选择菜单栏中的Accounts | Manage命令，或使用快捷键Ctrl+A打开账号管理对话

框，单击【添加】按钮，打开添加账号对话窗口。

Step 02 单击Basic标签页，进入基本设置对话框。单击Protocol微调按钮，选择QQ，在Screen name文本框中，输入QQ的号码；在Password文本框中，输入账号密码，在Local alias中，输入用户希望对方看到的名字，然后勾选Remember password复选框，如图6.43所示。

Step 03 单击Advanced标签页，进入QQ服务器配置对话框，因为Pidgin没有为QQ设置默认的访问服务器，所以，需要找到QQ的服务器的信息，包括服务器的IP地址和端口号，并把这些信息填写到相应的配置栏中，如图6.44所示。

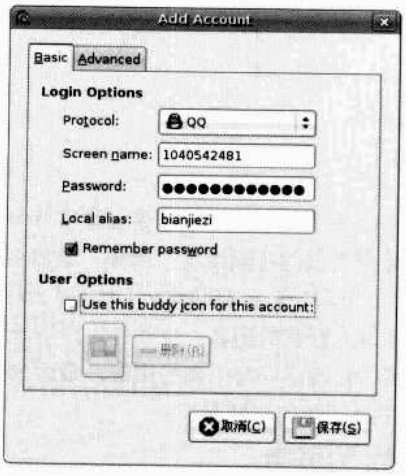


图 6.43 添加 QQ 账号

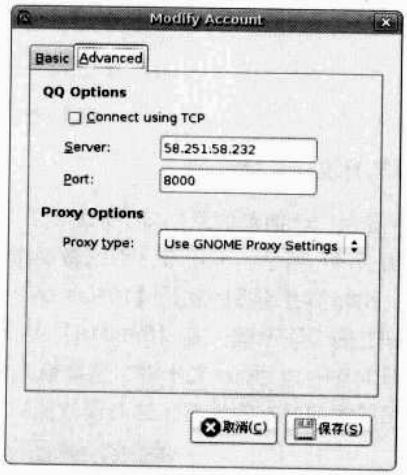


图 6.44 配置 QQ 服务器信息

Step 04 单击【保存】按钮，返回到账号管理窗口，看见新加的QQ账号，如图6.45所示。单击账号管理窗口的【关闭】按钮，完成QQ账号的配置。

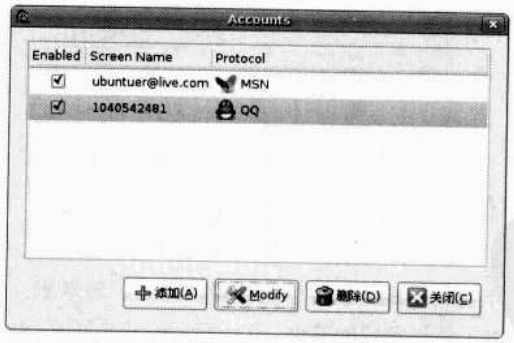


图 6.45 完成 QQ 账号的添加

与好友聊天

在 Pigdin 主窗口中，单击【QQ 好友】列表，然后双击希望与之聊天的好友，更可开始与 QQ 好友畅所欲言了，如图 6.46 所示。

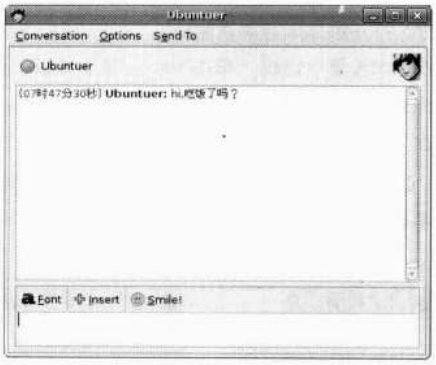


图 6.46 与 QQ 好友聊天

添加好友

Pigdin 提供一个简单的添加 QQ 好友的方法，执行 Buddies | Add Buddy 命令，弹出 Add Buddy 窗口，如图 6.47 所示。单击最上面的微调按钮，用户根据自己的实际情况，选择“添加相应 QQ 的好友”，比如作者选择的就是【1040542481 (bianjiezi) (QQ)】，在 Buddy's screen name 文本框中输入好友的 QQ 号码，在 (Optional) Alias 文本框中输入好友的别名，也就是用户所看到的名字，在 Add buddy to group 文本框中直接输入或单击微调按钮指定好友所属的组群。最后单击【添加】按钮完成添加好友的设置，这时候就能在好友列表中看到新加的好友了。

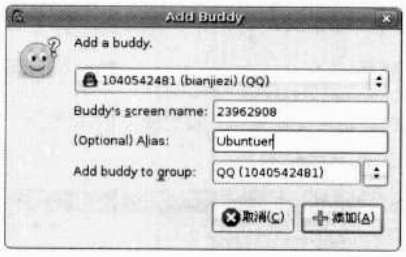


图 6.47 添加 QQ 好友

6.6 多媒体娱乐

Ubuntu 为用户提供了丰富的多媒体软件，为用户听音乐、看电影、收听 Internet 网络广播、收听播客和录制音频等都提供了各种各样的软件。这一节我们将介绍在 Ubuntu 中如何使用音频播放器听音乐和用视频播放器看电影。

6.6.1 用 Totem 观看视频

Totem 是 Ubuntu 系统默认安装的视频播放器。Totem 有定制播放列表、DVD 回放以及视频截图等功能。用户可以通过从桌面菜单系统中执行【应用程序】|【影音】|【电影播放器】命令来启动 Totem，

其主窗口如图 6.48 所示。Totem 使用 Gstreamer 解码器来读取视频文件，某些视频格式可能还需要添加其他 codec（编解码）支持，比如 rm 格式的文件。需要注意的是 rm 格式是非开源格式，虽然在 Ubuntu 下可以安装 W32codecs 来满足用户播放 rm 格式的需求，但是由于 rm 格式的版权问题，用户还是要注意在合理的范围内使用非开源工具。

图 6.48　Totem 电影播放机

6.6.2　播放音频文件

Ubuntu 默认安装一个音乐播放器（Rhythmbox），这个播放器支持广泛的音频格式（包括 mp3 和 ogg），除此之外，Rhythmbox 还支持收听 Internet 电台、收看网络播客、IPod 的整合，以及烧制 CD 和相关音频文件信息的修改。用户可以通过从桌面菜单系统中执行【应用程序】|【影音】|【Rhythmbox 音乐播放器】命令来启动 Rhythmbox 音乐播放器，其界面如图 6.49 所示。不过，在 Rhythmbox 中并不存放实际音乐文件而是将音乐文件的链接放置在中心音乐库中进行统一管理。

图 6.49　Rhythmbox 音乐播放器

6.6.3 录音机

如果你想录制一些音频，比如你的播客（Podcast）或者有趣的音频信息，都可以通过 Ubuntu 默认安装的录音机（Sound Recorder）记录下来。可以通过从桌面菜单系统中执行【应用程序】|【影音】|【录音机】命令来启动，如图 6.50 所示。

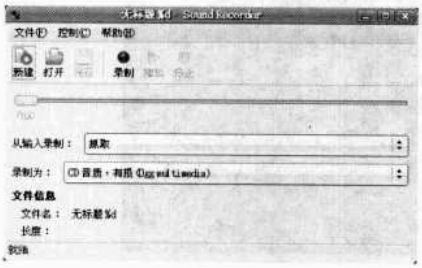


图 6.50 Ubuntu 的录音机

6.6.4 安装解码器

Ubuntu 中的播放器都使用 GStreamer 开源解码器。GStreamer 的编码译码器插件被放在不同的软件包中，并根据许可来应用不同的编码译码器。用户可以参见 GStreamer 网站。

其他应用程序，如 Mplayer 和 Xine，并不使用 GStreamer 框架。由于专利和版权的限制，这些程序的编码译码器并没有包含在 Ubuntu 中。更多相关信息可参阅 Restricted Formats wiki 页。

安装 GStreamer 解码器的方法很简单，可以使用新立得软件包管理器安装，单击新立得软件包管理器中的查找（Search）按钮，输入关键字 gstreamer 进行查找，然后在查询结果中选择相应的 gstreamer 软件包，如图 6.51 所示。如果磁盘有足够空间，建议把相关的 GStreamer 软件包都安装上。

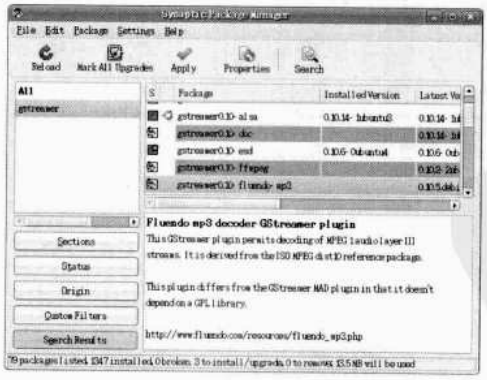


图 6.51 安装 GStreamer 解码器

当然，你也可以使用 apt-get install 命令在 Shell 下逐一安装或升级 GStreamer 解码器的各个包。

6.6.5 在 Ubuntu 中使用 RealPlayer 11

RealPlayer11 是一个很流行的音频/视频播放器。RealNetworks 公司日前发布了最新版的适用于 Linux 的 RealPlayer 11，此版本更新了如下特性：支持在 Ubuntu 下播放 Windows Media 的内容，内建集成播放列表编辑功能，支持 Perfect Play。另外，还加入了对 ALSA 和环绕立体声播放的支持。支持通过 ALSA 的 5.1 声道回放，在流媒体播放过程中，实现暂停时缓冲等。下面让我们来介绍一下如何安装 RealPlayer 11。

Step 01 从http://www.real.com/Linux下载RealPlayer11GoLD 安装包到本地目录。

Step 02 打开终端（按Alt+F2键）。

Step 03 给安装包添加可运行的权限，输入sudo chmod +x RealPlayer11GOLD.bin。

Step 04 输入sudo "./RealPlayer11GOLD.bin"。

Step 05 安装开始，根据安装提示完成RealPlayer 11的安装。

Step 06 完成安装后，通过执行【应用程序】|【影音】| RealPlayer 11命令启动RealPlayer 11。

Step 07 第一次进入RealPlayer11，RealPlayer11将启动设置助理，按照提示完成设置后进入RealPlayer11，如图6.52所示。

图 6.52 安装成功后的 RealPlayer

6.7 课后练习

1. Ubuntu 中主推的办公应用软件是什么？
2. 什么是 Openoffice 套件？Openoffice 包括哪些套件？它们的功能分别是什么？
3. 如何在 Ubuntu 中配置 Evolution？如何通过 Evolution 收发邮件？
4. 如何在 Ubuntu 中使用 gThumb 查看浏览图片？
5. 如何在 Firefox 下浏览网页，安装插件？
6. 请问如何在 Ubuntu 中安装配置 Pidgin？如何在 Pidgin 配置下使用 MSN 账号进行聊天？如何在 Pidgin 配置下使用 QQ 账号进行聊天？
7. Ubuntu 中包含哪些多媒体娱乐软件？请尝试用 Totem 观看视频文件。
8. 如何在 Ubuntu 中安装 Realplayer 软件？

Chapter07 Ubuntu 添加/删除程序及软件包管理

应用程序的添加、删除以及软件包的管理是 Ubuntu 使用中经常遇到的操作之一，因此掌握它们非常重要。其实，在前面介绍 Ubuntu 桌面环境及设置和 Ubuntu 应用软件两章中，我们已经接触到一点应用程序安装/删除的操作了。因此在本章中，将循着上两章的疑问，首先介绍 Ubuntu 应用程序安装/删除工具“添加/删除程序”的使用；第二节的主要内容是 Ubuntu 软件包的基础知识；从第三节开始系统地介绍管理 Ubuntu 软件包的安装、更新、删除等操作的软件包管理工具；另外，简单地介绍一下软件库；在本章的最后，将说明如何更新 Ubuntu 系统。

7.1 应用程序的安装/卸载

相对于其他 GNU/Linux 套件来说，Ubuntu 的一大优势是应用程序的易用性，包括其方便的安装与卸载。在 Ubuntu 中，应用程序的安装几乎和 Microsoft Windows 一样，轻点鼠标就可以使用“添加/删除程序”这个工具来完成用户在 Ubuntu 系统上应用软件的安装和卸载工作。

7.1.1 添加/删除程序概述及启动

“添加/删除程序”是在 Ubuntu 中安装和删除应用程序的简单图形化方式。这个工具功能非常强大，它会自动记录 Ubuntu 系统可安装的应用程序，并对其进行分类，同时还供用户进行检索；同时，系统还记载了各个应用软件的安装情况，当用户选择某个应用软件时，即可进行相应的安装或删除操作。通过“添加/删除程序”这个工具，用户在 Ubuntu 上安装软件将非常容易。

要启动“添加/删除程序”，需在桌面菜单系统中执行【应用程序】|【添加/删除程序】命令。启动后，添加/删除程序会先检测一下 Ubuntu 系统可用的应用程序，如图 7.1 所示。

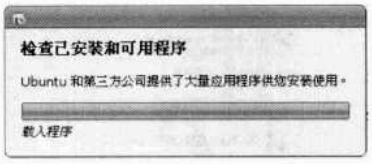


图 7.1 添加/删除程序启动检索窗口

检测完毕后，程序将进入如图 7.2 所示的界面。

窗口包括三大部分，左侧列出大的应用软件类别，右侧上半部分列出左侧指定类别对应的具体应用软件，右侧下半部分是对选中的应用软件的具体说明。默认打开状态下，因为没有选中任何应用软件类别或任何应用软件，这时窗口右侧下半部分是快速指南。

当系统已经安装某个软件时，在对应的软件列表中对应软件名称前边的复选框就被勾选。当某个应用软件名其前端对应的复选框没有被勾选时，就说明这个软件没有被安装。

图 7.2 添加/删除程序

因此当用户要安装某个软件时，在左边选择所属类别，然后勾选要安装的应用程序对应的复选框，然后单击右下方的【应用改变】按钮，即可进入安装程序。相反，当我们要删除某个应用程序时，我们只需要取消选中该应用程序前端的复选框，然后单击【应用改变】按钮即可进入卸载程序流程。下面我们结合例子来分别说明安装和删除应用程序。

7.1.2 安装应用程序

下面我们通过一个实例来说明应用程序的安装流程。

假如用户在添加/删除程序窗口中，选中了左侧应用软件分类中的“互联网”，然后在右侧的软件类别中，选中 XChat–GNOME 聊天，如图 7.3 所示。

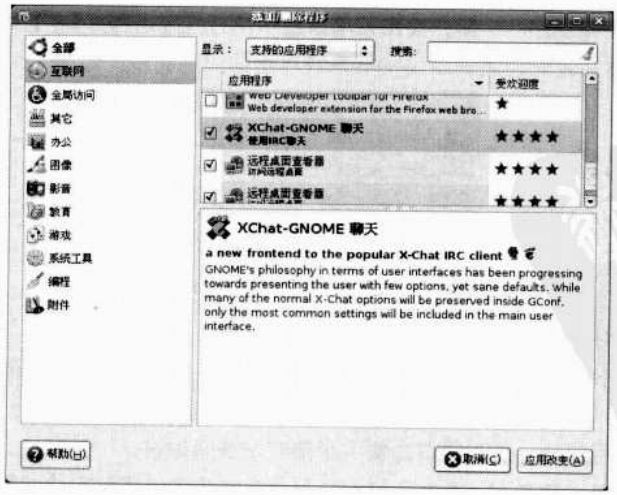


图 7.3 选择 XChat–GNOME 进行安装

上图中因为系统没有默认安装“XChat—Gnome 聊天”，因此这里选中该复选框，完成后单击【应用改变】按钮，“XChat—Gnome 聊天”将被程序自动下载并安装，同时安装其他被要求的附加程序，如图 7.4 所示。

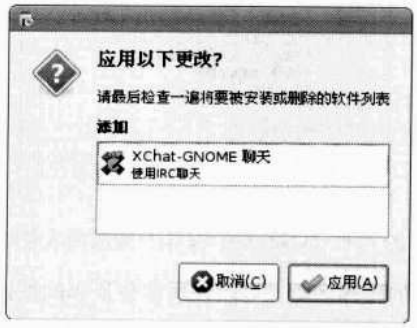


图 7.4　安装 XChat—GNOME 聊天确认

确定所选后单击【应用】按钮，因为安装程序需要用户具有管理员的权限，因此，安装程序会提示输入管理员密码，输入密码确认后，安装程序将开始下载安装的软件包，如图 7.5 所示。

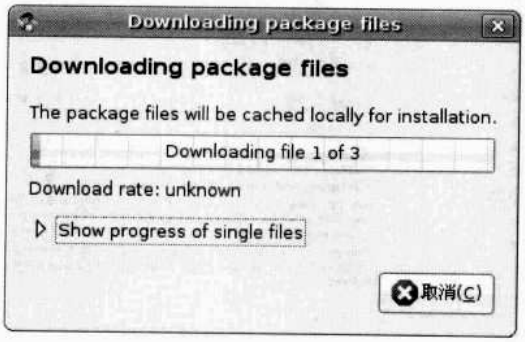


图 7.5　下载 XChat—GNOME 相应软件包窗口

下载完成后，安装程序自动进入程序安装流程，如图 7.6 所示。

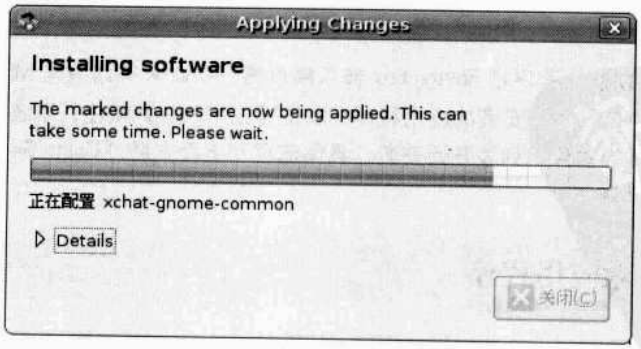


图 7.6　XChat—GNOME 安装配置过程窗口

安装完成后，系统弹出安装【新的程序已经被安装】窗口，如图 7.7 所示。单击【关闭】按钮，

即可开始使用新安装的应用程序了。

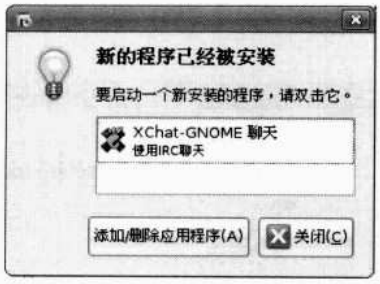


图 7.7 XChat—GNOME 安装完成确认窗口

当然，如果用户知道想要的程序名称，可以使用窗口顶部的搜索工具，然后搜索想安装软件的列表，比如输入关键字 MP3，这时添加/删除程序会自动搜索与 MP3 对应的应用程序，并在右侧上半部的应用程序类别中列出来，如图 7.8 所示。

图 7.8 通过搜索获取相应应用程序列表

可以看出，添加/删除程序把 Rhythmbox 音乐播放器、Amarok 等所有与 MP3 相关的应用程序都列出来了。这时选中自己想要安装的应用程序，单击【应用改变】按钮，系统将会要求用户确认安装，并显示将要安装的软件包列表开始安装，具体流程和上面安装 XChat—Gnome 聊天程序完全一样，在这里就不再重复了。

7.1.3 删除应用程序

当要删除某个应用程序时，只需要在【添加/删除程序】窗口中找到相应的应用程序，取消选中该应用程序前端的复选框，然后单击【应用改变】按钮即可进入卸载程序流程。下面简单地通过卸载在线字典这个例子来说明一下删除应用程序的流程。

首先选择“互联网”应用软件类，然后在右侧软件列表中选择【字典】，然后取消选中其前端复选框，这时【添加/删除程序】窗口中的【应用改变】按钮变亮可用，单击该按钮即可进入卸载在线字典流程。首先弹出提示是否删除“字典”的对话框，如图 7.9 所示。

图 7.9　卸载应用程序字典

单击【应用】按钮，确认要删除字典软件，这时会提示输入管理员密码，输入密码后，安装程序将开始卸载软件，如图 7.10 所示。

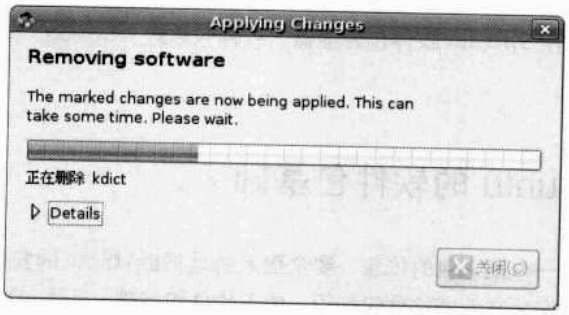


图 7.10　删除应用程序对应的文件

卸载完毕后，程序将弹出如图 7.11 所示的“程序已经被成功地移除了”窗口，提醒用户卸载成功。

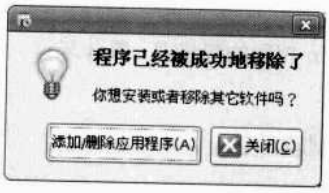


图 7.11　确认删除应用程序

需要说明的是，Ubuntu 应用程序中，一些应用程序是相互依赖的，当要卸载有多个应用程序依赖的应用程序时，系统会提示系统无法删除应用程序。如图 7.12 中，我们试图卸载应用程序“Pidgin 互联网通讯程序”，但是因它与其他应用程序存在依赖关系，系统提示“无法删除 pidgin”。

图 7.12 无法删除应用程序提示

除此之外，如果用户没有激活在线软件包库，也许会被要求插入 Ubuntu CD-ROM 以便安装某些软件包。一些应用程序和软件包不能通过添加/删除程序来安装。如果不能找到要找的软件包，单击【高级】按钮，打开 Synaptic 软件包管理器。软件包依赖、Synaptic 软件包管理器这些内容将是我们后面内容的主题。

7.2 Ubuntu 的软件包基础

上节中，我们涉及一个概念包的依赖，要全面系统地把握 Ubuntu 的安装/删除及包管理，我们需要了解一定的 Ubuntu 安装文件即软件包知识。关于软件包基础，具体包括软件包的具体类型、软件包的一般命名规则、Ubuntu 中各个软件包的依赖关系，以及软件包在 Ubuntu 中的状态等。

7.2.1 软件包类型

Ubuntu 软件包采用了和 Debian 相同的包格式。具体来说，Ubuntu 安装时涉及最多的是以下两种类型的软件包。

Binary packages（二进制软件包）

二进制软件包包含可执行文件、配置文件、man/info 页面、版权声明和其他文档。这些软件包以 Ubuntu 特定的格式发布，它们通常使用.deb 为扩展名以示区别。这种二进制软件包可使用 Ubuntu 工具 dpkg 解包。

Source packages（源码包）

源码包包含一个.dsc 文件——它用于描述源码包（包括下列文件的名称），一个.orig.tar.gz 文件——它是未经修改的原始源代码压缩文件，以及一个.diff.gz 文件——它包含了该软件包 Ubuntu 化时所做的修改。可以使用 dpkg-source 工具打包/解包 Ubuntu 源码包。

7.2.2 软件包命名约定

在 Ubuntu 系统中，对软件包的命名也遵循一定的约定。举个例子来说，假如有一个软件包名字叫 foo_ver-rev_arch.deb。

其中：

foo 是软件包的名称；
ver 是软件本身的版本号；
rev 是 Ubuntu 修订版本号；
arch 是目标架构名称。

关于软件包的具体信息，可以通过如下两个命令来查询。

- dpkg --info filename：找出文件 filename 实际是哪个软件包。
- dpkg --status package：列出一个软件包中包含哪些默认配置文件。

Ubuntu 有个机制，如果系统中安装了多个提供同种虚拟包的软件包，系统管理员可以指定一个为首选软件。

7.2.3 软件包依赖关系

Ubuntu 系统中，软件包之间往往有一定的关系，软件包管理系统依赖声明描述了这一事实：一些软件包需要安装其他软件包才能正常运行或运行得更好。在后面使用 Apt、Aptitude 等安装管理工具进行软件安装时，这些工具会考虑软件包的依赖关系。具体来说，软件包的依赖关系有如下几种情况。

软件包 A 依赖（depends）软件包 B

要运行 A 必须安装 B。在有些情况下，A 不仅依赖 B，还要求 B 的特定版本。版本依赖通常有最低版本限制，A 更依赖于 B 的最新版而非某个特定版本。

软件包 A 推荐（recommends）软件包 B

软件包维护者认为所有用户都不会喜欢缺少某些功能的 A，而这些功能需要 B 来提供。

软件包 A 建议（suggests）软件包 B

B 中某些文件与 A 的功能相关，并能够增强 A 的功能。这种关系通过声明软件包 B 增强（Enhances）软件包 A 来表示。

软件包 A 与软件包 B 冲突（conflicts）

如果系统中安装了 B 那么 A 无法运行。Conflicts 常和 replaces 同时出现。

软件包 A 替换（replaces）软件包 B

B 安装的文件被 A 中的文件移除和覆盖了。

软件包 A 提供（provides）软件包 B

A 中包含了 B 中的所有文件和功能。

在处理依赖关系上，apt-get 会自动下载安装依赖的软件包，但是不会处理所安装软件推荐的或者建议的软件包；相反，aptitude 可以设置成安装所安装软件推荐或者建议的软件包；dselect 给用户列出了所安装软件推荐或建议的软件包，可以进行单独选择。dselect 可以对 recommends 和 suggests 类软件包进行细操作，apt-get 只会简单地下载安装 depends 类软件包而不管 recommends 和 suggests 类软件包。这两个程序均正式使用 APT 作为其后台。

dpkg 通常将归档文件随意解包，不顾依赖性。如果是 Pre-Depends 包，则在所依赖的其他包被解包和配置之前，Pre-Depends 包不会被解包。关于 apt、dselect 和 dpkg 等软件包管理工具，我们将在下节中详细介绍。

7.2.4 软件包状态

在 Ubuntu 软件包管理中，如后面要重点介绍的 Synaptic，会列出软件包在系统中的状态，一个软件包在 Ubuntu 的存在状态值有如下几种。

- unknown：用户并有没描述他想对软件包进行什么操作。
- install：用户希望对软件包进行安装或升级。
- remove：用户希望删除软件包，但不想删除任何配置文件。
- purge：用户希望完全删除软件包，包括配置文件。
- hold：用户希望软件包保持现状。

7.3 软件包管理工具概述

在 Ubuntu 中，应用程序的添加或删除一般是通过软件包管理器进行的。通过将软件的安装文件处理成为 Ubuntu 优化的预配置软件包，软件包管理器工具可以容易地安装和删除这些应用程序。根据用户交互方式的不同，Ubuntu 主要有如下几种包管理工具，如表 7.1 所示。

表 7.1 软件包管理工具

工具	简要说明
dpkg	Debian 包安装工具
apt-get	APT 的命令行前端
Aptitude	APT 的高级的字符和命令行前端
Synaptic	图形界面的 APT 前端
Dselect	使用菜单界面的包管理工具
Tasksel	Task 安装工具

注意 在 Ubuntu 中一次只能运行一个软件包管理应用程序。例如，如果运行“添加/删除程序”并试着启动“更新 Ubuntu”的话，它将会因出错而失败。在重新启动新的软件包管理应用程序之前请关闭正在运行的软件包管理应用程序。另外，这些工具不是用来取代对方的，比如可以用 Dselect，也可以使用 APT 和 dpkg。

在本章后续章节中，我们会重点介绍图形化的 Synaptic 和强大的命令行程序 APT。本节只是先对各个管理工具进行一下概述，在后续节次中将对其进行专门介绍。

7.3.1 Synaptic

Synaptic Package Manager（新立得软件包管理器）是一个高级软件包管理应用程序，它可以安装和删除系统的每个软件包。界面就像“添加/删除程序”一样是图形化的，但可以向用户显示更多信息，这就意味着 Synaptic 可以完全控制系统的软件包管理。

Synaptic 是目前首选的基于 GTK 的图形化 APT 前端程序。它的包过滤器比 Aptitude 的好用多了。它包含了对 Debian Package Tags 的实验性支持。

7.3.2 APT

APT（The Advanced Packaging Tool，高级软件包工具），是一个强大的包管理系统，图形化程序如添加/删除应用程序和 Synaptic 都是建立在其基础之上的。APT 会自动处理依赖关系，运行 APT 时，要求用户具有管理员权限。

APT 由几个名字以“apt-”打头的程序组成，其中 apt-get、apt-cache 和 apt-cdrom 是处理软件包的命令行工具，它们也是其他用户前台程序的后端，如 Dselect 和 Aptitude。

7.3.3 Aptitude

Aptitude 为一个高级包管理工具，也是目前首选的字符界面 APT 前端程序。它会记住哪些包是你安装的，哪些是为了满足依赖关系而安装的；在已安装包不需要的情况下，Aptitude 会自动卸载后者。Aptitude 是全新的菜单操作的包安装工具，和 Dselect 类似，但是是针对 APT 从头设计的。它内建一套高级的包过滤器，但是比较难上手。

对大多数参数来讲，Aptitude 完全可以作为 apt-get 的一个兼容的代替品。但一旦开始使用 Aptitude，最好一直使用它，而不是选择其他替代工具。否则将失去 Aptitude 包存放的软件安装清单，就不能享受自动删除多余软件包的功能了。

全屏状态下 Aptitude 接受单键的命令大多数是小写的。主要的几个功能键如表 7.2 所示。

表 7.2 Aptitude 的命令描述

按键	动作
F10	菜单
?	按键命令帮助（完整的清单）

（续表）

按键	动作
U	更新软件包信息
+	标记软件包为升级或者新安装
-	标记软件包为删除（保留配置文件）
_	标记软件包为完全删除（删除配置文件）
= = =	保持软件包的当前版本，阻止其被升级
U	标记所有可以升级的软件包为升级
G	下载和安装选择的软件包
Q	退出当前屏幕，保存改变
X	退出当前屏幕，忽略改变
Enter	查看一个软件包的信息
C	查看一个软件包的更新日志
L	改变软件包树状显示限制
/	搜索第一个匹配的软件包
\	重复最后一次搜索

和 apt-get 一样，Aptitude 安装软件包的时候自动解决依赖问题。Aptitude 还能安装即将安装的软件包推荐或者建议的软件包。通过执行 F10 |【选项】|【处理依赖关系】命令来在菜单上更改这一默认设置。Aptitude 还有以下特点：

- Aptitude 能访问所有版本的软件包。
- Aptitude 的动作记录在/var/log/aptitude。
- Aptitude 能轻松地追踪陈旧的和本地建立的软件包，并在“过期的和在本地创建的软件包”上列出。
- Aptitude 内建强大的包搜索和显示功能。熟悉 mutt 的用户很容易上手，因为这个显示方法的灵感来源于 mutt。

7.3.4 dpkg

dpkg（Debian Package）是为 Debian 系统专门开发的软件包管理工具，以便软件的安装、更新及删除。以下是 dpkg 命令最常用的用法。

dpkg -i <package>

安装一个 Debian 软件包，当然这个软件包应该是在本地文件系统上存在。

dpkg -c <package>

列出 <package>的内容。

dpkg -l <package>

从 <package> 中提取软件包信息。

dpkg -r <package>

移除一个已安装的软件包。

- dpkg -P <package>

完全删除一个已安装的软件包。和 remove 不同的是，remove 只是删掉数据和可执行文件，purge 另外还删除所有的配置文件。

- dpkg -L <package>

列出 <package> 安装的所有文件清单。同时还使用 dpkg -c 来检查一个.deb 文件的内容。

- dpkg -s <package>

显示已安装软件包的信息。同时还使用 apt-c 显示 Debian 文件中的软件包信息，使用 dpkg -I 来显示从一个.deb 文件中提取的软件包信息。

- dpkg-reconfigure <package>

重新配制一个已经安装的软件包，如果它使用的是 debconf（debconf 为软件包安装提供了一个统一的配制界面）。

7.3.5 Dselect

Dselect 是 Ubuntu 软件包管理系统中菜单驱动的用户界面，特别适用于首次安装和大面积升级。Dselect 以前是主要的包维护工具，但现在可以考虑用 Aptitude 代替。

当启动程序的时候，Dselect 会自动选择所有 Required、Important 和 Standard 类型的包。Dselect 的用户界面有些奇怪，不过大部分人已经习惯了。它主要有 4 个命令，如表 7.3 所示。

表 7.3 Dselect 功能键描述

按键	描述
Q	退出
R	撤销
D	恢复到前一次升级状态建议按照预先设定进行操作
U	按照建议进行操作

注意 使用 D 和 U，可以选择有冲突的选项。请小心使用这个命令。如果用户的机器运行 Dselect 的速度很慢，可以考虑在速度快一点的机器上运行 Dselect，确定要安装的软件包之后，在慢的机器上通过 apt-get 来安装它们。

7.4 新立得软件包管理器

下面重点介绍 Ubuntu 应用最广的窗口化软件包管理工具——新立得软件包管理器（Synaptic Package Manager）。新立得软件包管理器是一个高级软件包管理应用程序，它可以安装和删除系统的每个软件包。特别是当需要安装“添加/删除程序”中没有列出的程序时，新立得软件包管理器

是最佳选择。

7.4.1 启动 Synaptic

要启动 Synaptic，在桌面菜单系统中执行【系统】|【系统管理】|【Synaptic 软件包管理器】命令，这时即可启动 Synaptic。需要说明的是，运行 Synaptic 需要用户具有管理员权限，所以在用户要打开 Synaptic 时，会弹出一个要求输入 root 用户密码的窗口，如图 7.13 所示。

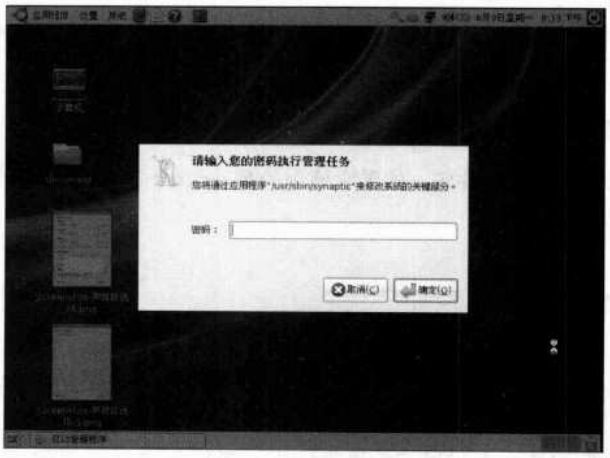


图 7.13 启动 Synaptic——输入管理员密码

输入正确密码后，即可看到如图 7.14 所示的 Synaptic 的主窗口。

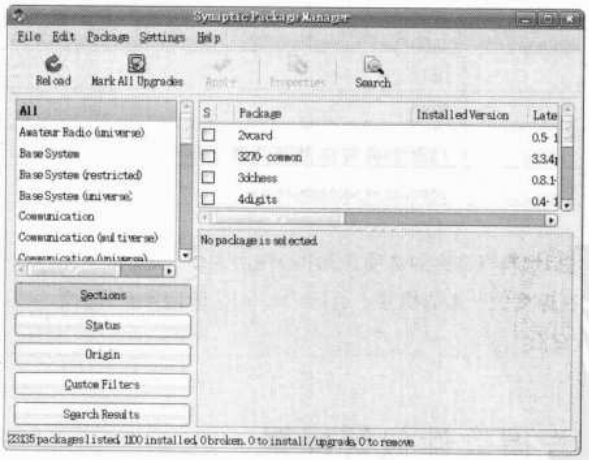


图 7.14 Synaptic 主窗口

Synaptic 的界面和"添加/删除程序"工具主窗口的布局有点相似。Synaptic 的界面分成 4 部分，最重要的两个是位于左边的软件包类和右边的软件包。单击左边的软件包类，右侧就会列出对应的软件包。关于软件包的分类可以依据 Sections、Status、Origin、Custom Filters、Search Results 来进

行区分。

7.4.2 通过 Synaptic 安装及管理软件包

要安装软件包，只需选择分类，找到列表中的软件包名，右击它并选择【标识以便安装】命令。如果用户对其选择感到满意，就可以在顶部的工具栏中单击 Apply 按钮。然后 Synaptic 将在线从软件库或 Ubuntu 安装 CD 中下载并安装所需软件包。

下面以安装 linux-libc-dev 为例进行 Synaptic 软件包安装的说明。

首先选择 Sections，然后再选择 Development 软件包类，然后在右侧的包列表中选择 linux-libc-dev，然后单击右键，在快捷菜单中选择 Mark For Install，这时界面如图 7.15 所示。

图 7.15 Synaptic 主窗口——选择安装 linux-libc-dev

做好一切准备后，单击工具栏上的 Apply 按钮，弹出如图 7.16 所示的窗口。

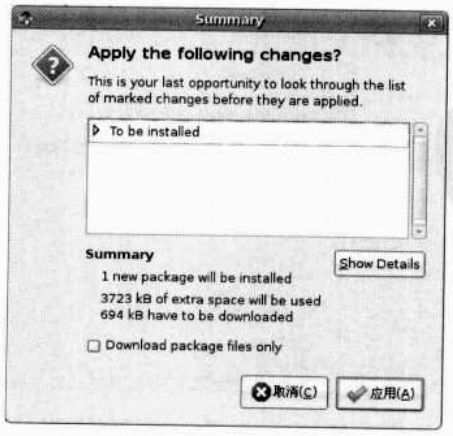


图 7.16 确认是否安装所选择的软件包

窗口提示 linux-libc-dev 包的一些信息，并提示将要安装的软件包列表，可以单击 Show Details 按钮查看详细信息，单击【应用】按钮，系统提示进入在线下载状态，如图 7.17 所示。

图 7.17 linux-libc-dev 相关文件下载界面

软件包下载后，系统会进行自动地配置安装，安装完成后，弹出如图 7.18 所示的提示窗口。

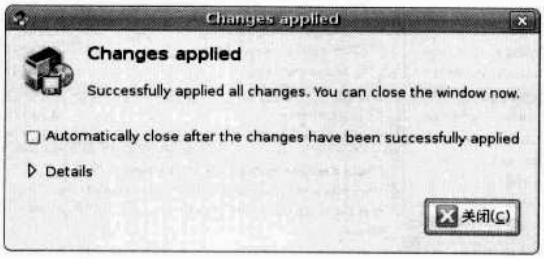


图 7.18 linux-libc-dev 安装完成界面

单击【关闭】按钮即完成了安装流程。

当然，我们也可以单击工具栏中的 Search 按钮，在 Search 文本框中输入软件包名或一个简短的搜索术语，并单击 Search 按钮，如图 7.19 所示。

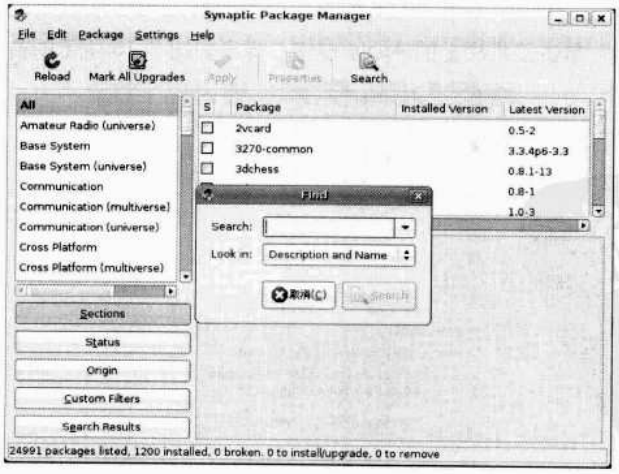


图 7.19 Synaptic 的 Search 操作界面

Synaptic 现在将根据用户输入的关键字去查找相关的软件包并把结果展现在右侧的子窗口中，这对用户来说要比浏览非常多的应用程序列表要方便许多。

上面介绍了软件包的安装，其实 Synaptic 还支持软件包的更新、删除等，具体操作和软件包的安装类似，在选中相应软件包后，单击右键，可以看见如图 7.20 所示的快捷菜单。

图 7.20　Synaptic 的可执行的软件包管理功能菜单

从菜单中我们可以看出 Synaptic 提供了完整的软件包管理操作，其各项功能如表 7.4 所示。

表 7.4　Synaptic 快捷菜单功能描述

菜单项	描述
Mark for Installation	安装软件包
Mark for Reinstallation	重新安装软件包
Mark for Upgrade	升级软件包
Mark for Removal	删除软件包（同时删除配置相关文件）
Mark for Complete Removal	全部删除软件包（删除所有软件相关的文件）

7.5　软件包的命令行安装及管理

安装应用程序的首选方式是使用“添加/删除程序”或新立得软件包管理器，它们都是图形化操作，操作简单、易于上手。当然，有时用户可能工作在 Shell 环境下，或由于其他客观条件，必须通过命令行方式从网站下载软件包，并对软件包进行安装维护。

7.5.1 Apt/Aptitude 安装更新卸载程序包

Linux 安装软件包文件有许多不同的类型。它们多数与特定 Linux 发行版的软件管理器相关联，如 Debian 软件包文件（.deb 文件）、Redhat 软件包管理器文件（.rpm 文件）和 Tarballs（.tar 文件）。本部分内容将主要介绍如何安装这些单个文件。

同时，也可以通过命令行安装一些软件包集合，这些集合是由使 Ubuntu 系统满足某些特定用途的典型软件包组成的，可称为 tasks。在初始化安装中，安装 tasks 最简单的方法就是使用 tasksel，在使用之前，用户需要运行 dselect update。但是本书建议使用 Aptitude 来安装 tasks，因为它允许用户在选择好 tasks 并准备安装之前，删除 tasks 中的某些软件包。

Apt 的一些常用操作命令

查看软件的信息：sudo apt-cache showsrc <package>
获得源代码：sudo apt-get source <package>
安装软件包：sudo apt-get install <package>
删除软件包：sudo apt-get remove <package>
获取新的软件包列表：sudo apt-get update
升级有可用更新的系统：sudo apt-get upgrade
列出更多命令和选项：apt-get help

Aptitude 软件管理程序的基本使用

安装软件：sudo aptitude install <package>
删除软件：sudo aptitude remove <package>
搜索/查找指定软件：sudo aptitude search <keywords>
更新软件列表（添加/删除源后必须进行的操作）：sudo aptitude update
更新已安装的软件包：sudo aptitude upgrade
更新整个系统：sudo aptitude dist-upgrade

7.5.2 安装/卸载 deb 包

Ubuntu 相关的软件包文件名使用.deb 作为后缀，这是因为 Ubuntu 与 Debian GNU/Linux 发行版有着紧密的关系。当用户具有管理员权限时，可以通过命令行方式或双击.deb 文件来安装.deb 软件包。

安装软件包

要安装.deb 文件，可以在文件浏览器里用鼠标双击要安装的软件包，然后选择【安装软件包】即可。或者，也可以打开一个终端并输入：

```
sudo dpkg -i package_<package>
```

删除软件包

要卸载.deb 文件，可以在软件包管理器中反选它，然后选择卸载操作，或者在终端输入：

```
sudo dpkg -r <package>
```

7.5.3 使用源代码包安装程序

以.tar.gz 或.tar.bz2 作为后缀名的文件是在 Linux 和 Unix 中被广泛使用的 Tarballs 打包文件。如果在任何 Ubuntu 软件库中都没有 Ubuntu 的软件包，用户可以按照软件包自带的指示使用命令行来安装和卸载 Tarballs 文件。Tarballs 通常包含程序的源代码，并且需要编译才能使用。要做到这一点，一般需要 gcc、g++等软件库。下面介绍一下 Ubuntu 中源代码包的操作。

源码包软件的安装

首先对源码包进行相应的解压操作，不同的压缩包有不同的解压操作方式（具体操作请查看第12章）。解压后，从源码编译安装软件，在进行源码编译安装前请确认已经建立好必要的编译环境（比如安装必需的 Libraries、Compilers、Headers）。

编译及安装工作的步骤如下：

Step 01 进入到解压目录下：cd /${解压路径}。

Step 02 配置软件的编译环境：./configure。

Step 03 编译：make。

Step 04 安装：make install。

如果在上面的操作过程中出现了异常，那就得通过别的方式（比如软件提供者的邮件列表、Ubuntu 论坛等）来了解相关源代码的缺陷或限制，然后再做处理。

由源码包制作 deb 包

如果可以从源代码正常编译/安装/使用该软件，可以考虑将它制作成 deb 包，以供将来使用。需要注意的是，用下面的方法制作出来的 deb 包尽量不要提供给他人使用，因为这个跟用户自身系统的编译环境有很大的关系。

Step 01 在制作deb包前，需要安装打包工具checkinstall，在命令行下输入下面的命令以安装checkinstall软件包。

```
sudo aptitude install checkinstall
```

Step 02 重新配置编译安装软件包，然后执行打包命令。

```
cd /${解压路径}
./configure
make
checkinstall  //执行打包命令
```

Step 03 将最后得到的.deb格式的软件包进行备份，它可以通过下面的方法进行安装。

```
sudo dpkg -i package.deb
```

注意 这里只是讲了最基本的操作，由于各种软件编译所需的开发包和编译参数各不相同，故在实际运用时需随机应变。

7.5.4 rpm 文件包的转换使用

文件名以.rpm 为后缀的文件是 Red Hat 软件包文件，我们不建议在 Ubuntu Linux 系统中安装它们。因为在绝大多数情况下，Ubuntu 自身的 deb 软件包是够用的。然而，如果有时候用户想要使用的软件并没有被包含到 Ubuntu 的软件库中，而程序本身也没有提供让 Ubuntu 可以使用的 deb 包，用户又不愿意从源代码编译，但软件提供了 rpm 包的话，在这种不得已的情况下，也可以在 Ubuntu 中安装 rpm 包。这时则需要先用 alien 把 rpm 文件包转换成 deb 文件，需要注意的是，用 alien 转换的 deb 包并不能保证 100%顺利安装。

将 rpm 文件包转换为 deb 文件包的具体步骤如下：

Step 01 安装 alien 程序，默认情况下Ubuntu并没有安装alien工具。在终端输入：

```
sudo apt-get install alien
```

Step 02 在终端使用管理员权限运行以下命令。

```
sudo alien package_file.rpm
```

上面的命令将rpm文件包转换为deb文件包，完成后会生成一个同名的package_file.deb文件。

Step 03 使用dpkg来安装转换后的软件包。在终端输入：

```
sudo dpkg -i xxxx.deb
```

7.5.5 源的添加和使用

在安装文件之前确认所下载的文件来自一个安全的源是非常重要的。如果软件包源配置不对，就可能无法保证这些文件与用户的系统兼容，如果安装这些文件的话，用户也将无法得到安全更新。也正是有鉴于此，如果想安装程序的话，请尽可能通过软件包管理器来安装由 Ubuntu 自身提供的应用程序软件包。下面简单介绍一下软件包源的相关操作。

备份当前的源列表文件

```
sudo cp -p /etc/apt/sources.list /etc/apt/sources.list_backup
```

用文本编辑器编辑源列表文件

Ubuntu（GNOME）用户：

```
sudo gedit /etc/apt/sources.list
```

Kubuntu（KDE）用户：

```
sudo kate /etc/apt/sources.list
```

其他系统用户，比如 Xubuntu：

```
sudo nano /etc/apt/sources.list
```

将以下内容作为模板，自己编辑 sources.list 中的软件源

用户可以试着用以下这些行来代替 sources.list 中当前的内容，要想使用本地的镜像，可以在 archive.ubuntu.com 之前添加“cc.”，其中 cc 是用户所在国家的代码。例如：

```
    deb http://lv.archive.ubuntu.com/ubuntu feisty main restricted universe
multiverse
    ## See http://help.ubuntu.com/community/UpgradeNotes for how to upgrade to
    ## newer versions of the distribution.

    ## Add comments (##) in front of any line to remove it from being checked.
    ## Use the following sources.list at your own risk.

    ## Uncomment deb-src if you wish to download the source packages

    ## If you have a install CD you can add it to the reposity using 'apt-cdrom add'
    ## which will add a line similar to the following:
    #deb cdrom:[Ubuntu 7.04 _Feisty Fawn_ - Beta i386 (20070322.1) ]/ feisty main
restricted
    deb http://us.archive.ubuntu.com/ubuntu/ feisty main restricted
    #deb-src http://us.archive.ubuntu.com/ubuntu/ feisty main restricted

    ## Major bug fix updates produced after the final release of the
    ## distribution.
    deb http://us.archive.ubuntu.com/ubuntu/ feisty-updates main restricted
    #deb-src http://us.archive.ubuntu.com/ubuntu/ feisty-updates main restricted

    ## N.B. software from this repository is ENTIRELY UNSUPPORTED by the Ubuntu
    ## team, and may not be under a free licence. Please satisfy yourself as to
    ## your rights to use the software. Also, please note that software in
    ## universe WILL NOT receive any review or updates from the Ubuntu security
    ## team.
    deb http://us.archive.ubuntu.com/ubuntu/ feisty universe
    #deb-src http://us.archive.ubuntu.com/ubuntu/ feisty universe

    ## N.B. software from this repository is ENTIRELY UNSUPPORTED by the Ubuntu
    ## team, and may not be under a free licence. Please satisfy yourself as to
    ## your rights to use the software. Also, please note that software in
    ## multiverse WILL NOT receive any review or updates from the Ubuntu
    ## security team.
    deb http://us.archive.ubuntu.com/ubuntu/ feisty multiverse
    #deb-src http://us.archive.ubuntu.com/ubuntu/ feisty multiverse

    ## Uncomment the following two lines to add software from the 'backports'
    ## repository.
    ## N.B. software from this repository may not have been tested as
```

```
    ## extensively as that contained in the main release, although it includes
    ## newer versions of some applications which may provide useful features.
    ## Also, please note that software in backports WILL NOT receive any review
    ## or updates from the Ubuntu security team.
    deb http://us.archive.ubuntu.com/ubuntu/ feisty-backports main restricted
universe multiverse
    #deb-src http://us.archive.ubuntu.com/ubuntu/ feisty-backports main restrict-
ted universe multiverse

    deb http://security.ubuntu.com/ubuntu feisty-security main restricted
    #deb-src http://security.ubuntu.com/ubuntu feisty-security main restricted
    deb http://security.ubuntu.com/ubuntu feisty-security universe
    #deb-src http://security.ubuntu.com/ubuntu feisty-security universe
    deb http://security.ubuntu.com/ubuntu feisty-security multiverse
    #deb-src http://security.ubuntu.com/ubuntu feisty-security multiverse

    ## PLF REPOSITORY (Unsupported. May contain illegal packages. Use at own risk.)
    ## Medibuntu - Ubuntu 7.04 "feisty fawn"
    ## Please report any bug on https://launchpad.net/products/medibuntu/+bugs
    deb http://medibuntu.sos-sts.com/repo/ feisty free non-free
    #deb-src http://medibuntu.sos-sts.com/repo/ feisty free non-free

    ## CANONICAL COMMERCIAL REPOSITORY (Hosted on Canonical servers, not Ubuntu
    ## servers. RealPlayer10, Opera, DesktopSecure and more to come.)
    deb http://archive.canonical.com/ubuntu feisty-commercial main

    ## enlightenment e17 beta, use at your own risk
    ## E17 is in Beta and may break or break your system
    #deb http://edevelop.org/pkg-e/ubuntu feisty e17
    #deb http://e17.dunnewind.net/ubuntu feisty e17
    #deb-src http://edevelop.org/pkg-e/ubuntu feisty e17
```

保存编辑完毕的文件。

下载需要的 gpg keys

例如，使用下面的命令以得到 PLF 软件源的 gpg key：

```
wget -q http://packages.medibuntu.org/medibuntu-key.gpg -O-|sudo apt-key add -
```

如果用户希望尝试 E17 beta，也需要下载其 gpg key

```
wget -q http://lutln.ifrance.com/repo_key.asc -O- | sudo apt-key add -
```

更新包的列表：sudo aptitude update

用户也可以生成自己的 sources.list，并且可在下面的网址找到其他软件源：http://www.ubuntuLinux.nl/source-o-matic。只有在用户知道自己在做什么的前提下才能修改 Ubuntu 默认的 sources.list，用户将不同软件源混杂在一起可能使系统崩溃。

7.6 Ubuntu 软件库

Ubuntu 下能够安装成千上万个程序，这些程序被放在软件库中。

软件库分类原则基于两个因素：软件开发团队对软件的支持程度和软件遵循自由软件哲学的程度。

标准 Ubuntu 安装 CD 包括来自 Main 和 Restricted 类的一些软件。一旦系统知道这些软件库在 Internet 上的位置，就可以安装更多的软件程序。使用在系统中已经安装的软件包管理工具，用户可以直接在 Internet 上搜索、安装和更新软件的任何部分，而无需 CD。

下面是维护软件库的简单操作步骤。

Step 01 执行【系统】|【系统管理】|【软件属性】命令。

Step 02 单击【添加】按钮。

Step 03 要启用Universe软件库，请勾选 【社区维护 (Universe) 】复选框。

添加这个软件库将意味着世界上大多数的自由软件都可以安装在用户的系统中。该软件被 Ubuntu 社区自愿者中选出的小组支持，但并不被 Ubuntu核心开发团队支持，同时也许没有包括安全更新。

Step 04 要启用Multiverse软件库，请勾选【非自由 (Multiverse) 】复选框。

添加这个软件库将意味着被归为非自由类的软件可以安装在用户的系统中。该软件在某些区域可能不被允许。当从该软件库中安装每一个软件包时，用户须确保其国家法律允许使用它。需要注意的是，这类软件也许没有安全更新。

Step 05 单击Close按钮保存所做的更改然后退出。

Step 06 要应用所做的改变，单击Reload按钮。

7.7 保持系统及应用软件最新

有时，Ubuntu 开发人员会为 Ubuntu 系统中的应用程序和软件包发布新特性和安全更新。出于人性化的设计，Ubuntu 会在每次开机时自动检查系统是否有更新。通过这种方式，可以轻松保持系统软件为最新状态。

7.7.1 获悉需要更新什么

当有新的升级时，Ubuntu 将在通知区域中弹出一个红色图标。当把鼠标移动到这个红色图标上时，Ubuntu 就会通过通知区的弹出信息框来告知用户需要更新，如图 7.21 所示。

要更新系统，可以单击上图中的红色按钮，这时系统将启动系统更新管理器。打开系统更新管理器需要管理员权限，用户输入密码并单击【确定】按钮后，更新管理器窗口即会打开并列出可用的升级或补丁，单击 Install Updates 按钮 Ubuntu 会自动通过网络下载并安装这些更新。待更新管理器完成系统更新后，单击【关闭】按钮关闭弹出窗口，然后关闭更新管理器结束系统更新。当安装某些重要的更新后，有可能需要重启计算机。关于更新管理器在下一小节详细介绍。

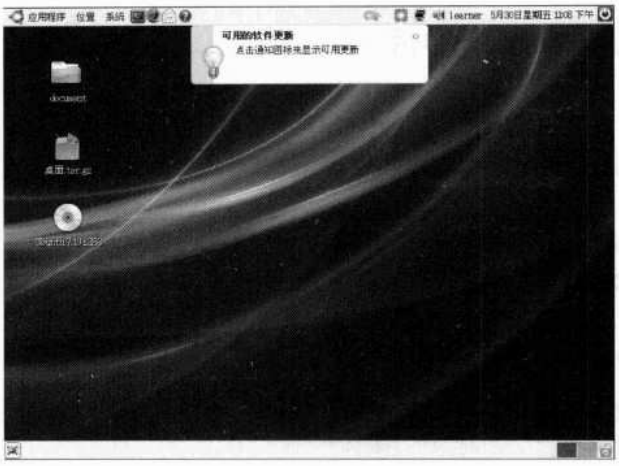


图 7.21 软件更新器

7.7.2 安装更新——更新管理器

Ubuntu 提供专门的更新管理器来实现统一的安装系统的更新。更新管理器类似于添加/删除程序、新立得软件包管理器，是 Ubuntu 系统特有的、方便易用的窗口化工具，可以通过 Ubuntu 的菜单系统找到更新管理器的入口。

打开更新管理器的方式是执行【系统】|【系统管理】|【更新程序】命令。打开后的更新管理器如图 7.22 所示。

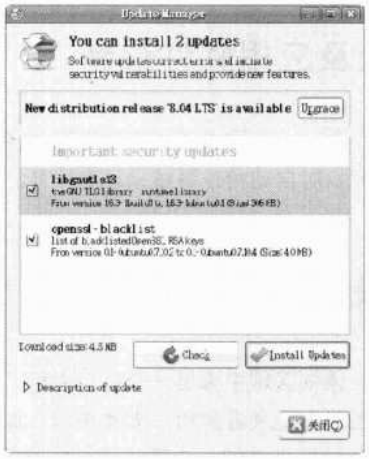


图 7.22 软件更新管理器

当然，我们还可以在 Shell 终端上通过命令行手动更新系统，更新系统的命令如下：

```
sudo aptitude update
sudo aptitude upgrade
sudo aptitude dist-upgrade
```

7.7.3 使用下一个版本的 Ubuntu

Ubuntu 系统的更新管理器不仅能实现系统中的软件包更新，而且当 Ubuntu 有大的发行版本可供升级时，还可以做系统的大版本升级。下面通过一个实例即 Ubuntu 系统由版本 7.10 通过系统升级将发行版升级到 8.04 来说明如何完成“使用下一个版本的 Ubuntu”，具体步骤如下。

Step 01 首先检测系统是否有新版版本，这个可以通过Update Manager来实现。当发现有新的发行版本时，单击【确认安装】，就进入如图7.23所示的确认窗口。

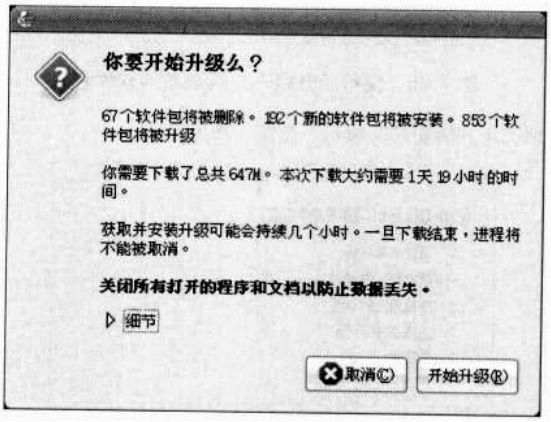


图 7.23 确定是否升级

Step 02 单击【开始升级】按钮，即可进入升级流程。发行版升级一般涉及的文件比较多，完成操作需要一定的时间。大致需要经过准备升级、设定新软件频道、获取新的软件包、安装升级、清理软件包、重新启动6个流程。第一步，准备升级，完成后进入第二步：设定新软件频道，窗口如图7.24所示。

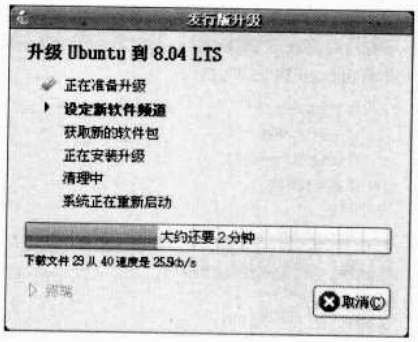


图 7.24 发行版升级——设定新软件频道

Step 03 获取新的软件包，升级程序将从Ubuntu升级站点中下载升级的软件包，这个过程会根据用户主机所处的网络的状况，消耗不少时间来下载升级软件，如图7.25所示。

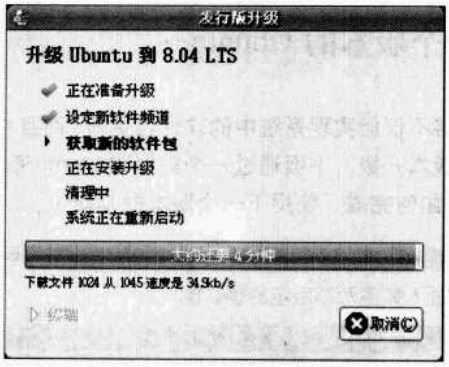


图 7.25 发行版升级——获取新的软件包

Step 04 下载升级包完毕后，将开始安装升级包，如图7.26所示。

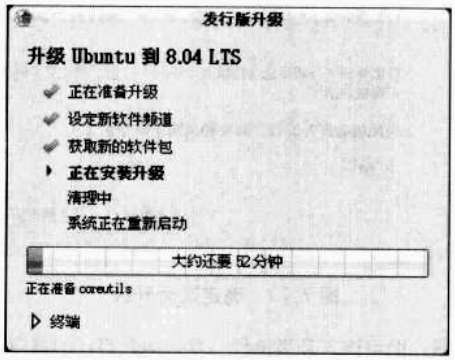


图 7.26 发行版升级——安装升级

Step 05 安装完成后，进行旧的软件包清理工作，如图7.27所示。

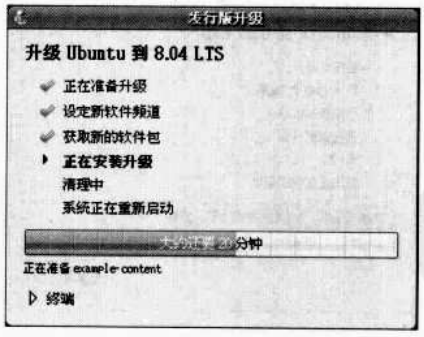


图 7.27 发行版升级——清理旧软件包

清理过程中，系统会提示“是否删除陈旧的软件包？”，单击【移除】按钮，即可完成整个升级流程，如图7.28所示。

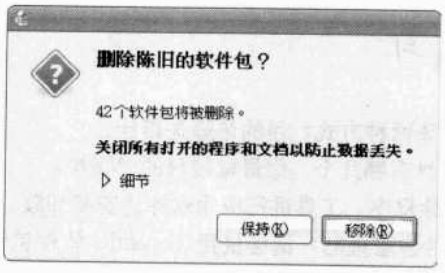


图 7.28　发行版升级——删除旧的软件包

Step 06 为了让升级后的系统即时生效，系统会提示重新启动系统，如图7.29所示，单击【现在重启】按钮，系统将进行重新启动。

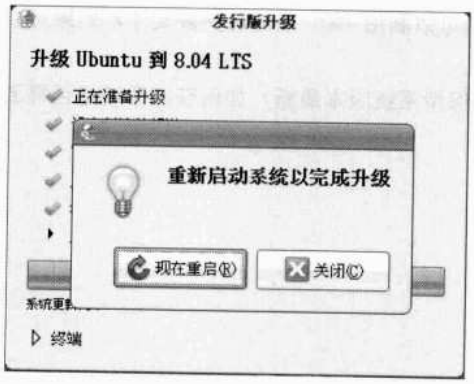


图 7.29　发行版升级——重新启动系统

重新启动后，系统将进入 Ubuntu 8.04 版本的界面，如图 7.30 所示。是不是眼前一亮？

图 7.30　更新后的桌面

7.8 课后练习

1. Ubuntu 安装包主要有哪两种方式？包的依赖是指什么？

2. Ubuntu 常用的安装工具有哪几个，你最常接触的是哪些？

3. 请尝试用“添加/删除程序”工具进行应用软件的安装卸载。

4. 什么是 Synaptic 软件包管理器？请尝试用 Synaptic 软件包管理器进行软件的安装更新及卸载。

5. 如何在 Ubuntu Shell 环境下下载安装程序 deb 包？可通过什么命令完成？

6. 在 Ubuntu 中，软件安装包主要是以什么格式存在？

7. 在 Ubuntu Shell 环境下是否可以使用源代码的 tar 包对程序进行安装？如果可以，如何操作？

8. 在 Ubuntu 中，是否可以利用 rpm 文件包安装文件？如果可以，如何操作？需要注意什么事项？

9. 在 Ubuntu 中，如何保持系统版本最新？如何获悉系统是否需要更新？若有更新，请尝试将系统更新到最新版本。

Chapter08 Shell 环境基础及设置

Shell 类似于 DOS 下的 command，它接收用户命令（如 ls 等），然后调用相应的执行程序。或许有人会说：命令行早就已经过时了。现在在 Ubuntu 中，用户也确实基本可以不懂任何 Shell，就能进行简单的应用，如看看电影，听听音乐。但是用 Linux 而不懂 Shell，就像开车只会用一挡一样，直接，简单，虽多数情况下管用却速度慢而无法真正体验驾驶的乐趣。学习 Shell 是非常必要的，通过 Shell 我们可以实现任何系统操作，从而高效管理 Ubuntu 系统。

本章的主要内容有：Ubuntu Shell 简介；Ubuntu Shell 的进入；Shell 的各种基本命令，通过它们了解 Shell 是如何工作的；当然，在 Shell 中还有能提高工作效率的应用技巧，本章也将系统地介绍一下；最后还将简单介绍 Shell 环境变量。

8.1 命令行 Shell

在学习使用 Shell 之前，我们首先了解一下什么是 Shell，Shell 的发展历程，以及当前 Shell 有哪些类型。

8.1.1 什么是 Shell

Shell 是一种具备特殊功能的程序，它是介于使用者和 Unix/Linux 操作系统之核心程序（kernel）间的一个接口。Shell 在操作系统的最外面一层，它管理用户与操作系统之间的交互，包括等待输入，向操作系统解释用户的输入，以及处理各种操作系统输出结果。可以说，Shell 提供了用户与操作系统之间通讯的方式，这种通讯可以以交互方式（从键盘输入，并且可以立即得到响应），或者以 Shell script（非交互）方式执行。

为什么说 Shell 是一种介于系统核心程序与使用者间的中介者呢？读过操作系统概论的用户都知道操作系统是一个系统资源的管理者与分配者，当有需求时，得向系统提出需求；另外从操作系统的角度来看，它也必须防止用户因为错误的操作而造成了系统的伤害。众所周知，对计算机下命令须通过命令行（command）或是程序（program）。程序由编译器（compiler）将程序转为二进制代码，可是命令呢？命令通过 Shell 完成。其实 Shell 也是一种程序，它由输入设备读取命令，再将其转为计算机可以了解的机械码，然后执行它。

各种操作系统都有它自己的 Shell，Unix/Linux 将 Shell 独立于核心程序之外，使它就如同一般的应用程序，可以在不影响操作系统本身的情况下进行修改、更新版本或是添加新的功能。

8.1.2 Shell 发展历史

要全面了解 GNU/Linux 系统 Shell，也需要了解一下 Shell 的发展历程。

Terminals

追溯到 Unix 诞生的那个年代，当时被称为计算机的机器，还是吞吐磁带与 magnetic memory（用术语 core 来表示系统 memory）的庞然大物。DEC 公司推出的 PDP-11，体积小（被称为 mini）而且价格低，在大学中引起了巨大的反响，很多学校直到那时才买得起一台计算机（PDP-11 物美价廉，只有$10 000）。这些机器的操作系统由汇编语言、机器语言写成，所以运行效率很高，但都无法移植（unportable）。每家计算机公司都给自己的机器配上独有的操作系统，然后再销售。

这种笨拙的做法很快就被人们意识到了，于是就开始兴建一个可以在不同品牌机器上运行的操作系统。1969 年，Ken Thompson 开始写后来成为 Unix 的第一行代码。其实，Dennis Ritchie 为这个新的操作系统设计了一种新的编程语言——C 语言后，事情才真正开始。虽然 Unix 的效率不及原来的操作系统，但有三个突出的优点：可以任意移植到其他机器，其中的 C 语言大大简化了编程，而且这些都免费的。很快，全美国的大学都忙着开始为机器安装 Unix 。

Unix 是可以在许多种机器上运行的操作系统，但人们又如何使用这些机器呢？他们是通过终端来连接到这些机器，也就是用键盘、显示器及足够的电子组件（electronics）组成的机器与中央计算机（central computer）相连。在这些终端上，用户可以敲字符（teletypy），这就是字符串 tty 表示终端设备文件和 getty 命令的名称来历。这些终端的厂家无法达成一项最终标准，这导致每种牌子的终端都有各自的键盘布局、各自的在屏幕上显示字符的方法、发送或接收什么信号表示什么字符、控制代码等。为了避免这些混乱，就创建了一个含有所有不同终端特性的文件（capability），这就是 termcap。大多数 Linux 终端用 vt100 或 Linux 作为终端类型。

xterms

在 80 年代初期，产生了一个 Unix 的图形子系统——the X-Window System。90 年代早期，为了更好地实现基于 Intel 的 Unix 类系统上的应用（如 FreeBSD、NetBSD、Linux），产生了一个系统分支——XFree86 。

X-Window 中一个很大的好处是可以运行多个虚拟（virtual）终端，因此在 X-Window 下就有应用程序 xterm。xterm 和 virtual terminal 在很多情况下都是一样的。有的地方说"打开一个 xterm"，其实不是非要用 xterm 程序，其他的终端模拟器（terminal emulator），如 rxvt、konsole、aterm、eterm、wterm 等等，一样有效。终端模拟器（又称为虚拟终端）通过伪（pseudo）tty 设备——pty 与系统相连，并且使用自己的显示标准——xterm 。这导致不同的终端模拟器可能在一些按键或程序上存在细小的差别，这取决于模拟器多大程度上遵守了 xterm 的显示标准。

Shell 的诞生及发展

为了在终端中运行程序，需要 Shell 。Shell 是操作系统的一部分，用来与用户打交道，并且可以用来协调各个命令。第一个真正的 Unix Shell——sh，亦称为 Bourne Shell，诞生于 1975 年，作者是 Steve Bourne。很快，出现了其他 Shell，如基于原始 Bourne Shell 的 ksh、zsh，后者常用作专属 Unix 系统中的标准 Shell。也有一些从 C 语言中衍生出来的 Shell，如 csh 或 tcsh。在 Linux 中，标准的 Shell 是 bash，即 the GNU Bourne-Again Shell。这个 Shell 功能非常强大，也可以说是过度强大了，它压缩后的 man 帮助文档就有 50 KB 。

第一个有重要意义的、标准的 Unix Shell 是 V7（AT&T 的第七版）Unix，它在 1979 年底被提出，且以它的创造者 Stephen Bourne 来命名。Bourne Shell 是以 Algol 语言为基础来设计，主要用来做自

动化系统管理工作。虽然 Bourne Shell 以简单和快速而受欢迎，但它缺少许多交互性使用的特色，例如历程、别名和工作控制。

加州大学柏克莱分校于 70 年代末期开发了 C Shell，它是作为 2BSD Unix 的部分发行。这个 Shell 主要是由 Bill Joy 写成，提供了一些在标准 Bourne Shell 所看不到的额外特色，例如命令列历程、别名和工作控制。但是 C Shell 是在大型机器上设计出来的，且增加了一些额外的功能，所以 C Shell 在小型机器上跑得较慢，即使在大型机器上跟 Bourne Shell 比起来也显得缓慢。

有了 Bourne Shell 和 C Shell 之后，Unix 用户就有了选择，且不断争论哪一个 Shell 较好。AT&T 的 David Korn 在 80 年代中期发明了 Korn Shell，在 1986 年发行且在 1988 年成为正式的部分：SVR4 Unix。Korn Shell 实际上是 Bourne Shell 的超集，且不只可在 Unix 系统上执行，同时也可在 OS/2、VMS 和 DOS 上执行。它提供了和 Bourne Shell 向上兼容的能力，且增加了许多在 C Shell 上受欢迎的特色，更增加了速度和效率。Korn Shell 已历经许多修正版，要了解使用的是哪一个版本可在 ksh 提示符号下按 Ctrl+V 键。

对于习惯 Windows 命令提示符的人，当第一次使用 Linux 命令行时，一定会感到无所适从。所熟悉的 DOS 命令 Linux 中基本不存在，摆在面前的是一大堆要记的命令。一种替代方案是利用强大的 Linux Shell 命令编写脚本，从而在 Linux 下也能用 DOS 命令。

8.1.3 Shell 的类型

在大部分 Unix 系统中，三种著名且广被支持的 Shell 是 Bourne Shell（AT&T Shell，在 Linux 下是 Bash）、C Shell（Berkeley Shell，在 Linux 下是 TCSH）和 Korn Shell（Bourne Shell 的超集）。这三种 Shell 在交互（interactive）模式下的表现相当类似，但作为命令文件语言时，在语法和执行效率上就有些不同了。

Bourne Shell

最初的 Unix Shell 是由 Stephen R. Bourne 于 20 世纪 70 年代中期在新泽西的 AT&T 贝尔实验室编写的，这就是 Bourne Shell。Bourne Shell 是一个交换式的命令解释器和命令编程语言。Bourne Shell 是标准的 Unix Shell，以前常被用来作为管理系统之用，大部分的系统管理命令文件，例如 rc start、stop 与 shutdown 都是 Bourne Shell 的执行文件，且在单一使用者模式（single user mode）下以 root 登录时它常被系统管理员使用。Bourne Shell 可以运行为 login Shell 或者 login Shell 的子 Shell（subShell）。只有 login 命令可以调用 Bourne Shell 作为一个 login Shell。此时，Shell 先读取/etc/profile 文件和$HOME/.profile 文件。/etc/profile 文件为所有的用户定制环境，$HOME/.profile 文件为本用户定制环境。最后，Shell 会等待读取的输入。Bourne Shell 提示符号的默认值是$。

C Shell

C Shell 是 20 世纪 80 年代早期柏克莱大学（Berkeley）所开发的，它主要是为了让用户更容易地使用交互式功能，并把 ALGOL 风格的语法结构变成了 C 语言风格。另外加入了一些新特性，如命令历史记录（history）、别名（alias）、内建算术、文件名完成（filename completion）和工作控制（job control）。对于常在交互模式下执行 Shell 的用户而言，他们较喜爱使用 C Shell；但对于系统管理者而言，他们则较偏好以 Bourne Shell 来做命令文件，因为 Bourne Shell 命令比 C Shell 命令

来得简单及快速。C Shell 提示符号的默认值是%。

Korn Shell

有很长一段时间，只有两类 Shell 供人们选择：Bourne Shell 用来编程，C Shell 用来交互。为了改变这种状况，AT&T 的 bell 实验室 David Korn 开发了 Korn Shell。ksh 结合了所有的 C Shell 的交互式特性，并融入了 Bourne Shell 的语法。因此，Korn Shell 广受用户的欢迎。

Korn Shell 是 Bourne Shell 的超集（superset），它新增了数学计算、进程协作（coprocess）、行内编辑（inline editing）等功能，所以也比 C Shell 更为先进。Korn Shell 的特色包括可编辑脚本、别名、函数、正规表达式万用字符（regular expression wildcard）、内建算术、工作控制（job control）、协作处理（coprocessing）和特殊的除错功能。Bourne Shell 几乎和 Korn Shell 完全向上兼容（upward compatible），所以在 Bourne Shell 下开发的程序仍能在 Korn Shell 上执行。Korn Shell 提示符号的默认值也是$。在 Linux 系统使用的 Korn Shell 叫做 pdksh，它是指 Public Domain Korn Shell。

除了执行效率稍差外，Korn Shell 在许多方面都比 Bourne Shell 微佳，但是，若将 Korn Shell 与 C Shell 相比就很困难，因为二者在许多方面都各有所长，就效率和易用性上看，Korn Shell 优于 C Shell，相信许多用户对于 C Shell 的执行效率都有负面的印象。

在 Shell 的语法方面，Korn Shell 比较接近一般程序语言，而且它具有子程序的功能及提供较多的数据类型。至于 Bourne Shell，它所拥有的数据类型是三种 Shell 中最少的，仅提供字符串变量和布尔类型。Korn Shell 是一个交互式的命令解释器和命令编程语言，它符合 POSIX，在源程序一级跨越多种平台。

在综合考虑之下 Korn Shell 是三者中表现最佳者，其次为 C Shell，最后才是 Bourne Shell，但是在实际使用中仍有其他应考虑的因素，如速度是最重要的选择时，很可能应该采用 Bourne Shell，因它是最基本的 Shell，执行速度最快。

其他 Shell

Bash（Bourne Again Shell）是 GNU 计划的一部分，用来替代 Bourne Shell。它用于基于 GNU 的系统，如 Linux。大多数的 Linux（Ubuntu，Red Hat，Slackware，Caldera）都以 Bash 作为默认的 Shell，并且运行 sh 时，其实调用的是 Bash。

POSIX Shell 是 Korn Shell 的一个变种。当前提供 POSIX Shell 的最大卖主是 Hewlett-Packard。在 HP-UX 11.0 中，POSIX Shell 就是/bin/sh，而 bsh 是/usr/old/bin/sh。

各主要操作系统下默认的 Shell：

AIX 下默认的是 Korn Shell。
Solaris 和 FreeBSD 下默认的是 Bourne Shell。
HP-UX 下默认的是 POSIX Shell。
Linux 下默认的是 Bourne Again Shell。

8.2 进入 Shell

Shell 根据交互情况，分为交互模式和非交互模式。交互式模式就是 Shell 等待输入，并且执行

提交的命令。这种模式被称作交互式是因为 Shell 与用户进行交互。这种模式也是大多数用户非常熟悉的：登录、执行一些命令、退出。Shell 也可以运行在另外一种模式——非交互式模式下。在这种模式下，Shell 不与用户进行交互，而是读取存放在文件中的命令，并且执行它们，当它读到文件的结尾时，Shell 也就终止了，主要是执行一些 Shell Script 脚本。本章将主要讲述交互式模式，下面就先介绍如何进入 Shell。

8.2.1 启动默认进入 Shell

在系统启动的时候，核心程序会被加载到内存，负责管理系统的工作，直到系统关闭为止，它建立并控制着执行程序、内存管理、文件系统、通信等。而其他的程序，包括 Shell 程序，都存放在磁盘中。核心程序将它们加载内存，执行它们，并且在它们中止后清理系统。Shell 是一个公用程序，它在用户登录时启动，用于解析用户输入的命令（命令列或命令文件），Shell 提供用户和核心程序进行交互的功能。

当用户登录（login）时，一个交互式的 Shell 会跟着启动，并提示输入命令。在输入一个命令后，接着就是 Shell 的工作了，它会进行以下操作。

- 语法分析命令列；
- 处理万用字符（wildcards）、转向（redirection）、管线（pipes）与工作控制（job control）；
- 搜寻并执行命令。

当刚开始学 Unix/Linux 系统时，大部分的时间会在提示符号（prompt）下执行命令。

如果经常会输入一组相同形式的命令，可以将这些命令放入一个文件（称为命令脚本，script），然后执行该文件。一个 Shell 命令文件类似 DOS 下的批处理文件（如 Autoexec.bat），它把一连串的 Unix 命令存入一个文件，然后执行该文件。

8.2.2 桌面终端 Shell

桌面上也提供了进入 Shell 提示（Shell prompt）的方式。Shell 提示是允许输入命令而非使用图形化界面来满足的计算需要的应用程序。虽然 Ubuntu 在图形化界面和图形化工具方面有很强大的功能，但有的时候从 Shell 提示下执行任务会更快、更有用。

通过执行【主菜单】|【应用程序】|【附件】|【终端】命令打开 Shell 提示。Shell 提示窗口如图 8.1 所示。

要退出 Shell 提示，有以下三种方式：

- 单击 Shell 提示窗口右上角的关闭按钮。
- 提示符下输入 exit。
- 按 Ctrl+D 组合键。

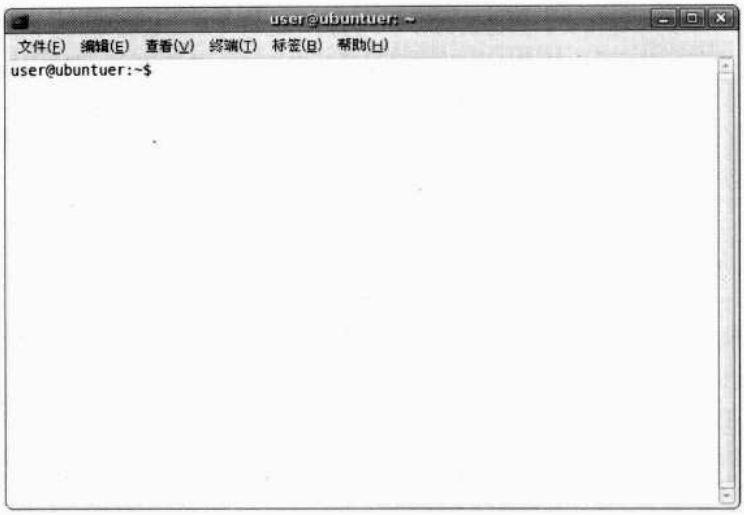


图 8.1 Shell 提示窗口

8.2.3 远程登录 Shell

如果 Linux 系统不在本地主机，而是在远程的其他计算机上，那么就必须要“远程登录”才能够在远程的主机上进行操作。假如当前主机的系统已经具有 ssh 服务或 telnet 服务，那就可以直接使用系统的 ssh 或 telnet 命令登录到远程的主机上。如果是在 Windows 下想要登录到远程的 Linux 主机，有两种方式。第一种可以打开“命令提示符”，使用 Windows 内置的 telnet 命令来登录。第二种就是使用远程登录软件 PUTTY (http://putty.nl/download.html)，或使用 PieTTY (http://ntu.csie.org/~piaip/pietty/)。PieTTY 的使用方式与 PUTTY 大同小异。如图 8.2 是远程登录的示意图。

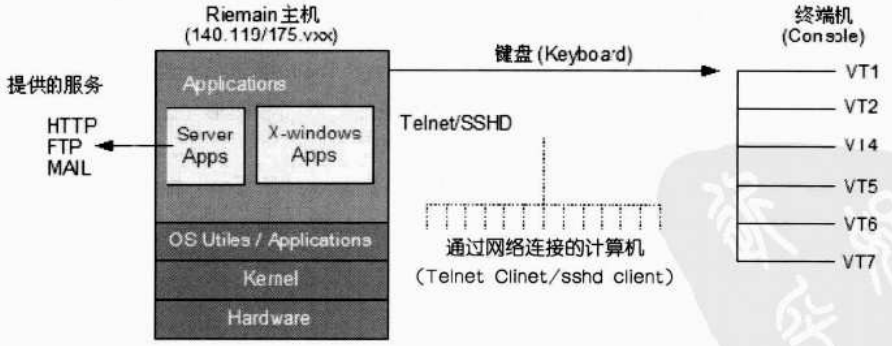


图 8.2 远程登录示意图

如果远程主机没有提供 SSH 服务，那么就需要安装和配置 SSH 服务。一般情况下，我们使用 OpenSSH 来实现远程主机的 SSH 功能，下面来介绍如何在 Ubuntu 下使用 OpenSSH 安装 ssh 服务，并对客户端进行连接配置。

OpenSSH 介绍及安装

OpenSSH 是 SSH （Secure Shell）协议的免费开源实现。它用安全、加密的网络连接工具代替了 telnet、ftp、rlogin、rsh 和 rcp 工具。OpenSSH 支持 SSH 协议的 1.3、1.5 和 2 版本。自从 OpenSSH 的 2.9 版本以来，默认的协议是 2 版本，该协议默认使用 RSA 密钥。使用 OpenSSH 工具将会增进系统的安全性。所有使用 OpenSSH 工具的通讯，包括密码，都会被加密。我们知道 telnet 和 ftp 使用纯文本密码，并被明文发送。这些信息可能在传输过程中被截取，造成密码泄露，未经授权的人员就有可能使用泄露的密码登录系统，从而对系统造成危害。所以在目前情况下，应该尽可能地使用 OpenSSH 的工具集合来避免这些安全隐患。

另外 OpenSSH 会自动把 DISPLAY 变量转发给客户机器。换句话说，如果在本地机器上运行 X 窗口系统，并且使用 ssh 命令登录到了远程机器上，当在远程机器上执行一个需要 X 的程序时，它会显示在本地机器上。如果偏爱图形化系统管理工具，却不能够总是亲身访问该服务器，这就会为工作打开方便之门。

默认情况下，Ubuntu 是不会安装 OpenSSH 的，因此用户使用 OpenSSH 之前必须先安装。在命令行终端输入：

```
sudo apt-get install openssh-server
```

执行过程如图 8.3 所示。

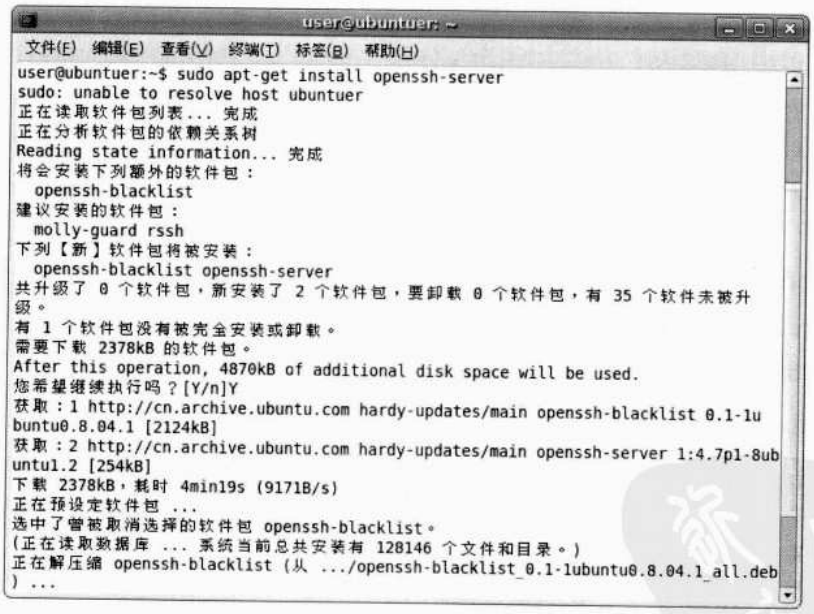


图 8.3 OpenSSH 的安装

OpenSSH 服务器端设置

OpenSSH 守护进程 sshd 使用/etc/ssh/sshd_config 配置文件。Ubuntu 安装的默认配置文件在多数情况下应该是相当完备，可以直接使用。如果想重修调整 sshd_config 配置文件中的默认值，请阅读 sshd 的说明书（man）页来获取能够在配置文件中定义的关键字列表。修改完配置文件，注意

重启 ssh 服务，以便修改能及时生效。

启动 OpenSSH 服务：

```
root@ubuntu: ~$ /etc/init.d/ssh start
* Starting OpenBSD Secure Shell Server sshd  [OK]
```

停止 OpenSSH 服务：

```
root@ubuntu: ~$ /etc/init.d/ssh stop
* Stopping OpenBSD Secure Shell Server sshd  [OK]
```

重启 OpenSSH 服务：

```
root@ubuntu: ~$ /etc/init.d/ssh restart
* Restarting OpenBSD Secure Shell Server sshd  [OK]
```

如果重新安装了 Ubuntu 系统，任何在它被重装前使用 OpenSSH 工具连接到这个系统上的客户端在它被重装后将会看到下列消息：

```
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@    WARNING: REMOTE HOST IDENTIFICATION HAS CHANGED!    @
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@

IT IS POSSIBLE THAT SOMEONE IS DOING SOMETHING NASTY!
Someone could be eavesdropping on you right now (man-in-the-middle attack) !
It is also possible that the RSA host key has just been changed.
```

重装后的系统会为自己创建一组新的身份标识密钥，因此客户会看到 RSA 主机密钥改变的警告。如果想保存系统原有的主机密钥，须备份/etc/ssh/ssh_host*key*文件，然后在系统重装后恢复它，该过程会保留系统的身份。当客户机在系统重装后试图连接它，客户端用户就不会看到以上的警告信息。

OpenSSH 客户端应用

要从客户端连接到 OpenSSH 服务器上，必须在客户机器上装有 openssh-clients 或者符合 ssh 标准的客户端软件。

使用 ssh 命令

ssh 命令是 rlogin、rsh 和 telnet 命令的安全替换。它允许在远程机器上登录并在其上执行命令。使用 ssh 来登录到远程机器和使用 telnet 相似。例如，要登录到一个 IP 地址为 192.168.1.103 的远程机器上，在 Shell 提示下输入下面的命令：

```
root@ubuntu: ~$ ssh 192.168.1.103
```

第一次使用 ssh 在远程机器上登录时，会看到和下面相仿的消息：

```
The authenticity of host '192.168.1.103' can't be established.
DSA key fingerprint is 94:68:3a:3a:bc:f3:9a:9b:01:5d:b3:07:38:e2:11:0c.
Are you sure you want to continue connecting (yes/no) ?
```

输入 yes 来继续。这会把该服务器添加到已知主机的列表中，如下面的消息所示：

```
Warning: Permanently added '192.168.1.103' (RSA) to the list of known hosts.
```

下一步，会看到询问远程主机密码的提示。在输入密码后，就会进入远程主机的 Shell 提示了。

如果没有指定用户名，在本地客户机器上登录用的用户名就会被传递给远程机器。如果想指定不同的用户名，使用下面的命令：

```
ssh username@192.168.1.103
```

或者

```
ssh -l username 192.168.1.103
```

ssh 命令可以用来在远程机器上不经 Shell 提示登录而执行命令。它的语法格式是：

```
ssh hostname command
```

譬如，如果想在远程主机 192.168.1.103 上执行 ls /usr/share/doc 命令，在 Shell 提示下输入下面的命令：

```
ssh 192.168.1.103 ls /usr/share/doc
```

在输入了正确的密码之后，/usr/share/doc 这个远程目录中的内容就会被显示，然后返回到本地 Shell 提示下。

使用 sftp 命令

sftp 工具可以用来打开一次安全互动的 FTP 会话。它与 ftp 相似，只不过它使用安全、加密的连接。它的一般语法是：

```
sftp username@hostname.com
```

然后输入密码，一旦通过验证，就可以使用一组和 FTP 相似的命令。请参阅 sftp 的说明书页（man）来获取这些命令的列表。要阅读说明书页，在 Shell 提示下执行 man sftp 命令。注意：sftp 工具只在 OpenSSH 2.5.0p1 版本以上才有。

OpenSSH 使用帮助

OpenSSH 和 OpenSSL 工程处于不断地开发中，因此关于它们的最新信息通常位于它们的官方网站中。OpenSSH 和 OpenSSL 工具的说明书（man）页也是个获取详细信息的好地方。ssh、scp、sftp、sshd 和 ssh-keygen 的说明书（man）页包括如何使用这些命令的信息，以及所有能与它们一起使用的参数。当然我们还可以参考下面的网站：

- http://www.openssh.com——OpenSSH FAQ 网页、错误报告、邮件列表、工程宗旨，以及关于安全功能的更技术性的解释。
- http://www.openssl.org——OpenSSL FAQ 网页、邮件列表，以及对于工程宗旨的描述。
- http://www.freessh.org——用于其他平台的 SSH 客户软件。

8.3 Shell 简单使用

在使用 Shell 之前，建议大家平常应用中不要用 root 账号运行 Shell，如果还是新手，这一点尤

其要注意。作为普通用户，不管有意还是无意，都无法破坏系统；但如果是root，那就不同了，只要敲几个字母，就可能导致灾难性的后果。

8.3.1 初次面对 Shell

当登入系统或打开一个 xterm 窗口，首先看到的是提示符（prompt）。Ubuntu 的标准提示符包括用户名、登入的主机名（没有设置的话是 localhost）、当前所在的目录（working directory）和提示符号。例如，以用户名 user 登入名为 ubuntuer 的主机，当前在用户的主目录——/home/user 中，提示符如下：

```
user@ubuntuer:~$
```

root 的提示符如下：

```
root@ubuntuer:~#
```

从上面两个例子看出，除了不同的用户名外，提示符号由$变成了#。根据 Bourne Shell 的传统，普通用户的提示符以$结尾，而超级用户用#。提示符的每个部分都可以定制。

要运行命令的话，只要在提示符后输入命令，然后再按回车键。Shell 将在其路径中（详情见后）搜索这个命令，找到该命令以后就运行它，并在终端输出相应的结果（如果有的话），命令结束后，再给出新的提示符，如下示例：

```
user@ubuntuer:~$ whoami
user
user@ubuntuer:~$
```

8.3.2 基本命令体验 pwd、cd、ls

作为 Ubuntu Shell 基本命令的使用入门，下面简单介绍一下最常用的命令 pwd 、cd、ls 的功能及使用方法。

使用 pwd 查看当前路径

pwd 命令代表 print working directory（打印工作目录）。当输入 pwd 时，系统便会在 Shell 提示窗口中打印当前目录名作为回应。

当系统对信息请求做出响应时，这个响应被称做“标准输出（standard output）”，它可以被打印到 Shell 提示下，也可以被重导入到其他程序或其他输出设备，如打印机。在目录中上下查看很容易迷失方向或者忘记当前目录全名。按照默认设置，Ubuntu 中的 Bash 提示只显示当前目录名，而不是整个路径。要判定当前目录在文件系统内的确切位置，那就可以使用 pwd 命令。示例如下：

```
[user@ubuntuer test]$ pwd
/var/test
```

以上例子表明，当前目录是 test，全路径是/var/test。

使用 cd 命令改变所在目录

只要知道所在的位置（的当前目录）与目标位置间的关系，要改变所在目录是很容易的，那就是使用 cd 命令。要转换到其他目录中，需要一个路径名（pathname）。可以使用绝对（absolute）路径或相对（relative）路径名。绝对路径从 /（指代根）开始，然后循序到所需的目录；相对路径从当前目录开始，当前目录可以是任何地方。

下面举个例子，假如有以下目录结构：

```
/
/directory1
/directory1/directory2
/directory1/directory2/directory3
```

如果当前是在 directory3 之下，想转换到 directory1，需要移到目录树的上一层。执行以下命令：

```
[user@ubuntuer directory3]$ cd directory1
```

当还在 directory3 目录中时，这个命令会给出一个错误消息，说明该目录不存在。这是因为在 directory3 之下并没有 directory1 目录。

要向上移到 directory1，输入以下命令：

```
[user@ubuntuer directory3]$ cd /directory1
```

这是一个绝对路径的例子。它告诉 Linux 从目录树的顶端（/）开始向下一直转换到 directory1 为止。如果一个路径的第一个字符是/，那么这个路径就是绝对路径，否则，它就是相对路径。

使用绝对路径会允许转换到从 /开始的目录中，它要求知道完整的路径。使用相对路径允许转换到相对于目前所在目录的目录中。如果要改换到当前目录下的子目录中，使用相对路径就会很方便。

命令 cd .. 告诉系统向上移到当前所在目录的直接上级目录中去。要向上移两级目录，请键入 cd ../..命令。

用下面的练习来测试一下目前所学的关于绝对路径和相对路径的知识。在用户 user 的主目录下，输入相对路径：

```
user@ubuntuer:~$ cd ../../etc/X11
```

在以上例子中，使用了全命令之后，应该是在目录 X11 中，其中会发现与 X 窗口系统相关的配置文件和目录。看一看这个 cd 命令执行的顺序：

(1) 向上移动一级，转到的用户主目录的父目录（也就是/home）中去；
(2) 再向上移动到该目录的父目录（根目录或/目录）中去；
(3) 向下移动到 etc 目录中；
(4) 向下移动到 X11 目录。

相反的，使用一个绝对路径会更快地转到 /etc/X11 目录中去。例如：

```
user@ubuntuer:~$ cd /etc/X11
```

使用 ls 来查看目录内容

已经知道如何改换目录，现在是学习如何查看这些目录内容的时候了。使用 ls 命令就可以显示当前目录的内容。ls 命令有许多可用的选项。ls 命令本身不会显示目录中的所有文件。某些文件是隐藏文件（又称“点文件”），只有在 ls 命令后指定附加的选项才能看到它们。另外，要查看 ls 命令的所有选项，可以通过在 Shell 提示下输入 man ls 来阅读其说明书页。如果想打印这个说明书页，在 Shell 提示下输入 man ls | col –b | lpr，再输入命令 ls –a，现在将会看到以“.”起始的文件，如图 8.4 所示。

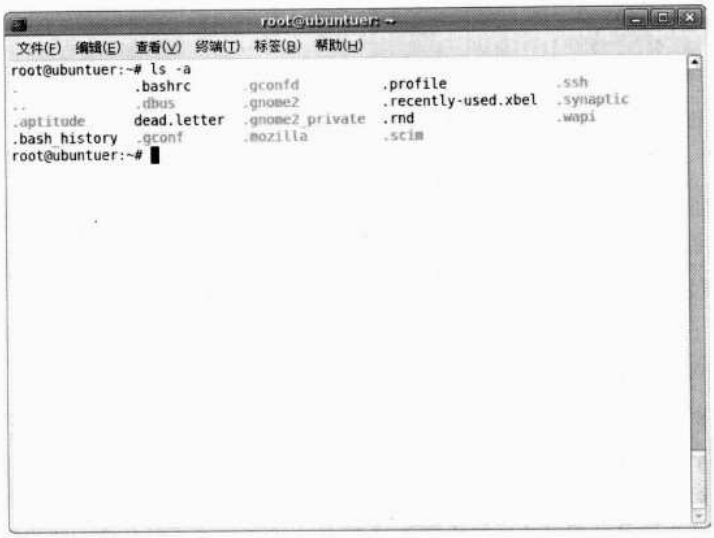


图 8.4 带有–a 选项的 ls 命令

隐藏文件多数是配置文件。它们给程序、窗口管理器、Shell 等设置首选项。它们被隐藏的目的是防止用户对其无意的篡改。当在目录中搜寻某个目录或文件时，一般不是在寻找这些配置文件，因而当在 Shell 下查看目录内容时把它们隐藏起来可以避免屏幕上的拥挤。

使用 ls –a 命令来查看所有的文件会显示大量的细节，但是通过添加更多的选项，可以看到更多的细节。

如果想查看一个文件或目录的大小、创建时间等，在 ls –a 命令后面添加 long（长）选项（–l）就可以了。这个命令显示了文件创建的日期、大小、所有者、权限等。

当想使用 ls 命令来查看目录内容时，不必位于该目录中。譬如，要在主目录中查看 /usr 目录中的内容，输入 ls –al /usr，这时效果如图 8.5 所示。

下面是一个与 ls 一起使用的一些常用选项的简短列表。当然，如果想获得更详细的 ls 帮助信息，可以通过阅读 ls 的说明书页（man ls）来获得完整的选项列表。

- –a：全部（all）。列举目录中的全部文件，包括隐藏文件（.filename）。位于这个列表起始处的.. 和.分别指父目录和当前目录。
- –l：长（long）。列举目录内容的细节，包括权限（模式）、所有者、组群、大小、创建日期、文件是否有到系统其他地方的链接，以及链接的指向。

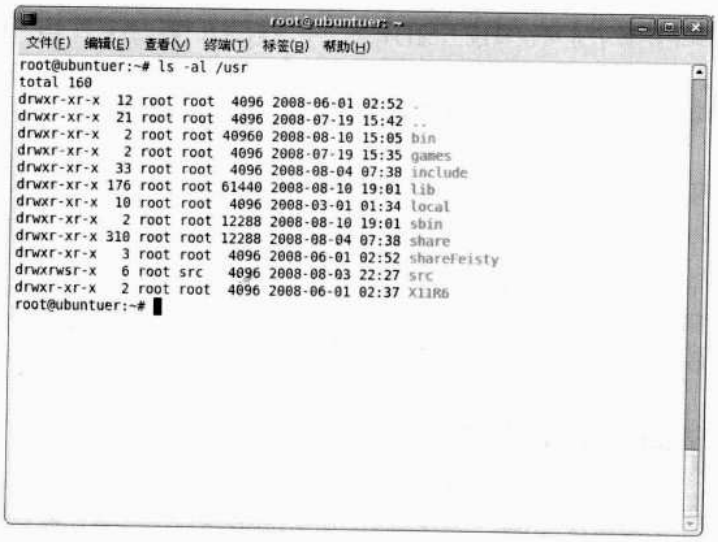


图 8.5 /usr 目录使用 ls 命令后的输出示例

- -F：文件类型（File type）。在每一个列举项目之后添加一个符号。这些符号包括：/ 表明是一个目录；@ 表明是到其他文件的符号链接；* 表明是一个可执行文件。
- -r：逆向（reverse）。从后向前列举目录中的内容。
- -R：递归（recursive）。该选项递归地列举所有目录（在当前目录之下）的内容。
- -S：大小（size）。按文件大小排序。

8.3.3 定位文件和目录 locate

有时候，知道某一文件或目录存在，但却不知该到哪里去找到它。可以使用 locate 命令来搜寻文件或目录。使用 locate 命令，将会看到每一个包括搜寻条件的目录或文件。例如，如果想搜寻所有名称中带有 finger 这个词的文件，输入以下命令：

```
[user@ubuntuer test]$ locate finger
```

locate 命令使用索引文件来定位文件或目录名中带有 finger 这个词的文件和目录。这个搜寻结果可能会包括一个叫做 finger.txt 的文件，一个叫做 pointerfinger.txt 的文件，一个被命名为 fingerthumbnails 的目录，诸如此类。只要索引文件是时时更新的，locate 命令的运行速度就会很快而且查找准确。这个索引文件在每晚都会用 cron 命令自动更新。cron 是一个在后台运行的进程，定时地执行各种任务。如果要手工地更新索引文件，可以 root 用户登录，然后输入命令 updatedb。几分钟之后，locate 命令使用的索引文件就会被更新。

8.3.4 从命令行中打印

不管是在 GUI 中单击一个按钮还是从命令行中键入命令，打印都不能算是一项很互动化的进程。

本节假定已经正确地配置了连接在系统上的打印机，并将解释如何从命令行中打印、取消打印，以及查看打印作业。

lpr 命令紧跟着一个文件名，会把指定的文件发送到打印队列中。譬如，lpr foo.txt 会打印 foo.txt 文件。

要查看在打印队列中等待的作业，在命令行中输入 lpq。输入 lpq 后，会看到和以下相似的输出。

```
active root  389  foo.txt
```

在这个例子中，389 是作业号码。

可以取消打印队列中的作业，方法是输入 lprm，再跟打印作业的号码，这个号码是在使用 lpq 命令后所显示的号码。要取消 foo.txt 打印作业，输入 lprm 389，然后按回车键。

8.3.5 清除和重设终端

在 Shell 提示下，即便只使用了一个 ls 命令，所工作的终端窗口也会开始显得拥挤。虽然可以从终端窗口中退出再打开一个新窗口，但是要清除终端中显示的内容，有一个更快、更简单的方法，即在 Shell 提示下输入 clear 命令。clear 命令会按照它字面上所暗示的来清除终端窗口。

有时，可能会无意间在一个终端窗口中打开一个程序文件或其他非文本文件。一旦关闭了那个文件，会发现输入的文本与显示器上的输出不符合。在这种情况下，输入 reset 来把终端窗口还原到它的默认状态。

8.4 Shell 应用技巧

在使用 Linux Shell 时，往往有一些执行命令的技巧，使用这些技巧，能够大大提高工作效率。本节介绍一下 Tab 自动补齐、巧用 History 等技巧。

8.4.1 Tab 自动补齐

如何用 cd 命令最快地从当前所在的 home 目录跳转到/usr/local/apache/目录下呢？看下面的例子：

```
user@ubuntuer:~$ cd /u<TAB>sr<TAB>r<TAB>
```

这称为命令行自动补齐（aulearneratic command line completion），这在平常使用 shell 命令中是不可缺少的。仔细看看下面这个例子：

```
user@ubuntuer:~$ cd /u<TAB>
```

扩展成了 cd /usr/，再接着输入：

```
user@ubuntuer:~$ cd /u<TAB>local<TAB>
```

扩展为 cd /usr/local/。如果只输入了 cd /u<TAB>s<TAB>，/usr 下匹配的（cd /u*/s*）三个子目录将列出供选择：/usr/sbin、/usr/share 和/usr/src。

因此，根据前几个字母来查找匹配的文件或子目录时使用 Tab 键会很方便。比如，ls /usr/bin/zip 将列出/usr/bin 下面所有以字符串 zip 开头的文件或子目录。当然，完成这类任务还有更强大的命令，但这个方法确实很管用。另外，碰到长文件名时 Tab 补齐就显得特别方便。假设要安装一个名为 jdk-1_5_0_12-Linux-i586-rpm.bin 的文件，输入./jdk，再使用 Tab 键，如果目录下没有其他文件能够匹配，那么 Shell 就会自动帮忙补齐。

我们都知道，Linux 是区分大小写的，在使用 Tab 时，如果不确定大小写，比如是/usr/src/Linux/Documentation 还是/usr/src/Linux/documentation。可以这样尝试性地跳转：

```
user@ubuntuer:~$ cd /u<TAB>s<TAB>l<TAB>/d
```

上面的操作如果扩展成了/usr/src/Linux/drivers/，那么就应该是 Documentation（大写的 D）。

8.4.2 命令 History

History 的中文意思就是历史，在 Shell 中是指命令的历史记录。通过按向上方向键，可以向后遍历近来在当前终端下输入的命令。用向下方向键可以向前遍历。与 Shift 键连用的话，还可以遍历以往在该控制台中的输出。也可以编辑旧的命令，然后再运行。

按 Ctrl+R 组合键后，Shell 就进入 reverse-i（ncremental）-search（向后增量搜索）模式。现在输入要找的命令的首字母：

```
(reverse-i-search) `':. //输入'test'可能会变成：
(reverse-i-search) `test': java -jar test.jar
```

如果再按回车键，上面的命令将再次执行。而如果按了向右、向左方向键或 Esc 键，上面的命令将回到普通的命令行，这样就可以进行适当编辑。

编辑命令行时，可以通过光标和功能键（Home、End 等键）浏览并编辑命令行，如果需要，还可以用键盘的快捷方式来完成一般的编辑，如下：

```
<CTRL+k>：删除从光标到行尾的部分
<CTRL+u>：删除从光标到行首的部分
<ALT+d>：删除从光标到当前单词结尾的部分
<CTRL+w>：删除从光标到当前单词开头的部分
<CTRL+a>：将光标移到行首
<CTRL+e>：将光标移到行尾
<ALT+a>：将光标移到当前单词头部
<ALT+e>：将光标移到当前单词尾部
<CTRL+y>：插入最近删除的单词
<!$>：重复前一个命令最后的参数
```

例如，用命令 mkdir peter/pan/documents/tinkerbell 新建了一个目录，现在想用命令 cd 进入该目录，可以用 cd !$，Shell 将把前一个命令 mkdir 的参数添加到现在的 cd 后面。

8.4.3 命令的别名

在使用 Shell 的过程中记住所有的命令及各自带的可选项，然后每次一一输入，这确实有点枯

燥。但幸运的是，可以为常用命令定义快捷方式。这些快捷方式可以用较简单的命令别名（alias）或复杂一些的 Shell 函数的语法来定义。

例如，用下面的命令来上传 MUO 中的文件：

```
user@ubuntuer:~$ rsync -e ssh -z -t -r -vv --progress
/home/learner/web/muo/rsmuo/docs muo:/www/mandrakeuser/docs
```

显然，如果每次都要逐一输入，那真是太痛苦了！因此方便起见，在~/.bashrc 中定义了别名：

```
alias upmuo='rsync -e ssh -z -t -r -vv --progress
/home/learner/web/muo/rsmuo/docs muo:/www/mandrakeuser/docs'
```

现在，只要输入 upmuo 就可以完成上传任务了。

定义别名的语法是：

```
alias shortcut='command'
```

命令中有空格的话，就需要用引号（如在命令与可选项间就有空格）。请注意，可以用单引号或双引号，但它们是有区别的。单引号将剥夺其中的所有字符的特殊含义，而双引号中的$（参数替换）和'（命令替换）例外。这意味着，如果想在别名中应用变量或命令的替换，就得用双引号。看一下上面的例子，作者在.bashrc 中定义了一个名为 MUOHOME 的变量：

```
export MUOHOME=$HOME/web/muo/rsmuo/docs
```

要在上面的别名中用上这个变量，就必须用双引号：

```
alias upmuo="rsync -e ssh -z -t -r -vv --progress
$MUOHOME muo:/www/mandrakeuser/docs"
```

否则，别名将查找一个名为$MUOHOME 的目录或文件。

可以用 alias 在命令行快速地创建别名，或将命令放到各自的~/.bashrc'中或放到系统级的/etc/profile.d/alias.sh 中。要删除一个别名，只要输入 unalias alias 。运行 alias 将列出系统中所有定义的别名。

如果看一下~/.bashrc 和/etc/profile.d/alias.sh，会发现系统已经定义了一些别名。可以为同一个命令定义多个别名。当然，得先确认别名与其他程序名不同，比如，像 alias rm='ls –l' 这样的别名就不能工作。可以在命令行输入这些快捷方式，测试一下。如果 Shell 找不到相同名称的命令，那就可以将其用作别名了。


注意 将有相似功能的别名以相同字母开头，比如，将所有目录的别名以 d 开头，这样有助于记忆。

8.4.4 Shell 快捷方式

Ubuntu 带有不少快捷方式，其中一部分是 Bash 原来就有的，而还有一些则是预先设置的。由于 home 目录是每位用户的活动中心，许多 Unix 版本对此有特殊的快捷方式。~就是 home 目录的简写形式。假设在其他目录，想把一个名为 sometext 的文件复制到 home 目录下的 docs 子目录中。除了输入：

```
user@ubuntuer:~$ cp sometext /home/user/docs
```

还可以用简写：

```
user@ubuntuer:~$ cp sometext ~/docs
```

理论上，这也可以应用在命令 cd 上。无论当前路径在哪里，cd ~ 将回到的用户的主目录。其实还可以进一步简化，只要输入 cd，就可以返回 home 目录了。

8.4.5 多命令执行

在 Shell 的使用中，有时候用户希望在一次执行中使用多个命令，然后在执行过程中把注意力转移到其他地方。这都是没问题的，Shell 允许在不同的命令之间放上特殊的排列字符（queuing characters），实现多命令执行。下面将介绍最常用的两种。

分号

使用方式：command1 ; command2。

先执行 command1，不管 command1 是否出错，接下来执行 command2 。

例如：

```
user@ubuntuer:~$ ls -a ; du -hs
```

将先在屏幕上列出目录中的所有内容，然后列出所有目录及其子目录所占磁盘空间大小。

&&

使用方式：command1 && command2。

只有当 command1 正确运行完毕后，才执行 command2 。

例如：

```
user@ubuntuer:~$ ls -a bogusdir && du -hs
```

将返回 ls：bogusdir：No such file or directory，而 du 则根本没有运行（这是因为没有 bogusdir 目录）。如果将符号换成了；，du 将被执行。为了进一步说明；和&&的区别，及一般命令排列的用处，下面举一个经典的例子：Linux 内核的编译和安装。要编译和安装 Linux，需要执行一串命令：make dep、make clean、make bzImage、make modules、make modules_install 和 make install。如果要等一个命令完成后，再输入下一个，再等，再输入那就太麻烦了。另一方面，每个命令只有当前面的命令都正确执行完毕后，才能开始执行。如果用；来排列命令，则即使有命令执行失败，后面的也照常运行，最后，可能在/boot 目录下得到一个有问题的内核映像（image）。而用&&输入以下命令：

```
make dep && make clean && make bzImage && make modules && make modules_
install && make install
```

不需要中途打断，就可以编译内核及其模块，并完成后面的安装。

8.4.6 命令的替换

命令替换（Command Substitution）是一项很实用的功能。假设用户想看看 XFree86 文档中的.mouse 文件，但不知道这个文件的位置。如果你已经听说了 locate 命令，也安装了 slocate 包，就可以用 locate README.mouse，发现那个文件在/usr/X11R6/lib/X11/doc 中。现在就可以在终端用 less 或在文件管理器中进入那个目录然后读取文件。而命令替换可以给带来一些便捷，如：

```
user@ubuntuer:~$ less $ (locate README.mouse)
```

一步到位。命令 locate README.mouse 的输出（= /usr/X11R6/lib/X11/doc/README.mouse）作为 less 的参数，然后就可以显示文件内容了。这种机制的语法是：

```
command1 $ (command2)
```

除了$ ()，还可以用后引号（backquote）：

```
command1 `command2`
```

这样虽然可以减少输入，但可读性差，而且很容易就和没有替换功能的一般单引号混淆。所以作者一般使用前一种方法，但这最终取决于用户的偏好。

除此之外，还有一个例子。假设打算结束一个名为 tomcatd 的程序，须先用命令 pidof 找出相应的进程号（Process ID），然后以这个 PID 为参数，运行 kill 命令，这样就可以结束 tomcatd 程序了。除了用：

```
root@ubuntuer:~$ pidof tomcatd
256
user@ubuntuer:~$ kill -9 256
```

还可以试试：

```
root@ubuntuer:~$ kill `pidof tomcatd`
```

8.4.7 命令的任务调度

当在终端运行一个命令或开启一个程序时，终端要等到命令或程序运行完毕后，才能再被使用。在 Linux 中，我们称这样的命令或程序在前台（foreground）运行。如果想在终端下运行另一个命令，则需要再打开一个新的终端。但这里还有一个更便捷的办法，即任务调度（jobbing）或后台（backgrounding）。当运用任务的调度或将命令置于后台，终端就立即解放了，这样一来，终端立即就可以接收新的输入。为实现这样的目的，只需在命令后面添加一个&：

```
user@ubuntuer:~$ gqview &
```

该命令告诉 Shell 将图片查看器 GQview 放到后台去执行（即当成 job 来运行）。

命令 jobs 将显示在这个终端窗口中运行着哪些命令与程序：

```
user@ubuntuer:~$ jobs
[1]+ Running gqview &
```

当要关闭终端窗口时，这一点就很重要，因为关闭终端将导致所有在其中运行的任务都将被中止，在此例中，如果关闭了终端，由这个终端开启的 GQview 程序也将被关闭。但如何将前台运行的一个程序放到后台去？可以使用 bg 命令，如下：

```
user@ubuntuer:~$ gqview
按Ctrl+z
[2]+ Stopped gqview
user@ubuntuer:~$ bg
[2]+ gqview &
```

组合键 Ctrl+Z 将挂起终端中正在运行的程序，然后就可以用 bg 命令将其放到后台去执行了。需要注意的是，在后台运行图形应用程序有时候是有用处的，这样可以在终端下显示这个程序的出错信息，虽然这可能没有直接的帮助，但如果碰到了麻烦，向别人询问时，这些出错提示就有用武之地了。特别是一些图形程序，很可能还处在测试期（Beta），尽管在后台执行，也会在终端输出一些信息。如果对此感觉不满意，可以用下面的命令：

```
user@ubuntuer:~$ command &>/dev/null &
```

这个操作不仅将程序送到后台执行，还将其输出发送到/dev/null 文件。而/dev/null 是系统的“碎纸机”（shredder），所有送到那里的信息都将消失殆尽。

8.5 Bash Shell 的配置文件

在使用 Shell 时，可以通过对用户的 Shell 环境进行一定的设置，以满足个性化或习惯上的要求。本节就将针对这个主题介绍一下 Bash 配置文件和 Bash 提示符。

8.5.1 Bash 配置文件

Bash 文件在用户的主目录下。在用户的主目录下运行 ls .bash*，如下：

```
user@ubuntuer:~$ ls .bash*
.bash_history .bash_logout .bash_profile .bashrc
```

- .bash_history：记录了命令历史记录；
- .bash_logout：退出 Shell 时要执行的命令；
- .bash_profile：登入 Shell 时要执行的命令；
- .bashrc：每次打开新的 Shell 时要执行的命令。

请注意后两个的区别：.bash_profile 只在会话开始时被读取一次，而.bashrc 则每次打开新的终端（如新的 xterm 窗口）时，都要被读取。按照传统，须将定义的变量，如 PATH，放到.bash_profile 中，而像 aliases（别名）和函数之类，则放在.bashrc 中。但由于.bash_profile 经常被设置成先读取.bashrc 的内容，如果图省事的话，就把所有配置都放进.bashrc。

这些文件是每一位用户的设置。系统级的设置存储在/etc/profile、/etc/bashrc 及目录/etc/profile.d 下的文件中，但最好用各自的配置文件，这样编辑不需要 root 权限，还可以使设置更有个

性，同时也不会受到别人设置的影响。当系统级与用户级的设置发生冲突时，将采用用户的设置。

8.5.2 提示符设置

在默认设置下，提示符将显示的是用户名、主机名（默认为 localhost）和当前所在目录。如下所示：

```
user@hostname ~ $
```

一般来说，最后一个字符可以标识是普通用户（$），还是 root（#）。可以通过$PS1 变量来查看提示符的设置，可以使用命令 echo $PS1：

```
user@ubuntuer:~$ echo $PS1
${debian_chroot:+ ($debian_chroot) }\u@\h:\w\$
```

但对于一些用户来说，默认设定可能有些不友好，因为提示符只显示当前目录的最后一部分。例如看到如下所示的提示符：

```
user@ubuntuer: bin $
```

从上面的提示看出，当前目录可能是/bin、/usr/bin、/usr/local/bin 及/usr/X11R6/bin。当然，可以通过 pwd 输出当前目录。能不能通过 Shell 自动提示当前目录呢？当然可以。Shell 的大部分设定，包括提示符，一般都包含在文件/etc/bashrc 中。可以通过编辑各自主目录下的.bash_profile 和.bashrc 来改变设置。在 man bash 中的 PROMPTING 部分，对这些参数（parameter）有详细说明。可以加入一些小设置，如不同格式的当前时间、命令的历史记录号，甚至不同的颜色。

在~/.bashrc 中，可以设定为：

```
PS1="\[\033[1m\][\w]\[\033[0m\] "
```

root 在~/.bashrc 中，可以设定为：

```
PS1="\[\033[0;31m\][\w]\[\033[0m\] "
```

这样得到的提示符就是：

```
[/usr/bin]
```

当用 root 时，变成：

```
[/usr/bin]
```

这里已经除掉了主机名和用户名（因为用不着这些），但如果用户想一眼就能看出其身份是普通用户还是 root。注意到，普通用户的提示符可以是黑底白字，或白底黑字。

要在终端上获得恰当的颜色调配，可以编辑下面这个脚本 color，赋予执行权限（chmod +x color），然后再运行。

```
#!/bin/bash
#
# This file echoes a bunch of color codes to the
# terminal to demonstrate what's available.  Each
# line is the color code of one forground color,
```

```
# out of 17 (default + 16 escapes) , followed by a
# test use of that color on all nine background
# colors (default + 8 escapes) .
#
T='gYw' # The test text
echo -e "\n    40m   41m   42m   43m\
   44m   45m   46m   47m";
for FGs in '  m' ' 1m' '  30m' '1;30m' '  31m' '1;31m' '  32m' \
  '1;32m' '  33m' '1;33m' '  34m' '1;34m' '  35m' '1;35m' \
  '  36m' '1;36m' '  37m' '1;37m';
  do FG=${FGs// /}
  echo -en " $FGs \033[$FG  $T  "
  for BG in 40m 41m 42m 43m 44m 45m 46m 47m;
  do echo -en "$EINS \033[$FG\033[$BG  $T  \033[0m";
  done
  echo;
done
echo
```

一种更适当的设定如下：

```
PS1="\u: \w\\$ "
```

这样，提示符就变成：

```
user: /usr/bin$
```

可以通过命令 export 来测试不同的设置（比如，export PS1="\u: \w\\$ "）。如果找到了适合的提示符，就将设置放到.bashrc 中。这样，每次打开控制台或终端窗口时，都会生效。

甚至可以给提示符设定主题（theme），也就是搭配不同的颜色。

8.6 Shell 环境命令

$PATH 与**$PS1** 一样，也是环境变量。输入 set 将列出所有当前定义的环境变量。看到的这些环境变量在 Shell 的配置文件中定义，可能是用户自己的配置文件，也可能是由 root 通过/etc 下面的系统级文件定义的。如果使用 X，更多的一些变量将由 X 的窗口管理器或桌面环境的启动文件配置。

如果对这些设置不很清楚，最好暂时不要随便改动。了解如何改变 $PATH 变量很有用，因为这个变量决定了 Shell 将到哪些目录中寻找命令或程序。如果要执行的命令的目录在$PATH 中，就不必输入这个命令的完整路径，直接输入命令就可以了。一些第三方软件没有将可执行文件放到 Linux 的标准目录中。因此，将这些非标准的安装目录添加到 $PATH 是一种解决办法。此外，也将看到如何处理一般的环境变量。

首先，作为惯例，所有环境变量名都是大写的。由于 Linux 区分大小写，这点要留意。当然，可以自己定义一些变量，如$path、$pAtH，但 Shell 不会理睬这些变量。

要注意的第二点是变量名有时以$开头，但有时又不是。当设置一个变量时，直接用名称，而不需要加$：

```
PATH=/usr/bin:/bin:/usr/local/bin:/usr/X11R6/bin
```

要获取变量值的话，就要在变量名前加$：

```
user@ubuntuer:~$ echo $PATH
/usr/bin:/bin:/usr/local/bin:/usr/X11R6/bin
```

否则的话，变量名就会被当作普通文本了：

```
echo PATH
PATH
```

处理 $PATH 变量要注意的第三点是：不能只替换变量，而要将新的字符串添加到原来的值中。在大多数情况下，不能用 PATH=/some/directory，因为这将删除 $PATH 中其他的所有目录，这样在该终端运行程序时，就不得不给出完整路径。所以，只能添加以下语句：

```
PATH=$PATH:/some/directory
```

这样，PATH 被设成当前的值（以$PATH 来表示）＋新添的目录。

到目前为止，只为当前终端设置了新的$PATH 变量。如果打开一个新的终端，运行 echo $PATH，将返回旧的$PATH 值，而看不到刚才添加的新目录，这是因为先前定义的是一个局部环境变量（仅限于当前的终端）。要定义一个全局变量，使其在以后打开的终端中生效，需要将局部变量输出（export），可以用 export 命令：

```
export PATH=$PATH:/some/directory
```

现在如果打开一个新的终端，输入 echo $PATH，也能看到新设置的$PATH 了。请注意，命令 export 只能改变当前终端及以后运行的终端的变量，对于已经运行的终端没有作用。为了将目录永久添加到$PATH，只要将 export 的那行添加到.bash_profile 文件中。请不要在.bashrc 中设置 PATH，否则会导致 PATH 中目录的意外增长。每次打开一个新的 Shell，.bashrc 都会作用。所以如果在该文件中添加目录，每次打开一个终端，目录又会被添加。这将导致 PATH 变量由于目录复制，不断地增长。下面将介绍与 path 有关的一些命令。

8.6.1 echo 指令

echo 用于显示变量内容：

语法：

```
user@ubuntuer:~$ echo $variable
```

参数说明示例：

```
user@ubuntuer:~$ echo $PATH
/bin:/sbin:/usr/sbin:/usr/bin:/usr/local/sbin:/usr/local/bin:/usr/loc
al/bin:/bin:/usr/bin:/usr/X11R6/bin
```

就如同上面的示例，当要显示目前的 PATH 这个变量时，使用 echo，而为了要分辨是否为变量，那么 Linux 系统预设变量名称前面需要加上一个“$”符号。

8.6.2 env 指令

显示目前系统中主要的默认变量内容：

```
user@ubuntuer:~$ env
ENV=/home/user/.bashrc          <==当前用户环境变量的配置文件
HISTSIZE=1000                   <==命令历史记录数量
HOME=/home/user                 <==登录用户的主目录
HOSTNAME=ubuntuer               <==这部主机的主机名称
LANGUAGE=C                      <==默认语系
LANG=zh_CN.UTF-8
```

与 LANGUAGE 类似，这个则是各个 Linux distribution 常用的默认语言变量，由于系统安装的时候，本书选择了中文系统，所以默认语言是中文，即 zh_CN.UTF-8，如果要修改这个变量，可以到 /etc/sysconfig/i18n 去修改。下面的 LC_xxx 均是与默认的表示语言有关的变量。

```
LESSOPEN=|/usr/bin/lesspipe.sh %s  <==用来设定 less 使用的一个脚本文件
LOGNAME=user                       <==登录用户的账号
MAIL=/var/mail/user                <==登录用户的邮件放置地点
PATH=/usr/local/bin:/usr/local/sbin:/usr/bin:/usr/sbin:/bin:/sbin:/home/use
r/bin
PWD=/home/user                     <==目前用户所在的目录（当下的目录）
SHELL=/bin/bash                    <==登录用户使用的 shell 类型
USER=user                          <==当前用户的名称
```

env 是 environment 的简写，所以说，这个命令主要用于将目前系统中的主要变量读出来。除了 env 这个读取环境变量的命令之外，还有一个可以将目前系统中所有的变量数据都读出来的命令——set。set 除了会将上面的数据都给读出来之外，还会有额外的信息也一起读出来，而且这些信息通常都与当前用户的设置有关。

8.6.3 set 指令

Set 指令用于显示目前系统中全部的变量内容。

语法：

```
user@ubuntuer:~$ set
BASH=/bin/bash                          <==Bash 的主程序放置路径
BASH_VERSINFO=3                         <==Bash 版本信息
BASH_VERSION= 3.2.39 (1) -release       <==BASH 的版本
COLUMNS=132                             <==目前这个终端机使用的字段有几个字符距离
HISTFILE= /home/user/.bash_history
                              <==目前用来存放命令历史记录的文件，为一个隐藏文件
HISTFILESIZE=500              <==命令历史记录的文件，命令的最大数（只记录 500 个指令）
LINES=66                      <==目前光标所在的位置为第几行
MAILCHECK=60                  <==每隔多久检查一次有无新信件（秒数）
PPID=1                        <==目前 Bash 这个父程序的 ID！
SHELLOPTS= braceexpand:emacs:hashall:histexpand:interactive-comments:monitor
```

```
TERM=xterm                    <==终端形式
UID=1003                      <==用户 ID (UID)
.....
```

Set 命令的输入就是直接输入 set 即可。使用 set 除了会将系统的默认变量列出来之外，连带的所有的用户设定的变量也会被列出来。同时需要注意的是，若当时有相当多人同时在线的话，那么你的变量只能给自己使用（除非改的是系统的配置文件，如/etc/profile），而不会干扰到别人。就如同前面所说的，由于用户登录到 Ubuntu Linux 之后会取得一个 PID，而用户的设定将只对这个 PID 与子程序起作用。此外，这次登录所进行的变量设定，如果没有更改到配置文件，那么这次设定的变量在下次登录时将被取消掉。所以，如果用户想要变量每次都能在其登录的时候自动配置好，那么就必须将其配置写入登录时加载的配置文件。

8.6.4 变量设定规则

现在读者应该已经知道一些系统的默认变量了，但是如果自己想要设定一些自己的变量，该如何设定呢？下面列举一下在 Bash 下的变量设定规则，以便进行管理。

- 变量与变量之间以等号=来连接。
- 等号两边不能直接接空格符。
- 变量名称只能是英文字母与数字，但是数字不能作为开头字符。
- 若有空格符，可以使用双引号"或单引号'来将变量内容结合起来，但需要特别留意，双引号内的特殊字符可以保有变量特性，但是单引号内的特殊字符则仅为一般字符。
- 必要时需要以字符 \来将特殊符号（如 Enter，$，\，空格符，'等）变成一般符号。
- 有些命令还需要其他的指令提供信息，可以使用 quote 'command'。
- 若该变量为扩增变量内容时，则须以双引号及$连接变量名称（如："$PATH":/home）来继续添加内容。
- 若该变量需要在其他子程序中起作用，则需要以 export 来使变量生效，如 export PATH。
- 通常，大写字母的变量为系统预设变量，自定义变量可以使用小写字母，以方便判断。
- 取消变量的方法为：unset 变量名称。

8.6.5 export 指令

当用户取得一个 Bash 之后，亦即得到一个程序了，但是若再次执行一个 Bash，那么将进入子程序。由于已经进入该子程序，所有父程序中的变量设定将不再起作用。如要让该变量继续在子程序中起作用，那么就需要执行：

```
user@ubuntuer:~$ export $variable
```

这个命令用在引用他人的设定或者其他程序时，相当重要。尤其是需要两三个文件互相引用时，如果忘记设定 export，那么不同文件中的相同变量名需要一再地重复设定才行。为了方便，只要在第一个文件使用 export，那么后续文件引用时，将会把该变量内容读进来。如果输入 export 而没有接变量时，那么此时将会把所有的环境变量导出来。也就是说，export 可以将一般自定义变量变成

环境变量。

```
user@ubuntuer:~$ export
declare -x HISTCONTROL="ignoreboth"
declare -x HOME="/home/user"
declare -x LANG="zh_CN.UTF-8"
declare -x LANGUAGE="zh_CN:zh:en_US:en"
declare -x LESSCLOSE="/usr/bin/lesspipe %s %s"
declare -x LESSOPEN="| /usr/bin/lesspipe %s"
declare -x LOGNAME="user"
declare -x LS_COLORS="no=00:fi=00:di=01;34:ln=01;36:pi=40;33:so=01;35:do=01;
35:bd=40;33;01:cd=40;33;01:or=40;31;01:su=37;41:sg=30;43:tw=30;42:ow=34;42:
st=37;44:ex=01;32:*.tar=01;31:*.tgz=01;
31:*.svgz=01;31:*.arj=01;31:*.taz=01;
31:*.lzh=01;31:*.lzma=01;31:*.zip=01;
31:*.z=01;31:*.Z=01;31:*.dz=01;31:*.gz=01;31:*.bz2=01;31:*.bz=01;
31:*.tbz2=01;
31:*.tz=01;31:*.deb=01;31:*.rpm=01;31:*.
jar=01;31:*.rar=01;31:*.ace=01;31:*.zoo=01;
31:*.cpio=01;31:*.7z=01;31:*.rz=01;
31:*.jpg=01;35:*.jpeg=01;35:*.gif=01;35:*.bmp=01;
35:*.pbm=01;35:*.pgm=01;35:*.
ppm=01;35:*.tga=01;35:*.xbm=01;35:*.xpm=01;
35:*.tif=01;35:*.tiff=01;35:*.png=01;
35:*.svg=01;35:*.mng=01;35:*.pcx=01;35:*.mov=01;
35:*.mpg=01;35:*.mpeg=01;35:*.
m2v=01;35:*.mkv=01;35:*.ogm=01;35:*.mp4=01;35:*.m4v=01;
35:*.mp4v=01;35:*.vob=
01;35:*.qt=01;35:*.nuv=01;35:*.wmv=01;35:*.asf=01;
35:*.rm=01;35:*.rmvb=01;35:*
.flc=01;35:*.avi=01;35:*.fli=01;35:*.gl=01;
35:*.dl=01;35:*.xcf=01;35:*.xwd=01;
35:*.yuv=01;35:*.aac=00;36:*.au=00;36:*.flac=00;
36:*.mid=00;36:*.midi=00;36:*.
mka=00;36:*.mp3=00;36:*.mpc=00;36:*.ogg=00;
36:*.ra=00;36:*.wav=00;36:"
declare -x MAIL="/var/mail/user"
declare -x OLDPWD="/usr/bin/X11"
declare  -x  PATH="/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/
bin:/usr/games"
declare -x PWD="/home/user"
declare -x SHELL="/bin/bash"
declare -x SHLVL="1"
declare -x SSH_AUTH_SOCK="/tmp/ssh-eKpKx19073/agent.19073"
declare -x SSH_CLIENT="192.168.1.100 18791 22"
declare -x SSH_CONNECTION="192.168.1.100 18791 192.168.1.103 22"
declare -x SSH_TTY="/dev/pts/2"
```

```
declare -x TERM="vt100"
declare -x USER="user"
declare -x XDG_SESSION_COOKIE="23e4b0c1107b4c28f1131f0047c84be1-1218387096.
283110-1423600647"
declare -x lang="C"
```

8.6.6 unset 指令

unset 指令即直接将该变量的内容去掉。

语法：

```
user@ubuntuer:~$ unset $variable
```

8.7 课后练习

1. 什么是 Shell？Shell 的发展与 Linux 的发展有什么关系？
2. 当前，Shell 包括哪些类型？请列举一些类型，并说明其中的区别。
3. 如何在 Ubuntu 环境中使用 Shell 终端？
4. 请列举一些常用的 Shell 命令，体验一下 cd、pwd、ls 等简单 Shell 命令。
5. 请尝试使用本节中提到的一些使用技巧，如 Tab 键自动补齐，通过向上键获取以前操作的指令。
6. 在 Ubuntu Shell 使用中，是否支持多命令执行？如果支持，多个命令如何分隔？
7. Ubuntu Shell 中常用的通配符有哪些？它们的功能是什么？
8. Linux Shell 中 Bash 配置文件包括哪些内容项？
9. Linux Shell 中环境变量的设置包括哪些指令？请简单说明其功用。

Chapter09 Vi/Vim 编辑器

Linux 平台有多种编辑器，如 Vi，emacs，xemacs，joe，e3，xedit，kedit，pico 等，其中 Vi 是几乎所有 Unix/Linux 系统默认安装的组件。Vi 有着非常强大的编辑功能，几乎可以实现所有的文字编辑功能，并且 GNU 推出了 Vi 进阶版本 Vim，可用额外功能就更加强大，如颜色显示等，极大地方便用户撰写文档。本章中，我们将重点讲述如下内容：Vi/Vim 的简单介绍、Vi/Vim 的工作模式、Vi 在各种模式下的使用、Vi 的高级编辑命令及 Vi 的系统配置等。

9.1 Vi/Vim 简介

Vi 作为 Unix 及 Linux 等类 Unix 系统下的标准编辑器，其功能非常强大，它的强大不逊色于任何最新的文本编辑器，包括 Windows 平台下的编辑器。下面就让我们追本溯源，了解一下 Vi 及其进阶版本 Vim 的来源、发展等基本情况。

9.1.1 Vi 概述

Vi（Visual edit）原意是 Visual，它是一个立即反应的编辑软件，也就是说可以立刻看到操作结果。Vi 是大多数 Unix 操作系统都支持的全屏文本编辑器，可以说，几乎任何一台 Unix 机器都会提供这套软体。Linux 当然也一样，其实 Linux 的 Vi 是 elvis（主要是因为版权问题），不过它们几乎一样，都具有字处理程序的灵活性和简单易用的特性。

Unix 提供一系列 ex 编辑器，包括 ex、edit 和 Vi。相对于全屏幕编辑器，现在可能很难想象如何使用 ex、edit 这种行编辑器（类似于 DOS 3.3 版以前所附的 EDLIN）。也由于 Vi 是全屏幕编辑器，所以它必须控制整个终端机屏幕哪里该显示些什么，而终端机的种类有许多种，特性又不尽相同，所以 Vi 有必要知道现在所使用的是哪一种终端机。这是根据 TERM 这个环境变量来设定，设定环境变量方面的介绍请查看第 8 章的内容。

由于 Vi 是从行编辑器 ex 发展而来的，因此就有可能在 Vi 中使用 ex 的命令。使用 Vi 时，用户对文件的修改会立刻在屏幕上显示出来，屏幕上光标的位置指明了用户在文件中的位置。Vi 有 100 多种命令，但简单的编辑工作只需要少数几个命令。

9.1.2 Vi 的进阶——Vim

Vim（Vi Improved）是 Bram Moolenaar 开发的与 Unix 下的通用文本编辑器 Vi 兼容并且更加强大的文本编辑器。Vi 可以说是老式的文书处理器，虽然功能已经很齐全了，但是还是有可以改进的地方。Vim 兼容了 Vi 所有功能，加入了很多高级编辑功能，如语法变色、正规表达式匹配搜寻与替换、插入补全、自定义键，还有区块复制、多文件编辑等功能，为编辑文本提供了极大方便。

另外，现在 Vim 里面加入了很多对于程序编辑的功能，例如支持许多程序语法（syntax），当使用 Vim 编辑程序时（不论是 C 语言，还是 shell 脚本），Vim 可直接进行“程序调试（debug）”。Vim 的这些强大的功能，对于编写程序来说也是一个很好用的工具，就连 Vim 的官方网站（http://www.Vim.org）自己也说 Vim 是一个“程序开发工具”而不是文书处理软件。

Vim 可以运行在任何操作系统上，包括我们常用的 Windows 和 Unix/Linux。一旦掌握了 Vim，就掌握了一项跨平台的利器。可以说，Vim 是一个历久弥新的编辑器。

9.1.3 Vi 和 Vim 的差异

那么 Vi 和 Vim 到底有多大差异呢？Vi 和 Vim 都是多模式编辑器，不同的是 Vim 是 Vi 的升级版本，它不仅兼容 Vi 的所有指令，而且还有一些新的特性。Vim 的这些优势主要体现在以下几个方面。

对 Vi 的完全兼容

某些情况下，可以把 Vim 当成 Vi 来使用。

多级撤销

我们知道，在 Vi 里，按 U 键只能撤销上次命令，而在 Vim 中可以无限制地撤销。

易用性

Vi 只能运行于 Unix 中，而 Vim 不仅可以运行于 Unix，还可以运行于 Windows、mac 等多操作平台。

语法加亮

Vim 可以用不同的颜色来高亮显示代码，可以用:sy on 来开启颜色。

可视化操作

Vim 不仅可以在终端运行，也可以运行于 X-Window、mac os、Windows。

正是因为 Vim 的上述特点，当前大部分的 Linux 版本都已采用 Vim 替代 Vi，包括 Ubuntu。如果使用 Vi 后，却看到画面右下角有显示当前光标所在的行号，那么就说明 Vi 已经被 Vim 所替换。如图 9.1 所示的文件是由 Vi 打开的，而如图 9.2 所示的文件是由 Vim 打开。

基于大部分 Linux 发行版本用 Vim 替代原始的 Vi，本书后面的章节中也将不特别的区别对待 Vi 和 Vim，大家学习时也不用去特别钻研 Vi 和 Vim 的区别。下节将介绍如何把 Vim 别名到 Vi。

图 9.1 Vi 打开文件的界面

图 9.2 Vim 打开文件的界面

9.1.4 Vim 别名到 Vi

在 Ubuntu Shell 环境下，输入 alias 时，应该出现如下画面：

```
user@ubuntuer:~$ alias
alias Vi='Vim'
```

这表示使用 Vi 这个命令时，其实就是执行 Vim。也就是说，vim 别名到 vi 了。如果没有这一行，那么就必须使用 Vim 文件名来启动 Vim。当然，也可以手动将 vim 别名到 vi，其操作方法是：编辑用户目录下的.bashrc 文件，添加 alias vi='vim'即可。具体实现如下所示：

```
user@ubuntuer:~$ vi ~/.bashrc

# .bashrc

# User specific aliases and functions
```

```
alias rm='rm -i'
alias cp='cp -i'
alias mv='mv -i'
alias vi='vim'

# Source global definitions
if [ -f /etc/bashrc ]; then
 . /etc/bashrc
fi
~
~
```

下面来看看输入 Vi 命令（其实是运行 Vim）的画面是什么样的。假设要编辑 /etc/man.config，则输入“Vi /etc/man.config”。

```
#
# Generated automatically from man.conf.in by the
# configure script.
#
# man.conf from man-1.5p
#
# For more information about this file, see the man pages man (1)
# and man.conf (5) .
"man.config" 138L, 4506C     1, 1 Top
```

可以看到，上面的编辑界面有以下几个特点：

- 行说明这个文件的特点，包括 138 行，共 4506 个字符。
- 1，1 表示当前光标在第一行的第一个字符上。可以看到第一行有个光标。
- Top 表示这个画面是整个文件的最上方。

至少应该有以上这些信息。在移动光标时，1，1 的光标定位也会跟着变动，这说明界面是 Vim 画面的一部分。

通过别名映射，可以把 Vim 当作 Vi 使用。在后面的叙述中，本书也统一只提 Vi，不再特别强调是 Vi 功能还是 Vim 功能，请大家注意。

9.2 Vi 使用入门

熟悉了 Vi 的背景后，我们开始学习如何使用 Vi 吧。Vi 和 Windows 下 notepad 视窗编辑器不一样，它要完全工作，需要涉及三个工作模式。工欲善其事，必先利其器。我们先在本节中了解一下 Vi 的工作模式，然后通过一个例子体验一下 Vi 的进入、退出等简单使用。

9.2.1 Vi 的工作模式

前面我们知道 Vi/Vim 是全屏编辑器，正因为这种特性，通常 Vi 共分为三种模式，分别是一般

模式、编辑模式与命令行命令模式。这三种模式的作用如下。

一般模式 Normal Mode（common Mode，c-Mode）

Vi 处理文件时，一进入该文件，就是一般模式了。在这个模式中输入的任何字符皆被视为指令。可以使用方向键来移动光标，可以使用“删除字符”或“删除整行”来处理文件内容，也可以使用“复制”、“粘贴”来处理文件数据。

文本写入模式 Insert Mode（i-Mode）

在一般模式中可以进行删除、复制、粘贴等操作，却无法进行编辑操作。要等到按下 i、I、o、O、a、A、r、R 等字母之后才会进入编辑模式。在文本写入模式下，键盘成了用户的打字机。Vi 显示用户的输入。按键不被解释为命令执行，只是作为文本写入到用户的文件中。注意，通常在 Linux 中，按下上述字母时，画面的左下方会出现“INSERT 或 REPLACE”的字样，才可以输入任何字符到文件中。如果要回到一般模式时，则必须要按下 Esc 键才可退出编辑模式。

命令行命令模式 Ed Mode（common-line Mode，e-Mode）

在一般模式中，输入：或/或?就可以将光标移到最下面的一行，在这个模式中，可以搜索数据，而且读取、存盘、大量删除字符、离开 Vi、显示行号等操作都是在此模式中实现的。简而言之，可以将这三种模式用图 9.3 来表示。

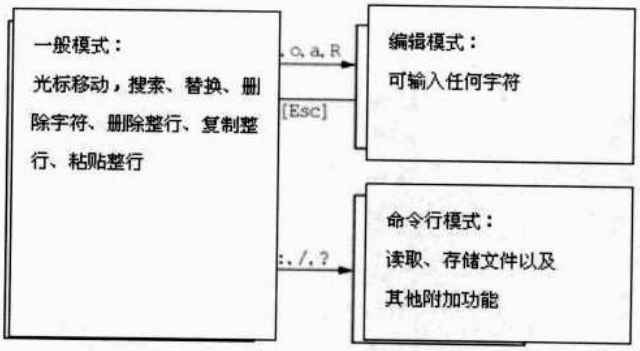


图 9.3 Vi 的三种模式

9.2.2 使用范例

使用 Vi 进入一般模式

```
user@ubuntuer:~$ Vi test.txt
```

直接输入“Vi 文件名”即可进入 Vi。如图 9.4 所示，左下角会显示这个文件的当前状态。如果是新建文件，会显示 New File，如果是已存在的文件，则会显示当前文件名、行数与字符数，例如，"/etc/man.config" 145L，4614C。

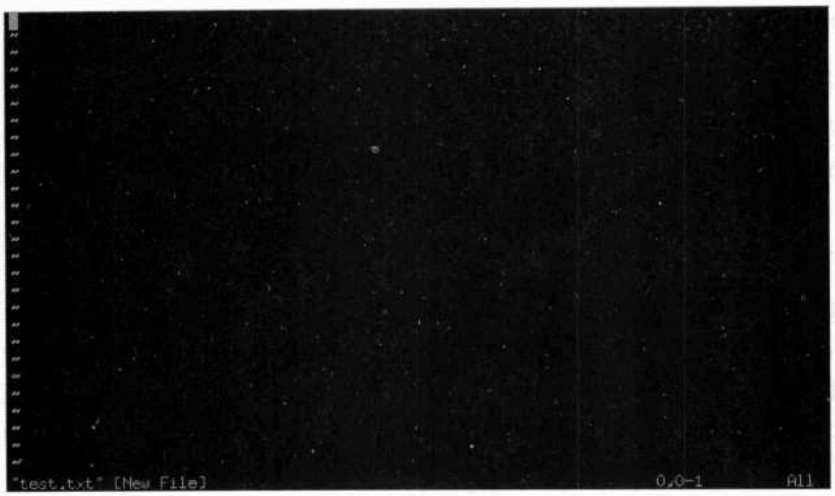


图 9.4 利用 Vi 打开一个文件

按 i 进入编辑模式，开始编辑文字

在一般模式中，只要按 I、o、a 等字符，就可以进入编辑模式了。在编辑模式中，可以发现在左下角会出现—INSERT—，意味着可以输入任意字符，如图 9.5 所示。这个时候，键盘上除了 Esc 键之外，其他键都可以视作为一般的输入键，可以进行任何编辑（在 Vi 中，Tab 键所得到的结果与空格符所得到的不一样，这里特别强调一下）。

图 9.5 进入 Vi 的编辑模式

按 Esc 键回到一般模式

假设已经按照上面的样式编辑完毕，那么，应该如何退出？就是按 Esc 键。马上就会发现画面左下角的—INSERT—不见了。

在一般模式中输入:wq 保存后退出 Vi

保存并退出的命令很简单，输入:wq 即可保存文件并退出（注意，输入：，该光标就会移到最下面那一行）。这时在提示符后面输入 ls -l 即可看到刚建立的 test.txt 文件，最后结果如图 9.6 所示。

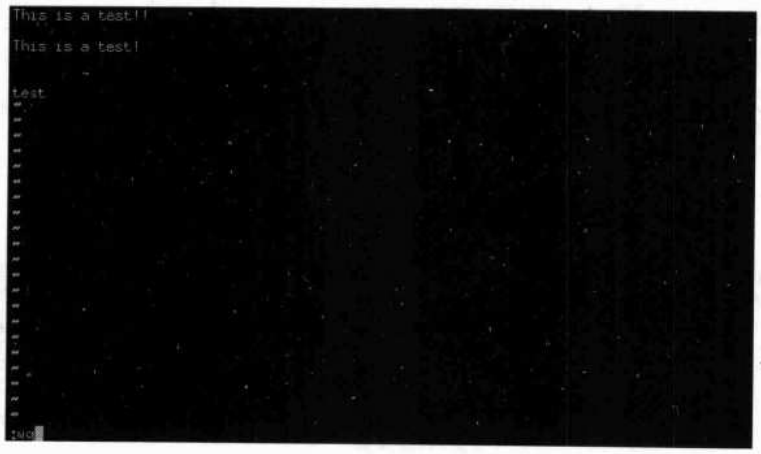


图 9.6 利用 Vi 存储文件

如此一来，文件 test.txt 已经建立好了，很简单。需要注意的是，如果文件权限不对，例如为 -r--r--r-- 时，那么可能会无法写入。可以使用“强制写入”的方式吗？可以。使用:wq!，多加一个感叹号即可。不过，需要特别注意的是，在“权限可以改变”的情况下才能进行这样的操作。

9.3 Vi 的基本操作指令

其实在上面一节的简单应用中，我们就用到了好几个简单操作指令了。可以说，Vi 编辑文档就是通过各种指令和编辑模式下的写入操作完成的。掌握并会熟练应用各个指令，也就掌握了 Vi。下面根据 Vi 的模式，先系统地介绍一下 Vi 的各个基本操作指令。

9.3.1 Normal Mode 下操作命令

Vi 打开文档的默认模式就是 Normal Mode，在这个模式下，可以进行任何的光标移动以定位文本，可以进行删除、复制、粘贴、查找等编辑功能，还能通过指令对前面的操作进行重复或取消。下面分别对上面的这些功能对应的指令进行介绍。

光标移动

在 Vi 下，鼠标几乎不可用，这时光标移动指令就尤为重要了，只有通过它我们方能定位到要写入或编辑的位置，表 9.1 是一般模式下移动光标的方法。

表 9.1 一般模式：移动光标的方法（1）

按键	说明
h 或向左方向键（←）	光标向左移动一个字符
j 或向下方向键（↓）	光标向下移动一个字符
k 或向上方向键（↑）	光标向上移动一个字符
l 或向右方向键（→）	光标向右移动一个字符

如果想要进行多次移动的话，例如向下移动 30 行，可以使用“30j”或“30↓”的组合键，即加上想要进行的次数（数字）操作即可。

当然，还可以整屏整页地移动，如表 9.2 所示。

表 9.2 一般模式：移动光标的方法（2）

按键	说明
Ctrl + f	屏幕向下移动一页，相当于 Page Down 按键（常用）
Ctrl + b	屏幕向上移动一页，相当于 Page Up 按键（常用）
Ctrl + d	屏幕向下移动半页
Ctrl + u	屏幕向上移动半页

前面提到，Vi 有非常强大的功能，在光标移动方面，Vi 不光支持单个字符移动和分屏移动，它还支持许多特殊需求的移动（见表 9.3）。

表 9.3 一般模式：移动光标的方法（3）

按键	说明
+	光标移动到非空格符的下一行
-	光标移动到非空格符的上一行
n <space>	n 表示数字，例如 20。按下数字后再按空格键，光标会向右移动这一行的 n 个字符。例如 20 <space>，光标会向后移动 20 个字符距离
0 或^	这是数字 0：移动到这一行的最前面字符处（常用）
$	移动到这一行的最后面字符处（常用）
H	光标移动到这个屏幕的最上方那一行
M	光标移动到这个屏幕的中间那一行
L	光标移动到这个屏幕的最下方那一行
G	移动到这个文件的最后一行（常用）
nG	n 为数字。移动到这个文件的第 n 行。例如 20G，光标会移动到这个文件的第 20 行（可配合:set nu 使用）
gg	移动到这个文件的第一行，相当于 1G（常用）
n <Enter>	n 为数字。光标向下移动 n 行（常用）

删除

作为文本编辑器，删除的功能非常重要。删除指令用来将不需要的文字段删除。表 9.4 是一些

常用的删除指令。

表 9.4 一般模式：删除指令

按键	说明
d$	删除光标所在位置到该行的最后一个字符
d0	d 的后面是数字 0，删除光标所在处到该行的最前面一个字符
d1G	删除光标所在位置到第一行的所有数据
dd	删除光标所在的那一整行（常用）
dG	删除光标所在位置到最后一行的所有数据
ndd	n 为数字。从光标位置开始，删除向下 n 列，例如 20dd 则是删除 20 列（常用）
nx	n 为数字，连续向后删除 n 个字符。例如要连续删除 10 个字符，则输入“10x”
x，X	在一行字中，x 为向后删除一个字符（相当于[Del]键），X 为向前删除一个字符（相当于[BackSpace]即退格键）（常用）

复制、粘贴

复制、粘贴同样是 Vi 的重要编辑功能。在 Vi 中，复制指令即 yank 的首字母 Y，粘贴即 paste 的首字母 P。yank 在 Vi 中是复制（copy）的意思，yank 在英文字典中是“猛拉”，还有中医上的“拔火罐”的意思，所以还有一种非常有意思的解释：在 Vi 中的复制粘贴，就是把文本块“拔”（yank）起来，“放”（paste）上去。

其实，复制的指令就是一个 y 而已，为什么要用一个单元来说明呢？因为 Vi 复制、贴贴的功能实在太独特了，再配合数字及 Vi 内部的缓冲区来使用的话，会发现，原来 Vi 内还暗藏着很多秘密武器。复制与粘贴的指令操作如表 9.5 所示。

表 9.5 一般模式：复制与粘贴

按键	说明
yy	复制光标所在的那一行（常用）
nyy	n 为数字。复制光标所在的向下 n 行，例如 20yy 则是复制 20 行（常用）
yw	复制一个字
y2w	复制两个字
y1G	复制光标所在行到第一行的所有数据
yG	复制光标所在行到最后一行的所有数据
y0 或 y^	复制光标所在的那个字符到该行行首的所有数据，不含光标所在处字符
y$	复制光标所在的那个字符到该行行尾的所有数据
p，P	p 为将已复制的数据粘贴到光标的下一行，P 则为粘贴在光标上一行。举例来说，当前光标在第 20 行，且已经复制了 10 行数据，则按 p 后，那 10 行数据会粘贴在原来的 20 行之后，即由 21 行开始粘贴。但如果是按 P，那么原来的第 20 行会被变成 30 行（常用）
J	将光标所在行与下一行的数据合成同一行
c	重复删除多个数据，例如[10cj 为向下删除 10 行

注意 在 Vi 中，数字是很有意义的。数字通常表示重复做几次的意思，也有可能表示要去第几行的意思。举例来说，要删除 50 行，则用 50dd，数字加在动作之前。要向下移动 20 行，使用 20j 或者 "20↓" 即可。

关于 Vi 的复制粘贴，还有一些高级应用指令，如下所示。

- "ayy：将本行文字复制到 a 缓冲区。
  a 可为 26 个英文字母中的一个，如果是小写的话，原先的内容会被清除掉，如果是大写的话，其作用相当于 append，会把内容附加到原先内容之后。
  "是 Enter 键旁边的那个上档字符。
- "ap：将 a 缓冲区的内容粘贴上。
  缓冲区的术语在 Vi 中称为 registers，Vi 扩充了相当多的功能，有兴趣深入了解的读者请用 :h registers 命令。用 d、c、s、x、y 等指令改变或删除的内容都是放在 registers 中的。例如，用 dd 删除的一行，也是可以使用 p 来粘贴的。只要是在缓冲区的内容都可以使用 p 来粘贴，不是一定要 y 起来的内容才能用 p。因此认为 p 是 paste 也可以，认为是 put 也可以。
- 5"ayy：复制 5 行内容至 a 缓冲区。
- 5"Ayy：再复制 5 行附在 a 内容之后，即现在 a 中有 10 行内容。

如果忘记了内容在哪个缓冲区，即输入 :reg（冒号命令），会列出所有 registers 的代号及内容。数字、特殊符号的缓冲区是刚刚删除（复制）的内容就默认放在"这个缓冲区中，然后依次是 0，1，2，...9。单独按 p，是取出默认缓冲区的内容。%指的是目前编辑的文件，# 指的是前一次编辑的文件。如果想获得更详细的信息，就输入 :h registers（registers 有个"s" 结尾，不要搞错了），而且 Tab 补全键在这个时候也是起作用的，也就是说，键入 :h regi 再按 Tab 键，Vim 就会帮补全，按了 Tab 后发现不是所要的，那就继续按 Tab 键，直到出现所要的。

重复、取消操作

.（英文句点）重复前次的编辑动作，这个命令非常有用，只要是编辑动作（移动光标不算，冒号命令也不算）都可以按 . 键来重复，要重复几次都可以。

例如，按 yy，然后按 p 就会复制、粘贴上一整行，如果要重复这个动作的话，就可以按 .，也可以把光标移到其他地方后再按。其他 dd，dw，r，cw 等编辑指令都可以这样来重复。如果要重复做某些编辑动作时，那么千万要记住使用这么一个英文句点重复指令。

另外，u 与 Ctrl+r 是很常用的命令：一个是复原，另一个则是重做一次（参见表 9.6）。利用这两个功能按键，编辑起来就得心应手。

表 9.6 一般模式：重复

操作类型	说明
重复（.）	这就是小数点。意思是重复前一个动作。如，如果想重复删除、重复粘贴，按下 . 就可以（常用）
取消（u）	取消前一个操作（常用）
重复 Ctrl+r	重做上一个操作（常用）

查找、替换

查找、替换的功能几乎是每个编辑器必备的功能，那在 Vi 中有没有特殊的地方呢？当然有，别忘了，Vi 是个性十足的编辑器。它最特殊的地方是与正则表达式结合在一起。简单地说，它是一种 pattern 表示法，在执行动作时，如寻找或替换，就会依据这个 pattern 去找，所有符合 pattern 的地方就会执行所有的动作。这里暂不讨论 regexp，要找什么就直接输入什么就是了。Vi 的查找、替换指令及其说明分别见表 9.7 和表 9.8。

表 9.7 一般模式：查找

指令	说明
/word	从光标位置开始，向下寻找一个名为 word 的字符串。例如要在文件内搜索 test 这个字符串，就输入/test 即可（常用）
?word	从光标位置开始，向上寻找一个名为 word 的字符串
n	n 是英文按键，表示“重复前一个搜索的动作”。举例来说，如果刚刚执行/test 去向下搜索 test 字符串，则按下 n 后，会向下继续搜索下一个名称为 test 的字符串。如果是执行?test 的话，那么按下 n，则会向上继续搜索 test 字符串
N	这个 N 是英文按键。与 n 刚好相反，为“反向”进行前一个搜索操作。例如执行/test 后，按下 N 则表示“向上”搜索 test
*	寻找光标所在处之 word（要完全符合）
#	同上，但 *是向前找，# 则是向后找
g*	同*，但部分符合即可
g#	同#，但部分符合即可
Shift+5	进行小括号、中括号、大括号匹配查找

表 9.8 一般模式：替换

指令	说明
:n1、n2s/word1/word2/g	n1 与 n2 为数字。在第 n1 与 n2 行之间寻找 word1 这个字符串，并将该字符串替换为 word2。举例来说，在 100 到 200 行之间搜索 test 并替换为 TEST 则执行:100、200s/test/TEST /g（常用）
:1、$s/word1/word2/g	从第一行到最后一行寻找 word1 字符串，并将该字符串替换为 word2（常用）
:1、$s/word1/word2/gc	从第一行到最后一行寻找 word1 字符串，并将该字符串替换为 word2，且在替换前显示提示符给用户确认（conform）是否需要替换（常用）

9.3.2 Insert Mode 的进出

前面我们说过，启动 Vi 后的默认模式是 Normal Mode。当要进行文本输入时，就需要进入 Vi 的编辑模式也就是 Insert Mode 了。表 9.9 中的指令是进入编辑模式的方式。

表 9.9 进入编辑模式

指令	说明
i、I	插入：在当前光标所在处插入输入文字，已存在的文字会向后退；其中，i 为“从当前光标所在处插入”，I 为“在当前所在行的第一个非空格符处开始插入”（常用）
a、A	a 为“从当前光标所在的下一个字符处开始插入”，A 为“从光标所在行的最后一个字符处开始插入”（常用）
o、O	这是英文字母 o 的大小写。o 为“在当前光标所在的下一行处插入新的一行”；O 为“在当前光标所在处的上一行插入新的一行”（常用）
r、R	替换：r 会替换光标所在的那一个字符；R 会一直替换光标所在的文字，直到按下 Esc 键为止（常用）

要退出 Insert Mode 回到一般模式，只需按 Esc 键即可。

9.3.3 Ed Mode 下操作指令

Ed Mode 是命令模式，在这个模式下，可以进行 Vi 所编辑文件的打开、写入、保存等操作，可以退出 Vi 进程等，还可以对 Vi 编辑器进行编辑环境设置。另外，Vi 的强大更体现在可以在 Vi 的 Ed Mode 中执行 Linux Shell 命令。

文件操作

文件的打开、保存操作是我们一使用 Vi 就必须记住的指令了，具体来说，我们需要掌握如表 9.10 所示的这些指令。

表 9.10 命令行命令模式

指令	说明
:w	将编辑的数据写入硬盘文件中（常用）
:w!	若文件属性为“只读”，强制写入该文件。不过，到底能不能写入，与文件权限有关
:q	退出 Vi（常用）
:q!	若曾修改过文件，又不想存储，使用！为强制退出并不存储文件
:wq	存储后退出，若为:wq! 则为强制存储后退出（常用）
:e!	将文件还原到最原始的状态
ZZ	若文件没有更改，则不存储退出；若文件已经更改，则存储后退出
:w [filename]	将编辑的数据存储成另一个文件（类似另存新文件）
:r [filename]	在编辑的数据中，读入另一个文件的数据。即将 filename 这个文件内容加到光标所在行的后面
:n1、n2 w [filename]	将 n1 到 n2 的内容存储成 filename 文件

注意 感叹号（!）在 Vi 当中常常具有强制的意思。

环境设置

如果以 Vi 编辑器来搜索一个文件内部的某个字符串时，这个字符串会被反白，而下次再次以 Vi 编辑这个文件时，该搜索的字符串还是存在。当编辑其他文件时，如果其他文件中也存在这个字符串，也会自动反白显示。另外，当重复编辑同一个文件时，第二次进入该文件，光标竟然就在上次离开的那一行上，非常方便。但是，怎么会这样呢？这是因为 Vi 会主动将曾经做过的行为记录下来，以便下次工作。那个记录动作的文件就是~/.Viminfo。每个人的用户主目录都应该有这个文件。这个文件是自动产生的，不必自行建立。在 Vi 中所做过的操作都可以在这个文件中找到。

由于在某些版本的 Vi 中利用搜索功能时，它并不会反白显示，这些版本则会主动地进行缩排（就是当按下 Enter 键编辑新行时，光标不会在行首，而是在与上一行的第一个非空格符处对齐）。其实这些都可以进行设置，即进行 Vi 的环境设置。Vi 的环境设置参数有很多，如果想知道当前设置值，可以在一般模式时输入 ":set all" 来查看。不过，设置项目实在太多了，表 9.11 列出了一些平时比较常用的简单的设置值，以供参考。

表 9.11 Vi 的部分环境设置参数

参数	说明
:set nu	设置行号。取消的话，就用:set nonu
:set hlsearch	设置是否将搜索的字符串反白。默认值是 hlsearch，如果不想反白，就用:set nohlsearch
:set autoindent	是否自动缩排。autoindent 是自动缩排，不想缩排就用:set noautoindent
:set backup	是否自动存储备份文件。一般用 nobackup，如果设置 backup，当更改任何一个文件时，则源文件会被另存为一个名为 filename~的文件。举例来说，编辑 hosts 时，设置 :set backup，那么当更改 hosts 时，在同一目录下，就会产生 hosts~文件名的文件，记录原始的 hosts 文件内容
:set ruler	是否在右下角显示状态行说明
:set showMode	是否在左下角的状态行显示-INSERT-
:set backspace=（012）	一般来说，如果按 i 键进入编辑模式后，可以利用退格键（BackSpace）来删除任意字符。但是，某些版本则不支持这样做。此时，就可以通过 backspace 来设置。当 backspace 为 2 时，可以删除任意值；当 backspace 为 0 或 1 时，仅可删除刚刚输入的字符，而无法删除原来就已经存在的字符
:set all	显示当前所有的环境参数设置值
:syntax（offlon）	是否根据程序相关语法显示不同的颜色。举例来说，在编辑一个纯文本文件时，如果是以# 开始，那么该行就会变成蓝色。如果懂得写程序，那么这个:syntax on 还会主动帮助调试。但是，如果仅是编写纯文本文件，要避免颜色对屏幕产生的干扰，则可以用:syntax off 取消这个设置

9.4 Vi 的高级应用

其实，如果大家熟悉了 9.3 节的操作指令，我们就可以在 Vi 编辑器中畅通无阻了。但是 Vi 这

个强大的编辑器还给我们提供了更多的惊喜。什么惊喜呢？Vi 还提供了很多高级应用指令，通过这些指令，可以大大提高编辑文档的效率。

9.4.1 块选择（Visual Block）

刚刚提到的简单的 Vi 操作过程中，几乎都是以行为单位操作。那么，如果想要解决一个块范围的问题呢？举例来说，像下面这种格式的文件：

```
192.168.1.1  host1.class.net
192.168.1.2  host2.class.net
192.168.1.3  host3.class.net
192.168.1.4  host4.class.net
.....中间省略......
```

如果想复制前面的 IP 地址部分，后面的主机名称部分不复制，怎么办？这时就需要使用块选择（Visual Block）的功能。当按下 v 键或者 V 键或者 Ctrl+v 键时，光标移动过的地方就会开始反白，这三个按键的意义参见表 9.12。

表 9.12 块选择的按键意义

按键	说明
v	字符选择，会将光标经过的地方反白选择
V	行选择，会将光标经过的行反白选择
Ctrl+v	块选择，可以用长方形的方式选择数据
y	复制反白的地方
d	将反白的地方删除掉

以上面的 IP 地址对应主机名称为例，如果想复制 IP 地址的话，而且仅想要前 4 行，那么可以按如下方式进行：

- 将光标移到第一行的第一个字符。
- 按 Ctrl+v 键（按着 Ctrl 键不放，再按 v 键）。
- 移动方向键，向下向右移动数格，让整个反白区域覆盖 191.168.1.1 到 192.168.1.4。
- 按 y 键进行复制（此时反白会自动不见）。
- 移动到任何想要插入的区域，按 p 键就可以插入刚刚复制的块内容。举例来说，移动到第 1 行的第 13 个字符处按小写的 p，看看会怎样。

这个块选择在格式整齐的文件中很有用。尤其是想要大量复制其中一个块，而不是整行复制的场合中，会非常有用。

9.4.2 排版功能

大家在文字编辑过程中常常要让文档有一个统一的格式，如一定的位置统一退格且对齐等。这时 Vi 的排版功能指令将大有帮助。

一般模式下的排版

- >>：整行向右移一个 shiftwidth（默认为 8 个字符，可重设）。
- <<：整行向左移一个 shiftwidth（默认为 8 个字符，可重设）。

set shiftwidth? 可得知目前的设定值。:set shiftwidth=4 可重设为 4 个字符。shiftwidth 可简写成 sw。

- n<<：将 n 行向左移动一个 Tab 键。
- n>>：将 n 行向右移动一个 Tab 键。
- gqip：整段重排。
- gqq：本行重排。

重排的依据也是 textwidth。这里的重排是指输入文字时没有按 Enter 键，这样会形成一个很长的行（虽然屏幕上会做假性折行），重排后，则会在每一行最后加入 EOL。gq 重排功能是 Vim 独有的功能。

命令行模式下的排版

- :ce（nter）：本行文字居中对齐。
- :ri（ght）：本行文字靠右对齐。
- :le（ft）：本行文字靠左对齐。

所谓居中、靠左、靠右，是参考 textwidth（tw）的设定。如果没有设定 tw，则默认为 80，就是以 80 个字符的总宽度为标准来放置。当然也可以像 sw 一样马上重设默认值。

9.4.3 Vi（m）书签功能

Vi（m）还有一个秘密武器，即书签功能，简单地说，可以在文章中某处做个记号（marks），然后到其他地方去编辑，在调用这个 mark 时又会回到原处。这在修改文件中非常有用。

- Mx：x 代表 26 个小写英文字母，这样光标所在处就会被 mark。
- 'x：回到书签原设定位置。
  '是 backward quote，位于 Tab 键上面。
- 'x：回到书签设定行行首。
  '是 forward quote，位于 Enter 键旁边。

下面是 Vim 对于书签的扩充功能。

- 小写字母：只作用于单一文件内。
- 大写字母：可作用于编辑中各文件间。
- 数字：可作用于前次编辑的十个文件。

数字的用法比较特殊，'0 是回到前一次编辑文件中退出前的最后位置，'1 则是回到倒数第二次编辑文件的最后位置，依此类推。不必使用 m 来标示，Vim 会自动记忆。其实这是 Viminfo 的功能，要认真追究的话，可以输入:h Viminfo-file-marks 查看。关掉 Viminfo，就没这个功能了。所谓前次指的是前次启动的 Vim。

- :marks 得知目前所有书签的列表。

9.4.4 多文件同时编辑

假设要将 hosts 内的 IP 地址复制到 /etc/hosts 文件，该如何编辑呢？我们知道，在 Vi 内可以使用:r filename 来读入某个文件的内容，不过，这样是将整个文件读入。如果只是想将部分内容读入时，多文件同时编辑就很有用了。我们可以选择将 Vim 后面接多个文件来同时打开它们。多文件编辑的按键请参考表 9.13。

表 9.13 多文件编辑按键说明

按键	说明
:n	编辑下一个文件
:N	编辑上一个文件
:files	列出当前 Vim 打开的所有文件

现在可以做一下练习。假设要将刚才笔者提供的 hosts 内的 IP 地址复制到 /etc/hosts 文件内，可以这样做。

```
user@ubuntuer:~$ Vi hosts /etc/hosts
# 在这个文件中利用上一小节提到的块选择，按下 Ctrl+v 键来进行块选择并复制。
# 然后输入:n，在命令行的地方输入这些，就会转到下一个文件，这个时候，
# 就可以按 p，将刚刚复制的 IP 地址粘贴到文件中。如果输入:files，则有以下结果：
192.168.1.4  host4.class.net
192.168.1.5  host5.class.net
~
~
:files
  1 %a "hosts"  line 1
  2 #  "/etc/hosts"    line 1
Hit ENTER or type command to continue

# 在命令行输入:files 就可以显示当前所编辑的文件信息。
```

由此可知，利用多文件编辑的功能，可以很快速地将需要的数据复制到正确的文件中。当然，这个功能也可以利用窗口界面来实现。

9.4.5 多窗口功能

当一个文件非常大，查看到后面的数据时，想要“对照”看前面的数据，是否需要使用 Ctrl+f

与 Ctrl+b 键来前后地查看？或者遇到有两个需要对照着看的文件，不想使用前一小节提到的多文件编辑功能。这样的情况下，可以使用 Vim 打开两个窗口。

在命令行模式下输入“:sp {filename}”，其中 filename 可有可无，如果想在新窗口启动另一个文件，就加入文件名，否则仅输入:sp，在两个窗口间会出现同一个文件。例如笔者使用 Vim hosts 后，再使用:sp /etc/hosts，会出现如下内容：

```
# Do not remove the following line, or various programs
# that require network functionality will fail.
127.0.0.1   localhost.localdomain localhost
192.168.1.11  vbird-work
192.168.1.2   vbird-server
~
/etc/hosts    5, 1 All
192.168.1.1  host1.class.net
192.168.1.2  host2.class.net
192.168.1.3  host3.class.net
192.168.1.4  host4.class.net
192.168.1.5  host5.class.net
hosts  1, 1 Top
```

两个文件同时在一个屏幕上显示，还可以利用 Ctrl+w+j 键及 Ctrl+w+k 键在两个窗口之间切换，这样，复制、查看等操作就变得很简单。命令的功能有很多，只要记住表 9.14 给出的这些即可。

表 9.14 多窗口情况下的按键功能

按键	功能
:sp [filename]	打开一个新窗口，如果加 filename，表示在新窗口打开一个新文件，否则表示两个窗口为同一个文件内容（同步显示）
Ctrl+w+j	按键的按法是：先按住 Ctrl 不放，再按 w 后放开所有的按键，然后再按 j，则光标可移到下方的窗口
Ctrl+w+k	同上，不过光标移动到上方的窗口
Ctrl+w+q	其实就是:q，结束并退出。举例来说，如果想结束下方的窗口，利用 Ctrl+w+j 键移到下方窗口后，输入:q 即可退出，也可以按 Ctrl+w+q 键

9.5 Vi 系统配置

在前面 9.3.3 Ed Mode 的指令介绍中，我们讲过打开 Vi 设置环境的一些简单指令，但是，是否每次使用 Vim 都要重新设置一次各个参数值呢？没有必要。可以通过设置文件来直接设定习惯的 Vim 操作环境。Vim 设置值一般放在/etc/Vimrc 文件中，不过，建议不要修改它。可以修改 ~/.Vimrc 文件（默认不存在，请自行手动建立），写入所希望的设置值。

需要注意的是，~/.vimrc 的设置项对于 Vim 是永久生效的，除非再次编辑～/.vimrc 取消相应设置。举例来说，可以是这样的一个文件：

```
user@ubuntuer:~$ Vi ~/.Vimrc
:set hlsearch
```

```
:set backspace=2
:set autoindent
:set ruler
:set showMode
:syntax on
```

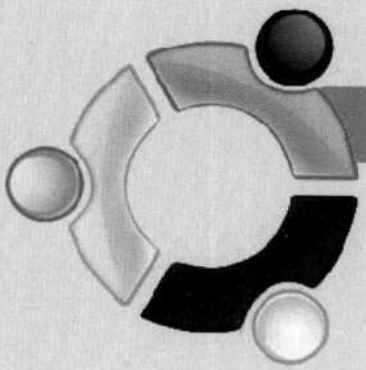
这样，以后每次重新打开 Vim 编辑某个文件时，默认编辑环境设置都是上面这样的，如此可以方便操作。本单元可说是 Vi（m）的微调功能，可依个人的喜好做有限度的调整。

另外，需要说明的是，Vimrc 这个路径在不同的 Linux 版本中可能会不同。有些版本存在 /usr/share/Vim/Vimrc 和/etc/Vim/Vimrc，但是它们也是同时指向一个 Vimrc 的，修改这个文件即可。

Vi 可以进行配置的项目比较多，一些平时比较常用的系统配置参数见表 9.11。

9.6 课后练习

1. Vi 和 Vim 的区别及联系是什么？
2. 如何进入 Vi ？如何在进入 Vi 时顺带打开一个文件？
3. 进入 Vi，可是没法写入想输入的文字，请问为什么？Vi 有哪三种操作模式？
4. 请问移动光标的指令有哪些？
5. 输入错了一段文字，如何最快地删除呢？
6. 编辑文档时，想互换一下两个段落文字的顺序，请问有什么最好的办法？
7. 编辑时想把一个文档中的某个单词改成首字母大写，请问如何操作最快？
8. 请问如何以另一个文件名保存文档？
9. 什么是块选择？
10. 请试试 Vi 的标签功能。
11. 如何给 Vi 编辑器设置显式行号？如何进行一些其他的环境变量设置？
12. 如何通过 Vi 打开两个编辑文档，并进行两个文档之间的文档内容复制？



Part

03

Ubuntu 日常管理

Ubuntu日常应用离不开系统文件、目录、用户、设备及进程的管理等，在本篇中，我们着重介绍以下几方面的内容：Ubuntu文件与目录管理，文件的属性与权限；Ubuntu压缩与查找系统，用户管理，硬盘管理，用户磁盘配额；Ubuntu设备管理及进程管理。通过这些内容，我们基本可以轻松地完成Ubuntu系统的日常管理工作。

Chapter10 文件与目录管理

在每个操作系统中，文件与目录的管理都是相当重要的，毕竟我们使用计算机就是为了对数据文件进行操作，包括如何进行文件复制、移动、删除、查找。本章讨论了 Ubuntu 的文件系统结构，以及各类能够用来管理文件和目录的工具或命令。

为了系统安全，在 Ubuntu 中只有超级用户才能获得所有系统级别的文件和目录的访问权。在本章的学习演练中，如果操作用户没有打开、删除或执行文件的权限，会看到一条错误消息，通知用户访问被拒绝，这是正常行为，它用来防止不具备权限的用户删除重要的系统文件。这部分内容将在下章重点介绍。

10.1 Ubuntu 的文件系统

在 GNU Linux 里面，所有的目录都是从根目录/扩展的树状结构。硬盘至少有一块分割区给根目录/（root），其他的目录就会以此为基础存储在根目录的某个子目录下。在早期，GNU/Linux 的每个发行版都有自己首选的目录分配方式，不同的版本有不一样的目录分配，因此而造成很多用户的困扰。为了解决这个问题，Filesystem Hierarchy Standard（FHS）就出面制定了一些标准，让各个发行版有个纲要可以知道哪个目录要放什么样的东西。当前 FHS 定义的是每个目录的大纲，所以很多配置文件的文件名或许还是有所不同，但是几乎都放在同一个目录底下了。这样用户在寻找的时候就方便许多了。

10.1.1 Ubuntu 目录体系

Ubuntu 目录体系也遵循 FHS 的目录命名规则，表 10.1 列出了 Ubuntu 下的基本目录结构。

表 10.1 Ubuntu Linux 目录结构

目录名称	功能描述
/	文件系统根目录
/bin	存放常用的可执行文件
/boot	存放系统的启动文件，包括 Grub，lilo 启动程序
/dev	设备文件，包括硬盘、键盘、光驱、鼠标等
/home	用户主目录的存放目录
/lib	共享的系统程序库文件
/lost+found	存放由 fsck 放置的文件，一般是非正常关机的时候产生
/media	系统自动挂载光驱、软驱、USB 存储器后，存放临时读入文件
/mnt	常用的挂载点
/opt	第三方程序的安装处

（续表）

目录名称	功能描述	
/proc	存放描述系统状态或系统进程的文件，例如，cpuinfo 文件存放了系统当前 CPU 的工作信息	
/root	根用户的主目录	
/sbin	存放可执行的文件，它和/bin 目录的区别是，/sbin 目录存放更多的是系统管理方面的命令	
/srv	存放系统提供的服务数据	
/sys	存放系统设备组织或层次结构，并向用户程序提供详细的内核数据信息	
/tmp	用户或程序的临时文件，所有的用户都对该目录有读写权限	
/usr	存放与系统用户相关的文件或目录，主要是应用程序及支持它们的库文件	
	/usr/X11R6	存放 X-Windows 的目录
	/usr/bin	应用程序的可执行文件
	/usr/sbin	用户或管理员的标准命令
	/usr/include	C/C++开发环境的标准 include 文件
	/usr/lib	应用程序及其链接库
	/usr/local	系统管理员安装的应用程序目录
	/usr/share	存放使用手册等共享文件的目录
	/usr/src	应用程序源代码目录
	/usr/games	存放 Ubuntu 下游戏的可执行文件
/var	存放那些不断扩充着的文件，为了保持/usr 的相对稳定，那些经常被修改的文件可以放在这个目录下	
	/var/cache	应用程序缓存目录
	/var/crash	系统错误信息目录
	/var/games	游戏运行数据存放目录
	/var/lib	系统运行时随时改变的文件
	/var/lock	文件锁定记录
	/var/log	日志目录
	/var/mail	电子邮件目录
	/var/opt	存放/opt 目录下变量数据
	/var/run	存放进程运行数据
	/var/spool	存放电子邮件、打印等的任务队列文件
	/var/tmp	存放临时文件

10.1.2 绝对路径和相对路径

对文件进行访问时，需要用到“路径”（Path）的概念。顾名思义，路径是指从树形中的某个目录层次到某个文件的一条路线。此路径的主要构成是目录名称，中间用/分开。任一文件在文件系统中的位置都是由相应的路径决定的。

用户在对文件进行访问时，要给出文件所在的路径。路径又分相对路径和绝对路径。绝对路径是指从/开始的路径，也称为完全路径；相对路径是从用户工作目录开始的路径。相对路径有 4 种符号表示方法，大家需要记住。

- .代表“当前目录（Current directory）”。
- ..代表“上一层目录”。
- ~代表“用户主目录”。
- -代表“上一个使用目录”。

应该注意到，在树型目录结构中到某一确定文件的绝对路径只有一条。绝对路径是确定不变的，而相对路径则随着用户工作目录的变化而变化。

10.2 文件目录的图形化管理

在 Ubuntu 下使用图形管理界面进行文件管理与在 Windows 下很相似，都是通过所见所得的方式及鼠标的点选来管理文件目录。

10.2.1 Ubuntu 位置菜单

在 Ubuntu 系统上，有一个专门的菜单来管理组织其目录及文件，通过桌面上的【位置】菜单打开文件管理菜单，如图 10.1 所示。

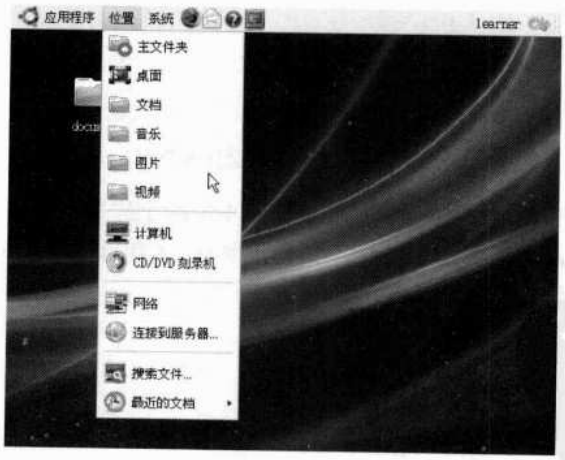


图 10.1 Ubuntu 的【位置】菜单

在菜单中，每一个选项代表一个存取资源，在第一条分割线和第二条分割线之间是【计算机】和【CD/DVD 刻录机】，执行【计算机】命令，系统将打开根目录为主的窗口，窗口里显示根目录下所有的可见目录，最上面一部分，也就是第一条分割线以上的部分都是个人文件夹，用户可以在这些文件夹中存储自己的文件资料，这个目录的划分跟 Windows 下的【我的文档】完全相似。

若想要在 Ubuntu 预设的桌面环境中像 Windows 一样轻松地通过【网上邻居】来访问网络上其他的计算机共享的文件资源（包含 Windows 系统），则可以通过执行【网络】或者【连接到服务器】命令来配置管理访问局域网内共享的文件资源。

10.2.2 Nautilus 管理器

GNOME 默认的文件浏览器是 Nautilus，功能和界面跟 Windows 下的资源管理器很相似。它为用户提供了一个浏览文件系统的图形化界面，使用 Nautilus 可以高效地管理文件和目录（包括新建、删除、剪切、复制、重命名等操作），而且还能根据文件的类型使用正确的应用程序打开。可以通过执行【位置】|【主文件夹】命令打开 Nautilus，如图 10.2 所示。

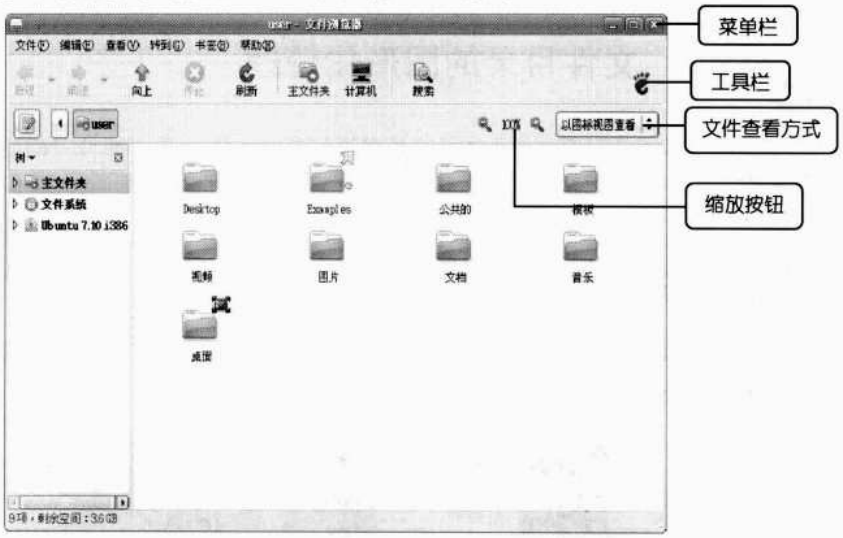


图 10.2 Ubuntu 的文件浏览器

图 10.2 中，左侧是 Nautilus 的导航面板，当前选中的是【主文件夹】，因此右侧面板显示的是【主文件夹】下的子目录，这些子目录以字母的顺序显示。如果没有出现导航面板，可以使用快捷键 F9 来打开它。接下来讲述 Nautilus 的一些基本操作。

导航工具栏

导航工具栏位于 Nautilus 的顶端，如图 10.3 所示。

图 10.3 Nautilus 导航工具栏

- 单击【后退】，Nautilus 就跳转到最近一次访问的目录。
- 单击【前进】，Nautilus 将返回到最近一次单击【后退】按钮前的目录。
- 单击【向上】，Nautilus 将返回当前目录的上一级目录，也就是父目录。

- 单击【主文件夹】，Nautilus 将返回用户的主文件夹。

文件迅速定位

有些时候我们已经知道了文件的具体路径位置，这时候可以在 Nautilus 下直接输入路径来访问该文件。执行【转到】|【位置】命令打开【位置】文本框（或者按快捷键 Ctrl+L），在【位置】文本框输入文件路径，然后按回车键，如图 10.4 所示。

图 10.4　文件迅速定位

搜索文件

执行【转到】|【搜索文件】命令打开搜索文件功能（按快捷键 Ctrl+F 或者单击导航工具栏中的【搜索】），输入文件名，然后按回车键，如图 10.5 所示。

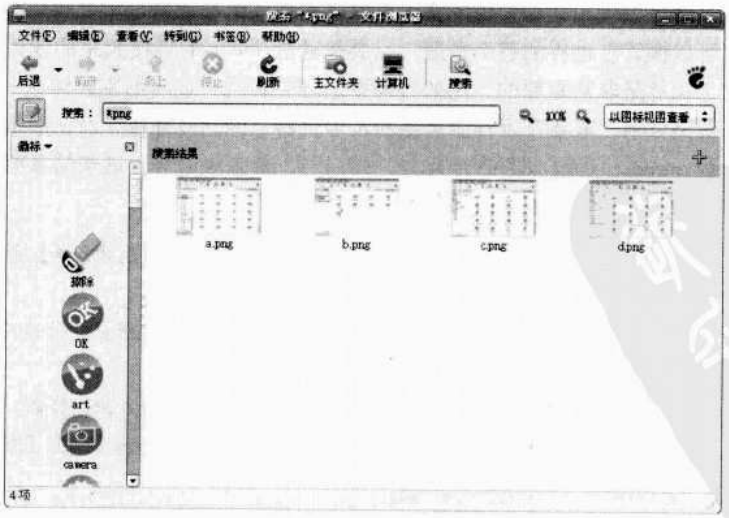


图 10.5　文件搜索

创建新文件

在 Nautilus 窗口中，右键单击空白处，弹出如图 10.6 所示的快捷菜单，选择【创建文件夹】或【创建文档】，再输入文件名。

图 10.6 Nautilus 右键菜单

复制/移动文件

打开 Nautilus 窗口，选择需要复制/移动的文件，单击右键，选择菜单中的【复制】或者【剪切】选项，然后找到目标目录，单击右键，从菜单中选择【粘贴】。

删除文件

在窗口中选中要删除的文件或目录，单击右键，选择菜单中的【转移到回收站】或者直接按键盘上的 Delete 键。

10.2.3 访问远程文件

在 Ubuntu 下，还可以使用功能强大的文件管理器来访问远程服务器的文件，这个服务器可以是在局域网内，也可以搁置在世界的各个角落，只要是通过 Internet 连接起来的就可以。如果想传送大量的文件，这个功能是非常重要的，比如当需要远程更新网页或者给别人传送一些文件的时候。在访问远程服务器文件前，需要在远程服务器和本地机器之间建立连接，而一般情况下，建立连接需要提供连接参数，包括：远程服务器 IP 地址、端口号、登录账号等，这些信息都需要远程服务器管理员提供给你。以下是访问远程服务器的文件的操作步骤。

Step 01 建立与远程服务器的连接，打开文件浏览器，单击【文件】|【连接到服务器】命令，看到如图10.7所示窗口。

Step 02 输入登录连接参数，包括：服务类型、服务器IP、端口、文件夹和连接要使用的名称。然后单击【连接】按钮，这时候桌面上就能看到新加了一个图标。

Step 03 双击桌面上新增的图标，这时候弹出如图10.8所示的对话框提示输入密码登录。

Step 04 单击【仍然登录】按钮，在弹出的如图10.9所示的对话框中输入密码，单击【连接】按钮。

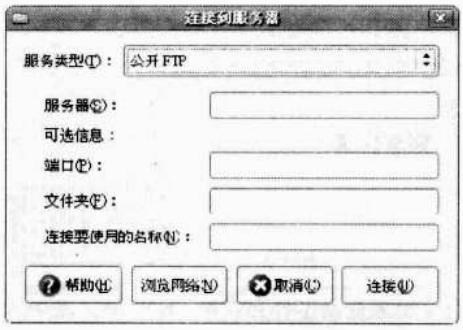


图 10.7　访问远程服务器文件的连接登录窗口

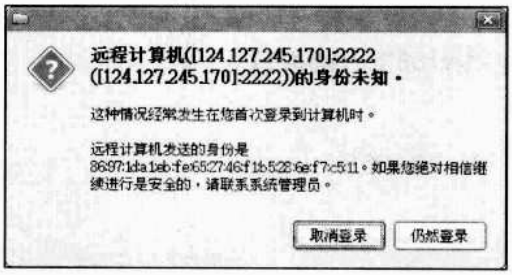


图 10.8　提示登录

图 10.9　输入登录密码

连接成功后，就可以像在本地机器上一样操作远程的文件。

10.3　文件和目录的日常使用

前面讲述了 Ubuntu 使用 Nautilus 来管理文件目录，在这一节中，将简单介绍对文件目录进行基本管理的命令。

- cd：改变目录位置。
- pwd：显示当前目录的绝对路径。
- ls：显示文件名称、属性等。

- cp：复制文件或目录。
- mv：移动文件或目录。
- rm：删除文件或目录。
- mkdir、rmdir：创建、删除目录。

10.3.1 cd 指令

cd（change directory）指令用来变换工作目录至 dirName，其中 dirName 表示法可以为绝对路径或相对路径。若省略目录名称，则变换至用户的 home directory（也就是刚登录时所在的目录）。

语法

```
user@ubuntu: ~$ cd 【相对路径或绝对路径】
```

参数说明

小心区分“相对路径”与“绝对路径”。

例子

```
user@ubuntu: ~$cd ..                    // 回到上一层目录
user@ubuntu: ~$cd ../home               //相对路径的写法
user@ubuntu: ~$cd /var/www/html         //绝对路径的写法
user@ubuntu: ~$ cd ~                    //回到用户主目录
user@ubuntu: ~$ cd ~test
//回到 test 用户的用户主目录，前提是该用户允许你访问它的目录
```

说明

注意，在 cd 命令与路径之间存在一个空格。用户登录系统后，默认的当前路径会自动切换到用户主目录下，比如根用户是/root，其他用户在/home/username 下，比如，系统下的一个用户 user，登录系统后就会切换到/home/user 路径下。

10.3.2 pwd 指令

pwd（print working directory）指令用来显示当前所在目录的命令。如果在目录中上下切换查看，会很容易迷失方向或者忘记当前目录的路径，这个时候可以输入 pwd 查看当前的目录全路径。

语法

```
user@ubuntu: ~$ pwd
```

例子

```
user@ubuntu: ~$pwd
/home/user          //显示当前所在的目录
```

10.3.3 ls 指令

ls 用于列出文件或目录的信息。ls 命令是 Linux 下最常用的命令之一。它有大量的参数选项，其中有很多参数非常有用。

语法

```
user@ubuntu: ~$ ls [-abcCilS]
```

参数说明

- -a：列出全部的文件（包括隐藏文件）。
- -b：对文件名中的不可显示字符用八进制字符显示。
- -B：--ignore-backups，不列出任何以~ 字符结束的项目。
- -c：按文件的修改时间排序。
- -C：分成多列显示各项。
- -d：--directory，当遇到目录时列出目录本身而非目录内的文件。
  -D：--dired，产生适合 Emacs 的 dired 模式使用的结果。
- -i：打印出 inode 的值。
- -l：长的列出，连同文件大小的数据等等。
- -S：以文件大小排序。
- --color=never：不显示颜色。
- --color=always：均显示颜色。
- --color=auto：由系统自行判断是否显示颜色。

例子

```
user@ubuntu: ~$ ls -al
total 48
drwxr-x---  4 root   root   4096 May 10 00:37 .
drwxr-xr-x 21 root   root   4096 May 10 20:16 ..
-rw-------  1 root   root 524 May 10 00:40 .bash_history
-rw-r--r--  1 root   root  24 Jun 11  2008 .bash_logout
-rw-r--r--  1 root   root 266 Jun 11  2008 .bash_profile
-rw-r--r--  1 root   root 249 May  6 20:50 .bashrc
-rw-r--r--  1 root   root 210 Jun 11  2008 .cshrc
drwx------  2 root   root   4096 May  9 11:06 .gnupg
-rw-------  1 root   root 524 Jan 16 14:37 .mysql_history
drwx------  2 root   root   4096 May  9 11:06 .ssh
-rw-r--r--  1 root   root 196 Jul 11  2007 .tcshrc
-rw-r--r--  1 root   root   1126 Aug 24  2006 .Xresources
user@ubuntu: ~$ ls
bin dev  etc lib   misc  opt root  tftpboot  usr
boot  disk1  home  lost+found  mnt proc  sbin  tmp   var
user@ubuntu: ~$ ls --color=never
bin dev  etc lib   misc  opt root  tftpboot  usr
boot  disk1  home  lost+found  mnt proc  sbin  tmp   var
```

说明

一般情况下，要知道某个目录下各个文件或子目录的具体名称和信息，可以使用 ll 命令，其实就是 ls -l 的意思（如果 ll 命令不生效，那先修改一下当前用户的.bashrc 文件，为 ls 命令添加一个别名，请参考 8.4.3 节命令别名）。另外，如果远程 telnet 登入系统后，由于 Ubuntu Linux 默认使用有颜色的方式显示，而使用的终端背景是黑色的话，那蓝色字体就很难看清楚了，这个时候通常使用 ls --color=never 来去掉颜色。那如果想让 ls 默认没有颜色的话，可以在/root/.bashrc 或者用户目录的.bashrc 这个文件中加入下面这一行：alias ls='ls --color=never'。这样，在使用 ls 命令的时候，默认情况下就不显示颜色了。

10.3.4 cp 指令

cp 指令用于复制文件到目的文件。这个命令平时会经常用到，因为我们常常需要复制文件，移动文件。

语法

```
user@ubuntu:~$ cp [-adfrsu] [来源文件] [目的文件]
```

参数说明

- -a：在复制目录时使用。它保留链接、文件属性，并递归地复制目录。
- -d：在进行复制的时候，如果是复制到链接文件，若不加任何参数，则默认情况下会将链接文件指向的源文件复制到目的地，若加-d 时，则链接文件可原封不动地将链接这个快捷方式复制到目的地。
- -f：删除已经存在的目标文件而不提示。
- -r：可以进行目录的递归循环复制。
- -s：做成链接文件，而不复制，功能与 ln 命令相同。
- -u，--update：如果来源文件比较新，或者是没有目的文件，那么才会进行复制的操作，可用于文件备份的策略中。

例子

```
user@ubuntu:~$ cp  .bashrc bashrc  #将.bashrc 复制成 bashrc 这个文件。
user@ubuntu:~$ cp -r /bin /tmp/bin #复制整个目录到指定目录。
user@ubuntu:~$ cp -s .bashrc bashrc #将.bashrc 建立一个链接文件，文件名为 bashrc
user@ubuntu:~$ cp -u /home/.bashrc .bashrc
#先检查 /home/.bashrc 是否与 .bashrc 不同，
#如果不同的话就开始复制一份！如果相同则不做任何动作！
```

说明

如果系统有大文件的备份需要，而这个文件的更新频率很低，那么每次备份都需要再复制一份吗？看来是不需要了！你可以使用“cp -u 来源文件 目的文件”来备份。这样，当文件被修改过后，才会进行复制的动作。

10.3.5 mv 指令

mv 指令用于移动或重命名现有的文件或目录。

语法

```
mv [-bfiuv][--help][--version][-S <附加字尾>][-V <方法>][源文件或目录]
[目标文件或目录]
```

参数说明

- –b 或––backup：若需要覆盖文件，则覆盖前先备份。
- –f 或––force：若目标文件或目录与现有的文件或目录重复，则直接覆盖现有的文件或目录。
- –i 或––interactive：覆盖前先询问用户，即采用交互方式。如果 mv 操作将导致对已存在的目标文件的覆盖，此时系统询问是否覆盖，要求用户回答 y 或 n，这样可以避免误覆盖文件。
- –S<附加字尾>或––suffix=<附加字尾>：与–b 参数一并使用，可指定备份文件所要附加的字尾。
- –u 或––update：在移动或更改文件名时，若目标文件已存在，且其文件日期比源文件新，则不覆盖目标文件。
- –v 或––verbose：执行时显示详细的信息。
- –V=<方法>或––version–control=<方法>：与–b 参数一并使用，可指定备份的方法。
- ––help：显示帮助。
- ––version：显示版本信息。

例子

```
user@ubuntu: ~$ mv  /usr/xu/* .  <==将/usr/xu/目录下所有文件移动到当前目录下。
user@ubuntu: ~$ mv text.txt text.txt.bak
                        <== 这种使用方式相当于给 text.txt 重命名为 text.txt.bak
user@ubuntu: ~$ mv text.txt text1.txt /tmp<==将 text.txt 与 text1.txt 移动到
/tmp 这个目录下。
```

注意 最后一个才是最终的目标文件，其他的都是源文件。

说明

当需要移动文件或目录的时候，这个命令就很重要。同样的，也可以使用–u (update) 来测试文件是否已经更新，是否需要移动到别的地方。除此之外，这个命令还有一个用途，那就是修改文件名，我们可以很轻易地使用 mv 来变更文件的名称，如上面例子中所示。

10.3.6 rm 指令

rm 指令用来删除档案、目录，以及当前目录有适当权限的所有用户。

语法

```
rm [options] name...
```

参数说明

- –i：删除前逐一询问确认。
- –f：即使原文件属性设为只读，亦直接删除，无需逐一确认。
- –r：将目录及以下文件逐一删除。

例子

```
user@ubuntu: ~$  cp .bashrc bashrc  <==建立一个新文件 bashrc
user@ubuntu: ~$  rm bashrc          <==会显示如下的提示：rm: remove `bashrc'?
user@ubuntu: ~$  mkdir testing
user@ubuntu: ~$  cp .bashrc testing
user@ubuntu: ~$  rmdir testing
rmdir: `testing': Directory not empty
                                    <==由于 testing 里面有 .bashrc，所以删除失败
user@ubuntu: ~$  rm -rf testing     <==递归删除该目录下的所有文件与目录
```

说明

在系统中创建文件很容易，而系统中随时会有文件变得过时且毫无用处，我们就可以用 rm 命令将其删除。该命令的功能为删除一个目录中的一个或多个文件或目录，它也可以将某个目录及其下的所有文件及子目录均删除。对于链接文件，只是删除了链接，原有文件保持不变。需要注意的是，为了避免文件的误删除操作，rm 命令默认情况下有–i 这个参数，意思是指每个文件被删除前需要用户确认一次，以防止误删除。如果要连目录下的文件都一起删除的话，例如子目录里面还有子目录时，那就要使用–rf 这个参数了。不过，使用 rm –rf 这个命令之前，请千万注意，因为该目录或文件肯定会被删除，而系统不会再次询问是否需要删除，所以只有在确定该目录不要了，那么再使用 rm –rf 来递归删除目录内容。

10.3.7 mkdir、rmdir 指令

mkdir 命令

语法

```
user@ubuntu: ~$ mkdir [-mp] [目录名称]
```

参数说明

- –m：对新建目录设定权限，当然也可以使用 chmod 来修改权限。
- –p：可以是一个路径名称。此时若路径中的某些目录尚不存在，加上此选项后，系统将自动建立那些尚不存在的目录，即一次可以建立多个目录。

例子

```
user@ubuntu: ~$ cd tmp
```

```
user@ubuntu: ~$ mkdir test   <==建立名称为 test 的目录
user@ubuntu: ~$ mkdir -p test1/test2/test3/test4 <==直接建立 test2 等上层目录
user@ubuntu: ~$ mkdir -m 711 testqq   <==建立权限为 711 的目录
user@ubuntu: ~$ ll test*
drwxrwxr-x  1 user  root 0  2008-06-13 22:11  test/
drwxrwxr-x  1 user  root 0  2008-06-13 22:11 test1/
drwx--x--x  1 user  root 0  2008-06-13 22:11 testqq/
```

⚙说明

该命令用来创建指定名称的目录，当然前提是创建目录的用户在当前目录中具有写权限，并且新建目录名称不能是当前目录中已有的目录或文件名称。在默认的情况下，用户所需要的目录要一层一层地建立才行。例如，假如要建立一个目录/home/user/test1/test2，那么首先必须要有/home，然后才能有/home/user，再有/home/user/test1，才可以建立 test2 这个目录。假如没有/home/user/test1 时，就没有办法建立 test2 目录了。不过，现在有个更简单有效的方法，那就是加上-p 这个参数，可以直接输入“mkdir -p /home/user/test1/test2”，则系统会自动把/home，/home/user、/home/user/test1 和/home/user/test1/test2 依次建立起目录的层次结构，并且，如果该目录本来就已经存在时，系统也不会显示错误信息。

rmdir 命令

⚙语法

```
user@ubuntu: ~$ rmdir [-p] [目录名称]
```

⚙参数说明

-p：递归删除目录，当子目录删除后其父目录为空时，也一同被删除。

⚙例子

```
user@ubuntu: ~$ rmdir test  <==删除名称为 test 的目录
user@ubuntu: ~$ ll
drwxrwxr-x  1 user  root 0  2008-06-13 22:11 test1/
user@ubuntu: ~$ rmdir test1
rmdir: 'test1': Directory not empty
user@ubuntu: ~$ rmdir -p test1/test2/test3/test4
user@ubuntu: ~$ ll
```

⚙说明

如果想要删除废弃过时的目录时，就使用 rmdir，该命令从一个目录中删除一个或多个子目录项。需要特别注意的是，一个目录被删除之前必须是空的。不过，你也可以尝试以-p 的参数加入来递归删除上层的目录。删除某目录时用户也必须具有对父目录的写权限。

10.4 链接文件的介绍

在开始介绍链接文件（Link）之前，我们得先来了解一下什么是 inode。说实在的，这个概念真的很重要，不了解他的时候，很容易搞错很多东西。

10.4.1 inode 基础

inode 译成中文就是索引节点。每个存储设备或存储设备的分区（存储设备是硬盘、软盘、U盘等）被格式化为文件系统后，应该有两部分，一部分是 inode，另一部分是 Block。Block 是用来存储数据用的；而 inode，就是用来存储这些数据的信息，包括文件大小、属主、归属的用户组、读写权限等。inode 为每个文件进行信息索引，所以就有了 inode 的数值。操作系统能根据命令通过 inode 值最快地找到相对应的文件。

比如一本书，存储设备或分区就相当于这本书，Block 相当于书中的每一页，inode 就相当于这本书前面的目录，一本书有很多内容，如果想查找某部分的内容，可以先查目录，通过目录能最快地找到想要看的内容。虽然不太恰当，但还是比较形象。

当用 ls 查看某个目录或文件时，如果加上-i 参数，就可以看到 inode 节点了。比如前面所说的例子。

```
user@ubuntu: ~$ ls -li lsfile.sh
2408949 -rwxr-xr-x 1 user root 7 2008-06-13 22:11 lsfile.sh
```

lsfile.sh 的 inode 值是 2 408 949 ； 查看一个文件或目录的 inode，要通过 ls 命令-i 参数。在 Ubuntu Linux 文件系统中，inode 值相同的文件是硬链接文件，也就是说，不同的文件名，inode 可能是相同的，一个 inode 值可以对应多个文件。理解链接文件并不难，看看例子就会了。在 Linux 中，链接文件是通过 ln 工具来创建的。

10.4.2 ln 指令

用 ln 创建文件硬链接

⚙语法

```
user@ubuntu: ~$ ln [源文件] [目标文件]
```

⚙例子

下面举一个例子，在这个例子中，要为 sun.txt 创建硬链接 sun002.txt，然后看一下 sun.txt 和 sun002.txt 的属性的变化。

```
user@ubuntu: ~$ ls -li sun.txt <==查看 sun.txt 的属性
2408263 -rw-r--r-- 1  root root 29 04-22 21:02 sun.txt <==这是 sun.txt 的属性
user@ubuntu:~$ln  sun.txt sun02.txt  <==通过 ln 来创建 sun.txt 硬链接文件 sun02.txt
user@ubuntu: ~$ ls -li sun* <==查看一下 sun.txt 和 sun002.txt 的属性
2408263 -rw-r--r-- 2 root root 29 04-22 21:02 sun002.txt
2408263 -rw-r--r-- 2 root root 29 04-22 21:02 sun.txt
```

⚙说明

可以看到，sun.txt 在没有创建硬链接文件 sun002.txt 的时候，其链接个数是 1（也就是-rw-r--r--后的那个数值），创建了硬链接 sun002.txt 创建后，这个值变成了 2。也就是说，每次为 sun.txt 创建一个新的硬链接文件后，其硬链接个数都会增加 1。

inode 值相同的文件，它们的关系是互为硬链接的关系。当修改其中一个文件的内容时，互为硬链接的文件的内容也会跟着变化。如果删除互为硬链接关系的某个文件时，其他的文件并不受影响。比如把 sun.txt 删除后，还是能看到 sun002.txt 的内容，并且 sun02.txt 仍是存在的。

可以这么理解，互为硬链接关系的文件，它们好像是克隆体，它们的属性几乎是完全一样；

下面的例子把 sun.txt 删除，然后看一下 sun002.txt 是不是能看到其内容。

```
user@ubuntu: ~$ rm -rf sun.txt
user@ubuntu: ~$ more sun002.txt
```

注意 不能为目录创建硬链接，只有文件才能创建硬链接。

创建软链接（也被称为符号链接）

语法

```
user@ubuntu: ~$ ln  -s [源文件或目录]  [目标文件或目录]
```

软链接也叫符号链接，它和硬链接有所不同，软链接文件只是其源文件的一个标记。当删除了源文件后，链接文件不能独立存在，虽然仍保留文件名，但却不能查看软链接文件的内容了。

例子

```
user@ubuntu: ~$ ls -li sun001.txt
2408274 -rw-r--r-- 1 root root 29 04-22 21:53 sun001.txt
user@ubuntu: ~$ ln -s sun001.txt sun002.txt
user@ubuntu: ~$ ls -li sun001.txt sun002.txt
2408274 -rw-r--r-- 1 root root 29 04-22 21:53 sun001.txt
2408795 lrwxrwxrwx 1 root root 15 04-22 21:54 sun002.txt -> sun001.txt
```

说明

针对上面的例子，首先查看 sun001.txt 的属性，比如 inode 、所属文件种类、创建或修改时间等，我们来对比一下：

首先，对比一下节点，两个文件的节点不同。

其次，两个文件归属的种类不同，sun001.txt 是-，也就是普通文件，而 sun002.txt 是 l，它是一个链接文件。

第三，两个文件的读写权限不同，sun001.txt 是 rw-r--r--，而 sun002.txt 的读写权限是 rwxrwxrwx。

第四，两者的硬链接个数相同，都是 1。

第五，两文件的属主和所归属的用户组相同。

第六，修改（或访问、创建）时间不同。

另外，sun002.txt 后面有一个标记 ->，这表示 sun002.txt 是 sun001.txt 的软链接文件。

值得注意的是，当修改链接文件的内容时，就意味着在修改源文件的内容。当然，源文件的属性也会发生改变，链接文件的属性并不会发生变化。当把源文件删除后，链接文件只存在一个文件名，因为失去了源文件，所以软链接文件也就不存在了。这一点和硬链接是不同的。

```
user@ubuntu: ~$ rm -rf sun001.txt    <==删除 sun001.txt
```

```
user@ubuntu: ~$ ls -li sun002.txt    <==查看 sun002 的属性
2408795 lrwxrwxrwx 1 root root 15 04-22 21:54 sun002.txt -> sun001.txt
user@ubuntu: ~$ more sun002.txt    <==查看 sun002.txt 的内容
sun002.txt: 没有那个文件或目录          <==得到提示，sun002.txt 不存在
```

我们可以看到，软链接文件其实只是源文件的一个标记，当源文件失去时，它也就不存在了。软链接文件只是占用了 inode 来存储软链接文件属性等信息，但文件存储是指向源文件的。软链接可以适用于文件或目录。无论是软链接还是硬链接，都可以用 rm 来删除。rm 工具是通用的。

10.5 查看文件内容命令

前面我们提到的都只是对文件的最基本的操作，下面来谈谈如何查看文件内容，在 Linux 系统下，提供了不同方式的查看文件的命令，而最常使用的查看文件内容的命令可以说是 cat、more 和 less，还有一些命令不常用，但是有特殊的功能，下面让我们来学习一下。

- head：查看文件的头几行。
- tail：只看文件的最后几行。
- more：一页一页地显示文件内容。
- less：与 more 类似，但是比 more 更好的是，它可以往前翻页。
- cat：由第一行开始显示文件内容。
- tac：从最后一行开始显示，可以看出 tac 是 cat 的倒写。
- nl：显示的时候输出行号。
- od：以二进制的方式读取文件内容。

10.5.1 head 指令

从 head 的字面意思看，它的作用自然就是显示出一个文件的前几行，如果没有加上 -n 这个参数时，默认只显示 10 行。如果要显示前 20 行，那就输入“head -n 20 filename”即可。

语法

```
user@ubuntu: ~$ head [-n number] [文件名称]
```

参数说明

-n number：显示前 number 行。

例子

head 显示 .bashrc 文件

```
user@ubuntu: ~$ head ~/.bashrc  <==默认情况下，显示头 10 行
```

指定 20 行 head 显示 .bashrc 文件

```
user@ubuntu: ~$ head -n 20 ~/.bashrc    <==显示头 20 行
```

10.5.2 tail 指令

Tail 指令查看文件的最后几行。

语法

```
user@ubuntu: ~$ tail [-n number] [文件名称]
```

参数说明

- -n number：显示文件最后 number 行。
- -f：实时查看文件更新的内容。

例子

```
user@ubuntu: ~$ tail ~/.bashrc
user@ubuntu: ~$ tail -n 5 ~/.bashrc <==只显示最后 5 行
user@ubuntu: ~$ tail -f /var/log/messages <==实时查看文件更新的内容
```

说明

与 head 命令恰恰相反的是 tail 命令。使用 tail 命令，默认可以查看文件结尾的 10 行，或者通过参数-n number 来指定显示结尾的行数。这有助于查看日志文件的最后几行来阅读重要的系统消息，还可以使用 tail 来观察日志文件被更新的过程。使用参数 -f 选项，tail 会自动实时地把打开文件中的新内容显示到屏幕上。

10.5.3 more 指令

more 指令用于显示指定文件的文本内容，more 使用空格键和 PageUp 键来向前向后移动。

语法

```
user@ubuntu: ~$ more [文件名称]
```

参数说明

- -n：一次显示的行数。
- -d：提示用户，在画面下方显示 "Press space to continue，q to quit."，如果用户按错键，则会显示 "Press h for instructions." 。
- -l：取消遇见特殊字元 ^L 时会暂停的功能。
- -f：计算行数时，以实际的行数而非自动换行过后的行数（有些单行字数太长的会被扩展为两行或两行以上）。
- -p：不以滚动的方式显示每一页，而是先清除屏幕后再显示内容。
- -c：跟 -p 相似，不同的是先显示内容再清除其他旧资料。
- -s：当遇到连续两行以上的空白行，就替换为一行的空白行。

- -u：不显示下引号（根据环境变数 TERM 指定的 terminal 而有所不同）。
- +/：在每个文件显示前搜寻该字符串（pattern），然后从该字符串之后开始显示。
- +num：从第 num 行开始显示。

注意 退出 more 动作的指令是 q 键。

例子

```
user@ubuntu：~$ more ~/.bashrc <==一页一页地显示文件内容
user@ubuntu：~$ ls -al | more <==一页一页地将 ls 的内容显示出来
```

说明

more 是个很好用的指令，当文件太大的时候，那么使用 cat 将没有办法看清楚，这个时候可以使用 more 来查看了。more 也可以用来作为管线的同时执行之用，比如在执行 find 这个查找命令时，如果符合条件的文件很多，可以使用| more，则查询结果就可以分页地显示出来，然后就可以通过快捷键查看结果：ctrl+f（或空格键）显示下一页，ctrl+b 是返回上一页。

10.5.4 less 指令

less 指令用于显示指定文本文件内容，一般作为管道命令与 at 等一起使用。

语法

```
user@ubuntu：~$less [文件名称]
```

参数说明

- -c：从上到下刷新屏幕，并显示文件内容。
- -f：强制打开文件，二进制文件显示时，不提示警告。
- -i：搜索时忽略大小写，除非搜索字符串中包含大写字母。
- -I：搜索时忽略大小写，除非搜索字符串中包含小写字母。
- -m：显示读取文件的百分比。
- -M：显示读取文件的百分比、行号及总行数。
- -N：输出行号。
- -p：根据 pattern 搜索内容。
- -s：把连续多个空白行作为一个空白行显示。
- -Q：在终端下不响铃。

例子

```
user@ubuntu：~$ less ~/.bashrc
```

说明

less 的用法比起 more 有更多的灵活性，比如说在使用 more 的时候，并没有办法向前面翻，只能往后面看，但若使用了 less，就可以使用 PageUp、PageDown 等按键来往前、往后翻看文件。

10.5.5 cat 指令

cat 指令用于在屏幕上显示整个文件的内容，如文件较长，则快速翻屏显示。

语法

```
user@ubuntu: ~$ cat [-nAE] [文件名称]
```

参数说明

- -n：显示时，连行号显示在屏幕上。
- -A：将 DOS 下的<tab> 与断行字符都列出来。
- -E：在 DOS 编辑的文件中，仅列出断行字符。

例子

```
user@ubuntu: ~$ cat ~/.bashrc   <==显示 .bashrc 这个文件
# .bashrc
# User specific aliases and functions
PATH="/bin:/sbin:/usr/sbin:/usr/bin:/usr/local/sbin:/usr/local/bin:$PATH"
alias rm='rm -i'
alias cp='cp -i'
alias mv='mv -i'
alias ll='ls -l --color=never'

user@ubuntu: ~$ cat ~/.bashrc -n  <==显示 .bashrc 并且加上行号
   1  # .bashrc
   2
   3  # User specific aliases and functions
   4  PATH="/bin:/sbin:/usr/sbin:/usr/bin:/usr/local/sbin:/usr/local/bin:
      $PATH"
   5  alias rm='rm -i'
   6  alias cp='cp -i'
   7  alias mv='mv -i'
   8  alias ll='ls -l --color=never'
user@ubuntu: ~$ cat -A regexp.txt
This is a cat, however, I need a dog.^M$
I want to "Happy" and <Happy> and /Happy/ here.^M$
OK! ^Ieverythins is OK^M$
Now, I will eat my food^M$
are you ^Ifinished your work^M$
what do you 123 goto where^M$
<==显示出 DOS 文件的几个特殊符号，以上面文件为例，
可发现 ^M 为断行符号，而每行的$ 为行尾符号，^I 则是<tab>
```

```
user@ubuntu: ~$ cat -b test1.txt text2.txt >> text3.txt
<==把 text1 和 text2 的文件内容加上行号（空白行不加）之后将内容追加到 text3
```

说明

cat（Concatenate，连续），主要的功能是将一个文件的内容连续地输出在屏幕上面。例如上面的例子中，将重要的用户配置文件.bashrc 显示出来，如果加上-n 的话，则每一行前面还会加上行号。cat 比较少用，毕竟当文件内容的行数超过 40 行时，用户根本来不及看，所以，配合 more 或者是| more 来执行比较好。此外，如果是一般的 DOS 文件，就需要特别留意一些特殊的符号了，例如断行符号与<tab>等要显示出来，就得添加-A 之类的参数了。

10.5.6 tac 指令

tac 指令用于反向显示文件的内容。

语法

```
user@ubuntu: ~$ tac [文件名称]
```

参数说明

- -h：帮助信息。
- -v：版本信息。
- -b：用后面的替换前边的，一般建议不用。
- -r：根据正则表达式进行显示。

例子

```
user@ubuntu: ~$tac ~/.bashrc <==发现了没？反向输出呦！
fi
. /etc/bashrc
if [ -f /etc/bashrc ]; then
# Source global definitions
alias h='history'
alias lm='ls -al|more'
alias ll='ls -l'
# alias ll='ls -l --color=never'
alias mv='mv -i'
alias cp='cp -i'
alias rm='rm -i'
export PATH
PATH="/bin:/sbin:/usr/sbin:/usr/bin:/usr/local/sbin:/usr/local/bin:$PATH"
# User specific aliases and functions
# .bashrc
```

说明

如 cat 是显示文件内容的命令，tac 是 cat 的倒写，意思也和它是相反的。cat 是从第一行显示到最后一行，而 tac 是从最后一行显示到第一行，从最后一个字符显示到第一个字符。

10.5.7 nl 指令

nl 指令用于按行号显示文本内容，并显示行号。

语法

```
user@ubuntu: ~$ nl [文件名称]
```

参数说明

- -b：指定如何进行“行”的计算，可用值包括以下 4 个。
  - a:计算所有行。
  - t：不计算空行或包含任何非图形符号（如tab）的行（默认）。
  - n：不计算任何行。
  - pPattern：只计算那些符合Pattern指定的行。
- -d：指定逻辑页的分割符。
- -f：选择逻辑页页脚的行来计算。
- -h：选择逻辑页头的行来计算。
- -i：设定逻辑页行号增加数值，默认值为 1。
- -n：设定计算行的格式，可用值包括以下 3 个。
  - ln：左对齐，前导零不计。
  - rn：右对齐，前导零不计（默认）。
  - rz：右对齐，前导零保留。

例子

```
user@ubuntu: ~$ nl ~/.bashrc
```

说明

这个指令的用法跟 cat -n 的用法类似，也就是可以输出行号的指令来查看文件。

10.5.8 od 指令

语法

```
user@ubuntu: ~$ od [-At] [文件名称]
```

参数说明

- -A：指定地址基数，主要的参数有以下 4 个。

- d：十进制；
- o：八进制（系统默认值）；
- x：十六进制；
- n：不输出位移植。

- -t：指定数据的显示格式，主要的参数有以下6个。
  - c：ASCII字符或转义字符；
  - d：有符号十进制数；
  - f：浮点数；
  - o：八进制数（系统默认值为02）；
  - u：无符号十进制数；
  - x：十六进制数。

例子

```
user@ubuntu:~$ od ~/.bashrc
0000000 020043 061056 071541 071150 005143 021412 052440 062563
0000020 020162 070163 061545 063151 061551 060440 064554 071541
0000040 071545 060440 062156 063040 067165 072143 067551 071556
0000060 050012 052101 036510 027442 064542 035156 071457 064542
0000100 035156 072457 071163 071457 064542 035156 072457 071163
0000120 061057 067151 027472 071565 027562 067554 060543 027554
0000140 061163 067151 027472 071565 027562 067554 060543 027554
0000160 064542 035156 050044 052101 021110 062412 070170 071157
0000200 020164 040520 044124 005012 066141 060551 020163 066562
0000220 023475 066562 026440 023551 060412 064554 071541 061440
0000240 036560 061447 020160 064455 005047 066141 060551 020163
0000260 073155 023475 073155 026440 023551 021412 060440 064554
0000300 071541 066040 036554 066047 020163 066055 026440 061455
0000320 066157 071157 067075 073145 071145 005047 066141 060551
0000340 020163 066154 023475 071554 026440 023554 060412 064554
0000360 071541 066040 036555 066047 020163 060455 076154 067555
0000400 062562 005047 066141 060551 020163 036550 064047 071551
0000420 067564 074562 005047 021412 051440 072557 061562 020145
0000440 066147 061157 066141 062040 063145 067151 072151 067551
0000460 071556 064412 020146 020133 063055 027440 072145 027543
0000500 060542 064163 061562 056440 020073 064164 067145 004412
0000520 020056 062457 061564 061057 071541 071150 005143 064546
0000540 000012
0000541
```

说明

使用od命令可以查看特殊格式的文件内容。通过指定该命令的不同选项可以以十进制、八进制、十六进制和ASCII码来显示文件。默认情况下的显示方式是八进制，这也是该命令的名称由来（Octal Dump）。

10.6 课后练习

1. 请了解一下 Ubuntu 的文件系统的目录结构，并看看各目录主要存放的文件各有什么功能。
2. 什么是绝对路径，什么是相对路径？描述绝对路径和相对路径的差异。
3. Ubuntu 的图形化文件管理器有哪些？
4. 如何通过文件浏览器访问远程文件？
5. Ubuntu Shell 环境下，文件与目录操作的指令包括哪些？
6. 创建目录、删除目录、修改目录的命令分别是什么？
7. 如何通过指令进入一个目录，如何查看当前所在目录的路径？
8. 什么是链接文件？描述软\硬链接的差异。
9. 使用命令 tail、more、cat 来查看文件内容。

Chapter11 Ubuntu 文件的属性与权限

在 Ubuntu Linux 系统下，每一个文件和目录都有其属性与权限。属性所代表的意义是一个重要课题，它描述了文件的具体信息，以帮助用户了解文件或目录的基本情况，为下一步的操作做好准备。文件或目录权限则体现了系统对多用户、群组的操作控制功能，这对于多用户系统来说是相当有用的。通过权限管理，用户就不用担心自己的私有文件和目录被修改或删除，当然用户也可以授予文件或目录权限给其他指定用户。

11.1 Ubuntu 文件与目录属性

Ubuntu Linux 文件或目录属性主要包括：文件或目录的类型、大小、权限模式、徽标、默认打开方式、备忘、最近访问或修改的时间等内容。

11.1.1 图形化文件属性

打开文件浏览器，选中目标文件或目录，单击右键，选择菜单中的【属性】命令，可以查看该文件的属性信息，如图 11.1 所示。

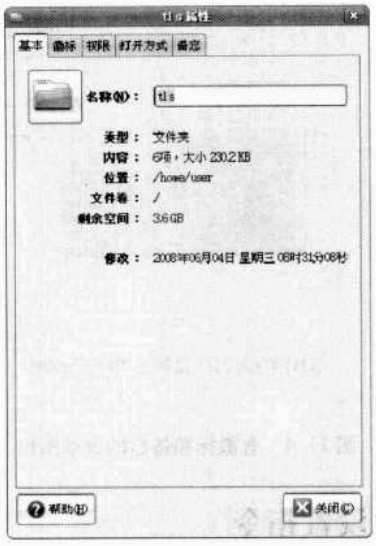


图 11.1 文件基本属性

从文件属性窗口中，还可以发现有【徽标】和【备忘】标签页，如图 11.2 和图 11.3 所示。实际上【徽标】和【备忘】是 Ubuntu 非常有趣的功能。

就像现在很多卖书的网站上贴标签的功能一样，给书的种类划分成不同的类型，用户也可以通

过贴【徽标】来标识文件或目录的种类和功用。而【备忘】更像一个小贴片一样的东西，如果需要给文件或目录写点什么东西的话，千万别忘了找【备忘】。

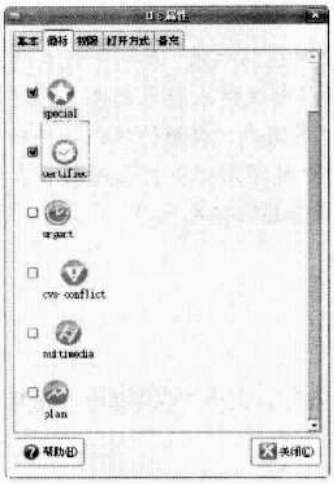


图 11.2 文件徽标

图 11.3 文件备忘

添加了徽标和备忘的目录跟普通的目录有着明显的区别：在目录右上角周围有和徽标备忘内容，如图 11.4 所示。

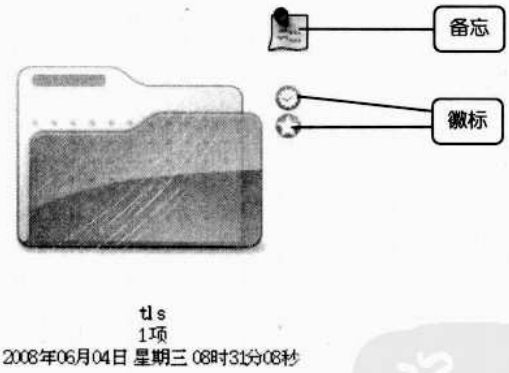


图 11.4 有徽标和备忘的目录图标

11.1.2 chattr 属性设置指令

chattr（change attribute）指令，用于改变文件属性，修改文件在 Linux 第二扩展文件系统（E2fs）上的特有属性，主要用户为超级管理员 root 用户。

语法

```
chattr [-RV] [-+=AacDdijsSu] [-v version]文件或目录
```

参数说明

- -R：递归处理，将指定目录下的所有文件及子目录一并处理。
- -v：<版本编号>设置文件或目录版本。
- -B：显示指令执行过程。
- +<属性>：开启文件或目录的该项属性。
- -<属性>：关闭文件或目录的该项属性。
- =<属性>：指定文件或目录的该项属性。

范例

```
user@ubuntuer: ~$ chattr +i /etc/shadow  <==无法改动这个文件
user@ubuntuer: ~$ chattr -i /etc/shadow <==去除该属性
```

说明

这个命令是很重要的，尤其是在系统的安全性方面，由于这些属性是隐藏的性质，所以需要使用 lsattr 才能看到该属性。ext2 文件系统上的文件或目录属性共有以下 8 种模式：

- a：让文件或目录仅供附加用途。
- b：不更新文件或目录的最后存取时间。
- c：将文件或目录压缩后存放。
- d：将文件或目录排除在备份（dump）操作之外。
- i：不得任意改动文件或目录。
- s：保密性删除文件或目录。
- S：即时更新文件或目录。
- u：预防意外删除。

其中，+i 这个属性，因为它可以让一个文件无法被改动，对于需要安全系数高的文件来说，真是非常的重要！而对于像日志这样的文件，就更需要 +a 这个设定：只可以增加不可以删除。

11.1.3 lsattr 属性显示指令

lsattr 指令用于显示文件属性。用 chattr 执行改变文件或目录的属性时，可先执行 lsattr 指令查询其属性。

语法

```
lsattr [-adlRvV][文件或目录...]
```

参数说明

- -a：显示所有文件和目录，包括名称以.为开头的额外内建，现行目录.与上层目录..。

- –d：显示目录名称，而非其内容。
- –l：此参数目前不起任何作用。
- –R：递归处理，将指定目录下的所有文件及子目录一并处理。
- –v：显示文件或目录版本。
- –V：显示版本信息。

范例

```
user@ubuntuer: ~$ chattr +i .bash_logout
user@ubuntuer: ~$ lsattr -a
-------------- ./.
-------------- ./..
---i---------- ./.bash_logout
-------------- ./.bash_profile
-------------- ./.bashrc
-------------- ./.emacs
-------------- ./.screenrc
```

11.2 文件与目录权限概述

Linux 系统中的每个文件和目录都有访问许可权限，用它来确定谁可以通过何种方式对文件和目录进行访问和操作。

文件或目录的访问权限分为可读、可写和可执行三种。以文件为例，可读权限表示只允许读其内容，而禁止对其做任何的更改操作。可执行权限表示允许将该文件作为一个程序执行。文件被创建时，文件所有者自动拥有对该文件的读、写和可执行权限，以便于对文件的阅读和修改。用户也可根据需要把访问权限设置为需要的任何组合。

有三种不同类型的用户可对文件或目录进行访问：文件所有者、同组用户和其他用户。所有者一般是文件的创建者。所有者可以允许同组用户有权访问文件，还可以将文件的访问权限赋予系统中的其他用户。在这种情况下，系统中每一位用户都能访问该用户拥有的文件或目录。

11.2.1 什么是文件权限

文件或目录的权限是和用户和用户组联系在一起的，所以要理解这部分内容，首先要了解一下 Linux 用户管理方面的知识。每个文件或目录都有一组 9 个权限位，每三位被分为一组，它们分别是属主权限位（占三个位置）、用户组权限位（占三个位置）和其他用户权限位（占三个位置）。比如 rwxr–xr–x，数一下就知道是不是 9 个位置了，正是这 9 个权限位来控制文件属主、用户组以及其他用户的权限，如图 11.5 所示。

其中：

属主权限位包括可读 r、可写 w、可执行 x；

属组权限位包括可读 r、可写 w、可执行 x；

图 11.5 文件权限位

其他用户权限位包括可读 r、可写 w、可执行 x；
如果权限位不可读、不可写、不可执行，则是用−来表示。

比如：

−rwx−−−−−−−：文件所有者对文件具有可读、可写和可执行的权限。

−rwxr−−r−−：文件所有者具有可读、可写与可执行的权限，其他用户则具有可读的权限。

−rw−rw−r−x：文件所有者与同组用户对文件具有可读、可写的权限，而其他用户仅具有可读和可执行的权限。

drwx−−x−−x：目录所有者具有可读、可写与进入目录的权限，其他用户只能进入该目录，却无法读取任何数据。

Drwx−−−−−−−：除了目录所有者具有完整的权限之外，其他用户对该目录没有任何权限。

每个用户都拥有自己的专属目录，通常集中放置在/home 目录下，这些专属目录的默认权限为 rwx−−−−−−−，表示目录所有者本身具有所有权限，其他用户无法进入该目录。执行 mkdir 命令所创建的目录，其默认权限为 rwxr-xr-x，用户可以根据需要修改目录的权限。

11.2.2 图形化文件权限

打开文件浏览器，选中目标文件或目录，单击右键，选择菜单中的【属性】命令，在弹出的属性窗口中选择【权限】标签页，如图 11.6 所示。

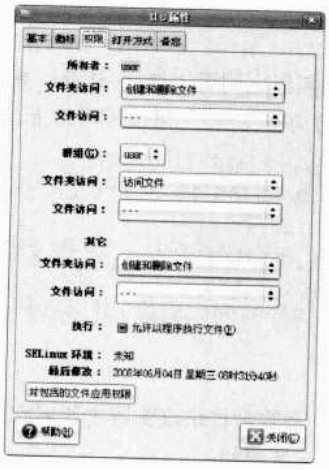


图 11.6 文件或目录权限

在窗口中可以通过改变下拉列表的选项修改文件或者目录的所有者、组群和其他用户的权限，而且可以设置子文件或目录的使用权限。

11.3 文件与目录权限设置

在 Linux 中，目录或文件拥有所有权限和操作（包括读、写、运行）权限，下面就来学习所有权限和操作权限的具体设置方法。

11.3.1 chown 所有者权限设置

当用户要改变一个文件的属主，所使用的用户必须是该文件的属主而且同时是目标属组成员，或超级用户 root。如果要连目录下的所有子目录或文件同时更改文件属主的话，直接加上-R 的参数，下面来看看语法与范例，假设系统本身有 test 用户和 test 群组。

语法格式

```
user@ubuntuer: ~$ chown [ -R ][用户名称][文件或目录]
user@ubuntuer: ~$ chown [ -R ][用户名称:组名称][文件或目录]
```

范例 1

将 test3.txt 文件的属主改为 test 用户。

```
user@ubuntuer: ~$ ls -l test3.txt
-rw-r--r-- 1 user root 0 2008-06-15 20:11 test3.txt

user@ubuntuer: ~$ chown test:root test3.txt
user@ubuntuer: ~$ ls -l test3.txt
-rw-r--r-- 1 test root 0 2008-06-15 20:11 test3.txt
```

范例 2

chown 所接的新的属主和新的属组之间可以使用:连接，属主和属组之一可以为空。如果属主为空，应该是":属组"；如果属组为空，":"可以不用带上。

```
user@ubuntuer: ~$ ls -l test3.txt
-rw-r--r-- 1 test root 0 2008-06-15 20:11 test3.txt

user@ubuntuer: ~$ chown :test test3.txt <==把文件 test3.txt 的属组改为 test
user@ubuntuer: ~$ ls -l test3.txt
-rw-r--r-- 1 test test 0 2008-06-15 20:11 test3.txt
```

范例 3

chown 也提供了-R 参数，这个参数对目录改变属主和属组极为有用，可以通过加-R 参数来改变某个目录下的所有文件到新的属主或属组。

```
user@ubuntuer: ~$ ls -l testdir <==查看 testdir 目录属性
```

```
drwxr-xr-x 2 user root 0 2008-06-15 20:00 testdir/
                                <==文件属主是 user 用户，属组是 root 用户

user@ubuntuer: ~$ ls -lr testdir <==查看 testdir 目录下所有文件及其属性
total 0
-rw-r--r-- 1 user root 0 2008-06-15 20:11 test1.txt
-rw-r--r-- 1 user root 0 2008-06-15 20:11 test2.txt
-rw-r--r-- 1 user root 0 2008-06-15 20:11 test3.txt


user@ubuntuer: ~$ chown -R test:test testdir/
                        <==修改 testdir 及它的下级目录和所有文件到新的用户和用户组

user@ubuntuer: ~$ ls -l testdir
drwxr-xr-x 2 test test 0 2008-06-15 20:00 testdir/

user@ubuntuer: ~$ ls -lr testdir
total 0
-rw-r--r-- 1 test test 0 2008-06-15 20:11 test1.txt
-rw-r--r-- 1 test test 0 2008-06-15 20:11 test2.txt
-rw-r--r-- 1 test test 0 2008-06-15 20:11 test3.txt
```

11.3.2 chgrp 用户组权限设置

chgrp（change group）是用来改变一个文件的群组的命令。要改变成为的群组名称必须是在系统中真实存在的名字才行，否则就会显示错误。

语法

```
user@ubuntuer: ~$ chgrp [群组名称][档案或目录]
```

它的用户和 chown 类似，只不过它仅是用来改变文件或目录的属组的；-R 参数用于目录及目录下所有文件改变属组的。这和 chown 也是一样的。下面举两个简单的例子。

范例 1

```
user@ubuntuer: ~$ ls -l test3.txt
-rw-r--r-- 1 user root 0 2008-06-15 20:11 test3.txt

user@ubuntuer: ~$ chgrp test test3.txt <==把文件 test3.txt 的属组改为 test
user@ubuntuer: ~$ ls -l test3.txt
-rw-r--r-- 1 test test 0 2008-06-15 20:11 test3.txt
```

范例 2

```
user@ubuntuer: ~$ ls -l testdir <==查看 testdir 目录属性
drwxr-xr-x 2 user root 0 2008-06-15 20:00 testdir/  <==文件属主是 user 用户，
                                                      属组是 root 用户
```

```
user@ubuntuer: ~$ ls -lr testdir <==查看 testdir 目录下所有文件及其属性
total 0
-rw-r--r-- 1 user root 0 2008-06-15 20:11 test1.txt
-rw-r--r-- 1 user root 0 2008-06-15 20:11 test2.txt
-rw-r--r-- 1 user root 0 2008-06-15 20:11 test3.txt


user@ubuntuer: ~$ chgrp -R test testdir/
                        <==修改 testdir 及它的下级目录和所有文件到 test 用户组

user@ubuntuer: ~$ ls -l testdir
drwxr-xr-x 2 user test 0 2008-06-15 20:00 testdir/

user@ubuntuer: ~$ ls -lr testdir
total 0
-rw-r--r-- 1 user test 0 2008-06-15 20:11 test1.txt
-rw-r--r-- 1 user test 0 2008-06-15 20:11 test2.txt
-rw-r--r-- 1 user test 0 2008-06-15 20:11 test3.txt
```

11.3.3 chmod 操作权设置

chmod 是用来改变文件或目录权限的命令，但只有文件的属主和超级权限用户 root 才有这种权限。通过 chmod 来改变文件或目录的权限有两种方法：一种是通过八进制的语法，另一种是通过助记语法。

助记语法设置文件权限

chmod 的助记语法相对简单，对文件或目录的权限改变时，是通过比较直观的字符形式来完成。在助记语法中，相关字母的定义如下。

⚙用户或用户组定义

- u：代表属主；
- g：代表属组；
- o：代表其他用户；
- a：代表属主、属组和其他用户，也就是上面三个用户（或组）的所有。

⚙权限定义

- r：代表读权限；
- w：代表写权限；
- x：代表执行权限。

⚙权限增减字符

- −：代表减去相关权限；
- +：代表增加相关权限。

⚙范例 1

```
user@ubuntuer: ~$ ls -l test1.txt
-rwxr-xr-x 1 user root 0 2008-06-15 20:11 test1.txt <==查看 test1.txt 的属性，
可以看到 test1.txt 的权限位是 rwxrwxrwx，用八进制数字表示是 777
user@ubuntuer: ~$ chmod ugo-x test1.txt <==把属主、用户组及其他用户的执行权限都减掉
user@ubuntuer: ~$ ls -l test1.txt
-rw-rw-rw- 1 user root 0 2008-06-15 20:11 test1.txt
```

⚙范例 2

```
user@ubuntuer: ~$ ls -l test1.txt
-rw-rw-rw- 1 user root 0 2008-06-15 20:11 test1.txt <==查看 test1.txt 的属性，
可以看到 test1.txt 的权限位是 rw-rw-rw-，用八进制数字表示是 666
user@ubuntuer: ~$ chmod u+x test1.txt  <==为文件的属主添加执行权限
user@ubuntuer: ~$ ls -l test1.txt
-rwxrw-rw- 1 user root 0 2008-06-15 20:11 test1.txt
```

⚙范例 3

```
user@ubuntuer: ~$ ls -l test1.txt
-rwxrw-rw- 1 user root 0 2008-06-15 20:11 test1.txt <==查看 test1.txt 的属性，
我们看到 test1.txt 的权限位是 rwxrw-rw-，用八进制数字表示是 766

user@ubuntuer: ~$ chmod u-x, go+x test1.txt  <==去掉文件属主对文件的执行权限，
                                                增加属组
和其他用户对文件的可执行权限

user@ubuntuer: ~$ ls -l test1.txt
-rw-rwxrwx 1 user root 0 2008-06-15 20:11 test1.txt
```

用助记语法比较灵活，组合起来比较方便，比如：

u=r+x 为文件属主添加读、写权限；
ug=rwx，o=r 为属主和属组添加读、写、执行权限，为其他用户设置读权限；
a+x 为文件的属主、属组和其他用户添加执行权限；
g=u 让文件的属组和属主和权限相同。

八进制表示法设置权限

在使用八进制语法设置权限前，读者必须了解一下八进制的含义，如表 11.1 所示。

表 11.1 八进制语法的数字说明：

单个权限	对应八进制数字
R	4
W	2
X	1
-	0

然后再把相应权限的数值相加，得到一个组的权限描述。因为一个文件包括三个权限组，所以对于一个文件的权限描述来说，八进制描述包括三个从 0 到 7 的数字，这三个数值分别对应属主、属组和其他用户的权限描述，比如文件的权限位 rwxr-xr-x，对应的八进制描述如下：

- 属主的权限用数字表达：属主的权限是 rwx，也就是 4+2+1，应该是 7；
- 属组的权限用数字表达：属组的权限是 r-x，也就是 4+0+1，应该是 5；
- 其他用户的权限数字表达：其他用户权限是 r-x，也就是 4+0+1，应该是 5。

所以整个文件的权限用八进制描述就是 755。

举例来说，如果要把文件 test1.txt 的权限完全开放，执行如下操作：

```
user@ubuntuer: ~$ ls -l test1.txt
-rwxr-xr-x 1 user root 0 2008-06-15 20:11 test1.txt <==查看 test1.txt 的属性，
我们看到 test1.txt 的权限位是 rwxr-xr-x，用八进制数字表示是 755

user@ubuntuer: ~$ chmod 777 sun.txt  <==我们改变它的权限为属主可读可写、属组可读、
                                        其他用
户可读，也就是 rwxrwxrwx，用数字表示就是 777

user@ubuntuer: ~$ ls -l test1.txt
-rwxrwxrwx 1 user root 0 2008-06-15 20:11 test1.txt
```

从上面的分析可以看出，每三位的权限代码（分别是属主、属组和其他用户）组合，有 8 种可能，如表 11.2 所示。

表 11.2 组权限与八进制数值的映射关系

八进制数值	三位组合权限
0	---
1	--x
2	-w-
3	-wx
4	r--
5	r-x
6	rw-
7	rwx

11.4 默认权限与 umask 设置

umask 是通过八进制数值来定义用户创建文件或目录的默认权限。umask 表示的是禁止权限。不过文件和目录有点不同。对于文件来说，umask 的设置是在假定文件拥有八进制 666 权限上进行，文件的权限就是 666 减去 umask 的掩码数值；对于目录来说，umask 的设置是在假定文件拥有八进制 777 权限上进行，目录八进制权限 777 减去 umask 的掩码数值。

范例

```
user@ubuntuer: ~$ umask 066

user@ubuntuer: ~$ mkdir testdir1
user@ubuntuer: ~$ ls -ld testdir1
drwx--x--x 2 user root 0 2008-06-15 22:00 testdir1/
```

说明

系统用户的主目录的权限是通过在配置文件中指定的，当创建用户时，假设用户目录 umask 的数值是 077。怎么理解这个 077 呢？当用户添加时，系统自动在/home 中创建用户目录，并且设置它的权限为 777–077=711，也就是 rwx––––––。

umask 一般都是放在用户相关 Shell 的配置文件中，比如用户目录下的.bashrc 或.profile，也可以放在全局性的用户配置文件中，还可以放在 Shell 全局的配置文件中，比如/etc/profile。umask 放在相关的配置文件中，目的是当管理员创建用户时，系统会自动为用户创建文件或目录时配置默认的权限代码。

11.5 课后练习

1. 文件属性包括哪些信息？
2. 如何在 Ubuntu 文件浏览器中查看某文件的属性？
3. 设置文件属性的指令是什么?使用这个指令需要注意什么？
4. 查看文件属性的指令是什么？试一试，通过这个指令可以查看到哪些信息？
5. 文件的权限包括哪些？在 Ubuntu 文件浏览器中，如何通过视窗查看修改文件权限？
6. Ubuntu Shell 中有哪两种文件权限的标识方法？
7. 可以通过什么命令进行文件的所有权限的设置？
8. 可以通过什么命令进行文件的读写执行权限的设置？

Chapter12 压缩命令与查找系统

这一章主要讲述系统日常维护中经常用到的两个功能：文件压缩和查找。文件压缩，顾名思义就是减少文件对磁盘的占用空间。在 Ubuntu Linux 中提供了多种样式的文件压缩命令，每种压缩方式都有它们各自的特点，这些将在下面的章节中详细讨论。而文件查找作用更显著，如果忘了文件存在什么地方或者几乎忘记了文件名称，不要着急，文件查找功能就是你的救星了。

12.1 压缩与归档

有时候，用户需要把一组文件存储成一个文件以便备份或传输到另一个目录甚至另一台计算机上。有时候，用户还需要把文件压缩成一个文件，因而它们仅使用少量磁盘空间并能更快地通过互联网下载。

理解归档文件（archive file）和压缩文件（compressed file）间的区别对用户来说十分重要。归档文件是一个文件和目录的集合，而这个集合被存储在一个文件中。归档文件没有经过压缩——它所使用的磁盘空间是其中所有文件和目录的总和。压缩文件也是一个文件和目录的集合，且这个集合也被存储在一个文件中，但是，它的存储方式使其所占用的磁盘空间比其中所有文件和目录的总和要少。如果用户在计算机上的磁盘空间不足，可以压缩不常使用的、或不再使用但想保留的文件，甚至可以创建归档文件，然后再将其压缩来节省磁盘空间。

12.2 图形化归档管理器

Ubuntu 提供了一个图形化归档压缩工具“归档管理器”。它可以压缩、解压文件和目录。归档管理器支持通用的 Unix 和 Linux 文件压缩和归档格式，而且它的界面和操作简单易用。它还被集成到桌面环境和图形化文件管理器中，使处理归档文件的工作更加简便易行。如果用户使用的是 Nautilus 文件管理器，可以双击想解除归档或解压的文件来启动归档管理器。归档管理器的浏览窗口就会出现，其中的文件夹里显示了要解压或解除归档的文件，以便用户抽取或浏览。如图 12.1 所示。

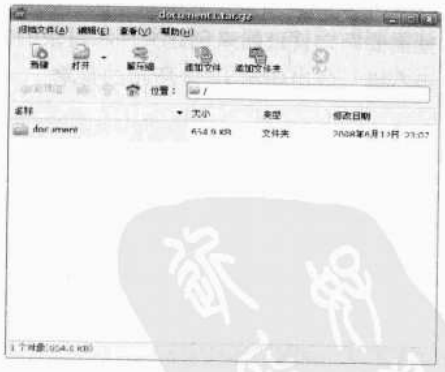


图 12.1 归档管理器

12.2.1 解压归档文件

用归档管理器解压归档文件有两种方式：一种是直接通过快捷菜单选项【解压到此处】来解压

归档文件到当前目录，另一种是使用归档管理器打开归档文件，然后解压到指定的目录。第一种方式很简单，也很容易操作，在这里将不作讨论，下面详细讨论第二种解压方法。

首先在文件浏览器中双击归档文件，系统将启动归档管理器并打开该文件，从归档管理器的菜单栏中选择【解压缩】，弹出如图 12.2 所示的窗口。

图 12.2 归档管理器解压缩

在上面的窗口中，选择解压缩文件的存储路径以及解压选项。需要注意的是，如果归档文件中刚好有文件与解压后存储路径下的文件重名，注意取消选中【覆盖已有文件】复选框，以免文件被解压后的文件覆盖，造成数据丢失。最后单击右下角【解压缩】按钮完成操作。

12.2.2 创建归档文件

如果需要腾出一些硬盘空间，或者把多个文件或某一目录下的所有文件发送给另一名用户，那就需要归档管理器来创建文件和目录的归档。要创建新归档，单击选中需要归档的文件或目录，单击右键，弹出如图 12.3 所示的菜单。

图 12.3 创建归档文件菜单

执行【创建归档文件】命令，弹出如图 12.4 所示的对话框，其中包含三个选项。

- 归档文件：也就是归档文件的名称；
- 归档文件类型：单击第一行右侧的微调框，可以看到归档类型有 tar.gz、tar.bz2、tar.lzma、tar、zip、ar、ear、jar、war 等；
- 位置：单击第二行右侧的微调框，弹出文件系统选项，可以自行定制归档路径。

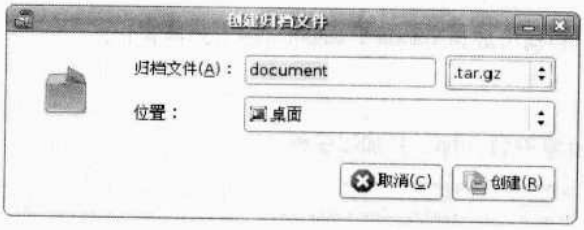


图 12.4 【创建归档文件】对话框

12.3 Shell 压缩指令

压缩文件使用较少磁盘空间，并且比未压缩的大文件下载速度要快。在 Ubuntu 中，可以使用的文件压缩工具有 gzip、bzip2 和 zip。这里推荐使用 bzip2 压缩工具，因为它可以最大限度地压缩，并且可在多数类似 Unix 的操作系统上找到。gzip 压缩工具也可以在类似 Unix 的操作系统上找到。如果你需要在 Linux 和其他操作系统如 Windows 间传输文件，你应该使用 zip，因为该命令与 Windows 上的压缩工具最兼容。如表 12.1 所示为压缩工具与扩展名的关系。

表 12.1 压缩工具与扩展名的关系

压缩工具	文件扩展名	解压工具
gzip	.gz	gunzip
bzip2	.bz2	bunzip2
zip	.zip	unzip

按照约定俗成，用 gzip 来压缩的文件的扩展名是.gz；用 bzip2 来压缩的文件的扩展名是.bz2；用 zip 压缩的文件的扩展名是.zip。

用 gzip 压缩的文件可以使用 gunzip 来解压；用 bzip2 压缩的文件可以使用 bunzip2 来解压；用 zip 压缩的文件可以使用 unzip 来解压。

12.3.1 compress 指令

compress 是个相当古老的 Unix 文件压缩指令，压缩后的文件会加上一个.Z 扩展名以区别于未压缩的归档文件，压缩后的文件能够以 uncompress 解压。若要将多个文件压成一个压缩文件，必须先将文件 tar 起来再压缩。

语法

```
user@ubuntuer: ~$ compress [-d] [文件名称 ...]
```

参数说明

- -d：将压缩文件解压缩。
- -r：连同目录下的文件一起压缩。
- -c：将压缩资料输出成为 standard output（输出到屏幕）。

范例 1

将/etc/man.config 复制到/tmp，并加以压缩。

```
user@ubuntuer: ~$ cd /tmp
user@ubuntuer: tmp$ cp /etc/man.config .
user@ubuntuer: tmp$ compress man.config
user@ubuntuer: tmp$ ls -l
 -rw-r--r-- 1 user root 2605 Jul 27 11:43 man.config.Z
```

范例 2

将刚刚的压缩文件解开。

```
user@ubuntuer: tmp$ compress -d man.config.Z
```

范例 3

将 man.config 压缩成另外一个文件来备份。

```
user@ubuntuer: tmp$ compress -c man.config > man.config.back.Z
user@ubuntuer: tmp$ ll man.config*
-rw-r--r-- 1 user root 4506 Jul 27 11:43 man.config
-rw-r--r-- 1 user root 2605 Jul 27 11:46 man.config.back.Z
```

说明

将 test.Z 解压成 test，如果 test 已存在，则系统会提示“该文件已存在，是否覆盖文件”，这时用户可以输入 y 以确定覆盖文件，n 将保留原有文件。如果使用-df 的参数，则解压缩过程中自动覆盖文件。除此之外，通过上面的例子可以知道，.Z 文件可以通过参数-d 或 uncompress 命令进行解压缩。

12.3.2 bzip2、bunzip2 和 bzcat 指令

bzip2 和 bunzip2 的功能基本相同，都是用来压缩或解压文件扩展名为 bz2 的命令，而 bzcat 命令则是用来读取数据而不需要解开。

语法

```
user@ubuntuer: ~$ bzip2 [-dz] filename  <==压缩解压缩文件
user@ubuntuer: ~$ bunzip2 [-dz] filename  <==压缩解压缩文件
user@ubuntuer: ~$ bzcat  filename.bz2  <==读取压缩文件内容
```

参数说明

- -d：将压缩文件解压缩。
- -z：压缩文件。

例子

```
user@ubuntuer: ~$ bzip2 -z test
user@ubuntuer: ~$ bzip2 -d test.bz2
user@ubuntuer: ~$ bunzip2 test.bz2
user@ubuntuer: ~$ bzcat test.bz2
```

12.3.3 gzip、gunzip 和 zcat 指令

gzip 和 gunzip 的功能基本上相同，都是用来压缩或解压文件扩展名为 gz 的命令，而用 zcat 命令来读取数据而不需要解开。

语法

```
user@ubuntuer: ~$ gzip [-d#] filename  <==压缩与解压缩
user@ubuntuer: ~$ gunzip [-d] filename  <==压缩与解压缩
user@ubuntuer: ~$ zcat filename.gz <==读取压缩文件内容
```

参数说明

- -d：将压缩文件解压缩。
- -#：设定压缩速度，0 表示不压缩，1 表示以最快速度压缩，9 表示以最慢速度压缩（最佳化的压缩），默认值为 6。

例子

```
user@ubuntuer: ~$ gzip test   <==会产生 test.gz 这个文件
user@ubuntuer: ~$ gzip -d test.gz <==解压缩，产生 test 这个文件
user@ubuntuer: ~$ gunzip test.gz <==解压缩，产生 test 这个文件
user@ubuntuer: ~$ gzip -9 test  <==以最大压缩比压缩 test 文件
user@ubuntuer: ~$ zcat test.gz  <==取出 test 的内容
```

12.3.4 .zip 和 unzip

zip 和 unzip 命令基本功能和参数类似，都可以用来进行文件的压缩和解压缩。

语法

```
user@ubuntuer: ~$ zip [filename] filename  <==压缩文件
user@ubuntuer: ~$  unzip filename.zip  <==解压缩文件
```

参数说明

- -b：暂存文件的路径。这个参数一般在要产生的 zip 文件存在而硬盘现有空间不足时使用。
- -c：替新增或更新的文件增加一行注解。
- -d：从 zip 文件移出一个文件。
- -D：不要在 zip 文件中存储文件的目录信息。
- -f：以新文件取代现有文件。
- -F：修复已经损毁的压缩文件。
- -g：将文件压缩附加到 zip 文件中。
- -h：显示辅助说明。
- -i：指定要包含的某些特定文件。
- -j：只存储文件的名称，不包含目录。
- -k：强迫使用 MSDOS 格式文件名。
- -l：将 CR（Carriage Return）LF（Line Feed）转换成 LF，一般要将 MS-DOS 上的文本文件压缩后拿到 Unix 下使用时才使用此参数。它只适用于文本文件（.txt），如果用于二进制文件则会造成二进制文件损毁。
- -L：显示 zip 命令的版权。
- -m：将特定文件移入 zip 文件中，并且删除特定文件。
- -n：不压缩特定扩展名的文件。
- -o：将 zip 文件的时间设成最后修正 zip 文件的时间。
- -q：安静模式，不会显示相关信息和提示。
- -r：包括子目录。
- -t：只处理 mmddyy 日期以后的文件。
- -T：测试 zip 文件是否正常。
- -u：只更新改变过的文件和新文件。
- -v：显示版本资讯或详细讯息。
- -x：不需要压缩的文件。
- -y：将 symbolic link 压缩，而不是压缩所连接到的文件。
- -z：为 zip 文件增加注解。
- -#：设定压缩速度，0 表示不压缩，1 表示以最快速度压缩，9 表示以最慢速度压缩（最佳化的压缩），默认值为 6。
- -@：从标准输入读取文件名称。

例子

```
user@ubuntuer: ~$zip test test  <==产生 test.zip 文件
user@ubuntuer: ~$ unzip test.zip <==解压缩 test.zip 文件
```

12.4 shell 归档指令

Ubuntu 和其他 Linux 一样，主要的处理归档文件（archive file）的指令有 tar 指令和 cpio 指令。

12.4.1 tar 指令

语法

```
user@ubuntuer: ~$ tar [-zxcvfpP] filename
user@ubuntuer: ~$ tar -N 'yyyy/mm/dd' /path -zcvf target.tar.gz source
```

参数说明

- -z：是否同时具有 gzip 的属性。
- -x：解开一个压缩文件的参数指令。
- -t：查看 tarfile 里面的文件。
- -c：建立一个压缩文件的参数指令。
- -v：压缩的过程中显示文件。
- -f：使用文件名。请留意，在 f 之后要紧接文件名，不要再加参数。例如，使用“tar -zcvfP tfile sfile”就是错误的写法，要写成“tar -zcvPf tfile sfile”才对。
- -p：使用原文件的原来属性（属性不会依据用户而改变）。
- -P：可以使用绝对路径
- -N：比后面接的日期（yyyy/mm/dd）还要新的才会被打包进新建的归档文件中。
- --exclude FILE：在压缩的过程中，不要将 FILE 打包。

范例 1

将整个/etc 目录下的文件全部打包成为/tmp/etc.tar。

```
user@ubuntuer: ~$$ tar -cvf /tmp/etc.tar /etc <==仅打包，不压缩
user@ubuntuer: ~$$ tar -zcvf /tmp/etc.tar.gz /etc <==打包后，以 gzip 压缩
user@ubuntuer: ~$$ tar -jcvf /tmp/etc.tar.bz2 /etc <==打包后，以 bzip2 压缩
```

注意 在参数 f 之后的文件是自己命名的，习惯上都用.tar 来作为标识。另外，如果加 z 参数，则以.tar.gz 或.tgz 来代表 gzip 压缩过的归档文件；如果加 j 参数，则以.tar.bz2 来作为文件后缀名。

在执行上面指令的时候，会显示一个警告信息：

```
tar:Removing leading '/' from member names
```

这是关于绝对路径的特殊设定。

范例 2

查阅上述 /tmp/etc.tar.gz 文件中有哪些文件。

```
user@ubuntuer: ~$$ tar -ztvf /tmp/etc.tar.gz
```

由于使用 gzip 压缩，所以要查阅该归档文件内的文件时，就需要加上 z 这个参数，这点需要特别注意。

范例 3

将/tmp/etc.tar.gz 文件解压缩在/usr/local/src 下。

```
user@ubuntuer: ~$ cd /usr/local/src
user@ubuntuer: src$$tar -zxvf /tmp/etc.tar.gz
```

在默认的情况下，可以将压缩文件在任何地方解开，以这个范例来说，先将工作目录变换到/usr/local/src 下，并且解开 /tmp/etc.tar.gz，则解开的目录会在/usr/local/src/etc 下。另外，如果进入/usr/local/src/etc 则会发现，该目录下的文件属性与 /etc/ 可能会有所不同。

范例 4

在/tmp 下，只将/tmp/etc.tar.gz 内的 etc/passwd 解压。

```
user@ubuntuer: ~$$ cd /tmp
user@ubuntuer: ~$$ tar -zxvf /tmp/etc.tar.gz etc/passwd
```

可以通过 tar -ztvf 来查看归档文件内的文件名称，如果只要一个文件，就可以通过这个方式来获取。需要注意 etc.tar.gz 内的根目录 /被去掉了。

范例 5

将/etc/内的所有文件备份下来，并且保存其权限。

```
user@ubuntuer: ~$$ tar -zxvpf /tmp/etc.tar.gz /etc
```

这个-p 的属性是很重要的，尤其是当要保留原本文件的属性时。

范例 6

备份 /home 和/etc，但不要 /home/dmtsai。

```
user@ubuntuer: ~$$ tar -exclude /home/dmtsai -zcvf myfile.tar.gz /home/* /etc
```

范例 7

将/etc/ 打包后直接解开在/tmp 下，而不产生归档文件。

```
user@ubuntuer: ~$$cd /tmp
user@ubuntuer: tmp$$tar -cvf - /etc | tar -xvf -
```

说明

tar 是一个多用途的归档压缩命令，刚刚提到的 compress 与 gzip 是可以适用在一个文件的压缩上面，但是如果是要压缩一个目录，tar 就派上用场了。tar 可以将整个目录或者是规定的文件或目录都打包成一个文件。例如上面的范例 1，它可以将/etc 目录的文件和子目录全部整合成一个文件。同时，tar 可以配合 gzip，同时进行整合并压缩，很是方便。因此，tar 在系统的日常维护中常用作备份操作。而由于 tar 打包过后的文件通常会取名为*.tar，而如果还含有 gzip 的压缩属性，那么就取名为*.tar.gz 。取这个文件扩展名只是为了方便记忆这个文件是什么属性罢了，并没有实际的

意义。

12.4.2 cpio 指令

语法

```
user@ubuntuer: ~$ cpio -covB  > [file|device] <==备份
user@ubuntuer: ~$ cpio -icduv < [file|device] <==还原
```

参数说明

- -o：将资料复制并输出到指定文件或设备上。
- -i：将数据自文件或设备复制到系统当中。
- -t：查看 cpio 建立的文件或设备的内容。
- -c：一种较新的 portable format 存储方式。
- -v：让存储过程中的文件名称可以在屏幕上显示。
- -B：让默认的 Blocks 可以增加至 5120B，默认值是 512B。这样的好处是可以让大文件的存储速度加快（请参考 i-nodes 相关内容）。
- -d：自动建立目录。由于 cpio 的内容可能不是在同一个目录内，如此恢复文件的过程会有问题。这个时候加上 -d 的话，就可以将需要的目录自动地建立起来了。
- -u：自动地将较新的文件覆盖较旧的文件。

例子

```
user@ubuntuer: ~$ find / -print | cpio -covB > /dev/st0
<==将查找到的文件存到磁带机
user@ubuntuer: ~$ cpio -icduv < /dev/st0
<==将磁带机的数据还原回来
user@ubuntuer: ~$ cpio -icdvt < /dev/st0 > /tmp/content
<==将磁带机的内容（文件名而已）转存到 /tmp/content
user@ubuntuer: ~$ find / -type -f | cpio -o > /tmp/root.cpio
user@ubuntuer: ~$ cpio -i < /tmp/root.cpio
<==上面这个例子可以马上换作看看！先输出到 /tmp/root.cpio 这个文件，然后再还原回来
```

说明

cpio 最适用于备份，因为它并不像 cp 一样，可以直接地将文件复制过去，例如 cp * /tmp 就可以将所在目录的所有文件复制到/tmp 目录下，在 cpio 这个命令的用法中，由于 cpio 无法直接读取文件，而是需要每一个文件或目录的路径连同文件名一起才可以被记录下来。因此，cpio 最常跟 find 这个命令一起使用了。从这方面看的话，cpio 好像不是很好用，但是，它可是备份的一项利器，因为它可以备份任何文件，包括/dev 下的任何设备。

cpio 有以下三种操作模式。

- 在 copy-out 模式中，cpio 把文件复制到归档包中。它从标准输入获得文件名列表（一行一个），把归档包写到标准输出。生成文件名列表的典型方法是使用 find 命令，需要注意的是，为了避免进入没有访问权限的目录而引起麻烦，要在 find 后面用上-depth 选项。

- 在 copy-in 模式中，cpio 从归档包里读取文件，或者列出归档包里的内容。它从标准输入读入归档包。任何不是选项的命令行参数都被视为 Shell 的通配符模式串 (globbing pattern)。在归档包中，只有文件名匹配这些模式串的文件才能复制出来。和 Shell 中不一样，文件名起始处的.可以匹配模式串起始处的通配符，文件名中的'/'也可以匹配通配符。如果没有给出模式串，那么将读出所有文件。
- 在 copy-pass 模式中，cpio 把文件从一棵目录树复制到另一目录棵，它结合了 copy-in 和 copy-out 的操作，但不使用归档包。cpio 从标准输入读取需要复制的文件名列表；目标目录作为非选项的命令行参数给出。

12.5 文件查找

每一种操作系统都是由成千上万个不同种类的文件所组成的。其中有系统本身自带的文件、用户自己的文件，还有共享文件等。用户有时候忘记某个文件或目录放在硬盘中哪个地方，这时候应该立马想到系统提供的查找功能，在 Ubuntu 下的查找功能跟 Windows 下的功能很相似，但是个人认为 Ubuntu 下的查找功能更加细化，提供更多的查询参数设置。另外，在 Shell 命令行下，可以使用专用的“查找”命令来寻找在硬盘上的文件。Linux 中查找文件的命令通常为 find 命令，find 命令是系统管理员日常维护系统的利器。而对于 Linux 新手来说，find 命令也是了解和学习 Linux 文件特点的方法。下面具体讨论文件查找。

12.5.1 文件搜索器

在桌面上执行【位置】|【搜索文件】命令即可进入 Ubuntu 的文件搜索器，如图 12.5 所示。

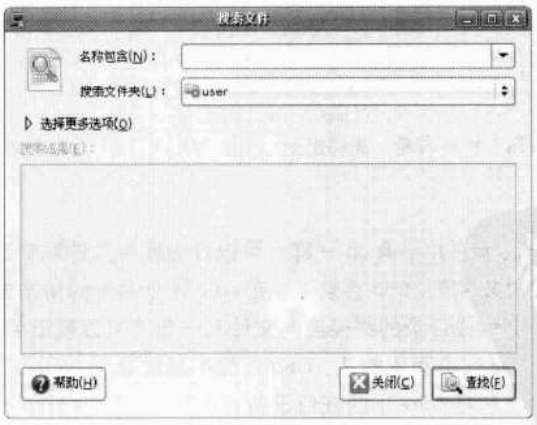


图 12.5 【搜索文件】窗口

窗口的最上端包含了搜索关键字，可以是*?等带有通配符的模糊查询，第二个选项是选择查询的根路径，也就是说查询将从这个目录递归查找，单击【选择更多选项】后，可以通过单击【添加】按钮添加更多的查询条件，如图 12.6 所示。

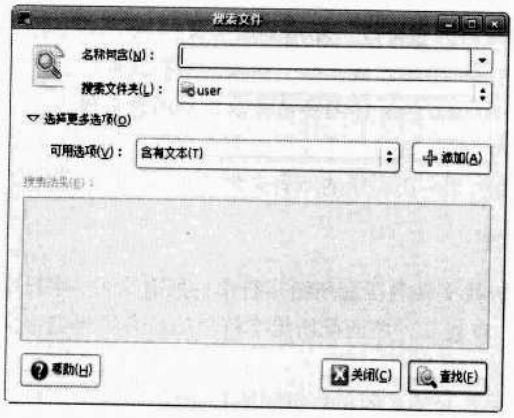


图 12.6　选择更多选项查询

其中支持的可用选项可以从下拉列表中选取，如图 12.7 所示。

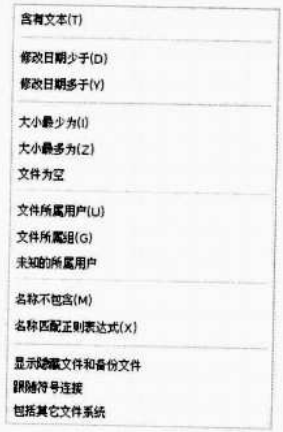


图 12.7　查询条件选项

12.5.2　find 指令

find 命令从指定的起始目录开始，递归地搜索其各个子目录，查找满足查找条件的文件并对其采取相关的操作。

语法

```
user@ubuntuer：~$ find [路径] [参数]
```

参数说明

时间

- -amin n：查找 n 分钟以内被访问过的所有文件。
- -atime n：查找 n*24 小时内被访问过的所有文件。

- –cmin n：查找 n 分钟内被修改、新增的所有文件。
- –ctime n：查找 n*24 小时内被修改、新增的所有文件。
- –mmin n：查找 n 分钟以内文件内容被修改过的所有文件。
- –mtime n：查找 n*24 小时内被修改过的所有文件。
- –newer file：查找比 file 还要新的所有文件。

⚙使用名称和文件属性

- –name '字符串'：查找文件名匹配所给字符串的所有文件，字符串内可用通配符有*、?和[ ]。
- –lname '字符串'：查找文件名匹配所给字符串的所有符号链接文件，字符串内可用通配符有*、?和[ ]。
- –gid n：查找属于 ID 号为 n 的用户组的所有文件。
- –uid n：查找属于 ID 号为 n 的用户的所有文件。
- –group '字符串'：查找属于用户组名为所给字符串的所有文件。
- –user '字符串'：查找属于用户名为所给字符串的所有文件。
- –empty：查找大小为 0 的目录或文件。
- –path '字符串'：查找路径名匹配所给字符串的所有文件，字符串内可用通配符有*、?、[]。
- –perm 权限：查找具有指定权限的文件和目录，权限的表示可以如 711，644。
- –size n[bckw]：查找指定文件大小的文件，n 后面的字符表示单位，默认为 b，代表 512 字节的块。
- –type x：查找类型为 x 的文件，x 为下列字符之一。
  - b：块设备文件；
  - c：字符设备文件；
  - d：目录文件；
  - p：命名管道（FIFO）；
  - f：普通文件；
  - l：符号链接文件（symbolic links）；
  - s：socket文件。
- –xtype x：与–type 基本相同，但只查找符号链接文件。

例子

```
user@ubuntuer: ~$find / -name test <==寻找名为 test 的文件或目录
user@ubuntuer: ~$find / -name 'test*' <==寻找包含 test 的文件或目录
user@ubuntuer: ~$find . -ctime 1   <==寻找目前目录下一天内新增的目录或文件
user@ubuntuer: ~$find /home/test -newer .bashrc
                    <==寻找/home/test 目录下比.bashrc 还要新的文件
user@ubuntuer: ~$find /home -user test <==寻找 /home 下属主为 test 的文件
user@ubuntuer: ~$find /dev -type b  <==寻找 /dev 下文件属性为 b 的文件
```

说明

find 可以根据不同的参数来给予文件不同的搜寻功能。例如，要寻找一个文件名为 httpd.conf 的文件，已知它在/etc 底下，那么就可以使用“find /etc –name httpd.conf”；如果仅记得有一个

文件名称包含了 httpd，但是不知道全名，这时就可以用通配符 *，如 "find /etc –name '*httpd*'" 就可将文件名中含有 httpd 的文件都列出来。除此之外，Find 命令还提供由逻辑运算符 not、and、or 组成的复合条件。

- and：逻辑与，在命令中用 "–a" 表示，是系统默认的选项，表示只有当所给的条件都满足时，才算满足寻找条件。例如$ find –name 'tmp' –xtype c –user 'inin'，该命令寻找三个给定条件都满足的所有文件。
- or：逻辑或，在命令中用 "–o" 表示。该运算符表示只要所给的条件中有一个满足时，寻找条件就算满足。例如$ find –name 'tmp' –o –name 'mina*'，该命令查询文件名为 tmp 或是匹配 mina*的所有文件。
- not：逻辑非，在命令中用！表示。该运算符表示查找不满足所给条件的文件。例如$ find ！–name 'tmp'。

12.6 文件定位

在系统日常使用过程中，我们常常需要知道哪个文件存放在什么位置，下面来谈一谈怎么进行文件或目录的定位。在 Ubuntu Linux 下有相当优异的文件定位系统，包括【最近的文档】，还有常用的一些 Shell 命令，但通常 find 不是很常用的，因为除了速度慢之外，它也很耗费硬盘资源，通常情况下我们都是先使用 whereis 或者是 locate 来检查，如果真的找不到才用 find 来查找。这是因为 whereis 与 locate 是利用数据库来查找数据，所以相当的快，而且实际并没有搜寻硬盘，因此比较省时间。

12.6.1 最近文档窗口

Ubuntu 跟 Windows 一样提供了一个【最近的文档】的查看功能，用户可以执行【位置】|【最近的文档】命令以查看最近看过或修改过的文件，如图 12.8 所示。

图 12.8 查看最近的文档

12.6.2 搜索和索引编制

通过上面的介绍，我们知道 whereis 和 locate 两个命令都是通过索引文件来查找文件的，那么这个索引文件的管理是怎么进行的呢？下面谈谈 Ubuntu 下索引文件的设置。在桌面上执行【系统】|【首选项】|【搜索和索引编制】命令，打开下面的索引属性设置，如图 12.9 所示。

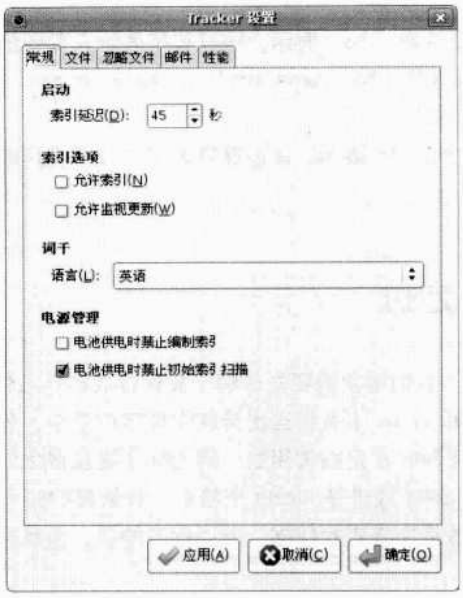


图 12.9 索引文件设置

通过这个设置窗口，用户可以定制索引文件的方方面面，包括指定索引文件的监视路径和非监视路径，排除指定的文件或文件类型，设定是否对邮件进行监视等。除此之外，为了提高搜索性能，还可以对索引文件的大小、内存占用等进行设置，以满足用户对系统部署的要求。

12.6.3 which 指令

Which 指令的基本功能是从系统变量 PATH 中指定的路径去寻找可执行文件，简单来说，其基本的功能在于查找可执行文件。

语法

```
user@ubuntuer: ~$ which  [文件名称]
```

参数说明

- -n<文件名长度>：指定文件名长度，所指定长度必须大于或等于查找文件的长度。
- -p<文件名长度>：与-n 类似，但长度包括路径长度。
- -w：指定输出栏的长度。

- -v：显示版本信息。

例子

```
user@ubuntuer: ~$ which passwd
```

12.6.4 whereis 指令

whereis 用于查找一个指定文件名的文件。

语法

```
user@ubuntuer: ~$ whereis [-bmsu] [目录名称]
```

参数说明

- -b：只找 binary 的文件。
- -m：只找说明文件 manual 路径下的文件。
- -s：只找 source 来源文件。
- -u：查找不寻常文件。

例子

```
user@ubuntuer: ~$ whereis passwd
passwd: /usr/bin/passwd /etc/passwd
<==将 passwd 相关字眼的文件或目录都列出来
user@ubuntuer: ~$whereis -b passwd
passwd: /usr/bin/passwd /etc/passwd
<==仅列出 binary 文件
user@ubuntuer: ~$whereis -m passwd
passwd: /usr/share/man/man1/passwd.1.bz2
<==仅查找 manual 路径下所在的目录
```

说明

如果使用 find 太麻烦，而且消耗很多时间，这个时候 whereis 就相当好用了！另外，whereis 可以加入参数来查找相关的资料，例如，如果是要找可执行文件（binary）那么加上参数 -b 就可以了，上面的范例就是针对 passwd 这个命令来说明；如果不加任何参数的话，那么就将所有的数据显示出来。那么 whereis 到底是使用什么机制呢？为何搜寻的速度会比 find 快这么多？这是因为 Linux 系统会将系统内的所有文件都记录在一个索引文件里面，而当使用 whereis 或者是下面将要说的 locate 时，都会以此索引文件的内容为准，因此，有的时候还会发现使用这两个命令时，会找到已经被删掉的文件。这就是因为文件被删除后，没有及时去更新这个索引文件。另外，Linux 基本上每天都会针对 Linux 主机进行 updatedb（就是更新索引文件）的动作，用户可以在/etc/cron.weekly/slocate.cron 这个文件中找到相关的机制，当然，也可以直接使用/usr/bin/updatedb 来手动更新索引文件。

12.6.5 locate 指令

locate 指令用于搜索指定文件，它比 find 速度快，但依赖于一个须每天更新的数据库。

语法

```
user@ubuntuer: ~$ locate [目录名称]
```

参数说明

- -u：建立数据库，由根目录开始。
- -U：建立数据库，于指定目录开始。
- -e：将该参数后文件排除。
- -q：安静模式，不提示任何错误。
- -n：最多显示几个。
- -r：使用正则表达式。
- -d：指定数据库路径。
- -f：将指定文件系统排除。

例子

```
user@ubuntuer: ~$locate root   <==相当多的符合条件的目录
user@ubuntuer: ~$updatedb <==立刻更新数据库
```

说明

locate 的使用方式很简单，其基本原理跟 whereis 命令类似，都是通过查询预先建立的索引文件来搜索指定文件的。

12.7 Ubuntu 全局搜索工具

google 和百度是众所周知的桌面搜索工具，同样，在 Ubuntu 系统下也提供了一个类似的工具——Deskbar-Applet，它可以搜索本地文件，搜索网络以及启动应用程序。要启用 Deskbar-Applet，右键单击桌面顶端面板，在弹出的快捷菜单中选择【添加到面板】命令，弹出下面的窗口，如图 12.10 所示。

从上面的窗口中，查找并选择【桌面栏】选项，然后单击【添加】按钮，这时候 Deskbar-Applet 程序将添加到顶端的面板上，就是这个小图标。怎么使用 Deskbar-Applet 呢？不要着急，在使用前，我们需要做个准备工作，即把相关的程序和允许启动的应用程序添加到 Deskbar-Applet。右键单击图标，在快捷菜单中选择【首选项】命令，弹出下面的窗口，如图 12.11 所示。

图 12.10　添加 Deskbar—Applet 到面板上

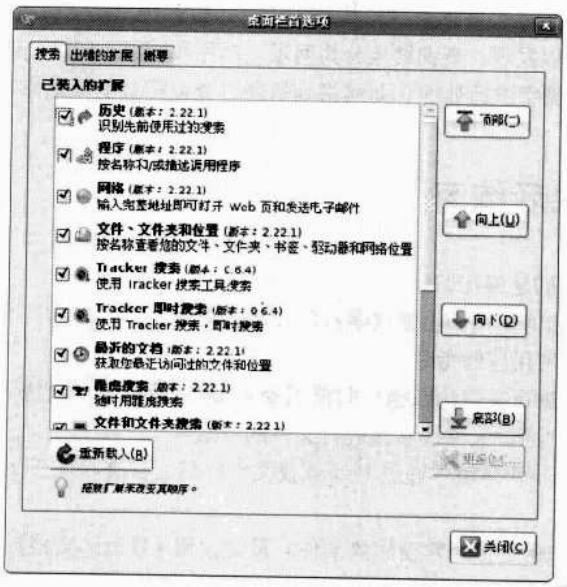


图 12.11　Deskbar—Applet 首选项

然后在窗口中勾选应用程序，最后单击【关闭】按钮。这时就可以使用 Deskbar—Applet 了，单击图标或者使用快捷键 Alt+F3，在文本框中输入查询关键字，Deskbar—Applet 自动显示可以执行的所有功能选择项，比如输入 find 就会看到如图 12.12 所示的查询结果。

注意　查询结果会依据 Deskbar—Applet 设置情况而有所不同。

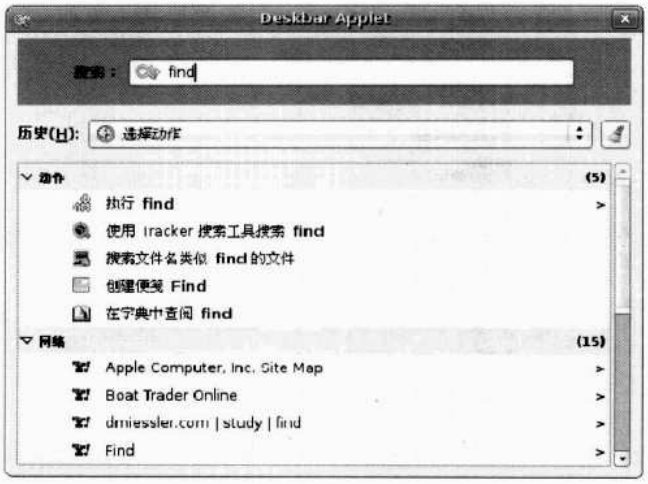


图 12.12　在 Deskbar—Applet 查找 find

在上面的窗口中可以发现，查询结果分为两项：动作和网络。所谓的动作，也就是本地系统启动相应的程序对 find 关键字进行处理，而网络则是通过查询相应的网站来获取 find 的相应信息。

12.8　课后练习

1．什么是 Lixux 下的压缩与归档？

2．在 Ubuntu 下，如何使用归档管理器打包和解压文件？

3．如何在 Shell 下使用压缩指令进行文件的压缩？

4．如何在 Shell 下进行文件的归档？归档指令有哪些？各有什么功能？

5．在 Ubuntu 中如何通过文件搜索器进行文件的搜索？

6．如何在 Ubuntu Shell 环境中通过 find 查找文件？请尝试查找大于 1MB 的文件，查找指定文件名的文件。

7．在 Ubuntu 中，如何定位经常使用的文件？请尝试用【最近的文档】窗口、which 指令、locate 等工具定位常用文件。

8．请尝试使用 Ubuntu 的全局搜索工具。

Chapter13 硬盘管理

作为一个 Ubuntu 系统管理员，你可能经常被狭小的硬盘空间困扰。新的应用程序要安装，不断增加的文件，这些都需要新的硬盘空间，那么如何新增硬盘来增加容量？怎么进行装载？新增一块硬盘的过程是怎样的？对系统的稳定性又有什么影响？这一章我们将讲述这些内容。

13.1 认识硬盘

硬盘是一种主要的电脑存储媒介，由一个或者多个铝制或者玻璃制的盘片组成。这些盘片外覆盖有铁磁性材料。绝大多数硬盘都是固定硬盘，被永久性地密封并固定在硬盘驱动器中。不过，现在可移动硬盘越来越普及，种类也越来越多。下面我们来了解一下硬盘的物理组成，再来说明一下怎么进行硬盘的分区。

物理组成

就硬盘的物理组件来说，硬盘其实是由许许多多的圆形硬盘盘片所组成的，依据硬盘能够容纳的数据量，就有所谓的单碟(一块硬盘里面只有一个硬盘盘片)或者是多碟(一块硬盘里面含有多个硬盘盘片)的硬盘，每个硬盘盘片上都有一个磁头（Head）进行该硬盘盘片的读写工作，当磁头固定不动，硬盘盘片旋转一周所走轨迹就是磁道（Track）。所有硬盘盘片上同一段磁道组成了磁柱（Cylinder）。一段磁道就是硬盘分区时的最小单位，即扇区（Sector），它是最小的硬盘存储物理单位，通常为 512Bytes。硬盘结构如图 13.1 所示。

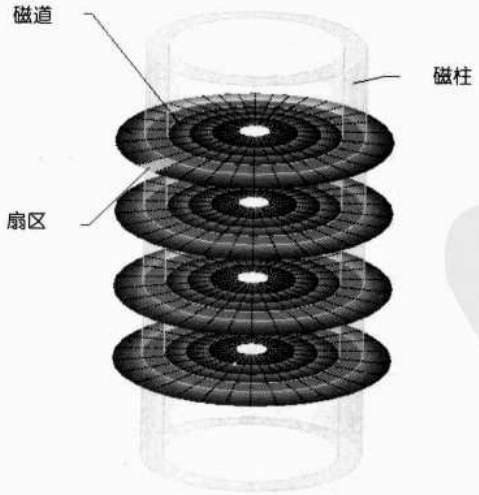


图 13.1 硬盘示意图

硬盘分区（Partition）

在了解了硬盘的物理组件之后，接下来介绍的是硬盘的分区（Partition）。为什么要进行硬盘分区？因为我们必须要告诉操作系统："我这块硬盘可以存取的区域是由 A 磁柱到 B 磁柱"。如此一来，操作系统才能够控制硬盘磁头去 A-B 范围内的磁柱存取数据。如果没有告诉操作系统这个信息，那么操作系统就无法利用硬盘来进行数据的读写了，因为操作系统将无法知道它要去哪里读取数据。这就是硬盘分区的重点，也就是记录每一个分区的起始与结束磁柱。那么分区的起始与结束磁柱的数据放在哪里呢？那就是开机扇区（Master Boot Recorder，MBR）。事实上，MBR 就是在一块硬盘的第零轨上面，这也是计算机开机之后要去读取的第一个区域。在这个区域内记录的就是硬盘里面的所有分割信息，以及开机的时候可以进行该机管理程序的写入的地方。所以，当一个硬盘的 MBR 坏掉时，由于分割的数据不见了，操作系统不知道该去哪个磁柱上读取数据，那么这个硬盘数据也就几乎可以说丢失了。

一块硬盘最多可分为四个主分区，或三个主分区+一个扩展分区（扩展分区可以包含多个逻辑分区）。Ubuntu 可以安装在主分区或逻辑分区。一般情况下，只需分/ 和 swap 两个区就足够了，不过根据作者个人经验还是建议把/home 独立分区，以后重装很多东西时就不必重新设置了。swap 是交换分区，如果内存<=256MB，请在分区时设置成 512MB，内存>=512MB 设置成 512MB 就足够了（如果要休眠，那么大于等于内存比较保险）。

文件系统（Filesystem）

在告诉系统分区所在的起始与结束磁柱之后，接下来需要将分区格式化，因为每个操作系统所识别的文件系统并不相同。例如，Windows 操作系统在默认状态下就无法识别 Linux 的标准文件系统，所以当然要针对操作系统来格式化分区。

不论是哪一种文件系统，总是需要存储数据。刚才提到硬盘的最小存储单位是一个 Sector，不过数据存储的最小单位并不是 Sector，因为用 Sector 来存储效率太低了。比如说，因为一个 Sector 只有 512Bytes，而磁头是一个一个 Sector 地进行读写，也就是说，如果文件大小有 100 MB，那么为了读这个文件，那么磁头必须要进行读取（I/O）204 800 次。

为了克服这个效率上的问题，所以就有逻辑区块（Block）的产生了。逻辑区块是在分区进行文件系统的格式化时指定的最小读写单位，这个最小读写单位当然是架构在 Sector 的大小上面，所以 Block 的大小为 Sector 的 2^n 倍数。此时，磁头一次可以读取一个 Block，如果假设在格式化的时候指定 Block 为 4 KB（亦即由连续的 8 个扇区（sector）构成一个 Block），那么同样一个 100 MB 的文件，磁头要读取的次数 uqf 大幅降为 25 600 次，这时文件的读取效率大大提高了。

不过，Block 单位的规划并不是越大越好，因为一个 Block 最多仅能容纳一个文件，而一个大小为 0.1KB 的小文件将占用掉一个 Block 的空间，也就是说，该 Block 虽然有 4 KB 的容量，然而由于文件只占用了 0.1 KB，所以，实际上剩下的 3.9 KBytes 是不能再被使用了，所以，在考虑 Block 的规划时，需要同时考虑到文件读取的效率及文件大小可能造成的硬盘空间的浪费。

当我们在进行硬盘分区时，每个硬盘分区就是一个文件系统（filesystem），而每个文件系统开始的位置的那个 Block 就称为 superblock，superblock 的作用是存储诸如文件系统的大小、空的和填满的块，以及它各自的总数和其他诸如此类的信息。也就是说，当要使用这一个硬盘分割槽（或者说是文件系统）来进行数据存取的时候，第一个要经过的就是 superblock 这个区块了，如果 superblock 坏了，那这个硬盘分区大概也就寿终正寝了。

inode 与 Block 来分别存储文件的属性（放在 inode 当中）与文件的内容（放置在 Block area 当中）。当我们要将一个 partition 格式化（format）为 ext2 时，就必须要指定 inode 与 Block 的大小才行，也就是说，当 partition 被格式化为 ext2 的文件系统时，一定会有 inode table 与 block area 这两个区域。

Block 已经在前面说过了，它是数据存储的最小单位。那么 inode 是什么？简单地说，Block 是记录文件内容的区域，至于 inode 则是记录文件相关属性，以及文件内容存放位置的信息。inode 除了记录文件的属性外，同时还必须要具有指向（pointer）的功能，亦即指向文件内容放置的区块，好让操作系统可以正确地去取得文件的内容。

文件系统格式

在 Microsoft Windows 的系统里，硬盘可以格式化成 NTFS、FAT32、FATI6 等不同的格式。同样地，在 GNU/Linux 下也有很多不同的文件系统格式可供选择。当前在 GNU/Linux 下，比较常用的有 Ext2 / Ext3、ReiserFS、XFS 和 JFS 等几种格式。当然各种格式都有其优缺点，所以我们将做一些简单的介绍。

除了 Ext2 以外，其他几种都是日志型文件系统。那什么是日志型文件系统呢?就是系统会多用一些额外的空间记录硬盘的数据状态，因而在不正常开关机后，不需重新扫描整个硬盘来恢复正常的系统状态。

（1）Ext2

此为一种非常老旧且不支持日志系统的文件系统格式，早期的 Linux 玩家应该还记得吧？在每次不正常关机后，重新开机时错误检查耗时很久，而且在不正常关机下，常常很多文件会消失了，现在已经很少人使用这类文件系统了。

（2）Ext3

Ext3 为 Ext2 的改良版，所以 Ext2 可以直接升级成为 Ext3 而不必重新格式化，这也可以让旧的 Ext2 系统更加稳定。和 Ext2 的主要差别是增加了日志系统(metadata)，所以在不正常开/关机后，可以迅速使系统恢复。因为它与旧有的文件系统兼容，因此很多发行版都默认使用 Ext3。但是在实际测试上，它的硬盘使用率不佳，大概只有真正空间的 93%会被使用到，至于其他性能测试表现则为中等。其格式化与创建文件系统的时间也是其他文件系统的数十倍。

（3）ReiserFS

采用日志型的文件系统，为 Hans Reiser 所创，因此以他的名字来命名。ReiserFS 技术上使用的是 B*—tree 为基础的文件系统，其特色为从处理大型文件到众多小文件都可以用很高的效率。实际上，ReiserFS 在处理文件小于 1KB 的小文件时，效率甚至可以比 Ext3 快约 10 倍，所以 ReiserFS 的专长是处理很多小文件。而在一般操作上，它的性能表现也有中上的程度。

（4）XFS

XFS 绘图工作站公司 SGI 为了高级绘图处理器系统 IRIX 所设计的文件格式，也是属于日志型文件系统，SGI 亦将其移植到 GNU/Linux 上。因为原本是针对高性能绘图设计，且为高阶工作站所使用，所以在稳定性和效率上是毋庸置疑的。论其在实务上的表现，它在处理各种文件大小混合的情况下效率最高，并且在一般使用上也有不错的表现。

（5）JFS

JFS 为全球最大的计算机供应商 IBM 为 AIX 系列设计的日志型文件系统，技术上使用的是 B+—tree 为基础的文件系统（和 ReiserFS 使用 B*—tree 不同）。IBM AIX 服务器在很多金融机构上使用，所以稳定性是没话说的。而它最重要的特色是所有文件系统里面在处理文件 I/O 的时候最不占 CPU 资源的，也就是 CPU 使用率最低。而且在这样节省使用 CPU 的情况下，它的效率表现还有中上以上的程度。

虽然 Ext3 性能不好(在日志型文件系统中效率上算是最糟糕的），那为何还有那么多人使用？那是因为当时 Ext3 可以直接从 Ext2 升级，而不需要先备份数据，然后格式化后再把文件复制回去，所以使用人数最多。但这也不能全然怪它，因为它为了和 Ext2 兼容，所以背负了很多的历史包袱。因此，这里推荐新的电脑考虑使用 ReiserFX、XFS 或 JFS。若是以性能为考虑因素，则可以选择 ReiserFS 或 XFS。若是系统资源不多，要使用最低的 CPU 使用率，那么可以选择 JFS，因为它有着最好的性能资源比。

以上就是硬盘的大致结构和划分，对于普通用户来说，上面的信息足可以让我们玩转 Ubuntu，如果读者有兴趣深入研究硬盘方面的技术，可以参考相关的专业硬盘书籍。

13.2 查看硬盘或目录的容量

在 Ubuntu 系统中，提供了一个很直观的硬盘空间查看工具（硬盘使用分析器），当然还有 Linux 系统下功能强大的命令，可以查看目前的硬盘最大容量、已经使用的容量、当前目录下使用的硬盘容量等，下面将讲述这方面的使用和操作。

13.2.1 硬盘使用分析器

在 Ubuntu 桌面上，执行【应用程序】|【附近】|【硬盘使用分析器】命令，启动应用程序，分析器可以扫描用户指定的目录、文件系统或远程目录，并且根据目录的树形关系，显示目录和硬盘空间对应关系的图形界面，如图 13.2 所示。

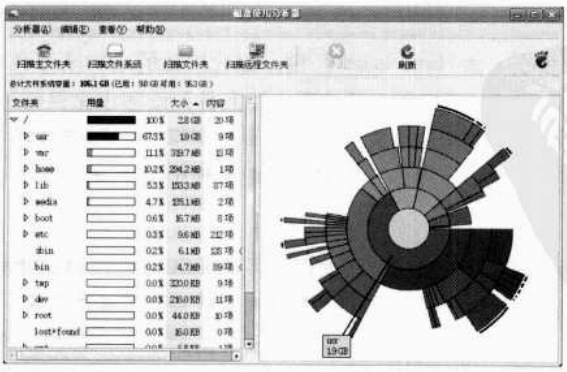


图 13.2 硬盘使用分析器

13.2.2 df 指令

在 Ubuntu 中，df 指令用来检查文件系统的磁盘空间占用情况，所有用户均有使用 df 命令的权限。可以利用该命令来获取硬盘被占用了多少空间，目前还剩下多少空间等信息。df 命令可显示所有文件系统对 i 节点和磁盘块的使用情况。

语法

```
root@ubuntuer: ~# df -[ikmh]
```

参数说明

- -a：显示所有文件系统的磁盘使用情况，包括 0 块（block）的文件系统，如/proc 文件系统。
- -k：以 k 字节为单位显示。
- -i：显示 i 节点信息，而不是磁盘块。
- -t：显示各指定类型的文件系统的磁盘空间使用情况。
- -x：列出不是某一指定类型文件系统的磁盘空间使用情况（与 t 选项相反）。
- -T：显示文件系统类型。

范例 1

以人易懂的方式显示当前磁盘占用情况。

```
root@ubuntuer: ~# df -h
Filesystem          Size  Used Avail Use% Mounted on
/dev/sda1           7.5G  3.5G  3.7G  49% /
varrun              252M  252K  252M   1% /var/run
varlock             252M     0  252M   0% /var/lock
udev                252M   48K  252M   1% /dev
devshm              252M   12K  252M   1% /dev/shm
lrm            252M   39M  213M  16% /lib/modules/2.6.24-19-generic/volatile
```

范例 2

列出各文件系统的 i 节点使用情况。

```
root@ubuntuer: ~# df -ia
Filesystem          Inodes   IUsed   IFree IUse% Mounted on
/dev/sda1           997472  155748  841724   16% /
proc                     0       0       0-  /proc
/sys                     0       0       0-  /sys
varrun               64447      76   64371    1% /var/run
varlock              64447       2   64445    1% /var/lock
udev                 64447    2853   61594    5% /dev
devshm               64447       2   64445    1% /dev/shm
devpts                   0       0       0-  /dev/pts
lrm       64447      19   64428    1% /lib/modules/2.6.24-19-generic/volatile
securityfs               0       0       0-  /sys/kernel/security
```

```
gvfs-fuse-daemon          0      0      0-  /home/user/.gvfs
```

范例 3

列出文件系统的类型。

```
root@ubuntuer: ~# df -T
Filesystem    Type   1K-blocks      Used Available Use% Mounted on
/dev/sda1     ext3     7850996   3633984   3818200  49% /
varrun       tmpfs      257788       252    257536   1% /var/run
varlock      tmpfs      257788         0    257788   0% /var/lock
udev         tmpfs      257788        48    257740   1% /dev
devshm       tmpfs      257788        12    257776   1% /dev/shm
lrm     mpfs   257788     39760   218028  16% /lib/modules/2.6.24-19-generic/
                                                volatile
```

说明

- Filesystem：所谓的硬盘扇区，另外，如果有加挂光驱的话，那么就会出现上述语句中倒数第三行。
- Used：使用的硬盘空间大小。
- Available：剩余的硬盘空间大小。
- Use%：硬盘的使用率，如果使用率高达 90% 以上时，最好需要注意一下，免得容量不足造成系统问题。
- Mounted on：硬盘挂载的所在目录，例如/dev/hda1 是挂载在根目录下，而光驱是 /dev/scd0 目录。

另外，需要注意的是，有的时候某些系统会出现 /proc 这个扇区，但是里面都是 0，/proc 下都是 Ubuntu 系统所需要加载的系统数据，而且是在内存中使用的，所以不会占用任何硬盘空间。

13.2.3 du 指令

du（disk usage）指令用来显示目录或文件的大小。

语法

```
root@ubuntuer: ~#du [-abcDhHklmsSx][-L <符号连接>][-X
<文件>][--block-size][--exclude=<目录或文件>][--max-depth=
<目录层数>][--help][--version][目录或文件]
```

参数说明

- -a 或-all：显示目录中个别文件的大小。
- -b 或-bytes：显示目录或文件大小时，以 Byte 为单位。
- -c 或—total：除了显示个别目录或文件的大小外，同时也显示所有目录或文件的总和。
- -D 或--dereference-args：显示指定符号连接的源文件大小。
- -h 或--human-readable：以 K、M、G 为单位，提高信息的可读性。
- -H 或--si：与-h 参数相同，但是 K、M、G 是以 1000 为换算单位。

- -k 或--kilobytes：以 1024 Bytes 为单位。
- -l 或--count-links：重复计算硬件连接的文件。
- -L<符号连接>或--dereference<符号连接>：显示选项中所指定符号连接的源文件大小。
- -m 或--megabytes：以 1MB 为单位。
- -s 或--summarize：仅显示总计。
- -S 或--separate-dirs：显示个别目录的大小时，并不含其子目录的大小。
- -x 或--one-file-xystem：以一开始处理时的文件系统为准，若遇上其他不同文件系统目录则略过。
- -X<文件>或--exclude-from=<文件>：在<文件>指定目录或文件。
- --exclude=<目录或文件>：略过指定的目录或文件。
- --max-depth=<目录层数>：超过指定层数的目录后，予以忽略。
- --help：显示帮助。
- --version：显示版本信息。

范例 1

列出/root 下的目录与文件所占的容量以 KB 输出。

```
root@ubuntuer: ~# du -k
12      ./.dbus/session-bus
16      ./.dbus
4       ./.gnome2_private
4       ./.scim/sys-tables
8       ./.scim
4       ./.gconfd
12      ./.mozilla/firefox/aiw8wwwh.default/chrome
4       ./.mozilla/firefox/aiw8wwwh.default/extensions
2248    ./.mozilla/firefox/aiw8wwwh.default
2256    ./.mozilla/firefox
4       ./.mozilla/extensions/{ec8030f7-c20a-464f-9b0e-13a3a9e97384}
8       ./.mozilla/extensions
2268    ./.mozilla
4       ./.gnome2/network-admin-locations
4       ./.gnome2/accels
32      ./.gnome2
8       ./.ssh
8       ./.gconf/apps/gedit-2/preferences/ui/statusbar
12      ./.gconf/apps/gedit-2/preferences/ui
16      ./.gconf/apps/gedit-2/preferences
20      ./.gconf/apps/gedit-2
24      ./.gconf/apps
8       ./.gconf/desktop/gnome/url-handlers/rtsp
12      ./.gconf/desktop/gnome/url-handlers
16      ./.gconf/desktop/gnome
20      ./.gconf/desktop
48      ./.gconf
80      ./.synaptic/log
92      ./.synaptic
```

```
4       ./.wapi
4       ./.aptitude
2516    .
```

可以使用-h 参数来显示 human-readable 的格式。在应用时，可以使用 du 这个指令来查看哪个目录占用最多的空间。不过，du 的输出结果通常很长，可以加上 -s 参数来省略指定目录下的子目录，而只显示该目录的总合即可。

范例 2

将/home 下的目录与文件的容量加总后输出。

```
root@ubuntuer: ~# du -s /home
1.7M /home
```

说明

在 Windows 下可以使用文件浏览器来管理硬盘，在 Ubuntu 下也可以轻易地用 du 指令来了解目前硬盘的文件容量。在默认的情况下，容量的输出是以 KB 来计的，如果想要知道目录占了多少 MB，那么就使用-m 这个参数，如果只想要知道该目录占了多少容量的话，使用-s 就可以了。

13.3 硬盘分割与格式化

对于一个系统管理者而言，硬盘的管理是相当重要的。尤其是硬盘已经渐渐地被当成消耗品，如果要分割一个新的硬盘，要使用什么程序来处理？而如果已经划分好分区，又要如何来格式化呢？前面提到了 inodes 这个概念，那么使用格式化(format)的软件可以怎样来格式化所需要的 inodes 的大小呢？以下是三种格式化分区工具。

- GParted（分区编辑器）：图形化的硬盘分割格式化工具。
- fdisk：硬盘切割的工具。
- mke2fs：Linux 下重要的格式化的工具。

13.3.1 GParted 分区工具

GParted 是一款功能非常强大的分区工具，和 Windows 下的“分区魔术师”类似，其操作和显示也很相似。GParted 可以方便地创建、删除分区，也可以调整分区的大小和移动分区的位置。GParted 支持多种 Linux 下常见的分区格式，包括 ext2、ext4、FAT、HFS、JFS、Reiser4、Reiserfs、XFS，甚至 NTFS。

安装 Gparted

Gparted 分区管理器在 Ubuntu 中不是默认安装的，使用之前，用户需要先进行安装。下面通过新立得软件管理器对 Gparted 进行安装。通过执行【系统】|【系统管理】|【新立得软件管理】命令打开新立得软件管理器，并选择 gparted，如图 13.3 所示。

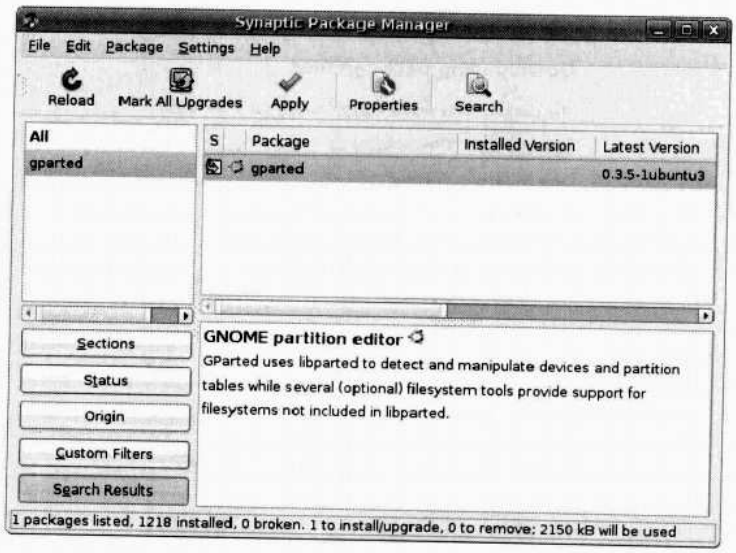


图 13.3 Synaptic 选择安装 Gparted

选好 Gparted 后，单击 Apply 按钮，弹出 Summary 对话框，该对话框提示即将要安装 Gparted，如图 13.4 所示。

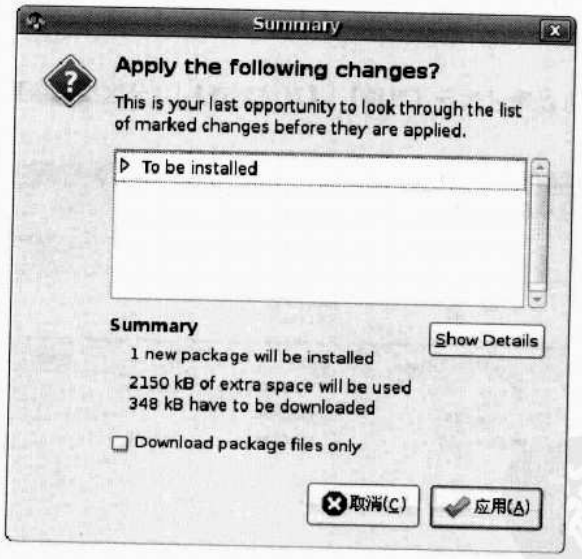


图 13.4 Summary 提示即将安装

单击【应用】按钮，即进入安装程序的下载并安装流程，这时弹出如图 13.5 所示的对话框，单击 Show progress of single files 可以查看当前安装进度信息。

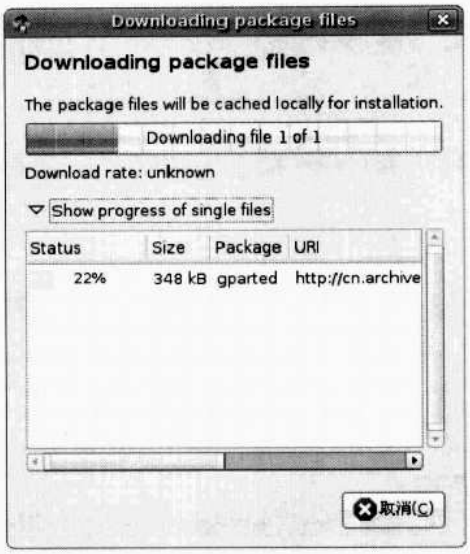


图 13.5 Gparted 安装进度提示

等待一段时间后，系统将自动下载并安装程序，安装成功后，弹出 Changes applied 对话框，提示安装成功，最后单击【关闭】按钮，这时就可以使用 Gparted 了。

使用 Gparted

要使用 Gparted，可以通过执行【系统】|【系统管理】|【分区编辑器】命令启动 Gparted 管理器，如图 13.6 所示。

图 13.6 Gparted 编辑器

在 Gparted 中，选中相应的硬盘及分区后，【新建】、【删除】、【更改大小】等按钮变为可

用状态，通过这些按钮可以方便地创建、删除分区，也可以调整分区的大小和移动分区的位置等。

13.3.2 fdisk 指令

fdisk 命令用来观察硬盘之实际使用情况与分割硬盘。fdsik 能将磁盘划分成为若干个区，同时也能为每个分区指定分区的文件系统，比如 Linux、FAT32、Linux Swap、FAT16 以及其类 Unix 操作系统的文件系统等。当然，用 fdisk 对磁盘进行分区并不是终点，我们还要对分区进行格式化所需要的文件系统。

语法

```
root@ubuntuer: ~# fdisk [-l] [设备名称]
```

参数说明

-l：直接列出该硬盘设备的分区表。

范例

```
root@ubuntuer: ~# fdisk /dev/hdb  <==进入分割硬盘的命令模式
Command (m for help): m <==显示所有命令列表
Command action
   a   toggle a bootable flag
   b   edit bsd disklabel
   c   toggle the dos compatibility flag
   d   delete a partition
   l   list known partition types
   m   print this menu
   n   add a new partition
   o   create a new empty DOS partition table
   p   print the partition table
   q   quit without saving changes
   s   create a new empty Sun disklabel
   t   change a partition's system id
   u   change display/entry units
   v   verify the partition table
   w   write table to disk and exit
   x   extra functionality (experts only)
Command (m for help): p <==显示硬盘分区状态
Disk /dev/hdb: 128 heads, 63 sectors, 523 cylinders
Units = cylinders of 8064 * 512 bytes
   Device Boot    Start       End    Blocks   Id  System
/dev/hdb1   *         1       250   1007968+  83  Linux
Command (m for help):q <==不存储离开 fdisk
root@ubuntuer: ~# fdisk -l /dev/hdb <==显示当前硬盘的使用状况
Disk /dev/hdb: 128 heads, 63 sectors, 523 cylinders
Units = cylinders of 8064 * 512 bytes
   Device Boot    Start       End    Blocks   Id  System
/dev/hdb1   *         1       250   1007968+  83  Linux
```

说明

Fdisk 的主要功能就是修改分区表，定义出某一个分区是从 n1 磁柱到 n2 磁柱之间这样的信息。因此，如果硬盘分区错误时，只要在格式化之前将分区表还原，那么就可以将硬盘原来的数据恢复回来。作为一个好的管理员，有时候也会将自己的分区表记录下来，以备不时之需。当然这个 fdisk 命令只有拥有 root 权限才能执行。此外，需要注意，设备名称上不要加上数字，因为分区是针对整个硬盘设备而不是某个分区，所以执行 fdisk /dev/sdb1 就会发生错误，要使用 fdisk /dev/sdb 才对，下面说一说进入 fdisk 之后的几个重要参数。

（1）硬盘信息

通常，需要知道这个硬盘的信息时，直接按 p 键就可以看见了，例如上面的例子中，一个硬盘分割情况如上，而系统中除了一个属于 swap 之外，其他的都属于 ext2。

（2）删除扇区

如果要删除一个已存在的扇区时，就需要按照以下步骤进行。

Step 01 输入fdisk /dev/hdb：先进入fdisk命令模式。

Step 02 输入p：先看一下分区，假如要删除 /dev/hdb1。

Step 03 输入d：这个时候会要用户选择一个分区，选1。

Step 04 输入w：存储到硬盘数据表中，并退出fdisk，如果用户反悔不删除了该怎么办？直接按q就可以取消刚才的删除操作。

```
root@ubuntuer: ~# fdisk /dev/hdb
Command (m for help): p
Disk /dev/hdb: 128 heads, 63 sectors, 523 cylinders
Units = cylinders of 8064 * 512 bytes
   Device Boot    Start       End    Blocks   Id  System
/dev/hdb1   *         1       250   1007968+  83  Linux
Command (m for help): d
Partition number (1-4): 1
Command (m for help): p
Disk /dev/hdb: 128 heads, 63 sectors, 523 cylinders
Units = cylinders of 8064 * 512 bytes
   Device Boot    Start       End    Blocks   Id  System
```

（3）新增扇区

如何增加一个扇区？前提是硬盘必须还有空余的硬盘空间。具体操作步骤如下。

Step 01 输入fdisk /dev/hdb：先进入fdisk命令模式。

Step 02 输入n：新增一个磁区，这个时候系统会进行询问。如果已经具有Extended 扇区时，那么系统会问，要新增的是Primary还是Logical，而如果用户还没有，那么系统仅会问要新增Primary还是Extended，除此之外，如果已经用完了4个P + E的话，那么就仅有Logical可以选择。如果是选择Primary的话，请按p，否则请按e或l。

Step 03 输入p：由于选择为Primary，所以按p。

Step 04 输入1–4：Primary只允许4个（仍然比 Windows只允许一个好多了！），所以这里请选择没有被使用的那一个扇区。

Step 05 输入w：存储并离开。

```
user@ubuntuer: ~$ fdisk /dev/hdb
Command (m for help): p <==输出信息
Disk /dev/hdb: 128 heads, 63 sectors, 523 cylinders
Units = cylinders of 8064 * 512 bytes
   Device Boot      Start         End      Blocks   Id  System
/dev/hdb1   *           1         250     1007968+  83  Linux
Command (m for help): n <==选择新增
Command action
   e   extended
   p   primary partition (1-4)
e  <==输入 e 来新增 extended
Partition number (1-4): 2
First cylinder (251-523, default 251):   <==这里按 Enter 即可！用默认值
Using default value 251
Last cylinder or +size or +sizeM or +sizeK (251-523, default 523): +100M
Command (m for help): p <==再输出信息
Disk /dev/hdb: 128 heads, 63 sectors, 523 cylinders
Units = cylinders of 8064 * 512 bytes
   Device Boot      Start         End      Blocks   Id  System
/dev/hdb1   *           1         250     1007968+  83  Linux
/dev/hdb2             251         276      104832    5  Extended  <==这行即是新增的
Command (m for help): n <==再次新增
Command action
   l   logical (5 or over)
   p   primary partition (1-4)
l <==这次选择 logical 的 l
First cylinder (251-276, default 251):   <==这里按 Enter 即可！用默认值
Using default value 251
Last cylinder or +size or +sizeM or +sizeK (251-276, default 276): +100M
Command (m for help): p
Disk /dev/hdb: 128 heads, 63 sectors, 523 cylinders
Units = cylinders of 8064 * 512 bytes
   Device Boot      Start         End      Blocks   Id  System
/dev/hdb1   *           1         250     1007968+  83  Linux
/dev/hdb2             251         276      104832    5  Extended
/dev/hdb5             251         276      104800+  83  Linux   <==这行即是新增的
```

通过上面的例子可以清楚地看到，第一个 logical 的 id 是 5，在 fdisk 完成之后，请记得使用 mke2fs 格式化。另外，需要注意，如果操作过程中发生错误，那么按下 q 退出即可。

（4）操作说明

以 root 的身份进行硬盘的分区时，最好是在单人维护模式下安全一些，此外，在进行 fdisk 的时候，如果该硬盘某个分区还在使用当中，那么很有可能系统核心会无法重新加载硬盘的分区表，解决的方法就是将该使用中的分区卸载，然后再重新进入 fdisk 操作一遍，重新写入分区表。

13.3.3 mke2fs 指令

mke2fs 指令用来建立 ext2 文件系统。

语法

```
root@ubuntuer: ~# mke2fs [-cFMqrSvV][-b <区块大小>][-f <不连续区段大小>][-i
<字节>][-N ][-l <文件>][-L <标签>][-m <百分比值>][-R=<区块数>][设备名称][区块数]
```

参数说明

- -b<区块大小>：指定区块大小，单位为字节。
- -c：检查是否有损坏的区块。
- -f<不连续区段大小>：指定不连续区段的大小，单位为字节。
- -F：不管指定的设备为何，强制执行 mke2fs。
- -i<字节>：指定“字节/inode”的比例。
- -N：指定要建立的 inode 数目。
- -l<文件>：从指定的文件中，读取文件系统损坏区块的信息。
- -L<标签>：设置文件系统的标签名称。
- -m<百分比值>：指定给管理员保留区块的比例，默认为 5%。
- -M：记录最后一次挂入的目录。
- -q：执行时不显示任何信息。
- -r：指定要建立的 ext2 文件系统版本。
- -R=<区块数>：设置磁盘阵列参数。
- -S：仅写入 superblock 与 group descriptors，而不更改 inode able inode bitmap 以及 block bitmap。
- -v：执行时显示详细信息。
- -V：显示版本信息。

范例

```
root@ubuntuer: ~# mke2fs /dev/hda5
<==以 mke2fs 默认情况下（ ext2）格式化 /dev/hda5 这个装置
root@ubuntuer: ~# mke2fs -c /dev/hda5
<==在格式化的过程中一起检查硬盘
root@ubuntuer: ~# mke2fs -j -b 8192 -i 8192 /dev/hda5
<==改变 block 由 4096 默认值改为 8192
root@ubuntuer: ~# mke2fs /dev/fd0       <==格式化软盘
mke2fs 1.26 (3-Feb-2002)
Filesystem label=
OS type: Linux
Block size=1024 (log=0)
Fragment size=1024 (log=0)
216 inodes, 1680 blocks
84 blocks (5.00%) reserved for the super user
First data block=1
1 block group
```

```
8192 blocks per group, 8192 fragments per group
216 inodes per group
Writing inode tables: done
Writing superblocks and filesystem accounting information: done
This filesystem will be automatically checked every 35 mounts or
180 days, whichever comes first.  Use tune2fs -c or -i to override.
```

说明

这是用来将硬盘格式化成 Ubuntu 系统文件的指令。基本上，只要写入对的装置文件就可以了。这个指令通常是在新的硬盘上面切割完之后，再加以格式化的。另外，如果要将旧的扇区格式化成 ext2 格式的话，也可以使用这个指令。格式化当中显示的信息有点像上面的最后几行，系统会显示目前的格式化的默认值，而如果要设定不同的 Block（就是前面提到的一个逻辑 sector 的大小），就可以使用-b 这个参数。需要注意的是，在默认情况下，Block 大小是 4096B。此外，您也可以自定义 inode table，当没有指定参数的时候，mke2fs 使用 ext2 为格式化文件格式，若加入-j 时，则格式化为 ext3 这个 Journaling 的文件系统。

13.4 检查硬盘坏轨与数据同步写入

建立了新的分区并格式化之后，有没有其他的关于硬盘的工作需要来做呢？有的，那就是需要来检查硬盘有没有坏轨，这也是 fsck 这个工具的用途所在，此外，在/这个目录下（其实有挂载硬盘的目录下都有这个目录）会有一个特殊的目录，即 lost+found 这个目录。当处理完 fsck 之后，如果程序发现有任何错误的文件，就会将该文件丢到这个目录当中，所以当发现 Ubuntu 目录当中有这个文件时，不要担心，这是正常的，而且只有挂载分区的目录才会有这个默认的目录。

还有，由于在 Linux 系统当中，为了增加系统效能，通常系统会默认数据写在内存当中，并不会直接将数据写入硬盘，这是因为内存的速度要比硬盘快上若干倍。但是有个问题需要考虑一下，万一系统由于断电或者是其他莫名的原因，造成系统的关闭，该怎么办？这时候，需要在某些特定的时候让数据直接写回到硬盘之中，这里提供几个惯用的指令。其中，fsck 是相当重要的，请参考其用法。

- fsck：检查硬盘有没有坏轨。
- badblocks：跟 fsck 一样，但是 fsck 的功能比较强，所以这个指令可以不学。
- sync：将内存中的数据同步化写入硬盘中。

13.4.1 fsck 指令

语法

```
root@ubuntuer: ~# fsck [-sACVRP] [-t fstype] [ - ] [fsck-options] filesys […]
```

参数说明

- filesys：device 名称(eg./dev/sda1)，mount 点 (eg. /或/usr) 。

- -t：指定文件系统的形式，若在/etc/fstab 中已有定义或 kernel 本身已支持的则不需加上此参数。
- -s：一个一个地依序执行 fsck 的指令来检查。
- -A：对/etc/fstab 中所有列出来的 partition 做检查。
- -C：显示完整的检查进度。
- -d：打印 e2fsck 的 debug 结果。
- -p：同时有-A 条件时，多个 fsck 的检查一起执行。
- -R：同时有-A 条件时，省略 / 不检查。
- -V：详细显示模式。
- -a：如果检查有错则自动修复。
- -r：如果检查有错则由用户回答是否修复。

范例

检查 msdos 文件系统的/dev/hda5 是否正常，如果有异常便自动修复。

```
root@ubuntuer: ~# fsck -t msdos -a /dev/hda5
```

说明

这个指令用来检查与修正硬盘错误的指令。通常只有 root 用户可以使用这个指令，而且系统有问题的时候才使用这个指令，否则在正常状况下使用此指令，可能会造成对文件系统的危害。通常，在系统出现极大的问题，而且开机的时候得进入单人单机模式进行维护时，才能使用该指令。另外，如果怀疑刚刚格式化成功的硬盘有问题的时候，也可以使用 fsck 来检查一下硬盘。其实这一点很像 Windows 的 scandisk。此外，由于 fsck 在扫描硬盘的时候，可能会造成部分文件系统的损坏，所以要执行 fsck 的时候，请将该分区卸载后再去执行。

13.4.2 badblocks 指令

Badblocks 命令用来检查磁盘装置中损坏的区块。执行指令时须指定所要检查的磁盘装置，及此装置的磁盘区块数。

语法

```
root@ubuntuer: ~# badblocks [-svw] [装置名称]
```

参数说明

- -b<区块大小>：指定磁盘的区块大小，单位为字节。
- -o<输出文件>：将检查的结果写入指定的输出文件。
- -s：在检查时显示进度。
- -v：执行时显示详细的信息。
- -w：在检查时，执行写入测试。
- [磁盘装置]：指定要检查的磁盘装置。

- [磁盘区块数]：指定磁盘装置的区块总数。
- [起始区块]：指定要从哪个区块开始检查。

范例

```
root@ubuntuer: ~# badblocks -sv /dev/hda1
```

说明

这是用来检查硬盘或软盘扇区有没有坏轨的指令，跟 Windows 的 scandisk 功能相同。不过，由于 fsck 的功能比较强，所以目前大多已经不使用这个指令了。

13.4.3 sync 指令

在 Linux 系统中，有的时候为了效率起见，欲写入硬盘的数据会写到文件系统的缓存中。这个缓存是一块记忆体空间，如果欲写入硬盘的数据存于此缓存中，而系统又突然断电的话，那么数据就会丢失了，sync 指令会将存于缓存中的数据强制写入硬盘中。sync 命令是在关闭 Linux 系统时使用的。sync 命令是强制把内存中的数据写回硬盘，以免数据的丢失。用户可以在需要的时候使用此命令。

语法

```
root@ubuntuer: ~# sync
```

参数说明

Comments：这个命令用法很简单，它只提供了两个 Linux 下公用的参数，包括命令的版本号和帮助说明。

- --help：显示命令的帮助。
- --version：显示该命令的版本号。

范例

```
root@ubuntuer: ~#sync
```

说明

在正常的状况中，为了提高系统的效率，因此，很多时候运行中的程序产生的临时文件都不会直接存至硬盘驱动器当中，而是存在内存当中。由于内存的数据传递速度比硬盘驱动器快了几十倍，所以如此一来将有助于提高整个系统的效率，然而这也产生了一个困扰，那就是当系统不正常关机的时候，可能会使得一些已经改变却还没有写入硬盘中的数据遗失（还在内存当中），所以这个时候 sync 的功能就发挥作用了，因为它可以直接将系统暂存在内存当中的数据写回硬盘当中。但是需要注意系统核心（kernel）必须要支持 sync 才行。

13.5 关于启动盘

以上工作完成之后，现在来制作一个可以开机的启动盘，怎么制作呢？当然就是利用mkbootdisk。mkbootdisk 用来制作软盘开机，fdformat 是用来低阶格式化软盘的工具。

13.5.1 mkbootdisk 指令

语法

```
root@ubuntuer: ~# mkbootdisk --device /dev/fd0 `uname -r`
```

参数说明

--device：后面接装置。通常接的就是软盘/dev/fd0。

范例

```
root@ubuntuer: ~# mkbootdisk --device /dev/fd0 `uname -r`
```

说明

这是制作启动盘的指令，其中，'uname -r'是目前 Ubuntu Linux 系统所使用的核心版本，如果有多个核心版本的话，可以直接输入核心版本。例如，某主机中装有两个核心 (kernel) 分别为 2.4.7 及 2.4.18，那么如果要直接以 2.4.18 来开机的话，就可以使用以下指令：

```
mkbootdisk --device /dev/fd0 2.4.18
```

建立启动盘一直是个好习惯，它可以在用户求助无门的时候给予莫大的帮助。

13.5.2 fdformat 指令

fdformat 是磁盘工具，所有用户都可以用。虽然由于 U 盘的出现，软盘几乎已经退出了历史的舞台，但是在 Linux 操作系统中依然有格式化软盘的命令，一个新买的软盘不经过低级格式化是不能使用的，所以还是有必要来了解一下如何格式化软盘。

语法

```
root@ubuntuer: ~# fdformat [设备名称]
```

参数说明

-n：关闭确认功能，会关闭格式化之后的确认提示功能。

范例

```
root@ubuntuer: ~# fdformat /dev/fd0
Double-sided, 80 tracks, 18 sec/track. Total capacity 1440 kB.
Formatting ... done
Verifying ... done
```

说明

在格式化软盘之前，如果软盘已经被挂载在某个挂载点下，需要使用 umount 命令来卸下软盘，否则会显示如下信息：

```
root@ubuntuer: ~# fdformat /dev/fd0
Double-sided, 80 tracks, 18 sec/track. Total capacity 1440 KB.
Formatting ...
ioctl(FDFMTTRK): Device or resource busy
```

Linux 下的 fdformat 和 Windows 下的格式化命令 format 的功能不一样，fdformat 只是格式化软盘，但并不建立文件系统类型，格式化后仍然不能使用，需要使用 mkfs 命令来建立相应的文件系统类型后才能正常使用。而 format 命令在完成格式化任务后还建立相应的文件系统（FAT）。

13.6 硬盘的装载

要将上面所建立起来的硬盘或软盘在 Ubuntu 上正式使用，还需要挂载文件系统。而所谓的挂载点则是该硬盘所在的目录，且该目录下的所有文件和目录都属于该硬盘所有。比如，根路径 /为/dev/hda1 而/home 为/dev/hda2，那么在/home/user 下的所有的文件目录就也都属于/dev/hda2 这个分区。需要特别留意的是，由于挂载文件系统需要挂载点，所以挂载的时候必须先建立起挂载的目录。除此之外，如果要用来挂载的目录里面并不是空的，那么挂载了文件系统之后，那么原目录下的东西就会暂时消失。举个例子来说，假设/home 原本是属于根目录/下的分区所有，其下原本就有/home/user 与/home/ubuntu 两个目录，现在想要加入新的硬盘，并且直接挂载在/home 下，那么当挂载上新的分区时，/home 显示的是该分区的内容，至于原先的 user 与 ubuntu 两个目录就会暂时被隐藏掉了。不过不要担心，并不是被覆盖掉，而是暂时隐藏了起来，等到分区卸载之后，该目录的内容就会重现出来。

13.6.1 mount 指令

mount 指令供系统管理者或/etc/fstab 中允许的用户使用。在 Linux 系统中，如果要使用硬盘、光盘、软盘或 MO 盘等存储设备，必须先进行挂载（Mount）。当完成存储设备挂载之后，就可以将其作为一个目录来进行访问了。挂载设备需要使用 mount 命令。执行这一命令，至少要先确定以下三种信息：

- 挂载（Mount）对象的文件系统类型；
- 挂载（Mount）对象的设备名称（/dev/????）；
- 要将设备挂载（Mount）到哪一目录。

语法

```
root@ubuntuer: ~# mount [-ahlV]
root@ubuntuer: ~# mount -t type /dev/hdxx /mountpoint
root@ubuntuer: ~# mount -o [options]
```

```
root@ubuntuer: ~# umount /mountpoint
```

参数说明

- –a：依照 /etc/fstab 的内容将所有相关的硬盘都装载上来。
- –h：只列出 mount 相关的参数，并不挂载任何装置。
- –l：列出目前已经挂载的硬件设备、文件系统名称与挂载点。
- –V：列出 mount 的版本信息。
- –F：这个命令通常和 –a 一起使用，它会为每一个 mount 的动作产生一个进程负责执行。在系统需要挂上大量 NFS 文件系统时可以加快挂载的速度。
- –f：通常用于除错。它会使 mount 不执行实际的动作，而是模拟整个挂上的过程。–f 通常会和–v 一起使用。
- –n：一般而言，mount 在挂上后会在/etc/mtab 中写入一些数据，但在系统中没有可写入档案系统存在的情况下可以用这个选项取消这个动作。
- type：将后面/dev/hdxx 这个装置以 type 的文件格式挂载到/mountpoint 这个点，常见的 type 有以下几个。
  - vfat，msdos：这个是支持 Windows 系统的文件格式，尤其是 vfat 常用。
  - ext，ext2：Linux 的主要文件格式。
  - iso9660：光驱的文件格式。
  - nfs，ntfs，ufs：Windows 2000 使用 NTFS 格式。
- –o async：打开异步模式，所有的文件读写操作都会用异步模式执行。
- –o sync：在同步模式下执行。
- –o atime/–o noatime：当 atime 打开时，系统会在每次读取文件时更新文件的“上一次调用时间”。当使用 Flash 文件系统时可能会把这个选项关闭以减少写入的次数。
- –o auto /–o noauto：打开/关闭自动挂上模式。
- –o defaults：使用默认的选项 rw、suid、dev、exec、auto、nouser 和 async。
- –o dev /–o nodev–o exec /–o noexec：允许执行可执行文件。
- –o suid /–o nosuid：允许在 root 权限下执行可执行文件。
- –o user /–o nouser：用户可以执行 mount/umount 的操作。
- –o remount：将一个已经挂下的文件系统重新用不同的方式挂上。例如原先是只读的系统，现在用可读写的模式重新挂上。
- –o ro：用只读模式挂上。
- –o rw：用可读写模式挂上。
- –o loop=：使用 loop 模式。

范例

```
root@ubuntuer: ~# mount -a
root@ubuntuer: ~# mount -t iso9660 /dev/cdrom /mnt/cdrom <==挂上光盘
root@ubuntuer: ~# mount -t vfat /dev/fd0 /mnt/floppy
                        <==挂上 Windows 文件系统的软盘
root@ubuntuer: ~# mount -t ext2 /dev/fd0 /mnt/floppy
                        <==挂上 Linux   文件系统的软盘
```

```
root@ubuntuer: ~# mount -t ext2 /dev/hdc6 /home
                        <==挂上 Linux 文件格式硬盘
root@ubuntuer: ~# mount -o remount, rw /          <==让根目录重新挂载为可读
```

说明

这个指令只有 root 权限才能执行。如果不想将某个单独的硬盘挂上来，那么执行：

```
mount -a
```

就可以依照 /etc/fstab 的参数内容将所有的硬盘重新挂上去。此外，需要注意的是，由于 Linux 系统中，每一个路径都有可能是一个独立的扇区系统，所以需要将每个扇区系统都挂上各自的挂载点。还需要注意的是，由于各个扇区的文件系统可能并不相同，所以必须先要了解该扇区的文件系统，这样才可以进行 mount 的工作。那么如何知道该硬盘的文件格式呢？可以使用 fdisk 显示的功能即可。

由于 mount 之后的文件格式是没有办法直接去掉的，尤其在使用 fsck 检查硬盘时，更不能挂上硬盘，这时需要使用 umount 命令来将硬盘从挂载点卸载。

安装新硬盘

如果想要安装一块新硬盘，这跟在 Windows 下操作一样，需要先 fdisk 然后再 format，之后就可以顺利地挂上 Linux 文件系统，假设安装的硬盘在 Primary 的第二个 IDE 上面，也就是 /dev/hdb 上，那么整个步骤如下所示。

```
root@ubuntuer: ~# fdisk /dev/hdb
.......（以下省略，直接以 fdisk 分割硬盘）
root@ubuntuer: ~# mke2fs /dev/hdb1
Linux 中的 format 是 mke2fs 这一个指令
<==上面的指令将硬盘扇区格式化成 Linux 的 ext2 格式
root@ubuntuer: ~# mkdir /disk2
<==建立一个名称为/disk2 的目录，用于挂载新硬盘
root@ubuntuer: ~# mount -t ext2 /dev/hdb1 /disk2
<==将硬盘挂上 Linux 系统
```

13.6.2 umount 指令

umount 就是直接将 mount 上来的文件系统卸载，卸载之后，可以使用 df 看看文件是否还存在。

语法

```
root@ubuntuer: ~# umount [-f] [device|mount_point]
```

参数说明

-f：强制将该文件系统退出，最常使用在无法退出的 NFS 文件系统中。

范例

```
root@ubuntuer: ~# umount /home
```

13.7 课后练习

1. 请尝试了解一下计算机硬盘的一些基础知识并回答：什么是磁盘分区？什么是文件系统？
2. 在 Ubuntu 中，如何通过可视化工具查看一个磁盘的使用情况？
3. 如何通过 Shell 命令了解整个磁盘的占用情况？了解当前目录的占用情况？
4. 在 Ubuntu 中如何进行磁盘的分区？有无可视化操作工具？如果有，如何操作？
5. 请尝试在 Ubuntu Shell 下使用 fdisk 指令、mke2fs 指令进行磁盘的分区操作。
6. 如何在 Ubuntu 中再挂载一个新硬盘？用什么指令，如何操作？

Chapter14 用户管理

Ubuntu Linux 为多用户操作系统，基于用户身份来控制他们对资源的访问，并允许将用户分组管理以简化访问控制以便为各用户分别设置权限。同时，Ubuntu 有完美的用户管理器，通过它可以安全、轻松地完成包括用户的添加、删除、属性修改，用户组的添加、删除、属性修改等用户管理功能，另外 Ubuntu 支持用户之间的相互切换。当然所有的用户管理功能都有相应的 Shell 命令对应，在 Shell 中也可以实现所有的用户管理。在 Ubuntu 中有/etc/shadow 和/etc/gshadow 文件，它们记录了经过加密的密码及相关用户信息，可以直接编辑这些文件来管理用户和组，但对于初级用户不建议这样操作。本章中我们将针对上面这些内容进行一一介绍。

14.1 用户的管理

Linux 支持多用户，其内核中有专门的用户管理模块。具体到 Ubuntu 套件，它提供了专门的图形化用户管理工具即用户管理器。通过用户管理工具，可以完成查看、修改、添加和删除本地用户和组群等。要从桌面启动用户管理器，执行【系统】|【系统管理】|【用户和组】命令，即弹出【用户设置】对话框，如图 14.1 所示。

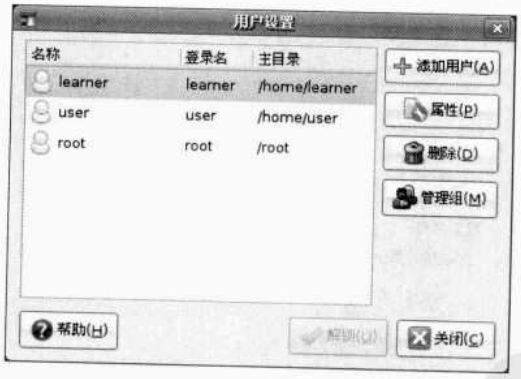


图 14.1 Ubuntu 用户设置

在【用户设置】主窗口的左侧区域，列出了当前所有的用户，默认列表界面中，用户信息包括【名称】、【登录名】和【主目录】；在主窗口的右侧区域则分别有【添加用户】、【属性】、【删除】和【管理组】几个功能按钮。

需要说明的是，Ubuntu 中用户及组的操作大部分需要管理员权限才能完成，所以当用户不具备这个权限时，相应的按钮为灰色状态，且在主窗口的右下侧有一个按钮【解锁】可用，单击它弹出窗口，提示用户选择管理员用户，并提示输入用户密码，如图 14.2 所示。

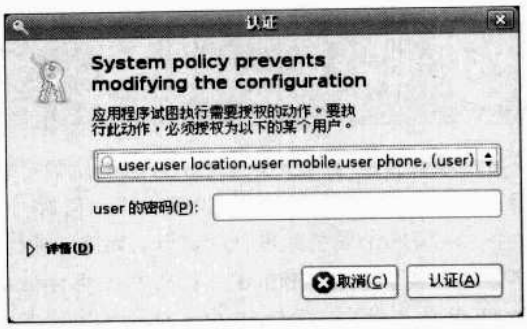


图 14.2 用户验证

输入相应用户对应的密码后，系统验证权限通过后，即可进行后面相应的用户管理操作了。下面分别讲述 Ubuntu 用户的添加、删除、修改属性等管理操作。

14.1.1 增加用户

窗口化添加用户

要添加新用户，在【用户设置】对话框中，单击右侧的【添加用户】按钮。这时即弹出【新建用户账户】对话框，如图 14.3 所示。

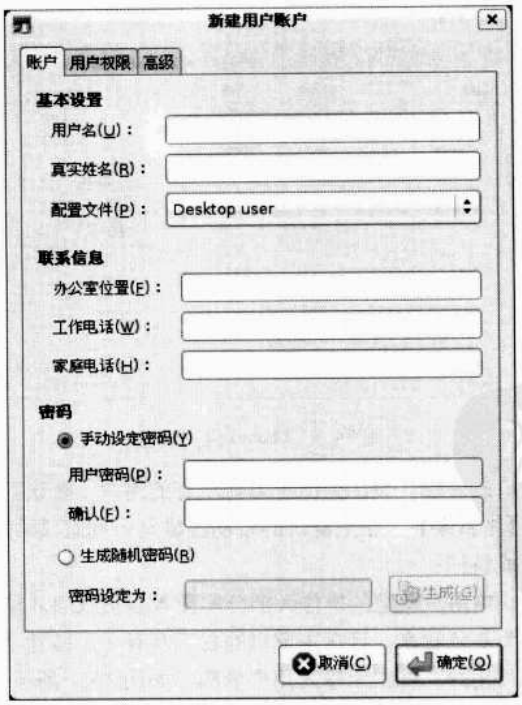


图 14.3 添加用户

在适当的字段内输入新用户的用户名和全称。在【用户密码】和【确认】字段内设置密码，密码必须至少有 6 个字符。用户的密码越长，其他人就越不容易猜到这个密码，从而避免他人不经许可就登录到用户的账号中。这里还建议用户不要根据现成词组来设置密码，密码最好是字母、数字和特殊字符的组合。

选择一个登录 shell。如果不能确定应该选择哪一个 shell，就请接受默认的/bin/bash。默认的主目录是 /home/用户名。可以改变为用户创建的主目录，或者通过取消选择【创建主目录】不为用户创建主目录。

如果选择要创建主目录，默认的配置文件就会从 /etc/skel 目录中复制到新的主目录中。

Ubuntu 使用用户私人组群（user private group，UPG）方案。UPG 方案并不添加或改变 Linux 处理组群的标准方法，它只不过提供了一个新约定。按照默认设置，每当创建一个新用户的时候，一个与用户名相同的独特组群就会被创建。

要为用户指定用户 ID，选择【手工指定用户 ID】。如果这个选项没有被选，从号码 500 开始后的下一个可用用户 ID 就会被分派给新用户。Ubuntu 把低于 500 的用户 ID 保留给系统用户。单击【确定】来创建该用户。

要把用户加入到更多的用户组群中，单击【用户】标签，选择该用户，然后单击【属性】。在【用户属性】窗口中，选择【高级】标签。选择想让该用户加入的组群，以及用户的主要组群，然后单击【确定】。

useradd 指令

我们还可以在 Shell 命令行下，使用 useradd 命令来创建一个锁定的用户账号。useradd 的使用方法如下。

语法

```
useradd [options] LOGIN
```

参数说明

- -b，--base-dir：新用户的基本目录。
- -c comment：用户的注释。
- -d home-dir：用来取代默认的/home/username 主目录。
- -e date：禁用账号的日期，格式为：YYYY-MM-DD。
- -f days：密码过期后，账号禁用前的天数（若指定了 0，账号在密码过期后会被立刻禁用；若指定了-1，密码过期后，账号将不会被禁用）。
- -g group-name：用户默认组群的组群名或组群号码（该组群在指定前必须存在）。
- -G group-list：指定用户所属的附加组，多个组用逗号分隔（组在指定前必须存在）。
- -m：若主目录不存在则创建它。
- -M：不要创建主目录。
- -n：不要为用户创建用户私人组群。
- -r：创建一个 UID 小于 500 的不带主目录的系统账号。
- -p password：使用 crypt 加密的密码。
- -s：用户的登录 shell，默认为/bin/bash。

- -u uid：用户的UID，它必须是唯一的，且大于499。

⚙示例

```
user@ubuntuer:~$ useradd --help
Usage: useradd [options] LOGIN

Options:
-b, --base-dir BASE_DIR      base directory for the new user account
                            home directory
-c, --comment COMMENT        set the GECOS field for the new user account
-d, --home-dir HOME_DIR      home directory for the new user account
  ……（略去部分内容）

user@ubuntuer:~$ useradd learner -d /home/learner
```

14.1.2 删除用户

窗口化删除用户

当具有管理权限的用户需要删除某个用户时，在【用户设置】主窗口中，选择要删除的用户，单击右侧的【删除】按钮，可进入删除用户流程。这时弹出删除用户的提示确认窗口，如图14.4所示。

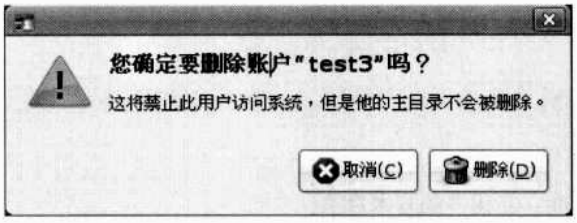


图14.4 删除用户

当确定要删除该用户时，单击【删除】按钮即可。当然要在Ubuntu中删除用户，还可以通过shell命令userdel来实现。

userdel指令

userdel用来删除用户账号及相关文件。该命令修改系统账号文件，删除所有该用户相关的部分。注意用户名称必须是存在的。

⚙语法

```
userdel [-r] login
```

⚙参数说明

-r：用户目录下的文件一并删除。在其他位置上的文件或目录也将找出并删除。

⚙示例

```
user@ubuntuer:~$ userdel testing <==只删除/etc/passwd与/etc/shadow的该账号内容
user@ubuntuer:~$ userdel -r testing
                  <==连该账号的/home/testing与/var/spool/mail/testing都删除
```

其中：

/etc/passwd－：用户账户数据。

/etc/shadow：用户账号信息加密。

/etc/group －：群组信息。

userdel 不允许删除正在线上的用户账号。必须关掉此账号现在在系统上执行的程序才能进行账号删除。不能在 NIS client 端删除 NIS 属性的东西。这些操作须在 NIS server 端上执行。

14.1.3 设置用户属性

窗口化设置用户属性

要查看某个现存用户的属性，在【用户设置】主窗口中，从用户列表中选择要操作的用户，单击【属性】按钮标签。这时弹出如图 14.5 所示的对话框。

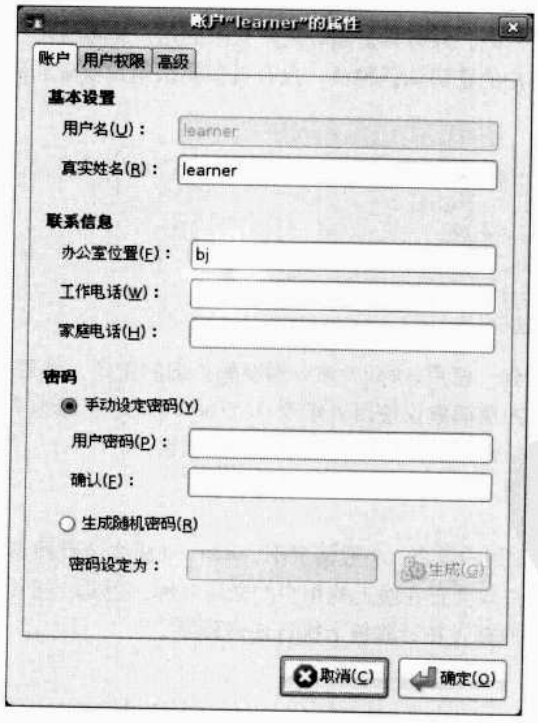


图 14.5 用户属性

账号属性窗口包含三个标签页。

- 【账户】：显示在添加用户时配置的基本用户信息。使用这个标签来设置用户的基本信息、联系方式和密码。
- 【用户权限】：主要用于描述用户在系统中具有的权限，比如是否允许使用打印机、扫描仪等。
- 【高级】：这一个标签页主要是设定用户的主目录路径、登录 Shell、所属组群和用户 ID。

passwd 指令

passwd 指令用于修改用户密码，及通过指派密码和密码失效规则来给某账号开锁。

语法

```
passwd [选项]账户名称
```

参数说明

- -l：锁定已经命名的账户名称，只有具备超级用户权限的用户方可使用。
- -u：解开账户锁定状态，只有具备超级用户权限的用户方可使用。
- -x，--maximum=DAYS：密码使用最大时间（天），只有具备超级用户权限的用户方可使用。
- -n，--minimum=DAYS：密码使用最小时间（天），只有具备超级用户权限的用户方可使用。
- -d：删除用户的密码，只有具备超级用户权限的用户方可使用。
- -S：检查指定用户的密码认证种类，只有具备超级用户权限的用户方可使用。

示例

```
user@ubuntuer:~$ passwd
Changing password for user.
(current) Unix password:
Enter new Unix password:
Retype new Unix password:
passwd：已成功更新密码
```

从上面的示例可以看到，使用 passwd 命令需要输入旧的密码，然后再输入两次新密码。

Root 用户修改其他用户密码建议使用 vi 修改/etc/passwd 文件，修改完毕后使用/usr/sbin/pwck 来检查/etc/passwd 文件的完整性。以免发生不必要的错误。

usermod 指令

usermod 命令用来修改用户信息，一般情况下，usermod 命令会参照命令指令的部分修改用户账号信息。但 usermod 不允许改变正在线上的用户的账号名称，因此，当用 usermod 来改变用户账号信息时，必须确认该用户此刻没在计算机上执行任何程序。

语法

```
# usermod[选项][用户名]
```

参数说明

- -d：更新用户新的登录目录。
- -e：设置新用户的停止日期，日期格式为 MM/DD/YY。
- -f：账户过期几日后永久停权。当值为 0 时账号则立刻被停权；而当值为-1 时则关闭此功能。默认值为-1。
- -g：新用户加入群组后更新用户。
- -G：定义用户为一组 groups 的成员。每个群组使用“？？”隔开，不可以夹杂空格符。
- -l：变更用户登录时的名称，同时用户目录名也会跟着变成新的名称。
- -s：指定新用户 Shell。
- -u：用户 ID 值，必须为唯一的 ID 值。用户目录树下所用的文档目录的 userID 会自动变更，放在支持目录外的文档则要自行手动更改。

⚙示例

```
user@ubuntuer:~$ usermod -d/home/user2 -s/bin/bash user2
//将用户名 user2 的主目录路径设置在/home/user2 下，登录的 Shell 设置为/usr/bin/gcc
```

chfh 指令

chfh 指令供用户更改个人信息，可供所有用户使用，但是在操作前，需要提供该用户的密码。

⚙语法

```
chfn [-f 全名] [-r 房间号] [-w 工作电话] [-h 家庭电话]
```

⚙示例

```
user@ubuntuer:~$ chfn
Password:
正在改变 user 的用户信息
请输入新值，或直接敲回车键以使用默认值
        全名：user
        房间号码[207]:
        工作电话 [64445844]:
        家庭电话 [elp]:
```

chsh 指令

chsh 指令用来更改用户的 shell 设定，该命令可供所有用户使用。

⚙语法

```
chsh [参数] [LOGIN]
```

⚙参数说明

- -h，--help：显示此帮助信息并退出。
- -s，--shell SHELL：该用户账号的新登录 shell。

⚙范例

```
user@ubuntuer:~$ chsh
Password:
```

```
正在更改 user 的 shell
请输入新值，或直接敲回车键以使用默认值
        登录 Shell [/bin/bash]:
```

14.2 用户组的管理

应该这样认识 Linux 中的组管理：不要吝惜对组的使用，在复杂的环境中，不要害怕创建很多组。应该根据资源访问权限而不是基于业务单位去创建组。用户和组信息分别存储在/etc/passwd 文件和 /etc/group 文件中。在图 14.1【用户设置】对话框中，单击【管理组】按钮，进入如图 14.6 所示的组管理界面，然后就可以通过这个对话框对组进行增删查改的操作了。

图 14.6 组管理窗口

14.2.1 增加用户组

窗口化增加用户组

要添加新用户组群，单击【添加组】按钮，弹出如图 14.7 所示的对话框。输入新组群的名称来创建。要为新组群指定组群 ID，通过单击右侧的微调按钮来更改组 ID。然后可以从当前的用户列表中选择属于当前组的组成员。最后单击【确定】按钮来创建组群。新组群就会出现在组群列表中。

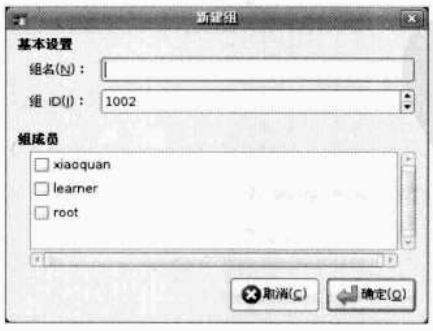


图 14.7 创建新组群

groupadd 指令

语法

```
groupadd [参数] GROUP
```

参数说明

- -g gid：组群的 GID。
- -r：创建小于 500 的系统组群。
- -f：若组群已存在，退出并显示错误（组群不会被改变）。如果指定了 -g 和-f 选项，而组群已存在，-g 选项就会被忽略。

示例

```
root@ubuntuer:~# groupadd ubuntu
```

14.2.2 删除用户组

窗口化删除用户组

在窗口下删除用户组很简单，在图 14.6 所示的组管理界面中，选择目标删除组，单击右侧的【删除】按钮，系统将弹出如图 14.8 所示的提示信息，单击【删除】按钮，用户组就被删除了，如果选择【取消】，系统将不对该用户组做进一步操作。

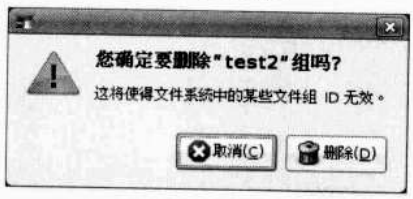


图 14.8 删除组群

groupdel 指令

语法

```
groupdel 组名
```

示例

```
root@ubuntuer:~# groupdel ubuntu
```

groupdel 命令会修改系统账号文件，会删除所有已存在的 group。当然前提是群组名必须存在。需要注意的是，如果有任何一个群组的用户在线上的话就不能删除该群组。最好先删除用户后再删除该群组。

14.2.3 管理用户组属性

窗口化设置用户组

要查看某一现存组群的属性，从图 14.6 组管理窗口中选择目标组群，然后在右侧的面板上单击【属性】按钮，弹出如图 14.9 所示的对话框。

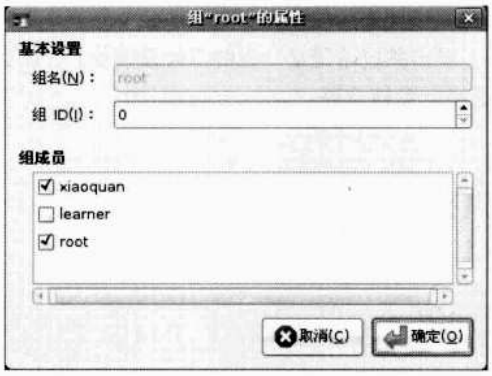


图 14.9 组群属性

组属性对话框显示了哪些用户是组群的成员。选择其他用户来把他们加入到组群中，或取消选择用户来把他们从组群中移除。单击【确定】按钮来修改该组群中的用户。

groups 指令

语法

```
groups [用户名].
```

示例

```
root@ubuntuer:~# groups user
root adm dialout fax cdrom floppy tape audio dip video plugdev scanner
fuse lpadmin admin sambashare
```

说明

Groups 命令用于查看用户的用户组信息，如果不带用户名作为参数的话，默认显示当前用户的用户组信息。

groupmod 指令

语法

```
groupmod [参数] [用户组]
```

参数说明

- -g GID：为用户组指定新的 ID。
- -o：与-g 选项同时使用，用户组的新 GID 可以与系统已有用户组的 GID 相同。

- -n：更改用户组名字。

示例

```
root@ubuntuer:~# groupmod -n modifyubuntuer ubuntu
//此命令将组 ubuntu 组名修改为 modifyubuntuer
```

14.3 用户查询命令

在 Linux 系统中，同时也提供一些查询用户相关信息的指令，如查看当前在线用户的 who 指令，查看指定用户详细信息的 finger 指令，显示最近登录用户的 last 指令，以及显示用户及所在组的系统 ID 的 id 指令。

14.3.1 who 指令

通过 who 命令，可以查看当前在线上的用户情况。who 这个命令非常有用。如果用户想和其他用户建立即时通讯，比如使用 talk 命令，那么首先要确定的就是该用户确实在线上，不然 talk 进程就无法建立起来。又如，系统管理员希望监视每个登录的用户此时的所作所为，也要使用 who 命令。who 命令的常用命令格式和常用选项如下所述。

语法

```
who - [husfV] [user]
```

参数说明

- -a：显示所有用户的所有信息。
- -m：显示运行该程序的用户名，和“who am I”的作用一样。
- -q：只显示用户的登录账号和登录用户的数量，该选项优先级高于其他任何选项。
- -s：使用简短的格式来显示。
- -u：在登录用户后面显示该用户最后一次对系统进行操作距今的时间。
- -H：显示列标题。

示例

```
root@ubuntuer:~# who -aH
NAME       LINE         TIME.            IDLE          PID COMMENT  EXIT
           system boot  2008-07-19 01:11
LOGIN      tty4         2008-07-19 01:11               4346 id=4
LOGIN      tty5         2008-07-19 01:11               4347 id=5
LOGIN      tty2         2008-07-19 01:11               4351 id=2
           run-level 2  2008-07-19 01:11                   last=
LOGIN      tty3         2008-07-19 01:11               4354 id=3
LOGIN      tty6         2008-07-19 01:11               4358 id=6
LOGIN      tty1         2008-07-19 01:12               5714 id=1
user     + tty7         2008-08-26 21:39 00:27          6120 (:0)
```

```
user     + pts/0        2008-08-27 00:42 00:27       7978 (:0.0)
          pts/1        2008-08-27 01:12            7998 id=ts/1  term=0 exit=0
user     + pts/2        2008-08-27 00:55   .         8122 (192.168.1.100)
```

说明

本示例中，主标题的含义如表 14.1 所示。

表 14.1 who 输出常用标题含义

标题	含义
NAME	用户登录
LINE	用户登录使用的终端
TIME	用户登录时间
IDLE	用户空闲时间，即停止操作的时间
PID	用户登录 shell 的进程 ID

也可以单独使用 who 命令，这时将显示登录用户名、使用终端设备以及登录到系统（来源地址）的时间三项内容，如下所示：

```
root@ubuntuer:~# who
user     tty7         2008-08-26 21:39 (:0)
user     pts/0        2008-08-27 00:42 (:0.0)
user     pts/2        2008-08-27 00:55 (192.168.1.100)
```

14.3.2 finger 指令

finger 指令的用法也很简单，直接使用 finger username 就可以知道任何一个人的相关信息。这个相关信息基本上都在/etc/passwd 中，当然，里面还有/var/spool/mail 这个邮件存放地点，所以还会显示是否有邮件信息。不过，finger 必须配合 chfn 指令，才能显示比较多的信息。但是这个指令不是很安全，所以不是每个系统都安装了这个程序。

语法

```
finger [options] user[@address]
```

参数说明

- -l：多行显示。
- -s：单行显示。这个选项只显示登入名称、真实姓名、终端机名称、闲置时间、登入时间、办公室号码及电话号码。如果所查询的用户是远端服务器用户，这个选项无效。

示例

```
root@ubuntuer:~# finger
Login    Name      Tty      Idle  Login Time   Office     Office Phone
user     user      tty7       34  Aug 26 21:39 (:0)
user     user      pts/0      34  Aug 27 00:42 (:0.0)
```

```
user     user      pts/2        Aug 27 00:55 (192.168.1.100)
root@ubuntuer:~# finger user
Login: hplip                        Name: HPLIP system user
Directory: /var/run/hplip           Shell: /bin/false
Never logged in.
No mail.
No Plan.

Login: user                         Name: user
Directory: /home/user               Shell: /bin/bash
Office: 207, 64445844               Home Phone: elp
On since Tue Aug 26 21:39 (CST) on tty7 from :0
    34 minutes 19 seconds idle
On since Wed Aug 27 00:42 (CST) on pts/0 from :0.0
   34 minutes 22 seconds idle
On since Wed Aug 27 00:55 (CST) on pts/2 from 192.168.1.100
No mail.
No Plan.
```

说明

finger 可以让用户查询一些其他用户的资料。会列出来的资料有：

- Login Name
- User Name
- Home directory
- Shell
- Login status
- mail status
- .plan
- .project
- .forward

其中.plan、.project 和.forward 就是用户的 Home Directory 里的.plan、.project 和.forward 等文件里的信息。finger 指令并不限于在本地主机上查询，也可以寻找某一个远端主机上的用户，只要提供一个类似 E-mail 的地址即可。

14.3.3 last 指令

last 指令用于显示系统开机以来或从每月初至今登入者的信息，可供所有用户使用。

语法

```
last [参数]
```

参数说明

- -R：省略 hostname 域。

- –num：展示前 num 个信息。

示例

```
wtmp begins Fri Aug  1 21:18:30 2008
root@ubuntuer:~# last -5
user     pts/2        192.168.1.100    Wed Aug 27 00:55   still logged in
user     pts/1        192.168.1.100    Wed Aug 27 00:42 - 01:12 (00:29)
user     pts/0: 0.0            Wed Aug 27 00:42   still logged in
user     tty7: 0              Tue Aug 26 21:39   still logged in
user     tty7: 0              Tue Aug 26 21:35 - 21:38 (00:03)

wtmp begins Fri Aug  1 21:18:30 2008
root@ubuntuer:~# last -5 user
user     pts/2        192.168.1.100    Wed Aug 27 00:55   still logged in
user     pts/1        192.168.1.100    Wed Aug 27 00:42 - 01:12 (00:29)
user     pts/0: 0.0            Wed Aug 27 00:42   still logged in
user     tty7: 0              Tue Aug 26 21:39   still logged in
user     tty7: 0              Tue Aug 26 21:35 - 21:38 (00:03)

wtmp begins Fri Aug  1 21:18:30 2008
```

14.3.4 id 指令

id 指令用于显示用户的 ID，以及所属群组的 ID。id 会显示用户以及所属群组的实际、有效的 ID。若两个 ID 相同，则仅显示实际 ID；若仅指定用户名称，则显示目前用户的 ID。

语法

```
id [-gGnru][--help][--version][用户名称]
```

参数

- –g，––group：显示用户所属群组的 ID。
- –G，––groups：显示用户所属附加群组的 ID。
- –n，––name：显示用户、所属群组或附加群组的名称。
- –r，––real：显示实际 ID。
- –u，––user：显示用户 ID。
- –help：显示帮助。
- –version：显示版本信息。

示例

```
root@ubuntuer:~# id
uid=0(root) gid=0(root) groups=0(root)
root@ubuntuer:~$ id -G
0 1 2 3 4 6 10
```

14.4 用户的切换

在 Linux 中，超级用户称为 root。root 用户可以控制所有的程序，访问所有文件，使用系统上的所有功能。对 root 用户来说没有不可以做的事情。就管理的角度而言，root 的权限是至高无上的。所以，root 账号一定要通过安全的密码保护起来，这一点非常重要。不应该使用 root 身份来处理日常的事务。其他用户也可以被赋予 root 特权，但一定要谨慎行事。通常，可以配置一些特定的程序由某些用户以 root 身份去运行，而不必赋予他们 root 权限。

14.4.1 Su 指令

要切换到 root 用户登录，使用 su 命令。命令 su 的意思是 substitute users（代替用户），它允许暂时以其他用户身份登录。当只输入 su 命令本身然后按 Enter 键，输入 root 密码，就会看到命令提示符已发生改变，这种改变显示了新获得的超级用户状态，root 在账号的提示符的前端，#在提示符的后端。当要使用根用户身份进行的工作结束后，在提示下输入 exit 命令，就会返回到普通用户账号。

语法

```
su [OPTION]... [-] [USER [ARG]...]
```

参数说明

- -f，--fast：不必读启动文件（如 csh.cshrc 等），仅用于 csh 或 tcsh。
- -m -p，--preserve-environment：执行 su 时不改变环境变量。
- -c command，--command=command：变更为账号为 USER 的用户并执行指令（command）后再变回原来的用户。
- -s shell，--shell=shell：指定要执行的 shell（bash csh tcsh 等），预设值为/etc/passwd 内的该使用者（USER）shell。
- --help：显示说明文件。
- --version：显示版本资讯。
- --l，--login：加了这个参数之后，就好像是重新登录（login）为该用户一样，大部分环境变量（HOME SHELL USER 等）都是以该用户（USER）为主，并且工作目录也会改变。如果没有指定 USER，内定是 root。
- USER：欲变更的用户账号。
- ARG：传入新的 shell 参数。

示例

```
user@ubuntuer:~# su -c ls root <==变更账号为 root 并在执行 ls 指令后退出变回原用户
user@ubuntuer:~# su root -f <==变更账号为 root 并传入-f 参数给新执行的 shell
user@ubuntuer:~# su - learner <==变更账号为 learner 并改变工作目录至 learner
                                的主目录 (home dir)
```

14.4.2 sudo 指令

以系统管理者的身份执行指令，也就是说，通过 sudo 所执行的指令就好像是 root 亲自执行，注意只供在/etc/sudoers 配置文件中出现的用户。

语法说明

```
sudo [参数] [命令]
```

参数

- -V：显示版本编号。
- -h：显示版本编号及指令的使用方式说明。
- -l：显示出自己（执行 sudo 的用户）的权限。
- -v：因为 sudo 在第一次执行时或是在 N 分钟内没有执行会提示输入密码，这个参数是重新做一次确认，如果超过 N 分钟，也会提示输入密码。N 一般预设为 5。
- -k：强迫用户在下一次执行 sudo 时提示输入密码（不论有没有超过 N 分钟）。
- -b：将要执行的指令放在后台执行。
- -p prompt：可以更改问密码的提示语，其中 %u 会替换为用户的账号名称，%h 会显示主机名称。
- -u username/#uid：不加此参数，代表要以 root 的身份执行指令，而加了此参数，可以以 username 的身份执行指令（#uid 为该 username 的用户号码）。
- -s：执行环境变量中的 SHELL 所指定的 shell，或是 /etc/passwd 里所指定的 shell。
- -H：将环境变量中的 HOME（用户主目录）指定为要变更身份的用户主目录。

示例

```
user@ubuntuer:~$ sudo -l  <==注：列出目前的权限
User root may run the following commands on this host:
   (ALL) ALL

user@ubuntuer:~$ sudo -V  <==注：列出 sudo 的版本信息
Sudo version 1.6.9p10
```

14.4.3 visudo 指令

visudo 即是编辑 sudo 的配置文件 soduers 的命令。sudo 允许经过同意的用户以 root 用户的身份执行指令，sudo 参考/etc/sudoers 这个配置文件来判定谁是被授权的用户。sudo 将会提示用户输入密码来启始一段 N 分钟的允许时间（其中 N 是在安装的时候定义的且默认值为 5 分钟）。

sudoers 这个文件是由一个选择性的主机别名（host alias）节区（section）、一个选择性的指令别名（command alias）节区以及用户说明（user specification）节区所组成的。所有的指令别名或主机别名必须以它们自己的关键字开始（Host_Alias/Cmnd_Alias）。

```
user@ubuntuer:~$ visudo
# sudoers file.
#
# This file MUST be edited with the 'visudo' command as root.
#
# See the sudoers man page for the details on how to write a sudoers file.
#
# Host alias specification
# User alias specification
# Cmnd alias specification
# Defaults specification
Defaults    env_reset
Defaults    env_keep = "COLORS DISPLAY HOSTNAME HISTSIZE INPUTRC KDEDIR \
                  LS_COLORS MAIL PS1 PS2 QTDIR USERNAME \
                  LANG LC_ADDRESS LC_CTYPE LC_COLLATE LC_IDENTIFICATION \
                  LC_MEASUREMENT LC_MESSAGES LC_MONETARY LC_NAME LC_NUMERIC \
                  LC_PAPER LC_TELEPHONE LC_TIME LC_ALL LANGUAGE LINGUAS \
                  _XKB_CHARSET"
# Runas alias specification
# User privilege specification
root    ALL=(ALL) ALL
# Uncomment to allow people in group wheel to run all commands
# %wheel        ALL=(ALL)       ALL
# Same thing without a password
# %wheel        ALL=(ALL)       NOPASSWD: ALL
# Samples
# %users  ALL=/sbin/mount /cdrom, /sbin/umount /cdrom
# %users  localhost=/sbin/shutdown -h now
```

注意 只有第一次使用时会有说明（在用户说明节区里有记录的用户）。

⚙用户说明节区语法格式

```
User-Alias 用户别名 = 用户列表
```

- User-Alias：这是一个关键字。
- 用户别名：一个大写的别名。
- 用户列表：以逗号间隔的目标用户。
- 示例：user-Alias ADMIN=mike，tome。

⚙主机别名节区语法格式

```
Host_Alias 主机别名 = 主机列表
```

- Host_Alias：这是一个关键字。
- 主机别名：一个大写的别名。
- 主机列表：以逗号间隔的一些主机名称。
- 示例：#Host-Alias FILESERV=fs1，fs2。

⚙指令别名节区语法格式

```
Cmnd_Alias 指令别名 = 指令列表
```

- Cmnd_Alias：这是一个关键字。
- 指令别名：一个大写的别名。
- 指令列表：以逗号间隔的一些指令。
- 示例：#Cmnd-Alias SERVICE=/sbin/service，/sbin/chkconfig。

所有在#符号后面的文字都会被当作是注解。太长的行可以使用倒斜线\字符来分成新的行。保留的别名'ALL 在{Host，Cmnd}_Alias 里都可以使用。不要用 ALL 来定义一个别名，该别名无效。注意到 ALL 暗示全部的主机和指令。可以使用这个语法从整个范围中减掉一些项目。

```
user  host=ALL，!ALIAS1，!/etc/halt...
```

14.5 用户配置文件

Linux 系统中并不识别账号名称，它识别的是账号 ID，账号 ID 保存在/etc/passwd 文件中。登录 Linux 主机时，在输入完账号和密码时，Linux 会先查找/etc/passwd 文件中是否有这个账号，如果没有则跳出；如果有的话，它会读取该账号的 user ID 和 group ID，同时该账号的根目录和 shell 也读了出来。然后再去核对密码表，在/etc/shadow 中找出刚刚输入的账号和 userID，核对输入的密码是否正确。核对无误后才可以登录到当前用户 shell。下面首先了解一下用户账号文件。

14.5.1 /etc/passwd 文件

/etc/passwd 是存放系统用户的文件，如下：

```
user@ubuntuer:~$ vi /etc/passwd

root:x:0:0:root，beijing，82118888，81238123:/root:/bin/bash
bin:x:1:1:bin:/bin:/sbin/nologin
daemon:x:2:2:daemon:/sbin:/sbin/nologin
adm:x:3:4:adm:/var/adm:/sbin/nologin
lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin
sync:x:5:0:sync:/sbin:/bin/sync
shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown
halt:x:7:0:halt:/sbin:/sbin/halt
mail:x:8:12:mail:/var/spool/mail:/sbin/nologin
news:x:9:13:news:/etc/news:
uucp:x:10:14:uucp:/var/spool/uucp:/sbin/nologin
operator:x:11:0:operator:/root:/sbin/nologin
games:x:12:100:games:/usr/games:/sbin/nologin
ftp:x:14:50:FTP User:/var/ftp:/sbin/nologin
nobody:x:99:99:Nobody:/:/sbin/nologin
dbus:x:81:81:System message bus:/:/sbin/nologin
rpm:x:37:37::/var/lib/rpm:/sbin/nologin
apache:x:48:48:Apache:/var/www:/sbin/nologin
distcache:x:94:94:Distcache:/:/sbin/nologin
nscd:x:28:28:NSCD Daemon:/:/sbin/nologin
vcsa:x:69:69:virtual console memory owner:/dev:/sbin/nologin
mysql:x:27:27:MySQL Server:/var/lib/mysql:/bin/bash
```

```
webalizer:x:67:67:Webalizer:/var/www/usage:/sbin/nologin
squid:x:23:23::/var/spool/squid:/sbin/nologin
netdump:x:34:34:Network Crash Dump user:/var/crash:/bin/bash
pcap:x:77:77::/var/arpwatch:/sbin/nologin
avahi:x:70:70:Avahi daemon:/:/sbin/nologin
named:x:25:25:Named:/var/named:/sbin/nologin
mailnull:x:47:47::/var/spool/mqueue:/sbin/nologin
smmsp:x:51:51::/var/spool/mqueue:/sbin/nologin
haldaemon:x:68:68:HAL daemon:/:/sbin/nologin
rpc:x:32:32:Portmapper RPC user:/:/sbin/nologin
xfs:x:43:43:X Font Server:/etc/X11/fs:/sbin/nologin
rpcuser:x:29:29:RPC Service User:/var/lib/nfs:/sbin/nologin
nfsnobody:x:65534:65534:Anonymous NFS User:/var/lib/nfs:/sbin/nologin
sshd:x:74:74:Privilege-separated SSH:/var/empty/sshd:/sbin/nologin
uploader:x:500:500::/home/uploader:/bin/bash
```

/etc/passwd 文件中，任何一个用户的内容格式如下：

```
Root:AAAAAA:0:0:root:/root:/usrbin/sh
```

一共有 7 项，中间使用 ":" 分开，即

```
用户名：密码：UID：GID：用户描述：用户主目录：登录 SHELL
```

其中各项说明如下。

- 用户名：账号名称由于对应用户 ID，这个是系统默认用户 root 超级管理员，在同一个系统账号名称是唯一的，长度根据不同的 Linux 系统而定，一般是 1~8 个字符，第一个字符必须是字母。如果超过 8 个字符，则只有 8 个字符是有效的。
- 密码：加密的密码，密码应为 6~8 个字符，其中必须有数字或特殊字符，由于系统中还有一个/etc/shadow 文件用于存放加密后的密码，所以在这里这一项用 x 来表示，如果用户没有设置密码，则该项为空。
- 用户 ID：该项是系统内部用来识别不同的用户的，不同的用户识别码不同，其中用户 ID 有以下几种：
  0 代表系统管理员，如果想建立一个系统管理员的话，可以建立一个普通账户，然后将该账户的用户 ID 改为 0 即可；1~500 是系统预留的 ID；500 以上则供普通用户使用。
- 组 ID：其实，该项和用户 ID 差不多，它用来规范群组，且与/etc/group 有关。
- 描述信息：这个字段几乎没有什么作用，只是用来解释这个账号的意义。
- 用户主目录：就是用户登录系统的起始目录，用户登录系统后将首先进入该目录。root 用户默认的是/root，普通用户的是/home/用户名。
- 用户登录 shell：就是用户登录系统时使用的 shell。

14.5.2 /etc/shadow 文件

在早期的 Unix 操作系统中，用户的账号信息和密码信息都保存在 passwd 文件中，尽管系统已经对密码进行了加密，并且以密文的方式保存在 passwd 文件中，但是由于 passwd 文件对于系统中的所有用户是可读的，密码比较容易破解，存在较大的安全隐患。现在使用 shadow 文件保存密文

的用户密码，使用 passwd 文件保存用户账号其他信息。只有管理员用户才可以读取 shadow 文件中的内容。由于这个文件可能被破解，所以一定不要将该文件内容泄露给他人，以保证系统安全。/etc/shadow 文件内容如下：

```
user@ubuntuer:~$ vi /etc/shadow

root:$1$tLbpGJDp$pgH.nHcH1jWasn1QUXMoy1:13912:0:99999:7:::
bin:*:13579:0:99999:7:::
daemon:*:13579:0:99999:7:::
adm:*:13579:0:99999:7:::
lp:*:13579:0:99999:7:::
sync:*:13579:0:99999:7:::
shutdown:*:13579:0:99999:7:::
halt:*:13579:0:99999:7:::
mail:*:13579:0:99999:7:::
news:*:13579:0:99999:7:::
uucp:*:13579:0:99999:7:::
operator:*:13579:0:99999:7:::
games:*:13579:0:99999:7:::
gopher:*:13579:0:99999:7:::
ftp:*:13579:0:99999:7:::
nobody:*:13579:0:99999:7:::
dbus:!!:13579:0:99999:7:::
rpm:!!:13579:0:99999:7:::
apache:!!:13579:0:99999:7:::
distcache:!!:13579:0:99999:7:::
nscd:!!:13579:0:99999:7:::
vcsa:!!:13579:0:99999:7:::
mysql:!!:13579:0:99999:7:::
webalizer:!!:13579:0:99999:7:::
squid:!!:13579:0:99999:7:::
netdump:!!:13579:0:99999:7:::
pcap:!!:13579:0:99999:7:::
avahi:!!:13579:0:99999:7:::
named:!!:13579:0:99999:7:::
mailnull:!!:13579:0:99999:7:::
smmsp:!!:13579:0:99999:7:::
haldaemon:!!:13579:0:99999:7:::
rpc:!!:13579:0:99999:7:::
xfs:!!:13579:0:99999:7:::
rpcuser:!!:13579:0:99999:7:::
nfsnobody:!!:13579:0:99999:7:::
sshd:!!:13579:0:99999:7:::
```

/etc/shadow 文件中，任何一个用户的内容格式如下：

```
root:$1$tLbpGJDp$pgH.nHcH1jWasn1QUXMoy1:13912:0:99999:7:::
```

一共有 9 项，每一项使用 ":" 分开，即：

用户名:密码:上次改动密码的日期：密码可被改动的天数：密码不可被改动的天数：密码变更期期限快到前的警告期：账号失效期：账号取消日期：保留

- 用户名：和 passwd 对应，和 passwd 的意思相同。
- 密码：这才是真正的密码，并且已经加密过了，只能看到一些特殊符号。需要注意的是这些密码很难破解，但是不等于不能破解。有些密码栏的第一个字符为*，表示这个用户账户不用来登录，如果不想让他登录了，可以在对应的账号前面加“*”。
- 上次改动密码的日期：这项记录了改动密码的最后日期，为什么是 13912 呢？这是因为 Linux 计算日期的方法是以 1970 年 1 月 1 日作为 1，1971 年 1 月 1 日就是 366，以此类推到作者修改密码的日期就表示为 13912 了。
- 密码可被改动的天数：设定用户可以修改密码的最小天数，超过这个时间后，密码将不能再修改。如果值为 0 表示没有做限定，随时可以修改密码。
- 密码不可被改动的天数：由于害怕密码被人盗取而危害到整个系统的安全，所以安排了这个字段，你必须在这个时间内重新修改密码，否则这个账号将暂时失效。上面的 99999，表示密码不需要重新输入，最好设定一段时间修改密码，以确保系统安全。
- 密码变更期期限快到前的警告期：当账号的密码失效期限快到时，系统将依据这个字段的设定发出警告，提醒用户“再过 n 天您的密码将过期，请尽快重新设定密码”。默认是 7 天。
- 账号失效期：如果用户过了警告期没有重新输入密码，使得密码失效，而该用户在这个字段限定的时间内又没有向管理员反映，让账号重新启用，那么这个账号将暂时失效。
- 账号取消日期：这个日期跟第三个字段一样，都是使用 1970 年以来的日期设定方法。这个字段表示这个账号在此字段规定的日期之后将无法再使用。这个字段通常用于收费服务系统中，可以规定一个日期让该账号不能再使用。
- 保留：最后一个字段是保留的，看以后有没有新功能加入。

14.5.3 /etc/group 文件

下面来看看群组文件的内容：

```
user@ubuntuer:~$ vi /etc/group

root:x:0:root
bin:x:1:root, bin, daemon
daemon:x:2:root, bin, daemon
sys:x:3:root, bin, adm
adm:x:4:root, adm, daemon
tty:x:5:
disk:x:6:root
lp:x:7:daemon, lp
mem:x:8:
kmem:x:9:
wheel:x:10:root
mail:x:12:mail
news:x:13:news
```

```
uucp:x:14:uucp
man:x:15:
games:x:20:
"/etc/group" 17L, 275C
```

/etc/group 文件中，任何一个群组的内容格式如下：

```
root:x:0:root
```

一共有 4 项，每一项使用“：”分开，即：

```
群组名称：群组密码：群组 ID：支持账号的名称
```

其中各项说明如下：

- 群组名称：就是群组的名称了。
- 群组密码：通常不需要设定，因为很少使用群组登录。不过这个密码也被记录在/etc/gshadow 中了。
- 群组 ID：也就是组 ID。
- 支持账号的名称：这个群组的所有账号。如果你想让用户 qiuri 也属于 root 这个群组，就在第一行最后加上“，qiuri”。注意添加的时候没有空格。

14.5.4 应用举例

下列步骤演示了在启用屏蔽密码的系统上使用 useradd ubuntuer 命令后的情形。

Step 01 在/etc/passwd 文件中新添了有关 ubuntuer的一行。

这一行的特点如下：

- 它以用户名 ubuntuer 开头。
- 密码字段有一个 x，表示系统使用屏蔽密码。
- 创建 UID。
- 创建 GID。
- 可选的 GECOS 信息被留为空白。
- ubuntuer 的主目录被设为/home/ubuntuer/。
- 默认的 shell 被设为/bin/bash。

Step 02 在/etc/shadow 文件中新添了有关 ubuntuer的一行。

这一行的特点如下：

- 它以用户名 ubuntuer 开头。
- 出现在/etc/shadow 文件中密码字段内的两个叹号（!!）会锁住账号。
- 密码被设置为永不过期。

Step 03 在/etc/group 文件中新添了一行有关 ubuntuer 组群的信息。

和用户名相同的组群叫做用户私人组群（user private group）。在/etc/group 文件中新添的这一行具有如下特点：

- 它以组群名 ubuntuer 开头。
- 密码字段有一个 x，表示系统使用屏蔽密码。
- GID 与列举/etc/passwd 文件中用户 ubuntuer 行中的相同。

Step 04 在/etc/gshadow 文件中新添了有关 ubuntuer 组群的一行。

这一行的特点如下：

- 它以组群名 ubuntuer 开头。
- 出现在/etc/gshadow 文件中密码字段内的一个叹号（!）会锁住该组群。
- 其他所有字段都为空白。

Step 05 用于用户ubuntuer的目录被创建在/home/ 目录之下。

该目录为用户ubuntuer 和组群 ubuntuer所有。它的读写和执行权限仅为用户ubuntuer所有，其他权限都被拒绝。

Step 06 /etc/skel/目录（包含默认用户设置）内的文件被复制到新建的/home/ubuntuer/目录中。

这时候，系统上就存在了一个叫做 ubuntuer 的被锁的账号。要激活它，管理员必须使用 passwd 命令给账号指派一个密码，或者还可以设置密码失效规则。

14.6 课后练习

1．Ubuntu 的用户管理器有哪些功能，如何打开用户管理器？

2．请尝试通过用户管理器为 Ubuntu 添加用户，删除用户。

3．在 Ubuntu Shell 中添加用户，删除用户的指令是什么？如何通过这些指令添加特定需求的用户，如何删除用户？

4．在 Ubuntu 中用户组是一个什么概念？如何通过用户组来管理用户的权限。

5．如何在命令行中查在线用户？Ubuntu Shell 下如何通过 finger、last、id 等指令定位特定用户信息？

6．请理解超级用户 root、sudo，并且拓展学习用户切换。

7．在 Ubuntu 中，passwd、shadow 两个文件分别有哪些内容项？它们的主要用途是什么？

Chapter15 用户磁盘配额

在 Linux 系统中，由于是多人多任务的环境，所以会有多人共同使用一个硬盘空间的情况发生。如果其中有少数几个用户大量占用了硬盘空间的话，那势必限制其他用户的使用权力，因此管理员应该适当地开放硬盘权限给用户，以妥善地分配系统资源。Linux 系统中，可以通过 quota 的设置来限制用户和用户组的硬盘空间配额。磁盘配额技术在 Linux 操作系统中的应用是十分广泛的，例如网站、ICP 等为用户设置信箱大小、磁盘使用空间、虚拟主机等都用到了磁盘配额技术。配额可以使各个用户和组无法占用分区的全部空间；可以通过限定 inode 数而配置配额，每个 inode 与特定文件相关联；也可以设置绝对极限。本章将介绍什么是用户磁盘配额、quota 的使用方法，以及对特定用户使用配额的操作步骤，最后将通过一个实例来演示对用户进行磁盘配额限制。

15.1 磁盘配额基础

quota 字面含义就是“配额”，也就是有多少“限额”的意思，在 Ubuntu Linux 容量空间上，是有多少容量限制的意思即磁盘配额。比如说，用户的预设目录都是在/home 下，如果/home 是个独立的 partition，假设为 10GB，而/home 下共有 30 个用户，这样每个用户平均有 333MB 的空间，但是偏偏有用户在其目录下塞了好多大文件，占掉了 8GB 的空间，想想看，是否造成其他正常用户的不便呢？这个时候就得要靠 quota 了！再比如，大家都曾申请过网络的 Mail 服务，申请邮件是肯定会注意到有 20MB 的邮件空间、30MB 的免费网页空间，这个 20MB、30MB 也是由 quota 来实现的。

在 Linux 当中，用于硬盘空间管理，比较常使用情况如下：

- 针对 Web 服务器，例如：每个人的网页空间的容量限制。
- 针对 Mail 服务器，例如：每个人的邮件空间限制。
- 针对 File 服务器，例如：每个人最大的可用网络硬盘空间。

15.1.1 quota 的使用限制

在 Ubuntu 中使用 quota 时，有几个基本的限制需要注意。

仅针对整个 partition

Quota 在实际运作的时候，是针对“整个 partition”进行限制的，例如，如果用户的/dev/hda5 是挂载在/home 下，那么在/home 底下的所有目录都会受到限制！

核心必须支持 quota

Linux 系统核心必须支持 quota 这个模块才行。如果使用 FC4 的默认核心，那么系统默认支持 quota 这个模块；如果是自行编译核心，那么请特别留意是否已经开启了 quota 这个模块。至于核心

编译的过程会在后面进行说明。

Quota 的记录文件

目前新版的 Linux 发行版如 Fedora Core 4 与 SuSE Server9 等使用的是 Kernel2.6.xx 的核心版本，这个核心版本支持新的 quota 模块，使用的默认文件（aquota.user，aquota.group）将不同于旧版本的 quota.user，quota.group。而旧版本的 quota 可以通过 convertquota 这个程序来转换。

只对普通用户有效

并不是所有在 Linux 上面的账号都可以设定 quota，例如 root 就不能设定 quota，因为整个系统所有的数据 root 都有权限访问。

15.1.2 quota 对硬盘配额的限制项目

quota 这个程序针对整个 partition 的限制项目主要分为如下几个部分。

soft

这是最低限制容量的意思，用户在宽限期间之内，他的容量可以超过 soft，但必须在宽限时间之内将磁盘容量降低到 soft 的容量限制之下。

hard

不可超过的容量。通常，hard limit 会比 soft limit 高，例如网络磁盘空间为 30 MB，那么 hard limit 就设定为 30 MB，但是为了让用户有一定的警戒心，所以当使用空间超过 25 MB 时，那么系统就会警告用户，让用户可以在宽限时间内将文件量降低至 25 MB 之内。

宽限时间

当用户使用的空间超过了 soft limit，却还没有到达 hard limit 时，那么在这个宽限时间之内，就必须请用户将使用的磁盘容量降低到 soft limit 之下。而当用户的磁盘容量超过 soft limit 时，宽限时间就会自动被启用，而在用户将容量降低到 soft limit 之下时，宽限时间自动取消。

15.2 Quota 的安装

Ubuntu 系统上并不默认安装 quota。因此在使用磁盘配额功能之前需要先安装 quote 软件。

15.2.1 窗口化安装 quota

用户可以通过新立得软件管理器进行 quota 的安装。在菜单中执行【系统】|【系统管理】|【新立得软件管理】命令，即弹出 synaptic package manage 窗口，如图 15.1 所示。

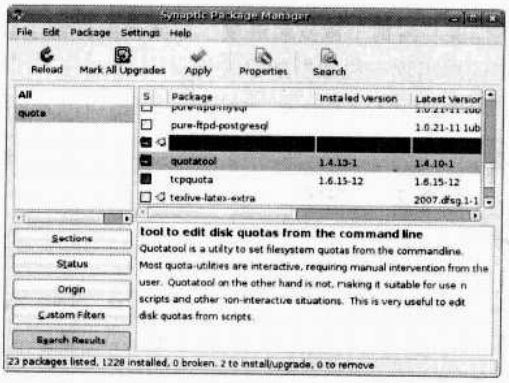


图 15.1　新立得软件包管理器中选择 quota

在上面的窗口中，单击 Search 按钮，查找 quota 软件；在结果集中选中 quota 和 quota tool，然后单击 Apply 按钮，弹出 Summary 提示窗口，并列出选择的项，提示确认安装/重新安装，如图 15.2 所示。

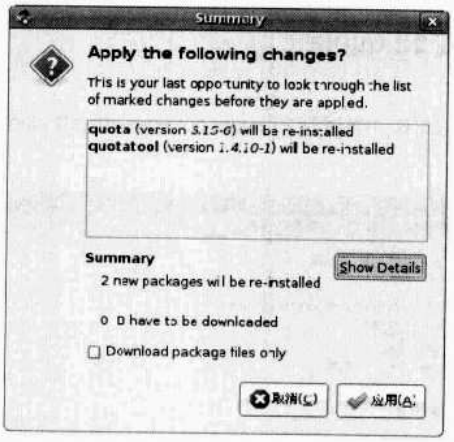


图 15.2　确认安装 quota

单击 Summary 对话框中的【应用】按钮，进入下载并安装窗口，如图 15.3 所示。

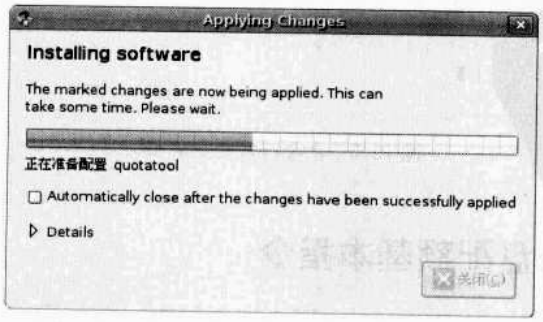


图 15.3　下载安装 quota 进度显示

Applying Changes 对话框提示当前下载安装进度，如果需要详细了解安装情况时，可以单击图中的 Details，这时，窗口中间即出现一栏黑屏的命令行。

最后安装完成后，窗口标题显示成 Changes applied，如图 15.4 所示。

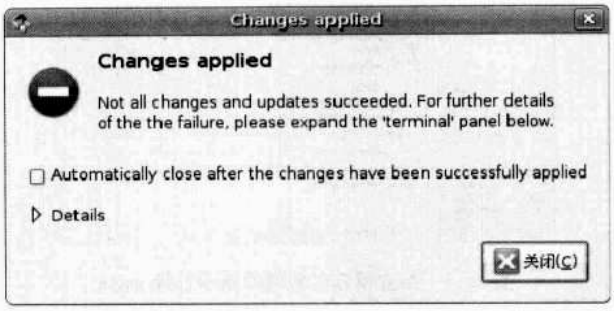


图 15.4 安装完成提示窗口

看到此窗口时，quota 已经安装完成了，单击【关闭】按钮就可以开始使用 quota 了。

15.2.2 命令行安装 quota

另外，也可以在 Shell 终端下，通过运行 sudo apt-get install quota 命令进行 quota 的安装，具体过程如图 15.5 所示。

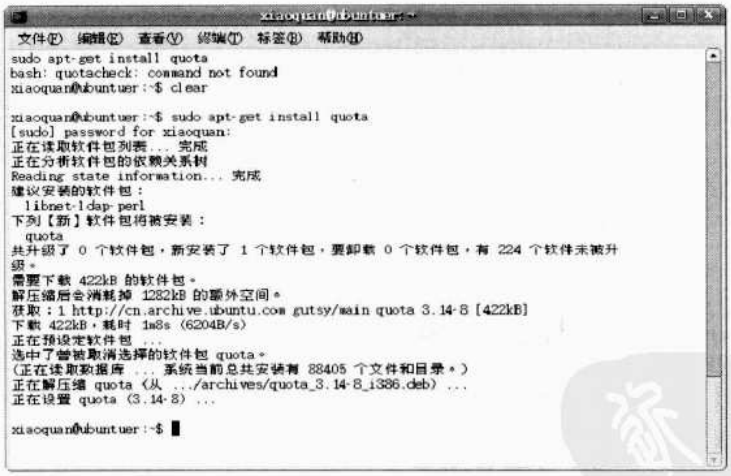


图 15.5 命令行方式安装 quota

15.3 磁盘配额基本指令

在开始进行 quota 的实践之前，需要先了解一下 quota 要使用的指令。quota 使用的指令基本上分为两种，一种是查询功能(quota、quotacheck、quotastats、warnquota、repquota)，另一种则是编辑

quota 的内容(edquota、setquota)。下面讲述这些基本的指令。

15.3.1 quota 指令

quota 指令用于显示磁盘已使用的空间与限制。

语法

```
quota [-quvV][用户名称...]或 quota [-gqvV][群组名称...]
```

参数说明

- -g：列出群组的磁盘空间限制。
- -q：简明列表，只列出超过限制的部分。
- -u：列出用户的磁盘空间限制。
- -v：显示该用户或群组在所有挂入系统的存储设备的空间限制。
- -V：显示版本信息。

在企业级应用中磁盘配额非常重要，普通用户要学会看懂自己的磁盘使用情况。要查询自己的磁盘配额可以使用下面的命令：

```
root@ubuntuer: ~# quota -guvs    <==显示目前执行者（就是 root ）的 quota 值
root@ubuntuer: ~# quota -uvs test <==显示 test 这个用户的 quota 值

# quota user_ztc1azbs
Disk quotas for user user_ztc1azbs (uid 1193):
     Filesystem  blocks   quota   limit   grace   files  quota limit   grace
      /dev/sda2  164236  512000  512000           1626   64000   64000
```

以上显示 ID 号为 1193 的 user_ztc1azbs 账号，文件个数设置为 64 000 个，已经使用了 1626 个，硬盘空间限制设置为 500MB，已经使用了 164MB。一旦磁盘配额要用完时，就需要删除部分文件或向系统管理员要求追加或者购买磁盘配额。

15.3.2 quotacheck 指令

quotacheck 指令用于检查磁盘的使用空间与限制。执行 quotacheck 指令，扫描挂入系统的分区，并在各分区的文件系统根目录下产生 quota.user 和 quota.group 文件，设置用户和群组的磁盘空间限制。该指令主要的目的在扫描某一个磁盘的 quota 空间，它会针对该磁盘进行扫描，并且，由于该磁盘若持续运作时，在扫描的过程中，文件可能会增减，造成 quota 扫描错误的发生，因此，当使用 quotacheck 时，该磁盘将自动被设定成为只读扇区（read-only）。至于扫描完毕之后，扫描所得的磁盘空间结果会写入该扇区最顶端。另外，Linux 也特别强调在使用 quota 的时候，需要特别注意在 reboot 时，须先将 quota 关闭。

语法

```
quotacheck [-adgRuv][文件系统...]
```

参数说明

- -a：扫描/etc/fstab 文件中加入 quota 设置的分区。
- -d：详细显示指令执行过程，便于排错或了解程序执行的情况。
- -g：扫描磁盘空间时，计算每个群组识别码所占用的目录和文件数目。
- -R：排除根目录所在的分区。
- -u：扫描磁盘空间时，计算每个用户识别码所占用的目录和文件数目。
- -v：显示指令执行过程。

范例

针对/home 这个 partition 进行 quota 的规划：

```
root@ubuntuer: ~# quotacheck -uvg /home        <==开始扫描 /home 这一个独立扇区的目录
quotacheck: Scanning /dev/hda3 [/home] done     <==显示 /home 扇区为/dev/hda3
quotacheck: Checked 35 directories and 342 files
                                  <==扫描完毕，有 35 目录和 342 个文件
root@ubuntuer: ~# ls -l /home          <==查看/home 这个目录下产生了两个文件了
-rw-------   1 root     root         7168 May 6 18:37 aquota.group
-rw-------   1 root     root         7168 May 6 18:37 aquota.user
```

在有些版本的 Linux 发行版当中，进行 quotacheck 时，可能会出现以下问题：

```
quotacheck: Cannot get quotafile name for /dev/hda3
```

这可能是 quota 在设计时的小问题，解决的方法有两个，一个是加上-m 参数来强制进行，即运行命令：

```
quotacheck -uvgm
```

还一种方法是，既然 quotacheck 找不到 quotafile，那么就手动将 quotafile 建立起来，然后再重新进行 quotacheck。具体操作方法是：先 touch 一下 aquota.user 和 aquota.group 文件，即使用命令 touch /home/aquota.user 和 touch /home/aquota.group，然后再运行命令 quotacheck -uvg。

15.3.3 edquota 指令

edquota（edit quota）指令用于预设会使用 Vi 来编辑用户或群组的 quota 设置数值。

语法

```
edquota [-p <源用户名称>][-ug][用户或群组名称...]或 edquota [-ug] -t
```

参数说明

- -u：后面接账号名称。可以进入 quota 的编辑画面（Vi）去设定 username 的限制值。
- -g：后面接群组名称。可以进入 quota 的编辑画面（Vi）去设定 groupname 的限制值。
- -t：可以修改宽限时间（就是超过 quota 的 soft limit 值后，还能使用硬盘的宽限期限）。
- -p：复制范本。-p 后跟一个已经存在并且已设定好 quota 的使用者，如-p username-demo，将 username_demo 这个人的 quota 限制值复制给目标用户。

范例 1

设定 dmtsai 这个用户的 quota 限制值。

```
root@ubuntuer: ~# edquota -u test
Disk quotas for user test (uid 501):
Filesystem          blocks       soft       hard     inodes     soft     hard
/dev/hda3             8          0          0          5        0        0
```

vi 修改一下成为：

```
Disk quotas for user test (uid 501):
Filesystem          blocks       soft       hard     inodes     soft     hard
/dev/hda3             8          5000       5000       5        5000     5000
```

示例中有 7 个字段，各个字段的意义说明如下。

- Filesystem
  代表这个 quota 是针对哪一个 partition。以范例 1 的情况来说，指的是/dev/hda3，也就是/disk1 那个目录下的 quota 限制值。
- blocks
  该字段是目前用户 test(uid 501)在/dev/hda3 这个 filesystem(参考上面一个信息)，所消耗的磁盘容量，也就是目前已使用的空间，单位是 KB。这个信息是 quota 程序自己计算出来的，所以请不要修改它！
- soft 与 hard
  该字段是目前的 test 用户在这个 filesystem 之内的 quota 限制值。至于 soft 与 hard 的意思在前一节最后已涉及，soft 代表的是一个“警告”限值，hard 则是一个“不可超过的限值”，soft 与 hard 中间的差值则为宽限的数值。而当 soft 与 hard 数值为 0 的时候，表示“没有限制”的意思，数值的单位仍是 KB。
- inodes
  该字段是目前已用 inode 的状态，也是 quota 自己计算得到的，所以不要去修改它。一般而言，inode 不容易控制。

范例 2

将 dmtsai 的 quota 限制值复制给 vbird1 这个用户。

```
root@ubuntuer: ~# edquota -p dmtsai -u vbird1
```

范例 3

下面是修订宽限时间的范例。

```
root@ubuntuer: ~# edquota -t
Grace period before enforcing soft limits for users:
Time units may be: days, hours, minutes, or seconds
  Filesystem          Block grace period     Inode grace period
  /dev/hdb1                 7days                  7days
```

上面例子中默认的宽限时间是 7 天。

15.3.4 quotaon 指令

quotaon 指令用于执行用户和群组的磁盘空间限制，各分区的文件系统根目录必须有 quota.user 和 quota.group 配置文件。

语法

```
quotaon [-aguv] [文件系统...]
```

参数说明

- -a：启用/etc/fstab 文件中批示的带有磁盘配额的所有分区的空间限制。
- -g：开启群组的磁盘空间限制。
- -u：开启用户的磁盘空间限制。
- -v：显示指令执行过程。

范例 1

启动所有具有 quota 的 filesystem。

```
root@ubuntuer: ~# quotaon -auvg
/dev/hdb1 [/disk2]: group quotas turned on
/dev/hdb1 [/disk2]: user quotas turned on
```

范例 2

仅启动/disk2 中的 user quota 设定值。

```
root@ubuntuer: ~# quotaon -uv /disk2
```

15.3.5 quotaoff 指令

quotaoff 指令用于关闭用户和群组的磁盘空间限制。

语法

```
quotaoff [-aguv][文件系统...]
```

参数说明

- -a：关闭在/etc/fstab 文件中指示带有磁盘配额所有分区的空间限制。
- -g：关闭群组的磁盘空间限制。
- -u：关闭用户的磁盘空间限制。
- -v：显示指令执行过程。

范例

```
root@ubuntuer: ~# quotaoff -a
```

关闭所有的 quotaoff 设定（自动寻找/etc/mtab 的设定）。

15.4 quota 应用实例说明

除了上面提到的 quota 的基本应用，quota 还有很多功能，如限制某一群组所能使用的最大磁盘配额（使用群组限制）：用户可以将主机上的用户分门别类，类似当前很流行的付费与免付费会员制的情况，对特定的用户群的使用配额就可以高一些。限制某一用户的最大磁盘配额（使用使用者限制）：在限制了群组之后，用户也可以再继续针对个人来进行限制，使得同一群组之下还可以有更公平的分配。

15.4.1 quota 应用操作实例

准备好测试的环境——用户与群组的建立

下面添加两个用户，quser1 和 quser2，以进行测试。同时将他们加到 qgroup 组中。具体操作命令如下所示：

```
root@ubuntuer: ~# groupadd qgroup
root@ubuntuer: ~# useradd -m -g qgroup quser1
root@ubuntuer: ~# useradd -m -g qgroup quser2
root@ubuntuer: ~# passwd quser1
root@ubuntuer: ~# passwd quser2
```

建立好 filesystem 的 quota 支持

由于 quota 较完整的支持需要在 ext2/ext3 的 Linux 文件系统下才可以启动，所以建议将准备开启 quota 的磁盘启动参数写入 quota 的磁盘设定（/etc/fstab）。以本例而言，要在/disk2 下进行 quota 限制 quser1、quser2 这两个用户。这是因为这个/disk2 是一个独立的扇区，这可以使用 df 来查询。此外，必须特别留意的是，最好不要以根目录亦即/进行 quota，否则容易出问题。另外，不要针对 root 进行 quota。

```
root@ubuntuer: ~# df
Filesystem           1K-blocks      Used Available Use% Mounted on
/dev/hda1              5952252 3193292 2451720  57% /
/dev/hdb1             28267608     77904  26730604 1% /disk2
/dev/hda5              9492644 227252 8775412 3% /disk1
```

说明 /disk2 是独立的 partition，并且设备名为/dev/hdb1，那么就必须要启动/disk2 这个/dev/hdb1 的 quota 文件格式。由于文件格式的设定是写在/etc/fstab 中，所以可以以用 Vi 来编辑它，只要在/etc/fstab 中增加了 usrquota、grpquota 即可。

```
root@ubuntuer: ~# vi /etc/fstab
LABEL=/        /       ext3 defaults                    1 1
LABEL=/disk1 /disk1     ext3 defaults                    1 2
```

```
LABEL=/disk2 /disk2      ext3 defaults, usrquota, grpquota   1 2
/dev/hda3    swap    swap defaults                           0 0
```

注意到所需要设定的那个/disk2的那一行，在第四字段多了usrquota,grpquota。注意，在"defaults, usrquota, grpquota"之间都没有空格。这样就算加入了quota的磁盘格式了。

不过，由于真正的quota在读取的时候是读取/etc/mtab这个文件的，偏偏这一个文件需要重新开机之后才能够以/etc/fstab的新数据进行改写，所以这时需要重启机器（reboot）。当然

对不太喜欢重新开机的人，可以这样做：

```
root@ubuntuer: ~# umount /dev/hdb1
root@ubuntuer: ~# mount -a
root@ubuntuer: ~# grep '/disk2' /etc/mtab
/dev/hdb1 /disk2 ext3 rw, usrquota, grpquota 0 0
                    <==事实上，也可以利用mount的remount功能
root@ubuntuer: ~# mount -o remount /disk2
```

这样就已经成功地将filesystem的quota功能加入了。另外，在这里是以ext3这个磁盘格式来测试quota的。

扫描磁盘的用户使用状况，并产生重要的aquota.group与aquota.user

接下来就要来扫描一下所需要的磁盘到底有没有多余的空间来设定quota，并且将扫描的结果输出到这个磁盘的最顶层（也就是/disk2下），这时需要quotacheck命令了！使用quotacheck就可以轻易地输出所需要的数据，并在/disk2下产生aquota.group与aquota.user这两个文件。

```
root@ubuntuer: ~# quotacheck -avug
quotacheck: Scanning /dev/hdb1 [/disk2] done
quotacheck: Checked 3 directories and 4 files
root@ubuntuer: ~# ll /disk2
-rw-------   1 root root   6144 Sep   6 11:44 aquota.group
-rw-------   1 root root   6144 Sep   6 11:44 aquota.user
```

使用quotacheck可以轻易地将所需要的数据输出，但是很奇怪的是，在某些Linux版本中，不能够以aquota.user（group）来启动quota，这有可能是因为旧版quota的关系，所以就另外做了一个link文件来欺骗quota。

```
root@ubuntuer: ~# cd /disk2
root@ubuntuer: ~# ln -s aquota.user quota.user
root@ubuntuer: ~#ln -s aquota.group quota.group
```

启动quota的限额

接下来就要启动quota了，启动quota的方式也很简单，就是使用quotaon -av即可。

```
root@ubuntuer: ~# quotaon -avug
/dev/hdb1 [/disk2]: group quotas turned on
/dev/hdb1 [/disk2]: user quotas turned on
```

注意 要看到上面有turned on的出现，才是真的成功了。

编辑用户的可使用空间

由于有两个用户要设置，先使用 edquota 设置 quser1。

```
root@ubuntuer: ~# edquota -u quser1
Disk quotas for user quser1 (uid 502):
   Filesystem blocks soft hard inodes soft hard
   /dev/hdb1       0 45000 50000        0    0    0
```

再次强调的是，因为这里/disk2 下没有任何数据，所以，在上面这个表格中，blocks 与 inodes 才会都是 0，如果是使用/home 来进行 quota 设定，那么 blocks/inodes 肯定不会是 0，这要特别留意。

同时将刚才配好的 quser1 的配额信息复制给 quser。

```
[root@linux~]# edquota -p quser 1 quser 2
```

用 edquota 来设定宽限时间

```
root@ubuntuer: ~# edquota -t
Grace period before enforcing soft limits for users:
Time units may be: days, hours, minutes, or seconds
   Filesystem          Block grace period     Inode grace period
   /dev/hdb1                    1days                7days
```

将时间改为 1 天（原本是 7 天），然后查询一下是否真的设定成功了。使用 quota –v 来查询：

```
root@ubuntuer: ~# quota -vu quser1 quser2
Disk quotas for user quser1 (uid 502):
    Filesystem   blocks quota limit grace files quota limit grace
   /dev/hdb1    0 45000 50000             0    0    0
Disk quotas for user quser2 (uid 503):
    Filesystem   blocks quota limit grace files quota limit grace
   /dev/hdb1    0 45000 50000             0    0    0
```

特别注意到，由于用户已使用空间尚未超过 45MB，所以 grace(宽限时间)就不会出现。

编辑群组可使用的空间

```
root@ubuntuer: ~# edquota -g qgroup
Disk quotas for group qgroup (gid 502):
   Filesystem blocks soft hard   inodes soft hard
   /dev/hdb1       0 80000 90000    0    0    0
root@ubuntuer: ~# quota -vg qgroup
Disk quotas for group qgroup (gid 502):
Filesystem   blocks quota limit grace files quota limit grace
/dev/hdb1    0 80000 90000             0    0    0
```

这样就设定好了 group 的 quota。同样的，因为整个群组的总使用量还没有到达 80 000KB，当然，那个 grace 就不会有任何信息显示了。但这个地方倒是有很多朋友问到一个小问题，那就是“为什么两个用户 quser1、quser2 的 soft 与 hard 设定值分别是 45/50MB，而 group 总量（hard）设定仅有 90MB 呢？”也就是说，当某个用户用了 50MB 的量，那另一个最多不就可以使用到 40MB 而已？因为如果是小型的系统，由于用户并不是很多，可以针对每个人来进行 quota 的设定值，所以，只针对 users 来进行设置即可，不需要额外地设定 group 的 quota 设置。

15.4.2 邮件主机的 quota 设定

下面再来看看邮件主机的一个 quota 设置。对于邮件主机的 quota 设置，大概有如下几个过程：

(1) 设定好用户 quota 的所有工作。
(2) 将/var/spool/mail 这个默认文件夹备份到其他目录。
(3) 建立/home/mail，假设这个目录为邮件目录。
(4) 修改/home/mail 这个目录的属性为 775。
(5) 将/home/mail 链接到/var/spool/mail。

具体操作如下：

Step 01 建立并修改/home/mail这个目录。

```
root@ubuntuer: ~# mkdir /home/mail
root@ubuntuer: ~# chown root:mail /home/mail
root@ubuntuer: ~# chmod 775 /home/mail
```

Step 02 备份并移动原来的mail到/home/mail下。

```
root@ubuntuer: ~# cp -r /var/spool/mail /var/spool/mail.back
root@ubuntuer: ~# mv /var/spool/mail /home/mail
root@ubuntuer: ~# rmdir /var/spool/mail
```

Step 03 建立链接。

```
root@ubuntuer: ~# ln -s /home/mail /var/spool/mail
```

15.5 课后练习

1. 什么是磁盘配额？quota 的含义是什么？quota 主要用于哪些方面？
2. 如何在 Ubuntu 中安装 quota？
3. quota 具体有哪些指令？它们的功能分别是什么？
4. 请尝试对一台主机上的用户进行磁盘配额，以合理分配用户磁盘空间。
5. 尝试对邮件服务器进行磁盘配额。

Chapter16 设备管理

在 Ubuntu 系统中，对于硬件设备的支持可以说比其他任何 Linux 系统都多，使用上也直接方便，大多数情况下只需要把外置设备连接到机器上，Ubuntu 就会自动把设备挂载（mount）上文件系统。在移除设备之前，注意不要忘记把设备卸载（unmount）下来。

16.1 使用 USB 设备

USB（Universal Serial Bus，通用串行总线）是一种应用在 PC 领域的新型接口技术。自从 1995 年 PC 机带有 USB 接口后，1998 年 USB 接口逐步走进大规模实用阶段。

这几年，随着大量支持 USB 的个人电脑的普及，USB 逐步成为 PC 机的标准接口已经是大势所趋。在主机(host)端，最新推出的 PC 机几乎 100%支持 USB；而在外设(device)端，使用 USB 接口的设备也与日俱增，例如数码相机、扫描仪、游戏杆、磁带和软驱、图像设备、打印机、键盘、鼠标等。

USB 设备之所以会被大量应用，主要具有以下优点：

- 可以热插拔，告别“并口和串口先关机，将电缆接上，再开机”的动作。
- 系统总线供电，低功率设备无需外接电源，采用低功耗设备，并可提供 5V/500mA 电源。
- 支持设备众多，支持多种设备类，例如鼠标，键盘，打印机等。
- 扩展容易，可以连接多个设备，最多可扩 127 个。
- 高速数据传输，USB 1.1 是 12 Mbps，USB 2.0 高达 480 Mbs。
- 方便的设备互连，USB OTG 支持点对点通信，例如数码相机和打印机直接互联，无需 PC。

Ubuntu Linux 对 USB 有着很好的支持，用户可以通过使用命令 lsusb 来查看系统内所有的 USB 设备(设备供应商和设备的 ID)。对于价格便宜而容量大的 USB 移动硬盘，用户只把 USB 接口接到主机上，Ubuntu 将会自动把移动硬盘挂接到文件系统上，并且在桌面上显示出相应的硬盘目录，如图 16.1 所示。

双击桌面上的移动硬盘盘符，就可以使用 Nautilis 文件管理器来查看硬盘内容。除了移动硬盘这个常用的 USB 设备外，Ubuntu Linux 还支持不同形式的 USB 设备，可以通过执行【系统】|【首选行】|【可移动驱动器和介质】命令来查看 USB 设备的设置，如图 16.2 所示。

USB 设备使用结束后，还需要做一件事情，那就是通知系统：USB 设备将要被移除，以免系统在读写硬盘过程中拔除 USB 设备，造成数据丢失或损坏。如何通知系统将要拔除设备呢？很简单。右键单击 USB 设备符号，在弹出的菜单中选择【卸载】命令。

在使用 USB 设备的时候，还有一个需要注意的事项：当发现 USB 设备不能显示在桌面的时候，请注意检查用户权限。通过执行【系统】|【系统管理】|【用户和组】命令打开【用户设置】对话框，如图 16.3 所示。

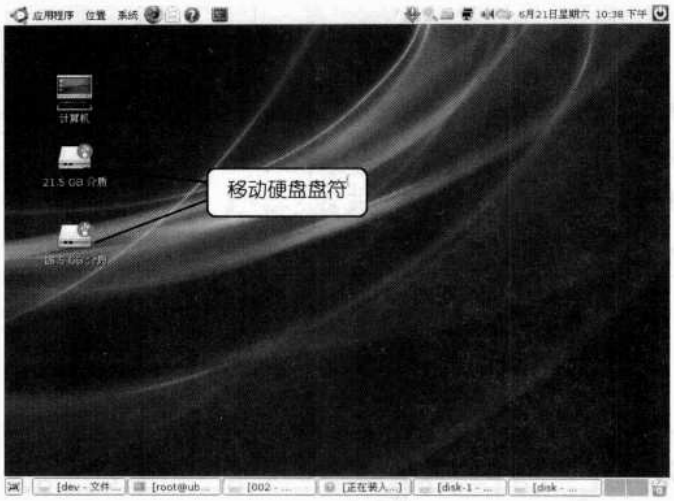


图 16.1 移动硬盘盘符

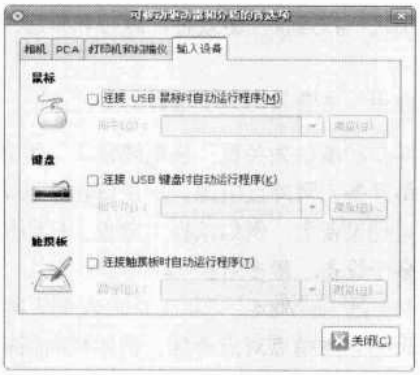


图 16.2 USB 设备设置

图 16.3 【用户设置】对话框

在弹出的【用户设置】对话框中中，选择需要操作的用户，单击【属性】按钮，选择【用户权限】标签页，确保【自动访问外部存储设备】被选中，如图 16.4 所示。

图 16.4　用户权限中的自动访问外部存储设备

16.2　CD/DVD 刻录

Brasero 是一款为 GNOME 桌面开发的 CD/DVD 刻录软件。Brasero 作为 Nautilus CD Burner 和 Serpentine 的补充和替代，Ubuntu 8.04 也将其整合进来了。虽然 Brasero 在功能上还是很简单，但对于日常的应用来说，Brasero 这款小而五脏俱全的刻录软件完全能胜任。

通过执行【应用程序】|【影音】|【Brasero 光盘刻录】命令打开 Brasero，如图 16.5 即为 Brasero 的界面，非常简洁。

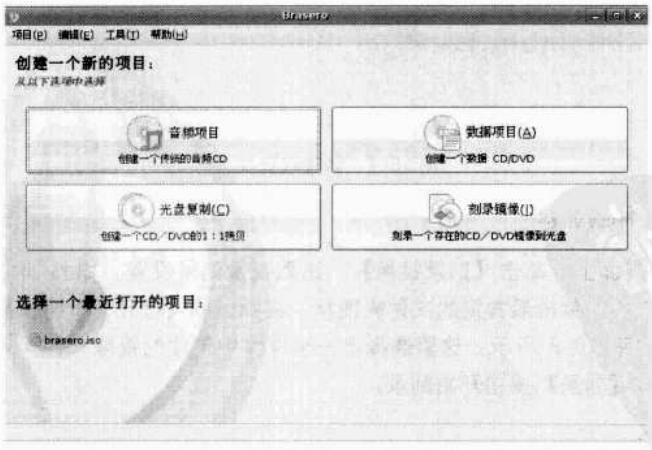


图 16.5　Brasero 刻录软件界面

从上图可以看到，Brasero 可用来刻录音乐项目光碟、数据盘等，具体操作方法可以按提示进行。

下面来简单介绍一下在 Ubuntu 中用 Brasero 创建镜像文件，并把它刻录到 CD 上。

创建镜像文件

单击【数据项目】按钮，打开创建 CD/DVD 项目的界面，如图 16.6 所示。

图 16.6 创建数据项目

在上图中，从左侧文件系统中选择需要刻录的文件或目录，然后单击【添加】按钮，把目标文件转移到右侧的窗口中。如果已经知道光盘的格式，可以在图中所示位置选择光盘的型号以及容量大小。最后单击【刻录】按钮，进入如图 16.7 所示的界面，这个时候 Brasero 开始创建镜像文件了。

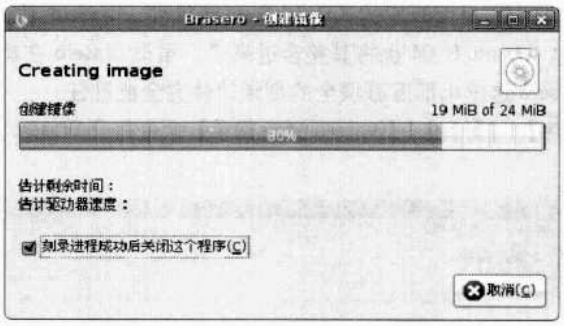


图 16.7 创建镜像文件

刻录镜像文件到光盘

在 Brasero 软件界面上，单击【刻录镜像】，进入镜像刻录设置，如图 16.8 所示。

在【路径】选项中，单击最右侧的文件夹图标，在弹出的【打开一个镜像】对话框中选择需要刻录的镜像文件，如图 16.9 所示，这里选择上一步操作中创建的镜像文件，单击【打开】按钮，完成设置。最后单击【刻录】按钮开始刻录。

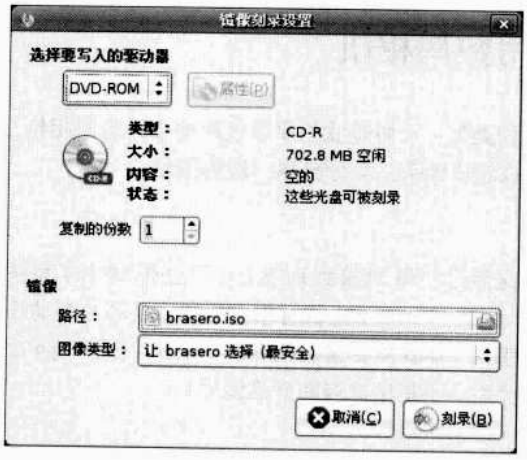


图 16.8　镜像刻录设置

图 16.9　选择镜像文件

16.3　使用软驱

在 Ubuntu 下使用软驱非常简单，只需要把软盘插入软驱中，然后执行【位置】|【软盘驱动器】命令，系统将自动挂载软盘内容。在软盘使用完后，右键单击【软盘驱动器】，在弹出的菜单中选择【卸载】，通知软盘驱动器完成操作，准备释放软盘。

16.4 使用数码相机

数码相机一般分为两种类型：一种是我们平常使用的手持数码相机，它拥有比较高的分辨率；另外一种就是所谓的网络视频摄像头，其分辨率一般都很低。

摄像头

摄像头通常情况下都是通过 USB 连接到系统上。不过在 Ubuntu 系统中，并不是所有的摄像头都适用，这个主要是看在 Ubuntu Linux 下是否有相应的摄像头芯片驱动程序。如果用户希望为自己的 Ubuntu 系统配置一个摄像头，那就需要提前做好以下工作：在 google 或百度搜索一下当前支持的芯片类型（注意：是芯片类型，不是生产商和产品型号）。

数码相机

非常幸运的是，对于数码相机来说，Ubuntu Linux 有着良好的支持，也可以说，数码相机都有着非常好的统一性，那就是现在的数码相机都是以 USB 的方式连接到 Ubuntu 上，系统自动把相机的存储卡当作一个移动硬盘来处理，同时会根据系统设定做相机图片的导入处理。执行【系统】|【首选行】|【可移动驱动器和介质】命令，然后单击进入【相机】标签页，设定数码相机在连接后的命令，使用 F-spot 来导入图片，如图 16.10 所示，这个设置的意思是说，当用户把数码相机连接上系统后，系统将自动启动 F-spot 软件来处理图片。

图 16.10 数码相机设置图片导入

当用户把相机连接到系统时，一个设备图标自动显示在桌面上，这时系统自动弹出一个窗口，开始扫描相机的存储卡，如图 16.11 所示。

扫描结束后，在窗口下部单击【目标位置】，在弹出的文件目录选择窗口中选择存放图片的路径，然后单击【复制】按钮，系统开始复制图片到本地硬盘上，如图 16.12 所示。

图 16.11　自动扫描相机存储卡

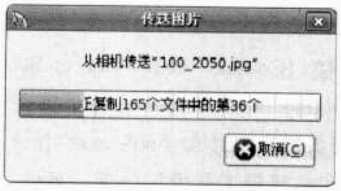


图 16.12　传送图片

图片传送完毕后，F-Spot 自动打开图片浏览窗口，可以通过 F-Spot 顶部的时间滚动条来查看图片或者双击图片进行编辑。如图 16.13 所示。

图 16.13　F-Spot 管理图片

F-Spot 是 Ubuntu 默认的数码相机图片管理工具，如果希望导入更多的图片，单击工具栏的【导入】按钮，从弹出的窗口中选择导入源，还可以给图片添加附加标签（tag），如图 16.14 所示。

图 16.14 导入图片到 F-Spot

16.5 使用打印机

Ubuntu 使用通用 Unix 打印系统（Common Unix Printing System，CUPS）处理打印事务。CUPS 使用 Internet 打印协议（Internet Printing Protocol，IPP）作为管理打印作业和队列的基础，也通过简化的功能支持行式打印机服务、服务器消息块和 AppSocket（a.k.a. JetDirect）协议。CUPS 打印机的配置和管理可以通过 GNOME 打印机管理工具进行处理。另外，CUPS 还提供了类似老的伯克利和 SystemV 的 lpq、lpstat 等打印命令。

在 Ubuntu 上安装打印机并不像想象中那样麻烦。可以通过执行【系统】|【系统管理】|【打印】命令调出打印机配置窗口，如图 16.15 所示。

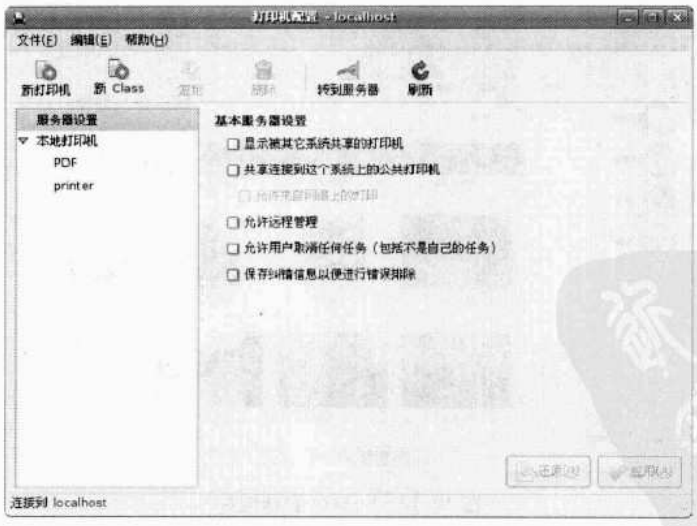


图 16.15 打印机配置

首先可以安装一个 pdf 虚拟打印机，就是将打印内容生成一个 pdf 文件，生成的 pdf 文件保存

在用户文件夹下的 pdf 文件夹下。单击【新打印机】按钮，在连接类型里选择 Print into PDF file，随后会出现一个选择打印机厂商驱动程序的对话框，这时选择 Generic，下一步选择 PDF file generator。下一步就会出现一个打印机名称设置，单击应用后就新增一台打印机。

添加普通针式打印机。一般是在 lpt 接口上，在连接类型中选择“LPT#1”。然后选择自己的打印机对应的厂商和打印机类型。如果不知道自己的打印机型号，可以选择同一厂商的通用 24 针式打印机驱动程序，当然如果是 9 针式打印机要选择 9 针式打印机驱动程序。一般情况下，系统默认的针式打印机的分辨率很低，这样打印出来的内容发虚，可以在随后的【打印机选项】中的 resolution 项进行调整，一般是 180dpi。

当定义了一个本地打印机（例如使用打印管理工具），按照服务器上 CUPS 配置文件的指示，这个“打印服务器”主机自动向网络发布该打印机。远程 Ubuntu 客户端主机就可以看到并使用安装在该服务器上的打印机。网络打印机自动出现在客户端的打印管理工具上。如果 CUPS 启动并配置正确，它就会出现，如果在打印服务器或本地计算机上停止 CUPS，它就会消失。

下面来介绍如何使用 CUPS 一步一步添加打印机，从上图打印机配置窗口中，单击【新打印机】按钮，打开【新打印机】窗口，如图 16.16 所示。

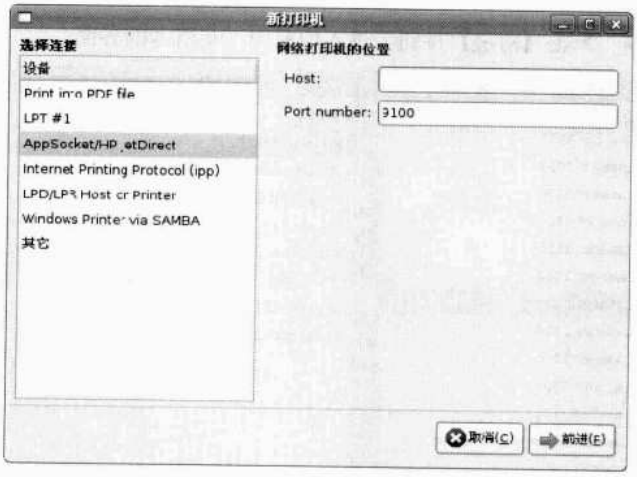


图 16.16 添加打印机步骤一

在上图中，选择打印机的型号，假设使用 HP 打印机，选择 AppSocket/HP JetDirect，然后输入网络打印机的位置，包括主机名称，如果是本地打印机，可以输入 localhost，然后就是端口号（Port number），默认情况下是 9100，设置完毕，单击【前进】按钮，向导程序自动开始检测打印机，如果打印机能够检测出来，系统会自动选中相应的打印机型号，反之，如果向导程序没有选中驱动程序，就需要手动选择打印机程序，如图 16.17 所示。

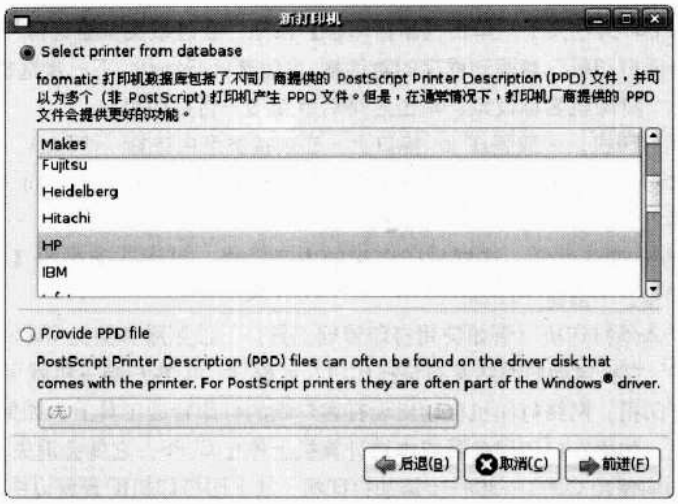


图 16.17 添加打印机步骤二—选择打印机厂商

选择打印机厂商，单击【前进】按钮，进入如图 16.18 所示的界面。

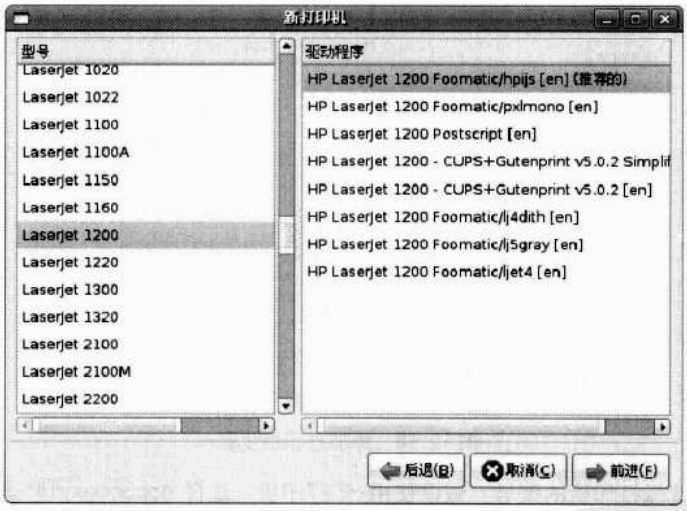


图 16.18 添加打印机步骤三—选择打印机型号及驱动程序

选择打印机型号及驱动程序，默认情况下，驱动程序最好使用“推荐的”驱动程序，因为这个是最常用，而且经过严格测试的驱动程序，出现问题的机会很少。再接着单击【前进】按钮，进入如图 16.19 所示的窗口。

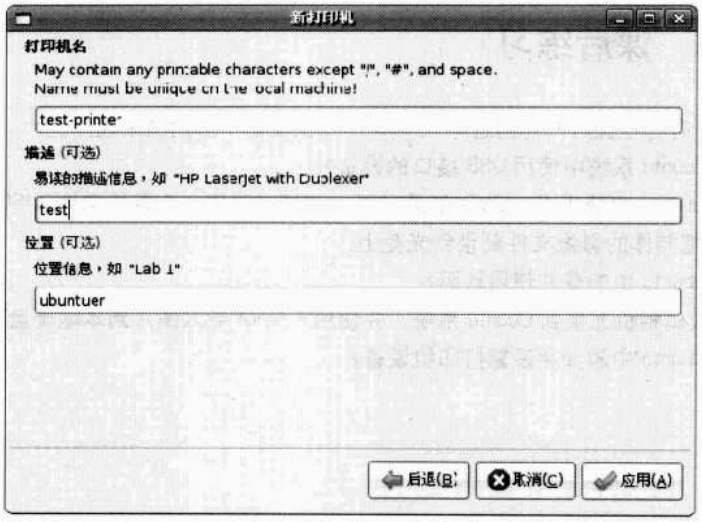


图 16.19　添加打印机步骤四—打印机描述

在上面的窗口中输入打印机的名称，以及打印机的描述、位置。最后单击【应用】按钮，完成添加打印机。这时候将在打印机管理窗口中看见新添加的打印机，如图 16.20 所示，单击【打印测试页】，测试打印机能否正常工作。

图 16.20　新加的打印机 test—printer

如果在使用添加打印机的向导中，没有正确地配置打印机的信息，可以通过上面的窗口修改打印机的设置。或者从另外一个角度说，通过上面的设置，不断优化打印机的配置属性，以满足使用要求。

16.6 课后练习

1．USB 设备有哪些优势？为何近年来 USB 接口在设备中应用越来越广泛？

2．如何在 Ubuntu 系统中使用 USB 接口的设备？

3．如何在 Ubuntu 中使用 CD/DVD 刻录光驱？请尝试制作一个影像文件（ISO 文件），如果有刻录光驱，可以把制作的影像文件刻录到光盘上。

4．如何在 Ubuntu 中加载并使用软驱？

5．如何将数码相机连接到 Ubuntu 系统，并使用 F-shot 导入图片到本地硬盘上？

6．如何在 Ubuntu 中添加并设置打印机设备？

Chapter17 进程管理及作业调度

Ubuntu Linux 作为多用户多任务的操作系统，可同时容纳多个用户同时运行多个进程。正在执行的一个或多个相关进程称为一个作业。通过使用作业调度，用户可以同时运行多个作业，并在需要时在作业之间进行切换。本章详细介绍进程管理及作业控制的命令，包括启动进程、查看进程、终结进程和调度作业的命令。

17.1 进程及作业的概念

Ubuntu Linux 是一个多用户多任务的操作系统。多用户是指多个用户可以在同一时间使用计算机系统。多任务是指可以同时执行多个任务，它可以在还未执行完一个任务时又执行另一个任务。一般来说，一个系统只有一个 CPU 和一个主内存，但一个系统可能有多个存储磁盘和多个输入/输出设备，操作系统管理这些资源并在多个用户间共享资源。当用户提出一个请求时，给其造成一种假象，好像系统只被他独自占用。而实际上操作系统监控着一个等待执行的任务队列，这些任务包括用户程序、操作系统任务、邮件和打印作业等。操作系统根据每个任务的优先级为每个任务分配合适的时间片，每个时间片大约都有零点几秒，虽然看起来很短，但实际上已经足够计算机完成成千上万的指令集。每个任务都会被系统运行一段时间，然后挂起，系统转而处理其他任务，过一段时间以后再回来处理这个任务，直到某个任务完成后从任务队列中去除。

Linux 系统上所有运行的东西都可以称之为进程。每个用户任务、每个系统管理守护进程，都可以称之为进程。Linux 用分时管理方法使所有的任务共同分享系统资源。我们讨论进程的时候，不会去关心这些进程究竟是如何分配的，或者是内核如何管理分配时间片的，我们所关心的是如何去控制这些进程，让它们能够很好地为用户服务。

进程的一个比较正式的定义是：在自身的虚拟地址空间运行的一个单独的程序。进程与程序是有区别的，进程不是程序，虽然它由程序产生。程序只是一个静态的指令集合，它不占用系统的运行资源；而进程是一个随时都可能发生变化的、动态的、使用系统运行资源的程序。而且一个程序可以启动多个进程。

Linux 操作系统包括三种不同类型的进程，每种进程都有自己的特点和属性。

- 交互进程：由一个 Shell 启动的进程。交互进程既可以在前台运行，也可以在后台运行。
- 批处理进程：这种进程和终端没有联系，是一个进程序列。
- 守护进程：Linux 系统启动时启动的进程，并在后台运行。

上述三种进程各有各的作用，使用场合也有所不同。

下面介绍一下作业的概念。作业控制是许多 Shell（包括 bash 和 tcsh）的一个特性，使用户能在多个独立作业间进行切换。进程和作业的概念也有区别，一个正在执行的进程称为一个作业，而且作业可以包含一个或多个进程，尤其是当使用了管道和复位向命令。例如 ps –ef | grep java 这个作业就启动了两个进程。在大多数情况下，用户在同一时间只运行一个作业，即它们最后向 Shell

输入的命令。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，但是这会有什么用途呢？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。其他事情做完以后，用户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，在下面的章节中将作更详细的介绍。

17.2 前后台工作管理

每个进程都可以两种方式存在：前台进程（Foreground）和后台进程（Background）。前台也就是用户当前窗口看见正在执行的进程，而后台进程则是当前窗口看不见，但是在系统中实际运行的程序。如果用户使用命令终端来进行工作，而又不喜欢同时开启多个命令终端窗口，这个时候就需要使用到前后台工作管理的一些命令，而这些命令是需要同时使用的，如&，Ctrl+z，bg，fg，jobs，与 kill 等，下面来介绍这些命令。

17.2.1 &符号

&符号用于将屏幕中的命令（command）在后台执行。不过，由于是在后台执行的，所以该程序的输出并不会显示在终端屏幕上。另外，如何使该程序重新在屏幕上面执行呢？使用 fg 即可。

语法

```
user@ubuntuer:~$ command &
```

例子

```
user@ubuntuer:~$ find / -name testing & <==将该执行程序放到后台执行
user@ubuntuer:~$ fg                     <==将该程序放回前台执行
```

17.2.2 Ctrl+z

Ctrl+z 用于将一个正在前台执行的命令放到后台，并且暂停。

语法

```
user@ubuntuer:~$ command
user@ubuntuer:~$ ^Z          <==按下【Ctrl+z】组合键
```

例子

如果你正在进行 vi，而且是在编辑一个重要数据文件，但是偏偏这个时候你需要去处理别的程序，因此需要退出 vi，不过，你并不想这个时候存储退出 vi，那么该如何是好呢？那就将编辑文件程序放到后台去运行。下面通过例子来说明，本例中当用户在执行编辑 /home/user/.bashrc 这个文

件时，想要暂时离开，那么就直接在 vi 的“一般模式”当中按 Ctrl+z 组合键，那么系统就会告诉你工作项目[1] 在后台当中，而且状态为 Stopped 即停止的状态。并且会离开 vi 进入到命令行当中，等待用户输入下一个命令。这个命令在日常使用中相当有用。当然，还需要使用 jobs 配合 bg 或 fg 命令来切换前后台工作。

```
user@ubuntuer:~$ vi .bashrc
^Z                                  <==在 vi 的一般模式中按 Ctrl+z 组合键
[1]+  Stopped                      vi .bashrc     <==这里会显示将 vi 编辑进程放到后台当中
user@ubuntuer:~$      <==回到命令提示符，等待用户输入下一个命令
```

17.2.3 jobs 指令

jobs 指令用于查看当前有多少在后台运行的命令。jobs 命令执行的结果中，+表示一个当前的作业；-表示一个当前作业之后的一个作业；

jobs -l 选项可显示所有任务的 PID，jobs 的状态可以是 running、stopped 或 Terminated。如果任务被终止（kill）了，Shell 从当前的 Shell 环境已知的列表中删除任务的进程标识，因此 jobs 命令显示的是当前 Shell 环境中后台正在运行或者被挂起的任务信息。

语法

```
user@ubuntuer:~$ jobs
```

例子

```
user@ubuntuer:~$ vi .bashrc
^Z                                  <==在 vi 当中的一般模式中键入【Ctrl+z】组合键
[1]+  Stopped                       vi .bashrc     <==这里会显示将数据放到后台当中了
user@ubuntuer:~$ jobs
[1]+  Stopped                      vi .bashrc     <==显示有一个作业在后台下，状态为停止
```

从上例看出，使用 jobs 可以知道目前后台中作业项目有 vi.bashrc 这一项。中括号（[]）里面的数字 1 就是作业的编号。

17.2.4 fg 与 bg 指令

fg 指令将后台中的命令调至前台继续运行，如果后台中有多个命令，可以用 fg %jobnumber 将选中的命令调出。%jobnumber 是通过 jobs 命令查到的后台正在执行的命令的序号（不是 pid）。

bg 指令让一个在后台暂停的命令继续执行。如果后台中有多个命令，可以用 bg %jobnumber 将选中的命令调出，%jobnumber 是通过 jobs 命令查到的后台正在执行的命令的序号（不是 pid）。

语法

```
user@ubuntuer:~$ fg %number
user@ubuntuer:~$ bg %number
```

参数说明

- %：后面接数字，表示 jobs 的作业编号。
- number：作业编号。

例子

```
user@ubuntuer:~$ find / -name test
^Z
[1]+  Stopped                 find / -name testing
user@ubuntuer:~$ vi .bashrc
^Z
[2]+  Stopped                 vi .bashrc    <==这里会显示将程序放到后台当中了!
user@ubuntuer:~$ jobs
[1]-  Stopped                 find / -name testing
[2]+  Stopped                 vi .bashrc
user@ubuntuer:~$ bg %1
user@ubuntuer:~$ jobs
[1]-  Running                 find / -name testing &
[2]+  Stopped                 vi .bashrc
user@ubuntuer:~$ fg %2
进入vi 编辑界面
```

17.2.5 kill 指令

语法

```
user@ubuntuer:~$ kill -sigal %number
```

参数说明

- %number：作业编号，可使用 jobs 查询。
- Signal：
  - -1：重新读取一次配置文件。
  - -2：中断该进程，类似 Ctrl+c 键来中断一个工作。
  - -9：立刻终止一个进程，不论该进程是否为死进程。
  - -15：停止一个程序（这是默认值）。

例子

```
user@ubuntuer:~$ jobs
[1]+  Stopped                 vi .bashrc
user@ubuntuer:~$ kill -9 %1
```

说明

如果想要直接终止后台程序中的程序，可以直接输入 kill。但是由于默认情况下只是把程序停止而已，不见得一定可以将该程序清除干净，因此需要送出一个讯号，告诉系统必须终止该程序，

这个时候就使用 kill –9，–9 有强制停止的意思，可以终止死进程。

17.3 进程资源管理

进程资源管理对于系统管理员来说是很重要的一部分内容，想必读者一定听说过木马程序，所谓的木马程序就是通过潜入用户的主机系统，以种种隐蔽的方式在系统启动时自动在后台执行的程序。它以“里应外合”的工作方式，用服务器/客户端的通信手段，当用户上网时控制主机，盗取信息，并可利用主机攻击远程目标而不必暴露身份等。因此良好的管理进程的习惯，也是系统管理员必须要做好的功课。那什么是程序？说白了，程序就是在执行或者启动一个事件的时候，系统会发给它的一个执行号，换句话说，当启动了一个命令或者 Shell 的时候，系统就会给这个事件或者 Shell 一个代码，而如果有任何其他服务要使用目前的进程资源时，就自动地来了解一下用户的 PID。另外，还需要说明一下父进程（PPID）与子进程（child process）。子进程是由父进程执行而创建的一个进程，当子进程死掉时，父进程通常不会被影响，但是当父进程死掉时，其所有子进程将一并结束。所以在终止进程的时候，如果发现该进程还存在子系统中，那你就要注意终止的是否为子进程。

在接下来的篇幅中我们将介绍在 Ubuntu 下使用系统监视器或命令来观察系统进程工作状态和资源使用状况。

17.3.1 系统监视器

Ubuntu 提供了一个图形化的进程管理工具，通过执行【系统】|【系统管理】|【系统监视器】命令打开系统监视器，单击【进程】标签页，如图 17.1 所示。在监视器默认状况下，可以查看当前每个进程的如下信息：进程名、当前状态、CPU 使用率、优先级、进程 ID 和内存使用率。此外，在窗口上部还显示了最近一分钟内、五分钟内，以及十五分钟内的系统平均负载。

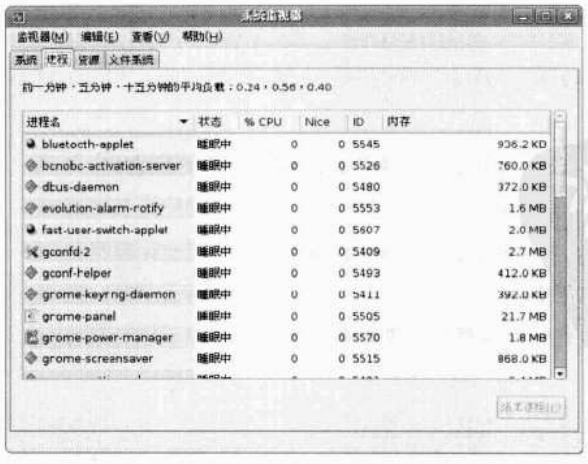


图 17.1 系统监视器–进程管理

在【系统监视器】窗口中，单击【资源】标签页，可以查看CPU、内存和网络资源的最近使用状况，如图17.2所示。

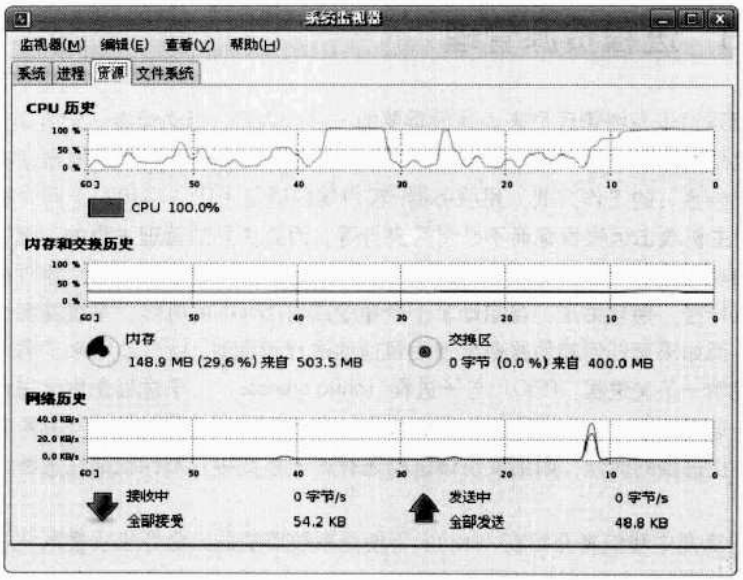


图17.2 系统监视器—资源管理

17.3.2 进程管理指令

ps指令

Linux的ps指令和top指令都是用来监视系统进程和资源使用情况的有用命令，ps命令更为常用，所以掌握ps命令是很有必要的。

语法

```
user@ubuntuer:~$ ps -aux
```

参数说明

ps的参数非常多，在此仅列出几个常用的参数并大概介绍含义。

- -A：列出所有的行程。
- -w：显示加宽，从而可以显示较多的信息。
- -au：显示较详细的信息。
- -aux：显示所有包含其他用户的进程。

例子

```
user@ubuntuer:~$ ps -aux
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root         1  0.0  0.1   1384   468 ?        Ss   07:30   0:05 init [3]
```

```
root         2  0.0  0.0     0     0 ?        S<   07:30   0:00 [keventd]
root         3  0.0  0.0     0     0 ?        S<   07:30   0:00 [ksoftirqd_CPU0]
root         4  0.0  0.0     0     0 ?        S<   07:30   0:11 [kswapd]
root         5  0.0  0.0     0     0 ?        S<   07:30   0:00 [bdflush]
root         6  0.0  0.0     0     0 ?        S<   07:30   0:00 [kupdated]
root       130  0.0  0.0     0     0 ?        S<   07:30   0:00 [kjournald]
root       131  0.0  0.0     0     0 ?        S<   07:30   0:01 [kjournald]
root       132  0.0  0.0     0     0 ?        S<   07:30   0:03 [kjournald]
root       133  0.0  0.0     0     0 ?        S<   07:30   0:12 [kjournald]
root       482  0.0  0.2  1444   528 ?        S    07:30   0:03 syslogd -m 0
........
```

⚙说明

Ps 指令用来查询目前主机环境中在后台执行的相关程序，通常使用 ps –aux 这个命令参数来列出所有的信息以供自己检查程序的问题。在上面的程序列表当中，各项说明如下。

- USER：说明该程序是哪一个用户使用的。
- PID：该进程的代号。
- %CPU：代表该进程使用了多少 CPU 资源。
- %MEM：代表该进程使用了多少内存。
- VSZ，RSS：占去的内存的大小（byte）。
- TTY：是否为登入者执行的程序？若为 tty1–tty6，则为本机登入者，若为 pts/??，则为远程登入者执行的程序。
- STAT：该程序的状态，R 为可执行的；S 为该程序正在睡眠中，即没有执行；T 为正在侦测或者是停止了；Z 为僵尸程序，就是 zombie 死掉的进程，需要以 kill 终止。
- START：该进程开始的日期。
- TIME：该进程运行了多长时间。
- COMMAND：该进程的执行命令。

这是一个很有用的命令，尤其是在检查系统的状态时。不过，这个命令也是入侵者最喜欢修改的命令，因为他可以写一个脚本来骗 root 用户，让某些木马程序没有办法显示出来。此外，PID 是很重要的信息，因为在后面的 kill 就是通过 PID 来进行进程的删除动作的。

top 指令

⚙语法

```
user@ubuntuer:~$ top
```

⚙参数说明

- A：以 age 即执行的先后顺序进行排序。
- c：显示整个命令行而不只是显示命令名。
- d：指定每两次屏幕信息刷新之间的时间间隔。当然用户可以使用 s 交互命令来改变它。
- T：由启动的时间排序。
- M：以所占的 memory 的大小排序。
- P：以所耗用的 CPU 资源排序。

- Q：该选项将使 top 没有任何延迟地进行刷新。如果调用程序有超级用户权限，那么 top 将以尽可能高的优先级运行。
- p：通过指定监控进程 ID 来仅仅监控某个进程的状态。
- S：指定累计模式。
- s：使 top 命令在安全模式中运行。这将去除交互命令所带来的潜在危险。
- i：使 top 不显示任何闲置或者僵死进程。

在 top 运行期间，还有相关的交互参数。从使用角度来看，熟练地掌握这些命令比掌握选项还重要一些。这些命令都是单字母的，如果在命令行选项中使用了 s 选项，则其中一些命令可能会被屏蔽掉。

- Ctrl+L：清除并且重写屏幕。
- h 或者?：显示帮助信息，给出一些简短的命令总结说明。
- k：终止一个进程。系统将提示用户输入需要终止的进程 PID，以及需要发送给该进程什么样的信号。一般的终止进程可以使用 15 信号；如果不能正常结束那就使用信号 9 强制结束该进程。默认值是信号 15。在安全模式中此命令被屏蔽。
- i：忽略闲置和僵死进程。这是一个开关式命令。
- q：退出程序。
- r：重新安排一个进程的优先级别。系统提示用户输入需要改变的进程 PID 以及需要设置的进程优先级值。输入一个正值将使优先级降低，反之则可以使该进程拥有更高的优先权。默认值是 10。
- S：切换到累计模式。
- s：改变两次刷新之间的延迟时间。系统将提示用户输入新的时间，单位为 s。如果有小数，就换算成 m s。输入 0 值则系统将不断刷新。默认值是 5 s。需要注意的是，如果设置太小的时间，很可能会引起不断刷新，从而根本来不及看清显示的情况，而且系统负载也会大大增加。
- f 或者 F：从当前显示中添加或者删除项目。
- o 或者 O：改变显示项目的顺序。
- l：切换显示平均负载和启动时间信息。
- m：切换显示内存信息。
- t：切换显示进程和 CPU 状态信息。
- c：切换显示命令名称和完整命令行。
- M：根据驻留内存大小进行排序。
- P：根据 CPU 使用百分比大小进行排序。
- T：根据时间/累计时间进行排序。
- W：将当前设置写入~/.toprc 文件。这是写 top 配置文件的推荐方法。

例子

```
user@ubuntuer:~$
top - 08:10:07 up 39 min, 2 users, load average: 0.59, 0.51, 0.49
Tasks: 116 total,  2 running, 113 sleeping,  0 stopped,  1 zombie
```

```
Cpu(s): 40.0%us, 10.0%sy, 0.0%ni, 49.3%id, 0.0%wa, 0.7%hi, 0.0%si, 0.0%st
Mem:   515580k total,  506852k used,    8728k free,  101560k buffers
Swap:  409616k total,       0k used,  409616k free,  232812k cached

  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
 5183 root      20   0 39888  14m 7068 S 38.0  2.9  10:16.39 Xorg
 5673 user      20   0 68612  19m  14m S  7.9  3.8   2:27.55 gnome-system-mo
 6251 user      20   0  102m  19m  11m R  1.3  3.9   0:01.78 gnome-terminal
 2545 root      15-5    0    0    0 S  0.7  0.0   0:00.52 kjournald
 5063 root      20   0  3420 1156 1004 S  0.7  0.2   0:01.90 hald-addon-stor
 5515 user      20   0 15744 4780 3816 S  0.7  0.9   0:02.48 gnome-screensav
 5563 user      20   0 47012  11m 9848 S  0.7  2.3   0:02.48 nm-applet
 6274 user      20   0  2308 1112  852 R  0.7  0.2   0:00.06 top
    1 root      20   0  2844 1688  544 S  0.0  0.3   0:01.98 init
    2 root      15-5    0    0    0 S  0.0  0.0   0:00.02 kthreadd
    3 root      RT-5    0    0    0 S  0.0  0.0   0:00.00 migration/0
    4 root      15-5    0    0    0 S  0.0  0.0   0:00.22 ksoftirqd/0
    5 root      RT-5    0    0    0 S  0.0  0.0   0:00.00 watchdog/0
    6 root      15-5    0    0    0 S  0.0  0.0   0:00.26 events/0
    7 root      15-5    0    0    0 S  0.0  0.0   0:00.06 khelper
   41 root      15-5    0    0    0 S  0.0  0.0   0:00.80 kblockd/0
   44 root      15-5    0    0    0 S  0.0  0.0   0:00.00 kacpid
   45 root      15-5    0    0    0 S  0.0  0.0   0:00.00 kacpi_notify
  110 root      15-5    0    0    0 S  0.0  0.0   0:00.04 kseriod
  144 root      20   0    0    0    0 S  0.0  0.0   0:00.28 pdflush
  145 root      20   0    0    0    0 S  0.0  0.0   0:00.12 pdflush
  146 root      15-5    0    0    0 S  0.0  0.0   0:00.10 kswapd0
  187 root      15-5    0    0    0 S  0.0  0.0   0:00.00 aio/0
```

⚙说明

ps 指令是一个不错的管理工具，但是 ps 毕竟不是动态的，而使用 top 指令可以以动态的方式（默认每五秒钟更新一次）来检查进程的运行状况。top 命令的功能非常丰富，如果读者真的需要深入了解，那就使用 man 命令来查看 top 的相关资料。

free

free 指令用来查看当前内存的使用情况。

⚙语法

```
user@ubuntuer:~$ free
```

⚙参数说明

- -k：以 KB 来显示内存。
- -m：以 MB 来显示内存。

⚙例子

```
user@ubuntuer:~$ free
             total       used       free     shared    buffers     cached
```

```
Mem:       4149156    4130412      18744          0      13220    2720160
-/+ buffers/cache:    1397032    2752124
Swap:      6289408        144    6289264
```

⚙说明

- 第 1 行
  - ➢ total：内存总数，4 149 156。
  - ➢ used：已使用的内存数，4 130 412。
  - ➢ free：空闲的内存数，18 744。
  - ➢ shared：当前已废弃不用的内存数，总是 0。
  - ➢ buffers：Buffer Cache 内存数，13 220。
  - ➢ cached：Page Cache 内存数，2 720 160。
  - ➢关系：total = used + free。
- 第 2 行

  −/+ buffers/cache 的意思相当于：
  - ➢ −buffers/cache 的内存数：1 397 032（等于第 1 行的 used − buffers − cached）。
  - ➢ +buffers/cache 的内存数：2 752 124（等于第 1 行的 free + buffers + cached）。

    可见，−buffers/cache 反映的是被程序实实在在使用的内存，而+buffers/cache 反映的是能够使用的内存总数。
- 第 3 行单独针对交换分区，就不用再说了。

为了提高磁盘存取效率，Linux 做了一些精心的设计，除了对 dentry 进行缓存(用于 VFS，加速文档路径名到 inode 的转换)，还采取了两种主要的 Cache 方式：Buffer Cache 和 Page Cache。前者针对磁盘块读写，后者针对文档 inode 读写。这些 Cache 有效缩短了 I/O 系统调用(比如 read，write，getdents)的时间。

读者如果感兴趣可以进一步参考文档/proc/meminfo。free 命令的结果就是根据其信息生成的。free 命令的源码可从 procps−xxx−.src.rpm 获取，xxx 为版本号，比如 procps−3.2.3−5.3.src.rpm。

kill、killall、pkill 和 xkill 指令

终止一个进程或终止一个正在运行的程序，一般是通过 kill、killall、pkill、xkill 等进行。比如一个程序已进入死锁，但又不能退出，这时就应该考虑应用这些工具。另外应用的场合就是在服务器管理中，在不涉及数据库服务器程序的父进程的停止运行时，也能够用这些工具来终止。为什么数据库服务器的父进程不能用这些工具呢？原因很简单，这些工具在强行终止数据库服务器时，会让数据库产生更多的文档碎片，当碎片达到一定程度的时候，数据库就有崩溃的危险。比如 mysql 服务器最好是按其正常的程序关闭，而不是用 pkill mysqld 或 killall mysqld 这样危险的动作。当然对于占用资源过多的数据库子进程，应该用 kill 来终止。

⚙kill

kill 是和 ps 或 pgrep 命令结合在一起使用的。

- 语法

```
kill [信号代码]    进程 ID
```

- 参数说明

➢ -9：立刻终止一个进程，不论该进程是否为死进程。

➢ -15：停止一个程序（这是默认值）。

- 举例

```
user@ubuntuer: ~$ ps auxf |grep httpd
root 4939 0.0 0.0 5160 708 pts/3 S+ 13:10 0:00 \_ grep httpd
root 4830 0.1 1.3 24232 10272 ? Ss 13:02 0:00 /usr/sbin/httpd
apache 4833 0.0 0.6 24364 4932 ? S 13:02 0:00 \_ /usr/sbin/httpd
apache 4834 0.0 0.6 24364 4928 ? S 13:02 0:00 \_ /usr/sbin/httpd
……
```

- 说明

本例查看 httpd 服务器的进程，也可以用 pgrep -l httpd 来查看。

第二列就是进程 PID 的列，其中 4830 是 httpd 服务器的父进程，4833－4834 的进程都是 4830 的子进程。假如终止父进程 4830 的话，其下的子进程也会跟着死掉。

```
user@ubuntuer: ~$ kill 4840  <==终止 4840 这个进程
user@ubuntuer: ~$ ps -auxf |grep httpd
<==查看一下会有什么结果？是不是 httpd 服务器仍在运行
user@ubuntuer: ~$ kill 4830   <==杀掉 httpd 的父进程
user@ubuntuer: ~$ ps -aux |grep httpd
<==查看 httpd 的其他子进程是否存在，httpd 服务器是否仍在运行
```

假如一个程序已完全死掉，使用 kill 指令不加信号强度是没有办法终止该进程的，最好的办法就是加信号强度 -9，后面要接父进程，比如：

```
user@ubuntuer: ~$ ps aux |grep gaim
beinan 5031 9.0 2.3 104996 17484 ? S 13:23 0:01 gaim
root 5036 0.0 0.0 5160 724 pts/3 S+ 13:24 0:00 grep gaim
或
user@ubuntuer: ~$ pgrep -l gaim
5031 gaim
user@ubuntuer: ~$ kill -9 5031
```

killall

Killall 指令通过程序的名字，直接终止任何进程。killall 也和 ps 或 pgrep 结合使用，比较方便。通过 ps 或 pgrep 来查看哪些程序在运行。

- 语法

```
killall 正在运行的程序名
```

- 举例

```
user@ubuntuer: ~$ pgrep -l gaim
2979 gaim
user@ubuntuer: ~$ killall gaim
```

⚙pkill

pkill 和 killall 应用方法差不多，也是直接终止运行中的程序。假如想终止单个进程，请用 kill 来执行。

- 语法

```
pkill 正在运行的程序名
```

- 举例

```
user@ubuntuer: ~$ pgrep -l gaim
2979 gaim
user@ubuntuer: ~$ pkill gaim
```

⚙xkill

xkill 是在桌面上用来关闭图像界面的指令。比如当 firefox 出现崩溃不能退出时，单击鼠标就能终止 firefox。假如想终止 xkill，就按右键取消。

- 语法

```
user@ubuntuer: ~$xkill
```

uname

⚙语法

```
user@ubuntuer:~$ uname [-apnr]
```

⚙参数说明

- -a：所有的系统信息均列出。
- -p：列出 CPU 信息。
- -n：列出 host name。
- -r：列出 kernel 版本信息。

⚙例子

```
user@ubuntuer:~$ uname -a
Linux ubuntuer 2.6.24-18-generic #1 SMP Wed May 28 20:27:26 UTC
 2008 i686 GUN/Linux
```

⚙说明

使用 uname 命令可以查看主机系统的的核心版本、主机名称、CPU 信息等。另外，如果对于 CPU 有兴趣的话，那么不妨在/proc 目录下看看，如下：

```
user@ubuntuer:~$ more /proc/cpuinfo
processor: 0
vendor_id: GenuineIntel
cpu family: 15
model: 2
model name: Intel® Pentium® 4 CPU 2.40GHz
stepping: 8
cpu MHz: 2398.380
```

```
    cache size: 512 KB
    fdiv_bug: no
    hlt_bug: no
    f00f_bug: no
    coma_bug: no
    fpu    : yes
    fpu_exception: yes
    cpuid level: 2
    wp     : yes
    flags: fpu vme de pse tsc msr mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush
dts acpi mmx fxsr sse sse2 ss up pebs bts sync_rdtsc
    bogomips: 4831.02
    clflush size: 64
```

上面就是主机 CPU 的信息。

17.4 进程优先级

Linux 是个多任务多用户系统，而系统的资源是有限的，通常，同一时间会有多个进程占用 CPU 的资源，那么哪个进程比较重要，就让 CPU 先执行该进程，这就是我们所要讨论的进程优先级问题。

Ubuntu 提供一个图形化的进程优先级调整工具，通过执行【系统】|【系统管理】|【系统监视器】命令打开系统监视器，单击【进程】标签页，右键单击选中欲修改的进程项，如图 17.3 所示。

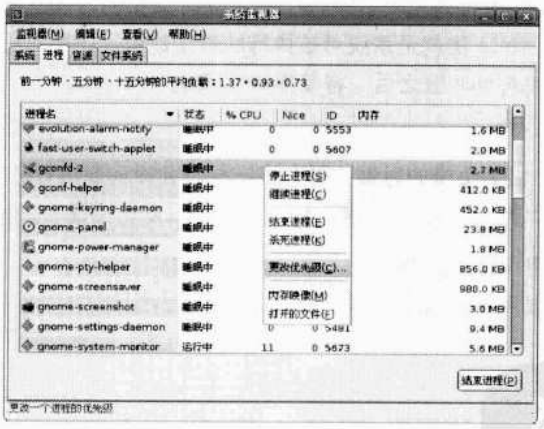


图 17.3 在系统监视器中修改进程优先级

在上面的快捷菜单中选择【更改优先级】，弹出如图 17.4 所示的对话框，然后通过拖拽窗口中的滑块来调整进程的优先级，nice 值越小优先级越高。

图 17.4 改变优先级

对于一个管理员来说，使用命令来管理进程的优先级更为专业和便捷，下面学习一下如何用命令来调整进程的优先级，首先来看一看输入 ps -l 这个命令所显示出来的信息。

```
user@ubuntuer:~$ ps -l
  F S   UID   PID  PPID  C PRI  NI ADDR    SZ WCHAN  TTY          TIME CMD
0 S     0  6687  6686  0  80   0-   1080 -      pts/0    00:00:00 bash
4 R     0  6719  6687  0  80   0-   537-       pts/0    00:00:00 ps
```

注意，上面信息中：

- UID 代表执行者的身份。
- PID 代表这个进程号。
- PPID 代表这个进程是由哪个进程发展出来的，即父进程。
- PRI 代表这个进程的优先级，值越小越早被执行。
- NI 代表这个程序的 nice 值。

PRI 就是该程序被 CPU 执行的先后顺序，所以当 CPU 繁忙的时候，那么 PRI 值越小的会越快被执行，而 NI 就是 nice 值，nice 值就是系统可被执行的修正数值。如前面所说的，由于 PRI 值越小越快被执行，而由于我们加入 nice 值之后，将使得 PRI 变为：

```
PRI(new) = PRI(old) + nice
```

这样一来，当 nice 值为负值的时候，那么该进程将会提前被执行。需要注意的是：只有具有 root 权限的用户，才可以将程序的 nice 调为负值。一般可以这样说：

- 一般用户可以用 nice 值：0~19
- root 管理员可以用 nice 值：-20~19

nice

⚙语法

```
user@ubuntuer:~$ nice [-n number] command
```

⚙参数说明

-n：其后那个 number 即为 nice 值。

⚙例子

```
user@ubuntuer:~$ nice -n -5 find / -name core > /tmp/core
```

⚙说明

就如同前面说的，nice 是用来调整进程的执行优先级，比如一些非重要的进程可以降低其优先

级，如备份工作，由于备份工作相当地耗费系统资源，这个时候就可以将备份的命令的 nice 值调大一些，从而使系统的资源分配得更为公平。

renice

语法

```
user@ubuntuer:~$ renice [number] PID
```

例子

```
user@ubuntuer:~$ ps -aux
user@ubuntuer:~$ renice 5 234
```

说明

由于 renice 是将一个正在进行当中的程序的优先级降低，所以，通常 renice 也与 ps 相互配合使用。先找到某个进程的 PID 之后，才来重新设定它的 nice 值。

17.5 信息管理

在 Ubuntu 中，还提供了一些帮助用户查看系统运行历史的工具，其中包括系统日志查看器和 dmesg、last 等 Shell 指令，下面分别对其进行阐述。

17.5.1 信息管理器

如果用户想要知道开机时的系统启动信息，或者目前有哪些用户登录在系统上，系统的当前时间或最近某个用户登入的时间，这都需要使用到 Linux 下信息管理器，而 Ubuntu 提供了一个图形化的程序来管理这些信息，可以通过执行【系统】|【系统管理】|【系统日志】命令打开系统日志查看器，如图 17.5 所示。

图 17.5 系统日志

从上图中可以看到，系统包括如下的日志：

- auth.log：授权日志
- daemon.log：daemon 进程日志
- debug：调试输出
- kern.log：内核输出日志
- syslog，messages：主要的系统日志
- Xorg.0.log：X-window 的日志

17.5.2 信息维护指令

除此之外，Linux 下还提供了 last，who，dmesg 等命令来帮助管理员识别和维护系统信息。

dmesg

语法

```
user@ubuntuer:~$ dmesg
```

例子

```
user@ubuntuer:~$ dmesg | more
```

说明

在开机的时候会发现有很多的信息显示在显示器上，例如 CPU 的信息、硬盘、光盘型号及硬盘分割表等。但是这些信息都是迅速地被新显示出的信息挤出了显示范围，而这些信息有时候对于系统管理员是很重要的，因为它提供了系统的启动过程中的信息，可以用 dmesg 这个命令来查看这些信息。通过加上 | more 这个管线指令来使画面暂停，方便找到信息。

uptime

语法

```
user@ubuntuer:~$ uptime
```

例子

```
user@ubuntuer:~$ uptime
 11:27pm  up 1 day, 4:16, 2 users, load average: 0.17, 0.06, 0.16
```

说明

想知道主机是否已经开机，那就使用 uptime 命令。还有，过去 1 分钟、5 分钟、15 分钟的系统平均负载是多少，都可以通过 uptime 命令获取信息。在上面的例子中，执行 uptime 之后，显示目前时间是 11:27pm，而系统已经开机了 1 天 4 小时 16 分之多，目前有两个用户在线，过去 1 分钟、5 分钟、15 分钟系统平均负荷为 0.17、0.06、0.16。

who & w

语法

```
user@ubuntuer:~$ who
user@ubuntuer:~$ w
```

例子

```
user@ubuntuer:~$ who
user     tty7    2008-07-14 07:34(:0)
user    pts/0   2008-07-14 07:51(:0.0)
user@ubuntuer:~$ w
  12:23:43 up  1 day, 5:08, 2 users, load average: 0.00, 0.01, 0.09
USER     TTY      FROM              LOGIN@   IDLE   JCPU   PCPU  WHAT
user     tty7: 0                  07:34    0.00s  1:18   0.90s x-session-manager
user     pts/0: 0.0                07:50    1.00s  1:14s   0.18s w
```

说明

who 和 w 是用来查看目前在系统上的用户命令。基本上，who 与 w 的功能是相同的，只是 who 仅列出用户名称与登入时间，至于 w 则会列出用户的以下信息：

- 来源地址（IP）：就是 FROM 那一项，即 IP。
- 登录时间：即 LOGIN@那一项。
- 工作内容：即 WHAT 那一项。

此外，在使用 w 的时候，开头会有一个信息，那个是 uptime 的输出结果。

whoami

语法

```
user@ubuntuer:~$ whoami
```

例子

```
user@ubuntuer:~$ whoami
user
```

说明

一位称职的系统管理员应该尽量不要使用 root 登入系统，而是通过使用 su 或者 sudo 来管理，不过可能由于执行程序的关系，常常会忘了自己的真实身份，这个时候 whoami 就发挥作用了。

last

语法

```
user@ubuntuer:~$ last
```

参数说明

–number：number 为数字，如果用户的登录记录太多了，可以使用这个参数。

⚙例子

```
user@ubuntuer:~$ last
user     pts/0:  0.0          Mon Jul  14 07:51  still logged in
wtmp begins Mon Jul  14 07:51:26 2008
```

⚙说明

要知道主机有没有被入侵，经常使用的就是 last 这个命令，包括 ftp，telnet，ssh 都会被记录在这个日志当中，不过目前只记录了一个月的日志量。

date

⚙语法

```
user@ubuntuer:~$ date [-s] [-R]
user@ubuntuer:~$ date +[format]
```

⚙参数说明

- -s：用来设置系统时间的参数。
- -R：如果发现是中文语言的，在纯文字模式下用该参数+[format]试试看。
- %a：星期几。
- %b：月份名称。
- %d：日期。
- %y：年份。

还有很多的参数，请使用 man 自行查询。

⚙例子

```
user@ubuntuer:~$ date +%a" "%b" "%y
Mon Jul 08
user@ubuntuer:~$ date -s 07/16/2008    <==改变日期
user@ubuntuer:~$ date -s 00:12:00      <==改变时间
user@ubuntuer:~$ clock -r          <==检查 BIOS 中的时间
user@ubuntuer:~$ clock -w          <==将目前的系统时间写到 BIOS 中去
```

⚙说明

date 的简单用法是只能查看时间，不过，更广义的用法是可以搭配很多种参数来进行时间输出的记录。此外，还可以用来更改时间，不过，在 date 改完时间后，还要使用 clock 命令将时间记录在 BIOS 中才算是完成了时间修改。

hostname

⚙语法

```
user@ubuntuer:~$ hostname
```

⚙例子

```
user@ubuntuer:~$ hostname
ubuntuer
```

说明

hostname 就是用来查看主机名称的命令。

17.6 作业调度

每个人或多或少都有一些约会或者作业，有的作业是例行性的，例如每年一次的旅行、每个月一次的工作报告、每天一次的下午茶等，有的作业则是临时发生的，例如某同事生病了，不能参加某个会议了等。

从上面的说明当中可以很清楚地发现两种事情发生的方式：

- 一种是例行性的，就是每隔一定的周期时间要来处理的事情；
- 另一种是突发性的，就是这次出现到结束就算告一段落。

在 Ubuntu Linux 下提供了两个命令来处理这两种作业：

- at：作业仅执行一次就从系统中的工作队列中取消；
- crontab：这个作业将持续、例行性地执行。

17.6.1 at 指令

语法

```
user@ubuntuer:~$ at [-m] TIME        <==作业命令 at
user@ubuntuer:~$ atq                 <==查看目前的作业队列
user@ubuntuer:~$ atrm [jobnumber]    <==删除作业队列
```

参数说明

- -m：执行 at 所规范的作业队列时，将屏幕输出结果 mail 给下达指令的用户。
- TIME：时间的格式，有以下几种。
  - HH:MM YYYY-MM-DD
  - HH[pm;am] + number [hours;days;weeks]
  - HH:MM
  - HH[pm;am] [Month] [Day]
- jobnumber：每一个 at 作业队列都有唯一的顺序号。

例子

```
user@ubuntuer:~$ at 5pm   <==在今天的 5pm 执行，如果今天已过 5pm 则明天执行
warning: commands will be executed using  /bin/sh
at> mail -s user user < /home/user/.bashrc <==作业具体执行命令
at> <EOT> <==这里是按下[Ctrl] + D就可以退出了
job 8 at Thu Jul 17 20:10:00 2008  <==这里会告诉你这个作业编号为 8 号，执行的日期为
后面所示
user@ubuntuer:~$ atq   <==查看当前用户(user)目前有多少作业
```

```
5       Thu Jul 16 20:10:00 2008 a user
8       Thu Jul 17 20:10:00 2008 a user
user@ubuntuer:~$ atrm 5 <==删除 5 号作业
user@ubuntuer:~$ atq
8       2002-05-30 17:00 a rest
```

说明

在 at 命令下达之后，便进入命令输入的模式，在这里可以重复地输入命令，在离开的时候按 Ctrl+D 组合键就可以离开了。离开之后，系统会告诉这个作业队列中的编号以及用户是谁。

17.6.2 crontab 指令

在介绍 crontab 命令前，首先要熟悉一下 cron 这个服务。当执行 crontab 命令之后，会将命令写入/var/spool/cron 这个目录当中，例如 user 用户执行 crontab 命令，那么就会自动产生/var/spool/cron/user 这个文件。需要注意的是，这个文件不能直接编辑。以后执行的命令记录会放置在/var/log/cron 这个文件中。

语法

```
user@ubuntuer:~$ crontab [-u user] [-l | -e | -r]
```

参数说明

- -u user：只有 root 能使用该参数。root 用户可以查看或编辑其他用户的 crontab 内容，其它用户只能编辑自己的 crontab 内容，因此该参数只供 root 用户使用。
- -l：列出 crontab 的内容。
- -e：编辑 crontab 的内容。
- -r：删除 crontab 的内容。

范例 1

```
user@ubuntuer:~$ crontab -e <==自己编辑自己的 crontab 内容
进入 crontab 编辑内容，使用 vi
0 12 * * * mail user < /home/user/user.txt
<==分时日月周 |========命令行================|
```

说明

上面的例子是说：假如用户需要在每天的中午 12:00 发一封信给自己，而且信的内容已经写好了，那要怎样做呢？而且，另一个假设是该用户在系统中的权限仅止于一般用户，并不是 root（管理员）身份，那要怎样设定其例行性命令呢？那就使用 crontab 这个命令，只要执行 crontab -e 就可以进入 vi 的编辑画面来编辑例行性命令。

在上面的例子中，输入 crontab -e 时，会出现一个 vi 画面，然后在 vi 画面中输入上面的一行字，之后输入:wq 存储后离开，即可完成编辑。那上面那一行字代表什么意义呢？你可以看到，在真正执行命令之前（就是 mail user < /home/user/user.txt ）总共有 5 个数字，这 5 个数字分别代表：

- 分　　(0~59)
- 小时　(0~23)
- 日期　(1~31)
- 月份　(1~12)
- 周　　(0~6)

数字的意义及范围如表 17.1 所示。

表 17.1　数字的意义及范围

数字代表的意义	分钟	小时	日期	月份	周
范围	0~59	0~23	1~31	1~12	0~6（0 为星期天）

另外，如果是*，代表所有数字都适用的意思，那上面例子中的时间就是：不论何月、何日、星期几的 12 点 0 分时，执行 mail user < /home/user/user.txt 这个命令。如果还不清楚的话，下面再讲几个例子。

范例 2　假如你每隔五分钟要去检查一次一个名为 test.sh 的脚本文件

```
user@ubuntuer:~$ crontab -e
*/5 * * * * /home/user/test.sh  <==新加入的一个作业
```

crontab 命令对应每个用户都只有一个文件，就是在/var/spool/cron 里面的文件，不过还有两件事要注意一下。

(1) 命令的路径最好是写成绝对路径，这样不容易出现找不到文件的问题；
(2) */5 表示每五分钟执行一次。

范例 3　假如每星期的星期五下午 4:30 要告诉朋友不要忘记星期六的约会

```
user@ubuntuer:~$ crontab -e
*/5 * * * * /home/user/test.sh
30 16 * * 5 mail friend@test.domain.name < /home/user/friend.txt
                                                    <==新加入的作业
```

范例 4　查看当前的作业状况，那就是使用 crontab –l

```
user@ubuntuer:~$ crontab -l <==这个-l 是 L 的小写
*/5 * * * * /home/user/test.sh
30 16 * * 5 mail friend@test.domain.name < /home/user/friend.txt
```

范例 5　使用 crontab –r 删除作业

```
user@ubuntuer:~$ crontab -r
user@ubuntuer:~$ crontab -l
no crontab for test
```

从上面的结果看到，crontab 整个内容都不见了，所以请注意：如果只是要删除 crontab 的某个作业，那么请使用 crontab –e 来重新编辑即可，如果使用–r 参数，会将所有的 crontab 数据内容都删掉，务必小心。

crontab –e 是针对用户的 cron 来设计的，如果是系统的例行性任务，该怎么办？是否还需要以

crontab -e 来管理例行性命令？当然不需要，只需要编辑/etc/crontab 文件就可以了。需要注意的是：crontab -e 的作用其实是编辑/usr/bin/crontab 这个执行文件，但是/etc/crontab 是个纯文本文件，可以 root 的身份编辑这个文件。基本上，cron 服务的最低检测时间单位是分钟，所以 cron 每分钟读取一次/etc/crontab 与/var/spool/cron 中的数据内容，因此，只要编辑完/etc/crontab 文件并且保存之后，crontab 时设定就会自动执行。

注意 Linux 下的 crontab 会自动每分钟重新读取一次/etc/crontab 的例行工作事项，但是某些原因或在其他的 Unix 系统中，由于 crontab 是读到内存中，所以在修改完/etc/crontab 之后可能并不会马上执行，这时请重新启动 crond 服务，执行以下命令：

```
/etc/rc.d/init.d/crond restart
```

接下来看看/etc/crontab 这的文件的内容。

```
user@ubuntuer:~$ vi /etc/crontab
# /etc/crontab: system-wide crontab
# Unlike any other crontab you don't have to run the 'crontab'
# command to install the new version when you edit this file
# and files in /etc/cron.d. These files also have username fields,
# that none of the other crontabs do.

SHELL=/bin/sh
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin

# m h dom mon dow user  command
17 *    * * *   root    cd / && run-parts --report /etc/cron.hourly
                        <==每小时执行的工作
25 6    * * *   root    test -x /usr/sbin/anacron || (cd / && run-parts --report
/etc/cron.daily) <==每天执行的工作
47 6    * * 7   root    test -x /usr/sbin/anacron || (cd / && run-parts --report
/etc/cron.weekly) <==每星期执行的工作
52 6    1 * *   root    test -x /usr/sbin/anacron || (cd / && run-parts --report
/etc/cron.monthly)
<==每个月执行的工作
#
```

看到这个文件的内容会发现，这个文件与刚刚运行的 crontab -e 的内容几乎一模一样，只是有几个地方不太相同。

17 ** * * root cd / && run-parts --report /etc/cron.hourly。读者可以发现，5 个数字后面接的是 root，这行代表的是执行的层级为 root 身份。当然，也可以将这一行改写成其他的身份。而 run-parts 后面接的/etc/cron.hourly 是一个目录内（/etc/cron.hourly）的所有可执行文件，也就是说，每个小时的 17 分，系统会以 root 层级的用户去/etc/cron.hourly 这个目录下执行所有可以执行的文件。后面的三行也都是类似的意思。你可以到/etc/ 目录底下去看看，系统本来就预设了这四个目录，可以将每天需要执行的命令直接写到/etc/cron.daily 中即可，还不需要使用到 crontab -e 命令，很方便。

/etc/crontab 支持两种执行命令的方式，一种是直接以命令形式输入，一种则是以“目录”形

式输入。

命令*/5 * * * * user /home/user/test.sh 的使用者是 user，且每 5 分钟执行一次 test.sh 脚本。

目录*/5 * * * * root run-parts /root/runcron 建立一个/root/runcron 的目录，将每隔 5 分钟执行的可执行文件都写到该目录下，就可以让系统每 5 分钟执行一次该目录下的所有可执行文件。

此外，与 crontab -e 使用当中最不相同的就是多了一个用户层级的关系，通常都是以 root 的角度来规划例行性命令，但是总有不需要 root 的命令的时候，就可以使用这个层级来规范该程序的用户属于哪个用户了。假设现在要建立一个目录，让系统可以每 2 分钟去执行这个目录下的所有可以执行的文件，可以在/etc/crontab 中写下面的命令：

```
*/2 * * * * root run-parts /etc/cron.min
```

前提是/etc/cron.min 这个目录是要存在的。如果需要执行的是一个程序而已，例如在检查网络流量时，希望每 5 分钟检查分析一次，可以这样写：

```
*/5 * * * * root /usr/local/mrtg-2/bin/mrtg /usr/local/apache/htdocs/mrtg/net/
mrtg.cfg
```

没有了 run-parts，就代表一个可执行文件。

现在可以知道，建立例行性命令很简单。如果是系统管理员，直接修改 /etc/crontab 这个文件即可，既便利，又方便管理。

17.7 课后练习

1. 什么是进程，什么是作业？进程包括哪几种类型？
2. 如何后台启动程序？后台启动程序有什么特点？如何切换前后台程序？
3. 在 Ubuntu 中如何通过系统监视器查看并管理进程信息？
4. 如何在 Ubuntu 中通过 ps、top 等指令查看进程管理信息？
5. 如何在 Ubuntu 中查看进程优先级，并设置进程的优先级？
6. 在 Linux Ubuntu 中通过信息管理器可以查看什么信息？
7. 在 Ubuntu 中，怎样实现作业调度？如何查询/etc/crontab 与 crontab 这个程序的用法和写法？

Chapter18 Shell 高级应用及 Shell 脚本

在第 8 章中，我们曾重点介绍 Ubuntu Shell 环境基础、应用及简单的设置。现在，随着对 Linux 系统知识掌握的深入，常常进行用户、文件、磁盘、设备以及进程等的管理，这时学习 Shell 的一些高级应用非常必要，这也是本章重点之一，包括通配符、管道及重定向等。另外 Linux 管理中，经常还需要执行一些批量命令，这时可以借助 Linux Shell 脚本。因此本章中的另一个重点即是 Linux Shell 脚本，包括 Shell 脚本入门，Shell 脚本基本语法，及调试使用。

18.1 通配符及正则表达式

如果忘记了要找的文件名，可以使用通配符或正则表达式。在不知道完整的文件名的情况下也可以在该文件上执行操作，只需填写所知的部分，剩余部分用通配符（wildcard）来替代。

18.1.1 文件名匹配

文件名匹配使得不必写出完整的名称，就可以指定多个文件。此时将用到一些特殊的字符，称为通配符(wildcards)。假设想用 rm 命令删除目录下所有以字符串 ".bak" 结尾的文件。除了在 rm 后跟上所有文件名作为参数，还可以用通配符'*'：

```
rm *.bak
```

可匹配一个或多个字符。在上面的例子中，告诉 Shell 将命令 rm 的参数扩展到 "所有以.bak 结尾的文件"，Shell 就将扩展后的参数告诉 rm 命令。

读者将看到，Shell 在命令执行前，就将读取并解释命令行。正是因为这个，才可以将通配符用于 Shell 命令的参数中。

下面进一步来认识通配符*。假如有个目录，其中含文件 124.bak、346.bak 及 583.bak。想只保留文件 583.bak，可以用：

```
rm *4*.bak
```

Shell 将*4*.bak 扩展成所有含 4 并以.bak 结尾的字符串。

注意到 rm 4*.bak 无法工作，因为这匹配的是以 4 开头的文件。由于目录中没有这样的文件，Shell 将这个模式扩展为空的字符串，故 rm 将返回出错信息：

```
rm: cannot remove '4*.bak': No such file or directory
```

如果想保留文件 346.bak'，而删除'124.bak'和'583.bak'。这看起来有些难度，因为被删文件的名称除了后缀其他都不同。但幸运的是，可以用不含有来指定文件：

```
rm *[!6].bak
```

这将被读为：除了以 6.bak 结尾的文件，删除其他所有以.bak 结尾的文件。必须将取反号(negation sign)与取反字符（这里是 6）放到括号中，不然的话，Shell 会将惊叹号(exclamation mark)解释成历史记录替换的开始(the beginning of a history substitution)。取反号在本篇介绍的所有匹配模式中都有效。

第二个通配符是问号(question mark)：?。在匹配时，一个问号只能代表一个字符。为了示范其用途，下面在上例的假设中添加两个新文件：311.bak~和 some.text。现在，列出所有在点号后有 4 个字符的文件：

```
ls *.????
```

问号通配符能够有效地避免上面提到的“取反号陷阱” (negation trap)：

```
rm *[!4]?.*
```

将扩展成“所有除了点号前倒数第二个字符为 4 的文件”，也就是只保留文件 346.bak。

有的读者可能会问，有没有其他的匹配方式？到目前为止，只看到了在指定位置匹配唯一字符的方法。其实也可以这样：

```
ls [13]*
```

将列出所有以字符 1 或 3 开头的文件。在本例中，文件 124.bak、311.bak~和 346.bak 匹配。注意到必须用中括号将匹配的模式括起来，否则模式只匹配以字符串 13 开头的文件。

接下来，将高兴地看到还可以定义匹配的范围：

```
ls *[3-8]?.*
```

将列出所有点号前倒数第二个字符落在 3~8 范围的文件。在本例中，匹配的文件是 346.bak 和 583.bak。

18.1.2 Shell 特殊字符

上面的那些机制存在一个缺点：Shell 总在命令执行前试着进行扩展，有时候会变得很棘手因为有时文件名包含特殊字符。假设在上面那个目录中还有一个名为!56.bak 的文件。

下面试图进行模式匹配：

```
rm !*
rm
rm: too few arguments
```

Shell 将“!*”解释成历史记录的替换（加入前一个命令的所有参数），而不是匹配方式。

Shell 命令本身带特殊字符作参数。Linux 下的一些命令行工具，比如 (e)grep、sed、awk、find 及 locate，都使用自己的正则表达式(regular expressions)。这些表达式与模式匹配看起来惊人地相似，但在某些地方又有所不同。

为了使这些特殊命令生效，Shell 就不能先将其当作模式匹配来解释。如：

```
find . -name [1-9]* -print
find: paths must precede expression
```

应该是：

```
find . -name '[1-9]*' -print
./346.bak
./124.bak
./583.bak
./311.bak~
```

可以通过反斜线(back slash)来引用特殊字符，比如!、$、?或空格：

```
ls \!*
!56.bak
```

或者用（单）引号：

```
ls '!'*
!56.bak
```

注意 要看清楚引号应该放在什么位置。命令 ls '!*' 将查找名为!*的文件，这是由于通配符也在引号中间，所以只能依照字面来解释。

18.1.3 正则表达式

简单地说，在 Linux 环境下，可以通过字符串以及一些特殊字符来进行文字的比对工作，好让用户筛选自己所需要的数据。这些特殊的字符与搭配使用的工具，就构成了正则表达式（Regular Expression，RE）的核心。

正则表达式的用途

对于系统管理员来说，正则表达式是一个“不可不学的好东西”。由于系统在繁忙的情况下每天产生的信息会多到你无法想象的地步，而我们也都知道，系统的“错误信息登录文件”的内容记载了系统产生的所有信息，当然，这包含系统是否被“入侵”的记录数据。但是系统的数据量太大了，要系统管理员每天去看这么多的信息数据，想不疯掉都很难。这个时候，就可以通过“正则表示式”的功能，将这些登录的信息进行处理，仅取出“错误”的信息来进行分析。

除了系统管理员之外，许多应用软件与服务的设置都是支持正则表达式的，最常见的例子就是“邮件服务器”。您是否常常收到电子邮件里最让人憎恶的“垃圾信件”呢？如果在邮件服务器端就将垃圾邮件给删除或剔除的话，邮件使用者就会减少很多不必要的时间损耗了。那么如何删除或剔除这些垃圾邮件呢？由于垃圾邮件几乎都有一定的标题或者是内容，因此，只要每次有来信时，都先将来信的标题与内容进行特殊字符串的比对，使用正则表达式发现有不良信件就予以删除或剔除。目前两大服务器软件 sendmail 与 postfix 都支持正则表达式的比对功能。另外 Apache2、vsftp 等也都支持正则表达式。

正则表达式语法

正则表达式中一些特殊字符所表示的意义如表 18.1 所示。

表 18.1 正则表达式语法

特殊字符	表示意义
^word	待搜寻的字符在行首
word$	待搜寻的字符在行尾
.	匹配任何一个可能的字符
\	跳脱符号，将特殊字符变成普通字符
?	任何一个单一字符
*	匹配模式中重复的字符
[list]	列表中的字符
[range]	列表中范围内的字符
[^list]	反向选择，与[list] 相反
[^range]	反向选择，与[range]相反
\{n\}	与前一个相同字符连续 n 个
\{n，m\}	与前一个相同字符连续 n-m 个

需要特别留意的是，正则表达式的特殊字符与一般在命令行输入命令的通配符并不相同，例如，在通配符当中，*代表的是 0~无限多个字符的意思，但是在正则表达式当中，*则是重复前一个字符的意思。

例如，在/etc 下，含有 XYZ 三个字符中任何一个字符的行都列出来。

```
grep [XYZ] /etc/*
```

又例如，在/etc 中，句首是 w-z 的都列出来。

```
grep ^[w-z] /etc/*
```


注意 使用正则表达式要养成良好的习惯，就是在匹配模式的两端加上单引号''，以和 Shell 的文件通配符号做区别。

Shell 文件通配字符和正则表达式的区别

(1) Shell 文件通配字符用于匹配文件名；正则表达式 RE 的主要用途是搜寻字符串、匹配文件内容和过滤特殊信息等。

(2) 由于严谨度的不同，正则表达式之上还有更严谨的延伸正则表达式。

(3) 正则表达式的处理方式，经常是以“整行”或称为“整段”来进行处理的。

(4) grep 与 egrep 在正则表达式中是很常见的两个程序语句，其中，egrep 可以用不同的模式去匹配，它支持更严谨的正则表达式的语法。

18.2 管道及重定向

当想存储及（或）打印信息以便以后阅读时，可以使用管道和输出重定向。例如可以使用 grep 来搜寻文件中的某一类内容，然后把结果保存在文件中或发送给打印机。例如，要打印 sneakers.txt

文件中关于 coffee 的行，只需输入：

```
grep coffee sneakers.txt | lpr
```

Unix 的理念是汇集许多小程序，每个小程序都有特殊的专长。复杂的任务不是由大型软件完成，而是运用 Shell 的机制，组合许多小程序共同完成。重定向就在其中发挥着重要的作用。

18.2.1 管道

如前所述，bash 命令执行的时候有输出的数据。那么如果这群数据必须经过几个步骤之后才能得到所想要的格式，应该如何来设置？这就涉及管道命令（pipe）的问题了，管道命令使用的是"|"这个界定符号。另外，管道命令与"连续下达命令"是不一样的。这点下面会再进行说明。先举一个例子来说明一下简单的管道命令。假设要从 last 这个命令中读取用户 root 在这个月内的登录次数，那么所进行的步骤如下。

Step 01 执行last命令，将所有这个月的登录数据取出来；
Step 02 使用grep命令将上面的输出数据（stdout）当中的root提取出来；
Step 03 使用wc这个可以计算行数的命令将上一步的数据计算行数。

由于 last 的输出是一行代表一次登入，所以只要计算几行就计算出登入几次。经由上面三个步骤，将 last 数据逐步筛选，就可以得到所要的数据了。整个命令可以写成：

```
user@ubuntuer: ~$ last
user@ubuntuer: ~$ last | grep root
user@ubuntuer: ~$ last | grep root | wc -l
```

可以先执行 last，然后再逐步增加为 last | grep root，最后到上面那一行，那么就马上可以清楚为何这么做。这个管道命令 | 仅能处理由前面一个命令传来的正确信息，也就是 standard output（STDOUT）的信息，对于 standard error 并没有直接处理的能力。那么整体的管道命令可以使用下图 18.1 表示。

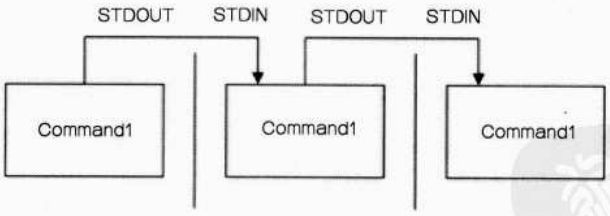


图 18.1 管道命令执行

通用的管道（pipe），由管道符号 | 来标识。语法是：

```
command1 | command2 | command3……
```

例如管道经常将一个程序的输出送到 more 或 less 来阅读。例如：

```
ls -l | less
```

其中，第一个命令提供目录内容，第二个则将其以翻页的方式显示。

更复杂的例子如：

```
rpm -qa | grep ^x | less
```

第一个命令给出所有已安装的 RPM 包，第二个则将其过滤(filter：'grep')，只剩下以^x 开头的包，第三个命令则将结果以翻页的方式显示。

18.2.2 管道命令

管道命令在 bash 的连续的处理程序中是相当重要的，在日志文件的分析当中管道非常有帮助，需特别留意。每个管道的子部分都是“命令”，其中后一个命令的输入乃是由前一个命令的输出而来的。下面介绍一些基本的管道命令。

cut 命令

语法

```
user@ubuntuer: ~$  cut -d "分隔字符" [-cf] fields
```

参数说明

- -d：后面接的是用来分隔的字符，默认是空格符。
- -c：后面接的是第几个字符。
- -f：后面接的是第几个区块。

示例

```
user@ubuntuer: ~$ cat /etc/passwd | cut -d ":" -f1
<==将 passwd 这个文件中每一行里头的：用来作为分隔符
而且列出第一个区块。也就是姓名
user@ubuntuer: ~$ last | cut -d " " -f1
<==以空格符为分隔符，并列出第一个区块
user@ubuntuer: ~$ last | cut -c1-20
<==将 last 之后的数据，每一行的 1-20 个字符取出来
```

说明

cut 主要的用途在于将同一行里面的数据进行分解。最常使用在分析一些数据或文字数据。这是因为有时候会以某些字符当作分割的参数，然后来将数据加以切割，以取得所需要的数据。

sort 命令

语法

```
user@ubuntuer: ~$ sort [-t 分隔符] [(+起始)(-结束)] [-nru]
```

参数说明

- -t 分隔符：使用分隔符来隔开不同区间，默认是 Tab.
- +起始-结束：由第 start 区间排序到 end 区间。
- -n：使用纯数字排序（否则就会以文字形态来排序）。

- -r：反向排序。
- -u：重复出现的行，只列出一次。

⚙示例

```
user@ubuntuer: ~$ cat /etc/passwd | sort
<==将列出来的个人账号排序
user@ubuntuer: ~$ cat /etc/passwd | sort -t: +2n
<==将个人账号中，以用户 ID 来排序
user@ubuntuer: ~$ cat /etc/passwd | sort -t: +2nr
<==反向排序
```

⚙说明

sort 同样是很常用的命令，因为常常需要比较一些信息。例如一个用户有很多账号，而且想要知道最大的用户 ID 目前是多少号。使用 sort 一下子就可以知道结果。

wc 命令

⚙语法

```
user@ubuntuer: ~$ wc [-lmw]
```

⚙参数说明

- -l：多少行
- -m：多少字符
- -w：多少字

⚙示例

```
user@ubuntuer: ~$ cat /etc/passwd | wc -l
<== 这个文件里面有多少行
user@ubuntuer: ~$ cat /etc/passwd | wc -w
<== 这个文件里面有多少字
```

⚙说明

wc 是相当有用的计算文件内容的一个命令。举个例子来说，当想要知道目前系统中已经有多少个账号时，就可以使用 wc -l。因为/etc/passwd 中一行代表一个用户，所以知道行数就等于知道有多少账号在里面了。而如果要计算一个文件里面有多少个字符时，就使用 wc -w 这个参数。

uniq 命令

uniq 命令用来删除文件中重复的行。

⚙语法

```
uniq [-c|-d|-u][-f Fields] [+character]
```

⚙参数说明

- -c：在输出行前面加上各行在输入文件中出现的次数。
- -d：仅显示重复的行。

- -f Fields：忽略 Fields 变量指定的字段的数目。
- +Character：忽略 Character 变量批定的字符的数目。
- -u：仅显示不重复的行。

⚙示例

```
user@ubuntuer: ~$ last | cut -d" " -f1 | sort | uniq
```

⚙说明

这个命令用来将重复的行删除掉只显示一个，如要知道这个月份登录主机的用户有谁，而不在乎他的登录次数，那么就使用上面的示例，具体步骤为：①先将所有的数据列出；②再将人名独立出来；③经过排序；④只显示一个。由于这个命令是在将重复的东西减少，所以当然需要配合排序过的文件来处理。

tee 命令

⚙语法

```
user@ubuntuer: ~$ last | tee last.list | cut -d " " -f1
```

⚙参数说明

- -a：将内容追加到指定的文件的尾部。
- -i：忽略中断信号。

⚙示例

```
user@ubuntuer: ~$ last | tee last.list | cut -d " " -f1
```

⚙说明

有时在命令重定向的时候，如果要将数据送出到文件，屏幕上不出现任何的数据。那么如果需要将数据同时显示在屏幕上和文件中，这个时候就需要 tee 这个命令了。使用 last 可以查看到这个月份的登入数据，而使用了 tee 之后，会将数据同时传给下一个命令去执行，也会将数据写入 last.list 文件中。

tr 命令

⚙语法

```
user@ubuntuer: ~$ tr [-ds] SET1
```

⚙参数说明

- -d：删除 SET1 这个字符串。
- -s：取代重复的字符。

⚙示例

```
user@ubuntuer: ~$ last | tr '[a-z]' '[A-Z]'<==将小写改成大写
user@ubuntuer ~$ cat /etc/passwd | tr -d: <== :这个符号在/etc/passwd 中不见了
user@ubuntuer: ~$ cat /home/test/dostxt | tr -d '\r' > dostxt-noM
                                          <==将 DOS 文件的字尾符号 ^M 的符号去除
```

⚙说明

这个命令也可以写在正则表达式里，因为它也是用正则表达式的方式来取代数据的。以上面的例子来说，使用[]可以设定一串字符，也常常用来取代文件中的特殊符号。例如上面第三个例子当中，可以去除 DOS 文件留下来的^M 这个换行符号。这个东西相当有用！相信处理 Linux & Windows 系统的最麻烦的一件事就是这个事情啦，即 DOS 下会自动地在每行行尾加入^M 这个换行符号。这个时候可以使用 tr 来将 ^M 去除。^M 可以使用\r 来代替之。

split 命令

⚙语法

```
user@ubuntuer: ~$ split [-bl] 输入文件输出文件
```

⚙参数说明

- -b：以文件大小来分。
- -l：以行数来分。

⚙示例

```
user@ubuntuer: ~$ split -l 5 /etc/passwd test    <==会产生 testaa, testab, testac…
                                                    等文件
```

⚙说明

如果要将文件分割的话，那么就使用这个命令来操作。-b size 参数指定分割后文件的大小，如果是行数的话，那么就使用-l line 来分割。这样一来，就可以轻易地将你的文件分割成各种大小，以方便复制。

18.2.3 重定向至文件

在 Linux 下，重定向到文件中，或以文件内容作为命令的参数，可以通过>，>>和<来实现，下面以两个例子来介绍这两个重定向命令。

示例 1

将 command 的输出保存到 file 中，这将覆盖 file 中的内容。

```
command > file
```

示例 2

将文件 dirlist 的内容送到命令 sort，然后再将排序后的结果送到文件 sdirlist。

```
sort < dirlist > sdirlist
```

需要特别注意的是，>重定向的文件将被覆盖掉。在上面的例子当中，sdirlist 文件（假如该文件已存在）的原有内容将被删除，取而代之的是 sort<dirlist 的执行结果。如果不想删除原有文件的内容，而是把结果追加在文件内容的后面，可以这样做：

```
sort < dirlist >>sdirlist
```

从上面的示例可以知道，>>有追加内容的意思。

18.3 Shell 脚本入门

如果想要更加了解与控制 Linux，使 Linux 运作更顺畅，那么 Shell 脚本是必须要学习的一个要点。Shell 脚本在处理自动循环或大的任务方面可节省大量的时间，且功能强大。如果要处理一个任务的命令清单，不得不一个一个敲进去，然后观察输出结果，再决定它是否正确，如果正确，再继续下一个任务，否则再回到清单一步步观察。任务可能是将文件分类、向文件插入文本、迁移文件、从文件中删除行、清除系统过期文件，以及系统一般的管理维护工作等。创建一个脚本，在使用一系列系统命令的同时，可以使用变量、条件、算术和循环快速创建脚本以完成相应工作。这比在命令行下一个个敲入要节省大量的工作时间。在这一节中，我们将以例子进行说明，好让读者能够了解 Shell 脚本的功能。

18.3.1 脚本的执行

在前面介绍的内容中讲过变量、管道命令和重定向等，这些都是为了给脚本做铺垫。现在讨论一下执行脚本时 bash 的执行步骤。

Step 01 如果读取到一个回车符号（CR），就尝试开始执行该行命令。
Step 02 如同前面 bash command 提到的，命令间的多个空白会被忽略掉。
Step 03 空白行也将被忽略掉，并且 tab 也是不会被处理的。
Step 04 如果一行的内容太多，则可以使用\来延伸至下一行。

此外，使用最多的#可作为批注。任何加在# 后面的字将全部被视为注释文字而被忽略。另外，在写脚本的时候，最好养成下面一些好的习惯。

- 先声明使用的 Shell 为哪种（特别留意这一点，在某些情况下，例如/etc/crontab 情况下，如果没有声明使用的 Shell，常常会出现错误信息而导致脚本无法被执行）。
- 注明该脚本的内容功能、版本信息、作者、文件创建日期等。
- 每一个大步骤的主要功能（以便自己将来修改之用）。

那么如何执行脚本文件呢？执行的方法有以下两种：

- 一个是将该脚本文件改成可以执行的属性，如 chmod 755 scripts.file，然后执行该脚本。
- 另一种则是直接以 sh 命令来执行脚本的内容，如 sh scripts.file。

18.3.2 第一个脚本

下面来建立第一个简单的脚本，这个简单的脚本就是要在屏幕上输出 "Hello ! How are you ?" 。

```
user@ubuntuer: ~$ mkdir test; cd test
```

```
user@ubuntuer: test$ vi test01-hello.sh
#!/bin/bash      <==在# 之后加上 !与 Shell 的名称，用来声明使用的 Shell
#这个脚本的用途在于在屏幕上输出 Hello ! How are you
# 创建日期: 2008/05/20
# Made by user

hello=Hello\ \!\ How\ are\ you\ \?       <==这就是变量
echo $hello
user@ubuntuer: test$ sh test01-hello.sh
Hello ! How are you ?                    <=输出的结果显示在屏幕上
```

这里需要注意以下几点：

- 所有脚本里面的东西、基本规则（如变量设定规则）需要与命令行下相同。
- 脚本的后缀名最好为.sh，以便他人识别。
- 并不是加上 .sh 就可以是执行文件，还需要查看其属性中是否有 x 这个属性。

这就是一个简单 shell 脚本了，接下来写稍微复杂一点的。如果一个脚本里头有两个以上的变量要相互引用，该如何处理？这个时候顺便来比较一下" 与' 的异同。

```
user@ubuntuer: test$ vi test02-2var.sh
#!/bin/bash
#这个脚本用途在于引用两个变量，顺便比较一下" 与' 的异同
# Date: 2008/05/20
# Made by user
name="user"
myname1="My name is $name"
myname2='My name is $name'
echo $name
echo $myname1
echo $myname2
user@ubuntuer: test$  sh test02-2var.sh
user
My name is user
My name is $name
```

从输出的结果看到，" 与' 最大的不同就在于能不能保存变量值。

18.3.3 交互式脚本

什么是交互式脚本？很简单，例如，在执行 Windows 的安装程序时，系统会时不时跳出一个窗口，问"下一步"、"上一步"或"取消"的操作，这就是交互。程序会依据输入的数据来进行判断，Shell 下最简单的交互式命令就是 read 命令。read 的功能就是把用户在键盘上输入的结果传递到变量中，例如：

```
user@ubuntuer: test$ read name
user <==这是键盘输入的结果
user@ubuntuer: test$ echo $name
user
```

设定如下逻辑：在执行脚本的时候，将由键盘输入的数据列出来。

```
user@ubuntuer: test$ vi test04-read.sh
#!/bin/bash
# This program is used to "read" variables
# user 2008/05/20
echo "Please keyin your name, and press Enter to start."
read name
echo "This is your keyin data ==> $name"
user@ubuntuer: test$ sh test04-read.sh
Please keyin your name, and press Enter to start.
user communication
This is your keyin data ==> user communication
```

就这么简单，后面还会继续谈到判断，那个时候输入的数据可就更重要了。下一步再来说一说怎样定义一个脚本的参数的代号。以下面的说明为例：

```
user@ubuntuer: test$ myscript opt1 opt2 opt3 opt4
                  $0       $1   $2   $3   $4
```

在这个脚本（myscript）里面，只要变量名称为$0就表示为myscript，也就是说：

- $0：myscript即脚本的名称。
- $1：opt1即第一个附加的参数（parameter）。
- $2：opt2。
- $3：opt3。

这样说或许不是很清楚，运行下面的脚本就知道是什么意思了。

```
user@ubuntuer: test$ vi test05-0123
#!/bin/bash
# This program will define what is the parameters
#user 2008/05/20
echo "This script's name => $0"
echo "parameters $1 $2 $3"
user@ubuntuer: test$ sh test05-0123 pa1 pa2 pa3
This script's name => test05-0123
parameters pa1 pa2 pa3
```

上面的功能定义在Shell脚本中非常有用，特别是在执行脚本前需要给脚本传递一些特定的多个参数时，这个功能变量就相当地重要了。

18.4 Shell 脚本基本语法

前面已经尝试做了几个简单的Shell脚本的例子，从中可以看出Shell类似C语言，也有自己的语法。下面就从Shell变量、Shell表达式、Shell运算符、Shell条件判断，Shell循环几个方面对Shell脚本的语法进行系统介绍。

18.4.1 变量及声明

了解了变量的脚本写法之后，现在来进行一个有趣的试验，在进行计算的时候，bash 能不能了解用户所给予的是数字还是单纯的字符串？这个很重要，因为这可能会造成系统的误判。来试试看！当需要输出 3 * 5 的结果时，需要如何做呢？用单纯的命令行逐行输入的结果如下：

```
user@ubuntuer: test$ a=3
user@ubuntuer: test$ b=5
user@ubuntuer: test$ c=$a*$b
user@ubuntuer: test$echo $c
3*5  <==变成了字符串
```

上面输出的不是所希望看到的 3*5 = 15 的结果，这是因为没有定义该变量，则该变量默认是呈现字符串类型，那么$c 就自然成为字符串类型了。所以需要来声明一下变量，声明变量使用的是 declare 这个命令。

语法

```
user@ubuntuer: test$ declare [-afirx]
```

参数说明

- -a：定义为数组 array。
- -f：定义为函数 function。
- -i：定义为整数 integer。
- -r：定义为只读。
- -x：定义为通过环境输出变量。

示例

```
user@ubuntuer: test$ declare -i a=3
user@ubuntuer: test$ declare -i b=5
user@ubuntuer: test$ declare -i c=$a*$b
user@ubuntuer: test$ echo $c
15  <==变成数字
```

现在，如果要在计算结果当中，需要输入为 2*3+5*13-32+25 时，并且在最后输出“Your result is==>”该怎样写这个简单的脚本呢？可以这样试试看。

```
user@ubuntuer: test$ vi test03-declare.sh
#!/bin/bash
# This program is used to "declare" variables
# user 2008/05/20
number1=2*3+5*13-32+25
declare -i number2=2*3+5*13-32+25
echo "Your result is ==> $number1"
echo "Your result is ==> $number2"
user@ubuntuer: test$ sh test03-declare.sh
Your result is ==> 2*3+5*13-32+25
Your result is ==> 64
```

18.4.2 逻辑判断式

脚本还有一个重要的功能就是逻辑判断。举个例子来说，当要建立一个目录的时候，首先检测有没有该目录，如果有的话，那么就不需要建立，如果没有的话，那么就建立该目录。这个就需要脚本使用逻辑判断，由于是判断式，那么应该都会与判断条件有关系，所以下面的判断式大多与if...then...fi 这种条件判断式有关系，这部分内容将在后面做详细介绍。这里先提一下逻辑判断式的几个重要的项，如表 18.2 所示。

表 18.2 Shell 的逻辑判断式

功能	逻辑标识	表示意思
关于文件与目录的检测逻辑标识	-f	检测文件是否存在（常用）
	-d	检测目录是否存在（常用）
	-b	检测是否为一个 block 文件
	-c	检测是否为一个 character 文件
	-S	检测是否为一个 socket 标签文件
	-L	检测是否为一个符号链接文件
	-e	检测某个东西是否存在
关于程序的逻辑标识	-G	检测是否由 GID 所执行的程序所拥有
	-O	检测是否由 UID 所执行的程序所拥有
	-p	检测是否为程序间传送信息的 name pipe 或是 FIFO
关于文件的属性检测	-r	检测是否为可读的属性
	-w	检测是否为可以写入的属性
	-x	检测是否为可执行的属性
	-s	检测是否为非空白文件
	-u	检测是否具有 SUID 的属性
	-g	检测是否具有 SGID 的属性
	-k	检测是否具有 sticky bit 的属性
两个文件之间的判断与比较	-nt	第一个文件比第二个文件新
	-ot	第一个文件比第二个文件旧
	-ef	第一个文件与第二个文件为同一个文件（link 之类的文件）
逻辑与（and）或（or）	&&	逻辑的 AND 的意思
	||	逻辑的 OR 的意思

比较有趣的应该是第 1、3 这两种判断，尤其是在建立一些与权限相关的文件时，这个就更重要了。还有，&&和||也很重要。

18.4.3 运算符

下面介绍一下在 Bash Shell 脚本中的运算符号的加减乘除（见表 18.3）。

表 18.3 Shell 的运算符号

运算符号	代表意义	运算符号	代表意义
=	等于	-gt	大于
!=	不等于	-le	小于或等于
<	小于	-ge	大于或等于
>	大于	-a	双方都成立（and）
-eq	等于	-o	单方成立（or）
-ne	不等于	-z	空字符串
-lt	小于	-n	非空字符串

18.4.4 条件判断

if...then...fi

刚刚建立了第一个脚本，即在屏幕上面用脚本来输出问候语。现在要让脚本加上条件进行判断，这就是所谓的条件判断。最常用的就是 if...then...else if...then...fi 语句。这个条件判断的语法为：

```
if [ 条件判断一 ] && (||) [ 条件判断二 ]; then
(if 即起始的意思，后面可以接若干个判断式，使用&&或||)
    执行内容程序
elif [ 条件判断三 ] && (||) [ 条件判断四 ]; then
(第二段的判断，如果第一段没有符合就来此搜寻条件 )
    执行第二段内容程序
else
(当前两段都不符合时，就以这段内容来执行)
    执行第三段内容程序
fi
(结束 if then 的条件判断)
```

在中刮号[]里面的是条件判断式，如果是复合式的条件判断(如若 A 及 B 则 C 之类的逻辑判断)，那么就需要在两个中刮号之间加上“&& (and)”或者是“|| (or)”这样的逻辑表达式才行。如果是多重选择的话，那么就需要以 elif（可选的）来新增另一个条件。如果所有的条件都不适用，则使用 else（可选的）给出最后的执行内容。

不过，这里有几个比较容易犯的错误，需要加强说明一下。

- 在[ ]当中，只能有一个条件判断式。
- 在[ ]与[ ] 当中，可以使用&&或||来组织条件判断式。
- 每一个独立的组件之间都需要用空格键来隔开。

尤其是最后一点，需要特别注意。下面来使用一个简单的判别式。

```
user@ubuntuer: test$ vi test06-ifthen.sh
#!/bin/bash
# This program is used to study if then
# user 2008/05/20
echo "Press y to continue"
```

```
read yn
if ["$yn" = "y" ]; then
        echo "script is running..."
else
        echo "STOP!"
fi
user@ubuntuer: test$ sh test06-ifthen.sh
Press y to continue
y
script is running...
user@ubuntuer: test$ sh test06-ifthen.sh
Press y to continue
n
STOP!
```

很简单的一个例子，当输入为 y 的时候，就进行，若非 y 则不予以进行。但是这里有个问题，就是如果输入 Y，程序还是停止了。怎么办？这个时候就需要使用到 || 来组织条件判断式。可以这样写：

```
user@ubuntuer: test$ cp test06-ifthen.sh test07-ifthen.sh
user@ubuntuer: test$ vi test07-ifthen.sh
#!/bin/bash
# This program is used to study if then
# user 2008/05/20
echo "Press y to continue"
read yn
if ["$yn" = "y" ] || ["$yn" = "Y" ]; then
        echo "script is running..."
else
        echo "STOP!"
fi
user@ubuntuer: test$  sh test07-ifthen.sh
Press y to continue
y
script is running...
user@ubuntuer: test$  sh test07-ifthen.sh
Press y to continue
Y
script is running...
```

如果再加上前面提过的参数，再来试试看。

```
user@ubuntuer: test$ vi test08-ifthen.sh
#!/bin/bash
# set parameters in the if then
# 需要加上 hello 这个参数才会正确显示
# user 2008/05/20
if ["$1" = "hello" ]; then
        echo "Hello! How are you ?"
elif ["$1" = " " ]; then
        echo "You MUST input parameters"
```

```
else
        echo "The only accept parameter is hello"
fi
user@ubuntuer: test$ sh test08-ifthen.sh hello
Hello! How are you ?
user@ubuntuer: test$ sh test08-ifthen.sh
You MUST input parameters
user@ubuntuer: test$ sh test08-ifthen.sh asdf
The only accept parameter is hello
```

下面再看一个例子，假设已经知道 netstat 与 grep 这两个命令的用法，那么如果要来检测主机上面的端口是否开启时，可以使用下面的示例来进行。

```
user@ubuntuer: test$ vi port.sh                    <==编辑一个文件为 port.sh 的 script
#!/bin/bash                                         <==声明使用的 Shell 类型
# program: Using to study the [if ... then ... fi] program
# user  2008/05/20
# content: I will using this program to show your services
# 1. print the program's work in your screen
  echo "Now, the services of your Linux system will be detect!"
  echo "The www, ftp, ssh, and sendmail + pop3 will be detect!"
  echo " "
# 2. www
  www='netstat -an|grep LISTEN|grep :80'   <==这个就是变量，并使用了管道命令
  if ["$www" != " " ]; then                 <==开始条件的判断
      echo "WWW is running"                   <==若条件成立，那么就输出这一行的内容
  else
      echo "WWW is NOT running"
  fi
# 3. ftp
  ftp='netstat -an|grep LISTEN|grep :21'
  if ["$ftp" != " " ]; then
      echo "FTP is running"
  else
      echo "FTP is NOT running"
  fi
# 4. ssh
  ssh='netstat -an|grep LISTEN|grep :22'
  if ["$ssh" != " " ]; then
      echo "SSH is running"
  else
      echo "SSH is NOT running"
  fi
# 5. sendmail + pop3
  smtp='netstat -an|grep LISTEN|grep :25'
  pop3='netstat -an|grep LISTEN|grep :110'
  if ["$smtp" != " " ]   && ["$pop3" != " " ]; then
                                    <==有两个以上的条件时，就使用&&或||来分隔
      echo "sendmail is OK!"
  elif ["$smtp" != " " ] && ["$pop3"  = " " ]; then
      echo "sendmail have some problem of your pop3"
```

```
    elif ["$smtp"  = " " ] && ["$pop3" != "" ]; then
        echo "sendmail have some problem of your smtp"
    else
        echo "sendmail is NOT running"
    fi
user@ubuntuer: test$ sh port.sh                          <==执行并查看输出的结果
Now, the services of your Linux system will be detect!
The www, ftp, ssh, and sendmail + pop3 will be detect!
WWW is running
FTP is running
SSH is running
sendmail is NOT running
```

case...esac

if...then...fi 语句是程序自行判断，那么如果已经规定好几个条件，只要选择执行的种类方式就可以正确地执行的话，要怎么做？最简单的例子就是常用到的/etc/init.d/中的脚本，例如，重新启动 apache2 时使用：

```
/etc/init.d/apache2 restart
```

需要注意的是 restart 项，然后脚本就会自动地去搜寻 restart 项中的情况并执行。这个就是 case...esac 的使用模式。有没有注意到，开始用 case，结束则是使用 case 的倒写 esac。它的用法如下：

```
case 种类方式(string) in
    <==开始阶段，其中的"种类方式"可分成两种类型，通常使用$1 这一种直接下达类型
    种类方式一)
        程序执行段
        ;;                  <==种类方式一的结束符号
    种类方式二)
        程序执行段
        ;;
    *)
        echo "Usage: {种类方式一|种类方式二}"      <==列出可以利用的参数值
        exit 1
esac                              <==case 结束处
```

种类方式(string)的格式主要有两种：

- 直接输入：就是以“执行文件 + string ”的方式来执行（/etc/init.d 中的基本设定方式），则 string 可以直接写成$1（在执行文件后面直接加入第一个参数）。
- 交互式：就是由屏幕输出可能的选项，然后让用户输入，这个通常必须配合 read variable，string 则写成 $variable 的格式。

同样的，建立一个名为 test09-case.sh 的文件来做试验看看。假如共有三个选项，分别为 one，two，three，并假设使用直接输入方式，则可以写成：

```
user@ubuntuer: test$ vi test09-case.sh
#!/bin/bash
# program:      Using case mode
```

```
# user  2008/05/20
# content:      I will use this program to study the case mode!
# 1. print this program
  echo "This program will print your selection!"
case $1 in                  <==使用直接输入选项方式
  one)
        echo "your choice is one";;
  two)
        echo "your choice is two"
        ;;
  three)
        echo "your choice is three"
        ;;
  *)
        echo "Usage {one|two|three}"
                                  <==列出可以使用的参数（如果用户下达错误的参数时）
        exit 1
esac
user@ubuntuer: test$ sh test09-case.sh
                                  <==执行结果。显示没有对应的参数，所以列出可用参数
This program will print your selection!
Usage {one|two|three}
user@ubuntuer: test$ sh test09-case.sh three
This program will print your selection!
your choice is three
```

那么交互的 case 语句是怎么样的？利用上面的例子进行修改。

```
user@ubuntuer: test$ cp test09-case.sh test10-case.sh
user@ubuntuer: test$ vi test10-case.sh
#!/bin/bash
# program:      Using case mode
# user  2008/05/20
# content:      I will use this program to study the case mode!
# 1. print this program
echo "Press your select one, two, three"
read number
case $number in
  one)
        echo "your choice is one"
        ;;
  two)
        echo "your choice is two"
        ;;
  three)
        echo "your choice is three"
        ;;
  *)
        echo "Usage {one|two|three}"
        exit 1
esac
```

```
user@ubuntuer: test$ sh test10-case.sh
Press your select one, two, three
two    <=这一行是您输入的选项
your choice is two
```

18.4.5 循环

在程序段当中，最常使用到的就是循环了。循环是很重要的一项编程功能，尤其是具有判断形式的循环，经常被使用来判断一些事项的可行性与否。但是程序怎么知道什么时候应该要停止这个程序呢？这就需要加入判断。最简单的循环判断式可以是以下几种：

- for（条件一；条件二；条件三）
- for 变量 in 变量1 变量2 ...
- while [条件] && (||) [条件二] ...
- until [条件] && (||) [条件二]...

for 语句是已经知道要运行多少次，至于 until 与 while 则分别是：

- until：直到条件相同的时候才跳出循环
- while：当条件相同的时候就继续循环

下面先来谈一下最简单的循环，就是利用 for 语句来实现。假设计算 1+2+3...+100，用脚本如何写？有很多的方式，这里谈一谈 do...done。

```
user@ubuntuer: test$ vi test11-loop.sh
#!/bin/bash
# Using for and loop
# user 2008/05/20
declare -i s  # <==声明变量
for ( (i=1; i<=100; i=i+1) )
do
        s=s+i
done
echo "The count is ==> $s"
user@ubuntuer: test$ sh test11-loop.sh
The count is ==> 5050
```

for（条件一；条件二；条件三）是必须要用到的，其中：

- 条件一：这可以看成是初始值，如上面的例子中，初始值是 i=1.
- 条件二：这可以看成是符合值，如上面的例子中，当 i<=100 的时候都是符合条件的。
- 条件三：这可以看成是步阶。也就是说，i 每次都加 1。

上面的例子是说，由 i=1 开始到 i<= 100，每次 i 都加 1 来执行下面的程序段（就是 s=s+i ），当 i >100（也就是 i=101）时就跳出这个循环的程序段。

那么使用 while 或者是 until 要怎样执行？其实情况都差不多。我们使用下面两个脚本来进行

1~100 的累加操作。

使用 while

```
user@ubuntuer: test$  vi test12-loop.sh
#!/bin/bash
# Using while and loop
# user 2008/05/20
declare -i i
declare -i s
while ["$i" != "101" ]
do
        s=s+i
        i=i+1
done
echo "The count is ==> $s"
```

使用 until

```
user@ubuntuer: test$ vi test13-loop.sh
#!/bin/bash
# Using until and loop
# user 2008/05/20
declare -i i
declare -i s
until ["$i" = "101" ]
do
        s=s+i
        i=i+1
done
echo "The count is ==> $s"
user@ubuntuer: test$ sh test12-loop.sh
The count is ==> 5050
```

下面的例子是另一种循环的方式，可以用来判断非数字的类型。

```
user@ubuntuer: test$ vi test14-for.sh
#!/bin/bash
# using for...do ....done
# user 2008/05/20
LIST="Tomy Jony Mary Geoge"
for i in $LIST
do
       echo $i
done
user@ubuntuer: test$ sh test14-for.sh
Tomy
Jony
Mary
Geoge
```

这一种格式是以空格当作 i 这个变量的选择项目。也就是说，上面的$LIST 这个变量当中，以

空格键来分隔的时候，共可以分离出 4 个。所以，当以 do...done 语句处理时，就可以分别写出 4 个项。那么有没有办法利用这个特性来将 Linux 主机上的账号（account）显示出来？很简单，利用 cut 跟 sort 以及/etc/passwd 这个文件来完成该脚本，如下：

```
user@ubuntuer: test$ vi test15-for.sh
#!/bin/bash
# Using for and loop to read the account of this linux server!
# user 2008/05/20
account='cut -d ":" -f1 /etc/passwd|sort'
echo "The following is your linux server's account"
for i in $account
do
        echo $i
done
user@ubuntuer: test$ sh test15-for.sh
The following is your linux server's account
adm
aerosol
alenchan
amanda
apache
...
```

接下来以交互式实现循环功能。当输入 y 或 Y 时，程序就结束。可以这样写脚本：

```
user@ubuntuer: test$  vi test16-loop.sh
#!/bin/bash
# Using until
# user 2008/05/20
echo "Press Y/y to stop"
until ["$yn" = "Y" ] || ["$yn" = "y" ]
do
        read yn
done
echo "Stop here"
user@ubuntuer: test$ sh test16-for.sh
Press Y/y to stop
GDSG
A
Y
Stop here
```

上面脚本的意思是：当输入 Y 或者 y 时才跳出 do...done 的循环，否则执行接下来的程序段。

接下来学习一下所谓的逻辑判断式的使用方法。刚才已经讲过，可以使用条件判断来断定到底有没有文件（用-e ）或者该名称是属于目录或者文件（ -d -f ），下面设置一下这个流程。

（1）先查看一下/home/user/logical 这个名称是否存在。

（2）若不存在，则使用 touch 来创建该文件，创建完成后退出。

（3）如果存在的话，判断该名称是否为文件，若为文件则将它删除后创建同样名称的目录，然后退出。

(4) 如果存在的话，而且该名称为目录，则删除此目录。

看起来似乎很复杂，其实很简单，下面来试试看。

```
user@ubuntuer: test$ vi test17-ifthen.sh
#!/bin/bash
# using if and then to select file or directory
# user 2008/05/20
if [ ! -e logical ]; then
        touch logical
        echo "Just make a file logical"
        exit 1
elif [ -e logical ] && [ -f logical ]; then
        rm logical
        mkdir logical
        echo "remove file ==> logical"
        echo "and make directory logical"
        exit 1
elif [ -e logical ] && [ -d logical ]; then
        rm -rf logical
        echo "remove directory ==> logical"
        exit 1
else
        echo "Does here have anything?"
fi
```

18.5 脚本调试

当脚本出现问题后，如何调试 Shell 脚本呢？有没有办法不需要通过直接执行该脚本就可以来判断是否有问题呢？当然有。下面就直接以 sh 命令来进行判断。

语法

```
user@ubuntuer: test$ sh [-nvx] scripts
```

参数说明

- -n：不要执行脚本，查询脚本内的语法，若有错误则予以列出。
- -v：在执行脚本之前，先将脚本的内容显示在屏幕上。
- -x：将有使用到的脚本内容显示在屏幕上，与-v 略有不同。

示例

```
user@ubuntuer: test$ sh -n test01-hello.sh
user@ubuntuer: test$ sh -v test01-hello.sh
#!/bin/bash
# This program will print the "Hello! How are you" in your monitor
# user 2008/05/20
hello="Hello! How are you"
```

```
echo $hello
Hello! How are you
user@ubuntuer: test$ sh -x test01-hello.sh
+ hello=Hello! How are you
+ echo 'Hello!' How are you
Hello! How are you
```

熟悉 sh 的用法，将可以使您在管理 Linux 的过程中得心应手。

对于 Shell 脚本的学习需要多看、多模仿，并加以修改成己用，这是最快的学习手段了。网络上有相当多的朋友在开发一些很有用的脚本，若可以将对方的脚本拿来，并且改成适合自己应用的脚本，那么学习效果会更好。

18.6 课后练习

1. 什么是通配符？请列举几个最常用的通配符，如何在 Linux Shell 中巧用通配符？
2. 什么是管道？请列举几个常用管道命令。
3. 什么是重定向？在 Linux Shell 中如何通过重定向将一个命令操作的结果保存下来？
4. 什么是 Shell 脚本？
5. 请尝试通过 Shell 脚本实现几个简单的 Shell 应用实例。

Chapter19 网络管理

Ubuntu 作为 Linux 家族成员之一，它集成了功能强大的网络功能，包括高度可靠性以及较为安全的网络防御能力，并且提供了许多完善的网络设置工具，帮助用户轻松完成各种复杂的网络设置，实现任何所需要的网络服务。为了让 Ubuntu 系统能够连接并访问 Internet，应当正确设置网络组件，这个可以通过命令行的方式，也可以通过 Ubuntu 提供的图形界面工具来完成网络组件的设置。

本章将介绍 Ubuntu 网络连接管理，包括如何通过图形界面工具设置网络及常用的网络设置命令。希望读者建立图形配置工具和命令配置的对应关系。

19.1 网络工具

用户在使用网络时，常常需要查询了解自己当前的网络状态信息，如知道自己的 IP，查询端口占用情况等。在 Ubuntu 系统中，提供了图形界面的网络工具（gnome-nettool）。网络工具是 GNOME 组织开发提供的一个囊括了 ifconfig、ping、netstat、traceroute、port scanning、DNS lookup、finger、whois 等常用命令的图形化用户界面的网络信息工具，由于其功能比较全面、操作便利，对于普通用户来说是相当受用的。

19.1.1 网络工具启动

网络工具（gnome-nettool）的启动可以通过两种方式进行。

- 执行【系统】|【系统管理】|【网络工具】命令。
- 在命令行下输入 gnome-nettool。

启动后的【设备-网络工具】窗口包括 8 个标签页：设备、Ping、网络统计、Traceroute、端口扫描、查阅、Finger 和 Whois。下面我们就以几个小节来谈谈如何通过网络工具（gnome-nettool）获取相关网络信息。

19.1.2 网络设备

查看网络设备的基本状况是了解网络设备运行的最基本操作，网络工具（gnome-nettool）的【设备】标签页为用户提供了命令 ifconfig 的图形操作界面，它帮助用户查看网络设备信息，并可以完成 IP 地址配置。如图 19.1 所示。

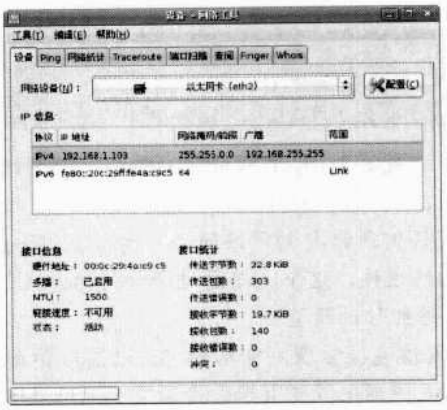


图 19.1 查看网络设备

从上图中，读者可以进行以下操作：

- 单击【网络设备】微调框，打开下拉列表，列表中显示的是当前系统安装的网络设备接口。
- 从下拉列表中选定需要查看的设备，在图形界面下方将显示选定设备的 IP 地址、接口信息和接口统计。
- 如果要为该网卡重新分配 IP 地址，则单击【配置】按钮，在弹出的对话框中设置新的 IP 地址，如何进行 IP 地址的设置，将在下面的章节中介绍。

19.1.3 测试网络的物理连通

通常情况下，如果主机连不上网，第一个反应就是网络是否断掉了？这个时候，可以使用网络工具（gnome-nettool）的 Ping 标签页来测试网络的物理连接正常与否。这个界面的功能跟命令行下的 ping 命令类似，只是把 ping 命令得到的统计数据用图形界面的方式来展现出来，如图 19.2 所示。

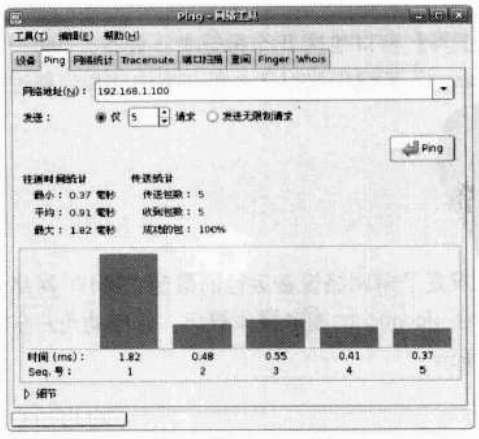


图 19.2 测试网络的物理连通

在上图中可以进行如下操作：

- 在【网络地址】下拉列表框中输入一个实际存在的测试目标的 IP 地址或域名。
- 在【发送】选项中输入发送测试请求的次数（默认情况下是 5 次），也可以选择【发送限制请求】。
- 单击 Ping 按钮，开始测试目标 IP 地址或域名是否可连通。
- 在测试结束后，对话框下半部分就显示相关的统计数据，包括往返时间统计、传送统计和传输数据的柱状图。
- 单击【细节】可以查看更加详细的数据。

19.1.4 网络统计

网络工具（gnome-nettool）的【网络统计】标签页为用户提供了一个方便地获取网络统计信息的方式，可以查看的网络统计信息有以下三项：

- 路由表信息
- 激活网络信息
- 多播信息

查看网络统计信息的具体操作如下：

Step 01 在【网络统计】标签页中，单击【路由表信息】单选按钮（或【激活网络服务】，或【多播信息】）。

Step 02 单击【网络统计】按钮，统计完毕后，在对话框下半部分显示相应的统计信息，如图19.3是路由表信息统计，图19.4是激活网络服务统计，图19.5是多播信息统计。

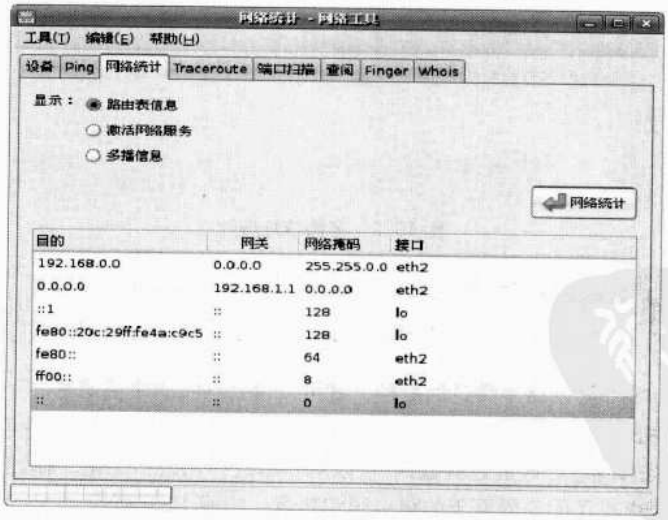


图 19.3 路由表信息统计

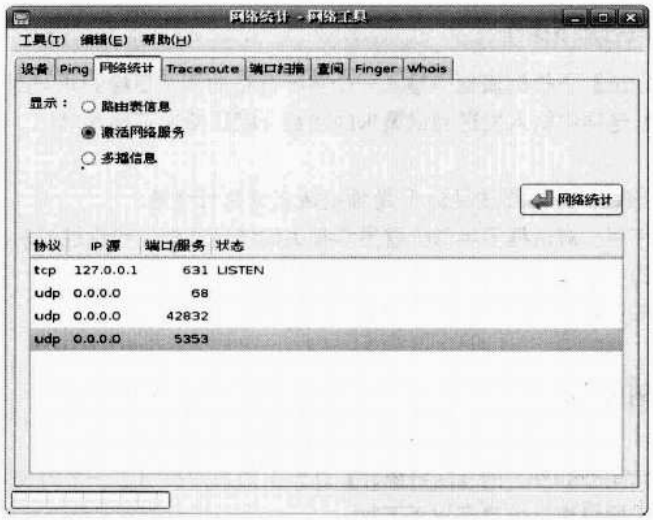


图 19.4 激活网络服务统计

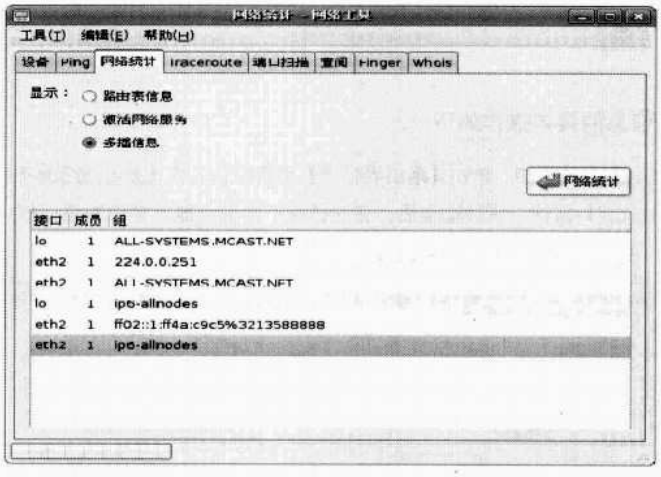


图 19.5 多播信息统计

19.1.5 路由跟踪

跟踪数据包在 Internet 网络中传递的路径，就可以得到网络的路由信息。Traceroute 就是这样一个工具，在网络的物理连接正常的情况下，它可以跟踪数据包的传输路径，并得到经过的网络节点信息和 IP 数据包的统计数据。通过这些数据，可以对网络进行调试和优化。网络工具(gnome-nettool)的 Traceroute 标签页提供了图形界面下的路由跟踪功能，如图 19.6 所示。

图 19.6 跟踪路由

具体操作如下：

Step 01 在【网络地址】下拉列表框内输入目标跟踪IP地址或域名。

Step 02 单击【跟踪】按钮，开始记录数据包发送的目标IP地址或域名所经过的路由器。

然而，像 ping 工具一样，很多网络管理员出于安全考虑，刻意阻止 Traceroute 的访问，防止大量的测试数据包造成网络拥堵。

19.1.6 端口扫描

当用户希望了解目标主机提供了哪些服务的时候，可以扫描目标主机端口。网络工具（gnome-nettool）的【端口扫描】标签页为用户提供了扫描目标主机端口的图形界面，如图 19.7 所示。

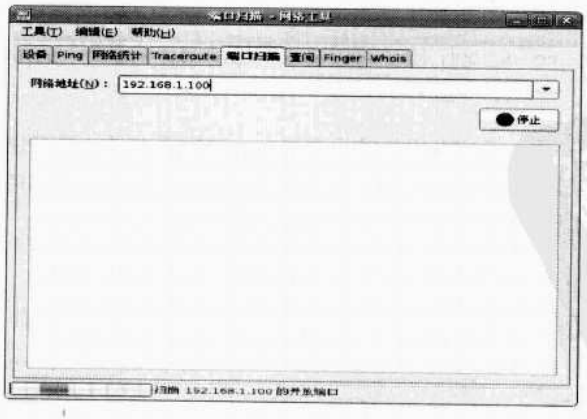


图 19.7 端口扫描

具体操作如下：

Step 01 在【网络地址】下拉列表框内输入目标跟踪IP地址或域名。
Step 02 单击【扫描】按钮，开始扫描端口。

19.1.7 查阅域名信息

网络工具（gnome-nettool）的【查阅】标签页为用户提供了一个查询域名相关信息的图形界面，它的功能跟命令行下的 nslookup 命令类似，如图 19.8 所示。

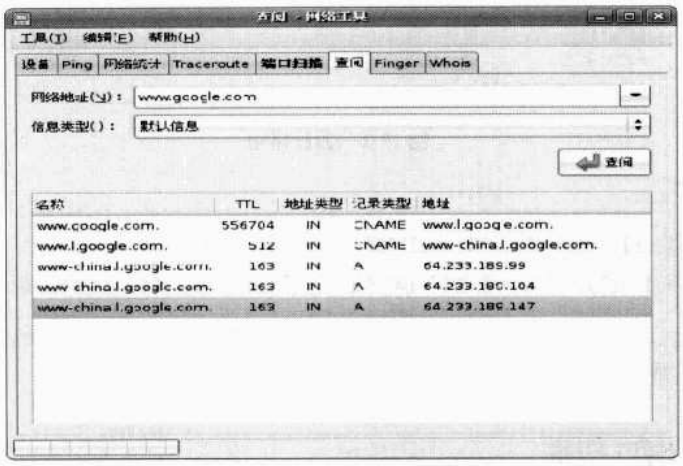


图 19.8 查阅域名信息

具体操作如下：

Step 01 在【网络地址】下拉列表框内输入目标域名，并选择域名解析返回的信息类型。
Step 02 单击【查阅】按钮，开始获取域名的相关信息。

使用网络工具（gnome-nettool）除了可以获取网络的相关信息外，还可以查询用户和域名注册信息。

- 在 Finger 标签页中，可以获得系统用户信息，包括用户名、是否登录、用户目录等。
- 在 Whois 标签页中，可以查询域名是否已经被注册，以及域名的相关信息，包括域名所有人、域名注册商、域名注册日期和过期日期等。

19.2 网络设置工具

在 Ubuntu 的使用过程中，网络设置可能会因环境的改变如从一个局域网搬到另一个局域网，或系统硬件的改变如更换网卡，而需要进行重新调整或设置。Ubuntu 为方便用户可视化操作，提供了专门的网络设置工具。本节就来介绍 Ubuntu 网络设置工具（network-admin）。

19.2.1 启动网络设置工具

网络设置工具（network-admin）的启动方式有以下两种。

- 执行【系统】|【系统管理】|【网络】命令。
- 在命令行下输入 sudo network-admin。

用户打开【网络设置】后，有时候并不能直接进行操作，需要用户进行“解锁”操作：单击右下角的【解锁】按钮，然后在系统弹出的窗口中选择操作用户并输入密码，确认后，才能看到如图 19.9 所示的操作界面。

图 19.9 网络设置

【网络设置】中包括 4 个标签页：连接、常规、DNS 和主机。用户可以通过该工具设置 IP 地址、设置主机信息和管理域名服务器地址等。

19.2.2 设置 IP

在 Internet 上有千百万台主机，为了区分这些主机，人们给每台主机都分配了一个专门的地址，称为 IP 地址。通过 IP 地址就可以访问到每一台主机。IP 地址由 4 部分数字组成，每部分数字对应于 8 位二进制数字，各部分之间用小数点分开。如某一台主机的 IP 地址为 211.157.65.169。Internet IP 地址由 NIC (Network Information Center) 统一负责全球地址的规划、管理，同时由 Inter NIC、APNIC、RIPE 三大网络信息中心具体负责美国及其他地区的 IP 地址分配。

IP 地址分为静态 IP 和动态 IP：

- 静态 IP：静态 IP 地址是长期固定分配给一台计算机使用的 IP 地址，一般是特殊的服务器才拥有静态 IP 地址。
- 动态 IP：因为 IP 地址资源非常短缺，通过电话拨号上网或普通宽带上网的用户一般不具备固定 IP 地址，而是由 DHCP 网络服务动态分配一个暂时的 IP 地址。普通人一般不需要去了解动态 IP 地址，这些都是计算机系统自动完成的。

设置静态 IP

与动态 IP 相比，此方式可以节省获取动态 IP 的时间。下面介绍一下如何进行静态 IP 地址的设置。假设现在主机需要的网络静态 IP 参数如下所示：

```
Hostname  Ubuntuer

IP:       192.168.1.10
Netmask   255.255.255.0
Gateway   192.168.1.100

DNS IP    210.82.5.1
```

配置静态 IP 的步骤如下：

Step 01 在【网络设置】对话框中选择【连接】标签页。
Step 02 选中需要配置IP地址的网卡，单击【属性】按钮，打开网卡属性对话框。
Step 03 单击【配置】微调按钮，选择【静态IP地址】选项。
Step 04 根据上面的网络静态IP参数表单，把参数输入到相应的文本框中，如图19.10所示。
Step 05 单击【确定】按钮，完成设置。

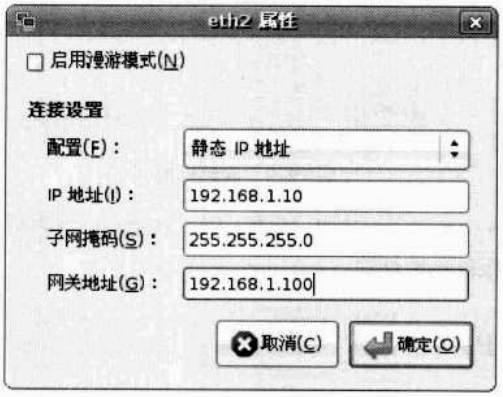


图 19.10 静态 IP 地址设置

设置动态 IP

如果需要设置动态 IP，那么前提是确认所处的网络提供了 DHCP 服务，因为这个 DHCP 服务将会自动地为主机分配 IP 地址、子网掩码、路由表和 DNS 服务器地址。

配置动态 IP 的步骤如下：

Step 01 在【网络设置】对话框中选择【连接】标签页。
Step 02 选中需要配置IP地址的网卡，单击【属性】按钮，打开网卡属性对话框。
Step 03 单击【配置】微调按钮，选择【自动配置(DHCP)】选项，如图19.11所示。
Step 04 单击【确定】按钮，完成设置。

图 19.11　动态 IP 地址设置

19.2.3　设置主机常规信息

通常情况下，主机命名采用层次树状结构的方式，包括主机名和域名。主机名是赋予主机的一个唯一识别的名字，域名是树状结构中层次节点的名字，可以划分为二级域名、三级域名等。我们可以通过网络设置工具（network-admin）来设置主机命名信息。

具体操作步骤如下：

Step 01　在【网络设置】对话框中选择【常规】标签页。

Step 02　分别在【主机名】、【域名】文本框中填入主机信息，如图19.12所示。

Step 03　当光标焦点离开【主机名】文本框后，系统将保存主机名。

图 19.12　主机信息设置

19.2.4　设置 DNS

DNS 域名解析服务器可以在大范围的计算机网络提供域名到 IP 地址的转换。网络中的每台计算机都是一个 DNS 客户端，向 DNS 服务器提交域名解析的请求，由 DNS 服务器完成域名到 IP 地址的

映射。因此，DNS 客户端至少需要有一个 DNS 服务器地址，作为命名解析的开始。下面介绍一下如何使用网络设置（network-admin）来设置 DNS。

具体操作步骤如下：

Step 01 在【网络设置】对话框中选择DNS标签页，如图19.13所示。

Step 02 单击【DNS服务器】列表框右侧的【添加】按钮，这时在【DNS服务器】列表框中出现输入提示，直接输入DNS服务器IP地址后，按回车键确认。

图 19.13 DNS 设置

如果要删除【DNS 服务器】列表框中的一个 IP 地址，选择该记录，然后单击右侧的【删除】按钮，完成删除操作。

19.2.5 基于 Host 列表的主机名解析

基于 Host 列表的主机名解析，就是将网络中的主机名与相应的 IP 地址的映射关系保存在本地的列表中。当用户使用其中一个主机名时，计算机就会迅速查找，将这个主机名映射为相应的 IP 地址。

假设现在有一台 Ubuntu Linux 主机，主机名是 Ubuntuer，IP 地址是 192.168.1.103，那么添加这个主机名映射关系的操作步骤如下：

Step 01 在【网络设置】对话框中选择【主机】标签页，如图19.14所示。

Step 02 单击右侧的【添加】按钮，打开【主机别名属性】对话框，如图19.15所示。

Step 03 在【IP地址】、【别名】文本框中填入别名解析信息，然后单击【确定】按钮，完成添加操作。

图 19.14　基于 Host 列表的主机名解析

图 19.15　主机别名属性

19.3　常用网络命令

通过上面网络工具和网络设置工具的介绍，对于初级用户来说，应该基本可以完成 Ubuntu 中与网络相关的操作了。其实 Ubuntu 作为 Linux 的一个发行套件，它还提供极其强大的网络管理相关的 Shell 命令，通过这些 Ubuntu 网络管理命令可以实现所有的网络管理功能。本节中，就来讲述 Shell 环境下几个最常用的网络操作命令。

19.3.1　ifconfig 指令

语法

```
user@ubuntuer: ~$ ifconfig interface
user@ubuntuer: ~$ ifconfig interface [options]
```

参数说明

- interface：网卡代号，例如 eth0，eth1 等。
- options：主要有以下几个参数。
  - network：网段
  - broadcast：广播网段
  - netmask：子网掩码
  - up|down：启动|关闭网络接口

示例

```
user@ubuntuer: ~$ ifconfig
#这个指令在没有加上网络卡时，会将所有的网络接口内容显示出来
eth0      Link encap:Ethernet  HWaddr 00:0c:29:4a:c9:c5
          inet addr:192.168.1.103  Bcast:192.168.1.255  Mask:255.255.255.0
          inet6 addr: fe8-::20c:29ff:fe4a:c9c5/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:287 errors:0 dropped:0 overruns:0 frame:0
          TX packets:581 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:31861 (31.1 KB) TX bytes:49179 (48.0 KB)
          Interrupt:18 Base address:0x2000

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:3258 errors:0 dropped:0 overruns:0 frame:0
          TX packets:3258 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:170063 (166.0 Kb) TX bytes:170063 (166.0 Kb)

user@ubuntuer: ~$ sudo ifconfig eth0 192.168.1.102 netmask 255.255.255.0
<==上面再将 eth0 这个网络接口的 IP 地址修改为 192.168.1.102
user@ubuntuer: ~$ ifconfig eth0
# 将修改完的网卡信息显示出来
eth0      Link encap:Ethernet  HWaddr 00:0c:29:4a:c9:c5
          inet addr:192.168.1.102  Bcast:192.168.1.255  Mask:255.255.255.0
          inet6 addr: fe8-::20c:29ff:fe4a:c9c5/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:287 errors:0 dropped:0 overruns:0 frame:0
          TX packets:581 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:31861 (31.1 KB) TX bytes:49179 (48.0 KB)
          Interrupt:18 Base address:0x2000

user@ubuntuer: ~$ sudo ifconfig eth0 down <==关闭 eth0 网卡
user@ubuntuer: ~$ sudo ifconfig eth0 up <==启动 eth0 网卡
```

说明

ifconfig 指令可以用来配置网络接口的 IP 地址、掩码、网关、物理地址等。值得一提的是用 ifconfig 为网卡指定 IP 地址，这只是用来调试网络用的，并不会更改系统关于网卡的配置文件。另外 ifconfig 的最大用处其实来自于可以查看网卡的参数，所以最常使用的就是直接输入 ifconfig 命令或者是 ifconfig eth0 等的用法，下面介绍一下上面几个参数的简单意义。

- eth0：表示网卡的代号，eth0 代表第一个网卡，eth1 代表第二个网卡，以此类推。另外还有以下几种接口类型（X 表示接口号）。
  - pppX：调制调解设备
  - wlanX：无线网卡
  - trX：令牌环网卡
- lo：表示为内部循环 IP 的网卡代号，需要注意的是这个内部的 interface 一定要存在，不能随便关掉它。
- HWaddr：网卡的硬件地址。
- inet addr：即网卡的 IP。
- Bcast：广播（broadcast）的地址。
- Mask：子网掩码。
- MTU：Maximum Trasmission Unit 最大传输单元(字节)，即此接口一次所能传输的最大数据包，这个数值并非越大越好，也非越小越好，需要根据实际情况设定一个合理值。
- RX：网络由启动到目前为止的数据接收状况。
- TX：网络由启动到目前为止的数据上传状况。
- collisions：网络信号冲突状况。
- txqueuelen：传输缓冲区长度大小。
- Interrupt：IRQ 中断地址。
- Base address：I/O 地址。这个 IRQ 与 I/O 在网卡上面是可以设定的。可以在/etc/lilo.conf 中来设定。如果用户的主机上有多张网卡的时候，就用的着它了。

19.3.2 route 指令

语法

```
user@ubuntuer: ~$ route [-nee]
user@ubuntuer: ~$ route add [-net|-host] 目标主机或网域 [netmask] [gw|dev]
user@ubuntuer: ~$ route del [-net|-host] 目标主机或网域 [netmask] [gw|dev]
```

参数说明

- -n：信息以 IP 来显示。
- -ee：显示较长列的信息。
- add：增加路由。
- net：删除一个路由。

- –net：增加一个网络的路由。
- –host：增加到某个 IP 主机的路由。
- netmask：网络掩码。
- gw：网络网关。
- dev：路由的网络接口。

示例

```
user@ubuntuer: ~$ route
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
192.168.0.0     *               255.255.0.0     U     0      0        0 eth0
default         192.168.1.1     0.0.0.0         UG    0      0        0 eth0

user@ubuntuer: ~$ route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
192.168.0.0     0.0.0.0         255.255.0.0     U     0      0        0 eth0
0.0.0.0         192.168.1.1     0.0.0.0         UG    0      0        0 eth0
# 加上 -n 的时候显示速度较快，此外，default gateway 就是 0.0.0.0

user@ubuntuer: ~$ route add -net 192.168.0.0 netmask 255.255.255.0 dev eth0
# 新增一个路由

user@ubuntuer: ~$ route del -net 192.168.0.0 netmask 255.255.255.0 dev eth0
# 删除一个路由

user@ubuntuer: ~$ route add default gw 192.168.1.100
# 增加一个默认网关
# 注意一下，gw 后面接的是 IP，而 dev 后面则是网卡号
```

说明

不带任何参数的 route 命令或 route –n 的主要用途是查看当前网络的路由表，包括所在子网地址和默认网关地址。下面谈一谈使用 route 时，显示内容的意义。

- Destination：目标地址，可以是 IP 也可以是网域。至于没有规定到的，则以 default 来表示。
- Gateway：目标地址经由哪一个网关发送。
- Genmask：该 Destination 的子网掩码。
- Flages：路由标志，有以下几种。
  - U (route is up)：该路由处于启动状态。
  - H (target is a host)：网关为主机。
  - G (use gateway)：此网关为路由器。
  - R (reinstate route for dynamic routing)：表示使用动态路由重新初始化的路由。
  - D (dynamically installed by daemon or redirect)：已经由守护进程或重定向功能设定为动态路由。

- M (modified from routing daemon or redirect)：路由已经被守护进程或重定向功能修改了。
- ! (reject route)：此路由处于关闭状态。

- Iface：使用的网络接口。

route 命令的另一个主要用途是修改、添加或删除路由，由上面的例子就可以知道，操作很简单。不过需要注意的是，当执行 route 命令的时候发现，显示的速度很慢时，通常是路由信息存在一定的问题。那么就得仔细地检查一下设定，否则网络速度会有迟滞。

19.3.3 ping 指令

语法

```
user@ubuntuer: ~$ ping [-b broadcast]
user@ubuntuer: ~$ ping [-c number] host
```

参数说明

- -b broadcast：当要 ping 一个网段时，可以使用这个方式来广播。
- -c number：后面加上 number (数字)可以限制 ping 的次数。

示例

```
user@ubuntuer: ~$ ping -c 5 192.168.1.100
PING 192.168.1.100 (192.168.1.100) 56(84) bytes of data.
64 bytes from 192.168.1.100: icmp_seq=1 ttl=128 time=1.44 ms
64 bytes from 192.168.1.100: icmp_seq=2 ttl=128 time=0.647 ms
64 bytes from 192.168.1.100: icmp_seq=3 ttl=128 time=0.564 ms
64 bytes from 192.168.1.100: icmp_seq=4 ttl=128 time=0.705 ms
64 bytes from 192.168.1.100: icmp_seq=5 ttl=128 time=8.20 ms

--- 192.168.1.100 ping statistics ---
5 packets transmitted, 5 packets received, 0% packet loss time 3999ms
Rtt min/avg/max/mdev = 0.564/2.312/8.200/2.960 ms
```

说明

从上的例子可以看出，ping 命令发出的 5 个测试包都得到了目标主机的应答，说明目标主机是连通的。不过由于目前很多主机设置防火墙，对 ping 命令不予应答。在这种情况下，ping 命令发送的测试包，就如同泥牛入海，有去无回了。

需要说明的是，在例子的最下边，min/avg/max/mdev 是 ping 命令测试完成后的统计结果，分别表示最小响应时间/平均响应时间/最大响应时间/均值，这些信息整体反映了网络的好坏。除此之外，例子中一些陌生名称的意思如下所述。

- icmp：指的是 ICMP 这个协议。
- ttl：指的是 time to live，当经过一个节点，ttl 就会减少 1，而默认值有 128 个，以上面的示例为例，主机连接到 192.168.1.100 共经过 0 个 gateway，所以 ttl 还是 128。

19.3.4 traceroute 指令

语法

```
user@ubuntuer: ~$ traceroute [-i interface] [-g gateway] [host|IP]
```

参数说明

- -i：使用这个网络接口来连出去，例如 eth0，ppp0 等。
- -g：使用这个网关来连出去，例如 192.168.1.2，140.116.141.29 等。

示例

```
user@ubuntuer: ~$ traceroute www.google.com
traceroute: Warning: www.google.com has multiple addresses; using 64.233.189.104
traceroute to www-china.l.google.com (64.233.189.104), 30 hops max, 38 byte
 packets
 1  10.64.20.254 (10.64.20.254) 0.531 ms  0.475 ms  0.400 ms
 2  123.127.245.129 (123.127.245.129) 1.325 ms  1.562 ms  1.484 ms
 3  202.106.58.81 (202.106.58.81) 5.702 ms  6.028 ms  5.692 ms
 4  61.148.155.93 (61.148.155.93) 5.772 ms  3.965 ms  3.846 ms
 5  61.148.156.229 (61.148.156.229) 4.383 ms  4.171 ms  4.285 ms
 6  202.96.12.57 (202.96.12.57) 4.927 ms  5.252 ms  4.703 ms
 7  219.158.4.46 (219.158.4.46) 59.555 ms  59.185 ms  59.055 ms
 8  219.158.3.246 (219.158.3.246) 58.789 ms  58.842 ms  58.644 ms
 9  219.158.3.130 (219.158.3.130) 52.091 ms  51.822 ms  52.138 ms
10  219.158.32.230 (219.158.32.230) 62.870 ms  68.398 ms  75.988 ms
11  64.233.175.207 (64.233.175.207) 65.197 ms  65.055 ms  64.988 ms
12  66.249.94.34 (66.249.94.34) 54.407 ms 66.249.94.6 (66.249.94.6) 69.654 ms
    66.249.94.34 (66.249.94.34) 61.934 ms
13  hk-in-f104.google.com (64.233.189.104) 65.414 ms  66.083 ms  65.412 ms
```

说明

traceroute 相当有用，它可以用来判断当一部远程主机无法联机时候，到底数据包是停顿在哪一个节点上，因为很多时候会发现网络速度突然间变慢了，而主机似乎没有问题，这个时候就可以使用这个命令检查一下，是否在联机的过程当中，有些节点被挡下来了。跟 ping 一样，如果网络被防火墙保护的话，traceroute 命令就会显示一大串的***了，这个时候就没法测试了。

19.3.5 netstat 指令

语法

```
user@ubuntuer: ~$ netstat [-r] [-i interface]
user@ubuntuer: ~$ netstat [-antulp]
```

参数说明

- -r：显示出路由信息。

- -i：显示出网络接口列表，跟 ifconfig 类似。
- -a：显示出目前所有的网络联机状态。
- -n：默认情况下，显示出的 host 会以主机名称来显现。若为 n 则可以使 port 与 host 都以数字显示。
- -t：仅显示 tcp 连接状态。
- -u：仅显示 udp 连接状态。
- -l：仅显示 LISTEN 的内容。
- -p：同时显示此连接的 PID。

示例

```
user@ubuntuer: ~$ netstat -r
#显示出目前的路由信息，与 route 命令的功能相同
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
192.168.0.0     *               255.255.0.0     U     0      0        0 eth0
default         192.168.1.1     0.0.0.0         UG    0      0        0 eth0

user@ubuntuer: ~$ netstat -i eth0
# 看看下面显示出的内容，是否跟 ifconfig eth0 类似
Kernel Interface table
Iface   MTU Met   RX-OK RX-ERR RX-DRP RX-OVR    TX-OK TX-ERR TX-DRP TX-OVR Flg
eth0   1500   0   615      0      0      0      2458      0      0      0 BMRU
lo    16436   0   534      0      0      0       534      0      0      0 LRU

user@ubuntuer: ~$ netstat -an  <==显示所有的联机状态，并且以数字形态显示
user@ubuntuer: ~$ netstat -tul <==显示 LISTEN 的及 tcp 与 udp 的联机状态，如下：
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 *:mysql                 *:*                     LISTEN
tcp        0      0 *:sunrpc                *:*                     LISTEN
tcp        0      0 *:http                  *:*                     LISTEN
tcp        0      0 *:607                   *:*                     LISTEN
tcp        0      0 *:8009                  *:*                     LISTEN
tcp        0      0 *:2222                  *:*                     LISTEN
tcp        0      0 localhost.localdomain:8015  *:*                 LISTEN
tcp        0      0 *:webcache              *:*                     LISTEN
udp        0      0 *:601                   *:*
udp        0      0 *:604                   *:*
udp        0      0 *:sunrpc                *:*
# 注意：上面的 LISTEN 表示该 port 是已经在监听网络服务，而左边的 tcp 指的是 tcp 数据包

user@ubuntuer: ~$ netstat -anp | more <==这个命令方式经常使用
```

说明

netstat 是经常使用的命令，它可以让我们了解目前主机的连接状态，包括网络连接、路由表、接口统计信息、最大分组传输单位等非常有价值的信息，所以一定得好好学习使用。

另外，-p 参数很有用，尤其是在一些莫名其妙的连接出现时，可以用-p 这个参数查到 PID，

然后再用 kill 来终止。

19.3.6 host 指令

语法

```
user@ubuntuer: ~$ host [-a] domain_name
```

参数说明

-a：显示出所有的信息。

示例

```
user@ubuntuer: ~$ host www.google.com  <==仅显示出主机的 IP
www.google.com is an alias for www.l.google.com.
www.l.google.com is an alias for www-china.l.google.com.
www-china.l.google.com has address 64.233.189.147
www-china.l.google.com has address 64.233.189.99
www-china.l.google.com has address 64.233.189.104

user@ubuntuer: ~$ host -a www.google.com <==显示所有的主机信息
Trying "www.google.com"
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 41494
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 4, ADDITIONAL: 4

;; QUESTION SECTION:
;www.google.com.                    IN      ANY

;; ANSWER SECTION:
www.google.com.         178403  IN      CNAME   www.l.google.com.

;; AUTHORITY SECTION:
google.com.             79388   IN      NS      ns2.google.com.
google.com.             79388   IN      NS      ns3.google.com.
google.com.             79388   IN      NS      ns4.google.com.
google.com.             79388   IN      NS      ns1.google.com.

;; ADDITIONAL SECTION:
ns1.google.com.         185295  IN      A       216.239.32.10
ns2.google.com.         204906  IN      A       216.239.34.10
ns3.google.com.         263915  IN      A       216.239.36.10
ns4.google.com.         44061   IN      A       216.239.38.10

Received 188 bytes from 224.95.193.97#53 in 67 ms
```

说明

host 功能跟 nslookup 几乎是相同的，但是 nslookup 功能更多。使用这个命令时，系统会自动去找 /etc/resolv.conf 下设定的 DNS 的 IP，然后根据该 IP 来检查所想要知道的主机对应的 IP。

19.3.7 nslookup 指令

语法

```
user@ubuntuer: ~$ nslookup [domain_name|IP]
```

示例

```
user@ubuntuer: ~$ nslookup www.google.com <==由域名查询 IP
Server:         211.95.193.97
Address:        224.95.193.97#53

Non-authoritative answer:
www.google.com  canonical name = www.l.google.com.
www.l.google.com        canonical name = www-china.l.google.com.
Name:   www-china.l.google.com
Address: 64.233.189.99
Name:   www-china.l.google.com
Address: 64.233.189.104
Name:   www-china.l.google.com
Address: 64.233.189.147Non-authoritative answer:
Name:   tw.yahoo.com
Address: 202.1.237.21
user@ubuntuer: ~$ nslookup 64.233.189.99 <==由 IP 查询域名
Server:         211.95.193.97
Address:        224.95.193.97#53

Non-authoritative answer:
99.189.233.64.in-addr.arpa      name = hk-in-f99.google.com.
Authoritative answers can be found from:
189.233.64.in-addr.arpa nameserver = ns1.google.com.
189.233.64.in-addr.arpa nameserver = ns2.google.com.
189.233.64.in-addr.arpa nameserver = ns3.google.com.
189.233.64.in-addr.arpa nameserver = ns4.google.com.
ns1.google.com  internet address = 216.239.32.10
ns2.google.com  internet address = 216.239.34.10
ns3.google.com  internet address = 216.239.36.10
ns4.google.com  internet address = 216.239.38.10
```

说明

这个命令就如同前面的 host 提到的，就是正查反查的命令，也是利用/etc/resolv.conf 的内容来查询。

19.4 网络配置文件

在 Linux 系统中，TCP/IP 网络是通过若干个文本文件进行配置的，需要编辑这些文件来完成联

网工作。系统中有关网络配置的重要文件有：

- /etc/sysconfig/network
- /etc/HOSTNAME
- /etc/hosts
- /etc/services
- /etc/host.conf
- /etc/nsswitch.conf
- /etc/resolv.conf
- /etc/rc.d/init.d/network

这些文件都可以在系统运行时进行修改，大多不用启动或者停止任何程序，更改会立刻生效(其中/etc/sysconfig/network 需要重启服务)。另外，这些文件都支持由#开头的注释，每一个文件都在 Unix 手册页中的第 5 部分中有一项，可以用 man 命令来获取它们。接下来将对这些文件进行逐一介绍。

19.4.1 网络设置 /etc/sysconfig/network

/etc/sysconfig/network 文件用来指定服务器上的网络配置信息，包括 IP 地址、掩码、网络、广播地址和默认路由器的变量等控制和网络有关的文件和守护程序的行为的参数。下面是一个示例文件。

```
NETWORKING=yes
HOSTNAME=machine1
GATEWAY=210.34.6.2
FORWARD_IPV4=yes
GATEWAYDEV=
```

其中：

NETWORK=yes/no 表示网络是否被配置；
HOSTNAME=hostname hostname 表示服务器的主机名；
GATEWAY=gw-ip gw-ip 表示网络网关的 IP 地址；
FORWARD_IPV4=yes/no 表示是否开启 IP 转发功能；
GAREWAYDEV=gw-dev gw-dw 表示网关的设备名，如 eth0 等。

为了和一些已有软件相兼容，/etc/HOSTNAME 文件应该用和 OSTNAME=hostname 相同的主机名。

19.4.2 主机名/etc/HOSTNAME

/etc/HOSTNAME 文件包含了系统的主机名称，包括完全的域名，如：

```
192.168.0.1 machine1.domain machine1
```

这个文件是在启动时从文件/etc/sysconfig/network 中的 HOSTNAME 行中得到的，用于在启动时设置系统的主机名。

19.4.3 IP 地址和主机名的映射/etc/hosts

/etc/hosts 中包含了 IP 地址和主机名之间的映射，还包括主机名的别名。IP 地址的设计使计算机容易识别，但对于人却很难记住它们，为了解决这个问题，创建了/etc/hosts 这个文件。下面是一个示例文件：

```
127.0.0.1 machine1 localhost.localdomain localhost
192.168.1.100 machine7
192.168.1.101 otherpc otheralias
```

在这个示例中，本机名是 machine1，otherpc，还有别名 otheralias，它可以指向 otheralias。一旦配置完机器的网络配置文件，应该重新启动网络以使修改生效，使用下面的命令来重新启动网络：

```
/etc/rc.d/init.d/network restart
```

/etc/hosts 文件通常含有主机名、localhost 和系统管理员经常使用的系统别名，有时候 telnet 到 Linux 机器要等待很长时间，可以通过在/etc/hosts 加入客户机的 IP 地址和主机名的匹配项，来减少登录等待时间。在没有域名服务器的情况下，系统上的所有网络程序都通过查询该文件来解析对应于某个主机名的 IP 地址，否则，其他的主机名通常使用 DNS 来解决，DNS 客户部分的配置在文件 /etc/resolv.conf 中。

19.4.4 服务与端口映射 /etc/services

/etc/services 中包含了服务名和端口号之间的映射，不少系统程序要使用这个文件，下面是 RedHat 安装时默认的/etc/services 中的前几行：

```
cpmux 1/tcp # TCP port service multiplexer
echo 7/tcp
echo 7/udp
discard 9/tcp sink null
discard 9/udp sink null
systat 11/tcp users
```

最左边一列是主机服务名，中间一列是端口号，/后面是端口类型，可以是 TCP 也可以是 UDP。任何后面的列都是前面服务的别名。在这个文件中也存在着别名，它们出现在端口号后面，在上述例子中 sink 和 null 都是 discard 服务的别名。

19.4.5 配置名字解析器 /etc/host.conf

有两个文件声明系统到哪里寻找名字信息来配置 Unix 名字解析器的库。文件/etc/host.conf 由版本 5 的 libc 库所使用，而/etc/nsswitch.conf 由版本 6 使用(glibc) 。问题在于一些程序使用其中

一个，而一些使用另一个，所以将两个文件都配置正确是必要的。

/etc/host.conf 文件指定如何解析主机名，Linux 通过解析器库来获得主机名对应的 IP 地址。

```
order hosts, bind
multi on
```

- order：指定主机名查询顺序，其参数为用逗号隔开的查找方法，支持的查找方法为 bind、hosts 和 nis，分别代表 DNS、/etc/hosts 和 NIS，这里规定先查询/etc/hosts 文件，然后再使用 DNS 来解析域名。
- trim：表明当通过 DNS 进行地址到主机名的转换时，域名将从主机名中被裁剪掉，trim 可以被多个域包含多次，对/etc/hosts 和 NIS 查询方法不起作用，注意在/etc/hosts 和 NIS 表中主机名是被适当地(有或没有全域名)列出的。
- multi：指定/etc/hosts 文件中指定的主机是否可以有多个地址，值为 on 表示允许拥有多个 IP 地址的主机一般称为具有多个网络界面。
- nospoof：指是否允许对该服务器进行 IP 地址欺骗，值为 on 表示不允许。IP 欺骗是一种攻击系统安全的手段，通过把 IP 地址伪装成别的计算机，来取得其他计算机的信任。
- alert：当 nospoof 指令为 on 时，alert 控制欺骗的企图是否用 syslog 工具进行记录，值为 on 表示使用，默认值为 off。
- rccorder：如果被设置为 on，所有的查询将被重新排序，所以在同一子网中的主机将首选被返回，默认值为 off。

19.4.6 配置名字解析器 /etc/nsswitch.conf

/etc/nsswitch.conf 文件是由 Sun 公司开发并用于管理系统中多个配置文件查找的顺序，它提供了比/etc/host.conf 文件更多的功能。/etc/nsswitch.conf 中的每一行或者是注释(以#号开头)或者是一个关键字后跟冒号和一系列要试用的有顺序的方法。每一个关键字是在/etc/目录下可以被/etc/nsswitch.conf 控制的/etc 文件的名字。下面是可以被包含的关键字。

- aliases：邮件别名；
- passwd：系统用户；
- group：用户组；
- shadow：隐蔽口令；
- hosts：主机名和 IP 地址；
- networks：网络名和号；
- protocols：网络协议；
- services：端口号和服务名称；
- ethers：以太网号；
- rpc：远程进程调用的名称和号；
- netgroup：网内组。

下面也是可以包含的关键字：

- files：除了 netgroup，对其他关键字都有效。在相应的/etc 文件中寻找记录。
- db：除了 netgroup，对其他关键字都有效。在相应的/var/db 数据库中寻找记录。对长文件很有效，如 passwd 文件已经超过 500 项。要从标准/etc 文件中产生这些文件，应改变目录到/var/db 并运行 run 命令。
- compat：兼容性模式，对 passwd、group 和 shadow 文件有效。在本模式中，将先在对应的/etc 文件中查找。如果想进行 NIS 查找，需要第一个值(用户名或组名)为加号（+），后面跟对应数量的冒号（:）（/etc/passwd 为 6 个，/etc/group 为 3 个，/etc/shadow 为 8 个）。如在/etc/passwd 文件中，下面一行应被包含在文件尾。

```
+: *: :: : :
```

- dns：只对 hosts 有意义。像在/etc/resolvconf 配置的，在 DNS 中进行查找。
- nis：对所有的关键字都有意义。如 NIS 是可以用的，在 NIS 服务器中查找。
- [STATUS = action]：控制名字服务的行为。STATUS 是 SUCCESS(操作被成功执行)、NOTFOUND（记录没找到）、UNAVAIL（所选择的服务不可用)和 TRYAGAIN（服务暂时不可用，请重试)中的一个。action 是 return（终止查找并返回当前状态)或 continue（继续这一行的其他项)中的一个。如 hosts：dns nis [NOTFOUND=return] files 将会首先在 DNS 中，然后在 NIS 中查找主机名。只有当前两项都不可用时才使用文件/etc/hosts。

19.4.7 配置 DNS 客户/etc/resolv.conf

文件/etc/resolv.conf 配置 DNS 客户，它包含了主机的域名搜索顺序和 DNS 服务器的地址，每一行应包含一个关键字和一个或多个由空格隔开的参数。下面是一个示例文件。

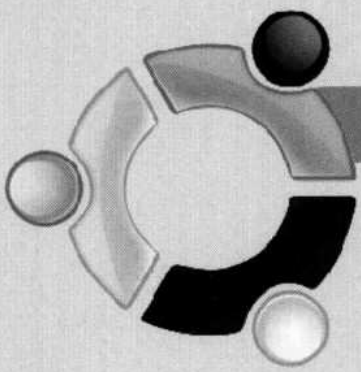
```
search mydom.edu.cn
nameserver 210.34.0.14
nameserver 210.34.0.2
```

合法的参数及其意义如下：

- nameserver：表明 DNS 服务器的 IP 地址。可以有很多行的 nameserver，每一个带一个 IP 地址。在查询时就按 nameserver 在本文件中的顺序进行，且只有当第一个 nameserver 没有反应时才查询下面的 nameserver。
- domain：声明主机的域名。很多程序用到它，如邮件系统，当为没有域名的主机进行 DNS 查询时，也要用到 domain。如果没有域名，主机名将被使用，删除所有在第一个点（.）前面的内容。
- search：它的多个参数指明域名查询顺序。当要查询没有域名的主机时，主机将在由 search 声明的域中分别查找。domain 和 search 不能共存。如果同时存在，将使用后面出现的。
- sortlist：允许将得到的域名结果进行特定的排序。它的参数为网络/掩码对，允许任意的排列顺序。Red Hat 中没有提供默认的/etc/resolv.conf 文件，它的内容是根据在安装时给出的选项动态创建的。

19.5 课后练习

1. 请列举常用的网络信息查询命令。
2. 如何打开 Ubuntu 中的可视化网络工具（gome nettool）？
3. 通过网络工具（gome nettool）可以实现哪些功能？
4. 如何设置 Ubuntu 的 IP、DNS 等网络属性？是否有可视化工具供用户设置？如有，如何打开？
5. 请尝试在 Shell 命令下进行 Ubuntu 的网络设置。



Part

04

Ubuntu 服务管理

本篇重点介绍系统服务及Linux常用服务，包括认识系统服务、WWW服务—Apache2、FTP服务—VSFTPD、邮件服务—Postfix、文件共享服务—SAMBA和DNS服务—BIND。通过这些服务的介绍，读者基本可以了解Ubuntu的服务。

Chapter20 系统服务管理

在类 Unix 系统中，大家常听到 daemon 这个英文单词，那么什么是 daemon 呢？这些 daemon 放在什么地方？起到什么样的作用？该如何启动这些 daemon？又如何有效管理这些 daemon？此外，要如何管理这些 daemon 占用的端口？这些都是作为系统管理员需要掌握的基础知识，而对于需要提供网络服务的服务器来说，系统服务的知识就显得更重要了，下面就来对系统服务基础进行介绍。

20.1 系统服务基础

daemon 就是一个在后台当中执行的程序，更具体地来说，daemon 通常是负责系统上面的某个服务（service），好让系统可以接受来自用户或者网络客户端的请求，并做出响应。当开机的时候，系统常常会在后台中自动运行一些程序，另外，服务管理员在开机完成后登录系统，也会触发运行的后台程序。

那什么又是服务（service）？Service 就是主机所能提供的功能。这些功能包括系统上面的以及针对网络的服务。系统上面的服务，如程序调度中的 cron 与 at 等，主要负责主机上面的任务调度；网络服务，则包括远程连接 SSH 服务，Web 服务等，它们让客户端连接上来取得数据。其实很多时候，并不需要去区分哪个是 daemon，哪个是 service，可以将 service 和 daemon 视为相同的东西，都是在后台中执行的程序。根据 daemon 的启动与管理方式，可以将 daemon 分为可独立启动的 stand alone 与通过统一安全机制管理的 super daemon 两大类。

- stand_alone
  就字面上的意思来说，stand alone 就是独立启动的意思，也就是说，该 daemon 启动之后，就直接常驻在内存当中，虽然它会一直占用系统的资源，但其最大的优点就是，它会一直在后台运行，所以当有请求过来的时候，它就能很快地做出响应。网络服务常常用这一种 daemon，如 Web 服务，因为它需要比较快的响应速度。
- super daemon
  相对于 stand alone 的执行方式，这种服务的启动方式则是通过一个统一的 daemon 来负责唤醒服务，这个统一负责的 daemon 就是 xinet 服务。当有网络请求过来的时候，该请求会先把这个服务发送给 xinet，然后 xinet 根据该网络请求的数据包的内容（包含记录 IP 与端口信息）将数据包送给实际提供的服务，而该服务这个时候才会启动，最常见到的就是 ftp 网络服务。这种方式最大的优点就是：当没有数据包来的时候，该服务不会一直占据系统资源（该服务会在睡眠状态），但是相对应的，它的响应时间也会比较慢，因为 xinet 还要花费一些时间去唤醒该服务。

这两种启动的方式哪一个比较好呢？要具体情况具体分析，主要看服务的工作负荷与实际的用途。例如当主机是用来作为 Web 服务器时，那么 httpd 自然就以 stand alone 的方式启动较佳。

另外，如果以 daemon 的工作状态来区分，则主要分为 signal-control 和 interval-control 两类。

- signal-control
  这种 daemon 是通过信号来管理的，只要有任何请求进来，它就会立即启动去处理，例如打印机的服务（cupsd）。
- interval-control
  这种 daemon 则主要是每隔一段时间就主动地去执行某项任务，所以，即使设定好配置文件后，它也不会立刻执行，而是在某个时间点才会执行。比如说，at 与 cron。

除此之外，为了更好地区分普通程序和 daemon 程序，daemon 程序加载到 Linux 使用时，通常在服务的名称之后会加上一个 d，例如例行性命令的建立的 at 与 cron 这两个服务，通常会被称为 atd 与 crond，这个 d 代表的就是 daemon 的意思。所以，使用 ps 与 top 来查看程序时，会发现很多 xxxd 的程序，这些通常就是 daemon 程序。

20.2 Ubuntu 的系统服务查看

在了解了 Daemon 和服务的关系后，下面来介绍查看服务，包括服务启动与否、它的 PID 和占用的端口号。同时，还要系统地了解一下 Ubuntu 常用的一些服务。

20.2.1 查看系统启动的服务

查看系统已启动的服务方式很多，可以使用 netstat、ps 等命令。ps 命令用来查看整个系统上的服务，因为它可以将全部进程服务都查找出来。但是，我们比较关心的还是那些启动了网络监听的服务，所以，查看服务的时候经常用到的是 netstat 命令。下面通过几个示例来说明。

示例 1

下面是找出目前系统开启的网络服务的示例。

```
root@ubuntuer: ~# netstat -tulp
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address  Foreign Address     State   PID/Program name
tcp        0    0 *:mysql          *:*               LISTEN  3538/mysqld
tcp        0    0 *:sunrpc         *:*               LISTEN  3373/portmap
tcp        0    0 *:http           *:*               LISTEN  5431/lighttpd
tcp        0    0 *:607            *:*               LISTEN  3393/rpc.statd
tcp        0    0 *:8009           *:*               LISTEN  15836/java
tcp        0    0 *:2222           *:*               LISTEN  3494/sshd
tcp        0    0 localhost.localdomain:8015  *:*    LISTEN  15836/java
tcp        0    0 *:webcache       *:*               LISTEN  15836/java
udp        0    0 *:601            *:*                       3393/rpc.statd
udp        0    0 *:604            *:*                       3393/rpc.statd
udp        0    0 *:32865          *:*                       18530/perl
udp        0    0 *:sunrpc         *:*                       3373/portmap
```

```
# 看一看上面，Local Address 的地方会出现主机名称与服务名称
# 可以加上 -n 来显示端口号，而服务名称与 port 对应则是写在/etc/services 里面
```

示例 2

下面是找出所有的监听网络的服务（包含 socket 状态）的示例。

```
root@ubuntuer: ~# netstat -lnp
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address  Foreign Address     State   PID/Program name
tcp     0   0 0.0.0.0:3306        0.0.0.0:*           LISTEN  3538/mysqld
tcp     0   0 0.0.0.0:111         0.0.0.0:*           LISTEN  3373/portmap
tcp     0   0 0.0.0.0:80          0.0.0.0:*           LISTEN  5431/lighttpd
tcp     0   0 0.0.0.0:607         0.0.0.0:*           LISTEN  3393/rpc.statd
tcp     0   0 :::8009             :::*                LISTEN  15836/java
tcp     0   0 :::2222             :::*                LISTEN  3494/sshd
tcp     0   0 ::ffff:127.0.0.1:8015 :::*              LISTEN  15836/java
tcp     0   0 :::8080             :::*                LISTEN  15836/java
udp     0   0 0.0.0.0:601         0.0.0.0:*                   3393/rpc.statd
udp     0   0 0.0.0.0:604         0.0.0.0:*                   3393/rpc.statd
udp     0   0 0.0.0.0:111         0.0.0.0:*                   3373/portmap
Active Unix domain sockets (only servers)
Proto RefCnt  Flags     Type      State      I-Node   PID/Program name Path
Unix 2        [ ACC ]   STREAM    LISTENING  4947704 18495/1   /tmp/ssh-IJmla18495/agent.18495
Unix 2        [ ACC ]   STREAM    LISTENING  10470   4650/X        /tmp/.X11-Unix/X0
Unix 2        [ ACC ]   STREAM    LISTENING  8052    3644/xfs      /tmp/.font-Unix/fs7100
Unix 2        [ ACC ]   STREAM    LISTENING  10109   3834/gdm-binary    /tmp/.gdm_socket
Unix 2        [ ACC ]   STREAM    LISTENING  7739    3483/acpid         /var/run/acpid.socket
Unix 2        [ ACC ]   STREAM    LISTENING  7925    3596/htt_server
                                                              /var/run/iiim/.iiimp-Unix/9010
Unix 2        [ ACC ]   STREAM    LISTENING  8334    3720/dbus-daemon-1
                                                            /var/run/dbus/system_bus_socket
Unix 2        [ ACC ]   STREAM    LISTENING  7831    3538/mysqld    /var/lib/mysql/mysql.sock
Unix 2        [ ACC ]   STREAM    LISTENING  7856    3564/gpm       /dev/gpmctl
# 仔细地看一下，除了原有的网络监听端口之外，还有 socket 显示在上面
# 我们可以清楚地知道有哪些服务被启动
```

示例 3

下面是查看所有的网络连接状态，查询是否有异常的连接的示例。

```
root@ubuntuer: ~# netstat -anp
# 利用这个指令可以查出有问题的连接，还可取得 PID
# 可以使用 kill 命令来终止任何一个可能有问题的连接
```

利用 netstat 可以取得很多跟网络有关的服务信息，通过这个指令，可以轻易地了解网络的使用状态，并且可以通过 PID 与 kill 的相关功能，将有问题的数据剔除。

20.2.2 Ubuntu 服务简要说明

随着 Ubuntu 软件的支持度越来越高，可以在 Ubuntu 上用的 daemon 也越来越多了。所以，在这

里简单介绍一下当前 Ubuntu 默认的各个服务（见表 20.1）。

表 20.1 Ubuntu 下的系统服务简介

Daemon 名称	说明
acpi	高级电源管理支持，用于电源管理，非常重要
acpid	acpi 守护程序，用于电源管理，非常重要
alsa	声音子系统
alsa- utils	音频设置管理
anacron	cron 的子系统，将系统关闭期间的计划任务，在下一次系统运行时执行
apmd	acpi 的扩展
atd	类似于 cron 的任务调度系统，建议关闭
binfmtt-support	核心支持其他二进制的文件格式，建议开启
Bluez-utiles	蓝牙设备支持
bootlogd	启动日志，建议开启它
cron	任务调度系统，建议开启
cupsys	打印机子系统
dbus	消息总线系统
dns-clean	使用拨号连接时，清除 dns 信息
evms	企业卷管理系统（Enterprise Volumn Management System）
fetchmail	邮件用户代理，用于收取邮件
gdm	gnome 登录和桌面管理器
gpm	终端中的鼠标支持
halt	建议别动它
hdparm	调整硬盘的脚本，配置文件为/etc/hdparm.conf
hibernate	系统休眠
hotkey- setup	笔记本功能键支持。支持类型包括 HP、Acer、ASUS、Sony、Dell 和 IBM。
hotplug and hotplug-net	即插即用支持，比较复杂，建议不要动它
hplip	HP 打印机和图形子系统
ifrename	网络接口重命名脚本。如果有 10 块网卡，那就应该开启它
inetd	在文件/etc/inetd.conf 中，注释掉所有你不需要的服务。如果该文件不包含任何服务，那关闭它是很安全的
klogd	重要
linux-restricted-modules-common	受限模块支持。/lib/linux-restricted-modules/文件夹中的模块为受限模块，例如某些驱动程序。如果没有使用受限模块，就不需要开启它
lvm	逻辑卷管理系统支持
makedev	创建设备文件，非常重要
mdamd	磁盘阵列
module-init-tools	从/etc/modules 加载扩展模块，建议开启
networking	网络支持。按/etc/network/interfaces 文件默认激活网络，非常重要
ntpdate	时间同步服务，建议关闭
pcmcia	pcmcia 设备支持

（续表）

Daemon 名称	说明
powernowd	移动 CPU 节能支持
ppp and ppp	DNS 拨号连接
readahead	预加载库文件
reboot	建议别动它
resolvconf	自动配置 DNS
rmnologin	清除 nologin
rsync	rsync 守护程序
sendsigs	在重启和关机期间发送信号
single	激活单用户模式
ssh	ssh 守护程序，建议开启它
stop-bootlogd	在 2，3，4，5 运行级别中停止 bootlogd 服务
sudo	检查 sudo 状态，重要
sysklogd	系统日志
Udev & udev-mab	用户空间 dev 文件系统（Userspace Dev Filesystem），重要
umountfs	卸载文件系统
urandom	随机数生成器
usplash	开机画面支持
vbesave	显卡 BIOS 配置工具，保存显卡的状态
xorg-common	设置 X 服务 ICE
adjtimex	调整核心时钟的工具
dirmngr	证书列表管理工具，和 gnupg 一起工作
hwtools	irqs 优化工具
libpam-devperm	系统崩溃之后，用于修理设备文件许可的守护程序
lm-sensors	板载传感器支持
mdadm-raid	磁盘陈列管理器
screen-cleanup	清除开机屏幕的脚本
xinetd	管理其他守护进程的一个 inetd 超级守护程序的重要配置文件

20.3 系统服务设置

接下来学习一下系统服务的设置，包括如何设定服务的启动，如何将服务设置为随机启动等。

20.3.1 启动/停止/重启服务

我们知道，所谓的 Daemon 服务就是一个可以在系统后台中运行的程序，所以，要启动该服务，就是找到它的可执行文件。不过，因为大多数的 Daemon 可执行文件中需要加入的参数很多，为了

方便系统的配置和管理，所以现在各主要的 Linux 发行版都会针对该服务设计一个比较方便的 Shell 脚本来启动程序，这些 Shell 脚本都集中放置在/etc/init.d/和/etc/xinetd.d/目录下。因此，启动服务的方法就变得很简单了。只要设定好该服务的启动文件，然后在命令行下输入以下代码：

```
# 启动服务的方式（以 cron 为例）
root@ubuntuer: ~# /etc/init.d/cron start

# 停止服务的方式（以 cron 为例）
root@ubuntuer: ~# /etc/init.d/cron stop

# 重启服务的方式（以 cron 为例）
root@ubuntuer: ~# /etc/init.d/cron restart
```

除了命令行外，Ubuntu 还提供一个简单的服务设置图形工具，通过执行【系统】|【服务管理】|【服务】命令启动程序，然后可以通过点选左侧的复选框来启动/关闭某个服务，如图 20.1 所示。

图 20.1 服务设置

20.3.2 开机自启动服务

使用 netstat 只能查看目前已经存在于系统当中的服务，使用/etc/init.d/* start 的方法，仅能在目前的环境下启动某个服务而已。那么重新启动机器后呢？该服务是否还能自动启动？这个时候要复习一下 Ubuntu Linux 系统到底是怎么开机的。

Ubuntu Linux 系统主要通过以下步骤启动：

Step 01 BIOS。
Step 02 MBR (boot loader)。
Step 03 kernel loading。
Step 04 init program。

Step 05 initial script (/etc/rc.d/rc.sysinit)。

Step 06 daemon start (/etc/rc.d/rc[0–6].d/*)。

Step 07 local setting (/etc/rc.d/rc.local)。

大致的情况是这样，整个服务可以设置开机启动的地方有两个。一个是在 daemon start (/etc/rc.d/rc[0–6].d/*) 目录下，该目录下的文件主要以 S 及 K 开头，分别代表开机时启动与关机时关闭的意思，也就是说，如果把需要开机启动的服务写入/etc/rc.d/rc[0–6].d 目录下，那么该服务就可以在开机的时候自动启动。

而另外一个支持开机启动的文件就是/etc/rc.d/rc.local，可以将任何想要在开机时启动的程序写入这个文件，这个文件是以 Shell 脚本的语法写成的，所以可以容易地设定所需要的启动参数。

既然如此，那么怎么设置 /etc/rc.d/rc[0–6].d 当中的参数呢？读者或许听说过 chkconfig 命令，但在 Ubuntu 中经常用到的还是 sysc–rc–conf。这里主要讲述 sysv–rc–conf 和 chkconfig 这两个命令的使用，sysc–rc–conf 是一个基于 ncurses 的工具，它可以控制系统内所有的服务。

sys-rc-conf 命令

在使用 sysc–rc–conf 前，需要安装一下该软件，在命令行下输入：

```
root@ubuntuer: ~# apt-get install sysv-rc-conf
```

安装完毕后，直接在命令行下输入 sysc–rc–conf，这个时候就能看见一个类似图形界面的窗口，如图 20.2 所示。

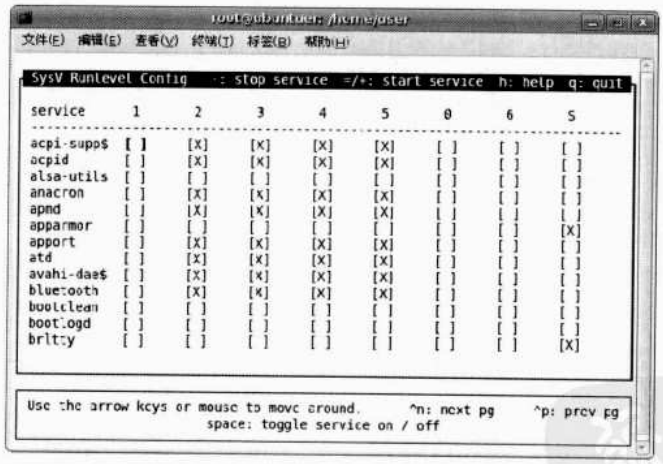


图 20.2 sysc–rc–conf 的 GUI

操作界面十分简洁，用户可以用鼠标单击，也可以用键盘方向键定位，用空格键选择运行级别，按 Ctrl+N 键翻下一页，按 Ctrl+P 键翻上一页，按 Q 键退出。

chkconfig 命令

语法

```
root@ubuntuer: ~# chkconfig --list
root@ubuntuer: ~# chkconfig [--add|--del] [service_name]
```

```
root@ubuntuer: ~# chkconfig --level [0123456] [service_name] [on|off]
```

⚙参数说明

- list：将目前的各项服务状态列出来。
- add：增加一个服务名称给 chkconfig 来管理，该 service_name 必须在/etc/init.d/ 目录内。
- del：删除一个 chkconfig 管理的服务。
- level：设定某个服务在该 level 下启动（on）或关闭（off）。

⚙示例 1　查看目前系统上所有被 chkconfig 管理的服务

```
root@ubuntuer: ~# chkconfig --list |more
NetworkManager  0:off   1:off  2:off   3:off   4:off   5:off   6:off
NetworkManagerDispatcher      0:off   1:off   2:off  3:off  4:off  5:off
6:off
acpid           0:off.  1:off   2:off   3:on    4:on    5:on    6:off
anacron         0:off   1:off   2:on    3:on    4:on    5:on    6:off
apmd            0:off   1:off   2:on    3:on    4:on    5:on    6:off
atd             0:off   1:off   2:off   3:on    4:on    5:on    6:off
autofs          0:off   1:off   2:off   3:on    4:on    5:on    6:off
avahi-daemon    0:off   1:off   2:off   3:on    4:on    5:on    6:off
avahi-dnsconfd  0:off   1:off   2:off   3:off   4:off   5:off   6:off
bluetooth       0:off   1:off   2:on    3:on    4:on    5:on    6:off
cpuspeed        0:off   1:on    2:on    3:on    4:on    5:on    6:off
crond           0:off   1:off   2:on    3:on    4:on    5:on    6:off
dc_client       0:off   1:off   2:off   3:off   4:off   5:off   6:off
dc_server       0:off   1:off   2:off   3:off   4:off   5:off   6:off
dhcdbd          0:off   1:off   2:off   3:off   4:off   5:off   6:off
diskdump        0:off   1:off   2:off   3:off   4:off   5:off   6:off
firstboot       0:off   1:off   2:off   3:on    4:off   5:on    6:off
gpm             0:off   1:off   2:on    3:on    4:on    5:on    6:off
haldaemon      0:off    1:off   2:off   3:on    4:on    5:on    6:off
hidd            0:off   1:off   2:on    3:on    4:on    5:on    6:off
httpd           0:off   1:off   2:off   3:on    4:off   5:off   6:off
iptables        0:off   1:off   2:on    3:on    4:on    5:on    6:off
irda            0:off   1:off   2:off   3:off   4:off   5:off   6:off
irqbalance      0:off   1:off   2:on    3:on    4:on    5:on    6:off
isdn            0:off   1:off   2:on    3:on    4:on    5:on    6:off
kudzu           0:off   1:off   2:off   3:on    4:on    5:on    6:off
mdmonitor       0:off   1:off   2:on    3:on    4:on    5:on    6:off
mdmpd           0:off   1:off   2:off   3:off   4:off   5:off   6:off
messagebus      0:off   1:off   2:off   3:on    4:on    5:on    6:off
mysqld          0:off   1:off   2:on    3:on    4:on    5:on    6:off
named           0:off   1:off   2:off   3:off   4:off   5:off   6:off
netdump         0:off   1:off   2:off   3:off   4:off   5:off   6:off
netfs           0:off   1:off   2:off   3:on    4:on    5:on    6:off
netplugd        0:off   1:off   2:off   3:off   4:off   5:off   6:off
network         0:off   1:off   2:on    3:on    4:on    5:on    6:off
nfs             0:off   1:off   2:off   3:off   4:off   5:off   6:off
```

```
nfslock         0:off   1:off   2:off   3:on    4:on    5:on    6:off
--More--
```

⚙示例 2　显示出目前在 run level 3 未启动的服务。

```
root@ubuntuer: ~# chkconfig --list | grep '3:on'
acpid           0:off   1:off   2:off   3:on    4:on    5:on    6:off
anacron         0:off   1:off   2:on    3:on    4:on    5:on    6:off
apmd            0:off   1:off   2:on    3:on    4:on    5:on    6:off
atd             0:off   1:off   2:off   3:on    4:on    5:on    6:off
autofs          0:off   1:off   2:off   3:on    4:on    5:on    6:off
avahi-daemon    0:off   1:off   2:off   3:on    4:on    5:on    6:off
bluetooth       0:off   1:off   2:on    3:on    4:on    5:on    6:off
cpuspeed        0:off   1:on    2:on    3:on    4:on    5:on    6:off
crond           0:off   1:off   2:on    3:on    4:on    5:on    6:off
firstboot       0:off   1:off   2:off   3:on    4:off   5:on    6:off
gpm             0:off   1:off   2:on    3:on    4:on    5:on    6:off
haldaemon       0:off   1:off   2:off   3:on    4:on    5:on    6:off
hidd            0:off   1:off   2:on    3:on    4:on    5:on    6:off
httpd           0:off   1:off   2:off   3:on    4:off   5:off   6:off
iptables        0:off   1:off   2:on    3:on    4:on    5:on    6:off
irqbalance      0:off   1:off   2:on    3:on    4:on    5:on    6:off
isdn            0:off   1:off   2:on    3:on    4:on    5:on    6:off
kudzu           0:off   1:off   2:off   3:on    4:on    5:on    6:off
mdmonitor       0:off   1:off   2:on    3:on    4:on    5:on    6:off
messagebus      0:off   1:off   2:off   3:on    4:on    5:on    6:off
mysqld          0:off   1:off   2:on    3:on    4:on    5:on    6:off
netfs           0:off   1:off   2:off   3:on    4:on    5:on    6:off
network         0:off   1:off   2:on    3:on    4:on    5:on    6:off
nfslock         0:off   1:off   2:off   3:on    4:on    5:on    6:off
portmap         0:off   1:off   2:off   3:on    4:on    5:on    6:off
readahead_early 0:off   1:off   2:on    3:on    4:on    5:on    6:off
rpcgssd         0:off   1:off   2:off   3:on    4:on    5:on    6:off
rpcidmapd       0:off   1:off   2:off   3:on    4:on    5:on    6:off
sendmail        0:off   1:off   2:on    3:on    4:on    5:on    6:off
smartd          0:off   1:off   2:on    3:on    4:on    5:on    6:off
sshd            0:off   1:off   2:on    3:on    4:on    5:on    6:off
syslog          0:off   1:off   2:on    3:on    4:on    5:on    6:off
xfs             0:off   1:off   2:on    3:on    4:on    5:on    6:off
```

⚙示例 3　让 atd 这个服务在 run level 为 3，4，5 时启动

```
root@ubuntuer: ~# chkconfig --level 345 atd on
```

chkconfig 是管理服务的好工具，功能简单且易用，只是直接在/etc/rc.d/rc[0–6].d 里面针对某服务进行连接文件的设置而已。例如示例 3，仅在/etc/rc.d/rc3.d/，/etc/rc.d/rc4.d/及/etc/rc.d/rc5.d/下建立一个链接文件，该链接文件链接到/etc/init.d/atd 而已。

那么如何将自己建立的服务加入 chkconfig 的管理当中呢？首先要把该服务加入/etc/init.d/目录当中，比如说，在/etc/init.d/下建立一个 testd 文件，该文件仅是一个简单的示例程序。

- testd 将在 run level 3 及 5 启动。

- testd 在/etc/rc.d/rc[35].d 当中启动时，以 S80 开始，以 K70 结束。

那么可以这样做：

```
root@ubuntuer: ~# vi /etc/init.d/testd
#!/bin/bash
# description: 开机启动服务的例子
echo "Sample"

root@ubuntuer: ~# chkconfig --add testd
root@ubuntuer: ~# chkconfig--level 35 testd on
root@ubuntuer: ~# chkconfig --list testd
testd          0:off   1:off   2:off   3:on    4:off   5:on    6:off
#加入了 chkconfig 的管理当中了！再去看看 /etc/rc.d/底下的文件

[root@linux ~]# find /etc/rc.d/ -type l | grep 'testd'|sort
/etc/rc.d/rc0.d/K70testd
/etc/rc.d/rc1.d/K70testd
/etc/rc.d/rc2.d/K70testd
/etc/rc.d/rc3.d/S80testd
/etc/rc.d/rc4.d/K70testd
/etc/rc.d/rc5.d/S80testd
/etc/rc.d/rc6.d/K70testd
# 如果要将这些数据都删除的话，那么就输入下面的命令
root@ubuntuer: ~# chkconfig --del testd
root@ubuntuer: ~# rm /etc/init.d/testd
```

20.4 课后练习

1. 什么是 Daemon 程序，如何区分 Daemon 程序和普通程序？
2. 如何查看系统启动的服务程序？包括查看系统的 PID、端口占用情况。
3. 如何启动服务，停止服务？如何设置开机后自动启动服务？
4. 请列举你知道的 Ubuntu 常用的系统服务。

Chapter21 WWW 服务器——Apache

Apache是一种功能强大的Web服务器。如今，Internet上运行在Linux上的无数Apache服务器正为Web世界的日益繁荣提供着有力的支撑。尽管Ubuntu是一种新兴的Linux分支，但Ubuntu组织却为Apache提供了丰富的支持软件，这些软件都可以从发行版的光盘获取，也可以从官方站点轻松下载。本文将向读者介绍WWW的概念以及Apache服务器的由来，如何在Ubuntu Linux系统中安装Apache2，如何配置Apache2以及迅速搭建Apache Web服务器。

21.1 WWW 与 Apache

工欲善其事，必先利其器。在进行Apache服务的安装和网站搭建配置之前，先来了解一下WWW的常识以及Apache服务器的发展历程，学习这些有利于学习WWW服务。

21.1.1 什么是 WWW

WWW（World Wide Web，环球信息网），也可以简称为Web，有人译作“万维网”、“环球网”、“Web网”、“3W网”。最初是由欧洲核物理研究中心（cern）提出来的。其创建者伯纳斯•李，在他1991年8月6日创建的第一个网址中解释了万维网的工作原理等内容。他也因此被《时代》杂志评价为二十世纪最重要的100位人物之一。

WWW，是一张附着在Internet上的覆盖全球的信息“蜘蛛网”，镶嵌着无数以超文本形式存在的信息，其中有璀璨的明珠，当然也有腐臭的垃圾。WWW是当前Internet上最受欢迎、最为流行、最新的信息检索服务系统。它把Internet上现有资源统统连接起来，使用户能在Internet上已经建立了WWW服务器的所有站点提供超文本媒体资源文档。这是因为，WWW能把各种类型的信息（静止图像、文本声音和音像）天衣无缝地集成起来。WWW不仅提供了图形界面的快速信息查找功能，还可以通过同样的图形界面（GUI）与Internet的其他服务器对接。

由于WWW为全世界的人们提供查找和共享信息的手段，因此，也可以把它看作是世界上各种组织机构、科研机关、大学、公司厂商热衷于研究开发的信息集合。它基于Internet的查询、信息分布和管理系统，是人们进行交互的多媒体通信动态格式。它的正式提法是：“一种广域超媒体信息检索原始规约，目的是访问巨量的文档。”WWW已经实现的部分是，给计算机网络上的用户提供一种兼容的手段，以简单的方式去访问各种媒体。它是第一个真正的全球性超媒体网络，改变了人们观察和创建信息的方法。因而，整个世界迅速掀起了研究开发使用WWW的巨大热潮。

WWW诞生于Internet之中，后来成为Internet的一部分，而今天，WWW几乎成了Internet的代名词。通过它，加入其中的每个人能够在瞬间抵达世界的各个角落，只要将一根网线插入你的PC（它可能是随身携带的笔记本电脑加上一部移动电话），此时全球的信息就在你的指尖上！

WWW并不是实际存在于世界的哪一个地方，事实上，WWW的使用者每天都赋予它新的含义。

Internet 社会的公民们（包括机构和个人），把他们需要公之于众的各类信息以主页（Homepage）的形式嵌入 WWW。主页中除了文本外还包括图形、声音和其他媒体形式；而内容则从各类招聘广告到电子版圣经，可以说包罗万象，无所不有。主页是在 Web 上出版的一些 HTML（Hyper Text Markup Language，超文本标识语言）文本。

随着手机上网的飞速发展，最近有的专家把 WAP 和 WWW 并称。WAP 目前已成为通过移动电话或其他无线终端访问无线信息服务的全球事实标准。它的发展与应用是无可限量的。

上过网的人都知道，在浏览站点前需要输入网址，以便浏览器能够向这个网址发出请求（不论是使用 Mozilla、IE 或是其他工具），而这个网址的格式如下：

```
<协议>://<主机地址>[:端口]/<目录资源>
```

- 协议：包括 http、https、ftp、ssh、telnet 这几种常见的方式。其中，http 和 https 是利用主机的 http 端口，通常为 80。至于 ftp 这个协议则是利用主机的 ftp 端口，通常为 21，需要注意的是，我们使用的 80 与 21 都是主机所提供的服务端口，而不是客户端的端口，所以，使用 http 与 ftp 连上同一部计算机，所取得的信息并不见得会一样，因为服务本身就不同。此外，如果没有指定协议的话，那么使用的协议就需要看客户端（Client）设置的默认协议了。举个例子来说，如果使用 Firefox 的话，默认的协议就是 http，因此，在网址列输入 google.com 时，Firefox 立刻就会以 http 协议连接出去。
- 主机地址：由于计算机仅识别 IP，所以，如果输入 IP 的话，可以立即进行连接主机了。但是，如果是输入域名（domain name）的话，那么就必须要让该域名可以经由解析器得到对应的 IP。解析器是什么？就是 /etc/hosts 或者/etc/resolv.conf 中的设定。当然，对外提供正常的 WWW 服务时，主机名就必须让大家可以解析到 IP，就需要去申请一个合法的域名。
- 目录资源（Uniform Resource Indicator，URI）：如果要访问的网站网页在主网页的目录下，那么可以直接输入目录与网页的名称，就可以直接取得那个页面的数据。此外，如果只输入网址，并没有输入网页名称呢？那么在 Server 端将会自动地判断（依据 Server 设定）该目录下是否有设置中引用的网页名称。
- 端口：一般来说，各个协议都有其独特的端口，例如众所皆知的 http 协议使用的是 80，而 ftp 使用的是 21 端口，这些都是默认的端口。所以，当要连接到某个网站时，输入 http://that.host.name 就会主动使用 80 端口来尝试连接到对方主机。但是如果不想要使用该端口呢？举个例子来说，假如网站使用的是 8080 这个端口来进行 WWW 的服务，那么除非进行防火墙内的端口对应，否则直接在网址列输入 http://your.host.name 结果将无法连接到 WWW 服务器，因为它会直接连接到 80 端口。所以，我们就要告诉浏览器，要向 Server 请求服务的是哪一个端口，因此，就要写成：http://your.host.name:8080 才可以连接到服务器的 8080 端口。

21.1.2 Apache 简介

Apache 起初由 Illinois 大学 Urbana-Champaign 的国家高级计算程序中心开发。此后，Apache 被开放源代码团体的成员不断地发展和加强。Apache 服务器拥有牢靠、可信的美誉，差不多超过半

数的因特网站都使用 Apache 提供服务，特别是几乎所有最热门和访问量最大的网站。

Apache 支持许多特性，从服务器端的编程语言支持到身份认证方案。一些通用的语言接口支持 Perl、Python、Tcl 和 PHP，流行的认证模块包括 mod_access、mod_auth 和 mod_digest，还有 SSL 和 TLS 支持（mod_ssl）、proxy 模块、很有用的 URL 重写（由 mod_rewrite 实现），定制日志文件（mod_log_config），以及过滤支持(mod_include 和 mod_ext_filter)。Apache 日志可以通过网页浏览器使用免费的脚本 AWStats 或 Visitors 来进行分析。

Apache 的 2.x 版本核心在 Apache 1.x 版本之上进行了重要的加强，包括线程、更好地支持非 Unix 平台（例如 Windows）、新的 Apache API，以及 IPv6 支持。不过，由于现行已经使用了 Apache1.x 的大多数网站，出于网站运营稳定性的考虑，并没有从 Apache1.x 升级到 Apache2.x，因而 Apache1.x 的使用仍然十分广泛。

21.2 搭建 Apache2 服务器

在了解了 WWW 与 Apache 的关系后，接下来将着重讲述 Apache 2 在 Ubuntu Linux 下的安装方法，并对 Apache 2 安装后的目录结构做一个概要介绍，让读者对 Apache 2 有个整体的认识。

21.2.1 Apache2 的安装

在 Ubuntu Linux 下安装 Apache2 服务器主要有两种方式，一种就是使用新立得软件包管理器安装，另外一种就是在命令行下输入命令 apt-get 来安装。选用哪种方式来安装，主要依据系统安装的类型，如果安装的服务器版本没有 GUI 功能，那只能通过第二种方式通过命令行来安装，当然如果安装的是桌面版本的系统，那么这两种方式都能为你所用了。

使用新立得软件包管理器安装 Apache2

Step 01 通过执行【系统】|【系统管理】|【新立得软件包管理器】命令打开程序。

Step 02 在新立得软件包管理器窗口中，单击工具栏中的Search按钮，在弹出的小窗口中输入关键字"apache2"进行查找，如图21.1所示。

图 21.1 查找 Apache2 软件包

Step 03 在查询得出的结果中，选中apache2和apache2.2-common两软件包进行安装，如图21.2所示。

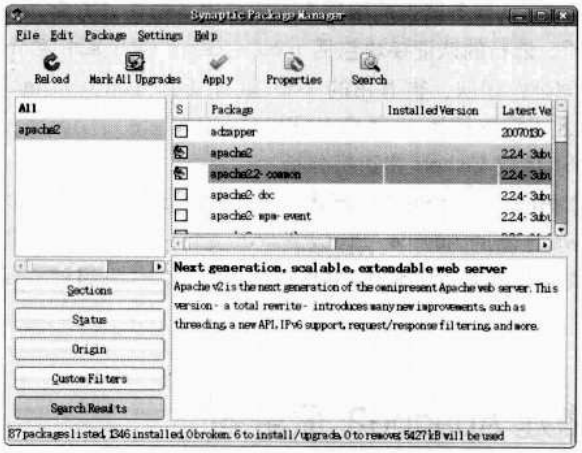


图 21.2 选择 Apache2 软件包进行安装

Step 04 选择完毕后，单击Apply按钮开始安装。完成安装后，就可以退出新立得软件管理器。

在命令行下安装 Apache2

Step 01 通过执行【应用程序】|【附件】|【终端】命令打开命令行。

Step 02 在命令提示行下输入sudo apt-get install apache2，按回车键，然后系统就会自动安装apache2，同时也会安装相应的依赖软件包，具体安装过程如下所示：

```
user@ubuntuer: ~$ sudo apt-get install apache2
[sudo] password for user:
正在读取软件包列表...完成
正在分析软件包的依赖关系树
Reading state information...完成
The following packages were automatically installed and are no longer
required:
  libarts1c2a libartsc0 xserver-xorg-video-amd kdelibs-data
  linux-headers-2.6.24-17-generic liblualib50 libavahi-qt3-1 libqt3-mt
  liblua50 libaudio2 linux-headers-2.6.24-17
Use 'apt-get autoremove' to remove them.
将会安装下列额外的软件包：
  apache2-mpm-worker apache2-utils apache2.2-common libapr1 libaprutil1 libpq5
建议安装的软件包：
  apache2-doc
下列【新】软件包将被安装：
  apache2 apache2-mpm-worker apache2-utils apache2.2-common libapr1
  libaprutil1 libpq5
共升级了 0 个软件包，新安装了 7 个软件包，要卸载 0 个软件包，有 29
个软件未被升级。
需要下载 1628KB 的软件包。
After this operation, 5763kB of additional disk space will be used.
您希望继续执行吗？[Y/n]Y
```

```
获取: 1 http://cn.archive.ubuntu.com hardy/main libapr1 1.2.11-1 [115KB]
.........
正在设置 apache2-mpm-worker (2.2.8-1ubuntu0.3) ...
 * Starting web server apache2
 * apache2: apr_sockaddr_info_get() failed for ubuntuer
apache2: Could not reliably determine the server's fully qualified domain
name, using 127.0.0.1 for ServerName                              [OK]
正在设置 apache2 (2.2.8-1ubuntu0.3) ...
Processing triggers for libc6 ...
ldconfig deferred processing now taking place
```

Apache 在安装期间将会新建一个目录：/var/www，该目录是 Apache 服务中存放文件目录的根目录。只要在浏览器的地址栏输入 localhost/或机器的 IP 地址就能访问放置在此目录中的所有文件。

21.2.2 Apache2 的目录结构

成功安装 Apache2 后，下面讲述一下 Apache2 的目录结构，让读者能知道 Apache 各个目录的功用，以便找到相应的配置文件。

- /etc/apache2：该目录下是 Apache2 服务器的配置文件，最主要的一个配置文件就是 apache2.conf。
- /etc/apache2/conf.d：该目录下是 Apache2 服务器的本地配置目录，包括相关的第三方或本地安装的包。
- /etc/apache2/envvars：配置 apache2ctl 脚本环境变量的文件，apache2ctl 脚本是用来管理维护 Apache2 服务器的一个脚本。
- /etc/apache2/mods-available：当前 Apache2 服务器下可用的模块以及这些模块的配置文件。
- /etc/apache2/mods-enabled：目录包含软连接文件，这些软连接指向可用的 Apache2 模块以及这些模块的配置文件，也就指向上面所提到的/etc/apache2/mods-available 目录下的内容，有点像/etc/init.d/目录下设置不同启动级别的概念。
- /etc/apache2/sites-available：定义了 Apache2 服务器支持的网站。
- /etc/default/apache2：配置文件用于定义是否在系统启动时自动启动 Apache2 服务器。
- /etc/init.d/apache2：Shell 脚本，使用 apache2ctl 工具来启动或停止 Apache2 服务器。
- /etc/mime.types：配置默认 MIME (Multipurpose Internet Mail Extensions) 的文件。
- /usr/lib/cgi-bin：CGI-BIN (Common Gateway Interface scripts)安装目录。
- /usr/sbin/apache2：可执行的 Apache2 服务器文件。
- /usr/sbin/apache2ctl：用于启动、停止、重启和监控 Apache2 服务器的 Shell 脚本。
- /usr/share/apache2-doc：目录下是 Apache2 服务器的使用手册，注意：只有在安装的时候要求安装 apache2-doc，这个目录才会存在。
- /usr/share/apache2/icons：目录下是用于 Apache2 服务器的默认 Icon。
- /var/log/apache2/access.log：默认 Apache2 服务器的访问 log 文件。

- /var/log/apache2/error.log：默认 Apache2 服务器的 error log 文件。
- /var/run/apache2/apache2.pid：在启动的时候，用于记录 Apache2 服务器的进程 ID，同时为/etc/init.d/apache2 脚本终止或重启 Apache2 服务器提供进程 ID 号。
- /var/www/apache2-default：目录包含 Apache2 服务器的默认 Web 首页内容。

21.2.3 启动和关闭 Apache2

在安装完 Apache2 服务器后，Apache2 服务器已经自动启动了。如果需要重新部署 Web 应用，或者说更改 Apache2 的系统设置后，那就一定要重新启动 Apache2 服务器，以便修改能够及时生效。

在命令行下管理 Apache2 服务器

启动 Apache2 服务器

```
root@ubuntuer: ~# /etc/init.d/apache2 start
* Starting web server apache2...          [OK]
```

停止 Apache2 服务器

```
root@ubuntuer: ~# /etc/init.d/apache2 stop
* Stopping web server apache2...          [OK]
```

重启 Apache2 服务器

```
root@ubuntuer: ~# /etc/init.d/apache2 restart
* Restarting web server apache2...          [OK]
```

使用图形界面管理 Apache2 服务器

在 Ubuntu 系统中，同样可以使用服务设置工具来管理 Apache2 服务器的状态，执行【系统】|【系统服务】|【服务】命令，打开【服务设置】对话框，点选【Web 服务器（apache2）】来启动或关闭 Apache2 服务器。如图 21.3 所示。

图 21.3 使用服务设置工具管理 Apache2 服务器

21.3 设置 Apache2 服务器

Apache2 服务器的管理和配置主要是通过修改配置文件的方式进行的，与 Apache1.x 配置相比，Apache2 的配置做了非常多的优化和改进，默认设置值也更加得合理，因而对于现在的系统管理员来说，无需过多地修改配置项，就能搭建一个性能可靠、安全稳定的 Web 服务器了。

21.3.1 Apache2 配置文件

Apache2 服务器的核心配置文件是/etc/apache2/apache2.conf，以及其他一些附属配置文件 httpd.conf、ports.conf、envvars 等。但从整体来说，Apache2 的配置文件里主要分为以下三个部分：

- Global Enviroment：全局环境参数设置。
- Main server：主服务器设置。
- Virtual Hosts：虚拟服务器设置。

21.3.2 全局环境参数设置

全局环境变量用于控制整个 Apache2 服务器的行为，可以说是 Apache2 服务器中的要害部位，正确合理地配置这些参数，将使得服务器能够安全流畅地运行。下面来讲述一下这些环境参数的意义（见表 21.1）。

表 21.1 Apache2 全局环境参数描述

参数名称	描述	默认值
ServerRoot	Apache2 服务器的根目录，用于存放配置、出错记录以及日志文件	/etc/apache2
LockFile	用于指定串行锁文件的存放路径，通常，该文件保存在本地磁盘上	/var/lock/apache2/accept.lock
PidFile	存放 Apache2 的父进程 PID 的文件名及路径	/var/run/apache2.pid
Timeout	设置超时时间。如果远程客户端超过限定的时间还没连接上 Apache2 服务器，或者 Apache2 服务器超过限定的时间没有响应给客户端，那么就自动断开连接	300
KeepAlive	是否允许建立长连接	on
MaxKeepAliveRequests	设置每次长连接所能发送的最大请求数，设为 0 表示允许无限制接入	100
KeepAliveTimeout	当 KeepAlive 设置为 On 时，该配置项用于设置相邻两个请求的超时时间，超过该时间段就断开连接	15
StartServers	启动服务器时初始化的进程数	5
MinSpareServers	空闲进程的最小数目	5

（续表）

参数名称	描述	默认值
MaxSpareServers	空闲进程的最大数目	10
MaxClient	允许启动的最大进程数目	150
MaxRequestPerChild	客户端连接服务器后所能发出的最大请求数目，设为 0 表示不限制	0
Listen	Apache2 服务器监听的端口或 IP 地址	80
LoadModule	设置以动态共享支持 DSO（Dynamic Shared Object）模式编译模块函数	N/A
ExtendedStatus	当管理程序允许时，检查 Apache2 的运行状态信息	Off

21.3.3 主服务器设置

这部分的参数设定主要涉及主服务器的行为特征，也是所有虚拟主机的默认参数设置。在全局环境参数设置中，很多参数设置都会直接或间接地影响到主服务器参数的设置。下面将对部分常用的主服务器参数进行简单的描述（见表 21.2）。

表 21.2 Apache2 主服务器配置参数描述

参数名称	描述	默认值
Port	如果全局环境参数 Listen 没有设定端口号，那么该参数设定服务器监听的端口号	80
User	允许运行 Apache2 的用户名	www-data
Group	允许允许 Apache2 的用户组	www-data
ServerAdmin	管理员的电子邮件地址，当 Apache2 服务器出现异常情况时，在错误信息中显示给客户端管理员的电子邮件	webmaster@localhost
ServerName	服务器的主机名，如果没有设置，Apache2 将尝试使用 IP 地址作为服务器名	localhost
UseCanonicalName	设置服务器构建自引用 URL 的方式，如果设置为 on，则使用 ServerName 和 port，否则使用客户端提供的主机名和端口	On
DocumentRoot	Apache2 放置网页目录路径	/var/www
AccessFileName	每个目录中用于控制访问信息的文件名	.htaccess
<Files ~ "^\.ht"> Order allow，deny Deny from all <Files>	不允许其他人访问以.ht 开头的文件，因为.htaccess 记录了相关访问权限的信息，而.htpasswd 记录了密码信息	无
DefaultType	如果 Apache2 不能识别文件的类型，则按照该设置值处理，一般情况下是以文本文件显示	text/plain

（续表）

参数名称	描述	默认值
HostnameLookups	设置 Apache2 是否连接到日志主机进行 DNS 搜索，因为 DNS 解析需要花费一定时间，所以默认值为 off，仅记录 IP 地址	off
ErrorLog	设置错误日志文件的位置	/var/log/apache2/error.log
LogLevel	设置日志文件的输出信息等级，从详细到简略，可分为 8 个级别：debug、info、notice、warn、error、crit、alert 和 emerg	warn
LogFormat	设置日志的输出格式，包括 combined、commen、referer 和 agent	无
ServerTokens	设置在 HTTP 响应信息的 header 里面加上 Apache2 服务器相关信息，设置值包括 Full、OS、Minor、Minimal、Major 和 Prod	Full
ServerSignature	设置为 On，当 Apache2 发生异常时，就在网页上显示服务器的版本信息、主机名、端口等；如果设置为 Off，不显示相关信息；设置为 Email 时，显示管理员的邮件地址给客户端	On
Alias	别名设定，可以将相对于 DocumentRoot 指定的目录链接到 ServerRoot 以为的目录	无
ScriptAlias	跟 Alias 有点相似，用来设置服务器脚本目录	无
IndexOptions	从浏览器中显示目录的文件列表，如果设置为 FancyIndexing，表示会用 AddIconByType 参数中所设置的图片来显示文件类型，否则只显示文件名称	FancyIndexing
AddIconByType	根据文件的 MIME 类型显示图标	无
DefaultIcon	无法识别文件的 MIME 类型时默认显示的图标	/icons/unknown.gif

21.3.4 虚拟服务器设置

Apache2 配置文件中的第三部分用于设置虚拟服务器。什么是虚拟服务器呢？就是让一台物理服务器提供多个 Web 站点的访问，站点间互不干扰，但是从站点网址上看，则好像有多台服务器在后面支撑运行。

Apache2 虚拟服务器有以下两种：

- 基于主机名的虚拟服务器：根据客户端提交的 HTTP 头中标识主机名的部分决定的。使用这种技术，很多虚拟服务器可以共享同一个 IP 地址。
- 基于 IP 地址的虚拟服务器：使用链接的 IP 地址来决定相应的虚拟服务器。这样，就需要为每个虚拟服务器分配一个独立的 IP 地址。

基于主机名的虚拟服务器相对比较简单，因为只需要配置 DNS 服务器，将每个主机名映射到

正确的 IP 地址，然后配置 Apache HTTP 服务器，令其识别不同的服务器名就可以了。基于域名的服务器也可以缓解 IP 地址不足的问题。所以，如果没有特殊原因要求必须使用基于 IP 的虚拟主机，最好还是使用基于名称的虚拟服务器。

无论使用哪种方式，只需要在 Apache2 配置文件的虚拟主机部分，为每个虚拟主机添加相应的配置描述。也就是在<VirtualHost></VirtualHost>段中为虚拟主机进行配置，下面来描述一下虚拟主机的基本配置参数（如表 21.3 所示）。

表 21.3 Apache2 虚拟主机配置描述

参数名称	描述
<VirtualHost>	用于封装一组仅作用于特定虚拟主机的配置。任何在虚拟主机配置中可以使用的配置同样可以在这里使用
NameVirtualHost	为一个基于域名的虚拟主机指定一个 IP 地址（和端口）
ServerName	服务器用于识别自己的主机名和端口号
ServerAlias	匹配一个基于域名的虚拟主机的别名
ServerPath	为兼容性不好的浏览器访问基于域名的虚拟主机保留的 URL 路径名

下面通过几个示例，演示使用不同的方式来配置虚拟服务器。

在一个 IP 地址上运行多个基于域名的 Web 站点

服务器只有一个 IP 地址，而在 DNS 中有很多域名(CNAMES)映射到这个 IP，需要在这个机器上运行 www.domain.com 和 www.domain.org 两个站点。

在 Apache 服务器配置中创建一个虚拟服务器并不会自动在 DNS 中对主机名做相应更新，用户必须自己在 DNS 中添加域名来指向自己的 IP 地址，否则别人是无法看到其 Web 站点的。用户可以在 hosts 文件中添加这一条目来进行测试，但这种方法仅适用于那些有这些 hosts 文件的计算机使用。

```
# 确保 Apache2 在监听 80 端口
Listen 80
#为虚拟主机在所有 IP 地址上监听
NameVirtualHost *:80

<VirtualHost *:80>
DocumentRoot /var/www/domain1
ServerName www.domain.com
# 你可以在这里添加其他配置参数
</VirtualHost><VirtualHost *:80>
DocumentRoot /var/www/domain2
ServerName www.domain.org
# 你可以在这里添加其配置参数
</VirtualHost>
```

因为*匹配所有 IP 地址，所以主服务器不接收任何请求。因为 www.domain.com 首先出现在配置文件中，所以它拥有最高优先级，可以认为是默认的主服务器。这意味着如果一个请求不能与某个 ServerName 字串相匹配，它将会由第一个<VirtualHost>所响应。

注意 如果愿意，用户可以用具体的 IP 地址来取代*。在这种情况下，VirtualHost 的参数必须与 NameVirtualHost 的参数相符。

```
NameVirtualHost 123.20.30.40

<VirtualHost 123.20.30.40>
# 其他
```

然而，当无法确定 IP 地址的时候，使用*是很方便的，比如说，ISP 配置的是动态 IP 地址，而用户又使用了某种动态域名解析系统时。因为*匹配任何 IP 地址，所以在这种情况下，不论 IP 地址如何变化，用户都不需要另外进行配置。

在多于一个 IP 的情况下使用基于域名的虚拟主机

服务器有两个 IP 地址。一个(123.20.30.40)用于主服务器 server.domain.com，另外一个(123.20.30.50)用于构建两个或多个虚拟主机。

```
Listen 80

# "主"服务器运行于: 123.20.30.40
ServerName server.domain.com
DocumentRoot /var/www/mainserver

#这是另外一个 IP 地址
NameVirtualHost 123.20.30.50

<VirtualHost 123.20.30.50>
DocumentRoot /var/www/domain1
ServerName www.domain.com
# 你可以在这里添加其他配置参数
</VirtualHost>

<VirtualHost 123.20.30.50>
DocumentRoot /var/www/domain2
ServerName www.domain.org
# 你可以在这里添加其他配置参数
</VirtualHost>
```

任何不是针对 123.20.30.50 的请求都将由主服务器来响应。而提交给 123.20.30.50 却没有主机名或没有 Host 头的请求，都将由 www.domain.com 响应。

在不同的 IP 的地址上提供相同的内容

服务器有两个 IP 地址(192.168.1.1 和 123.20.30.40)。这个机器位于内部网络（局域网）和外部网络（广域网）之间。在外部，域名 server.domain.com 指向外部地址(123.20.30.40)，而在内部则指向内部地址(192.168.1.1)。

服务器可以为来自内部和外部的请求提供同样的内容，只需要一个<VirtualHost>配置段就可以了。

```
NameVirtualHost 192.168.1.1
NameVirtualHost 123.20.30.40

<VirtualHost 192.168.1.1 123.20.30.40>
DocumentRoot /var/www/server1
ServerName server.domain.com
ServerAlias server
</VirtualHost>
```

这时候，从不同的网络提交的请求都会由同一个虚拟服务器来响应。

注意 在内网中，可以使用 server 这个名字来代替 server.domain.com 这个全名。跟上面一样，在上述的例子中，可以用*来代替具体的 IP 地址，这样就可以对所有的地址都返回相同的内容了。

在不同的端口上运行不同的站点

如果想让同一个 IP 的不同端口响应多个域名，可以借助在 NameVirtualHost 参数中定义端口的方法来达到这个目的。如果想使用不带“name:port”的<VirtualHost name:port>或直接用 Listen 指令，配置将无法生效。

```
Listen 80
Listen 8080

NameVirtualHost 123.20.30.40:80
NameVirtualHost 123.20.30.40:8080

<VirtualHost 123.20.30.40:80>
ServerName www.domain.com
DocumentRoot /var/www/domain-80
</VirtualHost>

<VirtualHost 123.20.30.40:8080>
ServerName www.domain.com
DocumentRoot /var/www/domain-8080
</VirtualHost>

<VirtualHost 123.20.30.40:80>
ServerName www.domain.org
DocumentRoot /var/www/otherdomain-80
</VirtualHost>

<VirtualHost 123.20.30.40:8080>
ServerName www.domain.org
DocumentRoot /var/www/otherdomain-8080
</VirtualHost>
```

建立基于 IP 的虚拟主机

一个有两个 IP 地址(123.20.30.40 和 123.20.30.50)的服务器分别对应域名 www.domain.com 和 www.domain.org 的配置如下：

```
Listen 80

<VirtualHost 123.20.30.40>
DocumentRoot /var/www/domain1
ServerName www.domain.com
</VirtualHost>

<VirtualHost 123.20.30.50>
DocumentRoot /var/www/domain2
ServerName www.domain.org
</VirtualHost>
```

如果存在主服务器，那么对没有出现在任一个<VirtualHost>段中的请求(比如，对 localhost 的请求)都会由主服务器来响应。

混用基于端口和基于 IP 的虚拟主机

如果服务器有两个 IP 地址(123.20.30.40 和 123.20.30.50)分别对应域名 www.domain.com 和 www.domain.org，希望在 80 端口和 8080 端口发布每个域名的网站，可以这样配置：

```
Listen 123.20.30.40:80
Listen 123.20.30.40:8080
Listen 123.20.30.50:80
Listen 123.20.30.50:8080

<VirtualHost 123.20.30.40:80>
DocumentRoot /var/www/domain1-80
ServerName www.domain.com
</VirtualHost>

<VirtualHost 123.20.30.40:8080>
DocumentRoot /var/www/domain1-8080
ServerName www.domain.com
</VirtualHost>

<VirtualHost 123.20.30.50:80>
DocumentRoot /var/www/domain2-80
ServerName www.domain.org
</VirtualHost>

<VirtualHost 123.20.30.50:8080>
DocumentRoot /var/www/domain2-8080
ServerName www.domain.org
</VirtualHost>
```

混合基于域名和基于 IP 的虚拟主机

如果想在一些地址上配置基于域名的虚拟主机而在另外一些地址上配置基于 IP 的虚拟主机，可以这样配置：

```
Listen 80
```

```
NameVirtualHost 123.20.30.40

<VirtualHost 123.20.30.40>
DocumentRoot /var/www/domain1
ServerName www.domain.com
</VirtualHost>

<VirtualHost 123.20.30.40>
DocumentRoot /var/www/domain2
ServerName www.domain.org
</VirtualHost>

<VirtualHost 123.20.30.40>
DocumentRoot /var/www/domain3
ServerName www.domain3.net
</VirtualHost>

# IP-based
<VirtualHost 123.20.30.50>
DocumentRoot /var/www/domain4
ServerName www.domain4.edu
</VirtualHost>

<VirtualHost 172.20.30.60>
DocumentRoot /var/www/domain5
ServerName www.domain5.gov
</VirtualHost>
```

21.4 课后练习

1. Linux 应用最广的 Web 服务器是什么？
2. 如何在 Ubuntu 中安装并启动 Apache 服务？
3. 说说 Apache 服务器的目录结构，如配置文件目录、运行 bin 目录、log 目录等。
4. Apache 的主配置文件是哪个？它的配置包括哪些项？
5. 在 Apache 中，如何设置虚拟服务器？

Chapter22 FTP 服务器——VSFTPD

在 Internet 上，很容易就能够找到某些机构或组织开放的 FTP 站点，从中可以下载所需要的资料文件。在这一章中将介绍 FTP 的基本原理，并通过实例讲述如何搭建 VSFTPD 服务，以及如何进行 VSFTPD 服务的配置。

22.1 FTP 简介

FTP（File Transfer Protocol，文件传输协议）的任务是从一台计算机将文件传送到另一台计算机，它与这两台计算机所处的位置、连接的方式，甚至是否使用相同的操作系统无关。假设两台计算机通过 FTP 协议对话，并且能访问 Internet，那就可以用 FTP 命令来传输文件。虽然每种操作系统使用上有一些细微差别，但是每种协议基本的命令结构是相同的。

FTP 标准命令使用的 TCP 端口号为 21，数据传输端口为 20。数据传输有两种方式：ASCII 传输模式和二进制数据传输模式。

- ASCII 传输方式：假定用户正在复制的文件包含简单的 ASCII 码文本，如果在远程机器上运行的不是 Unix，当文件传输时，FTP 通常会自动地调整文件的内容以便于把文件解释成另外那台计算机存储文本文件的格式。
- 二进制传输模式：在二进制传输中，保存文件的位序，以便原始和复制的是逐位一一对应的。如果在 ASCII 方式下传输二进制文件，即使不需要也会转成 ASCII。这会使传输稍微变慢，也会损坏数据，使文件变得不能用（在大多数计算机上，ASCII 方式一般假设每一字符的第一有效位无意义，因为 ASCII 字符组合不使用它。如果传输二进制文件，所有的位都是重要的）。如果你知道这两台机器是同样的，则二进制方式对文本文件和数据文件都是有效的。

除了传输模式外，FTP 还支持两种工作模式，一种方式叫做 Standard（也就是 Port 方式，主动方式），一种是 Passive（也就是 PASV，被动方式）。Standard 模式下 FTP 的客户端发送 PORT 命令到 FTP 服务器。Passive 模式下 FTP 的客户端发送 PASV 命令到 FTP 服务器。下面介绍这两种方式的工作原理。

- Port 模式：FTP 客户端首先和 FTP 服务器的 TCP 21 端口建立连接，通过这个连接通道发送命令，客户端需要接收数据的时候在这个通道上发送 PORT 命令。PORT 命令包含了客户端用什么端口接收数据。在传送数据的时候，服务器端通过自己的 TCP 20 端口连接至客户端的指定端口发送数据。FTP 服务器必须和客户端建立一个新的连接用来传送数据。
- Passive 模式：在建立控制通道的时候和 Standard 模式类似，但建立连接后发送的不是 PORT 命令，而是 PASV 命令。FTP 服务器收到 PASV 命令后，随机打开一个高端端口（端口号大于 1024）并且通知客户端在这个端口上传送数据的请求，客户端连接 FTP 服务器的此端口，然后 FTP 服务器将通过这个端口进行数据的传送，这个时候 FTP 服务器不再需要建立一个和客户端之间的新连接。

很多防火墙在设置的时候都是不允许接受外部发起的连接的，所以许多位于防火墙后或内网的FTP服务器不支持PASV模式，因为客户端无法穿过防火墙打开FTP服务器的高端端口。而许多内网的客户端不能用PORT模式登录FTP服务器，因为从服务器的TCP 20无法和内部网络的客户端建立一个新的连接，造成无法工作。

22.2 VSFTPD概述

VSFTPD是一个很好的FTP服务器软件。为什么这么说呢？因为VSFTPD是"very secure FTP daemon"的意思，所以它的发展本来就是以安全性为出发点的。VSFTPD在安全性的考虑上面，主要针对程序的权限概念来设计，因为任何服务在Linux上面运作时都会取得一个PID，而这个PID是有拥有者的身份的，也就是说，这个服务的PID在Linux上面是具有某些权限的。如果这个服务的PID所属拥有者的身份等级太高，例如root的权限，那么如果不幸该程序（PID）有些设计上的漏洞，使得该PID被入侵的话，入侵者将具有该PID的权限，也就是root的身份。所以，就现在的趋势看，开发的软件都是尽量将服务取得的PID权限降低，以便该服务即使被入侵了，入侵者也无法得到有效的系统管理权限，这样的话，系统就较为安全了。

VSFTPD是基于上面的说明来设计的一个较为安全的FTP服务器软件，它具有下面的特点：

- VSFTPD是以一般身份启动的服务，所以对于Linux系统的使用权限较低，对于Linux系统的危害就相对降低了。此外，VSFTPD利用chroot进行改换用户目录的动作，使得系统不会被VSFTPD服务所误用。
- 任何需要具有较高执行权限的VSFTPD命令均由一个特殊的上层程序（parent process）所控制，该上层程序享有的较高执行权限功能已经被限制得相当地低，以不影响Linux本身的系统为准。
- 来自客户端的请求想要使用这个上层程序所提供的较高执行权限之VSFTPD命令的请求，均被当作不可信任的请求来处理，必须经过一定的身份确认后，才能使用该上层程序的功能，例如chown()，Login命令等动作。
- 此外，上面提到的上层程序中，依然可以使用chroot功能来限制用户的执行权限。

由于具有这样的特点，所以VSFTPD比其他FTP服务器端软件具有更高的安全性。

22.3 搭建VSFTPD服务

在了解了FTP和VSFTPD的基本概念后，接下来将着重讲述VSFTPD在Ubuntu Linux下的安装方法，并对VSFTPD安装后的目录结构进行概要介绍，让读者对VSFTPD有个整体的认识。

22.3.1 VSFTPD的安装

在Ubuntu Linux下安装VSFTPD服务主要有两种方式，一种就是使用新立得软件包管理器安装，

另外一种就是在命令行下输入命令 apt-get 来安装。选用哪种方式来安装，主要依据系统安装的类型，如果安装的服务器版本没有 GUI 功能，那么就只能通过第二种方式通过命令行来安装，当然如果安装的是桌面版本的系统，那么这两种方式都能为你所用了。

使用新立得软件包管理器安装 VSFTPD

Step 01 通过执行【系统】|【系统管理】|【新立得软件包管理器】命令打开程序。

Step 02 在新立得软件包管理器窗口中，单击工具栏中的Search按钮，在弹出的小窗口中输入关键字vsftpd进行查找，如图22.1所示。

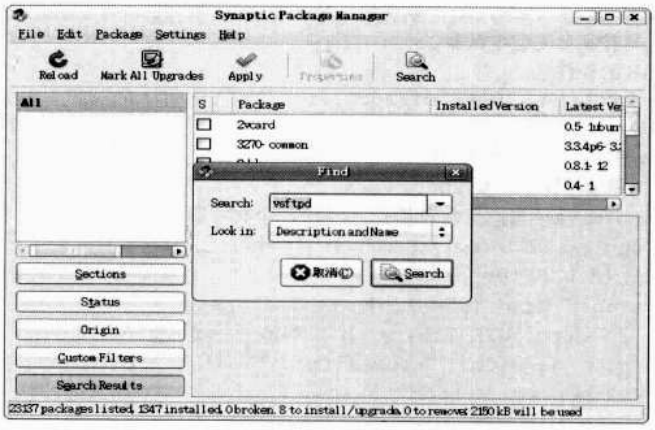


图 22.1 查找 VSFTPD 软件包

Step 03 在查询得出的结果中，选中vsftpd进行安装，如图22.2所示。

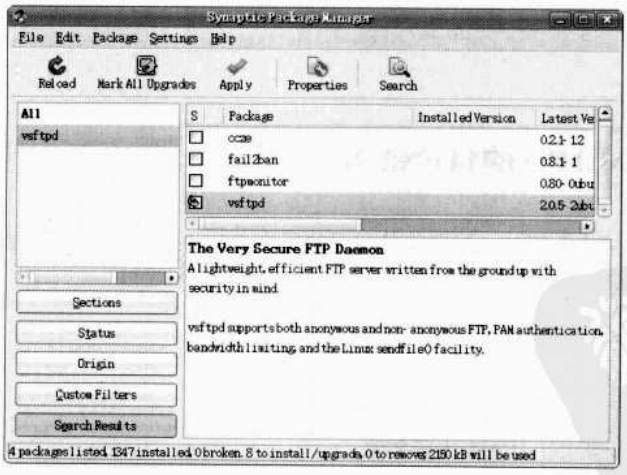


图 22.2 选择 vsftpd 软件包进行安装

在命令行下安装 VSFTPD

Step 01 通过执行【应用程序】|【附件】|【终端】命令打开命令行。

Step 02 在命令提示行下输入sudo apt-get install vsftpd，然后按回车键，系统就会自动安装vsftpd软件包，具体安装过程如下所示：

```
user@ubuntuer:~$ sudo apt-get install vsftpd
[sudo] password for user:
正在读取软件包列表...完成
正在分析软件包的依赖关系树
Reading state information...完成
下列【新】软件包将被安装:
  vsftpd
共升级了 0 个软件包，新安装了 1 个软件包，要卸载 0 个软件包，有 35 个软件未被升级。
有 1 个软件包没有被完全安装或卸载。
需要下载 96.8kB 的软件包。
After this operation, 401KB of additional disk space will be used.
获取: 1 http://cn.archive.ubuntu.com hardy/main vsftpd 2.0.6-1ubuntu1 [96.8kB]
下载 96.8kB，耗时 6s (15.6KB/s)
选中了曾被取消选择的软件包 vsftpd。
(正在读取数据库 ... 系统当前总共安装有 128027 个文件和目录。)
正在解压缩 vsftpd (从 .../vsftpd_2.0.6-1ubuntu1_i386.deb) ...
正在设置 vsftpd (2.0.6-1ubuntu1) ...
Adding system user 'ftp' (UID 116) ...
Adding new user 'ftp' (UID 116) with group 'nogroup' ...
Not creating home directory '/home/ftp'.
 * Starting FTP server: vsftpd                                    [ OK ]
```

从上面的过程中可以发现，安装程序没有创建 VSFTPD 的工作目录/home/ftp，这是因为当前用户 user 使用的 sudo 命令，暂时获得了根用户的权限，但是无法在/home 目录下创建用户 ftp 的个人目录。这时候，就需要在安装结束后手工创建该目录。

```
user@ubuntuer:~$ mkdir /home/ftp
```

22.3.2 VSFTPD 的目录结构

VSFTPD 的目录结构很简单，配置文件与可执行文件也不是很多，但不管怎么样，还是要了解一下，方便后续的学习。

- /usr/sbin/vsftpd：VSFTPD 的可执行文件。
- /etc/rc.d/init.d/vsftpd：启动脚本。
- /etc/vsftpd/vsftpd.conf：主配置文件。
- /etc/pam.d/vsftpd：PAM 认证文件。
- /etc/vsftpd.ftpusers：禁止使用 VSFTPD 的用户列表文件。
- /etc/vsftpd.user_list：禁止或允许使用 VSFTPD 的用户列表文件。
- /var/ftp：匿名用户主目录。

22.3.3 启动或关闭 VSFTPD

下命令行下管理 VSFTPD 服务器

在安装完 VSFTPD 服务后，VSFTPD 服务的守护进程以独占方式运行，持续不断地监听 21 端口，等待客户端发来的请求。用户可以使用下面的命令来手工启动和关闭 VSFTPD 服务。

启动 VSFTPD 服务器

```
root@ubuntuer: ~# /etc/init.d/vsftpd start
* Starting FTP server: vsftpd          [OK]
```

停止 VSFTPD 服务器

```
root@ubuntuer: ~# /etc/init.d/vsftpd stop
* Stopping FTP server: vsftpd          [OK]
```

重启 VSFTPD 服务器

```
root@ubuntuer: ~# /etc/init.d/vsftpd restart
* Stopping FTP server: vsftpd          [OK]
* Starting FTP server: vsftpd          [OK]
```

使用图形界面管理 VSFTPD 服务器

在 Ubuntu 系统中，同样可以使用服务设置工具来管理 VSFTPD 服务器的状态，执行【系统】|【系统服务】|【服务】命令，打开【服务设置】对话框，点选【FTP 服务（vsftpd）】来启动或关闭 VSFTPD 服务。如图 22.3 所示。

图 22.3 使用服务设置工具管理 VSFTPD 服务

22.4 VSFTPD 服务配置

VSFTPD 的服务配置跟其他服务一样，都是通过修改其配置文件来进行的。在其配置文件中，包含了 VSFTPD 的各个方面的配置参数，只有了解了这些配置参数的意义，才能保证 FTP 稳定可靠地运行。

22.4.1 VSFTPD 配置文件

从上面的章节中知道，VSFTPD 服务的主配置文件是/etc/vsftpd.conf，在这个配置文件中，每个配置参数都可以当作是一个规则，用于规范对外提供 FTP 服务的行为特性。比如，是否允许匿名用户登录，允许多少访问连接等。而这些规则都是以“参数名=值”的格式来表示，并且在“=”两边不能留有空格。在配置文件中，每行首字符如果是“#”，那么该行就是注释，通常用于追加配置参数的说明性文字或者是用于屏蔽某些配置参数。

VSFTPD 的配置参数大概可以分为用户设置、连接设置、服务性能设置、目录和文件操作设置、安全措施设置、提示信息设置和日志设置这 7 大类。表 22.1 至 22.7 对 VSFTPD 的配置参数按照上面的划分进行了归纳和说明，以供读者参考。

表 22.1 VSFTPD 用户设置参数

参数名称	描述	默认值
anonymous_enable=YES|NO	设置是否允许匿名用户登录，YES 允许，NO 不允许	YES
ftp_username=username	匿名用户所使用的系统用户名	ftp
no_anon_password=YES|NO	设置匿名用户登录时是否需要密码，YES 不需要，NO 需要。	YES
deny_email_enable=YES|NO	设置是否拒绝用户使用 banned_email_file 文件中所列出的 E-mail 进行登录。这对于阻止某些 Dos 攻击很有效。如果设置为 YES，需追加 banned_email_file 参数	NO
banned_email_file=file_name	指定包含被拒绝的 E-mail 地址的文件	/etc/vsftpd.banned_emails
anon_root=dir_name	设置匿名用户的根目录，即匿名用户登入后，进入此目录下	/var/ftp
anon_world_readable_only=YES|NO	设置是否只允许匿名用户下载可阅读文件。YES，只允许匿名用户下载可阅读的文件。NO，允许匿名用户浏览整个服务器的文件系统	YES
anon_upload_enable=YES|NO	设置是否允许匿名用户上传文件，YES 允许，NO 不允许	NO
anon_mkdir_write_enable=YES|NO	设置是否允许匿名用户创建新目录，YES 允许，NO 不允许	NO
anon_other_write_enable=YES|NO	设置匿名用户是否拥有除了上传和新建目录之外的其他权限，如删除、更名等。YES 拥有，NO 不拥有	NO

（续表）

参数名称	描述	默认值
chown_uploads=YES|NO	是否修改匿名用户所上传文件的所有权。YES，匿名用户所上传的文件的所有权将改为另外一个不同的用户所有，用户由 chown_username 参数指定	NO
chown_username=whoever	设置拥有匿名用户上传文件所有权的用户。此参数与 chown_uploads 联用。一般情况下不推荐使用 root 用户	无
local_enable=YES|NO	设置 vsftpd 所在的系统的用户是否可以登录 vsftpd	YES
local_root=dir_name	所有本地用户的根目录。当本地用户登入时，将被更换到此目录下	无
user_config_dir=dir_name	用户个人配置文件所在的目录。用户的个人配置文件为该目录下的同名文件	无
guest_enable=YES|NO	若是启动这项功能，所有的非匿名登入者都视为 guest	NO
guest_username=user_name	设置 VSFTPD 的 guest 用户在系统中的用户名	ftp

表 22.2　VSFTPD 连接设置参数

参数名称	描述	默认值
listen_address=ip address	该参数在 VSFTPD 使用独占(stand alone)模式下有效。设置主机在哪个 IP 地址上监听 FTP 请求，即在哪个 IP 地址上提供 FTP 服务。对于只有一个 IP 地址的主机，不需要使用此参数。对于多址主机，不设置此参数，则监听所有 IP 地址	无
listen_port=port_value	设置 FTP 服务器监听的端口号(控制端口)	21
port_enable=YES|NO	设置是否使用 PORT 模式进行数据传输。FTP 服务数据传送的两种模式：PORT 模式使用 21 和 20 端口；PASV 模式可以使用 1024 以上所有 TCP 端口和 21 端口	YES
connetc_from_port_20=YES|NO	设置在 PORT 模式进行数据传输时是否使用 20 端口(ftp-data)。YES 使用，NO 不使用	NO
ftp_data_port=port number	设置 ftp 数据传输端口(ftp-data)值。此参数用于 PORT FTP 模式	20
port_promiscuous=YES|NO	设置 PORT 模式下输出的数据只能连接到客户端。YES 为取消 PORT 安全检查，NO 为对 PORT 进行安全检查	NO

（续表）

参数名称	描述	默认值
pasv_enable=YES|NO	YES，允许数据传输时使用 PASV 模式。NO，不允许使用 PASV 模式	YES
pasv_min_port=port number	设置 PASV 模式下，建立数据传输所可以使用端口范围的下界，0 表示任意	0
pasv_max_port=port number	设置 PASV 模式下，建立数据传输所可以使用端口范围的上界，0 表示任意	0
pasv_promiscuous=YES|NO	设置PASV模式下输出的数据只能连接到客户端。YES 为取消 PASV 安全检查，NO 为对 PASV 进行安全检查	NO
pasv_address=	设置作为 PASV 命令响应的 IP 地址	无
ascii_upload_enable=YES|NO	设置是否允许使用 ASCII 模式上传文件，YES 允许，NO 不允许	NO
ascii_download_enable=YES|NO	设置是否允许使用 ASCII 模式下载文件，YES 允许，NO 不允许	NO

表 22.3 VSFTPD 服务性能设置参数

参数名称	描述	默认值
idle_session_timeout= numerical value	设置空闲（发呆）用户会话的超时时间，若是超出该时间没有数据的传送或是指令的输入，则会强迫断线，单位为秒	300
data_connection_timeout= numerical value	设置空闲的数据连接的超时时间，单位为秒	300
accept_timeout=numerical value	设置接受建立联机的超时时间，单位为秒	60
connect_timeout=numerical value	设定以 PORT 方式的数据联机的超时时间，单位为秒	60
max_clients=numerical value	该参数在 VSFTPD 独占(stand alone)模式下有效。设置 FTP 服务器最大的并发连接数，当超过此连接数时，服务器拒绝客户端连接。0 表示不限制	0
max_per_ip=numerical value	该参数在 VSFTPD 独占(stand alone)模式下有效。设置每个 IP 地址最大的并发连接数目。超过这个数目将会拒绝连接。此选项的设置将影响到像网际快车这类的多进程下载软件。0 表示不限制	0
anon_max_rate=value	设置匿名用户的最大数据传输速度 value，以 Byte/s 为单位	无
local_max_rate=value	设置普通用户的最大数据传输速度 value，以 Byte/s 为单位	无

表 22.4 VSFTPD 目录和文件操作设置参数

参数名称	描述	默认值
chroot_list_enable=YES|NO	限定某些用户在个人目录中。即当这些用户登录后，不可以转到系统的其他目录，只能在个人目录(及其子目录)下。具体的用户在 chroot_list_file 参数所指定的文件中列出。YES 为可以转到系统其他目录，NO 只能访问个人目录	NO

（续表）

参数名称	描述	默认值
chroot_list_file=file_name	指定被锁定在个人目录中的用户的列表文件	etc/vsftpd/chroot_list
chroot_local_users=YES|NO	设定本地用户是否只能访问个人目录	NO
hide_ids=YES|NO	设置是否隐藏文件的所有者和组信息。YES，当用户使用 ls -al 之类的指令时，在目录列表中所有文件的拥有者和组信息都显示为 ftp。NO 正常显示	NO
ls_recurse_enable=YES|NO	设置是否允许使用 ls -R 指令，YES 为允许，NO 为不允许	NO
write_enable=YES|NO	设置是否允许使用任何可以修改文件系统的 FTP 的指令。YES 为允许，NO 为不允许	NO
secure_chroot_dir=dir_name	设置一个空目录，并且 FTP 用户对此目录无写权限。当 VSFTPD 不需要访问文件系统时，这个目录将被作为一个安全的容器，用户将被限制在此目录中	/usr/share/empty
anon_umask=mask_decimal	设置匿名用户新增文件的 umask 数值	077
file_open_mode=mask_decimal	设置上传文件的 umask 数值	0666
local_umask= mask_decimal	设置本地用户新增文件时的 umask 数值	077

表 22.5　VSFTPD 安全措施设置参数

参数名称	描述	默认值
pam_service_name=vsftpd	设置 VSFTPD 进行 PAM 认证时所使用的 PAM 配置文件名	/etc/pam.d/vsftpd
userlist_enable=YES|NO	设置是否需要进行用户密码验证，YES 为当 userlist_file 文件中的用户请求登录时，无需验证密码，NO 为需要验证密码	NO
userlist_file=file_name	userlist_enable 为 YES 时，被读取的包含用户列表的文件	/etc/vsftpd.user_list
userlist_deny=YES|NO	设置只允许由 userlist_file 指定文件中的用户登录 FTP 服务器。此选项在 userlist_enable 选项启动后才生效。YES，禁止文件中的用户登录，同时也不向这些用户发出输入口令的提示。NO，只允许在文件中的用户登录 FTP 服务器	YES
tcp_wrappers=YES|NO	设置是否允许使用 TCP_Wrappers 远程访问控制机制。YES 为允许，NO 为不允许	YES

表 22.6　VSFTPD 提示信息设置参数

参数名称	描述	默认值
ftpd_banner=login banner string	设置登录欢迎语字符串	无
banner_file=file_name	指定一个文本文件，当使用者登入时，会显示该文件的内容，通常为欢迎话语或说明	无

（续表）

参数名称	描述	默认值
dirmessage_enable=YES|NO	设置是否启用目录提示信息功能。当用户进入某一个目录时，会检查该目录下是否有 message_file 参数所指定的文件，若有，则会出现此文件的内容，通常这个文件会放置欢迎话语，或是对该目录的说明。YES 启用，NO 不启用	YES
message_file=file_name	仅在 dirmessage_enable 为 YES 时生效，指定消息文件名	.message

表 22.7 VSFTPD 日志设置参数

参数名称	描述	默认值
xferlog_enable=YES|NO	设置是否启用一个日志文件，用于详细记录上传和下载。该日志文件由 xferlog_file 参数指定	NO
xferlog_file=file_name	指定记录传输日志的文件名	/var/log/vsftpd.log
xferlog_std_format=YES|NO	设置日志文件是否像 wu-ftpd 一样使用 xferlog 的标准格式。使用 xferlog 格式，可以重新使用已经存在的传输统计生成器。不过，默认的日志格式更具可读性	NO
log_ftp_protocol=YES|NO	设置是否把所有的 FTP 请求和响应都记录到日志中	NO

可以看出，VSFTPD 的配置参数是非常繁多的，不过对于大多数的用户来说，进行 VSFTPD 的配置只需要修改一些特定的参数就可以了，因为 VSFTPD 配置中的默认值已经是相当完备。如果真的要修改一个并不熟悉的参数项，那在操作前查阅一下该参数的设置说明，相信也并不是很困难。

22.4.2 配置欢迎信息

用户使用 FTP 客户端连接到服务器的时候，会看见一些欢迎信息，提示用户登录成功，或是 FTP 站点的一些基本信息等。如何在服务器端配置这些信息呢？很简单，如果信息很简短的话，可以直接修改/etc/vsftpd.conf 中的 ftpd_banner 配置参数，把信息字符串作为变量配置在参数上。当然，如果提示信息比较复杂，可以保存在一个独立的文件中，然后由 VSFTPD 启动脚本自动读取该文件。下面是具体的配置方法：

Step 01 创建一个.message文件，在文件中输入提示信息。

Step 02 把.message文件复制到VSFTP主目录中，默认情况下是/home/ftp。如果存在多个FTP账号，可以按需要到指定的账户目录中。

Step 03 修改/etc/vsftpd/vsftpd.conf配置文件。

```
Dirmessage_enable=YES

Message_file=.message
```

Step 04 重启服务，这样，用户登录到VSFTPD服务的时候，就可以看见这个欢迎信息了。

22.4.3 允许匿名用户上传文件

出于安全方面的考虑，默认情况下，VSFTPD 服务并不允许匿名用户上传文件。如果 VSFTPD 服务只是作为内部交流或家庭使用，完全在一个可以信任的环境中，这时候，可以放开匿名用户的限制，允许上传文件。配置方法如下：

Step 01 修改/etc/vsftpd/vsftpd.conf配置文件。

```
#允许匿名用户写权限
write_enable=YES
#允许匿名用户浏览目录
anon_world_readable_only=NO
#允许匿名用户上传文件
anon_upload_readable_only=YES
#允许匿名用户具有写创建目录的权限
anon_mkdir_write_enable=YES
```

Step 02 创建匿名用户上传文件的目录。

```
root@ubuntuer: ~# mkdir /home/ftp/incoming
root@ubuntuer: ~# chown ftp.ftp /home/ftp/incoming
    //因为匿名用户以 ftp 用户在文件内操作，而 incoming 目录是 root
      用户创建的，所以把 incoming 目录转移到 ftp 用户下，以便 ftp 用户能读写该目录
```

Step 03 重启服务。

22.4.4 限制下载速度

有的时候用户可能不希望 FTP 服务在上传/下载数据中被占用太多的带宽，而影响服务器的其他工作，所以限制用户传输带宽也是很有必要的，假设要限制所有用户的上传/下载最大速度为 100KB/s 秒，修改/etc/vsftpd/vsftpd.conf 配置文件如下：

```
local_max_rate=100000
```

然后再重启服务，设置就生效了。当然，可以根据自己的网络状况来限制用户具体的使用带宽。

22.4.5 限制来自同一 IP 的最大连接数

除了控制带宽使用外，还可以通过限制最大的连接数来控制用户对服务器资源的占用，假如希望最多只有 50 个人同时使用 FTP 的话，并且每个 IP 地址最多只能建立 5 个连接时，修改/etc/vsftpd/vsftpd.conf 配置文件如下：

```
max_clients=10
max_per_ip=1
```

重启服务以便修改生效。

22.4.6 虚拟路径设置

假定 VSFTPD 的默认目录是/home/ftp，现在想把/media/hda5/pub 文件夹映射到/home/ftp 目录下，操作过程如下：

```
root@ubuntuer: ~# mkdir /home/ftp/virtual  //在/home/ftp 目录中创建目录 virtual
root@ubuntuer: ~# sudo mount --bind /media/hda5/pub /home/ftp/virtual
```

非常简单，而且也比较实用，特别是 FTP 操作目录下磁盘空间不足的时候，可以通过这个方法加入更多的磁盘容量。

22.5 课后练习

1. 说说 FTP 协议，以及 FTP 服务器的起源、发展。
2. 简单了解一下 VSFTP 的产生、特点及应用。
3. 如何进行 VSFTP 的安装？安装中有哪些注意事项？
4. 说说 VSFTP 的目录结构及各目录的功能。
5. 如何在日常维护中对 VSFTP 进行配置？VSFTP 的主配置文件是哪个？包括哪些配置项？
6. 如何启动、停止、重启 VSFTP？
7. 如何设置虚拟下载路径？如何控制权限，包括读权限、写权限等？

Chapter 23 邮件服务器——Postfix

Postfix 是一个邮件传输客户端(MTA)，它也是 Ubuntu 中默认的邮件传输客户端。它是 Ubuntu 的 main 软件库中的一个软件。这意味着它拥有安全更新。Postfix 与传统 Linux 下的 Sendmail 类似，但因为 Postfix 比 Sendmail 出现得晚，而 Postfix 的开发人员吸取了 Sendmail 的许多经验教训，从而使得 Postfix 比 Sendmail 具有更好的安全结构和方便易用的特性。在这一章中，我们将主要讲述 Postfix 的安装配置、启动和关闭等，当然，在开始这一切之前，还是先从最基本的概念——邮件服务器基础开始。

23.1 邮件服务器基础

在开始介绍邮件的传送过程之前，先来想一想，你是如何寄出电子邮件？假设要寄信给一个用户，他的电子邮件是 user@gmail.com，也就是说，要寄一封邮件到 gmail.com 这个主机上。个人计算机是否能够将这封邮件直接通过网络送给 gmail.com 的主机上？ 当然不行啦！要设定转信的邮件主机才行！也就是说，必须先向某一台邮件主机注册，以取得一个合法的电子邮件账号使用权限后，才能够发送邮件。

所以说，在寄出一封邮件时是需要很多接口帮忙的，下面列出一个简单的示意图（见图 23.1）来说明。

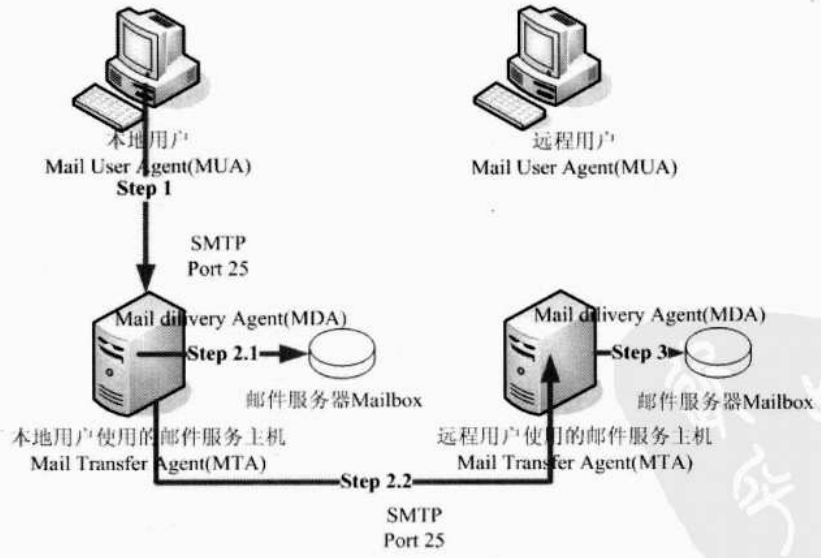


图 23.1 电子邮件传送的流程示意图

下面先来解释一些专有名词，然后再来说明传送的流程。

MUA

除非可以直接利用 telnet 之类的软件登录邮件主机来主动发出邮件，否则就要通过 MUA (Mail User Agent，邮件用户代理)来帮忙送信到邮件主机上去。最常见的 MUA 有 Mozilla 推出的 Thunderbird（雷鸟）自由软件、Linux 桌面 KDE 常见的 Kmail，及 Windows Office 的 Outlook Express (OE) 等。MUA 的主要功能就是接收邮件主机的电子邮件，以及提供用户浏览与撰写电子邮件的功能。

MTA

MUA 帮用户传送邮件到邮件主机上，那如果这台邮件主机能够帮用户将这封信寄出去，那它就是一台邮件传送主机，这台邮件传送主机就是邮件传送代理人 (Mial Transfer Agent，MTA)。MTA 的功能有以下几点：

(1) 收受邮件：将来自客户端或者是其他 MTA 的来信收下来，这个时候 MTA 使用的是简单邮件传输协议 (Simple Mail Transfer Protocol，SMTP)，使用的端口是 25。

(2) 传递邮件：如果该封邮件的目的地并不是本身的用户，且该信的相关数据符合使用 MTA 的协议，那么 MTA 就会将该封信再传送到下一台主机上。这即是所谓的传递 (Relay) 的功能。

(3) 响应用户的收信要求：用户可以通过 MTA 主机提供的邮件服务协议 (Post Office Protocol，POP) 来收下自己的邮件，也可以通过 (Internet Message Access Protocol，IMAP) 将自己的邮件保留在邮件主机上面，并进一步建立邮件文件夹等。

总之，通常说的邮件服务器就是 MTA，但严格上来说，MTA 其实只是指 SMTP 这个协议而已。而达成 MTA 的 SMTP 功能的主要软件包括老牌的 Sendmail、后起之秀的 Postfix，还有 Qmail 等。

MDA

MAD (Mait Delivery Agent，邮件发送代理人) 字面上的意思是邮件发送代理人。事实上，这个 MDA 是绑定在 MTA 下的一个小程序，它最主要的功能就是：分析由 MTA 所收到的邮件表头或内容等数据，来决定这封邮件的去向。所以说，上面提到的 MTA 的邮件传递功能，其实是由 MDA 实现的。举例来说，如果 MTA 收到的这封信的目标是自己，那么 MDA 会将这封信转到用户的信箱 (Mailbox) 去，如果不是，那就准备要传递出去。此外，MDA 还有分析与过滤邮件的功能。

(1) 过滤垃圾邮件：可以根据该封邮件的主题资料，或者是特定的邮件内容来加以分析过滤。例如某个广告信的主题都是固定的，如“法轮功”，“色情”等，都可以通过 MDA 来过滤并删除该邮件。

(2) 自动回复：如果用户出差了导致某一段时间内无法立即回复邮件时，就可以通过 MDA 的功能让邮件主机自动发出回复邮件，及时告知邮件发送人用户现在的状况，而避免不必要的误解。

Mailbox

Mailbox 就是电子邮件信箱，简单地说，就是某个账号专用的邮件接收文件夹。Ubuntu Linux 系统默认的信箱都是放在/var/spool/mail/用户账号中。如果 MTA 所收到的邮件是本机用户，MDA 就会将邮件送到该 Mailbox 当中去。

在了解了这些名词后，下面来讲述一下 MUA 是如何将邮件送到对方的邮件信箱中的。

Step 01 取得某个MTA的使用权限。

如图23.1所示，本地端的MUA 想要使用MTA来发出邮件时，必须取得 MTA的使用权限。通常，我们必须向MTA注册一组可用Email账号与密码才行。

Step 02 用户在MUA 上编写邮件后，传送至 MTA 上面。

用户在MUA上编写邮件，邮件的数据主要有：

- 邮件标头：包括发件人的E-mail以及收件人的E-mail地址，还有该封邮件的主题（subject）等。
- 邮件内容：就是需要告知收件人的具体内容。

编写完毕之后只要单击发送按钮，该封信就会送至MTA主机上面了。注意：是发件人的MTA 而不是对方的MTA。如果确定可以使用该MTA服务器，那么这封信就会被放置到MTA的队列（queue）当中并等待发送出去。

Step 03 如果该封信的目标收件人是本地 MTA 自己的账号。

用户是可以寄信给自己的，所以如果MTA 收到该封邮件的目标是自己的用户时，那就会通过 MDA 将这封信送到Mailbox 去。

Step 04 如果该封信的目标收件人为其他 MTA，则开始传递（Relay）的流程。

这个时候MTA就开始分析该封信是否具有合法的使用权限，如果具有使用权限时，则MDA会开始进行邮件传递，也就是将该邮件通过MTA向下一台MTA的smtp（端口 25）发送出去。如果该邮件顺利地发送出去了，那么该邮件就会由队列当中移除掉了。

Step 05 对方 MTA 主机接收邮件。

如果一切都没有问题的话，远程的MTA 会收到本地MTA所发出的那封信，并将该邮件放置到正确的使用者信箱当中，等待用户登入来读取或下载。

在整个过程当中，读者会发现邮件是由 MTA 帮忙发送出去的，此时 MTA 提供的协议是简单邮件传输协议，并且该封信最终是停留在对方主机的 MTA 上，并不是远程用户的 MUA 上。只有远程用户使用 MUA 收邮件时，才从 MTA 上收取邮件到 MUA 上。

23.2 Postfix 概述

Postfix 是一个 IBM 资助下由 Wietse Venema 负责开发的自由软件工程的一个产物，其目的是为用户提供除 Sendmail 之外的邮件服务器选择。Postfix 力图做到快速、易于管理、提供尽可能的安全性，同时尽量做到和 Sendmail 邮件服务器保持兼容性以满足用户的使用习惯。起初，Postfix 是以 VMailer 这个名字发布的，后来由于商标的原因改名为 Postfix。

23.2.1 设计目标

Postfix 的开发目标就是实现一个邮件服务器，提供给用户除 Sendmail 以外的选择。其中还包括以下更为具体的目标。

性能

比同类的服务器产品速度快三倍以上，一台安装 Postfix 的台式机一天可以收发百万封邮件。Postfix 设计中采用了 Web 服务器的设计技巧以减少进程创建开销，并且采用了其他一些文件访问优化技术以提高效率，但同时保证了软件的可靠性。

兼容性

Postfix 设计一开始就考虑了保持 Sendmail 的兼容性问题，以使移植变得更加容易。Postfix 支持/var[/spool]/mail、/etc/aliases、NIS 及 ~/.forward 等文件。然而 Postfix 为保证管理的简单性，并没有支持配置文件 Sendmail.cf。

健壮性

Postfix 设计上实现了程序在过量负载情况下仍然保证程序的可靠性。当出现本地文件系统没有可用空间或没有可用内存的情况时，Postfix 就会自动放弃，而不是重试使得服务器状况变得更加糟糕。

灵活性

Postfix 结构上由十多个小的子模块组成，每个子模块完成特定的任务，如通过 SMTP 协议接收一个消息，发送一个消息，本地传递一个消息，重写一个邮件地址等。当出现特定的需求时，可以用新版本的模块来替代老的模块，而不需要更新整个程序。而且它也很容易实现启动或关闭某个功能。

安全性

Postfix 使用多层防护措施防范攻击者来保护本地系统，几乎每一个 Postfix 守护进程都能运行在固定低权限的 chroot 之下，在网络和安全敏感的本地投递程序之间没有直接的路径，如果一个攻击者想侵入本地系统，那他必须首先突破若干个其他的程序，才有可能访问到本地系统。Postfix 甚至不绝对信任自己的队列文件或 IPC 消息中的内容以防止被欺骗。Postfix 在输出发送者提供的消息之前会首先过滤消息。

23.2.2 Postfix 的特点

Postfix 有以下特点。

支持多传输域

Sendmail 支持在 Internet、DECnet、X.400 及 UUCP 之间转发消息，Postfix 则灵活地设计为无需虚拟域（virtual domain）或别名来实现这种转发。

虚拟域

在大多数通用情况下，增加对一个虚拟域的支持仅仅需要改变一个 Postfix 查找信息表，其他的邮件服务器则通常需要多个级别的别名或重定向来获得这样的效果。

UCE 控制

Postfix 能限制哪台主机允许通过自身转发邮件，并且支持限定什么邮件允许接进。Postfix 实现通常的控制功能包括：黑名单列表、RBL 查找、HELO/发送者 DNS 核实，以及基于内容的过滤。

表查看

Postfix 没有实现地址重写语言，而是使用了一种扩展的表查看来实现地址重写功能。表可以是本地 dbm 或 db 文件等格式。

23.2.3 Postfix 体系结构

Postfix 是基于半驻留、互操作的进程的体系结构，每个进程完成特定的任务，没有任何特定的进程衍生关系(父子关系)。而且，独立的进程来完成不同的功能相对于“单块”程序具有更好的隔离性。此外，这种实现方式具有这样的优点：每个服务如地址重写等都能被任 何一个 Postfix 部件所使用，无需进程创建等开销，而仅仅需要重写一个地址，当然并不是只有 Postfix 采用这种方式。

Postfix 是按照这种方式实现的：一个驻留主服务器根据命令运行 Postfix 守护进程，守护进程完成发送或接收网络邮件消息，在本地递交邮件等功能。守护进程的数目由配置参数来决定，并且根据配置决定守护进程运行的次数(re-used times)，当空闲时间到达配置参数指定的限度时，自动消亡。这种方法明显地降低了进程创建开销，而单个进程之间仍然保持了良好的隔离性。

Postfix 的设计目标就是成为 Sendmail 的替代者。由于这个原因，Postfix 系统的很多部分，如本地投递程序等，可以很容易地通过编辑修改类似 inetd 的配置文件来替代。

Postfix 的核心是由十多个半驻留程序实现的。为了保证机密性，这些 Postfix 进程之间通过 Unix 的 socket 或受保护的目录之下的 FIFO 进行通信。即使使用这种方法来保证机密性，Postfix 进程并不盲目信任其通过这种方式接收到的数据。

Postfix 进程之间传递的数据量是有限制的。在很多情况下，Postfix 进程之间交换的数据信息只有队列文件名和接收者列表，或某些状态信息。一旦一个邮件消息被保存进入文件，将在其中一直保存直到被一个邮件投递程序读出。

Postfix 采用一些通常的措施来避免丢失信息。在收到确认信息以前通过调用 flush 和 fsync()保存所有的数据到磁盘中，检查所有的系统调用的返回结果来避免错误状况。

大多数构建邮件服务器者都会选择 Sendmail，公平地来讲，Sendmail 是一个不错的 MTA，最初开发时，Eric Allman 的设计考虑主要放在了邮件传递的成功性。不幸的是，Sendmai 开发时没有太多地考虑 Internet 环境下可能遇到的安全性问题。Sendmail 在大多数系统上只能以根用户身份运行，这就意味着任何漏洞都可能导致非常严重的后果，除了这些问题之外，在高负载的情况下 Sendmail 的运行情况也不是很好。

23.2.4 安全性

Postfix 必须以 root 的身份运行，这个主（master）程序以 root 身份运行后，其生成进程就可以用来处理接入、发出及本地邮件投递工作。通过使用一系列模块部件，每个任务由一个单独的程序

来运行（这样使升级变得容易一些）。例如发出邮件被卸载到一个队列目录中，在这里 pickup 程序取到该邮件，然后将邮件传递给 cleanup 程序，其再将邮件传递给 trivial-rewrite 程序，trivial-rewrite 程序负责处理邮件头，最后若邮件目的是别的系统则将邮件传递给 smtp 程序。而且相对于 Sendmail 来说，Postfix 也更容易设置 chrooted 环境，只要简单地通过编辑 master.cf 文件（一般位于/etc/postfix 内）即可实现，并且 Postfix 将运行 chrooted，以限定在其定义的队列目录之下（通常位于/var/spool/postfix），同样可以在 master.cf 中对 Postfix 的单一模块设置进程限制。用户可以限制 Postfix 以哪个用户的身份运行，一般来说是以 postfix 用户（概念上该用户和 Apache 的 nobody 类似）运行，该用户可以访问特定的队列目录。Postfix 其他的主要优点是它的配置文件清晰易懂。

如果想更多地了解 Postfix 的信息，可以访问下面的英文站点。

- Postfix 主页：www.postfix.org
- Postfix 资源和 FAQ：www.seaglass.com/postfix
- Postfix 圣地：www.stahl.bau.tu-bs.de/~hildeb/postfix

23.3 搭建 Postfix 服务

通过上面的学习，读者或许已经对邮件服务器和 Postfix 有了一个整体的认识，接下来将讲述如何在 Ubuntu Linux 下搭建 Postfix 服务器，包括安装 Postfix、Postfix 的目录结构以及如何启动和关闭 Postfix 服务。

23.3.1 Postfix 的安装

就像前面提到的一样，在 Ubuntu Linux 下安装软件主是有两种方式，一种就是使用新立得软件包管理器安装，另外一种就是在命令行下输入命令 apt-get 来安装。Postfix 也一样，可以选用任何一种方式来安装，但如果安装的是服务器版本而没有 GUI 功能的话，那么只能通过第二种方式通过命令行来安装，当然如果安装的是桌面版本的系统，那么这两种方式都能为你所用了。

使用新立得软件包管理器安装 Postfix

Step 01 通过执行【系统】|【系统管理】|【新立得软件包管理器】命令打开程序。

Step 02 在新立得软件包管理器窗口中，单击工具栏中的Search按钮，在弹出的小窗口中输入关键字postfix进行查找，如图23.2所示。

Step 03 在查询结果中，选择postfix、postfix-doc、postgrey软件包进行安装，如图23.3所示。

Step 04 在安装Postfix的过程中，系统会弹出选择邮件服务器配置模型的窗口，如图23.4所示。这个窗口可以提供不同的配置模型选项，一般情况下，选择Internet Site选项就可以了，但是也可以单击右侧的下三角按钮，在弹出的下拉列表中选择另外的配置类型，在选择完毕后，单击【前进】按钮。

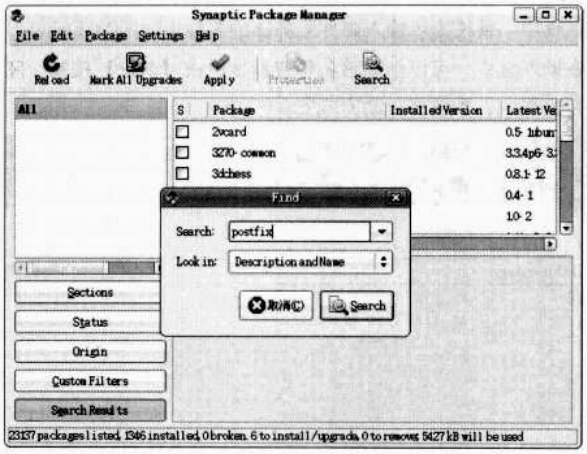


图 23.2　查找 postfix 软件包

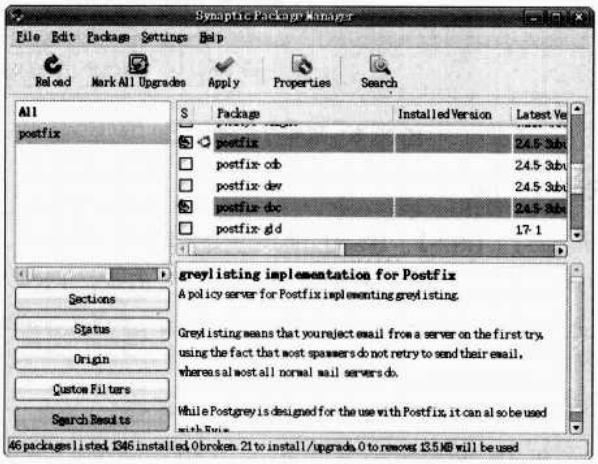


图 23.3　选中 postfix、postfix—doc 软件包进行安装

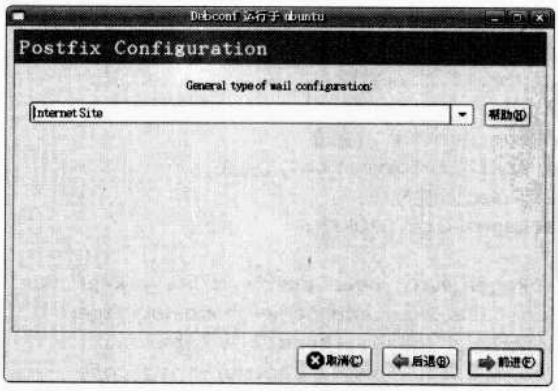


图 23.4　选择邮件服务器配置模型

Step 05 在接下来的对话框中，系统提示输入邮件服务器的名称，如图23.5所示。默认情况下，系统将以自己的名称显示在输入框中，可以修改该名称或直接单击【前进】按钮，完成邮件服务器名称设置。在接下来的安装过程中，系统将自动完成安装Postfix。

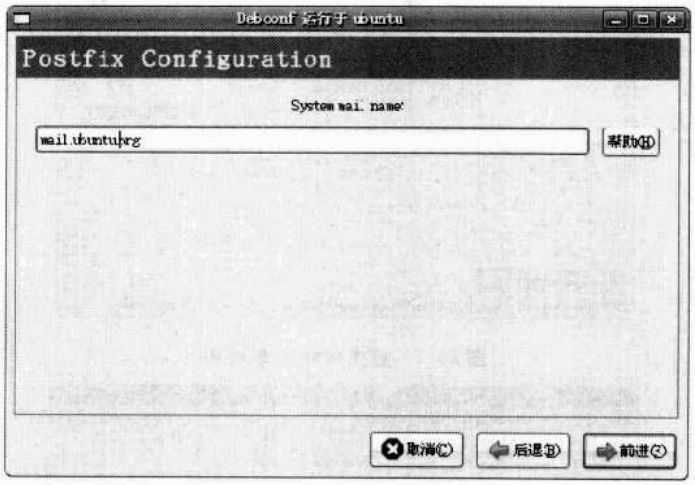


图 23.5 邮件服务器名称

在命令行下安装 Postfix

Step 01 通过执行【应用程序】|【附件】|【终端】命令打开命令行。

Step 02 为了安装Postfix服务器及其关联包，需要使用aptitude命令，在命令行下输入下面的命令：

```
sudo aptitude -r install postfix postfix-doc postgrey mailscanner mailx
qpopper clamav
```

Step 03 按回车键，然后系统就会自动安装Postfix，安装过程如下所示：

```
user@ubuntuer:~$ sudo aptitude -r install postfix postfix-doc postgrey
mailscanner mailx qpopper clamav
[sudo] password for user:
正在读取软件包列表...完成
正在分析软件包的依赖关系树
Reading state information...完成
Reading extended state information
Initializing package states...完成
Writing extended state information...完成
Building tag database...完成
The following packages are BROKEN:
  libperl5.8
The following packages have been automatically kept back:
  language-pack-en-base language-pack-gnome-en-base
  language-pack-gnome-zh-base libpng12-0 libxfont1
  linux-restricted-modules-common xserver-xorg-core
  xserver-xorg-video-intel
The following NEW packages will be automatically installed:
  arj clamav-base clamav-freshclam libarchive-tar-perl libarchive-zip-perl
```

```
  libberkeleydb-perl libclamav3 libcompress-raw-zlib-perl
  …
The following packages have been kept back:
  app-install-data-commercial apt apt-utils apturl avahi-autoipd
  avahi-daemon bind9-host bittorrent bzip2 capplets-data compiz compiz-core
  …
The following NEW packages will be installed:
  arj clamav clamav-base clamav-freshclam libarchive-tar-perl
  libarchive-zip-perl libberkeleydb-perl libclamav3
 …
The following packages will be upgraded:
  perl perl-base perl-modules
3 packages upgraded, 51 newly installed, 0 to remove and 197 not upgraded.
Need to get 27.1MB/27.2MB of archives. After unpacking 39.4MB will be used.
The following packages have unmet dependencies:
  libperl5.8: 依赖: perl-base (= 5.8.8-7ubuntu3) but 5.8.8-7ubuntu3.1 is to be
installed.
Resolving dependencies...
The following actions will resolve these dependencies:

Upgrade the following packages:
libperl5.8 [5.8.8-7ubuntu3 (gutsy, gutsy, now) -> 5.8.8-7ubuntu3.1
(gutsy-updates, gutsy-security)]

Score is 120

Accept this solution? [y/n/q/?]
```

Step 04 输入Y并按回车键接受选择的方案，然后系统将自动完成Postfix和相关软件包的安装，在安装的过程中，也将看到如图23.6所示的提示信息窗口。

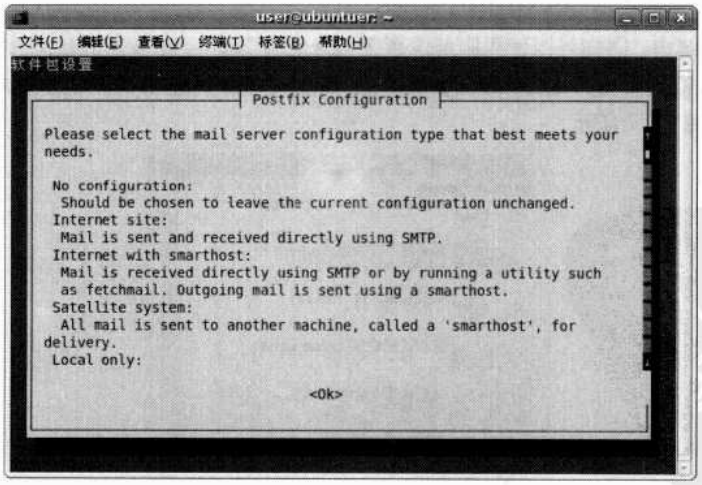


图 23.6 命令行下安装 Postfix 的配置提示

Step 05 使用Tab键选中OK按钮，按回车键进入到Postfix的安装提示界面，需要设置的项跟上面使用新立得软

件管理器安装时的设置项一样，包括邮件服务器配置模型和名称，在这里就不再一一讲述。

23.3.2 Postfix 的目录结构

在进行 Postfix 配置之前，得先了解一下 Postfix 的整体结构，以便将来处理配置文件，所以下面针对 Postfix 这个软件的目录结构进行简单的说明，然后再针对各个设定参数来进行说明。Postfix 的配置文件几乎都在/etc/postfix 中，至于可执行文件则放在/usr/sbin 中。

23.3.3 启动或关闭 Postfix

在命令行管理 Postfix

启动 Postfix

```
root@ubuntuer: ~# /etc/init.d/postfix start
* Starting Postfix Mail Transport Agent postfix          [OK]
```

停止 Postfix

```
root@ubuntuer: ~# /etc/init.d/postfix stop
* Stopping Postfix Mail Transport Agent postfix          [OK]
```

重启 Postfix

```
root@ubuntuer: ~# /etc/init.d/postfix restart
* Stopping Postfix Mail Transport Agent postfix          [OK]
* Starting Postfix Mail Transport Agent postfix          [OK]
```

使用图形界面管理 Postfix

在 Ubuntu 系统中，同样可以使用服务设置工具来管理 Postfix 服务器的状态，执行【系统】|【系统服务】|【服务】命令，打开【服务设置】对话框，点选【邮件代理（postfix）】来启动或关闭 Postfix 邮件服务器，如图 23.7 所示。

图 23.7 使用服务设置工具管理 Postfix 服务

23.4 Postfix 的配置

安装完 Postfix 后，默认情况下，Postfix 服务已经启动了，用户可以使用 mail 命令在本地测试邮件服务器的功能，但为了能接收或发送邮件到网络上的另外一台邮件服务器，这就需要扩展 Postfix 邮件服务器的功能，也就是配置 Postfix。

相对于 Sendmail 的配置文件数，Postfix 的配置文件就简单多了，主要有三个配置文件，这三个配置文件都搁置在/etc/postfix 目录下。

- dynamicmaps.cf：依据服务器类型，配置 Postfix 运行时启动的额外功能的文件。
- main.cf：这就是主要的 Postfix 配置文件，几乎所有的设置参数都是在这个文件中定义。这个配置文件就是一个完整的说明文件了，用户可以参考这个文件的内容设定属于自己的 Postfix 邮件服务器，当然修改过这个文件后，一定记得要重新启动 Postfix，以便设定及时生效。
- master.cf：该文件主要规定了 Postfix 每个程序的运行参数，也是很重要的一个配置文件。不过这个配置文件里面的默认设置相当完备，通常不需要再做上面的修改了。

从上面的讲述看出，一般情况下只需要对 main.cf 这个文件进行修改就已经足够了，下面来讲述 main.cf 中参数的意义，如表 23.1 所示。

表 23.1 Postfix 配置参数描述

参数名称	描述	默认值
smtpd_banner	设置 SMTP 连接建立时显示的标志名称	$myhostname ESMTP $mail_name（Ubuntu）
biff	本地邮件通知服务的开关，用于通知用户有新邮件，但是该通知服务对系统性能有影响而且用户只能登录到邮件服务器主机上才能接收到该通知，所以默认情况下该服务是关闭的	no
append_dot_mydomain	是否在邮件地址后加上邮件服务器域名，一般情况下，现在的 MUA 已经做了这个工作，所以在 MTA 里都是关闭的	no
delay_warning_time	如果邮件因故延迟发送，每 4 个小时就会给发送者发信提示邮件延迟发送。因为考虑到邮件流量的问题，一般情况下这个参数不做设置	4h

（续表）

参数名称	描述	默认值
smtpd_tls_cert_file	如果接收或发送邮件需要 TLS 加密，这个参数用来指定证书的绝对路径	/etc/ssl/certs/ssl-cert-snakeoil.pem
smtpd_tls_key_file	如果接收或发送邮件需要 TLS 加密，这个参数用来指定 SMTP 客户端 RSA 私钥的绝对路径	/etc/ssl/private/ssl-cert-snakeoil.key
smtpd_use_tls	如果远程的 SMTP 服务器支持 TLS 传送，设置 Postfix 邮件服务器是否使用 TLS 进行传输，如果远程 SMTP 服务器不支持 TLS，将以不加密的方式进行传输	yes
smtpd_tls_session_cache_database	设置 SMTP 服务器进行 TLS 传送时，tlsmgr 进程进行缓存的组织方式或路径	Btree:${data_directory}/smtpd_scache
smtp_tls_session_cache_database	设置 SMTP 客户端进行 TLS 传送时，tlsmgr 进程进行缓存的组织方式或路径	Btree:${data_directory}/smtp_scache
myhostname	设置主机名称，需要使用 FQDN	安装过程设置
alias_maps	设置本地邮件别名映射文件的绝对路径	hash:/etc/aliases
alias_database	设置本地邮件别名映射文件的绝对路径。与 alisa_maps 参数设定值可以相同，也可不同，因为这个文件可以通过原有的 newaliases 命令来修改内容	Hash:/etc/aliases
myorigin	设置服务器使用哪个域名来外发邮件，可以支持多个域名	$myhostname
mydestination	设置服务器使用哪个域名来接收邮件	localhost.$mydomain
relayhost	如果需要通过别的邮件服务器进行邮件转发，就在这里指定转发邮件的服务器域名，如果设置为空，则无需转发	无
mynetworks	设置可以转发哪些网络的邮件	
mailbox_size_limit	设置邮箱的最大值，0 表示不做任何限制	0
recipient_delimiter	设置邮件服务器对用户名和地址的分隔符号	+

（续表）

参数名称	描述	默认值
inet_interfaces	设置邮件服务器接收邮件的范围。如果想接收所有网络的邮件，可以设置为 all	NA

23.5 课后练习

1. 什么是邮件服务器？邮件服务器的工作原理是什么？
2. 为什么 Postfix 是 Linux 中应用最广泛的邮件服务器之一？它有哪些特点？
3. 如何在 Ubuntu 中安装 Postfix 服务？
4. 如何启动、停止、重启 Postfix 服务？
5. 说说 Postfix 服务的目录结构，如配置文件目录、运行 bin 目录等。
6. Postfix 的主配置文件是哪个？它的配置包括哪些项？

Chapter 24 SAMBA 服务配置

SAMBA 是 Linux 中特别是服务器版 Linux 中最常用的服务之一。通过 SAMBA 服务，用户可以实现让 Windows 用户访问 Linux 的文件及打印机等共享资源；Linux 也可以通过使用 SAMBA 允许用户在 Linux 下访问 Windows 的文件和打印机。本章将从 SAMBA 的基本认识开始，再介绍 SAMBA 系统安装方法以及一些初始配置，并通过一个实例让大家学习如何测试从 Windows 到 Linux 的连接以及测试从 Linux 到 Windows 的连接。

24.1 SAMBA 简介

SAMBA 最初出现在 1992 年。SAMBA 系统通过利用越来越多的开放源代码软件，获得了丰富多彩的性能，并且变得越来越稳定。随着 SAMBA 系统的不断进化，对于那些正在考虑将其文件和打印解决方案迁移到 Linux 的系统管理员来说，如今它已经成为一个真正的可选项。另外，在 Linux 系统上构建存储解决方案也是一种非常便宜的方法。首先，在机箱中安装一个支持 IDE RAID 卡，再安装 Linux，并启动 SAMBA，然后就可以在自己的网络上安排大量的存储空间，这是一种成本非常低的实现方法。对于那些需要经常做笔记本电脑的备份以及需要进行长期归档工作的企业来说，这种解决方案是非常理想的。

在安装 SAMBA 系统之前，很有必要了解一下 SAMBA 是如何工作的。SAMBA 之所以能够工作，是因为它模仿的是 Windows 内核的文件和打印共享协议，该协议称之为 SMB 协议（Server Message Block ）。SMB 在 Windows 出现之前就已经存在了。该协议可以追溯到 20 世纪 80 年代，它是由英特尔、微软、IBM、施乐以及 3com 等公司联合提出的。虽然在过去的 20 年中，该协议得到了扩展，但是该协议的基本理论仍然是相同的。

微软已经将 SMB 改名为公共因特网文件系统（Common Internet File System，CIFS）。这在一定程度上是由于它想与最初的基于 NetBIOS 的 SMB 保持一定的距离。最初，NetBIOS 是一个伟大的工具，但是渐渐地发现，该工具无法处理在内部网络中连接到计算机上的全部计算机的个数，或者在因特网上无法显示连接到当前计算机上的计算机的个数。

SAMBA 也执行了 SMB(或者叫做 CIFS)的一个版本，这个版本在很大程度上与大多数的 Windows 版本兼容。有时候，微软 SAMBA 系统会出现崩溃，例如在 Windows 2000 的补丁包中，当正常的认证方式被改变时，就会导致 SAMBA 系统的崩溃。唯一的能够让 SAMBA 重新工作的方法是通过注册表将认证方式改回来。尽管存在这些细小的缺陷，虽然这些缺陷在大量集成之后总是会出现的，但是，无论是从 Windows 连接到 Linux 机器还是从 Linux 连接到 Windows 机器，SAMBA 系统对于实现文件和打印服务来说总是很稳定的。

24.2 SAMBA 安装及启动

在 Ubuntu 中，SAMBA 并不会自动安装在系统中，需要用户根据需要进行安装。下面就重点介绍 SAMBA 在 Ubuntu 中的安装，并介绍一下 SAMBA 的启动。

24.2.1 SAMBA 安装

SAMBA 的安装可以通过新立得软件包管理器进行窗口化安装，也可以通过 apt-get 在命令行下进行安装。

使用新立得软件包管理安装

Step 01 通过新立得软件包管理器来进行SAMBA安装时，首先通过执行【系统】|【系统管理】|【新立得软件包管理器】命令打开新立得软件包管理器，查找并选中samba，如图24.1所示。

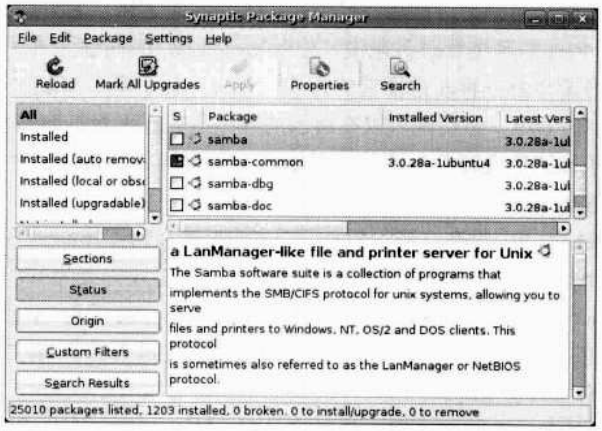


图 24.1 选择 samba 软件包进行安装

Step 02 选中samba复选框时，会弹出如图24.2所示的对话框。

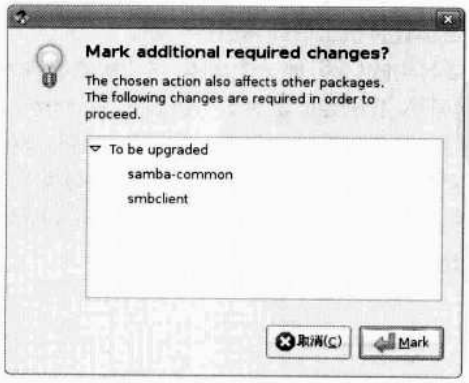


图 24.2 SAMBA 的关联软件包

Step 03 单击Mark按钮，则系统会进行相应的操作，并标明SAMBA及相关组件mark状态，如图24.3所示。

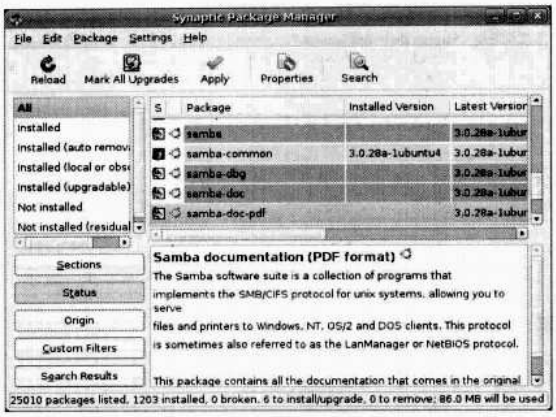


图 24.3 选择相关的 SAMBA 软件包进行安装

Step 04 单击Apply按钮，系统弹出相应的Summary对话框，标明各个软件包的安装状态，是全新安装、升级还是没有改变，如图24.4所示。

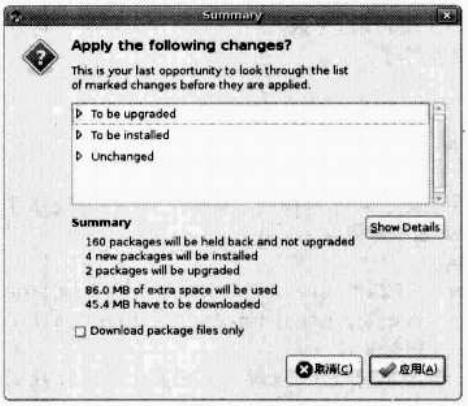


图 24.4 SAMBA 软件安装前的 Summary

Step 05 单击【应用】按钮，系统将自动运行安装，下载进度如图24.5所示。

图 24.5 SAMBA 软件安装进度

Step 06 安装结束后，弹出Changes applied对话框，提示安装后的信息，如图24.6所示。单击【关闭】按钮，结束安装过程。

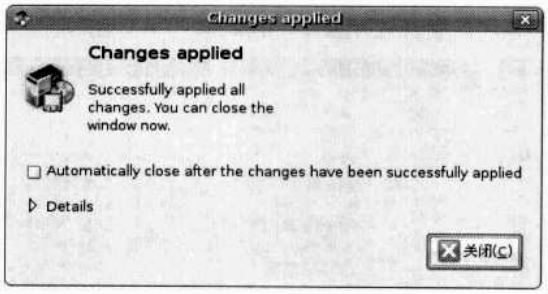


图 24.6 SAMBA 安装结束后的 Changes applied 提示

在命令行下安装 SAMBA

Step 01 通过执行【应用程序】|【附件】|【终端】命令打开命令行。

Step 02 在命令提示行下输入apt-get install samba smbclient smbfs，按回车键，然后系统就会自动安装SAMBA，同时也会安装相应的依赖软件包，具体安装过程如下所示：

```
root@ubuntuer:~# apt-get install samba smbclient smbfs
正在读取软件包列表...完成
正在分析软件包的依赖关系树
Reading state information...完成
smbclient 已经是最新的版本了。
建议安装的软件包：
  openbsd-inetd inet-superserver smbldap-tools
下列【新】软件包将被安装：
  samba smbfs
共升级了 0 个软件包，新安装了 2 个软件包，要卸载 0 个软件包，有 35 个软件未被升级。
有 1 个软件包没有被完全安装或卸载。
需要下载 3933KB 的软件包。
After this operation, 9622kB of additional disk space will be used.
获取：1 http://cn.archive.ubuntu.com hardy-updates/main samba
 3.0.28a-1ubuntu4.4 [3839kB]
获取：2 http://cn.archive.ubuntu.com hardy-updates/main smbfs
 3.0.28a-1ubuntu4.4 [94.0KB]
下载 3933KB，耗时 46s (84.1KB/s)
正在预设定软件包 ...
选中了曾被取消选择的软件包 samba。
(正在读取数据库 ... 系统当前总共安装有 128078 个文件和目录。)
正在解压缩 samba (从 .../samba_3.0.28a-1ubuntu4.4_i386.deb) ...
选中了曾被取消选择的软件包 smbfs。
正在解压缩 smbfs (从 .../smbfs_3.0.28a-1ubuntu4.4_i386.deb) ...
正在设置 mailscanner (4.58.9-2ubuntu1.2) ...
Checking/installing report files ...
hostname: Unknown host
正在设置 samba (3.0.28a-1ubuntu4.4) ...
Generating /etc/default/samba...
......
Importing account for nobody...ok
Importing account for learner...ok
```

```
Importing account for user...ok
Adding group 'sambashare' (GID 130) ...
Done.
Adding user 'learner' to group 'sambashare' ...
正在将用户"learner"加入到"sambashare"组中
Done.
Adding user 'user' to group 'sambashare' ...
正在将用户"user"加入到"sambashare"组中
Done.
*Starting Samba daemons                                    [ OK ]
```

24.2.2 SAMBA 目录结构

成功安装 SAMBA 后，下面讲述一下 SAMBA 的文件和目录结构，以便能知道 SAMBA 各个文件或目录的功用，方便用户找到相应的配置文件。

- /usr/sbin/smbd：SAMBA 守护进程启动程序，用于为 SMB 客户提供文件和打印服务。
- /etc/samba/smb.conf：SAMBA 服务程序的主配置文件。
- /usr/sbin/nmbd：提供 NetBIOS 服务和浏览支持的启动程序。
- /usr/bin/smbclient：SMB 客户程序，在 Linux 平台中实现类似于 FTP 的客户端。
- /usrbin/smbmount：SMB 共享文件系统加载程序。
- /usr/bin/testparm：提供给/etc/samba/smb.conf 配置文件进行语法检测的工具。
- /usr/bin/smbstatus：SMB 服务状态查询工具。
- /usr/bin/smbtar：SMB 服务数据资源备份工具。

24.2.3 SAMBA 服务的启停操作

在安装完 SAMBA 服务器后，SAMBA 服务器已经自动启动了。如果需要更改 SAMBA 的系统设置，那就一定要重新启动 SAMBA 服务器，以便修改能够及时生效。

在命令行下管理 SAMBA 服务器

启动 SAMBA 服务器

```
root@ubuntuer: ~# /etc/init.d/samba start
* Starting Samba daemons              [OK]
```

停止 SAMBA 服务器

```
root@ubuntuer: ~# /etc/init.d/samba stop
* Stopping Samba daemons              [OK]
```

重启 SAMBA 服务器

```
root@ubuntuer: ~# /etc/init.d/samba restart
* Stopping Samba daemons              [OK]
* Starting Samba daemons              [OK]
```

使用图形界面管理 SAMBA 服务器

在 Ubuntu 系统中，同样可以使用服务设置工具来管理 SAMBA 服务器的状态，执行【系统】|【系统服务】|【服务】命令，打开【服务设置】对话框，点选【文件夹共享服务（samba）】来启动或关闭 SAMBA 服务器，如图 24.7 所示。

图 24.7 使用服务设置工具管理 SAMBA 服务器

24.3 SAMBA 配置

上一节已经成功安装了 SAMBA 服务，并启动成功。但是目前仍然无法让 Linux 主机和 Windows 客户端互通共享数据，打开 Windows 客户端桌面上的网上邻居也找不到 Linux 服务器。为什么呢？因为还没有配置 SAMBA。下面就来讲述 SAMBA 的配置。通过 SAMBA 的配置，用户可以实现目录共享、打印共享、目录权限控制等各种设置。在安装完 Ubuntu 后，系统会自动生成 /etc/Samba/smb.conf 配置文件，注释也很详细。下面是 smb.conf 文件的示例：

```
[global]
   workgroup = UEC
   netbios name = UEC-server
   server string = Samba Server
   log file = /usr/local/Samba/var/log.%m
   security = user
   guest account = guest
[gcp docs]
   comment = this is shared gcp docs
   path = /gcp
   guest ok = yes
   read only = no
   browseable = yes
[john docs]
   comment = this is shared john docs
```

```
path = /john
guest ok = yes
read only = no
```

24.3.1 一个简单的示例

在讲述 smb.conf 配置各选项之前，先通过一个简单的例子来体验一下 SAMBA 的设置及共享服务，让读者有一个直观的体会。下面是具体的配置步骤：

Step 01 通过sudo gedit /etc/Samba/smb.conf打开smb.conf文件。将security=user那一行前的注释符；去掉，然后把user改为share，这样可以实现匿名访问。

Step 02 使用sudo /etc/init.d/Samba restart命令重启SAMBA。这样就可以访问文件夹了，但是有个限制，那是该文件夹只有读权限，无法进行写操作。

Step 03 修改文件夹的权限：chmod 777（文件夹名称）。

配置完成，SAMBA 服务就可以提供给各个操作主机访问了，现在上传一个文件夹到共享目录，可看见在 Ubuntu 下该目录的创建者是 nobody。

24.3.2 环境变量说明

在配置 smb.conf 之前，先介绍一下 SAMBA 中的环境变量。我们可以使用"变量"来增加 SAMBA 的使用弹性。每个变量都是以"%"开头，后面再加上一个大写或小写的字母代替不同的意义。在 SAMBA 具体实行时，这些变量都会取代特定的字符串或数值。环境变量的含义具体如下：

- %S：代表共享名；
- %P：代表共享的主目录；
- %u：代表共享的用户名；
- %g：代表用户所在的工作组；
- %U：代表用户名；
- %G：代表当前对话的用户的主工作组；
- %H：代表用户的共享主目录；
- %v：代表 SAMBA 服务器的版本号；
- %h：代表 SAMBA 服务机器的主机名；
- %m：代表客户机 NetBIOS 名称；
- %L：代表服务器 NetBIOS 名称；
- %M：代表客户机的主机名；
- %N：代表 NIS 服务器名；
- %p：代表 NIS 服务的 Home 目录；
- %I：代表客户机的 IP；
- %T：代表系统当前日期和时间。

24.3.3 全局参数设置

在/etc/Samba/smb.conf 中，用户可以根据需要通过配置它相关的参数来实现复杂或简单的访问控制。smb.conf 的格式中有多个段，每段由段名开始，一直到下个段名，每个段名放在方括号中间。其中文件中就有[global]段，这是全局参数的设置部分。它设置了工作组、NetBIOS 名和安全参数 security 等参数。表 24.1 是对全局参数的简单说明。

表 24.1 全局参数描述

参数名称	描述	默认值
workgroup	设置所属工作组，最好使用大写字母，不超过 9 个字符	MSHOME
server string	设置计算机描述，会出现在 Windows 的“网络邻居”中	%h server(Samba，Ubuntu)
host allow	设置允许访问的网络和主机 IP 地址，如果是多个 IP 地址，使用空格分开	无
load printer	允许自动共享打印机	yes
printcap name	设置打印机配置文件的路径	/etc/printcap
printing	设置打印机类型，可选项为 bsd、sysv、plp、lprng、aix、hpux、qnx	bsd
log file	设置服务日志文件的路径	/var/log/samba/log.%m
max log size	设置日志文件的大小，单位是 KB，如果是 0，表示无限制	1000
security	设置 SAMBA 服务的安全级别，有 4 个选项：user，要求用户在访问共享资源前提供用户名和密码；share，任何用户都可以访问共享资源而且不需要提供用户名和密码；server，要求用户提供用户名和密码到域名服务器；domain，网络中存在一台 Windows 主域控制器，SAMBA 用户提供用户名和密码到该机器进行验证	user
encrypt password	设置是否对密码进行加密	true
smb password file	设置密码的存放位置	无
wins support	设置是否支持 Windows WINS 服务，WINS 协议能把机器名转换为 IP	yes
invalid user	设置非法用户	root

24.3.4 共享资源参数设置

除了[global]段外，所有的段都可以看作是一个共享资源，段名是该共享资源的名字，而段里的参数就是共享资源的属性。表 24.2 是对共享资源参数的简单说明。

表 24.2 共享参数描述

参数名称	描述	默认值
comment	设置客户端用户界面显示的共享确认字符串	Home Directories
browseable	设置是否允许在浏览器中显示共享	no
writeable	设置共享资源是否可写	no
create mask	设置新建目录属性	0600
directory mask	设置目录属性	0700
valid users	设置能使用该资源的用户或用户组	%S
path	设置共享路径	
printable	设置该打印机资源是否可用	yes
path	设置通过 SAMBA 进行打印请求的时候，使用的脱机打印目录的路径	/var/spool/samba
create mode	设置所有文件生成时具有的用户所有权限	0700

24.4 SAMBA 配置完全实例

接下来讲述一个具体的添加并共享文件夹的方法。

Step 01 在/etc/samba/smb.cof文件末尾添加下面的内容。

```
[Public]
path = /home/bruce/public
available = yes
browseable = yes
public = yes
writable = no
```

其中第二行为共享文件路径，根据实际情况更改。

Step 02 重新启动SAMBA服务。这时共享目录就设置成功了，可以通过客户端访问/home/bruce/public 共享目录。

另外也可以在 Ubuntu 桌面环境下，通过图形化工具来进行共享文件的设置。

Step 01 通过执行【系统】|【系统管理员】|【共享文件】命令打开Shared Folders对话框，如图24.8所示。

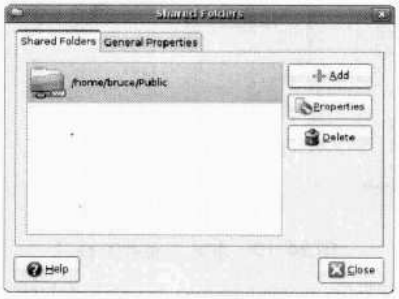


图 24.8 共享文件设置

Step 02 单击Add按钮，接着进行相应的设置。设置成功后，单击close按钮关闭窗口即可。

Step 03 设置结束后，先在本地使用testparm命令检查是否成功添加共享资源。

```
bruce@ubunuter:~$ testparm
    Load smb config files from /etc/Samba/smb.conf
    Processing section "[printers]"
    Processing section "[print$]"
    Processing section "[Public]"
    Loaded services file OK.
    Server role: ROLE_STANDALONE
    Press enter to see a dump of your service definitions
    ...
     [Public]
          path = /home/bruce/Public
          guest ok = Yes
```

在本地 Ubuntu 下，用户可以通过窗口查看是否共享成功，如图 24.9 所示。

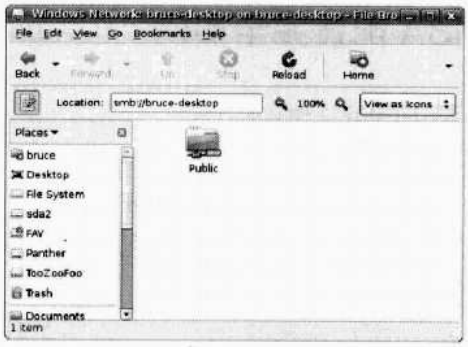


图 24.9 使用文件浏览器查看共享资源

当然可以通过局域网访问该共享资源，如图 24.10 所示。

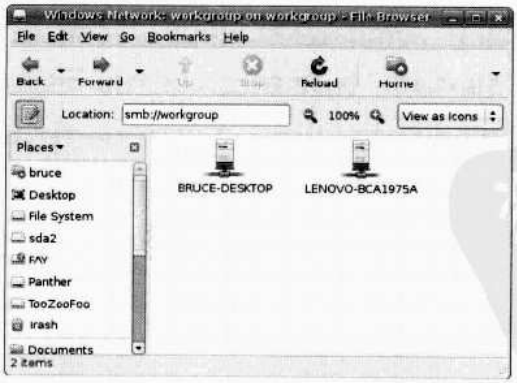


图 24.10 本地查看共享资源

除此之外，在命令行也可以使用 SAMBA Client 查看共享资源。

```
bruce@ubuntuer:~$ smbclient //localhost/Public
```

```
Password:
Domain=[BRUCE-DESKTOP] OS=[Unix] Server=[Samba 3.0.26a]
smb: \> ls
.         D     0 Fri May 23 22:34:50 2008
..        D     0 Tue May 21 18:45:13 2008
groovy.vim  A    21979 Sun Apr 20 15:11:36 2008 45832 blocks of size 524288.8497
blocks available
smb: \>
```

另外，也可以通过一个局域网中的 Windows 系统，查看共享中是否有相应的共享资源，如图 24.11 所示。

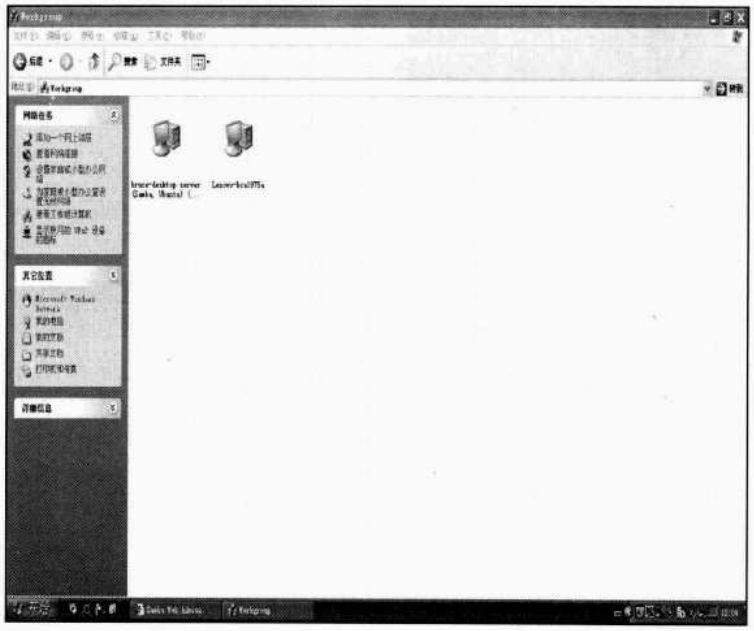


图 24.11　Windows 下通过局域网查看共享资源

24.5　SWAT 工具

SWAT（SAMBA Web Administration Tool，SAMBA Web 管理工具）是 SAMBA 开发团队为便于用户进行 SAMBA 相应设置开发的一个以浏览器为基础的可视化管理工具。

通过 SAMBA SWAT 可以实现上面通过修改 smb.conf 所实现的所有配置功能。另外，本节也顺带简述一下 SAMBA 客户端的安装及应用。

Step 01　安装SWAT。

因为SWAT依赖xinetd才能正常工作，所以在安装SWAT之前，需要先安装 xinetd。在命令行下输入：apt-get install xinetd。具体操作如图24.12所示。

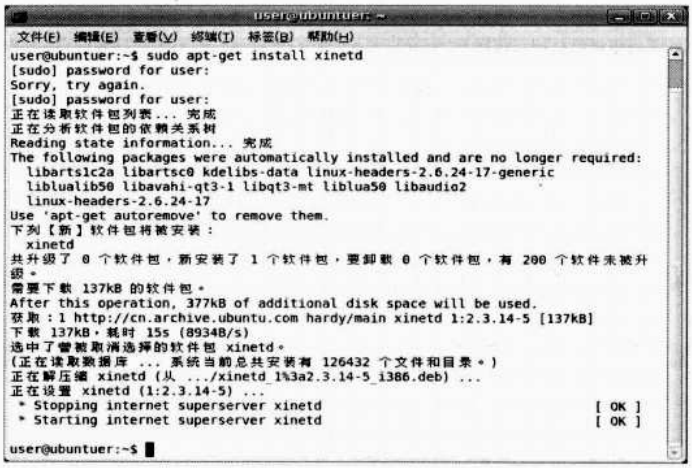


图 24.12 通过 apt-get 安装 xinetd

Step 02 安装完xinetd后，接着安装SWAT。在终端命令行下，运行命令：sudo apt-get Samba Samba-common smbfs smbclient swat。具体操作如图24.13所示。

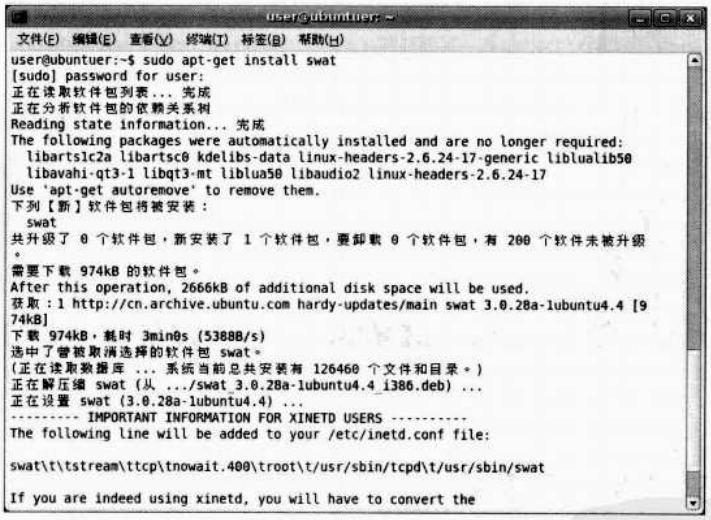


图 24.13 通过 apt-get 安装 SWAT

Step 03 修改inetd.conf。

将 ## swat stream tcp nowait.400 root /usr/sbin/tcpd \ /usr/sbin/swat

这一行的注解去掉，以便SWAT能够运行起来。

Step 04 新建SWAT文件 sudo vi /etc/xinetd.d/swat，内容如下：

```
# description: SAMBA SWAT
    service swat
    {
    disable = no
```

```
    socket_type = stream
    protocol = tcp
    #should use a more limited user here
    user = root
    wait = no
    server = /usr/sbin/swat
    }
```

Step 05 让xinetd重新读取配置，并检查SWAT是否正常运行。

```
user@ubuntuer:~$ udo dpkg-reconfigure xinetd

user@ubuntuer:~$ netstat -antp
(Not all processes could be identified, non-owned process info
 will not be shown, you would have to be root to see it all.)
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address  Foreign Address  State  PID/Program name
tcp        0      0 0.0.0.0:901    0.0.0.0:*              LISTEN-
tcp        0      0 0.0.0.0:139    0.0.0.0:*              LISTEN-
tcp        0      0 127.0.0.1:631  0.0.0.0:*              LISTEN-
tcp        0      0 0.0.0.0:445    0.0.0.0:*              LISTEN-
user@ubuntuer:~$
```

查看 swat 是否运行

如果出现 tcp 0 0 *:swat *:* LISTEN，表示 swat 配置运行成功，正在监听。

Step 06 在浏览器上输入http://localhost:901，然后按回车键访问该地址，显示如图24.14所示的窗口。

图 24.14 以 Web 方式使用 SWAT

Step 07 按照提示输入用户名和密码，按回车键确认，即可进入SWAT操作界面了，如图24.15所示。

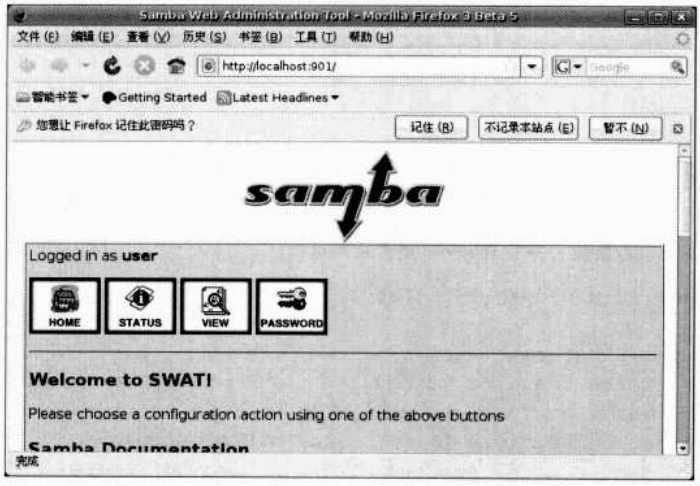


图 24.15 SWAT 首页

从上图中知道，在 SWAT 中，包括 home、status、view、password 等标签按钮，在这里可以通过这些标签按钮进行任何 SAMBA 相关的设置。

24.6 课后练习

1. 什么是 SAMBA？谈谈 SAMBA 协议的起源。
2. 如何在 Ubuntu 中安装 SAMBA 服务？
3. 如何启动、停止、重启 SAMBA 服务？
4. 说说 SAMBA 服务器的目录结构，如配置文件目录、运行 bin 目录等。
5. SAMBA 的主配置文件是哪个？它的配置包括哪些项？
6. 请尝试配置一个简单的 SAMBA 应用实例。

Chapter 25 DNS 服务器——BIND

人的大脑对于数字毕竟不如文字来得敏感，而计算机世界里只认识 0/1 而已。为了将两者绑定在一起，于是就有了主机名称与 IP 的对应，而这个对应的协议就是 DNS 了。在这一章里，将重点介绍 DNS 的由来，以及如何在 Ubuntu 下安装和设置 DNS 服务器，也就是 Ubuntu 下默认的 BIND 服务。希望通过这章的学习，读者能更深入了解 Ubuntu 的 DNS 服务和网络访问的知识。

25.1 DNS 基础

DNS（Domain Name System，域名系统）是因特网的一项核心服务，它作为可以将域名和 IP 地址相互映射的一个分布式数据库，能够使人们更方便地访问互联网，而不用记住能够被机器直接读取的 IP 地址。

域名系统作为一个分布式数据库，每个主机在本地负责控制整个分布式数据库的部分段，每一段中的数据通过客户服务器模式在整个网络上均可存取，通过采用复制技术和缓存技术使得整个数据库可靠的同时，又拥有良好的性能。

域名服务器包含数据库的部分段的信息，并可提供被称之为解析器的客户来访问。DNS 的数据库结构形成一个倒立的树状结构，根的域名用空字符串“”来表示，但在文本中用“.”来表示。树的每一个节点都表示整个分布式数据库中的一个分区（域），每个域可再进一步划分成子分区（域），每个域都有一个标签（Label），标明它与父域的关系。域也有一个域名（Domain Name），给出它在整个分布式数据库中的位置。在 DNS 中，域名全称是一个从该域到根的标签序列，以“.”分隔这些标签。该标签最多可包含 63 个字符。树中每一节点的完整域名为从该节点到根之间路径上的标签序列。

如果根域在节点的域名中出现，该域名看起来就像以点结尾（实际上是以点和空标签作为结尾）。这些以点结尾的域名被称为绝对域名（Absolute Domain Name），不以点结尾的域名被称为相对域名。域（Domain）即为树状域名空间中的一棵子树，域的域名同该子树根节点的域名一样。也就是说，域的域名就是该域中最高层节点的域名。

在 DNS 中，每个域分别由不同的组织进行管理。每个组织都可以将它的域再分成一定数量的子域并将这些子域委托给其他组织进行管理，域既能包括主机又能包括其他域（它的子域）。域名被用作 DNS 数据库中的索引。子域中任何域名被认为是域的一部分。事实上，主机即域，域名仅是 DNS 数据库中的索引，“主机”可由指向相关主机信息的域名来索引，域包含所有其域名在该域的主机。

在域名树中，叶节点的域通常代表主机，它们的域名可指向网络地址、硬件信息和邮件路由信息。在树内的节点，其域名既可命名一台主机，也可指向有关该域的子孙或子域的结构信息，在域名树中的内部域名并不受唯一性限制，它们既可表示它们所对应的域，又可代表网络中某台特定的主机。例如，sun.com 既是 sun 的域，又是在 sun 和 Internet 间转发信件的邮件服务器的域名。

网络上的每一台主机都有一个域名，域名给出有关主机的信息，该信息中包含 IP 地址，MAIL 路由信息等，主机也可以有一个或多个域名别名。

域名

判断域是否为另一域的子域的简单方法是比较它们的域名，子域名以其父域名结尾。设计域名系统的一个主要目的是让管理分散化，这是通过代理来实现的。管理域的组织将该域划分成子域，每一个子域可以由其他组织代理，这意味着那些代理组织负责维护在该子域的所有数据。它们可以自由地改变数据，甚至可以将它们管理的子域再划分成更多的子域并将它们再分配。父域中仅包含指向这些子域的指针，因而引用对子域的查询。

域名服务器

存储有关域名空间信息的程序被称为域名服务器（Name Server）。通常，域名服务器拥有部分域名空间（称之为区 zone）的完整信息，域名服务器可以拥有多个区的授权。

区与域的关系为：

区包含了域中除了代理给别处的子域外所含有的所有域名和数据。如果域的子域没有被代理出去，则该区包含该子域名和子域中的数据。

DNS 定义了两类域名服务器：Primary Master(PM) 和 Secondary Master(SM)。PM 域名服务器，从它所运行的主机上的文件获得它所负责的区的数据；SM 域名服务器则是从其他具有该区授权的域名服务器上获得它的区的数据，SM 域名服务器会定期查询 PM 域名服务器以保证区数据为最新版本。

一般情况下，最好设立一台 PM 域名服务器和若干台 SM 域名服务器，这样可以分担负载，以及确保区中所有主机都有比较靠近的域名服务器，以便提高访问速度。

解析器

运行在主机上并需要域名空间信息的解析器（Resolver），在 BIND 中解析器仅仅是一组库程序，并编译进像 telnet 和 ftp 这样的程序中，它们并非独立的进程。解析器所做的工作为汇集查询，发送查询并等待应答，未得到应答时重发查询。

地址到域名的映射

在域名空间的数据是通过域名来进行索引的，找到一个给定域名的地址相对容易，但是要找到映射给一定地址的域名就要在树上的每一个域名空间做穷尽搜索。如果这样的话，效率将相当低，为了解决这个问题，创建一个以地址为索引的域名空间。这部分域名空间被称为 in-addr.arpa 域。

缓存与生存期

域名服务器在处理递归查询时，可能要进行多次查询才能得到信息，在这个过程中，域名服务器可以获得很多有关域名空间的信息，域名服务器将所有这些信息都缓存起来以加速以后的查询。除了加速查询外，缓存还使得用户不必再次查询根域名服务器，这样可不必过分依赖根域名服务器而大大减轻根域名服务器的负载。

生存期（TTL）为所容许的域名服务器对数据缓存的时间长度，一旦生存期到了，域名服务器必须丢弃缓存数据并从授权的域名服务器中重新获取新的数据，这样可以确保域数据在整个网络上

的一致性。

DNS 在 RFC 1034 和 RFC 1035 中有详细说明，并在另外若干个 RFC 中作了更新。DNS 是一个复杂的系统，这里只讨论其操作的关键方面，感兴趣的读者可以参见其协议文档。

25.2 BIND 简介

BIND（Berkeley Internet Name Domain），最初是由加州大学柏克莱分校所发展出来的 BSD Unix 中的一部分，目前则由 ISC 组织来负责维护与发展。

BIND 域名服务器可以作为区域的管理服务器、从属服务器或者缓存服务器。虽然主服务器服务和缓存/递归服务器在逻辑上是不同的，但也可以放在不同的服务器上运行。一个主服务器可以禁止递归来提高可靠性和安全性。不作为任何区域的主服务器，只为本地客户提供递归查询则不需要暴露在互联网上，因而可以放在防火墙后面。

BIND 是一个被广泛使用的 DNS 服务器软件，它提供了强大及稳定的域名服务，因此有近九成的 DNS 服务器主机都使用 BIND，目前最新的版本为 BIND 9。

25.3 搭建 BIND 服务器

通过上面的学习，读者或许已经对域名服务器和 BIND 有了一个整体的认识，接下来将讲述如何在 Ubuntu Linux 下搭建 BIND 服务器，包括安装 BIND，BIND 的目录结构以及如何启动和关闭 BIND 服务。

25.3.1 BIND 9 的安装

在 Ubuntu Linux 下安装 BIND 9 软件主要有两种方式，一种就是使用新立得软件包管理器安装，另外一种就是在命令行下输入命令 apt-get 来安装。读者可以选用任何一种方式来安装，但如果安装的是服务器版本而没有 GUI 功能的话，那么就只能通过第二种方式通过命令行来安装，当然如果安装的是桌面版本的系统，那么这两种方式都能为你所用了。

使用新立得软件包管理安装 BIND 9

Step 01 通过【系统】|【系统管理】|【新立得软件包管理器】命令打开程序。

Step 02 在新立得软件包管理器窗口中，单击工具栏中的Search按钮，在弹出的小窗口中输入关键字bind9进行查找，如图25.1所示。

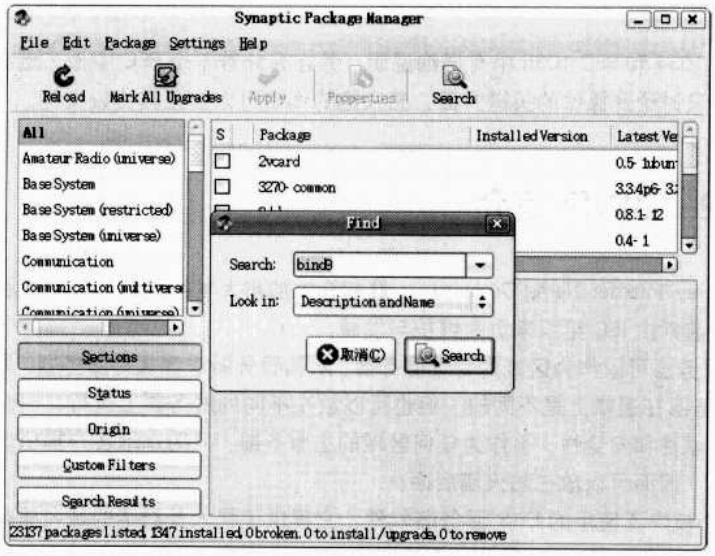


图 25.1 查找 BIND 9 软件包

Step 03 在查询结果中，选择bind9、bind9—doc软件包进行安装，选择bind9—host进行升级，如图25.2所示。

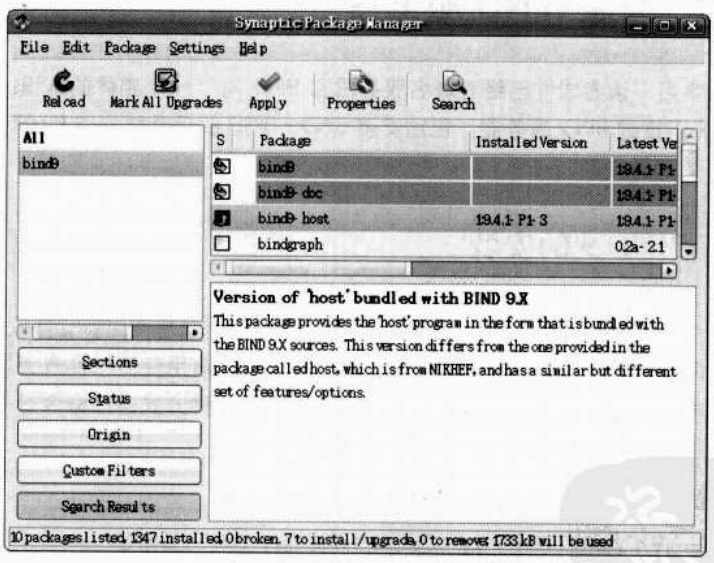


图 25.2 选中 bind9、bind9—doc 和 bind9—host 软件包进行安装或升级

Step 04 选择完毕后，单击工具栏上的Apply按钮，开始安装BIND 9软件，完成安装后，就可以退出新立得软件包管理器。

在命令行下安装 BIND 9

Step 01 通过执行【应用程序】|【附件】|【终端】命令打开命令行。

Step 02 安装BIND 9服务器及其关联包，在命令行下输入下面的命令：

```
apt-get install bind9
```

Step 03 按回车键，系统就会自动安装BIND 9，安装过程如下所示：

```
root@ubuntuer:~# apt-get install bind9
正在读取软件包列表...完成
正在分析软件包的依赖关系树
Reading state information...完成
建议安装的软件包：
  bind9-doc resolvconf
下列【新】软件包将被安装：
  bind9
共升级了 0 个软件包，新安装了 1 个软件包，要卸载 0 个软件包，有 35 个软件未被升级。
有 1 个软件包没有被完全安装或卸载。
需要下载 268KB 的软件包。
After this operation, 762KB of additional disk space will be used.
获取：1 http://cn.archive.ubuntu.com hardy-updates/main bind9
1:9.4.2-10ubuntu0.1 [268KB]
下载 268KB，耗时 8s (32.1KB/s)
选中了曾被取消选择的软件包 bind9。
(正在读取数据库 ... 系统当前总共安装有 127986 个文件和目录。)
正在解压缩 bind9 (从 .../bind9_1%3a9.4.2-10ubuntu0.1_i386.deb) ...
正在设置 mailscanner (4.58.9-2ubuntu1.2) ...
Checking/installing report files ...

hostname：Unknown host
dpkg：处理 mailscanner (--configure)时出错：
 子进程 post-installation script 返回了错误号 1
正在设置 bind9 (1:9.4.2-10ubuntu0.1) ...
Adding group 'bind' (GID 129) ...
Done.
Adding system user 'bind' (UID 115) ...
Adding new user 'bind' (UID 115) with group 'bind' ...
Not creating home directory '/var/cache/bind'.
wrote key file "/etc/bind/rndc.key"
Reloading AppArmor profiles: done.
 * Starting domain name service... bind                          [ OK ]
```

从安装的过程看，BIND 9 安装结束后，自动启动服务。

25.3.2 BIND 9 目录结构

BIND 9 服务的主要配置文件是 named.conf，该文件位于 Ubuntu Linux 的/etc/bind 目录下。named.conf 文件下还引入了不同的配置文件，形成了一个具有层次关系的配置结构，包括：

- /etc/bind/db.0：广播地址 0.*的反解。
- /etc/bind/db.127：localhost 反向区文件，用于将本地回送 IP 地址(127.0.0.1)转换为名字 localhost。
- /etc/bind/db.255：广播地址 255.*的反解。

- /etc/bind/db.local：localhost 正向区文件，用于将名字 localhost 转换为本地回送 IP 地址（127.0.0.1）。
- /etc/bind/db.root：根服务器指向文件，由 Internet NIC 创建和维护，无需修改，但是需要定期更新。
- /etc/bind/named.conf.local：分区和 IP 地址的映射信息，是主配置文件 named.conf 的补充。
- /etc/bind/named.conf.options：定义了本地域名服务配置的基本选项。

25.3.3 启动或关闭 BIND 9

在命令行下管理 BIND 9

启动 BIND 9

```
root@ubuntuer: ~# /etc/init.d/bind9 start
* Starting domain name service... bind          [OK]
```

停止 BIND 9

```
root@ubuntuer: ~# /etc/init.d/bind9 stop
* Stopping domain name service... bind          [OK]
```

重启 BIND 9

```
root@ubuntuer: ~# /etc/init.d/bind9 restart
* Stopping domain name service... bind          [OK]
* Starting domain name service... bind          [OK]
```

使用图形界面管理 BIND 9

在 Ubuntu 系统中，同样可以使用服务设置工具来管理 BIND 服务的状态，执行【系统】|【系统服务】|【服务】命令，打开【服务设置】对话框，通过点选【多播 DNS 服务发现（avahi-daemon）】来启动或关闭 BIND 服务。如图 25.3 所示。

图 25.3 使用服务设置工具管理 DNS 服务

25.4 BIND9 的配置

BIND9 配置与 BIND8.X 大致类似，尽管如此，也存在一些不同的地方，如视图（View）。

BIND9 对 BIND8.X 配置文件只做了一小部分变动，尽管应该做更复杂的检查以确定执行 BIND9 中的新功能是否更有效率。表 25.1 是 BIND 配置文件的元素说明。

表 25.1 BIND 配置文件的元素

元素名称	描述
acl_name	通过 acl 语句来定义访问控制列表名
address_match_list	一个或多个 ip_addr，ip_prefix，key_id 或者 acl_name 的列表
domain_name	用引号扩起来的 DNS 名，例如 my.test.domain
dotted_decimal	一个或多个整数，值的范围为 0~255，用点（.）分割（如 123.45.67 或 89.123.45.67）
ip4_addr	十进制数字，分 4 段的 IPV4 地址
ip6_addr	一个 IP6 地址，如 fe80::200:f8ff:fe01:9742
ip_addr	一个 IP4 或者 IP6 地址
ip_port	一个 IP 端口号，值限制从 0~65535，超级用户的进程使用小于 1024 的端口。在一些情况下星号（*）作为一个占位符选择一个随机高位端口
ip_prefix	表示一段网络。即在 ip_addr 后面跟/和网络掩码的位数。IP 地址是 0 的可以忽略。例如，127/8 是网络 127.0.0.0，掩码是 127.0.0.0；1.2.3.0/28 是网络 1.2.3.0，掩码是 255.255.255.240
key_id	共享密钥的名字，一般用 doamin_name，用于安全传输
key_list	一个或多个 key_id 的列表，以分号来分割和结束
number	一个非负 32 位无符号整数（包括从 0 到 4 294 967 295 的数）。它的数值可以进一步在使用时由上下文限制
path_name	一个用作路径名的用引号引起来的字符串，如 zones/master/my.test.domain
size_spec	为一个数字、unlimited 或是 default。unlimited，则说明可以无限使用或是使用最大的值；default，则说明服务器启动的时候会使用有限的值；如果是数字的话，可以带单位，如，K 或 k 表示 kilobytes(千字节)，M 或 m 表示 megabytes(兆字节)，G 或 g 表示 gigabytes(十亿字节)，大小分别是 1024，1024×1024 和 1024×1024×1024
yes_or_no	YES 或 NO。TRUE 和 FALSE 也可以，也就是数字 1 和 0
dialup_option	可以是 Yes、no、notify、notify-passive、refresh、passive 之一。当在域中使用时，notify-passive、refresh、passive 限制在子域和叶子节点中使用

下面来分析一个比较基础的配置文件（配置文件采用和 C 语言相同的注释符号）。

日志的设置

```
logging {
channel default_syslog { syslog local2; severity error; };
channel audit_log { file "/var/log/named.log"; severity error;
       print-time yes; };
category default { default_syslog; };
category general { default_syslog; };
```

```
    category security { audit_log; default_syslog; };
    category config { default_syslog; };
    category resolver { audit_log; };
    category xfer-in { audit_log; };
    category xfer-out { audit_log; };
    category notify { audit_log; };
    category client { audit_log; };
    category network { audit_log; };
    category update { audit_log; };
    category queries { audit_log; };
    category lame-servers { audit_log; };
    };
```

file "/var/log/named.log"这个设置指定了日志文件的位置，要正常启动 named，必须保证这一文件是存在的，并且 named 进程对它有读写权限。

基本配置项

```
options {
    directory "/etc/namedb";

    listen-on-v6 { any; };

    // If you've got a DNS server around at your upstream provider, enter
    // its IP address here, and enable the line below.  This will make you
    // benefit from its cache, thus reduce overall DNS traffic in the Internet.

    forwarders {
    your.upper.DNS.address;
    };

    /*
    * If there is a firewall between you and nameservers you want
    * to talk to, you might need to uncomment the query-source
    * directive below.  Previous versions of BIND always asked
    * questions using port 53, but BIND 8.1 uses an unprivileged
    * port by default.
    */
    // query-source address * port 53;

    /*
    * If running in a sandbox, you may have to specify a different
    * location for the dumpfile.
    */
    dump-file "/etc/named_dump.db";
    };
```

下面是一些基本的配置项。

- directory "/etc/namedb"：指定域名解析等文件的存放目录。
- listen-on-v6 { any; }：表示支持 ipv6 的请求。

- forwarders：指定前向 DNS，当本机无法解析域名时，就会被转发至前向 DNS 进行解析。
- dump-file "/etc/named_dump.db"：指定 named_dump.db 文件的位置。

线索域和回环域

```
// Setting up secondaries is way easier and the rough picture for this
    // is explained below.
    //
    // If you enable a local name server, don't forget to enter 127.0.0.1
    // into your /etc/resolv.conf so this server will be queried first.
    // Also, make sure to enable it in /etc/rc.conf.

    zone "." {
    type hint;
    file "named.root";
    };

    zone "0.0.127.IN-ADDR.ARPA" {
    type master;
    file "localhost.rev";
    };
```

指定线索域和本地回环域，这部分一般都是使用模板 file "mamed.root" 进行配置。

file "named.root"：指定该域的解析文件，其目录由 options 中的 directory "/etc/namedb" 指定。

自定义域

```
     zone "test.com" {
    type master;
    file "zone.test ";
};
//设定 test.com 域
    // file "zone.test "指定其解析文件为 zong.test，目录为 options 中设定的目录，
       本例中为/etc/named
// type master 指明该域主要由本机解析

zone "0.168.192.in-addr.arpa" {
    type master;
    file "zone. test.rev";
};
//指定 ipv4 地址逆向解析
    //type master 指明该域主要由本机解析
    //file "zone.test.rev "指定其解析文件为 zong.test.rev，目录为 options 中设定的
      目录，本例中为/etc/named

zone "4.0.0.f.0.5.2.0.1.0.0.2.IP6.ARPA" {
    type master;
    allow-transfer { any;};
    allow-query { any; };
    file "ipv6.rev";
```

```
    };
    //指定 ipv4 地址逆向解析
    //type master 指明该域主要由本机解析
    //file " ipv6.rev "指定其解析文件为 ipv6.rev，目录为 options 中设定的目录，
      本例中为/etc/named

zone "lowerlevelzone.test.com" {
    type slave;
    masters {
    192.168.1.1;
    };
};
//设定 lowerlevelzone.test.com 域
//type slave 指明该域主要由低一级的域名服务器解析
```

25.5 课后练习

1. 什么是域名系统（DNS）？域名系统中存有哪些信息？
2. Linux 中最常用的 DNS 服务器是什么？
3. 如何在 Ubuntu 中安装 BIND 服务器？如何启动、停止、重启 BIND 服务？
4. BIND 服务的主配置文件是哪个？它的配置包括哪些项？
5. 请尝试配置一个简单的 BIND 应用实例。