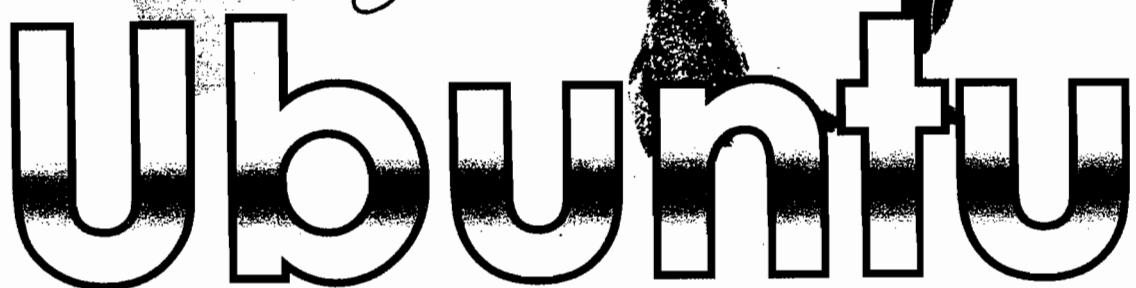




Ubuntu之父寄语：

*Follow your dreams!  
Marty Suh*



# 权威指南

邢国庆 张广利 邹浪 编著

中国Ubuntu实验室 审校

人民邮电出版社  
北京

## 图书在版编目 (C I P) 数据

Ubuntu权威指南 / 邢国庆, 张广利, 邹浪编著. —  
北京 : 人民邮电出版社, 2010.1  
ISBN 978-7-115-21267-2

I. ①U… II. ①邢… ②张… ③邹… III. ①  
Linux操作系统—指南 IV. ①TP316. 89-62

中国版本图书馆CIP数据核字(2009)第187163号

## 内 容 提 要

本书首先介绍 Ubuntu Linux 系统的安装与 GNOME 桌面环境, 然后从基本命令行入手, 由浅入深, 逐步阐述 Linux 系统的基本概念与原理, 同时给出大量的应用实例。在此基础上, 对 Linux 系统的各种文件系统、Shell 编程、进程管理、软件管理、磁盘空间管理、用户管理、系统启动过程、作业调度与系统日志、TCP/IP 网络管理与应用、OpenSSH、DNS、NFS、Samba、Apache 服务器与 MySQL 数据库等方面进行了深入的讨论。本书内容丰富, 语言流畅, 涵盖了 Linux 系统的主要课题, 可以作为学习、使用、管理与维护 Ubuntu Linux 系统的工具书, 也可作为学习 Linux 操作系统的主要参考书。

## Ubuntu 权威指南

- 
- ◆ 编 著 邢国庆 张广利 邹 浪
  - 审 校 中国 Ubuntu 实验室
  - 责任编辑 黄 紊
  - ◆ 人民邮电出版社出版发行 北京市崇文区夕照寺街 14 号
  - 邮编 100061 电子函件 315@ptpress.com.cn
  - 网址 <http://www.ptpress.com.cn>
  - 北京铭成印刷有限公司印刷
  - ◆ 开本: 787×1092 1/16
  - 印张: 40.75
  - 字数: 1021 千字 2010 年 1 月第 1 版
  - 印数: 1~3500 册 2010 年 1 月北京第 1 次印刷

ISBN 978-7-115-21267-2

定价: 69.00 元

读者服务热线: (010) 67132692 印装质量热线: (010) 67129223

反盗版热线: (010) 67171154

# 前　　言

自 1991 年 Linux 诞生以来，大批人士加入学习、使用、开发、交流以及研究 Linux 系统的队伍。这使得 Linux 系统流派纷呈，不同品牌的 Linux 系统各领风骚，其中比较著名的就有 Red Hat、Fedora、Debian、Ubuntu、SUSE、Mandriva、Slackware 及 Gentoo 等。系统的不断升级，及时反映了 IT 行业的最新研究成果与开发技术，使得 Linux 成为最流行的操作系统之一。在各种发行品牌的 Linux 系统中，Ubuntu Linux 系统尤其引人注目，其发展与风靡速度非常的快。

实际上，在不同的发行品牌或不同版本的 Linux 系统中，Linux 系统采用的内核与实用程序基本上都是一样的，故不同的 Linux 系统之间并没有本质的差别，而是具有很多共性。因此，对于立志学习 Linux 的人来讲，选用任何一种 Linux 系统即可。

与 Linux 系统本身拥有的强大功能相比，其桌面环境提供的功能毕竟有限，只能略尽部分辅助之力。在 Linux 系统中，桌面环境提供的任何系统工具实际上都是基于系统的基本命令实现的，不管桌面环境怎样丰富和发展，都离不开命令行的支持；Linux 系统的强大功能完全表现在命令行中，体现在命令的充分发挥与灵活运用方面。这也说明了许多专业人员为什么仍然喜欢使用命令行而不是桌面环境访问 Linux 系统。

因此，除了采用一整章的篇幅全面介绍 GNOME 桌面环境，使读者能够快速领略 Linux 系统的风范，激发学习 Linux 系统的兴趣之外，本书力图以命令行界面为主，从最基本的命令入手，由浅入深，逐步阐述 Linux 系统的基本概念与工作原理；同时给出大量的应用实例，使读者能够通过命令行访问 Linux 系统，深入了解、快速掌握 Linux 系统。此外，为了便于初学者快速入门，后续各章中也辅以必要的桌面工具的使用说明。

本书以 Ubuntu 8.04 和 Ubuntu 8.10 为基准，涵盖了 Linux 系统的主要课题，对 Linux 系统的文件系统、Shell 编程、进程管理、软件管理、磁盘空间管理、用户管理、系统启动过程、作业调度与系统日志、TCP/IP 网络管理与应用、OpenSSH、DNS、NFS、Samba、Apache 服务器与 MySQL 数据库等方面进行了深入的讨论。

在本书的例子中，需要用户输入的命令均以加黑的形式给出。其中，命令提示符为注释符“#”或加 sudo 前缀者表示只有超级用户才能使用的命令，命令提示符为美元符号“\$”者表示普通用户可以使用的命令。此外，为了保持书写的整洁，系统的命令提示符仅采用简单的美元符号“\$”或注释符“#”，省略了其他提示信息。

尽管 Linux 系统的发展势头如日中天，但其文档做得并不好，随机手册也不能令人完全满意，有些命令的说明非常简略，这也是作者编写本书的原因之一，希望能够为读者提供一定的帮助。

本书是作者学习和使用 Linux 系统的一点经验与体会的总结，如能对读者在学习 Linux 操作系统时有所裨益，将是作者莫大的荣幸。由于时间仓促，且限于作者的水平和能力，本书如有不当甚至谬误之处，恳请广大读者给予批评指正（gqxing@gmail.com）。



在本书的写作过程中，从写作宗旨的确定，到章节内容的安排，都得到了人民邮电出版社及编辑的热情鼓励与全力帮助。杨敏敏、庞俊华、王琳、陈逸飞、吕序效、王文睿、邢利荣、翟翔、张广利、邹浪、陈智建、常勇、朱朝辉、王芳、王奇伟、孙伟、仇鹏涛、赵东江、黄辰、曾伟玲、刘琪、李宗玉、梁志强、袁伟以及我的家人邸静与邢梦可也参与了本书的编写，在此一并表示感谢。

编者

2009年10月

# 目 录

<b>第1章 系统概述与安装</b>	1		
1.1 Linux的兴起与发展	2	2.3.6 游戏	36
1.2 充分利用网上资源	3	2.3.7 添加/删除软件	36
1.2.1 Ubuntu官方网站	3	2.4 位置菜单	36
1.2.2 GNU网站	4	2.4.1 主文件夹	38
1.2.3 Linux文档项目网站	4	2.4.2 桌面、文档等	39
1.2.4 网上求助	5	2.4.3 计算机	39
1.3 随时查询随机文档	6	2.4.4 CD/DVD刻录机	39
1.3.1 使用“-help”选项查询命令的简单说明	6	2.4.5 磁盘分区	40
1.3.2 使用man命令联机查询系统参考手册	6	2.4.6 网络	40
1.3.3 使用info命令查询命令的相关信息	8	2.4.7 连接到服务器	41
1.4 安装Ubuntu Linux系统	9	2.4.8 搜索文件	41
1.4.1 安装前的准备	10	2.4.9 最近的文档	42
1.4.2 安装Ubuntu Linux系统	11	2.5 系统菜单	42
1.4.3 安装后的软件维护与更新	16	2.5.1 首选项	42
<b>第2章 GNOME桌面环境</b>	18	2.5.2 系统管理	45
2.1 GNOME桌面环境概述	19	2.5.3 锁住屏幕	51
2.1.1 GNOME注册界面	19	2.5.4 注销	51
2.1.2 GNOME桌面环境	20	2.5.5 关机	51
2.2 GNOME桌面环境浏览	21	2.6 使用移动存储设备	51
2.2.1 GNOME菜单面板	21	2.6.1 浏览移动存储介质	52
2.2.2 GNOME桌面区	24	2.6.2 写入移动存储介质	53
2.2.3 GNOME窗口面板	26	2.7 定制GNOME桌面环境	54
2.3 应用程序菜单	27	2.7.1 定制面板	54
2.3.1 办公	28	2.7.2 定制桌面背景	54
2.3.2 附件	29	2.7.3 定制菜单面板	55
2.3.3 互联网	31	<b>第3章 命令行基础知识</b>	56
2.3.4 图形	33	3.1 命令行结构	57
2.3.5 影音	34	3.2 后台进程	60
		3.3 标准输入/标准输出与标准错误输出	61
		3.4 输入/输出重定向	61
		3.5 管道	65
		3.6 元字符与文件名生成	67



3.7 转义与引用	69	5.3 显示文件内容	113
3.8 命令历史	72	5.3.1 使用 cat 命令显示文件	113
3.8.1 fc 命令	72	5.3.2 使用 more 命令分页显示	
3.8.2 history 命令	74	文件	113
3.8.3 重复执行先前的命令	75	5.3.3 使用 less 命令分页显示	
3.8.4 编辑并执行校正后的命令	76	文件	114
3.8.5 命令行补充	77	5.3.4 使用 head 命令显示文件	
3.9 命令别名	79	前几行内容	115
3.10 作业控制	81	5.3.5 使用 tail 命令显示文件	
3.11 会话记录与命令确认	83	最后几行内容	116
3.11.1 保存会话记录	83	5.4 复制文件	116
3.11.2 确保使用的命令是		5.5 移动文件	117
正确的	84	5.6 删 除文件	118
<b>第 4 章 文件系统基础知识</b>	<b>86</b>	5.7 显示当前工作目录	119
4.1 文件系统的层次结构	87	5.8 改换目录	119
4.1.1 树形层次结构	87	5.9 创建目录	121
4.1.2 路径名	88	5.10 移动目录	121
4.2 文件系统的组织结构	88	5.11 复制目录	121
4.3 文件的类型	92	5.12 删 除目录	122
4.3.1 普通文件	92	5.13 比较文件之间的差别	123
4.3.2 目录文件	94	5.13.1 使用 diff 命令比较两个	
4.3.3 特殊文件	95	文件	123
4.3.4 链接文件	98	5.13.2 使用 diff3 命令比较 3 个	
4.3.5 符号链接文件	99	文件	123
4.3.6 管道文件	101	5.14 从系统中检索文件	124
4.4 文件的安全保护机制	101	5.14.1 简单检索	126
4.4.1 显示文件的访问权限	102	5.14.2 使用逻辑运算符	126
4.4.2 修改文件的访问权限	103	5.14.3 利用 find 命令本身实现	
4.4.3 设置文件的访问权限	104	其他处理功能	126
4.4.4 其他访问权限设置	106	5.14.4 利用管道实现其他	
<b>第 5 章 文件和目录操作</b>	<b>107</b>	处理功能	127
5.1 创建文件	108	5.15 检索文件内容	127
5.2 显示文件列表	108	5.15.1 利用 grep 检索文件	
5.2.1 使用 ls 命令显示文件		内容	127
列表	108	5.15.2 过滤其他命令的输出	
5.2.2 利用通配符显示文件	110	数据	128
5.2.3 显示隐藏文件	111	5.15.3 使用 grep 检索多个文件	128
5.2.4 递归地列出文件	112	5.15.4 检索不包含特定字符	
		串的文本行	129
		5.15.5 在 grep 中使用正则	

表达式 .....	129	6.9.1 删除或替换特殊字符 .....	148
5.15.6 检索元字符本身 .....	131	6.9.2 在编辑期间运行 Linux 命令 .....	149
5.15.7 在命令行中使用引号 .....	131	6.10 vim 编辑器命令总结 .....	149
5.16 排序 .....	132	<b>第 7 章 Shell 基础知识 .....</b>	<b>153</b>
<b>第 6 章 编辑文件 .....</b>	<b>133</b>	7.1 Shell 与 Shell 编程 .....	154
6.1 启动 vim 编辑器 .....	134	7.1.1 为什么需要 Shell 编程 .....	154
6.1.1 创建文件 .....	134	7.1.2 什么是 Shell 脚本 .....	155
6.1.2 状态行 .....	135	7.1.3 运行 Shell 脚本 .....	155
6.2 vim 编辑器的两种工作模式 .....	135	7.1.4 退出与出口状态 .....	156
6.2.1 输入模式 .....	135	7.1.5 调用适当的 Shell 解释程序 .....	158
6.2.2 命令模式 .....	135	7.1.6 位置参数 .....	159
6.3 保存编辑的文件并退出 vim .....	136	7.2 变量与变量替换 .....	161
6.4 vim 编辑器的基本命令 .....	137	7.2.1 变量分类 .....	162
6.4.1 移动光标位置 .....	137	7.2.2 变量的赋值 .....	162
6.4.2 输入文本 .....	138	7.2.3 内部变量 .....	163
6.4.3 修改与替换文本 .....	138	7.2.4 变量的引用与替换 .....	165
6.4.4 撤销先前的修改 .....	139	7.2.5 变量的间接引用 .....	167
6.4.5 删除文本 .....	139	7.2.6 特殊的变量替换 .....	167
6.4.6 复制、删除与粘贴文本 .....	140	7.2.7 变量声明与类型定义 .....	170
6.4.7 按指定的数量重复执行命令 .....	141	7.3 命令与命令替换 .....	171
6.5 使用 ex 命令 .....	141	7.3.1 Shell 内部命令 .....	171
6.5.1 显示行号 .....	141	7.3.2 部分命令介绍 .....	174
6.5.2 多行复制 .....	142	7.3.3 命令替换 .....	183
6.5.3 移动文本行 .....	142	7.4 test 语句 .....	185
6.5.4 删除文本行 .....	142	7.4.1 文件测试运算符 .....	186
6.6 检索与替换 .....	142	7.4.2 字符串测试运算符 .....	187
6.6.1 检索字符串 .....	142	7.4.3 整数值测试运算符 .....	188
6.6.2 模式检索 .....	143	7.4.4 逻辑运算符 .....	189
6.6.3 替换字符串 .....	144	7.5 命令行的解释执行过程 .....	190
6.7 编辑多个文件 .....	145	7.5.1 读取命令行 .....	191
6.7.1 编辑多个文件 .....	145	7.5.2 命令历史替换 .....	191
6.7.2 合并文件与合并文本行 .....	145	7.5.3 别名替换 .....	192
6.8 定制 vim 编辑器的运行环境 .....	145	7.5.4 花括号扩展 .....	192
6.8.1 临时设定 vim 的运行环境 .....	145	7.5.5 波浪号替换 .....	192
6.8.2 永久性地定制 vim 的运行环境 .....	148	7.5.6 I/O 重定向 .....	193
6.9 其他特殊说明 .....	148	7.5.7 变量替换 .....	194
		7.5.8 算术运算结果替换 .....	195



7.5.9 命令替换.....	195	8.15.2 Shell 脚本的调试 .....	242
7.5.10 单词解析.....	195	8.15.3 系统性能考虑 .....	246
7.5.11 文件名生成.....	196	<b>第 9 章 进程管理 .....</b>	248
7.5.12 引用字符处理.....	197	9.1 ps 命令概述 .....	249
7.5.13 进程替换.....	197	9.2 查询进程及其状态信息 .....	251
7.5.14 环境处理.....	198	9.2.1 查询当前活动的进程 .....	251
7.5.15 执行命令.....	198	9.2.2 查询系统中的所有进程 .....	251
7.5.16 跟踪执行过程.....	199	9.2.3 显示进程的重要状态 信息 .....	252
<b>第 8 章 Shell 高级编程 .....</b>	200	9.2.4 显示进程的详细状态 信息 .....	253
8.1 if 条件语句 .....	201	9.2.5 显示进程间的调用关系 .....	253
8.1.1 if 语句的表现形式 .....	201	9.2.6 pstree 命令 .....	254
8.1.2 嵌套的 if-then 条件测试 .....	202	9.3 监控进程及系统资源 .....	255
8.1.3 if-then 结构参考 .....	204	9.4 终止进程的运行 .....	259
8.2 case 分支语句 .....	205	9.5 调整分时进程的优先级 .....	261
8.3 for 循环语句 .....	207	9.5.1 nice 命令 .....	261
8.4 while 循环语句 .....	210	9.5.2 renice 命令 .....	263
8.5 until 循环语句 .....	212	9.5.3 调整进程优先级的作用 .....	263
8.6 select 循环语句 .....	213	<b>第 10 章 proc 文件系统 .....</b>	265
8.7 嵌套的循环 .....	214	10.1 进程内存映像文件 .....	266
8.8 循环控制与辅助编程命令 .....	215	10.2 系统配置信息 .....	270
8.8.1 break 和 continue 命令 .....	215	10.3 系统运行状态信息 .....	273
8.8.2 true 命令 .....	217	10.4 系统可调参数 .....	276
8.8.3 sleep 命令 .....	217	10.4.1 文件系统可调参数 .....	277
8.8.4 shift 命令 .....	217	10.4.2 系统内核可调参数 .....	277
8.8.5 getopt 命令 .....	218	10.4.3 sysctl 命令 .....	281
8.8.6 getopt 命令 .....	219	10.5 其他重要的子目录 .....	282
8.9 循环语句的 I/O 重定向 .....	221	<b>第 11 章 磁盘空间管理 .....</b>	285
8.9.1 while 循环的 I/O 重定向 .....	221	11.1 查询磁盘空间信息 .....	286
8.9.2 until 循环的 I/O 重定向 .....	222	11.1.1 常用的磁盘空间 管理工具 .....	286
8.9.3 for 循环的 I/O 重定向 .....	222	11.1.2 使用 df 命令检查磁盘 空间的使用情况 .....	286
8.10 Here 文档 .....	223	11.1.3 使用 du 命令检查目录 占用的存储空间 .....	289
8.11 Shell 函数 .....	227	11.1.4 使用 find 命令找出超过	
8.12 逻辑与和逻辑或并列结构 .....	232		
8.12.1 逻辑与命令并列结构 .....	232		
8.12.2 逻辑或命令并列结构 .....	233		
8.13 Shell 数组 .....	233		
8.14 信号的捕捉与处理 .....	238		
8.15 其他 Shell 课题 .....	241		
8.15.1 子 Shell .....	241		

一定容量限制的文件	290	12.3.5	删除软件包	332
11.1.5 使用 <code>find</code> 命令找出并删除 长期闲置不用的文件	290	12.3.6	图形界面	333
11.1.6 使用 <code>find</code> 命令找出并删除 <code>core</code> 文件	292	12.4	<code>synaptic</code> 图形界面软件管理 工具	333
11.1.7 使用 <code>ls</code> 命令检测文件的 大小	292	12.4.1	浏览软件包	335
11.2 采用标准工具备份与恢复数据	292	12.4.2	安装软件包	335
11.2.1 利用 <code>cpio</code> 实现备份和 恢复	294	12.4.3	删除软件包	336
11.2.2 利用 <code>tar</code> 实现备份和 恢复	298	12.4.4	软件升级	337
11.2.3 利用 <code>dd</code> 实现文件系统的 原样复制	304	12.5	<code>GNOME</code> 软件增删工具	338
11.3 采用专用工具备份与恢复数据	305	12.6	软件包的自动更新	339
11.3.1 利用 <code>dump</code> 命令实现数据的 备份	306	第 13 章	用户管理	341
11.3.2 利用 <code>restore</code> 命令实现 数据的恢复	308	13.1	增加与删除用户	342
11.4 文件系统限额管理	310	13.1.1	<code>/etc/passwd</code> 文件	342
11.4.1 限额概述	310	13.1.2	<code>/etc/shadow</code> 文件	343
11.4.2 设置限额	312	13.1.3	用户管理实例	344
11.4.3 限额的维护	315	13.2	定制用户的工作环境	349
第 12 章 软件管理	318	13.2.1	选择命令解释程序	349
12.1 软件管理概述	319	13.2.2	设置用户初始化文件	351
12.1.1 软件维护工具	319	13.2.3	定制 Shell 工作环境	353
12.1.2 软件管理基本概念	319	13.3	增加与删除用户组	359
12.2 利用 <code>apt-get</code> 管理软件包	321	13.4	监控用户	360
12.2.1 安装软件包	323	13.4.1	利用 <code>who</code> 命令查询 系统中的用户	361
12.2.2 系统的更新与升级	324	13.4.2	利用 <code>finger</code> 命令查询 系统中的用户	362
12.2.3 删除软件包	325	13.4.3	利用 <code>w</code> 命令查询系统 中的用户活动	362
12.2.4 安装本地存储介质中的 软件包	325	13.4.4	向注册用户发送消息	362
12.2.5 <code>sources.list</code> 配置文件	325	13.5	插件式认证模块	363
12.3 利用 <code>aptitude</code> 管理软件包	327	13.5.1	配置文件、模块类型与 控制标志	363
12.3.1 安装软件包	329	13.5.2	修改 PAM 配置文件	366
12.3.2 系统的升级	330	13.6	超级用户与 <code>sudo</code> 命令	366
12.3.3 查询软件包	330	13.6.1	超级用户的访问控制	367
12.3.4 检索软件包	330	13.6.2	利用 <code>sudo</code> 运行特权 命令	368
		13.6.3	<code>sudoers</code> 配置文件	369
		13.6.4	<code>admin</code> 用户组成员的 访问权限	373



13.6.5 直接使用 root 注册 .....	373	文件 .....	404
13.6.6 以不同的用户身份 访问系统.....	373	15.2.3 显示 crontab 文件 .....	405
<b>第 14 章 系统启动与关机.....</b>	<b>375</b>	15.2.4 删除 crontab 文件 .....	405
14.1 磁盘分区与 GRUB.....	376	15.2.5 crontab 命令的访问控制 .....	405
14.1.1 磁盘分区 .....	376	15.2.6 应用实例——数据库 定时备份 .....	406
14.1.2 GRUB.....	378	<b>15.3 调度一次性执行的作业 .....</b>	<b>407</b>
14.1.3 GRUB 配置文件.....	379	15.3.1 提交 at 作业 .....	408
14.1.4 安装或修复 GRUB.....	381	15.3.2 显示 at 作业及作业队列 .....	409
14.2 初始引导过程 .....	383	15.3.3 删除 at 作业 .....	409
14.2.1 GRUB 引导过程概述 .....	384	15.3.4 at 命令的访问控制 .....	409
14.2.2 补充说明 .....	386	15.3.5 应用实例——系统 定时关机 .....	410
14.3 系统生成过程 .....	386	<b>15.4 系统日志 .....</b>	<b>411</b>
14.3.1 基本概念 .....	388	15.4.1 系统日志文件 .....	412
14.3.2 init 进程与 /etc/event.d 目录 .....	391	15.4.2 应用程序日志文件 .....	413
14.3.3 启动用户定义的应用 程序 .....	394	15.4.3 无法直接查阅的日志 .....	413
14.4 login 进程 .....	395	15.4.4 系统日志守护进程 .....	414
14.4.1 login 进程与 passwd 文件 .....	395	<b>第 16 章 文件系统内部组织 .....</b>	<b>417</b>
14.4.2 Shell 进程与 profile 文件 .....	395	16.1 文件系统的组织结构 .....	418
14.5 系统关机过程 .....	396	16.1.1 引导块 .....	419
14.5.1 使用 shutdown 命令 关闭系统 .....	396	16.1.2 数据块组 .....	419
14.5.2 使用 init 命令关闭系统 .....	397	16.2 超级块 .....	422
14.5.3 使用其他命令关机 .....	397	16.3 信息节点 .....	424
<b>第 15 章 作业调度与系统日志 .....</b>	<b>398</b>	16.3.1 文件的类型与访问权限 .....	426
15.1 定时运行后台作业 .....	399	16.3.2 数据块地址数组 .....	426
15.1.1 cron 守护进程的 调度过程 .....	399	16.3.3 符号链接文件 .....	427
15.1.2 at 作业与 atd 守护进程 .....	400	16.3.4 特权标志位 .....	427
15.1.3 调度错失执行时间 的任务 .....	401	16.4 信息节点与目录及文件的关系 .....	428
15.2 调度重复执行的任务 .....	402	16.4.1 目录文件 .....	428
15.2.1 crontab 文件的工作原理 .....	402	16.4.2 目录、文件和信息节点 三者之间的关系 .....	429
15.2.2 创建和编辑 crontab		<b>第 17 章 文件系统管理 .....</b>	<b>431</b>
		17.1 划分磁盘分区 .....	432
		17.2 创建文件系统 .....	434
		17.2.1 mkfs 或 mke2fs 命令 介绍 .....	434
		17.2.2 创建 Ext2/Ext3 文件	

系统 .....	436	18.5 配置网络服务 .....	485
17.3 调整文件系统 .....	437	18.6 网络管理与维护 .....	487
17.4 安装与卸载文件系统 .....	439	18.6.1 使用 ifconfig 命令维护 网络接口 .....	487
17.4.1 安装文件系统概述 .....	439	18.6.2 使用 netstat 命令监控 网络状态 .....	489
17.4.2 mount 命令 .....	440	18.6.3 使用 ping 命令测试远程 主机的连通性 .....	494
17.4.3 /etc/fstab 文件 .....	441	18.6.4 使用 ping 命令检测网络 主机的性能 .....	495
17.4.4 安装文件系统 .....	442	18.6.5 使用 ftp 命令检测网络 主机的传输性能 .....	496
17.4.5 卸载文件系统 .....	444	18.6.6 使用 traceroute 命令跟踪 路由信息 .....	496
17.5 检测与修复文件系统 .....	446	18.6.7 利用 tcpdump 捕捉、分析 网络分组数据 .....	497
17.5.1 何时需要检测文件系统 .....	447		
17.5.2 文件系统检测的内容 .....	448		
17.5.3 交互地检测与修复 文件系统 .....	452		
17.5.4 自动检测与修复文件 系统 .....	453		
17.5.5 恢复严重受损的超级块 .....	454		
17.5.6 解决 fsck 命令无法修复 的文件系统问题 .....	454		
17.5.7 fsck 的阶段处理方式 .....	455		
17.6 调试文件系统 .....	458	第 19 章 TCP/IP 网络应用 .....	501
17.6.1 概述 .....	458	19.1 OpenSSH .....	502
17.6.2 交互式调试子命令 .....	459	19.1.1 安装 OpenSSH 服务器 .....	502
17.6.3 应用举例 1——恢复 误删的文件 .....	464	19.1.2 /etc/ssh/sshd_config 配置文件 .....	502
17.6.4 应用举例 2——恢复 误删的文件 .....	465	19.1.3 使用 SSH 注册到 远程系统 .....	505
17.7 其他文件系统维护工具 .....	467	19.1.4 使用 ssh 执行远程系统 中的命令 .....	506
17.7.1 dumpe2fs 命令 .....	467	19.1.5 使用 SCP 替代 FTP .....	506
17.7.2 e2image 命令 .....	468	19.1.6 使用 SFTP 替代 FTP .....	507
<b>第 18 章 TCP/IP 网络管理 .....</b>	<b>470</b>	19.1.7 SSH 与 SCP 的无 密码注册 .....	508
18.1 TCP/IP 简介 .....	471	19.1.8 OpenSSH 的安全考虑 .....	510
18.1.1 TCP/IP 的层次结构 .....	471	19.2 Telnet 远程注册 .....	510
18.1.2 TCP/IP 如何处理数据 通信 .....	473	19.2.1 设置 Telnet 服务器 .....	511
18.2 网络接口设置 .....	475	19.2.2 Telnet 服务器的安全 考虑 .....	512
18.2.1 以太网络设置 .....	475	19.3 FTP 文件传输 .....	513
18.2.2 ADSL 网络连接 .....	480	19.3.1 设置 vsftpd .....	513
18.3 主机名字解析 .....	483	19.3.2 vsftpd.conf 配置文件 .....	513
18.4 网络路由设置 .....	484	19.3.3 FTP 安全考虑 .....	517
		19.3.4 FTP 应用 .....	518



19.3.5 FTP 自动注册 .....	519	21.4.2 直接映射文件 .....	560
<b>第 20 章 DNS 域名服务器 .....</b>	<b>521</b>	21.4.3 间接映射文件 .....	560
20.1 DNS 基本概念 .....	522	21.5 NFS 故障修复 .....	561
20.1.1 域与区 .....	522	21.5.1 基本工具 .....	561
20.1.2 DNS 域名服务器 .....	523	21.5.2 其他注意事项 .....	564
20.1.3 DNS 域名与地址解析 .....	524	<b>第 22 章 Samba 资源共享 .....</b>	<b>565</b>
20.2 DNS 配置文件 .....	526	22.1 安装 Samba 服务器 .....	566
20.2.1 resolv.conf 文件 .....	527	22.2 smb.conf 配置文件 .....	567
20.2.2 named.conf 配置文件 .....	528	22.2.1 smb.conf 配置文件概述 .....	568
20.2.3 区配置文件 .....	532	22.2.2 Global 节 .....	569
20.2.4 DNS 资源记录 .....	533	22.2.3 homes 节 .....	572
20.3 DNS 服务器配置过程 .....	537	22.2.4 printers 节 .....	574
20.3.1 设置 resolv.conf 配置文件 .....	537	22.3 快速设置 Samba 服务器 .....	575
20.3.2 设置 named.conf 配置文件 .....	537	22.3.1 设定 Samba 服务器的工作组或域 .....	575
20.3.3 设置正向区配置文件 .....	538	22.3.2 设置 Samba 用户认证信息 .....	576
20.3.4 设置反向区配置文件 .....	539	22.3.3 共享用户主目录 .....	577
20.3.5 DNS 视图 .....	540	22.3.4 共享其他目录 .....	577
20.3.6 检测配置文件 .....	543	22.3.5 共享打印机 .....	578
20.4 测试 DNS 服务器 .....	544	22.3.6 验证 Samba 配置文件 .....	579
20.4.1 验证 DNS 服务器 .....	544	22.4 Samba 运行环境测试 .....	580
20.4.2 dig 命令 .....	545	22.4.1 在 Linux 系统中测试 Samba 服务器 .....	580
<b>第 21 章 NFS 网络文件系统 .....</b>	<b>548</b>	22.4.2 从 Windows 系统中连接 Samba 服务器 .....	583
21.1 NFS 简述 .....	549	22.5 访问共享资源 .....	584
21.2 配置 NFS 服务器 .....	550	22.5.1 从 Windows 系统中访问 Samba 服务器 .....	584
21.2.1 安装 NFS 服务器软件包 .....	550	22.5.2 从 Linux 系统中访问 Windows 服务器 .....	585
21.2.2 /etc/exports 文件 .....	551	<b>第 23 章 Apache 服务器 .....</b>	<b>588</b>
21.2.3 采用图形界面配置 NFS 共享资源 .....	553	23.1 Apache 服务器概述 .....	589
21.2.4 验证 NFS 共享资源的配置 .....	555	23.2 启动 Apache 服务器 .....	589
21.3 配置 NFS 客户系统 .....	557	23.2.1 Apache 软件包的目录结构 .....	589
21.3.1 安装远程文件系统 .....	557	23.2.2 apachc2 守护进程 .....	590
21.3.2 设置 /etc/fstab 文件 .....	558	23.2.3 设置 Apache 启动脚本 .....	591
21.4 NFS 自动安装 .....	559		
21.4.1 主映射文件 .....	559		

23.2.4 Apache 模块 .....	592
23.3 配置 Apache 服务器 .....	592
23.3.1 Apache 配置文件 .....	593
23.3.2 .htaccess 文件 .....	594
23.3.3 配置指令 .....	594
23.4 用户目录 .....	599
23.4.1 利用 UserDir 设定 目录路径 .....	600
23.4.2 限定用户目录的使用 .....	600
23.4.3 开放用户 CGI 目录 .....	600
23.5 虚拟主机 .....	601
23.5.1 配置基于主机名的虚拟 主机 .....	602
23.5.2 配置基于 IP 地址的 虚拟主机 .....	603
23.5.3 利用不同的 IP 地址提供 相同的网站服务 .....	604
23.5.4 利用不同的端口提供 不同的网站服务 .....	604
23.6 利用 CGI 提供动态内容服务 .....	605
23.6.1 启用 CGI 程序 .....	605
23.6.2 编写 CGI 程序 .....	606
23.6.3 CGI 的安全考虑与 suexec .....	608
23.6.4 Apache 与 LAMP .....	609
23.7 用户认证 .....	610
23.7.1 用户认证的实现 .....	610
23.7.2 用户认证方法的补充 说明 .....	612
23.8 日志文件 .....	613
23.8.1 错误日志文件 .....	614
23.8.2 访问日志文件 .....	615
23.8.3 虚拟主机日志 .....	617
<b>第 24 章 MySQL 数据库 .....</b>	<b>618</b>
24.1 安装与配置 MySQL 数据库 .....	619
24.1.1 安装 MySQL 数据库 .....	619
24.1.2 my.cnf 配置文件 .....	619
24.1.3 MySQL 数据库命令行 界面 .....	621
24.1.4 MySQL 数据库图形界面 .....	622
24.1.5 设置数据库用户及其 访问权限 .....	624
24.2 访问 MySQL 数据库 .....	624
24.2.1 创建、查询、使用与删除 数据库 .....	624
24.2.2 创建、查询与删除数 据库表 .....	625
24.2.3 录入数据 .....	626
24.3 查询 MySQL 数据库 .....	627
24.3.1 查询数据库表 .....	627
24.3.2 查询数据库表结构 .....	628
24.3.3 查询数据库表中的数据 内容 .....	628
24.4 SQL 脚本与批处理 .....	628
24.5 MySQL 数据库的备份与恢复 .....	630
24.5.1 数据库备份方法 .....	630
24.5.2 MySQL 数据库备份 .....	631
24.5.3 MySQL 数据库恢复 .....	632
24.5.4 MySQL 数据库表的 备份与恢复 .....	633
24.5.5 增量备份与恢复 .....	633
24.6 密码维护与网络安全 .....	635
24.6.1 维护数据库管理员密码 .....	635
24.6.2 恢复数据库管理员密码 .....	636
24.6.3 基本网络安全考虑 .....	637
<b>参考文献 .....</b>	<b>638</b>



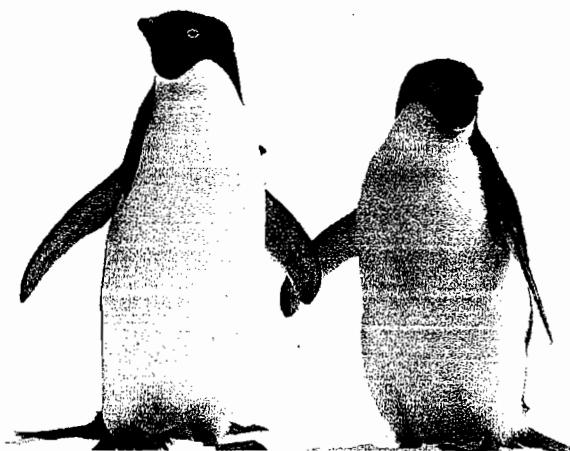
# LINUX

## 第1章

### 系统概述与安装

本章首先将简单介绍 Linux 的发展过程，然后概述 Linux 的各种网络资源与求助方法，最后详细介绍 Linux 系统的安装过程。其内容主要包括：

- Linux 的兴起与发展；
- 充分利用网上资源；
- 随时查询随机文档；
- 安装 Ubuntu Linux 系统。





## 1.1 Linux 的兴起与发展

提到 Linux 的缘起，不能不涉及 UNIX。UNIX 系统早期之所以能够取得巨大的成功并迅速得到普及，主要在于其 3 个重要特点：简洁性、开放性与可移植性。向大学和研究机构公开源代码，激发了软件开发人员对 UNIX 系统进行研究和移植的兴趣，致使 UNIX 成为操作系统的宠儿。许多大学均以 UNIX 作为操作系统课程的研究对象，从而出现了《UNIX 操作系统设计》等著名的 UNIX 教材，使 UNIX 成为大学操作系统课程的代名词，同时也培养了许多潜在的 UNIX 系统准用户。

而后期的商业化运作方式，使得 UNIX 系统及其源代码成为专属产品，从而限制了软件人员对 UNIX 系统的研究、开发和使用。另外，为了考虑特定的机器结构，商业化的 UNIX 也开始变得越来越复杂，基本上失去了可移植性的特点。而这一切则导致了开源软件运动的兴起，其中的一个结果就是催生了 Linux。

1984 年，Richard Stallman（UNIX 系统 emacs 编辑器的开发者）发起了一场自由软件共享活动，创建了一个非盈利性的自由软件基金会（Free Software Foundation），支持开发与共享自由软件。其中的 GNU 项目旨在开发一个完全免费的、类似于 UNIX 的 GNU 操作系统，但不使用 UNIX 系统的任何源代码。Stallman 希望通过社区参与的方式，促进 GNU 操作系统的发展，使用户能够自由地交流、学习，从而改进或不断地增强这一系统。由于开发一个完整的操作系统（包括内核与实用程序）是一项艰巨的任务，所以 GNU 决定采用模块化的设计方法，以便任何人都能够同时参与，共同开发各个操作系统模块，且能够非常容易地集成现有的自由软件。到了 1990 年，针对 UNIX 系统的所有实用程序、工具与核心库，GNU 几乎都有了自己的相应软件，其中包括 emacs 文本编辑器及 C 编译器 gcc 等，但缺乏一个内核。

与此同时，1991 年时尚在芬兰赫尔辛基大学读书的 Linus Torvalds 决定在个人计算机上创建一个新的、类似于 UNIX 操作系统的内核。Torvalds 一直使用由 Andrew Tannenbaum 设计与实现的 Minix 操作系统，因而熟悉 UNIX 系统的功能特性。Torvalds 决定开发一个可在个人计算机上运行的 UNIX 系统，并于 1991 年 9 月推出了 Linux 0.01 版。由于开发一个高质量的操作系统非一人之力所能及，于是，Torvalds 利用 Internet 对外公开了其源代码，任何人均可以免费下载和使用。Torvalds 邀请其他人下载其新内核的副本，帮助改善和增加新的功能特性。此举立即引起了世界各地软件开发人员的极大兴趣，许多人决定接受 Torvalds 的提议，开始参与 Linux 的开发与传播。作为一个团队，他们分工合作，改进 Linux，扩展了 Linux 内核，开发了许多系统程序和工具软件，把 BSD 与 System V 版 UNIX 的许多功能加到新的 Linux 系统中，从而构成了一个完整的操作系统。

组合了 GNU 软件的 Linux（有时也称作 GNU/Linux）包含类似于 UNIX 的实用程序、工具、核心库、编译器、文本编辑器、桌面环境及其他组成部分，构成了一个完整的 UNIX 系统环境。

从开始之日起，Linux 的所有开发工作一直都是在 Torvalds 的指导下，利用 Internet 相互交流，共同合作完成的。Linux 系统是世界各地许多软件开发人员共同努力的结果，也是借助于 Internet 协同开发的产品。Linux 是一种免费的操作系统，所有的软件，包括源代码、文档和技术支持（通过 Internet）都是免费的。任何人均可自由获取源代码，对其进行研究、修改和重新发行，而这一切都



C2392577B

是免费的。

目前，存在许多不同版本的 Linux 产品，其中比较著名的有 Red Hat、Fedora、Debian、Ubuntu、SUSE、Slackware、Mandriva、Turbolinux 及 Gentoo 等。尽管这些系统在安装和外部表现等方面有所不同，但其内部采用的 Linux 内核、标准实用程序等基本上是一致的，因而具有许多共性。

在 20 世纪 90 年代，南非的 Mark Shuttleworth 参与了 Debian Linux 系统的开发。2004 年 3 月份，他开始转向自己的自由软件世界，成立了 Canonical 公司，决定开发一个基于 Debian 的、用户友好的 Linux 系统，并以此公司作为技术支撑，提供服务。Ubuntu Linux 系统的迅速崛起，致使 Shuttleworth 又于 2005 年投资 1 000 万美金成立了 Ubuntu 基金会，专门致力于 Ubuntu Linux 系统的开发与推广，确保 Ubuntu Linux 系统未来的健康发展。

Ubuntu Linux 系统是众多 Linux 发行品牌之一。在 Linux 世界中，Ubuntu 只是一个后来者，是一个非常年轻的 Linux 发行品牌，用了短短几年的时间 Ubuntu 就发展成为一个流行的、成熟的以及桌面环境丰富的 Linux 系统，受到了从 Linux 初学者到资深专家的大批 Linux 用户的追捧。

Ubuntu 是一个古老的非洲词汇，表示人类之间的关怀、共享、和谐。它作为一种理念，倡导个人、文化以及民族之间的融合、博爱与相互合作。

## 1.2 充分利用网上资源

前面曾经说过，Linux 系统本身是一种“Internet 产品”，网上积累了大量的 Linux 资源与信息，其中包括 Ubuntu Linux 自己的官方网站、GNU 自由软件项目、Linux 文档项目、Linux 专题讨论组（Newsgroups）、电子邮件通讯录（Mailing Lists），以及其他各种各样的 Linux 社区（包括 Ubuntu 社区）和论坛。

用户可以充分利用 Internet，查阅相关的文档，寻求问题或故障的解决方案，也可以上网介绍自己的学习心得，与他人分享自己的学习经验等。

### 1.2.1 Ubuntu 官方网站

Ubuntu Linux 系统的官方网站地址为 <http://www.ubuntu.com>（相应的中文网站地址为 <http://www.ubuntu.org.cn>），其中包含怎样获取 Ubuntu，怎样获得支持，Ubuntu 简介等栏目，以及 Ubuntu 桌面版与服务器版的下载链接。可以此作为出发点，获取命令参考手册、文档与求助信息；也可以直接访问下列网址，从而获取相关的资讯。

- Ubuntu 官方文档 (<https://help.ubuntu.com>) —— 其中提供了 Ubuntu 桌面系统用户指南及 Ubuntu Linux 系统桌面版安装文档等，其界面如图 1-1 所示。
- Ubuntu 社区文档 (<https://help.ubuntu.com/community/UserDocumentation>) —— 其中提供了 Ubuntu 社区贡献的大量文档，包括系统安装、系统管理、服务器管理（如 MySQL 数据库、DNS、Apache、OpenSSH、NFS 与 Samba 等）以及故障修复等。
- Ubuntu 中文维客网站 (<http://wiki.ubuntu.org.cn>) —— 这是一个收集、整理、翻译和编写 Ubuntu Linux 系统中文文档，相互讨论与交流的场合，其中分门别类，提供了大量的中文文档，也欢迎有志者参与文档的翻译与校正工作，其界面如图 1-2 所示。



## Ubuntu 权威指南

- Ubuntu 社区论坛 (<http://www.ubuntu.com/support/community/webforums>) —— 从中可以寻求帮助，以收解疑释惑之效。
- 用户通信 (<https://lists.ubuntu.com>) —— 利用电子邮件，可以向 Ubuntu 同行或爱好者请教问题，获取建议，解答问题等。
- 如果想要了解怎样请求帮助，有什么可用的资源，可以访问 <https://wiki.ubuntu.com/HowToGetHelp>。

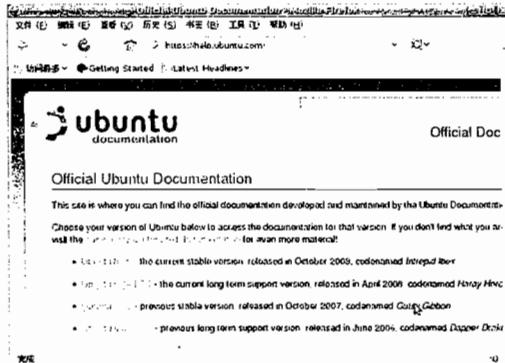


图 1-1 Ubuntu 官方文档

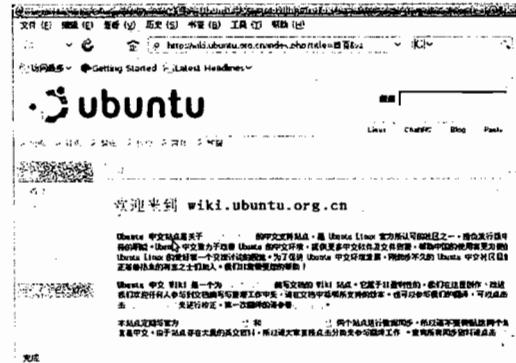


图 1-2 Ubuntu 维客网站

### 1.2.2 GNU 网站

<http://www.gnu.org> 是 GNU 项目的大本营，其中的 <http://www.gnu.org/manual> 提供了许多命令参考手册与文档，如 bash、finger、gawk、grep、info、sed 及 tar 等，如图 1-3 所示。

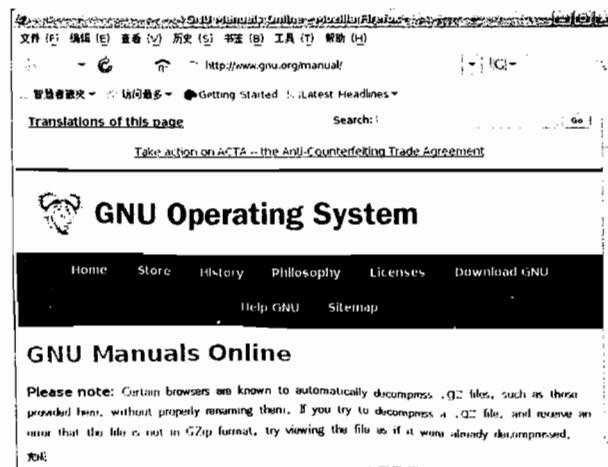


图 1-3 GNU 网站资源

### 1.2.3 Linux 文档项目网站

Linux 文档的一个重要资源是“Linux 文档项目”网站。为了查阅 Linux 文档，可以利用浏览

器，访问“Linux 文档项目”网站 <http://www.tldp.org>。其中的 HOWTO 链接提供了许多特定课题的帮助信息，如 KernelAnalysis-HOWTO、Linux Loadable Kernel Module HOWTO 以及 The Linux Bootdisk HOWTO 等。Guide 链接包含许多 PDF 文档，如 Linux System Administrator Guide、Advanced Bash-Scripting Guide 以及 Linux Kernel 2.4 Internals 等，如图 1-4 所示。

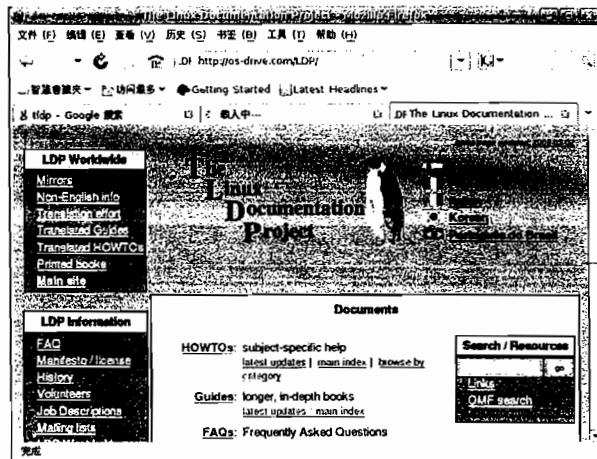


图 1-4 TLDP 网站资源

#### 1.2.4 网上求助

在日常的学习和上机过程中，如果遇到问题，可以利用 Google 等搜索引擎，从 Internet 中积累的庞大 Linux 资源中寻求答案。例如，为了查询怎样禁用 IPv6，以减少不必要的系统开销，提高系统与网络的性能，可在 Google 中搜索“disable ipv6 ubuntu”等，如图 1-5 所示。



图 1-5 Google 网络资源搜索



## 1.3 随时查询随机文档

### 1.3.1 使用“--help”选项查询命令的简单说明

大多数 GNU 命令都提供“--help”选项，用于显示相应命令的用法及其简单说明，对于熟悉和使用 Linux 系统具有极大的帮助，如图 1-6 所示。

```
gqxing@iscas:/etc/event.d$ ps --help
***** simple selection *****      ***** selection by list *****
-A all processes                  -C by command name
-N negate selection               -G by real group ID (supports names)
-a all w/ tty except session leaders -U by real user ID (supports names)
-d all except session leaders     -g by session OR by effective group name
-e all processes                   -p by process ID
-T all processes on this terminal -s processes in the sessions given
-a all w/ tty, including other users -t by tty
g OBSOLETE -- DO NOT USE        -u by effective user ID (supports names)
r only running processes         -U processes for specified users
x processes w/o controlling ttys -t by tty
***** output format *****        ***** long options *****
-o,o user-defined -f full        --group --User --pid --cols --ppid
-j,j job control   -s signal      --group --user --sid --rows --info
-0,0 preloaded -o v virtual memory --cumulative --format --select
-l,l long           -u user-oriented --sort --tty --forest --version
-E extra full      X registers   --heading --no-heading --context
                                ***** misc options *****
-V,V show version    l list format codes f ASCII art forest
-m,m,L,-T,H threads  S children in sum   g change --i format
-M,Z security data   c true command name e scheduling class
-w,w wide output     n numeric WCHAN,TID h process hierarchy
gqxing@iscas:/etc/event.d$
```

图 1-6 使用命令的“--help”选项

如果帮助信息太长，超过一个终端窗口能够显示的内容，可以利用管道机制，以及 less 或 more 命令（如下所示），逐页显示命令的帮助信息（详见第 3 章与第 5 章）。

```
$ ls --help | more
```

对于非 GNU Linux 命令，可以使用“-h”或“-help”选项获取命令的帮助信息。

### 1.3.2 使用 man 命令联机查询系统参考手册

Linux 系统提供大量的系统命令和用户命令，而许多命令又有众多的选项，一个人不可能记住这么多命令，即使具有多年使用 Linux 系统经验的资深人员也是如此。对于常用的命令，也不可能记住每一个选项。为此，Linux 系统提供了一种联机帮助文档，供用户随时查阅命令的说明与用法。

同 UNIX 系统一样，Linux 系统的联机文档也是通过 man 命令实现的。为了查询 Linux 系统提供的任何命令，了解命令的功能、用法、可用的选项以及参数，可以使用下列命令（其中的 command 可以是任何 Linux 命令，包括 man 命令本身）。

```
$ man command
```

Linux 系统中的 man 命令借助于 less 命令，逐页显示指定命令的联机文档，并在终端窗口的

左下角输出一个冒号“:”提示符。此时，可以使用空格键逐页显示余下的解释内容，或者使用“q”子命令退出 man 命令。因此，当使用 man 命令查询联机文档时，可以参照第 5 章中介绍的 less 命令，按照 less 命令的用法逐页显示，或者随机翻阅，如图 1-7 所示。

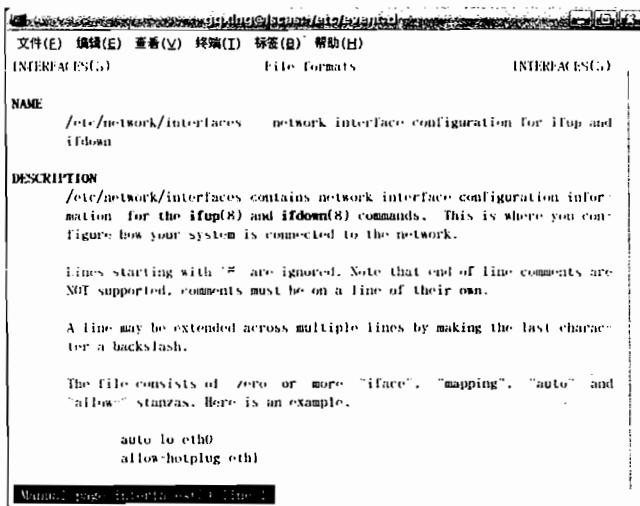


图 1-7 man 命令

对于任何用户而言，利用 man 命令随时查询命令的联机帮助信息是非常有用的。对于常用的命令，即使忘记不常用的命令选项及其用法，当只知其名不知其用法，或者只知其意不知其名（参见后面即将介绍的“man -k”命令）时，均可借助于 man 命令获取帮助信息。

由于 Linux 系统的参考手册非常庞大，为了便于快速地阅联机文档，按照 UNIX 系统的传统，以及文件系统层次组织的标准（Filesystem Hierarchy Standard, FHS），Linux 系统把各种参考手册的内容至少分为 10 节，其中的 8 节分别对应下列 8 种类型文档：

- (1) 用户级的命令、Shell 脚本和应用程序；
- (2) 系统调用（系统内核提供的函数）；
- (3) 库函数；
- (4) 特殊文件（通常是位于/dev 目录中的设备文件）；
- (5) 系统文件格式定义；
- (6) 游戏；
- (7) 系统数据定义文件（如信号和时间等）、网络协议（如 IP、TCP 与 UDP 等）以及字符集（如 UTF-8、unicode 与 iso-8859 等）；
- (8) 系统管理与维护等特权命令（通常是仅适用于超级用户 root 的命令等）。

chmod、chown、kill、mount、nice 与 uname 等既是 Linux 系统中的命令，也是同名的系统调用。passwd、locale 与 crontab 等既是命令名，又是系统文件名（或一类文件的总称）。如果不加以区别，man 命令通常只会给出命令的说明，而不会输出系统调用或系统文件格式等的说明。因此，为了查询 passwd 系统文件，可以使用下列命令（在查询的对象之前，指定对象所属的文档类别）。

```
$ man 5 passwd
```



当用户需要完成某种任务，而不知道究竟应使用哪一个命令及其确切的名字时，可以在 man 命令中使用 “-k” 选项，按照关键字进行检索。“man -k” 命令将会利用给定的关键字检索所有联机手册中每个命令的简单描述部分，并显示匹配的命令及其简单说明。

下面的例子说明了怎样使用 “man -k” 命令查询关键字 locale，其输出结果包括每个与之相关的命令、系统调用、库函数、文件、文档类型及简单描述。

```
$ man -k locale
locale (1)           - Get locale-specific information.
locale (5)           - Describes a locale definition file
locale (7)           - Description of multi-language support
locale-gen (8)        - compile a list of locale definition files
locale.alias (5)      - Locale name alias data base
Locale::gettext (3pm) - message handling functions
localedef (1)         - compile locale definition files
luit (1)             - Locale and ISO 2022 support for Unicode terminals
lxterm (1)           - locale-sensitive wrapper for xterm
update-locale (8)     - Modify global locale settings
validlocale (8)       - Test if a given locale is available
$
```

所有的 man 联机文档均位于 /usr/share/man 目录中。由于 man 联机文档是采用特殊的 nroff 格式实现的，无法直接打开阅读。为了获得文本格式的文档，以便复制或下载到计算机中以供随时查阅，可以使用 col 命令剔除其中的格式控制字符。例如，为了生成一个文本格式的 find 命令说明文件，可以使用下列命令。

```
$ man find | col -b > find
$
```

### 1.3.3 使用 info 命令查询命令的相关信息

info 是一个由 GNU 项目开发，且随 Linux 系统一同被发行的实用程序，它是一个基于菜单选择的超文本帮助工具。info 命令本身包含一个自学功能（也可以使用 “info info” 命令，或者访问 <http://www.gnu.org/software/texinfo/manual/info> 网站，获取其使用说明），其中提供 Shell、系统实用程序以及应用程序的说明文档。当使用 man 命令无法获取某个命令（如 cpio 或 dd 命令）的使用说明时，可以借助于 info 命令。图 1-8 给出的是执行 info 命令后显示的内容。此时，可以通过下列按键求得各种命令的帮助信息。

- h 交互地学习和了解 info 命令的用法和功能。
- ? 列出 info 提供的内部命令。
- 空格 在 info 列出的命令查询菜单中上下滚动，以便定位欲查阅的命令。
- d 返回命令查询菜单。
- m 使用 m 和随后的命令名快速定位一个欲查询的命令。
- q 退出 info 命令。

在 info 列出的命令查询菜单中，每个命令占用一行；除星号 “\*” 之外，第一列是各种 Linux 命令的名字，括号内的内容是命令所属的软件包，最后一列是命令的简要说明。

在 info 命令查询菜单中，每个菜单项实际上是一个链接，指向相应命令的说明文档。为了查阅某个命令，可以把光标移至相应的命令项，然后按 Enter 键；也可以直接输入 “m cmd”，然后

按 Enter 键，其中的 cmd 是命令的名字。例如，为了查询 sleep 命令的说明，可以输入“m sleep”，然后按 Enter 键。当输入字符 m 后，光标会立即移至终端窗口左下角最后一行的起始位置，同时显示“Menu item:”提示信息；接着输入 sleep，当按下 Enter 键之后，info 将会显示选定的命令说明信息，如图 1-8 所示。

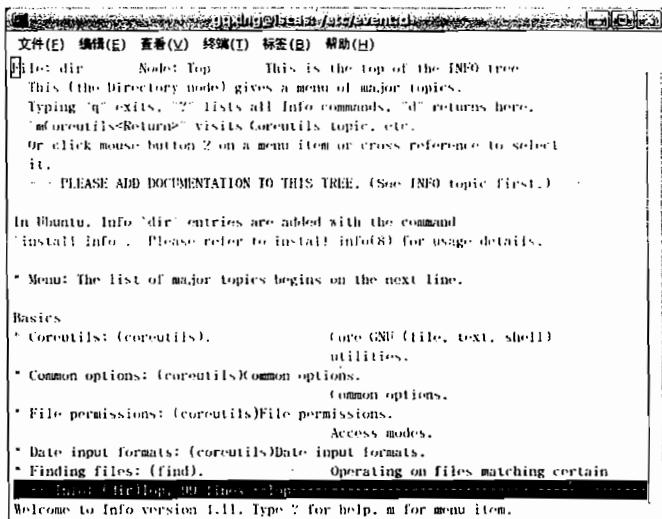


图 1-8 info 命令

为了深入了解 info 命令本身的用法，可以使用“info info”命令求得帮助信息，如果想了解 info 的更多信息，可以访问 <http://www.gnu.org/software/texinfo/manual/info>。

## 1.4 安装 Ubuntu Linux 系统

Ubuntu Linux 系统支持不同 CPU 类型的计算机，其中包括 Intel x86 系列处理器及其兼容机、PowerPC 处理器以及 64 位的 Intel/AMD 处理器等。它可以被安装到几乎所有的台式机、笔记本电脑及服务器中，其安装方式与安装过程极其灵活，即使先前没有 Linux 方面知识的用户，也能够毫无困难地安装一个 Linux 学习环境。

Ubuntu Linux 系统既可以单独安装，也可以与 Microsoft Windows 系统安装在同一台计算机上——把 Ubuntu Linux 安装到 Windows 系统未用的磁盘分区中。注意，在安装 Linux 与 Windows 双系统时，应首先安装 Windows，然后再安装 Ubuntu Linux 系统，否则无法正常启动 Ubuntu Linux 系统。

此外，在安装 Windows 系统时，必须为 Ubuntu Linux 系统预留出磁盘空间。如果 Windows 系统已经分为多个逻辑盘，如 C、D 和 E 3 个逻辑盘，需要事先删除一个逻辑盘。否则，不管 Windows 系统划分的分区是否已经使用，都无法安装 Ubuntu Linux 系统。例如，如果想把 Linux 系统安装在 E 盘位置，可以选择“管理工具→计算机管理→磁盘管理”，在磁盘窗口中右击 E 盘，然后从上下文菜单中选择“删除逻辑驱动器”。



## 1.4.1 安装前的准备

### 1. 硬件要求

在安装 Ubuntu Linux 系统时，不同的系统与版本对硬件的要求不尽相同。表 1-1 以 Intel x86 系列机和桌面版的 Ubuntu Linux 系统为例，给出了一个比较合理的基本硬件要求，供读者作为选择计算机系统时的参考，其中主要包括 CPU、内存及磁盘空间等需求。

表 1-1

硬件系统要求

硬件系统要求	简 单 说 明
CPU	至少选用 1.0 GHz 的 Intel x86 系列 CPU，建议采用 1.2 GHz 或更快的 Intel x86 系列 CPU
内存	至少配备 256MB（或 384MB）内存，建议配备 512MB 或更多的内存
磁盘或磁盘分区	标准桌面系统本身需要 2GB 的存储空间，完整桌面系统本身需要 4GB 的存储空间。此外还要考虑预留两倍内存容量的磁盘空间作为交换分区，以及用户数据的空间需求。因此，完整的安装至少需要 6~8GB 的磁盘空间
VGA 显卡/显示器分辨率	分辨率 1 024×768 像素
引导设备	CD/DVD 驱动器，或者采用 USB 移动盘、内置硬盘或其他安装方式

### 2. 磁盘分区

安装 Ubuntu Linux 系统时，至少需要两个磁盘分区，分别用于创建“/”文件系统与交换分区。对于初学者、个人使用的 Ubuntu Linux 系统而言，最简单或最佳的选择是，除了划分并预留一个磁盘分区作为交换分区之外，可以把整个磁盘分区作为一个大的“/”文件系统。在此情况下，可在 Windows 系统中重新划分 E 盘，将之分为两个分区，把较大的磁盘分区用作“/”文件系统，较小的磁盘分区用作交换分区。当然，也可以在安装 Ubuntu Linux 系统时重新划分 E 盘。

如果磁盘分区容量较大，如大于 6GB，设置分区类型时应选择 Ext3 文件系统。由于 Ext2 文件系统需要周期地执行完整性检测，故磁盘分区较大时容易引起系统引导性能下降。

Linux 系统使用交换分区提供虚拟内存，因此，交换分区的设置非常重要。在确定交换分区的大小时，通常应以系统配置的内存作为参照。如果系统配置的内存小于 1GB，可以把交换分区的容量设为内存容量的两倍。如果内存大于等于 1GB，交换分区的大小可以等于物理内存的容量。但在一个 32 位的计算机系统中，交换分区的大小不能超过 2GB。如果确实需要使用更多的交换分区，可以设置多个交换分区，如果可能，最好把每个交换分区分布到不同的磁盘中。这种解决方案既克服了交换分区的容量限制，又可借以实现负载平衡，从而提高系统的性能。

如果是一个多用户系统，且系统配有大量的磁盘存储空间，最好划分多个磁盘分区，在每个磁盘分区创建一个单独的文件系统，如/usr、/var 和/home 等文件系统。如果磁盘的容量太大，BIOS 无法直接访问 1023 之外的柱面时，还可以增加单独的/boot 文件系统分区。但不能把/bin、/dev、/etc、/lib、/root 和/sbin 目录作为单独的文件系统分区，这些目录均应位于“/”文件系统分区中。

每个文件系统分区都有一个安装点，表示相应文件系统在整个 Linux 文件系统目录树状结构中的安装位置。除了单独的文件系统分区，分别用于存储各自的文件或数据之外，其他所有的文

件或数据均被存储在“/”文件系统分区中。

当需要并决定划分多个磁盘分区，以便创建单独的/usr、/var 和/home 等文件系统时，可以参考表 1-2 给出的磁盘分区的要求与建议（假定磁盘的容量为 10~12GB）。

表 1-2 磁盘分区的要求与建议

文件系统	最小容量要求	建议的容量分配	分区的文件系统类型
/	250 MB	2 GB	Ext3
/usr	1.5 GB	2~3 GB	Ext3
/tmp	50 MB	900 MB（最好不创建单独的文件系统分区）	Ext3
/var	500 MB	2~3 GB	Ext3
/home	2 GB	取决于用户数量以及用户应用与数据的空间需求	Ext3
/boot	100 MB	100 MB	Ext3

### 3. 安装方式

Ubuntu Linux 系统的安装方式极其灵活，可以采用不同的方法引导与安装系统：可以下载各种 ISO 映像文件，采用刻录的 CD/DVD 安装介质，安装一个基本的 Ubuntu Linux 系统；可以利用 GRUB 引导程序与内置硬盘（或 USB 移动盘）中的 ISO 映像文件，直接安装 Ubuntu Linux 系统；也可以在 Windows 系统中安装一个虚拟的 Linux 系统，等等。安装过程也比较简单，整个安装过程只需 7 个步骤。

### 4. 下载并准备 CD/DVD 安装介质

在开始安装 Ubuntu Linux 系统之前，首先需要选择一个就近的网站，按照计算机的 CPU 类型，选择下载一个特定版本的 Ubuntu 映像文件，如 ubuntu-8.10-desktop-i386.iso 或 ubuntu-8.10-dvd-i386.iso。然后，利用刻录工具，制作一个 Ubuntu Linux 系统的 CD/DVD 安装介质。

除了正常安装之外，CD/DVD 安装介质也可用作系统维护 CD/DVD，以便在系统无法正常启动或者忘记超级用户密码等情况下恢复 Ubuntu Linux 系统。

## 1.4.2 安装 Ubuntu Linux 系统

当上述一切准备工作完成之后，即可开始安装 Ubuntu Linux 系统。下面以 Ubuntu 8.10 Linux 系统的 CD/DVD 安装介质、512MB 内存和 8GB 虚拟磁盘分区的笔记本电脑为例，详述 Ubuntu Linux 系统的安装过程。

### 1. 利用 CD/DVD 安装介质引导系统

把刻录的 CD/DVD 安装介质插入光驱，加电启动计算机（注意，在引导系统之前应确保计算机的 BIOS 设置能够利用光驱引导系统）。在正式开始安装 Ubuntu Linux 系统之前，首先需要选择安装过程使用的语言环境。经过短暂的引导过程之后，系统将会出现图 1-9 所示的语言选择界面，其中给出了 Ubuntu Linux 系统支持的一系列语言供用户选择。要安装一个中文环境的 Ubuntu Linux 系统，可以使用箭头键选择“中文（简体）”，按 Enter 键继续。

### 2. 选择安装方式

选择语言环境之后，将会出现 Ubuntu Linux 系统安装程序的主界面，如图 1-10 所示。



## Ubuntu 权威指南

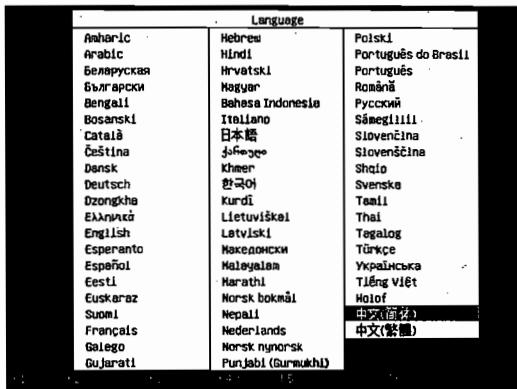


图 1-9 语言选择界面

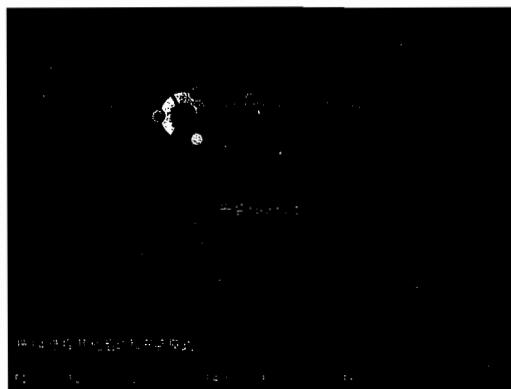


图 1-10 安装选择界面

Ubuntu Linux 系统 CD/DVD 安装程序的主界面主要提供下列 6 个菜单选项。

- 试用 Ubuntu 而不改变计算机中的任何内容 (T) ——表示利用 CD/DVD 安装介质启动系统，生成一个基于内存和硬盘原有系统的 Ubuntu Linux 运行环境。由此可见，也可以利用这个选项执行系统维护任务。Ubuntu Linux 系统运行环境提供大量的实用程序与维护工具，可用于修复各种系统问题。当现有的 Linux 系统无法正常启动时，可以选用这个选项修复系统问题。
- 安装 Ubuntu (I) ——采用图形界面安装程序，安装 Ubuntu Linux 系统。
- 在文本模式下安装 Ubuntu (I) ——采用文本模式安装 Ubuntu Linux 系统。
- 检查光盘是否损坏 (C) ——用于测试 CD/DVD 安装介质。如果 CD/DVD 的刻录质量与完整性有问题，将会导致系统安装的失败。为了避免安装过程出错，可在安装前进行验证。
- 内存测试 (M) ——用于测试系统内存，以避免由于内存问题导致 Linux 系统出现异常或故障。要停止内存测试，重新引导系统，可在任何时间按下 Esc 键。
- 从第一块硬盘启动 (B) ——从 CD/DVD 安装介质引导系统之后，如果又决定从系统硬盘中引导系统，可以选择此选项。

此时，可以利用箭头键，选择“安装 Ubuntu (I)”并按下 Enter 键，以图形界面方式安装 Ubuntu Linux 系统。

### 3. 选择语言环境

当出现“欢迎”界面时，需要再次选择安装过程使用的语言，被选中的语言也将成为最终系统的默认语言环境。如果继续使用“中文（简体）”，可以单击“前进”按钮继续，如图 1-11 所示。

### 4. 时区选择

在图 1-12 所示的“你在什么地方”界面中，可以设置系统的时区。Ubuntu 安装程序提供两种时区选择方法：世界地图与下拉菜单。

- 利用世界地图选择时区。根据自己所在的区域，上下左右移动鼠标，选择一个距离自己最近的城市黄点，被选中的黄点将会闪烁，同时在下拉框中显示选定的城市，以城市的地理位置表示用户选中的时区。
- 利用“所选城市”下拉菜单选择城市。选择一个距离自己最近的城市，如“Shanghai”（表示北京时间 CST，即 GMT+8:00）。

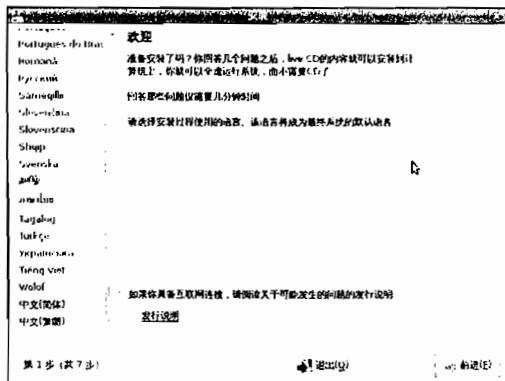


图 1-11 “欢迎”界面

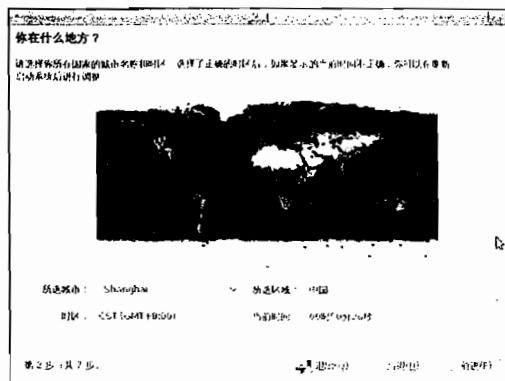


图 1-12 “你在什么地方”界面

选定城市也就选定了时区，然后单击“前进”按钮继续。注意，安装过程只能设置时区，不能设置系统的日期与时间。如果系统显示的日期与时间不正确，可在安装结束，注册到系统之后再做调整。

## 5. 键盘配置

在“键盘布局”界面左边的窗口中，可从 Ubuntu Linux 系统支持的不同键盘类型中，选择适当的键盘类型，如选择“USA”（默认为“China”），然后单击“前进”按钮，如图 1-13 所示。

注意，在选定键盘类型之后，可以利用按键测试选定的键盘是否正确。

## 6. 磁盘分区

Ubuntu Linux 系统可以被安装在任何空闲的磁盘或磁盘分区中。在“预备磁盘空间”界面中，安装程序会给出安装前后磁盘的使用与分配情况。如果想在整个磁盘上单独安装 Ubuntu Linux 系统，安装时可以选择“向导 - 使用整个磁盘”，由安装程序引导用户一步一步地完成安装过程。对于有经验的用户，也可以选择“手动”安装方式，自己划分磁盘分区，定制文件系统布局，如图 1-14 所示。

选择“手动”安装方式时，可以自己定义磁盘分区的布局，创建适当数量的磁盘分区，以便用于/home、/usr、/var 或/boot 等文件系统。如果仍想在整个磁盘上单独安装一个 Ubuntu Linux 系统，或者在 Windows 系统的 E 盘中安装一个 Ubuntu Linux 系统，需要自己对磁盘进行分区。为此，可以单击“新的分区表”按钮，此时系统将会给出一个警告信息，说明此举将会删除系统中现有的磁盘分区。如果没有疑义，单击“继续”按钮，如图 1-15 所示。

使用鼠标选中“空闲的空间”，然后单击“新的分区”按钮，对磁盘进行分区，安装程序将会弹出一个“创建新分区”界面。在“新建分区容量”栏中，输入以 MB 为单位的磁盘分区容量；在“用于”栏中，选择磁盘分区的文件系统类型，如“Ext3 日志文件系统”、“Ext2 文件系统”或“交换空间”等。如果选择的磁盘分区用作文件系统，尚需在“挂载点”栏中输入文件系统的安装点。在我们的例子中，可以把扣除交换空间之外的全部存储空间（7 560MB）用作 Ext3 类型的“/”文件系统，如图 1-16 所示。

单击“确定”按钮后，安装程序将会划出一个用作“/”文件系统的磁盘分区，同时重新计算并更新“空闲的空间”容量。此时，可以再次使用鼠标选中“空闲的空间”，单击“新的分区”按钮，对剩余的空闲磁盘空间进行分区。同样，安装程序又会弹出一个“创建新分区”界面。在“新建分区容量”栏中，输入剩余的磁盘分区容量（1 028MB），在“用于”栏中，选择“交换空间”



作为磁盘分区类型，如图 1-17 所示。

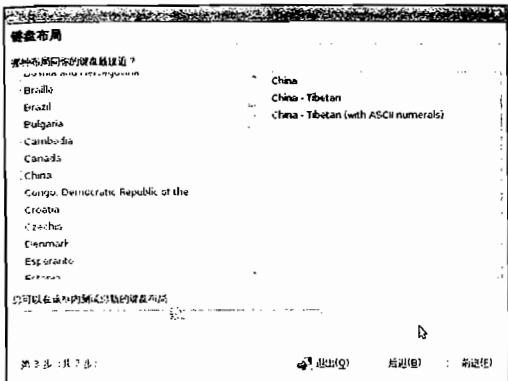


图 1-13 “键盘布局”界面

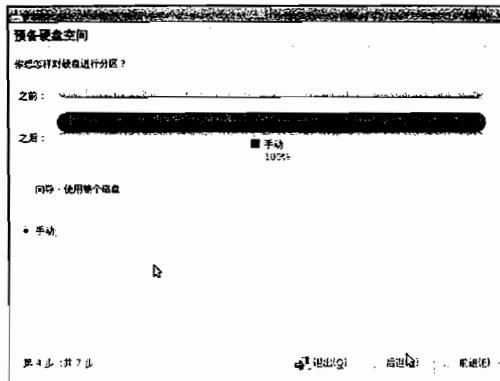


图 1-14 “预备磁盘空间”界面

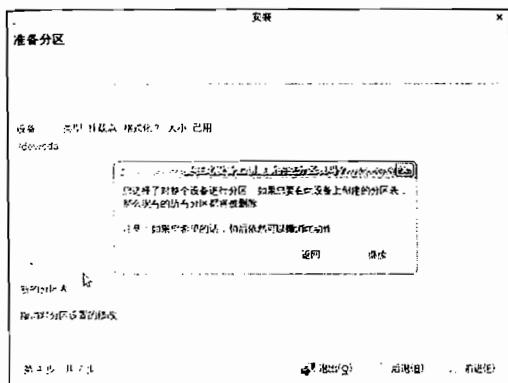


图 1-15 “准备分区”界面

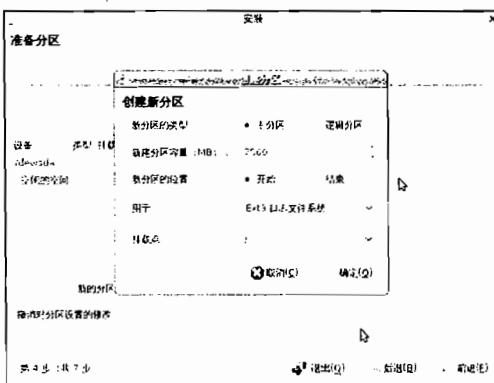


图 1-16 创建新分区界面 (1)

单击“确定”按钮后，即可创建一个交换分区。至此，磁盘分区的任务已经全部完成，单击“前进”按钮继续。分区后的最终结果如图 1-18 所示。

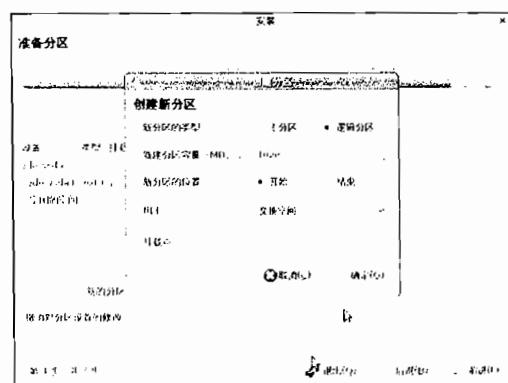


图 1-17 创建新分区界面 (2)

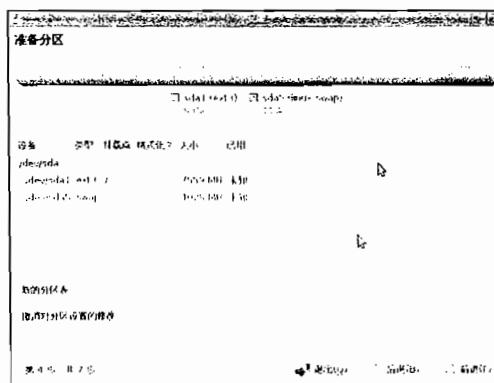


图 1-18 分区后的结果

如果是在 Windows 系统预留的 E 盘中安装 Ubuntu Linux 系统，“准备分区”界面的“设备”栏中将会列出 Windows 系统 C 盘和 D 盘占用的磁盘分区，如 /dev/sda1（“类型”一栏为 ntfs 或 fat32 等），以及空闲的 E 盘分区。此时，可以保持 C 盘和 D 盘等磁盘分区不变，把 E 盘划分为两个磁盘分区，分别用作“/”文件系统与交换分区，其步骤如前所述。

### 7. 创建用户账号

Ubuntu 建议用户总是使用自己的账号信息而非超级用户 root 注册 Linux 系统，以确保系统的安全。在随后出现的“你的身份是什么”界面中，可为自己创建一个普通用户账号。输入用户的全名、有效的注册用户名后，输入并确认自己的密码。此外，还需要设置系统的主机名。如果希望在系统启动之后能够自动注册，直接进入 GNOME 桌面环境，可以勾选“自动登录”复选框。最后单击“前进”按钮，如图 1-19 所示。

注意，除了同名的用户组，此时创建的第一个用户也会被自动添加到 admin 用户组中，因而拥有 admin 用户组成员的访问权限，能够执行一定的系统管理与维护任务。因此，当需要利用超级用户的访问权限执行系统管理任务时，可在命令行界面中使用 sudo 命令。即，在实际运行的命令前面增加一个 sudo 前缀，在提示输入密码时，输入此时设置的密码（为了便于叙述，以后把安装 Ubuntu Linux 系统时创建的第一个用户的密码简称作 sudo 密码）。

### 8. 恢复数据文件

如果计算机已经安装了 Windows 系统，或者先前已经安装过 Ubuntu Linux 系统，安装程序将会显示一个“Migrate documents and settings”界面，提供一个恢复用户主目录中的文档和设置的机会。如果想要导入或恢复其中的数据，可以单击相应的复选框。否则，直接单击“前进”按钮，进入下一步。

### 9. 准备安装

在“准备安装”界面中，安装程序汇总了先前所做的一系列选择与设置，如果没有问题，单击“安装”按钮开始正式安装 Ubuntu Linux 系统。如果设置有误，可以单击“后退”按钮，返回先前的界面重新设置，如图 1-20 所示。

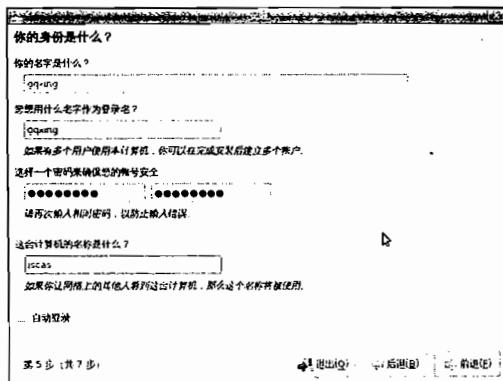


图 1-19 创建用户界面

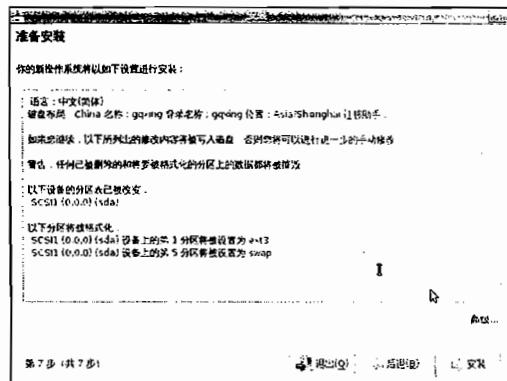


图 1-20 “准备安装”界面

实际上，在此之前的每一个设置步骤中，如果设置有误，均可单击“后退”按钮，返回先前的界面重新设置。



## 10. 安装软件包

至此, Ubuntu 安装程序开始安装软件, 同时在屏幕上显示整个安装的进度。安装程序首先会格式化磁盘分区(如果先前勾选了“格式化”复选框), 创建文件系统, 计算文件占用的磁盘空间, 复制文件, 安装软件包, 设置系统, 配置 apt, 检测镜像网站, 设定软件源, 设置时区, 与网络时间服务器同步时间, 设置键盘, 导入文档和设置(如果先前勾选了“导入”复选框), 检测硬件, 加载 USB 模块, 设置硬件, 以及删除临时文件, 等等, 如图 1-21 所示。

安装的速度取决于计算机的配置和网络的速度, 这一安装过程将会持续较长的时间, 但中间无需用户干预。安装完成之后, 安装程序将会执行安装后的软件配置, 把 GRUB 引导程序复制到磁盘中, 最后显示图 1-22 所示的对话框, 表示整个安装过程已经完成。

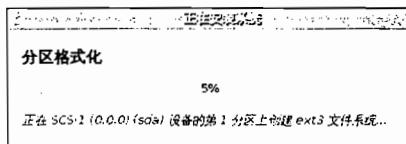


图 1-21 安装软件包界面

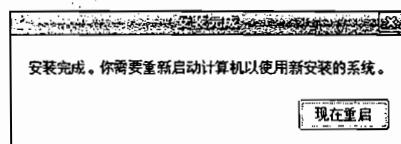


图 1-22 安装完成之后重新引导系统界面

单击“现在重启”按钮, 重新启动系统。在重启之前, 系统将会弹出 CD/DVD, 提示用户取出 CD/DVD 安装介质, 然后按 Enter 键, 如图 1-23 所示。

成功地重新启动之后, 系统将会显示图 1-24 所示的注册界面。此时, 可以使用先前创建的用户名与密码注册, 访问 Ubuntu Linux 系统。



图 1-23 重启系统界面



图 1-24 系统注册界面

### 1.4.3 安装后的软件维护与更新

#### 1. 安装附加的软件包

采用 CD/DVD 安装介质等方式安装的 Ubuntu Linux 系统只是一个基本系统, 可能还无法完全满足用户的需求。为了安装其他附加的软件包, 如 Apache 服务器、MySQL 数据库及 Samba 服务器等, 可以采用 apt-get、aptitude 及 synaptic 等软件管理工具。如果需要补充安装 GNOME 桌面环境支持的图形界面软件包, 比较简单的方法是在 GNOME 桌面中选择“应用程序→添加/删除”菜单, 从中选择期望安装的软件包。有关软件的安装与维护, 详见第 12 章。

## 2. 更新系统

在系统支持期间，Ubuntu 经常会发布更新软件包，而更新软件包通常会增加新的功能特性，以修正软件中的错误或安全漏洞，提高软件的可靠性。为了确保系统的安全，应定期地更新系统。

更新系统时，可以采用 apt-get、aptitude 以及 synaptic 软件管理工具，也可以从 GNOME 桌面中选择“系统→系统管理→更新软件包”，或者选择“系统→系统管理→新立得软件包管理器”菜单，安装更新软件包。有关软件的更新与维护，详见第 12 章。

此外，Ubuntu Linux 系统还以后台进程的方式，自动运行软件更新工具 update-notifier 和 update-manager（同“系统→系统管理→更新软件包”菜单），检测系统配置的软件仓库中是否存在可用的软件包。当存在更新软件包时，GNOME 信息公告区将会显示一个软件更新图标，同时在桌面上弹出一个软件更新消息框，提醒用户更新自己的系统（详见第 12 章）。

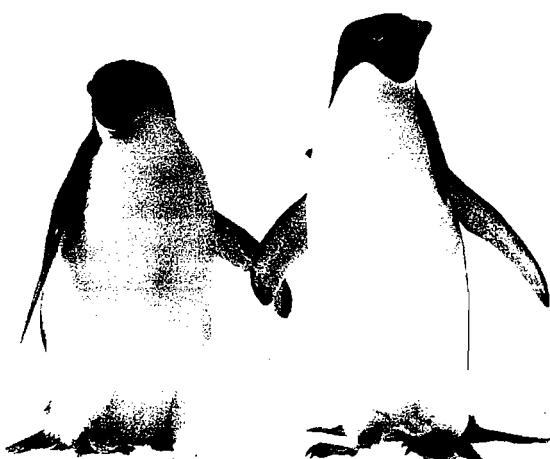


## 第2章

### GNOME 桌面环境

本章主要介绍 Ubuntu Linux 系统的 GNOME 桌面环境,简述 GNOME 桌面环境的主要组成部分,使读者对 GNOME 桌面环境有一个全面的了解。其内容主要包括:

- GNOME 桌面环境概述;
- GNOME 桌面环境浏览;
- 应用程序菜单;
- 位置菜单;
- 系统菜单;
- 使用移动存储设备;
- 定制 GNOME 桌面环境。



在不同发行品牌或不同版本的 Linux 系统中，桌面环境之间的差别比较大，但 Linux 系统的基本设计思想，尤其是作为底层支持的系统命令，除了新增加的功能之外，则很少有变化，这也是许多学习和使用 Linux 系统的人喜欢使用命令行界面的主要原因之一。因此，在学习 Linux 系统时，最好以熟悉命令行界面为主，把桌面环境作为一种辅助的手段。但对于初学 Linux 系统的新用户而言，采用可视化的图形界面（而不是输入命令）执行日常的处理任务，可能比较容易一些。因此，初学者不妨先从熟悉 GNOME/KDE 桌面环境开始，逐步转入 Linux 系统命令行界面。

桌面环境是一个介于用户与操作系统之间的中间层，有助于用户完成诸如管理文件、运行程序等操作。为了适应 Windows 系统用户，Linux 系统从纯粹的命令行界面逐步发展到提供 GNOME (GNU Network Object Model Environment) 和 KDE (K Desktop Environment) 等桌面环境。Ubuntu Linux 系统的默认桌面环境是 GNOME，但也支持 KDE 桌面环境，要想使用 KDE 桌面环境，可在 Ubuntu Linux 系统中安装 KDE 软件包，也可以安装一个纯 KDE 桌面环境的 Kubuntu Linux 系统。本章主要介绍 GNOME 桌面环境。

## 2.1 GNOME 桌面环境概述

### 2.1.1 GNOME 注册界面

在成功地启动系统之后，控制台终端上将会出现一个 Ubuntu Linux 系统默认的 GNOME 注册界面，提示用户注册、访问 Linux 系统。每次注册时，即开启一个新的注册会话（从系统注册开始，直至从系统中注销的整个系统访问过程），如图 2-1 所示。

在 Linux 系统中，超级用户 root 拥有无限的权力，如果使用不当，可能会损害或危及系统的安全，故 Ubuntu Linux 系统通常不允许用户使用 root 注册。安装后初次访问 Ubuntu Linux 系统时，只能使用安装系统时创建的用户名与密码注册，之后才能使用新增的用户名与密码注册。当然，经过一定的设置后，也可以使用超级用户 root 注册（详见第 13 章）。

Ubuntu Linux 系统的注册界面很简单，可以分为 4 个不同的功能区域：注册区、选项菜单区、信息区及背景主题区。

正中的注册区提供注册提示，提示用户输入用户名和密码，以进入 Linux 系统，访问 GNOME 桌面环境。

左下角的选项菜单区至少包含 7 个菜单项：“选择语言”用于选择 GNOME 会话期间使用的语言，如上一次注册时使用的语言、系统默认的语言，或者直接选择其他语言等；“选择会话”用于选用上一次注册时的会话环境、GNOME 桌面、安全的远程连接、GNOME 安全模式，以及终端安全模式等会话模式；“通过 XDMCP 远程登录”用于连接远程主机的显示管理器，以图形界

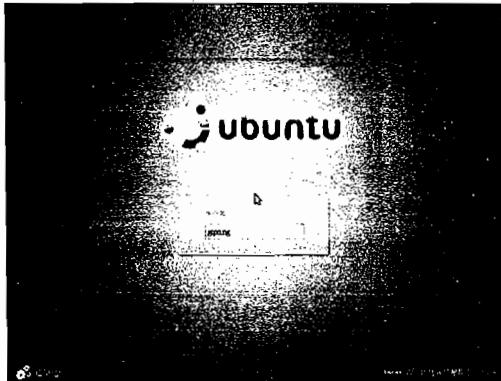


图 2-1 Ubuntu Linux 系统的注册界面



面的方式注册、访问远程系统；“重新启动”与“关机”可用于重新启动系统或直接关机，如果希望立即重启或关闭系统，无须注册，即可在此处选择相应的选项；“挂起”与“休眠”用于临时挂起系统，或使系统暂时处于休眠状态。

右下角的信息区（正中方框的右上角）用于显示系统的主机名、日期与时间。

背景主题区是整个背景，系统管理员可以根据个人的爱好选用不同的背景。

### 2.1.2 GNOME 桌面环境

Ubuntu GNOME 桌面环境旨在提供一种可视化的图形界面，使用户能够通过图标、下拉菜单及窗口等方式访问系统提供的功能。在 GNOME 注册界面中，一旦输入用户名并按下 Enter 键，系统将会提示用户输入密码。如果输入的用户名与密码是正确的，即可成功地注册，进入 Ubuntu Linux 系统的 GNOME 默认桌面环境，如图 2-2 所示。



图 2-2 Ubuntu Linux 系统的 GNOME 桌面

一旦注册到系统，GNOME 桌面即成为用户的主要工作环境。通常情况下，大多数处理任务都可以在桌面环境中完成。Ubuntu GNOME 桌面环境可以分为下列 3 个区域：

- 菜单面板；
- 桌面区；
- 窗口面板。

菜单面板位于 GNOME 桌面顶部的长条形区域，其中包含 3 个菜单和 3 个默认的应用启动图标，使用户能够快速地启动相应应用程序。菜单面板的最右边是“信息公告区”，提供软件更新、输入方法、网络控制、音量控制、当前注册的用户名、系统日期与时间以及系统控制等按钮与信息。

桌面区指的是菜单面板与窗口面板之间的整个屏幕区域。桌面区用于存放用户定义的应用程序启动器图标，存放安装的文件系统图标（如内置的 Windows 分区、CD/DVD、USB 移动硬盘以及 U 盘）等。

窗口面板位于屏幕的底部，其中最左边是一个“显示桌面”图标，紧接着可以是一系列表示用户当前打开的应用窗口图标，右边存在 2(4) 个工作区切换开关，最后是一个“回收站”图标。

屏幕顶部与底部的两个菜单面板可用于存放经常使用的项目，包括菜单、快速启动程序、按钮（图标）以及信息公告区等。Ubuntu GNOME 桌面环境的布局和内容可以定制，用户可根据自己的需要，进行必要的调整，详见第 2.7 节。

## 2.2 GNOME 桌面环境浏览

### 2.2.1 GNOME 菜单面板

#### 1. 主菜单

在 GNOME 桌面环境中，最常用的是菜单面板。菜单面板通常包含 3 个主要菜单：应用程序、位置和系统，用于访问 Ubuntu Linux 系统提供的各种功能。

- 应用程序——“应用程序”菜单包含各种菜单项，用于启动应用程序。GNOME 把用户能够执行的程序分类组织成一级或二级菜单，如“办公”、“附件”、“互联网”、“图形”、“影音”、“游戏”以及“添加/删除”软件等。“应用程序”菜单类似于 Microsoft Windows 系统的“开始”菜单。如果需要，也可以采用拖放的形式，在桌面中创建相应的快捷键。
- 位置——“位置”菜单用于调用文件浏览器，访问文件系统，包括主文件夹、定制的目录列表、安装的文件系统（如磁盘分区、USB 移动盘以及 CD/DVD 等）、计算机、网络与服务器访问、文件搜索功能及最近访问的文档等。
- 系统——“系统”菜单用于配置、管理与维护系统，其中包含“首选项（用于设置输入方法、窗口属性、鼠标、键盘与键盘快捷键、默认打印机、屏幕保护程序、屏幕分辨率、桌面外观、网络代理、远程桌面、音响效果以及主菜单等）”、“系统管理（用于设置打印机、系统服务、系统日期与时间、网络、用户与用户组、硬件驱动程序、软件包管理器、注册界面，以及查询系统日志，查询当前进程以及监控系统状态，安装、更新与升级软件包等）”、“帮助和支持”、锁屏、GNOME 与 Ubuntu 简介、“注销”以及“关机”等。

GNOME 桌面将许多常用的处理任务分类组织成 3 个菜单，如图 2-3 所示。用户可以从 3 个主要菜单开始，按照菜单的级联层次，逐层向下检索各级子菜单的菜单项，直到找到自己需要的处理任务，然后选择执行任何一项具体的处理任务。



图 2-3 Ubuntu 8.10 菜单面板



## 2. 快速启动区

除了 3 个菜单选项之外，菜单面板的中间部分被称作快速启动区，其中包含下列 3 个默认的通用软件图标（或称按钮），能够快速启动相应的应用程序：

- Firefox 网络浏览器；
- Evolution 电子邮件客户端；
- Ubuntu 帮助信息。

除了上述 3 个基本图标之外，如有必要，可以把一部分频繁调用的程序，以图标形式置于快速启动区，以便单击相应的图标，即可绕过菜单，快速启动相应的程序。如果桌面区存在已经创建的应用程序启动器，也可以选中启动器图标，将其拖放至快速启动区。

如果想在快速启动区中增加其他应用程序图标，可以右击菜单面板的空白区域，从上下文菜单中选择“添加到面板”菜单项，然后从弹出的“添加到面板”窗口中选择期望增加的功能，最后单击“添加”按钮，如图 2-4 所示。

## 3. 信息公告区

菜单面板的右边（屏幕右上角）是信息公告区。在 Ubuntu 8.10 的信息公告区中，从右到左，信息公告区通常依次为“用户切换与关机”、“时钟”、“音量控制”、“网络设置”及“输入方法”等图标，这些图标通常是由 Applet 实现的，能够不断地被更新，以提醒用户注意。在系统的运行过程中，当出现一定的事件，需要通知用户，由用户给予必要的干预时，通常也会增加额外的图标，如软件更新图标（红色下箭头）等。通常，只需单击图标即可启动相应的程序，也可以右击图标，从菜单中选择适当的功能。

“用户切换与关机”图标具有综合性的功能，通常用于显示当前注册的用户，单击这个图标将会弹出一个可用的用户列表，选则其中的任何一个用户名，即可暂时退出当前用户的注册会话，返回到注册界面，引导用户输入密码，以选定的用户身份重新注册到系统。之后还可以利用这种方法，在注册的用户会话之间切换，以不同的用户身份访问系统。“用户切换与关机”图标还包括“锁住屏幕”、“注销（系统中译为‘登出’）”、“挂起”、“休眠”、“重启”和“关闭”等菜单项，如图 2-5 所示。注意，当单击其中的任何一个菜单项时，将会立即执行，中间没有挽回的机会，这一点与 Ubuntu 8.04 不同。例如，如果选择“关机”菜单，系统将会立即开始关机。一旦关闭系统，即可切断计算机的电源。如果希望重新启动系统，可以选择“重启”。如果选择“挂起”或“休眠”，系统将会进入临时关机状态。一旦需要再次访问系统时，按下电源开关，即可进入先前的窗口，继续工作。

“时钟”图标用于显示系统的日期与时间。单击“时钟”图标将会弹出当月的日历，通过单击年份或月份左右的箭头键，可以分别显示其他年份或月份的日历。再次单击“时钟”图标将会恢复日期和时间显示。右击“时钟”图标时将会弹出一个上下文菜单，其中，“首选项”菜单项可用于修改显示的内容，以及日期与时间的显示形式，“调整日期和时间”菜单项允许拥有访问权限的用户调整系统的日期与时间，如图 2-6 所示。

利用“音量控制”图标，可以调整音响的音量。双击“音量控制”图标，还可以在新弹出的“音量控制”界面中，细调每个音响设备的音量。也可以利用音响设备下方的“静音/非静音”切换按钮，设置音响设备的静音状态等（参见图 2-7）。

右击“网络”图标，可以控制是否启用网络，查询以及配置网络连接等。“电源状态”图标（如果存在）是一个电源状态指示器，用于显示电源的蓄电量。

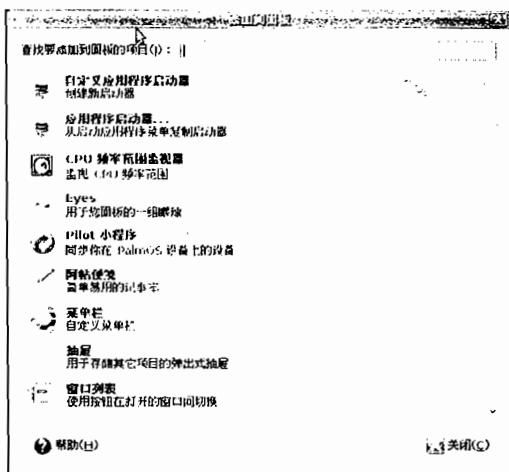


图 2-4 菜单面板添加图标



图 2-5 “用户切换与关机”图标

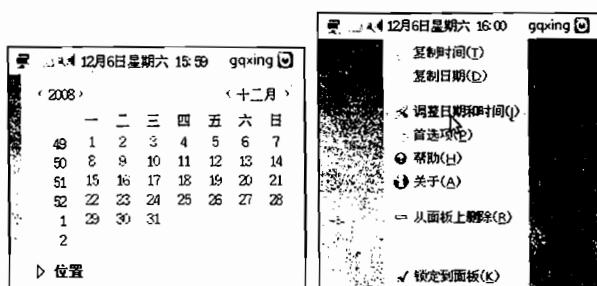


图 2-6 日期时间调整

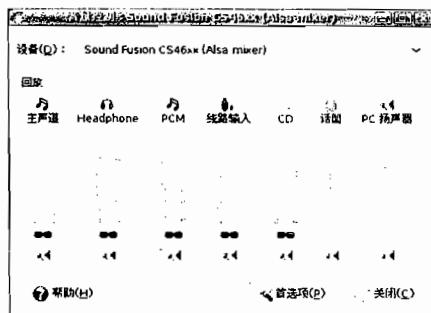


图 2-7 音量控制界面

“输入方法选择”图标（小键盘）用于选择输入方法，如英文或简体中文。右击小键盘图标，从弹出的上文菜单中选择“SCIM 设置”，可以查询、设置或增加输入方法。之后，图标将会变为表示相应输入方法的一个汉字，如“智”表示“智能拼音”。无论何时（如运行 Office 办公软件或文本编辑器，以及打开终端窗口时），如果需要输入中文，可以单击菜单面板信息公告区中的小键盘或输入方法图标，从弹出的菜单中选择“智能拼音”或“五笔字型”等输入方法，如图 2-8 所示。

需要输入中文时，可以同时按下 Ctrl 与空格键，桌面右下角将会出现一个中文输入方法条形框，此时即可输入中文，默认的输入方法为“智能拼音”。如果想要换用其他输入方法，可以单击输入方法条形框中的输入方法，如从弹出的菜单中选择“五笔字型”等其他输入方法。此外，同时按下 Ctrl 与空格键，还可以在中英文输入方法之间切换，如图 2-9 所示。

#### 4. 面板上下文菜单

在菜单面板（包括窗口面板）的空白处单击鼠标右键，可以弹出一个与菜单面板有关的上下文菜单，通过选择其中的菜单项，可以在面板中增加应用程序的快速启动图标，从而能够快速调用相应的处理程序。此外，还可以设置面板的属性，如面板的方向与浮动等，如图 2-10 所示。

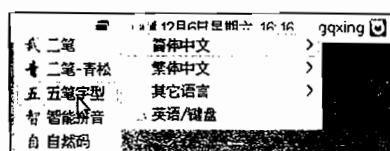


图 2-8 选择中文输入方法 (1)

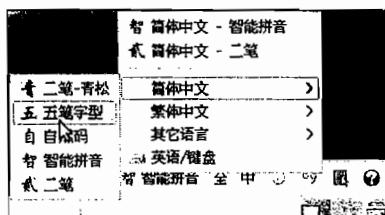


图 2-9 选择中文输入方法(2)

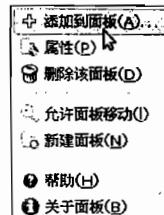


图 2-10 面板上下文菜单

表 2-1 列出了菜单面板上下文菜单提供的菜单项。

表 2-1

菜单面板的上下文菜单

菜单项	基本功能
添加到面板	在菜单面板上创建应用程序快速启动图标,如“搜索文件”、“网络监视器”、“系统监视器”、“Eyes(眼睛)”与“气象报告”等
属性	定义菜单面板的属性,如面板的方向与大小(像素),以及是否自动隐藏等
删除该面板	删除菜单面板
允许面板移动/锁定面板位置	表示是否允许用户上下左右移动面板
新建面板	创建一个新的面板,如在桌面的左右创建新的面板,以丰富自己的桌面环境等
帮助	介绍面板的基本概念与使用方法,如菜单、按钮(图标)与快速启动的简单说明与用法
关于面板	说明 GNOME 面板程序的版本等

## 2.2.2 GNOME 桌面区

### 1. 图标

GNOME 桌面区是一个存储空间,也是用户主目录下的一个子目录,可用于存放启动器图标、文件或目录等。最初,GNOME 桌面区是空的,当插入移动存储介质,如 CD/DVD、USB 移动盘或 U 盘时,GNOME 桌面区中将会自动出现表示相应介质的图标。

此外,如果计算机中还装有 Windows 系统,从“位置”菜单选择“Local Disk(C:盘的卷标名)”等菜单项,桌面区也会出现表示 Windows 系统所在磁盘分区的图标,说明系统已经把 Windows 文件系统自动安装到 Linux 系统的/media 目录中。当安装了 Windows 文件系统,接着又插入一个 1GB 的 U 盘时,桌面区如图 2-11 所示。

在 GNOME 桌面中,使用鼠标左键单击图标,即可激活图标表示的功能。如果使用鼠标右键单击图标,将会显示图标对应的上下文菜单,列出相应的功能选项以及能够执行的各项处理任务。

### 2. 桌面区上下文菜单

在 GNOME 桌面的空白处单击鼠标右键,可以弹出一个与桌面区有关的上下文菜单,通过选择其中的菜单项,可以执行相应的处理任务,如图 2-12 所示。

表 2-2 列出了桌面区上下文菜单提供的菜单项。

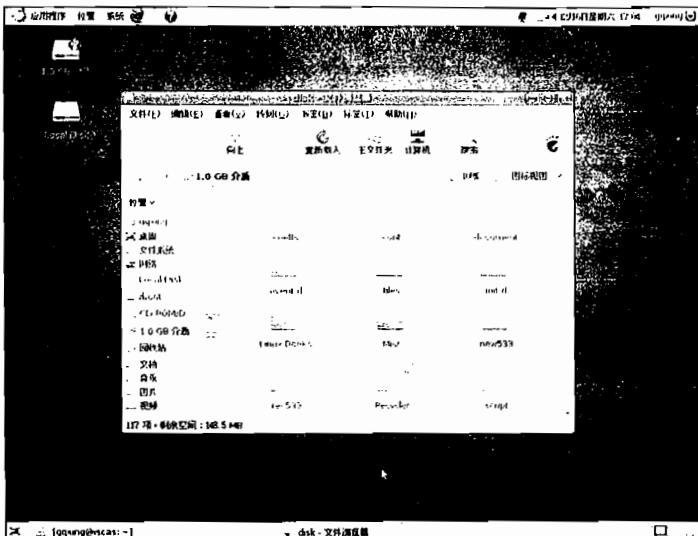


图 2-11 桌面区图标

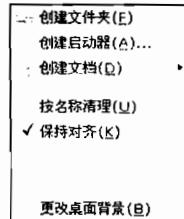


图 2-12 桌面区上下文菜单

表 2-2

桌面区上下文菜单

菜单项	基本功能
创建文件夹	在桌面上创建一个“未命名文件夹”（可以重新命名）
创建启动器	弹出一个“创建启动器”的对话框，以便在桌面上创建一个能够快速启动的应用启动器图标
创建文档	在桌面上创建一个名为“新文件”的空文件（可以重新命名）
按名称清理	按照字母顺序排列桌面上的图标
保持对齐	按栅格形式整齐排列桌面上的图标
粘贴	用于粘贴先前复制的数据副本
更改桌面背景	弹出一个能够修改桌面背景的对话框，使用户能够选择不同的图像或照片，作为桌面背景

除了菜单面板之外，也可以把桌面区作为一个快速启动区，创建应用程序启动器，把应用程序快速启动图标置于桌面区。通过双击桌面区存放的应用程序图标，即可绕过应用程序菜单，快速启动相应的应用程序。

当从桌面区上下文菜单中选择“创建启动器”时，将会弹出一个“创建启动器”的对话框，可用于创建一个快速启动器，如图 2-13 所示。

从“类型”下拉菜单中选择启动器的类型（如“应用程序”），在“名称”栏目中输入启动器图标的名字（如“文件浏览器”），在“命令”栏目中输入或单击“浏览”按钮选择实际运行的命令（如“nautilus”），如果需要，也可以在“注释”栏目中给出简单的注释信息。最后单击“确定”按钮，在桌面区创建一个启动器图标，单击这个图标即可快速启动文件浏览器，如图 2-14 所示。

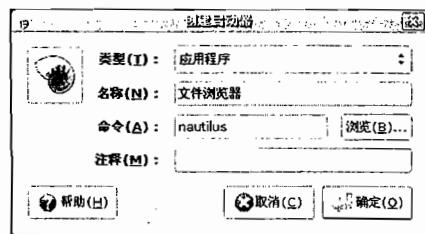


图 2-13 创建启动器对话框



如果选择的启动器类型是“应用程序”，每当双击启动器图标时，即可快速地启动相应的应用程序。如果选择的是“终端中的应用程序”，双击启动器图标时将会打开一个新的终端窗口，并在其中运行指定的命令。如果选择的启动器类型是“位置”，且命令字段给定的是一个文本文件，双击启动器图标时将会调用文本编辑器，打开相应的文件。如果命令字段给定的是目录而不是文件，GNOME 将会调用文件浏览器，打开相应的目录。如果命令字段给定的是脚本文件，GNOME 将会询问用户在终端窗口中运行，还是显示文件的内容。

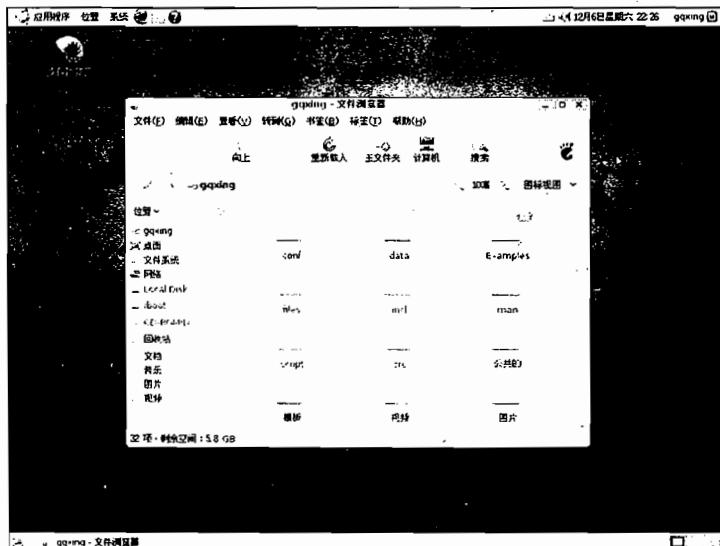


图 2-14 文件浏览器启动器图标及其应用

### 2.2.3 GNOME 窗口面板

GNOME 桌面底部的长条形区域称作窗口面板。窗口面板的主要作用是显示当前打开的应用窗口，切换桌面工作区，切换当前打开的应用窗口，以及显示或隐藏 GNOME 桌面区等。窗口面板通常包含下列 4 个组件（如图 2-15 所示）。



图 2-15 GNOME 窗口面板

- 桌面显示/隐藏按钮；
- 窗口列表；
- 工作区切换开关；
- 回收站图标。

屏幕左下角的“桌面显示/隐藏”按钮用于展示整个桌面区，隐藏桌面上已经打开的所有活动窗口。当桌面上打开大量的窗口时，这是一项很有用的功能。再次单击“桌面显示/隐藏”按钮时，又可恢复原来打开的窗口。

当前打开的窗口列表位于窗口面板的中部。打开任何一个应用程序时，除了在桌面上显

示一个活动的窗口外，还会在窗口面板中部的“窗口列表”中显示一个窗口图标。同 Windows 系统类似，单击任何一个应用窗口图标，即可激活相应应用窗口，并将其置于所有窗口的最上面。如果单击窗口右上角的“最小化”按钮，窗口将会从桌面上消失，但“窗口列表”中的窗口图标继续存在。单击“窗口列表”中的窗口图标，可以在其他所有窗口的上层恢复窗口显示，使窗口处于活动状态。另外，也可以使用组合键 Alt+Tab，切换至已经打开的任何应用窗口。

工作区切换开关位于窗口面板的右边。每个工作区都拥有一个单独的 GNOME 桌面，包括菜单面板、桌面区和窗口面板等。所有工作区的菜单面板和背景主题都是相同的。在每个工作区中，可以运行不同的应用程序。工作区切换开关使用户能够在工作区之间相互切换。

回收站图标位于窗口面板的最右边，用于缓存桌面环境中删除的文件等。

### 1. 窗口列表

窗口列表（或称任务栏）用于展示当前打开的所有窗口，使用户能够随时切换活动的窗口。GNOME 允许同时打开多个并发的窗口，窗口列表中将会根据打开的顺序，依次排列当前运行的每个窗口。当单击窗口列表中的任何一个窗口标识时，被选中的窗口就会变成活动的窗口。但如果单击的是一个活动的窗口，窗口将会消失。

### 2. 虚拟工作区

GNOME 桌面提供了 2 (4) 个虚拟工作区。除了当前使用的 GNOME 桌面，还有 1 (3) 个虚拟工作区桌面可供随时使用。当需要在其他虚拟工作区桌面中运行不同的程序时，可以单击工作区切换开关，直接切换到相应的虚拟 GNOME 桌面。当需要在不同的虚拟工作区桌面并发地运行不同的应用时，工作区切换开关能够使 GNOME 工作区桌面的切换变得简单快捷。

### 3. 回收站

回收站用于缓存删除的文件等。在桌面环境中，用户删除的任何文件、目录、图标等将会被自动移至回收站（命令行中删除的文件不在此列）。为了永久性地删除文件，或者定时清理回收站，可以右击“回收站”图标，在弹出的“回收站-文件浏览器”窗口中，选择删除选定的文件或者清空回收站。如果想在删除文件时能够绕过回收站，彻底删除文件，可先按住 Shift 键，然后再删除文件。注意，在删除大量文件之后，如果不及时清理回收站，将会挤占可用的存储空间。

## 2.3 应用程序菜单

在 GNOME 桌面环境中，常规的处理功能均已被纳入“应用程序”主菜单中。单击“应用程序”菜单时，将会分类列出各种内置实用程序的子菜单。单击任何一个子菜单将会列出一组可用的实用程序（或下一级子菜单），如图 2-16 所示。注意，如果系统里也安装了 KDE 桌面环境，KDE 的内置应用程序和实用程序也会被列在子菜单中，因而也可以在 GNOME 桌面环境中对其进行访问。

应用程序菜单通常包含办公、附件、互联网、图形、影音、游戏以及添加/删除软件等子菜单。



下面分别介绍其中的部分主要应用程序。

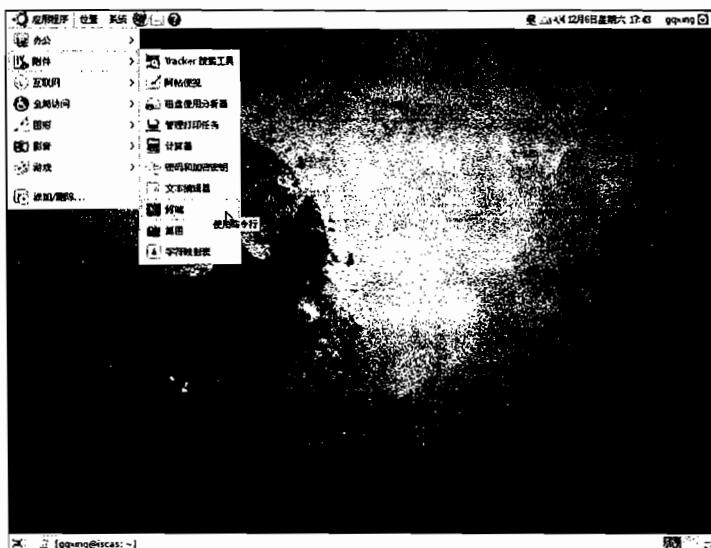


图 2-16 应用程序主菜单

### 2.3.1 办公

GNOME 提供一整套免费的办公应用程序 OpenOffice.org，使用户能够创建基于开放标准格式的文档、电子表格以及演示文稿等，如 OpenDocument、Rich Text 和 HTML 等。如有必要，也可以读写和编辑 Microsoft Office Word、Excel 和 PowerPoint 等格式的文档，并具有高度的兼容性。此外，还可以导出 PDF 格式的文件，而无需附加的软件支持。GNOME 还提供字典、Evolution 邮件和日历以及项目管理等工具。

#### 1. OpenOffice.org

OpenOffice.org 的功能类似于 Windows 系统中的 Office。其中 3 个最常用的 OpenOffice 应用是文字处理（Writer）、演示文稿（Impress）和电子表格（Calc）。

- Writer 类似于 Microsoft Office Word，可用于创建、编写文档、文本和网页。但不同的是，Writer 能够直接创建 PDF 文件。
- Impress 类似于 Microsoft Office PowerPoint，可用于创建、编写演示文稿以及幻灯片。同 Writer 一样，Impress 也能够直接创建 PDF 文件。
- Calc 类似于 Microsoft Office Excel，可用于创建、编写电子表格，计算、分析数据信息。Calc 也可以直接生成 PDF 文档。

同 Microsoft Office 一样，在任何一个 OpenOffice.org 应用中建立的对象都可以被导入另一个应用中。例如，电子表单可以被导入 Writer 文档，或者作为 Impress 幻灯片的一个组成部分；同样，Impress 中的图像也可以被导入 Writer 文档。此外，由于 OpenOffice.org 的文件都是与处理文件的应用相关联的，故双击文件也可以启动相应的应用程序，打开选定的文件。

OpenOffice.org 办公套件的一个重要特点或功能是能够读取并处理 Microsoft Office 格式的文件，把

Microsoft Office 格式的文件转换成 Open Document 格式的文件。例如，如果在 OpenOffice.org 中选择“文件→打开”，然后在对话框中选择 any.doc 文件，将会打开在 Microsoft Office Word 中创建的 any.doc 文档。

除了 Writer、Impress 与 Calc 之外，OpenOffice 还提供 Base 数据库和 Math 公式编辑等应用。Base 是一个类似于 Microsoft Access 的数据库程序，可用于建立关系数据库，或者连接 MySQL 等数据库。一旦创建了数据库表，即可输入、检索或打印数据库文件中的数据。Base 也可与其他 OpenOffice.org 应用交换数据，如利用数据库中的数据生成一个电子表单等。Math 公式编辑应用提供数学公式的编辑与计算功能，既可单独使用，也可以作为一种对象，用于 Writer 与 Calc 等。

限于篇幅，本章不打算详细介绍上述应用程序的具体操作，如想了解 OpenOffice.org 更多信息，可以访问 <http://www.openoffice.org> 等网站。

## 2. 字典

GNOME 采用默认的网络字典（dict.org）提供字典服务。用户可以查询英文单词的释义与用法，GNOME 将会在显示区内显示检索结果。如果单词拼写错误，GNOME 会启动拼写检查程序，提供一个列表供用户选择拼写正确的单词，然后再查询。此外，用户也可以在输出显示区内检索指定的关键字。字典查询的结果能够被复制到 GNOME 桌面的其他应用，也可以保存或打印。注意，GNOME 不使用本地字典，必须连接到 Internet 才能使用。除了默认的网络字典，用户也可以指定其他提供字典服务的网站作为字典服务器，如图 2-17 所示。

## 3. 项目管理

GNOME 还提供一个功能丰富的项目管理工具 Planner，能够帮助用户管理与跟踪项目的任务分配、完成日期以及所用的资源。可以按阶段分配任务，为任务赋予重要性级别，必要时，甚至可以重新分配任务等。用户可以利用可视化的 Gantt 图表显示项目的进度，了解项目的进展情况，采用高亮度方式标注项目阶段需要完成的关键任务等。

注意，Planner 软件包需要单独安装，有关 Planner 项目管理工具的更多信息，可以访问 <http://live.gnome.org/Planner> 网站。

### 2.3.2 附件

GNOME 提供一组标准的附件，以便用户执行基本的处理功能。单击“应用程序→附件”菜单，即可选用各种附件，如“Tracker 搜索工具”、“阿帖便笺”、“磁盘使用分析器”、“管理打印任务”、“计算器”、“密码和加密密钥”、“文本编辑器”、“终端”、“抓图”及“字符映射表”等。

#### 1. 终端

“终端”提供命令行终端界面，以便用户采用命令行的方式，直接访问 Shell 和 Linux 系统。用户可以修改终端窗口的背景图案（包括透明背景），设置文字和背景的颜色，以及设置窗口滚动缓冲区，等等。此外，终端窗口也支持标签操作等。这是许多资深 Linux 用户最喜欢使用的一种用户界面，后续各章介绍的 Ubuntu Linux 系统命令行功能都需要用到此界面，如图 2-18 所示。

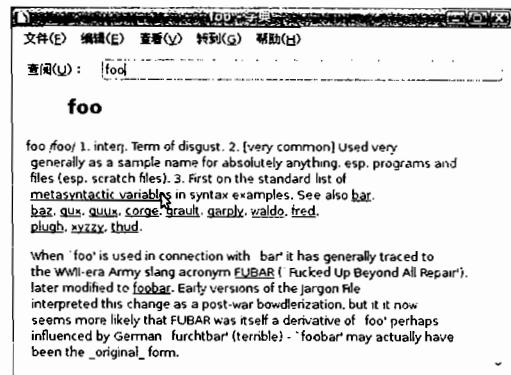


图 2-17 字典



## 2. 文本编辑器

文本编辑器 gedit 相当于 Windows 系统中的记事本功能，可用于创建和编辑简单的文本文件。文本编辑器可以同时打开和处理多个文件，每个文件位于一个单独的窗口。如果需要，可以从一个文件中复制、剪切文本数据，然后粘贴到另外一个文件中。

此外，gedit 文本编辑器还可用作一个软件开发工具。当通过“查看→突出显示模式”菜单选择不同的语言时，如 C/C++、Java、Shell、Perl、XML 或 HTML 等，gedit 文本编辑器会根据不同的语言，以不同的颜色或高亮度方式显示其中的关键字和标识符等，如图 2-19 所示。



图 2-18 终端窗口

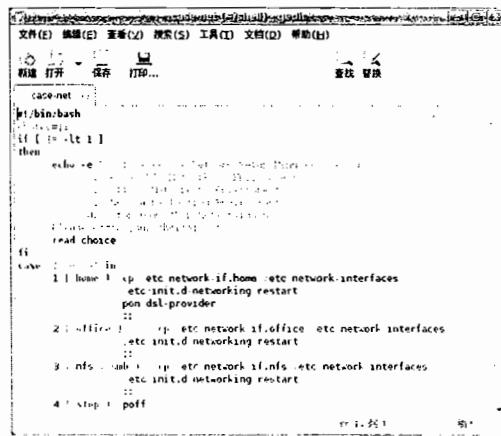


图 2-19 gedit 文本编辑器

### 3. 磁盘使用分析器

“磁盘使用分析器”是一个图形界面的磁盘维护工具，用于分析磁盘的使用情况。“磁盘使用分析器”能够扫描整个文件系统、用户主目录或指定的任何目录分支等，最后以分类统计的报表形式以及形象化的磁盘空间占用形式，给出磁盘使用情况的分析结果，如图 2-20 所示。

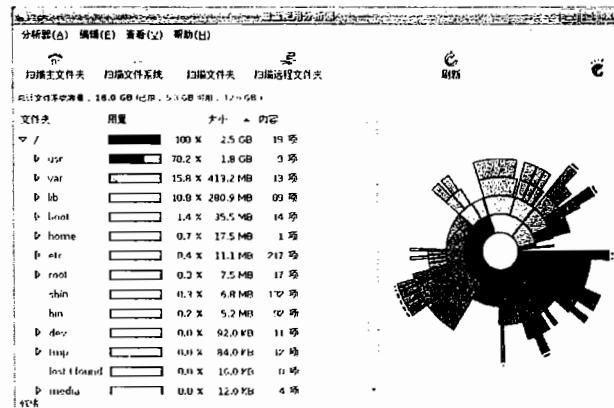


图 2-20 磁盘使用分析器

#### 4. 抓图

“抓图”用于捕捉和保存屏幕快照，如抓取整个桌面、当前窗口等。此外还可以设定抓图的延时。

迟时间，这对于抓取屏幕快照是非常有用的。本书采用的部分屏幕截图也是利用“抓图”功能制作的，如图 2-21 所示。

### 5. 计算器

GNOME 提供一个全功能的计算器，可以执行各种数学运算。默认的计算器是一个基本算术运算计算器，能够进行加、减、乘、除等数字运算。利用计算器上的“查看”菜单，可以选用 3 种其他形式的计算器：“高级”计算器提供更丰富的计算功能，如平方根、平方、分数、绝对值以及倒数运算；“金融”计算器提供金融运算，其中包括利率、现值、支付以及贬值等运算功能；“科学”计算器具有“高级”计算器的全部功能，还能够计算三角函数、阶乘、乘方、对数及逻辑运算等，能够采用计算公式自定义函数，也可以采用不同的数字表示方法（如二进制、八进制、十六进制以及十进制等），如图 2-22 所示。

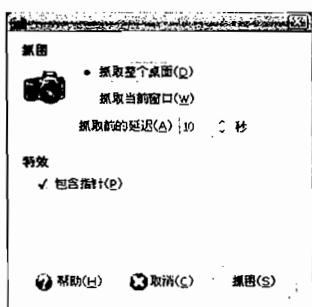


图 2-21 抓图



图 2-22 “科学”计算器

### 2.3.3 互联网

“互联网”菜单主要提供网络浏览器、电子邮件、即时消息、网上聊天以及 IP 电话与视频会议等功能。

#### 1. Firefox 浏览器

Firefox 是 GNOME 桌面环境提供的默认 Web 浏览器，它具有丰富且安全的浏览器功能。安装 Ubuntu Linux 系统之后，只要正确地设置并启动了网络，即可使用 Firefox 访问 Internet，浏览网页，复制网页内容，保存与打印网页，下载文件，等等，如图 2-23 所示。如果需要，也可以在 Firefox 中选择“编辑→首选项”菜单定制 Firefox，如设置主页等。

Firefox 是一种快速的小型浏览器，其插件功能

使 Firefox 能够根据需要随时扩展，以支持新的功能特性（详见 <http://www.mozilla.org/support/firefox>）。

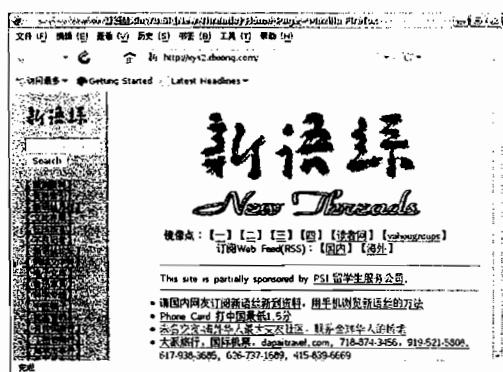


图 2-23 Firefox 网络浏览器



## 2. 电子邮件

GNOME 利用 Evolution 提供电子邮件功能。Evolution 是一种具有电子邮件、日历、计划和信息组织功能的通信工具软件，它使用户之间能够相互传递信息。在使用电子邮件之前，需要配置 Evolution 环境，指定邮件服务器、用户账号及邮件客户端类型等。有关 Evolution 的更多信息，包括配置和使用说明文档，请参见 <http://www.gnome.org/projects/evolution>。

## 3. IP 电话与视频会议

Ekiga IP 电话与视频会议是一种快捷、经济的通信交流方式，能够取代面对面的传统会议形式，前提是每个参与方都要联网，并配备视频与音频设备，如 PC 摄像头和话筒等。

Ekiga 能够利用 Internet 建立双方或多方视频会议。在一定的时间内，参与方均应连接到会议服务器上，由其中的一方作为会议主持。视频会议除了提供语音服务之外，还提供视频服务，如视图、报表显示、视频播放等，使与会用户如同参加面对面的会议。与会者讲话时可以让摄像头对准自己，以便其他人能够看到讲话者。

## 4. 即时消息

即时消息（Instant Messaging, IM）是目前常用的一种重要通信方式。利用短消息与同事或朋友联系，有时比电子邮件或电话更方便快捷。

Pidgin 是一个多用途的聊天工具，是一个模块化的即时通信客户程序，具有丰富的功能，支持 AIM、MSN、Yahoo、QQ、ICQ、IRC 和 Google Talk 等通信协议。

Pidgin 本身提供设置即时通信所需的配置界面，在该界面中能够设置网络、好友名单以及即时通信协议，实现注册选择、插件选择以及日志查询等。

如果想要使用 Pidgin 即时通信程序，首先需要选择“应用程序→互联网→Pidgin 互联网通信程序”菜单，设置即时通信程序的运行环境。事实上，第一次启动即时通信程序时将会直接进入“账户”配置对话框，单击“添加”按钮即可进入“添加账户”对话框，如图 2-24 所示。

在图 2-23 所示的对话框中，可首先选择采用哪一种即时通信协议。单击“协议”字段的下拉菜单，根据采用的即时通信方式，选择 MSN、QQ 或 IRC 等。然后输入与选定通信方式相对应的用户名、密码及本地别名等信息，视情况可勾选“记住密码”复选框，如图 2-25 所示。

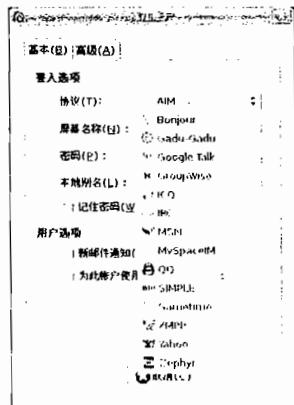


图 2-24 选择即时通信协议

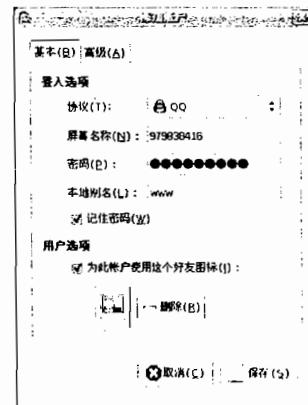


图 2-25 输入用户名和密码

最后，单击“保存”按钮即可增加一个用户，完成即时通信的初始设置。一旦增加了新账号，

“账户”窗口将会出现新加的用户账号，如图 2-26 所示。

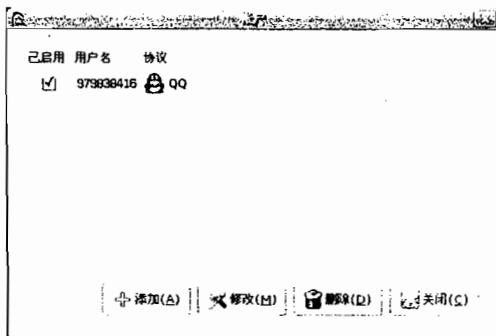


图 2-26 用户账号界面

此后，每当再次启动 Pidgin 即时通信程序时，桌面信息公告区将会出现一个信件加绿色球状的图标，单击该图标将会弹出一个“好友列表”窗口。从中选择在线交谈的对象，即可进入即时通信对话框，开始直接对话，如图 2-27 所示。

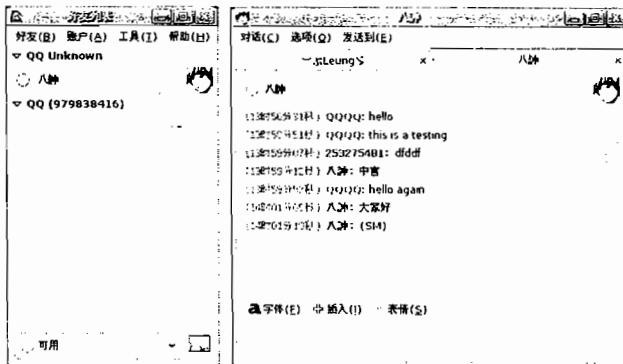


图 2-27 即时通信界面

注意，在设置和使用 Pidgin 即时通信功能之前，用户应当已经事先采用常规的方法与要求申请即时通信网络的账号。

### 2.3.4 图形

GNOME 桌面提供若干图像应用程序，每一种图像应用程序都有其特定的用途，如 F-Spot 照片管理器、GIMP 图片编辑器、OpenOffice.org 图画及 XSane 图像扫描器等。

#### 1. GIMP 图片编辑器

GIMP (GNU Image Manipulation Program) 是一个功能强大的图像处理器，提供一整套的画笔、铅笔和喷枪等编辑工具，可用于处理数字图像和照片，如创建、缩放、剪辑、改变颜色、分层组合图像、删除部分图像特征以及在不同的图像格式之间转换等。GIMP 还可用于生成简单的动画图像，也可用于捕捉、制作和保存屏幕快照。实际上，本书采用的部分屏幕截图就是利用 GIMP 生成的。要想了解怎样使用 GIMP，可以访问 <http://wiki.ubuntu.org.cn> 网站，其“站点导航”的“影



音图像”中提供了一个完整的用户手册。图 2-28 所示为 GIMP 的主要界面。

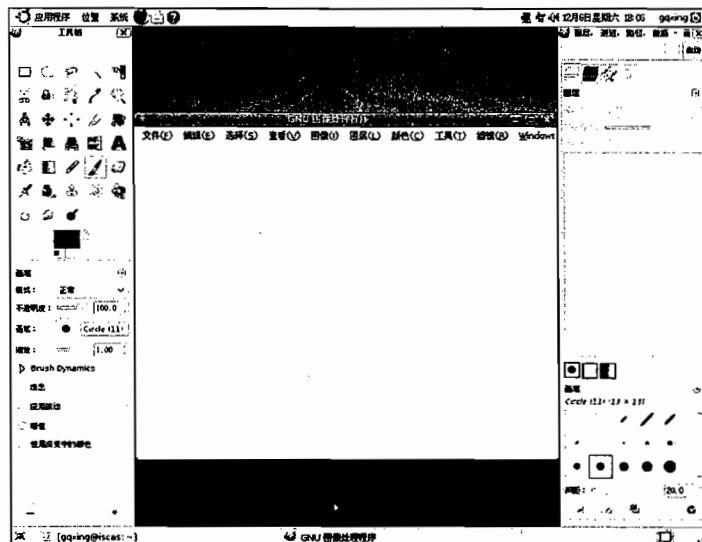


图 2-28 GIMP 图像工具

## 2. gThumb 图像浏览器

gThumb 是一种开源的图像浏览器，能够复制、移动、删除、打印、缩放图像以及转换图像文件格式等，能够编辑图像，如剪接、调色、旋转、增加注释以及增强图像效果等，能够浏览存储介质中的图像文件、分类组织、检索与浏览，也能够以幻灯片的形式播放图像，如图 2-29 所示。注意，在 Ubuntu 8.10 Linux 系统中，gThumb 图像浏览器需要单独安装。

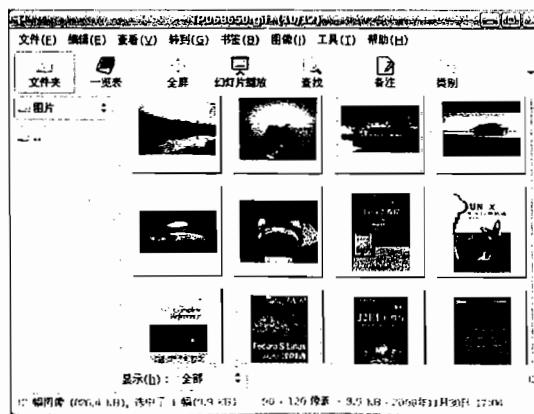


图 2-29 gThumb 图像浏览器

## 2.3.5 影音

Ubuntu GNOME 提供部分音频和视频播放工具，用于欣赏音乐和播放影视，其中包括光盘刻录器、Rhythmbox 音乐播放器、电影播放机及录音机等。为了访问这些多媒体工具，可以选择“应用程序→影音”菜单。通常，Ubuntu GNOME 桌面环境提供下列 4 个影音工具软件：

- Brasero 光盘刻录器;
- Rhythmbox 音乐播放器;
- 电影播放机;
- 录音机。

### 1. Brasero 光盘刻录器

Brasero 是一个 CD/DVD 刻录工具，支持音频 CD、视频 DVD/VCD 和数据 CD/DVD 等刻录模式，可用于录制 CD/DVD，利用 ISO 镜像刻录 CD/DVD，制作光盘数据文件备份，检测光盘文件的完整性，擦除可重写的光盘，支持光盘的封面设计，等等。Brasero 光盘刻录器的用户界面直观简洁，操作简便，如图 2-30 所示。

### 2. Rhythmbox 音乐播放器

Rhythmbox 是一个音乐管理与播放软件，配有在线商店 Jamendo 和 Magnatune，支持一系列网络电台，能够组织多媒体文件如音乐、CD 和网络电台等，构建自己的多媒体库，播放音乐文件，从音频 CD 上导入音乐文件，聆听 Internet 网络电台，等等。Rhythmbox 可以作为首选的 MP3 音乐播放器，其界面如图 2-31 所示。

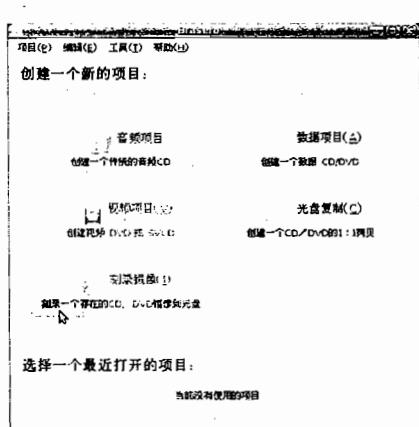


图 2-30 Brasero 光盘刻录器

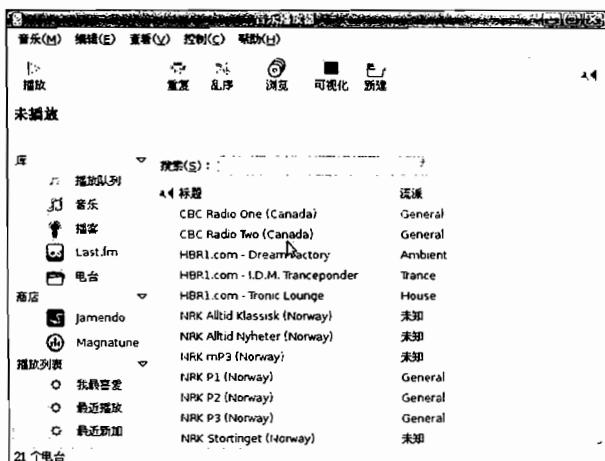


图 2-31 Rhythmbox 音乐播放器

### 3. 电影播放机

Totem 电影播放机 (Totem Movie Player) 是一个全功能的电影播放器，它支持视频和音频文件，能够播放 DVD/VCD 影视，播放音频 CD，具有屏幕影像控制及键盘控制功能，能够像 DVD 机一样控制影视的播放（使用前需要补充安装 gstreamer 系列的音频和视频插件等软件包，详见 2.5.2 小节的“硬件测试”），其界面如图 2-32 所示。

此外，还可以选用 MPlayer、SMPlayer 和 RealPlayer 等视频播放器，但需要单独安装，详见 <http://wiki.ubuntu.org.cn>。其中，MPlayer 能够播放 MPEG/VOB、AVI、Ogg/OGM、VIVO、ASF/WMA/WMV、QT/MOV/MP4、RealMedia、Matroska、NUT、NuppelVideo、FLI、YUV4MPEG、FILM、RoQ、PVA 等格式的视频文件，也可用于播放 VideoCD、SVCD、DVD、DivX 3/4/5、WMV 甚至 H.264 等格式的电影。



#### 4. 录音机

Sound Recorder 录音机用于录制.flac、.oga 及.wav 格式的音频文件，能够利用计算机系统配置的扬声器或耳机播放音乐，其界面如图 2-33 所示。



图 2-32 Totem 电影播放机

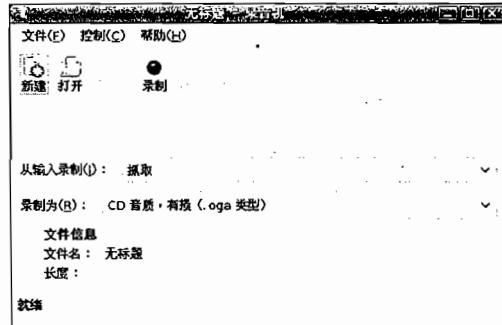


图 2-33 录音机

### 2.3.6 游戏

通常，GNOME 至少提供 17 个游戏，如 Gnome 方块、对对碰、空当接龙、国际象棋、扫雷等。安装之后，还可以利用“应用程序→添加/删除程序”菜单，从 Ubuntu 软件仓库提供的大量游戏软件中选择下载自己喜欢的游戏软件，如连连看等。

### 2.3.7 添加/删除软件

利用 gnome-app-install 软件工具，可以访问安装介质或 Internet 上的软件仓库，下载、安装新的软件包，或者删除已经安装的软件包（详见第 12 章）。

## 2.4 位置菜单

“位置”菜单的主要功能是快速定位系统资源，如快速进入用户主目录等。“位置”菜单提供的大部分功能均采用 nautilus 文件浏览器，供用户访问文件系统，浏览系统配置的各种外部设备，检索可用的网络资源，检索目录与文件，等等，如图 2-34 所示。

Nautilus 是 GNOME 内置的文件管理器，使用户能够浏览文件和文件夹，其中包括：

- 管理文件和文件夹，定制文件夹的显示形式；
- 将数据写入移动存储介质，如 USB 移动盘和 CD/DVD 等。

Nautilus 文件浏览器的功能类似于 Windows 系统中的资源管理器。其中，左边的“位置”窗口中列出了可以浏览的目录、文件系统或存储介质（右击存储介质后面的三角形图标，可以卸载相应的文件系统）。右边的主窗口中列出了目录、文件系统或存储介质中包含的文件或文件夹。利

用文件浏览器，可以浏览、复制、移动、创建、删除以及修改文件与文件夹，可以查阅文件和文件夹的内容，也可以按照指定的模式检索指定的文件与文件夹。

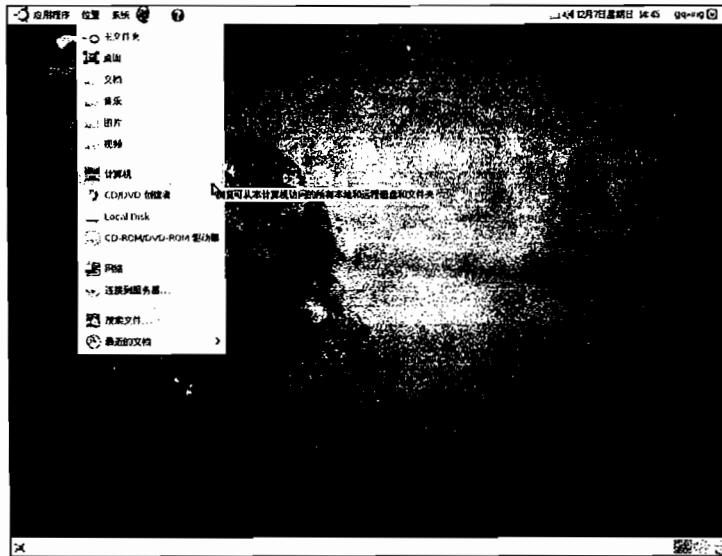


图 2-34 位置菜单

在 Ubuntu 8.10 Linux 系统中，Nautilus 文件浏览器还提供标签支持，可以通过标签浏览不同目录、文件系统或存储介质中的文件或文件夹。

Nautilus 文件浏览器能够以图标形式显示文件和文件夹，能够以详细列表的形式显示文件和文件夹，同时给出文件的大小、创建日期及文件的类型等信息，也能够以紧凑方式显示文件和文件夹。图 2-35 所示就是以紧凑方式显示“/”文件系统的结果。

Nautilus 文件浏览器采用不同的图标形式区别文件和文件夹，区分不同类型的文件。例如，以公文包图标表示文件夹，以计算机芯片图标表示可执行程序文件，以含有计算机芯片的图标形式表示脚本文件，以纸张加文字的图标表示文本文件，以图像图标表示图像文件，以盒状图标表示压缩文件，以文档、电子表和幻灯片等图标表示 OpenOffice.org 文件，以含有数字 0 的图标形式表示系统文件，等等。当以图标形式显示文件夹或文件时，将在图标的下方（或右边）显示文件夹或文件的名字，如图 2-36 所示。

在文件浏览器中，有些文件或文件夹上还带有附加的标记，用于表示文件或文件夹的访问权限，以及符号链接文件。例如，锁状附加标记表示当前的用户只能读但不能写相应的文件或文件夹，叉号（“×”）附加标记表示当前用户根本就不能读写相应的文件或文件夹，含有箭头附加标记的图标形式表示符号链接文件，等等。

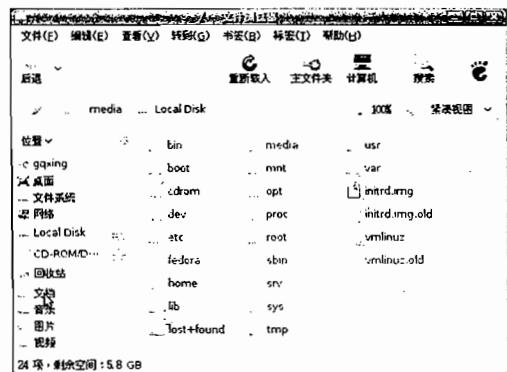


图 2-35 文件浏览器



此外，还可以使用其他附加标记标识文件夹或文件内容的属性，例如，利用感叹号附加标记表示一个重要文件。如果想为文件或文件夹增加附加标记，可以右击文件或文件夹，从弹出的上下文菜单中选择“属性”，然后选择“徽标（Emblems）”标签，从列出的可用徽标中选择期望的附加标记（可以同时选择多个）。一经选定，附加标记就会被自动地加到文件或文件夹，如图 2-37 所示。

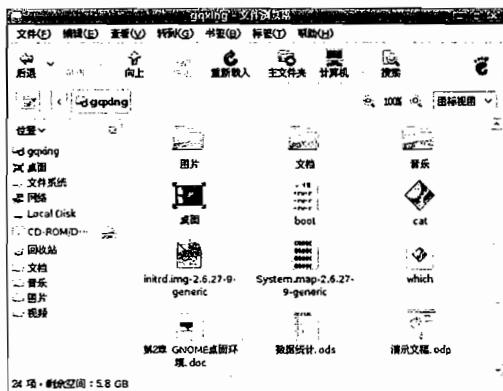


图 2-36 文件浏览器图标显示

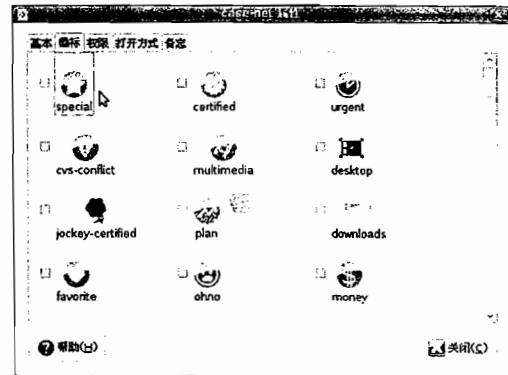


图 2-37 文件浏览器属性界面

在文件浏览器中，利用鼠标单击文件，可以打开或运行选中的文件。与 Windows 系统一样，文件的类型或扩展名可以决定与文件相关联的默认操作。双击一个可执行文件，将会启动相应的程序（假定用户拥有执行文件的访问权限）；双击一个文本文件，系统就会利用 gedit 文本编辑器打开文件（假定用户拥有读文件的访问权限）。用鼠标拖放文件或文件夹时，可以把文件或文件夹从一个文件夹移至另一个文件夹或文件系统，前提同样是需要拥有适当的访问权限。

### 2.4.1 主文件夹

“主文件夹”是当前注册用户的主目录，相当于 Windows 系统的“我的文档”，对于普通用户而言，主文件夹即/home/username 目录，其中的“username”是当前注册的用户名。

单击“主文件夹”菜单即可进入用户的主目录，利用文件浏览器浏览和访问其中的文件与目录，如图 2-38 所示。主文件夹用于存储用户自己的所有文件，不同用户的主文件夹是不同的。

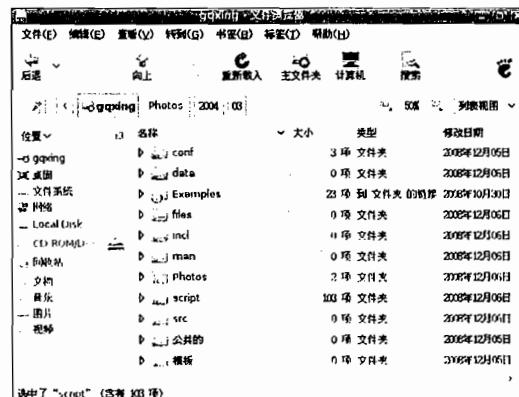


图 2-38 主文件夹

## 2.4.2 桌面、文档等

“桌面”、“文档”、“音乐”、“图片”以及“视频”是用户主目录下的子目录，选择任何一个菜单项，即可调用文件浏览器，快速访问其中的文件。

## 2.4.3 计算机

“计算机”用于访问计算机系统配置的所有硬件存储设备，包括已安装的移动介质，如“软盘驱动器”、“CD-ROM/DVD-ROM 驱动器”、Windows 系统分区“Local Disk”、位于系统磁盘上的“文件系统”和 USB 移动硬盘等，其功能等价于 Microsoft Windows 系统的“我的电脑”，可用于访问其中存储的文件等，其界面如图 2-39 所示。

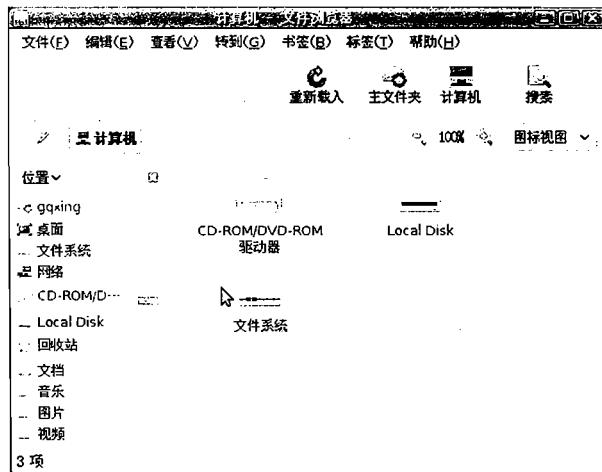


图 2-39 计算机界面

如果插入 CD/DVD 或 USB 移动盘，系统将会自动安装相应的文件系统。否则，可以直接单击相应的图标，安装相应的文件系统（其安装点位于/media 目录下面）。

## 2.4.4 CD/DVD 刻录机

“CD/DVD 创建者”用于刻录 CD/DVD，如图 2-40 所示。如果想把数据文件或文件夹写入 CD/DVD，或者复制 CD/DVD，可以利用 GNOME 桌面提供的“CD/DVD 创建者”功能。把空白 CD/DVD 插入刻录机之后，仿照下列步骤执行。

- (1) 选择“位置→CD/DVD 创建者”菜单。
- (2) 进入“CD/DVD 创建者—文件浏览器”窗口。
- (3) 打开另外一个文件浏览器，找出并选择准备复制的源文件，把文件拖放至“CD/DVD 创建者文件夹”中的空白处。
- (4) 单击右上角的“写入光盘”按钮。
- (5) 必要时，可以在弹出的“写入光盘”对话框中修改 CD/DVD 的卷标名称等。

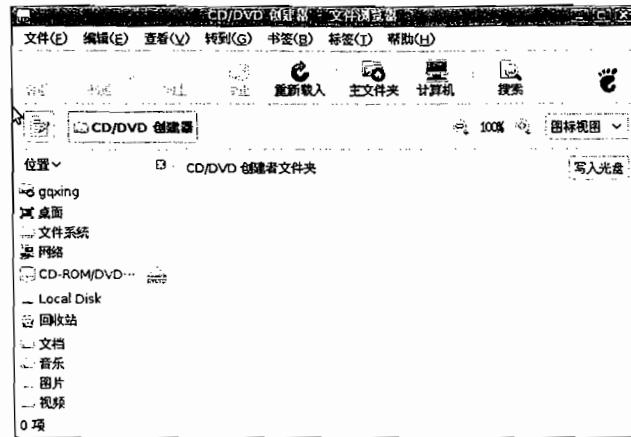


图 2-40 CD/DVD 创建者界面

(6) 单击“写入光盘”对话框右下角的“写入”按钮。

## 2.4.5 磁盘分区

如果系统中存在其他磁盘分区，如 Windows 系统分区或其他 Linux 分区，也可以通过位置菜单访问相应的文件系统。例如，在一个同时安装了 Microsoft Windows 与 Ubuntu Linux 双系统的计算机中，位置菜单中将会存在表示 Windows 系统磁盘分区的菜单项，其名字为磁盘分区的卷标，如“Local Disk”，如图 2-41 所示。单击磁盘分区菜单项即可安装、访问相应的文件系统。

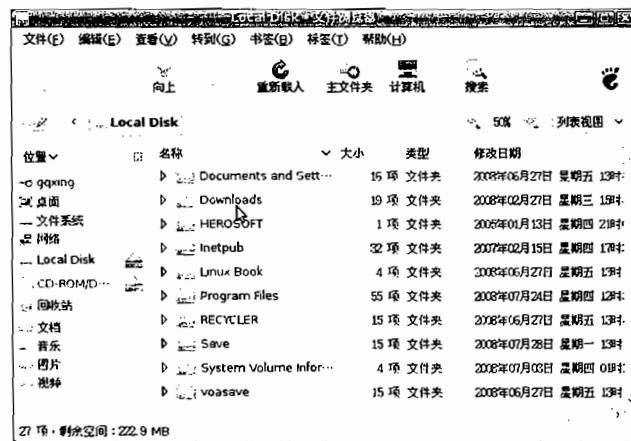


图 2-41 磁盘分区界面

## 2.4.6 网络

“网络”类似于 Microsoft Windows 系统中的“网络邻居”，用于检索可用的网络服务器，其界面如图 2-42 所示。

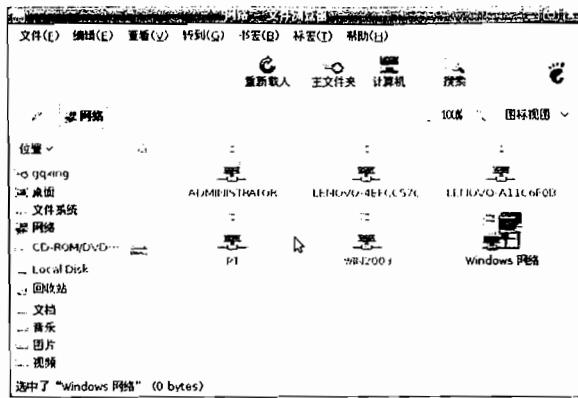


图 2-42 网络界面

### 2.4.7 连接到服务器

“连接到服务器”菜单用于连接 SSH、FTP、Samba 等服务器，以及其他共享网络资源等，其界面如图 2-43 所示。

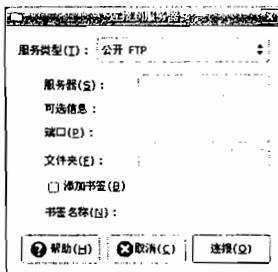


图 2-43 连接服务器界面

### 2.4.8 搜索文件

“搜索文件”用于检索文件系统或存储介质中的文件，其效果等同于在 Microsoft Windows 系统中运行 find 命令。例如，为了查询系统中存在哪些配置文件，可以使用“\*.conf”作为检索模式（大部分配置文件均以“.conf”作为文件扩展名），其结果如图 2-44 所示。

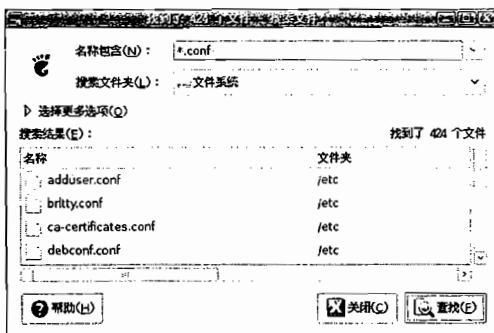


图 2-44 搜索文件界面



## 2.4.9 最近的文档

“最近的文档”等价于 Microsoft Windows 系统“开始”菜单中的“文档”，用于快速定位并打开最近访问过的文件。

# 2.5 系统菜单

系统菜单主要提供系统规律与维护方面的功能，如网络管理、用户与用户组管理、打印管理、系统服务管理、系统日期与时间设置、软件源维护以及更新软件包等，如图 2-45 所示。此外，还可以通过系统菜单访问 Ubuntu 帮助中心，获取联机帮助与支持信息，了解 Ubuntu 的特点及由来。

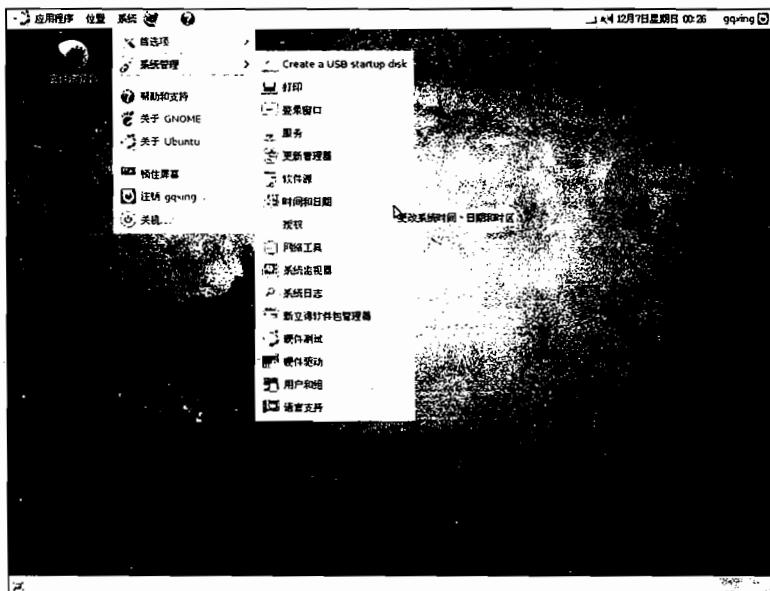


图 2-45 系统菜单

在“系统”菜单（或其他 GNOME 图形界面的程序）中，当选择的菜单项或执行的功能只有超级用户才能执行时，窗口将会呈现暗灰色。此时单击“解锁”按钮，在新弹出的“认证”窗口中输入 sudo 密码，然后才能继续。

## 2.5.1 首选项

作为系统管理工具，“首选项”主要用于设定各种 GNOME 桌面工具的默认参数和系统配置数据，如默认打印机设置、菜单的组织与布局、菜单项的选择、菜单和工具栏设置、是否在菜单中显示图标、快捷键的默认设置、显卡的分辨率与刷新率设置、当系统空闲时是否启动屏幕保护程序、屏幕保护程序开始运行时是否锁住屏幕及桌面背景等。本小节仅以 4 个例子予以说明。

## 1. 远程桌面

Ubuntu 提供一种非常有用的机制，能够在网络上共享远程 GNOME 桌面，使得系统能够接收远程技术支持，或向他人演示 GNOME 桌面的功能特性。用户也可以从其他系统中远程访问自己的桌面环境（注意，在使用之前，首先需要安装 xtightvncviewer 或类似的软件包）。

为了启用 GNOME 桌面共享功能，可以选择“系统→首选项→远程桌面”菜单，打开“远程桌面首选项”窗口，如图 2-46 所示。

然后采用下列最安全的步骤，设置远程 GNOME 桌面共享功能。

- (1) 在“共享”部分勾选“允许其他人查看您的桌面”。
- (2) 然后勾选“允许其他用户控制您的桌面”。
- (3) 在“安全”部分勾选“请您确认”。
- (4) 然后勾选“要求用户输入此密码”，接着输入自己选定的密码。
- (5) 单击“关闭”按钮结束设置。
- (6) 使用“vncviewer iscas:0”命令，检验是否能够连接自己的 GNOME 桌面环境（其中的 iscas 是作者所在系统的名字）。

设置完成之后，可以按照步骤(6)予以验证，然后即可告知远程技术支持人员，包括设定的密码。远程人员可以使用与下列例子类似的 vncviewer 命令连接指定的系统。

```
$ vncviewer iscas:0
VNC server supports protocol version 3.7 (viewer 3.3)
Password:
```

当远程系统连接自己的 GNOME 桌面要求确认时，可以单击“允许”按钮予以确认，如图 2-47 所示。注意，远程桌面共享功能存在严重的安全风险，仅当真正需要时才应启用这一功能，一旦用完后应立即禁用。

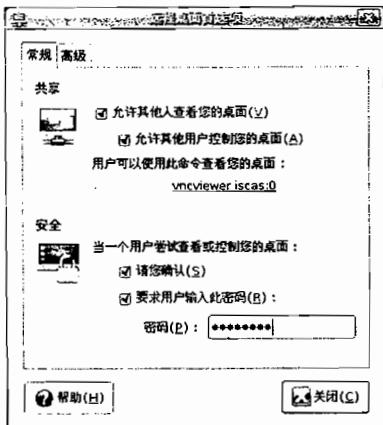


图 2-46 远程桌面设置

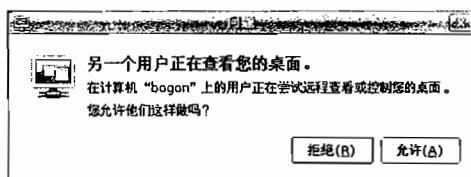


图 2-47 确认远程 GNOME 桌面的连接请求

## 2. 菜单布局

利用菜单布局功能，可以查询 GNOME 环境的菜单布局与配置信息，也可以根据自己的需要，定制菜单布局。例如，增加新的菜单，在现有的菜单中增加新的菜单项，隐藏甚至删除菜单项，上移或下移菜单项，调整菜单项的顺序，以及修改菜单项的属性，等等。



如果想要定制菜单布局，可以选择“系统→首选项→主菜单”，打开“主菜单”窗口，如图 2-48 所示。

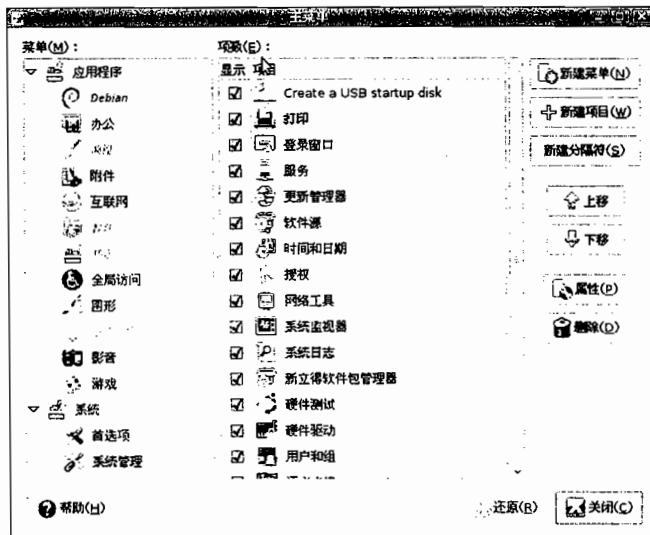


图 2-48 菜单布局

例如，在“系统”菜单中，如果想把“打印”菜单项改为“打印管理”，可在“主菜单”界面左边的窗口中单击“系统管理”，从右边的窗口中选择“打印”，然后单击右键，在弹出的“启动器属性”窗口中，把“名称”字段中的“打印”改为“打印管理”，如图 2-49 所示。

### 3. 快捷键

利用“系统→首选项→键盘快捷键”菜单，可以设置快捷键。实际上，GNOME 已经提供部分常用的快捷键，如图 2-50 所示。例如，使用“Print”键可以直接抓取整个屏幕的截图，使用“Alt+Print”键可以直接获取活动窗口的屏幕截图，而无需调用 GMIP 等截图工具。



图 2-49 修改菜单项属性

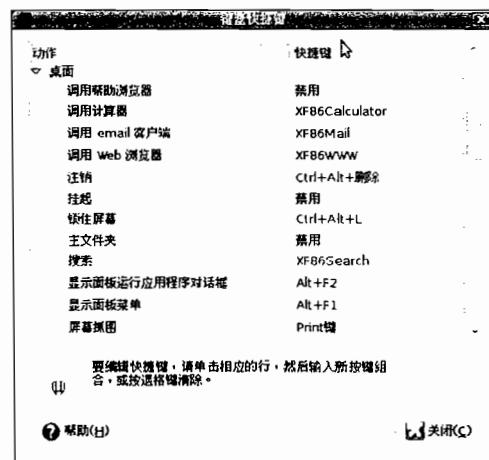


图 2-50 GNOME 快捷键

#### 4. 屏幕保护程序

“屏幕保护程序”用于设置、激活屏幕保护程序，设置在激活屏幕保护程序时是否锁定屏幕等，其界面如图 2-51 所示。

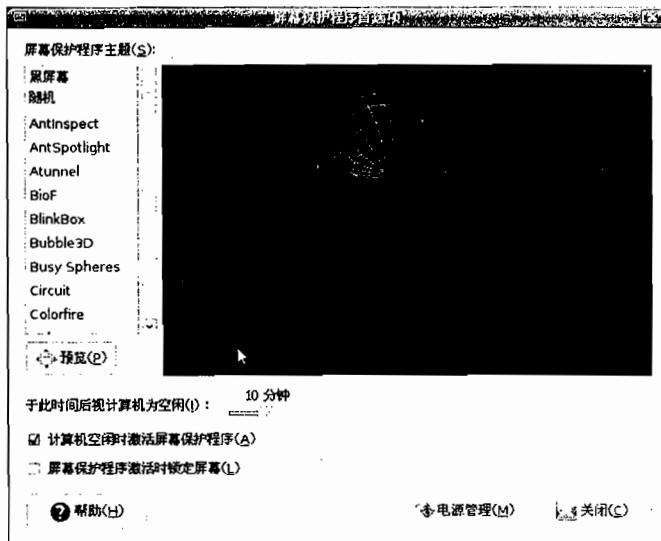


图 2-51 “屏幕保护程序”界面

### 2.5.2 系统管理

“系统管理”主要用于管理、配置以及维护系统与网络，如设置系统服务、TCP/IP 网络、系统日期与时间、用户与用户组、打印机及显示器等。

#### 1. 打印管理

在系统的会话过程中，用户可能需要打印文件。GNOME 桌面环境利用通用 UNIX 打印系统（Common UNIX Printing System, CUPS）管理和维护可用的打印机，包括本地和网络打印机。

在完成打印机的硬件连接之后，需要配置打印机时，可以选择“系统→系统管理→打印”菜单，启动图 2-52 所示打印机配置窗口，查询、增加和配置打印机。

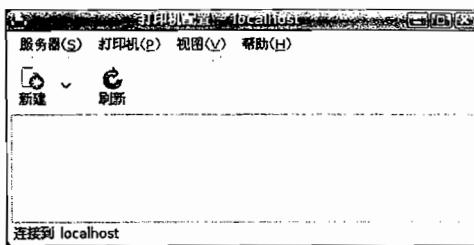


图 2-52 打印机配置

设置新的打印机时，可单击左上角的“新建”按钮，然后在弹出的“新打印机”窗口中选择连接方式，如并行口或 IPP 等，如图 2-53 所示。

单击“前进”按钮，在弹出的“新打印机”窗口中选择打印机的品牌，如 HP，如图 2-54 所示。

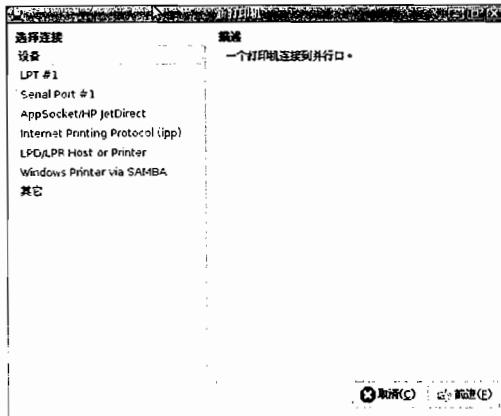


图 2-53 选择打印机连接方式

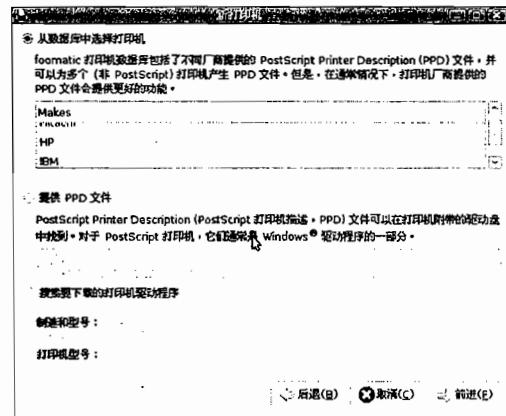


图 2-54 选择打印机品牌

单击“前进”按钮，在弹出的“新打印机”窗口中选择打印机型号，如 LaserJet 2300，然后选择驱动程序，如图 2-55 所示。

单击“前进”按钮，在弹出的“新打印机”窗口中设置打印机的名字，输入打印机的简单描述等，如图 2-56 所示。

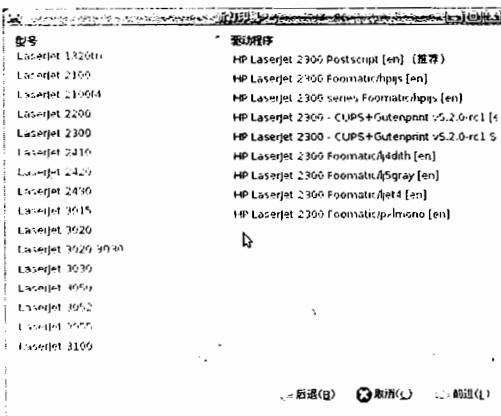


图 2-55 选择打印机型号

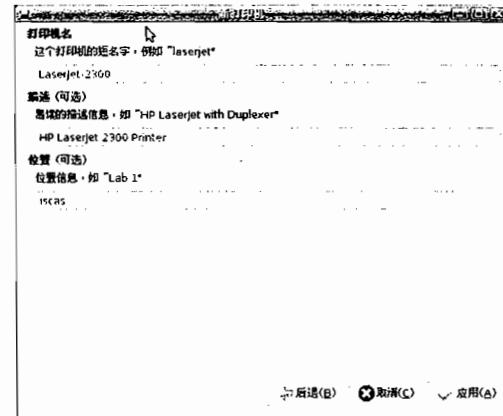


图 2-56 命名打印机

单击“应用”按钮之后，即可在弹出的“打印机配置”界面中见到新增加的打印机，如图 2-57 所示。

最后，可以单击“打印机”菜单，确认是否需要启用此打印机，以及是否需要从现在开始接收打印任务等，如图 2-58 所示。

也可以右击打印机图标，从弹出的上下文菜单中选择“属性”菜单，在弹出的“打印机属性”窗口中设置或修改打印机的属性，设置打印机的策略，如是否启用打印机以及是否可以接收打印任务等，设置打印机的访问控制，以及设置其他打印参数，如图 2-59 所示。

一旦配置好打印机，即可在应用程序中选择“文件→打印”菜单，在默认或选定的打印机上打印文件。

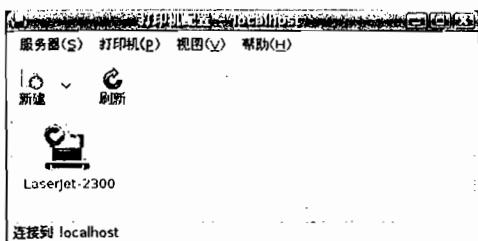


图 2-57 打印机配置界面

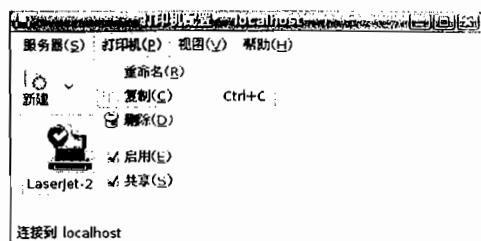


图 2-58 启用打印机

## 2. 注册界面

“注册界面”用于设置系统注册方面的配置参数与安全控制，选择不同风格的注册界面，设定是否启用自动注册功能，是否允许本地系统管理人员在控制台注册界面中使用 root 注册，等等。要设置注册界面，可以选择“系统→系统管理→登录窗口”菜单，进入“登录窗口首选项”界面，如图 2-60 所示。

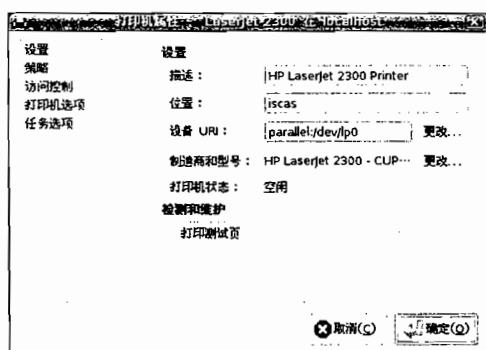


图 2-59 打印机属性界面



图 2-60 注册界面

## 3. 服务

“服务”菜单用于管理系统服务或守护进程。在系统的运行过程中，如果需要，系统管理人员无论何时都可以查询、改变及控制任何守护进程的运行状态，监控、管理与维护所有的系统服务，如启动、停止或重新启动选定的系统服务等。在 GNOME 桌面环境中，为了利用图形界面管理守护进程，可以选择“系统→系统管理→服务”菜单，进入“服务设置”界面，单击服务左边的复选框启用或关闭相应的系统服务，从而运行或终止相应的守护进程，如图 2-61 所示。

## 4. 更新管理器与新立得软件包管理器

通过“更新管理器”，可以利用 update-notifier 和 update-manager 软件工具，访问 Internet 上的软件仓库，确定现有系统中的哪些软件包需要更新。如果软件仓库中存在更新版本的软件包，则提醒用户下载、更新系统中的相关软件包（详见第 12 章）。

“新立得软件包管理器”用于安装与更新软件包，详见第 12 章。

## 5. 软件源

利用“软件源”，可以查询、设置以及维护 Ubuntu Linux 系统的软件源，如图 2-62（有关软件源的说明及其使用，详见第 12 章）所示。

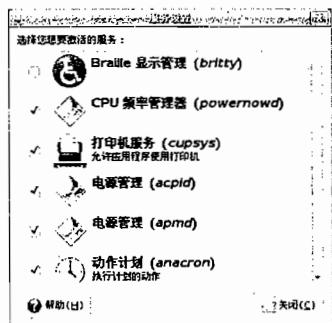


图 2-61 “服务设置”界面

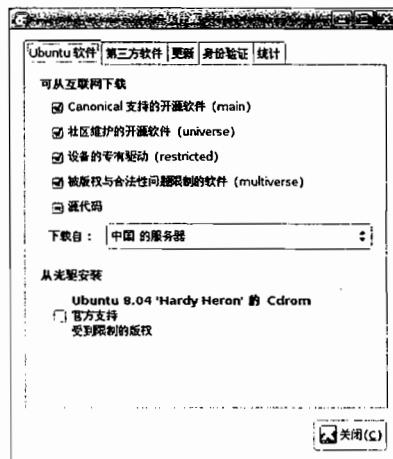


图 2-62 软件源界面

## 6. 时间和日期

“时间和日期”用于显示或设置系统时间，设置或启用网络时间协议 NTP，连接 NTP 服务器，实现时间同步，以及显示或设置时区等，其界面如图 2-63 所示。

## 7. 授权

利用“授权”功能，系统管理人员可以把不同的访问权限赋予不同的用户。例如，要赋予某个用户安装移动介质文件系统的访问权限，可以选择“系统→系统管理→授权”菜单，从“授权”界面左边的窗口中选择“mount file systems from removable drives”，在“授权”界面右边选择“显式授权”（如图 2-64 所示），然后单击“授予”按钮，从弹出的对话框中选择授权的用户。



图 2-63 “时间和日期设置”界面

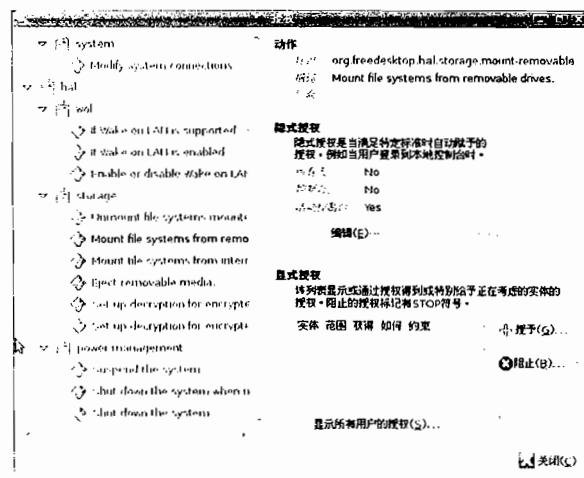


图 2-64 “授权”界面

## 8. 网络

“网络”是一个图形界面的网络管理工具，供系统管理人员查询现有的网络设备及其工作状态，配置网络接口，设置 IP 地址、子网掩码、默认的网关、DNS 和静态路由，激活和关闭系统中的网络设备，其界面如图 2-65 所示。普通用户只能查询网络的配置信息，但超级用户可以执行网络配置等操作（至于如何配置网络，详见第 17 章）。

注意，在 Ubuntu 8.10 Linux 系统，“网络”菜单项需要单独安装。在安装“网络”软件包时，最好选择“应用程序→添加/删除”菜单。

## 9. 网络工具

“网络工具”用于执行常规的网络操作，如查询网络信息、测试网络连接状态、跟踪网络路由、扫描网络端口等，其界面如图 2-66 所示。



图 2-65 “网络设置”界面

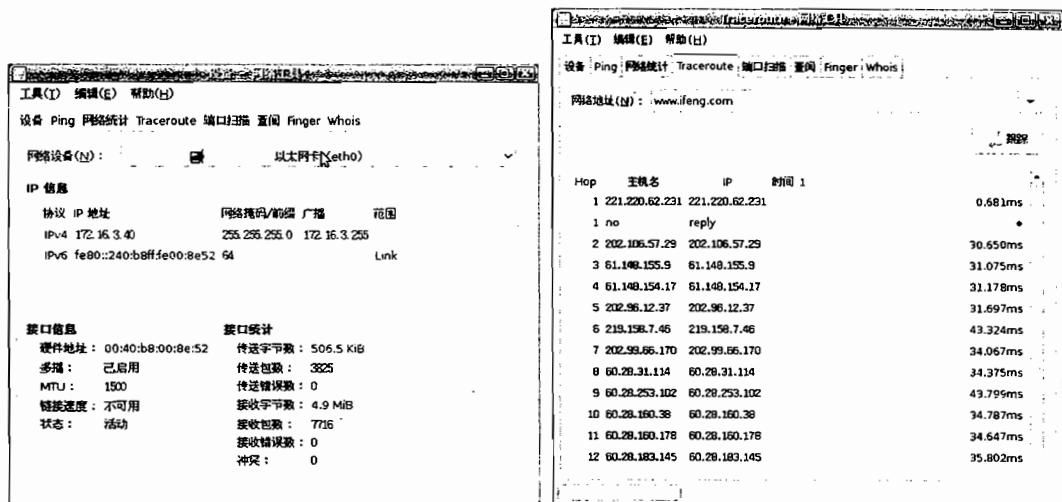


图 2-66 网络工具界面

## 10. 系统监视器

“系统监视器（System Monitor）”类似于 Microsoft Windows 系统中的任务管理器，可用于查询系统的硬件配置，监控、改变进程的运行状态（如暂停、恢复运行或终止进程等），修改进程的优先级，查询进程的内存映像及其打开的文件等，监控系统资源的使用情况（包括 CPU、内存与交换区的利用率以及网络的流量等）、已安装文件系统的使用情况（包括已经占用及空闲的存储空间等），其界面如图 2-67 所示。

## 11. 系统日志

“系统日志”用于查询各种系统日志，包括位于 /var/log 目录中的主要日志文件 messages 中的信息，以及 cron 与 secure 等日志文件中的信息，其界面如图 2-68 所示。

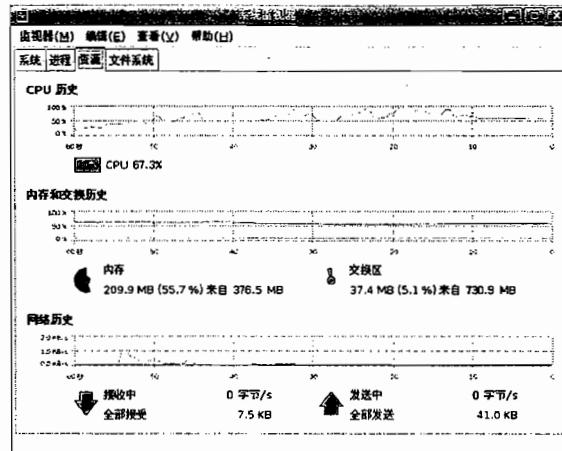


图 2-67 “系统监视器”界面

## 12. 硬件测试

“硬件测试”用于测试系统配置的各种硬件设备，如声卡、显示器的分辨率、色彩与图像、鼠标、键盘、网卡和网络连接等，其界面如图 2-69 所示。

在 Ubuntu Linux 系统中，通常需要使用 apt-get、aptitude 或 synaptic 等软件工具，补充安装 gstreamer 系列软件包（当前的版本为 0.10），确保计算机的音响设备能够发声。在上述硬件测试过程中，如果声卡测试失败，首先应利用菜单面板信息公告区中的“音量控制”图标，确保主声道和扬声器等设备均处于非静音状态；然后使用“apt-get install gstreamer0.10\*”命令，安装 gstreamer 及其相关的底层支撑软件包（参见第 12 章）。

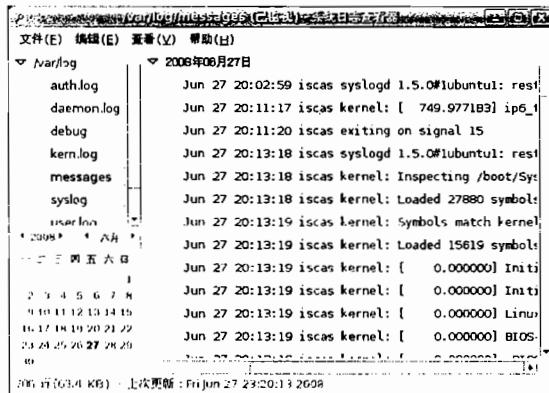


图 2-68 系统日志界面

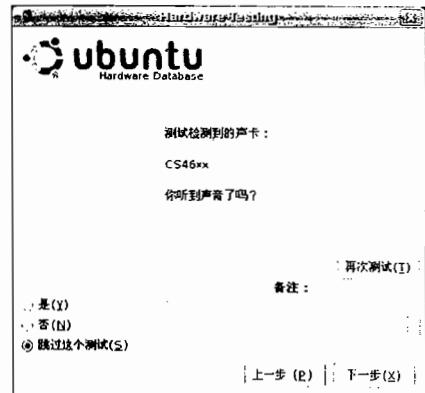


图 2-69 硬件测试界面

## 13. 用户和组

“用户和组”是一个用户管理工具，系统管理人员可以利用这个图形界面增加、删除或修改用户与用户组的注册信息。普通用户可以查询，但只有超级用户才能执行用户的增删改等操作。有关用户或用户组的管理，详见第 13 章。

## 14. 语言支持

“语言支持”用于查询系统当前支持的语言，设置新建用户或注册界面中使用的默认语言环境。

安装附加的语言支持软件包，等等，其界面如图 2-70 所示。

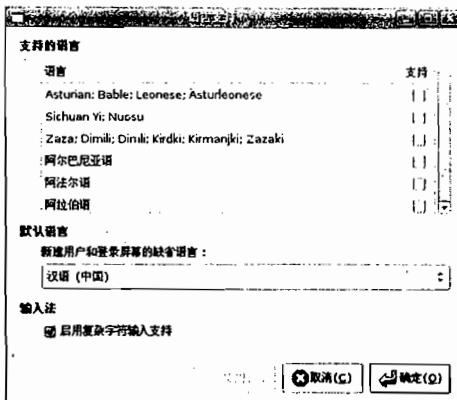


图 2-70 语言支持界面

### 2.5.3 锁住屏幕

顾名思义，“锁住屏幕”用于锁住控制台桌面，启动屏幕保护程序，防止其他人访问 Ubuntu Linux 系统。当需要继续工作时，可按任意键，但只有正确地输入自己的密码才能恢复桌面会话。

### 2.5.4 注销

“注销 xxx”功能用于注销当前的用户，结束系统会话，退出 Ubuntu Linux 系统。如果计算机是共享的，注销之后其他用户可以继续注册与访问 Linux 系统。选择“系统→注销 xxx”菜单之后，系统将会弹出一个窗口，此时可以从两个选项中做出选择：“注销”（默认选择）与“切换用户”。选择“注销”或者等待时间超过 60s，即可退出 GNOME 桌面环境。如果不注销，可以单击“取消”按钮。选择“注销”或“切换用户”，系统将会返回注册界面，此时可用其他用户名再次注册到系统。如果希望关闭系统，可在注册界面下方的菜单区域中选择“关机”，确认后系统将会自动关闭。

### 2.5.5 关机

“关机”菜单用于关机、重启、休眠或挂起 Linux 系统。当选择“系统→关机”菜单时，系统将会弹出一个窗口，用户可从 4 个选项中做出选择：“关机”、“重启”、“挂起”及“休眠”。如果选择“关机”或者等待时间超过 60s，系统即开始自动关机，一旦关闭系统，即可切断计算机的电源。如果希望重新启动系统，可以选择“重启”。如果选择“挂起”或“休眠”，系统将会进入临时关机状态，一旦需要再次访问系统时，按下电源开关，即可进入先前的环境继续工作。如果不关机或重启系统，可以单击“取消”按钮。

## 2.6 使用移动存储设备

GNOME 桌面环境支持各种常用的移动存储设备，如软盘、CD/DVD 及 USB 移动盘等。当插



入 CD/DVD、USB 移动盘或 U 盘等移动存储介质时，GNOME 桌面中会自动增加一个图标，同时弹出一个文件浏览器窗口，表示系统已经自动安装了相应的文件系统。单击 GNOME 桌面中的图标，即可浏览和访问移动存储介质中的文件。

在 Ubuntu Linux 系统中，如果插入的是软盘，系统不会自动安装其中的文件系统，因而也不会增加软盘图标。此时可以选择“位置→计算机”菜单，在弹出的“计算机—文件浏览器”窗口中单击“软盘驱动器”图标，即可安装相应的文件系统。

在用完移动存储介质之后，可以右击相应设备的图标，然后从上下文菜单中选择“卸载文件卷”或“弹出”（取决于所用存储介质的类型），如图 2-71 所示。



图 2-71 卸载移动介质

## 2.6.1 浏览移动存储介质

为了浏览移动存储介质中的文件，只需插入移动介质，从自动弹出的文件浏览器窗口中访问移动存储设备。如果 CD/DVD、USB 移动盘或 U 盘等因故没有自动安装，可在 GNOME 桌面中选择“位置→计算机”菜单，从弹出的“计算机—文件浏览器”窗口中单击相应的图标，即可安装、浏览和访问相应的文件系统，进入移动存储介质的根目录。

例如，在插入一个 2GB 的 U 盘之后，系统将会自动识别新加的存储介质，同时把相应的文件系统安装到/media/disk 目录，在 GNOME 桌面中增加一个表示 U 盘的“2.0GB 介质”图标（没有设置卷标），打开一个“disk—文件浏览器”窗口，如图 2-72 所示。然后，可以像使用 Windows 系统中的资源浏览器一样浏览、访问 U 盘中的文件。

单击“Ubuntu Save”文件夹，即可浏览其中的文件，如图 2-73 所示。

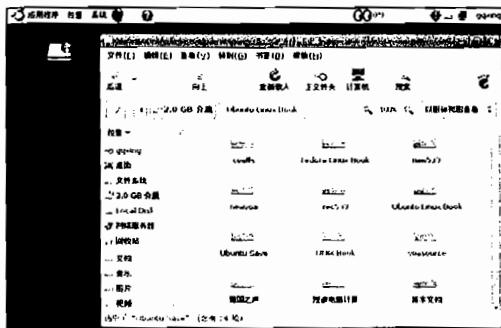


图 2-72 增加 U 盘

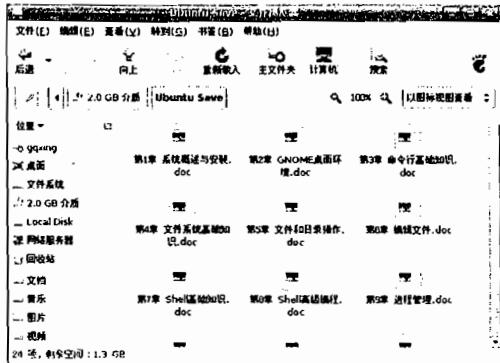


图 2-73 浏览文件

## 2.6.2 写入移动存储介质

在文件浏览器中，可以容易地把文件或文件夹从一个文件夹移至另一个文件夹。由于系统支持软盘、USB 移动盘或 CD/DVD 等类型的文件系统，故也能把文件或文件夹复制或移至不同的存储介质，从而实现数据的备份与交换。

在 GNOME 桌面环境中，为了复制数据，只需插入移动存储介质，系统就会以读写方式，把相应的文件系统自动安装到/media 目录下，然后即可复制数据。

例如，如果想在 Ubuntu GNOME 桌面环境中安装一个 USB 移动盘，可先把 USB 移动盘插入系统，识别后系统将会把 USB 移动盘自动安装到/media/disk 目录下，同时在 GNOME 桌面中增加一个表示 USB 移动盘的图标“New Disk”（设有卷标名），打开一个“NEW DISK—文件浏览器”窗口，如图 2-74 所示。

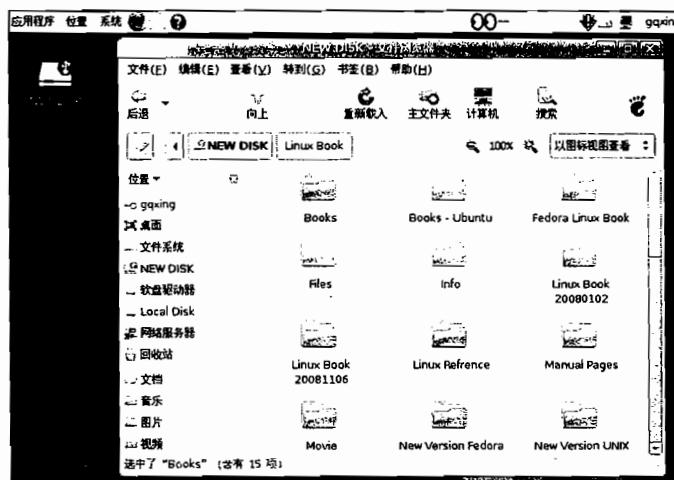


图 2-74 增加 USB 移动硬盘

此时，如果想要复制文件或文件夹，把选定的文件或文件夹拖放到 USB 移动盘图标上即可。也可以双击“New Disk”图标，或者双击文件浏览器中的“New Disk”卷标名，进入 USB 移动盘的文件系统，把文件或文件夹移至适当的目录位置。

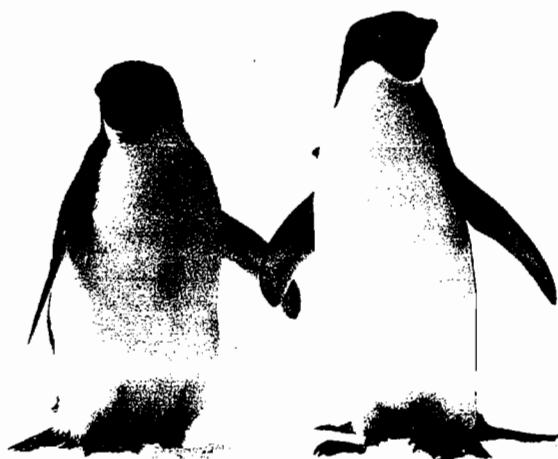


## 第3章

### 命令行基础知识

本章采用 Linux 系统默认的命令解释程序 Bash, 从最基本的命令行界面入手, 介绍 Shell 的命令行结构、标准输入与输出、输入/输出重定向、管道、命令历史与命令别名, 以及作业控制等。其内容主要包括:

- 命令行结构;
- 后台进程;
- 标准输入、标准输出与标准错误输出;
- 输入/输出重定向;
- 管道;
- 元字符与文件名生成;
- 转义与引用;
- 命令历史;
- 命令别名;
- 作业控制;
- 会话记录与命令确认。



Linux 系统提供了大量的命令和工具，如能熟练地掌握最基本的命令，灵活地利用系统提供的各种机制，组合运用 Linux 系统的命令和工具，就能够充分地发挥 Linux 系统的潜能。Linux 系统的强大功能完全体现在其命令行环境中，图形界面（如 GNOME 桌面系统）提供的所有功能实际上也是利用基本的命令和工具实现的。因此，熟练地掌握、灵活自如地运用一定数量的常用命令和工具是每个学习 Linux 系统的人都应当具备的基本功。

要进入命令行环境，在注册到系统之后，从“应用程序”菜单中选择“附件→终端”，即可进入终端仿真窗口（如图 3-1 所示），利用命令行界面访问 Linux 系统。此外，在终端窗口的命令行操作过程中，可以随时使用鼠标选中屏幕上的文本数据，再利用右键的上下文菜单，复制、剪切和粘贴选中的文本数据。

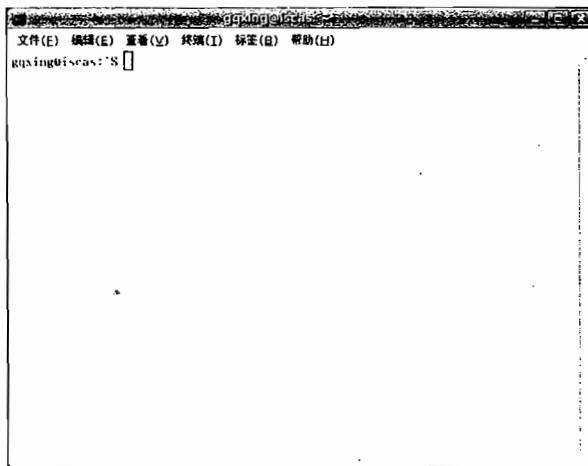


图 3-1 终端窗口

从本章开始，初学者最好使用安装 Ubuntu Linux 系统时创建的用户账号，以普通用户的身份注册到 Linux 系统中，在自己的主目录中开始学习 Linux 系统的各种命令和工具；尽量不要动用系统文件，等到有了一定的基础时，再以超级用户 root 的身份管理 Linux 系统，以免无意中损坏系统。

为了确保总是处于自己的主目录中，可通过输入 cd 命令，接着输入 pwd 命令进行验证。用户的主目录具有下列形式：

```
$ cd
$ pwd
/home/username
$
```

## 3.1 命令行结构

在 Linux 系统中，一个命令通常由命令名、命令选项和命令参数 3 部分内容组成，中间以空格或制表符等空白字符隔开。命令形式如下：

<命令名> <命令选项> <命令参数>

其中，命令选项通常是以减号“-”开始的单个字符。与 UNIX 系统不同的是，Linux 系统还提



供以双减号“-”为起始标志的命令选项（通称 GNU 选项），其选项通常为可按字面意义理解的单个英文单词，或由单词和连字符组成的词组。除了个别命令选项（如“-help”）之外，以双减号“-”为起始标志的命令选项大多是单字符命令选项的同义词，具有相同的意义，因而可以替换使用。

顾名思义，命令选项是可以省略的。同样，命令参数也可以省略。也就是说，在命令行结构中，只有命令名通常是必须提供的。一个最简单的命令可以仅仅包含命令名本身，在这种情况下，命令选项和参数均采用默认值。例如，下面就是一个最简单的命令，其中的命令选项和参数均采用默认值，即列出系统的当前日期和时间。

```
$ date  
2008年07月02日星期三00:31:12CST  
$
```

在实际应用过程中，可以根据具体要求，视情况选用或省略命令选项和参数。而且，命令选项和参数可与命令名以任意形式组合使用。例如，下列命令仅由命令名和一个命令选项“-n”组成，省略了命令参数，其作用是列出系统的名字。

```
$ uname -n  
iscas  
$
```

下列命令由命令名和命令参数组成，而省略了命令选项，其作用是以简单的输出形式列出指定目录下的文件。

```
$ ls /etc/network  
if-down.d if-post-down.d if-pre-up.d if-up.d interfaces  
$
```

取决于命令本身，命令参数可以是目录、文件或其他内容。例如，在下列命令中，ls 是命令的名字，“-l”是命令的选项，/etc/profile 文件是命令的参数。

```
$ ls -l /etc/profile  
-rw-r--r-- 1 root root 497 2008-04-23 01:49 /etc/profile  
$
```

下列命令使用了“-l”和“-a”两个命令选项，但省略了命令参数，其作用是列出当前目录中的所有文件，包括隐藏文件。

```
$ ls -la  
总用量 46964  
drwxr-xr-x 46 gqxing gqxing 4096 2008-07-02 00:32 .  
drwxr-xr-x 3 root root 4096 2008-06-28 00:00 ..  
drwx----- 2 gqxing gqxing 4096 2008-06-28 23:45 .aptitude  
-rw----- 1 gqxing gqxing 3875 2008-07-01 23:43 .bash_history  
-rw-r--r-- 1 gqxing gqxing 220 2008-06-28 00:00 .bash_logout  
-rw-r--r-- 1 gqxing gqxing 2928 2008-06-28 00:00 .bashrc  
....  
$
```

命令选项主要用于限定命令的具体功能，同时也决定了命令的最终运行结果。在 Linux 系统中，每个命令通常均提供大量的选项，因而具有丰富的功能。选项可以被单独给出，也可以被组合使用。如果选项本身也带有参数，则这样的选项及其参数必须被单独列出。在下列排序命令中，因为“-k”和“-o”等命令选项本身也要求提供参数，故需要分别给出。

```
$ sort -k 5 -n -o sorted tobesorted  
$
```

其中，“-k 5”中的 5 就是“-k”选项的参数，表示以第 5 个字段为关键字进行排序。“-n”选项表示按数值的大小排序。“-o sorted”中的 sorted 也是选项“-o”的参数，表示存储最终排序结果的输出文件。最后的 tobesorted 则是命令参数，表示需要排序的输入文件。

在后续章节介绍的 Linux 命令语法格式中，凡是用方括号“[]”形式给出的命令选项均为可

选项，可视具体需求选择使用或忽略。

在 Linux 系统的命令提示符下，一次通常仅输入一个命令。如果愿意，也可以一次输入多个命令，命令之间用分号隔开。例如，下列两个命令的作用是首先进入指定的目录/etc/vsftpd，然后列出其中的文件。

```
$ cd /etc/network; ls -l
总用量 20
drwxr-xr-x 2 root root 4096 2008-04-23 02:02 if-down.d
drwxr-xr-x 2 root root 4096 2008-04-23 01:53 if-post-down.d
drwxr-xr-x 2 root root 4096 2008-04-23 01:49 if-pre-up.d
drwxr-xr-x 2 root root 4096 2008-04-23 02:06 if-up.d
-rw-r--r-- 1 root root 233 2008-06-29 21:02 interfaces
$
```

另外，也可以使用圆括号把若干命令合并在一起，使之构成一个组合命令。

```
$ (cd /etc/network; ls -l)
总用量 20
drwxr-xr-x 2 root root 4096 2008-04-23 02:02 if-down.d
drwxr-xr-x 2 root root 4096 2008-04-23 01:53 if-post-down.d
drwxr-xr-x 2 root root 4096 2008-04-23 01:49 if-pre-up.d
drwxr-xr-x 2 root root 4096 2008-04-23 02:06 if-up.d
-rw-r--r-- 1 root root 233 2008-06-29 21:02 interfaces
$
```

除了括号之外，上述两种命令形式完全一样，有时其效果也完全一样，但两者的意义却大不相同。第一种命令形式只是在一个逻辑行上并列输入了多个命令，其效果同一次输入一个命令基本上没有区别，而且都是在当前 Shell 下运行。而第二种命令形式则把多个命令看作一个组合命令，在一个子 Shell 中运行，所有命令的输出数据将会被合并为一个输出流，其差别在管道操作中尤为明显。

例如，下列两组命令的最终结果是完全不同的（其中，“wc -l”命令用于计算读入的行数）。首先，分别观察 date 与 who 两个命令的输出，如下所示。

```
$ date
2008年07月02日星期三00:36:42CST
$ who
gqxing    tty7      2008-07-01 23:49 (:0)
gqxing    pts/0     2008-07-02 00:25 (:0.0)
gqxing    pts/1     2008-07-02 00:32 (:0.0)
$
```

然后，观察两个并列命令的输出结果，如下所示。

```
$ date; who
2008年07月02日星期三00:36:42CST
gqxing    tty7      2008-07-01 23:49 (:0)
gqxing    pts/0     2008-07-02 00:25 (:0.0)
gqxing    pts/1     2008-07-02 00:32 (:0.0)
$
```

接着，再使用管道把两个并列的命令与计算输入数据行数的 wc 命令连接起来，观察其输出结果。可以看到，wc 命令仅仅计数了 who 命令的输出结果——3 行。

```
$ date; who | wc -l
2008年07月02日星期三00:36:42CST
3
$
```

最后，再利用圆括号把两个命令组合到一起，通过管道与 wc 命令相连接，观察其结果。可以看到，两个命令各自的输出数据已被合并到一起，wc 命令计数的最终结果是 4 行。



```
$ (date; who) | wc -l  
4  
$
```

如果命令较长，超出一个物理行的宽度，可以使用反斜线“\”把命令写到多个物理行上。也可以继续输入，由系统自动延伸至后续行上。例如，下列 read 命令提示用户依次输入名字、电话号码和电子邮件地址，然后将其分别存于 name、phone 和 email 这 3 个变量中。由于命令行较长，其中采用了反斜线“\”，把超长部分延续到下一行，如下所示。

```
$ read -p "Please input name, phone and email address in order: " \  
name phone email  
$
```

如前所述，许多 GNU 实用程序都支持以双减号“--”为起始标志的选项。这些选项或者是原有单减号“-”选项的另外一种表现形式，或者是命令功能的扩充。例如，sort 命令的“-k”选项对应的双减号选项为“--key”。如果使用 GNU 命令形式，则可以把前述的 sort 命令改写如下：

```
$ sort --key=5 -n -o sorted tobesorted  
$
```

## 3.2 后台进程

在 Linux 系统中，Shell 通常以前台形式解释执行用户输入的命令。在 Shell 的命令提示符（超级用户的默认命令提示符为“#”，普通用户的默认命令提示符为“\$”）下，系统将会等待用户输入命令，直至用户按下 Enter 键。然后由 Shell 解释命令行，创建一个新的进程，执行用户提交的命令，最后给出命令的执行结果。

在 Shell 解释执行命令期间，用户需要等待命令执行的完成，中间不能做任何事情。即使一个命令的执行时间过长，中间不需要输入任何信息，不需要监控命令的执行过程时，也只能静等命令执行的完成。

为了解决此问题，Shell 提供了后台进程机制，允许用户以后台进程的方式运行命令，而无需等待命令执行的完成。在解释命令行，以后台进程方式执行命令的同时，Shell 将会立即输出命令提示符，等待用户输入新的命令，从而并发地运行多个命令。

要以后台进程方式运行命令，在命令的后面增加一个“&”符号即可。例如，为了在系统中检索究竟存在多少个名为 core 的文件，可以使用下列形式的命令。

```
$ find / -name core -print 2>/dev/null &  
[1] 3786  
$
```

上述命令行后面的“&”符号告诉 Shell，提交的命令应以后台进程的方式执行，无需等待命令执行的完成。因此，在给出作业号和进程 ID 之后，系统将会立即输出命令提示符，提示用户进一步输入其他命令。

在上述例子中，方括号中的“1”是以后台作业方式运行的 find 进程的作业号，“3786”是 find 进程的 PID。为了跟踪与控制后台作业，可以利用作业控制命令控制作业的运行状态。例如，可以使用作业控制命令 fg 把后台作业转为前台进程继续运行。有关后台作业控制的详细介绍，请参见 3.10 节。利用进程 ID，或者根据 ps 命令获取的进程信息，也可以使用进程控制命令（如 kill 等命令）控制进程的运行行为。有关进程控制的介绍，详见第 9 章。

如果后台进程有输出数据，其输出信息将会随时出现在用户的终端屏幕上。注意，后台进程的输出信息有可能会出现在交互会话期间的任何时刻，造成屏幕输出的混乱。例如，如果用户正在使用 vi/vim 编辑文件，后台进程的输出很可能会干扰编辑器的正常工作。

## 3.3 标准输入/标准输出与标准错误输出

在 Linux 系统中，任何命令，包括 Shell 本身，通常总是读取来自终端键盘输入的数据，这个数据输入源被称作标准输入（stdin），其文件描述符为 0。命令的运行结果通常总是被输出到用户终端的屏幕上，这个输出目的被称作标准输出（stdout），其文件描述符为 1。另外，在命令的执行期间，如果出现问题，相应的错误信息也将被输出到用户终端的屏幕上，这个输出目的通常被称作标准错误输出（stderr），其文件描述符为 2。

一旦注册到系统中，系统总是为用户打开 3 个默认的文件：标准输入（键盘）、标准输出（终端屏幕）和标准错误输出（也是终端屏幕，用于输出错误信息），如图 3-2 所示。

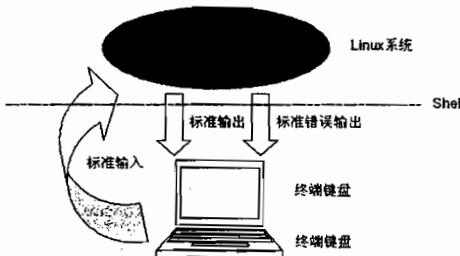


图 3-2 标准输入/标准输出和标准错误输出与终端键盘和屏幕间的关系

例如，当用户从键盘上输入下列 uname 命令时，其输出信息将出现在用户的终端屏幕上。

```
$ uname -a
Linux iscas 2.6.24-19-generic #1 SMP Fri Jul 11 23:41:49 UTC 2008 i686 GNU/Linux
$ uname -n
iscas
$ uname -s
Linux
$
```

当输入下列不带文件参数的 cat 命令时，cat 命令将会读取来自标准输入的数据，并逐字予以显示。也就是说，cat 命令将会等待用户使用键盘输入数据。当按下 Ctrl-D 组合键表示输入结束时，cat 命令将会把读取的数据再次显示在终端屏幕上，如下所示。

```
$ cat -v
input from keyboard
Ctrl-D
input from keyboard
$
```

## 3.4 输入/输出重定向

为了仔细分析命令的处理结果，有时需要把命令的标准输出保存到某个文件中，这就需要用



到 Shell 的输入/输出重定向机制。例如，利用输出重定向符号“>”，下列命令行可以把“ls -l”命令的输出结果保存到指定的文件中。

```
$ ls -l > fname  
$
```

在上述命令中，“> fname”意味着把“ls -l”命令的输出数据重定向并写到指定的文件 fname 中。如果指定的文件不存在，Shell 将会创建一个新的文件；然后把输出数据保存到其中；如果文件已经存在，文件中原有的内容将会被清除，并代之以命令的输出数据。

为了保留文件中原有的数据，把命令的输出数据附加到文件的后面，可以使用重定向符号“>>”，如下所示。

```
$ ls -l >> fname  
$
```

其中，“>> fname”意味着把“ls -l”命令的输出数据重定向并附加到指定文件 fname 的后面。同样，如果指定的文件不存在，Shell 将会创建一个新的文件，并把输出数据保存到其中；如果文件已经存在，Shell 将会把命令的输出数据附加到文件的后面，以保留文件中原有的内容。

任何命令（包括 Shell 本身）的标准输入都可以重定向，使命令直接读取某个文件而不是键盘输入。例如，wc 命令的功能是读取标准输入中的输入数据，分别计数输入数据中的字符数、字数和行数。为了使 wc 命令能够直接读取某个文件中的数据内容，可以使用重定向符号“<”，使之直接读取指定的文件，如下所示。

```
$ wc -l < fname  
42  
$
```

在 Linux 系统中，标准输入、标准输出和标准错误输出这 3 个文件通常总是打开的。这 3 个文件，包括其他打开的文件均可做 I/O 重定向处理。所谓的 I/O 重定向，只不过是由 Shell 读取或者捕捉来自文件、命令、程序或脚本中的输出，作为输入或输出信息传递给另外一个文件、命令、程序或脚本。

在 Linux 系统中，系统将会为每一个打开的文件分配一个文件描述符，每个文件都与一个文件描述符相关联。文件描述符是一个数字，便于 Linux 系统跟踪打开的文件（可以把文件描述符看作简化的文件指针）。标准输入、标准输出和标准错误输出的文件描述符分别是 0、1 和 2。

作为一种临时性的双向机制，把其他打开文件的描述符分配到标准输入、标准输出和标准错误输出，然后利用这些文件描述符执行输入/输出，有时是非常有用的。（注意，使用文件描述符 5 可能会引起问题。这是因为，当 Shell 利用 exec 命令创建子进程时，子进程将会继承文件描述符 5。因此，最好避开这个特殊的文件描述符。）

表 3-1 给出了 Shell 支持的各种 I/O 重定向的规定及其说明。

I/O 重定向	Shell I/O 重定向
<fname	使用指定的文件作为标准输入（其文件描述符为 0），以便从指定的文件中接收输入数据
>fname	使用指定的文件作为标准输出（其文件描述符为 1）。如果文件不存在，则创建命名的文件。如果文件存在，且 noclobber 标志已经设置，将会产生错误；否则，将会清除文件中原有的数据内容（参见第 7 章中介绍的 set 命令）
>>fname	除了忽略 noclobber 标志之外，其功能与“>fname”相同
<>fname	使用指定的文件作为标准输出。如果文件存在，则把输出内容附加到文件后面；否则，创建指定的文件
<>fname	以读写方式打开指定的文件，并使之作为标准输入

续表

I/O 重定向	简单说明
<< -fstr	以指定的标志字符中 fstr 之后的文档（称作 Here 文档）作为标准输入，从 fstr 之后逐行读取数据，直至遇到第二个 fstr（或 EOF）标志。此时，第一个 fstr 是标准输入的起始标志，第二个 fstr（或 EOF）是标准输入的结束标志。如果“<<”后面附带减号“-”标志字符，则 Shell 将会忽略后随文本行前面的制表符
<&digit	使用指定的文件描述符复制一个标准输入
>&digit	使用指定的文件描述符复制一个标准输出
<&-	关闭标准输入，而“n<&-”则表示关闭输入文件描述符 n
>&-	关闭标准输出，而“n>&-”则表示关闭输出文件描述符 n
<&j	把标准输入重定向到文件描述符 j 表示的输入文件中
>&j	把标准输出重定向到文件描述符 j 表示的输出文件中
&>fname	把标准输出和标准错误输出均重定向到指定的文件中

如果 I/O 重定向符号“<”或“>”前面有一个数字，则表示相应的文件描述符（默认值分别为 0 或 1）对应的文件，如表 3-2 所示。

表 3-2 Shell I/O 重定向

I/O 重定向	简单说明
0<fname	把标准输入重定向到指定的文件中
1>fname	把标准输出重定向到指定的文件中
1>>fname	把标准输出重定向并附加到指定的文件中
2>fname	把标准错误输出重定向到指定的文件中
2>>fname	把标准错误输出重定向并附加到指定的文件中
i>&j	把文件描述符 i 表示的输出文件重定向到文件描述符 j 表示的文件中
[ij]<fname	以读写方式打开指定的文件，并把文件描述符 j 分配到指定的文件。如果文件不存在，则创建该文件。如果未指定文件描述符 j，则表示默认的文件描述符 0，即标准输入

采用下列命令形式，可以把标准错误输出重定向到标准输出，使命令的错误信息和命令的实际输出数据均被写到标准输出中，即在终端屏幕上显示。

**command 2>&1**

在下面的例子中，前 3 个 echo 命令的标准输出已重定向到 script.log 文件，因而其输出信息将会被写到文件中，而不是出现在终端屏幕上。

```
$ LOGFILE=script.log
$ echo "This line is written to log file." > $LOGFILE
$ echo "This line is appended to log file." >> $LOGFILE
$ echo "This line is appended to log file also." >> $LOGFILE
$ echo "This line is echoed to stdout."
This line is echoed to stdout.
$ cat script.log
This line is written to log file.
This line is appended to log file.
This line is appended to log file also.
$
```

在下面的例子中，前两个命令（变量赋值语句除外）的标准错误输出已重定向到 script.errors 文件，因此，命令的错误信息将会被记录到指定的文件而不是写到终端屏幕上。而第 3 个命令的错误信息将会被写到标准错误输出，即终端屏幕上，而不会出现在 script.errors 文件中。

```
$ ERRFILE=script.errors
$ bad_command1 2>$ERRFILE
```



```
$ bad_command2 2>>$ERRFILE  
$ bad_command3
```

采用下列形式，可以把多个 I/O 重定向组合到一个命令行中。

```
command <input-file> >output-file
```

采用下列形式，可以把标准输出和标准错误输出重定向到同一个文件中。

```
command >command.log 2>&1
```

其中，命令的任何错误信息将会被写到 command.log 文件中，因为标准错误输出已经重定向到该文件。

注意，I/O 重定向的顺序是非常重要的。Shell 将会根据文件描述符（文件）出现的顺序决定 I/O 重定向的关联关系。例如，下列 I/O 重定向命令意味着把命令的标准输出和标准错误输出均重定向到给定的文件中。

```
command 1>fname 2>&1
```

假定 I/O 重定向之前命令的标准输入、标准输出和标准错误输出对应的文件描述符如图 3-3 所示。

Shell 将首先建立文件描述符 1（即终端屏幕）与文件 fname 之间的重定向关系，也即把 fname 对应的文件描述符 25000 写到用户打开文件表中的标准输出位置。然后建立文件描述符 2 与文件描述符 1 表示的文件（即 fname）之间的重定向关系，也即把标准输出对应的文件描述符 25000 写到用户打开文件表中的标准错误输出位置。最终的结果是，命令的标准输出和标准错误输出内容均被写到文件 fname 中，如图 3-4 所示。

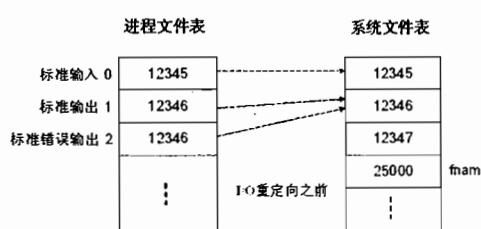


图 3-3 I/O 重定向之前

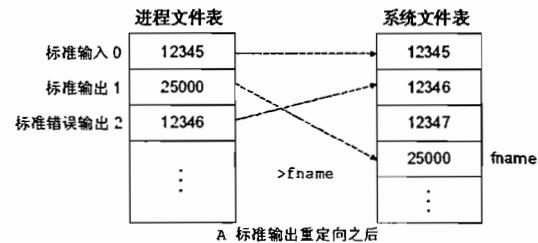


图 3-4 I/O 重定向之后

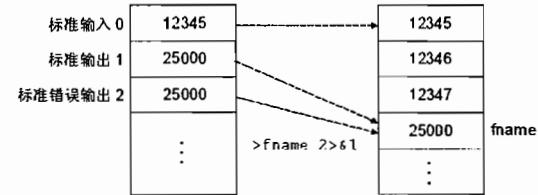


图 3-4 I/O 重定向之后

如下所示，如果 I/O 重定向的顺序相反，则最终结果将会是另外一个样子。

```
command 2>&1 1>fname
```

假定 I/O 重定向之前命令的标准输入、标准输出和标准错误输出对应的文件描述符仍如图 3-3 所示。文件描述符 2 首先与文件描述符 1 建立重定向关系。由于两者对应的文件描述符相同（均为 12346），故标准错误输出对应的文件描述符保持不变。然后，文件描述符 1 再与给定的文件 fname 建立重定向关系，把 fname 对应的文件描述符 25000 写到命令打开文件表中的标准输出位置。

最终结果是，命令的标准错误输出将会被写到标准输出，即在终端屏幕上显示。而命令的标准输出则会被写到给定的文件 fname 中，如图 3-5 所示。

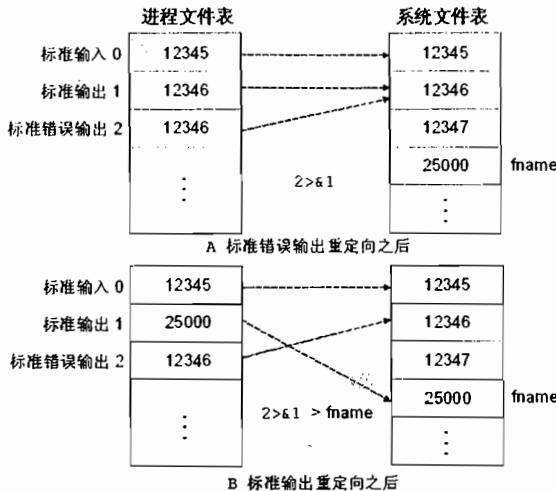


图 3-5 I/O 重定向之后

由此可见，如果 I/O 重定向的顺序不同，最终的结果将会完全不一样。例如，执行下列命令时，错误信息将会被输出到标准输出，即终端屏幕上，而不会被写到 command.log 文件中。

```
$ ls -yz 2>&1 >> command.log
ls: invalid option -- y
Try 'ls --help' for more information.
$
```

利用下列命令和 I/O 重定向的方法，可以创建一个新的空文件。如果文件存在，则清空文件 fname 中原有的内容。

```
$ > fname
$
```

利用 I/O 重定向的方法，还可以创建一个新文件，使之包含当前目录下的目录文件树列表，例如：

```
$ ls -lR > dir-tree
$
```

注意，子进程能够继承已经打开的文件描述符。为了防止文件被子进程继承，应注意随时关闭不再需要的文件描述符。

## 3.5 管道

在 Linux 系统中，管道是一种先进先出的单向数据通路。利用管道符号 “|”，可以把一个命令的标准输出连接到另一个命令的标准输入。例如，利用管道把 ls 和 wc 两个命令连接到一起，可以获知指定目录下的文件数量（“-w”选项表示以字为单位进行计数），如下所示。

```
$ ls /usr | wc -w
9
$
```

从上述命令的最终执行效果看，可以把组合命令分解为以下两个命令。

```
$ ls /usr > fname; wc -w < fname
9
```



\$

由此可以看出，当使用管道方式连接两个命令时，Shell 将会把两个进程连接起来，利用管道的单向通信特征，把一个进程的标准输出传递到另一个进程的标准输入。Shell 将会协调两个进程的同步，使两个进程能够并发地运行，这样就可以省略存储中间处理结果的临时文件。实际上，管道是一种特殊的 I/O 重定向。

管道的常见用法是为滤通程序提供原始数据，由滤通程序读取来自标准输入的数据，按照指定的检索原则和模式，从输入数据中提取期望的、包含给定字符串的数据。在 Linux 系统中，grep 就是这样一个常见的滤通程序。

例如，为了从 ps 命令输出的众多进程中找出某个特定的进程，可以使用管道连接 ps 和 grep 命令，如下所示。

```
$ ps -ef | grep cron
root      5791      1  0 12:00 ?        00:00:00 /usr/sbin/cron
gqxing    6806  3645  0 12:44 pts/1    00:00:00 grep cron
$
```

另外一个常见的用法是利用管道把进程的输出数据传递给 sort 命令，使之按照一定的排序原则进行排序，最终输出排序后的结果。例如，下列组合命令最终将会按照字符顺序输出注册的用户。

```
$ who | sort
cathy     pts/1    2008-12-23 15:17 (bogon)
gqxing    pts/0    2008-12-23 10:48 (:0.0)
gqxing    pts/2    2008-12-23 10:56 (bogon)
gqxing    tty7    2008-12-23 10:14 (:0)
$
```

利用管道，可以把多个命令组合到一起，把命令的标准输出依次传递到下一个命令的标准输入，最终得到经过多个命令依次处理的结果，如下所示。

```
command1 | command2 | command3 > output-file
```

若要依次加工处理多个命令、脚本和程序的输出数据，管道是非常有用的。例如，为了获取 cron 的进程 ID，以便使用 kill 命令终止该进程，可以利用管道连接下列命令，首先从进程列表中提取与 cron 有关的进程，然后删除可能存在的“grep cron”进程，最后再使用 gawk 命令截取位于第 2 个字段的进程 ID（参见第 9 章中介绍的简化版的 pgrep 和 pidof 命令），如下所示。

```
$ ps -ef | grep cron | grep -v grep | gawk '{print $2}'
5791
$
```

为了复制和备份一个完整的目录，也可以利用管道，组合使用 find 和 cpio 两个命令，把当前目录下的所有目录和文件按照原有的目录层次结构复制到一个新的目录位置，如下所示。

```
$ cd sourcedir
$ find . -print | cpio -pduv newdir
$
```

Linux 系统还提供了一个相当于三通管的实用程序 tee。tee 命令的主要功能是通过标准输入接收并显示数据，同时把数据存储到指定的文件中。因此，可以利用 tee 命令，在显示一个命令输出数据的同时，把输出结果存储到一个指定的文件中，以便将来再检查。例如，为了显示并保存当前所有注册用户的列表文件，可以使用下列命令（其效果如图 3-6 所示）。

```
$ who | tee userlist
cathy     pts/1    2008-12-23 15:17 (bogon)
gqxing    pts/0    2008-12-23 10:48 (:0.0)
gqxing    pts/2    2008-12-23 10:56 (bogon)
gqxing    tty7    2008-12-23 10:14 (:0)
$
```

tee命令的“T”型三通功能

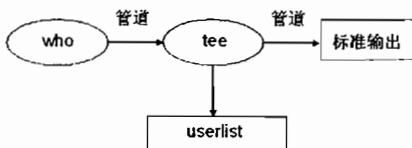


图 3-6 tee 命令功能图解

## 3.6 元字符与文件名生成

在 Linux 系统中，很多命令均使用文件作为命令参数。例如，下面的 ls 命令用于列出指定文件参数 atmmon.c 的访问权限、文件大小及文件属主等有关属性。

```
$ ls -l atmmon.c
-rw-r--r-- 1 ggxing ggxing 4589 2008-07-17 12:56 atmcom.c
$
```

当需要处理一组具有共同属性的文件时，怎样指定文件名参数呢？Shell 提供了一种文件名生成机制，使用户能够利用元字符（或称通配符）实行模式匹配，最终生成一个具有同一属性的文件列表。

为了简化手工输入，按照某种模式选择具有同一属性的文件，Shell 的文件名生成机制是非常有用的。例如，可以使用下列命令列出当前目录中的所有 C 程序文件。

```
$ ls -l *.c
-rw-r--r-- 1 ggxing ggxing 4589 2008-07-17 12:56 atmcom.c
-rw-r--r-- 1 ggxing ggxing 1668 2008-07-17 12:50 atmmon.c
-rw-r--r-- 1 ggxing ggxing 2790 2008-07-17 13:00 atmstat.c
-rw-r--r-- 1 ggxing ggxing 2764 2008-07-17 12:59 handler.c
-rw-r--r-- 1 ggxing ggxing 3198 2008-07-17 12:59 listerner.c
-rw-r--r-- 1 ggxing ggxing 4660 2008-07-17 12:53 scanner.c
$
```

Shell 命令使用生成的文件名作为参数，依次处理每一个文件。上述命令的处理结果就是依次列出当前目录下每一个以 “.c” 为文件名后缀的 C 程序文件。其中的星号 “\*” 就是一个元字符，可以匹配任何字符或字符串，包括空字符串。表 3-3 给出了 Shell 支持的与文件名生成有关的元字符及其说明。注意，元字符可以组合使用。

表 3-3 与文件名生成有关的元字符

元字符	简单说明
*	可以匹配任何数量的字符或字符串，包括空字符串。例如，“boc*”表示任何一个以“boc”为起始字符的字符串，“*.c”表示任何一个以“.c”为文件名后缀的 C 程序文件
?	可以匹配相应位置的任何一个字符。例如，“file?”表示任何一个以“file”为起始字符，后面附加单个字符的字符串
[...]	由方括号定义的字符集或字符范围，可以使用其中任何一个字符匹配文件名相应位置的一个字符。方括号中的字符集可以由任何字符组成，数量不限。字符可以一一列举，也可以在两个字符之间加一个减号“-”，表示一个字符范围。例如，[a-z]表示所有的小写字母，[0-9]表示任何数字
[!...]或[^...]	如果方括号中的第一个字符是感叹号 “!” 或上箭头 “^” 字符，则其意义恰好相反，表示可以匹配任何一个不属于给定字符集范围的字符

假定当前目录存在下列文件，我们可通过例子进一步说明怎样使用元字符匹配文件名，解释 Shell 的文件名生成机制。



```
$ ls -l  
总用量 32  
-rwxr-xr-x 1 gqxing gqxing 524 2008-06-28 10:54 Bubble  
-rwxr-xr-x 1 gqxing gqxing 789 2008-06-28 10:56 btree  
-rw-r--r-- 1 gqxing gqxing 1663 2008-06-28 18:56 file1  
-rw-r--r-- 1 gqxing gqxing 2556 2008-06-28 18:57 file2  
-rwxr-xr-x 1 gqxing gqxing 919 2008-06-28 10:54 ftpget  
-rwxr-xr-x 1 gqxing gqxing 184 2008-06-28 10:55 greeting  
-rwxr-xr-x 1 gqxing gqxing 603 2008-06-28 10:55 prime  
-rwxr-xr-x 1 gqxing gqxing 640 2008-06-28 10:53 whatday  
$
```

按照表 3-3 的说明, [a-z]可以匹配任何小写字母, “\*”可以匹配任何字符串, 故下面的命令可以列出当前目录下任何以小写字母为起始字符的文件名。

```
$ ls -l [a-z]*  
-rwxr-xr-x 1 gqxing gqxing 789 2008-06-28 10:56 btree  
-rw-r--r-- 1 gqxing gqxing 1663 2008-06-28 18:56 file1  
-rw-r--r-- 1 gqxing gqxing 2556 2008-06-28 18:57 file2  
-rwxr-xr-x 1 gqxing gqxing 919 2008-06-28 10:54 ftpget  
-rwxr-xr-x 1 gqxing gqxing 184 2008-06-28 10:55 greeting  
-rwxr-xr-x 1 gqxing gqxing 603 2008-06-28 10:55 prime  
-rwxr-xr-x 1 gqxing gqxing 640 2008-06-28 10:53 whatday  
$
```

由于问号 “?” 可以匹配任何一个字符, 故下列 ls 命令可以列出当前目录中文件名前 4 个字符为 file, 第 5 个字符为任何字符的所有文件。

```
$ ls -l file?  
-rw-r--r-- 1 gqxing gqxing 1663 2008-06-28 18:56 file1  
-rw-r--r-- 1 gqxing gqxing 2556 2008-06-28 18:57 file2  
$
```

要列出当前目录中以 p 或 w 为首字符的所有文件, 可以使用下列 ls 命令。

```
$ ls -l [pw]*  
-rwxr-xr-x 1 gqxing gqxing 603 2008-06-28 10:55 prime  
-rwxr-xr-x 1 gqxing gqxing 640 2008-06-28 10:53 whatday  
$
```

上述命令也可以改写如下。

```
$ ls -l p* w*  
-rwxr-xr-x 1 gqxing gqxing 603 2008-06-28 10:55 prime  
-rwxr-xr-x 1 gqxing gqxing 640 2008-06-28 10:53 whatday  
$
```

要列出当前目录中首字符为大写字母 (或其他非小写字母) 的所有文件, 则可以使用下列 3 种 ls 命令形式之一。

```
$ ls -l [A-Z]*  
-rwxr-xr-x 1 gqxing gqxing 524 2008-06-28 10:54 Bubble  
$ ls -l [!a-z]*  
-rwxr-xr-x 1 gqxing gqxing 524 2008-06-28 10:54 Bubble  
$ ls -l [^a-z]*  
-rwxr-xr-x 1 gqxing gqxing 524 2008-06-28 10:54 Bubble  
$
```

如上所述, 问号 “?” 可以匹配当前目录中以单个字符命名的所有文件名。星号 “\*” 能够匹配当前目录中的所有文件名 (隐藏文件除外)。如果不存在能够匹配检索模式的文件名, 指定的检

索模式将会不加修改地作为参数被传递给相应的命令。例如下列命令所示。(仅当目录为空时, 下述第二个例子才会出现如此的输出结果。)

```
$ ls -l ?
ls: cannot access ?: No such file or directory
$ ls -l *
ls: cannot access *: No such file or directory
$
```

元字符也可用于检索文件。例如, 可以使用下列命令检索并列出/home/gqxing 目录下任何子目录中名为 core 的文件。

```
$ echo /home/gqxing/*/core
```

注意, 任何元字符都不能匹配以句点“.”为首字符的隐藏文件名。换言之, 以句点“.”为起始字符的隐藏文件名必须采用明显的匹配形式。例如, 下列命令只能列出当前目录中的所有非隐藏文件。

```
$ echo *
Bubble btree file1 file2 ftpget greeting prime whatday
$
```

为了匹配以句点“.”为起始字符的隐藏文件名, 可以采用下列命令形式, 列出所有的隐藏文件。

```
$ echo .*
... .aptitude .bash_history .bash_logout .bashrc ....
$
```

注意, 使用 set 命令的“-f”选项(即“set -f”命令)能够禁止文件名的生成。当 Shell 无法解释元字符时, 应注意检查是否设置了这个标志。

## 3.7 转义与引用

转义和引用是两个截然相反的概念。在 Shell 中, 为了处理具有特殊意义的元字符, 如“<”、“>”、“\*”、“?”、“|”和“&”等, 使之作为普通字符显示, 可以采用转义符号“\”、单引号和双引号引用元字符, 而引用的元字符则失去其特殊意义。

根据上述说明, 本身具有特殊意义的元字符, 如果在前面加上转义符号“\”, 则失去其特殊意义。而对于某些普通字符, 如果前面加上转义符号“\”, 则具有特殊的意义, 这些字符被称作转义字符。表 3-4 给出了 Shell 支持的, 具有特殊意义的部分转义字符。

表 3-4

Shell 支持的部分转义字符

转义字符	简单说明
\a	生成声音提示
\b	退格符
\e	Esc 字符
\f	换页符
\n	换行符
\r	回车符
\t	制表符
\v	竖向制表符
\	反斜线



续表

转义字符	简单说明
\	单引号
\nnn	采用1、2或3位八进制数值表示的等价ASCII字符
\xHH	采用1或2位十六进制数值表示的等价ASCII字符
\cX	Ctrl-X字符

表3-4中所示的转义字符可用于echo等命令，以便控制输出或显示的格式等。下面的例子说明了怎样在一个echo命令中使用转义字符显示一个选择菜单。

```
$ echo -e "\n\t\t == Command Menu ==\n\t1. Data processing\n\t2. Print report\n\t3. end"
```

```
Command Menu\n1. Data processing\n2. Print report\n3. end
```

```
$
```

根据上一节的介绍可知，Shell中的元字符都具有特殊的含义，无法直接使用。要引用元字符本身，可以采用3种形式。一种是在元字符前面加转义符号“\”，用以表示字符文字本身，而非具有特殊意义的元字符。

例如，下列两个echo命令的最终结果大不相同。

```
$ echo *\nBubble btree chkroot ftpget greeting prime whatday\n$ echo \'*\n*\n$
```

转义符号“\”是一种引用单个（特殊）字符的最佳方法。一个（特殊）字符前如果增加了转义符号，等于告诉Shell，随后的字符应按普通字符文字解释。例如，“\\*”、“\\$”、“\\”、“\”以及“\”等分别表示“\*”、“\$”、“\”、“\”以及“\”等字符本身，参见下面的例子。

```
$ echo "Hello"\nHello\n$ echo "\"Hello\\\"", he said.\n"Hello", he said.\n$ echo "\$var"                                # "\$" 后面的变量不作解释\n$var\n$ echo "\\\"\\\"\n\\$\n$
```

由此可见，要引用单个字符，使用转义符号“\”是最方便的方法。但当需要引用多个字符时，“\”就显得非常笨拙了。因此，Shell提供了第二种引用方式，即采用单引号的方式引用元字符。在此情况下，单引号之间的所有字符（包括元字符）均按普通文字本身解释。例如，要引用一个字符串“\*\*”，可以在“\*\*”前后增加一对单引号，如下所示。

```
$ echo xx'*'*'yy\nxx**yy\n$
```

这意味着，单引号中的所有元字符（反向单引号“`”和上述的转义字符除外）均作为文字常量处理。因此，利用单引号可以同时引用多个元字符或字符串，如下所示。

```
$ echo '*.* $var "testing"'
*.c $var "testing"
$
```

从上述例子中还可以看出，单引号“`i`”不允许使用美元符号“\$”引用变量的值，其中的`$var`仅表示其本身。

在实际应用中，使用单引号引用参数或变量，其作用是防止字符串中的特殊字符被 Shell 提前解释或扩展。例如，下面的命令表示列出所有以大写字母 B 或小写字母 b 为起始字符的文件。

```
$ ls -l [Bb]*
-rwxr-xr-x 1 gqxing gqxing 524 2008-06-28 10:54 Bubble
-rwxr-xr-x 1 gqxing gqxing 789 2008-06-28 10:56 btree
$
```

如果在 “[Bb]\*” 前后增加单引号，则可理解为引用元字符本身，也就是把 “[Bb]\*” 作为一个字符串解释。在下面的例子中，单引号中的 “[Bb]\*” 就是作为一个文件名解释的，因而找不到匹配的文件。

```
$ ls -l '[Bb]*'
ls: cannot access [Bb]*: No such file or directory
$
```

在 Linux 系统中，某些命令或实用程序会重新解释或扩展使用引号传递的字符串中的特殊字符。使用引号引用参数或变量的一个重要用途就是确保能够把其中含有的特殊字符原封不动地传递给被调用的实用程序，由实用程序进行解释或扩展，例如下列命令所示。

```
$ grep '[Ff]irst' [Bb]*
Bubble:# First line should be #!/bin/bash.
btree:# The first line is just a comment
$
```

假定存在下列文件，怎样使用元字符匹配并删除 “test file”？

```
$ ls -l test*
-rw-r--r-- 1 gqxing gqxing 0 Sep 6 15:04 test file
-rw-r--r-- 1 gqxing gqxing 0 Sep 6 15:04 test.file
$
```

如果使用问号“?”或星号“\*”，将会误删 `test.file` 文件。此时可以使用转义符号加空格的形式匹配 “`test file`”，表示 `test` 与 `file` 之间的空格并非字段分隔符，如下所示。

```
$ rm test\ file
$ ls -l test*
-rw-r--r-- 1 gqxing gqxing 0 Sep 6 15:04 test.file
$
```

实际上，也可以使用 “`rm test" "file`” 或 “`rm test' 'file`”（使用双引号或单引号括住空格字符）的命令形式删除名字中间包含空格的文件。

第三种方式是利用双引号引用字符串，防止部分（但并非全部）元字符提前解释。在此情况下，除了 “!”、“\$”、“!”、“\” 和 “{” 字符之外，其他元字符均按文字本身处理，如下所示。

```
$ echo "The current working directory of **$LOGNAME** is `pwd`"
The current working directory of **gqxing** is /home/gqxing/script
$
```

表 3-5 总结了 3 种元字符引用方式的不同处理效果。

表 3-5 3 种元字符引用的效果

元字符 引用方式	\	\$	*	?	"	'	
\	不解释						
'	不解释	不解释	不解释	不解释	不解释		不解释
"	解释	解释	不解释	不解释		不解释	解释



## 3.8 命令历史

Bash 以及其他大多数 Shell（如 Korn Shell、TC Shell、C Shell 和 Z Shell 等）均支持命令历史机制，以便维护用户输入的命令。Shell 的命令历史机制和编辑功能使用户能够重复利用先前输入的命令，提高用户的交互访问能力。利用 Shell 的命令历史机制，能够不加修改地重复执行先前提交的任何命令，或者在先前命令的基础上，经过校正命令中的细小打字错误或稍加编辑后组成一个新的命令，然后再重复执行先前的命令。

命令历史机制主要是由 Shell 提供的下列内部命令和环境变量实现的。

- ❑ **fc**: 用于列出 (“-l” 选项)、编辑 (“-e” 选项) 或重新执行命令历史文件中记录的命令。
- ❑ **history**: 用于列出命令历史缓冲区或文件中记录的命令。
- ❑ **HISTFILE**: 用于指定命令历史文件。使 Shell 能够在停止运行之前把缓冲区中的命令历史记录写入指定的文件，以便在下一次启动时 Shell 能够读回其中保存的、上一次会话期间执行的命令历史记录。如果 HISTFILE 变量未被定义，或者定义的文件没有“可写”的访问权限，默认的命令历史文件为\$HOME/.bash\_history。
- ❑ **HISTSIZE**: 指定命令历史文件的大小。用于限定当前会话期间需要保存到命令历史文件中的命令数量。如果 HISTSIZE 变量未被定义，则文件容量的默认值为 500，即保存最近执行的 500 条命令。
- ❑ **HISTFILESIZE**: 指定命令历史文件的大小。用于限定不同会话之间需要保存到命令历史文件中的命令数量。如果 HISTFILESIZE 变量未被定义，则文件容量的默认值为 500，即保存最近执行的 500 条命令。

### 3.8.1 fc 命令

利用特殊的内置命令 **fc**，可以按照命令序号或利用命令的起始字符（或字符串）显示、编辑或运行先前执行的命令。在使用 **fc** 命令列举命令历史缓冲区或文件中的命令时，可以指定单个命令，也可以指定一个命令范围。

实际上，Shell 的命令历史机制主要是由 **fc** 内置命令实现的。**fc** 命令允许用户显示、不加编辑或稍加编辑地重新执行命令历史缓冲区或文件中保存的命令，其语法格式如下。

```
fc [ -e ename ] [ -nlr ] [ first [ last ] ]
fc -s [ old=new ] [ command ]
```

**fc** 命令的第一种语法格式表示从用户输入的命令历史缓冲区或文件中选择指定范围（从 **first** 到 **last**）的命令。范围的上限不能超过 **HISTSIZE** 变量指定的值。**first** 和 **last** 既可以是一个字符串，用于匹配最近执行的、以给定的值为起始字符串的命令；也可以是命令在命令历史缓冲区或文件中的序号（如果在序号前加一个负号 “-”，则表示相对于当前命令的偏移值）。如果未指定 **last**，则其默认值为 **first**。如果 **first** 和 **last** 均未被指定，则其默认值分为两种情况：在编辑命令历史缓冲区或文件时仅选择前一个命令，在显示命令历史缓冲区或文件时（使用 “-l” 选项）选择最近执行的 16 个命令，即从前一个命令开始回溯 16 个命令。如果使用 “-l” 选项，将会在标准输出

中列出选择的命令。“-n”选项意味着在列出选定范围的命令时，省略命令在命令历史文件记录中的序号。“-r”选项意味着由近至远反向输出选定范围的命令。

在第一种命令形式中，如果“-nlr”3个选项均未被指定，则调用指定或默认的编辑器，编辑命令历史缓冲区或文件中的命令。命令的范围由 first 和 last 确定。

“-e”选项用于定义在校正或编辑先前的命令时使用的编辑器。如果未指定编辑器的名字，则使用 FCEDIT 变量的值作为默认的编辑器。如果未设置 FCEDIT 变量，则使用 EDITOR 变量的值作为编辑器，否则，最终使用 nano 作为默认的编辑器。在完成命令的编辑之后，退出编辑器时即可输出并重新执行刚才校正过的命令。注意，如果选择编辑多个命令时，在退出编辑器时将会依次执行所选择的所有命令。

例如，要列出命令历史缓冲区或文件中序号为 10~20 的命令，可以使用下列命令。

```
$ fc -l 10 20
10      . setenv
11      set
12      ls -l
13      cd src
14      vim Makefile
15      vim atmmon.c
16      vim listener.c
17      vim scanner.c
18      make
19      atmmon
20      ps -ef
$
```

要列出最近输入的 10 条命令，可以使用下列命令。

```
$ fc -l -10
466      ps -ef
467      pstree -p
468      ipcs -m
469      ipcs -q
470      pmap -x 6312
471      ls -l /etc .
472      cat /etc(exports
473      vim /etc(exports
474      exportfs -r
475      fc -l 10 20
$
```

要列出最近一次输入的以 cat 为起始字符串的命令，可以使用下列命令。

```
$ fc -l cat
472      cat /etc(exports
473      vim /etc(exports
474      exportfs -r
475      fc -l 10 20
475      fc -l -10
$
```

要利用 vim 编辑并执行序号为 10~20 的命令，可以使用下列 fc 命令。

```
$ fc -e vim 10 20
```

第二种命令形式表示跳过编辑阶段。如果存在“old = new”形式的字符串替换，则由 Shell 直接执行命令行替换，然后重新执行编辑后的新命令。如果未给出命令，则表示执行之前刚执行



的命令。例如，要重复执行先前的 ls 命令，可以使用下列 fc 命令。

```
$ ls -l /etc/profile  
-rw-r--r-- 1 root root 497 Jul 25 11:50 /etc/profile  
$ fc -s  
ls -l /etc/profile  
-rw-r--r-- 1 root root 497 Jul 25 11:50 /etc/profile  
$
```

要重复执行先前的第 115 号命令，可以使用下列 fc 命令（调用 which 命令，查询匹配的 echo 命令）。

```
$ fc -s 115  
which echo  
/bin/echo  
$
```

又如，在列出/home/gqxing/incl 目录中的文件之后，如果还想查询与 incl 位于同一层次的 src 目录中的文件时，可以使用“incl=src”修正先前的 ls 命令，然后再执行替换后的 ls 命令，如下所示。

```
$ ls -l /home/gqxing/incl  
total 8  
-rw-r--r-- 1 gqxing gqxing 1682 Dec 26 15:15 atm.h  
-rw-r--r-- 1 gqxing gqxing 2490 Dec 26 15:16 event.h  
$ fc -s incl=src  
ls -l /home/gqxing/src  
total 32  
-rw-r--r-- 1 gqxing gqxing 4589 Dec 26 12:56 atmcom.c  
-rw-r--r-- 1 gqxing gqxing 1668 Dec 26 12:50 atmmon.c  
-rw-r--r-- 1 gqxing gqxing 2790 Dec 26 13:00 atmstat.c  
-rw-r--r-- 1 gqxing gqxing 2764 Dec 26 12:59 handler.c  
-rw-r--r-- 1 gqxing gqxing 3198 Dec 26 12:59 listerner.c  
-rw-r--r-- 1 gqxing gqxing 4660 Dec 26 12:53 scanner.c  
$
```

### 3.8.2 history 命令

内置命令 history 是 fc 命令的一个特例（在 Korn Shell 中，history 只是利用 fc 命令定义的一个命令别名，即“alias history='fc -l'”），用于读取、显示或清除命令历史记录中的命令。例如，为了列出命令历史缓冲区或文件中记录的命令，可以使用下列命令（在 Bash 中，通常会列出命令历史缓冲区或文件中的所有命令，而在 Korn Shell 中，仅列出 16 条命令）。

```
$ history  
1      ls -l /etc/profile  
2      sudo vi /etc/hosts  
.....  
65     last  
66     history  
$
```

要列出最近执行的 10 条命令，可以使用下列命令。

```
$ history 10  
58    cat /etc/host.conf  
59    sudo vi /etc/resolv.conf  
60    cd /etc/bind  
61    sudo vi named.conf  
62    sudo vi named.conf.options  
63    sudo vi named.conf.local  
64    sudo /etc/init.d/bind9 restart
```

```
65      host www.google.cn
66      last
67      history 10
$
```

要清除命令历史缓冲区中的命令，可以使用下列命令。

```
$ history -c
$ history
1      history
$
```

注意，命令历史机制仅适用于交互式 Shell，不能在 Shell 脚本中使用。

### 3.8.3 重复执行先前的命令

在 Bash 中，为了重复执行先前的命令，可以利用感叹号 “!” 引用机制实现。感叹号 “!” 表示引用命令历史缓冲区或文件中的命令。若要不加修改地重复执行最近刚执行的命令，可以使用 “!!” 命令。

例如，要重复执行刚执行的 ls 命令，可以使用下列 “!!” 命令。

```
$ ls -l /etc/profile
-rw-r--r-- 1 root root 497 Jul 25 11:50 /etc/profile
$ !!
ls -l /etc/profile
-rw-r--r-- 1 root root 497 Jul 25 11:50 /etc/profile
$
```

“!string”命令表示重新执行最近执行的，以给定的 string 为起始字符串的命令。而 “!?string[?]” 则表示重新执行最近运行的，其中包含给定字符串的命令。因此，如果输入 “!gcc” 命令，将会再次执行最近输入的以 gcc 为起始字符串的命令。例如，要重复执行最近一次执行的 uname 命令，可以使用下列命令。

```
$ uname -n
iscas
$ !un
uname -n
iscas
$
```

“!n” 表示重复执行命令历史缓冲区或文件中的第 n 号命令，而 “!-n” 则表示重新执行最近执行的倒数第 n 号命令，如下所示。

```
$ !36
date
Sat Sep 6 19:11:03 CST 2008
$
```

另外，利用 “!:[:g]s/old/new[/:]” 命令，还可以先修正刚执行的命令，然后再执行。即在执行之前，先以给定的字符串 new 替换先前命令中第一个出现的字符串 old，然后再执行校正后的新命令。例如，在执行下述第 2 个命令之前，Shell 首先会以给定的字符串 cat 替换第 1 个命令中首次出现的字符串 file，然后使用替换后的结果作为新的命令提交 Shell 执行：

```
$ file/bin/zcat/
/bin/zcat:Bourne-Again shell script text executable
$ !!:s/file/cat/
cat/bin/zcat
#!/bin/bash
PATH=$(GZIP_BINIR-'/bin'):PATH
exec gzip -cd "$@"
$
```

如果命令行中存在多个匹配的字符串，需要全部替换，可以在 “s” 之前增加一个字符 “g”，表示替换所有匹配的字符串。另外，上述第 2 个命令行中的最后一个斜线字符可以省略。



在 Korn Shell 中，可以利用命令别名“r”，重复执行先前输入的最近一个命令。例如，如果仅仅输入一个“r”字符，则意味着重复执行最近刚执行的命令，如下所示。

```
$ ksh  
$ echo Hello again  
Hello again  
$ r  
echo Hello again  
Hello again  
$
```

为了在 Bash 中也使用 Korn Shell 用户已经习惯的“r”命令，可用利用下列 fc 命令定义一个命令别名，然后即可使用“r”命令。

```
$ alias r='fc -s'
```

或

```
$ alias r='fc -e -'
```

表 3-6 中列出了对先前介绍的“!”命令的总结。

表 3-6 常用的部分“!”命令

命 令	简 单 说 明
!	表示引用命令历史缓冲区或文件中的命令（除非后面紧跟着空格、换行符、等号“=”或左圆括号“(”等字符）
!!	重复执行先前刚执行的命令，相当于输入“! -1”命令
!N	重复执行命令历史缓冲区或文件中序号为 N 的命令
!-N	重复执行从当前命令位置开始倒计数的第 N 个命令
!string	重复执行最近一次执行的，以给定的部分字符串 string 为起始字符串的命令
!?string[?]	重复执行最近一次执行的，包含给定字符串 string 的命令
!!string	引用前一个命令，附加给定的字符串 string，然后重复执行组合后的命令
!N string	引用第 N 个命令，附加给定的字符串 string，然后重新执行组合后的命令
#!	引用迄今为止已经输入的所有字符
!\$	引用前一个命令的最后一个参数
!!:[g]s/old/new/	重复执行先前的命令。在执行之前，先以给定的字符串 new 替换命令中出现的第一个或全部（如果“s”前面有一个字符“g”）字符串 old
^old^new^	重复执行先前的命令（快速替换形式）。在执行之前，先以给定的字符串 new 替换命令中第一个出现的字符串 old

Bash 等 Shell 还提供了其他一些替换与重复执行命令的形式，感兴趣的读者可以查阅联机文档有关 fc 命令的介绍。

### 3.8.4 编辑并执行校正后的命令

在 Bash 中，用户可以利用 Shell 提供的命令历史机制和命令行编辑功能，使用熟悉的编辑器，如 vi (vim) 或 emacs 等，对先前输入的命令进行编辑，从而生成新的命令，然后提交 Shell 执行。在调用 Bash 时，如果使用了“--noediting”选项，将会关闭命令行编辑功能。

进入 Bash 之后，Shell 将按照 FCEDIT 和 EDITOR 变量，以及 emacs 编辑器的顺序确定默认的命令行编辑器。如果未设置 FCEDIT 变量，Bash 将使用 EDITOR 变量的值作为命令行编辑器，如果未设置 EDITOR 变量，emacs 将成为默认的命令行编辑器。在 emacs 编辑模式下，命令行总是处于输入模式，可以利用上下箭头键，翻阅之前输入的命令（这些命令存储在用户主目录下的.bash\_history 文件或由 HISTFILE 变量指定的命令历史文件中）。利用左右箭头键可左右移动光标，插入任何字符，或使用退格键删除光标左边的字符，使用 Delete 键可删除光标所在位置的字

符，编辑命令行。

要使用 vi (vim) 编辑命令行，可以采用下列任何一种设置方式。

```
$ FCEDIT=vi; export FCEDIT
```

或

```
$ EDITOR=vi; export EDITOR
```

另外，也可以使用 set 命令设置默认的命令行编辑器，如下所示。

```
$ set -o vi
```

一旦采取了上述措施之一，即可进入 vi (vim) 命令行编辑模式。命令提示符所在行（通常是终端窗口底部的最后一行）就像一个只有一行数据的编辑窗口。

无论何时，只要按下 Esc 键，即可进入 vi (vim) 命令模式。当处于 vi (vim) 命令模式时，可以使用上下箭头键翻阅之前输入的命令，或者使用“-”或“+”（“j”或“k”键）上下移动命令历史记录，翻阅期望运行的命令。也可以使用“/”、“?”或 G 等 vi (vim) 命令前后检索命令，使用左（或空格）右箭头键左右移动光标。可使用编辑命令校正命令行，如使用“x”删除单个字符，使用“r”替换单个字符，使用“R”替换字符串，使用“i（或 I）”插入字符或字符串，使用“a（或 A）”附加字符或字符串，等等。

完成命令编辑之后，按下 Enter 键即可执行校正后的命令。

与 vi (vim) 编辑模式相比，emacs 编辑模式使用起来更容易。表 3-7 给出了两种编辑模式常用的命令行编辑命令。

表 3-7 常用的命令行编辑命令

vi（命令模式）	emacs	简单说明
上箭头、减号“-”或 k	上箭头或 Ctrl-P	获取前一个命令
下箭头、加号“+”或 j	下箭头或 Ctrl-N	获取下一个命令
左箭头、Ctrl-H 或 h、	左箭头或 Ctrl-B	左移一个字符位置
右箭头、空格键或 i	右箭头或 Ctrl-F	右移一个字符位置
b	Esc B	左移一个字
w	Esc F	右移一个字
退格键（编辑模式）	退格键	删除光标位置左边的一个字符
x	Delete 或 Ctrl-D	删除光标所在位置的一个字符

### 3.8.5 命令行补充

利用 Linux 系统提供的 Readline 库，Bash 还支持命令行补充功能。当输入的命令名、文件名或变量名不完整时，可以使用制表符键“Tab”实现命令的补充，由 Bash 提供名字的剩余部分，从而节省用户的输入时间。该机制也可用来查询和检索记不清的命令。

#### 1. 命令名补充

当输入部分命令名而按下 Tab 键时，Bash 将会按照命令检索路径，搜寻以给定文字为起始字符串的命令。如果找不到匹配的命令，Bash 将会发出鸣叫声。如果恰好发现一个匹配的命令，Bash 将会自动补充命令名的剩余部分。如果发现多个匹配的命令，在 vi（或 vim）编辑模式下，Bash 不会输出任何信息，而在 emacs 编辑模式中，Bash 将会发出鸣叫声。再按下 Tab 键之后，Bash 将会显示一系

列其前缀与用户输入部分匹配的命令，然后允许用户采用同样的方式继续完成命令的选择。

例如，当输入 bz，然后两次按下 Tab 键时，Bash 将会给出 11 个以 bz 为前缀的命令，如下所示。

```
$ bz-(Tab)-(Tab)
bzcat          bzgrep        bzgrep      bzless
bzcmp          bzexe         bzzip2     bzmore
bzdiff         bzfgrep      bzzip2recover
```

如果继续输入字符 c，接着再按 Tab 键两次，Bash 将会给出两个以 bzc 为起始字符串的命令。如果接着输入 a 再按 Tab 键，Bash 将会完成 bzcat 命令的补充，因为此时只有一个匹配的命令，如下所示。

```
$ bzc-(Tab)-(Tab)
bzcat          bzcmp
$ bzca-(Tab)-t ■
$ bzcat ■
```

至此，命令的补充即告完成，可以继续输入其他命令选项和参数了。

## 2. 文件名补充

同样，当用户输入一个文件名的起始部分，接着按下 Tab 键之后，Bash 将会提供文件名的剩余部分。如果用户提供的文件名前缀足以惟一地确定一个文件，Bash 将会显示一个完整的文件名。如果存在多个匹配的文件，Bash 将会尽可能地补充文件名的后续部分，直至遇到某个分界点需要由用户做出进一步的选择。

假定 src 目录中存在两个文件——atm\_status.c 和 atm\_statistics.c，当用户输入 atm 接着按下 Tab 键之后，Bash 将会把文件名补充到 atm\_stat，然后提示用户做出选择，如下所示。

```
$ vim src/atm-(Tab) vim src/atm_stat■
```

此时，用户可以根据文件列表做出适当的选择，如输入“u”或“i”，再按 Tab 键即可完成文件名的补充。但是，如果接着按 Tab 键，Bash 将会重新刷新屏幕，同时显示一系列可选的文件供用户做出抉择，如下所示。

```
$ vim src/atm-(Tab)-(Tab)
atm_statistics.c  atm_status.c
$ vim src/atm_stat■
```

当输入足够的信息（如“u”或“i”），且按下 Tab 键之后，Bash 将会完成文件名的补充，如下所示。

```
$ vim src/atm_stati-(Tab)-stics.c
```

至此，用户可以接着输入其他的命令行参数，或者按下 Enter 键以执行输入的命令。

## 3. 变量名补充

如同命令和文件名补充一样，Bash 也支持变量名的补充功能。当输入一个变量名的起始部分，接着按下 Tab 键之后，Bash 将会提供变量名的剩余部分，如下所示。

```
$ echo $HO-(Tab)-(Tab)
$HOME      $HOSTNAME    $HOSTTYPE
$ echo $HOM-(Tab)-E
```

## 4. 配置 Readline 库

使用 readline 库函数的 Bash 或其他程序需要读取 INPUTRC 环境变量指定的文件，以获取初始化信息。如果未设置 INPUTRC 环境变量，这些程序将会读取 \$HOME/.inputrc 文件。使用下列语法格式，可以设置 readline 库函数的控制变量，从而控制 readline 库函数的处理动作。

```
set variable value
```

表 3-8 列出了 readline 库函数的部分控制变量（完整的变量及其说明可以使用“man readline”或“info readline”命令获取），括号中为控制变量的默认值。

表 3-8

readline 库函数的部分控制变量

控制变量	简单说明
editing-mode(emacs)	该变量用于确定使用哪一个编辑器。如果把变量设置为 vi，则意味着使用 vim 模式启动 readline 库函数。设置为 emacs，意味着使用 emacs 模式启动 readline 库函数。设置该变量的效果相当于执行“set -o vi”或“set -o emacs”命令
horizontal-scroll-mode (Off)	如果把该变量设置为 On，则意味着超长的命令行将会延伸到屏幕显示区的右边界之外。当把光标移动到右边界时，将会引起长的命令行依次向左移动，从而能够看到命令行的超长部分。该变量的默认值为 Off，在此情况下，超长的命令行部分将会依次延续到第二行
completion-ignore-case (Off)	该变量用于确定在匹配和补充命令与文件名时是否区分大小写字母，将其设置为 On 意味着不区分大小写字母
disable-completion (Off)	该变量用于确定是否允许执行命令或文件名补充，将其设置为 On 意味着禁止执行命令或文件名的补充
expand-tilde (Off)	在执行文件名补充时，如果该变量被设置为 On，Shell 能够扩展波浪号“~”
match-hidden-files (On)	在执行文件名补充时，如果该变量被设置为 On，Shell 能够匹配以句点“.”为起始字符的隐藏文件
mark-directories (On)	把该变量设置为 Off，意味着在执行文件路径名补充时不会在目录名之后附加斜线字符“/”

## 3.9 命令别名

为了照顾用户的使用习惯和对不同操作系统命令的偏好，简化输入的命令，提供默认的命令选项，Bash 以及其他大多数 Shell（如 Korn Shell、TC Shell、C Shell 和 Z Shell 等）等均提供了一种别名机制，使用户能够定义自己喜欢使用的命令名字，以便在输入别名时，Shell 能够替换并执行实际的命令（包括选项）。

除了分号、“&”符号、括号、管道符号、书名符号、换行符、空白字符、元字符、引号、等号以及变量和命令替换字符之外，命令别名可以由任何任意数量的字符或字符串组成。

对于用户而言，别名机制主要是通过 alias 和 unalias 命令实现的。其中，alias 命令用于定义和列出用户设置的（包括系统定义的）命令别名，其语法格式简写如下。

```
alias [name[=value]]
```

在上述语法格式中，如果别名定义的 value 中包含空格等空白字符，value 前后应加单引号或双引号。例如，要使用一个简单的命令别名替代 chmod 命令，以便增加执行 Shell 脚本的访问权限，可以使用下列 alias 命令定义一个命令别名。

```
$ alias cx='chmod 755'
```

要用一个简单的命令别名替代一个复杂的命令，确保使用的 ls 等命令总是带有一组基本选项，可以使用下列 alias 命令定义一个命令别名。

```
$ alias ll="ls -l"
```

在 Linux 系统中，如果没有特意使用 set 命令设置 noclobber 特性，当使用 cp、rm 或 mv 命令复制、删除或重新命名文件时，如果目标文件与现有的文件同名，有可能会在无意之间覆盖或删除同名的文件。为此，可以定义下列命令别名，以便在遇到此类情况时能够提示用户做出选择。

```
$ alias cp='cp -i'
$ alias mv='mv -i'
$ alias rm='rm -i'
```

在定义命令别名时，等号右边可以包含多个命令，中间由分号隔开；也可以使用管道符号，



把多个命令连接起来，构成一个组合命令。例如，为了计数当前目录中的文件，可以定义下列别名（注意，当文件名中包含空格字符时，这种文件计数的结果并不准确）。

```
$ alias cf='ls | wc -w'
```

```
$
```

当执行上述别名命令时，即可给出当前目录中的文件计数，如下所示。

```
$ cf
```

```
8
```

```
$
```

在输入 alias 命令时如果未指定参数，alias 命令将会列出系统定义以及用户设置的所有命令别名。如果仅仅指定了命令别名，Shell 将会列出给定命令别名的定义，如下所示（其中的 ls 是系统定义的命令别名）。

```
$ alias
```

```
alias cf='ls | wc -w'
```

```
alias cp='cp -i'
```

```
alias cx='chmod 755'
```

```
alias ll='ls -l'
```

```
alias ls='ls --color=auto'
```

```
alias mv='mv -i'
```

```
alias rm='rm -i'
```

```
$ alias ls
```

```
alias ls='ls --color=auto'
```

```
$
```

在定义命令别名时，如果语法格式中的 value 中包含变量，则单双引号的选用是非常重要的。在定义命令别名时，如果使用的是双引号，value 中的任何变量将会在定义过程中进行变量替换；如果使用单引号，value 中的变量在调用命令别名之前不会进行变量替换。下面的例子解释了这一差异。

PWD 变量包含当前工作目录的路径名。假定用户 gqxing 在位于其主目录 (/home/gqxing) 时利用双引号定义了命令别名 dir1，如下所示。

```
$ echo $PWD
```

```
/home/gqxing
```

```
$ alias dir1="echo Current working directory is $PWD"
```

```
$ alias dir1
```

```
alias dir1='echo Current working directory is /home/gqxing'
```

```
$
```

在建立上述命令别名时，等号 “=” 右边的\$PWD 变量将会立即执行变量替换。因此，当使用“alias dir1”命令显示 dir1 命令别名的定义时，其输出结果表示变量替换已经发生。不管在何处执行 dir1 命令，其显示的当前工作目录都是替换后的/home/gqxing，而非实际的工作目录，如下所示。

```
$ cd /etc
```

```
$ dir1
```

```
Current working directory is /home/gqxing
```

```
$
```

当使用单引号定义 dir2 命令别名时，将会防止 Shell 提前解释\$PWD 变量。下列“alias dir2”命令的输出表示 dir2 命令别名仍然保持未替换的\$PWD 变量。

```
$ echo $PWD
```

```
/home/gqxing
```

```
$ alias dir2='echo Current working directory is $PWD'
```

```
$ alias dir2
```

```
alias dir2='echo Current working directory is $PWD'
```

```
$
```

当使用 cd 命令改换到其他目录时，dir2 命令显示的是正确的工作目录，如下所示。

```
$ cd /etc
```

```
$ dir2
```

```
Current working directory is /etc
```

```
$
```

`unalias` 命令用于删除已经定义的命令别名，其语法格式如下。

```
$ unalias [-a] [name ...]
```

其中，“-a”选项用于删除已经定义的所有命令别名，包括系统定义的命令别名。要删除或撤销某个特定的命令别名，可以在 `unalias` 命令中直接指定。例如，要删除已定义的命令别名 `ls`，可以使用下列命令。

```
$ unalias ls
$ alias ls
ls:
$
```

使用下列命令将会删除系统定义或用户设置的所有命令别名。

```
$ unalias -a
$ alias
$
```

注意，不能递归地定义命令别名。但是，在别名定义等号右边的别名值（value）中，如果最后一个字符是空格（或制表符），则紧跟命令别名的下一个命令也将做别名检查和替换。此外，使用 Shell 函数也可以实现别名机制的所有功能。

## 3.10 作业控制

现在，几乎所有的 Shell 均支持作业控制功能。在 Bash 中，`set` 命令的“-m”或“-o monitor”选项用于启用 Shell 的作业控制功能。通常，作业控制功能总是被启用的。如果作业控制功能不正常，可以检查这个设置是否正确。

除了进程 ID 之外，Shell 还会为每个作业分配一个数字较小的作业号。例如，当利用&符号启动后台作业，并使之异步地运行时，Shell 将会输出下列信息，其中第一个数字表示作业号，第二个数字是作业的进程 ID。

```
$ find / -name "*conf" -print > conf.log 2>&1 &
[2] 3690
$
```

Shell 采用作业控制表记录和跟踪当前的作业。利用 `jobs` 内部命令，可以显示作业控制表中保存的当前作业。因此，为了查询系统中的后台作业信息，可以使用 `jobs` 命令列出系统中的所有作业，其中包括作业号、作业的命令行以及正在运行或暂时停止运行等状态信息。下面的例子说明，当前系统中存在两个后台作业，其中一个作业正在运行，另外一个作业已处于停止运行状态。

```
$ jobs
[1]+  Stopped      overload.sh
[2]-  Running      find / -name "*conf" -print > conf.log 2>&1 &
$
```

当运行一个较长时间才能完成的程序时，为了能够在此期间继续执行其他任务，可以使用 Ctrl-Z 键以及 `bg` 命令，把程序放入后台运行。按下 Ctrl-Z 键时会向 Shell 和当前程序发送一个 STOP 信号。收到此信号之后，Shell 将会输出一个表示作业已经停止（“Stopped”）的信息，接着输出另一个命令提示符，示例如下。

```
$ overload.sh
^Z
[1]+  Stopped      overload.sh
$
```

此时，用户可以改变作业的运行状态，或者使用 `bg` 命令把作业放到后台运行，或者在完成其



他任务后，利用 fg 命令仍让作业回到前台运行。

例如，要把停止运行的作业放到后台运行，可以使用下列命令。

```
$ jobs  
[1]+ Stopped      overload.sh  
[2]- Running      find / -name "*conf" -print > conf.log 2>&1 &  
$ bg %1  
[1]+ overload.sh &  
[2]- Exit 1        find / -name "*conf" -print > conf.log 2>&1  
$ jobs  
[1]+ Running      overload.sh &  
$
```

要让一个后台作业回到前台继续运行，可以使用下列命令。

```
$ jobs  
[1]+ Stopped      overload.sh  
[2]- Running      find / -name "*conf" -print > conf.log 2>&1  
$ fg %2  
find / -name "*conf" -print > conf.log 2>&1  
$
```

要停止一个后台作业，可以使用下列命令。

```
$ jobs  
[1]+ Stopped      overload.sh  
[2]- Running      find / -name "*conf" -print > conf.log 2>&1  
$ kill %1  
[1]- Terminated  overload.sh  
[2]+ Exit 1        find / -name "*conf" -print > conf.log 2>&1  
$ jobs  
$
```

要等待一个当前正在运行的作业的完成，可以使用下列命令。

```
$ jobs  
[1]+ Stopped      overload.sh  
[2]- Running      find / -name "*conf" -print > conf.log 2>&1  
$ wait %2  
[2]- Exit 1        find / -name "*conf" -print > conf.log 2>&1  
$
```

通常，后台作业可以输出数据，但不允许从终端读取数据。如果后台作业需要从终端读取数据，作业将会停止运行。另外，如果使用“stty tostop”命令设置终端选项，将会禁止后台作业输出数据。此后，当遇到数据输出或需要从终端读取数据时，后台作业也会停止运行。

除了作业号之外，Shell 还提供了若干方法，以便用户选择后台作业，详列如下。

- **%number** 使用作业号引用后台作业；
- **%string** 使用给定的字符串作为命令行起始字符串引用作业；
- **%?string** 引用命令行含有给定字符串的作业；
- **%%** 引用当前作业；
- **%+** 等价于%%，表示当前作业；
- **%-** 引用前一个作业。

在 Ubuntu Linux 系统中，如果用户提交了后台作业，不管作业的当前运行状态如何，一旦使用 exit 命令、Ctrl-D 键或关闭终端窗口退出系统，系统都会不加警告地立即终止后台作业的运行。如果想在退出系统后仍然确保后台运行的作业能够继续运行，直至其正常结束，可以利用 nohup 命令实现。

在此情况下，Shell 会随时监控用户提交的后台作业及其运行状态。例如，当提交的后台作业处于停止运行状态而准备退出系统时，用户将会收到一条警告信息：“There are stopped jobs.”。此时，可以利用 jobs 命令查询后台作业，决定是否需要等待作业的完成。如果经过确认后仍然想要退出系统，而不关心作业是否继续运行，Shell 会立即终止已经处于停止运行状态的后台作业，如下例所示。

```
$ nohup overload.sh
nohup: ignoring input and appending output to 'nohup.out'
^Z
[1]+  Stopped                  nohup overload.sh
$ exit
logout
There are stopped jobs.
$ exit
logout
```

但是，如果提交的作业处于后台运行状态，当使用 exit 命令、Ctrl-D 键或关闭终端窗口而退出系统时，系统将会不加警告地立即退出系统，但后台作业仍会继续运行，如下例所示。

```
$ nohup overload.sh &
nohup: ignoring input and appending output to 'nohup.out'
[1] 7680
$ exit
logout
```

可以从另外一个终端窗口中，利用下列命令验证上述事实（overload.sh 脚本仍在继续运行）。

```
$ ps -ef | grep overload
gqxing    7680      1  0 14:48 ?        00:00:00 /bin/sh overload.sh
$
```

事实上，当使用 nohup 命令调用某个命令时，nohup 的功能是让调用的进程忽略 SIGHUP 信号，同时把进程的输出重定向到用户主目录下的 nohup.out 文件中。当需要提交一个运行时间很长，而用户又不想一直待在终端之前静等命令的最终处理结果时，可以借助于 nohup 命令。

在退出 Shell 时，系统将会向用户注册 Shell 的所有子进程发送一个 SIGHUP 信号。通常，这个信号将会引起用户的所有进程终止运行。如果使用 nohup 命令，所有正在运行的进程或后台作业，包括本身已经屏蔽了 SIGHUP 信号的程序，将会忽略 SIGHUP 信号而继续运行。

## 3.11 会话记录与命令确认

### 3.11.1 保存会话记录

Linux 系统提供了一个很有用的工具——script，通过 script 命令，可以记录一个用户从注册到退出系统的整个或部分会话过程，包括用户的输入和系统的响应信息。实际上，这个工具仅适用于字符终端设备，尽管也能够捕捉 vim 编辑器等会话过程，但由于在 vim 的编辑过程中很可能会使用控制字符，执行诸如光标移动等操作，因而使得捕捉的内容难于阅读。如果利用 cat 命令显示捕捉的 vim 会话过程，有时很可能会一闪而过来不及阅读，致使记录的内容意义不大。因此，最好使用编辑器（如 vim 等）等工具查阅会话过程的记录文件。

通常，script 命令捕捉的会话过程将会记录在当前目录下的 typescript 文件中。为了使用不同的文件存储会话过程，可以在 script 命令之后指定一个不同的文件名。为了依次记录用户的每一个注册会话过程，可以使用“-a”选项。否则，script 将会覆盖原有的文件内容，删除先前的会话记录。下面的例子说明了怎样使用 script 命令记录用户的会话过程。

```
$ script
Script started, file is typescript
$ date
2008 年 09 月 06 日 星期六 18:28:04 CST
$ uname -r
2.6.24-19-generic
$ ls -l
```



```
总用量 28
drwxrwx--- 2 gqxing gqxing 4096 2008-07-16 18:42 conf
drwxr-xr-x 2 gqxing gqxing 4096 2008-07-16 09:48 data
drwxr-xr-x 2 gqxing gqxing 4096 2008-07-16 09:57 files
drwxr-xr-x 2 gqxing gqxing 4096 2008-07-16 15:16 incl
-rw-r--r-- 1 gqxing gqxing 895 2008-07-16 17:51 makefile
drwxr-xr-x 2 gqxing gqxing 4096 2008-07-16 18:49 script
drwxr-xr-x 2 gqxing gqxing 4096 2008-07-16 15:37 src
-rw-r--r-- 1 gqxing gqxing 0 2008-09-06 19:03 typescript
$ uptime
 19:28:10 up 4:17, 3 users, load average: 0.01, 0.08, 0.08
$ exit
exit
Script done, file is typescript
$
```

要终止会话记录功能，可以使用 `exit` 命令退出 `script`（这里的 `exit` 命令并不是退出 Linux 系统）。之后，可以使用 `cat`、`less`、`more` 或编辑器查阅 `typescript` 文件。下面的例子就是利用 `cat` 命令查阅刚才创建的 `typescript` 文件的结果。

```
$ cat typescript
Script started on 2008年09月06日 星期六 19时03分25秒
$ date
2008年09月06日 星期六 18:28:04 CST
$ uname -r
2.6.24-19-generic
$ ls -l
总用量 28
drwxrwx--- 2 gqxing gqxing 4096 2008-07-16 18:42 conf
drwxr-xr-x 2 gqxing gqxing 4096 2008-07-16 09:48 data
drwxr-xr-x 2 gqxing gqxing 4096 2008-07-16 09:57 files
drwxr-xr-x 2 gqxing gqxing 4096 2008-07-16 15:16 incl
-rw-r--r-- 1 gqxing gqxing 895 2008-07-16 17:51 makefile
drwxr-xr-x 2 gqxing gqxing 4096 2008-07-16 18:49 script
drwxr-xr-x 2 gqxing gqxing 4096 2008-07-16 15:37 src
-rw-r--r-- 1 gqxing gqxing 0 2008-09-06 18:27 typescript
$ uptime
 19:28:10 up 4:17, 3 users, load average: 0.01, 0.08, 0.08
$ exit
exit

Script done on 2008年09月06日 星期六 19时07分46秒
$
```

注意，当使用 `vim`、`emacs` 或其他编辑器编辑 `typescript` 文件时，可以使用 `Ctrl-V` 键、`Ctrl-M` 键和“`s`”等编辑命令删除多余的回车字符（参见第 6 章），也可以事先利用 `dos2unix` 或 `tr` 命令消除文件中每一行后面附加的回车字符“`^M`”。

```
$ dos2unix -n typescript newfile
$
```

或

```
$ cat typescript | tr -d '\r' > newfile
$
```

在上述 `tr` 命令行中，“`-d`”选项意味着删除随后指定的转义字符“`\r`”，即可见的回车字符“`^M`”。此外，如果想用 `dos2unix` 命令，需要事先安装 `tofrodos` 软件包。

### 3.11.2 确保使用的命令是正确的

#### 1. `which` 命令

当输入一个 Linux 命令时，Shell 将会按照 `PATH` 变量列出的目录顺序，也即所谓的检索路径，依次检索与之匹配的命令。如果用户改变了 `PATH` 变量定义的检索路径的顺序，Shell 究竟执行的

是哪一个目录下的同名命令，则不得而知。

为此，可以使用 Linux 系统中的 which 命令予以验证。例如，当用户对命令的运行结果超出预期的结果而感到怀疑时，为了弄清当前究竟执行的是哪一个 tar 命令，可以使用下列命令。

```
$ which tar
/bin/tar
$
```

通过运行 which 命令，可以找出当前究竟执行的是哪一个目录下的同名命令。有关系统目录的说明，请参见第 4 章。注意，which 命令并不考虑内置命令，它仅检索并给出发现的第一个外部命令。

## 2. whereis 命令

另外一个类似的命令 whereis 则用于检索与给定命令相关的文件（根据标准的目录位置，而不是检索路径）。例如，下列命令可用于寻找与 tar 命令有关的文件及其位置。

```
$ whereis tar
tar: /bin/tar /usr/include/tar.h /usr/share/man/man1/tar.1.gz /usr/share/man/man5/tar.5.gz
$
```

在上述例子中，whereis 发现了 3 个与 tar 有关的文件：tar 命令本身、tar 头文件和 tar 的联机文档。

## 3. which 与 whereis 命令的比较

当给定一个命令或程序的名字时，可以使用 which 命令，按照命令检索路径变量 PATH 设定的目录及其顺序，检索究竟执行的是哪一个命令或程序。在设定的命令检索路径中，如果给定名字的命令或程序不止一个，which 命令将会显示第一个发现的命令或程序，也就是最有可能调用的命令或程序。

对于给定的命令或程序，whereis 将会采取一种独立于检索路径的方式，按照 Linux 系统中的一系列标准目录进行检索，找出相应的二进制程序文件、联机帮助手册和程序的源代码，以及所有与之相关的文件。

## 4. apropos 命令

当需要执行某个特定的处理任务，但又不知道确切的命令名字，甚至根本就不知道使用什么命令时，可以借助于 apropos 命令，利用关键字检索可用的命令。apropos 将会利用提供的关键字，检索所有手册页中的命令简述部分，找出匹配的命令。实际上，apropos 与“man -k”命令的功能完全一样。

下列例子表示，当利用 who 作为关键字运行 apropos 命令时，系统将会给出一系列与之相关的命令、每个命令所属的手册类型以及命令的简单描述，而这个描述即取自命令手册页前面的简要说明部分。

```
$ apropos who
at.allow (5)          - determine who can submit jobs via at or batch
at.deny (5)           - determine who can submit jobs via at or batch
from (1)              - print names of those who have send mail
w (1)                 - Show who is logged on and what they are doing.
w.procps (1)          - Show who is logged on and what they are doing.
who (1)               - show who is logged on
whoami (1)            - print effective userid
whois (1)             - client for the whois directory service
$
```

## 5. whatis 命令

与 apropos 命令可以执行模糊检索相比，whatis 命令只能检索与给定关键字完全匹配的命令，如下例所示。

```
$ whatis who
who (1)                - show who is logged on
$
```

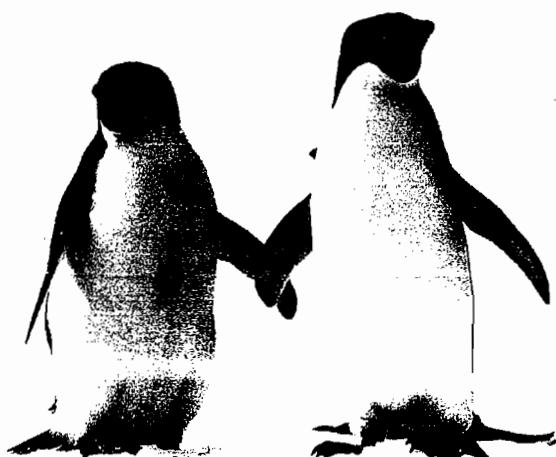


## 第4章

### 文件系统基础知识

文件系统是 Linux 操作系统的重要组成部分之一，承担信息处理的组织、管理和维护等任务。而文件则是 Linux 系统中信息的存储、读写和执行的基本单位。操作系统正是通过文件系统管理信息的存储、传输和加工等多种处理功能的。本章主要介绍文件系统的基础知识，讨论文件系统的层次结构和组织结构，介绍 Linux 系统支持的各种文件类型以及文件的安全保护机制。其内容主要包括：

- 文件系统的层次结构；
- 文件系统的组织结构；
- 文件的类型；
- 文件的安全保护机制。



## 4.1 文件系统的层次结构

在 Linux 系统中，信息的基本组织单位称作文件。Linux 文件系统采用一种逻辑的方法组织、存储、访问、操作和管理信息，把文件组织在一个层次目录结构的文件系统中，每个目录包含一组相关文件的组合。Linux 系统的一个重要特性是提供一种通用的文件处理方式，简化物理设备的访问；按文件方式处理物理设备，允许用户以同样的命令处理普通文件和物理设备。例如，在打印机上打印文件与在终端屏幕上显示文件的处理方式是类似的。

### 4.1.1 树形层次结构

从理论方面讲，文件系统是文件的一种逻辑组织结构。从用户的角度看，Linux 的文件系统只是一个树形层次组织结构的目录文件树，文件系统的起点是根目录 root。根目录相当于整个目录文件树的根，如图 4-1 所示。

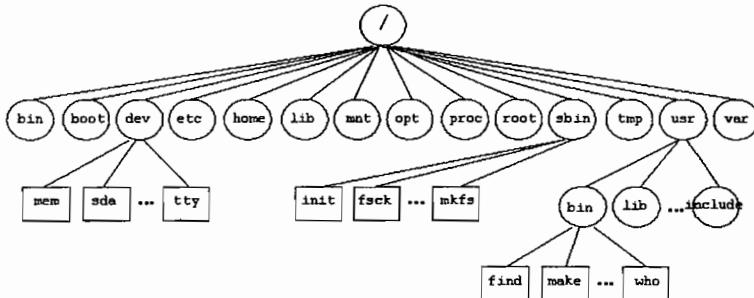


图 4-1 目录文件的层次组织结构

子目录是整个目录文件树形层次组织结构中的一个中间节点，是比当前目录层次低一级的目录。文件是整个目录树形层次组织结构中的一个叶子节点。例如，如果/usr 目录是当前目录，那么所有位于/usr 下面的目录及其子目录都是当前目录的子树，如 bin 和 lib 就是/usr 下边的子树。除非明确指定了目录路径，大多数 Linux 系统命令均把文件参数看作当前目录中的文件。

在文件系统中，若干文件可以组成一个目录，而若干不同的目录则可以构成一个目录的层次组织结构，而位于目录层次结构顶端的就是一个称为根目录的特殊目录。根目录包含了各种系统目录和文件，如/bin、/boot、/dev、/etc、/home、/lib、/proc、/sbin、/tmp、/usr 及/var 等标准目录。图 4-1 给出的是一个简化的文件系统层次组织结构。

在操作系统中，文件系统的设计目的就是把文件有序地组织在一起。Linux 系统提供了一种便于用户从逻辑上组织文件的文件系统。

Linux 系统鼓励用户按照一定的原则建立目录，例如，存储源程序的目录、存储目标程序的目录以及存储文档的目录等。Linux 文件系统的关键思想是其层次组织结构，不管系统中拥有多少用户，每个用户都可以创建若干目录及其子目录，分类存储自己的文件。

另外，文件的访问权限是多用户计算机文件系统的一个重要组成部分，用于保护用户的数据安全。Linux 系统的一个重要特性是，所有的 I/O 设备都与特殊文件联系在一起，用户无需了解硬件



设备的读写方式，只需像操作普通文件一样操作特殊文件，即可达到访问 I/O 设备的目的。例如，读取特殊文件相当于从硬件设备中直接读出数据，写特殊文件则相当于直接向硬件设备发送数据。利用特殊文件，实现了用户程序与硬件设备之间的通信，由文件系统管理硬件设备的 I/O 处理，使得硬件设备对用户是透明的。

### 4.1.2 路径名

无论何时，当前工作目录中的所有文件都是可以直接存储的。通过名字，可以直接引用文件。而对于非当前目录中的文件，必须在文件名之前加上各级目录路径才能访问。文件的路径名指的就是从某个目录开始，穿过整个文件系统，直至到达目标文件而经过的一条目录层次路径。例如，从“/”目录开始，中间经过 usr 和 bin 两级子目录的一条路径就是 find 文件的路径名，如下所示。

/ → usr → bin → find

把上述访问路径写成 Linux 文件系统中标准路径名就是 /usr/bin/find。

文件的访问路径可以有两个起始点：一是从当前工作目录开始，二是从根目录开始。凡是以前目录为起始位置，即以斜线字符“/”为起始字符的路径名称为绝对路径名。绝对路径名指定了文件在文件系统的层次组织结构中从根目录开始的存储位置。

相对路径是指相对于当前工作目录而言的目录。凡以当前工作目录或其他以非斜线字符为起始字符的所有路径名都是相对路径名。相对路径名指定了文件在文件系统中相对于当前工作目录的存储位置。

例如，路径名 /usr/include/stdio.h 就是一个从根目录开始的绝对路径名。其中，usr 是根目录的子目录，include 是 usr 目录的子目录，而 stdio.h 则是这个目录层次末端的一个文件。

include/stdio.h 是相对于 /usr 目录的一个相对路径名，而 stdio.h 则是相对于 /usr/include 目录的一个相对路径名。

每个目录中均包含以句点“.”和双句点“..”命名的两个特殊的目录文件，分别表示当前目录及其父目录。这两个特殊目录把文件系统中的各级目录有机地联结在一起。句点“.”是当前目录的别名，凡是期望访问当前目录中的文件时，都可以直接使用句点“.”而不必明确给出当前目录名。双句点“..”是当前目录父目录的别名。从任何目录位置开始，使用双句点“..”形式的父目录，可以逐层攀升到文件系统层次组织结构的最上层。

下面几个简单的规则适用于所有的路径名。

- 如果路径名以斜线字符开始，则说明路径名是从根目录开始的绝对路径名，除此之外，其他所有的路径名都是相对于当前目录的相对路径名。
- 路径名要么是由斜线字符分隔的一系列名字，要么是单个名字，在一串名字中，最后一个名字就是实际的文件，其他名字均为目录。文件可以是任何类型的文件。
- 在任何目录位置，在路径名中使用双句点“..”符号可以往上攀升文件系统的目录层次。在路径名中，除了双句点“..”之外的其他所有名字均为降低目录层次。

## 4.2 文件系统的组织结构

在 Linux 文件系统中，目录和文件的组织结构是有一定的规律可循的。这种有序的组织

结构有助于用户访问、管理和维护 Linux 系统，图 4-2 给出的是一个典型的文件系统目录组织结构。

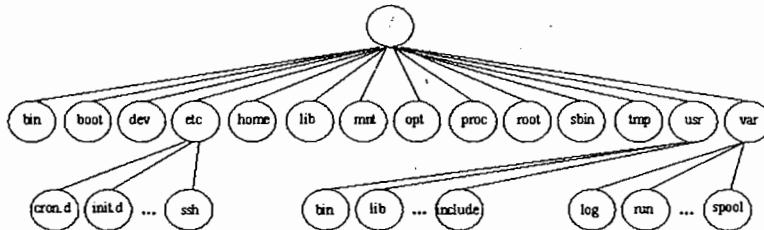


图 4-2 Linux 文件系统的典型目录结构

表 4-1 是对 Linux 系统部分主要目录的简要说明。

表 4-1

Linux 系统的部分主要目录

常见目录	典型子目录	说 明
/bin		其中包含系统、系统管理员和普通用户可以共享的各种通用程序，如 cat、cp、mv、rm、ls 及 ps 等常用的基本命令，以及如 bash 等的各种 Shell
/boot		其中包含系统引导程序、Linux 内核程序文件 vmlinuz、磁盘内存映像文件 initrd 以及 GRUB 初始引导程序和配置文件等
	grub	其中存有 GRUB 配置文件，以及 3 种不同类型的初始引导程序等
/dev		在 Linux 系统中，任何设备均对应一个或多个特殊文件（或称为设备文件），这个目录包含了系统支持的所有设备文件。例如，console 表示系统控制台，lp0 表示打印机，ttyXX 表示系统的串口设备，cdrom 表示 CD/DVD 驱动器，dsp 文件表示系统的音响设备，写到这个文件的任何数据，将会被重定向到音响设备。mem 表示系统的物理内存，kmem 表示系统内核占用的虚拟内存，sda 表示连接到主控制器的第一个磁盘，sda1 和 sda2 等则分别表示其中的第一个和第二个磁盘分区等
/etc		该目录是整个 Linux 系统的中心，其中包含所有系统管理和维护方面的配置文件，如 host.conf、inetd.conf、logrotate.conf、mke2fs.conf、nsswitch.conf、pam.conf、resolv.conf、sysctl.conf、syslog.conf 和 vsftpd.conf 等。此外，还有其他大量的配置文件分别位于单独的子目录中。通常应注意备份这个目录中的重要配置文件，以便需要时能够快速地恢复系统
	apache2	Apache 配置文件的根目录，其中包含 Apache 服务器的各种配置文件，如 apache2.conf 等。Apache 是一个通用的、高性能的 HTTP 服务器，也是世界上最流行的 Web 服务器。Apache 采用模块化的设计方式，支持运行时的动态模块选择、虚拟主机以及服务进程数量的动态调整等
	apt	其中包含软件管理工具使用的配置文件，如 sources.list 等
	bind	其中包含 named 域名服务守护进程使用的各种配置文件，如 named.conf 等
	cron.d	用于存储 cron 进程调度运行后台进程所用的配置和控制文件。其他有关的目录包括 cron.hourly、cron.daily、cron.weekly 和 cron.monthly 4 个目录
	cups	用于存储通用 UNIX 打印系统（Common UNIX Printing System，CUPS）使用的各种配置文件
	default	其中的文件用于提供部分工具软件使用的变量及其默认值
	gdm	用于存储 GNOME 显示管理器的配置文件，如 gdm.conf，其中定义了是否允许使用 root 注册
	init.d	用于存储系统启动过程中需要由 init 调度执行的脚本文件
	modprobe.d	其中包含系统加载的软件模块的配置文件，如 aliases 等
	mysql	其中包含 MySQL 数据库的配置文件，如 my.cnf 等
	network	其中包含网络接口的配置文件 interfaces，以及相关的配置工具



续表

常见目录	典型子目录	说 明
	pam.d	其中包含各种用户认证方法的配置文件
	ppp	用于存储 PPP 的脚本文件和配置文件
	profile.d	用于存储/etc/profile 使用的辅助初始化脚本文件
	rcN.d	用于存储进入相应运行级时需由 init 进程调度执行的脚本文件（其中的 N 为 0、1、2、3、4、5、6 或 S，表示系统的运行级）
	samba	Samba 配置文件的根目录。Samba 是一个网络共享软件的总称，Linux 系统中实现的 SMB 协议，允许 Linux 系统为 Windows 系统提供文件和打印共享服务
	security	用户存储的基本安全控制文件，包括注册控制文件、控制台访问控制文件以及资源限制控制文件等
	skel	用于存储最基本用户的用户初始化文件，如.bash_logout、.bashrc 和.profile 等。每当增加一个新用户时，都会把其中的部分基本初始化文件复制到用户的主目录中。注意，上述文件均为隐藏文件
	ssh	OpenSSH 网络服务所用配置和控制文件的根目录，其中含有 sshd_config 等重要配置文件
	X11	其中包含 X 服务器使用的各种配置文件，如 xorg.conf 等
/home		用户主目录的根目录。每当增加一个新用户，系统将会在/home 目录中创建一个形如 /home/\$USER 的子目录，作为用户的主目录，其中的\$USER 为用户名
/lib		该目录含有系统引导过程，以及运行系统命令所需要的内核模块和各种动态链接共享库文件（扩展名为.so，相当于 Windows 系统中的.dll 文件）。其中，内核模块（驱动程序）位于/lib/modules/kernel-version 子目录中
/lost+found		每个文件系统分区都存在一个 lost+found 目录，用于存储 fsck 命令在检测与修复文件系统时删除的文件或目录。Linux 系统要求正常地关机。一旦由于电源等硬件故障或系统软件故障致使系统异常停机，将会导致文件系统的损坏；在重新启动系统时，系统需要使用 fsck 命令对文件系统进行检测与修复。在检测文件系统期间，fsck 将会尝试修复任何受损的文件。部分被恢复的文件或无法恢复的数据块将会被放置在每个文件系统的 lost+found 目录中。如果这个目录中存有受损的数据文件，可以使用 debugfs 等工具，尝试自行恢复丢失的数据。如果是系统文件，也许需要重新安装相应的软件包
/media		移动存储介质的安装点。当利用 GNOME 界面安装移动存储介质时，系统将会自动地把移动介质安装到此目录下的某个子目录中
/mnt		文件系统的临时安装点。这是一个通用的安装点，可以临时安装任何文件系统或远程资源。如果愿意，也可以在此目录中创建若干子目录，以便安装不同的设备，例如，使用/mnt/cdrom 和/mnt/usb 分别安装 CD/DVD 和 U 盘等
/opt		应用程序等附加软件的安装目录
/proc		进程文件系统 proc 的根目录，其中的部分文件分别对应当前正在运行的进程，可用于访问当前进程的地址空间。/proc 是一个非常特殊的虚拟文件系统，其中并不包含“实际”的文件，而是可用以引用当前运行系统的系统信息，如 CPU、内存、运行时间、软件配置及硬件配置等信息。通过读取或修改/proc/sys 目录中的适当文件，可以显示或改变运行系统的回调内核参数，相当于执行 sysctl 命令。最具特色的是，除了 kcore 等少数几个文件之外，/proc 目录中绝大部分文件的大小均为 0。以数字命名的目录对应于一个实际的进程，而这个数字则表示进程的 ID。这些大小为 0 的文件相当于一个指向内核数据空间的指针，可据以查询相应的进程信息。由于这个原因，可以把/proc 文件系统看作系统内核的一个控制和信息中心。事实上，许多系统命令只是直接显示这个文件系统中的部分文件内容，例如，运行 lsmod 命令与使用 cat 命令显示/proc/modules 文件的内容，最终的结果是一样的。再如，执行 lspci 命令与显示/proc/pci 文件内容的结果也是一样的
	net	其中的文件分别表示各种网络协议（如 TCP、UDP 及 ARP 等）的状态与统计信息
	sys	该目录不仅存有各种系统信息，也包含系统内核与 TCP/IP 网络等的回调参数。其中的 kernel 子目录含有共享内存和消息队列的回调参数，net 子目录中含有 TCP/IP 的各种回调参数。例如，shmmmax 文件中含有系统的最大共享内存定义，如果使用“echo somevalue > /proc/sys/kernel/shmmmax”命令，可以直接修改运行系统的内核参数，而无需重新引导系统。但这一做法需要慎重，有的文件可能包含多个数值或不同类型的数值，因此，在修改以前一定要弄清参数的意义和实际的数据。为了在每次启动系统时都能使用定制的系统回调参数，可以设置 sysctl.conf 配置文件，或者编写自己的 Shell 启动脚本（参见第 10 章）

续表

常见目录	典型子目录	说 明
/root		超级用户 root 的主目录（在 Linux 系统中，“/”是整个系统的根目录，而非超级用户的主目录）
/sbin		其中包含与系统引导、管理和维护，以及与硬件配置等方面有关的命令或脚本文件，如 fdisk、init 和 ifconfig 等。在安装了/usr 等文件系统之后才需要执行的系统命令通常被放置在/usr/sbin 目录中。注意，/sbin 目录中的命令主要供超级用户使用，普通用户通常无法使用
/srv		用于存储本地系统提供的服务进程所用的数据文件（现为空目录）
/sys		系统各种设备配置信息的根目录。例如，block 子目录中含有磁盘及磁盘分区的配置信息，bus 子目录中含有 pci 和 usb 等的配置信息和驱动程序
/tmp		临时文件目录，用于存储系统运行过程中生成的临时文件，也可以供用户存储自己的临时文件。在系统的运行过程中，许多程序均使用这个目录存储临时数据文件，其中的许多文件对于当前正在运行的进程而言是非常重要的，删除这样的文件可能会导致系统瘫痪。因此，一般不要自己删除这个目录中的文件，除非确实有把握。在大多数系统，尤其是在 UNIX 系统中，伴随着系统的启动过程，将会清除这个目录中的所有文件
/usr		/usr 既可以作为一个单独的文件系统，也可以作为根目录下的一个子目录，其中存有系统提供的各种共享数据（如用户命令、库函数、头文件和文档等）
	bin	其中包含用户经常使用的各种命令
	games	其中包含游戏和教育等方面的程序
	include	用于存储各种 C 语言头文件。这个目录及其子目录中的头文件是 C 开发人员需要经常引用的文件。其中，sys、linux 和 bits 等子目录中定义的数据结构，对于深入学习、理解和掌握 Linux 系统具有极大的参考价值
	lib	其中包含各种共享的库函数，可供程序员以静态或动态方式链接自己开发的应用程序
	sbin	其中包含系统引导完成之后系统管理员经常使用的各种系统管理和维护命令
	share	共享目录，其中含有 man（联机文档的根目录）、info（GNU info 文档的根目录）、doc（各种软件包特定的文档）、locale（语言环境）、vim（用户指南）及 zoneinfo（时区定义）等子目录
	src	用于存放 Linux 系统内核的源代码和文档等
/var		/var 既可作为一个单独的文件系统，也可作为根目录下的一个子目录，用于存储各种可变长的数据文件（如日志文件）、暂存文件或待处理的临时文件等
	cache	APT 和 samba 等程序使用的工作目录。用于缓存程序使用的各种数据文件，尤其是缓存 APT 软件工具下载的软件包和信息文件
	lib	用于存储软件包特定的动态链接共享库、配置文件、数据文件和状态信息等
	lock	用于存储各种服务进程或应用程序访问特定的设备或文件时设置的封锁文件
	log	系统守护进程日志文件的存储目录，其中包括 lastlog（每个用户最后一次注册的时间记录）、messages（由 syslogd 记录的所有内核和系统程序的日志消息）以及 wtmp（所有用户的系统注册/注销记录）等重要文件。位于/var/log 目录中的文件会不断地增长，因而要求定期地备份或清除。通常，Linux 系统均采用以日、周或月为时间周期，定时执行例行检查，以循环截取（如使用/usr/sbin/logrotate 一类的程序）的方式，删除过时的数据，保留一定时间范围的最新数据，使文件的大小保持一个适中的规模。在 Ubuntu Linux 系统中，每日将会定时执行一次 logrotate 程序，检查并处理系统日志文件（参见 /etc/cron.daily/logrotate 文件）
	mail	这个目录存有每个用户电子邮件的邮箱文件，其中的每个文件均以用户的用户名命名
	run	系统运行信息文件的根目录，其中的各种.pid 文件存有相应守护过程的 PID，另外一个最典型的文件是/var/run/utmp，其中含有当前系统中的用户注册信息
	spool	用于缓存各种等待处理的文件，如打印任务等。通常，每一类待处理的缓存文件均位于各自的子目录中，如/var/spool/cups 等
	tmp	用于存储各种临时文件
	www	Apache 服务器的用户文档根目录，用于存储和发布各种 HTML 文档



## 4.3 文件的类型

从理论上讲，文件是由一系列连续的字节流组成的，最后以一个 EOF 字符结束。但从物理实现来讲，文件实际上是由磁盘（或其他存储介质）上的一系列数据块组成的，而组成文件的数据块，并不一定是连续的。

Linux 系统可以同时支持许多不同类型的文件系统，以及不同类型的文件。本章所述的文件仅指 Linux 文件系统支持的基本文件。也就是说，所谓文件是指驻留在本地磁盘上的，用户经常与之打交道的各种基本文件。

在 Linux 操作系统中，文件是 Linux 操作系统中的基本数据组织单位，所有的输入输出操作都是通过文件实现的，系统处理的任何设备和数据均可以归结为对文件的操作。从理论上讲，能够读写普通文件的任何程序都可以读写任何 I/O 设备。

从用途方面划分，一个文件可以是如下几种类型。

- 文档——也即普通的文本文件，包括脚本、程序源代码、配置数据和日志等。
- 命令——大多数命令都是可执行的二进制数据文件，也就是说，命令是可以执行的程序文件。例如，date 命令就是一个可以执行的程序文件，其功能是输出和设置当前的系统日期和时间。
- 目录——简言之，目录是一个包含文件名列表的数据文件。
- 设备——包括终端、打印机、磁盘和磁带机等。

在 Linux 系统支持的各种文件系统中，如最流行的 Ext2/Ext3 文件系统，通常均支持普通文件、目录文件、特殊文件、链接文件、符号链接文件及管道文件这 6 种不同类型的常规文件（套接字文件不在本章讨论之列），下面将分小节进行讨论。

### 4.3.1 普通文件

在 Linux 系统中，普通文件简称为文件。所谓文件，实际上是一个命名的数据集合，是一组信息的基本存储单位。通常，每个文件都拥有一个名字，通过名字，可以对文件的数据内容进行处理。

在早期的 Minix 文件系统中，文件名最多不能超过 14 个字符。而在 Ext2/Ext3 等文件系统中，文件名可以长达 255 个字符。在不同的目录中，文件可以同名，但在同一个目录中，则不允许同名的文件存在。原则上，文件名可由任何字符组成，但若包含不可打印字符、空白字符及 Shell 元字符，在实际应用过程中将是非常麻烦的。

普通文件可以存储任何数据，存储的内容可以是 ASCII 文本、源代码、Shell 脚本、配置数据及各种文档等，也可以是二进制程序代码等。由此可见，普通文件是 Linux 文件系统中应用最多的一种文件类型。

此外，Linux 系统中的文件并没有记录、结构以及内容之分，所有文件都是由一维的字符流组成的，存储在磁盘等存储设备中，在文件系统中占有零个或多个数据块。而且，数据也并非一定驻留在连续的磁盘块中。但操作系统隐藏了这一内部存储形式，用户看到的是一系列连续的字节流。对于用户而言，不用考虑文件的底层存储结构。

当然，也可以根据数据内容把文件分类为 C 源代码、Shell 脚本、文本数据以及二进制可执行

程序等。但 Linux 系统却不这样认为，对于存储不同数据的普通文件，Linux 操作系统不加任何区别，也不要求一个普通文件必须采用什么记录结构。

但是，为了便于处理与维护，Linux 系统也定义了若干标准数据文件格式，例如，二进制可执行文件必须采用 a.out 文件格式，一些档案文件前部通常都会包含一个文件格式代码（magic code），用于标识文件的内容。例如，cpio 命令生成的档案文件，其标识代码为 070707 等（参见 /etc/magic 文件）。此外，应用程序也可以按一定的记录格式组织和存储自己的文件。数据库等应用程序也是这样组织和使用文件的。

由于 Linux 系统并不刻意区分文件的类型，不像 Windows 系统那样，以扩展名区分文件的类型。因此，单从文件名本身来看，Linux 系统中的大部分文件都无从知道其类型。Linux 文件系统也没有对文件的命名强加任何约定，但由于应用程序的需要及用户的使用习惯，通常以“.c”作为 C 源程序文件名的后缀，以“.sh”作为 Shell 脚本文件名的后缀。按照惯例，可执行程序的文件名通常不加任何后缀。

当需要确定文件的内容、判断文件的类型时，可以选用一定的工具，其中最典型的就是 ls 和 file 命令。例如，下列 file 命令可用于确定/etc/apt 目录中所有文件的内容与类型（实际上，标识代码也是 file 命令用于确定文件内容与类型的方法之一）。

```
$ file /etc/apt/*
/etc/apt/apt.conf.d: directory
/etc/apt/secreng.gpg: empty
/etc/apt/sources.list: ASCII English text
/etc/apt/sources.list.d: directory
/etc/apt/trustdb.gpg: writable, regular file, no read permission
/etc/apt/trusted.gpg: GPG key public ring
/etc/apt/trusted.gpg: GPG key public ring
$
```

从用户的角度来看，普通文件可以分为两种类型：即文本文件和二进制数据文件。文本文件只包含可打印字符，如 ASCII 字符、中文字符等，而二进制数据文件中的每个字节允许有 256 种数值。

在 Linux 文件系统中，通常提供下述文件操作：打开文件（open）、创建文件（create）、读文件（read）和写文件（write）等。Linux 系统提供大量的文件操作命令，用于创建、编辑和处理文本文件。例如，为了显示一个文本文件，可以使用下列 cat 命令（有关文件的各种操作与处理方法，详见第 5 章）。

```
$ cat /etc/issue
Ubuntu 8.10 \n \1

$
```

对于二进制数据文件来说，其中的内容无法直接显示，因为每个字节的 256 种取值大部分都是不可打印字符。若想显示（而不是执行）二进制数据文件的内容，可以使用专门的命令，如 od 命令，以 ASCII 字符、八进制数据或十六进制数据等形式显示二进制数据文件。例如，若想检查一个二进制数据文件的内容，尤其头部是否包含特定的标识代码，可以使用下列命令。

```
$ od -c cpiofile
0000000 0 7 0 7 0 7 0 0 4 0 0 0 7 1 5 5 0
0000020 2 1 0 4 0 7 5 5 0 0 0 1 7 5 0 0 0
0000040 1 7 5 0 0 0 0 0 0 2 0 0 0 0 0 0 0
0000060 1 1 1 1 6 4 2 7 6 5 5 0 0 0 0 0 0
0000100 7 0 0 0 0 0 0 0 0 0 0 s c r i p t
```



```
0000120 p t \0 0 7 0 7 0 0 4 0 0 7 1  
.....  
$
```

### 4.3.2 目录文件

目录文件简称目录。目录也是一种文件，是一种特殊类型的文件，其中存储的内容不是普通意义上的数据，而是一系列文件名及其信息节点号。除了存储的内容不同之外，目录文件与普通文件在文件系统中的存储方式是一样的。目录用于提供文件名、信息节点与文件数据之间的关联关系。因而可以说，目录的结构也决定了文件系统的组织结构。

严格地讲，目录文件是由一系列目录项组成的，而每个目录项又主要是由两个不同的字段组成的：一个字段为信息节点号，用于引用信息节点，另一个字段为文件的名字。“ls -ai”命令可用于列出目录中的文件名及其信息节点号，下面是该命令的输出结果（其中第一列位信息节点号，第二列为文件名）。

```
$ ls -ai | sort -n  
2 .  
2 ..  
11 lost+found  
12 cdrom  
13 initrd.img  
14 vmlinuz  
.....  
$
```

在目录文件中，每个目录项通过信息节点号实现了文件名与文件数据的映射。通过文件名可以找到其信息节点，通过信息节点最终可以找到文件中的实际数据内容。在上述的例子中，文件名/lost + found 的相应信息节点号是 11，是继“/”目录和保留的信息节点位置（用于日志文件）之后创建的第一个目录。

如前所述，Linux 文件系统中的每个目录均包含两个特殊的目录，即以句点“.”和双句点“..”表示的当前目录及其父目录。因此，要进入当前目录或父目录，用户可以相应地引用“.”或“..”。例如，如果当前的工作目录为/usr/include/sys，输入“cd ..”命令后即可进入/usr/include 目录，如下所示。

```
$ pwd  
/usr/include/sys  
$ cd ..  
$ pwd  
/usr/include
```

“usr”是一个目录文件，其中包含的数据与普通文件并无实质的差别，但其与普通文件不同的是，目录文件是由 Linux 文件系统直接管理的，用户只能查询其中的文件列表，不能使用编辑器等工具直接修改目录文件。也就是说，用户只能读取目录文件，只有操作系统才能写目录文件。

超级用户可以利用 Linux 系统提供的若干实用程序维护目录文件，而普通用户则绝对不能直接写目录文件。例如，超级用户可以利用文件系统调试器 debugfs，直接操作文件系统的任何组成部分。应当了解的是，用于维护目录（或文件系统）的大多数实用程序都是文件系统特定的，例如，debugfs 主要适用于 Ext2/Ext3 类型的文件系统。

为了保证目录的完整性，操作系统不允许用户直接修改目录文件。用户可以在目录中创建文件，由操作系统把文件加到目录中。当用户删除文件时，由操作系统从目录文件中删除相应的文件名，目录文件始终是由操作系统负责维护的。

这是因为目录文件毕竟不是普通文件，如果不了解目录的组织结构，不了解文件系统的工作

原理，如果处理不当，将会造成无法预料的后果。即使熟悉文件系统，也不能直接操作目录文件，因为目录操作涉及一系列内部的处理步骤和过程，不是一般用户所能胜任的。

在任何目录中，可以存储普通文件、管道文件、特殊文件、符号链接文件，也可以创建目录文件——称作子目录。要创建一个目录，可以使用下列命令。

```
$ mkdir dirname
```

要进入不同的目录，可以使用下列命令。

```
$ cd dirname
```

在 Linux 系统中，每个用户都拥有一个属于自己的目录，称作主目录。一旦注册到 Linux 系统，系统将会自动地把用户引导至自己的主目录。用户可以在主目录中创建自己的文件，创建自己的子目录。在与 Linux 交互会话期间，用户也可以利用 cd 命令自由地从一个目录转移到其他目录中（如果具有足够的访问权限）。例如，如果想进入/usr/bin 目录，可以输入下列命令。

```
$ cd /usr/bin
```

```
$
```

在系统访问期间，用户所在的目录称为当前目录、工作目录或当前工作目录。在 Linux 系统中，通过使用不加任何选项的 cd 命令，即可返回到自己的主目录。例如，使用下列命令，任何用户都能够进入自己的主目录。

```
$ cd
```

```
$
```

要了解自己当前所处的目录，可以输入 pwd 命令。下列命令的输出结果表示用户的当前工作目录为/usr/include。

```
$ pwd
/usr/include
$
```

尽管目录文件通常是由系统维护的，用户不能直接写目录文件，但任何用户进程均可利用下列与目录操作有关的系统调用访问目录，自由地读取目录文件，只要具有相应的访问权限。

- mkdir() 创建一个新目录；
- rmdir() 删除一个空目录；
- getdents() 获取目录文件的内容。

### 4.3.3 特殊文件

特殊文件是 Linux 系统中最具特色的特性之一。特殊文件也称设备文件，为了便于用户访问外部设备而不必涉及各种 I/O 设备的具体操作细节，Linux 系统利用特殊文件作为用户与 I/O 设备之间的接口，使用户能够像读写普通文件一样访问外部设备。每个特殊文件均对应一个 I/O 设备，由 I/O 设备驱动程序实现用户与设备之间的数据通信。因此，用户可以像处理普通文件一样，通过打开、读写特殊文件等操作，实现访问外部设备的 I/O 处理要求。

特殊文件不包含任何数据。相反，特殊文件只是提供一种机制，在文件系统中建立一个物理设备与文件名之间的映射。系统支持的每个设备，包括内存，都与至少一个特殊文件相关联。特殊文件是利用下列 mknod 命令或系统调用创建的，而且必须提供相应的驱动程序，并集成到系统内核中。否则，即使创建了特殊文件，也无法访问相应的设备。

```
mknod special type [major minor]
```

上述命令中，special 为特殊文件名，如/dev/console（即系统控制台）。major 为主设备号，表示按设备类型组织的设备驱动程序指针数组的索引。minor 为次设备号，表示同类设备中的某个



子设备，可以用作调用相应驱动程序的参数。type 为特殊文件的类型，合法的特殊文件类型如下：

- c 字符特殊文件；
- b 块特殊文件；
- p 管道文件。

例如，要创建一个管道文件，可以使用下列命令。

```
$ mknod mypipe p  
$ ls -l mypipe  
prw-r--r-- 1 gqxing gqxing 0 2008-12-14 17:52 mypipe  
$
```

当提交一个读写特殊文件的请求时，系统将会直接调用相应的设备驱动程序。而驱动程序则负责在控制进程与相关的物理设备之间传输数据。例如，假定存在下列两个命令（其中的/dev/pts/0 是用户当前所用终端的设备名）。

```
$ cp /etc/passwd /tmp/garbage  
$ cp /etc/passwd /dev/pts/0  
root:x:0:0:root:/root:/bin/bash  
daemon:x:1:1:daemon:/usr/sbin:/bin/sh  
bin:x:2:2:bin:/bin:/bin/sh  
sys:x:3:3:sys:/dev:/bin/sh  
...  
$
```

第一个命令只是把/etc/passwd 文件的内容复制到/tmp/garbage 文件中，创建或复制一个相应的文件。第二个命令则意味着把/etc/passwd 文件复制到/dev/pts/0 文件中，而后者实际上是一个与用户终端相关联的特殊文件。因此，/etc/passwd 文件的内容将会显示在用户终端的屏幕上。

同 UNIX 系统一样，Linux 系统也支持两种特殊文件，即字符特殊文件和块特殊文件。除了一个目录项和一个信息节点之外，设备特殊文件并不占用文件系统的任何空间，这个特殊文件只是一个设备驱动程序的访问点。

系统中的每一个设备要么是块设备，要么是字符设备，二者必居其一。通常，块设备主要用于存储数据，字符设备主要用于传输数据。例如，磁盘和 CD/DVD 等均属于块设备，连接到串口和并口的设备等均属于字符设备。

设备除有块设备与字符设备等类型之分，还有主次设备号之分。主设备号主要用于区分不同的设备，次设备号主要用于区别同类设备的特定设备或分区。例如，假定磁盘设备的主设备号为 8，次设备号 1、2、… 分别表示磁盘设备的第一个、第二个、… 磁盘分区。

### 1. 块特殊文件

块特殊文件与采用数据块组织结构和处理方式的设备（如磁盘）相关联。所谓数据块组织结构的设备实际上指的是能够以固定长度的数据块传输数据，也能够随机访问其中任何数据块的存储设备，而且，访问每个数据块的时间差别也不大。

磁盘就是典型的数据块组织结构的设备。在 Linux 系统中，磁盘与系统内存之间通常以数据块的方式传输数据，以数据块为单位读写数据。借助文件系统，可以在磁盘的任何位置读写任意字节数量的数据。

在 Linux 系统中，一个典型的数据块是由 512、1 024 或 4 096 个字符组成的。因为文件系统通常都是驻留在块设备中的，故许多文件系统的维护命令均要求使用块特殊文件名作为文件参数。

例如，下列设备文件名就是一个对应于数据块组织结构的磁盘特殊文件（其中第一个字段的起始字符为 b）。

```
$ ls -l /dev/sda5
brw-rw---- 1 root disk 8, 5 2008-09-07 16:55 /dev/sda5
$
```

当通过块特殊文件接口传输数据时，系统通常会在其内存（或高速缓冲区）中缓存数据。在指定的时间间隔内，系统将会把内存中的数据写入外部存储设备，从而更新存储设备中的数据。在交互访问或应用程序中，用户可以利用 sync 命令或 fsync() 系统调用，强制系统把内存中的数据写入存储设备，实现存储设备与内存之间的数据同步。

采用这种设计方式的一个问题是，如果在内存与存储设备进行数据同步之前，系统突然瘫痪，数据的完整性将会遭到破坏。因此，如果 Linux 系统出现故障，内存中尚未写到存储设备上的最新数据很可能丢失。

更重要的是，内存中存有文件系统超级块中的最新数据，如果之前没有及时同步，文件系统（内存超级块与磁盘超级块之间）将会处于数据不一致的状态。如果再严重一点，文件系统有可能无法修复。

## 2. 字符特殊文件

任何非数据块组织结构的设备均为字符设备，而字符特殊文件就是与非数据块组织结构的任何设备相关联的特殊文件。与数据块组织结构的设备相反，字符设备无法使用定长的数据块，也不能随机访问，其最底层的 I/O 接口一次只能处理一个字符，这与块设备 I/O 接口一次能够处理一个完整的数据块的情况全然不同。

控制台终端、打印机、流式磁带机和其他串行设备均属于字符设备，因而具有相应的字符特殊文件名。当然，为了其他目的，数据块组织结构的磁盘设备也具有相应的字符特殊文件名。

通过字符特殊文件接口传输的数据将会在设备驱动程序和控制进程之间直接传递，中间并不经过系统的数据缓冲区。因此，这种形式的设备接口经常被称作原始接口，例如，下列设备文件名就是一个字符特殊文件（其中第一个字段的起始字符为 c）。

```
$ ls -l /dev/console
crw----- 1 root dialout 5, 1 2008-09-07 09:18 /dev/console
$
```

## 3. 定义特殊文件

根据前述说明，每个设备都对应于一个特殊文件，通过读写特殊文件，可以实现对设备的 I/O 处理要求。实际上，这些特殊文件只是一个接口，设备的 I/O 处理都是由设备的驱动程序完成的。因此，每当在系统配置中增加一个新的设备时，都需要完成下列任务：

- (1) 获取或编写相应设备的驱动程序；
- (2) 更新系统配置表，描述新加的设备以及相应的设备驱动程序；
- (3) 重新生成新的 Linux 内核，使其包含新的设备驱动程序；
- (4) 利用 mknod 命令，创建新增设备的特殊文件。

## 4. 特殊文件实例——4 个字符特殊文件

Linux 系统利用同一驱动程序模块与 4 个不同的次设备号，支持 4 个不同的字符特殊文件，作为用户与系统内存之间的设备接口。在安装 Linux 操作系统时，系统将会自动地创建 4 个不同的特殊文件，如下所示。

```
$ cd /dev
$ ls -l mem kmem null zero
crw-r---- 1 root kmem 1, 1 2008-09-07 12:53 mem
crw-r---- 1 root kmem 1, 2 2008-09-07 12:53 kmem
crw-rw-rw- 1 root root 1, 3 2008-09-07 12:53 null
```



```
crw-rw-rw- 1 root root 1, 5 2008-09-07 12:53 zero  
$
```

这 4 个特殊文件的作用简述如下。

- **/dev/null** 是一个数据回收站或漏斗文件，也是一个无底洞，只要写入这个文件，数据就会消失。这个文件经常用于过滤程序的输出。注意，当读取这个文件时，其功能等同于读取/dev/zero 文件，返回指定数量的数值 0。
- **/dev/zero** 可以提供任意数量的数值 0。例如，当程序从这个文件中读取 256 个字节时，将会收到 256 个 0。如果写入这个文件，数据将会消失，其作用等同于/dev/null。
- **/dev/mem** 提供计算机的物理内存接口，是 Linux 系统的内存映像。读取/dev/mem 文件的字节地址将被解释为物理内存地址。如果从头开始（偏移值为 0）读取这个文件，将会读取物理内存中从第一个字节开始的数据。
- **/dev/kmem** 提供系统内核的虚拟内存接口。如果从头开始（偏移值为 0）读取这个文件，将会读取系统内核从第一个字节开始的数据。如果没有这个接口，开发人员必须知道系统内核代码及其数据在物理内存中的起始位置。许多系统程序就是利用这个特殊文件访问系统内核，获取各种系统变量的内核地址，维护系统数据的。如果文件不存在，可以使用下列命令创建这个特殊文件。

```
# mknod -m 640 /dev/kmem c 1 2  
# chown root:kmem /dev/kmem
```

在上述 4 个文件中，**/dev/null** 最为常用。当不关心程序的标准输出或标准错误输出，而只是关注程序的出口状态时，经常会使用下列命令形式。

```
command > /dev/null  
command > /dev/null 2>&1
```

#### 4.3.4 链接文件

Linux 系统提供一种机制，使之能够采用不同的文件名引用同一数据或程序，也即把同一数据或程序赋予不同的文件名，这种文件称作链接文件，也称硬链接文件。当调用不同目录中的同名程序时，或以不同的名字调用同一个程序时，执行的是同一程序副本。

采用链接文件的好处是，只需在文件系统中保存一份数据或程序副本，多个文件名均指向同一数据内容，更新任何一个文件，即可反映到其他文件中。既能节省磁盘存储空间，又可保证数据的一致性以及文件存储位置的灵活性。

作为单独的文件，这 4 个中文字符定义文件存在于同一文件系统的不同目录位置，只不过与之对应的文件数据（存储在磁盘中的数据块）只有一个副本。通过以下两个命令，即可验证上述说法。

```
$ cd /usr/lib/locale  
$ ls -il zh_CN.utf8/LC_CTYPE  
157992 -rw-r--r-- 4 root root 254908 2008-04-23 06:21 zh_CN.utf8/LC_CTYPE  
$ find . -inum 157992 -exec ls -il {} \+  
157992 -rw-r--r-- 4 root root 254908 2008-04-23 06:21 zh_CN.utf8/LC_CTYPE  
157992 -rw-r--r-- 4 root root 254908 2008-04-23 06:21 zh_HK.utf8/LC_CTYPE  
157992 -rw-r--r-- 4 root root 254908 2008-04-23 06:21 zh_SG.utf8/LC_CTYPE  
157992 -rw-r--r-- 4 root root 254908 2008-04-23 06:21 zh_TW.utf8/LC_CTYPE  
$
```

由于上述 4 个文件具有相同的信息节点号，且其链接计数（第 3 列）为 4，说明这 4 个文件是采用硬链接的方式链接在一起的，因而拥有同一数据副本，但具有 4 个不同的文件名。注意，非链接文件（包括符号链接文件）的链接计数为 1。在 Linux 系统中，信息节点与数据是一一对应的。如果文

件的信息节点号相同，则其引用的就是同一个信息节点，因而拥有同一数据副本，参见图 4-3。

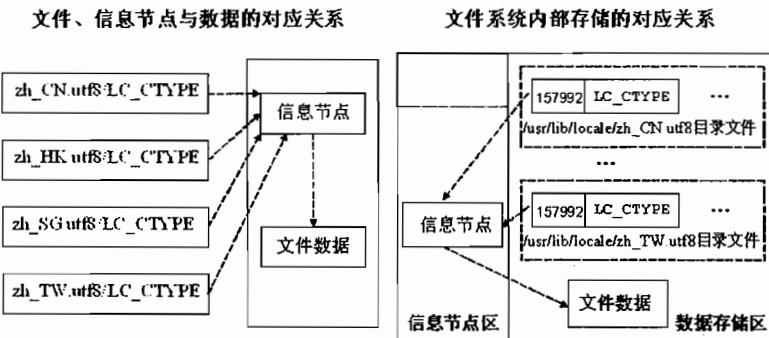


图 4-3 硬链接文件的数据引用关系

若想创建硬链接文件，可以使用 `ln` 命令或 `link()` 系统调用实现。创建硬链接文件时，文件引用的是同一信息节点号和文件数据，只是在同一目录或不同目录中增加一个新的文件名而已。注意，硬链接文件的唯一局限是链接的两个文件必须位于同一个物理文件系统中。

在 Linux 系统中，对于同一个数据内容，出于应用的需要，经常会赋予不同的文件名。例如，`/etc/init.d` 目录中的 Shell 脚本文件主要用于启动或关闭系统守护进程。同一个文件又可用于不同的运行级，因而需要放置到不同的目录中，如放置到 `/etc/rc0.d` 或 `/etc/rc2.d` 等目录中。

例如，在 Ubuntu 8.04 Linux 系统中，安装 Samba 服务器软件包时将会在 `/etc/init.d` 目录中增加一个 `samba` 脚本，但没有在 `rcN.d` 目录中增加必要的链接文件。为了把文件的副本分别存放到 `/etc/rc2.d` 和 `/etc/rc0.d` 两个目录中，以便达到自动启动与关闭 Samba 服务器的目的，可以使用 `ln` 命令，创建两个链接文件。

```
$ cd /etc/init.d
$ sudo ln samba ../rc2.d/S90samba
$ sudo ln samba ../rc0.d/K10samba
$ ls -l samba
-rwxr-xr-x 3 root root 2932 2008-09-10 20:10 samba
$ ls -l ../rc2.d/S90samba
-rwxr-xr-x 3 root root 2932 2008-09-10 20:10 ../rc2.d/S90samba
$ ls -l ../rc0.d/K10samba
-rwxr-xr-x 3 root root 2932 2008-09-10 20:10 ../rc0.d/K10samba
$
```

如此一来，就建立了 3 个文件，3 个文件共享同一个数据副本。也就说，3 个文件名均指向同一数据内容。

### 4.3.5 符号链接文件

类似于 UNIX 系统，Linux 系统也提供一种新的文件链接机制，以便用户能够跨越不同的物理文件系统建立链接文件，这种新的链接文件称作符号链接文件。为了创建符号链接文件，可以使用 “`ln -s`” 命令或 `symlink()` 系统调用实现。

与硬链接文件的实现方式不同，符号链接文件的本身也是一种数据文件，而且是一个单独的文件，只不过文件中的内容并非真正的数据，而是一个指向目的文件（或目录）的路径名而已。由此可见，利用符号链接机制，将会创建新的文件名和新的信息节点。而且，符号链接文件也会



擁有自己的数据块，其中包含的就是其引用的目的文件（或目录）的完整路径名，参见图 4-4。

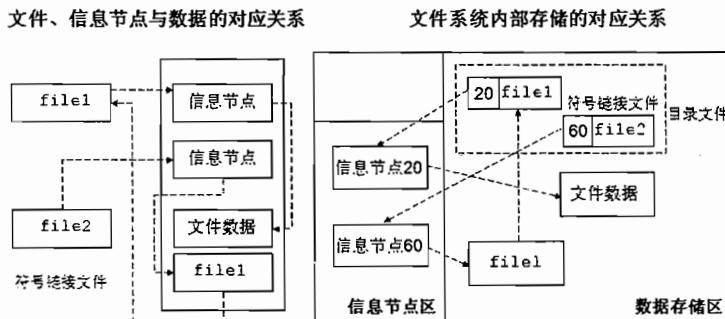


图 4-4 符号链接文件的数据引用关系

当引用一个符号链接文件时，系统首先需要打开符号链接文件本身，然后再根据文件的内容，打开其指向的文件。因此，当操作一个符号链接文件时，系统将会按照符号链接文件中的路径名进行二次检索，直至找到实际的文件。此外，符号链接文件与其链接的目的文件不必位于同一物理文件系统中，其引用的文件甚至可以不存在。符号链接文件可用于引用任何文件，尤其是目录，但硬链接不能链接目录，以免出现目录树引用的死循环。

采用符号链接文件的最大好处是能够确保目录文件结构的兼容性。当 Linux 系统的目录结构与文件组织需要调整时，为了保持文件系统的兼容性（这也是产生符号链接文件的主要原因之一），可以采用符号链接文件的形式，继续保持原有的目录结构或文件位置。例如，Ubuntu Linux 系统把/etc/motd 文件迁移至/var/run/motd，但仍然保持/etc/motd 文件的位置，只是把这个文件变成了符号链接文件，如下所示。

```
$ ls -ld /etc/motd
lrwxrwxrwx 1 root root 13 2008-12-05 13:18 /etc/motd -> /var/run/motd
$
```

利用符号链接文件，还可以照顾用户以往的上机习惯，把之前常用的命令名链接到新增的命令，实现命令名字的借用或间接引用。例如，halt、poweroff 与 reboot 命令引用的实际上是同一个 reboot 程序，vi 与 vim 命令引用的是同一 vim 程序。

在符号链接文件中，文件大小字段中的数值恰好等于符号链接文件引用的目的文件的路径名的长度（观察第 5 列<sup>14</sup> “->” 符号后的路径名长度）。

```
$ cd /etc/rc2.d
$ ls -l S*
...
lrwxrwxrwx 1 root root 15 2008-07-09 15:59 S11klogd -> ../init.d/klogd
lrwxrwxrwx 1 root root 14 2008-07-09 16:41 S12dbus -> ../init.d/dbus
lrwxrwxrwx 1 root root 15 2008-07-10 16:41 S15bind9 -> ../init.d/bind9
lrwxrwxrwx 1 root root 13 2008-08-06 15:13 S16ssh -> ../init.d/ssh
...
$
```

Linux 中也存在 3 个与符号链接文件有关的系统调用，其目的是访问符号链接本身及其引用的目的文件或目录的相关信息，这些系统调用如下。

- `readlink()`：用于读取符号链接文件引用的目的文件或目录的路径名，这些信息存储在符号链接文件的数据块中。

- lstat(): 其功能类似于 stat 系统调用，除用于获取目的文件的属性信息外，还可用于获取符号链接文件本身的相关信息。
- lchown(): 类似于 chown 系统调用，既可用于修改目的文件的属主属性，也可用于修改符号链接文件本身的属主属性。

如果想把上述的 samba 文件改为符号链接方式，可以利用 ln 命令的 “-s” 选项，创建下列两个符号链接文件。

```
$ cd /etc/init.d
$ sudo ln -s samba ../rc2.d/S90samba
$ sudo ln -s samba ../rc0.d/K10samba
$ ls -l samba
-rwxr-xr-x 1 root root 2932 2008-09-10 20:10 samba
$ ls -l ../rc2.d/S90samba
1rwxrwxrwx 1 root root 5 2008-09-10 20:10 ../rc2.d/S90samba -> samba
$ ls -l ../rc0.d/K10samba
1rwxrwxrwx 1 root root 5 2008-09-10 20:10 ../rc0.d/K10samba -> samba
$
```

在实际应用中，文件链接具有重要的用途。例如，在构建互为备份的双机系统中，通常采用两个系统共享同一个磁盘阵列的形式，实现数据共享。在这种实现方式中，有的 HA 软件要求比较高，如要求磁盘阵列的设备名必须一致。而由于种种原因，这一限制可能很难达到要求。为了解决这个问题，便可以利用 ln 命令建立链接或符号链接文件。

### 4.3.6 管道文件

Linux 系统存在两种管道，即普通管道（简称管道）和管道文件。普通管道是一个可用文件描述符标识和存取的数据缓冲区，管道内的数据按先进先出的方式处理。普通管道通常是在程序中利用 pipe(2) 系统调用建立的。当程序执行结束后，管道也就自动消失了。

管道文件与普通管道的功能基本相同，只是其创建的方式不同。管道文件是通过下列 mknod 命令或 mknod() 系统调用创建的，而且作为一个特殊文件，它存在于文件系统中，故管道文件也被称为命名的管道（named pipe）。

```
# mknod pipefile p
```

普通管道是一种进程之间的通信机制，而管道文件则是一种特殊文件。管道文件用于缓存接收到的数据，同时供从管道文件中读取数据的进程按照先进先出（FIFO）的方式读取数据。尽管管道文件具有文件名和信息节点，但这个特殊文件并不拥有任何数据。

在下面的例子中，kmsg 就是一个管道文件（第一列文件属性中的第一个字符 p 即管道文件的标志）。

```
$ ls -l /var/run/klogd
总用量 8
-rw-r--r-- 1 klog klog 5 2008-09-07 12:28 klogd.pid
prwx----- 1 klog klog 0 2008-09-07 12:29 kmsg
-rw-r--r-- 1 root root 5 2008-09-07 12:28 kmsgpipe.pid
$
```

## 4.4 文件的安全保护机制

Linux 是一个多用户、多任务的操作系统，为了保证系统和信息的安全，防止用户间未经许可擅自访问他人的文件等，Linux 系统从信息的基本组织单位——文件出发，把用户分为 3 类：



文件属主、同组用户和其他用户（即所有用户）。

对于任何一个文件，可以针对 3 类用户分别赋予一定的访问权限。这些访问权限分为读、写和执行。表 4-2 描述了文件的 3 种基本访问权限及其意义。

表 4-2

文件的 3 种基本访问权限

访问权限	解释
r (读)	如果文件具有读许可，则相应的用户可以读文件，如显示文件内容、复制文件等，但不能修改文件。如果允许用户进入某个目录（cd 命令），列举目录下的文件，则至少应赋予用户“读”目录的访问权限
w (写)	如果文件具有写许可，则相应的用户可以读、写文件，包括显示文件内容以及复制、修改、移动和删除文件等。对于目录而言，如果允许用户创建新文件和删除文件，则必须赋予用户“写”目录的访问权限
x (执行)	如果具有执行许可，则相应的用户可以运行文件（如程序文件）。对于目录而言，如果允许用户访问其中的任何子目录，则必须赋予用户“执行”目录的访问权限

通常，任何用户都可能会允许其他用户读自己的文件，但大多不会允许别人对文件作任何修改。对于普通的程序文件，也可能会允许任何人都能执行。Linux 系统的文件目录安全机制使用户能够控制文件和目录的访问权限。

#### 4.4.1 显示文件的访问权限

要了解文件的访问权限及其属性，可以使用带有“-l”选项的 ls 命令，按文件名的字符顺序以长格式显示每一个文件的各种属性信息，如图 4-6 所示。

```
$ ls -l /
总用量 82
drwxr-xr-x  2 root root  4096 2008-12-05 16:07 bin
drwrxr-xr-x  3 root root  4096 2008-12-05 17:53 boot
lcwrxrwxrwx  1 root root   11 2008-12-05 12:58 cdrom -> media/cdrom
drwxr-xr-x  15 root root 13900 2008-12-13 21:31 dev
drwxr-xr-x 124 root root 12288 2008-12-13 21:33 etc
...
      用户    用户组

        其他用户的访问权限（是否可读、可写、可执行）
        同组用户的访问权限（是否可读、可写、可执行）
        文件属主的访问权限（是否可读、可写、可执行）
```

图 4-5 显示文件目录的访问权限

如前所述，ls 命令输出第一列中的第一个字符表示文件的类型，后面 9 个字符表示文件或目录的访问权限。这些访问权限可分为 3 组，分别对应 3 组用户：文件属主、同组用户和其他用户。前 3 个字符为一组，表示文件属主对此文件的访问权限；中间 3 个字符为一组，表示与文件属主同组的用户对此文件的访问权限；最后 3 个字符为一组，表示其他所有用户对此文件的访问权限。每一组的 3 个字符依次为 r、w 和 x，分别表示读、写和执行。

如果 r、w 或 x 存在，则表示相应的用户分别具有读、写或执行的访问权限。下面以/etc 目录和/etc/profile 文件为例予以说明。

```
$ ls -ld /etc
drwxr-xr-x 132 root root 12288 2008-09-07 11:30 /etc
$ ls -l /etc/profile
-rw-r--r--  1 root root 497 2008-07-25 11:50 /etc/profile
$
```

在上述例子中，/etc 目录的访问权限是“rwxr-xr-x”，表示每个人都能读和执行这个目录，

但只有目录的属主 root 才能写这个目录。/etc/profile 文件的访问权限是“rw-r--r--”，表示文件属主 root 能够读写这个文件，同组用户和其他用户只能读这个文件，但任何人（包括文件属主）都不能执行这个文件。

## 4.4.2 修改文件的访问权限

若要修改文件或目录的访问权限，可以使用 chmod 命令。修改文件或目录访问权限的前提是，用户必须是文件或目录的属主——具有修改文件访问权限的权力，或者是超级用户。chmod 命令的语法格式如下。

```
chmod permissions dir-or-file
```

其中，permissions 表示新的访问权限，dir-or-file 是准备修改其访问权限的文件或目录。

在设置文件或目录的访问权限时，可以对现有的访问权限进行增减性的调整，也可以直接设置。下面简单说明怎样在现有文件访问权限的基础上进行修改。

(1) 使用一个或多个字符表示用户的类型：

- u (表示文件属主);
- g (表示同组用户);
- o (表示其他用户);
- a (表示所有用户)。

(2) 使用加号 “+” 和减号 “-” 分别表示增加或撤销相应的访问权限。

(3) 使用一个或多个字符表示实际的访问权限：

- r (表示读);
- w (表示写);
- x (表示执行)。

在下面的 chmod 命令中，“o+w” 表示增加其他用户写 script 目录的访问权限。

```
$ cd /home/gqxing
$ ls -ld script
drwxr-xr-x 2 gqxing gqxing 4096 2008-06-08 09:10 script
$ chmod o+w script
$ ls -ld script
drwxr-xrwx 2 gqxing gqxing 4096 2008-06-08 09:10 script
$
```

从 “ls -l” 命令的输出结果可以看出，执行 “chmod o+w script” 命令之后，script 目录相对于其他用户写目录的权限标志已由 “-” 变为 “w”。

为了撤销其他用户读和执行同一目录的访问权限，可使用下列 chmod 命令。

```
$ ls -ld script
drwxr-xr-x 2 gqxing gqxing 4096 2008-06-08 09:10 script
$ chmod o-rx script
$ ls -ld script
drwxr-x-- 2 gqxing gqxing 4096 2008-06-08 09:10 script
$
```

现在，表示其他用户读和执行目录的权限标志已由 “r” 和 “x” 全部变为 “-”。

创建一个新文件时，系统会自动设置下列访问权限（参见/etc/profile 文件中的 umask 命令）：

-rw-r--r--

创建一个新目录时，系统会自动设置下列访问权限：



```
drwxr-xr-x
```

因此，在创建了一个新的 `settcp` 脚本文件之后，为了确保只有文件属主才有执行脚本文件的访问权限，可以使用下列 `chmod` 命令。

```
$ ls -l settcp
-rw-r--r-- 1 gqxing gqxing 1670 2008-06-08 09:26 settcp
$ chmod u+x settcp
$ ls -l settcp
-rwxr--r-- 1 gqxing gqxing 1670 2008-06-08 09:26 settcp
$
```

如果想增加或撤销所有用户的访问权限，可以使用 `chmod` 命令的“`a+`”或“`a-`”选项。例如，为了使所有的用户都能够执行 `settcp` 文件，可输入下列 `chmod` 命令。

```
$ ls -l settcp
-rw-r--r-- 1 gqxing gqxing 1670 2008-06-08 09:26 settcp
$ chmod a+x settcp
$ ls -l settcp
-rwxr-xr-x 1 gqxing gqxing 1670 2008-06-08 09:26 settcp
$
```

在上述输出信息中，访问权限字段（第一列后 9 个字符）中的 3 个“`x`”表示所有用户均可以执行 `settcp` 文件。

此外，还可以使用星号“`*`”通配符，同时修改多个文件的访问权限。例如，为了设置当前目录中所有文件的访问权限，使得除了文件属主能够写之外，其他所有用户均不能修改这些文件，可以使用下列 `chmod` 命令。

```
$ ls -l
总用量 16
-rwxrwxrwx 1 gqxing gqxing 1608 2008-06-08 09:18 setftp
-rwxrwxrwx 1 gqxing gqxing 4589 2008-06-08 09:20 setnfs
-rwxrwxrwx 1 gqxing gqxing 1670 2008-06-08 09:26 settcp
$ chmod go-w *
$ ls -l
总用量 16
-rwxr-xr-x 1 gqxing gqxing 1608 2008-06-08 09:18 setftp
-rwxr-xr-x 1 gqxing gqxing 4589 2008-06-08 09:20 setnfs
-rwxr-xr-x 1 gqxing gqxing 1670 2008-06-08 09:26 settcp
$
```

### 4.4.3 设置文件的访问权限

上一小节讨论了怎样使用 `chmod` 命令，在文件现有访问权限的基础上进行调整。这种文件访问权限设置法称为相对权限设置法，本小节将要讨论的方法可以称作绝对权限设置法——使用 `chmod` 命令和数字代码，对文件的访问权限直接进行设置。`chmod` 命令的语法格式如下：

```
chmod numcode dir-or-file
```

其中，`numcode` 是一个数字代码，用于表示文件的访问权限，`dir-or-file` 是文件或目录的名字。

表示文件访问权限的数字代码由 3 位数字组成，分别对应于文件属主、同组用户和其他用户。例如，下列 `chmod` 命令设置的访问权限表示，`setftp` 文件的属主能够读、写和执行 `setftp` 文件，而其他用户只有读和执行的权限。

```
$ ls -l setftp
-rw-r--r-- 1 gqxing gqxing 1608 2008-06-08 09:18 setftp
$ chmod 755 setftp
$ ls -l setftp
-rwxr-xr-x 1 gqxing gqxing 1608 2008-06-08 09:18 setftp
$
```

表 4-3 解释了数字代码 755 表示的访问权限，以及 755 数字代码的由来。表中的后面三列每

列对应一类用户。

表 4-3

setftp 文件访问权限的设定

访问权限	文件属主	同组用户	其他用户
读	100 (4)	100 (4)	100 (4)
写	010 (2)	000 (0)	000 (0)
执行	001 (1)	001 (1)	001 (1)
全部访问权限	7	5	5

可以把表示访问权限的字符串“rwx”看作3个二进制数据，如果“r”存在，则表示相应数据位为1，写成二进制的数就是“100”，也就是十进制的4。同样，“w”对应的二进制数是“010”，即十进制的2。“x”对应的二进制数是“001”，即十进制的1。

若想设置读文件的访问权限，可在适当的行列交叉位置加一个4。若想设置写文件的访问权限，可在适当的行列交叉位置加一个2。同样，若想设置执行文件的访问权限，可在适当的行列交叉位置加一个1。把3列的3行数值分别加在一起，然后再把这3个数据组合起来，就是一个表示文件访问权限的完整数字代码。

例如，为了使所有的用户都能够读、写和执行 setnfs 文件，可以使用下列命令。其中，数字代码777是能够设置的最大文件访问权限。

```
$ ls -l setnfs
-rw-r--r-- 1 gqxing gqxing 4589 2008-06-08 09:20 setnfs
$ chmod 777 setnfs
$ ls -l setnfs
-rwxrwxrwx 1 gqxing gqxing 4589 2008-06-08 09:20 setnfs
$
```

表4-4解释了上述访问权限数字代码的意义及由来。

表 4-4

设置 setnfs 文件访问权限的数字代码

访问权限	文件属主	同组用户	其他用户
读	4	4	4
写	2	2	2
执行	1	1	1
全部访问权限	7	7	7

类似于相对权限设置法，在采用绝对权限设置法时也可以使用星号“\*”通配符，对选定的文件设置同一访问权限。例如，可以使用 chmod 命令，统一设置当前目录下所有文件的访问权限，如下所示。

```
$ ls -l
总用量 16
-rw-r--r-- 1 gqxing gqxing 1608 2008-06-08 09:18 setftp
-rw-r--r-- 1 gqxing gqxing 4589 2008-06-08 09:20 setnfs
-rw-r--r-- 1 gqxing gqxing 1670 2008-06-08 09:26 settcp
$ chmod 755 *
$ ls -l
总用量 16
-rwxr-xr-x 1 gqxing gqxing 1608 2008-06-08 09:18 setftp
-rwxr-xr-x 1 gqxing gqxing 4589 2008-06-08 09:20 setnfs
-rwxr-xr-x 1 gqxing gqxing 1670 2008-06-08 09:26 settcp
$
```



上述例子中的两个 ls 命令解释了使用 chmod 命令前后文件访问权限的变化。

使用绝对权限设置的最大好处是，无需知道文件当前的访问权限，用户只需按照自己的意愿直接设置即可。

#### 4.4.4 其他访问权限设置

##### 1. 默认访问权限

无论何时创建一个文件，Linux 系统通常都会为用户设置一个默认的访问权限。这种默认的访问权限是由 umask 命令实现的。在用户注册之后，通过执行系统范围的/etc/profile 初始化文件，Linux 系统会利用 umask 命令，为用户创建的文件设置访问权限。umask 命令的语法格式简写如下：

```
umask [-S] [nnn]
```

其中的 nnn 为访问权限的数字代码。这个数字代码与 chmod 命令中的数字代码意义恰好相反：在 chmod 命令中，八进制的数字代码 777 意味着任何人均具有读、写和执行文件的访问权限；而在 umask 命令中，数字代码 777 意味着任何人均没有访问权限，包括文件属主本人也是如此。

因此，在使用 umask 设置默认的文件访问权限时，可把八进制的数字代码“000”变为二进制的“000 000 000”。以此为基础，如果想屏蔽某一访问权限，可在相应的位置置 1，最后再转换成八进制的数字代码即可。例如，如果期望用户能够读、写和执行自己新建的文件，同组和其他用户只能读和执行文件，则其二进制数字代码为“000 010 010”，转换为八进制数字代码即为“022”。因此，可以使用“umask 022”命令设置默认的文件访问权限（参见第 13 章）。

##### 2. 设置特殊的访问权限

前面已经讨论了如何设置文件的访问权限，任何用户均可用以保护自己的文件。还有一些特殊的文件访问权限，可由超级用户进行设置。Linux 系统中的许多管理文件，只有超级用户才有权修改，但有时也需要普通用户能够像超级用户一样做必要的修改。例如，对于管理用户账号和密码的/etc/passwd 和/etc/shadow 文件，普通用户是不能修改的。但用户经常需要修改自己的密码，而普通用户又不能修改这些文件，每次都要经过系统管理员。

为此，Linux 系统也采用了实际用户 ID（实际用户组 ID）和有效用户 ID（有效用户组 ID）的概念。通过设置有效用户 ID，可使用户能够临时地以超级用户的身份执行某些特权命令。例如，任何用户在使用 passwd 命令修改密码期间，其有效用户 ID 暂时变为超级用户 ID，因而可以访问 shadow 等文件，修改自己的密码。一旦退出 passwd 命令，用户的有效 ID 又恢复原状。

为了设置某一文件的有效用户 ID 或有效用户组 ID，超级用户可以分别使用下列命令（访问权限字段中的第一个“s”表示已经设置了有效用户 ID。针对本例而言，无需再设置有效用户 ID，实际上也不必设置有效用户组 ID，这里只是说明怎样设置而已）。

```
# ls -l /usr/bin/passwd
-rwsr-xr-x 1 root root 29104 2008-04-03 09:08 /usr/bin/passwd
# chmod u+s /usr/bin/passwd
# chmod g+s /usr/bin/passwd
# ls -l /usr/bin/passwd
-rwsr-sr-x 1 root root 29104 2008-04-03 09:08 /usr/bin/passwd
#
```

在设置了有效用户 ID 之后，必须先由超级用户执行一次相应的命令，才能使上述设置发挥作用。当普通用户执行相应的命令时，其有效用户 ID 才能变为超级用户的有效用户 ID。注意，有效用户 ID 仅在执行过程中才能生效，因此，只有对命令或程序文件设置有效用户 ID 才有实际的意义。

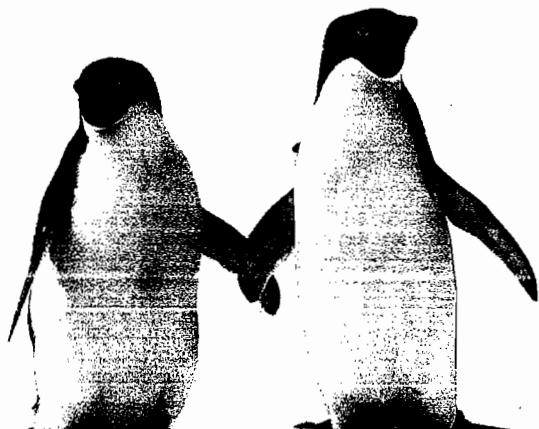
# LINUX

## 第5章

### 文件和目录操作

文件和目录操作是每个系统都必不可少的基本功能。Linux 系统提供了大量的文件与目录处理的命令和工具，用于创建、显示、移动、删除、复制和维护文件与目录。本章主要介绍 Ubuntu Linux 系统中的各种文件与目录操作命令，详细说明怎样创建、显示、移动、删除、复制和维护文件与目录。本章的后面还将介绍若干有用的实用程序，如 find、diff、grep 和 sort 等。其内容主要包括：

- 创建、显示、复制、移动和删除文件与目录；
- 确定文件内容及类型；
- 创建链接和符号链接文件；
- 比较文件之间的差别；
- 从系统中检索文件；
- 检索文件内容；
- 排序。





## 5.1 创建文件

尽管没有明显的专门命令用于创建新文件，但实现创建新文件的方法有许多种。

其中，第一种方法是利用 touch 命令创建一个新的空文件。touch 命令的本意是以当前（或指定）的时间更新给定文件的访问和修改时间。如果指定的文件不存在，则创建一个新的空文件，如下列命令所示。

```
$ touch emptyfile  
$
```

而第二种方法相对更简单，即借用重定向符号“>”，创建一个新的空文件，命令如下。

```
$ > emptyfile  
$
```

第 3 种方法是采用 echo 命令创建一个含有一行数据的新文件。echo 命令的本意是显示提示或说明信息，这里借用 echo 命令，将其标准输出重定向到一个文件中，从而创建一个新文件，如下所示。

```
$ echo "Only one line in file" > newfile  
$
```

第 4 种方法是用 cat 命令创建一个具有多行数据的新文件。cat 命令主要用于显示文件的内容，这里借用 cat 命令读取标准输入的特点，直接用键盘输入数据，从而创建一个拥有 3 行数据的新文件。其中，“Ctrl-D”是 Linux 系统中的文件结束符，例如下列命令所示。

```
$ cat > myfile  
The text I am typing will be stored in "myfile".<Enter>  
Press RETURN at the end of each line.<Enter>  
When finished, hold down the Ctrl key and press D.<Enter>  
Ctrl-D  
$
```

此外，还可以使用 vim 等文本编辑器创建和编辑文件（详见第 6 章）。实际上，创建文件的方法还有很多，等熟悉了 Linux 系统之后，读者还会找出其他更多的方法。

## 5.2 显示文件列表

### 5.2.1 使用 ls 命令显示文件列表

了解系统中都有什么文件，了解自己当前目录下都有哪些文件，也许是用户最常见的动作了。为此，可以使用 ls 命令。ls 命令的语法格式简写如下。

```
ls [options] [dir-or-file]
```

ls 命令具有很多选项，表 5-1 给出了部分常用的选项。

表 5-1

ls 命令的部分常用选项

选 项	GNU 选 项	简 单 说 明
-a	--all	列出指定（或当前）目录下的所有文件，包括以句点“.”作为起始字符的隐藏文件
-b	--escape	当文件名包含不可打印的特殊字符时，以八进制数字形式列出文件的名字

续表

选 项	GNU 选项	简 单 说 明
-d	--directory	在使用 ls 命令列举文件时，如果指定的参数是一个目录，仅列出目录的名字，而不是列出目录下的文件。“-d”选项经常与“-l”选项一起使用，以便用户了解目录的属性信息
-h	--human-readable	以 KB、MB 和 GB 的形式显示文件的大小（这个选项应与“-l”或“-s”等选项一同使用）
-i	--inode	对于每一个文件，在第一列列出其信息节点号
-k	--block-size=1K	以 KB 为单位给出文件的大小
-l	--format=long	以每行一个文件的长格式列出文件的类型、访问权限、链接数、用户属主、用户组、文件大小、最后修改时间和文件名等信息。如果是特殊文件，文件大小字段分为两个数字字段，分别表示设备的主次设备号。如果是一个符号链接文件，则以“filename → 被引用文件的路径名”的形式给出文件名字段
-r	--reverse	以文件名反向字符排序的顺序显示文件列表
-R	--recursive	递归地列出指定（或当前）目录及其子目录下的所有文件
-s	--size	显示分配给文件的数据块（1 024 字节）的数量，即文件占用的数据块数量，而非文件的实际大小。因此，对于文件大小相近的文件，“-s”选项给出的结果完全可能是一样的

例如，为了列出当前目录下的文件，最简单的 ls 命令形式如下。

```
$ ls
emptyfile myfile newfile
$
```

默认情况下，ls 命令将会按照文件名的字符顺序列出当前目录下的所有文件。上述例子中的 ls 命令未加任何选项，也没有明确地指定目录或文件参数，则默认为当前目录。输出结果说明当前目录下只有 3 个刚才创建的文件。如果想了解其他某个特定目录下都有什么文件，可在 ls 命令后面明确地指定目录名，例如下列命令所示。

```
$ ls /
bin  dev  initrd      lib      mnt  root  sys  var
boot etc  initrd.img  lost+found  opt  sbin  tmp  vmlinuz
cdrom home  initrd.img.old  media  proc  srv  usr  vmlinuz.old
$
```

上述 ls 命令按照文件名的字符顺序列出了根目录下的所有文件。读者可能已经发现，使用 ls 命令而不加任何选项，系统仅仅列出文件的名字，至于这些文件究竟是目录、普通文件、特殊文件、管道文件，还是链接文件，则不得而知。为了能够了解更多的信息，可使用“-l”等选项，例如下列命令所示。

```
$ ls -l
总用量 8
-rw-r--r-- 1 gqxing gqxing 0 2008-09-10 18:31 emptyfile
-rw-r--r-- 1 gqxing gqxing 138 2008-09-10 18:36 myfile
-rw-r--r-- 1 gqxing gqxing 22 2008-09-10 18:33 newfile
$
```

使用“-l”选项，ls 命令将会按照一行 8 列的形式逐行显示每个文件的属性。其中第一列包含 10 个字符，第一个字符表示文件的类型。常见的文件类型字符概述如下：

- - 表示相应的文件是一个普通文件；
- d 表示相应的文件是一个目录；
- l 表示相应的文件是一个符号链接文件；
- b 表示相应的文件是一个块特殊文件；
- c 表示相应的文件是一个字符特殊文件；
- p 表示相应的文件是一个管道（FIFO）文件；



□ s 表示相应的文件是一个套接字文件。

其余 9 个字符可分为 3 组，分别表示文件属主、同组用户和其他用户对相应文件的访问权限（详见第 4 章）。第二列是一个数字，表示文件的链接数，如果此数字大于 1，说明同一文件数据存在多个文件名字。第 3 列为用户名，表示文件的属主，说明文件归谁拥有。第 4 列是文件属主所在用户组的名字。第 5 列是文件以字节为单位的大小。第 6 和 7 列给出的是文件最后一次修改的日期和时间。最后一列才真正是文件的名字，如图 5-1 所示。

另外，为了照顾 DOS/Windows 用户的习惯，Linux 系统也提供了一个 dir 命令。只不过，除了命令名不同之外，其命令选项及输出形式与 ls 命令完全相同，但与 DOS/Windows 系统中的情况并不相同。可以说，dir 命令只是 ls 命令的一个替代品，例如下列命令所示。

```
$ dir  
emptyfile myfile newfile  
$ dir -l  
总用量 8  
-rw-r--r-- 1 gqxing gqxing 0 2008-09-10 18:31 emptyfile  
-rw-r--r-- 1 gqxing gqxing 138 2008-09-10 18:36 myfile  
-rw-r--r-- 1 gqxing gqxing 22 2008-09-10 18:33 newfile  
$
```

## 5.2.2 利用通配符显示文件

在 3.6 节中，我们已经介绍了通配符的作用及其与文件名之间的关系。当需要处理一组具有某一共同属性的文件时，可以利用通配符实行模式匹配，生成一个具有同一属性的文件列表。例如，在 Linux 系统中，C 程序大多以“.c”作为文件名后缀。为了显示当前目录下的所有 C 程序，可以使用下列命令。

```
$ ls -l *.c  
-rw-r--r-- 1 gqxing gqxing 4589 2008-09-10 12:56 atmcom.c  
-rw-r--r-- 1 gqxing gqxing 1668 2008-09-10 12:50 atmmon.c  
-rw-r--r-- 1 gqxing gqxing 2790 2008-09-10 13:00 atmstat.c  
-rw-r--r-- 1 gqxing gqxing 2764 2008-09-10 12:59 handler.c  
-rw-r--r-- 1 gqxing gqxing 3198 2008-09-10 12:59 listerner.c  
-rw-r--r-- 1 gqxing gqxing 4660 2008-09-10 12:53 scanner.c  
$
```

在上述命令中，星号“\*”就是一个通配符，能够匹配任何字符或字符串。实际上，可以利用的通配符还有问号“?”（用以匹配任何一个字符），以及字符集“[...]”（用以匹配字符集中的任何一个字符）等。

如前所述，如果不提供文件名参数，ls 命令将会列出当前目录下的所有文件，而星号“\*”通配符又能够匹配任何文件，如果使用下列两个 ls 命令，其输出结果会有什么不同呢？

```
ls -l  
ls -l *
```

上述的第一个 ls 命令中没有指定任何目录或文件参数，因此，系统将会把当前目录作为默认的参数，列出当前目录下的所有文件。第二个 ls 命令使用了星号“\*”通配符作为目录或文件参数，而此处的星号“\*”通配符则表示当前目录下的所有文件。因此，针对第二个 ls 命令，Shell 将会使

权限	链接数	属主	属组	大小	修改日期	时间	文件名
普通用户的访问权限：是否可读、可写、可执行							
同组用户的访问权限：是否可读、可写、可执行							
其他用户的访问权限：是否可读、可写、可执行							
文件类型：“-”表示普通文件，“d”表示目录							

图 5-1 文件属性



用当前目录下的每一个文件作为参数提供给 ls 命令。如果当前目录下只有普通文件，则上述两个命令将会产生同样的输出结果。但是，如果当前目录下含有任何子目录，第一个 ls 命令只是列出了目录的名字，而第二个 ls 命令不仅会列出了目录的名字，还将列出了目录下的所有文件。示例如下。

```
$ cd /etc/power
$ ls -l
总用量 8
drwxr-xr-x 2 root root 4096 2008-04-23 06:03 event.d
drwxr-xr-x 2 root root 4096 2008-04-23 06:03 scripts.d
$ ls -l *
event.d:
总用量 4
-rw-r-xr-x 1 root root 811 2007-12-12 07:00 laptop-mode

scripts.d:
总用量 4
-rw-r-xr-x 1 root root 884 2007-12-12 07:00 laptop-mode
$
```

在使用星号“\*”通配符的情况下，为了避免列出子目录中的文件，可以使用“-d”选项，示例如下。

```
$ ls -ld *
drwxr-xr-x 2 root root 4096 2008-04-23 06:03 event.d
drwxr-xr-x 2 root root 4096 2008-04-23 06:03 scripts.d
$
```

### 5.2.3 显示隐藏文件

在 Linux 系统中，如果文件名的第一个字符为句点“.”，如“.profile”、“.bash\_history”和“.bashrc”，这样的文件被称作隐藏文件。通常，ls 命令不会列出隐藏文件，但是使用 ls 命令的“-a”选项，就可以列出这种隐藏的文件，如下所示。

```
$ ls -la
总用量 422
drwxr-xr-x 55 gqxing gqxing 4096 2008-09-10 18:49 .
drwxr-xr-x 5 root root 4096 2008-09-06 15:03 ..
drwx----- 2 gqxing gqxing 4096 2008-07-15 14:03 .aptitude
-rw----- 1 gqxing gqxing 4492 2008-09-10 19:14 .bash_history
-rw-r--r-- 1 gqxing gqxing 220 2008-06-28 00:00 .bash_logout
-rw-r--r-- 1 gqxing gqxing 2928 2008-06-28 00:00 .bashrc
...
$
```

在 UNIX 系统中，隐藏文件通常位于文件列表的最前面，而在 Linux 系统中，隐藏文件在不考虑第一个句点“.”字符的情况下同普通文件一样参与排序。文件名仅有一个句点字符“.”的文件表示当前目录，文件名为双句点“..”的文件表示父目录。除此之外，其他隐藏文件通常均用于设置用户的工作环境。

此外，当由于某种原因，在命名文件时误输入了不可见的特殊字符时，ls 命令可能无法正确地列出文件的名字。而在使用文件处理命令操作这样的文件时，可能会显示“No such file or directory”等错误信息。例如，见到下面的信息，用户可能会感到莫名其妙，同一目录下怎么会有两个同名的文件呢？这完全不符合 Linux 文件系统的规定。

```
$ ls -l
总用量 16
-rw-r--r-- 1 gqxing gqxing 1268 2008-09-10 19:00 fi?le3
-rw-r--r-- 1 gqxing gqxing 1998 2008-09-10 18:57 fi?le3
-rw-r--r-- 1 gqxing gqxing 1663 2008-09-10 18:56 file1
-rw-r--r-- 1 gqxing gqxing 2556 2008-09-10 18:57 file2
```



```
$ ls -l fi?le3  
-rw-r--r-- 1 gqxing gqxing 1268 2008-09-10 19:00 fi?le3  
-rw-r--r-- 1 gqxing gqxing 1998 2008-09-10 18:57 fi?le3  
$
```

上述两个文件的大小并不相同，明显不是同一个文件。原来，当文件名中包含控制字符等不可打印的特殊字符时，Linux 系统将会采用问号“?”替代任何不可打印的特殊字符。为此，可以使用 ls 命令的“-b”选项，以八进制数字的形式列出文件名中不可见的特殊字符，如下所示。

```
$ ls -lb  
总用量 16  
-rw-r--r-- 1 gqxing gqxing 1268 2008-09-10 19:00 fi\0021e3  
-rw-r--r-- 1 gqxing gqxing 1998 2008-09-10 18:57 fi\0241e3  
-rw-r--r-- 1 gqxing gqxing 1663 2008-09-10 18:56 file1  
-rw-r--r-- 1 gqxing gqxing 2556 2008-09-10 18:57 file2  
$
```

此时，可以利用 mv 命令，重新命名文件，但不能采用问号“?”通配符，否则系统会误以为用户想把两个文件移至某个目录下。为此，可以借助于 Ctrl-V 键，在相应的字符位置上输入控制字符，即可解决文件的改名问题，如下所示。

```
$ mv fi^V^Ble3 file3  
$ mv fi^V^Tle3 file4  
$ ls -l  
总用量 16  
-rw-r--r-- 1 gqxing gqxing 1663 2008-09-10 18:56 file1  
-rw-r--r-- 1 gqxing gqxing 2556 2008-09-10 18:57 file2  
-rw-r--r-- 1 gqxing gqxing 1268 2008-09-10 19:00 file3  
-rw-r--r-- 1 gqxing gqxing 1998 2008-09-10 18:57 file4  
$
```

注意，Ctrl-V 是一个特殊的控制字符，在需要输入各种控制字符或其他特殊字符时，可以先按下 Ctrl-V 键，然后输入想要输入的任何控制字符，即可达到目的。从某种意义上说，Ctrl-V 相当于一个转义符号，可用于引入任何控制字符。

## 5.2.4 递归地列出文件

利用 ls 命令的“-R”选项，可以递归地逐层显示当前（或指定）目录及其子目录下的所有文件，示例如下。

```
$ cd /etc/power  
$ ls -lR  
.:  
总用量 8  
drwxr-xr-x 2 root root 4096 2008-04-23 06:03 event.d  
drwxr-xr-x 2 root root 4096 2008-04-23 06:03 scripts.d  
  
.event.d:  
总用量 4  
-rwxr-xr-x 1 root root 811 2007-12-12 07:00 laptop-mode  
  
.scripts.d:  
总用量 4  
-rwxr-xr-x 1 root root 884 2007-12-12 07:00 laptop-mode  
$
```

注意，“ls -l \*”与“ls -lR”两个命令的意义和输出结果也不相同。前者仅涉及当前目录及其直接子目录两个目录层次中的所有文件，而后者将会遍历当前目录及其所有子目录（包括子目录下的子目录）中的所有文件。

## 5.3 显示文件内容

### 5.3.1 使用 cat 命令显示文件

传统的文件显示命令是 cat，其语法格式简写如下。

```
cat [options] [file]
```

cat 命令的作用是连续地显示文件的内容。不管终端屏幕的窗口有多大，cat 总是从头到尾显示整个文件的所有内容。cat 命令的缺点是，如果文件太大、太长，最终能够见到的只是文件最后面一部分内容，之前的内容将会快速闪过。例如，为了显示 passwd 文件，可以输入下列命令。

```
$ cat /etc/passwd
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/bin/sh
bin:x:2:2:bin:/bin:/bin/sh
sys:x:3:3:sys:/dev:/bin/sh
sync:x:4:65534:sync:/bin:/bin/sync
...
$
```

cat 命令的另一个常见用法是合并两个或多个文件，最终组成一个大的文件。实际上，实现文件合并的方式还有很多种。例如，可以使用下列命令把分章编写的文件合并为一个完整的文件。

```
$ cat chap1 chap2 chap3 chap4 chap5 > user_guide
$ ls
chap1 chap2 chap3 chap4 chap5 user_guide
$
```

如果想把两个文件合并到其中的一个文件，可以使用 cat 命令和 I/O 重定向 (>>) 功能，把第二个文件中的内容附加到第一个文件中，如下所示。

```
cat file2 >> file1
```

其中，第二个文件 file2 的输出被重定向，附加到第一个文件 file1 的后面，最终结果是把第二个文件的内容合并到第一个文件中。

注意，如果采用下列命令形式合并文件，将会清除 file1 的数据内容，仅仅把 file2 的数据内容复制到 file1 中。

```
cat file1 file2 > file1
```

### 5.3.2 使用 more 命令分页显示文件

要从头到尾一页一页地仔细阅读文件，可以使用 more 命令，逐页逐屏地显示整个文件的内容。more 命令的语法格式简写如下。

```
more [options] [file]
```

例如，要逐页显示/etc/profile 文件，可以输入下列命令（其输出结果如图 5-2 所示）。

```
$ more /etc/profile
```

more 命令在显示一页文件内容之后，屏幕窗口的左下角将会出现“--More--(n%)”信息，其中的“n%”表示已显示的数据内容占整个文件的百分比（如果文件很短，上述信息将不会出现）。在 more 命令的控制下，可以使用下列字符命令或控制键，前后滚动终端窗口，逐页逐屏地查看文件。

- iSPACE 显示下一页，或者显示下“i”行（如果指定行数 i 后，接着按下空格键）。
- iEnter 显示下一行，或者显示下“i”行（如果指定行数 i 后，接着按下 Enter 键）。



- **iB (^B)** 回显前一页，或者回显前数第 “*i*” 页（如果指定页数 *i* 后，接着输入字符 “**b**” 或按下 **Ctrl-B** 键）。
- **id (^D)** 显示下半页（初始值为 11 行），或者显示下 “*i*” 行（如果指定行数 *i* 后，接着输入字符 “**d**” 或按下 **Ctrl-D** 键）。
- **if** 显示下一页或第 “*i+1*” 页（如果指定页数 *i* 后，接着输入字符 “**f**”）。
- **is** 跳过指定的行数之后，接着显示下一页。
- **^L** 重新显示终端窗口中原有的数据内容。
- **v** 调用默认的 vim 编辑器，编辑当前文件，并把光标定位在当前行。退出 vim 后再返回 more 命令的原位置。
- **il/pattern** 从当前位置开始，检索下一个或下面第 “*i*” 个匹配给定模式的字符串。
- **!command** 调用 Shell 执行指定的命令。
- **:n** 显示下一个文件（按照命令行列举的文件名顺序）。
- **:p** 显示前一个文件（按照命令行列举的文件名顺序）。
- **:f** 显示当前文件的名字与行号。
- **=** 显示当前的行号（数）。
- **.** 重复执行前一个命令。
- **h (?)** 给出 more 命令的简要说明。
- **q (Q)** 退出 more 命令。

注意，这里所谓的下一页或下一行意指文件结尾方向，前一页或前一行意指文件开始方向，“*i*” 的默认值为 1。

```
文件(E) 菜单(I) 菜单(O) 菜单(D) 菜单(H)  
* /etc/profile: system-wide .profile file for the Bourne shell (sh(1))  
* and Bourne-compatible shells (dash(1), ksh(1), ash(1), ...).  
if [ -d /etc/profile.d ]; then  
    for f in /etc/profile.d/*; do  
        if [ -r $f ]; then  
            . $f  
        fi  
    done  
fi  
if [ -r /etc/bashrc ]; then  
    . /etc/bashrc  
fi  
if [ -r /etc/bashrc.local ]; then  
    . /etc/bashrc.local  
fi  
else  
    if [ -r /etc/.bashrc ]; then  
        . /etc/.bashrc  
    else  
        echo "No .bashrc file found."  
    fi  
fi  
PS1=$PS1
```

图 5-2 more 命令的输出结果

### 5.3.3 使用 less 命令分页显示文件

Linux 系统还提供一个与 more 类似的 less 命令，其功能比 more 命令更强，也能够分页显示文件。less 命令的语法格式简写如下。

**less [options] [file]**

例如，要逐页地显示 /etc/inputrc 文件，可以输入下列 less 命令，其输出结果如图 5-3 所示。

\$ **less /etc/inputrc**

首次进入 less 命令，并在显示第一页文件内容之后，less 命令将会在终端窗口的左下角显示当前文件的名字。当使用下列命令，逐页逐屏地滚动翻阅文件后，less 命令将会在终端窗口的左下角显示一个冒号 “:” 提示符。在 less 命令的控制下，用户可以使用下列字符命令或控制键，前后滚动终端窗口，逐页查阅文件。

- **iSPACE** 显示下一页，或者显示下 “*i*” 行（如果指定行数 *i* 后，接着按下空格键）。
- **if (i^P)** 同上。

- ^V** 显示下一页。
- iEnter** 显示下一行，或者显示下“*i*”行（如果指定行数*i*后，接着按下Enter键）。
- i^N** 同上。
- ie (i^E)** 同上。
- ij (i^J)** 同上。
- id (i^D)** 显示下半页，或者显示下“*i*”行（如果指定行数*i*后，接着输入字符“d”或按下Ctrl-D键）。
- ib (i^B)** 回显前一页，或者回显前“*i*”行（如果指定行数*i*后，接着输入字符“b”或按下Ctrl-B键）。
- iu (i^U)** 回显前半页，或者回显前“*i*”行（如果指定行数*i*后，接着输入字符“u”或按下Ctrl-U键）。
- ir (i^R)** 重新显示屏幕中原有的数据内容。
- ^L** 同上。
- :n** 显示下一个文件（按照命令行列举的文件名顺序）。
- :p** 显示前一个文件（按照命令行列举的文件名顺序）。
- :f** 显示当前文件的名字与行号等信息。
- /pattern** 从当前位置开始，往下检索匹配给定模式的字符串。
- ?pattern** 从当前位置开始，往前检索匹配给定模式的字符串。
- !command** 调用Shell执行指定的命令。
- h (H)** 给出less命令的简要说明。
- q (Q)** 退出less命令。

同样，这里所谓的下一页或下一行意指文件结尾方向，前一页或前一行意指文件开始方向，“*i*”的默认值为1。

### 5.3.4 使用head命令显示文件前几行内容

有时，只需查看文件的前几行内容，即可对文件有一个清楚的了解。此时，可以使用head命令，其语法格式简写如下。

head [-number | -n number] [file]

其中，number是一个数字，表示需要输出的行数。默认情况下，head命令将显示给定文件的前10行内容（包括空行），例如下列命令所示。

```
$ head /etc/passwd
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/bin/sh
bin:x:2:2:bin:/bin:/bin/sh
sys:x:3:3:sys:/dev:/bin/sh
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/bin/sh
man:x:6:12:man:/var/cache/man:/bin/sh
lp:x:7:7:lp:/var/spool/lpd:/bin/sh
```

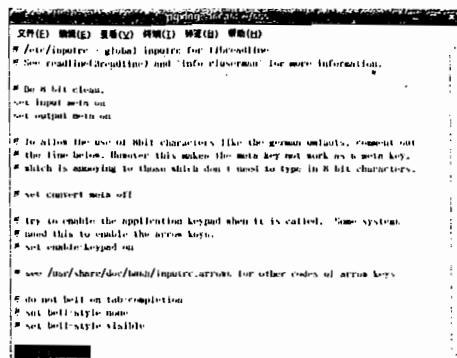


图5-3 less命令的输出结果



```
mail:x:8:8:mail:/var/mail:/bin/sh  
news:x:9:9:news:/var/spool/news:/bin/sh  
$
```

### 5.3.5 使用 tail 命令显示文件最后几行内容

对于比较大的日志文件，有时只想查看文件后面的最新信息。此时需要用到 tail 命令，其语法格式简写如下。

```
tail [+number [-lbcf]] [file]
```

其中，加号“+”表示从文件的起始位置开始计算，减号“-”表示从文件的结束位置开始计算。`number` 是一个数字，表示需要输出的行数。默认情况下，tail 命令将会显示给定文件的最后 10 行（包括空行）的数据内容，例如下列命令所示。

```
$ tail /var/log/dmesg  
[ 62.872216] eth0: link down  
[ 63.733755] ACPI: Battery Slot [BAT0] (battery present)  
[ 64.007658] lp0: using parport0 (interrupt-driven).  
[ 64.152591] Adding 748400k swap on /dev/sda8. Priority:-1 extents:1 across:748400k  
[ 64.660109] EXT3 FS on sda7, internal journal  
[ 64.977178] device-mapper: uevent: version 1.0.3  
[ 64.977258] device-mapper: ioctl: 4.12.0-ioctl (2007-10-02) initialised:  
dm-devel@redhat.com  
[ 67.482842] ip_tables: (C) 2000-2006 Netfilter Core Team  
[ 68.216591] PPP generic driver version 2.4.2  
[ 68.296146] NET: Registered protocol family 17  
$
```

上述 tail 命令只能查看静态文件的最后几行内容。对于不断增长的日志文件，有时需要持续地监控文件不断出现的最新信息，此时可以使用带有“-f”选项的 tail 命令。例如，为了监控某个日志文件的最新变化，可以使用下列命令。

```
tail -f somelogfile
```

## 5.4 复制文件

在开发程序时，经常需要复制以前的程序，并以此为基础进行修改，这时就要用到文件的复制命令 cp。cp 命令的语法格式简写如下。

```
cp [-ir] source_file target_file
```

其中，“-i”选项表示交互复制方式。当指定的目的文件存在时，cp 命令将会提示用户存在同名的文件。仅当用户输入 y 确认之后，cp 命令才会继续执行复制；其他任何回答将会终止复制的执行。“-r”选项表示递归复制方式。当指定的源文件为目录时，cp 命令将会递归地复制目录及其中的任何文件，包括子目录及其中的文件。

假定我们想利用先前创建的 newfile 文件复制一个 newcopy 文件，可以使用下列命令。

```
$ cp newfile newcopy  
$
```

此时，当前目录下将会增加一个新复制的文件，利用 ls 命令对其进行检验，其结果如下。

```
$ ls -l  
总用量 8  
-rw-r--r-- 1 gqxing gqxing 22 2008-09-10 18:35 newcopy  
-rw-r--r-- 1 gqxing gqxing 22 2008-09-10 18:33 newfile  
$
```

使用 cp 命令不仅可以复制当前目录下的文件，也可以复制其他目录中的文件。例如，下列命

令将会把/etc 目录下的 profile 文件复制到当前目录，文件名保持不变。

```
$ cp /etc/profile .  
$
```

值得注意的是，在使用 cp 命令而不加任何选项时，如果访问权限许可，有可能误删已有的同名文件，将同名的文件覆盖，造成不应有的损失。因此，比较保险的做法是在使用 cp 命令时，有意识地增加“-i”选项。使用“-i”选项时，如果目的文件不存在，cp 命令将会正常地执行；如果目的文件存在，系统将会输出提示信息，请求用户予以确认，示例如下。

```
$ cp -i somefile somefile2  
cp: 是否覆盖 "somefile2"?  
$
```

此外，利用“-r”选项，cp 命令还能够复制整个目录（包括目录下的所有子目录及文件），详情请参见 5.11 节。

## 5.5 移动文件

使用 mv 命令，可以把文件从一个目录移到另外一个目录中，或者重新命名一个文件。mv 命令的语法格式简写如下。

```
mv [-f] source_file target_file
```

其中，“-f”选项为强制移动或改名，“-i”选项意味着，一旦目标目录或文件存在，在取得用户的认可后才能采取下一步的处理动作。下面是一个利用 mv 命令把 newfile 文件移至/tmp 目录，而文件名保持不变的例子。

```
$ ls -l  
总用量 8  
-rw-r--r-- 1 gqxing gqxing 22 2008-09-10 18:35 newcopy  
-rw-r--r-- 1 gqxing gqxing 22 2008-09-10 18:33 newfile  
$ mv newfile /tmp  
$ ls -l  
总用量 4  
-rw-r--r-- 1 gqxing gqxing 22 2008-09-10 18:35 newcopy  
$ ls -l /tmp/newfile  
-rw-r--r-- 1 gqxing gqxing 22 2008-09-10 18:33 /tmp/newfile  
$
```

下面的例子则利用 mv 命令把 newcopy 文件改名为 oldcopy。

```
$ mv newcopy oldcopy  
$
```

再次使用 ls 命令，可以看到原来的文件已经不存在了，newcopy 已被改名为 oldcopy。

```
$ ls -l  
总用量 4  
-rw-r--r-- 1 gqxing gqxing 22 2008-09-10 18:35 oldcopy  
$
```

另外，rm 命令也存在一个潜在的问题，即当新命名的文件已经存在时，如果访问权限许可，同名的文件将被覆盖。为了防止此类情况的出现，可在 mv 命令中使用“-i”选项，一旦同名文件存在，系统将会提示用户，且仅在用户认可的情况下才会覆盖原有的文件，从而避免出现误删文件的问题，如下所示。

```
$ mv -i file1 file2  
mv: 是否覆盖 "file2"?  
$
```

为了重新命名一个文件，还有一个比较有用的命令——basename。basename 命令的本意是剔



除文件名中的目录部分，即文件路径名中最后一个斜线字符“/”之前的所有目录前缀，使之仅包含文件名本身，或者删除文件名的扩展名后缀。其语法格式简写如下。

```
$ basename string [suffix]
```

例如，假定有一个 C 程序/home/gqxing/src/atmcom.c，使用下列命令即可取出删除了目录路径名和“.c”后缀的文件名 atmcom。

```
$ basename /home/gqxing/src/atmcom.c .c  
atmcom  
$
```

为了编译任何 C 程序，把编译后的 a.out 文件改名为相应的程序文件，可以创建下列 Shell 脚本（注意，命令前后加反向单引号“`”表示用命令的输出结果替换当前位置的内容，详情请参见 7.3 节）：

```
$ cat comp  
#!/bin/bash  
  
gcc $1  
mv a.out `basename $1 .c`  
$
```

其中的\$1 是位置参数，可以是提供给 comp 脚本的任何一个 C 程序文件（参见第 7 章）。执行下列命令后即可编译 atmcom.c 源程序，把编译后的 a.out 程序文件改名为 atmcom。

```
$ cd /home/gqxing/src  
$ comp atmcom.c  
$
```

再如，为了把扩展名为“.old”的文件改名为扩展名为“.c”的 C 源程序文件，可以使用“\*.old”作为参数，运行下列 Shell 脚本。

```
$ cat rename  
#!/bin/bash  
  
for i in $*  
do  
    mv $i `basename $i .old`.c  
done  
$ ls -l *.old  
-rw-r--r-- 1 gqxing gqxing 6174 2008-12-24 17:27 atmcom.old  
-rw-r--r-- 1 gqxing gqxing 6202 2008-12-24 17:27 listener.old  
-rw-r--r-- 1 gqxing gqxing 6324 2008-12-24 17:29 status.old  
$ rename *.old  
$ ls -l *.c  
-rw-r--r-- 1 gqxing gqxing 6174 2008-12-24 17:27 atmcom.c  
-rw-r--r-- 1 gqxing gqxing 6202 2008-12-24 17:27 listener.c  
-rw-r--r-- 1 gqxing gqxing 6324 2008-12-24 17:29 status.c  
$
```

## 5.6 删除文件

如果需要删除文件，可使用 rm 命令，其语法格式简写如下。

```
rm [-rfi] [file]
```

其中，“-r”选项用于递归地删除目录中的文件及目录本身（参见 5.12 节），“-i”选项表示以交互方式执行文件删除操作。在删除任何文件之前，rm 命令将会请求用户予以确认。“-f”选项表示强制删除文件。此外，即使文件不存在，rm 命令也不会输出任何错误信息。

在 Linux 系统中，文件一旦被删除，就很难再恢复，因此在执行文件删除操作时一定要小心谨慎。在使用 rm 命令时，建议增加“-i”选项，这样可以使得在真正删除文件之前还有一次选择

的机会。下面以 newcopy 文件为例，说明怎样使用 rm 命令删除文件。

```
$ ls -l
总用量 8
-rw-r--r-- 1 gqxing gqxing 22 2008-09-10 18:35 newcopy
-rw-r--r-- 1 gqxing gqxing 22 2008-09-10 18:33 newfile
$ rm newcopy
$ ls -l
总用量 4
-rw-r--r-- 1 gqxing gqxing 22 2008-09-10 18:33 newfile
$
```

如果使用“-i”选项删除文件，其情况如下。

```
$ rm -i newcopy
rm: 是否删除普通文件 “newcopy”?
$
```

如果想要一次删除多个文件，可以把文件名一一列在 rm 命令之后，也可以使用通配符，如下所示。

```
$ rm -i new*
rm: 是否删除普通文件 “newfile”? n
rm: 是否删除普通文件 “newcopy”? y
$
```

注意，使用星号“\*”通配符时要小心谨慎，尤其是下列用法更要慎之又慎。

```
$ rm *
$
```

如果对删除操作确有把握，需要强制删除某些文件时（这在部分系统维护脚本中尤为常见），可以使用“-f”选项，强制删除指定的文件，例如下列命令所示。

```
$ rm -f *.tmp
$
```

## 5.7 显示当前工作目录

在 Bash 环境中，命令提示符中通常会包含当前工作目录最低一级的子目录，当需要确定自己当前所处的准确目录位置时，可以使用 pwd 命令。pwd 命令主要用于显示当前的工作目录，以便用户随时了解自己在文件系统目录层次中所处的位置，例如下列命令所示。

```
$ pwd
/usr/include
$
```

实际上，通过设置 PS1 内部变量，也可以在命令提示符中输出完整的当前工作目录（参见第 13 章）。

## 5.8 改换目录

在 Linux 系统中，每个用户都有一个属于自己的主目录。在注册之后，系统将会自动地把用户引导至自己的主目录。当需要转入某个工作目录时，可使用 cd 命令。cd（改换目录）命令可使用户能够进入文件系统的任何目录层次（除非目录的访问权限不允许），例如下例



命令所示。

```
$ cd /usr/lib  
$ pwd  
/usr/lib  
$
```

cd 命令的默认参数为\$HOME，即当前用户的主目录。因此，当输入一个未加任何命令选项与参数的 cd 命令时，相当于执行“cd \$HOME”命令，系统会把当前的工作目录改换到用户的主目录。例如，如果用户的主目录是/home/gqxing，输入下列命令即可返回自己的主目录。

```
$ cd  
$ pwd  
/home/gqxing  
$
```

在 Bash、Korn Shell 及 TC Shell 等 Shell 中，波浪号“~”可用作用户主目录的缩写符号。例如，使用下列命令，可把工作目录改换到当前用户主目录下的 music 子目录。

```
$ cd ~/music  
$
```

也可以使用下列缩写形式，指定任何一个用户的主目录。其中，username 可以是任何注册的用户名。

```
cd ~username
```

在不支持波浪号“~”缩写形式的 Shell（如 Bourne Shell）中，引用用户主目录的替代方法是引用\$HOME 变量，这也是每个 Shell 都支持的通用方法。例如，可以使用下列命令，把工作目录改换到当前用户主目录下的 script 子目录。

```
$ cd ${HOME}/script  
$  
$
```

如前所述，句点“.”表示当前目录，双句点“..”表示父目录。为了从子目录返回父目录，可使用“cd ..”命令，如下所示。

```
$ pwd  
/home/gqxing/script  
$ cd ..  
$ pwd  
/home/gqxing  
$
```

假定当前的工作目录是/home/user1，现准备转到/home/user2 目录下处理其中的文件，可以使用下列命令。

```
$ pwd  
/home/user1  
$ cd ../user2  
$ pwd  
/home/user2  
$
```

在使用 cd 命令时，可以指定绝对目录名，也可以使用相对目录名。所谓绝对目录名，是指从文件系统的根目录“/”开始，按层次结构逐级列出每一级子目录，直至最终子目录的路径名。所谓相对目录，是指相对于当前目录的路径名。

在上述例子中，“cd ../user2”引用的就是相对目录。如果改用下列“cd /home/user2”命令，其中引用的就是绝对路径。

```
$ pwd  
/home/user1  
$ cd /home/user2  
$ pwd  
/home/user2  
$
```

## 5.9 创建目录

许多用户习惯于创建不同的目录，分类存储各种不同的文件。要创建新的目录，可以使用 `mkdir` 命令，同时在命令后面指定新建目录的名字，例如下列命令所示。

```
$ mkdir src
$ cd src
$ mkdir java
$ mkdir c
$ cd java
$ pwd
/home/gqxing/src/java
$ cd ..
$ ls -l
总用量 8
drwxr-xr-x 2 gqxing gqxing 4096 2008-09-12 18:57 c
drwxr-xr-x 2 gqxing gqxing 4096 2008-09-12 18:58 java
$
```

## 5.10 移动目录

同文件操作一样，也可以利用 `mv` 命令，把目录从一个位置移动到另一个位置。例如，利用下列 `mv` 命令，可以把 `src` 目录下的 `java` 子目录移至上一级目录，即移至主目录。

```
$ cd /home/gqxing/src/java
$ mv java ..
$ cd ..
$ ls -l
总用量 16
drwxr-xr-x 2 gqxing gqxing 4096 2008-09-12 18:58 incl
drwxr-xr-x 2 gqxing gqxing 4096 2008-09-12 18:58 java
drwxr-xr-x 2 gqxing gqxing 4096 2008-09-12 18:58 script
drwxr-xr-x 2 gqxing gqxing 4096 2008-09-12 18:58 src
$
```

同样，也可以利用 `mv` 命令重新命名一个目录，为目录改换一个新的名字，如下所示。

```
$ mv script shell
$ ls -l shell
总用量 16
-rw-r--r-- 1 gqxing gqxing 919 2008-09-12 18:58 ftpget
-rw-r--r-- 1 gqxing gqxing 602 2008-09-12 18:58 prime
-rw-r--r-- 1 gqxing gqxing 789 2008-09-12 18:58 tree
-rw-r--r-- 1 gqxing gqxing 640 2008-09-12 18:58 whatday
$
```

## 5.11 复制目录

要把某个目录下的所有文件全部移至另外一个目录，可以使用 `cp` 命令的“`-r`”选项实现，例如下列命令所示。

```
$ cp -r dir1 dir2
$
```



cp 命令的“-r”选项意味着递归复制。执行上述命令后，源目录 dir1 下的所有文件、子目录及其子目录下的所有文件将全部被复制到目的目录 dir2 中。在复制目录时，如果不加“-r”选项，系统将会输出一个错误信息。

另外，还可以利用 cpio 和后面即将介绍的 find 命令实现目录的复制。通过管道组合使用这两个命令，可以构成一个功能非常丰富的复合命令。cpio 命令可以用于备份任何目录或文件系统中的文件，把其中的文件归档后复制到任何存储介质中。它的另一个重要功能就是能够把位于指定目录下的所有目录文件依其原有的层次结构原封不动地复制到另一个目录中。cpio 命令的语法格式简写如下（这里主要使用第 3 种命令格式）。

```
cpio -i [-bBcdfkmnrsStuvV] [-C bufsize] [-H header]
         [-I file] [-M message] [< file]
cpio -o [-aABcLvV] [-C bufsize] [-H header] [-O file] [-M message]
         < fname_list [> file]
cpio -p [-adllLmuVv] [-R id] directory < fname_list
```

其中，“-p”选项表示从标准输入中读取路径文件名，然后按照其他选项的要求，把指定目录中的目录和文件按原有的层次结构复制到目的目录中。表 5-2 给出了其他命令选项的说明。

表 5-2 find 命令的其他部分选项及其简单说明

选 项	GNU 选项	说 明
-a	--reset-access-time	完成文件复制后恢复源文件的访问时间属性，使得 cpio 命令的文件访问不留痕迹
-d	--make-directories	根据源目录层次结构的要求创建目录
-m	--preserve-modification-time	使复制的目的文件保持源文件原有的修改时间不变（但对新创建的目录无效）
-u	--unconditional	无条件的复制。通常，时间较早的老文件不能替换复制目的中同名的新文件
-v	--verbose	显示 cpio 命令处理的文件列表及其属性信息

例如，要把 dir1 目录中的所有子目录及文件原封不动地复制到一个新目录 dir2 中，可使用下列组合命令。

```
$ cd dir1
$ find . -print | cpio -padmuV dir2
...
$
```

## 5.12 删除目录

要删除一个空的目录，可以使用 rmdir 命令，其语法格式简写如下：

```
rmdir [-p] directory
```

如果目录中包含文件，使用 rmdir 命令删除目录时将会出错，此时可以使用带有“-r”选项的 rm 命令。利用“-r”选项删除目录是一种递归操作，可以把指定目录及其任何子目录中的所有文件全部删除，例如下列命令所示。

```
$ rm -r dir2
$
```

注意，使用“rm -r”命令删除目录及其文件时，命令一经提交，通常是无法挽回的。因此，使用这个命令时必须谨慎行事。

## 5.13 比较文件之间的差别

### 5.13.1 使用 diff 命令比较两个文件

当面对两个类似的文件，想找出其中的细微差别时，可以使用 diff 命令进行比较，从中找出两个文件的不同之处。diff 命令的语法格式简写如下。

```
diff file1 file2
```

diff 命令分别读取两个输入文件，逐行分析其中的异同点，从而找出两者之间的差别。当找出不同的行时，diff 命令将会尝试确定出现差别的行是否是由于插入、删除或修改文本行等原因造成的，如果确乎如此，还要检查有多少行受到影响。最后，diff 命令将会告诉用户，每个文件的哪一行，从哪个字符开始，有几个字符不同，同时给出两个文件中存在差别的文本行。

如果两者之间的差别是由于插入新的文本行造成的，diff 命令将会采用下列形式显示新增加的行。

```
lline#1[,lline#2] a rline#1[,rline#2]
```

其中，lline#1 和选用的 lline#2 是第一个文件中的行号，rline#1 和选用的 rline#2 是第二个文件中的行号。

如果两者之间的差别是由于删除文本行造成的，diff 程序将会采用下列形式显示哪一个文件删除了文本行。

```
lline#1[,lline#2] d rline#1[,rline#2]
```

如果两者之间的差别是由于修改文本行造成的，diff 程序将会采用下列形式显示发生变更的文本行。

```
lline#1[,lline#2] c rline#1[,rline#2]
```

在上述的任何情况下，两个文件中的相关文本行将会随行号一并给出。第一个文件中的文本行前面冠以小于号“<”，第二个文件中的文本行前面冠以大于号“>”。

假定有两个文件 test1 和 test2，其中的数据内容分别如下。

```
$ cat test1
You are in a maze of
twisty little passages
which are all alike.
$ cat test2
You are in a maze of
twisty little passages
which are all different.
```

利用 diff 命令对两者进行比较，其输出结果如下。

```
$ diff test1 test2
3c3
< which are all alike.
---
> which are all different.
```

其中的“3c3”表示两个文件的第 3 行数据内容不同，不同是由于部分数据内容的变动造成的。

如果两个文件完全相同，diff 命令将不会显示任何信息。有关 diff 命令的详细说明和其他功能，可以查阅 Linux 系统的随机文档。

### 5.13.2 使用 diff3 命令比较 3 个文件

要比较 3 个不同版本的文件，可使用 diff3 命令，其语法格式简写如下。



```
diff3 file1 file2 file3
```

经过对 3 个不同版本文件的比较，`diff3` 命令将会指出其中是否存在差别。如果三个文件均相同，则不输出任何信息。如果输出：

```
====
```

说明 3 个文件均不相同。如果输出：

```
====1
```

说明第一个文件不相同。同样，如果输出：

```
====2 或====3
```

说明第二或第 3 个文件不相同。然后，通过下列信息说明差别的原因，但不会把存在差别的文本行显示出来。

- *f:nla* 意味着第 “f (f=1,2,3)” 个文件第 “n1” 行之后插入了文本行。
- *f:n1,n2c* 意味着第 “n1” 至第 “n2” 行范围的文本数据已经被修改。如果 “n1=n2”，则文本行的范围可以缩写为 “n1”。

现在，以 Ubuntu Linux 系统的可调参数设置文件`/etc/sysctl.conf`为例，说明`diff3`命令的输出结果。在安装数据库软件的时候，通常需要设置系统的可调参数，以提高数据库的运行效率。假定我们修改了其中的部分参数，当前的`sysctl.conf`文件和`sysctl.conf2`文件的数据内容如下（省略了其他内容）。

```
kernel.shmmni = 4096  
kernel.shmall = 2097152  
kernel.shmmax = 67108864
```

而原始的`sysctl.conf.save`文件的数据内容如下。

```
kernel.shmmni = 4096  
kernel.shmall = 2097152  
kernel.shmmax = 33554432
```

执行`diff3`命令后，其输出结果如下。

```
$ diff3 sysctl.conf sysctl.conf2 sysctl.conf.save  
====3  
1:3c  
2:3c  
    kernel.shmmax = 67108864  
3:3c  
    kernel.shmmax = 33554432  
$
```

上述输出信息说明，第 3 个文件有差别（====3），第一和第二个文件的第 3 行与第 3 个文件的第 3 行不同，其原因是做过修改（c）。

## 5.14 从系统中检索文件

当用户想要找出具有某一特征的文件，或者要了解系统中是否存在某个文件，又不知文件究竟在哪个目录时，可以使用`find`命令。

`find`命令将按用户指定的检索条件，从指定的目录开始，找出满足匹配准则的所有文件。指定的检索条件可以是文件名（包括通配符）、文件大小及文件修改日期等。`find`命令的一般语法格式简写如下。

```
find directory [options]
```

其中，`directory`是检索的起始目录，`options`是一种表达式选项，用于指定各种匹配准则或检索条件。

每个选项均描述一种文件匹配或检索准则，一个文件必须满足所有的匹配或检索原则才能被选中，使用的选项越多，选中的文件越少。表 5-3 列出了`find`命令的部分常用选项及其简单说明。

表 5-3

find 命令的部分常用选项

选 项	说 明
<code>-name filename</code>	检索匹配指定文件名的所有文件。其中，指定的文件名不必包括目录路径。也就是说，只要文件名本身匹配即可。例如，如果指定的文件名为 hosts，则 /etc/hosts 和 /etc/avahi/hosts 等均为匹配检索准则的文件。如果指定的文件名中包括通配符“*”、“?”和 “[...]"，文件名前后应加单引号或双引号
<code>-user username</code>	检索其文件属主匹配指定用户的所有文件。其中，username 可以是任何一个合法的注册用户名，也可以是用户 ID 号
<code>-group groupname</code>	检索其用户组属性匹配指定用户组的所有文件。其中，groupname 可以是任何一个合法的用户组名，也可以是用户组 ID 号
<code>-nouser</code>	检索其文件属主并未在 /etc/passwd 文件中定义的所有文件，即检索其文件属主非本地系统用户的文件
<code>-nogroup</code>	检索其用户组名并未在 /etc/group 文件中定义的所有文件，即检索其用户组非本地系统用户组的文件
<code>-atime [±]n</code>	选择在 n 天之前、之内或恰好 n 天访问过的文件
<code>-ctime [±]n</code>	选择在 n 天之前、之内或恰好 n 天状态信息发生变动的文件
<code>-mtime [±]n</code>	选择在 n 天之前、之内或恰好 n 天修改过文件内容的文件
<code>-newer filename</code>	选择其修改日期比给定文件更近的文件
<code>-depth</code>	表示逐层深入各级子目录，采用先文件后目录的方式，自底向上依次检索所有的文件和目录
<code>-size [±]n[cwbkMG]</code>	按照指定的文件大小数值 n 检索符合条件的文件。其中，n 通常表示以 512 个字节数据块为单位的文件大小。如果 n 后面附加一个字符 c、w、b、k、M 或 G，则分别表示指定的文件大小以字节、双字节的字、512 字节的数据块（默认值）、1KB、1MB 或 1GB 为计数单位。其中： <input type="checkbox"/> +n 表示大于指定的数量； <input type="checkbox"/> n 表示恰好等于指定的数量； <input type="checkbox"/> -n 表示小于指定的数量
<code>-inum n</code>	检索匹配指定信息节点号的文件
<code>-type filetype</code>	检索指定类型的文件。其中的文件类型可以是： <input type="checkbox"/> f 表示普通文件； <input type="checkbox"/> d 表示目录； <input type="checkbox"/> b 表示块特殊文件； <input type="checkbox"/> c 表示字符特殊文件； <input type="checkbox"/> p 表示管道（FIFO）文件； <input type="checkbox"/> l 表示符号链接文件； <input type="checkbox"/> s 表示套接字文件
<code>-perm [±]mode</code>	检索匹配指定访问权限（以八进制数值或符号形式表示）的文件。如果 mode 字段之前存在一个减号“-”，表示文件的访问权限必须包括 mode 定义的所有访问权限。如果 mode 字段之前为加号“+”，表示文件的访问权限至少必须包括 mode 定义的一种访问权限。如果 mode 字段之前没有加减号，表示文件的访问权限必须完全匹配 mode 定义的所有访问权限
<code>-perm /mode</code>	其功能类似于 “-perm +mode”，用于检索匹配指定访问权限（以八进制数值或符号形式表示）的文件
<code>-links [±]n</code>	检索其链接计数大于、等于或小于指定数量 n 的文件
<code>-exec cmd {} \;   +</code>	把 find 命令的检索结果作为参数提交给定的命令，由给定的命令作进一步的加工处理。后面的花括号表示给定命令的参数将由 find 命令的输出结果予以替换。命令的后面必须以转义的分号 “\;” 或转义的加号 “+” 结束。 当命令以加号结束时，意味着把 find 命令的输出结果汇总为一个参数集合，然后一次性地提交给定的命令。因此，使用加号 “+” 而非分号 “;” 能够改善命令的运行性能
<code>-ok cmd;</code>	其功能类似于 “-exec” 选项，唯一的差别是在执行给定的命令之前输出请求信息，当且仅当用户输入 “y”（或 “Y”）确认之后才继续执行
<code>-empty</code>	文件为空，且为普通文件或目录
<code>-ls</code>	以 “ls -dils” 命令的输出格式输出匹配的文件，文件的大小以 1K 字节的数据块为单位，除非环境变量 POSIXLY_CORRECT 已经设置（表示按 512 字节的数据块计算文件的大小）
<code>-print</code>	打印检索结果，即输出符合检索条件的文件名



### 5.14.1 简单检索

#### 1. 按文件名匹配模式检索文件

假定想要检索当前工作目录及其子目录下所有以“.c”为后缀的 C 程序文件，可输入下列命令。

```
$ find . -name '*.c' -print  
./src/atmmon.c  
./src/scanner.c  
....  
$
```

上述 find 命令中的句点“.”表示当前目录。这意味着从当前目录开始，遍历当前目录及其子目录，检索匹配“-name”选项（“\*.c”）指定的所有 C 源程序文件。

注意，当文件名匹配模式中包含通配符时，一定要用单引号或双引号括起来，以便 Shell 能够正确地解释。

#### 2. 按文件创建日期检索文件

假定想要在/home/gqxing 目录及其子目录中找出在创建了某个标志文件 (myfile) 之后修改过的所有文件，可以使用下列命令。

```
$ find /home/gqxing -newer myfile
```

### 5.14.2 使用逻辑运算符

find 命令允许用户使用逻辑非“!”、逻辑与“-a”和逻辑或“-o”等逻辑运算符组合各种选项，定义更为严格的检索准则。在使用逻辑表达式时，组合选项前后要加一对转义的圆括号。

如果想要检索不符合某个特定属性选项定义的文件，可以使用逻辑非运算符“!”。例如，假定打算找出/etc 目录中所有不属于用户 root 的文件，可以使用下列命令。

```
$ find /etc \(! -user root \)  
....  
$
```

如果想要检索同时具有两种不同属性的文件，可使用逻辑与运算符“-a”定义组合选项。例如，假定要找出根目录中属于 gqxing 用户的所有子目录，可以使用下列命令。

```
$ find / \(! -type d -a -user gqxing \)  
....  
$
```

如果想要检索具有其中的一种或两种属性的文件，可以使用逻辑或运算符“-o”定义组合选项。例如，假定期望检索系统中一个月来从未被访问过的、文件扩展名为“.o”或文件名为 a.out 的所有文件，可以使用下列命令。

```
$ find / \(! -name '*.o' -o -name a.out \) -atime +30  
....  
$
```

### 5.14.3 利用 find 命令本身实现其他处理功能

利用 find 命令的“-exec”选项，采用下列两种命令形式，还能够以批处理的方式，把检索出来的文件作为参数，交由其他命令做进一步的加工处理。

```
-exec command {} \;  
-exec command {} \+
```

其中，command 可为任何文件处理命令，花括号表示其参数取自 find 命令的输出，即由 find 命令检索出来的文件名予以替换。注意，“-exec”选项的后面必须附加转义的分号“;”或转义

的加号 “\+” ，作为命令的终止符。

例如，如果想要删除当前目录及其子目录中扩展名为 “.tmp” 的所有文件，可以使用下列命令。

```
$ find . -name '*.tmp' -exec rm {} \;
```

假定因为某种原因，使得一个目录（及其各级子目录）中所有文件的文件属主或访问权限发生了变化，导致应用无法正常运行时，可以利用 find 命令列出所有的文件，由 chown 或 chmod 等命令予以修正，例如下列命令所示。

```
$ find . -type f -exec chown gqxing '{}' \;
```

#### 5.14.4 利用管道实现其他处理功能

通常，Linux 命令能够接受的参数具有一定的限量。虽然这个限量一般情况下并不妨碍我们执行任何 Linux 命令，但当使用上述 find 命令形式进行批处理时，find 命令的输出有时是很可观的，“-exec” 选项后面的命令可能无法接受如此多的参数。为了解决这个问题，一种常见的替代做法是利用管道，将 find 命令的输出提交给 xargs 命令协助执行。

例如，要把某个目录下所有普通文件的访问权限改为 644，所有子目录的访问权限改为 755，可以使用下列两个命令实现。

```
find pathname -type f -print | xargs chmod 644
find pathname -type d -print | xargs chmod 755
```

把 find 命令的输出通过管道提交给 xargs，由 xargs 命令协助执行，除了上述原因之外，还有性能方面的考虑。当把大量的文件和目录交由 “-exec” 选项后面的单个命令处理时，对每个文件或目录的处理都需要创建一个进程，因而需要创建太多的进程。而 Linux 系统允许每个用户创建进程的数量是有一定限制的。xargs 的做法是把文件和目录名收集在一起，组成多个参数提交给单个 Linux 命令执行。这种处理方式只需创建较少的进程，因而能够提高命令的运行速度，改善系统的性能。

但是，当 find 命令的输出并没有多到无法接受时，如果在 find 命令的 “-exec” 选项后面直接采用转义的加号 “\+” ，在提高运行速度，改善系统性能方面，也能达到同样的目的，两者的效果基本上是一样的。

## 5.15 检索文件内容

本节主要讨论功能强大的文本检索工具 grep。

#### 5.15.1 利用 grep 检索文件内容

要检索文件中的特定字符串，可以使用 grep 命令，其基本语法格式简写如下。

```
grep [-inv] string file
```

其中，“-i” 选项表示在进行比较时忽略字母的大小写，“-n” 选项表示在输出的检索结果之前给出文本行在文件中的行号（行号从 1 开始计算），“-v” 表示检索不包含给定字符串或模式的所有文本行。string 是一个检索模式。检索模式可以是一个准备检索的字符串、一个单词或短语。注意，检索模式可以包含空格、标点符号甚至控制字符，但需要在其前后增加引号。file 是准备检索的文件。

例如，要从电话号簿文件 Phonebooks 中检索 John Smith 的电话分机，可以使用部分或完整的



名字进行模式匹配，示例如下。

```
$ grep Smith Phonebooks  
John Smith      2810  
$
```

如果检索模式是一个较长的字符串，由多个字组成，中间也可能包含空格字符，可以在字符串前后加单引号或双引号，如下所示。

```
$ grep "Louisa May" Phonebooks  
Louisa May Alcott  2826  
$
```

检索模式越简短，输出的冗余数据就越多。反之，检索模式的限制越多，符合检索条件的输出结果就越少，如下所示。

```
$ grep Al Phonebooks  
Louisa May Alcott  2826  
David Allan        2866  
Edgar Allan Poe   2812  
$ grep Allan Phonebooks  
David Allan        2866  
Edgar Allan Poe   2812  
$
```

默认情况下，grep 命令是严格区分大小写字母的。也就是说，在输入检索模式的字符串时必须注意字母的大小写，例如下列命令所示。

```
$ grep allan Phonebooks  
$ grep Allan Phonebooks  
David Allan        2866  
Edgar Allan Poe   2812  
$
```

上述的第一个 grep 命令之所以检索失败（没有给出检索结果），就是因为字母“a”的大小写拼写错误。

## 5.15.2 过滤其他命令的输出数据

grep 命令的主要用途是从输入文件中获取感兴趣的数据，过滤掉不需要的数据内容。因此，通常把 grep 称作滤通程序或过滤程序。grep 命令的常见用法是过滤其他命令的输出结果，从命令输出中抽取含有某种特征的数据。因此，必须采用管道机制，把命令的输出数据提交给 grep 命令。

假定当前目录中存在一系列不同时间开发的 C 程序，如下所示。

```
$ ls -l *.c  
-rw-r--r-- 1 gqxing gqxing 833233 2008-09-10 16:22 buttons.c  
-rw-r--r-- 1 gqxing gqxing 739245 2008-07-20 09:38 changes.c  
-rw-r--r-- 1 gqxing gqxing 608368 2008-09-10 10:20 clock.c  
-rw-r--r-- 1 gqxing gqxing 827114 2008-07-20 16:49 commands.c  
....  
$
```

如果想要从中找出 7 月份开发的程序，可以使用下列命令组合，通过管道把 ls 命令的输出提交 grep，过滤后的输出结果如下。

```
$ ls -l *.c | grep 07-  
-rw-r--r-- 1 gqxing gqxing 739245 2008-07-20 09:38 changes.c  
-rw-r--r-- 1 gqxing gqxing 827114 2008-07-20 16:49 commands.c  
$
```

## 5.15.3 使用 grep 检索多个文件

grep 命令可以同时检索多个文件。当找出匹配检索模式的字符串时，grep 将会在输出信息前冠以文件的名字，然后输出匹配检索模式的文本行，如下所示。

```
$ ls
beijing newyork shanghai washington
$ grep capital *
beijing:Beijing is the capital of China.
washington:Washington is the capital of the United States.
$
```

### 5.15.4 检索不包含特定字符串的文本行

为了输出并不包含特定字符串的所有文本行，可以使用 grep 命令的“-v”选项。下面的例子说明了怎样在当前目录下的所有文件中找出不包含字符串 capital 的文本行。

```
$ grep -v capital *
newyork>New York is the biggest city in the United States.
shanghai:Shanghai is the biggest city in China.
$
```

### 5.15.5 在 grep 中使用正则表达式

在 grep 命令中，也可以使用正则表达式作为检索模式，检索匹配给定模式的字符串或文本行。正则表达式可以由普通字符、数字以及具有特殊意义的元字符组成。在 grep 的正则表达式中，可用的元字符包括“^”、“\$”、“.”、“\*”和“\”等（如表 5-4 所示）。注意，grep 命令仅支持简单的正则表达式，要实现复杂的模式匹配，可以使用 egrep 等命令。

表 5-4

grep 检索模式可用的元字符

元字符	匹 配
^	文本行的行首
\$	文本行的行尾
.	任何一个单字符
[...]	字符集或字符范围中的任何一个字符
[^...]	不属于字符集或字符范围中的任何一个字符
*	零个或多个同一字符或正则表达式
+	一个或多个同一字符或正则表达式
\	随后的元字符作为普通字符处理

表 5-4 中的特殊字符对 Linux 系统也有特殊的意义。因此，在 grep 命令中使用正则表达式时，需要通过转义机制，使系统在解释命令行期间忽略这些元字符的特殊含义。因此，当用户在系统提示符下输入带有正则表达式的 grep 命令时，应当用引号括住正则表达式。

此外，如果检索模式中包含元字符，且需要忽略其特殊意义时，可在元字符之前增加转义符号。

现在，我们将以下列 5 行数据（选自泰戈尔《Stray Birds》的两首诗）作为本小节的测试数据，说明如何在 grep 命令中使用正则表达式。

```
$ cat stray.birds
I cannot choose the best
The best choose me

Roots are the branches down in the earth
Branches are the roots in the air
$
```



由于元字符“^”表示行首，故下列命令将会找出输入文件中以字符“T”为行首字符的所有文本行。

```
$ grep '^T' stray.birds  
The best choose me  
$
```

若要仅仅列出当前目录中的子目录，可以使用下列命令。

```
$ ls -l | grep '^d'  
drwxr-xr-x 2 gqxing gqxing 4096 2008-09-10 22:45 bin  
drwxr-xr-x 2 gqxing gqxing 4096 2008-09-10 22:45 incl  
drwxr-xr-x 2 gqxing gqxing 4096 2008-09-10 22:45 script  
drwxr-xr-x 2 gqxing gqxing 4096 2008-09-10 22:45 src  
$
```

元字符“\$”表示行尾，故下列命令将会找出输入文件中所有以字符“t”为行尾字符的文本行。

```
$ grep 't$' stray.birds  
I cannot choose the best  
$
```

下列命令可用于显示文件中只有一个字符“b”的文本行（输入文件中没有这样的文本行，故没有任何输出）。

```
$ grep '^b$' stray.birds  
$
```

由于元字符“.”可以匹配任何一个字符，故下列命令能够匹配任何以“an”为头两个字符的3字符字符串，包括“any”、“and”、“cannot”和“plan”（因为空格也计数）等。

```
$ grep 'an.' stray.birds  
I cannot choose the best  
Roots are the branches down in the earth  
Branches are the roots in the air  
$
```

要列出当前目录中其他用户能够读写的文件，可以使用下列命令。

```
$ ls -l | grep '^.....rw'  
-rw-rw-rw- 1 gqxing gqxing 120 2008-09-10 22:32 stray.bird  
$
```

要找出文件中包含字符串“the”的文本行，且忽略大小写字母的区别，并在输出数据前冠以行号，可以使用下列命令。

```
$ grep -i -n the stray.birds  
1:I cannot choose the best  
2:The best choose me  
4:Roots are the branches down in the earth  
5:Branches are the roots in the air  
$
```

要找出文件中的所有空行，并给出行号，可以使用下列任何一个命令。

```
$ grep -n ^$ stray.birds  
3:  
$ grep -n -v . stray.birds  
3:  
$
```

当检索模式（如字符、字符串或正则表达式）后面附加一个星号 \* 字符时，grep 将把星号“\*”解释为“重复匹配零个或多个模式”。

按照上述定义，检索结果也可以包含零个模式，这有可能使用户对星号的使用及输出结果感到困惑。下列命令或许可以稍作阐释，其中的检索模式“ro\*”意味着找出字符串中含有一个字符“r”和零个或多个字符“o”（如“r”、“ro”或“roo”等）的所有文本行。

```
$ grep 'ro*' list  
Roots are the branches down in the earth
```

```
Branches are the roots in the air
$
```

在上述输出结果中，第一行匹配检索模式“ro\*”的字符串是“branches”，第二行匹配检索模式“ro\*”的字符串是“roots”。

同样，如果想要找出字符串中至少包含一个字符“d”的所有文本行，可以使用下列检索模式（其中第一个字符“d”仅表示字符文字“d”，第二个字符“d”和“\*”是一个正则表达式，表示零个或多个字符“d”）。

```
$ grep 'dd*' stray.birds
Roots are the branches down in the earth
$
```

但如果想要找出字符串中至少包含两个字符“oo”的所有文本行，可以使用下列模式检索（其中的前两个字符“o”仅表示字符串“oo”，第3个字符“o”和“\*”是一个正则表达式，表示零个或多个字符“o”）。

```
$ grep 'ooo*' stray.bird
I cannot choose the best
The best choose me
Roots are the branches down in the earth
Branches are the roots in the air
$
```

若要检索输入文件中匹配零个或多个任意字符的所有文本行，可以使用下列检索模式。

```
$ grep '.*' stray.bird
I cannot choose the best
The best choose me

Roots are the branches down in the earth
Branches are the roots in the air
$
```

由于句点“.”能够匹配任意一个字符，星号“\*”能够重复匹配零个或多个模式，故“.\*”能够匹配任何文本行，包括空行。

## 5.15.6 检索元字符本身

若要使用 grep 命令检索“&”、“!”、“.”、“\*”、“?”和“\”等元字符本身，可以在元字符前面加转义符号“\”。转义符号使 grep 命令能够忽略元字符的特殊含义。例如，下列检索模式可以返回以句点“.”为起始字符的文本行。这在检索经过 nroff 或 troff 加工处理的文档时是非常有用的。

```
$ grep '^\. somefile
```

## 5.15.7 在命令行中使用引号

正如先前所述，如果想把具有多个单词的短语作为一个字解释，可以使用引号把短语括起来。例如，要从输入文件中检索短语“in the air”，可以使用下列 grep 命令。

```
$ grep "in the air" stray.bird
Branches are the roots in the air
$
```

此外，还可以使用单引号把多字短语组合为一个检索单位。单引号的另外一个作用是能够确保系统按文字解释一定的元字符，如“\$”符号等。

注意，即使使用引号，命令历史机制中使用的元字符“!”总是作为元字符解释的，除非在前



面加转义符号“\”。推而广之，如果想要让 grep 命令按照普通字符解释“&”、“!”、“\$”、“?”、“.”、“;”和“\”等元字符，必须在每个元字符前面加转义符号“\”。

例如，如果输入下列命令，将会显示 `stray.bird` 文件中的所有文本行。

```
$ grep \$ stray.bird  
I cannot choose the best  
The best choose me  
  
Roots are the branches down in the earth  
Branches are the roots in the air  
$
```

然而，如果输入下列命令，则只会显示包含“\$”字符的文本行。

```
$ grep '\$' stray.bird  
$
```

有关 grep 命令的更多内容，可参考系统提供的联机文档。

## 5.16 排序

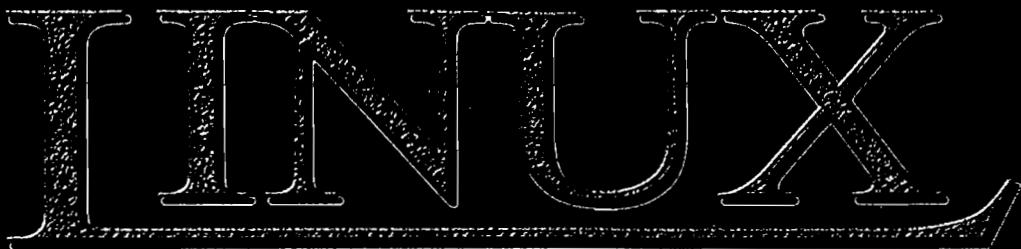
排序是一种需要经常用到的工具。sort 命令可对输入数据或文件内容进行排序，按照一定的顺序逐行显示，其语法格式简写如下。

```
sort [-bdfimnr] -k key -t sepchar -o output [file]
```

其中，“-n”选项表示按照字符串的数值而不是文字进行排序，“-r”选项表示按照从大到小或反向字符顺序排序，“-k”选项表示关键字的字段位置，或者关键字字段的起止字符位置或范围，“-o”选项用于指定存储排序结果的输出文件（默认值为标准输出），“-t”选项用于指定除空白字符之外的其他字段分隔符，“-b”选项表示忽略前置的空白字符，“-d”选项表示仅考虑字母数字和空格字符，按字典顺序排序。

例如，“ls -l”命令的输出通常是按文件名的字符顺序输出文件列表的，若要按照文件的大小，从大到小排序，可以观察“ls -l”命令的输出，确定文件大小字段的位置（第 5 列），然后利用“-k”、“-r”（表示从大到小排序）和“-n”（表示按数值的大小排序）选项进行排序，示例如下。

```
$ cd /var/log  
$ ls -l message*  
-rw-r----- 1 syslog adm 97397 2008-09-13 15:29 messages  
-rw-r----- 1 syslog adm 194564 2008-09-10 17:35 messages.0  
-rw-r----- 1 syslog adm 33290 2008-09-03 21:38 messages.1.gz  
-rw-r----- 1 syslog adm 18869 2008-08-27 12:30 messages.2.gz  
-rw-r----- 1 syslog adm 16154 2008-08-19 20:21 messages.3.gz  
$ ls -l /var/log/message* | sort -k5 -rn  
-rw-r----- 1 syslog adm 194564 2008-09-10 17:35 messages.0  
-rw-r----- 1 syslog adm 97397 2008-09-13 15:29 messages  
-rw-r----- 1 syslog adm 33290 2008-09-03 21:38 messages.1.gz  
-rw-r----- 1 syslog adm 18869 2008-08-27 12:30 messages.2.gz  
-rw-r----- 1 syslog adm 16154 2008-08-19 20:21 messages.3.gz  
$
```



## 第6章

### 编辑文件

在 UNIX 系统 vi 编辑器的基础上, Linux 系统提供了克隆版的 vim 编辑器(参见 <http://www.vim.org>)等。vim 编辑器是对 vi 的扩充与增强, 提供了许多附加的功能特性, 且与 vi 几乎是完全兼容的。vim 可以运行在许多平台上, 包括 Windows、Macintosh、UNIX 和 Linux 系统。利用 vim, 可以编辑文件, 开发应用程序。本章讨论的主要内容包括:

- 启动 vim 编辑器;
- vim 编辑器的两种工作模式;
- 保存编辑的文件并退出 vim;
- vim 编辑器的基本命令;
- 使用 ex 命令;
- 检索与替换;
- 编辑多个文件;
- 定制 vim 编辑器的运行环境;
- 其他特殊说明;
- vim 编辑器命令总结。





vim 是一个极其强有力的文本编辑工具，如果能够掌握其规律，它将是一个非常好的开发工具。但在使用 vim 编辑器时，用户需要记住编辑器当前所处的工作模式。即便如此，只要经过一定的实践，很快就会熟练地掌握 vim，得心应手地处理文本文件。vim 提供了大量的命令供用户编辑文件。本章将按照操作类型，介绍最基本的 vim 命令。

## 6.1 启动 vim 编辑器

### 6.1.1 创建文件

在 Linux 系统的命令提示符下输入下列命令，即可启动 vim 编辑器。

```
$ vim myfile
```

如果名为 myfile 的文件存在，上述命令将会打开指定的文件，同时在编辑窗口中显示文件第一页的数据内容。如果指定的文件不存在，vim 将会打开一个新文件，出现图 6-1 所示的编辑窗口。

此时，光标将处于编辑窗口左上角的位置，屏幕左边的波浪符“~”表示空行，说明这是一个空文件。注意，启动 vim 时可以同时指定多个文件名参数，意味着同时编辑多个文件；也可以不指定文件名，等到完成文件编辑之后再使用“w”命令写入一个新文件，然后退出 vim。

如果运行 vim 命令时未指定文件的名字，将会显示图 6-2 所示的内容。此时，可以直接编写文件，等完成之后再使用“:w filename”命令保存写就的文件，然后使用“:q”命令退出 vim。初学者也可以使用“:help”命令获取 vim 的帮助信息，如图 6-3 所示。

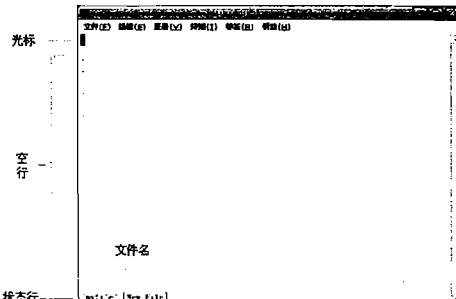


图 6-1 vim 编辑器

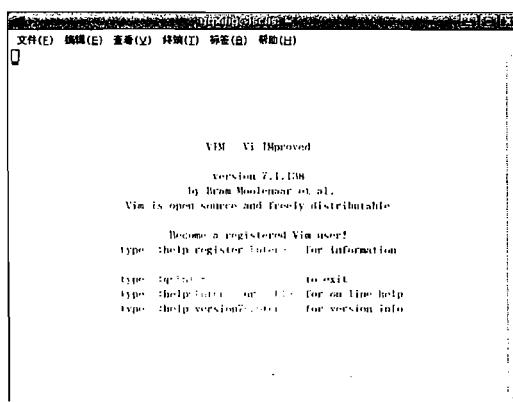


图 6-2 未指定文件名字的 vim 编辑器

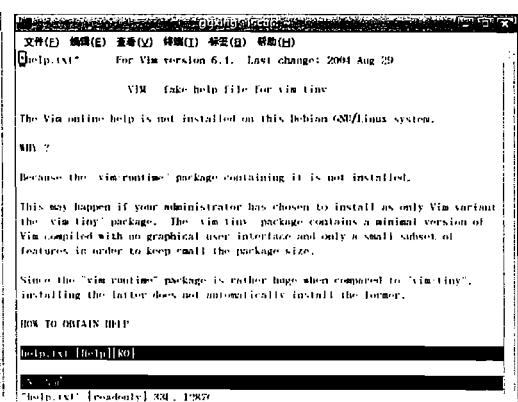


图 6-3 vim 的帮助信息

## 6.1.2 状态行

编辑窗口的最后一行是 vim 的状态行，用于显示编辑器的状态、编辑过程中出现的出错信息、光标所在的行列位置、删除或复制的行数等。初始启动时，状态行将会显示文件的名字、文件的行数以及文件的字节计数。在图 6-1 中，状态行表示打开的是一个新（空）文件。

# 6.2

## vim 编辑器的两种工作模式

vim 分为命令和输入两种工作模式。任何时刻，vim 编辑器总是处于命令和输入两种工作模式之一。当启动 vim 编辑器、打开或创建一个文件时，vim 即处于命令模式。通过发布 vim 命令，可使 vim 处于输入模式。在输入模式下，可以输入数据，编写自己的应用程序；而在命令模式下，可以输入 vim 命令，执行特定的 vim 编辑功能。命令模式是 vim 的默认工作模式。通过使用 vim 命令和 Esc 键，可以在两种工作模式之间相互转换。

在某些情况下，vim 并不提示编辑器当前所处的工作模式，因此对于 vim 的新用户来讲，区分命令模式与输入模式也许是最感困惑或茫然的问题。但只要记住一点，就可以做到心中有数。即无论何时，只要按下 Esc 键，不管 vim 当前处于何种工作模式，总是进入命令模式。因此，多敲几次 Esc 键是 vim 用户的常见动作。

第一次使用 vim 打开文件时，vim 总是处于命令模式。在能够输入任何文本之前，首先必须输入 vim 的数据输入命令。例如，输入“i”（“插入”）字符命令，即可在当前光标所处字符位置之前插入数据；输入“a”（“附加”）字符命令，即可在当前光标所处字符位置之后附加数据。本章的后面将详细介绍各种数据输入命令。

无论何时需要返回命令模式，按下 Esc 键即可。如果不能确定 vim 当前处于何种工作模式，只要按下 Esc 键，即可确保 vim 总是处于命令模式，然后再决定下一步怎么做。如果在 vim 处于命令模式时按下 Esc 键，或者按下其他不合法的键时，终端将会发出鸣叫或发生屏幕闪烁现象，但不会影响正在编辑的文件。

### 6.2.1 输入模式

为了在前面打开的 myfile 示例文件中输入数据，可输入 vim 的“插入”命令“i”。这一命令将使 vim 编辑器从命令模式转入输入模式。

现在，即可尝试输入几行数据，在每行数据输入结束后按下 Enter 键。在输入数据的过程中，可以利用退格键做简单的校正（在按下 Enter 键之前）。在整个输入过程结束之后，按下 Esc 键，即可返回命令模式。此时，光标将处于刚才输入的最后一个字符的位置。然后，还可以利用各种 vim 命令，对输入的数据进行校正。

### 6.2.2 命令模式

如上所述，当利用 vim 打开一个文件时，vim 将处于命令模式。在命令模式下，可以输入各种 vim 命令，以便完成各种编辑功能。vim 命令几乎都是由一个字符、两个字符或一个选用的数



字加字符组成的。通常，同一字母但大小写不同的字符命令，其意义是相同的，但其作用则完全不同。例如，字符命令“a”意味着在当前光标所处字符位置之后附加数据，而“A”则意味着在当前光标所在行的行尾附加数据。

大多数 vim 命令不需要按 Enter 键即可立即执行，但是，以冒号“:”开始的命令需要在输入命令之后再按 Enter 键。在命令模式下输入冒号“:”时，“:”将会出现在编辑窗口底部最后一行的左下角，然后即可接着输入编辑命令。

以冒号开始的命令实际上是 ex 命令。ex 与 vim 命令是同一编辑程序的两个不同的用户界面，vim 提供面向屏幕的用户界面，而 ex 则提供面向命令行的用户界面。所有的 ex 命令均可在 vim 中使用。在输入冒号之后，实际上已经切换到面向命令行的 ex 用户界面。这种切换方式使用户能够在不离开 vim 的情况下执行许多文件编辑命令，甚至可以执行其他 Shell 命令（参见 6.5 节）。

## 6.3 保存编辑的文件并退出 vim

在使用 vim 编辑文件期间，用户所做的任何编辑处理并未直接反映到实际的文件中。实际上，整个编辑过程将被保存到 vim 于内存中临时创建的一个文件副本中。仅当发布“w”（“写”）等命令时，内存缓冲区中的内容才能永久性地被保存到磁盘上的文件中。

vim 编辑器的这种处理方式既有积极的一面，也有丢失数据之忧。可取之处是在退出文件编辑时，用户可以放弃编辑期间所做的任何修改，而不影响原有的文件。缺点是当系统发生故障时，很有可能丢失内存缓冲区中保存的数据。

因此，最好的做法是注意随时保存数据，特别是当编辑重要的文件时。

vim 编辑器提供了许多命令，用于把内存缓冲区中的数据内容保存到磁盘文件中，然后退出 vim 编辑器。这些命令允许用户选择“保存并退出”、“强制退出而不保存”等编辑器退出方式（参见表 6-1）。

表 6-1

vim 的保存文件和退出命令

命 令	简 单 说 明
:w	保存编辑后的文件内容，但不退出 vim 编辑器。这个命令的作用是把内存缓冲区中的数据写到启动 vim 时指定的文件中
:w!	强制写文件，即强制覆盖原有的文件。如果原有文件的访问权限不允许写入文件，例如，原有的文件为只读文件，则可使用这个命令强制写入。但是，这种命令用法仅当用户是文件的属主时才适用，而超级用户则不受此限制
:wq	保存文件内容后退出 vim 编辑器。这个命令的作用是把内存缓冲区中的数据写到启动 vim 时指定的文件中，然后退出 vim 编辑器。另外一种替代的方法是用 ZZ 命令
:wq!	强制保存文件内容后退出 vim 编辑器。这个命令的作用是把内存缓冲区中的数据强制写到启动 vim 时指定的文件中，然后退出 vim 编辑器
ZZ	使用 ZZ 命令时，如果文件已经做过编辑处理，则把内存缓冲区中的数据写到启动 vim 时指定的文件中，然后退出 vim 编辑器。否则只是退出 vim 而已。注意，ZZ 命令前面无需加冒号“:”，也无需按 Enter 键
:q	在未做任何编辑处理而准备退出 vim 时，可以使用此命令。如果已做过编辑处理，则 vim 不允许用户使用“:q”命令退出，同时还会输出下列警告信息： No write since last change (:quit! overrides)
:q!	强制退出 vim 编辑器，放弃编辑处理的结果。如果确实不需要保存修改后的文件内容，可输入“:q!”命令，强行退出 vim 编辑器
:w filename	把编辑处理后的结果写到指定的文件中保存
:w! filename	把编辑处理后的结果强制保存到指定的文件中，如果文件已经存在，则覆盖现有的文件
:wq! filename	把编辑处理后的结果强制保存到指定的文件中，如果文件已经存在，则覆盖现有的文件，并退出 vim 编辑器

## 6.4 vim 编辑器的基本命令

本节将分类介绍 vim 提供的各种编辑命令：

- 移动光标位置；
- 输入文本数据；
- 修改和替换文本；
- 撤销先前执行的文本编辑命令；
- 删除文本；
- 重复执行先前的命令。

注意，vim 命令是严格区分大小写字母的。即使命令形式完全相同，如果不注意区分大小写字母，将会产生完全不同的效果。

### 6.4.1 移动光标位置

当启动 vim 编辑器时，光标将处于编辑窗口左上角的位置，并处于命令模式。在命令模式下，可以使用表 6-2 所示的字符命令移动光标位置。

表 6-2 光标移动命令

命 令	简 单 说 明
←↑↓→	方向箭头键。可在编辑窗口上将光标左移、上移、下移和右移一个字符（行）位置
h k j l	其功能与箭头键完全相同。在无法使用箭头键（如远程访问）时，可以使用这 4 个键分别替代相应的箭头键
-	把光标移至上一行的第一个起始字符位置（第一个非空白字符位置）
Enter 键	把光标移至下一行的第一个起始字符位置（第一个非空白字符位置）
退格键	光标左移一个字符位置
空格键	光标右移一个字符位置
Ctrl-F	往下（文件结尾方向）滚动一屏。在命令方式下按 Ctrl-F 键，编辑窗口中将会显示文件下一页的内容，光标也同时移至下一页的左上角位置
Ctrl-B	往上（文件开始方向）滚动一屏。在命令方式下按 Ctrl-B 键，编辑窗口中将会显示文件前一页的内容，光标也同时移至前一页的左下角位置
Ctrl-U	往下滚动半屏
Ctrl-D	往上滚动半屏
Ctrl-E	编辑窗口中的文件内容整体上移一行
Ctrl-Y	编辑窗口中的文件内容整体下移一行
H	把光标移至编辑窗口顶部第一行的起始字符位置（第一个非空白字符位置）
M	把光标移至编辑窗口中间一行的起始字符位置（第一个非空白字符位置）
L	把光标移至编辑窗口底部最后一行的起始字符位置（第一个非空白字符位置）
w	光标右移一个字。如果相邻的两个字之间有标点符号，光标将移至标点符号位置



续表

命 令	简 单 说 明
w	光标右移一个字。即使相邻的两个字之间有标点符号，也忽略之
b	光标左移一个字。如果相邻的两个字之间有标点符号，光标将移至标点符号位置
B	光标左移一个字。即使相邻的两个字之间有标点符号，也忽略之
e	把光标移至当前字（或下一个字）的最后一个字符位置
E	同上，只是以空格字符作为字的分隔符
^	把光标移至当前行的起始位置，即当前行的第一个非空白字符位置
0 (零)	同上
\$	把光标移至当前行的行尾，即当前行的最后一个字符位置
[n]G	将光标移至指定行的行首位置。其中 n 表示行号，默认情况下转至文件的最后一行。因此，要转到当前文件的最后一行，只需输入“G”命令。要移至文件的第一行，可输入“1G”命令。实际上，利用“nG”命令，可以转至当前文件的任何一行。例如，要修改文件的第 60 行，输入“60G”命令，即可立即跳转到第 60 行的行首位置
(	把光标移至一个完整句子的句首位置
)	把光标移至一个完整句子的句尾位置
{	把光标移至一个完整段落的段首位置
}	把光标移至一个完整段落的段尾位置

## 6.4.2 输入文本

vim 提供了许多命令，使用户能够输入文本数据。表 6-3 给出了 vim 编辑器中最常用的几种数据输入命令。注意，这些命令将使 vim 进入数据输入方式，同时在编辑窗口的左下角（即状态行左边）显示“— INSERT —”状态信息，表示 vim 当前正处于输入模式。为了使用这些命令，首先必须确保 vim 处于命令模式（多按几次 Esc 键总能保证 vim 处于命令模式）。

表 6-3

数据输入命令

命 令	简 单 说 明
a	使用“a”命令，可在光标当前所在字符位置之后输入数据，输入结束后可按 Esc 键退出输入方式。在发布“a”命令之前，光标可移至文本行的任何位置。发布“a”命令之后，输入的数据多少不限
A	使用“A”命令，可在光标当前所在行的行尾（即在最后一个字符之后）输入数据，输入结束后可按 Esc 键返回命令模式。在发布“A”命令之后，不管光标处于何位置，都会移至当前文本行的行尾，此时可输入任何数量的数据，直至按下 Esc 键
i	使用“i”命令，可在光标当前所在字符位置之前输入数据，输入数据的数量不限。输入结束后可按 Esc 键退出输入方式
I	使用“I”命令，可在光标当前所在行的行首（即在第一个非空白的起始字符之前）输入数据，不管光标之前处于何位置，都会移至当前文本行的行首，输入数据的数量不限，输入结束后可按 Esc 键退出输入方式
o	使用“o”命令，可在光标当前所在行之后插入数据，行数不限，直至按下 Esc 键结束
O	使用“O”命令，可在光标当前所在行之前插入数据，行数不限，直至按下 Esc 键结束

## 6.4.3 修改与替换文本

为了修改或替换文本数据，vim 提供了若干不同的命令，用于校正文本数据。用户可以根据

具体情况，灵活地予以选用（参见表 6-4）。在表 6-4 所列的编辑命令中，除了“r”和“~”命令不显示任何信息，“R”命令将会在编辑窗口的左下角显示“— REPLACE —”状态信息，其他命令均显示“— INSERT —”状态信息，表示 vim 当前正处于输入模式。

表 6-4

修改与替换命令

命 令	简 单 说 明
C	替换从光标位置开始直至行尾的所有数据内容，以 Esc 键结束
cw	替换单个字。要替换一个整字，可将光标移至准备替换的字的字首位置，然后输入“cw”命令，接着输入新的文字。输入的字符数量不限，输入结束后可按 Esc 键返回命令模式。要修改单字的一部分，可将光标移至单字保留部分右边的字符位置，输入“cw”命令后可对该字进行校正，结束后按 Esc 键返回命令模式
[n]cc	替换成行。要替换一行文本，只需把光标移至目标行的任何字符位置，然后输入“cc”命令。此时，当前文本行将会消失，留下一个空行位置，等待用户输入新的文本数据。此时可以输入任何数量或行数的数据。输入结束后，按 Esc 键可返回命令模式。要替换多行文本，可把光标移至目标行的第一行，然后输入“ncc”命令，其中 n 表示需要替换的文本行的数量
[n]s	替换字符。要替换光标当前所在位置的一个字符，可输入“s”命令，然后可以输入任何数量或行数的数据。输入结束后，按 Esc 键可返回命令模式。要替换从光标位置开始的多个字符，可输入“ns”命令，其中的 n 表示需要替换的字符的数量
S	替换当前行。要替换光标当前所在的行，可输入“S”命令，然后可以输入任何数量或行数的数据。输入结束后，按 Esc 键可返回命令模式
r	替换单个字符。使用“r”命令，可用随后输入的字符替换光标位置的单个字符。与“s”命令不同，“r”命令只能替换一个字符，替换后，vim 编辑器自动返回命令模式（不需要再按 Esc 键）
R	替换多个字符。使用“R”命令，可以从光标位置开始替换多个字符，数量不限，直至按下 Esc 键结束
[n]~	转换光标当前所在位置字母的大小写。一直按住波浪号“~”键能够连续转换多个字母的大小写。也可以在输入“~”之前输入一个数字，一次转换多个字母的大小写

#### 6.4.4 撤销先前的修改

在编辑文本期间，以及准备把加工后的数据保存到文件之前，有时可能需要放弃某些编辑处理的结果。vim 的撤销命令能够撤销先前执行的编辑命令的处理结果，使 vim 回退到先前的处理状态。然后，可从先前的处理位置开始继续进行编辑加工。撤销命令及其说明如表 6-5 所示。

表 6-5

撤销命令及其说明

命 令	简 单 说 明
u	用于撤销先前执行的编辑命令。如果在编辑过程中出现失误，或者编辑后又改变了决定，可以使用 vim 的“u”命令，撤销先前执行的编辑命令。注意，输入“u”命令后不需要按 Esc 键。连续输入“u”命令后能够以回溯的方式，依次撤销先前执行的所有编辑命令
U	撤销或恢复对当前文本行所做的全部编辑处理。使用 vim 的“U”命令可以撤销或恢复最近一次编辑处理的文本行。也就是说，“U”命令仅适用于最近一次修改的文本行。同样，输入“U”命令后也不需要按 Esc 键

#### 6.4.5 删 除 文 本

利用表 6-6 给出的 vim 命令，可以删除指定的字符、字或文本行数据。



表 6-6

删除命令及其说明

命令	简单说明
[n]x	删除字符。要删除单个字符，可将光标移至准备删除的字符位置，然后输入“x”命令。“x”命令除删除指定的字符之外，还将删除字符占用的空间位置——当从某个单字中间删除一个字符时，余下的字符将合并为一个新的字，中间不会留下任何间隙。利用“x”命令，也可以删除文本行中的空格字符。要删除多个字符，可将光标移至准备删除的字符串起始位置，然后输入“nx”命令，其中的n表示字符的数量
[n]X	删除字符。要删除光标当前所在位置的前一个字符，可以使用大写的“X”命令。要删除多个字符，可将光标移至准备删除的字符串的右边，然后输入“nX”命令，其中的n表示字符的数量
dw	删除单个字或部分字。要删除一个整字，可以把光标移至该字的起始字符位置，然后输入“dw”命令相应的字及其占用的空间位置将一并被删除。要删除单字的右边部分，可将光标移至该字要保留部分的后面，输入“dw”命令，即可删除单字的右边部分
[n]dd	删除文本行。要删除整个文本行，可以把光标移至文本行的任何位置，然后输入“dd”命令，整个文本行及其占用的空间将会一并被删除。要同时删除多个文本行，可将光标移至准备删除的第一个文本行，然后输入“ndd”命令，其中的n表示准备删除的行数（包括当前行在内）
D	删除文本行的行尾部分。要删除文本行右边的部分文字，可将光标移至文本行要保留部分的后面，输入“D”命令，即可删除文本行的行尾部分

## 6.4.6 复制、删除与粘贴文本

许多字处理软件都提供“复制—粘贴”与“剪切—粘贴”的文本行处理方式。vim 编辑器也提供这样的功能。在 vim 编辑器中，与“复制—粘贴”等价的处理过程是先用“yy”命令复制文本行，接着再用“p（或 P）”命令实现文本行的实际复制；与“剪切—粘贴”等价的处理过程是先用“dd”命令删除文本行，接着再用“p（或 P）”命令实现文本行的移动。在上述两种组合方式的基础上，如果在“yy”或“dd”命令之前再输入适当的数字，还可以实现整块文本行的复制和移动处理（参见表 6-7）。

表 6-7

复制与粘贴命令

命 令	简 单 说 明
[n]yy	<p>复制文本行。实际上，“yy”命令只是把文本行的数据内容保存到粘贴板中。一个复制动作需要同时使用两个命令才能完成，即“yy”与“p（或 P）”命令。</p> <p>因此，为了复制一个文本行，可按下列步骤执行：</p> <ol style="list-style-type: none"> <li>(1) 把光标移至准备复制的文本行的任何位置；</li> <li>(2) 输入“yy”命令；</li> <li>(3) 再把光标移至目标行的任何位置；</li> <li>(4) 输入“p”命令，将粘贴板中的数据内容复制到光标所在行的下面；</li> <li>(5) 输入“P”命令，将粘贴板中的数据内容复制到光标所在行的上面。</li> </ol> <p>如果在输入“yy”命令之前输入数字，则可以同时复制多个文本行。例如，如果想要复制 10 行数据，可输入“10yy”命令，这将把从光标当前所在行开始的 10 行数据复制到粘贴板中。此时，vim 编辑器将会在编辑窗口底部显示一条信息“10 lines yanked.”，表示命令已成功地执行</p>
[n]Y	其功能同“yy”命令
[n]dd	<p>删除文本行。为了把一个或若干文本行移至某个位置，需要先删除文本行，然后再粘贴到适当的位置，因此也需要同时使用两个命令才能完成，即“dd”与“p（或 P）”命令。</p> <p>例如，为了移动 5 行数据，可以把光标移至要删除文本行的任何位置，发布“5dd”命令，然后把光标移至插入位置，接着输入“p（或 P）”命令，即可把文本行移至当前行的下方（或上方）。</p>
p（小写）	把粘贴板中的文本数据复制到光标所在行的下面
P（大写）	把粘贴板中的文本数据复制到光标所在行的上面

#### 6.4.7 按指定的数量重复执行命令

许多 vim 命令前面都可以加一个计数值，表明相应的命令需要重复执行的次数。在前面的讨论过程中，实际上已经用到了这样的命令。例如，“3dd”命令意味着删除文本行的动作需要执行 3 次，最终结果是删除 3 行数据，“2dw”命令意味着删除两个字，“4x”命令意味着删除 4 个字符（包括空格）。

另外，也可以使用计数命令方式移动光标。例如，“3w”命令意味右移 3 个字，2Ctrl-F 意味往前滚动两屏。

使用句点“.”命令也可以重复执行先前的文本编辑命令。例如，如果用户刚刚使用“dd”命令删除了一个或多个文本行，此时可以把光标移至准备删除的文本行中，仅仅输入一个句点“.”命令即可重复执行删除文本行的处理动作。对于复杂的编辑命令，句点命令尤其方便。

## 6.5 使用 ex 命令

事实上，在需要处理大块文本行的情况下，与上述的“复制—粘贴”与“剪切—粘贴”处理方式相比，利用 ex 命令还可以实现更精确、更方便的编辑处理。使用 ex 命令时，无需在编辑窗口中计数文本行的数量，然后再寻找插入点。用户只需提供一个准备复制或移动的文本行的范围，然后指定插入点的行号即可（当然，删除时无需指定插入点）。

### 6.5.1 显示行号

使用 ex 命令时，通常需要知道文本行的编号。要在编辑的文件中显示行号，可输入下列命令。

:set nu

按下 Enter 键后，新加的行号将会出现在编辑窗口的左边。注意，这些行号实际上并不存在于文件中，只是为方便用户的编辑处理而出现在编辑窗口中，以增加文本数据的可读性，如图 6-4 所示。

The vi utility is a display-oriented text editor based on an underlying line editor ex. It is possible to use the command mode of ex from within vi and to use the command mode of vi from within ex. The visual commands are described on this manual page; how to set options (like automatically numbering lines and automatically starting a new output line when you type carriage return) and all ex line editor commands are described on the ex(1) manual page.

-

-

-

-

-

-

-

1 The vi utility is a display-oriented text editor  
2 based on an underlying line editor ex. It is possible  
3 to use the command mode of ex from within vi and to  
4 use the command mode of vi from within ex. The visual  
5 commands are described on this manual page; how to set  
6 options (like automatically numbering lines and  
7 automatically starting a new output line when you type  
8 carriage return) and all ex line editor commands are  
9 described on the ex(1) manual page.

图 6-4 显示行号

如果想要关闭行号显示，可输入下列命令。

```
:set nonu
```

另外，要随时了解当前文本行的行号位置，可在命令模式中按下 **Ctrl-G** 键。**vim** 编辑器将会在编辑窗口的底部显示当前行的行号位置，以及当前文件的名字等信息。



## 6.5.2 多行复制

ex 复制命令的基本语法格式如下。

```
:line#1,line#2 co line#3
```

其中，前两个数字（即 line#1 和 line#2，中间以逗号分开）指定需要复制的文本行的范围，第 3 个数字（即 line#3）表示插入点的行号。例如，要把 myfile 文件的第 1~5 行复制到当前文件的第 12 行之后，可以使用下列命令。

```
:1,5 co 12
```

在指定文本行的范围时，可以使用下列缩写形式：

- 句点 “.” 表示当前行，意味着从当前行开始；
- 美元符号 “\$” 表示文件的结尾，即文件的最后一行。

因此，要把当前行直至后续第 5 行复制到第 12 行之后，可以使用下列命令。

```
:.,5 co 12
```

为了把第 6 行直至文件最后一行复制到第 2 行之后，可以使用下列命令。

```
:6,$ co 2
```

## 6.5.3 移动文本行

ex 移动命令的基本语法格式类似于复制命令，如下所示。

```
:line#1,line#2 m line#3
```

在移动文本行时，可以采用与复制文本行相同的方式指定文本行的范围和插入点，包括使用缩写的句点 “.” 和美元符号 “\$”。两者的差别仅在于，“移动” 命令将会把指定范围的文本行从一个位置整块地搬到另一个指定的位置。

例如，为了把第 1~5 行移至第 12 行之后，可以使用下列命令。

```
:1,5 m 12
```

## 6.5.4 删 除 文 本 行

删除文本行时，也可以采用相同的方式指定文本行的范围，包括使用缩写的句点 “.” 和美元符号 “\$”。要删除多个连续的文本行，可以使用下列命令形式实现。

```
:line#1,line#2 d
```

例如，要删除文件中的第 1~5 行，可输入下列命令。

```
:1,5 d
```

# 6.6 检索与替换

vim 提供了若干命令，使得用户能够以检索指定字符串的方式，直接跳转至期望的文件位置。另外，vim 还提供了强有力的全局检索与替换功能。

## 6.6.1 检索字符串

字符串由一个或多个连续的字符组成，它可以包含字母、数字、标点符号、特殊字符、空格、制表符或回车字符。字符串可以是一个语义意义上的单词，也可以是单词的一部分。要检索字符串

串，可以使用如表 6-8 所示的命令。

表 6-8

检索命令

命 令	命 令 描 述
:/str	检索给定的字符串。vim 将从当前光标位置开始检索，当找到指定的字符串时，光标将移至第一个出现的字符串位置。例如，要检索 meta，可以输入 “/meta” 命令，然后按 Enter 键
:?str	从当前位置开始，反向检索给定的字符串
n	从当前位置开始，继续检索下一个匹配的字符串
N	从当前位置开始，反向检索下一个匹配的字符串
/	同 “n” 命令，从光标当前所在字符位置开始，重复执行先前的检索，但在输入 “/” 字符之后还需按 Enter 键
?	同 “N” 命令，从光标当前所在字符位置开始，反向重复执行先前的检索，但在输入 “?” 字符之后还需按 Enter 键
:/pat/+n	将光标移至匹配的字符串 “pat” 所在行之后的第 n 行
:?pat?-n	将光标移至匹配的字符串 “pat” 所在行之前的第 n 行

如果想要检索字符串，可在 “: /” 后面附加准备检索的字符串，然后按 Enter 键，vim 将会从光标当前所在位置开始向下（文件结尾方向）检索。如果发现匹配的字符串，vim 将会把光标移至检索方向第一个出现的字符串位置。

例如，要检索字符串 “meta”，可输入 “:/meta” 命令，然后按 Enter 键。如果接着输入 “n” 命令，可以继续检索下一个匹配的字符串。如果输入大写的 “N” 命令，则可以逆向检索前一个匹配的字符串。

如果想要在当前编辑的文件中向前（文件开始方向）逆向检索，可以使用 “:?” 替代 “:/” 命令。在任何情况下，“n” 和 “N” 的检索方向仍然保持不变。但从逻辑上讲，则恰好与 “:?” 的检索方向相反。

通常，vim 的字符串检索是严格区分大小写字母的。例如，在检索 “china” 时，vim 不会发现 “China”。如果想在检索期间忽略大小写的差异，可以输入 “:set ic” 命令。检索完成后，若要返回默认的匹配方式（区分大小写），可输入 “:set noic” 命令。

如果发现检索的字符串，vim 将会把光标移至（停留在）目标字符串的第一个字符位置。如果未发现检索的字符串，vim 将会在编辑窗口的底部（状态行）中输出 “Pattern not found” 信息，说明检索失败。

在检索过程中，某些特殊字符（/、&、!、.、^、\*、\$、\ 和 ?）具有特定的意义，如果检索字符串中本身也包含这样的字符，则必须在其前面增加转义符号——反斜线 “\”，使 vim 能够将其按普通字符处理。例如，要检索字符串 “anything?”，可输入 “:/anything\?” 命令。要检索一个本身就包含转义符号 “\” 的字符串，可在转义符号之前再加一个转义符号，即输入两个反斜线 “\\”。

## 6.6.2 模式检索

利用 vim 提供的模式检索命令（参见表 6-9），还可以使字符串的检索更精确、更有效：

- 仅检索出现在行首位置的字符串；
- 仅检索出现在行尾位置的字符串；



- 仅检索出现在字首位置的字符串；
- 仅检索出现在字尾位置的字符串；
- 使用通配符检索字符串。

表 6-9

模式检索命令

命 令	命 令 描 述
:/ <i>search</i>	要检索仅仅出现在行首位置的字符串，可以在字符串前面冠以一个上箭头“^”字符，使 vim 仅仅匹配行首位置，而忽略出现在其他位置的字符串。例如，若要从当前行开始检索以“From”为起始字符串的文本行，可以输入“/^From”命令。如果找到匹配的字符串，光标将会移至目的行的行首位置。如果想直接定位匹配的字符串，可接着输入“n”命令。如果找不到匹配的字符串，vim 将会在编辑窗口底部显示“Pattern not found”信息，而光标则继续停留在原来的位置
:/ <i>search\$</i>	要检索仅仅出现在行尾位置的字符串，可以在字符串后面附加一个“\$”字符，使 vim 仅仅匹配行尾位置，而忽略出现在其他位置的字符串。例如，若要从当前行开始检索以“end”字符串结尾的文本行，可以输入“/end\$”命令。同样，如果找到匹配的字符串，光标将会移至目的行的行首位置。如果想直接定位匹配的字符串，可接着输入“n”命令。如果找不到匹配的字符串，vim 将会在编辑窗口底部显示“Pattern not found”信息，而光标则继续停留在原来的位置
:^< <i>search</i> >	要匹配某个单字起始部分的字符串，可在检索字符串的前面冠以“\<”。若要匹配一个单字的结尾部分，可在检索字符串的后面附加“\>”。因此，若要准确地匹配一个完整的字而非字符串，可在检索字符串前后加上“\<”和“\>”。例如，如果想在整个文件中检索“search”这一英文单词，则可以输入“:^< <i>search</i> >”命令
使用“.”、“*”和“[...]" 等通配符	要匹配任何一个字符，可以在检索字符串的相应位置中使用点句通配符“.”。例如，为检索“fun”或“gun”，可以输入“./un”命令。为了指定一个匹配范围，也可以使用范围通配符进行检索。例如，可以使用“:/[a-z]string”命令，使第一个字符仅限于小写字母。另外，也可以使用列举的方式匹配限定的几个字符。例如，“/:[dm]string”命令表示仅检索以“m”或“d”为起始字符的字符串“string”，即仅检索“dstring”或“mstring”。要检索以某个字符范围开头，中间含有某种模式的字符串，可以组合使用多个通配符进行检索。例如，为了检索以小写字母开头，中间含有“string”的字符串，可以使用“:/[a-z]*string*”命令

### 6.6.3 替换字符串

字符串替换是在前述字符串检索的基础上实现的。因此，在检索与替换的过程中可以使用任何通配符。

替换字符串命令的基本语法格式如下。

:[g]/*search-string*/s//*replace-string*/[g][c]

其中，第一个字符命令“g”表示全文检索，“s”表示替换，第二个字符命令“g”表示替换匹配的所有字符串，“c”表示在替换之前须经用户确认。因此，要把文件中的所有字符串“BankA”替换为“BankB”，可输入下列命令。

:g/BankA/s//BankB/g

执行上述命令后，vim 将会把从当前行开始，直至文件结尾范围内的所有“BankA”全部替换为“BankB”。若要在处理之前须先经用户确认，然后再替换，可以增加“c”命令，使 vim 在执行每个替换之前请用户予以确认。例如，使用下列命令，可以使 vim 在用“BankB”替换“BankA”之前，先请用户予以确认。输入“y”表示同意替换，输入“n”表示不同意替换。

:g/BankA/s//BankB/gc

注意，利用 Ctrl-C 按键，可以在中途停止这种“确认一替换”的交互式字符串替换方式。

## 6.7 编辑多个文件

### 6.7.1 编辑多个文件

vim 允许用户同时编辑多个文件。例如，若想在编辑 file1 文件的同时，也编辑 file2 文件，可以使用下列命令打开两个文件。

```
vim file1 file2
```

此时，vim 首先显示第一个文件 file1，因此可以先行编辑 file1；编辑结束后输入“:w”命令，保存 file1 文件。若要编辑 file2，可以输入“:n”或“:n file2”命令；编辑结束后输入“:w”命令，保存 file2 文件。编辑结束后，可以输入“:q”命令，退出 vim 编辑器，或者使用“:q!”命令，中途强制退出 vim 编辑器。

在编辑多个文件期间，可以随时使用“:e filename”或“:n filename”命令，直接转到指定的文件，也可以使用“:n”命令转到下一个文件，或使用“:n #”命令交替编辑最近处理过的两个文件。如果对文件做过任何编辑加工，在跳转之前，vim 要求用户使用“:w”命令保存当前文件，除非设置了 vim 的 autowrite 选项——vim 会在跳转之前自动保存修改后的文件。如果想放弃当前的修改，可以使用“:e! filename”命令，强行转至指定的文件，或者使用“:q!”命令，强制退出 vim 编辑器。

### 6.7.2 合并文件与合并文本行

在编辑过程中，使用 vim 的“r”命令能够很方便地把指定的文件读入（插入）当前文件光标所在的位置，该命令的一般语法格式如下。

```
:line# r filename
```

如果未指定行号，vim 将在光标当前所在行的后面插入文件。例如，如果想在文件的第 10 行之后插入文件 chap2，可以输入下列命令。

```
:10 r chap2
```

另外，也可以先把光标移至第 10 行的位置，然后输入下列命令（这有助于确认插入位置的正确与否）。

```
:r chap2
```

若要把相邻的两个文本行合并为一行，可以把光标移至第一行，然后输入 J 命令。

## 6.8 定制 vim 编辑器的运行环境

### 6.8.1 临时设定 vim 的运行环境

通常，vim 编辑器采用一系列默认的选项定义作为自己的运行环境。这些预置的默认选项基本上能够满足大多数用户的日常编辑处理工作；但有时出于某些特定的需要，或者为了提高编辑效率，需要改变部分选项的默认值。若要查询 vim 编辑器各种选项的具体设置，可以使用下列命令。



:set all

“set all”将会输出 vim 编辑器当前支持的所有选项及其默认设置，如图 6-5 所示。

```

文件(F) 脱机(E) 查看(V) 转换(T) 帮助(H) 帮助(H)
:set all
    options
        ambwidth=single nobisearch readonly termencoding=
        nautoindent noignorecase remap noverse
        nautoread iinserts=0 report=2 textauto
        nosautowrite isearch=0 scroll=10 notextmode
        nautowritenl noisearch scrolljump=1 textwidth=0
        backgroundlight noinforcecase scrolloff=0 nohlsearch
        nobackup noinsertmode nosecure timeout
        backupcopy=none isprint=161-255 selectmode= timeoutlen=1000
        backupext= joinpaces shell=/bin/bash ntimeout
        backupskip=/tmp/ keywords= shellcmdflag=c timeoutlen=-1
        nobinary keywordprg=man shellquote= ttitlelin
        nobomb laststatus=1 shelltemp titfast
        nobuflisted nolazyredraw shellquotes= titscroll=999
        cmdheight=1 lines=24 nosiftaround tittype=term
        columns=80 nolist shiftwidth=8 undolevels=1000
        incompatible listchars=eol:8 noshortname updatecount=200
        nocpindent longplugins noshowfulling updateLines=1000
        coptions=ABeCdfv magic noshowmatch verbose=0
        debug matchtime=5 showmode verbosefiles
        nodecombine maxcombine=2 showtabline=1 nosubshell
        display maxmapdepth=1000 sidescroll=0 warn
        More

```

图 6-5 vim 编辑器当前使用的选项及其默认设置

如果想要改变某个选项的设置，可以使用下列 set 命令。

:set option

其中，option 是 vim 编辑器支持的选项名称。要取消某个编辑器选项的设置，可在选项前增加 no 字样，如下所示。

:set nooption

表 6-10 给出了 vim 编辑器支持的部分重要选项及其说明。

表 6-10

vim 编辑器支持的部分选项

选 项	缩 写	默 认 值	简 单 说 明
all	无	无	在编辑窗口中列出编辑器支持的所有选项
magic	无	magic	<p>在检索字符串时，下列字符默认情况下具有特殊的意义：</p> <ul style="list-style-type: none"> <li><input type="checkbox"/> “.” 表示匹配任何一个字符；</li> <li><input type="checkbox"/> “[... ]” 表示匹配指定字符集合或字符范围中的任何一个字符；</li> <li><input type="checkbox"/> “*” 表示匹配任何字符或字符串。</li> </ul> <p>在设置了 nomagic 选项之后，上述字符将失去其特殊的含义。注意，“^”（表示行首位置）和“\$”（表示行尾位置）总是具有特殊的意义，不受此选项设置的影响</p>
autoindent	ai	noai	这个选项应与 shiftwidth 选项一起发挥作用，使新输入的文本行与上一行起始位置自动对齐。当 vim 处于输入模式时，按下制表键 Tab 将会使光标移至下一个制表位置，按下 Enter 键将会使光标移至下一行，并与上一行的第一个字符位置对齐，按下 Ctrl-T 键将会使当前行移至下一个制表位置，而 Ctrl-D 键将会使当前行反向移至前一个制表位置。这一选项比较适用于程序员
autowrite	aw	now	在编辑多个文件的情况下，如果设置了 autowrite 选项，当使用 “:e” 或 “:n” 命令在文件之间切换时，vim 将会在切换之前自动保存对当前文件所做的编辑处理。否则，vim 将会提示用户把缓冲区中的内容写到磁盘文件中（如果当前编辑的文件中发生了任何变动）
ignorecase	ic	noic	在 vim 编辑器中，字符或字符串的匹配检索通常是严格区分大小写字母的。如果设置了这一选项，可以使 vim 在匹配检索过程中忽略大小写字母的差别
list	无	nolist	通常，vim 将会按照正常的方式显示文本文件，如按照要求把制表符扩展为适当数量的空白字符，不显示回车字符等。如果设置了这个选项，vim 将会把制表符显示为 “\t”，在每一个文本行的后部附加一个美元符号“\$”等

续表

选 项	缩 写	默 认 值	简 单 说 明
laststatus	ls	ls=1	这个选项用于确定是否在编辑窗口中显示状态行。状态行中包括当前文件的名字、加号“+”标志（如果编辑的文件发生变动后尚未被写到磁盘上）和光标的位置等。这个选项的有效值是 0、1 和 2。其中，0 表示关闭状态行的显示，1 表示仅当存在两个以上的文件编辑窗口时才显示状态行，2 表示总是显示状态行
number	nu	nonu	在编辑文本文件时，vim 编辑器通常不会显示文本行的行号。为了在每个文本行前面增加一个行号，可以设置这个选项。注意，vim 编辑器显示的行号并非文件中的部分，也不会被存储到文件中，只是为了编辑的方便，提供一种临时的标记而已。默认值为 nonu
readonly	无	noreadonly	对正在编辑的文件启用写保护机制，当编辑文件时，vim 将会向用户发出警告提示，以避免修改或损害重要的文件
report	无	report=2	这个选项使 vim 能够确定在删除或复制多少文本行时需要在状态行中显示影响的文本行信息，如“8 lines deleted”。当删除或复制较少的文本行（少于等于指定值）时，vim 不会显示任何信息。这个选项的默认值为 2
scroll	scr	scr=nn	这个选项用于控制按下 Ctrl-D 或 Ctrl-U 键时，前滚或后滚多少文本行，其默认值为编辑器窗口高度（窗口行数）的一半。要修改 Ctrl-D 或 Ctrl-U 键滚动的行数，通常有两种实现方式：一是在按下 Ctrl-D 或 Ctrl-U 键之前首先输入一个数字，二是使用这个选项事先设定一个数值，如“:set scroll=10”
shell	sh	sh=path	在使用 vim 编辑器时，用户可以临时或长久地调用一个 Shell，运行单个 Linux 命令，或者交互地访问 Shell，运行多个 Linux 命令。这个选项用于确定 vim 调用哪一个 Shell。默认情况下，vim 把这个选项设置为用户的注册 Shell。为了选用不同的 Shell，可以采用“:set sh=path”形式定义 Shell，其中 path 为 Shell 的绝对路径名
shiftwidth	sw	sw=8	这个选项用于设定制表符的跳转位置，以便在输入模式下，当按下制表键 Tab、Ctrl-T 或 Ctrl-D 键时，光标能够自动地跳转到下一个或前一个制表符位置。这个选项的默认值为 8
showmatch	sm	nosm	输入右圆括号、花括号或方括号时提示相应的左圆括号、花括号或方括号。此选项对输入数学表达式或编写程序用到圆括号或花括号时比较有用
showmode	smd	smd	根据 vim 编辑器当前所处的工作模式，在编辑窗口左下角显示输入模式“--INPUT--”及替换模式“-- REPLACE --”等信息
tabstop=8	ts	ts=8	设置制表键 Tab 的右移距离。默认值为 8 个空格
wrap	无	wrap	wrap 选项用于控制 vim 编辑器怎样显示较长的文本行。为了使 vim 能够把较长的文本行延续到下一行，可以利用这个选项实现自动折行。如果设置了 nowrap，vim 将会截断较长文本行后部的超长部分（只是不显示，并非真正截掉超常部分）。这个选项的默认值为 wrap
wrapmargin	wm	wm=0	指定编辑窗口的右边距。假定终端窗口为 80 列，如果使用“:set wm=8”命令设置右边距，则文本行的逻辑长度不能超出第 72 列的位置。当输入的文本行到达指定的边界，vim 编辑器将会在最接近边界的字（以空格字符为分界符）右边插入一个换行字符。这个选项可以帮助用户摆脱在输入每一行之后再按 Enter 键的麻烦。数值 0（默认值）表示关闭这一功能，其他非 0 数值表示启用并设定编辑窗口的右边距
wrapscan	ws	wrapscan	这个选项影响字符串的检索方式。如果设置了 ws 选项，当检索到达文件的结尾仍未发现匹配的字符串时，vim 将会从头开始继续检索；否则，在到达文件的结尾时，即使仍未发现匹配的字符串，vim 也会停止检索
compatible	cp	cp	除非.vimrc 文件存在，默认情况下 vim 将会尝试采用与 vi 兼容的工作方式

另外，为了加快数据输入的速度，还可以使用下列命令形式，定义用户自己常用的操作形式。

**ab abbr string**

其中，abbr 为 string 的缩写形式，string 为实际上应出现在编辑器中的数据。例如，如果使用



下列命令：

```
ab iscas 中国科学院软件研究所
```

则每当输入一个 iscas 单词时，vim 编辑器会自动代以“中国科学院软件研究所”字符串。

## 6.8.2 永久性地定制 vim 的运行环境

上面介绍的方法只能临时地设置 vim 编辑器的当前运行环境，一旦退出 vim，这种临时设置也随之作废。若要永久性地设置各种选项，以免每次进入 vim 时都要重新设置，可以设置 VIMINIT 变量。如果使用的是 bash，可以在.bash\_profile 中增加下列形式的变量设置。

```
export VIMINIT='set paral para2 ...'
```

例如，为了总是显示状态行，在匹配检索过程中忽略大小写字母的差别，可以把 VIMINIT 变量设置如下。

```
export VIMINIT='set laststatus=2 ignorecase'
```

另外，也可以把常用的 vim 选项及其定义（即 set 命令可以设置的选项和定义）加到系统范围的初始化文件/etc/vim/vimrc 文件，或用户主目录下的.vimrc（或.exrc）文件中。如此，则每当调用 vim 编辑器时，vim 都会自动读取此文件，使定制的选项成为 vim 的永久性运行环境。下面是一个.vimrc 文件示例（注意，不要在命令前加冒号）。

```
set showmode  
ab abc 中国农业银行  
ab boc 中国银行  
ab cbc 中国建设银行  
ab icbc 中国工商银行
```

实际上，还可以在每个目录中创建一个自己的.vimrc 文件，以适应不同的需要。例如，可以在 C++源程序所在的目录中创建一个适应于程序开发的.vimrc 文件，在编写文档的目录中创建一个适应于文档编写的.vimrc 文件。但应注意，仅当用户主目录的.exrc 文件包含“set exrc”命令时，其他目录中的.vimrc 文件才能发挥作用。

## 6.9 其他特殊说明

### 6.9.1 删 除或替 换 特殊字符

在编辑文件时，有时会遇到文件中含有不可打印的特殊字符。例如，由于 Windows 与 Linux 系统使用的行终止符不同，对于 Linux 系统而言，取自 DOS 或 Windows 系统中的文本文件会包含多余的回车字符，如图 6-6 所示。

为了替换或删除这种特殊字符，可以在 vim 的删除或替换命令中，利用“Ctrl-V”键输入特殊字符的 ASCII 编码，以删除或替换特殊字符。例如，回车字符的 ASCII 编码为 13，对应于 Ctrl-M 键，可以采用下列替换命令，删除多余的“^M”字符。

```
1,$ s/^M/
```

注意，上述命令中的“^M”字符不能直接输入，必须先按 Ctrl-V 键，接着再按 Ctrl-M 键，才能输入“^M”字符，如图 6-7 所示。

\*4L. 189C 1..1 All

图 6-6 DOS/Windows 文本文件

### 四、二、植物四大之竹

图 6-7 替换回车字符

### 6.9.2 在编辑期间运行 Linux 命令

有时，可能需要在编辑过程中运行 Linux 命令。例如，如果需要把某个命令的输出作为文件的一部分，则可以利用表 6-11 列出的命令，临时或长时间地运行其他 Linux 命令，或者把 Linux 命令的运行结果引入正在编辑的文件中。

表 6-11 辅助编辑命令

命 令	命 令 描 述
:sh	在编辑文件期间, 如果想长时间地运行 Shell 命令, 然后再返回 vim 编辑器, 可输入 “:sh” 命令。此时, 用户将处于正常的 Shell 命令行工作方式, 当使用 Ctrl-D 键或 exit 命令退出 Shell 时, 即可返回原来的 vim 编辑器, 继续进行其他编辑处理
:!command	在编辑文件期间, 如果想临时地运行某一个 Shell 命令, 然后继续进行编辑处理, 可输入 “:!command” 命令。此时, vim 将会在不退出编辑器的情况下, 执行指定的 Shell 命令。在 Shell 命令运行结束之后, 按下 Enter 键, 即可恢复原来的编辑处理状态
!!command	在编辑文件期间, 如果想把某个 Shell 命令的运行结果直接加到当前编辑的文件中, 使命令的输出替代当前行, 然后继续进行编辑处理, 可输入 “!!command” 命令

## 6.10 vim 编辑器命令总结

表 6-12 分类列出了 vim 编辑器的基本命令及其简要说明。

表 6-12

## 编辑器命令一览表

命 令	简 单 描 述
启动 vim 编辑器:	
<code>vim filename</code>	打开原有的文件或创建一个新文件
<code>vim</code>	打开一个新文件，在编辑过程中或结束编辑时再指定文件名
<code>vim -r filename</code>	恢复因意外停机或终端连接中断而未及时保存最终编辑结果的文件
<code>view filename</code>	以只读方式打开文件。除了不能把编辑处理的最终结果写入文件保存之外，view 的所有编辑功能均与 vim 无异
光标定位命令:	



续表

命 令	简 单 描 述
←↑→	将光标左移、上移、下移或右移一个字符（行）位置
h j k l	同上
-	光标上移一行
Enter 键（或加号“+”）	光标下移一行
退格键	将光标左移一个字符位置
空格键	将光标右移一个字符位置（命令模式）
Ctrl-F	往下（文件结尾方向）滚动一屏
Ctrl-B	往上（文件开始方向）滚动一屏
Ctrl-D	往下滚动半屏
Ctrl-U	往上滚动半屏
Ctrl-E	编辑窗口中的文件内容整体上移一行
Ctrl-Y	编辑窗口中的文件内容整体下移一行
w	将光标右移一个字。光标停留在下一个字的字首位置
W	将光标右移一个字。光标停留在下一个字的字首位置（即使两个字之间存在标点符号）
b	将光标左移一个字。光标停留在下一个字的字首位置
B	将光标左移一个字。光标停留在下一个字的字首位置（即使两个字之间存在标点符号）
e	把光标移至当前所在字（或下一个字）的最后一个字符位置
E	同上，只是以空格字符作为字的分隔符
^	把光标移至当前行的起始位置，即当前行的第一个非空白字符位置
0（零）	同上
\$	把光标移至当前行的行尾，即当前行的最后一个字符位置
H	把光标移至编辑窗口顶部第一行的行首位置
M	把光标移至编辑窗口中间一行的行首位置
L	把光标移至编辑窗口底部最后一行的行首位置
插入文本数据：	
a	在光标当前所在字符位置的后面输入文本数据
A	在光标当前所在行的行尾（即最后一个字符位置）后面输入文本数据
i	在光标当前所在字符位置的前面输入文本数据
I	在光标当前所在行的行首（即第一个非空白的起始字符）前面输入文本数据
o	在光标当前所在行下面的行首位置输入文本数据
O	在光标当前所在行上面的行首位置输入文本数据
修改文本：	
C	替换当前文本行光标所在字符位置之后的所有数据，以 Esc 键结束
cw	替换光标当前所在字符位置及之后的整个字或部分字，以 Esc 键结束
[n]cc	替换当前行，或从当前行开始的 n 行文本，以 Esc 键结束

续表

命 令	简 单 描 述
[n]s	替换光标当前所在位置的单个字符，或从光标当前位置开始的 n 个字符，以 Esc 键结束
S	替换当前行，以 Esc 键结束
r	替换光标当前所在位置的单个字符
r Enter	断行。也可使用“a”或“i”命令加 Enter 及 Esc 键实现
R	从光标当前所在的字符位置开始，替换随后的所有字符，直至按下 Esc 键
xp	交换字符位置。交换光标当前所在位置开始的两个字符位置
~	转换光标当前所在位置字符的大小写
u	撤销最近一次执行的编辑命令，或者依次撤销先前执行的编辑命令
:u	同上(ex 编辑命令)
U	撤销施与当前文本行的编辑处理
<b>删除文本：</b>	
[n]x	删除光标当前所在位置的字符，或者删除从光标当前位置开始的 n 个字符
[n]X	删除光标当前所在位置的前一个字符，或者删除光标当前所在位置之前的 n 个字符
dw	删除光标当前所在位置的一个整字或部分字符。如果光标在字首，则删除整字；如果光标在字的中间任何位置，则删除光标位置及之后的字符
[n]dd	删除光标当前所在的文本行，或者删除从当前行开始的 n 个文本行
D	删除当前文本行从光标位置开始之后的所有字符
dG	删除从当前行开始直至文件最后一行的所有文本行
d[n]G	删除从文件的第 n 行开始直至当前行的所有文本行
:line#1,line#2 d	删除指定的行号 line#1～line#2 之间的所有文本行
<b>复制与移动文本：</b>	
[n]yy	复制光标当前所在的文本行，或者从当前行开始的 n 个文本行
[n]Y	同上
p (小写)	把复制或删除(“dd”命令)的文本行粘贴到光标所在行的下面
P (大写)	把复制或删除(“dd”命令)的文本行粘贴到光标所在行的上面
:line#1,line#2 co line#3	把第 line#1～line#2 行复制到第 line#3 行之后
:line#1,line#2 m line#3	把第 line#1～line#2 行移至第 line#3 行之后
<b>设置行号显示：</b>	
:set nu	在编辑期间增加临时行号
:set nonu	撤销行号显示(默认情况)
Ctrl-G	显示当前文件的名字和当前文本行的行号
<b>设置大小写字母检索准则：</b>	
:set ic	检索字符串时忽略字母的大小写
:set noic	检索字符串时严格区分字母的大小写(默认情况)
<b>定位文本行：</b>	
G	将光标移至文件的最后一行



续表

命 令	简 单 描 述
[n]G	将光标移至文件的第 n 行
检索与替换:	
:/string	向前（文件结尾方向）检索指定的字符串
:?string	向后（文件开头方向）检索指定的字符串
n	按检索方向找出下一个匹配的字符串
N	逆检索方向找出前一个匹配的字符串
:[g]/search/s//replace/[g][c]	检索并替换字符串
清除屏幕:	
Ctrl-L	消除因其他进程的输出信息而干扰的编辑窗口
合并文件与合并行:	
:x filename	在光标所在行之后插入指定文件的内容
:line#l r filename	在第 line#l 行之后插入指定文件的内容
J	把相邻的两个文本行合并为一行（把下一行合并到光标当前所在行的后面）
保存编辑结果与退出 vim 编辑器:	
:w	保存编辑处理后的结果（把内存缓冲区中的数据写到文件中）
:w!	强制保存编辑处理后的结果
:wq	保存编辑处理后的结果，然后退出 vim 编辑器
:wq!	强制保存编辑处理后的结果，然后退出 vim 编辑器
ZZ	保存编辑处理后的结果，然后退出 vim 编辑器
:q	在未做任何编辑处理时，可以使用此命令退出 vim 编辑器
:q!	强制退出 vim 编辑器，放弃编辑处理后的结果
:w filename	把编辑处理后的结果写到指定的文件中保存
:w! filename	把编辑处理后的结果强制写到指定的文件中保存，即使文件已经存在
:wq! filename	把编辑处理后的结果强制写到指定的文件中保存，即使文件已经存在，然后退出 vim 编辑器
其他:	
:f 或 Ctrl-G	显示文件的名字、编辑状态、文件总的行数、光标当前所在的行号和列号，以及当前行之前的行数占整个文件总行数的百分比
Ctrl-V	用于输入其他控制字符

## 第7章

### Shell 基础知识

本章主要介绍 Shell 编程的基础知识。我们将从什么是 Shell 脚本开始，介绍 Shell 变量与变量替换、命令与命令替换及各种 test 语句。讨论怎样编写和运行 Shell 脚本，以及怎样利用 Shell 的编程机制和 Linux 系统提供的丰富命令，创建自己的 Shell 脚本。其内容主要包括：

- Shell 与 Shell 编程；
- 变量与变量替换；
- 命令与命令替换；
- 进程替换；
- test 语句；
- 命令行的解释执行过程。

器

之前





在本章及随后的第 8 章中，将以 Bash Shell 为基础，结合 Linux 系统的日常管理和例行维护，介绍 Shell 编程的基本概念和技巧。Shell 编程涉及的内容非常广泛，而且，如果想把 Shell 编程的功能发挥得淋漓尽致，还需要对 Linux 系统有深入的了解。随着用户对 Shell 编程和 Linux 系统的逐渐熟悉，在日常的系统管理和维护过程中，有关的内容自然而然也就掌握了，没有必要从一开始就要求大求全。因此，我们采取的原则是介绍 Shell 编程的原理，重在实用。

## 7.1 Shell 与 Shell 编程

在 Linux 系统的层次组织结构中，Shell 是一个重要的组成部分。Shell 是用户与 Linux 系统内核进行联系的桥梁。Linux 通过 Shell 界面，接受用户的请求，利用系统的资源，为用户提供服务。根据 Shell 的调用方式，Linux 系统中的 Shell 主要分为交互式注册 Shell、交互式非注册 Shell 及非交互式 Shell3 种。

如果调用方式不同，Shell 的初始化过程也不相同。在 GNOME 注册界面或“login:”提示下输入用户名与密码，并成功地注册之后，Linux 系统将会调用交互式注册 Shell。交互式注册 Shell 利用/etc/profile 和/etc/bash.bashrc 文件以及用户主目录中的~/.profile 等初始化文件（或称启动文件），设置用户的运行环境。

在 Shell 命令提示符下输入 sh 或 bash 等命令，将会进入交互式非注册 Shell（可以看作当前 Shell 的子 Shell）。此时，Shell 将会读取并执行/etc/bash.bashrc 和~/.bashrc 等初始化文件，同时还会继承注册 Shell 利用初始化文件设置的各种环境变量。

非交互式 Shell 主要用于运行 Shell 脚本。当利用命令行界面提交一个 Shell 脚本时，从开始执行到结束，其整个运行环境即为非交互式 Shell。非交互式 Shell 通常不使用任何用户初始化文件，但会继承注册 Shell 设置的运行环境，从实际效果来讲，也间接地使用或继承了用户初始化文件的运行结果。但非交互式 Shell 在开始运行之初，将会检查 BASH\_ENV 变量。如果变量已经设置，非交互式 Shell 将会以 BASH\_ENV 变量的值作为初始化文件予以执行。

### 7.1.1 为什么需要 Shell 编程

Linux 系统提供了丰富的系统命令和各种各样的实用程序，只要经过适当的组合，基本上都可以满足绝大部分应用需求，而不必重新编写新的程序，这是 Shell 脚本最大的功用所在。实际上，Linux 系统的很多系统配置、启动和管理任务都是通过 Shell 脚本实现的。

Linux 操作系统中的 Shell 既是一个命令解释程序，也是一种强有力的编程语言。作为操作系统内核与用户之间的一个交互界面，Shell 可以解释执行用户输入的 Linux 命令，可以实行 I/O 重定向，提供管道、元字符匹配以及文件名生成等功能。

作为一种编程语言，Shell 也可以执行简单的批处理，利用 Linux 系统的丰富命令、实用程序和各种工具，完成特定的功能需求。如果这样仍然不能满足需要，还可以利用 Shell 的编程机制，如测试语句、条件转移和循环控制结构，执行复杂的操作，增强 Shell 脚本的功能和灵活性。

实际上，借助于 Linux 系统本身的强大功能，利用 Shell 的编程机制，很容易实现各种系统管理和日常维护任务，而不需要采用其他高级编程语言，为每一项任务重新开发额外的工具。

在第 7 和第 8 章中，我们将利用大量的例子，介绍 Shell 的各种功能特性。其中的大部分例子都是从实践中挑选出来的，且经过了验证（注意，使用前应首先利用 chmod 命令，使 Shell 脚本具有可执行的访问权限，然后再运行，参见后面的说明）。

从 1979 年 Steve Bourne 开发出的第一个 Shell 与 UNIX 系统第 7 版一同推出开始，Bourne Shell 即成为 UNIX 系统必备的用户界面。随着 UNIX 系统的不断发展，David Korn 在 Bourne Shell 和 C Shell 的基础上，开发出 Korn Shell 之后，进一步增强了 Shell 的功能。Linux 系统中提供的 Bash Shell 与 Bourne Shell 和 Korn Shell 一脉相承，在功能上又有更进一步的提高。

如果想要成为专业水平的 Linux 系统管理员，熟练地掌握 Shell 编程的技巧是一项必不可少的基本要求（即使不一定非要切实编写 Shell 脚本）。例如，在系统的启动过程中，Linux 执行的就是位于 /etc/init.d 目录中的一系列 Shell 脚本，用以完成系统的各种配置、设置，以及启动各种系统服务等。理解和灵活地运用这些 Shell 脚本，对于分析系统的行为是非常重要的一项技能。

Shell 编程是一种简单直观的，能够快速实现复杂功能需求的解决方法。Shell 的语法规则简单、直观、易懂，学习和编写 Shell 脚本并不困难。实际上，只要熟悉 Linux 命令，了解 Shell 编程的基本规则和结构语句，利用 Shell 的编程机制，把若干 Linux 命令、实用程序或应用程序组合在一起，即可写出自己的 Shell 脚本。

Shell 编程吸取了 UNIX/Linux 系统把复杂任务分解成简单子任务的优良传统，使应用开发人员能够把各种系统工具和实用程序灵活地组合在一起，完成特定的功能需求。这是一种非常好的解决问题的手段，避免了针对各种各样的用户需求，都要重新开发一套复杂工具的烦恼。事实上，Linux 系统已经提供了丰富的、功能强大的系统工具和实用程序，只要灵活地运用这些现成的工具，就可以满足我们的绝大部分需求。

### 7.1.2 什么是 Shell 脚本

利用文本编辑器，事先把一系列 Linux 命令或可执行程序放到文件中，然后修改文件的访问权限，使之能够像系统命令或实用程序一样执行，这样的文本文件就是 Shell 脚本，或称 Shell 程序。简单地讲，Shell 脚本就是一种包含若干 Linux 命令或可执行程序的文本文件。

Shell 脚本可由任何 Linux 命令组成。当执行 Shell 脚本时，文件中的所有命令将从头到尾，一个一个地顺序执行（除非 Shell 脚本中含有控制结构语句）。就像用户在终端前面，以命令行方式，每次输入一个命令，让系统依次执行一样。

同其他面向过程的编程语言一样，除了普通 Linux 命令或可执行程序之外，Shell 脚本还可以包含控制结构和变量，也可以带有参数，使之完成一定的功能需求。

最简单的 Shell 脚本可以是仅仅包含一个或一组 Linux 系统命令的文件。使用 Shell 脚本不仅能够容易地实现批量处理，避免一遍又一遍地重复输入一系列常用的命令，还可以对 Shell 脚本进行修改或定制，从而满足不同的应用需求或功能需求。

如果再使用变量、控制结构和参数传递等一整套编程机制，灵活地运用 Linux 系统本身提供的丰富命令和工具，Shell 编程无疑会如虎添翼，将提供更加强大的功能。

### 7.1.3 运行 Shell 脚本

Shell 脚本有两种执行方式。第一种方式是利用 sh 或 bash 等命令（注意，在 Ubuntu 8.10 版



的 Linux 系统中不能使用 sh，除非脚本文件的第一行中指定了究竟使用哪一个 Shell，因为此时的 sh 是 dash 的符号链接文件，而 dash 与 bash 是不兼容的），把 Shell 脚本文件名作为 Shell 的参数，由 Shell 直接读取并解释执行 Shell 脚本文件中的每一个命令，这些命令就像直接从键盘上输入一样。这种 Shell 脚本执行方式只要求 Shell 脚本文件具有“可读”的访问权限。

假定 Shell 脚本文件 whogrep.sh 包含命令：

```
$ who | grep cathy
```

采用下列方式运行 Shell 脚本

```
$ sh whogrep.sh
```

的效果则相当于执行“who | grep cathy”命令。

第二种执行方式是首先利用 chmod 命令设置 Shell 脚本文件，使 Shell 脚本具有“可执行”的访问权限。然后在命令提示符下直接输入 Shell 脚本文件名，作为一个普通的命令，交由 Linux 系统执行，如下所示。

```
$ chmod 755 whogrep.sh
$ whogrep
cathy      6419  6403  0 20:56 pts/1    00:00:00 su - cathy
cathy      6425  6419  0 20:56 pts/1    00:00:00 -su
gqxing     6654  6591  0 21:00 pts/2    00:00:00 grep cathy
$
```

因此，一旦写好一个 Shell 脚本（假定为 *scriptname*），可以采取上述两种调用方式之一。但是，一般不建议使用第一种调用方式——“sh *scriptname*”，尤其不建议采用“sh < *scriptname*”调用方式，因为这将禁止 Shell 脚本读取标准输入。注意，以“sh *scriptname*”方式调用一个 Shell 脚本可能会禁止 Shell 的部分扩充功能，因而引起 Shell 脚本无法正确地执行。

在采用第二种调用方式直接运行可执行的 Shell 脚本之前，首先应当使用下列 chmod 命令之一，把 Shell 脚本文件设置为可执行文件。

- chmod 755 *scriptname*（文件属主可以读、写和执行，其他用户只能读与执行）；
- chmod a+rwx *scriptname*（增加任何用户读与执行的权限）；
- chmod u+rwx *scriptname*（仅增加文件属主读与执行的权限）。

按照上述要求设置 Shell 脚本文件的访问权限之后，可以采用下列方式，直接运行 Shell 脚本：

- *scriptname*（如果命令检索路径包含当前目录）；
- *path/scriptname* 或 *./scriptname*（如果命令检索路径不包含当前目录）。

在测试和调试最终完成之后，如果愿意，也可以作为一个工具，把 Shell 脚本文件移至某个公用的用户命令目录，如 /usr/local/bin 目录中，使所有的用户均可访问新增的 Shell 脚本。

### 7.1.4 退出与出口状态

当一个命令或进程终止运行时，将会自动地向父进程或 Shell 返回一个出口状态。如果命令或进程成功执行完毕，将会返回一个数值为零的出口状态。如果命令或进程在执行过程中出现异常而未正常结束，将会返回一个非零值的出错代码。

同样，一个 Shell 脚本结束运行时也会返回一个出口状态。在 Shell 脚本中，最后执行的一条命令将决定整个 Shell 脚本的出口状态。此外，Shell 的内部命令 exit 也可用于随时终止 Shell 脚本的执行，并返回 Shell 脚本的出口状态。

因此，在 Shell 脚本中，开发人员可以利用“exit [n]”命令，在终止执行 Shell 脚本的同时，

向调用 Shell 脚本的父进程(即 Shell)返回一个数值为 n 的出口状态。其中, n 必须是一个位于 0~255 范围内的整数值。按照惯例, 一个命令或 Shell 脚本正常终止时应返回 0, 运行有误时可返回一个 1~255 范围内的整数值。

当 Shell 脚本结束运行前执行的最后一个命令是不带任何数值参数的 exit 语句时, Shell 脚本的最终出口状态是 Shell 脚本执行 exit 语句之前执行的最后一条命令的出口状态。依据这一返回值, Shell 可以判断整个 Shell 脚本的运行是否正常结束。

例如, 假定存在下述 Shell 脚本, command\_last 命令的返回值就是整个 Shell 脚本的出口状态。

```
#!/bin/bash
command_1
command_2
.....
command_last
exit
```

在此情况下, 其效果等价于最后执行了一个“exit \$?”语句, 如下所示。

```
#!/bin/bash
command_1
command_2
.....
command_last
exit $?
```

在此情况下, 也可以干脆去掉 exit 语句, 直接使用最后一条命令的出口状态作为整个 Shell 脚本的返回值, 如下所示。

```
#!/bin/sh
command_1
command_2
.....
command_last
```

对于顺序执行的 Shell 脚本, 这种做法没有任何问题。但对于带有控制转移和结构语句的 Shell 脚本而言, 则需要在任何可能的 Shell 脚本终止位置, 尽量使用能够返回不同出口状态的“exit [n]”语句, 以便了解 Shell 脚本的实际运行结果。

在任何时候或任何位置, 都可以使用 Shell 的内部变量 \$? 给出之前执行的最后一条命令的出口状态。Shell 函数也是如此, 当从函数中返回时, \$? 变量的值表示函数中实际执行的最后一条命令的出口状态, 这也是 Shell 给出函数返回值的实现方法。

Shell 脚本运行终止之后, 也可以在 Shell 命令提示符下, 使用 \$? 内部变量返回 Shell 脚本的出口状态, 即脚本中执行的最后一条实际命令的出口状态。在 Linux 系统中, 为了测试一个命令或 Shell 脚本的执行结果, \$? 变量是非常有用的。

下面的例子说明了如何获取命令以及 Shell 脚本的出口状态。

```
$ cat testexit
#!/bin/bash
LANG=C      # 在英文语言环境中, 其说明信息有时更准确一些 (本书部分例子中的输出数据)
echo hello  # 也是在事先设置了英文语言环境后的结果)
echo $?      # 输出脚本中间单个命令的出口状态。这里的返回值为 0, 因为 echo 命令总是
            # 能够成功地执行
lsdf        # 不存在的命令
echo $?      # 输出脚本中间单个命令的出口状态。这里的返回值为非 0, 因为 lsdf 并不是
            # 一个合法的命令, 因而无法执行
echo
exit 100    # 脚本执行到此结束, 同时以 100 作为整个 Shell 脚本的返回值
$ testexit
hello
```



```
0  
testexit: line 6: lsdf: command not found  
127  
  
$ echo $?      # 脚本终止时再执行此命令，将会返回整个 Shell 脚本的出口状态 (100)  
100  
$
```

“!”是逻辑非运算符，表示命令返回值或测试结果的否定值，因而也能够影响其出口状态。下面的例子就说明了这个问题。

```
$ true          # true 是一个内部命令  
$ echo "exit status of \"true\" = $?"    # 返回值为 0  
exit status of "true" = 0  
$ ! true  
$ echo "exit status of \"! true\" = $?"    # 返回值为 1  
exit status of "! true" = 1  
$
```

注意，“!”与 true 之间必须留有一个空格，否则将会导致系统发出“命令不存在”的出错信息。

### 7.1.5 调用适当的 Shell 解释程序

通常，Shell 脚本的第一行均包含一个以“#!”为起始标志的文本行，这个特殊的起始标志告诉系统：当前文件包含一组命令，需要提交给指定的 Shell 解释执行。“#!”特殊标志实际上是一个两字节的文件类型“标识码（magic code）”，表示当前的文件是一个可执行的 Shell 脚本文件。紧随“#!”标志的是一个路径文件名，指向用于执行当前 Shell 脚本文件的命令解释程序。

在发现“#!”之后，系统将会采用指定的命令解释程序替代当前的 Shell，从第二行开始解释执行脚本中的每一个有效的命令（注释行除外）。指定的命令解释程序可以是一个 Shell、一个普通的 Linux 命令甚至一个应用程序。

下面列出的特殊标志行都是合法的，分别表示调用不同的命令解释程序或普通程序，解释执行当前的 Shell 脚本。其中，“#!/bin/bash”表示调用 Linux 系统默认的 Shell，即 Bash。Bash 是 Linux 系统环境最通用的 Shell。“#!/bin/ksh（或/usr/bin/ksh）”表示调用 Korn Shell 解释执行当前的 Shell 脚本，如此等等。

```
#!/bin/sh  
#!/bin/ksh  
#!/bin/bash  
#!/bin/tcsh  
#!/bin/zsh  
#!/bin/more  
....
```

依据 Shell 脚本中的特殊标志行，当指定的 Shell 开始解释执行脚本时，将会把标志行作为注释处理。注意，特殊标志后面给出的路径文件名必须是正确的。否则，系统将会给出命令解释程序有误的错误信息，并立即终止执行当前的 Shell 脚本。

按照上述说明，如果 Shell 脚本中存在特殊标志行，则调用适当的 Shell 解释执行当前的 Shell 脚本；如果 Shell 脚本中包含多个特殊标志行，只有第一个标志行起作用；如果 Shell 脚本中只用到一些常用的系统命令，而没用使用特殊的 Shell 内部命令，特殊标志行可以省略。

下面的 rmfile 是一个比较特殊的 Shell 脚本，其中的标志行表示需要调用/bin/rm 命令，执行当前的脚本文件。注意，执行这个 Shell 脚本时不会产生任何输出，尽管其中存在若干 echo 语句，

也不会执行到 exit 语句，唯一的运行结果（或者说唯一的作用）就是删除这个脚本文件本身。假定当前目录包含两个文件，其中之一就是 rmfile 脚本文件，运行后的结果如下。

```
$ ls -l
总用量 8
-rwxr--r-- 1 gqxing gqxing 856 2008-11-07 21:10 file2
-rwxr-xr-x 1 gqxing gqxing 111 2008-11-07 21:16 rmfile
$ cat rmfile
#!/bin/rm
echo "This line will never print on screen."
echo "And the current file will be deleted also."
exit 1
$ rmfile
$ ls -l
总用量 4
-rwxr--r-- 1 gqxing gqxing 856 2008-11-07 21:10 file2
$
```

依照上述脚本，可以尝试使用 “#!/bin/more” 作为特殊标志行，创建一个 Shell 脚本文件，使之显示紧随其后的文本信息。把文件的访问权限修改为可执行文件，然后再执行这个 Shell 脚本时，将会显示脚本文件中除了标志行之外的所有内容，如下所示。

```
$ cat readme.sh
#!/bin/more
This is a script to display this file self.
Contents of this file will be displayed
when you execute this script file.
$ readme.sh
#!/bin/more
This is a script to display this file self.
Contents of this file will be displayed
when you execute this script file.
$
```

### 7.1.6 位置参数

从命令行上传递给 Shell 脚本的参数、传递给函数的参数或通过 set 命令得到的参数通常被称作位置参数。位置参数可以依其出现的顺序按序号引用 (\$0、\$1、\$2、……)，故称作位置参数。按照惯例，\$0 是 Shell 脚本文件的名字，由调用 Shell 脚本的进程（即 Shell）负责设置\$0 参数。\$1 是第一个实际参数，\$2 是第二个实际参数，如此类推。注意，从第 10 个位置参数开始，必须加花括号，如\${10}、\${11} 和 \${12} 等。特殊变量\$\* 和 \$@ 表示所有的位置参数，\$# 表示位置参数的总数。

因此，利用特殊的内部变量 \$#、花括号和感叹号 “!”，可以容易地引用传递给 Shell 脚本的最后一个位置参数，如下所示。

```
args=$#          # 传递的参数数量
lastarg=${!args}      # 注意，不能使用“lastarg=${!$#}”直接引用
```

为了改变位置参数的内容，可以使用 shift 命令。顾名思义，shift 命令的功能就是重新分配传递给 Shell 脚本或函数的位置参数：把参数的位置从右到左依次左移一位，整个位置参数的总数也随之减 1。例如，假定位参数的总数为 N，执行 shift 命令之后，将会发生 \$2→\$1、\$3→\$2……\$N→\$N-1 等变化。原来的 \$1 和 \$N 不复存在，但 \$0（即 Shell 脚本的文件名）保持不变。

利用 shift 命令，可以依次处理每一个位置参数。因此，如果采用适当的循环结构与 test 语句，



即可处理任意数量的位置参数。尽管使用花括号也可以做到这一点，但 shift 命令更为灵活、方便。

下面是一个利用 shift 命令和 until 循环显示所有位置参数的例子。当使用命令行参数 “The arguments submitted to the script” 调用这个 Shell 脚本时，通过 shift 语句，until 语句只需检查第一个位置参数是否为空即可遍历每一个位置参数。在依次处理（显示）每一个位置参数之后，整个脚本运行结束。

```
$ cat echoarglist
#!/bin/bash
until [ -z "$1" ]          # 表示直至所有参数均已得到处理再结束
do
    echo -e "$1 \c"
    shift
done
echo
exit 0
$ echoarglist The arguments submitted to the script
The arguments submitted to the script
$
```

每当执行一次 shift 命令，\$@ 和 \$\* 也会依次舍弃第一个位置参数，保留其余的位置参数。同时，\$# 的数量也相应地减 1，示例如下。

```
$ cat echoarglist2
#!/bin/bash
echo "$# ----- $@"
shift
echo "$# ----- $@"
shift
echo "$# ----- $@"
exit 0
$ echoarglist2 1 2 3 4 5
5 ----- 1 2 3 4 5
4 ----- 2 3 4 5
3 ----- 3 4 5
$
```

下面是另一个利用 \$\* 和 \$@ 列出所有位置参数的例子。当使用命令行参数 “one two three” 调用 echoarglist3 脚本时，由于 \$\* 与 \$@ 之间的细微差别（参见 7.2.3 小节），且 \$\* 的引用方式不同，在终端窗口上显示的位置参数结果也不完全相同。

```
$ cat echoarglist3
#!/bin/bash
if [ -z "$1" ]
then
    echo "Usage: `basename $0` argument1 argument2 ...."
    exit 1
fi
index=1
echo "Listing arguments with \"\$*\":"
for arg in "$*"      # $* 加双引号意味着把所有的参数作为一个单词处理
do
    echo "Argument #$index = $arg"
    let "index+=1"
done
echo "Entire argument list is seen as single word."
echo
```

```

index=1
echo "Listing arguments with \"\$@\":"
for arg in "$@"          # $@加双引号表示把参数看作多个单词
do
    echo "Argument #$index = $arg"
    let "index+=1"
done
echo "Argument list is seen as separate words."
echo
index=1
echo "Listing arguments with \$* (unquoted):"
for arg in $*# 如果$*未加引号, 表示把参数看作多个单词
do
    echo "Argument #$index = $arg"
    let "index+=1"
done
echo "Argument list is seen as separate words."
exit 0
$ echo $arglist3 one two three
Listing arguments with "$*":
Argument #1 = one two three
Entire argument list is seen as single word.

Listing arguments with "$@":
Argument #1 = one
Argument #2 = two
Argument #3 = three
Argument list is seen as separate words.

Listing arguments with $* (unquoted):
Argument #1 = one
Argument #2 = two
Argument #3 = three
Argument list is seen as separate words.
$
```

在结束本节之前, 最后再说明一点。在编写 Shell 脚本时, 应注意使用模块化的方法组织 Shell 脚本。在学习 Shell 编程的过程中, 还应随时注意多积累, 多收集一些“模型”代码片段, 这对将来编写 Shell 脚本是非常有用的, 甚至可以考虑建立一个代码片断储备库。例如, 下列代码片段就是一个比较好的通用开场白, 其目的是测试调用 Shell 脚本时是否提供了正确数量的参数。

```

if [ $# -ne $Number_of_expected_args ]
then
    echo "Usage: `basename $0` script_arguments"
    exit 1
fi
```

## 7.2 变量与变量替换

变量是每一种编程语言和脚本语言中都不可缺少的重要组成部分之一。在 Shell 中, 变量可用于字符串操作、算术运算、参数传递及其他运算。实际上, 变量无非是系统分配的一个或一组内存位置, 用以存储各种数据。

除了特殊字符, Shell 变量名可以由任何字母、数字和下划线等字符组成, 但第一个字符必须



是字母或下划线。变量名的长度没有限制。

与其他编程语言不同的是，Shell 并不区分变量的类型。实际上，Shell 仅支持一种类型的变量，即字符串变量，也就是说，Shell 中的所有变量都是字符串类型的。但 Shell 对字符串变量的处理并不仅限于字符串操作。

在 Shell 的处理过程中，取决于上下文以及采用的运算符和命令工具，Shell 允许对变量执行诸如整数算术运算等多种操作。能够影响变量类型的另外一个因素是变量值的首字符。如果首字符为数字，表示还可以对变量执行整数算术运算，否则只能执行字符串操作。

### 7.2.1 变量分类

从用途上考虑，变量可以划分为内部变量、本地变量、环境变量、参数变量和用户定义的变量。内部变量是为便于 Shell 编程而由 Shell 设定的变量，如表示错误类型的 ERRNO 变量等（详见 7.2.3 小节）。在代码块或函数中定义的，且仅在定义的范围内有效的变量称作本地变量。环境变量是为系统内核、系统命令和应用程序提供运行环境而设定的变量，常见的有 PATH、HOME 和 LANG 等变量。参数变量指调用 Shell 脚本或函数时传递的变量（因此，在 Shell 或 Shell 脚本中，变量替换也称参数替换）。用户定义的变量是为运行用户程序或为完成某种特定的任务而设定的普通变量或临时变量。

广义地讲，每个进程都有一个运行“环境”，这个运行环境将影响进程的执行行为。Shell 也存在一组可以引用的环境变量，其中的每个变量都被赋予一定的信息。从这个意义上讲，Shell 同其他任何进程是一样的。

每当开始运行时，Shell 都会创建一组自己的环境变量。更新或增加新的环境变量将会引起 Shell 更新自己的运行环境。如果在 Shell 脚本中设置了环境变量，通常需要使用 export 命令予以公布，以便子进程能够继承。

注意，环境变量的空间分配是有一定限制的。建立太多的环境变量将会占用额外的空间，因而可能会引起问题。

### 7.2.2 变量的赋值

变量的赋值可以采用赋值运算符“=”实现，其语法格式如下。

**variable=value**

在 Shell 中，等号“=”是一种通用的赋值运算符，可用于数字和字符串赋值。变量赋值也可用于声明变量，对变量进行初始化或改变变量的值。另外，为使其他 Shell 脚本或程序能够使用定义的变量，需要利用“**export variable**”命令，把变量导至 Shell 运行环境。注意，赋值运算符“=”前后不能有空格。例如，要定义一个 EDITOR 环境变量，可以使用下列命令。

```
$ EDITOR=vi; export EDITOR  
$
```

在变量的赋值过程中，如果赋的值是由多个单词组成的字符串，则应在字符串前后增加一对单双引号，以便 Shell 能够把字符串作为一个值处理，如下所示。

```
$ TITLE="Name      Phone      Email-Address"; export TITLE  
$
```

未初始化的变量（即未赋值的变量）的值为“null”（注意，使用未赋值的变量有时会出现问

题)。利用下列变量赋值形式，即可声明一个未初始化的变量。

**variable=**

另外一种赋值方法是利用 `read` 语句在执行过程中为变量赋值(参见 7.3.2 小节有关 `read` 语句的介绍)。此外，还可以在 `for` 循环结构语句中，采用“`for variable in word-list`”的形式为变量赋值(详见第 8 章)。

在 Shell 脚本中，适当地使用变量能够提高脚本的可读性和灵活性，确保数据的一致性，而且便于维护。

### 7.2.3 内部变量

Shell 提供了一组丰富的内部变量，能够为用户的 Shell 编程提供支持。熟练地掌握 Shell 内部变量，对以后的 Shell 编程有极大的帮助。为了使读者能够对 Shell 内部变量的概貌有一个全面的了解，也为了方便阅读本书，以及在实际应用中作为参考，表 7-1 给出了部分常用的主要内部变量。

表 7-1 Shell 自动设置的内部变量

变 量	简 单 说 明
#	\$#变量是命令行参数或位置参数的数量
_	\$_变量是传递给 Shell 脚本的执行标志(参见 <code>set</code> 内部命令)
?	\$?变量是最近一次执行的命令或 Shell 脚本的出口状态
\$	\$\\$变量是 Shell 脚本的进程 ID。Shell 脚本经常使用\$变量组织临时文件名，确保文件名的唯一性
_	_是一个特殊的变量，在 Shell 开始运行时，变量的初始值为 Shell 或其执行的 Shell 脚本的绝对路径名，之后是就最近执行的命令的最后一个选项或参数等。例如，注册后打开一个终端窗口，接着执行一个 <code>du</code> 命令时，\$_变量的值如下。  \$ echo \$_ /etc/bash_completion \$ du /usr/bin /usr/sbin 171956 /usr/bin 38508 /usr/sbin \$ echo \$_ /usr/sbin \$
!	\$!变量的值是最近运行的一个后台进程的 PID
*	\$*变量表示所有的位置参数，其值是所有位置参数的值，每个参数均可看作一个单独的字。引用时，\$*相当于 \$1、\$2、\$3……，表示多个参数；而“\$*”则相当于“\$1 \$2 \$3……”，表示一个参数
@	\$@变量类似于\$*，表示所有的位置参数，其值也是所有位置参数的值，每个参数均可看作一个单独的字。引用时，\$@相当于\$1、\$2、\$3……，表示多个参数；但“\$@”则相当于“\$1”、“\$2”、“\$3”……，每个参数均为前后加引号的字符串。也就是说，参数是原封不动被传递的，未作解释或扩展
LINENO	Shell 脚本中当前执行的命令的行号。仅当调试 Shell 脚本时，这个变量才有意义
OLDPWD	利用 <code>cd</code> 命令改换到新目录之前所在的工作目录
OPTARG	<code>getopts</code> 命令已经处理的前一个选项参数
OPTIND	<code>getopts</code> 命令已经处理的前一个选项参数的索引
PPID	\$PPID 是当前进程的父进程的 PID
PWD	表示当前工作目录，其变量值等同于 <code>pwd</code> 命令的输出



续表

变    量	简    单    说    明
RANDOM	每次引用这个变量时，即可生成一个均匀分布于 0~32 767 范围内的随机整数。如果赋予 RANDOM 变量一个数值，可以达到初始化随机数序列的目的
REPLY	当使用 read 命令读取输入数据时，如果没有指定变量参数，可以把 REPLY 变量用作 read 命令的默认变量，从而把 read 命令读取的输入数据赋予 REPLY 变量
SECONDS	\$SECONDS 变量是脚本已经运行的时间（秒数）

表 7-2 给出了 Shell 使用的部分主要环境变量。

表 7-2

Shell 使用的环境变量

变    量	说    明
BASH_ENV	当调用非交互式的 Bash 运行 Shell 脚本时，如果这个变量已经被设置，Shell 将会使用这个变量值指定的绝对路径文件名，作为可在当前环境中执行的初始化文件。典型情况下，.profile 等初始化文件是在会话启动阶段执行的，而 BASH_ENV 变量指定的文件是在 Shell 运行的初始阶段执行的。对于 BASH_ENV 变量指定的文件，Shell 采取与点“.”命令类似的方法解释执行，即在当前环境中执行其中的命令。另外，BASH_ENV 变量指定的文件只要具有可读的访问权限即可，不必是可执行文件。但与使用点“.”命令执行脚本不同的是，执行 BASH_ENV 变量指定的文件时不会使用 PATH 变量检索命令文件，这主要是出于安全考虑。在设置 BASH_ENV 变量值时，其中可以包含变量替换和命令替换等
CDPATH	定义 cd 命令的检索路径。如果 cd 命令中指定的目的目录为相对路径名，cd 命令首先会在当前目录“.”中搜寻目的目录。如果未发现目的目录，则开始检索 CDPATH 变量中列举的每一路径名，直至找到目的目录，并成功地改换工作目录。如果最终仍未发现目的目录，则保持当前工作目录不变。例如，假定 CDPATH 变量设置为 /home/gqxing，其中存在两个子目录 bin 和 src。如果用户当前位于 /home/gqxing/bin 目录中，输入“cd src”命令后，即使没有指定全路径名，仍可把工作目录改换到 /home/gqxing/src 中
COLUMNS	用于定义终端窗口的列宽。select 内置命令使用此变量值确定数据显示的宽度，以便输出菜单选项列表。此变量值也用于确定 Shell 编辑窗口的列数
EDITOR	用于确定命令行编辑使用的编辑程序，通常可以设为 vi 或 emacs 等用户熟悉的编辑器
FCEDIT	用于设定 fc 内置命令使用的默认编辑器，以便用户能够使用熟悉的编辑器编辑命令行
HISTFILE	用于指定存储命令历史记录的文件，默认的文件为 ~/.bash_history
HISTFILESIZE	用于设定命令历史文件保存的最大命令记录数量，默认值为 500（条命令）
HISTSIZE	用于设定命令历史缓冲区保存的最大命令记录数量，默认值为 500（条命令）
HOME	用户主目录的路径名（也是 cd 命令的默认参数）。普通用户的主目录通常为 /home/username，超级用户的主目录为 /root
IFS	字段分隔符（默认值为空格、制表符和换行符）。字段分隔符通常用于解析命令行或字符串的基本构成元素。简言之，字段分隔符决定了 Shell 在解析命令行或字符串时怎样确定字段或单字等基本元素的边界。IFS 变量值中的第一个字符用于解析 \${*} 变量中的位置参数
INPUTRC	用于设定 readline 启动文件的名字，默认值为 ~/.inputrc
LANG	用于设置语言环境（参见第 13 章）
LC_ALL	用于统一设置 LC_* 系列变量的值
LC_CTYPE	用于确定系统怎样处理各种语言环境的字符集，包括字符的分类，字符的大小写转换规则，以及其他字符属性
LC_MESSAGES	用于确定采用何种语言输出系统提示信息
LC_NUMERIC	用于确定本地化千分数值的显示格式
LINES	用于定义终端窗口的行宽。select 内置命令使用此变量值确定数据显示的高度，以便输出菜单选项列表。此变量值用于确定 Shell 编辑窗口的行数
MAIL	用于定义用户邮箱的路径文件名
MAILCHECK	指定 Shell 检测邮件的频度（以秒为单位），默认值为 60 秒。如果未设置这个变量，或者把变量设置为小于或等于 0 的数值，Shell 将停止检测邮件

续表

变 量	说 明
MAILPATH	定义系统检查是否有新邮件到来的文件名
PATH	指定命令的检索路径及顺序（普通用户的命令检索路径通常包括/bin 和/usr/bin 等目录，超级用户的命令检索路径通常包括/sbin、/bin、/usr/sbin 和/usr/bin 等目录）。当用户输入命令时，Shell 将会根据 PATH 变量列举的目录及其顺序，从中检索并执行匹配的可执行程序。如果提交的命令其目录不在检索路径中，必须输入命令的完整路径名才能执行。另外，命令检索路径的顺序是非常重要的。当检索路径中的不同目录存在同名的命令时，Shell 将会使用第一个发现的命令。例如，假定 PATH 变量定义为 PATH=/bin:/usr/bin:/sbin:\$HOME/bin，且存在两个 sample 命令文件，分别位于/usr/bin 和/home/gqxing/bin 目录时，如果输入 sample 命令时未指定完整的路径名，Shell 将会调用/usr/bin 目录下的 sample 命令。定义 PATH 环境变量时，目录之间需加冒号分隔符。通常，PATH 环境变量是由/etc/profile 及\$HOME/.profile 等初始化文件设定的。在 Shell 脚本中，也可以把某个目录临时加到命令检索路径中。一旦终止脚本的运行，立即恢复到之前的 PATH 变量设置（作为一个子进程，Shell 脚本不能改变父进程 Shell 的运行环境）。注意，从安全的角度考虑，\$PATH 通常不应包含当前工作目录
PPID	表示父进程的 PID
PS1	第 1 级 Shell 命令提示符，或称主提示符。变量的默认值为 “[u@h W]\$” ，其中的 “\$” 表示普通用户的命令提示符为 “\$” ，超级用户的命令提示符为 “#” （详见第 13 章）
PS2	第 2 级 Shell 命令提示符，默认值为大于号 “>” 。如果输入的命令不完整，或者需要用户提供附加的数据时，系统将会按此变量的设置提示用户继续输入
PS3	第 3 级命令提示符，其默认值为 “#?” 。这个变量主要用于设置 select 循环控制结构使用的菜单选择提示符（详见第 8 章）
PS4	第 4 级命令提示符，其默认值为加号 “+” 。这个变量主要用作 Shell 脚本的调试标志符，在跟踪脚本执行的过程中，Shell 将会在显示其执行的每一个命令之前，首先输出这个变量定义的值，其默认值为 “+” （参见第 8 章）
SHELL	Shell 命令文件的完整路径名。vi/vim 等工具使用这个变量值作为默认的 Shell
TMOUT	用于定义用户与系统会话过程的超时值。在一个交互式的 Shell 中，TMOUT 变量值可以解释为自输出命令提示符之后等待用户输入的时间（以秒为单位）。Shell 输出命令提示符之后，如果在 TMOUT 规定的时间之内未输入任何命令，Shell 将会因超时而终止执行，导致系统关闭用户的终端窗口，或者断开用户的终端连接。对于 read 内置命令而言，如果 TMOUT 变量值大于 0，这个数值可以用作 read 内置命令默认的超时值。对于 select 内置命令而言，如果在 TMOUT 规定的时间之内一直没有收到输入数据，select 命令将会终止执行

## 7.2.4 变量的引用与替换

使用变量的主要目的是存储或引用其中的数据值。在大型应用程序、Shell 或 Shell 脚本中，经常使用变量，引用变量中的数值。引用变量中的数值称作变量替换。当变量用作参数时，其中的变量替换也称参数替换。

假定 variable 是一个变量，在变量名字前面加上一个美元符号 “\$” 前缀即可引用变量的值，即，使用变量中存储的数值替换变量名字本身。在 Shell 中，引用变量的值（或者说变量替换）主要有以下 3 种形式：

```
$variable;
${variable};
"$variable"或"${variable}"。
```

但在下列情况下，引用变量时无需在变量名字前面加美元符号 “\$” 前缀：

- 声明语句；
- 赋值语句；
- unset、export 命令；
- 引用系统定义的信号。

请注意区分变量的名字及其引用的数值。通常，如果 variable 是变量的名字，那么\$variable、



`${variable}`、`"$variable"`或`"${variable}"`就表示引用变量的值。例如，在执行下列变量赋值语句之后，`echo` 命令中变量引用`$variable` 意味着显示其对应的数值 100。

```
$ variable=100
$ echo $variable
100
$
```

在下列变量设置中，第一个 PATH 变量定义了命令的检索路径，第二个 PATH 变量定义采用变量替换的形式，重新定义了 PATH 变量，把 PATH 变量的值以及用户主目录下的子目录 bin 和当前目录赋值到 PATH 变量中。

```
$ PATH=/bin:/usr/bin:/sbin:/usr/sbin
$ PATH=$PATH:$HOME/bin:.
$
```

执行变量替换后，新的命令检索路径如下。

```
$ echo "Now the PATH is $PATH"
Now the PATH is /bin:/usr/bin:/sbin:/usr/sbin:/home/gqxing/bin:.
```

为了避免引起歧义，可以采用第二种变量替换形式引用变量。实际上，第一种变量替换形式只是第二种变量替换形式的简化。例如，可以把上述 PATH 变量的定义改写如下。

```
PATH=${PATH}:$HOME/bin:.
```

第 3 种变量引用方式是在第一或第二种变量引用方式的基础上加上双引号，这也是一种最保险的变量引用方式。注意，在某些情况下，尤其是传递参数时，第一或第二种变量引用形式并不可靠，有时还会产生意想不到的错误。例如，假定存在一个简单的 Shell 脚本，其功能是显示并处理接收的位置参数，如下所示。

```
$ cat disparg
#!/bin/bash
echo $1
exit 0
$
```

同时又声明了一个 var 变量，其中含有一个由 5 个英文单词组成的字符串，如下所示。

```
$ var="The variable contains five words"
$
```

如果分别采用上述 3 种变量引用形式显示同一变量的值，其输出结果如下。

```
$ disparg $var
The
$ disp ${var}
The
$ disp "$var"
The variable contains five words
$
```

双引号括住的参数可以被看作一个单词，即使其中包含空格分隔符。因此，如果希望把包含多个单词的字符串看作一个参数，应注意使用双引号，以防止单词的分割。

由此可以得出一个结论：当一个变量的值是由一个不包含任何字段分隔符的整体字符串或数值组成时，上述 3 种变量引用方式的最终结果是完全一样的。但是，如果变量的值包含空格、制表符、换行符，以及由内部变量 IFS 定义的字段分隔符时，只有第 3 种变量引用方式才是最保险的。在以后的 Shell 脚本编程过程中，这一点应切实记住。

此外还要注意，使用双引号引用变量时可以进行替换；如果使用单引号，则意味着按文字引用其中的字符串，即使存在变量引用也不能进行替换。单引号表示把特殊字符\$和变量名看作文字，禁止引用其中的变量值，因而不会出现变量替换。也就是说，单引号中的每个字符，包括特殊字

符，均作文字常量解释。

### 7.2.5 变量的间接引用

假定一个变量的值是另一个变量的名字，利用第一个变量是否能够引用第二个变量的值呢？例如，如果 `a=b`，而 `b=c`，仅仅引用变量 `a` 是否能够返回变量 `b` 的值（`c`）呢？回答是肯定的。实际上，采用“`eval var1=\${$var2}`”命令即可达到上述目的。这种情况被称作变量的间接引用。

下面是一个间接引用变量的例子。

```
$ Message=Hello
$ Hello="Good Morning"
$ echo "Message = $Message"      # 直接引用
Message = Hello
$ eval Message=\${$Message}      # 间接引用
$ echo "Now message = $Message"
Now message = Good Morning
$
```

如果把上述的最后两个命令合并到一起，其效果如下。

```
$ Message=Hello
$ Hello="Good Morning"
$ echo Message = $Message
Message = Hello
$ eval echo Now message = \$Message
Now message = Good Morning
$
```

在 Bash 中，还可以采用下列方式实现变量的间接引用。

```
$ Message=Hello
$ Hello="Good Morning"
$ echo ${Message}
Hello
$ echo ${!Message}
Good Morning
$ Hello="Hello again"
$ echo ${!Message}
Hello again
$
```

### 7.2.6 特殊的变量替换

为了保证 Shell、Shell 脚本或应用程序能够正常地运行，Shell 提供了一种特殊的变量替换机制。特殊变量替换的主要功能如下：

对于未设置的变量，采用特殊替换表达式赋予默认值；

采用特殊替换表达式替换或设置默认值；

采用特殊替换表达式给出错误提示信息。

表 7-3 给出了 Shell 支持的部分常用特殊变量替换形式。

表 7-3

特殊的变量替换

特殊变量替换	简 单 说 明
<code> \${var}</code>	其作用同 <code>\$var</code> 一样，表示变量 <code>var</code> 的值。在某些情况下，只有采用 <code> \${var}</code> 变量引用形式，其意义才是比较明确的。例如，在定义 PATH 环境变量时，如果需要连接字符串变量， <code>PATH=\${PATH}:./opt/bin</code> 比 <code>PATH=\$PATH:/opt/bin</code> 更易于阅读和理解
<code> \${var:-value}</code>	如果变量 <code>var</code> 未被设置或其值为 <code>null</code> ，使用 <code>value</code> 作为变量 <code>var</code> 的值进行变量替换。否则，使用变量 <code>var</code> 的值进行变量替换，但变量 <code>var</code> 的值保持不变



续表

特殊变量替换	简单说明
<code> \${var:=value}</code>	如果变量 var 未被设置或其值为 null, 把 value 赋予变量 var, 同时执行变量替换
<code> \${var:+value}</code>	如果变量 var 未被设置或其值为 null, 使用 null 进行变量替换。如果变量 var 已经设置, 则使用 value 进行变量替换, 变量 var 的值保持不变
<code> \${var:?value}</code>	如果变量 var 已经设置, 使用变量 var 的值进行变量替换。如果变量 var 未被设置或其值为 null, 使用 value 作为错误提示信息。如果省略了 value, 则输出默认的错误信息, 表示变量 var 未被设置, 然后, 终止 Shell 脚本的执行, 并返回一个非 0 的出口状态(交互式的 Shell 会话例外)。变量 var 的值保持不变

在上述表达式中, 冒号意味着需要测试给定的变量 var 是否已经被设置。如果变量已经被设置(声明), 再检查其值是否为 null。如果省略了冒号, 则意味着只需检查 var 是否尚未被设置。表 7-4 总结了各种变量替换形式中有无冒号的细微差别之处。

表 7-4 各种变量替换形式的比较

比较	var 已设置且其值为非 null	var 已设置但值为 null	var 未设置
<code> \${var:-value}</code>	使用 var 替换	使用 value 替换	使用 value 替换
<code> \${var-value}</code>	使用 var 替换	使用 null 替换	使用 value 替换
<code> \${var:=value}</code>	使用 var 替换	使用 value 赋值并替换	使用 value 赋值并替换
<code> \${var=value}</code>	使用 var 替换	使用 var 替换	使用 value 赋值
<code> \${var:?value}</code>	使用 var 替换	错误, 退出, 返回 1	错误, 退出, 返回 1
<code> \${var?value}</code>	使用 var 替换	使用 null 替换	错误, 退出, 返回 1
<code> \${var:+value}</code>	使用 value 替换	使用 null 替换	使用 null 替换
<code> \${var+value}</code>	使用 value 替换	使用 value 替换	使用 null 替换

例如, 假定变量 var 已经被设置, 但其值为 null, 则 `${var:-undefined}`变量替换后的结果是 undefined, 示例如下。

```
$ var=
$ echo $var

$ echo var is ${var:-undefined}
var is undefined
$
```

在调用 Shell 脚本时, 如果提供的命令行参数不足, 采用上述默认的变量或参数替换方法很有用。例如, 如果命令行中未给定文件参数, 下列 Shell 脚本中的 command 命令(任何合法的文件操作命令)将会对 run.data 文件进行处理。

```
$ cat substitute
#!/bin/bash
def_fname=run.data
fname=${1:-$def_fname}
command $fname
exit 0
$
```

在下面的例子中, 变量 var 的值为 null。因此,  `${var:=abc}`变量替换后的结果是: 首先把 abc 赋予变量 var, 然后使用变量 var 的值(abc)进行变量替换。

```
$ unset var
$ echo ${var:=abc}
```

```
abc
$
```

在下面的例子中，变量 var 的值为 null，且“?”后面为空。因此，\${var:=abc} 变量替换的结果是输出一条默认的出错信息：“parameter null or not set”。

```
$ unset var
$ echo ${var:?}
bash: var: parameter null or not set
$
```

set 命令可用于设置参数。在下面的例子中，set 命令设置的位置参数 \$1、\$2 和 \$3，其值分别为 a、b 和 c。如果使用 \${3:+posix} 进行变量替换，其结果是 posix。

```
$ set a b c
$ echo $3
c
$ echo ${3:+posix}
posix
$
```

在上述的特殊变量替换形式中，变量名后面的参数不仅可以是任何字符串，还可以是变量替换、命令替换和算术表达式的计算结果。

假定之前并未设置 username 变量，下列变量替换的结果是 “whoami” 命令的输出。

```
$ echo ${username:-`whoami`}
gqxing
$
```

在下面的例子中，只要变量 username 已经被声明，不管其值设置与否，变量替换的最终结果总是 “who am i” 命令输出的第一个字段。

```
$ username=
$ echo ${username+`who am i| cut -d' ' -f1`}
gqxing
$
```

在下面的例子中，仅当变量 var 未被声明，或者即使已经被声明但变量值为 null，才使用 pwd 命令的输出（关于 \$(pwd) 形式的命令替换，参见 7.3 节）替换变量 var 的值。

```
$ echo ${var:-$(pwd)}
/home/gqxing
$
```

表 7-5 给出了其他有用的特殊变量替换形式。

表 7-5

其他特殊变量替换形式

其他特殊变量替换	说 明
\${#var}	字符串的长度（即字符串变量 var 中的字符数量）。对于数组来讲，\${#array} 是数组中第一个元素的长度。注意下列例外情况：\${#*} 和 \${#@} 给出的是位置参数的个数，而针对数组的 \${#array[*]} 和 \${#array[@]} 给出的是数组元素的个数
`\${var#Pattern}`, `\${var##Pattern}`	从 \$var 变量值的前部删除与给定模式匹配的最短或最长部分子串。如果应用于文件路径名，`\${var##Pattern}` 的效果相当于 basename 命令
`\${var%Pattern}`, `\${var%%Pattern}`	从 \$var 变量值的后部删除与给定模式匹配的最短或最长部分子串。其中，`\${var%Pattern}` 可用于抽取路径名的目录部分

在下面的例子中，变量 DOCS 已经被声明，且设置为 /docs posix，变量替换的结果是给出 \$DOCS 变量值的字符串长度。

```
$ DOCS=/docs posix
$ echo ${#DOCS}
11
$
```



在下面的例子中，变量 var 已被设置为 file.c，变量替换的结果是从整个文件名中抽取除 “.c” 后缀之外的文件名部分。

```
$ var=file.c  
$ echo ${var%.*}.o  
file.o  
$
```

在下面的例子中，变量 var 已经被声明并设置为 posix/src/std，变量替换的结果是从路径名后部删除首次出现并包含斜线字符 “/” 的最长字符串。

```
$ var=posix/src/std  
$ echo ${var##*/}  
posix  
$
```

在下面的例子中，变量 var 已被设置为 \$HOME/src/cmd，变量替换的结果是抽取整个路径名除 \$HOME 变量值之外的目录名部分。

```
$ var=$HOME/src/cmd  
$ echo ${var##$HOME}  
/src/cmd  
$ echo ${var##$HOME/}  
src/cmd  
$
```

在下面的例子中，变量 var 被设置为 /one/two/three，变量替换的结果是抽取路径名的文件名部分。

```
$ var=/one/two/three  
$ echo ${var##*/}  
three  
$
```

在下面的例子中，变量 var 被设置为 /one/two/three，变量替换的结果是抽取路径名的目录部分。

```
$ var=/one/two/three  
$ echo ${var%/*}  
/one/two  
$
```

## 7.2.7 变量声明与类型定义

前面说过，Shell 并不严格区分变量的类型，一切取决于变量存储的内容。在第一次引用时，Shell 将根据变量的内容确定变量的类型。为了加快 Shell 的运算速度，提高 Shell 编程的严谨性，在 Bash 中，可以使用 typeset 或 declare 命令定义变量的类型，并在定义时对变量进行初始化。

变量无类型之分既是好事，也是坏事。这虽然增加了脚本编程的灵活性，简化了代码的编写，但也容易产生潜在的错误，容易给自己套上绳索，造成之后阅读的困难，养成不好的编程习惯。

注意，记住脚本变量的类型是 Shell 编程人员的责任，不要指望 Shell 做类型检查。

typeset 内部命令（declare 内部命令的使用说明详见第 8 章）用于设定变量的属性。利用 typeset 命令的 “-i” 选项，可以把变量声明为整数变量。在声明变量的同时，也可以为变量赋值，进行初始化。事先声明为整数的变量可以直接执行算术运算，而不需要使用 expr 和 let 命令，例如下列命令所示。

```
$ typeset -i number      # 把变量 number 声明为整数变量  
$ number=3                # 之后，可以使用整数为变量 number 赋值  
$ echo "Number = $number"  
Number = 3  
$ number=three            # 但不能使用字符串为变量 number 赋值  
$ echo "Number = $number"  
Number = 0  
$
```

如果事先并未把变量声明为整数变量，Shell 将会把赋予变量的值看作字符串，例如下列命令所示。

```
$ n=6/3
$ echo "n = $n"
n = 6/3
$ typeset -i n
$ n=6/3
$ echo "n = $n"
n = 2
$
```

利用“-r”选项还可以把变量声明为只读变量。此后，任何试图修改只读变量的操作将会失败，并产生错误信息。例如，“typeset -r var”将会把变量 var 声明为只读变量。

## 7.3 命令与命令替换

### 7.3.1 Shell 内部命令

Linux 系统提供了大量的命令供用户使用。但出于性能方面的考虑，Shell 也提供了若干具有同样功能的内部命令。内部命令的执行速度比外部命令要快得多。这是因为，在解释执行内部命令时，Shell 不需要创建子进程；而执行外部命令时需要创建单独的新进程，从而加大了系统的开销。

某些 Shell 内部命令与系统命令具有相同的名字，但其实现方式不同。例如，Shell 内部命令 echo 与系统命令 /bin/echo 的名字完全一样，功能也相同，但两者的执行效率大不相同。

例如，下面两条命令的作用都是在终端窗口上显示一条信息。

```
$ echo "This line uses the \"echo\" builtin to output."
This line uses the "echo" builtin to output.
$ /bin/echo "This line uses the /bin/echo system command to output."
This line uses the /bin/echo system command to output.
$
```

表 7-6 给出了 Shell 提供的部分主要内部命令及说明。

表 7-6 Shell 内部命令

内 部 命 令	简 单 解 释
:	用于从命令行中读取 Shell 脚本，并在当前的 Shell 环境中执行。在 Shell 脚本中使用点命令时，可以把指定的源文件读入当前脚本中，并从当前位置开始执行。当多个 Shell 脚本共用同一个数据文件或函数时，点命令是非常有用的。点命令最重要的用途也许就是执行环境变量设置脚本。在此情况下，即使退出环境变量设置脚本，环境变量的设置仍然保持有效。如果直接运行环境变量设置脚本，在脚本终止运行之后，脚本中设置的环境变量也将随之消失而不复存在
alias	用于设置或显示系统或用户定义的命令别名
bg	把指定的作业置入后台运行模式。如果未给定作业号，则把当前作业置入后台运行模式
break	用于退出 for、while 和 until 等循环语句
cd	改变目录。用于转换到指定的目录
continue	用于结束 for、while 和 until 等循环语句中的当前循环，并立即开始执行下一轮循环
declare	用于声明变量或定义变量的属性。如果未指定变量名，declare 命令将会给出所有的变量定义。“-p”选项用于显示给定变量的值和属性。下列选项用于声明变量，并限定变量的属性： <input type="checkbox"/> -a 把指定的变量定义为数组变量； <input type="checkbox"/> -f 用于声明一个函数或显示函数定义； <input type="checkbox"/> -i 把指定的变量定义为整数变量，当赋予变量值时，按算术扩展的要求执行算术运算； <input type="checkbox"/> -r 把指定的变量定义为只读变量，在随后的处理过程，不能为只读变量赋值； <input type="checkbox"/> -x 公布指定的变量，以便其他命令或 Shell 脚本能够继续使用



续表

内部命令	简单解释
echo	用于输出字符串、变量值或表达式的计算结果等参数，通常会在输出信息之后附加一个换行字符
enable	启用和禁用 Shell 内置命令。禁用 Shell 内置命令，使得用户无需指定完整的路径名即可运行与 Shell 内置命令同名的命令，尽管 Shell 内置命令优先于 PATH 变量指定的检索目录中的命令。例如，要运行一个非 Shell 内置命令的 test 命令（/usr/bin/test），可以使用“enable -n test”命令禁用 Shell 内置命令 test
eval	用于读取并组合随后的命令参数，然后提交 Shell 执行
exec	如果 exec 命令带有参数，则使用命令参数代替当前的 Shell 进程，而不是像通常那样创建一个新进程，执行给定的命令。如果在 Shell 脚本中使用 exec 命令，当由 exec 命令引入的命令终止运行时，将会强制 Shell 脚本也随之终止执行。因此，如果在脚本中使用 exec 命令，通常应作为最后一条命令。如果未带参数，exec 命令的作用只是按照 I/O 重定向的要求，修改文件描述符。例如，“exec < fname”命令意味着使用给定的文件 fname 代替标准输入
exit	用于无条件地终止 Shell 脚本的执行。exit 命令可以带一个整数参数，作为 Shell 脚本的出口状态返回 Shell。如果使用单独的 exit 命令（省略整数参数）终止脚本的执行，整个 Shell 脚本的出口状态则是在 exit 之前执行的最后一条命令的出口状态。一个好的编程习惯是在脚本执行结束处增加一条“exit N”命令，表示运行成功或出现的不同错误情况。此外，文件结束标志也会引起 Shell 脚本终止执行
export	用于导出设置的变量，使变量的设置能够用于正在运行的脚本或 Shell 的所有子进程。其一个重要用途是在 profile 文件中初始化环境变量，使随后的所有进程都能够访问
fc	用于列出、编辑或重复执行命令历史缓冲区或文件中记录的命令
fg	把指定的作业置入前台运行模式。如果未给定作业号，则把当前作业置入前台运行模式
getopts	用于解析传递给 Shell 脚本的命令行参数。getopts 使用两个变量：其中的 OPTIND 是指向下一个命令行选项的指针，OPTARG 是与选项有关的参数。getopts 命令通常主要用于 while 循环语句，通过循环方式处理所有的选项和参数
hash	用于记录给定命令的路径名，以便 Shell 或 Shell 脚本随后调用 hash 命令时不必再按 PATH 变量指定的目录检索命令。当调用 hash 命令而未给定参数时，系统仅会显示当前 Shell 散列表中记录的命令
jobs	显示指定或全部活动的作业。其中，“-l”选项表示显示附加的进程 ID 信息，“-n”选项表示仅显示已经停止或终止运行的作业
kill	用于向指定的进程或作业发送信号，从而终止相应的进程或作业。信号可以是一个数字代码或信号名字
let	用于实现算术运算
logout	退出注册 Shell
printf	按照给定的控制格式输出参数的值。输出格式是一个字符串，其中包括 3 种字符对象：一为普通字符，可直接输出到标准输出；二为转义字符序列，用于输出特殊字符；三为格式规范，用于定义参数值的输出形式（参见标准的 printf() 格式定义）
pwd	显示当前的工作目录，其作用等同于读取 PWD 变量的值。换言之，`pwd` 命令替换的结果等同于 PWD 变量的值
read	用于从标准输入读取数据，并把数据赋值到给定的变量中。也就是说，read 命令能够以交互方式读取来自键盘的输入数据。“-r”选项意味着忽略转义符号“\”，使之仅作为普通字符被处理
readonly	把指定的变量设置为只读变量，因而不允许在随后的操作中对相应的变量进行赋值。如果试图修改相应的变量，将会产生一个错误信息
return	引起 Shell 函数返回主程序的调用点。在非 Shell 函数的情况下，return 命令相当于 exit 命令

续表

内部命令	简单解释
set	<p>set 命令用于改变内部变量或脚本变量的值。set 命令的一个重要用途是重置位置参数（例如，<code>set 'command'</code>）。如果不带任何参数，set 命令将会输出所有环境变量的当前设置，这是 set 命令的另一个重要用途。set 命令的部分选项说明如下。</p> <ul style="list-style-type: none"> <li><input type="checkbox"/> -a 使定义的所有变量和函数设置能够自动导入 Shell 运行环境。</li> <li><input type="checkbox"/> -e 如果其中任何一条命令结束后返回非零值出口状态，立即终止 Shell 脚本的执行</li> <li><input type="checkbox"/> -f 禁用文件名生成机制。</li> <li><input type="checkbox"/> -m 监控模式（启用作业控制功能）。在交互式 Shell 中，这个选项的默认设置为 on。</li> <li><input type="checkbox"/> -n 读取 Shell 脚本中的命令，检查是否存在语法错误，但不执行。</li> <li><input type="checkbox"/> -o 使用下列参数设置各种标志。</li> <li>● emacs 使用 emacs 作为命令行编辑器。在交互式 Shell 中，自动启用 emacs 作为默认的命令行编辑器，除非调用 Shell 时指定了“--noediting”选项。</li> <li>● noclobber 防止标准输出 (&gt;) 重定向覆盖现有的同名文件。一旦打开此标志，仅当使用 “&gt; ” 时才能清除现有的文件。</li> <li>● nolog 不允许把函数定义写入命令历史文件中。</li> <li>● verbose 同 “-v” 选项。</li> <li>● vi 使用 vi/vim 作为命令行编辑器。在 vi/vim 命令行编辑器环境中，一开始总是处于输入模式。</li> <li><input type="checkbox"/> 如果不加任何参数，set 命令将会输出当前的各种标志设置。</li> <li><input type="checkbox"/> -t 读入并在执行任何一条命令之后立即终止 Shell 脚本的运行。</li> <li><input type="checkbox"/> -u 执行变量替换时，事先未设置的变量按错误处理。</li> <li><input type="checkbox"/> -v 显示 Shell 读入的命令语句。</li> <li><input type="checkbox"/> -x 显示执行的命令及其参数。</li> <li><input type="checkbox"/> - 关闭 “-x” 和 “-v” 标志。</li> <li><input type="checkbox"/> -- 重新设置位置参数，或者清除位置参数的设置</li> </ul>
shift	把位置参数 \$2, \$3, ……, \$N 依次重新命名为 \$1, \$2, ……, \$N-1。原来的 \$1 和 \$N 不复存在，位置参数总数相应地减 1，故新的 \$# 变量值也比原来的 \$# 变量值少 1
shopt	<p>用于启用或禁用 Shell 控制选项的开关状态，以规范 Shell 的行为。如果未带任何命令选项，或者使用了“-p”选项，将显示用户能够设置的所有 Shell 控制选项及其开关状态。shopt 支持的部分重要命令选项如下：</p> <ul style="list-style-type: none"> <li><input type="checkbox"/> -s 启用相应的控制选项；</li> <li><input type="checkbox"/> -u 禁用相应的控制选项。</li> </ul> <p>利用 shopt 命令可以设置的部分重要 Shell 控制选项如下：</p> <ul style="list-style-type: none"> <li><input type="checkbox"/> cmdhist 如果设置了这个 Shell 控制选项，bash 将会尝试把一个多行的命令作为一个命令行保存到命令历史文件中，以便用户能够容易地重新编辑多行命令，cmdhist 的默认设置为 on；</li> <li><input type="checkbox"/> histappend 设置这个 Shell 控制选项之后，当退出 Shell 时，bash 将会把命令历史缓冲区中的命令历史记录附加到 HISTFILE 变量指定的命令历史文件中，而不是覆盖这个文件，histappend 的默认设置为 off；</li> <li><input type="checkbox"/> huponexit 如果把这个 Shell 控制选项设置为 on，当退出交互式的注册 Shell 时，bash 将会向所有的作业发送一个 SIGHUP 信号，huponexit 的默认设置为 off；</li> <li><input type="checkbox"/> sourcepath 如果设置了这个 Shell 控制选项，“.” 或 source 内置命令将会使用 PATH 变量指定的目录检索指定的脚本文件，sourcepath 的默认设置为 on</li> </ul>
source	类似于句点 “.” 命令，用于读取指定的 Shell 脚本文件，并在当前的 Shell 环境中运行，运行结束后返回最后执行的一条命令的出口状态。如果文件名参数不是绝对路径，使用 PATH 变量指定的目录检索脚本文件。在非 posix 环境中，如果 PATH 变量指定的目录中不存在指定的文件，Bash 将会在当前目录中检索给定的脚本文件。如果 sourcepath 控制选项处于禁用状态，则禁止从 PATH 变量指定的目录中检索指定的文件



续表

内部命令	简单解释
test	计算条件表达式(参见7.5节)
times	显示Shell或进程累计占用的用户和系统时间
trap	trap命令主要用于Shell脚本的信号捕捉与错误处理。当收到指定的信号时，Shell将会读入并执行指定的命令。如果指定的命令为null("")，Shell将会忽略收到的每一个信号。在trap语句定义的处理动作结束之后，其返回值将是调用trap之前执行的最后一个命令的出口状态。如果不希望在执行过程中被Ctrl-C等按键打断，可以在Shell脚本中加入trap命令(详见第8章)
type	type命令用于说明Shell如何解释指定的命令，显示命令的完整路径名，示例如下。 \$ type find find is /usr/bin/find  \$ type echo echo is a shell builtin  \$ type ls ls is aliased to `ls --color=auto'  \$
typeset	用于声明Shell变量或函数的类型属性。其功能及支持的选项完全等同于declare命令
ulimit	用于设置或显示用户的资源限制。资源限制的值可以是一个数字，也可以是文字值unlimited。“-H”和“-S”选项分别用于指定硬性限制和软性限制。一旦设置，硬性限制值不能增加，但软性限制值可以增加，直至达到硬性限制值。如果“-H”或“-S”选项均未指定，则同时适用于软硬性限制。其他选项的意义说明如下(其中“-f”为默认的选项)： <input type="checkbox"/> -a 列出当前所有的资源限制； <input type="checkbox"/> -c 列出内存映像转储文件(core)的最大容量限制(以1KB的数据块为单位)； <input type="checkbox"/> -d 列出进程可用数据区(段)的最大容量限制(以1KB为单位)； <input type="checkbox"/> -e 列出可用的nice优先级调整值的最大值； <input type="checkbox"/> -f 列出可以创建的最大文件的容量限制(以1KB的数据块为单位)； <input type="checkbox"/> -n 列出可用的文件描述符的最大数量限制； <input type="checkbox"/> -s 列出可用的最大栈区的容量限制(以1KB为单位)； <input type="checkbox"/> -t 列出每个进程可用的最大CPU时间量限制(以秒为单位)； <input type="checkbox"/> -u 列出单个用户可用的最大进程数量限制； <input type="checkbox"/> -v 列出可用的虚拟内存的最大数量限制(以1KB为单位)；
umask	指定创建文件时应设定的默认访问权限(参见第5章)
unalias	撤销已经设定的命令别名。“-a”选项意味着撤销当前Shell运行环境中已经定义的所有命令别名
unset	用于清除Shell变量，把变量的值设为null。注意，这个命令并不影响位置参数
wait	等待指定的作业结束，显示作业结束时的出口状态。如果未指定作业号，意味着等待当前作业及其所有子进程的结束。例如，“wait %1”意味着等待作业号为1的后台进程执行结束，“wait 6618”意味着等待进程ID为6618的后台作业完成

### 7.3.2 部分命令介绍

#### 1. “.”、source与exec命令

当Shell运行一个Shell脚本时，通常需要创建一个新的Shell进程，新的Shell进程将会继承其父Shell进程的环境变量(包括全局变量和已公布的变量)，但不会继承未公布的本地变量。在

Shell 脚本中设置的变量，不管公布与否，只要退出 Shell 脚本，其中设置的变量将随之消失。为了利用 Shell 脚本设置应用程序可用的变量，通常需要使用“.”或 source 内部命令，在当前的 Shell 环境中解释执行 Shell 脚本。

与“.”或 source 命令类似，exec 也在当前进程的进程空间中执行指定命令。但与“.”或 source 命令不同的是，“.”或 source 命令只能运行 Shell 脚本，而 exec 既可以运行 Shell 脚本，也可以运行编译的程序或命令。而且，在“.”或 source 命令调用的 Shell 脚本运行结束之后，将会返回调用点，但 exec 则不返回。此外，“.”或 source 命令运行的 Shell 脚本能够访问当前进程空间中的本地变量。

exec 内部命令主要有两个用途：运行指定的命令时无需创建新的进程，重定向 Shell 脚本中的文件描述符。由于 exec 在运行指定的命令时并不创建新的进程，因而命令的执行速度更快。但是，由于 exec 命令并不返回到调用位置，因此只能用作 Shell 脚本中最后执行的一个命令。例如，在下列 Shell 脚本中，最后一个 echo 命令是不会执行的。

```
$ cat exec_demo
uname -n
exec date
echo "This line is never displayed."
$ exec_demo
iscas
2008年 11月 08日 星期六 01:01:58 CST
$
```

exec 命令的第二个主要用途是重定向 Shell 脚本中的文件描述符，包括标准输入、标准输出和标准错误输出。如果 Shell 脚本中的某个位置出现下列命令，在随后读取标准输入时，其数据均取自 exec 命令指定的文件 infile。

```
exec < infile
```

同样，下列 exec 命令将会把 Shell 脚本中随后命令的标准输出和标准错误输出分别重定向到指定的文件 outfile 和 errfile。

```
exec > outfile 2> errfile
```

当以上述方式使用 exec 命令时，不会替代当前进程，exec 命令之后仍然能够附加其他命令。

在使用 exec 命令重定向标准输出和标准错误输出之后，为了使用户能够看到 Shell 脚本中输出的任何提示信息，可以使用 /dev/tty 设备文件克服这一矛盾。/dev/tty 是一个伪设备文件，表示用户当前使用的终端窗口（和键盘）。无论何时，都可以使用这个设备文件引用用户的终端窗口，而不必知道当前用户终端的实际设备文件名是什么（如果需要，可以使用 tty 命令获取用户终端的实际设备文件名）。

把 Shell 脚本中的标准输出重定向到 /dev/tty，可以保证任何提示信息都能送达用户的终端窗口，而无需顾及用户究竟使用哪一个终端注册到 Linux 系统。另外，即使整个 Shell 脚本的标准输出和标准错误输出均重定向到某个文件，其间发送到 /dev/tty 的输出信息也不会受影响。

例如，下列 3 个 echo 命令分别把输出信息送到标准输出、标准错误输出和用户的终端窗口，当重定向整个 Shell 脚本的标准输出和标准错误输出时，发送到 /dev/tty 的输出信息仍然能够被送达用户的终端窗口，而 outfile 和 errfile 则分别存有发送到标准输出和标准错误输出的信息。

```
$ cat exec_demo2
echo "message to standard output"
echo "message to standard error" 1>&2
echo "message to the user" > /dev/tty
$ exec_demo2 >outfile 2> errfile
message to the user
```



```
$ cat outfile  
message to standard output  
$ cat errfile  
message to standard error  
$
```

下列 exec 命令直接把 Shell 脚本的标准输出重定向到用户的终端窗口（实际上，这种重定向有点画蛇添足）。

```
exec > /dev/tty
```

如果把上述 exec 命令加到第一行，创建新的脚本文件 exec\_demo3，则由于已经把 Shell 脚本中的标准输出重定向到 /dev/tty，此时，即使再把 Shell 脚本本身重定向到另一个文件，发送到标准输出的信息仍然会被输出到用户的终端窗口。但是，发送到标准错误输出的信息则不受影响。

```
S cat exec_demo3  
exec > /dev/tty  
echo "message to standard output"  
echo "message to standard error" 1>&2  
echo "message to the user" > /dev/tty  
S exec_demo3 >outfile 2> errfile  
message to standard output  
message to the user  
S cat outfile  
S cat errfile  
message to standard error  
S
```

在 Shell 脚本中，采用 exec 命令重定向的一大优点是，不必在随后需要重定向的每个命令中一一进行 I/O 重定向。

除了表示用户的终端窗口，/dev/tty 也表示终端的键盘输入。如果在 Shell 脚本中使用下列 exec 命令，意味着读取用户终端的键盘输入（实际上，这种重定向也属画蛇添足）。之后，读取标准输入的所有命令均需接收来自终端键盘的输入。

```
exec < /dev/tty
```

## 2. “:”与 true 命令

实际上，“:”与 true 命令并不执行任何处理动作，其作用只是返回一个出口状态为 0 的测试条件。尽管两个命令的功能一样，但与“:”不同的是，true 命令是 Linux 系统提供的一个外部命令。这两个命令经常用于实现不定循环，如用作 while 循环结构的无限循环测试条件。显然，循环语句中还应增加一个 break 语句，以便当某个条件满足时能够退出循环。当然，也可以使用 Ctrl-C 键等方式强行中断循环语句的运行。

例如，当需要连续地观察一个日志文件或备份文件的大小是否按预期的那样不断增长时，可以使用下列命令，每 5 秒钟显示一次。

```
$ while true  
> do  
> ls -l backupfile  
> sleep 5  
> done  
-rw-r--r-- 1 gqxing gqxing 11111 2008-11-7 17:18 backupfile  
-rw-r--r-- 1 gqxing gqxing 22222 2008-11-7 17:18 backupfile  
-rw-r--r-- 1 gqxing gqxing 33333 2008-11-7 17:18 backupfile  
^C  
$
```

## 3. echo 命令

echo 命令也许是 Linux 系统中应用最广泛的命令之一。echo 命令主要用于显示各种信息，也

可用于显示文件列表。作为显示信息的字符串，可以直接写在 echo 语句后面，也可以在字符串前后加引号，如下所示。

```
$ echo Please enter your choice:  
Please enter your choice:  
$ echo *.c  
atmcom.c atmmon.c atmstat.c handler.c listerner.c  
$
```

为了组织信息的显示格式，利用“-e”选项，echo 语句还支持少量的特殊字符，兹列举如下：

- \n 表示在相应的位置插入一个换行符；
- \t 表示在相应的位置插入一个制表符；
- \b 表示在相应的位置插入一个退格符号；
- \c 在输出所有的字符串参数之后，echo 语句通常还会输出一个换行符。使用 “\c” 意味着取消输出换行符。此外，利用 echo 命令的 “-n” 选项也能够达到同样的目的。

#### 4. read 命令

read 语句的主要功能是读取来自标准输入的数据，然后将其存储到指定的变量参数中，实现交互式的变量赋值。read 命令的语法格式简写如下（表 7-7 给出了 read 命令的部分选项及说明）。

```
read [options] vars
```

表 7-7 read 命令选项及其说明

选 项	简 单 说 明
-a <i>aname</i>	读取每一个字段（字或字符串），并依次赋予指定数组 <i>aname</i> 中的每一个元素
-d <i>delim</i>	使用指定的分隔符（而不是默认的换行符）作为输入数据的终止符
-e	如果输入来自键盘，则使用 Readline 库获取输入数据
-n <i>nchars</i>	读取指定数量 ( <i>nchars</i> ) 的字符后立即返回。只要用户输入了 <i>nchars</i> 个字符，无需按下 Enter 键，read 命令将会立即返回。如果输入的字符数量少于 <i>nchars</i> ，仅当按下 Enter 键，read 命令才会响应；否则，read 命令将一直处于等待状态
-p <i>prompt</i>	首先在标准错误输出中显示一个提示字符串（无换行，光标停留在提示字符串最后一个字符之后），然后等待用户输入。这个选项仅适用于从键盘中读取用户输入的数据
-s	禁止显示输入的任何字符，可用于输入密码数据
-t <i>timeout</i>	用于控制输入超时。如果在指定的时间（秒）内无法读取一个完整的输入行，将会引起 read 命令超时，并返回一个错误状态信息。如果不是从一个用户终端或管道读取输入数据，这个选项不起任何作用
-u <i>fd</i>	使用指定的整数 <i>fd</i> 作为文件描述符，以便 read 命令能够从指定的文件中读取数据。例如： <code>read -u 4 var1 var2</code> 等价于 <code>read var1 var2 &lt;&amp;4</code>

在实际应用中，为了在运行时直接从用户终端中读取数据，并为变量赋值，可在 Shell 脚本中增加下列 read 命令。

```
$ cat install
-----
echo "Enter a Serial number, then press <ENTER>."
read SNO
-----
$
```

如果 read 命令后面列有多个变量参数，输入的数据将按空格（IFS 变量定义的字段分隔符，



默认情况下为空格) 分隔的单词顺序依次为每个变量赋值。如果输入的数据多于变量参数，多余的数据将会全部赋予最后一个变量参数，如下所示。

```
$ cat readdata
echo -e "Please enter some data: \c"
read word1 word2 word3
echo "Word 1 is: $word1"
echo "Word 2 is: $word2"
echo "Word 3 is: $word3"
$ readdata
Please enter data: This is something
Word 1 is: This
Word 2 is: is
Word 3 is: something
$ readdata
Please enter data: This is something else
Word 1 is: This
Word 2 is: is
Word 3 is: something else
$
```

使用“-p”选项，可以省略 echo 语句，在 read 语句中直接增加提示信息。因此，上述代码的前两行可以合并为一个 read 语句，如下所示。

```
read -p "Please enter some data: " word1 word2 word3
```

read 命令的标准输入也可以重定向到一个文件。如果文件中含有多行数据，只有第一行数据将会赋值到变量中。但是，如果利用循环语句，可以依次读取文件的每一行数据。如果成功地读取了任何输入数据，read 命令将会返回一个值为 0 的出口状态。如果遇到文件结束标志，即读取了 EOF (End Of File) 标志字符，read 命令将会返回一个非 0 的出口状态。因此，可以使用 read 命令的这一特性作为循环结束的判断条件。下面是一个利用管道实现文件 I/O 重定向，从中读取数据且为变量赋值的例子。

```
$ cat install2
#!/bin/bash
while read line
do
    set $line
    if [ "$3" = "swap" ]
    then
        .....
    fi
done < /etc/fstab
$
```

利用“-s”、“-l”以及“-n”等选项，还可以在执行 read 命令时禁止回显键盘输入的数据，控制输入超时，以及读取指定数量的字符，而不必非按 Enter 键不可。例如，下列 read 命令用于读取单个字符：

```
$ read -s -n1 -p "Hit a key" keypress
$ echo "Keypressed was \"\$keypress\"."
```

其中，“-s”选项意味着不回显输入的数据。“-n1”选项意味着仅仅接受 1 个输入字符。

## 5. set 与 unset 命令

set 命令的一个重要用途是修改或重新设置位置参数的值。按照 Shell 的规定，用户不能直接为位置参数赋值，但是，利用 set 命令及其参数变量，则可以修改或重新设置位置参数的值，示例如下。

```
$ cat reset
echo $1 $2 $3 $4
set value1 value2
echo $1 $2 $3 $4
$ reset This is an argument
```

```
This is an argument
value1 value2
$ cat retain
echo $1 $2 $3 $4
set $1 value1 value2 $4
echo $1 $2 $3 $4
$ retain This is an argument
This is an argument
This value1 value2 argument
$
```

使用“set --”命令，也可以把参数变量的值赋予位置参数。如果“set --”命令后面未给定参数变量，意味着清除所有的位置参数。下面是一个重新分配位置参数的例子。

```
$ var="one two three four five"
$ set -- $var
$ echo $1 $2 $3 $4 $5
one two three four five
$ echo "$@"
one two three four five
$ set --          # 清除当前的所有位置参数
$ echo "$1"
$ echo "$@"

$
```

实际上，当使用给定的参数变量设置位置参数时，双减号“--”并非必需的，因而可以省略。下面是一个根据“uname -a”命令的输出，利用 set 命令设置位置参数的例子。

```
$ set 'uname -a'
$ echo "Positional parameters after set `uname -a` :"
Positional parameters after set `uname -a` :
$ echo "Field 1 = $1"
Field 1 = Linux
$ echo "Field 2 = $2"
Field 2 = iscas
$ echo "Field 3 = $3"
Field 3 = 2.6.24-21-generic
$
```

如果不带任何选项或参数，set 命令将会列出所有的环境变量、已经声明或设置的其他变量及函数定义等。这是 set 命令的另一个重要用途，示例如下。

```
$ set
.....
HOME=/home/gqxing
HOSTNAME=iscas
LANG=zh_CN.utf8
.....
$
```

unset 命令用于清除 Shell 变量，把变量的值设置为 null。注意，这个命令并不影响位置参数，示例如下。

```
$ variable=hello
$ echo "$variable"
hello
$ unset variable
$ echo "$variable"
$
```

## 6. set 与 shopt 命令

Shell 既是一个命令解释程序，提供用户与 Linux 系统的命令行界面，又是一个普通的系统命令，



如不特别指定，Shell 将会按照默认的环境设置运行。在 Shell 的运行过程中，如果需要，也可以使用 set 或 shopt 命令修改 Shell 的默认处理行为，定制自己的运行环境。为了启用 Shell 的某种功能特性，可以使用 set 命令的“-o”选项，而 set 命令的“+o”选项则用于关闭相应的功能特性。如果不加任何选项和参数，“set -o”命令将会显示 set 命令能够控制的每一个特性参数及其开关状态。例如，为了启用 noclobber 特性，防止重定向操作无意中覆盖同名的文件，可以使用下列命令。

```
$ set -o noclobber  
$
```

要关闭这一功能特性（默认），可以使用下列命令。

```
$ set +o noclobber  
$
```

shopt 是 Bash 中新增的一个内置命令。用于启用、关闭或显示 Shell 的部分功能特性。其中，“-s”选项用于启用指定的功能特性，“-u”选项用于关闭指定的功能特性。如果不加任何选项和参数，shopt 命令将会显示 shopt 命令能够控制的每个功能特性及其开关状态。例如，采用下列 shopt 命令设置 Shell 的命令历史机制特性，能够把多行命令合并为一个命令项，在每个命令语句或关键字之间增加一个适当的分号“;”分隔符。

```
$ shopt -s cmdhist  
$
```

要关闭这一功能特性，可以使用下列 shopt 命令。

```
$ shopt -u cmdhist  
$
```

要查询某一功能特性的开关状态，可以使用下列 shopt 命令。

```
$ shopt cmdhist  
cmdhist      on  
$
```

## 7. expr 命令

expr 命令用于计算表达式的值，然后把计算结果送到标准输出。其中的表达式可以是字符串比较表达式、整数算术运算表达式或模式匹配表达式。该命令的语法格式简写如下。

```
expr expression
```

表 7-8 给出了 expr 命令支持的各种字符串比较表达式，以及计算结果的说明。expr 命令基本上已被 test 命令或 “[...]" 结构所取代，且在 “[...]" 结构中，“>” 和 “<” 符号之前也无需加转义符号，故现在很少使用 expr 命令做字符串比较（但是，test 命令和 “[…]” 结构不支持 “>=” 和 “<=” 比较）。

表 7-8 expr 命令支持的字符串比较表达式

表达式	计算结果
<code>str1 = str2</code>	如果字符串 str1 等于 str2，则计算结果为真，同时输出 1，但返回值为 0；反之，如果计算结果为假，则输出 0，但返回值为 1
<code>str1 &gt; str2</code>	如果字符串 str1 大于 str2，则计算结果为真，同时输出 1，但返回值为 0；反之，如果计算结果为假，则输出 0，但返回值为 1
<code>str1 &lt; str2</code>	如果字符串 str1 小于 str2，则计算结果为真，同时输出 1，但返回值为 0；反之，如果计算结果为假，则输出 0，但返回值为 1
<code>str1 &gt;= str2</code>	如果字符串 str1 大于等于 str2，则计算结果为真，同时输出 1，但返回值为 0；反之，如果计算结果为假，则输出 0，但返回值为 1
<code>str1 &lt;= str2</code>	如果字符串 str1 小于等于 str2，则计算结果为真，同时输出 1，但返回值为 0；反之，如果计算结果为假，则输出 0，但返回值为 1
<code>str1 != str2</code>	如果字符串 str1 不等于 str2，则计算结果为真，同时输出 1，但返回值为 0；反之，如果计算结果为假，则输出 0，但返回值为 1

下面的例子说明了怎样利用 expr 命令测试字符串表达式。

```
$ TIMEZONE=CST
$ expr "$TIMEZONE" = "CST"
1
$ echo $?
0
$ expr "$TIMEZONE" = "GMT"
0
$ echo $?
1
$
```

expr 命令也可用于计算整数表达式的值，将计算结果发送到标准输出。因此，如果想把计算结果值保存到一个变量中，需要采用命令替换的方式实现。

表 7-9 给出了 expr 命令支持的各种整数算术运算表达式，以及有关计算结果的说明。expr 命令仅支持整数表达式的计算，如果给定的参数不是整数，expr 命令将会输出一个出错信息。

表 7-9 expr 命令支持的整数算术运算表达式

表达式	简单说明
$exp1 + exp2$	计算整数表达式 $exp1$ 与 $exp2$ 的和
$exp1 - exp2$	计算整数表达式 $exp1$ 与 $exp2$ 的差
$exp1 \* exp2$	计算整数表达式 $exp1$ 与 $exp2$ 的乘积。注意，星号前应加转义符号 “\”
$exp1 / exp2$	计算整数表达式 $exp1$ 与 $exp2$ 的商
$exp1 \% exp2$	计算整数表达式 $exp1$ 与 $exp2$ 的余数

下面的例子说明了怎样利用 expr 命令计算整数表达式，以及怎样利用命令替换的方式保存 expr 命令的计算结果。

```
$ n=3
$ expr $n + 7
10
$ n='expr $n + 1'
$ echo $n
4
$
```

#### 8. let 命令与 “((...))” 结构

let 命令和双括号 “((...))” 结构用于计算和测试整数算术表达式，执行整数算术运算。实际上，let 命令和 “((...))” 结构是对 expr 命令的简化，取代并扩展了 expr 命令的整数算术运算功能。

除了上述 5 种算术运算之外，let 命令和 “((...))” 结构还支持 +=、-=、 \*=、 /= 和 %= 等运算符。此外，在 let 语句中，表示乘法运算的符号 “\*” 之前无需增加转义符号。表 7-10 给出了 let 命令和 “((...))” 结构支持的各种整数算术运算及其简单说明。

表 7-10 let 命令和 “((...))” 结构支持的算术运算

运算符	简单说明
$exp1 + exp2$	计算整数表达式 $exp1$ 与 $exp2$ 的和
$exp1 - exp2$	计算整数表达式 $exp1$ 与 $exp2$ 的差
$exp1 * exp2$	计算整数表达式 $exp1$ 与 $exp2$ 的乘积
$exp1 / exp2$	计算整数表达式 $exp1$ 与 $exp2$ 的商



续表

运 算 符	简 单 说 明
<code>exp1 % exp2</code>	计算整数表达式 <code>exp1</code> 与 <code>exp2</code> 的余数。可用于生成位于某个范围内的数字
<code>var += exp</code>	组合加运算符，把表达式的值加到变量中。例如，执行 <code>let "var += 5"</code> 之后，变量 <code>var</code> 的值将增加 5
<code>var -= exp</code>	组合减运算符，从变量中减去表达式的值。例如，执行 <code>let "var -= 3"</code> 之后，变量 <code>var</code> 的值将减少 3
<code>var *= exp</code>	组合乘运算符，计算变量与表达式的乘积，再把结果赋值到变量中。例如，执行 <code>let "var *= 4"</code> 之后，变量 <code>var</code> 的值将扩大 4 倍
<code>var /= exp</code>	组合除运算符，把变量除以表达式后的商再赋值到变量中。例如，执行 <code>let "var /= 2"</code> 之后，变量 <code>var</code> 的值将缩小 2 倍
<code>var %= exp</code>	组合模运算符，把变量除以表达式后的余数再赋值到变量中

下面以不同的运算符说明怎样利用 `let` 命令计算整数表达式。

```
$ n=1
$ let "n = $n + 1"
$ echo "$n"
2
$ let "n = n + 1"
$ echo "$n"
3
$ let n=$n+1
$ echo "$n"
4
$ let n=n+1
$ echo "$n"
5
$ (( n = n + 1 ))
$ echo "$n"
6
$ let "n += 1"
$ echo "$n"
7
$ let n+=1
$ echo "$n"
8
$
```

`let` 命令和`((...))`结构也可以返回算术表达式计算结果的出口状态。如果算术表达式的计算结果为 0，出口状态为 1；如果计算结果为非 0 值，出口状态为 0。这一点与 `test` 语句、“`[...]`”和“`[[...]]`”结构恰好相反，但其比较运算的判断逻辑是一致的，参见下面的例子。

```
$ (( 5-5 ))
$ echo "The exit status is $?"
The exit status is 1
$ (( 5+5 ))
$ echo "The exit status is $?"
The exit status is 0
$ (( 5/5 ))
$ echo "The exit status is $?"
The exit status is 0
$ (( 5 > 4 ))
$ echo "The exit status is $?"
The exit status is 0
$ (( 5 > 9 ))
$ echo "The exit status is $?"
The exit status is 1
$ let "1<2"
$ echo "The exit status is $?"
The exit status is 0
```

```
$ let "5<2"
$ echo "The exit status is $?"
The exit status is 1
$
```

注意，在 let 语句中，如果未加双引号，表达式之间不能有空格。此外，为了加快 Shell 的运算速度，提高 Shell 编程的严谨性，事先最好利用 typeset 命令定义变量的类型，并对变量进行初始化。例如，下面的例子首先利用 typeset 命令把变量 z 定义为整数变量，同时初始化为 0，然后采用 let 命令和 “((...))” 结构进行计算。

```
$ typeset -i z=0
$ let z=z+1
$ let "z += 1"
$ z=$((z+1))
$ z=$((z+1))
$ echo $z
4
$
```

### 9. 数值常数

Shell 脚本按十进制数值解释字符串中的数字字符，除非数字前有特殊的前缀或记号。如果数字前面有一个数值零（0），表示这是一个八进制的数，0x 或 0X 表示一个十六进制的数。BASE#NUMBER 表示以 BASE (2~64) 为底数，以 NUMBER 为指数的数值。

```
$ let "dec = 32"          # 默认的十进制数值
$ echo "decimal number = $dec"
decimal number = 32
$ let "oct = 032"         # 八进制数值
$ echo "octal number = $oct"
octal number = 26
$ let "hex = 0x32"        # 十六进制数值
$ echo "hexadecimal number = $hex"
hexadecimal number = 50
$ let "bin = 2#111100111001101"    # 二进制数值
$ echo "binary number = $bin"
binary number = 31181
$
```

### 7.3.3 命令替换

命令替换的目的是获取命令的输出，为变量赋值，或者对命令的输出做进一步的处理。命令替换的实现有两种方法：一是使用反向引号引用命令，二是采用 “\$(...)" 形式引用命令。其语法格式简写如下。

'command'

或

\$ (command)

例如，为了获取 date 命令的输出，并把输出结果赋予 today 变量，可以使用下列命令替换形式。

```
$ today=`date`
$ echo $today
2008 年 11 月 08 日 星期六 10:49:36 CST
$
```

下面的例子是利用第二种命令替换形式获取 uname 命令输出系统的主机名，并把结果赋予 machine 变量。

```
$ machine=$(uname -n)
$ echo $machine
iscas
$
```



原则上，命令替换中的命令可以是任何命令，包括外部命令、Shell 脚本甚至 Shell 脚本中定义的函数。严格地讲，命令替换实际上蕴含着两个过程：执行相应的命令，获取命令标准输出中的数据。在获取命令的输出数据之后，可以使用赋值运算符 (=)，把数据赋予某个变量，或者做进一步的处理。

命令的输出也可以用作另一个命令的输入参数，甚至可用于生成 for 循环中的参数表（参见第 8 章）。例如，如果文件 filename 中包含需要删除的文件列表，使用下列命令可以删除文件 filename 中指定的所有文件。

```
$ rm -rf `cat filename`  
$
```

注意，在执行上述命令时，如果出现类似于“arg list too long”的出错信息，最好改用“xargs rm -- < filename”命令。

不管采用哪一种命令替换形式，其最终效果是完全一样的。下面两个例子显示了分别采用两种命令替换形式的输出结果。

```
$ cfile=`ls *.c`  
$ echo $myfile  
atmcom.c atmmon.c atmstat.c handler.c listerner.c  
$ cfile2=$(ls *.c)  
$ echo $myfile2  
atmcom.c atmmon.c atmstat.c handler.c listerner.c  
$
```

注意，在使用 echo 命令输出一个未加引号引用的变量，而相应的变量又是采取命令替换的形式赋值时，将会抛弃命令输出数据后面的换行符，从而有可能导致不期望的结果。在下列例子中，我们希望显示一个文件列表，但实际的输出数据却是如下所示。

```
$ dir_listing=`ls -l /etc/rpm`  
$ echo $dir_listing      # 未使用引号  
总用量 12 -rw-r--r-- 1 root root 244 2007-03-05 14:38 anacron -rw-r--r-- 1 root  
root 492 2008-05-10 00:38 php5 -rw-r--r-- 1 root root 582 2008-02-13 01:15 sysstat  
$
```

这显然不是我们预期的输出结果。如果在引用变量时增加一对双引号，其输出内容则恰是我们希望看到的结果，如下所示。

```
$ echo "$dir_listing"      # 使用引号  
总用量 12  
-rw-r--r-- 1 root root 244 2007-03-05 14:38 anacron  
-rw-r--r-- 1 root root 492 2008-05-10 00:38 php5  
-rw-r--r-- 1 root root 582 2008-02-13 01:15 sysstat  
$
```

在命令替换中，使用 I/O 重定向的方式或 cat 命令，还可以把文件的全部内容赋予一个变量。但在实际的应用过程中，把一个较大文本文件的内容赋予一个变量并非必要，尤其不应当把一个二进制文件的内容赋予变量。例如，如果确实需要，可以使用下列命令把文件/etc/timezone 的内容赋予变量 var1。

```
$ cat /etc/timezone  
Asia/Shanghai  
$ var1=`cat /etc/timezone`  
$ echo $var1  
Asia/Shanghai  
$
```

如果稍作变换，可以把上述命令替换改写如下（实际上，可以把“< fname”看作“cat fname”命令的一个特例）。

```
$ var2='< /etc/timezone'
$ echo $var2
Asia/Shanghai
$
```

注意，采用命令替换的形式为变量赋值时，变量中可能包含空格，甚至包含控制字符。

## 7.4 test 语句

每一种编程语言都具有条件测试功能，而条件测试的结果将决定程序的控制流向和下一步的处理动作。通过 test 命令及其简化形式（两种方括号测试结构），Shell 编程也支持这一功能。test 语句与 if/then 和 case 结构语句一起，构成了 Shell 编程的控制转移结构。

通常，Shell 执行的 test 命令是 Shell 自己提供的内部命令。在执行 test 语句时，如果不特别指定，Shell 不会调用外部的系统命令/usr/bin/test。

test 命令（及其简化形式）的主要功能是计算随后的表达式，如检查文件的属性，比较字符串，比较字符串表示的整数值，等等，然后以表达式的计算结果作为 test 命令的出口状态。如果测试条件为真，test 命令将会返回 0，否则返回一个非 0 数值。

test 命令经常用于控制 Shell 脚本的执行流向。test 语句中的选项与参数或表达式描述了测试的条件。测试条件可以是文件属性检查、字符串比较或整数值比较。在计算 test 语句中的表达式之前，Shell 首先执行变量替换或命令替换，然后再执行条件测试。

在 Bash 中，test 命令的语法格式主要有下列 3 种形式：

```
test expression;
[ expression ];
[[ expression ]].
```

“[...]" 是一种简化的但效率更高的 test 命令，而 “[...]" 结构则是一种比 “[...]" 结构更通用的测试结构，也是扩展的 test 命令。同 test 命令一样，这种测试结构用于计算括号内的表达式，测试文件的属性，或者比较字符串及其相应的整数值。然后再根据计算和测试的结果，返回相应的出口状态（0 或 1）。注意，方括号内侧的两边必须各加一个空格。

例如，在删除文件时，为了避免出错，可以先测试文件是否存在；如果文件确实存在，然后再执行文件删除操作，如下所示。

```
$ fname=/tmp/somefile
$ if test -e $fname
> then
> rm -f $fname
> echo "File $fname has removed."
> fi
File /tmp/somefile has removed.
$
```

采用 “[...]" 测试结构，可把上述代码片段改写如下。

```
$ fname=/tmp/somefile
$ if [ -e $fname ]
> then
> rm -f $fname
> echo "File $fname has removed."
> fi
File /tmp/somefile has removed.
$
```



在 Shell 脚本中，使用 “[...]]” 测试结构，而不用 “[...]” 结构，有时能够避免出现逻辑错误。例如，在 “[...]]” 测试结构中，可以使用 “&&”、“||”、“<” 和 “>” 等运算符，但如果在 “[...]” 测试结构中使用上述运算符就会出错，示例如下。

```
$ fname=/tmp/somefile
$ flag=yes
$ if [[ -e $fname && "$flag" = "yes" ]]
> then
> rm -f $fname
> echo "File $fname has removed."
> fi
File /tmp/somefile has removed.
$
```

注意，“[...]]” 测试结构不允许执行文件名生成和单字解析操作。

## 7.4.1 文件测试运算符

文件测试主要指文件的状态和属性测试，如文件是否存在、文件的类型、文件的访问权限以及其他属性等。

文件的访问权限分为 3 种类型：文件属主、同组用户和其他用户。如果 Shell 脚本的执行者是文件的属主，则检查相应于文件属主的访问权限；如果执行者不是文件属主，而是与文件属主同组的成员，则检查相应于同组用户的访问权限；如果执行者既非属主，又非同组成员，则检查相应于其他用户的访问权限。

在 test 语句中，文件名参数必须显式地被指定，不能为空。文件名参数可以是实际的文件名，也可以是变量替换、命令替换或文件名生成机制等的最终结果。如果文件名参数是由替换方式生成的，则必须使用双引号括起来。

表 7-11 列出了 test 语句中常用的部分文件测试表达式。

表 7-11

文件属性测试表达式

表达式	简单说明
-e file	如果给定的文件存在，则条件测试的结果为真
-r file	如果给定的文件存在，且其访问权限是当前用户可读的，则条件测试的结果为真
-w file	如果给定的文件存在，且其访问权限是当前用户可写的，则条件测试的结果为真
-x file	如果给定的文件存在，且其访问权限是当前用户可执行的，则条件测试的结果为真
-s file	如果给定的文件存在，且其大小大于 0，则条件测试的结果为真
-f file	如果给定的文件存在，且是一个普通文件（非目录或设备文件），则条件测试的结果为真
-d file	如果给定的文件存在，且是一个目录，则条件测试的结果为真
-L file	如果给定的文件存在，且是一个符号连接，则条件测试的结果为真
-c file	如果给定的文件存在，且是字符特殊文件（如终端等），则条件测试的结果为真
-b file	如果给定的文件存在，且是块特殊文件（如磁盘等），则条件测试的结果为真
-p file	如果给定的文件存在，且是命名的管道文件，则条件测试的结果为真
-u file	如果给定的文件存在，且其 setuid 标志位已经设置，则条件测试的结果为真。如果文件属主为超级用户的可执行文件已经设置了 setuid 标志，即使是普通用户，在执行期间也具有超级用户的特权。对于某些可执行文件而言，这是非常有用的。例如，passwd 命令需要更新系统文件/etc/shadow。如果没有设置 setuid 标志位，普通用户就无法使用 passwd 命令修改密码。要了解一个文件是否设置了 setuid 标志位，可以使用下列命令显示相应的文件，检查其文件属主访问权限字段是否有一个字符“s”。 \$ ls -l /usr/bin/passwd -rwsr-xr-x 1 root root 32988 2008-12-08 17:17 /usr/bin/passwd \$ 注意，设置 setuid 标志位会留下安全隐患。对 Shell 脚本来讲，setuid 标志位没有影响

续表

表达式	简单说明
-g file	如果给定的文件存在，且其 setgid 标志位已经设置，则条件测试结果为真。如果某个目录设置了 setgid 标志位，则在该目录中创建的文件属于同组成员。这对于工作组成员共享同一目录是非常有用的。
-k file	如果给定的文件存在，且其粘性标志位已经设置，则条件测试结果为真。粘性标志位是一种特殊的文件访问权限。如果某个可执行文件设置了粘性标志位，当调度相应的程序文件执行时，在整个运行过程中，相应的程序将始终保持在高速缓存中，访问和执行的速度因而更快捷。如果设置的是目录，将限制用户的访问权限。如果一个目录设置了粘性标志位，在使用下列命令显示相应的文件或目录时，其访问权限字段将会出现一个字符“t”。 <pre>\$ ls -ld /tmp drwxrwxrwt 12 root root 4986 2009-01-10 17:40 /tmp \$</pre> <p>如果某个共享目录已经设置了粘性标志位，且任何用户均具有写的访问权限，则用户只能删除目录中属于自己的文件。这将防止用户无意中覆盖或删除共享目录下其他用户拥有的文件。注意，在 Linux 系统中，粘性标志位仅适用于目录。</p>
f1 -nt f2	如果给定的文件 f1 存在，且其修改日期比文件 f2 新，则条件测试的结果为真
f1 -ot f2	如果给定的文件 f1 存在，且其修改日期比文件 f2 早，则条件测试的结果为真
f1 -ef f2	如果给定的文件 f1 和 f2 存在且指向同一个物理文件，则条件测试的结果为真
!	逻辑非运算符。当与上述表达式一起使用时，测试结果与其本意恰好相反

## 7.4.2 字符串测试运算符

表 7-12 给出了 test 语句中常用的字符串测试表达式。

表 7-12 字符串测试表达式

表达式	简单说明
-z str	如果给定字符串的长度为 0，则条件测试的结果为真。在“! -z”情况下，如果字符串未加引号，或者在 test 语句中单独使用未加引号的字符串，结果将是不可靠的。
-n str	如果给定字符串的长度大于 0，则条件测试的结果为真。“-n”测试要求字符串前后必须加引号。
s1 = s2	如果给定字符串 s1 等同于字符串 s2，则条件测试的结果为真。
s1 != s2	如果给定字符串 s1 不等同于字符串 s2，则条件测试的结果为真。
s1 < s2	基于字符的 ASCII 编码值，如果给定的字符串 s1 小于字符串 s2，则条件测试的结果为真。其部分用法列举如下： <ul style="list-style-type: none"> <li>□ test s1 &lt; s2;</li> <li>□ [ s1 &lt; s2 ];</li> <li>□ [[ s1 &lt; s2 ]].</li> </ul> <p>注意，在单方括号“[... ]”中，小于号“&lt;”前需加转义符号。</p>
s1 > s2	基于字符的 ASCII 编码值，如果给定的字符串 s1 大于字符串 s2，则条件测试的结果为真。其部分用法列举如下： <pre>test s1 &lt; s2; [ s1 &gt; s2 ]; [[ s1 &gt; s2 ]].</pre> <p>注意，在单方括号“[... ]”中，大于号“&gt;”前需加转义符号。</p>

### 1. 字符串等同性测试

test 语句支持两种字符串等同性比较测试：等于（=）测试用于比较两个表达式的相等性，不等（!=）测试用于比较两个表达式的不等性。

下面是两个 test 语句的例子。第一个例子测试两个字符串的相等性，第二个例子测试两个字符串的不等性。注意，比较运算符两端必须各有至少一个空格。如果漏掉了空格，test 语句就会把比较运算符看作赋值运算符或待测试字符串的一部分，如下所示。

```
$ name="John"
$ test "$name" = John
```



```
$ echo $?
0
$ name="John"
$ [ "$name" = John ]
$ echo $?
1
$
```

此外还要注意，在 test 语句的字符串比较表达式中，引用的变量或字符串前后一定要加双引号。否则，当变量或字符串表达式的值为 null 时，Shell 将会忽略变量或字符串表达式的存在，导致语法错误。例如，当 NOTSET 变量未设置时，第一个 test 命令认为 “!=” 运算符左边的参数不存在，故测试出错，如下所示。

```
$ echo ${NOTSET}

$ test ${NOTSET} != "hello"
-bash: test: !=: unary operator expected
$ test "${NOTSET}" != "hello"
$ echo $?
0
$
```

在上述例子中，第一个 test 语句执行有误，说明漏掉了双引号。Shell 按 IFS 处理机制从命令行中删除值为 null 的字符串参数，因此，当执行 test 语句时，Shell 只看到两个参数，即 “!=” 与 “hello”，因而认为不满足字符串比较所需的 3 个参数。

## 2. 字符串长度测试

test 语句也可用于测试字符串表达式的长度。test 语句中的 “-z” 和 “-n” 选项分别用于测试字符串表达式的长度是否为 0 或非 0。如果未明显地指定任何选项，“-n” 将会成为默认的选项。也就是说，如果 test 语句中只有一个参数，则当字符串参数包含一个或多个字符时，测试条件的结果为真，因而返回一个值 0。

字符串长度测试经常用于安装软件时检查变量是否已经设置，根据变量或形式参数的设置与否，组织控制结构，决定 Shell 脚本命令执行的控制流向，示例如下。

```
$ test -z "string"
$ echo $?
1
$ test -n "string"
$ echo $?
0
$ test "string"
$ echo $?
0
$
```

### 7.4.3 整数值测试运算符

test 语句还可用于比较字符串表达式中包含的整数值。整数字符串中不能包含任何非数字字符，但前后可以有空格字符。在 test 语句中，整数值的比较采用 C 语言中的 atoi() 函数，把字符转换成等价的 ASCII 整数值。

下面是一些合法的整数字符串表达式的例子(Bourne Shell 允许数字后面包含其他非数字字符)。

"486"	486
"15 "	15
" 123"	123

下列两个例子说明了怎样使用 test 语句比较整数字符串表达式。

```
$ test "123" -eq "123"
$ echo $?
0
$ test "123" -eq " 123 "
$ echo $?
0
```

表 7-13 给出了 test 语句中常用的整数测试表达式。

表 7-13 整数测试表达式

表达式	简单说明
<code>exp1 -eq exp2</code>	如果表达式 exp1 的整数值等于表达式 exp2 的整数值，则计算结果为真
<code>exp1 -ne exp2</code>	如果表达式 exp1 的整数值不等于表达式 exp2 的整数值，则计算结果为假
<code>exp1 -gt exp2</code>	如果表达式 exp1 的整数值大于表达式 exp2 的整数值，则计算结果为真
<code>exp1 -lt exp2</code>	如果表达式 exp1 的整数值小于表达式 exp2 的整数值，则计算结果为假
<code>exp1 -ge exp2</code>	如果表达式 exp1 的整数值大于等于表达式 exp2 的整数值，则计算结果为真
<code>exp1 -le exp2</code>	如果表达式 exp1 的整数值小于等于表达式 exp2 的整数值，则计算结果为假

#### 7.4.4 逻辑运算符

test 语句支持表达式的逻辑运算。其中，符号“!”表示逻辑非运算，符号“-a”或“&&”表示逻辑与运算，符号“-o”或“||”表示逻辑或运算（参见表 7-14）。

表 7-14 逻辑运算符

表达式	说明
<code>( exp )</code>	用于计算括号中的组合表达式。如果整个表达式的计算结果为真，则测试结果也为真
<code>! exp</code>	对表达式进行逻辑非运算，即对测试结果求反。如果表达式的计算结果为假，则最终的测试结果为真
<code>exp1 -a exp2</code> <code>exp1 &amp;&amp; exp2</code>	对两个表达式进行逻辑与运算。如果两个表达式的计算结果均为真，最终的测试结果才为真。注意，单方括号“[...]”结构中不允许使用&&运算符。逻辑与运算符的部分用法列举如下： <ul style="list-style-type: none"> <li>□ <code>test exp1 -a exp2;</code></li> <li>□ <code>test !( exp1 -a exp2 );</code></li> <li>□ <code>[ exp1 -a exp2 ];</code></li> <li>□ <code>[ exp1 ] &amp;&amp; [ exp2 ];</code></li> <li>□ <code>[[ exp1 &amp;&amp; exp2 ]].</code></li> </ul>
<code>exp1 -o exp2</code> <code>exp1    exp2</code>	对两个表达式进行逻辑或运算。只要两个表达式的计算结果中有一个为真，最终的计算结果就为真。同样，单方括号“[...]”结构中不允许使用“  ”运算符。逻辑或运算符的部分用法列举如下： <ul style="list-style-type: none"> <li>□ <code>test exp1 -o exp2;</code></li> <li>□ <code>test !( exp1 -o exp2 );</code></li> <li>□ <code>[ exp1 -o exp2 ];</code></li> <li>□ <code>[ exp1 ]    [ exp2 ];</code></li> <li>□ <code>[[ exp1    exp2 ]].</code></li> </ul>

在比较两个文件时，可以使用 diff 命令。比较之前，为了确保两个文件都存在，可以写出下列 Shell 代码片段。

```
if [ -f file1 -a -f file2 ]
then
    diff file1 file2
fi
```

下面的例子说明了怎样使用各种逻辑运算符。

```
$ test ! -d /tmp
$ echo $?
```



```
1
$ test \(-d /tmp -a -x /tmp \)
$ echo $?
0
$ ! [ -d /tmp -a -x /tmp ]
$ echo $?
1
$ [[ -d /tmp && -x /tmp ]]
$ echo $?
0
$
```

## 7.5 命令行的解释执行过程

Shell 命令行的解释执行是一个复杂的处理过程，了解命令行的解释执行过程有助于用户正确地访问系统，准确地输入命令，从而避免误用或出现意外的结果，影响命令解释执行的最终结果。例如，假定使用一个变量的值实现 I/O 重定向，把命令的输出保存到一个文件中，其结果如下。

```
$ SAVEFILE="> /tmp/savefile"
$ echo something $SAVEFILE
something > /tmp/savefile
$ cat /tmp/savefile
cat: /tmp/savefile: No such file or directory
$
```

从上述命令的运行结果可以看出，Shell 并没有把 echo 命令的输出重定向到文件 savefile 中，而是把“something”和 SAVEFILE 变量的值“>/tmp/savefile”一起输出到终端窗口（标准输出）上。实际上，Shell 的 I/O 重定向是在变量替换之前执行的。替换 SAVEFILE 变量之后，Shell 只是把变量的值作为 echo 命令参数的一部分送到标准输出，因而没有创建预期的文件/tmp/savefile。

因此，有必要在此讨论一下 Shell 命令行的解释执行过程。Shell 命令行的解释执行过程大体可以分为 16 个步骤。按照顺序列举如下：

- (1) 读取命令行；
- (2) 命令历史替换；
- (3) 命令别名替换；
- (4) 花括号扩展；
- (5) 波浪号替换；
- (6) I/O 重定向；
- (7) 变量替换；
- (8) 算术运算结果替换；
- (9) 命令替换；
- (10) 单词解析；
- (11) 文件名生成；
- (12) 引用字符处理；
- (13) 进程替换；
- (14) 环境处理；

- (15) 执行命令；
- (16) 跟踪执行过程。

### 7.5.1 读取命令行

Shell 命令行解释执行过程的第一步是读取命令行。命令行可以来自终端，也可以取自一个普通的文本文件，如 Shell 脚本文件。

不管是以交互方式访问 Shell，还是运行 Shell 脚本，Shell 都需要读取命令行。如果是交互式访问 Shell，还需要在用户终端上回显其读取的每一个字符，然后对读取的命令行进行解析和处理。在解析和处理每个命令行之前，Shell 需要读取完整的命令行。部分 Shell 内置命令，如 if/then、for 和 case 等结构语句，函数定义与长字符串等，通常都可能需要跨越多个物理行。当读取多行形式的命令语句时，在进一步解析和处理之前，Shell 将会连续地读取整个命令语句。在交互会话期间，Shell 会采用辅助命令提示符“>”(PS2 变量的默认值)的形式，提示用户输入后续的命令语句，此时用户可以继续输入，直至 Shell 认为命令行的输入已经结束。示例如下。

```
$ while true
> do
> ls -l /var/log/somefile
> sleep 5
> done
...
^C
$ echo ***** NOTICE *****
> Today is Christmas.
> Every body could go home after 12:00."
***** NOTICE *****
Today is Christmas.
Every body could go home after 12:00.
$
```

在读取命令行时，Shell 将会逐个判读读取的每一个字符，直至遇到一个分号“;”、后台进程符号“&”、逻辑与“&&”、逻辑或“||”或换行字符时，整个命令行的读取过程才算结束。如果读取的是一个结构语句，如 if/then、for 或 while 等，Shell 将会完整地读入整个结构语句。

在读入一个完整的命令或结构语句之后，Shell 开始对命令语句进行语法分析，把命令语句分解为一系列单词或关键字。通常，Shell 假定每个单词或关键字都是由空格或制表符分隔的一系列连续字符组成的（参见 IFS 变量的说明）。Shell 从命令语句行首的第一个字符开始解析，直至行尾结束，依次剥离出一系列单词，包括由“<”、“>”、“|”和“^”等特殊字符组成的单字。不管 IFS 变量的设置如何，这一解析过程总是如此处理。至于如何处理 IFS 变量，则是另外一个解析步骤。

### 7.5.2 命令历史替换

在读取一个完整的命令行之后，Shell 首先会检查是否需要使用命令历史缓冲区中记录的命令替换读取的命令，是否需要编辑命令。如果需要，则取出相应的命令，再经过适当地校正之后，使之作为当前需要执行的命令。例如，当输入“!!”命令时，意味着重新执行先前执行的最后一条命令，于是 Shell 会从命令历史记录中取出相应的命令，替换“!!”命令。



通常，交互式 Shell 总是会首先执行命令历史机制的命令替换处理过程。如果想要提高 Shell 命令行解释执行的效率，省略这一处理步骤，可以使用下列命令禁止 Shell 执行命令历史替换（注意，命令历史替换处理不适用于非交互式 Shell，如运行 Shell 脚本）。

```
$ set +o histexpand  
$
```

### 7.5.3 别名替换

命令行解释执行的第二步是别名替换。在此步骤中，Shell 将检查读取的命令是否为命令别名。如果确为命令别名，则根据定义，使用原始的命令予以替换。通常，交互式 Shell 总是会执行命令别名检查，非交互式 Shell 则禁用这一功能特性。为了省略这一处理步骤，可以使用下列命令禁止 Shell 执行命令别名替换。

```
$ shopt -u expand_aliases  
$
```

### 7.5.4 花括号扩展

花括号提供了一种简单的文件名生成方法。下列例子说明了怎样利用花括号扩展机制指定文件名，在`/home/gqxing/scripts` 目录中一次创建 4 个子目录。

```
$ mkdir ~/scripts/{old,new,tools,admin}  
$
```

在上述例子中，Shell 尝试把花括号中以逗号“，”分隔的一系列单词与花括号之外的字符串连接在一起，生成一系列由空格分隔的字符串，用于指定或匹配当前目录中的文件。

当文件的路径名较长时，花括号扩展机制是非常有用的。例如，为了把`/home/gqxing/src` 目录中的部分 C 程序（`main.c`、`list.c`、`scan.c` 和 `mon.c`）复制到当前目录，可以使用下列命令。

```
$ cp /home/gqxing/src/{main,list,scan,mon}.c .  
$
```

但是，Shell 并不总是尝试使用花括号扩展机制匹配现有的文件名，实际上，也可以使用花括号扩展机制生成任何字符串。例如，利用下列命令生成的字符串并非实际存在的目录文件名。

```
$ echo chap-{one,two,three,four}  
chap-one chap-two chap-three chap-four  
$
```

通常，交互式 Shell 总是启用花括号扩展机制，非交互式 Shell 则禁用花括号扩展机制。要禁止使用花括号扩展机制，可以输入下列命令。

```
$ set +o braceexpand  
$
```

### 7.5.5 波浪号替换

要指定用户的主目录，除了使用`$HOME` 之外，还可以在用户名之前加一个波浪号“~”，或者直接使用波浪号表示当前用户的主目录。当命令行中出现波浪符“~”特殊字符时，Shell 将会继续读取随后的字符串，直至遇到一个斜线字符“/”或空白字符（表示单字的结束）为止，然后验证读取的字符串是否为一个有效的注册用户名。如果读取的字符串为空（即后面只有一个斜线字符“/”，因而只有波浪符“~”本身），Shell 将会使用当前用户的`HOME` 变量值替换波浪号“~”。示例如下。

```
$ echo $HOME  
/home/gqxing  
$ echo ~  
/home/gqxing
```

```
$ ls -l ~/src
总用量 128
-rw-r--r-- 1 gqxing gqxing 18222 Nov  8 11:50 atmcom.c
-rw-r--r-- 1 gqxing gqxing 11802 Nov  8 11:52 atmmon.c
-rw-r--r-- 1 gqxing gqxing 23221 Nov  8 11:53 atmstat.c
-rw-r--r-- 1 gqxing gqxing 11742 Nov  8 11:53 handler.c
-rw-r--r-- 1 gqxing gqxing 55596 Nov  8 11:56 listener.c
$
```

如果波浪符“~”之后是一个有效的注册用户名，Shell 将以相应用户主目录的路径名替换波浪符“~”和用户名；如果“~”之后既不为空，也不是一个有效的用户名，Shell 不会做任何替换。示例如下。

```
$ echo $HOME
/home/gqxing
$ ls -l ~gqxing
总用量 20
drwxr-xr-x 2 gqxing gqxing 4096 2008-11-7 09:48 conf
drwxr-xr-x 2 gqxing gqxing 4096 2008-11-7 15:16 incl
drwxr-xr-x 2 gqxing gqxing 4096 2008-11-7 09:57 lib
-rw-r--r-- 1 gqxing gqxing 1020 2008-11-7 17:51 makefile
drwxr-xr-x 2 gqxing gqxing 4096 2008-11-7 18:37 src
$ echo ~xxx
~xxx
$
```

此外，如果波浪符“~”之后为加号“+”或减号“-”，则“~+”表示 PWD 变量的值，即当前工作目录；“~-”表示 OLDPWD 变量的值，即先前的工作目录。示例如下。

```
$ cd /etc
$ cd
$ echo $PWD
/home/gqxing
$ echo ~+
/home/gqxing
$ echo $OLDPWD
/etc
$ echo ~-
/etc
$
```

## 7.5.6 I/O 重定向

如果不了解文件描述符、进程文件表（也称文件描述符表或用户文件表）、系统文件表和信息节点表，很难理解 I/O 重定向。这些表都是 Linux 系统维护的数据结构。进程文件表中包含每个进程已打开文件指向系统文件表的指针，而系统文件表中的文件指针则指向位于内存中的信息节点表，信息节点表包含文件的信息节点，而信息节点含有文件的打开方式（如读或写等）、文件的当前读写位置及怎样获取文件的数据内容等信息。文件描述符是文件在系统文件表中的位置索引。每个进程对文件的所有读写操作都是通过文件描述符实现的。操作系统利用文件描述符，从系统文件表和信息节点中找出目标文件，最终完成文件的处理，如图 7-1 所示。

文件描述符 0 是标准输入，通常为终端的键盘输入。文件描述符 1 为标准输出，文件描述符 2 为标准错误输出，两者通常为终端显示。当注册到 Linux 系统时，这 3 个文件描述符是已经打开的默认文件描述符。

通常情况下，Shell 通过标准输入读取命令语句，通过标准输出显示命令执行的结果；当执行过程中发现问题时，则通过标准错误输出显示错误信息。

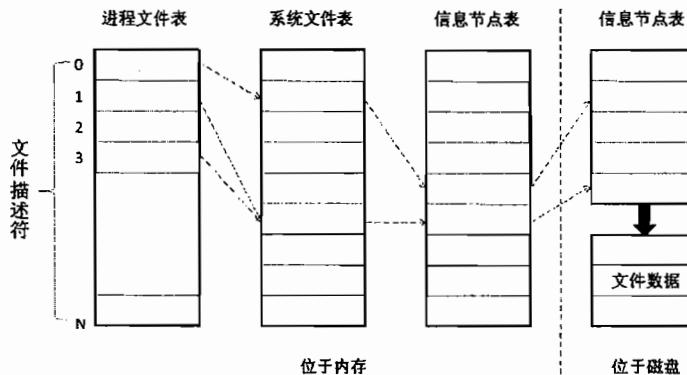


图 7-1 文件描述符与文件的关系

在输入输出重定向的情况下，Shell 需要执行若干处理动作：关闭指定的文件描述符，如关闭文件描述符 0（如果标准输入被重定向）或关闭文件描述符 1（如果标准输出被重定向）等；打开指定的文件；把新的文件指针放入文件指针数据结构表中刚才腾出的位置，以替代刚关闭的相应文件。

下面是一个 I/O 重定向的例子。

`command < datafile`

在这个例子中，标准输入被重定向到 `datafile` 中。在实现过程中，Shell 首先关闭文件描述符 0，然后以输入方式打开 `datafile` 文件，最后把打开操作产生的文件指针信息复制到原标准输入所在文件指针数据结构表中已腾出的位置，如图 7-2 所示。

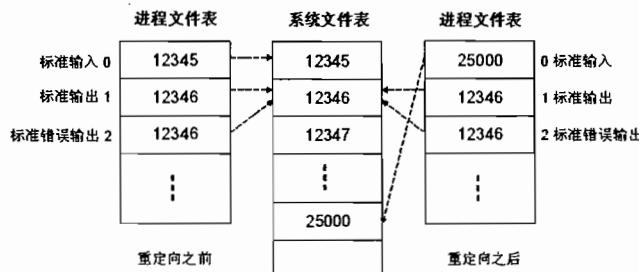


图 7-2 标准输入重定向

至此，当命令从标准输入（文件描述符 0）读取数据时，将会从新打开的 `datafile` 文件，而不是终端的键盘输入中读取数据。如果标准输出或标准错误输出也被重定向，Shell 将采取同样的动作。

### 7.5.7 变量替换

变量替换涉及 Shell 内部变量、用户自定义变量、命令行参数或位置参数等变量替换。

变量替换表达式由美元符号“\$”与随后的变量名构成，如`$var`。位置参数替换表达式由美元符号“\$”与随后的数字构成，如`$1`、`$2`、……等。变量名或参数名可以用花括号括起来，如 `${var}`。变量替换表达式前后也可以加单、双引号，如`fname="${var}"`。但在实际替换时，两者存在细微的差别（详见 7.2 节有关变量替换的介绍）。

## 7.5.8 算术运算结果替换

在 Bash 中，算术运算结果替换的语法格式如下。

```
$((expression))
```

当命令行中出现 “\$((expression))” 形式的基本元素时，Shell 首先需要计算双括号中的表达式，然后使用计算结果替换 “\$((expression))”。示例如下。

```
$ echo There are $((60*60*24*365)) seconds in a year
There are 31536000 seconds in a year
$
```

## 7.5.9 命令替换

命令替换表达式是由反向单引号 `` 或 \${...} 形式的命令语句组成的，命令的输出就是命令替换表达式的值。命令可以是任何 Linux 命令，包括普通命令、顺序执行的命令、组合命令或含有管道符号的并列命令等。

作为命令替换表达式的一部分，命令语句可以使用任何 Shell 机制，如变量替换和文件名生成机制等。因此，在执行命令之前，Shell 需要把整个命令行完全展开，最终形成能够实际执行的具体命令。

如果在命令的替换过程中出现多余的空格、制表符或换行符等，Shell 将会在第二次重读、命令语句解析、执行引用字符处理时予以删除。因此，如果这些空格、制表符或换行符等应当保留，输入时应使用双引号括住整个命令替换表达式。

在反向单引号之前增加转义符号 “\”，可以实现命令替换表达式的嵌套。下面的例子解释了命令表达式的嵌套是怎样实现的。

```
$ cat is.today
for fname in $(*)
do
    appoint=`grep \`date +%m/%d/%y\` ${fname}`
    echo "${appoint}"
done
$ cat calendar
11/08/08      Finish unit 15
11/09/08      Finish unit 16
11/10/08      Finish unit 17
$ date +%m/%d/%y
11/08/08
$ is.today calendar
11/08/08      Finish unit 15
$
```

在上述 is.today 脚本文件中，最内层的反向单引号各增加了一个转义符号 “\”。这个脚本首先确定当天的日期，然后利用这个日期作为模式，使用 grep 命令检索 fname 变量指定的文件，最后把检索到的内容赋予变量 appoint，并输出最终结果。

## 7.5.10 单词解析

在完成了变量替换、命令替换以及算术运算结果替换之后，Shell 将利用 IFS 变量设定的字符作为分隔符，对经过各种替换后生成的命令行再次进行解析，以分离出最基本的命令行元素或单词。此时，如果命令行中存在单引号或双引号，其中的内容（包括 null 参数）将被看作一个单词。否则，Shell 将会删除额外的空格、制表符和换行符，以及隐含的 null 参数。

IFS 变量的默认值为空格、制表符和换行符，如果需要，用户也可以修改或重新定义 IFS 变量设置。



在IFS单词解析处理阶段，变量赋值语句是受特殊保护的。除了赋值之外，Shell 不会对变量赋值语句做文件名生成等其他任何解析处理。因此，如果希望把一个变量表达式的值赋予等号左边的变量，而变量表达式又可能生成多个单词时，这样的变量表达式前后不应加双引号。

下面的例子解释了IFS变量的设置如何影响命令行的解释执行。

```
$ a=x:y:z
$ cat $a
cat: x:y:z: No such file or directory
$ IFS=$IFS:""
$ cat $a
cat: x: No such file or directory
cat: y: No such file or directory
cat: z: No such file or directory
$
```

## 7.5.11 文件名生成

命令行解释执行的下一步是扩展元字符，以生成文件名。在这个步骤里，Shell 将会检索解析出来的每一个单词，检查其中是否包含任何符合文件名生成规则的元字符，如星号“\*”、问号“?”和方括号“[...]"表示的字符范围等。

对于包含元字符的每一个基本单词，Shell 将会尝试匹配当前目录或指定目录中的文件名。如果找到匹配的任何文件名，Shell 将使用这些文件名替换命令语句中的元字符表达式。示例如下。

```
$ ls file*
file1 file2 file3
$ echo file*
file1 file2 file3
$ rm file?
$ echo file*
file*
$
```

在文件名生成过程中，涉及下一小节将要介绍的单双引号的引用处理。在下面的例子中，为了引用当前目录下的所有文件，首先为变量 files 赋值一个星号“\*”。

```
$ files=*
$
$ 使用 echo 命令显示变量 files 的值，且在变量的前后增加了双引号时，Shell 只做变量替换，但不执行文件名生成，如下所示。
```

```
$ echo "${files}"
*
$
```

如果未加双引号，Shell 将会扩展元字符“\*”，执行文件名的生成过程，如下所示。

```
$ echo ${files}
file1 file2 file3
$
```

如 7.6.10 小节所述，变量赋值时并不执行文件名生成。因此，在执行赋值语句“files=\*”时，变量 files 的值只是一个星号“\*”，而不是当前目录下的所有文件名。在此步骤中，Shell 不会解析被双引号括住的变量赋值，但参数和变量替换会照常执行。如果是单引号，Shell 将会禁止一切替换。因此，如果想要使用一个带有元字符的变量值，而又不希望扩展元字符，可用双引号括住变量表达式。

特别需要注意的是，文件名的生成是在变量替换、命令替换和 I/O 重定向之后进行的。因此，在涉及文件名生成的表达式中要特别小心，尤其是在带有 I/O 重定向的表达式中，一定要注意。

### 7.5.12 引用字符处理

引用字符处理的目的是删除引用符号。在此处理步骤中，Shell 开始删除转义符号 “\” 和单双引号等引用字符，展开引号中的表达式，完成命令执行前的所有预备工作。

引用字符处理涉及文件名的生成及各种替换。如果一个基本单词前后有双引号，Shell 将会禁止文件名生成，禁止除参数和变量替换之外的所有替换。如果有单引号，Shell 将会禁止文件名生成和所有的替换。下面的例子解释了单引号、双引号以及不加任何引号的引用字符处理与文件名生成。

```
$ ls file*
file1 file2 file3
$ var=file*
$ set | grep var=
var='file*'
$ echo '$var'
$var
$ echo "$var"
file*
$ echo $var
file1 file2 file3
$
```

### 7.5.13 进程替换

进程替换是相对于命令替换而言的。命令替换的主要作用是把命令的输出数据赋予某个变量，以便作进一步的处理，或者利用 Shell 的管道机制，直接接收某个命令的输出数据，或者其他命令直接提供输入数据等。例如，“var=`command`”或“command | command”。

而进程替换则是利用/dev/fd/<nn>特殊文件或管道文件，把一个进程的输出结果，作为输入数据提交给另一个进程。换言之，进程替换是利用/dev/fd/<nn>特殊文件或管道文件，实现进程之间标准输入与标准输出的交互通信的。

进程替换采用“<(command)”或“>(command)”的形式实现。当圆括号中的进程开始运行时，其标准输入和标准输出将会连接到/dev/fd 目录中的两个特殊文件或管道文件，同时把文件名作为参数传递给当前命令的标准输入或标准输出。如果采用的是“>(command)”形式的进程替换，当前命令的标准输出将会写到圆括号中指定进程的标准输入。如果采用的是“<(command)”形式的进程替换，作为参数传递的特殊文件或管道文件可用于读取圆括号中指定进程的标准输出。

注意：在“<”和“>”与圆括号之间没有空格，否则将会出现错误（信息）。

进程替换的一个重要功能特性是可用进程替换命令行中的文件名参数。当一个文件名参数位置出现“<(command)”形式的进程调用时，将会引起 Shell 执行圆括号中的命令，把进程的标准输出写到一个特殊文件或管道文件中，然后把此文件作为当前命令的标准输入，从指定进程的标准输出中读取输入数据。

同样，当一个文件名参数位置出现“>(command)”形式的进程调用时，也会引起 Shell 执行



给定的命令，但 Shell 将会把该进程的标准输出写到一个特殊文件或管道文件中，然后以此文件作为圆括号中指定进程的标准输入。

因此，可以使用进程替换比较两个不同的命令，或者同一命令但选项或参数不同时的输出结果。例如，下列例子说明了怎样使用进程替换比较两个目录中的文件。

```
$ diff <(ls -l doc) <(ls -l doc2)
2c2
< -rw-r--r-- 1 gqxing gqxing 498 2008-11-08 11:09 readme
---
> -rw-r--r-- 1 gqxing gqxing 516 2008-11-08 11:10 readme
$
```

下列命令将会按照文件的大小，对“ls -l”命令输出的数据进行排序（其中，“-n”选项表示按数值排序，“-k 5”表示按输入数据的第 5 列排序）。

```
$ sort -n -k 5 <(ls -l)
总用量 128
-rw-r--r-- 1 gqxing gqxing 11742 Nov  8 11:53 handler.c
-rw-r--r-- 1 gqxing gqxing 11802 Nov  8 11:52 atmmon.c
-rw-r--r-- 1 gqxing gqxing 18222 Nov  8 11:50 atmcom.c
-rw-r--r-- 1 gqxing gqxing 23221 Nov  8 12:05 atmstat.c
-rw-r--r-- 1 gqxing gqxing 55596 Nov  8 11:56 listener.c
$
```

sort 命令的“-o”选项本来用于指定一个输出文件，以便存储排序后的结果。下面的例子以进程替换的形式代替输出文件，借用 gawk 命令，从排序后的结果中抽取文件的名字与大小字段。

```
$ sort -n -k 5 <(ls -l) -o >(gawk '{print $9 "\t" $5}')
handler.c      11742
atmmon.c       11802
atmcom.c        18222
atmstat.c      23221
listener.c      55596
$
```

### 7.5.14 环境处理

命令行解释的第 14 步是环境处理，其中包括变量赋值，检索 PATH 变量指定的目录，找出命令文件的存储位置，等等。

在完成变量赋值之后，Shell 将根据 PATH 环境变量的定义，从左到右依次检索命令文件的存储位置，然后以命令文件的完整路径名替换命令行中的命令名。如果命令名中（最后一个字符位置之前）包含斜线字符“/”，Shell 将会绕过 PATH 变量检索，假定给定的命令是一个规范的路径文件名——相对路径文件名或绝对路径文件名。

### 7.5.15 执行命令

至此，终于到了真正开始执行命令的时候了。如果命令行要求执行的是一个普通的 Linux 命令，Shell 将启动一个单独的子进程执行相应的命令；如果命令行要求执行的是一个 Shell 内部命令，则由 Shell 直接执行。

如果命令是一个经过编译后生成的应用程序，Shell 启动的子进程将使用 exec 系统调用执行应用程序；如果是一个 Shell 脚本，Shell 将解释执行脚本文件中的每一行语句。具体执行过程是，如果访问权限表示命令文件是可执行的，为了确定命令文件是否能够直接运行，子进程将会尝试把命令文件加载到内存中，然后开始执行；如果无法加载到内存，子进程将假定相应的命令文件

是一个 Shell 脚本，因而由 Shell 采用解释执行的方式运行脚本文件。

如果程序是采用前台方式运行的，Shell 将会一直等待，直至子进程执行结束；如果程序采用的是后台运行方式，Shell 将不再关心子进程的执行是否终止，而是立即在标准输出上显示一个作业号和进程 ID，开始继续处理其他请求。

### 7.5.16 跟踪执行过程

如果先前使用 set 命令的 “-v” 或 “-x” 等选项设定了跟踪命令的执行过程，Shell 将会在这个处理步骤中输出当前正在执行的命令语句，同时在每个命令语句之前插入一个加号 “+” 前缀，然后再输出命令的运行结果。

因为命令替换已在命令执行之前完成，此时输出的是实际执行的命令，而且是按照命令执行的逻辑顺序依次输出的。

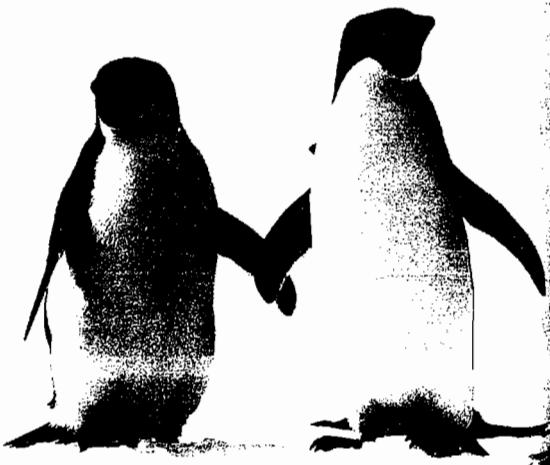
如果命令行包含管道形式的并列命令，并列命令的执行顺序并没有严格的定义，命令的调度执行将是随机的。通常，Shell 会首先调度执行管道字符之前的第一个命令，然后依次调度执行其他后续命令，但这并不等于每个命令都是严格按照其在命令行中出现的位置顺序执行的。



## 第8章 Shell 高级编程

本章主要介绍 Shell 的各种结构语句, 讨论怎样利用 Shell 提供的结构语句, 编写具有较强功能的 Shell 脚本, 介绍怎样利用 Shell 的编程机制和 Linux 系统提供的丰富命令, 创建自己的日常系统维护脚本。在此基础上, 建议读者仔细阅读利用 Shell 脚本实现的系统维护命令, 编写出高水平的 Shell 脚本。其内容主要包括:

- if 与 case 控制转移语句;
- for、while、until 与 select 循环语句;
- 嵌套的循环;
- 循环控制与辅助编程命令;
- 循环语句的 I/O 重定向;
- here 文档;
- Shell 函数;
- 逻辑与和逻辑或并列结构;
- Shell 数组;
- 信号的捕捉与处理;
- 其他 Shell 课题。



除了顺序执行命令之外，Shell 还提供了丰富的控制结构，如 if-then-else 和 case 等转移控制语句，允许用户通过条件计算或逻辑运算等测试机制，控制 Shell 脚本的执行流向，利用 for、while 和 until 等循环结构语句实现循环控制。

## 8.1 if 条件语句

在 Shell 脚本中，if 语句是最基本也是最常用的重要语句之一。利用 if-then-else 条件语句，可以实现脚本执行流程的控制转移。最简单的 if 语句语法格式如下。

```
if command
then
    command-list
fi
```

if 语句以给定命令的出口状态作为判断条件，确定转入哪一个控制流向。如果命令执行成功，返回一个 0 值的出口状态，则执行紧随 then 之后的命令；如果命令执行失败，返回一个非 0 值的出口状态，则执行紧随 fi 之后的命令。fi 表示 if 语句的结束。

if-then 之间的命令可以是一个普通命令，也可以是一组命令；可以是一个简单的测试语句，也可以是一组复杂的组合测试语句。当 if-then 之间存在一组命令时，if-then 控制结构测试的是 if 和 then 之间最后一条命令的出口状态。

### 8.1.1 if 语句的表现形式

许多 if 语句均使用 test 命令作为逻辑判断条件，根据 test 命令的测试结果控制 Shell 脚本的流向，或者决定下一步应采取的动作。例如，许多 Shell 脚本都会在正式开始运行之前检查命令行的参数个数是否正确，如果给定的参数数量不足，则显示 Shell 脚本的用法，然后结束运行。基于这一思路，如果 Shell 脚本要求最少提供两个参数，可以在 Shell 脚本的开始部分写出下列 if 语句（其中的“\$#”表示命令行参数的个数）。

```
$ cat chkargs
if test $# -lt 2
then
    echo "Usage: chkargs arg1 arg2 ..."
    exit 1
fi
echo "Go on to the next."
$
```

“if test ...” 结构等价于 “if [...]” 和 “if [[...]]” 结构，因此，上述 if 语句可以改写如下。

```
if [ $# -lt 2 ]
then
    echo "Usage: chkargs arg1 arg2 ..."
    exit 1
fi
```

if 语句也可以使用 else 子句处理 test 语句返回非 0 出口状态的情况，使之执行 else 子句后面的语句。完整的 if 语句语法格式如下。

```
if command
then
    command-list
else
    command-list
fi
```



例如，当运行某个进程（假定为 `atmmon`）时，如果需要把进程的运行结果记录到系统日志文件中，可根据进程的出口状态是否为 0 确定其运行是否正常结束，然后利用 `logger` 命令把结果写入系统日志文件。据此，可以写出下列代码。

```
atmmon                                # ATM 监控程序
if [ test $? -eq 0 ]
then
    logger "atmmon has run successfully."
else
    logger "atmmon terminated abnormally."
fi
```

`if` 语句后面既可以没有 `test` 语句，也可以没有 “[...]" 或 “[...]" 等测试结构，而是利用其他命令的执行结果作为测试条件。按照定义，这种形式也是符合 `if` 语句的语法规定的。

假定仅当某个目录存在，系统才会支持某一功能，因而才能执行其中的命令时，可以首先使用 `cd` 命令尝试进入指定的目录。如果 `cd` 命令执行成功，`Shell` 脚本才能正常运行；否则，退出 `Shell` 脚本。根据这个思路，可以写出下列 `Shell` 脚本。

```
dir=/opt/atm
if cd "$dir" 2>/dev/null
then
    echo "Now we are in $dir."
    ...
else
    echo "Can't change to $dir."
    exit 1
fi
```

在 “`if command`” 结构语句中，给定命令的出口状态用于控制 `Shell` 脚本的转移流向。下面的例子说明了怎样利用 `diff` 命令返回的出口状态判断文件比较的结果。

```
if diff fname1 fname2 > /dev/null 2>&1
then
    echo "Files fname1 and fname2 are identical."
else
    echo "Files fname1 and fname2 differ."
fi
```

下面是一个比较有用的 “`if-grep`” 结构。其中，`grep` 命令的 “`-q`” 选项用于抑制命令的输出。

```
if grep -q Linux fname
then
    echo "File contains at least one occurrence of Linux"
fi
```

当 `if`、`then` 以及条件测试命令位于同一行上时，`then` 之前必须加一个分号。尽管 `if` 和 `then` 是一个整体，`if` 和 `then` 是其中的两个关键字，但它们均表示一个子句的开始，加分号的目的是表示 `if` 子语句的终止。这与当多个语句位于同一命令行时中间必须加分号 “;” 分隔符是完全一样的，示例如下。

```
if [ condition ]; then
    command-list
else
    command-list
fi
```

## 8.1.2 嵌套的 if-then 条件测试

`if-then` 结构的条件测试还可以嵌套。第一种 `if-then` 嵌套结构的语法格式如下。

```
if [ condition1 ]
then
    if [ condition2 ]
    then
        command-list
```

```

fi
fi

```

上述情况相当于使用`&&`逻辑运算符的组合测试，如下所示。

```

if [ condition1 ] && [ condition2 ]
then
    command-list
fi

```

第二种 if-then 嵌套结构采用`elif`（`else if` 的缩写）子句，其语法格式如下。

```

if [ condition1 ]
then
    command-list
elif [ condition2 ]
then
    command-list
else
    default-command
fi

```

通过使用`elif` 子句，`if` 语句可以实现多重测试结构。同样，如果`elif` 子句后面的测试条件的出口状态为真，则执行紧随`elif` 子句中`then` 后面的语句。

例如，要在用户注册时，根据系统的当前时间显示不同的问候语，可以在`.profile` 初始化文件中增加下列语句。

```

$ cat greeting
now='date '+%H'
if [ ${now} -lt 12 ]
then
    greeting="Good morning."
elif [ ${now} -lt 18 ]
then
    greeting="Good afternoon."
else
    greeting="Good evening."
fi
echo "$greeting"
exit 0
$ greeting
Good morning.
$ date
2008 年 12 月 22 日 星期一 10:18:20 CST
$ 

```

每个嵌套的`if` 语句必须以`fi` 结束。作为一个组合语句，整个`if` 语句的出口状态是`then` 或`else` 子句中执行的最后一条命令的出口状态。如果没有执行任何命令，则出口状态为 0。

在下面的例子中，由于给定的文件不存在（文件名有误），`ls` 命令将会返回一个非 0 的出口状态。因此，Shell 脚本不会执行最后一个`echo` 语句。

```

$ cat ifthen
fname=/etc/password          # 正确的文件名应为 password
LANG=C                         # 采用英文语言环境，确保提示信息的准确性，下同
if echo "This statement always return true."
    ls -l $fname
then
    echo "Come here."
fi
$ ifthen
This statement always return true.
ls: cannot access /etc/password: No such file or directory
$ 

```

现在，交换`if-then` 之间的两个语句。尽管`ls` 命令将会返回一个非 0 的出口状态，但由于`echo` 命令总是能够成功地执行，故 Shell 脚本将会执行到最后一个`echo` 语句，然后才结束 Shell 脚本的执行。



```
$ cat ifthen2
fname=/etc/inittab          # Ubuntu Linux 系统不用这个文件启动系统
LANG=C
if ls -l $fname
    echo "This statement always return true."
then
    echo "Come here."
fi
$ ifthen2
ls: cannot access /etc/inittab: No such file or directory
This statement always return true.
Come here.
$
```

### 8.1.3 if-then 结构参考

下面是一个增加用户的 Shell 脚本，其目的是避免直接使用 useradd 系统命令，在命令行中输入较多的选项与参数。利用此脚本，调用时只需提供一个用户名，如“add-user *username*”，即可增加指定的用户与用户组。其中采用了 if-then、if-then-else 和 if-grep 等结构语句，如下所示。

```
$ cat add-user
#!/bin/bash
if [ $# -ne 1 ]
then
    echo "Usage: add-user username"
    exit 1
fi
username=$1
comment=$1
homedir=/home/$username
minuid=1000

grep "^\$username" /etc/passwd >/dev/null 2>&1
if [ $? -ne 0 ]
then
    alluids='cat /etc/passwd | cut -d: -f3 | sort -n | tr '\n' ' '
    set $alluids
    while [ $1 -le $minuid ]
    do
        shift
    done
    uid=$minuid
    sum=$#
    while [ $sum -ge 1 ]
    do
        uid='expr $uid + 1'
        if [ $uid -eq $2 ]
        then
            shift
            sum=$#
            continue
        fi
        if ! grep -q $uid /etc/group
        then
            break
        fi
    done
else
    echo "add-user: the username $1 exist."
    exit 2
fi
echo "Adding user: uid = $uid, gid = $uid, username = $username"
useradd -u $uid -d $homedir -m -c $comment -s /bin/bash $username
```

```
· exit 0
$
```

## 8.2 case 分支语句

case 分支语句能够提供多路分支转移控制功能。case 语句利用一个变量作为测试条件，变量可以具有多个值，不同的数值能够引起不同的程序走向，其语法格式如下。

```
case "$variable" in
    pattern1)
        command-list ;;
    pattern2)
        command-list ;;
    ...
    patternN)
        command-list ;;
esac
```

Shell 使用给定变量（variable）的值与每一个模式（pattern）依次进行比较。一旦发现匹配的模式，Shell 将会立即执行紧随模式之后的所有命令，直至遇到双分号 “;”。在双分号前的最后一个命令执行结束之后，Shell 将跳转至紧随 esac 语句之后的第一个语句，开始继续执行。如果用作测试条件的变量值与任何模式都不匹配，则跳转到紧随 esac 语句之后的第一个语句处开始继续执行，而不执行 case 语句中的任何命令。

每个模式之后必须加一个右括号 “)”，以便与随后的一组相关命令分隔。case 语句中的每一组命令或代码块之后必须以双分号结束（最后一组命令或代码块除外），这相当于告诉 Shell 已执行到相关命令或代码块的边界。

模式中可以使用元字符，也可以使用运算符 “|”，表示多个模式的逻辑或关系。由于元字符 “\*” 可以匹配任何数值、字符或字符串，因此可以用作默认的匹配模式。这个“万能的模式”必须作为 case 语句的最后一个模式，因为一旦检查到匹配的模式，Shell 将会停止执行 case 语句的模式匹配检查，致使其他模式永远得不到匹配的机会。

下面是一些与 case 语句语法格式有关的说明：

- 因为 case 语句中用作测试条件的变量值通常都是一个单独的数值或单词，不会出现包含分隔符的字符串，所以测试变量前后可以不加双引号；
- 每个用于匹配检查的模式之后必须附加一个右圆括号后缀；
- 除了最后一组命令或代码块，其他模式的相关命令或代码块之后均需附加一个双分号；
- 整个 case 语句以 esac (case 的反序拼写) 结束。

Shell 中的 case 控制结构相当于 C/C++ 中的 switch 语句，允许整个控制结构根据条件测试变量的值执行相应的一组命令或代码块。因此，case 控制结构是一种非常适合建立多路分支转移程序流向的工具，使用它也能够容易地创建选择菜单。实际上，case 控制结构是 if-then-else 嵌套结构的一种简化形式。

例如，利用 case 控制结构，可以设计一个简单的网络环境设置脚本 netchoice，针对家庭 ADSL 网络环境、办公室网络环境以及 Samba 测试环境，使用不同的网络接口配置文件替换 /etc/network/interfaces 配置文件，然后重新运行新的/etc/init.d/networking 启动脚本。运行 netchoice 脚本时，既可以采用菜单选择的形式，又可以采用命令行参数的形式，适当地设置网络环境（由



于修改 interfaces 文件，运行 networking 启动脚本等需要具有超级用户的访问权限，故需使用 sudo 命令（参见第 13 章）。示例代码如下。

```
$ cat netchoice
#!/bin/bash
choice=$1
if [ $# -lt 1 ]
then
    echo -e "\n\t----- Network Setup Menu -----"
    1. Home/ADSL Networking Environment
    2. Office Networking Environment
    3. Samba Testing Environment
    4. Stop Home/ADSL Networking\n
    Please enter your choice: \c"
    read choice
fi
case "$choice" in
    1 | home )      cp /etc/network/if.home /etc/network/interfaces
                    /etc/init.d/networking restart
                    pon dsl-provider
                    ;;
    2 | office )     cp /etc/network/if.office /etc/network/interfaces
                    /etc/init.d/networking restart
                    ;;
    3 | smb | samba) cp /etc/network/if.smb /etc/network/interfaces
                    /etc/init.d/networking restart
                    ;;
    4 | stop )       poff
                    ;;
    * )              echo "Invalid choice or parameter"
esac
exit 0
$ sudo netchoice

----- Network Setup Menu -----

1. Home/ADSL Networking Environment
2. Office Networking Environment
3. Samba Testing Environment
4. Stop Home/ADSL Networking

Please enter your choice:
```

在 Linux 系统的各种日常管理与维护 Shell 脚本中，按照使用的频繁程度，case 语句也许是仅次于 if-then-else 语句的一种控制结构。尤其是/etc/init.d 或 /etc/rcN.d 目录中的各种系统服务启动脚本，case 语句是其中最常见的固定控制结构：利用同一个 Shell 脚本，在系统的启动、关机或重启过程中，分别以 start、stop 或 restart 等作为参数，启动、关闭或重新启动各种系统服务的守护进程。

下面是取自 atd 系统服务启动脚本文件的部分内容，其中就采用了典型的 case 控制结构用法。

```
$ cat /etc/init.d/atd
#!/bin/sh -e
...
set -e

PATH=/bin:/usr/bin:/sbin:/usr/sbin
DAEMON=/usr/sbin/atd

test -x $DAEMON || exit 0

. /lib/lsb/init-functions

case "$1" in
    start)
        log_daemon_msg "Starting deferred execution scheduler" "atd"
```

```

        start_daemon $DAEMON          # 启动 atd 守护进程
        log_end_msg $?
        ;;
stop)    log_daemon_msg "Stopping deferred execution scheduler" "atd"
        killproc $DAEMON            # 终止 atd 守护进程
        log_end_msg $?
        ;;
force-reload|restart)
        $0 stop                     # 以 stop 作为参数运行本脚本, 终止 atd 守护进程
        $0 start                    # 以 start 作为参数运行本脚本, 启动 atd 守护进程
        ;;
*)
        echo "Usage: /etc/init.d/atd {start|stop|restart|force-reload}"
        exit 1
        ;;
esac

exit 0
$
```

在 case 命令语句中，还可以使用命令替换作为测试变量，以命令的输出作为测试变量的值。例如，系统提供的 diskdump 脚本使用“uname -r”命令的输出作为 case 语句中的测试变量值，以便在相应版本的系统中做特定的设置和处理。示例如下。

```

$ cat /etc/init.d/mountall.sh
.....
case "$(uname -s)" in
    *FreeBSD)           # 当"uname -s"命令输出结果的后缀
        INITCTL=/etc/.initctl # 为 FreeBSD 时, 执行左边的赋值语句
        ;;
    *)                 INITCTL=/dev/initctl
        ;;
esac
.....
$
```

## 8.3 for 循环语句

for 语句是 Shell 中一种最基本的循环控制结构。针对 for 语句中的每个参数，可以重复执行一组命令或代码块。for 语句的语法格式如下。

```

for var [ in word-list ]
do
    command-list
done
```

其中，word-list 是一系列参数（或称参数表），中间以空格作为分隔符。对于参数表中的每一个参数值，重复执行一次 do 与 done 之间的命令。do 与 done 之间的循环体 command-list 是一个代码块，可以是一个命令、一组命令，也可以是各种控制结构语句，包括 for 循环结构语句。

利用 for 循环结构，只要控制条件为真，即可重复执行一组命令或代码块。在每次循环过程中，Shell 将会从给定的参数表中，依次取出一个参数值，并把参数值赋予给定的变量 var，然后重复执行 for 循环体中的代码块。

下面以一个简单的例子，说明 for 循环结构的执行过程。

```

for var in arg1 arg2 ... argN
```



```
echo $var  
done
```

在上述例子中，第一次循环时把 arg1 赋予 var 变量，相当于执行“var=arg1”变量赋值语句，然后运行“echo \$var”命令；第二次循环时执行“var=arg2”变量赋值语句，然后再次运行“echo \$var”命令；依此类推，第 N 次循环时执行“var=argN”变量赋值语句，然后运行“echo \$var”命令。经过 N 次循环之后，整个 for 循环语句执行结束。

下面的例子进一步说明了 for 循环结构的执行过程。当提交一个命令，为了弄清执行的命令究竟位于哪个目录，尤其当存在不同版本的同名命令，而这些命令又处于不同的命令检索路径时，可以使用 PATH 变量中设置的命令检索路径，找出第一个发现的命令位于哪一个目录，即可得到答案。PATH 变量中包含一系列由冒号分隔的目录，Shell 正是利用 PATH 变量，检索用户输入的命令。

```
$ cat where  
IFS="$IFS":  
flag=0  
for dirname in `echo ${PATH}`  
do  
    if test -x "${dirname}/$1"  
    then  
        echo "${dirname}/$1"  
        flag=1  
    fi  
done  
if [ $flag -eq 0 ]  
then  
    echo "The $1 is not exist."  
fi  
$
```

执行上述 Shell 脚本时的输出结果如下。

```
$ where echo  
/bin/echo  
$ where findf  
The findf is not exist.  
$
```

注意，如果 do 与 for 位于同一行上，则 do 之前应加一个分号分隔符，如下所示。

```
for var in [word-list] ; do
```

下面的 for 循环利用 diff 命令，对当前目录下的 C 文件 alpha.c、beta.c 和 gamma.c 与 /home/gqxing/src 目录中的同名文件进行比较。

```
for fname in alpha beta gamma ; do  
    diff ${fname}.c /home/gqxing/src/${fname}.c  
done
```

参数表中的参数也可以包含元字符，故也可以利用 Shell 的“\*”、“?”或“[...]”等通配符建立参数表。例如，可以利用元字符重写先前的 for 循环语句，比较当前目录与 /home/gqxing/src 目录中的每一个 C 文件，代码如下。

```
for fname in *.c  
do  
    diff $fname /home/gqxing/src/$fname  
done
```

参数表中的每个元素还可以包含多个参数，当需要按组处理参数时，这种形式是非常有用的。在此情况下，可以使用 set 命令，强制解析参数表中的每一个元素，把其中的每个参数分配到一系列位置参数中。

例如，在下面的 for 循环语句中，参数表中的每个元素本身包含两个参数。

```
for planet in "Mercury 36" "Venus 67" "Earth 93" "Mars 142" "Jupiter 483"  
do
```

```
set $planet          # 解析变量 planet, 设置位置参数。
echo "$1 is $2,000,000 miles away from the sun."
done
```

此外，也可以使用命令替换生成 for 循环中的参数表。例如，假定因故需要修改当前目录及其子目录中所有文件的属主与同组属性，可以利用“ls -R”命令递归地列出各级目录中所有文件的特点，生成文件名参数，然后再使用 chown 和 chgrp 命令修改文件的属性，示例如下。

```
for fname in `ls -R`
do
  if [ -f $fname ]
  then
    chown gqxing "${fname}"
    chgrp gqxing "${fname}"
  fi
done
```

下面是另外一个使用命令替换生成 for 循环语句参数表的例子，这个 Shell 脚本的功能是从指定（或当前）目录中检索文件的大小超过 1MB 的文件（注意，由于 find 命令“-size”选项的参数值后面未加任何后缀，故默认的数据单位是 512 字节的数据块，而 ls 命令“-s”选项的数据单位是 1KB）。

```
$ cat bigfile
#!/bin/bash
dir=${1-'pwd'}
echo "Big file(in 1KB-block) under directory \"\$dir\""
cd ${dir}
for i in `find . -size +2048 -print'
do
  if [ -f "${i}" ]
  then
    ls -s "${i}"
  fi
done
exit 0
$ bigfile /boot
Big file(in 1KB-block) under directory "/boot"
7728 ./initrd.img-2.6.24-18-generic
7732 ./initrd.img-2.6.24-19-generic
7192 ./initrd.img-2.6.24-16-generic.bak
7728 ./initrd.img-2.6.24-19-generic.bak
7324 ./initrd.img-2.6.24-18-generic.bak
1864 ./vmlinuz-2.6.24-16-generic
1884 ./vmlinuz-2.6.24-18-generic
1884 ./vmlinuz-2.6.24-19-generic
7708 ./initrd.img-2.6.24-16-generic
$
```

在 for 循环语句中，“in word-list”部分可以省略，在此情况下，for 循环语句需要使用运行 Shell 脚本时提供的命令行参数作为参数表。这种 for 循环结构尤其适用于编写一种通用的 Shell 脚本，针对给定的任何参数，执行同样一组处理命令。其简化的语法格式如下。

```
for variable
do
  command-list .
done
```

假定公司有一个员工的电话号码簿，其数据内容如下。

```
$ cat phonebook
Cathy   617-495-1585      Boston
Jim     650-723-2300      Stanford
Ruth    516-367-8800      New York
Tom     410-516-8000      Baltimore
$
```

可以写出下列 Shell 脚本，用于查阅任何给定员工的信息。



```
$ cat findname.sh
#!/bin/bash
for name
do
    if grep $name phonebook
    then
        :
    else
        echo "$name is not in the phonebook."
    fi
done
$
```

例如，要查阅 Cathy、Jim 和 XXX 的电话号码，可以按下列方式运行 Shell 脚本。

```
$ findname.sh Cathy Jim XXX
Cathy      617-495-1585      Boston
Jim       650-723-2300      Stanford
XXX is not in the phonebook.
$
```

整个 for 循环语句的输出可以重定向或通过管道输出到一个文件、命令或一组命令。根据这个特点，可以利用管道和 sort 排序工具，对上述的 bigfile 脚本做少许改进，把整个 for 循环的输出作为 sort 命令的输入，使得 Shell 脚本能够按文件的大小从大到小排序，然后顺序列出指定目录（或当前目录）中大于 1MB 的文件。示例代码如下。

```
$ cat bigfile2
#!/bin/bash
dir=${1-'pwd'}                                # 使用指定的或当前目录
cd $dir
echo "Big files(in 1KB-block) under directory \'$dir\'"
find . -size +2048 2>/dev/null |   # 找出容量大于 1MB 的文件
while read file
do
    if [ -f "${file}" ]
    then
        ls -s "${file}"
    fi
done | sort -rn                                # 按文件容量从大到小排序
exit 0
$ bigfile2 /boot
Big files(in 1KB-block) under directory "/boot"
7732 ./initrd.img-2.6.24-19-generic
7728 ./initrd.img-2.6.24-19-generic.bak
7728 ./initrd.img-2.6.24-18-generic
7708 ./initrd.img-2.6.24-16-generic
7324 ./initrd.img-2.6.24-18-generic.bak
7192 ./initrd.img-2.6.24-16-generic.bak
1884 ./vmlinuz-2.6.24-19-generic
1884 ./vmlinuz-2.6.24-18-generic
1864 ./vmlinuz-2.6.24-16-generic
$
```

## 8.4 while 循环语句

while 语句的功能是根据一定的条件，循环执行一组特定的命令。while 循环结构在循环体的前部测试执行条件。只要控制条件的测试结果为真（返回值为 0），便继续执行循环体中的命令，否则立即结束整个 while 循环。与 for 循环相比，while 循环更适合循环次数事先不确定的情况，其语法格式如下。

```
while [ condition-is-true ]
do
    command-list
done
```

下面是一个简单的 while 循环结构，其目的是求出 1~100 之间所有整数的总和。当 \$Number 变量的值大于 100 时，Shell 将会跳到 done 后面的 echo 语句开始执行，输出 \$sum 变量的值，即 1~100 之间所有整数的总和，其代码如下。

```
$ cat sum100
number=1
sum=0
max=100

while [ "$number" -le "$max" ]
do

    sum='expr $sum + $number'
    number='expr $number + 1'
done
echo $sum
$
```

同 for 循环一样，如果 do 与 while 位于同一行上，do 与测试条件之间也应加一个分号分隔符，如下所示。

```
while [ condition-is-true ] ; do
```

在 while 循环结构中，测试条件可以是一个简单的 test 语句，也可以是由多个条件表达式与逻辑运算符组成的组合测试语句；可以是一个普通的 Linux 命令，也可以是一组命令。循环体中可以包含任何命令语句，当然也包括任何结构语句。

在 while 循环结构中，当控制条件由多个命令语句组成时，实际上只有最后一个命令语句的出口状态才能决定是否继续执行循环体。也就是说，while 语句仅检查 do 之前最后一条命令的返回值，如果返回值为 0，则继续执行 do 与 done 之间的代码体；如果返回值大于 0，Shell 将会终止执行整个 while 循环语句，然后立即跳转并执行紧随 done 之后的命令语句。

在下面的例子中，用作 while 循环判断条件的第一个语句，即 echo 命令总是能够正常地运行。而 read 命令的特性则恰好可以用作 while 循环结构的判断条件：当读取一个 Ctrl-D 键时，read 命令将会返回一个非 0 的出口状态，因而可用于结束 while 循环。

```
$ cat lookup
while
    echo "\nEnter name to searched (Press Ctrl-D to end): \c"
    read name
do
    if test "${name}" = ""
    then
        continue
    else
        lookup.sql "${name}"      # 这里的 lookup.sql 是一个 MySQL 脚本，用于检索数据库
    fi
done
exit 0
$
```

如果在关键字 done 后面增加一个重定向符号 “<”，可以将 while 循环结构的标准输入重定向到一个文件。另外，while 循环结构的标准输入也可以由一个管道提供。

下面是一个利用管道为 while 循环结构提供数据，由 while 循环体逐行读取 cat 命令输出的数据，并对数据进行加工处理的例子（这里的处理仅为显示读取的数据）。

```
$ cat check
#!/bin/bash
```



```
line_num=1
cat $1 |
while read text
do
    echo "${line_num}: ${text}"
    line_num='expr ${line_num} + 1'
done
exit 0
$
```

作为测试数据的文件内容如下（选自泰戈尔《Stray Birds》）。

```
$ cat stray.birds
"What language is thine, O Sea?"
"The language of eternal question."
"What language is thy, Osky?"
"The language of eternal silence."
$
```

运行上述脚本后，输出结果如下。

```
$ check stray.birds
1: "What language is thine, O Sea?"
2: "The language of eternal question."
3: "What language is thy, Osky?"
4: "The language of eternal silence."
$
```

## 8.5 until 循环语句

until 语句的功能是在一定的条件下，循环执行一组特定的命令。until 循环结构也是在循环体的前部测试循环控制条件，但与 while 循环结构不同的是，until 循环结构测试的是终止条件。如果控制条件的测试结果为真（返回值为 0），则立即结束整个 until 循环；如果控制条件的测试结果为假（返回值大于 0），便继续执行循环体中的命令，直至控制条件的测试结果为真，这一点恰与 while 循环相反。until 语句的语法格式如下。

```
until [ condition-is-true ]
do
    command-list
done
```

同 for 和 while 循环一样，如果 do 与条件测试语句位于同一行上，则中间需要加分号分隔符，如下所示。

```
until [ condition-is-true ] ; do
```

下面是一个采用 Euclid 算法，利用 until 循环求取最大公约数的例子。首先，选取第一个参数作为被除数并赋予 dividend 变量，第二个参数作为除数赋予 divisor 变量。在 until 结构语句的每一次循环过程中，再把当前的除数赋予 dividend 变量，余数赋予 divisor 变量，作为下一次运算的因子。重复上述运算，直至余数为 0，最后一个 dividend 即为最大公约数，整个循环运算过程结束。以下是该例子的代码。

```
$ cat gcd
#!/bin/bash
if [ $# -ne 2 ]
then
    echo "Usage: 'basename $0' Number#1 Number#2"
    exit 1
fi
dividend=$1
divisor=$2
remainder=1
```

```

until [ "$remainder" -eq 0 ]
do
    let "remainder = $dividend % $divisor"
    dividend=$divisor
    divisor=$remainder
done
echo "The greatest common divisor of $1 and $2 = $dividend"
exit 0
$ gcd 7856 9980
The greatest common divisor of 7856 and 9980 = 4
$
```

## 8.6 select 循环语句

select 循环结构源于 Korn Shell，主要用于建立选择菜单。select 菜单的最大特点是，编程人员只需提供作为菜单选择项的参数表，菜单的组织与表现形式则由 select 语句负责实现。select 语句的语法格式如下。

```

select variable [in list]
do
    command-list
    break
done
```

执行 select 语句时，Shell 将提示用户根据参数表中的选择项，通过输入相应的数字做出选择。默认情况下，select 语句使用 PS3 环境变量的值（#?）作为命令提示符。如果需要，编程人员可在 select 语句之前，赋予 PS3 环境变量一个新的值，以定制命令提示符。

下面是一个使用 select 语句建立菜单的例子。

```

$ cat menu
#!/bin/bash
PS3='Choose your favorite book: ' # 设置 select 语句使用的命令提示符
echo
select book in "The Da Vinci Code" "Pride and Prejudice" "Wuthering Heights" "The Old
Man and the Sea"
do
    echo
    echo "Your favorite book is $book."
    break                      # 使用 break 语句退出 select 循环
done
exit 0
$
```

运行上述 Shell 脚本时，其处理结果如下。

```

$ menu

1) The Da Vinci Code      3) Wuthering Heights
2) Pride and Prejudice   4) The Old Man and the Sea
Choose your favorite book: 1

Your favorite book is The Da Vinci Code.
$
```

在 select 语句中，“in list”部分可以省略。如果省略了参数表，select 语句需要使用运行 Shell 脚本时提供的命令行参数作为参数表。这种处理方式可使编写的 Shell 脚本更通用，更灵活。

```

$ cat menu2
#!/bin/bash
PS3='Choose your favorite book: '
echo
select book
```



```
do
    echo
    echo "Your favorite book is $book."
    break
done
exit 0
$ menu2 "The Da Vinci Code" "Pride and Prejudice" "Wuthering Heights" "The Old Man and the Sea"
1) The Da Vinci Code      3) Wuthering Heights
2) Pride and Prejudice   4) The Old Man and the Sea
Choose your favorite book: 2

Your favorite book is Pride and Prejudice.
$
```

下面是一个在函数中利用 select 语句实现菜单的例子，其效果与用法同第一个例子完全一样。

```
$ cat menu3
#!/bin/bash
PS3='Choose your favorite book: '
choice()
{
    echo
    select book
    do
        echo
        echo "Your favorite book is $book."
        break
    done
}
choice "The Da Vinci Code" "Pride and Prejudice" "Wuthering Heights" "The Old Man and the Sea"
exit 0
$
```

## 8.7 嵌套的循环

嵌套的循环意味着循环体中包含内嵌的循环语句。外层循环体每执行一次循环，内层循环体就要执行一个完整的循环，直至外层循环结束。当然，也可以利用循环体中的 break 语句，中断相应循环体的继续处理。

下面的例子就是利用嵌套的循环，列出指定目录（或当前目录）的目录层次结构树。

```
$ cat tree.sh
#!/bin/bash
# Written by Rick Boivie. Revised and simplified
# by Jordi Sanfeliu (patched by Ian Kjos).
# Last revised by GQ Xing.
search () {
for dir in `echo *`                                # 'echo *'用于列出当前目录下的所有文件
do
    if [ -d "$dir" ] ; then                         # 如果$dir是一个目录
        dirlevel=0                                    # 临时变量，用于记录目录层次
        while [ $dirlevel != $1 ]                     # 记录内部嵌套的循环
            do
                echo -e "| \c"                          # 表示目录层次，中间不换行
                dirlevel=`expr $dirlevel + 1`           # 增加目录计数
            done
        if [ -L "$dir" ] ; then                      # 如果目录是一个符号链接
            echo "|--$dir" `ls -l $dir | sed 's/^.*'$dir' //\'` # 显示横向连接符号并列出原目录及其指向的目录名
            # tt='ls -l $dir'
            # echo "|--'$tt | cut -f9-11 -d' ''"
        else
            echo "|--$dir"                            # 显示横向连接符号和目录名
            numdirs=`expr $numdirs + 1`               # 增加目录计数
        fi
    fi
done
```

```

if [ -x "$dir" ] ; then          # 检查目录执行权限
    cd "$dir"
    search 'expr $1 + 1'         # 递归地调用自己
    cd ..
else
    echo "Changed to above directory is denied"
fi
fi
done
}

cd ${1:-'pwd'}                  # 移动到指定目录或当前目录
echo "Initial directory = 'pwd'"
numdirs=0
search 0
echo "Total directories = $numdirs"
exit 0
$ tree /var
Initial directory = /var
|-- autofs
|   |-- misc
|   |-- backups
|   |-- cache
|       |-- apache2
|           |-- mod_disk_cache
|           |-- app-install
|           |-- apt
|               |-- archives
|                   |-- partial
...
$
```

## 8.8 循环控制与辅助编程命令

本节将主要介绍影响循环行为的控制命令与辅助编程命令语句。

### 8.8.1 break 和 continue 命令

break 和 continue 循环控制命令基本上等同于 C 或其他编程语言中 break 和 continue 语句的功能。break 命令用于跳出相应的循环体，终止循环体的继续执行。而 continue 命令则用于越过循环体中尚未执行的命令语句，结束本次循环，直接跳转到下一次循环迭代。

break 命令的主要用途是立即停止执行当前的循环体，然后跳转到循环体外（即关键字 done 后面）的命令语句处开始继续执行，其语法格式如下。

**break [n]**

break 命令后面可以附加一个数字参数。一个简单的 break 命令只是终止执行最内层的循环体。但是，如果 break 命令后面有一个数字 n，则使用“break n”命令能够跳出 n 层循环体，将控制转移到外部第 n 个关键字 done 后面的命令语句处开始继续执行。

continue 命令的主要功能是越过本次循环中余下的所有命令语句，从循环体的第一条命令语句开始继续执行循环体，其语法格式如下。

**continue [n]**

类似于 break，continue 命令也可以附带一个数字参数。一个简单的 continue 命令只是终止执



行当前循环中尚未执行的命令语句，继续执行下一次循环。如果 continue 命令后面有一个数字 *n*，则使用“continue *n*”命令能够终止执行当前循环，从嵌套的外部第 *n* 层循环体的起始命令语句开始继续执行下一次循环。

注意，break 和 continue 命令只能用于 for、while 和 until 等循环语句的循环体中，位于关键字 do 与 done 之间。

下面的 Shell 脚本说明了 break 和 continue 命令在 while 循环中的作用，其中的 break 和 continue 命令用于控制用户的输入过程。这个 Shell 脚本的功能是提示用户连续地输入由空格分隔的字符串数据，并把接收的数据存储到一个文件中，直至用户输入 quit 时，退出 while 循环。接着，对文件中的数据进行排序，最后把已排序的数据再存回原文件中。

```
$ cat whiledo
#!/bin/bash
> datafile
while :
do
    echo -e "Please enter data (enter 'quit' to end): \c"
    read response
    case "$response" in
        quit) break ;;          # 停止输入，结束 Shell 脚本的运行
        '')  continue ;;       # 继续读取下一个数据
        *)   echo "$response" >> datafile
    esac
done
sort -o datafile datafile      # 对文件中的数据进行排序，排序后仍保存到原文件
exit 0
$
```

下面是一个使用 break 和 continue 命令直接跳出多层循环，或者从外层循环体继续执行下一次循环的例子。

```
$ cat levels
#!/bin/bash
while :
do
    echo "Entering level 1"
    while :
    do
        echo -e "\tEntering level 2"
        while :
        do
            echo -e "\t\tEntering level 3"
            echo -e "\t\t\tInput c to continue \c"
            echo -e "or b to break: \c"
            read cmd
            echo -e "\t\t\tHow many levels? \c"
            read level
            case ${cmd} in
                [bB]*) break ${level} ;;
                [cC]*) continue ${level} ;;
            esac
            echo -e "\t\tLeaving level 3"
        done
        echo -e "\tLeaving level 2"
    done
    echo "Leaving level 1"
done
exit 0
$ levels
Entering level 1
    Entering level 2
```

```

Entering level 3
Input c to continue or b to break: c
How many levels? 2
Entering level 2
    Entering level 3
        Input c to continue or b to break: B
        How many levels? 2
Leaving level 1
Entering level 1
    Entering level 2
        Entering level 3
            Input c to continue or b to break: c
            How many levels? 3
Entering level 1
    Entering level 2
        Entering level 3
            Input c to continue or b to break: b
            How many levels? 3
$
```

注意，“continue N”命令的组织、应用和理解都比较困难，应尽量避免使用。

### 8.8.2 true 命令

true 命令除了用于返回一个表示测试成功的返回值 0 之外，不执行其他任何动作。因此，true 命令经常被用于无限循环的循环结构。

事实上，true 与冒号 “:” 命令可以相互替换，但 “:” 命令是一个 Shell 内置命令，其执行速度要快于 Linux 系统提供的外部命令 true。

### 8.8.3 sleep 命令

sleep 命令可使 Shell 暂时休眠一定的时间，然后再执行下一个命令。因此，要在执行两个命令之间暂停一定的时间，然后再执行，可以在 Shell 脚本中使用 sleep 命令，其语法格式如下。

```
sleep [n]
```

其中，选用的参数 n 是暂停继续执行的时间间隔（以秒为计量单位）。sleep 命令经常用于保持屏幕输出内容的暂时稳定，例如，要检查当前正在复制的文件大小的变化情况，或者查看某个进程是否继续活动，以便得出某种判断，可以使用下列命令。

```

$ while true
> do
> ls -l fname
> sleep 5
> done
$
```

或

```

$ while :
> do
> ps -ef | grep proc_name
> sleep 5
> done
$
```

### 8.8.4 shift 命令

shift 命令用于修改位置参数的值。按照命令参数指定的数量，所有的位置参数逐次向左平移。最左边的位置参数逐一移走，最右边的位置参数逐一左移后清除。同时，位置参数的总数也相应地减少。\$0 是脚本文件的名字，在左移的过程中不受影响。shift 命令语句的语法格式如下。



```
shift [n]
```

其中，n是左移的数量，默认值为1。每执行一次“shift n”命令语句，整个位置参数都会左移n个位置，表示位置参数总数的\$#变量值相应地减少n，表示全部位置参数的\$\*或\${@}变量也相应地删除最左边的n个参数。

下面的例子说明了shift命令的典型用法。其中，while语句总是对\${1}变量进行测试。由于使用了shift语句，每次循环之后，\${1}的变量值都会发生变化。case语句依次解析每个命令行参数，并输出相应的信息。这个脚本假定“-a”选项之后必须指定文件名参数，利用while、case和shift语句，连续地检测位置参数，直至匹配“-a something”或“-b”，然后执行相应的代码。

```
$ cat options
while test "${1}" != ""
do
    case "${1}" in
        -a*) fname='echo ${1} | sed 's/-a//'''
              echo "Option a selected"
              echo "File name is ${fname}"
              option_a="yes"
              ;;
        -b*) echo "Option b selected"
              option_b="yes"
              ;;
    esac
    shift
done
$ options -aindex
Option a selected
File name is index
$ options -b
Option b selected
$ options -aindex -b
Option a selected
File name is index
Option b selected
$
```

## 8.8.5 getopt 命令

getopt命令主要用于解析命令行选项，检查选项的合法性，为Shell编程提供方便，其语法格式如下。

```
set -- `getopt optstring $*`
```

其中，optstring是一个包含所有合法命令选项的字符串。如果选项字母之后附有一个冒号“：“，则意味着相应的选项需要提供一个选项参数（选项字母与选项参数之间既可以存在空格分隔符，也可以直接连接在一起）。“--”特殊选项用于界定命令选项的结束。不管存在与否，Shell都会在所有的普通选项之后而放置一个“--”选项。

下面是一个取自网上的应用实例，其主要功能是根据用户提供的参数，利用ftp实现文件的自动下载。其中说明了ftwget脚本是怎样利用getopt命令处理其合法选项“-h”、“-d”、“-c”、“-f”和“-m”，解析和验证命令行参数的。

```
$ cat ftwget
#!/bin/bash
CMDFILE=/tmp/ftp.$$
TMPFILE=/tmp/ftp2.$$
usage="Usage: $0 [-h rhost] [-d rdir] [-c ldir] [-f rfile:lfile] [-m fpattern]"
ftopts="-i -n -v"
set -f
set -- 'getopt h:d:c:f:m: $*'
# echo $*
if [ "$#" -lt 2 -o "$?" -ne 0 ]; then
```

```

echo $usage
exit 1
fi
trap 'rm -f ${CMDFILE} ${TMPFILE}; exit' 0 1 2 3 15
echo "user gqxing 123456" > ${CMDFILE}
for i in $*
do
    case $i in
        -h) remhost=$2; shift 2;;
        -d) echo cd $2 >> ${CMDFILE};
            echo pwd >> ${CMDFILE};
            shift 2;;
        -c) echo lcd $2 >> ${CMDFILE}; shift 2;;
        -m) echo mget "$2" >> ${TMPFILE}; shift 2;;
        -f) f1='echo "$2" | cut -d: -f1'; f2='echo "$2" | cut -d: -f2';
            echo get ${f1} ${f2} >> ${TMPFILE}; shift 2;;
        --) shift; break;;
    esac
done
if [ $# -ne 0 ]; then
    echo $usage
    exit 2
fi
echo quit >> ${TMPFILE}
cat ${TMPFILE} >> ${CMDFILE}
ftp ${ftpports} ${remhost:-169.254.78.100} < ${CMDFILE}
cat ${CMDFILE} >> ftplog
rm -f ${CMDFILE} ${TMPFILE}
exit 0
$ ftpget -d docs -c /tmp -f design
Connected to 169.254.78.100.
220 (vsFTPd 2.0.6)
331 Please specify the password.
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
250 Directory successfully changed.
257 "/home/gqxing/docs"
Local directory now /tmp
local: design remote: design
200 PORT command successful. Consider using PASV.
150 Opening BINARY mode data connection for design (308522 bytes).
226 File send OK.
308522 bytes received in 0.01 secs (53802.0 kB/s)
221 Goodbye.
$
```

## 8.8.6 getopt 命令

getopts 命令是对 getopt 命令的更新和替代品。最初推出 getopts 命令的本意是取代 getopt 命令，但实际上这两个命令一直共存于 UNIX 以及 Linux 系统中。getopts 命令用于获取以减号 “-” 或加号 “+” 为起始字符的命令选项、选项参数和命令参数，其语法格式如下。

**getopts opts string [arg...]**

在编写 Shell 脚本时，最好能够养成使用 getopts 或 getopt 命令获取命令行选项、选项参数和命令参数的习惯。利用 getopts 命令，用户可在运行 Shell 脚本时以任何顺序输入命令选项。不带选项参数的命令选项既可以单独给出，也可以组合给出，但本身带有选项参数的命令选项需单独给出。

getopts 命令语句采用一个合法的选项字符串确定 Shell 脚本能够接受的命令选项。例如，下述 getopts 命令定义的合法选项字符串为 ior，表示 Shell 脚本能够接受的命令选项为 “-i”、“-o” 和 “-r”：

**getopts ior string**



每次调用 `getopts` 命令语句时，命令行中的选项就会依次被存入给定的变量 `string` 中。因此，Shell 脚本可以对变量 `string` 的值做适当的处理。

运行 Shell 脚本时，在最后一个命令选项与命令参数之间还可以插入一个位置参数“`--`”。这是一个标志，表示命令选项的结束和命令参数的开始。

如果在运行脚本时提供了非法的参数，`getopts` 命令将会输出下列形式的出错信息，同时把一个问号“`?`”字符赋予 `string` 变量。

```
script_name: illegal option -- option_letter
```

`getopts` 命令的返回值取决于是否能够成功地获取命令选项。如果读取了命令选项（包括非法的选项），`getopts` 将返回数值 0；否则，当遇到第一个位置参数“`--`”、第一个非选项参数（即命令参数）或命令行结束标志时，`getopts` 命令语句将返回一个非 0 数值。

下面的例子说明了怎样使用 `getopts` 命令语句处理简单的命令行选项。

```
$ cat getoptoptions
#!/bin/bash
LANG=C
while getopts abc var
do
    case ${var} in
        a) echo Option -a found ;;
        b) echo Option -b found ;;
        c) echo Option -c found ;;
    esac
done
exit 0
$ getoptoptions -b -a
Option -b found
Option -a found
$ getoptoptions -adc
Option -a found
getoptions: illegal option -- d
Option -c found
$
```

如果选项带有参数，则需要在相应的选项字母后面附加一个冒号“`:`”字符。在读取冒号之后，`getopts` 将会把选项后的选项参数字符串赋予 `OPTARG` 变量。注意，选项与选项参数之间可以有空格分隔符，也可以连接在一起。

如果一个选项要求提供选项参数而未提供，`getopts` 语句将会输出下列形式的出错信息。

```
script_name: option requires an argument - options_letter
```

`getopts` 语句执行结束时，`OPTIND` 变量的值将会指向第一个非选项参数的位置参数，即第一个命令参数。如果位置参数的值为“`--`”，则忽略之。为了便于脚本按惯例处理命令参数，应将 `$1` 指向第一个命令参数，`$2` 指向第二个命令参数，如此等等。因此，可以利用 `shift` 命令，使之左移适当的位置。按照上面的说明，只需左移“`$OPTIND-1`”个位置，即可使 `$1` 指向第一个命令参数。

在下面的例子中，Shell 脚本可接受 4 个选项，其中“`-o`”和“`-r`”选项要求提供相应的参数。在运行 Shell 脚本时，对于任何一个选项，既可以提供，也可以不提供。如果提供了“`-o`”或“`-r`”选项，则必须提供相应的参数。

```
$ cat getoptoptions2
#!/bin/bash
LANG=C
while getopts o:r:nt var
do
    case ${var} in
        o) output_file="${OPTARG}" ;;
```

```

r) report_file="\${OPTARG}" ;;
n) number_option="yes" ;;
t) title="no" ;;
\?) exit 2 ;;
esac
done
echo "Output file = \${output_file}"
echo "Report file = \${report_file}"
echo "Numbering option = \${number_option:-no}"
echo "Title option = \${title:-yes}"
echo "Arguments before shift: \$(*)"
shift `expr \$OPTIND - 1'
echo "Arguments after shift: \$(*)"
exit 0
$ getoptns2 -tn -o ofile -r rfile unit.12
Output file = ofile
Report file = rfile
Numbering option = yes
Title option = no
Arguments before shift: -tn -o ofile -r rfile unit.12
Arguments after shift: unit.12
$ getoptns2 -tn *
Output file =
Report file =
Numbering option = yes
Title option = no
Arguments before shift: -tn unit.01 unit.02 unit.03
Arguments after shift: unit.01 unit.02 unit.03
$ getoptns2 unit.12
Output file =
Report file =
Numbering option = no
Title option = yes
Arguments before shift: unit.12
Arguments after shift: unit.12
$ getoptns2 -tn -o ofile -r
getoptns2: option requires an argument -- r
$
```

## 8.9 循环语句的 I/O 重定向

作为一个整体，while、until 与 for 循环结构语句中的循环体（包括 while 或 until 与 do 之间的测试条件语句）也可以实现 I/O 重定向。对于循环结构语句来讲，所谓的 I/O 重定向主要是重定向循环体中的标准输入。当然，也可以利用 I/O 重定向和管道的方法，把循环体中的输出结果保存到文件中，以备将来查阅。

另外，函数也可以采用同样的处理方式，实现 I/O 重定向。要实现标准输入的 I/O 重定向，可在循环结构语句的结束标志（如关键字 done）后面附加重定向符号“<”和适当的输入文件。

### 8.9.1 while 循环的 I/O 重定向

对于 while 循环结构语句而言，下列语法格式表示，循环体中需要的所有输入数据均取自指定的输入文件。

```

while [ condition-is-true ]
do
  command-list
done < datafile
```



下面的例子是取自一个安装脚本的部分代码片段。由于 while 循环结构语句的标准输入已重定向到/etc/fstab 文件，故循环体中 read 语句读取的数据均来自/etc/fstab 文件。在 while 循环中，每执行一次 read 语句，即从/etc/fstab 文件中读取一行数据，并把有效数据分别赋值到 fsdev、mntp、fstype、mntopts、fsdump 和 fpassno 等 7 个变量中。

这个脚本片段的目的是检查系统中是否存在独立的/var 文件系统。如果存在/var 文件系统，则利用 fsck 命令检测并修复/var 文件系统。

```
# cat checkfs
#!/bin/sh
while read fsdev mntp fstype mntopts fsdump fpassno
do
#  if [ "${mntp}" = "/var" ]      # 如果发现/var 文件系统，检测并安装
#  if [ "${mntp}" = "/mnt" ]      # 为了验证，这里改为/mnt
#  then
#    echo "The $mntp file system (${fsdev}) is being checked."
#    fsck -t ${fstype} -y ${fsdev}
#    break
#  fi
done < /etc/fstab           # 把 read 命令的标准输入重定向到/etc/fstab 文件
exit 0                      # 为了验证，这个文件也做了修改
# checkfs
The /mnt file system (/dev/fd0) is being checked.
fsck 1.40.8 (13-Mar-2008)
e2fsck 1.40.8 (13-Mar-2008)
/dev/fd0: clean, 15/184 files, 759/1440 blocks
#
```

## 8.9.2 until 循环的 I/O 重定向

until 循环结构的 I/O 重定向与 while 循环结构几乎完全雷同，下面的脚本也是根据上述例子改写的，除了采用 until 循环结构，其功能完全一样。

```
# cat checkfs2
#!/bin/sh
# 直至遇到/etc/fstab 文件结束标志才终止 until 循环
until ! read fsdev mntp fstype mntopts fsdump fpassno
do
#  if [ "${mntp}" = "/var" ]      # 如果发现 "/var" 文件系统，执行 fsck 命令，
#  then                          # 然后退出 until 循环
#    echo "The $mntp file system (${fsdev}) is being checked."
#    fsck -t ${fstype} -y ${fsdev}
#    break
#  fi
done < /etc/fstab           # 把 read 命令的标准输入重定向到/etc/fstab 的文件
exit 0
#
```

## 8.9.3 for 循环的 I/O 重定向

for 循环结构也可以实现 I/O 重定向，使循环体中的输入输出命令能够直接读取或写入指定的文件。许多日常维护任务就是利用 I/O 重定向，在不修改 Shell 脚本的情况下，只需修改输入文件，即可实现特定的系统维护目的。

在下面的例子中，我们的目的是找出指定目录或默认目录中两个月以来未曾被访问过的文件。为了保存查询的结果，我们稍做变化，利用 tee 命令实现 for 循环体标准输出的 I/O 重定向，在显示检索结果的同时，把输出数据保存到/tmp/filelist 文件中。示例代码如下。

```

$ cat oldfile
#!/bin/bash
dir=$(ls -d ./*)           # 使用指定的目录或当前目录
cd ${dir}
echo "Old files under directory \"${dir}\""
for file in `find . -type f -atime +60 -print 2>/dev/null`
do
    if [ -f "${file}" ]
    then
        ls -l "${file}"
    fi
done | tee /tmp/filelist      # 显示并保存查询结果
exit 0
$ oldfile /boot
Old files under directory "/boot"
.....
-rw-r--r-- 1 root root 1904248 Apr 11 2008 ./vmlinuz-2.6.24-16-generic
-rw-r--r-- 1 root root 1921528 May 29 10:39 ./vmlinuz-2.6.24-18-generic
-rw-r--r-- 1 root root 422607 Apr 11 2008 ./abi-2.6.24-16-generic
-rw-r--r-- 1 root root 422667 May 29 10:39 ./abi-2.6.24-18-generic
-rw-r--r-- 1 root root 7877430 Jul 9 10:32 ./initrd.img-2.6.24-16-generic
$
```

此外，if-then 条件转移语句中的代码块也可以实现 I/O 重定向，只是实际意义并不大，除非把 if-then 代码块放到一个循环结构中。

总之，在 Shell 脚本中，灵活地利用 I/O 重定向，能够生成必要的运行报告和日志文件，完整地记录脚本的处理过程，对脚本的调试和以后的档案留存都有重要的意义。

## 8.10 Here 文档

here 文档是一种具有特殊用途的代码块，是 I/O 重定向的一种特例。here 文档采用 I/O 重定向的方法，把一系列需要从键盘上输入的命令，模拟人工输入方式，一行一行地提交给交互式应用程序或命令，如 ftp 和 MySQL 数据库的 mysql 等。here 文档的语法格式如下。

```

program <<LimitString
command1
command2
....
commandN
LimitString
```

其中，特殊的 I/O 重定向符号“<<”与“LimitString”表示 here 文档的开始，单独另起一行的第二个“LimitString”表示 here 文档的结束。“LimitString”是启动指定程序后中间一系列交互命令的分界符。I/O 重定向的效果是把 here 文档列出的命令提交给交互式应用程序或命令的标准输入。here 文档的作用相当于执行下列命令。

```
interactive-program < command-file
```

其中，command-file 包含一系列需要人工输入的命令或数据：

```

command1
command2
....
commandN
```

下面是一个利用 here 文档维护数据库的例子。当 MySQL 数据库从一个服务器迁移到另一个服务器时，在安装 MySQL 数据库软件之后，首先需要赋予用户建立数据库的权限，然后还需要把先前利用 mysqldump 命令备份的数据库、数据库表及其数据加载到新的 MySQL 数据库中。为



了简化人工操作步骤，可以使用下列 Shell 脚本实现数据库的重建。

```
$ cat makedb
#!/bin/bash

mysql -u root -psqladmin <<EOF
USE mysql;
GRANT ALL PRIVILEGES ON books.* TO gqxing@"localhost" IDENTIFIED BY 'alb2c3';
FLUSH PRIVILEGES;
quit
EOF
mysql -u gqxing -palb2c3 books < ./tmp/bookdump.sql
exit 0
$
```

在上述 here 文档中，GRANT 语句表示赋予用户 gqxing 在当前的系统中创建数据库的权限。here 文档之后的 mysql 命令用于导入先前备份的数据库、数据库表及其数据，有关内容请参见第 24 章。

注意，选择字符串分界符“LimitString”时应确保其在 Shell 脚本中是唯一的，至少不应在其界定的一系列命令中间出现，以免产生混淆。

下面的例子说明怎样利用 here 文档运行 vim 编辑器，模拟 vim 编辑器的交互过程，输入“i”命令和 ESC 键，插入两行数据，最后把编辑的数据内容写入指定的文件。

```
$ cat emuvim
#!/bin/bash
if [ -z "$1" ]
then
    echo "Usage: 'basename $0' filename"
    exit 1
fi

vim $1 <<EOF
i
I cannot choose the best.
The best choose me.
^[
zz
EOF
exit 0
$
```

使用下列命令运行 Shell 脚本时，即可得到一个自动编辑的文本文件。

```
$ emuvim fname
Vim: Warning: Input is not from a terminal
$ cat fname

I cannot choose the best.
The best choose me.

$
```

注意，在上述 Shell 脚本中，“^[”是一个字符，表示 Esc 键。为了在 vim 编辑器中输入 Esc 键，可以先按 Ctrl-V 键，接着再按 Esc 键，即可得到一个单字符的“^[”字符。此外，上述编辑命令形式将会在实际数据内容前后各加一个空行。为了避免这种情况出现，可以对上述的编辑形式稍加修改，最终得到一个只含两行数据内容的文件。改造的结果如下。

```
vim $1 <<EOF
iI cannot choose the best.
The best choose me.^[
zz
EOF
```

对于非交互式的实用程序或命令，有效地利用 here 文档，有时也会取得非常好的效果。下面是一个利用 here 文档和 cat 命令输出多行信息的例子。

```
cat <<End-of-message
=====
The system ${NODENAME} will be shut down in ${time}.
Please logoff as soon as possible.
=====
End-of-message
```

事实上，如果单纯为了显示多行数据，大可不必使用 here 文档，只需使用简单的 echo 命令，即可实现多行信息的输出。here 文档的主要用途在于模拟执行各种交互式的程序或命令，示例如下。

```
echo "
=====
The system ${NODENAME} will be shut down in ${time}.
Please logoff as soon as possible.
====="
```

here 文档要求其中的输入数据，尤其是作为结束标志的字符串分界符“LimitString”必须位于单独另起一行的起始位置。为了使脚本的可读性更强，编程人员喜欢在数据行之前加制表符，使得脚本错落有致。为了使 here 文档能够正确地读取数据，可在第一个字符串分界符之前增加减号“-”标志（如<<-LimitString），使之在读取数据时能够删除数据行前的制表符（但“-”标志并不影响空格，也不影响文本行中间的制表符），示例如下。

```
$ cat heredoc
#!/bin/bash
cat <<-ENDOFMESSAGE
    I live in you, you live in me;
    We are two gardens haunted by each other.
    Sometimes I cannot find you there,
    There is only the swing creaking, that you have just left,
    Or your favourite book beside the sundial.
ENDOFMESSAGE
exit 0
$ heredoc
    I live in you, you live in me;
    We are two gardens haunted by each other.
    Sometimes I cannot find you there,
    There is only the swing creaking, that you have just left,
    Or your favourite book beside the sundial.
$
```

here 文档也支持变量和命令替换，因而能够把不同的参数传递到 here 文档的代码体中，相应地改变 here 文档的输出。下面是一个在 here 文档中使用变量替换的例子。

```
$ cat upload.sh
#!/bin/bash
if [ -z "$1" ]
then
    echo "Usage: 'basename $0' Filename-to-upload"
    exit 1
fi
if [ ! -f "$1" ]
then
    echo "Specified file is not exist."
    exit 2
fi
fname='basename $1'                      # 删除文件名的目录部分
host="169.254.78.100"
dir="/home/gqxing/docs"

ftp -inv $host <<End-Of-Session      # "-n"选项禁止执行自动注册过程
user gqxing 123456
put $fname $dir/$fname
bye
End-Of-Session
```



```
exit 0
$ upload.sh design
Connected to 169.254.78.100.
220 (vsFTPd 2.0.6)
331 Please specify the password.
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
local: design remote: /home/gqxing/docs/design
200 PORT command successful. Consider using PASV.
150 Ok to send data.
226 File receive OK.
308522 bytes sent in 0.01 secs (57641.3 kB/s)
221 Goodbye.
$
```

要禁止在 here 文档中使用变量替换或命令替换，可以在第一个字符串分界符前后增加单引号或双引号，或者在第一个字符串分隔符前增加转义字符。

下面就是一个禁止 here 文档执行变量替换的例子。

```
$ cat greeting2
#!/bin/bash
name="${1:-Zhang}"
from="the author of this script"
cat <<'Endofmessage'      # 也可使用“cat <<"Endofmessage"或cat <<\Endofmessage”，
                           # 3者效果一样
Hello, $name.
Greetings to you, $name, from $from.
Endofmessage
exit 0
$ greeting2 wang

Hello, $name.
Greetings to you, $name, from $from.
$
```

禁止变量替换使 here 文档能够输出纯文字的文本，生成 Shell 脚本或其他程序（如 C 程序等）代码。例如，下面的代码用于生成另一个 Shell 脚本。

```
$ cat gen.sh
#!/bin/bash
OUTFILE=generated.sh      # 生成的 Shell 脚本的文件名
(
cat <<EOF
#!/bin/bash
echo "This is a generated Shell script."
echo "and will be named: $OUTFILE"
exit 0
EOF
) > $OUTFILE

if [ -f "$OUTFILE" ]
then
    chmod 755 $OUTFILE      # 使生成的 Shell 脚本能够被执行
else
    echo "Problem in creating file: \"$OUTFILE\""
fi
exit 0
$ gen.sh
$ cat generated.sh
#!/bin/bash
echo "This is a generated Shell script."
echo "and will be named: generated.sh"
exit 0
$
```

利用 here 文档和 “:” 命令，还可以注解掉 Shell 脚本中的代码，或者增加自包含的文档数据。这种代码注解技术可用于调试 Shell 脚本。编程人员只需增加两行标志代码，即可临时注解掉大块的 Shell 代码，从而避免了在每行代码前增加一个注解字符“#”，调试结束后再逐行地一一删除。其代码如下。

```
$ cat commentblock.sh
#!/bin/bash
: << COMMENTBLOCK
      # Shell 代码或文档
COMMENTBLOCK

: << DEBUGXXX
for file in *
do
    cat "$file"
done
DEBUGXXX
exit 0
$
```

注意，here 文档中的第二个字符串分界符必须占用单独一行，并位于行首，前后不能包括任何空白字符（包括空格和制表符等）。否则将会妨碍 Shell 的识别，导致无法预料的后果。

## 8.11 Shell 函数

同其他编程语言一样，Bash 等 Shell 也支持函数，尽管实现的功能有一定的限度。一个函数是一个子程序，其中利用一组代码块，实现特定的处理功能。与 Shell 脚本相比，函数经常被直接存储在内存中，因而执行速度要快于 Shell 脚本。

无论何处，只要存在一组需要重复执行的处理动作，或者针对不同的参数，能够获取相应的返回值，都可以考虑使用函数。函数的语法格式如下。

```
function_name( ) {
    command-list
}
```

或

```
function function_name {
    command-list
}
```

第一种函数形式是 C 程序员比较熟悉的。如同 C 语言一样，函数的左括号也可以出现在第二行上。注意，如同下列函数定义形式所示，如果整个函数定义与其中的命令位于同一行上，则左花括号之后和右花括号之前必须存在一个空格。同时，每个命令之后除附加分号之外，至少还必须保有一个空格。

```
function_name( ) { cmd1; cmd2; ....; cmdn }
```

在定义函数时，函数名不能与现有的变量同名。如果函数与变量同名，Shell 将会给出一个错误信息。此外，函数名与左圆括号之间必须连在一起，中间不能有空格。这个圆括号相当于告诉 Shell，这是一个函数定义。

例如，为了在一个 Shell 脚本中重复执行两个变量值的交换，可以定义下列函数。

```
exchange()          # 交换两个变量的值
{
    temp=${color1}
    color1=${color2}
    color2=$temp
    return
}
```



注意，在定义函数时，不能在函数体内使用 exit 命令，因为函数的执行与当前的 Shell 脚本同属一个进程。否则的话，当函数执行到 exit 命令时，将会终止整个 Shell 脚本的继续执行。

在函数定义中，表示整个函数执行结束的替代方法是利用 Shell 的内部命令 return 实现的。也就是说，函数中 return 语句的功能同 Shell 脚本中的 exit 命令一样，其本意是终止函数的执行。在 Shell 脚本中，只有在函数定义中才能使用 return 命令，但也并非必需。实际上，仅当期望函数返回一个数值时，才真正需要使用 return 语句。如果函数中未用到 return 命令，则函数运行结束时的返回值就是函数体中执行的最后一条命令的出口状态。

同 exit 语句一样，return 语句也可以带一个整数参数，以便在函数执行结束时返回给定的数值，同时将这个返回值赋予 "\$?" 变量。编程人员可以使用 return 语句显式地指定返回值。否则，函数的返回值就是函数中执行的最后一条命令的出口状态（如果命令执行成功，返回 0；否则，返回一个非 0 的错误代码）。return 语句的语法格式如下。

```
return [n]
```

在 Shell 脚本中，可以通过 "\$?" 变量引用函数的返回值。这一点与 C 等语言中的函数类似；但与 C 等语言不同的是，在调用 Shell 函数时，只需直接写出函数的名字，然后给出必要的参数，而无须使用圆括号。当遇到一个函数名时，Shell 首先会检查给定的名字是否为一个函数，如果不是，再利用 PATH 环境变量按命令检索。

如同常规的函数调用或 Shell 脚本的运行方式一样，调用 Shell 函数时也可以提供参数。而函数可以处理传递给自己的参数，最终把处理结果返回 Shell 脚本，以便在脚本中做进一步的处理，示例如下。

```
function_name $arg1 $arg2 ... $argN
```

下面是一个求取指定整数范围内所有素数的例子。对于任何给定的整数，我们首先排除偶数，仅对奇数进行处理。通过一个 while 循环，主程序把 3、5 直至小于或等于给定整数的奇数依次传递给 prime 函数，prime 函数则以给定的整数参数作为被除数，依次除以 3、5，直至整数参数的平方根（取整）为止。如果中间出现余数为 0 的情况则返回 0（说明给定的整数参数不是素数），否则返回 1（说明给定的整数参数为素数）。

```
$ cat prime
#!/bin/bash
prime()
{
    declare -i num lim rem
    num=3
    lim='echo $1 | awk '{print sqrt($1)}''
    while [ $num -le $lim ]
    do
        rem='expr $1 % $num'
        if [ $rem -eq 0 ]
        then
            return 0
        fi
        num='expr $num + 2'
    done
    return 1
}
declare -i number
while true
do
    echo -e "\nEnter a number: \c"
    read number
    if [ $number -lt 2 ]
    then
        exit 1
    fi
    prime
done
```

```

fi
echo "The prime number list within $number"
echo -e "2 \c"
if [ $number -eq 2 ]
then
    echo
    exit 2
fi

i=3
while [ $i -le $number ]
do
prime $i
    ret=$?
    if [ $ret -eq 1 ]
    then
        echo -e "$i \c"
    fi
    i='expr $i + 2'
done
done
exit 0
$ prime

Enter a number: 200
The prime number list within 200
2 3 5 7 11 13 17 19 23 29 31 37 41 43 47 53 59 61 67 71 73
79 83 89 97 101 103 107 109 113 127 131 137 139 149 151 157 163
167 173 179 181 191 193 197 199
Enter a number:
$
```

return 语句的返回值仅限于 0~255 的整数。为了返回较大的数值，可以使用 echo 等命令语句，然后采用命令替换的形式获取函数的返回值。例如，为了求取两个变量中的最大值，可以定义下列函数。

```

$ cat max.sh
#!/bin/bash
if [ $# -ne 2 -a $# -lt 2 ]
then
    echo "Usage: 'basename $0' number#1 number#2"
    exit 1
fi
max() {
    if [ $1 -gt $2 ]
    then
        echo $1
    else
        echo $2
    fi
}
echo "The greatest number of $1 and $2 = 'max $1 $2'"
$ max.sh 7856 9980
The greatest number of 7856 and 9980 = 9980
$
```

不管在 Shell 脚本中，还是在交互式 Shell 环境中，在调用函数之前，首先必须对函数进行定义。定义之后，函数将位于用户的 Shell 运行环境中。使用不带参数的 set 命令，可以看到函数的定义。使用 unset 命令或终止当前运行的 Shell，即可取消当前的函数定义，因而也就不能再执行了，代码如下所示。

```

$ exchange() {
> temp=${color1}
> color1=${color2}
> color2=${temp}
> return
> }
$ set
.....
```



```
exchange ()  
{  
    temp=${color1};  
    color1=${color2};  
    color2=$temp;  
    return  
}  
$
```

函数也可以在 Shell 脚本中定义，在脚本中执行，最后随脚本的终止而取消。如果在 Shell 脚本中定义函数，函数定义可以出现在允许使用命令语句的任何位置。但是，函数的定义必须位于调用函数的语句之前。对于需要经常使用的函数代码，最好在用户自己的.profile 文件中定义，这将保证一旦注册，只要需要，随时都可以直接调用函数。

下面的例子中定义了一个函数，专门负责处理常用的提示与用户响应信息，以便能够在必要时供 Shell 脚本随时调用。

```
$ cat report.sh  
#!/bin/bash  
prompt()  
{  
    while  
        echo "Please answer yes or no: \c"  
        read yes_no  
    do  
        case "${yes_no}" in  
            Y* | Y*) yes_no="y"  
                break  
                ;;  
            N* | N*) yes_no="n"  
                break  
                ;;  
            *) echo  
                continue  
                ;;  
        esac  
    done  
}  
echo "Would you like headings?"  
prompt  
if test "${yes_no}" = "y"  
then  
    echo "Headings selected."  
    echo "Start to process ... done."  
fi  
echo "Would you like a summary only?"  
prompt  
if test "${yes_no}" = "y"  
then  
    echo "Summary only selected."  
    echo "Start to process ... done."  
fi  
exit 0  
$ report.sh  
Would you like headings?  
Please answer yes or no: y  
Headings selected.  
Start to process ... done.  
Would you like a summary only?  
Please answer yes or no: n
```

无论何时调用函数，都会重新设置当前 Shell 进程的位置参数，使调用函数时提供的参数成为新的位置参数。也就是说，Shell 将会使用调用函数时提供的参数重新为位置参数赋值。如果调

用函数时未提供参数，位置参数（中的值）将被清除。

函数按位置引用传递给自己的参数\$1、\$2、……。因此，也可以使用 shift 命令处理传递给函数的参数。有关位置参数与 shift 命令方面的内容，可以参考前面介绍的例子，这里不再细叙。

与其他编程语言不同，Shell 函数通常只接受数值参数。如果把变量名作为参数传递给函数，函数将会把变量名作为字符串常量处理。也就是说，函数将按字符串文字常量解释变量参数，示例如下。

```
$ cat varref
#!/bin/bash
echo_var () {
    echo "$1"
}
Message=Hello
Hello="Good Morning"
echo_var Message
echo_var Hello
echo_var "${Message}"
echo_var "${Hello}"
exit 0
$ varref
message
Hello
Hello
Good Morning
$
```

事实上，函数也是一个代码块，这意味着也可以重定向函数的标准输入、标准输出和标准错误输出。例如，在下面的函数定义例子中，“< \$file”表示把函数的标准输入重定向到\$file 变量指定的文件中。当调用 function\_name 函数时，函数中的 read 语句不再读取来自键盘的输入数据，而是直接读取自\$file 变量定义的文件。

```
#!/bin/sh
function_name () {
    ...
    read lines
    ...
} < $file      # 重定向函数的标准输入
```

在下面的例子中，“function\_name2 < \$file”语句表示仅在调用普通的 function\_name2 函数时，再把标准输入重定向到\$file 变量定义的文件中。因此，函数中的 read 语句读取的数据仍然取自\$file 变量定义的文件，而非键盘输入。

```
#!/bin/sh
function_name2 () {
    ...
    read lines
    ...
}
function_name2 < $file
```

下面再以一个冒泡排序的例子，说明怎样调用函数，作为本节的结束。另外，这个例子涉及到一些数组方面的概念，读者可以参考 8.13 节的介绍。

冒泡排序算法的基本思想是：对于 n 个需要排序的元素，执行 n-1 次循环。每次循环均从第 1 个元素开始，依次比较相邻的两个元素；如果第 i 个元素大于第 i+1 个元素，则交换两个元素的位置，直至比较到最后第 n-j+1 个元素（其中 j 为循环次数）结束。第一次循环结束后，最大的



元素将会被交换到最后一个数组元素位置。第 2 次循环结束后，仅次于最大元素的元素将会被交换到最后第 2 个数组元素位置。依此类推，直至循环结束。在每次循环中，较大的元素依次后移，较小的元素依次前移。当最后一次循环结束时，整个数组元素将完成从小到大的排序。

```
$ cat bubble.sh
#!/bin/bash
exchange()          # 交换两个数组元素
{
    temp=${colors[$1]}
    colors[$1]=${colors[$2]}
    colors[$2]=$temp
    return
}
colors=(red green blue yellow black white brown rose grey)
echo "0: ${colors[*]}"           # 输出排序前的全部数组元素
elements=${#colors[@]}           # 求出数组元素的个数
let "count = $elements - 1"
pass=1
while [ "$count" -gt 0 ]
do
    index=0                   # 每次循环均从第一个数组元素开始
    while [ "$index" -lt "$count" ]
    do
        if [[ ${colors[$index]} > ${colors['expr $index + 1']} ]]
        then
            exchange $index 'expr $index + 1'
        fi
        let "index += 1"
    done
    let "count -= 1"           # 一次冒泡排序循环结束
    echo "$pass: ${colors[@]}"   # 显示每次循环后的中间排序结果
    let "pass += 1"
done
exit 0
$ bubble.sh
0: red green blue yellow black white brown rose grey
1: green blue red black white brown rose grey yellow
2: blue green black red brown rose grey white yellow
3: blue black green brown red grey rose white yellow
4: black blue brown green grey red rose white yellow
5: black blue brown green grey red rose white yellow
6: black blue brown green grey red rose white yellow
7: black blue brown green grey red rose white yellow
8: black blue brown green grey red rose white yellow
$
```

## 8.12 逻辑与和逻辑或并列结构

命令的逻辑与（`&&`）和逻辑或（`||`）并列结构提供了一种依次执行一系列命令的手段，能够有效地替代复杂的、嵌套的 `if-then` 语句结构。

### 8.12.1 逻辑与命令并列结构

逻辑与命令并列结构的语法格式如下。

```
command-1 && command-2 && .... && command-n
```

命令的逻辑与并列结构表示从第一个命令开始，依次执行每一个命令。如果当前命令的返回值（出口状态）为 0，则继续执行下一个命令。如果中间某个命令的返回值大于 0，则整个并列命令链的执

行结束。也就是说，第一个返回非 0 值的命令即为逻辑与并列结构中最后执行的一个命令。

下面是取自/etc/init.d/waitnfs.sh 启动脚本中的一行代码。这一代码首先检查/etc/default/rcS 文件是否存在，如果文件存在，则在当前的 Shell 中执行该文件，以便设置系统引导过程中用到的环境变量。

```
[ -f /etc/default/rcS ] && . /etc/default/rcS
```

如果把逻辑与命令并列结构改为 if-then 结构语句，上述命令相当于：

```
if [ -f /etc/default/rcS ]
then
    . /etc/default/rcS
fi
```

## 8.12.2 逻辑或命令并列结构

逻辑或命令并列结构的语法格式如下。

```
command-1 || command-2 || ... || command-n
```

与命令的逻辑与并列结构相反，逻辑或命令并列结构意味着从第一个命令开始，依次执行每一个命令。如果当前命令的返回值（出口状态）大于 0，则继续执行下一个命令。如果中间某个命令的返回值为 0，则整个并列命令链的执行结束。也就是说，第一个返回 0 的命令即为逻辑或并列结构中最后执行的一个命令。

下面是取自/etc/init.d/networking 启动脚本中的一行代码。这一代码首先检查/sbin/ifup 文件是否存在且是否可以执行，如果文件存在且可以执行，则从下一行开始执行启动脚本中的后续代码。如果文件不存在，或者即使存在但不能执行，则结束启动脚本的继续执行，这是因为 Ubuntu Linux 系统需要使用 ifup 命令完成网络接口的配置。

```
[ -x /sbin/ifup ] || exit 0
```

如果把逻辑或命令并列结构改为 if-then 结构语句，上述命令相当于：

```
if [ ! -x /sbin/ifup ]
then
    exit 0
fi
```

注意，在逻辑与和逻辑或命令并列结构中，整个结构语句的出口状态是其中最后执行的一条命令的出口状态。

灵活地使用逻辑与和逻辑或命令并列结构及其组合，可以简化 Shell 编程，但有时也会增加理解的困难，因而需要做额外的调试。

## 8.13 Shell 数组

Bash 仅支持一维数组，但对数组的大小没有限制。数组采用整数索引，第一个数组元素从 0 开始。要引入一个数组，可以采用下列语法格式。

```
[declare -a] array-name=( element1 element2 ... elementN )
array-name[index]=value
```

采用第一种语法格式定义一个数组，能够在声明数组的同时，对数组进行初始化。如果在定义数组时再增加一个“declare -a”前缀，则由 declare 语句预先声明的数组能够加速数组的处理，提高 Shell 脚本的性能。

例如，为了定义并初始化一个表示周日的数组，可以使用下列语句。



```
$ weekday=( Sunday Monday Tuesday Wednesday Thursday Friday Saturday )
```

```
$
```

采用第二种语法格式定义并初始化任何一个数组元素，也可以构建一个数组。因此，如果采用下列方式，仅对部分数组元素进行赋值，能够构建一个部分初始化的数组。

```
$ array=( [0]="first" [1]="second" [3]="fourth" )
```

```
$
```

按照上述方式定义并初始化数组之后，其结果如下。

```
$ echo ${array[0]}
```

```
first
```

```
$ echo ${array[1]}
```

```
second
```

```
$ echo ${array[2]}
```

```
$ echo ${array[3]}
```

```
fourth
```

```
$
```

在具体应用过程中，可以像操作普通变量一样操作数组变量，只是引用形式稍作变化而已。许多标准的字符串操作也适应于数组，且同普通变量一样，每个数组元素也可以采用“variable[n]=value”形式赋值，因而也可以采用逐个数组元素赋值的形式，定义并初始化一个数组。同样，为了引用数组元素的内容，可以采用\${variable[n]}等形式实现变量替换。例如，可以采用下列等价的方式定义并初始化一个 weekday 数组。

```
weekday[0]=Sunday
```

```
weekday[1]=Monday
```

```
....
```

```
weekday[6]=Saturday
```

在声明并定义了一个 weekday 数组之后，即可使用带下标的变量引用数组的元素，如下所示。

```
$ echo ${weekday[0]}
```

```
Sunday
```

```
$ echo ${weekday[1]}
```

```
Monday
```

```
$
```

与\$@或\$\*变量值表示所有的位置参数类似，\${array\_name[@]}或\${array\_name[\*]}变量值表示数组的所有元素，示例如下。

```
$ echo ${weekday[@]}
```

```
Sunday Monday Tuesday Wednesday Thursday Friday Saturday
```

```
$ echo ${weekday[*]}
```

```
Sunday Monday Tuesday Wednesday Thursday Friday Saturday
```

```
$
```

同样，与\$#变量值表示位置参数的数量类似，\${#array\_name[@]}或\${#array\_name[\*]}变量值表示数组 array\_name 的长度，即所有数组元素的数量。下面的例子说明了数组 weekday 具有 7 个数组元素。

```
$ echo ${#weekday[@]}
```

```
7
```

```
$ echo ${#weekday[*]}
```

```
7
```

```
$
```

为了求取数组元素的长度（字符个数），可采用\${#array\_name[index]}的形式，指定具体的数组元素，示例如下。

```
$ echo ${#weekday[0]}
```

```
6
```

```
$ echo ${#weekday[6]}
```

```
8
```

```
$
```

但是，\${#array\_name}变量值仅表示第一个数组元素 array\_name[0]的长度，示例如下。

```
$ echo ${#weekday}
6
$
```

对于数组而言，某些 Shell 命令具有特殊的意义。例如，使用 `unset` 命令能够删除任何数组元素，甚至整个数组。

```
$ unset weekday[1]          # 相当于执行 "weekday[1]="" 语句
$ echo ${weekday[1]}

$ unset weekday
$ echo ${weekday[0]}
```

```
$
```

下面的例子说明了怎样利用数组和一定的算法，计算任何一天是星期几。

```
$ cat whatday
#!/bin/bash
typeset -i year month day mm=1 sum week yy
typeset -i leap=0
months=( 0 31 28 31 30 31 31 31 30 31 30 31 )
weekday=( Sunday Monday Tuesday Wednesday Thursday Friday Saturday )

month=${1:-`date '+%m'`}
day=${2:-`date '+%d'`}
year=${3:-`date '+%Y'`}

let yy=year-1
let sum=yy*365+yy/400+yy/4-yy/100

if [ `expr $year % 4` -ne 0 ] # 如果年份不能被 4 除尽，则不是闰年
then
    leap=0
elif [ `expr $year % 100` -ne 0 -o `expr $year % 400` -eq 0 ]
then
    leap=1
fi
# 如果年份能够被 4 除尽，但不能被 100 除尽，则为闰年
# 如果年份能够同时被 4、100 和 400 除尽，则为闰年
# 如果年份既能被 4 除尽，又能被 100 除尽，但不能被
# 400 除尽，则不为闰年

while [ $mm -lt $month ]
do
    sum='expr $sum + ${months[$mm]}'
    mm='expr $mm + 1'
done
sum='expr $sum + $day + $leap'
week='expr $sum % 7'
echo "$month $day, $year is ${weekday[$week]}"
exit 0
$ whatday
11 4, 2008 is Tuesday
$ whatday 12 22 2008
12 22, 2008 is Monday
$
```

灵活地组合使用数组初始化语句“`array=( element1 element2 ..... elementN )`”与命令替换，能够把一个文本文件的内容加载到数组中。因此，定义并初始化数组的第二种方法是命令替换法。采用命令替换法初始化数组，可以构建一个具有多个元素的数组，数组的大小依据命令输出的多少而定。

在采用命令替换形式初始化数组时，Shell 使用空格作为数组元素的分隔符，把每个字符串分配到一个数组元素中，最终构建成一个字符串数组。在遇到制表符和换行符时，Shell 首先把它们转换成空格，并用作元素分隔符。

例如，假定 `dict` 文件包含下列 4 行数据，采用命令替换形式的数组定义语句“`array=(`cat dict`)`”可以声明并初始化一个数组。



```
$ cat dict
aa
bb
cc
dd
$ array=( `cat dict` )
$ echo ${array[@]}
aa bb cc dd
$ echo ${#array[*]}
4
$
```

下列 Shell 脚本能够读取以命令行参数形式提供的文本文件，并以空格、制表符和换行符作为分隔符，把每个连续的字符串看作一个单词，使之构成一个数组元素。

```
$ cat array.sh
#!/bin/bash
array=( $(cat "$1") )           # 把命令行参数文件的内容存储到数组中
total=${#array[@]}
echo $total
element=0
while [ $element -lt $total ]
do
    echo ${array[$element]}
    element='expr $element + 1'
done
exit 0
$
```

如果使用 Shell 脚本本身作为文件参数，运行上述 Shell 脚本的输出结果如下。

```
$ array.sh array.sh
25
#!/usr/bin/bash
array=
$(cat
"$1")
)
total=${#array[@]}
echo
$total
element=0
while
[
$element
-lt
$total
]
do
echo
${array[$element]}
element='expr
$element
+
1'
done
exit
0
$
```

显然，如果用于处理普通数据文件，上述输出结果是可以接受的。但如果用于处理 Shell 脚本文件，上述的输出内容并非我们想要的结果。这里的问题是，如果把空格或制表符作为分隔符将会使脚本的语句支离破碎而变形。如果稍加改造，在构成数组元素之前，先把空格和制表符替换成文本文件中不会出现的其他特殊字符，然后在进一步处理之前，再还原成原来的字符，示例如下。

```
$ cat array2.sh
#!/bin/bash
```

```

array=( $(cat "$1" | tr ' ' '\001' | tr '\t' '\002') )
total=${#array[@]}
echo $total
element=0
while [ $element -lt $total ]
do
    echo ${array[$element]} | tr '\001' '' | tr '\002' '\t'
    element='expr $element + 1'
done
exit 0
$ array2.sh array2.sh
11
#!/usr/bin/bash
array=( $(cat "$1" | tr ' ' '\001' | tr '\t' '\002') )
total=${#array[@]}
echo $total
element=0
while [ $element -lt $total ]
do
    echo ${array[$element]} | tr '\001' '' | tr '\002' '\t'
    element='expr $element + 1'
done
exit 0
$ 
```

定义并初始化数组的第 3 种方法是采用 Shell 元字符与文件名生成方法。采用元字符与文件名生成形式定义数组，也可以构建并初始化一个具有多个文件名元素的数组，数组的大小依据文件的数量而定。同使用命令替换形式初始化数组类似，Shell 采用空格作为数组元素的分隔符，把每个文件名分配到一个数组元素中，最终构建成一个文件名数组。

例如，为了对当前目录下的 C 程序进行操作，可以使用下列语句生成一个文件名数组。

```

$ filelist=(*.c)
$ echo ${filelist[@]}
atmcom.c ammon.c atmstat.c handler.c listener.c
$ echo ${#filelist[@]}
5
$ 
```

此外，还可以把普通变量看作数组的一个特例（一个特殊的数组），即只有一个元素的数组。因此，即使并未明显地把变量声明为数组，Bash 也允许以数组或数组元素的处理方式处理普通变量，示例如下。

```

$ string=abcABC123ABCabc
$ echo ${string[0]}
abcABC123ABCabc
$ echo ${string[*]}
abcABC123ABCabc
$ echo ${string[0]}
abcABC123ABCabc
$ echo ${string[1]}      # 一个普通变量只能被看作第一个数组元素（下标为 0）
$ echo ${#string[@]}     # 数组中只有一个元素，即字符串本身
1
$ 
```

数组变量具有自己的语法规则，可以利用赋值语句实现整个数组的复制，也可以在数组的基础上增加新的元素。例如，下列两个语句均可实现整个数组的复制（其中的第一个语句实为包含变量替换的数组赋值语句）。

```

$ array2=( "${array1[@]}" )
$ array2="$array1[@]" 
```

还可以使用下列语句，把一个数组和一个新的元素复制到另一个数组中。

```

$ array=( "${array[@]}" "new element" ) 
```

Bash 仅支持一维数组，但稍加变通即可仿真多维数组。限于篇幅，这里不再赘述。



## 8.14 信号的捕捉与处理

信号是一种能够影响进程运行状态的外部事件，是由用户终端（如按下 Ctrl-C 键）、操作系统内核或其他进程（如 kill 命令）发送给当前或指定进程的消息，用于通知进程某种事件已经发生。进程可根据预定的方案，采取一定的处理措施或终止进程的执行。如果事先没有捕捉信号，默认的处理动作是停止进程的执行。

在 Shell 脚本中，trap 命令用于指定需要捕捉的信号以及应采取的相应处理动作。当接收到某个指定的信号时，Shell 将会立即触发相应的处理过程。trap 命令的语法格式简写如下。

```
trap ["command-list"] [signal ...]
```

或

```
trap ['command-list'] [signal ...]
```

其中，command-list 是无论何时收到指定的信号时都会立即执行的命令或一组命令。一旦命令执行结束，程序的控制逻辑将会恢复到因收到信号而中断的位置开始继续运行，除非命令中包含 exit 语句。如果在收到指定的信号时应当结束脚本的运行，可以使用 exit 语句作为指定命令的一部分（如果存在，exit 语句通常应为命令组中的最后一个命令）。通常情况下，指定的命令并非单个命令，而是一组命令，故前后应加单或双引号，命令中间以分号隔开。

signal 表示准备捕捉的信号。信号可以是一个标准的信号名，也可以是一个数字。针对 Shell 编程而言，比较重要的、能够捕捉的常用信号只有 0、1、2、3、15 和 20 等。如果捕捉的信号是 0 (EXIT)，则仅当脚本运行结束时才能执行相应的处理动作。这种例行处理方式经常用于清除 Shell 脚本运行过程中创建的临时文件。另外还有 3 个特殊的信号 DEBUG、ERR 和 RETURN，主要用于调试和控制 Shell 脚本的执行（详见下一节的讨论）。表 8-1 给出了这些信号的简单说明（至于系统定义的其他信号，详见第 9 章）。

表 8-1 Shell 脚本能够捕捉的信号

信 号	信 号 名	简 单 说 明
0	EXIT	进程结束信号。在 Shell 脚本的情况下，只要执行 exit 语句终止 Shell 脚本的执行时，都可产生此信号。在 Shell 脚本运行正常结束时，即使没有明显地执行 exit 语句，也可产生此信号
1	SIGHUP 或 HUP	挂断。终端通信连接断开或控制进程终止时产生的信号。通常，一旦捕捉到此信号，Shell 脚本将会立即停止运行，除非使用 nohup 命令
2	SIGINT 或 INT	中断。按下中断键（也即 Ctrl-C 键）时产生的信号
3	SIGQUIT 或 QUIT	退出。按下 Quit 键（也即按 Ctrl-\ 或 Ctrl-Shift-\ 键）时产生的信号
15	SIGTERM 或 TERM	终止信号。这是 kill 命令产生的默认终止信号
20	SIGTSTP 或 TSTP	键盘停止信号。在交互方式下，通过按 Ctrl-Z 键停止当前进程时产生的信号。它主要用于作业控制
	DEBUG	在执行 Shell 脚本中的每个命令之前，包括 for、case 和 select 命令语句，以及 Shell 函数中的第一个命令，都要运行 trap 语句中指定的命令
	ERR	Shell 脚本中的任何命令，一旦运行结束时返回非 0 值的出口状态，都要运行 trap 语句中指定的命令，但 while、until 或 if 等结构语句中的测试语句除外
	RETURN	每当调用一个 Shell 函数，或者利用 “.” 或 source 命令执行一个 Shell 脚本结束之后，都要运行 trap 语句中指定的命令

在 Linux 系统中，每个信号都有一个名字和数字。在编写 Shell 脚本时，建议使用名字指定捕捉的信号，这将使 Shell 脚本更具有可读性，并具有可移植性。

注意，在 Shell 脚本中，有 5 个信号不应捕捉：其中信号 9 和 19 是不能捕捉的，而信号 17 和 30 则不应捕捉，这些信号的定义详见第 9 章。在 Shell 脚本中捕捉信号 10 和 12 会引起管道通信方面的问题。

利用 trap 命令，程序员可以采取下列 3 种处理措施之一：

- 捕捉指定的信号，然后执行必要的命令或处理动作；
- 忽略收到的信号；
- 清除先前的信号捕捉设置。

如果 trap 语句中指定的信号为 0 或 EXIT，则当执行 exit 命令语句，或者 Shell 脚本的执行正常结束时，立即执行 trap 语句中指定的处理动作。例如，下面的例子说明了怎样在 Shell 脚本运行结束时捕捉 EXIT 信号，从而显示脚本执行过程中设定的变量内容。

```
$ cat test.sh
#!/bin/bash
trap 'echo -e "Variable setting ---\na = $a\nb = $b"' EXIT
echo "This line will print before the \"trap\" statement."
echo "even though the \"trap\" statement occurs first."
echo
a=""
b=36
exit 0
$ test.sh
This line will print before the "trap" statement.
even though the "trap" statement occurs first.

Variable setting ---
a =
b = 36
$
```

注意，上述 Shell 脚本的“exit 0”语句的存在与否并无太大的区别，因为在 Shell 脚本执行到最后一条命令语句时都会立即结束。如果最后一条命令语句的出口状态为 0，其效果等同于执行“exit 0”语句。Shell 脚本终止运行前执行的最后一条命令语句的出口状态即为整个 Shell 脚本的出口状态。

如果 trap 语句中指定的捕捉信号为 1 或 HUP，且在 Shell 脚本在运行过程中终端连接断开时，Shell 将会立即执行定义的处理动作。如果 trap 语句中指定的信号为 2 或 INT，当用户在 Shell 脚本执行过程中按下 Ctrl-C 键时，Shell 将会立即执行定义的处理动作。如果 trap 语句中指定的信号为 3 或 QUIT，当用户在 Shell 脚本执行过程中按下 Ctrl-\ 键时，Shell 将会立即执行定义的处理动作。

在下列例子中，如果在 Shell 脚本的运行过程中收到信号 1、2 或 3，Shell 将会执行 trap 命令语句中指定的命令序列，显示“Interrupted routine”，删除临时文件 tmp\$\$，终止脚本的执行，最终返回一个数值 1 作为 Shell 脚本的出口状态。

```
trap 'echo Interrupted routine; rm -f tmp$$; exit 1' 1 2 3
```

或

```
trap 'echo Interrupted routine; rm -f tmp$$; exit 1' HUP INT QUIT
```

有时，我们不希望任何人中途打断 Shell 脚本的运行。因此，为了防止任何人使用 Ctrl-C 键打断 Shell 脚本的正常运行，可以使用下列形式的 trap 语句，屏蔽中断信号。

```
trap '' 2
```

或

```
trap 'echo "Ctrl-C disabled."' 2
```



如下所示，如果 trap 语句的命令部分中没有指定任何处理动作，或者明显地指定一个 null 值，则表示忽略指定的信号。

```
trap '' signal
```

或

```
trap ':' signal
```

上述两个 trap 命令存在细微的差别：第一个 trap 语句表示完全忽略指定的信号，第二个 trap 语句则意味着在收到指定的信号时不做任何处理。这两者在同一 Shell 脚本中的差别并不明显，但在父进程与子进程的处理方面则大不相同。

在第一种情况下，如果父进程忽略收到的指定信号，则子进程也将忽略相应的信号。例如，下面的例子表示父进程将会忽略捕捉到的中断信号（Ctrl-C），同时告诉子进程也忽略这一信号。

```
trap "" 2
```

在第二种情况下，如果父进程捕捉到指定的信号，除了自己只是执行“：“语句而不做其他任何处理动作之外，还将告诉子进程清除其相应信号的设置，包括继承自父进程的信号设置（在收到信号 2 时执行“：“语句），同时把子进程未做“忽略”处理的所有信号恢复为默认的处理动作。在下列例子中，当捕捉到中断信号（按 Ctrl-C 键）时，父进程仅执行一个简单的“：“语句，但不会改变子进程对相应信号的默认处理动作。

```
trap ":" 2
```

在使用 trap 语句时，如果仅指定了准备捕捉的信号而未指定相应的处理动作，Shell 将会删除先前为相应信号定义的处理动作。这意味着从此时开始，如果捕捉到这些信号，Shell 将会按默认的处理动作执行。在下面的 trap 语句例子中，Shell 将会删除为信号 1、2 和 3 定义的处理动作。也就是说，如果收到信号 1、2 或 3，Shell 将会执行默认的处理动作——终止 Shell 脚本的执行。

```
trap 1 2 3
```

通常，trap 语句应是 Shell 脚本中的第一个语句，但程序员也可根据具体的编程要求做适当的调整。不管 trap 语句位于何处，在捕捉到指定的信号之前，Shell 只是把 trap 语句读入内存，实际上并不执行。只有在收到指定的信号时，才会开始执行 trap 语句中定义的命令。

例如，在一个数据库更新的 Shell 脚本中，数据的输入过程并不重要，即使中断脚本的执行也无关紧要。但在数据库更新时，通常需要屏蔽一切外来的干扰，防止用户的中断信号等打断脚本的执行，以免破坏数据的完整性。因此，程序员可在数据库更新的关键代码处增加 trap 语句，使之屏蔽任何可能的外部信号干扰。在关键代码执行结束之后，再使用不带命令参数的 trap 语句清除之前的信号捕捉设置。示例代码如下。

```
$ cat updatedb
#!/bin/bash
while true
do
    echo "Please enter name, phone number, and email address"
    echo "separated by blank character (or enter quit to end):"
    read name phone email
    if test "${name}" = "quit"
    then
        break
    fi

    trap "" 2 3
    # Critical code segment to update database
    update.sql $name $phone $email
    trap 2 3

done
$
```

Shell 脚本中的 trap 语句通常会读取两次。在开始执行 Shell 脚本时，Shell 将会读取并存储整个 trap 语句，其间如有变量替换或命令替换，也将随之进行替换处理。当收到指定的信号时，Shell 会再次读取并执行 trap 语句。同样，如果 trap 语句中仍然存在变量替换或命令替换，还会再次进行替换处理。

假定 trap 语句中存在变量替换表达式，如果变量是在脚本执行期间赋值的，则此情况下可能会存在一个问题：当第一次读取 trap 语句，并执行变量替换时，Shell 将会采用初始的 null 数值。

为了防止此类问题的发生，可在 trap 语句中使用单引号界定其中的命令，以避免在第一次读取 trap 语句时执行变量替换，使之仅当捕捉到指定的信号，需要执行相应的命令时再进行变量替换。

因此，下列两种命令引用形式是不同的：前者执行两次变量替换或命令替换，而后者只执行一次替换。

```
trap ["command-list"] [signal ...]
trap ['command-list'] [signal ...]
```

## 8.15 其他 Shell 课题

### 8.15.1 子 Shell

当用户在键盘上输入单个命令时，Shell 将以交互方式解释用户提交的命令，然后通过创建一个新的进程，执行用户提交的命令。而在运行 Shell 脚本时，当前 Shell 将会创建一个新的 Shell 进程，以批处理的方式处理用户提交的 Shell 脚本，解释执行脚本文件中的一系列命令。事实上，每个运行的 Shell 脚本都是当前 Shell 的子进程。

Shell 脚本本身也可以创建子进程，即子 Shell。这些子 Shell 采用并行处理的方式，能够同时执行多个处理任务。通常，脚本中使用的每个外部命令都会使 Shell 创建一个相应的子进程，但内部命令则不然。由于这个原因，内部命令的执行速度明显快于等价的外部命令。

如下所示，位于圆括号中的命令将以子 Shell 的形式运行。

```
( command1; command2; command3; ... )
```

子 Shell 中的变量仅在子 Shell 代码块中有效，在调用子 Shell 的 Shell 中是不可见的，也就是说，子 Shell 中的变量属于本地变量。参见下面的例子。

```
$ cat subshell.sh
#!/bin/bash
outer_var=Outer
(
inner_var=Inner
echo "From subShell, \"inner_var\" = $inner_var"
echo "From subShell, \"outer\" = $outer_var"
)

if [ -z "$inner_var" ]
then
    echo "inner_var undefined in main code body of Shell"
else
    echo "inner_var defined in main code body of Shell"
fi
echo "From main code body of Shell, \"inner_var\" = $inner_var"
# $inner_var will show as uninitialized because
# variables defined in a subShell are "local variables".
exit 0
```



```
$ subshell.sh
From subShell, "inner_var" = Inner
From subShell, "outer" = Outer
inner_var undefined in main code body of Shell
From main code body of Shell, "inner_var" =
$
```

在子 Shell 中使用 cd 命令所做的目录变动也不影响父 Shell。也就是说，在子 Shell 改变目录时，仅在子 Shell 中有效，这一点要特别注意，参见下面的例子。

```
$ cat subshell2
#!/bin/sh
echo "Current directory = `pwd`"
(cd /etc; cat timezone)
echo "Current directory = `pwd`"
exit 0
$ subshell2
Current directory = /home/gqxing/script
Asia/Shanghai
Current directory = /home/gqxing/script
$
```

## 8.15.2 Shell 脚本的调试

Shell 不提供 Shell 脚本的调试程序，甚至也不提供任何有关的调试命令或结构语句。脚本中的语法错误或明显的拼写错误产生的错误信息在调试脚本时经常无助于调试一个有误的 Shell 脚本。

下面是一个其中包含语法错误的 Shell 脚本，执行时将会出现下列错误信息。

```
$ cat test2.sh
#!/bin/bash
LANG=C
a=40
if [$a -gt 30 ]
then
    echo $a
fi
exit 0
$ test2.sh
test2.sh: line 4: [40: command not found
$
```

前面曾经讲过，方括号与表达式之间必须至少保留一个空格。而在上述脚本中，“[”和“\$a”连在一起，故 Shell 把 “[40” 看作一个单词，即看作一个 “[40” 命令，而这样的命令显然是不存在的。

下面是一个漏掉关键字的例子。执行 Shell 脚本后产生的错误信息如下。

```
$ cat test3.sh
#!/bin/bash
LANG=C
for a in 1 2 3
do
    echo "$a"
# done           # 不知何故注解掉了关键字 done
exit 0
$ test3.sh
test3.sh: 8: syntax error: end of file unexpected (expecting "done")
$ cat test4.sh
#!/bin/bash
LANG=C
for a in 1 2 3           # 遗漏了关键字 do
    echo "$a"
done
exit 0
$ test4.sh
test4.sh: line 4: syntax error near unexpected token `echo'
```

```
test4.sh: line 4: `    echo "$a"'
$
```

注意，在输出产生语法错误的行号时，Shell 是把注释行也计算在内的。另外，出错信息指出的行号也并不一定就是出错语句所在行的位置，但这一位置却是 Shell 解释程序最终发现脚本有误的地方。在上述第一个例子中，Shell 读到最后仍然没有发现应有的关键字 done，故给出的是一个根本就不存在的行号。

当遇到下述情况之一时，Shell 脚本将会终止运行：

- 控制结构语句（包括循环结构语句和条件转移结构语句）中出现语法错误；
- 接收到某种信号，包括外部信号（如按 Ctrl-C 键、终端关闭或线路断开等）和内部信号；
- Shell 内置语句（read 和 test 语句除外）执行失败；
- 特殊替换中出现语法错误（如引号不匹配或不配对）；
- 赋值语句出现故障（对使用 readonly 命令定义的只读变量赋值）。

但下列情况不会引起 Shell 脚本停止运行：

- 试图执行一个不存在的命令（如命令名字拼写错误，或者命令文件没有执行许可等）；
- I/O 重定向故障（如试图使用“>”重定向符号写入一个已存在的文件等），此时，除了 I/O 受到影响外，并不影响脚本的执行；
- 命令异常中止（即使一个程序产生了内存映像文件 core，Shell 仍会继续执行下一个命令）；
- 命令终止运行时返回非 0 值的出口状态（除非使用 trap 语句跟踪此种情况）。

根据上述说明及实际应用，Shell 脚本的错误情况可以归结为如下类型。

(1) 运行时出现“syntax error”信息，说明脚本中存在诸如语句不完整，语法格式有误，遗漏关键字，引号不匹配或不配对等语法错误。

(2) 可以运行，但最终的处理结果并非预期，这主要是由于逻辑错误造成的。

(3) 可以运行，处理结果也无问题，但存在严重的边界效应。

为了调试 Shell 脚本，可以采用下列方法及工具。

(1) 在脚本的关键位置加入 echo 语句，跟踪和显示关键变量的值，借以判断脚本是否存在逻辑问题。

(2) 在关键位置使用 tee 命令检查进程或数据流。

(3) 设置命令执行过程的跟踪标志（增加“-n”、“-v”或“-x”选项）。

- “sh -n scriptname”命令用于检查 Shell 脚本的语法错误，实际上并不运行脚本。这一命令形式等价于在脚本中插入“set -n”或“set -o noexec”语句。注意，这种检查并不能找出所有的语法错误。
- “sh -v scriptname”命令的作用是在执行之前显示每一个命令，然后显示命令的执行结果。这一命令形式等价于在脚本中插入“set -v”或“set -o verbose”语句。
- “-n”和“-v”选项可以一起工作。例如，“sh -nv scriptname”命令意味着给出详细的语法检查信息。
- “sh -x scriptname”命令的作用是以简化的方式显示每一条命令语句的执行结果，等价于在 Shell 脚本中插入“set -x”或“set -o xtrace”语句。
- 在脚本中插入并执行“set -u or set -o nounset”语句，使 Shell 在遇到试图使用未声明的变量时给出错误信息。



(4) 使用 trap 命令捕捉 Shell 脚本终止执行时的位置。在 Shell 脚本中，一旦执行到 exit 命令，将会生成一个 EXIT 信号 (0)，并终止 Shell 脚本的执行。因此，捕捉 EXIT 信号，输出运行结束时的变量值通常是一种很有用的方法。

(5) 在 trap 命令中，DEBUG 信号使 Shell 能够在执行脚本中的每个命令之前立即执行指定的处理动作。这种“trap 'commands' DEBUG”设置能够在调试 Shell 脚本时跟踪变量的赋值情况，从中找出 Shell 脚本中的逻辑错误。

(6) 在 trap 命令中，ERR 信号使 Shell 能够在执行脚本中的每个命令时，跟踪运行过程中出现异常的命令。这种“trap 'commands' ERR”设置有助于找出 Shell 脚本中出错的命令语句，从而找出 Shell 脚本出错的原因。

(7) 在 trap 命令中，RETURN 信号使 Shell 能够在调用脚本中的 Shell 函数，或者利用“.”或 source 命令执行其他 Shell 脚本之后，立即执行指定的处理动作。这种“trap 'commands' RETURN”设置能够跟踪 Shell 函数或其他脚本的执行情况。

信号捕捉机制对调试 Shell 脚本是很有用的。下面的例子就是利用“trap 'commands' DEBUG”设置，在执行每个赋值语句之前输出 var 变量的变化情况。

```
$ cat testing
#!/bin/bash
trap 'echo "TRACE VARIABLE-> \$var = \"$var\""' DEBUG
var=29
let "var = 3"
exit 0
$ testing
TRACE VARIABLE-> $var =
TRACE VARIABLE-> $var =
TRACE VARIABLE-> $var =
$
```

如果 trap 语句中跟踪的信号为 ERR，则 Shell 脚本中的任何命令无论何时返回非 0 值的出口状态，Shell 都会执行 trap 语句定义的处理动作。下列例子中的 trap 语句表示，只要某个命令返回非 0 值的出口状态，立即显示 Filename 变量的值。

```
$ cat testing2
#!/bin/bash
LANG=C
trap 'echo "$LINENO: Filename = \"$Filename\""' ERR
ls -l somefile
exit 0
$ testing2
ls: cannot access somefile: No such file or directory
4: VAR Filename = "
```

下面的例子说明，不管脚本是否正常终止，只要执行了 exit 语句或在 Shell 脚本执行结束时，立即执行 trap 语句中定义的处理动作。

```
$ cat testing3
#!/bin/bash
# EXIT is the signal name generated upon exit from a script.
trap 'echo Variable Setting List: min = $min max = $max' EXIT
echo "This line will print before the \"trap\" statement."
echo "even though the \"trap\" statement occurs first."
min=$1
max=$2

if [ $min -gt $max ]
then
```

```

    exit 1
fi
$ testing3 6 8
This line will print before the "trap" statement.
even though the "trap" statement occurs first.
Variable Setting List: min = 6, max = 8
$
```

下面是另外一种例子，说明怎样使用“set -v”命令，使 Shell 能够显示其读入的每一条命令（包括注释行），以及执行命令后的输出结果。需要说明的是，在调试较大的 Shell 脚本时，这种方法并不适用——面对满屏幕的命令与输出信息，第一个感觉恐怕就是头痛，简直无从下手。

```

$ cat trace.sh
#!/bin/bash
LANG=C
set -v
# This is a comment
date
echo ${TZ}
set +v
date
exit 0
$ trace.sh
# This is a comment
date
Wed Nov 5 00:52:50 CST 2008
echo ${TZ}

set +v                                # 关闭 "-v" Shell 执行过程跟踪设置
Wed Nov 5 00:52:50 CST 2008
$
```

下面的例子说明怎样使用“set -x”命令显示 Shell 读入并执行的每一条命令。与“set -v”命令不同的是，Shell 将会在输出其读入的每个命令语句之前增加一个标记，如加号“+”字符（除了输出 PS4 的变量值作为标记之外，Shell 还会在之前插入一个与 PS4 变量值第一个字符相同的字符），以区别于命令的输出数据。同时，Shell 还会在输出命令之前，执行必要的变量替换、命令替换以及元字符文件名生成，然后才显示命令与替换后的实际参数。

```

$ cat trace2.sh
#!/bin/bash
LANG=C
set -x
# This is a comment
date
dir='pwd'
echo $dir
set +x
date
echo $HOME
exit 0
$ trace2.sh
+ date
Wed Nov 5 00:56:50 CST 2008
++ pwd
+ dir=/home/gqxing/script
+ echo /home/gqxing/script
/home/gqxing/script
+ set +x                                # 关闭 Shell 执行过程跟踪设置
Wed Nov 5 00:56:50 CST 2008
/home/gqxing
$
```

原则上，Shell 脚本允许使用未声明的变量。但利用“set -u”命令，可以禁止 Shell 使用事先未声明的变量。利用这一特性，可以发现变量名的拼写错误。下面的例子说明怎样使用“set -v”



和“`set -u`”命令显示 Shell 执行的每一条命令，一旦遇到事先未曾声明的变量，将会停止 Shell 脚本的运行，以便找出是否存在变量名拼写有误的错误。示例如下。

```
$ cat trace3.sh
#!/bin/bash
LANG=C
set -v
set -u
name=first
cat ${nmae}.file
wc -l ${name}.file
set +n
set +v
exit 0
$ trace3.sh
set -u
name=first
cat ${nmae}.file
trace3.sh: line 6: nmae: unbound variable
$
```

### 8.15.3 系统性能考虑

#### 1. PATH 变量的组织

PATH 变量设置的目录顺序直接决定了命令检索的速度。每当执行一个非 Shell 的内置命令时，系统都需要按照 PATH 变量指定的目录顺序检索命令。如果 PATH 变量设置的目录顺序不恰当，将会影响命令的检索速度。一种典型的错误设置是把当前目录放到 PATH 变量的标准目录组织顺序的前面。这种设置造成的后果是，每次执行命令时，系统都会首先检索当前目录。这便引出两个问题：当前目录下的程序究竟能够执行多少次？`/bin` 和 `/usr/bin` 等目录中的命令需要执行多少次？因此，在下列两个 PATH 变量的目录顺序设置方式中，一般应取前者而不是后者。

```
PATH=/bin:/usr/bin:.
PATH=.: /bin:/usr/bin
```

把当前目录放到 PATH 变量的前面，对命令检索的时间影响究竟有多大，还取决于当前目录的大小。如果当前目录较大，将直接影响 Shell 的运行性能。还有一个实际问题，如果把当前目录放到 PATH 变量的前面，则无法创建与`/bin` 或 `/usr/bin` 等目录中的命令同名的文件。

#### 2. 文件的访问方式

当需要访问同一目录中的大量文件时，最好利用 `cd` 命令首先进入该目录。在解释较长的目录文件名时，操作系统需要从路径名的第一个目录开始，逐层检索各级子目录，直至找到最终的文件名，因而需要花费较多的时间。如果给定的是相对于当前目录的文件名，操作系统直接访问给定的文件即可。

例如，下列两个脚本都是处理`/home/icbc/credit/report/month` 目录中的所有文件，但因采取的方法不同，其运行效率也不相同。

```
for fname in /home/icbc/credit/report/month/*
do
    cat $fname
done

cd /home/icbc/credit/report/month
for fname in *
do
    cat $fname
done
```

上述第一种处理方法要求操作系统解释较长的路径名，而第二种方法虽然增加了一个语句，

但操作系统只需在当前目录下直接访问每一个文件。

### 3. 内部命令与外部命令

作为解释程序的一部分，Shell 提供了若干内置命令语句。内置命令语句的执行速度明显快于外部的系统命令，其原因有 3 个。

(1) 执行内置命令语句不必创建新的进程。对于提高运行效率，这是一个非常重要的关键因素。因为在运行一个命令时，耗费时间最多的是创建新进程，其过程包括分配内存空间，填写进程控制表，从磁盘中把程序调入内存等。

(2) 执行内置命令语句时不需要按照 PATH 变量检索指定的目录。例如，当试图执行一个 test 命令时，Shell 根本就不会检索与内置命令同名的/usr/bin/test 命令或用户自己编写的 test 程序，除非明确指定必须执行/usr/bin 目录中的 test 命令或自己提供的 test 程序。

(3) 当执行一个 Shell 脚本时，系统首先需要打开这个文件，然后再逐行解释执行其中的命令语句。如果命令是一个编译的应用程序，则需要创建一个新进程，等到把程序装入内存后才能再执行，而 Shell 内置语句只需在现有的 Shell 进程中执行。

### 4. 管道中的命令顺序

在使用管道时，应适当地考虑过滤程序的位置顺序，逐步减少数据处理的信息量，以改善整个管道命令的运行效率。其中的一个原则是，要尽可能地把过滤程序安排在最前面，把非过滤的程序安排在后面，尤其要注意把比较耗时的 sort 程序放在过滤程序的后面。grep 命令就是一个较好的过滤程序的例子，可以滤掉大量不必要的数据，从而减少后续命令的数据处理量，提高整个管道命令的运行速度。因此，应像下面第一个例子那样，尽可能地把 grep 等过滤程序放在前面，而不是后面。

```
who | grep something | sort
who | sort | grep something
```

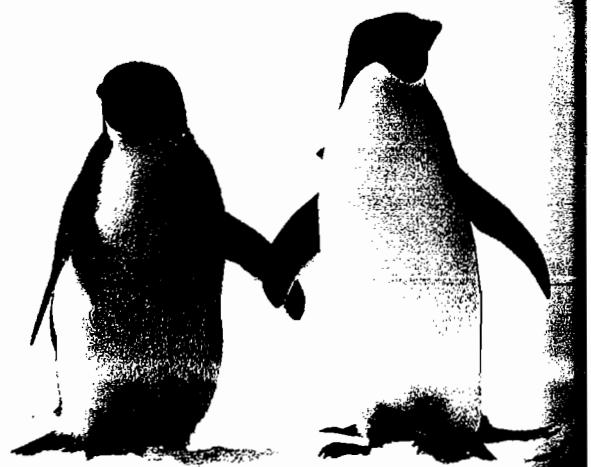


## 第9章

### 进程管理

进程是 Linux 系统中的重要概念。本章主要介绍怎样查询进程的状态信息，监控进程及系统资源。讨论怎样利用进程管理工具，找出耗时的无效进程，中止进程的运行，调整进程的优先级，使关键业务处理能够以较高的优先级运行，普通的例行业务能够以较低的优先级运行，从而提高系统的整体性能。其内容主要包括：

- ps 命令概述；
- 查询进程及其状态信息；
- 监控进程及系统资源；
- 终止进程的运行；
- 调整分时进程的优先级。



所谓的进程是指处于运行状态的程序。严格地说，进程是程序从开始调度运行直至终止执行的整个生命周期的全过程。进程管理是 Linux 系统中的一个重要组成部分，负责管理和控制所有的动态过程和资源（文件系统负责管理所有的静态信息和资源）。

通常，Linux 系统中的进程可分为两大类：系统进程和用户进程。系统进程主要负责 Linux 系统的生成、管理、维护和控制，包括 init 进程等。用户进程指的是由用户通过 Shell 命令行界面（或 GUI 桌面）提交系统运行的命令和应用程序等。除了少数进程外，系统中的所有进程几乎都是由 init 进程直接或间接启动的。也就是说，init 进程几乎是所有进程的直接或间接父进程。

每个进程都有一个系统赋予的进程标识（进程 ID），并与启动进程的用户（用户 ID）等相关联，所有的进程由进程子系统负责管理。用户可以查询进程的状态信息，但只能控制自己的进程，例如，向进程发送控制信号，重新启动和终止进程等等。只有超级用户才有权力控制所有的进程。

Linux 系统采用多用户、多任务的进程管理机制，保证所有处于竞争状态的进程都能够公平合理地分享系统资源，保证重要的进程能够优先分配到资源，普通用户和超级用户均可程度不同地、实时地调整活动进程的优先级。

## 9.1 ps 命令概述

在 Linux 系统中，获取进程状态信息的常用工具是 ps 命令。ps 命令可用于查询系统中活动进程的状态信息，如进程的起始运行时间和资源占用情况等，这些信息对进程和系统性能的管理都是非常有用的。ps 命令的语法格式简写如下。

```
ps [-aAcefFHlW] [-g grplist] [-p proclist] [-t term] [-u usrlist]
```

表 9-1 给出了 ps 命令的部分常用选项。

表 9-1 ps 命令的常用选项

选项	GNU 选项	描述
-a		显示系统中所有活动进程的当前状态信息（与终端无关的进程除外）
-A		显示系统中当前所有进程的状态信息，其作用等同于“-e”选项
-c		与“-l”选项一起使用时能够显示进程的调度信息，包括进程的调度类别与优先级等
-e		显示系统中当前所有进程的状态信息
-f		显示进程的重要状态信息，尤其是进程的起始运行时间和进程占用的 CPU 时间等
-F		与“-f”选项相比，能够显示更多重要的进程状态信息
-H		以表示进程调用层次关系的缩进形式显示所有进程的状态信息
--forest		以表示进程调用层次关系的树形结构显示所有进程的状态信息
-l		显示进程的详细状态信息（进程的起始运行时间除外）
-w		显示完整的进程信息。在显示进程信息时，每个进程通常仅占一行，如果进程的信息较长，命令字段的超长部分将会被截断。使用“-w”选项时，超长部分将会延续到下一行输出
-g group		显示与指定的有效用户组 ID 或用户组名有关的进程状态信息
-p pid		显示指定进程 ID 的进程状态信息
-t term		显示与指定的终端设备有关的进程状态信息
-u user		显示与指定的有效用户 ID 或用户名有关的进程状态信息



默认情况下，ps 命令仅显示当前用户自己的进程状态信息，示例如下。

```
$ ps
 PID TTY      TIME CMD
 6126 pts/0    0:00:00 bash
 6410 pts/0    0:00:00 ps
$
```

如果想要了解进程的更多信息，可在 ps 命令中增加“-e”或“-A”选项，这意味着显示所有进程的状态信息；增加“-f”、“-l”或“-F”等选项，将会输出更多的、重要的状态信息。也可以组合使用 ps 命令的各种选项，示例如下。

```
$ ps -ef
UID      PID  PPID  C STIME TTY      TIME CMD
root     1      0  0 10:22 ?        00:00:02 init
root     2      0  0 10:22 ?        00:00:00 [kthreadd]
root     3      2  0 10:22 ?        00:00:00 [migration/0]
root     4      2  0 10:22 ?        00:00:00 [ksoftirqd/0]
root     5      2  0 10:22 ?        00:00:00 [watchdog/0]
root     6      2  0 10:22 ?        00:00:00 [events/0]
root     7      2  0 10:22 ?        00:00:00 [khelper]
...
gqxing   6126  120  0 10:54 pts/0    00:00:00 bash
gqxing   6416  6126  0 11:18 pts/0    00:00:00 ps -ef
$
```

取决于提供的命令选项，ps 命令通常能够给出下列信息：

- 进程当前的工作状态 (S)；
- 进程标识 (PID)；
- 父进程标识 (PPID)；
- 用户标识 (UID)；
- 进程所处的调度级别 (CLS)；
- 进程的优先权 (PRI)；
- 进程的地址空间 (ADDR)；
- 进程占用的内存 (RSS)；
- 进程占用的 CPU 时间 (TIME)。

表 9-2 描述了 ps 命令的输出数据分为若干字段，这些字段是否能够全部出现以及何时出现，取决于提供的命令选项。

表 9-2 ps 命令输出的字段信息一览表

字段	描述
F	进程标志字段。下面是 ps 命令参考手册中提到的两个标志。 1：已经创建 (fork)，但尚未单独运行的 (exec) 的进程； 4：用到超级用户特权的进程
S 或 STAT	采用下列字符表示进程的当前状态： S 因等待某一事件的完成而处于休眠状态的进程（进程可以中断）； D 处于休眠状态，但不能中断的进程（通常为 I/O 进程）； R 表示正在运行的进程，或处于运行队列正在等待调度运行的进程，任何时刻，每个 CPU 只能有一个进程处于运行状态； X 进程已终止（通常见不到此类进程）； Z 僵尸进程（处于彻底终止运行之前中间阶段的进程）； W 已经完全导入磁盘交换区，没有任何页面仍位于内存中的进程（从 2.6.xx 内核开始已取消） T 由于跟踪或因作业控制信号导致进程中断执行而处于暂停运行状态的进程
UID	进程属主的有效用户 ID
PID	进程的进程 ID

续表

字段	描述
PPID	进程的父进程 ID
%CPU	进程迄今占用的 CPU 时间相对于全部 CPU 运行时间的百分比
%MEM	进程当前占用的实际物理内存数量相对于系统全部物理内存数量的百分比
CLS	进程的调度类别，如标准的分时进程调度类别 TS 等
PRI	进程的优先级。优先级的数字越大表示进程的优先级越高
NI	进程优先级的 nice 调整值（其范围为 -20~19），用于调整进程的优先级
ADDR	Proc 进程结构的地址空间
SZ	进程核心映像（包括代码段、数据段及栈空间）占用的物理内存页面的大小
WCHAN	因等待某一资源或事件的发生而使进程处于等待状态的内核函数的名字（“-”表示进程正在运行）
STIME 或 START	进程的起始运行时间。如果起始时间位于 24 小时之内，以 “HH:MM” 形式表示；如果超过 24 小时，则以 “mmmm dd” 形式表示起始运行时间，其中 mmmm 表示月（如英文月名的前 3 个字母），dd 表示日
TTY	控制终端。表示进程（或其父进程）是从哪一个终端上启动运行的。如果这个字段是问号 “?”，则表示进程与任何控制终端无关
TIME	从调度运行开始，进程迄今累计占用的 CPU 时间总和。以 “[dd-]hh:mm:ss” 形式表示
CMD 或 COMMAND	进程对应的命令或程序的名字。如果命令行过长，超出部分的选项或参数将被截断，使得每个进程仅占用一行显示位置（除非使用 “-w” 选项）
RSS	进程当前占用的物理内存数量（单位为 KB）
VSZ	进程当前占用的虚拟内存数量（单位为 KB）
C	进程生命周期的 CPU 利用率（百分比），即进程实际使用的 CPU 时间除以进程整个生命周期耗费的时间总量（包括等待时间）
PSR	进程当前分配到的处理器

## 9.2 查询进程及其状态信息

### 9.2.1 查询当前活动的进程

利用 “ps -a” 命令，可以显示当前系统中从终端中运行或调用的所有活动进程及其状态信息，示例如下。

```
$ ps -a
 PID TTY      TIME   CMD
 6256 pts/1    00:00:03  find
 7312 pts/0    00:00:00  ps
 $
```

从命令的输出结果可以看出，当前系统中直接从终端中运行或调用的活动进程只有 find 和 ps，其进程 ID 分别为 6256 和 7312，分别是在终端 /dev/pts/1 和 /dev/pts/0 中运行的。输出信息中的 TIME 字段表示进程从调度运行开始直至运行 ps 命令时占用的 CPU 时间，运行时间以时、分、秒为单位。其中，find 进程迄今耗费了 3 秒的 CPU 时间。

### 9.2.2 查询系统中的所有进程

ps 命令经常用于检查期望运行的进程是否已经启动，或者准备终止的进程是否已经停止运行。为了获取系统中当前调度运行的所有进程及其状态信息，可以使用 “ps -e” 或 “ps -A” 命令，示例如下。



```
$ ps -e
  PID TTY      TIME CMD
    1 ?        00:00:02 init
    2 ?        00:00:00 kthreadd
    3 ?        00:00:00 migration/0
    4 ?        00:00:00 kssoftirqd/0
    5 ?        00:00:00 watchdog/0
    6 ?        00:00:00 events/0
    7 ?        00:00:00 khelper
...
 6126 pts/0    00:00:00 bash
 6528 pts/0    00:00:00 ps
$
```

在上述输出信息中，TTY 字段的问号“?”表示系统中的许多进程都是由系统直接调度运行的，并非源于某个特定的终端。

利用“ps -e”命令能够给出当前系统中所有进程信息的特点，可以查询某个特定进程是否已经启动，这是 ps 命令的一个重要的应用。例如，要查询 Apache 服务器的守护进程 apache2 当前是否正在运行，可以使用下列组合命令。

```
$ ps -e | grep apache
 5580 ?        00:00:00 apache2
 5581 ?        00:00:00 apache2
 5584 ?        00:00:00 apache2
 5588 ?        00:00:00 apache2
$
```

也可以使用改进的 pgrep 或 pidof 命令，查询指定进程的运行状态。如果指定的进程已经启动且正在运行，下列命令将会给出活动进程的 PID。

```
$ pgrep apache
5580
5581
5584
5588
$ pidof mysqld
4765
$
```

### 9.2.3 显示进程的重要状态信息

系统管理员经常需要考察系统中的进程，监控并找出影响系统性能、危及系统安全的进程。使用 ps 命令的“-f”选项，可以获取进程的重要状态信息，示例如下。

```
$ ps -f
UID      PID  PPID  C STIME TTY      TIME CMD
gqxing   6126  6120  0 10:54 pts/0    00:00:00 bash
gqxing   6340  6126  0 10:59 pts/0    00:00:00 ps -f
$
```

从上方的输出结果可以看出，其中给出了很多重要的进程信息，包括用户标识、进程标识、父进程标识、进程起始运行时间和累计运行时间等。“-f”选项经常与“-e”选项一起使用，用于显示系统中所有进程的重要状态信息。这将是一个长长的进程列表，下面只是截取了其中的一部分。

```
$ ps -ef
UID      PID  PPID  C STIME TTY      TIME CMD
root     1      0  0 10:22 ?
root     2      0  0 10:22 ?
root     3      2  0 10:22 ?
root     4      2  0 10:22 ?
root     5      2  0 10:22 ?
root     6      2  0 10:22 ?
root     7      2  0 10:22 ?
$
```

```
gqxing 6120 1 0 10:54 ? 00:00:41 gnome-terminal
gqxing 6125 6120 0 10:54 ? 00:00:00 gnome-pty-helper
gqxing 6126 6120 0 10:54 pts/0 00:00:00 bash
gqxing 6255 6126 0 11:10 pts/0 00:00:00 ps -ef
$
```

利用 ps 命令可以监控系统中是否存在异常进程。针对 ps 命令输出的 STIME 和 TIME 字段，可以重点检查进程的起始运行时间和迄今为止累计占用的 CPU 时间。除非是系统进程，如果某个用户进程从启动至今无缘无故地耗费了大量的 CPU 时间，则很有可能该进程已处于无限循环状态。

### 9.2.4 显示进程的详细状态信息

利用 ps 命令的“-l”选项，可以列出每个活动进程的详细状态信息，示例如下。

```
$ ps -l
F S  UID  PID  PPID C PRI NI ADDR SZ WCHAN TTY      TIME CMD
4 S 1000 6126 6120 0 80 0 - 1478 wait pts/0 00:00:00 bash
0 R 1000 6286 6126 0 80 0 - 629 - pts/0 00:00:00 ps
$
```

利用 ps 命令的“-c”选项，可以列出每个活动进程所在的调度类别及其优先级，示例如下。

```
$ ps -cl
F S  UID  PID  PPID CLS PRI ADDR SZ WCHAN TTY      TIME CMD
4 S 1000 6126 6120 TS 19 - 1478 wait pts/0 00:00:00 bash
0 R 1000 6296 6126 TS 19 - 629 - pts/0 00:00:00 ps
$
```

如果再组合使用 ps 命令的“-e”选项，则可以列出所有进程所在的调度类别及其优先级等详细状态信息，示例如下。

```
$ ps -ecl
F S  UID  PID  PPID CLS PRI ADDR SZ WCHAN TTY      TIME CMD
4 S 0 1 0 TS 19 - 470 select ? 00:00:03 init
5 S 0 2 0 TS 24 - 0 kthrea ? 00:00:00 kthreadd
1 S 0 3 2 FF 139 - 0 migrat ? 00:00:00 migration/0
1 S 0 4 2 TS 24 - 0 ksfti ? 00:00:00 ksftirqd/0
5 S 0 5 2 FF 139 - 0 watchdog ? 00:00:00 watchdog/0
...
0 S 1000 6120 1 TS 19 - 30600 select ? 00:00:02 gnome-terminal
0 S 1000 6125 6120 TS 19 - 727 unix_s ? 00:00:00 gnome-pty-helper
0 S 1000 6126 6120 TS 19 - 1478 wait pts/0 00:00:00 bash
0 R 1000 6306 6126 TS 19 - 629 - pts/0 00:00:00 ps
$
```

上述输出信息说明，正处于休眠状态的进程 bash，其父进程是 6120，且是在进入 gnome-terminal 终端窗口后启动的。进程占用了 1478 个内存页面，现正于 wait 函数处等待用户的输入。

### 9.2.5 显示进程间的调用关系

ps 命令新增的“--forest”选项可以按照缩进形式，显示进程的调用层次关系，以及进程的状态信息，示例如下。

```
$ ps -ef --forest
UID  PID  PPID  C STIME  TTY      TIME      CMD
root 5428 1 0 10:23 ? 00:00:00 /usr/sbin/gdm
root 5431 5428 0 10:23 ? 00:00:00 \_ /usr/sbin/gdm
root 5435 5431 0 10:23 tty7 00:00:20 \_ \_ /usr/X11R6/bin/X :0 -br
gqxing 5729 5431 0 10:50 ? 00:00:00 \_ \_ x-session-manager
gqxing 5784 5729 0 10:50 ? 00:00:00 \_ \_ [scim] <defunct>
gqxing 5804 5729 0 10:50 ? 00:00:00 \_ \_ /usr/bin/seahorse-ag
gqxing 5806 5729 0 10:50 ? 00:00:00 \_ \_ /usr/lib/gnome-sessi
```



```

gqxing 5812 5729 0 10:50 ? 00:00:00 \_ /usr/bin/metacity --
gqxing 5834 5729 0 10:50 ? 00:00:03 \_ gnome-panel
gqxing 5843 5729 0 10:50 ? 00:00:02 \_ nautilus --no-desktop
gqxing 5916 5729 0 10:51 ? 00:00:00 \_ python /usr/share/sy
gqxing 5917 5729 0 10:51 ? 00:00:00 \_ update-notifier
...
gqxing 6120 1 0 10:54 ? 00:00:06 gnome-terminal
gqxing 6125 6120 0 10:54 ? 00:00:00 \_ gnome-pty-helper
gqxing 6126 6120 0 10:54 pts/0 00:00:00 \_ bash
gqxing 6542 6126 0 11:18 pts/0 00:00:00 \_ ps -ef --forest
$
```

## 9.2.6 pstree 命令

Linux 系统还提供一个 `pstree` 命令，可以采用树形缩进形式显示进程之间的调用关系，其语法格式简写如下。

```
pstree [-achlnpuH] [pid | user]
```

表 9-3 给出了 `pstree` 命令的部分常用选项及其简单说明。

表 9-3

`pstree` 命令的常用选项

选 项	简 单 描 述
<code>-a</code>	显示进程的命令行参数。如果进程已转储到交换区，将会在进程名的前后增加一对圆括号
<code>-c</code>	禁止压缩等同的进程子树。通常， <code>pstree</code> 命令将会尽可能地压缩进程子树的显示
<code>-h</code>	高亮度显示当前进程及其父进程
<code>-H [pid]</code>	高亮度显示指定的进程
<code>-l</code>	显示完整的进程信息。通常，超长部分将会被截断，以适应屏幕的显示宽度
<code>-n</code>	显示时按照进程 ID，而不是进程的名字排序父进程
<code>-p</code>	在每一个进程名之后，以外加圆括号的形式输出进程的 ID 号。通常不输出进程的 ID 号
<code>-u</code>	如果进程的 UID 与其父进程的 UID 不同，则在进程名之后以增加圆括号的形式输出用户 ID 号。通常不输出进程的 ID 号

取决于是否指定了进程，进程树形结构的根要么是指定的进程，要么是 `init` 进程。运行 `pstree` 命令时，如果未指定任何进程，则从 `init` 进程开始，按照树形缩进形式显示所有的进程；如果指定了某个进程 ID，则从指定的进程开始，按照树形缩进形式显示其后继的所有进程；此外，如果指定了用户名，则从某个进程开始显示指定用户拥有的所有进程的调用关系，示例如下。

```

$ pstree
init-->NetworkManager
|-acpid
|-apache2-->apache2
| `--2*[apache2---26*[(apache2)]]
|-atd
...
|---gnome-terminal---bash---pstree
|   |-gnome-pty-helper
|   |   |---(gnome-terminal)
|   |
|   $
```

“`gnome-terminal`”等行表示用户运行了多个进程，其中包括 `gnome-terminal`、`bash` 和当前正在运行的 `pstree` 进程。

如果增加了“`-p`”选项，`pstree` 命令还会给出每个进程的进程 ID，示例如下。

```
$ pstree -p
init(1)-->NetworkManager(5392)
```

```

|-acpid(4393)
|-apache2(5580)-+--apache2(5581)
|   |-apache2(5584)-+--{apache2}(5586)

.....
|   `--apache2(5588)-+--{apache2}(5590)

|-gnome-terminal(6120)-+--bash(6126)--pstree(7542)
|   |-gnome-pty-help(6125)
`--{gnome-terminal}(6127)

$
```

如果指定了进程 ID，则从指定的进程处开始，显示其所有后继进程之间的树形调用关系，示例如下。

```

$ pstree 6120
gnome-terminal---bash---pstree
                         |-gnome-pty-help
                         |-(gnome-terminal)
$ pstree -p 6120
gnome-terminal(6120)--bash(6126)--pstree(7602)
                         |-gnome-pty-help(6125)
                         |-(gnome-terminal)(6127)
$
```

## 9.3 监控进程及系统资源

前面介绍了怎样利用 ps 命令获取进程的各种状态信息。需要说明的是，ps 命令的输出只是系统执行命令那一刻的快照信息。为了能够连续地观察进程的实时状态信息以及其他系统信息，可以使用更受欢迎的 top 命令，其语法格式简写如下。

```
top [-hv | -bcisS] [-d delay] [-n iterations] [-p pids]
```

其中，“-d delay”选项表示数据取样和屏幕刷新的时间间隔，默认的取样时间间隔为 3 秒。delay 可以是一个整数，也可以是形如“ss.tt”的小数，时间单位为秒。

正如其名字所蕴涵的那样，top 命令仅仅列出系统中的前 17 个进程（取决于终端窗口的大小）及其他系统信息，并周期地进行更新。top 通常会按照进程的 CPU 占用率，从高到低对每个进程进行排序。

通常，top 每隔 3 秒钟更新一次数据，以反映系统的当前运行状态。top 命令的典型输出形式如图 9-1 所示。

dnlir@debs: ~																	
文件(E)		编辑(C)		查看(V)		终端(T)		标签(L)		帮助(H)							
PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND						
5851	gxqing	20	0	98452	356	16m	S	25.1	9.6	0:27.72	nautilus						
5433	root	20	0	218m	16m	7336	S	18.5	4.3	0:05.86	Xorg						
5810	gxqing	20	0	43284	14m	8696	S	10.6	3.8	0:20.94	metacity						
7083	root	20	0	4104	1152	680	S	7.9	0.3	0:01.12	mount.ntfs-3g						
5841	gxqing	20	0	62996	25m	15m	S	3.3	6.8	0:21.82	gnome-panel						
5898	gxqing	20	0	22856	9232	7708	S	2.6	2.4	1:48.09	geyes_applet2						
6025	gxqing	20	0	128m	26m	13m	S	2.0	7.1	1:04.38	gnome-terminal						
6965	gxqing	20	0	2412	1168	884	R	1.3	0.3	0:05.42	top						
5838	gxqing	20	0	49628	9388	5224	S	6.7	2.4	0:14.94	scim-panel-gtk						
5870	gxqing	20	0	25912	8164	6628	S	6.7	2.1	0:36.84	gnome-screensav						
7148	gxqing	20	0	18136	1692	1312	D	0.7	0.4	0:00.02	net						
1	root	20	0	1886	790	484	S	0.0	0.2	0:03.02	init						
2	root	15	-5	0	0	0	S	0.0	0.0	0:00.00	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	0:01.94	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	0:00.52	events/0						

图 9-1 top 命令主界面



`top` 命令输出的上半部分是系统运行状态的汇总信息，其中第 1 行包括系统的当前时间（18:39:47）、系统自启动以来的累计运行时间（6 小时 43 分）、注册到系统中的现有用户数量（3 个）以及系统的 3 个平均负载值（0.75、0.45 和 0.23）。第 2 行是进程的统计信息，其中包括系统中现有进程的总数（154）、处于运行状态的进程数量（1）、处于休眠状态的进程数量（151）、暂停运行的进程数量（1）以及处于僵尸状态的进程数量（1）等。第 3 行是对 CPU 工作状态的分类统计，其中包括 CPU 处于用户模式（61.8）、系统模式（10.0）、空闲状态（25.2）以及等待 I/O 状态（3.0）等的百分比。第 4 行是对内存使用情况的分类统计，其中包括系统配置的物理内存总量（384 356KB）、已用内存的数量（355 828KB）、空闲内存的数量（28 528KB）和用作缓冲区的内存数量（14 480KB）等。第 5 行是对交换区的分类统计，其中包括系统配置的交换区总量（748 400KB）、已用交换区的数量（13 704KB）、空闲交换区的数量（734 696KB）和用作缓冲区的交换区数量（74 212KB）等。表 9-4 给出了 `top` 命令上半部分的输出信息及简要说明。

表 9-4 `top` 命令上半部分的输出信息及说明

字 段	简 单 说 明
hh:mm:ss	第 1 行给出的是系统当前运行状态的汇总统计信息。其中的“hh:mm:ss”字段表示系统的当前时间，以小时、分和秒的形式，表示 <code>top</code> 命令给出的是此刻的系统状态信息
up	系统自启动以来迄今为止的运行时间
users	系统中现有的注册用户数量
load averages	其中给出了 3 个平均系统负载值。每个数值分别表示在最近 1 分钟、5 分钟和 15 分钟内系统运行队列的平均长度
Tasks	第 2 行是进程运行状态信息的汇总统计。其中第一个字段（total）表示系统当前共有多少个进程，其他字段分别表示处于下列每一种运行状态下的进程数量： <ul style="list-style-type: none"><li>□ running 正在运行，或已处于运行队列，一旦调度即可运行的进程数量。注意，这个数字有可能会大于系统配置的 CPU 个数。</li><li>□ sleeping 因等待外部事件完成而处于休眠状态的进程数量。</li><li>□ stopped 因跟踪调试或暂时停止运行的进程数量。</li><li>□ zombie 僵尸进程的数量</li></ul>
Cpu(s)	第 3 行是 CPU 工作状态的分类统计信息，分述如下： <ul style="list-style-type: none"><li>□ %us CPU 处于用户模式的时间量所占的百分比；</li><li>□ %sy CPU 处于系统模式的时间量所占的百分比；</li><li>□ %ni CPU 处理其优先级经 nice 值调整过的用户进程的时间量所占的百分比；</li><li>□ %id CPU 处于空闲状态的时间量所占的百分比；</li><li>□ %wa CPU 等待 I/O 完成的时间量所占的百分比；</li><li>□ %hi CPU 处理硬件中断的时间量所占的百分比；</li><li>□ %si CPU 处理软件中断的时间量所占的百分比</li></ul>
Mem	第 4 行是系统物理内存使用情况的汇总统计信息。其中每个字段的意义分述如下： <ul style="list-style-type: none"><li>□ total 系统配置的可用物理内存总数；</li><li>□ used 当前已经使用的物理内存计数；</li><li>□ free 当前仍然空闲的物理内存计数；</li><li>□ buffers 用作系统缓冲区的物理内存计数</li></ul>
Swap	第 5 行是交换区使用情况的汇总统计信息。其中的每个字段意义如下： <ul style="list-style-type: none"><li>□ total 系统配置的交换区总数；</li><li>□ used 当前已经使用的交换区计数；</li><li>□ free 当前仍然空闲的交换区计数；</li><li>□ cached 用作系统缓存的交换区计数</li></ul>

从图 9-1 的输出数据中可以看出, 截止到 18 时 39 分 47 秒, 在最近 1、5 和 15 分钟之内, 系统的平均负载量分别是 75%、45% 和 23%。总共有 154 个进程, 但除了文件浏览器 nautilus 进程之外, 其他大部分进程均处于休眠状态。CPU 有 25.2% 的时间是空闲的, 61.8% 的时间直接服务于用户, 10.0% 的时间用于系统开销。通过这些数据, 可以了解系统资源的使用情况。

**top** 输出的第二部分根据每个进程的 CPU 占用量, 按照从高到低的顺序, 给出每个进程的状态信息。其中, PID 是进程 ID, USER 是进程属主的用户名, PR 是进程的优先级, NI 是进程的 nice 调整值(范围是 -20~20), VIRT 是进程分配的整个虚拟内存空间(包括代码段和数据段等)的大小, RES 是进程当前实际驻留的物理内存大小, SHR 是进程占用的共享内存空间数量, S 是进程的当前状态(D、R、S、T 或 Z), %CPU 是进程占用 CPU 时间的百分比, %MEM 是进程当前占用的物理内存的百分比, TIME(TIME+) 是进程占用的系统和用户 CPU 时间的累计数量, COMMAND 是当前进程对应的命令或程序(参见表 9-5)。

表 9-5 top 提供的进程状态信息

字段	简单说明
PID	系统分配给每个进程的进程 ID。进程 ID 通常是一个小于 65 536 的正整数。一旦进程结束或被强行终止, 进程 ID 还可以重用
USER	启动或拥有进程的用户名
PR	分配给每个进程的优先级, 用于确定进程调度的优先顺序。Linux 内核在计算和分配优先级时需要考虑许多因素, 在进程的整个生命周期中, 这个数值通常不会存在大的起伏变动
NI	进程优先级的 nice 调整值。nice 调整值可以是负数, 表示提高优先级。在 Linux 系统中, nice 调整值的取值范围是 -20~20。大多数用户进程采用 0 作为优先级的 nice 调整值, 表示采用基本的优先级。但也可以选择使用一个大于 0 的 nice 调整值启动进程, 使系统能够降低进程的优先级。对于一个需要长时间占用 CPU(运行时间长, 以 CPU 处理为主)的处理任务来讲, 这是一种常规的做法, 可以避免干扰交互式进程。但只有超级用户才能采用负数提高进程的优先级。Linux 系统中的 nice 或 renice 命令可用于设置进程优先级的 nice 调整值
VIRT	进程分配的虚拟内存映像空间数量(以 KB 为单位), 是进程使用的虚拟内存总和, 其中包括代码段、数据段和共享库占用的所有空间, 以及转储到交换区的页面数量。进程的虚拟内存映像空间数量可用下式表示: $VIRT = SWAP + RES$
RES	进程占用的基本物理内存(即非交换内存)空间数量(以 KB 为单位)。一个进程可以申请分配较大的虚拟内存空间, 但通常只能使用非常小的物理内存。进程的基本物理内存可用下式表示: $RES = CODE + DATA$
SHR	进程占用的共享内存空间数量(以 KB 为单位)。这是为进程分配的全部内存空间数量, 即整个虚拟内存空间的数量。这个数量等于进程的代码段、数据段或栈段等内存空间的总和
S	当前的进程状态信息。任何进程总是处于下列状态之一: <input type="checkbox"/> D 进程处于不可中断的休眠状态; <input type="checkbox"/> R 进程正在运行, 或者已处于运行队列, 一旦调度即可运行; <input type="checkbox"/> S 进程因等待外部事件的完成而处于休眠状态; <input type="checkbox"/> T 进程因跟踪调试, 或因收到某个信号(如 Ctrl-Z)而暂时停止运行; <input type="checkbox"/> Z 进程已终止, 但其父进程尚未完成善后处理
%CPU	进程占用 CPU 的百分比。 <b>top</b> 通常以此列数据基准, 按照从大到小的顺序显示进程
%MEM	进程当前占用的物理内存的百分比
TIME/TIME+	进程累计占用的 CPU 时间(TIME 以秒为单位, TIME+以 1/100 秒为单位)。表示自开始运行以来, 迄今为止进程占用的 CPU 时间。这个时间量是运行相应进程时实际耗费的 CPU 时间
COMMAND	启动进程的命令行或程序名



实际上，top 命令还可以给出更多的统计信息（只是受限于屏幕的显示宽度）。top 采用一个由 26 个英文字母组成的字符串 AbcdEfgHijKlmnoPqrstuvwxyz，分别表示 PID、PPID、RUSER、UID、USER、GROUP、TTY、PR、NI、P、%CPU、TIME、TIME+、%MEM、VIRT、SWAP、RES、CODE、DATA、SHR、nFLT、nDRT、S、COMMAND、WCHAN 和 Flags 等 26 个字段。在上述字符串中，大写字母表示输出相应的字段，且按字母的排列顺序输出相应的字段。为了显示其他字段信息，可以输入“f”字符命令，然后键入相应字段的小写字母，把原来的小写字母改为大写字母。要改变输出字段的排列顺序，可以输入“o”字符命令，连续输入任何小写字母，可以把相应的输出字段依次后移一个位置；连续输入任何大写字母，可以把相应的输出字段依次前移一个位置。通过调整字母的排列位置，可以达到调整输出字段的先后顺序的目的，如图 9-2 所示。

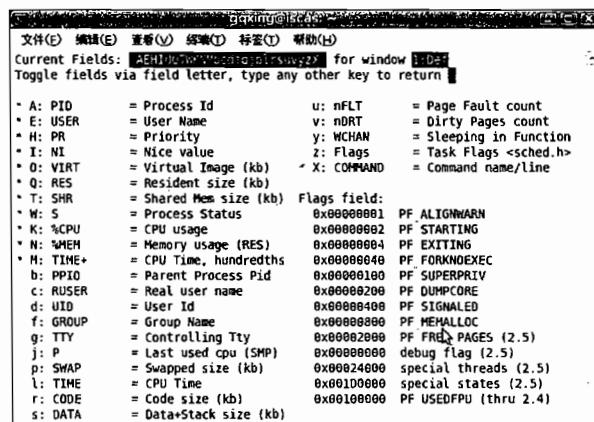


图 9-2 top 命令输出配置界面

表 9-6 是对 top 命令提供的其他进程状态信息的简单说明。

表 9-6

top 提供的其他进程状态信息

字 段	简 单 说 明
SWAP	进程的虚拟内存映像转储到交换区部分的空间数量（以 KB 为单位）
CODE	可执行代码段占用的物理内存数量（以 KB 为单位）
DATA	除了可执行代码之外的数据段和栈占用的物理内存数量（以 KB 为单位）
nFLT	页面故障计数。当进程尝试读取或写入一个虚拟的内存页面，而其地址空间不存在相应的页面时，即会出现页面故障，导致系统访问磁盘
nDRT	数据修改后尚未写入磁盘的内存页面计数。在能够把这些物理内存空间作为虚拟内存页面分配给其他进程之前，必须把其中的数据写入磁盘
WCHAN	进程处于休眠状态所在的函数。取决于系统内核的编译与链接状况，这个字段通常会给出系统内核休眠时所处函数的名字或地址。对于正在运行的进程而言，这个字段位置将可能存在一个减号“-”标志
Flags	表示进程的当前调度标志（参见/usr/include/linux/sched.h 文件和图 9-2）

top 命令在输出各种统计信息，不断刷新屏幕的同时，也提供一个交互界面。利用 top 命令支持各种交互命令，可以影响 top 命令的输出结果。例如，输入“n”或“#”字符命令，可以限定 top 能够显示的进程数量。输入“h”字符命令，可以显示 top 命令交互操作的使用说明，如图 9-3 所示。要退出 top 命令，可以直接输入“q”字符命令。

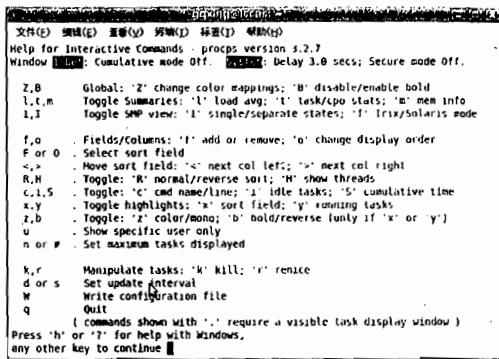


图 9-3 top 命令交互操作说明

## 9.4 终止进程的运行

有时，用户需要强行终止某个进程。例如，进程的运行时间过长，或者进程因逻辑错误而陷于无限循环状态等。普通用户只能终止自己的进程，只有超级用户才能终止系统中的任何进程（进程标识号较小的内核进程除外，终止此类进程通常会引起系统瘫痪）。

Linux 系统既支持 POSIX 标准信号，也支持 POSIX 实时信号。在收到信号之后，进程通常采取下列默认的处理动作之一：

- 终止进程的执行；
- 忽略收到的信号，进程继续执行；
- 终止进程的执行，并把进程的内存映像转储到一个 core 文件；
- （暂时）停止进程的执行。

要强行终止一个进程，可使用 Bash 的内部命令 kill 或/bin/kill 命令。kill 命令的语法格式简写如下。

```
kill [-s sigspec | -n signum | -sigspec] pid ...
kill -l [sigspec]
```

其中，sigspec 既可以是一个规范的信号名（如 SIGKILL），或缩写的信号名 KILL，也可以是一个信号编码。signum 是一个信号编码，表示 kill 命令发送给指定进程的信号（参见表 9-7 及 /usr/include/bits/signum.h 文件）。pid 是进程 ID。此外，利用“-l”选项还可以列出系统支持的所有信号（或指定信号）。

表 9-7 Linux 系统支持的部分信号

信 号	信 号 名	默 认 的 处 理 动 作	简 单 说 明
1	SIGHUP 或 HUP	终 止 进 程	挂断。终端通信连接断开或控制进程终止时产生的信号
2	SIGINT 或 INT	终 止 进 程	中断。按下中断键（即 Ctrl-C 键）时产生的信号
3	SIGQUIT 或 QUIT	终 止 进 程, 生成内存映像文件(core)	退出。按下 Quit 键（即 Ctrl-\ 或 Ctrl-Shift-\ 键）时产生的信号
4	SIGILL 或 ILL	终 止 进 程, 生成内存映像文件(core)	非法指令。执行非法机器指令时产生的信号，其中包括由非法操作码、非法操作数、非法寻址模式、寄存器或内部栈错误等原因产生的信号
5	SIGTRAP 或 TRAP	终 止 进 程, 生成内存映像文件(core)	硬件故障或断点跟踪等情况产生的信号
6	SIGABRT 或 ABRT	终 止 进 程, 生成内存映像文件(core)	异常终止信号（由 abort() 函数产生的异常终止信号）



续表

信 号	信 号 名	默认的处理动作	简 单 说 明
7	SIGBUS 或 BUS	终止进程,生成内存映像文件(core)	总线故障信号。包括由非法地址、不存在的地址或其他硬件错误等原因产生的信号
8	SIGFPE 或 FPE	终止进程,生成内存映像文件(core)	浮点算术运算异常(如除数为零或浮点运算溢出)时产生的信号
9	SIGKILL 或 KILL	终止进程	进程无法捕捉与封锁,也不能忽略的终止信号。可供系统管理员使用 kill 命令终止任何进程(部分核心进程除外)
10	SIGUSR1 或 USR1	终止进程	用户定义的信号 1, 供应用编程使用
11	SIGSEGV 或 SEGV	终止进程,生成内存映像文件(core)	内存地址越界或访问权限不足时产生的信号(当代码地址超出进程的地址空间时产生的信号)
12	SIGUSR2 或 USR2	终止进程	用户定义的信号 2, 供应用编程使用
13	SIGPIPE 或 PIPE	终止进程	管道断开信号(当进程写入一个管道或套接字,但读取管道或套接字的另一个进程已经终止时产生的信号)
14	SIGALRM 或 ALRM	终止进程	由 alarm()系统调用产生的时钟超时信号
15	SIGTERM 或 TERM	终止进程	终止进程信号。这是 kill 命令产生的默认终止信号
16	SIGSTKFLT 或 STKFLT	终止进程	栈故障信号
17	SIGCHLD 或 CHLD	忽略	子进程状态发生变动信号,如子进程结束或停止运行时向父进程发送的信号
18	SIGCONT 或 CONT	继续(或忽略)	令进程继续运行的信号,主要用于作业控制。如果进程已暂停运行,则默认的处理动作是继续运行。如果进程正在运行,则默认的处理动作是忽略收到的信号
19	SIGSTOP 或 STOP	停止进程	停止进程运行信号。这是用于停止一个进程的作业控制信号,这个信号主要用于作业控制,既不能被捕捉,也不能被忽略
20	SIGTSTP 或 TSTP	停止进程	键盘停止信号。在交互方式下,通过按 Ctrl-Z 键停止当前进程时产生的信号。主要用于作业控制
21	SIGTTIN 或 TTIN	停止进程	后台进程试图从控制终端中读取数据时产生的信号。主要用于作业控制
22	SIGTTOU 或 TTOU	停止进程	后台进程试图向控制终端输出数据时产生的信号。主要用于作业控制
23	SIGURG 或 URG	忽略	当从网络连接(套接字)中接收到的数据发生错误,通知进程出现紧急情况时发送的信号
24	SIGXCPU 或 XCPU	终止进程,生成内存映像文件(core)	当进程超过了其最大的软性 CPU 时间限制时产生的信号
25	SIGXFSZ 或 XFSZ	终止进程,生成内存映像文件(core)	当进程创建的文件超过了其能够创建的软性最大文件容量限制时产生的信号
26	SIGVTALRM 或 VTALRM	终止进程	由 setitimer()系统调用设置的虚拟间隔时钟超时信号
27	SIGPROF 或 PROF	终止进程	由 setitimer()系统调用设置的内核抽样间隔时钟超时信号
28	SIGWINCH 或 WINCH	忽略	窗口大小发生变动时产生的信号
29	SIGIO 或 IO	终止进程	异步 I/O 事件出现后产生的信号
30	SIGPWR 或 PWR	忽略	电源故障信号。当电源出现故障,改由 UPS 提供系统电源供电时产生的信号。这个信号通常主要由操作系统处理,应用程序不受影响,除非 UPS 电源供电也不充足
31	SIGSYS 或 SYS	终止进程,生成内存映像文件(core)	系统调用有误。发现非法系统调用(如参数有误)时产生的信号

如果运行 kill 命令时未指定任何信号，则发送默认的信号 15 (SIGTERM)。在表 9-5 中给出的信号中，信号 9 (SIGTERM) 是不能被捕捉的，因而可用于强行终止任何进程。也就是说，在 kill 命令中使用信号 9，可以确保进程无条件终止。然而，不能轻易使用信号 9。建议不要使用信号 9 终止诸如数据库服务器等进程，以免造成数据的丢失。

任何用户均可以终止自己的进程，但如果想终止其他用户的进程，则必须拥有超级用户的权限。要终止进程的继续运行，通常可参照下列步骤。

首先，可以使用 ps、pgrep 或 pidof 命令获取进程的 PID。例如，下列 ps 命令用于显示属于 telnetd 或以 telnetd 用户身份启动的所有进程。

```
$ ps -fu telnetd
UID      PID  PPID  C STIME   TTY      TIME CMD
telnetd  6370  5072  0 14:38 ?        00:00:00 in.telnetd: 169.254.78.56
telnetd  6881  5072  0 18:14 ?        00:00:00 in.telnetd: 169.254.78.66
$
```

此外，还可以使用 pgrep 和 pidof 命令直接获取指定进程的 PID。实际上，pgrep 和 pidof 命令是对 ps 或 ps 与 grep 组合命令的改进，可以直接给出指定进程的 PID。例如，下列 pidof 的输出结果表示，系统中当前共启动了 4 个 apache2 守护进程。

```
$ pidof apache2
5585 5581 5579 5578
$
```

下列 pgrep 命令的输出结果表示当前系统中共有 3 个注册 Shell——bash 在运行。

```
$ pgrep bash
6030
6377
6565
$
```

然后，可以使用 kill (或 pkill) 命令终止进程的继续运行。例如，要终止 Telnet 远程注册的守护进程 telnetd，禁止任何远程访问，可以使用下列 kill 命令。

```
$ sudo kill -9 6370 6881
$
```

注意，利用 kill 命令终止一个进程时，在正式使用上述 kill 命令之前，首先应尝试使用不带任何信号参数的 kill 命令，然后再观察进程是否已经终止。如果不奏效，再使用信号 9。

最后，可以利用 ps、pgrep 或 pidof 命令验证进程是否已经终止——如果进程确已终止，ps、pgrep 和 pidof 等命令不会输出任何结果。

## 9.5 调整分时进程的优先级

进程的优先级是由系统的进程调度程序根据调度策略分配的。用户可以利用 nice 或 renice 命令调整进程的优先级。

### 9.5.1 nice 命令

nice 命令是与 UNIX 系统兼容的进程优先级调整命令，renice 命令是伯克利版 renice 命令的实现，具有更灵活的进程管理功能。一个进程的优先级是由进程所在调度类别的调度策略确定的，具体地讲，是由进程的全局优先级与 nice 调整值共同确定的。实际上，确定进程的优先级涉及许多因素，其中包括进程调度类别、进程已经占用的 CPU 时间以及在分时进程情况下的 nice 调整值等。



在开始运行之后，也可以通过 nice 或 renice 命令对进程的优先级予以适当的调整。从下列 ps 命令输出的 NI 字段可以看到每个进程的优先级与 nice 调整值。

```
$ ps -el
F S  UID   PID  PPID C PRI  NI ADDR SZ WCHAN TTY          TIME CMD
4 S  0     1    0  0  80  0 -  470 select ?      00:00:03 init
5 S  0     2    0  0  75 -5 -  0 kthrea ?      00:00:00 kthreadd
1 S  0     3    2  0 -40 - -  0 migrat ?      00:00:00 migration/0
1 S  0     4    2  0  75 -5 -  0 ksofti ?      00:00:00 ksoftirqd/0
5 S  0     5    2  0 -40 - -  0 watchdog ?     00:00:00 watchdog/0
1 S  0     6    2  0  75 -5 -  0 worker ?     00:00:00 events/0
...
0 S 1000  6120    1  0  80  0 - 30923 poll  ?      00:00:17 gnome-terminal
0 S 1000  6125 6120  0  80  0 -  727 unix_s ?     00:00:00 gnome-pty-help
0 S 1000  6126 6120  0  80  0 - 1477 read_c pts/0  00:00:00 bash
0 R 1000  7773 6126  0  80  0 -  629 -  pts/0  00:00:00 ps
$
```

通过增加 nice 调整值，普通用户可以降低一个进程的优先级。但只有超级用户才能通过减少 nice 调整值，达到增加进程优先级的目的。这个限制可防止用户增加自己的进程优先级，因而无止境地独占共享的 CPU 时间。对于超级用户而言，nice 调整值的取值范围为-20~19，其中-20 是可用的最大 nice 调整值。对于普通用户而言，nice 调整值的取值范围为 0~19，其中 19 是可用的最小 nice 调整值。

nice 命令的语法格式简写如下。

```
nice [-n number] [command [arguments]]
```

其中，“-n”选项用于指定进程的 nice 调整值。如果未指定“-n”选项，默认的 nice 调整值是 10。

按照上述说明，普通用户只能降低进程的优先级，而超级用户既可降低进程的优先级，又能提高进程的优先级。要修改进程的优先级，针对普通用户与超级用户，分述如下。

作为一个普通用户，如果确实需要改变进程的优先级，使一个运行时间较长的普通应用程序能够以一个较低的优先级运行，以免影响其他业务的处理，可通过提高 nice 调整值，降低指定命令的优先级。

例如，下列 nice 命令通过增加 5 个 nice 调整值，以一个相对较低的优先级执行给定的命令（在未使用 nice 命令之前，find 进程的优先级与 nice 调整值分别为 80 与 0；使用 nice 命令之后，其优先级与 nice 调整值分别为 85 与 5，find 进程的优先级最终降低为 85）。

```
$ find / -name core > /dev/null 2>&1 &
[1] 6445
$ ps -el | grep find
0 D 1000 6445 6377 12 80  0 -  595 sync_b pts/0  00:00:02 find
$ nice -n 5 find / -name core > /dev/null 2>&1 &
[2] 6448
$ ps -el | grep find
0 D 1000 6448 6377  8 85  5 -  549 sync_b pts/0  00:00:03 find
$
```

作为一个超级用户，如果确实需要改变进程的优先级，使一个关键业务处理能够以一个较高的优先级运行，而普通的应用能够以较低的优先级运行，可以通过降低或提高 nice 调整值，相应地提高或降低进程的优先级。

例如，下列 nice 命令通过降低 10 个 nice 调整值，相应地提高指定命令的优先级（在未使用 nice 命令之前，find 进程的优先级与 nice 调整值分别为 80 与 0；使用 nice 命令之后，其优先级与 nice 调整值分别为 70 与-10，find 进程的优先级最终提高为 70）。

```
$ sudo nice -n -10 find / -name core > /dev/null 2>&1 &
[1] 6478
$ ps -el | grep find
4 D 1000 6478 6377  8 70 -10 -  963 sync_b pts/0  00:00:01 find
$
```

注意，如果赋予进程较高的 nice 调整值，有可能会过多地挤占其他进程的 CPU 时间，从而影响系统的整体性能。

### 9.5.2 renice 命令

Linux 系统提供的另外一个调整进程优先级的命令是 `renice`，可用于调整正在运行的一个或多个进程的优先级。`renice` 命令的语法格式简写如下。

```
renice priority [[-p] pids] [[-u] users]
```

其中，`priority` 是一个整数数值，表示赋予进程的 nice 调整值：正数表示降低进程优先级的 nice 调整值，负数表示增加进程优先级的 nice 调整值，0 表示保持进程的基本优先级不变。“`-p`”选项用于指定进程的进程 ID，“`-u`”选项用于指定属于某个特定用户的所有进程。

同样，在调整进程的优先级时，普通用户只能利用 `renice` 命令增加 nice 调整值（其范围为 0~20），从而降低进程的优先级。超级用户可以利用 `renice` 命令，调整任何进程的优先级，增加或减少 nice 调整值，其范围为 -20~20。如果把 nice 调整值设定为 20，则仅当系统中没有其他可以调度运行的进程时，才会调度执行 `renice` 命令指定的进程。采用负数增加进程的优先级时，进程能够得到更多的调度机会，从而运行得更快。nice 调整值 -20 是超级用户可用的最大优先级。

例如，下列命令可以改变进程 ID 号为 7040 和 7043，以及用户 `www-data` 拥有的所有进程的优先级。

```
$ sudo renice +1 -p 7040 7043 -u www-data
7040: old priority 0, new priority 1
7043: old priority 0, new priority 1
33: old priority 0, new priority 1
$
```

### 9.5.3 调整进程优先级的作用

下面将利用 `time` 命令，分别采用提高或降低进程优先级 nice 调整值的方式运行同一命令，观察修改 nice 调整值前后的运行效果。

在此之前，首先简单介绍一下 `time` 命令。`time` 命令可用于考察一个命令的执行效率。当利用 `time` 命令提交的程序运行结束之后，`time` 命令将会从以下 3 个方面，给出在运行指定的程序时占用的时间。

- **real**: 表示运行给定程序时耗费的全部时间（以秒为单位）——这个时间包括从调用程序开始，直至程序执行结束期间的全部时间。
- **user**: 程序的实际运行时间（以秒为单位）。这个时间仅包含系统执行用户代码时占用的时间，而不包括操作系统执行系统调用等开销。
- **sys**: 表示在操作系统核心地址空间中执行代码期间占用的时间（以秒为单位）。这个时间包括内存分配、系统调用及数据 I/O 等开销。

现在，让我们回过头来考察在提高或降低进程 nice 调整值前后，执行同一个命令时有何变化。

下面的例子是在提高 10 个 nice 调整值时，运行 `dd` 命令的结果。

```
$ sudo time nice -n -10 dd if=/dev/sda8 of=/dev/null
23768577+0 records in
23768577+0 records out
12169511424 bytes (12 GB) copied, 1676.81 s, 7.3 MB/s

real    27m56.990s
user    0m40.333s
sys     3m34.508s
$
```



下面的例子是在降低 10 个 nice 调整值时运行 dd 命令的结果。

```
$ sudo time nice -n 10 dd if=/dev/sda8 of=/dev/null
23768577+0 records in
23768577+0 records out
12169511424 bytes (12 GB) copied, 2124.78 s, 5.7 MB/s

real    35m25.154s
user    0m39.044s
sys     3m31.094s
$
```

从上面两个以不同进程优先级运行 dd 命令的输出结果可以看出两点。两者在程序的实际执行时间方面并无太大的差别。这是可以理解的，因为执行的是同一个程序，其代码和处理的内容完全一样。但两者在执行程序时耗费的全部时间方面差别却非常之大，这是由于具有不同优先级的进程在系统的进程调度方面得到的运行机会不同而造成的。进程的优先级越低，得到调度运行的机会就越少；反之，得到调度运行的机会就越多。

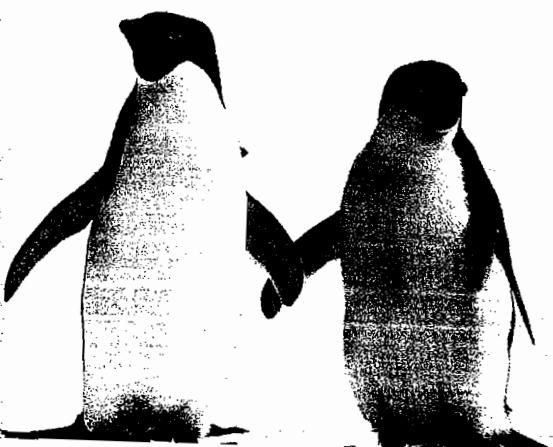
nice 和 renice 命令的主要用途是降低一些运行时间较长，但又并非紧急事务的进程，使之能够以较低的优先级运行，以提高系统对关键业务的响应速度，从而达到提高整体系统性能的目的。

# 第10章

## proc文件系统

proc文件系统是一个虚拟文件系统，不占用任何磁盘空间。proc文件系统是系统运行状况的动态反映，可以用作系统内核数据结构的接口，以便用户获取进程状态以及可调参数等信息，而不必直接读取或解释 /dev/kmem 核心内存文件。proc文件系统通常安装在/proc目录下，其中的大多数文件只能读，部分与系统内核变量有关的文件允许修改，借以调整系统内核的参数。本章讨论的主要内容包括：

- 进程内存映像文件；
- 系统配置信息；
- 系统运行状态信息；
- 系统内核可调参数；
- 其他重要的子目录。





proc 文件系统中的目录或文件大体上可以分为 3 部分：系统当前所有进程的内存映像、系统的当前配置与运行状态信息以及系统内核的可调参数。

## 10.1 进程内存映像文件

/proc 目录下所有以数字形式命名的目录均为进程的内存映像。对于当前正在运行的每一个进程，均存在一个对应的目录，目录的名字以相应进程的进程 ID 命名，其中，目录 1 对应于著名的 init 进程，示例如下。

```
$ ls -l /proc  
总用量 0  
dr-xr-xr-x 7 root      root      0 2008-12-29 11:25 1  
dr-xr-xr-x 7 root      root      0 2008-12-29 12:03 122  
dr-xr-xr-x 7 root      root      0 2008-12-29 12:03 126  
dr-xr-xr-x 7 root      root      0 2008-12-29 12:03 1266  
dr-xr-xr-x 7 root      root      0 2008-12-29 12:03 1267  
dr-xr-xr-x 7 root      root      0 2008-12-29 12:03 1280  
dr-xr-xr-x 7 root      root      0 2008-12-29 12:03 1282  
....  
$
```

在每个以进程 ID 命名的目录中，均存在一系列虚拟文件或目录。例如，目录 1 中存在下列文件和目录。

```
$ ls -l /proc/1  
总用量 0  
dr-xr-xr-x 2 root root 0 2008-12-29 12:05 attr  
-r----- 1 root root 0 2008-12-29 12:05 auxv  
-r--r--r-- 1 root root 0 2008-12-29 12:05 cgroup  
--w----- 1 root root 0 2008-12-29 12:05 clear_refs  
-r--r--r-- 1 root root 0 2008-12-29 12:05 cmdline  
-rw-r--r-- 1 root root 0 2008-12-29 12:05 coredump_filter  
-r--r--r-- 1 root root 0 2008-12-29 12:05 cpuset  
1rwxrwxrwx 1 root root 0 2008-12-29 12:05 cwd  
-r----- 1 root root 0 2008-12-29 12:05 environ  
1rwxrwxrwx 1 root root 0 2008-12-29 12:05 exe  
dr-x----- 2 root root 0 2008-12-29 11:25 fd  
dr-x----- 2 root root 0 2008-12-29 12:05 fdinfo  
-r--r--r-- 1 root root 0 2008-12-29 12:05 io  
-r--r--r-- 1 root root 0 2008-12-29 12:05 latency  
-r----- 1 root root 0 2008-12-29 12:05 limits  
-rw-r--r-- 1 root root 0 2008-12-29 12:05 loginuid  
-r--r--r-- 1 root root 0 2008-12-29 12:05 maps  
-rw----- 1 root root 0 2008-12-29 12:05 mem  
-r--r--r-- 1 root root 0 2008-12-29 12:05 mountinfo  
-r--r--r-- 1 root root 0 2008-12-29 12:05 mounts  
-r----- 1 root root 0 2008-12-29 12:05 mountstats  
dr-xr-xr-x 7 root root 0 2008-12-29 12:05 net  
-rw-r--r-- 1 root root 0 2008-12-29 12:05 oom_adj  
-r--r--r-- 1 root root 0 2008-12-29 12:05 oom_score  
-r----- 1 root root 0 2008-12-29 12:05 pagemap  
1rwxrwxrwx 1 root root 0 2008-12-29 12:05 root  
-rw-r--r-- 1 root root 0 2008-12-29 12:05 sched  
-r--r--r-- 1 root root 0 2008-12-29 12:05 schedstat  
-r--r--r-- 1 root root 0 2008-12-29 12:05 sessionid  
-r--r--r-- 1 root root 0 2008-12-29 12:05 smaps  
-r--r--r-- 1 root root 0 2008-12-29 12:05 stat  
-r--r--r-- 1 root root 0 2008-12-29 12:05 statm  
-r--r--r-- 1 root root 0 2008-12-29 12:05 status
```

```
dr-xr-xr-x 3 root root 0 2008-12-29 12:05 task
-r--r--r-- 1 root root 0 2008-12-29 12:05 wchan
$
```

在上述文件列表中，cmdline 文件包含进程的完整命令行信息（除非进程已经交换到 swap 区或处于 zombie 状态， cmdline 文件为空），如命令的名字、选项和参数等，示例如下。

```
$ cat /proc/1/cmdline
/sbin/init$
```

注意，在/proc 目录下的文件中，一个文件可能包含多个环境变量（或命令行参数），每个变量只占一个字段，采用内部格式表示变量的字符串值，以字符“\0”作为字段分隔符或结尾。因此，为了使文件中的字符串数据可读性更强，可以使用“od -c”或“tr "\000" "\n”命令形式显示相应的文件。此外，也可使用“echo 'cat <file>'”的命令形式显示文件的内容。

cwd 是一个符号链接文件，指向进程的当前工作目录。例如，为了找出进程 1 的工作目录，可以使用下列命令（注意，不能使用 sudo 直接运行 cd 命令，故运行下列 cd 命令时需首先运行 su 命令，进入超级用户的 Shell 环境。此外，采用 Shell 内部命令 pwd 会产生不正确的结果，故这里采用外部的 pwd 命令）。

```
$ su
Password:
# cd /proc/1/cwd; /bin/pwd
/
#
```

environ 文件包含进程运行环境的变量设置。由于其中的每一个字段中间均以“\0”字符作为分隔符，且最后仍以“\0”字符结束，因此，为了查询进程 1 的运行环境，可以使用下列命令。

```
$ sudo cat /proc/1/environ | tr "\0" "\n"
[sudo] password for gqxing:
ROOTFSTYPE=
HOME=/
DPKG_ARCH=i386
init=/sbin/init
ROOTFLAGS=
locale=zh_CN
.....
PATH=/sbin:/usr/sbin:/bin:/usr/bin
.....
PWD=/
readonly=y
rootmnt=/root
ROOT=/dev/disk/by-uuid/4c5ea7fa-d041-4128-a78a-43e171e8df76
BOOT=local
$
```

exe 是一个符号链接文件，其中包含进程命令文件的实际路径名，引用这个文件相当于执行相应的命令。对于进程 1 而言，则相当于执行/sbin/init 命令。

fdinfo 是从 Linux 内核 2.6.22 开始新增的一个目录，其中包含进程打开的每一个文件。文件以文件描述符命名，其中 0 表示标准输入，1 表示标准输出，2 表示标准错误输出。每个文件通常包含两个字段：pos 字段是一个十进制的数字，表示文件读写的偏移值；flags 字段是一个八进制的数字，表示文件的访问权限和文件状态（参见 open (2) 系统调用）。这个文件仅限于进程属主才能读取其中的内容。示例如下。

```
$ sudo ls -l /proc/1/fd
总用量 0
-r----- 1 root root 0 2008-12-28 15:59 0
-r----- 1 root root 0 2008-12-28 15:59 1
-r----- 1 root root 0 2008-12-28 15:59 2
-r----- 1 root root 0 2008-12-28 15:59 3
```



```
-r----- 1 root root 0 2008-12-28 15:59 4
-r----- 1 root root 0 2008-12-28 15:59 5
-r----- 1 root root 0 2008-12-28 15:59 6
$ sudo cat /proc/1/fdinfo/4
pos: 0
flags: 04001
$ sudo cat /proc/1/fdinfo/6
pos: 0
flags: 04000
$
```

`limits` 文件包含进程的各种资源的软性限制、硬性限制及其度量单位等，如 CPU 时间限制、创建文件的容量限制、创建进程的数量限制及打开文件的数量限制等。这个文件仅限于进程属主才能读取其中的内容。示例如下。

```
$ sudo cat /proc/1/limits
Limit           Soft Limit      Hard Limit      Units
Max cpu time   unlimited      unlimited      ms
Max file size  unlimited      unlimited      bytes
Max data size  unlimited      unlimited      bytes
Max stack size 8388608       unlimited      bytes
Max core file size 0            unlimited      bytes
Max resident set unlimited      unlimited      bytes
Max processes   3072          3072          processes
Max open files  1024          1024          files
Max locked memory 32768        32768         bytes
Max address space unlimited      unlimited      bytes
Max file locks  unlimited      unlimited      locks
Max pending signals 3072        3072          signals
Max msgqueue size 819200       819200        bytes
Max nice priority 0            0              bytes
Max realtime priority 0          0              bytes
Max realtime timeout    unlimited      unlimited      us
$
```

`maps` 文件包含当前映射的内存区及其访问权限，示例如下。

```
$ sudo cat /proc/1/maps
b7de8000-b7de9000 rw-p b7de8000 00:00 0
b7de9000-b7f41000 r-xp 00000000 08:07 261942 /lib/tls/i686/cmov/libc-2.8.90.so
b7f41000-b7f43000 r--p 00158000 08:07 261942 /lib/tls/i686/cmov/libc-2.8.90.so
b7f43000-b7f44000 rw-p 0015a000 08:07 261942 /lib/tls/i686/cmov/libc-2.8.90.so
b7f44000-b7f48000 rw-p b7f44000 00:00 0
b7f56000-b7f57000 rw-p b7f56000 00:00 0
b7f57000-b7f71000 r-xp 00000000 08:07 244339 /lib/ld-2.8.90.so
b7f71000-b7f72000 r-xp b7f71000 00:00 0 [vdsol]
b7f72000-b7f73000 r--p 0001a000 08:07 244339 /lib/ld-2.8.90.so
b7f73000-b7f74000 rw-p 0001b000 08:07 244339 /lib/ld-2.8.90.so
b7f74000-b7f8d000 r-xp 00000000 08:07 179219 /sbin/init
b7f8d000-b7f8e000 r--p 00018000 08:07 179219 /sbin/init
b7f8e000-b7f8f000 rw-p 00019000 08:07 179219 /sbin/init
b935e000-b9386000 rw-p b935e000 00:00 0 [heap]
bf977000-bf98c000 rw-p bffeb000 00:00 0 [stack]
$
```

其中，第 1 个字段是进程占用的地址空间，第 2 个字段是由 r（读）、w（写）、x（执行）、s（共享）和 p（专用）等字符表示的访问权限，第 3 个字段是相对于文件起始位置的读写偏移值，第 4 个字段是以主次设备号“`major:minor`”形式表示的设备，第 5 个字段是文件所在文件系统中的信息节点号（0 表示不存在与内存区相关联的信息节点，如 `bss`），第 6 个字段表示进程的路径文件名。

`mem` 文件反映的是进程的内存页面，可以利用 `open()`、`read()` 和 `fseek()` 等系统调用访问 `mem` 文件，进而访问进程的内存页面。

`root` 是一个符号链接文件，指向进程的虚拟根目录。虚拟根目录是 Linux 系统的重要功能特性，利用 `chroot` 命令或 `chroot(2)` 系统调用，可以针对每个进程设置文件系统的虚拟根目录。例如，

进程 1 (即 init 进程) 的虚拟根目录仍为实际的根目录, 如下所示。

```
$ sudo ls -l /proc/1/root
lrwxrwxrwx 1 root root 0 2008-07-05 17:04 /proc/1/root -> /
```

smaps 文件反映了每个进程的内存映射的使用情况。其中, 每个进程内存映射的第一行给出的信息与 maps 文件的内容完全相同。其他行分别给出了进程内存映射的大小、进程内存映射中当前仍驻留在内存部分的大小、共享页面中已处理和尚未写入磁盘的内存大小, 以及专用页面中已处理和尚未写入磁盘的内存大小。示例如下。

```
$ sudo cat /proc/1/smaps
...
b7f74000-b7f8d000 r-xp 00000000 08:07 179219      /sbin/init
Size:          100 kB
Rss:           56 kB
Pss:           56 kB
Shared_Clean:   0 kB
Shared_Dirty:   0 kB
Private_Clean: 56 kB
Private_Dirty:  0 kB
Referenced:    40 kB
Swap:          0 kB
...
$
```

stat 文件包含进程的运行状态信息。实际上, ps 命令也是利用这个文件输出进程状态信息的。文件内容的第一个字段表示进程的 ID, 第 2 个字段 (位于圆括号中) 给出的是进程的文件名, 第 3 个字段使用一个字符表示进程的当前状态, 其中包括 R (运行)、S (休眠)、D (等待 I/O)、Z (僵尸)、T (跟踪或停止) 和 W (页面调度) 等。第 4 个字段是父进程的进程 ID, 等等 (参见 ps 命令的输出结果)。

```
$ cat /proc/1/stat
1 (init) S 0 1 0 -1 4194560 2377 754919 39 917 2 300 14742 4053 20 0 1 0 105 1
925120 175 4294967295 3086434304 3086532700 3214456160 3214455296 3086423088 0 0
4096 671835171 3223061765 0 0 0 0 0 0 0 0 0
$
```

statm 文件以页面为单位, 提供内存的状态信息——从左边第一个字段开始, 其输出信息分别为: 整个程序的大小、程序内存驻留部分的大小、共享页面数量、代码部分的大小、库函数部分的大小、数据或栈段部分的大小, 以及尚未写入磁盘的数据页面数量。示例如下。

```
$ cat /proc/1/statm
470 175 121 25 0 68 0
$
```

status 文件提供的信息主要来自 /proc/pid/stat 和 /proc/pid/statm 文件, 只是以更容易阅读和理解的形式给出, 其中包括进程的名字、状态 (如 R 表示运行和 S 表示休眠等)、PID、有效用户 ID、用户组 ID、当前分配的文件描述符数量 (FDSize), 以及各种内存使用情况的详细数据, 如虚拟内存大小 (VmSize)、内存驻留部分的大小 (VmRSS)、共享库代码部分的大小 (VmLib)、代码段 (VmExe)、数据段 (VmData) 及栈段 (VmStk) 的大小等。示例如下。

```
$ cat /proc/1/status
Name:  init
State: S (sleeping)
Tgid:  1
Pid:   1
PPid:  0
TracerPid: 0
Uid:   0     0     0     0
Gid:   0     0     0     0
FDSize: 32
Groups:
VmPeak: 3056 kB
```



```
VmSize:      1880 kB  
VmLck:       0 kB  
VmHWM:      1884 kB  
VmRSS:       700 kB  
VmData:      188 kB  
VmStk:        84 kB  
VmExe:       100 kB  
VmLib:      1480 kB  
VmPTE:       12 kB  
....  
$
```

task 是一个目录。对于一个多线程的进程，其中可能包含若干子目录，每个子目录对应进程的一个线程。在每一个子目录中，存在一组与进程目录中同名的文件，且其意义也完全相同。

## 10.2 系统配置信息

通过查询 proc 文件系统，可以了解当前系统硬件设备的配置信息，如 CPU、内存、系统总线、I/O 设备、IDE 和 SCSI 磁盘设备、磁盘分区、硬件中断、I/O 端口的内存映射以及硬件设备的主次设备号等。

### 1. /proc/cpuinfo 文件

/proc/cpuinfo 文件含有 CPU 及其他系统结构的信息，包括 CPU 的数量、型号、缓存以及时钟频率等，示例如下。

```
$ cat /proc/cpuinfo  
processor       : 0  
vendor_id      : GenuineIntel  
cpu family    : 6  
model          : 8  
model name     : Pentium III (Coppermine)  
stepping        : 10  
cpu MHz        : 696.968  
cache size     : 256 KB  
fdiv_bug       : no  
htt_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 pae mce cx8 sep mtrr pge mca cmov pat pse36 mmx ...  
....  
$
```

其中的内容依次说明如下。

- processor——提供系统配备的每个 CPU 的标识号。在只有一个 CPU 的系统中，其标识号为 0。
- cpu family——CPU 系列的标识号。对于 Intel x86 系列的计算机系统而言，把这个数字加到“86”前面，即可确定 CPU 的系列类型。在老式的 586、486 甚至 386 系统中，这个数值是特别有用的。有些软件也是针对特定的 CPU 结构编译的，因此，这个数值有助于用户正确选择安装的软件包。
- model name——CPU 的通用名字。
- cpu MHz——CPU 的精确速度 (MHz)，精确到小数点后 3 位。
- cache size——CPU 二级缓存的大小。

- ❑ flags —— 说明 CPU 的品质, 如是否包括浮点运算单元(FPU), 以及是否具有处理 MMX 指令的能力, 等等。

## 2. /proc/filesystems 文件

/proc/filesystems 文件包含系统内核当前支持的文件系统列表, 表示相应的文件系统支持模块已被编译进系统内核。在安装文件系统时, 如果未指定文件系统的类型, mount 命令将会依次使用这个文件中的信息确定文件系统的类型, 尝试安装文件系统, 示例如下。

```
$ cat /proc/filesystems
nodev sysfs
nodev rootfs
nodev bdev
nodev proc
...
nodev sockfs
nodev pipefs
nodev anon_inodefs
nodev tmpfs
...
nodev ramfs
...
nodev usbfs
ext3
...
```

在上述输出信息中, 第一列表示文件系统是否位于块设备中。如果第一列为 nodev, 意味着相应的文件系统并非基于磁盘设备的文件系统。第二列给出的是 Linux 系统支持的文件系统的名字。

/proc/iomem 文件包含 I/O 内存映射, 示例如下。

```
$ cat /proc/iomem
00000000-0009f7ff : System RAM
0009f800-0009ffff : reserved
000a0000-000bffff : Video RAM area
000c0000-000cbfff : Video ROM
000dc000-000fffff : reserved
000e0000-000fffff : Extension ROM
000f0000-000fffff : System ROM
00100000-17fdffff : System RAM
00100000-00383359 : Kernel code
0038335a-004a567f : Kernel data
00515000-005c0a1f : Kernel bss
...
```

## 3. /proc/partitions 文件

/proc/partitions 文件包含磁盘与磁盘分区的主次设备号、容量(数据块数量)及设备文件名, 其内容类似于 “fdisk -l” 命令的输出结果, 示例如下。

```
$ cat /proc/partitions
major minor #blocks name

8      0    29302560  sda
8      1     6252088  sda1
8      2          1  sda2
8      3     816480   sda3
8      5     204088   sda5
8      6    11786008  sda6
8      7     9495297  sda7
8      8     748408   sda8
$ sudo fdisk -l /dev/sda

Disk /dev/sda: 30.0 GB, 30005821440 bytes
```



```
240 heads, 63 sectors/track, 3876 cylinders
Units = cylinders of 15120 * 512 = 7741440 bytes
Disk identifier: 0xcccdcccd

      Device Boot   Start     End   Blocks Id System
/dev/sda1 *        1      827   6252088+  7  HPFS/NTFS
/dev/sda2          828    3768  22233960   f  W95 Ext'd (LBA)
/dev/sda3         3769    3876   816480  1c  Hidden W95 FAT32 (LBA)
/dev/sda5         2183    2209  204088+  83  Linux
/dev/sda6         2210    3768  11786008+  8e  Linux LVM
/dev/sda7         828    2083   9495297  83  Linux
/dev/sda8         2084    2182   748408+  82  Linux swap / Solaris

Partition table entries are not in disk order
$
```

在上述 partitions 文件的输出信息中，每个字段的意义简述如下：

- major——磁盘及磁盘分区的主设备号；
- minor——磁盘及磁盘分区的次设备号；
- #blocks——磁盘及磁盘分区中包含的物理磁盘数据块数量；
- name——磁盘及磁盘分区的设备名。

#### 4. /proc/modules 文件

/proc/modules 文件包含已经加载到系统内核的所有模块列表，其内容等同于 lsmod 命令的输出结果。下面是 modules 文件中的部分内容，以及 lsmod 命令的部分输出结果。

```
$ cat /proc/modules
af_packet 23812 2 - Live 0xd8c3b000
nfssd 228848 13 - Live 0xd8cf3000
auth_rpcgss 43424 1 nfssd, Live 0xd8c51000
exportfs 6016 1 nfssd, Live 0xd8bf6000
uinput 10240 1 - Live 0xd8c2f000
thinkpad_acpi 51836 0 - Live 0xd8c43000
.....
$ lsmod
Module           Size  Used by
af_packet        23812   2
nfssd           228848  13
auth_rpcgss     43424   1 nfssd
exportfs        6016   1 nfssd
uinput          10240   1
thinkpad_acpi   51836   0
.....
$
```

在上述输出信息中，第 1 列是模块的名字，第二列是模块占用的内存字节数，第 3 列表示当前已经加载了多少个模块的实例（0 表示模块尚未加载）。第 4 列表明相应的模块是否依赖于其他模块的存在，否则无法正常运行；如是，则列出依赖的模块。第 5 列是模块的加载状态，如 Live、Loading 或 Unloading。第 6 列是加载的模块相对于当前系统内核起始内存地址的偏移值（这个信息可用于调试模块）。

#### 5. /proc/mounts 文件

/proc/mounts 文件能够提供系统当前安装的所有文件系统信息，其输出数据类似于/etc/mtab 文件的内容，只是其数据总是保持最新状态（实际上，/proc 目录中的所有文件总是保持最新状态）。这个文件的数据格式完全遵循 fstab (5) 文件的格式规定，示例如下。

```
$ cat /proc/mounts
rootfs / rootfs rw 0 0
none /sys sysfs rw,nosuid,nodev,noexec 0 0
none /proc proc rw,nosuid,nodev,noexec 0 0
udev /dev tmpfs rw,mode=755 0 0
```

\$

在上述输出信息中，第一列是安装的设备名，第二列是安装点，第 3 列是文件系统的类型，第 4 列表示安装的方式，如只读 (ro) 或读写 (rw) 等。第 5 和第 6 列并没有实际的意义，只是为了保持 /etc/mtab 文件格式的一致而已。

#### 6. /proc/version 文件

/proc/version 文件含有当前运行的 Linux 系统内核（包括 gcc）的版本信息，其中也提供了操作系统类型（参见 /proc/sys/kernel/ostype 文件）及版本（参见 /proc/sys/kernel/osrelease 及 /proc/sys/kernel/version 文件）等信息，示例如下。

```
$ cat /proc/version
Linux version 2.6.27-9-generic (buildd@rothera) (gcc version 4.3.2 (Ubuntu
4.3.2-1ubuntu1)) #1 SMP Thu Nov 20 21:57:00 UTC 2008
$ sudo cat /proc/sys/kernel/ostype
Linux
$ sudo cat /proc/sys/kernel/osrelease
2.6.27-9-generic
$ sudo cat /proc/sys/kernel/version
#1 SMP Thu Nov 20 21:57:00 UTC 2008
$
```

## 10.3 系统运行状态信息

#### 1. /proc/cmdline 文件

/proc/cmdline 文件含有启动系统时传递给 Linux 内核的引导参数，示例如下。

```
$ cat /proc/cmdline
root=UUID=4c5ea7fa-d041-4128-a78a-43e171e8df76 ro locale=zh_CN quiet splash
$
```

上述输出信息表示系统内核位于 UUID 表示的磁盘分区（即 /dev/sda7，参见 /etc/fstab 文件）中，ro 表明以只读方式安装系统内核（详见第 14 章）。

#### 2. /proc/kcore 文件

/proc/kcore 文件是系统物理内存的映像，采用 ELF 文件格式存储。利用这个虚拟文件与完整的二进制内核文件（/usr/src/linux/vmlinux），可以使用 gdb 等工具考察任何内核数据结构的当前状态。文件的大小等于物理内存的大小再加上 4KB，以字节为单位。

#### 3. /proc/kmsg 文件

/proc/kmsg 文件用于存储系统内核生成的各种信息。注意，只有超级用户才能读取这个文件，且同时只能有一个进程读取这个文件。因此，如果采用 syslog (2) 系统调用读取并记录系统内核信息的 syslog 进程正在运行，则不应当读取这个文件，否则将处于等待状态，因而读不到任何数据。通常应使用 dmesg 命令读取这个文件中的系统信息。

#### 4. /proc/loadavg 文件

/proc/loadavg 文件的前 3 个字段是最近 1 分钟、5 分钟及 15 分钟之内系统负载的平均值。所谓系统负载指的是位于运行队列（状态为 R）或等待 I/O 完成（状态为 D）的进程数量。这 3 个字段值与 uptime 命令输出结果中的最后 3 个系统负载平均值是相同的。第 4 个字段包括一对数值，中间增加一个斜线 “/” 分隔字符，其中的第一个数值是系统中当前调度执行的进程或线程的数量，



这个数值应小于或等于系统配置的 CPU 数量；第二个数值是系统当前调度运行的进程或线程的总数。第 5 个字段是系统中最近一次创建的进程 ID。示例如下。

```
$ cat /proc/loadavg  
0.33 0.26 0.10 2/292 7005  
$ uptime  
15:08:54 up 4:27, 3 users, load average: 0.33, 0.26, 0.10  
$
```

### 5. /proc/meminfo 文件

/proc/meminfo 文件提供了系统中整个内存、空闲内存和已用内存（包括实际内存和虚拟内存）的数量，以及系统内核使用的共享内存和缓冲区的数量。事实上，这个文件的内容与 free 命令的输出结果完全一致，不过 free 命令给出的只是其中的一部分，示例如下。

```
$ cat /proc/meminfo  
MemTotal:      384356 kB  
MemFree:       90128 kB  
Buffers:        10844 kB  
Cached:         89328 kB  
SwapCached:    22500 kB  
Active:         198056 kB  
Inactive:       42592 kB  
HighTotal:      0 kB  
HighFree:       0 kB  
LowTotal:       384356 kB  
LowFree:        90128 kB  
SwapTotal:     748400 kB  
SwapFree:       655392 kB  
Dirty:          0 kB  
Writeback:      0 kB  
AnonPages:     127900 kB  
Mapped:         37544 kB  
Slab:           19832 kB  
....  
VmallocTotal:   634872 kB  
VmallocUsed:   8116 kB  
VmallocChunk:  625652 kB  
....  
$ free  
total        used        free      shared      buffers      cached  
Mem:        3 84356     294228     90128          0      10844      89328  
-/+ buffers/cache: 194056    190300  
Swap:        7 48400     93008      655392
```

由于其中提供了大量有价值的系统内存及其使用情况，这个文件是/proc 目录最常用的文件之一，free、top 和 ps 等命令的许多输出信息大都取自这个文件。

- MemTotal —— 系统配备的物理内存的总和（以 KB 为单位）。
- MemFree —— 系统中空闲物理内存的数量（以 KB 为单位）。
- Buffers —— 系统中用作文件缓冲区的物理内存数量（以 KB 为单位）。
- Cached —— 系统中用作缓冲区内存的物理内存数量（以 KB 为单位）。
- SwapCached —— 用作缓冲区内存的交换区的数量（以 KB 为单位）。
- Active —— 当前正在使用的缓冲区或页面内存（即最近一直在用，通常无法回收利用的内存）的总和（以 KB 为单位）。
- Inactive —— 空闲可用的缓冲区或页面内存（即最近一直没有使用，因而可以回收利用的内存）的总和（以 KB 为单位）。
- HighTotal —— 没有直接映射为系统内核空间的内存总和（以 KB 为单位）。

- HighFree ——非系统内核映射空间中的空闲内存数量（以 KB 为单位）。
- LowTotal ——直接映射为系统内核空间的内存总和（以 KB 为单位）。
- LowFree ——系统内核映射空间中的空闲内存数量（以 KB 为单位）。
- SwapTotal ——可用交换区的容量总和（以 KB 为单位）。
- SwapFree ——空闲交换区的容量总和（以 KB 为单位）。
- Dirty ——其中的数据正等待写入磁盘的内存总和（以 KB 为单位）。
- Writeback ——其中的数据正在写入磁盘的内存总和（以 KB 为单位）。
- Mapped ——使用 mmap 命令映射设备、文件或库函数后占用的内存总和（以 KB 为单位）。
- Slab ——系统内核使用的数据结构缓冲区占用的内存总和（以 KB 为单位）。
- VMallocTotal ——分配作虚拟地址空间的内存总和（以 KB 为单位）。
- VMallocUsed ——已用于虚拟地址空间的内存总和（以 KB 为单位）。
- VMallocChunk ——虚拟地址空间可用的最大连续内存块（以 KB 为单位）。

#### 6. /proc/stat 文件

/proc/stat 文件提供了自最近一次系统启动以来，系统内核的各种统计信息。其输出内容也可从 vmstat 和 iostat 等命令中获得，只不过 stat 文件给出的是时间的分类统计，而 vmstat 和 iostat 命令给出的是每个统计数据占整个时间的百分比，示例如下。

```
$ cat /proc/stat
cpu0 68638 153426 64333 1407893 18931 507 304 0 0
intr 1912352 1610700 477 0 3 5 2 0 8251 2 29300 3 48235 149 0 37145 178074 0 0 0
...
ctxt 4000827
btime 1230518487
processes 7219
procs_running 2
procs_blocked 0
$ iostat
Linux 2.6.27-9-generic (iscas) 12/29/08      _i686_
avg-cpu: %user  %nice  %system  %iowait  %steal  %idle
          4.00    8.95    3.80    1.10    0.00   82.14
Device:     tps  Blk_read/s  Blk_wrtn/s  Blk_read  Blk_wrtn
sda          2.17      53.24       26.59    912754    455836
...
sda7         1.90      47.93       13.41    821700    229808
sda8         0.25      5.07        13.18    86842     226016
$ vmstat
procs -----memory----- swap-- io---- system-- cpu-----
r b swpd free  buff cache si so bi bo in cs us sy id wa
1 0 92888 60620 13832 116588 3 7 27 13 112 233 13 4 82 1
$
```

在上述文件的输出信息中，每个字段的意义简述如下。

- cpu ——7 个数据分别表示 CPU 处于用户模式、低优先级用户模式（nice）、系统模式、空闲状态、等待 I/O 完成、处理硬件中断，以及处理软件中断的时间度量（对于 Intel x86 系列的计算机而言，时间计数单位为 1/100 秒）。其中第一行是所有 CPU 的累加值，随后依次列出的是针对每个 CPU 的统计信息。上述例子表示当前系统仅配有一个 CPU。
- intr ——自系统启动以来处理的中断数量。其中第一列是处理的中断总数，随后的数字是针对每一种中断的分类统计。



- ❑ ctxt ——系统经历的 CPU 上下文切换的次数。
- ❑ btime ——系统启动的时间，其数值是从 1970 年 1 月 1 日算起的时间累加值（秒数）。
- ❑ processes ——自系统启动以来创建的进程总和。
- ❑ procs\_running ——处于可运行状态的进程数量。
- ❑ procs\_blocked ——因等待 I/O 完成而处于等待状态的进程数量。

注意，运行 iostat 命令之前需要安装 sysstat 软件包。

### 7. /proc/swaps 文件

swaps 文件包含交换区存储空间的配置及其使用情况。其中包括每个交换区的文件名、交换区存储空间的类型、交换区的总容量（KB）、当前占用的交换区容量（KB），以及交换区的优先级。优先级越低，交换区的选用越优先。如果系统中仅配备一个交换区，其输出信息示例如下。

```
$ cat /proc/swaps
Filename           Type      Size     Used     Priority
/dev/sda8          partition 748400   92888    -1
```

### 8. /proc/uptime 文件

/proc/uptime 文件包含两个数值：系统自最近一次启动迄今为止的运行时间（秒）和系统的空闲时间（秒）。uptime 文件第一个字段的值等同于 uptime 命令输出的第 3 个字段的值，只不过两者的时间单位及表示形式不同。示例如下。

```
$ cat /proc/uptime
17560.29 14692.03
$ uptime
15:34:09 up 4:52, 3 users, load average: 0.00, 0.03, 0.12
$
```

### 9. /proc/vmstat 文件

/proc/vmstat 文件包含各种虚拟内存的统计信息，具体内容请参见 vmstat 命令。

## 10.4 系统可调参数

/proc/sys 是一个非常重要的目录，其中含有大量的文件，分别位于 fs、net 和 kernel 等子目录中，每个文件对应于系统内核的一个或多个可调参数变量。/proc/sys 包含下列子目录。

```
$ ls -l /proc/sys
总用量 0
dr-xr-xr-x 0 root root 0 2008-12-29 12:41 debug
dr-xr-xr-x 0 root root 0 2008-12-29 12:41 dev
dr-xr-xr-x 0 root root 0 2008-12-29 10:42 fs
dr-xr-xr-x 0 root root 0 2008-12-29 11:17 kernel
dr-xr-xr-x 0 root root 0 2008-12-29 12:41 net
dr-xr-xr-x 0 root root 0 2008-12-29 12:41 sunrpc
dr-xr-xr-x 0 root root 0 2008-12-29 12:41 vm
$
```

利用 kernel 和 net 等目录中的文件，可以查询系统内核的可调参数值。作为超级用户，可以利用 sysctl 命令或 sysctl() 系统调用，修改系统内核的可调参数，也可以使用 echo 等命令把新的参数值直接写到相应的参数变量文件中（注意，不能使用 sudo 运行下列 echo 命令，此处需首先运行 su 命令，进入超级用户的 Shell 环境，下同）。示例如下。

```
$ cat /proc/sys/net/ipv4/ip_forward
0
$ su
Password:
# echo 1 > /proc/sys/net/ipv4/ip_forward
# cat /proc/sys/net/ipv4/ip_forward
1
#
#
```

但需要记住的是，这种内核参数的修改方式是临时的，仅对当前运行的系统有效，一旦重新启动，系统将会重新恢复到默认的内核参数设置。要使定制的系统参数总是保持有效，详见 10.4.3 小节的介绍。

### 10.4.1 文件系统可调参数

/proc/sys/fs 目录主要包含与文件系统有关的系统内核参数，其中的 file-max 文件定义了系统范围内所有进程能够打开的文件数量，其默认值如下。

```
$ cat /proc/sys/fs/file-max
36659
$
```

如果在运行过程中经常出现文件描述符不足等错误信息，可以采用下列命令增加打开文件的数量限制（也可以使用 setrlimit(2) 系统调用设置每个进程能够打开的文件数量限制）。

```
# echo 100000 > /proc/sys/fs/file-max
#
#
```

注意，设置 file-max 文件中的参数值时，不能超过内核变量 NR\_OPEN 规定的上限，参见 /usr/include/linux/fs.h 文件。

/proc/sys/fs/file-nr 是一个只读文件，其中给出了 3 个重要的参数值：已分配的文件描述符的数量、空闲文件描述符的数量以及文件描述符的最大数量。Linux 系统内核采用动态的方式分配文件描述符，而且不会释放，因而无法回收再使用。如果已分配的文件描述符的数量接近于最大值，应当考虑增加最大值，示例如下。

```
$ cat /proc/sys/fs/file-nr
5760    0    36659
$
```

/proc/sys/fs/inode-state 文件含有 7 个数值：nr\_inodes、nr\_free\_inodes、preshrink 和 4 个空值（/proc/sys/fs/inode-nr 文件只包含前两个参数值）。nr\_inodes 是系统已经分配的信息节点的数量（这个参数可能稍微大于 inode-max，因为 Linux 系统分配信息节点采用的是一次分配一个整页面的方式）。nr\_free\_inodes 是空闲信息节点的数量。示例如下。

```
$ cat /proc/sys/fs/inode-state
15132  2118  0  0  0  0  0
$
```

### 10.4.2 系统内核可调参数

/proc/sys/kernel 是一个非常重要的子目录，其中的文件表示系统内核的可调参数。常见或经常需要调整的内核参数包括 msgmax、msgmnb、msgmni、sem、shmll、shmmmax 和 shmmni 等。

#### 1. 消息队列

/proc/sys/kernel/msgmax 文件定义了系统范围内单个 System V 消息的最大字节数。/proc/sys/kernel/msgmni 文件定义了系统范围内消息队列标识符的最大数量限制。/proc/sys/kernel/msgmnb 文件定义了系统范围内可以写入消息队列的最大字节数。下面的例子给出了系统的默认值。

```
$ cat /proc/sys/kernel/msgmax
8192
```



```
$ cat /proc/sys/kernel/msgmni  
749  
$ cat /proc/sys/kernel/msgmnb  
16384  
$
```

/proc/sys/fs/mqueue 目录存在 3 个文件：msg\_max、msgsize\_max 和 queues\_max，用于控制 POSIX 消息队列使用的资源。POSIX 消息队列是一个全新的消息队列，与 System V 的消息队列完全不同，详见 mq\_overview (7) 的说明。下面的例子给出了 3 个消息队列参数的默认值。

```
$ cat /proc/sys/fs/mqueue/msg_max  
10  
$ cat /proc/sys/fs/mqueue/msgsize_max  
8192  
$ cat /proc/sys/fs/mqueue/queues_max  
256  
$
```

## 2. 共享内存

/proc/sys/kernel/shmall 文件包含系统范围内共享内存页面的数量限制，示例如下。

```
$ cat /proc/sys/kernel/shmall  
2097152  
$
```

/proc/sys/kernel/shmmmax 文件可用于查询和设置用户当前能够创建和使用的最大共享内存段的大小限制。Linux 系统内核当前支持的最大共享内存段为 1GB。示例如下。

```
$ cat /proc/sys/kernel/shmmmax  
33554432  
$
```

/proc/sys/kernel/shmmni 文件用于指定系统范围内用户能够创建和使用的共享内存段的最大数量限制。示例如下。

```
$ cat /proc/sys/kernel/shmmni  
4096  
$
```

## 3. 信号灯

/proc/sys/kernel/sem 文件包含 4 个 IPC 信号灯参数值：SEMMSL 表示每个信号灯集合支持的最大信号灯数量，SEMMNS 表示系统范围内所有信号灯集合能够支持的最大信号灯数量限制，SEMOPM 表示每个 semop() 系统调用中能够指定的最大信号灯操作数量，SEMMNI 表示系统范围内信号灯标识符的最大数量限制。示例如下。

```
$ cat /proc/sys/kernel/sem  
250    32000   32      128  
$
```

## 4. 网络可调参数

/proc/sys/net/ipv4 目录存在的虚拟文件涉及各种网络设置，可用于查询和修改 IP、TCP 及 UDP 等网络协议方面的可调参数。其中的许多设置可用于防止系统攻击或设置系统的路由功能。

下面是 /proc/sys/net/ipv4 目录中的部分重要文件。

- ❑ **icmp\_echo\_ignore\_all** —— 表示系统内核是否忽略任何主机（包括自己）的 ICMP ECHO 分组数据。这个参数的默认值为 0，意味着系统应响应 ICMP ECHO 分组数据。1 表示忽略所有的 ICMP ECHO 请求，此时，任何主机都无法使用 ping 命令或其他类似的工具联系当前的主机系统。
- ❑ **icmp\_echo\_ignore\_broadcasts** —— 除了区分是否发至广播或多播地址的 ICMP 分组数据外，其意义同上。变量的默认值为 0。

- **ip\_default\_ttl** —— 用于设置默认的生存时间 (Time To Live, TTL) 参数，限制在到达目的主机之前，分组数据能够跳转的网络数量。**ip\_default\_ttl** 的默认值为 64。注意，增加这个参数值可能会降低系统的性能。每当经由一个路由器或网关时，分组数据的 TTL 值将减 1。当 TTL 值减为 0 时仍未到达目的主机，分组数据将被扔掉。示例如下。

```
$ cat /proc/sys/net/ipv4/ip_default_ttl
64
$
```

- **ip\_forward** —— 用于启用或禁用 IP 转发功能，即允许或禁止系统中的网络接口向另外一个网络接口转发分组数据。这个参数的默认值为 0（即禁止转发分组数据）。如果把这个参数设置为 1，表示允许转发网络分组数据。利用这个参数，可以启用或关闭在网络接口之间转发分组数据的功能，允许 Linux 系统作为一个防火墙或路由器。注意，这是一个极其重要的特殊变量，可用于设置 NAT、防火墙、路由和伪装等。一旦改动这个参数值，能够导致其他配置参数的重置，使其恢复到默认状态。示例如下。

```
$ cat /proc/sys/net/ipv4/ip_forward
0
$
```

- **ip\_local\_port\_range** —— 用于指定 TCP 或 UDP 使用的端口范围。第一个数值是可用的最低端口号，第二个数值是可用的最高端口号。对于任何系统，当默认的端口号 1 024～4 999 仍不能满足要求时，可以利用这个参数，在 32 768 至 61 000 之间选定一个端口号范围。示例如下。

```
$ cat /proc/sys/net/ipv4/ip_local_port_range
32768 61000
$
```

- **tcp\_keepalive\_probes** —— 这个参数值告诉系统内核，在确认一个网络连接已经断开之前应发出多少 keepalive 试探分组数据。**tcp\_keepalive\_probes** 的默认值为 9，即在发出 9 个 keepalive 试探分组数据仍未收到回应时，可据此以断定网络连接已经断开。示例如下。

```
$ cat /proc/sys/net/ipv4/tcp_keepalive_probes
9
$
```

- **tcp\_keepalive\_intvl** —— 用于指定链路空闲时仍然保持网络连接存活状态的时间长度（秒）。**tcp\_keepalive\_intvl** 的默认值为 75 秒。这个参数值告诉系统内核，对于发出的每一个 keepalive 试探分组数据，能够容忍的等待回应时间有多长。这个参数值也用于计算网络连接的超时值。按照 **tcp\_keepalive\_probes** 的默认值为 9 计算，默认的等待时间约为 11 分钟（75 秒×9），超过此限，则断开连接。示例如下。

```
$ cat /proc/sys/net/ipv4/tcp_keepalive_intvl
75
$
```

- **tcp\_keepalive\_time** —— 这个参数告诉 TCP/IP 协议栈，如果网络连接一直空闲，应以何种时间频率（以秒为单位）发送 TCP keepalive 分组数据，以便保持网络连接的存活。**tcp\_keepalive\_time** 变量的默认值为 7 200 秒，即 2 小时。如果需要调整，通常不应低于默认值，否则会造成不必要的网络资源消耗。示例如下。

```
$ cat /proc/sys/net/ipv4/tcp_keepalive_time
7200
$
```

## 5. 其他系统可调参数

/proc/sys/kernel/ctrl-alt-del 文件中的参数值用于控制怎样处理控制台键盘的 Ctrl-Alt-Del 组合



键。当 `/proc/sys/kernel/ctrl-alt-del` 文件中的参数值为 0 时，系统会随时检测 Ctrl-Alt-Del 组合键，一旦按下此组合键，将会把这一信号发送给 init 进程处理。init 进程的默认处理方式是从容地重新启动系统。当这个参数值大于 0 时，将会立即重新启动系统，示例如下。

```
$ cat /proc/sys/kernel/ctrl-alt-del  
0  
$
```

`/proc/sys/kernel/domainname` 和 `/proc/sys/kernel/hostname` 文件用于查询和设置当前系统的域名和主机名，其作用相当于执行 `domainname` 和 `hostname` 命令。例如，执行下面两个命令：

```
# echo "iscas" > /proc/sys/kernel/hostname  
# echo "mydomain" > /proc/sys/kernel/domainname
```

相当于执行下列两个命令：

```
# hostname "iscas"  
# domainname "mydomain"
```

`/proc/sys/kernel/panic` 含有 `panic_timeout` 内核参数值。如果这个数值为 0，在遇到致命的故障时，系统将会处于暂停状态；如果这个参数值为非 0 值，系统将会在遇到故障时暂停参数值指定的时间（秒），然后重新引导系统。默认的参数值如下。

```
$ cat /proc/sys/kernel/panic  
0  
$
```

`/proc/sys/kernel/panic_on_oops` 文件用于控制系统遇到故障时的下一步处理动作。如果这个文件的参数值为 0，系统将会尝试继续运行；如果参数值为 1，系统将会在延迟数秒（以便 `klogd` 有时间记录错误信息）之后停机；如果 `/proc/sys/kernel/panic` 文件的参数值也是一个非 0 数值，将会重新启动系统。默认的参数值如下。

```
$ cat /proc/sys/kernel/panic_on_oops  
0  
$
```

`/proc/sys/kernel/pid_max` 文件用于指定进程 ID 循环计数的最大值。在一个 32 位字长的系统中，这个参数的默认值是 32 768，进程的 ID 号不能超过这个参数值指定的范围。当达到这个文件指定的数值时，系统将会从头开始重新分配进程的 ID 号。示例如下。

```
$ cat /proc/sys/kernel/pid_max  
32768  
$
```

`/proc/sys/kernel/printk` 文件包含 4 个数值：`console_loglevel`、`default_message_loglevel`、`minimum_console_level` 和 `default_console_loglevel`。这 4 个参数值用于设定控制台错误信息输出或日志记录的级别。有关日志记录级别的说明详见 `syslog(2)`。如果信息的优先级高于 `console_loglevel` 定义的级别，则相应的信息就能够输出到控制台。没有明显定义优先级的信息将按照 `default_message_level` 的设置处理。`minimum_console_loglevel` 是 `console_loglevel` 能够设置的最小（最高）值。`default_console_loglevel` 是 `console_loglevel` 的默认值。示例如下。

```
$ cat /proc/sys/kernel/printk  
4 4 1 7  
$
```

`/proc/sys/kernel/pty` 目录存在两个文件，其中 `max` 文件定义了系统当前可以支持的最大伪终端设备数量，只读文件 `nr` 表示系统当前正在使用的伪终端设备数量。示例如下。

```
$ cat /proc/sys/kernel/pty/max  
4096  
$ cat /proc/sys/kernel/pty/nr  
2  
$
```

### 10.4.3 sysctl 命令

在查询、修改系统可调参数时，除了能够使用前述的 cat 和 echo 命令之外，比较正规的方法是采用 sysctl 命令和 sysctl.conf 配置文件替代 echo 命令，修改/proc/sys 目录中的任何文件，设置相应的系统参数。例如，可以使用下列 sysctl 命令：

```
$ sudo sysctl -w kernel.hostname=iscas
kernel.hostname = iscas
$
```

代替下列 echo 命令，设置系统的主机名。

```
# echo iscas > /proc/sys/kernel/hostname
#
```

sysctl 命令可用于查询、动态地设置所有的系统内核参数，可以在系统运行期间观察、修改位于/proc/sys 目录中的任何文件。sysctl 命令的语法格式如下。

```
sysctl [-n] [-e] variable
sysctl [-n] [-e] [-q] -w variable=value
sysctl [-n] [-e] [-q] -p file
sysctl [-n] [-e] [-a | -A]
```

其中，“-n”选项表示仅输出系统可调参数的值，“-N”选项表示仅输出系统可调参数的参数名，“-e”选项表示忽略参数名字等出错信息，“-q”选项表示安静方式，禁止输出设置的系统可调参数值，“-a”选项表示输出系统内核的所有参数值，“-A”选项表示以表格形式输出系统内核的所有参数值。

variable 是表示系统可调参数的变量，由/proc/sys 子目录直至文件的完整路径名组成，但需要把路径名中的斜线 “/” 换成句点 “.”（实际上，不替换也可以）。例如，要查询当前的最大共享内存设置，可以使用下列命令。

```
$ sysctl kernel.shmmmax
kernel.shmmmax = 33554432
$
```

要查询当前的 TCP/IP 设置是否支持路由功能，可以使用下列命令。

```
$ sysctl net.ipv4.ip_forward
net.ipv4.ip_forward = 0
$
```

“-w variable=value” 选项用于修改系统内核参数。其中，variable 是系统内核参数的关键字，value 是准备设置的值。如果 value 中包含引号或易被 Shell 解析的元字符，则需要使用双引号。注意，等号 “=” 前后不能有空格。例如，为了设置系统内核参数 kernel.hostname，修改系统的节点名，可以采用下列命令。

```
$ sudo sysctl -w kernel.hostname=iscas
kernel.hostname = iscas
$
```

如果想要了解系统提供哪些变量，能够用于设置系统内核的可调参数，可以访问/proc/sys 目录，也可以使用 “sysctl -a” 或 “sysctl -A” 命令。示例如下。

```
$ sysctl -a
...
kernel.hostname = iscas
kernel.domainname = <none>
kernel.shmmmax = 33554432
kernel.shmall = 2097152
kernel.shmmni = 4096
...
$
```

上述介绍的任何方法都是一种临时性的设置，仅对当前正在运行的系统有效，一旦关机后重



新启动系统，系统将会恢复默认的设置，故仅能用于测试的目的。为了一劳永逸地设定系统的可调参数，可以利用 sysctl 命令的“-p file”选项，借助/etc/sysctl.conf 文件实现。“-p file”选项表示从指定的文件（默认的文件为/etc/sysctl.conf）中读入并设置其中列出的所有系统参数。

每次引导系统时，init 进程都会运行/etc/rcS.d/S17procps 脚本，这个脚本包含一个 sysctl 命令，使用 /etc/sysctl.conf 文件作为参数，设置系统的可调参数。下面是一个取自/etc/init.d/procps 脚本的代码片段。

```
$ cat /etc/init.d/procps
...
case "$1" in
    start|restart|force-reload)
        quiet="-q"
        if [ "$VERBOSE" = "yes" ]; then
            uiert=""
        fi
        for file in /etc/sysctl.conf /etc/sysctl.d/*.conf ; do
            if [ -r "$file" ] ; then
                log_action_begin_msg "Setting kernel variables ($file)"
                sysctl $quiet -p "$file"
                log_action_end_msg $?
            fi
        done
    ;;
...
$
```

因此，加到/etc/sysctl.conf 文件中的任何参数设置将会在系统的启动过程中立即生效。如果需要改变系统内核参数的默认设置，如增加系统的共享内存，使系统支持路由功能等，可以把有关的参数设置加到/etc/sysctl.conf 文件中，以便在系统的启动过程中设定系统的可调参数。下面是系统提供的/etc/sysctl.conf 文件的部分内容。

```
$ cat /etc/sysctl.conf
#
# /etc/sysctl.conf - Configuration file for setting system variables
# See /etc/sysctl.d/ for additional system variables.
# See sysctl.conf (5) for information.
#
#kernel.domainname = example.com
#
# Uncomment the following to stop low-level messages on console
#kernel.printk = 4 4 1 7
#####
# Functions previously found in netbase
#
# Uncomment the next two lines to enable Spoof protection (reverse-path filter)
# Turn on Source Address Verification in all interfaces to
# prevent some spoofing attacks
#net.ipv4.conf.default.rp_filter=1
#net.ipv4.conf.all.rp_filter=1
...
$
```

## 10.5 其他重要的子目录

### 1. /proc/fs 子目录

/proc/fs 目录包含对外公布的共享文件系统，如果系统正在运行 NFS 服务器软件，使用 cat

命令显示 /proc/fs/nfsd/exports 文件，可以看到当前系统提供的共享文件系统，以及赋予文件系统的访问权限。示例如下。

```
$ cat /proc/fs/nfsd/exports
# Version 1.1
# Path Client(Flags) # IPs
/home/gqxing/docs iscas(rw,root_squash,sync,wdelay,no_subtree_check,uuid=7
3e5dfbe:ae7a4aae:9cf6525:cc411ba2)
$
```

## 2. /proc/net 子目录

/proc/net 子目录存在各种网络文件，这些文件给出了每个网络层的状态信息。文件的内容均为普通的 ASCII 数据，因而可以利用 cat 等命令进行查询。实际上，netstat 和 ifconfig 等命令就是利用这些文件作为其信息源，提供各种统计数据的。而且，netstat 等命令提供的数据可读性更强，显示格式更规范，故这里仅以两个文件为例，予以说明。

### (1) /proc/net/arp 文件。

/proc/net/arp 文件包含系统内核实现地址解析的 ARP 表。其中的 ARP 解析项要么是预定义的，要么是动态获取的。在下述输出信息中，有关“HW type”和“Flags”字段的详细说明，可以参考/usr/include/linux/if\_arp.h 文件，也可以比照 netstat 命令的输出信息。

```
$ cat /proc/net/arp
IP address      HW type      Flags        HW address          Mask       Device
169.254.78.100  0x1          0x2          00:17:31:C9:22:92  *          eth0
$
```

### (2) /proc/net/dev 文件。

/proc/net/dev 文件含有网络设备的状态统计信息，包括发送与接收的分组数据统计、错误与碰撞的统计数据以及其他统计数据。dev 文件的内容类似于 ifconfig 命令的输出结果，但 ifconfig 命令的输出信息可读性更强，示例如下。

```
$ cat /proc/net/dev
Inter-| Receive                                | Transmit
      bytes   packets errs drop fifo frame compressed multicast|bytes   packe
ts errs drop fifo colls carrier compressed
    lo: 13052     163   0   0   0     0        0        0    13052     1
63   0   0   0     0        0        0        0
    eth0: 486002   6061   0   0   0     0        0        0    146546    16
86   0   0   0     0        0        0        0
...
$ ifconfig
eth0      Link encap:Ethernet HWaddr 00:40:b8:00:8e:52
          inet addr:169.254.78.100 Bcast:169.254.255.255 Mask:255.255.0.0
          UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
          RX packets:6103 errors:0 dropped:0 overruns:0 frame:0
          TX packets:1703 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:488522 (477.0 KB) TX bytes:149012 (145.5 KB)
          Interrupt:11 Base address:0x2400

lo      Link encap:Local Loopback
          inet addr:127.0.0.1 Mask:255.0.0.0
          UP LOOPBACK RUNNING MTU:16436 Metric:1
          RX packets:163 errors:0 dropped:0 overruns:0 frame:0
          TX packets:163 errors:0 dropped:0 overruns:0 carrier:0
```



```
collisions:0 txqueuelen:0  
RX bytes:13052 (12.7 KB) TX bytes:13052 (12.7 KB)
```

```
$
```

### 3. /proc/sysvipc 子目录

/proc/sysvipc 子目录包含 3 个虚拟文件：msg、sem 和 shm，分别用于查询消息队列、信号灯和共享内存的运行状态信息。这些文件的内容类似于 ipcs 命令的输出结果，示例如下。

```
$ cat /proc/sysvipc/shm  
key      shmid perms      size cpid lpid nattch   uid   gid   cuid   cgid  
atime    dtime   ctime  
0        32768 1600    393216 5856 5438       2 1000 1000 1000 1000  
1230518826          0 1230518826  
0        65537 1600    196608 5808 5438       2 1000 1000 1000 1000  
1230518828          0 1230518828  
....
```

## 第11章

### 磁盘空间管理

本章主要介绍磁盘空间管理，说明如何查询磁盘空间的使用与空闲情况，怎样找出超大容量的文件，找出长期闲置不用的文件，定期地清除磁盘中的垃圾文件，利用 Linux 系统提供的各种工具备份或恢复文件，配置文件系统磁盘空间和信息节点的使用限额，等等。其内容主要包括：

- 查询磁盘空间信息；
- 采用标准工具备份与恢复数据；
- 采用专用工具备份与恢复数据；
- 文件系统的限额管理。





## 11.1 查询磁盘空间信息

磁盘空间管理的主要目的是了解磁盘存储空间的使用情况，包括系统当前已经使用的空间、可用的空闲空间、现有的文件数量、空闲的信息节点等；及时清理垃圾文件（如内存影像转储文件），删除长期闲置不用的文件，清除临时目录或文件，以及删除超大容量的文件。在此基础上，可以利用 Linux 系统提供的标准工具，复制、备份或恢复文件甚至整个文件系统，设置磁盘空间的限额，确保磁盘空间资源的合理分配与使用，等等。

当文件系统空间容量的使用接近 90% 时，需要利用 cp 等命令，把其中的文件转存到相对空闲的其他磁盘中，或者利用 tar、cpio、dd 及 dump 等命令把文件转存到磁带上，或者干脆删除其中无需继续保存的文件。

### 11.1.1 常用的磁盘空间管理工具

表 11-1 给出了部分磁盘空间管理的常用工具。

表 11-1

磁盘空间管理常用工具

命 令	简 单 描 述
df	查询文件系统中的可用或已用存储空间及文件信息节点数量
du	查询指定（或当前）目录中每个文件或目录占用的磁盘空间
find -size	检索指定目录中指定大小的文件
ls -s	以 1KB 数据块为单位，显示文件的大小
cpio	用于创建、转存或恢复 cpio 档案文件，实现文件或文件系统的备份与恢复，也可用于实现整体目录层次结构的复制
tar	用于创建、转存或恢复 tar 档案文件，实现文件或文件系统的备份与恢复
dd	用于实现原始数据复制。可以复制文件甚至文件系统（即整个磁盘分区）

### 11.1.2 使用 df 命令检查磁盘空间的使用情况

系统管理员经常需要监控磁盘空间的使用情况。即使系统配置的硬盘比较大，如果分区不当，如“/”文件系统过小，仍然会产生磁盘空间紧张的情况。要监控磁盘空间的使用情况，自然会用到 df 命令。利用 df 命令，可以查询每个文件系统磁盘空间的使用与空闲状况。df 命令的语法格式简写如下。

**df [-ahiklv] [-B size] [-t fstype] [-x fstype] [filesystem]**

表 11-2 给出了 df 命令的部分常用选项及其简单说明。

表 11-2

df 命令的部分常用选项

选项与参数	GNU 选 项	简 单 说 明
-a	--all	显示所有文件系统（包括虚拟文件系统，如/proc）的存储空间及其使用情况
-B size	--block-size=size	以指定的字节数量为单位，显示每个已安装文件系统的磁盘空间使用情况。输出信息包括文件系统的设备文件名、文件系统总容量、已分配的存储空间容量、可用的存储空间容量、已用存储空间占文件系统总容量的百分比，以及文件系统的安装点。其中，size 可以是一个单独的数字，也可以附加一个字符 K、M 或 G 等，如 1K、1M 或 1G，分别表示以 1KB、1MB 或 1GB 为单位的数值

续表

选项与参数	GNU 选项	简 单 说 明
-h	--human-readable	以 KB、MB 或 GB 为单位，显示每个已安装文件系统的存储空间使用情况。输出信息包括文件系统的设备文件名、文件系统总容量、已分配的存储空间容量、可用的存储空间容量、已用存储空间容量占文件系统总容量的百分比，以及文件系统的安装点
-i	--inode	显示文件系统的设备文件名、文件系统的信息节点（文件）总量、空闲信息节点数量、已用信息节点数量、已用信息节点数量占信息节点总量的百分比，以及文件系统的安装点
-k		以 KB 为单位，显示每个文件系统的存储空间使用情况。输出信息包括文件系统的设备文件名、文件系统的总容量、已分配的存储空间容量、可用的存储空间容量、已用存储空间占文件系统总容量的百分比，以及文件系统的安装点。在 Ubuntu Linux 系统中，“-k”是 df 命令的默认选项
-l	--local	显示已安装的本地文件系统的存储空间使用情况，包括可用的存储空间容量，以及可用的文件信息节点数量等
-t ftype	--type=fstype	显示指定文件系统类型的磁盘空间总量与可用容量、信息节点（文件）总数与可用信息节点数量
-T	--print-type	同时输出每个文件系统的类型
-x ftype	--exclude-type=fstype	显示除指定文件系统类型之外的其他文件系统的磁盘空间总量与可用容量、信息节点（文件）总数与可用信息节点数量
filesys		指定文件系统、磁盘分区的设备文件名或文件系统的安装点。通常，df 命令仅显示本地系统已经安装的所有文件系统的存储空间使用信息

如果不加任何选项和参数，df 命令通常会以 KB 为单位，显示系统中所有已经安装的文件系统的存储空间使用情况，包括可用数据块数量。下列 df 命令的输出数据表明，“/”文件系统中尚有 5 584 908 个可用的数据块，即还有约 5.5GB 的可用存储空间。

```
$ df
Filesystem 1K-blocks Used Available Use% Mounted on
/dev/sda7 9346012 3286340 5584908 38% /
tmpfs 192176 0 192176 0% /lib/init/rw
varrun 192176 360 191816 1% /var/run
varlock 192176 0 192176 0% /var/lock
udev 192176 2764 189412 2% /dev
tmpfs 192176 104 192072 1% /dev/shm
lrm 192176 2000 190176 2% /lib/modules/2.6.27-9-generic/volatile
/dev/sda5 197625 19114 168307 11% /fedora
/dev/sdb1 1003216 877040 126176 88% /media/disk
$
```

表 11-3 给出了 df 命令输出信息中各个字段的简单说明。

表 11-3 df 命令输出字段的说明

字 段 名	简 单 说 明
1K-blocks、1M-blocks 或 1G-blocks	文件系统中存储空间的总容量
Used	文件系统中已经占用的存储空间数量
Available 或 Avail	文件系统中可用的空闲存储空间数量
Use%	文件系统中已用存储空间数量占全部数据存储空间总量的百分比
Size	文件系统中全部存储空间的总容量（参见“df -h”命令的输出）
Mounted on	安装点

如果 df 命令的输出信息表明文件系统的使用已经接近或超过整个容量的 90%，则需要清除不



必要的文件，删除临时文件或长期闲置不用的文件，或者使用后面将要介绍的各种系统工具，复制或备份部分文件，以增加可用的存储空间。

在较大的文件系统中，以 KB 为固定度量单位计数存储空间，尚需经过一番换算才能准确地知道输出数值的意义，故不太符合人的直观思维与阅读习惯。为此，可以使用“df -h”命令，以更容易阅读的形式，即以 KB、MB 或 GB 等为计数单位，显示文件系统的相关信息，包括每个文件系统的总容量、当前已经使用了多少存储空间、尚有多少可用的存储空间、实际使用的空间占整个文件系统总容量的百分比以及文件系统的安装点等信息。示例如下。

```
$ df -h
Filesystem  Size  Used   Avail Use% Mounted on
/dev/sda7   9.0G  3.2G   5.4G  38% /
tmpfs       188M   0      188M  0%   /lib/init/rw
varrun      188M  360K   188M  1%   /var/run
varlock     188M   0      188M  0%   /var/lock
udev        188M  2.7M   185M  2%   /dev
tmpfs       188M  104K   188M  1%   /dev/shm
lrm         188M  2.0M   186M  2%   /lib/modules/2.6.27-9-generic/volatile
/dev/sda5   193M   19M    165M  11%  /fedora
/dev/sdb1   980M  857M   124M  88%  /media/disk
$
```

上述 df 命令仅给出了文件系统的存储空间及其使用情况。要获取文件系统的信息节点及其使用情况，可以使用带有“-i”选项的 df 命令，如下所示。

```
$ df -i
Filesystem  Inodes  IUsed   IFree  IUse%  Mounted on
/dev/sda7   594512  129978  464534  22%   /
tmpfs       48044   4      48040   1%    /lib/init/rw
varrun      48044   75     47969   1%    /var/run
varlock     48044   3      48041   1%    /var/lock
udev        48044  5042   43002   11%   /dev
tmpfs       48044   2      48042   1%    /dev/shm
lrm         48044   17     48027   1%    /lib/modules/2.6.27-9-generic/volatile
/dev/sda5   51200   36     51164   1%    /fedora
/dev/sdb1   0       0      0       -      /media/disk
$
```

在上述输出信息中，Inodes 字段表示文件系统的信息节点总量，IUsed 字段表示已用信息节点的数量，IFree 字段表示空闲信息节点的数量，IUse% 字段表示已用信息节点数量占信息节点总量的百分比。

从上述例子中可见，df 命令的输出数据中还包含部分虚拟文件系统信息，如 udev 等。对于了解存储空间的实际使用情况，此类信息是无关紧要的，我们真正关心的是实际的文件系统。因此，为了避免输出其他干扰信息，可以使用“-t”选项，指定文件系统的类型，从而限定 df 命令的输出结果，如下所示。

```
$ df -t ext3
Filesystem      1K-blocks  Used  Available Use% Mounted on
/dev/sda7      9346012   3286340  5584908  38% /
/dev/sda5      197625    19114    168307   11% /fedora
$ df -t vfat
Filesystem      1K-blocks  Used  Available Use% Mounted on
/dev/sdb1      1003216   877040   126176   88% /media/disk
$
```

如果不知道每个已安装文件系统的类型，可以查阅/etc/fstab 文件，或者使用 df 命令的“-T”选项，如下所示。

```
$ df -T
Filesystem  Type      1K-blocks  Used   Available  Use%  Mounted on
/dev/sda7   ext3      9346012   3286336  5584912   38% /
tmpfs      tmpfs      192176    0      192176    0%   /lib/init/rw
varrun     tmpfs      192176   360     191816    1%   /var/run
$
```

```

varlock    tmpfs    192176      0      192176      0% /var/lock
udev       tmpfs    192176    2764    189412      2% /dev
tmpfs      tmpfs    192176     104    192072      1% /dev/shm
lrm        tmpfs    192176   2000    190176      2% /lib/modules/
           tmpfs    192176          2.6.27-9-generic/volatile
/dev/sda5  ext3     197625   19114    168307     11% /fedora
/dev/sdb1  vfat    1003216  877040   126176     88% /media/disk
$ 

```

### 11.1.3 使用 du 命令检查目录占用的存储空间

du 命令用于显示指定目录（或当前目录）中每个子目录或文件占用的磁盘空间数量。如果不加任何选项和参数，df 命令通常会输出以 1KB 数据块为单位的统计数据。du 命令的语法格式简写如下。

**du [-aBchkms] [directory]**

表 11-4 给出了 du 命令的部分常用选项和参数。

表 11-4 du 命令的部分常用选项和参数

选项与参数	GNU 选项	简单说明
-a	--all	以 KB 为单位，列出指定目录或当前目录中的每个文件、每个子目录，以及其中每个文件占用的磁盘空间数量，最终给出整个目录占用的磁盘空间总和。如果指定的参数是一个普通文件，则显示指定文件占用的存储空间
-B size	--block-size=size	以指定的字节数为计数单位，显示指定目录或当前目录，以及其中所有子目录占用的磁盘空间数量。其中，size 可以是一个单独的数字，也可以附加一个 K、M 或 G 等字符，如 1K、1M 或 1G，分别表示以 1KB、1MB 或 1GB 为单位的数值
-b	--bytes	等价于指定了“--block-size=1”选项
-c	--total	以 KB 为单位，显示指定目录或当前目录，以及其中所有子目录占用的磁盘空间容量，最后给出累计占用的磁盘空间总量
-h	--human-readable	以 KB、MB 或 GB 为单位，显示指定目录或当前目录，以及其中所有子目录占用的磁盘空间数量
-k	--kilobytes	以 KB 为单位，显示指定目录或当前目录，以及其中所有子目录占用的磁盘空间数量。在 Ubuntu Linux 系统中，“-k”是 du 命令的默认选项
-m	--megabytes	以 MB 为单位，显示指定目录或当前目录，以及其中所有子目录占用的磁盘空间数量
-s	--summarize	以 KB 为单位，显示指定目录或当前目录（包括其中的所有子目录和文件）占用的磁盘空间总量
directory		指定准备检查的目录。如果同时指定多个目录，目录之间需加空格分隔符

利用 du 命令，当发现较大的目录或文件时，可以视具体情况确定是否保留、删除或异地备份，以节省磁盘的存储空间。下面是一个 du 命令输出的例子。

```

$ du /usr/share/scim
172    /usr/share/scim/icons
11508   /usr/share/scim/tables
4588   /usr/share/scim/pinyin
16272   /usr/share/scim
$ 

```

同样，为了避免按数据块形式列出统计数据，可以使用“-h”选项，以 1KB、1MB 或 1GB 等符合阅读习惯的形式输出每个目录或文件占用的存储空间统计数据。下面的例子说明了怎样利用“-h”选项，以适当的单位输出存储空间的使用情况。

```

$ du -h /usr/share/scim
172K   /usr/share/scim/icons
12M    /usr/share/scim/tables
4.5 M  /usr/share/scim/pinyin
$ 

```



```
16M    /usr/share/scim  
$
```

当输出的文件比较多，且文件的大小参差不齐时，可以采用管道机制，利用 sort 命令进行排序。其中，“-r”选项表示反向排序，按从大到小的顺序列出每一个目录和文件占用的磁盘空间数量，“-n”选项表示按数值而非字符顺序排序。示例如下。

```
$ du -k /usr/share/scim | sort -rn  
16272  /usr/share/scim  
11508  /usr/share/scim/tables  
4588   /usr/share/scim/pinyin  
172    /usr/share/scim/icons  
$
```

如果只想查询某个目录（包括其中的所有文件和子目录）占用的全部存储空间数量，可以使用带有“-s”选项的 du 命令。下面的例子仅统计了 /boot 目录占用的存储空间总和。

```
$ du -s /boot  
24016  /boot  
$
```

### 11.1.4 使用 find 命令找出超过一定容量限制的文件

当存储空间紧张，需要尽快腾出磁盘空间时，应找出超大容量的文件，将其删除或备份到其他存储介质上是一种快速有效的方法。要列出超过指定大小的文件，可以使用 find 命令，其语法格式简写如下。

```
find directory -size +nnn -print
```

其中，directory 是起始检索目录，“-size +nnn”选项表示大于指定数量的数据块（512 个字节）的数量，“-print”选项表示输出 find 命令的检索结果。这个命令将会列出超过指定大小的所有文件。

下面的例子说明了怎样利用 find 命令，从指定的 /usr/share/scim 目录中找出大小超过 2 048 个数据块（1MB）的所有文件。

```
$ find /usr/share/scim -size +2048 -print  
/usr/share/scim/tables/Erbi-QS.bin  
/usr/share/scim/tables/CNS11643.bin  
/usr/share/scim/tables/Wubi.bin  
/usr/share/scim/tables/EZ-Big.bin  
/usr/share/scim/pinyin/phrase_lib  
/usr/share/scim/pinyin/pinyin_phrase_index  
$
```

上述命令的输出结果仅仅给出了文件的名字，这些文件究竟有多大并不知道。同样，为了便于观察，最好能够按文件的大小输出最终的结果。为此，可以使用管道，把 find、ls、gawk 和 sort 等命令组合起来，按从大到小的顺序显示一个文件列表。示例如下。

```
$ cd /usr/share/scim  
$ find . -size +2048 -exec ls -ls {} \+ | gawk '{print $1 "\t" $10}' | sort -rn  
2492  ./pinyin/phrase_lib  
1972  ./table/EZ-Big.bin  
1332  ./pinyin/pinyin_phrase_index  
1140  ./table/CNS11643.bin  
1124  ./table/Erbi-QS.bin  
1060  ./table/Wubi.bin  
$
```

### 11.1.5 使用 find 命令找出并删除长期闲置不用的文件

作为系统管理员，日常系统维护的一个主要工作就是清除系统中长期闲置不用的文件或垃圾文件，清除临时文件或目录。要找出在指定的时间内一直没有被使用过的文件，可以使用下列 find 命令。

```
find directory -type f [-atime +nnn] [-mtime +nnn] -print
```

其中，`directory` 表示准备检索的起始目录，“`-atime +nnn`”选项用于找出指定天数（`nnn`）内没有访问过的文件，“`-mtime +nnn`”选项用于找出指定天数（`nnn`）内没有改动的文件。

注意，采用上述 `find` 命令检索长期闲置的文件，其输出结果可能包含一系列系统文件或用途不明的文件，而这些文件即使长时间未用也不应轻易删除，除非自己有绝对把握。在不确定其用途时，不能轻易地删除系统文件。即使确实没有用处，也不要直接删除，最好能够利用下列 `dpkg` 命令形式，确定文件所属的软件包，然后再利用软件包管理工具删除不必要的软件包。

```
$ dpkg -S filename
```

因此，最保险的方法是先利用 `find` 命令，找出在指定的时间内一直没有被使用过的文件，把检索结果储存到一个临时文件（如 `filelist`）中，确定是否应当删除。如果检索出来的文件均属无用文件，可剔除其中需要保留的文件，再使用下列命令删除检索出来的文件。

```
$ rm 'cat filelist'
```

下面的例子展示了怎样找出`/home/guest` 目录及其子目录中两个月来一直没有被访问过的文件，并把这些文件列表存入`/tmp/filelist` 文件中，经过确认之后，再使用 `rm` 命令予以删除。

```
$ cd /home/guest
$ find . -type f -atime +60 -print > /tmp/filelist
$ cat /tmp/filelist
./doc/TechnicalSpecifications.doc
./doc/TroubleShootingGuide.doc
./Config/NetworkSetup.doc
./Config/PrinterSetup.doc
$ rm 'cat /tmp/filelist'
$
```

使用上述 `find` 命令，如果从根目录“`/`”开始检索，可能会检索出大量的系统文件，而大部分系统文件是不能删除的。为了避免每次都逐一辨别每个文件，可以制作一个系统文件列表文件 `/home/gqxing/filesave`，把需要保留的、不能随意删除的重要文件均写到其中。然后编写一个小的 Shell 脚本，在输出过期的文件之前，先利用 `grep` 过滤程序查询 `filesave` 文件，把符合检索条件且 `filesave` 文件中不存在的文件写入`/tmp/filelist` 文件中（在制作 `filesave` 时，可以使用第一次检索出来的文件列表 `filelist` 作为基础）。示例代码如下。

```
$ cat findfile60
#!/bin/bash
dir=${1-'pwd'}
cd ${dir}
echo "Old files under directory \"${dir}\""
for file in `find . -type f -atime +60 -print 2>/dev/null`
do
    if ! grep 'basename ${file}' /home/gqxing/filesave >/dev/null
    then
        ls "${file}"
    fi
done | tee /tmp/filelist
exit 0
$
```

为简单起见，我们创建了一个测试目录，其中包含 3 个过期的文件，执行上述脚本时，其输出结果如下。

```
$ findfile60
Old files under directory "/home/gqxing/testing"
/home/gqxing/testing/logs
/home/gqxing/testing/note
/home/gqxing/testing/memo
$
```

假定其中的 `logs` 文件需要保留，我们把这个文件名写到`/home/gqxing/filesave` 中（如果输出结果



较多，且需要保留的文件也较多，可以把生成的/tmp/filelist 文件复制到 filesave 中，通过简单的编辑，保留或删除适当的文件）。再运行上述脚本，新的/tmp/filelist 文件将会包含下列两个需要删除的文件。

```
$ findfile60  
Old files under directory "/home/gqxing/testing"  
/home/gqxing/testing/note  
/home/gqxing/testing/memo  
$
```

### 11.1.6 使用 find 命令找出并删除 core 文件

在开发和测试期间，由于程序可能存在这样或那样的问题，开发系统中经常会有大量的内存映像文件，即 core 文件。这样的文件一多，既影响目录文件的清洁性，又会占用大量宝贵的存储空间。为此，作为系统管理员或开发者本人，应当经常清理系统或开发者自己的工作目录中存在的 core 文件。

要找出并删除这种 core 垃圾文件，可以使用下列 find 命令，在指定的目录或当前工作目录中，检索并删除 core 文件。

```
find directory -name core -exec rm {} \;
```

其中，directory 是起始检索目录，“-name core”选项表示检索名为 core 的内存映像文件，“-exec rm {} \;” 选项表示执行 rm 命令，删除检索出来的 core 文件。

下面的例子展示了怎样使用 find 命令找出并删除 hwang 用户主目录中的 core 文件。

```
$ cd /home/hwang  
$ find . -name core -exec rm {} \;  
$
```

### 11.1.7 使用 ls 命令检测文件的大小

前面曾经介绍过，ls 命令的 “-s” 选项也可用于显示以 1KB 为计数单位的文件大小，示例如下。

```
$ cd /home/gqxing/doc  
$ ls -ls  
总用量 2180  
464 -rw-r--r-- 1 gqxing gqxing 471040 2008-07-04 18:10 design.doc  
460 -rw-r--r-- 1 gqxing gqxing 465920 2008-08-16 15:20 project.doc  
316 -rw-r--r-- 1 gqxing gqxing 317440 2008-06-08 09:40 proposal.doc  
416 -rw-r--r-- 1 gqxing gqxing 419840 2008-10-21 16:50 report.doc  
412 -rw-r--r-- 1 gqxing gqxing 414720 2008-06-20 11:48 requirement.doc  
112 -rw-r--r-- 1 gqxing gqxing 110240 2008-09-26 10:42 testing.doc  
$
```

为了更便于观察，还可以组合使用 sort 命令，按照从大到小的顺序列出每个文件，示例如下。

```
$ cd /home/gqxing/doc  
$ ls -s | sort -nr  
464 design.doc  
460 project.doc  
416 report.doc  
412 requirement.doc  
316 proposal.doc  
112 testing.doc  
总用量 2180  
$
```

## 11.2

### 采用标准工具备份与恢复数据

备份与恢复是两个互逆的数据处理过程，如果运用得当，能够防止系统丢失重要的数据。备

份通常指的是从系统磁盘中把系统数据和业务数据等复制到另一个存储介质（通常为移动介质，如磁带、硬盘、软盘或 CD/DVD 等）的过程。恢复是一个逆过程，即从备份介质中把重要文件、文件系统或业务数据复制到系统中。

此外，还要考虑备份什么数据以及怎样备份，这涉及备份的策略，也影响到数据的恢复。在 Linux 系统中，通常可以备份单个文件、目录、文件系统或数据分区，这取决于系统中数据修改的频繁程度、文件的重要性以及其他因素。备份的频率可以采用每天一次、每周一次甚至每月一次等。

从备份的数据考虑，可以采用下列备份方式之一。

- 文件备份——采用这种备份方式，可以备份文件系统中的重要系统文件和目录等。一旦系统文件出现问题，可以直接恢复单个或部分文件。
- 文件系统备份——采用这种备份方式，可以备份从根目录开始的整个文件系统，也可以备份 /var、/home 或用户自建的单个文件系统等。当文件系统出现问题时，可以恢复整个或单个文件系统。
- 数据分区备份——在某些数据库应用系统中，其数据通常采用原始分区而非文件系统存储数据。针对这种情况，可以采用数据分区的备份方式。当需要恢复这样的备份数据时，必须恢复整个磁盘分区。

从备份的方式考虑，可以采用下列备份方式之一。

- 完整备份——这种备份方式主要用于备份文件系统。利用这种备份方式，能够完整地恢复一个文件系统。
- 增量备份——增量备份是在完整备份或其他增量备份的基础上，仅仅备份自上次备份以来发生变动的文件。增量备份用于补充相应时间周期内的完整备份。在一个特定的时间周期内，数据的备份应当包括一个完整备份与 0 或多个增量备份。

在备份与恢复数据文件或文件系统时，可以直接使用 Linux 系统提供的标准工具软件实现，如 cpio、tar 或 dd 等，也可以采用 Linux 系统的专用工具软件，如 dump/restore、amanda、BackupPC、rsync 或 bacula 等。

cpio、tar 和 dd 等命令是最基本的数据备份与恢复工具，不仅适用于 Linux 系统，而且广泛应用于各种 UNIX 系统。要备份单个文件、部分文件或整个文件系统，完全可以使用 Linux 系统的标准工具实现。因此，本节仅讨论 Linux 系统提供的标准工具，下一节再介绍专用的备份与恢复工具。至于备份使用的介质，则不外乎磁带、磁盘或光盘等。

在选择备份工具时，也可以考虑各种应用软件提供的备份工具，如 MySQL 数据库的 mysqldump 或 mysqlhotcopy 等命令。

为了备份一组文件、一个完整的目录或文件系统，如 /home/gqxing/data 目录，通常应使用 du 和 df 等命令，事先确定目录或文件系统的容量，以便能够确定生成的档案文件究竟有多大。如果需要把生成的档案文件写入外部的存储介质中，也可以据此选择存储介质的类型、规格和数量。示例如下。

```
$ cd /home/gqxing/data
$ du -s .
1235224 .
```

du 命令的输出数据表示，上述目录中的文件总共占用了 1 235 224 个数据块 (1KB) 的存储空间。如果想要备份这些文件，至少需要 1.2GB 的存储容量。



### 11.2.1 利用 cpio 实现备份和恢复

cpio 是一个常用的数据备份与恢复工具，能够复制单个文件、一组文件、一个完整的目录结构甚至一个完整的文件系统，也把选定的文件复制为一个档案文件，直接存储于磁带、磁盘或其他存储介质中。例如，cpio 命令能够把一个磁盘分区中的文件系统完整地复制到另一个磁盘分区。

在复制过程中，cpio 命令将会在每个文件之间（之前）插入一个头信息，以便能够恢复任何选定的文件。cpio 命令也能够识别存储介质的结束标志，因而能够自动提示用户更换新的介质，根据备份文件的存储容量要求，把档案文件存储到多个介质上。当需要使用多个介质（如多个磁带）创建档案文件、备份文件或文件系统时，cpio 是一个比 tar 等更灵活、更有效的工具。

cpio 的工作原理是，首先利用 find 等命令生成一个文件列表，然后通过管道提交给 cpio 命令，cpio 再把给定的文件，包括其属性信息，复制到指定的档案文件或存储介质上。

因此，在使用 cpio 命令时，可能需要频繁地使用 find 等命令，为 cpio 命令提供需要复制的文件或文件列表，并通过管道送交 cpio 命令。

cpio 具有如下 3 种运行模式，或者说，具有以下 3 种用途。

- 恢复，利用“-i”选项，把先前创建的档案文件恢复到系统中。
- 备份，利用“-o”选项和 find 命令等工具，从标准输入中读取文件列表，创建一个档案文件，把选定的文件，包括路径名及文件属性等信息，写入其中，以便能够把文件原样恢复到系统中。
- 复制，利用“-p”选项和 find 命令等工具，从标准输入中读取文件列表，把选定目录或文件系统中的文件原样复制到另外一个目录位置或文件系统中，从而创建一个完整的目录或文件系统副本。

cpio 命令的语法格式简写如下。表 11-5 给出了 cpio 命令的常用选项及其简单说明。

```
cpio -i [options] [-C bufsize] [-I [-M message]] [pattern]
cpio -o [options] [-C bufsize] [-O file [-M message]]
cpio -p [options] directory
```

表 11-5 cpio 命令的常用选项

选 项	GNU 选 项	说 明
-i	--extract	输入模式，用于恢复档案文件
-o	--create	输出模式，用于备份档案文件
-p	--pass-throuth	复制模式，用于复制一个完整的目录或文件系统
-a	--reset-access-time	在复制完成之后，复原输入文件的访问时间，使输入文件仍然保持原来的访问时间，就像 cpio 命令根本没有读过这些文件一样
-A	--append	把新复制的文件附加到档案文件的后面。这个选项仅适用于输出模式。在与“-O”或“-F”选项一起使用时，指定的档案文件必须是一个磁盘文件
B		以 5120 个字节作为数据块的读写单位。默认的数据块缓冲区为 512 个字节。注意，输入和输出时最好采用相同的数据块参数
	--block-size=n	以“n × 512”个字节作为数据块的读写单位
-c		以 ASCII 字符格式读写头信息，确保不同 Linux 或 UNIX 系统之间的兼容性
-C size	--io-size=size	以指定的字节数作为数据块的读写单位
-d	--make-directories	在复制过程中，根据需要创建必要的目录。注意，这个选项只能与“-i”或“-p”选项一起使用

续表

选 项	GNU 选项	说 明
-f	--nonmatching	仅复制与指定模式不匹配的文件
-F file	--file=file	使用指定的档案文件代替标准输入或标准输出
-I file		使用指定的档案文件作为输入数据，而不是读取标准输入。如果指定的文件是一个字符特殊设备文件，且当前介质已经完全读入时，可根据提示更换新的介质，然后按下 Enter 键，以便继续读取下一个介质中的数据。注意，这个选项只能与“-i”选项一起使用
-m	--preserve-modification -time	使复制的文件仍然保持先前的修改时间。注意，这个选项在复制目录时无效
-M message	--message=message	定义一个提示信息。当读写达到存储介质结尾，需要更换新的介质时，提示用户更换下一个介质。如果提示信息中包含字符串 "%d"，系统将代之以当前的介质序号（介质从 1 开始编号）
-O file		使用指定的档案文件作为输出文件，而不是写到标准输出。如果指定的文件是一个字符特殊设备文件，且当前介质已经完全写满时，可根据提示更换新的介质，然后按下 Enter 键，以便继续写入下一个介质。注意，这个选项只能与“-o”选项一起使用
-t	--list	读取并显示档案文件中的文件列表
-u	--unconditional	无条件地强行复制。如果不加此选项，cpio 命令的常规处理惯例是禁止同名的老文件替换或覆盖新文件
-v	--verbose	显示方式。输出文件名列表及其相关属性。当与“-t”选项一起使用时，其效果如同“ls -l”命令输出的文件名列表
-V	--dot	特殊显示方式。对于读写的每一个文件，仅输出一个句点“.”记号

假定想要备份/home/gqxing/data 目录中的所有文件，把生成的档案文件写入磁盘中，可以使用下列 cpio 命令。

```
$ cd /home/gqxing/data
$ find . -print | cpio -oc > /backup/data.cpio
1024050 blocks
$
```

或

```
$ find /home/gqxing/data -print | cpio -oc > /backup/data.cpio
1024050 blocks
$
```

在上述例子中，find 命令用于提供 cpio 命令需要复制的文件。由于“find . -print”命令使用当前目录“.”作为起始检索路径，其生成的文件列表为相对路径名，故前者将会采用相对路径名的形式，把所有文件备份到一个指定的 data.cpio 档案文件中；而后者则采用绝对路径名实现档案文件的复制。

采用相对路径复制档案文件的最大好处是，在恢复档案文件时能够重新定位，不必仍然把文件恢复到原来的目录位置，故而能够避免覆盖原有的文件。因此，在使用 cpio 命令复制档案文件时，通常应事先进入指定的目录，然后在当前目录下执行 find 和 cpio 命令，使输出的档案文件不包含绝对路径名。否则，无法在恢复文件时重新定位，因而也就无法改变文件的存储位置，只能按原目录原样恢复。如果使用相对路径名，恢复文件时就会灵活得多，把抽取的文件放在什么位置都可以。

对于采用相对路径名创建的 cpio 档案文件，如果想把整个档案文件全部恢复到一个新的目录



中，可以事先进入目的目录，然后使用下列 cpio 命令。

```
$ cd /newdir  
$ cpio -icdmu < /backup/data.cpio  
1024050 blocks  
$
```

其中，“-i”选项表示以输入模式执行 cpio 命令，从输入的档案文件中抽取文件。“-c”选项表示读取 ASCII 字符格式的文件头信息。“-d”表示必要时可以在复制过程中创建相应的目录。“-m”选项表示复制的文件仍保持原先的访问时间。“-u”选项表示无条件的强行复制文件，即使存在同名的新文件，不管新旧与否，均强行覆盖，否则老的文件无法替代新文件。在原封不动地复制一个目录或文件系统时，这个选项尤其有用，因为同一文件之前可能已经存在于目的目录或文件系统中。

在完成文件的复制之后，cpio 将会输出其读写的数据块（512 字节）数量，表示生成或读取的档案文件的大小。

除了备份与恢复功能之外，利用“-p”选项的复制功能，cpio 命令还能够复制整个目录或文件系统。要复制一个完整的目录或文件系统，把其中的文件全部复制到另外一个目录或磁盘分区（文件系统）中，应首先使用 cd 命令进入源目录或文件系统的根目录，然后再利用 find 和 cpio 命令实现整个目录或文件系统的复制，示例如下。

```
$ cd dir1  
$ find . -depth -print | cpio -pdmu dir2  
1024048 blocks  
$
```

其中，find 命令的“-depth”选项表示逐层深入各级子目录，自底向上依次检索所有的文件，“-print”选项表示输出检索出来的文件名。cpio 命令的“-p”选项表示采用复制模式，复制一个完整的目录。

执行上述命令之后，cpio 将会把 dir1 目录的文件，按照原有的目录层次结构，完全复制到新的目录 dir2 中。进入 dir2 目录之后，即可看到新复制的目录或文件系统与原来完全一样。

下面以磁带（其设备文件名为/dev/st0）为例，说明怎样利用 cpio 和 find 命令备份或恢复文件、目录甚至整个文件系统。实际上，这里介绍的备份与恢复过程同样也适用于其他存储介质，如磁盘分区或 CD/DVD 等，只需把输入或输出文件名改换为相应的设备文件名即可。

### 1. 使用 cpio 命令把文件复制到磁带上

要把某个目录下的所有文件复制到一个磁带上，可以采用 find 等命令提供需要复制的文件列表，然后通过管道提交给 cpio 命令，再由 cpio 把文件复制到磁带上。例如，假定希望备份 /home/gqxing/data 目录中的所有数据文件，可首先进入该目录，然后执行下列 cpio 命令。

```
$ cd /home/gqxing/data  
$ find . -print | cpio -ocvB > /dev/st0  
  
. ./data.0630  
. ./data.0731  
. ./data.0831  
. ./data.0930  
. ./data.1031  
1024050 blocks  
$
```

在上述例子中，find 命令用于提供 cpio 命令需要复制的文件。cpio 命令中的“-o”选项表示以备份方式执行 cpio 命令，创建一个档案文件。“-c”选项表示采用 ASCII 字符格式写入头信息，以便确保新建档案文件的兼容性。“-v”选项表示在屏幕（标准输出）上显示复制的文件名。“-B”选项表示以 5120 个字节的数据块为读写单位。“> /dev/st0”表示指定的输出

文件为磁带机。

执行上述命令之后，当前目录中的所有文件将会被归档并复制到磁带上。如果磁带中有数据，原有的数据将会被覆盖。命令输出的最后一行数据是复制的数据块（512 字节）数量，表示生成的档案文件的大小。在这个例子中，生成的档案文件约为 512MB。

为了验证数据是否确实已经被复制到磁带上，可以使用下列命令。

```
$ cpio -ict < /dev/st0
.
.
.
data.0630
data.0731
data.0831
data.0930
data.1031
1024050 blocks
$
```

其中，“-t”选项意味着仅读取并显示档案文件中的文件列表，“</dev/st0”表示指定的输入文件为磁带机。输出结果表示，读取的文件内容与上述档案文件制作过程的输出结果完全一致。

## 2. 使用 cpio 命令把文件复制到多个磁带上

如前所述，当使用 cpio 命令，把单个文件、一组文件或整个文件系统复制到磁带时，应注意整个数据的容量。如果数据量过大，一个磁带无法容纳所有的数据，因而需要使用多个磁带介质时，应注意使用“-I”选项，以便在磁带复制完成之后，由系统提示用户更换一个新的磁带，从而把档案文件备份到多个磁带上。示例如下。

```
$ cd /home/gqxing/data
$ find . -depth -print | cpio -ocvB -M "The tape has reached end. Please replace it with
#2d. Press Enter when ready." -O /dev/st0
```

当复制到中途时，系统将会输出下列提示信息，提醒用户更换新的磁带。

The tape has reached end. Please replace it with #2. Press Enter when ready.

更换新的磁带后，按下 Enter 键，系统将会继续复制文件，直至把所有的文件均写入磁带中。

在上述例子中，“-M”选项用于指定更换介质时输出的提示信息，“-O /dev/st0”选项表示指定的输出文件为磁带机。

## 3. 使用 cpio 命令从档案文件中恢复所有的文件

如果先前复制到磁带上的档案文件采用的是相对路径名，当使用 cpio 命令读取磁带档案文件时，读入的文件将会被存放到当前工作目录中。但是，如果先前复制时采用的是绝对路径名，读入的文件将会采用相同的绝对路径名，覆盖原有的目录和文件。

由此可见，使用绝对路径名的文件备份方式具有潜在的危险性。如果现有的文件有所变动，恢复文件后将会使改动的文件丢失数据（当然，有时也许恰恰需要这样做——覆盖修改有误的文件）。

为了把先前复制到磁带上的档案文件恢复到系统中，首先选择一个目录，然后执行下列 cpio 命令（其中，“-v”选项表示在屏幕上显示抽取的文件）。

```
$ cd /newdir
$ sudo cpio -icduvB < /dev/st0
.
.
.
data.0630
data.0731
data.0831
data.0930
data.1031
1024050 blocks
$
```

执行上述命令后，将会把先前复制的所有文件恢复到选定的目录中。通过下列命令可以获得



### 简单的验证。

```
$ ls -l  
总用量 512544  
-rw-r--r-- 1 gqxing gqxing 104857600 2008-06-30 23:30 data.0630  
-rw-r--r-- 1 gqxing gqxing 104859648 2008-07-31 23:30 data.0731  
-rw-r--r-- 1 gqxing gqxing 104861696 2008-08-31 23:30 data.0831  
-rw-r--r-- 1 gqxing gqxing 104863744 2008-09-30 23:30 data.0930  
-rw-r--r-- 1 gqxing gqxing 104865792 2008-10-31 23:30 data.1031  
$
```

### 4. 使用 cpio 命令从档案文件中恢复指定的文件

上述 cpio 命令把档案文件中的文件全部恢复到系统中。事实上，也可以从复制的档案文件中抽取单个文件或一组文件。为此，可以使用下列 cpio 命令。

```
$ cpio -icduv "pattern" < /dev/st0
```

其中，“pattern”表示从档案文件中抽取匹配指定模式的所有文件，并恢复到当前工作目录中（注意，如果需要同时指定多个模式，每个模式前后必须加双引号）。下面的例子说明了怎样抽取文件名后缀为“0831”的文件。

```
$ cd /newdir  
$ sudo cpio -icduv "*0831" < /dev/st0  
data.0831  
1024050 blocks  
$ ls -l *0831  
-rw-r--r-- 1 gqxing gqxing 104861696 2008-08-31 23:30 data.0831  
$
```

### 5. 使用 cpio 命令查询先前备份的档案文件

要查询先前备份的档案文件，可以使用下列 cpio 命令（其中，“-t”选项表示仅读取并显示档案文件中的文件列表）。

```
$ sudo cpio -ict < /dev/st0  
  
data.0630  
data.0731  
data.0831  
data.0930  
data.1031  
1024050 blocks  
$
```

如果使用“-v”选项，cpio 命令还会给出更多的文件属性信息，示例如下。

```
$ sudo cpio -icvt < /dev/st0  
drwxr-xr-x 2 gqxing gqxing 0 Nov 9 23:58 .  
-rw-r--r-- 1 gqxing gqxing 104857600 Jun 30 11:30 data.0630  
-rw-r--r-- 1 gqxing gqxing 104859648 Jul 31 11:30 data.0731  
-rw-r--r-- 1 gqxing gqxing 104865792 Oct 31 11:30 data.1031  
-rw-r--r-- 1 gqxing gqxing 104861696 Aug 31 11:30 data.0831  
-rw-r--r-- 1 gqxing gqxing 104863744 Sep 30 11:30 data.0930  
1024050 blocks  
$
```

## 11.2.2 利用 tar 实现备份和恢复

tar 命令可用于备份数据，把指定的文件、目录或文件系统及其中的所有文件组合后生成一个新的档案文件。生成的档案文件可以存储到磁带上，也可以存储到磁盘文件系统中。用户可以检索档案文件，必要时也可以把档案文件恢复到系统。在 Linux 系统中，tar 还具有压缩功能，可以生成压缩的档案文件，因而能够采用 ftp 等网络通信工具，以二进制数据的形式实现系统间的文件复制与交换。

tar 具有如下 8 种运行模式，或者说，具有 8 种用途。

- 创建，利用“-c”选项，创建一个包含多个文件的档案文件，实现文件的备份。
- 替换，利用“-r”选项，把新文件写入档案文件后部，以便能够在恢复文件时实现新老文件的替换。
- 显示，利用“-t”选项，显示档案文件中的文件列表。
- 更新，利用“-u”选项，更新档案文件中的指定文件。
- 抽取，利用“-x”选项，把档案文件恢复到系统中。
- 合并，利用“-A”选项，把一个档案文件附加到另外一个档案文件的后面，实现档案文件的合并。
- 比较，利用“-d”选项，比较档案文件与磁盘中的文件。
- 删除，利用“--delete”选项，从档案文件中删除指定的文件。

tar 命令的语法格式简写如下。表 11-6 给出了常用的 tar 命令选项及其简单说明。

```
tar [-]c [options] file-list
tar [-]r [options] file-list
tar [-]t [options] [file-list]
tar [-]u [options] file-list
tar [-]x [options] [file-list]
tar [-]A tarfile1 -t tarfile2
tar [-]d -t tarfile
tar --delete delete-file -t tarfile
```

表 11-6 tar 命令的常用选项

选 项	GNU 选项	说 明
-c	--create	创建。组合指定的文件，创建一个新的档案文件。如果同名的档案文件已经存在，在创建新的档案文件之前将会清除原有的档案文件
-r	--append	替换。把指定的文件写到档案文件的后部。采用这个选项时，原有的同名文件将会继续保留在档案文件中，并与新的文件共存于档案文件内。当使用“-x”选项抽取文件时，最新的同名文件将会取代原有的文件而被保留下
-t	--list	显示。显示档案文件中的文件列表。文件列表的输出格式类似于“ls -l”命令。如果使用“-t”选项时未指定文件参数，tar 将会显示档案文件中的所有文件；如果指定了文件参数，tar 只会列出与文件参数匹配的文件
-u	--update	更新。如果档案文件中不存在，或者与档案文件中的同名文件相比，指定的文件更新，则把指定的文件写到档案文件的后部。由于这个选项要求执行更多的检查，故其运行速度相对较慢
-x	--extract 或--get	抽取或恢复文件。从档案文件中抽取指定的文件，按照档案文件的目录结构，复制到当前工作目录中。如果使用“-x”选项时未指定文件参数，tar 将会抽取档案文件中的所有文件。如果指定了文件参数，tar 只会抽出与文件参数匹配的文件。如果指定的文件参数含有目录，tar 除抽取匹配的目录之外，还将递归地抽取目录中的所有文件。如果档案文件中包含若干同名的文件，后抽取的文件将会覆盖先前抽取的文件。在抽取文件时，tar 将会尽可能地使抽取文件的文件属主、修改时间和访问权限等属性与档案文件中的初始文件保持一致。注意，在使用“-c”选项创建档案文件时，应尽可能采用相对路径名，否则，tar 命令可能无法正确地匹配文件
-A	--catenate 或 --concatenate	把第一个档案文件附加到第二个档案文件后面，最终形成一个合并的档案文件
	--atime-preserve	在读写文件时不改变文件的访问时间
-d	--diff 或--compare	比较档案文件与磁盘中的文件，检查两者之间是否存在差别，如文件大小和修改时间等
	--delete	从档案文件中删除指定的文件（不适用于磁带档案文件）
-b n	--blocking-size = n	在访问存储介质时，采用“n × 512”个字节的逻辑数据块（n 的默认值为 20）作为读写单位，创建或读取档案文件。通常，在读取档案文件时，tar 将会自动地确定逻辑数据块的大小



续表

选 项	GNU 选项	说 明
	<code>--exclude=pattern</code>	排除匹配指定模式的文件
<code>-f tarfile</code>	<code>--file=tarfile</code>	指定档案文件。档案文件可以是一个普通文件，也可以是一个设备文件（如磁带机等）
<code>-F file</code>	<code>--info-script=file</code> 或 <code>--new-volume-script=file</code>	在读写到每个存储介质（如磁带）的结尾时运行指定的脚本文件（蕴涵着使用“-M”选项）
<code>-g</code>	<code>--listed-incremental</code>	创建/显示/抽取 GNU 新格式的增量备份
<code>-G</code>	<code>--incremental</code>	创建/显示/抽取 GNU 老格式的增量备份
	<code>--help</code>	给出 tar 命令及其有关选项的简单说明
<code>-j</code>	<code>--bzip2</code>	创建和抽取档案文件时分别采用 bzip2 和 bunzip2 压缩或解压档案文件
<code>-L n</code>	<code>--tape-length=n</code>	指定存储介质的容量或档案文件的大小，以 kB 为单位（即 “n × 1 024” 字节）。如果复制的档案文件大于这个选项指定的容量，当写出 n-kB 的数据时，tar 将会提示用户更换新的存储介质（如磁带），从而能够把档案文件分布存储到多个介质上。对于具有固定容量的存储介质（如磁带和软盘），这个选项是非常有用的，可用于创建多卷形式的档案文件
<code>-m</code>	<code>--touch</code>	采用抽取文件时的时间作为文件的修改时间。如果未指定“-m”选项，tar 将恢复文件在档案文件中的初始修改时间。这个选项仅适用于“-x”选项
<code>-M</code>	<code>--multi-volume</code>	创建/显示/抽取多卷档案文件
<code>-N</code>	<code>--after-date date</code> 或 <code>--newer date</code>	仅读写比指定时间更新的文件。指定日期时可以采用 yyymmdd 或 yyyy-mm-dd 等形式表示年月日，指定时间时可以采用 hhmm 或 hh:mm:ss 的形式分别表示时分和时分秒
	<code>--newer-mtime date</code>	仅读写在指定时间之后其内容已经发生变动的文件（参见“-n”选项）
<code>-o</code>	<code>--no-same-owner</code>	以执行 tar 命令的用户及其所属的用户组作为文件恢复后的用户和用户组，而不采用档案文件中文件原有的用户和用户组属性。对于普通用户而言，这是 tar 命令的默认处理方式。如果是超级用户运行 tar 命令，且没有指定“-o”选项，抽取的文件将采用档案文件中文件原有的用户和用户组属性。这个选项仅适用于“-x”选项
	<code>--totals</code>	创建档案文件之后显示已写出的字节总数
<code>-v</code>	<code>--verbose</code>	显示处理过程中读写的每一个文件名。与“-t”选项一起使用时，tar 将会列出文件的详细属性信息，如文件属主、访问权限和文件大小等，其输出结果类似于“ls -l”命令的输出
	<code>--wildcards</code>	在指定文件模式时，允许使用星号“*”等元字符
<code>-X file</code>	<code>--exclude-from=file</code>	表示归档或抽取指定文件中未列举的其他所有文件（即排除指定文件中包含的任何文件）。如果指定的文件是一个目录，则排除指定的目录及其中的任何文件和子目录
	<code>--exclude=file</code>	表示归档或抽取除了指定文件之外的其他所有文件（即排除指定的任何文件）。如果指定的文件是一个目录，则排除指定的目录及其中的任何文件和子目录
<code>-z</code>	<code>--gzip</code> 或 <code>--gunzip</code>	创建和抽取档案文件时分别采用 gzip 和 gunzip 压缩或解压档案文件
<code>-Z</code>	<code>--compress</code> 或 <code>--uncompress</code>	创建和抽取档案文件时分别采用 compress 和 uncompress 压缩或解压档案文件

假定要备份/home/gqxing/src 目录中的所有文件，把归档后生成的档案文件存放在/data 目录中，可以使用下列 tar 命令。

```
$ cd /home/gqxing
$ tar -cvf /data/atmsrc.tar src
src/
src/atmmmon.c
```

```
src/atmcom.c
src/listerner.c
src/handler.c
src/atmstat.c
$
```

如果生成的 atmsrc.tar 档案文件过大，可以选用“-j”或“-z”等选项，使 tar 能够调用 bzip2、gzip 或 gunzip 等工具压缩档案文件，最终生成一个压缩的档案文件，示例如下。

```
$ ls -l /data/*.tar
-rw-r--r-- 1 gqxing gqxing 133120 2008-11-10 11:20 /data/atmsrc.tar
$ tar -cjvf /data/atmsrc.tar src
src/
src/atmmon.c
src/atmcom.c
src/listerner.c
src/handler.c
src/atmstat.c
$ ls -l /data/*.tar
-rw-r--r-- 1 gqxing gqxing 25185 2008-11-10 11:25 /data/atmsrc.tar
$
```

必要时，可以从备份的档案文件中恢复指定的文件，或者恢复全部文件。如果需要恢复归档的全部文件，可以使用下列 tar 命令。

```
$ cd /newdir
$ tar -xvf /data/atmsrc.tar
src/
src/atmmon.c
src/atmcom.c
src/listerner.c
src/handler.c
src/atmstat.c
$
```

下面以 1.44MB 软盘（其设备名为/dev/fd0）为例，说明怎样使用 tar 命令备份或恢复单个文件、一组文件甚至整个文件系统。实际上，这里介绍的备份与恢复过程同样也适用于其他存储介质（如磁带、磁盘或 CD/DVD 等），只需把设备文件名改换为相应存储介质的设备文件名即可。

### 1. 使用 tar 命令把文件归档备份到软盘上

在使用 tar 命令把文件复制到存储介质上时，应当注意下列事项。

- 使用 tar 命令的“-c”选项创建档案文件时，将会毁灭存储介质上原有的任何文件。
- 在复制文件时，可以使用文件名生成元字符“?”、“\*”和“[...]”指定文件名。例如，为了复制所有以“.doc”为后缀的文件，可以使用“\*.doc”作为文件名参数。
- 在抽取文件时，也可以使用文件名生成元字符“?”、“\*”和“[...]”，从 tar 档案文件中抽取匹配的文件。
- 包含 tar 档案文件的存储介质不能作为文件系统安装。

若要使用 tar 命令把档案文件复制到软盘上，首先应进入需要复制文件的目录，然后使用下列 tar 命令复制文件。

```
tar -cvf /dev/fd0 filenames
```

其中，“-c”选项表示创建一个档案文件，“-v”选项表示在屏幕（标准输出）上显示正在处理的每一个文件及其相关的属性信息，“-f /dev/fd0”表示把创建的档案文件写到软盘中，filenames 是准备归档的文件或目录名。示例如下。

```
$ cd /home/gqxing
$ ls -l src
总用量 128
```



```
-rw-r--r-- 1 gqxing gqxing 18222 2008-11-08 11:50 atmcom.c
-rw-r--r-- 1 gqxing gqxing 11802 2008-11-08 11:52 atmmon.c
-rw-r--r-- 1 gqxing gqxing 23221 2008-11-08 12:05 atmstat.c
-rw-r--r-- 1 gqxing gqxing 11742 2008-11-08 11:53 handler.c
-rw-r--r-- 1 gqxing gqxing 55596 2008-11-08 11:56 listener.c
$ tar -cvf /dev/fd0 src
src/
src/atmmon.c
src/atmcom.c
src/listerner.c
src/handler.c
src/atmstat.c
$
```

要验证指定的文件是否已经被复制到软盘上，可以使用下列命令（其中，“-t”选项表示读取并显示档案文件中的文件列表）。

```
$ tar -tvf /dev/fd0
drwxr-xr-x gqxing/gqxing 0 2008-11-10 00:32 src/
-rw-r--r-- gqxing/gqxing 23221 2008-11-08 12:05 src/atmstat.c
-rw-r--r-- gqxing/gqxing 11742 2008-11-08 11:53 src/handler.c
-rw-r--r-- gqxing/gqxing 11802 2008-11-08 11:52 src/atmmon.c
-rw-r--r-- gqxing/gqxing 18222 2008-11-08 11:50 src/atmcom.c
-rw-r--r-- gqxing/gqxing 55596 2008-11-08 11:56 src/listerner.c
$
```

### 2. 使用 tar 命令把文件归档备份到多个软盘上

当使用 tar 命令把档案文件写入软盘时，应注意整个档案文件的大小和软盘的容量。如果数据量过大，一个软盘无法容纳所有的数据，因而需要使用多个软盘时，可以使用“-L”选项指定软盘的容量，以便能够在一个软盘复制完成之后，由系统提示用户更换新的软盘。下面例子中使用“-L 1440”选项指定 1.44MB 软盘的容量。

```
$ cd /opt/atm
$ tar -L 1440 -cvf /dev/fd0 doc
doc/
doc/user-guide
doc/reference-manual
Prepare volume #2 for '/dev/fd0' and hit return:
doc/release-notes
doc/technical-specifications
$
```

当复制到中途时，系统将会提醒用户更换新的软盘。在更换下一个软盘之后，按下 Enter 键，系统将会继续执行文件的复制，直至把所有的文件均写入软盘中。

### 3. 使用 tar 命令从档案文件中恢复所有的文件

要从先前备份到软盘的档案文件中恢复所有的文件，首先应进入某个选定的目录，然后使用下列命令抽取其中文件。

```
tar -xvf /dev/fd0 [filenames]
```

其中，“-x”选项表示从指定的档案文件中抽取文件，并把抽取的文件恢复到当前工作目录中。“-v”选项表示在屏幕（标准输出）上显示抽取的每一个文件及其相关的属性信息。“-f /dev/fd0”表示软盘设备。Filenames 是准备抽取的文件。如果同时指定多个文件，文件名之间应加空格字符。默认情况下（未指定文件名参数），tar 命令将会抽取所有的文件。示例如下。

```
$ cd /var/tmp
$ tar -xvf /dev/fd0
src/
src/atmcom.c
src/atmmon.c
src/listerner.c
```

```

src/handler.c
src/atmstat.c
$ ls -l src
总用量 128
-rw-r--r-- 1 gqxing gqxing 18222 2008-11-08 11:50 atmcom.c
-rw-r--r-- 1 gqxing gqxing 11802 2008-11-08 11:52 atmmon.c
-rw-r--r-- 1 gqxing gqxing 23221 2008-11-08 12:05 atmstat.c
-rw-r--r-- 1 gqxing gqxing 11742 2008-11-08 11:53 handler.c
-rw-r--r-- 1 gqxing gqxing 55596 2008-11-08 11:56 listener.c
$
```

#### 4. 使用 tar 命令从档案文件中抽取指定的文件

如前所述，在“tar -x”命令中，如果命令行中未指定文件参数，则意味着抽取档案文件中的所有文件。要抽取特定的文件，需要在命令行中指定文件名，指定文件名时必须严格地匹配档案文件中的文件名。如果无法肯定文件名或路径名，可以使用下面下一小节将要介绍的过程，获取档案文件中的完整文件列表。

假定需要从先前备份的档案文件中抽取 atmmon.c 文件，可以使用下列命令。

```

$ cd /newdir
$ tar -xvf /dev/fd0 src/atmmon.c
src/atmmon.c
$ ls -l src
总用量 12
-rw-r--r-- 1 gqxing gqxing 11802 2008-11-08 11:52 src/atmmon.c
$
```

在 tar 命令中，也可以使用文件名生成元字符指定抽取的文件。例如，要抽取档案文件中所有以“.c”为后缀的文件，可以使用下列命令。

```

$ tar --wildcards -xvf /dev/fd0 *.c
src/
src/atmmon.c
src/atmcom.c
src/listerner.c
src/handler.c
src/atmstat.c
$
```

但在使用“?”、“\*”和“[...]”文件名生成元字符时，需要使用准确的匹配模式，才能够抽取期望的文件。例如，假定新建的档案文件中还存在一个 Makefile 或 makefile 文件，在抽取这个文件时，可以使用“src/[Mm]\*”作为匹配模式，但不能简单地使用的 “[Mm]\*”，因为 tar 会把整个目录和文件的路径名看作一体。示例如下。

```

$ tar -xvf /dev/fd0 src/[Mm]*
src/makefile
$
```

#### 5. 使用 tar 命令查询先前备份的归档文件

要查询先前备份的档案文件，可以使用下列命令。

**tar -tvf /dev/fd0**

其中，“-t”选项表示读取并显示档案文件中的文件列表，“-v”选项表示在屏幕（标准输出）上显示每一个文件及其相关的属性信息，“-f /dev/fd0”表示软盘设备。示例如下。

```

$ tar -tvf /dev/fd0
drwxr-xr-x gqxing/gqxing 0 2008-11-10 01:11 src/
-rw-r--r-- gqxing/gqxing 11802 2008-11-08 11:52 src/atmmon.c
-rw-r--r-- gqxing/gqxing 18222 2008-11-08 11:50 src/atmcom.c
-rw-r--r-- gqxing/gqxing 55596 2008-11-08 11:56 src/listerner.c
-rw-r--r-- gqxing/gqxing 11742 2008-11-08 11:53 src/handler.c
-rw-r--r-- gqxing/gqxing 23221 2008-11-08 12:05 src/atmstat.c
-rw-r--r-- gqxing/gqxing 2025 2008-11-08 12:31 src/makéfile
$
```



### 11.2.3 利用 dd 实现文件系统的原样复制

**dd** 命令的最大特点是能够采用原始读写的方式，逐块逐道地把位于存储介质上的数据原封不动地复制到另一个存储介质上。此外，还可以把一个文件系统完整地复制到另一个磁盘分区中，从而实现文件系统的备份。通常，**dd** 命令的功能是把标准输入复制到标准输出。

在复制过程中，**dd** 命令还具有数据转换功能，可以把不同代码格式的数据转换为另一种编码的数据，实现不同系统之间的数据交换。

与其他大多数命令相比，**dd** 命令的语法格式明显不同。在 **dd** 命令中，命令选项采用“keyword=value”的形式。其中，keyword 相当于其他命令中的选项，value 是选项的参数。例如，当需要复制某个文件时，可以采用下列简单的命令语法格式。

```
dd if=input-file of=output-file
```

表 11-7 给出了 **dd** 命令的部分常用选项、参数及其说明。

表 11-7

**dd** 命令的部分常用选项

选项与参数	说 明
if= <i>file</i>	指定输入文件。文件可以是普通数据文件，也可以是磁盘分区、磁带、软盘或 CD/DVD 等设备文件。默认值为标准输入
of= <i>file</i>	指定输出文件。文件可以是普通数据文件，也可以是磁盘分区、磁带、软盘或 CD/DVD 等设备文件。默认值为标准输出
ibs = <i>n</i>	指定输入数据块的大小，数值后面可以附加一个表示数据块单位的字符，如 b、K、M、G 和 T 等，分别表示以 <i>n</i> 个 512 字节、1KB、1MB、1GB 和 1TB 的数据块为数据输入单位。如果未指定表示数据块单位的字符，则以字节为计数单位。如果均未指定，默认值为 512 字节
obs = <i>n</i>	指定输出数据块的大小，数值后面可以附加一个表示数据块单位的字符，如 b、K、M、G 和 T 等，分别表示以 <i>n</i> 个 512 字节、1KB、1MB、1GB 和 1TB 的数据块为数据输出单位。如果未指定表示数据块单位的字符，则以字节为计数单位。如果均未指定，默认值为 512 字节
bs = <i>n</i>	指定读写数据块的大小，数值后面可以附加一个表示数据块单位的字符，如 b、K、M、G 和 T 等，分别表示以 <i>n</i> 个 512 字节、1KB、1MB、1GB 和 1TB 的数据块为输入单位。如果未指定表示数据块单位的字符，则以字节为计数单位。如果均未指定，默认值为 512 字节。此选项可以同时强制替代 ibs 和 obs 两个选项
skip = <i>n</i>	在开始复制之前，先从输入文件的起始位置跳过指定数量的数据块（以 ibs 或 bs 定义的输入数据块为计数单位）
seek = <i>n</i>	在开始复制之前，先从输出文件的起始位置跳过指定数量的数据块（以 obs 或 bs 定义的输出数据块为计数单位）
count = <i>n</i>	指定复制的数据块数量。每个数据块的字节数以 ibs、obs 或 bs 的定义为准

例如，要复制一个软盘，可以使用 **dd** 命令先把软盘复制到一个临时文件中，如下所示。

```
$ dd if=/dev/fd0 of=/tmp/tmpfile  
2880+0 records in  
2880+0 records out  
1474560 bytes (1.5 MB) copied. 48.6294 seconds. 30.3 kB/s  
$
```

然后，再使用 **dd** 命令，交换输入和输出文件的位置，把临时文件复制到一个新的软盘中，如下所示。

```
$ dd if=/tmp/tmpfile of=/dev/fd0  
2880+0 records in  
2880+0 records out  
1474560 bytes (1.5 MB) copied. 98.822 seconds. 14.9 kB/s  
$
```

运行结束之后，**dd** 命令将会给出已经读写的数据块数量。其中，加号“+”之前的数字表示复制过程中读写的完整数据块数量，加号“+”之后的数字表示复制了多少个不完整的数据块。

要把磁盘分区 2 中的文件系统复制到磁盘分区 3 中，可以使用下列命令。

```
$ sudo dd if=/dev/sda2 of=/dev/sda3 bs=1024K
xxxx+0 records in
xxxx+0 records out
xxxxxxxxxx bytes (x.x GB) copied. xxx.xxx seconds. x.x MB/s
$
```

然后，可以使用“fsck -f”命令，检测新复制的文件系统。最后，再使用mount命令安装新的文件系统。命令如下。

```
$ sudo fsck -f /dev/sda3
...
$ sudo mount /dev/sda3 /mnt
$
```

在使用dd命令复制磁盘分区（文件系统）时，应注意下列事项：

- 确保源磁盘分区与目的磁盘分区具有相同的容量（且最好为同一型号的磁盘）；
- 使用fsck命令检测新复制的文件系统，然后利用mount命令予以检验；
- 使用dd命令复制磁盘分区时，应确保系统处于单用户运行模式，从而保证源磁盘分区中的数据不会发生变化。

同样，dd命令也可用于复制磁带。如果系统配有两个磁带机，可以使用下列命令，实现从磁带到磁带的直接复制。否则，需要先把数据复制到磁盘上，然后再复制到新的磁带中。

```
$ dd if=/dev/st0 of=/dev/st1
```

## 11.3 采用专用工具备份与恢复数据

除了cpio、tar与dd等基本的软件工具之外，Ubuntu Linux系统还提供诸如dump/restore、amanda、rsync、BackupPC及bacula等专业备份工具，可以选择使用。但这些软件工具需要单独安装，分别简述如下。

- **dump/restore:** dump与restore是一对分工明确的数据备份与恢复工具，其中，dump仅用于备份数据，restore负责恢复数据。注意，dump与restore仅适用于Ext2/Ext3文件系统的数据备份，能够准确地备份与恢复其中的任何类型的文件，如设备特殊文件等。此外，dump/restore只能用于备份与恢复整个文件系统，其备份方式包括完整备份与增量备份。dump/restore采用/etc/dumpdates文本文件的形式，提供哪个文件系统已经备份，执行的备份级别等信息。dump创建的档案文件能够跨越多个磁带等存储介质，而restore能够以交互方式运行，且能够仿真备份数据的实际目录层次，供用户查询、选择准备恢复的目录与文件。
- **amanda:** amanda是“Advanced Maryland Automated Network Disk Archiver”的英文缩写，由马里兰大学开发。利用amanda，能够非常容易地把任何远程系统的数据集中备份到中心服务器，同时提供完整的管理功能。采用不同的配置，amanda可以支持多种备份方式与手段，生成详细的备份报告与日志信息，通过电子邮件，能够自动提交备份报告。中心服务器与远程系统的数据通信采用加密方式，从而确保数据的安全。amanda分为两个软件包——amanda-server和amanda-client，这两个软件包均需单独安装。amanda的网站地址是<http://www.amanda.org>。
- **rsync:** rsync是一个命令行的文件与目录同步软件工具，便于用户在系统之间复制文件和目录。当本地与远程系统均存在文件和目录的副本时，rsync能够有效地减少传输的数据。



量，确保本地与远程系统的文件和目录副本是等同的。rsync 采用远程更新协议，能够确保仅传输两组文件和目录副本的不同部分。rsync 是 Ubuntu Linux 系统的基本组成部分之一，不需要单独安装，但在运行前需要对准备在本地系统备份的远程系统作适当的配置。

- **BackupPC:** BackupPC 提供一个基于浏览器界面的备份工具，能够利用 Samba、tar 或 rsync 备份远程系统。BackupPC 创建的远程系统备份可在 BackupPC 服务器中集中存储和管理，授权的用户能够从中恢复自己的文件。针对每一个系统，BackupPC 服务器能够分别存储不同的配置数据，从而能够针对每一个系统，采用不同的命令、协议与备份方式，实现不同类型的数据备份。BackupPC 的另外一个特点是采用标准的 Linux 系统命令创建数据备份，因而远程系统不需要安装额外的软件。有关 BackupPC 软件工具的详情，可以访问 <http://backuppc.sourceforge.net>。
- **bacula:** bacula 包含一组强有力的软件与文档，提供网络备份功能，支持 Linux、UNIX 和 Windows 系统。与 amanda 相比，在如何存储和存储位置的选择方面，bacula 显得更灵活。但是，正由于其功能强大，因而使用起来也较复杂。尽管其提供了一个图形界面的控制台，但 bacula 仍然是一个命令行的软件工具。有关 bacula 软件工具的使用说明，详见 <http://www.bacula.org>。

在上述备份软件工具中，dump/restore 只能用于本地系统的数据备份，其他软件工具可用于网络环境的数据备份。

本节将以 dump/restore 命令为例，介绍系统数据的备份与恢复过程。除了少数的增量备份之外，在实施大量数据的备份时，磁带、磁盘或 CD/DVD 是必不可少的备份介质，软盘仅适用于少量数据的备份。

在开始备份数据之前，通常应事先采用 df、du 或 dump 等命令，对需要备份的文件系统或磁盘分区的数据量进行评估，以确定应采用何种类型的备份介质，以及需要准备多少存储介质。

在安装且适当地设置了全部系统软件之后，可以从根目录开始，对整个系统做一个完整的备份，或者对业务数据所在的文件系统单独做一个完整的备份。之后，只需定期备份/var、/home 和业务数据所在的文件系统。一旦系统受到破坏，可以利用系统或业务数据的完整备份或增量备份，恢复系统与数据。

在系统运行期间，/var、/home 和业务数据所在的文件系统经常会频繁地发生变动，因此最好每周执行一次完整的文件系统备份，每天执行一次增量备份。

### 11.3.1 利用 dump 命令实现数据的备份

dump 命令用于备份指定的文件系统，把整个文件系统中的目录文件写入指定的备份介质中，如磁带、磁盘或 CD/DVD 等。也可以采用增量备份的方式，仅备份一定时间内发生变动的文件。dump 命令的语法格式简写如下。

```
dump [-level#] [options] [-A file] [-B records] [-b blocksize] [-D file]
      [-f file] [-F script] [-L label] [-T date] files-to-dump
```

dump 命令的选项可用于指定备份的方式（如完整备份和增量备份）、备份目的（如备份文件或备份介质）、备份文件列表、备份记录的数据块数量、脚本文件、介质卷标、多卷备份以及其他备份要求。表 11-8 给出了部分常用的选项及其简单说明。files-to-dump 可以是一个文件系统的安

装点，也可以是作为文件系统子集的一组文件与目录列表，用于指定想要备份的数据。

表 11-8 dump 命令的部分常用选项

选 项	简 单 说 明
-level#	备份级别。备份级别可以是任何整数（之前，合法的备份级别为 0~9，现可为任意整数），其中 0 表示完整备份整个文件系统。大于 0 的任何备份级别均为增量备份，这意味着仅备份自上次执行较低级别的备份以来新建或修改过的所有文件。默认的备份级别为 9
-a	表示绕过存储介质的容量计算，只管写入存储介质，直至遇到介质结束标志
-A archive_file	生成一个备份文件列表，以便 restore 命令能够确定准备恢复的文件是否位于备份介质中
-b blockszie	指定每个备份记录包含多少 1KB 的数据块。通常，每个备份记录包含 10 个 1KB 的数据块
-B records	指定每个存储介质能够容纳 1KB 数据块的数量。通常不需要指定，dump 能够检测存储介质的结束标志。当达到指定的容量或写满存储介质时，dump 将会提示并等待用户更换介质
-D file	指定存有完整或增量备份信息的文件路径名。默认的文件路径名为 /etc/dumpdates
-f file	把备份的数据写到指定的文件。文件参数可以是设备文件（如 /dev/st0 磁带机）、普通文件或减号 “-”（标准输出）。如果需要指定多个文件，文件名之间需加逗号 “,” 分隔符。每个文件可以按给定的顺序用作多卷备份中的一个备份卷。如果指定的文件为 “host:file” 或 “user@host:file”，dump 将会利用 rmt 命令，把备份的数据写入远程主机中的指定文件（文件事先必须存在，dump 不会创建新的文件）。远程 rmt 命令的默认路径名为 /etc/rmt（必要时可以使用环境变量 RMT 重新定义）
-F script	在写入每个备份卷（最后一个除外）结束之后，运行指定的脚本。设备文件名和备份卷的当前卷号将作为命令行参数被传递给指定的脚本。如果脚本运行结束后返回 0，意味着 dump 可以继续执行备份，无需请求用户更换磁带；否则，如果脚本返回 1，dump 将会提示并请求用户更换介质，然后才能继续执行备份。其他返回码将会导致 dump 终止执行
-L label	指定的字符串可以写入备份介质的头部，作为备份介质的卷标，使 restore 和 file 等工具软件能够快速访问以及识别文件的类型。注意，字符串的长度当前仅限于 16 个字符（包括终止符 “\0”）
-M	支持多卷备份功能。“-f”选项指定的文件名将会作为一个前缀，用于命名一系列备份文件，如 <filename>001、<filename>002 等。dump 将会依次写入每个备份文件。在一个 Ext2 文件系统中，为了存储备份文件，同时克服 2GB 文件大小的限制，这个选项是非常有用的
-S	显示指定文件系统、目录或文件的字节数量，以便估算存储空间需求。在开始备份之前，可用此选项确定存储空间的容量需求，从而能够确定需要多少备份介质
-T date	使用指定的时间（而不是 /etc/dumpdates 文件中的时间）作为备份的起始时间，时间的格式为 “www mmm dd hh:mm:ss yyyy”，如 “Thu Jan 17 00:55:05 2008”，时区偏移值的格式为 “±HHMM”，如 +0800（北京时间）。如果未指定时区偏移值，可以把备份时间看作本地时间。注意，“-T” 选项不能与 “-u” 选项一起使用
-u	在成功地完成备份之后，更新 /etc/dumpdates 文件。文件的记录格式如下： <文件系统名> <增量备份级别> <备份时间（同上）> <时区偏移值（同上）>

如前所述，事先了解备份的数据量究竟有多大是非常有用的，这有助于确定应采用什么备份介质，以及需要准备多少存储介质。例如，要确定 /appdata 文件系统的数据量，以便制作一个业务数据的完整备份，可以利用 dump 命令的 “-S” 选项，如下所示。

```
$ sudo dump -S /appdata
4026958848
$
```

上述 dump 命令的返回结果 4 026 958 848 表示指定文件系统现有数据的字节数，经计算约为 4GB，这意味着至少需要使用一个 7GB 的 DLT 磁带。要制作一个备份级别为 0 的完整备份，把数据写到一个 7GB 的磁带上，且在完成备份后更新备份记录文件 /etc/dumpdates，可以使用下列命令。

```
$ sudo dump -0u -f /dev/st0 /appdata
DUMP: Date of this level 0 dump: Mon Nov 10 22:00:09 2008
DUMP: Dumping /dev/sdb6 (/appdata) to /dev/st0
```



```
DUMP: Label: /appdata  
DUMP: Writing 10 Kilobyte records  
.....  
$
```

在制作了完整备份之后，可以选用一个大于 0 的备份级别，采用下列 dump 命令，对发生变动的文件或数据，定期地执行增量备份。

```
$ sudo dump -9u -f /dev/st0 /appdata  
DUMP: Date of this level 9 dump: Mon Nov 10 22:05:10 2008  
DUMP: Date of last level 0 dump: Mon Nov 10 22:00:09 2008  
DUMP: Dumping /dev/sdb6 (/appdata) to /dev/st0  
DUMP: Label: /appdata  
DUMP: Writing 10 Kilobyte records  
.....  
$
```

利用 “-f” 选项，dump 命令还支持远程备份。当备份的数据大于单个存储介质的容量时，可以采用多卷备份方式，把数据分布于多个存储介质上。

注意，在运行 dump 命令时，文件系统应处于静止状态，否则，无法确保当前文件系统与实际备份数据的一致性，从而导致 restore 命令无法正确地恢复文件。一个文件系统能够保持静止状态，仅当文件系统已经卸载，或者系统处于单用户运行模式时。

### 11.3.2 利用 restore 命令实现数据的恢复

在恢复备份的数据时，需要准确地知道究竟要恢复多少数据，恢复哪些数据。如果选择完整的恢复，可能会浪费时间。如果恢复选定的文件，必须确保已经恢复了受到影响的所有文件。在一个业务系统中，通常也许只需恢复最近一次的增量备份，而在其他情况下，可能还需要恢复最近一次的完整备份，然后再依次恢复之后的增量备份。在恢复期间，切忌恢复不必要且其数据已经过时的文件。

restore 命令用于恢复先前采用 dump 命令制作的任何备份数据，如文件系统的完整备份或增量备份。它可以恢复单个文件、选定的部分文件或整个文件系统，也可以先恢复完整的文件系统，在此基础上，恢复随后制作的增量备份。此外，利用 dump/restore 的远程备份与恢复功能，还可以从远程主机中恢复备份的数据（参见 “-f” 选项）。restore 命令的语法格式简写如下。

```
restore -C [options] [-b blocksize] [-D filesystem] [-f file]  
restore -i [options] [-A file] [-b blocksize] [-f file]  
restore -R [options] [-b blocksize] [-f file]  
restore -r [options] [-b blocksize] [-f file]  
restore -t [options] [-A file] [-b blocksize] [-f file] [-X filelist] [file]  
restore -x [options] [-A file] [-b blocksize] [-f file] [-X filelist] [file]
```

在运行 restore 命令时，restore 必须处于下列 6 种常用的运行模式之一，其他命令选项和参数是可选的。

- 比较文件，利用 “-C” 选项，能够比较备份介质与磁盘中的文件。
- 交互恢复，利用 “-i” 选项，能够采用交互方式，浏览备份介质，恢复选定的文件。
- 断点恢复，利用 “-R” 选项，能够从指定的备份介质开始，执行先前中断的恢复。
- 完整恢复，利用 “-r” 选项，能够重建并恢复一个完整的文件系统。
- 显示文件，利用 “-t” 选项，可以从备份介质读取并显示文件列表。
- 部分恢复，利用 “-x” 选项，能够从指定的备份介质恢复指定的文件。

在 restore 命令的语法格式中，“-C”、“-i”、“-R”、“-r”、“-t” 或 “-x” 等选项用于确定 restore 命令的运行模式，运行时必选其一。文件参数 file（包括 “-f” 选项中的文件参数 file）用于指定准备

恢复的文件、目录或文件系统的根目录。表 11-9 给出了 restore 命令支持的常用选项及其简单说明。

表 11-9 restore 命令的部分常用选项

选 项	简 单 说 明
-C	利用这个选项，restore 能够读取备份介质，并与系统中现有的文件进行比较。在开始比较之前，restore 首先会把当前工作目录改换到备份的文件系统的根目录，然后逐一比较系统与备份介质中的文件
-i	<p>采用交互方式，从备份介质中恢复文件。在从备份介质中读取文件目录信息之后，restore 将会提供一个命令行界面，使用户能够遍历目录树，从中选择想要恢复的文件。下面是部分可用的交互命令（对于需要提供参数的交互命令，默认的参数是当前目录）。</p> <ul style="list-style-type: none"> <li>□ add [arg] —— 把当前或指定目录及其中的所有文件（包括子目录中的文件）加到要恢复的文件列表中，除非在命令行中指定了“-h”选项。当利用 ls 交互命令显示文件时，位于要恢复文件列表中的文件名之前冠有一个星号“*”前缀。</li> <li>□ cd arg —— 把当前目录改换到指定的目录。</li> <li>□ delete [arg] —— 从要恢复的文件列表中删除当前或指定目录及其中的所有文件（包括子目录中的文件），除非在命令行中指定了“-h”选项</li> <li>□ extract —— 根据要恢复的文件列表，从备份介质中恢复其指定的所有文件。执行前，restore 将会询问从哪一个备份介质开始恢复文件。恢复部分文件时，最好能够从一个确定的备份介质开始（而非总是从头开始）恢复文件。</li> <li>□ help —— 列出可用的交互命令及其简单说明。</li> <li>□ ls [arg] —— 列举当前或指定目录中的文件和目录。如果是一个目录，其名字后面将会附加一个“/”后缀。如果是一个要恢复的文件或目录，其名字前面有一个星号“*”标记。</li> <li>□ pwd —— 显示当前工作目录的完整路径名。</li> <li>□ quit —— 退出 restore 命令</li> </ul>
-R	从多卷备份的指定介质开始继续恢复（参见“-r”选项）。在全面恢复期间，如果 restore 的恢复过程因故中断，使用这个选项可以从断点处开始继续恢复
-r	重建或恢复一个文件系统。全面恢复时，restore 将会利用 mke2fs 命令创建并安装一个新的文件系统，然后进入文件系统的根目录，递归地全面恢复采用备份级别 0 备份的整个文件系统。如果已经成功地恢复了一个完整备份，还可以利用“-r”选项，恢复任何必要的增量备份
-t	如果存在，列出备份介质中的指定文件。如果未指定文件参数，则从根目录开始，列出备份介质中的所有目录和文件，除非指定了“-h”选项（参见“-X”选项）
-x	从给定的备份介质中抽取指定的文件，并把文件恢复到系统中。如果指定的文件参数匹配备份介质中的某个目录，且未指定“-h”选项，则递归地抽取目录中的文件，并尽可能地恢复文件的属主、修改时间和访问权限等属性。如果未指定文件参数，则从根目录开始，恢复备份介质中的所有文件，除非指定了“-h”选项（参见“-X”选项）
-a	当使用“-i”或“-x”选项恢复文件时，restore 将会请求用户提供一个卷号，表示从哪一个备份介质开始恢复文件，这一做法能够直接读取目标备份介质，减少文件恢复的时间。“-a”选项表示禁止采用这种处理方式，即总是从第一个备份卷开始，依次恢复所有的备份介质。如果不知道究竟应从哪一个备份介质开始恢复文件，这个选项是比较有用的
-A archive_file	从指定的文件而非备份介质中读取文件列表，以便在不访问备份介质的情况下，能够确定要恢复的文件是否存在于备份介质。这个选项可与“-t”、“-i”或“-x”选项一起组合使用
-b blocksize	指定每个备份记录包含多少个 1KB 的数据块。如果未指定“-b”选项，restore 将会尝试自动确定存储介质中备份记录的数据块数量
-D filesystem	当采用“-C”选项比较备份介质时，可用“-D”选项指定文件系统的名称
-f file	从指定的文件中恢复备份的数据。文件参数可以是一个设备文件（如磁带机/dev/st0、磁盘分区/dev/sda1 等）、普通文件或减号“-”（标准输出）。如果文件参数是一个形如“host:file”或“user@host:file”的远程文件，restore 将会利用 rmt 命令，从远程主机中恢复指定的文件
-h	仅表示目录本身而不涉及其中的文件。这将防止 restore 递归地恢复整个目录层次子树
-M	多卷恢复。用于恢复采用“-M”选项备份的多卷存储介质。“-f”选项指定的文件名将会作为一个前缀，使 restore 能够按照“<filename>-001”和“<filename>-002”等的顺序，尝试读取多个存储介质或备份文件
-N	利用“-N”选项，restore 能够执行“-i”、“-R”、“-r”、“-t”或“-x”选项定义的恢复动作，但实际上并不把任何文件写入磁盘中



续表

选 项	简 单 说 明
<code>-o</code>	与“-i”或“-x”恢复模式不同，利用“-o”选项， <code>restore</code> 能够自动恢复当前目录，而无需操作员干预
<code>-u</code>	恢复文件时，如果存在同名的文件， <code>restore</code> 将会发出一个警告信息。要避免这种情况出现，可以使用“-u”选项，使 <code>restore</code> 能够强行覆盖原有的文件
<code>-v</code>	在 <code>restore</code> 的恢复过程中，通常不会显示任何处理信息。“-v”选项使 <code>restore</code> 能够显示其处理的每一个文件，并在文件名之前冠以文件的类型
<code>-V</code>	支持非磁带（如CD/DVD）的多卷恢复功能
<code>-X filelist</code>	除了命令行中指定的文件之外，再从指定的文件中读取文件列表，组成一个完整的文件列表。这个选项可与“-t”或“-x”选项一起使用。在制作文件列表时，每个文件名占一行

如前所述，当使用“-i”选项调用`restore`命令时，可以进入交互恢复方式，此时可以浏览备份介质，查询其中的文件内容，确定要恢复或放弃哪些文件。因此，下列命令将会启动一个交互会话。

```
$ sudo restore -i -f /dev/st0  
restore >
```

在“`restore >`”提示符下，可以使用`cd`和`ls`等交互命令，查询指定备份磁带（其设备文件名为`/dev/st0`）中的备份数据，从显示的文件列表中选择要恢复的文件和目录，然后利用`add`交互命令，把选定的目录或文件加到要恢复的文件列表中；或者利用`delete`命令，从文件列表中删除不准备恢复的目录或文件。最后使用`extract`交互命令，恢复选定的文件。在交互恢复模式中，也可以使用`help`交互命令查阅可用的命令，了解命令的简单用法。

除了“-C”、“-i”、“-R”、“-r”、“-t”和“-x”等运行模式选项之外，`restore`命令的其他选项可用于指定恢复的处理模式、恢复文件列表、读写数据块的大小、备份数据的位置、多卷恢复，以及显示命令的处理结果等。

例如，要从指定的备份介质`/dev/st0`中，从头开始完整地恢复`/appdata`文件系统，而不管是否存在同名的文件，强制执行文件恢复，可以使用下列命令。

```
$ sudo restore -rau -f /dev/st0
```

## 11.4 文件系统限额管理

本节将以Ext2/Ext3文件系统为例，说明文件系统磁盘空间和信息节点的限额设置。

### 11.4.1 限额概述

#### 1. 什么是限额

限额机制使系统管理员能够控制文件系统的空间分配与使用。利用限额可以限制每个用户能够使用的磁盘空间和信息节点数量（即文件的数量）。由于这个原因，对于用户主目录所在的文件系统，限额是特别有用的，而在公用的文件系统（如`/tmp`）中建立限额则没有太多的实际意义。

即使已经设定了限额，也可以随时调整用户可用的磁盘空间与信息节点的数量，而且可以根据系统需求的变化增加甚至删除限额。

另外，限额的状态也可以监控。作为系统管理员，可以利用`repquota`和`quota`命令显示文件系统的限额信息，或者检索超限的用户。

## 2. 限额的软性设置与硬性设置

在设置磁盘空间的限额时，可以采用软性限制和硬性限制两种方式。系统不允许用户超越其硬性限制，但允许用户临时超越其软性限制。软性限制必须小于硬性限制，一旦用户超越了软性限制，系统将会给予一个宽限周期，同时启动一个限额时钟。在宽限周期之内，允许用户暂时超越软性限额进行操作，但不能超越硬性限额。一旦由于清理文件使得磁盘空间低于软性限制，限额时钟将会置 0。但是，如果在超过软性限制后的宽限周期之内用户一直未采取任何措施，一旦限额时钟到期，软性限制将强行变为硬性限制。默认情况下，超越软性限制的宽限周期为 7 天时间。

repquota 和 quota 命令中的 grace 字段将给出宽限周期的设置。例如，假定用户的软性限制为 10 000 个数据块，硬性限制为 12 000 个数据块。如果已使用了 10 000 多个数据块，且 7 天的宽限周期也已到期，在清理文件直至其占用的空间低于软性设置之前，用户不能继续申请文件系统中的磁盘存储空间，分配更多的数据块。

## 3. 数据块与文件数量限制之间的差别

文件系统能够为用户提供两种资源：用于存储文件数据的数据块，用于创建文件的信息节点。每个文件均占用一个信息节点，而文件的实际数据将存储在数据块中。在限额管理中，数据块通常以 1KB 为单位。

假定不考虑目录，一个用户即使不占用任何数据块，如完全创建一系列空文件，也可能超过其信息节点的限制。另外，即使仅用一个信息节点，只需令文件大得足以耗尽用户限额规定的所有数据块数量，也可能超过其存储空间限额。

## 4. 设置限额的步骤

要设置文件系统的限额，可以参照下列步骤进行：

(1) 安装 quota 软件包。在 Ubuntu Linux 系统中，用作文件系统限额控制的软件包 quota 需要单独安装，为此，可使用软件维护工具，如 apt-get、aptitude 或 synaptic 等命令，安装 quota 软件包（参见第 12 章）。

(2) 设置文件系统，使之支持限额控制。为了在文件系统中设置限额，需要编辑/etc/fstab 文件，在文件系统的安装选项中增加限额标志“usrquota”，以确保每次引导系统时都会强制实行文件系统的限额控制。

(3) 此外，还需要在文件系统的顶级目录中创建一个空的 quota.user 或 aquota.user 文件（按用户定义限额），或者 quota.group 或 aquota.group 文件（按用户组定义限额）。

(4) 使用 edquota 命令，为每个用户设置磁盘空间和信息节点限额，包括软性限制、硬性限制和宽限周期。

(5) 在启用限额设置之前，使用 quotacheck 命令，检测每个文件系统的限额设置与实际使用情况是否一致，确保两者之间没有冲突。如果系统很少关机重启，需要周期地运行 quotacheck 命令。

(6) 使用 quotaon 命令启用文件系统的限额设置。在启用文件系统的限额控制之前，上述限额设置并不会立即发生作用。但是，如果已经适当地设置了/etc/fstab 文件，并在准备实行限额控制的文件系统根目录中创建了 aquota.user 或 aquota.group 文件之后，则每当重新引导系统，在安装了已经实行限额控制的文件系统之后，系统将会自动地启用限额控制。

## 5. 设置与维护限额的工具

表 11-10 给出了可用于设置和维护文件系统与用户限额的工具及其简单说明。



表 11-10

限额的设置与维护工具

命 令	基 本 功 能
edquota	调用默认的 vi 编辑器，设置、修改和关闭指定用户的磁盘空间与信息节点限额，包括硬性限制和软性限制
quotacheck	考察已安装的每个文件系统，检测文件系统的限额设置与实际使用情况的一致性，初始化限额数据库
quotaon	激活指定文件系统的限额设置与控制
quotaoff	关闭指定文件系统的限额设置与控制
quota	查询用户的限额设置和实际磁盘使用情况，检查是否存在超限的用户，也可据此确认限额是否已经被正确地设置
repquota	查询文件系统的限额设置，显示用户限额设置与使用情况（包括磁盘空间和文件数量）的汇总信息
setquota	命令行限额编辑器，可用于设置文件系统的限额

## 6. 设置限额的原则

在设置限额之前，系统管理员首先需要确定究竟应为每个用户分配多少磁盘空间和信息节点。如果想要绝对保证所有用户占用的磁盘总量不会超过文件系统的全部空间，可以按用户数平均分配每个用户的可用文件系统存储空间。例如，如果 4 个用户共享一个 1GB 的文件系统，且每个用户需要的磁盘空间大体相同，则可以按均分的方式为每个用户分配 256MB 的磁盘空间。

在一个实际的应用系统环境中，并非所有的用户都会同样地攀比使用或超限使用磁盘存储空间，因而可以分别为用户设置限额，并分配一个大于平均数的存储空间，使设定的全部用户存储空间大于文件系统的实际容量。例如，如果 4 个用户共享一个 1GB 的文件系统，则可以为每个用户分配 300MB 的磁盘空间。

### 11.4.2 设置限额

如前所述，为了在文件系统中实行限额控制，需要编辑/etc/fstab 文件，创建适当的限额文件，启用文件系统限额控制，以及设置用户限额等。

#### 1. 设置文件系统限额标志

编辑/etc/fstab 文件，对于准备实行限额控制的每一个文件系统，在其安装选项字段中增加一个“usrquota”标志，重新安装相应的文件系统或重新启动系统，以便使文件系统的限额设置发生作用。例如，要在“/”文件系统上实行限额控制，可修改/etc/fstab 文件，把下列两行

```
# /dev/sda7
UUID=73e5dfeb..... / ext3 relatime,errors=remount-ro 0 1
```

改为：

```
# /dev/sda7
UUID=73e5dfeb..... / ext3 relatime,errors=remount-ro,usrquota 0 1
```

#### 2. 创建限额文件

使用 touch 命令，在相应文件系统的根目录中创建一个空的 aquota.user 文件（此处假定按用户设定限额。如果按用户组设置限额，可以创建一个空的 aquota.group 文件），如下所示。

```
$ sudo touch /aquota.user
$
```

使用下列命令，修改新建限额文件的访问权限，使得只有超级用户才能够读写此限额文件。

```
$ sudo chmod 600 /aquota.user
$
```

#### 3. 维护限额文件

在系统的引导过程中，通过/etc/init.d/quota 启动脚本，系统将会自动地运行 quotacheck 命令。

`quotacheck` 命令用于考察每一个文件系统的使用情况，必要时创建、初始化限额文件，检测与维护文件系统的限额设置与限额文件的一致性，更新限额文件。

对于新增的空文件系统，即使设定了限额，通常也不必运行 `quotacheck` 命令。但是，如果在一个已经建有用户文件的文件系统中设置限额，则需要运行 `quotacheck` 命令，检查每个文件系统中用户已经占用的磁盘空间和信息节点，然后对限额文件进行同步与更新。而且还应记住，在较大的文件系统中运行 `quotacheck` 命令是一项非常耗时的处理任务。

限额初始化的最终目的是生成一个限额文件。限额文件（`aquota.user` 或 `aquota.group`）中包含了相应文件系统的用户限额设置、用户已经使用的存储空间，以及用户已经创建的文件或目录数量等信息。在用户的系统访问期间，Linux 系统会自动和透明地更新这个文件，并根据其中的限额设置，确定用户是否有权力使用更多的存储空间或创建更多的文件。

要初始化或更新限额文件，或者在之后定期地检测文件系统的限额设置与限额文件的一致性，可以使用下列 `quotacheck` 命令。

```
quotacheck [-vugfm] -a | filesystem
```

其中，“`-v`”选项意味着显示限额检测与维护的处理过程。“`-u`”和“`-g`”选项分别表示按用户和用户组检测与维护文件系统的限额设置。“`-f`”选项表示强制检测文件系统的限额设置。“`-m`”选项表示在检测限额设置时不要以只读方式重新安装文件系统。“`-a`”选项表示检测与维护 `/etc/fstab` 文件中具有限额标志（`usrquota`）的所有文件系统的限额设置。`filesystem` 表示仅检测与维护指定文件系统的限额设置。

下面的例子说明了如何初始化所有文件系统的限额设置。假定“/”文件系统是 `/etc/fstab` 文件中唯一具有限额标志的文件系统。

```
$ sudo quotacheck -vagum
quotacheck: Scanning /dev/sda7 [/] done
quotacheck: Old group file not found. Usage will not be subtracted.
quotacheck: Checked 24072 directories and 194960 files
$
```

注意，为了确保磁盘数据的准确性，在手工运行 `quotacheck` 命令时，应尽量设法使文件系统处于静止状态。

#### 4. 设置用户限额

在设置用户限额时，可以首先选定一个用户，参照下列步骤执行。

(1) 使用下列命令调用默认的 nano 编辑器，创建一个临时文件，对于实行了限额控制的每一个文件系统，`edquota` 命令将会自动增加一行限额信息。

```
edquota -u username
```

(2) 针对实行了限额控制的每个文件系统，把 1KB 数据块软性和硬性限制的默认值 0 改为一个适当的限额值，把信息节点软性和硬性限制的默认值 0 修改为一个适当的限额值。

(3) 使用下列命令，验证指定用户的限额是否已经适当地设置。其中，“`-v`”选项用于显示指定用户在实行了限额控制的文件系统中的限额信息。

```
quota -v username
```

假定整个系统中只有一个“/”文件系统实行了限额控制，利用“`edquota -u gqxing`”命令打开临时文件后，其初始数据内容如下。

```
$ sudo edquota -u gqxing
Disk quotas for user gqxing (uid 1000):
Filesystem    blocks   soft    hard   inodes   soft    hard
/dev/sda7     111516     0       0     2842      0       0
```



上述输出信息中的每个字段的意义说明如下。

- ❑ Filesystem 文件系统的安装点。
- ❑ blocks 用户当前使用的磁盘空间数量（以 1KB 数据块为单位）。
- ❑ inodes 用户当前已有的文件数量（当前信息节点的实际使用情况）。
- ❑ soft 数据块或信息节点数量限额的软性限制。当超越软性限额设置时，用户将会受到警告信息；超过宽限周期后如果一直未采取措施，系统将会禁止用户使用附加的磁盘空间与创建新的文件。
- ❑ hard 数据块或信息节点数量限额的硬性限制。如果这个数值为 0，则意味着未加限制。用户能够超过其软性限额设置，但不能超过其硬性限额设置。

利用 nano 编辑命令修改限额文件，把磁盘空间的软性和硬性限制分别设为 200 000 和 204 800 个 1kB 的数据块，信息节点的软性和硬性限制分别设为 4 000 和 4 096 个文件，保存限额设置后退出 edquota 命令。利用 quota 命令验证刚才的设置，其结果如下。

```
$ sudo quota -v gqxing
Disk quotas for user gqxing (uid 1000):
Filesystem      blocks      quota      limit      grace      files      quota      limit      grace
/dev/sda7        111516    200000    204800          2842        4000      4096
```

上述输出信息中的每个字段的意义说明如下。

- ❑ Filesystem 文件系统的安装点。
- ❑ blocks 当前实际使用的数据块数量。
- ❑ quota 数据块的软性限值。
- ❑ limit 数据块的硬性限制。
- ❑ grace 宽限周期。
- ❑ files 当前实际使用的信息节点数量。
- ❑ quota 信息节点的软性限制。
- ❑ limit 信息节点的硬性限制。

## 5. 为其他用户设置限额

在为选定的第一个用户设置了限额之后，可以此限额设置作为模板，利用下列 edquota 命令为其他用户设置限额。

```
edquota -p prototype-user username
```

其中，prototype-user 是用作限额设置模板的用户名。username 是尚未设置限额的其他用户名（如果同时指定多个用户名，用户名之间需加空格分隔符）。

下面的例子说明了怎样利用用户 gqxing 的限额设置模板，为 cathy、david 与 sarah 等其他用户设置限额。

```
$ sudo edquota -p gqxing cathy david sarah
$
```

## 6. 启用限额控制

要启用文件系统的限额控制，可以使用下列命令。

```
quotactl [-vugfp] filesystem
quotactl [-avugfp]
```

其中，“-v”选项意味着显示激活限额设置的每一个文件系统。“-a”意味着激活/etc/fstab 文

件中具有限额标志的所有文件系统的限额设置。“-p”选项表示显示文件系统限额控制的当前状态。filesystem 表示激活指定文件系统的限额设置。如果同时指定多个文件系统，文件系统名之间应加空格分隔符。

在安装 quota 软件包时，将会自动地启用文件系统的限额控制。下面的例子说明了必要时怎样激活“/”文件系统的限额设置。

```
$ sudo quotaon -v /
/dev/sda7 [/]: user quotas turned on
$
```

### 11.4.3 限额的维护

在设置限额、启用存储空间限额和信息节点限额控制之后，系统管理员可以检查超过限额设置的用户，检查文件系统的限额控制信息。在合理的情况下，通过调整用户能够继续使用的存储空间或信息节点数量，以修改和删除限额设置。如果必要，也可以关闭单个用户，甚至关闭整个文件系统的限额控制。

下面列出了文件系统限额维护的若干任务及其简单说明。

- 检测用户的限额设置。使用 quota 命令，针对已经实行限额控制的文件系统，检查指定用户的限额设置及其实际使用情况。
- 检测文件系统的限额设置。使用 repquota 命令，针对已经实行限额控制的文件系统，检查所有用户的限额及其实际使用情况。
- 修改宽限周期的默认值。使用 edquota 命令，针对已经超过存储空间和信息节点限额限制的用户，修改其剩余时间长度的限制。
- 修改用户的限额设置。使用限额编辑器 edquota，修改指定用户的限额控制。
- 关闭用户的限额设置。使用限额编辑器 edquota，关闭指定用户的限额控制。
- 关闭文件系统的限额设置。使用 quotaoff 命令，关闭指定文件系统的限额控制。

#### 1. 检测用户的限额设置

作为系统管理员，可以使用下列 quota 命令，针对已经实行限额控制的文件系统，检查指定用户的限额设置，确定是否存在超限使用的情况。

```
quota [-v] username
```

其中，“-v”选项表示显示指定用户在实行限额控制的文件系统中的限额，username 是用户名或用户 ID。

下面的例子说明，用户 cathy 的软性和硬性存储空间限额分别为 200 000 和 204 800 个 1KB 的数据块，目前已经使用了 11 200 个 1KB 的数据块。软性和硬性信息节点限额分别为 4 000 和 4 096 个，目前已经创建了 1 866 个文件。

```
$ sudo quota -v cathy
Disk quotas for user cathy (uid 1002):
Filesystem    blocks   quota   limit   grace   files   quota   limit   grace
/dev/sda7      11200   200000  204800
$
```

#### 2. 检测文件系统的限额设置

利用 repquota 命令，系统管理员可以检查所有用户的文件系统限额设置和存储空间的实际使用情况，如下所示。



```
repquota [-vug] -a | filesystem
```

其中，“-v”选项意味着显示所有用户的限额设置，包括没有占用任何资源的用户。“-a”选项意味着显示所有文件系统的限额设置。filesystem 表示仅显示指定文件系统的限额设置。

下面是一个执行 repquota 命令时的输出实例（系统中只有“/”文件系统实行了限额控制）。

```
$ sudo repquota -a
*** Report for user quotas on device /dev/sda1
Block grace time: 7days; Inode grace time: 7days
      Block limits          File limits
User      used     soft    hard grace     used   soft   hard grace
-----
root    -- 4183108      0      0      215901   0      0
...
gqxing  -- 111548  200000  204800      2847  4000  4096
cathy   -- 11200   200000  204800     1866  4000  4096
david   --      20  200000  204800       6  4000  4096
sarah   --      20  200000  204800       6  4000  4096
$
```

其中，“Block limits”部分各个字段的说明如下：

- used 当前实际使用的数据块；
- soft 数据块的软性限制；
- hard 数据块的硬性限制；
- grace 通常为空，当超过限额设置时，这个字段为宽限周期剩余的天数。

“File limits”部分各个字段的说明如下：

- used 当前实际使用的信息节点；
- soft 信息节点的软性限制；
- hard 信息节点的硬性限制；
- grace 通常为空，当超过限额设置时，这个字段为宽限周期剩余的天数。

第二列的两个字符位置分别表示是否超过数据块和信息节点的限额设置。其中，“--”表示两者均未超过限额设置，如果某一项超过了限额设置，相应的字符位置为加号“+”。

### 3. 修改限额的宽限周期

限额的宽限周期是在超过软性限制之后系统强行禁止用户继续使用系统资源的缓期时间。默认情况下，在超过软性限额但未超过宽限周期规定的一周时间内，用户能够超限使用文件系统的资源。超过宽限周期之后，系统将会禁止用户继续申请存储空间或信息节点。

如果需要，可以使用 edquota 命令和 vi 的编辑命令，把系统默认的宽限周期改为一个适当的时间限制。在修改宽限周期时，设定的时间限制可以包含一个数值和一个表示时间单位的关键字，如 days、hours、minutes 和 seconds，分别表示天、小时、分和秒等。注意，数字和关键字之间没有空格。

下面的“edquota -t”命令（“-t”选项表示编辑每个文件系统的宽限周期）给出了当前的宽限周期，其中的“7days”意味着系统默认的宽限周期为一周。

```
$ sudo 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/sda1            7days                  7days
```

增加宽限周期之后，用户能够继续超限使用存储空间或信息节点。注意，这个过程并不影响“当前已经超限的用户”。

#### 4. 修改与关闭用户的限额

如果限额设置不当，可能会影响用户的正常工作，必要时可以修改其限额。要修改指定用户的限额设置，可以使用 edquota 命令，针对实行了限额控制的每一个文件系统，按照软性和硬性限制，重新设定其 1KB 数据块的磁盘空间和信息节点的数量。

要关闭用户的限额设置，仍然可以使用 edquota 命令，针对已经实行了限额控制的每个文件系统，只需把其中已设定的软性和硬性磁盘空间限额，以及软性和硬性信息节点限额均设置为 0 即可。注意，最好不要删除相应的文本行。

#### 5. 关闭文件系统的限额设置

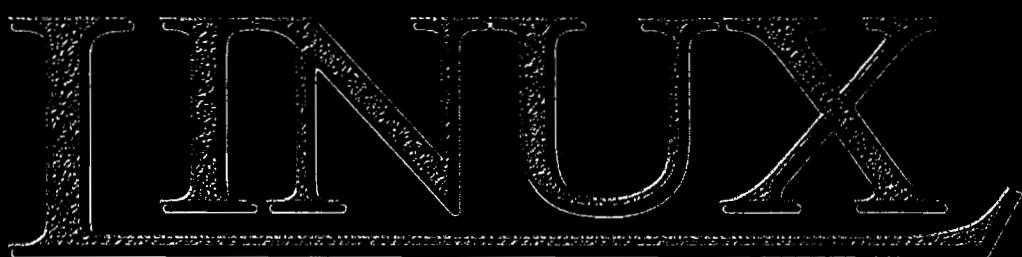
作为系统管理员，若要关闭文件系统的限额设置，可使用下列命令。

```
quotaoff [-vugp] filesystem
quotaoff [-avugp]
```

其中，“-v”选项意味着显示关闭限额设置的每一个文件系统，“-a”选项表示关闭所有文件系统的限额设置，filesystem 表示关闭指定文件系统的限额设置。如果同时给出多个文件系统，文件系统名之间应加空格分隔符。

下面的例子说明了怎样关闭“/”文件系统的限额设置。

```
$ sudo quotaoff -v /
/dev/sda7 [/]: user quotas turned off
$
```

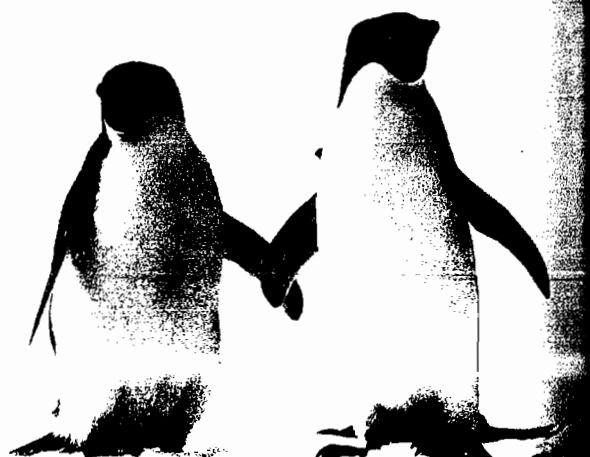


## 第12章

### 软件管理

本章主要介绍软件管理以及 Linux 系统的更新与升级，讨论怎样利用 Linux 系统提供的软件包管理工具，安装、更新、升级及查询系统软件。其内容主要包括：

- 软件管理概述；
- 利用 apt-get 管理软件包；
- 利用 aptitude 管理软件包；
- synaptic 图形界面软件管理工具；
- GNOME 软件增删工具；
- 软件包的自动更新。



# 12.1 软件管理概述

## 12.1.1 软件维护工具

Ubuntu Linux 系统提供了一系列命令行的软件维护工具，如 apt-get、aptitude 及 dpkg 等。这些软件维护工具各有侧重，但在安装软件包、更新以及升级 Linux 系统，确保系统总是处于最新状态方面，基本上具有等效的功能。而且，均能按照用户的意图，以网上发布的最新软件为准，安装或更新软件包及其依赖的相关软件。为了确保系统的安全性，增强系统的可靠性与可用性，定期地更新系统是一项必不可少的系统维护任务。

Ubuntu Linux 系统也提供了若干图形界面的软件维护工具，用以实现软件的安装、更新及系统升级。在 GNOME 桌面环境中，可以使用 gnome-app-install 与 synaptic。synaptic 是一个通用的图形界面软件维护工具，既可用于安装软件包，也可用于更新已安装的软件包，或者升级整个系统。gnome-app-install 主要用于补充安装尚未安装的 GNOME 软件包。

如果想要使用图形界面的软件维护工具 synaptic，可从 GNOME 桌面中选择“系统→系统管理→新立得软件包管理器”菜单，也可在命令行界面中输入 synaptic 命令（参见 12.4 节）。

要使用图形界面的软件维护工具 gnome-app-install，补充安装或删除 GNOME 软件包，可从 GNOME 桌面中选择“应用程序→添加/删除”菜单，也可在命令行界面中输入 gnome-app-install 命令（参见 12.5 节）。

此外，Ubuntu Linux 系统还提供一对软件更新工具——update-notifier 和 update-manager，用于检测系统配置的软件仓库中是否存在可用的软件包。当存在更新软件包时，将会在 GNOME 信息公告区显示一个软件更新图标（红色下箭头“↓”），同时在桌面上弹出一个“可用的软件更新”消息框，提醒用户更新自己的系统。如果更新软件包之后需要重新引导系统，信息通告区会出现一个圆形的双箭头图标，也会显示一个对话框，询问用户是否立即重新启动系统。此时可以酌情处理，如选择方便时再重新引导系统等（参见 12.6 节）。

## 12.1.2 软件管理基本概念

### 1. 软件包

在 Ubuntu Linux 系统中，所有的软件与文档都是以软件包档案文件的形式提供的。软件包可以分为二进制软件包（用于封装可执行程序、相关文档及配置文件等）和源代码软件包（其中主要包含源代码以及生成二进制软件包的制作方法）。

二进制软件包通常是一种压缩格式的档案文件，其中包含产品信息、程序文件、配置文件、随机文档、启动脚本及控制信息等，使用户能够容易地安装、更新、升级及删除软件（注意，若想查询一个软件包中究竟包含哪些文件，可以使用“dpkg -L *pkename*”命令）。

软件包管理工具可以利用其中的说明与控制信息，检索、安装、删除、更新或升级软件与系统。此外，除了基本的系统引导过程之外，Ubuntu 安装程序也是利用 Ubuntu 项目组提供的软件包，按照用户的要求安装 Linux 系统的。



按照封装格式，常见的 Linux 系统软件包可以分为下列 3 种类型（Ubuntu Linux 系统主要支持 Debian 格式的软件包）：

- Debian 格式软件包（文件扩展名为 “.deb”）——Debian 软件包是 Linux 系统中最常用的两种软件包之一，也是 Debian 及其派生 Linux 系统（包括 Ubuntu）主要支持的标准软件包。Ubuntu 软件仓库中提供的软件包均采用这种封装格式，apt-get、aptitude 及 synaptic 等软件管理工具也均支持此类的软件包。
- Red Hat 格式软件包（文件扩展名为.rpm）——RPM（Red Hat Package Manager）是另外一种流行的 Linux 系统软件包，也是 Red Hat 及其派生 Linux 系统（如 Fedora）支持的软件包。
- Tarball——是一种由大量文件（包括其目录结构）组装成单个档案文件的大型文件集合。其中，tar 命令用于组合多个文件，生成一个档案文件，以便于发行；gzip 命令用于压缩文件的容量，以便节省文件的存储空间。Tarball 非常类似于 Windows 系统中的“.zip”文件。Tarball 文件具有“.tar.gz”、“.tar.bz2”或 TGZ 形式的文件扩展名，可用于装配源代码或二进制代码文件。在开源社区中，Tarball 文件经常用于发行源代码，如果下载的文件包含一个.tar.gz 后缀，首先需要双击文件项或图标，解压相应的文件，然后才能安装其中包含的软件。在命令行终端窗口，可以使用“tar -xzf filename”命令解压相应的文件，然后再执行其中包含的软件安装命令。

软件包也包含数字签名，以证实其来源真实有效。软件包管理工具利用 GPG 公钥验证软件包数字签名的合法性。

## 2. 软件仓库

软件仓库指的是一个网站或存储目录，其中提供按一定组织形式存储的软件包与索引文件。利用软件仓库，软件维护工具能够检索与获取正确的软件包，完成软件包的安装，以及 Linux 系统的更新与升级。利用软件仓库，使用单个命令即可安装选定的软件包，更新所有的系统软件，或者找出匹配指定检索准则的软件包。

针对不同版本的 Ubuntu Linux 系统，Ubuntu 项目组提供数千个可直接安装的程序。这些程序以软件包的形式被存储在不同的软件仓库中，供用户利用 Internet 下载与安装，使用户能够非常容易地安装新的程序或更新现有的程序。此外，还提供诸如安装介质或映像以及源码等。

基于 Ubuntu 项目组能够提供的软件支持程度，以及是否遵循 Ubuntu 的自由软件理念，Ubuntu 的软件仓库主要分为以下 4 种类型（或渠道）。

- Main——Ubuntu 官方完全支持的软件，也是 Ubuntu Linux 系统的基本软件包，能够构成一个完整的 Linux 系统。其中包含了大多数流行的、稳定的、可以自由发布的开源自由软件。它也是安装 Ubuntu Linux 系统时自动安装的软件。
- Restricted——Ubuntu 支持的，但没有自由软件版权的通用软件。Ubuntu 项目组无权直接修改此类软件，因而也无法提供完全的技术支持，即使如此，Ubuntu 项目组还是会向作者反馈软件问题。
- Universe——由 Ubuntu 社区维护，Ubuntu 项目组不提供官方支持，只是为扩展与丰富 Linux 基本系统的功能与应用范围而提供的附加软件包。Universe 软件仓库包含数以千计的软件，是整个自由、开源 Linux 世界的缩影，从中可以找到大多数开源版权的软件。此类软件大都是基于 Main 中的组件编写的，因而能够与 Main 提供的软件相安无事地共

同运行，只是在稳定性与安全升级方面缺乏保障。

- Multiverse——“非自由软件”，这意味着其中的软件并不满足自由软件的版权政策。使用这些软件时，如何处理版权以及尊重版权所有者的劳动，完全依靠用户自己把握。这些软件通常较少修改和更新，须有用户自己承担任何风险。

Ubuntu 的 CD/DVD 安装映像主要包含来自 Main 和 Restricted 两个仓库中的软件，而且不完整，尤其是 CD，其中仅提供一小部分软件。在安装过程中，如果能够联网，可以在安装过程中补充安装诸如中文环境等更多的软件包。安装就绪之后，利用系统中已经安装的软件包管理工具，也可以直接检索、安装包括 Universe 或 Multiverse 在内的 4 个软件仓库中的任何软件包。

应当总是使用软件仓库，以便确保总能收到最新版本的软件。如果同一软件包存在多个可用的更新版本，apt-get、aptitude 或 synaptic 等软件维护工具能够自动地选用最新的软件版本。

### 3. 软件包的相互依赖关系

一个软件包是一个相对独立的基本功能单元，但大多数软件包通常都需要一定的底层支持，如函数库或底层协议支持等，而一个函数库或底层协议可能会同时支持多个软件包。当一个软件包需要某个特定的函数库或底层协议支持时，包含函数库或协议支持的软件包就是当前软件包依赖的软件包。要适当地安装一个软件包，必须首先满足其依赖关系。

apt-get 等软件维护工具使用软件包的依赖关系数据，确保在安装或更新期间能够安装必要的软件包，自动地安装系统中任何不存在的、与新装软件具有依赖关系的附加软件包，满足软件包的依赖关系。

## 12.2 利用 apt-get 管理软件包

APT (Advanced Package Tool) 是一个通用的综合软件管理与维护工具，功能完整，易于使用。apt-get、aptitude 和 synaptic 等绝大多数软件包工具都是基于 APT 及其配置文件发展起来的，是 APT 的前端软件管理工具。

早期，APT 的配置指令均被存储在一个单独的配置文件/etc/apt/apt.conf 中，Ubuntu Linux 系统把这个文件分解成多个小型文件，存储在/etc/apt/apt.conf.d 目录中。APT 的 cron 脚本被存储在 /etc/cron.daily 目录中，因此每天都会自动运行，读取/etc/apt/conf.d 目录中的配置文件，根据其中的配置要求，维护 APT 本地软件包索引文件以及 APT 本地缓存区。

/var/lib/apt/lists 目录存有 APT 本地软件包索引文件。对于/etc/apt/sources.list 文件中配置的每一个软件仓库，这个目录中均存在一个文件，其中列出了相应软件仓库中每一个软件包的最新版本信息。APT 使用这些文件确定软件包是否已经安装到本地系统，哪些软件包位于本地缓冲目录，软件包的最新版本是什么，等等。

/var/cache/apt/archives 目录是 APT 的本地缓冲目录，其中缓存了最近下载的 deb 软件包文件。通常，APT 的 cron 脚本将会限制这个目录占用的存储空间及文件的存放时间。

除了/etc/apt/sources.list 之外，APT 还使用/etc/apt/apt.conf.d 目录中的所有文件作为配置文件。稍后提及的 APT 配置参数就是指位于/etc/apt/apt.conf.d 目录各种 APT 配置文件中的配置参数，特此说明。通常，用户只需关注 sources.list 配置文件，不必考虑其他文件中的配置参数，故这里不



再赘述。

apt-get 是一个命令行软件管理工具，能够利用软件仓库安装选定的软件包，或者删除、更新系统中已经安装的软件包，升级 Linux 系统。apt-get 命令的语法格式简写如下。

```
apt-get [-hvs] [-o=config string] [-c=file]
        { [update] | [upgrade] | [dselect-upgrade] | [install pkgs] | [remove pkgs]
          | [purge pkgs] | [check] | [clean] | [autoclean] | [autoremove] }
```

apt-get 命令的选项分为功能选项与普通意义的选项，功能选项表示一个具体动作，如 install 表示安装，update 表示升级等。除了“-h”或“—help”等选项之外，apt-get 命令要求用户必须提供一个功能选项。表 12-1 给出了 apt-get 命令支持的部分功能选项及其简单说明。

表 12-1 apt-get 命令支持的部分功能选项

功 能 选 项	简 单 说 明
install pkgs	安装。用于安装指定的一个或多个软件包。在指定软件包时，只需给出简单的名字，不必写出软件包的完整名字，例如，对于软件包 libc6_1.9.6-2.deb，指定 libc6 即可。同时，apt-get 还会安装指定软件包依赖的所有底层支持软件包，以满足软件包的依赖关系。/etc/apt/sources.list 文件用于指定期望的软件源。如果没有恰好匹配的软件包，则假定指定的软件包名字是一个模式，apt-get 将会安装匹配指定模式的任何软件包。如果软件包名字后面附加一个减号“-”（中间没有空格）后缀，且软件包已经安装，apt-get 将会删除指定的软件包。类似地，如果软件包名字后面附有一个加号“+”后缀，表示安装指定的软件包。要选择安装一个特定版本的软件包，可在软件包名字后面附加一个“=version”后缀，如“aptitude install apt=0.3.1”。同样，要从一个特定的发行中选择一个软件包，可在软件包名字后面附加一个“/distribution”或“/archive”后缀，如 stable、testing 或 unstable 等。软件包的名字也可以看作一个表达式，如果没有软件包能够匹配给定的表达式，且表达式中包含句点“.”、问号“?”或星号“*”等特殊字符之一，则意味着这是一个正则表达式，因而可用之与软件仓库中的所有软件包进行比较，然后安装（或删除）与之匹配的任何软件包。注意，所谓匹配指的是字串意义上的匹配，因此，“lo.*”能够匹配“how-lo”和“lowest”。此外，除了上述 3 个特殊字符，还可以在正则表达式中使用上箭头“^”或美元符号“\$”等
update	更新。用于同步软件源的软件包索引文件，从/etc/apt/sources.list 文件中指定的软件源中获取可用软件包的索引。例如，当使用 deb 格式的软件包档案文件时，apt-get 命令将会检索 Packages.gz 文件，从中获取可用的新软件包或更新软件包的信息。因此，在利用 upgrade 或 dist-upgrade 功能选项升级整个系统之前，首先应当利用 update 功能选项，更新可用软件包的索引
upgrade	升级。从/etc/apt/sources.list 文件指定的软件源中，下载并安装比当前系统已安装的版本更新的所有软件包，但不会删除系统中已安装的软件包，也不会下载与安装系统中尚未安装的软件包。在执行系统升级之前，首先必须执行 update，更新软件包索引，以便 apt-get 能够知道是否存在可用的新版软件包
remove pkgs	删除。从系统中删除（卸载）指定的软件包，同时删除依赖于指定软件包的其他软件包。除了删除软件包，remove 还等同于 install 功能选项。例如，如果指定的软件包名之后附加一个加号“+”（中间没有空格），将会安装而不是删除指定的软件包
autoremove	自动删除。用于删除为满足依赖关系而自动安装的，且当前不再需要的软件包
purge pkgs	消除。除了彻底清除软件包提供的配置文件等之外，其功能等同于 remove 功能选项
check	诊断。用以更新软件包缓冲区，检测软件包的依赖关系是否存在
clean	消除。用于消除缓存存在本地目录中的软件包文件等。除了位于 /var/cache/apt/archives 和 /var/cache/apt/archives/partial 目录中的封锁文件，clean 功能选项将会消除软件包的任何文件。当以 dselect 方法运行 APT 软件包管理工具时，将会自动地执行清除功能。在不采用 dselect 方法维护软件包时，应注意随时运行“apt-get clean”命令，以释放磁盘空间
autoclean	类似于 clean，autoclean 也用于消除缓存存在本地目录中的软件包文件等。其差别在于后者仅删除不再继续下载且基本上不再继续使用的软件包文件。这将防止缓存空间由于长期没有清空时导致的增长失控。如果把配置参数“APT::Clean-Installed”设置为 off，将会防止删除已经安装的软件包
check	诊断。用以更新软件包缓冲区，检测受损的软件包依赖关系
dist-upgrade	除了执行 upgrade 的功能之外，dist-upgrade 还能够智能地处理由新版软件包导致的依赖关系变化。apt-get 具有一个“智能的”冲突解决机制，如果需要，它将会尝试优先升级最重要的软件包。/etc/apt/sources.list 配置文件包含一系列软件源的定义，使 apt-get 能够获取期望的软件包

表 12-2 给出了 apt-get 命令支持的其他选项及简单说明。

表 12-2 apt-get 命令支持的其他选项

选项	GNU 选项	简 单 说 明
-h	--help	显示简明的帮助信息，然后退出 apt-get
-c file	--config-file file	指定 apt-get 命令使用的，除默认配置文件之外的其他配置文件，其中包含软件仓库的 http、ftp、cdrom 及本地文件的地址或路径
-y	--yes, --assume-yes	对于需要用户确认是否 (yes/no) 的所有请求，总是使用 yes 作为回答。这意味着采用非交互式的方式自动运行 apt-get 命令。相应的 APT 配置参数为 “APT::Get::Assume-Yes”
	--no-download	禁止下载软件包。最好与 “--ignore-missing” 一同使用，以便强制 APT 完全使用已下载的、缓存的 .deb 软件包，执行软件包的安装与更新。相应的配置参数为 “APT::Get::Download”
-d	--download-only	仅下载软件包，既不解压，也不安装软件包。相应的 APT 配置参数为 “APT::Get::Download-Only”
	--purge	在删除软件包时，可以使用 purge 替代 remove 功能选项。对于准备消除的软件包，apt-get 将会在软件包后面附加一个星号 (*) 标记。相应的 APT 配置参数为 “APT::Get::Purge”
	--reinstall	对于当前已经安装的软件包，重新安装最新版本的软件包。相应的 APT 配置参数为 “APT::Get::ReInstall”
	--allow-unauthenticated	无需考虑软件包是否已经认证，即使软件包未认证，也不输出任何提示信息。相应的配置参数为 “APT::Get::AllowUnauthenticated”
-u	--show-upgraded	显示已经升级的所有软件包列表。其相应的 APT 配置参数为 “APT::Get::Show-Upgraded”
	--no-remove	如果需要删除任何软件包，apt-get 将会立即停止运行，而且不会给出任何提示信息。相应的配置参数为 “APT::Get::Remove”
	--auto-remove	如果 apt-get 命令的功能选项是 install 或 remove，这个选项的作用相当于执行 autoremove 功能选项，即删除未用的、当前软件包依赖的软件包。相应的 APT 配置参数为 “APT::Get::AutomaticRemove”
	--no-upgrade	禁止升级软件包。与 install 功能选项一起使用时，如果命令行中指定的软件包已经安装，这个选项将会禁止升级指定的软件包。相应的 APT 配置参数为 “APT::Get::Upgrade”
-q	--quiet	安静模式。生成适合于日志的输出信息，忽略进度指示信息。相应的 APT 配置参数为 “quiet”

要使用 apt-get，可从表 12-1 中列举的功能选项中选择其中的一个，同时再具体指定准备安装或删除的软件包。对于每一个处理请求，apt-get 将会从配置的软件仓库中下载最新的软件包信息，然后检索下载的数据文件，确定需要采取的下一步处理动作，其中包括为满足软件包的依赖关系，是否需要安装、更新或删除其他软件包等。

### 12.2.1 安装软件包

install 功能选项主要用于安装指定的软件包。在正式开始下载与安装指定及其相关的软件包之前，apt-get 将会提请用户予以确认。如果没有异议，可输入 “y”，接着按 Enter 键表示确认，然后即可开始下载与安装；如果不同意，可以输入 “n” 接着按 Enter 键终止下载，从而终止安装指定的软件包。在安装过程中，apt-get 首先会依次下载每一个软件包，下载后再一一解压，然后替换之前安装的软件包，执行安装后的设置。

在安装指定的软件包时，如果其依赖的软件包尚未安装，apt-get 将会尝试从软件仓库中检索、下载必要的软件包（如库函数或通用软件），以满足软件包的相互依赖关系。在此情况下，apt-get



仍会提请用户予以确认。

例如，在利用 CD/DVD 介质安装 Ubuntu Linux 系统时，安装程序不会安装文件系统限额软件包 quota。要安装 quota 软件包，可以使用下列 apt-get 命令。

```
$ sudo apt-get install quota
Reading package lists... Done
Building dependency tree
Reading state information... Done
Suggested packages:
  libnet-ldap-perl
The following NEW packages will be installed:
  quota
0 upgraded, 1 newly installed, 0 to remove and 150 not upgraded.
Need to get 456kB of archives.
After this operation, 1376kB of additional disk space will be used.
Get:1 http://cn.archive.ubuntu.com hardy/main quota 3.15-6 [456kB]
Fetched 456kB in 42s (10.7kB/s)
Preconfiguring packages ...
Selecting previously deselected package quota.
(Reading database ... 181680 files and directories currently installed.)
Unpacking quota (from .../archives/quota_3.15-6_i386.deb) ...
Setting up quota (3.15-6) ...
$
```

当因故需要重新安装某个软件包时，可以利用 apt-get 命令的“—reinstall”选项与 install 功能选项。例如，要重新安装 quota 软件包，可以使用下列 apt-get 命令。

```
$ sudo apt-get --reinstall install quota
Reading package lists... Done
Building dependency tree
Reading state information... Done
0 upgraded, 0 newly installed, 1 reinstalled, 0 to remove and 0 not upgraded.
Need to get 0B/456kB of archives.
After this operation, 0B of additional disk space will be used.
Do you want to continue [Y/n]? y
Preconfiguring packages ...
(Reading database ... 183152 files and directories currently installed.)
Preparing to replace quota 3.15-6 (using .../archives/quota_3.15-6_i386.deb) ...
 * Stopping quota service rpc.rquotad                                [ OK ]
Unpacking replacement quota ...
Setting up quota (3.15-6) ...
$
```

## 12.2.2 系统的更新与升级

为了全面更新或升级当前的 Ubuntu Linux 系统，首先需要使用 apt-get 命令的 update 功能选项，同步软件源的软件包索引文件，获取最新的可用软件包版本信息。然后，再利用 upgrade 功能选项下载与安装新版的软件包。因此，在利用 apt-get 命令升级整个 Ubuntu Linux 系统时，首先应运行下列 apt-get 命令，确保软件包索引文件是最新的。

```
$ sudo apt-get update
[sudo] password for gqxing:
Get:1 http://security.ubuntu.com hardy-security Release.gpg [189B]
Get:2 http://security.ubuntu.com hardy-security Release [58.5kB]
...
Get:10 http://security.ubuntu.com hardy-security/multiverse Sources [14B]
Hit http://cn.archive.ubuntu.com hardy Release.gpg
Hit http://cn.archive.ubuntu.com hardy-updates Release.gpg
...
Hit http://cn.archive.ubuntu.com hardy-updates/multiverse Sources
Fetched 195kB in 1min21s (2398B/s)
Reading package lists... Done
$
```

然后，运行下列 apt-get 命令，下载、安装新版的软件包，从而升级整个系统。

```
$ sudo apt-get upgrade
```

系统的升级实际上是一个软件包的删除与重装过程，作为软件包更新与升级过程的一部分，apt-get 将会自动地删除老的软件包。

对于系统内核软件包而言，作为一个备份，更新之前的系统内核软件包将会继续保留在系统中，以便在使用新的系统内核引导系统出现故障时，用户能有机会选用之前的系统内核继续引导系统。

### 12.2.3 删除软件包

利用 apt-get 命令的 remove 与 purge 功能选项，可以删除指定的软件包。注意，remove 与 purge 功能选项的主要作用都是删除指定的软件包，但两者之间的不同之处在于：如果软件包中含有配置文件，remove 功能选项在删除软件包时会保留配置文件，因此可称作部分删除；而 purge 功能选项将会删除整个软件包，包括配置文件，故可称作彻底删除。

在正式开始删除软件包之前，apt-get 首先会考察系统中是否存在依赖于指定软件包的相关软件包。如果存在，apt-get 会列出需要同时删除的其他软件包，提请用户确认。例如，为了从系统中删除不再继续使用的软件包 quota，可以使用下列 apt-get 命令。

```
$ sudo apt-get remove quota
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following packages will be REMOVED:
  quota
  0 upgraded, 0 newly installed, 1 to remove and 0 not upgraded.
After this operation, 1376kB disk space will be freed.
Do you want to continue [Y/n]? y
(Reading database ... 200852 files and directories currently installed.)
Removing quota ...
  * Turning off quotas...                                [ OK ]
  * Stopping quota service rpc.rquotad                [ OK ]
$
```

### 12.2.4 安装本地存储介质中的软件包

通常情况下，应尽量使用 Ubuntu 项目或本地镜像网站提供的软件仓库和标准的 apt-get 等软件管理工具下载、安装以及更新软件包。如果没有可用的软件仓库，可以利用 CD/DVD 安装介质或存储在本地系统的软件包档案文件，安装尚未安装的软件包。

在安装 CD/DVD 或存储在本地系统中的软件包时，需要事先配置 sources.list 文件（参见 12.2.5 小节）。

### 12.2.5 sources.list 配置文件

#### 1. 简介

由于 apt-get、aptitude 和 synaptic 等大部分软件包维护工具均以 APT 作为底层支撑，而 APT 又主要使用/etc/apt/sources.list 配置文件定义软件的发行源，故本小节主要介绍 sources.list 配置文件。在安装了 Ubuntu Linux 系统之后，系统将会创建一个默认的配置文件，使 APT 能够根据 sources.list 配置文件的设置，利用 Ubuntu 项目或镜像网站提供的软件仓库，执行软件包的安装、更新与升级等。

除了 sources.list 主配置文件之外，也可以在/etc/apt/sources.list.d 目录中定义其他辅助配置文件，作为 sources.list 配置文件的补充。辅助配置文件定义软件源的语法格式以及文件扩展名同



sources.list 主配置文件。

要增加、修改软件仓库，或者改变 APT 的默认处理动作，如增加本地软件仓库的镜像站点，通常需要修改/etc/apt/sources.list 配置文件。在修改 sources.list 配置文件或软件仓库配置文件时，可以使用任何文本编辑器。

sources.list 配置文件用于定义各种可用的软件源，如网站、镜像站点、CD/DVD 及本地的存储目录等。在 sources.list 配置文件中，每行只能定义一个软件源，其语法格式如下。

```
type uri distribution [component1] [component2] [...]
```

其中，“type”表示软件包档案文件的类型，Ubuntu Linux 系统支持的软件包档案文件类型主要有 deb（二进制代码）与 deb-src（源代码）两种。“uri”用于定义软件源的地址类型。“distribution”通常可以是 stable、unstable 或 testing 等表示软件版本类型的关键字之一。每个“component”可以是 main、contrib、non-free 或 non-us 等表示软件仓库类型的关键字之一。

在定义 deb 类型的软件包时，必须在 uri 字段中指定软件源的主要发行版本（基准存储位置），以便 APT 能够从中找出其需要的信息。存储位置可以是一个路径名，如果路径名以斜线字符“/”结束，必须省略后续的软件仓库定义部分。

主要发行版本的基准存储位置可以包含一个\$ (ARCH) 变量，以便 APT 能够在运行过程中替换成符合本地系统的机型，如 i386、m68k 或 powerpc 等。这种定义方式允许设定一个通用的 sources.list 配置文件。

在 sources.list 配置文件中，每行只能定义一个软件源，故对同一个 URI 而言，如果其中存在不同类型的软件包档案文件，可能需要提供多个软件源定义。在读取所有的软件源定义之后，APT 将会对所有的 URI 进行排序，把相同的 URI 合并为对同一个主机的引用，建立单一的网络连接。APT 也会并发地连接到不同的主机，以便能够更有效地传输文件。

在 sources.list 配置文件中，定义软件源的优先顺序非常重要。通常应把最常用的、速度最快的软件源放在最前面，例如，通常应把 CD/DVD、本地主机以及速度快的本地镜像网站依序排列在前面。

sources.list 配置文件当前支持的 URI 地址类型包括 file、cdrom、http 和 ftp 等，分别简述如下。

- file——表示可使用文件系统中的任何目录作为软件包的软件源。这对于采用 NFS 或本地镜像方式安装 Ubuntu Linux 系统或软件包是非常有用的。
- cdrom——表示使用 CD/DVD 作为软件包的软件源。
- http——用于指定一个 HTTP 服务器，其中提供软件包的档案文件。
- ftp——用于定义一个 FTP 服务器，其中提供软件包的档案文件。
- copy——除了把软件包复制到本地系统的缓存目录，而不是在原目录位置直接使用之外，copy 完全等同于 file。对于使用 zip 磁盘作为软件源的用户而言，这种地址类型是非常有用的。
- rsh/ssh——表示采用 rsh/ssh 方法连接远程主机，以预定的用户身份访问其中的软件包档案文件。采用此方式时，不能出现密码认证过程，因此，必须事先配置好 RSA 密钥或 rhosts 主机认证。在访问远程文件时，采用标准的 find 与 dd 命令，实现本地与远程主机系统之间的文件传输。

## 2. 示例

当本地（包括采用 NFS 安装的远程资源）存有可安装的软件包时，可以选用 file 作为 URI 地址类型，指定软件仓库的位置。例如，假定/opt/debian 目录中存有稳定的软件发行，如 stable/main、

stable/contrib 及 stable/non-free3 种类型的软件包时，可以在 sources.list 文件中增加下列定义。

```
deb file:/opt/debian stable main contrib non-free
```

如果想要使用 CD/DVD 作为软件源，可以采用 cdrom 作为 URI 地址类型，指定软件存储的目录位置。以 8.10 版本的 Ubuntu Linux 系统的 DVD 安装介质为例，可以在 sources.list 文件中增加下列软件源定义（如果不知道怎样在文件中增加 CD/DVD 软件源定义，可以运行 apt-cdrom 命令）。

```
deb cdrom:[Ubuntu 8.10 _Intrepid Ibex_ - Release i386 (20081030)]/ intrepid main restricted
```

要使用 FTP 协议，访问存储在 ftp.debian.org 网站 debian 目录中的 stable/contrib 类型的软件包档案文件，可以在 sources.list 文件中增加下列定义。

```
deb ftp://ftp.debian.org/debian stable contrib
```

如果想要使用 HTTP 协议，访问 cn.archive.ubuntu.com 网站 ubuntu 目录中的软件包档案文件，可以在 sources.list 文件中增加下列定义。

```
deb http://cn.archive.ubuntu.com/ubuntu/ hardy main restricted
```

下面是在安装了 Ubuntu 8.10 Linux 系统之后，系统提供的 sources.list 配置文件。

```
$ cat /etc/apt/sources.list
#deb cdrom:[Ubuntu 8.10 _Intrepid Ibex_ - Release i386 (20081030)]/ intrepid main
restricted
# See http://help.ubuntu.com/community/UpgradeNotes for how to upgrade to
# newer versions of the distribution.
deb http://cn.archive.ubuntu.com/ubuntu/ intrepid main restricted
deb-src http://cn.archive.ubuntu.com/ubuntu/ intrepid main restricted
## Major bug fix updates produced after the final release of the
## distribution.
deb http://cn.archive.ubuntu.com/ubuntu/ intrepid-updates main restricted
deb-src http://cn.archive.ubuntu.com/ubuntu/ intrepid-updates main restricted
## N.B. software from this repository is ENTIRELY UNSUPPORTED by the Ubuntu
## team. Also, please note that software in universe WILL NOT receive any
## review or updates from the Ubuntu security team.
deb http://cn.archive.ubuntu.com/ubuntu/ intrepid universe
deb-src http://cn.archive.ubuntu.com/ubuntu/ intrepid universe
deb http://cn.archive.ubuntu.com/ubuntu/ intrepid-updates universe
deb-src http://cn.archive.ubuntu.com/ubuntu/ intrepid-updates 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://cn.archive.ubuntu.com/ubuntu/ intrepid multiverse
deb-src http://cn.archive.ubuntu.com/ubuntu/ intrepid multiverse
deb http://cn.archive.ubuntu.com/ubuntu/ intrepid-updates multiverse
deb-src http://cn.archive.ubuntu.com/ubuntu/ intrepid-updates multiverse
.....
deb http://security.ubuntu.com/ubuntu intrepid-security main restricted
deb-src http://security.ubuntu.com/ubuntu intrepid-security main restricted
deb http://security.ubuntu.com/ubuntu intrepid-security universe
deb-src http://security.ubuntu.com/ubuntu intrepid-security universe
deb http://security.ubuntu.com/ubuntu intrepid-security multiverse
deb-src http://security.ubuntu.com/ubuntu intrepid-security multiverse
$
```

## 12.3

### 利用 aptitude 管理软件包

在 Ubuntu Linux 系统中，aptitude 是一个完全可以替代 apt-get 的软件管理工具，两个命令中



的大多数功能选项与参数也是完全兼容的。aptitude 命令可用于安装、查询或删除软件包，以及升级整个 Ubuntu Linux 系统，其语法格式简写如下。

```
aptitude [options] { update | autoclean | clean | safe-upgrade )
aptitude [options] { install | reinstall | full-upgrade | download
| purge | remove | show } pkgs
aptitude [options] search patterns
aptitude help
```

其中，pkgs 为软件包档案文件的名字。在运行 aptitude 命令时，必须选择一个具体的操作，如 install（安装）、update（更新）、upgrade（升级）、show（查询）、remove（删除）及 search（检索）等，参见表 12-3。考虑到实用性，这里主要介绍软件包的安装、查询、删除与检索，以及系统的升级等功能。

表 12-3

aptitude 命令的部分选项

功 能 选 项	简 单 说 明								
install pkgs	<p>安装指定的软件包。如果指定的软件包名字中包含波浪号“~”，则表示这是一个检索模式，匹配检索模式的所有软件包都会被安装。要安装一个特定版本的软件包，可以在软件包的名字后面附加一个“=version”后缀，如“aptitude install apt=0.3.1”。类似地，要从一个特定的档案文件中选择一个软件包，可以在软件包的名字后面附加一个“/archive”后缀，如“aptitude install apt/experimental”。在 aptitude 命令行中，如果在指定的软件包名字后面附加一个“强制修饰符”，可以对相应的软件包执行不同的处理动作。例如，“aptitude install quota-”命令表示删除而非安装 quota 软件包。aptitude 命令支持下列强制修饰符：</p> <ul style="list-style-type: none"> <li><input type="checkbox"/> &lt;package&gt;+ 安装指定的软件包；</li> <li><input type="checkbox"/> &lt;package&gt;+M 安装指定的软件包，并立即增加一个自动安装标记（注意，如果没有任何软件包依赖于指定的软件包，将会立即删除指定的软件包）；</li> <li><input type="checkbox"/> &lt;package&gt;- 删除指定的软件包；</li> <li><input type="checkbox"/> &lt;package&gt;_ 清除指定的软件包（删除指定的软件包及有关的配置文件和数据文件）；</li> <li><input type="checkbox"/> &lt;package&gt;= 保持指定的软件包，取消任何安装、删除或升级动作，防止将来自动升级指定的软件包；</li> <li><input type="checkbox"/> &lt;package&gt;: 保持当前版本的软件包不变，取消任何安装、删除或升级动作，但不能阻止将来的自动升级；</li> <li><input type="checkbox"/> &lt;package&gt;&amp;M 把指定的软件包标记为自动安装的软件包；</li> <li><input type="checkbox"/> &lt;package&gt;&amp;m 把指定的软件包标记为手工安装的软件包。</li> </ul> <p>注意，一旦在最终的确认提示下输入了“Y”，“aptitude install”命令将会修改 aptitude 保存的动作执行信息。因此，如果输入了“aptitude install foo bar”命令，一旦在下载和安装过程中间终止了安装软件包，就需要运行“aptitude remove foo bar”命令，以取消前者造成的一致性问题</p>								
update	从/etc/apt/sources.list 配置文件定义的软件源中更新可用的软件包索引，其功能等价于“apt-get update”命令								
safe-upgrade	对系统中已经安装的软件包，升级到最新的版本。除非不再继续使用，否则不会删除原先已经安装的软件包。注意，这一功能选项也不会安装之前尚未安装的软件包。与 full-upgrade 相比，safe-upgrade 相对更保守一些。要升级或安装更多的软件包，可以使用 full-upgrade 功能选项								
full-upgrade (dist-upgrade) pkgs	对已经安装的软件包，升级到最新的版本，删除或安装必要的软件包。与 safe-upgrade 相比，full-upgrade 功能选项有可能执行一些不需要的动作，但能够升级或安装一些 safe-install 功能选项无法升级或安装的软件包								
reinstall pkgs	重新安装指定的软件包								
remove pkgs	删除指定的软件包								
purge pkgs	彻底清除指定的软件包								
search pattern	<p>检索匹配命令行指定模式的所有软件包。通常，“aptitude search”命令的输出形式如下所示：</p> <table style="margin-left: 20px;"> <tr><td>i apt</td><td>- Advanced front-end for dpkg</td></tr> <tr><td>pi apt-build</td><td>- frontend to apt to build, optimize and in-</td></tr> <tr><td>cp apt-file</td><td>- APT package searching utility -- command-</td></tr> <tr><td>i A mysql-server-5.0</td><td>- MySQL database server binaries</td></tr> </table> <p>在 search 功能选项的输出信息中，每行显示一个检索结果。其中，第一列中的第一个字符表示软件包的当前安装状态，其中最常见的安装状态是“p”，意味着系统中不存在相应的软件包记录；“c”表示相应的软件包已经删除，但其配置文件仍留存在系统中；“i”表示相应的软件包已经安装；“v”表示虚拟软件包。如果存在，第二个字符表示软件包的存储动作，其中最常见的存储动作是“i”（意味着将要安装的软件包）、“d”（意味着将要删除的软件包）和“p”（意味着将要删除的软件包，包括其配置文件）。如果存在第 3 个字符“A”，表示相应的软件包将会自动地安装（有关完整的安装状态与动作标志的说明，详见 Aptitude 参考手册中的“Accessing Package Information”一节）</p>	i apt	- Advanced front-end for dpkg	pi apt-build	- frontend to apt to build, optimize and in-	cp apt-file	- APT package searching utility -- command-	i A mysql-server-5.0	- MySQL database server binaries
i apt	- Advanced front-end for dpkg								
pi apt-build	- frontend to apt to build, optimize and in-								
cp apt-file	- APT package searching utility -- command-								
i A mysql-server-5.0	- MySQL database server binaries								

续表

功能选项	简单说明
show pkgs	显示指定软件包的详细信息。如果指定的软件包名字中包含一个波浪号“~”，将会把软件包名看作检索模式，显示与之匹配的所有软件包信息。如果命令行中提供一个或多个“-v”选项，将会显示所有版本的软件包信息；否则，仅显示“候选版本的软件包”，即“aptitude install”命令可能会下载的那个版本的软件包信息。如果在软件包的名字后面附加一个“=version”后缀，可以显示指定版本的软件包信息。类似地，在软件包的名字后面附加一个“/archive”后缀，可以显示一个特定档案文件中的软件包信息。此外，如果命令行中提供一个或多个“-v”选项，将会显示所有版本的软件包信息，显示软件包的适用机型、压缩后的文件大小以及文件名等信息
clean	从软件包缓存目录（/var/cache/apt/archives）中删除先前下载的所有软件包文件
autoclean	从软件包缓存目录（/var/cache/apt/archives）中删除不再继续下载的任何软件包文件。这将防止在长期没有清空缓存空间时导致其增长失控
download pkgs	下载指定的软件包文件，并存储在当前目录中。通常，aptitude 将会下载与“aptitude install”命令安装版本相同的软件包。如果在软件包名字后面附加一个“=version”后缀，可以下载指定版本的软件包。如果在软件包名字后面附加一个“/archive”后缀，还可以从指定的档案文件中下载相应版本的软件包
help	显示 aptitude 命令的简要使用说明

在 aptitude 命令的安装、升级及更新等功能选项中，还可以选用部分普通选项，以限定功能选项的处理动作，详见表 12-4。

表 12-4 aptitude 命令部分可用的普通选项

选项	GNU 选项	简单说明
-h	—help	显示 aptitude 命令的简要使用说明。等同于 help 功能选项
	—purge-unused	清除已安装的任何软件包都不再需要的软件包。这等价于在命令行中加入了“-o Aptitude::Purge-Unused=true”选项及其参数
-D	—show-deps	对于软件包的安装（如 install、full-upgrade）、删除（如 remove、purge）等命令，这个选项表示输出自动安装和删除的简要说明信息。相应的配置参数为“Aptitude::CmdLine::Show-Deps”
-d	—download-only	仅把软件包下载到本地缓存目录，而不执行任何安装或删除动作。默认情况下，软件包的缓存目录为 /var/cache/apt/archives。相应的配置参数为“Aptitude::CmdLine::Download-Only”
-y	—assume-yes	对于需要用户予以确认（yes/no）的所有请求，总是使用 yes 作为回答。这也意味着采用非交互方式自动运行 aptitude 命令，执行软件包的安装、升级或删除
-Z		对于安装、升级或删除的软件包，相应地显示每个软件包占用或释放的磁盘空间。相应的配置参数为“Aptitude::CmdLine::Show-Size-Changes”
-v	—verbose	输出 aptitude 命令运行过程的说明信息，这个选项能够使部分功能选项（如 show）显示更多的信息。多个“-v”选项表示提供更多的信息

### 12.3.1 安装软件包

install 功能选项用于安装指定的软件包。因此，要安装尚未安装的软件包，可以使用下列 aptitude 命令。

```
$ sudo aptitude install quota
```

如同 apt-get 命令一样，aptitude 能够在安装软件包的同时，自动解决软件包的相互依赖关系问题。如果准备安装的软件包所依赖的软件包尚未安装，aptitude 将会在提请用户确认之后同时下载并安装相关的软件包。

如果因故需要重新安装一个已经安装的软件包，如误删了其中的文件，或者要恢复初始的配



置文件，可以使用 aptitude 命令的 reinstall 功能选项。例如，使用下列 apt-get 命令，可以重新安装 quota 软件包。

```
$ sudo aptitude reinstall quota
```

### 12.3.2 系统的升级

aptitude 的升级功能基本上类似于 apt-get 命令。例如，要全面升级当前的 Ubuntu Linux 系统，首先需要使用下列 aptitude 命令，同步软件源的软件包索引文件，获取最新的可用软件包版本信息。

```
$ sudo aptitude update
```

然后使用下列命令，下载与安装所有需要更新的软件包，从而升级整个系统。

```
$ sudo aptitude safe-upgrade
```

### 12.3.3 查询软件包

aptitude 命令的“show”功能选项用于查询各种软件包信息。例如，利用下列 aptitude 命令，可以查询软件包的安装状态、版本、依赖关系、档案文件的大小以及简单说明等信息。

```
$ aptitude show vsftpd
Package: vsftpd
State: installed
Automatically installed: yes
Version: 2.0.6-1ubuntu1.1
Priority: extra
Section: net
Maintainer: Ubuntu Core Developers <ubuntu-devel-discuss@lists.ubuntu.com>
Uncompressed Size: 401k
Depends: adduser, libc6 (>= 2.4), libcap1, libpam-modules, libpam0g (>=
          0.99.7.1), libssl0.9.8 (>= 0.9.8f-1), libwrap0, lsb-base (>=
          1.3-9ubuntu3), netbase, ssl-cert (>= 1.0-11ubuntu1), sysv-rc (>=
          2.86.ds1-14.1ubuntu2), update-inetd
Recommends: logrotate
Conflicts: ftp-server
Provides: ftp-server
Description: The Very Secure FTP Daemon
A lightweight, efficient FTP server written from the ground up with security
in mind.

vsftpd supports both anonymous and non-anonymous FTP, PAM authentication,
bandwidth limiting, and the Linux sendfile() facility.
$
```

### 12.3.4 检索软件包

在安装软件包时，可以事先利用 aptitude 命令的 search 功能选项，从系统缓存的软件包索引文件或/etc/apt/sources.list 配置文件设定的软件仓库中检索可用的软件包，也可以检索系统中已经安装的软件包。aptitude 检索功能的语法格式简写如下。

```
aptitude search pkg-pattern
```

其中，pkg-pattern 是软件包的名字或名字模式。在指定软件包的名字时，可以指定一个确切的名字或部分名字，指定软件包简述中的一个关键字，也可以指定一个特殊的名字模式。实际上，在指定 search 功能选项的软件包检索模式时，aptitude 将会按照包含的原则进行检索，给出匹配检索模式的所有软件包。

软件包的检索结果取决于检索模式，如果检索结果为空，即 aptitude 命令没有输出任何信息，则意味着没有匹配检索模式的软件包。

如果已经知道软件包的准确名字，可以按照名字查询具体的软件包。例如，要查询 gawk 软件包的相关信息，可以使用下列 aptitude 命令。

```
$ aptitude search gawk
i gawk           - GNU awk, a pattern scanning and processing
p gawkdoc        - Documentation for GNU awk
$
```

在 search 功能选项的输出信息中，每一行显示一个检索结果。其中，第一列表示软件包的安装与存储状态（参见表 12-3），第二列是软件包的名字，第 3 列是软件包的简单说明。

aptitude 命令的 search 选项检索功能非常强，除了能够按照指定的名字模式检索软件包之外，aptitude 还可以根据软件包检索的特点，提供一些特殊的检索模式。表 12-5 给出了 Ubuntu Linux 系统中可用的部分特殊检索模式。

表 12-5 aptitude 命令可用的部分特殊检索模式

特殊检索模式	简 单 说 明
<code>~aaction</code>	检索已经标记做指定安装动作的软件包。安装动作可以是 <code>install</code> 、 <code>upgrade</code> 、 <code>downgrade</code> 、 <code>remove</code> 、 <code>purge</code> 、 <code>hold</code> （可用于测试软件包是否需要保持固定不变）或 <code>keep</code> （可用于测试软件包是否需要保持不变）等关键字之一
<code>'pattern1 pattern2'</code>	检索同时匹配指定模式的软件包
<code>~Aarchive</code>	检索指定软件包类型（如 <code>stable</code> 、 <code>unstable</code> 或 <code>testing</code> 等）中的所有软件包
<code>~b</code>	检索依赖关系不满足的软件包
<code>~BdepType</code>	检索不满足指定依赖类型的软件包，依赖类型可以是 <code>depends</code> 、 <code>predepends</code> 、 <code>recommends</code> 、 <code>suggests</code> 、 <code>breaks</code> 、 <code>conflicts</code> 或 <code>replaces</code> 等关键字之一
<code>~Cpattern</code>	检索与指定软件包冲突的软件包
<code>~c</code>	检索已经被删除但其配置文件等尚未完全消除的软件包
<code>~ddescription</code>	检索其描述信息匹配指定描述信息的软件包
<code>~E</code>	检索基本系统软件包（其控制文件中含有“ <code>Essential: yes</code> ”控制语句）
<code>~i</code>	检索系统中已经安装的软件包
<code>~mmaintainer</code>	检索指定作者维护的软件包
<code>~nname</code>	检索匹配指定名字模式的软件包
<code>~N</code>	检索新的软件包
<code>'pattern1   pattern2'</code>	检索匹配指定的表达式 <code>pattern1</code> 、 <code>pattern2</code> 或同时匹配两个表达式的软件包
<code>~T</code>	显示所有的软件包
<code>~U</code>	检索系统中已经安装且需要升级的所有软件包
<code>~v</code>	检索虚拟软件包

### 1. 查询所有的软件包

利用下列 aptitude 命令，可以列出 Ubuntu 提供的（不管系统中是否已经安装）所有软件包，这将是一个长长的软件包列表。

```
$ aptitude search ~T
```

### 2. 列出软件仓库中可供更新的软件包

利用下列命令，可以列出软件仓库中能够用于更新当前系统的所有软件包（如果不经常更新，这将是一个长长的软件包列表）。



```
$ aptitude search ~U
i  linux-generic          - Complete Generic Linux kernel
i  linux-headers-generic   - Generic Linux kernel headers
i  linux-image-generic     - Generic Linux kernel image
i  linux-restricted-modules-generic - Restricted Linux modules for generic kernel
$
```

### 3. 列出系统中已经安装的软件包

利用下列命令，可以列出系统中已安装的所有软件包（这将是一个长长的软件包列表）。

```
$ aptitude search ~i
```

### 4. 组合检索功能

利用逻辑与及逻辑或特殊检索模式，可以组合两个检索模式，检索具有特定意义的软件包。

例如，要检索系统中已经安装的 Apache 服务器软件包，可以使用下列 aptitude 命令。

```
$ aptitude search '~i apache'
i  apache2                  - Next generation, scalable, extendable web
i  apache2-doc               - documentation for apache2
i A apache2-mpm-prefork      - Traditional model for Apache HTTPD
i A apache2-utils             - utility programs for webservers
i A apache2.2-common         - Next generation, scalable, extendable web
i  libapache2-mod-perl2       - Integration of perl with the Apache2 web s
i  libapache2-mod-php5        - server-side, HTML-embedded scripting langu
i  libapache2-mod-python      - Apache 2 module that embeds Python within
$
```

在上述输出信息中，第 3~5 行的第一列中的“A”标志说明相应的软件包是自动安装的，参见表 12-3。

## 12.3.5 删 除 软 件 包

aptitude 命令的 remove 或 purge 功能选项用于删除或彻底清除指定的软件包。当利用 remove 功能选项删除指定的软件包时，如果软件包中含有配置文件，配置文件将会继续被保留在系统中。而 purge 功能选项能够彻底清除指定的软件包。当不再需要某个软件包时，可以根据具体情况，采用 remove 或 purge 功能选项，部分或全部删除指定的软件包。例如，采用下列 aptitude 命令将会删除 quota 软件包，但其配置文件将会遗留在系统中。

```
$ sudo aptitude remove quota
Reading package lists... Done
Building dependency tree
Reading state information... Done
Reading extended state information
Initializing package states... Done
Building tag database... Done
The following packages will be REMOVED:
  quota
0 packages upgraded, 0 newly installed, 1 to remove and 0 not upgraded.
Need to get 0B of archives. After unpacking 1376kB will be freed.
Writing extended state information... Done
(Reading database ... 200852 files and directories currently installed.)
Removing quota ...
  * Turning off quotas...
  * Stopping quota service rpc.rquotad
Reading package lists... Done
Building dependency tree
Reading state information... Done
Reading extended state information
Initializing package states... Done
Writing extended state information... Done
Building tag database... Done
$ aptitude search ~c
```

```
c libgd2-noxpm           - GD Graphics Library version 2 (without XPM)
c quota                  - implementation of the disk quota system
$
```

如同安装软件包一样，删除软件包也存在软件包依赖关系的问题。如果删除的软件包是其他软件包的底层支撑软件包，`aptitude` 将会提请用户决定是否继续删除指定的软件包。

### 12.3.6 图形界面

`aptitude` 既是一个灵活的命令行工具，也提供了一个简单的图形界面。如果在运行 `aptitude` 命令时未指定任何功能选项与参数，将会出现如图 12-1 所示的界面。

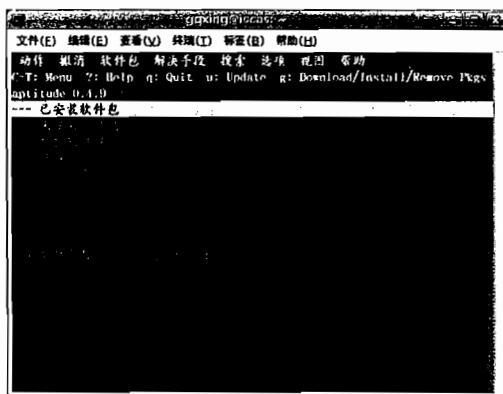


图 12-1 `aptitude` 软件维护界面

尽管这个图形界面的软件管理工具能够替代命令行，但并不容易使用。如果希望使用图形界面维护软件，最好选用 `synaptic`，故这里不再赘述。

## 12.4 synaptic 图形界面软件管理工具

`synaptic` 软件包管理器是一种基于 APT 开发的高级图形界面软件管理工具，其中实现了 `apt-get` 命令行工具的所有功能，启动时能够自动连接与检索预先定义的软件仓库，可用于安装或删除选中的软件包，浏览或检索可用的软件包，升级整个系统，以及查询软件包的说明信息等。

运行 `synaptic` 时，可在 GNOME 桌面中选择“系统→系统管理→新立得软件包管理器”菜单，或者在命令行界面输入 `synaptic` 命令。`synaptic` 软件包管理器的主界面如图 12-2 所示。

在“新立得软件包管理器”主界面中，左下角的“组别”、“状态”、“源自”、“自定义过滤器”以及“搜索结果”5 个按钮分别用于选择软件分组，查询软件包的安装状态，查询与设置软件源，定义过滤器，以及查询新近执行的搜索结果。

当单击不同的按钮时，左上角的窗口将会随之显示相应的内容。初始进入 `synaptic` 主界面时，窗口中将会显示所有的软件包（“全部”）。

单击左上角窗口中的任何一个软件组时，右上角的窗口将会显示相应软件组中包含的软件包，其中第一列的复选框表示软件包的安装状态。第 2~6 列分别是软件源的标志、软件包的名字、当



前已安装的软件包版本、最新版本以及软件包的简单描述。如果第一列的复选框为空，表示相应的软件包尚未安装；如果复选框中涂有颜色，说明相应的软件包已经安装。

单击右上角窗口中的任何一个软件包时，右下角的窗口中将会给出软件包的更多说明信息。此外，选中右上角窗口中的某个软件包，单击“属性”按钮或者右击右上角窗口中的任何软件包时，将会弹出一个菜单，单击“属性”菜单项也会给出相应软件包的各种说明信息，如图 12-3 所示。

在 synaptic 图形界面中，除了使用鼠标选择菜单与按钮之外，还可以使用快捷键，实现软件包的选择、标记以及软件包的安装、删除与升级等（参见表 12-6）。

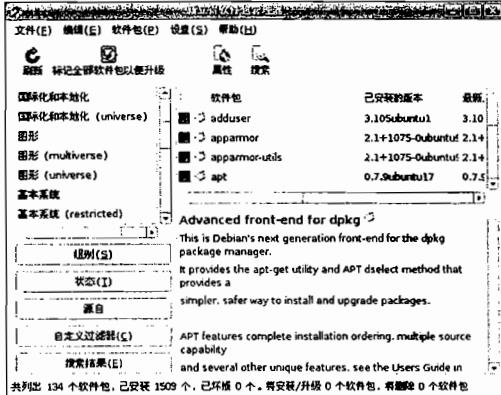


图 12-2 新立得软件包管理器主界面

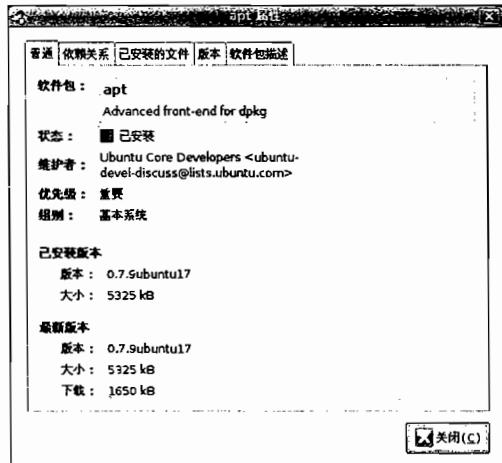


图 12-3 软件包属性界面

表 12-6

synaptic 支持的快捷键

快 捷 键	简 单 说 明
Ctrl+R	更新软件包列表
Ctrl+F	打开软件包搜索对话框
Ctrl+O	显示选定软件包的属性窗口
Ctrl+I	标记选定的软件包以便安装
Ctrl+U	标记选定的软件包以便升级
Delete	标记选定的软件包以便删除
Shift+Delete	标记所选的软件包以便彻底删除
Ctrl+N	取消全部标记
Ctrl+G	标记所有更新
Ctrl+E	强制安装某个版本的软件包
Ctrl+Z	撤销最近一次标记的变动
Ctrl+Shift+Z	重做最近一次变动
Ctrl+P	应用标记的所有变动
Ctrl+Q	退出新立得软件包管理器

### 12.4.1 浏览软件包

在 synaptic 软件包管理器主界面中，单击左下角中的相应按钮，可以按照软件源、软件分组、软件包安装状态、自定义过滤器或最近的检索结果浏览可用的软件包。如果需要，用户也可以创建自己的过滤器。要按照名字或说明信息检索软件包，可在工具栏的“快速搜索”字段中输入检索关键字，或者选择“编辑→搜索”菜单，synaptic 将会弹出一个“查找”对话框，如图 12-4 所示。

此时，可以在“搜索”字段中输入检索关键字，在“搜索位置”的下拉框中选择检索原则，如软件包的名称、软件包描述和名称、维护者、版本、依赖关系以及软件包的提供者，然后执行准确或模糊检索，搜索后的结果如图 12-5 所示。

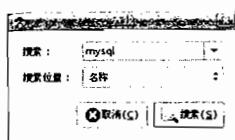


图 12-4 软件包检索界面

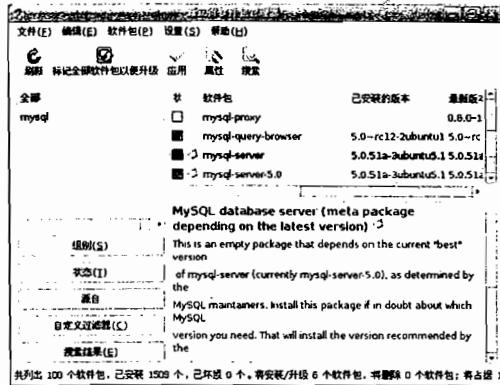


图 12-5 软件包搜索结果界面

在“新立得软件包管理器”右下角的窗口中，可以查看软件包的详细说明信息，如软件包的大小、依赖关系、建议安装的附加软件包及软件包的简单说明等。

### 12.4.2 安装软件包

在安装软件包时，可以首先单击“刷新”按钮（或按 Ctrl-R 键），以便 synaptic 再次联系软件源，获取最新的软件包信息。然后在左上角的窗口中选择软件分组，在右上角的窗口中选择软件包，双击软件包左边的复选框（或按 Ctrl-I 键）；也可以右击选定的软件包，从弹出的上下文菜单中选择“标记以便安装”菜单项，如图 12-6 所示。

选中后，复选框中将会出现一个弧形箭头键，如图 12-7 所示。

如果选定的软件包要求安装其他相关的软件包，synaptic 将会弹出一个对话框，列出尚需安装的底层支撑软件包，如图 12-8 所示。

单击“标记”按钮，表示同意安装附加的软件包。一旦选定了所有的软件包，单击工具栏中的“应用”按钮（或按 Ctrl-P 键），弹出一个对话框，其中给出了即将执行的软件变动的概要说明，如图 12-9 所示。

单击“应用”按钮，表示同意安装本次选定的软件包，即可开始安装选定的所有软件包。

如果选定的软件包与系统中已安装的软件包有冲突，synaptic 将会给出一个警告信息。在此情况下，对话框中将会给出需要删除的软件包。如果无法肯定是否不再需要将要删除的软件包，在单击“应用”按钮之前，应仔细检查其功能和用途。



## Ubuntu 权威指南

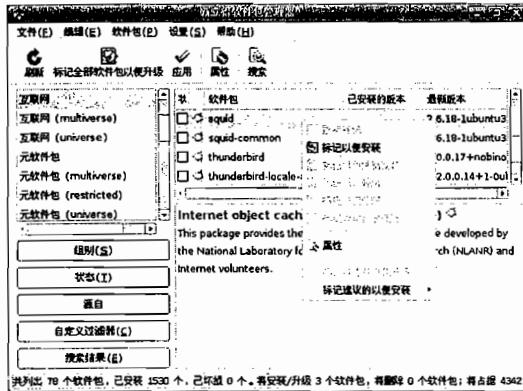


图 12-6 选择要安装的软件包



图 12-7 选定待安装的软件包

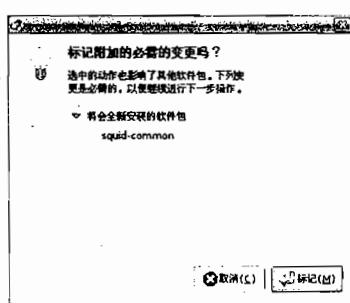


图 12-8 尚需安装的底层支撑软件包

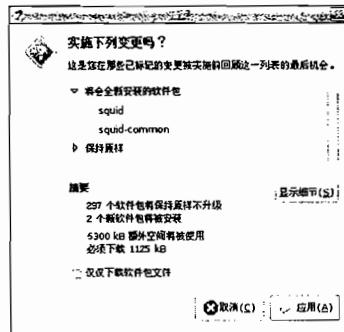


图 12-9 软件包安装汇总信息

### 12.4.3 删 除 软 件 包

要删除或完全清除一个软件包，可在主界面右上角的窗口中右击选定的软件包，从弹出的下文菜单中选择“标记以便删除”菜单项。如果选择“标记以便彻底删除”选项，则意味着让 synaptic 删除选定的软件包及其任何有关的配置文件，如图 12-10 所示。

选中后，复选框中将会出现一个红叉“×”，表示准备删除相应的软件包，如图 12-11 所示。

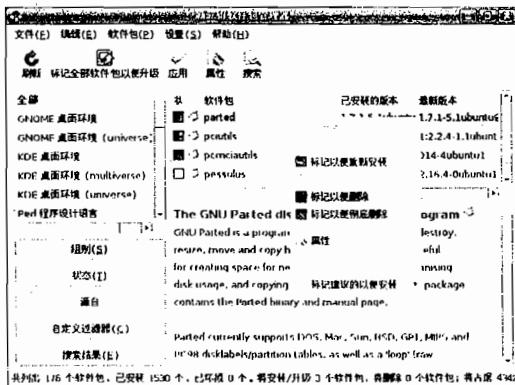


图 12-10 选择要删除的软件包

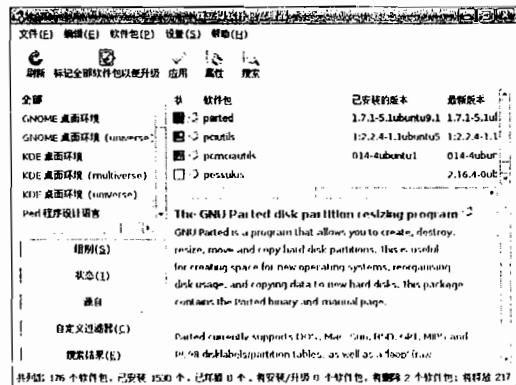


图 12-11 选定待删除的软件包

如果选定的软件包与系统中已安装的软件包有冲突, synaptic 将会给出一个警告信息, 说明需要连带删除的其他软件包, 如图 12-12 所示。

在标记了选定的软件包之后, 单击工具栏中的“应用”按钮(或按 Ctrl-P 键), 系统将会弹出一个对话框, 其中包含选择结果的汇总信息, 如图 12-13 所示。如果没有疑义, 单击状态栏中的“应用”按钮予以确认。

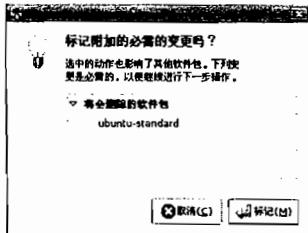


图 12-12 删除软件包影响的其他软件包



图 12-13 软件包删除汇总信息

#### 12.4.4 软件升级

在开始升级之前, 可以单击主界面工具栏中的“刷新”按钮(或按 Ctrl-R 键), 以便 synaptic 连接软件源, 获取最新的软件包信息。如果安装的软件包需要更新, 其左边涂有颜色的复选框右上角将会存在一个星形标记。之后, 可以按照软件包的分组, 双击选定的软件包, 或者右击选定的软件包, 从弹出的上下文菜单中选择“标记以便升级”菜单项(或按 Ctrl-U 键), 如图 12-14 所示。

如果选定的软件包涉及的其他底层支持软件包需要一同更新, 系统将会弹出一个确认对话框, 如图 12-15 所示。

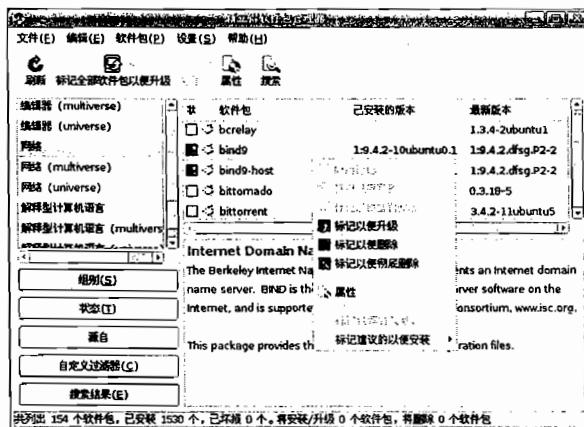


图 12-14 软件升级选择界面

如果没有疑义, 单击主界面状态栏中的“应用”按钮(或按 Ctrl-P 键)予以确认。synaptic 将会弹出一个对话框, 其中包含软件包选择结果的汇总信息, 如图 12-16 所示。



确认之后，synaptic 将会按照用户的选择，更新系统中已经安装的软件包。

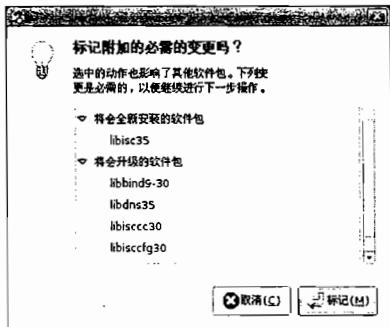


图 12-15 确认对话框

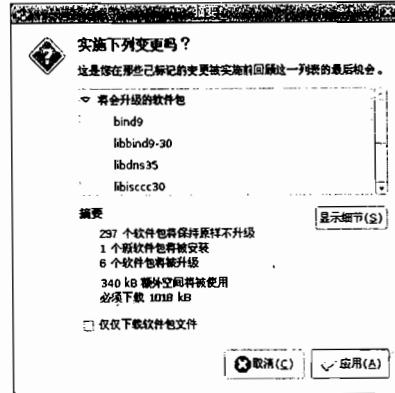


图 12-16 软件升级汇总信息

## 12.5 GNOME 软件增删工具

GNOME 软件包增删工具是一种针对于 GNOME 桌面菜单的软件管理工具，可以根据软件的功能或菜单分类，补充安装或删除选定的软件包，也可用于浏览、检索及查询软件包的说明信息。

要运行 GNOME 软件包增删工具，可在 GNOME 桌面中选择“应用程序→添加/删除”菜单，或者在命令行界面中输入 `gnome-app-install` 命令，启动后自动连接到预定义的软件仓库，检索可用的软件包，最终出现如图 12-17 所示的“添加/删除程序”主界面。

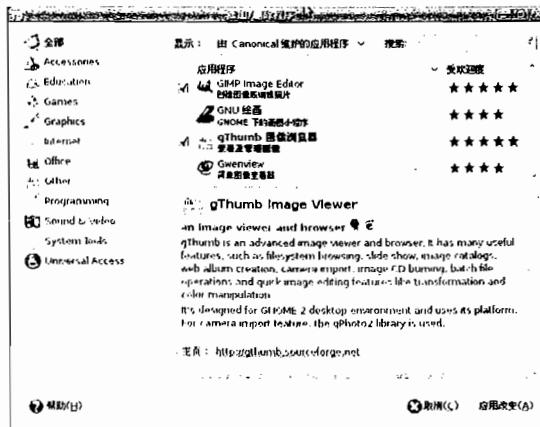


图 12-17 软件增删对话框

在 GNOME 软件增删工具的主界面中，左边的窗口按照 GNOME 菜单的组织结构分类列出了每一个菜单组项；当选择不同的菜单组项时，右上角的窗口将会给出其中包含的软件包列表。在软件包列表中，第一列的复选框用于表示软件包的安装状态，如果复选框含有一个对号“√”，表示相

应的软件包已经安装；第二列是应用程序，给出软件包的名字；第3列以五角星的数量表示软件包受欢迎的程度。当单击某个软件包时，右下角的窗口将会给出相应软件包的简要说明信息。

安装或删除软件包时，可先从左边窗口中选择软件的类型或菜单项，然后从右上角的窗口中选择准备增加或删除的软件包。如果发现了希望安装或删除的软件包，单击软件包右边的复选框，表示选中了相应的软件包，即选中了准备安装或删除的软件包。

在主界面上边的“显示”栏目中，根据Ubuntu软件仓库的组织方式，可以从下拉菜单中选择显示来自不同软件源的软件包。例如，选择显示所有可安装的应用程序、所有开源应用程序、Ubuntu支持的应用程序或第三方提供的应用程序，也可以选择显示已安装的软件包。通常，GNOME软件增删工具主界面是按照所有可安装的应用程序分类显示的。

主界面右上角的“搜索”栏目用于检索或过滤软件包。如果知道软件包的名字或部分名字，可以采用其中的关键字检索软件包，在窗口左上部的“检索”框中，输入表示软件包的关键字，如“mysql”。通常，系统仅从Ubuntu支持的软件仓库中检索软件包，如果检索不到自己需要的软件包，或者需要安装其他软件仓库中的软件包，可从窗口中上部的“显示”下拉框中选择不同类型的软件仓库，以便能够从更多的软件源中选择期望的软件包。一旦发现期望的软件包，可以单击软件包左边的复选框。

在选定了所有的软件包之后，单击窗口右下部的“应用改变”按钮，将会出现一个新的窗口，其中列出了用户选中的所有软件包，同时提请用户予以确认，如图12-18所示。

如果列出的软件包是正确的，可单击“应用”按钮。取决于选定的软件包，系统可能会提示用户输入sudo密码。之后出现一个新的窗口，显示软件包的下载与安装进度，开始安装或删除选定的软件包及其相关软件包。一旦完成安装，单击“关闭”按钮，即可立即使用新装的软件包，如图12-19所示。

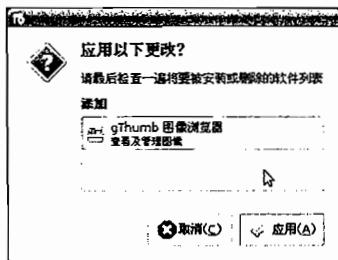


图 12-18 软件增删确认对话框

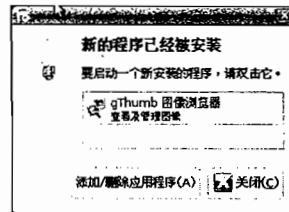


图 12-19 软件增删完成对话框

注意，GNOME软件增删工具能够安装的软件包仅限于GNOME桌面菜单的软件组成部分，并非Ubuntu提供的全部软件包。对于其他软件而言，即使存在尚未安装的软件包，也无法利用GNOME软件增删工具安装软件。例如，如果想要安装MySQL、Apache或DNS服务器等软件包，只能采用其他软件维护工具，如apt-get、aptitude或synaptic等。

## 12.6 软件包的自动更新

Ubuntu Linux系统提供的软件更新工具update-notifier程序与update-manager脚本主要用于检测系统配置的软件仓库中是否存在可用的软件包。如果存在更新的软件包，GNOME菜单面板的



信息公告区中将会出现一个软件更新图标（红色下箭头“↓”），同时在桌面上弹出一个“可用的软件更新”消息框，提醒用户更新自己的系统，如图 12-20 所示。

单击消息公告区中的软件更新图标，即可调用 update-manager。输入 sudo 密码，GNOME 桌面中将会出现一个“更新管理器”窗口，如图 12-21 所示。

“更新管理器”窗口中列出了所有可用的更新软件包。第一列是一个复选框，其中的对号“√”表示选定了相应的软件包；第二列是软件包的名字和简单说明。通常，直接单击窗口左下角的“安装更新”按钮，在随后弹出的界面中输入 sudo 密码，即可开始下载和安装软件包。之后就是观察软件包的下载和安装进度，等待软件包安装结束。

如果长时间没有更新系统，也可以先单击“检查”按钮，再次检索是否还存在其他更新软件包。然后再单击“安装更新”按钮。

当需要更新的软件包较多，网速又比较慢时，最好按照软件包的重要程度，分批更新软件包。不然的话，一旦网络中断或因无法等待更新的完成而退出“更新管理器”窗口，将会前功尽弃，之前下载的软件包也无法再用。为此，可以单击部分软件包左侧的复选框，清除其中的对号“√”，暂不安装相应的软件包。

update-manager 是一个简单易用的软件维护工具，也是 Ubuntu Linux 基本系统的一部分，每天都以后台进程的形式自动运行，执行系统的全面更新检查，能够帮助用户实现系统软件的及时更新。

在软件包更新结束之后，如果需要重新启动系统，消息公告区中的将会出现一个圆形的双箭头图标，如图 12-22 所示。



图 12-21 更新管理器窗口

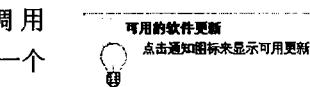


图 12-20 软件更新消息框

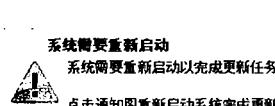


图 12-22 系统重启图标

根据具体的情况，此时可以选择立即重启系统，也可选择在之后方便的时间再重新启动系统。

# 第13章

## 用户管理

本章主要介绍 Linux 系统的用户管理，讨论怎样增加、修改或删除用户与用户组，介绍与用户管理有关的系统文件，说明如何设置用户的运行环境，详细讨论插件式认证模块，以及超级用户与 sudo 命令。其内容主要包括：

- 增加与删除用户；
- 定制用户的工作环境；
- 增加与删除用户组；
- 监控用户；
- 插件式认证模块；
- 超级用户与 sudo 命令。





用户与用户组是 Linux 系统管理的一个重要部分，也是系统安全的基础。在 Linux 系统中，所有的文件、程序或正在运行的进程都从属于一个特定的用户。每个文件和程序都具有一定的访问权限，用于限制不同用户的访问行为。作为系统管理员，管理用户和用户组是一项重要的任务，有助于防止用户越权访问与其身份不相符的文件，执行系统任务，对系统造成破坏。总之，如果没有有效的用户管理，就无法保证系统的安全。

## 13.1 增加与删除用户

在 Linux 系统中，每个用户都具有一个唯一的身份标识，称做用户 ID（简称 UID），以区别于其他用户。Linux 系统按一定的原则把用户划分为用户组，以便相关的同组用户之间能够共享文件。除了系统用户之外，在使用 Linux 系统之前，每个人都应分配一个用户名。在利用用户名和密码注册成功之后，才能访问 Linux 系统，执行系统命令，开发和运行应用程序，访问数据库，提交自动运行的后台任务等。

一般地讲，Linux 系统中的用户可以分为 3 类：超级用户（root）、管理用户和普通用户。但也可以把超级用户和管理用户通称为系统用户。

超级用户是一个特殊的用户（其用户标识号为 0），它拥有至高无上的访问权限，可以访问任何程序和文件。任何系统都会自动提供一个超级用户账号。在 Ubuntu Linux 系统中，为了确保系统的安全，超级用户账号通常是被锁住的。Ubuntu Linux 系统强烈建议，应尽量避免使用超级用户注册到系统中，如果确实需要执行系统管理与维护任务，可以在具体的命令前冠以 sudo 命令。

管理用户用于运行一定的系统服务程序，支持和维护相应的系统功能。这些用户的 ID 号位于 1~999 的范围之内。例如，ftp 就是一个管理用户，用做 FTP 匿名用户，维护匿名用户的文件传输，同时提供匿名用户的默认主目录。

除了超级用户与管理用户之外，其他均为普通用户。访问 Linux 系统的每个用户，都需要有一个用户账号。只有利用用户名和密码注册到系统之后，才能够访问系统提供的资源和服务。因此，在使用系统之前，必须由系统管理员为用户分配一个注册账号，把用户名及其他有关信息加到系统中。

对于自己的文件，用户均拥有绝对的权力，可以赋予自己、同组用户或其他用户访问文件的权限。例如，允许同组用户共享自己的文件，其他用户只能阅读或执行，但不能写文件等。

### 13.1.1 /etc/passwd 文件

在安装 Linux 系统之后，系统已经事先创建了若干系统用户账号，其中包括超级用户 root 和管理用户 daemon、bin 和 sys 等，用于执行不同类型的系统管理和日常维护任务。

用户的账号信息是由/etc/passwd 和/etc/shadow 文件共同维护的。在这两个文件中，每个用户都有一个相应的记录。当利用控制台等终端注册，按照系统的提示输入用户名和密码之后，系统将会根据用户提供的用户名检查/etc/passwd 文件是否存在，然后根据用户提供密码，利用同一加密算法加密后再与/etc/shadow 文件中的密码字段进行比较，同时检查其他诸如密码有效期等字段。如果通过了验证，按照 passwd 文件指定的主目录和命令解释程序，用户即可进入自己的主目录，通过命令解释程序访问 Linux 系统。

除了用户名和密码之外，每个用户还具有用户 ID、用户组 ID、主目录和命令解释程序等其

他相关信息。这些信息分别存放在 passwd 和 shadow 文件中。

passwd 文件中包含 Linux 系统中每个用户除密码之外的重要信息，每个用户信息占用一行，每一行由 7 个字段组成，中间以冒号分开。passwd 文件的格式定义如下。

```
username:password:uid:gid:comment:home_dir:login_shell
```

表 13-1 给出了 passwd 文件中的每个字段及其简单说明。

表 13-1 /etc/passwd 文件中的字段及其说明

字段	简单说明
username	注册用户名。由 2~8 个字母 (A~Z, a~z) 和数字 (0~9) 等字符组成。在 Linux 系统中，用户名必须是唯一的，且至少第一个字符应选用字母
password	这个字段原为用户密码，但实际的密码已移至/etc/shadow 文件。如果用户设有密码，这个字段会包含一个小写字母 “x”，加密后的密码存放在 shadow 文件中。如果这个字段为星号 “*”，表示相应的用户无法正常地注册到系统
uid	用户 ID (或称用户标识)。用户 ID 是系统或系统管理员分配给每个用户的一个唯一的数字标识，是系统识别每个用户的主要手段。当系统需要了解用户信息 (如账号字段的内容) 时，通常均以用户 ID 作为索引，检索 passwd 文件。用户 ID 是一个 32 位的无符号整数，每个 ID 必须位于 0~65 536 的整数范围内，其中 0~999 保留作系统用户使用，自定义的用户 ID 应位于 1 000~65 536 的范围之内。考虑到与其他系统的兼容性，建议使用 16 位无符号整数的最大值 32 767 作为用户 ID 的上限
gid	用户组 ID (或称用户组标识)。Linux 系统中的每个用户均属于某个用户组，每个用户组除有组名之外，也有一个相应的用户组 ID。同样，ID 号 0~999 保留作系统用户组使用
comment	注释信息。通常包含用户全名、电话号码和电子邮件地址等用户信息
home_dir	指定用户主目录 (全路径名)，也是用户注册后的起始工作目录。用户主目录是文件系统中的一个子目录，分配给用户，供存储个人文件用。按照惯例，用户主目录通常位于/home 目录下面，形如/home/username。通常，HOME 等环境变量就是按这个字段设置的
login_shell	指定用户注册后调用的 Shell，即命令解释程序。如果这个字段为空，则默认的命令解释程序为/bin/bash。用户也可以根据自己的爱好，选用其他命令解释程序，如 Korn Shell (/bin/ksh)、zsh (/bin/zsh) 或 tcsh (/bin/tcsh) 等

下面是 Linux 系统安装后自动提供的默认用户信息。

```
$ cat /etc/passwd
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/bin/sh
bin:x:2:2:bin:/bin/sh
sys:x:3:3:sys:/dev:/bin/sh
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/bin/sh
man:x:6:12:man:/var/cache/man:/bin/sh
lp:x:7:7:lp:/var/spool/lpd:/bin/sh
mail:x:8:8:mail:/var/mail:/bin/sh

...
syslog:x:102:103::/home/syslog:/bin/false
klog:x:103:104::/home/klog:/bin/false

...
gqxing:x:1000:1000:gqxing,,,,:/home/gqxing:/bin/bash
$
```

### 13.1.2 /etc/shadow 文件

/etc/shadow 是一个限制访问的系统文件，其中存有加密形式的密码和其他相关信息。与 passwd 文件相对应，shadow 文件的每个用户密码信息占用一行，每一行由 9 个字段组成，中间以冒号分开。其文件格式定义如下。

```
username:password:lastchanged:mindays:maxdays:warn:inactive:expire:reserve
```



表 13-2 给出了 shadow 文件中的每个字段及其简单说明。

表 13-2 /etc/shadow 文件中的字段及其说明

字段	简单说明
username	用户注册名（参见/etc/passwd 文件）
password	加密形式的密码（通常是由 crypt(3) 函数生成的）。通常，用户输入的密码至少应包含 6~8 个字符，但生成的密码较长，由 13~15 个字符组成。如果加密后的密码以美元符号 “\$” 为起始字符，意味着加密的密码是由其他算法（非 DES 算法）生成的。例如，如果加密后的密码以 “\$1\$” 为起始字符串，意味着加密的密码是采用 MD5 之类的加密算法生成的。如果这个字段包含感叹号 “!” 或星号 “*” 字符，则表示用户无法正常地注册
lastchanged	从 1970 年 1 月 1 日开始算起，直至最后一次修改密码之日的天数
mindays	保持密码稳定不变的最小天数，仅当超过此限，才能修改密码。这个字段必须大于等于 0，才能启用密码有效期检查
maxdays	保持密码有效的最大天数。超过此限，系统将会强制提请用户更换新的密码
warn	指定在密码有效期到期之前需提前多少天向用户发出警告信息
inactive	指定在密码有效期到期之后一直不访问系统，但仍保证其账号信息有效的最多天数，超过此限将封锁用户账号。用户最后一次访问系统的信息记录于/var/log/lastlog 文件中
expire	指定用户账号有效期的截止日期。到期后，账号将会自动失效，用户无法再注册到系统
reserve	保留字段

下面是 Linux 系统安装后提供的初始 shadow 文件。

```
$ sudo cat /etc/shadow
root:$1$jp2QHF7W$c71u36dnLJP5vgNOnTBhz.:14069:0:99999:7:::
daemon:*:13991:0:99999:7:::
bin:*:13991:0:99999:7:::
sys:*:13991:0:99999:7:::
sync:*:13991:0:99999:7:::
games:*:13991:0:99999:7:::
man:*:13991:0:99999:7:::
lp:*:13991:0:99999:7:::
mail:*:13991:0:99999:7:::
....
syslog:*:13991:0:99999:7:::
klog:*:13991:0:99999:7:::
....
gqxing:$1$IoW133.R$VM8i2cABtDJ6i186wraRM/:14068:0:99999:7:::
$
```

### 13.1.3 用户管理实例

在 Ubuntu Linux 系统中，为了管理用户和用户组，可在 GNOME 桌面中选择“系统→系统管理→用户和组”菜单，进入图 13-1 所示的对话框。单击“解锁”按钮，在新弹出的“认证”界面中输入 sudo 密码，即可开始增加新的用户或用户组，修改以及删除用户或用户组。注意，在使用 GNOME 桌面系统管理用户和用户组时，用户 ID 和用户组 ID 的默认起始编号为 1001（安装 Ubuntu Linux 系统时创建的第一个用户占用了编号为 1000 的用户 ID 和用户组 ID）。

在“用户设置”界面中单击“添加用户”按钮，即可进入图 13-2 所示的“新建用户账户”界面。此时，可在“基本设置”一栏的“用户名”和“真实姓名”字段中分别输入用于系统注册的用户名和实际的用户名，对于“配置文件”字段，可选择默认的“Desktop user”，或者从下拉菜单中选择“Administrator（赋予用户运行 sudo 命令的权限，但并非超级用户）”或“Unprivileged

(表示普通用户)”, 最后单击“用户权限”标签, 进入图 13-3 所示的界面。

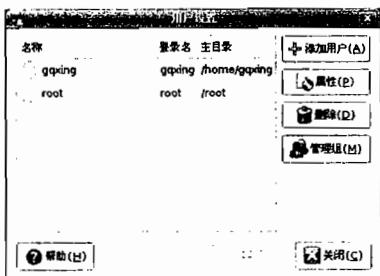


图 13-1 用户设置界面

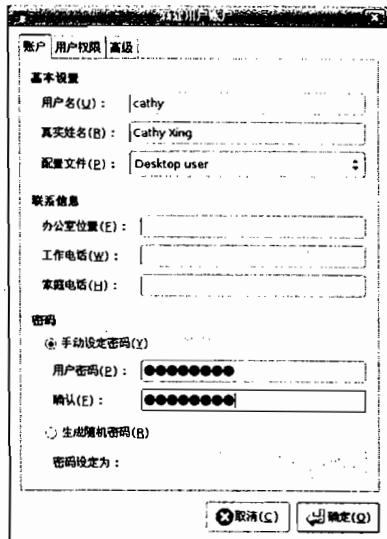


图 13-2 新建用户账户界面 (1)

“新建用户账户(用户权限)”界面列出了可赋予用户的各种访问权限。选中左边的复选框, 即可赋予相应权限(实际上是把用户加到各种用户组中, 使用户拥有多个次用户组, 从而拥有相应的访问权限)。例如, 如果赋予用户“管理用户”的权限, 相当于把用户加到用户组 admin, 因而拥有运行 sudo 命令, 执行特权命令的权限。

单击“高级”标签, 可进入图 13-4 所示的“新建用户账户(高级)”界面。在“高级设置”一栏中, 可以设置用户的主目录、注册 Shell、用户组以及用户 ID。

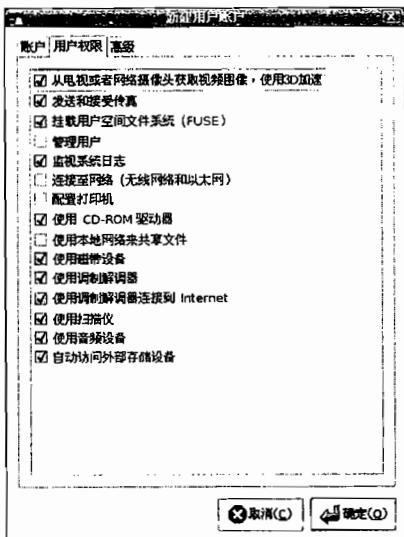


图 13-3 新建用户账户界面 (2)

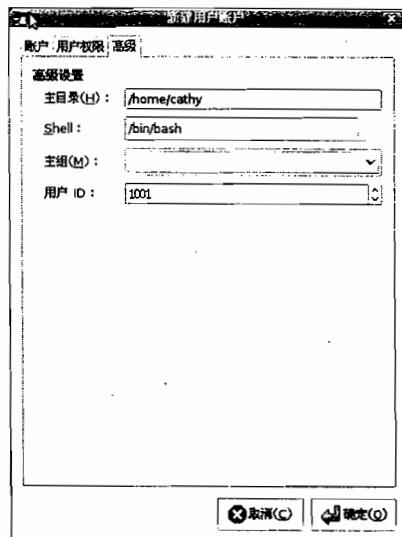


图 13-4 新建用户账户界面 (3)



在完成上述设置之后，单击“确定”按钮，即可返回图 13-5 所示的“用户设置”界面，表示已经把用户加到了系统中。

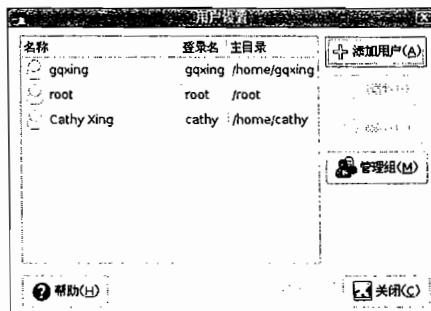


图 13-5 完成设置后的用户设置界面

除了 GNOME 桌面桌面环境，还可以使用 Linux 系统的基本命令，实现用户和用户组的管理与维护。下面以命令行方式为例，说明怎样增加、修改和删除用户。

### 1. 增加用户

在命令行方式中，要增加用户，可以使用 useradd 命令，其语法格式如下。

```
useradd [-u uid] [-g group] [-d home_dir] [-s shell] [-c comment]
        [-m [-k skel_dir]] [-N] [-f inactive] [-e expire] login
```

其中，login 表示新增用户的注册用户名，其他选项的说明如表 13-3 所示。

表 13-3

useradd 命令的常用选项

选 项	GNU 选 项	简 单 说 明
-u uid	--uid uid	用于指定新增用户的用户 ID。通常，用户 ID 是当前已经分配的最大 ID 号加 1。例如，如果当前已分配的用户 ID 为 1000、1005 和 2000，则默认的下一个可用用户 ID 将是 2001
-g group	--gid group	用于指定一个现有用户组的用户组 ID 或用户组名
-N	--no-user-group	通常，增加新用户时也能够同时增加一个与用户名同名的用户组。利用这个选项可以禁止创建同名的用户组，把用户加到/etc/default/useradd 文件指定的默认用户组（users）中
-d home_dir	--home home_dir	用于指定新增用户的主目录。通常，用户主目录为/home/username。其中，username 为新增用户的注册用户名，即命令语法格式中的“login”参数
-s shell	--shell shell	用于指定命令解释程序 Shell 的完整路径名。默认的命令解释程序为/bin/bash
-c comment	--comment comment	用于指定用户全名、电话号码以及电子邮件地址等注释信息
-m	--create-home	在增加新用户时，如果用户的主目录不存在，则创建用户主目录。同时，把/etc/skel 目录或“-k”选项指定目录中的初始化文件复制到用户主目录中
-k skel_dir	--skel skel_dir	用于指定存储用户初始化文件（如.profile）的目录，以便 useradd 命令能够把其中的文件复制到用户主目录。通常，系统将会提供一组标准的初始化文件，为用户设置最基本的运行环境。用户也可以此为起点，增加自己的设置。系统默认的目录位置为/etc/skel
-f inactive	--inactive inactive	用于指定相应用户一直未访问系统，但仍保证其注册账号信息有效的最多天数。超过此限将锁住用户账号
-e expire	--expiredate expire	指定注册用户的有效期，即截止日期。如果超过指定的日期，则不允许使用相应的用户名访问系统。有效的日期格式为“YYYY-MM-DD”。这个选项主要用于增加一个临时用户

例如，要增加一个用户 cathy，同时增加一个用户组 cathy，可以使用下列命令。

```
$ sudo useradd -u 1001 -d /home/cathy -m -s /bin/bash cathy
[sudo] password for gqxing:
$
```

执行上述命令之后，系统将会在/etc/passwd、/etc/shadow 和/etc/group 文件中各增加一行与用户 cathy 有关的信息，如下所示。

```
$ grep cathy /etc/passwd
cathy:x:1001:1001::/home/cathy:/bin/bash
$ sudo grep cathy /etc/shadow
cathy:!:14251:0:99999:7:::
$ grep cathy /etc/group
cathy:x:1001:
$
```

在选用用户名、用户 ID 和用户组 ID 时通常应遵循下列原则。

- 用户名应包含 2~8 个字母和数字，第一个字符必须是字母，而且至少有一个字符是小写字母。注意，即使用户名中允许包含句点 “.”、下划线 “\_” 或减号 “-”，也不建议使用。
- 避免使用/etc/passwd 和/etc/group 文件中已有的用户名、用户 ID、用户组名及用户组 ID。此外，用户 ID 和用户组 ID 号 0~999 是系统保留的，任何普通用户均不应使用。

一旦创建了用户账号，就可以使用 passwd 命令设置密码，使用 usermod 等命令修改 passwd 和 shadow 文件，更改用户的其他相关属性。

注意，在增加、修改和删除用户时，如果采用编辑器，手工修改 passwd 和 shadow 文件，必须考虑以下因素：

- 用户属于哪个用户组，相应的用户组是否存在；
- 除了修改 passwd 文件之外，还要在/etc/shadow 文件中增加、修改和删除相应的用户项；
- 创建和删除用户主目录；
- 另外，还要创建或复制用户初始化文件，如.profile 和.bashrc 等文件（取决于选定的命令解释程序）。

## 2. 修改用户

除非用户名或用户 ID 与其他用户冲突，一般情况下不要轻易修改用户账号中的用户名和用户 ID，因为这将涉及用户已经创建的所有文件和目录。否则，需要同时修改用户所有文件和目录的文件属主等属性。但可以放心地修改用户账号的其他字段：

- 注释字段；
- 命令解释程序；
- 密码及密码的其他属性（shadow 文件）；
- 主目录及主目录的访问权限。

修改用户信息时，可以利用编辑器，以手工方式直接编辑 passwd 和 shadow 文件，也可以使用 usermod 命令。usermod 命令的语法格式如下。

```
usermod [-u uid] [-g group] [-d home_dir] [-s shell] [-c comment] [-f inactive]
        [-e expire] [-l new_logname] login
```

其中，“-l”选项用于指定一个新的注册用户名，其他选项的说明参见表 13-3。

例如，要把用户 cathy 的命令解释程序 bash 更换为 ksh，可以使用下列命令。

```
$ sudo usermod -s /bin/ksh cathy
$
```

注意，除了超级用户 root 的密码之外，不要试图修改和删除系统用户账号的任何信息。



### 3. 删除用户

一旦用户不再需要访问 Linux 系统，理应从系统中删除其账号信息。删除用户账号时可以使用 userdel 命令，也可以采用手工编辑的方式。使用 userdel 命令的好处是，只需一个命令即可删除 passwd、shadow 和 group 文件中的相应用户和用户组信息，同时还会把用户主目录中的所有文件和目录一同删除。而采用手工方式，既需编辑 passwd、shadow 或 group 文件，还需要删除主目录及其中的文件。userdel 命令的语法格式如下。

```
userdel [-r] login
```

其中，“-r”选项意味着同时从系统中删除用户的主目录，包括其中的文件和子目录。

### 4. 封锁用户账号

有时，也许需要临时或永久地禁止或者封锁某个用户账号。为此，可以使用 passwd 命令，也可以在/etc/shadow 文件的密码字段增加一个感叹号“!”前缀，或者在/etc/passwd 文件的密码字段增加一个星号“\*”字符，以封锁相应的用户账号，从而防止用户使用此账号再注册到系统中。另外一种方法是修改 shadow 文件中的 expire 字段，把其中的日期改成一个过时的日期。

### 5. 定期更改密码

普通用户可以使用不带任何选项和参数的 passwd 命令修改自己的密码，而超级用户则可以利用此命令的大量选项及参数维护系统中的用户账号信息。例如，利用密码的有效期等设置，可以强制用户定期修改自己的密码。为此，可以使用 passwd 命令。注意，passwd 通常不允许创建无密码的用户账号。passwd 命令的语法格式如下。

```
passwd [-adehlsu] [-i inactive] [-m min] [-w warn] [-x max] [login]
```

表 13-4 给出了 passwd 命令的部分选项及其说明（除非特别说明，相应的选项仅限于超级用户使用）。

表 13-4

passwd 命令的部分选项

选 项	GNU 选项	简 单 说 明
-a	--all	这个选项只能与“-S”选项一起使用，以查询所有用户账号的状态信息
-d	--delete	删除指定用户的密码，同时在 passwd 文件中的 password 字段中增加一个星号“*”字符，以封锁指定用户的注册账号，使用户无法再注册到系统
-e	--expire	令指定用户的密码立即失效，强制用户在下一次注册时立即更换密码
-i inactive	--inactive inactive	指定在密码有效期到期之后一直不访问系统，但仍保证其账号信息有效的最多天数，超过此限将封锁用户账号（实际上相当于设置 shadow 文件中的 inactive 字段）
-l	--lock	封锁指定用户的密码（在加密形式的密码字段中增加一个感叹号“!”前缀），即封锁指定用户的注册账号，拒绝相应用户再次注册。当某用户离开公司，有关文件尚未完全交接时，可以使用此选项先封锁用户账号，待清理完毕之后再删除用户账号信息
-m min	--mindays min	指定密码保持不变的期限——即在更换密码之前必须保持密码不变的最少天数。换言之，即指定多少天之内不能更改密码（实际上相当于设置 shadow 文件中的 mindays 字段）
-S	--status	显示指定用户的密码状态属性
-u	--unlock	解除被封锁的用户注册账号（删除密码字段前面的感叹号“!”前缀）
-w warn	--warndays warn	指定在密码有效期到期之前的多少天向用户发出警告信息（实际上相当于设置 shadow 文件中的 warn 字段）。如果禁止检查密码有效期（shadow 文件中的 mindays 字段为 0），这个选项没有意义。默认情况下是有效期截止日的前一天
-x max	--maxdays max	指定密码的最大有效期限，以天数计算（相当于设置 shadow 文件中相应用户的 maxdays 字段）。如果禁止检查密码有效期（shadow 文件中的 mindays 字段为 0），这个选项（或字段）没有意义

例如，要强制用户 hwang 下一次注册时立即更换密码，可以使用下列命令。

```
$ sudo passwd -e hwang
Password set to expire.
```

要查询用户 cathy 的密码属性，可以使用下列命令。

```
$ sudo passwd -S cathy
cathy P 01/06/2009 0 99999 7 -1
$
```

在上述输出信息中，P 表示用户 cathy 已经设置了密码。其他密码属性包括 NP，表示未设置密码，L 表示用户账号已经锁住。

## 6. 检验用户的有效用户 ID

要检查当前用户的有效用户 ID，可以使用 id 命令。例如，当 cathy 注册到系统之后，id 命令的输出结果如下。

```
$ id
uid=1001(cathy) gid=1001(cathy) groups=1001(cathy)
```

使用 su 命令能够改变用户的有效用户 ID，而不必另行注册。在输入其他用户名和密码之后，一个用户可以“合法地变成”另一个用户身份。除此之外，如果在 su 命令之后加上“-”、“-l”或“--login”选项，还可以直接进入其他用户的主目录，如同使用其他用户名和密码直接注册一样。否则，即使改变用户身份，仍会保持之前的工作目录不变，示例如下。

```
$ su cathy
Password:
$ pwd
/home/gqxing
$ exit
exit
$ su - cathy
Password:
$ pwd
/home/cathy
$
```

## 13.2 定制用户的工作环境

除了分配用户名和用户 ID，设定一个主目录，供用户创建和存储文件之外，还需要有一个工作环境，使用户能够借助系统提供的各种工具和资源，访问系统，开发和运行自己的应用。当用户注册到系统之后，用户的工作环境是由选定的命令解释程序和相应的用户初始化文件确定的。因此，用户管理的另一个任务是选择作为用户界面的命令解释程序和用户初始化文件。

### 13.2.1 选择命令解释程序

Linux 系统配备的标准命令解释程序是 Bash (bash)，但同时也支持 Korn Shell (ksh)、基于 C Shell 的 TC Shell (tcsh) 以及 Z Shell (zsh) 等，可以自由选用。每个 Shell 各具特色，根据自己的需要，用户可以选用不同的命令解释程序。/etc/shells 文件中列出了 Ubuntu Linux 系统支持的所有命令解释程序（注意，ksh、tcsh 及 zsh 等需要单独下载），如下所示。

```
$ cat /etc/shells
# /etc/shells: valid login shells
/bin/csh
/usr/bin/zsh
```



```
/usr/bin/ksh  
/bin/ksh  
/usr/bin/rc  
/usr/bin/tcsh  
/bin/tcsh  
....  
/bin/sh (从 6.10 版开始, /bin/sh 已由 bash 改为 dash 的符号链接文件。注意, 这两者是不兼容的)  
/bin/dash  
/bin/bash  
....  
/bin/zsh  
/usr/bin/zsh  
$
```

下面简单介绍 4 个应用比较广泛且最具代表性的 Shell, 供用户在选用命令解释程序时参考。

### 1. Bourne Again Shell

Bash(Bourne Again Shell)是基于 POSIX 1003.2 标准开发的一个免费版的 Shell, 与 Borne Shell 和 Korn Shell 一脉相承, 在功能上又有较大的提高。Bash 兼收并蓄, 广泛采纳了其他 Shell 的优点, 支持 emacs 与 vi 两种命令行编辑功能, 支持命令历史、命令别名和作业控制等机制, 改善了人机交互能力, 具有良好的用户界面和工作环境。作为基本配置, Bash 是 Linux 系统首选的 Shell, 其地位就像 Bourne Shell 之于 UNIX。

### 2. Korn Shell

Korn Shell (/bin/ksh) 是 UNIX 系统中继 Borne Shell 与 C Shell 之后推出的第 3 个著名的 Shell。Korn Shell 以 Borne Shell 为基础, 同时充分吸纳了 C Shell 的优点, 兼具 Bourne Shell 的功能与 C Shell 的交互能力, 支持命令行编辑、命令历史、命令别名和作业控制等功能, 极大地增强和丰富了 Bourne Shell 的基本功能, 并与 Borne Shell 完全兼容, 同时也奠定了现代 Shell(如 Bash)的基础。

### 3. TC Shell

C Shell 原为 BSD 版 UNIX 系统的命令解释程序, 由加州大学伯克利分校计算机系的 Bill Joy 开发。TC Shell (/bin/tcsh) 是在 C Shell 的基础上开发的, 继承并发展了 C Shell 的全部功能特性, 支持 emacs 命令行编辑、命令历史、命令别名和作业控制等机制, 改善了人机交互能力, 具有良好的用户界面和工作环境。但需要注意的是, TC Shell 与 Bash 和 Korn Shell 等 Bourne 系列的 Shell 是不兼容的, 两个系列 Shell 环境中开发的脚本不能互换使用。

### 4. Z Shell

同上述其他 Shell 一样, zsh 既是一个交互式的命令解释程序, 也是一个强有力的编程语言。zsh 吸收并集成了 bash、ksh 及 tcsh 等 Shell 的许多功能特性, 能够提高用户与 Linux 系统交互的效率, 非常适合用作交互式 Shell。可以说, zsh 是上述 Shell 的超集。zsh 自己也增加了许多先进的功能, 这些功能位于动态库中, 可根据具体需要加载或卸载, 以节省内存。

任何时刻, 每个用户只能使用一个命令解释程序, 但可以随时从一个 Shell 切换到另外一个 Shell 环境。为了确定当前究竟处于哪一个 Shell 环境, 可以使用“ps -f”命令, 找出 ps 的父进程, 即可确定当前使用的是哪一个 Shell, 示例如下。

```
$ ps -f  
UID      PID  PPID  C STIME TTY      TIME CMD  
gqxing   6909  6906  0 21:47 pts/0    00:00:00 bash  
gqxing   6975  6909  0 21:58 pts/0    00:00:00 ps -f  
$
```

增加用户时, 每个用户都可以一次性地选定一个命令解释程序, 如果未做出选择, 系统默认

的命令解释程序为 Bash。如果需要，也可以通过 usermod 等命令修改用户账号信息，或者直接修改 passwd 文件，随时更换命令解释程序。一经注册，即可调用指定或默认的命令解释程序。

Shell 是大多数用户访问 Linux 系统的主要命令行界面。Shell 功能的任何改进，如命令行编辑、命令或文件名补充以及命令提示符定制等，都有助于提高用户访问 Linux 系统的工作效率。Shell 的功能和特点是用户选择 Shell 的依据。

表 13-5 给出了 Shell 功能特性的简单比较，可供用户选择 Shell 时参考。至于究竟选用哪一个 Shell，还要取决于用户的爱好和习惯。

表 13-5

Shell 功能特性比较

Shell 的主要特性	bash	ksh	tcsh	zsh
是否用作 Linux 系统的标准 Shell	Y	N	N	N
作业控制	Y	Y	Y	Y
命令历史	Y	Y	Y	Y
命令别名	Y	Y	Y	Y
命令行编辑	Y	Y	Y	Y
vi 命令行编辑	Y	Y	Y	Y
emacs 命令行编辑	Y	Y	Y	Y
文件覆盖保护 (noclobber)	Y	Y	Y	Y
Shell 函数	Y	Y	N	Y
命令或文件名补充	Y	Y	Y	Y
用户名补充	Y	Y	Y	Y
主机名补充	Y	Y	Y	Y
命令历史补充	Y	N	Y	Y
拼写校正	N	N	Y	Y
进程替换	Y	N	N	Y
是否易于定制命令提示符	Y	Y	Y	Y
是否具有完整的信号捕捉与处理机制	Y	Y	N	Y
内置的数学运算	Y	Y	Y	Y
是否支持较大的参数列表	Y	Y	Y	Y

### 13.2.2 设置用户初始化文件

初始化文件（也称启动文件）是一种 Shell 脚本，其中包含每次注册后用户希望执行的命令，在注册到系统之后，能够设置用户的工作环境。原则上讲，初始化文件能够与普通 Shell 脚本一样执行任何处理任务。但是，初始化文件的主要任务是设定用户的工作环境，如设置命令检索路径、环境变量和终端特性等。每个 Shell 均有自己的系统及用户初始化文件，参见表 13-6。当用户注册后，Shell 将会自动读取并执行系统及用户主目录中的初始化文件。



表 13-6 Shell 使用的初始化文件

Shell	初始化文件	目的
Bash	/etc/profile	用于设置系统范围的工作环境，如设置 PATH 变量，设置命令提示符等
	\$HOME/.profile	用于设置用户自己的工作环境，如设置 PATH 变量，定义命令别名等
	\$HOME/.bash_profile	作用同上，但 Ubuntu Linux 系统并不使用这个用户初始化文件
	\$HOME/.bash_login	作用同上，但 Ubuntu Linux 系统并不使用这个用户初始化文件
	/etc/bash.bashrc	用于设置交互式非注册 Shell 系统范围的工作环境
	\$HOME/.bashrc	用于设置交互式非注册 Shell 的用户工作环境
	BASH_ENV	如果变量已经设置，使用变量的值作为初始化文件，执行文件中的命令，设置非交互式 Shell 的用户工作环境
Korn Shell	\$HOME/.bash_logout	用于定义退出 Bash 时需要执行的善后处理任务，如清除临时文件等
	/etc/profile	用于设置系统范围的工作环境，如设置 PATH 变量等
	\$HOME/.profile	用于设置用户自己的工作环境，如设置 PATH 变量，定义命令别名等
TC Shell	ENV	如果变量已经设置，使用变量的值作为初始化文件，执行文件中的命令，设置用户工作环境
	/etc/csh.cshrc	用于设置系统范围的工作环境，如设置 path 变量等
	/etc/csh.login	同上，用于设置系统范围的用户工作环境
	\$HOME/.tcshrc	用于设置用户自己的工作环境，如设置环境变量，定义命令别名等
	\$HOME/.cshrc	同上
	\$HOME/.login	用于定义每次注册时需要执行的命令，如使用 setenv 命令声明环境变量、设置终端类型等
	/etc/csh.logout	用于定义退出 TC Shell 时需要执行的系统范围的善后处理任务，如清除临时文件等
Z Shell	\$HOME/.logout	用于定义退出 TC Shell 时需要执行的善后处理任务，如清除自己的临时文件等
	/etc/zsh/zshenv	系统范围的初始化文件。不管是哪一种 Z Shell（注册、交互式或非交互式 Shell），总是首先运行这个初始化文件
	\$HOME/.zshenv	用户初始化文件，用于设置用户自己的工作环境，如设置命令检索路径等环境变量
	/etc/zsh/zprofile	系统范围的初始化文件，主要用于注册 Shell
	\$HOME/.zprofile	用户初始化文件，用于设置用户自己的工作环境，如设置命令别名和函数等
	/etc/zsh/zshrc	系统范围的初始化文件，主要用于交互式 Shell
	\$HOME/.zshrc	用户初始化文件，用于设置用户自己的工作环境，如设置命令历史文件等
	/etc/zsh/zlogin	系统范围的初始化文件，主要用于注册 Shell
	\$HOME/.zlogin	用户初始化文件，用于设置用户自己的工作环境，如设置终端类型等
	/etc/zsh/zlogout	用于定义退出 Z Shell 时需要执行的善后处理任务，如清除临时文件等
	\$HOME/.zlogout	用于定义退出 Z Shell 时需要执行的善后处理任务，如清除自己的临时文件等

每个 Linux 系统通常都提供若干标准的用户初始化文件，其中的默认设置基本上都能够满足用户的常规要求。如有特殊需求，如定制系统提示符等，只需在此基础上进行适当的修改，增加特定的变量设置或命令。

下面是 Ubuntu Linux 系统提供的用户初始化模板文件。

- /etc/skel/.bash\_logout
- /etc/skel/.bashrc
- /etc/skel/.profile

在利用 useradd 命令增加新的用户账号时，如果使用“-k”和“-m”选项指定了/etc/skel 目录，useradd 将会把/etc/skel/目录中的所有文件复制到用户的主目录。此时，应根据选用的命令解释程序，删除不必要的文件（如果存在）。然后，利用.profile 文件作为用户初始化文件的起点，定制自己的 Shell 工作环境。

在 Ubuntu Linux 系统的 Bash 运行环境中，用户需要设置和使用的通常只有～/.profile 和～/.bashrc 两个初始化文件。而且，如果想要增加自己的设置，应尽量加在～/.profile 文件中，因为这个文件仅在注册时执行一次，但～/.bashrc 文件则有可能多次执行，因而影响系统的性能。

### 13.2.3 定制 Shell 工作环境

#### 1. Shell 环境

每个 Shell 都会维护一组由 login 程序、系统以及用户初始化文件定义的各种变量（login 程序的功能之一就是根据初始化文件的定义，自动设置默认的 PATH 变量）。其中包括：

- 环境变量——可用于 Shell 调度执行的所有进程的变量称为环境变量。使用 env 或 set 命令可以查阅这些变量的设置情况。
- Shell 本地变量——仅对当前 Shell 有效的变量称为本地变量。在 TC Shell 中，一组 Shell 本地变量（如 user、term、home 和 path）与相应的环境变量具有特殊的关系，二者之间互相影响。

在 Bash 中，通常本地变量和环境变量均统一采用大写字母（但并不禁止使用小写字母和其他字符）。如果期望变量的设置能够用于随后调度执行的任何命令或程序，必须使用 export 命令公布用户设置的变量。

在 TC Shell 中，可以使用 set 命令和小写字母变量名设置 Shell 本地变量，使用 setenv 命令和大写字母变量名设置环境变量。如果用户设置了 Shell 本地变量，Shell 将会自动设置相应的环境变量，反之亦然。例如，如果利用新的路径更新了 path 本地变量，Shell 也将使用新的路径更新 PATH 环境变量。注意，引用变量时通常均采用大写字母形式。

在用户初始化文件中，可以采用修改预定义变量的值或指定附加变量的方式定制自己的 Shell 工作环境。要在 Bash 和 Korn Shell 的用户初始化文件中设置变量，可以使用下列命令。

```
VARIABLE=value; export VARIABLE
```

例如，为了使 vim 编辑器等能够正常地工作，需要根据自己使用的具体终端，采用“TERM=termtyp; export TERM”的命令形式定义终端的类型，以便系统能够检索 TERMINFO 数据库，确定终端的特性（注意，TC Shell 采用“setenv VARIABLE value”的命令形式设置变量，如 setenv TERM ansi）。有关变量的更多信息，请参见第 7 章。

#### 2. 设置环境变量

在数据库等应用程序中，需要用到许多环境变量。一些软件厂商通常也都提供一个环境变量设置脚本，供用户设置运行环境。

为使环境变量的设置能够用于当前的 Shell 运行环境，可以采用两种方法：一是利用“.”或 source 命令读取并执行 Shell 脚本，在当前的 Shell 中设置运行环境；二是把环境变量的设置语句加到.profile 等用户初始化文件中。



“.”或 source 命令的主要功能是在当前 Shell 的控制下，读取指定的 Shell 脚本文件，由当前的 Shell 解释执行。“.”或 source 命令的调用方法如下。

```
. script
```

或

```
source script
```

其中，script 是一个 Shell 脚本文件，文件中包含必要的环境变量设置语句。

使用“.”或 source 命令的主要目的是执行指定脚本文件中的命令，在当前的 Shell 中设置环境变量。例如，在使用 MySQL 数据库之前，用户也许需要运行诸如“mysql\_env.sh”的环境变量设置脚本，或者在.profile 文件中增加下列内容，从而设置自己的运行环境。

```
MYSQL_HOST=iscas  
MYSQL_TCP_PORT=3306  
TMPDIR=/tmp  
export MYSQL_HOST MYSQL_TCP_PORT TMPDIR
```

假定 mysql\_env.sh 是一个环境变量设置脚本，可使用“.”或 source 命令读取并运行 mysql\_env.sh 脚本文件中的语句，在当前的 Shell 中设置环境变量。即使退出脚本，环境变量的设置仍将保持有效。

```
$ . ./mysql_env.sh  
$
```

如果使用下列命令，则在退出子 Shell 或 Shell 脚本之后，当前的 Shell 环境不会留存脚本中设置的任何变量。

```
$ sh mysql_env.sh
```

或

```
$ ./mysql_env.sh
```

### 3. 设置命令检索路径

设置环境变量的一个重要内容是设置 PATH 变量。Shell 变量 PATH 用于设置命令的检索路径，定义命令所在的目录。PATH 变量包含一系列目录，目录名之间由冒号“：“分隔。目录的列举顺序确定了命令的检索顺序。为了提高系统的响应速度，常用命令所在的目录应放在靠前的位置。

假定环境变量 PATH 设置为“/usr/bin:/usr/sbin:”，则意味着命令的检索路径依次为 /usr/bin、/usr/sbin 和当前目录。指定当前目录至少有 3 种方法：一是使用两个相邻的冒号，二是在原有路径名的前部或后部增加一个冒号“：“，三是增加一个句点“.”。如果输入的命令名以斜杠“/”开始，则省略检索 PATH 变量的步骤，直接加载并执行指定的命令。否则，系统将会按照 PATH 变量指定的目录顺序，依次检索与之匹配的命令文件。

在找到指定的命令后，如果命令文件的访问权限是可执行的，而且也不是目录文件，则创建一个新的进程，执行用户输入的命令。如果指定的命令既非绝对路径，从 PATH 变量列举的目录中也找不到匹配的命令文件，系统将会输出下列错误信息。

```
bash: cmd: command not found
```

上述错误信息表明：要么命令名的输入有误，要么给定的命令确实不在 PATH 变量指定的检索目录中。对于第二种情况，校正的办法一是使用绝对路径名，二是修改 PATH 变量，把命令所在的目录加到 PATH 变量中，如附加到 PATH 变量现有值的后面。

在安装第三方的软件时，通常需要修改 PATH 变量，以免出现“命令不存在”或使用绝对路径名运行命令的麻烦。

当用户使用完整的绝对路径执行命令时，Shell 将会利用给定的路径检索命令。但当用户仅输入了命令名本身时，Shell 将会按照 PATH 变量指定的目录顺序检索命令。如果在某个目录中发现

命令，即执行该命令。用户初始化文件通常已经设定了 PATH 变量，包括命令路径的检索顺序。但是，大多数用户会修改这个变量，增加自己的命令检索路径，或者调整命令路径的检索顺序。例如，要把当前目录作为 PATH 变量中的一个命令检索路径，可以使用下列命令。

```
PATH=$PATH:.; export PATH
```

注意，如果 PATH 变量的命令检索路径设置不当，有可能导致系统响应时间过长，也可能导致系统执行不正确的命令。为了提高 PATH 变量检索的有效性，下面是设置目录检索路径的若干建议：

- 如果想增加当前工作目录“.”，通常应把当前目录作为最后一个检索路径。如果安全并非主要问题，可以把当前工作目录“.”作为第一个检索路径；
- 应尽可能地缩短命令检索路径，以减少响应时间，提高系统的性能；
- 命令的检索顺序是从左到右，经常使用的命令，其所在目录应位于检索路径的最前面；
- 确保命令检索路径不包含重复的目录，以免执行重复的检索；
- 应尽量避免检索大的目录，如果可能，应把大的目录置于检索路径的最后面；
- 如果存在，应把本地目录置于 NFS 远程目录前端，以减少出现系统挂起的机会，也减轻不必要的网络负载。

为了设置用户的默认命令检索路径，使之能够首先检索当前目录，同时把用户主目录和其他 NFS 远程安装目录也包括在检索路径中，可在用户初始化文件中增加下列内容（假定其中的 /net/project/bin 是一个 NFS 远程目录）。

```
PATH=.:$PATH:$HOME/bin:/net/project/bin
export PATH
```

#### 4. 设定语言环境

LANG 和 LC 系列环境变量用于设置特定的语言环境（locale），包括系统提示信息、格式转换、应用惯例和表达方式、字符排序原则，以及日期、时间、货币和数值的输出形式等。

通常，只需设置 LANG 环境变量，即可确定一个语言环境。实际上，在设置 LANG 环境变量时，系统也能够连带设置所有的 LC 系列变量（LC\_ALL 变量除外），因而能够提供正确的语言环境。同样，设置 LC\_ALL 变量也能同时设置其他 LC 系列变量，但 LAN 与 LC\_ALL 两个变量互不影响。此外，也可以单独设置下列 LC 系列变量，以设置部分语言环境。

- LC\_COLLATE（指定字符排序原则）；
- LC\_CTYPE（指定语言类型，包括字符类型、字符转换与多字节字符的宽度等）；
- LC\_MESSAGES（指定系统提示信息采用的语言）；
- LC\_NUMERIC（指定小数点与千分位的表达方式）；
- LC\_MONETARY（指定货币符号等）；
- LC\_TIME（指定日期与时间的表示方式）；
- LC\_ALL（意味着设定上述的所有 LC\* 变量）。

Ubuntu Linux 系统采用 POSIX 格式“xx\_YY.CHARSET”定义语言环境。其中，xx 是 ISO-639 标准定义的语言代码，YY 是 ISO-3166 标准定义的国家或地区代码，charset 是 /usr/share/i18n/locales 目录中列举的字符集之一。注意，Ubuntu Linux 系统仅支持 Unicode（UTF-8）字符集。

为了利用环境变量设置简体中文语言环境，可以把 LANG 变量设置为 zh\_CN.UTF-8 或 zh\_CN.utf8。因此，可在用户初始化文件中增加下列内容。

```
LANG=zh_CN.UTF-8; export LANG
```



要查询 Linux 系统支持的语言环境，可以运行“`locale -a`”命令。要想了解当前设定的语言环境，可以运行下列 `locale` 命令。

```
$ locale  
LANG=zh_CN.UTF-8  
LC_CTYPE="zh_CN.UTF-8"  
LC_NUMERIC="zh_CN.UTF-8"  
LC_TIME="zh_CN.UTF-8"  
LC_COLLATE="zh_CN.UTF-8"  
LC_MONETORY="zh_CN.UTF-8"  
LC_MESSAGES="zh_CN.UTF-8"  
.....  
LC_ALL=  
$
```

如果想要在中英文语言环境之间切换，可把环境变量 `LANG` 交替地设置成 `zh_CN.utf8` 或 `C`。

## 5. 默认的文件访问权限

当用户创建一个目录或文件时，系统将会赋予新建目录或文件一个默认的访问权限，这个访问权限是由用户掩码（user mask）控制的，而用户掩码则是由 `umask` 命令设置的。`umask` 命令通常位于系统提供的`/etc/profile` 初始化文件中。为了查阅当前的用户掩码设置，可以直接输入不加任何选项和参数的 `umask` 命令，示例如下。

```
$ umask  
0022  
$
```

用户掩码包含 4 组八进制的数字：

- 第 1 组数字用于设置 `suid`、`sgid` 和粘性位等特权标志（`umask` 命令中通常不考虑）；
- 第 2 组数字用于设置用户自己的访问权限；
- 第 3 组数字用于设置同组用户的访问权限；
- 第 4 组数字用于设置其他用户的访问权限。

为了确定 `umask` 命令中究竟应当使用什么用户掩码值，可以从系统默认的文件访问权限 666 或系统默认的目录访问权限 777 中减去文件或目录的实际访问权限对应的八进制数值，余数即为可用于 `umask` 命令中的用户掩码值。例如，假定用户想要把文件的访问权限设置为 644（`rw-r--r--`），而 666 与 644 的差为 022，故可以把 022 用作 `umask` 命令的参数。“`umask 000`”命令意味着把文件的访问权限设置为 666（`rw-rw-rw-`）（参见第 4 章）。

## 6. 定制命令提示符

当以命令行方式注册到 Linux 系统，或者在 GNOME 桌面打开一个终端窗口时，窗口的左边将会上出现 Shell 命令提示符，说明调用的 Shell 已经就绪，用户可以输入命令。通常，Linux 系统的提示符为 “[`u@h W]$ (PS1=[u@h W]$)”。其中，“u”表示注册的用户名，“h”表示系统的名字，“W”表示当前工作目录最后一个子目录的名字，“$”表示根据用户的身份显示“$”（普通用户）或“#”（超级用户）命令提示符标志。例如，当使用 gqxing 注册时，其命令提示符如下（为使命令的输出清晰起见，本书的示例中均省略了方括号及其中的内容，在此特别予以说明）。`

```
[gqxing@iscas ~]$
```

Linux 系统的命令提示符共分为 4 级，依次由 Shell 的内置变量 `PS1`、`PS2`、`PS3` 和 `PS4` 分别定义。当输入的命令不完整，输入的引号未对出现，或者直接在命令行上输入 Shell 编程语句时才会用到第三级命令提示符。`PS3` 和 `PS4` 分别用于 `select` 循环结构和 Shell 调试。

最常见的命令提示符主要是第一级命令提示符。对于用户来讲，系统默认的命令提示符在提供较多信息的同时，也可能会显得冗长，且易与命令的输出混杂在一起。当需要重复执行命令时，也不知道命令在命令历史记录中的索引号。因此，用户有时也许需要改变这种命令提示符。实际上，只要修改 PS1 变量的定义，即可达到改变命令提示符的目的。假定希望命令提示仅包含命令序号、当前工作目录和用户身份，可以使用下列命令予以设定。

```
$ PS1='[\! \$]\$ '
[160 ~]$
```

在上述命令提示符中，“!”表示命令在命令历史记录中的序号。这就解决了我们的问题：能够随时知道命令的序号，便于重复执行命令。例如，假定之前执行了某个较长的命令（如 find / -name [Mm]akefile -print），其命令序号为 126。如果想要重复执行这一命令，直接输入下列命令即可。

```
[166 ~]$ !126
```

当输入了部分命令，或者误输入的引号未成对出现而按下 Enter 键时，Shell 将会输出第二级命令提示符（通常为大于号 “>”）。出现这一情况时，解决的办法一是继续完成命令的输入，二是利用中断键 Ctrl-C 终止误输入的命令，使 Shell 能够继续处理新的命令。要把默认的第二级命令提示符改为 more，可以使用下列命令。

```
$ PS2=more
$
```

表 13-7 中给出了设置 Bash 命令提示符时可用的部分特殊字符。这些特殊字符可用于显示系统或 Shell 的当前状态信息，用户可根据自己的爱好予以选用。

表 13-7 Bash 命令提示符中可用的部分特殊字符

字 符	简 单 说 明
\\$	输出表示用户身份的提示符。超级用户为 “#”，普通用户为 “\$”
\!	显示当前命令在命令历史记录中的序号
\#	显示当前命令自调用 Shell 起的序号。从 1 开始编号
\a	输出提示音
\d	按照“星期 月 日”的格式显示日期，如“Mon May 08”
\h	显示系统的主机名（不包括域名）
\H	显示系统的规范主机名（包括域名）
\j	显示后台作业号（包括当前正在运行或暂停的作业）
\l	显示终端的设备名
\n	在提示信息中插入一个换行字符
\r	在提示信息中插入一个回车字符（相当于清除回车字符之前的所有提示信息）
\s	显示当前使用的 Shell
\@	以 12 小时“AM/PM”的格式显示时间，如“03:30 PM”或“10:12 上午”
\T	以 12 小时“HH:MM:SS”的格式显示时间，如上述的“3:30 PM”将会显示为“03:30:00”，没有 AM 和 PM 之区分
\t	以 24 小时“HH:MM:SS”的格式显示时间，如“15:30:00”
\A	以 24 小时“HH:MM”的格式显示时间，如“15:30”
\u	显示当前的注册用户名
\v, \V	显示 Bash 的版本信息
\w	显示当前工作目录的完整路径名
\W	显示当前工作目录最后一个子目录的名字
\`	在命令提示符中插入一个反斜杠
\[ \]	“\[”标志一个不可打印字符序列的开始，以便在命令提示符中嵌入一个终端控制序列，“\]”表示不可打印字符序列的结束



## 7. 定义命令别名

命令别名的定义仅在当前的 Shell 中有效，一旦退出 Shell，所有的命令别名定义也将随之消失。为了设置永久性的命令别名，可在\$HOME/.profile 文件中定义，不管何时注册到 Linux 系统，随时都可以直接使用。

正如第 5 章中所述，cp、rm 或 mv 等命令存在误删或覆盖原有同名文件的问题。为了防止此类情况出现，可将下列命令别名加到自己的.profile 或.bashrc 文件中（为了彻底解决这个问题，也可以由系统管理员一次性地修改/etc/skel 目录中的.profile 或.bashrc 文件，把下列命令别名定义加到其中）。

```
alias rm='rm -i'  
alias cp='cp -i'  
alias mv='mv -i'
```

注意，每当启动 Shell 时，Shell 都会读取并执行.profile 文件，如果定义的命令别名过多，Shell 的启动过程将会减慢。因此，一般不要定义太多的命令别名。

## 8. 定制用户初始化文件举例

下面以 Bash 为例，说明怎样以标准的用户初始化文件为基础，定制自己的用户初始化文件。

通常，系统提供的标准.profile 文件内容如下。

```
$ cat $HOME/.profile  
# ~/.profile: executed by the command interpreter for login shells.  
# This file is not read by bash(1), if ~/.bash_profile or ~/.bash_login  
# exists.  
# see /usr/share/doc/bash/examples/startup-files for examples.  
# the files are located in the bash-doc package.  
  
# the default umask is set in /etc/profile  
#umask 022  
  
# if running bash  
if [ -n "$BASH_VERSION" ]; then  
    # include .bashrc if it exists  
    if [ -f "$HOME/.bashrc" ]; then  
        . "$HOME/.bashrc"  
    fi  
fi  
  
# set PATH so it includes user's private bin if it exists  
if [ -d "$HOME/bin" ] ; then  
    PATH="$HOME/bin:$PATH"  
fi  
$
```

要定制自己的运行环境，修改 Shell 的检索路径，使之包含用户主目录及当前目录，设置中文语言环境，以 vi/vim 作为命令行编辑器等，可以利用任何编辑器，修改.profile，增加下列内容。

```
PATH=$PATH:$HOME/bin:  
LANG=zh_CN.utf8  
EDITOR=vi  
export PATH LANG EDITOR  
alias r='fc -e -'  
alias rm='rm -i'  
alias cp='cp -i'  
alias mv='mv -i'  
.....
```

## 13.3 增加与删除用户组

为了使相关的用户能够访问共同的资源与服务, Linux 系统支持用户组的概念。Linux 系统中的每个用户都从属于某个用户组, 同组的用户具有相同的用户组访问权限。此外, 每个用户还可以临时改换其有效用户组, 与其他用户组相关联。这种灵活性意味着用户除了可以按照工作性质或所在部门(如产品部或技术支持部)从属于一个主要用户组, 还可以根据工作的需要(如项目组)与其他用户组共享数据和访问权限。在 Ubuntu Linux 系统中, 最典型的一个例子就是, 如果一个用户从属于用户组 admin, 即可运行 sudo 命令, 执行系统管理与维护方面的系统命令。

在 Linux 系统中, 与用户组有关的所有信息通常被存储在/etc/group 系统文件中(参见表 13-8)。文件中的每一行定义一个用户组, 其语法格式如下。

```
group_name:password:gid:user_list
```

表 13-8 /etc/group 文件中的字段及其说明

字段	简单说明
group_name	用户组的名字。用户组名可由 2~8 个字符组成, 例如, 可以定义一个用户组 math, 使数学系的用户均归属于 math 用户组
password	通常为“x”。这个字段当前并无实际的意义
gid	用户组 ID。系统中的用户组 ID 必须是唯一的。同用户 ID 一样, 用户组 ID 是一个 32 位的无符号整数。每个 ID 必须位于 0~65 536 的整数范围内, 其中 0~999 保留给系统用户组使用, 自己定义的用户组 ID 应位于 1 000~65 536 的范围内。考虑到与其他系统的兼容性, 建议使用 16 位无符号整数的最大值 32 767 作为用户组 ID 的上限
user_list	用户组成员列表。其中可以包含属于当前用户组的所有用户名, 用户名之间需加逗号分隔符

下面是安装 Ubuntu Linux 系统之后系统提供的默认用户组信息。

```
$ cat /etc/group
root:x:0:
daemon:x:1:
bin:x:2:
sys:x:3:
adm:x:4:gqxing
...
lp:x:7:
mail:x:8:
...
sudo:x:27:
...
backup:x:34:
operator:x:37:
...
users:x:100:
...
syslog:x:103:
klog:x:104:
...
admin:x:115:gqxing
...
gqxing:x:1000:
$
```

要在系统中增加新的用户组, 可以手工编辑/etc/group 文件, 也可以使用 groupadd 命令。groupadd 命令的语法格式如下。



```
groupadd [-g gid [-o]] group
```

其中，“-g gid”选项用于指定新增的用户组 ID。如果忽略此选项，默认的用户组 ID 为已分配的最大 ID 号加 1。例如，如果用户组 ID 号 1000、1005 和 2000 已经分配，则默认的用户组 ID 为 2001。

如果想要增加一个名为 banking，用户组 ID 为 2000 的用户组，可以使用下列命令。

```
$ sudo groupadd -g 2000 banking  
$
```

然后，可以使用 grep、cat 命令或其他工具验证用户组是否已经被加到系统中。

```
$ grep banking /etc/group  
banking:x:2000:  
$
```

如果想要修改或删除用户组，可以分别使用 groupmod 和 groupdel 命令。因为用户组的修改和删除比较简单，故此处不再赘述。对于管理和维护用户组而言，最简单的方法其实是利用编辑器，手工编辑/etc/group 文件。

在实际应用过程中，用户可以使用 newgrp 命令改变自己的有效用户组 ID。例如，如果希望在交互式会话中将自己的有效用户组 ID 从主用户组改为次用户组，以确保创建的文件都能够与正确的用户组 ID 相关联，需要使用 newgrp 命令。假定超级用户 root 具有下列主用户组成员关系。

```
# id  
uid=0(root) gid=0(root) groups=0(root)  
#
```

如果超级用户 root 希望改变其有效用户组 ID，以其他用户组（如 sys）成员的身份工作，可以使用下列 newgrp 命令。

```
# newgrp sys  
# id  
uid=0(root) gid=3(sys) groups=0(root),3(sys)  
#
```

之后，超级用户 root 的有效用户组 ID 将会变为 sys，这意味着之后创建的文件或生成的进程都会与新的用户组 sys 相关联。因此，在使用 newgrp 命令之后，超级用户 root 执行的任何操作，如创建文件，其用户组是 sys 而非 root。

例如，在更改有效用户组 ID 之前，如果超级用户 root 使用主用户组创建了一个新文件，那么，与新文件相关联的用户组就是 root，如下所示。

```
# touch root.txt  
# ls -l root.txt  
-rw-r--r-- 1 root root 0 Sep 26 01:12 root.txt  
#
```

如果超级用户 root 把用户组 ID 改为 sys 之后又创建了一个新文件，那么，与该文件相关联的有效用户组将是 sys 而非 root，如下所示。

```
# newgrp sys  
# touch sys.txt  
# ls -l sys.txt  
-rw-r--r-- 1 root sys 0 Sep 26 01:15 sys.txt  
#
```

## 13.4 监控用户

用户管理的一个重要目的就是保证用户的活动总是处于一个正常的范围之内。如果某个用户正在过度地占用 CPU 资源，则可以使用户进程的优先级别（参见第 9 章）。从系

从安全的角度考虑，监控用户的活动可能会发现一些未经授权或非正常的用户行为。一旦发现诸如此类现象，可以使用“passwd -l”命令封锁相应用户的注册账号。

### 13.4.1 利用 who 命令查询系统中的用户

要查询当前系统中都有哪些用户，这些用户都来自哪里，可以使用 who 命令。who 命令能够输出当前已注册到系统中的所有用户列表，其语法格式如下。

```
who [-abdHlmprstTu] [file]
```

其中，“-a”选项表示同时选用其他多个选项，从而给出较多的信息，“-H”选项表示在输出数据之上增加一个标题行，“-r”选项用于查询系统当前所处的运行级。如果未指定文件名，表示读取并显示默认的/var/run/utmp 文件中的内容，即当前系统中的用户注册与活动信息。要查询早期的历史记录，可以使用/var/log/wtmp 文件。

who 命令的输出数据通常包括下列字段。

```
NAME LINE TIME [IDLE] [PID] COMMENT [EXIT]
```

其中，NAME 字段表示注册的用户名，LINE 字段表示用户使用的终端设备名，TIME 字段表示用户注册的起始时间，IDLE 字段表示用户自上一个处理活动以来的空闲时间，PID 字段表示用户的进程 ID，COMMENT 字段通常给出用户所在系统的系统名，EXIT 字段表示用户的退出状态。

例如，下列信息表明当前系统中只有 3 个用户，3 个用户均来自控制台终端（其中后两个用户是由于打开终端窗口注册的）。

```
$ who
gqxing  tty7      Sep 25 22:50 (:0)
gqxing  pts/0      Sep 26 01:38 (:0.0)
gqxing  pts/1      Sep 26 01:15 (:0.0)
$
```

如果使用“-H”选项，将会在输出数据之上增加一个标题行，以便用户了解每一行输出数据的意义，示例如下。

```
$ who -H


| NAME   | LINE  | TIME         | COMMENT |
|--------|-------|--------------|---------|
| gqxing | tty7  | Sep 25 22:50 | (:0)    |
| gqxing | pts/0 | Sep 26 01:38 | (:0.0)  |
| gqxing | pts/1 | Sep 26 01:15 | (:0.0)  |


$
```

使用“-a”选项可以显示更多的系统与用户活动数据，示例如下。

```
$ who -Ha


| NAME   | LINE        | TIME         | IDLE               | PID  | COMMENT | EXIT |
|--------|-------------|--------------|--------------------|------|---------|------|
|        | system boot | Sep 25 22:48 |                    |      |         |      |
|        | run-level 2 | Sep 25 22:48 |                    |      | last=   |      |
| LOGIN  | tty4        | Sep 25 22:48 |                    | 4590 | id=4    |      |
| LOGIN  | tty5        | Sep 25 22:48 |                    | 4591 | id=5    |      |
| LOGIN  | tty2        | Sep 25 22:48 |                    | 4595 | id=2    |      |
| LOGIN  | tty3        | Sep 25 22:48 |                    | 4596 | id=3    |      |
| LOGIN  | tty6        | Sep 25 22:48 |                    | 4598 | id=6    |      |
| LOGIN  | tty1        | Sep 25 22:49 |                    | 5901 | id=1    |      |
| gqxing | +           | tty7         | Sep 25 22:50       | 5939 | (:0)    |      |
| gqxing | +           | pts/0        | Sep 26 00:38 00:04 | 7883 | (:0.0)  |      |
| gqxing | -           | pts/1        | Sep 26 01:15       | 7883 | (:0.0)  |      |


$
```

who 命令的另外一个用途就是确定系统当前所处的运行级，示例如下。

```
$ who -r
run-level 2 2008-09-23 20:47
$
```

其中，“run-level 2”表示系统当前所处的运行级为 2，即多用户图形界面运行模式（参见第



14 章)。“2008-09-23 20:47”表示最近一次改变系统运行级的日期和时间。“last=”表示进入当前运行级之前系统所处的运行级, 等号后为空表示系统是直接引导至当前运行级的。

### 13.4.2 利用 finger 命令查询系统中的用户

finger 命令用于查询当前注册到系统中的用户(包括本地和远程注册的用户), 输出注册用户名及全名、所用终端的设备名、空闲时间、注册时间或远程主机名等。finger 命令的语法格式简写如下。

```
finger [-lmsp] [user] [user@host]
```

finger 命令的输出数据通常包含下列字段。

```
Login Name Tty Idle Login-Time Office Office-Phone
```

其中, Login 字段表示注册的用户名, Name 字段是用户的全名, 即 passwd 文件中的注释字段, Tty 字段是用户注册时使用的终端设备名, Idle 字段表示自上一个处理活动以来的空闲时间, “Login-Time”字段表示用户注册的日期和时间, Office 字段包含用户所在系统的系统名或 IP 地址, “Office-Phone”字段给出的是用户的电话号码等信息。

例如, 下列信息表明当前系统中只有 3 个用户, 3 个用户均来自控制台终端(其中后两个用户是由于打开终端窗口注册的)。

```
$ finger
Login      Name      Tty      Idle      Login Time   Office      Office Phone
gqxing    gqxing    tty7      Idle      Sep 25 22:50 (:0)
gqxing    gqxing    pts/0      6        Sep 26 00:38 (:0.0)
gqxing    gqxing    *pts/1      Sep 26 01:15 (:0.0)
$
```

### 13.4.3 利用 w 命令查询系统中的用户活动

要查询系统中现有的注册用户, 以及每个用户当前正在做什么等信息, 还可以使用 w 命令, 其语法格式如下。

```
w [-husfVo] [user]
```

其中, “-h”选项表示禁止输出包括标题的前两行信息。在输出的信息中, USER 字段表示注册的用户名, TTY 字段表示用户使用的终端设备名, FROM 字段表示用户所在系统的名字, “LOGIN@”字段表示用户注册的起始时间, IDLE 表示空闲时间(从上一次输入命令后至今的持续时间), JCPU 字段表示从同一终端设备上运行的所有进程及其子进程所用的整个 CPU 时间, PCPU 表示当前进程使用的 CPU 时间, WHAT 字段表示当前进程的命令名及其参数。

如果未加任何选项, w 命令的输出信息如下。

```
$ w
01:23:17 up 2:35, 3 users, load average: 0.04, 0.06, 0.01
USER    TTY      FROM      LOGIN@     IDLE      JCPU      PCPU WHAT
gqxing  tty7      :0        22:50      0.00s  2:48m  0.36s /usr/bin/gnome-
gqxing  pts/0      :0.0      00:38      1:27m  0.48s  0.48s bash
gqxing  pts/1      :0.0      01:15      0.00s  0.48s  6.26s gnome-terminal
$
```

在上述命令输出信息中, 第一行信息分别表示当前的系统时间、系统已经运行了多长时间、当前有多少个注册的用户, 以及在过去的 1 分钟、5 分钟和 15 分钟内系统的平均负载。

### 13.4.4 向注册用户发送消息

Linux 系统提供了大量的工具供注册的用户之间相互进行实时通信。通常, 为注册用户发送

日常消息的方法是利用/etc/motd (message of the day) 文件。管理人员可以随时修改这个文件，一旦用户注册，即可见到 motd 文件中的内容。

但上述方法仅在用户注册时有效，对于已经注册的用户，则看不到 motd 文件修改后的内容，除非再次注册。一旦有紧急情况需要通知用户，如因停电或系统故障需要立即停机时，可利用 wall (write to all) 命令向注册的用户发送紧急通知，示例如下。

```
$ wall
This system will shut down for emergency maintenance now.
You have 5 minutes to save your work and log out.
^D
```

不管用户当前正在做什么，wall 命令将会把按下 Ctrl-D 键之前输入的文字信息立即写到用户的终端窗口中，如下所示。

```
Broadcast Message from gqxing@iscas
(/dev/pts/0) at 1:24 ...
This system will shut down for emergency maintenance now.
You have 5 minutes to save your work and log out.
```

## 13.5 插件式认证模块

插件式认证模块 PAM (Linux Pluggable Authentication Modules) 使系统管理员能够决定应用程序如何利用通用的认证技术验证用户的身份。PAM 提供一个共享模块库 (位于/lib/security 目录中)，当需要认证用户时，应用程序可以调用共享模块库。位于/etc/pam.d 目录中的每个配置文件均包含一组需要调用的模块或模块栈，这些模块决定了每个系统服务的认证方法。必要时，PAM 也可以使用其他文件，如/etc/passwd 实现用户认证。“插件式”认证模块意味着用户能够容易地增加或者从认证调用栈中删除任何模块。

在应用的开发过程中，程序员只需充分调用 PAM 提供的共享认证模块库，不必自己编写认证代码，即可实现用户认证要求。此外，对于给定的应用，系统管理员也能够利用 PAM 改变认证机制，而无需修改现有的应用。PAM 为各种系统服务提供了认证机制，如 login、ftp、su 和 sudo 等。用户也可以利用认证栈技术的方法及其灵活性，采用不同的认证机制，组合一定的认证模块，集成一个诸如 RSA、DCE 或 Kerberos 的系统认证服务。

从用户注册到执行 sudo 命令，从执行 su 命令到关闭系统，无论何时需要用户提供密码验证，系统管理员都可以使用 PAM 配置认证处理过程，而且，对使用 PAM 实现认证的所有系统服务或应用程序而言，其处理过程基本上都是相同的。对于每一个系统服务或应用程序，存储在/etc/pam.d 目录中的配置文件描述了其认证处理过程。这些配置文件通常都有一个与系统服务或应用程序相同或类似的名字。例如，login 注册认证的配置文件为/etc/pam.d/login。

在认证过程中，如果遇到问题，PAM 发出的警告信息将被记录到/var/log/messages 或 /var/log/secure 文件中，如果在修改 PAM 配置文件后没有达到预期的结果或出现异常，可以查询这些日志文件，找出问题所在。为了防止恶意的用户看到 PAM 信息，PAM 仅把错误信息写入文件而非送到终端屏幕。

### 13.5.1 配置文件、模块类型与控制标志

PAM 采用/etc/pam.conf 或 /etc/pam.d 目录中以各种服务命名的文件作为配置文件，配置文件中



的每一行表示 PAM 需要在认证过程中执行的处理动作。如果/etc/pam.d 目录存在，PAM 将会忽略 /etc/pam.conf 配置文件。/etc/pam.d 目录中每个配置文件的语法格式如下。

```
type control module-path [module-arguments]
```

其中，type 是认证模块类型标识符，用于指定后随的模块属于哪一种模块类型；control 表示认证的控制标志；取决于具体的系统，module-path 可以是调用 PAM 模块文件的完整路径名，也可以是相对于默认存储位置（/lib/security）的相对路径名；module-arguments 是可选的，表示认证模块的参数。如果存在，可以是一个或多个由空格分隔的参数，用于调整给定模块的处理功能。

下面是一个 PAM 配置文件的实例（采用 grep 命令的目的是剔除以注释符“#”为起始字符的大量注释行）。

```
$ grep '^#[^#]' /etc/pam.d/login
auth      requisite    pam_securetty.so
auth      requisite    pam_nologin.so
session   required    pam_selinux.so close
session   required    pam_env.so readenv=1
session   required    pam_env.so readenv=1 envfile=/etc/default/locale
@include  common-auth
auth      optional     pam_group.so
session   required    pam_limits.so
session   optional    pam_lastlog.so
session   optional    pam_motd.so
session   optional    pam_mail.so standard
@include  common-account
@include  common-session
@include  common-password
session   required    pam_selinux.so open
$
```

在上述配置文件中，每一行的第一个字段是一个表示模块类型的标识符，如 account、auth、password 及 session 等，第二个字段的控制标志表示一旦认证失败时 PAM 应采取的动作，从第三个字段开始是 PAM 模块（位于/lib/security 目录）的名字或选用的参数。PAM 模块库利用/etc/pam.d 目录中的配置文件名确定由哪一个模块负责相应的认证处理。以“@include”为起始字符串的配置行表示在此处插入指定的文件。表 13-9 给出了各种认证模块的分类及其说明。

表 13-9

认证模块类型

模 块 类 型	控 制 功 能 描 述
account	用于实现非认证式的账号管理。此类模块通常主要用于限制或允许访问基于时间、当前可用系统资源（如最大用户数）以及用户所在位置（如超级用户 root 只能在系统控制台注册）的服务。例如，检测用户在当前的时间点是否能够访问相应的服务
auth	采用密码和其他认证机制认证用户，授权用户使用相应的服务。此类模块提供两种用户认证模式：首先，通过提示用户输入密码或其他标识信息，获知用户的身份；其次，认证模块可根据用户提供的身份信息，赋予用户一定的用户组成员权限或其他权限
password	修改用户密码。此类模块主要用于提请用户更新密码等信息。对于每一种基于“提示/响应”的认证，通常都存在一个相应的认证模块，以实现相应的处理过程。实际上，password 类型的认证模块并不常用
session	此类模块通常主要用于会话启停管理。在服务开始运行（如用户注册）时，事先设置相应的运行环境，如记录日志，安装用户主目录等；当服务终止运行（如用户注销）时，执行相应的善后处理过程，如关闭日志，卸载用户主目录等

account 认证模块的功能类似于 auth 认证模块，其主要差别是仅当用户通过认证之后才调用

前者。account 认证模块可以作为一种附加的安全检测措施，以便在用户能够访问系统之前再做进一步的限定检测。例如，可以利用 account 认证模块强制用户只能在上班时间注册到系统等。

session 认证模块用于启动和终止一个用户会话。在 Linux 系统中，一个通用的会话模块是 pam\_mail，当用户注册到传统字符界面的系统环境时，可以使用这个模块通知用户是否有新的邮件到来。

在 PAM 配置文件中，可以采用表 13-10 所示的关键字，设置认证模块的控制标志。

表 13-10

认证控制标志

控制标志	功能说明
required	要求指定的认证模块都成功地运行之后才能报告整个认证成功。也就是说，认证栈中的所有模块都必须执行一次，PAM 才能报告整个认证究竟是成功还是失败。这种认证技术用于延迟程序调用的返回结果，直至所有的认证模块都执行完毕，从而防止黑客确切地了解究竟在哪一个认证环节出现了问题，增加系统入侵的难度
requisite	类似于 required，指定的认证模块必须都成功地运行才能保证认证栈的成功。如果存在多个认证模块，一个模块的认证成功并不能保证整个认证成功。如果某个模块的认证失败，PAM 将会停止执行后续的认证，立即返回调用程序，报告认证失败。这种认证技术相当于多项认证技术的逻辑与运算，只有所有的模块都认证成功才能满足要求。在实际运用中，这种认证方法能够防止用户在一种不安全的网络连接中仅靠输入正确的密码访问系统
sufficient	如果当前模块的认证结果已经成功，表示整个认证栈已经成功，无需再执行随后的其他认证模块。如果认证失败，也并不影响整个认证栈的最终认证结果，PAM 将会采用其他模块继续执行认证。这种认证技术相当于多项认证技术的逻辑或运算，只要认证栈中的任何一种认证成功即足以满足要求。例如，当使用 rsh 连接到另外一个系统时，pam_rhosts_auth 模块首先会检测当前的连接是否为信任的。如果连接是信任的，pam_rhosts_auth 模块将会报告认证成功，继而 PAM 也会立即报告 rsh 守护进程认证成功，此时不会提示用户输入密码；如果连接不是可信任的，PAM 再开始执行其他认证模块，提示用户输入密码。如果后者认证成功，PAM 将会忽略 pam_rhosts_auth 模块报告的认证失败结果。如果两种认证都不成功，用户注册将会遭到拒绝
optional	认证结果通常可忽略。仅当相应服务的认证模块只有一个，选用模块的认证结果才有效
include	插入指定的配置文件，逐行引用其中的所有内容

PAM 可以使用应用程序要求的各类认证模块，分别实现用户账号的状态检测、会话管理以及密码修改等认证方式。PAM 可用的认证模块通常均位于 /lib/security 目录中。

/etc/pam.d 目录中的配置文件定义了执行用户认证时需要用到的一组认证模块。同一类型的认证模块称作认证栈，PAM 将会按照配置文件中列举的顺序，从栈顶开始依次调用每一个认证模块。每个认证模块依次向 PAM 返回认证成功或失败的结果，根据认证模块的控制标志，再由 PAM 向调用程序返回整个处理过程的认证结果。

我们知道，注册过程是由 login 程序引导用户输入用户名与密码实现用户认证的。login 命令首先提示用户提供一个用户名，然后请求 PAM 开始认证用户。下面以 login 系统服务的部分认证模块略作说明（参见表 13-9 与表 13-10）。

(1) 按照 login 文件中的列举顺序，PAM 首先会调用 tty 安全模块 pam\_securetty，确保超级用户 root 只能从允许的终端（如控制台）注册。如果想要确保整个认证结果是成功的，必须保证 pam\_securetty 模块的认证结果也是成功的。在这一认证过程中，仅当有人试图从禁用的终端中以超级用户 root 的身份注册，pam\_securetty 认证模块才会报告失败。如果接受认证的用户并非超级用户，或者即使是超级用户，但却是从一个允许使用的终端注册的，pam\_securetty 模块仍会报告认证成功。注意，PAM 认证的成功与失败是一种说明性的概念，与 Linux 系统中的真假概念是完全不同的。

(2) 采用 pam\_nologin 认证模块时，如果存在 /etc/nologin 文件，系统仅允许超级用户 root 注册，而禁止其他普通用户注册。也就是说，仅当 /etc/nologin 文件不存在或当前注册的用户是 root 时，pam\_nologin 模块才能报告认证成功。

(3) @include 指定的 common-auth 文件含有专门检测注册的用户是否属于合法用户的软件模



块，作为认证栈的补充部分，用于验证用户名与密码。

### 13.5.2 修改 PAM 配置文件

有些 UNIX 或 Linux 系统要求用户必须是超级用户组（俗称 wheel 组，实际上指的是 root 用户组）中的一个成员，才能使用 su 命令。尽管 Ubuntu Linux 系统并未采用这种配置方法，但通过编辑/etc/pam.d/su 文件，仍可利用 PAM 达到同样的目的，如下所示。

```
$ cat /etc/pam.d/su
.....
# Uncomment this to force users to be a member of group root
# before they can use 'su'. You can also add "group=foo"
# to the end of this line if you want to use a group other
# than the default "root" (but this may have side effect of
# denying "root" user, unless she's a member of "foo" or explicitly
# permitted earlier by e.g. "sufficient pam_rootok.so").
# (Replaces the 'SU_WHEEL_ONLY' option from login.defs)
# auth required pam_wheel.so

# Uncomment this if you want wheel members to be able to
# su without a password.
# auth sufficient pam_wheel.so trust
.....
$
```

上述 “# auth required pam\_wheel.so” 一行前面存在一个注释符 “#” 前缀。如果删除注释符 “#”，意味着只有 root 用户组的成员才能使用 su 命令（其认证控制标志为 required）。此外，如果删除 “auth sufficient pam\_wheel.so trust” 一行中的注释符 “#”，则表示允许 root 用户组的成员在运行 su 命令时不必提供密码（其认证控制标志为 sufficient）。

注意，除非确实了解怎样配置 PAM，一般不要轻易修改/etc/pam.d 目录中的文件。修改 PAM 配置文件的错误，有可能导致无法正常访问系统。为避免此类问题，在修改之前应事先备份 PAM 配置文件，同时打开一个新的终端窗口，使用 “sudo -i” 命令进入一个超级用户的 Shell 环境。然后再仔细地测试修改后的配置文件，确保能够正常注册，新的设置能够达到预期的目的。如果出现错误，可以在打开的超级用户终端窗口，利用备份的配置文件恢复修改有误的文件。

## 13.6 超级用户与 sudo 命令

超级用户 root 拥有无限的权利，在长时间的系统访问过程中，如果运用不当，或者在无意中出现失误，有可能损害系统，如破坏文件系统等，严重时甚至可能引起系统瘫痪。有些系统特权命令还可能会侵犯用户的隐私或危及系统的安全。为了确保 Linux 系统能够安全地正常运行，Ubuntu Linux 系统不允许使用 root 直接注册。当需要以超级用户 root 的身份和权限运行特权命令，执行系统管理与维护任务时，Ubuntu Linux 系统至少提供下列 3 种途径或方法，使用户能够获得超级用户的访问权限。

- 利用 sudo 命令，可以临时获取超级用户的访问权限，执行系统管理与维护等特权命令，运行结束之后，立即恢复用户的本来身份及访问权限。但是，Ubuntu Linux 系统只允许 admin 用户组的成员运行 sudo 命令。
- 利用 su（或带有 “-i” 选项的 sudo）命令，进入超级用户的 Shell 运行环境，用户可以

在较长的时间内，以超级用户的身份和权限访问系统，直至运行 exit 命令或按 Ctrl-D 键退出。在使用 su 命令之前，首先需要为超级用户账号解锁（即为 root 用户账号设置密码）。在 Ubuntu Linux 系统中，sudo 命令完全可以替代 su 命令。

- 启用超级用户 root 的账号，使用 root 直接注册系统，在整个会话过程中，一直以超级用户的身份和权限访问系统，直至从系统中注销。

### 13.6.1 超级用户的访问控制

利用/etc/security 和/etc/security/access.conf 等配置文件，PAM 能够控制或限定哪一些用户，何时以及如何以超级用户 root 的身份注册到系统。例如，/etc/security 文件用于控制用户能够从哪些网络或终端上，以超级用户 root 的身份注册；/etc/security/access.conf 文件又进一步增加了用户与注册方式的组合控制措施。

禁止用户以超级用户 root 的身份从网上注册是 Ubuntu Linux 系统采取的一项默认的安全策略，这一安全措施是利用 PAM 安全模块与/etc/security/access.conf 文件实现的。access.conf 配置文件中包含允许以超级用户 root 身份注册的所有用户以及网络与终端访问方式。在初始安装 Ubuntu Linux 系统之后，这个文件中均为注释行，表示禁止任何用户在网络上以超级用户 root 的身份注册访问。

/etc/security/access.conf 配置文件可以被看作一个注册访问控制表，用于指定系统接受或拒绝注册的用户及其注册方式（如通过 TTY 设备、主机或网络等方式访问系统）。当用户注册到系统时，系统将会扫描这个文件，找出第一个匹配的“用户/TTY 设备”、“用户/主机”或“用户/网络”组合，根据其中的访问权限定义，可以确定究竟是接受还是拒绝用户的注册。access.conf 文件的每一行由三个字段组成，中间以冒号 “:” 作为分隔符，其语法格式如下。

```
permission:users:origins
```

其中，permission 字段可以是一个加号 “+”（表示允许相应的用户注册）或减号 “-”（表示拒绝相应的用户注册）。users 字段是一个或多个注册用户名、用户组名或 ALL（表示可以匹配任何用户或用户组）。origins 字段可以是一个或多个 TTY 设备名（表示非网络连接的注册）、主机名、域名（最前面需增加一个句点 “.” 前缀）、主机 IP 地址、网络号（最后面需附加一个句点 “.” 后缀）、网络地址加子网掩码（子网掩码可以是一个十进制的数值或 IP 地址）、ALL（表示匹配任何设备）或 LOCAL（表示匹配不包含句点 “.” 的任何字符串）。

下面的例子表示允许超级用户 root 通过 X 终端“:0”与 pts/0~pts/9 等终端设备注册访问系统。

```
+ : root : :0 pts/0 pts/1 pts/2 pts/3 pts/4 pts/5 pts/6 pts/7 pts/8 pts/9
```

下面的例子表示超级用户 root 能够从指定 IP 地址的主机中注册访问系统。

```
+ : root : 192.168.100.1 192.168.100.2 192.168.100.6
+ : root : 127.0.0.1
```

下面的例子表示，除了采用 sudo 或 su 命令获取超级用户访问权限的途径之外，禁止超级用户 root 以任何方式直接注册系统。

```
- : root : ALL
```

注意，利用 OpenSSH 网络，用户仍然能够以超级用户 root 的身份从网上注册。这是因为在 Ubuntu Linux 系统中，ssh 并不受 security 安全模块或 access.conf 文件的限制。此外，在 Ubuntu Linux 系统的/etc/ssh/sshd\_config 文件中，PermitRootLogin 已设置为 yes，这意味着允许使用 ssh 以超级用户 root 的身份从网上注册。



### 13.6.2 利用 sudo 运行特权命令

传统上，用户大都喜欢采用超级用户 root 直接注册到系统，或者使用 su 命令获取超级用户的访问权限。出于系统安全方面的考虑，Ubuntu Linux 系统强烈建议用户使用 sudo 命令运行特权命令。事实上，在安装 Ubuntu Linux 系统之后，尽管系统也提供了 root 用户账号，但由于未设置密码，从而锁住了超级用户账号，无法用以注册，也无法利用 su 命令等传统技术成为超级用户（系统启动后直接进入恢复模式时除外）。

为了使部分用户也有机会运行特权命令，处理系统管理与维护任务，Ubuntu Linux 系统提供了 sudo 命令及其相关机制，以替代传统的超级用户权限获取技术。利用 sudo 命令，能够像利用 root 注册成为超级用户一样，在命令行界面中运行任何系统管理方面的特权命令。sudo 命令的语法格式简写如下。

```
sudo [-b] [-u user] command  
sudo [-i|Lls]
```

其中，“-b”选项表示以后台方式运行指定的命令；“-i”选项表示调用超级用户 root 的注册 Shell，运行其初始化文件（如.profile），在超级用户的 Shell 环境中访问系统；“-k”选项表示重新设置 sudo 命令的时戳，这意味着，当再次运行 sudo 命令时，必须重新输入 sudo 密码；“-L”选项用于显示当前用户在/etc/sudoers 文件 Defaults 一行中能够设置的参数；“-l”选项用于显示当前用户利用 sudo 能够在本地系统中运行的命令；“-s”选项表示调用超级用户的注册 Shell（类似于“-i”选项，但不改变现有的运行环境）；“-u user”选项表示使用指定用户的身份和权限执行指定的命令，如果未加此选项，sudo 将以超级用户的身份和权限运行当前的命令；command 是准备执行的任何命令。

当需要以超级用户的身份和权限运行特权命令，执行系统管理与维护任务时，可在命令行界面中使用 sudo 命令，即在实际运行的命令前面增加一个 sudo 前缀，当提示用户输入密码时，输入 sudo 密码。密码通常会保存 15 分钟，在此期间，如果再次运行 sudo 命令，则无需提供密码。超过此限，需要重新输入密码。sudo 命令将会影响同一命令行上的所有命令，直至遇到换行符或另一个 sudo 命令。

但是，sudo 命令不能调用图形界面的程序。如果想在命令行中运行图形界面的系统管理程序，Ubuntu Linux 系统提供了一个 gksudo 命令（gksudo 又调用 sudo 命令实现用户认证），其使用方式与 sudo 命令完全相同（在 KDE 桌面环境中，可以利用 kdesu 命令启动图形界面的管理程序）。

注意，并非任何用户都能运行 sudo 命令，按照系统的默认配置，只有 admin 用户组的成员才能利用 sudo 命令运行系统特权命令。

当第一次运行 sudo 命令时，sudo 通常会提示用户输入密码（注意，此时不要输入超级用户 root 的密码）。如果输入的 sudo 密码是正确的，sudo 命令将会开始计时；如果在 15 分钟之内继续运行 sudo 命令，sudo 不会再提示用户输入密码。

下面的例子表明，普通用户 gqxing 无权直接使用 date 命令设置系统的时钟。但由于 gqxing 也是 admin 用户组的成员，故可运行 sudo 命令。当利用 sudo 运行 date 命令设置系统时钟时，sudo 命令在收到正确的密码后，便能确保用户成功地执行 date 命令。

```
$ date 122218502008  
date: cannot set date: Operation not permitted  
Mon Dec 22 18:50:00 CST 2008  
$ sudo date 122218502008  
[sudo] password for gqxing:  
Mon Dec 22 18:50:00 CST 2008  
$
```

使用“-l”选项运行 sudo 命令时，可以查询当前用户能够利用 sudo 命令运行哪些命令。由于 gqxing 是安装系统时创建的用户，因而也是 admin 用户组中的一个成员，故能够以任何用户（包括超级用户）的身份运行任何命令（包括特权命令），如下所示。

```
$ sudo -l
[sudo] password for gqxing:
User gqxing may run the following commands on this host:
    (ALL) ALL
$
```

当有若干特权命令需要以超级用户的身份和权限运行时，最简单的方法是启动一个超级用户的 Shell 运行环境，直至运行完所有的特权命令之后，再利用 exit 命令或按下 Ctrl-D 键，退出超级用户的 Shell 环境，而无需在每次执行特权命令时都输入一个 sudo 命令前缀。为此，可以使用带有“-i”选项的 sudo 命令，如下所示。

```
$ sudo -i
[sudo] password for gqxing:
# id
uid=0(root) gid=0(root) groups=0(root)
# pwd
/root
# exit
logout
$
```

在上述情况下，调用“sudo -i”命令将会进入超级用户的运行环境。如果仅想拥有超级用户的访问权限，而保持用户当前的运行环境不变，可以使用“sudo -s”命令，如下所示。

```
$ sudo -s
# id
uid=0(root) gid=0(root) groups=0(root)
# pwd
/home/gqxing
# exit
exit
$
```

### 13.6.3 sudoers 配置文件

在 Ubuntu Linux 系统中，sudo 命令根据/etc/sudoers 配置文件的设置，确定是否赋予用户访问特权命令的权限。sudoers 配置文件的初始设置不一定能够完全适应用户的需求，有时可能需要调整其配置。要修改 sudoers 文件，最好的方法是采用专用的编辑程序 visudo：

```
$ sudo visudo
```

visudo 编辑程序能够提供加锁、编辑以及语法检测等功能。通常，visudo 主要调用 nano 编辑器修改 sudoers 文件。通过设置 VISUAL 环境变量，用户也可以使用自己熟悉的编辑器，如 vi/vim 编辑 sudoers 文件，如下所示。

```
$ export VISUAL=vi
```

注意，如果直接使用 vi/vim 等文本编辑器修改/etc/sudoers 文件，当出现语法错误时，可能会导致用户无法利用 sudo 命令获得超级用户的访问权限。因此，建议用户使用 visudo 命令修改 sudoers 文件。visudo 程序能够检测语法错误。一旦出现错误，visudo 不允许用户轻易写入 sudoers 文件，并提供一个修正错误的机会，重新编辑、放弃修改而退出或者强制保存修改（这是一个比较危险的选择，visudo 标记为 DANGER!）。

/etc/sudoers 文件含有两种类型的信息：用户别名与用户权限规范的定义，每个定义占用一行。



通过增加反斜线 “\” 后缀，也可以延续到后续行上。此外，以注释符 “#” 为起始字符的行是注释行，注释行可以出现在任何位置。

### 1. 用户权限规范

在/etc/sudoers 文件中，定义用户权限规范的语法格式如下（等号 “=” 前后的空格可有可无）：

```
user_list host_list = [(runas_list)] [tag_list] command_list
```

在用户权限规范的语法格式中，每个字段的意义及用途说明如下。

- ❑ user\_list——用于指定单个或一组用户，表示当前定义的权限适用的用户对象。指定用户时可以采用用户名、用户组（增加一个百分号 “%” 前缀）和用户别名。此外，还可以使用内部定义的别名 ALL，表示所有的用户，意味着当前的权限定义适用于所有的用户。
- ❑ host\_list——用于指定当前的权限定义施加的主机对象。主机对象可以是一个或多个主机名、IP 地址或主机别名。此外，还可以使用内部定义的别名 ALL，表示所有的主机系统，意味着当前的权限定义适用于含有此 sudoers 文件的所有系统。
- ❑ runas\_list——类似于 user\_list，用于指定单个或一组用户，两者唯一的差别是指定 runas\_list 时可以采用用户 ID。当使用 “-u” 选项调用 sudo 命令时，command\_list 中定义的命令能够以这里指定的用户身份运行（同样，指定的用户可以是用户名、加百分号 “%” 前缀的用户组和用户别名）。runas\_list 是可选的，如果省略了 runas\_list，sudo 将会以超级用户 root 的身份运行 command\_list 中指定的命令。
- ❑ tag\_list——用于设置或限定某些特殊的命令。例如，可以使用 PASSWD 和 NOPASSWD 限定用户执行某些管理和维护命令时是否需要提供密码，使用 NOEXEC 能够防止任何程序启动新的 Shell（因为一旦使用 sudo 命令运行某个程序，程序在整个运行期间都会拥有超级用户的访问权限，因而也能够启动拥有超级用户特权的 Shell，最终绕过 sudoers 文件设定的任何限制）。
- ❑ command\_list——指定权限定义适用的实用程序。指定实用程序时可以采用程序的名字、存有实用程序的目录名和命令别名，而且，所有的文件目录名必须是绝对路径名。

如果命令后面附加一个空的双引号，表示运行时用户不能指定任何命令行参数和选项。此外，也可以在命令后面指定固定的参数或通配符，限制用户能够使用的参数。

下面是系统提供的 sudoers 文件，其中的 “%admin ALL=(ALL) ALL” 表示用户组 admin 中的任何成员均可利用 sudo 命令获取超级用户的访问权限，运行系统管理与维护方面的特权命令。

```
$ sudo cat /etc/sudoers
...
Defaults env_reset
# Uncomment to allow members of group sudo to not need a password
# %sudo ALL=NOPASSWD: ALL
...
# User privilege specification
root ALL=(ALL) ALL

# Members of the admin group may gain root privileges
%admin ALL=(ALL) ALL
$
```

下列用户权限规范使用户 cathy 能够在含有此 sudoers 文件的所有系统（ALL）中使用 sudo 运行 mount 和 umount 命令，安装和卸载文件系统。

```
cathy ALL=(root) /bin/mount, /bin/umount
```

runas list 是可选的，如果省略了 runas list，sudo 允许普通用户以超级用户 root 的身份运行

用户权限规范中定义的命令。在下例中，cathy 用户可以利用 sudo 命令，在命令行中使用 umount 命令卸载/music 文件系统。

```
$ umount /music
umount: only root can umount /dev/sdb7 from /music
$ sudo umount /music
[sudo] password for cathy:
$
```

如果把上述的 sudoers 文件改写如下，则意味着不允许用户 hwang 卸载/music 文件系统，但可以使用 mount 和 umount 命令安装或卸载其他任何文件系统。

```
hwang ALL=(root) /bin/mount, /bin/umount, !/bin/umount /music
```

在此情况下，当用户 hwang 使用 sudo 运行下列 mount 命令时，将会遭到拒绝。

```
$ sudo umount /music
Sorry, user hwang is not allowed to execute '/bin/umount /music' as root on localhost.
$
```

## 2. 别名

别名机制用于定义或组合一组用户、主机或命令等，以便在 sudoers 文件中引用，其语法格式如下。

```
alias_type alias_name = alias_list
```

其中，alias\_type 用于定义不同类型的别名，如 User\_Alias、Runas\_Alias、Host\_Alias 及 Cmnd\_Alias 等；alias\_name 是一个便于引用的别名，按惯例，别名通常采用大写字母命名；alias\_list 是由一个或多个名字（如用户名、主机名和命令等）组成的名字列表，列举的名字之间需加空格或逗号“，”分隔符。别名定义中也可以嵌套其他别名，例如，在指定用户时可以使用先前定义的用户别名。系统提供一个内部定义的别名 ALL，可以匹配其所在位置的任何对象，包括用户名或系统等。例如，如果在用户名位置使用 ALL，则意味着匹配所有的用户；如果某个名字前面加有感叹号“！”前缀，意味着取其反义。

- User\_Alias——用于定义用户别名，以便引用一组限定的用户。指定的用户可以是单个用户名或系统中的用户组（用户名前面需增加一个百分号“%”前缀，如%admin，表示用户组 admin 中的所有成员）等。下面的例子表示 sudoers 文件定义了 3 个用户别名：OFFICE、ADMIN 和 ADMIN2。用户别名 OFFICE 包括 cathy、hwang 和 wzhang 3 个用户，用户别名 ADMIN 包括 wzeng、qchen 以及 admin 用户组的所有成员，用户别名 ADMIN2 包括除 dzhao 之外的所有 admin 用户组成员，如下所示。

```
User_Alias OFFICE = cathy, hwang, wzhang
User_Alias ADMIN = wzeng, qchen, %admin
User_Alias ADMIN2 = %admin, !dzhao
```

- Runas\_Alias——也用于定义用户别名，以便指定的用户能够以超级用户之外的其他用户身份运行指定的命令。下面定义的用户别名 GUEST 表示用户能够以 guest 或 visitor 的用户身份运行指定的命令。

```
Runas_Alias GUEST = guest, visitor
```

- Host\_Alias——用于定义主机别名，以便引用一组限定的主机。仅当在多个主机上运行的 sudo 命令引用的是同一 sudoers 文件时，主机别名才有意义。如果想要采用规范域名，如 iscas.abc.net，而非 iscas，必须设置 fqdn 标志。注意，在定义主机别名时采用规范域名会影响 sudo 命令的运行效率。下面定义的主机别名 TESTING 包括 iscas 与 sinosoft 两个主机系统。

```
Host_Alias TESTING = iscas, sinosoft
```

- Cmnd\_Alias——用于设置命令别名，以便引用一组限定的命令。下列命令别名定义了 3 个命令文件和一个目录（其后面附加了一个斜线字符“/”，表示目录中的所有命令文件）。



```
Cmnd_Alias BASIC = /bin/cat, /usr/bin/vim, /bin/df, /usr/local/bin/
```

下面的例子说明了怎样使用别名。在使用 sudo 命令运行 shutdown、halt、poweroff 等命令关闭系统时，通常需要提供 sudo 密码。为了避免总是输入密码，可以定义下列命令别名。

```
Cmnd_Alias SHUTDOWN_CMDS = /sbin/shutdown, /sbin/halt, /sbin/poweroff
```

然后，在 sudoers 配置文件中 “%admin ALL=(ALL) ALL” 一行的后面再额外增加下列权限规范定义即可。

```
%admin ALL=(ALL) NOPASSWD: SHUTDOWN_CMDS
```

此外，还可以彻底禁止运行 sudo 命令时提示用户输入密码。如前所述，第一次使用 sudo 命令时，用户必须提供密码，才能运行随后的系统管理与维护命令。密码的有效持续时间是 15 分钟，如果超过此限，再次使用 sudo 命令时仍须提供密码。为了继续使用 sudo 命令获取超级用户的访问权限，但禁止 sudo 命令提示用户输入密码，可以编辑 sudoers 文件，在文件的后面增加下列配置。

```
user_or_group_name ALL=NOPASSWD: ALL
```

例如，下述设置表示，用户 gqxing 任何时间运行 sudo 命令时，均不需要提供密码。

```
gqxing ALL=NOPASSWD: ALL
```

### 3. 默认值

利用关键字 Defaults，可以修改配置选项的默认值。在 sudoers 文件中，大多数配置选项的值都是布尔值 on 或 off，部分配置选项的值可以是一个字符串。在一个 Defaults 定义行中，增加一个配置选项表示启用相应的标志，如果在配置选项前增加一个感叹号 “!” 前缀则意味着关闭相应的标志。例如，sudoers 初始文件中的下列 Defaults 定义行表示启用 env\_reset 标志。

```
Defaults env_reset
```

下面是部分常用的配置选项及其默认值（完整地说明可参考 sudoers 手册页）。

- **fqdn=on/off** 如果想在 sudoers 文件中使用规范域名，应当启用这个标志，以便能够调用 DNS 服务器实现主机的域名解析。在此情况下，可以混合使用简单的主机名或规范的域名。采用规范域名的缺点是影响 sudo 运行的性能，尤其是当 DNS 系统工作不正常时。一旦设置这个标志，必须引用主机的实际域名，而不能使用 CNAME 记录定义的别名。注意，如果 hostname 命令返回的是规范域名，不需要设置这个标志。这个标志的默认值是 on。
- **passwd\_tries=num** 设置允许用户尝试输入密码的次数。当输入的密码不正确时，sudo 允许用户尝试输入其他密码，超过此限，sudo 不会再提示用户，并立即退出。默认值是 3。
- **rootpw=on/off** 设置这个标志之后，当调用的 sudo 命令提示用户输入密码时，只能输入超级用户 root 的密码。由于 sudo 命令输出的提示信息完全相同，启用这个标志后可能会给用户造成混淆。此外，如果超级用户 root 没有解锁（即未设置密码），不要启用这个标志，否则无法运行 sudo 命令。一旦出现此类问题，可以把系统引导至恢复模式，然后关闭这个标志。这个标志的默认值为 off，这意味着 sudo 提示的是安装系统时创建的第一个用户的密码。如果 rootpw 设置为 on，且已经为超级用户 root 设置了密码，当调用要求提供密码的图形界面程序时，也需要提供超级用户 root 的密码。
- **shell\_noargs=on/off** 如果设置了这个标志，当未加任何参数调用 sudo 命令时，系统将会调用超级用户 root 的 Shell，但不改变当前的运行环境，其效果等同于使用 “-s” 选项调用 sudo 命令。这个标志的默认值是 off。
- **timestamp\_timeout=mins** 用于设置 sudo 命令密码保持有效的最长时间长度（分钟数）。这

个时效的默认值是 15。如果把这个值设置为 -1，表示密码的时效没有限制。

### 13.6.4 admin 用户组成员的访问权限

在初始安装 Ubuntu Linux 系统之后，sudoers 文件包含下列用户权限规范。

```
# Members of the admin group may gain root privileges
%admin  ALL=(ALL)  ALL
```

在上述用户权限规范的定义中，等号 “=” 左边的 ALL 表示这个权限规范适用于所有的主机系统。正如其注释行所言，%admin 表示 admin 用户组的所有成员都能够以任何用户（中间的 ALL）的身份运行任何命令（最后面的一个 ALL），尤其能够利用 sudo 命令，以超级用户的身份和权限运行任何特权命令。

此外，如果使用 “(root)” 替代 “(ALL)” 或省略 “(ALL)”，用户仍然能够使用超级用户的权限运行任何命令，但不能使用 “-u” 选项以其他用户的身份运行命令（如果拥有超级用户的访问权限，这个限制通常不会造成问题）。

安装系统时创建的第一个用户，除了同名的用户组，也会被自动加到 admin 用户组中，因而具有运行系统特权命令，执行系统管理任务的访问权限。

### 13.6.5 直接使用 root 注册

在初始安装的 Ubuntu Linux 系统中，超级用户 root 的账号没有设定密码，因而不能使用 root 直接注册。如果想要运行系统特权命令，唯一的方法就是在实际运行的命令之前增加 sudo 前缀。若想在注册界面中直接使用超级用户 root 注册，首先需要启用超级用户的账号，即为 root 用户账号设置密码。为此，可在终端窗口中输入下列 sudo 和 passwd 命令，根据提示首先输入 sudo 密码，然后设置超级用户的密码。

```
$ sudo passwd root
[sudo] password for gqxing:
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
$
```

其次，使用 sudo 命令和任何文本编辑器，修改 /etc/gdm/gdm.conf 配置文件，把其中的 “AllowRoot=false” 一行改为 “AllowRoot=true”。

此外也可以在 GNOME 桌面环境中，选择“系统→系统管理”菜单，在“认证”对话框中输入 sudo 密码，在“登录窗口首选项”界面中单击“安全”标签，然后勾选中“允许本地系统管理员登录”复选框。

再次注册时就能够以超级用户 root 的身份直接注册了。之后，如果需要再次封锁超级用户 root 的账户，可以使用下列命令。

```
$ sudo passwd -l root
$
```

### 13.6.6 以不同的用户身份访问系统

总是使用 sudo 运行特权命令也并非最佳的选择。在一个单用户的个人系统中，经常使用 sudo 命令与审慎地使用 su 命令并没有太大的差别。但在一个多用户的共享系统中，尤其是在一个集中管理的服务器中，使用 sudo 命令可能比使用 su 命令更安全一些。作为一个 UNIX/Linux 系统的资



深用户，如果不习惯或根本就不喜欢使用 sudo 命令，简单易行的方法是为 root 用户账号设置一个密码，从而能够随时使用 su 命令，进入超级用户的 Shell 运行环境。

su 命令的主要功能是改变用户的身份，使普通用户成为超级用户，当然也可以改换为其他普通用户。为了使用 su 命令获取超级用户的访问权限，必须首先为超级用户 root 设置密码，从而解除对超级用户账号的封锁。

利用 su (substitute user) 命令，能够以指定的用户或超级用户 root (默认值) 的身份和权限，调用新的注册 Shell，在新的运行环境中访问系统。通过运行 exit 命令或按下 Ctrl-D 键，可以返回先前的 Shell 工作环境。当然，这样做的前提是知道超级用户 root 或其他用户的密码。

单独运行 su 命令而不加任何选项与参数时，系统将会调用具有超级用户权限的 Shell，进入超级用户的 Shell 工作环境，改变当前用户的用户 ID 与用户组 ID，但会保持之前的工作目录。下面的例子说明了用户 cathy 使用 su 命令前后的环境变化情况。

```
$ id  
uid=1001(cathy) gid=1001(cathy) groups=1001(cathy)  
$ echo $PATH  
/usr/local/bin:/usr/bin:/bin:/usr/games  
$ pwd  
/home/cathy  
$ su  
Password:  
# id  
uid=0(root) gid=0(root) groups=0(root)  
# echo $PATH  
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games  
# pwd  
/home/cathy  
#
```

当运行 “su-”、“su-l” 或 “su—login” 形式的命令时，可以启动超级用户的 Shell 工作环境，直接进入超级用户的主目录，如同使用 root 注册一样。不仅是用户 ID 与用户组 ID，整个运行环境也完全等同于超级用户 root。下面的例子说明了用户 hwang 使用 “su-” 命令前后的环境变化情况。

```
$ id  
uid=1002(hwang) gid=1002(hwang) groups=44(video),1002(hwang)  
$ echo $PATH  
/usr/local/bin:/usr/bin:/bin:/usr/games  
# pwd  
/home/hwang  
$ su -  
Password:  
# id  
uid=0(root) gid=0(root) groups=0(root)  
# echo $PATH  
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin  
# pwd  
/root  
#
```

使用 “-c” 选项运行 su 命令时，普通用户能够以超级用户的身分及其访问权限执行特权命令，在命令运行结束后，立即返回原来的 Shell 环境。下例表明普通用户无法使用 kill 命令终止一个进程，但当使用 “su -c” 命令并提供超级用户的密码之后，即可正常执行 kill 命令（注意，“-c” 选项后面的命令、选项及其参数前后必须加双引号）。

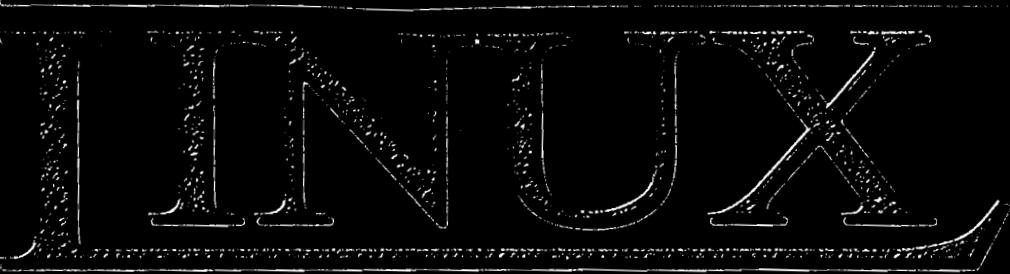
```
$ kill -15 5575  
-bash: kill: (5575) - Operation not permitted  
$ su -c "kill -15 5575"  
Password:  
$
```

# 第14章

## 系统启动与关机

Linux 系统的启动是一个复杂的内部引导过程。本章将从 GRUB 引导程序与磁盘分区，以及加电自检开始，讨论系统的引导与生成过程，最后还将介绍系统的关机过程。掌握系统引导过程的工作原理，不仅有助于理解系统引导过程中出现的问题，修复与系统启动有关的故障，也有助于了解从何处入手，定制应用程序的启动与关闭脚本。其内容主要包括：

- 磁盘分区与 GRUB；
- 初始引导过程；
- 系统生成过程；
- login 进程；
- 系统关机过程。





Linux 系统的启动涉及系统的磁盘分区、加电引导、运行初始引导程序及 Linux 系统引导程序等一系列复杂的引导与处理过程。在加电进行系统自检之后，计算机开始加载并执行系统硬盘第 0 号扇区中的初始引导程序，根据 0 号扇区的定义，确定引导哪一个操作系统。然后，加载并启动操作系统提供的引导程序，检测系统的硬件配置，设置操作系统内核的硬件运行环境，加载操作系统内核，执行系统初始化，加载软件模块，组织系统的内部数据结构，运行系统启动脚本，建立多用户运行环境，直至最终完成系统的启动，其间经过一系列复杂的处理过程。

下面以 Ubuntu Linux 在 Intel x86 硬件系统上的实现为例，从磁盘分区和 GRUB 引导程序开始，介绍系统的引导过程，讨论系统的生成过程，从而说明系统的整个启动过程。最后介绍系统的关机过程。

## 14.1 磁盘分区与 GRUB

在 Intel x86 体系结构的计算机系统中，系统磁盘的划分如图 14-1 所示。其中，位于第 0 号扇区的 512 字节数据块为主引导记录（Master Boot Record，MBR）。GRUB 使用这个扇区存储第一阶段引导程序（stage1）。紧随第 0 号磁道的第 1~62 号扇区（31KB）可以用作第 1.5 阶段引导程序的存储区。从第 63 号扇区开始是操作系统分区。

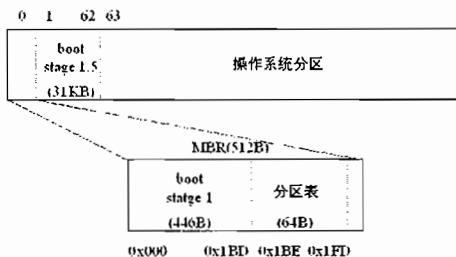


图 14-1 系统磁盘布局

### 14.1.1 磁盘分区

512 个字节的主引导记录可以分为 3 部分内容。其中前 446 个字节为计算机系统的初始引导程序，也称第一阶段引导程序。中间的 64 个字节为磁盘分区表，用于描述 4 个操作系统的分区信息，包括分区引导标志（说明是否可以引导）、每个操作系统分区的起始位置与容量，以及分区的系统类型。最后一部分只占 2 个字节，用于存储主引导记录扇区的标志代码，其中的 0xAA55 表示当前的 0 号扇区即为主引导记录，如图 14-2 所示。

在磁盘分区表中，如果分区引导标志为 0x0，说明相应的分区为非活动分区；如果引导标志为 0x80，说明相应的磁盘分区是可引导的。至于究竟启动哪一个分区中的操作系统，取决于分区表中每个分区的引导标志或用户在初始启动过程中给出的选择。操作系统类型字段表示相应磁盘分区的系统性质，如 0x82 表示 Linux swap 分区，0x83 表示 Linux 引导分区，0x85 表示 Linux 扩展分区，0x8e 表示 Linux LVM 分区，0x07 表示 Windows NTFS 分区，0x0b 表示 Win95 FAT32 分区，0x0c 表示 Win95 FAT32 (LBA) 分区，等等。

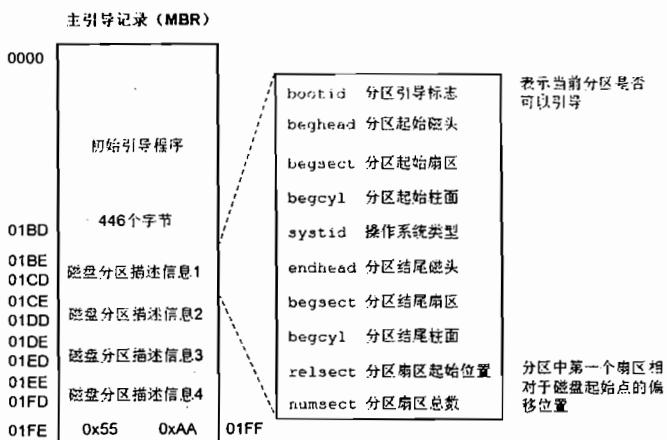


图 14-2 系统硬盘主引导块

如果系统中仅仅安装了一个 Ubuntu Linux 操作系统，即除了 0 号磁道之外，其他所有的存储空间均为 Linux 系统分区，其典型的磁盘空间布局结构如图 14-3 所示（假定在安装 Ubuntu Linux 系统时采用的是默认的磁盘分区布局，即没有额外划分其他文件系统分区）。

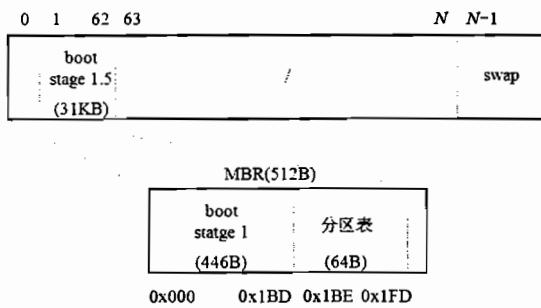


图 14-3 Linux 磁盘分区布局结构 (1)

如果计算机中安装了多个系统，如安装 Windows 之后又安装了 Ubuntu Linux 系统，其系统磁盘分区布局的典型结构如图 14-4 所示（假定在安装 Ubuntu Linux 系统时采用的是默认的磁盘分区布局，即没有额外划分其他文件系统分区）。

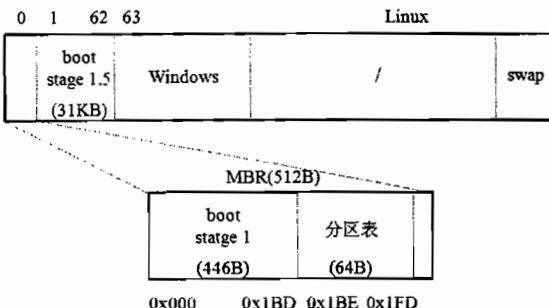


图 14-4 Linux 磁盘分区布局结构 (2)



从图 14-4 可知，第 1~62 号数据块为 GRUB stage1.5 引导程序区，其中可以存储一个中间阶段的引导程序，如 `e2fs_stage1_5`。

### 14.1.2 GRUB

Ubuntu Linux 采用 GRUB 作为引导程序。GRUB (Grand Unified Bootloader, 大一统引导管理器) 是一个通用的引导程序，使用户能够在引导系统时选择启动的操作系统，或者选择同一操作系统的不同版本。GRUB 是目前应用最广泛的引导程序，许多 Linux 系统现均采用 GRUB 作为默认的引导程序。

GRUB 提供 3 个功能强大的用户界面，每个界面都允许用户直接引导操作系统，在系统启动期间甚至可以在 GRUB 的 3 个界面之间切换。

第一个是菜单界面，采用 GRUB 的 Linux 系统都将 GRUB 菜单作为默认的引导界面。当安装完毕，计算机引导时，菜单界面就会出现在屏幕上，利用上下箭头选择需要引导的系统，按下 Enter 键即可。如果长时间没有用户输入，GRUB 会自动引导默认的操作系统。

第二个是菜单配置编辑器界面，在引导操作系统的菜单中按下“e”键，便可进入菜单编辑器。此时，可以临时性地修改操作系统引导菜单。例如，按下“o”键便可以在当前行后面增加引导项，按下“O”键可以在当前行前面增加引导项，使用“d”键可以删除引导项，使用“e”键可以编辑引导项等。修改后，可以按 Enter 键确认或按 Esc 键取消修改。

注意，在编辑操作系统引导菜单期间所做的任何修改都是临时性的，在下次引导时，这些修改将会消失（如果想要永久性地修改引导菜单，需要编辑`/boot/grub/menu.lst`文件）。在测试新编译的系统内核时，这个界面比较有用。

第三个是命令行界面，在引导操作系统的菜单中按下“c”键，便可进入命令行界面。命令行是 GRUB 最基本的界面，也是最灵活的界面。这个界面是一个小型的 Shell 环境，具有 Bash 的部分功能特性。

GRUB 采用下列形式命名存储设备及其分区。

(*<device-type><device-number>*,*<partition-number>*)

其中，*<device-type>*表示设备的类型。最常用的设备类型是 `hd`（表示 IDE 磁盘）、`sd`（表示 SCSI 磁盘或 USB 移动硬盘）、`ed`（表示 ESDI 磁盘）、`xd`（XT 磁盘）和 `fd`（表示 3.5 英寸软盘）。其他设备类型还有 `nd`（表示网络）和 `cd`（表示 CD/DVD）等。

*<device-number>*是 BIOS 能够识别的设备号，从 0 开始编号。第一个磁盘的设备编号为 0，第二个磁盘的编号为 1，如此类推。例如，GRUB 把第一个 IDE 磁盘称作 `hd0`，第二个 IDE 磁盘称作 `hd1`。这种设备命名和编号方式类似于 Linux 系统早期内核采用的设备命名方式。例如，Linux 系统内核使用的设备名`/dev/hda`等价于 GRUB 的 `hd0`，`/dev/hdb` 等价于 `hd1`。

*<partition-number>*用于指定设备分区编号。同 BIOS 设备号一样，设备分区编号也是从 0 开始的。例如，GRUB 把第一个磁盘的第一个分区称作“`(hd0,0)`”，把第二个磁盘的第三个分区称作“`(hd1,2)`”，等等。

实际上，在命名和引用设备与分区时，GRUB 并不区分 IDE 硬盘和 SCSI 硬盘，所有的硬盘均以 `hd` 命名。当指定整个硬盘，且不考虑其分区时（如需要将 GRUB 安装到一个硬盘的主引导记录时）只需将逗号“,”和分区号去掉即可。

### 14.1.3 GRUB 配置文件

在 Ubuntu Linux 系统中，`/boot/grub/menu.lst` 是默认的 GRUB 配置文件，用于创建 GRUB 的操作系统引导选择菜单界面，以便用户能够选择引导不同的系统。实际上，这个配置文件相当于一个 GRUB Shell 脚本文件，其中包括一系列 GRUB 命令或配置参数，使 GRUB 能够知道在引导系统时选用哪一个系统内核，同时指出了相关的文件及命令选项。此外也用于组织在初始引导时显示的操作系统引导选择菜单。除了以注释符“#”为起始字符的注释行之外，`menu.lst` 文件主要由下列 3 部分内容组成。

- 全局命令或配置参数。其中，`timeout` 用于控制 GRUB 在用户选择启动哪一个操作系统之前等待多长时间再自动引导默认的系统，`default` 用于指定默认的菜单项顺序号（菜单项从 0 开始编号）。
- 默认的引导选项。其中，“`memtest86=true/false`”表示是否应当在操作系统引导菜单中增加一个 `memtest86` 引导选项，用于测试系统内存。“`savedefault=true/false`”表示是否应使用 `savedefault` 设定默认的引导选项等。
- 操作系统引导菜单项。每个引导菜单项以关键字 `title` 开始，至下一个 `title` 关键字或文件结束标志为止。其中的命令或配置参数告诉 GRUB，在选定引导的操作系统之后，加载哪一个系统内核与内存映像文件，使用哪一些引导选项，等等。

表 14-1 给出了部分配置参数及其简单说明。

表 14-1 配置参数

配置参数	简  单  说  明
<code>default number</code>	指定引导的默认操作系统。在 <code>menu.lst</code> 文件中，每个 <code>title</code> 命令表示一个操作系统引导菜单项（从 <code>title</code> 命令开始，直至下一个 <code>title</code> 命令或文件终止符之前的所有命令用于完整地描述一个菜单项），菜单项从 0 开始编号，0 表示第一个操作系统菜单选项。在后述的例子中，第一个操作系统菜单选项为“Ubuntu 8.10, kernel 2.6.27-9-generic”
<code>timeout number</code>	引导时的默认等待时间，这一时间使用户有机会选择引导不同的操作系统。10 表示默认的等待时间为 10 秒
<code>hiddenmenu</code>	禁止 GRUB 在引导时显示操作系统选择菜单。超时后，立即引导默认的操作系统。在此期间，按下 Esc 键或空格键仍可看到标准的 GRUB 菜单界面
<code>title string</code>	指定准备引导的系统内核或操作系统名称
<code>root (device)</code>	指定“/”文件系统所在的磁盘与分区，如“ <code>root (hd0,2)</code> ”，表示系统内核位于第一个磁盘的第三个分区。如果 <code>/boot</code> 目录位于单独的磁盘分区，即存在单独的 <code>/boot</code> 文件系统。 <code>root</code> 指定的应是 <code>/boot</code> 文件系统对应的磁盘与分区
<code>kernel path parameter</code>	指定需要引导的系统内核或操作系统，其中的文件名必须是位于 <code>root</code> 配置参数指定文件系统分区中的一个绝对路径名，如 <code>/boot/vmlinuz-2.6.27-9-generic</code>
<code>initrd path</code>	指定引导系统时使用的初始磁盘内存映像，如 <code>/boot/initrd.img-2.6.27-9-generic</code> 文件。当系统内核需要一定的模块才能适当地引导时， <code>initrd</code> 是非常必要的。例如，如果 <code>root</code> 所在磁盘分区的文件系统为 Ext3 文件系统时，就需要用到 <code>initrd</code> 映像
<code>savedefault [number]</code>	把当前的菜单项（即操作系统）或指定的数字对应的菜单项设置为下一次启动系统时的默认引导项
<code>makeactive</code>	把主盘的活动分区设置为 GRUB 的 <code>root</code> 设备。这个命令仅适应于磁盘的主分区
<code>chainloader path</code>	使用指定的文件（文件必须是绝对路径名）作为下一个级联引导程序。如果指定的文件位于指定磁盘分区的第一个扇区中，可以采用数据块记号“+1”作为文件名，表示使用相应磁盘分区第一个扇区中的内容作为引导程序



此外，Linux 系统内核还可以接受一定的命令行选项或引导参数，如 `root=value`、`ro`、`rw`、`single` 或 `init=value` 等，以便系统内核能够正确地确定硬件配置等参数信息，解释如下。

- `root=value` —— 指定用作“/”文件系统的设备或磁盘分区名。如果`/boot`目录位于单独的磁盘分区，即存在一个单独的`/boot`文件系统，`root`参数指的是`/boot`文件系统所在的设备或磁盘分区。因此，`root`参数指定的设备必须是一个可安装的设备或磁盘分区，其中含有适当的“/”（或`/boot`）文件系统。利用`root`引导参数，使GRUB能够知道在引导其间采用哪一个设备或磁盘分区作为“/”文件系统。通常，`root`的参数值是系统内核所在的设备或磁盘分区，且在编译时就已经事先确定。如果想要修改这个参数值，如使用系统盘的第3个磁盘分区引导Linux系统，可以使用“`root=(hd0,2)`”作为引导参数。
- `ro`或`rw` —— `ro`表示先以只读方式安装磁盘分区对应的“/”文件系统，使`fsck`能够检测处于静止状态的文件系统。`rw`选项表示以读写方式安装“/”文件系统（这是系统内核采用的默认安装选项）。
- `quiet` —— 表示只需输出简短明了的信息，以便用户能够清晰地了解系统引导期间执行的动作。
- `splash` —— 表示在系统引导期间使用指定的徽标作为背景图像。
- `locale` —— 设置系统引导期间使用的语言环境。例如，“`locale=zh_CN`”表示把系统启动过程中的语言环境设置为简体中文。
- `single` —— 如果存在，`single`表示把系统引导至单用户的恢复模式（recovery mode）。
- `init` —— 设置系统内核完成系统引导过程之后调用的第一个进程。如果未设置或设置的程序文件不存在，系统内核将会按照`/sbin/init`、`/etc/init`、`/bin/init`和`/bin/sh`顺序，尝试调用发现的第一个程序。如果所有的尝试均失败，则系统只能转入瘫痪状态。

下面是安装Ubuntu Linux系统后生成的GRUB配置文件。

```
$ cat /boot/grub/menu.lst
...
default      0
...
timeout      10

## hiddenmenu
# Hides the menu by default (press ESC to see the menu)
#hiddenmenu
...
title        Ubuntu 8.10, kernel 2.6.27-9-generic
uuid         4c5ea7fa-d041-4128-a78a-43e171e8df76
kernel       /boot/vmlinuz-2.6.27-9-generic root=UUID=4c5ea7fa-d041-4128-a78a-43e171e8df76
ro locale=zh_CN quiet splash
initrd       /boot/initrd.img-2.6.27-9-generic
quiet

title        Ubuntu 8.10, kernel 2.6.27-9-generic (recovery mode)
uuid         4c5ea7fa-d041-4128-a78a-43e171e8df76
kernel       /boot/vmlinuz-2.6.27-9-generic root=UUID=4c5ea7fa-d041-4128-a78a-43e171e8df76 ro
locale=zh_CN single
initrd       /boot/initrd.img-2.6.27-9-generic
...
title        Ubuntu 8.10, memtest86+
uuid         4c5ea7fa-d041-4128-a78a-43e171e8df76
kernel       /boot/memtest86+.bin
quiet
...
```

```

title      Windows NT/2000/XP (loader)
root      (hd0,0)
savedefault
makeactive
chainloader +1
$
```

下面以 menu.lst 文件中的第一个菜单项，即“title Ubuntu 8.10, kernel 2.6.27-9-generic”菜单项为例，说明每个命令或参数的意义。

配置参数 title 用于指定引导菜单项的名字，即操作系统的名字，如“Ubuntu 8.10, kernel 2.6.27-9-generic”；Ubuntu 8.04 使用“root (hd0,6)”指定“/”或/boot 文件系统分区，Ubuntu 8.10 使用 UUID 指定“/”或/boot 文件系统分区。通过查询/etc/fstab 文件可知，两者的意义其实是一样的，均指第 1 个磁盘的第 7 个分区，只不过前者更符合 GRUB 规范。kernel 用于指定需要引导的系统内核文件名，如/boot/vmlinuz-2.6.27-9-generic。文件名之后是需要由 GRUB 引导程序传递给系统内核的引导参数，其中，“root=UUID=...”表示把指定的磁盘分区安装作“/”文件系统。initrd 用于指定初始的磁盘映像文件名，如/boot/initrd.img-2.6.27-9-generic。这是一个经由 gzip 命令压缩的文件系统映像，当 GRUB 引导程序把系统内核与磁盘映像文件加载到内存，系统内核开始运行时，首先解压并把磁盘映像文件的内容复制到/dev/ram0 设备中，然后释放磁盘映像文件占用的内存。之后，系统内核以读写方式安装/dev/ram0 设备，作为 Linux 系统引导初始阶段使用的临时“/”文件系统。

如果配置参数 hiddenmenu 之前未加注释符“#”，GRUB 将会采用默认的操作系统引导系统，而不会显示操作系统引导选择菜单，除非在设定的超时值之内按下 Esc 或空格键。增加注释符之后，GRUB 将会显示由配置参数 title 表示的操作系统引导选择菜单。

#### 14.1.4 安装或修复 GRUB

在使用 CD/DVD 安装介质安装 Ubuntu Linux 系统时，GRUB 将会随之一同被安装到系统盘的 MBR、第 1~62 号数据块以及 Linux 系统的/boot 文件系统中。如果准备在一台计算机中同时安装 Windows 与 Linux 两个系统，通常需要首先安装 Windows，然后再安装 Linux 系统，否则无法启动 Linux 系统。如果因故缺失，或者在安装 Linux 系统之后又重新安装了 Windows 系统，致使部分 GRUB 映像被覆盖，无法正常启动 Linux 系统，可以使用 CD/DVD 安装介质启动系统，单独安装或修复 GRUB。

重新安装或修复 GRUB 时，可以使用 grub-install 命令。grub-install 命令主要用于安装 stage1（即 MBR）、e2fs\_stage1\_5 及 stage2 等 GRUB 映像文件，以便能够引导系统。其语法格式简写如下。

**grub-install [ --root-directory=dir ] /dev/sdD**

或

**grub-install [ --root-directory=dir ] hdN**

其中的“D”可以是 a、b……，N 可以是 0、1……，分别表示系统配备的第 1 个磁盘、第 2 个磁盘等。例如，要把/boot/grub 目录中的 stage1 等 GRUB 映像文件安装到系统磁盘/dev/sda 的适当存储位置，可以输入下列命令。

```

$ sudo grub-install /dev/sda
Installation finished. No error reported.
$
```

由于 GRUB 只是一个引导程序，当系统内核存储在一个单独的/boot 分区，即/boot 文件系统时，GRUB 并不知道/boot 文件系统的安装点，因而需要使用“--root-directory”选项告知 GRUB，如下所示。

```
# grub-install --root-directory=/boot /dev/sda
```

一旦安装，GRUB 将会自动成为系统默认的引导程序。当再次启动计算机系统时，GRUB 的



操作系统引导菜单选项就会出现在屏幕上。

如果上述命令出错，或者不知道如何确定磁盘设备，可以查阅 /boot/grub/device.map 文件，如下所示。

```
$ cat /boot/grub/device.map
(hd0)  /dev/sda
$
```

因此，不管出于何种原因，一旦需要重新安装或修复 GRUB 时，可以参照下列步骤，在不安装 Linux 系统的情况下，生成一个基于内存和硬盘原有系统的 Ubuntu Linux 运行环境，以便能够利用 Linux 命令和工具修改系统配置文件，修复一个已安装到计算机硬盘中的 Linux 系统。

- (1) 使用 Ubuntu Linux 系统的 CD/DVD 安装介质重新引导系统；
- (2) 选择安装语言环境“中文（简体）”；
- (3) 在安装界面选择“试用 Ubuntu 而不改变计算机中的任何内容（T）”；
- (4) 进入 Linux 系统之后，在 GNOME 界面中选择“Applications→附件→终端”菜单项，打开一个终端窗口；
- (5) 如果安装的 Ubuntu Linux 系统没有单独的 /boot 文件系统分区，使用下列命令，把 MBR 等 GRUB 映像文件安装到指定的设备中（其中 X 表示 root 文件系统的磁盘分区设备名）。

```
ubuntu@ubuntu:~$ sudo mount /dev/sdax /mnt
ubuntu@ubuntu:~$ sudo grub-install --root-directory=/mnt /dev/sda
Installation finished. No error reported.
```

- (6) 如果安装的 Ubuntu Linux 系统存在单独的 /boot 文件系统分区，使用下列命令，把 MBR 等 GRUB 映像文件安装到指定的设备中（其中，sda 表示系统配备的第一个磁盘，X 表示“/”文件系统的磁盘分区设备名，Y 表示 /boot 文件系统的磁盘分区设备名）。

```
ubuntu@ubuntu:~$ sudo mount /dev/sdax /mnt
ubuntu@ubuntu:~$ sudo mount /dev/sday /mnt/boot
ubuntu@ubuntu:~$ sudo grub-install --root-directory=/mnt /dev/sda
Installation finished. No error reported.
```

- (7) 使用下列 umount 命令卸载 /mnt 等文件系统。
- (8) 关闭终端窗口，选择“System→Shut Down”菜单，然后选择“Restart”，重新启动系统，按照系统的提示，取出 CD/DVD 安装介质之后按下 Enter 键。

- (9) 从设定的磁盘中重新引导系统。

此外，还可以使用 GRUB 提供的 root 和 setup 等命令，重新安装或修复 GRUB。具体处理过程如下。

- (1) 按照前述的操作步骤（1）至步骤（4）启动系统，打开一个终端窗口。

- (2) 输入 grub 命令，进入 GRUB 命令行界面。

```
ubuntu@ubuntu:~$ grub
grub>
```

- (3) 根据安装 Linux 系统的磁盘分区，输入下列命令。

```
grub> root (hdx,y)
grub> setup (hdx)
```

在上述命令中，“x”表示磁盘的编号，“y”表示磁盘分区编号。如果系统中只配有一个磁盘，则“x”为 0。如果 Linux 系统安装在第二个磁盘上，则“x”为 1，“y”是安装了 Linux “/”文件系统的磁盘分区。例如，如果 Linux 系统安装在第 3 个磁盘分区中，且系统中只配有一个磁盘，可以使用下列命令。

```
grub> root (hd0,2)
grub> setup (hd0)
```

上述的“setup (hd0)”命令表示把 GRUB 复制到系统硬盘第 0 号扇区的 MBR 中。当出现“successful.....”等字样时，表示已成功地重装了 GRUB 引导程序，此时可以使用 quit 命令退出 GRUB，重新启动系统即可。

表 14-2 给出了部分常用的 GRUB 命令及其简单说明。

表 14-2

常用的 GRUB 命令

GRUB 命令	简单说明
boot	用于引导操作系统或级联引导程序。这个命令仅适用于命令行界面的交互运行方式
chainloader file	使用指定的文件作为下一个级联引导程序。如果加载的文件位于指定磁盘分区的第一个扇区，可以采用“chainloader +1”的命令形式指定加载的数据块号，使 GRUB 读取磁盘分区的第一个扇区，而不必指定文件名
initrd file	用于指定引导时需要加载的初始磁盘内存映像文件，如/boot/initrd.img-2.6.27-9-generic 文件。当系统内核需要加载一定的附加模块才能适当地引导时，使用 initrd 命令是很有必要的。例如，当系统分区为 Ext3 文件系统时，则需要使用下列 initrd 命令加载初始内存磁盘映像文件，并适当地设置内存启动区域中的内核参数： initrd/boot/initrd.img-2.6.27-9-generic
install stage1_file device stage2_file [p config_file]	这个命令相当复杂，除非特别熟悉 GRUB，通常不建议使用这个命令。类似的需求可以采用 setup 命令实现。简言之，这个命令的功能是把 GRUB 的初始引导程序（stage1）安装到指定磁盘设备的 MBR 中。其中，stage1_file 用于指定一个磁盘、磁盘分区或文件，其中包含初始引导程序映像，如(hd0,0)/grub/stage1。device 用于指定一个磁盘设备，如(hd0)，表示把初始引导程序 stage1 安装到其 MBR 中。stage2_file 表示把 stage2 引导程序（如(hd0,0)/grub/stage2）的存储位置传递给 stage1 初始引导程序。“p config_file”选项的目的是让 install 命令检索由 config_file 指定的，包含操作系统引导菜单选项的配置文件，如(hd0,0)/grub/menu.lst
kernel file [options]	指定引导操作系统时需要加载的系统内核文件，以尝试从指定的文件中加载主引导映像（如 /boot/vmlinuz-2.6.27-9-generic 文件）。选项部分用作传递给系统内核的命令行参数，如“root=UUID=4c5ea7fa-d041-4128-a78a-43e171e8df76 ro locale=zh_CN quiet splash”，其中的 UUID 是/dev/sda7 的另外一种表示形式。在指定系统内核文件时，应使用 root 命令设置的磁盘分区的绝对路径名替换 file 参数，使用 Linux 系统内核能够接受的选项替换 options，以指定要加载系统分区的设备名。如果提供多个选项，选项之间需加空格字符。例如，上述设置说明 Linux 的“/”文件系统位于第一个磁盘的第 7 个分区（参见/etc/fstab 文件）
module file [options]	用于加载系统引导期间需要的附加模块。选用的参数将会传递给模块，用作模块的命令行参数。注意，在加载任何模块之前必须先加载系统内核映像
reboot	重新引导计算机系统
root device	指定系统内核或操作系统所在的磁盘与磁盘分区，如(hd0,2)。然后，尝试安装磁盘分区含有的文件系统
rootnoverify device	指定 GRUB 所在的磁盘与磁盘分区。当操作系统超出了 GRUB 能够直接读取的磁盘存储区域时，这个命令是非常有用的
setup [--stage2=stage2_file] [--prefix=dir] device [image]	利用 install 命令，采用灵活的安装方式，把 GRUB 映像文件自动安装到指定的设备中。如果指定的 GRUB 映像文件 image 是一个设备，则使用设备中存储的 GRUB 映像文件。否则，从 root 命令指定的设备中找出并使用其中存储的 GRUB 映像文件。如果准备安装 GRUB 映像文件的目的设备是一个磁盘，把适当的 stage 1.5 引导程序也写入磁盘。“--prefix”选项用于指定 GRUB 映像文件所在的目录，如果未指定目录，GRUB 将会自动地在/boot/grub 和/grub 目录中检索 GRUB 映像文件
help	显示 GRUB 的帮助信息
quit	退出 GRUB Shell（这个命令仅适用于 GRUB 交互式 Shell）

## 14.2 初始引导过程

在计算机加电或重新启动时，系统将会自动跳到 ROM BIOS 中的加电引导程序入口点开始运行。一个加电引导程序究竟都具有什么功能，因计算机系统的不同而有所不同，但其基本功能和



目的却是一样的：首先检查系统的硬件配置，运行硬件诊断程序，执行加电自检；然后找出系统磁盘，并把系统盘第 0 号扇区中的初始引导程序读入内存，执行初始的系统引导。

初始引导程序的主要功能是确定活动的分区（当系统中装有多个操作系统时），执行第二阶段（或第 1.5 阶段）的引导程序，或者找出活动分区中由操作系统提供的系统引导程序，加载并运行系统引导程序。

Linux 系统的一个重要功能特性是采用一种开放的、用户能够控制的方法引导或启动操作系统。用户可以根据自己的需要，配置操作系统的引导过程——指定不同的引导程序、引导参数及其他设置。

### 14.2.1 GRUB 引导过程概述

GRUB 是一个较大的程序，由于 MBR 容量的限制，通常把 GRUB 分拆成 2（或 3）个不同的程序，即 stage1 和 stage2（或再加一个可选的 stage1.5）。因此，GRUB 通常提供若干引导程序映像：两个基本的引导程序（stage1 和 stage2）、一系列选用的引导程序（如 e2fs\_stage1\_5）、一个 CD-ROM 引导程序，以及两个网络引导程序（如 nbgrub 和 pxegrub）等。

stage1 是最基本的初始引导程序，可用于引导 GRUB，从而引导目标操作系统。通常，随着操作系统的安装，这个初始引导程序将会被复制到 MBR 引导扇区中。由于计算机的 MBR 引导扇区只有 512 个字节的存储空间，这个程序映像也恰好是 512 个字节。任何 stage1 引导程序的主要功能都是从本地系统磁盘中加载 stage2 或可选的 stage1.5 引导程序。由于 MBR 存储空间的限制，stage1 通常无法识别和理解任何文件系统，因而只采用数据块偏移值的形式，把 stage2（或 stage1.5）的起始磁盘位置编写到自己的跳转代码中。

stage2 是 GRUB 的核心引导程序映像，除了引导自身之外，stage2 能够完成任何系统引导任务。通常，stage2 引导程序存储在 /boot 文件系统中。但需注意的是，stage2 并非必需的引导程序。

stage1.5 引导程序是第一与第二阶段引导程序之间的一个桥梁，也是一组引导程序集合的总称。针对不同的 Linux 系统及其文件系统，GRUB 提供若干 stage1.5 中间阶段（也可称第 1.5 阶段）的引导程序，如 e2fs\_stage1\_5、fat\_stage1\_5、jfs\_stage1\_5、reiserfs\_stage1\_5 及 xfs\_stage1\_5 等。整个引导序列是：第一阶段引导程序加载并运行第 1.5 阶段引导程序，第 1.5 阶段引导程序又接力加载并运行第二阶段引导程序。第一阶段引导程序与第 1.5 阶段引导程序之间的差别是，前者不能识别任何文件系统，而后者能够支持一种特定的文件系统（如 e2fs\_stage1\_5 支持 Ext2 文件系统）。因此，可以安全地把 stage2 引导程序移至任何存储位置，即使已经安装了 GRUB。

stage1 初始引导程序通常存储在 MBR 或某个磁盘分区的引导扇区，把 stage2 引导程序放置到 /boot 目录或文件系统中，选用的 stage1.5 引导程序可以存储在紧随 MBR 之后的 62 个连续扇区中。这是因为 stage1.5 引导程序较小，而 MBR 之后的 62 个扇区通常又是空闲的。

下面以 Intel x86 系列计算机上安装的 Ubuntu 8.04/8.10 Linux 系统为例，概述 Linux 系统的基本引导步骤，如图 14-5 所示。

(1) 加电自检。加电后，计算机中的 BIOS 开始检测系统的硬件配置，自动执行硬件诊断程序，然后找出能够引导系统的存储设备，如系统硬盘、软盘、CD/DVD、网络设备或 USB 移动硬盘等。根据第 0 号扇区最后两个字节是否为 0xAA55，确定其是否为 MBR。一旦找出引导设备，BIOS 将会立即加载第 0 号扇区 MBR 中的初始引导程序（即第一阶段引导程序），然后把系统引导的控制权交给初始引导程序。

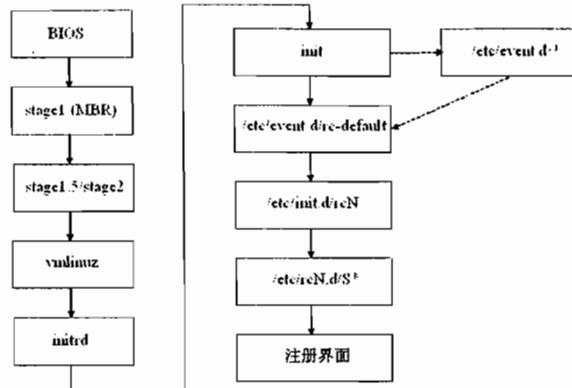


图 14-5 Ubuntu Linux 系统引导过程示意

(2) 执行第一阶段引导程序 (stage1)，进行初始引导。stage1 引导程序的主要目的是找出并加载系统引导程序，即第 1.5 阶段 (e2fs\_stage1\_5) 或第二阶段引导程序 (stage2)。加载并启动磁盘设备第 1~62 号扇区中的引导程序 e2fs\_stage1\_5，或者查询 MBR 中的磁盘分区表，找出活动的磁盘分区，加载并运行位于 “/” 或 /boot 分区中的第二阶段引导程序 stage2。通常，第 1.5 阶段引导程序能够识别 Ext2/Ext3 类型的文件系统，因而能够访问磁盘分区中的 “/” 或 /boot 文件系统。

(3) 执行第 1.5 或第二阶段引导程序 (其最终目的是加载系统内核与初始的磁盘内存映像文件)，读取 /boot/grub/menu.lst 文件，根据其中的设置，在控制台屏幕上显示一个操作系统引导菜单。

(4) 根据用户的选择 (用户可以使用箭头键，选择一个想要引导的操作系统，然后按下 Enter 键。在一定的时间内，如果用户一直置之不理，即没有做任何选择，则采用事先设定的默认操作系统)，加载系统内核文件 /boot/vmlinuz-version-generic 与磁盘内存映像文件 /boot/initrd.img-version-generic。最后把控制权交给系统内核。

(5) 系统内核开始运行时，首先解压并把磁盘内存映像文件复制到 /dev/ram0 设备中，然后释放磁盘内存映像占用的内存空间。

(6) 根据引导参数，以只读或读写方式安装 /dev/ram0 设备，把初始磁盘内存映像文件安装成一个临时的 “/” 文件系统。

(7) 系统内核开始检查并设置 Linux 系统的硬件运行环境，配置计算机系统的 CPU、内存、磁盘设备、I/O 设备及其他硬件设备等，设置系统时钟，初始化软件模块，组织和初始化各种数据结构，构造运行环境。然后再根据需要，加载其他必要的驱动程序模块。

(8) 如果 “/” 文件系统中存在可执行的 /linuxrc 文件，以超级用户的身份执行 /linuxrc。/linuxrc 可以是任何可执行文件，包括 Shell 脚本。

(9) 卸载临时 “/” 文件系统，释放其占用的内存空间，安装实际的 “/” 文件系统。

(10) 访问 “/” 文件系统，运行 /sbin/init 程序，把引导过程的控制权转交给 init 进程，由 init 进程完成系统的生成过程。init 是系统引导过程中执行的第一个 C 程序，之前运行的引导程序等均非 C 程序 (注意，在嵌入式 Linux 系统中，init 进程并非必须。实际上，可以调用一个简单的 Shell 脚本，启动嵌入式应用)。

(11) init 开始检索 /etc/event.d 目录，找出并运行 rc-default 文件，确定默认的运行级 (Ubuntu Linux 系统的默认运行级为 2)。



- (12) 触发 startup 事件，根据/etc/event.d/rcS，调度运行/etc/init.d/rcS 脚本。而 rcS 又调度运行“/etc/init.d/rc S”，按照优先顺序，依次启动/etc/rcS.d 目录中的 Shell 脚本，设置系统的主机名、硬件时钟、内核参数、网络，以及检测与安装文件系统等。
- (13) 触发 rcS 运行终止事件，根据/etc/event.d/rc-default，调度运行 telinit 程序。
- (14) 触发“runlevel 2”事件运行，根据/etc/event.d/rc2，调度运行/etc/event.d/rc2 及其他作业（如/etc/event.d/logd 等）。
- (15) /etc/event.d/rc2 以运行级 2 为参数，调度运行/etc/init.d/rc 脚本。
- (16) /etc/init.d/rc 脚本再按照优先顺序，依次启动/etc/rc2.d 目录中所有的 Shell 脚本（包括用户定义的服务程序），启动系统日志守护进程，启动 DNS、sshd、mysql、NFS、samba 以及 apache2 等服务器进程，启动 cron、atd 及 anacron 等守护进程，启动 X 服务器以及 GNOME 显示管理器 GDM 等，直至在控制台上显示一个注册界面。至此，说明 Linux 系统已经就绪，用户可以注册、访问 Linux 系统了。

### 14.2.2 补充说明

磁盘内存映像文件 initrd 的主要功能是提供一个临时的“/”文件系统，系统内核可以利用这个虚拟的“/”文件系统，加载必要的驱动程序模块，使系统内核能够在不安装任何物理磁盘的情况下完成系统的引导与初始化。如果系统中配有 SCSI 硬盘，这一步骤是非常重要的。由于 initrd 能够提供附加的外设驱动模块，故系统内核可以做得非常小，但仍然能够支持许多硬件配置。

利用内存映像文件 initrd，可以创建一个小型的 Linux 系统内核，把驱动程序编译成可以动态加载的模块。利用动态加载的驱动程序模块，系统内核能够访问各种硬件设备，包括磁盘及其中的文件系统。由于“/”或/boot 文件系统通常位于磁盘设备中，initrd 提供的磁盘访问功能确保最终能够安装实际的“/”文件系统。在一个没有硬盘设备的嵌入式 Linux 系统中，initrd 仍然可以作为“/”文件系统继续使用。

## 14.3 系统生成过程

系统启动过程的最后一个阶段是系统生成。在 init 进程开始运行之前，Linux 系统已基本成形，但还有其他必要的系统进程尚未启动，无法向用户提供系统注册与访问服务，因此还远未到完全可用的正常工作状态，还有大量的系统生成任务需要 init 进程完成。

在系统生成阶段中，init 进程需要根据/etc/event.d 目录中的配置文件，调度运行各种守护进程，提供附加的系统与网络功能，设置用户工作环境，最终完成系统启动过程中的后续处理任务。

init 守护进程是系统引导起来之后创建的第一个真正意义上的进程，是其他所有系统进程的父进程，除了少数的几个内核进程，系统中几乎所有的进程都是 init 守护进程的直接或间接子进程。一旦开始运行，init 进程将会立即检查是否存在/etc/inittab 文件，检查/etc/event.d 目录。

系统引导过程中的 startup 事件将会触发 init 进程执行/etc/event.d/rcS 任务，rcS 任务在设置必要的变量之后，开始调度运行/etc/init.d/rcS 脚本。接着，rcS 脚本再以 S 作为参数，执行/etc/init.d/rc 脚本，进而调度运行/etc/rcS.d 目录中的 Shell 脚本。

/etc/rcS.d 目录中的 Shell 脚本用于执行必要的系统初始化任务，如设置系统的主机名、系统

时钟、系统内核可调参数，启动网络功能，以及检测与安装/etc/fstab 文件中定义的文件系统等，以确保在用户注册之前，能够生成一个可用的基本系统。

在rcS任务运行结束之后，系统将会触发rcS终止事件，这将使init进程开始执行/etc/event.d/rc-default任务。如果存在，init将会基于/etc/inittab文件中的内容，按照传统的方式启动系统，即根据/etc/inittab文件的设置调度执行其中指定的命令。/etc/inittab文件描述了在进入特定的运行级时如何定制系统。运行级是利用/etc/rcN.d目录中的Shell启动脚本定制的一种运行状态或工作模式（其中的“N”是0~6中的一个数字或S，表示系统的运行级）。/etc/event.d/rc-default文件的内容如下。

```
$ cat /etc/event.d/rc-default
# rc - runlevel compatibility
#
# This task guesses what the "default runlevel" should be and starts the
# appropriate script.
start on stopped rcS

script
    runlevel --reboot || true

    if grep -q -w -- "-s\\!single\\!S" /proc/cmdline; then
        telinit S
    elif [ -r /etc/inittab ]; then
        RL="$($ sed -n -e "/^id:[0-9]*:initdefault:/ { s/^id://; s/:.*//; p }" /etc/inittab
|| true)"
        if [ -n "$RL" ]; then
            telinit $RL
        else
            telinit 2
        fi
    else
        telinit 2
    fi
end script
$
```

如果/etc/inittab文件不存在，rc-default将会根据引导系统内核时传递的参数，以下列方式之一启动系统。

- 使用S作为参数，运行telinit命令，从而触发“runlevel S”事件，导致init进程开始执行/etc/event.d/rcS-slogin任务，使系统进入单用户恢复模式；
- 使用2作为参数，运行telinit命令，从而触发“runlevel 2”事件，导致init进程开始执行/etc/event.d/rc2任务，使系统进入多用户模式。

按照rc-default任务确定的默认运行级，init将会继续执行尚未完成的后续系统生成任务。在Ubuntu Linux系统中，默认的运行级为2，因此，在“runlevel 2”事件出现之后，init进程将会执行/etc/event.d/rc2任务，检测/etc/rc2.d目录，确定需要启动哪些系统守护进程，进而采用“/etc/init.d/rc2”的命令形式，启动/etc/rc2.d目录中以S加两位数字序号为起始文件名的Shell脚本。

当前，Ubuntu Linux系统提供的守护进程启动脚本均位于/etc/init.d目录中。根据需要以及启动与终止的顺序，Linux系统采用符号链接的方式，把一组选定的Shell启动脚本分别链接到相应的/etc/rcN.d目录中，使之能够在进入不同的运行级时，执行一组特定的Shell脚本，但系统只需维护一套完整的Shell脚本。当需要在不同的运行级中运行同一个守护进程时，把守护进程的相应Shell脚本分别链接到多个/etc/rcN.d目录中即可。

在/etc/rcN.d目录中，每个符号链接文件的名字前都有一个S或K字符，S表示在进入指定的运行级时应启动相应的守护进程，K表示终止相应的守护进程。除了K或S之外，每个文件名还



包含两个数字编号，表示启动或终止的顺序。通过改变数字编号，可以改变启动或终止守护进程的顺序。数字编号越小，启动或终止时执行得越早。如果数字编号相同，则按整个文件名的字符排序原则确定先后次序。

在 Ubuntu Linux 系统中，绝大部分守护进程当前仍然是通过 /etc/event.d 目录中的 rcN，由位于 /etc/init.d 目录中的 rc 或 rcS 脚本启动和关闭的。采用 “/etc/rcN.d/script stop” 形式的命令，可以终止以 K 为起始文件名的所有守护进程；采用 “/etc/rcN.d/script start” 形式的命令，可以启动以 S 为起始文件名的所有守护进程。其中，script 是守护进程的相应脚本文件名。

### 14.3.1 基本概念

与传统的基于运行级与 /etc/inittab 文件的进程调度方式不同，从 6.10 版本开始，Ubuntu Linux 系统逐步采用 Upstart 替代传统的 init 进程，以基于事件触发机制的进程调度方式，负责守护进程的管理与维护。但为了保持兼容性，Upstart 仍保留了传统 init 的部分特性，且将其仍然称作 init 进程。

新的 init 守护进程实际上是一个状态机，用于跟踪系统中定义的所有作业、作业的状态以及触发作业状态改变的事件，实现进程的调度。当作业从一个状态转变到另外一个状态时，init 进程将会执行与作业相关的命令，或者终止作业的执行。例如，等待文件系统安装的作业会由于安装事件的出现而启动，由于卸载事件的出现而终止。

#### 1. 事件

事件是一种状态或状态变化信息，这种状态或状态变化信息可以来自系统内部，也可以来自系统外部；可以是系统预定义的，也可以是应用程序自定义的。例如，引导程序触发的 startup 事件，系统进入运行级 2 时触发的“runlevel 2”事件，以及安装移动介质触发的文件系统 mount 事件等。利用“initctl emit”命令，也可以手工触发一个事件。目前，新的 init 进程能够识别的事件可以分为下列 3 种类型。

- 边界事件（Edge event）——一个描述系统状态分界点的字符串，如 startup，一旦定义的事件出现，系统将会调度运行或终止等待事件发生的作业。例如，“start on startup” 表示在启动系统时执行相应的作业，“stop on shutdown” 表示在关闭系统时终止相应的作业。
- 运行级事件（Level event）——类似于边界事件，其差别是除了描述性的字符串之外，还有一个相关的值，如 up 或 down 等。字符串值的变动将会触发一个运行级事件和一个同名的边界事件。例如，“start on runlevel 2” 表示当系统进入运行级 2 时执行相应的作业。
- 时间点事件（Temporal event）——用于执行诸如“15 分钟之后启动”、“每日”或“两点半”执行之类的动作。

#### 2. 作业

作业由一系列指令组成，这些指令通常包含目标代码程序（或 Shell 脚本）与事件的名字。在事件触发之后，init 进程将会运行相应的程序。利用“initctl start”和“initctl stop”命令，也可以手工运行或终止一个程序。作业可以进一步细分为任务与服务。

/etc/event.d 目录中的文件都定义了一个作业，每个作业通常都至少包含一个事件和命令，当定义的事件发生时，init 进程就会执行相应的命令。

#### 3. 任务

任务通常由二进制目标代码或 Shell 脚本组成，用于执行其内定的处理动作，完成后返回等待状态。实际上，任务是对 /etc/rcS.d 目录中现有的大多数 Shell 脚本与 cron 脚本的模拟实现。

## 4. 服务

第二种作业是服务，服务是一种连续运行的作业，自身无法也无需终止。例如，logd 等守护进程都是以服务的方式实现的。init 进程负责监控每一个服务，如果某个服务运行失败，init 进程负责重启相应的服务；如果经由系统管理员手工要求，或者某个事件要求停止运行，init 进程负责终止相应的进程。

## 5. 作业定义文件

/etc/event.d 目录存有一系列作业定义文件，其中规定了 init 进程应当执行的作业。当前，Ubuntu Linux 系统及安装的软件包均开始在这个目录中创建自己的作业定义文件，以控制服务的运行状态，而不是完全在/etc/init.d 和/etc/rcN.d 目录中增加启动脚本文件。

目前，Ubuntu Linux 系统主要采用 exec 与 script 两种命令形式定义作业。其中，exec 主要用于启动随后给出的单个命令，其语法格式如下。

```
exec command arguments
```

例如，下面是取自/etc/event.d/control-alt-delete 配置文件中的一个片段，表示当用户在控制台上按下 Ctrl-Alt-Del 键时执行 shutdown 命令。

```
$ cat /etc/event.d/control-alt-delete
.....
exec /sbin/shutdown -r now "Control-Alt-Delete pressed"
$
```

script 用于执行一组命令，相当于执行一个内置的 Shell 脚本文件，其语法格式如下。

```
script
    command-List
end script
```

例如，下面是取自/etc/event.d/rc6 配置文件中的一个片段。

```
$ cat /etc/event.d/rc6
.....
script
    set $(runlevel || true)
    if [ "$2" != "0" ] && [ "$2" != "6" ]; then
        set $(runlevel --set 6 || true)
    fi

    if [ "$1" != "unknown" ]; then
        PREVLEVEL=$1
        RUNLEVEL=$2
        export PREVLEVEL RUNLEVEL
    fi

    exec /etc/init.d/rc 6
end script
$
```

从上述例子中可以看出，exec 与 script 的功能基本上是一样的，只不过前者执行的是一个命令，而后者执行的是一个内置的 Shell 脚本。

## 6. 运行级仿真

传统的 init 进程利用运行级及运行级的变化确定何时以及怎样启动或终止进程，而 Ubuntu Linux 系统中采用的 init 进程不采用运行级的概念。为了便于从基于运行级的系统迁移到基于事件的系统，同时兼顾与其他软件的兼容性，Ubuntu Linux 系统采用了运行级仿真的运行方式。

运行级实际上是一种运行模式或运行状态，故运行级有时也称作运行状态（Run State）。在不同的运行级中，系统能够提供不同的功能或服务。从系统管理的角度来看，运行级实际上是一种软件配置方式，在任何一个运行级中，只有选定的一组守护进程才能够运行。

采用运行级定义系统的运行模式时，针对每一个运行级，早先（也包括现在）的 Linux 系统



中都定义了一组选定的守护进程，以便构建一种特定的运行环境，从而使系统能够满足不同的工作需求。Linux 系统可以处于不同的运行状态，但在任何时刻，Linux 系统只能处于 8 种可能的运行级之一，如常见的多用户工作模式或用于维护目的的单用户工作模式等。

定义运行级的目的是组织和划分系统进程的适用场合，根据不同的工作模式和功能需求确定不同的进程配置。因此，每个运行级具有不同的作用，以适应不同的功能需求，守护进程的主要作用是激活硬件设备，提供网络服务，以及维护系统日志等。每个系统进程既可属于一个特定的运行级，也可用于多个运行级。这就需要使用运行级划分系统的工作模式，相应地配置一组选定的系统进程。每个进程只能在其定义的运行级中才能运行。

在 Linux 系统中，init 进程可以识别下列 7 个主要运行级（0~6）及一个内部定义的单用户运行级——即 S 运行级。

- 0 —— 关机。用于停止 Linux 操作系统。
- 1 —— 单用户工作模式。
- 2 —— 多用户运行模式。这是 Ubuntu Linux 系统默认的运行级，在系统初始化过程完成之后，Linux 系统将会自动进入这个运行级。
- 3、4、5 —— 暂未定义，可供用户定义其他多用户工作模式。
- 6 —— 重新启动系统。这个运行级包括两个工作过程：首先停止 Linux 系统，其处理的步骤和内容与运行级 0 完全一样；然后重新引导 Linux 系统，其效果如同加电启动系统。在进行系统配置、调整系统参数及安装系统软件包或更新 Linux 系统时，通常需要用到这一运行级。
- S —— 系统内部定义的单用户恢复模式。在系统引导过程中，如果选用任何恢复模式，系统将会自动进入这一运行级。系统管理员可以通过控制台访问系统，执行系统维护任务。

要改变系统的运行级，超级用户可以运行 init 或 telinit 命令。如需了解当前的运行级，可以使用下列命令。

```
$ who -r
      run-level 2  2008-12-10 15:48           last=
$ 或
$ runlevel
N 2
$
```

在上述第一个命令的输出信息中，第 1 个字段“run-level 2”表示当前的运行级为 2，第 3 个字段“last=”表示先前的运行级为空，即不知道从哪一个运行级转入当前的运行级。

/etc/event.d/rcN 文件中定义的 rcN 作业运行/etc/init.d/rc 脚本。通过/etc/rcN.d 目录中的符号链接文件，这个脚本又调度运行/etc/init.d 目录中的 Shell 启动脚本，仿真传统 init 进程的传统做法。当系统进入某个运行级时，Ubuntu Linux 系统利用运行级作为参数，使用 rcN 作业启动传统的 Shell 脚本；当系统离开当前的运行级时，则不采取任何动作。Ubuntu Linux 系统也实现了 runlevel 与 telinit 等实用程序，提供与 System V 系统的兼容性。

## 7. initctl 命令

利用 initctl 命令，拥有超级用户访问权限的系统管理员能够与新的 init 进程进行通信，启动、终止和查询作业，触发指定的事件，等等。例如，“initctl list”命令能够列出当前的作业及其状态，如下所示。

```
$ sudo initctl list
[sudo] password for gqxing:
```

```

control-alt-delete (stop) waiting
last-good-boot (stop) waiting
logd (stop) waiting
rc-default (stop) waiting
rc0 (stop) waiting
rc1 (stop) waiting
rc2 (stop) waiting
.....
rc6 (stop) waiting
rcS (stop) waiting
rcS-sulogin (stop) waiting
sulogin (stop) waiting
tty1 (start) running, process 5697
tty2 (start) running, process 4217
.....
tty6 (start) running, process 4221
$
```

利用“`initctl help`”命令，能够查询`initctl`命令支持的各种子命令，而使用下列命令则能够给出子命令的更多说明信息。

```

$ initctl list --help
Usage: initctl list [OPTION]...
List known jobs.

Options:
  --show-ids          show job ids, as well as names
  -p, --pid=PID       destination process
  -q, --quiet          reduce output to errors only
  -v, --verbose         increase output to include informational messages
  --help                display this help and exit
  --version             output version information and exit

Report bugs to <upstart-devel@lists.ubuntu.com>
$
```

使用`start`、`stop`或`status`替换`initctl`命令中的`list`子命令，可以获取相应子命令的更多说明信息。

### 14.3.2 init 进程与/etc/event.d 目录

理论上，可以在`/etc/event.d`目录中定义系统生成过程中需要自动调度运行的任何作业，但在实际的应用过程中，`/etc/event.d`目录中定义的作业大体上可以分为3类：在系统进入指定的运行级之前尚需执行的部分初始化任务（如`/etc/event.d/rcS`执行的任务），在进入指定的运行级之后需要启动的各种守护进程——即执行`/etc/rcN.d`目录中的Shell启动脚本，其中N为实际的运行级，以及启动必要的终端守护进程或应用程序提供的守护进程。

#### 1. 系统初始化

在`/etc/event.d`目录中，只有下列`rcS`配置文件在监听`startup`事件。这意味着，当`startup`事件出现时，将会执行`rcS`配置文件中定义的任务，最终执行`/etc/init.d/rcS`脚本。

```

$ cat /etc/event.d/rcS
# rcS - runlevel compatibility
#
# This task runs the old sysv-rc startup scripts.

start on startup

stop on runlevel

# Note: there can be no previous runlevel here, if we have one it's bad.
# information (we enter rc1 not rcS for maintenance).
console output
script
```



```
runlevel --set S >/dev/null || true  
  
PREVLEVEL=N  
RUNLEVEL=S  
export PREVLEVEL RUNLEVEL  
  
exec /etc/init.d/rcS  
end script  
$
```

接着，/etc/init.d/rcS 脚本将会以 S 作为参数，调度运行“/etc/init.d/rc”脚本，依次执行/etc/rcS.d 目录中以 Sxx 为起始文件名的脚本文件，设置系统的主机名、系统时钟、系统内核可调参数，以及检测与安装文件系统，等等。

如果仔细考察一下/etc/rcS.d 目录中的文件，可以看到，在进入指定的运行级之前，系统将会执行下列最后部分的系统初始化任务。

- 通过预读的方式，把系统启动过程中需要运行的程序事先读入内存，以便提高系统启动的速度，改善系统启动的性能，加快系统的启动过程。注意，仅当系统的可用内存大于 384MB 时才有效（S01readahead）。
- 根据/etc/fstab 文件的要求，检测、修复与安装文件系统；检测限额设置的一致性，启用限额机制（S01mountkernfs.sh、S11mountdevsubfs.sh、S20checkroot.sh、S30checkfs.sh、S35mountall、S45mountnfs.sh）。
- 设置系统的主机名（S02hostname.sh）。
- 设置系统时钟（S08hwclockfirst.sh、S11hwclock.sh）。
- 设置系统内核可调参数（S17procps）。
- 清除并重建文件系统安装表/etc/mtab（S22mtab.sh）。
- 配置与激活网络接口，设置 IP 地址及默认的路由，启动与网络有关的守护进程（S40networking）。
- 清理/tmp 目录，删除其中的临时文件（S46mountnfs-bootclean.sh——/etc/init.d/bootclean）。
- 设置系统控制台（S49console-setup、S90console-screen.sh）等。

## 2. 定制运行环境

当/etc/event.d/rc-default 最终确定了系统的运行级时，例如，当系统进入默认的运行级 2 即进入多用户工作模式时，init 进程将会运行下列/etc/event.d/rc2 脚本，然后以运行级 2 作为参数，调度运行/etc/init.d/rc 脚本。

```
$ cat /etc/event.d/rc2  
# rc2 - runlevel 2 compatibility  
#  
# This task runs the old sysv-rc runlevel 2 ("multi-user") scripts. It  
# is usually started by the telinit compatibility wrapper.  
  
start on runlevel 2  
  
stop on runlevel [!2]  
  
console output  
script  
    set $(runlevel --set 2 || true)  
    if [ "$1" != "unknown" ]; then  
        PREVLEVEL=$1  
        RUNLEVEL=$2  
        export PREVLEVEL RUNLEVEL
```

```

    fi

    exec /etc/init.d/rc 2
end script
$
```

/etc/init.d/rc 脚本开始调度运行/etc/rc2.d 目录中以 Sxx 为起始文件名的所有 Shell 脚本，并按其排列顺序，依次启动相应的守护进程，从而完成最后的系统生成过程。

表 14-3 给出了/etc/rc2.d 目录中部分重要的 Shell 启动脚本，同时说明了在进入多用户运行模式时都会启动哪些守护进程或应用程序。注意，在不同版本的 Linux 系统中，甚至在同一版本的 Linux 系统中，由于选择安装的软件包不同，/etc/rc2.d 目录中的文件并不完全一样。实际上这一点并不重要，因为我们关心的只是这样一种机制，以及怎样利用这种机制，把应用程序的启动（与关闭）脚本加到系统的生成过程中，使应用程序能够随着系统的启动而自动启动，而并不关心目录中都有哪些启动文件，其功能如何。

表 14-3 /etc/rc2.d 目录中的部分重要 Shell 启动脚本

启 动 脚 本	简 单 说 明
S10acpid	用于监听系统内核的 ACPI 事件，管理和控制电源。对于笔记本电脑和部分台式机，建议启用这个守护进程，服务器需视具体情况酌情处理
S10sysklogd	用于启动系统日志守护进程 sysklogd。sysklogd 是一个系统范围的日志守护进程，其他许多守护进程均利用 sysklogd，把自己的输出信息记录到系统日志文件中
S11klogd	用于启动系统内核日志守护进程 klogd，以便获取 Linux 内核消息
S12dbus	用于启动 D-BUS 系统消息总线的守护进程 dbus-daemon。D-BUS 是一个系统范围的 IPC(Interprocess Communication) 消息总线，用于广播系统事件或消息，如通报设备的配置变动等。通常应当总是启用这个守护进程
S14avahi-daemon	用于启动 Avahi mDNS/DNS-SD 守护进程，提供名字解析功能，实现网络的自动配置
S15bind9	用于启动 DNS 域名服务器的 named 守护进程，实现域名与 IP 地址解析
S16ssh	用于启动 OpenSSH 守护进程 sshd。sshd 守护进程允许外部用户利用 SSH 协议访问本地系统。除非作为一个 Linux 系统，必须提供客户系统利用 SSH 协议访问本地系统的远程注册服务，否则不应启用这个守护进程（参见第 18 章）
S19autofs	用于启动 automount 守护进程，以便在用户进入安装点，访问远程共享资源时能够自动安装 NFS 网络文件系统
S19mysql	用于启动 MySQL 数据库服务器
S20apmd	apmd 用于监控电源的状态，并通过 syslog 记录电源的状态信息。当电源的电压较低时，也可用于关闭系统。如果系统支持 acpi，应当禁用 apmd（acpi 将会替代 apmd）
S20cups	用于启动通用 UNIX 打印系统（Common UNIX Printing System，CUPS）守护进程 cupsd
S20nfs-common	用于启动 rpc.statd 或 rpc.idmapd 等守护进程，为 NFS 服务器和 NFS 客户机提供底层支持
S20nfs-kernel-server	用于启动 NFS 服务器的 nfsd、nfsd4、lockd 和 rpc.mountd 等守护进程，提供 NFS 服务器功能，公布 /etc/exports 文件中设定的目录或文件系统等共享资源
S20openbsd-inetd	用于启动 inetd 网络监控程序，调度执行传统的网络守护进程。inetd 是一种特殊的网络总控服务，用于统一管理与调度运行多个网络守护进程。根据针对特定端口的访问请求，inetd 将会启动相应的守护进程。例如，telnet 通常利用端口 23 访问服务器，如果检测到访问端口 23 来的 telnet 连接请求，inetd 将会调度执行 telnetd 守护进程
S20samba	用于启动 Samba 服务器的守护进程 smbd 与 nmbd，提供 Linux 与 Windows 系统之间的文件与打印资源共享服务。除非确实需要，否则不应启用这个守护进程
S20sysstat	用于启动系统活动数据收集器 sadc，以便能够利用 sar 工具分析系统性能
S20vsftpd	用于启动 FTP 服务器，提供文件传输服务
S24dhcddb	用于启动 DHCP D-Bus 守护进程 dhcddb，协助 dhclient 实现网络接口的自动配置
S24hal	用于启动 hal 守护进程，收集与维护系统硬件信息



续表

启动脚本	简单说明
S25bluetooth	用于启动蓝牙(Bluetooth)守护进程 bluetoothd，管理蓝牙无线设备，如蓝牙无线键盘、鼠标和耳机等，支持基于蓝牙协议的拨号网络，以及连接以太网，等等。蓝牙是一种无线通信协议(非802.11)，支持便携式蓝牙无线设备，支持服务的发现、认证和人机接口设备等。有些笔记本电脑支持蓝牙无线设备，但大多数台式机、服务器或部分笔记本电脑并不支持蓝牙协议，因而应禁用这个守护进程
S28NetworkManager	用于选择并自动切换可用的网络连接。在同时支持多种网络连接手段(如无线网络与以太网)的电脑中实现网络的自动检测与切换
S30gdm	用于启动GNOME显示管理器，在控制台终端显示GNOME图形注册界面，使用户能够注册访问系统
S89anacron	用于调度运行因系统停机等原因而错失执行机会的cron作业。anacron是对cron的补充，如果停机期间存在应当运行而没有运行的cron作业，可由anacron辅助调度运行
S89atd	用于启动atd守护进程，调度运行利用at命令提交的定时执行作业；在系统的平均负载较低时，调度运行利用batch命令提交的作业
S89cron	用于启动后台作业调度守护进程cron。cron是一个标准的Linux守护进程，用于周期地调度定时执行的后台作业。系统应当总是启用cron守护进程，但atd或anacron则不一定需要
S91apache2	用于启动Apache服务器，提供网络浏览服务
S99rc.local	最后执行的一个启动脚本。用户可以利用这个启动脚本，增加自己的初始化功能，这样即可避免自己编写一个完整的启动脚本

在上述脚本文件中，每一个文件基本上都是采用case分支控制语句，按照start、stop和restart等情况，确定怎样启动、停止或重新启动相应的守护进程。

在了解了上述Shell启动脚本的基础上，可以根据系统的安装情况及具体配置，确定应当启用和禁用哪些守护进程，从而达到提高系统性能的目的。在不了解一个系统服务对系统究竟有什么影响时，一般不要轻易关闭系统提供的守护进程，尤其是不要轻易关闭cron、hald、klogd、syslogd及dbus-daemon等守护进程。

### 3. 其他任务

在正常的系统启动过程中，control-alt-delete等配置文件通常不会被处理。但在系统运行期间(包括在系统的生成过程中)，如果按下Ctrl-Alt-Del组合键，将会执行预定义的shutdown程序，示例如下。

```
$ cat /etc/event.d/control-alt-delete
# control-alt-delete - emergency keypress handling
#
# This task is run whenever the Control-Alt-Delete key combination is
# pressed. Usually used to shut down the machine.

start on control-alt-delete

exec /sbin/shutdown -r now "Control-Alt-Delete pressed"
$
```

在适当地处理完/etc/rcN.d目录中的所有Shell脚本之后，init进程将会按照/etc/event.d/ttYN(其中N为0~6)配置文件的定义，调度运行一个/sbin/getty进程，确保当系统处于2和3等运行级别时都打开一个虚拟控制台。

## 14.3.3 启动用户定义的应用程序

### 1. 创建作业配置文件

除了创建自己的启动脚本，利用init进程和/etc/rcN.d目录的控制机制，实现应用程序的自动启动之外，任何时间，系统管理员都可以创建作业配置文件，把需要执行的进程作为任务或服务加到/etc/event.d目录中，由init进程负责调度执行。此外，还可以考虑充分利用cron守护进程。

## 2. 修改 rc.local 文件

实际上，还可以采用其他方法在系统引导过程中启动用户定义的应用程序。在系统引导过程中或在改变运行级时，init 进程都会执行/etc/rc.local 脚本。根据这一特点，用户可以在这个脚本文件的后面，增加自己的命令，执行必要的任务。这一做法比较简单，能够避免在/etc/init.d 目录中编写复杂的初始化与启动脚本，然后在相应的/etc/rcN.d 中建立符号链接文件。

# 14.4 login 进程

## 14.4.1 login 进程与 passwd 文件

当用户在注册界面或“login:”提示下输入用户名之后，Linux 系统将会调用 login 进程，由 login 进程处理用户的注册。login 进程首先会提示用户输入密码，在用户输入密码时，键入的字符通常不回显。读取密码之后，login 进程将根据/etc/passwd 文件，比较输入的注册用户名和密码。如果有一项不符，login 进程将会再次提示用户输入用户名与密码。如果用户名和密码通过了验证，login 进程将会按照/etc/passwd 文件中“home-dir”与“login-shell”字段的设置（参见第 13 章），确定用户的初始工作目录和交互式注册 Shell，然后以用户名作为参数，通过 exec 系统调用，调度执行 Shell 程序。

## 14.4.2 Shell 进程与 profile 文件

如上所述，用户注册后究竟调用哪一个 Shell，取决于/etc/passwd 文件中的定义（默认的注册 Shell 为/bin/bash）。当调用一个注册 Shell 时，Shell 将会运行相关的启动文件，初始化各种必要的变量，设置运行环境。至于究竟运行哪些启动文件，取决于调用的是哪一个 Shell。

这里以 Bash 为例，说明注册 Shell 的启动过程。在开始运行之后，Shell 首先会读取并执行 /etc/profile 初始化文件，设置系统范围内的运行环境，如 PATH 和 PS1 等变量。期间，如果 /etc/bash.bashrc 文件存在，/etc/profile 将会间接调用该文件。然后进入用户主目录，执行其中的.profile (.profile 又调用.bashrc) 等用户初始化文件，进一步定制自己的运行环境。最后进入 GNOME 桌面（当调用终端窗口时，取决于用户的身份，输出后缀为“\$”或“#”的命令提示符，开启 Shell 会话过程）。下面是系统提供的标准/etc/profile 文件。

```
$ cat /etc/profile
# /etc/profile: system-wide .profile file for the Bourne shell (sh(1))
# and Bourne compatible shells (bash(1), ksh(1), ash(1), ...).

if [ -d /etc/profile.d ]; then
    for i in /etc/profile.d/*.sh; do
        if [ -r $i ]; then
            . $i
        fi
    done
    unset i
fi

if [ "$PS1" ]; then
    if [ "$BASH" ]; then
        PS1='\u@\h:\w\$ '
    else
        PS1='[\u@\h:\w]\$ '
    fi
fi
```



```
if [ -f /etc/bash.bashrc ]; then
    . /etc/bash.bashrc
fi
else
    if [ "`id -u`" -eq 0 ]; then
        PS1='# '
    else
        PS1='$ '
    fi
fi
fi

umask 022
$
```

至此，Linux 系统的整个启动过程全部完成，系统已处于待命状态。用户可以访问 GNOME 桌面环境，打开终端窗口，输入任何 Shell 命令，访问 Linux 系统。为了查询系统启动过程中系统内核输出的一系列引导信息，可在注册之后立即选择“应用程序→附件→终端”菜单，在终端窗口中输入下列命令，观察系统内核的引导信息。如果引导过程中出现问题，系统内核的引导信息有助于诊断系统问题，分析产生问题的原因。

```
$ dmesg | less
```

dmesg 命令能够显示系统内核环形缓冲区中存储的信息。在系统引导过程中，系统内核将会把引导过程中硬件与软件模块的初始化信息写到这个缓冲区中，同时也会把这些信息记录到 /var/log/dmesg 日志文件中。因此，也可以使用 more 或 less 命令查询 dmesg 日志文件。

顺便提一句，当用户退出系统，即退出 Shell 时，系统将会自动执行~/.bash\_logout 文件。因此，可以使用这个脚本文件执行某些善后处理，如清除临时文件等。

## 14.5 系统关机过程

### 14.5.1 使用 shutdown 命令关闭系统

shutdown 命令能够采用一种比较安全的方式关闭系统。在真正关闭系统之前，系统将会不断地通知或提醒已经注册到系统中的所有用户，使用户能够早做准备，及时保存尚未完成的处理工作，同时封锁其他用户再注册到系统。shutdown 命令也会向所有的进程发送 SIGTERM 信号，使进程能够执行善后处理。最终，shutdown 命令将会调用 init 进程，请求改变运行级。除了在指定的时间延迟之后实施实际的关机关动作之外，shutdown 命令也能够立即关机，其语法格式如下。

```
shutdown [-cchrHP] time [warning-message]
```

其中，“-r”选项用于重新引导系统。time 用于指定停机的时间。时间参数可以采用不同的表示格式：可以采用 24 小时制的 hh:mm 形式表示开始停机的绝对时间，其中 hh 表示小时，mm 表示分钟；也可以采用相对时间形式+m，其中 m 表示开始停机前缓冲时间（分钟数）。此外还可以使用专门的英文单词 now（即+0），表示立即执行关机。warning-message 用于指定发送给所有注册用户的消息。

在关机实际开始之前，系统将会按照一定的时间间隔向所有注册的用户发出警告信息。下面的例子表示在输入 shutdown 命令之后，将会在凌晨 1:10 关机，并在实际执行关机之前的 2 分和 1 分钟之际输出警告信息。

```
$ sudo shutdown -r 1:10
Broadcast message from gqxing@iscas
  (/dev/pts/0) at 1:08 ...
The system is going down for reboot in 2 minutes!
Broadcast message from gqxing@iscas
  (/dev/pts/0) at 1:09 ...
The system is going down for reboot IN ONE MINUTE!
```

### 14.5.2 使用 init 命令关闭系统

要快速地关机，可以直接使用“init 0”命令。在使用“init 0”命令关闭系统时，系统将会以0作为参数，执行/etc/init.d/rc脚本，然后执行/etc/rc0.d目录中以“Knn”为起始文件名的所有Shell脚本，最终关闭系统。示例如下。

```
$ sudo init 0
```

### 14.5.3 使用其他命令关机

除了shutdown和init命令之外，还可以使用halt、poweroff或reboot等命令关闭或重新启动系统。

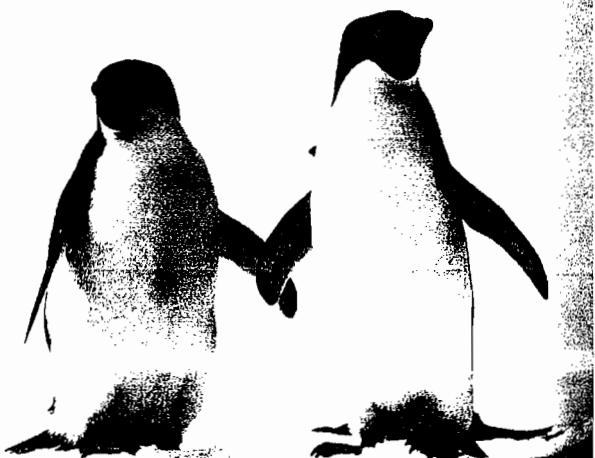


## 第15章

### 作业调度与系统日志

在 Linux 系统中，cron、atd 与 anacron 是 3 个重要的守护进程，负责调度各种定时运行的后台作业。本章主要讨论怎样实现后台作业调度，定时执行各种系统任务与用户程序，包括重复执行的任务以及一次性执行的作业，同时给出相应的应用实例。同时简单介绍系统提供的日志功能与各种日志文件。其中包括：

- 定时运行的后台作业；
- 调度重复执行的任务；
- 调度一次性执行的作业；
- 系统日志。



## 15.1 定时运行后台作业

在调度定时执行的系统任务与用户程序方面, cron、atd 与 anacron 这 3 个守护进程各有侧重与分工。cron 是 Linux 系统中的主要作业调度程序, 负责调度执行所有的系统服务; atd 负责调度执行用户利用 at(或 batch) 命令提交的定时作业; anacron 是对 cron 的补充, 如果存在停机期间应当运行而没有运行的 cron 作业, 在系统启动之后, 将由 anacron 协助调度执行。

利用后台作业调度机制, 用户也可以提交自己的后台作业, 以便能够定时、自动地执行自己的应用程序(包括系统命令和 Shell 脚本)。例如, 在每天晚上的某一时间或每周的周末执行系统备份, 按一定的时间间隔收集系统数据等。

表 15-1 给出了与作业调度有关的命令、守护进程、crontab 文件、cron 调度执行的 Shell 脚本以及相关的控制文件。

表 15-1 与作业调度有关的命令和文件

命令与守护进程	调度功能	crontab 文件	调度的脚本文件	控制文件
crontab/cron	在指定的时间重复执行指定的任务	/etc/crontab /etc/cron.d/* /var/spool/cron/crontabs/*	/var/cron.hourly/* /etc/cron.daily/* /etc/cron.weekly/* /etc/cron.monthly/*	/etc/cron.deny (选用的) /etc/cron.allow (选用的)
at/batch/atd	定时执行单个任务		/var/spool/cron/atjobs/*	/etc/at.deny (选用的) /etc/at.allow (选用的)
anacron	调度错失执行时间的作业	/etc/anacrontab	/etc/cron.daily/* /etc/cron.weekly/* /etc/cron.monthly/*	

注意, crontab 文件主要用于提交重复执行的任务, 而 at 命令则仅仅用于提交一次性执行的任务。

### 15.1.1 cron 守护进程的调度过程

cron 守护进程的主要功能是管理和调度以 crontab 文件形式提交的后台作业, 按照指定的日期和时间自动执行指定的命令。在系统的启动过程中, 通过执行/etc/init.d 目录中的 cron 脚本, Linux 系统将会自动启动 cron 守护进程。一经启动, cron 守护进程将首先检测/var/spool/cron/crontabs 目录中是否存在用户定义的, 并以用户名命名的 crontab 文件, 然后检测/etc/crontab 文件, 最后检测/etc/cron.d 目录中是否存在系统 crontab 文件。如果存在, cron 守护进程将会执行下列任务:

- 读取并加载 crontab 文件中定义的任务, 其中包括执行时间和相应的命令(或 Shell 脚本);
- 每分钟运行一次, 检查已加载到内存中的每一项任务, 如果某一任务的指定时间与当前的系统时间匹配, 则调度执行相应的命令或 Shell 脚本;
- 在执行指定的命令或 Shell 脚本时, 其中的任何输出信息将会通过电子邮件的形式发送给 crontab 文件的属主或由 crontab 文件中 MAILTO 环境变量指定的用户;
- 检查/var/spool/cron/crontabs 和/etc/cron.d 目录, 以及/etc/crontab 文件自加载后其修改时间是否发生变化。如有变化, cron 将会重新考察上述两个目录中的所有 crontab 文件, 重新加载已经编辑或修改过的 crontab 文件。

为了利用 cron 守护进程, 提交定时、重复执行的例行任务(命令或 Shell 脚本), 可以选用/etc/crontab



或/etc/cron.d 目录中的一个 crontab 文件，手工编辑选定的文件，增加自己的定时执行任务；也可以使用 crontab 命令，在/var/spool/cron/crontabs 目录中创建、编辑或删除自己的 crontab 文件（对于只需执行一次的命令或 Shell 脚本，可以使用 at 命令，提交 atd 守护进程，使之按照指定的时间调度运行）。

利用 crontab 文件，可以按时、按日、按周或按月调度执行各种日常的系统管理，使系统能够在指定的时间重复执行例行的维护任务，如根据预定的保留时间，轮换过时的日志文件，执行系统记账任务（参见/etc/cron.d/sysstat 文件）以及系统备份等。

此外，还可以利用 crontab 执行其他日常的系统管理与维护任务，定时备份数据库中的重要数据等（详见 15.2 节以及 15.2.7 小节中的应用实例）。

注意，请勿混淆 crontab 命令与所谓的 crontab 文件。实际上，Linux 系统既提供一个 crontab 命令，用于创建、编辑、显示和删除/var/spool/cron/crontabs 目录中的文件，也提供一个/etc/crontab 文件。但通常所谓的 crontab 文件实际上只是一种通称，泛指/etc/crontab 文件本身以及 /var/spool/cron/crontabs 与/etc/cron.d 等目录中的所有文件，尽管其文件名并非 crontab。

除了人为因素，一经启动，cron 守护进程通常会无休止地连续运行。而且，整个系统只能运行一个 cron 进程。这是因为，系统使用/var/run/crond.pid 文件作为封锁文件，防止用户执行多个 cron 进程。

### 15.1.2 at 作业与 atd 守护进程

利用 at 命令，用户能够提交在指定的时间开始执行的作业。作业可以是一个命令，也可以是一个 Shell 脚本。类似于 crontab，at 命令允许用户调度执行一定的系统管理与维护任务。但与 crontab 文件不同的是，at 命令提交的任务仅在指定的时间执行一次。执行之后，定期从其作业目录中删除相应的文件。因此，当需要一次性地定时运行某个命令或 Shell 脚本时，at 命令是非常有用的。注意，在利用 at 命令提交作业时，应把命令或 Shell 脚本的输出信息重定向并存储到一个文件中，以备将来查看。

当用户利用 at 命令提交 at 作业时，系统将会在/var/spool/cron/atjobs 目录中以文件方式保存用户提交的命令或 Shell 脚本，以及用户当前工作环境的副本。at 作业文件采用较长的字符串命名，文件名反映了提交的命令或 Shell 脚本在 at 作业队列中的位置及执行状态。

文件名的第一个字符表示作业的类型和状态，其中“a”表示利用 at 命令提交的作业，“b”表示利用 batch 命令提交的作业。如果文件名的第一个字符变为等号“=”，说明作业已经执行完毕。文件名的第 2~6 个字符以十六进制的数值形式表示作业的队列号。

例如，a00008012eb8e0 是一个利用 at 命令提交的作业，作业的队列号为 8；b0000b012f627f 是利用 batch 命令提交的作业，作业的队列号为 11；文件名“=00003012eb8ea”意味着作业已经执行。

一经启动，atd 守护进程即开始检测/var/spool/cron/atjobs 目录中是否存在 at 作业，同时随时监听用户是否利用 at 命令提交了新的作业。在完成 at 作业之后，atd 守护进程将会重新命名 /var/spool/cron/atjobs 目录中的作业文件，把文件名的第一个字符改为等号“=”，文件名的其他部分保持不变。15.3 节将会详细说明怎样提交定时执行的 at 作业。

此外，还可以利用 batch 命令提交可在任何时间执行的命令或 Shell 脚本，此时无需控制作业的执行时间。当存在一个或一组需要执行的命令，但不关心究竟何时执行时，batch 命令是特别方便的。atd 守护进程将会对提交的 at 作业进行排队，以便在系统负载允许的情况下调度执行。除非采用了重定向，命令的标准输出和标准错误输出将会通过电子邮件的方式发送给用户。

### 15.1.3 调度错失执行时间的任务

anacron 能够按照指定的天数作为时间间隔，周期地执行命令。与 cron 不同，anacron 假定计算机系统并非不间断地运行。因此，在一个非 24 小时连续运行的 Linux 系统中，可以利用 anacron 协助 cron 控制每天、每周或每月调度执行的作业。anacron 命令的语法格式简写如下。

```
anacron [-s] [-f] [-n] [-d] [-q] [-t newcrontab] [job] ...
anacron -u [-t anacrontab] [job] ...
```

其中，“-f”选项表示强制执行作业，忽略相关的时间属性。“-u”选项表示仅仅更新作业的时间属性，把作业的时间属性设置为当前日期，但并不调度执行任何作业。“-s”选项表示依次执行每个作业。在前一个作业尚未结束运行之前，anacron 通常不会开始调度执行新的作业。“-n”选项表示立即运行作业，忽略/etc/anacrontab 文件中规定的延迟时间。这个选项也蕴涵着“-s”选项。“-d”选项表示不创建后台进程。“-t”选项表示使用指定的文件替代默认的 anacrontab 文件。

当随着系统的启动开始运行时，anacron 将会从默认的配置文件/etc/anacrontab（除非另行指定）中读取需由 anacron 调度执行的一系列作业定义。每个作业定义包含以天数为单位的时间周期，以分钟为单位的时间延迟、作业标识符，以及指定的命令或 Shell 脚本。

对于每个作业，anacron 将会检查作业是否已经在最近的 n 天之内执行过，n 表示指定的时间间隔。如果未执行，anacron 将会在指定的时间延迟之后，调度执行指定的命令或 Shell 脚本。

在指定的命令或 Shell 脚本执行结束之后，anacron 将会在一个作业特定的时戳文件中记录作业的执行日期，以便能够确定何时需要再次调度运行。anacron 仅用这个日期进行时间计算。如果没有需要调度执行的作业，anacron 将会结束运行。/var/spool/anacron 目录用于存储 anacron 作业的时戳文件。

除非指定了“-d”选项，anacron 将会创建一个后台子进程，当子进程开始执行时立即结束自己的运行。另外，除非指定了“-s”或“-n”选项，anacron 还会在指定的延迟时间之后立即启动指定的作业。

如果作业经由标准输出或标准错误输出产生任何输出信息，这些输出信息将通过电子邮件的形式，被送交运行 anacron 的用户（通常为 root）。而一般的维护性信息则由 cron 送交 syslogd 处理。

anacron 的配置文件/etc/anacrontab 描述了 anacron 调度执行的作业。这个配置文件由两种文本行组成：作业描述行和环境变量赋值行。作业描述行的语法格式如下。

```
period delay job-identifier command
@period_name delay job-identifier command
```

其中，period 是以天数为单位的时间间隔，delay 是以分钟为单位的延迟时间。job-identifier 可以包含任何非空格字符及斜线字符“/”，用于标识 anacron 作业，同时用作作业时戳文件的名字。command 可以是任何命令或 Shell 脚本。@period\_name 表示预定义的时间周期，当前唯一可取的一个有效值是@monthly，其作用是确保每月仅运行一次，而不管一个月究竟有几天。

环境变量赋值行的语法格式如下。

```
variable=value
```

其中，variable 是环境变量名，value 是变量的值。配置文件中允许有空行或以注释符“#”为起始字符的注释行。下面是 Ubuntu Linux 系统提供的默认 anacrontab 文件。

```
$ cat /etc/anacrontab
# /etc/anacrontab: configuration file for anacron
# See anacron(8) and anacrontab(5) for details.
```



```
SHELL=/bin/sh  
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin  
  
# These replace cron's entries  
1 5 cron.daily nice run-parts --report /etc/cron.daily  
7 10 cron.weekly nice run-parts --report /etc/cron.weekly  
@monthly 15 cron.monthly nice run-parts --report /etc/cron.monthly  
$
```

本述配置文件中，`run-parts`是一个Shell脚本，用于执行指定目录参数中的所有Shell脚本文件（部分特殊的Shell脚本文件除外）。

## 15.2 调度重复执行的任务

本节开始详细介绍怎样创建、编辑、显示和删除`crontab`文件，讨论如何实现定期、定时、自动执行系统的日常管理与例行维护任务，以及怎样控制对`crontab`文件的访问。

### 15.2.1 crontab 文件的工作原理

除了`/etc/crontab`文件之外，Ubuntu Linux系统提供的`crontab`文件通常存储在`/etc/cron.d`目录中。此外，用户也可以创建以自己的用户名命名的`crontab`文件，以便调度、运行自己的各种处理任务。用户创建的`crontab`文件均位于`/var/spool/cron/crontabs`目录中。

`cron`守护进程每分钟考察一次利用各种`crontab`文件定义的后台作业，并按照每个`crontab`文件中的指令，定时执行指定的命令或Shell脚本。`crontab`文件通常由若干指令组成，每个指令的前5个字段用于设定日期和时间，最后一个字段用于设定应自动运行的命令或Shell脚本。这些信息告诉`cron`守护进程应在何时调度执行指定的命令或Shell脚本。

`crontab`文件分为系统`crontab`文件和用户`crontab`文件两种形式。其中，系统提供的`crontab`文件由下列两部分内容组成。第一部分是选用的，用于设置环境变量；第二部分包括一系列指令，每个指令由7个字段组成，中间以空格作为分隔符。

```
variable=value  
minute hour day month week user command
```

用户定义的`crontab`文件由下列两部分内容组成。第一部分是选用的，用于设置环境变量；第二部分包括一系列指令，每个指令由6个字段组成，中间以空格作为分隔符。

```
variable=value  
minute hour day month week command
```

运行时，`cron`将会自动设置若干环境变量，例如，把`SHELL`设置为`/bin/sh`，并根据`crontab`文件的属主，适当地设置`LOGNAME`和`HOME`变量。注意，在`crontab`文件中，用户可以强制修改`HOME`和`SHELL`变量的默认值，但不能强制修改`LOGNAME`变量。

除了`LOGNAME`、`HOME`和`SHELL`变量之外，`cron`还会寻找`MAILTO`变量，以便能够向用户报告运行结果。如果设置了`MAILTO`，`cron`将会向`MAILTO`变量定义的用户发送邮件。如果设置了`MAILTO`，但其变量值为空（`MAILTO=""`），则不会发送任何邮件，否则，`cron`将会向`crontab`文件的属主发送邮件。

在`crontab`文件的指令部分，每一行的前5个字段分别表示分、小时、日、月和周。如果是系统`crontab`文件，第6个字段是用户名，表示以什么身份运行后台作业。最后一个字段用于指定需要调度执行的命令或Shell脚本。根据前5个字段设定的日期和时间，系统能够自动地重复执行指定的

命令或Shell脚本。注意，命令字段中可以同时指定多个命令，命令之间需加分号“;”分隔符。表15-2给出了前5个字段的说明。

表 15-2

crontab 文件中的时间字段

时间字段	取值范围
分	0~59
小时	0~23
日	1~31
月	1~12
周	0~7 (0 或 7 均表示星期日)

在crontab文件的每个时间字段中，还可以使用下列特殊字符，以提高时间定义的灵活性。

- 可以使用逗号并列多个数值，如“1, 3, 5, 7, 9”。
- 可以使用减号“-”指定一个数值范围，如“1-9”。
- 可以使用除号“/”和一个数字表示增量，如“1-9/2”，相当于“1, 3, 5, 7, 9”。
- 可以使用星号“\*”通配符表示所有可能的合法数值。
- 除了数字之外，月和周两个字段还可以使用其英文名字的缩写（前3个字符），如 feb 表示二月，sun 表示星期日等。
- 可以在每一行的起始位置采用注释符“#”增加一个注释行或加一个空行。

例如，下列 crontab 指令表示，每个星期五的下午4时，在控制台上显示一条提示信息。

```
0 16 * * 5 root echo -e "\n***** Timesheets Due *****" > /dev/pts/0
```

下面是Ubuntu Linux系统提供的/etc/crontab文件，其主要目的是按小时、日、周和月，分别调度运行位于cron.hourly、cron.daily、cron.weekly和cron.monthly目录中的Shell脚本，以便完成不同的系统维护任务。

```
$ cat /etc/crontab
...
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
25 6 * * * root test -x /usr/sbin/anacron || (cd / && run-parts --report /etc/cron.daily)
47 6 * * 7 root test -x /usr/sbin/anacron || (cd / && run-parts --report /etc/cron.weekly)
52 6 1 * * root test -x /usr/sbin/anacron || (cd / && run-parts --report /etc/cron.monthly)
$
```

第一个指令表示在每小时的17分，调度执行/etc/cron.hourly目录中的crontab文件；第2个指令表示在每天的6:25，调度执行/etc/cron.daily目录中的crontab文件；第3个指令表示在每个星期日的6:47，调度执行/etc/cron.weekly目录中的crontab文件；第4个指令表示在每月第一天的6:52，调度执行/etc/cron.monthly目录中的crontab文件。

crontab文件中的每个指令必须由一个完整的逻辑文本行组成，即使文本行太长，需要占用多个物理行也是如此，中间不能有回车、换行字符。例如，下列/etc/cron.d/sysstat文件是sysstat软件包提供的一个crontab文件，其目的是定时收集系统性能的统计数据，供系统维护人员使用sar命令分析系统的性能，从中找出影响系统性能的原因。

```
$ cat /etc/cron.d/sysstat
```



```
DEFAULT=/etc/default/sysstat
# default setting, overriden in the above file
ENABLED=false
SAL_OPTIONS=""

# Activity reports every 10 minutes everyday
5-55/10 * * * * root [ -x /usr/lib/sysstat/sal ] && { [ -r "$DEFAULT" ] && . "$DEFAULT" ;
[ "$ENABLED" = "true" ] && exec /usr/lib/sysstat/sal $SAL_OPTIONS 1 1 ; }

# Additional run at 23:59 to rotate the statistics file
59 23 * * * root [ -x /usr/lib/sysstat/sal ] && { [ -r "$DEFAULT" ] && . "$DEFAULT" ;
[ "$ENABLED" = "true" ] && exec /usr/lib/sysstat/sal $SAL_OPTIONS 60 2 ; }
$
```

第一个指令表示在 5、15、…、55 分时，每 10 分钟运行一次/usr/lib/sysstat/sal 脚本，按指定的时间间隔（1s）和抽样次数（1 次）收集系统活动数据，并存储到/var/log/sysstat/sadd 文件中（其中的 dd 为 01~31 范围内的一个数字，表示当天的日期）；第二个指令表示在每天晚上 23:59 再次执行/usr/lib/sysstat/sal 脚本，以 60s 为时间间隔，收集 2 次系统活动数据。

## 15.2.2 创建和编辑 crontab 文件

在实际应用过程中，最简单、最正确的方法是利用“crontab-e”命令创建或编辑 crontab 文件。这个命令将会调用由 EDITOR 环境变量设定的文本编辑器。如果事先没有设置这个环境变量，crontab 命令将会提示用户选用一个编辑器，如 vim、ed 或 nano。

通常，无需具有超级用户的访问权限，任何用户都可以利用“crontab-e”命令，创建或编辑自己的 crontab 文件。如果 crontab 文件存在，则调用编辑器，打开现有的文件；如果 crontab 文件不存在，在写入并退出 nano 编辑器之后，创建一个新的 crontab 文件。创建的文件被自动存储在 /var/spool/cron/crontabs 目录中，并以用户的注册名命名。为了采用 vim 编辑器，可以事先定义 EDITOR 变量。示例如下。

```
$ EDITOR=vi; export EDITOR
$ crontab -e
```

如果是超级用户，可以使用下列命令形式，为自己或其他用户创建或编辑一个 crontab 文件。注意，只有具有超级用户的特权，才能创建或编辑其他用户的 crontab 文件。如果未指定用户名，则创建或编辑自己的 crontab 文件，并以 root 命名。

```
# crontab -e [-u username]
```

输入上述命令之后，crontab 将会调用事先设定或中间选定的编辑器。然后，可以利用编辑命令，按照 crontab 文件的语法格式，把新增的指令加到 crontab 文件中。

下面以运行 MySQL 数据库备份的 mysqldump 命令为例，说明怎样在 /var/spool/cron/crontabs 目录中创建一个以 gqxing 用户身份运行的 crontab 文件。首先输入下列 crontab 命令。

```
$ crontab -e
```

然后利用编辑命令，把下列两行内容（其中第一行为注释）加到打开的临时文件中，使系统能够在每天的午夜 23:55，利用 mysqldump 命令自动备份数据库、数据库表及其业务数据。

```
# Backup MySQL database at 23:55 everyday.
55 23 * * * gqxing mysqldump --add-drop-table -u gqxing -p1b2c3 books > /tmp/books.`date
'+%m%d'` >> /home/gqxing/backup/backup.log 2>&1
```

保存文件，退出编辑器。最终将会在 /var/spool/cron/crontabs 目录中创建一个名为 gqxing 的文件。

mysqldump 命令后面的“>> /home/gqxing/backup/backup.log 2>&1”意味着把其标准输出及标准错误输出重定向到 “/home/gqxing/backup/backup.log” 日志文件中，从而把 mysqldump 命令的

任何输出信息均记录下来，以备在上班时间随时查阅。

### 15.2.3 显示 crontab 文件

通常，普通用户不能直接访问/var/spool/cron/crontabs 目录。为了检验系统中是否存在 crontab 文件，超级用户可以使用“ls -l”命令访问/var/spool/cron/crontabs 目录，列出其中的文件。例如，下列输出信息表明存在一个属于用户 gqxing 的 crontab 文件。

```
$ sudo ls -l /var/spool/cron/crontabs
总用量 4
-rw----- 1 gqxing crontab 180 2008-11-14 11:19 gqxing
$
```

但是，任何用户均可使用“crontab -l”命令显示或检查属于自己的 crontab 文件。如同 cat 等命令能够显示任何文本文件的内容一样，“crontab -l”命令能够专门显示 crontab 文件的内容，且在输入这个命令时，无需指定 crontab 文件的路径名。

下面的例子说明了怎样使用“crontab -l”命令显示当前用户 gqxing 自己的 crontab 文件。

```
$ crontab -l
# Backup MySQL database at 23:55 everyday.
55 23 * * * mysqldump --add-drop-table -u gqxing -palb2c3 books > /tmp/books.`date
'+'%m%d'` >> /home/gqxing/backup/backup.log 2>&1
$
```

如果是超级用户，还可以利用下列命令形式，显示自己或其他用户的 crontab 文件。注意，只有具有超级用户的特权，才能查询其他用户的 crontab 文件。如果未指定用户名，则显示自己的 crontab 文件内容。

```
# crontab -l [-u username]
```

下面的例子说明了超级用户怎样显示其他用户的 crontab 文件。

```
$ sudo crontab -l -u cathy
# Backup MySQL database at 23:55 everyday.
59 23 * * * mysqldump --add-drop-table -u cathy -pitsme books > /tmp/books.`date ')+'%m%d'` >> /home/cathy/backup/backup.log 2>&1
$
```

### 15.2.4 删 除 crontab 文件

通常，/var/spool/cron/crontabs 目录的访问权限不允许普通用户直接删除 crontab 文件。要删除自己的 crontab 文件，可以采用下列命令。

```
$ crontab -r
```

上述命令只能删除属于自己的 crontab 文件，且无需指定 crontab 文件的路径名。如果是超级用户，可以利用下列命令，删除自己或其他用户的 crontab 文件。注意，只有具有超级用户的特权，才能删除其他用户的 crontab 文件。如果未指定用户名，则删除自己的 crontab 文件。

```
# crontab -r [-u username]
```

下面的例子说明了怎样使用“crontab -r”命令删除 gqxing 用户自己的 crontab 文件，然后使用“crontab -l”命令，验证 crontab 文件已经删除。

```
$ crontab -r
$ crontab -l
no crontab for gqxing
$
```

### 15.2.5 crontab 命令的访问控制

Ubuntu Linux 系统通过可选的/etc/cron.deny 文件或/etc/cron.allow 文件限制用户使用 crontab



命令。只有这两个文件认可的用户才能够使用 crontab 命令创建、编辑、显示和删除自己的 crontab 文件。如果需要，超级用户可以设置 cron.deny 或 cron.allow 文件，限制或允许指定的用户使用 crontab 命令。注意，只有超级用户才能够创建和访问 cron.deny 以及 cron.allow 文件。

cron.deny 和 cron.allow 文件可以包含一系列用户名，每个用户名占用一行。两个访问控制文件共同作用的结果如下：

- 如果 cron.allow 文件存在，则只有这个文件中列出的用户才能够创建、编辑、显示和删除 crontab 文件；
- 如果 cron.allow 文件不存在，则除了 cron.deny 文件中列举的用户之外，其他用户均可创建、编辑、显示和删除 crontab 文件；
- 如果 cron.allow 或 cron.deny 文件均不存在，任何用户都能够运行 crontab 命令。

初始安装 Ubuntu Linux 系统之后，系统并不提供 /etc/cron.allow 和 /etc/cron.deny 文件。按照上述说明，这意味着任何用户均可使用 crontab 命令。如果需要，超级用户可以创建一个 /etc/cron.deny 文件，增加用户名单，禁止其使用 crontab 命令，因而也就禁止其提交后台作业。

之后，除了 cron.deny 文件中列出的用户之外，其他任何用户均可运行 crontab 命令。如果创建了 cron.allow 文件，则只有其中指定的用户才能够运行 crontab 命令。

因此，为了禁止某些用户运行 crontab 命令，可以直接编辑 cron.deny 文件，增加禁止使用 crontab 命令的用户名单（每个用户名占一行）。例如，下面给出的 cron.deny 文件表示不允许 guest 和 visitor 两个用户使用 crontab 命令。

```
$ cat /etc/cron.deny
guest
visitor
$
```

为了使指定的用户能够提交 crontab 作业，也可以创建一个 cron.allow 文件，首先把用户名 root 加到 cron.allow 文件中（如果不把 root 用户加到 cron.allow 文件中，超级用户访问 crontab 命令时也会遭到拒绝），然后再增加其他用户名，每行一个。加入的用户即可使用 crontab 命令。

为了验证普通用户是否能够访问 crontab 命令，可以注册到相应的用户，然后执行下列 crontab 命令。

```
$ crontab -l
```

如果用户能够访问 crontab 命令，而且已经创建了 crontab 文件，上述命令将会显示出文件中的内容。否则，如果相应的 crontab 文件不存在，系统将会输出类似如下的错误信息。

```
$ crontab -l
no crontab for guest
$
```

上述信息说明，当前的用户要么位于 cron.allow，要么不在 cron.deny 文件列举的名单中。

如果用户无权运行 crontab 命令，不管相应的 crontab 文件是否存在，都会显示下列出错信息，意味着用户要么不在 cron.allow 文件列举的名单中，要么位于 cron.deny 文件中。

```
$ crontab -l
You (guest) are not allowed to use this program (crontab)
See crontab(1) for more information
$
```

### 15.2.6 应用实例——数据库定时备份

在 MySQL 数据库应用中，为了能够在每天的指定时间定时备份数据库中的数据，可以利用 cron 机制，创建自己的 crontab 文件，也可以在 /etc/cron.d 目录中创建一个新的 crontab 文件中，

增加一个数据库备份任务。

例如，假定有一个 MySQL 数据库应用项目，需要在每天的晚上 11 点 55 分，利用 MySQL 数据库的 mysqldump 命令备份一次数据。考虑到数据非常重要，系统中至少应保留 7 天的备份数据。

为此，可以利用 crontab 命令，以用户 gqxing 的名义创建一个 crontab 文件，在文件中增加一个定时执行 MySQL 数据库备份的 Shell 脚本，每天晚上 11:55 执行一次数据备份，命令如下。

```
55 23 * * * /home/gqxing/script/sqldump > /dev/null 2>&1
```

其中，sqldump 是一个 Shell 脚本，其功能是以 gqxing 用户的身份，执行 mysqldump 命令，把 gqxing 用户的数据库、数据库表及其业务数据备份到 books.mmdd 文件中。然后，检查备份文件的数量，如果超出 7 个文件，则删除早期的备份文件。sqldump 脚本的内容如下。

```
$ cat /home/gqxing/script/sqldump
#!/bin/bash
BAKDIR=/home/gqxing/backup
mysqldump --add-drop-table -u gqxing -p1b2c3 books > $BAKDIR/books.`date '+%m%d'`
amount=`ls -l books* 2>/dev/null | wc -l`
if [ "${amount}" -gt 7 ]
then
    fname=`echo books*`
    set $fname
    mv -r $1
fi
exit 0
$
```

采用上述备份方法之后，将会在/home/gqxing/backup 目录中保留 7 个数据库备份文件，示例如下。

```
$ cd /home/gqxing/backup
$ ls -l
总用量 28
-rw-r--r-- 1 gqxing gqxing 2760 2008-10-15 23:55 books.1015
-rw-r--r-- 1 gqxing gqxing 2760 2008-10-16 23:55 books.1016
...
-rw-r--r-- 1 gqxing gqxing 2760 2008-10-21 23:55 books.1021
$
```

## 15.3 调度一次性执行的作业

本节将开始介绍怎样使用 at 或 batch 命令执行下列任务：

- 提交 at 作业（命令或 Shell 脚本），使之在某个指定时间开始执行；
- 显示和删除已经提交的 at 作业；
- 控制用户是否能够使用 at 命令提交定时作业。

at 命令的语法格式简写如下。

```
at [-mld] time [date]
```

其中，“-m”选项表示，一旦作业执行之后立即向用户发送电子邮件。time 是以时分形式（HHMM 或 HH:MM）指定作业开始运行的时间。如果按 12 小时制指定时间，时间后面应附加 am 或 pm，其他可接受的关键字是 midnight、noon 和 now。如果指定的时间为整点时间，分可以省略。date 是以月日年（MMDYY、MM/DD/YY 或 MM.DD.YY）、“月名 日（如 June 1）”、星期几（如 Monday）、关键字 today 或 tomorrow 等表示的日期。为了简化输入，也可以使用月、星期几或其他特殊关键字的前 3 个字符。

通常，任何用户均可以创建、显示和删除自己的 at 作业文件，但只有超级用户才有权访问其



他用户的 at 作业文件。当利用 at 或 batch 命令提交一个 at 作业之后，系统将会赋予一个作业队列号码作为文件名的一部分，保留提交的 at 作业，以文件形式存储在 /var/spool/cron/atjobs 目录中，由 atd 守护进程负责处理以 at 或 batch 命令形式提交的作业。示例如下。

```
$ sudo ls -l /var/spool/cron/atjobs  
总用量 8  
-rwx----- 1 gqxing daemon 3232 2008-11-15 12:26 a0000a0137f982  
-rwx----- 1 gqxing daemon 3245 2008-11-15 12:49 b000100137f8e1  
$
```

文件名的第一个字符表示作业的类型和状态，其中“a”表示利用 at 命令提交的作业，“b”表示利用 batch 命令提交的作业。如果文件名的第一个字符变为等号“=”，说明作业已经执行完毕，暂时尚未清理。文件名的第 2~6 个字符以十六进制的数值形式表示作业的队列号。

例如，a0000a0137f982 就是一个利用 at 命令提交的作业，作业的队列号为 10 (0000a)；b000100137f8e1 是利用 batch 命令提交的作业，作业的队列号为 16 (00010)。

### 15.3.1 提交 at 作业

提交 at 作业包括 3 个要素：输入 at 命令，按照 at 命令的语法要求指定作业执行的时间，输入准备执行的命令或 Shell 脚本。

要提交一个 at 作业，可在系统命令提示符下输入 at 命令，同时指定作业的执行时间，按下 Enter 键；在 at 命令提示符“at>”下，输入准备执行的命令或 Shell 脚本；之后按下 Ctrl-D 键，即可提交 at 作业。

例如，下列命令提交的 at 作业将会在当天晚上的 6:30 删除当前用户 gqxing 主目录中的 core 文件。

```
$ at 1830  
warning: commands will be executed using /bin/sh  
at> rm -r /home/gqxing/core > /dev/null 2>&1  
at> <EOT>                                (即按下 Ctrl-D 键, 下同)  
job 18 at Sat Nov 15 18:30:00 2008  
$
```

如果希望同时输入多个命令或 Shell 脚本，每个命令或 Shell 脚本应占用一行，以 Enter 键结束。在输入所有命令或 Shell 脚本之后，按下 Ctrl-D 键，结束 at 命令的输入，从而完成 at 作业的提交。

例如，下列命令提交的 at 作业将会在 11 月 30 日午夜时分进入 /home/gqxing/backup 目录，压缩其中的所有数据文件。

```
$ at 11:59 pm nov 30  
warning: commands will be executed using /bin/sh  
at> cd /home/gqxing/backup  
at> bzip2 data*  
at> <EOT>  
job 19 at Sun Nov 30 12:00:00 2008  
$
```

当需要提交一个作业，但不关心究竟何时执行时，可以使用 batch 命令。下面仍以上述的数据文件压缩为例，说明怎样利用 batch 命令提交一个 at 作业，使之压缩 /home/gqxing/backup 目录中的数据文件，同时把开始执行时间和压缩完成时间等信息保存到一个日志文件 bzip2.log 中。

```
$ batch  
warning: commands will be executed using /bin/sh  
at> cd /home/gqxing/backup  
at> date >> bzip2.log  
at> bzip2 data*  
at> date >> bzip2.log  
at> <EOT>  
job 20 at Sat Nov 15 15:41:00 2008  
$
```

此外，还可以利用 Here 文档（参见第 8 章），提供 at 或 batch 命令需要的输入数据。示例如下。

```
$ batch <<!
> cd /home/gqxing/backup
> date >> bzip2.log
> bzip2 data*
> date >> bzip2.log
> !
warning: commands will be executed using /bin/sh
job 21 at Sat Nov 15 15:44:00 2008
$
```

注意，如果执行命令或 Shell 脚本的输出信息很重要，可以使用 I/O 重定向的方式把输出信息写到一个文件中，以便将来能够查阅。

### 15.3.2 显示 at 作业及作业队列

同样，普通用户也不能直接访问 /var/spool/cron/atjobs 目录。要查询已经创建的，且当前仍然位于 at 队列中的作业，可以使用 atq 或“at -l”命令。

在下面例子中，atq 命令列出了已经提交到 at 队列中的所有作业，以及 at 作业的执行时间信息。

```
$ atq
17   Sun Nov 30 12:00:00 2008 a gqxing
21   Sat Nov 15 15:50:00 2008 b gqxing
19   Sun Nov 30 23:59:00 2008 a gqxing
18   Sat Nov 15 18:30:00 2008 a gqxing
$
```

上述命令也会显示已提交的 at 作业的状态信息。其中的 a 表示利用 at 命令提交的作业，b 表示利用 batch 命令提交的作业，且两者尚未执行。

### 15.3.3 删 除 at 作业

在 at 作业执行之前，无需具有超级用户的特权，任何用户均可以使用下列命令，从队列中删除自己的 at 作业。其中，“job-id”表示 at 作业的编号。

```
$ atrm job-id
或
$ at -d job-id
```

如果是超级用户，可以利用同样的命令，删除自己或其他用户的 at 作业。注意，只有具有超级用户的特权，才能删除其他用户的 at 作业。如果未指定用户名，则删除自己的 at 作业。

在下面例子中，假定准备删除预定在 11 月 30 日午夜时分执行的 at 作业。首先，可以使用 atq 命令显示 at 作业队列，找出相应的作业号。接着，使用 atrm 命令从 at 作业队列中删除指定的作业。最后，检验指定的 at 作业已经被删除。

```
$ atrm 19
$ atq
17   Sun Nov 30 12:00:00 2008 a gqxing
21   Sat Nov 15 15:50:00 2008 b gqxing
18   Sat Nov 15 18:30:00 2008 a gqxing
$
```

删除指定的 at 作业之后，可以使用 atq 命令，验证指定的 at 作业已经被删除（此时，删除的 at 作业不应出现）。

### 15.3.4 at 命令的访问控制

Ubuntu Linux 系统通过 /etc/at.deny 文件或选用的 /etc/at.allow 文件限制用户使用 at 系列命令，



只有这两个文件认可的用户才能够使用 at 系列命令提交自己的 at 作业。如果需要，超级用户可以设置 at.deny 或 at.allow 文件，限制或允许指定的用户使用 at 系列命令。注意，只有超级用户才能够访问 at.deny 和 at.allow 文件。

at.deny 和 at.allow 文件之与 at 系列命令，犹如 crontab.deny 和 crontab.allow 文件之与 crontab 命令，其组合作用的结果可以限定用户是否能够提交 at 作业等。

安装 Ubuntu Linux 操作系统之后，系统提供的 at.deny 文件包含一系列系统管理用户。这意味着超级用户与任何普通用户均可使用 at 系列命令，创建、删除或显示自己的 at 作业及其队列信息。必要时，超级用户可以编辑这个文件，增加禁止使用 at 系列命令的用户名单，限制其使用 at 系列命令，禁止其提交 at 作业。

Linux 系统通常不提供 at.allow 文件。因此，在安装 Linux 系统之后，除了 at.deny 文件中列出的用户之外，其他任何用户均可以提交 at 作业；如果创建了 at.allow 文件，则只有其中指定的用户才能够提交 at 作业。

/etc/at.deny 文件由一系列用户名组成，每个用户占用一行。at.deny 文件中列举的用户不能使用 at 系列命令提交 at 作业。下面是经过修改之后的 at.deny 文件，其中增加的 guest 和 visitor 两个用户不能再使用 at 系列命令提交定时执行的作业。

```
$ sudo cat /etc/at.deny
...
guest
visitor
$
```

要验证新加入/etc/at.deny 文件中的用户是否能够使用 at 系列命令，可以使用 su 命令改变用户身份，然后运行 atq 命令。例如，如果已经限定用户 guest 不能使用 at 命令，系统将会输出下列信息。

```
$ su - guest
Password:
$ atq
You do not have permission to use atq.
$
```

同样，如果用户试图提交一个 at 作业，系统将会显示下列出错信息，说明用户 guest 属于 at.deny 文件列出的限制用户。

```
$ at 10:30 am Saturday
You do not have permission to use at.
$
```

### 15.3.5 应用实例——系统定时关机

为了在指定的时间关机，以应对用电高峰期间的停电或紧急系统维护，可以使用 wall 命令提前向注册的用户发出停机通知，示例如下。

```
$ sudo wall <<!
> THE SYSTEM iscas WILL BE SHUT DOWN AT 16:30 !!!
> Please log off ASAP or risk your files being damaged.
>

Broadcast Message from root@iscas
(/dev/pts/1) at 16:15 ...

THE SYSTEM dbserver WILL BE SHUT DOWN AT 16:30 !!!
Please log off ASAP or risk your files being damaged.
$
```

然后利用 at 命令，按照预定的时间关闭系统，示例如下。

```
$ sudo at 1630
warning: commands will be executed using /bin/sh
at> poweroff
at> <EOT>
job 29 at Sat Nov 15 16:30:00 2008
$
```

输入 poweroff 命令，接着按下 Enter 键和 Ctrl-D 键之后，将会看到 /var/spool/cron/atjobs 目录中增加了一个 at 作业文件，如下所示。

```
$ sudo ls -l /var/spool/cron/atjobs
total 4
-rwx----- 1 root daemon 1870 2008-11-15 16:16 a0001d0137fee0
$
```

查阅上述文件可以看到，这个 at 作业文件主要包括 3 部分内容。其中 atrun 和 mail 两行主要说明这是一个 at 作业，执行作业的用户身份是超级用户 (uid=0/gid=0)；从 umask 开始直至 cd 命令之前，主要是环境变量设置；最后一行即为提交的 poweroff 命令，示例如下。

```
$ sudo cat /var/spool/cron/atjobs/a0001d0137fee0
#!/bin/sh
# atrun uid=0 gid=0
# mail gqxing 0
umask 22
LS_COLORS=no=00:fi=00:di=01....: export LS_COLORS
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/X11R6/bin;
export PATH
LANG=C; export LANG
HOME=/home/gqxing; export HOME
COLORTERM=gnome-terminal; export COLORTERM
XAUTHORITY=/home/gqxing/.Xauthority; export XAUTHORITY
LANGUAGE=zh_CN:zh; export LANGUAGE
LOGNAME=root; export LOGNAME
USER=root; export USER
USERNAME=root; export USERNAME
SUDO_COMMAND=/usr/bin/at\ 1630; export SUDO_COMMAND
SUDO_USER=gqxing; export SUDO_USER
SUDO_UID=1000; export SUDO_UID
SUDO_GID=1000; export SUDO_GID
cd /home/gqxing || {
    echo 'Execution directory inaccessible' >&2
    exit 1
}
poweroff
$
```

## 15.4 系统日志

在 Ubuntu Linux 系统的启动及运行过程中，系统发生的任何重要事件，都会记入各种日志文件中。对系统或应用程序进行诊断或排错时，这些信息具有非常重要的参考价值。如果了解每个日志文件的作用，很快就能够从中找出自己需要的信息。

利用各种日志文件，Ubuntu Linux 系统提供了大量的系统事件、操作过程、说明信息以及错误信息。这些日志文件大多位于 /var/log 目录中，且是 ASCII 文本文件，采用标准的日志文件格式。此外，许多日志文件都是由系统日志守护进程 syslogd 根据系统和应用的配置要求生成的。而部分应用程序则生成并维护自己的日志文件，把需要记录的信息直接写入 /var/log 中某个子目录下的日志文件。

本节将简单介绍系统中的各种日志文件，以及其中包含何种日志信息。除了后续将要讨论的



部分二进制日志文件，大多数日志文件均可利用文件显示命令，如 less 查阅其中的内容，也可以利用 grep 命令，按照关键字检索其中的特定信息。如果想要查询最早出现的信息，可以使用 head 命令；如想查询最新的日志信息，可以使用 tail 命令等。

如果想要实时监控某个日志文件，可以使用 “-f” 选项运行 tail 命令。例如，如果想要实时观测客户系统如何访问本地主机中的 Apache 服务器，可以使用下列命令（在客户系统的访问过程中，终端上将会断续地出现随时产生的访问信息）。

```
$ tail -f /var/log/apache2/access.log
```

### 15.4.1 系统日志文件

所谓的系统日志文件主要是指 Ubuntu Linux 的基本系统功能，如授权机制、守护进程以及系统消息等使用的日志文件，不包括附加的系统应用或用户应用提供的日志文件。

#### 1. 授权日志

授权日志文件/var/log/auth.log 用于跟踪记录各种授权机制（提示用户输入密码），如 PAM 插件认证技术、sudo 命令、su 命令以及 OpenSSH 远程注册等报告的授权信息。要获取用户的注册过程、sudo 和 su 命令等的执行情况，可以查阅这个日志文件。

#### 2. 守护进程日志

守护进程日志位于/var/log/daemon.log 文件，其中主要包含与系统守护进程以及应用守护进程（如 GNOME 显示管理器守护进程 gdm、MySQL 数据库守护进程 mysqld）有关的信息。在获取各种守护进程的运行状况，对特定守护进程排错方面，这个日志文件中的信息是非常有用的。

#### 3. 调试信息日志

调试日志文件/var/log/debug 能够提供详细的调试信息，其中包括来自 Linux 系统内核或应用程序输出的，经由 syslogd 记录的 DEBUG 级别的信息。在调试程序时，如调试硬件驱动程序或应用程序时，其中的信息是非常有用的。

#### 4. 系统内核日志

系统内核日志位于/var/log/kern.log 文件，其中能够提供 Linux 系统内核报告的详细日志信息，包括系统配置信息，如 CPU、内存、磁盘、I/O 总线及以太网卡等的配置信息。对于一个新系统，尤其是对用户定制系统的故障排除，这些信息是非常有用的。

#### 5. 系统内核环形缓冲区

系统内核环形缓冲区并非实际存在的日志文件，而是位于系统内核的一个内存区，故无法直接查阅其中的信息。为了获取其中存储的系统内核引导信息，需要在命令行中运行 dmesg 命令。dmesg 命令主要用于考察和控制系统内核环形缓冲区，如在显示缓冲区中的引导信息，清除其中的数据，以及设置缓冲区记录的信息级别，等等。

实际上，/etc/init.d/bootmisc.sh 系统初始化脚本就是利用 dmesg 命令，把所有的系统引导信息写入/var/log/dmesg 日志文件。因此，也可以直接查询这个日志文件，获取系统引导过程中在系统控制台上显示的信息。

#### 6. 消息日志

消息日志文件/var/log/messages 主要包含系统中各种实用程序以及应用程序提供的一般性信息。为了考察实用程序以及应用程序输出的，经由 syslogd 或 sysklogd 守护进程记录的各种 INFO

级别的信息，可以查阅这个日志文件。

### 7. 系统日志

顾名思义，系统日志文件/var/log/syslog 是 Linux 系统中最大的信息集中存储位置，其中也许包含其他日志文件中没有留存的信息。如果从其他日志文件中找不到预期的信息，可以查询这个系统日志文件。

## 15.4.2 应用程序日志文件

在 Ubuntu Linux 系统中，除了上述的各种系统日志文件之外，还存在部分应用程序专用的日志文件。/var/log/apache2 目录中包含 Apache 服务器提供的日志文件，/var/log/samba 目录中包含 Samba 服务器提供的日志文件。

### 1. Apache 服务器日志

在 Ubuntu Linux 系统中安装 Apache 服务器时，通常会在 /var/log 目录中创建一个 apache2 子目录（/var/log/apache2），其中包含两个具有不同用途的日志文件（详见第 21 章）。

- /var/log/apache2/access.log——包含 Apache 服务器所有客户系统的访问记录。
- /var/log/apache2/error.log——包含 Apache 服务器报告的所有出错记录。

### 2. CUPS 打印系统日志

通用 UNIX 打印系统（Common Unix Printing System，CUPS）采用 /var/log/cups/error\_log 作为默认的日志文件，存储各种日志或错误信息。如果打印方面出现了问题，可以查阅或检索这个日志文件，从中找出端倪。

### 3. Rootkit Hunter 日志

Rootkit Hunter 实用程序 rkhunter 用于侦测 Ubuntu Linux 系统中是否存在隐秘的漏洞，危及系统的安全。rkhunter 使用 /var/log/rkhunter.log 作为其日志文件。

### 4. Samba 服务器日志

Samba 服务器是 Linux 与 Windows 系统之间常有的一种文件共享技术。Samba 在 /var/log/samba 目录中维护下列 3 种不同类型的日志文件。

- log.nmbd——包含基于 IP 协议的 NETBIOS 网络通信方面的信息。
- log.smbd——包含 Samba 服务器启动，以及 SMB/CIFS 文件与打印共享方面的信息。
- log.ip\_address——用于记录来自特定客户系统的服务请求信息，文件名中的 ip\_address 是客户机的 IP 地址，如 log.192.168.100.1。

### 5. X11 服务器日志

在 Linux 系统中，默认的 X 服务器是 Xorg X11 服务器，如果计算机只配有一个显示器，则其日志文件为 /var/log/Xorg.0.log。X11 服务器的日志信息有助于诊断 X11 环境中出现的问题。

## 15.4.3 无法直接查阅的日志

/var/log 目录中的部分日志文件是二进制数据文件，不能直接查阅，只能使用特定的程序对其进行读写。

### 1. 注册失败日志

注册失败记录位于 /var/log/faillog 日志文件，这是一个典型的二进制数据文件，不能直接读取，需要时可利用 faillog 命令查阅。



## 2. 最近一次注册日志

最近一次（或者说最后一次）用户注册的时间记录位于`/var/log/lastlog` 日志文件中。这也是一個典型的二进制数据文件，需要使用`lastlog` 命令查阅。

## 3. 用户注册日志

`/var/run/utmp` 和 `/var/log/wtmp` 日志文件包含系统中每个用户的注册记录。在调度各种进程时，系统将会自动记录每个进程的起止运行时间。对于当前活动的进程，其相关信息均记录在 `/var/run/utmp` 文件中。当收到一个信号，告知其调度的进程已经终止运行时，如果 `/var/log/wtmp` 文件存在，系统将会把相应的进程信息转录到 `wtmp` 文件中，说明进程是什么时间启动和终止的，以及终止的原因等。也就是说，`/var/log/wtmp` 文件将会记录整个进程调度的历史。

`/var/log/wtmp` 文件中记录的部分信息如下（详见`/usr/include/bits/utmp.h` 文件的定义或 `utmp/wtmp` 手册页）：

- 进程或记录类型（其中，1 为运行级改变时间，2 为系统引导时间，5 为 init 调度的进程，6 为 login 进程，7 为用户进程，8 为终止的进程等）；
- 进程 ID；
- 缩写的终端设备名（如 pts/2 等）；
- `/etc/inittab` 文件的 id 字段（保留字段）；
- 用户名；
- 远程系统名（远程注册时）；
- 进程终止及出口状态；
- 时间记录（即进程或会话的起止运行时间）；
- 远程系统的 IP 地址。

与 `/var/log/lastlog` 日志文件不同，`/var/log/wtmp` 文件记录的是用户的注册与工作状态，如注册访问方式（终端设备名、远程系统名或 IP 地址等）、注册会话的起止运行时间等。利用 `who` 命令可以查询当前注册的用户信息，而利用 `last` 命令可以查询 `wtmp` 文件记录的用户注册历史等信息。如果再使用 `last` 命令的“-f”选项，指定`/var/log/wtmp.N` 作为输入文件（其中 `N` 是一个从 1 开始编号的数字），还可以查询早期的用户注册历史等信息。例如：

```
$ last
gqxing  pts/1      169.254.78.56   Mon Jun  1 11:40  still logged in
gqxing  pts/1      169.254.78.56   Mon Jun  1 11:40 - 11:40 (00:00)
gqxing  pts/0      :0.0          Mon Jun  1 10:55  still logged in
gqxing  tty7       :0            Mon Jun  1 10:49  still logged in
reboot  system boot 2.6.27-9-generic Mon Jun  1 10:47 - 11:40 (00:53)
...
$
```

### 15.4.4 系统日志守护进程

系统日志守护进程 `syslogd`（也称作 `sysklogd`）是一个重要的系统守护进程，它以后台方式运行。`syslogd` 守护进程是一个总控开关，用于接收来自各种子系统或应用程序的信息，记录系统范围的日志信息，如果子系统或应用程序拥有自己专用的日志文件，则把信息交予系统或应用程序特定的日志文件。考虑到系统的存储空间，每个日志文件均采用先进先出的方式，保存最新的

日志信息。当超过日志文件的容量限制，过期的信息将会逐步移出日志文件。

**syslogd** 记录的信息通常包含若干重要的通用字段，如记录时间、主机名以及信息的来源（如来自系统内核还是应用程序）等。

### 1. 配置 **syslogd**

**syslogd** 是在系统启动时自动启动的日志守护进程，其根据/etc/syslog.conf 配置文件的定义，确定应当如何记录系统日志。如有必要，可以修改 **syslogd** 的配置文件，定制日志文件应记录的数据内容及其格式。**/etc/syslog.conf** 配置文件主要包含两个字段，其语法格式如下。

**Selector action**

其中，**selector** 是一个选择器，本身包含两部分数据，即写入日志文件的主体程序与优先级，中间以句点“.”作为分隔符。优先级也称日志信息级别，从低到高依次为 **debug**（调试信息）、**info**（普通信息）、**notice**（提醒信息）、**warning**（警告信息）、**err**（错误信息）、**crit**（重要信息）、**alert**（警报信息）及 **emerg**（紧急信息）。设定的优先级越低，记录的日志信息越多。**action** 动作字段用于定义日志文件，即日志信息的记录目的存储位置，如标准的日志文件/var/log/syslog，专用的日志文件或把日志信息发送到远程主机的主机名（如@somehost）等。

**/etc/syslog.conf** 配置文件的设置非常灵活，能够采用任何组合方式定义选择器与动作字段。

下面是取自/etc/syslog.conf 配置文件的部分内容。

```
$ cat /etc/syslog.conf
...
auth,authpriv.*          /var/log/auth.log
*.*,auth,authpriv.none   -/var/log/syslog
#cron.*
daemon.*                /var/log/daemon.log
kern.*                   /var/log/kern.log
lpr.*
mail.*                  /var/log/mail.log
user.*                   -/var/log/user.log
...
$
```

在上述配置中，优先权部分的星号“\*”表示所有的信息级别，意味着应记录所有的日志信息；日志文件名前的减号“-”表示禁止在每次写入日志文件之后立即执行内存与磁盘的数据同步。

### 2. 利用 **logger** 命令提交日志信息

Unbuntu Linux 系统提供了一个日志工具 **logger**，使用户能够把自己的信息随时加入系统日志文件/var/log/syslog。这一个非常有用的工具，可用于系统维护脚本，如 Perl 或 Shell 脚本文件中，提供标准的日志功能。**logger** 命令的语法格式如下。

```
logger [-s] [-f file] [-p pri] [-t tag] [-u socket] message
```

其中，“-s”选项表示既把指定的信息写到标准输出，也写入日志文件；“-t”选项表示在写入日志文件的每一行信息之前增加一个指定的标志；“-f”选项表示把信息写入指定的文件，而不是默认的日志文件/var/log/syslog；**message** 表示写入日志文件中的信息。

例如，如果在命令行或脚本文件中使用下列命令，系统将会在运行期间把指定的信息插入 /var/log/syslog 日志文件。

```
$ logger This is a testing for user logging
```

```
$
```

使用下列命令，将会在/var/log/syslog 日志文件中看到给定的信息及其运行时间。

```
$ grep testing /var/log/syslog
```

```
Sep 27 23:22:16 iscas gqxing: This is a testing for user logging
```



在 Shell 脚本中，更专业的方法则是利用“-t”选项，在插入系统日志文件中的信息中增加一个标志字段（而非使用默认的用户名），标示信息的来源，以区别于其他日志信息。示例如下。

```
$ logger -t MyINFO This is a testing for user logging  
$ grep MyINFO /var/log/syslog  
Sep 27 23:22:20 iscas MyINFO: This is a testing for user logging  
$
```

此外，还可以利用“-s”选项，在写入日志文件的同时，也在终端窗口上显示给定的信息。

示例脚本如下。

```
$ cat ~/script/chkroot.sh  
#!/bin/bash  
# logger demo -- use logger utility in Shell script  
#  
logmsg="/usr/bin/logger -s -t MyScript "  
  
# Print following message, write message to syslog in the same time  
$logmsg "Root account status checking script."  
  
# Test the status of root account on current machine  
status=`grep root /etc/shadow | cut -d: -f2`  
if [ "$status" = "*" ]; then  
    $logmsg "The root account is locked."  
else  
    $logmsg "The root account has unlocked."  
fi  
$
```

取决于超级用户账号的具体状态，执行上述脚本的结果如下（假定超级用户账号已经解锁）。

```
$ sudo ~/script/chkroot.sh  
[sudo] password for gqxing:  
MyScript: Root account status checking script.  
MyScript: The root account has unlocked.  
$ grep MyScript /var/log/syslog  
Sep 28 23:52:18 iscas MyScript: Root account status checking script.  
Sep 28 23:52:18 iscas MyScript: The root account has unlocked.  
$
```

上述输出和日志文件的检索结果表示，指定的信息既通过标准错误输出在终端窗口显示，也同时被写入了 syslog 日志文件。

### 3. 日志文件的轮换

上述的任何日志文件 *filename.log*，除了主日志文件之外，/var/log（及其子目录）中还包含 *filename.log.0*、*filename.log.1.gz*、*filename.log.2.gz* 及 *filename.log.3.gz* 等形式的日志文件，这些文件是逐步轮换出的旧日志文件。也就是说，如果超过预定的时间，系统将会自动地删除最后一个文件，如 *filename.log.3.gz*，然后依次重新命名日志文件，把 *filename.log.2.gz* 重命名为 *filename.log.3.gz*，把 *filename.log.1.gz* 重命名为 *filename.log.2.gz*，再利用 gzip 命令，把压缩后的 *filename.log.0* 重命名为 *filename.log.1.gz*，把 *filename.log* 重命名为 *filename.log.0*，最后重新创建新的主日志文件。日志文件的轮换与压缩，其主要目的是在节省磁盘空间的前提下，仍然保留一定的备份数据，以便用户查询其中的信息。

为了实现日志文件的轮换，Ubuntu Linux 系统利用/etc/cron.daily/logrotate 脚本（也可以利用 /etc/cron.daily/sysklogd 脚本），由 cron 守护进程定时调用 logrotate 命令；logrotate 命令再根据 /etc/logrotate.conf 配置文件或单独加到 /etc/logrotate.d 目录中的其他配置文件（如 apache2 和 mysql 配置文件等）的定义，确定每个日志文件的轮换要求。

# 第16章

## 文件系统内部组织

本章主要介绍 Linux 系统中流行的 Ext2/Ext3 文件系统在磁盘或磁盘分区中的详细布局，及其内部组织结构，并以 Ext2/Ext3 文件系统为例，详细讨论文件系统的核心——超级块、信息节点、目录的构成、文件的读写过程，以及信息节点与目录文件的关系，以使读者对文件系统能有深入的了解和认识。其内容主要包括：

- 文件系统的组织结构；
- 超级块；
- 信息节点；
- 信息节点与目录和文件的关系。





## 16.1 文件系统的组织结构

Linux 系统的第一个版本最初是在 MINIX 系统上开发的，为了便于在两个系统间共享文件，Linux 采用且唯一支持的文件系统也是 MINIX 文件系统。但这个文件系统存在两个致命的问题：一是容量较小，文件系统与文件的最大容量均不能超过 64MB；二是与传统 UNIX System V 文件系统一样，文件名的长度也不能超过 14 个字符，因而限制了 Linux 系统的功能。

为了支持新的文件系统，且能够容易地把新的文件系统加入系统内核，Linux 系统采用了与 UNIX 同样的方法，开发了虚拟文件系统（Virtual File System）。接着，于 1992 年实现了第一个扩展的文件系统（Extended File System，简称 Extfs 或 Ext），这个文件系统解决了 MINIX 文件系统的两个限制问题，把文件系统和文件的容量扩展到 2GB，文件名也可以长达 255 个字符。尽管如此，由于其采用链接表的方式管理空闲数据块和信息节点等问题，使得文件系统的性能表现不佳。

鉴于上述问题，后来又设计出第 2 版的扩展文件系统（简称 Ext2fs 或 Ext2）。在 Ext 文件系统的基础上，Ext2 文件系统对其组织结构做了重大的改造与调整，经过广泛的应用和不断改进，Ext2 目前已成为一个比较稳定和功能完善的文件系统，且成为 Linux 系统中的标准文件系统，大多数 Linux 基本上都支持这一文件系统或以 Ext2 为主的文件系统。

针对日志文件系统 JFS（Journal File System）的挑战，新推出的第 3 版扩展文件系统（简称 Ext3fs 或 Ext3）也支持日志功能。与 Ext2 相比，Ext3 实际上只是增加了日志功能，与 Ext2 文件系统是向下兼容的，而且现有的 Ext2 很容易升级到 Ext3 文件系统。因此，本章将以 Ext2/Ext3 文件系统为例，介绍 Linux 文件系统的物理布局与内部组织结构，以便读者了解文件系统的工作原理。

由于采用了虚拟文件系统的概念，Linux 系统能够同时支持多种不同类型的文件系统。目前，大多数 Linux 系统基本上都支持 Minix、Ext2、Ext3、Xia、ReiserFS、XFS、JFS、ISO-9660，以及微软的 FAT 与 NTFS 等文件系统。

文件系统主要是在磁盘介质上实现的。磁盘通常由一组盘片和磁头组成，围绕圆心，每个盘片被划分为若干磁道。然后以磁盘控制器及其驱动程序一次能够读写的数据量作为基本的数据处理单位，再把每个磁道划分为若干扇区（或称物理数据块，简称为物理块或数据块），每个扇区的存储容量为 512 字节。为了提高数据传输的吞吐能力和访问速度，文件系统通常均以 1KB、2KB、4KB 甚至 8KB 作为数据存储和处理的基本单位，称作逻辑数据块（在不至于混淆的情况下也简称为数据块）。

文件系统将磁盘或磁盘分区的所有数据块看作一个连续的线性存储区，每个数据块均可通过其块号地址进行访问。磁盘或磁盘分区中的部分数据块用于存储文件系统的管理与维护信息，其他大部分存储空间用于存储实际的数据。

在 Linux 系统支持的文件系统中，Ext2 文件系统是最有代表性的文件系统，其内部的组织结构也比较直观。Ext3 文件系统只是在 Ext2 文件系统的基础上增加了日志功能，以便在文件系统因故遭到破坏时能够快速地恢复数据。Ext3 与 Ext2 文件系统的硬盘布局与组织结构是一样的，其差别仅仅是保留部分特殊的信息节点（日志文件）与数据块，用作文件系统的日志。

Linux 中的所有文件系统均继承和采用了 UNIX 文件系统的概念，虽然其内部的组织形式并不完全相同，但基本上都是由引导块、超级块（super block）、信息节点区及数据存储区等部分组

尤其是 Ext2/Ext3 文件系统，充分借鉴了 UFS 文件系统中的柱面组概念，可以说，其物理上受伯克利版 UFS 文件系统的影响。

Ext2/Ext3 文件系统由若干数据块组 (block group) 组成，其数据块组相当于 UFS 文件系统中的柱面组。但 Ext2/Ext3 文件系统的数据块组并不过多地依赖磁盘数据块的物理组织与布局，因为磁盘设备在顺序访问方面都做了优化处理，隐藏了其物理实现的细节，无需外界过分干预。在创建 Ext2/Ext3 文件系统之前，文件系统创建程序 mkfs 首先需要把磁盘（或磁盘分区）划分为一系列大小规格相同的逻辑数据块。Ext2/Ext3 文件系统支持 3 种不同规格的逻辑数据块：1 024 字节、2 048 字节和 4 096 字节。在每个 Ext2/Ext3 文件系统中，所有逻辑数据块的大小都是固定的，但不同的 Ext2/Ext3 文件系统，逻辑数据块的大小可以有区别。逻辑数据块的大小在创建文件系统时确定，可以由系统管理员指定，也可以由 mkfs 命令根据磁盘或磁盘分区的大小，自行选择一个适当的默认值（如 1 024 字节或 4 096 字节）。一经确定，在文件系统的整个生命周期中，逻辑数据块的大小是不能改变的，除非重建文件系统。

在划分好逻辑数据块之后，mkfs 命令将根据 “-g” 选项指定的逻辑数据块数量（或采用默认值），把全部逻辑数据块划分为若干数据块组，使每个数据块组中具有相同数量的逻辑数据块（最后一个数据块组除外）。然后，再以数据块组为单位，组建文件系统。同样，一经确定，每个数据块组中拥有的逻辑数据块数量也随之固定，同时也确定了一个 Ext2/Ext3 文件系统拥有多少个数据块组。图 16-1 给出了 Ext2/Ext3 文件系统在磁盘或磁盘分区中的物理布局。

由图 16-1 可知，一个 Ext2/Ext3 文件系统由多个数据块组组成。每个数据块组均包含一个数据块组描述，同时也包含各自的数据块位图、信息节点位图、信息节点区以及数据块区。但是，除了第一个数据块组（编号为 0）包含一个主超级块之外，只有部分数据块组包含冗余的超级块副本。

Ext2/Ext3 文件系统具有 3 个主要特点：采用较大的逻辑数据块（1 024 字节、2 048 字节或 4 096 字节），以便提高数据传输的吞吐能力；以数据块组为单位组织文件系统，借以提高数据的访问速度；采用多个超级块，每个数据块组各存储一个副本，从而提高文件系统的可靠性。下面将进一步说明文件系统的组织结构，以及各种组成部分的基本功能。

### 16.1.1 引导块

每个 Ext2/Ext3 文件系统，在其磁盘或磁盘分区的起始位置（数据块 0）都有一个 1KB 的引导块，用于存储引导系统时使用的引导程序或数据。同 UNIX 系统中的引导块一样，Linux 系统实际上并不使用这个引导块启动系统。但是，即使文件系统所在的磁盘不是系统盘，文件系统也不用于引导系统，仍然保留引导块的位置，只是其中并不存储任何信息。

### 16.1.2 数据块组

数据块组 (block group) 的管理与控制信息用于维护其所在的数据块组。一旦安装了文件系统之后，数据块组中的信息将会随着文件系统的操作，不断地发生相应的变化。在 Ext2/Ext3 文件系统中，每个数据块组都由下列 6 部分组成（参见图 16-2）。

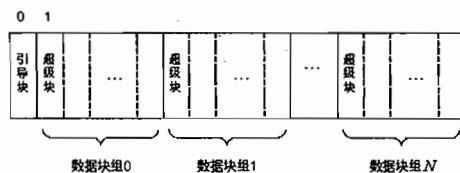


图 16-1 Ext2/Ext3 文件系统的物理布局



- 超级块。超级块是文件系统的核心，其中含有文件系统的关键数据及各种汇总信息，用于描述文件系统的总体结构，管理和维护文件系统。其中包括信息节点的总数、逻辑数据块的总数、空闲的逻辑数据块计数、空闲的信息节点计数、逻辑数据块的大小、信息节点的大小、每个数据块组拥有的逻辑数据块与信息节点的数量以及文件系统的状态信息等。
- 数据块组描述。其中包括数据块组中现有目录的数量，空闲逻辑数据块与空闲信息节点的数量。
- 数据块位图。用于标记数据块组中已用或空闲的数据块。
- 信息节点位图。用于标记数据块组中已用或空闲的信息节点。
- 信息节点区。用于存储文件的信息节点。一个信息节点含有除文件名与信息节点号之外的所有文件属性信息，如文件的类型、访问权限及文件大小等。
- 数据块区。用于存储文件的实际数据。

### 1. 超级块

超级块是文件系统的核心，其中包含文件系统的大小以及空闲信息节点与数据块的统计计数，供创建文件申请磁盘空间时使用。当启动 Linux 系统，把文件系统安装到系统后，系统将会把超级块读入内存。磁盘上的超级块信息是静态的，而位于内存中的超级块信息是动态的。在系统运行期间，系统将会不断地更新位于内存中的超级块，并定时地把更新后的超级块写回磁盘中。当系统因意外事件而未正常关机时，有可能会造成两个超级块中的信息不一致，因而引起文件系统的损坏。出现此类情况之后，当再次引导系统时，Linux 系统可能不得不修复文件系统，严重时甚至无法启动系统。

### 2. 数据块组描述

数据块组描述 (group descriptor) 位于超级块之后，用于描述数据块组的基本信息。其中主要包括下列内容（参见图 16-2）：

- 数据块位图的起始地址；
- 信息节点位图的起始地址；
- 信息节点区的起始地址；
- 空闲数据块计数；
- 空闲信息节点计数；
- 现有目录计数。

这里所说的数据块地址指针指的是磁盘（或磁盘分区）中数据块的块号。例如，如果地址指针的值为 100，则表明指向磁盘或磁盘分区中的数据块 100。下面是数据块组描述的结构定义。

```
struct ext3_group_desc
{
    __le32 bg_block_bitmap;           /* 数据块位图的起始地址 */
    __le32 bg_inode_bitmap;          /* 信息节点位图的起始地址 */
```

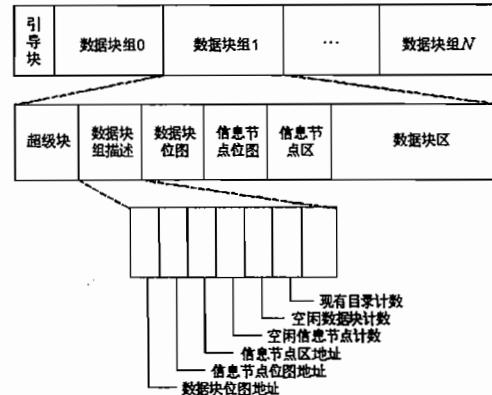


图 16-2 Ext2/Ext3 文件系统的组织结构

```

__le32 bg_inode_table;           /* 信息节点区的起始地址 */
__le16 bg_free_blocks_count;    /* 空闲数据块计数 */
__le16 bg_free_inodes_count;    /* 空闲信息节点计数 */
__le16 bg_used_dirs_count;     /* 现有目录计数 */
__le16 bg_pad;
__le32 bg_reserved[3];
};

```

### 3. 数据块位图

在数据块组中，最重要的数据是数据块和信息节点的分配位图。在每个数据块组的数据块组描述中，都存在两个位图：数据块位图与信息节点位图。在数据块位图中，每个二进制数据位表示一个逻辑数据块，用于标记一个逻辑数据块的分配与空闲状态。位图中的所有二进制数据位均初始化为“1”。如果某个二进制数据位为“0”，表示相应的逻辑数据块已经分配，其中已存有数据；如果某个二进制数据位为“1”，则表示相应的逻辑数据块为空闲（参见图 16-3）。

数据块位图（包括信息节点位图）通常仅占用一个逻辑数据块。因此，如果逻辑数据块的大小为 4 096 字节，则一个数据块位图只能确定  $8 \times 4\,096$ （即 32 768）个逻辑数据块的位置。也就是说，每个数据块组的数据块区中最多只能容纳 32 768 个逻辑数据块。

### 4. 信息节点位图

同样，信息节点也使用位图表示其分配与空闲状态。信息节点位图中的每个二进制数据位表示一个信息节点，对应于文件系统中的一个文件或目录。如果一个二进制数据位为 0，则表示相应的信息节点已经被占用；如果某个二进制数据位为 1，则表示相应的信息节点为空闲。

信息节点位图通常也占用一个逻辑数据块，如果逻辑数据块的大小为 4 096 字节，则一个信息节点位图可以寻址  $8 \times 4\,096$ （即 32 768）个信息节点。在 Ext2/Ext3 文件系统中，每个信息节点本身通常为 128 字节，因此，一个 4 096 字节的逻辑数据块可以容纳 32 个信息节点，而一个信息节点区最多占用 1 024（ $32\,768/32$ ）个逻辑数据块。

### 5. 信息节点区

数据块组描述中的信息节点区起始地址指向随后的信息节点区。按照上面的计算，一个信息节点区最多占用 1 024 个逻辑数据块。

同 UNIX 的 System V 文件系统一样，在 Ext2/Ext3 文件系统的信息节点区中，信息节点也是从 1 开始编号的（Linux 系统不允许使用 0 作为信息节点号，在目录文件中，如果信息节点号为 0，则表示相应的目录项为空）；信息节点 1 用于虚拟文件系统，信息节点 2 总是用作根目录的信息节点。根目录的信息节点是在建立文件系统时分配的。除了为预防数据块组描述随机增长而保留的数据块之外，根目录的第一个数据块总是位于第一个数据块组数据区的第一个可用数据块中。

### 6. 数据块区

数据块区用于存储文件或目录的实际数据。信息节点区之后即为数据块区。数据块区的大小取决于磁盘或磁盘分区的容量和文件系统的类型，以及创建文件系统时指定的参数。一旦创建文件系统，确定了信息节点区的大小之后，就确定了数据存储的起始位置，从而也确定了数据存储区的大小。

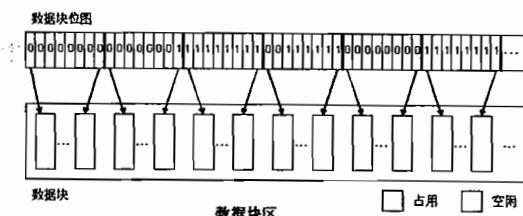


图 16-3 数据块位图与数据块的分配关系



下面以 1.44MB 的软盘为例，说明文件系统的内部组织结构及其物理布局。整个软盘为一个 Linux 分区，使用 1.44MB 的容量创建一个逻辑数据块为 1KB 的 Ext2 文件系统。数据块 0 为引导块，从数据块 1 到数据块 1439 的整个存储空间组成一个数据块组，即整个文件系统只有一个数据块组——数据块组 0（参见图 16-4）。

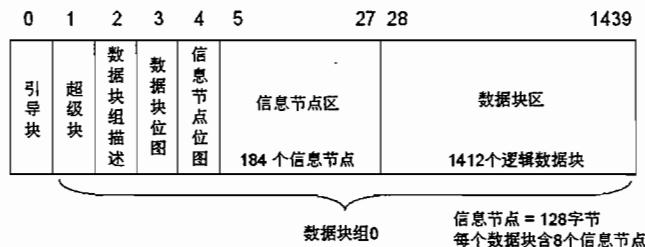


图 16-4 Ext2 文件系统在 1.44MB 软盘上的物理布局

下面是 64MB U 盘文件系统的内部组织结构及其物理布局。我们把整个 U 盘作为一个 Linux 分区，创建了一个 Ext3 文件系统，文件系统中包含 8 个数据块组，如图 16-5 所示。



图 16-5 Ext2 文件系统在 64MB U 盘上的物理布局

引导块、超级块、数据块组描述、数据块位图和信息节点的位置是固定的，而且只占 5 个数据块的存储空间。现在的问题是怎样确定信息节点区和数据块区的划分。实际上，只要确定了信息节点区的大小，就确定了数据存储区的大小。而确定信息节点区的大小可以有两种方法：一是在使用 `mkfs` 命令建立文件系统时指定（参见第 17 章），二是采用系统默认的分配方式。在创建文件系统时，如果没有指定信息节点的数量，系统默认的信息节点数量与数据块数量之比约为 1:4。

## 16.2 超级块

超级块是 Linux 文件系统的核心，其中包含磁盘或磁盘分区的关键数据与总体结构信息。操作系统就是利用超级块，实现文件系统的管理，对文件系统的所有操作，都是通过超级块实现的。为了保证文件系统的可靠性，部分数据块组中也保存一个超级块副本。其中，位于柱面组 0 中的超级块称为主超级块，其他超级块称为次超级块。

在安装 Ext2/Ext3 文件系统时，系统将会把第一个数据块组（即数据块组 0）中的主超级块读入内存。相对而言，磁盘上的超级块信息是静态的，而位于内存中的超级块信息则是动态的。

在 Linux 系统运行期间，系统将会不断地更新位于内存中的超级块，并定时把更新后的超级块

数据写回磁盘中。当卸载文件系统时，系统将会把超级块的最新变化写到磁盘上的每个超级块中。

如果在写入磁盘之前系统因意外事件突然停机，或者未正常关机，将会造成两个超级块中的信息不一致，因而引起文件系统的损坏。这种情况将会导致系统在引导过程中修复文件系统，严重时甚至无法启动系统。

在Ext2/Ext3文件系统中，超级块占用1024个字节，位于第一个与其他部分数据块组的起始位置。下面给出的结构定义详细说明了超级块包含的各种文件系统信息（参见/usr/include/linux/ext3\_fs.h文件），如文件系统的大小及信息节点区的大小等。

```
struct ext3_super_block {
/*00*/ __le32 s_inodes_count; /* 文件系统中信息节点的总数 */
__le32 s_blocks_count; /* 文件系统中逻辑数据块的总数 */
__le32 s_r_blocks_count; /* 为确保文件系统的安全而保留的数据块数量 */
__le32 s_free_blocks_count; /* 文件系统中空闲数据块的总数 */
/*10*/ __le32 s_free_inodes_count; /* 文件系统中空闲信息节点的总数 */
__le32 s_first_data_block; /* 文件系统中第一个可用数据块的位置 */
__le32 s_log_block_size; /* 文件系统逻辑数据块的大小 */
__le32 s_log_frag_size; /* 文件系统逻辑数据片的大小（未实现） */
/*20*/ __le32 s_blocks_per_group; /* 每个数据块组中的逻辑数据块数量 */
__le32 s frags_per_group; /* 每个数据块组中的逻辑数据片数量（未实现） */
__le32 s_inodes_per_group; /* 每个数据块组中的信息节点数量 */
__le32 s_mtime; /* 最近一次安装文件系统的时间 */
/*30*/ __le32 s_wtime; /* 最近一次更新超级块的时间 */
__le16 s_mnt_count; /* 文件系统安装次数 */
__le16 s_max_mnt_count; /* 文件系统最大安装次数 */
__le16 s_magic; /* Magic 签名, Ext2/Ext3 文件系统均为 0xEF53 */
__le16 s_state; /* 文件系统的状态标志 */
__le16 s_errors; /* 当检测到文件系统出现故障时的应对措施 */
__le16 s_minor_rev_level; /* 文件系统的次版本号 */
/*40*/ __le32 s_lastcheck; /* 最近一次检测文件系统的时间 */
__le32 s_checkinterval; /* 检测文件系统的最大时间间隔 */
__le32 s_creator_os; /* 创建文件系统的操作系统，如 Linux 或 FreeBSD 等 */
__le32 s_rev_level; /* 文件系统的主版本号 */
/*50*/ __le16 s_def_resuid; /* 能够使用文件系统保留数据块的默认用户 ID 号 */
__le16 s_def_resgid; /* 能够使用文件系统保留数据块的默认用户组 ID 号 */
__le32 s_first_ino; /* 第一个非保留（可用）的信息节点，默认值为 11 */
__le16 s_inode_size; /* 信息节点结构数据的大小（字节数），默认值为 128 */
__le16 s_block_group_nr; /* 超级块所在的数据块组号 */
__le32 s_feature_compat; /* 兼容的特性集标志 */
/*60*/ __le32 s_feature_incompat; /* 不兼容的特性集标志 */
__le32 s_feature_ro_compat; /* 只读兼容特性集标志 */
/*68*/ __u8 s_uuid[16]; /* 文件系统卷的 128 位 uuid */
/*78*/ char s_volume_name[16]; /* 文件系统卷名 */
/*88*/ char s_last_mounted[64]; /* 最近一次安装文件系统时的安装点（目录） */
/*C8*/ __le32 s_algorithm_usage_bitmap; /* 用于压缩算法 */
__u8 s_prealloc_blocks; /* 能够为文件预分配的最大数据块数 */
__u8 s_prealloc_dir_blocks; /* 能够为目录预分配的最大数据块数 */
__u16 s_reserved_gdt_blocks; /* 为预防数据块组描述的增长而保留的数据块数量 */
/*D0*/ __u8 s_journal_uuid[16]; /* 日志文件系统超级块的 uuid */
/*E0*/ __le32 s_journal_inum; /* 日志文件的信息节点号 */
__le32 s_journal_dev; /* 日志文件的设备号 */
__le32 s_last_orphan; /* 欲删除的信息节点链表的起始地址 */
__le32 s_hash_seed[4]; /* HTREE 散列算法的种子 */
__u8 s_def_hash_version; /* 采用的默认散列算法的版本 */
__u8 s_reserved_char_pad;
__u16 s_reserved_word_pad;
__le32 s_default_mount_opts;
__le32 s_first_meta_bg; /* 第 1 个元数据块所在的数据块组 */
__u32 s_reserved[190]; /* 超级块数据后的填充位置，确保超级块的固定长度 */
};
```

在超级块中，s\_magic 字段用于唯一地标识一个文件系统的类型。需要说明的是，Ext2 与 Ext3



文件系统的 `s_magic` 字段均为 0xEF53，这也从一个侧面说明了 Ext2 与 Ext3 是两个非常兼容的文件系统。`s_blocks_per_group` 与 `s_inodes_per_group` 字段分别表示每个数据块组中含有多少个逻辑数据块与信息节点。`s_log_block_size` 字段表示 Ext2/Ext3 文件系统中逻辑数据块的大小。注意，这个字段中的数字实际上是 1 024 的倍数 ( $1\,024 \times 2^n$ )，有效数值是 0、1 和 2，其意义如下：

- 0 表示逻辑数据块的大小为 1 024 个字节；
- 1 表示逻辑数据块的大小为 2 048 个字节；
- 2 表示逻辑数据块的大小为 4 096 个字节。

注意，`s_inodes_count` 表示整个文件系统中共有多少个信息节点，`s_free_inodes_count` 表示文件系统中尚有多少个空闲的信息节点。`s_blocks_count` 表示整个文件系统中共有多少个逻辑数据块，`s_free_blocks_count` 表示文件系统中尚有多少个空闲的逻辑数据块。

另外，`s_inode_size` 字段表示每个信息节点结构的字节数。`s_state` 字段表示文件系统的状态——文件系统是否完好，是否需要检测与修复，等等。

`s_errors` 字段表示当检测到文件系统出现问题时，Linux 系统内核应当采取的动作：继续正常地运行，或者把文件系统重新安装为只读模式，以免进一步损坏文件系统，还是重新引导系统，以便运行文件系统的检测程序 `fsck`，修复文件系统。

在创建文件系统时，Ext2/Ext3 文件系统通常会为超级用户保留 5% 的存储空间。一旦用户进程无限制地使用存储空间，系统管理员能够利用这些存储空间避免文件系统发生瘫痪。`s_r_blocks_count` 字段的值表示 Ext2/Ext3 文件系统预留的逻辑数据块数量。

`s_mnt_count` 和 `s_max_mnt_count` 两个字段供 Linux 确定是否应当对文件系统进行全面检查。每当安装 Ext2/Ext3 文件系统时，`s_mnt_count` 字段的值都会加 1。当这个字段的值等于 `s_max_mnt_count` 字段的值时，Linux 系统将会显示“maximal mount count reached, running e2fsck is recommended”的提示信息。

通过 `s_prealloc_blocks`（和 `s_prealloc_dir_blocks`）字段，Ext2/Ext3 文件系统提供许多分配存储空间的优化机制，其中包括：

- 尽量把相关的文件分配到同一数据块组；
- 通过数据块位图，尽量把相临的数据块一次预分配给同一文件。

数据块组可用于汇集相关的信息节点与数据块。在为文件分配数据块时，Linux 内核总是尝试使文件的数据块与信息节点位于同一数据块组中，以便之后读取信息节点与其相关的文件数据时，能够减少磁头的移动时间，提高数据的传输效率。

当用户进程需要把数据写入一个文件时，Ext2/Ext3 文件系统还会在分配新的数据块时一次预分配高达 8 个相邻的数据块，从而能够确保把连续的数据块分配给同一个文件。即使在满负载的文件系统中，这种预分配的命中率也会达到约 75%，因而能够保证在写数据时达到较高的效率。这种分配机制也确保在之后读取文件时能够从连续的数据块中快速地获取文件数据。

## 16.3 信息节点

每个数据块组均存在一个由一系列逻辑数据块组成的信息节点区，用于存储文件的信息节点。任何一个信息节点只能位于某一个数据块组中的信息节点区。在 Linux 系统中，每当创建一个文

件（包括目录和特殊文件）时，文件系统都会为其分配一个信息节点，也就是说，每个文件都对应一个信息节点。根据信息节点的数据结构定义（参见/usr/include/linux/ext3\_fs.h），每个信息节点占用 128 个字节。因此，一个 1KB 的逻辑数据块可以容纳 8 个信息节点，而一个 4KB 的逻辑数据块可以容纳 32 个信息节点。

除了文件名与信息节点号位于目录文件之外，信息节点含有文件的所有信息，其中包括文件的属性、文件的实际数据在数据区中的存储位置等，详列如下（参见图 16-6）。

<code>i_mode</code>	文件模式。其中包括文件的类型和访问权限信息
<code>i_uid</code>	文件属主的用户 ID 号
<code>i_size</code>	文件的大小，即文件的字节数
<code>i_atime</code>	文件最近一次的访问时间
<code>i_ctime</code>	文件创建的时间
<code>i_mtime</code>	文件最近一次的修改时间
<code>i_dtime</code>	文件删除时间
<code>i_gid</code>	文件属主所在用户组的用户组 ID 号
<code>i_links_count</code>	文件链接计数
<code>i_blocks</code>	文件分配的数据块计数
<code>i_flags</code>	文件标志
<code>i_block[]</code>	数据块地址指针
<code>i_generation</code>	文件版本号（主要用于 NFS）
<code>i_file_acl</code>	文件访问控制表 ACL（Access Control List）
<code>i_dir_acl</code>	目录访问控制表 ACL
<code>i_faddr</code>	数据片（fragment）地址指针（未实现）
<code>i_i_frag</code>	数据片的数量（未实现）
<code>i_i_fsize</code>	数据片的大小（未实现）
...	

每个信息节点占用 128 个字节

图 16-6 信息节点的结构定义

- 文件的类型：
  - 普通文件；
  - 目录文件；
  - 块特殊文件；
  - 字符特殊文件；
  - 管道（FIFO）文件；
  - 符号链接文件；
  - 套接字。
- 文件读、写和执行的访问权限。
- 文件的硬连接计数（多个文件名共享同一信息节点和数据内容）。
- 文件属主的用户 ID。
- 文件属主所在用户组的用户组 ID。
- 文件的大小（文件的字节数）。
- 一个拥有 15 个数据块地址指针的数组，用于引用分配给文件的数据块。在符号链接文件的情况下，为了节省存储空间，如果引用的路径文件名不超过 60 个字符，将直接存储在数据块地址指针数组中。在字符和块特殊文件的情况下，数据块地址指针数组中第一个直接地址的内容为特殊文件的主次设备号。
- 最后一次访问文件的日期和时间。



- 创建文件的日期和时间。
- 分配给文件的数据块数量。
- 最后一次访问文件的时间。
- 最后一次修改文件的时间。
- 首次创建文件的时间。

在信息节点中，`i_size` 字段记录的是文件或目录的实际大小。要想知道一个信息节点对应的文件或目录究竟占有多少个逻辑数据块，使用 `i_size` 字段的值除以逻辑数据块的大小，即可得出结果。

### 16.3.1 文件的类型与访问权限

根据信息节点结构中 `i_mode` 字段的定义，文件的类型与访问权限字段占用两个字节（16位），其详细定义及其意义如图 16-7 所示。

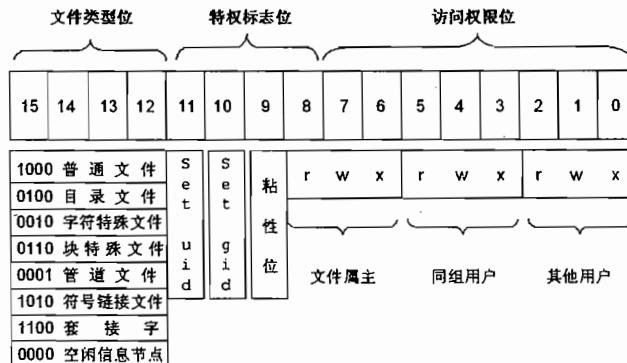


图 16-7 文件的类型和访问权限定义

如果文件类型位为 0000，表示当前的信息节点是空闲的。当创建一个文件时，取决于文件的类型，文件系统将会设置文件的类型位。当使用 `chmod` 命令设置访问权限、`UID`、`GID` 和粘性位时，将会设置相应的二进制数据位（参见第 4 章）。

### 16.3.2 数据块地址数组

按照信息节点结构的定义，信息节点的数据块地址数组含有 15 个地址指针（0~14），占用 60 个字节位置，用于指向存储文件内容的数据块。其中前 12 个地址是直接地址，直接指向含有文件内容的前 12 个逻辑数据块，后 3 个地址分别为一级、二级和三级间接索引。如果文件大于 12 个逻辑数据块能够容纳的数量，则第 13 个地址指向一个间接地址块，其中包含数据块的地址而非文件的实际数据内容；第 14 个地址指向一个二级间接地址块，其中包含一级间接地址块的地址；第 15 个地址指向一个三级间接地址块的地址。图 16-8 给出了从信息节点开始的地址指针、间接地址块与数据块的指向关系。

在间接地址块中，每 4 个字节引用一个数据块号。以 4 096 个字节的逻辑数据块为例，一个间接地址块可以容纳  $1\ 024$  个数据块号。故从理论上讲，Ext2/Ext3 文件系统中的一个文件最大可以达到  $(12 + 1\ 024 + 1\ 024^2 + 1\ 024^3) \times 4\ KB = 4\ 299\ 165\ 744\ KB$ ，约为 4 100 GB 或 4TB。实际上，Ext2/Ext3 文件系统当前支持的最大文件容量仅为 2GB（当前支持的最大文件系统为 4TB）。表 16-1 给出了文件大小与信息节点地址数组的理论计算关系。

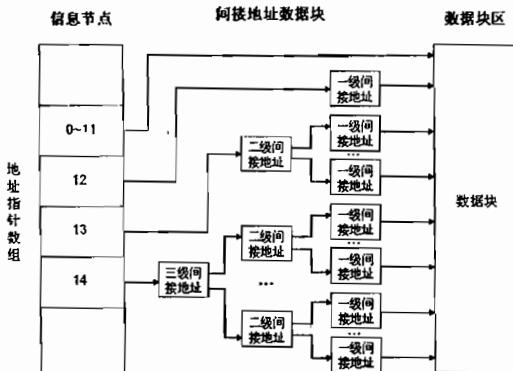


图 16-8 信息节点中的地址指针与间接地址块和数据块的指向关系

表 16-1 文件大小与地址数组的理论计算关系

4 096 字节数据块	直接寻址	一次间接寻址	二次间接寻址	三次间接寻址
数据块数	12	1 024	1 048 576	1 073 741 824
字节数	49 152	4 194 304	4 294 967 296	4 398 046 511 104
合计	49 152	4 243 456	4 299 210 752	4 402 345 721 856

### 16.3.3 符号链接文件

符号链接文件是一种特殊的文件，其中包含的数据是其引用的实际文件的路径名。当遇到一个符号链接文件时，Linux 系统将以文件中的内容作为目的文件，重新执行文件的检索与访问。因为符号链接文件能够引用任何类型的文件，甚至根本就不存在的文件。但与硬链接文件（只占用一个目录项，但不占用信息节点和数据块空间）相比，符号链接文件需要占用额外的信息节点与数据块空间，而且需要执行两次文件名检索，才能访问实际的文件。

为此，Ext2/Ext3 文件系统支持一种快速的符号链接文件。Ext2/Ext3 文件系统不使用任何数据块存储文件的路径名，而是把文件路径名直接存储到符号链接文件信息节点的 15 个地址指针数组中。这种解决方案既节省了数据块存储空间，又避免了再访问数据块，从而能够加快文件的访问速度。

当然，由于地址指针数组的空间有限，最多只能存储 60 (4×15) 个字符，故当符号链接文件引用的目的文件的路径名过长时，暂时只能采用传统的做法。

### 16.3.4 特权标志位

特权标志位的设置使系统在调度执行相应的程序时能够给予特殊的关注。Linux 系统提供下列 3 种特权标志位。

(1) **setuid** (设置用户 ID)：当设置了 setuid 特权标志位时，在程序执行期间，程序调用者的有效用户 ID 将会临时替换为文件属主的用户 ID。采取这种方式，普通用户能够执行通常仅限于文件属主（如超级用户）才有权执行的操作。

例如，当普通用户想要更换自己的密码时，需要使用 passwd 命令，修改/etc/shadow 文件。出于系统安全的考虑，Linux 系统规定，只有超级用户才能访问 shadow 文件；而从用户安全的角度考虑，系统又鼓励用户经常修改自己的密码。



为了解决这个矛盾，Linux 系统把 passwd 命令的 setuid 特权标志位设置为 1，使普通用户在运行 passwd 命令期间，其有效用户 ID 能够临时改为 passwd 文件的属主，即超级用户 root 的用户 ID，因而具有超级用户的访问权限，自然也就能够修改 shadow 文件。如果使用下列 ls 命令显示于 passwd 文件，可以看到文件访问权限字段中存在一个字符“s”。

```
$ ls -l /usr/bin/passwd  
-rwsr-xr-x 1 root root 32988 2008-11-13 14:54 /usr/bin/passwd  
$
```

文件属主执行权限位置中的“s”表示，在运行这个命令期间，普通用户也能拥有超级用户的访问权限。在更换密码之后（即退出 passwd 命令之后），其有效用户 ID 又恢复到用户自己的实际用户 ID。

要设置一个文件的 setuid 特权标志位，可以使用下列命令。

```
# chmod u+s file
```

(2) setgid（设置用户组 ID）：与 setuid 类似，当设置了 setgid 特权标志位时，在程序执行期间，程序调用者的有效用户组 ID 将会临时替换为文件所属用户组的 ID。采取这种方式，只要与文件同属于一个用户组，普通用户也能执行部分特权操作。在下面的例子中，同组用户访问权限的字符“s”表示 wall 文件设置了 setgid 特权标志位。

```
$ ls -l /usr/bin/wall  
-rwxr-sr-x 1 root tty 13808 2008-09-25 21:08 /usr/bin/wall  
$
```

要设置一个文件的 setgid 特权标志位，可以使用下列命令。

```
# chmod g+s file
```

(3) 粘性位（sticky）：粘性位表示共享代码段（text）方式。如果设置了粘性位，则当第一次调用相应的程序时，一经执行，系统就会把程序的代码段副本存储在交换区中，即使程序已经终止执行。这种处理方式使得一旦再次执行程序，便能够减少把程序代码段加载到系统内存的时间。这是因为访问交换区要快于访问文件系统中的程序文件。

如果某个共享目录设置了粘性位，普通用户只能删除该目录下属于自己的文件。对于目录中的其他文件，即使其访问权限允许文件属主和同组用户之外的其他用户写文件，其他用户也无法写入和删除，这将防止用户无意中覆盖或删除共享目录中属于每个用户的文件。注意，在新的 Linux 系统中，粘性标志位仅适用于目录，已不再适用于文件。

如果某个目录设置了粘性位特权标志，使用“ls -ld”命令查询时，可以看到其访问权限字段出现一个字符“t”，示例如下。

```
$ ls -ld /tmp  
drwxrwxrwt 12 root root 4096 2008-12-20 21:030 /tmp  
$
```

要设置一个目录的粘性位特权标志，可以使用下列命令。

```
# chmod t+ dirname
```

## 16.4 信息节点与目录及文件的关系

### 16.4.1 目录文件

目录也是一种文件，是一种存储其他文件名的文件。目录提供了文件的组织方法。在 Ext2/Ext3 文件系统中，每个目录项均包含一个信息节点号、目录项长度、文件名长度、文件类型及文件的

名字。每个目录都是由上述结构的数据组成的，如图 16-9 所示。

在 Ext2/Ext3 文件系统中，除了存储的内容之外，目录文件与普通文件在文件系统中的存储方式是一样的。每当创建一个目录，除了分配信息节点之外，系统也会分配一个数据块，用于存储目录中包含的文件名与信息节点号。取决于文件系统逻辑数据块的大小，每个目录的初始容量为 1KB、2KB 或 4KB。

目录项（即文件名）的长度是可变的，但每个目录项的字节数必须是 4 的倍数，不足者必须在后面附加适当数量的填充字符。通过使用变长目录项的组织形式，既可支持较长的文件名，又不至于浪费目录文件的存储空间。例如，如果目录中包含 3 个文件 filename、long\_file\_name 和 fn，以及一个目录 tools，则其目录项分别如下所示。

ino1	16	08	01	filename
ino2	24	14	01	long_file_name\0\0
ino3	12	02	01	fn\0\0
ino4	16	05	02	tools\0\0\0

注意，第一个字段应为实际的信息节点号，随后的两个长度字段为十六进制数值，文件类型字段的 01 表示普通文件，02 表示目录文件。下面是从上述目录文件中导出来的实际数据，读者可以对照。（利用 tune2fs 命令能够确定逻辑数据块的大小，利用 debugfs 命令可以确定目录文件所在的数据块号，这里的 33 是 3.5 英寸软盘/dev/fd0 文件系统根目录的数据块号。）

```
# dd if=/dev/fd0 bs=1024 skip=33 count=1 | hexdump -C
1+0 records in
1+0 records out
1024 bytes (1.0 kB) copied. 0.000266809 s. 38.8 MB/s
000000 02 00 00 00 0c 00 01 02 2e 00 00 00 02 00 00 00 |.....
000010 0c 00 02 02 2e 2e 00 00 0b 00 00 00 10 00 08 01 |.....
000020 66 69 6c 65 6e 61 6d 65 0c 00 00 00 18 00 0e 01 |filename....|
000030 6c 6f 6e 67 5f 66 69 6c 65 5f 6e 61 6d 65 00 00 |long_file_name...|
000040 0d 00 00 00 0c 00 02 01 66 6e 00 00 0e 00 00 00 |....fn....|
000050 b4 03 05 02 74 6f 6c 73 00 00 00 00 00 00 00 00 |....tools....|
000060 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |.....|
*
000400
#
```

在上述的输出数据中，前两项分别是当前目录“.”和父目录“..”，各占 12（8+4）个字节。后 4 项分别对应 filename、long\_file\_name、fn 及 tools 这 4 个文件与目录。以 filename 文件为例，其中的“0b 00 00 00”表示文件的信息节点号为 11，“10 00”表示记录项的长度为 16 个字节，“08”表示文件名的长度为 8 个字节，“01”表示 filename 是一个普通文件。

## 16.4.2 目录、文件和信息节点三者之间的关系

在文件系统中，每个文件（包括目录和特殊文件）都有一个与之对应的信息节点，信息节点中的信息确定了文件的属性（文件的属主、文件的大小及文件的访问权限等）和文件数据所在数据块的地址。实际上，除了文件名与信息节点号之外，文件的所有信息均可从信息节点中找到。文件名与信息节点号存放在所属目录的目录文件中，其关系如图 16-10 所示（注意，在 Ext2/Ext3 文件系统中，lost+found 的实际信息节点号为 11，/etc、/home、/var 和/usr 等目录的信息节点号比

较大，为简化起见，这里只是借用了第 3~7 文件系统保留的信息节点号)。

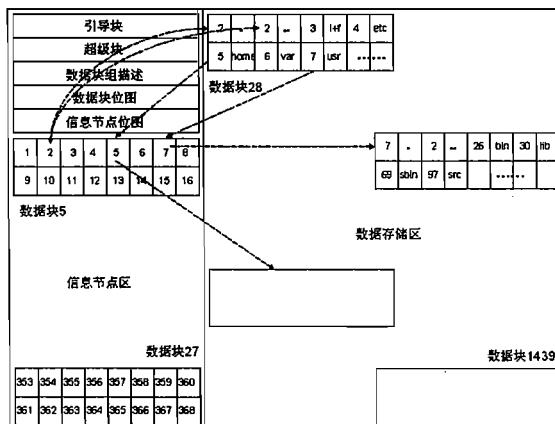


图 16-10 信息节点与目录或文件之间的关系

每个 Ext2/Ext3 文件系统都有一个根目录。这里所谓的根目录是相对于一个文件系统或硬盘分区而言的，并不一定是整个 Linux 操作系统中的根目录。根目录位于一个固定的信息节点中，即文件系统中的 2 号信息节点。注意，同传统的 UNIX System V 文件系统一样，Ext2/Ext3 文件系统中信息节点也是从 1 开始编号的，而逻辑数据块则是从 0 开始编号的。

当访问一个文件时，通过目录检索可以检索到匹配的文件名，从而能够获取相应的信息节点。之后，可以采用下列公式，求取信息节点所在的数据块组及其偏移位置，从而能够访问信息节点中的数据。

$$\text{数据块组号} = \lceil (\text{ino} - 1) / \text{s\_inodes\_per\_group} \rceil$$

$$\text{信息节点在信息节点区中的偏移值} = (\text{ino} - 1) \% \text{s\_inodes\_per\_group}$$

在上述计算公式中，`ino` 表示信息节点号。`s_inodes_per_group` 是文件系统超级块中的一个字段，表示每个数据块组中拥有多少个信息节点。信息节点号除以 `s_inodes_per_group`，运算结果的整数部分表示信息节点位于哪一个数据块组，余数为信息节点在信息节点区中的偏移值，即信息节点位于相应数据块组信息节点区中的位置。

经过上述计算即可找到数据块组，根据其中的数据块组描述，又可找出信息节点的起始数据块号地址，从而找出信息节点区，乃至找出信息节点。之后，就可以使用或读取信息节点对应的文件数据了。

假定获取的信息节点号为 100，每个逻辑数据块为 1KB，因而含有 8 (1 024/128) 个信息节点，信息节点区的起始数据块号地址为 5，则由此可以计算出信息节点所在的数据块号：

$$\text{信息节点数据块号} = \lceil (100 - 1) / 8 \rceil + 5 = 12 + 5 = 17$$

确定了信息节点所在的数据块之后，还需要根据下列公式计算出信息节点起始字符位置在数据块中的偏移值（其中，`s_inode_size` 是信息节点的长度，`s_log_block_size / s_inode_size` 是每个数据块中含有的信息节点数量）：

$$\text{偏移值 (字节位置)} = ((\text{ino} - 1) \bmod (\text{s\_log\_block\_size} / \text{s\_inode\_size})) \times \text{s\_inode\_size}$$

按照上述公式，可以计算出信息节点起始字符位置在数据块中的偏移值（假定信息节点的长度为 128）：

$$\text{偏移值 (字节位置)} = ((100 - 1) \bmod 8) \times 128 = 3 \times 128 = 384$$

找到信息节点之后，根据其地址数组中的数据块号，即可确定文件实际数据所在的数据块位置。

# 第17章

## 文件系统管理

本章主要介绍文件系统的管理，包括文件系统的创建、安装、卸载、检测与修复。首先将主要以 Ext2/Ext3 文件系统为例，详细说明怎样创建和调整文件系统，怎样安装和卸载文件系统；然后讨论怎样检测与修复因断电等原因而受损的文件系统，保证系统的正常启动与运行；最后介绍用于文件系统调试的工具。其内容主要包括：

- 划分磁盘分区；
- 创建文件系统；
- 调整文件系统；
- 安装与卸载文件系统；
- 检测与修复文件系统；
- 调试文件系统；
- 其他文件系统维护工具。





当需要增加新的磁盘或磁盘阵列时，首先需要使系统能够识别新增加的设备，并把设备配置到系统中。具体步骤通常涉及到安装硬件，连接电缆，安装驱动程序（如采用主机适配器连接磁盘阵列）等。在系统的引导和启动过程中，系统会自动识别并生成相应的设备文件。

在用户能够以文件系统形式使用新的磁盘等存储设备之前，通常需要执行下列步骤。

(1) 首先利用 fdisk 命令（类似的命令还有 cfdisk、sfdisk、parted 及 partman 等）对磁盘等存储设备进行分区。分区时，可以把整个磁盘划分为一个 Linux 分区。如果磁盘容量较大，或者为了支持不同的应用，也可以把磁盘划分为若干分区，以便在每个分区中创建一个单独的文件系统，然后再把每个文件系统安装到 Linux 系统中。

(2) 利用 mkfs 或 mke2fs 等命令，在指定的磁盘分区中创建文件系统。

(3) 利用 e2label 命令，为新建的文件系统增加卷标（这一步是可选的）。

(4) 利用 tune2fs 命令，对文件系统的参数进行必要的调整（仅当需要时）；

(5) 利用 mount 等命令把新建的文件系统安装到指定的目录位置。

## 17.1 划分磁盘分区

每个物理磁盘可以划分为一个或多个逻辑磁盘，逻辑磁盘通常称作磁盘分区。磁盘分区信息位于磁盘扇区 0 的分区表中。在磁盘等存储设备上创建文件系统之前，首先必须对物理磁盘进行分区。在 Linux 系统中，一个磁盘分区既可用作文件系统，也可用作交换空间。

在 Ubuntu Linux 系统中，所有的磁盘设备（包括 USB 移动硬盘）均采用 /dev/sdDN 的形式统一命名。其中，D 可以是字母 a、b、……分别表示第一个磁盘、第二个磁盘，依此类推。数字 N 从 1 开始编号，表示磁盘设备中的磁盘分区。例如，/dev/sdal 是系统配置的第一个磁盘设备中的第一个磁盘分区。

通常，每个 Linux 系统至少需要两个磁盘分区，分别用做 “/” 文件系统和交换分区。也可以创建第三个磁盘分区，用作 /boot 文件系统，存储系统内核映像，以及引导系统时需要用到的其他辅助文件。在一个 Intel x86 系列及其兼容的计算机系统中，用于引导系统的 BIOS 通常只能访问磁盘的前 1 024 个柱面。因此，划分磁盘分区时需要确保 BIOS 能够访问到 /boot 分区。如果系统配置的磁盘容量较大，且出于安全，易于管理、备份或检测等原因，也可以把磁盘设备划分为多个分区，创建多个文件系统。

fdisk 命令可用于划分磁盘设备，显示磁盘分区信息。在划分磁盘分区时，通常均采用交互方式，运行 fdisk 命令，其语法格式简写如下。

```
fdisk [-u] [-b sectorsize] [-C cyls] [-H heads] [-S sects] device  
fdisk -l [-u] [device]  
fdisk -s partition
```

其中，“-l” 选项用于显示指定磁盘设备的分区表。如果未给定磁盘设备，则输出 /etc/fstab 文件中列举的，以及系统检测到的每个存储设备的分区表。“-u” 选项表示以扇区（而不是以柱面）为单位列出每个设备分区的起始数据块位置。“-s” 选项表示以数据块为单位显示指定设备分区的容量。

下面的例子说明了怎样利用 fdisk 命令，把一个 64MB 的 U 盘划分为两个 Linux 分区，以便创建两个文件系统（由于 U 盘容量通常比较小，没有必要划分为多个分区。实际上，也可以不分区，直接在整个 U 盘上创建文件系统）。

```

$ sudo fdisk /dev/sdb
.....
Command (m for help): m
Command action
.....
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
t  change a partition's system id
.....
w  write table to disk and exit
.....
Command (m for help): n
Command action
  e  extended
  p  primary partition (1-4)
p
Partition number (1-4): 1
First cylinder (1-1015, default 1): <Enter>
Using default value 1
Last cylinder or +size or +sizeM or +sizeK (1-1015, default 1015): 508

Command (m for help): n
Command action
  e  extended
  p  primary partition (1-4)
p
Partition number (1-4): 2
First cylinder (509-1015, default 509): <Enter>
Using default value 509
Last cylinder or +size or +sizeM or +sizeK (509-1015, default 1015): <Enter>
Using default value 1015

Command (m for help): p

Disk /dev/sdb: 65 MB, 65536000 bytes
3 heads, 42 sectors/track, 1015 cylinders
Units = cylinders of 126 * 512 = 64512 bytes
Disk identifier: 0x00000000

      Device Boot      Start        End    Blocks   Id  System
/dev/sdb1            1       508     31983   83  Linux
/dev/sdb2          509      1015     31941   83  Linux
Command (m for help): w
The partition table has been altered!
Calling ioctl() to re-read partition table.
Syncing disks.
$
```

在上述操作过程中，对磁盘分区表的任何改变，都是在内存中进行的。为了使磁盘分区信息生效，在退出 fdisk 交互命令之前，必须使用 w (write) 子命令写入磁盘。如果需要，还可以使用 t (type) 子命令设置磁盘分区的类型。如果不知道磁盘分区的类型代码，可以使用 l (list) 子命令显示磁盘分区类型代码表。

如果一个磁盘已经分区，并分别创建了不同的文件系统，为了获取磁盘的分区信息，可以使用下列 fdisk 命令。

```

$ sudo fdisk -l /dev/sda
Disk /dev/sda: 30.0 GB, 30005821440 bytes
240 heads, 63 sectors/track, 3876 cylinders
Units = cylinders of 15120 * 512 = 7741440 bytes
      Device Boot      Start        End    Blocks   Id  System
```



```
/dev/sda1      *     1     827    6252088+   7  HPFS/NTFS
/dev/sda2          828    3768   22233960   f  W95 Ext'd (LBA)
/dev/sda3          3769    3876    816480   1c  Hidden W95 FAT32 (LBA)
/dev/sda5          2183    2209   204088+   83  Linux
/dev/sda6          2210    3768   11786008+   8e  Linux LVM
/dev/sda7          828    2083    9495297   83  Linux
/dev/sda8          2084    2182    748408+   82  Linux swap / Solaris
Partition table entries are not in disk order
$
```

从上述输出信息中可以得知，磁盘分区 1 的数据块总数为 6 252 088，其文件系统为 NTFS。磁盘分区 5 和 6 的数据块总数分别为 204 088 和 11 786 008，其中前者为 Fedora Linux 系统的 boot 分区，后者为 Fedora Linux 系统的逻辑磁盘卷。磁盘分区 7 和 8 的数据块总数分别为 9 495 297 和 748 408，其中前者为 Ubuntu Linux 系统的“/”文件系统，后者为交换分区。

## 17.2 创建文件系统

在安装操作系统时，作为其中的一个步骤，安装程序将会引导用户提供必要的数据或做出相应地选择，然后自动地划分磁盘分区，创建文件系统（详见第 1 章）。在日常的应用过程中，仅当需要增加新盘或使用移动硬盘，改变现有的磁盘分区结构时，才有可能需要自己手工创建文件系统。

在磁盘等存储设备分区之后，即可着手创建文件系统。创建文件系统时，主要有两种方法：一是使用最基本的、通用的 mkfs 命令，在选定的磁盘分区中创建指定的文件系统；二是利用各种特定的工具，如 mke2fs、mkfs.ext2 或 mkfs.vfat 等，在选定的磁盘分区上直接创建特定类型的文件系统。

### 17.2.1 mkfs 或 mke2fs 命令介绍

创建文件系统时，首先想到的、最基本的工具通常是 mkfs 命令。在 Linux 系统中，mkfs 命令实际上只是各种文件系统创建程序的一个总控程序，根据指定的文件系统类型，mkfs 将会调用特定文件系统的创建程序“mkfs.fs”，其中 fs 可以是 ext2、ext3 或 vfat 等。因此，在创建一个具体的文件系统时，可以使用特定的文件系统创建命令。例如，为了创建一个 Ext2/Ext3 文件系统，可以直接使用 mkfs.ext2 或 mkfs.ext3 等命令，也可以使用等价的 mke2fs 命令。对于 Ext2/Ext3 文件系统而言，mke2fs 命令是最基本的工具。

用于创建 Ext2/Ext3 文件系统的 mkfs 与 mke2fs 命令的语法格式简写如下。

```
mkfs [-V] [-t fstype] [fs-options] device [blocks]
mke2fs [-b block-size] [-g blocks-per-group] [-i bytes-per-inode]
        [-I inode-size] [-j] [-J journal-options] [-L volume-label]
        [-m reserved-blocks-percentage] [-n] [-N number-of-inodes]
        [-F] [-S] [-t fstype] [-T usage-type] [-Vv] device [blocks]
```

表 17-1 给出了 mkfs 和 mke2fs 命令的选项及简单说明。

表 17-1

mkfs/mke2fs 命令选项及说明

选 项	解 释
-t fstype	指定创建的文件系统类型，如 ext2 或 ext3 等。如果未指定这个选项，则默认的文件系统类型为 ext3
-b block-size	指定文件系统逻辑数据块的大小（字节数）。有效的逻辑数据块是 1 024、2 048 或 4 096 个字节。如果未指定这个选项，mke2fs 将根据整个文件系统的大小和文件系统的预期用途（参见“-T”选项）综合考虑而定

续表

选 项	解 释
<b>-F</b>	强制运行 <code>mke2fs</code> , 创建一个新的文件系统, 即使指定的分区并非块设备文件, 或者指定的设备已经作为文件系统安装到系统中
<b>-g blocks-per-group</b>	指定一个数据块组包含的逻辑数据块数量
<b>-i bytes-per-inode</b>	指定字节数与信息节点之比率, 即每个文件平均占用多少个字节数。这个数值反映了一个文件系统中每个文件的平均大小。数值越大, 文件系统的信息节点数越少。这个数值通常不应小于文件系统逻辑数据块的大小。注意, 文件系统一经创建, 信息节点的数量是不能随意变动的, 故应仔细地确定这个数值
<b>-l inode-size</b>	指定信息节点的大小(字节数)。对于 Ext2/Ext3 文件系统而言, 这个选项的默认值为 128 个字节
<b>-j</b>	创建一个具有 Ext3 日志功能的文件系统
<b>-J journal-options</b>	在创建 Ext2 文件系统时, 使用指定的日志选项创建 Ext3 日志。多个日志选项中间可以加逗号“,”分隔符。实际上, 创建一个具有日志功能的 Ext2 文件系统, 不如直接创建一个标准的 Ext3 文件系统
<b>-L volume-label</b>	设置文件系统的卷标
<b>-m reserved-blocks-percentage</b>	指定创建文件系统时需要为超级用户或系统服务进程保留的数据块占整个文件系统容量的百分比。当达到这个数值时, 只有超级用户才能继续请求分配空间。采用这一措施的目的是防止用户进程过多地占用存储空间而导致文件系统瘫痪, 使超级用户能够在文件系统处于满负荷的情况下维护文件系统, 或者系统服务进程能够继续运行。这个选项的默认值是 5%
<b>-n</b>	使用这个选项的目的只是为了让 <code>mke2fs</code> 显示, 一旦需要创建文件系统时, 最终将会创建一个什么样的文件系统, 实际上并不会真正地创建文件系统。利用这个选项可以确定一个特定文件系统中的备份超级块的存储位置。当然, 还要保证除“-n”选项之外, 其他所有的选项与初次使用的命令选项是完全一致的
<b>-N number-of-inodes</b>	这个选项允许用户在创建文件系统时直接指定信息节点的数量, 而不是采用 <code>mke2fs</code> 命令计算的信息节点默认值
<b>-S</b>	仅仅写入超级块和数据块组描述信息, 而不是真正地创建文件系统。当所有的超级块和超级块备份均遭损坏时, 这个选项是非常有用的。如果所有的文件系统修复方法都不奏效的话, 这是恢复文件系统的最后一招。使用这个选项时, <code>mke2fs</code> 将会重新初始化超级块和数据块组描述部分, 但不会触及信息节点区以及数据块和信息节点位图。之后, 应立即运行 <code>e2fsck</code> 程序。注意, 这种方法并不能保证能够挽救所有的数据
<b>-T usage-type</b>	指定文件系统的预期用途, 以便 <code>mke2fs</code> 能够有针对性地选择优化的文件系统参数。Ubuntu Linux 支持的文件系统类型:
	<input type="checkbox"/> news 每个 4KB 的数据块对应一个信息节点; <input type="checkbox"/> largefile 每个 1MB 的数据块对应一个信息节点; <input type="checkbox"/> largefile4 每个 4MB 的数据块对应一个信息节点
<b>-V</b>	用于显示 <code>mkfs</code> 或 <code>mke2fs</code> 命令的版本号
<b>-v</b>	在运行过程中输出详细的处理信息
<b>device</b>	设备文件名, 表示准备创建文件系统的磁盘分区, 如 <code>/dev/sdb1</code> 等
<b>blocks</b>	指定文件系统中数据块的总和。注意, 这个值应等于准备创建文件系统的整个磁盘分区的大小。如果忽略, 可由 <code>mkfs</code> 或 <code>mke2fs</code> 命令根据自动计算的结果确定整个文件系统的容量

利用“-n”选项并不实际创建文件系统的特性, 可据以获取创建文件系统时 `mkfs` 命令使用的默认参数值, 了解备份超级块的位置, 以便当文件系统损坏严重时, 可以利用备份超级块修复文件系统(当然, 这样做的前提条件是除“-n”选项之外, 其他所有的选项与初次使用 `mkfs` 命令时提供的选项必须是完全一致的), 示例如下。

```
$ sudo mkfs -t ext2 -n /dev/sdb
mke2fs 1.40.8 (13-Mar-2008)
/dev/sdb is entire device, not just one partition!
Proceed anyway? (y,n) y
Filesystem label=
OS type: Linux
```



```
Block size=1024 (log=0)
Fragment size=1024 (log=0)
16000 inodes, 64000 blocks
3200 blocks (5.00%) reserved for the super user
First data block=1
Maximum filesystem blocks=65536000
8 block groups
8192 blocks per group, 8192 fragments per group
2000 inodes per group
Superblock backups stored on blocks:
          8193, 24577, 40961, 57345
$
```

由上述命令的输出信息可以看出，除了位于数据块 1 的主超级块之外，其他备份超级块分别位于 8 193、24 577、40 961 和 57 345 这 4 个数据块中。当文件系统的主超级块严重损坏时，可以利用任何一个备份超级块修复文件系统。

## 17.2.2 创建 Ext2/Ext3 文件系统

下面的例子说明了怎样利用 mkfs 命令，在未分区的 64MB U 盘上创建一个 Ext3 类型的文件系统。其中，由于没有明确指定，文件系统逻辑数据块的大小采用默认值 1 024 字节，整个文件系统的最大容量为 64 000 个数据块。此外，文件系统还保留 3 200 个数据块（5%）的存储空间，供超级用户在紧急情况下使用。文件系统分为 8 个数据块组，每个数据块组拥有 8 192 个数据块，2 000 个数据块用作信息节点区，整个文件系统因而拥有 16 000 个信息节点，可以创建或存储 16 000 个文件。除了位于数据块 1 的主超级块之外，其他 4 个备份的超级块分别位于 8 193、24 577、40 961 和 57 345 号数据块中。执行 mkfs 命令之后，创建的文件系统具有一个根目录和一个 lost+found 子目录。

```
$ sudo mkfs -t ext3 /dev/sdb
mke2fs 1.40.8 (13-Mar-2008)
/dev/sdb is entire device, not just one partition!
Proceed anyway? (y,n) y
Filesystem label=
OS type: Linux
Block size=1024 (log=0)
Fragment size=1024 (log=0)
16000 inodes, 64000 blocks
3200 blocks (5.00%) reserved for the super user
First data block=1
Maximum filesystem blocks=65536000
8 block groups
8192 blocks per group, 8192 fragments per group
2000 inodes per group
Superblock backups stored on blocks:
          8193, 24577, 40961, 57345

Writing inode tables: done
Creating journal (4096 blocks): done
Writing superblocks and filesystem accounting information: done

This filesystem will be automatically checked every 33 mounts or
180 days, whichever comes first. Use tune2fs -c or -i to override.
$
```

最后还要说明的是，在使用 mkfs 命令创建文件系统时，如果未利用“-L”选项指定文件系统的卷标，可在事后采用 e2label 命令设定文件系统的卷标。e2label 命令的主要功能就是显示和命名未安装文件系统的卷标，其语法格式如下。

```
e2label device [new-label]
```

其中，“device”是磁盘分区的设备文件名，如/dev/sda6 等，“new-label”表示文件系统的卷标。如果未指定“new-label”，e2label 命令将会输出文件系统当前的卷标。注意，在指定文件系统的卷标时，不能超过 16 个字符，示例如下。

```
$ sudo e2label /dev/sdb
$ sudo e2label /dev/sdb data01
$ sudo e2label /dev/sdb
data01
$
```

## 17.3 调整文件系统

一经创建，文件系统的基本组织结构原则上是不能随意变动的。如果需要，可以利用 tune2fs 命令，简单调整 Ext2/Ext3 文件系统的部分可调参数，如在 Ext2 文件系统中增加日志功能，增加文件系统的卷标，增加索引，以提高大型目录的文件检索速度等；也可以显示文件系统超级块中的重要信息。tune2fs 命令的语法格式简写如下。

```
tune2fs [-l] [-c max-mount-count] [-C mount-count] [-e error-behavior] [-f]
[-u user] [-g group] [-i check-interval] [-j] [-J journal-options]
[-L volume-name] [-m reserved-blks-percentage] [-M last-mounted-dir]
[-O [^]features] [-r reserved-blks-count] [-T time-last-checked] device
```

表 17-2 给出了 tune2fs 命令的部分选项及其简单说明。

表 17-2 tune2fs 命令的部分选项

选 项	简 单 说 明
-c max-mount-count	调整文件系统的最大安装次数。超过此限，则强制系统采用 e2fsck 命令检测文件系统。如果把最大安装次数设定为 0 或 -1，e2fsck 命令与 Linux 系统内核将会忽略文件系统的安装次数。在使用具有日志功能的文件系统时，合理地设置文件系统的最大安装次数可以避免 Linux 系统同时检测所有的文件系统
-C mount-count	设置文件系统已经安装的次数。如果设置的安装次数大于最大安装次数（“-c”选项），在下一次引导系统时，Linux 系统将会使用 e2fsck 命令检测文件系统
-e error-behavior	指定当检测到文件系统出现错误时 Linux 内核应当采取的处理动作。文件系统的错误有可能导致 Linux 系统采用下列 3 种处理方式之一： <input type="checkbox"/> continue 继续正常运行； <input type="checkbox"/> remount-ro 以只读方式重新安装文件系统； <input type="checkbox"/> panic 引起系统内核按系统瘫痪处理。 在任何情况下，文件系统的错误都会促使 Linux 内核在下一次引导系统时利用 e2fsck 命令检测文件系统
-f	强制运行 tune2fs 命令，即使操作过程中出现问题
-g group	设置能够使用文件系统保留数据块的用户组成员。给定的用户组参数既可以是一个数值形式的用户组 ID，也可以是用户组名。不管采用哪一种指定方式，系统最终都会以数值形式把用户组 ID 写入超级块中
-i check-interval [d m w]	设定检测文件系统的最大时间间隔。如果指定的数值后面没有后缀，或后缀为 d，则表示天数。后缀为 m 表示月数，后缀为 w 表示周数。如果指定的数值为 0，表示禁止使用与时间有关的文件系统检测。Linux 系统强烈建议用户启用“-c”或“-i”选项表示的文件系统检测功能，强制系统周期地采用 e2fsck 命令执行全面的文件系统检测。否则，有可能因坏块或内核漏洞等原因导致文件系统损害而长期毫无觉察，直至最终引起数据的丢失或文件系统的严重受损
-j	在 Ext2 文件系统中增加 Ext3 日志功能。如果未指定“-J”选项，采用默认的日志参数创建适当大小的日志



续表

选 项	简 单 说 明
<b>-J journal-options</b>	替换默认的 Ext3 日志选项 size 与 device。日志选项采用等号 “=” 形式赋值。利用 “-J” 选项能够同时指定多个日志选项，选项之间需加逗号 “,” 分隔符。Ext3 文件系统支持下列日志选项。 <input type="checkbox"/> <b>size=journal-size</b> 在文件系统中创建指定大小（单位 MB）的日志。日志至少必须具有 1 024 个逻辑数据块。也就是说，如果逻辑数据块为 1KB，则日志至少必须设为 1MB；如果逻辑数据块为 4KB，则日志至少必须设为 4MB。但最大不能超过 102 400 个逻辑数据块。 <input type="checkbox"/> <b>device=external-journal</b> 利用外部日志设备实现文件系统的日志功能（注意，外部日志设备与当前的文件系统必须具有相同大小的逻辑数据块）。在设置这个日志选项之前，必须首先使用下列命令设定外部日志设备： <pre>mke2fs -O journal_dev external-journal</pre>
<b>-l</b>	输出文件系统超级块中的数据内容
<b>-L volume-label</b>	设置文件系统的卷标。Ext2/Ext3 文件系统的卷标可以长达 16 个字符，如果给定的卷标超过 16 个字符，tune2fs 命令将会截掉超常部分，并输出一个警告信息。在 mount 和 fsck 命令以及 /etc/fstab 文件中，可以采用“LABEL=卷标”的形式，以卷标替代设备名
<b>-m reserved-blks-percentage</b>	设置文件系统保留数据块的百分比
<b>-M last-mounted-dir</b>	设置文件系统最后一次安装的目录
<b>-O [+]features</b>	设置或清除文件系统的功能特性选项。如果在某个特性选项前加一个上箭头字符 “^”，表示清除相应功能特性选项；如果特性选项前存在一个加号字符 “+” 或没有任何前缀，则表示增加相应功能特性。多个选项之间需加逗号 “,” 分隔符。tune2fs 命令能够设置或清除的文件系统功能特性选项如下： <input type="checkbox"/> <b>dir_index</b> 采用散列 B 树提高大型目录的文件检索速度； <input type="checkbox"/> <b>filetype</b> 在目录项中存储文件类型信息； <input type="checkbox"/> <b>has_journal</b> 使用日志确保文件系统的一致性，设置这个文件系统功能特性等价于使用 “-j” 选项； <input type="checkbox"/> <b>sparse_super</b> 限制备份超级块的数量，以节省大型文件系统的空间。 在设置或清除文件系统的 sparse_super 与 filetype 功能特性之后，必须运行 e2fsck，使文件系统保持一致性。如果需要，tune2fs 将会给出一个消息，要求系统管理员运行 e2fsck 命令。在设置文件系统的 dir_index 功能特性之后，可以运行 “e2fsck -D” 命令，把现有的目录转换成散列 B 树格式的目录
<b>-r reserved-blks-count</b>	设置文件系统保留的数据块数量
<b>-T time-last-checked</b>	设置最后一次使用 e2fsck 命令检测文件系统的时间。时间的表示形式采用国际日期格式，即 YYYYMMDD[HHMM]SS。也可以使用关键字 now，表示把系统的当前时间作为最后一次检测文件系统的时间。
<b>-u user</b>	设置能够使用文件系统保留数据块的用户。指定用户时，既可以使用用户 ID，也可以使用用户名，但不管采用哪一种指定方式，系统最终都会以数值形式把用户 ID 写入超级块中

下面是一个例子，说明如何使用 tune2fs 命令建立目录索引，提高文件系统检索大型目录的速度。

```
$ sudo tune2fs -O dir_index /dev/sdb2
tune2fs 1.40.8 (13-Mar-2008)
$
```

要增加 Ext2 文件系统的日志功能，可以使用下列 tune2fs 命令。

```
$ sudo tune2fs -j /dev/sdb
tune2fs 1.40.8 (13-Mar-2008)
$ sudo e2fsck -D /dev/sdb
e2fsck 1.40.8 (13-Mar-2008)
data01: clean, 11/16000 files, 7400/64000 blocks
$
```

要查询 Ext2/Ext3 文件系统的原有信息，可以使用下列 tune2fs 命令。

```
$ sudo tune2fs -l /dev/sdb
tune2fs 1.40.8 (13-Mar-2008)
Filesystem volume name: data01
Last mounted on: <not available>
Filesystem UUID: fbf4a89b-2df2-46aa-b391-c979fb2d0290
Filesystem magic number: 0xEF53
```

```

Filesystem state:          clean
Errors behavior:          Continue
Filesystem OS type:        Linux
    count:                16000
    bck count:             64000
    i block count:         3200
    blocks:                56600
    bodes:                15989
    block:                  1
    size:                 1024
    GDT blocks:            249
    er group:              8192
    s per group:           8192
    er group:              2000
    blocks per group:      250
    em created:            Sun Oct 12 15:29:45 2008
    nt time:               n/a
    e time:                Sun Oct 12 15:29:46 2008
    nt:                     0
    count count:           24
    ked:                   Sun Oct 12 15:29:45 2008
    rerval:                15552000 (6 months)
    k after:               Fri Apr 10 15:29:45 2008
    blocks uid:             0 (user root)
    blocks gid:             0 (group root)
    ie:                     11
    :::
    mode:                  8

```

## 安装与卸载文件系统

怎样在 Linux 系统中安装和卸载文件系统。

### 文件系统概述

为了文件系统之后，需要把新建的文件系统安装到 Linux 文件系统目录层次结构上，然后才能使用新建的文件系统。对于任何文件系统，包括 iso9660 格式的 CD/DVD、T32 类型的 DOS 文件系统以及 NTFS 类型的 Windows 文件系统，都需要先安装（原始存储介质除外）。

对于 CD/DVD、移动硬盘和 U 盘等文件系统，Ubuntu Linux 系统提供文件系统的自动安装功能。当把上述存储介质插入系统时，Linux 系统将会尝试安装相应的文件系统。必要时，也可以在命令行界面中，使用 mount 和 umount 命令手工安装与卸载文件系统。

mount 命令用于安装文件系统或远程共享资源，umount 命令用于卸载文件系统或远程共享资源。除非特别设定，在使用 mount 命令安装文件系统时，必须具有超级用户的身份，或者使用 sudo 命令（必要时还需要使用 mkdir 命令，事先创建适当的安装点）。在使用 mount/umount 命令时通常应注意下列事项。

- mount 命令不能安装一个已经受损且需要以读写方式安装的文件系统。如果在运行 mount 命令时出现错误信息，通常需要使用 fsck 命令检测与修复文件系统。
- umount 命令不能卸载一个当前正在使用的文件系统，例如：



- 用户正在访问文件系统中的文件或目录；
- 某个程序打开了文件系统中文件；
- 文件系统正在被共享。

- 使用 `remount` 选项，可以把一个以只读方式安装的文件系统重新安装为读写方式的文件系统，但不能把以读写方式安装的文件系统重新安装为只读方式的文件系统。
- 对于移动硬盘一类的文件系统，在用完卸下之前，不要忘记使用 `umount` 命令等方式卸载文件系统。

### 17.4.2 mount 命令

安装文件系统的基本命令是 `mount`，其语法格式简写如下。

```
mount [-lhV]
mount -a [-rw] [-t fstype] [-o options]
mount [-rw] [-o options] device | directory
mount [-rw] [-t fstype] [-o options] device directory
```

其中，“`-a`”选项表示安装 `fstab` 文件中列举的所有文件系统。当系统进入多用户等运行模式时，系统将会自动地运行“`mount -a`”命令。“`-r`”选项表示以只读方式安装给定的文件系统。“`-w`”选项表示以读写方式（默认）安装指定的文件系统。“`-t`”选项用于指定文件系统的类型，如 `ext2`、`ext3`、`iso9660`、`vfat` 及 `ntfs` 等。“`device`”表示包含文件系统的设备文件名，如 `/dev/sda5`。“`directory`”表示安装点。

上述 `mount` 命令可用于不同场合。实际上，由于 `mount` 命令的主要功能是把指定设备中的文件系统安装到指定的目录中，故最基本、也是最常用的安装命令的标准语法格式如下。

```
mount -t fstype device directory
```

上述安装命令相当于告诉 Linux 系统内核，把指定的设备（`device`）、指定类型（`fstype`）的文件系统安装到指定的目录（`directory`）中。

如果省略了“`-t fstype`”选项，即没有指定文件系统的类型，`mount` 命令需要读取文件系统的超级块或 `/proc/filesystems` 文件，最终确定文件系统的类型。在不知道存储设备中的文件系统类型时，可以使用这种方法尝试安装。

安装文件系统之后，在作为安装点的目录中，不管原先存在什么文件和子目录，此时均不可见。安装目录的属主与访问权限也将随之发生变化。从安装文件系统开始，凡是引用安装目录，即相当于引用存储设备中的文件系统的根目录。

在安装文件系统时，针对不同的文件系统类型，可以使用“`-o`”选项提供文件系统特定的选项。以 `Ext2/Ext3` 文件系统为例，表 17-3 给出了使用“`mount -o`”命令时常用的选项。如果需要同时指定多个选项，可以使用逗号分隔每个选项（注意，选项中间不能有空格），如“`-o ro,nosuid`”。

表 17-3

常用的“`-o`”选项

<code>-o</code> 选项	简单说明
<code>async/sync</code>	采用异步或同步方式处理文件系统的所有 I/O 操作
<code>atime/noatime</code>	允许或禁止更新文件系统中的文件访问时间。通常，每当访问文件时，都会更新信息节点的访问时间字段。 <code>noatime</code> 选项意味着即使访问文件也不更新信息节点的访问时间属性（此举的主要目的是加速磁盘访问）。在文件修改时间并不重要的情况下，采用 <code>noatime</code> 选项能够减少文件系统的磁盘处理活动，从而提高磁盘访问的性能。默认值为 <code>atime</code>
<code>auto/noauto</code>	<code>auto</code> 选项意味着可以使用“ <code>-a</code> ”选项安装相应的文件系统。 <code>noauto</code> 意味着只能采用完整的 <code>mount</code> 命令安装，不能使用“ <code>-a</code> ”选项安装相应的文件系统

续表

-o 选项	简单说明
defaults	采用默认的安装选项 (rw、suid、dev、exec、auto、nouser 和 async) 安装文件系统
dev/nodev	允许或禁止按文件系统解释字符或块设备
exec/noexec	允许或禁止直接执行正在安装的文件系统中的程序
group	如果用户的用户组 ID 匹配设备文件的用户组 ID，允许普通用户（非超级用户 root）安装相应的文件系统
owner	允许设备文件的属主（即使是普通用户）安装与指定设备文件对应的文件系统
remount	重新安装一个已经安装的文件系统。这个选项的主要目的或常见用法是改变已安装文件系统的安装标志，特别是把一个以只读方式安装的文件系统重新安装为可读写的文件系统。注意，使用这个选项时既不能更换设备文件，也不能改变安装点
ro/rw	以只读 (ro) 或读写 (rw) 方式安装文件系统。默认值为 rw (iso9660 等文件系统除外)
suid/nosuid	访问远程文件系统时允许或禁止文件的 suid 或 sgid 标志位发生作用
user/nouser	允许或禁止普通用户安装与卸载文件系统。安装时，安装用户的名字将会写入 /etc/mtab 文件，以便同一用户能够卸载文件系统。默认值为 nouser，即禁止普通用户安装文件系统
users	允许每个用户安装和卸载文件系统。这个选项蕴含着同时指定了 noexec、nosuid 和 nodev 三个选项，除非随后又附加了其他抑制选项，如 “users,exec,dev,suid”

### 17.4.3 /etc/fstab 文件

在 Linux 操作系统中，/etc/fstab 是一个重要的系统文件，其主要功能是维护系统中安装的文件系统。如果需要在系统引导过程中自动安装某个或某些文件系统，必须把相应的文件系统加到 fstab 文件中。fstab 文件也可用于指定交换 (swap) 分区，以便在系统引导过程中自动设置系统交换区。

/etc/fstab 文件的格式如下，其中每一行包含下列 6 个字段，表 17-4 给出了每个字段的简单说明。

```
file_system mount_point type options dump pass
```

表 17-4 /etc/fstab 文件中的每个字段及其说明

字段名	简单说明
file_system	块设备文件名、远程文件系统资源或 UUID 表示的设备文件，表示准备安装的文件系统。其有效数据如下： <input type="checkbox"/> 文件系统所在的磁盘分区，如 /dev/sda7； <input type="checkbox"/> 远程文件系统的资源名，其格式为 “<host>:<dir>”，如 “iscs:/docs”（参见第 19 章）； <input type="checkbox"/> UUID 表示的设备文件； <input type="checkbox"/> 用作交换区的磁盘分区，如 /dev/sda8； <input type="checkbox"/> 虚拟文件系统
mount_point	文件系统的安装点，表示把文件系统安装到哪个目录位置。如果安装点的目录名中包含空格字符，可以使用转义字符 “\040” 替代
type	标识文件系统的类型。Linux 系统支持大量的文件系统，如 autofs、ext2、ext3、iso9660、vfat、ntfs、jfs、nfs、sysv、ufs、proc 及 tmpfs 等。至于当前的 Linux 系统都支持哪些文件系统，可以查阅 /proc/filesystems 文件。如果这个字段为 swap，表示相应的磁盘分区用作交换区。一个特殊的 “ignore” 标志表示忽略当前行，意味着暂时停止安装相应的文件系统
options	使用 mount 命令安装相应的文件系统时需要提供的安装选项，多个安装选项中间需加逗号分隔符（其间没有空格）。有关安装选项的说明详见 mount (8) 和 nfs (5) 手册页
dump	dump 命令使用这个字段确定相应文件系统中的数据是否需要备份。如果这个字段为 0，表示不需要使用 dump 命令备份相应的文件系统
pass	在启动或重新引导系统时，fsck 使用这个字段确定文件系统的检测顺序。“/”文件系统应当总是为 1，其他非虚拟文件系统均应为 2。位于同一磁盘不同分区中的文件系统只能顺序地执行检测，但对位于不同磁盘中的文件系统，fsck 将会采用并发机制，同时进行检测。如果这个字段为 0，表示不需要使用 fsck 命令检测相应的文件系统



注意，/etc/fstab 文件中的每个字段都不能为空，字段值也不允许含有空格，字段之间必须使用空白字符（空格或制表符）作为分隔符。否则，会导致安装失败，甚至可能导致系统无法成功地启动。

下面是取自 Ubuntu Linux 系统中的一个 fstab 文件实例。

```
$ cat /etc/fstab
# /etc/fstab: static file system information.
#
# <file system> <mount point>          <type>  <options>           <dump>  <pass>
proc            /proc    proc defaults        0       0
#/dev/sda7      UUID=4c5ea7fa-d041-4128-a78a-43e171e8df76 /      ext3    relatime,errors=remount-ro 0   1
#/dev/sda8      UUID=fc504f41-ae94-4b36-9550-cbd4727f9d8e none    swap    sw                0       0
/dev/scd0       /media/cdrom0 udf,iso9660 user,noauto,exec,utf8 0   0
/dev/fd0        /media/floppy0 auto    rw,user,noauto,exec,utf8 0   0
$
```

## 17.4.4 安装文件系统

本小节将介绍怎样使用 mount 命令手工安装文件系统，或者通过在/etc/fstab 文件中增加安装项的方式自动安装文件系统。Ubuntu Linux 系统也能够自动识别与安装 CD/DVD 及 USB 等移动存储设备，只要插入移动存储设备，即可打开一个文件浏览器窗口，显示其根目录中的子目录与文件。但是，软盘或内置硬盘中的文件系统需要手工安装，在 GNOME 桌面的“位置”菜单中，通过单击软盘或 Windows 系统分区菜单项，安装相应的文件系统。

在使用 mount 命令手工安装文件系统时，原则上应当利用“-t”选项指定文件系统的类型。如果确实不知道文件系统的类型，或者图省事，也可以不指定，由 Linux 系统自己确定文件系统的类型。

### 1. 确定系统中已安装了哪些文件系统

要确定系统当前已经安装了哪些文件系统，可以使用下列 mount 命令。

```
$ mount [-lv]
```

其中的“-v”选项意味着显示已安装文件系统的详细信息。下面的例子说明了怎样利用 mount 命令显示系统中当前已安装的文件系统及有关信息。

```
$ mount
/dev/sda7 on / type ext3 (rw,relatime,errors=remount-ro)
tmpfs on /lib/init/rw type tmpfs (rw,nosuid,mode=0755)
/proc on /proc type proc (rw,noexec,nosuid,nodev)
sysfs on /sys type sysfs (rw,noexec,nosuid,nodev)
varrun on /var/run type tmpfs (rw,nosuid,mode=0755)
varlock on /var/lock type tmpfs (rw,noexec,nosuid,nodev,mode=1777)
udev on /dev type tmpfs (rw,mode=0755)
tmpfs on /dev/shm type tmpfs (rw,nosuid,nodev)
.....
$
```

另外，还可以查阅文件系统安装表/etc/mtab，确定系统中当前已安装了哪些文件系统。无论何时安装或卸载文件系统，Linux 系统都会修改文件系统安装表。因此，/etc/mtab 文件总是能够反映当前最新的文件系统安装信息。

任何用户均可利用 cat 或 more 等命令查询文件系统的安装信息，但任何人都不应直接修改这个文件。mtab 文件示例如下。

```
$ cat /etc/mtab
/dev/sda7 / ext3 rw,relatime,errors=remount-ro 0 0
tmpfs /lib/init/rw tmpfs rw,nosuid,mode=0755 0 0
/proc /proc proc rw,noexec,nosuid,nodev 0 0
sysfs /sys sysfs rw,noexec,nosuid,nodev 0 0
```

```
varrun /var/run tmpfs rw,nosuid,mode=0755 0 0
varlock /var/lock tmpfs rw,noexec,nosuid,nodev,mode=1777 0 0
udev /dev tmpfs rw,mode=0755 0 0
tmpfs /dev/shm tmpfs rw,nosuid,nodev 0 0
.....
$
```

## 2. 文件系统的自动安装

为了增加一个新的文件系统安装项，使文件系统能够在系统的启动过程中自动安装，需要以超级用户的身份编辑/etc/fstab 文件，把相应的文件系统加到/etc/fstab 文件中。同时确保每个字段都不为空，必要时可用 mkdir 命令，事先为准备安装的文件系统创建一个安装点。

下面的例子说明了怎样修改/etc/fstab 文件，以便在启动系统时能够自动地把磁盘分区 /dev/sda5 中的文件系统安装到/fedora 目录处。其中的 pass 字段为 2，意味着可以并发地检测与修复相应的文件系统。修改后的 fstab 文件如下，其中最后一行即为新加的安装项。

```
$ cat /etc/fstab
...
# /dev/sda7
UUID=4c5ea7fa-d041-4128-a78a-43e171e8df76 /      ext3 relatime,errors=remount-ro 1
# /dev/sda8
UUID=fc504f41-ae94-4b36-9550-cbd4727f9d8e none    swap  sw      0 0
...
/dev/sda5           /fedora ext3  rw      0 0
$
```

## 3. 利用/etc/fstab 文件手工安装文件系统

在利用/etc/fstab 文件安装指定或所有的文件系统时，应确保文件系统已经加到/etc/fstab 文件中。然后，可以使用下列命令安装/etc/fstab 文件中列举的文件系统。

```
# mount [-a] [directory]
```

其中，“-a”选项表示安装/etc/fstab 文件中列举的所有文件系统。目录 directory 可以是位于 /etc/fstab 文件内 mount-point 字段中的安装点，也可以是 file\_system 字段中的设备文件名。

对于手工操作来讲，安装单个文件系统的简单做法是指定安装点。下面的例子说明了怎样安装/etc/fstab 文件中列举的/backup 文件系统。

```
$ sudo mount /backup
$
```

在使用下列命令安装/etc/fstab 文件中列举的文件系统时，如果需要，系统将会在开始安装文件系统之前，对 pass 字段值大于 0 的所有文件系统进行检测与修复。

```
$ sudo mount -a
$
```

## 4. 使用 mount 命令安装 Ext2/Ext3 文件系统

要安装 Ext2/Ext3 文件系统，可以直接使用下列 mount 命令。

```
# mount -t fstype device directory
```

其中，fstype 是 ext2 或 ext3，device 是 Ext2/Ext3 文件系统所在磁盘分区的设备名，如/dev/sda5。要查询磁盘分区信息，可以使用 fdisk 等命令。directory 用于指定安装文件系统的目录位置，即安装点。

下面的例子说明了怎样利用 mount 命令，把/dev/sda5 中的 Ext2/Ext3 文件系统安装到/fedora 目录上。

```
$ sudo mount -t ext2 /dev/sda5 /fedora
$
```

## 5. 使用 mount 命令安装 FAT 文件系统

Ubuntu Linux 系统内核支持 FAT/FAT32 文件系统，其文件系统类型为 vfat。如果需要在两个操作系统上交换数据，可以使用软盘、USB 移动硬盘或 U 盘，把 FAT/FAT32 文件系统安装到 Linux 系统中。

Ubuntu Linux 系统能够自动识别位于系统盘中的 FAT/FAT32 文件系统。从 GNOME 桌面的“位



置”菜单中点击相应的菜单项，即可安装 FAT/FAT32 文件系统。当插入 USB 移动硬盘或 U 盘时，系统将会自动安装相应的文件系统，同时在 GNOME 桌面区增加一个图标，打开一个文件浏览器窗口。如果需要手工安装 FAT/FAT32 文件系统，可以使用下列 mount 命令。

```
# mount -t vfat device directory
```

其中，device 是 FAT/FAT32 文件系统的设备名，如/dev/fd0（软盘）或/dev/sda1 等。directory 用于指定安装文件系统的目录位置。

Ubuntu Linux 系统不会自动安装软盘中的文件系统，GNOME 桌面中也没有相应的图标和菜单项可用。下面的例子说明了怎样把 3.5 英寸软盘中的 FAT/FAT32 文件系统，手工安装到/mnt 目录中。

```
$ sudo mount -t vfat /dev/fd0 /mnt
```

```
$
```

## 6. 使用 mount 命令安装 ISO 9660 CD 文件系统

在 Ubuntu Linux 系统中，把 CD/DVD 插入光驱时，iso9660 类型的文件系统会自动地把 CD/DVD 安装到/media/cdrom0 目录中。如果需要手工安装，可以使用下列 mount 命令，把 CD/DVD 安装到指定的目录中。

```
$ sudo mount -t iso9660 -o ro /dev/cdrom /mnt
```

```
$
```

## 7. 使用 mount 命令安装 NTFS 文件系统

Ubuntu Linux 系统内核直接支持 NTFS 文件系统，能够自动识别位于系统盘中的 NTFS 文件系统。从 GNOME 桌面的“位置”菜单中点击相应的菜单项，即可以 NTFS 文件系统的卷标（如“Local Disk”）作为子目录名，安装到“/media/local Disk”目录中。

在安装之前，为了查询系统盘中 NTFS 文件系统的设备名，可以使用下列 fdisk 命令，列出系统中的所有 NTFS 磁盘分区。

```
$ sudo fdisk -l /dev/sda | grep NTFS
/dev/sda1      *           1     827   6252088+    7  HPFS/NTFS
$
```

采用手工安装方式时，可以把 NTFS 文件系统安装到任何目录位置，如常用的/mnt 目录等。如果同时访问的 NTFS 分区较多，可能需要创建若干安装点，如/media/c/、/media/d……，以便能够把每个 NTFS 文件系统分别安装到 Linux 文件系统的不同目录层次结构中。例如，采用下列命令可以把 Windows 系统的“C:”盘安装到/mnt 目录中。

```
$ sudo mount -t ntfs /dev/sda1 /mnt
$
```

## 8. 使用 mount 命令安装 NFS 文件系统

假定 iscas 是一个 NFS 服务器，为了把其中提供的共享目录/share/tools，作为 NFS 文件系统安装到本地系统的/tools 目录中，可以使用下列命令（参见第 19 章）。

```
$ sudo mount -t nfs -o ro iscas:/share/tools /tools
$
```

## 17.4.5 卸载文件系统

卸载文件系统意味把文件系统从安装点移走，删除/etc/mtab 文件中的安装项。在文件系统的管理与维护任务中，部分处理任务只能在未安装的文件系统中执行。此外，当不再需要临时安装的文件系统时，也要及时卸载文件系统。因此，当出现下列情况时，应当考虑卸载文件系统：

- 文件系统用过之后已不再需要；
- 文件系统受损，因而需要使用 fsck 命令检测与修复文件系统；

- 为了增加卷标、目录索引以及日志功能，因而需要使用 tune2fs 命令调整文件系统；
- 误删文件，因而需要使用 debugfs 命令调试文件系统；
- 在执行完整的文件系统备份之前，通常也需要卸载文件系统。

注意，作为系统关机过程的一部分，文件系统将会自动卸载。

卸载文件系统时，可以使用 umount 命令，其语法格式简写如下。

```
umount [-dflnrv] device | directory
umount -a [-dflnrv] [-t fstype] [-O options]
```

其中，“-f”选项表示强制卸载指定的文件系统，“-a”选项表示卸载/etc/mtab 文件中列举的所有文件系统（/proc 文件系统除外），“-t”选项表示卸载指定类型的文件系统，device 是文件系统所在的设备文件名或 NFS 远程资源名等，irectory 是准备卸载的文件系统安装点。

### 1. 卸载文件系统前的准备工作

在卸载文件系统之前，通常需要做一定的准备工作，或者具备一定的条件，如下所示。

- 通常情况下，只有超级用户才能卸载文件系统。
- 文件系统必须处于空闲状态才能够被卸载。通常，不能卸载一个尚处于工作状态的文件系统。所谓文件系统处于工作状态，意味着可能还有用户正处于或正在访问文件系统中的某个目录，用户或进程打开了文件系统中的某个文件，或者文件系统正处于共享状态。为了确保文件系统能够正常卸载，可以采取下列措施：
- 改换到不同文件系统中的目录；
- 退出 Linux 系统；
- 使用 fuser 命令列出访问文件系统的所有进程，然后通知用户停止使用准备卸载的文件系统，或者由超级用户停止运行相应的进程（如果必要或可能）；
- 取消文件系统的共享（参见第 21 章）。

### 2. 终止正在访问文件系统的所有进程

要终止正在访问文件系统的所有进程，首先可使用下列命令，显示当前正在访问文件系统的所有进程，以便了解需要终止运行哪些进程。

```
* fuser -c [-u] mount-point
```

其中，“-c”选项表示显示当前正在访问指定安装点（文件系统）中任何文件的所有进程，“-u”选项意味着在显示的每个进程 ID 后面给出与进程相关的注册用户名，“mount-point”表示需要检查的安装点（文件系统）。

然后，可以使用下列命令，终止运行当前正在访问文件系统的所有进程。

```
* fuser -c -k mount-point
```

在执行上述命令时，一个 SIGKILL 信号将会发送到当前正在使用指定文件系统的每个进程。

注意，在事先没有通知用户时，通常不应强行终止用户的进程。

最后，可以使用下列命令，验证当前是否还有任何进程正在访问文件系统。

```
* fuser -c mount-point
```

下面的例子说明了怎样停止一个正在访问/fedora 文件系统的进程 6968。

```
$ sudo fuser -c -u /fedora
/fedora:          6968c(gqxing)
$ sudo fuser -c -k /fedora
/fedora:          6968c
$ sudo fuser -c /fedora
$
```



### 3. 卸载文件系统

除了“/”、单独的/usr 或/var 文件系统之外，为了卸载其他非虚拟的文件系统，首先应确保已经完成文件系统卸载前的准备工作，然后可以使用 umount 命令卸载文件系统。注意，仅当关闭系统时才能卸载“/”、单独的/usr 以及/var 等文件系统，否则系统无法正常运行。

下面的例子说明了怎样卸载一个安装在/mnt 目录处的本地文件系统。

```
$ sudo umount /mnt  
$
```

下面的例子说明了怎样卸载一个位于磁盘分区 6 中的文件系统。

```
$ sudo umount /dev/sda6  
$
```

注意，上述命令并不能卸载当前正处于工作状态的文件系统。在紧急情况下，可以使用“umount -f”命令强制卸载一个尚在工作的文件系统。除非不得已，通常不建议采取这种卸载方式，因为这种强制卸载有可能会导致已打开的文件丢失数据。下面的例子说明了怎样强制卸载位于安装点/mnt 处的文件系统。

```
$ sudo umount -f /mnt  
$
```

### 4. 验证文件系统已经卸载

要验证文件系统是否已经被卸载，可以使用下列 mount 命令，考察命令的输出信息是否包含指定的文件系统。

```
$ mount | grep unmounted-file-system
```

## 17.5 检测与修复文件系统

fsck 或 e2fsck 命令可用于交互地检测与修复文件系统。在实施修复之前，fsck 命令通常会提示用户确认建议的处理动作。

下面以 Ext2/Ext3 文件系统为例，说明怎样利用 fsck 或 e2fsck 命令实现文件系统的检测与修复。fsck 或 e2fsck 命令的主要功能是检测文件系统的完整性，并对发现的问题尝试进行适当的修复。fsck 和 e2fsck 命令的语法格式简写如下。

```
fsck [options] [-t fstype] [fs-specific-options] [filesystem]  
e2fsck [options] [-b superblock] [-B blocksize] filesystem
```

其中，“filesystem”可以是一个磁盘分区的设备文件名，如/dev/sdb2；可以是一个文件系统的安装点，如/var；也可以是一个 Ext2/Ext3 文件系统的卷标或 UUID。如果命令行中未指定文件系统，也未指定“-A”选项，fsck 将依次检测/etc/fstab 文件中列举的所有文件系统。fsck 与 e2fsck 命令的其他常用选项如表 17-5 所示。

表 17-5

fsck 与 e2fsck 命令的常用选项

选 项	简 单 说 明
-A	遍历/etc/fstab 文件，同时检测其中列举的所有文件系统。除非指定了“-P”选项，否则首先检测“/”文件系统，然后按照/etc/fstab 文件 pass 字段指定的顺序，依次检测每个文件系统。如果 pass 字段为 0，则跳过相应的文件系统。pass 字段的值确定了文件系统检测的先后顺序，数值越小，检测的顺序越靠前。如果多个文件系统的 pass 字段具有相同的值，fsck 将会尝试并行检测相应的文件系统。在/etc/fstab 文件中，通常把“/”文件系统的 pass 字段设置为 1，其他文件系统均设置为 2，以便能够充分利用 fsck 的并行自动检测功能

续表

选 项	简 单 说 明
<b>-b superblock</b>	使用指定的备份超级块（而不是常规的主超级块）检测与修复文件系统。当文件系统的主超级块严重受损时，可以使用这个选项修复文件系统。备份超级块的位置依赖于具体的文件系统，以及文件系统的大小与逻辑数据块的大小等。在逻辑数据块为 1KB 的文件系统中，第一个备份超级块位于 8 193 号数据块；在逻辑数据块为 2KB 的文件系统中，第一个备份超级块位于 16 384 号数据块；在逻辑数据块为 4KB 的文件系统中，第一个备份超级块位于 32 768 号数据块。要确定其他备份超级块的位置，可以使用“mke2fs -n”命令查询其输出结果
<b>-B blocksize</b>	通常，e2fsck 将会尝试以各种不同的数据块规格检索和确定超级块的位置与大小。“-B”选项强制 e2fsck 按照指定的超级块规格找出超级块。如果指定规格的超级块不存在，e2fsck 将会输出一个错误信息，然后结束程序的执行
<b>-D</b>	优化文件系统中的目录。利用这个选项，通过建立索引或重建索引（如果文件系统支持目录索引，参见 tune2fs 命令），排序或压缩目录，或者采用传统的线性目录，e2fsck 将会尝试优化所有的目录
<b>-f</b>	强制执行文件系统的检测，即使文件系统完好无损
<b>-n</b>	以只读方式打开指定的文件系统，对于 fsck 的所有修复请求，均以“no”作为响应，使 fsck 能够以非交互的方式自动检测指定的文件系统，报告（而不是修复）文件系统中存在的问题
<b>-N</b>	实际上并不执行，只是向用户展示，一旦执行时 fsck 命令究竟能够做什么
<b>-p</b>	并发地检测文件系统，自动地修复部分简单的问题，而无需用户干预。一旦检测到严重的错误，立即结束文件系统的检测与修复。如果发现某个问题需要提请用户做进一步的校正处理，fsck 或 e2fsck 将会输出一个错误描述信息，然后结束程序的执行
<b>-r</b>	交互地检测与修复文件系统，在修复之前请求用户予以确认。注意，这也是 e2fsck 默认的文件系统检测与修复方式
<b>-t fstype</b>	指定文件系统的类型
<b>-y</b>	对于 fsck 的所有修复请求，均以“yes”作为响应，使 fsck 或 e2fsck 能够以非交互的方式自动检测与修复指定的文件系统

### 17.5.1 何时需要检测文件系统

#### 1. 文件系统受损的外部原因

Ext2/Ext3 等文件系统主要依赖于超级块，用以跟踪和管理信息节点与数据块的分配及释放。当系统内存中的超级块与磁盘文件系统上的超级块没有适当地同步时，就会导致磁盘上的超级块与文件系统中的实际数据不一致。此时的文件系统就需要修复，否则无法正常地安装文件系统。

由于下列原因而没有正常地停止操作系统时，将会造成超级块与实际数据的不一致，从而破坏文件系统：

- 电源故障，包括突然断电、人为造成的电源故障以及内部电源故障等；
- 强行关机，即在未正常停止 Linux 系统之前就断开电源；
- 硬件故障，如磁盘出现坏块等；
- 因 Linux 系统的内核故障而引起异常关机。

一旦文件系统受损，按照/etc/fstab 文件的设置，在引导过程中，系统将会自动地执行 fsck 命令，检测文件系统的完整性。在文件系统的检测过程中，fsck 将会尝试修复已经受损的文件系统，把修复后的文件系统安装到指定的目录位置。如果文件系统损坏严重，fsck 可能无法完全修复，从而有可能导致系统无法正常地启动。

在文件系统的修复过程中，fsck 命令将会把已分配但未引用的文件与目录置于 lost + found 目



录下，并用其信息节点号作为文件或目录的名字。如果 `lost + found` 目录不存在，`fsck` 命令将会自动创建一个。如果 `lost + found` 目录中没有足够的空间，`fsck` 命令将会尝试增加其容量。

## 2. Ext2/Ext3 文件系统受损的内部因素

在一个正常运行的 Linux 系统中，每个工作日都可能会创建、修改或删除数百个文件。每当修改文件时，操作系统都会执行一系列文件系统的更新操作。这些更新如果能够及时地全部被写到磁盘上，就不会破坏文件系统的完整性。通常，磁盘数据的更新是异步处理的，当用户进程访问文件系统，如写文件数据时，这些数据首先被复制到内存缓冲区中。此时，系统允许用户进程继续写数据，即使先前的数据尚未真正写到磁盘中。

因此，在任何给定的时刻，磁盘文件系统的状态总是滞后于内存超级块中表示的文件系统。当缓冲区需要分配他用，或者按一定的时间间隔，系统把缓冲区中的数据写到磁盘文件系统时，磁盘上的文件系统才能真正等同于内存超级块表示的文件系统。如果在停机之前没有把内存超级块中的数据写到磁盘上，磁盘上的文件系统就会出现不一致的状态，造成文件系统受损。

## 3. 文件系统状态标志字段

超级块中有一个记录文件系统状态的标志字段，`fsck` 命令使用这个标志字段确定文件系统状态，决定是否需要检测文件系统的完整性。表 17-6 给出了 Ext2/Ext3 文件系统状态标志字段的可能取值及其说明。

表 17-6

文件系统状态标志字段

状态标志字段	简单说明
<code>EXT2_ERROR_FS</code> <code>EXT3_ERROR_FS</code>	表示文件系统存在错误，如包含不一致的文件系统数据等
<code>EXT2_VALID_FS</code> <code>EXT3_VALID_FS</code>	表示未安装的文件系统是完好的、未受损的
<code>EXT3_ORPHAN_FS</code>	文件系统中存在游离的文件或目录

## 17.5.2 文件系统检测的内容

在检测 Ext2/Ext3 文件系统时，`fsck` 命令将会针对文件系统的超级块、数据块组描述、信息节点位图、数据块位图、信息节点、间接地址块和数据块等重要组成部分，检测其完整性和一致性。注意，`fsck` 并不验证文件本身的数据内容。

### 1. 超级块

超级块（包括数据块组描述、信息节点位图与数据块位图）是文件系统的核心，其中存有文件和目录的关键数据与汇总信息。每当文件数据发生变动时都需要更新超级块、数据块组描述、信息节点位图与数据块位图。因此，超级块、数据块组描述、信息节点位图与数据块位图是文件系统中更新最频繁的部分，通常也最容易受到损坏。

所谓文件系统受损，主要就是指超级块（包括数据块组描述等）中的数据受损。每当改动文件系统的信息节点或数据块时，都要修改超级块、数据块组描述、信息节点位图和数据块位图。如果卸载文件系统之前执行的最后一个文件系统操作不是 `sync` 命令时，几乎肯定会损坏超级块。

超级块的完整性检测主要包括以下 4 个方面：

- 文件系统的大小；
- 信息节点的数量；

□ 空闲数据存储块计数;

□ 空闲信息节点计数。

(1) 检测文件系统和信息节点表的大小。

文件系统的大小和信息节点区的大小均属于静态数据，一旦创建文件系统之后，这些数据是不会改变的。因此，fsck 命令实际上并没有办法严格检测这些数据，因为这些数据在创建文件系统时就已确定。但是，fsck 命令能够依据合理的推断，检测这些静态数据。原则上，文件系统的大小必须大于超级块与信息节点表示的数据块数，信息节点的数量必须小于文件系统允许的最大值。信息节点含有除文件名和信息节点号之外的所有文件属性信息，文件系统的大小和其他统计数据是 fsck 命令最为关注的重要信息。针对文件系统的所有检测都要求这些数字必须是正确的。如果检测到主超级块的静态数据有误，fsck 命令将会要求操作员指定备份超级块的位置。

(2) 空闲数据存储块检测。

空闲数据块是在数据块组的数据块位图中维护的，fsck 命令将会检测所有未占用的空闲存储块。在计数了所有的空闲存储块之后，fsck 命令将据以确定空闲存储块的数量加上信息节点已声明占用的数据块数量是否等于文件系统全部存储块的总量。如果发现任何数据块组中的数据块位图有问题，fsck 命令将会适当地修正数据块位图，并剔除分配有误的空闲存储块。

超级块的汇总信息含有文件系统中所有空闲存储块的统计计数，fsck 命令将会根据此计数，与文件系统检测过程中发现的实际空闲存储块总数进行比较。如果两个数字不一致，fsck 命令将会以实际的空闲存储块计数替换超级块中的统计数字。

(3) 空闲信息节点检测。

超级块中的汇总信息也含有文件系统中所有空闲信息节点的统计计数，fsck 命令也会根据此计数，与文件系统中的实际空闲信息节点总数进行比较。如果两个数字不一致，fsck 命令将会用实际的空闲信息节点数量替换超级块中的统计数字。

## 2. 信息节点

对信息节点的完整性检测将从信息节点表的 2 号信息节点（1 号保留）位置开始，顺序检测每个信息节点的完整性。信息节点的完整性检测包括以下几个方面的内容。

□ 类型与状态;

□ 链接计数;

□ 重复的数据块;

□ 无效的数据块;

□ 信息节点中的数据块计数;

□ 信息节点的连接性。

(1) 类型与状态检测。

信息节点中的 mode 字段描述了文件的类型与信息节点的状态。每个信息节点可以处于下列 3 种状态之一：

□ 信息节点已经被分配或占用（其中的 mode 字段为非 0）；

□ 信息节点尚未被分配或空闲（其中的 mode 字段为 0）；

□ 信息节点部分被分配（信息节点中必须具备的字段不完整）。

创建文件系统时，Linux 系统将会预留一定数量的信息节点，而且仅当必要时才会分配这些



信息节点。已经分配的信息节点会指向相应的文件，未分配的信息节点没有指向任何文件，因此其中（如地址字段）应当是空的，而处于部分分配状态的信息节点则意味着其中的信息是不正确的。出现这种状态的一种可能是由于硬件故障致使垃圾数据被写入信息节点中。对此情况，fsck 命令应当采取的唯一正确动作就是清除这样的信息节点。

#### （2）链接计数检测。

每个信息节点均包含一个链接到当前信息节点的文件或目录项的计数。fsck 命令将从根目录 “/” 开始考察整个目录结构，计算每个信息节点的实际链接计数，以验证每个信息节点的链接计数是否正确。信息节点中的链接计数与 fsck 命令确定的实际链接计数之间的差异可以归结为下列 3 种情况之一：

- 信息节点中的链接计数不为 0，但实际计数为 0。如果信息节点对应的文件或目录项不存在，就会出现此种差异，在此情况下，fsck 命令将会把没有归属的（断开链接的）文件置于 lost+found 目录中。
- 信息节点中的链接计数不为 0，实际计数也不为 0，但两个计数并不相同。如果已经增加或删除了某个目录项，但信息节点尚未更新，就会出现此种差异。在这种情况下，fsck 命令将会使用实际的链接计数替换信息节点中的链接计数。
- 信息节点中的链接计数为 0，但实际的链接计数不为 0。在这种情况下，fsck 命令将会把信息节点中的链接计数改为实际的链接计数。

#### （3）重复数据块检测。

在文件系统中，每个数据块只能分配给一个信息节点，否则即为空闲数据块，两者只能居其一。如果信息节点中的一个数据块同时归属于另外一个信息节点或空闲数据块区，则可认为这个数据块是重复的。

每个信息节点中的地址数组均直接或间接地指向归属于信息节点的所有数据块。因为间接地址块也归属于信息节点，如果间接地址块的归属也出现问题，将会直接影响拥有间接地址块的信息节点。

fsck 命令将会对归属于信息节点的每个数据块号与已分配数据块位图进行比较。如果其他信息节点也声明拥有其中的某个数据块号，fsck 将会把该数据块号置于重复数据块表中。否则，更新已分配数据块位图，使之包括相应数据块号。

如果发现重复的数据块，fsck 命令将会对信息节点位图进行第二次扫描，以便找出声明拥有每个重复数据块的其他信息节点。在此情况下，fsck 命令本身无法肯定究竟哪一个信息节点有误，因此，fsck 命令将会提示用户选择保留哪—个信息节点，清除哪—个信息节点。注意，如果信息节点中出现大量的重复数据块，通常都是由于间接地址块没有被正确写入文件系统引起的。

#### （4）无效数据块检测。

在文件系统中，所有的数据块都是从 0 开始顺序编号的。超级块中文件系统的大小确定了最大的数据块编号，编号超出这个范围的数据块均可看作无效的数据块，或简称坏块。

fsck 命令将会检测信息节点声明拥有的每一个数据块号，确定其编号是否高于文件系统中第一个数据块号并低于最后一个数据块号。如果数据块号超出了这个范围，则认为这是一个无效的数据块。

导致信息节点中出现无效数据块有可能是因为间接地址块没有写入文件系统。对此，fsck 命令将会提示用户清除这样的信息节点。

#### （5）信息节点中的数据块计数检测。

每个信息节点均包含一个数据块引用计数。实际的数据块计数应是已分配的数据块与间接地

址块之和。fsck 命令将会据此计算数据块的数量，然后与信息节点声明拥有的数据块数进行比较。如果信息节点中的数据块计数有误，fsck 命令将会提示用户予以修正。

每个信息节点也包含一个 64 位的表示文件大小的字段，这个字段给出了相应文件中的字节数。利用这个字段的字节数可以概略地计算出相应的文件应当占用多少个数据块，然后再用这个数值与信息节点声明拥有的实际数据块数进行比较，即可得出信息节点中的数据块计数是否有误。

#### (6) 信息节点的连接性检测。

每个已分配的信息节点都应当存在于文件系统中的某个目录中。如果某个信息节点没有任何引用关系，即任何目录中都没有引用这个信息节点，也非空闲信息节点，则认为此信息节点已经游离于文件系统，因而称作“未引用的”信息节点。

### 3. 目录结构

目录与普通文件的区别主要依据于信息节点中的 mode 字段。目录数据块含有一系列目录项。对目录数据块的完整性检测主要包括以下 3 个方面：

- 未分配信息节点检测，即检测目录项中的信息节点号是否指向一个尚未分配的信息节点；
- 无效信息节点检测，即检测目录项中的信息节点号是否大于文件系统中信息节点的数量；
- 不正确的“.” 和 “..” 目录项检测，即检测“.” 和 “..” 目录是否存在不正确的信息节点号；
- 游离目录检测，即检测是否存在游离的目录。

#### (1) 尚未分配的信息节点检测。

如果目录数据块中的信息节点号指向一个尚未分配的信息节点，fsck 命令将会删除相应的文件或目录项。如果包含新目录项的目录数据块已经修改并被写到磁盘文件系统中，但相应的信息节点却没有被写到磁盘文件系统中，就会出现此类问题，这种情况通常是由于意外关机造成的。

#### (2) 无效信息节点检测。

如果目录项中的信息节点号超出了信息节点区的范围，fsck 命令将会删除相应的目录项。当由于某种原因把垃圾数据写入目录数据块时，就会出现这种情况。

#### (3) 不正确的“.” 和 “..” 目录项检测。

“.” 目录项的信息节点号必须位于目录数据块中的第一个目录项，且该目录的信息节点号必须指向自身。也就是说，其数值必须等于目录的信息节点号。

“..” 目录项的信息节点号必须位于目录数据块中的第二个目录项，且此信息节点号必须等于父目录或自身（如果是根 “/” 目录）的信息节点号。

如果 “.” 和 “..” 目录的信息节点号有误，fsck 命令将会使用正确的数值予以替换。如果目录存在多个硬链接，fsck 将会把第一个发现的硬链接看作 “..” 目录应当指向的真正父目录。在此情况下，fsck 命令会建议用户删除其他名字。

#### (4) 游离目录检测

fsck 命令还将检测整个文件系统的连接关系。如果发现某个目录与文件系统没有任何连接关系，fsck 命令将会把相应的目录置于文件系统的 lost+found 目录中。当由于某种原因仅把信息节点写入文件系统，但尚未及时地把相应的目录数据块写入文件系统时，就会出现这种情况。

### 4. 间接地址块

间接地址块也归信息节点拥有，因此，间接地址块的问题将影响相应的信息节点。fsck 检测



的间接地址块问题主要包括以下两个方面：

- 多个信息节点声明拥有同一间接地址块；
- 间接地址块的块号超出文件系统的范围。

## 5. 数据块

信息节点可以直接或间接地引用 3 种数据块：

- 普通数据块；
- 符号链接数据块；
- 目录数据块。

其中，从属于文件的普通数据块含有文件的实际数据内容，`fsck` 命令不会检测普通文件数据块中数据内容的有效性。符号链接数据块含有符号链接文件中存储的实际路径名。目录数据块含有目录中的所有目录项，`fsck` 命令只能检测目录数据块的有效性。

### 17.5.3 交互地检测与修复文件系统

遇到下列情况时，需要交互地检测文件系统。

- 文件系统无法安装。
- 文件系统在使用过程中报错。当文件系统在运行过程中报错时，错误信息可能会出现在控制台上，或者被写到系统的错误信息日志文件中（严重时甚至可能会引起系统瘫痪）。

在运行 `fsck` 命令检测和修复文件系统之前，应当记住下列注意事项。

- 当使用 `fsck` 命令检测文件系统时，相应的文件系统应处于非工作状态。这是因为位于内存的超级块会定时地更新磁盘中的超级块，两个超级块之间的数据同步存在一个时间延迟问题。在 `fsck` 检测文件系统期间，如果由于文件系统发生变化，致使两个超级块的信息不一致时，`fsck` 将会误认为文件系统受到损坏。
- 在使用 `fsck` 命令检测文件系统之前，应卸载相应的文件系统。这将确保文件系统的数据结构处于完好状态。唯一的例外是“/”与“/usr”文件系统，因为这两个文件系统必须总是安装并处于工作状态，否则无法运行 `fsck` 命令（但可考虑使用其他方式引导系统，参见下列说明）。
- 如果需要修复“/”等文件系统，应尽可能使用其他设备或方式引导系统，以便这些文件系统能够处于非安装或非工作状态。

另外，只有超级用户才能使用 `fsck` 命令检测与修复文件系统。因此，在检测与修复文件系统时，必须拥有超级用户的访问权限。

检测和修复其他文件系统的步骤如下。

- (1) 卸载文件系统，然后使用相应的设备名作为 `fsck` 命令的参数，检测文件系统。
- (2) 根据 `fsck` 命令的提示，修正 `fsck` 命令报告的错误，校正和修复文件系统。
- (3) 如果执行一次 `fsck` 命令并未完全修正所有的错误，可再次运行 `fsck` 命令，重复检测文件系统。但如果多次运行 `fsck` 命令仍然不能修复所有的问题，则需要参考 17.5.6 小节中的内容。
- (4) 安装已修复的文件系统，检查 `lost+found` 目录中是否存在任何文件。`fsck` 命令置入

`lost+found` 目录中的文件均按其原有的信息节点号命名。

(5) 使用 `file` 命令确定这些文件的类型，再使用 `cat`、`od` 或 `grep` 等命令检查其内容，然后重新命名能够挽回的文件，并把它们移至适当的目录中。最后，删除或备份 `lost+found` 目录中剩下的一时仍无法辨别和处理的文件或目录，以免其过多地占用该目录的目录项空间。

如果是“`/`”、“`/usr`”或“`/var`”等文件系统有问题，应尽可能采用重新引导系统的方式，由 Linux 系统自动修复文件系统。如果多次重新启动系统仍然无法修复文件系统，则需要使用 CD/DVD 等安装介质把系统引导至维护模式，然后按照上述步骤进行恢复。

下面的例子说明了怎样检测与修复软盘 (`/dev/fd0`) 中的文件系统，校正其文件大小计数不正确的问题（在响应“`Fix<y>?`”的询问时，可输入“`y`”接着按 Enter 键，或者直接按 Enter 键）。

```
$ sudo fsck -t ext3 /dev/fd0
fsck 1.40.8 (13-Mar-2008)
e2fsck 1.40.8 (13-Mar-2008)
/dev/fd0 was not cleanly unmounted, check forced.
Pass 1: Checking inodes, blocks, and sizes
Inode 18, i_size is 66, should be 2048. Fix<y>? yes

Pass 2: Checking directory structure
Pass 3: Checking directory connectivity
Pass 4: Checking reference counts
Pass 5: Checking group summary information

/dev/fd0: ***** FILE SYSTEM WAS MODIFIED *****
/dev/fd0: 19/184 files (5.3% non-contiguous), 60/1440 blocks
$
```

#### 17.5.4 自动检测与修复文件系统

利用“`fsck -p`”命令，可以任由 `fsck` 命令自动检测与修复由于非正常关机而引起的文件系统受损问题。注意，如果遇到需要由操作员干预的问题，`fsck` 命令将会输出一个简单的错误描述，然后立即停止运行。此外，“`fsck -p`”命令能够并发地检测文件系统。

下面的例子说明了怎样自动地检测与修复`/dev/sdb2` 文件系统。

```
$ sudo fsck -t ext3 -p /dev/sdb2
fsck 1.40.8 (13-Mar-2008)
/dev/fd0: was not cleanly unmounted, check forced.
/dev/fd0: Inode 12, i_blocks is 2, should be 522. FIXED.
/dev/fd0: /lost+found not found. CREATED.
/dev/fd0: 15/184 files (6.7% non-contiguous), 759/1440 blocks
$
```

采用交互方式检测文件系统时，也可以使用“`-y`”选项运行 `fsck` 命令，实现文件系统的自动检测。“`-y`”选项表示，对于 `fsck` 程序提出的每一个修复建议，总是以“`yes`”给予确认。示例如下。

```
$ sudo fsck -t ext3 -y /dev/fd0
fsck 1.40.8 (13-Mar-2008)
e2fsck 1.40.8 (13-Mar-2008)
/dev/fd0 was not cleanly unmounted, check forced.
Pass 1: Checking inodes, blocks, and sizes
Pass 2: Checking directory structure
Pass 3: Checking directory connectivity
lost+found not found: Create? yes

Pass 4: Checking reference counts
Pass 5: Checking group summary information
Block bitmap differences: +603
Fix? Yes
```



```
/dev/fd0: ***** FILE SYSTEM WAS MODIFIED *****
/dev/fd0: 15/184 files (6.7% non-contiguous), 759/1440 blocks
$
```

### 17.5.5 恢复严重受损的超级块

当 fsck 命令检测到文件系统的超级块已经严重受损，文件系统无法正常安装时，必须采用备份超级块才能恢复文件系统。Ext2/Ext3 文件系统存在多个超级块副本，可以利用 fsck 命令的“-b”选项，选用任何一个超级块副本替换主超级块，具体步骤如下。

首先使用下列 mkfs 命令显示超级块中的数据（注意一定要用“-n”选项，如果忽略了这个选项，将会毁灭文件系统中的所有数据，而代之以一个空的文件系统）。

```
# mkfs -t fstype -n /dev/device-name
```

上述命令将会显示之前使用 mkfs 命令创建文件系统时用作超级块副本的数据块号。

然后，使用下列 fsck 命令，同时给出超级块副本，使之替换受损的超级块。

```
# fsck -t fstype -b block-number /dev/device-name
```

上述 fsck 命令将会使用指定的超级块副本恢复主超级块。通常情况下，几乎总是能够使用 8193 号数据块作为备份超级块，参见“mkfs -t fstype -n /dev/device-name”命令的输出信息。

如果是“/”、“/usr”或“/var”等文件系统中的超级块严重受损，导致无法正常地启动系统，此时有两种选择：重装 Linux 系统，或者利用 CD/DVD 安装介质引导系统进入维护模式，然后按照上述步骤进行恢复。

下面的例子说明了怎样使用 mkfs 命令获取超级块的副本，然后利用 fsck 命令和第一个超级块副本，即 8193 号数据块，修复超级块受损的文件系统。

```
$ sudo mkfs -t ext3 -n /dev/sdb
mke2fs 1.40.8 (13-Mar-2008)
/dev/sdb is entire device, not just one partition!
Proceed anyway? (y,n) y
Filesystem label=
OS type: Linux
Block size=1024 (log=0)
Fragment size=1024 (log=0)
16000 inodes, 64000 blocks
3200 blocks (5.00%) reserved for the super user
First data block=1
Maximum filesystem blocks=65536000
8 block groups
8192 blocks per group, 8192 fragments per group
2000 inodes per group
Superblock backups stored on blocks:
      8193, 24577, 40961, 57345

$ sudo fsck -t ext3 -b 8193 /dev/sdb
fsck 1.40.8 (13-Mar-2008)
e2fsck 1.40.8 (13-Mar-2008)
/dev/sdb was not cleanly unmounted, check forced.
Pass 1: Checking inodes, blocks, and sizes
Pass 2: Checking directory structure
Pass 3: Checking directory connectivity
Pass 4: Checking reference counts
Pass 5: Checking group summary information
/dev/sdb: ***** FILE SYSTEM WAS MODIFIED *****
/dev/sdb: 11/16000 files (9.1% non-contiguous), 7400/64000 blocks
$
```

### 17.5.6 解决 fsck 命令无法修复的文件系统问题

fsck 命令采用多次扫描的分析处理方式，而后期扫描分析时修正的错误可能还会暴露出需要

在早期扫描分析阶段才能检测和处理的其他问题。因此，有时需要多次运行 fsck 命令，直至 fsck 命令不再报错。这种做法能够确保发现和修复所有的错误。但是，fsck 命令不会自己连续地运行直至排除所有的错误，因此必须手工地重复输入命令。

用户应当关注 fsck 命令显示的各种信息，这些信息有助于修正错误。例如，假定有一条信息指出某个目录已经损坏，如果删除该目录，fsck 命令修复文件系统的工作可能很快就会成功地结束。

如果无法完全修复一个文件系统，但能够以只读方式进行安装，则可以尝试使用 cp、tar 或 cpio 等命令从文件系统中取出所有或部分数据。

如果 fsck 命令无法完全修复文件系统，可以尝试使用 debugfs 或 e2image 等命令修正错误。在最坏的情况下，也许需要利用 mkfs 命令重建文件系统，然后再利用备份介质恢复数据。

如果由于磁盘的硬件故障导致文件系统错误，也许需要重新格式化磁盘（分区）或重建文件系统，利用 mkfs 命令的“-c”选项，剔除坏块，然后再恢复其中的用户数据。注意，如果是硬件错误，通常会重现甚至多次重复地显示相同的错误信息。

### 17.5.7 fsck 的阶段处理方式

通常，当系统非正常地停机，导致最新的文件系统的变化没有被及时更新到磁盘时，在重新引导系统的过程中，fsck 命令将会以非交互的方式，自动地检测并修复文件系统。但这种修复只能解决基本的问题，一旦遇到严重的情况，fsck 不会进一步尝试自行修复，而是报告出错信息，然后停止运行。

当用户以交互方式运行 fsck 命令时，fsck 将会报告检测到的所有错误，并自行解决自己能够修复的简单问题。对于比较严重的错误，fsck 命令将给出简明的错误信息，同时提请用户做出选择。当使用“-y”或“-n”选项运行 fsck 命令时，用户的选择总是按照预定的“yes”或“no”响应 fsck 命令每一个提问。

在这种文件系统的修复过程中，某些校正动作可能导致数据的丢失。数据丢失的数量和严重程度可以从 fsck 命令的输出信息中做出判断。

在正式开始检测文件系统之前，fsck 首先会执行命令行的语法检查，然后设置工作表，打开必要的文件。如果存在，此时的错误信息主要涉及用户提供的命令行选项是否正确，运行 fsck 命令时的内存分配申请，打开文件的请求是否能够得到满足，文件的状态及大小检测能否通过，创建文件是否成功。上述初始化阶段的任何错误都会终止以非交互形式运行的 fsck 命令。

fsck 命令是一个采用多阶段分析处理方式检测文件系统的程序。在完成初始化之后，fsck 命令将分 5 个阶段，分别检测信息节点、数据块与文件的大小，检测文件的路径名、连接性、引用计数以及信息节点与数据块位图，进行必要的纠正，或者请求用户做出选择。各检测阶段处理的具体内容如下所示：

- 第 1 阶段，检测信息节点、数据块和文件的大小；
- 第 2 阶段，检测目录的结构及目录项；
- 第 3 阶段，检测目录的连接性；
- 第 4 阶段，检测信息节点的引用计数；
- 第 5 阶段，检测数据块组及文件系统的汇总信息。

下面是以交互方式运行 fsck 命令，在成功地结束文件系统检测之后给出的信息。

```
$ sudo fsck -t ext3 /dev/sdb
fsck 1.40.8 (13-Mar-2008)
```



```
e2fsck 1.40.8 (13-Mar-2008)
/dev/sdb was not cleanly unmounted, check forced.
Pass 1: Checking inodes, blocks, and sizes
Pass 2: Checking directory structure
Pass 3: Checking directory connectivity
Pass 3A: Optimizing directories
Pass 4: Checking reference counts
Inode 12030 ref count is 1, should be 2. Fix<y> yes

Unaattached inode 12038
Connect to /lost+found<y> yes

Inode 12038 ref count is 2, should be 1. Fix<y> yes

Pass 5: Checking group summary information

/dev/sdb: ***** FILE SYSTEM WAS MODIFIED *****
/dev/sdb: 188/16000 files (1.6% non-contiguous), 8751/64000 blocks
$
```

检测与修复文件系统的处理过程需要大量地访问磁盘，Ext2/Ext3 文件系统采用了一种优化的算法，能够避免重复地访问文件系统的结构数据。在检测处理的过程中，fsck 将会按照数据块地址号对信息节点和目录进行排序，以便随后的处理能够减少磁头的移动，提高磁盘访问的效率。

### 1. 第 1 阶段：检测信息节点、数据块计数与文件的大小

在第 1 阶段，fsck 将会遍历文件系统中的所有信息节点，检测每一个信息节点的连接状态，除信息节点之外，这种检测不会对其他任何对象进行交叉检测。例如，此阶段的检测只是确保文件的模式字段是合法的，信息节点中涉及的所有数据块的块号地址都是合法有效的。最后，fsck 本身也采用数据块位图和信息节点位图的方式，把所有已占用的数据块和信息节点汇集到一起，以便为后续阶段的其他检测做好准备。

如果 fsck 发现某个数据块存在多个信息节点同时引用的情况，将会调用“Pass 1B”直至“Pass 1D”等多个辅助处理过程解决此类冲突：或者复制共享的数据块，以便每个信息节点都有一个共享数据块的副本；或者从信息节点中删除重复的引用。

第 1 处理阶段的处理过程需要较长的运行时间，因为 fsck 需要把所有的信息节点都读入内存并进行检测。为了减少之后的 I/O 时间，fsck 将会把所有的重要文件系统信息（如文件系统内每个目录数据块的磁盘位置等）缓存在内存中，这将避免在第 2 个处理阶段（Pass 2）处理目录时再重新读取目录的信息节点结构及其相关数据。

这个阶段主要检测信息节点表，考察信息节点的组成部分（如文件类型、数据块计数、文件的大小以及信息节点结构的格式）是否有误，检测并修复下列问题：

- 检测信息节点对应的文件的类型；
- 考察信息节点指向的数据块中是否存在无效的数据块，不同信息节点是否包含重复的数据块，分析并处理重复的数据块；
- 检测信息节点对应的文件或目录的大小；
- 检测信息节点的格式。

### 2. 第 2 阶段：检测目录的结构及目录项

第 2 阶段主要检测目录的结构数据。任何一个目录项只能存在于目录文件所在的某个数据块中，不可能涉及多个数据块，因此，可以单独地检测每个目录文件的数据块，使 fsck 能够按块号

地址对目录文件的所有数据块进行排序，并按升序检测每个目录文件的数据块，从而减少磁头的移动，提高检测的效率。检测目录的目的是确保所有的目录项都是合法有效的，其引用的信息节点号也确实是当前正在使用的信息节点。

在检测目录时，针对每个目录信息节点的第一个目录数据块，首先验证其中的当前目录“.”与父目录“..”两个目录项是否确实存在，确认当前目录项“.”中的信息节点号与当前的实际目录的信息节点号是否匹配（此时暂不考虑父目录项“..”，而是留在第3阶段进行处理）。

在第2阶段的检测过程中，fsck将会把每个目录的父目录信息缓存在内存中。如果目录间存在多个引用关系，fsck将会把多余的引用作为不合法的硬链接删除。

值得一提的是，在第2阶段的后期处理过程中，fsck需要执行的所有磁盘I/O操作几乎都已完成。第3、第4和第5阶段检测需要的信息均缓存在内存中。因此，剩下的检测处理过程基本上都是在CPU和内存中执行的，其处理时间仅占整个文件系统检测时间的5%~10%。

在这个处理阶段中，fsck将检查并分析文件系统中的所有目录，检测其中的目录项是否指向未分配的或坏的信息节点。同时检测根目录对应的信息节点，删除前一阶段发现的含有无效信息节点号的目录项，报告并修复下列错误：

- 根目录信息节点中的模式（即访问权限）与状态字段是否有误；
- 目录信息节点中地址指针是否超出文件系统的范围；
- 目录项中的信息节点号是否有效；
- 目录的完整性是否存在问题。

### 3. 第3阶段：检测目录的连接性

第3阶段主要检测目录的连接性。fsck将利用第2阶段缓存的信息，跟踪每一个目录直至到达根目录的整个路径。此时，fsck将会同时检测每个目录文件中的父目录项“..”，确认其是合法有效的。在跟踪目录的过程中，如果发现任何无法回溯到根目录的目录，fsck将会把这样的目录移至lost+found目录。

在这个阶段，fsck将根据第2阶段对目录的考察，重点检测信息节点表中是否存在没有目录归属的信息节点，因为每一个分配的信息节点至少都应有一个目录归属。该阶段报告并修复下列错误：

- 是否存在没有目录归属的信息节点；
- lost+found目录是否存在，或者目录项位置是否已全部被占用。

除了lost+found目录下没有空间之外，这个阶段发现的所有错误都是可以校正的（包括自动检测与修复方式）。

### 4. 第4阶段：检测信息节点的链接计数

在第4阶段的检测过程中，fsck将重点检测所有信息节点的链接计数。fsck将会遍历所有的信息节点，读取其中的链接计数，然后使用第1阶段检测期间缓存的信息节点链接计数，与第2和第3阶段计算出来的链接计数，逐一进行比较。如果存在未被删除，但链接计数为0的文件，则将其移至lost+found目录。该阶段报告并修复下列错误：

- 是否存在未引用的文件、符号链接文件与目录；
- 文件、目录、符号链接文件以及特殊文件的链接计数是否正确；
- 空闲信息节点的统计计数是否正确。

### 5. 第5阶段：检测数据块组及文件系统的汇总信息

在第5阶段即最后一个检测阶段，fsck将检测文件系统汇总信息的有效性。fsck将利用第1



阶段构造的数据块及信息节点位图与文件系统中的实际位图进行比较。如果需要，则对文件系统中的实际位图进行必要的校正。

在这个处理阶段，fsck 主要检查数据块区，检测坏块、重复的数据块、缺失的数据块、空闲和已用数据块计数、空闲和已用信息节点以及相应的位图，报告并修复下列错误：

- 已用信息节点位图是否遗漏了已分配的信息节点；
- 已用信息节点位图中是否出现了空闲信息节点；
- 空闲数据块位图是否遗漏了空闲数据块；
- 空闲数据块的统计计数是否正确；
- 已用信息节点的统计计数是否正确。

一旦完成文件系统的检测，fsck 命令将会在最后给出一个分析处理的总结报告，即给出下列文件系统使用情况的汇总信息。

*filesys: used/max files (percent non-contiguous), used/max blocks*

其中，每个字段的意义表述如下：

- *filesys* 当前检测的文件系统的名字；
- *used/max files* 当前已经占用的文件数量/最大的文件容量；
- *percent non-contiguous* 磁盘数据块的非线性化比率；
- *used/max blocks* 当前已经占用的数据块数量/最大的数据块数量。

例如，当以交互方式运行 fsck 命令结束之后，fsck 将会给出类似如下的信息。

```
$ sudo fsck -t ext3 /dev/sdb
fsck 1.40.8 (13-Mar-2008)
e2fsck 1.40.8 (13-Mar-2008)
/dev/sdb was not cleanly unmounted, check forced.
Pass 1: Checking inodes, blocks, and sizes
Pass 2: Checking directory structure
Pass 3: Checking directory connectivity
Pass 3A: Optimizing directories
Pass 4: Checking reference counts
Pass 5: Checking group summary information
Inode bitmap differences: -12
Fix<y> yes

/dev/sdb: ***** FILE SYSTEM WAS MODIFIED *****
/dev/sdb: 187/16000 files (1.6% non-contiguous), 8750/64000 blocks
$
```

上述汇总信息表示，/dev/sdb 存储设备对应的文件系统中共有 187 个文件，占用了 8750 个数据块，文件系统总共可以容纳 16 000 个文件，拥有 64 000 个数据块，数据块的非线性化比率为 1.6%。

## 17.6 调试文件系统

### 17.6.1 概述

debugfs 是一个交互式文件系统调试程序，可用于考察与修复 Ext2/Ext3 文件系统，恢复误删的文件等，其语法格式简写如下。

```
debugfs [-wci] [-b blocksize] [-s superblock] [-f cmd_file] [-R request] [device]
```

其中，`device` 是一个包含 Ext2/Ext3 文件系统的设备文件，如`/dev/sdXX`等。其他选项的说明参见表 17-7。

表 17-7 debugfs 命令的常用选项及其简单说明

选 项	简 单 说 明
<code>-w</code>	以读写方式打开文件系统。如果未指定这个选项，将会以只读方式打开文件系统。在文件系统调试期间发生的任何改动，无法写入文件系统
<code>-c</code>	假定文件系统已经严重受损，因而只能以非常规的只读方式打开文件系统
<code>-i</code>	表示指定的文件是一个采用 e2image 命令创建的 Ext2/Ext3 文件系统映像文件。注意，由于 Ext2/Ext3 文件系统映像文件仅包含超级块、数据块组描述、数据块位图与信息节点位图以及信息节点表，许多 debugfs 子命令（如 <code>ls</code> 和 <code>dump</code> 等）无法正确地执行
<code>-b blocksize</code>	强制使用指定的数据块大小作为文件系统逻辑数据块的大小，而不是由 <code>mkfs</code> 命令根据磁盘分区的容量确定的逻辑数据块大小
<code>-s superblock</code>	使 <code>debugfs</code> 按给定的数据块号读取文件系统的超级块（而不是读取默认的主超级块）。注意，“ <code>-s</code> ”选项要求同时给出“ <code>-b</code> ”选项
<code>-f cmd_file</code>	以批处理的方式，从指定的脚本文件 <code>cmd_file</code> 中读取并执行 <code>debugfs</code> 支持的子命令
<code>-R request</code>	使 <code>debugfs</code> 仅执行一个指定的请求，然后即结束命令的执行

## 17.6.2 交互式调试子命令

`debugfs` 命令提供一组丰富的交互式调试子命令，用于调试、修改文件系统。表 17-8 给出了部分常用的子命令及其简单说明。

表 17-8 debugfs 支持的部分交互式调试子命令

交互式调试子命令	简 单 说 明
<code>bmap filespec logical_block</code>	显示指定信息节点中与指定逻辑数据块号相应的物理数据块号。利用这个命令，可以找出一个文件的实际数据存储位置。例如，下列命令表示信息节点号 2（即根目录）占用的第一个（0）逻辑数据块，其物理数据块号为 33。 <code>debugfs: bmap &lt;2&gt; 0</code> 33 <code>debugfs:</code>
<code>cat filespec</code>	显示指定文件的数据内容。例如，假如我们已经利用 <code>write</code> 子命令，把 <code>/etc/hostname</code> 文件复制到当前正在调试的文件系统，其文件名也为 <code>hostname</code> ，信息节点号为 14，下列命令将会显示出这个文件的内容。 <code>debugfs: cat &lt;14&gt;</code> <code>iscas</code> <code>debugfs:</code>
<code>cd filespec</code>	进入指定的目录，并把新的目录作为当前工作目录
<code>chroot filespec</code>	把指定的目录作为虚拟根目录
<code>close</code>	关闭当前打开的文件系统
<code>clri file</code>	清除指定文件中的数据
<code>dump [-p] filespec out_file</code>	把指定文件中的数据写入指定的输出文件。如果给出了“ <code>-p</code> ”选项，则按源文件信息节点中的属主、用户组和访问权限等属性设置输出文件。注意，输出文件位于调用 <code>debugfs</code> 命令之前所在的文件系统，而非当前正在调试的文件系统。因此，如果给出的是相对路径名，则输出文件应当是相对于运行 <code>debugfs</code> 命令时所处的工作目录
<code>expand_dir filespec</code>	扩充指定目录文件的容量，即再为目录文件分配一个数据块
<code>feature [fs_features] [-fs_features]</code>	用于设置或清除超级块中的指定文件系统特性。之后，输出文件系统特性集合的当前状态



续表

交互式调试子命令	简单说明
<b>ffb [count [goal]]</b>	find_free_block 子命令的简化形式，用于找出空闲的数据块，或者从指定的数据块位置（goal）开始，找出指定数量（count）的空闲数据块地址（数据块号）。例如，下列命令表示文件系统中第一个空闲数据块的地址是 50。 debugfs: find_free_block Free blocks found: 50 debugfs:
<b>fin [dir [mode]]</b>	find_free_inode 子命令的简化形式，用于找出空闲的信息节点。如果以信息节点号的形式指定了某个目录（dir），则从指定的目录（信息节点号）开始找出空闲的信息节点。例如，下列命令表示文件系统中第一空闲的信息节点号为 15。 debugfs: find_free_inode Free inode found: 15 debugfs:
<b>freeb block [count]</b>	把指定的数据块标记为未分配的空闲数据块。如果给出了数据块计数，则从指定的数据块开始依次处理
<b>freei filespec</b>	释放指定的信息节点
<b>help 或 ?</b>	输出 debugfs 命令支持的子命令列表。当进入 debugfs 命令的交互环境时，可以使用 help 列出 debugfs 命令支持的所有子命令
<b>icheck blocks</b>	输出占用了指定数据块的信息节点（列表）。例如，下列命令表示数据块 47、48 和 49 分别对应于信息节点 12、13 和 14。 debugfs: icheck 47 48 49 Block Inode number 47 12 48 13 49 14 debugfs:
<b>imap filespec</b>	输出指定文件信息节点在信息节点区中的位置。例如，根目录的信息节点 2 在信息节点区中的位置如下。 debugfs: imap <2> Inode 2 is part of block group 0 locate at block 10, offset 0x0080 debugfs:
<b>init_filesys device blocksize</b>	在指定的设备上，使用指定的数据块容量，创建一个 Ext2 文件系统。注意，这种做法只是调用低级的库函数，设置超级块和数据块组描述而已，不会完全初始化所有的数据结构
<b>kill_file filespec</b>	释放指定文件的信息节点及其数据块，但这种文件删除方式并不能删除指定文件在目录文件中的目录项
<b>lcd directory</b>	把运行 debugfs 命令时的工作目录改换为系统中的指定目录（而非当前正在调试的文件系统中的目录）
<b>logdump [-acs] [-b&lt;block&gt;] [-i&lt;filespec&gt;] [-f&lt;journal_file&gt;] [output_file]</b>	转储 ext3 文件系统中的日志信息。通常，日志的信息节点是由超级块指定的，但可以利用“-i”选项指定日志实际使用的信息节点。“-f”选项用于指定包含日志数据的文件。“-s”选项表示利用超级块中的备份信息确定日志的存储位置。“-a”选项表示输出数据块组描述所在的所有数据块中的内容。“-b”选项表示输出指定数据块中的所有日志记录。“-c”选项意味着输出“-a”和“-b”选项选择的所有数据块中的内容
<b>ln filespec dest_file</b>	使用指定的文件（filespec）创建一个硬链接文件（dest_file）。注意，这种做法不会调整文件的链接计数
<b>ls [-l] [-d] [filespec]</b>	列出指定目录中的文件。其中的“-l”选项表示以长列表的格式显示更多的文件属性信息。“-d”选项表示仅列出删除的目录项。例如，下列命令用于显示根目录中的文件列表。 debugfs: ls -l <> 2 40755 (2) 0 0 1024 2-Oct-2008 16:31 . 2 40755 (2) 0 0 1024 2-Oct-2008 16:31 .. 11 40700 (2) 0 0 12288 2-Oct-2008 16:31 lost+found (END)
<b>mi filespec</b>	modify_inode 子命令的简化形式，用于修改指定文件的信息节点数据结构中的数据

续表

交互式调试子命令	简 单 说 明
<b>mkdir filespec</b>	创建一个指定的目录。例如，下列命令将会在当前目录中创建一个 newdir 子目录。 debugfs: <b>mkdir newdir</b> debugfs: <b>ls -l</b> 2 40755 (2) 0 0 1024 2-Oct-2008 16:31 . 2 40755 (2) 0 0 1024 2-Oct-2008 16:31 .. 11 40700 (2) 0 0 12288 2-Oct-2008 16:31 lost+found 12 40755 (2) 0 0 1024 2-Oct-2008 16:35 newdir (END)
<b>mknod filespec [p   c   b major minor]</b>	创建一个指定的设备特殊文件（如命名的管道文件、字符特殊文件或块特殊文件）
<b>ncheck inode_nums</b>	输出指定信息节点号对应的文件名（列表），示例如下。 debugfs: <b>ncheck 12 13 14</b> Inode Pathname 12 /newdir 13 /profile 14 /hostname debugfs:
<b>open [-w] [-f] [-i] [-c] [-b blocksize] [-s superblock] device</b>	在进入 debugfs 交互命令环境之后，打开准备调试的文件系统。其中的“-w”选项表示以写方式打开文件系统。“-f”选项表示强制打开文件系统，即使文件系统存在问题，无法正常打开文件系统。“-c”、“-b”、“-i”和“-s”具有与 debugfs 命令同名选项相同的功能
<b>pwd</b>	显示当前工作目录。例如，下面的例子表示 debugfs 命令当前处于正在调试的文件系统的根目录。 debugfs: <b>pwd</b> [pwd] INODE: 2 PATH:/ [root] INODE: 2 PATH:/ debugfs:
<b>quit 或 q</b>	退出 debugfs 命令
<b>rdump directory destination</b>	把指定目录中的所有文件（包括普通文件、符号链接文件以及子目录等）递归地复制到指定的目的目录位置（位于非调试的文件系统中）。注意，在使用这个子命令之前，目的目录必须已经存在。例如，为了把调试文件系统当前目录中 Project 子目录下的所有文件备份到 /var/backup 目录，可以使用下列命令。 debugfs: <b>rdump Project /var/backup</b> debugfs:
<b>rm pathname</b>	删除指定的文件。如果删除文件时导致信息节点的链接计数为 0，则释放相应的信息节点。这个子命令的功能等同于 unlink() 系统调用
<b>rmdir filespec</b>	删除指定的目录。例如，下列表示删除新建的 newdir 目录。 debugfs: <b>rmdir newdir</b> debugfs:
<b>setb block [count]</b>	把指定的数据块标记为已分配的数据块。如果给出了数据块计数，则从指定的数据块开始，把指定数量的数据块均标记为已分配的数据块
<b>set_bg bgnnum field value</b>	set_block_group 子命令的简化形式，用于修改指定的数据块组描述，把其中的指定字段设置为指定的值。此外，还可以利用“-l”选项，列出 set_block_group 子命令能够设置的数据块组描述中的所有字段
<b>seti filespec</b>	把指定的信息节点标记为已分配的信息节点
<b>sif filespec field value</b>	set_inode_field 子命令的简化形式，用于修改指定的信息节点，把其中的指定字段设置为指定的值。此外，还可以利用“-l”选项，列出 set_inode_field 子命令能够设置的所有信息节点字段



续表

交互式调试子命令	简 单 说 明
<b>ssv field value</b>	set_super_value 子命令的简化形式，用于修复超级块，把其中的指定字段设置为指定的数值。此外，还可以利用“-l”选项，列出 set_super_value 子命令能够设置的所有超级块字段
<b>stat filespec</b>	show_inode_info 子命令的简化形式，用于显示指定信息节点结构定义中的数据内容
<b>stats [-h]</b>	show_super_stats 子命令的简化形式，用于显示超级块和数据块组描述中的数据。如果给出了“-h”选项，则仅显示超级块中的数据
<b>testb block [count]</b>	测试指定的数据块是否已经分配。如果同时给出了数据块计数，则从指定的数据块开始，测试指定数量的数据块是否均已分配，示例如下。 debugfs: testb 33 Block 33 marked in use debugfs:
<b>testi filespec</b>	测试指定的信息节点是否已经分配，示例如下。 debugfs: testi <2> Inode 2 is marked in use debugfs:
<b>undel &lt;inode num&gt; [pathname]</b>	undelete 子命令的简化形式，用于恢复删除的信息节点号，把指定的信息节点标记为当前正在使用的信息节点，同时，如果指定了路径名，则把恢复的信息节点链接到指定的目录。在使用 undel 命令恢复误删的文件之后，应当总是运行 e2fsck 命令，修复文件系统。注意，如果是恢复大量误删的文件，把信息节点连接到一个目录有可能需要扩展目录的存储空间，导致分配的数据块取自某个已删的文件，使相应的文件无法恢复。因此，比较安全的做法是在恢复所有的信息节点时，不指定目的目录，最后，再单独使用 debugfs 的 ln 子命令，把信息节点链接到一个目的目录，或者使用 e2fsck 命令修复文件系统，把恢复的所有信息节点链接到 lost+found 目录（参见后面的讨论）
<b>unlink pathname</b>	删除指定的文件。注意，如果删除的文件是一个硬链接文件，debugfs 命令不会真正调整信息节点中的链接计数字段
<b>write source_file out_file</b>	把文件系统内指定文件（source_file）中的数据复制到新建的输出文件（out_file）。注意，这里的输出文件指的是调试文件系统中的文件。例如，为了把系统文件/etc/profile 复制到当前正在调试的文件系统中，可以使用下列命令。 debugfs: write /etc/profile profile Allocated inode: 13 debugfs: ls -l 2 40755 (2) 0 0 1024 2-Oct-2008 16:31 . 2 40755 (2) 0 0 1024 2-Oct-2008 16:31 .. 11 40700 (2) 0 0 12288 2-Oct-2008 16:31 lost+found 12 40755 (2) 0 0 1024 2-Oct-2008 16:36 newdir 13 100644 (1) 0 0 497 2-Oct-2008 16:39 profile 14 100644 (1) 0 0 1024 2-Oct-2008 16:35 hostname (END)

在 debugfs 命令的语法格式中，子命令中的参数 filespec 用于指定文件系统中的信息节点或文件的路径名。在指定 filespec 参数时，可以采用两种形式：第一种形式是在信息节点号前后加单书名号，如<2>；第二种形式是采用常规的路径名。路径名的第一个字符如果为斜线字符“/”，表示 debugfs 当前打开的文件系统的根目录；否则，表示相对于 debugfs 当前工作目录的相对路径名。要改换到不同的目录位置，可以使用 debugfs 的 cd 子命令。

要显示超级块和数据块组描述中的当前数据，可以使用下列 show\_super\_stats (stats) 命令。

```

debugfs: stats
Filesystem volume name: <none>
Last mounted on: <not available>
Filesystem UUID: 19b0c37f-4000-4667-a1a3-9d77ee130ab6
Filesystem magic number: 0xEF53
Filesystem revision #: 1 (dynamic)
Filesystem features: ext_attr resize_inode dir_index filetype sparse_super
Filesystem flags: signed_directory_hash
Default mount options: (none)
Filesystem state: clean
Errors behavior: Continue
Filesystem OS type: Linux
Inode count: 184
Block count: 1440
Reserved block count: 72
Free blocks: 1404
Free inodes: 170
First block: 1
Block size: 1024
Fragment size: 1024
Reserved GDT blocks: 5
Blocks per group: 8192
Fragments per group: 8192
Inodes per group: 184
:

```

在修复超级块时，可以利用下列命令列出 set\_super\_value (ssv) 命令能够设置的所有超级块字段。

```

debugfs: ssv -l
Superblock fields supported by the set_super_value command:
    inodes_count          unsigned integer
    blocks_count          unsigned integer
    r_blocks_count        unsigned integer
    free_blocks_count    unsigned integer
    free_inodes_count    unsigned integer
    first_data_block     unsigned integer
    log_block_size       unsigned integer
    log_frag_size        integer
    blocks_per_group     unsigned integer
    frags_per_group      unsigned integer
    inodes_per_group     unsigned integer
    mtime                date/time
    wtime                date/time
    mnt_count            unsigned integer
    max_mnt_count        integer
    state                unsigned integer
.....
:

```

要显示根目录信息节点数据结构中的数据内容，可以使用下列 stat 命令。

```

debugfs: stat <2>
Inode: 2 Type: directory Mode: 0755 Flags: 0x0 Generation: 0
User: 0 Group: 0 Size: 1024
File ACL: 0 Directory ACL: 0
Links: 3 Blockcount: 2
Fragment: Address: 0 Number: 0 Size: 0
ctime: 0x48e4504f -- Thu Oct 2 16:31:15 2008
atime: 0x48e45052 -- Thu Oct 2 16:31:15 2008
mtime: 0x48e4504f -- Thu Oct 2 16:31:15 2008
BLOCKS:
(0):33
TOTAL: 1
(END)

```



### 17.6.3 应用举例 1——恢复误删的文件

一提到文件系统调试器，自然会联想到，在不经意地删除了文件之后，能否恢复误删的文件。在一定的条件下，回答是肯定的，但并非总是如此。在使用 GNOME 桌面环境访问 Linux 系统时，同 Windows 系统一样，被删除的文件和目录总是缓存在回收站中。如果发现误删了文件，可以非常方便地从回收站中恢复文件。

但在使用命令行界面访问 Linux 系统时，使用 rm 等命令删除的文件，一经执行，便会立即释放文件的信息节点和数据块，回收站在此不起任何作用。如果误删了文件，仅当满足下列条件时，才能利用 debugfs 命令挽救删除的文件。

(1) 删除文件后，系统或用户没有在文件系统中创建任何新的文件（包括目录、符号链接文件等各种文件）。这个条件表示因删除文件释放的信息节点和数据块没有被再次分配使用。

(2) 删除文件后，系统或用户也没有在文件系统中扩充任何原有的文件。这个条件表示删除文件释放的数据块没有被再次分配使用。

如果满足上述两个条件，可以立即卸载文件系统，仿照下列步骤，利用 debugfs 命令恢复误删的文件。

(1) 利用 find\_free\_inode (ffi) 子命令，找出第一个空闲的信息节点。

(2) 利用 stat 子命令，检查信息节点数据结构中的 BLOCKS 字段。如果其中数据块指针的数据为空，说明当前的信息节点根本就没有用过。如果其中给出了数据块号——单个数据块号或一个数据块号范围，说明相应的信息节点是一个被删除的文件。

(3) 利用 cat 子命令，检查其中的数据是否为误删的文件。

(4) 如果是，利用 dump 子命令，把数据保存到系统中的指定文件。文件恢复过程结束。

(5) 否则，利用 seti 子命令，在信息节点位图中，把指定的信息节点标记为已分配。

(6) 从步骤 (1) 开始重复执行上述过程。

假定 1.44MB 软盘的 Ext2 文件系统中存在下列文件，在操作过程中不慎误删了其中的重要文件 info，如下所示。

```
$ sudo mount /dev/fd0 /mnt
[sudo] password for gqxing:
$ ls -l /mnt
总用量 17
drwxr-xr-x 2 gqxing gqxing 1024 2008-10-12 23:15 doc
-rw-r--r-- 1 gqxing gqxing 1729 2008-10-12 23:16 info
drwx----- 2 root   root  12288 2008-10-12 23:10 lost+found
-rw-r--r-- 1 gqxing gqxing 1945 2008-10-12 23:18 memo
$ rm /mnt/info
$
```

由于发现及时，没有对文件系统再做其他进一步的操作，故文件系统满足文件恢复的要求。此时可立即卸载文件系统，调用 debugfs 命令，执行文件的恢复过程。示例如下。

```
$ sudo umount /mnt
$ sudo debugfs -w /dev/fd0
debugfs 1.40.8 (13-Mar-2008)
debugfs: ls -l
  2  40755 (2)    1000    1000    1024 12-Oct-2008 23:09 .
  2  40755 (2)    1000    1000    1024 12-Oct-2008 23:09 .
  11 40700 (2)      0      0    12288 12-Oct-2008 23:10 lost+found
  17 40755 (2)    1000    1000    1024 12-Oct-2008 23:15 doc
  19 100644 (1)    1000    1000    1945 12-Oct-2008 23:18 memo
(END)
```

```

debugfs: ffi
Free inode found: 18
debugfs: stat <18>
Inode: 18 Type: regular Mode: 0644 Flags: 0x0 Generation: 1433589997
User: 1000 Group: 1000 Size: 1729
File ACL: 0 Directory ACL: 0
Links: 0 Blockcount: 4
Fragment: Address: 0 Number: 0 Size: 0
ctime: 0x48f1cae0 -- Sun Oct 12 23:16:08 2008
atime: 0x48f1c966 -- Sun Oct 12 23:16:41 2008
mtime: 0x48f1c95b -- Sun Oct 12 23:16:41 2008
ctime: 0x48f1cae0 -- Sun Oct 12 23:19:08 2008
BLOCKS:
(0-1):66-67
TOTAL: 2
(END)
debugfs: cat <18>
# This is a testing file for debugfs
#
# There is a lot of important data in this file
.....
debugfs: dump <18> rescued
debugfs: quit
$ ls -l rescued
-rw-r--r-- 1 gqxing gqxing 1729 2008-10-12 23:36 rescued
$
```

#### 17.6.4 应用举例 2——恢复误删的文件

利用 undel 命令恢复误删的文件，其过程实际上可能更简单。例如，假定在使用 rm 子命令时误删了文件 hostname，之前我们知道其信息节点号为 14（否则可利用 freei 等命令找出其信息节点），可以利用下列 undel 子命令。

```

debugfs: rm hostname

debugfs: undel <14> newdir
debugfs: ls -l newdir
    12  40755 (2)      0      0   1024  2-Oct-2008  23:19 .
    2  40755 (2)  1000  1000   1024  2-Oct-2008  23:19 ..
   14  100644 (1)      0      0       6  2-Oct-2008  23:18 <14>

debugfs: cat newdir/<14>
iscas
debugfs:
```

undel 子命令将会以信息节点号的形式“<inode>”命名恢复的文件。因此，在退出 debugfs 命令之后，可以安装文件系统，重新命名恢复的文件，如下所示。

```

debugfs: quit
$ sudo mount /dev/fd0 /mnt
[sudo] password for gqxing:
$ cd /mnt/newdir
$ ls -l
总用量 1
-rw-r--r-- 1 root root 6 2008-10-02 23:18 <14>
$ sudo mv '<14>' hostname
$ ls -l
总用量 1
-rw-r--r-- 1 root root 6 2008-10-02 23:18 hostname
$ cd
$ sudo umount /mnt
$
```

当删除的文件比较多，目录项的字符数超过一个目录现有的容量时，可采用另外一种文件恢复方式——即在使用 undel 子命令恢复文件时，不要指定目的目录。具体做法是首先使用下列 undel



子命令，逐一恢复每个文件的信息节点。

`undel <inode>`

然后再使用下列 `ln` 子命令，把每个文件逐一链接到指定的目录。

`ln <inode> directory`

最后再安装文件系统，重新命名恢复的所有文件，参见上述步骤。

采取上述方法的好处是，在恢复了所有的信息节点之后，再把文件连接到一个目录，即使目录项超出指定目录现有的存储容量，因而需要申请新的数据块时，也不会占用已恢复文件的数据块。但缺点是，需要针对恢复的文件，要逐一运行 `undel` 和 `ln` 子命令。

还有一种简单的方法是借助 `fsck` 命令，实现文件的快速恢复。其具体做法是，在运行 `undel` 子命令之后，修改文件系统超级块的状态标志，然后退出 `debugfs` 命令，再运行 `fsck` 命令。

假定当前调试的软盘文件系统中存在 `fstab`、`sudoers`、`passwd`、`group` 和 `shadow` 等 5 个重要文件，如下所示。

```
$ sudo debugfs -w /dev/fd0
debugfs: ls -l
 2 40755 (2) 1000 1000 1024 12-Oct-2008 17:19 .
 2 40755 (2) 1000 1000 1024 12-Oct-2008 17:19 ..
11 40700 (2) 0 0 12288 12-Oct-2008 17:19 lost+found
12 100644 (1) 1000 1000 555 12-Oct-2008 17:20 fstab
13 100440 (1) 0 0 470 12-Oct-2008 17:20 sudoers
14 100644 (1) 1000 1000 1945 12-Oct-2008 17:20 passwd
15 100644 (1) 1000 1000 1174 12-Oct-2008 17:20 group
16 100640 (1) 0 0 1272 12-Oct-2008 17:20 shadow
```

debugfs:

后来由于某种原因遭到误删。此时，可以利用 `debugfs` 的 `undel` 与 `set_super_value (ssv)` 子命令以及 `fsck` 等命令，恢复误删的文件，具体步骤如下。

```
debugfs: undel <12>
debugfs: undel <13>
debugfs: undel <14>
debugfs: undel <15>
debugfs: undel <16>
debugfs: ssv state 200
debugfs: quit
$ sudo fsck /dev/fd0
fsck 1.40.8 (13-Mar-2008)
e2fsck 1.40.8 (13-Mar-2008)
/dev/fd0 was not cleanly unmounted, check forced.
Pass 1: Checking inodes, blocks, and sizes
Pass 2: Checking directory structure
Pass 3: Checking directory connectivity
Pass 4: Checking reference counts
Unattached inode 12
Connect to /lost+found<y>? yes

Inode 12 ref count is 2, should be 1. Fix<y>? yes

Unattached inode 13
Connect to /lost+found<y>? yes

Inode 13 ref count is 2, should be 1. Fix<y>? yes

Unattached inode 14
Connect to /lost+found<y>? yes

Inode 14 ref count is 2, should be 1. Fix<y>? yes

Unattached inode 15
Connect to /lost+found<y>? yes
```

```
Inode 15 ref count is 2, should be 1. Fix<y>? yes
Unattached inode 16
Connect to /lost+found<y>? yes
Inode 16 ref count is 2, should be 1. Fix<y>? yes
Pass 5: Checking group summary information
/dev/fd0: ***** FILE SYSTEM WAS MODIFIED *****
/dev/fd0: 16/184 files (6.3% non-contiguous), 55/1440 blocks
$ sudo mount /dev/fd0 /mnt
$ sudo ls -l /mnt/lost+found
总用量 8
-rw-r--r-- 1 gqxing gqxing 555 2008-10-12 17:08 #12
-r--r---- 1 root root 470 2008-10-12 17:08 #13
-rw-r--r-- 1 gqxing gqxing 1945 2008-10-12 17:08 #14
-rw-r--r-- 1 gqxing gqxing 1174 2008-10-12 17:08 #15
-rw-r----- 1 root root 1272 2008-10-12 17:09 #16
$
```

最后，可以根据文件的内容，重新命名#12～#16这5个文件。注意，在上述步骤中，利用 set\_super\_value 命令修改超级块中文件系统的状态字段是至关重要的，否则 fsck 不会修复文件系统。

## 17.7 其他文件系统维护工具

### 17.7.1 dumpe2fs 命令

dumpe2fs 命令用于输出指定设备的文件系统或文件系统映像文件中的超级块和数据块组描述等信息，其语法格式简写如下。

```
dumpe2fs [-bfhixV] device
```

其中，“-b”选项表示输出文件系统保留的坏块。“-f”选项表示强制 dumpe2fs 命令显示文件系统信息，即使其中存在某些尚无法理解的文件系统特性标志。“-h”选项表示仅显示超级块中的信息，但禁止输出数据块组描述等信息。“-i”选项表示输出 e2image 命令生成的文件系统映像文件中的数据。“-x”选项表示以十六进制数值显示数据块组中的数据块信息。

下面是一个利用 dumpe2fs 命令显示 1.44MB 软盘中 Ext2 文件系统超级块信息的例子。

```
$ dumpe2fs -h /dev/fd0
dumpe2fs 1.40.8 (13-Mar-2008)
Filesystem volume name: <none>
Last mounted on: <not available>
Filesystem UUID: 291f0256-ae42-4bb4-9606-0649105250bd
Filesystem magic number: 0xEF53
...
Filesystem state: clean
Errors behavior: Continue
Filesystem OS type: Linux
Inode count: 184
Block count: 1440
Reserved block count: 72
Free blocks: 1380
Free inodes: 165
First block: 1
Block size: 1024
```



```
Blocks per group:          8192
Fragments per group:      8192
Inodes per group:          184
Inode blocks per group:    23
Filesystem created:        Sun Oct 12 17:07:31 2008
Last mount time:           Sun Oct 12 17:52:38 2008
Last write time:           Sun Oct 12 18:22:43 2008
...
First inode:                11
Inode size:                 128
...
$
```

## 17.7.2 e2image 命令

### 1. e2image 命令简介

e2image 命令用于抽取指定设备中 Ext2/Ext3 文件系统的重要元数据，把文件系统映像保存到一个文件中。保存的文件系统映像文件可以供 dumpe2fs 和 debugfs 命令使用“-i”选项进行考察，这有助于资深人员恢复严重受损的文件系统。e2image 的命令语法格式简写如下。

```
e2image [-rsI] device image-file
```

其中，“-r”选项表示创建一个原始文件系统的映像文件，“-I”选项用于抽取映像文件中的元数据，恢复指定设备中的文件系统。

如果指定的文件系统映像文件为减号“-”，则意味着把 e2image 命令的输出将送到标准输出，使 e2image 命令的输出能够经过管道递交另一个命令（如 bzip2 等压缩命令）处理。注意，仅当使用“-r”选项创建原始映像文件时才能利用管道机制，因为在创建常规映像文件的处理过程中需要随机地访问文件，而此举是管道机制不支持的。

在系统的管理与维护过程中，一个比较明智的做法是，定期地（如每周一次）创建系统中所有文件系统的映像文件，同时利用“fdisk -l”命令保存磁盘的分区布局。文件系统映像文件应当保存在其他文件系统或存储介质中，确保文件系统一旦遭受严重损坏，仍然能够获取映像文件数据，并据以恢复受损的文件系统。

Ext2 文件系统映像文件的大小主要取决于文件系统的大小以及其中存有多少个文件。对于一个典型的 10GB、拥有 1 200 000 个信息节点，存有 200 000 个文件的文件系统而言，其映像文件约为 35 MB；一个 4GB、拥有 550 000 个信息节点，其中存有 15 000 个文件的文件系统，其映像文件约为 3 MB；文件系统映像文件具有较大的压缩比，一个 32MB 的映像文件通常能够压缩到 3MB 或 4MB。下例中的文件系统基本上是一个空的文件系统，故能够压缩得更多。

```
$ e2image /dev/fd0 fdimage
e2image 1.40.8 (13-Mar-2008)
$ ls -l fdimage
-rw----- 1 gqxing gqxing 28672 2008-10-12 18:24 fdimage
$ bzip2 fdimage
$ ls -l fdimage*
-rw----- 1 gqxing gqxing 623 2008-10-12 18:24 fdimage.bz2
$
```

### 2. 使用映像文件恢复文件系统的元数据

利用“-I”选项，e2image 命令能够把映像文件中的元数据写到指定的存储设备。在紧急情况下，可以使川 e2image 命令恢复存储设备上的文件系统元数据。

警告：“-I”选项只能是在其他恢复尝试均告失败之后的无奈之举，不到万不得已，不应采用此法恢复文件系统。在创建映像文件之后，如果文件系统已经发生了变动，将会丢失其中的数据。

通常，在尝试使用其他工具修复文件系统之前，应首先创建一个完整的文件系统映像文件备份。

### 3. 原始文件系统映像文件

利用“-r”选项，e2image 命令将会创建一个原始文件系统的映像文件，而非常规的映像文件。原始映像文件与常规映像文件的差别主要有二：首先，原始映像文件中的文件系统元数据都会存放在适当的位置，以便 e2fsck、dump2fs 或 debugfs 等命令能够直接处理原始映像文件，故文件较大；其次，原始映像文件也包括间接地址数据块和目录文件的数据块，而常规的标准映像文件目前则不包括这样的数据。要生成原始映像文件，可以采用下列 e2image 命令。

```
$ e2image -r /dev/fd0 fdimage
e2image 1.40.8 (13-Mar-2008)
$ ls -l fdimage
-rw----- 1 gqxing gqxing 1474560 2008-10-12 18:26 fdimage
$
```

当需要向分析人员提交映像文件，分析和修复文件系统的问题时，可以采用下列 e2image 命令，生成压缩的映像文件。

```
$ e2image -r /dev/fd0 - | bzip2 > fdimage.bz2
e2image 1.40.8 (13-Mar-2008)
$ ls -l fdimage.bz2
-rw----- 1 gqxing gqxing 747 2008-10-12 18:28 fdimage.bz2
$
```

上述映像文件仅包含文件系统的元数据信息，而不包括任何数据块。但是，目录文件中的文件名仍会透露文件系统中的数据内容。为了防止任何信息泄密，可以使用“-s”选项，使 e2image 命令能够在生成映像文件之前遍历目录，清除目录文件中未用的任何目录项。但有一点需要注意，“-s”选项将会妨碍分析采用散列树形式建立目录索引的有关问题。

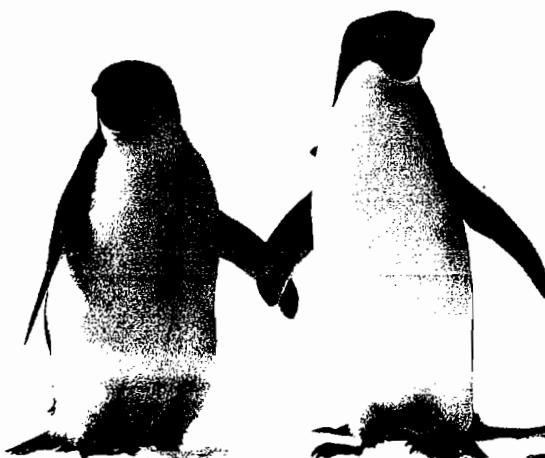


## 第18章

### TCP/IP 网络管理

网络与通信是操作系统的一个重要组成部分，Linux 系统提供了丰富的网络通信功能。本章主要介绍 IPv4 即传统的 TCP/IP，以及网络接口、网关和路由等的设置，最后讨论 TCP/IP 网络的管理与维护。其内容主要包括：

- TCP/IP 简介；
- 网络接口设置；
- 主机名字解析；
- 网络路由设置；
- 配置网络服务；
- 网络管理与维护。



# 18.1 TCP/IP 简介

## 18.1.1 TCP/IP 的层次结构

ISO 的 OSI 网络互连参考模型描述了一种理想的网络层次结构及相应的通信协议，但 TCP/IP 与 OSI 的层次模型并不完全吻合。TCP/IP 要么没有相应的结构层次，要么干脆把若干层合并为一层。表 18-1 展示了 TCP 与 OSI 模型的对照关系。

表 18-1 TCP/IP 层次结构与 OSI 层次模型对照表

OSI 分层	OSI 层次模型	TCP/IP 层次结构	TCP/IP 协议栈
7	应用层	应用层	Telnet、SSH、FTP、TFTP、SFTP、SCP、SMTP、DNS、NIS、LDAP、SNMP、NFS 等 套接字、端口
6	表示层		
5	会话层	传输层	
4	传输层		TCP、UDP
3	网络层	Internet 层 网络访问层	IP、ICMP、RIP
2	数据链路层		ARP、RARP IEEE 802.2、PPP、设备驱动程序
1	物理层		Ethernet、IEEE 802.3、FDDI、ATM、Token Ring 等

表 18-1 也给出了 TCP/IP 协议栈中每个协议层支持的协议，了解这些协议与协议层的对应关系，有助于加深对 TCP/IP 协议的理解，也有助于在应用过程中分析与隔离网络故障。

### 1. 网络访问层

在 TCP/IP 协议中，网络访问层涵盖了 OSI 层次模型的物理层、数据链路层和部分网络层的功能。网络访问层的主要功能如下。

- 利用地址解析协议 ARP (Address Resolution Protocol)，把 IP 地址映射为介质访问控制地址，即 MAC (Media Access Control) 地址，实现 IP 地址与 MAC 地址之间的转换。
- 把来自 Internet 层的数据报封装为网络上实际传输的数据帧，并增加数据帧的头信息。其中包括源和目的 MAC 地址和 CRC 循环冗余校验等字段。
- 利用设备驱动程序及实际的网络设备和传输介质，提供错误控制，以及按帧发送与接收数据的功能，以数据帧的形式及其错误控制机制，实现数据的传输。

### 2. Internet 层

Internet 层也称作 IP 层，其主要功能是接收和发送来自传输层或网络访问层的数据。Internet 层包括强有力的 IP 协议和 ICMP 协议。

#### (1) IP 协议 (Internet Protocol)。

在 TCP/IP 协议系列中，IP 协议及其有关的路由协议也许是最重要的协议。IP 支持下列功能。



- IP 地址——IP 地址是 IP 协议的一个重要组成部分，用于确定数据传输的源和目的主机，确定数据传输的路由。
- 负责网络访问层与传输层之间的数据传输——基于接收系统的 IP 地址，IP 协议负责确定分组数据到达目的系统必须经由的路径，实现主机与主机间的通信。
- 定义和封装数据报——数据报是 Internet 数据传输的基本单位，是 Internet 协议定义的分组数据格式。传输数据时，IP 协议层把来自传输层的分组数据进一步细分为较小的数据报（datagram），然后通过网络访问层实现数据的传输。
- 数据的分片与装配——如果分组数据太大，发送系统的 IP 协议层将会把数据报分成较小的数据片（fragment），而接收系统中的 IP 协议层则需要把数据片重组为初始的数据报。
- 利用路由信息协议 RIP 和 ICMP 路由发现（Router Discovery）协议等，收集、转发和维护路由表，为数据的传输提供路由服务。

#### (2) ICMP 协议（Internet Control Message Protocol）。

ICMP 协议的主要功能是控制、检测和报告网络错误，如下所示。

- 数据流控制——当分组数据到达得太快，接收系统来不及处理，或者因网络通信问题导致分组数据中途丢失时，目的主机或中间的路由器将会向发送主机回送“ICMP Source Quench Message”报文，通知发送主机暂时停止发送数据。
- 检测网络或主机的连接问题——因网络硬件或协议软件故障，导致分组数据无法到达目的系统时，ICMP 协议将会向发送分组数据的主机回送“Destination Unreachable Message”报文，表明无法联系远程主机。
- 重定向路由——通过 ICMP 重定向报文（Redirect Message），通知发送主机使用另外一个路由。
- 检测远程主机——通过发送 ICMP 回显报文（Echo Message），本地主机可以检测远程主机的网络协议是否已处于工作状态。当远程主机收到回显报文时，将向发送主机回送同样的分组数据。Linux 系统中的 ping 命令就是利用 ICMP 回显报文协议实现的。

### 3. 传输层

传输层的协议包括传输控制协议（Transmission Control Protocol, TCP）和用户数据报协议（User Datagram Protocol, UDP）。通过建立主机与主机之间的通信连接，接收数据后的确认，以及丢失分组数据后的重传等机制，TCP 确保分组数据能够准确无误地按顺序到达接收系统。这种类型的数据通信称作面向连接的、端到端的通信。TCP 能够提供可靠的、端到端的通信服务，而 UDP 只能提供非可靠的数据报服务。

#### (1) TCP 协议。

TCP 使应用程序能够实现主机到主机的通信。当收到应用层提供的数据流后，TCP 协议将会把数据流分为一系列 TCP 协议能够识别的数据段（segment），并在每个数据段之前增加一个头信息。头信息包含源和目的端口号、数据段的顺序号以及校验和等字段，以便目的主机能够正确地接收数据，并把数据传输到相应的应用程序中。

通过在源和目的主机之间建立端到端的连接，TCP 能够确保准确无误地把数据发送到目的主机，故通常把 TCP 称作“可靠的、面向连接的”协议。

## (2) UDP 协议。

UDP 协议提供数据报传输服务。UDP 协议并不负责建立和检验发送与接收主机之间的通信连接。因此，只有传输少量数据的应用程序才使用 UDP 协议。实际上，由于协议层的开销较少，在网络条件和接收主机性能比较好的情况下，使用 UDP 协议比使用 TCP 协议的效率要高。

### 4. 应用层

应用层定义任何用户均可使用的标准 Internet 服务和网络应用。这些服务通过传输层发送或接收数据。TCP/IP 应用层提供的服务和应用如下：

- 标准的 TCP/IP 服务，如文件传输协议 FTP、TFTP 和 Telnet 等；
- 名字服务 DNS 和 NIS 等；
- 目录服务 LDAP；
- 网络文件服务 NFS；
- 简单邮件传输协议 SMTP；
- 简单网络管理协议 SNMP。

## 18.1.2 TCP/IP 如何处理数据通信

当用户输入一个 TCP/IP 应用的命令时，将会触发一系列处理过程。用户的命令或数据将会穿过本地系统的 TCP/IP 协议栈，经由网络介质到达远程系统的相应协议层。发送系统的每一层协议将会在初始的数据中增加协议控制头信息，而接收系统的每个协议层将会逐层剥离相应的协议控制头信息。两个系统之间的同一层协议按照协议的约定进行交互，实现数据的封装、传输和剥离，如图 18-1 所示。

### 1. 数据封装与 TCP/IP 协议栈

分组是网络传输的基本数据单位。一个分组数据是由发送和接收主机地址等头信息和实际数据组成的。当分组数据经过 TCP/IP 协议栈时，每个协议层或者增加或者删除必要的协议字段。在 TCP/IP 协议中，通常把发送系统每个协议层增加头信息的处理过程称作封装。注意，每个协议层所用数据单位的术语并不完全相同（在不致引起混淆的情况下，通常统称为分组数据），如图 18-2 所示。

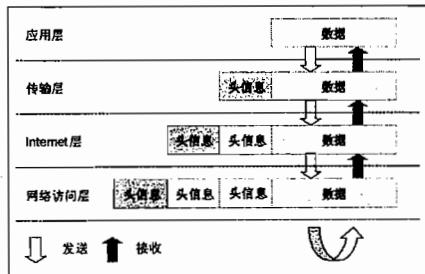


图 18-1 分组数据经由 TCP/IP 协议栈的处理过程

	TCP	UDP
应用层	数据流	报文
传输层	段	分组
Internet 层	数据报	数据报
网络访问层	帧	帧

图 18-2 数据结构及术语

下面简单说明分组数据经由 TCP/IP 处理的生命周期，即数据的封装、传输和剥离等处理过程。当用户输入一个 TCP/IP 命令或发送一个数据时，标志分组数据生命周期的开始。当接收系统的相应应用收到分组数据时，分组数据的生命周期即告结束。

### 2. 应用层：数据通信的发起者

如上所述，当用户输入一个必须访问远程主机的 TCP/IP 命令时，即启动了 TCP/IP 的分层处



理过程。应用层协议依据选用的传输层协议是 TCP 或 UDP，把数据转换为数据流（stream）或报文（message）格式。

假定用户输入一个 telnet 命令，准备注册到远程主机系统。telnet 使用的底层协议是传输层的 TCP，而 TCP 期望收到的数据是由一系列字节组成的数据流，其中包含 telnet 命令的所有信息。因此，telnet 需要以 TCP 数据流的形式向传输层协议发送数据。

### 3. 传输层：数据封装的起始点

当数据到达传输层时，传输层协议将启动一个执行数据封装的进程，把应用数据封装成若干传输层能够处理的数据单位。然后，传输层协议将在接收与发送应用之间建立一个由传输端口号标识的虚拟数据流。这个端口号用于标识一个内存位置，以便接收或发送数据。另外，传输协议层亦提供其他服务，如数据传输可靠性保障措施等，最终取决于采用 TCP 还是采用 UDP 处理数据传输。

#### (1) TCP 数据段。

TCP 经常称作“面向连接”的协议。TCP 提供的保障措施能够确保数据成功地传输到接收系统。如图 18-1 所示，TCP 把来自应用层命令的数据流分成若干数据段（segment），然后在每个数据段前面增加一个头信息。数据段的头信息包含发送和接收的端口号、数据段的顺序号及校验和等字段。发送和接收主机中的 TCP 协议均使用这个校验和，确定数据的传输是否有误。

#### (2) 建立 TCP 连接。

TCP 采用一个传输控制块（Transmission Control Block, TCB），维护每个活动连接的状态信息，用以确定接收系统是否能够接收数据。当发送主机的 TCP 协议准备建立连接时，TCP 将向接收主机的 TCP 协议发送一个称作 SYN 的数据段。如果接收主机已经就绪，接收 TCP 协议将返回一个称作 ACK 的数据段，确认已经成功地收到了连接请求数据段。

在收到接收主机的 ACK 确认信息之后，发送主机中的 TCP 协议也需要发送一个 ACK 数据段，然后才能开始实际的数据发送。这种交互传输控制信息的方式称作“三次握手(three-way handshake)”。

#### (3) UDP 分组。

UDP 是一种无连接的协议。与 TCP 不同，UDP 并不检查数据是否已经到达接收主机。而且，UDP 协议把来自应用层的报文分成 UDP 分组（packet），然后在每个分组前面增加一个头信息。分组的头信息包含发送和接收的端口号、分组数据长度以及校验和等字段。

UDP 发送进程只管尝试向接收主机的 UDP 进程发送分组数据，由应用层确定 UDP 接收进程是否已经确实收到了分组数据。UDP 不要求接收方确认，也不事先使用上述握手方式建立连接。

### 4. Internet 层：传输分组数据的准备阶段

传输层的 TCP 和 UDP 把其分装的数据段或分组数据传输到 Internet 层之后，由 Internet 层的 IP（或 ICMP）协议处理收到的数据段或分组数据。在开始传输之前，IP 协议首先把数据再进一步细分为 IP 数据报，然后，IP 协议开始确定数据报的 IP 地址，以便把数据传输到接收主机。

在实际传输之前，IP 协议将在 TCP 或 UDP 协议增加的数据段或分组头信息之前再增加一个 IP 头信息。IP 头信息包括发送和接收主机的 IP 地址、数据报的长度以及数据报的顺序号等。注意，如果数据报的大小超过了网络允许的字节数，IP 协议层还需要把数据报分解为数据片（fragment）。

### 5. 网络访问层：数据分帧以及按帧传输数据

在收到 IP 数据报之后，数据链路层协议（如 PPP）将会把数据报组装成数据帧（frame），并增加第 3 个头信息。数据帧的头信息包含由 IP 地址转换成的 MAC 地址（源和目的），以及

CRC 循环冗余校验字段。经过网络介质传输之后，接收主机可以使用这个校验字段检查传输后的数据帧是否存在错误。最后，数据链路层把组装好的数据帧传输到物理层。

当发送主机的物理层收到数据帧时，物理层将会依据目的 MAC 地址，通过网络传输介质，把数据帧发送到接收系统的物理层硬件设备中。

### 6. 接收主机对分组数据的处理过程

当分组数据到达接收主机时，分组数据将以相反的顺序逐层穿越接收主机的 TCP/IP 协议栈，如图 18-1 所示。而且，接收主机中的每一个协议层将会剥离发送主机对等层依次加到分组数据前部的头信息。其详细处理过程如下。

(1) 在收到帧格式的分组数据时，网络访问层将会计算帧格式数据的 CRC。当确认数据帧的 CRC 正确无误时，网络访问层将会剥离数据帧的头信息，包括 CRC。然后利用反向地址解析协议 RARP（Reverse Address Resolution Protocol），把 MAC 地址映射为 IP 地址，实现 MAC 地址与 IP 地址之间的转换。最后把剥离后的数据发送到 Internet 层。

(2) Internet 层读取头信息中的信息，确认传输的目的系统无误之后，Internet 层还需要确定接收的数据是否为数据片。如果确乎如此，IP 协议层需要把数据片组装成最初的数据报。最后再剥离 IP 头信息，把数据段或分组数据上传到传输层协议。

(3) 传输层协议（TCP 或 UDP）读取头信息，确定应当由哪一个应用层协议接收这一数据。然后，TCP 或 UDP 协议层将剥离相应的传输层头信息，把数据流或报文发送到相应的目的应用中。

## 18.2 网络接口设置

本节将以以太网和 ADSL/PPPoE 网络为例，说明怎样设置 TCP/IP。

### 18.2.1 以太网络设置

在 Linux 系统中，网络接口的参数设置通常都是利用配置文件实现的。Ubuntu Linux 系统采用 /etc/network/interfaces 配置文件定义每一个网络接口，因此，所有的网络接口参数均位于 interfaces 配置文件中。系统启动脚本 /etc/init.d/networking 利用 ifup 和 ifdown 命令，依据配置文件中的定义，启动和关闭系统的网络功能。

要设置或修改网络接口参数，可以使用编辑器，直接编辑网络接口的配置文件，也可以利用 GNOME 图形界面设置网络。

#### 1. 命令行方式直接编辑 interfaces 文件

interfaces 配置文件定义的网络接口参数可由若干“iface”、“mapping”、“auto”或“allow-”节组成，示例如下。

```
auto lo
allow-hotplug eth1
iface lo inet loopback

mapping eth0
script /usr/local/sbin/map-scheme
map HOME eth0-home
map WORK eth0-work
```



```
iface eth0-home inet static
    address 192.168.1.1
    netmask 255.255.255.0
    up flush-mail
```

```
iface eth0-work inet dhcp
iface eth1 inet dhcp
```

其中，网络接口前的关键字 `auto` 表示，当启动脚本/etc/init.d/networking 采用 “`-a`” 选项运行 `ifup` 命令时，系统将会自动启用相应的网络接口。网络接口紧随 `auto` 关键字之后，位于同一行上。例如，“`auto lo eth0`” 表示在启动系统时自动配置并启用环回接口以及第一个以太网络接口。当配置文件包含多个 `auto` 节时，`ifup` 命令将会按照列举的顺序，依次启用每一个网络接口。

关键字 `allow-auto` 和 `allow-hotplug` 等也表示自动启用相应的网络接口，但在执行诸如 “`ifup--allow-hotplug eth0 eth1`” 形式的命令时，只能启用 “`allow-hotplug`” 后面列举的网络接口，如 `eth0` 或 `eth1`。注意，“`allow-auto`” 与 “`auto`” 的意义是相同的。

`mapping` 用于定义网络的物理接口名与逻辑接口名的映射关系。`mapping` 节的第一行由 `mapping` 和一个 Shell 模式组成，每个 `mapping` 节必须包含一个脚本定义，指定的脚本以物理网络接口名为参数。如果存在，随后的映射关系通过标准输入提交脚本。最终，脚本必须通过标准输出，输出一个表示网络逻辑接口名的字符串。

`iface` 节用于定义一个逻辑网络接口名（如果 `mapping` 节不存在，逻辑接口名等同于物理接口名），其语法格式如下。

```
iface ifname inet-address-family method
```

其中，“`ifname`” 表示逻辑网络接口名。“`inet-address-family`” 是网络地址类型，表示支持的网络协议，如 `inet`(IPv4)、`inet6`(IPv6) 及 `iipx` 等。“`method`” 表示网络接口的配置方法，如 `loopback` 方法（用于定义 IPv4 环回接口）、`static` 方法、`dhcp` 方法、`bootp` 方法（用于定义 BOOTP 服务器的 IP 地址）及 `ppp` 方法等。

对于 IPv4 网络协议，如果需要，可在随后的附加行中定义 IP 地址、子网掩码以及广播地址等参数。此外，还可以增加各种接口选项，表示启用还是禁用，以及如何启用或禁用相应的网络接口。例如，`up command`、`pre-up command`、`post-up command`、`down command`、`pre-down command` 以及 `post-down command` 分别表示在网络接口启用时、启用前后、禁用时，以及禁用前后执行指定的命令（详见/usr/share/doc/ifupdown/examples/network-interfaces.gz 文件给出的各种设置实例）。

在 IPv4 网络协议中，可以采用各种方法配置网络接口。其中，`static` 方法适用于静态分配网络接口的 IP 地址等参数，如 IP 地址、子网掩码、广播地址、网络地址、默认的网关、与网关的路由距离以及 MTU 大小等：

- ❑ `address address` # IP 地址
- ❑ `netmask netmask` # 子网掩码
- ❑ `broadcast broadcast_address` # 广播地址
- ❑ `network network_address` # 网络地址
- ❑ `gateway address` # 网关地址
- ❑ `metric metric` # 路由距离
- ❑ `pointopoint address` # PPP 协议 IP 地址
- ❑ `media type` # 物理网络类型

- `hwaddress class address` # 网络接口的 MAC 地址
- `mtu size` # MTU 的大小

例如，假定系统只配有一个以太网络接口，要采用 static 方法设置网路接口，interfaces 文件可以配置如下：

```
$ cat /etc/network/interfaces
auto lo
iface lo inet loopback

auto eth0
iface eth0 inet static
    address 192.168.90.100
    netmask 255.255.255.0
    gateway 192.168.90.1
$
```

dhcp 方法适用于存在 DHCP 服务器的网络环境，系统可利用 dhclient、pump、udhcpc 或 dhcpcd 等工具获取动态 IP 地址、主机名以及 DNS 服务器等。每个工具通常都有自己的配置文件，例如，dhclient 使用的配置文件是 /etc/dhcp2/dhclient.conf。

Ubuntu Linux 系统主要采用 dhclient 实现 IP 地址的自动配置。动态主机配置协议（DHCP）的客户端软件 dhclient 能够利用 DHCP 或 BOOTP 协议，联系 DHCP 服务器，获取 IP 地址等网络接口参数，配置系统中的任何网络接口。当 DHCP 或 BOOTP 网络协议运行失败时，还能够静态地分配 IP 地址。

要在办公室环境中采用 dhcp 方法设置网路接口，interfaces 文件可以配置如下。

```
$ cat /etc/network/interfaces
auto lo
iface lo inet loopback

auto eth0
iface eth0 inet dhcp
$
```

ppp 方法采用 pon/poff 命令配置 PPP 网络接口。在 ppp 方法中，唯一可用的选项为“provider name”。其中，name 是 ISP 的名字（取自 /etc/ppp/peers 目录中的文件），具体的配置步骤，可以参见 18.2.2 小节的讨论。

## 2. 在 GNOME 桌面中设置网络接口

要利用 GNOME 图形界面设置网络接口参数，可在 GNOME 桌面中选择“系统→系统管理→网络”，进入图 18-3 所示的网络设置对话框。

之前，首先需要单击“解锁”按钮，在出现的“认证”对话框中输入 sudo 密码，然后单击“认证”按钮。在“网络设置”对话框中，可使用鼠标选择“有线连接”，然后单击“属性”按钮（见图 18-3）。在“eth0 属性”对话框“配置”栏目的下拉菜单中选择“自动配置（DHCP）”，然后单击“确定”按钮，即可完成动态 IP 地址的设置（如图 18-4 所示）。如果选择“静态 IP 地址”，还需要输入“IP 地址”、“子网掩码”及“网关地址”等参数，以便提供 interfaces 配置文件所需的各种网络参数，然后单击“确定”按钮。

此外，也可以在“网络设置”对话框中选择“DNS”或“主机”标签，分别设置 DNS 服务器地址和 /etc/hosts 文件。

在完成上述配置过程之后，可在“网络设置”对话框中单击“关闭”按钮，保存网络参数设置，使系统在 /etc/network 目录中生成一个新的网络接口配置文件 interfaces。

不管是直接编辑配置文件，还是采用 GNOME 桌面的图形界面设置方式，若想让网络接口参数的配置立即生效，都需要使用下列命令，重新启动 /etc/init.d/networking 脚本。或者，在系统启



动过程中，利用/etc/rcS.d 目录中的相应启动脚本（S40networking），按照 interfaces 配置文件的定义设置网络接口。



图 18-3 网络设置对话框

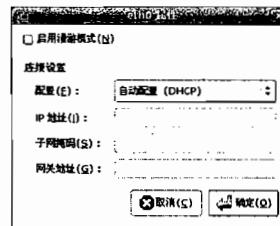


图 18-4 eth0 属性对话框

```
$ sudo /etc/init.d/networking restart
```

### 3. 网络接口基本配置命令

/etc/init.d/networking 脚本利用 ifup（或 ifdown）命令和/etc/network/interfaces 文件，配置每个网络接口，而 ifup（或 ifdown）命令底层采用的基本工具主要是 ip、ifconfig 和 route 等命令。

ip 命令是 Linux 系统中的一个功能非常强大的工具软件，不仅可用于显示、设置网络接口设备，还可用于显示和设置路由等，其语法格式简写如下。

```
ip [ OPTIONS ] OBJECT { COMMAND | help } parameters
OPTIONS := { -s[statistics] | -f[amily] { inet | inet6 ... } ... }
OBJECT := { link | addr | route ... }
COMMAND := { set | show | add | del ... }
```

为了便于理解，根据 ip 命令的功能，可以把 ip 命令分解为 link（用于设置链路层的网络参数）、addr（用于设置网络层的参数，如 IP 地址等）和 route（用于设置网络路由）3 种常用类型的命令，如下所示。

```
ip link show [ DEVICE ]
ip link set DEVICE { up | down | arp { on | off } |
    dynamic { on | off } | multicast { on | off } |
    address ADDR | broadcast ADDR | mtu MTU }

ip addr show [ dev DEVICE ]
ip addr { add | del } ADDR dev DEVICE [ broadcast ADDR ]
```

```
ip route show [ ROUTE ]
ip route { add | del | change | append | replace | monitor } ROUTE
```

其中，“-s” 选项用于显示各种统计数据与时间信息。“-f” 选项用于指定协议类型，如 inet（IPv4）或 inet6（IPv6）等。DEVICE 表示网络接口的设备文件名。ADDR 是网络接口的 MAC 地址（“ip link” 命令）或分配给网络接口的 IP 地址（“ip addr” 命令）。如果需要指定非标准的子网掩码，可在 IP 地址后附加一个斜线字符 “/”，然后给出子网的长度（二进制数值的位数）。ROUTE 表示路由项。parameters 是 ip 命令支持的各种参数。

例如，要查询一个网络接口物理设备的工作状态，可以使用下列 ip 命令。

```
$ ip link show eth0
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast qlen 1000
    link/ether 00:40:b8:00:8e:52 brd ff:ff:ff:ff:ff:ff
```

上述输出信息中的 up 表示网络接口已经启用。如果尚未启用（其标志为 down），可以使用下列 ip 命令，启用指定的网络接口设备。

```
$ sudo ip link set eth0 up
$
```

要设置并启用一个网络接口设备，可以使用下列 ip 命令。

```
$ sudo ip addr add 192.168.90.101 dev eth0
$
```

要查询一个网络接口设备的工作状态，可以使用下列 ip 命令。

```
$ ip addr show dev eth0
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast qlen 1000
    link/ether 00:40:b8:00:8e:52 brd ff:ff:ff:ff:ff:ff
        inet 172.16.3.25/24 brd 172.16.3.255 scope global eth0
            inet6 fe80::2a0:b8ff:fe00:8e52/64 scope link
                valid_lft forever preferred_lft forever
$
```

要查询网络的路由信息，可以使用下列 ip 命令。

```
$ ip route show
172.16.3.0/24 dev eth0 proto kernel scope link src 172.16.3.25
169.254.0.0/16 dev eth0 scope link metric 1000
default via 172.16.3.254 dev eth0
$
```

作为一个常规的网络管理工具，Linux 系统仍然保留 ifconfig 命令。ifconfig 命令主要用于配置网络接口的各种参数，包括 IP 地址、广播地址和网络掩码等。如果愿意，也可以在系统的生成过程中使用 ifconfig 命令配置系统中每个网络接口的 IP 地址。为了测试或维护网络接口，可以随时使用 ifconfig 命令，显示、修改或重新设置网络接口的 IP 地址等参数。注意，只有超级用户才能修改网络接口的配置参数。

ifconfig 命令的一般语法格式如下。

```
ifconfig [-a] [interface]
ifconfig interface [addr_family] [address] [parameters]
```

其中，interface 是网络接口的设备文件名。addr\_family 表示地址类型，如 inet(IPv4) 或 inet6(IPv6) 等。address 是分配给网络接口的 IP 地址。parameters 是 ifconfig 命令支持的各种参数，如表 18-2 所示。

表 18-2

ifconfig 命令支持的部分参数

参数	简单说明
up	启用网络接口，使得用户程序能够通过相应的网络接口访问 TCP/IP 应用和服务。在利用 ifconfig 命令为网络接口分配一个 IP 地址时，也蕴涵着启用相应的网络接口。当使用这个参数启用一个网络接口时，也会把相应的网络接口设置为 UP 和 RUNNING 状态。在使用 down 参数临时关闭一个网络接口之后，也可以使用这个参数重新启用网络接口
down	关闭网络接口，禁止任何程序通过相应的网络接口访问 TCP/IP 应用和服务，同时把网络接口标记为 DOWN 状态。注意，设置这个参数时也会自动地删除使用这个网络接口的所有路由表项
arp	默认情况下，启用地址解析协议（Address Resolution Protocol，ARP）实现网络地址与物理地址之间的映射，以便检测主机网络接口的物理地址。如果禁用 ARP，ifconfig 命令的输出信息中将会存在一个 NOARP 标志
-arp	禁止使用 ARP 地址解析协议
netmask mask	指定网络接口使用的子网掩码。子网掩码是一个 32 位的二进制数值，其中 1 表示网络地址部分，0 表示主机地址部分。其表示方法可以采用 8 个十六进制的数值，加上 0x 前缀；也可以采用 4 组十进制的数值，中间加句点“.” 分隔符。注意，mask 至少应当包括标准的网络地址部分
broadcast addr	指定网络的广播地址。默认的广播地址是网号不变，主机地址部分全为 1 的 IP 地址。如果设置了广播地址，同时也会设置网络接口的 BROADCAST 标志
pointopoint addr	启用点对点连接模式，这意味着把相应的网络接口直接连接到另一个主机的网络接口。pointopoint 后面给出的是链路另一端的 IP 地址。在启用点对点连接模式的同时也设置网络接口的 IFF_POINTOPOINT 状态标志



续表

参 数	简 单 说 明
<code>mtu n</code>	使用指定的数值，设置网络接口的最大传输单位（Maximum Transmission Unit, MTU），即一次数据传输最多能够处理的字符数量，这个数值也限制了分组数据的最大尺寸。不同类型的网络接口，MTU 的上限值也不相同。例如，Ethernet 默认的 MTU 的数值为 1500
<code>metric number</code>	这个参数用于设置路由表项的度量值。路由信息协议（Routing Information Protocol, RIP）使用这个度量值建立路由表。ifconfig 命令使用的默认度量值为 0。如果没有启用 RIP 服务进程（如 ripd），根本就不需要设置这个参数；即使启用了 RIP 服务进程，通常也不必关心这个度量值的设置
<code>promisc</code>	使用这个参数可以把相应的网络接口置于混杂模式。在一个广播模式的网络中，这种混杂方式使得网络接口能够收到所有的分组数据，而不管分组数据的最终目的是不是网络接口所在的主机。这一技术使网络管理员能够利用分组数据过滤器或探测仪分析网络数据，从中找出网络问题。tcpdump 等工具就是在此基础上实现的。这个参数也会同时设置网络接口的 PROMISC 状态标志
<code>txqueuelen length</code>	设置传输队列的长度

在第一种语法格式中，“-a”选项用于显示系统中配置的所有网络接口。如果未提供任何选项，ifconfig 命令只显示当前活动的网络接口的状态；如果指定了网络接口，ifconfig 命令仅仅显示给定网络接口的状态。

例如，要手工设置主机系统中的以太网接口 eth0，可以使用下列 ifconfig 命令。

```
$ sudo ifconfig eth0 192.168.90.100  
$
```

要检查刚才的设置是否已经生效，可以使用下列命令。

```
$ ifconfig eth0  
eth0      Link encap:Ethernet HWaddr 00:40:b8:00:8e:52  
          inet addr:192.168.90.100 Bcast:192.168.90.255 Mask:255.255.255.0  
          UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1  
          RX packets:23258 errors:7 dropped:7 overruns:7 frame:0  
          TX packets:22626 errors:0 dropped:0 overruns:0 carrier:0  
          collisions:0 txqueuelen:1000  
          RX bytes:33893537 (32.3 MB)  TX bytes:1635376 (1.5 MB)  
          Interrupt:11 Base address:0x2400  
$
```

## 18.2.2 ADSL 网络连接

Ubuntu Linux 系统也支持无线网络和 ADSL 拨号网络连接。为了配置基于 PPPoE 协议的拨号网络功能，利用 ADSL 访问 Internet，可以采用 pppoeconf 命令，在一系列对话框的引导下，完成 ADSL 网络的设置。

具体步骤是，首先运行 pppoeconf 命令，当其找到以太网接口之后，可以使用制表键选中“是”按钮，按下 Enter 键，参见图 18-5。pppoeconf 命令使用/etc/ppp/peers/dsl-provider 文件存储 ADSL 网络配置信息。如果原有的数据需要保存，可在此前进行备份，否则选中“是”按钮继续，如图 18-6 所示。

在弹出的“经典模式”对话框中选择“是”按钮，按 Enter 键继续，如图 18-7 所示。在图 18-8 的对话框中输入连接 ADSL 网络的注册用户名，单击“确定”按钮继续。

在图 18-9 所示的对话框中输入连接 ADSL 网络的密码，选中“确定”按钮，按 Enter 键继续。在图 18-10 所示的对话框中，使用 ISP 提供的 DNS，选中“确定”按钮，按 Enter 键继续。

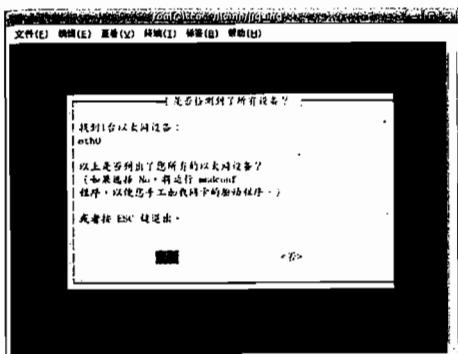


图 18-5 以太网接口检测对话框

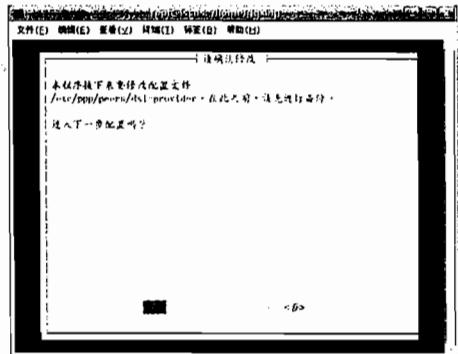


图 18-6 备份提示对话框

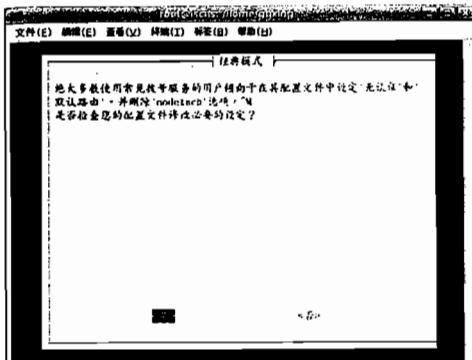


图 18-7 经典模式对话框



图 18-8 输入用户名对话框



图 18-9 输入密码对话框

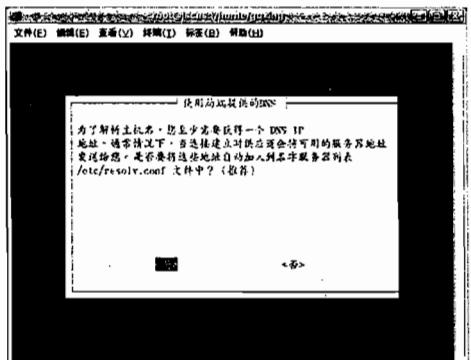


图 18-10 DNS 确认对话框

在图 18-11 所示的对话框中选择“是”按钮，按 Enter 键继续。在“完成”对话框中，如果想要在每次启动系统时都建立 PPPoE 连接，利用提供的用户名和密码，经过 ASDL 调制解调器，连接 Internet，选择“是”按钮，按 Enter 键继续，参见图 18-12 所示。

此时，ppoeconf 命令提示用户可以使用“pon dsl-provider”建立网络连接，使用 poff 命令关闭网络连接。选择“是”按钮将会启动 PPPoE 网络，参见图 18-13。启动 PPPoE 网络之后，可以使用 plog 命令查询网络连接的状态，或者用“ifconfig ppp0”命令查询网络接口的状态信息，参见图 18-14。

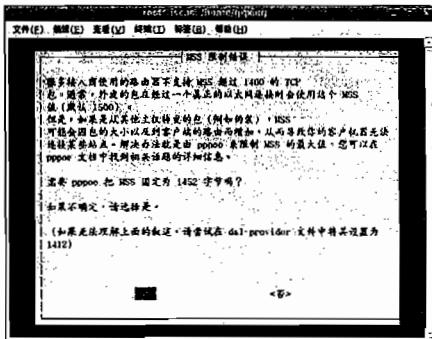


图 18-11 MSS 限制对话框



图 18-12 设置完成对话框

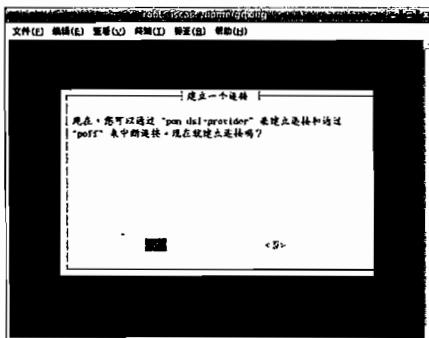


图 18-13 建立连接对话框



图 18-14 连接完成对话框

在图 18-15 所示的界面中，plog 和 “ifconfig ppp0” 命令的运行结果表示已经成功地建立了 PPPoE 网络连接，此后即可使用浏览器访问 Internet 了。

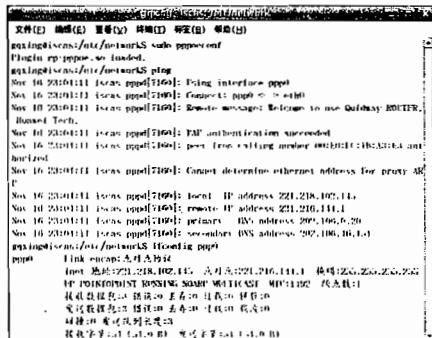


图 18-15 网络连接验证界面

在使用 pppoeconf 命令设置 ADSL 连接时，系统将会在 /etc/ppp/peers 目录中自动创建两个配置文件——dsl-provider 和 ppp0，其内容分别如下。

```
$ cat /etc/ppp/peers/dsl-provider
# Minimalistic default options file for DSL/PPPoE connections

noipdefault
defaultroute
replacedefaultroute
```

```

hide-password
#lcp-echo-interval 30
#lcp-echo-failure 4
noauth
persist
#mtu 1492
#persist
#maxfail 0
#holdoff 20
plugin rp-pppoe.so eth0      # 使用以太网接口连接 ADSL 调制解调器
usepeerdns
user "100001234567"
$ cat /etc/ppp/peers/ppp0
noipdefault
replacedefaultroute
hide-password
persist
usepeerdns

plugin rp-pppoe.so eth0      # 同上
user "100001234567"          # 同上
$
```

而密码（包括注册用户名）等信息则存储在/etc/ppp/chap-secrets 和/etc/ppp/pap-secrets 文件中，示例如下。

```

$ cat /etc/ppp/chap-secrets
# Secrets for authentication using CHAP
# client server secret           IP addresses

"username100001234567" * "alb2c3d4"

"100001234567" * "alb2c3d4"
$
```

在笔者配有一个以太网卡的系统中，经过上述配置步骤之后，其生成的 interfaces 文件以及 resolv.conf 文件内容如下。

```

$ cat /etc/network/interfaces
auto lo
iface lo inet loopback

iface ppp0 inet ppp
provider ppp0

auto dsl-provider
iface dsl-provider inet ppp
pre-up /sbin/ifconfig eth0 up # line maintained by pppoeconf
provider dsl-provider

auto eth0
iface eth0 inet manual
$ cat /etc/ppp/resolv.conf
nameserver 202.106.0.20
nameserver 202.106.46.151
$
```

## 18.3 主机名字解析

在系统之间利用 TCP/IP 协议进行网络通信时，其中主要以 IP 地址标识数据的来源和目的。



为了便于用户访问网络服务器，TCP/IP 提供名字服务，以便实现从主机名到 IP 地址的转换，其中常见的有/etc/hosts 文件、DNS 和 NIS 等。

当存在多种主机名字的解析方式时，究竟采用哪一种方法解析主机名字呢？为此，Ubuntu Linux 系统提供了一个/etc/hosts.conf 文件，用于定义主机名字解析的顺序。文件中的每一行均以一个关键字开始，后面跟有适当的配置信息。其中一个常用的关键字为 order，表示主机名字解析的顺序，后面跟有一个或多个名字解析方法，中间以逗号“,”作为分隔符。在 Ubuntu Linux 系统中，可以选用的的解析方法是 hosts 和 bind（即 DNS）等，示例如下。

```
$ cat /etc/hosts.conf
# The "order" line is only used by old versions of the C library.
order hosts,bind
multi on
$
```

上述文件内容表示，在解析主机名字时，首先使用/etc/hosts 文件。如果 hosts 文件不能满足要求，再使用 DNS 解析主机名字。

与 DNS 等名字解析方法相比，hosts 文件相对比较简单。本节主要说明怎样使用 hosts 文件实现名字解析，有关 DNS 的讨论，详见第 20 章。

hosts 文件的格式如下：

```
IP-address hostname [aliases...]
```

其中，IP-address 为网络接口的 IP 地址，hostname 是主机的名字，aliases 是主机的别名。例如，下面是 iscas 系统中的一个 hosts 文件例子。

```
$ cat /etc/hosts
127.0.0.1      localhost
169.254.78.100  iscas
169.254.78.101  sinosoft
169.254.78.56   winxp          # ISP 分配给 Windows XP 系统的 IP 地址
...
$
```

## 18.4 网络路由设置

当网络中的所有主机均位于同一网段时，通过 IP 地址或主机名，系统之间可以随意进行访问。当准备访问的主机位于不同的网段时，则无法直达，必须通过网关或路由器才能访问。因此，为了访问外部网络，必须考虑路由设置。

按照网络的规模、网络拓扑结构的稳定性，以及主机在网络中扮演的角色——提供路由功能的主机或普通主机，有两种设置路由的方式。

在一个大型的、拓扑结构经常发生变化的网络中，通常应当考虑使用动态路由。在一个小型或拓扑结构保持稳定不变的网络中，作为数据服务器或客户系统的普通主机，在完成本章前述的 TCP/IP 设置之后，必要时只需设置静态路由。考虑到路由的开销及其对计算机性能的影响，即使主机配有两个网络接口（如数据中心的数据库服务器），也以使用静态路由为最佳选择，动态路由可由专门的路由设备提供，或使用 quagga 等路由软件包实现。

在 Ubuntu Linux 系统中，可以采用 route 或 ip 等命令设置静态路由，其中最常用的基本工具仍然是 route 命令。本节主要介绍如何利用 route 命令设置静态路由。

**route** 命令的主要功能是维护网络路由表。利用 **route** 命令，可以增加、修改、删除、显示和监控路由表，而与增加或删除路由有关的 **route** 命令语法格式如下。

```
route [-v] [-A family] add | del [-net | -host | default] target
      [netmask Nm] [gw Gw] [[dev] if]
route [-v] [-A family] add default [gw Gw] [[dev] if]
```

其中，“-A family”表示地址类型，如 **inet** (IPv4) 或 **inet6** (IPv6) 等，“-host”表示把指定的目的地址强制解释为主机，“-net”表示把指定的目的地址强制解释为网络，**target** 表示要访问的目的主机或网络，**Nm** 表示子网掩码，**Gw** 表示通过哪一台主机 (IP 地址) 或接口转发访问外部网络的数据。

假定本地系统的网络接口 (其 IP 地址为

192.168.90.101)，连接到主机 (其内部网络接口的 IP 地址为 192.168.90.100)，主机的另外一个网络接口 (其 IP 地址为 153.78.26.145) 连接到外部网络。要使本地系统能够访问 153.78.26.0 网络中的任何主机，可以使用下列命令，增加静态路由 (参见图 18-16)。

```
$ sudo route add -net 153.78.26.0 netmask 255.255.255.0 gw 192.168.90.100
$
```

要利用路由器访问路由器能够达到的任何网络和主机，最简单的方法是利用下列命令，增加默认的路由。

```
$ sudo route add default gw 192.168.90.100 dev eth0 metric 1000
$
```

上述设置方式只是临时性的，一旦重新启动系统，路由的设置也就不复存在了。在 Ubuntu Linux 系统中，为了永久性地设置静态路由，可以利用 /etc/network/if-up.d/avahi-autoipd 配置文件实现。

在系统的启动过程中，/etc/init.d/networking 脚本将会运行 avahi-autoipd 脚本文件，根据其中的设置，利用 **route** 命令增加静态路由。因此，必要时可以把上述的 **route** 命令加到 avahi-autoipd 文件中。

注意，在设置网络路由时，应尽量使用 IP 地址或网络接口的设备名，而不要使用主机名或网络名。

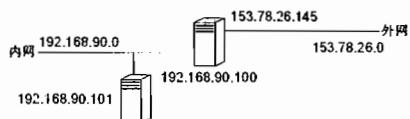


图 18-16 增加静态路由示意图

## 18.5 配置网络服务

当需要使用各种传统的网络服务时，需要用到 **inetd** 守护进程，设置其配置文件 /etc/inetd.conf。**inetd** 是一个超级的守护进程，负责集中管理许多标准的传统 Internet 服务，如 **telnet**、**ftp**、**finger** 和 **talk** 等，这些 Internet 服务的相应服务程序 **telnetd**、**ftpd**、**fingerd** 和 **talkd** 都是由 **inetd** 负责调度运行的。**inetd** 同时支持 UDP 和 TCP 协议。

**inetd** 是在系统启动过程中由 /etc/init.d/openbsd-inetd 脚本调度运行的。**inetd** 守护进程用于监听 Internet 套接字的连接请求。当从套接字中发现一个连接请求时，**inetd** 需要确定与套接字关联的相应网络服务，调用服务程序以响应连接请求。在服务程序结束运行之后，**inetd** 将继续监听其负责管理的套接字。采用一个 **inetd** 守护进程调用多个服务程序的目的是减少系统的负载。

在开始运行之后，**inetd** 首先会读取配置文件 /etc/inetd.conf 中的配置信息。当收到 SIGHUP 信号时，**inetd** 守护进程将会重新读取 **inetd.conf** 配置文件。因此，在修改 **inetd.conf** 文件，增加、删除或修改服务程序之后，需要使用 **kill** 命令，向 **inetd** 守护进程发送 SIGHUP 信号，或重新启动 **inetd** 守护进程，才能使修改后的配置文件生效。为此，可利用 **pgrep**、**pidof** 或 **ps** 等命令找出 **inetd** 的



进程 ID 号，再使用“kill -s SIGHUP”命令强制 inetc 重读其配置文件。

```
$ sudo kill -s SIGHUP `pidof inetc`  
$
```

也可以利用下列命令，重新启动 inetc 守护进程。

```
$ sudo /etc/init.d/openbsd-inetc restart  
* Restarting internet superserver inetc  
$ [ OK ]
```

inetd.conf 配置文件包含需由 inetc 守护进程调度运行的所有网络服务、服务程序的路径名及运行时使用的参数等，每个服务占用一行，其格式定义如下（字段之间以空格或制表符作为分隔符）。

```
srv_name socket_type protocol wait/nowait user srv_prog arguments
```

表 18-3 给出了 inetc.conf 配置文件每个字段的简单说明。

表 18-3

inetd.conf 文件及其说明

字段	简单说明
srv_name	/etc/inet/services 文件中定义的有效服务名。当指定的服务是一个基于 Sun-RPC 的服务时，这个字段应是 /etc/rpc 文件中定义的一个有效服务名，同时附加一个斜线“/”与 RPC 版本号后缀
socket_type	套接字类型可以是下列关键字之一： <input type="checkbox"/> stream 表示 STREAMS 套接字； <input type="checkbox"/> dgram 表示数据报套接字； <input type="checkbox"/> raw 表示原始套接字； <input type="checkbox"/> rdm 表示可靠的消息传递； <input type="checkbox"/> seqpacket 表示按序传输分组数据的套接字
protocol	/etc/inet/protocols 文件中定义的有效协议，如 tcp 或 udp 等。除了协议，还可以采用“protocol[,sndbuf=size][,rcvbuf=size]”的形式指定发送和接收套接字缓冲区的大小。例如： <input type="checkbox"/> tcp,rcvbuf=16384; <input type="checkbox"/> tcp,sndbuf=64k; <input type="checkbox"/> tcp,rcvbuf=64k,sndbuf=1m。 其中 k 和 m 分别表示以 1KB 和 1MB 为单位
wait/nowait	这个字段的有效值是 wait 或 nowait，说明当 inetc 调用相应的服务程序时是否需要等待服务进程运行终止。对于数据报类型的多线程服务进程，这个字段应为 nowait。对于数据报类型的单线程服务进程，这个字段应为 wait。采用 TCP 协议的服务程序通常应设为 nowait。但是，如果单个服务进程需要处理多个连接请求，则应设为 wait
user	指定一个用户名，表示以什么用户身份运行相应的服务程序。服务程序能够以普通用户的身份运行
srv_prog	指定服务程序的完整路径名，以便 inetc 能够正确地调用，使之执行请求的服务。如果请求的服务是由 inetc 本身提供的，这个字段应为 internal
arguments	如果调用服务程序时需要提供命令行参数，必须在这个字段中指定整个命令行（包括命令名及其所有参数）。如果服务是由 inetc 内部提供的，这个字段应为 internal

下面是一个 inetc.conf 文件的实例。

```
$ cat /etc/inetc.conf  
telnet stream tcp nowait telnetd /usr/sbin/tcpd /usr/sbin/in.telnetd  
$
```

上述文件说明，telnet 服务是基于 TCP 实现的，以 telnetd 的用户身份运行。in.telnetd 服务程序的完整路径名为 /usr/sbin/in.telnetd。一旦启用了 in.telnetd 服务程序，任何用户，只要有注册的账号，就可以利用 telnet 访问远程主机系统。

实际上，inetd.conf 文件反映的是主机系统当前支持的各种 Internet 服务。如果想要封锁某个 Internet 服务，删除相应的服务项或在服务项前增加一个注释符“#”即可。

因此，为了确保数据库服务器或业务主机的系统安全，严禁任何用户使用 telnet 远程访问主机系

统，可以利用编辑器，在 inetd.conf 文件的 telnet 定义前增加注释符“#”，禁止 inetd 守护进程调用 telnetd，封锁 telnet 服务，从而关闭主机系统的远程访问功能，防止 telnetd 响应用户的 telnet 访问请求。

## 18.6 网络管理与维护

对于新安装或首次接入网络的系统，如果根本就无法与网络中的其他任何主机进行通信，其主要问题也许是网络的设置不当。如果主机一直工作正常，突然出现网络故障，网络接口卡可能是主要原因。如果主机能够与同一网络中的任何主机通信，但无法联系外部网络中的主机，问题很可能出在路由器上：要么主机中的路由设置不当，要么路由器的工作不正常。当然，也可能是网络的硬件连接存在问题。

如果网络通信存在问题，可以采用下列方法，利用 ifconfig、netstat、ping 等命令检测网络的硬件连接和软件配置，确定问题的原因。原因不明的问题轻则影响网络的性能，次则丢失数据，重则无法通信，无法传输任何数据。

- (1) 使用 ping 命令界定问题的性质，是丢失数据包，还是网络根本就不通；
- (2) 使用 ifconfig 命令，检查网络接口的状态信息，确定网络接口工作是否正常；
- (3) 使用 netstat 命令检查网络接口、路由表和协议统计数据等网络状态信息；
- (4) 检查 hosts 文件，确保文件中的内容是正确的；
- (5) 使用 ps 等命令，确定相关的服务进程（如 sshd、vsftpd 或 telnetd 等）是否已经启动。

### 18.6.1 使用 ifconfig 命令维护网络接口

利用 ifconfig 命令，可以手工配置、修改或删除网络接口的 IP 地址，设置其他参数，获取网络接口的基本配置与状态信息，等等。一个简单的 ifconfig 查询（未指定任何选项和参数）能够给出：

- 系统配置的所有网络接口及其设备名；
- 网络接口的 MAC 地址及链路协议封装类型；
- 赋予网络接口的 IP 地址、广播地址及子网掩码等；
- 网络接口的初始化状态，如启用标志及 MTU 设置等；
- 网络接口当前的运行状态，如分组与字节数据的收发统计等。

如果指定了网络接口的设备名，ifconfig 命令将会给出特定网络接口的各种设置与状态信息。例如，下列 ifconfig 命令将会给出第一个以太网接口的相关信息：第一行是网络接口的链路协议封装类型（如 Ethernet）与硬件 MAC 地址等，第二行是 IP 地址（如 192.168.90.100）、广播地址（192.168.90.255）及子网掩码（255.255.255.0）等，第三行是网络接口的状态标志，如启用标志（如 UP 等）、广播标志（如 BROADCAST）及 MTU 设置（如 1500）等，从第 4 行开始是网络接口的各种统计数据，包括分组数据的收发统计及字节计数等。

```
$ ifconfig eth0
eth0      Link encap:Ethernet HWaddr 00:40:b8:00:8e:52
          inet addr:192.168.90.100  Bcast:192.168.90.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:23258  errors:7  dropped:7  overruns:7  frame:0
```



```
TX packets:22626 errors:0 dropped:0 overruns:0 carrier:0  
collisions:0 txqueuelen:1000  
RX bytes:33893537 (32.3 MB) TX bytes:1635376 (1.5 MB)  
Interrupt:11 Base address:0x2400
```

\$

表 18-4 给出了 ifconfig 命令输出字段的简单说明（有关 RX packets 与 TX packets 等分组数据统计信息字段的说明，详见表 18-6）。

表 18-4 ifconfig 命令的部分输出字段及其说明

输出字段	屏幕显示	简单说明
	eth0	网络接口的设备名
Link encap	Ethernet	链路封装协议
HWaddr	00:17:31:C9:22:92	网络接口的 MAC 地址
inet addr	192.168.90.100	网络接口的 IP 地址
Bcast	192.168.90.255	广播地址
Mask	255.255.255.0	子网掩码
	UP	网络接口状态标志。表示网络接口当前是否已经启用，或者是否已经初始化（UP 或 DOWN）
	BROADCAST	广播标志，表示网络接口是否支持广播
	RUNNING	传输标志，表示网络接口是否已经开始传输分组数据
	MULTICAST	多播标志，表示网络接口是否支持多播传输
MTU	1500	最大传输单位，表示网络接口的最大传输单位为 1500 个字节
Metric	1	度量值，这个数值主要供 RIP 建立网络路由表之用
RX bytes	23 366 (22.8 KiB)	接收数据字节统计
TX bytes	24 251 (23.6 KiB)	发送数据字节统计

当一个系统配置多个网络接口时，也可以使用“ifconfig -a”命令显示系统中配置的所有网络接口的状态信息，包括 IP 地址分配等，如下所示。

```
$ ifconfig -a  
eth0      Link encap:Ethernet HWaddr 00:40:b8:00:8e:52  
          inet addr:192.168.90.100 Bcast:192.168.90.255 Mask:255.255.255.0  
          UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1  
          RX packets:25690 errors:18 dropped:23 overruns:18 frame:0  
          TX packets:25187 errors:0 dropped:0 overruns:0 carrier:0  
          collisions:0 txqueuelen:1000  
          RX bytes:36811945 (35.1 MB) TX bytes:1828880 (1.7 MB)  
          Interrupt:11 Base address:0x2400  
  
lo        Link encap:Local Loopback  
          inet addr:127.0.0.1 Mask:255.0.0.0  
          UP LOOPBACK RUNNING MTU:16436 Metric:1  
          RX packets:177 errors:0 dropped:0 overruns:0 frame:0  
          TX packets:177 errors:0 dropped:0 overruns:0 carrier:0  
          collisions:0 txqueuelen:0
```

```
RX bytes:12941 (12.6 KB) TX bytes:12941 (12.6 KB)
$
```

## 18.6.2 使用 netstat 命令监控网络状态

netstat 命令可以给出各种网络状态信息，包括网络接口设置、IP 路由以及各种网络协议的分类统计数据。根据选用的命令选项，netstat 命令能够给出各种网络统计数据，其基本语法格式简写如下。

```
netstat [-s] [-i [ifname]] [-r] [-n] [-atuwp] [-c] [delay]
```

其中，“-s”选项表示按照协议分类显示各种统计数据。“-i”选项用于显示网络接口的状态信息。ifname 表示网络接口的设备名。“-r”选项用于显示核心路由表信息。“-n”选项表示只需显示 IP 地址，而不必把 IP 地址解析成相应的主机名或网络名。“-a”选项表示显示所有套接字的状态信息。“-t”选项表示仅显示 TCP 套接字的状态信息。“-u”选项表示仅显示 UDP 套接字的状态信息。“-w”选项表示显示原始套接字的状态信息。“-p”选项表示显示每个套接字所属程序的名字和进程 ID。“-c”选项表示每秒一次，连续显示选定的信息。delay 表示连续显示抽样统计数据的延迟时间或时间间隔（以秒为单位）。

### 1. 显示套接字的状态信息

默认情况下，netstat 命令能够显示各种协议（如 IPv4 或 UNIX 等）已经打开的所有套接字的状态信息，示例如下。

```
$ netstat
Active Internet connections (w/o servers)
Proto Recv-Q Send-Q Local Address          Foreign Address        State
Tcp      1      1 bogon:57060                jujube.canonical.co:www LAST_ACK
Tcp      1      1 bogon:57671                jp-in-f127.google.c:www LAST_ACK
Tcp      1      1 bogon:33881                avocado.canonical.c:www LAST_ACK
Tcp      0      28 bogon:52535               cg-in-f136.google:https LAST_ACK
Active UNIX domain sockets (w/o servers)
Proto RefCnt Flags     Type      State      I-Node    Path
unix    2      [ ]      DGRAM   6239      @/com/ubuntu/upstart
unix    2      [ ]      DGRAM   6379      @/org/kernel/udev/udevd
unix    2      [ ]      DGRAM   13992     @/org/freedesktop/hal/udev_event
unix    9      [ ]      DGRAM   12961     /dev/log
.....
$
```

表 18-5 解释了上述输出信息中每个字段的意义。

表 18-5 netstat 命令的部分输出字段及说明

输出字段	简单说明
<b>Active Internet connections</b>	
Proto	套接字连接采用的协议，如 tcp、udp 和 raw
Recv-Q	本地主机接收队列中用户程序尚未读取的数据字节计数
Send-Q	本地主机发送队列中远程主机尚未确认是否已经读取的数据字节计数
Local Address	套接字连接一端的本地主机地址与端口号。除非使用了“-n”选项，主机地址通常采用主机名或规范域名，端口号采用相应的服务名。地址的表示形式为“主机：端口号”或“网络：端口号”。其中，“主机”是源或目的主机的名字或 IP 地址（如/etc/hosts 文件中定义的主机名或 IP 地址），“网络”是源或目的网络的名字或网络地址（如/etc/networks 文件中定义的网络名或网络地址），“端口号”表示一个网络服务，既可以是/etc/services 文件中定义的端口号，也可以是相应的服务名（如 telnet）。在上述地址格式中，可以存在星号“*”通配符



续表

输出字段	简单说明	
Foreign Address	套接字连接一端的远程主机地址和端口号。用法同上	
	表示网络连接的状态。下面是可能出现的部分常见状态（注意，raw 模式与 UDP 协议不提供网络连接的状态信息，故这一列通常为空）	
	CLOSED	网络连接已经关闭
	CLOSE_WAIT	本地主机已经收到远程主机断开连接的请求，且已经给予确认，正在等待本地应用程序关闭连接
	CLOSING	在经过 LAST_ACK 和 TIME_WAIT 两个状态之后，本地与远程均进入 CLOSING 状态，开始正式关闭网络连接
	ESTABLISHED	网络连接已经建立，可以双向传输分组数据
State	FIN_WAIT1	本地主机已向远程主机发送断开连接的请求，正在等待远程主机的响应
	FIN_WAIT2	本地主机在向远程主机发出断开连接的请求之后，已经收到远程主机的确认
	LAST_ACK	本地主机在收到应用程序关闭连接的请求之后，再次向远程主机发出确认信息
	LISTEN	本地主机处于监听状态，正在监听来自远程主机的连接请求
	SYN_RECV	本地主机已经收到远程主机的连接请求，且已经给予确认，现正等待远程主机的最终确认
	SYN_SENT	本地主机已经发出连接请求，正在尝试建立套接字连接，并等待远程主机的确认
	TIME_WAIT	本地主机在第二次收到远程主机断开连接的确认之后，接着向远程主机发出最终的确认，然后开始着手关闭网络连接
Active UNIX domain Sockets		
Proto	套接字采用的协议（通常为 unix）	
RefCnt	引用计数，表示加接到相应套接字的进程数量	
Flags	标志字段。可能出现的标志包括 SO_ACCEPTON (ACC)、SO_WAITDATA (W) 和 SO_NOSPACE (N) 等	
Type	Ubuntu Linux 系统支持的套接字访问类型如下	
	SOCK_DGRAM	相应的套接字用于数据报（无连接模式）
	SOCK_STREAM	相应的套接字是一个 Stream（连接模式）套接字
	SOCK_RAW	相应的套接字用于原始套接字模式
	SOCK_RDM	相应的套接字用于提供可靠的消息传递服务
	SOCK_SEQPACKET	相应的套接字支持顺序分组数据传输
	SOCK_PACKET	相应的套接字支持原始接口访问
State	状态字段通常包含下列关键字之一	
	FREE	相应的套接字尚未分配
	LISTENING	相应的套接字正在监听一个连接请求
	CONNECTING	相应的套接字连接正在建立
	CONNECTED	相应的套接字连接已经建立
	DISCONNECTING	相应的套接字正在断开连接
	(空)	相应的套接字尚未完成连接
Path	套接字的路径名	

如果想要查询与某个特定协议，如 TCP、UDP 或原始套接字有关的状态或统计信息，可以分别使用“-t”“-u”或“-w”选项。下面的例子说明了怎样使用“-t”选项专门显示与 TCP 协议有关的状态信息。

```
$ netstat -t
Active Internet connections (w/o servers)
Proto Recv-Q Send-Q Local Address          Foreign Address        State
Tcp      1      0 bogon:59793              auckland.canonical.:www CLOSE_WAIT
Tcp      0      0 bogon:46497              222.73.255.64:www    ESTABLISHED
$
```

## 2. 显示所有套接字的状态信息

使用 netstat 命令的“-a”选项，可以查询本地主机所有套接字（包括正在监听和尚未监听的套接字）的状态信息。下面是“netstat -a”命令的部分输出结果。

```
$ netstat -a
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address          Foreign Address        State
Tcp      0      0 *:44160                *:*                  LISTEN
Tcp      0      0 *:40961                *:*                  LISTEN
tcp      0      0 *:nfs                 *:*                  LISTEN
tcp      0      0 localhost:mysql       *:*                  LISTEN
.....
Active UNIX domain sockets (servers and established)
Proto RefCnt Flags  Type      State     I-Node Path
unix  2      [ ACC ] STREAM LISTENING 16065  /tmp/orbit-gqxing/linc-1793-0-1ec40ea6d7dd
unix  2      [ ACC ] STREAM LISTENING 16232  /tmp/orbit-gqxing/linc-17be-0-132406752344e
unix  2      [ ACC ] STREAM LISTENING 16599  /tmp/orbit-gqxing/linc-17cc-0-61ce34d6b70e
unix  2      [ ACC ] STREAM LISTENING 16745  /tmp/orbit-gqxing/linc-17c6-0-61ce34c93b288
.....
$
```

根据套接字的状态信息，可以确定某个网络服务进程是否已经启动。例如，为了查询是否已经启动了 FTP 服务器守护进程，可以使用下列命令。

```
$ netstat -a | grep ftp
tcp      0      0 *:ftp                 *:*                  LISTEN
$
```

上述输出信息的状态字段为 LISTEN，说明 FTP 服务器守护进程已经启动。

## 3. 显示网络接口的状态信息，检测网络主机的可靠性

要检测网络主机的可靠性和数据通信能力，可以利用 netstat 命令的“i”选项，显示本地系统网络接口的状态信息，借以确定系统发送和接收的分组数据的数量。下面是“netstat -i”命令输出的一个例子，其中给出了网络接口的 TCP/IP 流量统计数据。

```
$ netstat -i
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    12573    4      4 4      12224    0      0      0  BMRU
lo       16436  0     86      0      0 0      86      0      0      0  LRU
$
```

表 18-6 给出了“netstat -i”命令的输出结果及简单说明。



表 18-6 netstat 命令的部分输出字段及其解释

输出字段	简单说明
Iface	网络接口的名字
MTU	网络接口当前支持的最大传输单位。MTU 是 IP 模块的设备驱动程序一次收发能够处理的最大字节数量。对于以太网络接口，MTU 的默认值为 1500；对于回环网络接口（loopback），其数值通常为 8232；对于 IEEE 802.3 接口，其数值为 1492
Met	网络接口当前支持的路由度量值
RX-OK	正确无误地接收了多少分组数据
RX-ERR	接收的分组数据本身有误的分组数据数量
RX-DRP	接收时由于各种原因而丢弃的分组数据数量
RX-OVR	由于接收错误和处理不及时等原因而遗失的分组数据数量
TX-OK	正确无误地发送了多少分组数据
TX-ERR	发送有误的分组数据数量
TX-DRP	发送时丢弃的分组数据数量
TX-OVR	由于发送错误而遗失的分组数据数量
Flg	网络接口设置的标志。可能出现的接口设置标志如下： □ B 广播地址设置标志； □ L 回环网络接口标志，表示相应的网络接口是回环接口； □ M 混杂接收模式标志，意味着网络接口能够接收所有的分组数据； □ O 网络接口禁用 ARP 标志； □ P 点到点链接标志，表示相应的网络接口采用的是点到点连接方式； □ R 网络接口已处于运行状态； □ U 网络接口已经启用

上述输出信息给出了当前主机中每个网络接口已经发送和接收的分组数据数量。利用发送和接收的分组数量，可以判定网络的运行是否正常。在一个正常运行的网络系统中，反映网络流量的 RX-OK 和 TX-OK 字段应当连续不断地增长。

例如，假定客户系统在尝试请求某个服务器引导自己的系统时，服务器接收的分组数据计数（RX-OK）不断地增加，而输出的分组数据计数（TX-OK）保持不变。这一事实表明，服务器正在接收和处理来自客户机含有引导请求的分组数据，但服务器不知道如何响应，故只有输入的请求，而服务器没有数据输出。这个结果说明网络的设置存在问题：要么客户系统的配置文件如 hosts 文件中的服务器地址设置不正确，要么服务器的远程引导服务没有启动。

此外，如果 RX-OK 与 TX-OK 字段的计数一直保持稳定不变，说明系统根本没有接收和发送任何分组数据。这个结果表明，要么网络连接存在问题，要么软件设置存在问题，总之，本地系统与其他系统之间没有任何数据通信。

#### 4. 显示当前路由的状态

netstat 命令的“-r”选项用于显示本地主机维护的路由信息，其中反映了本地主机已知的所有路由及其当前状态，示例如下。

```
$ netstat -r
Kernel IP routing table
Destination      Gateway        Genmask        Flags    MSS Window irtt Iface
172.16.3.0        *          255.255.255.0    U        0 0          0 eth0
link-local        *          255.255.0.0     U        0 0          0 eth0
default          bogon        0.0.0.0       UG       0 0          0 eth0
$
```

表 18-7 给出了“netstat -r”命令输出信息的简单说明。

表 18-7 “netstat -r” 命令的部分输出字段及其说明

输出字段	简单说明
Destination	表示路由的目的主机或网络
Gateway	表示把分组数据转发到目的主机或网络需要用到的网关。如果这个字段为星号“*”，则意味着无需使用网关
Genmask	确定路由时使用的子网掩码。当给定一个IP地址，期望找到一个适当的路由时，系统内核将会利用这个子网掩码对IP地址进行逻辑与运算，然后依次检索每一个路由表项，从中找出匹配的路由
Flags	表示路由的当前状态。其中可能出现的状态标志如下： <input type="checkbox"/> U 表示相应的路由已经建立； <input type="checkbox"/> G 表示路由是利用网关实现的； <input type="checkbox"/> H 表示路由的目的地是一个主机，如同环网络接口表示的主机（127.0.0.1）； <input type="checkbox"/> R 表示路由是已恢复的动态路由； <input type="checkbox"/> D 表示路由是由路由守护进程或 ICMP 重定向消息动态建立的； <input type="checkbox"/> M 表示路由是根据路由守护进程或 ICMP 重定向消息修改后而发生了变动； <input type="checkbox"/> C 表示缓存的路由
MSS	MSS (Maximum Segment Size) 称作最大数据段尺寸，也是系统内核通过相应的路由能够构造和传输的最大数据报尺寸
Window	Window 是本地系统能够接受远程主机一次传输的最大数据量
irtt	irtt 是“initial round trip time”的缩写，意即初始的往返传输时间。TCP 协议能够确保在通信的主机之间可靠地传输数据。如果数据传输有误，TCP 将会重传丢失的数据报。TCP 协议采用一个计数器，记录数据报传输远程主机花费的时间，以及收到一个确认需要多长时间，因此知道在重传数据报之前需要等待多长时间。这个时间称作数据报传输与确认的往返时间。在开始建立网络连接时，TCP 协议将会采用一个初始的往返时间作为默认值。对于大多数网络而言，TCP 协议采用的默认往返时间值是适当的，但对于一些速度较慢的网络而言，如果这个时间太短，将会引起不必要的数据报重传。必要的话，可以利用 route 命令设置 irtt 值。如果这个字段的值为 0，则意味着采用默认值
Iface	表示路由经由的网络接口

## 5. 按协议显示统计信息

“netstat -s” 命令用于显示不同协议（IP、UDP、TCP 或 ICMP 等）的分类统计数据。利用这些统计数据，可以确定哪个网络协议存在问题，示例如下。

```
$ netstat -s
Ip:
    8701 total packets received
    0 forwarded
    0 incoming packets discarded
    8698 incoming packets delivered
    8567 requests sent out
    20 dropped because of missing route
Icmp:
    2387 ICMP messages received
    0 input ICMP message failed.
    ICMP input histogram:
        echo replies: 2387
    2414 ICMP messages sent
    0 ICMP messages failed
    ICMP output histogram:
        destination unreachable: 1
        echo request: 2413
IcmpMsg:
    InType0: 2387
    OutType3: 1
    OutType8: 2413
Tcp:
    649 active connections openings
    15 passive connection openings
    620 failed connection attempts
    5 connection resets received
    0 connections established
```



```
6056 segments received  
6016 segments send out  
63 segments retransmited  
0 bad segments received.  
625 resets sent  
Udp:  
 73 packets received  
 1 packets to unknown port received.  
 0 packet receive errors  
 71 packets sent  
$
```

在利用“netstat -s”命令获取统计数据时，要重点考察输入输出错误、校验和错误、重传次数、分组数据丢失统计等字段。这些数据能够反映网络和主机的运行状态与性能，以及硬件连接和软件安装是否存在问題，等等。

### 18.6.3 使用 ping 命令测试远程主机的连通性

为了测试远程主机的连通性，最常见、最简单的方法是采用 ping 命令。运行时，ping 命令将会利用 ICMP 协议向指定的远程主机发送 ECHO\_REQUEST 请求信息，期望远程主机回以 ECHO\_REPLY 响应信息。利用 ping 命令，可以检查与指定的远程主机是否已经建立了 TCP/IP 连接，其语法格式简写如下。

```
ping [-aAfn] [-c count] [-i interval] [-s size] [-w dead-line] dest-host
```

其中，“-a”选项表示每次响应时均给出声音警示。“-A”选项表示以实际的往返响应时间为间隔，连续地发送数据报信息。“-f”选项表示连续不断地发送数据报信息，而不管是否收到响应信息。“-n”选项表示只需显示主机的 IP 地址，不必把 IP 地址解析成主机名。“-c”选项表示发送指定次数的数据报信息之后停止运行。“-i”选项表示发送每个数据报信息之间的时间间隔，默认值为 1 秒。“-s”选项表示分组数据的大小，默认值为 56 个字节（加上 8 个 ICMP 头字节，共 64 个字节）。“-w”选项表示以秒为单位的超时值，不管发送或接收了多少信息，一旦达到超时值，立即停止运行。dest-host 是远程主机的名字。

#### 1. 确定与远程主机的连通性

ping 命令的主要用途是测试本地系统与远程主机之间的网络连通性，以及远程主机是否正在运行。如果指定的远程主机能够接收并响应 ICMP 请求，ping 命令的运行结果如下。

```
$ ping iscas  
PING iscas (192.168.90.100) 56(84) bytes of data.  
64 bytes from iscas (192.168.90.100): icmp_seq=1 ttl=64 time=0.054 ms  
64 bytes from iscas (192.168.90.100): icmp_seq=2 ttl=64 time=0.039 ms  
64 bytes from iscas (192.168.90.100): icmp_seq=3 ttl=64 time=0.043 ms  
Ctrl-C  
--- iscas ping statistics ---  
3 packets transmitted, 3 received, 0% packet loss, time 1998ms  
rtt min/avg/max/mdev = 0.039/0.045/0.054/0.008 ms  
$
```

如果指定的远程主机关机，或者收不到 ICMP 响应信息，ping 命令的运行结果如下。

```
$ ping iscas  
PING iscas (192.168.90.100) 56(84) bytes of data.  
From iscas (192.168.90.100) icmp_seq=1 Destination Host Unreachable  
From iscas (192.168.90.100) icmp_seq=2 Destination Host Unreachable  
From iscas (192.168.90.100) icmp_seq=3 Destination Host Unreachable  
Ctrl-C  
--- iscas ping statistics ---
```

```
4 packets transmitted, 0 received, +3 errors, 100% packet loss, time 3006ms,
pipe 3
S
```

## 2. 确定网络通信是否丢失分组数据

利用 ping 命令的“-A”选项，不仅可以确定本地系统与远程主机间的连通性，还可以检测主机之间的网络通信是否丢失分组数据。“-A”选项表示以实际的往返响应时间为间隔，向指定的远程主机连续地发送分组数据，直至使用中断键终止命令的执行。对于普通用户而言，最小的往返响应时间间隔为 200 毫秒。示例如下。

```
$ ping -nA www.google.cn
PING cn.l.google.com (203.208.37.99) 56(84) bytes of data.
64 bytes from 203.208.37.99: icmp_seq=1 ttl=243 time=2.45 ms
64 bytes from 203.208.37.99: icmp_seq=2 ttl=243 time=3.38 ms
64 bytes from 203.208.37.99: icmp_seq=3 ttl=243 time=3.66 ms

--- cn.l.google.com ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 400ms
rtt min/avg/max/mdev = 2.458/3.167/3.660/0.516 ms, ipg/ewma 200.054/2.709 ms
```

“packet loss”字段是分组数据丢失的统计数据。如果ping命令失败，或者丢失的分组数据数量较大，则需要利用ifconfig和netstat等命令进一步检查网络的工作状态。

#### 18.6.4 使用 ping 命令检测网络主机的性能

ping 命令不仅能够用于检测网络和主机的硬件连接以及软件运行状态等连通性问题，还可用  
于检验网络和主机的响应能力与传输性能。

利用 ping 命令的“-f”选项，可以连续不断地向远程主机发送数据报信息，而且完全不必顾及是否收到响应信息。ping 命令每发送一个 ECHO\_REQUEST 信息，即在屏幕上输出一个句点“.”，本地系统每收到一个 ECHO\_REPLY 响应信息，即在屏幕上输出一个退格符，擦除一个句点“.”。屏幕上输出句点“.”的多少可以大致说明远程主机的响应能力。当按下 Ctrl-C 键之后输出的“rtt min/avg/max/mdev”能够给出更准确的统计数据，说明往返传输时间的最小、平均及最大时间值（毫秒）。由此可以得到网络与主机的数据传输性能。示例如下。

```
$ sudo ping -f www.google.com
PING www-china.l.google.com (64.233.167.99) 56(84) bytes of data.
..... Ctrl-C
--- www-china.l.google.com ping statistics ---
240 packets transmitted, 187 received, 22% packet loss, time 3297ms
rtt min/avg/max/mdev = 300.658/303.499/307.801/1.577 ms, pipe 25, ipg/ewma 13.794/
303.967 ms

$ sudo ping -f 172.16.3.254
PING 172.16.3.254 (172.16.3.254) 56(84) bytes of data.
..... Ctrl-C
--- 172.16.3.254 ping statistics ---
380 packets transmitted, 301 received, 20% packet loss, time 1699ms
rtt min/avg/max/mdev = 1.767/2.637/58.112/4.466 ms, pipe 5, ipg/ewma 4.485/2.119 ms
$
```

此外，如果使用“-s”选项指定较大尺寸的分组数据，还可以进一步检测网络和主机的响应能力和数据传输性能。



## 18.6.5 使用 ftp 命令检测网络主机的传输性能

利用 ftp 命令也可以检测网络主机的传输性能。进入 ftp 会话之后，可以使用下列 ftp 子命令，以 /dev/zero 作为输入，以 /dev/null 作为输出，传输一个较大的文件。避免使用硬盘（硬盘本身也有开销），也不必使用内存缓存整个文件，因而能够纯粹测试网络传输的性能。

```
put "|dd if=/dev/zero bs=32k count=10000" /dev/null
```

在下面的例子中，put 命令最后一行的统计数据可以给出网络传输的速度。在执行实际的测试时，数据块的大小与传输的次数可由用户视具体情况而定。

```
$ ftp iscas
...
ftp> put "|dd if=/dev/zero bs=32k count=10000" /dev/null
local: |dd if=/dev/zero bs=32k count=10000 remote: /dev/null
200 PORT command successful. Consider using PASV.
150 Ok to send data.
10000+0 records in
10000+0 records out
327680000 bytes (328 MB) copied, 4.84348 s, 67.7 MB/s
226 File receive OK.
327680000 bytes sent in 4.84 secs (66088.1 kB/s)
ftp> put "|dd if=/dev/zero bs=64k count=10000" /dev/null
local: |dd if=/dev/zero bs=64k count=10000 remote: /dev/null
200 PORT command successful. Consider using PASV.
150 Ok to send data.
10000+0 records in
10000+0 records out
655360000 bytes (655 MB) copied, 10.5424 s, 62.2 MB/s
226 File receive OK.
655360000 bytes sent in 10.54 secs (60708.1 kB/s)
ftp>
```

## 18.6.6 使用 traceroute 命令跟踪路由信息

在查询指定的主机时，traceroute 命令利用 IP 协议的 TTL (Time-to-Live) 字段，要求途经的每个路由器或目的主机返回一个“ICMP TIME EXCEEDED”响应信息。在 traceroute 命令的跟踪过程中，TTL 字段从默认值 1 开始，每经过一个中间路由器，TTL 字段将依次加 1，直至接收到一个“ICMP PORT UNREACHABLE”信息（表示已经到达指定的主机），或者已经达到 TTL 的最大值，跟踪过程立即结束。针对每一个 TTL 值，即途经的每一个中间路由器，traceroute 命令将会发送 3 个 UDP 数据报或“ICMP ECHO”查询，因而期望得到 3 个“ICMP TIME EXCEEDED”响应信息。

traceroute 命令的主要功能是跟踪 IP 分组数据到达目的主机时经由的整个路径，用于显示相互通信的两个系统之间，IP 分组数据从源主机到目的主机穿越的所有中间路由。另外一个功能就是发现任何不合理的路由或路径不正确的路由。如果无法到达指定的主机，可以使用 traceroute 命令，检查 IP 分组通过什么路径联系远程主机，中间哪一个环节可能存在问题。

traceroute 命令的语法格式如下。

```
traceroute [-dn] -f first-ttl -w wait -m max-ttl -p port -q queries dest-host
```

其中，“-d”选项表示启用套接字级的调试功能。“-n”表示仅显示中间路由器或目的主机的 IP 地址，无需把 IP 地址解析成主机名。“-f”选项用于指定起始 TTL 值，默认值为 1。“-w”选项表示等待中间路由器或目的主机返回 ICMP 响应信息的时间，默认值为 5 秒。“-m”选项用于指定 TTL 的最大值（即最大的中转路由器数量），默认值为 30。“-p”选项表示 traceroute 命令使用的 UDP 端口号。“-q”选项表示针对每一个 TTL，中间路由器或目的主机应返回几个 ICMP 响应

信息，默认值为 3。dest-host 表示查询的主机。

traceroute 命令将会显示到达目的主机期间途经的每一个路由器（包括目的主机）的 IP 地址，以及分组数据往返传输耗费的时间。这个时间信息对于分析两个主机之间任何环节的网络流量或通信能力是非常有用的。如果在发出 UDP 数据报或“ICMP ECHO”查询信息的 5 秒钟之内一直没有接收到 ICMP 响应信息，traceroute 命令将会在相应的中间路由器或目的主机的返回时间字段输出一个星号“\*”字符。

traceroute 命令的每一行输出信息通常包含下列 3 个字段：

- TTL 值（默认情况下从 1 开始，最大值为 30）；
- 中间路由器或目的主机的 IP 地址；
- 每一个 ICMP 响应信息的返回时间。

例如，traceroute 命令的下列输出信息表明，一个分组数据从本地主机到达远程主机 www.google.cn，中间跨越 9 个网络路径。这个输出信息同时也给出了分组数据途经每个网段时耗费的时间。

```
$ traceroute www.google.cn
traceroute to www.google.cn (203.208.37.104), 30 hops max, 40 byte packets
1 * *
2 202.106.57.29 (202.106.57.29) 17.161 ms 19.006 ms 20.952 ms
3 61.148.155.81 (61.148.155.81) 22.324 ms 24.341 ms 25.900 ms
4 61.148.156.221 (61.148.156.221) 28.560 ms 30.234 ms 32.406 ms
5 202.96.12.49 (202.96.12.49) 35.071 ms 37.101 ms 38.352 ms
6 219.158.11.90 (219.158.11.90) 40.676 ms 25.791 ms *
7 219.158.32.226 (219.158.32.226) 26.538 ms 27.036 ms 26.440 ms
8 203.208.62.27 (203.208.62.27) 27.771 ms 25.818 ms 26.805 ms
9 203.208.62.121 (203.208.62.121) 33.355 ms 30.822 ms 29.088 ms
10 203.208.37.104 (203.208.37.104) 26.597 ms 26.135 ms 26.572 ms
$
```

## 18.6.7 利用 tcpdump 捕捉、分析网络分组数据

tcpdump 用于捕捉网络接口中匹配指定表达式的分组数据头信息，其语法格式简写如下（注意，运行 tcpdump 要求具有超级用户的访问权限）。

```
tcpdump [-AdDeflLnNOpqRStuUvxX] [-c count] [-C file_size]
          [-F file] [-i interface] [-m module] [-r file]
          [-s snaplen] [-w file] [-W filecount] [expression]
```

其中，“-t”选项表示无需在每一行之前都输出时间，“-v”、“-vv”和“-vvv”选项表示输出更多的说明信息。利用“-w”选项，可以把捕捉的分组数据保存到指定的文件中，以便将来再分析。利用“-r”选项，可以从指定的文件而不是网络接口中读取分组数据进行分析。在任何情况下，tcpdump 仅处理匹配表达式的分组数据。

如果没有指定“-c”选项，tcpdump 将会连续地捕捉分组数据，直至按下 Ctrl-C 中断键。如果指定了“-c”选项，在捕捉了指定数量的分组数据，或者按下 Ctrl-C 中断键之后，tcpdump 将停止处理分组数据。当结束捕捉分组数据时，tcpdump 将会给出下列统计数据。

- packets captured ——表示 tcpdump 读取和处理的分组数据数量。
- packets received by filter ——表示 tcpdump 接收到的分组数据数量。取决于操作系统，这个数字表示接收到的分组数据总量、匹配指定表达式的分组数据数量，以及匹配和处理的分组数据数量。
- packets dropped by kernel ——表示由于缺乏缓冲区空间导致 tcpdump 丢失分组数据的数量。



## 1. 基本表达式

作为过滤器，表达式 expression 用于选择匹配的分组数据。如果未指定表达式，则意味着捕捉所有的分组数据。表达式可由一个或多个基本元素（如主机名、网络名、IP 地址或端口号等）组成，而基本元素之前通常可以增加一个或多个限定符。tcpdump 支持 3 种不同的限定符：类型、传输方向和协议。

在指定表达式时，可以按照捕捉类型、传输方向和协议，限定捕捉的分组数据。tcpdump 支持的捕捉类型包括 host、net、port 和 portrange（端口范围）等，如“host beijing”、“net 192.168”、“port 80”或“portrange 2600-2608”等。如果未指定捕捉类型，默认值为 host。

传输方向用于限定捕捉来自何方或去至哪里的分组数据，其中包括 src、dst、“src or dst”和“src and dst”等，如“src beijing”、“dst net 172.16”或“src or dst port ftp-data”。如果未指定方向限定符，默认值为“src or dst”。

协议用于限制匹配指定协议的分组数据，其中包括 ether、ip、arp、rarp、tcp 及 udp 等，如“ip src beijing”、“arp net 192.168”、“tcp port 21”或“udp portrange 8000-8080”。如果未指定协议，表示与相应类型一致的所有协议。例如，“src beijing”意味着“(ip or arp or rarp) src beijing”，“net officenet”意味着“(ip or arp or rarp) net officenet”，“port 53”意味着“(tcp or udp) port 53”，等等。

表 18-8 列出了部分常用的基本表达式及其简单说明。

表 18-8

基本表达式及其简单说明

基本表达式	简 单 说 明
dst host <i>host</i>	表示捕捉目的地址字段匹配指定主机的分组数据，其中的 host 可以是主机名或 IP 地址。“dst host”之前可以加 ip、arp 或 rarp 前缀
src host <i>host</i>	表示捕捉源地址字段匹配指定主机的分组数据。同样，“src host”之前也可以加 ip、arp 或 rarp 前缀
host <i>host</i>	表示目的或源地址字段匹配指定主机的分组数据。host 之前也可以加 ip、arp 或 rarp 前缀
dst net <i>net</i>	表示捕捉目的地址字段匹配指定网络的分组数据，其中的 net 可以是网络名(取自/etc/networks 文件)或 IP 地址
src net <i>net</i>	表示捕捉源地址字段匹配指定网络的分组数据
net <i>net</i>	表示捕捉目的或源地址字段匹配指定网络的分组数据
net net mask <i>netmask</i>	表示捕捉地址字段匹配指定网络与指定子网的分组数据。net 之前可以加 src 或 dst 前缀
net net/len	表示捕捉地址字段匹配指定网络与指定子网掩码长度的分组数据。net 之前可以加 src 或 dst 前缀
dst port <i>port</i>	表示捕捉目的端口字段匹配指定端口的分组数据，其中 port 可以是端口名(取自/etc/services 文件)或端口号。“dst port”之前可以加 tcp 或 udp 前缀
src port <i>port</i>	表示捕捉源端口字段匹配指定端口的分组数据。“src port”之前可以加 tcp 或 udp 前缀
port <i>port</i>	表示捕捉目的或源端口字段匹配指定端口的分组数据。port 之前可以加 tcp 或 udp 前缀
dst portrange <i>port1-port2</i>	表示捕捉目的端口字段匹配指定端口范围的分组数据。“dst portrange”之前可以加 tcp 或 udp 前缀
src portrange <i>port1-port2</i>	表示捕捉源端口字段匹配指定端口范围的分组数据。同样，“src portrange”之前也可以加 tcp 或 udp 前缀
portrange <i>port1-port2</i>	表示捕捉目的或源端口字段匹配指定端口范围的分组数据。portrange 之前也可以加 tcp 或 udp 前缀
ip proto <i>protocol</i>	表示捕捉的分组数是 IPv4 协议类型的分组数据。指定的协议 protocol 可以是一个数字，或者 icmp、igmp、igrp、udp 或 tcp 等名字之一。注意，tcp、udp 和 icmp 也是关键字，之前必须加转义符号“\”(在 bash 或 tcsh 中需加两个转义符号)
tcp, udp, icmp	“ip proto protocol”的缩写，其中的 protocol 可以是 ip、arp 或 rarp 等协议之一。注意，tcp、udp 和 icmp 也是关键字，之前必须加转义符号“\”
ether proto <i>protocol</i>	表示捕捉的分组数是 Ethernet 协议类型的分组数据。指定的 protocol 可以是一个数字，或者 ip、arp 或 rarp 等名字之一。注意，ip、arp 和 rarp 也是关键字，之前必须加转义符号“\”
ip, arp, rarp	“ether proto protocol”的缩写，其中的 protocol 可以是 ip、arp 或 rarp 等协议之一

此外，通过关系运算符或逻辑运算符，基本表达式可以构成更为复杂的组合表达式。其中，关系运算符可以是“>”、“<”、“>=”、“<=”、“=”或“!=”之一，逻辑运算符可以是“!(not)”、“&&(and)”或“||(or)”之一。当表达式是一个算术表达式时，还可以使用二进制算术运算符“+”、“-”、“\*”、“/”、“&”、“|”、“<<”或“>>”等。tcpdump 支持的组合表达式比较复杂，感兴趣的读者可以查询 tcpdump 命令手册页。

例如，要捕捉主机 beijing 收到和发出的所有分组数据，可以使用下列命令。

```
# tcpdump host beijing
```

要捕捉主机 beijing 与 iscas 或 beijing 与 yantai 之间的所有分组数据，可以使用下列命令。

```
# tcpdump host beijing and \(`iscas or yantai`\)
```

要捕捉主机 beijing 与其他任何主机 (iscas 除外) 之间的所有 IP 分组数据，可以使用下列命令。

```
# tcpdump ip host beijing and not iscas
```

## 2. 输出格式

tcpdump 命令的输出数据及其格式依赖于选择的协议，下面仅以 TCP 协议为例予以说明，其他协议的输出数据及其格式详见 tcpdump 命令手册页。

与 TCP 协议有关的输出信息通常具有下列格式。

```
time protocol src > dst: flags data-seqno ack window urgent options
```

其中，time 是捕捉到分组数据的系统时间，通常以“hh:mm:ss.frac”表示。protocol 是 IP、arp 或 rarp 等协议之一。src 和 dst 分别是源和目的主机的地址及端口。flags 是 S (SYN)、F (FIN)、P (PUSH)、R (RST)、W (ECN CWR) 或 E (ECN-Echo) 等标志的组合，句点“.”表示没有任何标志。data-seqno 是数据的顺序号，以“first:last(nbytes)”形式表示分组数据中的数据部分与字节数。ack 是预期收到的下一个数据的顺序号。window 是接收缓冲区的字节数。urgent (urg) 表示分组数据中存在需要“紧急”处理的数据。options 是位于单书名号中的 tcp 选项，如“<mss 1024>”。

在实际的输出数据中，src、dst 和 flags 字段总是存在，至于其他字段是否存在，取决于分组数据 TCP 协议头字段的内容。下面是在 Windows 系统 169.254.78.56 (winxp) 使用 ftp 连接到主机系统 169.254.78.100 (iscas) 的注册会话过程中捕捉到的部分数据。

```
$ sudo tcpdump -t host 169.254.78.100 and tcp
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 96 bytes
IP winxp.1273 > iscas.local.ftp: S 1766343524:1766343524(0) win 65535 <mss 1460,nop,nop,sackOK>
IP iscas.local.ftp > winxp.1273: S 2891770256:2891770256(0) ack 1766343525 win 5840 <mss
1460,nop,nop,sackOK>
IP winxp.1273 > iscas.local.ftp: . ack 1 win 65535
IP iscas.local.ftp > winxp.1273: P 1:21(20) ack 1 win 5840
IP winxp.1273 > iscas.local.ftp: . ack 21 win 65515
IP winxp.1273 > iscas.local.ftp: P 1:14(13) ack 21 win 65515
IP iscas.local.ftp > winxp.1273: . ack 14 win 5840
IP iscas.local.ftp > winxp.1273: P 21:55(34) ack 14 win 5840
IP winxp.1273 > iscas.local.ftp: . ack 55 win 65481
IP winxp.1273 > iscas.local.ftp: P 14:29(15) ack 55 win 65481
IP iscas.local.ftp > winxp.1273: . ack 29 win 5840
IP iscas.local.ftp > winxp.1273: P 55:78(23) ack 29 win 5840
IP winxp.1273 > iscas.local.ftp: . ack 78 win 65458

13 packets captured
13 packets received by filter
0 packets dropped by kernel
$
```

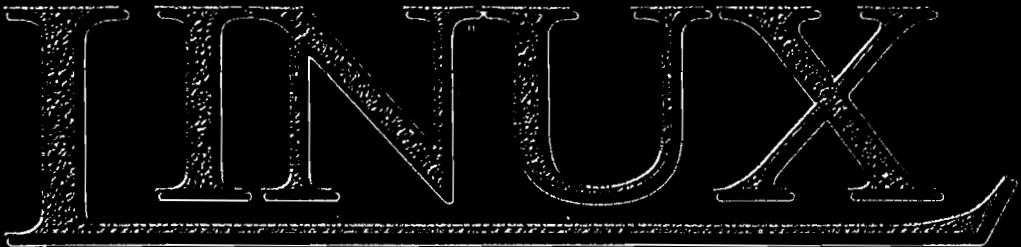


第 1 行表示 winxp 向 iscas 系统的 TCP 端口 ftp 发送了一个分组数据。S 表示设置了 SYN 标志，分组数据的顺序号是 1766343524，“1766343524:1766343524 (0)”表示分组数据中不包含实际的数据，接收缓冲区为 65 535 个字节，最大段尺寸（max-segment-size, mss）为 1460 个字节。

第 2 和第 3 行表示 iscas 采用一个类似的分组数据确认 winxp 发送的 SYN 连接请求，winxp 接着又确认了 iscas 设置的 SYN，“.”表示没有设置任何标志，分组数据中也不包含任何数据，因此没有数据顺序号字段。

注意，ack 后的顺序号是一个小的整数（1）。当第一次捕捉到 tcp 分组数据时，tcpdump 将会输出分组数据的绝对顺序号。对于随后捕捉的分组数据，tcpdump 将会输出当前分组数据与初始分组数据顺序号的差。这意味着从第一个分组数据之后，顺序号可以解释作为数据流的相对字节位置（每个传输方向的第一个数据字节从 1 开始编号）。利用 tcpdump 命令的“-S”选项，可以强制 tcpdump 输出分组数据的绝对顺序号。

第 4 行表示 iscas 向 winxp 发送了 20 个字节的数据（字节 1~20），同时设置分组数据的 PUSH 标志。第 5 行表示 winxp 收到了 iscas 发送的数据，并给予确认。第 6 行表示 winxp 向 iscas 发送了 13 个字节的数据（字节 1~13），同时设置分组数据的 PUSH 标志。整个交互过程就是一方发送，另一方接收、确认，接着再发送，如此往复，直至结束。



## 第19章

### TCP/IP 网络应用

Linux 系统提供的丰富网络功能有助于用户实现数据通信和文件传输。本章主要介绍 TCP/IP 网络应用，其内容主要包括：

- OpenSSH;
- Telnet 远程注册；
- FTP 文件传输。





## 19.1 OpenSSH

在 Linux 系统中，OpenSSH 是目前最流行的远程系统注册与文件传输应用，也是传统的 Telenet、FTP 与 R 系列等网络应用的换代产品。其中，ssh（Secure Shell）可以替代 telnet、rlogin 和 rsh，scp（Secure Copy）与 sftp（Secure FTP）能够替代 ftp。

OpenSSH 不仅适用于 Linux 系统，也可用于 AIX、Solaris 和 HP-UX 等 UNIX 系统，而且其 ssh 与 scp 等客户端软件也可用于 Windows 等其他操作系统。如有必要，Windows 用户可以下载一个 Windows 版的 SCP 客户端软件，即 WinSCP，在 Windows 系统与 Linux 系统之间实现文件的复制，详情参见 <http://winscp.vse.cz/eng/>。

OpenSSH 采用密钥的方式对数据进行加密，确保数据传输的安全。在正式开始传输数据之前，双方首先要交换密钥；当收到对方的数据时，再利用密钥和相应的程序对数据进行解密。这种加密的数据传输有助于防止非法用户获取数据信息。

OpenSSH 采用随机的方式生成公私密钥。密钥通常只需生成一次，必要时也可以重新制作。

当使用 ssh 命令注册到远程系统时，OpenSSH 服务器的 sshd 守护进程将会发送一个公钥，OpenSSH 客户端软件 ssh 将会提请用户确认是否接受发送的公钥。同时，OpenSSH 客户机也会向服务器回送一个密钥，使 OpenSSH 连接双方的每个系统都拥有对方的密钥，因而能够解密对方经由加密链路发送的加密数据。

OpenSSH 服务器的公钥与私钥均存储在/etc/ssh 目录中。在 OpenSSH 客户端，用户收到的所有公钥，以及提供密钥的 OpenSSH 服务器的 IP 地址均存储在用户主目录下的~/.ssh/known\_hosts 文件中（注意，.ssh 是一个隐藏的目录）。如果密钥与 IP 地址不再匹配，OpenSSH 将会认为某个环节出了问题。例如，重新安装操作系统或升级 OpenSSH 都会导致系统再次生成新的密钥。当然，恶意的网络攻击也会造成密钥的变动。因此，当密钥发生变化时，总是应当了解密钥发生变化的原因，以确保网络访问期间的数据安全。

### 19.1.1 安装 OpenSSH 服务器

在 Ubuntu Linux 系统的安装过程中，作为一个基本的系统软件，OpenSSH 的客户端软件包将会随着 Linux 系统一起安装，但 OpenSSH 服务器需要单独安装。要安装 openssh-server 服务器软件包，可以使用 apt-get、aptitude 或 synaptic 等软件工具，示例如下。

```
$ sudo aptitude install openssh-server
```

安装之后，为了验证 OpenSSH 服务器的 sshd 守护进程是否已开始运行，可以使用下列命令（参见第 9 章）。

```
$ pidof sshd  
5016  
$
```

### 19.1.2 /etc/ssh/sshd\_config 配置文件

/etc/ssh/sshd\_config 是 OpenSSH 服务器的默认配置文件，sshd 守护进程根据其中的定义，规范其处理动作。如果需要，也可以在 sshd 的命令行中使用“-f”选项指定其他配置文件。

`sshd_config` 配置文件的每一行要么是一个注释行（起始字符为注释符“#”），要么是一个参数设置。`sshd_config` 配置文件的语法格式如下。

`parameter value`

表 19-1 给出了部分重要的配置参数极其简单说明。

表 19-1                   `sshd` 的部分配置参数

配置参数	简单说明
<code>AcceptEnv</code>	指定客户端发送的哪些环境变量能够复制到当前会话的运行环境（客户端需要设置其配置文件 <code>ssh_config</code> 中的 <code>SendEnv</code> 参数）。注意，只有 SSH 协议版本 2 支持环境变量的传递。设置时只需指定变量的名字，其中可以包含通配符星号“*”与问号“?”。指定多个环境变量时，中间需加空格分隔符。此外，还可把环境变量分布到多个 <code>AcceptEnv</code> 配置参数中。需要注意的是，某些环境变量会对限制的用户运行环境带来负面影响，因此，应当仔细地使用这个配置参数。这个配置参数的默认值是拒绝接受任何环境变量
<code>AddressFamily</code>	指定 <code>sshd</code> 守护进程支持的地址系列，即网络协议，其有效值是 <code>any</code> 、 <code>inet</code> （仅支持 IPv4）或 <code>inet6</code> （仅支持 IPv6）。默认值是 <code>any</code> ，表示同时支持两种网络协议
<code>AllowGroups</code>	设定这个配置参数时，可以列举多个用户组的名字或模式，中间加空格分隔符。如果设定了这个配置参数，只允许其用户组（包括附加用户组）匹配指定用户组名或模式的用户注册。注意，这里只能指定用户组名，不能使用用户组 ID。通常，所有用户组的成员均允许注册。 <code>sshd</code> 将会按照 <code>DenyUsers</code> 、 <code>AllowUsers</code> 、 <code>DenyGroups</code> 与 <code>AllowGroups</code> 的顺序依次处理用户注册的限定策略。下同
<code>AllowUsers</code>	设定这个配置参数时，可以列举多个用户名或模式，中间加空格分隔符。如果设定了这个配置参数，只允许匹配指定用户名或模式的用户注册。同样，这里只能指定用户名，不能使用用户 ID。通常，所有用户均可注册。如果用户名模式采用 <code>USER@HOST</code> 形式，需要单独检测 <code>USER</code> 与 <code>HOST</code> ，从而能够确保特定主机中的特定用户注册
<code>AuthorizedKeysFile</code>	指定包含用户认证公钥的文件。 <code>AuthorizedKeysFile</code> 配置参数的值可以包含%T 形式的标记符，这些标记符能够在连接建立期间予以替换，其中，%T 是单个百分号“%”，%h 可替换为认证用户的主目录，%u 可替换为注册用户的用户名。经过上述替换后， <code>AuthorizedKeysFile</code> 指定的文件将会成为一个绝对路径名或相对于用户主目录的相对路径名，默认值是 <code>.ssh/authorized_keys</code>
<code>ChallengeResponseAuthentication</code>	指定是否允许使用提示/应答式认证。 <code>sshd</code> 支持 <code>login.conf</code> 文件中定义的所有认证类型。默认值是 <code>yes</code> （Ubuntu Linux 系统的实际设置为 <code>no</code> ）
<code>DenyGroups</code>	设定这个配置参数时，可以列举多个用户组的名字或模式，中间加空格分隔符。如果设定了这个配置参数，则禁止其用户组（包括附加用户组）匹配指定用户组名或模式的用户注册。同样，这里只能指定用户组名，不能使用用户组 ID。通常，所有用户组的成员均允许注册
<code>DenyUsers</code>	设定这个配置参数时，可以列举多个用户名或模式，中间加空格分隔符。如果设定了这个配置参数，则禁止匹配指定用户名或模式的用户注册。同样，这里只能指定用户名，不能使用用户 ID。通常，所有用户均可注册。如果用户名模式采用 <code>USER@HOST</code> 形式，需要单独检测 <code>USER</code> 与 <code>HOST</code> ，从而能够确保正确地限制特定主机中的特定用户注册
<code>HostKey</code>	指定 SSH 使用的包含主机私钥的文件，默认值是 <code>/etc/ssh/ssh_host_key</code> （协议版本 1），以及 <code>/etc/ssh/ssh_host_rsa_key</code> 与 <code>/etc/ssh/ssh_host_dsa_key</code> （协议版本 2）。注意， <code>sshd</code> 将会拒绝使用一个同组用户或其他任何用户均可访问的文件。SSH 允许采用多个主机私钥文件，私钥 <code>rsa1</code> 用于 SSH 协议版本 1， <code>dsa</code> 或 <code>rsa</code> 用于 SSH 协议版本 2
<code>ListenAddress</code>	指定 <code>sshd</code> 应当监听的本地 IP 地址。在指定监听地址时，可以采用下列形式： <input type="checkbox"/> <code>ListenAddress host   IPv4_addr   IPv6_addr;</code> <input type="checkbox"/> <code>ListenAddress host   IPv4_addr:port;</code> <input type="checkbox"/> <code>ListenAddress [host   IPv6_addr]:port.</code> 其中， <code>host</code> 表示主机名， <code>IPv4_addr</code> 表示 IPv4 地址， <code>IPv6_addr</code> 表示 IPv6 地址， <code>port</code> 表示端口号。如果 <code>ListenAddress</code> 配置参数中未指定端口（第一种定义形式）， <code>sshd</code> 将会监听指定的地址及其所有端口，除非 <code>Port</code> 配置参数之前另有限定。因此，任何 <code>Port</code> 配置参数（如果存在）必须位于未加端口限制的 <code>ListenAddress</code> 配置参数之前。配置文件允许指定多个 <code>ListenAddress</code> 配置参数。通常， <code>sshd</code> 将会监听本地系统的所有 IP 地址
<code>LoginGraceTime</code>	如果用户未能成功地注册， <code>sshd</code> 将会在该配置参数指定的时间过后断开连接。如果参数值为 0，则表示没有时间限制，默认值为 120 秒



续表

配置参数	简单说明
LogLevel	指定 sshd 的日志信息级别, 可取的参数值是 SILENT、QUIET、FATAL、ERROR、INFO、VERBOSE、DEBUG、DEBUG1、DEBUG2 和 DEBUG3, 默认值是 INFO。DEBUG 与 DEBUG1 是等同的, DEBUG2 与 DEBUG3 表示记录较高级别的调试信息。采用 DEBUG 级别有可能会侵犯用户的隐私, 因而不建议使用
MaxAuthTries	指定每个连接请求容许的最大认证尝试次数。一旦注册失败的次数达到这个数值的一半, 之后再出现的失败尝试将会记录在日志文件中。这个配置参数的默认值是 6
PasswordAuthentication	指定是否允许使用密码认证, 默认值是 yes
PermitEmptyPasswords	当采用密码认证方式时, 指定 sshd 是否允许使用空的密码注册, 默认值是 no
PermitRootLogin	指定超级用户 root 是否能够使用 ssh 注册。可选的参数值必须是 yes、without-password、forced-commands-only 或 no 之一, 默认值是 yes。如果把这个配置参数设置为 without-password, 意味着禁止 root 用户采用密码认证。如果设为 forced-commands-only, 则允许 root 用户使用公钥认证方式注册, 但仅当在命令行的 “-o” 选项中指定了公钥认证 (PubkeyAuthentication)。对于远程备份而言, 这是非常有的, 即使在禁止 root 注册的情况下亦然。此时, 禁止 root 用户使用其他认证方法。如果把这个配置参数设置为 no, 表示禁止 root 用户注册
PidFile	指定包含 sshd 守护进程 PID 的文件, 默认值是 /var/run/sshd.pid
Port	指定 sshd 监听的端口号, 默认值是 22。配置文件允许同时指定多个 Port 配置参数, 参见 ListenAddress 的说明
PrintLastLog	指定以交互方式注册时, sshd 是否应当输出用户上一次注册的日期和时间, 默认值是 yes
PrintMotd	指定当用户以交互方式注册时, sshd 是否应当显示 /etc/motd 文件中的内容。在有些 Linux 系统中, 这一处理动作是由 Shell 或 /etc/profile 实现的。默认值是 yes
Protocol	指定 sshd 支持的协议版本, 可取的值是 1 和 2。如果同时指定多个版本号, 之间必须加逗号 “,” 分隔符, 默认值是 “2,1” (Ubuntu Linux 系统的实际设置为 2)。注意, 协议版本的列举顺序并不表示其优先级, 因为客户端可在服务器提供的多个协议版本中做出自己的选择, 故 “2,1” 等同于 “1,2”
PubkeyAuthentication	指定是否允许采用公钥认证, 默认值是 yes。注意, 这个配置参数仅适用于协议版本 2
RSAAuthentication	指定是否允许使用纯 RSA 认证, 默认值是 yes。这个配置参数仅适用于 SSH 协议版本 1
StrictModes	指定 sshd 是否应在接受用户注册之前检测文件的访问权限, 以及用户的文件与主目录的属主等属性。默认值是 yes
Subsystem	用于配置一个外部的服务程序, 如文件传输服务器 sftp-server。配置参数的值应是一个系统名与命令 (如果命令需要选项与参数, 必须同时给出), 能够基于客户系统的请求开始运行。sftp-server 命令实现了 sftp 文件传输子系统。注意, 这个配置参数仅适用于 SSH 协议版本 2
TCPKeepAlive	指定 sshd 是否向客户系统发送 TCP keepalive 消息。如果发送此消息, 服务器便能够及时了解客户的网络连接状态、关机或系统崩溃等情形。但这也意味着, 如果网络路由临时失灵, 导致网络连接暂时断开, 有可能会引起不必要的麻烦, 而这也许并非设置此参数之本意。从另一方面讲, 如果不发送 TCP keepalives 消息, sshd 可能会无限期地维持当前的网络会话, 空耗服务器的资源。这个配置参数的默认值是 yes, 即发送 TCP keepalive 消息, 以便服务器能够及时了解网络连接是否已经断开, 客户系统是否已经关机等, 从而避免无限期地维持无效的网络会话。为了禁止发送 TCP keepalive 消息, 可以把这个配置参数设为 no
UseLogin	指定 login 命令是否可用于交互式注册会话, 默认值是 no。注意, login 绝对不能用于远程命令执行
UsePAM	启用 PAM 插件式认证模块。如果把这个参数设置为 yes, 除了 PAM 账户与会话模块之外, PAM 认证还能够使用 ChallengeResponseAuthentication 与 PasswordAuthentication 模块处理所有的认证类型。由于 PAM 提示/应答式认证基本上等同于密码认证, 故应当禁用 PasswordAuthentication 或 ChallengeResponseAuthentication 配置参数。如果启用了 UsePAM, 则不能以普通用户的身份运行 sshd。这个配置参数的默认值是 no (Ubuntu Linux 系统的实际设置为 yes)

下面是取自 /etc/ssh/sshd\_config 配置文件的部分内容。

```
$ cat /etc/ssh/sshd_config
...
# What ports, IPs and protocols we listen for
Port 22
# Use these options to restrict which interfaces/protocols sshd will bind to
#ListenAddress ::

#ListenAddress 0.0.0.0
Protocol 2
# HostKeys for protocol version 2
HostKey /etc/ssh/ssh_host_rsa_key
HostKey /etc/ssh/ssh_host_dsa_key
...
# Authentication:
LoginGraceTime 120
PermitRootLogin yes
StrictModes yes

RSAAuthentication yes
PubkeyAuthentication yes
#AuthorizedKeysFile      %h/.ssh/authorized_keys
...
Subsystem sftp /usr/lib/openssh/sftp-server
UsePAM yes
$
```

根据表 19-1 中的说明以及 Ubuntu Linux 系统提供的/etc/ssh/sshd\_config 配置文件可知，sshd 将会监听本地系统所有 IP 地址的 22 号 TCP 端口，支持 OpenSSH 协议版本 2，允许超级用户 root 注册，采用 PAM 插件式认证模块，支持公钥认证与纯 RSA 认证，采用单独的子系统 sftp，采用 sftp-server 作为服务器，实现文件的安全传输。

### 19.1.3 使用 SSH 注册到远程系统

在 OpenSSH 中，ssh 是一个重要的客户端应用程序。利用 ssh，可以采用加密的通信方式，注册到远程系统，其语法格式简写如下。

```
ssh [options] [-l login_name] [user@]hostname [command]
```

其中，“-l login\_name”选项用于指定用户名，表示以哪一个用户身份注册到远程系统。如果不提供用户名，则以当前用户的身份注册到远程系统。例如，下列命令形式表示仍以本地系统的 gqxing 用户身份，采用默认的端口 22，注册到远程系统。（注意，从“Linux iscas...”一行开始，直至“http://help.ubuntu.com/”一行为止，这些内容出自/etc/motd 文件，之后的例子中均作省略。）

```
$ ssh iscas
gqxing@iscas's password:
Linux iscas 2.6.24-21-generic #1 SMP Mon Aug 25 17:32:09 UTC 2008 i686
```

```
The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/*copyright.
```

```
Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.
```

```
To access official Ubuntu documentation, please visit:
http://help.ubuntu.com/
Last login: Wed Nov 12 19:08:58 2008 from sinosoft
$
```

利用“-l”选项，可以使用不同的用户身份注册到远程系统。例如，下列命令意味着本地系统的当前用户以 cathy 的用户身份注册到远程系统 iscas。

```
$ ssh -l cathy iscas
cathy@iscas's password:
```



```
Linux iscas 2.6.24-21-generic #1 SMP Mon Aug 25 17:32:09 UTC 2008 i686
```

```
.....  
Last login: Wed Nov 12 18:45:23 2008 from sinosoft  
$
```

除了“-l”选项之外，为了以其他用户的身份注册到远程系统，还可使用“user@hostname”的形式注册到远程系统。

```
$ ssh gqxing@iscas  
gqxing@iscas's password:  
Linux iscas 2.6.24-21-generic #1 SMP Mon Aug 25 17:32:09 UTC 2008 i686
```

```
.....  
Last login: Wed Nov 12 15:45:11 2008 from sinosoft  
$
```

当第一次使用 ssh 注册到远程系统时，ssh 将会给出一个警告信息，提示用户确认连接的远程系统是否正确。如果用户回答 yes，ssh 将会在用户主目录的 ~/.ssh/known\_hosts 文件中存储远程系统的密钥，同时也会把客户端用户的密钥发送到远程系统，示例如下。

```
$ ssh iscas  
The authenticity of host 'iscas (169.254.78.100)' can't be established.  
RSA key fingerprint is 53:e1:e2:3f:c4:af:99:74:78:05:7a:dc:c2:25:4f:4c.  
Are you sure you want to continue connecting (yes/no)? yes  
Warning: Permanently added 'iscas' (RSA) to the list of known hosts.  
gqxing@iscas's password:  
Linux iscas 2.6.24-21-generic #1 SMP Mon Aug 25 17:32:09 UTC 2008 i686  
.....  
Last login: Mon Nov 10 12:11:06 2008 from sinosoft  
$
```

此后，当用户再次注册到同一远程系统时，就不会再出现上述提示信息了。

### 19.1.4 使用 ssh 执行远程系统中的命令

除了类似于 Telnet 的远程终端仿真功能之外，ssh 还能够在注册后执行远程系统中的单个命令后立即返回。具体的用法是，在 ssh 命令后面增加一条命令，命令前后使用双引号括起来。在下面的例子中，当前系统中的用户需要知道远程系统 iscas 的 Linux 内核版本号，因而需要在远程系统中运行“uname -r”。执行下列命令之后，返回的结果表明 iscas 系统 Linux 内核的版本号为 2.6.24-21。

```
$ ssh iscas "uname -r"  
gqxing@iscas's password:  
2.6.24-21-generic  
$
```

这种一次性地注册，执行远程命令，然后立即返回的功能是非常有用的。结合后面将要介绍的无密码注册功能，无论何时都能够非常容易地获取远程系统的各种状态信息。

### 19.1.5 使用 SCP 替代 FTP

从网络通信的角度来看，FTP 的数据传输方式是不安全的，这是因为 FTP 协议在网络中传输的用户名、密码和数据没有采取任何加密措施。比较安全的方法是采用 OpenSSH 的 SFTP（Secure FTP）和 SCP（Secure Copy）。

scp 是 OpenSSH 中的另一个重要客户端软件。Linux 系统中的 scp 命令具有在不同的系统之间复制文件的功能，其语法格式类似于常规的 cp 命令，如下所示。

```
scp [[user@]host1:]file1 [[user@]host2:]file2
```

其中，第一个参数是源文件，第二个参数是目的文件。当需要复制远程系统中的文件时，SCP 首先需要成功地注册到远程系统中，然后才能开始传输文件，因此要求提供远程系统的名字、用

户名和密码。在引用远程系统中的文件名时，文件名前面应加用户名与系统名前缀（两者之间需要插入一个“@”字符，之后再加一个冒号“：“）。远程文件名或目录名的表示形式如下。

```
username@servername:filename
username@servername:directoryname
```

其中，servername既可以是主机名，也可以是主机系统的IP地址。另外，除非引用的是用户主目录中的文件，指定文件时应给出绝对路径名或针对主目录的相对路径名。例如，如果远程系统的IP地址为172.16.3.40，为了访问其中的/etc/profile文件，可以使用“172.16.3.40:/etc/profile”的形式表示远程系统中的文件。为了访问远程系统guest用户主目录中的.profile文件，可以使用“guest@iscas:.profile”的形式表示其中的文件。

### 1. 利用 scp 下载文件

例如，要把远程系统中的/etc/profile文件复制本地系统的指定目录/tmp中，可以使用下列命令。

```
$ scp gqxing@iscas:/etc/profile /tmp
gqxing@iscas's password:
profile                                100%   497    0.5KB/s  00:00
$
```

假定已经把sshd守护进程监听的TCP端口修改为435。要使用这个端口，把远程系统中的一个文件复制到本地系统的指定目录中，可以使用下列命令。

```
$ scp -P 435 gqxing@iscas:/etc/profile /tmp
gqxing@iscas's password:
profile                                100%   497    0.5KB/s  00:00
$
```

### 2. 利用 scp 上传文件

反之，把本地Linux系统中的文件复制到远程系统也是一样的。例如，要把本地系统中的/etc/hosts文件复制到远程的/tmp目录中，可以使用下列命令。

```
$ scp /etc/hosts gqxing@iscas:/tmp
gqxing@iscas's password:
hosts                                100%   368    0.4KB/s  00:00
$
```

如上所述，要使用TCP端口435，把本地系统中的/etc/hosts文件复制到远程的/tmp目录中，可以使用下列命令。

```
$ scp -P 435 /etc/hosts gqxing@iscas:/tmp
gqxing@iscas's password:
hosts                                100%   368    0.4KB/s  00:00
$
```

## 19.1.6 使用 SFTP 替代 FTP

OpenSSH也提供一个SFTP程序。SFTP采用加密的SSH会话，实现了FTP的文件传输功能。由于SFTP也具有FTP的目录浏览功能，故SFTP比SCP更方便实用，尤其是在不知道要复制文件的确切位置时。但请注意，SFTP并不支持传统FTP的匿名注册功能。

下面是一个使用SFTP注册、查询命令的用法，然后下载文件的例子。

```
$ sftp iscas
Connecting to iscas...
gqxing@iscas's password:
sftp> help
Available commands:
cd path          Change remote directory to 'path'
lcd path          Change local directory to 'path'
```



```
chgrp grp path          Change group of file 'path' to 'grp'  
chmod mode path         Change permissions of file 'path' to 'mode'  
chown own path          Change owner of file 'path' to 'own'  
df [path]                Display statistics for current directory or  
                         filesystem containing 'path'  
help                     Display this help text  
...  
sftp> ls -l  
lrwxrwxrwx  1 gqxing  gqxing      26 Nov 10 13:37 Examples  
drwxr-xr-x  2 gqxing  gqxing     4096 Nov 12 16:16 conf  
drwxr-xr-x  2 gqxing  gqxing     4096 Nov 12 16:56 docs  
-rw-r--r--  2 gqxing  gqxing    20445 Nov 12 16:10 filelist  
drwxr-xr-x  2 gqxing  gqxing     4096 Nov 12 16:16 incl  
drwxr-xr-x  2 gqxing  gqxing     4096 Nov 10 18:20 script  
drwxr-xr-x  2 gqxing  gqxing     4096 Nov 11 19:16 src  
...  
sftp> lcd /tmp  
sftp> get filelist  
Fetching /home/gqxing/filelist to filelist  
/home/gqxing/filelist           100%   20KB  20.0KB/s  00:00  
sftp> exit  
$
```

### 19.1.7 SSH 与 SCP 的无密码注册

常规情况下，每次使用 ssh 注册或使用 scp 复制远程系统的文件时，都需要提供密码，然后才能做进一步的处理。为了省略输入密码这一步骤，有时也可以采用 Shell 脚本的方法解决，但这需要把手工输入的密码（即明码）放在脚本文件中。

利用密钥配置文件，OpenSSH 可以使 ssh 远程注册与 scp 文件复制的操作过程中省略密码验证的中间环节。为此，ssh 客户端首先应建立 OpenSSH 连接，然后自动地向服务器发送其密钥（公钥）。之后，服务器即可根据相应用户主目录中预定义的密钥列表，对收到的密钥进行比较。如果存在匹配的密钥，服务器将会允许 ssh 或 scp 自动注册。

密钥文件需要单独生成，生成后的密钥文件存储在服务器用户主目录的`~/.ssh`子目录中。其中，私钥和公钥分别存储在`id_dsa`和`id_dsa.pub`文件中，而`authorized_keys`文件则用于存储所有授权的远程客户系统的公钥，使得远程客户系统能够以此用户的身份注册到本地系统而无需提供密码。

为了实现 ssh 与 scp 的无密码注册，需要事先生成并安装远程注册与数据传输的密钥，其唯一的限制是需要把密钥绑定到两个系统的 IP 地址上。然后，服务器即可使用预安装的密钥，对每一次的远程注册和文件传输进行验证。但是，这一功能特性并不适合于利用 DHCP 协议动态分配 IP 地址，致使 IP 地址经常发生变化的情况。

但是，由于只需知道用户名即可实现远程注册和文件传输，而无需提供密码，这一功能特性是有安全隐患的。因此，在具体实现时，在服务器与客户端之间最好采用普通用户账号，以免造成较大的破坏。

下面以`gqxing`用户为例，详细说明在实现 ssh 与 scp 的无密码注册之前，OpenSSH 的客户端与服务器双方事先都需要做哪些准备工作。

#### 1. OpenSSH 客户端配置

要实现无密码的系统注册，OpenSSH 客户端需要完成下列准备工作。

首先，在客户端与服务器系统中分别创建一个同名的`gqxing`用户，然后以`gqxing`用户的身份注册到客户端系统，使用`ssh-keygen`命令生成一对密钥。当系统提示输入一个与密钥相关联的密码时，直接按下 Enter 键。示例如下。

```
$ ssh-keygen -t dsa
Generating public/private dsa key pair.
Enter file in which to save the key (/home/gqxing/.ssh/id_dsa): <Enter>
Enter passphrase (empty for no passphrase): <Enter>
Enter same passphrase again: <Enter>
Your identification has been saved in /home/gqxing/.ssh/id_dsa.
Your public key has been saved in /home/gqxing/.ssh/id_dsa.pub.
The key fingerprint is:
4c:57:c4:5e:b5:41:af:12:25:1b:ad:8f:16:98:bf:d5 gqxing@sinosoft
$
```

其次，验证上述步骤生成的两个密钥文件是否均存储在 gqxing 用户主目录的.ssh 子目录中。其中，id\_dsa 文件中存储的是私钥，id\_dsa.pub 文件中存储的是公钥（用于提交远程服务器，解密客户端传输的加密数据）。

```
$ ls -l .ssh
总用量 12
-rw----- 1 gqxing gqxing 668 2008-11-12 23:19 id_dsa
-rw-r--r-- 1 gqxing gqxing 602 2008-11-12 23:19 id_dsa.pub
-rw-r--r-- 1 gqxing gqxing 1328 2008-11-12 21:05 known_hosts
$
```

最后，采用下列命令，把生成的公钥文件 id\_dsa.pub 复制到远程系统的 gqxing 用户主目录中：

```
$ cd .ssh
$ scp id_dsa.pub gqxing@iscas:pub_key
gqxing@iscas's password:
id_dsa.pub                                100%   602      0.6KB/s   00:00
$
```

## 2. OpenSSH 服务器配置

在 OpenSSH 服务器中，以 gqxing 用户的身份注册到系统，然后使用 cat 命令和“>>”重定向符号，把 pub\_key 文件中的数据附加到 authorized\_keys 文件的后面。

```
$ cat ~/pub_key >> .ssh/authorized_keys
$ rm ~/pub_key
$
```

authorized\_keys 文件包含所有 OpenSSH 客户端系统的公钥列表，如果表中列举的客户系统中的 gqxing 用户仍以同一用户身份连接到服务器，则无需提供密码。

## 3. 示例

在完成上述的全部设置之后，当 OpenSSH 客户端的 gqxing 用户仍以 gqxing 用户的身份，使用 ssh、scp 或 sftp 连接到远程主机，就不会再提示用户输入密码了。示例如下。

```
guest@sinosoft:~$ ssh iscas
Linux iscas 2.6.24-21-generic #1 SMP Mon Aug 25 17:32:09 UTC 2008 i686
...
Last login: Wed Nov 12 22:25:08 2008 from sinosoft
gqxing@iscas:~$
```

下面是一个利用 scp 命令复制文件的例子，其情况同 ssh 命令一样，远程服务器也不要求用户提供密码。

```
gqxing@sinosoft:~$ scp iscas:/etc/hosts .
hosts                                         100%   368      0.4KB/s   00:00
gqxing@sinosoft:~$
```

但这一方法对客户端的其他用户（包括超级用户）无效，即使他们也以 gqxing 用户的身份注册到远程服务器，仍需提供密码，示例如下。

```
cathy@sinosoft:~$ ssh -l gqxing iscas
gqxing@iscas's password:
Linux iscas 2.6.24-21-generic #1 SMP Mon Aug 25 17:32:09 UTC 2008 i686
```



```
Last login: Wed Nov 12 23:20:10 2008 from sinosoft  
gqxing@iscas:~$
```

### 19.1.8 OpenSSH 的安全考虑

从表 19-1 中可知，Ubuntu Linux 系统中的 sshd 都会监听本地系统所有 IP 地址的 22 号 TCP 端口。为安全起见，可以把默认的 22 号端口改为自己约定的其他空闲端口号（如 435 等）。同时，还需要把/etc/services 文件中的下列两行端口定义

```
ssh      22/tcp  
ssh      22/udp
```

改为

```
ssh      435/tcp  
ssh      435/udp
```

此外，也可以使用 AllowUsers、AllowGroups、DenyGroups 以及 DenyUsers 配置参数，或者这些参数的组合，限定用户或用户组的访问。例如，为了限定只有 gqxing 和 cathy 两个用户能够访问系统，可以在/etc/ssh/sshd\_config 配置文件中增加下列配置参数。

```
AllowUsers gqxing cathy
```

重新启动 sshd 之后，除了 gqxing 与 cathy 两个用户之外，系统将会拒绝接受其他用户的注册，并输出拒绝访问的错误信息，示例如下。

```
$ ssh -l guest iscas  
guest@iscas's password:  
Permission denied, please try again.  
guest@iscas's password: <Enter>  
Permission denied (publickey,password).  
$
```

修改 OpenSSH 配置文件之后，为使新的设置立即生效，需要重新启动 sshd 守护进程。示例如下。

```
$ sudo /etc/init.d/ssh restart  
* Restarting OpenBSD Secure Shell server shhd [ OK ]  
$
```

## 19.2 Telnet 远程注册

尽管 OpenSSH 可以取代 telnet，telnet 也确实存在安全方面的潜在问题，但并不是每一个系统都直接支持 OpenSSH。如在 Windows 系统中，如果想用 OpenSSH，需要从网上下载免费的开源软件，故 telnet 仍然是应用比较广泛的通信手段之一。因此，本节将简单介绍怎样在 Linux 系统中使用 telnet。

在 Ubuntu Linux 系统中，telnet 等传统的网络应用通常分为两个软件包：客户端软件包（如 telnet，其中含有 telnet 应用程序及其参考手册等）和服务器软件包（如 telnetd，其中含有 in.telnetd 服务程序及其配置文件等）。新的 Ubuntu Linux 系统安装介质通常仅提供客户端软件包，因而不会安装网络服务器软件包。

由此可见，Linux 系统中的用户通常只能利用 telnet 注册到其他系统（如 UNIX 系统），不能注册到初始安装的任何 Ubuntu Linux 系统，甚至也不能使用 telnet 注册到自己的 Ubuntu Linux 系统中。

如同第 14 章中所述，大部分系统守护进程（包括 sshd 等网络守护进程）通常都是由/etc/init.d

目录中的 Shell 脚本直接控制，随着系统的启动而运行，随着系统的关机而停止的。但是，传统的网络守护进程是由 inetc 统一控制的，而且按照实际的需要和请求才会启动。inetd 网络总控进程是由/etc/init.d 目录中的 Shell 脚本 openbsd-inetc 控制的，也会随着系统的启动而自动运行。

在系统启动之后的运行过程中，为了查询 inetc 当前的运行状态，可以使用下列命令（参见第 9 章）。

```
$ pidof inetc
6387
$
```

## 19.2.1 设置 Telnet 服务器

由于 Ubuntu Linux 系统的安装程序不会自动安装 telnet 服务器软件包，为使 Ubuntu Linux 系统支持 telnet 远程注册访问，可用 apt-get、aptitude 和 synaptic 等软件工具安装 telnetd 服务器软件包，示例如下。

```
# aptitude install telnetd
```

在安装 telnet 服务器软件包之后，可以参照下列步骤设置 telnet 服务器。

(1) 检查/etc/inetc.conf 配置文件，确保文件包含下列内容。

```
$ cat /etc/inetc.conf
telnet stream tcp nowait telnetd /usr/sbin/tcpd /usr/sbin/in.telnetd
$
```

(2) 使用下列命令启动 inetc 守护进程，从而激活 telnetd 服务器（参见第 18 章）。

```
$ sudo /etc/init.d/openbsd-inetc start
* Starting internet superserver inetc                                [ OK ]
$
```

在完成上述设置之后，只要拥有注册的用户账号，任何人都可以利用 telnet 命令注册，访问 Linux 主机系统。

下面的例子给出了一个完整的注册过程。其中，iscas 系统中的普通用户以 gqxing 的用户身份注册到远程系统 iscas，执行一个 uname 命令后立即退出远程系统。

```
$ telnet iscas
Trying 169.254.78.100...
Connected to iscas.
Escape character is '^]'.
Ubuntu 8.04.1
iscas login: gqxing
Password:
Last login: Wed Nov 12 16:19:15 CST 2008 from sinosoft
Linux iscas 2.6.24-21-generic #1 SMP Aug 25 17:32:09 UTC 2008 i686
...
$ uname -n
iscas
$ exit
logout
Connection closed by foreign host.
$
```

但是，这种注册访问仅限于普通用户账号，即只能使用普通用户账号注册到 Linux 系统，不能以超级用户 root 的身份直接注册。要直接使用 root 注册（通常不建议这样做），尚需做进一步的设置。

通常，即使 Ubuntu Linux 系统允许超级用户在控制台上注册，也不允许任何人在控制台之外的其他终端上，以超级用户 root 的身份注册到系统中。这一限制是由/etc/securetty 控制文件实现的。securetty 文件包含一系列设备名（其中省略了“/dev/”前缀），表示只有在该文件列举的设备上使用 root 注册才是可以接受的。默认的/etc/securetty 文件内容如下，其中列举的设备名实际上都是系统控制台。

```
$ cat /etc/securetty
console
```



```
....  
tty1  
tty2  
....  
tty63  
vc/1  
vc/2  
....  
vc/63  
$
```

当使用 telnet 注册时，不管是在同一个系统中，还是在不同的 Linux 系统之间，其设备名通常均为 pts/n，其中的 n 是 0、1、2……中的一个数字。为了突破上述限制，直接使用超级用户 root 注册，可以在 security 文件中增加下列内容。

```
pts/0  
pts/1  
pts/2  
....
```

之后，即可使用 root，以超级用户的身份注册到远程系统，如下所示。

```
$ telnet iscas  
Trying 169.254.78.100...  
Connected to iscas.  
Escape character is '^]'.  
Ubuntu 8.04.1  
iscas login: root  
Password:  
Last login: Wed Nov 12 16:19:15 CST 2008 from sinosoft  
Linux iscas 2.6.24-21-generic #1 SMP Aug 25 17:32:09 UTC 2008 i686  
....  
$
```

## 19.2.2 Telnet 服务器的安全考虑

尽管 telnet 不安全，但仍可采用一些补救的措施，例如，采用非常规的 TCP 端口等。实际上，这是一项简单的措施，并不能确保系统的绝对安全。

要采用 23 之外的其他 TCP 端口，如 4 096，可以编辑/etc/services 文件，把下列 TCP 端口定义

```
telnet      23/tcp
```

改为

```
telnet      4096/tcp
```

重新启动 inetd 守护进程，然后使用 netstat 命令，验证 telnet 服务器当前监听的 TCP 端口是否为 4 096。

```
$ netstat -an | grep 4096  
tcp      0      0  ::::4096      ::::*          LISTEN  
$
```

在完成上述设置步骤之后，即可利用下列命令与新设的 TCP 端口 4 096，注册到 telnet 服务器，示例如下。

```
$ telnet iscas 4096  
Trying 169.254.78.100...  
Connected to iscas.  
Escape character is '^]'.  
Ubuntu 8.04.1  
iscas login: gqxing  
Password:  
Last login: Wed Nov 12 16:19:15 CST 2008 from sinosoft  
Linux iscas 2.6.24-21-generic #1 SMP Aug 25 17:32:09 UTC 2008 i686  
....  
$
```

此外，为了确保数据库服务器或业务主机的系统安全，严禁任何用户使用 telnet 远程访问系统。还可以利用编辑器，在/etc/inetd.conf 配置文件中的 telnet 一行增加注释字符“#”，重新启动 inetd 守护进程，从而禁止 inetd 调用 telnetd，封锁 telnet 网络服务，关闭主机系统的远程访问功能，防止 telnetd 响应用户的 telnet 访问请求（参见第 18 章）。

## 19.3 FTP 文件传输

FTP (File Transfer Protocol) 是最常用的一种文件传输工具。作为一个基本的组成部分，大多数操作系统均提供完整的 FTP 网络应用，Linux 系统也不例外。Ubuntu Linux 系统采用安全的 FTP 守护进程 vsftpd，支持 FTP 服务器功能。

### 19.3.1 设置 vsftpd

同其他 telnet 网络应用一样，初始安装的 Ubuntu Linux 基本系统总是包含 FTP 客户端软件 ftp，但不包括服务器的软件包 vsftpd。为了提供 FTP 服务器功能，需要使用 apt-get、aptitude 和 synaptic 等软件工具，安装 vsftpd 服务器软件包，示例如下。

```
$ sudo aptitude install vsftpd
```

一旦安装 vsftpd 服务器软件包之后，/etc/init.d 目录中将会增加一个 vsftpd 脚本文件，并会在系统的启动过程中自动运行 vsftpd 守护进程。

要查询 vsftpd 守护进程当前的运行状态，可以使用下列 pidof 命令获取指定进程的 PID(参见第 9 章)。

```
$ pidof vsftpd
5551
$
```

也可以使用 netstat 命令，查询系统当前正在监听的所有 TCP 和 UDP 端口，检查其中是否存在 ftp。如果 vsftpd 守护进程没有运行，下列 netstat 命令不会产生任何输出信息。

```
$ netstat -a | grep ftp
tcp      0      0  *:ftp          *:*      LISTEN
$
```

vsftpd 守护进程采用/etc/vsftpd.conf 配置文件规范其处理动作，其中包含大量的运行时配置信息，以及以注释符号“#”为起始字符的注释行与配置示例，注释行提供了许多有用的解释。必要时，只需删除配置示例前的注释符号即可激活相应的配置变量。在修改 vsftpd 服务器的配置文件之后，为使修改后的设置立即生效，需要重新启动 vsftpd 守护进程。为此，可以使用下列命令，重新启动 vsftpd 守护进程。

```
$ sudo /etc/init.d/vsftpd restart
* Stopping FTP server: vsftpd                                [ OK ]
* Starting FTP server: vsftpd                                [ OK ]
$
```

通常，vsftpd 守护进程拒绝超级用户 root 访问 FTP 服务器。为了取消这一限制，可以编辑 /etc/ftpusers 文件，在 root 之前插入一个注释符号“#”。

### 19.3.2 vsftpd.conf 配置文件

在开始运行时，vsftpd 守护进程将会读取其配置文件，按照配置文件中的参数设置采取相应



的处理动作。vsftpd 的默认配置文件为 /etc/vsftpd.conf，必要时也可以在 vsftpd 的命令行参数中指定其他配置文件。

vsftpd.conf 配置文件的语法格式非常简单，每一行或者是一个注释行，或者是一个参数设置，以注释符号“#”为起始字符的注释行将被忽略。vsftpd.conf 配置文件参数设置的语法格式如下。

*parameter=value*

注意，在设置配置参数时，等号“=”前后不能加任何空格字符。vsftpd.conf 配置文件中的每个配置参数都有一个默认的设置。如有必要，可以做适当的修改（在修改任何配置参数之后，不要忘记重新启动 /etc/init.d/vsftpd 脚本）。

在使用或修改 vsftpd 的配置文件之前，首先需要了解 vsftpd 守护进程的默认处理方式，下面是其中的一部分。

- vsftpd 支持 FTP 匿名访问。如果想要禁止匿名访问，可把 anonymous\_enable 参数设置为 NO。同时还需要删除 local\_enable 参数前面的注释符号“#”，使远程用户能够利用服务器系统中的用户账号访问 FTP 服务器。
- vsftpd 只允许 FTP 匿名用户下载文件，但不能上传文件。如果允许上传文件，需要把 anon\_upload\_enable 设置为 YES（同时还需把 write\_enable 参数设置为 YES）。此外还需使用下列命令，修改 FTP 匿名用户主目录/home/ftp 中的 pub 共享目录的访问权限；否则，/home/ftp 与 /home/ftp/pub 目录的默认访问权限不允许其他用户创建文件，因而无法上传文件。

# chmod 777 /home/ftp/pub

- vsftpd 禁止匿名用户在 FTP 服务器中创建子目录。如果允许匿名用户创建子目录，需要删除 anon\_mkdir\_write\_enable 参数前面的注释符号“#”（同时还需把 write\_enable 参数设置为 YES）。同样，还需要使用下列命令修改 FTP 匿名用户主目录/home/ftp 中的 pub 共享目录的访问权限；否则，/home/ftp 和 /home/ftp/pub 目录的访问权限不允许其他用户创建子目录。

# chmod 777 /home/ftp/pub

- vsftpd 采用 /var/log/vsftpd.log 文件作为 FTP 服务器默认的日志文件。如有必要，也可以利用 xferlog\_file 配置参数，改换为其他文件。
- FTP 匿名用户的主目录为 /home/ftp。如想改用其他目录，需要修改 anon\_root 的参数设置。FTP 的匿名访问总是具有一定的风险，如果采用默认值，且允许上传文件，用户能够随心所欲地把文件写到 FTP 公共目录中，从而把整个 /home 文件系统分区耗费殆尽。因此，最好的办法是把 FTP 匿名用户主目录改换到一个专用的文件系统分区中。

有关 vsftpd.conf 配置文件参数设置的更多说明，参见表 19-2。

表 19-2 vsftpd.conf 配置文件的部分配置参数

配置参数	默认值	简单说明
anon_mkdir_write_enable	NO	指定是否允许匿名用户创建新的子目录。如果设为 YES，意味着允许匿名用户在 FTP 匿名用户主目录的共享子目录 pub 中创建新目录。启用这个功能特性要求同时启用 write_enable，且 FTP 匿名用户还应具有写共享子目录 pub 的访问权限
anon_other_write_enable	NO	如果设为 YES，意味着允许匿名用户执行除上传文件和创建目录之外的其他写操作，如删除文件、重新命名文件等。通常不建议启用这个功能特性
anon_upload_enable	NO	指定是否允许 FTP 匿名用户上传文件。如果设为 YES，意味着允许匿名用户上传文件。启用这个功能特性要求同时启用 write_enable 配置参数，且 FTP 匿名用户还应具有写上传目录（如 pub）的访问权限

续表

配置参数	默认值	简单说明
anon_world_readable_only	YES	如果设置为 YES，匿名用户只能下载任何用户均可读的文件
anonymous_enable	YES	指定是否允许匿名注册访问。如果启用，意味着可以使用用户名 anonymous 或 ftp 匿名注册
ascii_download_enable	NO	启用时，表示允许采用 ASCII 模式下载数据文件。通常均采用二进制模式传输数据
ascii_upload_enable	NO	启用时，表示允许采用 ASCII 模式上传数据文件。通常均采用二进制模式传输数据
background	NO	启用时，vsftpd 守护进程将会在启动后进入“监听”模式，以后台方式运行。也就是说，在 vsftpd 开始运行之后，其控制权将会立即转交给 Shell
dirlist_enable	YES	指定是否允许用户使用目录文件的列表显示命令（如 dir 或 ls 等）。如果设为 NO，表示拒绝用户使用任何目录文件的列表显示命令
download_enable	YES	指定是否允许用户下载文件。如果设为 NO，表示拒绝执行任何文件下载请求
force_dot_files	NO	指定是否在列表显示目录文件时总是显示以句点“.”为起始字符的隐藏文件和目录，而不管是否采用“-a”选项。注意，这个配置参数不影响“.”和“..”两个特殊目录项的显示规则
guest_enable	NO	如果设为 YES，vsftpd 将会把所有非匿名用户看作 guest 用户，而 guest 用户将映射为 guest_username 配置参数指定的用户，即 ftp
hide_ids	NO	如果设置为 YES，在显示目录文件列表时，所有的用户与用户组均显示作 ftp
listen	NO（实为 YES）	启用时，vsftpd 可以采用独立运行模式，这意味着无需由 inetd 控制，可以直接运行 vsftpd，由 vsftpd 自己负责监听和处理 FTP 访问服务
local_enable	NO	控制是否允许本地系统中的注册用户访问 FTP 服务器。如果设为 YES，意味着可以使用服务器所在系统/etc/passwd 文件中的普通用户账号访问 FTP 服务器。如果允许以非匿名方式访问 FTP 服务器，这个参数必须设置为 YES
log_ftp_protocol	NO	启用时，vsftpd 将会采用标准的 xferlog 格式，记录所有的 FTP 请求与响应等日志信息。这个配置参数主要用于调试的目的
ls_recurse_enable	NO	指定是否允许递归地列表显示目录中的所有文件。如果设为 YES，表示允许用户使用“ls -R”命令
no_anon_password	NO	指定匿名注册时是否提示用户输入密码。如果设为 YES，vsftpd 不会提示用户输入匿名密码，这意味着输入 ftp 或 anonymous 匿名用户名之后将会直接注册到 FTP 服务器
syslog_enable	NO	如果设为 YES，通常写入/var/log/vsftpd.log 日志文件的任何日志信息将会转而写入系统日志文件中
tilde_user_enable	NO	如果设置为 YES，vsftpd 将会尝试解析“~username/dirfile”形式的路径名。注意，vsftpd 通常总是解析“~”和“~/something”形式的路径名，其中波浪号“~”表示用户的主目录
use_localtime	NO	如果设置为 YES，vsftpd 将以本地时区的时间显示目录文件列表。默认的时区为 GMT
userlist_deny	YES	如果 userlist_enable 设为 YES，需要进一步考察 userlist_deny 配置参数。如果 userlist_deny 采用默认值 YES，vsftpd 将会拒绝 user_list（参见 userlist_file 配置参数）文件中列举的用户注册访问；如果设为 NO，则允许 user_list 文件中列举的用户注册访问，而拒绝其他用户注册访问。在拒绝用户注册时，vsftpd 将会直接输出拒绝信息，而不会提示用户输入密码
userlist_enable	NO	如果设为 YES，vsftpd 守护进程将会从 userlist_file 配置参数指定的文件（即/etc/vsftpd/user_list）中加载用户名列表。如果试图使用其中列举的某个用户名注册，将会立即遭到拒绝，而且根本就不提供密码验证的机会。这样做是非常有用的，可以防止明文密码的传输。参见 userlist_deny 配置参数



续表

配置参数	默认值	简单说明
write_enable	NO	用于控制用户是否能够使用与文件写操作有关的 FTP 子命令，其中包括 STOR、DELE、RNFR、RNTO、MKD、RMD、APPE 和 SITE。实际上，这个配置参数主要用于间接控制用户是否能够上传文件，在 FTP 服务器中创建目录，以及删除文件等
xferlog_enable	NO（实为 YES）	如果设为 YES，vsftpd 将会利用日志文件详细记录与文件上传和下载有关的信息。vsftpd 守护进程默认的日志文件为 /var/log/vsftpd.log，但可以利用配置参数 vsftpd_log_file 指定一个新的日志文件
xferlog_std_format	NO	如果设为 YES，vsftpd 将会按照标准的 xferlog 格式把文件传输的日志信息记录到 /var/log/xferlog 日志文件中。如果设为 NO，其数据格式的可读性更强
anon_max_rate	0（没有限制）	对于匿名用户，vsftpd 允许的最大数据传输速率（字节/秒）
data_connection_timeout	300（实为 120）	以秒为单位的超时值。这是在 FTP 数据传输期间 vsftpd 允许的最大停顿时间，如果出现超时，vsftpd 将会断开用户的 FTP 连接
file_open_mode	0666	上传文件时赋予新建文件的访问权限。如果上传的文件大多为可执行程序，应把这个参数设置为 0777
idle_session_timeout	300（实为 600）	以秒为单位的超时值。这是 vsftpd 在用户执行 FTP 命令期间允许的最大空闲时间，如果出现超时，vsftpd 将会断开用户的 FTP 连接
listen_port	21	如果采用独立运行模式，vsftpd 守护进程将会采用这个参数定义的端口监听用户的 FTP 连接请求
local_max_rate	0（没有限制）	指定 vsftpd 允许本地认证用户的最大数据传输速率（字节/秒）
max_clients	0（没有限制）	指定 vsftpd 能够接受的最大 FTP 连接数量。超出此限的 FTP 连接请求将会收到一个出错信息
max_loginfails	3	指定最大的注册尝试次数。当注册失败超过指定的次数，FTP 将会终止注册会话
max_per_ip	0（没有限制）	指定 vsftpd 能够接受同一 IP 地址 FTP 连接的最大数量。超出此限的 FTP 连接请求将会收到一个出错信息
anon_root	/home/ftp	这个配置参数用于指定匿名用户的主目录。在匿名用户注册成功之后，vsftpd 将会使用这个目录作为初始工作目录
banner_file	无	这个配置参数用于指定一个文件名，其中包含当用户注册到 FTP 服务器时显示的信息。如果设定了此配置参数，指定文件中的内容将会取代 ftpd_banner 配置参数提供的字符串信息
deny_file	无	这个配置参数用于设置一个文件名（或目录名）模式，表示根本不允许访问匹配指定模式的目录文件，如拒绝下载匹配的文件、拒绝改换到匹配的目录以及拒绝访问其中的文件等。注意，vsftpd 能够处理的正则表达式只是一个简单的实现，或者说，只是正则表达式的一个子集。因此，设置后需要仔细地测试。此外，建议尽可能采用文件系统的访问权限设置。vsftpd 能够识别星号“*”、问号“？”和花括号“{}”，而且匹配检查仅限于路径名的文件名部分，如 a/b/?，但不支持 a/?/c。例如，deny_file={*.mp3,*.mov} 表示拒绝访问其扩展名为.mp3 和.mov 的任何文件
ftp_username	ftp	指定用作 FTP 匿名访问的用户名，其主目录也是 FTP 匿名用户能够访问的根目录
ftpd_banner	(vsFTPD 2.0.6)	定义一个字符串，如“Welcome to FTP service”，作为建立 FTP 连接后的欢迎信息，以代替 vsftpd 提供的默认信息“(vsFTPD 2.0.6)”，其中 2.0.6 是 vsftpd 当前的版本号
guest_username	ftp	用于定义“guest” 用户映射的真实用户，以统一规范非匿名用户的访问权限
hide_file	无	这个配置参数用于设置一个文件名（或目录名）模式，表示在显示目录文件列表时隐藏匹配指定模式的目录文件。其注意事项参见 deny_file 配置参数的说明

续表

配置参数	默认值	简单说明
listen_address	none	如果 vsftpd 处于独立运行模式，可以使用这个配置参数指定一个 IP 地址，强制 vsftpd 监听指定的地址（通常，vsftpd 将会监听所有的本地网络接口）
local_root	\$HOME (用户主目录)	这个配置参数用于确定在非匿名的本地用户注册之后，vsftpd 将引导用户进入的工作目录
userlist_file	/etc/vsftpd.user_list	如果 userlist_enable 设为 YES，vsftpd 将会使用 userlist_file 设定的文件作为用户访问控制文件的名字，加载其中的用户列表
vsftpd_log_file	/var/log/vsftpd.log	指定 vsftpd 常规日志文件的名字。注意，仅当配置参数 xferlog_enable 设为 YES，xferlog_std_format 设为 NO 时，才会使用这个参数指定的文件记录 vsftpd 的常规日志信息。如果 syslog_enable 配置参数设为 YES，日志信息将会写到系统日志文件，而非 vsftpd 自己的日志文件。
xferlog_file	/var/log/vsftpd.log	指定文件传输日志文件的名字。注意，仅当配置参数 xferlog_enable 设为 YES 之后，vsftpd 才会使用这个文件记录文件传输的日志信息

在完成上述设置，以及适当地修改配置文件之后，即可开始利用 `ftp` 命令传输文件了。示例如下。

```
$ ftp iscas
Connected to iscas.
220 (vsFTPD 2.0.6)
Name (iscas:gqxing):
331 Please specify the password.
Password:
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp>
```

### 19.3.3 FTP 安全考虑

FTP 具有一定的安全缺陷，但可以采取必要的措施加以克服。例如，可以限制 Linux 系统用户访问非匿名的 FTP 等。但需要注意的是，在用户注册和数据传输过程中，FTP 并未采取任何加密措施。

#### 1. /etc/ftpusers 文件

为了实现 FTP 的访问控制，可以采用 `/etc/ftpusers` 文件，限制指定的用户访问 FTP 服务器。在安装 `vsftpd` 软件包之后，其中提供的 `ftpusers` 文件通常包含超级用户 `root`，以及 `daemon`、`bin` 与 `sys` 等系统用户。

为了限制其他用户访问 FTP 服务器，可以把相应的用户名加到 `ftpusers` 文件中。其中，每个用户名占用一行。如果允许用户访问 FTP 服务器，可以在用户名前增加注释符号“#”，或者从 `ftpusers` 文件中直接删除相应的用户名。

下面是一个 `ftpusers` 文件的例子。其中，由于用户名 `root` 之前增加了一个注释字符“#”，因而允许超级用户 `root` 访问 FTP 服务器，但禁止 `guest` 和 `visitor` 等用户访问。

```
# cat /etc/ftpusers
# /etc/ftpusers: list of users disabled. See ftpusers(5).

# root
# guest
# visitor
#
```



在数据传输期间，FTP 不会加密用户提供的密码，因而有可能增加数据或密码在网上传输的风险。最好的做法是继续保持 `ftpusers` 文件中原有的用户设置，尤其不要为超级用户 `root` 开禁。为了安全起见，还可以再增加其他用户名。

### 2. `/etc/user_list` 文件

`vsftpd.user_list` 文件具有比 `ftpusers` 文件更灵活的访问控制措施。取决于 `vsftpd.conf` 配置文件中 `vsftpd.userlist_deny` 参数的设置，可以拒绝 `vsftpd.user_list` 文件中列举的用户访问 FTP 服务器，也可以允许 `vsftpd.user_list` 文件中列举的用户访问 FTP 服务器。如果采用 `vsftpd.user_list` 文件控制 FTP 服务器的访问，通常可以不必考虑 `ftpusers` 文件。尤其是，如果采用 `vsftpd.user_list` 文件允许列举的用户访问 FTP 服务器的服务，相应的用户名不应出现在 `ftpusers` 文件中。

是否采用 `vsftpd.user_list` 文件控制 FTP 服务器的访问，取决于 `vsftpd.conf` 配置文件中的两个参数设置——`userlist_enable` 与 `userlist_deny`。`userlist_enable` 参数的默认值为 NO，表示不使用 `vsftpd.user_list` 文件。如果启用了 `userlist_enable`，还需要进一步考察 `userlist_deny` 参数。`userlist_deny` 的默认值为 YES，表示 FTP 服务器会拒绝 `vsftpd.user_list` 文件中列举的用户访问；如果把 `userlist_deny` 设置为 NO，意味着允许 `vsftpd.user_list` 文件中列举的用户访问 FTP 服务器，而拒绝其他用户访问。在拒绝用户访问 FTP 服务器时，`vsftpd` 将会直接输出拒绝信息，而不会提示用户输入密码。

即使安装了 `vsftpd` 软件包，Ubuntu Linux 系统也不提供 `vsftpd.user_list` 文件，必要时可由网络管理员自行创建。如同 `ftpusers` 文件，`vsftpd.user_list` 文件通常应包含超级用户 `root`，以及 `bin`、`daemon` 与 `adm` 等系统用户。根据需要，可以增加新的用户名，或者删除（注销掉）其中的任何用户。

### 3. 匿名上传文件

如果匿名用户想把文件上传到 FTP 服务器，首先应修改 `/home/ftp/pub` 目录的访问权限，然后按照下列步骤，在 `/home/ftp/pub` 目录中创建一个具有写访问权限的共享子目录（这将允许用户上传自己的文件，但不能访问其他用户上传的文件）。

```
$ sudo chmod 777 /home/ftp/pub  
$ sudo mkdir /home/ftp/pub/upload  
$ sudo chmod 722 /home/ftp/pub/upload  
$
```

### 4. 使用 SCP 替代 FTP

FTP 的一大缺点是采用明码方式传输数据，包括用户提交的用户名和密码，使用户的账号等信息容易遭受非法侵袭和窃取。SCP（Secure Copy）与 SFTP（Secure FTP）则能够提供加密的数据传输功能，因而可用以替代 FTP。但美中不足的是，SCP 并不支持 FTP 的匿名用户访问功能。

## 19.3.4 FTP 应用

在传输文件时，可以使用 `ftp` 的 `get`（下载）或 `put`（上传）等 FTP 子命令。要查询 FTP 服务器当前目录中存在的文件，然后下载其中的某个文件，可以使用 `ls` 或 `dir` 等 FTP 子命令，示例如下。

```
ftp> ls  
200 PORT command successful. Consider using PASV.  
150 Here comes the directory listing.  
lrwxrwxrwx 1 1000 1000 26 Jun 27 16:40 Examples ->/usr/share/example-content  
drwxr-xr-x 2 1000 1000 4096 Jun 28 10:49 script  
drwxr-xr-x 2 1000 1000 4096 Jun 28 10:49 src  
-r--r--r-- 1 1000 1000 2032 Sep 20 10:49 typescript  
.....
```

```

226 Directory send OK.
ftp> dir
200 PORT command successful. Consider using PASV.
150 Here comes the directory listing.
lrwxrwxrwx  1 1000    1000     26 Jun 27 16:40 Examples ->/usr/share/example-content
drwxr-xr-x  2 1000    1000     4096 Jun 28 10:49 script
drwxr-xr-x  2 1000    1000     4096 Jun 28 10:49 src
-rwxr--r--  1 1000    1000    2634 Sep 20 10:49 typescript
....
226 Directory send OK.
ftp> lcd /tmp
local directory now /tmp
ftp> get typescript
local: typescript remote: typescript
200 PORT command successful. Consider using PASV.
150 Opening BINARY mode data connection for typescript (2634 bytes).
226 File send OK.
2634 bytes received in 0.02 secs (107.9 kB/s)
ftp>

```

当需要一次传输多个文件时，可以使用 FTP 的 mget 或 mput 子命令。但在使用 mget/mput 命令时，ftp 通常总是在传输每一个文件之前请用户确认一次。如果文件太多，这一确认方式将使用户不胜其烦。为了避免这一麻烦，可以使用 ftp 的“-i”选项，关闭提示与确认模式，示例如下。

```

$ ftp -i iscas
Connected to iscas.
220 (vsFTPd 2.0.6)
Name (icas:gqxing):
331 Please specify the password.
Password:
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> cd /usr/include/sys
250 Directory successfully changed.
ftp> lcd tmp
Local directory now /tmp
ftp> mget *.h
...
local: wait.h remote: wait.h
200 PORT command successful. Consider using PASV.
150 Opening BINARY mode data connection for wait.h (6472 bytes).
226 File send OK.
6472 bytes received in 0.00 secs (15120.4 kB/s)
local: xattr.h remote: xattr.h
200 PORT command successful. Consider using PASV.
150 Opening BINARY mode data connection for xattr.h (4337 bytes).
226 File send OK.
4337 bytes received in 0.00038 seconds (9665.5 kB/s)
ftp> quit
221 Goodbye.
$ 

```

### 19.3.5 FTP 自动注册

每次使用 ftp 命令传输文件之前，都需要提供用户名和密码，才能建立与 FTP 服务器之间的连接。要启用 FTP 的自动注册功能，从而简化操作步骤，用户可以在自己的主目录中创建一个.netrc 文件，把远程主机名、用户名和密码等信息加到其中。.netrc 文件的语法格式如下。

```
machine rhostname login username [password password]
```

其中，关键字 machine 用于定义远程系统名，login 用于指定远程主机中的用户名，password 用于提供明文形式的密码（为安全起见，这个字段可以省略）。一旦设置了这个文件，即可实现



### FTP 服务器的自动注册。

例如，为使一个系统中的用户 gqxing 能够自动注册到 beijing、iscas 和 sinosoft 这 3 个 FTP 服务器，可以创建下列.netrc 文件（假定 gqxing 在这 3 个 FTP 服务器中注册的用户名也是 gqxing）。

```
$ cat $HOME/.netrc
# $HOME/.netrc file
#
machine beijing login gqxing password hereiam
machine iscas login gqxing password itsme
machine sinosoft login gqxing password helloagain
$
```

注意，由于.netrc 文件中可能包含密码，且密码是以明文形式给出的，FTP 要求必须使用下列命令设置.netrc 文件的访问权限，确保只有用户自己能够读写此文件。否则，FTP 将会拒绝执行自动注册过程。

```
$ chmod 400 $HOME/.netrc (或 chmod 600 $HOME/.netrc)
$
```

在完成上述两个步骤之后，只要输入 ftp 命令，立刻就会建立 FTP 连接，进入 ftp 命令状态，从而省略输入用户名和密码两个步骤。示例如下。

```
$ ftp iscas
Connected to iscas.
220 (vsFTPd 2.0.6)
331 Please specify the password.
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp>
```

# 第20章

## DNS 域名服务器

本章主要介绍 DNS 的基本概念及其各种配置文件，讨论 DNS 服务器和客户系统的配置过程，最后介绍如何启动和测试 DNS 服务器。其内容主要包括：

- DNS 基本概念；
- DNS 配置文件；
- DNS 服务器配置过程；
- 测试 DNS 服务器。





在 TCP/IP 网络系统中，网络访问、数据传输以及路由选择的主要依据是 IP 地址。但为了便于记忆，通常采用容易记忆的名字表示网络中的每一个主机。这就要求系统提供一种机制，负责实现名字与地址之间的转换。在一个小型网络中，利用 /etc/hosts 实现从名字到地址的转换是一种最为常见的方法。当网络的规模较大，尤其是连接到 Internet 上时，/etc/hosts 文件显然是无能为力的。

为了从根本上实现从名字到 IP 地址的转换，引入了域名服务（Domain Name Service，DNS）的概念。用户可以使用域名访问远程服务器，由 DNS 实现 IP 地址解析。例如，当访问 www.google.cn 网站时，DNS 将会把符号名字转换成 IP 地址（203.208.37.99 或 203.208.37.104），从而实现浏览器与网络服务器之间的数据通信。

取决于采用静态 IP 地址，还是采用 DHCP 协议动态分配的 IP 地址，DNS 服务器可以分为静态 DNS 和动态 DNS。为简化起见，本章仅介绍静态 DNS。在即将给出的示例环境中，ns.abc.net 域名服务器是集主域名服务器和缓冲域名服务器的功能于一体的。

域名服务器实际上是一组程序，负责维护相关域的域名和 IP 地址等资源信息，为客户系统的域名解析器（resolver）程序提供域名地址查询服务。域名服务器程序由 named 守护进程和 resolver 库函数等组成。Named 守护进程负责监听针对端口 53 的访问请求，提供域名地址查询服务。resolver 是一组驻留在系统库中的例程，提供应用编程接口，使应用程序能够访问域名服务。

## 20.1 DNS 基本概念

### 20.1.1 域与区

按照组织结构的性质与类型，或者根据国家与地区的地理位置，Internet 通常把整个网络划分为若干逻辑域（domain），如 com、net、org 和 edu 等普通域，以及 cn、ca 和 au 等国家与地区域。这些逻辑域通称为顶级域（top-level domain），顶级域可以进一步细分为若干子域，以此类推。域的层次结构就像一棵倒立的树，位于树根部的节点称作根域（root domain），位于树末梢的节点就是网络中的每一个主机，参见图 20-1。

#### 1. 域

树形结构中间的每一个节点称作子域，域的名字由从顶级域开始到当前域为止的所有中间节点名，中间加句点“.”的字符串组成，如 abc.net。从顶级域到一个叶子节点的域名是域内主机的规范域名，如 beijing.abc.net。

当组建一个新的域，并希望加入 Internet 公共网络时，需要向负责上层域的网管组织提出申请，注册新建的域及域名服务器，由自己的域名服务器负责新建域的域名地址解析，详情请参见 [ftp://rs.interic.net](http://rs.interic.net)、VeriSign、Register Free 和 Yahoo 等网站。为了叙述简便起见，本章将以一个假定的 abc.net 域为例，介绍 DNS 域名与 IP 地址解析的基本概念，说明域名服务器的配置过程。

#### 2. 区

区（zone）是由域名服务器管理和维护的一个域或域中的部分子域组成的。也就是说，一个

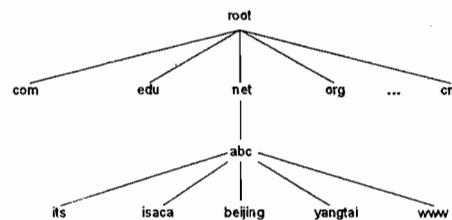


图 20-1 域的树形组织结构

域可以分为一个区或若干个区，每个区存在一个主域名服务器，用于维护区内所有主机的域名与 IP 地址映射关系数据库。这些数据通常存储在 named.conf 配置文件指定的区配置文件中，在 Ubuntu Linux 系统实现的 BIND 9 域名服务器中，配置文件的名字可由用户自己命名。

尽管 DNS 称作域名服务器，但其实现却是按照区的概念划分、管理和维护的。named.conf 配置文件中的关键字 master 和 slave 指定的是区，而不是域。

在每一个区中，除了主域名服务器之外，还可以配置多个辅助域名服务器，如从域名服务器和缓冲域名服务器等，但只有主域名服务器是必需的。

一个域名服务器既可以是一个单独的主、从或缓冲服务器，也可以是一个区（或若干区）的主域名服务器，同时又是其他区的从或缓冲域名服务器。主、从服务器均负责维护其相应区的域名地址映射数据，但根据其角色的定位，由主域名服务器优先回答客户系统的查询请求。所有的服务器均采用缓冲区保存获取的数据，直至数据超过时效。

## 20.1.2 DNS 域名服务器

在网络中，没有运行 named 守护进程的任何系统通称为 DNS 客户系统，这些系统不存储任何域名解析信息。当需要按域名访问其他服务器时，它们总是向 DNS 服务器提交域名解析的查询请求，包括从域名到地址的正向查询，和由地址到名字的反向查询。DNS 客户系统需要设置的唯一配置文件是 /etc/resolv.conf，其中定义了 DNS 服务器的 IP 地址。

### 1. 主域名服务器

主域名服务器（master name server，简称主服务器）是 DNS 区的域名管理系统，负责管理指定区的数据库文件，是其辖区内域名地址信息的权威数据来源。主域名服务器中的数据源自磁盘上的区配置文件，由网络管理员负责维护。主域名服务器既提供域名地址查询服务，又周期地向从域名服务器提供配置数据，传输配置数据副本（如果存在从域名服务器的话）。

DNS 域名服务器的主要功能是响应客户系统的 DNS 查询，提供域名解析服务，把标准的规范域名（Fully Qualified Domain Name，FQDN），如 www.abc.net 解析成 IP 地址，这个过程也称作正向解析。

在 Ubuntu Linux 系统中，DNS 服务器软件包称作 bind9。BIND 是 Berkeley Internet Name Domain 项目的简称，其核心程序是 named，负责维护 Linux 系统中运行的 DNS 软件。

### 2. 从域名服务器

从域名服务器（slave 或 secondary name server，简称从服务器）是主服务器的备份，负责维护定期从主域名服务器获取的配置数据副本。当主服务器出现故障或关机时，从服务器将会接替主服务器。从服务器维护一个区配置文件的副本，并按照指定的时间间隔（参见 SOA 资源记录中的 Refresh 字段），获取主服务器中的域名地址映射数据。如果存在从服务器，则所有的从服务器都必须在配置文件中的 NS 资源记录中定义。

### 3. 缓冲域名服务器

缓冲域名服务器（catching only name server，简称缓冲服务器）本身并不负责维护任何区配置数据，但会在其缓冲区中保存从其他服务器中收到的域名地址解析数据，直至数据超过有效期。数据的有效期是由域名服务器映射数据中的 TTL（Time-To-Live）字段确定的。如果缓冲区中存在客户系统查询的信息，缓冲服务器将会直接回答客户系统的查询。否则，缓冲服务器将会转而查询其他适当的域名服务器，最后再把查询结果返回客户系统。



缓冲服务器可以利用外部网络的转发服务器（参见 named.conf 文件中的 options 语句）查询指定的域名服务器。在此情况下，缓冲服务器首先查询转发服务器（而不是采用默认的解析处理过程），如果需要，转发服务器将会继续查询其他域名服务器，直至找出期望的数据。如果转发服务器没有响应，缓冲服务器将尝试查询区内的主域名服务器。

如果缓冲服务器没有配置可用的转发服务器，缓冲服务器可以直接查询区外的其他域名服务器，包括根域名服务器，以寻求其协助回答 DNS 查询。

如果配置的缓冲服务器只能使用指定的转发服务器，缓冲服务器不会查询此外的其他任何域名服务器。通过配置，DNS 域名服务器中运行的解析器（resolver）也可以查询其他 DNS 域名服务器，但其查询结果不能加到本地的域名缓冲区中。

大多数 DNS 客户系统通常并不直接向实际的 DNS 服务器提交地址解析的查询请求，而是向具有 DNS 功能的缓冲服务器发送查询请求。这些 DNS 服务器通过一个递归的查询进程，从根域开始依次向下查询每一级 DNS 服务器，如主域 DNS 服务器、子域 DNS 服务器等，以获取地址解析信息。缓冲域名服务器的作用是缓存累次查询获取的地址解析信息，供其他查询请求直接使用，以减小 DNS 查询引起的系统开销。

在设置好缓冲服务器之后，网络中的每一个系统均可以利用缓冲服务器获取域名地址解析信息。如果缓冲服务器的 IP 地址是采用 DHCP 协议动态分配的，则需要配置 DHCP 服务器，使其了解 DNS 服务器新设的 IP 地址，以便 DHCP 服务器能够向每一个系统公布 DNS 服务器。

#### 4. 转发服务器

转发服务器（forwarder）本身并不负责与根域名服务器或其他域名服务器进行交互查询，相反，当其负责维护或缓存的数据无法满足客户系统的查询请求时，转发服务器总是向预设的一组服务器转发客户系统的查询请求。转发的查询也称递归查询，其工作过程同客户系统向服务器查询一样。在此查询过程中，可能需要把查询请求转发到一个或多个服务器，直至穷尽预设的所有服务器，或者得到查询结果。

如果想要分区维护 DNS 信息，避免所有的服务器均与 Internet 直接交互，可以采取内置转发服务器，外加主域名服务器的组合方式。例如，可以把内部网中的 DNS 服务器配置成转发服务器，使其服务于内网的所有主机。当查询请求超出内网的服务范围时，则把查询请求转发给位于外网的主域名服务器，使之代而实现域名地址查询。

在实际的应用过程中，主、从域名服务器也可以声明为转发服务器。这样做的结果是，服务器将会使用缓存的或自己维护的域名地址映射数据回答针对本地域的查询，采用转发查询请求，寻求预设服务器协助，获取查询结果的方式，回答针对其他域的查询。

### 20.1.3 DNS 域名与地址解析

#### 1. 查询类型

DNS 地址解析的查询类型分为以下两种。

迭代查询：迭代查询意味着，当 DNS 客户系统提交一个地址解析查询时，DNS 服务器将根据自已缓存的数据或区内定义的数据返回其能够提供的最好回答。如果被查询的域名服务器中没有恰好匹配查询请求的答案，它将会提供下一级域名服务器的地址指针。然后，DNS 客户系统可根据返回的域名服务器地址，向新的域名服务器再次提交一个地址解析查询，直至找到一个能够

解析查询请求的域名服务器，或者出现错误或超时。

**递归查询：**递归查询意味着，当 DNS 客户系统提交一个地址解析查询时，DNS 服务器将承担全部的地址解析责任，最终返回一个完整的答案。此时，域名服务器将会作为一个 DNS 客户系统，负责执行迭代查询，依次查询其他域名服务器，直至返回一个最终的解析结果，然后把解析结果返回提交查询的客户系统。

## 2. 域名与地址解析的查询过程

假定 DNS 客户系统准备访问 www.abc.net，并向 DNS 服务器提交一个递归查询。收到地址解析查询的域名服务器将会采用迭代查询的方式，依次查询每一个相关的域名服务器，直至获得一个满意的答案，最后返回 DNS 客户系统。整个地址解析的查询过程简述如下（如图 20-2 所示）。

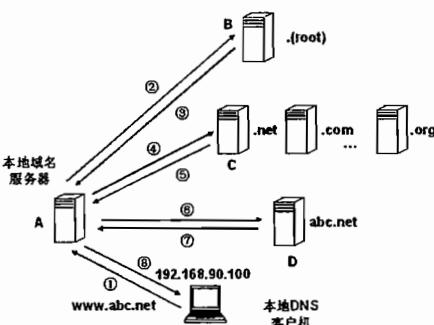


图 20-2 DNS 域名地址解析的过程

(1) 当因访问 www.abc.net 而生成一个 IP 地址查询请求时，DNS 客户系统将根据/etc/resolv.conf 配置文件的定义，向 DNS 域名服务器提交一个递归的地址解析查询。

(2) DNS 域名服务器 A 收到递归查询之后，发现自己的域名地址数据库中没有匹配 www.abc.net 的检索结果。于是，服务器 A 向负责根域地址解析的域名服务器发送一个迭代查询。

(3) 负责根域地址解析的域名服务器 B 收到迭代查询之后，发现自己的域名地址数据库中也没有匹配 www.abc.net 的检索结果。于是，服务器 B 向查询服务器 A 返回一个负责 net 域地址解析的 DNS 域名服务器的 IP 地址。

(4) 在收到 DNS 域名服务器的 IP 地址之后，承担递归查询的服务器 A 接着向负责 net 域地址解析的域名服务器发送一个迭代查询。

(5) 负责 net 域地址解析的域名服务器 C 收到迭代查询之后，发现自己的域名地址数据库中也没有匹配 www.abc.net 的检索结果。于是，服务器 C 向查询服务器 A 返回一个负责 abc.net 域地址解析的域名服务器的 IP 地址。

(6) 在收到另一个 DNS 域名服务器的 IP 地址之后，承担递归查询的服务器 A 接着向负责 abc.net 域地址解析的域名服务器发送一个迭代查询。

(7) 负责 abc.net 域地址解析的域名服务器 D 收到迭代查询之后，发现自己的域名地址数据库中确实存在匹配 www.abc.net 的检索结果。于是，服务器 D 向查询服务器 A 返回一个 www.abc.net 的 IP 地址。

(8) 在收到 www.abc.net 的 IP 地址之后，承担递归查询的服务器 A 则向 DNS 客户系统返回最终的查询结果。



### 3. DNS 域名与地址查询

DNS 的重要用途是提供域名地址解析功能，方便用户或应用程序查询已知域名的 IP 地址，或者根据 IP 地址，反向查询主机的规范域名。为此，Linux 系统提供了 host、nslookup 和 dig 等一系列用户工具。

#### (1) host 命令。

在 Linux 系统中，host 是一个简单实用的 DNS 域名地址解析命令，用于执行从域名到 IP 地址或从 IP 地址到域名的查询，即根据给定主机的域名查询相应的 IP 地址，或者根据给定的 IP 地址查询主机的规范域名。host 命令的语法格式简写如下。

```
host [options] hostname [server]
```

其中，hostname 可以是主机的规范域名，以便查询主机的 IP 地址；也可以是 IP 地址，执行反向的地址解析，查询主机的域名。server 是一个选用的参数，可用于指定一个域名服务器的名字或 IP 地址，表示使用指定的域名服务器，而不是/etc/resolv.conf 文件设定的域名服务器，执行域名地址解析。

例如，要正向查询指定系统的 IP 地址，或者执行反向地址解析，可以分别使用下列 host 命令。

```
$ host xys.dropin.org  
xys.dropin.org has address 68.178.232.99  
$ host 128.8.10.90  
90.10.8.128.in-addr.arpa domain name pointer d.root-servers.net.  
$
```

#### (2) nslookup 命令。

类似于 host 命令，nslookup 命令也可用于查询域名服务器，获取域名地址解析信息。nslookup 命令具有两种运行方式：交互方式与非交互方式。交互方式允许用户访问域名服务器，查询各种主机和域的相关信息，或列出域内的所有主机；非交互方式用于打印主机或域的名字及其他信息，其语法格式如下。

```
nslookup [options] [hostname | -] [server]
```

下面的例子说明了怎样使用 nslookup 命令，正向查询指定系统的 IP 地址，或者执行反向地址解析。

```
$ nslookup www.chinaelections.org  
Server: 172.16.3.6  
Address: 172.16.3.6#53  
  
Non-authoritative answer:  
www.chinaelections.org canonical name = chinaelections.org.  
Name: chinaelections.org  
Address: 211.100.30.135  
  
$ nslookup 199.7.83.42  
Server: 172.16.3.6  
Address: 172.16.3.6#53  
  
Non-authoritative answer:  
42.83.7.199.in-addr.arpa name = l.root-servers.net.  
  
Authoritative answers can be found from:  
83.7.199.in-addr.arpa nameserver = a.iana-servers.net.  
83.7.199.in-addr.arpa nameserver = b.iana-servers.net.  
a.iana-servers.net internet address = 192.0.34.43  
b.iana-servers.net internet address = 193.0.0.236  
$
```

## 20.2 DNS 配置文件

为了把 Ubuntu Linux 系统配置成一个域名服务器，提供域名地址解析服务，需要单独安装

bind9 及其相关的软件包，以 bind 用户的身份运行 named 守护进程。安装 bind9 等软件包时，可以使用 apt-get、aptitude 或 synaptic 等软件工具，示例如下。

```
$ sudo apt-get install bind9
```

在安装 bind9 软件包之后，其主要配置文件均位于/etc/bind 目录下，如表 20-1 所示。

表 20-1 配置文件的位置及其简单说明

配置文件	简单说明
/etc/resolv.conf	DNS 客户系统使用的配置文件，用于确定使用哪一个域名服务器提供域名地址解析服务
/etc/bind/named.conf	域名服务器的主配置文件，用于定义当前域名服务器负责维护的域名地址解析信息
/etc/bind/named.conf.options	用于单独定义本地域名服务器主配置文件的全局选项部分（options 语句）
/etc/bind/named.conf.local	其中包含当前域名服务器负责维护的所有区的定义
/etc/bind/db.local	用于定义环回接口主机名 localhost (127.0.0.1) 所在的域及其 A 记录。换言之，其中包含的是环回接口的主机名 localhost 与 IP 地址 127.0.0.1
/etc/bind/db.root	其中包含根域名服务器的主机名及其 IP 地址
/etc/bind/db.127	其中包含环回接口 IP 地址 127.0.0.1 反向解析的 PTR 记录
/etc/bind/db.0	其中包含网络地址 “0.*” 的反向解析信息
/etc/bind/db.255	其中包含广播地址 “255.*” 的反向解析信息
/etc/bind/zone-files	用于定义域名服务器负责管理与维护的所有正向区配置文件与反向区配置文件，是当前域名服务器能够提供的权威域名解析信息源
/etc/bind/rndc.key	其中包含 named 守护进程使用的认证信息

## 20.2.1 resolv.conf 文件

/etc/resolv.conf 主要是供 DNS 客户系统使用的配置文件，其主要用途是确定究竟由哪一个域名服务器提供域名地址解析功能。如果这个文件不存在或者为空，则假定本地系统就是运行 named 守护进程的域名服务器。

所有的 DNS 客户系统（包括缓冲域名服务器）均使用/etc/resolv.conf 文件确定 DNS 服务器的 IP 地址及其所在域或区的相关信息。故可以认为 resolv.conf 文件的主要作用就是指定域名服务器，说明由哪一个主机执行域名地址解析。在/etc/resolv.conf 文件中，每个记录通常包含两个字段，第一个字段是一个关键字，第二个字段包含相应的定义，多个定义值中间以逗号“,”作为分隔符（参见表 20-2）。

表 20-2 /etc/resolv.conf 文件中可用的关键字及其简单说明

关键字	简单说明
nameserver	指定 DNS 域名服务器的 IP 地址。每个 nameserver 关键字对应一个域名服务器。如果存在多个域名服务器，则需要增加多个包含关键字 nameserver 的域名服务器定义，每个定义单独占一行，但最多不能超过 MAXNS 变量（当前为 3，参见/usr/include/resolv.h 文件）规定的域名服务器。在此情况下，如果某个域名服务器超时，DNS 客户系统将会依次查询下一个域名服务器。如果不存在 nameserver 定义，则直接使用本地主机作为域名服务器
domain	定义默认情况下使用的本地域名（如果服务器的规范域名为 www.abc.net，则本地域名为 abc.net，下同）
search	定义执行地址查询时使用的域名检索表，默认情况下为本地域名。当定义多个域名时，域名之间必须加空格或制表符分隔符。在执行域名地址查询时，如果仅给出主机名而未指定其所在的域，系统将会把主机名依次加到关键字 search 定义的每一个域中，然后逐一提交 DNS 域名服务器解析，以获取给定主机的 IP 地址。这是一个简化用户输入的功能特性，用户只需提供主机名即可由系统自动组合成一个完整的规范域名。当前，域名检索表最多只能列举 6 个，总长不能超 256 个字符

假定 DNS 客户系统的主域为 abc.net，同时其也是 abc.org 和 abc.com 域的成员，如果选用两个 DNS 服务器（202.106.196.115 和 128.8.10.90）提供域名地址解析，为了简化域名查询，可以写



出下列 /etc/resolv.conf 配置文件。

```
search abc.net abc.org abc.com
nameserver 202.106.196.115
nameserver 128.8.10.90
```

注意，关键字 search 后面列出的第一个域名必须是当前系统所在的域（这里假定当前系统是 abc.net 域的成员）。

在 DNS 服务器中，必须使用 localhost、127.0.0.1 或网络接口的 IP 地址设置第一个 nameserver 关键字，把自己定义为域名服务器，其配置文件示例如下。

```
search abc.net abc.org abc.com
nameserver 127.0.0.1
nameserver 202.106.196.115
nameserver 128.8.10.90
```

## 20.2.2 named.conf 配置文件

named.conf 文件是 DNS 服务器中最重要的一个配置文件，其中包含了 DNS 服务器的总体配置信息，定义了 named 守护进程使用的所有配置文件，使 named 守护进程能够知道其维护的每一个域或区配置文件的名字及其目录位置。named.conf 文件支持 acl、controls、include、key、logging、options、server、trusted-keys、view 和 zone 等配置语句，但最常用的主要有 options、zone、view 及 acl 等配置语句，其语法格式简写如下。

```
options {
    [ directory path_name; ]                                     // 用于指定 named 的工作目录
    [ allow-notify { address_match_list }; ]                      // 主要用于从服务器
    [ allow-query { address_match_list }; ]                        // 主要用于视图
    [ allow-transfer { address_match_list }; ]                     // 主要用于主服务器
    [ transfer-source ip_addr; ]                                  // 主要用于从服务器
    [ recursion yes_or_no; ]                                      // 是否执行递归查询
    [ allow-recursion { address_match_list }; ]                   // 递归查询的细化
    [ notify yes_or_no; ]                                         // 主要用于主服务器
    [ also-notify { address_match_list }; ]                        // 主要用于主服务器
    [ notify-source (ip4_addr | *) [port port]; ]                // 主要用于从服务器
    [ allow-update { address_match_list }; ]                       // 主要用于主服务器
    [ forwarders { ip_addr_list }; ]                               // 主要用于主服务器
    [ forward { only | first }; ]                                 // 主要用于从服务器
    [ query-source [ address (ip_addr | *) ] [port (port | *)]; ]
    [ listen-on [ port port ] { address_match_list }; ]
    ....
};

acl acl-name { address_match_list };
include filename;
....
```

```
zone "domain_name" {
    type ( master | slave | hint | stub | forward );
    file path_name;
    [ allow-notify { address_match_list }; ]                      // 指定域名服务器的类型
    [ allow-query { address_match_list }; ]                        // 指定区配置文件
    [ allow-transfer { address_match_list }; ]                     // 主要用于从服务器
    [ allow-update { address_match_list }; ]                       // 主要用于主服务器
    [ masters [port port] { ip_addr_list; } ; ]                  // 主要用于从服务器
    [ transfer-source ip_addr; ]                                  // 主要用于从服务器
    [ forwarders { ip_addr_list }; ]                             // 主要用于从服务器
    [ forward { only | first }; ]                                // 主要用于主服务器
    [ notify yes_or_no; ]                                         // 主要用于主服务器
    [ notify-source (ip4_addr | *) [port port]; ]                // 主要用于从服务器
}
```

```

};

view "view_name" {
    match-clients { address_match_list };
    [ view_options; ]
    [ zone "domain_name" { ..... }; ]
    [ notify yes_or_no | explicit; ]                                // 主要用于主服务器
    [ allow-notify { address_match_list }; ]                         // 主要用于从服务器
    [ allow-query { address_match_list }; ]                          // 主要用于主服务器
    [ allow-transfer { address_match_list }; ]                      // 是否执行递归查询
    [ recursion yes_or_no; ]                                         // 递归查询的细化
    [ allow-recursion { address_match_list }; ]                     .....
};

1. options 语句

```

options 语句用于设置 named.conf 配置文件的全局选项。在 named.conf 配置文件中，options 语句只能出现一次。如果省略了 options 语句，options 语句中的每个选项均采取默认值。在 Ubuntu Linux 系统中，为了便于维护，options 语句及其定义已经从 named.conf 配置文件全部转移到一个单独的 named.conf.options 文件中。

在 options 语句中，通常只需设置 named 守护进程的工作目录，如果必要，也可以在此设置诸如 allow-query、allow-transfer、listen-on 或 recursion 等，否则均采用默认值，示例如下。

```

$ cat /etc/bind/named.conf.options
options {
    directory "/var/cache/bind";      // 指定域名服务器（即 named 守护进程）的工作目录
    .....
    listen-on { 192.168.90.100; 153.78.26.145; 127.0.0.1; };
    allow-transfer { none; };
};

$ 

```

## 2. zone 语句

在域名服务器中，对于管理与维护的每一个区（域），都需要使用 zone 语句做出详细的定义，包括域的名字（通常，zone 语句后边指定的名字即域的名字——参见随后即将介绍的 \$ORIGIN 定义）、域名服务器的类型以及配置文件的名字等。在 zone 语句中，通常至少应包括下列定义。

```

zone "example.com" {
    // 定义域名服务器的类型或角色，下列语句表示当前的服务器为主域名服务器
    type master;
    // 把域名服务器使用的区配置文件命名为 myzone.db
    file "myzone.db";
};


```

在域名服务器中，通常只需定义自己负责维护的域（如 abc.net），设置其正反向区配置文件。其他域只需利用 named.conf 文件中提供的定义及其区配置文件即可，如 localhost、“127.\*”、“0.\*”、“255.\*”和“.”等。

## 3. view 语句

视图是 BIND 9 的一个新特性。许多企业都希望内部和外部用户能够分别看到不同的网络视图，使得内部用户能够查询和访问所有的主机，外部用户仅限于查询和访问对外提供服务的部分主机。为了解决此类问题，BIND 引入了视图的概念。利用 view 语句，可以针对特定子网的主机定义不同的区配置文件，从而提供不同的域名解析信息。对于来自不同网络的客户系统查询，域名服务器能够采用不同的解析数据回答 DNS 查询。

每个 view 语句用于定义一个 DNS 域的视图，其中的 match-clients 选项用于控制客户系统是



否能够访问当前的视图。只有匹配 `match-clients` 选项中列举的 IP 地址，DNS 客户系统才能查询当前视图中的域名地址解析数据。

在不同的 `view` 语句中，`zone` 语句定义的区可以采用相同的名字，引用不同的配置文件，因而可以引用不同的域名地址数据。这意味着 DNS 服务器能够使用两组不同的区配置文件分别处理来自外部网络和内部网络的 DNS 查询。如果把同名但引用不同配置数据的区置于多个视图中，则不同的客户系统可以访问不同的数据。例如，在一个区分内网和外网的 DNS 设置中，可以使用不同的区配置文件，提供不同的域名地址解析服务。

利用视图的概念、`view` 语句和 `match-clients` 选项，可以从逻辑上把一个域中的主机划分为内外网。下面是一个利用 `view` 语句配置 DNS，采用相同的区名和不同的配置数据，使内外网的主机具有不同的查询与访问权限的例子。

```
view "internal" {
    // 把下列网络看作内网，其中的主机可以使用这个视图
    match-clients { 192.168.90.0/24; 192.168.100.0/24; };
    // 对位于内网中的主机，提供域名地址解析的递归查询服务
    recursion yes;
    // 内网中的主机能够看到 example.com 域的完整视图，包括内网中的主机
    zone "example.com" {
        type master;
        file "example-internal.zone";           // 其中包含所有主机的定义
    };
}
view "external" {
    // 任何 DNS 客户系统均可访问这个视图
    match-clients { any; };
    // 对于外部网络的主机，不提供域名地址解析的递归查询服务
    recursion no;
    // 外网中的主机只能看到 example.com 域的有限视图即只能查询外网能够访问的主机
    zone "example.com" {
        type master;
        file "example-external.zone";           // 其中仅包含对外提供服务的主机
    };
}
```

`named` 守护进程按照 `view` 语句在 `named.conf` 文件中出现的顺序依次处理每个视图，因此，限制最严格的视图应当置于配置文件的最前面。在上述例子中，如果互换内外视图的前后位置，则任何主机都无法访问内部视图。因为一经发现匹配的视图（`any` 表示匹配任何主机），`named` 守护进程就不会继续处理其他视图。

`options` 语句中的许多选项也可用于 `view` 语句，而这些选项仅适用于针对相应视图的查询。如果没有在视图中设置选项的值，则使用 `options` 语句中相应选项的值作为默认值。此外，也可以在 `view` 语句中设置 `zone` 语句中所用选项的默认值，而且，`view` 语句中同名选项的值将会取代 `options` 语句中相应选项的值。

如果 `named.conf` 配置文件中没有 `view` 语句，DNS 系统将会自动创建一个匹配任何客户系统的默认视图，配置文件中的任何 `zone` 语句均属于这一默认视图。如果使用视图明确地定义了 `view` 语句，则所有的 `zone` 语句必须出现在某个 `view` 语句之内。

#### 4. acl 语句

利用 `acl` 语句，可以定义命名的访问控制表（Access Control List，ACL），以便在其他语句或选项中直接引用。ACL 包含的 IP 地址，用于控制 DNS 客户系统访问 DNS 服务器的权限。在 `allow-notify`、`allow-query`、`allow-recursion` 或 `allow-transfer` 等配置语句中，可以使用 ACL 限制 DNS

客户系统的访问权限。下列关键字是系统内置的 ACL，可以直接引用。

any	能够匹配所有主机；
none	不匹配任何主机；
localhost	能够匹配本地系统所有网络接口的 IP 地址；
localnets	能够匹配本地系统网络接口直连网络中的任何主机。

利用 ACL，可以针对域名服务器的不同功能特性，分别调整不同 DNS 客户系统的访问权限，无需一一列出每个 DNS 客户系统的 IP 地址。限制外部客户系统随意访问域名服务器，可以防止各种恶作剧或 DoS 攻击。下面的例子说明了怎样使用 acl 语句。

```
// 首先设置一个针对内网的 ACL——mynet
acl mynet { 192.168.90.0/24; 192.168.100.0/24; };
// 然后使用 mynet 作为 ACL，允许内网主机访问服务器，迭代查询域名地址解析信息
options {
    ...
    allow-query { mynet; };
    allow-recursion { mynet; };
};

zone "example.com" {
    type master;
    file "/etc/bind/example-com.zone";
    allow-query { any; };
};
```

### 5. include 语句

include 语句用于组合指定的配置文件，在出现 include 语句的位置插入指定的配置文件内容。include 语句的主要作用是提高配置文件的模块化，以便于管理与维护。例如，下列 include 语句意味着把私钥文件加入域名服务器主配置文件。

```
include "/etc/rndc.key";
```

表 20-3 给出了 options、zone、view 及 acl 这 4 个配置语句中的常用选项及其简单说明。

表 20-3 options、zone、view 及 acl 语句中的常用选项及其简单说明

关 键 字	简 单 说 明
options 语句：	
directory	域名服务器的工作目录，如 /var/cache/bind。在 named.conf 等配置文件中，凡以相对路径名出现的任何文件均相对于这里指定的工作目录。工作目录也是域名服务器大多数输出文件的默认目录位置。如果未指定工作目录，默认的工作目录为启动 named 守护进程时的当前目录
dump-file	指定 “rndc dumpdb” 命令转储域名服务器数据库中的数据使用的文件名。如果未指定，默认的文件是 named_dump.db
statistics-file	指定 “rndc stats” 命令存储域名服务器统计信息使用的文件名。如果未指定，默认的文件是位于服务器当前目录下的 named.stats
notify	DNS 的通知机制允许主服务器在区配置数据发生变化时通知其从服务器。为了响应主服务器的通知信息，从服务器会检查自己的区配置文件；如果配置文件并非当前最新的版本，则启动配置文件的传输过程。如果把 notify 设置为 yes（默认值），当自己维护的区配置数据发生变化时，主服务器将会向区配置文件中 NS 记录列举的从服务器发送 DNS NOTIFY 消息，否则不发送。options 语句中 notify 选项设置的是全局选项，适用于所有的区。也可把 notify 选项置于 zone 语句中，这意味着仅当特定区的配置数据发生变化时，才通知相应的从服务器。zone 语句中设置的 notify 可以取代 options 语句中的 notify 设置
forwarders	指定是否转发 DNS 查询请求，以及代理服务器的 IP 地址，由指定的服务器代为实现 DNS 查询。这个选项的默认值为空，表示不转发查询请求，由当前的服务器处理 DNS 查询请求。也可以在 zone 语句中指定 forwarders，以取代 options 语句中的 forwarders 设置
forward	仅当 forwarders 选项的设置非空时，forward 选项才有意义。默认值 first 表示域名服务器首先查询转发服务器。如果转发服务器无法回答查询请求，域名服务器再设法由自己寻求答案。如果设定为 only，域名服务器仅查询转发服务器
allow-notify	当区配置数据发生变化时，除了主域名服务器，这个选项指定还允许哪些主机通知从服务器。如果未指定，从服务器仅接受来自主域名服务器的通知。从服务器的 zone 语句中也可以指定 allow-notify，以取代 options 语句中的 allow-notify 设置



续表

关 键 字	简 单 说 明
allow-query	指定可为哪一些客户系统提供常规的 DNS 查询服务。也可以在 zone 语句中指定 allow-query 选项，以取代 options 语句中的 allow-query 设置。如果未指定，默认的处理方式是为所有的客户系统提供 DNS 查询服务
allow-transfer	指定哪一些从服务器能够以批量传输的方式接收主服务器中的区配置数据。也可以在 zone 语句中指定 allow-transfer，以取代 options 语句中的 allow-transfer 设置。如果未指定，默认的处理方式是允许所有的从服务器以批量传输方式获取区配置数据副本
transfer-source	在从服务器的配置中，指定从哪一个源 IP 地址（或端口）中执行区域配置数据传输，获取配置数据副本。这个选项应与主服务器中的 allow-transfer 选项相互对应
query-source	如果当前服务器不知道 DNS 查询请求的答案，需要查询其他域名服务器，可以使用 query-source 选项指定查询其他域名服务器时使用的 IP 地址或端口号
listen-on	指定服务器接受 DNS 查询时监听的网络接口或端口号。如果未指定端口号，则监听默认的 53 号端口。如果省略了 listen-on 选项，服务器将会监听所有网络接口的 53 号端口
port	指定域名服务器接收和发送 DNS 协议分组数据时使用的 UDP/TCP 端口，默认值为 53。设置这个选项的主要目的是测试域名服务器。这是因为，如果使用除了 53 之外的其他端口号，将无法与网上的其他域名服务器通信。如果需要设定，应把 port 选项放到配置块的前面，位于涉及 IP 地址或端口号的任何选项之前，确保设置的端口号能够发生作用
recursion	指定是否由当前服务器替代客户系统执行递归查询。如果设置为 yes，域名服务器将尝试执行递归查询，直至获取最终的查询结果。如果设置为 no，当域名服务器不知道查询答案时，则只会返回一个线索信息。recursion 的默认值为 yes。注意，把 recursion 设置为 no 只是禁止服务器执行新的查询，获取新的查询结果，但并不妨碍 DNS 客户系统从域名服务器的缓冲数据中获取答案
allow-recursion	使用这个选项，可以更细化地控制域名服务器的递归查询。仅当匹配指定的地址列表时，当前域名服务器才会替代客户系统执行递归查询。如果未指定，意味着服务器能够替代所有客户系统进行递归查询。注意，禁止递归查询并不能防止客户系统获取已经位于域名服务器缓冲区中的数据
zone 语句:	
type	指定区的类型，即指定域名服务器的类型。类型可以是 master、slave、stub、forward 或 hint 等关键字之一。其中，master 表示当前服务器中包含区配置数据的正本，能够提供权威的 DNS 查询；slave 表示当前服务器中包含区配置数据的副本；hint 关键字用于指定一组初始的根域名服务器，当启动域名服务器时，域名服务器能够利用 “hint zone” 语句获取可用的根域名服务器；type 选项中定义的 master 或 slave，确定了域名服务器的角色，表示授权由谁回答当前区的域名地址解析查询
file	指定区配置文件的名字
master	用于指定主域名服务器的 IP 地址，以便从服务器能够利用这个 IP 地址联系主服务器，实现数据的定期更新。通常，域名地址数据的传输是利用 DNS 服务器的 53 号端口实现的
allow-transfer	同上
allow-update	指定允许哪些主机向主域名服务器提交按区进行的动态 DNS 数据更新。默认值是拒绝接收来自所有主机的 DNS 更新数据
match-clients	指定能够访问相应视图的 DNS 客户系统
notify	同上
view 语句:	
match-clients	表示相应的视图仅适用于匹配指定 ACL 的 DNS 客户系统

### 20.2.3 区配置文件

在 DNS 域名服务器中，域名与 IP 地址的映射关系等数据均位于区配置文件中。除了\$TTL 和\$ORIGIN 之外，每个区配置文件主要由 SOA、NS、A、PTR、CNAME 和 MX 等资源记录组成。

#### 1. \$TTL

\$TTL（Time-To-Live）配置指令用于设定一个时间长度（秒），表示配置数据的有效期，即缓存的域名解析数据在清除之前至少应当保持多长时间。一旦超过规定的有效期时间，缓冲 DNS 服务器需要从相应的域名服务器中获取更新的数据。缓冲 DNS 服务器通常会缓存来自各級域的 DNS 服务器

的响应信息。各级域的 DNS 服务器不仅提供域名地址解析的结果，还会附带提供数据的有效期。

设置 TTL 的目的是降低查询外部 DNS 服务器的频率。如果把 TTL 设置为 3 天 (\$TTL 3D)，则缓冲 DNS 服务器可以使用缓存的、未超过 3 天时效的数据直接响应客户系统的域名解析查询，而不必访问外部的 DNS 服务器。表示时间值的数字后可以加若干后缀，其中 D 表示天，W 表示周，H 表示小时。如果没有后缀，默认的时间单位是秒。

## 2. \$ORIGIN

\$ORIGIN 配置指令用于设置域名。如果区配置文件中未加 “\$ORIGIN *domain-name*” 形式的定义，named 将会把 named.conf 等配置文件中 “zone *domain-name*” 语句定义的名字作为默认的域名。在任何资源记录中，如果定义的主机名不是规范域名，或者域名后面没有以句点 “.” 结束，named 将会把域名附加到主机名的后面，构成完整的规范域名。\$ORIGIN 配置指令的语法格式如下。

```
$ORIGIN domain-name {comment}
```

其中，*domain-name* 表示新设定的域名（注意，域名的后面必须以句点 “.” 结束）。除了设定域名之外，在区配置文件的上下文环境中，\$ORIGIN 的值主要有两个用途：用于替换 “@” 符号（详见下一节的说明），表示当前的域；用于补充不完整的域名（主机名字后面没有以句点 “.” 结尾），以构成规范的域名。例如，如果区配置文件中存在下列\$ORIGIN 定义：

```
$ORIGIN example.com.
```

则资源记录

```
IN      NS      ns
```

相当于

```
IN      NS      ns.example.com.
```

如果 named.conf 配置文件包含下列定义，且区配置文件 example.com.zone 未定义\$ORIGIN，则\$ORIGIN 的初始值为 “example.com.”。注意，在下列 zone 语句中，example.com 后面的句点 “.” 可有可无。

```
zone "example.com." {
    type master;
    file "/etc/bind/example-com.zone";
};
```

## 3. DNS 资源记录

zone 配置文件是由若干标准的 DNS 资源记录组成的。可用的标准资源记录包括 SOA、NS、MX、A、PTR 和 CNAME 等，参见 RFC 1035。这些资源记录确定了域名服务器的性质，给出了域名服务器维护的数据、域或区的定义等。资源记录的语法格式如下。

```
Name [ Class ] Type Data
```

其中，Name 字段是域或主机的名字，Class 字段表示地址的类型，目前可用的地址类型只有 IN。Type 字段表示资源记录的类型，用于区分不同的资源记录。Data 字段是资源记录的相关数据，对于不同的资源记录，这个字段的变化比较大，详见每个记录的具体说明。

### 20.2.4 DNS 资源记录

#### 1. SOA 记录

除了 TTL 定义之外，区配置文件中的第一个资源记录是 SOA (Start of Authority)。SOA 记录标志着一个区的开始，其中的配置包括区或域的控制范围、管理与维护的域名定义等信息。直至遇到下一个 SOA 记录，表示当前区的配置已经结束。SOA 资源记录的语法格式如下。

```
Zone-Name Class Type Name-Server Email-Address Serial-No Refresh Retry Expiry
Minimum-TTL
```



表 20-4 给出了资源记录每个字段的简单说明。

表 20-4

SOA 资源记录

字 段	简 单 说 明
Zone-Name	区（或域）的名字。这个字段通常包含一个“@”缩写符号，表示\$ORIGIN 定义的域名或 named.conf 等配置文件中引用的域名。
Class	指定地址的类型。目前唯一合法的地址类型仍是 IP 地址，故这个字段总是为 IN
Type	DNS 资源记录的类型。对于 SOA 资源记录而言，Type 字段就是 SOA
Name-Server	主服务器的名字或规范域名（后面必须加一个句点“.”）
Email-Address	域名服务器管理员的电子邮件地址。电子邮件地址中的“@”字符必须更换为句点“.”，而且最后也必须以句点“.”结束
Serial-No	配置数据的系列号。可以采用“YYYYMMDD”格式的日期加一个增量数字表示配置数据的系列号。这种表示方法便于每天进行多次配置。从服务器使用这个系列号确定自上一次同步更新之后，配置数据文件是否发生变动。如果修改了配置文件，而未增加系列号的值，从服务器将不会理睬
Refresh	指定从服务器检测主服务器的频率（秒），以便确定是否需要更新。尽管 RFC 1033 的建议值为 3600（1 小时），但在实际应用中，设置为 86400（1 天）可能更恰当。在一个小型网络中，通常不需要使用从服务器
Retry	当连接主服务器出现故障时，从服务器再次尝试建立连接的时间间隔（秒）。RFC 1033 的建议值为 600（10 分钟）
Expiry	从服务器保持数据有效的上限值（秒）。如果超过此时间间隔仍未进行更新，强制从服务器进行同步更新，而不管主服务器中的配置数据是否已经发生变动。RFC 1033 的建议值为 3600000（约为 42 天）
Minimum-TTL	定义资源记录中 TTL 字段的默认值（秒）。在放弃之前，从服务器维护的缓冲数据不得低于这个时间。尽管 RFC 1033 的建议值为 86400（1 天），但在实际应用中可以设为若干天

SOA 记录通常很长，为了清晰起见，位于圆括号中的每个字段均可以占用一行。此外，每一行的后面也可以增加一个以分号“;”为起始字符的注解。示例如下。

```
;name    class   type    data
@        IN      SOA     ns.abc.net.      gqxing@gmail.com. (
                           200812011 ; serial #
                           4H       ; refresh
                           1H       ; retry
                           1W       ; expiry
                           1D )    ; minimum
```

在上述例子中，“@”符号表示当前的域（当前的域名取决于\$ORIGIN 的定义或 named.conf 配置文件中“zone domain-name”语句定义的域名，这里假定为 abc.net）。ns.abc.net 是 abc.net 域中的主域名服务器，域名服务器管理员的电子邮件地址为 gqxing@gmail.com，域名地址解析数据的系列号为 200812011，refresh、retry、expiry 和最小 TTL 值分别为 4 小时、1 小时、1 周和 1 天。

## 2. NS 记录

NS (Name Server) 资源记录用于标识一个区内的所有域名服务器。如果一个区包含多个域名服务器，则区配置文件中需要增加多个 NS 记录。NS 记录的语法格式如下。

```
[Zone-Name] Class Type Name-Server
```

表 20-5 给出了 NS 资源记录每个字段的简单说明。

表 20-5

NS 资源记录每个字段的简单说明

字 段	简 单 说 明
Zone-Name	区（或域）的名字。这个字段是可选的，如果省略，表示域名服务器位于当前的区
Class	指定地址的类型。目前唯一合法的地址类型仍是 IP 地址，故这个字段总是为 IN
Type	DNS 资源记录的类型。对于 NS 资源记录而言，Type 字段就是 NS
Name-Server	域名服务器的主机名或规范域名（规范域名后面必须以句点“.”结束）

例如，下面的 NS 记录表示，主机 ns 定义为当前区（或域）中的域名服务器，其完整的规范

域名应为“ns+当前域名”。

```
;name    class     type      data
      IN        NS       ns           ; IP Address of Name Server
```

### 3. A 记录

A (Address) 记录用于定义一个区内管理和维护的每个主机的 IP 地址，以便实现从域名到 IP 地址的映射。同一个主机（如网关）可能会存在多个 A 记录，分别定义每个网络接口的 IP 地址。A 记录的语法格式如下。

```
[Host-Name] Class Type Address
```

表 20-6 给出了 A 资源记录每个字段的简单说明。

表 20-6

A 资源记录每个字段的说明

字 段	简 单 说 明
Host-Name	主机的名字
Class	指定地址的类型。目前唯一合法的地址类型仍是 IP 地址，故这个字段总是为 IN
Type	DNS 资源记录的类型。对于 A 资源记录而言，Type 字段就是 A
Address	主机的 IP 地址

例如，下面的 A 记录表示，主机 iscas 是当前区（或域）中的一个主机，其完整的规范域名应为“iscas+当前域名”，IP 地址为 192.168.90.101。

```
;name    class     type      data
iscas    IN        A         192.168.90.101
```

### 4. PTR 记录

PTR (Pointer) 记录用于实现 IN-ADDR.ARPA 域中从 IP 地址到域名的映射。域名服务器中的反向区配置文件必须包含一个区内管理和维护的每个主机的 PTR 记录。PTR 记录的语法格式如下。

```
Address Class Type Host-name
```

表 20-7 给出了 PTR 资源记录每个字段的简单说明。

表 20-7

PTR 资源记录每个字段的说明

字 段	简 单 说 明
Address	主机 IP 地址的最后一组数字，或反向表示的部分 IP 地址
Class	指定地址的类型。目前唯一合法的地址类型仍是 IP 地址，故这个字段总是为 IN
Type	DNS 资源记录的类型。对于 PTR 资源记录而言，Type 字段就是 PTR
Host-name	主机名或规范域名（规范域名后面必须以句点“.”结束）

例如，下面的 PTR 记录表示，beijing.abc.net 是 IN-ADDR.ARPA 主域中一个主机，其反向 IP 地址的最后一组数字为 100，完整的规范域名为“100+zz.yy.xx.in-addr.arpa”。

```
;name    class     type      data
100     IN        PTR      beijing.abc.net.
```

### 5. CNAME 记录

CNAME (Canonical Name) 别名记录用于定义一个主机的别名。在域名服务器的正向区配置文件中，一个主机可以存在多个 CNAME 记录，但反向的区配置文件不能包含任何 CNAME 记录。CNAME 记录的语法格式如下。

```
Alias Class Type Host-Name
```

表 20-8 给出了 CNAME 资源记录的每个字段及其简单说明。



表 2-8

CNAME 资源记录每个字段的说明

字 段	简 单 说 明
Alias	主机的别名
Class	指定地址的类型。目前唯一合法的地址类型仍是 IP 地址，故这个字段总是为 IN
Type	DNS 资源记录的类型。对于 CNAME 资源记录而言，Type 字段就是 CNAME
Host-Name	主机的名字

例如，下面的 CNAME 记录表示，ns 实际上只是主机 beijing 的一个别名，其规范域名应为“ns+当前域名”，与主机 beijing 具有相同的 IP 地址——192.168.90.100。

```
;name class type data
ns IN CNAME beijing
```

注意，在实际的配置文件中，最好使用 A 记录取代 CNAME 记录，以提高域名查询的响应速度，也可避免 named-checkzone 的语法检查错误（参见 20.3 节介绍的区配置文件）。

## 6. MX 记录

MX (Mail Exchange) 记录用于定义和控制怎样向一个区或一个区内的主机传递电子邮件。当向一个区或区内的主机传递电子邮件存在多条路径时，priority 字段用于设置中转服务器的优先级，以便按顺序选用最佳的电子邮件服务器。数值越小，表示相应的主机越适合传递电子邮件。如果多个中转服务器的优先级相同，则选用遇到的第一个服务器。MX 记录的语法格式如下。

```
Host-Name Class Type Priority Mail-Server
```

表 20-9 给出了 MX 资源记录每个字段的简单说明。

表 20-9

MX 资源记录每个字段的说明

字 段	简 单 说 明
Host-Name	目的主机（即接收电子邮件的主机）的域名。在域名的第一个字段中，可以使用星号“*”通配符，表示指定域或当前区中的所有主机
Class	指定地址的类型。目前唯一合法的地址类型仍是 IP 地址，故这个字段总是为 IN
Type	DNS 资源记录的类型。对于 MX 资源记录而言，Type 字段就是 MX
Priority	这个字段是一个整数，用作选择最佳电子邮件服务器的优先级
Mail-Server	电子邮件服务器的主机名或规范域名（规范域名后面必须以句点“.”结束）

例如，下面的 MX 记录表示，mail 是 abc.net 域中的一个电子邮件服务器。其中，0 表示主机 mail 是最适合为 abc.net 域中的主机提供电子邮件的中转服务器。

```
;name class type data
abc.net. IN MX 0 mail ; IP Address of Mail Server
```

同 SOA 资源记录一样，每个 NS、A、PTR、CNAME 和 MX 资源记录均占用一行，其记录格式也非常类似。表 20-10 简单概括了前述的每个资源记录的用法。

表 20-10

资源记录用法一览表

记录类型	字 段 说 明			
	Name 字段	Class 字段	Type 字段	Data 字段
NS	通常为空，表示当前的区	IN	NS	域名服务器的 IP 地址、别名或规范域名
MX	电子邮件服务器所在域的域名。通常与 zone 文件所在域的域名是相同的	IN	MX	优先级以及 Mail 服务器的主机名或规范域名
A	主机的名字	IN	A	主机的 IP 地址
CNAME	主机的别名	IN	CNAME	与主机的 A 记录的 Name 字段相对应，即主机的名字
PTR	主机 IP 地址的最后一组数字，或反向表示的部分 IP 地址	IN	PTR	主机的规范域名

注意，除了主机名之外，SOA、NS、A、CNAME 和 MX 等资源记录的规范域名后面必须附加一个句点“.”。例如，如果 A 记录前面的主机名为 beijing，BIND 将会自动地在主机名后面追加区配置文件中定义的域名，把 beijing 看作 beijing.abc.net。

## 20.3 DNS 服务器配置过程

配置 DNS 服务器主要包括三个步骤：设置 resolv.conf 文件，设置 named.conf 等配置文件，设置区配置文件。

假定有一个网络——192.168.90.0/24，网内存在 4 个主机，每个主机的名字与 IP 地址如图 20-3 所示。作为一个独立存在的域，我们把这个域命名为 abc.net。如果需要连接到 Internet 中，则需要申请一个正式的域名，并把新设的域名服务器注册到上级域的域名服务器中。下面将以 abc.net 域为例，说明服务器的配置过程。

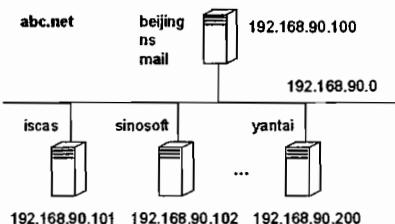


图 20-3 abc.net 域

### 20.3.1 设置 resolv.conf 配置文件

在 Ubuntu Linux 系统中，安装 bind9 软件包且在 named 守护进程开始运行之后，系统本身的基本配置已经具备一个简单的缓冲域名服务器的功能。通常，Linux 系统可以利用 /etc/hosts、/etc/resolv.conf 文件或 DNS 服务器等方法，获取或实现域名地址解析。如果想要利用本地主机实现域名地址解析，使本地系统成为域名服务器，首先需要编辑 /etc/resolv.conf 文件，设置 DNS 域名服务器的 IP 地址，把 nameserver 定义为本地主机，即 localhost、127.0.0.1 或本地网络接口的 IP 地址，使所有的 DNS 查询均通过本地系统实现。在我们的例子中，/etc/resolv.conf 文件可以配置如下。

```
search abc.net
nameserver 127.0.0.1
```

在实际应用过程中，即使不是一个真正的域名服务器，把本地系统配置成缓冲域名服务器也会获得较好的性能。DNS 客户系统调用的解析器（resolver）不会缓存获取的查询结果，这是由于其生命周期比较短。而域名服务器则会缓存查询的结果数据，因此，本地系统或同一网络中的其他 DNS 客户系统可以充分利用这一优势，提高查询的性能。

### 20.3.2 设置 named.conf 配置文件

设置 named.conf 配置文件的目的是定义 named 守护进程负责维护的域及其配置文件，使 named 守护进程能够知道其维护的每一个域、相应的区配置文件及其路径名。在设置 named.conf 配置文件时，由于其中采用了 include 语句，通常只需设置 named.conf.local 配置文件，不必修改



named.conf 配置文件（注意，仅在使用视图时才需要设置 named.conf.options 配置文件）。

除了系统提供的各种默认的区配置文件之外，在 named.conf.local 配置文件中，通常至少需要定义两个区配置文件：

- 正向的区配置文件——用于指定包含域名到 IP 地址映射关系的数据库文件；
- 反向的区配置文件——用于指定包含 IP 地址到域名映射关系的数据库文件。

在下面的例子中，首先在 named.conf.local 文件中定义一个正向的区配置文件，并把这个文件命名为 abc-net.zone。

```
zone "abc.net" {
    type master;
    file "/etc/bind/abc-net.zone";
    allow-query { any; };
};
```

在上述配置中，“type master”表示当前系统是一个主域名服务器。关键字 allow-query 用于限定能够查询 DNS 服务器的客户系统，其中的 any 表示任何客户系统均可查询 DNS 服务器负责管理的 abc.net 域。如果想要限定只有 192.168.90.0 网络中的客户系统能够查询 DNS 服务器，上述定义应修改如下。

```
allow-query { 192.168.90.0/24; };
```

接着，还需要定义处理反向 IP 地址查询的 zone 配置文件，以便位于 192.168.90.0/24 网络中的客户系统或应用程序能够根据 IP 地址查询系统的规范域名。

正向域名解析的处理过程是从右到左依次解析一个规范的域名，而反向地址解析的处理过程是从左到右逐次解析一个标准的 IP 地址。其差别可以从 /etc/named.conf 文件中的 zone 语句中反映出来：在一个属于 in-addr.arpa 域的定义中，其 IP 地址的前 3 个数字均以逆序的形式出现，这一点要特别注意。

在 named.conf.local 配置文件中，我们把 192.168.90.0/24 网络，属于 90.168.192.in-addr.arpa 域的反向区配置文件命名为 192-168-90.zone，示例如下。

```
zone "90.168.192.in-addr.arpa" {
    type master;
    file "/etc/bind/192-168-90.zone";
    allow-update { none; };
};
```

完成上述配置之后，named.conf.local 配置文件的设置就完成了，其最终结果如下。

```
$ cat /etc/bind/named.conf.local
...
zone "abc.net" {
    type master;
    file "/etc/bind/abc-net.zone";
    allow-query { any; };
};
zone "90.168.192.in-addr.arpa" {
    type master;
    file "/etc/bind/192-168-90.zone";
    allow-query { any; };
};
$
```

### 20.3.3 设置正向区配置文件

根据前面的介绍，现在可以给出 abc.net 域的正向区配置文件，如下所示。

```
$ cat /etc/bind/abc-net.zone
; Zone file for abc.net
$TTL 3D
@           IN      SOA      ns.abc.net.      gqxing@gmail.com. (
```

```

200812012      ; serial#
3600          ; refresh, seconds
3600          ; retry, seconds
3600          ; expire, seconds
3600 )        ; minimum, seconds
IN      NS      ns.abc.net.    ; Name Server
IN      MX      0 mail.abc.net. ; Mail Server
Localhost IN      A      127.0.0.1
beijing   IN      A      192.168.90.100
ns       IN      A      192.168.90.100 ; 取代“ns” IN CNAME beijing”
mail     IN      A      192.168.90.100 ; 取代“mail” IN CNAME beijing”
iscas    IN      A      192.168.90.101
sinosoft  IN      A      192.168.90.102
yantai   IN      A      192.168.90.200
$
```

在上述配置文件中，ns.abc.net 是 abc.net 域的域名服务器。其中定义的最小 TTL 值 (\$TTL) 是 1 小时 (3600 秒)，因此，缓冲域名服务器获取的 DNS 信息最多只能保存 1 小时，过期后将会从缓冲区中清除。abc.net 域的 MX 记录指向电子邮件服务器 mail.abc.net，而 mail 实际上是主机系统 beijing 的别名。DNS 服务器 ns 实际上也是主机系统 beijing 的别名。iscas、sinosoft 和 yantai 是 abc.net 域中的 3 个普通主机。

此后，每当修改区配置文件，增加或删除主机记录时，都需要增加序列号，以便通知从服务器等更新自己的数据库文件。在一个具有主、从服务器的 DNS 环境中，从服务器将会周期地轮询主服务器的区配置文件，并使用序列号确定主服务器中的配置数据是否已经更新。如果不增加记录号，即使已经修改了区配置文件中的内容，也不会引起从服务器的注意。

至此，上述配置已经完成了从规范域名到 IP 地址的正向解析，可以查询 abc.net 域，即 192.168.90.0 网络中所有主机的 IP 地址。

### 20.3.4 设置反向区配置文件

下面针对 192.168.90.0 网络，说明怎样配置反向区配置文件。在下述例子中，第一列的所有数字项均为相应网络中每个主机 IP 地址的最后一组数字。其中，100 表示 IP 地址 192.168.90.100，指向主机的规范域名 beijing.abc.net。示例如下。

```

# cat /etc/bind/192-168-90.zone
; Zone file for 192.168.90.0
$TTL 3D
@      IN      SOA      ns.abc.net.      gqxing.gmail.com. (
                           200812012           ; serial number
                           3600                ; refresh, seconds
                           3600                ; retry, seconds
                           3600                ; expire, seconds
                           3600 )              ; minimum, seconds
IN      NS      ns.abc.net.    ; Name Server
100    IN      PTR      beijing.abc.net.
100    IN      PTR      ns.abc.net.
100    IN      PTR      mail.abc.net.
101    IN      PTR      iscas.abc.net.
102    IN      PTR      sinosoft.abc.net.
200    IN      PTR      yantai.abc.net.
#
```

注意，除了 SOA 和 NS 资源记录之外，正向与反向区配置文件的主要差别在于：正向区配置文件主要由 A 资源记录组成，如果需要，还可以加入 CNAME 和 MX 等资源记录；反向区配置文件主要由 PTR 资源记录组成，不能加入 CNAME 和 MX 等资源记录。

注意，在完成上述配置之后，域名服务器的功能仅限于 192.168.90.0 网络中的客户系统查询



abc.net 域中的主机系统，以及利用 named.conf 文件中配置的根域“.”（db.root）查询 Internet 外部网络中的主机，外部网络中的客户系统尚无法查询 abc.net 域。为了使外部网络中的客户系统也能查询和访问 abc.net 域中的各种服务器，必须把 abc.net 域正式注册到上一级域中。

### 20.3.5 DNS 视图

假定一个企业的网络拓扑如图 20-4 所示。主机 beijing.abc.net 位于 153.78.26.0 和 192.168.90 内外两个网络之间，其外部网络地址为 153.78.26.145，内网 IP 地址是 192.168.90.100。beijing 既是 abc.net 域的主域名服务器，又是电子邮件服务器。www.abc.net 和 news.abc.net 分别为 Apache 服务器和新闻服务器，对内对外提供服务，其外部网络地址分别为 153.78.26.166 和 153.78.26.168。iscas.abc.net、sinosoft.abc.net 和 yantai.abc.net 等主机位于内部网络，其 IP 地址分别为 192.168.90.101、192.168.90.102 和 192.168.90.200。

101、192.168.90.102 和 192.168.90.200。

为了保护内部网络，使得内部网络中的主机具有完整的网络视图，能够查询和访问 abc.net 域中的所有主机系统，外部网络只能查询和访问 abc.net 域内对外公开提供服务的主机系统，如 www、news 和 mail 主机，获取不同的域名解析，我们采用拆分的 DNS，设置不同的视图，将 DNS 分成内外两个域名空间。

首先，把网络中的主机划分为两个区：把位于外网（153.78.26.0）中的主机定义为一个区，对应于 abc.net 域；把位于内网（192.168.90.0）中的主机定义为同名的区，也对应于 abc.net 域，但使用不同的配置文件与域名地址解析数据。主机 beijing 位于两个区中，作为域名服务器，负责管理两个不同的区。

在 named.conf 配置文件中，把区配置文件适当地分布到两个 view 语句中，使所有的 zone 语句位于两个 view 语句之一。第一个 view 语句定义内部视图，指定内部网络使用的区配置文件；第二个 view 语句定义外部视图，指定互联网外部网络使用的区配置文件。利用内外视图实现拆分的 DNS，可使不同的客户系统只能看到不同的名字空间。

把内部视图使用的区配置文件命名为 abc-net-internal.zone，置于 named.conf 配置文件中的内部视图，供内部网络查询 192.168.90.0 和 153.78.26.0 网络中的主机，如 iscas.abc.net（192.168.90.101）。新建一个用于 Internet 外部网的区配置文件 abc-net.zone，把这个区配置文件置于 named.conf 配置文件中的外部视图，供外部网络查询 153.78.26.0 网络中的服务器，如 www.abc.net（153.78.26.166）。

其次，必须告诉 DNS 服务器，哪些地址属于内部网络，哪些地址属于外部网络。因此，可在相应的视图中使用 match-clients 语句定义匹配的网络。此外还要注意，当使用 view 语句定义视图时，所有的 zone 语句必须位于某个视图之内，故在配置视图时，通常均把 localhost、127.\*、0.\*、255.\* 和“.” zone 语句置于定义内部视图的 view 语句中。正因为如此，需要修改 named.conf 文件，把 named.conf.local 中的 zone 语句合并到 named.conf 配置文件中。下面是经过改造后的 named.conf.options 与 named.conf 配置文件。

```
$ cat /etc/bind/named.conf.options
options {
    directory "/var/cache/bind";
```

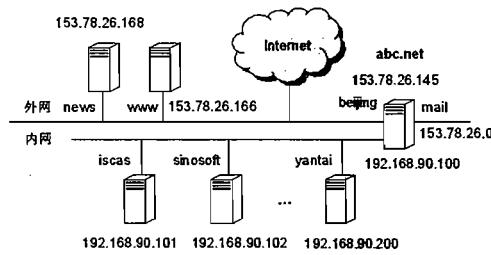


图 20-4 网络拓扑示意图

```

.....
listen-on { 192.168.90.100; 153.78.26.145; 127.0.0.1; };
allow-transfer { none; };
};

$ cat /etc/bind/named.conf
.....
include "/etc/bind/named.conf.options";           // 省略了注释部分
.....
//include "/etc/bind/named.conf.local"; // named.conf.local 文件中的内容已合并到此文件
acl internalnet { 192.168.90.0/24; };

view "internal" {                                     // 仅供内网客户系统查询 DNS 服务器使用的视图
    match-clients { localhost; internalnet; };
    recursion yes;
    zone "." {
        type hint;
        file "/etc/bind/db.root";
    };
};
.....
zone "localhost" {
    type master;
    file "/etc/bind/db.local";
};
zone "127.in-addr.arpa" {
    type master;
    file "/etc/bind/db.127";
};
zone "0.in-addr.arpa" {
    type master;
    file "/etc/bind/db.0";
};
zone "255.in-addr.arpa" {
    type master;
    file "/etc/bind/db.255";
};
zone "abc.net" {
    type master;
    file "/etc/bind/abc-net-internal.zone";
    allow-update { none; };
};
zone "90.168.192.in-addr.arpa" {
    type master;
    file "/etc/bind/192-168-90.zone";
    allow-update { none; };
};
zone "26.78.153.in-addr.arpa" {
    type master;
    file "/etc/bind/153-78-26.zone";
    allow-update { none; };
};
};

view "external" {                                     // 外部网络客户系统查询 DNS 服务器使用的视图
    match-clients { any; };
    recursion no;
};

zone "abc.net" {
    type master;
    file "/etc/bind/abc-net.zone";
    allow-update { none; };
};
zone "26.78.153.in-addr.arpa" {
    type master;
    file "/etc/bind/153-78-26.zone";
    allow-update { none; };
};
};

$ 

```

在上述的 match-clients 语句中，localhost 表示当前的 DNS 服务器，internalnet 表示 DNS 服务



器直接连接的内网中的所有主机。由此可见，localhost（当前的 DNS 服务器）以及内部网络（192.168.90.0/24）中的任何主机均可以从内部视图的区配置文件中获取 DNS 域名地址解析数据。

由于内部视图在前，外部视图在后，且内部视图中采用了最严格的 localhost 与 internalnet 访问控制，所以这种配置限制了外网主机在查询 abc.net 域中的主机时进入内部视图。同时，也保证了内网主机的查询不会进入外部视图。

此外，内部视图中的“recursion yes”语句表示域名服务器可以代替内网中的主机执行递归查询，外部视图中的“recursion no”语句表示域名服务器不会代替外网中的任何主机执行递归查询。

新建的 abc-net-internal.zone 和 abc-net.zone 文件内容如下。

```
$ cat abc-net-internal.zone
; Zone file for abc.net internal
$TTL 3D
@       IN      SOA    ns.abc.net.      gqxing.gmail.com. (
                           200812012           ; serial number
                           8H                  ; refresh, seconds
                           2H                  ; retry, seconds
                           4W                  ; expire, seconds
                           1D )                ; minimum, seconds
                           IN      NS      ns
                           IN      MX      0 mail
beijing   IN      A       192.168.90.100
ns        IN      A       192.168.90.100 ; 取代 “ns” IN CNAME beijing”
mail      IN      A       192.168.90.100 ; 取代 “mail” IN CNAME beijing”
iscas     IN      A       192.168.90.101
sinosoft   IN      A       192.168.90.102
yantai    IN      A       192.168.90.200
www       IN      A       153.78.26.166
news      IN      A       153.78.26.168
$ cat abc-net.zone
; Zone file for abc.net external
$TTL 3D
@       IN      SOA    ns.abc.net.      gqxing.gmail.com. (
                           200812012           ; serial#
                           8H                  ; refresh, seconds
                           2H                  ; retry, seconds
                           4W                  ; expire, seconds
                           1D )                ; minimum, seconds
                           IN      NS      ns.abc.net.
                           IN      MX      0 mail.abc.net.
beijing   IN      A       153.78.26.145
ns        IN      A       153.78.26.145
mail      IN      A       153.78.26.145
www       IN      A       153.78.26.166
news      IN      A       153.78.26.168
$
```

注意，当同一个主机具有多个名字时，应尽可能使用 A 记录而不是 CNAME 别名记录定义主机。否则，当请求解析 CNAME 记录定义的主机名时，还需要额外做一次实际主机的关联查询，因而使解析 IP 地址的时间加倍。

新建的方向区配置文件 192.168.90.zone 与 153.78.26.zone 内容如下。

```
$ cat 192-168-90.zone
; Zone file for 192.168.90.x
;
$TTL 3D
@       IN      SOA    ns.abc.net.      gqxing.gmail.com. (
                           200812012           ; serial number
                           8H                  ; refresh, seconds
                           2H                  ; retry, seconds
                           4W                  ; expire, seconds
                           1D )                ; minimum, seconds
                           IN      NS      ns.abc.net.

```

```

100      IN      PTR      beijing.abc.net.
100      IN      PTR      ns.abc.net.
100      IN      PTR      mail.abc.net.
101      IN      PTR      iscas.abc.net.
102      IN      PTR      sinosoft.abc.net.
200      IN      PTR      yantai.abc.net.

$ cat 153-78-26.zone
; Zone file for 153.78.26.x
;
$TTL 3D
@      IN      SOA     ns.abc.net.      gqxing.gmail.com. (
                           200812012 ; serial number
                           8H          ; refresh, seconds
                           2H          ; retry, seconds
                           4W          ; expire, seconds
                           1D )        ; minimum, seconds
                           ns.abc.net. ; Name Server
145      PTR      beijing.abc.net.
145      PTR      ns.abc.net.
145      PTR      mail.abc.net.
166      PTR      www.abc.net.
168      PTR      news.abc.net.

$
```

### 20.3.6 检测配置文件

在 Ubuntu Linux 系统中, named 守护进程是以用户 bind 的身份运行的。在完成 DNS 服务器上述配置文件的设置之后, 必须确保/etc/bind 目录中新建配置文件的访问权限是适当的。然后可以使用 named-checkconf 和 named-checkzone 命令, 先对 named.conf 和区配置文件分别进行语法检查, 如下所示。

```

$ cd /etc/bind
$ named-checkconf
$ named-checkzone abc.net abc-net.zone
zone abc.net /IN: loaded serial 200812012
OK
$ named-checkzone 90.168.192.in-addr.arpa 192-168-90.zone
zone 90.168.192.in-addr.arpa/IN: loaded serial 200812012
OK
$
```

当区配置文件比较多时, 上述的语法检查过程非常麻烦, 为此, 可以使用 named-checkconf 命令的 “-z” 选项, 对 named.conf 配置文件引用的所有区配置文件都进行语法测试。示例如下。

```

$ named-checkconf -z
zone localhost/IN: loaded serial 2
zone 127.in-addr.arpa/IN: loaded serial 1
zone 0.in-addr.arpa/IN: loaded serial 1
zone 255.in-addr.arpa/IN: loaded serial 1
zone abc.net/IN: loaded serial 200812012
zone 90.168.192.in-addr.arpa/IN: loaded serial 200812012
zone 26.78.153.in-addr.arpa/IN: loaded serial 200812012
zone abc.net/IN: loaded serial 200812012
zone 26.78.153.in-addr.arpa/IN: loaded serial 200812012
$
```

在启动 named 守护进程时, 如果配置文件有误, 系统将会立即给出错误信息。此外, 也可以使用 tail 命令检查/var/log/syslog, 确认 named 的启动与 named.conf 等配置文件的加载是否正确无误。下面的例子表明, named 守护进程已成功地开始运行。

```

$ tail /var/log/syslog
Nov 1 18:53:27 iscas named[9930]: zone 0.in-addr.arpa/IN/internal: loaded serial 1
Nov 1 18:53:27 iscas named[9930]: zone 127.in-addr.arpa/IN/internal: loaded serial 1
Nov 1 18:53:27 iscas named[9930]: zone 26.78.153.in-addr.arpa/IN/internal: loaded
serial 200812012
Nov 1 18:53:27 iscas named[9930]: zone 90.168.192.in-addr.arpa/IN/internal: loaded
serial 200812012
Nov 1 18:53:27 iscas named[9930]: zone 255.in-addr.arpa/IN/internal: loaded serial 1
Nov 1 18:53:27 iscas named[9930]: zone localhost/IN/internal: loaded serial 2
```



```
Nov 1 18:53:27 iscas named[9930]: zone abc.net/IN/internal:  
Nov 1 18:53:27 iscas named[9930]: zone 26.78.153.in-addr.  
serial 200812012  
Nov 1 18:53:27 iscas named[9930]: zone abc.net/IN/external:  
Nov 1 18:53:27 iscas named[9930]: running  
$
```

注意，每当修改 DNS 配置文件时，都需重新启动 DNS 服务器的命令。读取配置文件，以便确保修改后的配置立即生效。此外，还需要注意增加口值，否则从服务器不会理睬。要重启 DNS 服务器，可以使用下列命令。

```
$ sudo /etc/init.d/bind9 restart  
* Stopping domain name service... bind  
* Starting domain name service... bind  
$
```

## 20.4 测试 DNS 服务器

### 20.4.1 验证 DNS 服务器

在成功地启动域名服务器之后，可以使用 host、nslookup 或 dig 命令验证是否正确，验证正向与反向域名地址的解析是否正常，确保每个配资源记录的设置都是正确的。

下面的例子说明了怎样查询本地域名服务器 ns.abc.net，获取 ns.abc.net 的 IP 地址（仅当/etc/hosts 文件中存在类似于“192.168.90.100 ns.abc.net”的使用域名服务器的 IP 地址）。

```
$ host iscas.abc.net ns.abc.net  
Using domain server:  
Name: ns.abc.net  
Address: 192.168.90.100#53  
Aliases:  
  
iscas.abc.net has address 192.168.90.101
```

下面的例子说明了怎样利用本地 DNS 域名服务器 ns.abc.net，查询 www.ios.ac.cn 的 IP 地址。

```
$ host www.ios.ac.cn  
www.ios.ac.cn has address 124.16.136.250
```

然后再利用某个已知的域名服务器，如 linedns.bta.net.cn（202.106.196.115），验证查询的结果是否一致，如下所示。

```
$ host www.ios.ac.cn 202.106.196.115  
Using domain server:  
Name: 202.106.196.115  
Address: 202.106.196.115#53  
Aliases:  
  
www.ios.ac.cn has address 124.16.136.250
```

实际上，在利用 host 或 nslookup 命令查询本地系统的 DNS 服务时，并不需要指定域名服务器。在 host 或 nslookup 命令中，默认自动文件中指定的服务器，在我们的例子中即本地系统的域名服务器。

的命令形式，查询域名地址解析信息。

```
$ host iscas.abc.net
iscas.abc.net has address 192.168.90.101
$ host beijing.abc.net
beijing.abc.net has address 192.168.90.100
$ host 192.168.90.100
100.90.168.192.in-addr.arpa domain name pointer ns.abc.net.
100.90.168.192.in-addr.arpa domain name pointer mail.abc.net.
100.90.168.192.in-addr.arpa domain name pointer beijing.abc.net.
$ nslookup ns.abc.net
Server:      127.0.0.1
Address:     127.0.0.1#53

Name:      ns.abc.net
Address:   192.168.90.100
$ nslookup 192.168.90.102
Server:      127.0.0.1
Address:     127.0.0.1#53

102.90.168.192.in-addr.arpa    name = sinosoft.abc.net.
$
```

此外，还应使用 host、nslookup 或 dig 命令，查询其他域名服务器，获取本地主机的域名（如 iscas.abc.net），验证其是否能够收到新建域名服务器 ns.abc.net 维护的域名信息（注意，在新建或修改 DNS 域名服务器的配置文件之后，需要一定的时间才能将其传播到 Internet 上的其他域名服务器中）。

## 20.4.2 dig 命令

在 Linux 系统中，dig (domain information groper) 命令是一个重要的 DNS 域名服务器查询工具，使用灵活，输出信息丰富。利用 dig 命令可以查询指定域中的全部配置信息，许多 DNS 管理人员经常使用 dig 命令解决 DNS 配置方面的问题，其语法格式简写如下（第二个 dig 命令语法格式是最常用的调用形式）。

```
dig [name-server] [-b address] [-p port] [-t type] [-x address]
[domain-name] [type] [query-options]
dig [name-server] [domain-name] [type]
```

其中，name-server 是域名服务器的名字或 IP 地址。如果指定的是域名服务器的名字，dig 命令将会在查询域名服务器之前首先解析给定的名字。如果省略了 name-server 参数，dig 命令将访问/etc/resolv.conf 文件，查询其中列举的域名服务器。domain-name 是准备查询的域名。type 表示查询的资源记录类型，如 ANY (查询任何资源记录)、A (查询 IP 地址)、NS (查询域名服务器)、SOA (查询区域的授权记录)、MX (查询电子邮件服务器) 和 AXFR (查询区域传输记录) 等。如果省略了 type 参数，dig 主要查询 A 记录，即查询 IP 地址。

query-options 是 dig 命令支持的一系列能够影响其输出结果的附加选项，表 20-11 给出了部分常用的重要选项。

表 20-11

dig 命令支持的部分附加选项

附加选项	简单说明
+[no]additional	输出或禁止输出附加部分。dig 命令通常会输出附加部分
+[no]authority	输出或禁止输出授权信息部分。dig 命令通常会输出授权信息部分
+[no]cmd	输出或禁止输出 dig 命令的版本信息和预置的查询选项。dig 命令通常会输出版本等信息
+[no]all	设置或清除所有的显示标志
+[no]comments	输出或禁止输出注解和标题信息。dig 命令通常会输出注解和标题信息
+[no]multiline	输出或禁止输出多行形式的 SOA 资源记录。通常，dig 命令每行显示一个资源记录，因此，SOA 记录将会压缩在一行中输出



续表

附加选项	简单说明
+[no]question	返回查询结果时是否输出查询请求部分。作为注解, dig 命令通常会输出查询请求部分
+[no]recurse	设置或清除查询请求中的 RD (recursion desired) 标志位。RD 标志位是一个默认设置, 这意味着 dig 命令通常都会执行递归查询。当使用 “+trace” 附加选项时, dig 命令将会自动地禁用递归查询功能
+[no]search	使用或禁止使用/etc/resolv.conf 文件中 search 关键字列举的域名或 domain 关键字定义的域名组合完整的域名。通常, dig 命令不使用/etc/resolv.conf 文件中 search 关键字列举的域名
+[no]short	提供详细还是简短的查询结果, 即确定是否采用最简洁的形式回答查询请求。dig 命令通常会提供详细的查询结果
+[no]stats	输出或禁止输出查询统计信息。dig 命令通常会输出查询统计信息
+[no]trace	在查询域名解析信息时, 确定是否启用从根域 DNS 服务器开始的查询路径跟踪功能。dig 命令通常禁用查询路径的跟踪功能。启用跟踪功能时, dig 命令将会采用迭代方式查询域名解析信息, 并从根域 DNS 服务器开始, 显示每一个 DNS 服务器处理域名解析请求时返回的查询结果

除非明确指定了域名服务器, dig 将会依次尝试查询/etc/resolv.conf 文件中列举的每一个域名服务器。如果省略了域名服务器或命令行参数, dig 将会查询根域 (“.”)。

注意, 在指定域名服务器或要查询的域及主机时, 必须给出完整的域名, dig 命令通常不会引用/etc/resolv.conf 文件中的 search 或的 domain 关键字, 自动组合一个完整的域名, 除非使用 “+search” 选项。

在下面的例子中, AXFR 参数用于获取 abc.net 域中维护的区配置文件的信息。

```
$ dig abc.net AXFR
; <>> DIG 9.4.2-P1  <>> abc.net AXFR
;; global options:  printcmd
abc.net.          259200  IN      SOA     ns.abc.net. gqxing.gmail.com. 20
0812012 28800 7200 2419200 86400
abc.net.          259200  IN      NS      ns.abc.net.
abc.net.          259200  IN      MX      0 mail.abc.net.
beijing.abc.net. 259200  IN      A       192.168.90.100
iscas.abc.net.   259200  IN      A       192.168.90.101
mail.abc.net.    259200  IN      A       192.168.90.100
news.abc.net.    259200  IN      A       153.78.26.168
ns.abc.net.       259200  IN      A       192.168.90.100
sinosoft.abc.net. 259200  IN      A       192.168.90.102
www.abc.net.     259200  IN      A       153.78.26.166
yantai.abc.net.  259200  IN      A       192.168.90.200
abc.net.          259200  IN      SOA     ns.abc.net. gqxing.gmail.com. 20
0812012 28800 7200 2419200 86400
;; Query time: 53 msec
;; SERVER: 127.0.0.1#53(127.0.0.1)
;; WHEN: Sat Nov 1 19:05:54 2008
;; XFR size: 12 records (messages 1, bytes 318)
$
```

在一个内部网络中, 通常都会禁止传输区配置文件, 以免遭受非法入侵和攻击。为了禁止传输区配置文件, 可在 named.conf.options 配置文件中增加下列 allow-transfer 语句。

```
options {
    ...
    allow-transfer { none; };
};
```

一旦增加了上述语句, 重新启动 named 守护进程之后, 即可禁止传输区配置文件, 示例如下。

```
$ dig abc.net AXFR
; <>> DIG 9.4.2-P1 <>> abc.net AXFR
;; global options:  printcmd
;; Transfer failed.
$
```

dig 命令的输出通常包含 7 部分。下面是一个利用 “dig domain ns” 命令形式和默认的域名服务器, 查询指定域 (如 abc.net) 的域名服务器的例子。

```
$ dig abc.net ns
; <>> DiG 9.4.2-P1 <>> abc.net ns
;; global options: printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 32348
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; QUESTION SECTION:
abc.net.           IN      NS

;; ANSWER SECTION:
abc.net.        259200  IN      NS      ns.abc.net.

;; ADDITIONAL SECTION:
ns.abc.net.      259200  IN      A       192.168.90.100

;; Query time: 1 msec
;; SERVER: 127.0.0.1#53(127.0.0.1)
;; WHEN: Sat Nov 1 19:27:22 2008
;; MSG SIZE rcvd: 58
$
```

在上述输出信息中，第一部分给出了 dig 命令的版本号（如 9.4.2）与设置的全局选项（如 printcmd）等信息。要禁止输出这一部分内容，可在 dig 命令中增加“+nocmd”选项（通常应放在命令选项和参数的最前面）。

第二部分是对 DNS 服务器返回的查询结果的简单说明。如想禁止输出这一部分内容，可在 dig 命令中增加“+nocomments”选项。但是，禁止输出这一部分内容也会禁止输出其他部分的标题信息。

在 QUESTION 部分，dig 命令提示用户执行的是哪一种查询。默认的查询为 IP 地址，相当于指定的查询类型为 A。如果想要禁止输出这一部分内容，可在 dig 命令中增加“+noquestion”选项。

“ANSWER”部分是 dig 命令返回的查询结果，如查询的域名服务器为 ns.abc.net。

如果存在，“AUTHORITY”部分表示哪个 DNS 服务器能够提供权威的查询结果。如想禁止输出这一部分内容，可在 dig 命令中增加“+noauthority”选项。

“ADDITIONAL”部分通常包含“AUTHORITY”部分列举的域名服务器的 IP 地址，如 192.168.90.100。如果想要禁止输出这一部分内容，可在 dig 命令中增加“+[no]additional”选项。

最后一部分是查询的统计信息，包括查询的时间、使用的域名服务器以及返回的记录数量等。如想禁止输出这一部分内容，可在 dig 命令中增加“+nostats”选项。

这种常规的输出形式包含太多不必要的信息，显得重点不突出，为此，可以利用 dig 命令的附加选项，仅输出想要的信息。示例如下。

```
$ dig +noall abc.net ns +answer +additional
abc.net.        259200  IN      NS      ns.abc.net.
ns.abc.net.     259200  IN      A       192.168.90.100
$
```

使用下列 dig 命令和默认的域名服务器（ns.abc.net），可以获取根（root）域名服务器的最新信息。

```
$ dig +noall . ns +additional
B.ROOT-SERVERS.NET. 603420  IN      A       192.228.79.201
C.ROOT-SERVERS.NET. 603420  IN      A       192.33.4.12
L.ROOT-SERVERS.NET. 603420  IN      A       199.7.83.42
F.ROOT-SERVERS.NET. 603420  IN      A       192.5.5.241
F.ROOT-SERVERS.NET. 603420  IN      AAAA    2001:500:2f::f
K.ROOT-SERVERS.NET. 603420  IN      A       193.0.14.129
K.ROOT-SERVERS.NET. 603420  IN      AAAA    2001:7fd::1
M.ROOT-SERVERS.NET. 603420  IN      A       202.12.27.33
M.ROOT-SERVERS.NET. 603420  IN      AAAA    2001:dc3::35
H.ROOT-SERVERS.NET. 603420  IN      A       128.63.2.53
H.ROOT-SERVERS.NET. 603420  IN      AAAA    2001:500:1::803f:235
G.ROOT-SERVERS.NET. 603420  IN      A       192.112.36.4
J.ROOT-SERVERS.NET. 603420  IN      A       192.58.128.30
J.ROOT-SERVERS.NET. 603420  IN      AAAA    2001:503:c27::2:30
$
```

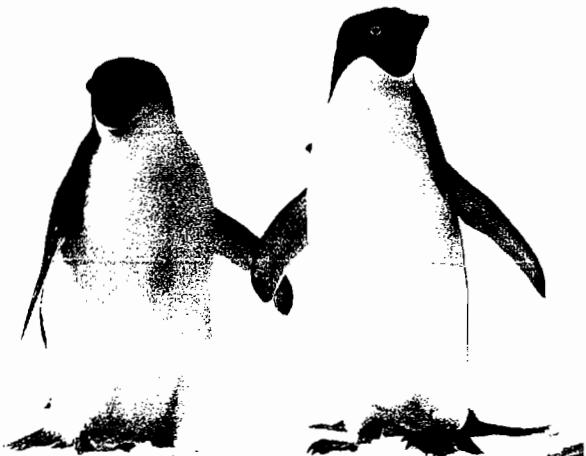


## 第21章

### NFS 网络文件系统

网络文件系统（Network File System， NFS）是一种分布式文件系统，使用户能够透明地访问远程资源。利用 NFS，用户能够共享文档、数据和程序等。对于用户而言，远程资源无非就是本地文件系统中的一个目录层次，与其他目录和文件没有任何差别。本章将讨论怎样设置 NFS 服务器和 NFS 客户系统，以及怎样共享远程资源。其内容主要包括：

- NFS 简述；
- 配置 NFS 服务器；
- 配置 NFS 客户系统；
- NFS 自动安装；
- NFS 故障修复。



## 21.1 NFS 简述

在 NFS 网络文件系统中，客户系统与服务器用于描述每个系统在文件系统共享中各自扮演的角色。通常把通过网络提供共享文件系统资源的主机称作 NFS 服务器，而访问共享资源的系统称作 NFS 客户系统。

NFS 是一种基于远程过程调用（Remote Procedure Call, RPC）协议，采用客户端/服务器结构实现的分布式文件系统。在 NFS 的网络环境中，一个 Linux 系统既可以作为 NFS 服务器，提供共享的文件系统或目录，供 NFS 客户系统访问，也可以作为 NFS 客户系统，安装其他 NFS 服务器提供的远程文件系统或目录，访问其中的目录和文件资源。

远程资源可安装于本地系统的任何目录位置。一旦安装了 NFS 服务器提供的远程共享文件系统，只要给予足够的访问权限，客户系统用户就可以像访问本地文件系统一样，透明地访问服务器中的目录和文件，可以读写、创建和删除其中的文件与目录。

NFS 是一种强有力的数据集中存储与客户访问的工具，使客户系统能够快速地访问大批量的数据，如存储在中央数据库中的业务数据。尤其是，还可以采用 NFS 的方式，利用 NFS 服务器中存储的 CD/DVD 影像数据，直接安装各种系统软件，如采用 NFS 方式安装 Ubuntu Linux 或 Solaris 系统等。另外，还可以利用 NFS 服务器存储引导程序，直接引导客户系统。

当前可用的 NFS 主要有 3 个版本，即版本 2、3 和 4（版本 1 只是一个原型）。NFS 版本 2 的主要功能特点如下：

- 数据传输单位的上限为 8KB；
- 处理的文件最大不能超过 2GB；
- 使用 UDP 作为底层传输协议；
- 仅当把数据完全写到 NFS 服务器的磁盘之后才算完成数据的写请求。

NFS 版本 3 是在版本 2 基础上的增强与改进，其主要功能特点如下：

- 数据传输单位的上限为 64KB，实际的数据传输限制是客户系统与服务器协商后共同确定的一个优化值；
- 能够处理大于 2GB 的文件；
- 支持 TCP 和 UDP，并以 TCP 作为首选的底层传输协议；
- 支持异步处理，把数据写到 NFS 服务器的缓冲区之后就算完成了数据的更新，客户系统不需要等待，因而改善了响应时间；
- NFS 服务器能够以批处理的方式集中、统一处理每个客户系统的访问请求，因而能够提高 NFS 服务器的数据处理性能。

NFS 版本 4 又是在版本 3 基础上的增强与改进，其主要功能特点如下：

- 支持互联网与万维网等网络应用环境，改善并提高了其性能与安全性；
- 扩展了 NFS 文件系统的文件属性，支持访问控制表 ACL；
- 采用强制性的文件锁，而之前是选用的，文件的加锁与远程安装均集成到 NFS 的后台守护进程；



- 使用单一的 TCP 端口，用于增强网络安全；
- 增加了基于 RPCSEC\_GSS (Remote Procedure Call Security - Generic Security Services) 的 API，采用 GSS-API (Generic Security Services Application Programming Interface)，从客户系统与服务器中选用双方均支持的安全机制。

注意，客户系统与服务器中运行的 NFS 版本必须匹配和兼容，才能确保 NFS 网络文件共享的顺利实现。

## 21.2 配置 NFS 服务器

### 21.2.1 安装 NFS 服务器软件包

NFS 服务器与客户系统需要运行相应的 NFS 软件包。配置 NFS 服务器通常包括两个步骤：安装 NFS 服务器软件包，以及配置/etc(exports 文件，公布目录或文件系统等共享资源。在安装 Ubuntu Linux 系统之后，要想增加 NFS 服务器功能，需要单独安装 nfs-kernel-server 等软件包。NFS 主要提供下列软件包。

- **nfs-kernel-server**: Ubuntu 数据仓库提供了 2 个 NFS 服务器软件包，其中，nfs-kernel-server 在 Linux 核心空间模式下运行，nfs-user-server 在用户空间模式下运行。前者运行速度较快，且提供若干命令行工具，如 exportfs。后者易于手工调试与控制。
- **nfs-common**: 正如其名字所示，nfs-common 是一个通用的支撑软件包，包括 rpc.statd、rpc.idmapd 及 rpc.gssd 等实用程序，为 NFS 服务器和 NFS 客户系统提供底层支持，包括 NFS 安装功能。
- **portmap**: portmap 是 NFS 的 RPC 端口映射服务器，也是其他 NFS 守护进程的底层支持软件，其功能是把 RPC 程序号转换成 TCP/IP 端口号。portmap 用于提供 NFS 协议需要的 RPC 连接服务，管理和维护基于 RPC 规范的网络连接与数据传输。通常，portmap 将会监听 TCP 端口 111 的初始连接请求，然后以此为基础，由 NFS 服务器与客户系统双方协商一个 TCP 端口范围（通常使用大于 1024 的端口），以便进行随后的数据传输。在 NFS 服务器与客户系统中，都需要启用 portmap 守护进程。

要安装 NFS 服务器软件包，可以使用 apt-get、aptitude 和 synaptic 等软件维护工具，示例如下。

```
$ sudo apt-get install nfs-kernel-server
```

在安装过程中，安装程序会要求同时安装 nfs-common 和 portmap 等相关的底层支持软件包。安装之后，/etc/init.d 目录中将会存在 nfs-kernel-server、nfs-common 及 portmap3 个启动脚本。使用 NFS 启动脚本以及 ps、pgrep 或 pidof 等命令，可以检查 portmap、nfs-common 与 nfs-kernel-server 的运行状态，确保 NFS 服务器的守护进程 nfsd、nfsd4、rpc.mountd、lockd、rpc.statd 与 portmap 等均已正常地运行，如下所示。

```
$ sudo /etc/init.d/portmap status
* portmap is running.
$ sudo /etc/init.d/nfs-common status
all daemons running
$ sudo /etc/init.d/nfs-kernel-server status
nfsd running
$
```

## 21.2.2 /etc(exports) 文件

在 NFS 服务器中，要公布 NFS 客户系统共享的目录或文件系统，可以通过/etc(exports) 文件和 exportfs 命令实现。/etc(exports) 是 NFS 服务器的主要配置文件，用于定义 NFS 服务器提供的目录或文件系统等共享资源，设置共享资源的访问控制表。exportfs 命令用于公布、撤销 NFS 服务器提供的共享资源，监控 NFS 共享资源的状态。

在 exports 配置文件中，每一行由两个字段组成：第一个字段用于指定 NFS 服务器提供的共享目录或文件系统等，第二个字段由一个或多个 NFS 客户系统组成（中间可加任何空白字符分隔符），表示允许其安装、访问相应的共享资源。第二个字段定义的每个 NFS 客户系统又由两部分组成：第一部分用于指定 NFS 客户系统的名字、IP 地址、网络或 DNS 意义上的域，第二部分是位于圆括号中的 NFS 安装与访问选项，其语法格式如下。

```
Directory Host (Options) Host (Options) ...
```

其中，“Directory”可以是 NFS 服务器中的一个目录或文件系统。“Host”可以是单个主机名或 IP 地址，如果在主机名中增加星号通配符“\*”与问号“？”，也可以表示一个域中的所有主机。IP 地址可以是一个特定的 IP 地址，也可以采用“address/netmask”的形式表示一个网段。“Options”选项用于指定目录或文件系统的共享方式与访问限制，多个选项之间需加空格分隔符。注意，“Host”与“(Options)”之间不能有任何空白字符。

在指定 DNS 客户系统时，可以采用下列形式之一或其组合。

- 单个主机——这是最常用一种主机定义格式，可以指定一个普通的主机名、标准域名或 IP 地址。
- 多个主机——在指定主机名时，可以采用通配符星号“\*”与问号“？”，简化配置文件的设置。例如，\*.example.edu 表示 example.edu 域中的所有主机。
- 网络地址——还可以采用“address/netmask”的形式同时指定位于某个子网范围内的所有主机。其中，address 可以是子网中的一个 IP 地址或 IP 网号，netmask 既可是一个标准的子网掩码，也可以是一个表示网络部分位长的数字。例如，192.168.90.0/24 表示 192.168.90.0 子网中的所有主机。注意，在指定 IP 地址时，不要使用通配符。

注意，每个选项之间的逗号“，”前后不能加任何空白字符。此外，在指定的共享目录与 NFS 客户系统之间，可以采用“-option,option...”的形式，指定若干安装选项，如“-sync,rw”。这些选项适用于当前行中指定的任何主机，可以用作 NFS 客户系统安装共享资源的默认选项。表 21-1 给出了部分常用的选项。

表 21-1

exports 文件的部分选项

选 项	简 单 说 明
secure / insecure	secure 选项（默认）要求采用低于 IPPORT_RESERVED（1024）的端口号建立连接。要取消这一限制，需要明确指定 insecure 选项
rw/ro	rw 表示允许 NFS 客户系统读写共享目录或文件系统。ro（默认值）表示 NFS 客户系统只能读取共享资源中的文件或目录，不允许创建、写入及修改共享资源中的文件或目录
async	允许 NFS 服务器采用异步方式处理客户系统的数据读写请求。使用这个选项能够改善 NFS 服务器的性能，但在 NFS 服务器不稳定的情况下也有可能引起数据的丢失与损坏
sync	表示强制 NFS 服务器采用同步方式处理客户系统的数据写操作，客户系统将会等待，直至服务器的数据写操作完成之后，才把控制权返回客户系统用户。这是一种默认的选项



续表

选 项	简 单 说 明
root_squash	NFS 服务器基于每个 NFS 请求提供的 UID 和 GID 实现文件的访问控制，使每个用户都能按照符合其身份的权限访问服务器中的文件。但对于超级用户而言，为确保系统的安全起见，NFS 服务器不希望 NFS 客户系统中的超级用户 root 仍以超级用户的身分访问服务器中的文件。因此，利用 root_squash 选项，NFS 服务器把 NFS 客户系统中超级用户 root 的 UID/GID (0) 映射到匿名用户 nobody 的 UID/GID (65 534)。这是 NFS 服务器默认的一种处理方式
no_root_squash	取消对 NFS 客户系统超级用户的上述安全限制，使 NFS 客户系统中的超级用户具有与服务器系统中的超级用户相同的访问权限
all_squash / no_all_squash	all_squash 选项表示把 NFS 客户系统中所有读写请求的 UID/GID 均映射为匿名用户 nobody 的 UID/GID (65534)。对于提供公共资源的 NFS 服务器，如文件下载服务器，经常采用这个选项。NFS 服务器的默认设置是 no_all_squash，其意义恰好相反，即保持一对一的映射关系
subtree_check / no_subtree_check	禁止 NFS 目录子树的检测。如果仅公布了文件系统的一个子目录，而没用公布整个文件系统，无论何时一个 NFS 请求到达 NFS 服务器，服务器首先必须检查访问的文件是否位于适当的文件系统，而后还要检查其是否位于公布的共享目录树中。这种检测称作 subtree_check。由于目录子数检测有可能导致潜在的问题，新版的 NFS 服务器采用 no_subtree_check 作为默认选项，但尽管如此，如果未指定两者之一，运行 exportfs 命令时仍会给出警告信息

下面是一个/etc(exports)示例文件，我们将逐一说明其中每个配置选项的实际意义，并借以说明怎样配置 exports 文件。

```
$ cat /etc(exports
# sample /etc(exports file
/
master(rw) icas(ro,no_root_squash)
/projects    proj*.local.com(rw)
/data        *.local.com(ro) 192.168.90.100/24(rw)
/pub         (ro,insecure,all_squash)
/srv/www     -sync,rw icas sinosoft *.example.com(ro)
$
```

上述第 1 行定义意味着 master 与 icas 两个 NFS 客户系统均可读写整个文件系统，除此之外，还取消了 icas 系统中超级用户的 UID 映射限制。第 2 行采用的通配符表示，local.com 域中的所有主机，如果其名字前 4 个字符为“proj”，均具有读写/projects 目录的权限。第 3 行表示 local.com 域中的所有主机只能读取/data 目录中的文件，但“192.168.90.0”网段中的所有主机则可以读写/data 目录中的中文件。第 4 行意味着在网上公布了一个 FTP 目录，但所有的主机只能以匿名用户 nobody 的身份访问/pub 目录中的文件，其中的“insecure”选项表示可以不必使用 NFS 的保留端口。第 5 行公布了一个共享目录，赋予 icas 与 sinosoft 读写的访问权限，但 example.com 域中的任何主机只能读取其中的文件。此外，在安装相应的远程资源时，icas、sinosoft 及 example.com 域中的所有主机将会自动采用 sync 选项。

假定 NFS 服务器计划提供两个共享目录/share/tools 和/share/docs，允许任何客户端系统读取 /share/tools 目录中的文件，但仅允许 192.168.90.0 子网中的客户端系统读写/share/docs 目录中的文件。据此，可以写出下列/etc(exports)文件。

```
$ cat /etc(exports
#/etc(exports
/share/tools      *(ro,all_squash)
/share/docs       192.168.90.0/24(rw,sync,no_root_squash)
$
```

其中，第一行表示所有 NFS 客户端系统中的任何用户只能以匿名用户 nobody 的身份读取 /share/tools 目录中的文件。第二行表示“192.168.90.0”网段中的任何主机均可以读写/share/docs 目录中的文件，且在安装时采用“sync”选项确保内存缓冲与磁盘文件之间的数据完整性和一致性，同时取消了任何主机中超级用户的 UID 映射限制。

### 21.2.3 采用图形界面配置 NFS 共享资源

若要配置与公布 NFS 服务器提供的共享目录或文件系统等资源，还可以在命令行中调用一个图形界面的工具 shares-admin，以简化上述手工配置的过程。当出现一个共享文件夹界面时，单击“解锁”按钮，输入 sudo 密码之后，最终将会出现图 21-1 所示的界面。

要定义一个 NFS 共享目录，如 /home/gqxing/docs，可以点击“添加”按钮，出现图 21-2 所示的“共享文件夹”界面。



图 21-1 共享的文件夹界面

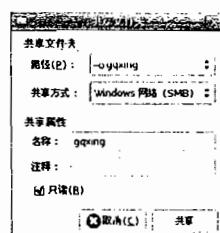


图 21-2 共享文件夹界面 (1)

由图 21-3 可知，默认的设置是 SMB 共享方式，适用于 Samba 资源共享环境（参见第 22 章）。要公布一个 NFS 共享目录，可点击“共享方式”下拉框，选择 NFS 共享方式，出现一个适用于 NFS 环境的共享资源配置界面，参见图 21-3 所示。

在图 21-3 中，默认的共享资源是当前工作目录，在此例中即笔者的主目录 /home/gqxing。要共享其中的文档子目录 docs，可点击“路径”下拉框，在出现的下拉菜单（目录列表）中选择“其它”，随之出现一个目录选择界面，参见图 21-4。

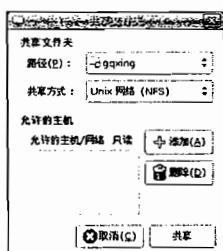


图 21-3 共享文件夹界面 (2)

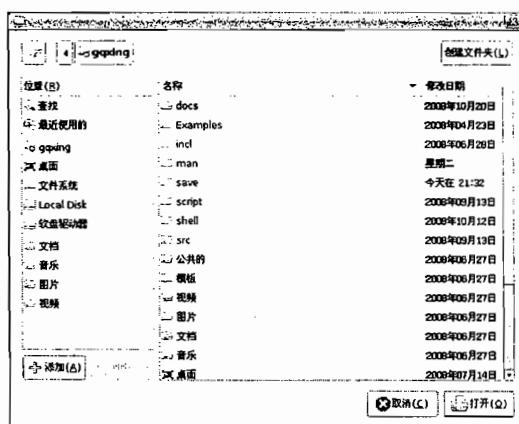


图 21-4 目录选择界面

要选择系统中的其他目录，可以点击“位置”窗口中的“文件系统”，从中选择一个准备公布的共享目录。如果想要共享当前目录中一个子目录，如 docs，可在右边的窗口中选择 docs，然后点击“打开”按钮，即可出现图 21-5 所示的界面。其中，“路径”选择框中的路径名已变为新选择的 docs。注



意，如果目录的层次较多，可以逐次双击右边窗口的目录，直至进入目标目录，再点击“打开”按钮。

然后，需要指定哪些 NFS 客户系统能够安装、访问当前设定的共享目录。为此，可以单击“添加”按钮，显示图 21-6 所示的界面。在这个界面中，通过单击“允许的主机”下拉菜单，可以采用 3 种方式指定主机——指定主机名、指定 IP 地址以及指定网络。

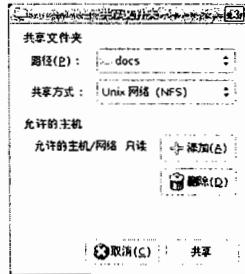


图 21-5 共享文件夹界面 (3)

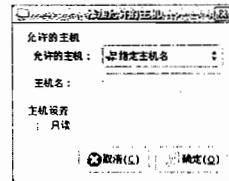


图 21-6 共享文件夹界面 (4)

- 指定主机名：用于指定主机的名字，允许指定的客户系统安装与访问相应的共享目录。选择此项时将会显示一个附加的“主机名”字段，参见图 21-7 所示。在“主机名”字段中，可输入一个普通的主机名或规范的域名。
- 指定 IP 地址：用于指定主机的 IP 地址，允许指定的客户系统安装与访问相应的共享目录。选择此项时会显示一个附加的“IP 地址”字段，参见图 21-8 所示。在“IP 地址”字段中，可输入一个 IP 地址。

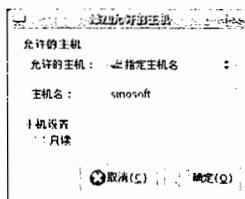


图 21-7 添加允许的主机 (1)

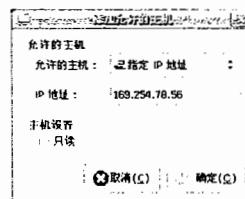


图 21-8 添加允许的主机 (2)

- 指定网络：用于指定一个子网，允许其中的任何客户系统安装与访问相应的共享目录。选择此项时会显示两个附加的“网络”与“子网掩码”字段，参见图 21-9 所示。在“网络”字段中，可输入一个网络地址或一个有代表性的主机 IP 地址。在“子网掩码”字段中，可输入一个标准的网络掩码或一个表示网络部分位长的数字。图 21-9 的例子表示，192.168.90.0 子网中的任何主机均可读取/home/gqxing/docs 共享目录中的文件，但不能创建新文件，也不能修改任何文件。

注意，在采用指定主机名与指定 IP 地址（也包括指定网络）的情况下，如果想要指定多个主机，使之能够安装、共享同一个共享目录资源，需要多次点击“添加”按钮，重复上述步骤。此外，在指定主机、IP 地址与网络的界面中，如果只允许指定的主机读取相应的共享资源，防止它们修改其中的文件，可以勾选“只读”复选框。

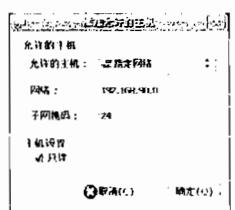


图 21-9 添加允许的主机 (3)

单击图 21-7、图 21-8 或图 21-9 所示界面中的“确定”按钮，将会出现图 21-10 所示的界面，其中反映了最终的设置。在本例中，新增的/home/gqxing/docs 共享目录允许 sinosoft 主机（读写）、IP 地址 169.254.78.56 对应的主机（读写）以及 192.168.90.0 子网中的所有主机（只读）访问。

在图 21-10 所示的界面中，再次点击“共享”按钮后，即可出现图 21-11 所示的界面，其中将会增加一个新定义的 NFS 共享资源。

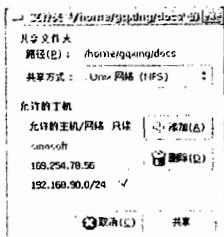


图 21-10 共享文件夹界面 (5)

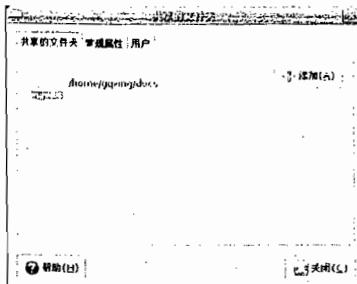


图 21-11 最终的共享文件夹设置界面

之后，如果想要修改或删除任何共享目录，仍可调用 shares-admin 命令，在图 21-11 所示的界面中，选择其中的某个共享目录，单击“属性”按钮修改相应的共享资源，或者单击“删除”按钮，删除相应的共享资源。

#### 21.2.4 验证 NFS 共享资源的配置

在修改/etc(exports 配置文件之后，为使共享资源的变动立即生效，可使用“exportfs -ra”命令，立即公布或取消变动后的目录或文件系统等共享资源，或者重新启动 NFS 服务器，重读 exports 文件，并按照其中的设置公布共享的目录或文件系统资源。exportfs 命令的语法格式简写如下。

```
exportfs [-auv] [-o options]
exportfs -r [-v]
```

其中，“-a”选项用于公布或取消 NFS 服务器中所有的共享目录或文件系统。“-r”选项用于重新公布新加到/etc(exports 文件中的共享目录或文件系统，使用/etc(exports 文件中的内容对/var/lib/nfs/etab 文件进行同步，确保 etab 与 exports 文件中的内容完全一致。“-u”选项用于取消已经公布的共享目录或文件系统。“-v”选项用于提供更详细的说明信息。在显示公布的共享目录或文件系统时，也会显示相关的选项。

为了避免重新启动 NFS 服务器或系统，危及数据的完整性，可采用下列方法处理共享资源的变动。

- 采用“exportfs -a”命令，使 NFS 服务器重读/etc(exports 文件，然后按照新的设置公布所有的共享资源。
- 每当增加一个新的共享目录或文件系统资源时，可使用“exportfs -r”命令公布新增的共享资源，以便 NFS 客户系统能够及时访问。
- 为了从/etc(exports 文件中删除已经公布的本地目录或文件系统，需要 NFS 客户系统先行卸载已经安装的远程资源（必要时还需要编辑/etc/fstab 文件，临时注销或永久删除相应的 NFS 安装项），然后编辑 NFS 服务器中的/etc(exports 文件，临时注销或永久删除相应的本地目录或文件系统。最后使用“exportfs -u”命令撤销已经公布的共享资源。

在采用前面介绍的图形界面工具 shares-admin 设置共享资源时，最终生成的/etc(exports 文件如下。



```
$ cat /etc/exports
```

```
....  
/home/gqxing/docs sinosoft(rw) 169.254.78.56(rw) 192.168.90.0/24  
$
```

要验证上述设置是否正确，是否已经正确地公布了/home/gqxing/docs 共享目录，可以使用下列 exportfs 命令。

```
$ sudo exportfs -a  
exportfs: /etc/exports [1]: Neither 'subtree_check' or 'no_subtree_check' specified for  
export "sinosoft:/home/gqxing/docs".  
Assuming default behaviour ('no_subtree_check').  
NOTE: this default has changed since nfs-utils version 1.0.x  
  
exportfs: /etc/exports [1]: Neither 'subtree_check' or 'no_subtree_check' specified for  
export "169.254.78.56:/home/gqxing/docs".  
Assuming default behaviour ('no_subtree_check').  
NOTE: this default has changed since nfs-utils version 1.0.x  
  
exportfs: No options for /home/gqxing/docs 192.168.90.0/24: suggest  
192.168.90.0/24(sync) to avoid warning  
exportfs: /etc/exports [1]: Neither 'subtree_check' or 'no_subtree_check' specified for  
export "192.168.90.0/24:/home/gqxing/docs".  
Assuming default behaviour ('no_subtree_check').  
NOTE: this default has changed since nfs-utils version 1.0.x  
$
```

上述输出信息表明，利用 shares-admin 图形界面工具生成的/etc/exports 配置文件并不完全适用，exports 命令建议增加一个 sync 选项，同时明确指定究竟是采用 subtree\_check，还是采用 no\_subtree\_check（默认）选项。因此，可以把 exports 文件修改如下。

```
$ cat /etc/exports  
....  
/home/gqxing/docs sinosoft(rw, sync, no_subtree_check) 192.168.90.101(rw, sync, no_sub  
tree_check) 192.168.90.0/24(no_subtree_check)  
$
```

再次运行 exports 命令后，其输出结果表明/home/gqxing/docs 共享目录已经公布，sinosoft 和 169.254.78.56 主机中的用户均可读写新设的共享目录，而 192.168.90.0 网段中的其他主机只能读取新设共享目录中的文件。

```
$ sudo exportfs -av  
exporting sinosoft:/home/gqxing/docs  
exporting 169.254.78.56:/home/gqxing/docs  
exporting 192.168.90.0/24:/home/gqxing/docs  
$
```

注意，exportfs 命令只是建议，不一定非要增加其提示的选项，尤其是 no\_subtree\_check。至于究竟是否采用 sync 同步方式，还取决于用户与实际的应用场合。同步方式相对较慢，因为必须等待 NFS 服务器完成数据读写操作之后才能继续访问服务器，但比较安全；async 异步方式虽然运行速度较快，但数据安全性相对差一点。

此外，要查询共享目录或文件系统资源的公布与使用情况，也可以使用 showmount 命令。showmount 命令主要用于查询 NFS 服务器提供的共享资源、运行状态与客户系统的安装信息，其语法格式简写如下。

```
showmount [-ade] [hostname]
```

其中，“-a”选项表示以“host:directory”的形式显示已经安装了远程资源的 NFS 客户主机名（或 IP 地址）以及安装的目录或文件系统。“-d”选项仅用于显示 NFS 客户系统安装的目录或文件系统。“-e”选项用于显示 NFS 服务器公布的共享目录或文件系统。

例如，下列 showmount 命令也可用于显示 NFS 服务器已经公布的共享目录或文件系统。

```
$ showmount -e  
Export list for iscas:  
/home/gqxing/docs 192.168.90.0/24,169.254.78.56,sinosoft  
$
```

下列命令将会列出客户系统已经安装的远程目录或文件系统。

```
$ showmount -d
Directories on sinosoft:
/home/gqxing/docs
$
```

## 21.3 配置 NFS 客户系统

如果已经安装了 NFS 服务器，无需单独安装 NFS 客户系统软件包，否则至少还需要安装 nfs-common 与 portmap 等软件包。配置 NFS 客户系统主要包括两个任务：采用 mount 命令，手工安装 NFS 服务器提供的远程目录或文件系统等共享资源；利用/etc/fstab 文件，自动安装远程目录或文件系统等共享资源。必要时，还需要创建本地目录，作为远程资源的安装点。

### 21.3.1 安装远程文件系统

在 NFS 客户系统中，可以使用 mount 命令，直接安装 NFS 服务器提供的远程文件系统或目录，其语法格式如下。

```
mount [-t nfs] [generic_opts] [-o specific_opts] resource mount_point
```

其中，“-t nfs”表示安装 NFS 网络文件系统，“generic\_opts”为通用的安装选项，“-o specific\_opts”为 NFS 网络文件系统特定的安装选项，“resource”表示远程共享资源的文件系统或目录名，由形如“host.pathname”的远程主机名与路径名组成，“mount\_point”为本地系统的安装点，即安装文件系统的目录位置。mount 命令的通用选项、NFS 特定的选项及其说明如表 21-2 所示。

表 21-2

mount 命令的部分 NFS 选项

选 项	简 单 说 明
ac/noac	指定 NFS 客户系统是否缓存文件属性信息。如果 ac 与 noac 选项均未被指定，或者指定了 ac 选项，NFS 客户系统将会缓存文件的属性信息。为了改善性能，NFS 客户系统通常都会缓存文件属性信息，定时查询服务器，更新文件的属性信息。noac 选项表示禁用文件属性缓冲处理机制，防止 NFS 客户系统缓存文件属性信息，使得应用程序能够更快地检测到 NFS 服务器中的文件属性变化，同时也强制应用程序采用同步方式实现文件的写操作，确保服务器总是能够维护最新的文件数据与属性信息，其他客户系统也能够及时了解文件的最新变动。由此可见，采用 noac 选项能够确保数据的一致性，但以降低性能为代价。因此，究竟选用 ac 还是 noac 选项，取决于具体的应用场合与需求
bg   fg	在安装远程目录或文件系统时，如果第一次安装因超时而失败，客户系统将会尝试以“bg”（后台）或“fg”（前台）方式重新安装远程目录或文件系统，直至安装成功，或因超过指定的尝试安装时间而放弃。默认的安装方式为“fg”，尝试安装时间为 10 000 分钟（约为 1 周）。后者将会导致一个问题，如果 NFS 服务器因网络的原因而无法建立连接，mount 进程有可能挂起约 1 周的时间才能返回。而采用“bg”选项安装远程目录或文件系统则意味着，如果服务器没有响应，mount 将以后台方式尝试重新安装，直至安装成功，或者达到“retry=n”选项指定的尝试安装时间（默认值为 2 分钟）（参见“retry”选项）
hard   soft	表示一旦当 NFS 服务器在规定的超时值范围内没有响应 NFS 客户系统的访问请求时如何处理。“soft”安装选项意味着，当 NFS 访问失败时放弃继续处理，然后返回一个错误信息。由此可见，采用“soft”选项安装的远程目录或文件系统无法保证数据的完整性和一致性。而“hard”安装选项表示，当 NFS 访问失败时将会在控制终端上输出一个“server not responding”的出错信息，然后继续不断地尝试，直至服务器响应访问请求为止，因而能够确保数据的完整性和一致性（默认值为“hard”）。以读写形式访问 NFS 远程资源，或安装一个包含可执行文件的远程目录或文件系统时，应当总是使用“hard”选项
intr   nointr	指定是否允许使用键盘的中断键（Ctrl-C）或 kill 命令中止因等待 NFS 服务器响应而挂起的文件访问进程。对于采用“hard”选项安装的远程目录或文件系统，一旦出现了超时，允许使用键盘的中断键或 kill 命令中止调用的程序
retrans=n	指定因超时而导致 NFS 访问请求重发的次数，默认值为 3 次。在连续重发“retrans”选项指定的次数之后，要么放弃远程文件的继续处理，要么在控制终端上输出一个“server not responding”的出错信息



续表

选 项	简 单 说 明
retry= <i>n</i>	指定安装远程目录或文件系统失败时，在放弃安装前尝试以前台方式或后台方式重新安装的持续时间， <i>n</i> 的时间单位为分钟。前台安装方式的默认值为 2 分钟，后台安装方式的默认值为 10 000 分钟（约为 1 周的时间）
ro   rw	指定以“ro”（只读）或“rw”（读写）方式安装和访问远程目录或文件系统，默认值为“rw”
rsize= <i>n</i>	设置从 NFS 服务器中读取文件时所用缓冲区的最大字节数。默认值取决于 NFS 的版本，NFS V3 和 V4 的默认值为 32 768，NFS V2 的默认值为 8 192。实际的数值是 NFS 服务器与客户系统能够支持的最大值经过协商后的结果。注意，设置这个数值时不能小于 NFS 服务器与客户系统双方都支持的最大值，否则将会影响 NFS 的性能
wsize= <i>n</i>	设置向 NFS 服务器写文件时所用缓冲区的最大字节数。默认值取决于 NFS 的版本，NFS V3 和 V4 的默认值为 32 768，NFS V2 的默认值为 8 192。实际的数值是 NFS 服务器与客户系统能够支持的最大值经过协商后的结果。同样，设置这个数值时不能小于 NFS 服务器与客户系统双方都支持的最大值，否则将会影响 NFS 的性能
timeo= <i>n</i>	指定初始超时值（以 1/10 秒为单位），默认值为 0.7 秒。从第一次超时后开始，新的超时值依次加倍，直至达到最大的超时值 60 秒，或者达到“retrans= <i>n</i> ”选项规定的重传次数
nfsvers= <i>n</i>	使用指定的 NFS 版本尝试安装远程目录或文件系统。通常，NFS 客户系统与服务器最终采用的 NFS 版本是双方系统均支持的高版本
vers= <i>n</i>	“vers”是“nfsvers”选项的另外一种表示方法，主要目的是与其他许多操作系统保持兼容
nolock	禁止使用 NFS 的封锁机制
port= <i>n</i>	指定连接 NFS 服务器时使用的端口号。如果未指定端口号或指定的端口号是 0（默认值），则查询并采用 NFS 服务器 portmapper 使用的端口号。如果无法使用 NFS 服务器 portmapper 的端口号，则使用标准的 NFS 端口号 2049
tcp	采用 TCP 传输协议实现 NFS 网络文件系统（如果 NFS 服务器与客户系统双方都支持 TCP 协议，则 tcp 是默认的安装选项；否则，udp 为默认的安装选项）
udp	采用 UDP 传输协议实现 NFS 网络文件系统

假定 NFS 服务器的设置已经就绪，NFS 客户系统可以在本地文件系统的目录层次结构中确定一个安装点或创建一个新的目录，然后利用 mount 命令，把 NFS 服务器提供的远程资源安装到自己的文件系统中。例如，要安装 iscas 服务器提供的共享资源，可以建立下列目录，然后把远程目录安装到新建的安装点。

```
$ sudo mkdir /docs
$ sudo mount -t nfs iscas:/home/gqxing/docs /docs
$ ls /docs
chap10.pdf chap14.pdf chap18.pdf chap21.pdf chap2.pdf chap6.pdf
chap11.pdf chap15.pdf chap19.pdf chap22.pdf chap3.pdf chap7.pdf
chap12.pdf chap16.pdf chap1.pdf chap23.pdf chap4.pdf chap8.pdf
chap13.pdf chap17.pdf chap20.pdf chap24.pdf chap5.pdf chap9.pdf
$
```

### 21.3.2 设置/etc/fstab 文件

上述安装方法属于临时性的安装，重新启动系统之后，原来安装的远程文件系统将不复存在。在许多情况下，用户可能需要永久性地安装远程共享目录。要一劳永逸地自动安装远程文件系统，需要借助于 Linux 系统的/etc/fstab 文件。按照 fstab 文件的格式要求，把上述命令经过适当地修改之后再加到文件中（参见第 17 章中有关 fstab 文件的介绍）。

以上述远程共享资源为例，为了自动安装远程目录或文件系统，可在 NFS 客户系统的 fstab 文件中增加下列一行内容（第一行仅作字段说明）。

```
# <file system>      <mount point> <type> <options> <dump> <pass>
iscas:/home/gqxing/docs /docs nfs rw,sync 0 0
```

然后，可用下列“mount -a”命令，安装新增的远程目录（每当重新启动系统时，系统都会

读取/etc/fstab 文件，执行“mount -a”命令，从而实现 NFS 远程共享资源的安装)。

```
$ sudo mount -a
$ ls /docs
chap10.pdf chap14.pdf chap18.pdf chap21.pdf chap2.pdf chap6.pdf
chap11.pdf chap15.pdf chap19.pdf chap22.pdf chap3.pdf chap7.pdf
chap12.pdf chap16.pdf chap1.pdf chap23.pdf chap4.pdf chap8.pdf
chap13.pdf chap17.pdf chap20.pdf chap24.pdf chap5.pdf chap9.pdf
$
```

注意，在 NFS 的所有版本中，目前最流行的仍是版本 2，因为大多数 NFS 客户系统均使用这一版本，但较新的 NFS 服务器一般都支持 NFS 版本 4。为了确保具有较好的兼容性与适应性，最好在 NFS 客户系统的/etc/fstab 文件中增加 nfsvers=2 安装选项，从而强制 NFS 服务器按照版本 2 公布共享目录或文件系统。

此后，每当启动 NFS 客户系统时，都会自动安装 iScsi 服务器提供的/home/gqxing/docs 共享目录。一旦在 NFS 客户系统中安装了远程目录或文件系统，用户就可以像使用本地文件系统一样地访问远程目录或文件系统。

如果不需要继续访问远程共享资源，可以利用下列 umount 命令卸载远程目录或文件系统。

```
$ sudo umount /docs
$
```

## 21.4 NFS 自动安装

远程文件系统的永久性安装存在一定的缺陷。由于每个 Linux 服务器中的/etc/fstab 文件是唯一的，因而需要编辑每一个客户系统中的 fstab 文件。因此，对 NFS 客户系统的管理与维护便是一个非常困难的问题。从资源利用方面来看，不管是否访问 NFS 服务器，这种永久性的安装对 NFS 客户系统与服务器都是一种 CPU 与网络带宽等资源的极大浪费。

利用 NFS 特定的映射文件与自动安装功能，可以绕过/etc/fstab 文件，直接实现 NFS 的远程安装，因而也就克服了上述的诸多问题。此外，还可以利用映射文件，指定预期的安装时间。超过预定的时间周期之后，可自动卸载已经安装的 NFS 远程资源，删除安装点。当再次收到 NFS 文件访问请求时，automount 守护进程将会按照映射文件的定义，把远程资源重新安装到指定的安装点，而安装点是由 automount 守护进程根据需要自动创建的。

为了实现 NFS 的远程共享资源自动安装功能，还需要利用 apt-get、aptitude 和 synaptic 等软件维护工具，安装 autofs 软件包。示例如下。

```
$ sudo apt-get install autofs
```

### 21.4.1 主映射文件

在 NFS 的自动安装机制中，除了前述的 NFS 守护进程与 automount 之外，NFS 服务器几乎可以不知道，也不关心 NFS 客户系统究竟是采用 mount，还是采用 automount 安装共享目录或文件系统。而在 NFS 客户端，除了前述的 NFS 守护进程与 automount 之外，还要准备下列 3 个映射文件：

- 主映射文件 auto.master；
- 间接映射文件（如 auto.home，文件名可由主映射文件指定）；
- 直接映射文件（如 auto.direct，文件名可由主映射文件指定）。

主映射文件 auto.master 是一个总的映射配置，其中定义了另外两个映射文件，以便进一步说



明安装点与远程资源之间的映射关系，使 automount 能够根据这 3 个文件的定义按照需要自动安装远程资源。主映射文件的语法格式如下。

```
mount_point map-file [mount_options]
```

其中，mount\_point 用于定义准备安装的目录，以便在此目录中安装 NFS 服务器的远程共享资源。对于直接映射而言，这个字段为“/-”，而实际的安装点是直接映射文件中第一个字段的值。对于间接映射而言，这个字段只是安装点的根目录，实际的安装点应是这个字段值与间接映射文件第一个字段值的组合。第二个字段的 map-file 用于指定相应的直接映射文件或间接映射文件。第三个字段的 mount\_options 是 mount 或 automount 命令可用的安装选项。

在下列 auto.master 示例文件中，/home 目录是 NFS 客户系统中的一个安装点。最终的安装点是 /home 与间接映射文件 /etc/auto.home 中第一个字段所列目录的组合。最后的安装选项 “--timeout=300” 意味着每次安装的空闲持续时间不能超过 300 秒（5 分钟），超过此限，automount 将会自动卸载相应的远程目录或文件系统。如果没有特别指定，这个安装选项将会作为 /etc/auto.home 文件中所有安装项的默认值。示例如下。

```
$ cat /etc/auto.master
...
/home   /etc/auto.home      --timeout=300
/-     /etc/auto.direct
$
```

## 21.4.2 直接映射文件

直接映射文件用于定义安装点，以及从 NFS 服务器上安装的远程目录或文件系统，其语法格式如下。

```
mount_point [mount_options] remote_resource
```

其中，mount\_point 是 NFS 客户系统上的一个目录，用于安装 NFS 服务器中的远程共享资源。mount\_options 是 mount 或 automount 命令可用的安装选项。注意，mount\_options 字段是可选的，可以不存在。如果存在，第一个安装选项前应加一个减号“-”字符，多个选项之间需加逗号“，”分隔符。remote\_resource 是 NFS 服务器上的一个远程共享资源，其表示形式为“*server.pathname*”。

在下面的例子中，/docs 是 NFS 客户系统上的一个目录，用于安装 NFS 服务器 iscas 中的远程共享资源 /home/gqxing/docs。

```
$ cat /etc/auto.direct
...
/docs  -rw, sync    iscas:/home/gqxing/docs
$
```

直接映射文件用于定义相对独立的远程安装要求，任何不规则的安装点或远程共享资源均应采用直接映射文件的配置方式。

## 21.4.3 间接映射文件

间接映射文件用于定义主映射文件给出的同一安装点下的子目录或相对路径，以及从 NFS 服务器上安装的远程目录或文件系统，其语法格式如下。

```
mount_point [mount_options] remote_resource
```

其中，mount\_point 是 NFS 客户系统上的一个相对目录，mount\_options 是 mount 或 automount 命令可用的安装选项。同样，mount\_options 字段是可选的，可以不存在。如果存在，第一个安装选项前应加一个减号“-”，多个选项之间需加逗号“，”分隔符。remote\_resource 是 NFS 服务器上的一个远程共享资源，其表示形式为“*server.pathname*”。

在下面的例子中，cathy、gqxing 和 guest 是 3 个子目录，组合 auto.master 文件定义的共同根目录/home，可以构成 3 个完整的安装点，分别用于安装 NFS 服务器 iscas 中的 3 个用户主目录。

```
$ cat /etc/auto.home
.....
cathy    iscas:/home/cathy
gqxing   iscas:/home/gqxing
guest    iscas:/home/guest
$
```

在间接映射文件中，mount\_point 与 remote\_resource 字段中间可以使用星号“\*”和“&”通配符。其中，星号“\*”表示所有可能的目录，“&”表示使用相应的目录替换“&”字符位置。据此，可以利用“&”通配符，把/etc/auto.home 间接映射文件改写如下。

```
$ cat /etc/auto.home
.....
cathy    iscas:/home/&
gqxing   iscas:/home/&
guest    iscas:/home/&
$
```

在上述间接映射文件中，第一个安装点（相对路径名）是 cathy，因而需要把“&”解释为 cathy，这意味着远程资源为 iscas:/home/cathy。同样，当安装点（相对路径名）为 gqxing 和 guest 时，也需要把“&”分别解释为 gqxing 和 guest，表示远程资源为 iscas:/home/gqxing 和 iscas:/home/guest。

如果配合使用星号“\*”与“&”通配符，还可以进一步简化，最终可以把/etc/auto.home 间接映射文件改写如下。

```
$ cat /etc/auto.home
.....
*       iscas:/home/&
$
```

在上述间接映射文件中，第一个字段中的星号“\*”表示主映射文件指定目录（/home）中所有可能的子目录。如果 NFS 客户系统引用的是/home/cathy，则“&”应解释作 cathy，因而需要安装的远程资源就是 iscas:/home/cathy；如果 NFS 客户系统访问的是/home/gqxing，需要安装的远程资源就是 iscas:/home/gqxing，如此等等。

在完成上述配置，重新启动 NFS 服务器和客户系统的相应守护进程之后，automount 守护进程将会根据用户对远程资源的访问请求，自动安装或卸载远程目录或文件系统。例如，如果 NFS 客户系统中的用户想要查阅/home/cathy/docs/readme.doc 文件，automount 将会检查其中引用的目录是否涉及/etc/auto.master 文件中列举的安装点，以及/home/cathy 目录是否由间接映射文件 auto.home 控制。如果确乎如此，automount 将会拦截用户的访问请求，检查/home/cathy 目录是否已经安装；如果尚未安装，则自动安装。至此，用户即可访问指定的文件了。

注意，如果修改了映射文件，需要重新启动 automount，才能使修改后的配置立即生效。为此，可以使用下列命令。

```
$ sudo /etc/init.d/autofs restart
```

## 21.5 NFS 故障修复

### 21.5.1 基本工具

NFS 网络文件系统的安装、运行和设置相对比较简单。在同一个网络中，只要相应的守护进



程能够正常地运行，一般都不会存在问题。比较常见的问题是在编辑或修改/etc/fstab 与/etc(exports 文件后忘记重启系统或 NFS 守护进程，没有及时公布新增的共享资源等。另外一个常见的问题是访问权限，尤其是 NFS 服务器对客户系统超级用户的访问限制。表 21-3 给出了部分常见的出错信息。此外，也许还需要查阅/var/log/messages 文件，以便找出更多的线索和故障原因。

表 21-3 常见的部分出错信息及其简单说明

NFS 出错信息	简 单 说 明
Too many levels of remote in path	试图安装一个已经安装的文件系统
Permission denied	拒绝用户访问。一个常见的原因是 NFS 客户系统中的用户 root 仍以超级用户的身份访问 NFS 服务器，而服务器在利用/etc(exports 文件公布自己的共享资源时通常采用默认的 root_squash 选项，把超级用户 root 的 UID/GID (0) 映射为 NFS 匿名用户 (nobody) 的 UID/GID (65 534)。另外一个原因是 NFS 服务器中不存在客户系统中的同名用户
No such host	NFS 服务器的名字拼写错误，或者提供的 IP 地址不准确
No such file or directory	文件或目录名的拼写可能有误，或者根本就不存在
NFS server is not responding	NFS 服务器正在超负荷运行，无法对 NFS 访问请求做出及时的响应，或者 NFS 服务器已经关机
Stale file handle	NFS 客户系统先前访问的服务器文件在关闭之前已经被删除、移动位置或重新命名
Fake hostname	NFS 客户系统的 DNS 正向记录与反向记录均不存在

在跟踪与处理 NFS 的问题时，通常可从服务器、客户系统和网络三个方面入手考虑，分析出出现故障的原因。在任何情况下，NFS 服务器都必须运行前述的各种守护进程，才能保证 NFS 客户系统能够成功地安装与访问远程共享目录或文件系统。

此外，nfsstat 命令提供的统计信息也有助于用户了解 NFS 是否存在问题，找出 NFS 的有关问题。nfsstat 命令主要用于查询 NFS 服务器与客户系统的各种统计数据，以了解 NFS 的共享特性、运行状态以及总体性能，其语法格式如下。

**nfsstat [-234cmnrs] [-o options]**

其中，“-s”选项用于获取 NFS 服务器方面的统计信息，“-c”选项用于获取 NFS 客户系统方面的统计信息，“-m”选项用于显示已经安装的每个 NFS 远程目录或文件系统的统计信息。“-2”、“-3”或“-4”选项表示仅显示指定版本的统计信息。通常，nfsstat 命令将会给出 NFS 所有版本的统计信息，参见表 21-4。

表 21-4 nfsstat 命令的部分常用选项及说明

选 项	简 单 说 明
-2	仅显示 NFS 版本 2 的统计信息
-3	仅显示 NFS 版本 3 的统计信息
-4	仅显示 NFS 版本 4 的统计信息
-c	显示 NFS 客户端的 NFS 与 RPC 统计信息
-m	显示每个已安装的 NFS 文件系统的统计信息。统计信息包括服务器的名字与目录、安装标志、读写缓冲区的大小、重传计数及超时值等。“-m”选项只能单独使用。如果“-m”选项存在，nfsstat 将会忽略其他选项
-n	显示与 NFS 有关的统计信息
-r	显示与 RPC 有关的统计信息
-s	显示 NFS 服务器端的 NFS 与 RPC 统计信息

例如，如果客户系统采用 NFS V3 安装了远程服务器 iscas 的共享目录/home/gqxing/docs，“nfsstat -m”命令的输出信息如下。

```
$ nfsstat -m
/docs from iscas:/home/gqxing/docs
Flags: rw,relatime,vers=3,rsize=65536,wsize=65536,hard,nointr,proto=tcp,timeo=6
00,retrans=2,sec=sys,addr=169.254.78.100
$
```

下面的输出信息是取自 NFS 服务器统计数据的一个例子。其中，第一组数据主要与 RPC 协议层的服务有关，其他均为各种 NFS 调用的统计数据。

```
$ nfsstat -s
Server rpc stats:
calls    badcalls   bdauth   badclnt   xdrcall
106      0          0         0         0

Server nfs v3:
null      getattr   setattr   lookup   access   readlink
4        4% 40      44% 4       4% 10     11% 5      5% 0      0%
read      write     create    mkdir    symlink  mknod
0        0% 3       3% 4       4% 0       0% 0      0% 0      0%
remove    rmdir    rename   link     readdir  readdirplus
0        0% 0       0% 0       0% 0      0% 0      0% 4      4%
fsstat    fsinfo   pathconf commit
0        0% 9       10% 3      3% 3      3%
$
```

在上述 3 组统计数据中，通常需要关注 badcalls、xdrcall、getattr、retrans、read、write 及 commit 等字段，据以考察 NFS 是否存在问题。如果 NFS 服务器 commit 字段的数值较大，需要检查 NFS 客户系统是否具有足够的内存。当 NFS 客户系统没有可用的资源时，NFS 服务器 commit 操作的数量将会不断地增长。表 21-5 给出了需要重点考察的几个输出字段。

表 21-5

nfsstat 输出信息的重要字段

输出字段	阈值	简单说明
badcalls	> 0	NFS 服务器的 RPC 服务拒绝接收或处理的 RPC 调用的统计计数。badcalls 字段值反映的是 NFS 客户系统 RPC 调用失败的次数，如果数值较大或突然增大，表明网络连接本身存在问题，NFS 客户系统的用户试图以超级用户的身份访问采用 root_squash 选项安装的远程资源，以及用户身份认证方面的配置不当，等等
xdrcall	> 0	其头信息无法解析的 RPC 调用的数量
getattr	> 50%	除了文件数据，NFS 服务器的缓冲机制也存储文件属性。getattr 字段值反映的是没有从缓冲区中读取文件属性的百分比。通常，造成这一结果的原因是采用了 NFS 的“noac”安装选项，致使 NFS 服务器没有实行文件属性缓存处理
retrans	> 5%	由于得不到 NFS 服务器的及时响应而超时，导致客户系统不得不重新传输各种服务请求的百分比。可能的原因是 NFS 服务器的负载过重，或者网络通信环境不佳
writes	> 10%	由于缓存机制的问题导致数据写入性能降低的百分比。检查“noac”和“wsize”安装选项的设置是否合理
read	> 10%	由于缓存机制的问题导致读取数据性能降低的百分比。检查“noac”和“rsize”安装选项的设置是否合理

在所有的 NFS 安装操作中，“intr”都是一个默认的选项。如果程序挂起，且显示一个“server not responding”的出错信息，可以使用键盘的中断键“Ctrl-C”或 kill 命令终止程序的执行。

同一个应用程序，当因为网络或 NFS 服务器存在问题而失败时，访问硬性和软性安装的远程文件系统，其结局是不同的。

对于采用“hard”安装选项的远程目录或文件系统，NFS 客户系统的内核程序将会尝试重发没有得到回应的请求，直至 NFS 服务器最终给予响应。对于采用“soft”安装选项的远程目录或文件系统，客户系统的系统调用将会在尝试一次之后，不管成功与否，返回一个出错信息。由于这种错误可能引起应用程序无法预料的后果，如数据错误或损坏等，因此，应尽量避免使用“soft”安装选项。



当采用“hard”选项安装远程目录或文件系统时，如果服务器没有给予响应，访问文件系统的应用程序可能会被挂起。在这种情况下，NFS 客户系统的控制终端上将会显示下列信息。

*NFS server hostname not responding still trying*

当采用“soft”选项安装远程目录或文件系统，访问文件系统的应用程序没有得到服务器的响应时，控制终端上将会出现下列信息。

*NFS operation failed for server hostname: error # (error\_message)*

注意，由于容易出错，建议一般不要采用“soft”选项以读写方式安装远程文件系统，也不要使用“soft”选项安装其中含有可执行程序且需要从中执行程序的远程文件系统。有关“soft”安装选项的说明，参见表 21-2。

### 21.5.2 其他注意事项

在使用 NFS 网络文件系统时，下列说明可供参考。

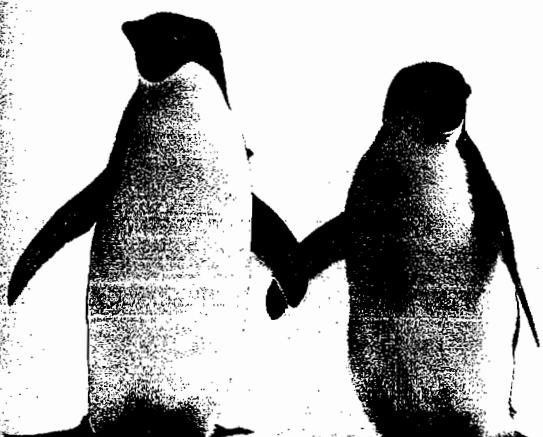
- 如果 NFS 服务器运行不稳定，出现故障或关机，NFS 客户系统可能需要等待较长时间才能返回。针对此情况，可以适当地考虑在 NFS 客户系统的/etc/fstab 文件中使用 soft 安装选项，一旦出现超时，NFS 将会报告一个 I/O 错误信息。为了减少 NFS 挂起的风险，还可以采用下列预防措施：
  - 在一个相对可靠的网络上运行 NFS；
  - 尽可能地总是使用 sync 安装选项；
  - 不要在命令检索路径 PATH 变量中增加 NFS 远程目录，以免 Shell 因等待 NFS 远程目录就绪而无法继续检索命令，尤其不要把 NFS 远程目录加到 PATH 变量的前部。
- NFS 不允许公布嵌套的共享目录，即不允许公布位于已公布目录下的子目录，除非两者位于不同的磁盘分区。
- 通常，NFS 不允许 NFS 客户系统中的 root 用户在 NFS 服务器上也拥有超级用户的特权。在/etc(exports 文件中采用默认的 root\_squash 安装选项，有助于减少客户系统的 root 用户以超级用户的身份访问 NFS 服务器造成的潜在风险。如果确实需要，可以在/etc(exports 文件中增加 no\_root\_squash 安装选项。
- NFS 本身并不支持以用户为基础的安全访问限制。如果 NFS 客户系统与服务器中存在同名的用户，如 nfsuser，NFS 客户系统中的 nfsuser 用户也能够访问 NFS 服务器中同名用户名下的所有文件。因此，最好利用/etc(exports 文件限定只有信任的系统或网络才能访问 NFS 服务器。当然，也可以使用防火墙保护 NFS 服务器。注意，NFS 客户系统与服务器之间建立的主要通信控制信道通常使用 111 号 TCP 端口，但经常会采用经过双方协商后随机选择的 TCP 端口传输数据。

# 第22章

## Samba 资源共享

为了实现 Linux 与 Windows 系统异构网络环境的资源共享，Linux 系统提供了一套 Samba 组件。利用 Samba 组件，可以把 Linux 系统配置成一个 Windows 服务器（或客户端），使 Windows 用户能够共享 Linux 系统中的目录、文件与打印机等资源。本章主要讨论怎样配置 Samba 服务器，实现 Windows 与 Linux 系统的资源共享，以便 Windows 客户能够注册到 Linux 系统，访问自己的主目录或其他共享资源，在资源浏览器中直接操作文件，或者把 Linux 系统的共享资源作为一个逻辑驱动器，实现文件的共享与交换；连接 Linux 系统中的共享打印机，提交打印作业。其内容主要包括：

- 安装 Samba 服务器；
- smb.conf 配置文件；
- 快速设置 Samba 服务器；
- Samba 运行环境测试；
- 访问共享资源。





Windows 系统采用 TCP/IP 作为底层传输协议，发送 NetBIOS 请求，而在 NetBIOS 的上层，则采用 SMB（Server Message Block）协议作为高级接口，实现 Windows 网络环境中的文件与打印机等资源共享。新的 CIFS（Common Internet File System）是一个增强版的 SMB 协议，使 Windows 用户能够通过 Internet 实现更大范围的网络资源共享。

除了 Linux 系统本身的功能，Samba 服务器通常会扮演一个 Windows 域控制器或普通服务器的角色，提供文件与打印共享。Samba 服务器与 Linux 系统共享相同的用户名与密码信息，因此，Windows 用户可以使用 Linux 系统的用户名及密码注册到 Samba 服务器，访问 Linux 系统用户主目录中的文件。此外，出于安全性的考虑，也可以在 Samba 服务器与 Linux 系统中采用不同的密码。

Samba 服务器采用 /etc/samba/smb.conf 作为配置文件。在启动 Samba 服务器或 Windows 用户请求访问时，Samba 服务器的守护进程将会读取 smb.conf 配置文件，确定自己的工作模式。在确定 Samba 服务器究竟提供什么共享目录与打印服务时，最主要的方法是修改其配置文件。故可根据实际的需求，使用文本编辑器直接编辑 /etc/samba/smb.conf 配置文件，或者采用基于浏览器的图形界面管理工具 SWAT 设置 Samba 服务器的 smb.conf 配置文件。

在配置 Samba 服务器时，通常需要从下列几个方面考虑：

- 确定 Samba 服务器的角色定位，如用作域控制器，还是普通的 Windows 服务器；
- 提供什么资源供 Windows 用户共享，如提供用户主目录、普通目录抑或是打印机共享等；
- 提供什么验证机制，确保 Windows 用户能够合法地访问 Samba 服务器。

安装 Samba 软件之后，Windows 与 Linux 系统中的用户能够透明地访问彼此的文件系统，尤其是，Windows 用户能够像访问 Windows 环境中的共享文件与打印机等网络资源一样，透明地访问 Linux 系统中的文件系统和打印机等共享资源，而不会意识到他们访问的是 Linux 系统。利用 Samba 服务器软件，Windows 用户能够充分利用 Linux 系统的速度、主机处理能力及海量存储空间。

注意，如果仅想在 Linux 系统环境中访问 Windows 文件系统，达到文件交换的目的，则不必使用 Samba 软件，直接安装 Windows 系统的 FAT 或 NTFS 文件系统即可。

Samba 服务器主要由两个守护进程（smbd 和 nmbd）组成，每个守护进程均可从命令行中单独启动，也可以通过 /etc/init.d/smb 脚本自动启动。smbd 守护进程的主要功能是为 Windows 用户提供文件与打印共享服务，nmbd 守护进程是一个 NetBIOS 名字服务器，能够把 Windows SMB 请求信息中的 NetBIOS 名字映射为 Linux 系统中使用的 IP 地址。

## 22.1 安装 Samba 服务器

Samba 套件分为服务器、客户端、文档及图形界面管理工具等多个软件包。其中，客户端软件使 Linux 系统中的用户也能与 Windows 服务器联网通信；服务器软件能够使 Linux 系统成为一个 SMB/CIFS 服务器，提供 SMB 接口，实现 Windows 环境中的网络文件与打印机共享；samba-doc 或 samba-doc-pdf 软件包分别提供不同格式的 Samba 文档；system-config-samba 和 swat 等软件包提供图形界面的维护工具。

在 Ubuntu Linux 系统中，Samba 系列的软件包需要单独安装。用户可根据自己的实际需求，选择安装下列 Samba 软件包：

- samba（服务器）；
- samba-common（通用实用程序）；
- smbclient（客户端，不需要单独安装）；
- samba-doc（HTML 版文档）；
- samba-doc-pdf（PDF 版文档）；
- system-config-samba（GNOME 管理工具）；
- swat（基于浏览器的管理工具）；
- smbfs（SMB 文件系统安装与卸载工具）。

安装 Samba 服务器时，可以使用 apt-get、aptitude 或 synaptic 等命令。示例如下。

```
$ sudo apt-get install samba samba-doc system-config-samba swat smbfs
```

在运行上述命令时，apt-get 将会自动选择安装底层支持软件包 samba-common 及客户端软件包 smbclient。

注意，在 Ubuntu 8.04 Linux 系统中安装 Samba 服务器软件包时，除了在 /etc/init.d 目录中增加了一个 samba 脚本外，并没有在 rcN.d 目录中增加必要的链接文件，故即使重新启动系统，系统也不会自动启动 Samba 服务器的守护进程。为了能够自动启动 Samba 服务器，还需要把 /etc/init.d/samba 启动脚本链接到 /etc/rc2.d 和 /etc/rc0.d 目录中，具体步骤如下。

```
$ cd /etc/init.d
$ sudo ln -s samba ../rc2.d/S90samba
$ sudo ln -s samba ../rc0.d/K10samba
$
```

## 22.2 smb.conf 配置文件

Samba 服务器的默认配置文件为 /etc/samba/smb.conf，其中提供了运行时的配置信息。目前，Samba 服务器配置文件的初始设置必须采用传统的 Linux 方法，即利用文本编辑器直接编辑配置文件。

事实上，编辑 /etc/samba/smb.conf 文件，配置 Samba 服务器是相当简单的。当然，前提是必须对其中的配置变量有比较清晰的理解。直接编辑 /etc/samba/smb.conf 文件有助于用户了解 Samba 服务器配置信息的存储位置，以及配置文件的组织结构。

在完成初始配置之后，可以选用 Samba 服务器管理工具 SWAT（Samba Web Administration Tool）或 webmin 等，利用图形界面设置与维护 Samba 服务器。在开始介绍配置文件及其配置变量之前，首先简单说明怎样调用 SWAT 配置工具，再分类解释部分重要的配置变量。

SWAT 是一个基于浏览器的 Samba 配置工具，便于用户设置 smb.conf 配置文件，而无需关心配置文件的语法格式。SWAT 的设置界面实际上是一个根据 smb.conf 配置文件各个组成部分转换成的表格，只需根据表格填写变量值即可。为便于用户填写，每个配置变量都提供了相应的帮助信息。要访问 SWAT，可在浏览器中输入 “<http://127.0.0.1:901>”，在 SWAT 注册界面输入用户名（如超级用户 root）与密码，最终进入 SWAT 主界面，如图 22-1 所示。

注意，如果使用普通用户而非超级用户 root 注册，只能查询 Samba 服务器的配置与运行状态，了解 Samba 服务器提供的共享资源，而无权设置任何配置变量。因此，如果确实想使用 SWAT 维护 Samba 服务器，需要启用超级用户账户（参见第 13 章）。此外还应注意，在利用 SWAT 修改 smb.conf



配置文件之前，建议复制一个 smb.conf 配置文件副本，因为 SWAT 有可能会删除其中的注释信息。

与其他服务器软件不同的是，修改 smb.conf 配置文件之后，通常不需要重启 Samba 守护进程 smbd，因为每当收到 Windows 客户的访问请求时，smbd 都会读取配置文件。

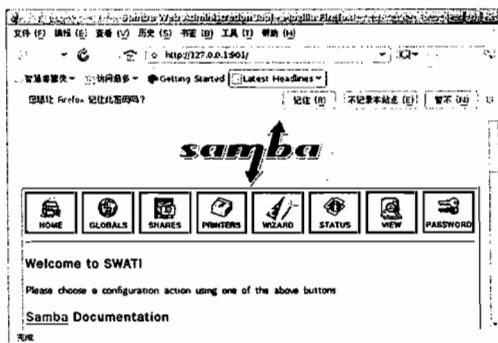


图 22-1 SWAT 主界面

### 22.2.1 smb.conf 配置文件概述

不管是手工直接编辑，还是利用 SWAT 工具间接修改 smb.conf 配置文件，都需要了解 Samba 服务器支持的配置变量。Samba 采用大量的配置变量，定制 Samba 服务器的功能。对于管理人员而言，其中常用的只有一小部分，大部分配置变量均可采用其默认设置。

Samba 服务器的/etc/samba/smb.conf 配置文件分为若干节（section），每节采用一个节名（前后加方括号）作为标识，如[global]、[homes]和[printers]等。一个 smb.conf 配置文件通常至少应包含[global]、[homes]及[printers]这 3 节，每一节的用途如表 22-1 所示。

表 22-1

smb.conf 文件格式

节	简单说明
[global]	用于设置 Samba 服务器的全局配置变量，其作用范围适用于 Samba 服务器提供的所有共享资源。在其他节的共享资源配置没有明确规定的情况下，还可以用作相应配置变量的默认值
[homes]	主要用于定义用户主目录，以便能够在 Windows 系统中连接、访问自己在 Linux 系统中的主目录。注意，[homes] 节定义的共享资源，其名字是实际的用户名，而非 homes，这与其他普通共享资源不同。在配置[homes]节时，可以使用与普通共享资源节相同的配置变量。此外，如果没有定义 path 配置变量，默认的共享资源是用户的主目录。 当 Windows 用户请求连接 Samba 服务器时，Samba 服务器将会扫描其配置文件中定义的所有共享资源。如果发现了匹配的资源名，则将其用作主目录；如果未发现匹配的资源，则使用请求的资源名作为用户名，检索本地系统中的密码文件。如果用户存在，且提供的密码是正确的，则参照[homes]节的设置，创建一个共享资源。在定义 path 变量时，可以使用变量替换，如%S 等
[printers]	用于配置打印机，为 Windows 客户提供集中打印服务。如果定义了共享打印机，Windows 用户能够连接到本地系统 printcap 文件指定的任何打印机。 当 Windows 用户请求连接 Samba 服务器时，Samba 服务器将会扫描其配置文件中定义的所有共享打印资源。如果发现了匹配的资源名，则将其用作共享打印机；如果未发现匹配的资源，但[homes]节存在，则使用请求的资源名作为用户名，检索本地系统中的密码文件。如果用户存在，且提供的密码是正确的，则参照[homes]节的设置，创建一个共享资源；否则，使用请求的资源名作为打印机名，检索适当的 printcap 文件，验证请求的资源是否为一个有效的打印机资源名。如果发现匹配者，则参照[printers]节的设置，创建一个新的共享打印资源。同样，[printers]节定义的共享资源，其名字是实际的打印机名，而非 printers。[printers]节定义的共享资源必须是可打印的，否则，Samba 服务器将会拒绝加载配置文件。通常，共享打印资源的路径名必须是任何用户均可读写的缓存目录

在 smb.conf 配置文件中，除了[global]节，每一节均可用于设置一个共享资源。节的名字就是共享资源的名字（[homes]和[printers]节除外），节中的配置变量用于定义共享资源的属性。共享资

源或者是一个目录，为 Windows 客户提供文件共享服务，或者是一个打印机，为 Windows 客户提供打印服务。通常，大多数共享资源都是由目录及其相关的用户访问权限等属性组成的。

在 smb.conf 配置文件中，管理人员需要配置的主要有 [global]、[homes] 以及 [printers] 节。必要时，还可以增加自己定义的共享资源节，以提供其他共享资源，如共享 CD/DVD 等。

在设置 smb.conf 配置文件时，许多字符串类型的配置变量值均可嵌入一些特殊变量，以便在连接过程中进行变量替换。例如，如果访问共享资源的用户是 gqxing，配置变量“path = /home/%u”可以解释为 path = /home/gqxing。下面是 smb.conf 配置文件支持的部分常用的特殊变量。

- %U —— 当前会话的用户名；
- %G —— 当前用户的（主）用户组名；
- %h —— Samba 服务器的 Internet 主机名；
- %m —— Windows 客户系统的 NetBIOS 名字；
- %L —— Samba 服务器的 NetBIOS 名字；
- %M —— Windows 客户系统的 Internet 主机名；
- %d —— 当前服务器进程的 PID；
- %I —— Windows 客户系统的 IP 地址；
- %i —— Windows 客户系统连接的本地网络接口的 IP 地址；
- %T —— 当前的日期与时间；
- %D —— 当前用户所在的域名或工作组名。

下列特殊变量仅适用于连接建立之后才能涉及的配置变量。

- %S —— 当前共享资源的名字（如果存在）；
- %P —— 当前共享资源的根目录（如果存在）；
- %u —— 当前共享资源的用户名（如果存在）；
- %g —— %u 用户的（主）用户组；
- %H —— %u 用户的主目录。

## 22.2.2 Global 节

global 节用于定义 Samba 服务器的全局或通用配置变量，这些配置变量也可用作其他节中没有明确定义的配置变量的默认值。通常，Samba 服务器的网络标识与认证等全局配置变量均位于 /etc/samba/smb.conf 配置文件的 [global] 节中。表 22-2 给出了设置 Samba 服务器时需要用到的部分常用的配置变量。有关 smb.conf 文件的完整说明，详见 smb.conf 手册页。

表 22-2

[Global]节常用的部分配置变量

配置变量	默认值	简单说明
default service	无	指定一个可用的默认共享资源，如果用户请求的共享资源不存在，可以访问这个共享资源。如果未指定默认的共享资源，当用户请求访问的共享资源不存在时，将会返回一个错误信息。通常，默认的共享资源应当是一个与“guest ok”与“read only”变量设置有关的资源
dns proxy	yes	指定是否需要把 Samba 服务器配置成一个 DNS 代理服务器。如果这个配置变量的设置为 yes，作为一个 WINS 服务器，当发现某个 NetBIOS 名尚未注册时，Samba 服务器应当以 DNS 客户机的身份，把 NetBIOS 名完全看作一个 DNS 主机名，查询 DNS 服务器，完成名字解析。注意，NetBIOS 名字的最大长度为 15 个字符，因此，DNS 域名（或别名）最多也不能超过 15 个字符



续表

配置变量	默认值	简单说明
domain logons	no	表示是否把 Samba 服务器配置成一个主域控制器 (PDC)，为 Windows 客户系统提供网络注册 (netlogon) 服务
domain master	auto	用于控制 Samba 服务器是否能够承担工作组网络中的主域控制器的角色。如果把“domain logons”设置为 yes，则蕴含着把“domain master”也设置为 yes，同时表示把 Samba 服务器设置为主域控制器。如果“domain logons”采用默认值 no，通常不会启用“domain master”。但是，如果把“domain master”设为 no，则意味着把 Samba 服务器设为 BDC
encrypt passwords	yes	用于控制是否需要与 Windows 客户系统协商使用加密的密码
guest account	nobody	用于设定一个 guest 用户名，以便能够访问“guest ok”配置变量指定的共享资源。任何 Windows 用户都能够利用 guest 用户拥有的权限连接到其可以访问的共享资源。在 Linux 系统中，指定的 guest 用户必须存在于 /etc/passwd 文件中，但并不要求一定能够用之注册，如 nobody 或 ftp 等
hostname lookups	no	指定 Samba 服务器使用主机名，还是使用 IP 地址检索。例如，在检测“hosts deny”与“hosts allow”配置变量的设置时，Samba 服务器当前就是使用主机名检索的
interfaces	无	<p>利用这个配置变量，可以强制改变 Samba 服务器使用的默认网络接口。通常，Samba 服务器会查询系统内核，获取所有活动的网络接口，使用具有广播功能的任何网络接口 (127.0.0.1 除外)。设置这个配置变量时，可以采用下列任何形式之一，指定一个或多个网络接口 (中间加空格分隔)：</p> <ul style="list-style-type: none"> <li>□ 网络接口名 (如 eth0。在指定网络接口名时，可以采用通配符星号 “*” 同时定义多个同类网络接口，例如，“eth*” 表示所有以“eth”为起始字符串的以太网络接口)；</li> <li>□ 主机名；</li> <li>□ IP 地址；</li> <li>□ IP 地址/mask (其中，子网掩码 mask 可以是表示位长的整数，如 24 对应于 C 类网。也可以采用标准的 IP 地址形式，下同)；</li> <li>□ broadcast/mask</li> </ul>
keepalive	300	这个配置变量的值是一个以秒为单位的整数，表示发送 keepalive 分组数据的间隔时间，以便 Samba 服务器能够了解 Windows 客户系统是否仍然联机。如果这个配置变量的值为 0，意味着不发送 keepalive 分组数据
log file	无	用于另行指定 Samba 服务器的日志文件 (也称作调试日志文件)，如“log file = /var/log/samba/log.%m”
log level	无	设置这个配置变量能够决定是否允许记录多种调试级别的日志信息。其默认值为 0，或者以 smbd 命令行“-d”选项指定的调试级别为准
max connections	0	用于指定一个共享资源并发连接的最大数量限制，默认值 0 意味着没有限制。超过此限，新的连接请求将会遭到拒绝
max disk size	0	用于设置共享资源可用存储空间的上限，默认值 0 意味着没有限制。注意，这个限制并不影响用户能够存储的实际数据数量，只是当 Windows 用户查询空闲磁盘空间或整个磁盘空间时，给出的结果将会受限于这个配置变量指定的数量
max log size	5 000 (实为 10 00)	用于设置日志文件以 KB 为单位的最大容量限制。Samba 服务器将会周期地检测日志文件的大小。如果超过此限，将会在日志文件名后面附加一个.old 后缀
max open files	10 000	这个配置变量的值用于限制一个 smbd 守护进程支持的，每个 Windows 用户在任何时刻都能够打开的最大文件数量。通常无需修改这个配置变量
name resolve order	lmhosts host wins broadcast	<p>用于确定在解析主机名与 IP 地址时应选用的名字服务及其选用顺序，其主要目的是控制怎样执行 NetBIOS 名字解析。这个配置变量的有效值是 lmhosts、host、wins 和 broadcast，其作用简述如下：</p> <ul style="list-style-type: none"> <li>□ lmhosts —— 从 Samba 的 lmhosts 文件中检索 IP 地址；</li> <li>□ host —— 利用/etc/hosts 文件、NIS 或 DNS 实现标准的主机名与 IP 地址解析；</li> <li>□ wins —— 从“wins server”配置变量指定的 WINS 服务器查询系统名与 IP 地址，如果未设置 WINS 服务器，则忽略此检索方法；</li> <li>□ broadcast —— 在已知的每个本地网络接口 (参见 interfaces 配置变量) 中，通过广播方式实现主机名的解析，这是最不可靠的一种主机名解析方法。</li> </ul> <p>默认的变量设置表示先利用本地的 lmhosts 文件，然后依次利用 hosts、wins 和 broadcast 等方法解析主机名。当采用“security ads”安全模式时，Samba 服务器通常应采用“name resolve order = wins broadcast”主机名解析顺序</p>

续表

配置变量	默认值	简单说明
netbios name	'uname -n'	用于设置 Samba 服务器认可的 NetBIOS 名字。通常，这个名字等同于主机名（规范主机域名的第一部分）
null passwords	no	表示是否允许 Windows 用户使用没有密码的用户账号访问 Samba 服务器
obey pam restrictions	no	用于控制 Samba 是否应当服从 PAM 的用户账户与会话管理指令。默认的做法是仅在明文认证时采用 PAM 认证，且忽略任何账户或会话管理。注意，在“encrypt passwords = yes”的情况下，Samba 总是忽略 PAM 认证，其原因是 PAM 模块不支持 SMB 密码加密需要的提示/应答式认证机制
os level	20	用于设置 Samba 服务器参与 PDC 竞争时使用的优先级别。例如，当与其他 Windows 服务器竞争时，如果把这个配置变量设为一个较大的数值（如 65），将会提高 Samba 服务器获胜的机会
passdb backend	smbpasswd (实为 passdb.tdb)	表示采用哪一种方式存储用户、用户组、密码及 Windows 客户系统等信息。这个配置变量的值可以分为两部分，即存储方式与文件，中间加一个冒号“:”分隔符，如 tdbsam:/var/lib/samba/passdb.tdb（后面的文件路径名可省略）。可用的存储方式如下： <input type="checkbox"/> smbpasswd —— 默认的用户密码存储方式（参见 smbpasswd (5) 手册页）； <input type="checkbox"/> tdbsam —— TDB (Trivial DataBase) 用户密码存储方式； <input type="checkbox"/> ldapsam —— LDAP 用户密码存储方式
pid directory	/var/run	指定 Samba 服务器守护进程 PID 文件所在的目录位置
preferred master	auto	用于控制 Samba 服务器是否需要提升为 Windows 网络中的主域控制器，如果设为 yes，Samba 服务器启动后将会展开竞争。在此情况下，建议把“domain master”配置变量同时设置为 yes，确保 Samba 服务器能够成为主域控制器
printcap name	无	用于指定 Samba 服务器使用的 printcap 文件。当使用 CUPS 打印服务（如 Ubuntu Linux 系统）时，这个配置变量应设置为“printcap name = cups”。此外，还需要在[global]节中设置“printing = cups”
printing	无	这个配置变量用于控制如何解释服务器的打印机状态信息。当前，Samba 服务器仅支持 9 种打印类型：BSD、AIX、LPRNG、PLP、SYSV、HPUX、QNX、SOFTQ 及 CUPS
security	user	用于设置 Samba 服务器采用的安全模式。可设置的安全模式为 user、share、domain、server 和 ads。其中，user 表示采用用户级的安全模式，即 Windows 客户必须通过用户名与密码验证后才能访问 Samba 服务器，share 表示采用共享资源级的安全模式，domain 等表示须经 Windows 域控制器验证后才能访问 Samba 服务器。实际上，不管采用哪一种安全模式，Samba 服务器总是以用户为基础验证 Windows 客户系统。 如果 Windows 系统中的用户名与 Linux 系统中维护的用户名相同，应采用“security = user”安全模式；如果 Windows 系统中的绝大多数用户名都不在 Linux 系统中，通常应使用“security = share”安全模式；如果大多数共享资源都不需要密码认证，也应采用“security = share”安全模式；对于共享的打印机服务器，常用的安全模式也是“security = share”。 在 Samba 3.0 中，user 是默认的安全模式。在 user 安全模式下，Windows 客户首先必须拥有且使用一个合法的用户名与密码才能注册，而且还能够使用加密的密码。此外，Samba 服务器还提供大量与用户限制有关的配置变量，这些配置变量均可在 user 安全模式中发挥作用。注意，直至 Samba 服务器已经成功地认证 Windows 客户之后，请求访问的共享资源名才被发送到 Samba 服务器。这也就是在 user 安全模式下，如果不允许 Samba 服务器把未知用户自动映射到“guest”用户时，“guest”共享资源无法工作的原因。 在 share 安全模式下，当 Windows 客户系统访问 Samba 服务器时，在真正开始连接共享资源之前不需提供合法的用户名与密码即可注册，只是在连接每个具体的共享资源时才需要提供密码信息。注意，即使在 share 安全模式下，Samba 服务器总是采用一个合法的 Linux 用户身份限定 Windows 客户的行为。 在 domain 安全模式下，仅当采用 net 实用程序，把 Samba 服务器加入 Windows 域时，domain 安全模式才能正常运行，此外还需要设置“encrypted passwords = yes”配置变量。处于 domain 安全模式时，Samba 服务器将会尝试把用户名/密码传递给 Windows 域控制器，其角色就像 Windows 域中的一个普通 Windows 服务器。 在 server 安全模式下，Samba 服务器首先会把用户名/密码传递给其他 Windows 服务器进行验证。如果失败，再把安全模式恢复到“security = user”模式。在设置 domain 安全模式时，也需要设置“encrypted passwords = yes”配置变量，除非 SMB 服务器不支持。此外还需要维护一个有效的 smbpasswd 文件。 在 ads 安全模式下，Samba 服务器将会成为 Windows ADS 域的一个成员。在设置为 ads 模式之前，Samba 服务器需要安装 Kerberos 软件包，然后还需要采用 net 实用程序加入 ADS 域。注意，采用 ads 安全模式并不意味着 Samba 服务器具有活动目录域控制器的功能



续表

配置变量	默认值	简单说明
server string	%h Server (Samba, Ubuntu)	用于设定在“网络邻居”、“net view”命令输出，以及打印管理的打印机说明界面中显示的信息。当使用“%v”和“%h”等特殊变量时，“%h”将会替换成 Samba 服务器的主机名，“%v”替换成 Samba 的软件版本号
smb ports	445 与 139	指定 Samba 服务器应监听的 TCP 端口，处理其中的 SMB 数据，如“smb ports = 445 139”
socket address	0.0.0.0	用于设定或控制 Samba 服务器监听的 IP 地址，其目的是能够在同一个服务器上同时支持多个虚拟接口，每个接口具有不同的配置。通常，Samba 服务器将会接受对本地系统任何 IP 地址的连接请求
syslog	1	用于设置 Samba 调试信息级别的一个阈值，只有低于这个阈值的调试信息才会通过 syslog 写入系统日志文件。Samba 的调试信息级别 0 等同于系统日志信息级别的 LOG_ERR, 1 等同于 LOG_WARNING, 2 等同于 LOG_NOTICE, 3 等同于 LOG_INFO, 其他调试信息级别对应于 LOG_DEBUG
syslog only	no	如果把这个配置变量设置为 yes，Samba 服务器的调试信息将会通过 syslog 写入系统日志文件，而不是写入 Samba 自己的调试日志文件
time server	no	用于确定是否需要 samba 服务器作为一个时间服务器，为 Windows 系统提供时间同步服务
unix password sync	no	当 smbpasswd 维护的密码文件中加密的 SMB 密码发生变动时，这个配置变量用于控制 Samba 服务器是否需要与 Linux 系统中的密码进行同步。如果把这个配置变量设置为 yes，Samba 服务器需要以超级用户的身份，调用“passwd program”配置变量指定的程序，重新设置 Linux 系统中的用户密码
username map	无	这个配置变量用于指定一个文件，如“username map = /var/lib/samba/usermap”。其中包含从 Windows 客户到 Samba 服务器的用户名映射，其主要用途是把 Windows 用户名映射为 Linux 用户名。此外，还可以把多个用户映射为单个用户，以便 Windows 用户能够容易地共享文件。在用户名映射文件中，每行只能包含一个 Linux 用户名，位于等号“=”的左边，右边是一系列 Windows 用户名。例如，如果在 Linux 系统中的用户名是 gqxing，而在 Windows 2000 和 Windows XP 系统中的用户名分别是 administrator 和 admin，为了连接到自己的主目录，可以定义下列用户名映射。 gqxing = administrator admin 注意，在 user 或 share 安全模式中，用户名的映射是在用户认证之前完成的。而在 domain 或 ads 安全模式中，用户名的映射是在域控制器成功地认证用户之后完成的
wins support	no	用于控制 Samba 服务器的 nmbd 守护进程是否作为 WINS 服务器，提供 WINS 名字服务。通常不应设为 yes
workgroup	WORKGROUP	设置 Samba 服务器所属工作组或域（当“security = domain”时）的名字

### 22.2.3 homes 节

在 Windows 系统中，当用户想要以 Linux 系统用户的身份连接到自己在 Linux 系统中的主目录，使之作为一个逻辑驱动器，实现文件的传输与交换时，需要在 smb.conf 文件中增加一个[homes]节，以定义用户主目录。

除了用户主目录之外，还可以在 Samba 服务器中定义任何目录，作为共享资源，并对每个共享资源采取不同的安全措施。表 22-3 给出了可用于定义[homes]节及其他共享资源节时常用的部分配置变量。

表 22-3 [homes]节常用部分的配置变量

配置变量	默认值	简单说明
admin users	无	用于指定一个用户名列表，其中的用户有权限管理相应的共享资源，这意味着指定的用户能够像超级用户一样，执行所有的文件操作。设置这个配置变量时应当特别小心，由于用户名列表中的任何一个用户都能够对共享资源做他们想要做的任何事情，而不必顾及文件本身的访问权限
available	yes	这个配置变量相当于一个共享资源的开关，用于控制 Windows 用户是否能够访问相应的共享资源。如果设置为 no，意味着“关闭”共享资源，拒绝任何 Windows 用户连接或访问相应的共享资源

续表

配置变量	默认值	简单说明
browsable	yes	用于控制相应共享资源的可视性。当 Windows 用户通过“网络邻居”或“net view”命令浏览服务器，查询共享资源时，这个配置变量决定了相应的共享资源是否能够出现在可用的共享资源列表中。
comment	无	用于指定共享资源的说明信息。当 Windows 用户通过“网络邻居”或“net view”命令查询服务器，浏览共享资源时，将会显示此信息。
create mask	0744	用于指定新建文件的默认访问权限。默认的 8 进制数值 0744 表示，除了文件属主之外，同组和其他用户只能读，但不能写和执行新建的文件。Samba 服务器将会以此为基础，按位“逻辑或”配置变量“force create mode”指定的数值，作为新建文件的最终访问权限设置。
directory mask	0755	用于设置新建目录的默认访问权限。默认的 8 进制数值 0755 表示，除了目录属主之外，同组和其他用户没有写的权限，即不能在新建的目录中创建或修改文件。Samba 服务器将会以此为基础，按位“逻辑或”配置变量“force directory mode”指定的数值，作为新建目录的最终访问权限设置。
force create mode	000	用于指定 8 进制的 Linux 文件访问权限。Samba 服务器总是使用这个设置，连同“create mask”配置变量的设置，按位执行逻辑或运算之后，作为新建文件的最终访问权限。
force directory mode	000	用于指定 8 进制的 Linux 目录访问权限。同样，Samba 服务器总是使用这个设置，连同“directory mask”配置变量的设置，按位执行逻辑或运算之后，作为新建目录的最终访问权限。
force group	无	用于指定一个 Linux 系统的用户组名，作为连接到相应共享资源的所有 Windows 用户的默认用户组名。
force user	无	用于指定一个 Linux 系统的用户名，作为连接到相应共享资源的所有 Windows 用户的默认用户名。这个用户名仅在连接建立之后发生作用，之前，Windows 用户仍需使用一个合法的用户名，提供正确的密码，才能连接相应的共享资源。一旦建立连接，所有的文件操作将以“强制替换”的用户身份执行，而不管 Windows 用户采用什么用户名连接 Samba 服务器。
guest ok	no	这个配置变量如果设置为 yes，意味着无需使用密码即可连接或访问相应的共享资源，其访问权限与 Linux 系统中的“guest”用户（即 nobody）相同。
guest only	no	如果把这个配置变量设置为 yes，意味着只允许“guest”用户连接或访问相应的共享资源。但是，如果共享资源中没有设置“guest ok=yes”，则不管这个配置变量是否设置都没有影响。
hide dot files	yes	用于控制是否把以句点“.”为起始字符的文件作为隐藏文件处理。
hide files	无	用于指定不能浏览，但可访问的文件或目录列表。文件或目录名之间以斜线字符“/”作为分隔符，这意味着文件或目录名中可以包含空格字符。此外，还可使用星号“*”和问号“?”作为文件或目录名通配符。在设定文件或目录时，其名字中不能包含路径名分隔符“/”，即不能包含上一级的目录。例如，配置变量“hide files = ./公共的/模版/视频/图片/文档/音乐/桌面/”表示不显示 Linux 系统的隐藏文件，以及用户主目录中的常规中文目录等。注意，“case sensitivity”配置变量的设置也会影响隐藏文件的设置。此外，设置这个配置变量会影响 Samba 服务器的性能，由于这需要强制检测所有的文件与目录是否匹配设定的隐藏原则。还有，在显示目录文件列表时，Windows 系统的隐藏规则仍然适用。
hide special files	no	如果把这个配置变量设置为 yes，能够防止 Windows 客户显示特殊文件，如套接字、设备文件及管道文件等。
hide unreadable	no	如果把这个配置变量设置为 yes，能够防止 Windows 客户显示无权读取的文件。
hide unwriteable files	no	如果把这个配置变量设置为 yes，能够防止 Windows 客户显示无权写的文件。注意，无权写的目录不受影响，照样能够显示。
hosts allow 或 allow hosts	无	用于指定允许访问相应共享资源的 Windows 系统。在指定 Windows 系统时，可以使用主机名、表示网号的 IP 地址或“IP 地址/netmask”。在列举多个 Windows 系统时，中间可加逗号“，”、空格或制表符等分隔符。如果“hosts allow”配置变量是在[global]节中设置的，则适用于所有的共享资源，而不管每个具体的共享资源是如何设置的。注意，除非在“hosts deny”配置变量中明确拒绝，总是允许本地主机 localhost (127.0.0.1) 访问 Samba 服务器。如果在列举 Windows 系统时采用通配符，也可以使用关键字 EXCEPT 排除其中的部分 Windows 系统。例如，下列设置允许 150.203.0.0 子网中的任何 Windows 系统访问，但 150.203.6.6 系统除外。 hosts allow = 150.203. EXCEPT 150.203.6.6



续表

配置变量	默认值	简单说明
hosts deny 或 deny hosts	无	用于指定不允许访问相应共享资源的 Windows 系统。如果这个配置变量的设置与“hosts allow”冲突，则以“hosts allow”配置变量的设置为准。通常最好先采用“hosts deny”配置变量拒绝“所有的”Windows 系统（使用关键字 ALL 等），然后再采用“hosts allow”配置变量逐一列出“允许访问的”Windows 系统
invalid users	无	用于指定不允许注册 Samba 服务器访问相应共享资源的用户。这是一种绝对保险的附加措施，能够确保其他不当设置不会危及 Samba 服务器的安全
max connections	0	指定共享资源的最大并发连接数量限制。默认值 0 意味着没有并发连接的数量限制
only user	no	这个配置变量用于控制是否允许 Windows 用户使用用户列表中指定的用户名连接 Samba 服务器，访问相应的共享资源。这个配置变量的默认设置是 no，意味着 Windows 用户能够使用任何一个 Samba 服务器可用的用户名。如果设置为 yes，将会强制 Samba 服务器只能使用用户列表中指定的用户名，这种情况仅在“security = share”安全模式中有实用价值
path	无	用于指定一个目录，作为 Windows 用户能够访问的共享资源。在共享打印资源的情况下，这个配置变量指定的是在提交系统打印之前暂存打印数据的缓存目录。如果指定的目录含有“%u”和“%m”等特殊变量，Samba 服务器将会使用 Windows 系统建立连接时提供的 Linux 系统的用户名替换“%u”，使用 Windows 系统的 NetBIOS 名替换“%m”。在设置非常规的用户名目录时，这种替换是非常有用的
read list	无	不管“read only”配置变量的设置如何，这个配置变量指定的所有用户只能读取不能写入相应的共享资源。在指定用户时，可以采用“invalid users”配置变量中说明的语法“@group”指定用户组名。例如，配置变量“read list = mary, @students”表示，用户 mary 与用户组 students 中的所有成员只能读取相应的共享资源
read only	yes	这个配置变量如果设置为 yes，意味着不能在相应的共享目录资源中创建或修改文件。多数情况下需要把这个配置变量设置为 no，从而允许 Samba 用户读写共享目录。注意，可打印的共享资源（其配置变量为“printable = yes”）总是允许通过假脱机方式写入目录
valid users	无	这个配置变量用于指定允许注册 Samba 服务器，访问相应共享资源的用户。如果这个配置变量为空（默认值），任何用户均可访问 Samba 服务器。如果用户名同时出现在“invalid users”与这个配置变量中，则拒绝相应的用户访问 Samba 服务器
writable	无	与“read only”配置变量的意义恰好相反
write list	无	不管配置变量“read only”的设置如何，这个配置变量列举的用户均具有读写相应共享资源的访问权限。如果某个用户同时位于“read list”与“write list”两个配置变量中，则后者为准。同样，在指定用户时，可以采用“invalid users”配置变量中说明的语法“@group”指定用户组名。例如，配置变量“write list = admin, root, @staff”表示，用户 admin 与 root 及用户组 staff 中的所有成员均可读写相应的共享资源

## 22.2.4 printers 节

Samba 服务器辟有专门的打印共享资源[printers]节，用于设置共享打印机。[printers]节的定义方法类似于[homes]节，只是针对打印机而已。在[printers]节中，配置变量 printable 总是应当设置为 yes。此外，[homes]节中介绍的配置变量也可直接用于[printers]节，表 22-4 仅给出了只能用于[printers]节的部分配置变量。

表 22-4 [printers]节常用的部分配置变量

配置变量	默认值	简单说明
printable	No	这个配置变量如果设置为 yes，则 Windows 用户能够在共享打印资源指定的目录中打开、写入以及提交打印文件。注意，共享打印资源总是允许用户通过 Linux 系统的打印服务写入共享打印资源对应的缓存目录。“read only”配置变量仅用于非打印共享资源的访问控制
printer name	无	用于设定网络共享打印机的名字，以便 Windows 用户能够通过共享打印资源提交打印作业。如果是在 global 节中设置“printer name”配置变量，则适用于未指定打印机名的任何共享打印资源

## 22.3 快速设置 Samba 服务器

假定存在一个小型网络，Ubuntu Linux 系统配置为 Samba 服务器，其中提供用户主目录、文档目录/var/docs 以及网络打印机等共享资源。两个专业版的 Windows 系统作为客户系统（或服务器），分别以 admin 和 administrator 注册到各自的系统，如见图 22-2。下面几小节以此为例，说明如何配置 Samba 服务器。

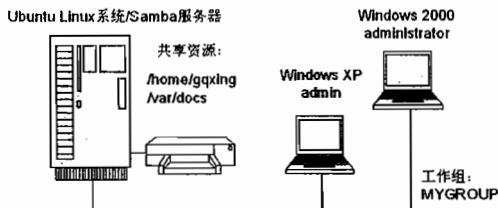


图 22-2 Samba 运行环境示例

### 22.3.1 设定 Samba 服务器的工作组或域

在 Samba 服务器的 [global] 节中，尽管存在大量的配置变量，但必须设置的配置变量只有 workgroup，用于定义 Samba 服务器所在的 Windows 工作组或域，其他配置变量尽可采用其默认设置。按照图 22-1 的要求，可以把 Samba 服务器配置为 MYGROUP（或保持默认值 WORKGROUP 不变）工作组中的一个服务器成员，如下所示。

```
[global]
workgroup = MYGROUP
```

此外，如果需要把 Samba 服务器定义为一个主域控制器（Primary Domain Controller, PDC），还需要把“domain master”配置选项设置为“yes”或“auto”，示例如下。

```
domain master = auto
```

上述配置意味着，如果所在的 Windows 域中没有域控制器，Samba 服务器将会自动提供主域控制器的功能。

在一个小型的本地工作组网络中，通常不需要在 Samba 服务器中启用 WINS（Windows NetBIOS 名字服务）。如果确有必要，可以使用 Linux 系统中的 DNS 域名服务器解析 Windows 系统，有关的配置变量可设置如下。

```
wins support = no
dns proxy = yes
name resolve order = lmhosts host wins bcast
```

上述最后一个配置变量告诉 Samba 服务器，在解析主机名字时，Windows 系统用户首先会检索自己本地的 lmhosts 文件，然后检索 Samba 服务器中的/etc/hosts 文件，接着检索 WINS（由 DNS 域名服务器代理实现），最后采用广播的方式检索出正确的主机。

为了便于不同的 Windows 用户以同一用户身份与权限访问共享资源，尚需增加下列“username map”配置变量，定义一个用户名影射文件。

```
username map = /var/lib/samba/smbuser.map
```

此外，还需要创建配置变量中定义的 /var/lib/samba/smbuser.map 文件，其中包含必要的用户



名映射关系。例如，为了使 Window 2000 系统中的 administrator 与 Windows XP 系统中的 admin 用户能够以 gqxing 用户的身份访问 Samba 服务器，可在 smbuser.map 文件中增加下列映射关系。

```
gqxing = administrator admin
```

如果想要利用 SWAT 配置 Samba 服务器，修改 smb.conf 文件的全局配置变量，需要以超级用户 root 的身份注册 SWAT。在 SWAT 主界面中，单击“GLOBALS”按钮，然后即可在全局变量设置界面，查询或修改任何配置变量。例如，要修改“username map”配置变量，可在相应的栏目中输入用户名映射文件的完整路径名。在完成所有的设置之后，单击“Commit Changes”按钮，以保存所做的任何修改。

### 22.3.2 设置 Samba 用户认证信息

初始配置的 Samba 服务器可以直接用作 Windows 工作组的一个成员服务器，如果想要访问其中提供的共享资源，Windows 用户必须首先获得 Samba 服务器的认证。Samba 服务器维护自己的用户名与密码等用户认证信息。在 Ubuntu Linux 系统中，Samba 服务器的用户密码存储在“passdb backend”配置变量指定的文件中，如/var/lib/samba/passdb.tdb 文件。如果需要，可以利用 smbpasswd 命令增加或修改 Samba 服务器维护的用户，设置用户名及其密码。smbpasswd 命令的语法格式如下。

```
smbpasswd [-adenx] [-h] [username]
```

其中，“-a”选项表示把命令行中指定的用户加到 Samba 服务器的 smbpasswd 文件中，同时输入新的密码。通常，指定的用户名应存在于本地系统的/etc/passwd 文件中。“-x”选项表示从 smbpasswd 文件中删除指定的用户。“-n”选项表示应把指定用户的密码设为空，即把“NO PASSWORD”作为起始字符串写入 smbpasswd 文件中，同时还需要事先在 smb.conf 配置文件的[global]节中设置“null passwords = yes”配置变量，否则无法达到无密码注册 Samba 服务器的目的。“-d”选项用于禁用指定的用户账号。“-e”选项用于启用先前禁用的指定用户账号。“-h”选项用于显示 smbpasswd 命令的帮助信息，username 用于指定 Windows 系统用户（通常也必须是 Samba 服务器所在系统中的用户）。注意，运行 smbpasswd 命令需要拥有超级用户的访问权限。

从安全的角度考虑，Samba 服务器的用户密码可以不同于 Linux 系统的用户密码。采用不同用户密码的目的是增加用户管理的灵活性，便于控制访问 Samba 服务器的用户。

通常，Linux 系统的超级用户 root 也是 Samba 服务器的管理员。此外，SWAT 也要求使用 Linux 系统的 root 用户及其密码注册，才能设置任何共享资源。必要时，还可以使用下列 smbpasswd 命令，设置或修改 Samba 服务器管理员的密码。

```
# /usr/bin/smbpasswd -a root
```

使用下列 smbpasswd 命令，可以把 Linux 系统中的用户 gqxing 加到 Samba 服务器（首先输入 sudo 密码，然后再输入指定用户在 Samba 服务器中的密码）。

```
$ sudo smbpasswd -a gqxing
[sudo] password for gqxing:
New SMB password:
Retype new SMB password:
$
```

当需要由多个用户共同管理 Samba 服务器的不同共享资源时，还可以在 smb.conf 配置文件中设置“admin users”等配置变量。

在 Samba 配置文件中，默认的安全级别设置是“security = user”，表示采用用户级的认证方

式。用户认证方式要求 Samba 服务器支持的用户也必须是 Ubuntu Linux 系统中的用户，否则在使用 `smbpasswd` 命令增加用户时将会收到一条出错信息。若想采用其他安全方式，可参考 `samba-doc` 软件包中的 `/usr/share/doc/samba-doc/htmldocs/ServerType.html` 文件。

### 22.3.3 共享用户主目录

在设置了工作组，创建了 Samba 用户之后，下一步需要考虑的是 Samba 服务器能够为 Windows 用户提供什么共享资源。

在 Samba 运行环境中，最常用的共享资源主要是用户主目录和打印机等，而最常见的资源共享方式是从 Windows 系统中访问自己在 Linux 系统中的主目录。利用用户主目录，能够容易地实现文件的集中存储与交换。

在 Ubuntu 8.10 Linux 系统中，默认的 `/etc/samba/smb.conf` 配置文件并未把用户主目录设置为共享资源，如果需要共享用户主目录，可在配置文件中增加一个简单的 [homes] 节及适当的配置变量。然后，只要提供的用户名与密码匹配 Linux 与 Windows 系统中的用户与密码设置，即可在 Windows 环境中访问自己在 Linux 系统中的主目录，示例如下。

```
[homes]
comment = Home Directories
browseable = Yes
read only = No
create mask = 0644
directory mask = 0775
```

注意，把 `browseable` 配置变量设置为 `Yes`，可以在 Windows 系统的“网络邻居”或“net view”命令中观察到相应的共享资源，否则无法知道可用的共享资源，初次访问时也无法通过单击鼠标进入相应的共享资源。此外，把“`read only`”配置变量设置为 `No`，意味着允许用户在 Samba 环境中读写自己的主目录，否则，即使用户主目录是一个可用的共享资源，仍然是不可写的。

针对上述设置，可以直接编辑 `/etc/samba/smb.conf` 配置文件，把上述配置变量直接写入 `smb.conf` 配置文件的适当位置；也可以使用超级用户 `root` 注册 SWAT，利用 SWAT 工具进行设置。要设置用户主目录，可在 SWAT 主界面中单击“SHARES”按钮，查询或修改现有的配置变量。在“Choose Share”下拉框中选择“`homes`”，或者在“Create Share”栏目中输入 `homes`，然后单击“Advanced”按钮，选择修改每一个配置变量，修改结束后单击“Commit Changes”按钮，即可创建一个新的 `smb.conf` 配置文件。最后，可以单击界面顶部的“STATUS”按钮，重新启动 Samba 服务器的 `smbd` 与 `nmbd` 守护进程。

### 22.3.4 共享其他目录

下面的例子说明了怎样定义一个普通的共享目录资源，用于存储或交换文件，其共享资源名为 `docs`，路径名为 `/var/docs`，通过认证的任何用户均可访问新设置的共享资源。

```
[docs]
comment = Shared folder with username and password
path = /var/docs
force user = nobody
force group = nogroup
create mask = 0644
directory mask = 0755
read only = No
guest ok = Yes
```



同样，为了实现上述设置，可以直接编辑 smb.conf 配置文件，也可以利用 SWAT 工具，单击 SWAT 主界面中的“SHARES”按钮，在“Create Share”栏目中输入一个准备创建的共享目录名，如 docs，然后单击“Create Share”按钮。此时，可在相应的配置变量栏目中输入上述数据，如路径名/var/docs 等。输入结束后单击“Commit Changes”按钮，创建一个新的 smb.conf 配置文件。最后点击界面顶部的“STATUS”按钮，重新启动 Samba 服务器的 smbd 与 nmbd 守护进程。

注意，除了设置 smb.conf 配置文件，还需要使用下列命令，事先创建相应的目录，设置目录的属主、用户组及其访问权限。

```
$ sudo mkdir /var/docs  
$ sudo chown nobody /var/docs  
$ sudo chgrp nogroup /var/docs  
$ sudo chmod 777 /var/docs  
$
```

### 22.3.5 共享打印机

在设置[printers]节时，可以使用定义用户主目录或普通共享目录资源时需要的任何配置变量。在 Ubuntu Linux 系统中，为了配置网络共享打印机，需要确保 Samba 配置文件[global]节中的下列配置变量已经被适当地设置。

```
load printers = yes  
printing = cups  
printcap name = cups
```

在[printers]与[print\$]节的配置中，需要提供 Windows 用户与 Samba 服务器的交互信息。如果配置文件中存在[printers]节，当用户请求连接打印机时，Samba 服务器将会扫描[printers]节的定义。如果发现匹配的打印机，即可连接到本地主机 printcap 文件中指定的任何打印机。安装 Samba 软件包之后，默认的 smb.conf 配置文件内容如下（如果需要，可根据实际情况，对配置变量做出适当的修改）。

```
[printers]  
comment = All Printers  
path = /var/spool/samba  
create mask = 0700  
printable = yes  
browseable = No  
  
[print$]  
comment = Printer Drivers  
path = /var/lib/samba/printers
```

[printers] 节说明了 Samba 服务器怎样处理 Windows 系统的打印机标识与打印请求。[print\$] 节用于定义一个附加的共享资源（即 Linux 系统中的一个目录），其中可以存储有关的打印机驱动程序，以便 Windows 用户能够通过网络方式，安装自己的打印机驱动程序。

下面是一个典型的共享打印资源定义。其中的共享目录资源是只读的，但可以打印，即能够通过打印系统，提交打印文件，达到访问共享打印资源的目的。“guest ok = yes” 配置变量意味着“guest” 用户也可访问设定的共享打印机。

```
[Laser-2300]  
comment = HP LaserJet 2300 Printer  
path = /var/spool/cups  
printable = yes  
guest ok = yes
```

在配置 Samba 打印机之前，首先需要在 Linux 系统中增加与设置打印机，为此可以在命令行调用“system-config-printer”命令，或者从 GNOME 桌面中选择“系统→系统管理→打印”菜单，利用图形界面的打印机管理工具，设置打印机（参见第 2 章）。

然后，可以开始配置 Samba 服务器控制的打印机。如果采用 SWAT 工具配置打印机，需用 root 注册 SWAT，在进入 SWAT 主界面后，参照下列步骤设置共享打印机。

(1) 单击“PRINTERS”按钮，从“Choose Printer”栏目的下拉菜单中选择一个共享打印机。然后单击“Basic”或“Advanced”按钮，根据上述内容，设置相应的配置变量。注意，如果可选的打印机名前边有一个星号“\*”，表示可以把相应的打印机作为共享打印机。

(2) 如果 Samba 服务器没有可自动配置的打印机，需要自己编辑/etc/samba/smb.conf 配置文件，增加打印机配置变量。

(3) 单击“Commit Changes”按钮，更新 smb.conf 配置文件。

(4) 单击 SWAT 界面顶部的“Status”标签，重新启动 Samba 服务器的 smbd 和 nmbd 守护进程。

至此，Samba 服务器的打印机配置已经就绪，可以供 Windows 客户访问共享打印资源了。但还有最后一步，即需要在 Windows 系统中安装相应的打印机驱动程序。

### 22.3.6 验证 Samba 配置文件

在修改 Samba 服务器的配置文件之后，应当检验其中是否存在语法问题。如果配置文件中包含任何设置无效的配置变量，在启动过程中，Samba 服务器将会显示出错信息，并拒绝加载。Samba 服务器软件包提供的 testparm 命令可用于配置文件的语法检测。在运行 testparm 命令时如果不加任何参数，默认的配置文件是/etc/samba/smb.conf。示例如下。

```
$ testparm
Load smb config files from /etc/samba/smb.conf
Processing section "[homes]"
Processing section "[printers]"
Processing section "[LaserJet-2300]"
Processing section "[print$]"
Processing section "[docs]"
Loaded services file OK.

WARNING: You have some share names that are longer than 12 characters.
These may not be accessible to some older clients.
(Eg. Windows9x, WindowsMe, and smbclient prior to Samba 3.0.)
Server role: ROLE_STANDALONE
Press enter to see a dump of your service definitions
```

在显示部分概括信息之后，testparm 命令提示用户按 Enter 键，以便查阅 Samba 服务器的现行配置能够提供的共享资源详情。示例如下。

```
[global]
server string = %h server (Samba, Ubuntu)
map to guest = Bad User
obey pam restrictions = Yes
passdb backend = tdbsam
pam password change = Yes
passwd program = /usr/bin/passwd %u
passwd chat = *Enter\snew\s*\spassword:*\n\n*Retype\snnew\s*\spassword:*\n\n
*password\supdated\ssuccessfully* .
unix password sync = Yes
syslog = 0
log file = /var/log/samba/log.%m
max log size = 1000
name resolve order = lmhosts host wins bcast
printcap name = cups
panic action = /usr/share/samba/panic-action %d

[homes]
comment = Home Directories
read only = No
```



```
create mask = 0644
browseable = No

[printers]
comment = All Printers
path = /var/spool/samba
create mask = 0700
printable = Yes
browseable = No

[LaserJet-2300]
comment = HP LaserJet 2300 Printer
path = /var/spool/cups
printable = Yes

[print$]
comment = Printer Drivers
path = /var/lib/samba/printers

[docs]
comment = Shared folder
path = /var/docs
force user = nobody
force group = nogroup
read only = No
create mask = 0644
$
```

testparm 命令只能检测 Samba 配置文件的语法是否正确，无法查询 Samba 服务器的运行状态及提供的服务。例如，上述警告信息说明共享资源名 LaserJet-2300 太长，超过了 12 个字符。至于共享资源的配置是否合理，是否反映了用户的意愿，需要自己测试和验证。如果想要查询 Samba 服务器提供的共享资源等服务信息，可以使用下面即将介绍的 smbclient 等命令。

## 22.4 Samba 运行环境测试

在确保 TCP/IP 网络正常运行的前提下，通常还需要测试 Samba 服务器的配置是否正确，运行是否正常。实际测试时，可以在 Linux 和 Windows 系统中分别进行。

### 22.4.1 在 Linux 系统中测试 Samba 服务器

Samba 提供的 smbclient 是一个命令行的客户端软件，用于连接、浏览 Samba 服务器，直接与 SMB/CIFS 对话。在交互运行模式下，smbclient 提供一个类似于 ftp 程序的界面，可用于下载、上传文件，改换目录，查询文件属性，显示目录文件列表，等等；也可用于测试 Samba 服务器是否已经正常运行，显示其提供的共享资源。smbclient 命令的语法格式如下。

```
smbclient [-L netbios-name] [-U username[%password]] [-I IPaddress] [-p port]
smbclient {servicename} {password} [-U username[%password]] [-I IPaddress] [-p port]
```

其中，servicename 是 SMB/CIFS（Samba）服务器提供的服务名，一个完整的服务名通常采用`//server/service`的形式定义，其中 server 是 NetBIOS 主机名，service 是服务名。例如，为了连接 SMB/CIFS 服务器 smbserver 中的 printer 服务，可以使用服务名`//smbserver/printer`。password 是访问指定服务器的指定服务时需要提供的密码。如果命令行中忽略这个参数，可在交互方式下提供密码。“-I”选项用于指定 Samba 服务器的 IP 地址。注意，通常应当首先使用 NetBIOS 主机名解析方式确定 SMB/CIFS 服务器。“-p”选项用于指定连接 Samba 服务器时使用的 TCP 端口号。

对于 SMB/CIFS 服务器而言，标准的 TCP 端口号是 139。“-L”选项用于显示指定服务器提供的共享资源等服务，以及服务器的主机名等信息。“-U”选项用于指定用户名或用户名与密码。如果未指定“%password”形式的密码，smbclient 将会提示用户输入密码。如果未指定用户名，smbclient 首先会检查 USER 变量，然后检查 LOGNAME 变量，确定大写字母形式的用户名。如果仍未获取用户名，则假定当前的用户为 guest。

例如，使用下列 smbclient 命令，可以查询 Samba 服务器提供的共享资源。

```
$ smbclient -L iscas
Enter gqxing's password:
Domain=[ISCAS] OS=[Unix] Server=[Samba 3.2.3]

      Sharename          Type          Comment
-----  -----          -----
homes        Disk          Home Directories
print$       Disk          Printer Drivers
docs         Disk          Shared folder
IPC$         IPC           IPC Service (iscas server (Samba, Ubuntu))
LaserJet-2300  Printer      HP LaserJet 2300 Printer
gqxing       Disk          Home Directories
Domain=[ISCAS] OS=[Unix] Server=[Samba 3.2.3]

      Server          Comment
-----
      Workgroup        Master
-----
      WORKGROUP        ISCAS
$
```

下列 smbclient 命令用于显示一个默认的、非认证的用户从服务器中能够得到何种共享资源或服务。

```
$ smbclient -L iscas -U %
Domain=[WORKGROUP] OS=[Unix] Server=[Samba 3.2.3]

      Sharename          Type          Comment
-----  -----          -----
homes        Disk          Home Directories
print$       Disk          Printer Drivers
docs         Disk          Shared folder
IPC$         IPC           IPC Service (iscas server (Samba, Ubuntu))
LaserJet-2300  Printer      HP LaserJet 2300 Printer
Domain=[WORKGROUP] OS=[Unix] Server=[Samba 3.2.3]

      Server          Comment
-----
      ISCAS           iscas server (Samba, Ubuntu)
USER-281A27FCAB

      Workgroup        Master
-----
      WORKGROUP        ISCAS
$
```

在上述输出信息中，第一部分列出了一个未认证的用户能够访问的共享资源，后续部分提供了 Samba 服务器的主机名，以及服务器所在工作组或域的名字等信息。

下面的例子是查询一个经过认证的用户能够访问的共享资源。

```
$ smbclient -L iscas -U gqxing
Enter gqxing's password:
Domain=[ISCAS] OS=[Unix] Server=[Samba 3.2.3]

      Sharename          Type          Comment
-----  -----          -----
homes        Disk          Home Directories
```



```
print$          Disk      Printer Drivers
docs           Disk      Shared folder
IPC$           IPC       IPC Service (iscas server (Samba, Ubuntu))
LaserJet-2300  Printer   HP LaserJet 2300 Printer
gqxing         Disk      Home Directories
Domain=[ISCAS] OS=[Unix] Server=[Samba 3.2.3]

Server          Comment
-----
Workgroup       Master
-----
WORKGROUP       ISCAS
$
```

上述两个命令的运行过程与输出结果均有不同：后者提示输入指定用户在 Samba 服务器中的密码，显示的共享资源多了一个用户主目录 gqxing。也就是说，经过认证的用户具有访问其在 Samba 服务器上主目录的权限。

上述输出信息也表示，Samba 服务器的配置与运行均正常，允许用户 gqxing 访问自己在 Linux 系统中的主目录。

利用 smbclient 命令，还可以采用交互方式访问 Samba 服务器或 Windows 服务器。例如，下列 smbclient 命令能够注册到 Samba 服务器，以交互方式访问指定的 Samba 服务器与共享资源。当系统提示输入密码时，输入 gqxing 用户在 Linux 系统中的密码。也可以在命令行后面增加一个“-U username”选项，明确指定一个用户名。如果 Samba 服务器的配置与运行均正常，即可进入交互会话方式，如下所示。

```
$ smbclient //iscas/gqxing
Enter gqxing's password:
Domain=[ISCAS] OS=[Unix] Server=[Samba 3.2.3]
smb: \> ?
?
allinfo      altname      archive      blocksize
cancel       case_sensitive cd           chmod
close        del           dir           du
exit         get           getfacl      hardlink
history      iosize       lcd           link
lowercase    ls            l             mask
mget         mkdir        more          mput
open         posix        posix_encrypt posix_open
posix_rmdir  posix_unlink print        prompt
pwd          q             queue        quit
recurse     reget        rename       reput
rmdir       showacl      setmode      stat
tar         tarmode      translate   unlock
vuid        wdel         logon        listconnect
..
!
smb: \>
```

在“smb: \>”提示符下，可以使用诸如 cd、dir、pwd、ls、get、put 及 mkdir 等命令，显示文件列表，传输文件，等等。从字面上即可看出，这些命令完全类似于 ftp 的命令，其用法也基本上完全一样，故不再赘述。

Samba 软件包还提供一个 nmblookup 命令，可在网络上广播一个查询信息，用于确定 Samba 软件是否能够正确地运行。其中，iscas 是 Samba 服务器的 NetBIOS 名字。如果 Samba 服务器运行正常，下列命令将会返回 Samba 服务器 iscas 的 IP 地址。

```
$ nmblookup iscas
querying iscas on 169.254.255.255
169.254.78.100 iscas<00>
$
```

在 Samba 服务器中，还可以使用下列 nmblookup 命令，查询指定的工作组中是否存在主浏览服务器。如果测试成功，将会输出主浏览服务器的 IP 地址。否则，需要考察 smb.conf 文件[global]节中的“preferred master”配置变量是否已经设置为 yes。

```
$ nmblookup -M workgroup
querying workgroup on 169.254.255.255
169.254.78.100 workgroup<1d>
$
```

## 22.4.2 从 Windows 系统中连接 Samba 服务器

在 Windows 客户系统中，通常不需要做任何设置，只要提供的用户名与密码是正确的，即可访问 Samba 服务器。

如果网络连接正常，且 Samba 服务器的配置是正确的，在 DOS 命令行中输入“net view \\samba-server”命令后，将会返回 Samba 服务器提供的共享资源。如果测试失败，需要检查 smb.conf 文件中的“hosts allow”、“hosts deny”及“invalid users”等配置变量的设置是否正确。示例如下。

```
C:\Documents and Settings\admin>net view \\iscas
在 iscas 的共享资源
iscas server (Samba, Ubuntu)
共享名          类型      使用为    注释
-----
docs           Disk       Shared folder
homes          Disk       Home Directories
LaserJet-2300 Print      HP LaserJet 2300 Printer
命令成功完成。
```

```
C:\Documents and Settings\admin>
```

在 Windows 客户系统中，也可以使用“net use x: \\samba-server\share”命令，把自己在 Samba 服务器中的主目录映射到某个逻辑驱动器。下面的例子表示把 Linux 系统用户 gqxing 的主目录映射到 Windows 系统的逻辑驱动器 P。

```
C:\Documents and Settings\admin>net use P: \\iscas\gqxing "mypasswd" /user:"gqxing"
命令成功完成。
C:\Documents and Settings\admin>dir p:
驱动器 P 中的卷是 gqxing
卷的序列号是 053B-082D
P:\ 的目录
2009-01-19  00:15    <DIR>
2009-01-07  14:02    <DIR>        ..
2008-12-07  23:42    <DIR>        data
2008-12-23  16:30    <DIR>        conf
2008-10-30  08:49    <DIR>        Examples
2009-01-12  17:19    <DIR>        桌面
2008-12-05  14:39    <DIR>        文档
2008-12-05  14:39    <DIR>        音乐
.....
C:\Documents and Settings\Admin>
```

下列命令将会尝试利用 Windows 系统的注册用户名（如 admin）与 Linux 系统用户（gqxing）的密码注册到 Samba 服务器。此前应确保已经创建了相应的 Samba 用户，且设置了“username map”配置变量及相应的用户名映射文件。

```
C:\Documents and Settings\Admin>net use P: /del
P: 已经删除。
```

```
C:\Documents and Settings\Admin>net use P: \\iscas\gqxing "mypasswd"
命令成功完成。
C:\Documents and Settings\Admin>
```



如果上述命令运行失败，需要检查“username map”配置变量及相应的用户名映射文件是否已正确地设置。此外，Windows 系统当前仅发送加密的密码，如果上述命令仍然无法运行，可能还需要检查 smb.conf 文件[global]节中的“encrypt password”配置变量是否已经设置为 yes。

如果上述测试一切正常，从 Windows 系统的“网络邻居”中能够看到定义的 Windows 工作组或域，也可以进一步浏览 Samba 服务器及其提供的共享资源，访问其中的文件等。

## 22.5 访问共享资源

### 22.5.1 从 Windows 系统中访问 Samba 服务器

在 Windows 客户系统中，当需要连接 Samba 服务器，访问其中提供的共享资源时，可以参照下列步骤连接到自己在 Linux 系统中的主目录或打印机等共享资源。

- (1) 在 Windows 桌面中双击“网络邻居”。
- (2) 在“网络邻居”窗口中双击“整个网络”。
- (3) 双击“Microsoft Windows Network”。
- (4) 双击“Workgroup”。
- (5) 双击“iscas server (Samba, Ubuntu) (Iscas)”，在新弹出的“连接到 xxx”对话框中输入用户名（如 gqxing）和密码。
- (6) 双击用户主目录（如 gqxing）或打印机，即可连接到 Samba 服务器中的共享资源，出现图 22-3 所示的窗口。

利用上述方法，可以随时访问共享资源，但当重新启动 Windows 时，还需要重新建立资源连接。如果想要在启动系统之后快速建立连接，可采用下列步骤，建立驱动器与 Linux 共享目录的映射关系。

- (1) 右击桌面上的“我的电脑”图标，从上下文菜单中选择“映射网络驱动器”。
- (2) 在“映射网络驱动器”对话框中选则逻辑驱动器，如“Z:”。
- (3) 单击“浏览”按钮，在新弹出的“浏览文件夹”窗口中浏览“Workgroup”，选择 Samba 服务器（如 iscas），选择共享资源（如 docs），或者选择用户主目录（如 gqxing）。注意，第一次连接用户主目录时需要在“映射网络驱动器”对话框的“文件夹”字段中直接输入“\\samba-server\share”形式的共享资源（如\\iscas\gqxing），然后单击“使用其他用户名进行连接”超链接，在弹出的“连接身份”对话框中输入用户名和密码。
- (4) 在“映射网络驱动器”对话框中勾选“登录时重新连接”复选框，以便总是能够自动连接共享资源。
- (5) 单击“完成”按钮，即可连接到 Samba 服务器中的共享资源，最终出现图 22-3 所示的“我的电脑”共享资源窗口。

此后，即使重新启动系统，只要在“我的电脑”或“Windows 资源管理器”中点击相应的共享资源（参见图 22-4），在弹出的“连接到 xxx”对话框中输入用户名和密码，即可快速地连接到远程共享资源，最终出现图 22-5 所示的窗口。

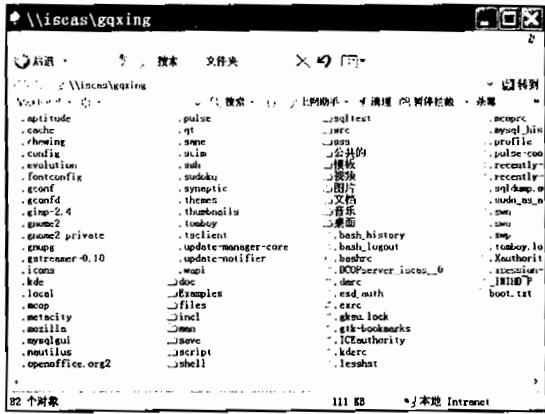


图 22-3 主目录共享资源窗口

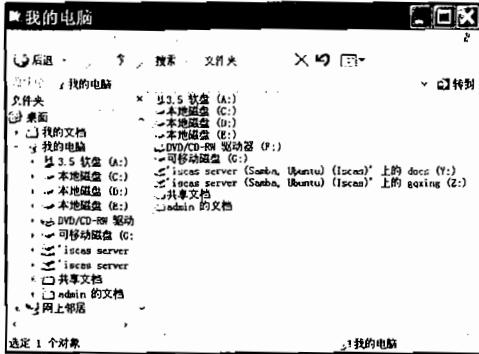


图 22-4 Windows 资源管理器



图 22-5 逻辑驱动器“Z:”窗口

### 22.5.2 从 Linux 系统中访问 Windows 服务器

利用 Samba 的客户端软件 `smbclient`, 可以模拟一个普通 Windows 系统, 访问 Windows 服务器。本小节以 Windows XP 系统中的 CD/DVD 驱动器为例, 说明怎样利用 `smbclient` 命令, 从 Linux 系统中访问 Windows 服务器提供的共享资源。

首先需要设置 Windows 系统。取决于 Windows 系统的版本，其设置过程稍有不同，下面是在 Windows XP 系统中的设置步骤。

- (1) 双击“我的电脑”。
  - (2) 右击“CD/DVD-RW 驱动器”，从上下文菜单中选择“共享和安全”（或“属性”）。
  - (3) 在“DVD/CD-RW 属性—安全”对话框中勾选“在网络上共享这个文件夹”复选框，在“共享名”栏目中设置共享名（如 DVD），最后点击“应用”按钮。

之后，可以使用下列 smbclient 命令，测试 Windows 服务器提供的 CD/DVD 共享资源，其中的 gqxing-xp 是 Windows 系统的 NetBIOS 名字，admin 是 Windows 系统中的用户名。



```
$ smbclient -L gqxing-xp -U admin  
Enter admin's password:  
Domain=[GQXING-XP] OS=[Windows 5.1] Server=[Windows 2000 LAN Manager]  
  
Sharename      Type      Comment  
-----  
E$            Disk      默认共享  
IPC$          IPC       远程 IPC  
D$            Disk      默认共享  
DVD           Disk        
ADMIN$         Disk      远程管理  
C$            Disk      默认共享  
Domain=[GQXING-XP] OS=[Windows 5.1] Server=[Windows 2000 LAN Manager]  
  
Server        Comment  
-----  
  
Workgroup     Master  
-----  
$
```

至此，尚需安装 smbfs 软件包，以便能够把 Windows 系统的共享资源 CD/DVD 作为一个文件系统，安装到 Linux 系统中。另外，还需要利用 mkdir 命令，在 Samba 服务器中创建一个 CD/DVD 驱动器的安装点，以便访问其中的文件。假定这个安装点为/windvd，可以使用下列 mount 命令和 username 选项指定用户的身份，如 admin，把 CD/DVD 驱动器安装到/windvd 目录中。

```
$ sudo mkdir /windvd  
$ sudo mount -t smbfs -o username=admin //gqxing-xp/dvd /windvd  
Password:  
$ ls -l /windvd  
README.diskdefines  casper  install      md5sum.txt  pool      ubuntu  wubi.exe  
autorun.inf        dists   isolinux    pics       preseed  umenu.exe  
$
```

执行上述 mount 命令时，系统还会提示用户输入 Windows XP 系统 admin 用户的密码（注意，此处非 sudo 密码）。也可以利用 password 选项，把密码直接嵌入 mount 命令中，如下所示。

```
$ sudo mount -t smbfs -o username=admin,password=admin,ro //gqxing-xp/dvd /windvd  
$
```

此外，还可以使用 smbfs 软件包提供的 smbmount 命令，安装 Windows 系统的远程共享资源，如下所示。

```
$ sudo smbmount //gqxing-xp/dvd /mnt/windvd -o username=admin,ro  
Password:  
$ ls /windvd  
README.diskdefines  casper  install      md5sum.txt  pool      ubuntu  wubi.exe  
autorun.inf        dists   isolinux    pics       preseed  umenu.exe  
$
```

如果必要，还可以利用/etc/fstab 文件，实现 Windows 系统共享资源的自动安装。为此，可以把下列内容加到 fstab 文件中，如下所示。

```
//gqxing-xp/dvd /windvd smbfs credentials=/etc/windvd,ro 0 0
```

然后创建一个/etc/windvd 文件，按照下列格式，把 Windows XP 系统的用户名与密码加入其中。

```
username = admin  
password = admin
```

一旦完成上述设置，即可达到 Windows 系统共享资源自动安装的目的。也可使用下列 mount 命令，立即安装/etc/fstab 文件中定义的，但目前尚未安装的文件系统，即 Windows XP 系统中的 CD/DVD 驱动器。

```
$ sudo mount -a  
$ ls /windvd  
README.diskdefines  casper  install      md5sum.txt  pool      ubuntu  wubi.exe
```

```
autorun.inf      dists      isolinux      pics      preseed      umenu.exe
$
```

在访问 Samba 或 Windows 服务器之前，如果想要查询当前的 Windows 工作组网络中究竟存在哪些系统，可以使用下列命令（注意，前者只能获取 IP 地址，后者还可以获取系统的 NetBIOS 名字）。

```
$ nmblookup -t workgroup
querying workgroup on 169.254.255.255
169.254.78.56 workgroup<00>
169.254.78.100 workgroup<00>
$ nmblookup -S workgroup
querying workgroup on 169.254.255.255
169.254.78.56 workgroup<00>
Looking up status of 169.254.78.56
    GQXING-XP      <00> -          B <ACTIVE>
    WORKGROUP      <00> - <GROUP>  B <ACTIVE>
    GQXING-XP      <20> -          B <ACTIVE>
    WORKGROUP      <1e> - <GROUP>  B <ACTIVE>

MAC Address = 00-0F-FE-22-29-D2

169.254.78.100 workgroup<00>
Looking up status of 169.254.78.100
    ISCAS          <00> -          B <ACTIVE>
    ISCAS          <03> -          B <ACTIVE>
    ISCAS          <20> -          B <ACTIVE>
    .._MSBROWSE_.  <01> - <GROUP> B <ACTIVE>
    WORKGROUP      <1d> -          B <ACTIVE>
    WORKGROUP      <1e> - <GROUP>  B <ACTIVE>
    WORKGROUP      <00> - <GROUP>  B <ACTIVE>

MAC Address = 00-00-00-00-00-00
```

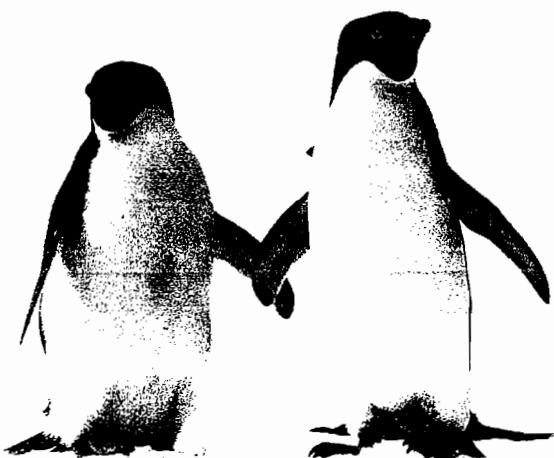


## 第23章

# Apache 服务器

Apache 也许是当今最流行，应用最广泛的 Web 服务器，几乎已成为 UNIX 与 Linux 平台中的标准 Web 服务器。本章主要介绍 Apache 服务器在 Ubuntu Linux 系统中的启动、配置与管理，以及 Apache 配置文件的常用配置指令及其设置方法；讨论怎样创建简单的 CGI 程序、CGI 的安全问题、Apache 的 suexec 机制以及测试 Apache 服务器；还将讨论怎样利用 Apache 的基本用户认证实现密码保护的网页，确保 Apache 服务器的安全运行；解释怎样理解和维护 Apache 服务器的日志文件。其内容主要包括：

- Apache 服务器概述；
- 启动 Apache 服务器；
- 配置 Apache 服务器；
- 用户目录；
- 虚拟主机；
- 利用 CGI 提供动态内容服务；
- 用户认证；
- 日志文件。



## 23.1 Apache 服务器概述

目前，Web 服务器能够提供极其丰富的内容服务。最初，服务器主要提供静态的网页浏览功能，其中绝大部分是文本信息。现在则发展到静态与动态网页共存，且内容服务的范围也越来越广泛。Web 服务器的技术日趋完善，足以实现各种网络应用所需的动态内容服务，如电子商务、流式多媒体应用等，但其底层支持均为 Web 服务器。

作为 Web 服务器，Apache 已成为当今最流行，应用最广泛的 Web 服务器。Apache 几乎是主流 UNIX、Linux 和 BSD 系统（如 FreeBSD）选用的唯一 Web 服务器软件。Apache 一开始就是一个开源项目，不收取任何许可费用，从而能够得到快速的发展。Apache 的源代码在网上完全公开，感兴趣的人可以自由下载，无偿使用；志愿者也可以协助 Apache 项目组，使之更加完善。

Apache 服务器具有下列功能特性。

- 配置——利用配置文件或图形界面，定制服务器的功能特性。
- 静态与动态内容支持——Apache 同时支持静态内容（存储在服务器文件系统中的页面内容）与动态内容。动态内容是由一组相关技术提供的，包括 SSI、CGI、PHP、Perl 及服务器 API 等。
- 模块支持——采用模块化的定制方式，能够动态地扩展 Apache 服务器的功能。软件模块可以直接连接到核心服务器程序，或者根据需要动态加载到服务器中。
- 虚拟主机——采用基于主机名或基于 IP 地址的虚拟主机技术与概念，利用单个或多个 IP 地址同时支持多个网站。
- HTTPS 支持——利用 SSL 支持 HTTPS，采用加密的方式和服务器的标准网络端口 443（而不是通常的 HTTP 端口 80），实现安全的网络连接。
- 日志——Apache 服务器具有完整的日志功能，能够定义信息记录的级别，把客户机的请求和服务器的响应信息详细地记录到日志文件中，使网络管理员能够手工或利用专门的程序分析日志文件，收集服务器的使用统计信息。
- 认证——在访问重要的资源之前，事先需要通过用户名和密码认证。

## 23.2 启动 Apache 服务器

### 23.2.1 Apache 软件包的目录结构

安装 Apache 软件包之后，Apache 的各种组件分布在 Linux 系统的不同目录位置，同时也会创建必要的应用目录。例如，Apache 的配置文件位于 /etc/apache2 目录。图 23-1 展示了在 Linux 系统中安装 Apache 后创建的部分典型目录与文件。

文档根目录 /var/www 是 Apache 对外提供 HTML 文档服务的起始目录位置。Apache 网站应在文档根目录及其子目录中构建对外提供的内容服务。模块文件目录 /usr/lib/apache2/modules 存有



Apache 能够自动加载的各种动态模块文件，可以提供除 Apache 核心功能之外的其他附加功能。日志文件目录 /var/log/apache2 包含日志文件，用于跟踪 Apache 服务器的状态信息，记录服务器（也是浏览器）访问的 HTML 页面与对象，记录服务器产生的任何错误信息。启动文件 /etc/init.d/apache2 是 Apache 服务器的启动脚本，可随系统引导时自动启动，随系统关机时停止运行，以及在修改配置时重新启动 Apache 服务器。当 Apache 守护进程开始运行时，状态文件 /var/run/apache2.pid 包含守护进程 apache2 的进程 ID (PID)，这个 PID 主要供启动脚本停止或重新启动 Apache 时使用。

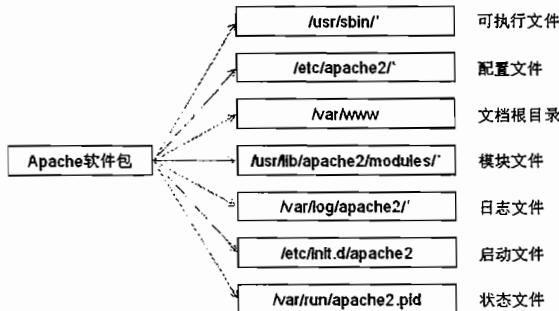


图 23-1 Apache 服务器的目录与文件分布

### 23.2.2 apache2 守护进程

apache2 是 Apache 服务器的守护进程，用于管理、控制和维护 HTTP 网络服务。调用时，apache2 将会创建若干子进程，处理客户机的访问请求。通常，应使用 /etc/init.d/apache2 启动脚本启动 apache2 守护进程，而不应直接调用 apache2 命令。apache2 命令的语法格式简写如下。

```
apache2 [-d serverroot] [-f config] [-e level] [-E file] [-hLMStvX]
          [-k start | restart | stop]
```

表 23-1 给出了 apache2 命令的常用选项及其简单说明。

表 23-1

apache2 命令的常用选项

选 项	简 单 说 明
-d serverroot	用于设置 ServerRoot 配置指令的初始值，指定 Apache 服务器的根目录。配置文件中的 ServerRoot 配置指令能够强制替代这个参数值。在 Ubuntu Linux 系统中，Apache 服务器根目录的默认值是 /etc/apache2
-f config	使用指定的配置文件启动 Apache 服务器。如果给定的配置文件并非绝对路径名，意味着配置文件是相对于 ServerRoot 指定目录的相对路径名。默认的配置文件是 /etc/apache2/apache2.conf
-k start   restart   stop	启动、重启或停止 apache2 守护进程
-e level	设置启动期间 Apache 错误日志文件的信息记录级别 (LogLevel)，其目的是在启动过程中临时增加错误信息的详细程度，以便出错时能够找出原因
-E file	把 Apache 服务器启动过程中出现的错误信息写入指定的文件
-h	输出 apache2 命令的用法，同时列出可用的命令选项及其简单说明
-l	输出已编译并链接到服务器内核的模块列表，但不包括使用 LoadModule 配置指令动态加载的模块
-L	输出配置指令列表，包括配置指令的简单说明、适用的范围及可能的取值等

续表

选 项	简 单 说 明
<b>-M</b>	显示 Apache 服务器已加载的静态与动态（共享）模块列表
<b>-S</b>	显示配置文件的设置（当前仅提供虚拟主机的设置）
<b>-t</b>	执行配置文件的语法检查。语法检查结束之后，立即停止程序的运行。如果语法没有问题，返回 0；否则，返回非 0
<b>-v</b>	显示 apache2 守护进程的版本信息，然后退出
<b>-X</b>	以调试模式运行 apache2 守护进程

### 23.2.3 设置 Apache 启动脚本

在正式启用和部署 Apache 服务器之前，通常需要首先注册一个有效的主机名和 IP 地址，确保把 Apache 服务器的主机名加到 DNS 域名服务器中，以便外部的浏览器能够访问这个网站。

在安装 Ubuntu Linux 系统时，安装程序不会安装 Apache 服务器软件包。为了提供 Apache 服务器功能，可以使用 apt-get、aptitude 或 synaptic 等软件管理工具，安装 Apache 服务器软件包 apache2。示例如下。

```
$ sudo apt-get install apache2
```

使用上述命令安装 apache2 软件包时，将会同时安装 apache2-utils 和 apache2.2-common 等软件包。此时，Apache 服务器已经具备基本的功能，能够提供各种静态内容服务。为了实现动态内容服务，提供联机文档服务，通常还需要单独安装下列软件包。

- apache2-doc —— 其中包含完整的“Apache HTTP Server Version 2.2 Documentation”文档。利用浏览器和“server/manual”网址，可以在本地系统上访问 Apache 文档，server 是 Apache 服务器的域名、IP 地址或 localhost。
- libapache2-mod-perl2 —— 嵌入的 Perl 脚本语言支持功能（mod\_perl）。
- libapache2-mod-python —— 嵌入的 Python 脚本语言支持功能（mod\_python）。
- libapache2-mod-php5 —— PHP 脚本语言（包括 IMAP 和 LDAP 等）支持功能（mod\_php）。

通常，apache2 守护进程使用 LANG=C 作为语言运行环境。通过修改/etc/init.d/apache2 脚本文件中的“ENV=”env -i LANG=C ...”代码，可以设定本地语言环境。

在成功地安装了 Apache 软件包，且使用默认的配置文件启动了 Apache 服务器之后，可以运行 Firefox 浏览器，以 http://localhost 作为 URL，访问 Apache 服务器。如果 Apache 服务器拥有规范的域名，如 beijing.abc.net，也可以在 URL 地址栏中输入 http://beijing.abc.net，此时应出现图 23-2 所示的测试页面。

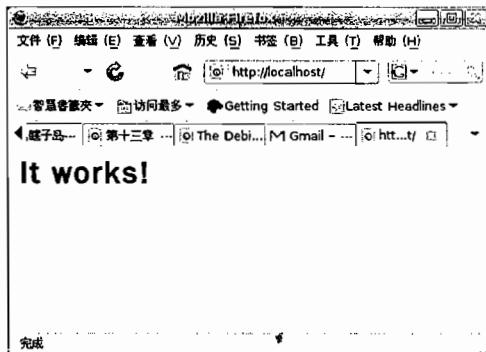


图 23-2 Apache 服务器测试页面



### 23.2.4 Apache 模块

Apache 服务器的核心是一个小而精的引擎，其设计目标是能够快速有效地处理静态 HTML 页面的访问请求。Apache 中的其他功能特性都是采用辅助模块实现的。Apache 辅助模块提供的功能包括用户目录、访问控制、日志、CGI（Common Gateway Interface）程序和目录索引等。

Apache 软件包提供 60 多个软件模块。此外，第三方提供的模块，如其他编程语言支持的模块等，也可以被编译并加载到 Apache 中，从而扩展 Apache 的功能特性。模块可以静态地编译并链接到 Apache 中，也可以在需要时动态加载。要了解哪些模块已经静态地编译到 Apache 服务器中，可以利用 apache2 命令的 “-l” 选项。在 Ubuntu Linux 系统中，“apache2 -l” 命令的典型输出结果如下。

```
$ apache2 -l
Compiled in modules:
core.c
mod_log_config.c
mod_logio.c
worker.c
http_core.c
mod_so.c
$
```

从上述输出信息中可以看出，只有核心与日志模块已经编译到 apache2 中，其他 Apache 模块则存储在 /usr/lib/apache2/modules 目录中。利用 mod\_so.c 模块和 Apache 的模块加载机制，按照 /etc/apache2/mods-enabled/\*.load、/etc/apache2/mods-enabled/\*.conf、/etc/apache2/apache2.conf 及 /etc/apache2/ports.conf 等配置文件中 LoadModule 配置指令的设置，可以在启动 Apache 服务器时动态地加载各种辅助模块，以扩展 Apache 服务器的功能。

## 23.3 配置 Apache 服务器

在 Ubuntu Linux 系统中，Apache 采用 /etc/apache2/apache2.conf 作为主配置文件，定制服务器的功能。随着 Apache 的不断发展，功能特性的不断扩充，网络技术的不断改善，Apache 的配置也日益复杂。因此，把大型文件拆分成较小的文件，然后在主配置文件中使用 “Include” 语句，把所有小型文件从逻辑上汇合成一个完整的配置文件，可以实现系统配置的模块化管理。而且，Apache 提供了合理的默认配置，能够容易地完成 Apache 服务器的安装与配置。

利用 Include 配置指令与 Shell 通配符，还可以增加其他辅助配置文件。apache2.conf 文件采用下列 Include 配置指令，提供一个完整的配置文件。

```
$ grep '^Include' /etc/apache2/apache2.conf
Include /etc/apache2/mods-enabled/*.load
Include /etc/apache2/mods-enabled/*.conf
Include /etc/apache2/httpd.conf
Include /etc/apache2/ports.conf
Include /etc/apache2/conf.d/
Include /etc/apache2/sites-enabled/
$
```

实际上，Apache 主要采用下列配置实现服务器的配置。

- /etc/apache2/apache2.conf——Apache 默认的主配置文件，其中包含全局配置指令，控制 Apache 服务器的整体运行行为。利用 apache2 命令的 “-f” 选项，可以采用不同的配置文件。

- /etc/apache2/envvars——环境变量配置文件，用于定义 Apache 服务器运行时使用的环境变量，如 APACHE\_RUN\_USER 等。当使用/etc/init.d/apache2 脚本启动 Apache 服务器时，将会读取其中的环境变量设置。
- /etc/apache2/htpd.conf——其中存储的配置指令可用于定制全局变量，如超时值（Timeout 等）与各种限制（MaxClients 等）。为了定制任何全局变量，可以把配置指令复制到这个文件中，然后予以修改。
- /etc/apache2/ports.conf——其中包含 Listen 配置指令，用于定义 Apache 服务器监听的 IP 地址与端口。
- /etc/apache2/sites-available——用于配置虚拟主机。对于每个虚拟主机，可以在这个目录中创建单独的配置文件，根据每个虚拟主机的实际情况，增加适当的配置指令，定制自己的运行环境。
- /etc/apache2/sites-enabled——其中含有符号链接文件，用于指向 sites-available 目录中的配置文件。
- /etc/apache2/conf.d——在这个目录中，用户可以创建任何附加的配置文件。
- /etc/apache2/mods-available——主要用于配置需要动态加载的模块。
- /etc/apache2/mods-enabled——其中含有符号链接文件，用于指向 mods-available 目录中的配置文件。

在 Ubuntu Linux 系统中，在定制 Apache 服务器时通常不用需要修改 apache2.conf 文件，必要时只需修改 Include 配置指令指定的文件，其效果等同于修改 apache2.conf 配置文件。在增加或修改配置文件时，请注意 Include 配置指令（即配置文件）的顺序，如果同一配置指令出现在不同的配置文件中，将以最后一个出现的配置指令为准。修改配置文件之后，可以使用下列命令，运行 apache2 启动脚本，重新启动 apache2 守护进程，使新的配置立即生效。

```
$ sudo /etc/init.d/apache2 restart
```

### 23.3.1 Apache 配置文件

在实际应用时，为了提供各种网站服务，通常需要适当地定制 Apache 的配置文件。Apache 根据配置文件中的配置指令设定其运行环境。apache2.conf 是 Apache 默认的主配置文件，位于 /etc/apache2 目录中。利用 apache2 命令的“-f”选项，可以采用不同的配置文件。此外，利用 Include 配置指令与 Shell 通配符，还可以增加其他辅助配置文件。

Apache 的配置文件包括全局变量（如 HTML 文档的根目录）设置、动态共享模块控制、安全设置、文档访问控制、CGI 模块支持、虚拟主机与用户主目录设置以及日志文件的位置与记录格式定义等。

#### 1. 配置文件的语法格式

主配置文件 apache2.conf 是一个的普通文本文件，其中的注释行用于解释和说明 apache2.conf 文件中能够设置或修改的各种配置指令。注释行下面是 Apache 的常用配置指令及其设置。

以井号 “#” 为起始字符的文本行为注释行。除了注释行，Apache 配置文件的每一行只能包含一个配置指令。如果配置指令太长，可以在行尾加反斜线 “\” 延续符转到下一行。配置指令所在行后面不能加注释，空行、配置指令之前的空白字符将被忽略。



在正式启动 Apache 服务器之前，可以使用“apache2 -t”命令，检查配置文件是否存在语法错误。最终，Apache 将会输出语法检测的处理结果（“apache2 -t”命令并不执行/etc/apache2/envvars 文件，故 APACHE\_RUN\_USER 变量为空），如下所示。

```
$ apache2 -t  
apache2: bad user name ${APACHE_RUN_USER}  
$
```

## 2. 配置指令的作用范围

从用途和适用范围来讲，Apache 配置文件中的配置指令可以分为全局配置指令（如 ServerRoot、ServerName、DocumentRoot、User、Group、PidFile、Listen 和 ScriptAlias 等）、容器配置指令（如<Directory>、<DirectoryMatch>、<Files>、<FilesMatch>、<Location> 和 <LocationMatch>等）及虚拟主机配置指令（如 NameVirtualHost）等。

位于主配置文件 apache2.conf 中的全局配置指令适用于整个服务器。为了限定配置指令的适用范围，可以把配置指令置于<Directory>、<DirectoryMatch>、<Files>、<FilesMatch>、<Location> 及<LocationMatch>等配置块中。这些配置块用于限制配置指令适用的特定目录位置或 URL。此外，这些配置块指令还可以嵌套，以便进一步调整服务器的配置。

Apache 能够同时支持多个不同的网站，这种功能称作虚拟主机。虚拟主机由<VirtualHost>配置指令定义，而位于<VirtualHost>配置块中的配置指令，其作用范围也仅限于访问特定的虚拟主机。

## 23.3.2 .htaccess 文件

Apache 采用一种特殊的文件，通过使之分布在不同的树形目录结构中，实现分散的配置管理。这个特殊的配置文件称作 .htaccess，其采用的语法格式与主配置文件完全相同。利用 AccessFileName 配置指令，也可以将其重新命名为其他任何文件名。位于 .htaccess 文件中的配置指令适用于这个文件所在的目录及其所有子目录。针对每一次访问请求，Apache 都会读取 .htaccess 文件，因此对这个文件的任何修改将会立即生效。

## 23.3.3 配置指令

Apache 提供了大量的配置指令，本小节仅对部分常用的重要配置指令做简单的介绍。在介绍容器配置指令时，如果涉及配置块，配置指令前后将会加单书名号，如<Directory>和</Directory>；但在一般的叙述过程中，则省略前后的单书名号，如 Directory。因此，同名的配置指令，不管是否加单书名号，指的是同一个配置指令，特此说明。

### 1. ServerName 配置指令

ServerName 配置指令用于设置 Apache 服务器的主机名（和端口号），使 Apache 服务器能够据此以确定是否访问自己。为了设置主机名和端口号，可在 /etc/apache2/sites-available/default 文件中增加 ServerName 配置指令。例如，如果服务器的主机名为 beijing.abc.net，且还有一个 www.abc.net 别名，如果想让 Apache 服务器监听针对 www.abc.net 和端口 80 的访问请求，可采用下列配置指令（如果未联网，也可以使用 localhost 或 127.0.0.1 作为主机名）。

```
ServerName www.abc.net:80
```

如果未用 ServerName 配置指令指定主机名，Apache 服务器将会尝试使用 IP 地址执行反向地址解析，以获取自己的主机名。如果 ServerName 配置指令中未指定端口，服务器将使用来自访问请求中的端口。

号。为了提高系统的性能，应当使用 `ServerName` 配置指令，明确地设置服务器的主机名与端口号。

如果配置的是基于主机名的虚拟主机，`<VirtualHost>` 配置块中的 `ServerName` 用于匹配访问请求头信息中的主机名。

## 2. ServerRoot 配置指令

`ServerRoot` 配置选项用于设置 Apache 服务器树形目录结构的根目录，其中包含 `conf`、`logs` 和 `modules` 等子目录。服务器的配置文件和日志文件（实为符号链接文件）分别存储在相应的子目录下面。如果其他配置指令（如 `Include`）采用相对路径名，则意味着是相对于这个目录而言的，示例如下。

```
ServerRoot "/etc/apache2"
```

## 3. Listen 配置指令

`Listen` 配置选项的设置使 Apache 能够仅仅监听指定的 IP 地址或端口。默认情况下，Apache 将监听服务器配置的所有 IP 地址及其关联的端口，响应对这些 IP 地址和端口的访问请求。`Listen` 配置指令是一个必要的配置选项，如果不存在，Apache 服务器将无法正常启动。利用 `Listen` 配置选项，可以使 Apache 监听一个除标准端口 80 之外的其他端口。例如，如果另外一个服务器软件已经占用了端口 80，则可以指定其他端口，如下所示。

```
Listen 8080
```

`Listen` 配置选项表示 Apache 服务器仅接受针对指定端口或地址端口组合的访问请求。如果仅指定了端口号，Apache 服务器将会监听其所有网络接口的指定端口。如果同时指定了 IP 地址与端口号，Apache 服务器只需监听指定的网络接口与端口。

如果想要监听多个 IP 地址与端口，可以使用多个 `Listen` 配置指令。Apache 服务器将会响应对其监听的任何地址与端口的访问请求。例如，要让服务器仅接受针对端口 80 和 8000 的访问请求，可以增加下列两个配置指令。

```
Listen 80
Listen 8000
```

要使 Apache 服务器能够接受针对多个指定 IP 地址与端口的访问请求，可以增加下列配置指令。

```
Listen 192.170.2.1:80
Listen 192.170.2.5:8000
```

通常，Apache 采用 HTTPS 协议处理针对端口 443 的访问请求，采用标准的 HTTP 协议处理针对其他所有端口的访问请求。如果其他端口（如 8000）也要求采用加密方式的 HTTP 协议处理，则需要使用下列形式的配置指令特别指定。

```
Listen 192.170.2.1:8000 https
```

## 4. User/Group 配置指令

`User` 与 `Group` 配置指令用于指定用户和用户组的名字（或 ID），表示以哪一个用户或用户组的身份运行 `apache2` 守护进程。这是两个非常重要的配置指令，涉及服务器系统的安全。一个折衷的解决方案是从文件系统中分配一个特定的区域，使 Apache 只能读写其中的文件。因此，通常建议增加一个专用的普通用户和用户组，以普通用户的身份运行 `apache2`。在 Ubuntu Linux 系统中，Apache 的用户与用户组预置为 `www-data`，其配置示例如下（参见 `/etc/apache2/envvars` 文件）。

```
User    ${APACHE_RUN_USER}
Group   ${APACHE_RUN_GROUP}
```

## 5. DocumentRoot 配置指令

`DocumentRoot` 用于设置 Apache 提供的 HTML 文档根目录，以便对外提供网页服务。浏览器指定的 URL 地址就是相对于这个文档根目录的访问地址。例如，如果 Apache 服务器的域名为 `www.abc.net`，`DocumentRoot` 设置为 `/var/www`，把文件 `index.html` 保存到 `/var/www` 目录之后，访问



index.html 文件的 URL 地址就是 `http://www.abc.net/index.html`。下面是 DocumentRoot 的默认设置。

```
DocumentRoot "/var/www"
```

在收到浏览器的访问请求之后，Apache 的默认处理动作是截取 URL 地址除主机名和选用的端口之外的地址部分，然后将其附加到 DocumentRoot 定义的文档根目录之后，作为客户机访问的目的 HTML 文档。因此，位于 DocumentRoot 目录下的所有文件和目录构成了客户机能够访问的树形文档结构。

在使用虚拟主机的情况下，Apache 采用不同的 DocumentRoot 定义，分别提供不同的 HTML 文档。

### 6. Directory 配置指令

`<Directory>`与`</Directory>`用于指定 Apache 服务器能够访问的每个目录及其子目录（即浏览器能够访问的目录），使其中的配置指令的作用范围仅限于指定的目录及其子目录。指定的目录可以是一个绝对路径名，也可以包含 Shell 通配符。其中，问号“?”可以匹配任何一个字符，星号“\*”可以匹配任何字符串，方括号“[]”可以定义一个字符匹配范围。但是，任何通配符都不能匹配斜线字符“/”。因此，`<Directory /home/*/public_html>`能够匹配`/home/user/public_html`，但`<Directory */public_html>`并不匹配`/home/user/public_html`。

在 `apache2.conf` 文件中，每个目录的配置指令均以起始标记符“`<Directory directory_name>`”开始，以结束标记符`</Directory>`表示配置指令的结束。下面是一个取自 `apache2.conf` 文件的 DocumentRoot 目录配置指令的例子。

```
<Directory "/var/www">
    Options Indexes FollowSymLinks MultiViews      # 详见 Options 配置指令
    AllowOverride None                            # 忽略 .htaccess 文件
    Order allow, deny
    Allow from all                                # 设置谁能够访问当前的 Apache 服务器
</Directory>
```

在增加波浪号“~”前缀之后，也可以使用正则表达式指定目录。例如，下列配置指令表示匹配`/www` 目录下以 3 个数字命名的任何子目录。

```
<Directory ~ "^\~/www/.*[0-9]{3}">
```

默认情况下，Apache 服务器赋予`<Directory>`指定目录以“Allow from All”的访问权限，这意味着浏览器客户能够访问 URL 指定的任何文档。因此，建议采用下列方法，修改`<Directory>`配置块，首先锁定指定的目录，然后以此为基础，再适当开放一定的访问权限与限制。

```
<Directory />
    Order Deny, Allow
    Deny from All
</Directory>
```

### 7. DirectoryMatch 配置指令

同`<Directory>`一样，`<DirectoryMatch>`与`</DirectoryMatch>`配置指令用于限定一组配置指令的作用范围仅限于指定的目录及其子目录。但不同的是，后者可以直接使用正则表达式作为参数。例如，下列配置指令表示匹配`/www` 目录下以 3 个数字组成的任何子目录。

```
<DirectoryMatch "^\~/www/(.+)/?[0-9]{3}">
```

### 8. DirectoryIndex 配置选项

这个配置选项用于设置 Apache 服务器网站主页的目录索引文件，该文件通常称作 `index.html`。在安装 Apache 之后，DocumentRoot 定义的目录中将会存在一个默认的 `index.html` 文件。启动 Apache 之后，当利用浏览器，以 `http://localhost` 作为 URL 地址访问网站主页时，Apache 提供的 HTML 文档就是位于 DocumentRoot 目录中的 `index.html` 文件。如果指定多个目录索引文件，Apache 将会返回其依次发现的第一个索引文件。在下列配置例子中，`index.htm` 与 `index.php` 都是合法的主页目录索引文件。

```
DirectoryIndex index.html index.htm index.php
```

### 9. Files 配置指令

同 `<Directory>` 和 `<Location>` 配置指令一样, `Files` 配置指令用于限定一组配置指令在匹配指定的文件时的作用范围。当访问的文件名字(不包括目录)匹配 `Files` 指定的文件时, `<Files>` 配置块中的配置指令才会发挥作用。对 `<Files>` 配置块的处理, 将按照 `<Files>` 配置块在配置文件中出现的顺序——即在 `<Directory>` 配置块和 `.htaccess` 配置文件之后, `<Location>` 配置块之前处理。

`Files` 配置指令指定的文件可以是一个文件名、文件扩展名或包含通配符的字符串。在增加波浪号“~”前缀时, 也可以使用正则表达式。例如, 下列配置指令能够匹配 Internet 中最常用的图像文件。

```
<Files ~ "\.(gif|jpe?g|png)$">
```

注意, 与 `<Directory>` 和 `<Location>` 配置块不同, `<Files>` 配置块可用于 `.htaccess` 文件, 以便用户能够以文件为基础, 控制文件的访问权限。

### 10. FilesMatch 配置指令

`FilesMatch` 配置指令的作用完全等同于 `Files`, 唯一的差别是可以在 `<FilesMatch>` 配置块中直接使用正则表达式。示例如下。

```
<FilesMatch "\.(gif|jpe?g|png)$">
```

### 11. VirtualHost 配置指令

`<VirtualHost>` 与 `</VirtualHost>` 用于限定适用于特定虚拟主机的一组配置指令。当服务器收到访问虚拟主机的文档请求时, 将会使用 `<VirtualHost>` 配置块中的配置指令确定访问的对象。

`VirtualHost` 配置指令的地址参数可以是:

- 虚拟主机的 IP 地址;
- 虚拟主机的规范域名(不建议这样用);
- 星号字符“\*”——仅当 `NameVirtualHost` 也使用星号“\*”作为地址时, 以匹配所有的 IP 地址;
- 字符串“\_default\_”——仅适用于基于 IP 地址的虚拟主机, 以处理 IP 地址与任何虚拟主机都不匹配的情况。

每个虚拟主机必须对应一个不同的 IP 地址、不同的端口号或不同的主机名。下面是一个定义虚拟主机的例子。

```
<VirtualHost 11.22.33.44>
  DocumentRoot /var/www/host1
  ServerName hostname.abc.net
</VirtualHost>
```

### 12. NameVirtualHost 配置指令

在配置基于主机名的虚拟主机时, `NameVirtualHost` 是必不可少的配置指令。尽管可以使用主机名定义虚拟主机名, 但建议总是采用 IP 地址, 以避免导致额外的域名解析过程。利用 `NameVirtualHost` 配置指令, 可以指定一个 IP 地址, 以便 Apache 服务器能够接收对基于主机名的虚拟主机的访问请求。示例如下。

```
NameVirtualHost 11.22.33.44
```

在指定 IP 地址时, 也可以附加一个选用的端口号。此外, 为了接受到达所有网络接口的访问请求, 也可以使用星号“\*”通配符。示例如下。

```
NameVirtualHost 11.22.33.44:8080
NameVirtualHost *
```

在设置基于主机名的虚拟主机时, `VirtualHost` 配置指令定义的地址必须严格匹配



NameVirtualHost 定义的地址。示例如下。

```
NameVirtualHost 11.22.33.44  
<VirtualHost 11.22.33.44>  
....  
</VirtualHost>
```

### 13. Include 配置指令

Include 配置指令用于包括其他配置文件，以增加和整合其他配置指令，或者修改 Apache 服务器的默认配置。指定的配置文件可以是绝对路径名，也可以是相对于 ServerRoot 指定目录的相对路径名。

下面的例子包括一个外部配置文件 httpd-userdir.conf，其中的设置使用户能够在自己主目录的 public\_html 子目录（~/public\_html）中存储 HTML 文件，从而提供个人的网页服务。如果 ServerRoot 设置为 /etc/apache2，则 Include 定义的外部配置文件 httpd-userdir.conf 的完整路径名就是 /etc/apache2/conf.d/httpd-userdir.conf，示例如下。

```
Include conf.d/httpd-userdir.conf
```

Shell 通配符可用于一次包括多个配置文件。此外，如果 Include 指向一个目录，而不是文件，Apache 将会读取指定目录及其子目录中的所有文件，但一般不建议这样做。例如，下列配置指令表示应包括 /etc/apache2/conf.d 目录中以 .conf 为扩展名的所有配置文件。

```
Include conf.d/*.conf
```

### 14. Options 配置指令

Options 配置指令用于控制一个特定目录中可用的服务器特性。Options 配置指令可以设置为 None，表示禁用一切附加的功能特性，也可以设置为下列一个或多个关键字。

- All——表示除 MultiViews 之外的所有选项。这是 Options 配置指令的默认设置。
- ExecCGI——允许利用 mod\_cgi 模块执行 CGI 脚本。
- FollowSymLinks——允许访问目录中的符号链接文件，即使目标文件超出了目录的限定范围。
- Includes——允许执行 mod\_include 模块支持的服务器端的包括（include）功能。
- IncludesNOEXEC——允许执行服务器端的包括（include）功能，但不支持“#exec cmd”与“#exec cgi”功能。
- Indexes——如果 URL 引用的是一个目录，且该目录中没有 DirectoryIndex 配置指令定义的索引文件（如 index.html），mod\_autoindex 模块将会返回目录的格式化文件列表。
- MultiViews——允许使用 mod\_negotiation 模块实现内容协商或内容选择功能，以满足浏览器客户的需求。
- SymLinksIfOwnerMatch——仅当访问的目标文件或目录与符号链接文件的属主相同时，才允许访问符号链接文件。

### 15. LoadModule 配置指令

LoadModule 配置指令用于动态加载辅助的功能模块，以扩展 Apache 服务器的功能。Apache 提供的大量辅助模块，位于 /usr/lib/apache2/modules 目录。例如，为了加载 CGI 支持模块 mod\_cgi.so，可以使用下列配置指令（其中的 modules/mod\_cgi.so 文件相对于 ServerRoot 定义的相对路径名）。

```
LoadModule cgi_module modules/mod_cgi.so
```

### 16. PidFile 配置指令

PidFile 配置指令用于设置 Apache 服务器守护进程 apache2 的进程 ID 文件。如果指定的文件名不是绝对路径名，则是相对于 ServerRoot 指定目录的相对路径名。例如：

```
PidFile /var/run/apache2.pid
```

进程 ID 文件主要用于向守护进程发送信号，以便能够重新启动 apache2，重新读取配置文件，以及关闭并重新打开错误日志文件等。

### 17. ErrorLog 配置指令

ErrorLog 配置指令用于指定 Apache 使用的错误日志文件。如果指定的文件名不是绝对路径名，则是相对于 ServerRoot 指定目录的相对路径名。示例如下。

```
ErrorLog /var/log/apache2/error.log
```

### 18. ErrorDocument 配置指令

在出现问题或错误的情况下，可以配置 Apache，采取下列 4 种处理动作之一：

- 输出一个错误代码和简单的说明信息；
- 输出一个定制的说明信息；
- 重定向到一个本地的 URL 地址，以处理出现的问题或错误；
- 重定向到一个外部的 URL 地址，以处理出现的问题或错误。

第一种是默认的处理动作，第 2~4 种情况需要使用 ErrorDocument 配置指令，采用 HTTP 响应代码与 URL 地址（或一条错误信息）的形式，专门定义。Apache 有时也会提供附加的错误说明信息。

定义的 URL 地址之前可以加一个斜线字符“/”前缀，表示本地路径名（相对于 DocumentRoot），也可以指定一个浏览器客户能够解析的完整 URL 地址，甚至还可以提供一个回送给浏览器的信息，示例如下。

- ErrorDocument 500 http://foo.example.com/cgi-bin/tester
- ErrorDocument 404 /cgi-bin/bad\_urls.pl
- ErrorDocument 401 /subscription\_info.html
- ErrorDocument 403 "Sorry can't allow you access today"

## 23.4 用户目录

在 Linux 系统中，可以使用～user 引用用户 user 的主目录。mod\_userdir 模块也沿用了这一思想，允许浏览器采用下列形式的 URL 访问每个用户主目录中的文档。

```
http://www.example.com/~user/file.html
```

从安全的角度考虑，从浏览器中直接访问用户主目录是不恰当的。因此，可以使用 UserDir 配置指令，指定用户主目录下的某个子目录作为 HTML 文档根目录。如果采用 Userdir 的默认设置 public\_html，上述的 URL 将会访问 /home/user/public\_html/file.html 目录中的文件，其中的 /home/user 是 /etc/passwd 文件中设定的用户主目录。

Apache 服务器支持用户目录的概念，使用户能够在自己的～/public\_html 子目录中提供网页服务。在一个多用户的 Linux 系统中，每个用户都可以利用自己的主目录，设定个人主页。为了实现用户目录，可以在配置文件中增加配置指令。当浏览 http://www.abc.net/～user 网页时，可以访问用户 user 的主目录，或者由 UserDir 配置指令设定的子目录。

用户目录功能是由 mod\_userdir 模块实现的。在 Ubuntu Linux 系统中，利用 Apache 的配置文件 apache2.conf 和 LoadModules 配置指令，在启动 Apache 服务器时，mod\_userdir 模块已动态地加载到系统中。为了启用用户目录，Apache 配置文件中给出了建议的最少设置，以及配置实例（假



定用户的主目录位于/home 目录中), 示例如下。

```
# UserDir: The name of the directory that is appended onto a user's home
# directory if a ~user request is received.
#
UserDir public_html
#
# Control access to UserDir directories.
#
<Directory /home/*/~>
    AllowOverride FileInfo AuthConfig Limit
    Options MultiViews Indexes SymLinksIfOwnerMatch IncludesNoExec
    .....
</Directory>
```

在启用了用户目录功能之后, 通过创建/home/user/public\_html/index.html 文件, 用户 user 就能够建立个人的网络主页。如果运行 Apache 的系统为 www.abc.net, 用户 user 个人主页的 URL 地址就是 http://www.abc.net/~user。

### 23.4.1 利用 UserDir 设定目录路径

UserDir 配置指令用于设定一个路径名 (目录名或模式), 以便把用户主目录的访问请求映射到每一个用户的特定子目录, 其中可以存储每个用户自己的 HTML 文档。

UserDir 配置指令指定的路径名可以采用不同的设置形式。如果路径名的起始字符不是斜线字符 “/”, 则假定指定的目录相对于用户主目录的相对路径。例如, 如果采用下列 UserDir 配置指令设置用户目录, 则 URL http://www.abc.net/~gqxing/file.html 引用的实际文件路径名就是 /home/gqxing/public\_html/file.html。

```
UserDir public_html
```

如果路径名的首字符为斜线字符 “/”, 则使用指定的路径加上指定的用户名组成文件的引用路径。例如, 如果采用下列配置指令设置用户目录, 则 URL http://www.abc.net/~gqxing/file.html 引用的实际文件路径名就是 /var/html/gqxing/file.html。

```
UserDir /var/html
```

如果指定的路径名中包含星号 “\*”, 则引用的实际文件路径名就是利用用户名替换星号 “\*” 后的文件路径名。例如, 在使用下列配置指令设置用户目录之后, URL http://www.abc.net/~gqxing/file.html 引用的实际文件路径名就是 /var/www/gqxing/docs/file.html。

```
UserDir /var/www/*/docs
```

### 23.4.2 限定用户目录的使用

利用 UserDir 配置指令, 还可以限制什么用户能够使用这一功能特性。例如, 下列配置指令表示用户目录适用于除了 disabled 子句中列举的 root、visitor 与 guest 之外的所有用户。

```
UserDir enabled
UserDir disabled root visitor guest
```

反之, 也可以利用 UserDir 配置指令, 赋予少数用户使用这一功能特性的权利, 但禁止其他用户使用。例如, 下列配置指令表示, 除了 enabled 子句中列举的 gqxing 和 cathy 用户之外, 其他用户都不能使用这一功能。

```
UserDir disabled
UserDir enabled gqxing cathy
```

### 23.4.3 开放用户 CGI 目录

为了赋予每个用户拥有自己的 CGI 目录的权利, 可以使用 Directory 配置指令, 把用户主

目录中的某个子目录作为用户自己的 CGI 目录。下列配置指令表示，每个用户主目录（/home/\*）下的 public\_html/cgi-bin 子目录均可用作 CGI 目录。

```
<Directory /home/*/public_html/cgi-bin/>
    Options ExecCGI
    SetHandler cgi-script
</Directory>
```

假定 UserDir 已设置为 public\_html 目录，CGI 目录中包含一个 CGI 程序 example.cgi，使用下列 URL 即可访问这个 CGI 程序。

<http://www.abc.net/~gqxing/cgi-bin/example.cgi>

## 23.5 虚拟主机

Apache 的一个重要特点是引入了虚拟主机的概念。利用虚拟主机技术，一个 Apache 服务器能够同时支持多个网站。例如，在 Linux 系统主机 beijing.abc.net 中，可以把 Apache 服务器配置为 www.abc.net 与 news.abc.net 两个虚拟主机，同时支持 http://www.abc.net 和 http://news.abc.net 两个网站。

虚拟主机的实现方式有两种：一是基于 IP 地址的虚拟主机，二是基于名字的虚拟主机。基于 IP 地址的虚拟主机意味着系统具有多个 IP 地址，每个虚拟主机可以使用不同的 IP 地址，构成一个网站。基于名字的虚拟主机意味着一个 IP 地址能够对应多个不同的主机名。不管采用哪一种实现方式，浏览器用户最终访问的都是同一个物理服务器。

基于 IP 地址的虚拟主机采用 IP 地址确定访问的目的虚拟主机。因此，每个虚拟主机都需要有一个单独的 IP 地址。而基于主机名的虚拟主机依赖于浏览器提供的 HTTP 头信息中的主机名。其优点是，采用基于主机名的虚拟主机技术，不同的服务器能够共享同一个 IP 地址。

基于主机名的虚拟主机相对比较简单，因为只需配置 DNS 服务器，把每个主机名映射到一个正确的 IP 地址，然后配置 Apache HTTP 服务器，使之能够识别不同的主机名即可。基于主机名的虚拟主机也容易解决 IP 地址资源缺乏的问题。因此，除非有特别的理由选用基于 IP 的虚拟主机，通常应当采用基于主机名的虚拟主机。

为了配置虚拟主机，通常至少需要用到下列配置指令：

- VirtualHost;
- NameVirtualHost;
- ServerName;
- DocumentRoot.

在完成虚拟主机的配置之后，如果想要测试配置的正确与否，可以使用下列命令。

```
# /usr/sbin/apache2 -S
```

上述命令将会输出 Apache 解析配置文件的说明，仔细地考察其中的 IP 地址和服务器名有助于发现配置错误。

注意，如果想在一个现有的 Apache 服务器上增加虚拟主机，必须为现有的主机创建一个 <VirtualHost> 配置块，使之作为一个虚拟主机，其中的 ServerName 和 DocumentRoot 与全局 ServerName 和 DocumentRoot 的配置指令应当是相同的。而且，要把这个虚拟主机放在配置文件的最前面，以便其能够作为默认的主机。



### 23.5.1 配置基于主机名的虚拟主机

为了使用基于主机名的虚拟主机，利用单个 IP 地址实现多个网站，必须指定 Apache 服务器接受访问请求的 IP 地址（或选用的端口）。为此，可以使用 NameVirtualHost 配置指令。在通常情况下，可以使用服务器的任何一个或所有 IP 地址配置虚拟主机，因此，可以使用星号“\*”作为 NameVirtualHost 配置指令的参数。如果打算使用多个端口（如运行 SSL），则应当在参数中指明端口，如“\*:80”。注意，在 NameVirtualHost 配置指令中指定了 IP 地址之后，并不意味着服务器能够自动监听给定的 IP 地址。因此，还应增加相应的 Listen 配置指令。此外，这里指定的任何 IP 地址都必须与服务器的网络接口相关联。

下一步是针对每一个虚拟主机创建一个<VirtualHost>配置块。<VirtualHost>配置指令与 NameVirtualHost 配置指令的 IP 地址参数应当是相同的。在每个<VirtualHost>配置块中，最少需要提供一个 ServerName 配置指令（指定服务器的名字）和 DocumentRoot 配置指令（说明虚拟主机文档数据在文件系统中的目录位置）。

假定 Apache 服务器仅配有一个 IP 地址 192.168.90.100，主机名为 beijing.abc.net，但具有两个别名 www.abc.net 与 news.abc.net，均指向同一个服务器（参见第 20 章中 CNAME 资源记录的说明），只需在配置文件中增加下列设置。

```
# Ensure that Apache listens on port 80
Listen 80

# Listen for virtual host requests on all IP addresses
NameVirtualHost *:80

<VirtualHost *:80>
    ServerName www.abc.net
    DocumentRoot /var/www/html
    .....
</VirtualHost>

<VirtualHost *:80>
    ServerName news.abc.net
    DocumentRoot /var/www/news
    .....
</VirtualHost>
```

在上述的 NameVirtualHost 与<VirtualHost>配置指令中，星号“\*”位置也可以给出明确的 IP 地址。例如，假定主机 beijing.abc.net 的 IP 地址为 192.168.90.100，使用同一个 IP 地址实现基于主机名的两个虚拟主机的同样配置如下（参见图 23-3）。

```
NameVirtualHost 192.168.90.100

<VirtualHost 192.168.90.100>
    ServerName www.abc.net
    DocumentRoot /var/www/html
    .....
</VirtualHost>

<VirtualHost 192.168.90.100>
    ServerName news.abc.net
    DocumentRoot /var/www/news
    .....
</VirtualHost>
```

最后，需要仔细地调整虚拟主机的配置，在<VirtualHost>配置块中增加必要的配置指令。大多配置指令均可用于<VirtualHost>配置块，以设定相关虚拟主机的配置属性。仅当虚拟主机中设置的配置指令

## Apache 服务器

没有被强制修改时，主服务器环境中（即<VirtualHost>配置块之外）设置的配置指令才能继续发挥作用。

当收到访问请求时，服务器首先会检测其 IP 地址是否匹配 NameVirtualHost。如果匹配，接着检测匹配 IP 地址的<VirtualHost>配置块，找出匹配访问请求的主机名（ServerName 或 ServerAlias）。如果找到匹配的虚拟主机，则使用该虚拟主机的配置处理访问请求。如果找不到匹配的虚拟主机，则使用匹配 NameVirtualHost 指定 IP 地址之后的第一个虚拟主机作为服务器。

因此，NameVirtualHost 配置指令之后列出的第一个虚拟主机可以看作默认的虚拟主机。

当请求信息中的 IP 地址匹配 NameVirtualHost

配置指令定义的 IP 地址时，肯定不会访问主服务器。如果想要定义一个特别配置，使之能够处理不匹配任何虚拟主机的访问请求，必须把这个配置加到<VirtualHost>配置块的最前面。当请求访问的 IP 地址不匹配 NameVirtualHost 指定的 IP 地址时，将由主服务器负责处理。

当然，不仅需要使用 ServerName 定义主机名，还必须事先注册 DNS 服务器，适当地设置 Apache 服务器每个主机名与 IP 地址的映射关系。在 abc.net 域名服务器中，建立 www.abc.net 和 news.abc.net 两个主机与 beijing.abc.net 主机 IP 地址之间的关联关系。

### 23.5.2 配置基于 IP 地址的虚拟主机

当利用多个 IP 地址实现多个网站时，需要在 Apache 服务器上配置基于 IP 地址的虚拟主机，服务器要求配多个网络接口卡或采用虚拟网络接口，在同一个网卡上设置多个不同的 IP 地址，每个虚拟主机对应一个 IP 地址。

尽管采用的是基于 IP 地址的虚拟主机，但并不要求用户在访问 Apache 服务器时输入 IP 地址。访问虚拟主机的方法仍然是采用域名，这就要求在 DNS 系统中增加一条地址记录，建立主机名与 IP 地址的映射关系，从而能够采用常规的方法访问基于 IP 地址的虚拟主机。

假定 Apache 服务器的主机名为 beijing.abc.net，其 IP 地址为 192.168.90.100，另外两个网络接口的 IP 地址分别为 192.168.90.101 和 192.168.90.102。可以此为基础，建立两个虚拟主机 www.abc.net 和 news.abc.net，其配置如下（参见图 23-4）。

```
Listen 80

# This is the "main" server running on 192.168.90.100

ServerName beijing.abc.net
DocumentRoot /var/www/html

<VirtualHost 192.168.90.101>
    serverName www.abc.net
    DocumentRoot /var/www/html1
</VirtualHost>

<VirtualHost 192.168.90.102>
    serverName news.abc.net
    DocumentRoot /var/www/html2
</VirtualHost>
```

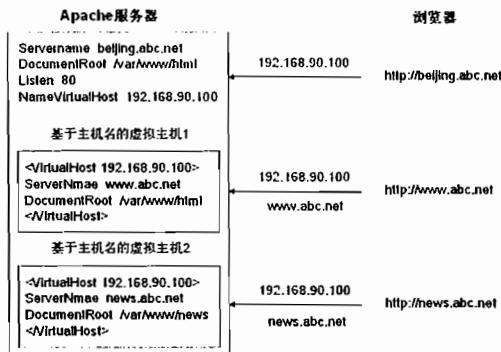


图 23-3 基于主机名的虚拟主机

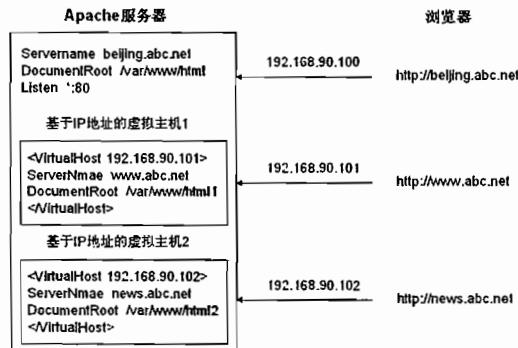


图 23-4 基于 IP 地址的虚拟主机

要访问 Apache 主服务器，可以在浏览器中输入 URL 地址 `http://beijing.abc.net`。访问第一个虚拟主机时，可在浏览器中输入 URL 地址 `http://www.abc.net`，而访问第二个虚拟主机时，可在浏览器中输入 URL 地址 `http://news.abc.net`。当请求访问`<VirtualHost>`配置块中未指定的任何 IP 地址（如 `localhost`）时，将由主服务器负责处理。

### 23.5.3 利用不同的 IP 地址提供相同的网站服务

假定服务器具有两个 IP 地址，分别为 192.168.90.100 和 172.20.30.40。服务器位于内网和外网之间，外部网络的用户可以使用域名 `server.example.com` 访问服务器的外网 IP 地址 172.20.30.40，内部网络的用户可以使用相同的域名访问服务器的内网 IP 地址 192.168.90.100。

采用下列一个`<VirtualHost>`配置块，Apache 服务器能够使用相同的内容响应内部和外部网络用户的访问请求。

```
NameVirtualHost 192.168.90.100
NameVirtualHost 172.20.30.40

<VirtualHost 192.168.90.100 172.20.30.40>
    DocumentRoot /var/www/html/beijing
    ServerName beijing.abc.net
    ServerAlias beijing
</VirtualHost>
```

注意，在内部网络中，用户只能使用主机名 `beijing`，而不能用规范域名 `beijing.abc.net` 访问 Apache 服务器。

### 23.5.4 利用不同的端口提供不同的网站服务

假定有两个域名，但指向同一 IP 地址。为了提供两个不同内容的网站服务，可以利用 `NameVirtualHost` 配置指令定义两个端口，配置两个虚拟主机。注意，如果仅用“`<VirtualHost name:port>`”形式配置两个虚拟主机，而未采用“`NameVirtualHost name:port`”形式定义端口，或者仅用 `Listen` 配置指令定义端口，虚拟主机并不能正常运行。下面是一个可用的配置示例。

```
Listen 80
Listen 8080

NameVirtualHost 172.20.30.40:80
NameVirtualHost 172.20.30.40:8080

<VirtualHost 172.20.30.40:80>
```

```

    ServerName www.example.com
    DocumentRoot /var/www/html80
  </VirtualHost>

  <VirtualHost 172.20.30.40:8080>
    ServerName www.example.org
    DocumentRoot /var/www/html8080
  </VirtualHost>

```

## 23.6 利用 CGI 提供动态内容服务

CGI (Common Gateway Interface) 是 Web 服务器与外部应用之间的标准接口，用于执行与网页相关联的外部程序，使 Web 服务器能够与生成动态内容的外部程序交互，这种程序通常称作 CGI 程序或 CGI 脚本。

CGI 程序是在浏览器访问 Web 服务器时实时执行的，其生成的输出能够动态地成为 Web 服务器提供的 HTML 代码的一部分。在提供动态内容服务方面，CGI 是 Web 服务器中用得最早、最简单也是最常用的方法。

CGI 程序主要用于访问搜索引擎或数据库，解析用户输入网页表格中的信息。本节将介绍怎样在 Apache 中配置 CGI，使之支持 CGI 程序，提供动态内容服务。

### 23.6.1 启用 CGI 程序

为使 CGI 程序能够正常地运行，需要设置 Apache 服务器的配置文件 `apache2.conf`，使之支持 CGI 程序的执行。要达到这一目的，有若干方法可供选用。

#### 1. ScriptAlias 配置指令

在 Apache 服务器中，CGI 脚本的标准目录位置是 `/var/www/cgi-bin`。要采用不同的目录位置存储 CGI 脚本，可以使用 `ScriptAlias` 配置指令。`ScriptAlias` 用于指定一个目录，表示其中存储的是 CGI 程序。Apache 服务器假定此目录中的每个文件都是 CGI 程序。注意，`ScriptAlias` 配置指令通常用于指定位于 `DocumentRoot` 指定目录之外的某个目录。当浏览器请求其中的文件资源时，Apache 将会尝试执行引用的文件。下面是一个取自 `apache2.conf` (`sites-available/default`) 文件中的配置实例。

```
ScriptAlias /cgi-bin/ /usr/lib/cgi-bin/
```

这个配置实例表示 `/var/www/cgi-bin` 目录是 Apache 服务器的 CGI 程序目录。因此，当收到任何以 `/cgi-bin/` 为起始地址的资源访问请求时，Apache 服务器应当检索 `/var/www/cgi-bin` 目录，执行引用的 CGI 程序。

例如，如果浏览器请求访问 URL `http://www.abc.net/cgi-bin/test.pl`，Apache 将会尝试执行 `/var/www/cgi-bin/test.pl` 文件，然后把输出返回浏览器。当然，引用的文件必须存在，且是可执行的，否则 Apache 将会返回一个错误信息。

#### 2. 用户 CGI 程序目录

出于系统安全的考虑，经常把 CGI 程序限制到 `ScriptAlias` 设定的目录。采用这种处理方式时，系统管理员能够控制允许哪个用户使用 CGI 程序。但是，如果事先采取了适当的安全措施，没有理由限制在其他目录中提供可运行的 CGI 程序。例如，如果用户拥有且需要运行自己的 CGI 程序，但又无法访问服务器的 `cgi-bin` 目录时，完全可以利用 `UserDir` 配置指令，让用户在自己的主目录下提供 CGI 程序。



为了达到在任何目录中都能够执行 CGI 程序的目的，需要完成两个步骤：首先，必须使用 AddHandler 或 SetHandler 配置指令启用 cgi-script 处理器；其次，必须在 Options 配置指令中设定 ExecCGI。

#### (1) 使用 Options 启用 CGI 程序执行。

在主服务器配置文件中，使用 Options 配置指令，明确地指定存储 CGI 程序的目录，如下所示。

```
<Directory "/usr/lib/cgi">
    AllowOverride None
    Options +ExecCGI -Multiviews +SymLinksIfOwnerMatch
    Order allow, deny
    Allow from all
</Directory>
```

上述的 Options 配置指令表示 Apache 允许执行 CGI 程序。此外，还需要告诉 Apache 服务器什么文件是 CGI 程序。下列 AddHandler 配置指令表示，Apache 应把具有.cgi 或.pl 扩展名的文件作为 CGI 程序处理。

```
AddHandler cgi-script .cgi .pl
```

#### (2) 用户目录。

为了使用户目录中以.cgi 为扩展名的所有 CGI 程序文件能够执行，至少应当增加下列配置指令。

```
<Directory /home/*/~public_html>
    Options +ExecCGI
    AddHandler cgi-script .cgi
</Directory>
```

如果想要在用户主目录中设置一个 cgi-bin 子目录，使其中的任何文件能够作为 CGI 程序处理，至少应当增加下列配置指令。

```
<Directory /home/*/~public_html/cgi-bin>
    Options ExecCGI
    SetHandler cgi-script
</Directory>
```

#### (3) .htaccess 文件。

针对某个特定的目录，也可以使用 .htaccess 文件允许在相应的目录中执行 CGI 程序。为此，可以使用下列配置指令实现。

```
Options +ExecCGI
AddHandler cgi-script cgi pl
```

此外，如果想把给定目录中的所有文件均看作 CGI 程序，可以使用下列配置指令实现。

```
Options +ExecCGI
SetHandler cgi-script
```

注意，必须在主配置文件中增加“AllowOverride Options”和“AllowOverride FileInfo”配置指令，才能使 .htaccess 文件中的上述配置发生作用。

## 23.6.2 编写 CGI 程序

“普通”程序与 CGI 程序之间存在两个主要的差别：首先，在 CGI 程序的所有输出之前，必须增加一个 MIME 类型的头信息，即 HTTP 头信息，告诉浏览器输出的是什么类型的数据。在大多数情况下，这个头信息如下所示。

```
Content-type: text/html
```

其次，输出信息需要增加能够在浏览器中适当显示数据的 HTML 标记符，也可以包括一些浏览器支持的影像文件及其他非 HTML 内容。

CGI 程序是在 Apache 服务器上运行的。在 Apache 服务器中，可以采用任何语言编写 CGI

程序，常用的编程语言包括 Perl、Python 甚至 C 语言等。

### 1. 利用 Shell 编写 CGI 程序

下面是一个利用 echo 命令与 Shell 编程方法编写的简单 CGI 脚本，这个例子揭示了怎样利用一个外部程序，动态地生成一个 HTML 代码。

```
# cat /usr/lib/cgi-bin/hello.cgi
#!/bin/sh
echo 'Content-type: text/html'
echo
echo "<html><head><title>A small CGI program</title></head>"
echo "<body><center><h1>Hello, World!</h1></center></body></html>"
#
```

在上述 CGI 脚本中，第一个 echo 命令通知请求访问的浏览器，随后输出的数据类型是 text/html。第二个 echo 命令输出一个换行字符。第 3 个 echo 命令中的字符串为“*A small CGI program*”，前后为若干 HTML 标记符，其用意是在浏览器的标题栏中显示“*A small CGI program*”。最后一个 echo 命令中的字符串为“*Hello, World!*”，前后也存在若干 HTML 标记符，目的是在浏览器的主窗口中显示“*Hello, World!*”。

把上述代码写入一个 hello.cgi 文件，并保存在 ScriptAlias 配置指令定义的目录（如 /usr/lib/cgi-bin）中。CGI 程序是由 apache2 进程调用的，故应设置 CGI 脚本文件的访问权限，使运行 apache2 进程的用户（如 www-data）具有可读可执行的访问权限。为此，可使用下列 chmod 命令。

```
# chmod o+rwx hello.cgi
#
```

要调用这个 CGI 脚本，可在 Apache 服务器上启动 Firefox 浏览器，输入 URL 地址 <http://localhost/cgi-bin/hello.cgi>，浏览器窗口中将会出现图 23-5 所示的输出结果。

至此，上述 CGI 测试脚本的运行结果说明 Apache 软件包的安装与设置已经就绪，Apache 服务器的 CGI 功能模块已经可以正常地运行，能够支持更复杂、更实用的 CGI 程序。

### 2. 利用 Perl 编写 CGI 程序

在传统的 Web 应用中，Perl 一直用于开发 CGI 程序。下面是一个利用 Perl 语言编写的 CGI 程序的例子，其作用是在浏览器中显示一行信息。

把这个 CGI 程序保存到 first.pl 文件中，并置于 cgi-bin 程序目录中。

```
$ cat /usr/lib/cgi-bin/test.pl
#!/usr/bin/perl
print "Content-type: text/html\n\n";
print "<html><head><title>A small Perl program</title></head>";
print "<body><center><h1>Testing Perl.</h1></center></body></html>";
$
```

上述第一行语句告诉 Apache，这个程序应送交 /usr/bin/perl 解释执行。第二行输出数据内容类型信息，然后输出两个换行字符，即在头信息之后增加一个空行，表示 HTTP 头信息的结束和数据的开始。第 3 行输出字符串为“*A small Perl program*”，前后为若干 HTML 标记符，其用意是在浏览器的标题栏中显示“*A small Perl program*”。第 4 行输出一个“*Testing Perl.*”字符串，以 Perl 编写的整个 CGI 程序结束。



图 23-5 hello.cgi 在浏览器窗口中的输出结果



如果此时打开浏览器，访问 `http://localhost/cgi-bin/test.pl`，将会看到“Testing Perl.”信息出现在浏览器窗口中，参见图 23-6。

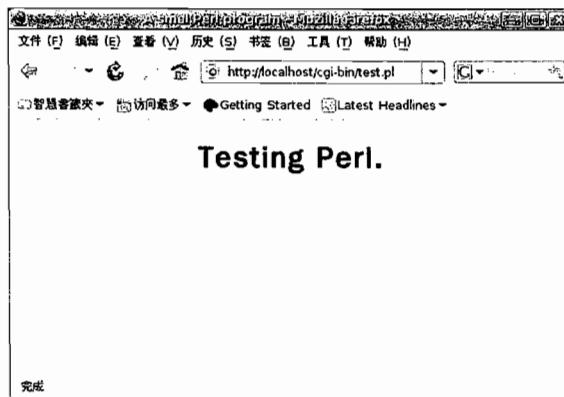


图 23-6 `first.pl` 在浏览器窗口中的输出结果

### 23.6.3 CGI 的安全考虑与 suexec

网络管理员应充分关注 Apache 服务器对 Linux 系统安全的影响，如果配置不当，Apache 服务器会给浏览器用户留下非法访问文件系统的机会，这也就是通常总是建议 Apache 进程应归属于某个普通用户（如 `www-data`）拥有的原因所在。其中尤其要关注 CGI 程序的安全，因为不合理地使用 CGI 程序写入文件系统，访问远程服务器系统的根目录，存在潜在的安全风险。

此外，CGI 程序还存在一个访问安全的问题。正如先前所述，CGI 程序是由 Apache 直接调用执行的，而 Apache 又从属于一个非特权用户 `www-data`。在先前的 `chmod` 示例中，我们把 `hello-world.cgi` 文件设置为任何人都可以读和执行（文件的属主为 `root`），因此 Apache 进程也能读并执行这个 CGI 程序文件。

把所有 CGI 程序均设置成任何人都可以读是有问题的，例如，有些 CGI 程序需要嵌入用户 ID 和密码，有些 CGI 程序需要从 Apache 的子目录中读取文件，这些子目录及其中的文件因而也需要设置为任何人都可以读，在某种情况下，可能还需要设置为任何人都可以写。

因此，最好把 CGI 脚本的属主改为 `www-data`，也就是把其文件属主改为与 Apache 进程的属主一致的用户，使得只有这个用户才具有可读、可执行的访问权限。因此，对于上述 `hello-world.cgi` 示例而言，如果 Apache 进程的属主是 `www-data`，则应以超级用户的身份执行下列命令。

```
$ sudo chown www-data hello.cgi  
$ sudo chmod 700 hello-world.cgi  
$
```

如果 `hello-world.cgi` 还需要访问 Apache 的任何子目录和文件，则还应修改这些子目录和文件的属主与访问权限，使得只有 `www-data` 用户能够访问。

Apache 的 `suexec` 特性能够灵活地实施 CGI 程序的访问控制。`suexec` 特别适用于普通用户使用或测试位于 Apache 用户目录 (`~/public_html`) 中属于自己的 CGI 程序。通常，CGI 程序的运行与 Apache 的 `apache2` 进程具有相同的用户 ID 和访问权限。但利用 `suexec` 功能，Apache 允许 CGI 程序以其属主的用户 ID 运行。

例如，假定用户 gqxing 正在测试一个 Perl CGI 程序 myscript.pl，这个文件保存在主机系统 beijing.abc.net 自己的用户目录中，其路径名为~/public\_html/cgi-bin/myscript.pl。当 Apache 通过 suexec 执行 myscript.pl 时，这个程序将会以 gqxing 的用户 ID，而不是常规的 CGI 用户（如 www-data）的身份运行。

由于 myscript.pl 是以 gqxing 的用户身份运行的，因而能够访问 gqxing 拥有的任何文件和目录。因此，不需要把这些文件和目录的访问权限设置为任何人都可读、可写，从而增强了安全。在运行 CGI 程序之前，suexec 也会对 CGI 程序执行安全检查。

此外还应注意，为了使普通用户（如 gqxing）能够在~/public\_html/cgi-bin 目录之外使用 Apache 运行 CGI 程序，必须把类似于下列示例的<Directory>配置选项加到 apache2.conf 文件中。

```
<Directory "/home/gqxing/public_html/cgi-bin">
    Options +ExecCGI
    SetHandler cgi-script
</Directory>
```

在 www.abc.net 系统中的 Apache 重新启动之后，gqxing 能够使用 Web 浏览器，以访问 URL 地址 http://www.abc.net/~gqxing/cgi-bin/myscript.pl 的方式，测试其 myscript.pl 脚本。

### 23.6.4 Apache 与 LAMP

目前，Apache 已成为网站应用开发平台 LAMP（Linux、Apache、MySQL 和 Perl/Python/PHP 的缩写）的一个重要组成部分。在 LAMP 应用中，MySQL 数据库管理系统能够为 LAMP 应用提供后台数据存储支持，Perl/Python/PHP 用于编写 CGI 程序，访问存储在 MySQL 后台数据库中的数据，为浏览器用户访问 Apache 服务器提供动态内容服务。

在 LAMP 应用中，广泛使用的语言是 PHP (<http://www.php.net>)。与 Perl 或 Python 语言不同的是，PHP 在研发时就充分考虑了网站应用的特点。PHP 最初的设计目标是与 Web 服务器一起使用，作为其中的一个过滤器，读取包含 PHP 语言代码的文件，然后转换成 HTML 文档，以便在浏览器中显示动态内容。

Apache 利用 php5\_module 模块解释嵌入 HTML 文档中的 PHP 语言代码，而不是通过 CGI 运行 PHP 程序。为了支持 PHP 语言，应确保 Apache 配置文件中存在下列 LoadModule 配置指令，以便能够在启动 Apache 服务器时动态加载 PHP 模块（在 Ubuntu Linux 系统中，这一设置是在 /etc/apache2/conf.d/php.conf 配置文件中实现的）。

```
LoadModule php5_module modules/libphp5.so
```

另外，还要确保 Apache 配置文件中存在下列配置指令，使之能够根据文件扩展名.php 解释 PHP 文件，示例如下。

```
AddHandler php5-script .php
AddType text/html .php
```

在启动并加载了 PHP 模块之后，Apache 能够解释执行嵌入 HTML 文档、文件扩展名为.php 的 PHP 程序。下面是一个简单的 PHP 示例文件，其作用是在浏览器窗口中输出一个“Hello PHP!”信息，然后调用 phpinfo() 函数，输出 PHP 的配置信息。

```
<html>
  <head>
    <title>A little PHP program</title>
  </head>
  <body>
    <?php
```



```
print '<center><h1>Hello PHP!</h1></center>';
phpinfo();
?>
</body>
</html>
```

如果把上述 HTML/PHP 代码保存到 Apache 文档根目录下面的一个 info.php 文件中，当利用浏览器访问 Apache 时，能够看到图 23-7 所示的输出结果。这表示，作为 Apache 的一个动态加载模块，PHP 已经就绪，Apache 服务器能够支持基于 PHP 的、使用 MySQL 作为后台数据库支持的各种网站应用。

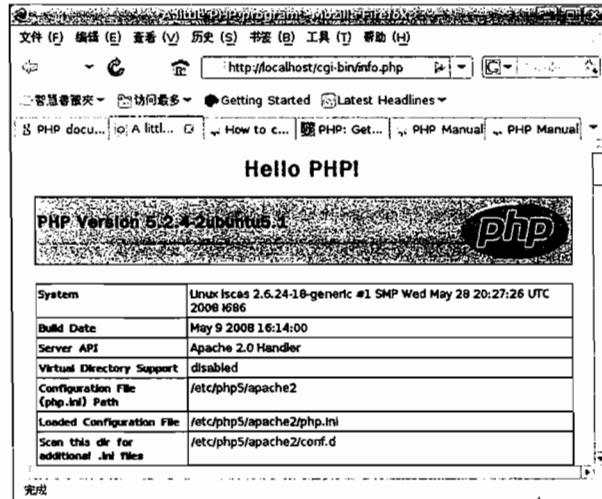


图 23-7 phpinfo()生成的 PHP 配置信息

## 23.7 用户认证

如果 Apache 服务器提供的信息比较敏感，只能供部分用户访问，可以采用用户认证技术保护资源。Apache 提供的用户认证技术，能够确保只有限定的用户才能浏览保密的数据。除了用户认证之外，还可以考虑使用加密的 HTTPS 协议与 mod\_ssl 模块，确保数据的安全。本节仅介绍 Apache 提供的基本用户认证技术。

### 23.7.1 用户认证的实现

Apache 提供的基本认证方法能够根据用户名和密码验证用户访问的合法性，对选定的目录或网页实行简单的密码保护措施。为了实现基本的用户认证，需要设置 Apache 服务器的虚拟主机配置文件 /etc/apache2/site-available/default 或基于目录的配置文件 .htaccess。

在设置密码保护的文件访问时，可以把有关的配置指令加到主配置文件的 <Directory> 配置块中，也可以创建基于目录的配置文件 .htaccess。如果使用 .htaccess 配置文件，还需要修改上述的 default 配置文件，在 /var/www 目录配置中注销 “AllowOverride None” 一行，同时增加下列配置指令，表示允许在 .htaccess 配置文件中使用认证配置指令（AllowOverride 配置指令的作用就是指

定 .htaccess 配置文件中能够使用什么配置指令)。

```
AllowOverride AuthConfig
```

假定文档根目录下的 docs 目录(即 /var/www/docs 目录)存有敏感的数据文件, 只能供 gqxing 一个用户访问, 在此情况下, 可以采用基于目录的 .htaccess 配置文件, 对指定的目录实施密码保护。

要采用基本认证技术, 最简单的方法是创建两个隐藏的文本文件: 第一个是 .htaccess 配置文件, 存放在限制访问的 /var/www/docs 目录中; 第二个是 .passwords 密码文件, 用于存储用户名与密码信息(假定这个文件存放在 /usr/lib/apache2 目录中)。

### 1. 设置 .htaccess 配置文件

在创建 .htaccess 文件时, 需要告诉 Apache 服务器采用哪一种用户认证技术, 采用什么方式存储用户密码信息和密码文件的存储位置, 以及允许哪一个用户访问这个目录, 等等。基于这个思路, 可以写出下列配置指令。

```
AuthType Basic
AuthBasicProvider file                                     # 这个配置指令是可选的
AuthUserFile /usr/lib/apache2/.passwords
AuthName "Restricted directory! Please login."
Require user gqxing
```

在上述配置指令中, AuthType 配置指令用于选择用户认证方法, Basic 表示采用最基本认证(由 mod\_auth\_basic 模块实现)。但是, 采用基本认证方法时, 浏览器发送到服务器的密码并未加密。因此, 高敏感的数据不应采用这种认证方法, 除非是在 HTTPS 协议(由 mod\_ssl 模块实现)的基础上。Apache 还支持另外一种更安全的认证方法, 即 HTTP 摘要认证(HTTP Digest Authentication)。必要时, 可以选择“AuthType Digest”(由 mod\_auth\_digest 模块实现)。目前, 并非所有的浏览器都支持摘要认证, 故这里不再赘述。

AuthName 配置指令用于设置认证提示信息。当用户请求访问 /var/www/html/docs 目录时, 一个包括指定信息的认证对话框将会出现在用户的浏览器中, 提示用户输入密码。

AuthBasicProvider 配置指令用于设置采用哪一种方式提供用户认证功能, file 表示利用文本文件存储用户名和密码等认证信息。AuthBasicProvider 配置指令是可选的, 因为其默认值就是文件。如果用户数过多, 认证每个用户时可能都需要检索较大的文本文件, 因而会影响服务器的性能。Apache 也支持其他选用的存储方法, 如把用户信息存储到较快的数据库文件或 LDAP 等。

AuthUserFile 配置指令用于设置由 htpasswd 命令创建的用户密码文件的路径名。

最后的 Require 配置指令用于指定经过认证后能够访问相关资源的用户。“user gqxing”表示只有 gqxing 名义的用户才能访问 .htaccess 配置文件限定的相关资源。

### 2. 创建用户密码文件

其次, 需要创建用户密码文件, 即 .htaccess 配置文件中指定的 .passwords 文件。用户密码文件应存放在浏览器用户访问不到的目录位置。Apache 软件包提供的 htpasswd 命令可用于生成 .htpasswd 文件。为了创建针对这个例子所需的 .htpasswd 文件, 可以输入下列命令(其中的“-c”选项表示创建新的密码文件。如果指定的文件存在, 将会覆盖其中原有的数据)。

```
# htpasswd -c /usr/lib/apache2/.htpasswd gqxing
```

当运行上述命令时, 系统将会提示用户输入密码, 并采用默认的 crypt 函数对密码加密, 然后把密码写入新建的 .htpasswd 文件。在设置限制访问的目录以及 .htaccess 和 .htpasswd 文件的访问权限时, 必须保证 Apache 的 apache2 进程能够读取这些目录和文件, 而这通常意味着要保证用户 www-data 具有可读的访问权限。



应当妥善保管用户密码文件，将其存储在浏览器用户无法访问的目录位置，如/usr/lib/apache2 目录。为了创建/usr/lib/apache2/.passwords 密码文件，可以输入 Apache 提供的 htpasswd 命令，然后在命令的提示下输入密码，示例如下。

```
$ sudo htpasswd -c /usr/lib/apache2/.passwords gqxing
New password: *****
Re-type new password: *****
Adding password for user gqxing
$
```

如果上述的基本用户认证设置一切都正确，当访问限制的/var/www/html/docs 目录时，浏览器中将会弹出图 23-8 所示的用户认证窗口。

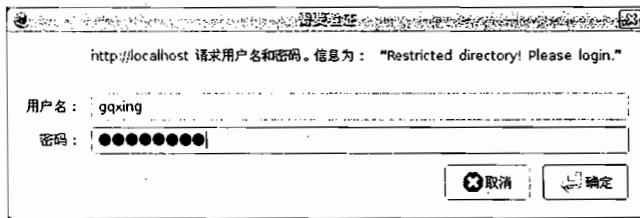


图 23-8 Apache 的用户认证窗口

### 23.7.2 用户认证方法的补充说明

#### 1. 允许多个用户访问

在先前的 Required 配置指令示例中，只有一个用户 gqxing 能够访问限定的目录。而在大多数情况下，可能存在一组用户，都需要访问限定的资源，这就需要用到 AuthGroupFile 配置指令，创建一个用户组文件.groups。AuthGroupFile 配置指令用于指定一个用户组文件，其中存储了用户组与一组用户的映射关系，使用一个用户组名能够关联一组用户。用户组文件的格式相当简单，可以使用任何文本编辑器直接创建。用户组文件的内容示例如下。

```
DevGroup: gqxing cathy hwang
```

上述示例定义了一个用户组 DevGroup，其中包含 3 个用户。用户组名之后需要附加一个冒号 `:`，用户名之间以空格作为分隔符。要把用户名及其密码加到现有的密码文件中，仍需使用如下所示的 htpasswd 命令。

```
# htpasswd /usr/lib/apache2/.passwords gqxing
```

同样，系统将会提示输入用户密码。在把所有的用户名及其密码信息加到密码文件之后，需要设置 .htaccess 配置文件，示例如下。

```
AuthType Basic
AuthBasicProvider file
AuthUserFile /usr/lib/apache2/.passwords
AuthGroupFile /usr/lib/apache2/.groups
AuthName "By Invitation Only"
Require group DevGroup
```

现在，位于用户组 DevGroup 中的任何用户都可以利用用户名和密码，访问限定的资源。

实际上还有另外一种方法，无需建立用户组文件，也能够允许多个用户访问限定的资源，其实现的方法是换用下列配置指令。

```
Require valid-user
```

采用上述配置指令之后，密码文件中列举的任何用户，只要能够通过用户名与密码认证，都

可以访问限定的资源。`valid-user` 只是 `Require` 配置指令中的一个关键字，其作用相当于定义了一个用户组，其成员就是位于相应密码文件中的所有用户。因此，使用多个密码文件，可以仿真多个用户组。采用 `valid-user` 方法的优点是 Apache 只须检索一个用户组文件（`valid-user`），缺点是需要维护多个密码文件，并记住每个密码文件与 `AuthUserFile` 配置指令的对应关系。

## 2. 其他密码存储方法

随着用户数量的增多，采用普通文本文件存储用户名和密码信息，将会降低文件的检索速度，影响服务器的性能。因此，可以考虑把用户名与密码存储在其他位置，如数据库中。

要使用数据库文件存储用户名和密码信息，可以利用 `mod_authn_dbm` 和 `mod_authn_dbd` 两个认证支持模块，在设置 `AuthBasicProvider` 配置指令时，使用 `dbm` 或 `dbd` 替换 `file`。例如，要采用 DBM 格式的文件存储用户名和密码信息，可以使用下列配置指令。

```
<Directory /var/www/docs/private>
    AuthName "Private"
    AuthType Basic
    AuthBasicProvider dbm
    AuthDBMUserFile /usr/lib/apache2/passwd.dbm
    Require valid-user
</Directory>
```

## 23.8 日志文件

Apache 服务器提供了完善和灵活的日志功能，一个正常运行的 Apache 网站，通常会产生大量的日志。Apache 日志能够揭示 Apache 运行时产生的任何错误，包括 Apache 配置中可能存在的安全问题、Apache 使用的网络宽带及其他有用的信息。为了有效地管理 Apache 服务器，了解服务器的处理活动、性能以及可能出现的任何问题，网络管理员应当理解日志文件中记录的信息，并采取适当的管理和维护措施。本节主要介绍怎样配置 Apache 的日志功能，以及怎样解释日志文件中的信息。

在 Ubuntu Linux 系统中，Apache 的日志文件位于 `/var/log/apache2` 目录中。通常，Apache 提供 `access.log` 和 `error.log` 两个主要日志文件。其中最大的是 `access.log` 日志文件，其中包含访问 Apache 守护进程 `apache2` 的所有 HTML 文档和对象信息、HTTP 请求的类型以及每个访问请求的 HTTP 状态代码等。`error.log` 文件包含 Apache 产生的错误信息，如访问过限制访问的页面或对象的 HTTP 请求。`access.log` 和 `error.log` 文件均包含发送 HTTP 请求、访问 `apache2` 的远程系统的 IP 地址以及 HTTP 请求的日期和时间等信息。下面是一条取自 `access.log` 文件的日志信息。

```
172.16.3.40 - - [16/Nov/2008:15:00:40 +0800] "GET /docs/ HTTP/1.1" 200 720...
```

上述日志信息表明，IP 地址为 172.16.3.40 的远程客户，在 2008 年 11 月 16 日的下午 3:00，曾采用 HTTP 协议，以 GET 方法，请求访问 Apache 文档根目录下的 docs 子目录。`apache2` 返回的状态代码 200 表示已成功地给予响应，720 是整个数据传输的累计字节数。

Apache 之所以记录远程客户系统的 IP 地址而非主机名，是为了避免花费大量的时间执行域名地址解析，把每个 IP 地址转换成相应的主机名，从而大大降低 Apache 服务器的性能。

Apache 包括一个 `logresolve` 命令，可以在脱机方式下把 IP 地址转换为主机名。下面是一个例子，说明怎样利用 `logresolve` 命令和 `access.log` 文件，创建一个把日志文件中的 IP 地址转换成主机名的新文件。

```
# logresolve < /var/log/apache2/access.log > /tmp/access.log.hostnames
```



### 23.8.1 错误日志文件

Apache 服务器的错误日志文件 (Error Log) 的名字与位置是由 ErrorLog 配置指令设定的。在 Ubuntu Linux 系统中，其错误日志文件是 `error.log`，位于 `/var/log/apache2` 目录中。通过设置配置文件，也可以采用用户指定的文件，或者让服务器把错误信息发送到 `syslog`，甚至通过管道提交给其他程序处理。

错误日志文件是最重要的日志文件。Apache 守护进程 `apache2` 在处理访问请求时遇到的任何诊断信息和错误信息均被记录到这个日志文件中。当服务器启动或服务器操作出现问题时，错误日志文件是第一个需要考察的地方，因为其中经常包含何处出错、出错的原因以及怎样修正错误等比较详细的信息。

相对而言，错误日志文件的格式比较随意，而且是自描述的，但大多数错误日志项的记录格式都具有一定的共性。例如，下面是一个典型的错误信息，表示远程客户访问的 Perl 程序由于访问权限的问题而拒绝执行。

```
[Sun Nov 16 13:52:40 2008] [error] [client 172.16.3.40] (13)Permission denied:  
exec of '/var/www/cgi-bin/first.pl' failed
```

下面是一条取自 `error.log` 文件的日志信息，表示 IP 地址为 172.16.3.40 的远程客户曾访问过一个不存在的目录。

```
[Sun Nov 16 13:52:40 2008] [error] [client 172.16.3.40] File does not exist:  
'/var/www/html/intro.html', referer: http://beijing/
```

错误日志文件记录的第一项是信息的登记日期和时间。第二项是表示错误严重程度的等级。LogLevel 配置指令用于定义错误日志文件的错误等级，以便控制究竟存储哪些类型的错误信息。第 3 项是导致错误产生的浏览器客户的 IP 地址。从第 4 项开始是错误信息本身。这个例子表示服务器的现有配置拒绝浏览器客户访问列举的文档。

错误日志文件包括 Apache 运行时遇到的所有错误信息。错误信息的格式大多如上例所示，同时也包括 CGI 脚本输出的调试信息，CGI 脚本写到标准错误输出的任何信息将会直接被复制到错误日志文件中。

为了控制写入错误日志文件的错误信息，可以使用 LogLevel 配置指令。LogLevel 配置指令用于定义错误日志文件记录的数据类型，调整记录错误信息的详细程度。Apache 守护进程 `apache2` 支持如表 23-2 所示的日志数据分类级别（按重要性降序排列）。

表 23-2

日志数据分类级别

级 别	描 述	举 例
emerg	非常紧急（系统已不可用）	“Child cannot open lock file. Exiting”
alert	必须立即采取措施	“getpwuid: couldn't determine user name from uid”
crit	严重错误	“socket: Failed to get a socket, exiting child”
error	错误信息	“Premature end of script headers”
warn	警告信息	“child process 1234 did not exit, sending another SIGHUP”
notice	常见的信息	“httpd: caught SIGBUS, attempting to dump core in ...”
info	一般信息	“Server seems busy, (you may need to increase StartServers, or Min/MaxSpareServers)...”
debug	调试信息	“Opening config file ...”

当指定一个特定的级别时，意味着服务器将会记录指定级别及其以上所有重要级别的错误信息。例

如，当使用“LogLevel info”配置指令时，属于 notice、warn 和 error 等级别的信息也将被记录其中。

通常，建议的最少错误日志级别是 crit。示例如下。

```
LogLevel crit
```

注意，错误日志文件无法通过增删信息的方式定制，但错误日志项涉及的访问请求也可从访问日志文件中找到相应的记录。由于访问日志文件能够定制，故可以利用定制的访问日志文件获取服务器错误的更多参考信息。

在测试 Apache 服务器期间，使用下列 tail 命令连续地监控错误日志文件，有时可能是非常有用的，可以借此发现存在的问题。

```
$ tail -f error.log
```

## 23.8.2 访问日志文件

Apache 服务器的访问日志（Access Log）文件用于记录服务器处理的所有访问请求。访问日志文件的位置与记录内容由 CustomLog 配置指令控制。在 Ubuntu Linux 系统中，其访问日志文件 access.log 位于 /var/log/apache2 目录中。

access.log 文件的记录格式是可以配置的。用户可以使用 LogFormat 配置指令简化或选择记录的日志内容，以便分析其中的信息，最终得出有用的统计结果。要定制日志文件记录项的内容，可以使用格式字符串（类似于 C 语言 printf() 函数的格式字符串）定义 access.log 日志文件的记录格式，参见表 23-3。

表 23-3 access.log 日志文件格式定义字符串

格式字符串	简单说明
%%	百分号“%”本身
%a	远程 IP 地址
%A	本地 IP 地址
%B	响应信息的字节数，不包括 HTTP 头信息
%b	响应信息的字节数，不包括 HTTP 头信息。如果响应信息的字节数为 0，采用默认的访问日志格式时将会记录一个减号字符“-”而不是 0
%D	服务于当前的访问请求时花费的时间（微秒）
%f	文件名
%h	远程主机
%H	请求协议
%l	远程用户注册名
%m	请求方法
%p	服务器支持访问请求采用的规范端口
%P	为访问请求提供服务的子进程 ID
%q	查询字符串（如果存在查询字符串，在前面增加一个问号“？”；否则，为空串）
%r	请求信息的第一行数据内容
%s	状态信息
%t	收到请求信息的时间（采用标准的美式时间格式）



续表

格式字符串	简单说明
%{format}t	按照指定的格式表示时间，参见 <code>strftime(3)</code>
%T	处理访问请求时花费的时间（秒）
%u	远程用户名
%U	请求访问的 URL 路径，但不包括任何查询字符串
%v	为访问请求提供服务的服务器的规范名字（ <code>ServerName</code> ）
%V	按照 <code>UseCanonicalName</code> 设置给出的服务器名
%X	完成响应时的连接状态： x 在完成响应之前连接断开 + 发送响应信息后仍保持连接 - 发送响应信息后断开连接
%l	收到的字节数，包括请求信息和头信息，因而不能为 0。使用这个格式定义时需要启用 <code>mod_logio</code> 模块
%O	发送的字节数，包括头信息，因而不能为 0。使用这个格式定义时需要启用 <code>mod_logio</code> 模块

`access.log` 访问日志文件采用的典型记录格式是由下列配置指令定义的。

```
LogFormat "%h %l %u %t \"%r\" %>s %b" common
CustomLog logs/access_log common
```

上述 `LogFormat` 配置指令定义了一个日志文件格式字符串别名 `common`，`CustomLog` 配置指令使用这个别名定义日志文件的记录格式。每个格式字符串均以百分号“%”为起始字符，用于告诉服务器需要记录哪一个特定的信息。格式字符串中也可以包含文字字符，这些文字信息将会被直接写到日志文件中。如果文字信息中存在双引号“”，双引号前必须加转义符号“\”，以防止把双引号解释作格式字符串的结束。格式字符串中也可以包含特殊的控制字符，如换行符“\n”和制表符“\t”等。

`CustomLog` 配置指令使用已定义的别名 `common` 设定新的日志文件需要记录的数据内容。其中指定的访问日志文件名是相对于 `ServerRoot` 指定目录的相对路径，除非路径名以斜线字符“/”为起始字符。

下面是一个取自 `access.log` 日志文件的记录项，其中的信息是按照上述配置指令定义的格式生成的。

```
172.16.3.40 - guest [16/Nov/2008:15:18:40 +0800] "GET /docs/ HTTP/1.1" 200 720...
```

表 23-4 描述了 `access.log` 日志文件记录项每个字段的意义。

表 23-4

access.log 日志文件记录项

字段	意义
172.16.3.40 (%h)	访问服务器的客户系统的 IP 地址。如果把 <code>HostnameLookups</code> 设置为 <code>On</code> ，服务器将会尝试确定客户系统的主机名，并以主机名替代 IP 地址。但通常不建议采用这种配置，因为这将大大降低服务器的性能。最好的做法是利用日志的事后处理工具，如 <code>logresolve</code> ，确定客户系统的主机名。这里记录的 IP 地址并不一定是客户系统的实际 IP 地址，如果用户与服务器之间存在代理服务器，这个地址将是代理服务器的 IP 地址，而不是客户系统的 IP 地址。
(%)	减号“-”表示服务器无法提供用户请求的信息

续表

字 段	意 义
guest (%u)	HTTP 认证机制确定的远程用户名。如果返回的状态码是 401，确定的结果并不可靠，因为并未经过认证。如果请求访问的资源并非密码保护的，这个记录项为减号字符“-”
[16/Nov/2008:15:18:40 +0800] (%t)	收到访问请求的时间。时间的表示格式如下。 [day/month/year:hour:minute:second zone] 其中 day = 2 位数字； month = 3 位字符； year = 4 位数字； hour = 2 位数字； minute = 2 位数字； second = 2 位数字。 zone = (+ -) 4 位数字 也可以利用格式字符串%{format}t，按照 strftime(3) 的格式定义，以不同的形式显示时间
"GET /docs/ HTTP/1.1" ("%r")	双引号中是浏览器客户请求信息中的第一行数据内容。其中包含大量有用的信息：首先是浏览器使用的方法（如 GET），其次是浏览器请求的资源，如/docs 目录，第三是使用的协议，如 HTTP/1.1。如果需要，也可以独立地记录访问请求中的其他数据信息。例如，如果采用格式字符串 "%m %U %q %H"，将会记录方法、路径名、查询字符串和协议，其结果与 "%r" 完全相同
200 (%>s)	服务器回送浏览器的状态代码。这个信息很有价值，从中可以判断用户请求的信息是否已成功地得到服务器的响应（编码以 2 开始），是否已经重定向（编码以 3 开始），是否因客户机的原因出现错误（编码以 4 开始），或者是否因服务器的原因出现错误（编码以 5 开始）。完整的状态代码列表见 HTTP 规范（RFC2616 第 10 节）
720 (%b)	最后一项是服务器返回浏览器的数据大小，不包括头信息。如果返回客户机的内容为空，这个值将是一个减号“-”。要在内容为空时能够记录 0 而非减号“-”，可使用%B

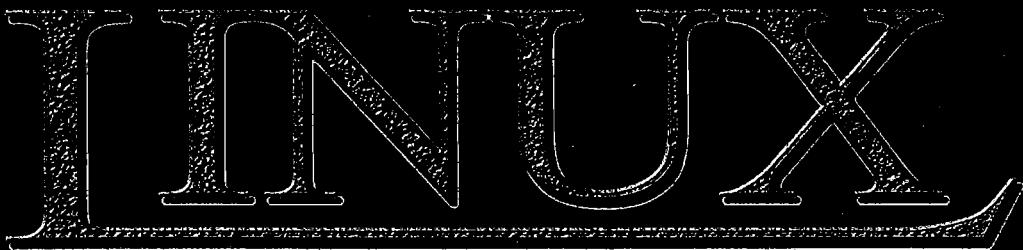
### 23.8.3 虚拟主机日志

当服务器支持多个虚拟主机时，可以针对虚拟主机定制日志文件。首先，把日志文件配置指令置于主配置文件的<VirtualHost>配置块之外，使整个 Apache 服务器及所有虚拟主机均使用同一日志文件，如 access.log 和 error.log。但是，这种日志记录方式无法以虚拟主机为单位收集统计信息。

如果把 CustomLog 或 ErrorLog 配置指令放到<VirtualHost>配置块之内，则针对相应虚拟主机的访问或错误日志将会被记录到单独的日志文件中。当虚拟主机的数量较少时，这种日志记录方式非常有用；但如果虚拟主机的数量较多，这将会增加管理与维护的困难，且有时会引起文件描述符不足的潜在问题。

对于访问日志而言，可以采用一个非常好的折衷解决方案，即利用格式字符串，增加虚拟主机日志信息，把虚拟主机的访问信息记录到同一类日志文件中，之后再把日志文件按虚拟主机分解成单独的文件。例如，采用下列日志格式配置指令（其中的%v 表示记录虚拟主机的名字 comonvhhost），从而能够在将来使用一定的工具分解 access.log 日志文件。

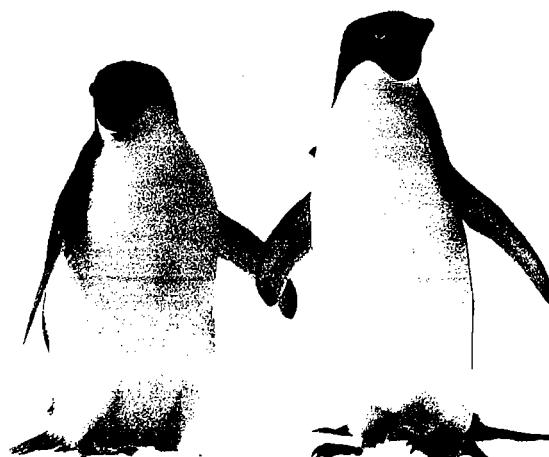
```
LogFormat "%v %l %u %t \"%r\" %>s %b" comonvhhost
CustomLog logs/access_log comonvhhost
```



## 第24章 MySQL 数据库

MySQL 是 Linux 系统中最流行的数据库之一，也是 LAMP（Linux、Apache、MySQL 与 Perl/Python/PHP）或 AMP（Apache、MySQL 与 Perl/Python/PHP）网站应用开发平台的一个重要组成部分，MySQL 数据库能够为 LAMP 应用提供后台的数据存储支持。因篇幅所限，本章主要讨论 MySQL 数据库的配置、管理与维护，间或介绍数据库的简单访问等内容，但不会详细讨论各种 SQL 语句及其使用。其内容主要包括：

- 安装与配置 MySQL 数据库；
- 访问 MySQL 数据库；
- 查询 MySQL 数据库；
- SQL 脚本与批处理；
- MySQL 数据库的备份与恢复；
- 密码维护与网络安全。



# 24.1 安装与配置 MySQL 数据库

## 24.1.1 安装 MySQL 数据库

在安装 Ubuntu Linux 基本系统之后，如果想要使用 MySQL 数据库，创建和开发自己的数据库应用，需要单独安装 MySQL 服务器（mysql-server）或 MySQL 客户端（mysql-client）软件包。在安装的过程中，安装程序还会安装相关的软件包，如 mysql-common 等。MySQL 服务器用于管理与维护数据库，MySQL 客户端用于连接与访问 MySQL 服务器。要安装 MySQL 数据库，可以使用下列命令，安装 MySQL 数据库的服务器与客户端。

```
$ sudo apt-get install mysql-server
```

在安装期间，安装程序将会提示用户输入并确认 MySQL 数据库管理员的密码。

## 24.1.2 my.cnf 配置文件

在初始的启动过程中，MySQL 数据库将会利用/etc/mysql/my.cnf 配置文件定制自己的运行环境。MySQL 服务器支持大量的配置变量，用于确定服务器的运行模式，其中，除了 port 和 socket 可用于客户端之外，大部分配置变量仅适用于 MySQL 数据库服务器，参见表 24-1。

表 24-1 MySQL 数据库配置变量

配置变量	mysqld 命令选项	简单说明
basedir	--basedir= <i>path</i>	MySQL 数据库的安装目录，凡以相对路径出现的文件均以这个目录为根目录。默认的安装目录为/usr
bind-address	--bind-address= <i>IP</i>	绑定的 IP 地址，表示 MySQL 服务器仅监听并接受针对指定 IP 地址的客户访问。这个配置变量的默认值为 127.0.0.1，这意味着 MySQL 服务器仅监听本地主机
datadir	-h <i>path</i> 或 --datadir= <i>path</i>	设定存储 MySQL 数据库及其数据库表的目录位置，默认的目录位置为 /var/lib/mysql
flush	--flush	如果把这个配置变量设定为 ON，表示在执行完每个 SQL 语句之后，MySQL 服务器需要立即清除内存缓冲区，把所有的数据变动都写到磁盘中，实现内存与磁盘的数据同步。通常，MySQL 服务器执行的每个 SQL 语句仅影响系统缓冲区，由操作系统实现内存与磁盘的数据同步
language	-L <i>langname</i> 或 --language= <i>langname</i>	当出现错误时，MySQL 数据库服务器将会以设定的语言向客户端返回错误信息。其中， <i>langname</i> 可以是一个语言的名字，也可以是语言文件的绝对路径名，其默认值为 /usr/share/mysql/english
log	-l [ <i>filename</i> ] 或 --log[= <i>filename</i> ]	用于设定和启用常规的查询日志文件，记录客户端的连接请求及提交的 SQL 语句。默认的查询日志文件为 /var/log/mysql.log
log-bin	--log-bin[= <i>basename</i> ]	用于设定和启用二进制日志文件。MySQL 数据库使用这个日志文件记录所有涉及数据变动的 SQL 语句。二进制日志文件可用于数据库的备份与恢复。利用这个配置变量设定的文件名是日志文件的基本名字，MySQL 服务器将在创建的文件名后面增加一个顺序号后缀。如果未指定文件名，MySQL 服务器将会以 “ <i>host_name-bin</i> ” 作为基本名字



续表

配置变量	mysqld 命令选项	简单说明
log-error	--log-error[=file]	设定 MySQL 数据库服务器的错误日志文件，用于记录 mysqld 守护进程的启停时间以及运行期间产生的严重错误信息。如果省略了文件名，MySQL 服务器将会使用 host_name.cnf 作为错误日志文件；如果指定的文件未加扩展名，MySQL 服务器会自动增加一个.err 文件名后缀。默认的错误日志文件为 /var/log/mysql.err
log-warnings	-W [level] 或 --log-warnings[=level]	设定是否在错误日志文件中记录警告信息，如网络故障、连接终止以及重新连接等。默认值为 1，表示记录警告信息；0 表示禁用；大于 1 的值表示记录更多的信息，如拒绝访问信息等
pid-file	--pid-file file	指定存储 MySQL 服务器守护进程 mysqld 的 PID 的文件名
port	--port portnum	设定 MySQL 服务器或客户端监听的 TCP/IP 端口号，其默认值为 3306
socket	--socket=path	客户端用于连接 MySQL 数据库或服务器用于监听本地连接的套接字文件的路径名。在 Ubuntu Linux 系统中，默认的套接字文件是 /var/run/mysqld/mysqld.sock
tmpdir	-t path 或 --tmpdir=path	MySQL 数据库守护进程 mysqld 用于存储临时文件的目录位置，其默认值为 /tmp
user	--u {username   userid} 或 --user={username   userid}	以指定的用户身份运行 mysqld 守护进程，默认值为 mysql

/etc/my.cnf 配置文件主要用于设定 MySQL 数据库文件的存储位置、数据库的各种日志文件、守护进程的 PID 文件及其他配置参数。下面是安装 MySQL 数据库之后系统提供的默认配置文件。

```
$ cat /etc/mysql/my.cnf
...
[client]
port      = 3306
socket    = /var/run/mysqld/mysqld.sock
...
# This was formally known as [safe mysqld]. Both versions are currently parsed.
[mysqld safe]
socket    = /var/run/mysqld/mysqld.sock
nice     = 0
[mysqld]
...
user      = mysql
pid-file  = /var/run/mysqld/mysqld.pid
socket    = /var/run/mysqld/mysqld.sock
port      = 3306
basedir   = /usr
datadir   = /var/lib/mysql
tmpdir    = /tmp
language  = /usr/share/mysql/english
...
$
```

按照上述配置文件的定义，MySQL 数据库文件通常位于 /var/lib/mysql 下的一个特定子目录中。例如，如果创建一个 customer 数据库，其相应的数据库文件将位于 /var/lib/mysql/customer 目录中。

注意，每当修改 my.cnf 配置文件时，都需要重新启动 mysqld 守护进程，才能确保修改后的配置立即生效。为此，可以使用下列命令。

```
$ sudo /etc/init.d/mysql restart
* Stopping MySQL database server mysqld                                [ OK ]
* Starting MySQL database server mysqld                                [ OK ]
* Checking for corrupt, not cleanly closed and upgrade needing tables.
```

### 24.1.3 MySQL 数据库命令行界面

MySQL 提供了一个简单的命令解释程序，用以支持交互或非交互式的 MySQL 数据库访问方式。当处于交互式命令行界面时，用户可以输入 MySQL 数据库提供的辅助命令，也可以输入标准的 SQL 语句。mysql 命令解释程序将会解释执行用户输入的命令，以表格形式返回命令的执行结果，以实现 MySQL 数据库的访问、管理与维护任务。当以非交互方式访问 MySQL 数据库时，则以制表符分隔数据字段的形式返回命令的执行结果。如果使用适当的命令选项，还能够以 HTML 或 XML 等形式返回命令的执行结果。

要进入 MySQL 命令行环境，可以使用 mysql 命令，其语法格式简写如下（表 24-2 给出了部分常用的选项及说明）。

```
mysql [options] [dbname]
mysql --user=username --password=password [dbname]
```

表 24-2 mysql 命令的部分选项

选 项	GNU 选项	简 单 说 明
<b>-?</b>	<b>--help</b>	显示帮助信息及可用的命令，然后退出并终止命令
<b>-B</b>	<b>--batch</b>	使用制表符作为字段分隔符，显示查询结果，每行显示一个记录
<b>-D dbname</b>	<b>--database=dbname</b>	指定使用哪一个数据库
<b>-e statement</b>	<b>--execute=statement</b>	执行指定的语句之后立即终止并退出 mysql
<b>-h hostname</b>	<b>--host=hostname</b>	连接指定主机中的 MySQL 服务器
<b>-H</b>	<b>--html</b>	生成 HTML 形式的输出
	<b>--pager[=command]</b>	使用指定的命令逐页显示查询结果。默认的命令取决于 PAGER 环境变量的设置。有效的命令是 less、more 或 “cat [> filename]” 等
<b>-p[password]</b>	<b>--password[=password]</b>	指定连接 MySQL 服务器时需要提供的密码。如果在 “-p” 选项后面直接输入密码，选项与密码之间不能有空格字符。如果 “-p” 或 “--password” 选项之后没有提供密码，MySQL 数据库将会提示用户输入密码，这样能够避免提供明文形式的密码
<b>-P portnum</b>	<b>--port=portnum</b>	指定连接 MySQL 服务器时使用的 TCP/IP 端口号
	<b>--prompt=prompt</b>	设置 MySQL 命令解释程序的提示符，默认的命令提示符是 “mysql>”
<b>-t</b>	<b>--table</b>	以表格形式输出查询结果。这也是交互查询的默认输出形式。注意，“-t” 选项主要用于设定非交互式的查询结果输出形式
	<b>--tee=filename</b>	除了在终端窗口中显示查询结果之外，同时也把输出数据写入指定的文件
<b>-u username</b>	<b>--user=username</b>	指定以哪一个用户身份连接 MySQL 服务器
<b>-X</b>	<b>--xml</b>	生成 XML 形式的输出

例如，为了以数据库管理员的身份进入 MySQL 命令行环境，可以使用下列 mysql 命令。由于没有在命令行中提供密码，MySQL 服务器将会提示用户输入 root 用户的密码。

```
$ mysql -u root -p
Enter password:
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 36
Server version: 5.0.67-0ubuntu6 (Ubuntu)
```



```
Type 'help;' or '\h' for help. Type '\c' to clear the buffer.
```

```
mysql>
```

在进入 MySQL 命令行界面之后，除了标准的 SQL 语句之外，还可以使用 help、pager 或 status 等命令。参见下列 help 命令的输出。

```
mysql> help;
...
List of all MySQL commands:
Note that all text commands must be first on line and end with ';'
?          (\?) Synonym for `help'.
clear      (\c) Clear command.
connect    (\r) Reconnect to the server. Optional arguments are db and host.
delimiter  (\d) Set statement delimiter. NOTE: Takes the rest of the line as new delimiter.
edit       (\e) Edit command with $EDITOR.
ego        (\G) Send command to mysql server, display result vertically.
exit       (\q) Exit mysql. Same as quit.
go         (\g) Send command to mysql server.
help       (\h) Display this help.
nopager   (\n) Disable pager, print to stdout.
notee     (\t) Don't write into outfile.
pager     (\P) Set PAGER [to_pager]. Print the query results via PAGER.
print     (\p) Print current command.
prompt    (\R) Change your mysql prompt.
quit      (\q) Quit mysql.
rehash    (\#\#) Rebuild completion hash.
source    (\.) Execute an SQL script file. Takes a file name as an argument.
status    (\s) Get status information from the server.
system    (\!!) Execute a system shell command.
tee       (\T) Set outfile [to outfile]. Append everything into given outfile.
use       (\u) Use another database. Takes database name as argument.
charset   (\C) Switch to another charset. Might be needed for processing binlog with
multi-byte charsets.
Warnings  (\W) Show warnings after every statement.
Nowarning (\w) Don't show warnings after every statement.

For server side help, type 'help contents'
mysql>
```

MySQL 数据库提供的辅助命令既可以采用规范的命令形式，如 help、pager 或 status，也可以采用缩写形式，如 “\h”、“\P”或“\s”等。规范的命令形式不区分大小写字母，但缩写命令形式区分大小写字母。此外，规范的命令形式通常需要附加一个分号 “;”，但缩写命令形式不需要加分号 “;” 后缀。注意，除了 help、exit 或 quit 等命令，几乎所有的 MySQL 交互命令（包括 SQL 语句）都需要附加一个分号字符 “;”。在交互访问环境中，也可以利用 source 或 “\.” 命令，运行 SQL 脚本文件。

在上述辅助命令中，charset 用于指定默认的字符集，以便在必要时能够保持客户端与 MySQL 数据库之间的同步（改动字符集之后，仅当再次连接时才能生效）。status 命令能够查询 MySQL 服务器以及客户端与服务器的连接状态信息。若想使用文件记录查询结果等输出信息，可以使用 tee 命令，终端窗口显示的所有数据将会附加到指定的文件。对于调试而言，这一功能是非常有用的。利用 pager 命令，还能够采用 less、more 或其他程序逐页显示查询的结果。

#### 24.1.4 MySQL 数据库图形界面

在 Ubuntu Linux 系统中，如果安装了 mysql-query-browser、mysql-admin 及 mysql-gui-common 等软件包，还可以在 GNOME 桌面中访问 MySQL 数据库。例如，若想管理与维护 MySQL 数据库，可以选择“应用程序→编程→MySQL Administrator”菜单，在“MySQL Administrator”连接

界面输入 MySQL 服务器的主机名、端口号、用户名以及密码等信息之后，即可进入图 24-1 所示的“MySQL Administrator”主界面。

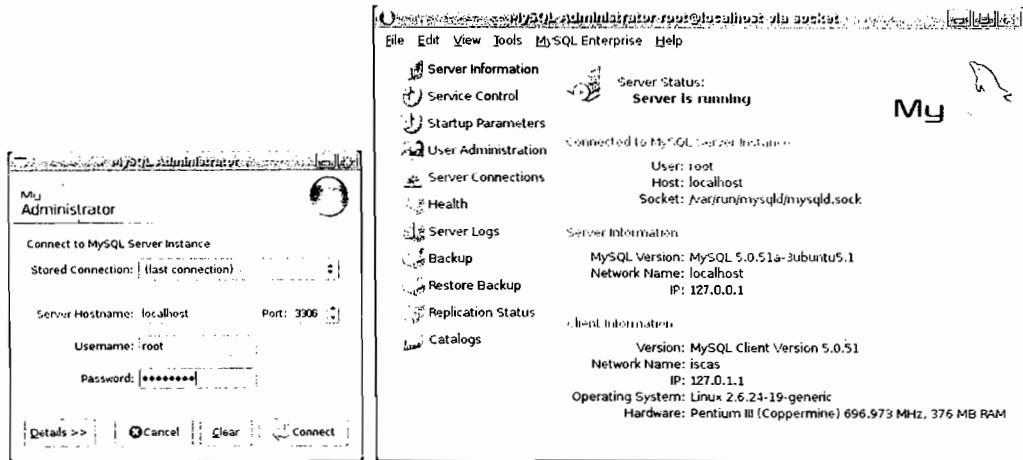


图 24-1 “MySQL Administrator”连接与主界面

此时，可以查询 MySQL 数据库的相关信息，监控 mysqld 守护进程的运行状态，设置 MySQL 数据库的主配置文件，管理 MySQL 数据库用户(如增加用户、授权以及资源限制等)，查询 MySQL 服务器的连接状态，了解客户机的 SQL 查询数量，查询日志文件的内容，设置、调度数据库的备份，以及恢复备份的数据，等等。

若想访问 MySQL 数据库，以图形界面的形式查询数据库，可以选择“应用程序→编程→MySQL Query Browser”菜单，在“MySQL Query Browser”连接界面输入 MySQL 服务器的主机名、端口号、用户名及密码等信息之后，即可进入图 24-2 所示的“MySQL Query Browser”主界面。

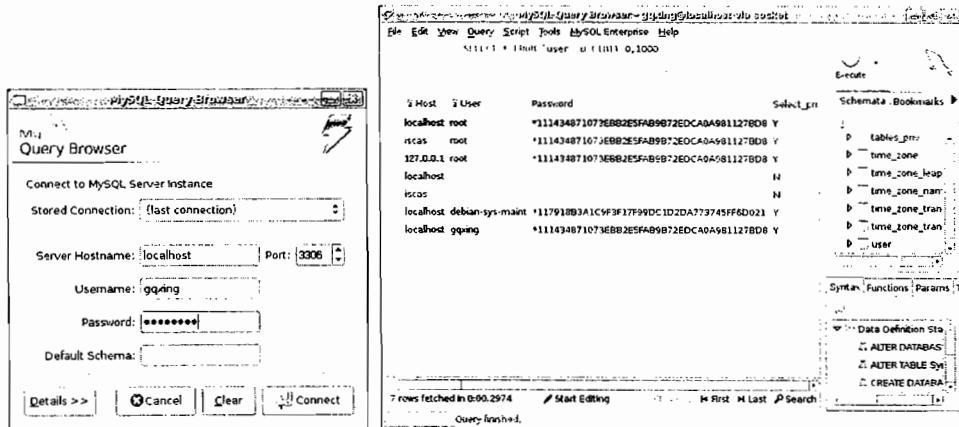


图 24-2 “MySQL Query Browser”连接界面

此时，可以在“Schemata”窗口中选择数据库，进而选择数据库表，在正上方的窗口中输入 SQL 语句，或者使用自动提示的 select 语句作为原型，经过适当的修改之后，点击“Execute”按钮查询数据。右下方的窗口提供 SQL 语句的联机查询，供用户随时查阅。



## 24.1.5 设置数据库用户及其访问权限

安装 MySQL 服务器之后，只有一个用户能够访问数据库管理系统，这就是数据库管理员 root。MySQL 服务器的数据库管理员 root 具有最大的访问权限，能够创建和删除数据库等，其地位类似于 Linux 系统中的超级用户。

由于数据库管理员 root 具有无限的权力，一般仅限于数据库的管理与维护，不建议以 root 用户的身份访问数据库，故需要增加必要的 MySQL 用户，设定密码，赋予用户适当的数据库访问权限，如创建应用数据库等。

通常，MySQL 维护的所有用户名及其密码数据都存储在一个专用的数据库 mysql 中。数据库管理员可以利用 create、grant 及 insert 等语句，把用户加到这个数据库中。此外，利用 grant 语句还可以同时赋予用户访问数据库或数据库表的权限，其语法格式简写如下。

```
mysql> GRANT ALL PRIVILEGES ON database TO username@"servername" IDENTIFIED BY 'password';
```

其中，database 是一个数据库名，username 表示新增加的用户，servername 是 MySQL 数据库服务器的主机名或 IP 地址，password 是用户的密码。

例如，要增加一个 MySQL 数据库用户 gqxing（假定其密码为 a1b2c3），赋予其访问本地主机（localhost）数据库 books 的所有访问权限，首先需要使用 root 注册到 MySQL 数据库，选定含有用户账号及其访问权限信息的 mysql 数据库，如下所示。

```
$ mysql -u root -p  
Enter password:  
.....  
mysql> USE mysql;  
Reading table information for completion of table and column names  
You can turn off this feature to get a quicker startup with -A  
Database changed  
mysql>
```

然后利用下列 grant 命令，赋予用户使用 books 数据库的所有访问权限。

```
mysql> GRANT ALL PRIVILEGES ON books.* TO gqxing@"localhost" IDENTIFIED BY 'a1b2c3';  
Query OK, 0 rows affected (0.23 sec)  
mysql>
```

如果 flush 配置变量没有设置为 ON，或者在启动 MySQL 服务进程时没有使用 “--flush” 选项，还需要使用 flush 命令，把上述设置信息写入 mysql 数据库中，才能使新加的用户账号立即生效，如下所示。

```
mysql> FLUSH PRIVILEGES;  
Query OK, 0 rows affected (0.00 sec)  
  
mysql>
```

## 24.2 访问 MySQL 数据库

### 24.2.1 创建、查询、使用与删除数据库

在使用 MySQL 数据库之前，首先需要创建数据库。一个数据库通常是由一个或多个数据库表组成的，用于存储不同的关系数据。为了创建一个数据库，可在 MySQL 命令行环境输入下列“create database”命令。

```
mysql> CREATE DATABASE books;
Query OK, 1 row affected (0.02 sec)
```

```
mysql>
```

利用“show databases”命令，可以查询 MySQL 服务器管理的所有数据库。例如，下列命令表示 MySQL 服务器中当前存在 3 个数据库。

```
mysql> SHOW DATABASES;
+-----+
| Database      |
+-----+
| information_schema |
| books          |
+-----+
2 rows in set (0.01 sec)
```

```
mysql>
```

在使用 SQL 语句访问数据库时，通常需要指定一个默认的数据库，作为 SQL 命令操纵的对象。

例如，列举数据库表，查询其中存储的数据等。要指定一个默认的数据库，可以使用下列 use 命令。

```
mysql> USE books;
Database changed
mysql>
```

如果需要删除一个不再需要的数据库，可以使用“drop database”命令。例如，下列命令用于删除刚才建立的 books 数据库。

```
mysql> DROP DATABASE books;
Query OK, 0 rows affected (0.00 sec)

mysql>
```

## 24.2.2 创建、查询与删除数据库表

在创建数据库之后，还需要按照关系范式创建一个或多个数据库表，否则无法存储任何数据。在创建数据库表之前，也可以使用“show tables”命令查询数据库中当前存在的数据库表，如下所示。

```
mysql> USE books;
Database changed
mysql> SHOW TABLES;
Empty set (0.00 sec)

mysql>
```

数据库表及其字段定义决定了数据库的结构。要创建一个包含各种参考书的数据库表，可以使用“create table”语句，设定下列数据库表结构。

```
mysql> CREATE TABLE booklist (bookname VARCHAR(30), press VARCHAR(20), isbn VARCHAR(18),
publish char(4));
Query OK, 0 rows affected (0.38 sec)
```

```
mysql>
```

一旦创建了任何数据库表，就可以使用“show tables”语句，查询数据库存在哪些数据库表，如下所示。

```
mysql> SHOW TABLES;
+-----+
| Tables_in_books |
+-----+
| booklist        |
+-----+
1 row in set (0.01 sec)

mysql>
```

为了验证创建的数据库表是否正确，或者忘记数据库表的字段名字时，可以使用下列



DESCRIBE 语句，查询数据库表的结构描述。

```
mysql> DESCRIBE booklist;
+-----+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+
| bookname | varchar(30) | YES | | NULL | |
| press | varchar(20) | YES | | NULL | |
| isbn | varchar(18) | YES | | NULL | |
| publish | char(4) | YES | | NULL | |
+-----+-----+-----+-----+-----+
4 rows in set (0.01 sec)
```

```
mysql>
```

当需要删除数据库表时，可以使用“drop table”命令。例如，要删除数据库表 booklist，可以使用下列 drop 命令。

```
mysql> DROP TABLE booklist;
Query OK, 0 rows affected (0.00 sec)
```

```
mysql>
```

### 24.2.3 录入数据

创建数据库表之后，即可输入实际的数据。为此，可以使用 insert 语句或“load data”语句，直接输入或从文本文件中加载数据。

新建的数据库表是一个空表，如果需要录入大量的数据，最简单易行的方法是创建一个文本文件，其中每一行包含一个数据库表的记录。然后使用 load 语句，把所有的数据一次性地录入数据库表中。

例如，可以创建下列文件，其中每一行都是一个完整的图书记录，每个数据字段逐一对应数据库表的每个字段定义（以“create table”语句中指定的字段顺序为准），中间以制表符作为字段分隔符。对于省略的字段，可以使用“\N”作为 NULL 值。

```
$ cat book.txt
UNIX: The Complete Reference McGraw-hill 978-0-07-22626-7 2007
TCP/IP Network Administration O'Reilly & Associates 0-937175-82-X 1994
.....
$
```

制作好 book.txt 文本文件之后，为了把其中的数据导入数据库表 booklist，可以使用下列 load 语句。

```
mysql> LOAD DATA LOCAL INFILE '~/book.txt' INTO TABLE booklist;
Query OK, 2 rows affected, 1 warning (0.18 sec)
Records: 2 Deleted: 0 Skipped: 0 Warnings: 1
```

```
mysql>
```

在 load 语句中，默认的字段分隔符与行终止符分别是制表符与换行符。如果必要，也可以在上述 load 语句中明确指定其他字段分隔符或行终止符。注意，如果数据文件是在 Windows 环境中创建的，通常需要增加一个“lines terminated by '\r\n'”选项，表示使用“\r\n”作为行终止符。

如果输入的记录数量有限，如一次只需增加一两个新记录，可以直接使用 INSERT 语句。使用 insert 语句时，最简单的形式是按照“create table”语句列举的字段顺序，依次提供每个数据字段值。假定需要在上述的数据表 booklist 中增加一个图书记录，可以使用下列 insert 语句。

```
mysql> INSERT INTO booklist VALUES ('UNIX System Security', 'Addison-Wesley',
'0-201-57030-0', '1991');
Query OK, 1 row affected (0.02 sec)
```

```
mysql>
```

在提供数据字段值时，字符串和日期等数值前后应加单引号。在使用 insert 语句时，如果想要省

略相应的数据字段，可以直接插入一个文字值 NULL，但不能像 load 语句那样使用替代的字符“\N”。

## 24.3 查询 MySQL 数据库

MySQL 提供了大量的命令或 SQL 语句，可用于查询数据库、数据库表或实际的数据。在查询数据库时，由于访问权限的差异，不同的用户身份查询的结果是不同的。例如，在查询可用的数据库时，普通用户的查询结果如下。

```
$ mysql -u gqxing -p books
Enter password:
-----
mysql> SHOW DATABASES;
+-----+
| Database      |
+-----+
| information_schema |
| books          |
+-----+
2 rows in set (0.00 sec)
```

mysql>

而数据库管理员的查询结果如下（mysql 是一个重要的数据库，其中存有 MySQL 数据库服务器维护的数据库、数据库表、用户以及用户访问权限等重要信息）。

```
$ mysql -u root -p
Enter password:
-----
mysql> SHOW DATABASES;
+-----+
| Database      |
+-----+
| information schema |
| books          |
| mysql          |
+-----+
3 rows in set (0.32 sec)
```

mysql>

### 24.3.1 查询数据库表

“show tables”命令用于显示数据库中维护的所有数据库表。在使用 show 命令查询之前，首先必须使用 use 命令指定默认的数据库，以便 MySQL 知道究竟查询哪一个数据库中的数据库表。例如，下列 use 命令表示使用 books 作为当前的数据库操作对象，show 命令表示查询其中的数据库表。

```
mysql> USE books;
Database changed
mysql> SHOW TABLES;
+-----+
| Tables in books |
+-----+
| booklist         |
+-----+
1 row in set (0.00 sec)
```

mysql>



### 24.3.2 查询数据库表结构

`describe` 命令用于显示指定数据库表中的所有数据字段及其属性定义，如数据类型、是否关键字、是否允许为空及默认值等。在下面的例子中，数据库表 `booklist` 仅含有 4 个字段：`bookname`（书名）、`press`（出版社）、`isbn`（ISBN）及 `publish`（出版日期）。

```
mysql> DESCRIBE booklist;
+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+-----+-----+-----+
| bookname | varchar(30) | YES | | NULL | |
| press | varchar(20) | YES | | NULL | |
| isbn | varchar(18) | YES | | NULL | |
| publish | char(4) | YES | | NULL | |
+-----+-----+-----+-----+
4 rows in set (0.01 sec)

mysql>
```

### 24.3.3 查询数据库表中的数据内容

利用 `select` 交互命令，可以查询指定数据库表中存有的选定数据或所有数据。例如，输入下列命令可以查询数据库表 `booklist` 中的所有数据。

```
mysql> SELECT * FROM booklist;
+-----+-----+-----+-----+
| bookname | press | isbn | publish |
+-----+-----+-----+-----+
| UNIX: The Complete Reference | McGraw-Hill | 978-0-07-22636-7 | 2007 |
| TCP/IP Network Administration | O'Reilly & Associate | 0-937175-82-x | 1994 |
| UNIX System Security | Addison-Wesley | 0-201-57030-0 | 1991 |
+-----+-----+-----+-----+
3 rows in set (0.01 sec)

mysql>
```

## 24.4 SQL 脚本与批处理

除了在交互方式输入命令或 SQL 语句之外，`mysql` 也接受以脚本文件形式提供的批量命令或 SQL 语句，实现 MySQL 数据库的访问，显示查询结果等。若要以批量处理方式访问与查询 MySQL 数据库，可以把交互过程使用的命令或 SQL 语句写入一个文件，然后以 I/O 重定向的方式运行 `mysql`，其语法格式简写如下。

```
mysql -u username -p[password] dbname < scriptfile
```

如果把“`use dbname`”语句也加到 SQL 脚本文件，作为查询前的第一个语句，则无需在命令行中指定当前使用的数据库，其调用方式如下。

```
mysql -u username -p[password] < scriptfile
```

注意，与交互访问相比，采用批处理方式运行 SQL 脚本文件时，其数据输出形式稍有简化。例如，假定存在下列 SQL 脚本文件，其运行时的数据输出结果有别于交互式查询。

```
$ cat showbook.sql
show databases;
use books;
show tables;
select * from booklist;
```

```

quit
$ mysql -u gqxking -p < showbook.sql
Enter password:
Database
information_schema
books
Tables_in_books
booklist
bookname      press      isbn      publish
UNIX: The Complete Reference McGraw-hill   978-0-07-22626-7   2007
TCP/IP Network Administration O'Reilly & Associate 0-937175-82-X 1994
UNIX System Security Addison-Wesley 0-201-57030-0 1991
$
```

如果当前已经处于 mysql 交互运行环境，还可以采用下列 source 或 “\.” 命令形式，运行 SQL 脚本文件，其输出效果完全等同于交互操作。

```
mysql> source scriptfile
```

或

```
mysql> \. scriptfile
```

例如，如果在交互环境中运行 SQL 脚本文件 showbook.sql，其输出结果如下。

```

mysql> source showbook.sql
+-----+
| Database          |
+-----+
| information_schema |
| books             |
+-----+
2 rows in set (0.00 sec)

Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
+-----+
| Tables_in_books |
+-----+
| booklist         |
+-----+
1 row in set (0.00 sec)

+-----+-----+-----+-----+
| bookname        | press     | isbn      | publish |
+-----+-----+-----+-----+
| UNIX: The Complete Reference | McGraw-hill   | 978-0-07-22626-7 | 2007   |
| TCP/IP Network Administration | O'Reilly & Associate | 0-937175-82-X | 1994   |
| UNIX System Security       | Addison-Wesley | 0-201-57030-0  | 1991   |
+-----+-----+-----+-----+
3 rows in set (0.00 sec)
```

```
mysql>
```

如果数据库的查询语句产生大量的数据输出，超出一个终端窗口能够容纳的内容，将会导致查询结果的滚动输出，此时可以利用 more 或 less 等命令，采用管道机制逐页地显示查询结果，其语法格式如下。

```
mysql -u username -p[password] < batch-file | more
```

也可以采用下列命令语法，把查询结果写入一个文本文件，以便做进一步的处理。

```
mysql -u username -p[password] < batch-file > mysql.out
```

以批处理方式运行 SQL 脚本时，如果运行时间过长，通常可能需要显示运行 SQL 脚本的进度信息。为此，可在 SQL 脚本文件的适当位置增加下列形式的 select 语句，以便输出指定的信息。

```
SELECT 'info_to_display' AS ' ';
```



下面的例子说明怎样以 SQL 脚本的形式，实现创建数据库 books、创建数据库表 booklist 以及录入数据等交互操作过程。

```
$ cat create_books.sql
create database books;
use books;
create table booklist (bookname VARCHAR(30), press VARCHAR(20), isbn VARCHAR(18),
publish char(4));
load data local infile '~/book.txt' into table booklist;
quit
$ mysql -u gqxing -p < create_books.sql
Enter password:
$
```

## 24.5 MySQL 数据库的备份与恢复

为了确保数据的安全，备份与恢复数据库是必不可少的一项重要功能。MySQL 提供了各种备份策略与方式方法，用户可以根据具体的应用需求，选用最适合的备份与恢复方法。

### 24.5.1 数据库备份方法

#### 1. 文件备份

在 MySQL 数据库中，所有数据库表均以文件形式存储。因此，通过文件备份方式，即可实现数据库的备份。为了确保数据的一致性，可在备份前对相关的数据库表依次执行“lock tables”（加锁，但允许用户查询数据库表）和“flush tables”（把缓冲区中的数据写入磁盘文件）操作。

#### 2. 数据备份

另外一种方法是采用下列 SQL 语句，仅备份数据库表中的数据，而忽略数据库表的结构数据，最终把实际的数据写入文本文件。当需要恢复数据时，可以使用“load data infile”语句或 mysqlimport 命令，恢复备份的数据，如下所示。

```
SELECT * INTO OUTFILE 'filename' FROM tablename
```

#### 3. 利用备份工具实现完整备份

利用 MySQL 数据库提供的 mysqldump 命令或 mysqlhotcopy 脚本，也可以实现数据库的备份。与 mysqlhotcopy 相比，mysqldump 是一个更通用的备份工具，能够备份任何数据库及其数据库表，生成一个包含 SQL 语句的文本文件。因此，为了制作一个完整的数据库备份，可以选用下列命令之一。

```
mysqldump --tab=path --opt dbname
```

或

```
mysqlhotcopy dbname path
```

注意，mysqlhotcopy 实际上只是复制所有的数据库表文件，其中包括\*.frm、\*.MYD 和\*.MYI 等文件。

#### 4. 利用更新日志文件实现增量备份

MySQL 数据库也支持增量备份，为了实现增量备份，必须确保在 my.cnf 配置文件中启用了 bin\_log 配置选项（删除其前面的注释符“#”），或者在启动服务器时使用“--log-bin”选项启用了更新日志功能。

更新日志文件包含数据库中执行修改动作的所有数据更新语句，在需要执行增量备份时，应采用“flush logs”语句轮换 MySQL 数据库的更新日志文件。之后，还需要复制自上次做完完整备份

或增量备份之后新创建的所有二进制日志文件，这些包含二进制数据的日志文件可以看作增量备份。恢复时，可以使用 mysqlbinlog 命令，利用二进制日志文件，从指定的时间点（如上一次的备份时间）开始，直至当前时间或指定的截止时间点位置，恢复其间发生变动的任何数据。

当下一次再做完整备份时，也应当使用“flush logs”、“mysqldump --flush-logs”或“mysqlhotcopy --flushlog”命令，轮换二进制的增量备份日志文件。

### 5. 利用 BACKUP 语句实现数据库备份

新版（6.0.5）的 MySQL 数据库提供了“backup database”与 restore 语句，用于实现数据库的备份与恢复。“backup database”命令能够根据给定的时间点，制作一个 MySQL 数据库实例（实示数据与元数据）的完整副本。利用备份的副本，能够恢复相应时间点的数据。增加“backup latabase”与 restore 语句的目的是希望能够最大限度地并发处理其他数据库操作，而不必脱机执行数据库的备份与恢复，限制客户端的数据库访问。

利用“backup database”语句，可以同时备份一个或多个数据库（数据库名之间需加逗号“,”分隔符），把数据写到指定的文件，如下所示。

```
BACKUP DATABASE books TO '/tmp/backupfile';
BACKUP DATABASE books, papers TO '/tmp/backupfile';
```

如果想要备份所有的数据库，可以使用星号“\*”通配符，示例如下。

```
BACKUP DATABASE * TO '/tmp/backupfile';
```

当需要恢复数据库时，可以利用下列 restore 语句与备份的数据文件，恢复数据库及其中的数据。

```
RESTORE FROM '/tmp/backupfile';
```

## 24.5.2 MySQL 数据库备份

在备份 MySQL 数据库时，可以使用 mysqldump 或 mysqlhotcopy 命令。这两个工具均可用于备份指定的数据库或一组数据库。备份的数据中通常包含建立数据库表以及加载数据等 SQL 语句，可以导入另一个 MySQL 数据库服务器或其他 SQL 数据库服务器。此外，mysqldump 还可用于生成 CSV 格式的文件、带有分隔符的文本文件或 XML 格式的文件，其语法格式简写如下。

```
mysqldump [options] dbname [tables]
mysqldump [options] --databases dbname1 [dbname2 dbname3...]
mysqldump [options] --all-databases
```

其中，dbname 是准备备份的数据库，tables 表示数据库表，部分常用的选项及其说明参见表 24-3。

表 24-3 mysqldump 命令的部分选项

选 项	GNU 选 项	简 单 说 明
-?	--help	显示帮助信息，列举可用的命令，然后退出并终止命令
	--add-drop-database	在每个“create database”语句之前增加一个“drop database”语句
	--add-drop-table	在每个“create table”语句之前增加一个“drop table”语句
	--add-locks	在备份的每个数据库表前后分别增加一个“lock tables”与“unlock tables”语句。备份时采用这个选项能够加快备份数据的恢复
-A	--all-databases	备份所有数据库中的所有数据库表
-B	--databases dbnames	备份指定的数据库。通常，mysqldump 命令将会把第一个命令行参数看作数据库，随后的参数看作数据库表。利用这个选项，可以把所有的命令行参数作为数据库处理。在处理每个数据库之前，mysqldump 将会增加一对“create database”与 use 语句



续表

选 项	GNU 选项	简 单 说 明
	--fields-terminated-by= <i>char</i> --fields-enclosed-by= <i>char</i>	指定字段的终止符，默认的字段终止符为制表符“\t”。这个选项需与“-T”选项一起使用
-h <i>hostname</i>	--host= <i>hostname</i>	备份指定主机上 MySQL 服务器中的数据，默认的主机是 localhost
	--ignore-table= <i>dbname.tablename</i>	禁止备份指定的数据库表。指定数据库表时必须采用 “ <i>database.table</i> ” 的形式同时给出数据库名与数据库表名。如果想要禁止备份多个数据库表，可以重复使用这个选项，同时列举多个数据库表
	--lines-terminated-by= <i>char</i>	指定行的终止符，默认的行终止符为换行符 “\n”。这个选项需与 “-T” 选项一起使用
	--opt	这个选项实际上是一种简化形式，表示同时指定了 “--add-drop-table”、“--add-locks”、“--create-options”、“--disable-keys”、“--extended-insert”、“--lock-tables”、“--quick” 及 “--set-charset” 等选项。使用这个选项能够快速地执行备份操作，生成一个备份文件，同时也能够把数据快速地重新加载到 MySQL 数据库中。“--opt” 选项是一个默认的选项，如想禁用，可增加一个 “--skip-opt” 选项
-p[ <i>password</i> ]	--password[= <i>password</i> ]	指定连接 MySQL 服务器时需要提供的密码。注意，“-p” 选项与密码之间不能有空格
-P <i>portnum</i>	--port= <i>portnum</i>	指定连接 MySQL 服务器时使用的 TCP/IP 端口号
-r <i>file</i>	--result-file= <i>file</i>	把备份数据写到一个指定的文件。在 Windows 系统中应当使用这个选项，以防止把换行符 “\n” 转换成回车换行符 “\r\n”
-T <i>path</i>	--tab= <i>path</i>	默认情况下，生成一个以制表符为字段分隔符，以换行符为行终止符的数据文件。对于备份的每一个数据库表，mysqldump 将会分别创建一个 <i>tbl_name.sql</i> 文件（其中包含 “create table” 语句，以便在恢复数据时能够创建数据库表）与 <i>tbl_name.txt</i> 文件（其中包含数据库表中的实际数据）。选项参数 <i>path</i> 用于指定一个目录，以便存储上述两个文件。注意，运行 mysqldump 命令时，仅当 MySQL 服务器位于同一主机才能使用这个选项
	--tables	取消 “--databases” 或 “-B” 选项的意义。mysqldump 命令将会把所有的命令行参数看作数据库表
-u <i>username</i>	--user= <i>username</i>	以指定的用户身份访问 MySQL 服务器，备份用户自己拥有或能够访问的数据库

在指定数据库时，如果没有指定数据库表，或者在命令行中使用了 “--databases” 或 “--all-databases” 选项，则意味着备份整个数据库。

假定已经赋予用户 gqxing 使用密码 alib2c3 访问 books 数据库的所有权限，则 gqxing 用户可以利用下列命令，把 books 数据库备份到 /tmp/books.sql 文件中。

```
$ mysqldump --add-drop-table -u gqxing -p alib2c3 books > /tmp/books.sql
```

注意，在备份 MySQL 数据库时，还应以数据库管理员的身份，备份 mysql 数据库，因为其中包含数据库用户的所有账号与访问权限信息。

### 24.5.3 MySQL 数据库恢复

采用 mysqldump 命令备份数据库时将会创建一个由一系列 SQL 语句组成的脚本文件，而且，这些 SQL 语句是可以直接执行的。因此，在恢复 MySQL 数据库时，可以使用下列命令，把备份文件作为批量运行的 SQL 脚本文件。

```
mysql -u [username] -p[password] [database] < [backupfile]
```

针对先前的例子，可以使用下列命令恢复整个数据库及其中的数据内容。

```
$ mysql -u gqxing -p1b2c3 books < /tmp/books-bak.sql
```

注意，由于数据库 mysql 中包含所有数据库用户的访问权限信息，有时可能还需要以数据库管理员的身份恢复 mysql 数据库。

#### 24.5.4 MySQL 数据库表的备份与恢复

如果只需备份数据库中的一个或多个数据库表，可以使用 select 语句，从选定的数据库及其数据表中，把数据导入备份文件。而在恢复数据库表时，可以使用 load 语句，把数据导入数据库表中。在下述例子中，我们把 books 数据库 booklist 数据库表中的数据导入/tmp/books.sql 文件中，然后又从/tmp/books.sql 备份文件中，把数据导入 books-new 数据库的同名数据库表中。

```
mysql> USE books;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> SELECT * FROM booklist INTO OUTFILE '/tmp/books.sql';
Query OK, 4 rows affected (0.01 sec)

mysql> USE books-new;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> LOAD DATA INFILE '/tmp/books.sql' REPLACE INTO TABLE booklist;
Query OK, 4 rows affected (0.02 sec)
Records: 4 Deleted: 0 Skipped: 0 Warnings: 0
```

mysql>

上述 load 语句中的 replace 表示，如果两个表中的记录是相同的，或者具有同一主键，源数据库表中的数据可以覆盖目的数据库表中已有的记录。如果换作关键字 ignore，则意味着仅插入主键不同的记录，而跳过主键相同的记录。

#### 24.5.5 增量备份与恢复

为了从二进制日志文件中恢复数据，必须知道二进制日志文件的当前存储位置与文件名。通常，数据库服务器是在指定的数据目录中创建二进制日志文件的，但也可以利用 my.cnf 文件中的配置变量 “log\_bin”，或者在启动 mysqld 守护进程的命令行中利用 “--log-bin” 选项指定的路径名，把二进制日志文件放到一个不同的存储位置。为了确定二进制日志文件当前的存储位置及其名字，可在命令行中输入下列命令。

```
$ mysql -u root -p -e "SHOW MASTER STATUS"
Enter password:
+-----+-----+-----+-----+
| File | Position | Binlog_Do_DB | Binlog_Ignore_DB |
+-----+-----+-----+-----+
| mysql-bin.000002 | 98 | | |
+-----+-----+-----+-----+
```

如果已经位于 MySQL 数据库交互访问模式，可以输入下列命令语句。

```
mysql> SHOW MASTER STATUS;
+-----+-----+-----+-----+
| File | Position | Binlog_Do_DB | Binlog_Ignore_DB |
+-----+-----+-----+-----+
| mysql-bin.000002 | 98 | | |
+-----+-----+-----+-----+
```



```
+-----+-----+-----+-----+
1 row in set (0.00 sec)
```

```
$
```

## 1. 指定恢复的时间点

要指定数据恢复的起止时间点，可在 mysqlbinlog 命令中分别使用“`--start-date`”和“`--stop-date`”选项，同时采用 DATETIME 格式表示时间。例如，假定在 2008 年 12 月 20 日上午 10:00 左右执行的 SQL 语句删除了一个数据库表。为了恢复数据库表及其中的数据，首先需要恢复最近一次作的（完整）备份，然后执行下列命令。

```
$ mysqlbinlog --stop-date="2008-12-20 9:59:59" \
    /var/log/mysql/mysql-bin.000002 | mysql -u root -p
```

上述命令意味着以“`--stop-date`”选项指定的日期和时间为截止时间，恢复其间发生变动的所有数据。如果其中没有发现错误的 SQL 语句，也许还需要恢复之后执行的操作。鉴于此，可能还需要指定起始运行日期和时间，再次运行下列 mysqlbinlog 命令。

```
$ mysqlbinlog --start-date="2008-12-20 10:01:00" \
    /var/log/mysql/mysql-bin.000002 | mysql -u root -p
```

上述命令表示重新执行日志文件中从 12 月 20 日上午 10:01 开始记录的所有 SQL 语句。由此可见，也许还需要仔细地考察日志文件，找出准确的恢复时间，以免出现想当然的错误或重复执行 mysqlbinlog 命令。

为了直接查阅二进制日志文件中的内容而不是重新执行，可以采用下列 mysqlbinlog 命令（其中的`/var/log/mysql/mysql-bin.000002`为二进制日志文件）。

```
$ mysqlbinlog /var/log/mysql/bin.000002 > /tmp/mysql_restore.sql
```

## 2. 指定日志文件的记录位置

除了指定起止时间点之外，还可以在 mysqlbinlog 命令中使用“`--start-position`”和“`--stop-position`”选项指定日志文件的存储位置，其效果等同于指定起止时间选项，除了指定的是日志文件的记录位置编号而非时间。使用记录的位置编号能够更准确地确定从日志文件中的哪一部分开始恢复，尤其是当许多交易与操作有误的 SQL 语句同时出现时。为了确定位置编号，可以运行下列 mysqlbinlog 命令，指定一个出现误操作的时间范围，把输出结果重定向到一个文本文件，以便做进一步的考察。

```
$ mysqlbinlog --start-date="2008-12-20 9:55:00" \
--stop-date="2008-12-20 10:05:00" \
/var/log/mysql/mysql-bin.000002 > /tmp/mysql_restore.sql
```

使用文本编辑器打开新生成的文件，其中包含指定时间周期范围内执行的所有 SQL 语句。从中找出有误因而不想再执行的 SQL 语句，确定语句在日志文件中的位置编号（其标记为“# at number”），在相应的 SQL 语句前面增加注释符“#”。同样，首先恢复最近一次制作的（完整）备份，然后使用下列 mysqlbinlog 命令处理二进制日志文件，恢复位置编号 6688 之前，6690 之后执行的 SQL 语句，绕过有问题的 6689 号 SQL 语句。

```
$ mysqlbinlog --stop-position="6688" /var/log/mysql/mysql-bin.000002 \
    | mysql -u root -p
$ mysqlbinlog --start-position="6690" /var/log/mysql/mysql-bin.000002 \
    | mysql -u root -p
```

上述第一个命令表示恢复从上一次备份开始直至指定位置为止的所有交易。第二个命令表示从指定的位置开始，直至日志文件结束，恢复其中所有的交易。在 mysqlbinlog 命令的输出中，由于每个 SQL 语句之前都包含一个“`set timestamp`”语句，故恢复的数据及其相关的 MySQL 日志能够反映每个交易的初始执行时间。

# 24.6

## 密码维护与网络安全

### 24.6.1 维护数据库管理员密码

MySQL 服务器的数据库管理员 root 具有无限的权力，为了确保数据库的安全，通常需要定期更换数据库管理员 root 的密码，以避免被窃取。例如，要修改数据库管理员 root 地用户密码，把之前的密码改为 newpasswd，可以 root 的用户身份访问 MySQL 数据库，在 mysql 交互运行环境中，利用 update 命令设置新的密码（注意，设置新密码时需要输入先前的密码）。

```
$ mysql -u root -p mysql
Enter password:
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 36
Server version: 5.0.67-0ubuntu6 (Ubuntu)

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql> update user set password=password('newpassword') where user='root';
Query OK, 0 rows affected (0.00 sec)
Rows matched: 3  Changed: 0  Warnings: 0

mysql> flush privileges;
Query OK, 0 rows affected (0.00 sec)

mysql> quit
Bye
$
```

在上述的 update 命令中，应使用实际的密码替换 newpassword。执行后，可以采用下列命令验证新设的密码是否已经生效。

```
$ mysql -u root -p mysql
Enter password:
.....
mysql>
```

在管理与维护 MySQL 数据库时，也可以使用 mysqladmin 命令创建、删除数据库，查询 MySQL 数据库的运行状态信息，设置数据库管理员密码，等等，其语法格式简写如下。

**mysqladmin [options] command [command-arguments]**

表 24-4 给出了 mysqladmin 命令支持的部分选项及其说明。

表 24-4 mysqladmin 命令支持的部分选项

选 项	GNU 选 项	简 单 说 明
-?	--help	显示帮助信息，列出可用的命令，然后退出并终止命令
-h <i>hostname</i>	--host= <i>hostname</i>	连接指定主机中的 MySQL 服务器
-p[ <i>password</i> ]	--password[=] <i>password</i>	指定连接 MySQL 服务器时需要提供的密码。注意，如果在命令行上提供密码，“-p”选项与密码之间不能留有空格
-P <i>portnum</i>	--port= <i>portnum</i>	指定连接 MySQL 服务器时使用的 TCP/IP 端口号
-u <i>username</i>	--user= <i>username</i>	指定以哪一个用户身份连接 MySQL 服务器



表 24-5 给出了 mysqladmin 命令支持的部分命令语句及其说明。

表 24-5 mysqladmin 命令支持的部分命令语句及其参数

命令及其参数	简单说明
<code>create dbname</code>	创建指定的数据库
<code>drop dbname</code>	删除指定的数据库及其拥有的所有数据库表
<code>password password</code>	为指定的用户设置新密码。下次调用 mysqladmin 命令或连接 MySQL 服务器时，需要使用新设定的密码
<code>shutdown</code>	终止 MySQL 数据库服务器
<code>status</code>	显示 MySQL 服务器的状态信息，如 MySQL 数据库运行的时间（以秒为单位）、客户端的查询数量、已经打开和当前打开的数据表的数量及所有查询的平均响应时间等
<code>variables</code>	显示 MySQL 服务器的系统变量及其设置
<code>Version</code>	显示 MySQL 服务器的版本信息

例如，要把数据库管理员 root 的用户密码设置为 newpassword，可以使用下列 mysqladmin 命令（注意，设置新密码时需要输入先前的密码）。

```
$ mysqladmin -u root -p password newpassword
Enter password:
$
```

要查询 MySQL 数据库服务器的当前运行状态（参见表 24-5），可以使用下列 mysqladmin 命令。

```
$ mysqladmin -u root -p status
Enter password:
Uptime: 33874 Threads: 1 Questions: 196 Slow queries: 0 Opens: 27
Flush tables: 1 Open tables: 18 Queries per second avg: 0.006
$
```

## 24.6.2 恢复数据库管理员密码

如果忘记了 MySQL 数据库管理员 root 的密码，如同忘记了 Linux 系统超级用户 root 的密码一样，无法正常地管理与维护 MySQL 数据库。要恢复 root 用户密码，可以采用下列步骤处理。

(1) 终止 MySQL 数据库的守护进程 mysqld，命令如下。

```
$ sudo /etc/init.d/mysql stop
* Stopping MySQL database server mysqld                                [ OK ]
```

(2) 以安全模式启动 MySQL 数据库的 mysqld\_safe 守护进程，从而绕过密码检测与验证阶段，命令如下。

```
$ sudo mysqld_safe --skip-grant-tables --skip-networking &
[1] 6896
$ nohup: ignoring input and redirecting stderr to stdout
Starting mysqld daemon with databases from /var/lib/mysql
mysqld_safe[9235]: started
$
```

(3) 此时，MySQL 数据库处于没有密码保护的运行状态，利用“mysql -u root”命令，可以直接进入 MySQL 服务器，获得“mysql>”命令提示符（中间不会提示用户输入密码），如下所示。

```
$ mysql -u root
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 1
Server version: 5.0.67-Ubuntu6 (Ubuntu)

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql>
```

(4) 选用 mysql 数据库，修改 user 数据库表（其中含有所有用户的密码），即修改数据库管理员 root 的用户密码，如改为 a1b2c3；然后退出 MySQL 数据库服务器，命令如下。

```
mysql> USE mysql;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> update user set password=password("a1b2c3") where user="root";
Query OK, 3 rows affected (0.00 sec)
Rows matched: 3  Changed: 3  Warnings: 0

mysql> flush privileges;
mysql> exit
Bye
$
```

(5) 利用 kill 命令，终止 mysqld\_safe 守护进程，然后重新启动 MySQL 服务器守护进程 mysqld，如下所示。

```
$ sudo kill -9 6896
$ sudo /etc/init.d/mysql start
* Starting MySQL database server mysqld                                [ OK ]
```

(6) 至此，即可使用 root 用户的新密码访问 MySQL 数据库，如下所示。

```
$ mysql -u root -p
Enter password:
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 3
Server version: 5.0.67-Ubuntu6 (Ubuntu)

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql>
```

### 24.6.3 基本网络安全考虑

通常，MySQL 服务器守护进程会监听服务器所有网络接口的 3306 端口，接收任何远程 MySQL 客户端的数据库访问请求。例如，利用下列 netstat 命令，可以验证 MySQL 守护进程监听的 IP 地址与端口号分别为 0.0.0.0 和 3306，表示可以接受来自任何 IP 地址的数据库访问请求。

```
$ netstat -an | grep 3306
tcp      0      0      0.0.0.0:3306      0.0.0.0:*      LISTEN
$
```

如果 MySQL 数据库仅供内部网络或本地系统中的客户端应用访问，可以限定 MySQL 数据库服务器监听的网络接口，如仅监听环回接口。为此，可编辑/etc/my.cnf 配置文件，修改[mysqld] 节中的 bind-address 配置指令，赋予特定的 IP 地址，如下所示。

```
[mysqld]
bind-address=127.0.0.1
```

然后，重新启动 MySQL 数据库服务器。注意，每当修改配置文件时，都需要重新启动 MySQL 数据库服务器，使修改后的配置立即生效。此时，再运行上述 netstat 命令将会发现，MySQL 仅监听本地系统环回接口的 3306 端口，如下所示。

```
$ netstat -an | grep 3306
tcp      0      0      127.0.0.1:3306      0.0.0.0:*      LISTEN
$
```

## 参 考 文 献

- 1 William von Hagen. Ubuntu Linux Bible. Wiley Publishing, Inc., 2007
- 2 Mark G. Sobell. A Practical Guide to Ubuntu Linux. Prentice Hall, 2008
- 3 Mark G. Sobell. A practical Guide to Linux Commands, Editors, and Shell Programming. Prentice Hall PTR, 2005
- 4 Chris Negus. Linux Bible 2007 Edition. Wiley Publishing, Inc., 2007
- 5 Peter Harrison. Linux Quick Fix Notebook. Prentice Hall PTR, 2005
- 6 Keir Thomas. Beginning Ubuntu Linux From Novice to Professional. APress, 2006
- 7 Kenneth Rosen. UNIX: The Complete Reference, Second Edition. McGraw-Hill, 2007
- 8 Ubuntu Linux Documentations (<https://help.ubuntu.com>)
- 9 BIND 9 Administrator Reference Manual (<http://www.isc.org>)
- 10 Apache Documentation (<http://www.apache.org>)
- 11 Mendel Cooper. Advanced Bash-Scripting Guide (<http://download.csdn.net/source/949704>)
- 12 Design and Implementation of the Second Extended Filesystem (<http://e2fsprogs.sourceforge.net/ext2intro.html>)

TP316.89 X609  
Ubuntu权威指南 邢国庆，张广利，  
邹浪编著  
C2392577B