应用型人才培养校企合作双元教材
新时代“互联网 + 教育”可视化教程

GNU&Linux基础应用教程

– 以CentOS 7为例

主　编：莫兴福
副主编：余伟红　唐遥芳　陈若梦
区铁毅　关家堡

湖南师范大学出版社
· 长沙 ·

图书在版编目（CIP）数据

GNU&Linux 基础应用教程 – 以 CentOS 7 为例 / 莫兴福主编 .—长沙：湖南师范大学出版社 , 2021.9
ISBN 978–7–5648–4300–7

Ⅰ. ① G… Ⅱ. ①莫… Ⅲ. ① Linux 操作系统 – 程序设计 – 高等学校 – 教材 Ⅳ. ① TP316.85

中国版本图书馆 CIP 数据核字 (2021) 第 165070 号

GNU&Linux 基础应用教程 – 以 CentOS 7 为例
GNU&Linux JICHU YINGYONG JIAOCHENG - YI CentOS 7 WEILI
◇主　　编：莫兴福

◇责任编辑：廖小刚　吴玉梅
◇校对编辑：胡晓军
◇出版发行：湖南师范大学出版社
地址/长沙市岳麓山
邮编//410081
电话/0731–88872751
传真/0731–88872636
网址/http://press.hunnu.edu.cn
◇经销：全国新华书店
◇印刷：长沙长大成彩印有限公司
◇开本：787mm×1092mm　1/16
◇印张：17.5
◇字数：480 千字
◇版次：2021 年 9 月第 1 版
◇印次：2021 年 9 月第 1 次印刷
◇书号：ISBN 978–7–5648–4300–7
◇定价：45.00 元

前　言

Linux 是一种计算机操作系统，由多种基础程序构成。它使计算机可以与用户进行交流并接受指令，读取数据或将其写入硬盘、磁带或打印机，控制内存的使用，以及运行其他软件。操作系统最重要的组成部分是内核，在 GNU/Linux 系统中，Linux 就是内核组，该系统的其余部分主要是由 GNU 工程编写和提供的程序组成。因为单独的 Linux 内核并不能成为一个可以正常工作的操作系统，所以我们更倾向使用“GNU/Linux”来表达人们通常所说的“Linux”。

本教材以 CentOS 7.4 为平台，依据学生实际情况和行业发展，对原有教学资源进行选择、整合和拓展，以满足当前岗位对 Linux 系统应用的需求。本教材是以学校为主体，与广州粤嵌通信科技股份有限公司共同合作开发的校企合作教材，主要是对 Linux 网络操作系统的应用进行详细讲解。本教材主要包括 Linux 操作系统的安装及配置、文件管理与常用命令、用户和组的管理、文件系统管理、GNU/Linux 网络配置、Vim 与 shell 脚本程序设计、CentOS 7 软件包管理、磁盘管理、进程管理、网络信息安全等内容。

本教材在编写过程中，始终遵循“适用、够用、会用、好用”的原则，强调理论和实践并重，突出先进性和科学性，体现实用性和可操作性。本教材每章节前都设“内容简介”和“学习要求”，每章末尾都设“本章小结”和“习题”等。

本教材由莫兴福老师担任主编，由余伟红、唐遥芳、陈若梦、区铁毅、关家堡老师担任副主编。

出版社同仁在本教材的出版过程中给予了大力支持，为本教材的出版花费了大量的时间和精力。在编写本教材的过程中，我们也参考了大量的著作、教材。在此，我们表示衷心感谢。

我们全体编写人员虽然尽心尽力，但由于时间仓促，加之编者水平有限，新的知识和技术资料不断涌现，书中难免有错误和疏漏之处，敬请广大师生及各位读者给予批评和指正。编者的 E-mail 地址是：hingfook@yeah.net。

编　者

2021 年 6 月

目　录

第一章　Linux 操作系统的安装及配置 / 01

1.1　认识 Linux 操作系统 / 01

1.1.1　Linux 发展的前因后果 / 01

1.1.2　认识 GNU 计划 / 02

1.1.3　Linux 与 Windows 的关系 / 03

1.2　在 VMWARE 安装 CentOS 7 / 05

1.3　CentOS 7 开机运行级别和设置 / 12

综合实训 / 15

第二章　文件管理与常用命令 / 16

2.1　Linux 文件系统 / 16

2.2　Linux 文件类别 / 20

2.3　Linux 目录结构概述 / 24

2.4　Linux 的绝对路径与相对路径 / 30

2.5　文件与目录基本操作 / 31

2.5.1　显示目录内容与路径及进入目录命令 / 32

2.5.2　目录的创建与删除命令 / 35

2.5.3　文件的复制、移动和删除命令 / 38

2.5.4　文件的新建命令 / 40

2.5.5　文件内容查看命令 / 42

2.5.6　文件查找命令 / 47

2.6　文件 / 目录访问权限管理 / 51

2.6.1　查看文件 / 目录的访问权限及用户与组 / 51

2.6.2　改变文件 / 目录的文件拥有者与所属组 / 53

2.6.3　更改文件 / 目录的所有权限 / 56

2.6.4　更改文件 / 目录的默认权限 / 59

2.7　文件的归档、压缩与解压 / 62

2.7.1　用 gzip 对文件进行压缩与解压缩 / 63

2.7.2　用 bzip2 和 bunzip 对文件进行压缩与解压缩 / 67

2.7.3 用 tar 命令进行文件归档 / 71
综合实训 / 80

第三章 用户和组的管理 / 81

3.1 用户账号和群组概述 / 81
3.1.1 用户账号 / 82
3.1.2 群组 / 82
3.2 用户与组文件 / 83
3.2.1 用户账号文件 / 83
3.2.2 用户影子文件 / 83
3.2.3 用户组账号文件 / 84
3.2.4 验证用户和组文件 / 85
3.3 管理用户账号 / 85
3.3.1 添加用户 / 86
3.3.2 修改用户信息 / 86
3.3.3 删除用户 / 87
3.4 管理群组 / 87
3.4.1 创建用户组 / 87
3.4.2 修改用户组属性 / 88
3.4.3 删除用户组 / 88
3.5 使用管理器管理用户和组 / 88
3.5.1 启动用户管理者 / 89
3.5.2 创建用户 / 89
3.5.3 修改用户属性 / 89
3.5.4 创建用户组 / 90
3.5.5 修改用户组属性 / 90
3.6 常用的账户管理命令 / 90
综合实训 / 94

第四章 文件系统管理 / 95

4.1 文件系统基础知识 / 95
4.1.1 硬盘结构 / 96
4.1.2 系统引导 / 96
4.1.3 硬盘的分区、逻辑卷和文件系统之间的关系 / 97
4.1.4 虚拟文件系统 / 97
4.1.5 文件的概念 / 98
4.1.6 日志文件系统 / 98
4.1.7 FAT 文件系统 / 99

4.1.8 Ext4 文件系统 / 99
4.1.9 XFS 文件系统 / 100
4.1.10 procfs / 100
4.1.11 交换空间 / 101
4.1.12 LVM / 102
4.1.13 文件系统存在的形态 / 102
4.2 一切皆文件 / 102
4.2.1 GNU/Linux 的文件系统目录结构 / 103
4.2.2 XDG 基本目录规范 / 105
4.2.3 一切皆文件 / 105
4.2.4 主流发行版之差异 / 106
4.2.5 符号链接 / 108
4.2.6 设备的符号链接 / 109
4.2.7 硬链接 / 109
4.2.8 管道命令符 / 110
4.2.9 特别的文件 / 112
4.2.10 伪设备 / 113
4.3 重定向 / 113
4.3.1 文件描述符 / 113
4.3.2 输入重定向 / 114
4.3.3 输出重定向 / 114
4.4 文件的权限 / 116
4.4.1 文件用户的层级 / 116
4.4.2 文件的权限 / 117
4.4.3 图形化管理 / 118
4.4.4 文件权限的变更 / 118
4.4.5 直接指定文件的权限 / 119
4.5 文件的属性 / 120
综合实训 / 122
第五章 GNU/Linux 网络配置 / 124
5.1 网络配置文件 / 124
5.1.1 /etc/hosts 文件 / 125
5.1.2 /etc/resolv.conf 文件 / 126
5.1.3 /etc/host.conf 文件 / 126
5.1.4 /etc/sysconfig/network-scripts/ifcfg-X 文件 / 127
5.1.5 /etc/services 文件 / 128
5.1.6 /etc/sysconfig/network 文件 / 129

5.2 常用的网络配置命令 / 130
5.2.1 网络配置命令 ifconfig / 130
5.2.2 hostname 配置主机名 / 132
5.2.3 ping 命令 / 134
5.2.4 route 命令 / 136
5.2.5 arp 命令 / 139
5.2.6 traceroute 命令跟踪路由 / 141
5.2.7 使用 setup 命令 / 143
5.3 常用的网络测试工具 / 144
5.3.1 使用 netstat 命令 / 144
5.3.2 systemd 和 systemctl / 146
综合实训 / 149

第六章 Vim 与 Shell 脚本程序设计 / 151
6.1 Vim 文本编辑器基础知识 / 151
6.1.1 Vim 文本编辑器的三种工作模式 / 152
6.1.2 Vim 的基本使用 / 153
6.1.3 Vim 的常用命令 / 156
6.2 重定向、管道和环境变量 / 158
6.2.1 输入输出重定向 / 158
6.2.2 管道命令 / 159
6.2.3 命令行通配符 / 160
6.2.4 变量与环境变量 / 161
6.3 Shell 程序设计 / 162
6.3.1 Shell 脚本的基本语法结构 / 163
6.3.2 流程控制语句 / 164
综合实训 / 176

第七章 CentOS 7 软件包管理 / 178
7.1 使用 rpm 命令管理 RPM 包 / 178
7.1.1 RPM 软件包参数说明 / 178
7.1.2 RPM 软件包的查询 / 179
7.1.3 RPM 软件包的安装 / 183
7.1.4 RPM 软件包安装可能出现的问题 / 183
7.1.5 RPM 软件包的卸载 / 183
7.1.6 RPM 软件包的升级 / 184
7.1.7 RPM 软件包的验证 / 184
7.2 使用 RPM 软件包管理器 / 186
7.2.1 打开软件包管理器 / 187

7.2.2 添加删除软件 / 187
7.2.3 其他软件包管理器 / 187
7.3 安装 gcc 并编写 C 语言文件 / 189
7.4 yum 的使用 / 190
综合实训 / 194

第八章 磁盘管理 / 195

8.1 磁盘管理命令 / 195
8.1.1 磁盘分区和挂载 / 196
8.1.2 mount 与 umount / 202
8.1.3 常用的磁盘管理命令 / 203
8.2 磁盘配额 / 207
8.2.1 磁盘配额的限制对象 / 207
8.2.2 磁盘配额的限制类型 / 208
8.2.3 磁盘配额的限制方法 / 208
8.2.4 磁盘配额大致可分为 4 个步骤 / 208
8.3 RAID 的使用 / 211
8.3.1 RAID 概述 / 211
8.3.2 基本原理 / 212
8.3.3 RAID 等级 / 213
8.3.4 创建 RAID5 / 215
8.4 配置与管理逻辑卷 / 222
8.4.1 LVM 的基本组成块（building blocks） / 222
8.4.2 创建 LVM / 224
8.4.3 扩容 LVM 逻辑卷 / 229
8.4.4 缩小 LVM 逻辑卷 / 233
综合实训 / 236

第九章 进程管理 / 238

9.1 进程概述 / 238
9.1.1 程序、进程与线程 / 238
9.1.2 进程的类型 / 240
9.1.3 常见的 GNU/Linux 系统进程 / 240
9.2 进程管理及其常用命令 / 241
9.2.1 查看和监视系统进程 / 241
9.2.2 进程调度命令 kill / 244
9.2.3 作业管理 / 246
综合实训一 / 247
综合实训二 / 248

第十章　网络信息安全 / 249

10.1　防火墙技术 / 249

10.1.1　防火墙概述 / 249

10.1.2　包过滤防火墙与网络地址转换 NAT / 250

10.1.3　Linux 防火墙技术 / 252

10.2　iptables 服务配置防火墙 / 259

10.2.1　iptables 基础知识 / 259

10.2.2　iptables 命令 / 261

综合实训 / 265

第一章

Linux 操作系统的安装及配置

内容简介

- Linux 是知名和常用的开源操作系统。作为一个操作系统，Linux 是一个软件，位于计算机上的所有其他软件的下面，从这些程序接收请求并将这些请求转发到计算机硬件。我们使用术语“Linux”来指代 Linux 内核，也是通常与 Linux 内核捆绑在一起的程序、工具和服务，以提供所有必需的组件全功能操作系统。有些人，特别是自由软件基金会的成员，将此集合称为 GNU / Linux，因为包括的许多工具都是 GNU 组件。但是，并不是所有的 Linux 安装都使用 GNU 组件作为其操作系统的一部分。例如，Android 使用 Linux 内核，但对 GNU 工具依赖性很低。那么我们就知道了，通常我们说的“Linux”其实是指 Linux 内核，而 Linux 操作系统其实是 GNU/Linux（Linux 内核 +GNU 组织的软件）。

学习要求

- 通过本章的学习，了解 C 语言、UNIX、GNU、Linux 的历史与由来，了解自由软件的主张和思想。

1.1 认识 Linux 操作系统

1.1.1 Linux 发展的前因后果

Linux 是一套自由加开放源代码的类 Unix 操作系统，由就读于芬兰赫尔辛基大学的林纳斯·托瓦兹（Linus Torvalds）在一些网友的帮助下共同开发完成的。

Linux 操作系统的诞生、发展和成长过程始终依赖着五个重要支柱——UNIX 操作系统、MINIX 操作系统、GNU 计划、POSIX 标准和 Internet 网络。

Linux 是一个基于 POSIX 和 Unix 的多用户、多任务、支持多线程和多 CPU 的操作系统。20 世纪 60 年代，MIT 开发分时操作系统（Compatible Time-Sharing System），支持 30 台终端访问主机，主机负责运算，而终端负责输入输出。

1965 年，Bell 实验室、MIT、GE（通用电气公司）为了同时支持 300 个终端访问主机，准备开发 Multics 系统，但是于 1969 年失败。刚开始并没有鼠标、键盘，输入设备只有卡片机，因此如果要测试某个程序，则需要将读卡纸插入卡片机，如果有错误，还需要重新来过。

1969 年，Ken Thompson（C 语言之父）利用汇编语言开发了 File Server System（Unics，即 Unix 的原型），因为汇编语言对于硬件的依赖性，因此只能针对特定的硬件。

1973 年，Dennis Ritchie 和 Ken Thompson 发明了 C 语言，而后写出了 Unix 的内核，将 B 语言改成 C 语言，由此产生了 C 语言之父。90% 的代码用 C 语言写的，10% 的代码用汇编语言写的，因此移植时只要修改那 10% 的代码即可。

1977 年，Berkeley 大学的 Bill Joy 针对他的机器修改 Unix 源码，称为 BSD（Berkeley Software Distribution）。Bill Joy 是 Sun 公司的创始人。

1979 年，Unix 发布 System V，用于个人计算机。

1984 年，因为 Unix 规定“不能对学生提供源码”，所以 Tanenbaum 老师自己编写兼容于 Unix 的 Minix，用于教学。

1984 年，Stallman 开始 GNU（GNU's Not Unix）项目，创办 FSF（Free Software Foundation）基金会，产品有 GCC.Emacs、Bash Shell、GLIBC，倡导“自由软件”。GNU 的软件缺乏一个开放的平台运行，只能在 Unix 上运行。自由软件指用户可以对软件做任何修改，甚至再发行，但是始终要挂着 GPL 的版权；自由软件是可以卖的，但是不能只卖软件，而且要卖服务、手册等。

图 1-1　林纳斯・托瓦兹

1985 年，为了避免 GNU 开发的自由软件被其他人用作专利软件，因此创建 GPL（General Public License）版权声明。

1988 年，MIT 为了开发 GUI，成立了 XFree86 的组织。

1991 年，Linus Torvalds 基于 gcc、bash 开发了针对 386 机器的 Linux 内核。

1994 年，Linus Torvalds 发布 Linux-v1.0。

1996 年，Linus Torvalds 发布 Linux-v2.0，确定了 Linux 的吉祥物——企鹅。

1.1.2　认识 GNU 计划

GNU 是“GNU's Not Unix”的递归缩写。UNIX 是一种广泛使用的商业操作系统的名称。由于 GNU 将要实现 UNIX 系统的接口标准，因此 GNU 计划可以分别开发不同的操作系统部件。GNU 计划采用了部分当时已经可自由使用的软件，例如 TeX 排版系统和 X Window 视窗系统等。不过 GNU 计划也开发了大批其他的自由软件。为保证 GNU 软件可以自由地“使用、复制、修改和发布”，所有 GNU 软件都有一份禁止其他人在添加任何限制的情况下授权所有权利给任何人的协议条款，GNU 通用公共许可证（GNU General Public License，GPL，如图 1-2 所示）。这个就是被称为“反版权”（或称

Copyleft）的概念。

图 1-2　GPL 标志

GNU 是一个类 Unix 操作系统。它是由多个应用程序、系统库、开发工具乃至游戏构成的程序集合。

GNU 计划，又称革奴计划，是由 Richard Stallman 在 1983 年 9 月 27 日公开发起的。它的目标是创建一套完全自由的操作系统。Richard Stallman 最早是在 net.unix-wizards 新闻组上公布该消息的，并附带一份《GNU 宣言》等解释为何发起该计划的文章，其中一个理由就是要“重现当年软件界合作互助的团结精神”。

GNU 是一个自由软件操作系统，就是说，它尊重其使用者的自由。GNU 操作系统包括 GNU 软件包（专门由 GNU 工程发布的程序）和由第三方发布的自由软件。GNU 的开发使你能够使用电脑而无须安装可能会侵害你自由的软件。

什么是自由软件运动？自由软件运动致力于通过自由软件使计算机用户获得自由权利。自由软件的用户可以自主控制自己的计算机。非自由软件使用户受制于软件开发者。

什么是自由软件？自由软件意味着使用者有运行、复制、发布、研究、修改和改进该软件的自由。自由软件是权利问题，不是价格问题。要理解这个概念，你应该考虑“free”是“言论自由（free speech）”中的“自由”，而不是“免费啤酒（free beer）”中的“免费”。

更精确地说，自由软件赋予软件使用者四项基本自由：

- 不论目的为何，有运行该软件的自由（自由之零）。
- 有研究该软件如何工作以及按需改写该软件的自由（自由之一）。取得该软件源代码为达成此目的之前提。
- 有重新发布拷贝的自由，这样你可以借此来敦亲睦邻（自由之二）。
- 有向公众发布改进版软件的自由（自由之三），这样整个社群都可因此受惠。取得该软件源代码为达成此目的之前提。

1.1.3　Linux 与 Windows 的关系

在接触 Linux 之前，应该先接触的都是 DOS 与 Windows 吧？但一般我们接触 Linux 后，特别是习惯 Linux 的管理和使用方法后，我们再回过头来使用 Windows 的时候，内心其实是拒绝的。我们会觉得图形好麻烦，这个时候我们差不多是一个 Linux 的重度爱好者了。

和 Linux 一样，Windows 系列是完全的多任务操作系统。它们支持同样的用户接口、网络和安全性。但是，Linux 和 Windows 的真正区别在于，Linux 事实上是 Unix 的一种版本，而且来自 Unix 的贡献非常巨大。是什么使得 Unix 如此重要？在于对多用户机器来说，不仅 Unix 是最流行的操作系统，而且在于它是免费软件的基础。在 Internet 上，大量免费软件都是针对 Unix 系统编写的。由于有众多的 Unix 厂商，所以 Unix 也有许多实现方法。没有一个单独的组织负责 Unix 的分发。现在，存在一股巨大的力量推动 Unix 社团以开放系统的形式走向标准化。另一方面 Windows 系列是专用系统，由开发操作系统的公司控制接口和设计。在这个意义上来说，这种公司的利润很高，因为它对程序设计和用户接口设计建立了严格的标准，和那些开放系统社团完全不一样。一些组织正在试图完成标准化 Unix 程序设计接口的任务。特别要指出的是，Linux 完全兼容 POSIX.1 标准。

对于用户来说，Linux 和 Windows 的不断更新引发了两者之间的竞争。用户可以有自己喜欢的系统，同时也在关注竞争的发展。微软的主动性似乎更高一些，这是由于业界“冷嘲热讽”的“激励”与 Linux 的不断发展。最显著的表现是：微软更加关注改进可用性的同时增强系统的安全性。比如：2003 年许多针对微软的漏洞攻击程序都使用可执行文件作为电子邮件的附件（例如 MyDoom）。Service Pack2 包括一个附件执行服务，为 Outlook/Exchange、 Windows Messenger 和 Internet Explorer 提供了统一的环境。这样就能降低用户运行可执行文件时感染病毒或者蠕虫的威胁性。另外，禁止数据页的可执行性也会限制潜在的缓冲区溢出的威胁。不过，微软在 WindowsXP Service Pack2 中并没有修改 Windows 有问题的架构以及安全传输的部分，而是将这部分重担交给了用户。

Linux 的应用目标是网络而不是打印，Windows 最初出现的时候，这个世界还是一个纸张的世界。Windows 的伟大成就之一在于你的工作成果可以方便地看到并打印出来。这样一个开端影响了 Windows 的后期发展。

同样，Linux 也受到了其起源的影响。Linux 的设计定位于网络操作系统。它的设计灵感来自于 Unix 操作系统，因此它的命令设计比较简单，或者说是比较简洁。由于纯文本可以非常好地跨网络工作，所以 Linux 配置文件和数据都以文本为基础。

对于那些熟悉图形环境的人来说，Linux 服务器初看可能比较原始。但是 Linux 开发更多关注的是它的内在功能而不是表面上的东西。即使是在纯文本的环境中，Linux 同样拥有非常先进的网络、脚本和安全能力。执行一些任务所需的某些表面上看起来比较奇怪的步骤是令人费解的，除非你认识到 Linux 是期望在网络上与其他 Linux 系统协同执行这些任务。Linux 的自动执行能力也很强，只需要设计批处理文件就可以让系统自动完成非常详细的任务。Linux 的这种能力来自于其基于文本的本质。

1. 软件与支持

Windows 平台：数量和质量的优势，不过大部分为收费软件；由微软官方提供重要支持和服务。

Linux 平台：大都为开源自由软件，用户可以修改定制和再发布，由于基本免费没有资金支持，部分软件质量和体验欠佳；由全球所有的 Linux 开发者和自由软件社区提供支持。

2. 安全性

Windows 平台：三天两头打补丁安装系统安全更新，有时还是会中病毒木马。

Linux 平台：要说 Linux 没有安全问题，那当然是不可能的。相对来说，肯定比 Windows 平台更加安全，而且使用 Linux，你也不用装杀毒软件了。

3. 使用习惯

Windows 平台：普通用户基本都是在纯图形界面下操作使用，依靠鼠标和键盘完成一切操作，用户上手容易、入门简单。

Linux 平台：兼具图形界面操作（需要使用带有桌面环境的发行版）和完全的命令行操作，可以只用键盘完成一切操作，新手入门较困难，需要一些学习和指导（这正是我们要做的事情），一旦熟练之后效率极高。

4. 可定制性

Windows 平台：基本上是全封闭的，系统可定制性很差。

Linux 平台：你想怎么做就怎么做，Windows 能做到的它都能，Windows 做不到的，它也能。

5. 应用范畴

或许你之前不知道 Linux ，但是要知道，你之前在 Windows 环境下使用百度、谷歌，上淘宝，聊 QQ 时，支撑这些软件和服务的，是后台成千上万的 Linux 服务器主机，它们时时刻刻都在进行着忙碌的数据处理和运算，可以说世界上大部分软件和服务都是运行在 Linux 之上的。

6. 开源

开源就是指对外部开放软件源代码。如果一个小程序员写了个软件，里面有他独创的新技术，他想靠这个赚钱，甚至还为此申请了专利，这时某些团体以安全为由，要求他公开源代码（这样就可以仿制），并且最好免费给大家使用，身边一群眼红程序员赚钱的人也在跟着起哄。开源与否，软件厂商有选择的权利；是否购买使用这个产品，这才是用户的权利。其实想深一点，如果 Linux 不开源，它还能有现在这个市场吗？因为很多人就是冲着开源才使用 Linux 的。

7. 服务器市场

Linux 是一个与 UNIX 相像的操作系统，它拥有 UNIX 的安全性和稳定性，当然还有网络支持能力，它比 UNIX 更加优秀的是它的友好界面。对于 IT 网络管理人员来说，它比 UNIX 更易部署和管理；对于开发人员来说，它是一个可以随心所欲改变的操作系统。Linux 因为类 UNIX 的架构的关系，比 Windows Server 更稳定，而且从 Windows Server 价格来看，Linux 价格低廉，服务器操作系统市场 Linux 占有率比 Windows 要高。

8.Windows 没有的

稳定的系统，安全性和漏洞的快速修补；多用户，用户和用户组的规划；相对较少的系统资源占用，可定制裁剪，移植到嵌入式平台（如安卓设备）；可选择的多种图形用户界面（如 GNOME，KDE）。

9.Linux 没有的

没特定的支持厂商；游戏娱乐支持度不足；专业软件支持度不足。

1.2　在 VMWARE 安装 CentOS 7

在 Windows 下安装 Linux 虚拟机后要进行虚拟机配置，如图 1-3 所示。

图 1-3　VMWARE 的内存配置

选择安装方式：第一行：安装 CentOS 7；第二行：测试这个媒体并安装 CentOS 7；第三行：故障排除。如图 1-4 所示。

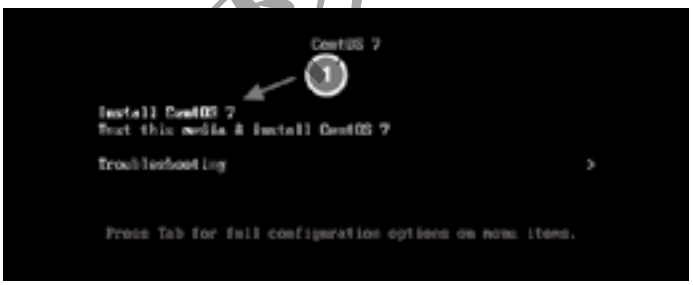

图 1-4　选择安装方式

CentOS 7 与 CentOS 6 网卡名称命名方式有所改变，如果需要 CentOS 7 与 CentOS 6 的网卡名称一样，操作步骤如下：

选择第一个安装方式，并按下 tab 键，输入 net.ifnames=0 biosdevname=0。如图 1-5 所示。

图 1-5　选择第一栏

选择语言：这里直接选择中文，安装配置方式如图 1-6 所示。

图 1-6　选择语言

配置系统安装信息：本地化可以忽略；语言选择英文的注意更改时区为上海，不然会出现其他时区时间。修改顺序如图 1-7 所示。

图 1-7　修改顺序

配置系统分区如图 1-8 所示。

图 1-8　配置系统分区

推荐使用自定义分区设置，具体可根据实际情况更改；/boot 分区 1G；swap 分区 2G；/ 分区：剩余全部空间。如图 1-9 所示。

图 1-9　增加标准分区

/ 分区：剩余全部空间（如图 1-10、图 1-11 所示）。

图 1-10　添加新挂载点

图 1–11　添加 swap 分区

分区配置好如图 1–12 所示。

图 1–12　选择文件系统

CentOS 7.x 文件系统采用的是 xfs，检查是否为 xfs；CentOS 6.x 文件系统采用的是 ext4；swap 不用更改文件系统；接受分区更改。如图 1–13 所示。

图 1–13　分区摘要

关闭 KDUMP（如图 1–14 所示）：在虚拟机中这个功能有些鸡肋，还会拖慢虚拟机，而且虚拟机有快照功能，玩坏了恢复快照即可。

图 1–14　关闭 KDUMP

配置网卡信息：修改虚拟机的虚拟网卡信息，这样虚拟机就可以上外网了。如图 1–15 所示。

图 1–15　修改虚拟机的虚拟网卡信息

选择 VMnet8，关闭 DHCP 服务，如图 1–16、图 1–17 所示。

图 1–16　选择 VMnet8NAT 模式

图 1–17　设置网关 IP

修改主机名和配置第一块网卡，如图 1–18 所示。

图 1–18　修改主机名和配置第一块网卡

修改连接名称和 DNS 服务器如图 1-19、图 1-20 所示。

图 1-19　修改连接名称

图 1-20　修改 DNS 服务器

配置好后，网卡会被激活；第二块网卡同样配置方法，也可以在系统中去修改；关闭 security policy（如图 1-21 所示）：关闭安全策略，当熟练 CentOS 时，可以去试着了解这个机制。

图 1-21　关闭 security policy

定制系统环境：这里选择最小化安装，有需求的可以安装其他熟悉的基本环境，如图 1-22 所示。

图 1-22　定制系统环境

选择好后就开始安装 CentOS 7，如图 1-23 所示。

图 1-23　设置 root 密码并开始安装

安装时配置好 root 密码，需要设置简单密码确认两次，才可以将密码设置成功，安装包数量大概 471 个，因系统而定；个人使用配置，根据个人喜好设置，虚拟机环境，尽量简化，操作方便，出现问题时，请断开连接重新连接即可。

1.3　CentOS 7 开机运行级别和设置

Linux 分为 7 个启动级别：

0 - 系统停机状态	halt
1 - 单用户工作状态	Single user mode
2 - 多用户状态（没有 NFS）	Multiuser, without NFS (The same as 3, if you do not have networking)
3 - 多用户状态（有 NFS）	Full multiuser mode
4 - 系统未使用，留给用户	unused
5 - 图形界面	X11
6 - 系统正常关闭并重新启动	reboot (Do NOT set initdefault to this)

现在很多 Linux 系统都默认启动等级为 5。如果我们想切换到多用户状态且带有网络文件系统——命令行模式，可使用 init 命令：init 3 即可。

Linux 中一切皆文件，要想永久保存使用，需将相关设置写入文件中。

而在 CentOS 7 在 /etc/inittab 略有不同：

1. 其运行级别对应关系

init level	systemctl target
0	shutdown.target

1	emergency.target
2	rescure.target
3	multi-user.target
4	无
5	graphical.target
6	无

runlevel 命令：例如：[mk@localhost ~]$ runlevel

N 5　# 如上结果：第二个数字表示当前的运行级别

2. 运行级别设置

（1）语法：systemctl [command] [unit.target]。

（2）命令及参数：command 部分；get-default：获取当前的 target；set-default：将默认运行级别设置为指定的 target；isolate：切换至指定的运行级别；unit.target 部分：为上面 1 节部分中给出的运行级别。

3. 常用运行级别相关命令

（1）systemctl

Ll /etc/systemd/system/default.target：# 查看默认目标链接文件详细信息；

systemctl get-default：# 获取当前的运行级别；

systemctl set-default multi-user.target：# 将默认运行级别设置为 mulit-user（字符模式）；

systemctl set-default graphical.target：# 将默认运行级别设置为 graphical（图形界面）；

systemctl isolate multi-user.target：# 不重启系统的情况下，将运行级别切换至 mulit-user；

systemctl isolate graphical.target：# 不重启系统的情况下，将运行级别切换至图形模式。

上述命令对 CentOS 7 也适用。但想修改默认启动级别，CentOS 7 同其他 Linux 就稍显不同，对于传统的设置方法，也可以用 init 3，直接运行改命令即可，如图 1-24 所示。

例如：ln -sf /lib/systemd/system/runlevel5.target /etc/systemd/system/default.target

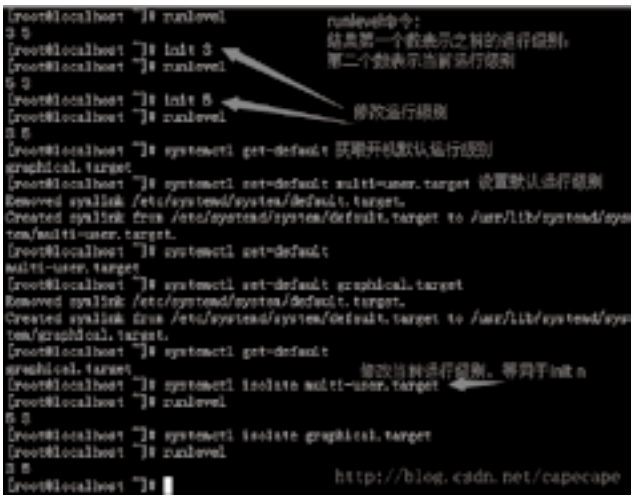

图 1-24　开机运行设置语句

本章小结

本章主要学习了 Linux 操作系统的历史，以及 Linux 操作系统的安装。

习题

一、简答题

1. 从图形界面转到完整字符界面的命令是什么？

2. 从字符界面转到图形界面的命令是什么？

二、论述题

1. 简述 Linux 的体系结构。

2. Linux 有哪些安装方式？

3. 简述 CentOS 7 开机运行级别设置。

4. 简述 X-Window 与 Windows 的区别。

✔ 综合实训

一、实训题目

安装 CentOS 操作系统。

二、实训目的

掌握 CentOS 操作系统的安装。

掌握分区。

三、实训内容

由于公司部分 Windows 服务器频繁遭受病毒、木马的威胁，同时鉴于 Linux 系统在服务器领域的稳定性，公司决定安装 CentOS 7 操作系统，并在该系统之上构建各种服务器。要求如下：

1. 在宿主机上安装 VM Workstation。

2. 安装 CentOS 7 虚拟机 server1，并进行网络配置，使之能连上互联网络。

3. 重置 server1 的管理员密码。

4. 克隆生成一个 CentOS 7 操作系统 client1，对该系统进行基本网络配置。

5. 对于 server1 和 client1，利用不同的网络连接方式，测试两台计算机的连通性，从而了解虚拟机中不同网络连接方式的不同。

第二章

文件管理与常用命令

内容简介

- Linux 支持多种文件系统，包括 ext2、ext3、vfat、ntfs、iso9660、jffs、romfs 和 nfs 等。而系统里又分很多种文件，一句话，在 Linux 里，一切都是文件，针对这些文件的处理，从而出现了好多命令。

学习要求

- 通过本章的学习，掌握文件管理与常用命令的知识，熟记各种命令。

2.1 Linux 文件系统

Windows 上的文件系统以 NTFS 和 Fat32 等为主。

Linux 的文件系统和 Windows 的文件系统有很大的不同，对于微软视窗系统的文件结构在这里不再赘述，我们主要了解一下 Linux 的文件系统结构。为了对各类文件系统进行统一管理，Linux 引入了虚拟文件系统 VFS（Virtual File System），为各类文件系统提供一个统一的操作界面和应用编程接口。

Linux 下常见的几种文件系统（包括 ext2、ext3、ext4、ZFS 和 Btrfs）对比如表 2-1 所示。

表 2-1 Linux 下几种文件系统的对比

文件系统	创建者	创建时间	最开始支持的平台
ext2	Rémy Card	1993	Linux,Hurd
XFS	SGI	1994	IRIX,Linux,FreeBSD
ext3	Dr. Stephen C. Tweedie	1999	Linux
ZFS	Sun	2004	Solaris

续表

文件系统	创建者	创建时间	最开始支持的平台
ext4	众多开发者	2006	Linux
Btrfs	Oracle	2007	Linux

从创建时间可以看出它们所处的不同时代，因为 Btrfs 的实现借鉴自 ZFS，所以这里也将 ZFS 列出来作为参考，如表 2-2 所示。

表 2-2　各种文件系统支持大小

文件系统	最大文件名长度	最大文件大小	最大分区大小
ext2	255 bytes	2 TB	16 TB
ext3	255 bytes	2 TB	16 TB
ext4	255 bytes	16 TB	1 EB
XFS	255 bytes	8 EB	8 EB
Btrfs	255 bytes	16 EB	16 EB

最大文件和分区大小受格式化分区时所采用的块大小（block size）所影响，块越大，所支持的最大文件和分区越大，也越可能浪费磁盘空间，上表列出的数据基于 4K 的块大小。

代码规模（如表 2-3 所示）：从代码规模可以看出文件系统的功能丰富程度以及复杂度，下面列出的数据来自于 kernel-4.1-rc8，只是简单地用 wc-l 来统计，没有过滤空行、注释等。

表 2-3　Linux 文件系统的代码规模

文件系统	源文件 (.c)	头文件 (.h)
ext2	8363	1016
ext3	16496	1567
ext4	44650	4522
XFS	89605	15091
Btrfs	105254	7933

Btrfs 还在快速地开发过程中，代码行数可能还有比较大的变化。

XFS 和 Btrfs 都使用了 B-tree。

ext2：优点是比较简单，文件比较少时性能较好，比较适合文件少的场景。主要缺点如下：inode 的数量是固定不变的，在格式化分区的时候可以指定 inode 和数据块所占空间的比例，但一旦格式化好了之后，后续就没法再改变。当块大小为 4K 时，单个文件大小不能超过 2TB，分区大小不能超过 16TB（目前硬盘大小一般都只有几 TB，所以也不是什么大问题）。一个目录下最多只能有 32000 个子目录。由于目录里面存储的文件和子目录都是以线性方式来组织的，所以遍历目录效率不高，尤其当目录下文件个数达到 10K 以上规模的时候，速度会明显地变慢。当底层的磁盘分区空间变大时（使用 LVM 时很常见），ext2 没法动态地扩展来使用增加的空间，没有日志（Journal）功能，所以数据的安全性不高。

ext3：在 ext2 的基础上实现了下面几个功能，其他的都保持不变，即 ext2 的缺点 ext3 也有，支持

日志（Journal）功能，数据的安全性较 ext2 有很大提高。当底层的分区空间变大时，ext3 可以自动扩展来使用增加的空间，使用 HTree 来组织目录里面的文件和子目录，使目录下的文件和子目录数不再受性能限制（数量超过 10K 也不会有性能问题）。

ext4：借鉴了当前成熟的一些文件系统技术，在 ext3 上增加了一些功能，并且对性能做了一些改进，主要变化如下：当块大小为 4K 时，支持的最大文件和最大分区大小分别达到了 16TB 和 1EB，不再受 32000 个子目录数的限制；支持不限数量的子目录个数；支持 Extents，提高了大文件的操作性能，内部实现了支持一次分配多个数据块，较 ext3 的性能有所提高；支持延时分配（支持 fallocate 函数，fallocate 是 libc 的函数，在不支持该功能的文件系统上，libc 会创建一个占用磁盘空间的文件）；支持在线快速扫描；支持在线碎片整理（单个文件或者整个分区）日志（Journal）；支持校验码（Checksum），数据的安全性进一步提高；支持无日志（No Journaling）模式（ext3 不支持该功能），这样就和 ext2 一样，消除了写日志对性能的影响；支持纳秒级的时间戳，记录了文件的创建时间，由于相关的应用层工具还不支持，所以只能通过 debug 的方式看到文件的创建时间。

这里是一个查看文件 /etc/fstab 创建时间的例子（文件存在 /dev/sda1 分区上）：

```
1. dev@ubuntu:~$ ls -i /etc/fstab
2. 10747906 /etc/fstab
3. dev@ubuntu:~$ sudo debugfs -R ‘stat <10747906>’ /dev/sda1
4. inode: 10747906   Type: regular    Mode:  0644   Flags: 0x80000
5. Links: 1   Blockcount: 8
6. ctime: 0x5546dc54:6e6bc80c -- Sun May  3 22:41:24 2015
7. atime: 0x55d1b014:8bcf7b44 -- Mon Aug 17 05:57:40 2015
8. mtime: 0x5546dc54:6e6bc80c -- Sun May  3 22:41:24 2015
9. crtime: 0x5546dc54:6e6bc80c -- Sun May  3 22:41:24 2015
10. Size of extra inode fields: 28
11. EXTENTS: (0):46712815
```


Extents：在最开始的 ext2 文件系统中，数据块都是一个一个单独管理的，inode 中存有指向数据块的指针，文件占用了多少个数据块，inode 里面就有多少个指针（多级），想象一下一个 1G 的文件，4K 的块大小，那么需要（1024 * 1024）/4=262144 个数据块，即需要 262144 个指针，创建文件的时候需要初始化这些指针，删除文件的时候需要回收这些指针，这样会影响性能。现代的文件系统都支持 Extents 的功能，简单地说，Extent 就是数据块的集合，以前一次分配一个数据块，现在可以一次分配一个 Extent，里面包含很多数据块，同时 inode 里面只需要分配指向 Extent 的指针就可以了，从而大大减少了指针的数量和层级，提高了大文件操作的性能。

inode 数量固定：在 ext2/3/4 系列的文件系统中，inode 的数量都是固定的，其缺点是如果存储很多小文件的话，就有可能造成 inode 被用光，但磁盘还有很多剩余空间无法被使用的情况；不过它也有一个优点，就是一旦磁盘损坏，恢复起来相对要简单些，因为数据在磁盘上的布局相对固定简单。

xfs：和 ext4 相比，xfs 不支持下面这些功能：

- 不支持日志（Journal）校验码。

• 不支持无日志（No Journaling）模式。

• 不支持文件创建时间。

• 不支持数据日志（Data Journal），只有元数据日志（Metadata Journal）。

但 xfs 有下面这些特性：

• 支持的最大文件和分区都达到了 8EB。

• inode 动态分配，从而不受 inode 数量的限制，再也不用担心存储大量小文件导致 inode 不够用的问题。

• 更大的 xattr（Extended Attributes）空间，ext2/3/4 及 btrfs 都限制 xattr 的长度不能超过一个块（一般是 4K），而 xfs 可以达到 64K。

• 内部采用 Allocation groups 机制，各个 group 之间没有依赖，支持并发操作，在多核环境的某些场景下性能表现不错。

• 提供了原生的 dump 和 restore 工具，并且支持在线 dump。

Btrfs：是一个和 ZFS 类似的文件系统，支持的功能非常多，据说将来会替换 ext4 成为 Linux 下的默认文件系统。这里列举一些重要的功能：

• 支持的最大文件和分区达到 16EB。

• 支持 COW（Copy on Write）。

• 针对小文件和 SSD 做了优化 inode 动态分配。

• 支持子分区（Subvolumes），子分区可以单独挂载。

• 支持元数据和数据的校验（crc32）。

• 支持压缩，去重。

• 支持多个磁盘和分区，可动态扩展。

• 支持 lvm，raid 的功能（有了 btrfs，就不再需要 lvm 和软 raid）。

• 增量备份和恢复。

• 支持快照。

• 将 ext2/3/4 转换成 btrfs（反过来不行）。

Btrfs 最大的缺点就是由于其 COW 的实现方式，导致碎片化问题比较严重，不太适合频繁写的场景，比如数据库、虚拟机的磁盘文件等。不过大部分场合不需要担心，btrfs 有在线的碎片整理工具。

Linux 文件系统的选择如表 2-4 所示。

表 2-4　Linux 文件系统的选择

文件系统	适用场景	原因
ext2	U 盘	U 盘一般不会存很多文件，且 U 盘的文件在电脑上有备份，安全性要求没那么高，由于 ext2 不写日志（Journal），所以写 U 盘性能比较好。当然由于 ext2 的兼容性没有 fat 好，目前大多数 U 盘格式还是用 fat
ext3	对稳定性要求高的地方	有了 ext4 后，好像没什么原因还要用 ext3，ext4 现在的问题是出来时间不长，还需要一段时间变稳定
ext4	小文件较少	ext 系列的文件系统都不支持 inode 动态分配，所以如果有大量小文件需要存储的话，不建议用 ext4

续表

文件系统	适用场景	原因
xfs	小文件多或者需要大的 xttr 空间，如 openstack swift 将数据文件的元数据放在了 xttr 里面	xfs 支持 inode 动态分配，所以不存在 inode 不够的情况，并且 xttr 的最大长度可以达到 64K
btrfs	没有频繁的写操作，且需要 btrfs 的一些特性	btrfs 虽然还不稳定，但支持众多的功能，如果你需要这些功能，且不会频繁地写文件，那么选择 btrfs

另外，ext 系列文件系统内部结构相对简单一些，出问题后恢复相对容易。

2.2 Linux 文件类别

与 Unix 一样，在 Linux 中，一切都是文件，但文件都得有类型。

在 Linux 上，任何软件和 I/O 设备都被视为文件，且 Linux 中的文件名最大支持 256 个字符，分别可以用 A~Z，a~z，0~9 等字符来命名。和 Windows 不同，Linux 的文件名是区分大小写的，所有的 UNIX 系列目录都遵循这个规则。Linux 下也没有盘符的概念（如 C 盘、D 盘），而只有目录，不同的硬盘分区是被挂载在不同目录下的。

此外，Linux 的文件没有扩展名，所以 Linux 下的文件名称和它的种类没有任何关系。例如，abc.exe 可以是文本文件，而 abc.txt 也可以是可执行文件。

那如何查看文件是什么类型？在 Linux 中可以使用以下命令：ls -l path

在显示文件的属性通常会以如下形式进行显示：drwxr-xr-x

第 1 个字母：代表文件类型

第 2~4 字母：代表用户的权限

第 5~7 字母：代表用户组的权限

第 8~10 字母：代表其他用户的权限

在 Linux 常见的文件类型有 7 种，如表 2-5 所示。

表 2-5　七种文件类型

文件属性	文件类型
-	常规文件，即 file
d	目录文件
b	block device 即块设备文件，如硬盘；支持以 block 为单位进行随机访问
c	character device 即字符设备文件，如键盘支持以 character 为单位进行线性访问
l	symbolic link 即符号链接文件，又称软链接文件
p	pipe 即命名管道文件
s	socket 即套接字文件，用于实现两个进程进行通信

示例如图 2-1 所示。

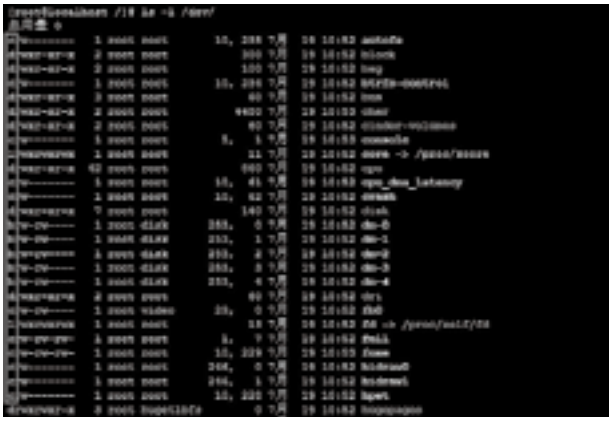

图 2-1　文件的首字母

1. 七种文件类型

普通文件类型：Linux 中最多的一种文件类型，包括纯文本文件（ASCII）、二进制文件（binary）、数据格式的文件（data）、各种压缩文件。第一个属性为 [-]。

目录文件：就是目录，能用 # cd 命令进入的。第一个属性为 [d]，例如 [drwxrwxrwx]。

块设备文件：就是存储数据以供系统存取的接口设备，简单而言就是硬盘。例如一号硬盘的代码是 /dev/hda1 等文件。第一个属性为 [b]，例如：sda，cdrom。

字符设备文件：即串行端口的接口设备，例如键盘、鼠标等。第一个属性为 [c]，例如：虚拟控制台或 tty0。

套接字文件：这类文件通常用在网络数据连接。可以启动一个程序来监听客户端的要求，客户端就可以通过套接字来进行数据通信。第一个属性为 [s]，最常在 /var/run 目录中看到这种文件类型，例如：我们启用 mysql 时，会产生一个 mysql.sock 文件。

管道文件：FIFO 也是一种特殊的文件类型，它主要的目的是解决多个程序同时存取一个文件所造成的错误。FIFO 是 first-in-first-out（先进先出）的缩写。第一个属性为 [p]。

链接文件：类似 Windows 下面的快捷方式。第一个属性为 [l]，例如 [lrwxrwxrwx]。

[root@xuegod63 ~]# ll /dev/sda /dev/cdrom /etc/passwd /dev/tty0 # 观察第一个字母，命令运行结果如下面四行，注意观察首字母：

```
lrwxrwxrwx  1 root root    3 9 月  19 2017 /dev/cdrom -> sr0
brw-rw----  1 root disk 8, 0 9 月  19 2017 /dev/sda
crw--w----  1 root tty  4, 0 9 月  19 2017 /dev/tty0
-rw-r--r--. 1 root root 2053 9 月  19 2017 /etc/passwd
```

2. 查看文件类型的三种方法

① ls -l / ls -ld 或者 ll [ls -l —查看文件 ls -ld —查看路径 ll —– 跟 ls -l 一样]。

```
ll anaconda-ks.cfg                  # 查看第一个字符
-rw-------. 1 root root 2460 6 月   1 23:37 anaconda-ks.cfg
```

```
[root@localhost log]# ls -ld /etc
drwxr-xr-x. 81 root root 4096 Jan 29 03:25 /etc
```

② file 命令

```
[root@localhost data]# file a.txt
a.txt: ASCII text
```

③ stat 命令

```
[root@localhost data]# stat a.txt     # 查看文件的详细属性（其中包括文件时间属性）
File: ‘a.txt’
Size: 3               Blocks: 8          IO Block: 4096   regular file
Device: 803h/2051d      inode: 544365      Links: 1
Access: (0644/-rw-r--r--)  Uid: (    0/    root)   Gid: (    0/    root)
Access: 2018-01-28 20:56:01.965885036 +0800
Modify: 2018-01-28 20:55:27.181876154 +0800
Change: 2018-01-28 20:55:27.181876154 +0800
```

3. Linux 中文件扩展名

Windows 里是通过扩展名来区分文件类型的。Linux 里文件扩展名和文件类型没有关系。但为了容易区分和兼容用户使用 Windows 的习惯，我们还是会用扩展名来表示文件类型。举例如下：

- 源码 .tar、.tar.gz、.tgz、.zip、.tar.bz 表示压缩文件，创建命令一般为 tar、gzip、zip 等。
- .sh 表示 shell 脚本文件，通过 shell 语言开发的程序。
- .pl 表示 perl 语言文件，通过 perl 语言开发的程序。
- .py 表示 python 语言文件，通过 python 语言开发的程序。
- .html、.htm、.php、.jsp、.do 表示网页语言的文件。
- .conf 表示系统服务的配置文件。
- .rpm 表示 rpm 安装包文件。

4. 文件属性

```
例如：[root@localhost /]# ls  -lhi   # 加 i 选项可以显示文件的 inode 号
total 90K
12 dr-xr-xr-x.  2 root root 4.0K Jan 28 18:30 bin
2 dr-xr-xr-x.  5 root root 1.0K Aug  7  2016 boot
4 drwxr-xr-x. 18 root root 3.7K Jan 29 01:29 dev
652802 drwxr-xr-x. 81 root root 4.0K Jan 29 03:25 etc
130563 drwxr-xr-x.  3 root root 4.0K Jan 29 00:57 home
13 dr-xr-xr-x. 12 root root 4.0K Jan 28 18:30 lib
391685 dr-xr-xr-x.  9 root root  12K Jan 28 18:30 lib64
```

```
11 drwx------.  2 root root  16K Aug  7  2016 lost+found
130564 drwxr-xr-x.  2 root root 4.0K Sep 23  2011 media
391689 drwxr-xr-x.  2 root root 4.0K Sep 23  2011 mnt
130565 drwxr-xr-x.  3 root root 4.0K Aug  7  2016 opt
1 dr-xr-xr-x. 97 root root    0 Jan 29  2018 proc
391682 dr-xr-x---.  2 root root 4.0K Jan 28 21:08 root
130566 dr-xr-xr-x.  2 root root  12K Jan 28 18:30 sbin
1 drwxr-xr-x.  7 root root    0 Jan 29  2018 seLinux
15 drwxr-xr-x.  2 root root 4.0K Sep 23  2011 srv
1 drwxr-xr-x. 13 root root    0 Jan 29  2018 sys
522242 drwxrwxrwt.  5 root root 4.0K Jan 29 05:15 tmp
522244 drwxr-xr-x. 14 root root 4.0K Jan 28 20:04 usr
261121 drwxr-xr-x. 20 root root 4.0K Aug  7  2016 var
```

分析：544365 -rw-r--r--. 1 root root 3 Jan 28 20：55 a.txt

inode 索引节点编号：544365。

inode：在 Linux 中创建文件系统时，同时将会创建大量的 inode，通常文件系统磁盘空间中大约百分之一空间分配给了 inode 表，文件引用的是一个 inode 号，因此，当用户搜索或者访问一个文件时，Linux 系统通过 inode 表查找正确的 inode 编号。在找到 inode 编号之后，相关的命令才可以访问该 inode ，并对其进行适当地更改，所以，一个目录是目录下的文件名和文件 inode 号之间的映射，使用 df-i 可以查看分区的 inode 使用率，使用 ls -i filename 可以查看 inode 号。

所以 inode 就是索引节点，它用来存放档案及目录的基本信息，包含时间、档名、使用者及群组等。inode 是 UNIX/Linux 操作系统中的一种数据结构，其本质是结构体，它包含了与文件系统中各个文件相关的一些重要信息，每一个索引节点都是一个表项，包含有关文件的信息（元数据）：

- 文件类型，权限，UID，GID
- 链接数（指向这个文件名路径名称个数）
- 该文件的大小和不同的时间戳
- 指向磁盘上文件的数据块指针
- 有关文件的其他数据

使用 cp 命令时，系统会分配一个空闲的 inode 号，在 inode 表中生成新条目，在目录中创建一个目录项，将名称与 inode 编号关联拷贝数据生成新的文件。

使用 rm 命令时，会使链接数递减，从而释放的 inode 号可以被重用，把数据块放在空闲列表中，删除目录项数据实际上不会马上被删除，但当另一个文件使用数据块时将被覆盖。

使用 mv 命令时，如果 mv 命令的目标和源在相同的文件系统，作为 mv 命令，用新的文件名创建对应新的目录项，删除旧目录条目对应的旧的文件名，不影响 inode 表（除时间戳）或磁盘上的数据位置，没有数据被移动！如果目标和源在一个不同的文件系统，mv 相当于 cp 和 rm。

分析：544365 -rw-r--r--. 1 root root 3 Jan 28 20：55 a.txt

文件类型：文件类型是‘-’，表示这是一个普通文件。

文件权限：rw-r--r-- 表示文件可读、可写、可执行，文件所归属的用户组可读可执行，其他用户可读可执行。

硬链接个数：表示 a.txt 这个文件没有其他的硬链接，因为连接数是 1，就是它本身。

文件属主：表示这个文件所属的用户，这里的意思是 a.txt 文件被 root 用户拥有，是第一个 root。

文件属组：表示这个文件所属的用户组，这里表示 a.txt 文件属于 root 用户组，是第二个 root。

文件大小：文件大小是 3 个字节。

文件修改时间：这里的时间是该文件最后被更新（包括文件创建、内容更新、文件名更新等）的时间。可用如下命令查看文件的修改、访问、创建时间。

```
例如，[root@localhost data]# stat  a.txt     # 深度查询 a.txt
File: ‘a.txt’
Size: 3               Blocks: 8          IO Block: 4096   regular file
Device: 803h/2051d      inode: 544365      Links: 1
Access: (0644/-rw-r--r--)  Uid: (    0/    root)   Gid: (    0/    root)
Access: 2018-01-28 20:56:01.965885036 +0800      ---------- 访问时间
Modify: 2018-01-28 20:55:27.181876154 +0800      ---------- 修改时间
Change: 2018-01-28 20:55:27.181876154 +0800      ---------- 创建时间
```

索引节点 inode，硬盘分区，格式化、创建文件系统，被格式化的磁盘分为两部分：第一部分是 inode；第二部分是 block。block 是用来存储实际数据用的，例如：照片、视频等普通文件数据，inode 是用来存储这些数据的属性的（也就是 ls-l 的结果）。

inode 包含的属性信息有文件大小、属主、归属的用户组、读写权限、文件类型、修改时间，还有指向文件实体指针的功能，但是唯独不包含文件名。

访问一个文件【通过文件名找到 inode-->block 】

例如，查看 inode 大小：

```
[root@localhost ~]# dumpe2fs  /dev/sda1|grep  -i  “inode size”  # 加 i 选项才行
dumpe2fs 1.41.12 (17-May-2010)
inode size:           128
```

2.3 Linux 目录结构概述

CentOS 7 类的 Linux 系统的目录大体上可分为如下：

/ 根目录，一般根目录下只存放目录，不要存放文件，/etc、/bin、/dev、/lib、/sbin 应该和根目录放置在一个分区中。

内核类目录：

/boot：Linux 的内核及引导系统程序所需要的文件目录。

程序与工具类目录：

/bin：存放标准 Linux 程序的工具，在终端里输入 ls，系统就会到该目录查看是否存在该命令程序。如：login；Shells；文件操作实用程序；系统实用程序；压缩工具等。

/sbin：大多是涉及系统管理的命令的存放，是超级权限用户 root 的可执行命令存放地，普通用户无权限执行这个目录下的命令。如：fsck；fdisk；mkfs；shutdown；lilo；init。

/bin、 /sbin、 /usr/bin、/usr/sbin 这几个目录是系统预设的执行文件的放置目录，例如 root 常常使用的 userconf， netconf， perl， gcc， c++ 等数据都放在这几个目录中，所以如果你在提示字符下找不到某个执行档时，可以在这四个目录中查一查！

存放在 /bin 与 /sbin 这两个目录中的程序的主要区别是：/sbin 中的程序只能由 root（管理员）来执行。使用和维护 UNIX 和 Linux 系统的大部分基本程序都包含在 /bin 和 /sbin 里，这两个目录的名气之所以包含 bin，是因为可执行的程序都是二进制文件（Binary Files）。

额外工具类目录：

/usr：存放一些不适合放在 /bin 和 /sbin 目录下的额外工具，如个人安装的程序或工具。

文件系统经常很大，因为所有程序安装在这里。

/usr 里的所有文件一般来自 Linux distribution；本地安装的程序和其他东西在 /usr/local 下。这样可能在升级新版系统或新 distribution 时无须重新安装全部程序。

/usr/bin：可执行二进制文件的目录，几乎所有用户命令，例如常用的命令 ls、tar、mv、cat 等。

/usr/sbin 与 /usr/local/sbin 放置系统管理员使用的可执行命令，如 fdisk、shutdown、mount 等。与 /bin 不同的是，这几个目录是给系统管理员 root 使用的命令，一般用户只能“查看”而不能设置和使用。

/usr/local：这是系统预设的用于升级已安装软件的套件目录。

例如，当你发现有更新的 Web 套件（如 Apache ）可以安装，而你又不想以 rpm 的方式升级你的套件，则你可以将 Apache 这个套件安装在 /usr/local 底下。安装在这里有个好处，因为目前大家的系统都是差不多的，所以如果你的系统要让别人接管的话，也比较容易上手并找到数据。因此，如果你有需要，可以将 /usr/local/bin 这个路径加到“我的 path”中。

/usr/share：用于存放一些共享数据，也存放结构独立的数据。

/usr/share/doc：系统说明文件存放目录。

/usr/share/man：程序说明文件存放目录，使用 man ls 时会查询 usr/share/man/man1/ls.1.gz 的内容，建议单独分区，设置较大的磁盘空间。

/usr/share/man、/usr/share/info、/usr/share/doc：手册页、GNU 信息文档和各种其他文档文件。

/usr/share/man， /usr/local/man：这两个目录为放置各类套件说明档的地方。

例如，你如果执行 man man，则系统会自动去找这两个目录下的所有说明文件。

/usr/include：一些 distribution 套件的头文件放置目录，安装程序时可能会用到。

C 编程语言的头文件。为了一致性，这实际上应该在 /usr/lib 下，但传统上支持这个名字。

/usr/src：存放程序的源代码，是内核源代码目录。

/usr/etc：存放设置文件。

/usr/games 存放游戏和教学文件。

/usr/lib 程序或子系统的不变的数据文件，包括一些 site-wide 配置文件。名字 lib 来源于库（library）。编程的原始库存在 /usr/lib 里。

/opt 主要存放可选程序，直接删除程序不影响系统及设置。安装到 /opt 目录下的程序，它所有的数据、库文件等都是放在同一个目录下面。是给主机额外安装软件所摆放的目录。如：FC4 使用的 Fedora 社群开发软件，如果想要自行安装新的 KDE 桌面软件，可以将该软件安装在该目录下。也有的 Linux 系统，习惯放置在 /usr/local 目录下。

应用函数库类目录：

/lib 该目录用来存放系统动态链接共享库，几乎所有的应用程序都会用到该目录下的共享库，/lib：/usr/lib：/usr/local/lib：系统使用的函数库的目录，程序在执行过程中，需要调用一些额外的参数时需要函数库的协助，比较重要的目录为 /lib/modules。

/usr/lib 与 /usr/local/lib 存放不能直接运行的，却是许多程序运行所必需的一些函数库文件。

日志类目录：

/var 这个目录的内容是经常变动的，用来存储经常被修改的文件，如日志、数据文件、邮箱等，如随时更改的日志文件 /var/log。

/var/log/message 所有的登录文件存放目录。

/var/spool/mail 邮件存放的目录，/var/run：程序或服务启动后，其 PID 存放在该目录下。建议单独分区，设置较大的磁盘空间。

/var/catman 当要求格式化时的 man 页的 cache.man 页的源文件一般存在 /usr/man/man 中；有些 man 页可能有预格式化的版本，存在 /usr/man/cat 中。而其他的 man 页在第一次看时需要格式化，格式化完的版本存在 /var/man 中，这样其他人再看相同的页时就无须等待格式化（/var/catman 经常被清除，就像清除临时目录一样）。

/var/lib 系统正常运行时要改变的文件。

/var/local 在 /usr/local 中安装的程序的可变数据（系统管理员安装的程序）。注意，如果必要，即使本地安装的程序也会使用其他 /var 目录。

/var/lock 锁定文件。许多程序遵循在 /var/lock 中产生一个锁定文件的约定，以支持它们正在使用某个特定的设备或文件。其他程序注意到这个锁定文件，将不试图使用这个设备或文件。

/var/log 各种程序的 Log 文件，特别是 login 和 syslog（/var/log/messages 里存储所有核心和系统程序信息（/var/log/wtmp log 所有到系统的登录和注销）。/var/log 里的文件经常不确定地增长，应该定期清除。

/var/run 保存到下次引导前有效的关于系统的信息文件。例如，/var/run/utmp 包含当前登录的用户的信息。

/var/spool 是 mail， news，打印队列和其他队列工作的目录。每个不同的 spool 在 /var/spool 下有自己的子目录，例如，用户的邮箱在 /var/spool/mail 中。

/var/tmp 比 /tmp 允许大的或需要存在较长时间的临时文件（虽然系统管理员可能不允许 /var/tmp 有很旧的文件）。

配置类目录：

/etc 主要存放整个文件系统配置方面的文件，其中的一些重要文件如下：

passwd； shadow； fstab； hosts； motd； profile； shells； services；lilo.conf

不建议在此目录下存放可执行文件，重要的配置文件有 /etc/inittab、/etc/fstab、/etc/init.d、/etc/X11、/etc/sysconfig、/etc/xinetd.d。修改配置文件之前记得备份。

这个目录相当重要，CentOS 7 开机与系统数据文件均在这个目录之下，因此当这个目录被破坏，那你的系统大概也会坏掉。

/etc/X11：存放与 xwindow 有关的设置。

/etc/rc.d：放置开机和关机的脚本。

/etc/rc.d/init.d：放置启动脚本，开启一些 Linux 系统服务的 scripts （可以想成是批次档或脚本）的目录。

/etc/rc、/etc/rc.d、/etc/rc*.d 是启动或改变运行级时运行的 scripts 或 scripts 的目录。

/etc/rc.d/rc.local 这个文件是开机的执行档。

/etc/xinetd.d 配置 xinetd.conf 可以配置启动其他额外服务。

/etc/passwd 用户数据库，其中的域给出了用户名、真实姓名、家目录、加密的口令和用户的其他信息，它是用户信息文件。

/etc/fstab 启动时 mount –a 命令（在 /etc/rc 或等效的启动文件中）自动 mount 的文件系统列表。Linux 下，也包括用 swapon –a 启用的 swap 区的信息，它是系统开机启动自动挂载的分区列表。

/etc/group 类似 /etc/passwd ，但说明的不是用户而是组，它存储了用户组信息。

/etc/inittab init 的配置文件。

/etc/issue getty 在登录提示符前的输出信息，通常包括系统的一段短说明或欢迎信息，内容由系统管理员确定。

/etc/motd Message Of The Day，成功登录后自动输出，内容由系统管理员确定，经常用于通告信息，如计划关机时间的警告。

/etc/mtab 当前安装的文件系统列表，由 scripts 初始化，并由 mount 命令自动更新，需要一个当前安装的文件系统的列表时使用，例如 df 命令。

/etc/shadow 在安装了影子口令软件的系统上的影子口令文件，影子口令文件将 /etc/passwd 文件中的加密口令移动到 /etc/shadow 中，而后者只对 root 可读，这使破译口令更困难，它是用户密码文件。

/etc/hosts 设定用户自己的 IP 与主机名对应的信息。

/etc/login.defs login 命令的配置文件。

/etc/printcap 类似 /etc/termcap ，但针对打印机，语法不同。

/etc/profile 、/etc/csh.login 、/etc/csh.cshrc 登录或启动时 Bourne 或 C shells 执行的文件。这允许系统管理员为所有用户建立全局缺省环境。

/etc/securetty 确认安全终端，即哪个终端允许 root 登录。一般只列出虚拟控制台，这样就不可能（至少很困难）通过 modem 或网络闯入系统并得到超级用户特权。

/etc/shells 文件中，列出可信任的 shell.chsh 命令，允许用户在本文件指定范围内改变登录 shell，并检查用户 shell 是否列在 /etc/shells 文件中，如果不是，将不允许该用户登录。如：

/bin/sh

/bin/bash

/usr/bin/sh

/usr/bin/bash

/bin/tcsh

/bin/csh

/etc/sysconfig 网络配置相关目录

设备类目录：

/dev 是设备（device）的英文缩写，主要存放与设备有关的文件，访问该目录下某个文件，相当于访问某个设备，例如：挂载光驱 mount /dev/cdrom /mnt。

这个目录对所有的用户都十分重要。因为在这个目录中包含了所有 Linux 系统中使用的外部设备。但是这里并不是存储外部设备的驱动程序，这一点和 Windows、Dos 操作系统不一样。它实际上是一个访问这些外部设备的端口。我们可以非常方便地去访问这些外部设备，和访问一个文件、一个目录没有任何区别。

Linux 沿袭 Unix 的风格，将所有设备当成是一个文件。

设备文件分为两种：块设备文件（b）和字符设备文件（c）。

常见设备文件作如下说明：

/dev/hd[a-t]：IDE 设备

/dev/sd[a-z]：SCSI 设备

/dev/fd[0-7]：标准软驱

/dev/md[0-31]：软 raid 设备

/dev/loop[0-7]：本地回环设备

/dev/ram[0-15]：内存

/dev/null：无限数据接收设备，相当于黑洞

/dev/zero：无限零资源

/dev/tty[0-63]：虚拟终端

/dev/ttyS[0-3]：串口

/dev/lp[0-3]：并口

/dev/console：控制台

/dev/fb[0-31]：framebuffer

/dev/cdrom => /dev/hdc

/dev/modem => /dev/ttyS[0-9]

/dev/pilot => /dev/ttyS[0-9]

/dev/random：随机数设备（依赖于系统中断）

/dev/urandom：随机数设备（不依赖于系统中断）

/mnt 与 /media 这个目录一般是用于存放挂载储存设备的挂载目录的，比如有 cdrom 等目录。可以参看 /etc/fstab 的定义。

用户类目录：

/root：系统管理员 root 的家目录，系统第一个启动的分区为 /，所以最好将 /root 和 / 放置在一个分区下。

/home：系统默认的用户家目录，新增用户账号时，用户的家目录都存放在此目录下，~ 表示当前用户的家目录，~test 表示用户 test 的家目录。建议单独分区，并设置较大的磁盘空间，方便用户存放数据。

信息类目录：

/tmp：临时目录，有些 Linux 会定期清理。

/lost+found：在 ext2 或 ext3 文件系统中，当系统意外崩溃、机器意外关机、突然停电、非正常关机、系统异常产生错误后而产生的一些文件碎片存放在这里，有些文件也临时存放在这里，会将一些遗失的片段放置于此目录下，通常这个目录会自动出现在装置目录下。如加载硬盘于 /disk 中，此目录下就会自动产生目录 /disk/lost+found。

/tmp：一般用户或正在执行的程序临时存放文件的目录，任何人都可以访问，重要数据不可放置在此目录下。

/srv：服务启动之后需要访问的数据目录，如 www 服务需要访问的网页数据存放在 /srv/www 内。

/sys：系统目录，存放硬件信息的相关文件。

/proc：操作系统运行时，进程信息及内核信息（比如 cpu、硬盘分区、内存信息等）存放在这里，此目录的数据都在内存中，如系统核心、外部设备、网络状态，由于数据都存放于内存中，所以不占用磁盘空间，比较重要的目录有 /proc/cpuinfo、/proc/interrupts、/proc/dma、/proc/ioports、/proc/net/* 等。

/proc/cmdline：加载 kernel 时所下达的相关参数。查阅此文件，可了解系统是如何启动的。

/proc/cpuinfo：本机的 CPU 的相关资讯，包含时脉、类型与运算功能等。

/proc/devices：这个文件记录了系统各个主要装置的主要装置代号，与 mknod 有关。

/proc/filesystems：系统已经加载的文件系统。

/proc/interrupts：目前系统的 IRQ 分配状态。

/proc/ioports：目前系统各个装置所配置的 I/O 位址。

/proc/kcore：内存大小。

/proc/loadavg：top 和 uptime 的平均数值。

/proc/meminfo：使用 free 列出的内存资讯。

/proc/modules：Linux 已经加载的模块列表，也可以视为驱动程序。

/proc/mounts：系统已经挂载的数据。

/proc/swaps：已使用的 partition 记录。

/proc/partitions：使用 fdisk -l 出现的 partition 纪录。

/proc/pci：在 PCI 汇流排上面，每个装置的详细情况，可用 lspci 来查阅。

/proc/uptime：用 uptime 时会出现的资讯。

/proc/version：核心版本，用 uname -a 显示的内容。

/proc/bus/*：一些汇流排的装置，还有 U 盘的装置也记录在此。

Linux 目录结构如图 2-2 所示。

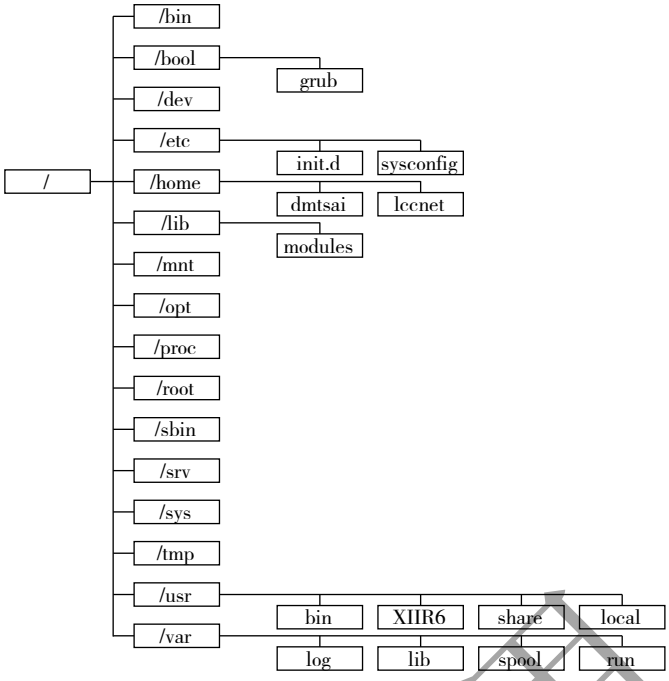

图 2-2　Linux 目录结构简化图

2.4　Linux 的绝对路径与相对路径

系统目录结构对比：在 Windows 系统中，查看文件先进入相应的盘符，然后进入文件目录，所以 Windows 系统它属于多根系统（c：\；d：\；e：\……），如图 2-3 所示。

图 2-3　Windows 目录结构简化图

Linux 只有一个根目录 /，如图 2-4 所示。

图 2-4　Linux 目录结构简化图

根下的目录作用说明：/ 处于 Linux 系统树形结构的最顶端，它是 Linux 文件系统的入口，所有的目录、文件、设备都在 / 之下。

Linux 的其他目录说明：在 Linux 中，一切都被看作文件。终端设备、磁盘等都被看作文件。

路径：在我们平时使用计算机时要找到需要的文件就必须知道文件的位置，而表示文件位置的方式就是路径。

绝对路径：在 Linux 中，绝对路径是从“/”开始的，比如 /usr、/etc/passwd。如果一个路径是从根（/）开始的，它一定是绝对路径，例如：/home/mk。

相对路径：相对路径是以 . 或 .. 开始的，而路径的写法不是由根目录“/”写起的。

【例 1】 相对于当前目录开始，a.txt 可用 ./a.txt 来表示路径。

【例 2】 首先用户进入 / 然后再进入 home ，命令为 cd /home，然后 cd test 此时用户所在的路径为 /home/test。第一个 cd 命令后跟 /home 第二个 cd 命令后跟 test ，并没有斜杠，这个 test 是相对于 /home 目录来讲的，所以叫作相对路径。

【例 3】 当前有一个路径为：

/（根）
└── home
└── mk
　　└── f1
　　　　└── f11
|　　├── f111
|　　└── f112（目的地）
└── f12（当前）

当前处于 f12，如何到达 f112 呢？

绝对路径：cd /home/mk/f1/f11/ f112

相对路径：cd ../f1/f11/f112

总结：

./ 当前目录

cd .. 返回上一级目录

cd ../.. 返回上两级目录

cd 进入个人的主目录 = cd ~

cd - 返回上次所在的目录

2.5 文件与目录基本操作

UNIX 是以目录为主的，Linux 也继承了这一优良特性。Linux 是以树形目录结构的形式来构建整个系统的，可以理解为树形目录是一个用户可操作系统的骨架。虽然本质上无论是目录结构还是操作系统内核都是存储在磁盘上的，但从逻辑上来说 Linux 的磁盘是“挂在”（挂载在）目录上的，每一个目录不

仅能使用本地磁盘分区的文件系统，也可以使用网络上的文件系统。举例来说，可以利用网络文件系统（Network File System，NFS）服务器载入某特定目录等。

2.5.1 显示目录内容与路径及进入目录命令

在 Linux 中，“cd”（改变目录）命令，是对新手和系统管理员来说，最重要最常用的命令。对管理无图形界面的服务器的管理员，“cd”是进入目录，检查日志，执行程序 / 应用软件 / 脚本和其余每个任务的唯一方法。对新手来说，是他们必须自己动手学习的最初始命令。

pwd：Print Working Directory 显示目前所在目录的命令，显示全路径的命令，具备验证当前目录的功能。

【例 4】 从当前目录切换到 /usr/local。

```
[mk@localhost ~]$ pwd   # 显示家目录的路径
/home/mk
[mk@localhost ~]$ cd  /usr/local   # 进入目录
[mk@localhost local]$ pwd  # 显示目录
/usr/local
```

【例 5】 使用绝对路径，从当前目录切换到 /usr/local/lib。

```
[mk@localhost local]$ cd  /usr/local/lib  # 以 / 斜杠开头的都是绝对路径，进入
[mk@localhost lib]$ pwd  # 查看目录
/usr/local/lib
```

【例 6】 使用相对路径，从当前路径切换到 /usr/local/lib。

```
[mk@localhost lib]$ cd  ..  # 退回上一层目录
[mk@localhost local]$ pwd  # 验证
/usr/local
[mk@localhost local]$ cd  lib  # 用相对路径方式一进入
[mk@localhost lib]$ pwd   # 验证
/usr/local/lib
[mk@localhost lib]$ cd ..  # 退回上一层目录再试
[mk@localhost local]$ cd  ./lib  # 用相对路径方式二进入，两种方式要比较
[mk@localhost lib]$ pwd  # 验证
/usr/local/lib
```

【例 7】 从任何目录返回到用户 home 目录，

```
[mk@localhost lib]$ cd  ~   # 回到家目录的方式一，用波浪号
[mk@localhost ~]$ pwd   # 验证
/home/mk
```

```
[mk@localhost ~]$ cd  /usr/local/lib  # 回到原目录，再来一次
[mk@localhost lib]$ cd  # 回到家目录的方式二，就用 cd 命令
[mk@localhost ~]$ pwd   # 验证
/home/mk
```

【例 8】　切换当前目录到上级目录。

```
[mk@localhost ~]$ cd  ..  # 从当前目录到上级目录
[mk@localhost home]$ pwd   # 验证
/home
[mk@localhost ~]$ cd  /usr/local/lib  # 绝对路径方式进入目录
[mk@localhost lib]$ pwd   # 验证
/usr/local/lib
[mk@localhost lib]$ cd  ../..  # 从当前目录到上级的上级目录
[mk@localhost usr]$ pwd  # 验证
/usr
```

【例 9】　回到我们最后一个离开的工作目录（使用“-”和“--”选项）。

```
[mk@localhost ~]$ cd  /usr/local/lib  # 绝对路径方式进入目录
[mk@localhost lib]$ cd  -  # 回到我们最后一个离开的工作目录
/home/mk
[mk@localhost ~]$ cd  -   # 回到我们最后一个离开的工作目录
/usr/local/lib
```

Linux 中一个基本命令是 ls。没有这个命令，我们会在浏览目录条目时遇到困难。

ls 命令，全拼 list，功能是列出目录的内容及其内容属性信息，它的长清单模式如下：

第 1 列：代表文件类型。

下面的 9 个字符是关于文件权限。前 3 个 rwx 字符是文件的拥有者的权限，第二组 3rwx 是文件的所有组的权限，最后的 rwx 是对其他人访问文件的权限。

第 2 列：有多少链接指向这个文件。

第 3 列：谁是这个文件 / 文件夹的所有者。

第 4 列：谁是这个文件 / 文件夹的所有组。

第 5 列：这个文件 / 文件夹的以字节为单位的大小。目录的大小总是 4096 字节。

第 6 列：这告诉我们文件最后的修改时间。

第 7 列：这告诉我们文件名或者目录名。

【例 10】　不带参数运行 ls 会只列出文件或者目录。看不到其他信息输出（译注：有时候你发现无参数的 ls 命令和这里描述的不同，那有可能是你的 ls 命令实际上带参数的 ls 别名）。

```
[mk@localhost usr]$ cd  /usr  # 进入 /usr 目录
[mk@localhost usr]$ pwd  # 查验路基
```

```
/usr
[mk@localhost usr]$ ls  # 列出改目录下有什么文件与子目录
bin  etc  games  include  lib  lib64  libexec  local  sbin  share  src  tmp
```

【例 11】 ls –l = ll 显示文件和目录的详细资料。

```
[mk@localhost usr]$ ls  -l
总用量 280
dr-xr-xr-x.  2 root root 49152 10 月 14 09:11 bin
drwxr-xr-x.  2 root root     6 11 月  5 2016 etc
drwxr-xr-x.  2 root root     6 11 月  5 2016 games
[mk@localhost usr]$ ll  # 等于 ls  -l
总用量 280
dr-xr-xr-x.  2 root root 49152 10 月 14 09:11 bin
drwxr-xr-x.  2 root root     6 11 月  5 2016 etc
drwxr-xr-x.  2 root root     6 11 月  5 2016 games
```

显示文件大小，以字节为单位看大小可能会不方便。6.5M 读起来比 6727680 字节更简单。要这么做，我们可以使用 –h 与 –l 结合的参数。–h 参数意味着便于人识别。

【例 12】 更方便地显示文件大小。

```
[mk@localhost usr]$ ls  -hl  # 与上例对比，多了 -h 参数，文件大小更好理解
总用量 280K
dr-xr-xr-x.  2 root root  48K 10 月 14 09:11 bin
drwxr-xr-x.  2 root root    6 11 月  5 2016 etc
drwxr-xr-x.  2 root root    6 11 月  5 2016 games
```

【例 13】 显示隐藏文件。

```
[mk@localhost usr]$ cd
[mk@localhost ~]$ ls
公共模板视频图片文档下载音乐桌面
[mk@localhost ~]$ ls  -a  # 加 -a 参数，可显示隐藏文件。
.bash_profile  .config        .local 公共图片音乐
```

【例 14】 以尺寸大小排列文件和目录。

```
[mk@localhost usr]$ ls  -lSr  #-S 代表已大小排列，-r 代表升序排列
总用量 280
drwxr-xr-x.  2 root root     6 11 月  5 2016 games
drwxr-xr-x.  2 root root     6 11 月  5 2016 etc
lrwxrwxrwx.  1 root root    10 10 月 14 09:04 tmp -> ../var/tmp
```

```
drwxr-xr-x.   4 root root    34 10 月 14 09:04 src
drwxr-xr-x.  12 root root   131 10 月 14 09:04 local
[mk@localhost usr]$ ls  -lS  # 不加 -r，默认已降序排列。
总用量 280
dr-xr-xr-x. 159 root root 86016 10 月 14 09:11 lib64
dr-xr-xr-x.   2 root root 49152 10 月 14 09:11 bin
dr-xr-xr-x.   2 root root 20480 10 月 14 09:11 sbin
drwxr-xr-x.  47 root root 12288 10 月 14 09:10 libexec
```

【例 15】　将终端划分成栏显示。

```
[mk@localhost usr]$ ls  | pr  -T5  -W$COLUMNS  # 分 5 栏显示
bin         include         libexec         sbin    src
etc         lib       local   share  tmp
games     lib64
[mk@localhost usr]$ ls  | pr  -T2  -W$COLUMNS  # 分 2 栏显示
bin         libexec
etc         local
games     sbin
include    share
lib src
lib64       tmp
[mk@localhost usr]$ ls  -l | pr  -T2  -W$COLUMNS   # 长格式并且分 2 栏显示
总用量 280         drwxr-xr-x.  47 root root 12288 10 月 14
dr-xr-xr-x.   2 root root 49152 10 月 14 drwxr-xr-x.  12 root root   131 10 月 14
drwxr-xr-x.   2 root root     6 11 月  5 dr-xr-xr-x.   2 root root 20480 10 月 14
drwxr-xr-x.   2 root root     6 11 月  5 drwxr-xr-x. 254 root root  8192 10 月 14
```

2.5.2　目录的创建与删除命令

mkdir，创建目录命令：该命令只能针对目录。

语法：mkdir （选项）目录名

【例 16】　在根目录下创建 dir2、dir3，在 /home 目录下创建 dir4。

```
[mk@localhost /]$ su  root   # 在根目录在根目录下创建，需要切换到 root 用户才够权限
密码：
[root@localhost /]# mkdir  dir2  dir3  /home/dir4  # 用一条命令创建 3 个目录，高效
[root@localhost /]# ls  /  /home/  # 用一条命令验证目录是否创立
/:
bin   dev   dir3  home  lib64  mnt  proc  run   srv  tmp  var
```

```
boot  dir2  etc   lib   media  opt  root  sbin  sys  usr
/home/:
dir4  mk
```

【例 17】 创建多级目录，如”/tmp/a/b/c”。

```
[root@localhost /]# mkdir  /tmp/a/b/c
mkdir: 无法创建目录”/tmp/a/b/c”：没有那个文件或目录
[root@localhost /]# mkdir  -p  /tmp/a/b/c
[root@localhost /]# ls  /  /tmp /tmp/a  /tmp/a/b   # 验证目录是否创建
/:
bin   dev   dir3  home  lib64  mnt  proc  run   srv  tmp  var
boot  dir2  etc   lib   media  opt  root  sbin  sys  usr
/tmp:
a
/tmp/a:
b
/tmp/a/b:
C
```

删除目录命令：rmdir，它其实是 rmove directory 缩写，其只有一个选项 -p 类似与 mkdir 命令，这个参数的作用是将上级目录一起删除。例如，新建目录 mkdir -p d1/d2/d3，rmdir -p d1/d2/d3 相当于是删除了 d1，d1/d2， d1/d2/d3。如果一个目录中还有目录，那么当你直接用 rmdir 删除该目录时，会提示该目录不为空，不能删除。

【例 18】 rmdir 命令的运用，验证是否只能删除空目录的命令。

```
[root@localhost /]# mkdir  -p  /dir2  /dir3/dir4  # 先创建目录
[root@localhost /]# touch  /dir2/b.txt    # 再创建文件
[root@localhost /]# ls  /dir2  /dir3    # 验证目录与文件有否创建
/dir2:
b.txt
/dir3:
dir4
[root@localhost /]# rmdir  dir2   # 看能否删除带文件的目录
rmdir: 删除 “dir2” 失败：目录非空
[root@localhost /]# ls   # 验证 dir2 目录是否还在
bin   dev  dir2  home  lib64  mnt  proc  run   srv  tmp  var
boot  dir  etc   lib   media  opt  root  sbin  sys  usr  dir3
[root@localhost /]# rmdir  /dir3/dir4   # 这种写法只能删除 dir4
[root@localhost /]# ls
```

```
bin  dev  dir2  etc  lib  media  opt  root  sbin  sys  usr
boot  dir  dir3  home  lib64  mnt  proc  run  srv  tmp  var
[root@localhost /]# ls /dir3
[root@localhost /]#    验证出 dir4 不在，dir3 还在
[root@localhost /]# mkdir  /dir3/dir4  # 再创建 /dir3/dir4 目录树
[root@localhost /]# rmdir  -p  /dir3/dir4  # 加 -p 参数，尝试删除 /dir3 目录树
rmdir: 删除目录 “/” 失败 : 设备或资源忙
[root@localhost /]# ls  # 验证 /dir3 目录树不见了，出现提示表明根目录也要被删除，不严谨
bin  dev  dir2  home  lib64  mnt  proc  run  srv  tmp  var
boot  dir  etc  lib  media  opt  root  sbin  sys  usr
[root@localhost /]# mkdir  -p  /dir3/dir4  # 再创建 /dir3/dir4 目录树
[root@localhost /]# rmdir  -p  dir3/dir4  # 加 -p 参数，尝试删除 /dir3 目录树，不删 / 目录
[root@localhost /]# ls  # 验证 /dir3 目录
bin  dev  dir2  home  lib64  mnt  proc  run  srv  tmp  var
boot  dir  etc  lib  media  opt  root  sbin  sys  usr
```

【例 19】　删除不为空的目录，还可 rdirm 指令吗?

```
[root@localhost /]# ls  dir2  # 验证 dir2 目录是非空的
b.txt
[root@localhost /]# rmdir  dir2  # 尝试删除不为空的目录
rmdir: 删除 “dir2” 失败 : 目录非空
[root@localhost /]# ls  # 验证 dir2 是否被删除
bin  dev  dir2  home  lib64  mnt  proc  run  srv  tmp  var
boot  dir  etc  lib  media  opt  root  sbin  sys  usr
[root@localhost /]# rmdir  -p dir2  # 加 -p 参数再尝试删除不为空的目录
rmdir: 删除 “dir2” 失败 : 目录非空
[root@localhost /]# ls  # 验证 dir2 是否被删除
bin  dev  dir2  home  lib64  mnt  proc  run  srv  tmp  var
boot  dir  etc  lib  media  opt  root  sbin  sys  usr
```

如果你需要删除不为空的目录，那就需要用 rm 指令。

语法：rm （选项）处理对象。

删除命令 2：rm，作用：可以删除一个目录中的一个或多个文件或目录，对于链接文件，只是删除整个链接文件，而原文件保持不变。

语法：rm （选项）处理对象

选项：-f 强制删除，没有提示，-r 用递归遍历的方式删除全部目录，小心。

【例 20】　删除根目录下内里有文件的 dir2 目录。

```
[root@localhost /]# rm  -rf  dir2  # 加 rf 参数，可以强制删除非空的目录
```

```
[root@localhost /]# ls
bin   dev  etc   lib    media  opt   root  sbin  sys  usr
boot  dir  home  lib64  mnt    proc  run   srv   tmp  var
```

使用 rm -rf （慎用，一定要在删除之前确定一下所在目录，防止误删除重要数据）

2.5.3 文件的复制、移动和删除命令

命令：cp 源文件 / 目录目录文件 / 目录

复制文件：全拼 copy，其功能为复制文件或目录。

选项：-R/r：递归处理，将指定目录下的所有文件与子目录一并处理。

【例 21】 运用 cp 命令复制文件与命令。（不带任何参数下，运行 cp）

```
[root@localhost mk]# cp  /etc/passwd  /opt/  # 复制 /etc/passwd 文件到 /opt 目录
[root@localhost mk]# ll  /opt  # 长格式验证
总用量 4
-rw-r--r--. 1 root root 2156 3 月  13 06:22 passwd
drwxr-xr-x. 2 root root    6 3 月  26 2015 rh
```

【例 22】 拷贝一个目录，需要添加 -r 选项，递归操作。无论该目录是否为空目录，这个选项都是必要的。

```
[root@localhost mk]# cp  -r  /boot/grub  /opt
# 加 -r 参数（连子目录都包含了）复制 /boot/grub 目录到 /opt 目录
[root@localhost mk]# ll  /opt
总用量 0
drwxr-xr-x. 2 root root 27 3 月  14 00:49 grub
drwxr-xr-x. 2 root root  6 3 月  26 2015 rh
[root@localhost mk]# ll  /opt  # 长格式验证
总用量 4
drwxr-xr-x. 2 root root   27 3 月  13 06:27 grub  #d 字头，grub 是目录
-rw-r--r--. 1 root root 2156 3 月  13 06:22 passwd  #- 字头，passwd 是文件
drwxr-xr-x. 2 root root    6 3 月  26 2015 rh
```

【例 23】 显示复制的过程。

```
[root@localhost mk]# cp  -v  /etc/passwd  /etc/shadow  /opt
# 如果你想了解在拷贝文件时都发生了什么，可以用 -v 选项，也试试多文件的复制
“/etc/passwd” -> “/opt/passwd”
“/etc/shadow” -> “/opt/shadow”
[root@localhost mk]# ll /opt
总用量 8
```

```
drwxr-xr-x. 3 root root   18 3 月  14 01:10 grub
-rw-r--r--. 1 root root 2156 3 月  14 10:20 passwd
drwxr-xr-x. 2 root root    6 3 月  26 2015 rh
----------. 1 root root 1222 3 月  14 10:20 shadow
```

【例 24】　使用交互模式复制。

```
[root@localhost mk]# cp  -i  /etc/passwd  /etc/shadow  /opt
# 交互模式下会询问是否覆盖目标目录下的文件。使用 -i 选项，启用交互模式。
cp：是否覆盖“/opt/passwd”？ n
cp：是否覆盖“/opt/shadow”？ n
```

【例 25】　创建备份文件。

```
[root@localhost mk]# cp  -v  --backup  /etc/passwd  /opt
# 使用 --backup 选项，cp 命令会为每一个现有的目标文件做一个备份。
cp：是否覆盖“/opt/passwd”？ y
“/etc/passwd” -> “/opt/passwd”（备份：“/opt/passwd~”）
[root@localhost mk]# ll  /opt  # 查验
总用量 16
drwxr-xr-x. 3 root root   18 3 月  14 01:10 grub
-rw-r--r--. 1 root root 2156 3 月  14 10:32 passwd
-rw-r--r--. 1 root root 2156 3 月  14 10:20 passwd~  # 生成的备份文件
drwxr-xr-x. 2 root root    6 3 月  26 2015 rh
```

mv 命令是 move 的缩写，是一个与 cp 类似的命令，但是它并非创建文件或目录的复制品 / 副本。不管你在使用什么版本的 Linux 系统，mv 都默认安装在你的 Linux 系统上了。可以用来移动文件或者将文件改名（move （rename） files）。

【例 26】　移动文件到另外的目录，需要注意的是文件的源地址和目标地址必须不同。

```
[root@localhost mk]# mkdir  dir1  # 在 mk 用户的家目录下新建 dir1 目录
[root@localhost mk]# mv  /opt/passwd  ./dir1  # 移动 /opt 下的 passwd 文件到 dir1
[root@localhost mk]# ls  ./dir1  # 验证 passwd 文件是否存在
Passwd
[root@localhost mk]# ls  /opt  # 验证 passwd 文件是否已消失
grub  rh
```

【例 27】　在移动文件的时候支持改名操作。

```
[root@localhost mk]# mv  ./dir1/passwd  /opt/pass  # 移动文件的同时，顺便更改文件名
[root@localhost mk]# ls  ./dir1  /opt
./dir1:  # 验证 dir1 目录里的文件是否已消失
```

```
/opt:   #opt 目录新增的文件
grub  pass  rh  #opt 目录新增 pass 文件
```

【例 28】 移动目录。

```
[root@localhost mk]# mv  ./dir1  /opt  # 把当前目录下的 dir1 目录移动到 /opt 下
[root@localhost mk]# ll  /opt  # 验证
总用量 16
drwxr-xr-x. 2 root root    6 3 月  14 15:21 dir1
```

【例 29】 用 mv 命令来更改文件或目录时目标位置和源位置必须相同才可以

```
[root@localhost mk]# mv  /opt/dir1  /opt/dir2  # 用 mv 命令更改目录名
[root@localhost mk]# ls  -l  /opt  # 验证
总用量 16
drwxr-xr-x. 2 root root    6 3 月  14 15:21 dir2   #dir1 变成 dir2
```

【例 30】 获得详细的移动信息要用到 -v 选项。

```
[root@localhost mk]# mv  -v  /opt/dir2  /opt/pass  ./
# 把 dir2 目录与 pass 文件移动到当前目录下，并显示过程
“/opt/dir2” -> “./dir2”
“/opt/pass” -> “./pass”
```

【例 31】 rm 命令的交互式删除文件及显示删除过程。

```
[root@localhost mk]# rm  -i  ./pass  # 加 -i, --interactive 参数进行交互式删除，n 取消
rm：是否删除普通文件 “./pass”？ n
[root@localhost mk]# rm  -v  ./pass   # 加 -v, --verbose 详细显示进行的步骤
rm：是否删除普通文件 “./pass”？ y
已删除“./pass”     # 详细显示删除文件的步骤
```

2.5.4 文件的新建命令

命令：touch，作用：常用来创建空文件，如果文件存在，则修改这个文件的时间。

语法：touch 文件名

1. 命令格式：

touch ［选项］文件

2. 命令参数：

-a 或 - time=atime 或 - time=access 或 - time=use　只更改存取时间。

-c 或 - no-create　不建立任何文档。

-d 使用指定的日期时间，而非现在的时间。

-f 此参数将忽略不予处理，仅负责解决 BSD 版本 touch 指令的兼容性问题。
-m 或 - time=mtime 或 - time=modify　只更改变动时间。
-r 把指定文档或目录的日期时间，统统设成和参考文档或目录的日期时间相同。
-t 使用指定的日期时间，而非现在的时间。

【例 32】　生成一个空文件。

```
[mk@localhost ~]$ su root
密码：
[root@localhost mk]# cd  /opt  # 进入 /opt 目录需要 root 账号
[root@localhost opt]# ll  # 查验目录初始有什么文件及目录
总用量 0
drwxr-xr-x. 2 root root 6 3 月  26 2015 rh
[root@localhost opt]# touch  file0  # 生成一个空文件 file0
```

【例 33】　一次生成多个有规律但不连续的空文件。

```
[root@localhost opt]# touch  file1  file2  # 一次生成多个空文件的方法 1
[root@localhost opt]# touch  file{3,5}  # 一次生成多个空文件的方法 2
```

【例 34】　生成多个连续文件。

```
[root@localhost opt]# touch  file{6..9}.txt  # 创建 file6 到 file9 扩展名为 .txt 的文件。
[root@localhost opt]# ll  # 验证
总用量 0
-rw-r--r--. 1 root root 0 3 月  14 17:51 file0
-rw-r--r--. 1 root root 0 3 月  14 17:52 file1
-rw-r--r--. 1 root root 0 3 月  14 17:52 file2
-rw-r--r--. 1 root root 0 3 月  15 08:33 file3
-rw-r--r--. 1 root root 0 3 月  15 08:33 file5
-rw-r--r--. 1 root root 0 3 月  15 08:31 file6.txt
-rw-r--r--. 1 root root 0 3 月  15 08:31 file7.txt
-rw-r--r--. 1 root root 0 3 月  15 08:31 file8.txt
-rw-r--r--. 1 root root 0 3 月  15 08:31 file9.txt
```

【例 35】　如果文件存在，则修改这个文件的时间。

通过上面的 ll 命令记住 file0 文件旧的时间，如：3 月 14 17：51 file0

```
[root@localhost opt]# touch  -d  “20181019 21:30”  file0 # 通过 -d 选项可改变文件时间
[root@localhost opt]# ll  file0
-rw-r--r--. 1 root root 0 10 月 19 2018 file0
```

【例 36】 令两个文件的时间戳相同，被更新的放前面，被参照的放后面。

```
[root@localhost opt]# touch  -r  file1  file0  # 加 -r 参数
[root@localhost opt]# ll  file0  file1  # 验证
-rw-r--r--. 1 root root 0 3 月  14 17:52 file0
-rw-r--r--. 1 root root 0 3 月  14 17:52 file1
```

-t time 使用指定的时间值 time 作为指定文件相应时间戳记的新值。此处的 time 规定为如下形式的十进制数：[[CC] YY] MMDDhhmm [.SS]

这里，CC 为年数中的前两位，即“世纪数”；YY 为年数的后两位，即某世纪中的年数。如果不给出 CC 的值，则 touch 将把年数 CCYY 限定在 1969 - 2068 之内。MM 为月数，DD 为天数，把年数 CCYY 限定在 1969 - 2068 之内。MM 为月数，DD 为天数，hh 为小时数（几点），mm 为分钟数，SS 为秒数。此处秒的设定范围是 0 - 61，这样可以处理闰秒。这些数字组成的时间是环境变量 TZ 指定的时区中的一个时间。由于系统的限制，早于 1970 年 1 月 1 日的时间是错误的。

【例 37】 设定文件的时间戳。

```
[root@localhost opt]# touch  -t  201211142234.50  file1  # 用 -t 参数改 file1 的时间戳
[root@localhost opt]# ll  file0  file1 # 验证
-rw-r--r--. 1 root root 0 3 月  14 17:52 file0
-rw-r--r--. 1 root root 0 11 月 14 2012 file1
通常情况下我们使用的 ls –l 指示的是文件的 modify time。
ls –lu  file0  file1 或者 ls –l –time=atime 指示的是文件的 access time
ls –lc  file0  file1 或者 ls –l –time=ctime 指示的是文件的 change time
```

【例 38】 重定向，用 echo 与管道命令新建文件并追加字符（运用管道符号 > 与 >>）。

```
[root@localhost mk]# echo  123456  >  file10.txt  # 新建 file10.txt，并输入 123456
[root@localhost mk]# ls   # 验证
file0  file10.txt  file3  file6.txt  file8.txt 公共视频文档音乐
file1  file2       file5  file7.txt  file9.txt 模板图片下载桌面
[root@localhost mk]# echo abcdefg >> file10.txt  # 把 abcdefg 字符追加到下行。
```

2.5.5 文件内容查看命令

cat 命令的用途是连接文件或标准输入并打印。

命令格式：cat [选项] [文件] ...

常用命令参数：

-n，--number　对输出的所有行编号，由 1 开始对所有输出的行数编号

-s，--squeeze-blank　有连续两行以上的空白行，就代换为一行的空白行

-b，--number-nonblank　对非空输出行编号

-E，--show-ends　　在每行结束处显示 $

【例 39】　一次显示整个文件：cat filename　常用来显示文件内容。

```
[root@localhost mk]# cat  file10.txt  # 用 cat 命令查看文本型的文件
123456
abcdefg
```

标准输入文件（stdin）：它对应的是外设键盘输入，而在 Linux 系统中它被抽象成一个文件，准确地说是一个流文件。这种文件和上面普通文本文件最大的区别就是它的文件大小是不固定的，它就像是一个水管的进水端，可以在任何时候都接收输入。

Linux 中，在新的一行的开头，按下 Ctrl-D，就代表 EOF（如果在一行的中间按下 Ctrl-D，则表示输出“标准输入”的缓存区，所以这时必须按两次 Ctrl-D）；Windows 中，Ctrl-Z 表示 EOF。（顺便提一句，Linux 中按下 Ctrl-Z，表示将该进程中断，在后台挂起，用 fg 命令可以重新切回到前台；按下 Ctrl-C 表示终止该进程。）

那么，如果真的想输入 Ctrl-D 怎么办？这时必须先按下 Ctrl-V，然后就可以输入 Ctrl-D，系统就不会认为这是 EOF 信号。Ctrl-V 表示按“字面含义”解读下一个输入，要是想按“字面含义”输入 Ctrl-V，连续输入两次就行了。

【例 40】　将几个文件合并为一个文件：cat file1 file2 > file　-- 将几个文件连接起来显示

```
[root@localhost mk]# cat file11 file12  file13  file14 > file15 # 将几个文件连接成新文件
[root@localhost mk]# cat file15  # 检查新文件 file15
123456
abcdef
7890
ghij
11 12 13
klmn
14
OPQ
```

【例 41】　显示时加上行号，并生成带行号的文件。

```
[root@localhost mk]# cat  -n  file15  #-n 参数可显示行号
1  123456
2  abcdef
3  7890
4  ghij
5  11 12 13
6  klmn
7  14
```

```
8  OPQ
[root@localhost mk]# cat  -n  file15 > file16  # 并生成带行号的文件 file6
```

说明：tac 是将 cat 反过来写，所以它的功能就跟 cat 相反，cat 是由第一行到最后一行连续显示在屏幕上，而 tac 则是由最后一行到第一行反向在屏幕上显示出来。

【例 42】 使用 tac 命令可以从尾到头显示。

```
[root@localhost mk]# tac  file16
8  OPQ
7  14
6  klmn
5  11 12 13
4  ghij
3  7890
2  abcdef
1  123456
tac  file16
[root@localhost mk]# tac  file16  > file17  # 生成一个倒转的文件
[root@localhost mk]# cat file17  # 验证
8  OPQ
7  14
6  klmn
5  11 12 13
4  ghij
3  7890
2  abcdef
1  123456
```

【例 43】 cat 命令的追加，用 >> 定向符号。

```
[root@localhost mk]# cat << EOF >> file17  # 追加字符到 file17，注意区分大小写
> 15 16
> eof  # 小写无效
> EOF  # 大写 ok
[root@localhost mk]# cat file17
8  OPQ
7  14
6  klmn
5  11 12 13
```

```
4  ghij
3  7890
2  abcdef
1  123456
15 16
```

【例 44】　显示一些系统文件。

```
cat  /proc/cpuinfo 显示 CPU info 的信息
cat  /proc/interrupts 显示中断
cat  /proc/meminfo 校验内存使用
cat  /proc/swaps 显示哪些 swap 被使用
cat  /proc/version 显示内核的版本
cat  /proc/net/dev 显示网络适配器及统计
cat  /proc/mounts 显示已加载的文件系统
```

more 命令：more [-dlfpcsu] [-num] [+/ pattern] [+ linenum] [file ...]

命令功能类似 cat：cat 命令是整个文件的内容从上到下显示在屏幕上， more 会以一页一页显示的方便使用者逐页阅读， more 可以按页来查看文件的内容，还支持直接跳转行等功能，按空白键（space）就往下一页显示，按 b 键就会往回（back）一页显示。

命令参数：

+n 从第 n 行开始显示

-n 定义屏幕大小为 n 行

-c 从顶部清屏，然后显示

-d 提示“Press space to continue，'q' to quit（按空格键继续，按 q 键退出）”，禁用响铃功能

-l 略 Ctrl+l（换页）字符

-p 通过清除窗口而不是滚屏来对文件进行换页，与 -c 选项相似

-s 把连续的多个空行显示为一行

-u 把文件内容中的下画线去掉

+/pattern 在每个档案显示前搜寻该字串（pattern），再从该字串前两行之后开始显示。常用操作命令：

Enter　向下 n 行，需要定义。默认为 1 行

Ctrl+F　向下滚动一屏

空格键向下滚动一屏

Ctrl+B　返回上一屏

=　　输出当前行的行号

：f　　输出文件名和当前行的行号

V　　调用 vi 编辑器

！命令调用 Shell，并执行命令

q　　退出 more

【例 45】　指定一屏显示 5 行，并且显示操作帮助

```
[root@localhost mk]# more  -5  -d  /etc/passwd  # 加 -5，一屏 5 行，-d，底行帮助
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:/sbin/nologin
daemon:x:2:2:daemon:/sbin:/sbin/nologin
adm:x:3:4:adm:/var/adm:/sbin/nologin
lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin
--More--(8%)[Press space to continue, ‘q’ to quit.] # 帮助行，按空格键继续往下翻按 q 退出
```

【例 46】　从第 3 行起，显示文件中的内容。

```
[root@localhost mk]# more  +3  file16
3  7890
4  ghij
5  11 12 13
6  klmn
7  14
8  OPQ
```

【例 47】　从文件中查找第一个出现“mn”字符串的行，并从该处前两行开始显示输出：more +/pattern 文件名

```
[root@localhost mk]# more +/mn file16  # 用 +/ 字符选项来查找文件里的字符行
... 跳过
4  ghij
5  11 12 13
6  klmn     # 找出来了
7  14
8  OPQ
```

【例 48】　列出一个目录下的文件（由于内容太多，我们应该学会用 more 来分页显示），这得和管道 | 结合起来。

```
[root@localhost mk]# ls  -l / etc | more -5   # 每页显示 5 行信息
总用量 1416
drwxr-xr-x.  3 root root      101 10 月 14 09:06 abrt
-rw-r--r--.  1 root root       16 10 月 14 09:15 adjtime
-rw-r--r--.  1 root root     1518 6 月   7 2013 aliases
-rw-r--r--.  1 root root    12288 10 月 14 09:29 aliases.db
```

2.5.6　文件查找命令

find 命令是我们在 Linux 系统中用来进行文件搜索用得最多的命令，功能特别强大。

命令名称：find，命令所在路径：/bin/find，。

语法：find【搜索范围】【匹配条件】

```
-name   filename          # 查找名为 filename 的文件
-perm                      # 按执行权限来查找
-user    username          # 按文件属主来查找
-group groupname          # 按组来查找
-mtime   -n +n             # 按文件更改时间来查找文件，-n 指 n 天以内，+n 指 n 天以前
-atime   -n +n             # 按文件访问时间来查找文件，-n 指 n 天以内，+n 指 n 天以前
-ctime   -n +n             # 按文件创建时间来查找文件，-n 指 n 天以内，+n 指 n 天以前
-nogroup                   # 查无有效属组的文件，即文件的属组在 /etc/groups 中不存在
-nouser                    # 查无有效属主的文件，即文件的属主在 /etc/passwd 中不存在
-type    b/d/c/p/l/f       # 查块设备、目录、字符设备、管道、符号链接、普通文件
-size     n[c]             # 查长度为 n 块 [ 或 n 字节 ] 的文件
-mount                     # 查文件时不跨越文件系统 mount 点
-follow                    # 如果遇到符号链接文件，就跟踪链接所指的文件
-prune                     # 忽略某个目录
```

【例 49】　根据文件或目录名称搜索。

find【搜索目录】【-name 或者 -iname】【搜索字符】：-name 和 -iname 的区别一个区分大小写，一个不区分大小写。

```
[mk@localhost ~]$ su root  #find 命令需要 root 用户
密码：
[root@localhost mk]# find  /etc  -name  init  # 精准搜索，必须为 init 才能搜索得到
/etc/seLinux/targeted/active/modules/100/init
/etc/sysconfig/init  # 找到两个
[root@localhost mk]# find  /etc  -iname  init

# 精准搜索，名字必须为 init 或者有大写字母也能搜索得到
/etc/seLinux/targeted/active/modules/100/init
/etc/sysconfig/init
/etc/gdm/Init  # 有大写字母的名字已找到
[root@localhost mk]# find  /etc  -name *init  # 模糊搜索以 init 结尾的文件或目录名
/etc/X11/xinit
/etc/gdbinit
/etc/security/namespace.init  # 以 init 结尾的文件也找到
/etc/seLinux/targeted/active/modules/100/init
```

```
/etc/sysconfig/init
[root@localhost mk]# find  /etc  -name  init???  # 模糊搜索，“?”表示单个字符
/etc/inittab
[root@localhost mk]# find  /etc  -name  init?  # 收不到文件证明，“?”表示单个字符
```

【例 50】 根据文件大小搜索，在根目录下查找大于 200M 的文件，这里 +n 表示大于，-n 表示小于，n 表示等于，1 数据块 = 512B =0.5KB，也就是 1KB 等于 2 数据块。

```
[root@localhost mk]# find  /  -size  +409600  #200MB=204800KB=409600 数据块
/proc/kcore
```

【例 51】 根据所有者和所属组搜索。

```
[root@localhost mk]# cd / # 回到根目录
[root@localhost /]# find  /home  -group  root  # 查询所属组为 root 的文件
/home    # 结果
[root@localhost /]# find  /home  -user  root  # 在查询所有者为 root 的文件
/home     # 结果
```

atime 是指 access time，即文件被读取或者执行的时间，修改文件是不会改变 access time 的。网上很多资料都声称 cat、more 等读取文件的命令会改变 atime，但是我试验时发现使用 cat、more 时 atime 没有被修改。这个问题需要另外做研究探讨。

ctime 即 change time 文件状态改变时间，指文件的 i 结点被修改的时间，如通过 chmod 修改文件属性，ctime 就会被修改。

mtime 即 modify time，指文件内容被修改的时间。

【例 52】 根据时间属性搜索，在 /etc 目录下查找 5 分钟内被修改过属性的文件和目录。

```
[root@localhost /]# find  /etc  -ctime  -5  #-ctime 是最近改动，-5 找 5 分钟内的
find . –atime  n       find . –ctime  n          find . –mtime  n
find . –atime  –n    find . –ctime  –n        find . –mtime  -n
find . –atime  +n      find . –ctime  +n        find . –mtime  +n
+n：大于 n；  -n：小于 n；n：等于 n
```

【例 53】 根据文件类型或 i 节点搜索。

-type 根据文件类型查找：f 表示文件，d 表示目录，l 表示软链接

```
[root@localhost /]# ls  -i  /etc/passwd  # 加 -i 参数可显示节点值
17867351 /etc/passwd
[root@localhost /]# find  -inum  17867351  # 通过节点查找文件
find:  ‘./run/user/1000/gvfs’：权限不够
./etc/passwd    # 已查出
[root@localhost /]# find  /usr/bin  -type l  # 找目录下文件类型是 l 的链接文件
```

【例 54】　按组合条件搜索。这里有两个参数：

-a 表示两个条件同时满足（and）；-o 表示两个条件满足任意一个即可（or）

[root@localhost /]# find /etc -size +163840 -a -size -204800

查找 /etc 目录下大于 80MB 同时小于 100MB 的文件

总结 find 的实例：

find / -name test.txt 在所有目录中查找名字为 test.txt 的文件

find / -name '*.txt' 在所有目录中查找后缀名为 .txt 的文件

find . -name test.txt 在当前目录中查找名字为 test.txt 的文件

find /etc -name '*srm*' 查找 /etc 文件夹下所有名字中包含 srm 的文件

find / -amin -10 查找在系统中最后 10 分钟访问的文件

find / -atime -2 查找在系统中最后 48 小时访问的文件

find / -empty 查找在系统中为空的文件或者文件夹

find / -group cat 查找在系统中属于 group cat 的文件

find / -mmin -5 查找在系统中最后 5 分钟里修改过的文件

find / -mtime -1 查找在系统中最后 24 小时里修改过的文件

find / -nouser 查找在系统中属于作废用户的文件

find / -user fred 查找在系统中属于 FRED 这个用户的文件

find / -name file1 从 '/' 开始进入根文件系统搜索文件和目录

find / -user user1 搜索属于用户 'user1' 的文件和目录

find /home/user1 -name *.bin 在目录 '/ home/user1' 中搜索带有'.bin' 结尾的文件

find /usr/bin -type f -atime +100 搜索在过去 100 天内未被使用过的执行文件

find /usr/bin -type f -mtime -10 搜索在 10 天内被创建或者修改过的文件

find / -name *.rpm -exec chmod 755 '{}' \; 搜索以 '.rpm' 结尾的文件并定义其权限

find / -xdev -name *.rpm 搜索以 '.rpm' 结尾的文件，忽略光驱、捷盘等可移动设备 。

find /etc -not -perm /222 -type f –ls # 查找 /etc/ 目录下所有用户都没有写权限的文件

find /etc -size +1M -type f -exec ls -lh {} \; 查找目录下大于 1M 且类型为普通文件的文件

find / \(-nouser -o -nogroup\) -atime -7 -ls

查找当前系统上没有属主或属组，且最近一周内曾被访问过的文件或目录

find /etc -mtime -7 -a -not -user root -a -not -user hadoop -ls

find /etc -mtime -7 -a -not \(-user root -o -user hadoop\) –ls　# 查找 /etc/ 目录下最近一周内其内容修改过，且属主不是 root 用户也不是 hadoop 用户的文件或目录

find /etc -not -perm -111 -type f –ls 查找 /etc 目录下至少有一类用户没有执行权限的文件

find /etc -perm -113 -type f –ls

查找 /etc 目录下所有用户都有执行权限，且其他用户有写权限的所有文件

Linux 中除了 find 可以查找文件之外，还有 locate 命令也可以，locate 与 find 不同：

find 是实时查找工具，遍历指定路径下文件系统层级结构完成文件查找，速度慢。

locate 是通过 updatedb 命令扫描硬盘中的所有目录和文件来建立一个索引数据库 /var/lib/slocate/slocate.db，然后直接查询这个数据库来找，所以一般来说比 find 速度要快，但是 locate 的查找并不是实时的，而是以数据库的更新为准，一般是系统通过 /etc/cron.daily/mlocate 这个定时任务来自动更新，也可以手动执行 updatedb 来更新。

安装：yum install mlocate，语法 locate [OPTION] PATTERN

-b：只匹配路径中的基名

-c：统计处共有多少个符合条件的文件

-r：正则表达式

-i 不区分大小写

【例 55】 locate 的基础用法。

```
[root@localhost /]# cd /tmp    # 进入 /tmp 目录
[root@localhost tmp]# locate inittab   # 查找 inittab 文件
/etc/inittab
/usr/share/augeas/lenses/dist/inittab.aug
/usr/share/man/zh_CN/man5/inittab.5.gz
/usr/share/Vim/Vim74/syntax/inittab.Vim    # 结果
[root@localhost tmp]# mkdir -p /tmp/locatetest  # 新建 /tmp/locatetest 目录
[root@localhost tmp]# locate /tmp/locatetest  # 马上查找刚建的目录，没结果
[root@localhost tmp]# find /tmp/locatetest  # 用 find 命令查找
/tmp/locatetest   # 得出结果
[root@localhost tmp]# updated   # 手动更新文件索引资料库，记住这个命令
[root@localhost tmp]# locate /tmp/locatetest  # 再查询，还是没有结果
```

结论：对于 /tmp 目录下的新建文件，用 updated 命令是更新不到文件资料库的，因为 /tmp 目录不属于文件资料库的收录范围。locate 查不到 /tmp 目录下的新建文件，但能查到旧的。

```
[root@localhost tmp]# mkdir -p /home/locatehometest  # 在 /home 目录新建目录再试
[root@localhost tmp]# locate locatehometest  # 马上查询刚建的目录，又查不到
[root@localhost tmp]# updated   # 手动更新文件索引资料库
[root@localhost tmp]# locate locatehometest  # 查询刚建的目录
/home/locatehometest   # 查到结果
```

搜索命令所在的目录及别名信息：which，命令所在路径：/usr/bin/which。

语法：which【命令】执行权限：所有用户

【例 56】 查询 ls 命令所在目录以及别名信息。

```
[root@localhost tmp]# which ls
alias ls='ls --color=auto'
/usr/bin/ls
```

搜索命令所在的目录及帮助文档路径：whereis，命令所在路径：/usr/bin/whereis

语法：whereis【命令】执行权限：所有用户

【例 57】　查询 ls 命令所在的目录。

```
[root@localhost tmp]# whereis ls
ls: /usr/bin/ls /usr/share/man/man1/ls.1.gz /usr/share/man/man1p/ls.1p.gz
```

type 命令主要用于区分一个命令到底是 shell 自带的还是外部独立的二进制文件提供的。如果是 shell 自带的则会提示此命令为 shell buildin，否则会列出命令的位置。

【例 58】　cd 为 shell 自带的命令，当用 which 查找时，which 会按照 PATH 变量设置的路径进行搜索，结果显示 no cd in...；用 type cd 则显示 cd 为 shell buildin 命令。ssh 不是 shell 自带命令，用 type 时会显示 ssh 的路径。

```
[root@localhost tmp]# which  cd
/usr/bin/cd
[root@localhost tmp]# type  cd
cd 是 shell 内嵌
[root@localhost tmp]# type  ssh
ssh 是 /usr/bin/ssh
```


2.6　文件 / 目录访问权限管理

在 Linux 世界中，可以说万物皆文件。Linux 文件一般分为两种：一般文件和目录文件。文件权限对于数据安全至关重要，有必要清楚地知道一般文件权限和目录文件权限的意义。权限的作用，通过对文件设定权限可以达到以下三种访问限制权限：

只允许用户自己访问：

允许一个预先指定的用户组中的用户访问：

允许系统中的任何用户访问。

2.6.1　查看文件 / 目录的访问权限及用户与组

对于文件的权限：

r：读的权限，读取此文件的实际内容，如读取文本文件的文字内容。

w：写的权限，编辑，新增或者是修改该文件的内容（但不含删除该文件），当你对一个文件具有 w 权限，并不具备删除该文件本身的权限。

x：执行权限，该文件可以被系统执行的权限，我们的文件是否能被执行是由“x”这个权限决定的，而跟文件名是没有绝对的关系。

总结：对于文件的 r、w、x 来说，主要都是针对“文件的内容”而言，与文件的存在与否没有

关系。

对于目录的权限：

r：表示具有读取目录结构列表的权限，（看到目录里面有什么），命令：ls。

w：具有更改该目录结构列表的权限。包括：新建新的文件和目录；删除已经存在的文件与目录。将已存在的目录或文件进行重命名；转移该目录内的文件到其他目录位置，命令：touch mkdir rm mv cp。

x：代表用户能否进入该目录成为工作目录的途径，命令：cd cat。

总结：文件是存放实际数据的所在，目录只要的内容是记录文件名列表，文件名与目录有强烈的关联。

【例 59】 假设我有一个账户名称为 mk，他的主文件夹在 /home/mk/，此目录下新建名为 the_root.date 的文件，查看 the_root.date 的权限，并尝试这种权限是否可以删除 the_root.date。

```
[mk@localhost ~]$ touch  the_root.date    # 新建 the_root.date 文件
[mk@localhost ~]$ ls  -l  # 用长格式可查看文件的权限
总用量 0
-rw-rw-r--. 1 mk mk 0 2 月  28 21:33 the_root.date
[mk@localhost ~]$ rm  the_root.date     # 删除文件
[mk@localhost ~]$ ls  -l        # 删除文件后验证
总用量 0
```

【例 60】 新建 test 目录，并查看这个目录的权限，并尝试把它删除。

```
[mk@localhost ~]$ mkdir  test
[mk@localhost ~]$ ls  -l  test  # 用长格式可查看目录的权限，这方式能查？
总用量 0
[mk@localhost ~]$ ls  -l   # 用长格式可查看目录的权限
总用量 0
drwxrwxr-x. 2 mk mk 6 2 月  28 21:43 test
```

u-g-o：所有者 -- 用户组 -- 其他用户。

所有者：就是创建文件的用户，这个用户拥有对它所创建的文件的一切权限，所有者可以允许其所在的用户组可以访问所有者的文件。

用户组：用户组是具有相同特征用户的逻辑集合，有时我们需要让多个用户具有相同的权限，比如查看、修改某一个文件的权限，一种方法是分别对多个用户进行文件访问授权，如果有 10 个用户的话，就需要授权 10 次，显然这种方法不太合理；另一种方法是建立一个组，让这个组具有查看、修改此文件的权限，然后将所有需要访问此文件的用户放入这个组中，那么所有用户就具有了和组一样的权限。这就是用户组。

其他用户：系统内的其他所有者用户就是 other 用户类。

常见几种文件权限组成：

- rwx --- ---：文件所有者对文件具有读取、写入和执行的权限。

- rwx r-- r--：文件所有者具有读、写与执行的权限，用户组里用户及其他用户则具有读取的权限。

- rw- rw- r-x：文件所有者与同组用户对文件具有读写的权限，而其他用户仅具有读取和执行的权限。

drwx--x—x：目录所有者具有读写与进入目录的权限，其他用户近能进入该目录，却无法读取任何数据。

drwx------：除了目录所有者具有完整的权限之外，其他用户对该目录完全没有任何权限。

【例 61】　每个用户都拥有自己的专属目录，通常放置 /home 下，查询 mk 用户的。

```
[mk@localhost ~]$ su  root   # 切换都超级用户 root
密码：
[root@localhost mk]# cd    # 超级用户 root 回它的家目录
[root@localhost ~]# ll  /home/   # 用 ll 命令查 home 目录是否有 mk 目录，并查其权限
总用量 4
drwx------. 16 mk mk 4096 2 月  28 21:43 mk
```

注：[rwx------] 表示目录所有者本身拥有的权限，其他用户是无法进入的，root 可以。

【例 62】　你以什么用户身份登录，那么你创建的文件或目录，自动成为该文件的所属主和组，再切换回 mk 账号，分别新建文件 a.txt 和 b 目录，再观察其权限。

```
[root@localhost ~]# su  mk    # 切换回 mk 账号
[mk@localhost root]$ cd    # 回 mk 的家目录
[mk@localhost ~]$ pwd    # 验证回家目录没有
/home/mk
[mk@localhost ~]$ touch  a.txt   # 新建 a.txt
[mk@localhost ~]$ ll  a.txt    # 长格式显示 a.txt 的权限
-rw-rw-r--. 1 mk mk 0 2 月  28 22:04 a.txt
[mk@localhost ~]$ mkdir  b   # 新建目录 b
[mk@localhost ~]$ ll    # 长格式显示 b 目录的权限
总用量 0
-rw-rw-r--. 1 mk mk 0 2 月  28 22:04 a.txt
drwxrwxr-x. 2 mk mk 6 2 月  28 22:04 b
drwxrwxr-x. 2 mk mk 6 2 月  28 21:43 test
```

2.6.2　改变文件 / 目录的文件拥有者与所属组

改变文件的所属关系用到命令：

chown：可以用来改变文件（或目录）的属主
chgrp：可以用来改变文件（或目录）的默认属组
如果你要对目录进行操作，加参数 -R
chown 语法：
chown user：group filename 比如：chown hr：san a.txt 把文件的属主和属组改为 hr，san
chown user filename 比如：chown san a.txt 把文件的属主改为 san 用户
chown ：group f ilename 比如：chown ：miao a.txt 把文件的属组改为 miao 这个组
chown user：filename 比如：chown san：a .txt 自动继承这个用户所有的组
chgrp hr filename 比如：chgrp hr f.txt
选项 -R：递归（注意大写）（目录下的所有内容都更改，否则只修改目录）。
【例 63】 把 a.txt 改组。

```
[mk@localhost ~]$ su root  # 登录 root 账号
```

密码：

```
[root@localhost mk]# pwd   # 查看当前目录地址
/home/mk
[root@localhost mk]# ll  *.txt   # 查看目录里的 txt 文件
-rw-rw-r--. 1 root   root   0 2 月  28 22:04 a.txt
[root@localhost mk]# chown  mk  a.txt   # 把 a.txt 改为 mk 用户的
[root@localhost mk]# ll    # 验证
```

总用量 0

```
-rw-rw-r--. 1 mk   root   0 2 月  28 22:04 a.txt
drwxrwxr-x. 2 mk   mk   6 2 月  28 21:43 test
[root@localhost mk]# chown  root:root  a.txt  # 把 a.txt 改成 root 账号的，root 账号组的
[root@localhost mk]# ll    # 验证
```

总用量 0

```
-rw-rw-r--. 1 root root 0 2 月  28 22:04 a.txt
drwxrwxr-x. 2 mk   mk   6 2 月  28 21:43 test
[root@localhost mk]# chown  :mk a .txt    # 把 a.txt 改成 mk 账号组的
[root@localhost mk]# ll  # 验证
```

总用量 0

```
-rw-rw-r--. 1 root mk   0 2 月  28 22:04 a.txt
drwxrwxr-x. 2 mk   mk   6 2 月  28 22:04 b
-rw-r--r--. 1 root root 0 2 月  29 01:46 b.txt
```

```
-rw-r--r--. 1 root root 0 2 月  29 01:46 c.txt
-rw-r--r--. 1 root root 0 2 月  29 01:46 d.txt
drwxrwxr-x. 2 mk   mk   6 2 月  28 21:43 test
```

【例 64】　对 test.txt 文件更改用户与所属组。

```
[root@localhost mk]# touch  test.txt  # 新建 test.txt
[root@localhost mk]# ll  # 验证新建是否成功
```

总用量 0

```
-rw-rw-r--. 1 root mk   0 2 月  28 22:04 a.txt
drwxrwxr-x. 2 mk   mk   6 2 月  28 22:04 b
drwxrwxr-x. 2 mk   mk   6 2 月  28 21:43 test
-rw-r--r--. 1 root root 0 2 月  29 02:15 test.txt
[root@localhost mk]# chown  mk  test.txt  # 把 test.txt 改成 mk 账号用户的
[root@localhost mk]# ll   # 验证
```

总用量 0

```
-rw-rw-r--. 1 root mk   0 2 月  28 22:04 a.txt
drwxrwxr-x. 2 mk   mk   6 2 月  28 22:04 b
drwxrwxr-x. 2 mk   mk   6 2 月  28 21:43 test
-rw-r--r--. 1 mk   root 0 2 月  29 02:15 test.txt
[root@localhost mk]# chown  :mk  test.txt  # 把 test.txt 改成 mk 账号用户所属组的
[root@localhost mk]# ll  # 验证
```

总用量 0

```
-rw-rw-r--. 1 root mk 0 2 月  28 22:04 a.txt
drwxrwxr-x. 2 mk   mk 6 2 月  28 22:04 b
drwxrwxr-x. 2 mk   mk 6 2 月  28 21:43 test
-rw-r--r--. 1 mk   mk 0 2 月  29 02:15 test.txt
```

[root@localhost mk]# chown root：yu test.txt # 把 test.txt 改成 root 账号，yu 用户所属组的，应该不成功的，因为 yu 账号用户所属组不存在，请看下面的提示：

chown：无效的组：“root：yu”

[root@localhost mk]# chown yu：root test.txt # 把 test.txt 改成 yu 账号，root 用户所属组的，应该不成功的，因为 yu 账号用户不存在，请看下面的提示：

chown：无效的用户：“yu：root”

[root@localhost mk]# ll # 验证

总用量 0

```
-rw-rw-r--. 1 root mk 0 2 月  28 22:04 a.txt
drwxrwxr-x. 2 mk   mk 6 2 月  28 22:04 b
drwxrwxr-x. 2 mk   mk 6 2 月  28 21:43 test
-rw-r--r--. 1 mk   mk 0 2 月  29 02:15 test.txt
[root@localhost mk]# chown  root:root  test.txt  # 把 test.txt 改成 root 账号，root 用户所属组的，
[root@localhost mk]# ll  # 验证
```

总用量 0

```
-rw-rw-r--. 1 root mk   0 2 月  28 22:04 a.txt
drwxrwxr-x. 2 mk   mk   6 2 月  28 22:04 b
drwxrwxr-x. 2 mk   mk   6 2 月  28 21:43 test
-rw-r--r--. 1 root root 0 2 月  29 02:15 test.txt
```

【例 65】 一个文件只有读的权限，拥有者是否可以写这个文件？

```
[root@localhost mk]# su  mk  # 切换到 mk 账号
[mk@localhost ~]$ touch  b.txt  # 新建 b.txt
[mk@localhost ~]$ ll  b.txt    # 查看 b.txt 权限
-rw-rw-r--. 1 mk mk 0 3 月   1 04:53 b.txt
[mk@localhost ~]$ chmod  000  b.txt  # 取消对 b.txt 的所有权限
[mk@localhost ~]$ Vim b.txt  # 写入 12345, 并用 :wq!  保存
[mk@localhost ~]$ cat b.txt   #mk 用户下查看 b.txt
cat: b.txt: 权限不够        # 没有权限的缘故
[mk@localhost ~]$ su  root   # 切换到超级用户
```

密码：

```
[root@localhost mk]# cat  b.txt  #root 用户下可查看 b.txt
12345
```

实验结果：文件所有者一定可以强制写文件。就像 root 可以对 shadow 强制写。因 shadow 的拥有者是 root。

2.6.3 更改文件 / 目录的所有权限

修改权限用的命令：chmod，作用：修改文件，目录的权限，语法：chmod [对谁操作] [操作符] [赋予什么权限] 文件名

对谁操作：

u----> 用户 user，表示文件或目录的所有者

g----> 用户组 group，表示文件或目录所属的用户组

o----> 其他用户 others

a----> 所有用户 all

操作符：

+ # 添加权限；　- # 减少权限；　= # 直接给定一个权限

权限：r w x

用文字法表达，如表 2-6 所示。

表 2-6　文字表达法

u-w	user	拥有者
g+x	group	组
o=r	other	其他人
a+x	all	所有人

【例 66】　chmod 修改权限的几种用法。

```
[root@localhost mk]# touch  1.txt  # 新建 1.txt
[root@localhost mk]# ll  1.txt  # 查看 1.txt
-rw-r--r--. 1 root root 0 3 月  1 05:20 1.txt
[root@localhost mk]# chmod  u-w  1.txt  # 去除 1.txt 文件的拥有者改写的权限
[root@localhost mk]# ll  1.txt  # 查看 1.txt
-r--r--r--. 1 root root 0 3 月  1 05:20 1.txt
[root@localhost mk]# chmod  g+x  1.txt  # 增加 1.txt 文件的属组用户的执行权限
[root@localhost mk]# ll 1.txt   # 查看 1.txt
-r--r-xr--. 1 root root 0 3 月  1 05:20 1.txt
[root@localhost mk]# chmod  a+x  1.txt  # 增加 1.txt 文件的所有用户执行权限 [root@local-
host mk]# ll  1.txt  # 查看 1.txt
-r-xr-xr-x. 1 root root 0 3 月  1 05:20 1.txt
[root@localhost mk]# chmod  a=rwx  1.txt   # 让 1.txt 文件的所有用户都具备读、写和执行权
限
[root@localhost mk]# ll  1.txt  # 查看 1.txt
-rwxrwxrwx. 1 root root 0 3 月  1 05:20 1.txt
```

使用八进制（0-7）数字表示权限法（也称数学法）如表 2-7 所示，权限的八进制表示法如图 2-5 所示。

表 2-7　使用八进制（0-7）数字表示权限法（也称数字法）

权限	二进制值	八进制值	描述
---	000	0	没有任何权限
--x	001	1	只有执行权限

续表

权限	二进制值	八进制值	描述
-w-	010	2	只有写入权限
-wx	011	3	有写入和执行权限
r--	100	4	只有读取权限
r-x	101	5	有读取和执行权限
rw-	110	6	有读取和写入权限
rwx	111	7	有全部权限

图 2-5　权限的八进制表示法

例如：rw- 的值是多少？答：4+2=6

rwx r-x r-x 的值是多少？答：rwx=4+2+1=7；r-x=4+1=5 rwx r-x r-x=755

语法：chmod 755 文件或文件夹名字：

chmod a=rwx b.txt 等于 chmod 777 b.txt

【例 67】　新建 c.txt，并用用数字法调整权限。

```
[root@localhost mk]# touch  c.txt   # 新建 c.txt
[root@localhost mk]# mkdir test2    # 新建 test2 目录
[root@localhost mk]# ll   # 查看
总用量 4
-rwxrwxrwx. 1 root root   0 3 月   1 05:20 1.txt
-rw-rw-r--. 1 root mk     0 2 月  28 22:04 a.txt
drwxrwxr-x. 2 mk   mk     6 2 月  28 22:04 b
----------. 1 mk   mk   616 3 月   1 05:03 b.txt
-rw-r--r--. 1 root root   0 3 月   2 06:12 c.txt
drwxrwxr-x. 2 mk   mk     6 2 月  28 21:43 test
drwxr-xr-x. 2 root root   6 3 月   2 06:18 test2
-rw-r--r--. 1 root root   0 2 月  29 02:15 test.txt
```

结论：通过观察 c.txt 与 test2 目录，刚刚新建的空文件默认权限是 644，空目录是 755。

```
[root@localhost mk]# chmod  755  c.txt  # 用数字法对文件调整权限
[root@localhost mk]# chmod  777  ./test2  # 用数字法对目录调整权限，777 是满权限
[root@localhost mk]# ll   # 查看验证有否调整好
```

```
总用量 4
-rwxrwxrwx. 1 root root   0 3 月   1 05:20 1.txt
-rw-rw-r--. 1 root mk     0 2 月  28 22:04 a.txt
drwxrwxr-x. 2 mk   mk     6 2 月  28 22:04 b
----------. 1 mk   mk   616 3 月   1 05:03 b.txt
-rwxr-xr-x. 1 root root   0 3 月   2 06:12 c.txt
drwxrwxr-x. 2 mk   mk     6 2 月  28 21:43 test
drwxrwxrwx. 2 root root   6 3 月   2 06:18 test2
```

权限对文件和目录的影响：

有三种权限可以应用：读取、写入与执行，这些权限对访问文件和目录的影响如表 2-8 所示。

表 2-8　权限对访问文件和目录的影响

权限	对文件的影响	对目录的影响
r（读取）	可以读取文件的内容	可以列出目录的内容（文件名）
w（写入）	可以更改文件的内容	可以创建或删除目录中的任意文件
x（执行）	可以作为命令执行文件	可以访问目录的内容（取决于目录中文件的权限）

2.6.4　更改文件 / 目录的默认权限

为什么我们创建的文件的权限是 644 呢？我们创建文件的默认权限是怎么来的？

umask 命令允许你设定文件创建时的缺省模式，对应每一类用户（文件属主、同组用户、其他用户）存在一个相应的 umask 值中的数字。

文件默认权限 = 666，目录默认权限 = 777

我们一般在 /etc/profile、$ [HOME]/.bash_profile 或 $[HOME]/.profile 中设置 umask 值。

永久生效，编辑用户的配置文件 Vim .bash_profile。

【例 68】　观察 /etc/profile 的默认配置。

```
[root@xuegod63 ~]# Vim  /etc/profile   # 编辑 /etc/profile 配置文件，找到如下 5 行
if [ $UID -gt 199 ] && [ "`/usr/bin/id -gn`" = "`/usr/bin/id -un`" ]; then
umask 002
else
umask 022
fi
：q 退出 Vim
```

注：UID 大于 199 且用户的组名和用户名一样，那么 umask 值为 002，否则为 022。

注：-gt 在 shell 中表示大于； id -g 显示用户组 ID ，id -gn 显示组名。

```
[root@localhost mk]# umask   # 查询默认权限值
0022
```

```
[root@localhost mk]# touch  d.txt  # 在 umask 为 022 的默认值下，新建空文件 d.txt
[root@localhost mk]# mkdir  test3 # 在 umask 为 022 的默认值下，新建空目录 test3
[root@localhost mk]# ll
总用量 4
-rwxrwxrwx. 1 root root   0 3 月   1 05:20 1.txt
-rw-rw-r--. 1 root mk     0 2 月  28 22:04 a.txt
drwxrwxr-x. 2 mk   mk     6 2 月  28 22:04 b
----------. 1 mk   mk   616 3 月   1 05:03 b.txt
-rwxr-xr-x. 1 root root   0 3 月   2 06:12 c.txt
-rw-r--r--. 1 root root   0 3 月   2 16:35 d.txt
drwxrwxr-x. 2 mk   mk     6 2 月  28 21:43 test
drwxrwxrwx. 2 root root   6 3 月   2 06:18 test2
drwxr-xr-x. 2 root root   6 3 月   2 16:36 test3
-rw-r--r--. 1 root root   0 2 月  29 02:15 test.txt
```

通过 ll 命令，观察得出结论：022 默认下，文件的权限为 644，目录权限为 755。
权限的算法：一般情况是，目录默认权限 umask 值
文件：666-022=644 ；目录：777-022=755。

【例 69】 自定义 umask 让自定义的权限临时生效。

```
[root@localhost mk]# umask  # 查询默认权限值
0022
[root@localhost mk]# umask  044  # 自定义 umask 值为 044
[root@localhost mk]# umask     # 查询 umask 值
0044
[root@localhost mk]# touch  e.txt   # 在新的 umask 值下生成空文件，观察它的权限
[root@localhost mk]# mkdir  test4   # 在新的 umask 值下生成空目录，观察它的权限
[root@localhost mk]# ll
总用量 4
-rwxrwxrwx. 1 root root   0 3 月   1 05:20 1.txt
-rw-rw-r--. 1 root mk     0 2 月  28 22:04 a.txt
drwxrwxr-x. 2 mk   mk     6 2 月  28 22:04 b
----------. 1 mk   mk   616 3 月   1 05:03 b.txt
-rwxr-xr-x. 1 root root   0 3 月   2 06:12 c.txt
-rw-r--r--. 1 root root   0 3 月   2 16:35 d.txt
-rw--w--w-. 1 root root   0 3 月   3 01:07 e.txt
drwxrwxr-x. 2 mk   mk     6 2 月  28 21:43 test
drwxrwxrwx. 2 root root   6 3 月   2 06:18 test2
drwxr-xr-x. 2 root root   6 3 月   2 16:36 test3
```

```
drwx-wx-wx. 2 root root   6 3 月   3 01:40 test4
-rw-r--r--. 1 root root   0 2 月  29 02:15 test.txt
```

通过 ll 命令，观察得出结论：044 默认下，文件 e.txt 的权限为 666-044=622，目录 test4 权限为 777-044=733。

【例 70】 互动：umask 掩码为 033 创建普通文件后，权限是什么？

```
互动：umask 掩码为 033 创建普通文件后，权限是什么？ 666-033=633 ( rw- -wx -wx) ?
[root@localhost mk]# umask 033  # 自定义 umask 值为 033
[root@localhost mk]# umask    # 检验 umask 值
0033
[root@localhost mk]# mkdir  test5   # 在新的 umask 值下生成空目录，观察它的权限
[root@localhost mk]# touch  f.txt    # 在新的 umask 值下生成空文件，观察它的权限
[root@localhost mk]# ll
总用量 4
-rwxrwxrwx. 1 root root   0 3 月   1 05:20 1.txt
-rw-rw-r--. 1 root mk     0 2 月  28 22:04 a.txt
drwxrwxr-x. 2 mk   mk     6 2 月  28 22:04 b
----------. 1 mk   mk   616 3 月   1 05:03 b.txt
-rwxr-xr-x. 1 root root   0 3 月   2 06:12 c.txt
-rw-r--r--. 1 root root   0 3 月   2 16:35 d.txt
-rw--w--w-. 1 root root   0 3 月   3 01:07 e.txt
-rw-r--r--. 1 root root   0 3 月   3 01:57 f.txt
drwxrwxr-x. 2 mk   mk     6 2 月  28 21:43 test
drwxrwxrwx. 2 root root   6 3 月   2 06:18 test2
drwxr-xr-x. 2 root root   6 3 月   2 16:36 test3
drwx-wx-wx. 2 root root   6 3 月   3 01:40 test4
drwxr--r--. 2 root root   6 3 月   3 01:57 test5
-rw-r--r--. 1 root root   0 2 月  29 02:15 test.txt
```

通过 ll 命令，观察得出结论：033 默认下，文件 f.txt 的权限为 666-033=633，实际为 644。显然不对，默认权限科学的计算方法：

1. 将默认权限（目录 777，文件 666）和 umask 值都转换为 2 进制。
2. 对 umask 取反。
3. 将默认权限和 umask 取反后的值做与运算。
4. 将得到的二进制值再转换八进制，即为权限。

例如：umask 为 022

```
6  6  6         umask   0  2   2
110 110 110           000  010  010    # 转成二进制
```

```
111  101  101   # umask 取反的值
110 110 110   与    # 第二步，默认权限和 umask 取反后的值做与运算
111 101 101       # umask 取反的值
110 100 100
6   4   4    # 转成八进制
```

例如：umask 为 033 结果为：644

```
6  6  6        umask   0  3   3
110 110 110          000 011  011   # 转成二进制
111 100  100   # umask 取反的值
110 110 110   与    # 默认权限和 umask 取反后的值做与运算
111 100 100       # umask 取反的值
110 100 100
6   4   4   # 转成八进制
```

例如：umask 为 033 目录默认权限为什么?

```
7   7   7       umask   0  3   3
111 111 111          000 011  011   # 转成二进制
111 100  100   # umask 取反的值
111 111 111   与    # 默认权限和 umask 取反后的值做与运算
111 100 100       # umask 取反的值
111 100 100
7   4   4
```


2.7 文件的归档、压缩与解压

Linux 文件归档的目的是为了保证文件或目录的安全，在本地存储介质或网络上以归档的方式备份数据，建立归档文件就是每个系统使用者或运维人员的必要素质，可用压缩与不压缩两种方案进行，在损坏或丢失数据时可用于恢复。

文件压缩之后，占用的空间变小，方便传输，也可以节省磁盘空间。

常见的压缩文件：

Windows：.rar .zip .7z

Linux：.zip .gz .bz2 .xz .tar.gz .tar.bz2 .tar.xz

Linux 中 gzip、bzip2、xz 都有 1-9 压缩等级划分，数字越大，压缩率越高。

2.7.1　用 gzip 对文件进行压缩与解压缩

gzip 的 0.1 版本是在 1992 年发布的，gzip 是 Linux 中常见的压缩 / 解压工具，最常见的使用对象是 *.gz 格式的文件，同时 gzip 压缩的文件在 Windows 中也可以被解压，gzip 指令可以解压 compress 压缩的文件。这里简单介绍下它最常见的用法：

gzip：压缩命令

格式：gzip 源文件（不保留源文件）

格式：gzip -r 目录（只能压缩目录下的文件，不能压缩目录）

gunzip：解压缩命令

压缩文件格式：gunzip （不保留压缩文件）

格式：gunzip -r 压缩目录

选项：压缩文件 -a 或 --ascii：使用 ASCII 文字模式。

-c 或 --stdout 或 --to-stdout：把解压后的文件输出到标准输出设备。

-f 或 -force：强行解开压缩文件，不理会文件名称或硬连接是否存在，以及该文件是否为符号连接。

-h 或 --help：在线帮助。

-l 或 --list：列出压缩文件的相关信息。

-L 或 --license：显示版本与版权信息。

-n 或 --no-name：解压缩时，若压缩文件内含有原来的文件名称及时间戳记，则将其忽略不予处理。

-N 或 --name：解压缩时，若压缩文件内含有原来的文件名称及时间戳记，则将其回存到解开的文件上。

-q 或 --quiet：不显示警告信息。

-r 或 --recursive：递归处理，将指定目录下的所有文件及子目录一并处理。

-S 或 --suffix：更改压缩字尾字符串。

-t 或 --test：测试压缩文件是否正确无误。

-v 或 --verbose：显示指令执行过程。

-V 或 --version：显示版本信息。

#：#为数字。压缩等级，-1 最快，-9 最慢，默认 -6。

gzip 不能压缩目录，可以指定压缩级别 1-9，默认级别是 6，压缩格式是 gz。

【例 71】 压缩文件，原文件名为 file1.txt，压缩后原文件消失，压缩后文件名为 file1.txt.gz

```
[mk@localhost ~]$ touch  file1.txt  # 先生成 file1.txt 文件
[mk@localhost ~]$ ls  -l  file*    # 验证
-rw-rw-r--. 1 mk mk 0 3 月   4 06:10 file1.txt
[mk@localhost ~]$ gzip  file1.txt  # 压缩 file1.txt 文件
[mk@localhost ~]$ ll  # 验证，压缩后 file1.txt 原文件消失，多出一个 file1.txt.gz 的压缩包
总用量 4
-rw-rw-r--. 1 mk mk 30 3 月   4 06:10 file1.txt.gz
```

【例 72】 解压文件，观察压缩包是否还在。

```
[mk@localhost ~]$ gzip  -d  file1.txt.gz   # 解压文件，用 -d 选项方式
[mk@localhost ~]$ ls  –lh
# 加 -h 参数验证，可以显示隐藏文件，验证压缩包是否还在。若不在，则解压 file1.txt
总用量 0
-rw-rw-r--. 1 mk mk 0 3 月   4 06:10 file1.txt
[mk@localhost ~]$ gzip file1.txt  # 再压缩 file1.txt 文件
[mk@localhost ~]$ ll  # 验证
总用量 4
-rw-rw-r--. 1 mk mk 30 3 月   4 06:10 file1.txt.gz
[mk@localhost ~]$ gunzip  file1.txt.gz   # 解压文件，用 gunzip 命令方式
[mk@localhost ~]$ ll  # 验证
总用量 0
-rw-rw-r--. 1 mk mk 0 3 月   4 06:10 file1.txt
总结：gzip  -d 等于 gunzip ，解压后还原出文件后压缩包会消失；ls  –l 等于 ll；
```

【例 73】 压缩的时候，显示压缩率。

```
[mk@localhost ~]$ gzip  -v  file1.txt   # 压缩 file1.txt 文件
file1.txt:    0.0% -- replaced with file1.txt.gz
# 由于 file1.txt 是空文件，压缩率为 0，效果不明显
[mk@localhost ~]$ echo  1234567890  > file1.txt   # 加十个字符并重新再创建 file1.txt
[mk@localhost ~]$ ll   # 验证
总用量 8
-rw-rw-r--. 1 mk mk 11 3 月   4 06:44 file1.txt      # 新创建的 file1.txt
-rw-rw-r--. 1 mk mk 30 3 月   4 06:10 file1.txt.gz    # 原来的 file1.txt 的压缩包
[mk@localhost ~]$ gzip  -d  file1.txt.gz   # 尝试解压相同名的文件，观察会发生什么
gzip: file1.txt already exists; do you wish to overwrite (y or n)? n   # 选 n，不覆盖，终止
    not overwritten
[mk@localhost ~]$ ll  # 验证
总用量 8
-rw-rw-r--. 1 mk mk 11 3 月   4 06:44 file1.txt
-rw-rw-r--. 1 mk mk 30 3 月   4 06:10 file1.txt.gz
[mk@localhost ~]$ rm  file1.txt.gz   # 删除原空文件的 file1.txt 的压缩包
[mk@localhost ~]$ ll  # 验证
总用量 4
-rw-rw-r--. 1 mk mk 11 3 月   4 06:44 file1.txt
[mk@localhost ~]$ gzip  -v  file1.txt   # 压缩的时候，加 -v 可显示压缩率
```

```
file1.txt:  -18.2% -- replaced with file1.txt.gz   # 压缩率为 18.2
[mk@localhost ~]$ ll  # 验证
总用量 4
-rw-rw-r--. 1 mk mk 41 3 月   4 06:44 file1.txt.gz    # 压缩后 file1.txt 原文件消失
```

【例 74】　一条命令压缩多个文件，压缩之后，是各自分开压缩包。

```
[mk@localhost ~]$ touch  file{2..5}  # 新创建 file2-5 连续文件 4 个
[mk@localhost ~]$ ll  # 验证
总用量 4
-rw-rw-r--. 1 mk mk 41 3 月   4 06:44 file1.txt.gz
-rw-rw-r--. 1 mk mk  0 3 月   4 07:02 file2
-rw-rw-r--. 1 mk mk  0 3 月   4 07:02 file3
-rw-rw-r--. 1 mk mk  0 3 月   4 07:02 file4
-rw-rw-r--. 1 mk mk  0 3 月   4 07:02 file5
[mk@localhost ~]$ gzip  file{2..5}  # 压缩 file2-5 连续文件 4 个
[mk@localhost ~]$ ll  # 验证
总用量 20
-rw-rw-r--. 1 mk mk 41 3 月   4 06:44 file1.txt.gz
-rw-rw-r--. 1 mk mk 26 3 月   4 07:02 file2.gz
-rw-rw-r--. 1 mk mk 26 3 月   4 07:02 file3.gz
-rw-rw-r--. 1 mk mk 26 3 月   4 07:02 file4.gz
-rw-rw-r--. 1 mk mk 26 3 月   4 07:02 file5.gz   # 生成各自的压缩包也是四个
[mk@localhost ~]$ rm  file{2..5}.gz    # 删除刚生成的压缩包
[mk@localhost ~]$ ll  # 验证
总用量 4
-rw-rw-r--. 1 mk mk 41 3 月   4 06:44 file1.txt.gz
```

【例 75】　压缩过程中，保留原文件（用压缩到标准输出来实现）。

```
[mk@localhost ~]$ touch  file{2,4}   # 生成两个不连续的文件
[mk@localhost ~]$ ll  # 查看生成情况
总用量 4
-rw-rw-r--. 1 mk mk 41 3 月   4 06:44 file1.txt.gz
-rw-rw-r--. 1 mk mk  0 3 月   4 07:23 file2
-rw-rw-r--. 1 mk mk  0 3 月   4 07:23 file4
[mk@localhost ~]$ gzip  -c  file{2,4} > file24.gz   # 用 -c，将两个文件压缩到一个包里
[mk@localhost ~]$ ll
总用量 8
-rw-rw-r--. 1 mk mk 41 3 月   4 06:44 file1.txt.gz
```

```
-rw-rw-r--. 1 mk mk  0 3 月   4 07:23 file2
-rw-rw-r--. 1 mk mk 52 3 月   4 14:15 file24.gz
-rw-rw-r--. 1 mk mk  0 3 月   4 07:23 file4    # file2、file4 文件还在
```

【例 76】 压缩到标准输出中，并保留原文件的另一方法。

```
[mk@localhost ~]$ touch  file3.txt  # 创建新文件
[mk@localhost ~]$ echo  abcdefg  >  file3.txt  # 把 abcdefg 字符加进新创建的文件中
[mk@localhost ~]$ ll  # 验证
总用量 12
-rw-rw-r--. 1 mk mk 41 3 月   4 06:44 file1.txt.gz
-rw-rw-r--. 1 mk mk  0 3 月   4 07:23 file2
-rw-rw-r--. 1 mk mk 52 3 月   4 14:15 file24.gz
-rw-rw-r--. 1 mk mk  8 3 月   4 23:31 file3.txt
-rw-rw-r--. 1 mk mk  0 3 月   4 07:23 file4
[mk@localhost ~]$ cat  file3.txt  | gzip >file3   # 标准输出到 file3 压缩包中
[mk@localhost ~]$ ll  # 验证
总用量 16
-rw-rw-r--. 1 mk mk 41 3 月   4 06:44 file1.txt.gz
-rw-rw-r--. 1 mk mk  0 3 月   4 07:23 file2
-rw-rw-r--. 1 mk mk 52 3 月   4 14:15 file24.gz
-rw-rw-r--. 1 mk mk 28 3 月   4 23:32 file3    # 标准输出到 file3 压缩包，不带 .gz
-rw-rw-r--. 1 mk mk  8 3 月   4 23:31 file3.txt
-rw-rw-r--. 1 mk mk  0 3 月   4 07:23 file4
```

【例 77】 zcat：查看压缩文件。

```
[mk@localhost ~]$ cat   file3  # 用 cat 命令查看，没显示，证明 file3 不是文本文件
[mk@localhost ~]$ zcat  file3  # 查看压缩文件命令
abcdefg
[mk@localhost ~]$ mv file3 file3.gz
[mk@localhost ~]$ ll
总用量 16
-rw-rw-r--. 1 mk mk 41 3 月   4 06:44 file1.txt.gz
-rw-rw-r--. 1 mk mk  0 3 月   4 07:23 file2
-rw-rw-r--. 1 mk mk 52 3 月   4 14:15 file24.gz
-rw-rw-r--. 1 mk mk 28 3 月   4 23:32 file3.gz
-rw-rw-r--. 1 mk mk  8 3 月   4 23:31 file3.txt
-rw-rw-r--. 1 mk mk  0 3 月   4 07:23 file4
```

【例 78】　解压缩保留源文件的方法：

```
[mk@localhost ~]$ ll  # 查看 file2 与 file4 是否存在
总用量 16
-rw-rw-r--. 1 mk mk 41 3 月   4 06:44 file1.txt.gz
-rw-rw-r--. 1 mk mk  0 3 月   4 07:23 file2
-rw-rw-r--. 1 mk mk 52 3 月   4 14:15 file24.gz
-rw-rw-r--. 1 mk mk 28 3 月   4 23:32 file3.gz
-rw-rw-r--. 1 mk mk  8 3 月   4 23:31 file3.txt
-rw-rw-r--. 1 mk mk  0 3 月   4 07:23 file4
[mk@localhost ~]$ rm  file{2,4}  # 若在，则把这两个文件删除
[mk@localhost ~]$ ll  # 删后验证
总用量 16
-rw-rw-r--. 1 mk mk 41 3 月   4 06:44 file1.txt.gz
-rw-rw-r--. 1 mk mk 52 3 月   4 14:15 file24.gz
-rw-rw-r--. 1 mk mk 28 3 月   4 23:32 file3.gz
-rw-rw-r--. 1 mk mk  8 3 月   4 23:31 file3.txt
[mk@localhost ~]$ gunzip  -c  file24.gz  > file2   # 用 -c 选项和用管道输出文件
[mk@localhost ~]$ ll
总用量 16
-rw-rw-r--. 1 mk mk 41 3 月   4 06:44 file1.txt.gz
-rw-rw-r--. 1 mk mk  0 3 月   5 01:25 file2      # 解压出来的文件
-rw-rw-r--. 1 mk mk 52 3 月   5 01:24 file24.gz   # 原来的压缩包还在
-rw-rw-r--. 1 mk mk  8 3 月   4 23:31 file3.txt
-rw-rw-r--. 1 mk mk  8 3 月   4 23:32 file3.txt.gz
```


2.7.2　用 bzip2 和 bunzip 对文件进行压缩与解压缩

bzip2 的 0.1 版本是在 1996 年发布的，可见 bzip2 的开发是要晚于 gzip 的。由于 bzip2 与 gzip 相比，bzip2 压缩后的文件大小比 gzip 压缩后的文件小，算法不一样，且 bzip2 耗费 cpu 的资源比较多，所以 bzip2 一经推出，便受到了广大用户的欢迎，bzip2 压缩后的格式：.bz2。

同样地，bzip2 也不能压缩目录。也有一个选项 -c，可以保留原文件，在指定的目录下生成新的压缩文件。解压的时候，也可以使用 -c 来保留压缩文件，在指定的目录下生成新的解压文件，并且解压文件的名字是可以更改的。

选项：

-c：将压缩与解压缩的结果送到标准输出。

-d：执行解压缩。

-f 或 -force：bzip2 在压缩或解压缩时，若输出文件与现有文件同名，预设不会覆盖现有文件。若

要覆盖。请使用此参数。

-h：在线帮助。

-k：bzip2 在压缩或解压缩后，会删除原始文件。若要保留原始文件，请使用此参数。

-l，--license。

-s：降低程序执行时内存的使用量。

-t：测试 .bz2 压缩文件的完整性。

-v：压缩或解压缩文件时，显示详细的信息。

-z：强制执行压缩。

-V：显示版本信息（注意是大写，Linux 总是区分大小写）。

--repetitive-best：若文件中有重复出现的资料时，可利用此参数提高压缩效果。

--repetitive-fast：若文件中有重复出现的资料时，可利用此参数加快执行效果。

【例 79】 bunzip 2 其实是 bzip2 的软链接，因此压缩解压都可以通过 bzip2 -d 实现。

```
[mk@localhost ~]$ ls -Fhl /usr/bin/bunzip 2
lrwxrwxrwx. 1 root root 5 10 月 14 09:04 /usr/bin/bunzip 2 -> bzip2*
[mk@localhost ~]$ ls -i /bin/bunzip 2 /bin/bzip2  # 查询 两个命令的 inode 节点
50367246 /bin/bunzip 2  50367251 /bin/bzip2
```

总结：bunzip 2 是 bzip2 的一个符号连接。

【例 80】 压缩文件，原文件名为 b1.txt，压缩后原文件消失，观察压缩后的文件名。

```
[mk@localhost ~]$ rm file*.*  # 删除一些过渡文件
[mk@localhost ~]$ ll  # 验证删除
总用量 0
[mk@localhost ~]$ touch b1.txt  # 生成 b1.txt 文件
[mk@localhost ~]$ echo abcdefghijk > b1.txt  # 填 10 个字母给 b1.txt，让它不是空文件
[mk@localhost ~]$ ll  # 验证
总用量 4
-rw-rw-r--. 1 mk mk 12 3 月  5 02:03 b1.txt
[mk@localhost ~]$ bzip2 b1.txt  # 压缩后原文件消失，生成压缩包
[mk@localhost ~]$ ll # 验证
总用量 4
-rw-rw-r--. 1 mk mk 49 3 月  5 02:03 b1.txt.bz2  # 压缩后原文件消失，生成压缩包
```

【例 81】 解压文件，观察压缩包是否还存在。

```
[mk@localhost ~]$ bunzip 2 b1.txt.bz2  # 解压缩
[mk@localhost ~]$ ll # 压缩包不在，解压出 b1.txt
总用量 4
-rw-rw-r--. 1 mk mk 12 3 月  5 02:03 b1.txt
```

总结：bzip2 -d 等于 bunzip 2，解压后还原出文件后压缩包会消失。

【例 82】 压缩与解压缩的时候，显示状态。

```
[mk@localhost ~]$ bzip2 -v b1.txt # 压缩时加 -v，显示压缩率
b1.txt:  0.245:1, 32.667 bits/byte, -308.33% saved, 12 in, 49 out.
[mk@localhost ~]$ bzip2 -d -v b1.txt.bz2 #-d 解压缩，再加 -v，显示状态
b1.txt.bz2: done
[mk@localhost ~]$ ll
总用量 4
-rw-rw-r--. 1 mk mk 12 3 月  5 02:03 b1.txt
```

【例 83】 按最高压缩率压缩，并模拟解压实际并不解压（达到检查压缩包好坏目的）。

```
[mk@localhost ~]$ bzip2 -v -9 b1.txt # 按最高压缩率压缩（-9）再次压缩 b1.txt
b1.txt:  0.245:1, 32.667 bits/byte, -308.33% saved, 12 in, 49 out.
[mk@localhost ~]$ ll # 验证
总用量 4
-rw-rw-r--. 1 mk mk 49 3 月  5 02:03 b1.txt.bz2  # 已生成压缩包
[mk@localhost ~]$ bzip2 -tv b1.txt.bz2 # 加 -tv 选项，模拟解压
b1.txt.bz2: ok    #ok 是压缩包完好的标志
```

总结：-9 其实是 bzip2 命令采用的默认级别。

【例 84】 压缩过程中，加 -k 参数，保留原文件。

```
[mk@localhost ~]$ touch b3 bc10 # 创建 b3、bc10 两个新文件
[mk@localhost ~]$ ll # 验证有没有生成
总用量 4
-rw-rw-r--. 1 mk mk 49 3 月  5 02:03 b1.txt.bz2
-rw-rw-r--. 1 mk mk  0 3 月  5 21:37 b3
-rw-rw-r--. 1 mk mk  0 3 月  5 21:37 bc10
[mk@localhost ~]$ bzip2 -k b3 bc10 # 压缩过程中，加 k 参数，保留原文件
[mk@localhost ~]$ ll  # 验证生成什么压缩包
总用量 12
-rw-rw-r--. 1 mk mk 49 3 月  5 02:03 b1.txt.bz2
-rw-rw-r--. 1 mk mk  0 3 月  5 21:37 b3
-rw-rw-r--. 1 mk mk 14 3 月  5 21:37 b3.bz2    # 按每个文件各自生成压缩包
-rw-rw-r--. 1 mk mk  0 3 月  5 21:37 bc10
-rw-rw-r--. 1 mk mk 14 3 月  5 21:37 bc10.bz2  # 按每个文件各自生成压缩包
```

总结：bzip2 命令可用 -k 选项进行保留原文件压缩，而 gzip 不能用 -k 选项。而压缩到标准输出从

而保留原文件，bzip2 与 gzip 都可用通过 -c 选项进行。

【例 85】 bzcat：查看压缩文件。

```
[mk@localhost ~]$ touch  b4   # 创建 b4 空文件
[mk@localhost ~]$ echo  123456  > b4  # 用 echo 管道命令写入 123456 字符进 b4
[mk@localhost ~]$ cat  b4  # 显示 b4 文件内容
123456
[mk@localhost ~]$ bzip2  b4  # 压缩 b4 文件
[mk@localhost ~]$ ll  # 验证压缩包是否已创建，b4 原文件是否还存在
总用量 16
-rw-rw-r--. 1 mk mk 49 3 月   5 02:03 b1.txt.bz2
-rw-rw-r--. 1 mk mk  0 3 月   5 21:37 b3
-rw-rw-r--. 1 mk mk 14 3 月   5 21:37 b3.bz2
-rw-rw-r--. 1 mk mk 44 3 月   5 22:10 b4.bz2
-rw-rw-r--. 1 mk mk  0 3 月   5 21:37 bc10
-rw-rw-r--. 1 mk mk 14 3 月   5 21:37 bc10.bz2
[mk@localhost ~]$ bzcat  b4    尝试不加 # 不加 .bz2
bzcat: Can't open input file b4: No such file or directory.   # 结果不行
[mk@localhost ~]$ bzcat  b4.bz2   # 查看压缩文件里的文本类型文件
123456
```

【例 86】 多文件追加到一个压缩包：如，创建 b5 文本文件并追加到 b4.bz2 中。

```
[mk@localhost ~]$ touch  b5   # 创建 b5 文本文件
[mk@localhost ~]$ echo  hello  > b5  # 为 b5 填入 hello 字符
[mk@localhost ~]$ cat  b5  # 查验
hello
[mk@localhost ~]$ bzip2  -c  b5  >> b4.bz2
# 加 -c 选项，用标准输出方式追加到 b4.bz2 中
[mk@localhost ~]$ ll  # 查验 b4.bz2
总用量 20
-rw-rw-r--. 1 mk mk 49 3 月   5 02:03 b1.txt.bz2
-rw-rw-r--. 1 mk mk  0 3 月   5 21:37 b3
-rw-rw-r--. 1 mk mk 14 3 月   5 21:37 b3.bz2
-rw-rw-r--. 1 mk mk 86 3 月   6 00:53 b4.bz2
-rw-rw-r--. 1 mk mk  6 3 月   6 00:52 b5
-rw-rw-r--. 1 mk mk  0 3 月   5 21:37 bc10
-rw-rw-r--. 1 mk mk 14 3 月   5 21:37 bc10.bz2
[mk@localhost ~]$ bzcat  b4. bz2  #bzcat 命令查看文本类型文件的压缩包
```

```
123456
Hello
```

2.7.3　用 tar 命令进行文件归档

tar 是 Unix 和 Linux 系统上的压缩打包工具，可以将多个文件合并为一个文件，打包后的文件后缀亦为“tar”。tar 文件格式是 POSIX 标准，最初是 POSIX.1-1988，当前是 POSIX.1-2001。

tar 是“tape archive”（磁带存档）的简称，它出现在还没有软盘驱动器、硬盘和光盘驱动器的计算机早期阶段，随着时间的推移，tar 命令逐渐变为一个将很多文件进行存档的工具，目前许多用于 Linux 操作系统的程序就是打包为 tar 档案文件的形式。在 Linux 里面，tar 一般和其他没有文件管理的压缩算法文件结合使用，用 tar 打包整个文件目录结构成一个文件，再用 gz，bzip 等压缩算法压缩成一次。也是 Linux 常见的压缩归档的处理方法。

tar 这里首先需要明确两个概念：打包和压缩。打包是指将一大堆文件或目录合并成一个总的文件；压缩则是将一个大容量的文件通过一些压缩算法（gzip、bzip2 等命令）变成一个小容量的文件。Linux/Unix 下的压缩命令通常都只能对一个文件进行压缩操作，所以通常需要使用 tar 命令对文件进行打包，然后进行压缩操作。

（1）tar 命令常用的功能

-c：创建新的 tar 文件

-x：解开 tar 文件

-t：列出 tar 文件中包含的文件的信息

-r：附加新的文件到 tar 文件中

注意：以上功能是独立的，一次只能使用一个。注意：.tgz 是 .tar.gz 的简称。

（2）tar 命令常用的参数

-z：使用 gzip 进行解压缩

-j：使用 bzip2 进行解压缩

-Z：使用 compress 进行解压缩（大写）

-v：显示解压缩执行过程

-f：指定要处理的文件名（注意：如果需要使用 -f 参数，需要将 f 参数放在所有参数最后面，在 f 之后要立即接文件名，不能有其他参数。）

例如：tar　-zxvf　/tmp/etc.tar.gz　# 是正确的，

tar　-zxfv　/tmp/etc.tar.gz　# 则是错误的。

【例 87】　打包当前目录下 dir1 下的文件，不需要压缩。

```
[mk@localhost ~]$ time  tar  -cvf  ./dir1.tar  ./dir1
# 打包 /etc/x11 目录下所有文件，不需要压缩，加 time 命令可以显示归档的速度。
./dir1/
./dir1/dir2/
./dir1/dir2/dir3/
```

```
./dir1/dir2/dir3/dir3.txt
./dir1/dir2/dir2.txt
./dir1/dir1.txt
[mk@localhost ~]$ ll  # 验证是否生成 .tar 的文件。
总用量 16
drwxrwxr-x. 3 mk mk    34 3 月   8 09:50 dir1
-rw-rw-r--. 1 mk mk 10240 3 月  10 01:51 dir1.tar   # 生成了
-rw-rw-r--. 1 mk mk   958 3 月   8 10:19 dir1.zip
real        0m0.004s       #time 命令起的作用
user        0m0.001s
sys         0m0.003
ssys        0m0.004s
```

【例 88】 打包当前目录下 dir1 下的文件，以 gzip 方式压缩。

```
[mk@localhost ~]$ tar  -zcvf  dir1.tar.gz  ./dir1   # 加 -z 参数，以 gzip 方式压缩
./dir1/
./dir1/dir2/
./dir1/dir2/dir3/
./dir1/dir2/dir3/dir3.txt
./dir1/dir2/dir2.txt
./dir1/dir1.txt
[mk@localhost ~]$ ll  # 验证有否 .tar.gz 格式的文件生成
总用量 20
drwxrwxr-x. 3 mk mk    34 3 月   8 09:50 dir1
-rw-rw-r--. 1 mk mk 10240 3 月  10 01:51 dir1.tar
-rw-rw-r--. 1 mk mk   195 3 月  11 00:38 dir1.tar.gz   # 已生成
-rw-rw-r--. 1 mk mk   958 3 月   8 10:19 dir1.zip
```

【例 89】 打包当前目录下 dir1 下的文件，以 bzip2 方式压缩。

```
[mk@localhost ~]$ tar  -jcvf  dir1.tar.bz2  ./dir1  # 加 -j 参数，以 bzip2 方式压缩
./dir1/
./dir1/dir2/
./dir1/dir2/dir3/
./dir1/dir2/dir3/dir3.txt
./dir1/dir2/dir2.txt
./dir1/dir1.txt
[mk@localhost ~]$ ll  # 验证是否有 .tar.bz2 格式的文件生成
总用量 24
```

```
drwxrwxr-x. 3 mk mk    34 3 月   8 09:50 dir1
-rw-rw-r--. 1 mk mk 10240 3 月  10 01:51 dir1.tar
-rw-rw-r--. 1 mk mk   207 3 月  11 00:42 dir1.tar.bz2  # 已生成
-rw-rw-r--. 1 mk mk   195 3 月  11 00:38 dir1.tar.gz
-rw-rw-r--. 1 mk mk   958 3 月   8 10:19 dir1.zip
```

【例 90】　打包当前目录下 dir1 下的文件，以 compress 方式压缩。

```
[mk@localhost ~]$ tar  -Zcvf  dir1.tar.Z  ./dir1
# 加 -Z（大写）参数，以 compress 方式压缩
./dir1/
./dir1/dir2/
./dir1/dir2/dir3/
./dir1/dir2/dir3/dir3.txt
./dir1/dir2/dir2.txt
./dir1/dir1.txt
[mk@localhost ~]$ ll    # 验证是否有 .tar.Z 格式的文件生成
总用量 28
drwxrwxr-x. 3 mk mk    34 3 月   8 09:50 dir1
-rw-rw-r--. 1 mk mk 10240 3 月  10 01:51 dir1.tar
-rw-rw-r--. 1 mk mk   207 3 月  11 00:42 dir1.tar.bz2
-rw-rw-r--. 1 mk mk   195 3 月  11 00:38 dir1.tar.gz
-rw-rw-r--. 1 mk mk   435 3 月  11 00:48 dir1.tar.Z  # 已生成
-rw-rw-r--. 1 mk mk   958 3 月   8 10:19 dir1.zip
```

【例 91】　查看打包压缩文件的文件内容。

```
[mk@localhost ~]$ tar  -tvf  dir1.tar  # 查看打包 .tar 文件（没压缩）的文件。
drwxrwxr-x mk/mk          0 2020-03-08 09:50 ./dir1/
drwxrwxr-x mk/mk          0 2020-03-08 09:50 ./dir1/dir2/
drwxrwxr-x mk/mk          0 2020-03-08 09:50 ./dir1/dir2/dir3/
-rw-rw-r-- mk/mk          0 2020-03-08 09:50 ./dir1/dir2/dir3/dir3.txt
-rw-rw-r-- mk/mk          0 2020-03-08 09:50 ./dir1/dir2/dir2.txt
-rw-rw-r-- mk/mk          0 2020-03-08 09:50 ./dir1/dir1.txt
[mk@localhost ~]$ rm  dir1.tar   # 删除 dir1.tar
[mk@localhost ~]$ ll  # 验证 dir1.tar 是否被删除
总用量 16
drwxrwxr-x. 3 mk mk  34 3 月   8 09:50 dir1
-rw-rw-r--. 1 mk mk 207 3 月  11 00:42 dir1.tar.bz2
-rw-rw-r--. 1 mk mk 195 3 月  11 00:38 dir1.tar.gz
```

```
-rw-rw-r--. 1 mk mk 435 3 月  11 01:05 dir1.tar.Z
-rw-rw-r--. 1 mk mk 958 3 月   8 10:19 dir1.zip
[mk@localhost ~]$ tar  -tvf  dir1.tar.gz  # 查看打包兼压缩 .tar.gz 的文件
drwxrwxr-x mk/mk          0 2020-03-08 09:50 ./dir1/
drwxrwxr-x mk/mk          0 2020-03-08 09:50 ./dir1/dir2/
drwxrwxr-x mk/mk          0 2020-03-08 09:50 ./dir1/dir2/dir3/
-rw-rw-r-- mk/mk          0 2020-03-08 09:50 ./dir1/dir2/dir3/dir3.txt
-rw-rw-r-- mk/mk          0 2020-03-08 09:50 ./dir1/dir2/dir2.txt
-rw-rw-r-- mk/mk          0 2020-03-08 09:50 ./dir1/dir1.txt
[mk@localhost ~]$ tar  -tvf  dir1.tar.bz2  # 查看打包兼压缩 .tar.bz2 的文件
drwxrwxr-x mk/mk          0 2020-03-08 09:50 ./dir1/
drwxrwxr-x mk/mk          0 2020-03-08 09:50 ./dir1/dir2/
drwxrwxr-x mk/mk          0 2020-03-08 09:50 ./dir1/dir2/dir3/
-rw-rw-r-- mk/mk          0 2020-03-08 09:50 ./dir1/dir2/dir3/dir3.txt
-rw-rw-r-- mk/mk          0 2020-03-08 09:50 ./dir1/dir2/dir2.txt
-rw-rw-r-- mk/mk          0 2020-03-08 09:50 ./dir1/dir1.txt
[mk@localhost ~]$ tar  -tvf  dir1.tar.Z  # 查看打包兼压缩 .tar.Z 的文件
drwxrwxr-x mk/mk          0 2020-03-08 09:50 ./dir1/
drwxrwxr-x mk/mk          0 2020-03-08 09:50 ./dir1/dir2/
drwxrwxr-x mk/mk          0 2020-03-08 09:50 ./dir1/dir2/dir3/
-rw-rw-r-- mk/mk          0 2020-03-08 09:50 ./dir1/dir2/dir3/dir3.txt
-rw-rw-r-- mk/mk          0 2020-03-08 09:50 ./dir1/dir2/dir2.txt
-rw-rw-r-- mk/mk          0 2020-03-08 09:50 ./dir1/dir1.txt
```

【例 92】 解开 tar 打包文件。

```
[mk@localhost ~]$ rm  -r  ./dir1  # 删除 dir1 目录
[mk@localhost ~]$ ll  # 查看是否已删除 dir1 目录
总用量 16
-rw-rw-r--. 1 mk mk 207 3 月  11 00:42 dir1.tar.bz2
-rw-rw-r--. 1 mk mk 195 3 月  11 00:38 dir1.tar.gz
-rw-rw-r--. 1 mk mk 435 3 月  11 01:05 dir1.tar.Z
-rw-rw-r--. 1 mk mk 958 3 月   8 10:19 dir1.zip
[mk@localhost ~]$ tar  -xvf  dir1.tar.Z  # 加 -xvf 参数解压缩
./dir1/
./dir1/dir2/
./dir1/dir2/dir3/
./dir1/dir2/dir3/dir3.txt
./dir1/dir2/dir2.txt
```

```
./dir1/dir1.txt
[mk@localhost ~]$ ll  # 查看解压缩是否成功
总用量 16
drwxrwxr-x. 3 mk mk  34 3 月   8 09:50 dir1  # 有生成
-rw-rw-r--. 1 mk mk 207 3 月  11 00:42 dir1.tar.bz2
-rw-rw-r--. 1 mk mk 195 3 月  11 00:38 dir1.tar.gz
-rw-rw-r--. 1 mk mk 435 3 月  11 01:05 dir1.tar.Z
-rw-rw-r--. 1 mk mk 958 3 月   8 10:19 dir1.zip
[mk@localhost ~]$ ll  ./dir1  ./dir1/dir2/dir3  # 深入查验目录树里的文件还原是否正确
./dir1:
总用量 0
-rw-rw-r--. 1 mk mk  0 3 月   8 09:50 dir1.txt
drwxrwxr-x. 3 mk mk 34 3 月   8 09:50 dir2
./dir1/dir2/dir3:
总用量 0
-rw-rw-r--. 1 mk mk 0 3 月   8 09:50 dir3.txt
```

【例 93】　解开以 gzip 压缩的 tar 打包文件。

```
[mk@localhost ~]$ rm  -r  dir1
[mk@localhost ~]$ tar  xvf  dir1.tar.gz   # 不用加 -z 参数
drwxrwxr-x mk/mk          0 2020-03-08 09:50 ./dir1/
drwxrwxr-x mk/mk          0 2020-03-08 09:50 ./dir1/dir2/
drwxrwxr-x mk/mk          0 2020-03-08 09:50 ./dir1/dir2/dir3/
-rw-rw-r-- mk/mk          0 2020-03-08 09:50 ./dir1/dir2/dir3/dir3.txt
-rw-rw-r-- mk/mk          0 2020-03-08 09:50 ./dir1/dir2/dir2.txt
-rw-rw-r-- mk/mk          0 2020-03-08 09:50 ./dir1/dir1.txt
```

【例 94】　解开以 bzip2 压缩的 tar 打包文件。

```
[mk@localhost ~]$ rm  -r  dir1
[mk@localhost ~]$ tar  -xvf  dir1.tar.bz2  # # 不用加 -j 参数
drwxrwxr-x mk/mk          0 2020-03-08 09:50 ./dir1/
drwxrwxr-x mk/mk          0 2020-03-08 09:50 ./dir1/dir2/
drwxrwxr-x mk/mk          0 2020-03-08 09:50 ./dir1/dir2/dir3/
-rw-rw-r-- mk/mk          0 2020-03-08 09:50 ./dir1/dir2/dir3/dir3.txt
-rw-rw-r-- mk/mk          0 2020-03-08 09:50 ./dir1/dir2/dir2.txt
-rw-rw-r-- mk/mk          0 2020-03-08 09:50 ./dir1/dir1.txt
```

【例 95】　解开以 compress 压缩的 tar 打包文件。

```
[mk@localhost ~]$ rm  -r  dir1
[mk@localhost ~]$ tar  -xvf  dir1.tar.Z  # # 不用加 -Z 参数
drwxrwxr-x mk/mk          0 2020-03-08 09:50 ./dir1/
drwxrwxr-x mk/mk          0 2020-03-08 09:50 ./dir1/dir2/
drwxrwxr-x mk/mk          0 2020-03-08 09:50 ./dir1/dir2/dir3/
-rw-rw-r-- mk/mk          0 2020-03-08 09:50 ./dir1/dir2/dir3/dir3.txt
-rw-rw-r-- mk/mk          0 2020-03-08 09:50 ./dir1/dir2/dir2.txt
-rw-rw-r-- mk/mk          0 2020-03-08 09:50 ./dir1/dir1.txt
```

【例 96】 还原 tar 包时候，通过增加参数 --exclude 排除某些文件。

```
[mk@localhost ~]$ rm  -r  dir1  # 删除原 dir1 目录才能还原
[mk@localhost ~]$ ll  # 检测
总用量 16
-rw-rw-r--. 1 mk mk 207 3 月  11 00:42 dir1.tar.bz2
-rw-rw-r--. 1 mk mk 195 3 月  11 00:38 dir1.tar.gz
-rw-rw-r--. 1 mk mk 435 3 月  11 01:05 dir1.tar.Z
-rw-rw-r--. 1 mk mk 958 3 月   8 10:19 dir1.zip
[mk@localhost ~]$ tar  -xvf  dir1.tar.gz  --exclude  dir3.txt
# 还原 tar 包，通过增加参数 --exclude 排除 dir3.txt 文件
./dir1/
./dir1/dir2/
./dir1/dir2/dir3/
./dir1/dir2/dir2.txt
./dir1/dir1.txt
```

【例 97】 归档 tar 包时候，通过增加参数 --exclude 排除某些文件。

```
[mk@localhost ~]$ rm  dir1.tar.bz2   # 删除 dir1.tar.bz2 包，准备再生成
[mk@localhost ~]$ tar  -jcvf  dir1.tar.bz2  ./dir1  --exclude=dir2
# 以 bz2 方式压缩归档时，排除 dir2 目录树
./dir1/
./dir1/dir1.txt
[mk@localhost ~]$ ll  # 验证 dir1 是否生成
总用量 16
drwxrwxr-x. 3 mk mk  34 3 月   8 09:50 dir1      #ok
-rw-rw-r--. 1 mk mk 144 3 月  11 02:16 dir1.tar.bz2
-rw-rw-r--. 1 mk mk 195 3 月  11 00:38 dir1.tar.gz
-rw-rw-r--. 1 mk mk 435 3 月  11 01:05 dir1.tar.Z
-rw-rw-r--. 1 mk mk 958 3 月   8 10:19 dir1.zip
```

```
[mk@localhost ~]$ rm  dir1.tar.gz   # 删除 dir1.tar.gz 包，准备再生成
[mk@localhost ~]$ tar  -zcvf  dir1.tar.gz  ./dir1  --exclude=dir2.txt
# 以 bz2 方式压缩归档时，排除 dir2.txt
./dir1/
./dir1/dir2/
./dir1/dir2/dir3/
./dir1/dir1.txt
```

总结与强调：等号后面直接跟目录名。跟了 ./ 或者绝对路径，会导致全部打包压缩而达不到预期效果。

错误写法：

```
tar -zcvf  tomcat.tar.gz  --exclude=tomcat/logs/  --exclude=tomcat/libs/ tomcat
```

正确写法：

```
tar –zcvf  tomcat.tar.gz  --exclude=tomcat/logs  --exclude=tomcat/libs tomcat
```

本章小结

本章主要学习了 Linux 的文件基础知识，包括常用文件的类别，目录常见的概述，并且对文件与目录的基本操作进行了深入的学习，能通过文件命令对文件进行查询、复制、移动、删除等操作，同时也能对目录的访问权限进行设置。在此基础上，还讲解了文件的压缩和归档、压缩和解压缩的使用。

习题

一、选择题

1. 使用 mkdir 命令创建新的目录时，在父目录不存在时先创建父目录的选项是（　　）。

A.–m　　B.–d　　C.–f　　D.–p

2. 在 Linux 中，要查看文件内容，可使用（　　）命令。

A.more　　B.cd　　C.login　　D.logout

3. 在 Linux 下重命名文件可用如下哪些命令？（　　）

A.ren　　B.ls　　C.mv　　D.copy

4. 一个文件的权限是 –rw–rw–r––，这个文件所有者的权限是什么？（　　）

A.read–only　　B.read–write　　C.write　　D.read

5.CentOS 7 系统默认使用的文件系统类型是（　　）。

A.swap　　B.ext3　　C.ext4　　D.ntfs

6. 文件 test1 的访问权限为 rw–r––r––, 现要增加所有用户的执行权限和同组用户的写权限 , 下列命令正确的是（　　）。

A.chmod a+x g+w test1　　B.chmod 765 test1　　C.chmod o+x test1　　D.chmod g+w test1

7. 用最详细方式查看 /etc/passwd 文件属性（其中包括文件时间属性）的命令是（　　）。

A.type/etc/passwd　　B. cat/etc/passwd　　C.ls–l /etc/passwd　　D.stat/etc/passwd

8. 在 CentOS 中，要上下查看文件内容，可使用（　　）命令。

A.more　　B.cd　　C.login　　D.less

9. 显示虚拟终端号的命令是什么？（　　）

A.stat　　B.type　　C.echo　　D.tty

10. 在 mk 用户的家目录下创建 a 目录及其下的 b 目录的时候，如果 a 这个目录不存在的话，要加参数（　　）。

A.–r　　B.–d　　C.–p　　D.–s

11./abc 目录访问权限原为 rw–r––r––，现要增加其他用户的进入该目录的权限 , 下列命令正确的是？（　　）。

A.chmod a+x g+w/abc　　B.chmod 765/abc　　C.chmod o+x/abc　　D.chmod g+w /abc

二、Linux 名词（命令）解释

1. 路径：

2. 目录文件：

3. 文件系统：

4. $ca > test1 test2 > test3：

5. cd ..：

6. tail–1/etc/passwd：

7. cat/etc/shells：

三、简答题

1. Linux 下的文件可以分为 5 中不同的类型，分别是什么？

2. cp 命令的功能是什么？

3. 对文件内容统计的命令是什么？

4. 对于无用文件，用户可以用什么命令将其删除？

5. gunzip 命令的功能是什么？

6. 将 hello.c 和 hello.c.bz2 打包后用 gzip 压缩并输出为 hello.tar.gz，显示执行过程。

7. 文件类型 , 可以为 p、d、l、s、c、b 和 －，请按顺序详细列出这七种文件中文名字。

8. 桌面 Linux 和嵌入式 Linux 的区别是什么？我们用的 CentOS 7 属于哪种？

四、应用题

1. 把光盘驱动器 sr0 挂载到 /cdrom/。

2. 创建 file1 到 file10 的连续号码新文件。

3. 为 /etc/passwd 文件创建名字为 pw 的软链接。

4. 切换到 student 用户的家目录，建立如下图所示的子树 (其中，文件 st1、st2、app1 和 app2 为目录，gcc1、gcc2、gcc3、gcc4 为普通文本文件)。

/home/student--gcc1

|---st1—app1—gcc2

|---st2—app2—gcc3,gcc4

5. 把 st2 目录与 gcc1 文件一起压缩打包成文件名为 test 的压缩包，并放在家目录下，并显示 test.tar.gz 的内容，然后把 test.tar.gz 包解包在 st1 目录下。

6. 在用户工作目录 /home 下，建立如下图所示的子树 (其中，文件 poem、song、user 和 user1 为目录，jiyeshi、test1 和 test2 为文件)。

/home

---poem

|---jiyeshi

---song

|---user

```
|---user1
---test1
---test2
```

✔ 综合实训

一、实训名称

文件 / 目录所有权限管理。

二、实训目的

掌握：查看、改变文件与目录的所有权限，自定义文件与目录的默认权限

三、实训工具或设备

Win7 计算机、投影、多媒体音箱、机房局域网、U 盘、vm9-cts74-r16-mk16-g-mk 老师版 -17311 通过 .rar 等。

四、实训步骤（必须有截图）

1. 解压 vm9-cts74-r16-mk16-g-mk 老师版 -17311 通过 .rar 到 c：。
2. 通过 vmware 软件进入虚拟机，并还原快照。
3. 按书本 2.6.3-2.6.4 的例题步骤做，提供相应的运行截图。

五、收获与体会

第三章
用户和组的管理

内容简介

- Linux 支持多用户，保证了各用户之间互不影响，系统管理员需要管理好系统中的用户和组，包括用户的添加、修改与删除，用户口令的管理以及用户组的管理等。本章主要介绍如何利用命令行方式来管理用户和组。

学习要求

- 了解用户账号和群组类型；熟练运用各种常用用户账号管理命令进行用户的管理；熟练运用各种常用用户组账号管理命令进行群组的管理。

3.1 用户账号和群组概述

Linux 系统是多用户、多任务的操作系统，各个用户可以通过本地或远程登录到系统并对自己的系统资源、文件设备有特殊的权利，每个用户都拥有一个唯一的账号，所以任意一个需要使用系统资源的用户，都必须首先向系统管理员申请一个账号，用户的账号一方面可帮助系统管理员对使用系统的用户对系统使用情况进行跟踪，限制用户对系统资源的访问，用户以该账号的身份，并输入对应的口令，只有当该用户账号存在，并且口令与账号相匹配的时候，用户才能进入 Linux 系统和自己的主目录；另一方面可帮助用户组织文件，为用户提供安全保护，在 Linux 系统中，任意文件都归属于某一特定的用户，而任意用户都至少隶属于一个群组，用户对文件的访问、读写等操作是受到系统约束的。

3.1.1 用户账号

在 Linux 系统中，不同类型用户所具有的权限和所要完成的任务都不同，用户的类型通过用户标志符 UID（User Identify）来区分，系统根据 UID 来给每个用户提供特定的工作环境，如：shell 版本、工作目录等，用来区分每个用户的任务、进程和文件等让每个用户的工作相互独立且不受干扰地进行。

Linux 系统的用户有 3 个类型，包括：系统管理员、系统用户和普通用户。

1. 系统管理员

也称超级用户、root 用户，对本机拥有最高的权限，类似于 Windows 系统中的 Administrator，UID=0。

2. 系统用户

是指与系统服务相关的用户，在安装 Linux 系统及部分应用程序时，会添加一些特定的低级权限用户账号，这些用户一般不允许登录到系统，仅用于维持系统或某个程序的正常运行，0<UID<1000。

3. 普通用户

通常需要由 root 用户或其他管理员用户创建，一般不需要改其默认配置，拥有的权限受到一定限制，一般在用户的宿主目录中拥有完整的权限，UID1000。

3.1.2 群组

在 Linux 系统中，将具备相同特征或功能的用户划分到一个群组，这样可方便系统管理员对用户访问权限进行管理，如：当多个用户需对某个文件夹或子文件拥有相同的访问权限时，需将用户定义到同一个群组，通过修改群组拥有对文件或文件夹的相关访问权限，该群组下的所有用户对该文件或文件夹就拥有相同指定的访问权限。

每个用户至少属于一个群组，即一个用户可以属于多个群组，其中一个称为该用户的主要群组，其他组称为用户的附属群组（标准群组）。

Linux 系统的群组分为 3 个类型，包括：私有组、系统组和标准组。

1. 私有组

在建立用户账号的时候，如果没有具体指定所属的组，Linux 系统就会建立一个和用户名相同的组，该组就是私有组，只存放了一个用户。

2. 系统组

可存放多个用户，存放在该组的用户均具有所在组的所有权利。

3. 标准组

Linux 系统自动建立的组。

3.2　用户与组文件

Linux 系统采用文本文件来保存账号的信息，重要的文件包括：用户账号文件 /etc/passwd、用户影子文件 /etc/shadow、用户组账号文件 /etc/group 等，Linux 用户登录系统的过程其实就是系统读取并核对这几个文件的过程。

3.2.1　用户账号文件

/etc/passwd 文件时用户账号管理过程中最重要的一个文件，该文件为纯文本文件，用以保存除了用户口令以外的用户账号信息，其中包括：用户名和 UID，所有用户都可以读取该文件，所有注册用在该文件里都有一个对应的记录行，记录账号的重要信息。

/etc/passwd 文件部分内容如下：

```
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:/sbin/nologin
daemon:x:2:2:daemon:/sbin:/sbin/nologin
adm:x:3:4:adm:/var/adm:/sbin/nologin
……
```

文件中的每一行描述一个用户配置信息，第一行是系统管理员，紧接着的是系统用户，普通用户通常在文件的末端。/etc/passwd 文件每一行通过“：”将用户的每个属性信息分为 7 个部分，文件格式如下：

用户账号：口令：UID：GID：个人资料：主目录：Shell

每个字段的含义及说明如表 3-1 所示。

表 3-1　/etc/passwd 各字段含义及说明表

字段序号	含义	说明
1	用户账号	用户在登录时所使用的名称，是唯一的，由字母、数字和符号组成
2	口令	是经过加密后保存在 /etc/shadow 文件中，这部分的内容在总是用“x”来填充
3	UID	用户 ID，一般由一个整数表示，范围是 0~65535，Linux 系统中所有 UID 具有唯一性
4	GID	组 ID，一般由一个整数表示，Linux 系统中的每个用户都属于某个组，每个用户组除了组名外还会有一个相应的组 ID，相同组会拥有相同的 GID
5	个人资料	包含用户全名、号码、住址等描述信息
6	主目录	用户登录 Linux 系统后默认所在的目录就是主目录，用来给用户存储个人文件。root 的主目录为 /root，除非创建用户时有指定目录，否则普通用户通常为 /home 下的与同用户名同名的子目录
7	Shell	用来定义用户登录 Linux 系统后使用的命令解释器，默认需要执行的程序是 /bin/bash

3.2.2　用户影子文件

因为所有用户对 /etc/passwd 文件都有读取的权限，虽然口令经过加密处理，但也不能完全避免有人会获取加密后的口令，因此，为了提高 Linux 系统安全性，系统中的用户账号口令是经过加密后保存

在 /etc/shadow 文件中，只有系统管理员才能读取 /etc/shadow 文件，这部分的内容在 /etc/ passwd 文件中每一记录行的口令字段总是用“x”来填充。

/etc/shadow 文件部分内容如下：

```
root:66fwAek.nZ$dIxcvynnakhnvuyZOmP2cJMnq4vAPq0xnuAyjYtqq1RUhJtvSIwx5361FZObl1aw7BKCCUUp63UcmUKI/IwHK/:17636:0:99999:7:::
bin:*:16579:0:99999:7:::
daemon:*:16579:0:99999:7:::
adm:*:16579:0:99999:7:::
……
```

与 /etc/passwd 文件类似地，/etc/shadow 文件中的每一行描述一个用户配置信息，通过“：”将用户的每个属性信息分为 9 个部分，文件格式如下：

用户账号：加密口令：最后一次修改时间：最小时间间隔：最大时间间隔：警告时间：不活动时间：失效时间：标志字段

每个字段的含义及说明如表 3-2 所示。

表 3-2　/etc/shadow 各字段含义及说明表

字段序号	含义	说明
1	用户账号	用户在登录时使用的名称，是唯一的，由字母、数字和符号组成
2	加密口令	加密后的口令。若为“*”，则表示对应的用户被禁止登录；若为“！”，则表示用户被锁定，无法登录系统
3	最后一次修改时间	表示用户从 1970 年 1 月 1 日期到最近一次修改口令的间隔天数
4	最小时间间隔	口令自最近一次修改后，需要间隔多少天才能被再次修改。若为 0，则表示没有时间限制
5	最大时间间隔	口令自最近一次修改后，需要间隔多少天定要被再次修改。若为 99999，则表示口令没有必须要修改
6	警告时间	表示系统提前多少天警告用户口令即将过期
7	不活动时间	表示用户口令过期多少天后，系统将自动禁止该用户账号
8	失效时间	表示用户账号的生存期，超过该时间，则失效
9	标志字段	保留字段，目前为空

3.2.3　用户组账号文件

/etc/group 文件保存所有用户组账号的信息，所有用户组成员都可以读取，与用户账号一样，用户组也是由唯一的一个身份来标记，用 GID 表示，对某个文件的访问是以其 UID 和 GID 为基础的。

/etc/group 文件部分内容如下：

```
root:x:0:root
bin:x:1:root,bin daemon
daemon:x:2:root,bin,daemon
```

```
adm:x:3:root,bin,adm
……
```

/etc/group 文件中的每一行描述一个用户组信息，每一行通过“：”将用户的每个属性信息分为 4 个部分，文件格式如下：

用户组名：用户组口令：GID：组成员列表

每个字段的含义及说明如表 3-3 所示。

表 3-3　/etc/group 各字段含义及说明表

字段序号	含义	说明
1	用户组名	用户组的名称
2	用户组口令	通常不需要设置，总是以“x”来填充
3	GID	用户组的 ID，是唯一的
4	组成员列表	该用户组包含的所有用户账号，用户间用“，”分开

3.2.4　验证用户和组文件

1. 查看所有用户信息

cat /etc/passwd

2. 查找某个用户

cat /etc/passwd|grep 用户名

3. 查看所有用户组信息

cat /etc/group

4. 查找某个用户组

cat /etc/group|grep 用户组名

5. 查看当前登录用户的组内成员

groups

6. 查看某个用户所在的组，以及组内成员

groups 用户名

3.3　管理用户账号

所有用户账号的管理其实就是对这几个文件的内容进行添加、修改和删除的操作。

3.3.1 添加用户

在 Linux 系统中，添加新用户使用 useradd 命令来实现，用来建立用户账号和创建用户的起始目录，只有系统管理员才有权限使用该命令。useradd 命令格式如下：

```
useradd [ 选项 ] 用户名
```

主要选项参数如表 3-4 所示。

表 3-4 useradd 命令主要选项参数

参数	说明
-c	加上备注文字，备注文字保存在 passwd 的备注栏里
-d	指定用户登录时主目录的初始目录，默认为 /home/username
-e	指定用户账号的有效期限，格式为 MM/DD/YY，缺省表示永久有效
-f	在口令过期后多少天关闭该用户账号
-g	指定用户账号所在的群组
-m	新用户的登录目录
-n	取消建立以用户名为名的群组
-r	建立系统账号
-s	指定用户登录后所使用的 shell
-u	指定用户的 UID

3.3.2 修改用户信息

1. 修改用户口令

系统管理员添加完新用户之后，应立即用 passwd 命令为新用户设置口令，否则在没有设置口令的用户账号是锁定处于状态，系统管理员可以修改其他用户的口令，但普通用户只能在没有被系统管理员锁定的前提下修改自己的口令。passwd 命令格式如下：

```
passwd [ 选项 ] 用户名
```

主要选项参数如表 3-5 所示。

表 3-5 passwd 命令主要选项参数

参数	说明
-d	删除用户口令，仅系统管理员有权限操作
-f	强制操作，仅系统管理员有权限操作
-i	口令过期后多少天，用户被禁用，仅系统管理员有权限操作
-k	保留用户账号在过期后仍能使用
-l	锁定用户无权限修改口令，仅系统管理员有权限操作
-n	两次口令修改间的最小天数，仅系统管理员有权限操作
-S	查询用户口令状态，仅系统管理员有权限操作
-x	两次口令修改间的最大天数，仅系统管理员有权限操作
-u	解除锁定

续表

参数	说明
-w	距多少天提醒用户修改口令，仅系统管理员有权限操作
-stdin	允许通过输入修改用户口令

2. 修改用户账号属性

在 Linux 系统中使用 usermod 命令来修改用户账号的各项属性，仅能由系统管理员使用。usermod 命令格式如下：

```
usermod [ 选项 ] 用户名
```

usermod 命令的选项及其功能大部分与 useradd 命令相同，另新增的选项主要选项参数如表 3-6 所示。

表 3-6　usermod 命令主要选项参数

参数	说明
-l	修改用户账号名称
-L	锁定用户账号
-U	解锁用户账号

3.3.3　删除用户

想要删除指定的用户账号，可用 userdel 命令来实现。userdel 命令格式如下：

```
userdel [ 选项 ] 用户名
```

主要选项参数如表 3-7 所示。

表 3-7　userdel 命令主要选项参数

参数	说明
-f	强制删除用户账号，即使用户当前已登录
-r	完全删除用户账号，不保留任何文件

3.4 管理群组

3.4.1　创建用户组

在 Linux 系统中，创建用户组使用 groupadd 命令来实现，用来将新的用户组加入到系统中，只有系统管理员才有权限使用该命令。groupadd 命令格式如下：

```
groupadd [ 选项 ] 用户组名
```

主要选项参数如表 3-8 所示。

表 3-8　groupadd 命令主要选项参数

参数	说明
-f	加入已有的组时，程序退出
-g	指定用户组的 GID，必须唯一的
-o	一般与 -g 选型同时使用，表示新的 GID 可以与已有的 GID 相同，不必唯一

3.4.2　修改用户组属性

创建用户组后，可根据需要对用户组属性进行修改，修改用户组属性主要包括修改 GID 和用户组名，都可以用 groupmod 命令来实现修改。

1. 修改 GID

对 GID 进行重新修改，前提是不能与已有的 GID 重复，且不会改变用户组名，可以用带 -g 参数的 groupmod 命令来实现，命令格式如下：

```
groupmod -g 新 GID 用户组名
```

2. 修改用户组名

对用户组名进行重新修改，不会改变用户组名，可以用带 -n 参数的 groupmod 命令来实现，命令格式如下：

```
groupmod -n 新用户组名旧用户组名
```

3.4.3　删除用户组

想要删除指定的用户组账号，仅能由系统管理者用 groupdel 命令来实现。groupdel 命令格式如下：

```
groupdel 用户组名
```

当该用户组中存在用户账号时，必须先删除组内所有用户账号后，然后才能删除用户组。

3.5　使用管理器管理用户和组

系统管理者在 Linux 中拥有全部权限，在 Linux 中执行命令行一般需要系统管理者权限，尤其是影响系统文件的命令，也正因为系统管理者权限如此强大，所以一般建议只有在必要时使用，这样避免了重要系统文件意外受损而无法恢复。

3.5.1　启动用户管理者

1. 从终端获取系统管理员权限

打开终端，输入“su -”，然后按回车键，这样就默认以系统管理员身份登录，出现“password：”提示时，输入系统管理员口令，输入正确口令后会发现命令提示符应该以“#”，而不是“$”结尾，Linux 系统有两种命令提示符：“#”表明是系统管理员的权限；“$”表明是普通用户的权限。

通过“su -”命令以系统管理员登录后，就可以运行需要系统管理员权限的任意命令了。su 命令将保留到会话结束，因此每次需要运行命令时，不需要再重新输入系统管理员口令。

通过 sudo（超级用户执行）命令，可临时以系统管理员身份运行其他命令。这是大多数用户运行系统管理员权限命令的最佳方式，因为这样既不用维护系统管理员，也不需要知道系统管理员口令，只要输入自己的用户口令，就能获得临时的 root 权限。输入“sudo command”，并按回车键。注：提示输入口令时，须输入用户口令，而不是 root 用户口令。

2. 以系统管理员身份登录

登录 Linux 系统时，如果系统管理员账号并未锁定，而且在知道口令的前提下，当系统提示以用户账户登录时，就可以输入“root”作为用户名，并输入 root 账户口令，很多时候，系统管理员口令可能是“password”。

如果不知道系统管理员口令，或者忘记口令，可以重置口令，需要以恢复模式启动系统，才能更改口令。

3.5.2　创建用户

【例 1】　新建一个名为 test 的新用户，其 UID 为 666，主目录存放在 /home/test，备注文字为“normal user”，过期时间为 2035 年 12 月 31 日：

```
# useradd -u 666 -d /home/test -c “normal user” -e 12/31/2035 test
```

3.5.3　修改用户属性

【例 2】　先为用户 test 设置口令，然后锁定用户 test：

```
# passwd tes     # 为用户 test 设置口令
Changing password for user test.
New password:  # 输入新口令，输入的口令无回显
Retype new password: # 重新输入新口令，两次口令要求一致
Passwd: all authentication token updated successfully.
# passwd -l test # 锁定用户 test
Locking password for user test.
Passwd: Success# 锁定成功
```

【例 3】　用带有“-u”选项的 passwd 命令解除锁定的用户后才能重新使用：

```
# passwd -u test
Unlocking password for user test.
Passwd: Success      # 解锁成功
```

【例 4】 将 test 的用户账号名称修改为 test1，然后锁定：

```
# usermod -l test1 test
# usermod -L test1
```

3.5.4 创建用户组

【例 5】 创建一个名为 computer 的用户组，并设置其 GID 为 600：

```
# groupadd -g 600 computer
```

3.5.5 修改用户组属性

【例 6】 将用户组 computer 的 GID 改为 560：

```
# groupmod -g 560 computer
```

【例 7】 将用户组 computer 组名改为 computer1：

```
# groupmod -n computer1 computer
```

3.6 常用的账户管理命令

1. 切换用户身份

使用 su 命令可以更改 UID 和 GID，从而达到从当前用户切换到指定用户的目的，普通用户账号切换到其他用户账号时，需要输入被切换用户的口令，切换后即拥有该用户的权限。su 命令格式如下：

```
# su [ 选项 ] 用户名
```

主要选项参数如表 3-9 所示。

表 3-9 su 命令主要选项参数

参数	说明
-	当前用户不仅切换为指定用户的身份，同时所用的工作环境也切换为该用户的环境
-c	仅执行一次命令，执行后自动切换回来，该选项后通常会带有要执行的命令
-l	同“-”的使用类似，即在切换用户身份的同时，完整切换工作环境，但后面需要添加欲切换的使用者账号
-p	切换到指定用户的身份，但不改变当前的工作环境

su 和 su - 的区别：

当命令为“su”时，默认切换到系统管理者，仅改变权限，但不改变其环境变量。

当命令为“su -”时，切换到系统管理者的环境变量和权限。

当命令为“su 用户名”时，从当前用户账号切换到已存在的指定用户账号，但用户的环境设置不变。

当命令为“su - 用户名”时，从当前用户账号切换到已存在的指定用户账号，并且使用新用户账号的环境变量和权限。

执行 exit 命令可以返回原来的用户账号。

2. whoami 和 who am i 命令的用法和区别

whoami 命令和 who am i 命令是不同的两个命令，前者用来返回当前执行操作的用户名，后者则用来返回登录当前 Linux 系统的用户名。

以下为了更好地区分两者，使用用户名为“stu”登录 Linux 系统，然后执行命令：

```
$ whoami
stu
$ who am i
stu pts/0 2020-2-1 16:30 (:0.0)
```

以下为了更好地区分两者，使用 su 命令切换到系统管理员，再执行以上命令：

```
$ su root
Password:
# whoami
root
# who am i
stu pts/0 2020-2-1 16:32 (:0.0)
```

由以上两种输出结果来看，在还没切换到系统管理员之前，whoami 命令和 who am i 命令输出是一样的，但使用了 su 命令切换用户后，使用 whoami 命令返回当前执行操作即切换后的用户名，而 who am i 命令返回登录系统的用户名。

3. 将用户加入或移除群组

gpasswd 命令是 Linux 系统下用户组文件 /etc/group 和 /etc/gshadow 管理工具，为避免系统管理员太忙碌而无法及时管理群组，我们可以使用 gpasswd 命令给群组设置一个群组管理员，代替系统管理员完成将用户加入或移出群组的操作。

gpasswd 命令格式如下：

```
# gpasswd [ 选项 ] 用户组名
```

主要选项参数如表 3-10 所示。

表 3-10　gpasswd 命令主要选项参数

参数	说明
	选项为空时，表示给群组设置口令，仅供系统管理员使用
-a	后面带用户名：user，将指定用户加入指定群组
- A	后面可带多个用户：user1，user2…，将群组的控制权交给指定若干用户管理，也就是说将指定若干用户设为群组的管理员，仅供系统管理员使用
-d	后面带用户名：user，将指定用户移出指定群组
-M	后面可带多个用户：user1，user2…，将指定若干用户加入指定群组，仅供系统管理员使用
-r	清除群组口令，仅供系统管理员使用
-R	让群组口令失效，仅供系统管理员使用

【例 8】　创建新用户组 gro，将用户 lamp1 和 lamp2 加入 gro 群组，并将群组交给 lamp1 管理：

```
# groupadd gro          # 创建群组 gro
# gpasswd gro           # 设置口令
Changing password for group gro.
New password:           # 输入新口令
Retype new password:    # 重新输入新口令
Passwd: all authentication token updated successfully.
# gpasswd -a lamp1 gro    # 将用户 lamp1 加入群组 gro
# gpasswd -a lamp2 gro    # 将用户 lamp2 加入群组 gro
# gpasswd -A lamp1 gro    # 设置群组管理员为 lamp1
```


本章小结

本章介绍了 Linux 系统里用户账号和群组的相关概念，掌握用户账号和群组管理命令对后面权限管理具有非常重要的作用。用户账号文件 /etc/passwd、用户影子文件 /etc/shadow、用户组账号文件 /etc/group。添加用户使用 useradd 命令，修改用户账号口令使用 passwd 命令，修改用户账号属性使用 usermod 命令，删除用户账号 userdel 命令；创建用户组使用 groupadd 命令，修改用户组属性使用 groupmod 命令，删除指定的用户组账号使用 groupdel 命令。另外还介绍了几个常用的账户管理命令：临时切换用户身份的 su 命令，返回当前执行操作的用户名的 whoami 命令，返回登录当前 Linux 系统的用户名的 who am i 命令，将用户加入或移除群组使用 gpasswd 命令。

习题

一、选择题

1. 以下哪个目录存放用户账号口令信息？（　　）

A. /boot　　B. /dev　　C. /etc　　D. var

2. 默认情况下系统管理员创建了一个新用户，会在（　　）目录下创建一个用户主目录。

A. /etc　　B. /home　　C. /root　　D. /usr

3. 以下（　　）命令可以将普通用户转换为系统管理员。

A. passwd　　B. su　　C. super　　D. tar

4. 在运行多用户模式下，可以用 Ctrl+Alt+F 切换（　　）个虚拟用户终端。

A. 2　　B. 3　　C. 6　　D. 12

5. Linux 文件权限一共十位长度，分成四段，其中第三段的内容表示为（　　）。

A. 文件所有者所在组的权限　　B. 文件类型

C. 文件所有者的权限　　D. 其他用户的权限

6. 用户账号口令存放在（　　）文件中。

A. /etc/passwd　　B. /etc/shadow　　C. /etc/group　　D. /etc/gshadow

7. usermod 命令无法实现的是（　　）。

A. 用户账号重命名　　B. 加锁或解锁用户账号

C. 加锁或解锁用户账号口令　　D. 删除指定用户账号和对应的主目录

8. 添加新用户是可以使用参数（　　）到指定用户目录。

A. -c　　B. -d　　C. -p　　D. -u

9. 修改用户账号口令可使用（　　）命令。

A. passwd　　B. passwd -d　　C. passwd -k　　D. passwd -l

10. 显示当前执行操作的用户名使用（　　）命令。

A. passwd　　B. usermod　　C. whoami　　D. who am i

二、程序编写题

1. 创建用户 kengoo，附加组为 bin 和 root，默认 shell 为 /bin/csh，注释信息为“kengoo Distribution”。

2. 创建一个新用户 admin，指定登录时起始目录 /www，指定 UID 为 526，加入 admins 附加组中，且不检查 UID 唯一性。

3. 修改 stu 用户 UID 为 532、主组 root、添加新的附加组 apache 且保留旧的附加组，然后锁定用户。

4. 写出锁定 marry 的两种方法。

✔ 综合实训

一、实训题目

Linux 系统用户和组的管理基本命令复习和练习。

二、实训目的

掌握在 Linux 系统下利用命令行方式实现用户的管理。

掌握在 Linux 系统下利用命令行方式实现组的管理。

三、实训内容

某公司今年新录用员工 5 人：Zhaoyi、Qianer、Sunsan、Lisi、Zhouwu，每名员工的分工不同，分别隶属于不同部门，具有不同的权限。请分别以员工姓名拼音作为用户名设置用户账号，初始口令为“123456”，并实现分组：Sunsan、Lisi 属于管理员用户，属于 Manager 组，Zhaoyi、Qianer、Zhouwu 属于普通用户，属于 Normal 组，另由于 Qianer 属于外派出差外地，所以需要暂时禁用其用户账号。请根据要求写出具体的命令。

第四章

文件系统管理

内容简介

- 这一章首先讲述文件系统的基本知识，介绍常见的文件系统，并且介绍逻辑卷管理器和一些特殊的文件系统。然后根据一切皆文件的思想展开论述，然后介绍 GNU/Linux 虚拟文件系统的目录结构规范，并介绍设备的符号链接、硬链接与软链接，还有管道的概念，并提到特殊的文件。再认识重定向的概念：包括输入重定向与输出重定向，并扼要介绍文件及目录的访问权限及其属性。

学习要求

- 理论与实操相结合，本章中的示例命令皆可在虚拟化环境中进行试验。
- 认知磁盘分区、文件系统和（逻辑）卷的概念，并能够做出区别。
- 了解 GNU/Linux 所使用的文件系统的目录结构及常用文件存放的位置。
- 能够区分硬链接与软链接。
- 认知在 GNU/Linux 中“一切皆文件”的思想。
- 了解管道命令符的应用。
- 认识输入及输出重定向及重定向及其模式的区别。
- 了解文件及目录的权限访问及属性。

4.1 文件系统基础知识

自 1956 年诞生以来，硬盘就是计算机主要的数据存储媒介。认识以硬盘为主的文件系统和虚拟文件系统的概念是理解掌握文件系统管理的基础，而文件系统管理是 GNU/Linux 运行与维护的基本组成部分。

4.1.1 硬盘结构

经典的硬盘机（hard disk drive）是计算机的主要外部存储设备，是非易失性数据存储器，它的内部的组成部件包括：在同一个圆心位置上的若干个能记录磁性的盘片（platter），每个盘片通常有至少一个可用作磁记录的面，且每个盘面对应有一个悬浮在盘面上方的磁头（head）对数据存储区域进行读写操作，如图 4-1 所示。每个盘面又可以划分为很多半径大小不一的同心圆。所有盘面上半径相同的同心圆组合在一起就形成了柱面（cylinder）的概念，每个柱面上有若干个扇区（sector），其经典容量能存储 512 个字节（byte）的数据，而使用先进格式化（Advanced Format）的硬盘上的每个扇区能存储 4 KiB（$4 \times 2^{10} = 4096$ 个字节）。每个扇区上除了储存数据的区域，还存在标记扇区位置信息和校验码的区域，一般这两个区域在使用计算机的过程中不由操作系统访问。

传统硬盘的数据寻址方式是通过指定柱面、磁头还有扇区的组合来进行寻址（称作“柱面 - 磁头 - 扇区”，CHS），其中磁头的位置同时也唯一确定了数据所在的盘面。

传统硬盘对盘面进行读写的最小储存单元是一个扇区。在 21 世纪初，伴随着硬盘的技术发展，这种寻址方式已经显得相对低效。这时就发展出了逻辑区块地址（Logical Block Address）的寻址方式：通过将每一个扇区作为一个区块单元，使整块硬盘的全部储存区块成为由一个可计数的连续有序索引序列来进行标记各储存单元，利用上述索引来进行不可再细分的读写操作寻址。

数据簇（从集，或曰单位配置，data cluster）是对应在操作系统中对外部储存设备（包括硬盘机、软盘机、磁带机和光碟机等）进行数据存取的基本单位，其确切尺寸在建立文件系统时可以指定。一般建议与实体储存介质的最小储存单元的尺寸相一致。

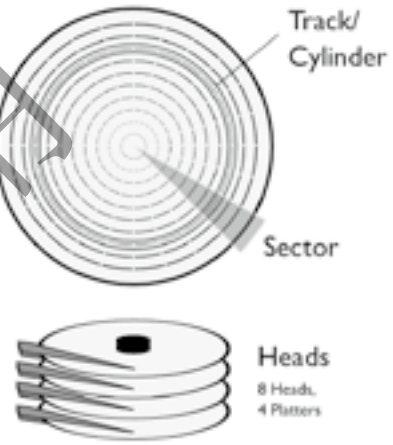

图 4-1 硬盘的结构

4.1.2 系统引导

传统的 IBM 兼容机能够使用基于主引导记录（Master Boot Record）扇区的分区表（disk partition table）来进行标记硬盘内数据存储区域划分情况。这种分区表最多能够存储 4 个主要（或称“基本”）的分区（primary partition），且每个分区最大可以支持 2 TiB。

随着硬盘技术的发展，在大量数据存储的应用场景中，存在超过以上 MBR 限制的数据存储需求。故发展出了 GPT（GUID 分区表），其增强了硬盘分区记录的校验，并增加了备份 GPT，以防主 GPT 损坏。

在 x86 型计算机的启动过程中，传统是需要读取第一个扇区的主引导记录（MBR，master boot record）来进行开启引导。由于传统的 BIOS（基本输入及输出系统）功能及安全性稍逊，故发展出了具有 EFI（可扩展固件接口）的 BIOS。EFI 系统分区（EFI System Partition， ESP）是具有 EFI BIOS 的计算机进行启动引导所必需的硬盘分区，其使用 FAT 文件系统（通常是 FAT32）。具有 EFI 的计算机主要的启动引导方式不再只依赖 MBR，而是可以在硬盘中寻找被标记为 ESP 的硬盘分区中是否存有能引导启动计算机加载主操作系统的 EFI 应用（通常位于 ESP 的 /boot 目录中）。因 EFI 的存在，这使得在只有一个硬盘且使用 EFI BIOS 的计算机系统中的储存设备上进行硬盘分区具有了必要性。

4.1.3　硬盘的分区、逻辑卷和文件系统之间的关系

硬盘分区（partition）是直接建立在硬盘中的实体存储区域划分。

在硬盘使用了一段时间过后有可能会使硬盘产生一定的坏块。除了可以把硬盘送回原厂进行低级格式化屏蔽坏块以外，一般用户也可以对硬盘进行读写扫描，将坏块较多的硬盘实体区域划分为单独的分区，从而避开在坏块较多的实体储存位置上面进行数据存储。这便是对硬盘进行分区的重要原因。

而在操作系统层面上，传统是可以把计算机外部数据存储部件包括硬盘中的分区、软碟、光碟，乃至 U 盘这些存储器当中的实体存储区域用一个称作卷（volume）的概念相统一。其中，当一个完整的外部储存器整体或一个单独且完整的磁盘分区对应一个卷的时候，可以在使用上将卷这个逻辑概念等同于一整个外部储存区域。在每个卷上可以进行对应数据存储的逻辑划分，意味可以在每个卷上建立文件系统。这种能够找到与实体储存设备唯一对应、属于在独立的文件系统上可以进行数据访问的一整块连续的完整区域上组成的卷，亦可称作实体卷（physical volume）。磁盘分区与卷的概念的主要区别在于：卷通常是需要在操作系统启动并加载相应驱动程序后才能进行管理的逻辑设备。

文件系统（file system）是为了在人机交互和在数据及其储存管理的过程中，以符合人的思维方式，将储存在设备上的数据，以抽象的文件（file）和目录树（directory tree）的概念，替代计算机数据通信过程中通过区块地址对储存设备进行数据寻址及储存管理而设计的一套方法。

每一个文件是一组相关数据在文件系统中用路径和文件名进行唯一标记的数据集合：每一个文件都会有一个文件名，存放在某一目录（directory，在部分操作系统当中亦称作“文件夹（folder）”）中。文件通常还会记录其最后修改时刻等属性，以及所有者、所有者所在用户组及以上两者对应的存取权限，部分较新的文件系统类型通常还可以储存文件相关由用户自定义的元数据。通常文件名不应该包含一些可以在操作系统或文件系统上做特殊用途的字符，但是在部分较新的操作系统上可以用引号包裹的方法指定含部分特殊字符的文件名。

在不同的操作系统上，支持的文件系统有所差异。在使用上，文件系统等价于在格式化磁盘时所指定的“格式”。基于 Windows NT 的操作系统主要使用 FAT32 和 NTFS 来作为主要的文件系统，在 macOS 上主要使用 HFS 和 HFS+，而在 GNU/Linux 中，对于硬盘储存而言，目前主流是使用 ext4、XFS 或 Btrfs 等日志型文件系统。

在为 GNU/Linux 而设计的文件系统通常会在格式化的时候确定其所能容纳的最多文件及目录的数量，这一限制主要是因为每个文件和目录都会有一个索引节点（index node，亦称 i-node、i 节点）与之对应。

4.1.4　虚拟文件系统

在 GNU/Linux 中为了将不同的文件系统实现通过一系列统一的操作接口暴露给用户，这种方式我们叫作虚拟文件系统（synthetic/pseudo file system）。虚拟文件系统是通过操作系统上的文件系统程序将自己进行隐藏和区别，并将共同的特性接口暴露给用户，相当于给各种不同的文件系统套了一层虚拟层，使得用户能用相同或相似的方法（例如命令）对各种不同的实体文件系统中的文件和目录进行操作。在功能上虚拟文件系统，主要让上层的软件能够用一种统一的方式来跟不同的文件系统沟通，在操作系统与各种实体文件系统之间，虚拟文件系统提供了标准化的接口，让操作系统能够以相对统一的标准模式来支持各种不同的实体文件系统。

4.1.5 文件的概念

文件是具有独立意义的一组信息，其可以保存在储存设备上并能按名存取。故在一般用户对文件的使用过程中无须了解文件在储存设备上的实体区块地址。

有了文件系统的概念，只需要确定路径（path）及文件名（filename），即可对存储的数据进行结构化的管理。在 FAT 文件系统当中，目录树及其中的文件名会存放在文件分配表（File Allocation Table）当中。

4.1.6 日志文件系统

因磁盘通常会含有缓存的部件：用磁盘储存数据时，先将要写入的数据存至缓存，并同时由磁头寻址到真正写入数据的区块地址处，这种方法可以提升将数据写入磁盘时的性能。亦因为缓存的存在，在使用传统的基于文件分配表的文件系统的磁盘在数据写入的过程中，当数据传输到缓存但还没被完整写入磁盘的时候，若计算机发生故障，则文件分配表中所记载的文件块和实际文件块所储存的内容有可能出现不一致的情况，而日志文件系统（journaling file system）则是在软件层面上做出对这种情况的应对措施。在当操作系统向文件系统提交变更时，日志文件系统先将文件变化的相关信息写入一个称作日志的区域，然后再将实际文件数据的变化写入文件系统的数据存储区域，将原本在传统文件分配表文件系统上具有原子性（或称不可中断性，atomic）的操作分作多步走，使得文件系统的一致性和可恢复性得到提高。

在 GNU/Linux 上常见支持的日志文件系统包括 ext3、ext4、XFS.Btrfs、HFS+（Hierarchical File System Plus）、JFS（Journaled File System）和 NTFS（New Technology File System）等。

各种文件系统都具有不同的特性，其中最主要的区别就是所支持的最大单文件尺寸和最大单卷文件系统尺寸，而以上两个特性通常又与计算机系统中所载入的文件系统驱动程序有关。可参考下方 Red Hat 公司给出的两个对比表格中的数据（如表 4-1 和表 4-2 所示）作为参考：

表 4-1 单文件尺寸（确认值和［理论最大］值）

File system	RHEL 3	RHEL 4	RHEL 5	RHEL 6	RHEL 7
Ext2/3	1TiB (3.0) 2TiB (3.5+)	2TiB	2TiB	2TiB	2TiB
Ext4	n/a	n/a	16TiB (5.6+)2	16TiB	16TiB
GFS1	2TiB	16TiB [8EiB]	16TiB [8EiB]	n/a	n/a
GFS21	n/a	n/a	100TiB (5.3+) [8EiB]	100TiB [8EiB]	100TiB [8EiB]
XFS3	n/a	n/a	100TiB [8EiB]	100TiB [8EiB]	500TiB [8EiB]

表 4-2 文件系统容量尺寸（确认值和［理论最大］值）

文件系统	RHEL 3	RHEL 4	RHEL 5	RHEL 6	RHEL 7
Ext2/3	1TiB(3.0) 2TiB(3.5+) [8TiB]	8TiB	8TiB (5.0), 16TiB (5.1+)	16TiB	16TiB
Ext4	n/a	n/a	16TiB [1EiB] (5.6+)	16TiB [1EiB]	50TiB [1EiB]
GFS	2TiB	16TiB [8EiB]	16TiB [8EiB]	n/a	n/a

续表

文件系统	RHEL 3	RHEL 4	RHEL 5	RHEL 6	RHEL 7
GFS21	n/a	n/a	100TiB (5.3+) [8EiB]	100TiB [8EiB]	100TiB [8EiB]
XFS3	n/a	n/a	100TiB [16EiB]	300TiB [16EiB]	500TiB [16EiB]

注：所使用的二进制前缀：

TiB = Tebibyte = 2^{40}

PiB = Pebibyte = 2^{50}

EiB = Exbibyte = 2^{60}

4.1.7　FAT 文件系统

文件分配表（file allocation table）文件系统是微软发明的文件系统，原本设计是为 DOS（磁盘操作系统）使用。截至 21 世纪的第 2 个 10 年，FAT 文件系统主要发展出了 4 个版本。其中 3 个经典的版本的命名与其文件系统内部的数据定址位宽相关，分别是：FAT12、FAT16，还有 FAT32；另外还有一个被称作 exFAT 的版本，主要用于大于 32 GB 的 SDXC 闪存卡等大容量储存设备。其中，FAT32 文件系统的最大单文件限制在 4 GiB 以内，并且文件系统的最大尺寸被限制在 8 TiB 以内。计算机中的 EFI 系统分区通常使用 FAT32 文件系统。

在 GNU/Linux 中，最著名的 FAT 文件系统进行读写的驱动程序为 vfat，在名字上体现了与为 Windows 95 开发的长文件名（Long File Name）版本的 FAT16 文件系统驱动程序相兼容的特性。

正因为 FAT 本非为 GNU/Linux 而设，故其结构上并没有 i-node 的概念。如下所示，当使用 df -i 命令查询系统中所挂载的文件系统中仍可用的文件节点数量时，尽管此处 /dev/sda 是已存有一个文件的 FAT 文件系统，但其 i-node 相关数据项皆为 0：

```
[user@courseware ~]$ ls /mnt/vfat/
demo.txt
[user@courseware ~]$ df -i /mnt/vfat/
文件系统 inode 已用 (I) 可用 (I) 已用 (I)% 挂载点
/dev/sdb          0      0      0        - /mnt/vfat
```

4.1.8　Ext4 文件系统

其前身可追溯到 1980 年 Andrew Stuart Tanenbaum 所做的 MINIX 文件系统，这是一个参照 Unix 文件系统为教学用途精简而做的文件系统，后被 Linux 核心所用。为提高 MINIX 文件系统的存取性能，Rémy Card 开发出了延伸文件系统（extended file system，简称 ext）。ext 文件系统的单文件尺寸仅限在 2 GiB 内。为了解决 ext 的单文件尺寸限制，第二代延伸文件系统（ext2）诞生了，其可支持最大 2 TiB 的单文件。而后推出的 ext3 增加日志功能。随时代变迁，在 21 世纪首个年代末，基于 Flash 芯片的固定状态储存设备开始普及，ext3 的效能已显稍逊，故最终发展出 ext4 文件系统。

Ext4 文件系统最高可支持 1 EiB 的卷，且最大单文件可达 16 TiB。新引入的 extent（区段）存储

方式对使用先进格式化（Advanced Format）技术的硬盘及尤其对 Flash 芯片等具有动辄 16 KiB 乃至 32 KiB 等较大基本储存区块的储存设备而言提升了存取性能。Ext4 文件系统在不启用 extent 功能时向下兼容 ext3，意味着在支持 ext3 的系统上可以直接以 ext3 的驱动程序进行挂载。新增的预留空间功能可以在文件系统内建立虚拟储存设备时免去写零占位的操作预留到充分的存储空间。预设情况下 ext4 文件系统的单个目录中最多可存放 64000 个子目录，与 ext3 相比翻了一倍。Ext4 还将可以不存有数据的区块进行标记，故在检查文件系统的时候这些不存有数据的区块将被略过，而使得在其他条件相同下可以更快速完成磁盘检查。在启用日志功能时，日志本身亦须计算校验和以确保文件系统日志的健壮性。在必要时，ext4 还可停用日志功能。

4.1.9 XFS 文件系统

XFS 是硅谷图形公司（亦称“视算电脑”，Silicon Graphics， Inc.）在 1993 为其 IRIX 操作系统开发的文件系统，发布于 1994 年。它支持不超过尺寸为 9 EB 的单个文件，目前允许的最大的单卷容量是 18 EB。

确定速率 I/O 是 XFS 文件系统支持的一项特性，是给应用程序提供了预留文件系统带宽的应用编程接口。XFS 会动态计算底层存储设备能提供的性能，并在给定的时间内预留足够的带宽以满足所要求的性能。

XFS 提供了一个文件系统碎片整理工具 xfs_fsr（XFS filesystem reorganize）。这个工具可以对一个已被挂载、正在使用中的 XFS 文件系统进行碎片整理。借助 xfs_growfs 工具，可以在线增加 XFS 文件系统的尺寸，但暂时还不能在线减小其尺寸，只能采取先备份再还原来实现减小文件系统所占用空间的操作。XFS 还可以将文件系统的日志放在非文件系统数据存储的其外置储存器上，例如可以将文件系统的日志放到固定状态驱动器（SSD）中以提升索引节点等小量数据的存取效率。

XFS 可以动态地增加 i-node，故不会出现在储存空间没用尽，反而先用尽预分配的 i-node 的情况。

ext4 文件系统不支持直接对目录所存储的内容尺寸进行限制，而 XFS 可以透过将目录作为项目（project）的方式来进行文件和目录配额（quota）；除此，XFS 还支持以用户（user）和用户（群）组（user group）的方式进行配额。

4.1.10 procfs

在 GNU/Linux 中，procfs 指的是进程文件系统（process file system）。它是个在系统启动时动态生成的伪文件系统，用于通过内核访问进程信息。这个文件系统通常被挂载到 /proc 目录。由于 /proc 只是个伪文件系统，它不占用存储空间，只占用有限的内存。

每个正在运行的进程都能对应上 /proc 中目录，目录名就是进程的进程标志号 PID，每个目录包含：

- /proc/PID/cmdline，启动该进程的命令
- /proc/PID/cwd，当前工作目录的符号链接
- /proc/PID/environ，影响进程的环境变量的名与值
- /proc/PID/exe，最初的可执行文件的符号链接
- /proc/PID/fd，包含每个打开的文件描述符的符号链接的目录
- /proc/PID/fdinfo，包含每个打开的文件描述符的位置和标记的目录

- /proc/PID/maps，一个包含内存映射的文件与块信息的文本文件
- /proc/PID/mem，一个表示进程的虚拟内存的二进制图像
- /proc/PID/root，该进程所能看到的根路径的符号链接。通常进程的根路径是“/”
- /proc/PID/status，包含了进程的基本信息，包括运行状态、内存使用

4.1.11　交换空间

GNU/Linux 中的交换是指内存页面被复制到预先设定好的硬盘空间的过程，目的是释放对于页面的内存。以上所述预先设定好的硬盘空间就叫作交换空间（swap）。物理内存和交换空间的总大小是可用的虚拟内存的总量。

交换空间通常是一个磁盘分区，但是也可以是一个文件。用户可以在安装系统的时候创建交换空间，或者在安装后的任何时间建立交换空间。对于 RAM 小于 1GB 的用户，交换空间通常是推荐的，但是对于拥有大尺寸实体内存的系统来说，交换空间不是必需的资源。

要检查当前交换空间的状态，可使用带参数的 swapon 或 free 命令。

命令如下：

```
[user@courseware 2529]$ swapon -s
文件名类型大小已用权限
/dev/dm-1                               partition       839676  300800  -2

[user@courseware 2529]$ free -m
total       used       free     shared  buff/cache   available
Mem:          990        426        250         14        313        363
Swap:         819        293        526
```

命令说明：

可见，目前在 /dev/dm-1 已经启用交换空间。

若要将一个设备作为交换空间使用，可以在 swapon 命令后跟上对应的设备文件。但若要系统在启动时自动挂载完成系统安装后新增的交换空间则需要在 /etc/fstab 中增加以下条目：

```
< 设备文件路径 > swap swap defaults 0 0
```

其中占位表示的 < 设备文件路径 > 既可以是一个分区：

```
/dev/mapper/centos_courseware-swap swap swap    defaults    0 0
```

也可以用分区的 UUID 来指明：

```
UUID=9e3817ae-cf00-4154-b445-76e88becbd00 swap swap    defaults     0 0
```

4.1.12 LVM

LVM（Logical Volume Manager）是一种可用在 GNU/Linux 的逻辑卷管理器软件，用于管理外部储存设备的储存空间划分。

LVM 利用 Linux 内核的 device-mapper 来实现存储系统的虚拟化（系统分区独立于底层硬件）。通过 LVM，可以实现存储空间的抽象化并在上面建立虚拟分区（virtual partitions），可以更简便地扩大和缩小分区，可以增删分区时无须担心某个硬盘上没有足够的连续空间，避免为正在使用的磁盘重新分区的麻烦。

LVM 的基本组成部分如下：

物理块（PE， physical extent）

物理块也称作实体块，是一个卷组中最小的连续区域，预设尺寸是 4 MiB，多个物理块将被分配给一个逻辑卷。可以把它看成物理卷的一部分，这部分可以被分配给逻辑卷。

物理卷（PV， physical volume）

物理卷也称作实体卷，是一个可供存储 LVM 的块设备，可以是一块硬盘、一个分区或一个回环文件。物理卷中部分空间会用于储存 LVM 相关的元数据。

卷组（VG， volume group）

卷组由一个或多个物理卷组合而成，作为存放逻辑卷的容器。卷组是从物理卷划分而来。

逻辑卷（LV， logical volume）

逻辑卷存放在一个卷组中并由物理块组成。是一个类似于物理设备的块设备，可以把它当作普通磁盘分区，直接在它上面创建文件系统。

PE.PV.VG 和 LV 之间的关系如图 4-2 所示。

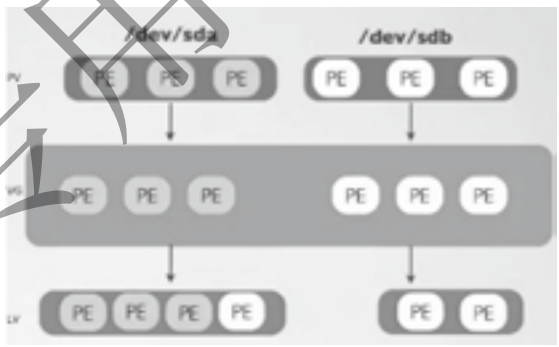

图 4-2 PE.PV.VG 和 LV 之间的关系

4.1.13 文件系统存在的形态

文件系统既可以存在于整个储存设备之中，如在 U 盘中；但更普遍的是存在于经过 GPT 或 MBR 分区表管理下的磁盘和磁盘阵列中：既可以在实体上寄于磁盘分区，也可以寄于逻辑卷之上。

在系统启动后，由引导系统启动的配置文件和文件系统相关配置文件共同决定系统启动后实际能使用的文件系统结构。相关的配置文件包括了通常挂载于 /boot 的 GRUB 的配置文件 /boot/efi/EFI/centos/grub.cfg 和 /etc/fstab。

4.2 一切皆文件

GNU/Linux 内的一切资源都可在逻辑上映射为文件。因此，对一切系统资源的调用在操作上均通过对文件进行操作而实现。

4.2.1　GNU/Linux 的文件系统目录结构

因 GNU/Linux 有诸多发行版本，为使得各发行版本的目录结构在基本使用上相同，Linux 基金会发布了文件系统层次结构标准（Filesystem Hierarchy Standard，FHS），该标准的最初版本发布于 1994 年，初期名为 FSSTND（Filesystem Standard）。该标准规定了系统中应有的一、二级目录及各应有目录的用途，定义了所需要的最小构成的文件和目录结构。大多 GNU/Linux 发行版都基本但不完全地遵循此标准。

GNU/Linux 的文件系统目录结构与 Unix 相若但有些许差异，其自根目录（/）作为开始。在同一个计算机系统上，整个文件系统在结构上形如树状。实际上，ext4 和 XFS 便是将文件的索引节点（i-node）以树这种数据结构来进行组织。

其主要包括的目录层级关系如表 4-3 所示。

表 4-3　目录层级关系

目录	描述
/	整个文件系统的根目录
/bin/	需要在单用户模式可用的必要命令（包含所对应的可执行文件，可以是 shell script 或二进制可执行文件）；其中多数亦是面向所有用户的命令，例如：cat、ls、cp
/boot/	引导相关的程序及配置文件，例如：efi 应用、grub2 引导程序、Linux 内核（通常被包含在名为 initrd 或 vmlinuz 的文件当中）等；通常是是一个单独的硬盘分区，在使用 EFI 的系统上，通常使用 FAT32 文件系统
/dev/	设备文件，例如：/dev/sda 和 /dev/null 等
/etc/	适用于特定（某一）主机系统范围内的配置文件。FHS 限制该目录只能存放静态配置文件，不能包含二进制文件。目前该目录的名称可以理解为”可编辑的文本配置”（Editable Text Configuration）
/etc/opt/	/opt/ 目录中装所装载的程序的配置文件
/etc/X11/	X 图形系统的配置文件
/home/	非 root 用户的家（或称“属主”）目录，包含对应各用户其自身保存的文件、个人设置等。一般用户对其家目录及其子目录内有完全的读、写和执行的权限
/lib/	/bin/ 和 /sbin/ 中二进制文件运行时所依赖的必要的库文件
/media/	该目录自 FHS 2.3 引入，其设计为可移除的外部储存设备的挂载点
/mnt/	临时挂载的文件系统，包含循环设备的文件系统
/opt/	额外的应用软件套件，例如 WPS 和 QQ 等第三方软件
/proc/	一个主要与系统中运行的进程相关的虚拟文件系统，其将内核与进程的状态用文本文件的形态来表示。其中含有 uptime、version 等文件
/root/	超级用户（super user，即 root）的家目录.
/sbin/	系统所需的可执行文件，例如： init、 ip、 mount。通常这里的命令是为多用户环境而设
/srv/	系统所提供的服务相关的具体数据，如 HTTP 服务的 WWW 目录便预设在其内
/tmp/	在系统重启后不必保留的临时文件
/usr/	主要用于存储只读的用户相关数据，包含大多数的多用户工具和应用程序，是“Unix Software Resource”的缩略。内含诸多二级目录
/usr/bin/	面向所有用户的非系统所必需的可执行文件
/usr/include/	C 和 C++ 的头文件
/usr/lib/	/usr/bin/ 和 /usr/sbin/ 中二进制文件的库文件
/usr/sbin/	非必须的系统相关的可执行文件文件，主要包含各种网络服务的守护（后台）进程
/usr/src/	发行版本相关的源码

续表

目录	描述
/usr/X11R6/	X 图形系统
/usr/local/	主要存储本机相关但与发行版关联不密切的文件。通常内含目录如：bin/、lib/、share/ 等
/var/	在正常运行的系统中其内容不断变化的变量文件，如日志、离线文件和临时的电子邮件文件
/var/cache/	应用程序缓存数据。这些数据通常可以其对应的应用关闭后删除
/var/lib/	存放程序在运行时维护的持久性数据，例如：数据库和应用的元数据等
/var/lock/	跟踪当前使用中资源的锁定状态。例如某个应用要独占使用某个块设备，可以由应用在此目录内进行记录，将该设备的占用转台标记为一个锁文件
/var/log/	内含各种系统服务的日志文件
/var/mail/	系统内用户的电子信箱
/var/run/	其功能现已经被 /run 代替。通常是指向 /run 的符号链接
/var/spool/	等待处理的任务的离线文件，如：打印队列、定时任务和未读的邮件等
/var/spool/cron/	存放系统定时任务
/var/spool/mail/	存放新收到的邮件
/var/tmp/	在系统重启过程中可予以保留的临时文件
/run/	最后一次启动以来，系统运行中只能单实例运行的程序及其相关资源可在此建立对应程序及资源的锁文件。如：当前登录的用户和一些服务的守护进程的进程标志（PID）文件、socket 文件等

在 CentOS 7 中，执行 file 命令加上具体的文件名可获知具体的文件类型。如下所示：当检查 /var/run 类型的时候可以看到，这实际是一个符号链接。

```
[user@courseware ~]$ file /var/run
/var/run: symbolic link to `../run'
```

若要给在 FHS 中所定义的目录作各自功能的定义，则可按照目录的内容来进行分类，如表 4-4 所示。

“可分享”（shareable）文件是可以存储在某一个系统上，并在其他系统上也同样可以存在的文件，通常对于同一个发行版的某个确定的版本而言，文件的内容会一致。“不可分享”（unshareable）文件是唯一对应于某一个确定的系统的文件，例如：用户属主目录中的文件是可分享的目录，而设备锁文件则不可分享。

“静态”（static）文件包括二进制可执行文件、库文件、文档文件等，在没有系统管理员干预的情况下不会更改文件；而“可变”（variable）文件则是在日常使用过程中常常会变化的文件，即使使用系统的用户并非系统管理员。

表 4-4　目录功能性质分类

	可分享（shareable）	不可分享（unshareable）
静态（static）	/usr	/etc
	/opt	/boot
可变（variable）	/var/mail	/var/run
	/var/spool/news	/var/lock

当下 FHS 3 的原文可访问 https：//refspecs.Linuxfoundation.org/fhs.shtml 查阅。

4.2.2　XDG 基本目录规范

Freedesktop.org（亦称 fd.o，最初名为 X Desktop Group）是维护 GNU/Linux 上的众多不同的 X 桌面环境，在各用户使用体验上尽可能相统一的一系列项目。其中，XDG 基本目录（Base Directory）规范定义在桌面环境中，每个用户都应该将其使用的应用数据存放在具有一定结构的目录中，并给出了具体目录路径的建议。每个具体定义的目录在用户登录后以环境变量的形式予以呈现。XDG 基本目录清单如表 4-5 所示。

表 4-5　XDG 基本目录清单

XDG 环境变量	含义	建议的预设值
XDG_DATA_HOME	存放当前用户专属的应用数据文件	$HOME/.local/share
XDG_CONFIG_HOME	存放当前用户专属的应用配置文件	$HOME/.config
XDG_DATA_DIRS	存放 X 系统数据文件的优选目录	/usr/local/share/:/usr/share/
XDG_CONFIG_DIRS	存放 X 系统配置文件的优选目录	/etc/xdg
XDG_CACHE_HOME	存放当前用户专属的应用缓存文件	$HOME/.cache
XDG_RUNTIME_DIR	存放系统运行时所需要的文件	

在 CentOS 7 中，当检查当前系统的环境变量时查询到具体的 $XDG_RUNTIME_DIR 目录的路径。

```
[user@courseware]$ export | grep XDG
declare -x XDG_DATA_DIRS=”/home/user/.local/share/flatpak/exports/share:/var/lib/flatpak/exports/share:/usr/local/share:/usr/share”
declare -x XDG_RUNTIME_DIR=”/run/user/1000”
declare -x XDG_SESSION_ID=”343”
```

其中，XDG_RUNTIME_DIR 如果设定具体的目录，则必须将该目录放在所在系统的文件系统中。在用户退出登录后将被清空。

然而，XDG 基本目录规范并非一个强制实施在所有发行版上的规范，目前就 CentOS 7 而言，并未完全遵循之。

4.2.3　一切皆文件

“一切皆文件（everything is a file）”，或更准确地说“一切皆文件描述符（everything is a file）”是 Unix 及其分支（包含 GNU/Linux）的典型特性：将一切系统内的资源，包括储存设备上的文件及目录、系统的输入和输出设备，乃至进程和网络通信资源都在虚拟文件系统空间内用字节流进行表示。如此，系统中的所有资源都可以用唯一的路径和文件名进行定位，能以读写文件的应用编程接口（application programming interface）进行访问，并能用管理文件和目录的方法进行创建、文件名和路径的变更、属性的变更乃至删除的操作。

典型的虚拟文件系统便是 /proc，其中布满进程相关信息的文件，但这些文件并非真实存在于外部储

存设备当中。在 GNU/Linux 当中，一个称作 sysfs 的虚拟文件系统亦挂在 /proc 当中，该文件系统提供了一些系统运行时相关的信息。其中通常包含以下文件：

/proc/crypto：可利用的加密模块列表。

/proc/devices：字符设备与块设备列表，按照设备索引排序。

/proc/diskstats：给出了每一块逻辑磁盘设备的一些信息。

/proc/filesystems：当前内核支持的文件系统的列表。

/proc/interrupts、/proc/iomem、/proc/ioports 和 /proc/irq 设备的一些与中断、内存访问有关的信息。

/proc/kmsg：内核输出的一些信息。

/proc/meminfo：包含内核管理内存的一些汇总信息。

/proc/modules：包含了当前加载的内核模块列表。

/proc/mounts：包含了当前安装设备及安装点的符号链接。

/proc/net/：一个目录包含了当前网络栈的信息。

/proc/partitions：一个设备号、尺寸与 /dev 名的列表，内核用于辨别已存在的储存设备分区。

/proc/scsi：给出任何通过 SCSI 或 RAID 控制器挂接的设备的信息。

/proc/self：同 /proc/PID/，其中 PID 是当前进程的进程标志号。是当前进程的符号链接。

/proc/slabinfo：内核频繁使用的对象的统计信息。

/proc/swaps：活动交换分区的信息，如尺寸、优先级等。

/proc/sys：动态可配置的内核选项。

/proc/sysvipc：包括共享内存与进程间通信信息。

/proc/tty：包含当前终端信息。

/proc/uptime：内核启动后经过的秒数。

/proc/version：含内核版本、发布号（distribution number）、编译内核的编译器版本和其他相关的版本信息。

4.2.4 主流发行版之差异

要说不同发行版最主要的还是软件包管理和预设两方面。

由于不同的软件仓库及其管理机制和软件包适配的原因，使得就算是同一个软件在不同发行版上预设的配置文件路径亦有可能不同。典型以 MySQL 为例：在 CentOS 上预设生效的配置文件是 /etc/my.cnf。而在 Debian 上预设生效的配置文件是 /etc/mysql/my.cnf。在进行软件配置时应认识到这个差异的存在，并在变更配置不生效时应考虑到有存在配置文件路径相关问题的可能。

但就文件系统上来说，目录结构自身和一些特定的文件，如系统自身版本说明文件所在的路径也是有所差异的地方。

就核对发行版的版本而言，相对通用一点的办法有多种：一是检查 /etc/os-release：

```
user@ltt-llk2:/mnt/c/Users/devhood/ackt/tutorial-gnu_Linux-app-basic/src/ch4$cat /etc/os-release
NAME=”Ubuntu”
```

```
VERSION=” 18.04.4 LTS (Bionic Beaver)”
ID=ubuntu
ID_LIKE=debian
PRETTY_NAME=” Ubuntu 18.04.4 LTS”
VERSION_ID=” 18.04”
HOME_URL=” https://www.ubuntu.com/”
SUPPORT_URL=” https://help.ubuntu.com/”
BUG_REPORT_URL=” https://bugs.launchpad.net/ubuntu/”
PRIVACY_POLICY_URL=” https://www.ubuntu.com/legal/terms-and-policies/privacy-policy”
VERSION_CODENAME=bionic
UBUNTU_CODENAME=bionic
```

二是检查 /etc/issue：

```
user@ltt-llk2:/mnt/c/Users/devhood/ackt/tutorial-gnu_Linux-app-basic/src/ch4$ cat /etc/issue
Ubuntu 18.04.4 LTS \n \l
```

三是可以通过 uname -a 来查看：

```
[user@courseware ~]$ uname -a
Linux courseware.local 3.10.0-957.el7.x86_64 #1 SMP Thu Nov 8 23:39:32 UTC 2018 x86_64 x86_64 x86_64 GNU/Linux
```

但就不同系列的发行版本而言，都还有其特定的方法可以查询。

RHEL 系列

以 CentOS 7 为例，要查明当前系统的版本，可以检查下面三个文件中的任意一个。

- /etc/centos-release
- /etc/redhat-release
- /etc/system-release

如：

```
[user@courseware ~]$ cat /etc/redhat-release
CentOS Linux release 7.6.1810 (Core)
[user@courseware ~]$ cat /etc/system-release
CentOS Linux release 7.6.1810 (Core)
[user@courseware ~]$ cat /etc/centos-release
CentOS Linux release 7.6.1810 (Core)
```

- Debian 系列

以 Ubuntu 为例，要查明当前系统的版本，可以检查以下的文件：

```
user@ltt-llk2:/mnt/c/Users/devhood/ackt/tutorial-gnu_Linux-app-basic/src/ch4$ cat /etc/l
sb-release
DISTRIB_ID=Ubuntu
DISTRIB_RELEASE=18.04
DISTRIB_CODENAME=bionic
DISTRIB_DESCRIPTION=”Ubuntu 18.04.4 LTS”
```

4.2.5 符号链接

符号链接（symbolic link）亦称软链接（soft link），是一类相对特殊的文件，它在实现上能指向一个文件或目录，这个所指向的文件或目录可称为“目标文件”。在使用上，操作符号链接，就相当于操作它所指向的目标文件。

符号链接是一个独立文件，占用至少一个文件系统索引节点，其存在并不依赖于目标文件。如果删除一个符号链接，它指向的目标文件不受影响。但若目标文件经过了移动、重命名或者删除这样的操作，即使符号链接仍存在文件系统上，没有跟着变更所指向的目标文件的符号链接将指向一个不存在的文件，此时相当于符号链接失效了。

使用上，见如下例：

```
[user@courseware files]$ ls cmd*
cmd
[user@courseware files]$ cat cmd
date
[user@courseware files]$ ln -s cmd cmd-the-same
[user@courseware files]$ cat cmd-the-same
date
[user@courseware files]$ file cmd-the-same
cmd-the-same: symbolic link to `cmd’
[user@courseware files]$ mv cmd cmd-backup
[user@courseware files]$ cat cmd-the-same
cat: cmd-the-same: 没有那个文件或目录
```

首先，在当前目录中存在一个名为 cmd 的文件，而后使用 ln 命令并加上 -s 参数即可建立名为 cmd-the-same 的符号链接。而后，输出 cmd-the-same 的内容可知与原 cmd 文件的内容如出一辙。并用 file 检查其文件类型后可知，实际 cmd-the-same 文件只是个符号链接。最后，对 cmd 文件进行文件名变更操作后再检查 cmd-the-same 可见其内容已不可访问，因其目标文件已不存在。

再如，建立指向目录的符号链接，在使用上与直接操作目录相同，但用 file 检查可明确实际还是个符号链接：

```
[user@courseware symlnk]$ ls /var
account  cache  db    ftp    gopher    lib    lock  mail  opt      run    target  yp
```

```
adm     crash empty games kerberos local log  nis  preserve spool tmp
[user@courseware symlnk]$ ln -s /var v && ls v
account cache db    ftp   gopher   lib   lock mail opt     run   target yp
adm     crash empty games kerberos local log  nis  preserve spool tmp
[user@courseware symlnk]$ file v
v: symbolic link to `/var'
[user@courseware symlnk]$ echo "T" > /var/tmp/demo-text
[user@courseware symlnk]$ cat v/tmp/demo-text
T
```

4.2.6　设备的符号链接

设备的符号链接的概念通常是在为使用户使用上更便利而设的文件。以下面的终端交互为例：

```
[user@courseware ~]$ ls -l /sys/block/sd*
lrwxrwxrwx. 1 root root 0 2 月  28 20:57 /sys/block/sda -> ../devices/
pci0000:00/0000:00:1f.1/ata1/host0/target0:0:1/0:0:1:0/block/sda
lrwxrwxrwx. 1 root root 0 2 月  28 20:57 /sys/block/sdb -> ../devices/
pci0000:00/0000:00:1f.1/ata2/host1/target1:0:0/1:0:0:0/block/sdb
lrwxrwxrwx. 1 root root 0 2 月  28 20:57 /sys/block/sdc -> ../devices/
pci0000:00/0000:00:1f.1/ata2/host1/target1:0:1/1:0:1:0/block/sdc
lrwxrwxrwx. 1 root root 0 2 月  28 20:57 /sys/block/sdd -> ../devices/
pci0000:00/0000:00:1f.2/ata3/host2/target2:0:0/2:0:0:0/block/sdd
[user@courseware ~]$ file /sys/block/sda
/sys/block/sda: symbolic link to `../devices/pci0000:00/0000:00:1f.1/ata1/host0/tar-
get0:0:1/0:0:1:0/block/sda'
```

在挂在于 /sys 的虚拟文件系统 sysfs 中，/sys/block 存在四个以“sd”开头的文件，而这些文件实际上是指向 /sys/devices 中具体块设备文件的符号链接。而从 file 命令的结果可知，该符号链接所指向的实际设备路径通常对用户而言并不友好。在实际使用中，用户直接对系统自动产生的设备符号链接来进行访问会更方便。

4.2.7　硬链接

实际上，在多数的文件系统（除了 VFAT 外）中，GNU/Linux 在处理储存在外部储存设备上的文件时，都是在文件系统中将为每个文件分配一个确切的索引节点（I node），硬链接在链接到具体文件的时候是直接指向文件的索引节点，故在当遇到被指向的目标文件更名之时，不会产生如使用符号链接（软连接）时所产生的目标文件失效的情况（索引节点、硬链接与软链接的关系如图 4-3 所示）。

图 4-3　索引节点、硬链接与软链接的关系

例：

```
[user@courseware files]$ ln cmd-backup cmd2
[user@courseware files]$ file && cat cmd2
[user@courseware files]$ file cmd2 && cat cmd2
cmd2: ASCII text
Date
[user@courseware files]$ mv cmd-backup cmd-backup2
[user@courseware files]$ cat cmd2
date
[user@courseware files]$ ls -il cmd-backup2 cmd2
3137987 -rw-rw-r--. 2 user user 5 3 月  2 15:13 cmd2
3137987 -rw-rw-r--. 2 user user 5 3 月  2 15:13 cmd-backup2
```

通过 ln 命令建立指向 cmd-backup 文件的索引节点的硬链接 cmd，用 file 命令检查时会被认为是一个有具体内容的文件，并且。在使用 ll 命令检查时可见其第三列的硬链接数码是 2，意味着同一个索引节点被硬链接了两次，且第一列所指明的索引节点号都是 3137987。

当一个文件所对应的索引节点的硬链接次数变更为零时，即在文件系统上删除该文件。

4.2.8 管道命令符

管道命令符“|”对应的按键通常在主键盘区右上角，其作用是将在管道命令符前的命令执行完毕后的结果，作为该命令符后的命令所需的标准输入数据并传入。故应用到管道命令符的情形通常会涉及两条或更多的命令。

以下面的命令为例：

```
cat /proc/cpuinfo | less
```

若单独执行该命令的管道命令符前方的“cat /proc/cpuinfo”能在终端中输出当前系统的处理器信息。然而通常处理器相关完整信息篇幅都有几十行，一个屏幕看不完整，在此可将处理器信息的结果传递给 less 命令来进行更方便地查看。在键入完整的上述命令后，可见屏幕会出现底部多一行以冒号开头的空行的全屏文字界面：

```
processor       : 0
vendor_id       : GenuineIntel
cpu family      : 6
model           : 158
model name      : Intel(R) Core(TM) i5-7300HQ CPU @ 2.50GHz
stepping        : 9
cpu MHz         : 2495.998
cache size      : 6144 KB
```

```
physical id     : 0
siblings        : 2
core id         : 0
cpu cores       : 2
apicid          : 0
initial apicid  : 0
fpu             : yes
:
```

在这个界面中可方便地利用上下方向键等浏览相关键盘按键进行滚动，并且可以通过键入“/”进行文本查找，并结合“n”和“N”等 less 内建的功能键进行往回式搜索（更多关于 less 的使用可以在终端键入 man less 查看手册）。这在检查配置文件的时候十分实用。

再有，若要查询文本文件的内容统计信息，如行数，亦可利用 wc（意为 word counter）命令结合管道命令符，如：

```
[user@courseware ~]$ cat /proc/cpuinfo | wc -l
50
```

可知完整的处理器信息有 50 行。再有，日常系统运维过程中，去筛查进程时，可结合正则表达式工具 grep 和管道命令符来缩小排查范围。例如，要查询进程名称中包含 init 字样的进程可以这样操作：

```
[user@courseware ~]$ ps aux | grep init
root     3036  0.0  0.0  16892   176 ?        SNs  2月28   0:00 /usr/sbin/alsactl -s -n 19 -c
-E ALSA_CONFIG_PATH=/etc/alsa/alsactl.conf --initfile=/lib/alsa/init/00main rdaemon
user  5188  0.0  0.0 112728   988 pts/0    S+   17:48   0:00 grep --color=auto init
```

以上例子都是在一个命令中只有一个管道命令符的应用情形。而管道命令符不止可以在一次执行的命令中含有一个，还可以含有多个。管道命令符链式应用与数学中复合函数求导数的链式法则在形式上十分类似：都是将对上一级处理完的结果传递给后一级处理。不过，在管道命令符链式应用所传递的是命令运行后的结果，通过标准输出与标准输入就管道命令符两侧的两个命令进行关联。如：

```
[user@courseware ~]$ cat /proc/modules | grep xfs | tee fs-xfs-module.log
xfs 997127 2 - Live 0xffffffffc04c7000
libcrc32c 12644 3 nf_nat,nf_conntrack,xfs, Live 0xffffffffc0380000
[user@courseware ~]$ cat fs-xfs-module.log
xfs 997127 2 - Live 0xffffffffc04c7000
libcrc32c 12644 3 nf_nat,nf_conntrack,xfs, Live 0xffffffffc0380000
```

以上命令在列出当前系统中所装载的模块中筛选出或与 XFS 文件相关的行，并将结果传递给 tee 命令。tee 命令在接收到 grep 通过管道传递过来的结果作为输入参数后，将结果显示在屏幕之余，亦将结果储存在 fs-xfs-module.log 文件当中。

4.2.9 特别的文件

字符文件与块文件合称为设备文件。

设备文件在 GNU/Linux 中是设备驱动程序的接口，像常规文件般存在文件系统中。

设备文件允许应用程序使用设备驱动程序，通过标准输入输出系统调用与驱动程序交互，从而简化了日常对设备进行的操作方式，如调用、读取数据、写入数据和释放。

设备文件可以映射自一个完整的设备，如打印机；也可以是设备的某些特定的资源，如硬盘分区；还可以是由软件虚拟的系统资源，如随机数生成器。

在 GNU/Linux 中包含两类设备节点，分别为字符文件与块文件。它们的区别是系统向它们读写数据的方式。

字符设备是指每次与系统传输一个字符的设备。这些设备节点通常为虚拟终端、串口和调制解调器之类的外部设备提供流通信服务，通常不支持随机存取数据。在实现时，大多不使用缓存器，直接对相应的设备读取或写入每一个字符。

块设备是指与系统间用块的方式传输数据的设备。这些设备节点通常代表可寻址、随机存取的设备，如硬盘、光碟和内存区域。块设备通常使用缓存器。系统为输入、输出分配了缓存，以存储一块数据，当程序向设备发送了读取或写入数据的请求时，系统把数据中的每一个字符存储在缓存中。当缓存被填满时，会采取相应的操作以将数据传输到计划的目的设备，而后清空缓存。

使用 lsblk 命令可以列出当前系统中的块设备：

```
[user@courseware ~]$ lsblk
NAME                 MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
sda                  8:0    0    8G  0 disk
├─sda1                  8:1    0  200M  0 part /boot/efi
├─sda2                  8:2    0    1G  0 part /boot
└─sda3                  8:3    0  6.8G  0 part
  ├─centos_courseware-root 253:0    0    6G  0 lvm  /
  └─centos_courseware-swap 253:1    0  820M  0 lvm  [SWAP]
sdb                  8:16   0    8G  0 disk /mnt/vfat
sdc                  8:32   0    8G  0 disk
sdd                  8:48   0    8G  0 disk
sr0                 11:0    1 1024M  0 rom
```

设备节点一般在 /dev 目录下，通常使用如表 4-6 所示的前缀：

表 4-6 设备前缀

fd	软碟（floppy），现较少见
hd	PATA 储存设备
lp	打印机（printer）
par	并口
pt	伪终端

续表

fd	软碟（floppy），现较少见
sd	SCSI 和 SATA 储存设备
tty	终端，原意系电传打字机（teletype）
ttyS	串口

多数设备的前缀名后面紧跟一个顺序数码，用以唯一指定某一特定的设备。硬盘的前缀名后面跟随一个字母，并且硬盘分区会在硬盘机名称后再接上一个顺序数码，即：字母用于指明设备，而数字用于指明设备上的分区。因此，/dev/sdb2 可能使某一块硬盘上的某一个分区，/dev/pts/0 可能是一个网络终端会话。

如：通过 PuTTY 连接到机器上，可以通过访问 /dev/pts/0 来向当前终端输出文本：

```
[user@courseware ~]$ ls /dev/pts/
0  ptmx
[user@courseware ~]$ file /dev/pts/0
/dev/pts/0: character special
[user@courseware ~]$ echo text > /dev/pts/0
text
```

4.2.10　伪设备

在 GNU/Linux 中，设备节点并不一定要对应物理设备。没有这种对应关系的设备就是伪设备，其提供了多种部分系统管理所用到的功能。部分常见的伪设备如表 4-7 所示。

表 4-7　伪设备

/dev/null	接受并丢弃所有输入。通常在重定向中使用，使得执行的结果不被输出
/dev/full	永远在被填满状态的设备
/dev/loop	循环（loop）设备
/dev/zero	产生连续的 NUL 字元（每个字节都为 0x00）的数据流
/dev/random	产生一个伪随机的任意长度数据流
/dev/urandom	以非阻塞的方式产生一个伪随机的任意长度数据流

4.3　重定向

改变数据输入或输出的方向，是信息处理自动化的基本操作。

4.3.1　文件描述符

在介绍重定向之前有必要先了解文件描述符（File descriptor）的概念。

文件描述符是用于表述指向文件的引用。

用 C 语言编程中的概念作类比，如果系统中的文件就像是一个存放在内存中的变量，那在使用上来说，该文件的文件描述符就相当于指向上述变量值的变量名。

文件描述符在形式上是一个索引值，故必然地也是非负整数。它指向内核为每一个进程所维护的该进程打开文件的记录表。当程序打开一个文件时，无论文件是本已存在还是新建立的，系统内核都向进程返回一个文件描述符。通常，每个 GNU/Linux 上运行的进程均有三个标准的 POSIX 文件描述符，对应于三个标准化的数据流，如表 4-8 所示。

表 4-8　文件描述符

索引值	名称	<unistd.h> 常量	<stdio.h> 常量	说明
0	标准输入（standard input）	STDIN_FILENO	stdin	预设是键盘，也可从是其他文件或命令的执行结果
1	标准输出（standard output）	STDOUT_FILENO	stdout	预设是终端（TTY）
2	标准错误（standard error）	STDERR_FILENO	stderr	预设是终端（TTY）

4.3.2　输入重定向

输入重定向中用到的操作符及其作用如表 4-9 所示。

表 4-9　输入重定向中用到的操作符及其作用

操作符	作用
命令 < 文件名	将文件名所对应的文件的内容作为命令的标准输入
命令 << 结束标记	从标准输入中读入，直到遇见结束标记（End Of File）。EOF 结束标记可通过 Ctrl + Z 的组合按键进行输入

例，将文件的内容作为标准输入传递给 cat 命令：

```
user@ltt-llk2:/mnt/c/Users/devhood/ackt/tutorial-gnu_Linux-app-basic/src/ch4$ cat < cantonese-intro.txt
Cantonese is a dialect of Chinese.
```

4.3.3　输出重定向

输出重定向中用到的操作符及其作用如表 4-10 所示。

表 4-10　输出重定向中用到的操作符及其作用

操作符	作用
命令 > 文件	将标准输出重定向到文件中。若文件原本有内容，则原内容会被覆盖（覆写）
命令 2> 文件	将错误输出重定向到文件中。若文件原本有内容，则原内容会被覆盖（覆写）
命令 >> 文件	将标准输出重定向到文件中。若文件原本有内容，则重定向的内容追加在原本内容的后面
命令 2>> 文件	将错误输出重定向到文件中。若文件原本有内容，则重定向的内容追加在原本内容的后面
命令 &>> 文件	等用于【命令 >> 文件 2>&1】。将标准输出和错误输出都写入文件。若文件原本有内容，则重定向的内容追加在原本内容的后面

相对而言，输出重定向比输入重定向更常用。

以下用一个在 CentOS 中存在的命令 ifconfig 和一个不存在的命令 ipconfig 为例做演示。当执行不存在的命令时，Shell 预设会把错误输出到标准错误输出设备上。直接分别执行以上两命令预设的终端输出如下：

```
[user@courseware tmp]$ ifconfig
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
inet 10.0.2.15  netmask 255.255.255.0  broadcast 10.0.2.255
inet6 fe80::703f:b956:a3b7:514f  prefixlen 64  scopeid 0x20<link>
ether 08:00:27:b3:70:d9  txqueuelen 1000  (Ethernet)
RX packets 755387  bytes 924716579 (881.8 MiB)
RX errors 0  dropped 0  overruns 0  frame 0
TX packets 159959  bytes 11338666 (10.8 MiB)
TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
inet 127.0.0.1  netmask 255.0.0.0
inet6 ::1  prefixlen 128  scopeid 0x10<host>
loop  txqueuelen 1000  (Local Loopback)
RX packets 0  bytes 0 (0.0 B)
RX errors 0  dropped 0  overruns 0  frame 0
TX packets 0  bytes 0 (0.0 B)
TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
virbr0: flags=4099<UP,BROADCAST,MULTICAST>  mtu 1500
inet 192.168.122.1  netmask 255.255.255.0  broadcast 192.168.122.255
ether 52:54:00:59:b1:5b  txqueuelen 1000  (Ethernet)
RX packets 0  bytes 0 (0.0 B)
RX errors 0  dropped 0  overruns 0  frame 0
TX packets 0  bytes 0 (0.0 B)
TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
[user@courseware tmp]$ ipconfig
bash: ipconfig: 未找到命令 ...
```

当在构建自动化运维工具时，通常需要把相关的错误信息保存下来，最通常的做法便是将错误输出存到日志文件中去。以下为例，即可将执行 ipconfig 时的错误信息保存到日志文件 runtime.log 中。

```
[user@courseware files]$ ipconfig 2> runtime.log
[user@courseware files]$ cat runtime.log
bash: ipconfig: 未找到命令 ...
```

而在使用输出重定向时，观察下例：

```
user@courseware files]$ cat runtime.log > example.txt
[user@courseware files]$ cat runtime.log > example.txt
[user@courseware files]$ cat example.txt
bash: ipconfig: 未找到命令 ...
[user@courseware files]$ cat runtime.log >> example.txt
[user@courseware files]$ cat runtime.log >> example.txt
[user@courseware files]$ cat example.txt
bash: ipconfig: 未找到命令 ...
bash: ipconfig: 未找到命令 ...
bash: ipconfig: 未找到命令 ...
```

可知，在使用“>”进行覆写时，无论此种输出重定向被执行多少次，所被输出的文件的内容会被最后一次所输出的内容覆盖。而使用“>>”进行追加时，内容会依次追加到日志文件的末尾。

事实上，在记录错误日志的文件中，我们可以使用追加的方式将标准错误输出的内容重定向到指定文件，也可以结合 date 命令记录发生错误的时刻。

4.4 文件的权限

权限控制，是保障信息安全、防止未经授权的误操作的基本手段。

GNU/Linux 中的文件权限模型中，文件的权限分为三个不同的层级，分别是：文件属主、群组和非文件所属群组的其他用户（下亦称作“其他人”）而言的概念。

4.4.1 文件用户的层级

1. 属主

文件属主（owner）在概念上即是文件的所有人，汉语亦有译称拥有者、物主和所有者等，预设情况下等同于这个文件的建立者。文件的属主可以使用 chown（意为 change owner）命令变更文件的属主为其他用户。

2. 群组

每个用户都会属于至少一个用户群组，简称用户组。一个文件的属主所在的群组中，除属主以外的所有用户都会对该文件具有该层级所指明的文件使用权限。

3. 其他人

其他人（other）即为属主和属主所在群组以外的其他所有用户，对每一个具体的文件而言，其他人也有对应一个层级的访问权限。

4.4.2　文件的权限

对每个文件的三个层级的访问权限而言，每一层都有至少包含读（read）、写（write）和执行（execute）的权限。

以下通过在终端中对 /bin/tar 文件执行 file 和 ls –l 命令为例进行讲解。

```
[user@courseware files]$ file /bin/tar
/bin/tar: ELF 64-bit LSB executable, x86-64, version 1 (SYSV), dynamically linked (uses shared libs), for GNU/Linux 2.6.32, BuildID[sha1]=76d3a7dafbd5404630d64e7fb43a38922f680fdc, stripped
```

以上输出指明该文件是 64 位的可执行文件。对于可执行文件，主要留意其对三个层级的文件用户而言是否具有可执行的权限。

```
[user@courseware files]$ ls -l /bin/tar
-rwxr-xr-x . 1  root  root  346136 10 月 31 2018 /bin/tar
【1】  【2】【3】【4】【5】  【6】  【7】
```

执行 ls 命令加上 –l 参数可列出所查询的文件和目录的详细信息。关于 ls 更详细的用法可以通过在终端输入 man ls 来查询，在此主要阐述其输出中各部分的含义。

第一部分“–rwxr–xr–x”指明了这个文件的类型和三个层级的用户的访问权限：

最开头所标记的“–”指明了这是一个普通文件。下表是这个位置可能出现的标志：

- 普通文件
- d 目录
- l 链接文件
- b 块特殊设备
- c 特殊设备文件
- s 套接字文件
- p 管道文件

由此可见，在 GNU/Linux 中目录（或称“文件夹”）实际上也是一种特殊的文件。

第一部分紧接下来又可分为每三个字母一组，分别对应属主、群组和其他人的文件访问权限，一共三组（九个字母）。

以上例子中的“rwxr–xr–x”开头的“rwx”属主具有可读、可写且可执行的权限，其中，读（r）、写（w）和执行（x）的标志所占的位置是有序的。当对应的文件用户不具备对应某项权限时，该标志符会被置为“–”。

以此类推，后两组中的“r–x”意为该文件对群组和其他人均可读、不可写但可执行。

以上用 rwx 三个字母标记文件权限的方式可称为文件权限的字符方式。除了字符方式以外，文件权限还可以用八进制数字标记的数字方式来进行表示。关于字符方式与数字方式的对应关系如表 4–11 所示。

4-11　字符与数字的对应关系

字符	数字
rwx	7
rw-	6
r-x	5
r--	4
-wx	3
-w-	2
--x	1
---	0

从上表可知，将以上数码转写成二进制时，二进制的各个数位上的值实际上相当于各个层级的文件用户的三种（读、写和执行）权限的有无。

在以上例子中，第二部分的数码 1 表示该文件所对应的 inode 的硬链接数目，意即该文件的硬链接数为 1。

第三部分和第四部分依次表示文件属主和文件所属的用户群组。

第五部分是以字节为单位表示的文件尺寸。

第六部分表示的是文件的最后修改时间。

第七部分表示的是文件路径及其文件名。

4.4.3　图形化管理

在安装了 GNOME 桌面环境的 GNU/Linux 系统中，还可以通过快捷按键组合 alt + F2 开启的对话方块中输入 nautilus 命令开启文件管理应用来查看和在具有权限的前提下变更文件权限。以下图中的文件（文件名为 no-text.txt）为例，在 GNOME 桌面环境中，对文件右键点选【属性】，并点选【权限】标签后可以图形化的方式检查文件权限的情况，且因此文件的所有者是当前登录的用户自身，故可直接在此界面以图形化的方式更改文件权限。

图 4-4　文件的权限

4.4.4　文件权限的变更

可以使用 chmod（意思为 change mode）命令去改变文件（包括目录）的权限（如图 4-4 所示），设置方式可以使用字符或数字两种方式。该命令详细的使用方法可在终端键入 man chmod 来查看。在此，简要介绍该命令的主要使用方式：

```
chmod [options] mode file…
```

其中，[options] 是可选参数，可不传入，但权限模式 mode 和所作用到的文件 file 两个参数是必传项。

其中，mode 可以用 u、g、o 和 a 四个字符来进行权限层级的指定。

• u：User，即文件的属主。

• g：Group，即文件所属群组。

• o：Other，其他人。

• a：All，所有用户，相当于同时指定 u、g 和 o。

• 在指定了权限层级外，还应指定所期望需变更的文件（读、写和执行）权限。在这里同样可以用字符方式，主要使用到的字符及其含义如下：

• r：读取权限。

• w：写入权限。

• x：执行权限。

• -：不具备任何权限之含义。

例：

```
[user@courseware files]$ ll script-test.sh
-rw-rw-r--. 1 user user 0 3月  1 15:18 script-test.sh
[user@courseware files]$ chmod ugo+x script-test.sh
[user@courseware files]$ ll script-test.sh
-rwxrwxr-x. 1 user user 0 3月  1 15:18 script-test.sh
```

以上三个命令中，第一个命令查询 script-test.sh 的具体信息，包括其权限信息；第二个命令就该文件对其增加属主、群组和其他人的可执行权限；第三个命令的作用同第一个命令，可见执行第三条命令时系统上的所有用户皆可执行该文件。

从以上例子可见，增加权限可用“+”符号。另，当要减少文件权限可使用“-”符号。

4.4.5　直接指定文件的权限

变更文件的模式（mode）也可以用数字方式指定所需变更后的文件权限。例：

```
[user@courseware files]$ cd /tmp
[user@courseware tmp]$ mkdir courseware-tmp
[user@courseware tmp]$ ls -al courseware-tmp
总用量 4
drwxrwxr-x.  2 user user    6 3月  1 17:38 .
drwxrwxrwt. 17 root root 4096 3月  1 17:39 ..
[user@courseware tmp]$ chmod 777 courseware-tmp
[user@courseware tmp]$ ls -al courseware-tmp
总用量 4
drwxrwxrwx.  2 user user    6 3月  1 17:38 .
drwxrwxrwt. 17 root root 4096 3月  1 17:39 ..
```

在这个例子中，刚建立的 courseware-tmp 目录本身对其他人并无写入权限，而在变更权限后任何

人都可以写入该目录。

另外，使用“=”符号，也可以指明想要指定的文件权限：例：

```
[user@courseware tmp]$ chmod o=r courseware-tmp
[user@courseware tmp]$ ls -al courseware-tmp
总用量 4
drwxrwxr--. 2 user user  6 3 月  1 17:38 .
drwxrwxrwt. 17 root root 4096 3 月  1 17:53 ..
```

上例中可见，其他人对 courseware-tmp 目录皆没有写入和执行的权限。

4.5 文件的属性

上文介绍过，在 GNU/Linux 中，文件类型不是像 Windows 那般通过文件名的后缀名来分辨，可以通过 file 命令来检查确切的文件类型。这是因为除了基于文本的文件类型以外，多数的专有文件格式的文件都以一些特征的编码组合作为开头。如：GIF 图像文件的文件头便以“GIF8”开头。

在 CentOS 7 上，用于辨识文件类型的数据存在于 /usr/share/magic。

在有图形环境的系统中，通常对单一文件来说，通过文件管理器直接查看文件属性（如图 4-5 所示）是最便利的操作，如同在 Windows 桌面环境一般，可以直观地看到文件类型，文件的尺寸大小，乃至访问和最后修改时间等信息。

图 4-5 文件的属性

在终端中，常用 ls 命令来检查目录中的文件。在使用 ls 命令时，加上参数 -l 可以列表的形式来查看文件清单：

```
[user@courseware files]$ ls -l /dev/sd*
brw-rw----. 1 root disk 8,  0 2 月  28 20:57 /dev/sda
brw-rw----. 1 root disk 8, 16 2 月  28 20:57 /dev/sdb
brw-rw----. 1 root disk 8, 32 2 月  28 20:57 /dev/sdc
brw-rw----. 1 root disk 8, 48 2 月  28 20:57 /dev/sdd
brw-rw----. 1 root disk 8, 49 2 月  28 20:57 /dev/sdd1
brw-rw----. 1 root disk 8, 50 2 月  28 20:57 /dev/sdd2
brw-rw----. 1 root disk 8, 51 2 月  28 20:57 /dev/sdd3
```

在 CentOS 7 上，还可以使用更顺手的 ll 来达到相同的效果。因使用 type 命令检查 ll 可知，ll 实际上是个带上参数的 ls 命令别名：

[user@courseware files]$ type ll

ll 是‘ls -l --color=auto’的别名

而 ls 其实还有如下常用的参数：

-C 多列输出，纵向排序。

-F 每个目录名加“/”后缀，每个可运行文件名加“*”后缀。

-R 递归列出遇到的子目录。

-a 列出所有文件，包括以“.”开头的隐含文件。

-c 使用“状态改变时间”代替“文件修改时间”为依据来排序或列出。

-d 将目录名像其他文件一样列出，而不是列出它们的内容。

-i 输出文件前先输出文件索引节点号

-q 将文件名中的非打印字符输出为问号。

-r 逆序排列。

-t 按时间排序。

-u 使用最近访问时间代替最近修改时间为依据来排序或列出。

-1 单列输出。

预设列出文件时所显示的文件时刻是最后修改时刻。要查看最后的存取时间，可将参数 --time 指定为 atime；若要查看文件的最后只变更属性等文件元数据的时刻，则可指定为 ctime。以下为例：

```
[user@courseware files]$ ll script-test
-rwxrwxr-x. 1 user user 20 3 月   5 01:13 script-test
[user@courseware files]$ ll --time=ctime  script-test
-rwxrwxr-x. 1 user user 20 3 月   5 01:32 script-test
[user@courseware files]$ ll --time=atime  script-test
-rwxrwxr-x. 1 user user 20 3 月   5 01:33 script-test
```

可见，对同一个文件来说，这三个时刻可以有不同的值。

在 GNU/Linux 中，目录也是种特殊的文件。故以上介绍有关文件属性的操作通常也适合于目录。

本章小结

本章主要学习了 Linux 的文件系统管理的基础知识，包括文件系统的基础知识，常见文件系统的概述，“一切皆文件”的文件系统逻辑结构，文件描述符与输入输出重定向，文件权限的设立与变更，文件的属性等内容。

习题

一、简答题

1. 列举不少于两种主要用于 GNU/Linux 的日志型文件系统。

2. 如何确定当前发行版的版本？

3. ~/h.txt 硬链接到文本文件 ~/a.txt，~/s.txt 软链接到 ~/a.txt。当执行 mv ~/a.txt ~/o.txt 对文件进行更名后，~/h.txt 和 ~/s.txt 各自能否访问到其内容？

4. 设备文件分为哪两类？简述其二者传输数据时的主要差异。

5. 重定向分为哪两类？将标准错误输出流写入文件可用哪个重定向操作符？

6. 对一个当前工作目录下，名为 debug 的 bash script 以 + 号的形式赋予执行权限可用什么命令？

7. 如何查询文件的最后访问时刻？

✔ 综合实训

一、实训题目

读取处理器信息。

二、实训目的

认识符合 FHS 的 GNU/Linux 文件系统，并能应用管道命令符和 tee 命令。

三、实训内容

通过 /proc 获取机器的 CPU 信息，并将获取到的信息在屏幕输出之余同时写入当前工作目录中名为 cpu-info.log 的文件中。

第五章

GNU/Linux 网络配置

内容简介

- 作为经常使用 Linux 系统的开发人员或者网络管理员，学习 Linux 服务器的网络配置是非常重要的，掌握 Linux 网络配置，是为后续网络服务配置打下基础，要求大家认真学习。
- Linux 操作系统的网络配置通常包括配置主机名、IP 地址、子网掩码、默认网关以及 DNS 服务器等地址。本章主要介绍在命令行模式下配置 Linux 网络的方法。

学习要求

- 掌握常见的网络配置的文件以及相关参数
- 掌握常用的网络配置命令
- 掌握常用的网络调试工具
- 管理 Linux 的常用的网络服务

Linux 系统下配置网络服务有三种方式：第一种是安装 Linux 系统的过程中根据安装的选项配置网络；第二种是在安装系统完成后，使用图形化界面进行设置（应用程序 —— 设置 —— 网络），打开有线网络（自动分配 IP 地址）、勾选自动连接（重启后自动连接）；第三种是使用网络配置命令进行配置，本章主要介绍在命令行模式下配置 Linux 网络的方法。

5.1 网络配置文件

Linux 系统下可以通过修改相应的配置文件来改变主机名、域名、域名服务器、IP 地址、网络掩码和默认网关地址等，并且这些文件一般存放在 /etc 目录下。在 Linux 中，TCP/IP 网络的配置信息分别

存储在不同的配置文件中。相关的配置文件有 /etc/sysconfig/network、网卡配置文件、/etc/hosts、/etc/resolv.conf 以及 /etc/host.conf 等文件。下面分别介绍这些配置文件的作用和配置方法。

5.1.1　/etc/hosts 文件

hosts 文件是 Linux 系统中一个负责 IP 地址与域名快速解析的文件，以 ASCII 格式保存在“/etc”目录下，文件名为“hosts”（不同的 Linux 版本，这个配置文件也可能不同。比如 Debian 的对应文件是 /etc/hostname）。hosts 文件包含了 IP 地址和主机名之间的映射，还包括主机名的别名。在没有域名服务器的情况下，系统上的所有网络程序都通过查询该文件来解析对应于某个主机名的 IP 地址，否则就需要使用 DNS 服务程序来解决。通常可以将常用的域名和 IP 地址映射加入到 hosts 文件中，实现快速方便的访问。

这个文件可以配置主机 IP 及对应的主机名，对于服务器类型的 Linux 系统其作用还是不可忽略的。在局域网或是 INTERNET 上，每台主机都有一个 IP 地址，它区分开每台主机，并可以根据 IP 进行通信。但 IP 地址不方便记忆，所以又有了域名。在一个局域网中，每台机器都有一个主机名，用于区分主机，便于相互访问。访问文件默认内容如下：

```
[root@192 ~]# cat /etc/hosts
127.0.0.1 localhost localhost.localdomain localhost4 localhost4.localdomain4
::1  localhost localhost.localdomain localhost6 localhost6.localdomain6
```

通常情况下这个文件首先记录了本机的 IP 和主机名。

一般 /etc/hosts 的内容一般有如下类似内容：

```
127.0.0.1 localhost.localdomain localhost
192.168.199.128 CentOS 7
192.168.1.120 www.abc.com ftpserver
```

一般情况下 hosts 文件的每行为一个主机，每行由三部分组成，每个部分由空格隔开。其中 # 号开头的行做说明，不被系统解释。

hosts 文件的格式如下：

IP 地址主机名 / 域名

第一部分：网络 IP 地址。

第二部分：主机名或域名。

第三部分：主机名别名。

当然每行也可以是两部分，即主机 IP 地址和主机名；比如 192.168.199.128 CentOS 7。

这里可以稍微解释一下主机名（hostname）和域名（Domain）的区别：主机名通常在局域网内使用，通过 hosts 文件，主机名就被解析到对应 IP；域名通常在 internet 上使用，但如果本机不想使用 internet 上的域名解析，这时就可以更改 hosts 文件，加入自己的域名解析。

5.1.2 /etc/resolv.conf 文件

文件 /etc/resolv.conf 配置 DNS 客户，它包含了主机的域名搜索顺序和 DNS 服务器的地址，每一行应包含一个关键字和一个或多个的由空格隔开的参数。下面是一个例子文件：

```
[root@192 ~]# cat /etc/resolv.conf
# Generated by NetworkManager
search
nameserver 192.168.199.2
domain localdomain
```

这里 domain 和 search 是一个作用，当它们同时存在时，前者失效。

合法的参数及其意义如下：

nameserver：表明 DNS 服务器的 IP 地址。可以有很多行的 nameserver，每一个带一个 IP 地址。在查询时就按 nameserver 在本文件中的顺序进行，且只有当第一个 nameserver 没有反应时才查询下面的 nameserver。

domain：声明主机的域名。很多程序用到它，如邮件系统；当为没有域名的主机进行 DNS 查询时，也要用到。如果没有域名，主机名将被使用，删除所有在第一个点（.）前面的内容。

search：它的多个参数指明域名查询顺序。当要查询没有域名的主机，主机将在由 search 声明的域中分别查找。domain 和 search 不能共存；如果同时存在，后面出现的将会被使用。

sortlist：允许将得到域名结果进行特定的排序。它的参数为网络 / 掩码对，允许任意的排列顺序。

注意：在 Red Hat 中没有提供缺省的 /etc/resolv.conf 文件，它的内容是根据在安装时给出的选项动态创建的。

5.1.3 /etc/host.conf 文件

该文件指定如何解析主机名。Linux 通过解析器库来获得主机名对应的 IP 地址。

下面是一个“/etc/host.conf”的示例：

```
[root@192 ~]# cat /etc/hosts.conf
order bind,hosts
multi on
ospoof on
```

“order bind，hosts”指定主机名查询顺序，这里规定优先使用 DNS 来解析域名，如果 bind 不能满足要求，再查询“/etc/hosts”文件（也可以相反）。

“multi on”指定是否“/etc/hosts”文件中指定的主机可以有多个地址，拥有多个 IP 地址的主机一般称为多穴主机。

“nospoof on”指不允许对该服务器进行 IP 地址欺骗。IP 欺骗是一种攻击系统安全的手段，通过把 IP 地址伪装成别的计算机，来取得其他计算机的信任。

5.1.4　/etc/sysconfig/network-scripts/ifcfg-X 文件

安装 CentOS 7 系统后，网卡设备名、IP 地址、子网掩码、网关等配置信息都保存在网卡配置文件中。每一块网卡对应一个配置文件，配置文件存放在 /etc/sysconfig/network-scripts 目录中。网卡的命名一般为以“ifcfg-”开始后跟网卡类型（CentOS 7 里面的网卡名称不在是 eth0，1，2 而改成 enoxxxxxx 的格式，en 代表的是 enthernet 以太网）加网卡的信息索引号，如 ens33。

注意：CentOS6 及以前的版本，网卡命名方式会根据情况有所改变，不是唯一的不是固定的。网络接口使用连续号码命名：eth0、eth1 等。当增加或删除网卡时，名称可能会发生变化。

CentOS 7 采用 dmidecode 采集命名方案，一次来得到主板信息；它可以实现网卡命名唯一化（dmidecode 命令可以采集有关硬件方面的信息），对网络设备的命名方式如下：

（1）如果 Firmware（固件）或 BIOS 为主板上集成的设备提供的索引信息可用，且可预测则根据此索引进行命令，例如 ifcfg-ens33。

（2）如果 Firmware（固件）或 BIOS 为 PCI-E 扩展槽所提供的索引信息可用，且可预测则根据此索引进行命名，例如 ifcfg-enp33。

（3）如果硬件接口的物理位置信息可用，则根据此信息进行命名，例如 enp2s0。

（4）上述均不可用时，则使用传统命名机制。

知识扩展：

在 CentOS 7 中，en 表示 ethernet 以太网

enX（X 常见有下面 3 中类型）：

o：主板半载网卡，集成设备的设备索引号

p：独立网卡，PCI 网卡

s：热插拔网卡，USB 之类

nnn（数字）表示：MAC 地址 + 主板信息计算得出的唯一序列

网上配置文件中每一行进行一项内容的配置，采用“项目名称 = 项目设备值”的格式。其内容如下所示：

```
[root@192 ~]# cat /etc/sysconfig/network-scripts/ifcfg-ens33
TYPE=Ethernet
PROXY_METHOD=none
BROWSER_ONLY=no
BOOTPROTO=dhcp
DEFROUTE=yes
IPV4_FAILURE_FATAL=no
IPV6INIT=yes
IPV6_AUTOCONF=yes
IPV6_DEFROUTE=yes
IPV6_FAILURE_FATAL=no
IPV6_ADDR_GEN_MODE=stable-privacy
```

```
NAME=ens33
UUID=e7756e76-365a-4e87-b05d-8b8e22618871
DEVICE=ens33
ONBOOT=yes
```

以上配置文件中各项目名称的具体含义如下：

```
TYPE=Ethernet 网卡类型：以太网
PROXY_METHOD=none 代理方式：关闭状态
BROWSER_ONLY=no 只是浏览器 (yes|no)
BOOTPROTO=static 设置网卡获得 ip 地址的方式 (static|dhcp|none|bootp)
DEFROUTE=yes 设置为默认路由 (yes|no)
IPV4_FAILURE_FATAL=no 是否开启 IPV4 致命错误检测 (yes|no)
IPV6INIT=yes                          IPV6 是否自动初始化
IPV6_AUTOCONF=yes                     IPV6 是否自动配置
IPV6_DEFROUTE=yes                     IPV6 是否可以为默认路由
IPV6_FAILURE_FATAL=no 是否开启 IPV6 致命错误检测
IPV6_ADDR_GEN_MODE=stable-privacy     IPV6 地址生成模型
NAME=ens33                            网卡物理设备名称
UUID=e7756e76-365a-4e87-b05d-8b8e22618871   UUID 识别码
DEVICE= ens33                         网卡设备名称
ONBOOT= yes 开机自启 (yes|no)
```


5.1.5 /etc/services 文件

/etc/services 文件中保存了服务和端口以及协议的对应关系。但是我们通常会在服务的配置文件里会进行端口定义，那么自行定义的端口与 /etc/services 文件中服务与端口中，两者之间是什么关系呢？事实上，服务最终采用的方案仍然是配置文件用户自行定义的端口。但是 /etc/services 的存在有几个意义：

1. 如果每一个服务都能够严格遵循该机制，在此文件里标注自己所使用的端口信息，则主机上各服务间对端口的使用，将会非常清晰明了，易于管理。

2. 在该文件中定义的服务名，可以作为配置文件中的参数使用。

例如：在配置路由策略时，使用“www”代替“80”，即为调用了此文件中的条目“www 80”。

3. 且当有特殊情况，需要调整端口设置，只需要在 /etc/services 中修改 www 的定义，即可影响到服务。

例如：在文件中增加条目“Myport 8099”，在某个私有服务中多个配置文件里广泛应用，进行配置。当有特殊需要，要将这些端口配置改为 8098，则只需修改 /etc/services 文件中对应行即可。为了避免自定义端口与 etc/services 文件中的端口冲突，推荐的做法是在 /etc/services 里面加入新端口的定义或是使用已有的端口的定义，然后在监听的时候使用从 /etc/servies 里面得到的端口定义。这样和其他

程序有没有冲突，一目了然。而且一旦需要调整和重新分配端口的时候也容易。

文件中的每一行对应一种服务，它由 4 个字段组成，中间用 TAB 或空格分隔，分别表示“服务名称”“使用端口”“协议名称”以及“别名”。

```
[root@192 ~]# cat /etc/services
ftp          21/tcp
ftp          21/udp        fsp fspd
ssh          22/tcp                    # The Secure Shell (SSH) Protocol
ssh          22/udp                    # The Secure Shell (SSH) Protocol
telnet       23/tcp
telnet       23/udp
http         80/tcp        www www-http    # WorldWideWeb HTTP
http         80/udp        www www-http    # HyperText Transfer Protocol
http         80/sctp                   # HyperText Transfer Protocol
```

注意：由于 /etc/services 文件包含了服务名称和端口号之间的映射关系，很多系统程序都要使用这个文件。一般情况下，不建议修改该文件的内容，以免产生系统冲突造成用户无法正常访问资源。

5.1.6　/etc/sysconfig/network 文件

/etc/sysconfig/network 文件主要用于设置基本的网络配置，包括主机名称、网关等。文件中的内容如下所示：

```
[root@192 ~]# cat /etc/sysconfig/network
# Created by anaconda
NETWORKING=yes
HOSTNAME=CentOS 7
GATEWAY=192.168.199.128
```

其中各项含义如下：

NETWORKING：用于设置 Linux 网络是否运行，取值为 yes 或者 no。

HOSTNAME：用于设置主机名称。

GATEWAY：用于设置网关的 IP 地址。

对于 /etc/sysconfig/network 文件进行修改之后，需要重启网络服务或者注销系统以使配置文件生效。另外，在 CentOS 7 以后的版本中如果要修改主机名称（hostname），修改 /etc/sysconfig/network 文件不会起作用，只能修改 /etc/hostname 文件内容来进行，具体做法是将文件原有内容全部删除，然后写上新的主机名。

5.2 常用的网络配置命令

作为 Linux 系统管理员，掌握一定的网络配置命令是很有必要的，它可以节省很多在图形化界面中繁琐的步骤和一些无法操作的配置等。

5.2.1 网络配置命令 ifconfig

ifconfig 命令是一个用来查看、配置、启用或禁用网络接口的命令。用 ifconfig 命令配置网卡信息，在机器重启后，修改的配置就不存在。要想保存修改的配置，就需要修改网卡的配置文件。该命令的语法如下：

```
ifconfig [ 接口 ] [ 选项 |IP 地址 ]
```

ifconfig 命令各选项的含义如表 5-1 所示。

表 5-1 ifconfig 命令常用选项含义

选型	选项含义
-a	显示所有网络接口的状态
add< 地址 >	设置网络设备的 IPv6 地址
del< 地址 >	删除网络设备的 IPv6 地址
media< 类型 >	设置网络设备的媒介类型
netmask< 子网掩码 >	设置网络设备的子网掩码
up	激活指定的网络设备
down	关闭指定的网络设备
hw< 类型 >< 硬件地址 >	设置这个接口的硬件地址（MAC 地址）

【例 1】 查看本机网卡的状态

```
[root@192 ~]# ifconfig
ens33: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
inet 192.168.199.128  netmask 255.255.255.0  broadcast 192.168.199.255
inet6 fe80::9011:5d14:f12d:267a  prefixlen 64  scopeid 0x20<link>
ether 00:0c:29:a7:33:7a  txqueuelen 1000  (Ethernet)
RX packets 2563  bytes 248922 (243.0 KiB)
RX errors 0  dropped 0  overruns 0  frame 0
TX packets 1804  bytes 230210 (224.8 KiB)
TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
inet 127.0.0.1  netmask 255.0.0.0
inet6 ::1  prefixlen 128  scopeid 0x10<host>
loop  txqueuelen 1000  (Local Loopback)
RX packets 76  bytes 6472 (6.3 KiB)
```

```
RX errors 0  dropped 0  overruns 0  frame 0
TX packets 76  bytes 6472 (6.3 KiB)
TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
virbr0: flags=4099<UP,BROADCAST,MULTICAST>  mtu 1500
inet 192.168.122.1  netmask 255.255.255.0  broadcast 192.168.122.255
ether 52:54:00:50:bb:90  txqueuelen 1000  (Ethernet)
RX packets 0  bytes 0 (0.0 B)
RX errors 0  dropped 0  overruns 0  frame 0
TX packets 0  bytes 0 (0.0 B)
TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

从网卡信息可以看到，网上的名称是 ens33，物理地址（MAC）为：00：0c：29：a7：33：7a，IP 地址为：192.168.199.128，掩码 netmask：255.255.255.0，广播地址 broadcast：192.168.199.255。

本机环回地址（lo）主要用来测试网络，它代表设备的本地虚拟接口，其 IP 地址表示为：127.0.0.1，主要用来检查本地网络协议，基本数据接口是否正常等。

由于作者的 Linux 操作系统安装在虚拟机上，所以会显示虚拟网卡 virbr0 信息。

【例 2】 使用 ifconfig 命令设置服务器的 IP 地址，修改 IP 地址为 172.128.199.200，子网掩码为：255.255.0.0

```
[root@192 ~]# ifconfig ens33 172.128.199.200 netmask 255.255.0.0
```

注意：ifconfig 命令修改 IP 地址和 MAC 地址都是临时生效的，重新启动系统后设置失效。我们可以通过修改网卡配置文件使其永久生效，具体操作已经在前面叙述。

【例 3】 网卡的禁止和启用

```
[root@localhost ~]# ifconfig ens33 down      # 网卡的禁用
[root@localhost ~]# ifconfig ens33 up        # 网卡的启用
```

说明：使用 ifconfig ens33 down 命令后，在 Linux 主机还可以使用终端 ping 通本机网卡信息，但是在其他主机上就无法 ping 通 ens33 网上地址，而且如果使用 ssh 登录 Linux 服务器后，关闭了就不能现开启，必须在 Linux 系统中使用终端来启用。

在实际工作中，可能会出现一个网卡需要拥有多个 IP 地址的情况，可以通过设置虚拟网卡来实现，具体操作格式如下所示：

```
ifconfig 网卡名：虚拟网卡 ID  IP 地址 netmask 子网掩码
```

为第 1 块网卡 ens33 设置一个虚拟网卡，IP 地址为：172.168.199.200，子网掩码为：255.255.0.0，如果不设置 netmask，可以使用默认值。

【例 4】 配置虚拟网卡 IP 地址

```
[root@192 ~]# ifconfig ens33:1 172.168.199.200 netmask 255.255.0.0
```

结果如图 5-1 所示。

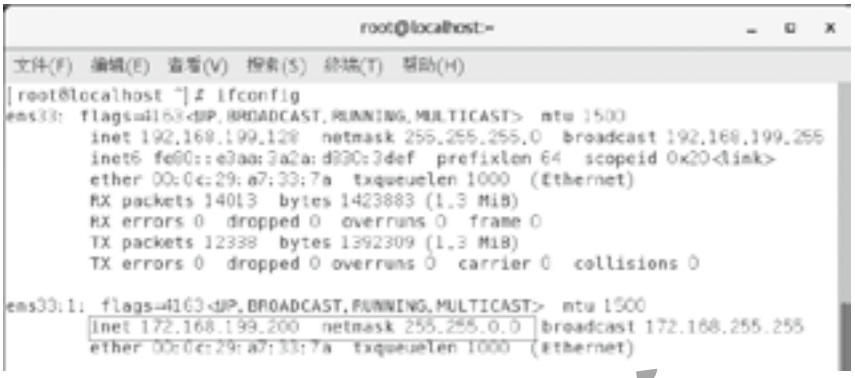

图 5-1 虚拟网卡效果图

5.2.2 hostname 配置主机名

主机名是网络上的唯一标志，所以设置主机名时要有规则地进行设置。Hostname 命令用于显示和修改设备系统的主机名称，但它的修改只是临时性的，重启系统后失效。如果需要永久修改主机名，需要修改 /etc/hosts 文件或者 /etc/hostname 文件。语法格式如下：

```
hostname [ 主机名 ]
```

【例 5】 使用 hostname 查看本机的主机名

```
[root@192 ~]# hostname
192.168.199.128
```

【例 6】 使用 Vim 修改 /etc/hosts 文件，修改主机名 localhsot 为 CentOS 7.8，修改后效果图如 5-2 所示。

图 5-2 修改主机名为 CentOS 7.8

【例 7】 使用 Vim 修改 /etc/sysconfig/network 文件中的 hostname 字段修改主机名为 CentOS 7.8，结果如图 5-3 所示。

图 5-3　修改 /etc/sysconfig/network 文件

大家修改完以上两个文件后，使用 hostname 命令查看本机主机名时，仍然是没有改变。这是因为在 CentOS 7 之后的版本，不支持修改 /etc/hosts 和 /etc/sysconfig/network 两个文件，需要修改 /etc/hostname 文件才真正起作用，如：vi /etc/hostname，修改主机名称为 CentOS 7 如图 5-4 所示。

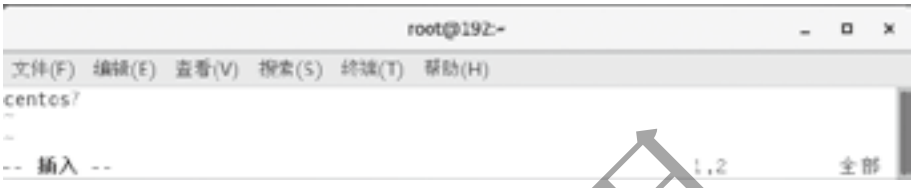

图 5-4　修改 /etc/hostname 文件

也可以直接在终端输入以下命令直接修改主机名：

```
[root@192 ~]# hostnamectl set-hostname CentOS 75.2.1 DHCP client
```

【例 8】　查看一下当前主机名的情况，可以使用 hostnamectl 查看全部三种主机名。

```
[root@192 ~]# hostnamectl
Static hostname: CentOS 7          # 静态主机名
Icon name: computer-vm
Chassis: vm
Machine ID: f7ddef19f6f3408dbd046e74b2f64214
Boot ID: 05572fe53ebc496ab084c32ab6779cdd
Virtualization: vmware
Operating System: CentOS Linux 7 (Core)
CPE OS Name: cpe:/o:centos:centos:7
Kernel: Linux 3.10.0-1062.el7.x86_64
Architecture: x86-64
或者：
[root@192 ~]# hostnamectl --static   # 静态主机名
CentOS 7
[root@192 ~]# hostnamectl --transient # 瞬态主机名
CentOS 7
[root@192 ~]# hostnamectl --pretty   # 灵活主机名
```

注意：在修改静态 / 瞬态主机名时，任何特殊字符或空白字符会被移除，而提供的参数中的任何大写字母会自动转化为小写。一旦修改了静态主机名，/etc/hostname 将被自动更新。然而，/etc/hosts 不会更新已保存所做的修改，所以你每次在修改主机名后一定要手动更新 /etc/hosts，之后再重启 CentOS 7。

5.2.3 ping 命令

不管在 Windows 平台，还是在 Linux 平台，ping 都是非常常用的网络命令。ping 命令通过 ICMP（Internet 控制消息协议）工作，并且 ping 可以用来测试本机与目标主机是否联通、联通速度如何、稳定性如何。

ping 命令运行在命令提示符终端，用法如下：

```
ping 参数目标主机
```

其中参数为零到多个，目标主机可以是 IP 或者域名，常用参数如表 5-2 所示。

表 5-2 ping 命令常用选项

参数	详解
-a	Audible ping.
-A	自适应 ping，根据 ping 包往返时间确定 ping 的速度；
-b	允许 ping 一个广播地址；
-B	不允许 ping 改变包头的源地址；
-c count	ping 指定次数后停止 ping；
-d	使用 Socket 的 SO_DEBUG 功能；
-F flow_label	为 ping 回显请求分配一个 20 位的“flow label”，如果未设置，内核会为 ping 随机分配；
-f	极限检测，快速连续 ping 一台主机，ping 的速度达到 100 次每秒；
-i interval	设定间隔几秒发送一个 ping 包，默认一秒 ping 一次；
-I interface	指定网卡接口、或指定的本机地址送出数据包；
-l preload	设置在送出要求信息之前，先行发出的数据包；
-L	抑制组播报文回送，只适用于 ping 的目标为一个组播地址
-n	不要将 ip 地址转换成主机名；
-p pattern	指定填充 ping 数据包的十六进制内容，在诊断与数据有关的网络错误时这个选项就非常有用，如：“-p ff”；
-q	不显示任何传送封包的信息，只显示最后的结果
-Q tos	设置 Qos(Quality of Service)，它是 ICMP 数据包相关位；可以是十进制或十六进制数，详见 rfc1349 和 rfc2474 文档；
-R	记录 ping 的路由过程 (IPv4 only)；注意：由于 IP 头的限制，最多只能记录 9 个路由，其他会被忽略；
-r	忽略正常的路由表，直接将数据包送到远端主机上，通常是查看本机的网络接口是否有问题；如果主机不直接连接的网络上，则返回一个错误
-S sndbuf	Set socket sndbuf. If not specified, it is selected to buffer not more than one packet.
-s packetsize	指定每次 ping 发送的数据字节数，默认为“56 字节”+“28 字节”的 ICMP 头，一共是 84 字节；包头 + 内容不能大于 65535，所以最大值为 65507（Linux:65507，Windows:65500）；

续表

参数	详解
-t ttl	设置 TTL(Time To Live) 为指定的值。该字段指定 IP 包被路由器丢弃之前允许通过的最大网段数;
-T timestamp_option	设置 IP timestamp 选项，可以是下面的任何一个：'tsonly' (only timestamps) 'tsandaddr' (timestamps and addresses) 'tsprespec host1 [host2 [host3]]' (timestamp prespecified hops).
-M hint	设置 MTU（最大传输单元）分片策略 可设置为： 'do'：禁止分片，即使包被丢弃; 'want'：当包过大时分片; 'dont'：不设置分片标志（DF flag）;
-m mark	设置 mark;
-v	使 ping 处于 verbose 方式，它要 ping 命令除了打印 ECHO-RESPONSE 数据包之外，还打印其他所有返回的 ICMP 数据包;
-W timeout	以毫秒为单位设置 ping 的超时时间;
-w deadline	deadline;

【例 9】　测试到环回地址为 127.0.0.1 的连通性。

```
[root@192 ~]# ping 127.0.0.1 -c 5
PING 127.0.0.1 (127.0.0.1) 56(84) bytes of data.
64 bytes from 127.0.0.1: icmp_seq=1 ttl=64 time=1.65 ms
64 bytes from 127.0.0.1: icmp_seq=2 ttl=64 time=0.091 ms
64 bytes from 127.0.0.1: icmp_seq=3 ttl=64 time=0.100 ms
64 bytes from 127.0.0.1: icmp_seq=4 ttl=64 time=0.093 ms
64 bytes from 127.0.0.1: icmp_seq=5 ttl=64 time=0.107 ms

--- 127.0.0.1 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4002ms
rtt min/avg/max/mdev = 0.091/0.409/1.655/0.623 ms
```

该测试表明到 127.0.0.1 的连通性正常，没有丢失分组。返回内容具体的含义如下：

① ping 目标主机的域名和 IP（ping 会自动将域名转换为 IP）。

②不带包头的包大小和带包头的包大小（参考“-s”参数）。

③ icmp_seq：ping 序列，从 1 开始；如果数字不是按顺序递增也就意味着丢包了。

ttl：剩余的 ttl（它就是 Time To Live，指的是报文在网络中能够‘存活’的限制）

time：响应时间，数值越小，联通速度越快

④发出去的包数，返回的包数，丢包率，耗费时间。

⑤最小 / 最大 / 平均响应时间和本机硬件耗费时间。

5.2.4 route 命令

Linux 系统中的 route 命令能够用于 IP 路由表的显示和操作。它的主要作用是创建一个静态路由让指定一个主机或者一个网络通过一个网络接口，如 ens33。当使用“add”或者“del”参数时，路由表被修改，如果没有参数，则显示路由表当前的内容。

一个网络中，需要一个路由器来转发不同广播域之间的数据，或是转发 lan 和 internet 之间的数据。有时我们需要设定这个路由器作为 Linux 系统的默认路由，那么就可以通过 route 命令来操作。甚至我们也可以用我们的 Linux 系统来充当路由器。route 命令格式如下所示：

```
route [ 选项 ][ 参数 ] [Command [Destination] [mask Netmask] [Gateway][metric Metric]] [if Interface]]
```

Command：指定你想运行的命令（Add/Change/Delete/Print）。

Destination：指定该路由的网络目标。

mask Netmask：指定与网络目标相关的网络掩码（也被称作子网掩码）。

Gateway：指定网络目标定义的地址集和子网掩码可以到达的前进或下一跃点 IP 地址。

metric Metric：为路由指定一个整数成本值标（从 1 至 9999），当在路由表（与转发的数据包目标地址最匹配）的多个路由中进行选择时可以使用。

if Interface：为可以访问目标的接口指定接口索引。若要获得一个接口列表和它们相应的接口索引，使用 route print 命令的显示功能，可以使用十进制或十六进。

route 命令常用选项如表 5-3 所示。

表 5-3　route 命令常用选项

选项	说明
-c	显示更多信息
-n	不解析名字
-v	显示详细的处理信息
-F	显示发送信息
-C	显示路由缓存
-f	清除所有网关入口的路由表
-p	与 add 命令一起使用时使路由具有永久性

route 命令常用参数如表 5-4 所示。

表 5-4　route 命令常用参数

参数	说明
add	添加一条新路由
del	删除一条路由
-net	目标地址是一个网络
-host	目标地址是一个主机
netmask	当添加一个网络路由时，需要使用网络掩码

续表

参数	说明
gw	路由数据包通过网关。注意，你指定的网关必须能够达到
metric	设置路由跳数

【例 10】　添加一条指向某个网络的路由

```
[root@192 ~]# route add -host 172.168.199.2 dev ens33
```

运行结果如图 5-5 所示。

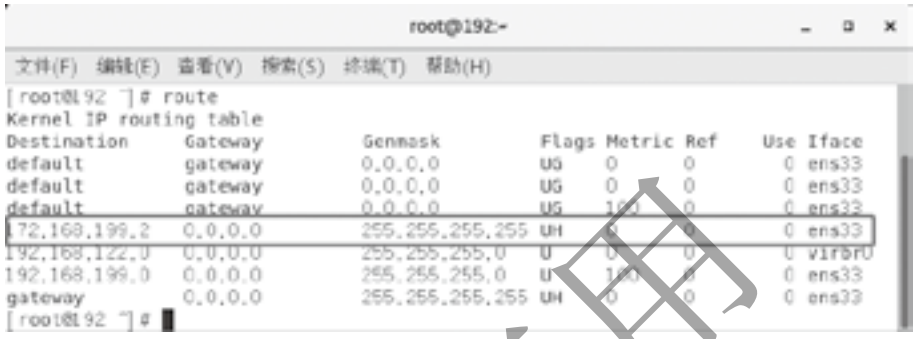

图 5-5　添加路由

可以发现添加的是主机的话，默认是会帮我们添加一个全 255 的子网掩码，表示子网范围就只有一个而已，那就是这台主机啦。

另外，Flags 标志说明

U Up：表示此路由当前为启动状态

H Host：表示此网关为一主机

G Gateway：表示此网关为一路由器

R Reinstate Route：使用动态路由重新初始化的路由

D Dynamically：此路由是动态性地写入

M Modified：此路由是由路由守护程序或导向器动态修改

要注意的是，直接在命令行下执行 route 命令来添加路由，不会永久保存，当网卡重启或者机器重启之后，该路由就失效了；可以在 /etc/rc.local 中添加 route 命令来保证该路由设置永久有效。当然，如果加上了 -p 参数的话那就会永久的生效了。

【例 11】　添加到网络的路由

```
[root@192 ~]# route add -net 10.20.30.40 netmask 255.255.255.248 ens33
[root@192 ~]# route
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
10.20.30.40     0.0.0.0         255.255.255.248 U     0      0        0 ens33
192.168.1.2     0.0.0.0         255.255.255.255 UH    0      0        0 ens33
192.168.122.0   0.0.0.0         255.255.255.0   U     0      0        0 virbr0
```

```
192.168.199.0   0.0.0.0         255.255.255.0   U     100    0        0 ens33
[root@192 ~]# route add -net 10.20.30.48 netmask 255.255.255.248 gw 10.20.30.41
[root@192 ~]# route
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
10.20.30.40     0.0.0.0         255.255.255.248 U     0      0        0 ens33
10.20.30.48     10.20.30.41     255.255.255.248 UG    0      0        0 ens33
192.168.1.2     0.0.0.0         255.255.255.255 UH    0      0        0 ens33
192.168.122.0   0.0.0.0         255.255.255.0   U     0      0        0 virbr0
192.168.199.0   0.0.0.0         255.255.255.0   U     100    0        0 ens33
```

【例 12】 添加到主机的路由、添加到网络的路由和添加默认路由

```
// 添加到主机的路由
[root@CentOS 7 ~]# route add -host 10.20.30.148 gw 192.168.199.2
// 添加到网络的路由
[root@CentOS 7 ~]# route add -net 10.20.30.40 netmask 255.255.255.248 ens33
[root@CentOS 7 ~]# route add -net 10.20.30.48 netmask 255.255.255.248\
gw 192.168.199.2
[root@CentOS 7 ~]# route add -net 192.168.1.0/24 ens33
// 添加默认路由
[root@CentOS 7 ~]# route add default gw 192.168.199.2
```

运行结果如图 5-6 所示。

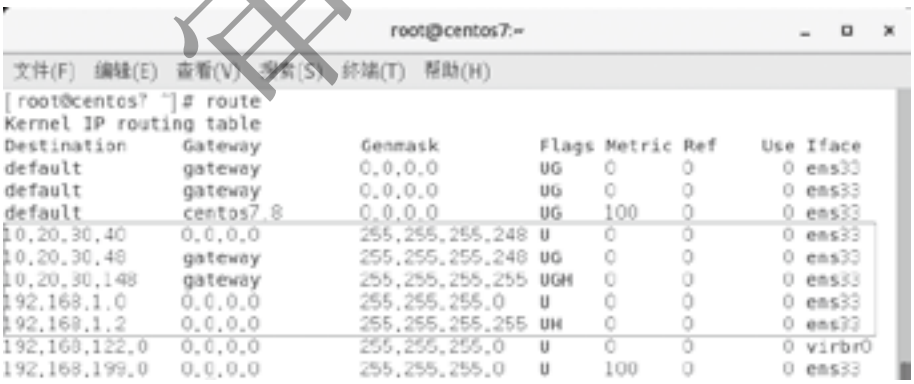

图 5-6 运行结果

注意：如果在 CentOS 7 配置路由时，出现“不可达”时或者在 /etc/sysconfig/network-scripts 目录中找不到 ifcfg-ens33 网卡配置文件时，需要在 Linux 系统中手动配置静态 IP；在虚拟机中查看虚拟网络编辑器 NAT 桥接 IP 以及网关，然后在 /etc/sysconfig/network-scripts 文件中编辑 ifcfg-ens33 网卡配置文件，并重启网卡服务即可解决。如图 5-7、图 5-8 所示。

图 5-7　虚拟网络编辑器信息

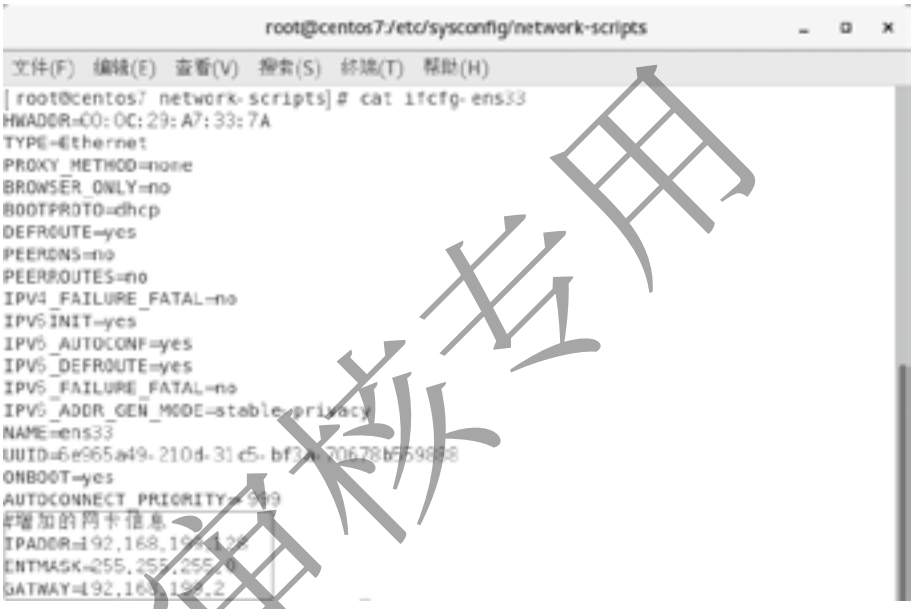

图 5-8　ifcfg-ens33 网卡配置信息

【例 13】　删除路由

```
[root@CentOS 7 ~]# route del -host 192.168.1.2 dev ens33
[root@CentOS 7 ~]# route del -net 10.20.30.40 netmask 255.255.255.248 ens33
[root@CentOS 7 ~]# route del -net 10.20.30.48 netmask 255.255.255.248\
gw 192.168.199.2
[root@CentOS 7 ~]# route del -net 192.168.1.0/24 ens33
[root@CentOS 7 ~]# route del  default gw 192.168.199.2
```

5.2.5　arp 命令

arp（地址解析协议）命令用于操作主机的 arp 缓冲区，它可以显示 arp 缓冲区中的所有条目，删除指定的条目或者添加静态的 IP 地址与 MAC 地址对应关系。语法如下：

```
arp（选项）（参数）
```

arp 常用选项如表 5-5 所示。

表 5-5　常用选项说明

-a< 主机 >	显示 arp 缓冲区的所有条目
-H< 地址类型 >	指定 arp 指令使用的地址类型
-d< 主机 >	从 arp 缓冲区中删除指定主机的 arp 条目
-D	使用指定端口的硬件地址
-e	以 Linux 的显示风格显示 arp 缓冲区中的条目
-i< 接口 >	指定要操作 arp 缓冲区的网络接口
-s< 主机 ><MAC 地址 >	设置指定的主机的 IP 地址与 MAC 地址的静态映射
-n	以数字的方式显示 arp 缓冲区中的条目
-v	显示详细的 arp 缓冲区条目，包括缓冲区条目的统计信息
-f< 文件 >	设置主机的 IP 地址与 MAC 地址的静态映射

arp 常用参数：

主机：查询 arp 缓冲区中指定主机的 arp 条目

【例 14】　添加静态映射，将目标 IP 地址映射固定 mac。

```
// 将目标 ip 地址映射固定 mac
[root@CentOS 7 ~]# arp -i eth0 -s 192.168.1.6 ff:ee:ee:ee:ee:ee
// 查看 arp 缓冲区
[root@CentOS 7 ~]# arp -a
? (192.168.199.2) at 00:50:56:eb:2a:2f [ether] on ens33
? (192.168.199.1) at 00:50:56:c0:00:08 [ether] on ens33
? (192.168.199.254) at 00:50:56:eb:10:19 [ether] on ens33
? (192.168.1.6) at ff:ee:ee:ee:ee:ee [ether] on ens34
```

【例 15】　以数字方式显示映射。

```
[root@CentOS 7 ~]# arp -vn
Address          HWtype  HWaddress          Flags Mask      Iface
192.168.199.2    ether   00:50:56:eb:2a:2f C        ens33
192.168.199.1    ether   00:50:56:c0:00:08 C        ens33
192.168.199.254  ether   00:50:56:eb:10:19 C        ens33
192.168.1.6      ether   ff:ee:ee:ee:ee:ee C        ens34
Entries: 4     Skipped: 0     Found: 4
```

5.2.6　traceroute 命令跟踪路由

traceroute 是用来检测发出数据包的主机到目标主机之间所经过的网关数量的工具。traceroute 的原理是试图以最小的 TTL（存活时间）发出探测包来跟踪数据包到达目标主机所经过的网关，然后监听一个来自网关 ICMP 的应答。发送数据包的大小默认为 38 个字节。

原理：程序利用增加存活时间（TTL）来实现其功能。每当数据包（3 个数据包包括源地址，目的地址和包发出的时间标签）经过一个路由器，其存活时间就会减 1。当其存活时间是 0 时，主机便取消数据包，并传送一个 ICMP（Internet 控制报文协议。它是 TCP/IP 协议族的一个子协议，用于在 IP 主机、路由器之间传递控制消息。控制消息是指网络通不通、主机是否可达、路由是否可用等网络本身的消息。这些控制消息虽然并不传输用户数据，但是对于用户数据的传递起着重要的作用）。 TTL 数据包给原数据包的发出者。

traceroute 程序完整过程：首先它发送一份 TTL 字段为 1 的 IP 数据包给目的主机，处理这个数据包的第一个路由器将 TTL 值减 1，然后丢弃该数据包，并给源主机发送一个 ICMP 报文（“超时”信息，这个报文包含了路由器的 IP 地址，这样就得到了第一个路由器的地址），然后 traceroute 发送一个 TTL 为 2 的数据包来得到第二个路由器的 IP 地址，继续这个过程，直至这个数据包到达目的主机。

1. 命令格式：

```
traceroute [ 参数 ] [ 主机 ]
```

2. 命令功能：

traceroute 指令让你追踪网络数据包的路由途径，预设数据包大小是 40Bytes，用户可另行设置。

3. 具体参数格式：

traceroute [-dFlnrvx][-f< 存活数值 >][-g< 网关 >…][-i< 网络界面 >][-m< 存活数值 >][-p< 通信端口 >][-s< 来源地址 >][-t< 服务类型 >][-w< 超时秒数 >][主机名称或 IP 地址][数据包大小]

4. 命令参数如表 5-6 所示。

表 5-6　命令参数说明

参数	说明
-d	使用 Socket 层级的排错功能
-f	设置第一个检测数据包的存活数值 TTL 的大小
-F	设置勿离断位
-g	设置来源路由网关，最多可设置 8 个
-i	使用指定的网络界面送出数据包
-I	使用 ICMP 回应取代 UDP 资料信息
-m	设置检测数据包的最大存活数值 TTL 的大小
-n	直接使用 IP 地址而非主机名称
-p	设置 UDP 传输协议的通信端口
-r	忽略普通的 Routing Table，直接将数据包送到远端主机上
-s	设置本地主机送出数据包的 IP 地址
-t	设置检测数据包的 TOS 数值
-v	详细显示指令的执行过程

续表

参数	说明
-w	设置等待远端主机回报的时间
-x	开启或关闭数据包的正确性检验

【例 16】 跟踪到达网址 www.baidu.com 的路由

```
[root@CentOS 7 ~]# traceroute www.baidu.com
traceroute to www.baidu.com (183.232.231.174), 30 hops max, 60 byte packets
1  gateway (192.168.199.2)  0.295 ms  0.165 ms  0.158 ms
2  * * *
3  * * *
4  * * *
5  * * *( 往下省略 )
```

说明：

记录按序列号从 1 开始，每个记录就是一跳，每跳表示一个网关，我们看到每行有三个时间，单位是 ms，其实就是 -q 的默认参数。探测数据包向每个网关发送三个数据包后，网关响应后返回的时间；如果你用 traceroute -q 4 www.baidu.com ，表示向每个网关发送 4 个数据包，如图 5-9 所示。

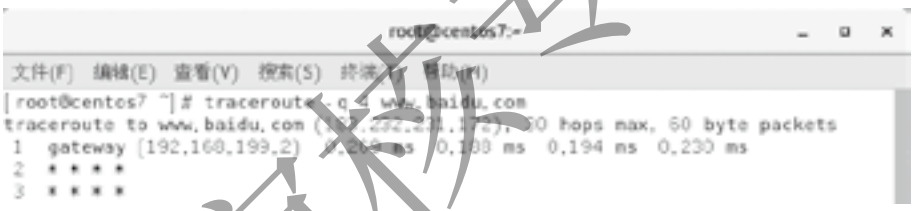

图 5-9 向网关发数据包

【例 17】 设置跟踪跳数

```
[root@CentOS 7 ~]# traceroute -m 5  www.baidu.com
traceroute to www.baidu.com (183.232.231.172), 5 hops max, 60 byte packets
1  gateway (192.168.199.2)  0.245 ms  0.242 ms  0.181 ms
2  * * *
3  * * *
4  * * *
5  * * *
[root@CentOS 7 ~]#
```

【例 18】 探测包使用的基本 UDP 端口设置 6868

```
[root@CentOS 7 ~]# traceroute -p 6868 www.baidu.com
traceroute to www.baidu.com (183.232.231.174), 30 hops max, 60 byte packets
1  gateway (192.168.199.2)  0.254 ms  0.216 ms  0.214 ms
```

```
2 * * *
3 * * *
4 * * *
5 * * *
```

Linux 下，traceroute 程序发送一个 UDP 数据包给目的主机，但是它选择一个不可能的值作为 UDP 端口号（大于 30000），使目的主机的任何一个应用程序都不可能使用该端口，因此该数据包到达目的主机时，目的主机会产生一个“端口不可达”错误的 ICMP 报文，这样 traceroute 程序要做的就是区分接收到的 ICMP 报文是超时还是端口不可达，从而来区分是路由器还是目的主机。

5.2.7　使用 setup 命令

Linux setup 命令设置公用程序，是一个启动图形设置系统的命令。

setup 命令：用来配置 X，打印设置，时区设置，系统服务，网络配置，配置，防火墙配置，验证配置，鼠标配置。

setup 是一个设置公用程序，提供图形界面的操作方式。在 setup 中可设置 7 类的选项：

1. 登录认证方式
2. 键盘组态设置
3. 鼠标组态设置
4. 开机时所要启动的系统服务
5. 声卡组态设置
6. 时区设置
7. X Windows 组态设置

CentOS 7 支持以文本窗口的方式对网络进行配置，CLI 命令行模式下使用 setup 命令就可以进入文本窗口，效果如图 5-10 所示。

图 5-10　0setup 命令文本窗口模式

```
[root@CentOS 7 ~]# setup
```

用“Tab”/“Alt+Tab”在元素间进行切换，选择同“网络配置”选项，按回车确认进入配置界面，界面简明易懂，不再详述。

5.3 常用的网络测试工具

熟练掌握和使用网络测试工具，对我们进行网络管理起到很大的作用。我们前面已经学习了常用的网络管理命令，这小节主要是在网络管理命令的基础上，再进一步学习其他常用的测试工具。

5.3.1 使用 netstat 命令

netstat 的功能主要是显示网络连接、路由表、接口状态、伪装连接、网络链路信息和组播成员组，可以查看显示网络连接、系统路由表、网络接口状态等。Netstat 支持 Unix、Linux 及 Windows 系统，功能非常强大。netstat 命令格式：

```
netstat [ 选项 ]
```

netstat 命令常用选项如表 5-7 所示。

表 5-7 netstat 命令常用选项说明

-a(all)	显示所有选项，默认不显示 LISTEN 相关
-t(tcp)	仅显示 tcp 相关选项
-u(udp)	仅显示 udp 相关选项
-n	拒绝显示别名，能显示数字的全部转化成数字
-l	仅列出有在 Listen（监听）的服务状态
-p	显示建立相关链接的程序名
-r	显示路由信息，路由表
-e	显示扩展信息，例如 uid 等
-s	按各个协议进行统计
-c	每隔一个固定时间，执行该 netstat 命令

【例 19】 禁用反向域名解析，加快查询速度

默认情况下 netstat 会通过反向域名解析技术查找每个 IP 地址对应的主机名。这会降低查找速度。如果你觉得 IP 地址已经足够，而没有必要知道主机名，就使用 -n 选项禁用域名解析功能。

```
[root@CentOS 7 ~]# netstat -atn
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 127.0.0.1:631           0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:111             0.0.0.0:*               LISTEN
tcp        0      0 192.168.122.1:53        0.0.0.0:*               LISTEN
```

```
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN
tcp        0      0 192.168.199.128:22      192.168.199.1:50207     ESTABLISHED
tcp6       0      0 ::1:631                 :::*                    LISTEN
tcp6       0      0 :::111                  :::*                    LISTEN
tcp6       0      0 :::22                   :::*                    LISTEN
tcp        0      0 192.168.199.128:22      192.168.199.1:50207     ESTABLISHED
```

因为作者是采用 ssh 协议登录到虚拟中的主机，所以可以看到这是建立的一条 tcp 连接。

【例 20】　只列出监听中的连接

任何网络服务的后台进程都会打开一个端口，用于监听接入的请求。这些正在监听的套接字也和连接的套接字一样，也能被 netstat 列出来。使用 –l 选项列出正在监听的套接字。

```
[root@CentOS 7 ~]# netstat -tnl
Active Internet connections （only servers）
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 127.0.0.1：631          0.0.0.0：*              LISTEN
tcp        0      0 0.0.0.0：111            0.0.0.0：*              LISTEN
tcp        0      0 192.168.122.1：53       0.0.0.0：*              LISTEN
tcp        0      0 0.0.0.0：22             0.0.0.0：*              LISTEN
tcp6       0      0 ：1：631                ：：*                   LISTEN
tcp6       0      0 ：：111                 ：：*                   LISTEN
tcp6       0      0 ：：22                  ：：*                   LISTEN
```

现在我们可以看到处于监听状态的 TCP 端口和连接。如果你查看所有监听端口，去掉 –t 选项。如果你只想查看 UDP 端口，使用 –u 选项，代替 –t 选项。

【例 21】　获取进程名、进程号以及用户 ID

查看端口和连接的信息时，能查看到它们对应的进程名和进程号对系统管理员来说是非常有帮助的。举个例子，Apache 的 httpd 服务开启 80 端口，如果你要查看 http 服务是否已经启动，或者 http 服务是由 apache 还是 nginx 启动的，这时候你可以看看进程名。使用 –p 选项查看进程信息。

```
[root@CentOS 7 ~]# netstat -tnlp
Active Internet connections (only servers)  # 激活 Internet 连接 ( 仅服务器 )
Proto Recv-Q Send-Q Local Address   Foreign   Address  State   PID/Program name
tcp    0      0    127.0.0.1:631   0.0.0.0:*    LISTEN     1146/cupsd
tcp    0      0    0.0.0.0:111     0.0.0.0:*    LISTEN     1/systemd
tcp    0      0  192.168.122.1:53   0.0.0.0:*    LISTEN     1390/dnsmasq
tcp    0      0    0.0.0.0:22      0.0.0.0:*    LISTEN     1143/sshd
tcp6   0      0    ::1:631         :::*         LISTEN     1146/cupsd
tcp6   0      0    :::111          :::*         LISTEN     1/systemd
tcp6   0      0    :::22           :::*         LISTEN     1143/sshd
```

使用 -p 选项时，netstat 必须运行在 root 权限之下，不然它就不能得到运行在 root 权限下的进程名，而很多服务包括 http 和 ftp 都运行在 root 权限之下。相比进程名和进程号而言，查看进程的拥有者会更有用。使用 -ep 选项可以同时查看进程名和用户名。

【例 22】 显示内核路由信息

使用 -r 选项打印内核路由信息。打印出来的信息与 route 命令输出的信息一样。我们也可以使用 -n 选项禁止域名解析。

```
[root@CentOS 7 ~]# netstat -rn
Kernel IP routing table
Destination     Gateway         Genmask         Flags   MSS Window  irtt Iface
0.0.0.0         192.168.199.2   0.0.0.0         UG        0 0          0 ens33
192.168.122.0   0.0.0.0         255.255.255.0   U         0 0          0 virbr0
192.168.199.0   0.0.0.0         255.255.255.0   U         0 0          0 ens33
```

【例 23】 查看服务是否在运行

如果你想看看 http，smtp 或 tcp 服务是否在运行，使用 grep。

```
[root@CentOS 7 ~]# netstat -aple | grep tcp
tcp 0 0  CentOS 7.8:ipp      0.0.0.0:*   LISTEN   root  26207  1146/cupsd
tcp 0 0  0.0.0.0:sunrpc      0.0.0.0:*   LISTEN   root  18435   1/systemd
tcp 0 0  192.168.122.1:domain 0.0.0.0:*  LISTEN   root  27454 1390/dnsmasq
tcp 0 0  0.0.0.0:ssh         0.0.0.0:*   LISTEN   root  26088 1143/sshd
tcp 0 0 192.168.199.128:ssh 192.168.199.1:50207 ESTABLISHED root 41417     2622/
sshd: root@pts
tcp6 0 0 localhost:ipp       [::]:*      LISTEN   root   26206  1146/cupsd
tcp6 0 0 [::]:sunrpc         [::]:*      LISTEN   root   18441  1/systemd
tcp6 0 0 [::]:ssh            [::]:*      LISTEN   root   26111  1143/sshd
```


5.3.2 systemd 和 systemctl

早期的 CentOS 5 系统上，使用的 init 系统是 SysV init。这套系统运行稳定，使用 shell 脚本的形式，串行地一个个启动进程，当前一个进程启动完毕后，再启动后一个。这种方式带来的缺点是：计算机的启动速度很慢，后面的进程必须等待前面的进程依次串行地启动完毕方可启动。有的服务可能在启动后很长的一段时间或者几乎不会使用到，例如服务器中的打印服务。发展到 CentOS 6 的时候，init 系统换成了 Upstart，这个系统有效地解决了上述的缺点，实现了并行启动和按需启动。而到了 CentOS 7 的时候，则出现了 systemd 这套 init 系统，它更加完善了并行启动和按需启动，使得启动的速度更上一层楼。这里的按需启动，指的是某个服务被设置为开机启动，当开机时该服务并没有真正的启动，只是注册好了 socket 占用，等待第一次有请求进来的时候，systemd 悬挂该请求然后启动服务，服务启动完毕后就恢复该请求，从而实现按需启动，这种方式也变相加速了系统的启动速度。

systemd 不仅仅是一个服务，还可以用于管理系统自身的方方面面，例如日志、类似 crond 的任务调度程序（timer）、电源管理等。systemd 支持向后兼容（backward compatible）SysV init 的脚本等其他的诸多特性，使得其目前在慢慢成为 Linux 发行版的一种标准。

在 systemd 中使用配置文件来进行管理，这些配置文件叫作 unit。unit 文件存在于三个位置：

1./usr/lib/systemd/system/：软件程序包所提供的 unit 文件。例如 httpd 程序包提供了 httpd.service 文件。

2./run/systemd/system/：运行时所创建的 unit 文件。优先级高于已安装的服务 unit 文件所在的目录。

3./etc/systemd/system/：通过 systemctl enable 所创建的和用于扩展一个服务所添加的 unit 文件目录。优先级高于运行时 unit 文件。

优先级：/etc/systemd/system/、/run/systemd/system/、 /usr/lib/systemd/system/。

由于向后兼容的特性的存在，旧命令依然可用，但是官方并不建议。新旧命令的对应关系如表 5-8 所示。service 命令和 systemctl 命令对照表，service 命令用于管理一个服务的启动和停止。

表 5-8　service 命令和 systemctl 命令对照表

service	systemctl	描述
service name start	systemctl start name.service	启动一个服务
service name stop	systemctl stop name.service	停止一个服务
service name restart	systemctl restart name.service	重启一个服务
service name condrestart	systemctl try-restart name.serivce	只有当服务处于运行状态时才重启服务
service name reload	systemctl reload name.service	重载服务，一般用于配置文件的修改后执行
serivce name status	systemctl status name.service systemctl is-active name.service	查看服务的运行状态
service --status-all	**systemctl list-units --type service --all**	**查看所有服务的运行状态**

chkconfig 命令和 systemctl 命令对照表如表 5-9 所示。chkconfig 命令用于管理服务的开机启动情况以及将服务纳入 chkconfig 管理。

表 5-9　chkconfig 命令和 systemctl 命令对照表

chkconfig	systemctl	描述
chkconfig name on	systemctl enable name.service	使服务开机启动
chkconfig name off	systemctl disable name.service	禁止服务开机启动
chkconfig --list name	systemctl status name.service systemctl is-enable name.service	查看服务是否开机启动

在书写 unit 名称的时候，一般是建议写全名，这样比较能顾名思义，例如“httpd.service”。不过

简写，不写 unit 类型后缀，那么 systemctl 也是可以识别的，如下两个命令的效果是一样的，systemctl 会自动识别 unit 类型。例如：

```
[root@CentOS 7 ~]# systemctl stop nfs-server.service
[root@CentOS 7 ~]# systemctl stop nfs-server
```

本章小结

本章主要要求学生掌握常见的网络配置的文件以及相关参数，并能独立完成网络的配置，同时要重点掌握常用的网络配置命令，学会对网络故障的排查；还需要掌握常用的网络调试工具和管理 Linux 的常用的网络服务。

习题

一、选择题

1. 局域网的网络地址 192.168.1.0/24，局域网络连接其他网络的网关地址是 192.168.1.1。主机 192.168.1.20 访问 172.16.1.0/24 网络时，其路由设置正确的是（　　）。

A. route add －net 192.168.1.0 gw 192.168.1.1 netmask 255.255.255.0 metric 1

B. route add －net 172.16.1.0 gw 192.168.1.1 netmask 255.255.255.0 metric 1

C. route add －net 172.16.1.0 gw 172.168.1.1 netmask 255.255.255.0 metric 1

D. route add default 192.168.1.0 netmask 172.168.1.1 metric 1

2. 下列提法中，不属于 ifconfig 命令作用范围的是（　　）。

A. 配置本地回环地址　B. 配置网卡的 IP 地址　C. 激活网络适配器　D. 加载网卡到内核中

3. 下列文件中，包含了主机名到 IP 地址的映射关系的文件是（　　）。

A. /etc/HOSTNAME　B. /etc/hosts　C. /etc/resolv.conf　D. /etc/networks

4. 当我们与某远程网络连接不上时，就需要跟踪路由查看，以便了解在网络的什么位置出现了问题，满足该目的的命令是（　　）。

A. ping　B. ifconfig　C. traceroute　D. netstat

5. 以下哪个命令能用来显示 server 当前正在监听的端口？（　　）

A. ifconfig　B. netlst　C. iptables　D. netstat

6. 哪个文件存放机器名到 IP 地址的映射？（　　）

A. /etc/hosts　B. /etc/host　C. /etc/host.equiv　D. /etc/hdinit

7. 快速启动网卡“eth0”的命令是（　　）。

A. ifconfig eth0 noshut　B. ipconfig eth0 noshut　C. ifnoshut eth0　D. ifup eth0

8. Linux 系统提供了一些网络测试命令，当与某远程网络连接不上时，就需要跟踪路由查看，以便了解在网络的什么位置出现了问题，请从下面的命令中选出满足该目的的命令。（　　）

A. ping　B. ifconfig　C. traceroute　D. netstat

9. 拨号上网使用的协议通常是（　　）。

A. PPP　　B. UUCP　　C. SLIP　　D. Ethernet

10. 设置 Linux 系统默认运行级别的文件是（　　）

A. /etc/init　　B. /etc/inittab　　C. /var/inittab　　D. /etc/initial

二、填空题

1. ping 命令用于测试网络的连通性，ping 命令通过_____协议来实现。

2. 在使用手工的方法配置网络时，可通过修改_____文件来改变主机名，若要配置该计算机的域名解析客户端，需配置_____文件。

3. 在 Linux 系统中，测试 DNS 服务器是否能够正确解析域名的客户端命令，使用命令。

4. 欲发送 10 个分组报文测试与主机 abc.tuu.edu.cn 的连通性，应使用的命令和参数是_____。

5. _____文件主要用于设置基本的网络配置，包括主机名称、网关等。

6. 一块网卡对应一个配置文件，配置文件位于目录_____中，文件名以_____开始后跟网卡类型（通常使用的以太网卡用_____代表）加网卡的序号（从“0”开始）。如第二块以太网卡的配置文件名为_____。

✓综合实训

一、实训名称

TCP/IP 网络接口配置

二、实训目的

1. 掌握 Linux 下 TCP/IP 网络的设置方法。

2. 学会使用命令检测网络配置。

3. 学会启用和禁用系统服务。

三、项目背景

某企业新增了 Linux 服务器，但还没有配置 TCP/IP 网络参数，请设置好各项 TCP/IP 参数，并连通网络。

四、实训内容

练习 Linux 系统下 TCP/IP 网络设置、网络检测方法。

五、实训步骤

子项目 1. 设置 IP 地址及子网掩码

1. 查看网络接口 eth0 的配置信息。

2. 为此网络接口设置 IP 地址、广播地址、子网掩码、并启动此网络接口。利用 ifconfig 命令查看系统中已经启动的网络接口。仔细观察所看到的现象，记录启动的网络接口。

子项目 2. 设置网关和主机名

1. 显示系统的路由设置。

2. 设置默认路由。并再次显示系统的路由设置，确认设置成功。

3. 显示当前的主机名设置；并以自己姓名缩写重新设置主机名。再次显示当前的主机名设置，确认修改成功。

子项目 3. 网络设置检测

1.ping 网关的 IP 地址，检测网络是否连通。

2. 用 netstat 命令显示系统核心路由表。

3. 用 netstat 命令查看系统开启的 TCP 端口。

子项目 4. 设置域名解析

1. 编辑 /etc/hosts 文件，加入要进行静态域名解析的主机的 IP 地址和域名。

2. 用 ping 命令检测上面设置好的网关的域名，测试静态域名解析是否成功。

3. 编辑 /etc/resolv.conf 文件，加入域名服务器的 IP 地址，设置动态域名解析。

4. 编辑 /etc/host.conf 文件，设置域名解析顺序为：hosts，bind。

5. 用 nslookup 命令查询一个网址对应的 IP 地址，测试域名解析的设置。

六、实训思考题

1. 当无法连接远程主机的时候，例如，用 telnet 命令无法连接到远程主机 remost.net，此时应该按什么顺序，用什么方法，分别检测系统中的哪些设置?

2. 静态域名解析和动态域名解析有什么区别？分别在哪些文件里进行设置？系统如何决定用哪种方式对一个域名进行解析?

3. 利用 ifconfig 和 route 命令配置的 IP 地址、子网掩码和默认网关等信息和利用 netcofig 及编辑 /etc/syscofig/network-scripts/if-eth0 文件配置的 IP 地址、子网掩码和默认网关等信息有什么不同?

第六章

Vim 与 Shell 脚本程序设计

内容简介

- 本章节的内容主要是围绕 Linux 文本编辑器 Vim 以及 Shell 脚本程序设计两大内容展开的。首先，对 Vim 编辑器的三种工作模式进行详细说明，同时介绍了三种工作模式下的各种常用命令；然后解释说明其他非常重要的 Linux 命令：数据流重定向、管道命令以及环境变量等，为后续的 shell 脚本程序设计提供一些必要的预备知识；最后针对 Shell 脚本程序设计有关的语法规则、程序控制语句等内容进行详细地解释。

学习要求

- 熟悉掌握并使用 Vim 文本编辑器的三种工作模式下的常用命令，使用对应快捷键加快日常工作速度；熟练掌握 Shell 脚本编写语法规则，根据实际需求采用相应的脚本完成功能编写。

6.1 Vim 文本编辑器基础知识

Linux 系统是以文本模式进行工作，在 Linux 系统中使用文本编辑器来编辑对应的 Linux 参数配置文件是非常重要的一件事情。因此，普通用户或系统管理员至少应该熟悉一种文字处理器。基于 CentOS 7 的 Linux 系统中所默认附加的文本编辑器就是我们这个章节要学习的 Vim。

Vim 是 Vi 的增强版，后者工作在其他大部分的 UNIX 系统中。在很多非正式的场合中，Vim 和 Vi 是一回事，这个编辑器是所有 UNIX 和 Linux 系统上的默认标准软件，因此对于系统管理员有非常重要的意义。

Vim 具有程序编辑的能力，可以主动地以字体严肃辨别语法的正确性，方便程序设计。

本节主要介绍 Vim 的基本使用，包括编辑保存、搜索替换和针对程序员的配置 3 个部分。最后以一张命令表结束本节的内容，更为详细的 Vim 使用请参考 Vim 手册。

Vim 文本编辑器可分为三种模式，分别是一般指令模式、编辑模式和指令列命令模式。三种模式的作用如下所述：

6.1.1 Vim 文本编辑器的三种工作模式

1. 一般指令模式

当用 Vim 打开一个文件时就直接进入一般指令模式了，该模式为默认的模式，也简称为一般模式。在这个模式中，你可以使用“上下左右”按键或“H、J、K、L”（H、J、K、L 分别代表向左、向上、向下、向右）这 4 个键来移动光标，你可以使用“删除字符”或“删除整列”来处理文件内容，也可以使用“复制、粘贴”来处理文件数据。简而言之，在该模式下，可以控制光标的移动，同时可以对文本进行删除、复制、粘贴等工作。

2. 编辑模式

在一般指令模式中可以进行删除、复制、粘贴等操作，但是无法编辑文件中的内容。只有当你按下“i、I、o、O、a、A、r、R”等任何一个字母之后才能进入编辑模式。

注意：通常在 Linux 中，按下这些按键时，在画面的左下方会出现“INSERT 或 REPLACE”（插入或替换的意思）的字样，此时，才可以进行编辑。如果要回到一般指令模式时，则必须按下“Esc”这个按键方可退出编辑模式。总的来说，在该模式下，我们可以完成正常的文本录入工作。

3. 指令列命令模式

在一般命令模式时，输入“：/ ?”三个中的任何一个按键，就可以将光标移动到最底下那一列。在这个模式当中，可以给你提供“搜寻资料”的动作，而读取、存盘、大量取代字符、离开 Vim、显示行号等的动作都是在此模式中达成的。总的来说，在该模式下，我们可以完成保存文，退出 Vim 及设置编辑环境。其中：q 表示退出 Vim；：wq 表示保存并退出 Vim；：w file 表示当前文件以文件名 file 保存在当前目录中等。/string 用于搜索一个字符串，例如 /today 指令，用于在文件中搜索 today 字符串。使用 n 跳转到下一个出现字符串的地方。Vim 的搜索是可以循环进行的。

Linux 系统中对于字母的大小写是非常敏感的，如果想让 Vim 搜索字符串过程中忽略大小写，可以输入“：set ignorecase”指令。设置该命令之后，意味着搜索 /Today 和 /today 是没有任何区别的。如果要重新开启大小写敏感，只要使用下面这条命令即可。

```
:set noignorecase       # 开启大小写敏感
:set ignorecase  # 忽略大小写
```

简单地说，我们可以将这三个模式之间的关系通过图 6-1 来表示。

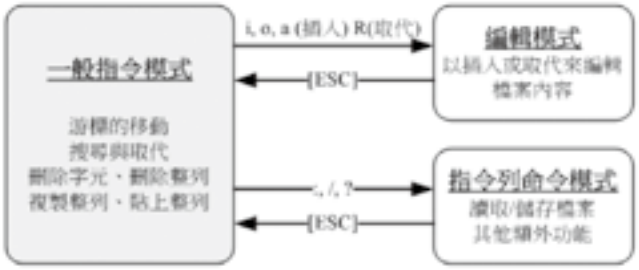

图 6-1　Vim 三种模式的相互关系

从上图中，你会发现一般指令模式可与编辑模式及指令列模式切换，但编辑模式与指令列模式之间可以互相切换。同时记住，每次运行 Vim 编辑器后都模式是“一般命令模式”，需要进入“输入模式”后才能进行编写文档的工作，而每次编辑完成需通过“Esc”键先返回到“一般命令模式”后再进入“指令列命令模式”对文本的保存或退出操作。

6.1.2　Vim 的基本使用

通过上小节的学习，我们已经基本了解 Vim 文本编辑器的三种工作模式。那么在该小节中，我们将学习使用 Vim 文本编辑器如何编写和处理一个简单的文档，加强熟悉 Vim 编辑器的使用。如何使用 Vim 文本编辑器建立一个名为 welcome.txt 的文件?

1. 首先，要编辑一个文件，可以在终端命令行下输入 Vim file。如果 file 不存在，那么 Vim 会自动新建一个名为 file 的文件。如果使用不带任何参数的 Vim 命令，那么就需要在保存的时候指定文件名。同时，Vim 会认为这个人应该是第一次使用这个软件，从而给出一些版本和帮助信息，如图 6-2 所示。

图 6-2　Vim 初始界面

2. 进入命令终端，如图 6-3 所示。

图 6-3　Linux 终端命令行窗口

3. 在命令行中输入“Vim welcome.txt”，便可新建 Vim welcome.txt 文档，如图 6-4 所示。直接输入“Vim 文档名”便能进入 Vim 的一般指令模式。

图 6-4　Vim 创建新文件

4. 敲入【Enter】键进入文档中，如图 6-5 所示，属于一般指令模式。整个 Vim 界面主要分为两部分，上半部分显示的是文件的实际内容，最底下一列则是状态显示列（如下图的 [New File] 信息），或者是命令下达列。

图 6-5　进入 Vim 一般命令模式窗口

5. 输入 a，进入编辑模式，在 Vim 文本编辑器的左下角出现“-- INSERT --”，表明当前状态可以进行正常编辑工作，除了键盘上的【ESC】，因为只要按该键便会切回“一般指令模式”，如图 6-6 所示（在一般指令模式之中，只要按下 i、o、a 等字符就可进入编辑模式）。

图 6-6　使用“i、o、a”等进入 Vim 的编辑模式

6. 编辑文档，比如：输入一行文字，“Hello，Vim！”，如图 6-7 所示。

图 6-7　编辑文档

7. 接着，敲入【Esc】键，切换到“一般指令模式”，如图 6-8 所示，左下角的提示“-- INSERT --”消失不见。

图 6-8　通过【Esc】按键切换到一般指令模式

8. 存档离开：敲入“：wq”保存文档并退出 Vim（注意，按下“：”时，光标就会移动到最底下一列），如图 6-9 所示。

图 6-9　输入“:wq”保存文档并退出 Vim

9. 最后查看文档“welcome.txt”，输入“cat welcome.txt”，结果如图 6-10 所示。

图 6-10　使用“cat 文件名”查看文档内容

10. 其他命令，此处就不多进行演示，请大家自行练习。

6.1.3　Vim 的常用命令

这里为大家总结了 Vim 文本编辑器中最常用的快捷键命令，大家尽量记一下，忘记了来查也可以。

“一般命令模式”中常用的快捷键，主要分为三大类：光标移动、复制、粘贴及删除等命令以及文字搜索和替换等命令；具体命令详细说明如表 6-1、表 6-2、表 6-3 所示。

表 6-1　移动光标的方法

命令	作用
h 或向左箭头键（←）	光标向左移动一个字符
j 或向下箭头键（↓）	光标向下移动一个字符
k 或向上箭头键（↑）	光标向上移动一个字符
l 或向右箭头键（→）	光标向右移动一个字符
如果你将右手放在键盘上的话，你会发现“hjkl”是排列在一起的，因此可以使用这四个按钮来移动光标。如果进行多次移动的话，例如向下移动 30 列，可以使用“30j”或“30 ↓”的组合按键，即加上想要进行的次数（数字）后，按下动作即可	
【ctrl】+【f】	屏幕“向下”移动一页，相当于【Page Down】按键
【ctrl】+【b】	屏幕“向下”移动一页，相当于【Page Down】按键
【ctrl】+【d】	屏幕“向下”移动半页
【ctrl】+【u】	屏幕“向上”移动半页
0 或功能键【Home】	这是数字“0”：光标移动到这一列的最前面字符处
$ 或功能键【End】	光标移动到这一列的最后面字符处
G	移动到这个文件的最后一列
nG	n 为数字。光标移动到这个文件的第 n 列。例如 20G 表示光标会移动到这个文件的第 20 列（可配合 :set nu ，显示行号命令）
gg	移动到这个文件的第一列，相当于 1G
n<Enter>	n 为数字。光标向下移动 n 列

表 6-2　复制、删除（剪贴）、粘贴等相关指令

命令	作用
x,X	在一列文字当中，x 表示向后删除一个字符（相当于【delete】键），X 则表示向前删除一个字符（相当于【backspace】键）
nx	n 为数字，连续向后删除 n 个字符。例如，要连续删除 10 个字符，则输入“10x”.
dd	删除（剪切）光标所在整行
ndd	n 为数字，删除（剪切）从光标处开始的 n 行。例如 5dd，表示删除从光标处开始的 5 行文字
yy	复制光标所在整行
nyy	n 为数字，复制从光标处开始的 n 行。例如 5yy，表示复制从光标处开始的 5 行文字
P，P	p 为将之前剪切（dd）或复制（yy）过的文字粘贴到光标下一列，P 则表示将文字粘贴在光标的上一列
u	撤销上一步的操作
【ctrl】+r	重做上一个操作
.	这个是小数点，表示重复前一个动作。例如，如果你想要重复删除、重复粘贴等动作，按下小数点“.”即可

表 6-3　搜索及替换

命令	作用
/word	向光标之下寻找 word 这一字符串。例如要在文件内搜索 Linux 这个字符串，就输入 /Linux 即可
？ Word	向光标之上寻找 Word 这一字符串
n	显示搜索命令定位到的下一个字符串。举例来说，如果刚刚你执行 /Linux 去向下搜索 Linux 字符串，则按下 n 后，会继续搜索下一个 Linux 字符串。如果执行 ?Linux 的话，那么按下 n 后则会向上继续搜索 Linux 字符串
N	与 n 刚好相反，显示搜索命令定位到的上一个字符串。例如 /Linux 后，按下 N 则表示“向上”继续搜索 Linux 字符串，即继续执行“反向”搜索操作
:n1,n2s/word1/word2/g	n1 与 n2 为数字。表示在第 n1 与 n2 列之间寻找 word1 这个字符串，并将该字符串替换为 word2。例如，在 100 到 200 列之间搜索 Linux 并取代为 Linux 则：“:100,200s/Linux/Linux/g”
:1,$s/ word1/word2/g	从第一列到最后一列搜索 word1 字符串，并将该字符串替换为 word2
:1,$s/ word1/word2/g	从第一列到最后一列搜索 word1 字符串，并将该字符串替换为 word2。且在替换前显示提示符给用户确认（confirm）是否需要替换
:	切换到“指令行命令模式”

“一般指令模式”切换到“编辑模式”中常用的快捷键，具体命令如表 6-4 所示。

表 6-4　“插入”或“替换”的编辑模式

命令	作用
a,A	进入插入模式（Insert mode）： a 表示在光标所在的下一个字符处开始插入，A 则表示在光标所在列的最后一个字符处开始插入
i,I	进入插入模式（Insert mode）： i 表示在光标所在位置插入，I 则表示在当前所在列的第一个非空格符开始插入
o,O	进入插入模式（Insert mode）： o 表示在光标所在位置的下一列插入新的一列，O 则表示在光标所在列的上一列插入新的一列
r,R	进入替换模式（Insert mode）： r 表示替换光标所在的那一个字符一次，R 则表示会一直替换光标所在的字符，直到按下 ESC 为止

续表

命令	作用
当输入以上按键时，在 Vim 画面的左下角会出现“--INSERT--”（表示插入的意思）或“--REPLACE--”（表示替换的意思）的字样	
ESC	退出编辑模式，切换到“一般命令模式”中

“指令列命令模式”中常用的快捷键，主要是有关文档的存储、离开以及 Vim 环境更改等命令；具体命令的详细说明如表 6-5、表 6-6 所示。

表 6-5　指令列模式的存储、离开等命令

命令	作用
:w	将编辑的数据写入硬盘文件中
:w!	若文件属性为“只读”时，强制写入该文件。不过，能不能成功写入，还是跟你对该文件的文件权限有关
:w filename	将编辑的数据存储成另一个名为 filename 的文件（类型另存文档）
:q	退出 Vim
:q!	若曾修改过文件，但想存储，使用！强制退出 Vim（放弃对文本的修改内容）
:wq	存储文件后并退出 Vim
:wq!	强制存储文件后退出 Vim
:r filename	在编辑的数据中，读入另一个文件的数据。即将 filename 这个文件内容加载到光标所在列后面
:n1,n2 w filename	将 n1 到 n2 之间的内容储存成 filename 这个文件
:! command	暂时离开 Vim 到指令列模式下执行 command 的显示结果。例如“:! ls /home”即可在 Vim 当中看到 /home 底下以 ls 输出的文件信息

表 6-6　Vim 环境的变更

命令	作用
:set nu	显示行号
:set nonu	不显示行号
: 命令	执行该命令
: 整数	跳转到该行

6.2　重定向、管道和环境变量

下章节将学习 Shell 脚本的使用方法，所以该小节起到承上启下的作用，学习管道命令符、输入输出重定向、通配符以及环境变量后便可将命令组合得更加恰当、高效。

6.2.1　输入输出重定向

输入输出重定向就是指某个指令执行后应该要出现在屏幕上的数据，给他传输到其他的地方，例如文件或者装置（例如打印机之类的）。该功能在 Linux 的文本模式下是非常重要的，尤其是如果我们想要

将某些数据储存下来，就更为有用。

首先我们需要了解文件描述符，其中包括：

标准输入（STDIN，文件描述符为 0）：默认从键盘输入，为 0 时表示从其他文件或命令的输出，使用 < 或 <<。

标准输出（STDOUT，文件描述符为 1）：是指“命令执行成功后所传回的正确的信息”，默认输出到屏幕，为 1 时表示是文件，使用 > 或 >>。

错误输出（STDERR，文件描述符为 2）：是指“命令执行失败后所传回的错误信息”，默认输出到屏幕，为 2 时表示是文件，使用 2> 或 2>>。

具体的输入输出重定向符的作用如表 6-7、表 6-8 所示。

表 6-7　输出重定向符表

符号	作用
命令 > 文件	以覆盖的方式将“正确的信息”输出到指定的文件上
命令 >> 文件	以附加的方法将“正确的信息”输出到指定的文件上
命令 2> 文件（注意 2 与 > 之间不允许空格）	以覆盖的方式将“错误的信息”输出到指定的文件上（附加到原有内容的后面）
命令 2>> 文件（注意 2 与 > 之间不允许空格）	以附加的方法将“错误的信息”输出到指定的文件上（附加到原有内容的后面）
命令 >> 文件 2>$1	将标准输出与错误输出一起写入文件中（附加到原有内容的后面）

表 6-8　输入重定向符表

符号	作用
命令 < 文件	将文件作为命令的标准输入
命令 << 分界符	从标准输入中读取，知道遇见“分界符”才停止
命令 < 文件 1 > 文件 2	将文件作为命令的标准输入并将标准输出到文件 2

注意：/dev/null 垃圾桶黑洞装置与特殊写法：当不想“错误信息”输出到任一文件或屏幕，此时将其重定向到该装置。例如：

```
# find /home -name .basshrc 2> /dev/null    # 将错误数据丢弃，屏幕上显示正确的数据
```

6.2.2　管道命令

管道命令符“|”的作用是将前一个命令的标准输出当作后一个命令的标准输入，格式为“命令 A | 命令 B”。在 Linux 下通过管道命令符连接不同命令，让数据流在各命令之间自动传递。举一个简单的例子，我们想知道 Linux 系统中当前运行的那些进程（ps 命令），并希望对它们进行排序（sort 命令）。在我们不使用管道命令符时，就必须分几个步骤来完成任务，如下所示：

```
# ps > psout.txt
# sort psout.txt > pssort.out
```

而使用管道命令符，可以让以上命令变得更简洁，如下所示：

```
# ps | sort > pssort.out
```

如果想在屏幕上分页显示输出结果，我们可以再连接第三个命令 more，将它们放在同一个命令行上，如下所示：

```
# ps | sort  | more
```

管道命令符允许连接的命令个数是没有限制的，看实际需求，比如：命令 A | 命令 B | 命令 C | … 举例说明管道符的更多应用：

例如 grep 命令进行文本搜索命令，通过匹配关键词“/sbin/nologin”找出所有被限制登录系统的用户；并统计所有不允许登录系统的用户个数如下所示：

```
# grep “/sbin/nologin” /etc/passwd | wc -l
```

用翻页的形式查看 /etc 目录中有哪些文件：

```
# ls -l /etc/ | more
```

使用非交互式设置用户密码，将 root 的密码修改为 admin：

```
# echo “admin” | passwd --stdin root
```


6.2.3 命令行通配符

在 Linux 系统中，当我们相对一类文件批量操作时，例如批量查看硬盘文件属性，或者是对某个文件进行修改，但是一时想不起文件的全程时等情况，我们便可以使用通配符来搞定所有的匹配情况。那么，在 Linux 系统中默认的 Bash 解释器中支持多种文本通配符，具体通配符的含义如表 6-9 所示。

表 6-9 通配符及对应含义

通配符	含义
*	匹配零个或多个字符
?	匹配任意单个字符
[0-9]	匹配范围内的数字
[abc]	匹配给出的任意字符

Bash 解释器还支持很多的特殊字符扩展，以下字符是非常有用的，希望能认真理解和掌握它们的用法，具体内容如表 6-10 所示。

表 6-10 特殊字符作用

字符	作用
\（反斜杠）	转移后面单个字符
‘’（单引号）	由单引号括起来的字符都视为普通字符对待

续表

字符	作用
“”（双引号）	由双引号括起来的字符，一般保留特殊字符的功能，如美元符号（$）、反引号（``）、反斜线（\）
``（反引号，即 Esc 下面的键）	由反引号括起来的字符串被当作 shell 命令执行，其标准输出结果取代整个反引号部分

6.2.4　变量与环境变量

1. 变量

变量在 Bash 环境中是非常重要的组成，接下来我们将介绍重要的环境变量、变量的取用与设定等内容。那么什么是变量呢？举例来说：n=m+2，即在等号的左边 n 就是变量，在等号右边的 m+2 就是变量的值或内容。简单来说，变量是计算机系统用于保存可变值的数据类型，我们可以直接通过变量名来提取到对应的变量值。变量是由固定的变量名与用户或系统设置的变量值两部分组成，如果需要可直接修改。在 Shell 中，使用变量之前通常并不需要事先为它们声明；当给变量赋值时，只需要使用变量名即可，该变量会根据需要被自动创建；当访问变量中所存储的内容则必须在变量名前加一个 $ 符号。

【例 1】　变量的使用：这个例子显示了引号在变量输出中的作用：

```
#！ /bin/sh
myvar= “Hi there”       # 将 Hi there 复制给 myvar
echo $myvar     # 输出为 Hi there
echo  “$myvar”          # 输出为 Hi there
echo  ‘$myvar’          # 输出为 $myvar
echo \$myvar    # 输出 $myvar
echo Enter some text    # 假设键盘输入 “Hello World”
read myvar      # 通过 read 命令将输入复制给 myvar 变量
echo  ‘$myvar’  now equals $myvar # 输出 $myvar now equals Hello word
exit 0
```

将以上代码保存为 variable 文件。使用绝对路径执行以上脚本：[root@admin~]# ./variable，便可得到相应结果，大家可以验证一下以上脚本，加深对各特殊符号在变量中的使用作用的理解。

2. 环境变量

在 Linux 系统中，环境变量是用来定义系统运行环境的一些参数，比较常用的环境变量及作用如下表所述。注意，Linux 系统中的环境变量的名称都是用大写字母表示，这是一种约定俗成的规定。Linux 常见环境变量及对应作用如表 6-11 所示。

表 6-11　Linux 常见环境变量

变量名称	作用
HOME	用户的主目录“家”
SHELL	用户使用的当前的 Shell 解释器名称
PATH	在路径中的目录查找执行文件。注意：在变量 $PATH 中目录之间用冒号“:”间隔开的

续表

变量名称	作用
EDITOR	用户默认的文本编辑器
LANG	语系数据
RANDOM	随机数字
HISTSIZE	输出的历史命令记录条数
HISTFILESIZE	保存的历史命令记录条数
MAIL	邮件信箱文件
PS1	bash 提示符

当然，我们可以使用 env 命令来查看 Linux 系统中所有的环境变量。实际上，Linux 系统能够正常运行并且为用户提供服务，需要数百个环境变量来协同工作，以上只是列举了几个常用的环境变量。

Linux 系统是多用户多任务工作模式，每个用户有各自的环境变量值，那么如何将某个用户设置的环境变量设置成其他用户也可使用呢？这里的原因主要在于变量的作用范围，属于局部变量，需要将局部变量提升为全局变量，方可应用于所有用户环境变量。

export 命令用于将局部变量提升为全局变量，格式为：“export 变量名 [= 变量值]”。此处还有一个命令非常有用，叫作命令别名设定：alias 以及 unalias，具体格式为：“别名 = ‘指令选项 ...’ ”。当我们惯用指令特别长的时候，可以使用命令别名来简化，例如：

```
#alias lm= ‘ls -al | more’      # 查询隐藏文档，并采用一页一页的方式查看
```

如果想取消命名别名的话，就使用 unalias。

```
#unalias lm
```


6.3 Shell 程序设计

Shell 是一个程序，它在用户和操作系统之间提供了一个面向行的可交互接口。用户在命令行中输入命令，运行在后台的 Shell 把命令转换成指令代码发送给操作系统。

Shell 提供了很多高级特性，使得用户和操作系统间的交互变得简单和高效。

目前，在 Linux 环境下有几种不同类型的 shell，常用的有 Bourne Again Shell（BASH）、TCSH Shell、Z-Shell 等。不同的 Shell 提供不尽相同的语法和特性，用户可以使用任何一种 Shell。在 Linus 上，BASH shell 是默认安装和使用的 Shell。本书的所有的命令都是在 BASH 下测试通过。

Shell 作为用户与 Linux 内核系统通信的媒介，定义了各种变量与参数，并且提供了诸如循环、分支等高级语言才有的控制结构特性。如何正确地使用这些功能，准确下达命令就显得尤为重要。Shell 的工作形式非为两种：一种为交互式（Interactive）：用户输入一条命令，Shell 解释并执行一条；另一种为批处理（Batch）：用户事先编写一个 Shell 脚本（Script），其中包含诸多命令，Shell 会一次执行完所有命令。在该章节之前，我们学习 Linux 命令，大致都是属于交互式；Shell 脚本是将各种命令通过逻辑语句组合而成的程序。Shell 脚本需要用到很多的 Linux 命令，结合正在表达法、管道命令以及数据流重定向

等语法规则来完成指定任务。查看系统中所有可用的 Shell 解释器，输入如下错误！未找到引用源。所示的命令如图 6-11 所示。

```
[root@bogon ~]# cat /etc/shells
/bin/sh
/bin/bash
/usr/bin/sh
/usr/bin/bash
/bin/tcsh
/bin/csh
```

图 6-11　本地所有 Shell 解释器

查看当前使用的 Shell 解释器，输入如图 6-12 所示的命令：

```
[root@bogon ~]# echo $SHELL
/bin/bash
```

图 6-12　当前 Shell 解释器

Shell 脚本为了能够让用户更灵活地完成工作需求，内设了相应的用于接收用户参数的参数变量，例如执行以下命令：

```
[root@admin~] #./Example.sh one two three four five six
```

以上命令中：$0 表示 ./Example.sh；$1 表示 one 这一参数；$2 表示 two 这一参数……以此类推。具体的其他参数变量的意义如表 6-12 所示。

表 6-12　参数变量及对应作用

参数变量	说明
$0	当前执行的 Shell 脚本的程序名
$1，$2，...	脚本程序的参数的位置变量
$#	一共有多少参数
$*	在一个变量中列出所有的参数书，各个参数之间用环境变量 IFS 的第一个字符分隔开
$@	是 $* 的一种精巧变体，它不使用 IFS 环境变量，所以即使 IFS 位空，参数也不会挤在一起（如果想访问脚本程序的参数，使用 $@ 是明智的选择）
$?	判断上一条命令是否执行成功，0 表示成功，否则表示失败

6.3.1　Shell 脚本的基本语法结构

Shell 脚本的编写使用 Vim 文本编辑器，按照命令的执行顺利依次编写，每行写一条了 Linux 命令，并且一个完整的 Shell 脚本应该包含“脚本声明”“注释信息”和“可执行语句”。“脚本声明”使用 #！表示，用于告知系统用何种 Shell 来解释；“注释信息”使用 # 表示，用于对可执行语句或程序功能做解释说明，不强制；“可执行语句”是指需执行的具体命令。先来写一个简单的 Shell 脚本，功能是显示当前的工作路径并列出当前目录下的所有文件与属性，Shell 脚本如下所示：

```
# 代码区：
```

```
# 在终端命令行，输入以下命令，创建 ShowWorkPath.sh 基本文件
# Vim ShowWorkPath.sh

# 进入 sh 脚本编辑区，输入以下脚本程序
#!/bin/bash
pwd
ls -al
```

以上已完成 Shell 脚本的编写，接下来如何执行？执行 Shell 脚本有三种方式：

直接使用脚本文件路径：./ShowWorkPath.sh；此处要注意需要修改脚本文件的权限，否则将不被允许执行。使用直接访问路径的方式提示出现错误，权限不足，如图 6-13 所示。

```
[root@bogon ~]# ./ShowWorkPath.sh
bash: ./ShowWorkPath.sh: Permission denied
```

图 6-13　直接脚本路径执行脚本文件时的错误提示

如果出现以上问题，我们需要为脚本文件设置可执行权限后才能顺利运行，输入以下命令即可：

```
# chmod u+x ShowWorkPath.sh
```

sh 脚本文件路径：sh ShowWorkPath.sh。

source 脚本文件路径：source ShowWorkPath.sh。

只要脚本文件路径没有错，“sh”或“source”命令都可以直接执行该脚本，当然也是可以直接访问脚本路径来执行。

6.3.2　流程控制语句

流程控制语句能够让 Shell 脚本根据实际工作灵活调整工作内容，例如判断系统的状态后执行指定的工作，或创建指定数据量的用户，批量修改用户密码等，这些都可以让 Shell 脚本通过流程控制语句完成。

1. 条件语句

所有程序设计语言的基础是对条件进行测试判断，并根据结果采取不同行动的能力。此处要注意 Shell 与其他编程语言的条件测试上的表现非常不同。Shell 中的 if 语句本身并不执行任何判断，它实际上接受一个命令串作为参数，然后执行这个命令串；当命令串返回值为 0，就表示为“真”，if 语句进入对应的语句块；当命令串返回值为非 0 时，则表示“假”，if 语句跳过对应的语句块。在 Shell 基本中的 if 条件语句分为单分支结构、双分支结构、多分支结构，复杂都逐级上升，但可以让 Shell 脚本更加地灵活。

（1）条件判断：test 命令和 [ ]

Shell 脚本中的布尔判断（或条件判断）命令可使用 [或 test，这两个命令的作用是一样的，只是为

了增强可读性，当使用 [命令时，需要使用符号] 来结尾。下文中我们将使用 [] 来进行条件判断，当然 test 命令也可以完成，以下脚本都能实现检查一个文件“fred.c”是否存在。

```
# 使用 test 命令进行条件判断
if test -f fred.c
then
...
fi
```

或

```
# 使用 [  ] 进行条件判断，一定要注意，[ 与命令之间一定要有空格
if [ -f fred.c ]
then
...
fi
```

此处需要注意 [符合和被检查的条件之间留出空格。如果想将 if 和 then 放在同一行上，就必须用一个分号将条件语句与 then 分隔开，如下所示：

```
#test 命令条件判断
if test -f fred.c ; then
...
if
或
#[ ] 符号进行条件判断
if [ -f fred.c ]; then
...
f
```


（2）判断类型：字符串比较、数字比较和文件有关的条件测试

test 命令或 [ ] 可以使用的条件判断类型可以归纳为 3 类：字符串比较、数字比较和文件有关的条件测试，表 6-13 ~表 6-15 描述了这 3 种条件类型具体使用方法。

表 6-13　字符串比较方法

字符串比较	结果
str1 = str2	如果两个字符串相同则结果为真，返回 0，否则返回非 0
str1! = str2	如果两个字符串不同则结果为真，返回 0
-n str	如果字符串不为空则结果为真，返回 0
-z str	如果字符串为 null（一个空串）则结果为真

表 6-14　数值比较方法

数字比较	结果
表达式 1 -eq 表达式 2	如果两个表达式相等则结果为真
表达式 1 -ne 表达式 2	如果两个表达式不等则结果为真
表达式 1 -gt 表达式 2	如果表达式 1 大于表达式 2 则结果为真
表达式 1 -ge 表达式 2	如果表达式 1 大于等于表达式 2 则结果为真
表达式 1 -lt 表达式 2	如果表达式 1 小于表达式 2 则结果为真
表达式 1 -le 表达式 2	如果表达式 1 小于等表达式是 2 则结果为真
! 表达式	如果表达式为假则结果为真，反之亦然

表 6-15　文件条件测试方法

文件条件测试	结果
-d file	表示文件是一个目录则结果为真
-e file	如果文件存在则结果为真。要注意的是，历史上 -e 选项不可移植，所以通常使用的是 -f 选项
-f file	如果文件是一个普通文件则结果为真
-g file	如果文件的 set-group-id 位被设置则结果为真
-r file	如果文件可读则结果为真
-s file	如果文件的大小不为 0 则结果为真
-u file	如果文件的 set-user-id 位被设置则结果为真
-w file	如果文件可写则结果为真
-x file	如果文件可执行则结果为真

（3）if 条件语句

• 单分支结构，仅用 if、then、fi 关键词组成，只有当条件成立后才执行 then 后面的语句。具体语法结构如下：

```
if 条件测试操作
then 命令序列
fi
```

【实验 01】

单分支 if 语句 —— 判断目录是否存在，若不存在则自动创建。编写 Shell 脚本如下：

```
# 首先使用命令行创建脚本文件
# Vim Example.sh

# 进入脚本文件编辑
#!/bin/bash
DIR=”/media/cdrom”
if [! -e $DIR]
then
```

```
mkdir -p $DIR
fi
```

执行后默认没有回显，可动手添加 echo 语句显示创建过程。

• 双分支结构是由 if、then、else、fi 关键词组成，做出条件成立或条件不成立的判断，执行相应操作。具体语法结构如下：

```
if 条件测试操作
then 命令序列 1
else 命令序列 2
fi
```

【实验 02】

双分支 if 语句 —— 判断指定主机能否 ping 通，根据返回结果分别给出提示或警告。

```
[root@admin~]# Vim Example.sh
#!/bin/bash
Ping -c 3 -i 0.2 -W 3 $1 &> /dev/null  # 为了减少用户的等待时间，需要为 ping 命令追加 -c 参数代码发送数据包的个数，-i 代表每 0.2 秒发送一个数据包，-W 则为 3 秒即超时，$1 为用户输入的第一个参数（IP 地址）
if [$? -eq 0]       # $？为上一条命令的执行结果，判断是否等于 0（成功）
then
echo  “Host $1 is up.”
else
echo  “Host $1 is down.”
fi
```

给予脚本可执行权限，否则请用 sh 或 source 命令执行：

```
[root@admin~]# chmod u+x Example.sh
```

参数为要检测书籍 IP 地址，根据返回值判断是否在线：

```
[root@admin~]# ./Example.sh 192.168.10.10
Host  192.168.10.10 is up.
```

多分支结构相对就比较复杂，由 if、then、else、elif 、fi 关键词组成，根据多种条件成立的可能性执行不同的操作。具体语法结构如下：

```
if 条件测试操作 1
then 命令序列 1
elif 条件测试操作 2
then 命令序列 2
```

```
else
命令序列 3
fi
```

【实验 03】

对用户输入的分数进行等级判定。

```
[root@admin~]# Vim Example.sh
#!/bin/bash
read -p “Enter your score (0-100)：” GRADE  # 使用 read 命令让用户为 GRADE 变量赋值
if [$GRADE -ge 85] && [$GRADE -le 100]
then
echo “$GRADE is 优秀”
elif [$GRADE -ge 70] && [$GRADE -le 83]
then
echo “$GRADE is 合格”
else
echo “$GRADE is 不合格”
fi
```

举例说明：多分支 if 语句 —— 判断用户输入的分数在哪个区间内，然后判定为优秀、合格或不合格。

按照以前的执行方法，完成后续操作，输入 80，查看输出结果。

并请修改以上 Shell 脚本中添加判断语句，将所有小于 0 分或大于 100 分的输入都予以警告。

（4）case 条件语句

case 条件语句可以根据变量的不同取值，分别执行不同的命令动作。具体语法结构如下所示：

```
case $ 变量名称 in
模式 1）  # 每个变量内容建议用双引号括起来，）属于关键词
commands
;;  # 每个类别结尾使用两个连续的 ;;
模式 2)
commands
;;
.....
*)  # 最后一个变量的内容都会用 * 来代表所有其他值
commands
;;
esac
```

基于以上语法结构，需要注意构成中的关键词，特别注意右括号）、双分号；以及 * 这些特别的关键词，“；”用来代表该命令块的结束。

【实验 04】

举例说明以上 case 条件语句的用法：提示用户输入一个字符，判断该字符是字母、数字或特殊字母，具体脚本如下所示。

```
[root@admin~]# Vim Example.sh
#!/bin/bash
read -p “请输入一个字符，并按 Enher 键确认：” KEY
case “$KEY” in
z] | [A-Z])
echo “你输入的是字母。”
;;
[0-9])
echo “你输入的是数字。”
;;
*)
echo “你输入的是空格、功能键或其他控制符。”
;;
esac
```

通过执行以上脚本程序，来验证 case 条件语句的执行流程，验证过程如下所述：

```
[root@admin~]# chmod u+x Example.sh
[root@admin~]# ./Example.sh
请输入一个字符，并按 Enter 键确认：6
你输入的是数字。
[root@admin~]# ./Example.sh
请输入一个字符，并按 Enter 键确认：p
你输入的是字母。
[root@admin~]# ./Example.sh
请输入一个字符，并按 Enter 键确认：^[[
你输入的是空格、功能或其他控制字符。
```

此处再次强调以下，Shell 脚本执行的方式有三种：

• 使用绝对路径执行，但需要修改文件的读写属性，具体参考：

chmod u+x file.sh

./file.sh

• 直接使用 sh 命令，即 sh file.sh 即可；

• 直接使用 source 命令，即 source file.sh 即可。

2. 循环语句

循环结构用于反复执行一段语句，是程序设计中的基本结构之一。Shell 中的循环结构有三种：while、until 和 for。下面逐一介绍这 3 中循环语句。

（1）while 语句

while 语句重复执行命令，直到测试条件为假。该语句的基本结构如下，注意，commands 可以是多条命令语句组成的语句块。

```
while test-commands
do
    commands
done
```

运行时，Shell 首先检查 test-commands 是否为真（注意，是返回 0），如果是，就执行命令 commands。commands 执行完成后，Shell 再次检查 test-commands，如果为真，就再次执行 commands……这样的“循环”一直持续到条件 test-commands 为假（非 0）。为了更好地说明这一过程，下面这个脚本是使用 Shell 完成计算 1+2+3+…+100。

【实验 05】写 Shell 脚本程序完成计算 1+2+3+…+100。

```
[root@admin~]# Vim One2hundred_1.sh
#!/bin/bash
sum=0
number=1
while test $number -le 100      # 此处可替换 [ $number -le 100 ]，表示测试条件为 number 数值小于或等于 100 的时候才执行 do 和 done 之间命令块。
do
sum=$[ $sum + $number ]
let number=$number+1
done
echo  “The summary is $sum”
```

简单地分析以下这段小程序。在程序的开头，首先将变量 sum 和 number 初始化为 0 和 1，其中变量 sum 保存最终结果，number 则用于保存每次累加的数。测试条件“$number -le 100”告诉 Shell 仅当 number 中的数值小于或等于 100 的时候才执行包括在 do 和 done 之间的命令。注意，每次循环之后都将 number 的值加上 1，循环在 number 达到 101 的时候结束。

注意：while 语句的测试条件不一定都要使用 test（或 [ ]）命令。在 Linux 中，命令都是有返回值的。例如，read 命令在接收到用户的输入时就返回 0，如果用户用【Ctrl】+【D】快捷键输入一个文件结束符，那么 read 命令就返回一个非 0 值（通常是 1）。利用这个特性，可以使用任何命令来控制循环。

【实验 06】

下面这段脚本从用户处接收一个大于 0 的数值 n，并且计算 1+2+3+…+n。

```
[root@admin~]# Vim One2n.sh
#!/bin/bash
echo -n   “Enter a number(>0):”
while read n
do
sum=0
count =1
if [ $n -gt 0]
then
while [ $count -le $n ]    # 此处注意 [  ] 一定要和命令之间有空格
do
$sum=$[ $sum + $count ]
let count=$count+1
done
echo   “The summary is $sum”
else
echo   “Please enter a number greater than 0”
fi
echo -n  “Enter number(>0):”
done
```

这段脚本不停地读入用户输入的数字，并判断这个数是否大于 0。如果是，就计算从 1 一直加到这个数的和。如果不是，就提示一条信息，然后继续等待用户的输出，知道用户输入快捷键【Ctrl】+【D】（代表文件结束）结束输入。下面先显示了这个脚本的执行效果。

```
[root@admin~]# ./One2n
Enter a number(>0):100
The summary is 5050
Enter a number(>0):50
The summary is 1540
Enter a number(>0):-1
Please enter a number greater than 0
Enter a number(>0):<Ctrl+D>          # 这里按下【Ctrl】+【D】快捷键
```

（2）until 语句

until 是 while 语句的另一种写法——除了测试条件相反，其基本语法结构如下：

```
until 条件测试
do
    commands
done
```

单从字面上理解，while 说的是当“测试条件为真”（返回值为 0）时，就执行 commands；当“测试条件为假”（返回值为非 0）时，便停止执行。而 until 说的是当“测试条件为假”（返回值为非 0），执行 commands；直到“测试条件为真”（返回值为 0），便停止执行”。while 语句相对好理解，until 语句有点逆向思维，但愿读者没有被搞糊涂。

【实验 07】下面通过使用 until 语句完成上各问题：计算 1+2+3+…+n 来帮助大家理解它们之间的区别。

```
[root@admin~]# Vim One2hundred_2.sh
#!/bin/bash
sum=0
number=1
until ! [ $number -le 100]       #-le 表示小于等于，这句话等价于 while [ $number -le 100]
do
sum=$[ $sum + $number ]
let number=$number+1
done
echo  “The summary is $sum”
```

（3）for 语句

使用 while 语句可以完成 Shell 编程中的所有循环任务，但有些时候用户希望从列表中逐一取一系列的值（例如去除用户提供的参数），此时使用 while 和 until 就显得不太方便。Shell 提供了 for 语句来循环处理一组值，这组值可以是任意字符串的集合。for 的基本语法如下：

```
for 变量名 in 取值列表
do
commands
done
```

【实验 08】举例说明：使用 for 语句完成统计当前目录下的文件的个数。

```
[root@admin~]#Vim flies_count.sh
#!/bin/bash
count=0
for file in `ls`     # 此处注意是反引号，用于执行命令”ls”
do
if ! [ -d $file ]    #-d 判断当前 file 是否为文件
```

```
then
let count=$count+1
fi
done
echo  “There are $count files”
```

这段脚本每次从 ls 命令生成的文件列表中取出一个值赋值给 file 变量中，判断 file 变量是否为文件，如果是便给计数器增加 1；直到文件列表完全遍历。

3. 函数

当编写大型的 Shell 脚本程序时，我们会考虑在 Shell 中定义函数来实现脚本的简洁以及复用。要定义一个 Shell 函数，只需写出它的函数名，然后是一对空括号，然后将函数中的语句放在一对大括号中，其函数定义语法如下所示：

```
function name(){        # 函数定义的简化写法，可省略 function 关键字
statements
[return value]
}
```

对各部分的解释说明：

• function 是 Shell 中的关键字，专门用来定义函数。

• name 是函数名。

• statements 是函数要执行的代码，也就是语句块。

• return value 表示函数的返回值，其中 return 是 Shell 的关键字，专门用于函数存在返回值，这一部分可以写也可以不写，根据实际需要。

• 由 {} 包围的部门称为函数体，调用一个函数，实际上是执行函数体中的脚本。

函数的调用：

调用函数时可以给它传递参数，也可以不传递，如果不需要传递参数，直接给出函数名字即可完成函数的调用，如下脚本所示：

```
function_nam
```

如果传递参数，那么多个参数之间以空格分隔，具体使用见以下脚本演示：

```
function_name 参数 1 参数 2 ...
```

注意：不管是哪种形式，函数名字后面都不需要带括号。

【实验 09】一个简单的函数。

```
[root@admin~]#Vim function_demo1.sh
#!/bin/bash
```

```
foo(){
echo “函数正在被执行!”
}
echo “Shell 脚本开始执行”
foo        # 直接使用函数名完成函数调用
echo “Shell 脚本执行结束”
```

运行这个脚本程序，查看函数运行结果。注意：我们必须在调用一个函数之前先对它进行定义。因此所有脚本程序都是从顶部开始执行，所以只要把所有函数定义都放在任何一个函数被调用之前，就可以保证所有的函数在被调用之前就被定义了。

当一个函数被调用时，脚本程序的位置参数（$*、$@、$#、$1、$2 等，具体意义请参考之前章节内容）会被替换为函数的参数。除此之外，我们还可以通过 return 命令让函数返回数字值。让函数返回字符串值的常用方法是让函数将字符串保存在一个变量中，该变量便可在函数结束之后被使用。

本章小结

重点介绍 Vim 文本编辑器的三种工作模式及对应工作模式下的常用命令，帮助用户快速完成文本编辑工作。

重点讲述了基于 Vim 文本编辑器、重定向相关指令、管道的使用、命令行通配符、变量与环境变量等理论知识学习，为我们后续的 Shell 程序设置做了很好的预备知识的储备。

后续重点学习 Shell 程序设计中的基础语法结构、流程控制语句以及函数的创建与使用，为我们能够在 Linux 系统上进行 shell 的灵活使用奠定了坚定的基础。

习题

一、简答题

1. 用 Vim 打开某个文件后，要将光标在第 34 列向右移动 15 个字符，应该在一般指令模式中下达什么指令？

2. 在 Vim 的环境中，如何将目前正在编辑的文件另存新档名为 newFilename?

3. 在 Linux 底下最常使用的文本编辑器为 Vim，请问如何进入编辑模式?

4. 在 Vim 编辑器中，如何由编辑模式切换到一般指令模式?

5. 在 Vim 的一般指令模式中如何搜寻 string 这个字符串？

6. 在 Vim 的一般指令模式中，如何取代 word1 成为 word2，而若需要使用者确认机制，又该如何？

7. 在 Vim 的一般指令模式中，如何存档、离开、存档后离开、强制存档后离开？

8. 如何显示 HOME 这个环境变量？

9. 如何得知当前的所有变量与环境变量的设定值？

10. 如何定义一个变量名为 name 内容为 It's my name?

11. 转义字符 \ 有什么用途？

12. 如何展示 /bin 文件夹里任何文件名以 a 开头的文件的详细资料？

二、编程题

1. 请编写 Shell 脚本文件，完成 1+2+3+.... 一直累加到用户输入的数字为止。

2. 在 /etc/passwd 文件里面，以 : 来分隔，第一栏为账号名称。请写一段 Shell 脚本，可以将 /etc/password 的第一栏取出，而且每一栏都以一行字符串例如：The 1 account is "root" 来显示，其中 1 表示行数。

✔ 综合实训

一、实训题目

Vim 编辑器的使用

二、实验目的

学习使用 Vim 编辑器的使用，包括编辑、显示及加工处理文本等操作。

三、实验内容

- 进入、退出 Vim。
- 利用文本插入方式是建立一个文件。
- 在新建的文本文件上移动光标位置。
- 对文件执行删除、复原、修改、替换等操作。

1. 预备知识

（1）Vim 简介。系统管理员的重要工作就是修改与设定某些重要软件的配置文件，因此至少得学会一种以上的命令界面文件编辑器。在所有 Linux 发行版本上都会有的一套文字编辑器就是 Vi，而且很多软件默认的也是使用 Vi 作为它们的编辑接口。

Vim 是高级版本的 Vi，Vim 不但可以用不同颜色显示文字内容，还能够进入诸如 Shel 脚本、C 语言等程序的编辑功能。

（2）Vim 的使用。基本上 Vim 分为三种工作模式，分别是一般指令模式、编辑模式、指令列命令模式。这三种模式的作用分别是：

一般指令模式：以 Vim 打开一个文档就直接进入一般指令模式（这是默认的模式）。在这个模式中，可以使用“上下左右”键来移动光标，可以使用“删除字符”或“删除整行”来处理文档内容，也可以使用“复制、粘贴”来处理文件数据。

编辑模式：在一般模式中可以进行删除、复制、粘贴等操作，但是无法编辑文件内容。要按下“i、I、o、O、a、A、r、R”等任何一个字母后才会进入编辑模式。而如果要回去一般指令模式时，则必须按下【Esc】按键即可退出编辑模式。

指令列命令模式：在一般指令模式时，输入：/？三个中的任何一个按钮，就可以将光标移动到最底下那一行。在这个模式当中，可以提供查找数据的操作。读取、保存、大量替换字符、离开 Vim、显示行号等操作是在此模式中完成的。

2. 实验步骤

具体的操作要求步骤如下：

（1）进入 Vim。

（2）请在 /temp 这个目录下建立一个名为 Vimtest 目录。

（3）进入 Vimtest 这个目录当中。

（4）将 /etc/man_db.conf 复制到本目录底下。

（5）使用 Vim 打开目录下的 man_db_conf 这个文件。

（6）在 Vim 中设定一下行号。

（7）将光标移动到第 43 列，向右移动 59 个字符，请问你看到的是什么文字？

（8）将光标移动到第一列，并向下搜索“gzip”这个字符串，请问它在第几列？

（9）接下来，请将 29 到 41 列之间的“man”替换为“MAN”，并且一个一个挑选是否需要修改，如何下达指令？如果在挑选过程中一直按“y”，结果会在最后一列改变了几个 man?

（10）修改完之后，突然反悔，要全部复原，有哪些方法？

（11）请复制 66 到 71 之间这 6 列的内容，并且粘贴到最后一列之后。

（12）113 到 128 列之间的开头为 # 符号的批注数据去掉，要如何删除？

（13）将这个文件另存为一个名为“man.test.confi”的文件。

（14）将光标移动到第 25 列，并且删除 15 个字符，结果出现的第一个单字是什么？

（15）在第一列新增一列，该列内容为“I am a student...”。

（16）存储后离开。

整个步骤可以如下操作：

```
mkdir /tmp/Vimtest
cd /tmp/Vimtest
cp /etc/man_db.conf
Vim man_db.conf
:set nu    # 然后你会在 Vim 的画面中看到左侧出现数字即为行号。
```

先按下“43G”，再按下“59→”会看到“as”这个单词在小括号内。

先执行“1 G”或“gg”后，直接输入“/gzip”，则光标会去到第 93 列。

直接按下“:29,41 s/man/MAN/gc”即可，若已知按“y”最终会出现在第 13 列替换 13 个字符串。

方法一：一直按“u”回复到原始状态；方法二：使用不储存离开“:q!”。

“66G”然后再“6yy”之后最后一列会出现“man.test.config”[New].. 的字样。

因为 113~128 共 16 列，因此“113G”→“16dd”就能删除 16 列，此时你会发现游标所在 113 列的地方变成 [#Flags.] 开头。

“:w man.test.config ”，最后一列便出现“man.test.config”[New].. 的字样。

“25G”之后，再给他“15x”即可删除 15 个字符，出现“tree”的字样。

先执行“1G”去掉第一列，然后按下大写“O”便新增一列且进入插入命令模式；开始输入“I am a student....”后，按下【Esc】切回一般指令模式等待后续工作。

“:wq”。

四、实验总结和体会

第七章

CentOS 7 软件包管理

内容简介

- Linux 系统下的软件包可以分为源码包和二进制包，源码包一般包含多个文件，为了方便发布，通常会将源码包做打包压缩处理，相比源码包，二进制包是在软件发布时已经进行过编译的软件包，所以安装速度比源码包快得多。使用 RMP 包安装软件具有以下优点：1. 包管理系统简单，只通过几个命令就可以实现软件包的查询、安装、卸载、升级等；2. 安装速度比源码包安装快得多。本章将从使用 rpm 命令管理 RPM 包、使用 RPM 软件包管理器、安装 gcc 并编写 C 语言文件以及 yum 的使用来了解 CentOS 7 软件包的管理。

学习要求

- 通过本章的学习，要求掌握如何使用 rpm 命令管理 RPM 软件包，掌握如何使用 RPM 软件包管理器，掌握安装 gcc 并编写 C 语言文件，了解 yum 的相关使用。

7.1 使用 rpm 命令管理 RPM 包

RPM 包的命名须遵守统一的命名规则，用户通过名称即可直接获取这类包的版本、适用平台等信息。

7.1.1 RPM 软件包参数说明

RPM 包命名的一般格式如下：

软件包名 – 软件版本号 – 发布次数 – 适合软件运行的硬件平台 – 软件包扩展名

如：bash-4.1.2-15.el6_4.x86_64.rpm，

rpm 命令是 RPM 软件包的管理工具。rpm 命令格式如下：

```
rpm [ 选项 ] [ 参数 ]
```

主要选项参数如表 7-1 所示。

表 7-1　rpm 命令主要选项参数

参数	说明
-a	查询所有已安装的软件包，通常和 qa 一起使用
-b	设置软件包的完成阶段，并指定套件档的文件名称
-c	只列出组态配置文件，本参数需配合”-l”参数使用
-d	只列出文本文件，本参数需配合”-l”参数使用
-e	卸载已安装的软件包
-f	查询已安装软件包的包全名，反向使用，后跟文件
-h	显示安装的进度，安装时以 ##### 输出
-i	安装 rpm 包
-l	查询软件包的安装位置
-p	查询未安装软件包的相关信息，后跟软件全名
-q	使用询问模式，当遇到任何问题时，rpm 指令会先询问用户
-R	显示软件包的依赖性信息
-s	显示文件状态，本参数需配合”-l”参数使用
-U	升级，一般和 vh 组合使用
-v	显示指令执行过程，提供更加详细的安装系统

7.1.2　RPM 软件包的查询

rpm 命令还可用来对 RPM 软件包做查询操作，具体包括：查询软件包是否已安装、查询系统中所有已安装的软件包、查看软件包的详细信息、查询软件包的文件列表、查询某系统文件具体属于哪个 RPM 包。

使用 rpm 做查询命令的格式如下：

```
rpm [ 选项 ] [ 查询对象 ]
```

1. 查询软件包是否已安装

使用 rpm 查询软件包是否已安装的命令格式如下：

```
rpm -q 软件包名
```

【例 1】　获取系统中安装的 mysql 软件包全名，从中可以获得当前软件包的版本信息：

```
# rpm -q mysql
```

2. 查询系统中所有安装的软件包

使用 rpm 查询系统中所有安装的软件包的命令格式如下：

```
rpm -qa 软件包名
```

【例 2】 查询 httpd 是否已经安装

```
# rpm -qa httpd
#                    # 没有输出软件包的信息，就说明没有安装
```

【例 3】 查询 lrzsz 是否已经安装

```
# rpm -qa lrzsz
lrzsz-0.12.20-36.el7.x86_64    # 安装了 lrzsz 命令，即可输出软件包的全名
```

此外也可以查找出所有安装过的包含某个字符串 lrzsz 的软件包：

```
# rpm -qa | grep lrzsz
lrzsz-0.12.20-36.el7.x86_64     # 通过过滤命令也可以查询是否安装
```

相比于“rpm –q 软件包名”命令，这种方式可以找到含有软件包名的所有软件包。

3. 查询软件包的详细信息

使用 rpm 查询软件包的详细信息，包括：软件包名、版本和厂商、发行版本和建立时间、安装时间、组和源 RPM 包文件名、软件包说明等，rpm 命令格式如下：

```
rpm -qi 软件包名
```

【例 4】 查询 zsh 软件包的详细信息：

```
# rpm -qi zsh
Name : zsh
Version : 5.0.2
Release : 28.el7
Architecture: x86_64
Install Date: Fri 7 Mar 2019 11:52:01 AM CST
Group : System Environment/Shells
Size : 5855982
License : MIT
Signature : RSA/SHA256, Fri 11 Aug 2017 04:28:17 AM CST, Key ID 24c6a8a7f4a80eb5
Source RPM : zsh-5.0.2-28.el7.src.rpm
Build Date : Wed 02 Aug 2017 06:52:37 PM CST
Build Host : c1bm.rdu2.centos.org
```

```
Relocations : (not relocatable)
Packager : CentOS BuildSystem http://bugs.centos.org
Vendor : CentOS
URL : http://zsh.sourceforge.net/
Summary : Powerful interactive shell
……
```

此外还可以查询未安装的软件包的详细信息，命令格式如下：

```
rpm -qip 软件包全名
```

这里使用的是软件包的全名，且未安装的软件包必须使用“绝对路径 + 软件包全名”的方式方可确定软件包。

4. 查询软件包的文件列表

rpm 软件包通常采用默认路径安装，各安装文件会分门别类安放在适当的目录文件下。使用 rpm 命令可以查询到已安装软件包中包含的所有文件及各自安装路径，命令格式为：

```
rpm -ql 软件包名
```

【例 5】　查询 zsh 会生成哪些文件：

```
# rpm -ql zsh | grep bin
/bin/zsh
/usr/share/zsh/5.0.2/functions/_bind_addresses
/usr/share/zsh/5.0.2/functions/_bindkey
/usr/share/zsh/5.0.2/functions/_combination
```

一般安装的软件包会将一些文件存放在固定目录下：

/etc 配置文件

/bin /sbin 可执行文件

/lib /lib64 库文件

/usr/include 头文件

/usr/share/doc 使用手册帮助文件

/usr/share/man man 手册

rpm 命令还可以查询未安装软件包中包含的所有文件以及打算安装的路径，由于软件包还未安装，因此需要使用“绝对路径 + 包全名”的方式才能确定包。命令格式如下：

```
# rpm -qlp 软件包全名
```

5. 查询系统文件属于哪个 RPM 包

rpm –ql 命令是通过软件包查询所含文件的安装路径，rpm 还支持反向查询，即查询某系统文件所

属哪个 RPM 软件包，命令格式如下：

```
# rpm -qf 系统文件名
```

只有使用 RPM 包安装的文件才能使用该命令，手动方式建立的文件无法使用此命令。

【例 6】 查询 ls 命令所属的软件包：

```
# rpm -qf /bin/ls
coreutils-8.4-19.el6.i686
```

6. 查询软件包的依赖关系

使用 rpm 命令安装 RPM 包，需考虑与其他 RPM 包的依赖关系，rpm -qR 命令就用来查询某已安装软件包依赖的其他包，该命令的格式为：

```
# rpm -qR 软件包名
```

【例 7】 查询 apache 软件包的依赖性：

```
# rpm -qR httpd
/bin/bash
/bin/sh
/etc/mime.types
/usr/sbin/useradd
apr-util-ldap
chkconfig
……
```

同样，在此命令的基础上增加“-p”选项，需要使用“绝对路径 + 包全名”的方式即可实现查找未安装软件包的依赖性：

```
# rpm -qRp 软件包全名
```

【例 8】 bind 软件包尚未安装（绝对路径为：/mnt/cdrom/Packages/bind-9.8.2-0.10.rc1.el6.i686.rpm），查看此软件包的依赖性：

```
# rpm -qRp /mnt/cdrom/Packages/bind-9.8.2-0.10.rc1.el6.i686.rpm
/bin/bash
/bin/sh
bind-libs = 32:9.8.2-0.10.rc1.el6
chkconfig
chkconfig
config(bind) = 32:9.8.2-0.10.rc1.el6
……
```

7.1.3　RPM 软件包的安装

RPM 软件包的安装可以使用程序 rpm 来完成，rpm 命令格式如下：

```
rpm -ivh 软件包全名
```

有些软件包是以 .rpm 结尾的，这类软件包是包含了源代码的 rpm 包，在安装时需要进行编译。

【例 9】　用 rpm 命令安装 apache 软件包：

```
# rpm -ivh zsh-5.0.2-28.el7.x86_64.rpm
warning: zsh-5.0.2-28.el7.x86_64.rpm: Header V3 RSA/SHA256 Signature, key ID f4a80eb5: NOKEY
Preparing… ################# [100%]
Updating / installing…
1:zsh-5.0.2-28.el7 ################# [100%]
```

直到出现两个 100% 才是真正的安装成功，第一个 100% 仅表示完成了安装准备工作。

此命令还可以一次性安装多个软件包，仅需将包全名用空格分开即可，如下所示：

```
# rpm -ivh a.rpm b.rpm c.rpm
```

7.1.4　RPM 软件包安装可能出现的问题

安装过程中可能出现下面的警告或者提示：

```
… conflict with …
```

可能是要安装的包里有一些文件可能会覆盖现有的文件，缺省时这样的情况下是无法正确安装的，可以用“rpm force –i”强制安装即可。

```
… is needed by …
… is not installed …
```

此包需要的一些软件你没有安装可以用 rpm nodeps –i 来忽略此信息，也就是说 rpm –i force nodeps 可以忽略所有依赖关系和文件问题，什么包都能安装上，但这种强制安装的软件包不能保证完全发挥功能。

7.1.5　RPM 软件包的卸载

RPM 软件包的卸载要考虑包之间的依赖性。软件包卸载和拆除大楼是一样的，本来先盖的 2 楼，后盖的 3 楼，那么拆楼时一定要先拆除 3 楼。例如，我们先安装的 httpd 软件包，后安装 httpd 的功能模块 mod_ssl 包，那么在卸载时，就必须先卸载 mod_ssl，然后卸载 httpd，否则会报错。

RPM 软件包的卸载使用如下命令即可：

```
# rpm -e 软件包名
```

【例 10】 使用 rpm 命令卸载 zsh 包：

```
# rpm -qa zsh        # 查询是否安装 zsh 包
zsh-5.0.2-28.el7.x86_64
# rpm -e zsh         # 卸载 zsh 包
# rpm -qa zsh        # 再次查询是否安装
#                    # 没有任何输出说明已经卸载成功
```

7.1.6 RPM 软件包的升级

用如下命令即可实现 RPM 包的升级：

```
# rpm -Uvh 软件包全名
```

-U 选项的含义是：如果该软件没安装过则直接安装；若没安装则升级至最新版本。

【例 11】 使用 rpm 命令将 zsh-5.0.2-28.el7.x86_64.rpm 软件包升级：

```
# rpm -Uvh zsh-5.0.2-28.el7.x86_64.rpm
warning: zsh-5.0.2-28.el7.x86_64.rpm: Header V3 RSA/SHA256 Signature, key ID
f4a80eb5: NOKEY
Preparing… ################## [100%]
Updating / installing…
1:zsh-5.0.2-28.el7 ################## [100%]
```

7.1.7 RPM 软件包的验证

Linux 系统中装有大量的 RPM 包，且每个包都包含了大量的安装文件，因此，为了能及时发现文件误删、误修改文件数据、恶意篡改文件内容等问题，Linux 提供了 RPM 包校验和 RPM 包数字证书校验两种验证方式。

1. RPM 包校验

其实就是将已安装文件和 /var/lib/rpm/ 目录下的数据库内容进行比较，确定文件内容是否被修改，可使用的命令格式分为以下 3 种：

```
# rpm -Va
```

-Va 表示校验系统中已安装的所有软件包。

```
# rpm -V 已安装的软件包名
```

-V 表示校验指定 RPM 包中的文件。

```
# rpm -Vf 系统文件名
```

-Vf 表示校验某个系统文件是否被修改。

【例 12】　验证 apache 软件包中所有的安装文件是否被修改：

```
# rpm -V httpd      # 执行后无任何提示信息，表明所有用 apache 软件包安装的文件均未改动过，还和从原软件包安装的文件一样
```

接下来尝试对 apache 的配置文件 /etc/httpd/conf/httpd.conf 做适当修改，修改格式如下：

```
#Vim /etc/httpd/conf/httpd.conf
……
DirectoryIndex index.html index.html.var index.php
```

为防止崩溃，这里尝试修改 apache 的默认网页文件，按照以上格式对文件进行修改后保存退出，再次使用“rpm -V”命令对 apache 软件包进行验证：

```
# rpm -V httpd
S.5....T. c /etc/httpd/conf/httpd.conf
```

可以看到，结果显示了文件被修改的信息。该信息可分为以下 3 部分：

（1）最前面的 8 个字符（S.5....T）都属于验证信息，各字符的具体含义如下：

S：文件大小是否改变。

.：若相关项没发生改变，用“.”表示。

5：文件 MD5 校验和是否改变，或看成文件内容是否改变。

T：文件的修改时间是否改变。

D：设备的主从代码是否改变。

G：文件的属组是否改变。

L：文件路径是否改变。

M：文件的类型或文件的权限是否改变。

U：文件的所有者是否改变。

（2）被修改文件类型，大致可分为以下几类：

c：配置文件（configuration file）。

d：普通文档（documentation）。

l：授权文件（license file）。

r：描述文件（read me）。

被修改文件所在绝对路径，包含文件名。

由此，S.5....T. c /etc/httpd/conf/httpd.conf 表达的完整含义是：配置文件 httpd.conf 的大小、内容、修改时间被人为修改过。

值得注意的是，并非所有对文件做修改的行为都是恶意的。通常情况下，对配置文件做修改是正常的，比如说配置 apache 就要修改其配置文件，而如果验证信息提示对二进制文件做了修改，除非是自己故意修改的，否则就需要特别留意。

2. RPM 包数字证书校验

RPM 包校验方法只能用来校验已安装的 RPM 包及其安装文件，如果 RPM 包本身就被动过手脚，此方法将无法解决问题，则需要使用 RPM 数字证书验证方法。RPM 包校验其实就是将现有安装文件与最初使用 RPM 包安装时的初始文件进行对比，如有改动则提示给用户，因此这种方式无法验证 RPM 包本身被修改的情况。

数字证书，又称数字签名，由软件开发商直接发布。Linux 系统安装数字证书后，若 RPM 包做了修改，此包携带的数字证书也会改变，将无法与系统匹配成功，则软件无法安装。可以将数字证书想象成自己的签名，是不能被模仿的（厂商的数字证书是唯一的），只有我认可的文件才会签名（只要是厂商发布的软件，都符合数字证书验证）；如果我的文件被人修改了，那么我的签名就会变得不同（如果软件改变，数字证书就会改变，从而通不过验证。当然，现实中人的手工签名不会直接改变，所以数字证书比手工签名还要可靠）。

使用数字证书验证 RPM 包的方法具有如下两个特点：

必须找到原厂的公钥文件，然后才能进行安装。

安装 RPM 包会提取 RPM 包中的证书信息，然后和本机安装的原厂证书进行验证，如果验证通过，则允许安装；如果验证不通过，则不允许安装并发出警告。

安装数字证书的命令如下：

```
# rpm -import 数字证书系统位置
```

数字证书安装完成后，可使用如下命令进行验证：

```
# rpm -qa|grep gpg-pubkey
```

在装有数字证书的系统上安装 RPM 包时，系统会自动验证包的数字证书，验证通过则可以安装，反之将无法安装，即系统会报错。数字证书本身也是一个 RPM 包，因此可以用 rpm 命令查询数字证书的详细信息，也可以将其卸载。

查询数字证书详细信息的命令如下：

```
# rpm -qi gpg-pubkey-c105b9de-4e0fd3a3
```

卸载数字证书可以使用 –e 选项，命令如下：

```
# rpm -e gpg-pubkey-c105b9de-4ead3a3
```

7.2 使用 RPM 软件包管理器

RPM，Redhat Package Manager，使用 RPM，用户可以自行安装和管理 Linux 系统上的应用程序和系统工具。RPM 可以让用户直接安装软件包，并可替用户查询是否已经安装了有关的库文件；在用 RPM 删除程序时，会询问用户是否要删除有关的程序；如果使用 RPM 来升级软件，RPM 会保留原有的

配置文件，这样用户就不用重新配置新的软件了。

7.2.1　打开软件包管理器

只有系统管理员有权限使用 RPM 软件包管理器，因此先使用 su 命令进入 root 账号，然后就可使用 rpm 命令对 RPM 软件进行安装、查询、验证、删除等管理。

7.2.2　添加删除软件

1. 添加软件相关参数

-h：以 # 来表示安装进度

--percent：以百分比形式表示安装进度

-v：显示安装过程的详细信息

-vv：显示更详细的安装过程

--test：不执行真正的安装过程，仅报告依赖关系及冲突信息。此为调试模式

--nodeps：忽略依赖关系，能成功安装，但未必能运行成功，不建议使用

--replacepkgs：重新安装并覆盖原有文件

--force：强制安装

--noslgnature：不检查包签名信息，不检查来源合法性

--nodigest：不检查包完整性信息

2. 删除软件相关参数

--nodeps：忽略依赖关系

--test：测试卸载，dry-run 模式

--allmatches：如果一个程序包同时安装了多个版本，则此选项会一次性全部卸载

7.2.3　其他软件包管理器

软件包管理器的作用是提供在操作系统中安装、升级、卸载需要的软件的方法，并提供对系统中所有软件状态信息的查询，在 Linux 系统中，RPM 和 DPKG 为最常见的两类软件包管理器，分别应用于基于 RPM 软件包的 Linux 发行版本和 DEB 软件包的 Linux 发行版本。

一个 DEB 软件包包含了已压缩的软件文件集和该软件在头文件中保存的内容信息，通常表现为以 .deb 扩展名结尾的文件，如：package.deb。对其需要使用 dpkg 命令进行操作。下面介绍 dpkg 工具的参数和使用方法。

1. dpkg 命令常用参数

dpkg 命令格式如下：

```
dpkg [ 选项 ] 软件包名
```

主要选项参数如表 7-1 所示。

表 7-1　dpkg 命令主要选项参数

参数	说明
-l	在系统中查询软件内容信息
--info	在系统中查询软件或查询指定 rpm 包的内容信息
-i	在系统中安装 / 升级软件
-r	在系统中卸载软件，不删除配置文件
-P	在系统中卸载软件以及其配置文件

【例 13】　查询系统中已安装的软件：

```
# dpkg -l package
```

【例 14】　安装 DEB 包 package.deb：

```
# sudo dpkg -i package.deb
```

【例 15】　卸载 DEB 包 package.deb：

```
# sudo dpkg -r package.deb                    # 卸载软件，不删除配置文件
# sudo dpkg -P package.deb                    # 卸载软件以及其配置文件
```

2. apt 包管理软件

APT，AdvancedPackaging Tools。通过 apt-rpm 也支持 rpm 管理，APT 的主要包管理工具为 APT-GET，通过此工具可满足和上述 YUM 相似的功能要求。

【例 16】　更新源索引：

```
# sudo apt-get update
```

【例 17】　安装：

```
# sudo apt-get install package-name
```

【例 18】　下载指定源文件：

```
# sudo apt-get source package-name
```

【例 19】　升级所有软件：

```
# sudo apt-get upgrade
```

【例 20】　卸载：

```
# sudo apt-get remove package-name            # 卸载软件，不删除配置文件
# sudo apt-get remove -purge package-name     # 卸载软件以及其配置文件
```

7.3　安装 gcc 并编写 C 语言文件

Linux 系统下使用最广泛的 C/C++ 编译器是 GCC，大多数的 Linux 发行版本都默认安装，GCC 仅仅是一个编译器，没有界面，必须在命令行模式下使用，通过 gcc 命令就可以将源文件编译成可执行文件。

1. 生成可执行程序

生成可执行文件的命令如下：

```
# cd demo          # 进入源文件所在的目录
# gcc main.c       # 在 gcc 命令后面紧跟源文件名
```

打开 demo 目录，会看到多了一个名为“a.out”的文件，Linux 系统下的可执行文件后缀理论上可以是任意的，这里的“.out”只是用来表明它是 GCC 的输出文件。不管源文件的名字是什么，GCC 生成的可执行文件的默认名字始终是“a.out”，这就是最终生成的可执行文件，如图 7-1 所示。

图 7-1　a.out 文件

若不想使用默认的文件名，那么可以通过“-o”选项来自定义文件名，如：

```
# gcc main.c -o main.out
```

这样生成的可执行程序的名字就是“main.out”。

Linux 可执行文件的后缀仅仅只是一种形式上的，故可执行文件也可以不带后缀，如：

```
# gcc main.c -o main
```

这样生成的可执行程序的名字就是“main”。

通过“-o”选项也可以将可执行文件输出到其他目录，并非一定要在当前目录下，如：

```
# gcc main.c -o ./out/main.out
```

或

```
# gcc main.c -o out/main.out
```

表示将可执行文件输出到当前目录下的 out 目录，并命名为“main.out”。“./”表示当前目录，如果不写，也默认是当前目录。

注：out 目录必须存在，如果不存在，gcc 命令不会自动创建，而是报错。

2. 运行可执行程序

在控制台中输入程序的名字就可以运行可执行程序，命令如下所示：

```
# ./a.out
```

“./”表示当前目录，该命令的意思是运行当前目录下的“a.out”程序。如果不写 ./，系统会到系统路径下查找“a.out”，而系统路径下不存在这个程序，所以会运行失败。

所谓系统路径，就是环境变量指定的路径，我们可以通过修改环境变量添加或删除某个路径，一条命令对应一个可执行程序，如果执行命令时没有指明路径，那么就会到系统路径下查找对应的程序。

如果程序在其他目录下，运行程序时还要带上目录的名字，如：

```
# ./out/main.out
```

或

```
# out/main.out
```

这个时候加不加“./”都一样，Linux 能够识别出 out 是一个目录，而不是一个命令，它默认会在当前路径下查找该目录，而不是去系统路径下查找，所以不加“./”也不会报错。

从编辑源文件到运行可执行程序的全过程：

```
# cd demo          # 进入源文件所在目录
# touch main.c     # 新建空白的源文件
# gedit main.c     # 编辑源文件
# gcc main.c       # 生成可执行程序
# ./a.out          # 运行可执行程序
```


7.4 yum 的使用

yum，全称“Yellow dog Updater, Modified”，提供了查找、安装、删除某一个、一组甚至全部软件包的命令，可以使系统管理人员交互和自动化地更细与管理 RPM 软件包，能够从指定的服务器自动下载 RPM 包并且安装，可以自动处理依赖性关系，并且一次安装所有依赖的软件包，无须繁琐地一次次下载、安装。

yum 软件可以用 rpm 命令安装，安装之前可以通过如下命令查看 yum 是否已安装：

```
# rpm -qa | grep yum
```

使用 yum 安装软件包之前，需指定好 yum 下载 RPM 包的位置，即软件安装包的来源。使用 yum 安装软件时至少需要一个 yum 源。

1. yum 配置文件

```
# cat /etc/yum.conf
[main]
cachedir=/var/cache/yum/basearch/ basearch/basearch/releasever   # rpm 包存放路径
```

```
keepcache=0   # 是否保存下载的 rpm 包。0 表示不保存，1 表示保存
logfile=/var/log/yum.log      #yum 日志文件
obsoletes=1              # 更新软件包
gpgcheck=1               # 是否进行校验
```

2. yum 命令详解

y：直接安装，不进行提示

install：安装软件包

update：更新软件包

remove：卸载软件包

search：查询软件包

provides：查询软件包的依赖关系，后接文件名。例：yum provides /usr/bin/ls

history：查询使用的 yum 历史命令，可以根据序号进行调用

grouplist：组列表，可以安装和卸载组软件包

info：查询软件包的相关信息

【例 21】　使用 yum 命令安装 zsh 软件包：

```
# yum -y install zsh
Loaded plugins: fastestmirror, langpacks
Loading mirror speeds from cached hostfile
*base: mirrors.aliyun.com
*epel: mirrors.tuna.tsinghua.edu.cn
*extras: mirrors.aliyun.com
*updates: mirrors.aliyun.com
Resolving Dependencies
–> Running transaction check
—> Package zsh.x86_64 0:5.0.2-31.el7 will be installed
–> Finished Dependency Resolution
Dependencies Resolved
ackage Arch Version Repository Size
Installing:
zsh x86_64 5.0.2-31.el7 base 2.4 M
……                    # 省略号代替中间生成的文件
Installed:
zsh.x86_64 0:5.0.2-31.el7
Complete!
```

【例 22】　使用 yum 命令卸载 zsh 软件包：

```
# yum -y remove zsh
```

```
Loaded plugins: fastestmirror, langpacks
Resolving Dependencies
–> Running transaction check
—> Package zsh.x86_64 0:5.0.2-28.el7 will be erased
–> Finished Dependency Resolution
Installed size: 5.6 M
Downloading packages:
Running transaction check
Running transaction test
Transaction test succeeded
… # 省略中间的命令
Running transaction
Erasing : zsh-5.0.2-28.el7.x86_64 1/1
Verifying : zsh-5.0.2-28.el7.x86_64 1/1
Removed:
zsh.x86_64 0:5.0.2-28.el7
Complete!
```

【例 23】 使用 yum 命令更新 zsh 软件包：

```
# yum -y update zsh                    # 需要提前准备好升级包
# yum -y update                        # 后面不加包名，是升级所有包
```

【例 24】 使用 yum 命令查询 zsh 软件包：

```
# yum search zsh
Loaded plugins: fastestmirror, langpacks
Loading mirror speeds from cached hostfile
========== N/S matched: zsh ===========
zsh.x86_64 : Powerful interactive shell
zsh-lovers.noarch : A collection of tips, tricks and examples for the Z shell
Name and summary matches only, use “search all” for everything.
```

【例 25】 使用 yum 命令查询 zsh 软件包的相关信息：

```
# yum info zsh
Loaded plugins: fastestmirror, langpacks
Loading mirror speeds from cached hostfile
Installed Packages
Name : zsh
Arch : x86_64
```

```
Version : 5.0.2
Release : 28.el7
Size : 5.6 M
Repo : installed
From repo : CentOS 7
Summary : Powerful interactive shell
URL : http://zsh.sourceforge.net/
```

【例 26】 使用 yum 命令查询“ls”命令是哪个软件包安装的：

```
# yum provides /usr/bin/ls
Loaded plugins: fastestmirror, langpacks
Loading mirror speeds from cached hostfile
coreutils-8.22-21.el7.x86_64 : A set of basic GNU tools commonly used in shell scripts
Repo : CentOS 7
Matched from:
Filename : /usr/bin/ls
coreutils-8.22-21.el7.x86_64 : A set of basic GNU tools commonly used in shell scripts
……
```


本章小结

本章介绍了使用 rpm 命令管理 RPM 包，包括：查询、安装、卸载、升级和验证，细分了不同情况下使用的项目不同，也介绍了如何使用 RPM 软件包管理器，通过 gcc 命令将源文件编译成可执行文件，了解了 GCC 生成的可执行文件的默认名字始终是“a.out”，最后介绍了如何利用 yum 命令来查找、安装、删除某一个、一组甚至全部软件包的命令。

习题

一、填写题

1. 使用 rpm 命令安装、卸载、更新 ntp–0.7.12x86_64.rpm 软件包？

安装：rpm _____ ntp–0.7.12x86_64.rpm

卸载：rpm _____ ntp–0.7.12x86_64.rpm

更新：rpm _____ ntp–0.7.12x86_64.rpm

2. 如何查询 Linux 系统上是否安装有某 rpm 包，如何强制删除 rpm 包？

查询：rpm ___“软件包名”

删除：rpm ___“软件包名”

3. 哪个命令可查看安装 openssl.x86.rpm 包的依赖关系，查询会安装哪几个文件，分别到哪个目录，

而不实际安装？

rpm _____

4. 如何查询 openssl 安装的时间？

rpm _____

5. 如何查询 /usr/lib/libssl.so.6 属于哪个包安装的？

rpm _____

二、 程序编写题

1. 光盘里有一个 httpd–2.3.15.xxxx.rpm 软件包，如何挂载并安装？

2. 使用 yum 搜索包含关键词“stu”的 rpm 包并安装，然后再使用 yum 将其卸载。

3. 如果在 ./configure 这一步出现这样的错误“configure: error: no acceptable C compiler found in $PATH”，该如何处理？

✓ 综合实训

一、实训题目

在 CentOS 7 下安装 MySQL。

二、实训目的

熟悉使用 rpm、yum 命令管理软件包。

三、实训内容

1. 查看 Linux 操作系统版本和系统内核版本。
2. 下载对应版本的 MySQL 安装文件。
3. 使用 rpm 命令安装 MySQL 组件。

第八章

磁盘管理

内容简介

- 作为Linux系统网络管理员，学习Linux文件系统和磁盘管理是对系统的日常管理起到非常重要的作用。我们在前面已经学习了 Linux 文件系统管理和相关操作，本章主要讲解磁盘管理的相关知识，请读者务必掌握。
- 如果你的 Linux 服务器有多个用户经常存取数据时，为了维护所有用户对磁盘容量的公平使用，磁盘配额（Quota）就是一项非常有用的工具。另外，硬盘阵列（RAID）及逻辑卷这些工具都可以帮助你管理与维护用户可用的硬盘容量。

学习要求

- 掌握 Linux 下的磁盘管理工具。
- 掌握 Linux 硬盘配额管理。
- 掌握 Linux 逻辑卷管理。

8.1 磁盘管理命令

在 Linux 系统安装时，其中有一个步骤是进行硬盘分区。在分区时可以采用 Disk Duid、RAID 和 LVM 等方式进行分区。除此之外，在 Linux 系统中还有 fdisk、cfdisk、parted 等分区工具。

注意：下面所有磁盘操作，都是以新增一块 SISC 硬盘为前提，新增的硬盘为 /dev/sdb。请在开始本章内容学习前在虚拟机中增加一块硬盘。

8.1.1 磁盘分区和挂载

Linux 系统在磁盘、U 盘以及光盘等设备分区和挂载操作才能使用。

1. 磁盘分区原理与规则

分区规则：

（1）主分区 + 扩展分区的数量不能超过 4 个，且扩展分区只能有 1 个。逻辑分区要在扩展分区之上进行划分，逻辑分区没有数量限制，可以任意多个。

（2）扩展分区是不能直接用的，他是以逻辑分区的方式来使用的，所以说扩展分区可分成若干逻辑分区。他们的关系是包含的关系，所有的逻辑分区都是扩展分区的一部分。

（3）硬盘的容量 = 主分区的容量 + 扩展分区的容量；扩展分区的容量 = 各个逻辑分区的容量之和。

备注：主分区就是普通磁盘分盘，但是由于磁盘设备由大量的扇区组成，一个扇区的容量为 512 字节。磁盘的第一个扇区最为重要，记录了主引导记录与分区表信息。就第一个扇区而言，主引导信息记录需要占用 466 个字节，分区表 64 个字节，结束符占用 2 个字节；其中分区表中每记录一个分区信息就需要 16 个字节，所以最多只有 4 个分区信息可以记录在第一个扇区中，所以主分区 + 扩展分区的数量不能超过 4 个。但是为了创建更多的分区，就使用扩展分区做份下若干个分区的指针，划分若干个逻辑分区，来满足分区数大于 4 个的需求。扩展分区不需要挂载，但是可以格式化。

2. CentOS 7 磁盘分区和挂载

硬盘说明：

• Linux 硬盘分 IDE 硬盘和 SCSI 硬盘，目前基本上是 SCSI 硬盘

• 对于 IDE 硬盘，驱动器标志符为“hdx~”，其中“hd”表明分区所在设备的类型，这里是指 IDE 硬盘了。“x”为盘号（a 为基本盘，b 为基本从属盘，c 为辅助主盘，d 为辅助从属盘），“~”代表分区，前四个分区用数字 1 到 4 表示，它们是主分区或扩展分区，从 5 开始就是逻辑分区。例，hda3 表示为第一个 IDE 硬盘上的第三个主分区或扩展分区，hdb2 表示第二个 IDE 硬盘上的第二个主分区或扩展分区。

• 对于 SCSI 硬盘则标记为“sdx~”，SCSI 硬盘是用“sd”来表示分区所在设备的类型的，其余则和 IDE 硬盘的表示方法一样。

（1）使用 lsblk 或 lsblk –f 查看所有设备挂载情况，如图 8-1 所示。

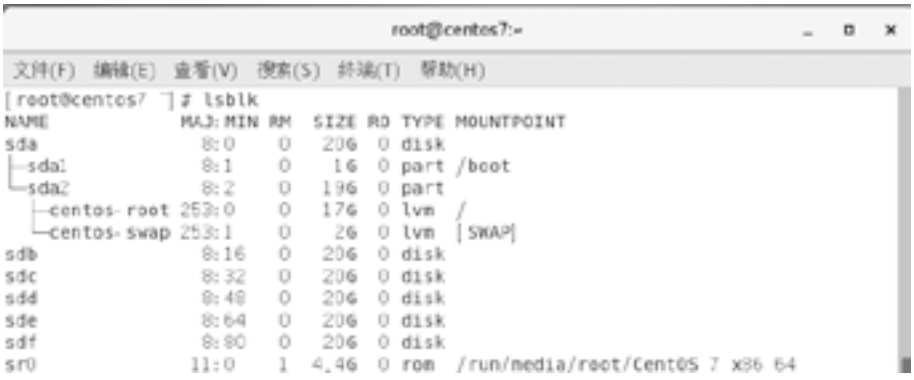

图 8-1 查看所有设备挂载

7 个栏目名称如下：

NAME：这是块设备名。

MAJ：MIN：本栏显示主要和次要设备号。

RM：本栏显示设备是否可移动设备。注意，在本例中设备 sr0（光驱）的 RM 值等于 1，这说明它们是可移动设备。

SIZE：本栏列出设备的容量大小信息。例如 20G 表明该设备大小为 20GB。

RO：该项表明设备是否为只读。在本案例中，所有设备的 RO 值为 0，表明他们不是只读的。

TYPE：本栏显示块设备是否是磁盘或磁盘上的一个分区。在本例中，sda-f 是磁盘，而 sr0 是只读存储（rom）。

MOUNTPOINT：本栏指出设备挂载的挂载点。

lsblk 命令主要用于列出所有可用块设备的信息，而且还能显示他们之间的依赖关系，但是它不会列出 RAM 盘的信息，lsblk 其他选项如表 8-1 所示。

表 8-1　lsblk 命令选项

选项	说明
-a, --all	显示所有设备
-b, --bytes	以 bytes 方式显示设备大小
-d, --nodeps	不显示 slaves 或 holders
-D, --discard	print discard capabilities
-e, --exclude <list>	排除设备 (default: RAM disks)
-f, --fs	显示文件系统信息
-h, --help	显示帮助信息
-i, --ascii	use ascii characters only
-m, --perms	显示权限信息
-l, --list	使用列表格式显示
-n, --noheadings	不显示标题
-o, --output <list>	输出列
-P, --pairs	使用 key=”value”格式显示
-r, --raw	使用原始格式显示
-t, --topology	显示拓扑结构信息

（2）fdisk 命令的使用。

使用 fdisk 命令进行分区前，先在虚拟机新增一块硬盘，本例使用的硬盘是：/dev/sdb。

fdisk 是 Linux 的磁盘分区表操作工具。利用 fdisk 对硬盘进行分区，可以在 fdisk 命令后加上要分区的硬盘作为参数，该命令格式为：

```
fdisk [ 选项 ] 设备名称
```

常用选项如表 8-2 所示。

表 8-2　fdisk 常用选项

选项	说明
a	调整硬盘启动分区
d	删除已有分区
t	修改分区类型
l	查看所有已有 ID
w	保存并退出
q	不保存并退出
m	查看帮助信息
n	创建新分区
p	显示现有分区信息
u	切换所显示的分区大小单位
x	列出高级选项

下面以在 /dev/sdb 硬盘上创建大小为 500MB，文件系统类型为 ext3 的 /dev/sdb1 主分区为例，讲解 fdisk 命令的使用。

【例 1】　显示 /dev/sdb 硬盘的分区信息。

```
[root@CentOS 7 ~]# fdisk -l /dev/sdb
磁盘 /dev/sdb：21.5 GB, 21474836480 字节，41943040 个扇区
Units = 扇区 of 1 * 512 = 512 bytes
扇区大小 ( 逻辑 / 物理 )：512 字节 / 512 字节
I/O 大小 ( 最小 / 最佳 )：512 字节 / 512 字节
```

【例 2】　对 /dev/sdb 硬盘进行分区

```
[root@CentOS 7 ~]# fdisk /dev/sdb
欢迎使用 fdisk (util-Linux 2.23.2)。
// 分区前提示
更改将停留在内存中，直到你决定将更改写入磁盘。
使用写入命令前请三思。
Device does not contain a recognized partition table
使用磁盘标志符 0x67f6f599 创建新的 DOS 磁盘标签。
命令 ( 输入 m 获取帮助 )：m    # 获取帮助
命令操作
a   toggle a bootable flag
b   edit bsd disklabel
c   toggle the dos compatibility flag
d   delete a partition
g   create a new empty GPT partition table
G   create an IRIX (SGI) partition table
```

```
l   list known partition types
m   print this menu
n   add a new partition
o   create a new empty DOS partition table
p   print the partition table
q   quit without saving changes
s   create a new empty Sun disklabel
t   change a partition's system id
u   change display/entry units
v   verify the partition table
w   write table to disk and exit
x   extra functionality (experts only)
命令 ( 输入 m 获取帮助 )：p    # 打印当前分区表
磁盘 /dev/sdb：21.5 GB, 21474836480 字节，41943040 个扇区
Units = 扇区 of 1 * 512 = 512 bytes
扇区大小 ( 逻辑 / 物理 )：512 字节 / 512 字节
I/O 大小 ( 最小 / 最佳 )：512 字节 / 512 字节
磁盘标签类型：dos
磁盘标志符：0x67f6f599
设备 Boot      Start         End      Blocks   Id  System
命令 ( 输入 m 获取帮助 )：n      # 创建第一个分区
Partition type:
p   primary (0 primary, 0 extended, 4 free)
e   extended
Select (default p): p     # 创建第一个主分区，大小为 500M
分区号 (1-4，默认 1)：
起始扇区 (2048-41943039，默认为 2048)：
将使用默认值 2048
Last 扇区 , + 扇区 or +size{K,M,G} (2048-41943039，默认为 41943039)：+500M
分区 1 已设置为 Linux 类型，大小设为 500 MiB
命令 ( 输入 m 获取帮助 )：n   # 创建第二个主分区，大小为 500M
Partition type:
p   primary (1 primary, 0 extended, 3 free)
e   extended
Select (default p): p
分区号 (2-4，默认 2)：
起始扇区 (1026048-41943039，默认为 1026048)：
将使用默认值 1026048
```

```
Last 扇区，+ 扇区 or +size{K,M,G} (1026048-41943039，默认为 41943039)：+500M
分区 2 已设置为 Linux 类型，大小设为 500 MiB
命令 ( 输入 m 获取帮助 )：n   # 创建第三个主分区，大小为 10G
Partition type:
p   primary (2 primary, 0 extended, 2 free)
e   extended
Select (default p): p
分区号 (3,4，默认 3)：
起始扇区 (2050048-41943039，默认为 2050048)：
将使用默认值 2050048
Last 扇区，+ 扇区 or +size{K,M,G} (2050048-41943039，默认为 41943039)：+10G
分区 3 已设置为 Linux 类型，大小设为 10 GiB
命令 ( 输入 m 获取帮助 )：p   # 打印当前分区信息
磁盘 /dev/sdb：21.5 GB, 21474836480 字节，41943040 个扇区
Units = 扇区 of 1 * 512 = 512 bytes
扇区大小 ( 逻辑 / 物理 )：512 字节 / 512 字节
I/O 大小 ( 最小 / 最佳 )：512 字节 / 512 字节
磁盘标签类型：dos
磁盘标志符：0x67f6f599
设备 Boot     Start       End     Blocks   Id  System
/dev/sdb1         2048    1026047    512000   83  Linux
/dev/sdb2      1026048    2050047    512000   83  Linux
/dev/sdb3      2050048   23021567   10485760   83  Linux
命令 ( 输入 m 获取帮助 )：w   # 保存分区表
The partition table has been altered!
Calling ioctl() to re-read partition table.
正在同步磁盘。
```

【例 3】 格式化新建的分区，在各分区上建立 ext4 类型的文件系统，建立时要求检查磁盘坏块并显示详细信息。

```
[root@CentOS 7 ~]# mkfs -t ext4 -V -c /dev/sdb1
[root@CentOS 7 ~]# mkfs -t ext4 -V -c /dev/sdb2
[root@CentOS 7 ~]# mkfs -t ext4 -V -c /dev/sdb3
```

【例 4】 将 /dev/sdb1 挂载到 /user1 目录，/dev/sdb2 挂载到 /user2，/dev/sdb3 挂载到 /user3。

```
[root@CentOS 7 ~]# mkdir /user1
[root@CentOS 7 ~]# mkdir /user2
```

```
[root@CentOS 7 ~]# mkdir /user3
[root@CentOS 7 ~]# mount /dev/sdb1 /user1
[root@CentOS 7 ~]# mount /dev/sdb2 /user2
[root@CentOS 7 ~]# mount /dev/sdb3 /user3
```

通过以上四个例子，对一块磁盘进行分区、挂载后，可以进去相对应的分区进行文件的创建等操作。

（3）设置自动挂载。如果要实现每次开机自动挂载文件系统，可以通过编辑 /etc/fstab 文件来实现。在 /etc/fstab 中列出了引导系统时需要挂载的文件系统以及文件系统的类型和挂载参数，系统在引导过程中会读取 /etc/fstab 文件，并根据该文件的配置参数挂载相应的文件系统。以下是一个 fstab 文件内容：

```
[root@CentOS 7 ~]# cat /etc/fstab
#
# /etc/fstab
# Created by anaconda on Tue Jan 28 00:52:47 2020
#
# Accessible filesystems, by reference, are maintained under '/dev/disk'
# See man pages fstab(5), findfs(8), mount(8) and/or blkid(8) for more info
#
/dev/mapper/centos-root     /           xfs    defaults       0 0
UUID=41cae9f5-8a97-4558-8e0b-509203cec067 /boot xfs    defaults       0 0
/dev/mapper/centos-swap    swap          swap    defaults       0 0
```

fstab 文件的每一行代表一个文件系统，每一行又包含 6 列，这 6 列内容如下所示：

fs_spec　fs_file　fs_vfstype　fs_mntops　fs_freq fs_passno

具体含义说明如下。

fs_spec：要挂载的设备或伪文件系统：设备名称，LABEL，UUID，伪文件系统名称。

fs_file：挂载点（扩展为 swap）。

fs_vfstype：文件系统类型（auto 寻找默认的）：

普通设备挂载：ext#/xfs 等文件系统类型。

交换分区 swap：swap。

网络挂载地址 Linux 之间：nfs，windos 挂载：cifs。

光盘 iso9660，文件 cifs。

fs_mntops：挂载选项：defaults 有需要的功能可以添加，不能为空。

fs_freq：转储频率：0：不做备份 1：每天转储 2：每隔一天转储。

fs_passno：开机时自检（非 0），自检次序：（如果开机系统自检不过，就无法正常开机）

0：不自检。

1：最先自检，一般为 / 2...：数越小优先级最大。

如果设备不小心被破坏，开机自检不过，无法正常启用，该怎么办？原理很简单，只需要修复一下

文件系统就好了（如果不能修复，可以去 /etc/fstab 中把自检改为 0）。

如果需要每次开机自动将文件系统类型为 ext4 的分区 /dev/sdb1 自动挂载到 /user1 目录下，需要在 /etc/fstab 文件中添加下面一行内容，重新启动系统后，/dev/sdb1 就能自动挂载了。

```
/dev/sdb1     /user1  ext4   defaults      0     0
```

8.1.2 mount 与 umount

1. mount 命令

Linux 中的根目录以外的文件要想被访问，需要将其“关联”到根目录下的某个目录来实现，这种关联操作就是“挂载”，这个目录就是“挂载点”，解除次关联关系的过程称之为“卸载”。Linux 系统中提供了 /mnt 和 /media 两个专门的挂载点。一般而言，挂载点应该是一个空目录，否则目录中原来的文件将被系统隐藏。通常将光盘和软盘挂载到 /media/cdrom 和 /media/floppy 中，其对应的设备文件名分别为 /dev/cdrom 和 /dev/fd0。因此，“挂载点”的目录需要以下几个要求：

- 目录事先存在，可以用 mkdir 命令新建目录。
- 挂载点目录不可被其他进程使用到。
- 挂载点下原有文件将被隐藏。

文件系统的挂载可以在系统引导过程中自动挂载，也可以手动挂载。手动挂载文件系统的挂载命令是 mount。该命令的语法格式如下：

mount [选项] 设备挂载点

（1）挂载设备常用有以下几种：

- 设备文件：例如 /dev/sdb2
- 卷标：-L ‘LABEL’，例如 -L ‘MYDATA’
- UUID，-U ‘UUID’：例如 -U ‘0c50523c-43f1-45e7-85c0-a126711d406e’
- 伪文件系统名称：proc，sysfs，devtmpfs，configfs

（2）常用命令选项有如下几种：

- -t（vsftype）：指定要挂载的设备上的文件系统类型。
- -r（readonly）：只读挂载。
- -w（read and write）：读写挂载。
- -n：不更新 /etc/mtab。
- -a：自动挂载所有支持自动挂载的设备（定义在 /etc/fstab 文件中，且挂载选项中有“自动挂载”功能）。
- -L（LABEL）：以卷标指定挂载设备。
- -U（UUID）：以 UUID 指定要挂载的设备。
- -B（bind）：绑定目录到另一个目录。

注意：查看内核追踪到的已挂载的所有设备：cat /proc/mounts

【例 5】　把文件系统类型为 ext4 的硬盘分区 /dev/sdb1 挂载到 /media/sdb1 目录下。

```
[root@CentOS 7 ~]# mount -t ext4 /dev/sdb1 /media/sdb1
```

2. umount 命令

文件系统可以被挂载也可以被卸载。卸载文件系统使用的命令是 umount。Umount 命令的格式为：

```
umount 设备挂载点
```

【例 6】　卸载 /dev/sdb1

```
[root@CentOS 7 ~]# umount /dev/sdb1 /media/sdb1
```

注意：光盘在没有卸载之前，无法从驱动器中弹出；正在使用的文件系统不能卸载。

8.1.3　常用的磁盘管理命令

1. df

Linux 中 df 命令的功能是用来检查 Linux 服务器的文件系统的磁盘空间占用情况。可以利用该命令来获取硬盘被占用了多少空间，目前还剩下多少空间等信息。

命令格式：

```
df [ 选项 ] [ 文件 ]
```

命令功能：

显示指定磁盘文件的可用空间。如果没有文件名被指定，则所有当前被挂载的文件系统的可用空间将被显示。默认情况下，磁盘空间将以 1KB 为单位进行显示，除非环境变量 POSIXLY_CORRECT 被指定，那样将以 512 字节为单位进行显示。命令参数如表 8-3 所示。

表 8-3　命令参数说明

参数	说明
-a	全部文件系统列表
-h	方便阅读方式显示
-H	等于“-h”，但是计算式，1K=1000，而不是 1K=1024
-i	显示 inode 信息
-k	区块为 1024 字节
-l	只显示本地文件系统
-m	区块为 1048576 字节
- no-sync	忽略 sync 命令
-P	输出格式为 POSIX
- sync	在取得磁盘信息前，先执行 sync 命令
-T	文件系统类型
- block-size=< 区块大小 >	指定区块大小

续表

参数	说明
-t< 文件系统类型 >	只显示选定文件系统的磁盘信息
-x< 文件系统类型 >	不显示选定文件系统的磁盘信息
- help	显示帮助信息
- version	显示版本信息

【例 7】 列出文件系统空间占用信息

```
[root@CentOS 7 ~]#df-h
文件系统                容量    已用    可用    已用 %  挂载点
devtmpfs                470M    0       470M    0%      /dev
tmpfs                   487M    0       487M    0%      /dev/shm
tmpfs                   487M    8.6M    478M    2%      /run
tmpfs                   487M    0       487M    0%      /sys/fs/cgroup
/dev/mapper/centos-root 17G     4.3G    13G     26%     /
/dev/sda1               1014M   171M    844M    17%     /boot
tmpfs                   98M     4.0K    98M     1%      /run/user/42
tmpfs                   98M     32K     98M     1%      /run/user/0
/dev/sr0                4.4G    4.4G    0       100%    /run/media/root/CentOS 7 x86_64
/dev/sdb1               477M    2.3M    449M    1%      /user1
```

说明：

Linux 中 df 命令的输出清单的第 1 列是代表文件系统对应的设备文件的路径名（一般是硬盘上的分区）；第 2 列给出分区包含的数据块（1024 字节）的数目；第 3、4 列分别表示已用的和可用的数据块数目。用户也许会感到奇怪的是，第 3、4 列块数之和不等于第 2 列中的块数，这是因为缺省的每个分区都留了少量空间供系统管理员使用。即使遇到普通用户空间已满的情况，管理员仍能登录和留有解决问题所需的工作空间。清单中“已用 %”列表示普通用户空间使用的百分比，即使这一数字达到 100%，分区仍然留有系统管理员使用的空间。最后，“挂载点”列表示文件系统的挂载点。

【例 8】 列出系统内所有特殊文件格式及名称

```
[root@CentOS 7 ~]#df-aT
文件系统        类型        1K- 块    已用    可用        已用 %  挂载点
sysfs           sysfs       0         0       0           -       /sys
proc            proc        0         0       0           -       /proc
devtmpfs        devtmpfs    480840    0       480840      0%      /dev
securityfs      securityfs  0         0       0           -       /sys/kernel/security
tmpfs           tmpfs       497872    0       497872      0%      /dev/shm
devpts          devpts      0         0       0           -       /dev/pts
tmpfs           tmpfs       497872    8800    489072      2%      /run
```

```
tmpfs                tmpfs        497872  0      497872  0%      /sys/fs/cgroup
（其他略）
```

2. du

du 命令用于显示指定文件（夹）在磁盘中所占的空间信息。假如指定的文件参数实际上是一个目录，就要计算该目录下的所有文件。假如没有提供文件参数，执行 du 命令，显示当前目录内的文件占用空间信息。该命令的语法格式为：

```
du [ 参数选项 ] [ 文件或目录名称 ]
```

命令参数选项如表 8-4 所示。

表 8-4　du 命令参数说明

选项	说明
-a	显示目录中个别文件的大小
-b	显示目录或文件大小时，以 byte 为单位
-c	除了显示个别目录或文件的大小外，同时也显示所有目录或文件的总和
-D	显示指定符号连接的源文件大小
-h	以 K、M、G 为单位，提高信息的可读性
-H	与 -h 参数相同，但是 K、M、G 是以 1000 为换算单位
-k	以 1024 bytes 为单位
-l	重复计算硬链接文件
-L< 符号连接 >	显示选项中所指定符号链接（软链接）的源文件大小
-m	以 1MB 为单位
-s	显示总计大小
-S	显示个别目录的大小时，并不含其子目录的大小
-x	以一开始处理时的文件系统为准，若遇上其他不同的文件系统目录则略过
- exclude=< 目录或文件 >	略过指定的目录或文件
- max-depth=< 目录层数 >	超过指定层数的目录后，予以忽略

【例 9】　查看计算目录大小的命令。

```
[root@CentOS 7 ~]# du centos
4       centos
```

3. mkfs

mkfs 命令是“make filesystem”的缩写，是用来在指定设备创建一个 Linux 文件系统，通常是在硬盘上。文件系统既可以是设备名（如：/dev/hda1，/dev/sdb2），也可以是文件系统的挂载点（如：/，/usr，/home）。块是指该文件系统用的块数。如果 mkfs 成功执行时返回值为 0，反之是 1。事实上，mkfs 是在 Linux 下各文件系统专用程序（mkfs.fstype）的前端程序。各文件系统专用程序可以在 /sbin，/sbin/fs，/sbin/fs.d，/etc/fs，/etc 等目录中找到（精确定义一般都在编译内核时定义，但通常包含有 /sbin 和 /sbin/fs），并最终在环境变量 PATH 列出的目录中。

该命令用来在特定的分区创建 Linux 文件系统，常见的文件系统有 ext2、ext3、ext4、vfat 等，执

行 mkfs 命令其实是在调用：mkfs.ext3、 mkfs.reiserfs、mkfs.ext2、 mkdosfs 、mkfs.msdos、mkfs.vfat。

```
[root@CentOS 7 ~]# ls -l /sbin/mkfs.*
-rwxr-xr-x. 1 root root 375240 8 月   7 2017 /sbin/mkfs.btrfs
-rwxr-xr-x. 1 root root  37024 8 月   9 11:10 /sbin/mkfs.cramfs
-rwxr-xr-x. 4 root root  96384 8 月   9 07:15 /sbin/mkfs.ext2
-rwxr-xr-x. 4 root root  96384 8 月   9 07:15 /sbin/mkfs.ext3
-rwxr-xr-x. 4 root root  96384 8 月   9 07:15 /sbin/mkfs.ext4
-rwxr-xr-x. 1 root root  28720 10 月 31 2018 /sbin/mkfs.fat
-rwxr-xr-x. 1 root root  37136 8 月   9 11:10 /sbin/mkfs.minix
lrwxrwxrwx. 1 root root      8 1 月  28 00:56 /sbin/mkfs.msdos -> mkfs.fat
lrwxrwxrwx. 1 root root      8 1 月  28 00:56 /sbin/mkfs.vfat -> mkfs.fat
-rwxr-xr-x. 1 root root 368424 8 月   9 11:23 /sbin/mkfs.xfs
```

比如：

```
mkfs.ext3 /dev/sda6      # 把该设备格式化成 ext3 文件系统
mkfs.vfat /dev/sda6      # 格式化成 fat32 文件系统
mkfs.msdos /dev/sda6     # 格式化成 fat16 文件系统 ,msdos 就是 fat16
mkfs 命令格式：
mkfs [-V] [-t fstype] [fs-options] filesys [blocks]
mkfs 命令常用参数选项如下：
device：预备检查的硬盘分区，例如：/dev/sda1。
-V：详细显示模式。
-t：给定档案系统的型式，Linux 的预设值为 ext2。
-c：在制作档案系统前，检查该 partition 是否有坏轨。
-l bad_blocks_file：将有坏轨的 block 资料加到 bad_blocks_file 里面。
block：给定 block 的大小。
```

【例 10】 在 /dev/sdb3 上建一个 ext3 文件系统，同时检查是否有坏轨存在，并且将过程详细列出来。

```
[root@CentOS 7 ~]# mkfs -V -t ext3 -c /dev/sdb3
```

4. fsck

fsck（file system check）用来检查和维护不一致的文件系统。若系统掉电或磁盘发生问题，可利用 fsck 命令对文件系统进行检查。fsck 命令格式如下：

```
fsck [ 参数选项 ] 文件系统
```

注意：千万不能在运行的系统上面直接执行 fsck，特别是 RHEL6.0 以下 ext3 的文件系统，否则很可能会损坏根文件系统。fsck 命令常用参数选项如表 8-5 所示。

表 8-5　fsck 命令常用参数说明

参数	说明
-a	自动修复文件系统，不询问任何问题
-A	依照 /etc/fstab 配置文件的内容，检查文件内的全部文件系统
-N	不执行指令，仅列出实际执行会进行的动作
-P	当搭配”-A”参数使用时，则会同时检查所有的文件系统
-r	采用互动模式，在执行修复时询问问题，让用户得以确认
-R	当搭配”-A”参数使用时，则会略过 / 目录的文件系统不予检查
-s	依序执行检查作业，而非同时执行
-t< 文件系统类型 >	指定要检查的文件系统类型
-T	执行 fsck 指令时，不显示标题信息
-V	显示指令执行过程

【例 11】　检查分区 /dev/sdb1 是否有错误，如果有错误自动修复

```
[root@CentOS 7 ~]# fsck -a /dev/sdb1
fsck，来自 util-Linux 2.23.2
/dev/sdb1: 正在修复日志
/dev/sdb1: clean, 12/128016 files, 26701/512000 blocks
```


8.2　磁盘配额

当 Linux 根分区的磁盘空间耗尽时，Linux 系统将无法再建立新的文件（包括程序运行的临时文件），从而出现服务程序崩溃、系统无法启动等故障现象。为了避免在服务器中出现类似的磁盘空间不足的问题，可以设置启用磁盘配额功能，对用户在指定文件系统（分区）中使用的磁盘空间、文件数量进行设置，以防止个别用户恶意或无意间占用大量磁盘空间，从而保持系统存储空间的稳定性和持续可用性。

Linux 作为一个多用户的操作系统，实际生活中会发生多人共同使用一个磁盘的情况，为了保证一个磁盘的有效利用，必须对磁盘进行使用率的限制，因此磁盘配额（quota）会是一个非常有用的工具。在 Linux 系统中可以通过索引节点数和磁盘块数来限制用户和组群对磁盘空间的使用。

8.2.1　磁盘配额的限制对象

只在指定的文件系统（分区）内有效，用户使用其他未设置配额的文件系统时，将不会受到限制。

主要针对指定的用户账号、组账号进行限制，没有被设置限额的用户或组将不受影响。对组账号设置配额后，组内所有用户使用的磁盘容量、文件数量的总和不能超过限制。

8.2.2 磁盘配额的限制类型

磁盘容量：限制磁盘数据块（Block）大小，也就是磁盘空间大小（默认单位为 KB）。

文件数量：限制用户能够拥有的文件个数。在 Linux 系统中，每一个文件都有一个对应的数字标记，称为 i 节点（inode）编号，这个编号在同一个文件系统内是唯一的，一次 quota 通过限制 i 节点的数量来实现对文件数量的限制。

8.2.3 磁盘配额的限制方法

软限制：指定一个软性的配额数值，在固定的宽限期（默认为 7 天）内允许暂时超过这个限制，但系统会给出警告信息（通俗地说就是有的商量，不会一下子就不能用了）。

硬限制：指定一个硬性的配额数值，是绝对禁止用户超过的限制值，当达到硬限制值时，系统也会给出警告并禁止继续写入数据。硬限制的配额值大于相应的软限制值，否则软限制将失效。

对 EXT 系列文件系统，quota 仅能针对整个文件系统进行设计，无法对单一的目录进行磁盘配额；而在 xfs 的文件系统中，可以使用 quota 对目录进行磁盘配额，因此在进行磁盘配额前，一定要对文件系统进行检查。

注意：在 CentOS 7 系统中，内核已经制定了支持 Linux 文件系统的磁盘配额功能，而且在系统中默认安装了 quota 软件包，用于配置和管理磁盘配额。如果没有，请使用 rpm -q quota 安装。

8.2.4 磁盘配额大致可分为 4 个步骤

- 启用磁盘配额管理（修改挂载配置文件）
- 生成配置文件
- 设置用户和组的磁盘配额
- 激活磁盘配额

下面以在 /dev/sdb2 分区上启用硬盘功能为例，讲解磁盘配额的具体配置。（注意：/dev/sdb2 挂载到 /user2 目录上，并在 /etc/fstab 配置文件上完成自动挂载功能，并重启系统。）

1. 启用磁盘配额管理（修改挂载配置文件）

修改 /etc/fstab 文件，为了启用用户的磁盘配额功能，需要在 /etc/fstab 文件中加入 usrquota，启用用户组的磁盘配额功能，需要在 /etc/fstab 文件中加入 grpquota。

```
/dev/sdb2      /user2  ext4    defaults,usrquota,grpquota      0      0
```

重启系统，或者使用 mount -a 或者 mount -o remount /user2 重新挂载增加了磁盘配额功能的文件系统。使用以下命令查看是否启用了 usrquota 和 grpquota。

```
[root@CentOS 7 ~]# mount | grep /user2
/dev/sdb2 on /user2 type ext4 (rw,relatime,seclabel,quota,usrquota,grpquota,
data=ordered)
```

2. 生成配额文件

quotacheck 命令可以对文件系统进行磁盘配额检测，发现哪些文件系统启用了磁盘配额功能，并在这些文件系统中生成配额文件 aquota.user 和 aquota.group。

```
[root@CentOS 7 ~]# quotacheck -cvug /user2
quotacheck: Your kernel probably supports journaled quota but you are not using it. Consider switching to journaled quota to avoid running quotacheck after an unclean shutdown.
quotacheck: Scanning /dev/sdb2 [/user2] done
quotacheck: Checked 3 directories and 2 files
[root@CentOS 7 ~]# ls /user2
aquota.group  aquota.user  lost+found
```

相关选项的作用：

-c：创建配额文件。

-v：显示详细信息。

-u：检查用户配额信息，创建 aquota.user 文件。

-g：检查组配额信息，创建 aquota.group 文件。

3. 设置用户和组的磁盘配额

（1）编辑用户的配额设置

使用 edquota 命令结合“-u”“-g”选项可用于编辑用户或组的配额设置。

```
[root@CentOS 7 ~]# edquota -u myquota1
```

设置用户 myquota1 的磁盘配额

```
Disk quotas for user myquota1 (uid 1001):
filesystem   blocks     soft     hard    inodes    soft    hard
/dev/sdb2      0       400000   500000    0        0       0
```

用户配额说明：

filesystem：表示本行配置对应的文件系统（分区），即配额的作用范围。

blocks：表示当前已使用的磁盘容量，默认单位为 KB。该值由 edquota 程序自动计算生成。

soft：

第 3 列中的 soft 对应为磁盘容量的软限制数值，默认单位为 KB。

第 6 列中的 soft 对应为文件数量的软限制数值位为个。

hard：

第 4 列中的 hard 对应为磁盘容量的硬限制数值，默认单位为 KB。

第 7 列中的 hard 对应为文件数量的硬限制数值，默认单位为个。

inodes：表示当前已拥有的文件数量。该值由 edquota 程序自动计算生成。

如果需要将 myquota1 用户的设置值复制给其他用户如 myquota2、myquota3 等。

```
[root@CentOS 7 ~]# edquota -p myquota1 -u myquota2
[root@CentOS 7 ~]# edquota -p myquota1 -u myquota3
```

（2）编辑用户组（myquotagrp）的配额设置

```
[root@CentOS 7 ~]# edquota -g myquotagrp
```

注意：配额设置仅对基本组生效。如用户 myquota1 所属的基本组是“myquotagrp”，所属的附加组是“technology”，那么只有针对“myquotagrp”组设置的配额才对 myquota1 有效，而针对“technology”组设置的配额则对 myquota1 没有限制。

4. 激活磁盘配额

在设置好用户和组的磁盘配额后，磁盘配额功能还不能产生作用，需要使用 quotaon 命令来激活磁盘配额功能，如果要关闭该功能则使用 quotaoff 命令。下面是启动和关闭磁盘配额命令。

```
[root@CentOS 7 ~]# quotaon -vug /user2
/dev/sdb2 [/user2]: group quotas turned on
/dev/sdb2 [/user2]: user quotas turned on
[root@CentOS 7 ~]# quotaoff -vug /user2
/dev/sdb2 [/user2]: group quotas turned off
/dev/sdb2 [/user2]: user quotas turned off
```

相关选项的作用：

-u：激活用户磁盘配额。

-g：激活组磁盘配额。

-v：显示详细信息。

5. 检查硬盘配额的使用情况

验证时，需要放入大文件或者用 dd 命令生成指定大小的测试文件进行测试。磁盘配额设置生效之后，如果要查看某个用户的硬盘配额及其使用情况可以使用 quota 命令。查看指定用户的磁盘配额可以使用“quota -u 用户名”，查看指定组的磁盘配额可以使用“quota -g 组名称”。对于普通用户而言可以直接使用“quota”命令查看自己的磁盘配额使用情况。利用“quota -a ”可以列出系统中所有用户的磁盘配额信息。另外，系统管理员可以利用 repquota 命令生成完整的硬盘空间使用报告。例如，查看 /dev/sdb2 上的硬盘使用报告。

```
[root@CentOS 7 ~]# repquota /dev/sdb2
*** Report for user quotas on device /dev/sdb2
Block grace time: 7days; inode grace time: 7days
```

```
Block limits                File limits
User            used    soft    hard  grace    used  soft  hard  grace
----------------------------------------------------------------------
root      --      13       0       0              2     0     0
myquota1  --       0  400000  500000              0     0     0
```

注意：用户名后的“--”分别用于判断该用户是否超出磁盘空间限制及索引节点数目限制。当磁盘空间及索引节点数的限制超出时，相应地“--”就会变为“+”。最后的 grace 列通常是空的。要查看所有启用了磁盘配额的文件系统的磁盘使用情况，可以使用命令“repquota -a”。

8.3　RAID 的使用

RAID 技术相信大家都有接触过，尤其是服务器运维人员，RAID 概念很多，有时候会概念混淆。下面我们来对 RAID 技术的概念特征、基本原理、关键技术、各种等级和发展现状进行全面地阐述，并为用户如何进行应用选择提供了基本原则，对于初学者应该有很大的帮助。

8.3.1　RAID 概述

1988 年美国加州大学伯克利分校的 D. A. Patterson 教授等首次在论文“A Case of Redundant Array of Inexpensive Disks”中提出了 RAID 概念，即廉价冗余磁盘阵列（Redundant Array of Inexpensive Disks）。由于当时大容量磁盘比较昂贵，RAID 的基本思想是将多个容量较小、相对廉价的磁盘进行有机组合，从而以较低的成本获得与昂贵大容量磁盘相当的容量、性能、可靠性。随着磁盘成本和价格的不断降低，RAID 可以使用大部分的磁盘，“廉价”已经毫无意义。因此，RAID 咨询委员会（RAID Advisory Board， RAB）决定用“独立”替代“廉价”，于时 RAID 变成了独立磁盘冗余阵列（Redundant Array of Independent Disks）。但这仅仅是名称的变化，实质内容没有改变。

RAID 这种设计思想很快被业界接纳，RAID 技术作为高性能、高可靠的存储技术，已经得到了非常广泛的应用。RAID 主要利用数据条带、镜像和数据校验技术来获取高性能、可靠性、容错能力和扩展性，根据运用或组合运用这三种技术的策略和架构，可以把 RAID 分为不同的等级，以满足不同数据应用的需求。D. A. Patterson 等的论文中定义了 RAID1 ~ RAID5 原始 RAID 等级，1988 年以来又扩展了 RAID0 和 RAID6。近年来，存储厂商不断推出诸如 RAID7、RAID10/01、RAID50、RAID53、RAID100 等 RAID 等级，但这些并无统一的标准。目前业界公认的标准是 RAID0~RAID5，除 RAID2 外的四个等级被定为工业标准，而在实际应用领域中使用最多的 RAID 等级是 RAID0、RAID1、RAID3、RAID5、RAID6 和 RAID10。

从实现角度看，RAID 主要分为软 RAID、硬 RAID 以及软硬混合 RAID 三种。软 RAID 所有功能均有操作系统和 CPU 来完成，没有独立的 RAID 控制 / 处理芯片和 I/O 处理芯片，效率自然最低。硬 RAID 配备了专门的 RAID 控制 / 处理芯片和 I/O 处理芯片以及阵列缓冲，不占用 CPU 资源，但成本很

高。软硬混合 RAID 具备 RAID 控制 / 处理芯片，但缺乏 I/O 处理芯片，需要 CPU 和驱动程序来完成，性能和成本在软 RAID 和硬 RAID 之间。

RAID 每一个等级代表一种实现方法和技术，等级之间并无高低之分。在实际应用中，应当根据用户的数据应用特点，综合考虑可用性、性能和成本来选择合适的 RAID 等级，以及具体的实现方式。

8.3.2 基本原理

RAID（Redundant Array of Independent Disks）即独立磁盘冗余阵列，通常简称为磁盘阵列。简单地说，RAID 是由多个独立的高性能磁盘驱动器组成的磁盘子系统，从而提供比单个磁盘更高的存储性能和数据冗余的技术。RAID 是一类多磁盘管理技术，其向主机环境提供了成本适中、数据可靠性高的高性能存储。SNIA 对 RAID 的定义是：一种磁盘阵列，部分物理存储空间用来记录保存在剩余空间上的用户数据的冗余信息。当其中某一个磁盘或访问路径发生故障时，冗余信息可用来重建用户数据。

RAID 的初衷是为大型服务器提供高端的存储功能和冗余的数据安全。在整个系统中，RAID 被看作是由两个或更多磁盘组成的存储空间，通过并发地在多个磁盘上读写数据来提高存储系统的 I/O 性能。大多数 RAID 等级具有完备的数据校验、纠正措施，从而提高系统的容错性，甚至镜像方式，大大增强系统的可靠性。

磁盘阵列可以在部分磁盘（单块或多块，根据实现而论）损坏的情况下，仍能保证系统不中断地连续运行。在重建故障磁盘数据至新磁盘的过程中，系统可以继续正常运行，但是性能方面会有一定程度上的降低。一些磁盘阵列在添加或删除磁盘时必须停机，而有些则支持热交换（Hot Swapping），允许不停机下替换磁盘驱动器。这种高端磁盘阵列主要用于要求高可能性的应用系统，系统不能停机或尽可能少的停机时间。一般来说，RAID 不可作为数据备份的替代方案，它对非磁盘故障等造成的数据丢失无能为力，比如病毒、人为破坏、意外删除等情形。此时的数据丢失是相对操作系统、文件系统、卷管理器或者应用系统来说的，对于 RAID 系统来身，数据都是完好的，没有发生丢失。所以，数据备份、灾备等数据保护措施是非常必要的，与 RAID 相辅相成，保护数据在不同层次的安全性，防止发生数据丢失。

RAID 中主要有三个关键概念和技术：镜像（Mirroring）、数据条带（Data Stripping）和数据校验（Data parity）。镜像，将数据复制到多个磁盘，一方面可以提高可靠性，另一方面可并发从两个或多个副本读取数据来提高读性能。显而易见，镜像的写性能要稍低，确保数据正确地写到多个磁盘需要更多的时间消耗。数据条带，将数据分片保存在多个不同的磁盘，多个数据分片共同组成一个完整数据副本，这与镜像的多个副本是不同的，它通常用于性能考虑。数据条带具有更高的并发粒度，当访问数据时，可以同时对位于不同磁盘上数据进行读写操作，从而获得非常可观的 I/O 性能提升。数据校验，利用冗余数据进行数据错误检测和修复，冗余数据通常采用海明码、异或操作等算法来计算获得。利用校验功能，可以很大程度上提高磁盘阵列的可靠性、鲁棒性和容错能力。不过，数据校验需要从多处读取数据并进行计算和对比，会影响系统性能。不同等级的 RAID 采用一个或多个以上的三种技术，来获得不同的数据可靠性、可用性和 I/O 性能。至于设计何种 RAID（甚至新的等级或类型）或采用何种模式的 RAID ，需要在深入理解系统需求的前提下进行合理选择，综合评估可靠性、性能和成本来进行折中地选择。

RAID 思想从提出后就广泛被业界所接纳，存储工业界投入了大量的时间和财力来研究和开发相关产

品。而且，随着处理器、内存、计算机接口等技术的不断发展，RAID 不断地发展和革新，在计算机存储领域得到了广泛的应用，从高端系统逐渐延伸到普通的中低端系统。RAID 技术如此流行，源于其具有显著的特征和优势，基本可以满足大部分的数据存储需求。

总体说来，RAID 主要优势有如下几点：

1. 大容量

这是 RAID 的一个显然优势，它扩大了磁盘的容量，由多个磁盘组成的 RAID 系统具有海量的存储空间。现在单个磁盘的容量就可以到 1TB 以上，这样 RAID 的存储容量就可以达到 PB 级，大多数的存储需求都可以满足。一般来说，RAID 可用容量要小于所有成员磁盘的总容量。不同等级的 RAID 算法需要一定的冗余开销，具体容量开销与采用算法相关。如果已知 RAID 算法和容量，可以计算出 RAID 的可用容量。通常，RAID 容量利用率在 50% ~ 90% 之间。

2. 高性能

RAID 的高性能受益于数据条带化技术。单个磁盘的 I/O 性能受到接口、带宽等计算机技术的限制，性能往往很有限，容易成为系统性能的瓶颈。通过数据条带化，RAID 将数据 I/O 分散到各个成员磁盘上，从而获得比单个磁盘成倍增长的聚合 I/O 性能。

3. 可靠性

可用性和可靠性是 RAID 的另一个重要特征。从理论上讲，由多个磁盘组成的 RAID 系统在可靠性方面应该比单个磁盘要差。这里有个隐含假定：单个磁盘故障将导致整个 RAID 不可用。RAID 采用镜像和数据校验等数据冗余技术，打破了这个假定。镜像是最为原始的冗余技术，把某组磁盘驱动器上的数据完全复制到另一组磁盘驱动器上，保证总有数据副本可用。比起镜像 50% 的冗余开销，数据校验要小很多，它利用校验冗余信息对数据进行校验和纠错。RAID 冗余技术大幅提升数据可用性和可靠性，保证了若干磁盘出错时，不会导致数据的丢失，不影响系统的连续运行。

4. 可管理性

实际上，RAID 是一种虚拟化技术，它对多个物理磁盘驱动器虚拟成一个大容量的逻辑驱动器。对于外部主机系统来说，RAID 是一个单一的、快速可靠的大容量磁盘驱动器。这样，用户就可以在这个虚拟驱动器上来组织和存储应用系统数据。从用户应用角度看，可使存储系统简单易用，管理也很便利。由于 RAID 内部完成了大量的存储管理工作，管理员只需要管理单个虚拟驱动器，可以节省大量的管理工作。RAID 可以动态增减磁盘驱动器，可自动进行数据校验和数据重建，这些都可以大大简化管理工作。

8.3.3　RAID 等级

SNIA.Berkeley 等组织机构把 RAID0、RAID1、RAID2、RAID3、RAID4、RAID5、RAID6 七个等级定为标准的 RAID 等级，这也被业界和学术界所公认。标准等级是最基本的 RAID 配置集合，单独或综合利用数据条带、镜像和数据校验技术。标准 RAID 可以组合，即 RAID 组合等级，满足对性能、安全性、可靠性要求更高的存储应用需求。下面介绍最常用的 RAID 等级。

1. RAID3

RAID3（图 8-2）是使用专用校验盘的并行访问阵列，它采用一个专用的磁盘作为校验盘，其余磁盘作为数据盘，数据按位可字节的方式交叉存储到各个数据盘中。RAID3 至少需要三块磁盘，不同磁盘上同一带区的数据作 XOR 校验，校验值写入校验盘中。RAID3 完好时读性能与 RAID0 完全一致，并行从多个磁盘条带读取数据，性能非常高，同时还提供了数据容错能力。向 RAID3 写入数据时，必须计算与所有同条带的校验值，并将新校验值写入校验盘中。一次写操作包含了写数据块、读取同条带的数据块、计算校验值、写入校验值等多个操作，系统开销非常大，性能较低。

如果 RAID3 中某一磁盘出现故障，不会影响数据读取，可以借助校验数据和其他完好数据来重建数据。假如所要读取的数据块正好位于失效磁盘，则系统需要读取所有同一条带的数据块，并根据校验值重建丢失的数据，系统性能将受到影响。当故障磁盘被更换后，系统按相同的方式重建故障盘中的数据至新磁盘。

RAID3 只需要一个校验盘，阵列的存储空间利用率高，再加上并行访问的特征，能够为高带宽的大量读写提供高性能，适用大容量数据的顺序访问应用，如影像处理、流媒体服务等。目前，RAID5 算法不断改进，在大数据量读取时能够模拟 RAID3 ，而且 RAID3 在出现坏盘时性能会大幅下降，因此常使用 RAID5 替代 RAID3 来运行具有持续性、高带宽、大量读写特征的应用。带奇偶校验码的并行传送，只能查错不能纠错。

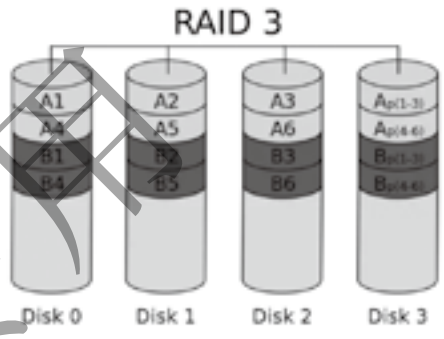

图 8-2　RAID3 带有专用位校验的数据条带

5. RAID4

RAID4（图 8-3）与 RAID3 的原理大致相同，区别在于条带化的方式不同。RAID4 按照块的方式来组织数据，写操作只涉及当前数据盘和校验盘两个盘，多个 I/O 请求可以同时得到处理，提高了系统性能。RAID4 按块存储可以保证单块的完整性，可以避免受到其他磁盘上同条带产生的不利影响。

RAID4 在不同磁盘上的同级数据块同样使用 XOR 校验，结果存储在校验盘中。写入数据时，RAID4 按这种方式把各磁盘上的同级数据的校验值写入校验盘，读取时进行即时校验。因此，当某块磁盘的数据块损坏，RAID4 可以通过校验值以及其他磁盘上的同级数据块进行数据重建。

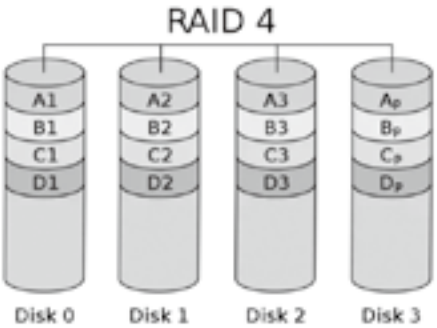

图 8-3　RAID4

RAID4 提供了非常好的读性能，但单一的校验盘往往成为系统性能的瓶颈。对于写操作，RAID4 只能一个磁盘一个磁盘地写，并且还要写入校验数据，因此写性能比较差。而且随着成员磁盘数量的增加，校验盘的系统瓶颈将更加突出。正是如上这些限制和不足，RAID4 在实际应用中很少见，主流存储产品也很少使用 RAID4 保护。

6.RAID5

RAID5 应该是目前最常见的 RAID 等级，它的原理与 RAID4 相似，区别在于校验数据分布在阵列中

的所有磁盘上，而没有采用专门的校验磁盘。对于数据和校验数据，它们的写操作可以同时发生在完全不同的磁盘上。因此，RAID5 不存在 RAID4 中的并发写操作时的校验盘性能瓶颈问题。另外，RAID5 还具备很好的扩展性。当阵列磁盘数量增加时，并行操作量的能力也随之增长，可比 RAID4 支持更多的磁盘，从而拥有更高的容量以及更高的性能。

RAID5（图 8-4）的磁盘上同时存储数据和校验数据，数据块和对应的校验信息存保存在不同的磁盘上，当一个数据盘损坏时，系统可以根据同一条带的其他数据块和对应的校验数据来重建损坏的数据。与其他 RAID 等级一样，重建数据时，RAID5 的性能会受到较大的影响。

RAID5 兼顾存储性能、数据安全和存储成本等各方面因素，它可以理解为 RAID0 和 RAID1 的折中方案，是目前综合性能最佳的数据保护解决方案。RAID5 基本上可以满足大部分的存储应用需求，数据中心大多采用它作为应用数据的保护方案。

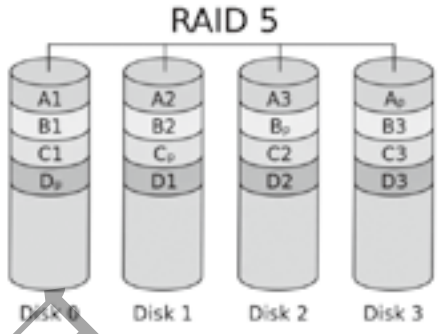

图 8-4　RAID5：带分散校验的数据条带

8.3.4　创建 RAID5

下面以 4 块硬盘 /dev/sdc、/dev/sdd、/dev/sde、/dev/sdf 为例来讲解 RAID5 的创建方法，以实现硬盘容错，保护重要数据。（本例使用 VMware 虚拟机，并安装好四块 SCSI 硬盘）

（1）在 VMware 中添加四块 SCSI 硬盘，硬盘空间大小为 20G，具体步骤略。效果如图 8-5 所示。

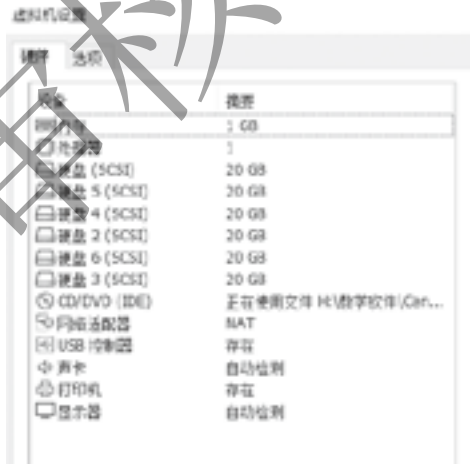

图 8-5　添加的四块硬盘（3-6）

（2）启动系统后，使用磁盘分区命令 fdisk 查看硬盘信息，发现已经新增加了四块硬盘。

```
[root@CentOS 7 ~]# fdisk -l
//……前面部分省略
磁盘 /dev/sde：21.5 GB, 21474836480 字节，41943040 个扇区
Units = 扇区 of 1 * 512 = 512 bytes
扇区大小 ( 逻辑 / 物理 )：512 字节 / 512 字节
```

```
I/O 大小 ( 最小 / 最佳 )：512 字节 / 512 字节

磁盘 /dev/sdd：21.5 GB, 21474836480 字节，41943040 个扇区
Units = 扇区 of 1 * 512 = 512 bytes
扇区大小 ( 逻辑 / 物理 )：512 字节 / 512 字节
I/O 大小 ( 最小 / 最佳 )：512 字节 / 512 字节

磁盘 /dev/sdf：21.5 GB, 21474836480 字节，41943040 个扇区
Units = 扇区 of 1 * 512 = 512 bytes
扇区大小 ( 逻辑 / 物理 )：512 字节 / 512 字节
I/O 大小 ( 最小 / 最佳 )：512 字节 / 512 字节

磁盘 /dev/sdc：21.5 GB, 21474836480 字节，41943040 个扇区
Units = 扇区 of 1 * 512 = 512 bytes
扇区大小 ( 逻辑 / 物理 )：512 字节 / 512 字节
I/O 大小 ( 最小 / 最佳 )：512 字节 / 512 字节
```

（3）使用 fdisk 命令将四块硬盘分别创建 1 个逻辑分区或者主分区，并将分区类型改为：Linux raid autodetect。

```
[root@CentOS 7 ~]# fdisk /dev/sdc
命令 ( 输入 m 获取帮助 )：n
Partition type:
p   primary (0 primary, 0 extended, 4 free)
e   extended
Select (default p): e   # 扩展分区
分区号 (1-4，默认 1)：
起始扇区 (2048-41943039，默认为 2048)：
将使用默认值 2048
Last 扇区 , + 扇区 or +size{K,M,G} (2048-41943039，默认为 41943039)：+5G
分区 1 已设置为 Extended 类型，大小设为 5 GiB

命令 ( 输入 m 获取帮助 )：n
Partition type:
p   primary (0 primary, 1 extended, 3 free)
l   logical (numbered from 5)  # 在扩展分区基础上建立逻辑分区
Select (default p): l
添加逻辑分区 5
起始扇区 (4096-10487807，默认为 4096)：
```

```
将使用默认值 4096
Last 扇区，+ 扇区 or +size{K,M,G} (4096-10487807，默认为 10487807)：
将使用默认值 10487807
分区 5 已设置为 Linux 类型，大小设为 5 GiB

命令 ( 输入 m 获取帮助 )：t
分区号 (1,5，默认 5)：
Hex 代码 ( 输入 L 列出所有代码 )：fd   # 更改分区类型
已将分区“Linux”的类型更改为“Linux raid autodetect”

命令 ( 输入 m 获取帮助 )：p

磁盘 /dev/sdc：21.5 GB, 21474836480 字节，41943040 个扇区
Units = 扇区 of 1 * 512 = 512 bytes
扇区大小 ( 逻辑 / 物理 )：512 字节 / 512 字节
I/O 大小 ( 最小 / 最佳 )：512 字节 / 512 字节
磁盘标签类型：dos
磁盘标志符：0xfc750449

设备 Boot      Start         End      Blocks   Id  System
/dev/sdc1            2048    10487807     5242880    5  Extended
/dev/sdc5            4096    10487807     5241856   fd  Linux raid autodetect

命令 ( 输入 m 获取帮助 )：w   # 保存
The partition table has been altered!

Calling ioctl() to re-read partition table.
正在同步磁盘。
```

（4）用同样的方法创建其他三个磁盘分区，最后用 fdisk –l 查看分区结果。

```
[root@CentOS 7 ~]# fdisk -l /dev/sdc /dev/sdd /dev/sde /dev/sdf

磁盘 /dev/sdc：21.5 GB, 21474836480 字节，41943040 个扇区
Units = 扇区 of 1 * 512 = 512 bytes
扇区大小 ( 逻辑 / 物理 )：512 字节 / 512 字节
I/O 大小 ( 最小 / 最佳 )：512 字节 / 512 字节
磁盘标签类型：dos
磁盘标志符：0xfc750449
```

```
设备 Boot     Start       End     Blocks   Id  System
/dev/sdc1         2048   10487807    5242880   5  Extended
/dev/sdc5         4096   10487807    5241856   fd  Linux raid autodetect

磁盘 /dev/sdd：21.5 GB, 21474836480 字节，41943040 个扇区
Units = 扇区 of 1 * 512 = 512 bytes
扇区大小 ( 逻辑 / 物理 )：512 字节 / 512 字节
I/O 大小 ( 最小 / 最佳 )：512 字节 / 512 字节
磁盘标签类型：dos
磁盘标志符：0x4527f577

设备 Boot     Start       End     Blocks   Id  System
/dev/sdd1         2048   10487807    5242880   5  Extended
/dev/sdd5         4096   10487807    5241856   fd  Linux raid autodetect

磁盘 /dev/sde：21.5 GB, 21474836480 字节，41943040 个扇区
Units = 扇区 of 1 * 512 = 512 bytes
扇区大小 ( 逻辑 / 物理 )：512 字节 / 512 字节
I/O 大小 ( 最小 / 最佳 )：512 字节 / 512 字节
磁盘标签类型：dos
磁盘标志符：0x5ec86cb0

设备 Boot     Start       End     Blocks   Id  System
/dev/sde1         2048   10487807    5242880   5  Extended
/dev/sde5         4096   10487807    5241856   fd  Linux raid autodetect

磁盘 /dev/sdf：21.5 GB, 21474836480 字节，41943040 个扇区
Units = 扇区 of 1 * 512 = 512 bytes
扇区大小 ( 逻辑 / 物理 )：512 字节 / 512 字节
I/O 大小 ( 最小 / 最佳 )：512 字节 / 512 字节
磁盘标签类型：dos
磁盘标志符：0xbc434961

设备 Boot     Start       End     Blocks   Id  System
/dev/sdf1         2048   10487807    5242880   5  Extended
/dev/sdf5         4096   10487807    5241856   fd  Linux raid autodetect
```

（5）在 Linux 系统中建立软 RAID 可以使用 mdadm 命令创建和管理 RAID 设备。

mdadm 是 multiple devices admin 的简称，它是 Linux 下的一款标准的软件 RAID 管理工具，作者是 Neil Brown。

mdadm 的基本语法：

```
mdadm [mode] <raid-device> [options] <component-devices>
```

目前 mdadm 支持 LINEAR，RAID0（striping），RAID1（mirroring），RAID4，RAID5，RAID6，RAID10，MULTIPATH 和 FAULTY 等。它有七种模式，分别是：

assemble：加入一个以前定义的阵列。

build：创建一个没有超级块的阵列。

create：创建一个新的阵列，每个设备具有超级块。

manage：管理阵列（如添加和删除）。

misc：允许单独对阵列中的某个设备进行操作（如停止阵列）。

follow or Monitor：监控 RAID 的状态。

grow：改变 RAID 的容量或阵列中的设备数目。

另外，mdadm 选项有如下：

-A，--assemble：加入一个以前定义的阵列。

-B，--build：创建一个没有超级块的阵列（Build a legacy array without superblocks）。

-C，--create：创建一个新的阵列。

-F，--follow，--monitor：选择监控（Monitor）模式。

-G，--grow：改变激活阵列的大小或形态。

-I，--incremental：添加一个单独的设备到合适的阵列，并可能启动阵列。

--auto-detect：请求内核启动任何自动检测到的阵列。

-h，--help：帮助信息，用在以上选项后，则显示该选项信息。

--help-options：显示更详细的帮助。

-V，--version：打印 mdadm 的版本信息。

-v，--verbose：显示细节。

-b，--brief：较少的细节。用于 --detail 和 --examine 选项。

-Q，--query：查看一个 device，判断它为一个 md device 或是一个 md 阵列的一部分。

-D，--detail：打印一个或多个 md device 的详细信息。

-E，--examine：打印 device 上的 md superblock 的内容。

-c，--config=：指定配置文件，缺省为 /etc/mdadm.conf。

-s，--scan：扫描配置文件或 /proc/mdstat 以搜寻丢失的信息。配置文件 /etc/mdadm.conf。

【例 12】 使用 mdadm 命令将 /dev/sd[c-f] 四块硬盘创建 RAID5。

```
[root@CentOS 7 ~]# mdadm -C /dev/md255 -ayes -l5 -n3 -x1 /dev/sd[c,d,e,f]5
mdadm: Defaulting to version 1.2 metadata
mdadm: array /dev/md255 started.
```

上述命令中指定 RAID 设备名称为 /dev/md255，级别为 5，使用 3 个设备建立 RAID，空余一个留作备份。上面的语法中，最后面是设备文件名，这些设备可以是整个磁盘，例如 /dev/sdc，也可以是硬盘上的分区，例如 /dev/sdc5 之类。不过，这些设备文件名的总数必须等于“-n3”和“-x1”的个数总和。本例中，/dev/sd[c，d，e，f]5 也可以用 /dev/sd[c-f]5 来简写，表示 /dev/sdc5、/dev/sdd5、/dev/sde5、/dev/sdf5，其中 /dev/sdf5 为备用。另外，以上命令也可以用选项的英文全称，如：

```
[root@CentOS 7 ~]# mdadm --create /dev/md255 --level=5 --raid-devices=3
--spare-devices=1 /dev/sd[c-f]5
```

（6）为新建立的 /dev/md255 建立类型为 ext4 的文件系统。

```
[root@CentOS 7 ~]# mkfs -t ext4 -c /dev/md255
```

（7）查看建立的 RAID5 的具体情况。

```
[root@CentOS 7 ~]# mdadm --detail /dev/md255
/dev/md255:
Version : 1.2
Creation Time : Wed Feb 12 22:06:36 2020
Raid Level : raid5
Array Size : 10473472 (9.99 GiB 10.72 GB)
Used Dev Size : 5236736 (4.99 GiB 5.36 GB)
Raid Devices : 3
Total Devices : 4
Persistence : Superblock is persistent

Update Time : Wed Feb 12 22:34:56 2020
State : clean
Active Devices : 3
Working Devices : 4
Failed Devices : 0
Spare Devices : 1

Layout : left-symmetric
Chunk Size : 512K

Consistency Policy : resync

Name : CentOS 7.8:255  (local to host CentOS 7.8)
UUID : 5a4d95b5:3d1ecc6e:a5074b21:47698ae4
```

```
Events : 18

Number   Major   Minor   RaidDevice State
0      8      37       0      active sync   /dev/sdc5
1      8      53       1      active sync   /dev/sdd5
4      8      69       2      active sync   /dev/sde5

3      8      85       -      spare   /dev/sdf5
```

（8）将 RAID5 设备挂载到 /mnt/md255 目录中，并在该设备中新建测试文件 text.txt 和 raid5.doc 两个文件。

```
[root@CentOS 7 ~]# mkdir /mnt/md255
[root@CentOS 7 ~]# mount /dev/md255 /mnt/md255
[root@CentOS 7 ~]# touch /mnt/md255/text.txt raid5.doc
[root@CentOS 7 ~]# ls /mnt/md255
lost+found  text.txt
```

此刻，RAID5 创建成功。

（9）如果 RAID 设备中某个硬盘发生故障，系统会自动停止该硬盘的工作。它会让备份磁盘 /dev/sdf5 代替损坏的磁盘继续工作。例如，假设 /dev/sdd5 损坏，更换损坏的 RAID 设备中成员的方法如下：

将损坏的 RAID 磁盘标记为失效。

```
[root@CentOS 7 ~]# mdadm /dev/md255 --fail /dev/sdd5
mdadm: set /dev/sdd5 faulty in /dev/md255
[root@CentOS 7 ~]# mdadm --detail /dev/md255  # 查看损坏的磁盘
/dev/md255:
// 中间省略部分
Number   Major   Minor   RaidDevice State
0      8      37       0      active sync     /dev/sdc5
3      8      85       1      spare rebuilding /dev/sdf5# 备份盘已替换
4      8      69       2      active sync     /dev/sde5

1      8      53       -      faulty   /dev/sdd5 # 标记为失效
```

移除失效的 RAID 成员。

```
[root@CentOS 7 ~]# mdadm /dev/md255 --remove /dev/sdd5
mdadm: hot removed /dev/sdd5 from /dev/md255
```

如果系统没有自动更换备份硬盘设备，可以手动添加 /dev/sdb5。

```
[root@CentOS 7 ~]# mdadm /dev/md255 --add /dev/sdb5
```

注意：mdadm 命令参数中凡是以两个短线“--”引出的参数选项，与“-”加单词首字母的方式等价。如“--detail”等价于“-D”。

当不再使用 RAID 设备时，可以使用命令“mdadm -S /dev/md255”的方式停止 RAID5 设备。

8.4 配置与管理逻辑卷

LVM 是 Logical Volume Manager（逻辑卷管理）的简写，它是 Linux 环境下对磁盘分区进行管理的一种机制，它由 Heinz Mauelshagen 在 Linux 2.4 内核上实现，目前最新版本为：稳定版 1.0.5，开发版 1.1.0-rc2，以及 LVM2 开发版。Linux 用户安装 Linux 操作系统时遇到的一个常见的难以决定的问题就是如何正确地评估各分区大小，以分配合适的硬盘空间。普通的磁盘分区管理方式在逻辑分区划分好之后就无法改变其大小，当一个逻辑分区存放不下某个文件时，这个文件因为受上层文件系统的限制，也不能跨越多个分区来存放，所以也不能同时放到别的磁盘上。而遇到出现某个分区空间耗尽时，解决的方法通常是使用符号链接，或者使用调整分区大小的工具，但这只是暂时解决办法，没有从根本上解决问题。随着 Linux 的逻辑卷管理功能的出现，这些问题都迎刃而解，用户在无须停机的情况下可以方便地调整各个分区大小。

与传统的磁盘与分区相比，LVM 为计算机提供了更高层次的磁盘存储。它使系统管理员可以更方便地为应用与用户分配存储空间。在 LVM 管理下的存储卷可以按需要随时改变大小与移除（可能需对文件系统工具进行升级）。LVM 也允许按用户组对存储卷进行管理，允许管理员用更直观的名称代替物理磁盘名（如“sda”“sdb”）来标记存储卷。

8.4.1 LVM 的基本组成块（building blocks）

1. 物理卷 Physical volume(PV)

我们实际的 partition（或 Disk）需要调整系统识别码（system ID）成为 8e（LVM 的识别码），然后再经过 pvcreate 的指令将他转成 LVM 最底层的实体卷轴（PV），之后才能够将这些 PV 加以利用。system ID 是通过 fdisk 命令调整的。

2. 卷组 Volume group (VG)

所谓的 LVM 大磁盘就是将许多 PV 整合成这个 VG 的东西，所以 VG 就是 LVM 组合起来的大磁盘！这么想就好了。那么这个大磁盘最大可以到多少容量呢？这与下面要说明的 PE 以及 LVM 的格式版本有关。在默认的情况下，使用 32 位的 Linux 系统时，基本上 LV 最大仅能支持到 65534 个 PE，若使用默认的 PE 为 4MB 的情况下，最大容量则仅能达到约 256GB，不过这个问题在 64 位的 Linux 系统上面已经不存在，LV 几乎没有容量限制。

3. 物理区域 Physical extent (PE)

LVM 默认使用 4MB 的 PE 区块，而 LVM 的 LV 在 32 位系统上最多仅能含有 65534 个 PE （lvm1 的格式），因此默认的 LVM 的 LV 会有 4M × 65534 ÷（1024M/G）=256G（限于 lvm1 版本）。这个 PE 就有点像文件系统里面的 block 的性质一样。在使用 lvm2 的版本中，以及系统转为 64 位，因此这个限制已经不存在了。

4. 逻辑卷 Logical volume (LV)

虚拟分区，由物理区域（physical extents）组成，类似于非 LVM 系统中的硬盘分区，在逻辑卷之上能建立文件系统（比如 /home 或 /usr 等）。

最终的 VG 还会被切成 LV，这个 LV 就是最后可以被格式化使用的类似分区的东西了！那么 LV 是否可以随意指定大小呢？当然不可以！既然 PE 是整个 LVM 的最小储存单位，那么 LV 的大小就与在此 LV 内的 PE 总数有关。为了方便使用者利用 LVM 来管理其系统，因此 LV 的设备文件名通常指定为“/dev/vgname/lvname”的样式！

5. 逻辑盘区：LE（Logical Extent）

逻辑盘区是逻辑卷中可用于分配的最小的存储单元，逻辑盘区的大小取决于逻辑卷在卷组物理盘区的大小。LE 的大小为 PE 的倍数（通常为 1 : 1）。

LVM 的基本组成块如图 8-6 所示。

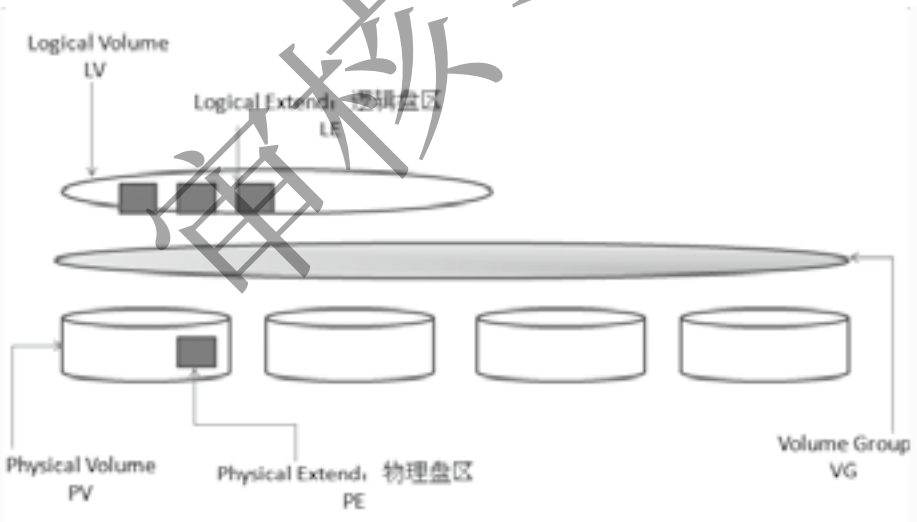

图 8-6　LVM 的基本组成块

LVM 进行逻辑卷管理时，创建顺序是 PV-VG-LV。也就是说，首先创建一个物理卷（对应一个物理硬盘分区或者一个物理硬盘），然后把这些分区或者硬盘加入到一个卷组中（相当于一个逻辑上的大硬盘），再在这个大硬盘上划分分区 LV（逻辑上的分区，就是逻辑卷），最后，把 LV 逻辑卷格式化后，就可以像使用一个传统分区一样，把它挂载到一个挂载点上，需要的时候，这个逻辑卷可以被动态地缩放。

8.4.2 创建 LVM

创建 LVM 系统分区方式的主要步骤如下：

添加新的物理磁盘→创建新的分区→创建 PV →创建 VG →创建 LV →格式化 LV，并添加文件系统 ext4 →挂载 LV 到指定目录→将挂载信息写入：/etc/fstab。

物理卷可以建立在整个物理硬盘上，也可以建立在硬盘分区中。如在整个硬盘上建立物理卷则不要在该硬盘上建立任何分区，如使用硬盘分区建立物理卷则要事先对硬盘进行分区并设置该分区为 LVM 类型，其类型 ID 为 8e。

启动虚拟机通过 pvs 命令查看物理卷的情况，目前只看到有虚拟机初始安装时有个 pv 为 /dv/sda2 lvm 为 centso 的物理卷大小大于 19G

```
[root@CentOS 7 ~]# pvs
PV         VG     Fmt  Attr PSize   PFree
/dev/sda2  centos lvm2 a--  <19.00g    0
```

1. 添加新的物理磁盘

在 VMare 虚拟机中添加一块 20G 的硬盘：/dev/sdb。重启系统后，使用 fdisk 命令查看。

```
[root@CentOS 7 ~]# fdisk -l /dev/sdb

磁盘 /dev/sdb：21.5 GB, 21474836480 字节，41943040 个扇区
Units = 扇区 of 1 * 512 = 512 bytes
扇区大小 ( 逻辑 / 物理 )：512 字节 / 512 字节
I/O 大小 ( 最小 / 最佳 )：512 字节 / 512 字节
磁盘标签类型：dos
磁盘标志符：0x67f6f599

设备 Boot      Start         End      Blocks   Id  System
```

2. 创建分区

对硬盘创建四个新的分区，并设置该分区为 LVM 类型，其类型 ID 为 8e。

```
[root@CentOS 7 ~]# fdisk /dev/sdb
// 前面部分省略
命令 ( 输入 m 获取帮助 )：n
Partition type:
p   primary (0 primary, 0 extended, 4 free)
e   extended
Select (default p): p
分区号 (1-4，默认 1)：
起始扇区 (2048-41943039，默认为 2048)：
```

```
将使用默认值 2048
Last 扇区，+ 扇区 or +size{K,M,G} (2048-41943039，默认为 41943039)：+100M
分区 1 已设置为 Linux 类型，大小设为 100 MiB

命令 ( 输入 m 获取帮助 )：t
已选择分区 1
Hex 代码 ( 输入 L 列出所有代码 )：8e
已将分区“Linux”的类型更改为“Linux LVM”
// 省略其他三个分区的创建步骤
命令 ( 输入 m 获取帮助 )：p

磁盘 /dev/sdb：21.5 GB, 21474836480 字节，41943040 个扇区
Units = 扇区 of 1 * 512 = 512 bytes
扇区大小 ( 逻辑 / 物理 )：512 字节 / 512 字节
I/O 大小 ( 最小 / 最佳 )：512 字节 / 512 字节
磁盘标签类型：dos
磁盘标志符：0x67f6f599

设备         Boot      Start     End     Blocks   Id      System
/dev/sdb1   2048      206847   102400   8e      Linux   LVM
/dev/sdb2   206848    411647   102400   8e      Linux   LVM
/dev/sdb3   411648    616447   102400   8e      Linux   LVM
/dev/sdb4   616448    821247   102400   8e      Linux   LVM
命令 ( 输入 m 获取帮助 )：w
The partition table has been altered!

Calling ioctl() to re-read partition table.
正在同步磁盘。
```

3. 创建 PV

将物理硬盘格式化成 PV（物理卷）使用的是 pvcreate 命令。

```
[root@CentOS 7 ~]# pvcreate /dev/sdb1
WARNING:ext3 signature detected on /dev/sdb1 at offset 1080.Wipe it? [y/n]: y
Wiping ext3 signature on /dev/sdb1.
WARNING: dos signature detected on /dev/sdb1 at offset 510. Wipe it? [y/n]: y
Wiping dos signature on /dev/sdb1.
Physical volume “/dev/sdb1” successfully created.
```

使用 pvdisplay 或 pvs 查看当前的 pv 信息。

```
[root@CentOS 7 ~]# pvdisplay /dev/sdb1
  “/dev/sdb1” is a new physical volume of “100.00 MiB”
--- NEW Physical volume ---
PV Name               /dev/sdb1
VG Name
PV Size               100.00 MiB
Allocatable           NO
PE Size               0
Total PE              0
Free PE               0
Allocated PE          0
PV UUID               3GzHKa-Ccup-ieHY-sIuO-YuGe-JO0A-mJZPLb
```

使用同样的方法建立 /dev/sdb3、/dev/sdb4。

4. 创建卷组 VG

在创建好物理 PV 物理卷后，使用 vgcreate 命令创建卷组。卷组设备文件使用 /dev 目录下与卷组同名的目录表示。该卷组中的所有逻辑设备文件都将建立在该目录下。卷组中可以包含一个或者多个物理卷。下面使用 vgcreate 命令创建卷组 vg1。

```
[root@CentOS 7 ~]# vgcreate vg1 /dev/sdb1
Volume group “vg1” successfully created
```

使用同样方法创建 vg2、vg3、vg4。然后通过 vgdisplay 或 vgs 命令查看 vg 的信息。

```
[root@CentOS 7 ~]# vgs
VG     #PV #LV #SN Attr   VSize   VFree
centos   1   2   0 wz--n- <19.00g     0
vg1      1   0   0 wz--n-  96.00m 96.00m
vg2      1   0   0 wz--n-  96.00m 96.00m
vg3      1   0   0 wz--n-  96.00m 96.00m
vg4      1   0   0 wz--n-  96.00m 96.00m
```

看到 vg 已经创建好了，大小是分区时的大小 100M，这里显示 96M。通过 vgdisplay vg1 查看显示详细信息可知，PE 值使用默认的 4MB，如果需要增大可以使用 -L 选项进行修改，但是一旦设定以后不可以再更改 PE 值。

5. 创建 LV

基于卷组（VG）创建逻辑卷（LV）通过 lvcreate 命令。

```
// 基于 vg1 创建逻辑卷名为 lv1, 大小为 50M
[root@CentOS 7 ~]# lvcreate -n lv1 -L 50M vg1
Rounding up size to full physical extent 52.00 MiB
Logical volume “lv1” created.
```

其中 -L 选项用于设置逻辑卷大小，-n 参数用于指定逻辑卷的名称和卷组名称。

用 lvdisplay 或 lvs 命令查看创建好的逻辑卷。可以看到名字为 lv1 的逻辑卷已经创建好了，它是基于 vg1 创建的，大小为 52M。

```
[root@CentOS 7 ~]# lvs
LV    VG     Attr    LSize Pool Origin Data% Meta% Move Log Cpy%Sync Convert
root centos -wi-ao---- <17.00g
swap centos -wi-ao----   2.00g
lv1  vg1    -wi-a-----  52.00m
```

要查看 lv1 的详细信息可以使用 lvdisplay 命令，逻辑卷存放在 /dev 中的卷组中。

```
[root@CentOS 7 ~]# lvdisplay /dev/vg1/lv1
--- Logical volume ---
LV Path                /dev/vg1/lv1
LV Name                lv1
VG Name                vg1
LV UUID                kFh3Cj-PzcM-Nalz-u6tY-WmjX-3eBB-N6w2Ip
LV Write Access        read/write
LV Creation host, time CentOS 7.8, 2020-02-14 14:24:26 +0800
LV Status              available
# open                 0
LV Size                52.00 MiB
Current LE             13
Segments               1
Allocation             inherit
Read ahead sectors     auto
- currently set to     8192
Block device           253:2
```

到这里，lv 就创建好了，但是要用起来，还得格式化并挂载到我们的文件系统。

6. 格式化

格式化 LV1，并添加文件系统 ext4。

使用 mkfs 格式化逻辑卷 /dev/vg1/lv1 ，并添加到 ext4 文件系统。

```
[root@CentOS 7 ~]# mkfs -t ext4 /dev/vg1/lv1
mke2fs 1.42.9 (28-Dec-2013)
文件系统标签 =
OS type: Linux
块大小 =1024 (log=0)
分块大小 =1024 (log=0)
Stride=0 blocks, Stripe width=0 blocks
13328 inodes, 53248 blocks
2662 blocks (5.00%) reserved for the super user
第一个数据块 =1
Maximum filesystem blocks=33685504
7 block groups
8192 blocks per group, 8192 fragments per group
1904 inodes per group
Superblock backups stored on blocks:
8193, 24577, 40961

Allocating group tables: 完成
正在写入 inode 表 : 完成
Creating journal (4096 blocks): 完成
Writing superblocks and filesystem accounting information: 完成
```

7. 挂载 LV

将逻辑卷挂载到指定目录，也可以直接修改 /etc/fstab，让它开机自动启动。

```
[root@CentOS 7 ~]# mkdir /lv1   # 创建挂载点
[root@CentOS 7 ~]# mount /dev/vg1/lv1 /lv1   # 挂载逻辑卷
[root@CentOS 7 ~]# df -h    # 查看挂载情况
文件系统                 容量    已用    可用    已用 %  挂载点
devtmpfs                 470M    0       470M    0%      /dev
tmpfs                    487M    0       487M    0%      /dev/shm
tmpfs                    487M    8.4M    478M    2%      /run
tmpfs                    487M    0       487M    0%      /sys/fs/cgroup
/dev/mapper/centos-root  17G     4.3G    13G     26%     /
/dev/sda1                1014M   171M    844M    17%     /boot
```

```
tmpfs                   98M   12K   98M   1%   /run/user/42
tmpfs                   98M   0     98M   0%   /run/user/0
/dev/mapper/vg1-lv1     47M   1.1M  42M   3%   /lv1          // 挂载成功
```

8. 写入挂载信息

挂载成功后，就可以在逻辑卷上写入文件了。

设置开机加载，将挂载信息写入：/etc/fstab

```
[root@CentOS 7 ~]#echo “/dev/vg/app /app ext4 defaults 0 0” >>/etc/fstab
```

8.4.3　扩容 LVM 逻辑卷

这里扩容分两种情况：一种情况是 VG 还有足够的空间，那么就可以直接扩容 LV 就可以了。另外一种情况是要扩容的空间已经超过了 VG 的大小，那么就可以通过加物理磁盘扩充到 VG 里，然后再扩 LV。

1. 在 VG 的容量范围之内扩大空间大小

```
[root@CentOS 7 ~]# vgs
VG     #PV #LV #SN Attr   VSize   VFree
centos   1   2   0 wz--n- <19.00g    0
vg1      1   1   0 wz--n-  96.00m 44.00m
```

现在 vg1 的大小为 96M，/lv1 是 50M，计划扩容到 80M，没有超过 vg 的大小那么可以直接扩容 lv 就可以了。

第一步：首先卸载设备和挂载点的关联

```
[root@CentOS 7 ~]# umount /lv1
```

第二步：将逻辑卷 /dev/vg1/lv1 扩展到 80M

使用 lvextend –L 8G /dev/vg1/lv1 命令，可以清楚地看到 vg1/lv1 从 50M 扩容到了 80M。

```
[root@CentOS 7 ~]# lvextend -L 80M /dev/vg1/lv1
Size of logical volume vg1/lv1 changed from 52.00 MiB (13 extents) to 80.00 MiB (20 extents).// 增加到 80M
Logical volume vg1/lv1 successfully resized.
```

第三步：检查硬盘（lv1）完整性，并重置硬盘（lv1）容量

（1）e2fsck –f /dev/vg1/lv1 检查硬盘完整性。

```
[root@CentOS 7 ~]# e2fsck -f /dev/vg1/lv1
e2fsck 1.42.9 (28-Dec-2013)
```

```
第一步：检查 inode, 块 , 和大小
第二步：检查目录结构
第三步：检查目录连接性
Pass 4: Checking reference counts
第五步：检查簇概要信息
/dev/vg1/lv1: 13/13328 files (7.7% non-contiguous), 6825/53248 blocks
```

（2）resize2fs /dev/vg/app 重置硬盘（lv1）容量，这一步必须做，否则即使扩了容量，但看到的还是扩容之前的容量。

```
[root@CentOS 7 ~]# resize2fs /dev/vg1/lv1
resize2fs 1.42.9 (28-Dec-2013)
Resizing the filesystem on /dev/vg1/lv1 to 81920 (1k) blocks.
The filesystem on /dev/vg1/lv1 is now 81920 blocks long.
```

第四步：重新挂载硬盘并查看

```
[root@CentOS 7 ~]# mount /dev/vg1/lv1 /lv1
[root@CentOS 7 ~]# df -h
文件系统                 容量  已用  可用  已用 %  挂载点
devtmpfs                 470M  0     470M  0%     /dev
tmpfs                    487M  0     487M  0%     /dev/shm
tmpfs                    487M  8.4M  478M  2%     /run
tmpfs                    487M  0     487M  0%     /sys/fs/cgroup
/dev/mapper/centos-root  17G   4.3G  13G   26%    /
/dev/sda1                1014M 171M  844M  17%    /boot
tmpfs                    98M   12K   98M   1%     /run/user/42
tmpfs                    98M   0     98M   0%     /run/user/0
/dev/mapper/vg1-lv1      74M   1.6M  67M   3%     /lv1   // 已增加到 74M
```

使用 ls 查看 /lv1 里面的文件还在，说明扩容对文件数据没有影响。

如果扩容的大小超过了 vg 的大小怎么办呢？可以通过扩硬件的方式，加块硬盘到 vg 然后再扩 lv。

2. 扩容的大小超过了 VG 的大小

现在 vg1 的大小为 96M，/lv1 是 74M，计划扩展到 20G，已经超过 vg1 的 96M 大小，那么就需要先加硬盘，然后扩容 vg1，再扩容 lv1。

```
[root@CentOS 7 lv1]# vgdisplay /dev/vg1
--- Volume group ---
VG Name               vg1
System ID
```

```
  Format                lvm2
  Metadata Areas        1
  Metadata Sequence No  3
  VG Access             read/write
  VG Status             resizable
  MAX LV                0
  Cur LV                1
  Open LV               1
  Max PV                0
  Cur PV                1
  Act PV                1
  VG Size               96.00 MiB  #VG 大小 96M
  PE Size               4.00 MiB
  Total PE              24
  Alloc PE / Size       20 / 80.00 MiB
  Free  PE / Size       4 / 16.00 MiB
  VG UUID               cW9dOQ-yCNA-NzmG-e414-1fRz-17fM-Q8WJh7
```

第一步：添加一个 20G 硬盘（步骤略），本次使用 /dev/sdc 硬盘。也可以将硬盘中的一个或多个分区扩展到 VG 里，请读者自己测试。

第二步：扩容 vg1。将新的硬盘扩到 vg1 卷组里。

先卸载挂载点 /lv1，再添加硬盘 /dev/sdc 添加到 vg1 卷组里。

```
[root@CentOS 7 ~]# umount /lv1
[root@CentOS 7 ~]# vgextend vg1 /dev/sdc # 将 /dev/sdc 硬盘添加到 vg1
Physical volume “/dev/sdc” successfully created.
Volume group “vg1” successfully extended
```

第三步：扩容 lv1 将逻辑卷 /dev/vg1/lv1 扩展 20G。

使用命令 lvextend –L 20G /dev/vg1/lv1 ，可以看到 vg1/lv1 从原来的 80M 扩展到 20G。

```
[root@CentOS 7 ~]# lvextend -L 20G /dev/vg1/lv1
Size of logical volume vg1/lv1 changed from 80.00 MiB (20 extents) to 20.00 GiB (5120 extents).  # 从原来的 80M 扩展到 20G
Logical volume vg1/lv1 successfully resized.
[root@CentOS 7 ~]# lvdisplay /dev/vg1/lv1   # 查看 lv1 的详细情况
--- Logical volume ---
LV Path                /dev/vg1/lv1
LV Name                lv1
VG Name                vg1
```

```
LV UUID                kFh3Cj-PzcM-Nalz-u6tY-WmjX-3eBB-N6w2Ip
LV Write Access        read/write
LV Creation host, time CentOS 7.8, 2020-02-14 14:24:26 +0800
LV Status              available
# open                 0
LV Size                20.00 GiB  # 已扩展到 20G
Current LE             5120
Segments               2
Allocation             inherit
Read ahead sectors     auto
- currently set to     8192
Block device           253:2
```

第四步：同样检查硬盘（lv1）完整性，并重置硬盘（lv1）容量。

使用 e2fsck -f /dev/vg1/lv1 命令检查硬盘完整性，使用 resize2fs /dev/vg1/lv1 命令重置硬盘（lv）容量。

```
[root@CentOS 7 ~]# e2fsck -f /dev/vg1/lv1  # 检查硬盘完整性
e2fsck 1.42.9 (28-Dec-2013)
第 1 步：检查 inode, 块 , 和大小
第 2 步：检查目录结构
第 3 步：检查目录连接性
第 4 步：  Checking reference counts
第 5 步：检查簇概要信息
/dev/vg1/lv1: 13/19040 files (7.7% non-contiguous), 8061/81920 blocks
[root@CentOS 7 ~]# resize2fs /dev/vg1/lv1   # 重置硬盘 (lv) 容量
resize2fs 1.42.9 (28-Dec-2013)
Resizing the filesystem on /dev/vg1/lv1 to 20971520 (1k) blocks.
The filesystem on /dev/vg1/lv1 is now 20971520 blocks long.
```

第五步：重新挂载硬盘并查看扩展情况。

```
[root@CentOS 7 ~]# mount /dev/vg1/lv1 /lv1
[root@CentOS 7 ~]# df -h
文件系统                 容量    已用   可用    已用 %   挂载点
devtmpfs                 470M    0      470M    0%       /dev
tmpfs                    487M    0      487M    0%       /dev/shm
tmpfs                    487M    21M    466M    5%       /run
tmpfs                    487M    0      487M    0%       /sys/fs/cgroup
/dev/mapper/centos-root  17G     4.3G   13G     26%      /
```

```
/dev/sda1                1014M 171M 844M 17%  /boot
tmpfs                    98M   12K  98M  1%   /run/user/42
tmpfs                    98M   0    98M  0%   /run/user/0
/dev/mapper/vg1-lv1      20G   1.7M 19G  1%   /lv1          // 已扩展到 20G
```

可以看到 /lv1 已经成功扩容到 20G 了，并且 /lv1 目录下的文件不受影响。

8.4.4　缩小 LVM 逻辑卷

相对于逻辑卷扩容，缩小逻辑卷，数据丢失的风险更大。所以在生产环境中操作一定要注意提前备份好数据。在对 LVM 逻辑卷进行缩小操作之前，先把要缩小的文件系统卸载并检查文件系统的完整性。

现在我们将 /lv1 由现在的 20G 缩到 10G：

第一步：卸载 /app 并检查文件系统完整性

```
[root@CentOS 7 ~]# umount /lv1
[root@CentOS 7 ~]# e2fsck -f /dev/vg1/lv1
e2fsck 1.42.9 (28-Dec-2013)
第 1 步：检查 inode, 块 , 和大小
第 2 步：检查目录结构
第 3 步：检查目录连接性
第 4 步：Checking reference counts
第 5 步：检查簇概要信息
/dev/vg1/lv1: 13/4874240 files (7.7% non-contiguous), 622899/20971520 blocks
```

第二步：把逻辑卷缩容到 10G。

使用 resize2fs /dev/vg1/lv1 10G 命令，重置硬盘 lv1 容量为 10G。使用 lvreduce -L 10G /dev/vg1/lv1 减少 lv1 的容量。

```
[root@CentOS 7 ~]# e2fsck -f /dev/vg1/lv1 # 重置硬盘 lv1 容量为 10G
e2fsck 1.42.9 (28-Dec-2013)
第 1 步：检查 inode, 块 , 和大小
第 2 步：检查目录结构
第 3 步：检查目录连接性
第 4 步：Checking reference counts
```

第 5 步：检查簇概要信息

```
/dev/vg1/lv1: 13/4874240 files (7.7% non-contiguous), 622899/20971520 blocks
[root@CentOS 7 ~]# resize2fs /dev/vg1/lv1 10G
resize2fs 1.42.9 (28-Dec-2013)
Resizing the filesystem on /dev/vg1/lv1 to 10485760 (1k) blocks.
The filesystem on /dev/vg1/lv1 is now 10485760 blocks long.
```

```
[root@CentOS 7 ~]# lvreduce -L 10G /dev/vg1/lv1  # 减少 lv1 的容量
WARNING: Reducing active logical volume to 10.00 GiB.
THIS MAY DESTROY YOUR DATA (filesystem etc.)
Do you really want to reduce vg1/lv1? [y/n]: y
Size of logical volume vg1/lv1 changed from 20.00 GiB (5120 extents) to 10.00 GiB
(2560 extents).
Logical volume vg1/lv1 successfully resized.
```

第三步：重新挂载并查看状态。

```
[root@CentOS 7 ~]# mount /dev/vg1/lv1 /lv1
[root@CentOS 7 ~]# ls /lv1  # 由于文件少，所以不受影响。
a  lost+found
[root@CentOS 7 ~]# df -h
文件系统                容量  已用  可用  已用 %  挂载点
devtmpfs                470M  0     470M  0%     /dev
tmpfs                   487M  0     487M  0%     /dev/shm
tmpfs                   487M  21M   466M  5%     /run
tmpfs                   487M  0     487M  0%     /sys/fs/cgroup
/dev/mapper/centos-root 17G   4.3G  13G   26%    /
/dev/sda1               1014M 171M  844M  17%    /boot
tmpfs                   98M   12K   98M   1%     /run/user/42
tmpfs                   98M   0     98M   0%     /run/user/0
/dev/mapper/vg1-lv1     9.8G  2.7M  9.2G  1%     /lv1          // 缩小容量为 10G
```

通过 LVM 的管理，创建、扩容、缩容，可以看到通过 LVM 技术可以实现系统存储空间的动态调整。另外，如果要检查物理卷、卷组和逻辑卷，可以分别使用 pvscan、vgscan、lvscan，在这里不再叙述。

此外，要删除 LVM，必须遵循删除逻辑卷（LV）– 卷组（VG）– 物理卷（PV）的顺序来执行。删除逻辑卷使用 lvremove 命令，删除卷组使用 vgremove 命令，删除物理卷使用 pvremove 命令。例如：

```
[root@CentOS 7 ~]# umount /lv1   # 卸载挂载点目录
[root@CentOS 7 ~]# lvremove /dev/vg1/lv1  # 删除逻辑卷
Do you really want to remove active logical volume vg1/lv1? [y/n]: y
Logical volume “lv1” successfully removed
[root@CentOS 7 ~]# vgremove vg1     # 删除卷组
Volume group “vg1” successfully removed
[root@CentOS 7 ~]# pvremove /dev/sdc  # 删除物理卷（扩展的硬盘）
Labels on physical volume “/dev/sdc” successfully wiped.
[root@CentOS 7 ~]# pvremove /dev/sdb1 #// 删除物理卷
```

```
Labels on physical volume “/dev/sdb1” successfully wiped.
```

LVM 逻辑管理器的基本原理是将多个物理硬盘创建成 pv（物理卷），这些物理卷是动态调整的物理基础，通过 vg 将 pv 管理起来形成一个整体的资源池。在 vg 中划分 lv 来动态调整逻辑卷的大小。

本章小结

通过本章的学习，我们掌握 Linux 下的磁盘管理工具的使用，并能使用管理工具进行硬盘分区管理，同时也掌握 Linux 硬盘配额管理，对不同分区进行磁盘配额，也掌握 Linux 逻辑卷管理，解决生活中磁盘空间不足的问题。

习题

一、填空题

1. 将 /dev/sdb 硬盘进行分区的命令是：_____，指定分区类型为 LVM 的编号是：_____。
2. 让内核读取最新的分区信息的命令是：_____。
3. 扫描 pv 卷的命令是：_____，将 sdb1/sdb2 指定为 PV 格式的命令是：_____。
4. 查看 PV 的详细状态：_____。
5. 用 sdb1 和 sdb2 建立 vg，vg 名为 ckvg, 并指定 PE 为 32MB。实现命令是：_____。
6. 显示名为 ckvg 的 VG 的详细信息：_____。
7. 将 sdb2 从 ckvg 中抽除，并查看 ckvg 的详细信息：_____。
8. 将 sdb2 加入到 ckvg 中，并查看 ckvg 的详细信息：_____。
9. 在 ckvg 上创建 lv, 空间大小为 1000M,vl 取名为 cklv：_____。
10. 用 ls 查看新创建的 lv：_____。
11. 搜索并显示 lv：_____。
12. 编写命令，格式化 cklv, 新建目录 /mnt/lvm, 并将 cklv 挂载到 /mnt/lvm。

_____

二、根据提示编写实现命令

1. 扩展空间

（1）卸载 cklv

_____

（2）增加新分区 /dev/sdb3

_____

（3）将 /dev/sdb3 做成 pv

_____

（4）将 /dev/sdb3 加入到 ckvg

（5）将 cklv 再扩展 1000M

（6）用 resize2fs 处理 cklv

（7）重新挂载 cklv

2. 移除 LVM

（1）先卸载系统上面的 LVM 磁区

（2）使用 lvremove 移除 LV

（3）使用 vgchange –a n VGname 让 VGname 不具有 Active 的标志

（4）使用 vgremove 移除 VG

（5）使用 pvremove 移除 PV

（6）使用 fdisk 把 ID 修改回来

✓ 综合实训

一、实训题目

LVM 逻辑卷管理器

二、实训目的

1. 掌握利用 LVM 创建磁盘分区的方法。
2. 掌握利用 Disk Druid 中的 LVM 创建磁盘分区的方法。

三、项目背景

某企业在 Linux 服务器中新增了一块硬盘 /dev/sdb，要求 Linux 系统的分区能自动调整磁盘容量。请使用 fdisk 命令在新建 /dev/sdb1、/dev/sdb2、/dev/sdb3 和 /dev/sdb4 为 LVM 类型，并在这四个分区上创建

物理卷、卷组和逻辑卷。最后将逻辑卷挂载。

四、实训内容

物理卷、卷组、逻辑卷的创建；卷组、逻辑卷的管理。

五、根据要求，完成实训步骤

1. 创建 LVM 分区

（1）利用 fdisk 命令在 /dev/sdb 上建立 LVM 类型的分区。

（2）建立物理卷

（3）建立卷组

（4）建立逻辑卷

2. LVM 逻辑卷的管理

（1）增加新的物理卷到卷组

（2）逻辑卷容量的动态调整

（3）删除逻辑卷 – 卷组 – 物理卷 (必须按照先后顺序来执行删除)

3. 物理卷、卷组和逻辑卷的检查

（1）物理卷的检查

（2）卷组的检查

（3）逻辑卷的检查

六、实训思考题

1. 怎样实现将 /dev/vg0/lv0 自动挂载到 /mnt/lv0 挂载点下？

2. 利用 LVM 逻辑卷管理器和使用 fdisk 等基本磁盘管理工具实现磁盘管理有什么不同？

第九章
进程管理

内容简介

- 当计算机安装好操作系统以后，需要人工装载程序到计算机的储存器中，然后让处理器读入并按照装载好的程序来使计算机执行程序，以使计算机发挥其计算的功用。在具有多处理核心或多线程技术的计算平台上，多个程序还可以同时被装载在一个或多个处理器核心中并行运作。本章主要介绍计算机中程序被调用后，所发起的运行活动（进程）的调用、管理和终止的方法。

学习要求

- 通过本章的学习，认识程序、进程与线程的概念与 GNU/Linux 中进程的分类，了解常见的系统进程，掌握常用的查看和监视系统进程的命令。

9.1 进程概述

进程是具有一定独立功能的程序，是关于某一个数据集合的执行活动。

9.1.1 程序、进程与线程

计算机程序（programme）是计算机为了完成某个功能的有序指令集合，一个程序通常由若干个程序段组成。在计算机编程语言中，通常将程序描述为数据结构加算法的结合体。这从本质上说明了程序的本质是逻辑实体。从概念来看，程序、运行程序所需的数据（data）和文档（document）都属于计算机软件。

当程序还没被载入计算机内存中并被处理器执行时，程序存在储存器中，此时其作用可以看作与数

据无异。如存在可执行文件 /usr/bin/tar，在没用命令调用之前它也只是躺在储存器当中。当且仅当程序被调用，并由操作系统安排适当的处理器时间片来实施程序的执行活动时，程序才能够被计算机真正所用到。

用户在终端上输入命令后，操作系统会在文件系统上寻找与该命令相对应的存在储存器上的程序文件，并将其载入内存中。经过一定的规则，分配一定的 CPU 执行时间片给处在运行状态中的程序以执行其活动，让软件在硬件上实际发挥作用，即程序的运行。对于每一个在储存器上的程序文件每一次被载入执行的活动过程便是进程（process）。在不同类型的操作系统中，进程可能被称作任务（task）或活动（activity）。

为了防止操作系统及其关键资料受用户程序有意或无意地破坏，通常将处理器的自身运行状态分成核心态和用户态。其中核心态又称管态或者系统态，是操作系统管理程序执行时处理器所处的状态，具有较高的优先权，能执行对操作系统中较为敏感的指令，也可以访问操作系统运行时较为敏感的寄存器与储存区。

GNU/Linux 的进程，包含了以下进程状态：可运行态、运行态、暂停态、可中断等待态、不可中断等待态和僵死态（如图 9-1 所示）。进程的状态之间，通过信号进行调度、创建、终止和唤醒等操作。

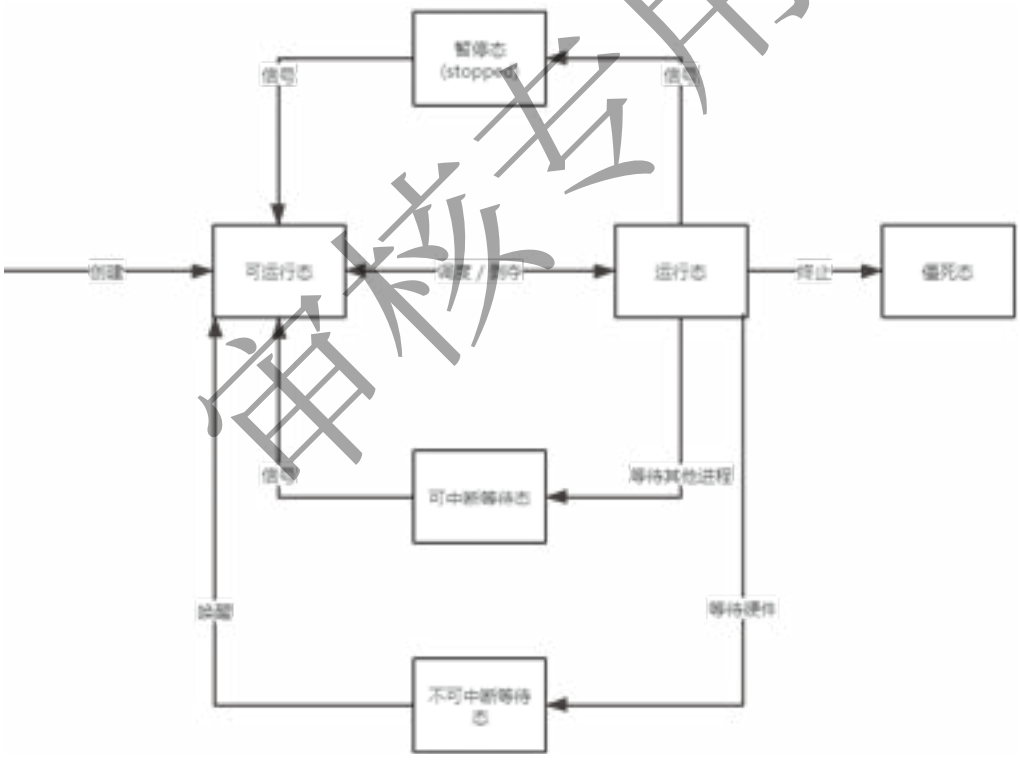

图 9-1　GNU/Linux 进程状态及其转换

进程实体由代码段、数据段和进程控制块三部分组成。进程控制块包含：进程标志符（PID）、处理器状态、进程调度信息和进程控制信息。

而线程是程序运行过程中粒度最小的单位。一个进程可以对应单个线程，也可以对应多个线程。进程与线程最主要的区别特征是有无共享寄存器，其关系如图 9-2 所示。

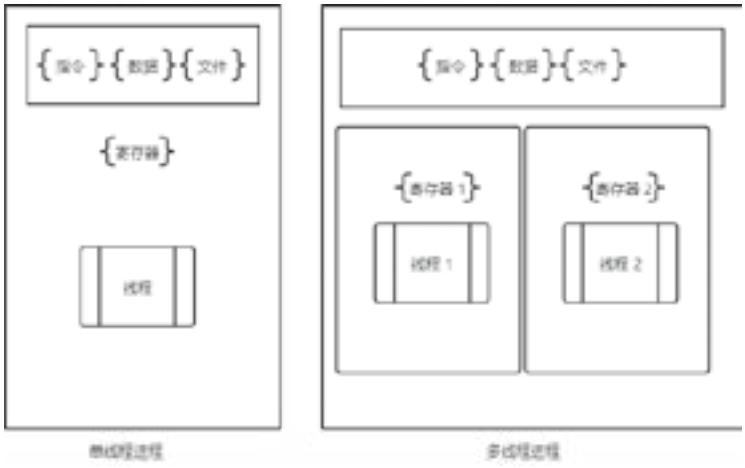

图 9-2　进程与线程

9.1.2　进程的类型

运行中的进程又可以分为前台进程和后台进程。前台进程占用当前的终端，使得用户不能与其他的进程进行直接的命令交互；后台进程在运行时不占用终端的交互设备，使得用户可以在感受上接近于同时执行其他的命令。

在了解了前后、台进程的概念后，接下来认识按启动形态分类的三类进程：

（1）交互进程是在 Shell 上启动的进程。如在终端上直接执行命令。交互进程既可以在前台运行，也可以在后台运行。

（2）批处理进程和终端没有联系，是一个进程序列。如通过 Shell script 启动的进程。

（3）监控进程，也称守护进程，是系统启动时运行的进程，通常常驻后台。如 httpd 是 Apache 服务器的监控进程。

9.1.3　常见的 GNU/Linux 系统进程

内核所提供常见进程如表 9-1 所示，其中进程名称用 [ ] 包裹起来。

表 9-1　内核所提供常见进程

[kthreadd]	内核线程管理
[migration/0]	用于进程在不同的 CPU 间迁移
[ksoftirqd/0]	内核调度 / 管理第 1 个 CPU 软中断的守护进程
[migration/1]	管理多核心
[ksoftirqd/1]	内核调度 / 管理第 1 个 CPU 软中断的守护进程
[events/0]	处理内核事件守护进程
[events/1]	处理内核事件守护进程
[cpuset]	在每个处理器上单独运行进程，通过文件系统实现
[khelper]	内核帮助进程
[async/mgr]	异步加密管理进程
[pm]	包管理
[sync_supers]	特权同步，将缓冲区文件强制写入硬盘

续表

[kthreadd]	内核线程管理
[bdi-default]	JTAG 调试器默认进程
[kintegrityd/0]	内核完整性检查
[kintegrityd/1]	内核完整性检查
[kblockd/0]	管理磁盘块读写
[kblockd/1]	管理磁盘块读写
[kacpid]	高级配置和电源管理接口
[kacpi_notify]	acpi 进程的通知进程
[kacpi_hotplug]	acpi 热插拔管理
[ata/0]	ATA 硬盘接口管理
[ata/1]	ATA 硬盘接口管理
[ata_aux]	ATA 硬盘接口管理
[khubd]	内核的 usb hub
[kseriod]	内核线程
[kswapd0]	内存回收，确保系统空闲物理内存的数量在一个合适的范围
[ksmd]	作为内核中的守护进程存在，它定期执行页面扫描，识别副本页面并合并副本，释放这些页面以供它用

另，systemd 是 GNU/Linux 下的系统及设置管理程序。其目的是提供框架以表示系统服务间的依赖关系，并依此实现系统初始化时服务的并行启动，同时达到降低 Shell 的开销的目的，是系统启动时第一个启动的进程，其 PID 通常为 1。

9.2　进程管理及其常用命令

进程是度量系统计算资源的基本单位，对进程进行管理，实际上就是在管理计算机的计算资源分配。

9.2.1　查看和监视系统进程

查看 GNU/Linux 系统上的进程，常用 ps 命令。其主要可选参数如表 9-2 所示。

表 9-2　ps 可选参数

参数	含义
-a	显示所有终端下执行的进程
a	显示现行终端下的所有进程，包括其他用户的进程
-A	显示所有进程
-c	显示 CLS 和 PRI 栏位
c	列出进程时，显示每个进程真正的命令名称，而不包含路径、选项或常驻服务的标志
-C	指定执行命令的名称，并列出该命令的进程的状态
-d	显示所有进程
-e	此选项的效果和指定 A 选项相同
e	列出进程时，显示每个进程所使用的环境变量

续表

参数	含义
-f	显示 UID、PPIP、C 与 STIME 栏位
f	用 ASCII 字符显示树状结构，表达进程间的相互关系
-g	此选项的效果和指定 -G 选项相同
g	显示现行终端下的所有进程
-G	列出属于该用户组的进程的状态，也可使用用户组名称来指定
h	不显示标题列
-H	显示树状结构，表示进程间的相互关系
-j 或 j	采用工作控制的格式显示进程状态
-l 或 l	采用详细的格式来显示进程状态
L	列出进程的相关信息
-m 或 m	显示所有的线程
n	以数字来表示 USER 和 WCHAN 栏位
-N	显示所有的进程，除了执行 ps 命令终端下的进程之外
-p	指定进程识别码，并列出该进程的状态
p	此选项的效果和指定 -p 选项相同，只在列表格式方面稍有差异
r	只列出当前终端正在执行中的进程
-s	指定作业的进程识别码，并列出隶属该作业的进程的状态
s	采用进程信号的格式显示进程状态
S	列出进程时，包括已中断的子进程信息
-t	指定终端编号，并列出属于该终端的进程的状态
t	效果和指定 -t 选项相同，只在列表格式方面稍有差异
-T	显示当前终端下的所有进程
-u	效果和指定 -U 选项相同
u	以用户为主的格式来显示进程状态
-U	列出属于该用户的进程的状态，也可使用用户名称来指定
U	列出属于该用户的进程的状态
v	采用虚拟内存的格式显示进程状态
-V 或 V	显示 ps 版本信息
-w 或 w	采用宽格式来显示进程状态
x	显示所有进程，不以终端来区分
X	采用旧式的 Linux i386 登录格式显示进程状态
-y	配合选项 -l 使用时，不显示 F (flag) 栏位，并以 RSS 栏位取代 ADDR 栏位
--cols	设置每列的最多字符数
--columns	此选项的效果和指定 --cols 选项相同
--cumulative	此选项的效果和指定 S 选项相同
--deselect	此选项的效果和指定 -N 选项相同
--forest	此选项的效果和指定 f 选项相同
--headers	重复显示标题列
--help	在线帮助
--info	显示排错信息
--lines	设置显示画面的列数
--no-headers	此选项的效果和指定 h 选项相同，只在列表格式方面稍有差异

续表

参数	含义
-group	此选项的效果和指定 -G 选项相同
-Group	此选项的效果和指定 -G 选项相同
-pid	此选项的效果和指定 -p 选项相同
-rows	此选项的效果和指定 -lines 选项相同
-sid	此选项的效果和指定 -s 选项相同
-tty	此选项的效果和指定 -t 选项相同
-user	此选项的效果和指定 -U 选项相同
-User	此选项的效果和指定 -U 选项相同
-version	此选项的效果和指定 -V 选项相同
-widty	此选项的效果和指定 -cols 选项相同

top 命令用于实时动态显示当前运行的进程的情况，包括处理器占用等信息，在功用上与 Windows 中的任务管理器相近。其运作时的画面如下所示：

```
top - 12:40:27 up 8 days, 13:25,  0 users,  load average: 0.52, 0.58, 0.59
Tasks:   5 total,   1 running,   3 sleeping,   1 stopped,   0 zombie
%Cpu(s):  2.3 us,  8.4 sy,  0.0 ni, 87.3 id,  0.0 wa,  1.9 hi,  0.0 si,  0.0 st
KiB Mem : 66941632 total, 40433788 free, 26278492 used,   229352 buff/cache
KiB Swap:  4041728 total,  3881184 free,   160544 used. 40529408 avail Mem
PID USER      PR  NI    VIRT    RES    SHR S  %CPU %MEM     TIME+ COMMAND
1 root     20   0    8936    244    196 S   0.0  0.0   0:00.10 init
7 root     20   0    8944    160    104 S   0.0  0.0   0:00.00 init
8 user      20   0   17052   3336   3224 S   0.0  0.0   0:00.90 bash
42 user      20   0   13960    484    364 T   0.0  0.0   0:00.03 tee
489 user      20   0   17652   2072   1504 R   0.0  0.0   0:00.01 top
```

top 命令的模式是 top [-] [d delay] [q] [c] [S] [s] [i] [n] [b]，其中括号内的是可选参数，意味着最简短只需要输入命令本身即可运行。部分常用选项说明如表 9-3 所示。

表 9-3　top 命令选项

d	改变显示的更新速度
Q	没有任何延迟的显示速度，如果使用者是有超级用户的权限，则 top 将会以最高的优先序执行
C	切换显示模式，共有两种模式：一是只显示执行档的名称；另一种是显示完整的路径与名称
S	累积模式，会将已完成或消失的子进程（dead child process）的 CPU time 累积起来
S	安全模式，将交谈式指令取消，避免潜在的危机
i	不显示任何闲置（idle）或无用（zombie）的进程
N	更新的次数，完成后将会退出 top
B	批次档模式，搭配“n”参数一起使用，可以用来将 top 的结果输出到档案内

例如，要显示进程信息，可用：

```
top
```

要显示完整命令，可用：

```
top -c
```

要以批处理模式显示进程信息，可用：

```
top -b
```

要以累积模式显示进程信息，可用：

```
top -S
```

若要设置只更新两次，可执行：

```
top -n 2
```

设置信息更新时间为 3 秒，可执行：

```
top -d 3
```

显示指定 PID 的进程信息，可：

```
top -p 666
```

要禁止用户使用交谈式指令，可执行：

```
top -s
```


9.2.2 进程调度命令 kill

kill 命令用于终止执行中的进程，其可将指定的信号送至进程。预设的信号为 SIGTERM，可将指定程序终止。若仍无法终止该进程，可使用 SIGKILL 信号尝试强制终止程序。进程或工作的编号可分别利用 ps 指令或 jobs 指令查看。

语法：

```
kill [-s < 信号名称或信号的编号 >][ 进程 ]
```

参数说明：

-s：指定要送出的信号。

其中，[进程] 可以是进程的 PID 或是 PGID，也可以是进程对应的工作编号。

要查看所有可用的信号及其对应的编号，可使用 kill -l 命令列出。如：

```
user@ltt-llk2:/mnt/c/Users /devhood/ackt/tutorial-gnu_Linux-app-basic/src/ch4$ kill -l
```

```
 1）SIGHUP       2）SIGINT       3）SIGQUIT      4）SIGILL       5）SIGTRAP
 6）SIGABRT      7）SIGBUS       8）SIGFPE       9）SIGKILL      10）SIGUSR1
 11）SIGSEGV     12）SIGUSR2     13）SIGPIPE     14）SIGALRM     15）SIGTERM
 16）SIGSTKFLT   17）SIGCHLD     18）SIGCONT     19）SIGSTOP     20）SIGTSTP
 21）SIGTTIN     22）SIGTTOU     23）SIGURG      24）SIGXCPU     25）SIGXFSZ
 26）SIGVTALRM   27）SIGPROF     28）SIGWINCH    29）SIGIO       30）SIGPWR
 31）SIGSYS      34）SIGRTMIN    35）SIGRTMIN+1  36）SIGRTMIN+2  37）SIGRT-
MIN+3
 38）SIGRTMIN+4  39）SIGRTMIN+5  40）SIGRTMIN+6  41）SIGRTMIN+7  42）SIGRT-
MIN+8
 43）SIGRTMIN+9  44）SIGRTMIN+10 45）SIGRTMIN+11 46）SIGRTMIN+12 47）SIGRT-
MIN+13
 48）SIGRTMIN+14 49）SIGRTMIN+15 50）SIGRTMAX-14 51）SIGRTMAX-13 52）SIGRT-
MAX-12
 53）SIGRTMAX-11 54）SIGRTMAX-10 55）SIGRTMAX-9  56）SIGRTMAX-8  57）SIGRT-
MAX-7
 58）SIGRTMAX-6  59）SIGRTMAX-5  60）SIGRTMAX-4  61）SIGRTMAX-3  62）SIGRT-
MAX-2
 63）SIGRTMAX-1  64）SIGRTMAX
```

其中比较常用的信号有：

- 1（HUP）：重新加载一个进程。
- 9（SIGKILL）：结束一个进程。
- 15（SIGKILL）：正常停止一个进程。

例：

结束 PID 是 666 的进程：

```
kill 666
```

也可以如下操作：

```
kill -SIGKILL 666
```

发送 SIGHUP 信号，重启进程，如下：

```
kill -HUP 666
```

用一般的方式正常结束进程：

```
kill -15 666
```

9.2.3 作业管理

以下是作业管理中常用的操作，其中具体“% 作业 ID”通常是以数字形态出现：

- 命令的最后加一个 & 可将作业放入后台执行。如果没有进行重定向，数据流仍然会输出到前台
- 按键 Ctrl+C 强制中断前台当前作业执行
- 按键 Ctrl+Z 将作业挂到后台
- 命令 jobs -l 查看所有作业，保护你作业 ID 和作业对应进程的 PID
- fg % 作业 ID 将后台作业提至前台
- bg % 作业 ID 将后台作业由暂停变为执行
- kill -signal % 作业 ID 向指定作业的所有进程发送指定的信号

作业管理中所言的后台与系统守护进程所言的后台有别，上述的任务管理在当前终端的会话相关，当终端会话结束时，预设情况下终端会话上所有的作业都将停止。

例：对本机的环回网卡进行 ping 操作过程中体验作业管理：

```
ping 127.0.0.1
```

先按 Ctrl + Z。再执行

```
jobs
```

可以观察到刚刚启动的 ping 进程赫然在列，并已停止（stopped）。然后输入命令：

```
fg
```

可以观察到刚刚已经移动到后台的 ping 进程出现在前台。

要终止当前的前台应用，只需要按 Ctrl + C。

本章小结

本章从应用的层面讲解了程序、进程与线程的概念及其三者之间的关系，重点讲述了常用于查看和监视进程的命令。

习题

一、简答题

1. 列出系统当前以来 Python 执行的命令。

_____

_____

2. 实时监控系统中的进程，每两秒更新一次数据，可用何命令？

_____

3. 查看当前终端所有的作业，可用何命令?

4. 强制结束一 PID 为 523083 的进程，可用何命令?

5. 进程和作业有何区别?

✔ 综合实训一

一、实训题目

作业管理。

二、实训目的

熟悉终端下作业管理的方法。

三、实训内容

执行 ping 127.0.0.1
按 Ctrl+Z
执行 ping 127.0.0.2
再按 Ctrl+Z
执行命令：

```
jobs
```

查看并选择其中一个作业，用 fg 加作业编号的方法将其带回终端的前台继续运行。
按 Ctrl+C 传递终止信号给当前的前台作业，令其结束。
用 kill –9 加上作业编号的方法结束余下的一个终端后台暂停中的作业。

✔ 综合实训二

一、实训名称

进程管理

二、实训目的

掌握 Linux 系统中运行级别的设置

掌握 Linux 下进程的管理。

三、实训内容

练习 Linux 系统 init、ps、top、crontab 等常用进程管理命令的使用方法。

四、项目背景

某企业的 Linux 服务器由于环境需要，更改为默认运行级别为 3，并且在运行过程中发现有进程占用系统大量资源，使用 top 等命令对有问题的进程进行关闭，最后要求利用 cron 指定系统计划任务。

五、实训步骤

子项目 1　更改运行级别

1. 更改系统默认运行级别为 3，重启后利用更改运行级别的方式进入图形用户界面

2. 利用引导管理程序 GRUB 更改系统管理员密码为 123456

子项目 2　管理进程

1. 运行 manage_processes_start 命令之后，系统现在运行应该非常缓慢

2. 确定内存使用量过多的进程，终止该进程

3. 确定 cpu 使用量过多的进程，调整优先级为 15

子项目 3. 调度任务的设定

1. 某系统管理员需要每天做一定的重复工作，请按照下列要求，编制一个解决方案：

每周一早上 7:50 自动清空 FTP 服务器公共目录“/var/ftp/pub”

从早上 8：00~ 下午 6：00 每小时读取 /xyz 目录下 x1 文件中最后 5 行的全部数据加入到 /backup 目录下的 bak01.txt 文件内

在下午 5：55 将 IDE 接口的 CD-ROM 卸载（假设 CD-ROM 的设备名为 hdc）

每天晚上 10:30 分自动执行任务，保存磁盘使用情况信息

第十章

网络信息安全

内容简介

- 本章节首先介绍了 Linux 防火墙技术相关的理论基础，最后重点讲述两种 Linux 防火墙服务，分别是 firewalld 动态防火墙管理服务以及 iptables 静态防火墙服务。

学习要求

- 掌握基于 Linux 系统的防火墙技术中的两种服务（firewalld 服务与 iptables 服务），熟悉防火墙基本配置方法。

10.1 防火墙技术

防火墙是网络安全的基本工具。通过在服务器和外部访客之间建立过滤机制，防火墙在网络层面上实现了安全防范。本节首先介绍防火墙的基本概念，然后介绍 Linux 防火墙的组成及工作原理，接着介绍 CentOS 中的 firewalld 守护进程和配置工具 firewa ll-cmd 的使用，以及兼容于低版本的 iptables 服务和配置工具 lokkit 的使用，最后介绍如何使用底层的 iptables 命令配置防火墙。

10.1.1 防火墙概述

1. 什么是防火墙?

防火墙是架设在不同信任级别的计算机网络之间的一种检测和控制设备。它是一个控制和监测的阻流点，不同级别网络间的所有数据都必须经过检查啊，实现边界防护，根据其配置的安全策略和规则允许、拒绝或代理数据的通过，必要时，可以同时提供 NAT/VPN 功能。

2. 防火墙的功能

（1）提供边界防护功能。因为内外网之间的所有网络数据流都必须经过防火墙，所以防火墙可以控制内外网之间网络系统的访问，提高内部网络的保密性和私有性。

（2）提供网络服务访问限制功能。只符合安全策略的数据流才能通过防火墙，保护易受攻击的服务。

（3）提供审计和监控功能。防火墙可以记录网络的使用状态，实现对异常行为的报警，集中管理内网的安全性，降低管理成本。

（4）防火墙自身应具有非常强的抗攻击免疫力。

3. 防火墙的局限性

（1）不能保护绕过防火墙的攻击。防火墙不能保护非授权的网络连接（Modem 拨号连接、Wireless 连接等）和执行 CD/DVD/USB 等介质上的恶意软件。

（2）不能保护被防火墙信任的攻击。防火墙不能保护被防火墙信任的用户 / 组织的攻击以及被防火墙信任的服务（如 SSL/SSH）的攻击。

（3）不能防止内部威胁，如心怀不满的雇员的攻击。

（4）不能防止所有病毒感染的程序或文件的传输。由于病毒类型甚多，由防火墙检测他们将严重映像数据传输速度，应交由专业的病毒检测软件处理。

4. 防火墙的类型

按照是否使用专用设备划分，防火墙可分为以下两类：一是硬件防火墙，专用的硬件或软硬件结合的实现；二是软件防火墙，基于普通 PC 或 Server 硬件上的通用操作系统加防火墙软件实现。如图 10-1 所示为 Linux 防火墙的基本应用环境，它建设在私有网和公共网之间实施防护，而我们在本章节讨论主要是软件防火墙。

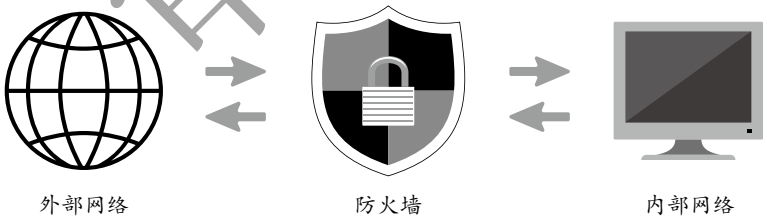

图 10-1　防火墙隔离内外网

10.1.2　包过滤防火墙与网络地址转换 NAT

本小节主要介绍包过滤防火墙及 NAT 的基本概念，以及它们的工作原理，为后续 Linux 系统的防火墙的相关安全配置策略提供理论基础知识。

1. 包过滤防火墙

包过滤防火墙查看所流经的数据包的包头（herder），由此决定整个数据包的命运。它可能会决定丢弃（DROP）这个包，可能会接收（ACCEPT）这个包（让这个包通过），也可能执行其他更复杂的动

作。传统的包过滤可以通过对数据包的 IP 头和 TCP 或 UDP 头的检查来实现，可检查的信息有：源或目标 IP 地址、协议（TCP.UDP.ICMP）、数据包的源或目标端口、ICMP 消息类型、TCP 包头中的控制标志位等。传统的网络层包过滤不检查更高层上下文，新型的状态包过滤器检查每个数据包的上下文，跟踪客户端 / 服务器的绘画，检查每一个数据包属于哪个会话，检查哪些包是首次连接的新包；哪些是已建立连接的回应包；哪些是属于某个已经建立的连接所建立的先连接包；哪些是无效包。包过滤器的工作过程的流程图如图 10-2 所示。

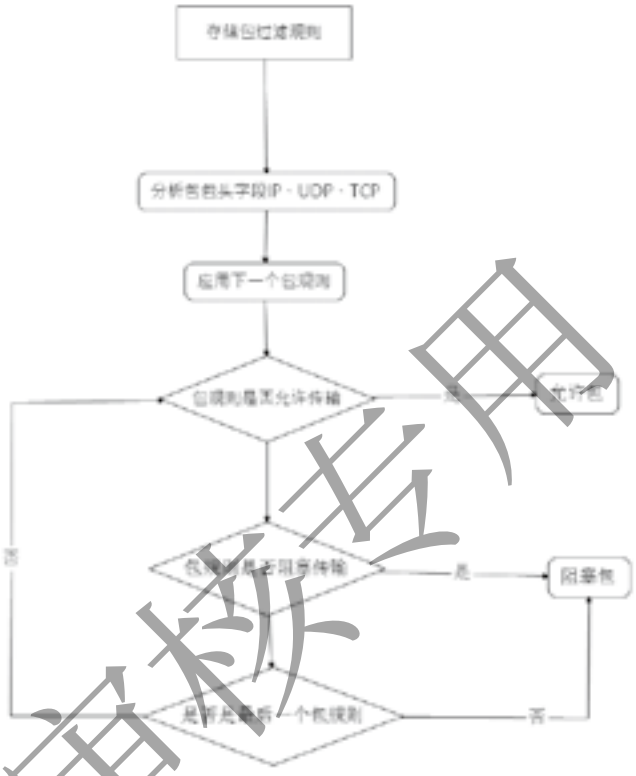

图 10-2　包过滤防火墙工作流程图

包过滤技术的优缺点如下所述：

（1）优点。一个包过滤可以保护整个网络，若过滤规则简单，则过滤路由器的效率会很高，同时安全性自然也就降低；由于包过滤工作在 IP 层和 TCP 层，所以处理包的速度比工作在应用层的代理服务器快；包过滤为用户提供了一种透明的服务，用户不需要改变客户端的任何应用程序，也不需要用户学习任何新的东西。过滤路由器有时也被称为“包过滤网关”。

（2）缺点。包过滤不支持有效的用户认证；规则表很快会变得很大而且复杂，规则很难测试。随着表的增大和复杂性的增加，规则结构出现漏洞的可能性也会增加；包过滤防火墙只能组织一种类型的 IP 欺骗，即外部主机伪装内部主机的 IP，对于外部主机伪装外部主机的 IP 欺骗却不可能被阻止，而且它不能防止 DNS 欺骗。

2. 网络地址转换 NAT

网络地址转换（Network Address Translation，NAT）主要提供了将地址域（如专用 Intranet）映

射到另一个地址域（如 Internet）的标准方法。NAT 允许一个机构专用 Intranet 中的主机透明地连接到公共域中的主机，无须内部主机拥有注册的 Internet 地址。NAT 也可以应用到防火墙技术里，把个别 IP 地址隐藏起来不被外界发现，使外界无法直接访问内部网络设备。

网络地址转换（NAT）是通过改写数据包的源 IP 地址、目的 IP 地址、源端口、目的端口号来实现的。按照改写内容不同，NAT 分为两种不同的类型：源 NAT（Source NAT，SNAT）以及目的 NAT（Deatination NAT，DNAT）。SNAT 是指修改包的源地址，即改变连接的来源地。NAT 防火墙会在包送出之前的最后一刻（出站路由之后）做好 SNAT 动作。Linux 世界中的 IP 伪装（IP Masquerading）非常出名，它是 SNAT 的一种特殊形式。目的 DNAT 是指修改包的目标地址，即改变连接的目的地。NAT 防火墙会在包进入之后（入站路由之前）立刻进行 DNAT 动作。端口转发（Port forwarding）、负载均衡和透明代理（Transparent Proxy）都属于 DNAT。

10.1.3 Linux 防火墙技术

Linux 防火墙系统由内核控件的 Netfilter 框架及内核模块和用户空间用于操作 Netfilter 的一些列管理工具两部分组成。表 10-1 列出了内核空间的基于 Netfilter 框架的内核模块以及用户空间的管理工具。在 CentOS 下，基于 Netfilter 框架实现的相关的内核模块保存在如下目录：

- /lib/modeules/$（uname -r）/kernel/net/bridge/netfilter/。
- /lib/modeules/$（uname -r）/kernel/net/ipv{4，6}/netfilter/。
- /lib/modeules/$（uname -r）/kernel/ne/netfilter/。

表 10-1 内核空间的 Netfiter 与用户空间的工具

功能	内核空间 Netfilter 框架中的模块		用户空间的管理工具
检查以太网帧	ebtables.ko	etables_broute.ko etables_filter.ko etables_nat.ko	ebtables
检查 IPv4 数据包	ebt_*.ko	iptable_filter.ko iptable_mangle.ko iptable_nat.ko iptable_raw.ko iptable_security.ko	iptables
检查 IPv6 数据包	ip_tables.ko nf_connetrack_ipv4.ko nf_nat_ipv4.ko	ip6table_filter.ko ip6table_mangle.ko ip6table_nat.ko ip6table_raw.ko ip6table_security.ko	ip6tabes
链接跟踪	ip6_tables.ko nf_conntrack_ipv6.ko nf_nat_ipv6.ko	nf_conntrack_*.ko	conntrack
网络地址转换	nf_fonntrack.ko	nf_nat_*.ko	

NetFilter 是一个由 Linux 内核提供的框架，它以自定义的处理形式（不同的内核模块）实现各种网络相关的操作。基于 Netfilter 框架的各种 Linux 内核模块提供了包过滤、网络地址转换、端口重定向等各种功能和操作。

基于 CentOS 7 发行的 Linux 系统中，配置防火墙的核心任务是配置防火墙规则。更深入地来说，使用 iptables 服务或 firewalld 服务等工具对 Netfilter 在内核中所维护的规则表、规则链、规则进行追加、插入或删除等操作，而底层真正执行这些规则的是 Netfilter 及相关模块。所以 CentOS 7 提供了两套基于 iptables 的前端配置工具以及与之协同工作的服务或守护进程使配置持久化，如图 10-3 所示。

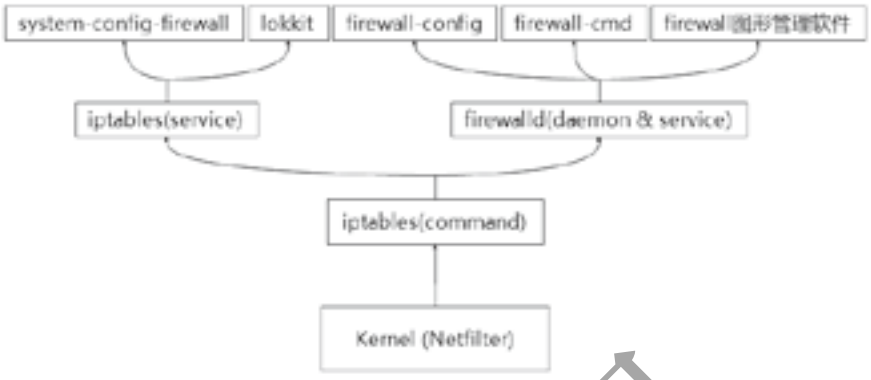

图 10-3　CentOS 7 的防火墙系统组成

• 动态防火墙配置系统

在 CentOS 7 系统中提供了动态防火墙配置系统，包括：

• firewalld 守护进程。

• 图形界面配置工具 firewall-config 和命令行界面配置工具 firewall-cmd。

为了兼容低版本，CentOS 7 仍旧支持静态防火墙配置系统，包括：

• iptables 服务。

• 图形界面配置工具 system-Config-firewall 和命令行界面配置工具 lokkit 等。

这两种服务是互斥的，不能同时使用，只能选择其中一个进行配置。使用其中一个服务时要把另外一个服务关闭。在后续章节我们将针对 firewalld 服务以及 iptables 服务如何完成相应的防火墙安全配置进行详细解说。当然在 CentOS 7 中 firewalld 服务取代了 iptables 服务，在此之前主要是通过 iptables 服务来控制的。firewalld 和 iptables 的区别，如表 10-2 所示。

表 10-2　firewalld 服务与 iptables 服务的区别

	firewalld	iptables
配置文件	/usr/lib/firewalld/、/etc/firewalld/	/etc/sysconfig/iptables
对规则的修改	不需要全部刷新策略，不丢失现行连接	需要全部刷新策略，丢失连接
防火墙类型	动态防火墙（灵活）	静态防火墙

其中，/etc/firewalld/：firewalld 防火墙用户自定义配置文件，需要时可通过从 /usr/lib/ firewalld/ 中拷贝；/usr/lib/firewalld/：firewalld 防火墙的默认配置文件，不见修改，若回复默认配置，可直接删除 /etc/firewalld/ 中的配置即可。

1. firewalld 防火墙

Firewalld（Dynamic Firewall Manager fo Linux systems）服务是 CentOS 7 系统中默认的防火墙管理工具，是 iptables 的前端控制器，用于实现持久的网络流量规则；特点是拥有运行时配置（动态防火墙功能）与永久配置（静态防火墙配置，重启系统时仍可重新加载之前设置好的防火墙规则）选项。除此之外，firewalld 服务支持动态更新技术并加入了区域（zone）的概念。

简单来说，区域就是 firewalld 将所有网络流量分为多个区域，从而简化防火墙管理。根据数据包的源 IP 地址或传入网络接口等条件，流量将转入相应区域的防火墙规则。对于流入系统的每个数据包，将首先检查其源地址。

- 若此源地址关联到特定的区域，则会执行改区域的规则。
- 若此源地址未能关联到某区域，则使用传入网络接口的区域并执行区域规则。
- 若网络接口未与某区域关联，则使用默认区域并执行区域规则。默认区域并非单独的区域，而是预定义区域之一，默认情况下默认区域是 public，但管理员可以更改。

每个区域都可以配置其要打开或关闭的一系列服务，firewalld 的每个预定义区域都设置了默认打开的服务。表 10-3 列出了 firewalld 服务的预定义区域说明。

表 10-3　firewalld 服务的预定义区域

区域	默认规则及策略
trusted（信任）	可接收所有的网络连接
home（家庭）	用于家庭网络，仅接受 ssh，msdns,ipp-client，samba-client，dhcpv6-client 服务连接
internal（内部）	用于内部网络，仅接受 ssh，msdns,ipp-client，samba-client，dhcpv6-client 服务连接
work（工作）	用于工作区域，仅接受 ssh ipp-client 或 dhcpv6-client 服务连接
public（公共）	在公共区域内使用，仅接受 ssh 或 dhcpv6-client 服务连接
external（外部）	出去的 ipv4 网络连接通过此区域，伪装和转发，仅接受 ssh 服务连接
dmz（非军事区）	仅接受 ssh 服务连接
block（限制）	拒绝所有网络连接
drop（丢弃）	任何接受的网络数据包都被丢弃，并没有任何回复

简单来讲就是为用户预先准备了几套规则集合，我们可以根据不同的场景选择合适的规则聚合，而默认的区域是 pubilc。

firwalld 服务的配置方法有以下三种：使用图形化工具 firewall-config，文本管理工具 firewall-cmd 以及直接编辑 /etc/firewalld/ 目录中的配置文件。后续章节将对前两种方法进行详细说明。

若系统还未安装 firewalld 防火墙，可以输入如下命令安装并开启。

```
[root@admin~]$yum -y install firewalld      # 安装 filrewalld
[root@admin~]$systemctl start firewalld     # 启动 firewalld
[root@admin~]$systemctl enable firewalld    # 设置 firewalld 为开机自启动
```

2.firewall-cmd 命令管理工具

在本章节中将对 firewall-cmd 命令进行详细地解析说明。

（1）启动、停止、查看 firewalld 服务相关命令。

```
# systemctl start firewalld      # 启动 firewalld
# systemctl enable firewalld    # 设置 firewalld 为开机自启动
# systemctl status firewalld    # 如果 firewalld 正在运行，通过此命令或 firewall-cmd 命令可以查看其运行状态
```

（2）获取预定义信息相关命令。

firewall-cmd 预定义信息主要包括三种：可用区域、可用服务以及可用的 ICMP 阻塞类型。

①显示预定义的区域。

```
# firewall-cmd --get-zones   # 千万注意命令之间的空格
```

②显示预定义的服务

```
[root@admin~]# firewall-cmd --get-services
```

③显示预定义的 ICMP 类型

```
[root@admin~]# firewall -cmd --get-icmptypes
```

（3）区域管理相关命令。使用 firewall-cmd 命令可以实现获取和管理区域，为指定区域绑定网络接口等功能，具体相关命令如下：

--get-default-zone：显示网络连接或接口的默认区域。

--set-default-zone=<zone>：设置网络连接或接口的默认区域。

--get-active-zones：显示已激活的所有区域。

--get-zone-of-interface=<interface>：显示指定接口绑定的区域。

--zone=<zone> --add-interface=<interface>：为指定接口绑定区域。

--zone=<zone> --change-interface=<interface>：为指定区域更改绑定的网络接口。

--zone=<zone> --remove-interface=<interface>：为指定的区域删除绑定的网络接口。

--list-all-zones：显示所有区域即其规则。

[--zone=<zone>] --list-all：显示所有指定区域的所有规则，省略 --zone=<zone> 时表示仅显示默认区域的所有规则。

结合以上的规则命令，给出一些特定配置的演示样例：

①显示当前系统中的默认区域。

```
# firewall-cmd --get-default-zone
```

②显示默认区域的所有规则。

```
# firewall-cmd --list-all
```

③显示网络接口 ens33 对应区域。

```
# firewall-cmd --get-zone-of-interface=ens33
```

④将网络接口 ens33 对应区域更改为 internal 区域。

```
# firewall -cmd --zone=internal --change-interface=ens33
```

⑤显示所有激活区域。

```
# firewall -cmd --get -active-zones
```

⑥查询 eno16777728 网卡的区域。

```
# firewall -cmd --get-zone-of-interface=eeno16777728
```

⑦设置默认规则为 DMZ。

```
# firewall -cmd --set-default-zone=dmz
```

（4）服务管理及端口管理相关命令。

为了方便管理，firewalld 预先定义了很多服务，存放在 /usr/lib/firewalld/services/ 目录中，服务通过单个的 XML 配置文件来指定。这些配置文件则按以下格式命令：service-name.xml，每个文件对应一项具体的网络服务，如 ssh 服务等。与之对应的配置文件中记录了各项服务所使用的 tcp/udp 端口。在最新版本的 firewalld 中默认已经定义了 70 多种服务供我们使用，对于每个网络区域，均可以配置允许访问的服务。当默认提供的服务不适用或者需要自定义某项服务的端口时，我们需要将 service 配置文件放置在 /usr/lib/firewalld/services/ 目录中。service 服务具有以下有点：通过服务名字来管理规则更加人性化；通过服务来组织端口分组的模式更加高效，如果一个服务使用了若干个网络端口，则服务的配置文件相关于提供了到这些端口的规则管理的批量操作快捷方式。

[--zone=<zone>] --list-services：显示指定区域内允许访问的所有服务。

[--zone=<zone>] --add-service=<service>：为指定区域设置允许访问的某项服务。

[--zone=<zone>] --remove-service=<service>：删除指定区域已设置的允许访问的某项服务。

[--zone=<zone>] --list-ports：显示指定区域内允许访问的所有端口号。

[--zone=<zone>] --add-port=<portid>[-<portid>]/<protocol>：为指定区域设置允许访问的某个 / 某段端口号（包括协议名）。

[--zone=<zone>] --remove-port=<portid>[-<portid>]/<protocol>：删除指定区域已设置的允许访问的端口号或某段端口号（包括协议名）。

[--zone=<zone>] --list-icmp-blocks：显示指定区域内拒绝访问的所有 ICMP 类型。

[--zone=<zone>] --add-icmp-block=<icmptype>：为指定区域设置拒绝访问的某项 ICMP 类型。

[--zone=<zone>] --remove-icmp-block=<icmptype>：删除指定区域已设置的拒绝访问的某项 ICMP 类型，如果省略 --zone=<zone> 时表示对默认区域操作。

结合以上的规则命令，给出一些特定配置的演示样例：

①为默认区域设置允许访问的服务。

显示默认区域内允许访问的所有服务。

```
# firewall-cmd --list-services
```

设置默认区域允许访问 http 服务。

```
# firewall-cmd --add-service=http
```

设置默认区域允许访问 https 服务。

```
# firewall-cmd --add-service=https
```

②为 internal 区域设置允许访问的服务。

设置 internal 区域允许访问 mysql 服务。

```
# firewall-cmd --zone=internal --add-service=mysql
```

设置 internal 区域不允许访问 samba-client 服务。

```
# firewall-cmd --zone=internal --remove-service=samba-client
```

显示 internal 区域内允许访问的所有服务

```
# firewall-cmd --zone=internal --list-services
```

③在 internal 区域打开 443/TCP 端口。

```
# firewall-cmd --zone=internal --add-port=443/tcp
```

④在 internal 区域禁止 443/TCP 端口访问。

```
# firewall-cmd --zone=internal --remove-port=443/tcp
```

⑥在 public 区域中查询 ssh 与 http 服务是否被允许。

```
# firewall-cmd --zone=public --query-service=ssh
# firewall-cmd --zone=public --query-service=http
```

（5）两种配置模式命令。

前面提到过 firewalld 防火墙有两种配置模式：运行时模式（Runtime mode）和永久模式（Premanent mode）。此处需要注意，firewall-cmd 命令工具与以上两种配置模式相关的选项有三个。

--reload：重新加载防火墙规则并保持状态信息，即将永久配置应用为运行时配置，命令如下：

```
# firewall-cmd --reload
```

--permanent：带有此选项的命令用于设置永久性规则，这些规则只有在重新启动 firewalld 或重新加载防火墙规则时才会生效；若不带有此选项，表示用于设置运行时规则。

--runtime-to-permanent：将当前的运行时配置写入规则配置文件中，使之成为永久性。

若用户想同时配置运行时规则和持久性规则，有以下 3 种方法。结合以上的规则命令，给出一些特

定配置的演示样例：

①允许 https 服务流量通过 public 区域，要求立即生效且永久有效。

方法一：

```
# firewall-cmd --zone=public --add-service=https
# firewall-cmd --permanent --zone=public --add-service=https
```

方法二：

```
# firewall-cmd --zone=public --add-service=https
# firewall-cmd --runtime-to-permanent
```

方法三：

```
# firewall-cmd --zone=public --add-service=https
# firewall-cmd --reload
```

②允许 8080 与 8081 端口流量通过 public 区域，立即生效且永久生效。

```
# firewall-cmd --permanent --zone=public --add-port=8080-8081/tcp
# firewall-cmd --reload
```

③查看②命令要求加入的端口操作是否成功。

```
# firewall-cmd --zone=public --list-ports
```

（6）端口转发相关命令。端口转发功能可以将原本到某端口的数据包转发到其他端口：

[root@admin~]# firewall-cmd --permanent --zone=< 区域 > --add-forward-port=port=< 源端口号 >：proto=< 协议 >：toport=< 目的端口号 >：toaddr=< 目标 IP 地址 >

结合以上的规则命令，给出一些特定配置的演示样例：将访问 192.168.10.10 主机 888 端口的请求转发至 22 端口，配置命令如下：

```
# firewall-cmd --permanent --zone=public --add-forwadport=port=888:prot
=tcp:toport=22:toaddr=192.168.10.10
```

（7）firewalld 服务的富规则用于对服务、端口、协议进行更详细的配置，规则的优先级最高。演示“拒绝 192.168.10.0/24 网段的用户访问 ssh 服务”。

```
# firewall-cmd --permanent --zone=public --add-rich-rule=” rule family =” ipv4”
source address=”192.168.10.0/24” service name=”ssh” reject”
```

（8）恐慌模式相关命令。当服务器遭受严重的网络攻击时，可以启用 Panic 模式，进入此模式后会丢弃所有的入站和出站流量并丢弃已建立的连接包。

```
# firewall-cmd --panic-on
```

使用如下命令可以关闭 Panic 模式：

```
# firewall-cmd --panic-off
```

使用如下命令可以显示当前是否处于 Panic 模式：

```
# firewall-cmd --query-panic
```

10.2　iptables 服务配置防火墙

Linux 防火墙是一种典型的包过滤防火墙。通过检测到达的数据包头中的信息，确定哪些数据包可以通过，哪些应该被丢弃。防火墙行为的依据主要是数据包的目的地址、端口号和协议类型，所有这些都应该由管理员指定。而防火墙将有一系列规则组成一些“链（chains）”应用到网络数据包上。iptables 又进一步把一些功能相似的“链”组合成一个个“表（tables）”。

10.2.1　iptables 基础知识

其中设置数据过滤或处理数据包的策略叫作规则；将多个规则合成一个链；规则链的集合称之为规则表。

1. 规则链

规则链是防火墙规则（策略）的集合，详细的规则链说明如表 10-4 所示。

表 10-4　规则链的说明

链	作用
INPUT	进来的数据包应用此规则链中的策略
OUTPUT	外出数据包应用此规则链中的策略
FORWARD	转发数据包应用此规则中的策略
PREROUTING	对数据包作路由选择前应用此链中的策略（所有数据包进来的时候都由这个链处理）
POSTROUNTING	对数据包作路由选择后应用此链中的策略（所有的数据包出来的时候由这个链处理）

2. 规则表

规则表是规则链的集合。

（1）默认的 4 个规则表。

raw 表：确定是否对该数据包进行状态跟踪。（不经过内核）

mangle 表：修改 IP 数据包头（如 TTL 值），同时也用于为数据包设置标记。

nat 表：处理网络地址转换，以及修改数据包中的源、地址 IP 地址或端口等。

filter 表：过滤数据，确定是否放行该数据包。（经过内核）

通常定义 filter 表就可以满足大部分的安全需求，因为这个表包含了包过滤的左右内容。

（2）4 个规则表中默认的规则链如图 10-4 所示。

图 10-4　4 个规则表中对应的链

（3）规则表间的优先顺序，依次为：raw → mangle → nat → filter。

（4）规则链间的匹配顺序如下所述：

入站数据：PREROUTING → INPUT。

出站数据：OUTPUT → POSTROUNTING。

转发数据：PREROUTING → FORWARD → POSTROUNTING。

（5）规则链内的匹配顺序：

没指定规则表则默认指 filter 表。

不指定规则链则指表内所有的规则链。

按顺序依次进行检查，找到相匹配的规则即停止（LOG 规则例外）；若没有匹配规则按该链的默认策略处理。

3. 防火墙处理数据包流程

基于以上规则链及规则表的理论，数据包通过防火墙时的处理流程如下所述，并通过图 10-5 进行更详细地说明。

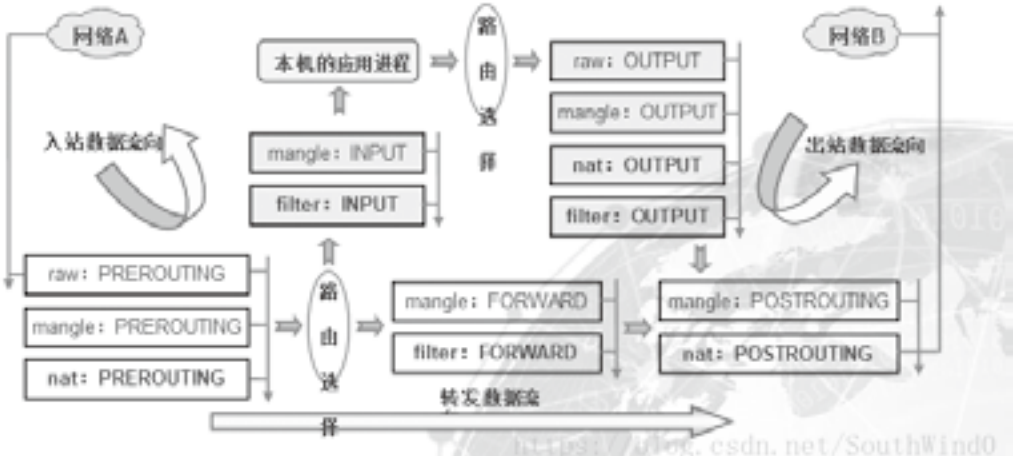

图 10-5　数据包在防火墙中的处理流程图

（1）入站数据流。来自外界的数据包到达防火墙后，首先被 PREROUNTING 规则链处理（是否修改数据包地址等），之后会进行路由选择（判断数据包应该发往何处），如果数据包的目的地址是防火墙本机（如 Internet 用户访问防火墙主机中 Web 服务的数据包），那么内核将其传递给 INPUT 链进行处理（决定是否通过等），通过以后再交给系统上层的应用程序（如 httpd 服务器）进行响应。

（2）转发数据流。来自外界的数据包到达防火墙后，首先被 PREROUNTING 规则链处理，之后会进行路由选择，如果数据包的目的地址时其他外部地址（如局域网用户通过网关访问 QQ 站点的数据包），则内核将其传递给 FORWARD 链进行处理（是否转发或拦截），然后再交给 POSTROUNTING 规则链（是否修改数据包的地址等）进行处理。

（3）出站数据流。防火墙本机向外部地址发送的数据包，首先被 OUTPUT 规则链处理，之后进行路由选择，然后传递给 POSTROUNTING 规则链（是否修改数据包的目的地址等）进行处理。

iptables 默认使用的表是“filter（过滤器）”，其中默认包含 3 个链，分别是 FORWARD、INPUT 和 OUTPUT。FORWARD 链中定义的规则作用于那些需要转发到另外一个网络接口的数据包。INPUT 链中定义的规则作用于发送到本机的数据包。相应的，OUTPUT 链中定义的规则作用于从本机发送出去的数据包。

10.2.2　iptables 命令

1. 使用 iptables 命令构建防火墙

在使用 iptables 设置防火墙时，一般遵循以下 3 个步骤进行：

清除所有规则：为了避免新建的防火墙与系统中已经运行的防火墙相互干扰，一般应该先清除所有规则。

设置防火墙策略：设置当数据包没有匹配到链中的规则时，应该如何对待（是拒绝还是放行）。

设置防火墙规则：设置数据包的匹配规则以及匹配后的处理动作（指定目标）。

从以上步骤可以看出，策略的应用范围比规则的应用范围要大。

2. 安装 iptabled 服务

为了熟悉 CentOS 5、CentOS 6 的用户仍然可以使用原来的 iptables 服务及其配置工具，为了使用 iptables 遗留服务，需执行如下命令行安装所需的软件包。

```
# yum -y install iptables-services
# yum -y install system-config-firewall-{base,tui}
```

由于 iptables 服务和 firewalld 服务是互斥的，所以需执行如下命令屏蔽 firewalld 服务并开启 iptables 服务。

```
# systemctl stop firewalld
# systemctl mask firewalld
# systemctl start iptables
# systemctl enable iptables
```

3. iptables 命令语法结构

iptables 命令用于管理防火墙的规则策略，格式为：“iptables [-t 表名] 选项 [链名] [条件] [-j 控制类型]”。

如果不指定 -t 选项，就并表明默认 filter 表。

4. 描述规则的基本参数

表 10-5 中的这些规则参数用于描述数据包的协议、源地址、目的地址、允许经过的网络接口，以及如何处理这些数据包。

表 10-5　基本参数说明

参数	作用
-P	设置默认策略：iptables -P INPUT (DROP|ACCEPT)
-F 或 (-flush)	清空规则链
-L 或 (-list)	查看规则链
-A	在规则链的末尾加入新规则
-I num	在规则链的头部加入新规则
-D num	删除某一条规则
-s 或 (-src / -source)	匹配来源地址 IP/MASK，加“!”表示除了这个 IP 外 参数可以使用 IP 地址、网络地址、主机名 例如：-s 192.168.1.101　匹配 IP 地址 -s 192.168.1.10/24 匹配网络地址
-d 或 (-dst / -destination)	匹配目标地址，参数和 -s 相同
-i interface (-in-interface)	匹配从这块网卡流入的数据包。“! -i eth0”表示匹配出除了 eth0 以外的接口流入的数据包 例如：-i eth0：匹配经由 eth0 进入的数据包 如果不指定 -i 参数，那么将处理所有接口的数据包
-o interface (-out -interface)	匹配从这块网卡流出的数据包 如果不指定 -o 选项，那么将匹配系统所有接口流出的数据包
-p 或 (-protocol)	匹配协议，如 tcp,udp,icmp 等，可以用 all 来指定所有协议 如果不指定 -p 参数，则默认是 all 值。但并不推荐这么操作，建议总是明确指定协议名称 可以使用协议名，如 tcp；或者是协议值，比如 6 代表 tcp 来指定协议。（映射关系请查看 /etc/protocols）
--dport num	匹配目标端口号
--sport num	匹配来源端口号
-j	执行目标（jump to target） 匹配与规则（Rule）匹配时如果处理数据包 可能的值是： ACCEPT：允许防火墙接收数据包 DROP：直接丢弃，不给出任何回应 QUEUE：防火墙将数据包移交到用户空间 RETURN：防火墙停止执行当前链中的后续 Rules，并返回到调用链

5. iptables 服务配置防火墙命令演示。

（1）简单命令规则。查看已有的规则：

```
# iptables -L
```

将所有 INPUT 链的默认策略设置为拒绝：

```
# iptables -P INPUT DROP
```

当 INPUT 链默认规则设置为拒绝时，那么我们就需要写入允许的规则策略。这个动作的目的是当接收到数据包时，按顺序匹配所有的允许规则策略，当全部规则不匹配时，拒绝这个数据包。

允许所有的 ping 操作：

```
#iptables -I INPUT -p icmp -j ACCEPT
```

在 INPUT 链的末尾加入一条规则，允许所有未被其他规则匹配上的数据包：

```
# iptables [-t filter] -A input -j ACCEPT   #[] 表示默认，可写可不写
```

（2）复杂命令规则。仅允许来自于 193.168.10.0/24 域的用户连接本机的 ssh 服务。iptables 防火墙会按照顺序匹配规则，请一定要保证“允许”规则时在“拒绝”规则之上。

```
# iptables -I INPUT -s 192.168.10.0/24 -p tcp --dport 22 -j ACCEPT
# iptables -A INPUT -p tcp --dport 22 -j REJECT
```

不允许任何用户访问本机的 12345 端口。

```
# iptables -I INPUT -p tcp --dport 12345 -j REJECT
# iptables -I INPUT -P -udp --dport 12345 -j REJECT
```

拒绝其他用户从“eno 1677736”网卡访问本机的 http 服务的数据包。

```
# iptables -I INPUT -i eno1677736 -p tcp --dport 80 -j REJECT
```

禁止用户访问“www.my133.org”。

```
# iptables -I FORWARD -d www.my133.org -j DROP
```

禁止 IP 地址是 192.168.10.10 的用户上网

```
# iptables -I FORWARD -s 192.168.10.10 -j DROP
```

注意：iptables 命令执行后的规则策略仅当前生效，若想重启后依然保存规则需执行“service iptables save”。

6. iptables 规则

牢记以下三点式理解 iptebles 规则的关键：

- Rules 包括一个条件和一个目标（target）。
- 如果满足条件，就执行目标中（target）的规则或者特定值。

• 如果不满足条件，就判断下一条 Rules。

本章小结

本章主要介绍防火墙技术是什么，防火墙的功能及局限性；并重点分析包过滤防火墙及 NAT（地址转换）的工作原理。

重点讲述了 Linux 防火墙的基本配置原理及相关的配置服务：firewalld 服务以及 iptables 服务。特别针对两种防火墙管理服务相关的配置命令进行了详细的解释说明。

习题

一、简答题。

1. 什么是 NAT 技术，NAT 的分类和用途分别是什么？

2. CentOS 7 提供的防火墙配置系统有哪两种？两者之间的区别是什么？

3. iptables 命令中有哪些规则链、规则表，以及它们各自的作用？

二、程序题

1. 使用 iptables 命令设置 INPUT、OUTPUT、FORWARD 链设置为允许接收数据包。

2. 使用 iptables 命令定制源地址访问策略：接收来自 192.168.0.3 的 IP 访问；拒绝来自 192.168.0.0/24 网段的访问。

3. 使用 iptables 命令定制防火墙的 NAT 访问策略。

4. 清除所有 NAT 策略。

5. 充值 ip_forward 为 1。

6. 通过 SNAT 设定来源于 192.168.6.0 网段通过 teth1 转发出去。

7. 用 iptables 观察转发的数据包。

✔ 综合实训

一、实训题目

iptables 静态防火墙配置

二、实验目的

1. 理解 iptable 的工作原理。
2. 掌握 iptables 防火墙的安装与配置。
3. 掌握 iptables 防火墙的基本操作方法，能够熟练使用防火墙。

三、实验内容

1. 完成 iptables 防火墙的安装与配置。
2. 完成 iptables 防火墙规则的管理，满足实验的场景要求。

四、实验步骤

（1）iptables 的安装与管理

1. 防火墙检测

①关闭 firewalld 防火墙，并取消开机自动启动，其操作命令如下：

systemctl stop firewalld

systemctl disable firewalld

②检查 iptables 是否安装，一半情况下，iptables 已经包含在 Linux 系统中，可以通过命令来检测系统是否已经安装 iptables，具体命令如下：

iptables －version

③检查是否安装 iptables-services，查看 iptables 服务是否安装，其命令如下所示：

systemctl status iptables

（2）安装 iptables 软件

①安装 iptables 以及 iptables services 服务软件，其操作命令如下所示：

yum install -y iptables

yum install -y iptables-service

② Iptables 服务配置，进行 iptables 服务管理，其操作命令如下所示：

systemctl start iptables　　## 开启 iptables 服务

systemctl enable iptables　## 设置开机自启动 iptables 服务

systemctl stop iptables　　## 关闭 iptables 服务

systemctl restart iptables　## 重启 iptables 服务

systemctl disable iptables　## 取消开机自启动 iptables 服务

（3）Iptables 的基本配置

①规则查看：使用命令进行防火墙规则查看，并将防火墙规则信息记录下来。

iptables -n -L

②开启需要的端口，如配置 TCP 协议的 22 端口允许进出系统，其配置命令如下：

iptables -A INPUT -p tcp --dport 22 -j ACCEPT

iptables -A OUTPUT -p tcp --sport 22 -j ACCEPT

③关闭不安全的端口，如配置不允许通过 TCP 协议的 445 端口进出系统，其配置命令如下所示：

iptables -A INPUT -p tcp -dport 445 -j DROP

iptables -A OUTPUT -p tcp --sport 445 -j DROP

④配置服务端口，如不配置允许通过 HTTP 访问系统的 80 端口，其配置命令如下：

iptables -A OUTPUT -p tcp --dport http -j DROP

（4）IP 地址配置

①拒绝某单一 IP 地址，如拒绝某一单独 IP 地址访问系统，且系统拒绝访问该 IP 地址，其配置命令如下：

iptables -A INPUT -s xxx.xxx.xxx.xxx -j DROP

iptables -A OUTPUT -d xxx.xxx.xxx.xxx -j DROP

②拒绝某 IP 地址段，如拒绝某 IP 地址段中任一地址访问系统，且系统拒绝访问该 IP 地址段中任一 IP 地址，其配置命令如下：

iptables -A INPUT -s xxx.xxx.xxx.xxx/xx -j DROP

iptables -A OUTPUT -d xxx.xxx.xxx.xxx/xx -j DROP

（5）IP 地址与端口结合

①拒绝某 IP 地址访问某端口，如拒绝某一单独 IP 地址访问系统的 22 端口（TCP 协议），其配置命令如下：

iptables -A INPUT -s xxx.xxx.xxx.xxx -p tcp --dport 22 -j DROP

②允许某段 IP 地址访问系统的服务端口，如允许某段 IP 地址访问系统的 HTTP 服务端口，其配置命令如下：

iptables –A INPUT –s xxx.xxx.xxx.xxx –p tcp ––dport http –j ACCEPT

③网络协议配置：配置拒绝 ICMP 协议通过，如配置拒绝网络中通过 PING 方式发现系统 IP 地址，其配置命令如下：

iptables –A INPUT –p icmp –j DROP

④网络接口配置：iptables 防火墙可单独为某个网卡接口设定不同的策略规则，如不允许任何主机通过防火墙本机的 ethn0 网卡访问系统的 80 端口，其配置命令如下：

iptables –A INPUT –i eth0 –p tcp ––dport 80 –j DROP

（6）MAC 地址配置

①拒绝某 MAC 地址主机的所有通信请求访问，其配置命令如下：

iptables –A INPUT –m mac –mac–source xx:xx:xx:xx:xx:xx –j DROP

②拒绝网络中某一固定 IP 地址且固定 MAC 地址的主机访问系统任意端口，其配置命令如下：

iptables –A INPUT –s xxx.xxx.xxx.xxx/x –m mac –mac–source xx:xx:x x :xx:xx:xx –j DROP

③允许网络中某一固定 IP 地址且固定 MAC 地址的主机访问系统的 22 号端口，其配置命令如下：

iptables –A INPUT –p tcp ––dport 22 –s xxx.xxx.xxx.xxx/x –m mac –mac–source xx:xx:x x :xx:xx:xx –j ACCEPT

（7）规则的测试

①软件获取

获取端口扫描工具 Zenmap 软件，可通过 Zenmap 官方网站（https://nmap.org/zenmap）下载获得。

②软件安装

点击下载的 EXE 执行安装文件，可根据安装过程提示进行默认选择安装。

③软件使用

打开工具，在工具界面中的“配置”下拉框中选择“Regular scan”(使用规则扫描)，在“命令”输入框中输入“nmap –p 1–1024 –T4 –A –v 172.16. 124.127”命令规则，点击【扫描】按钮，工具将自动扫描 IP 地址为“172.16.124.127”的主机，其 1–1024 端口的状态情况。

④信息查看

在“Nmap 输出”选项卡中查看扫描的过程，可以查看主机端口的状态信息。通过该工具可以测试防火墙规则配置是否正确且生效。

⑻ iptables 的应用

请根据下面的要求，写出防火墙的配置，将配置命令写入实训报告中。

允许来自 IP A 地址的报文，通过 UDP 方式，访问系统的 4486 端口。

丢弃来至 IP B 地址的使用 TCP 方式访问系统 20 和 21 端口的报文。

---

③允许 IP 地址属于 xxx.xxx.xxx.xxx/x 段的主机、由指定 eth0 网口，通过 ssh 远程连接本机。

允许 IP 地址为 C 的主机通过 442 端口进行 SSH 远程连接本机。

⑤将来自 IP D 地址的主机使用 TCP 协议，访问 21 端口的数据包信息记录到 messages 日志中。

⑥当超过 100 个用户同时访问系统的 80 端口时，限制每分钟最大连接数为 25 个，防止系统遭受 DOS 攻击。

五、实验总结