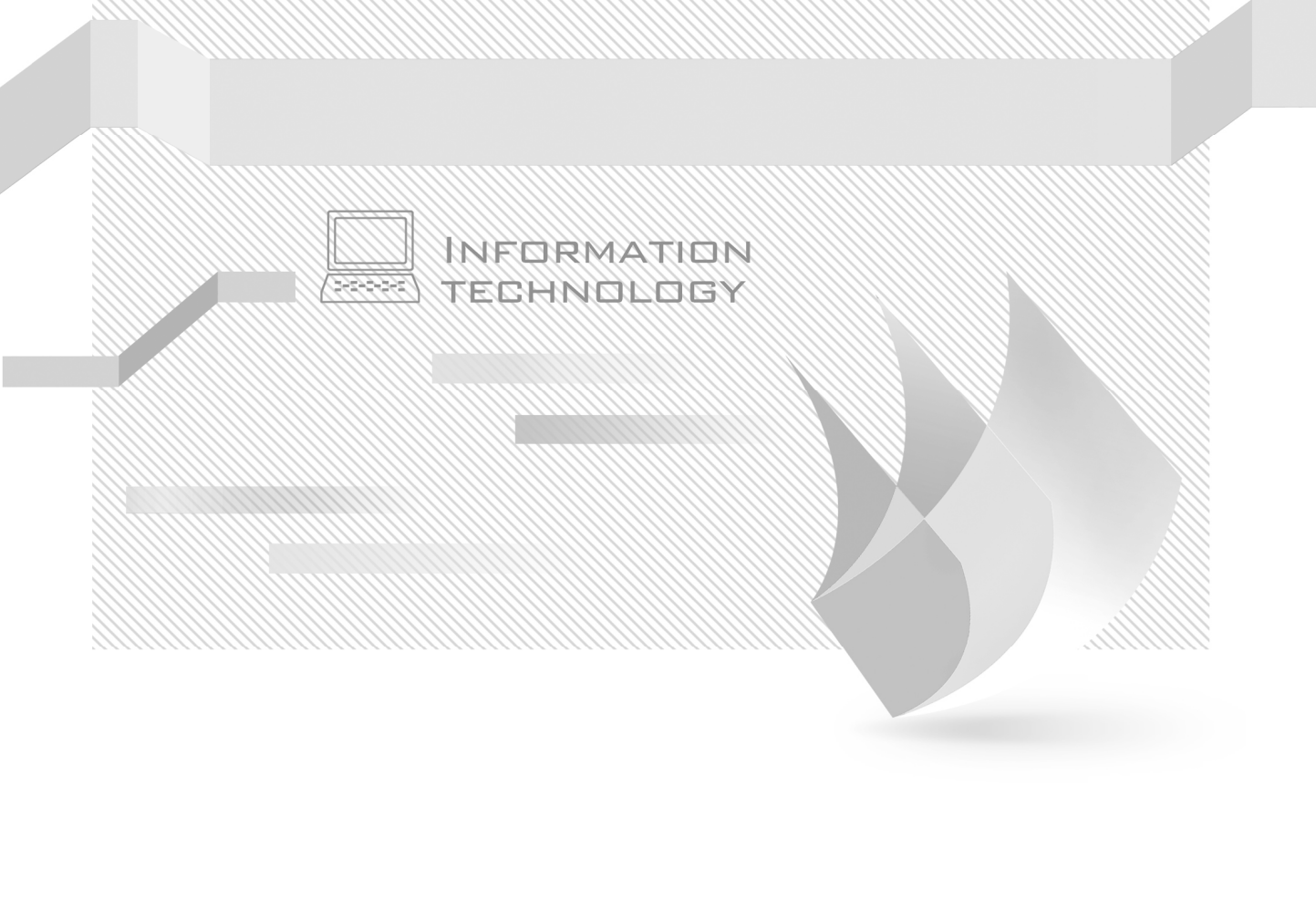
第1部分　基　础　篇



第 1 章　Linux 系统的基本使用方法

早期由于处理器硬件性能和资源的限制，嵌入式系统通过专用实时操作系统和精心设计的应用软件保证其性能。随着硬件性能的不断提高，很多民用产品的实时性已经不是问题，而更多关注的是产品的功能和扩展性。由于桌面系统有丰富的软件支持和便捷的开发手段，越来越多的应用开始将桌面系统移植到嵌入式产品中。

本章介绍 Linux 系统的基本使用方法。重点介绍与嵌入式开发关系比较密切的功能。

嵌入式系统发展的历史几乎和微处理器历史一样长。近十几年来，随着处理器性能的不断提高和计算机应用领域的不断扩展，嵌入式成为一个持续热门的词汇。所谓嵌入式系统，是一种由机械和电子系统构成的专用计算机系统，它的研究对象包括嵌入式硬件和软件。目前，98%的微处理器产品是嵌入式系统的组成部分。软件研究对象包括嵌入式操作系统和应用软件。

作为通用计算机操作系统的 Linux 系统，在嵌入式领域中表现同样出色。以手机市场为例，以 Linux 为内核的 Android 系统曾经与 Windows 和 iOS 成三足鼎立，如今在移动终端上则主要由 Android 和 iOS 统治。（据 https://www.netmarketshare.com 统计，2017 年 7 月，使用 Android 系统的手机/平板市场份额占 60%以上。）在大型计算机系统中，Linux 更是出类拔萃。据www.top500.org 的数据显示，在全世界性能最高的超级计算机 500 强中，Linux 系统自 1998 年 6 月开始进榜（此前一直是 UNIX 的天下）以来，占比持续上升，至 2015 年 6 月，使用 Linux 各种发行版的超级计算机达到 489 台。在最近几次的更新榜单中，使用非 Linux 系统的超级计算机仅占个位数。

在通用计算机系统中，Linux 系统和 UNIX 系统非常相似，而与 Windows 系列操作系统的特征则相差比较大。Linux 除了作为嵌入式系统的运行环境以外，也是理想的嵌入式开发环境。熟练掌握 Linux 系统是嵌入式 Linux 开发的基础。

1.1　Linux 系统的使用环境

软件是计算机系统的重要组成部分。软件包括程序、数据、文档，它们存储在计算机的存储设备上，存储设备通过文件系统组织。这种文件系统又称为“分区”，在 Windows 操作系统中，访问分区通过盘符“C:”、“D:”这样的符号。每个分区又有一些目录，Windows 称之为“文件夹”（Folders）。目录之下可以有文件和子目录，目录和目录之间用反斜线“\”分隔。

Linux 系统没有盘符的概念。操作系统内核启动的最后阶段，会挂载一个分区作为它的根文件系统，它可以是一个磁盘分区，也可以是一块内存，这块内存被组织成某种分区的格式。根文件系统中有操作系统赖以正常运行的基本程序。其他分区可以在系统启动后，根据需要，通过“mount”命令挂载到指定目录上。一经挂载，访问这个分区的方式和访问目录的形式是完全一样的。Linux 系统中用斜线“/”分隔目录层次。顶层的目录被称为“根目

录”。Linux 不使用反斜线分割目录。反斜线在命令行中的功能是转义符，这与它在许多编程语言及命令中的功能是一致的。

1.1.1　Linux 系统的目录结构

Linux 系统的目录结构遵循文件系统结构层次标准 FHS（Filesystem Hierarchy Standard）。该标准普遍被类 UNIX 系统采用，它定义了目录结构和内容。图 1.1 是 Linux 系统的主要目录。

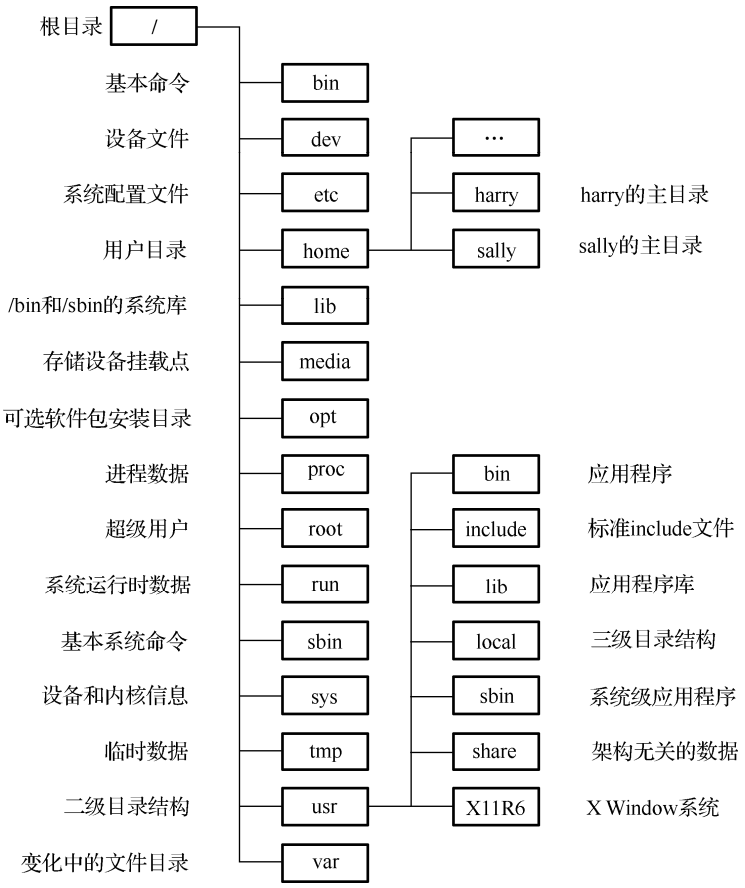


图 1.1　Linux 系统的主要目录

Linux 依据不同程序在系统中的地位，采用一级目录结构、二级目录结构（/usr）、三级目录结构（/usr/local）管理。不同程序在系统中的地位不同，对系统的影响不同，限制着用户的使用权限。供普通用户使用的命令放在/bin（binary）和/usr/bin 目录里；供超级用户使用的命令放在各级结构的 sbin（system binary）目录里，普通用户要么无权使用这些命令，要么命令的功能会受到一定限制。

1.1.2　Linux 系统的用户

UNIX 类的操作系统对用户有比较严格的管理策略。超级用户（super user，用户名为“root”，用户 ID 为 0）可以运行系统中的所有程序，支配系统的所有资源。拥有这个权限

的通常是系统的管理员。管理员可以为其他使用者创建账户，这些账户通常拥有普通用户的权限。普通用户可以正常使用计算机，但不能对计算机的设置进行改变。根据使用者在系统中的关系，管理员还可以对他们进行分组。在以组为单位的项目开发中，这种分组关系既满足了资源共享要求，同时也隔离了与其他项目之间的交叉。

有些 Linux 的发行版还有为临时使用计算机的人创建“访客用户”的做法。访客用户只有有限的计算机支配资源，其数据也不会长期保存。

供个人使用的计算机中，尽管用户可以以超级用户身份登录使用，但出于安全性方面的考虑，仍然建议用户日常操作以普通用户的身份完成。超级用户的误操作会给系统造成较大的损害，此外超级用户的使用环境会给恶意软件带来方便。很少听说 Linux 操作系统有病毒泛滥，对用户权限的限制可能是一个重要的原因。如果发现有这样的软件，非超级用户无法正常使用（通常这是一个可能改变系统设置的软件）。这时可以通过临时切换身份（使用“sudo”命令）的方式运行这个软件。如果这个问题频繁出现，要么是用户权限设置有问题，要么是软件本身的设计问题。前者应由管理员调整用户权限，后者的问题可以向软件开发者反映。

作为嵌入式系统的开发对象，使用 root 身份是正常的。因为系统运行伊始仅有这个权限，甚至还没建立用户。但作为开发环境使用的计算机，以上原则仍然有效。

1.2 命令行工作方式

在 Linux 上进行软硬件开发，命令行接口（Command-Line Interface，CLI）是主要的工作方式。虽然 Linux 也有集成开发环境，但命令行能调动的资源更多，命令行操作方式更加灵活。在网络环境下，使用图形界面可能会对服务器和客户端都有较高的要求，而命令行方式则基本上不受限制。开发嵌入式 Linux，初期的目标系统肯定不具备图形工作环境，甚至终生都不会为它移植图形界面，这时在它上面的操作只能通过命令行方式完成。命令行方式的唯一缺点是需要记住很多命令的用法。

1.2.1 终端

提供命令行工作环境的设备称为“shell”（有时也称为“终端”）。目前使用最多的是 bash（Bourne-Again SHell）。shell 可以运行系统的所有程序，包括图形界面程序。多数情况下，在终端上运行程序等效于在图形界面下用鼠标单击程序的图标。而在使用字符界面程序时，可以看到程序正常的运行过程，比如用 C 语言函数“printf( )”在标准设备上输出的信息，这在图形方式下有时是做不到的。此外，如果是可以带参数执行的命令，对于在图形操作方式下单击鼠标如何输入参数，目前也没有通用的解决方案。

在 shell 中执行命令或程序，就是直接在提示符下用键盘输入程序的名字，以回车确认。一般格式如下：

```
$ command [options] parameters
```

“command”是程序名，[①]绝大多数这样的程序在“/bin”、“/sbin”、“/usr/bin”、

① “$”表示普通用户使用 shell 的提示符，无须用键盘输入。超级用户的提示符是“#”，区别于普通用户，起到一定的警示作用。本书将以这两个符号区分命令的权限。

“/usr/sbin”目录中以可执行文件的形式存在。Linux 系统中一个重要的环境变量“PATH”包含了这些目录的列表。如果想执行的程序不在这个目录列表中，则必须明确指明这个程序所在目录。例如：

```
$ ./hello
```

表示执行当前目录下一个名为“hello”的程序。“./”表示当前目录。

命令可以带有一些选项，选项前面通常会使用“-”或“--”以便和命令的参数相区别，有的命令选项甚至不使用“-”。具体使用哪一种方式，取决于该程序作者的风格。

Linux 系统中，命令中的选项、参数，包括命令本身，都严格区分字母大小写，除非程序内部对字母大小写有专门的合并处理。[①]

1.2.2　目录操作

shell 中访问目录的命令主要有转移目录、创建和删除目录等操作。

- 列目录、文件清单“ls”。该命令不带选项时仅列出文件名。用“ls -l”可以看到列出某个目录下的文件较完整的属性：

```
$ ls -l
drwxr-xr-x 19 harry harry  4096    6月 14 12:47 docs
drwxr-xr-x 23 harry harry  4096    10月 20 2016 programs
drwx------  3 harry harry  4096    6月 21 14:28 videos
-rw-rw-r--  1 harry harry  1830    11月 30 2016 problem1_3.m
```

上面列出的文件清单中，每一行描述了一个文件的主要特性。其中，第一个字段是文件的访问权限属性，它显示了该文件的性质和读写权限。第一个字母“d”表示它是一个目录文件（Directory），普通文件则用“-”表示。后面紧跟的 9 个字母中，每三个为一组，依次表示文件拥有者、文件属组以及其他人对该文件的访问权限。访问权限的三个字母 r、w、x 分别表示读（Read）、写（Write）和运行（eXecute）许可。Linux 系统的文件可执行属性就体现在“x”标志位上，而与其名称及后缀无关。目录的“x”标识表示该目录可以访问。对应位置的“-”表示其权限的缺失。属性字段后面的数字表示该文件的链接数。接下来的两个名字分别表示文件拥有者和属组。上面的清单显示出，docs 和 programs 目录允许所有人访问，但只有 harry 本人有修改权限，而 videos 只有 harry 本人有访问权限。文件 problem1_3.m 不是一个可执行文件，允许 harry 及 harry 组员读写，其他人只能读不能修改。

- 转移目录“cd”，这是一条 shell 的内部命令。从当前的工作目录转移到另一个目录的命令格式是[②]

```
$ cd pathname
```

表示路径的方式有两种：从根目录开始，将各级目录名按顺序用斜线分隔，这种形式称为“绝对路径”；以当前目录为起点，向上或向下达到目标目录的形式称为“相对路径”。

---

① 归根到底，这种大小写敏感的特性不是由操作系统决定的，而是由是由文件系统决定的。如果使用了文件名不区分字母大小写的文件系统（如 FATFS），命令行操作会在一定程度上失去大小写的区别意义。

② 目录（directory）有时又被称为“路径”（path）。

图 1.2 中，假设用户“harry”想从当前目录“docs”转移到“videos”目录下工作，他可以采用绝对路径操作方式“cd /home/harry/videos”，也可以采用相对路径操作方式“cd ../videos”。“../”表示上一级目录。采用后一种方式时，转移目录的操作方式与当前位置有关。

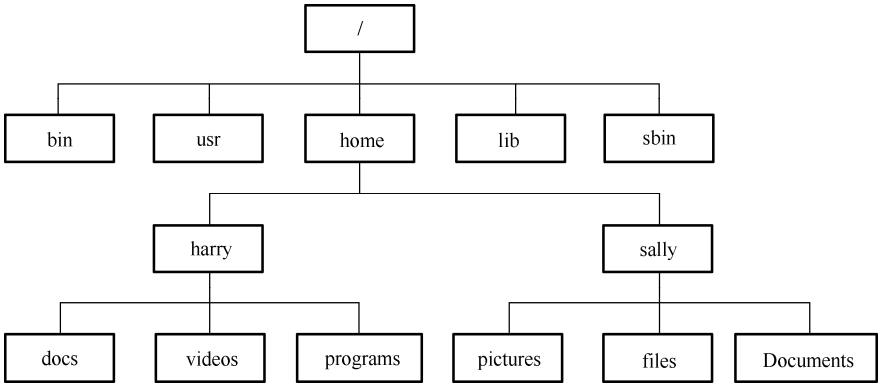


图 1.2 相对路径和绝对路径

“cd”命令不带参数时，将回到用户的主目录（环境变量 HOME 指向的目录）。

“cd -”表示回到上一次所在的工作目录。

- 创建目录 mkdir。用户 harry 想在 programs 下面创建一个新目录“proj1”，再在“proj1”下面创建一个子目录“new”。他可以先用“mkdir”创建“proj1”，再转移到这个目录下创建“new”；也可以：mkdir -p proj1/new 一次完成。选项“-p”表示：如果所创建目录的上层目录（父目录）还不存在，就同时创建。
- 删除一个空目录，使用“rmdir pathname”。非空目录不能使用 rmdir 删除。
- pwd 可以打印当前的绝对路径名。此命令在一般的操作过程中作用不大，因为通常 shell 的提示符中已经显示了路径名。它主要用在软件开发过程中不便于人工干预的场合。

1.2.3 文件操作

文件操作包括文件复制、改名、移动、删除等。对文件的操作命令，多数也可以对目录操作。

- 修改属性 chmod。拥有文件操作权限的用户可以根据需要修改文件的权限属性。例如，harry 将自己辖下的 videos 目录开放给其他用户，可以使用命令“chmod 777 videos”。其中的数字是八进制，与属性字段除第一位以外的每一位相对应。
- 文件复制 cp。“cp oldfile newfile”，将 oldfile 复制一份到 newfile。如果最后一个参数是目录，则表示把前面几个复制到该目录下。当复制的源包含目录时，需要使用选项“-r”。
- 删除文件 rm。被删除的文件无法通过正常操作恢复。这与在桌面环境中将文件移除至垃圾桶的行为不同。删除目录同样也需要选项“-r”。
- 文件改名 mv。“mv oldfile newfile”，旧文件名被新文件名替代。如果需要一次性给多个文件改名，Linux 没有原生的批量重命名命令，此项功能可以通过一些命令的组合实现。1.4.3 节提供了一种方案。

当最后一个参数是目录时，mv 操作的结果是将前面所列的文件和目录移至该目录下。

- 链接命令 ln。链接的结果有点像文件复制，但本质上是不一样的。链接的文件和原始文件只有名字的差别，数据在存储设备上只存储了一份。带有选项“-s”创建的链接又称为软链接或符号链接，它比较像 Windows 系统的“快捷方式”，链接文件依赖于原始文件的存在。当原始文件被移动或删除后，软链接文件就成为“死链”。
  硬链接则是所有链接文件都指向相同的数据区，每个链接文件的地位是平等的，不存在依赖关系。只有当最后一个链接被删除后，存储空间才被真正释放。硬链接只能是同一个分区里的文件（不包括目录），软链接则无此限制。
  链接功能由文件系统支持，只有支持 inode 型或 vnode 型结构的文件系统才能创建链接。（inode 普遍见于 Linux 系统的文件系统，vnode 主要见于 BSD 系统。）
  链接在 Linux 系统中使用非常普遍。动态链接库通过链接的形式允许多个版本共存。Linux 有一个基本软件包 BusyBox，里面包含了上百个 Linux 的基本操作命令，这些命令都指向同一个文件 busybox。如果要升级 BusyBox，只需要更换这一个文件就可以了。
- 查找文件 find。find 查找文件方式很灵活，通过不同选项，可以根据文件名、文件大小、文件最后修改时间等多种属性查找。对于找到的文件，还可以使用命令直接对这些文件进行操作。例如，查找当前目录下（包括子目录）超过 10KB 的文件并复制到 harry 的 videos 目录：

```
$ find . -size +10K -exec cp {} /home/harry/videos \;
```

  解释：选项“-size”告诉 find 按文件大小查找，“-exec”告诉 find 找到的文件后执行后面的命令，“{}”匹配找到的每一个文件；命令行中的分号“;”是 shell 中顺序执行一系列命令的分割符，为避免被 shell 误解为是“find”命令的分隔符，这里用“\”转义。

对文件和目录的操作可以使用通配符。通配符是用于代替某一个或一串具体字符的特殊字符。常用的通配符有“*”、“?”和“[...]”。例如，图 1.2 中，harry 要把当前目录（假设是 videos）里所有“.c”和“.h”结尾的文件复制到 programs 目录中，他可以这样操作：

```
$ cp *.c *.h ../videos/
```

这里的“*”用于替代任意字符串，它省去了对逐个文件操作的烦琐过程。通配符“?”用于替代任意单个字符；“[...]”中的符号相当于一个清单，例如“[Cc]”表示“C”或“c”，“[a-z]”表示所有小写字母。

1.2.4　浏览文件

除了利用编辑器观察文件内容以外，还可以用一些轻量级工具直接在终端上查看文件内容：

- more。此命令将长文件分屏显示，每满一屏暂停一下，等待用户敲键以后再继续上滚一屏。
- head/tail。用于浏览文件头部或尾部。选项“-n”用于指定行数。默认的是 10 行；也可以用“-c”选项指定以字节为单位。tail 还有一个有用的选项“-f”（follow 的意思），当浏览一个不断变化着的文件（例如某个软件的日志文件）时，可以跟踪显示这个文件。
- cat。此命令可以将文件和标准输入设备合并，送到标准输出设备上。

1.2.5 打包、压缩和解压

打包工具可以将多个文件组装成单个文件，便于携带和传输。多数打包工具同时具备数据压缩功能。Linux 系统最常用的打包压缩命令是 tar，它会根据选项自动调用压缩/解压程序。常见的压缩格式后缀是 gz、bz2 和 xz，它们各自表示采用不同的压缩算法。不同的压缩算法影响到数据压缩率、压缩时间和解压时间。一般而言，xz 格式的压缩率最高，而 gz 格式的压缩、解压时间最短，但压缩率最低。

将一个.tar.gz 、.tar.bz2 或.tar.xz 文件解压的命令可以统一使用“-xf”选项，例如（tar 命令的选项前面的“-”可以去掉）：

```
$ tar xf hello-2.10.tar.gz
```

打包压缩时，通过选项“-z”、“-j”、“-J”指定压缩格式，分别对应使用 gzip、bzip2 和 xz 进行压缩，例如：

```
$ tar Jcf hello-2.10.tar.xz hello-2.10/
```

1.2.6 进程控制

当程序在终端运行时，操作人员可以通过终端与程序交互：通过键盘输入，程序把输出打印在终端上。交互过程中，以下是一些特殊的操作：

- 按 Ctrl+C 组合键[①]表示提前结束程序。
- 按 Ctrl+Z 组合键时暂停程序，此时用户获得终端控制权。如果需要继续运行程序，可以在此终端输入命令“fg”或者“bg”，前者表示前台方式（foreground），后者表示后台方式（background）。
- 如果程序向终端打印的内容太快、太多，可以通过按 Ctrl+S 组合键暂停终端显示。暂停状态下，按任意键继续。

有些程序运行时不需要通过终端和操作人员交互（特别是图形界面程序），这时可以在程序启动时直接使用后台方式。在命令行最后加一个字符“&”就表示以后台方式运行。后台运行的程序一般不再占用终端，终端可以继续其他操作。

系统命令“ps”可以显示系统中所有进程的状态（使用选项“-ax”）。进程状态列表中的第一列数字表示进程标识符（进程号）。通过“kill”命令向进程发送信号可以改变进程的运行状态。命令格式是：

```
$ kill -<signal> processID
```

该命令的本义是向进程发送信号，而不是英文单词的原义。命令格式中，“signal”是一个整数，对应表 1.1 中的不同含义。默认方式下，kill 向进程发送 SIGTERM 信号，该信号通常导致进程的结束。（该命令因此得名。）有时候，进程会忽略这个信号，因此，为了确保让该进程结束，可以使用命令“kill -9 processID”。SIGKILL 是不可阻塞的。另一个发送信号的命令是“killall”，它以进程名为操作参数而不是进程 ID。

① 表示在按下 Ctrl 键的同时按下 C 键。

表 1.1　Linux 系统中一些常用的信号描述

信　号　名	signal 值	说　　明
SIGHUP	1	挂起
SIGINT	2	中断
SIGQUIT	3	进程退出
SIGABRT	6	异常中止
SIGKILL	9	杀死进程（不可阻塞）
SIGUSR1	10	用户定义的信号 1
SIGSEGV	11	段错误（存储器访问非法）
SIGUSR2	12	用户定义的信号 2
SIGPIPE	13	管道错误
SIGALRM	14	闹钟
SIGTERM	15	进程中止
SIGCHLD	17	子进程状态改变
SIGCONT	18	进程继续
SIGSTOP	19	进程停止（不可阻塞）

1.2.7　管道与重定向

管道是进程间通信的一种方式。Linux 每个进程运行时，默认打开了三个设备：标准输入设备、标准输出设备和标准错误输出设备。通常，标准输入设备是键盘，后两个是用于字符显示的终端。在命令行中使用管道，可以理解为将前一个程序的输出送到另一个程序作为输入。输入和输出必须是标准设备。

管道的操作符是“|”。例如，要浏览某个文件的第 200～225 行，可以将 head 和 tail 两个命令通过管道连接起来：

```
$ head -n 225 file|tail -n 26
```

tail 将最后结果仍显示在终端上。

对于原本在终端上输出的内容，如果想将它送到其他地方（如保存到文件），或者原来从键盘上输入的内容用文件代替，可以通过重定向功能实现。

重定向操作符是“>”和“<”，分别表示输出重定向和输入重定向。上面的命令改成

```
$ head -n 225 file|tail -n 26> newfile
```

则将打印在屏幕上的内容保存到 newfile 文件里。如果文件已经存在，则被覆盖。“>>”也是输出重定向，不同的是，它在重定向输出的文件后面追加而不是覆盖。

Linux 的基本命令大都包含在 BusyBox 软件包中（桌面系统不一定直接使用 BusyBox，但功能一样）。这些命令短小精悍、功能专一。由于选项太多，给使用者造成一定的困难。为此，系统默认方式下安装了一份详细的帮助手册，通过命令“man”可以查看命令的详细用法（“man”是英语“手册页”的前三个字母）。手册按以下方式分类：

（1）可执行程序或 shell 命令。

（2）系统调用（内核提供的函数）。

（3）库调用（程序库中的函数）。

（4）特殊文件（通常位于/dev 目录）。

（5）文件格式和规范，如/etc/passwd。

（6）游戏。

（7）杂项（包括宏包和规范，如 man(7)，groff(7)）。

（8）系统管理命令（通常只针对 root 用户）。

手册中不仅包括命令，还包括系统调用函数和库函数，给编程带来了很大方便。例如，“man 3 printf”可以打印函数 printf( )的格式化帮助文档。如果“man”命令中不提供分类号，则按顺序打印第一个找到的函数或命令帮助文档。

1.2.8 shell 脚本程序

设想这样一个情景：管理员拿到一份名单，要求根据名单为每个人创建一个新账户。

新用户名单 userlist.txt 如表 1.2 所示。

表 1.2 新用户名单 userlist.txt

harry	Harry Potter
ron	Ron Weasley
hermione	Hermione Granger
charity	Charity Burbage
armando	Armando Dippet
albus	Albus Dumbledore

创建账户需要用“useradd”命令为系统添加一个用户名，用“passwd”为其设置初始口令，在该用户的主目录下建立初始化工作环境的脚本文件。这是一项烦琐枯燥的工作。幸好，shell 除了提供一个操作环境外，它也是一种脚本程序的运行环境。根据名单要求，可以编写这样一个脚本（如清单 1.1 所示）。

清单 1.1 批量创建账户的脚本 account.sh

```
1   #!/usr/bin/bash
2
3   while read line; do
4       name =` echo $line | sed -e "s/ .* $ // " `
5       useradd -m $name
6       echo -e $name "\n" $name | passwd $name
7       cp -f /etc/profile /home/$name/.profile
8       chown $name.$name /home/$name/.profile
9   done
```

文件的头两个字节“#!”是脚本文件的标识。如果文件具有可执行属性，直接运行这个程序会调用这个标识符后面的命令对该脚本加以解释。也可以直接指定命令运行该脚本（如“sh account.sh”），此时，标识符不起作用。

用上面的脚本创建批量用户就变成了

```
# sh account.sh <userlist.txt
```

程序循环地从标准输入设备上读取一行（第 3 行。“read”是 shell 命令，从标准输入设备读取字符串赋值给变量“line”，此处标准输入重定向自文件 userlist.txt），将读到的一行文字从空格开始直至行尾的字符去除，得到账户的用户名（第 4 行），使用命令“useradd”（需

要超级用户权限）创建账户。再为其创建初始口令（第 6 行），初始口令与用户名相同，设置口令时需要输入两次。最后，将系统的环境配置文件复制给该用户并改变其权限属性。

除了 shell 脚本以外，Linux 还有 Perl、Python 等其他多种语言的脚本程序。脚本程序简化了编程的手续，减少了重复性操作，大大提高了工作效率。

1.3　规则表达式

当我们在编辑一个文件时，想把文件中的一串字符用另一串字符替代，绝大多数的编辑器都可以打开一个对话框，只要在对话框中填写替换的字符串就行了。但如果想替换的不是一个明确的字符串，而是具有某一类特征的字符串，如“找出含有 4 个字母的单词，其中间两个字母相同”，或者“将所有句号后面的多余的空格删除，只保留一个空格”这样的要求，用常规的编辑手段就会很辛苦了。

规则表达式（Regular Expression，又称正则表达式，缩写为“RegExp”或“RegEx”）是用一组特定的字符集描述文本特征的数学语言。搜索引擎使用规则表达式进行关键字查找和过滤；使用规则表达式的文本编辑器和文字处理软件，可以使搜索、替换和匹配的工作更为准确高效。一些编程语言通过内建或动态库的方式支持规则表达式。

规则表达式中，除了直接匹配的字母、数字和下画线以外，定义了下面的匹配规则：

\d 匹配任意一个数字，\D 匹配非数字字符。

\w 匹配任意一个字母、数字或下画线，等效于[A-Za-z0-9_]，\W 匹配字母、数字和下画线以外的字符。

\s 匹配空白字符，包括空格、制表符、回车换行，\S 表示匹配除这些字符以外的字符。

\b 匹配单词边界，\B 匹配非边界。

exp1|exp2 匹配表达式 exp1 或 exp2。

^该符号本身表示匹配字符串起始位置。如果在[...]中，表示匹配除[　]中所列字符以外的字符。

$匹配字符串结尾。

.匹配除换行符以外的任意一个字符。

[...]匹配[　]中的任意一个字符。如果字符的 ASCII 码处于连续范围，可以用“-”连接，如[A-Za-z]，表示所有大小写字母。

*重复前面的表达式任意次（包括 0 次）。

+重复前面的表达式 1 次以上。如[0-9]+表示由任意个（至少是 1 个）数字组成的数字串。

?重复前面表达式 0 次或 1 次。

{*m, n*}重复前面的表达式 *m*～*n* 次。

（...）匹配括号内的表达式，并将其作为一个群组。其后可用*n* 形式替代，其中 *n* 是一个数字，用于指代该群组出现在表达式中的顺序编号。编号从 1 开始。

用于规则表达式功能的特殊字符，如“?”、“.”、“[”、“+”等，如需要表示其本身，应使用转义符“\”。例如要匹配“www.example.com”，准确的写法是“www\.example\.com”，否则可能匹配到“wwwwexample/com”。

表示反斜线本身也需转义，即“\\”匹配一个反斜线。

本节开头所提到的两个例子，分别可以用“\b[A-Za-z]([A-Za-z])\1[A-Za-z]\b”和“\. +”进行匹配。在一些软件中实际使用规则表达式时，对一些符号可能要做转义处理。

1.4 与开发相关的常用命令

1.4.1 文件比较

软件开发经常会涉及代码修改和版本升级。对于修改前后的代码有哪些差别，两个相似的文件中有哪些不同，程序“diff”可以帮助找到答案。“diff”并非只是逐字逐句地对比。有些场合下，文字差异可能没有实质意义。例如对于多数程序设计语言来说，一个空格和连续多个空格或者制表符没有差别；对于有些语言，字母大小写也不加区分。此时，可以利用“diff”的不同选项将这些差异排除，开发人员可以更集中精力关注主要问题。

以一个标准的“Hello, world”程序编写过程为例，清单 1.2 是第一版 hello.c。

清单 1.2 第一版 hello.c

```
1   #include <stdio.h>
2
3
4   int main(int    argc, char ** argv)
5   {
6       printf(" hello, world \n");
7       return 0;
8   }
```

开发人员随后发现，打印的字符串首字母应该大写。于是，将 hello.c 复制一份到 hello.c.orig 作为备份，重新修改原文档为如清单 1.3 所示的第二版 hello.c。

清单 1.3 第二版 hello.c

```
1   #include <stdio.h>
2
3   int main(int argc, char ** argv)
4   {
5       printf("Hello,world!\n");
6       return 0;
7   }
```

为了找出这两个版本的不同，运行“diff -wB hello.c.orig hello.c”，程序打印如下信息：

```
6c5
<   printf("hello,  world \n");
---
>   printf("Hello, world!\n");
```

选项“-w”忽略空格，“-B”忽略空行。根据这个显示结果，我们就可以知道新的文件在原来的文件基础上做了哪些修改。比较的结果不仅可供阅读分析，还可以直接用于软件打补丁。为此，需要先将比较的结果作为文件保存：

```
$ diff -wB hello.c.orig hello.c> hello.patch
```

再使用补丁命令“patch”（正如它的名字）将补丁文件作用在原文件上。该命令的输入通常以文件重定向方式处理：

```
$ patch hello.c.orig <hello.patch
```

经此操作后的“hello.c.orig”就与目前新版的一致了。

diff 和 patch 经常配合使用。“diff”不仅可以比较单个文件，如果使用选项“-r”，还可以对两个相似的目录结构进行对比，但通常不用于二进制文件的比较，仅限于文本文件（包括编码的文本文件）。虽然选项“-a”可以将二进制文件以文本文件的方式进行对比，但限于目前的技术手段，如果对一个非文本文件进行修改（打补丁），只能是全文替换。

压缩的 Linux 内核源码超过 100MB，但每一次升级更新，与之前相比，改动的内容要远少于 100MB。开发人员如果已经有了当前版本的源码，下次升级时下载补丁比下载一个完整的源码会省很多时间。一些版本控制系统（见 2.6 节）正是采用这样的思路。数据库备份也经常采用这种方法。

1.4.2　文本搜索

“grep”是一个基于规则表达式的文本搜索工具，可以用于查找文件中的关键字，默认的方式是将含有关键字的行打印出来。下面的例子可查找当前各级子目录中含有“CFLAGS”开头行的所有 Makefile 文件：

```
$ grep -n --color=auto "^CFLAGS" `find . -name "Makefile"`
```

bash 环境中反引号（“`”）的功能是，将其中的命令的执行结果作为字符串输出。“--color”是一个有用的选项，它将符合规则的信息用特殊的颜色显示出来（需要终端支持）。如果只想找到包含关键字的文件，而不关心关键字在文件中的位置，可以使用选项“-l”。

grep 曾经有几个变体“egrep”、“fgrep”、“rgrep”，目前 Linux 系统已不再使用，仅出于兼容性目的，作为 grep 的符号链接而保留，具体功能通过 grep 的选项实现。

1.4.3　流编辑

流编辑工具“sed”不同于普通文本编辑器。在编辑文件时，它不需要直接介入编辑过程，而是通过一些命令对文件进行修改；它还有一个文本编辑器无法替代的功能是，它可以修改 shell 的变量（见清单 1.1 中对变量 line 的处理语句）。

下面的命令在当前目录下所有.c 文件第一行前面插入一行注释信息“/* This is a demo project */”：

```
$ sed -i "1i\/\* This is a demo project \*\/" *.c
```

想想看，如果这个目录下面有若干个子目录，有 100 个各式各样的文件，用普通的文本编辑器干这件事会多么麻烦。

再举一例，将所有后缀为“.jpeg”的文件重命名为“.jpg”：

```
$ for f in `ls *.jpeg`; do mv $f `echo $f|sed "s/\.jpeg/.jpg/"`; done
```

解释如下：将“ls”命令的输出字符串逐一赋值给变量“f”；使用命令“sed”将变量“f”中的字符串“.jpeg”修改为“.jpg”，作为“mv”命令的第二个参数；如此循环。

默认方式下，“sed”将修改的结果打印在终端。选项“-i”表示直接在文件里修改。对文件的操作命令包含在引号中。常用的文件编辑命令有以下几种。

*n*iexp：在第 *n* 行前插入表达式 exp。

*n*aexp：在第 *n* 行后插入表达式 exp。

*m, n*cexp：将第 *m*～*n* 行用表达式 exp 替换。

*m, n*d：删除第 *m*～*n* 行。

s/exp1/exp2/：将第一个表达式 exp1 替换成 exp2。这里，用于分割两个表达式的符号“/”称为定界符。如要全部替换，在最后一个定界符后面增加一个字母“g”。“s”前面可以用 *m, n*（*m, n* 是数字）限定行号作用范围。

sed 使用的定界符很灵活，紧跟在 s 命令后面的符号都被认为是定界符，可以根据个人习惯、风格使用“:”、“|”、“#”等，但在同一命令中应保持前后一致。定界符用于表示字符本身时，需要转义。命令行中的引号作为字符本身使用时也要转义。

sed 的功能不限于编辑文件，例如“sed -n "200,225p" file”可打印“file”文件的第 200～225 行，它可以代替前面的例子中用管道串联的 tail 和 head 的功能。

1.5 文本编辑工具

软件开发离不开文本编辑工具。良好的文本编辑器有助于提高编程效率，减少错误。Linux 有两款非常著名的文本编辑器（vim 和 emacs），它们都有丰富的扩展特性，支持个性化配置，各自拥有众多的用户。这里主要介绍 vim 的特点及使用。

vim（Vi IMproved）在 UNIX 的全屏编辑器 vi 基础上发展而来。vim 的作者 Bram Moolenaar 最早在 Amiga 系统上开发了这款编辑器。从 4.0 版本开始，vim 增加了图形界面支持。图形界面版本具备命令行版本的全部功能，只是由于一些按键操作用于菜单选择，在使用图形界面版本时可能要适当调整一些键盘映射。

1.5.1 vim 工作模式

用 vim 编辑文件，只要在命令行输入“vim 文件名”，即可进入 vim 环境，开始对指定文件的编辑工作。用“gvim”或者“vim -g”则可打开图形界面窗口。vim 有以下三种工作模式。

1．全屏控制模式

以默认方式运行 vim，最先进入的就是这种模式。这种模式下可以对整个文档进行操作，包括查找、替换、删除、粘贴，控制编辑界面、编辑方式等。当输入插入、修改、替换等按键命令时则进入全屏编辑模式；输入“:”则进入底行命令模式。

2．全屏编辑模式

在这种工作模式下，用键盘输入的字符成为文本内容的一部分，直至按 Esc 键，退出全屏编辑模式，回到全屏控制模式。

3．底行命令模式

在全屏控制模式下输入“:”，光标落到底行，并出现提示符“:”。在此提示符下可以

输入面向整个文档的操作命令。底行命令结束后，如果不是退出 vim，则自动回到全屏控制模式[①]。

全屏控制模式下输入“/”或“?”也会将光标定位到底行，此时将进入字符串搜索模式。“/”表示前向搜索，“?”表示反向搜索。

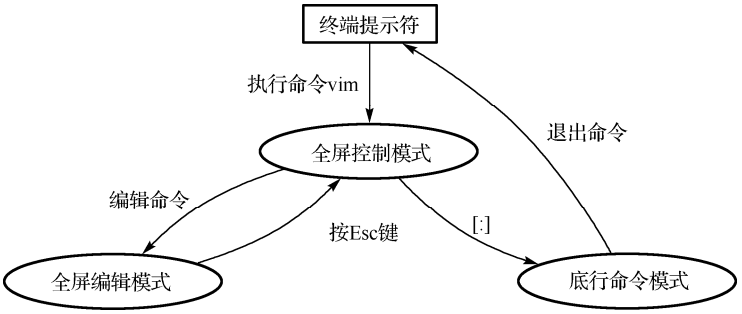


图 1.3　vim 工作模式

1.5.2　vim 常用编辑命令

vim 随带一份详细的联机帮助手册。底行命令输入“help”即可进入帮助首页。这里仅介绍一些常用的操作命令，以便能在较短的时间内上手。

在全屏控制模式下，控制命令一般由一两个按键操作完成。如果在前面加上数字，则表示该命令的重复次数。这些键盘操作通常没有回显，要做到心中有数。

1．光标定位操作

vim 保留了 vi 的操作习惯，全屏控制模式下使用 k、j、h、l 控制光标的上下左右移动，完整版的 vim 也支持方向键。除此以外，可以根据需要利用下面的操作快速定位光标位置：

|　光标移动到当前行行首。

^　光标移动到当前行行首第一个非空字符上。

$　光标移动到当前行行尾。

%　将光标移动到配对的项目上，可用于括号“{ }、[ ]和( )”的配对检查，也可以用于 C 语言的注释“/* ... */”和“#if/#ifdef ... #else ...#endif”的配对。

(　退回句子首部。

)　前移到句子尾部。

{　光标移动至本段段首。

}　光标移动至本段段尾。

[　光标前移至节分界处。

]　光标回退至前一节分界处。

−　退到上一行第一个非空格字符处。

/pattern　向前搜索 pattern，光标定位到 pattern 位置。pattern 是一个规则表达式。

?pattern　类似“/pattern”，但方向是向回搜索。

n　重复上一次匹配模式搜索。

① 底行命令：:a、:i 也可以进入全屏编辑方式，目前较少使用。

N　重复上一次匹配模式搜索，但方向与上次相反。

w　光标前进至下一个字首。

b　光标回退至上一个字首。

e　光标移到下一个字末。

ma　用字母“a”标记当前位置，以后可以用“`a”迅速回位光标。多个位置可以用不同字母标记。

`a　“`”（制表符 Tab 键上面的键）后跟字母表示将光标移至字母标记处。连续两个“``”将光标移至其移动前的位置。

2．文件修改操作

以下操作用于修改当前光标处的文字：

D　删除本行光标后的内容。

J　删除行尾的换行符，将两行连接成一行。

Y　当前行存入缓冲区，以后可以用“p”或“P”取出。

P　将上次删除的文本（缓冲区内容）插在当前光标前面。

p　将上次删除的文本（缓冲区内容）插在当前光标后面。

u　取消上一次的修改操作。

x　删除当前光标位置的一个字符。

X　删除当前光标前的一个字符。

dw　删除一个字。

dd　删除一行。

ZZ　保存文件，退出 vim。

3．进入编辑方式的操作

以下是进入编辑方式的常用操作：

a　在当前光标位置后开始插入文本。

A　在当前行尾开始插入文本。

C　删除从光标处开始到行尾的文本，并进入编辑模式。

i　在当前光标位置前开始插入文本。

I　在当前行首非空字符前开始插入文本。

o　在当前行上方新开一行，进入编辑模式。

O　在当前行下方新开一行，进入编辑模式。

s　删除光标位置的字符，并进入编辑模式。

S　删除光标所在行，并进入编辑模式。

4．底行命令

:w file　将当前编辑的文本写入指定文件。如不指定文件名，则以正在编辑的文件名存盘。

:*m, n*w file　将正在编辑文件的第 *m*～*n* 行写入文件。

:q　退出 vim。文件修改后未保存时，为防止数据丢失，vim 不允许简单地退出。“:q!”表示强制退出。

:x　保存文件并退出 vim。

:e file　编辑另一个文件“file”。

:ar　当同时打开多个文件时，“ar”显示文件列表。

:n　切换到文件列表的下一个文件编辑界面。

:N　切换到文件列表的上一个文件编辑界面。

:rew　切换到文件列表的第一个文件编辑界面。

:la　切换到文件列表的最后一个文件编辑界面。

:r file　将文件“file”插入到当前光标位置，与当前文件合并。

:! command　运行程序“command”后再回到 vim。

:s/exp1/exp2/g　将光标所在行所有 exp1 替换为 exp2。exp1、exp2 均为规则表达式。去掉最后一个字母“g”表示只替换第一个。命令前面可以用“*m*, *n*”指定行号范围，“1,$”或“%”表示整篇文档。

:d　删除光标所在行。前面可以用数字或数字范围表示删除的行号。

:g/exp/d　删除带有表达式 exp 的行。

:*m, n*m*j*　将第 *m*～*n* 行移到第 *j* 行之后。

:*n*　光标定位到第 *n* 行。数字前面加“+”或“-”表示光标相对移动 *n* 行。

1.5.3　vim 高级操作

vim 启动时会读入用户主目录下的“.vimrc”，进行初始化设置。个性化配置通常也写在这个文件里。vim 还提供了丰富的插件，针对不同的编程语言、不同的使用场合做了专门的设置，大大提高了文件编辑效率。例如不同语言的关键字语法高亮、回车后的自动缩进、括号匹配、关键字自动补全，这些功能不仅简化了操作，同时也减少了程序的语法错误。

vim 从 3.0 版本开始支持多窗口操作。在全屏控制模式下输入 Ctrl+W s[①]或者 Ctrl+Wv，可以将编辑窗口水平或者垂直分隔。获取联机帮助时可以在底行输入“:help CTRL-W”这一字符串。这里“CTRL”是 C、T、R、L 四个字母（不区分大小写），不是 Ctrl 键。

在每个窗口中可编辑不同文件，也可以编辑同一文件的不同位置，方便参照对比。Ctrl+W w 用于光标在多窗口之间轮流切换，也可以用 Ctrl+W 配合方向键切换窗口。Ctrl+W<、Ctrl+W> 用于水平调整窗口大小，Ctrl+W+、Ctrl+W-用于垂直方向调整窗口大小。多窗口操作时，“:q”、“:x”命令对单个窗口有效。

全屏控制模式下输入“qa”，vim 以标记“a”记录之后的操作，直至下一次在全屏控制模式中输入“q”。此间记录的操作可以用“a”提取。此功能可简化重复的操作。

全屏控制模式下输入 Ctrl+V 进入列块可视模式，辅以方向控制键选择一个文本块。这种模式打破了行的界限，可以以文本块为单位进行编辑处理。

本 章 练 习

1．观察一下你的 shell 工作环境，其中的环境变量 PATH 包含哪些目录？如果想在环境变量中增加一个目录/usr/local/bin，应该怎样操作？

① 此处的操作是：在按下 Ctrl 键的同时按下 W 键，然后释放，再按下 S 键。

2．如果你有 root 权限，试创建一个拥有普通用户权限的账户，并将其主目录设置为他人不可访问（使用另一个普通用户验证你的结果）。

3．使用“ls -l”命令列出的文件清单中包含一个时间信息，这是与文件相关的什么时间？

4．学习使用 man 了解 wget 和 curl 命令的用法，并使用其中的一个命令下载 GNU hello（https://ftp.gnu.org/gnu/hello/hello-2.10.tar.gz）。

5．将下载后的 GNU hello 解压，将其中所有.c 文件和.h 文件行首插入一行 C 语言格式的注释，注释内容是文件名。可以考虑用脚本实现此功能。

本章参考资源

1．有关 Linux 的基本操作命令，参考随机帮助文档。

2．Linux 文件目录结构标准 FHS，参考网络资源 https://wiki.linuxfoundation.org/lsb/fhs-30。

3．关于规则表达式，参考《核心 PYTHON 应用编程（第 3 版）》（Core PYTHON Applications-Programming，Third Edition）第一章，其作者是 Wesley J. Chun，由 Prentice Hall 出版社于 2012 年出版。

4．vim 编辑器参考自带帮助手册。