第 2 章 入门

1. 在第一行的开头处使用#!这两个字符来告诉系统使用哪一种 shell

```
#! /bin/sh
```

注意：

- #!这一行尽量不要超过 64 个字符。
- 脚本是否具有可移植性取决于是否有完整的路径名称。
- 别在选项之后放置任何空白，因为空白也会跟着选项一起传递给被引用的程序。

2. 以两个破折号（--）来表示选项结尾的用法源自 System V，不过已被纳入 POSIX 标准。自此之后命令行上看起来像选项的任何项目，都将一视同仁地当成参数处理。

```
Patch –verbose –backup -p1 < /tmp/whizprog-1.1-1.2-patch
```

3. 分号（;）可用来隔同一行里的多条命令，如果你使用的是&符号而不是分号，则 Shell 将在后台执行其前面的命令，这意味着，Shell 不用等到该命令完成，就可以继续执行下一条命令。
4. Shell 识别三种基本命令：内建命令、Shell 函数以及外部命令：

- 内建命令就是由 Shell 本身所执行的命令。
- Shell 函数是功能健全的一系列程序代码，以 Shell 语言写成，它们可以像命令那样引用。
- 外部命令就是由 Shell 的副本（新的进程）所执行的命令。

5. 当你想取出 Shell 变量的值时，需于变量名称前面加上$字符。当所赋予的值内含空格时，请加上引号：

```
Fullname = “$first $middle $last”
```

6. 简单的 echo 输出：echo 的任务是产生输出，可用来提示用户，或是用来产生数据供进一步处理。

```
$echo -n “Enter your name:”
Enter your name:”_                    // 键入数据，下划线是终端光标
```

7. 华丽的 printf 输出：如同 echo 命令，printf 命令可以输出简单的字符串：

```
Printf “Hello,world\n”
```

printf 不像 echo 那样会自动提供一个换行符号，必须显式地将换行符指定成\n。

8. printf 命令的完整语法分为两部分：

```
Printf format-string [arguments……]
```

其中，format-string 是一个字符串，用来描述输出的排列方式，最好为此加上引号。arguments 是与格式声明相对应的参数列表，例如一系列的字符串或变量值。

```
$Printf “The first program always prints ‘%s,%s!’\n” Hello world
The first program always prints ‘ Hello world!’
```

9. 重定向

- 以<改变标准输入：

  program < file 可将 program 的标准输入修改为 file 文件。

- 以>改变标准输出：

  program > file 可将 program 的标准输出修改为 file 文件。

- 以>>附加到文件：

  program >> file 可将 program 的标准输出附加到 file 的结尾处。

- 以|建立管道

  program1 | program2 可将 program1 的标准输出修改为 program2 的标准输入。

10. 管道可以把两个以上执行中的程序衔接在一起，第一个程序的标准输出可以变成第二个程序的标准输入。这样做的好处是，管道可以使得执行速度比使用临时文件的程序快上十倍。

11. Shell 中 tr 命令用来从标准输入中通过替换或删除操作进行字符转换。

12. 当程序打开/dev/tty 时，UNIX 会自动将它重定向到一个终端（一个实体控制台或串行端口，也可能是伪终端），再秘程序结合，这在程序必须读取人工输入时（例如密码）特别有用。

13. 所谓的位置参数指的是 Shell 脚本的命令行参数。在 Shell 函数里，它们同时也可以是函数的参数。各参数都由整数来命名。当它超过 9，就应该用大括号把数字框起来：

```
Echo first arg is $1
echo tenth arg is ${10}
```

14. Shell 注释由#开头，Shell 会忽略由#开头的每一行。

15. 代码的执行跟踪使用 set -x 命令打开，使用 set +x 关闭，也可以使用+号打开，加号后面跟着一个空格：

```
+ who                    #跟踪命令，该命令输出会显示出来
```

16. 对用户而言，用来控制让哪种语言或文化环境生效的功能就叫 locale。

第 3 章 查找与替换

1. 使用 grep 程序查找文本：

```
Who | grep -F austen
```

2. 基本正则表达式(BRE)和扩展正则表达式(ERE)：

   (1) 正则表达式由两个基本组成部分所建立：一般字符与特殊字符。特殊字符常称为元字符，使用 meta 字符表示。

   (2) 如果需要使用 meta 字符本身，如*号，可使用转义字符\，如*就代表星号自身。

   (3) 要匹配单个字符，可使用方括号表达式，如 c[aeiouy]t 匹配 cat、cot、cut 等等。在方括号里面,^放在字首表示取反，所以^[aeiouy]指小写元音字符以外的任何字符。[0-9a-fA-F]表示匹配所有 0 到 9、小写 a 到 f、大写 A 到 F 的符号。

   (4) 在方括号表达式中，所有其他的 meta 字符都会推动其特殊含义。所以[*\.]匹配的就是星号、反斜杠以及句号自身。要将]进入该集合，可以将它放在列表的最前面：[]*\.]，要让减号进入该集合，也请将它放到列表最前端：[-*\.]。若需要右方括号与减号进入列表，请将右方括号放到第一个字符，减号放到最后一个字符：[]*\.-]

   (5)星号表示匹配 0 个或多个前面的单个字符，但不能用*表示“匹配三个字符，而不是四个字符”。当需要匹配次数时，需要一个复杂的方括号表达式：

\{*n*\}	前置正则表达式所得结果重现 *n* 次
\{*n*,\}	前置正则表达式所得的结果重现至少 *n* 次
\{*n*,*m*\}	前置正则表达式所得的结果重现 *n* 至 *m* 次

   有了区间表达式，要表达像“重现 5 个 a”或是“重现 10 到 42 个 q”就变得简单了，分别是：a\{5\}和 q\{10,42\}。

3.