下面介绍的特殊“变量”提供了对传递的参数的总数的访问，以及一次对所有参数的访问：

$#

提供传递到Shell脚本或函数的参数总数。当你是为了处理选项和参数而建立循环时，它会很有用（在稍后的6.4节里会说明）。举例如下：

```
while [ $# != 0 ]                    以shift逐渐减少$#，循环将会终止
do
     case $1 in
     ...                             处理第一个参数
     esac
     shift                           移开第一个参数（见稍后内文说明）
done
```

$*, $@

一次表示所有的命令行参数。这两个参数可用来把命令行参数传递给脚本或函数所执行的程序。

"$*"

将所有命令行参数视为单个字符串。等同于"$1 $2 ..."。$IFS的第一个字符用来作为分隔字符，以分隔不同的值来建立字符串。举例如下：

```
printf "The arguments were %s\n" "$*"
```

"$@"

将所有命令行参数视为单独的个体，也就是单独字符串。等同于"$1" "$2" ...。这是将参数传递给其他程序的最佳方式，因为它会保留所有内嵌在每个参数里的任何空白。举例如下：

```
lpr "$@"                                显示每一个文件
```

set命令可以做的事很多（详见7.9.1节说明）。调用此命令而未给予任何选项，则它会设置位置参数的值，并将之前存在的任何值丢弃：

```
set -- hi there how do you do       -- 会结束选项部分，自hi开始新的参数
```

shift命令是用来“截去(lops off)”来自列表的位置参数，由左开始。一旦执行shift，$1的初始值会永远消失，取而代之的是$2的旧值。$2的值，变成$3的旧值，以此类推。$#值则会逐次减1。shift也可使用一个可选的参数，也就是要位移的参数的计数。单纯的shift等同于shift 1。以下范例将这些操作串联在一起，并添加了注释：

```
$ set -- hello "hi there" greetings      设置新的位置参数
$ echo there are $# total arguments      显示计数值
there are 3 total arguments
$ for i in $*                            循环处理每一个参数
> do   echo i is $i
> done
```

```
i is hello                                    注意，内嵌的空白已消失
i is hi
i is there
i is greetings
$ for i in $@                                 在没有双引号的情况下，$* 与 $@ 是一样的
> do   echo i is $i
> done
i is hello
i is hi
i is there
i is greetings
$ for i in "$*"                               加了双引号，$* 表示一个字符串
> do   echo i is $i
> done
i is hello hi there greetings
$ for i in "$@"                               加了双引号，$@ 保留真正的参数值
> do   echo i is $i
> done
i is hello
i is hi there
i is greetings
$ shift                                       截去第一个参数
$ echo there are now $# arguments             证明它已消失
there are now 2 arguments
$ for i in "$@"
> do   echo i is $i
> done
i is hi there
i is greetings
```

6.1.2.3 特殊变量

除了我们看过的特殊变量（例如$#及$*）之外，Shell还有很多额外的内置变量。有一些也具有单一字符、非文字或数字字母的名称；其他则是全由大写字母组成的名称。

表6-3列出内置于Shell内的变量，以及影响其行为的变量。所有Bourne风格的Shell提供的变量都比这里所列的多很多，它们会影响交互模式下的使用，也可以在处理Shell程序时用于其他的用途。不过下面要说明的这些，是在写Shell程序时，可以完全倚赖实现可移植性脚本编程的变量。

表6-3：POSIX内置的Shell变量

变量	意义
#	目前进程的参数个数。
@	传递给当前进程的命令行参数。置于双引号内，会展开为个别的参数。
*	当前进程的命令行参数。置于双引号内，则展开为一单独参数。
-（连字号）	在引用时给予Shell的选项。

表6-3：POSIX内置的Shell变量（续）

变量	意义
?	前一命令的退出状态。
$	Shell进程的进程编号（process ID）。
0（零）	Shell程序的名称。
!	最近一个后台命令的进程编号。以此方式存储进程编号，可通过wait命令以供稍后使用。
ENV	一旦引用，则仅用于交互式Shell中；$ENV的值是可展开的参数。结果应为要读取和在启动时要执行的一个文件的完整路径名称。这是一个XSI必需的变量。
HOME	根（登录）目录。
IFS	内部的字段分隔器；例如，作为单词分隔器的字符列表。一般设为空格、制表符（Tab），以及换行（newline）。
LANG	当前locale的默认名称；其他的LC_*变量会覆盖其值。
LC_ALL	当前locale的名称；会覆盖LANG与其他LC_*变量。
LC_COLLATE	用来排序字符的当前locale名称。
LC_CTYPE	在模式匹配期间，用来确定字符类别的当前locale的名称。
LC_MESSAGES	输出信息的当前语言的名称。
LINENO	刚执行过的行在脚本或函数内的行编号。
NLSPATH	在$LC_MESSAGES(XSI)所给定的信息语言里，信息目录的位置。
PATH	命令的查找路径。
PPID	父进程的进程编号。
PS1	主要的命令提示字符串。默认为"$"。
PS2	行继续的提示字符串。默认为"> "。
PS4	以set -x设置的执行跟踪的提示字符串。默认为"+ "。
PWD	当前工作目录。

特殊变量$$可在编写脚本时用来建立具有唯一性的文件名（多半是临时的），这是根据Shell的进程编号建立文件名。不过，系统里还有一个mktemp命令也能做同样的事，这些都会在第10章中探讨。

6.1.3 算术展开

Shell的算术运算符与C语言里的差不多，优先级与顺序也相同。表6-4列出支持的算术运算符，优先级由最高排列至最低。虽有些是（或包含）特殊字符，不过它们不需以反