Bash shell 编程

Linux 将命令合并处理过程转为文件执行，不使用命令，该文件的内容就是shell脚本，他是非交互式的。执行非交互式的shell时，在/etc/bashrc和.bashrc文件中检索BASH\_ENV(ENV)环境变量。先读取BASH\_ENV，然后在shell脚本执行命令。执行交互式的shell时，使用-norc或者--norc选项就不能读取BASH\_ENV或ENV变量.

#!称为魔术数字(magic number)向内核告知需要在脚本解析行的程序。该行必须位于脚本的顶端。

Bash shell程序由Linux命令和Bash shell命令、程序创建者、注释等组成。可以在echo命令的双引号内使用反单引号`｀执行命令，如：

Echo “现在的时间是:`date`”

Bash shell 脚本中可以使用$(shell指令)格式来执行shell命令。

echo -e “your name:\c” ; read answer ;echo “the answer is ${answer}”

7.2.2 read命令

read命令是内置命令，用于从终端或文件当中读取用户的字符串。Read命令会读取一行，直到发现newline为止，行尾的newline读取为null。Ｒｅａｄ在用户输入ｅｎｔｅｒ之前，程序将保持终止状态。Ｒｅａｄ命令使用-r选项则忽略反斜杠和一对newline，反斜杠可以看成是行的一部分。read有-a\-e\-p\-r 四种选项。

|  |  |
| --- | --- |
| read first second | 从标准输入读取一行，以空格或者是newline为准，将第一个词分配给first、其余的分配给second |
| Read | 从标准输入读取一行，将结果值分配给REPLY内置变量 |
| Read -a arrayname | 读取命名为arrayname的数组词目录 |
| Read -e | 用于交互式shell. |
| read -p prompt | 显示prompt,等待用户输入，输入的内容保存到REPLY内置变量 |
| Read -r line | 允许输入包含反斜杠的内容 |

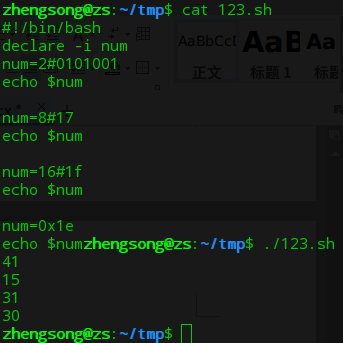
7.3　算术运算

使用declare　-i命令可以将变量声明为整数型，Bash给已经声明为整数型的变量分配字符串０，若给整数型变量分配字符串时，该变量的值会变为０。已经声明为整形数的变量可以进行算术运算。若变量没有声明为整形数，则可以使用let内置命令进行算术运算。若想给声明为整型变量的变量分配实数，Bash就会提示语法错误。



7.3.1.3 进制的标记和使用

格式：变量名＝进制＃数字



、

7.3.1.4 let命令

Let是Bash的内置命令，执行整数型算术运算并测试数字表达式。

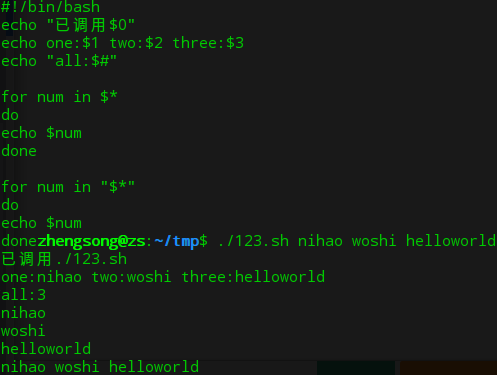
7.3.2 实数型算术运算

Bash只支持整数型运算，但是使用bc、awk实用工具可以执行更复杂的运算

7.4.1 位置参数

可以通过命令行将需要的参数信息传递给脚本，每一个以脚本名后空格分开的词将成为参数。$9后面的参数会像${10}一样使用花括号。使用$#变量测试参数的个数，$\*或$@变量显示所有的参数。可以使用set命令设置或重置位置参数。使用set命令可以删除之前设置的所有位置参数。

|  |  |
| --- | --- |
| 位置参数变量 | |
| $\* | 拥有位置参数的所有目录 |
| $@ | 等于$\*，使用双引号的情况除外 |
| “$\*” | 扩展为单一参数，如”$1 $2　$3” |
| “$@” | 分隔并扩展参数，如”$1” “$2” ”$3” |



7.5 条件语句和分支语句

7.5.1 终止状态

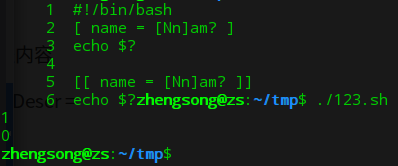
条件语句以条件结果是否成功决定是否执行。终止状态变量“？”拥有终止状态的数值，使用echo $?　查看终止变量的值就可以判断上一条命令的执行结果，值为０表示执行成功，为非零表示执行失败。

7.5.2 test命令和let命令

7.5.2.1单方括号[]和test命令

Bash2.x版本的双方括号[[]]可以用于判断表达式，前方括号之后必须输入有空格。包含空格的常数字符串必须使用引用符号，若是字符串就检测为正确的字符串，而非模式的一部分。Test命令的模式运算符使用-a和-o运算符。

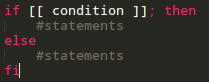
test命令可以评估字符串和数字，也可以测试文件的执行情况，test命令和所有命令一样可以返回终止状态，返回值为０表示正确，为非零表示错误。可以使用方括号[]代替test命令，使用方括号[]时，前方括号必须要有空格，且＝、！＝、-a、-o等测试符号的前后必须要有空格。Test命令和单方括号不允许使用通配符，因为?变量会被视为常数字符。但是当使用的是双方括号时，则允许使用通配符。

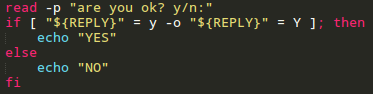


7.5.2.2 let命令和双括号的算术运算

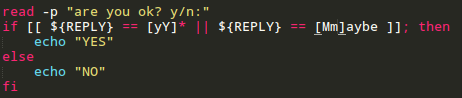
Test命令可以评估表达式，但是let命令的使用就像c语言一样，使用let命令等于在双括号(( ))内使用表达式。

7.5.2.3　if命令





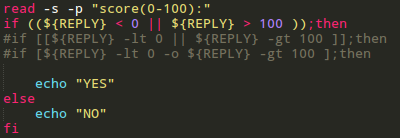
为什么这里使用双方括号的话会提示有语法错误。[[ ]]双方括号应该怎样使用才正确。



7.5.2.4 exit命令和 ? 变量

使用exit命令终止脚本并返回命令行，脚本发生情况时可以使用exit命令终止脚本的运行，exit命令参数可以使用０～２５５之间的数字，exit (0)表示正常退出。退出的参数将保存于 ? 变量当中。

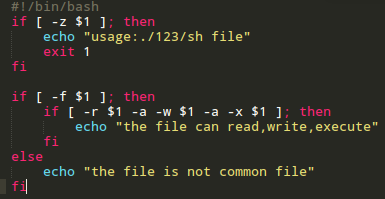
7.5.4 if/else命令



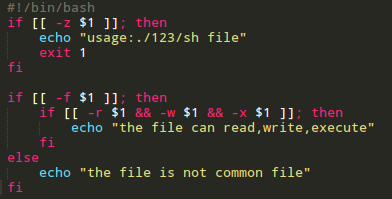
7.5.5 文件测试命令

创建shell脚本时经常使用文件测试命令，有时需要测试文件的可用性、文件具备的许可权限（读、写、执行）是什么，此时可以使用几种选项。文件测试运算符的种类如下：

|  |  |
| --- | --- |
| 文件测试运算符 | 为真时的含义 |
| -b file | 文件存在且为块特殊文件就显示为真 |
| -c file | 文件存在且为字符特殊文件就显示为真 |
| -d file | 文件存在且为目录就显示为真 |
| -e file | 文件存在就显示为真 |
| -f file | 文件存在且为一般文件就显示为真 |
| -G file | 文件存在且属于有效组ＩＤ就显示为真 |
| -g file | 存在且为set-group id就显示为真 |
| -k file | 文件设置粘滞位就显示为真 |
| -L file | 存在且为链接 |
| -p file | 存在且为已命名（named）管道 |
| -O file | 存在且为所属有效用户ＩＤ |
| -r file | 存在且可读 |
| -S file | 存在且为套接字 |
| -s file | 存在且大小大于０ |
| -t fd | 打开ｆｄ且为终端就显示为真 |
| -u file | 存在且设置set-user-id位就显示真 |
| -w file | 存在且可写就显示真 |
| -x file | 存在且可执行 |

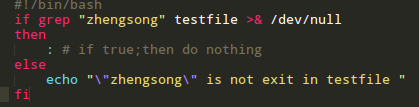


上面的格式 [ ] 是test旧版本的格式，使用混合格式　[[ ]] test命令可以进行如下的更改：

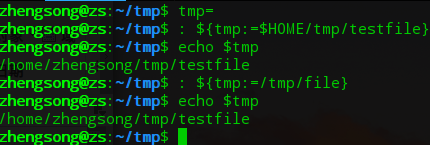


7.5.6 null命令

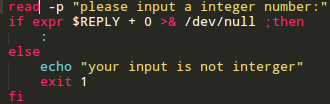
Null命令是内置的命令，以冒号（：）标记，不执行任何命令并返回终止状态值０。If命令后面没有任何执行命令时，可以使用null命令。但是因then 语句后需要命令，所以显示错误信息。例：



使用grep命令检索testfile中是否存在字符串zhengsong，将结果和错误发送到/dev/null,若存在该字符串，则什么也不做，若不存在，则显示文件不存在。Then后面的（：）既是null命令，不执行任何的操作。



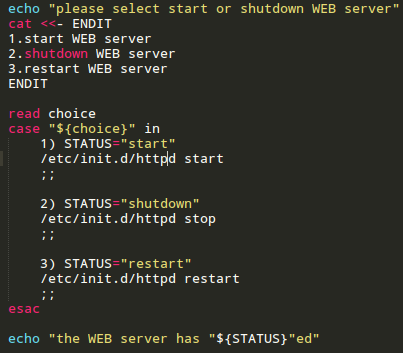
上面给tmp变量赋值为空，冒号：命令不执行任何作业，　:=修饰符返回分配给变量的值。上面的示例将表达式传递给不执行任何作业的null命令参数。此时shell会替换变量，若未给tmp变量分配值，就分配该路径，此时，tmp变量的属性设置为无法修改。因此在之后再一次的使用:=重新分配值的时候，并不能分配成功。



使用expr命令评估表达式，只有当REPLY为整数时，$REPLY + 0表达式才显示为真，否则为假。　expr命令评估表达式需要详细了解。

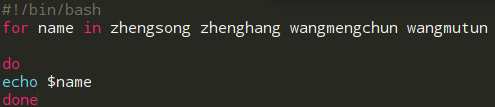
7.5.7 case 命令

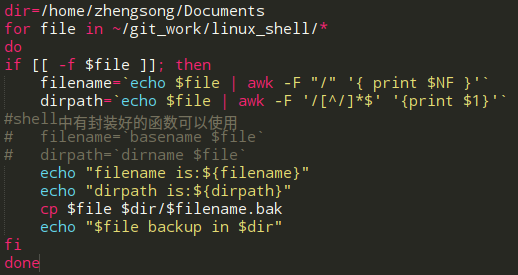
Case可以用if/then/elif/then/else/fi来代替，case命令末尾使用esac，他是case的倒写，表示case的结束。若未匹配case变量，程序将一直执行直到发现\*)之后的；；或者是esac为止。Case值可以使用shell通配符，也可以使用竖杠（｜）；例：



上面的例子中创建使用了here文档。　　cat <<- ENDIT命令是here文档，该文档使用ENDIT字符串标记cat命令的结束。也就是说，通过cat命令显示到屏幕的字符串是以ENDIT字符串开始的行的前端。

7.6.1 for循环语句命令





如上的示例脚本可以用来备份文件，获取文件名和文件的路径名可以使用awk 来获取，也可以使用shell提供的filename和dirname命令来获取。

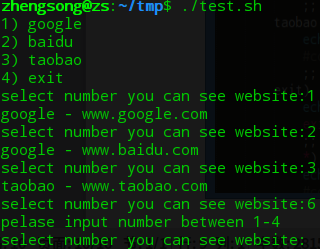
7.6.3 while循环语句命令

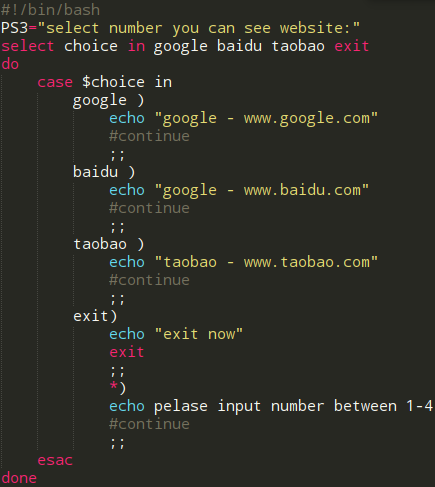
While命令评估接下来输入的命令，若是终止状态为0就执行循环语句本身（do～done）

7.6.5 select命令和菜单

使用here文档可以轻松的创建菜单，但是bash提供了另外的一种循环语句机制select循环语句，数字目录项目的菜单显示标准错误。PS3提示符用来输入，默认设置的PS3为#？。显示PS3提示符之后，shell就等待着用户的输入。此时的输入值必须是菜单目录数字之一。该输入值保存于特殊的shell变量REPLY，保存在REPLY变量中的数字与选择目录括号右侧的字符串相关联。

Case命令与select命令一起使用，用户可以从菜单和选择、命令执行当中选着select命令，使用LINES和COLUMNS变量可以决定现实到终端的菜单项目页面。以标准输出显示结果，每一个项目位于数字和括号之前。PS3提示符显示于菜单的底端。Select命令是循环语句命令，故使用break命令退出循环语句，使用exit命令终止脚本。





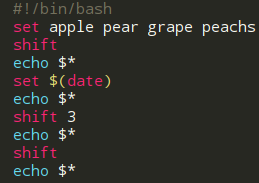
7.6.6 循环语句管理命令

有些情况需要退出循环语句、回到循环语句顶端或者终止无限循环。此时提供可以用于Bash shell的循环语句管理命令。

7.6.6.1 shift命令

使用shift命令按照指定的数字向左移动参数目录。无参数的shift命令可将参数目录向左移动一次。也就是说，删除最左侧的参数。永久删除已移动的参数，有时会在while循环语句中使用shift命令重复位置参数目录。





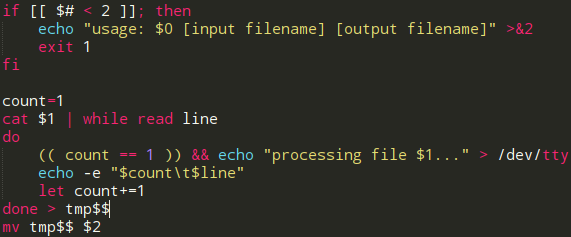
上述示例使用了shift命令，首先使用set命令设置了4个参数，执行shift命令之后，使用echo命令显示$\*值就能够显示所有的参数，之前使用shift向左移动了一个参数，所以全部的参数变为了3个。再次使用set命令将date命令的结果设置为参数并执行shift 3，也就是向左移动并删除3个参数。

7.6.6.4重叠循环语句和循环语句管理

使用重叠循环语句时，可以向break命令和continue命令赋予数字（整形参数），所以可以从内侧循环语句向外侧移动。

7.6.7 IO重定向和子shell

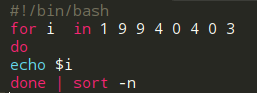
I/O均可以在文件中使用循环语句建立管道或重定向。Shell开启子shell处理I/O重定向和管道，终止循环语句时，定义于其内部的变量不会留在脚本当中。



上例使用cat命令显示第一个位置参数$1的文件，然后使用while循环语句执行管道操作。此时read命令给循环语句文件的第一行文件的第一行，给下一个循环语句分配第二行的同时执行循环语句，若是成功读取输入的内容read命令的状态值就返回0.否则返回1.

do～done的命令中，count变量就变为1执行echo命令，向/dev/tty发送结果值。使用echo命令显示count、tab键、line变量值，再重复执行循环语句。读取了所有的行以后执行重定向，并保存到done后面的语句tmp$$文件，最后将tmp$$文件更改为输入的第二个参数名。

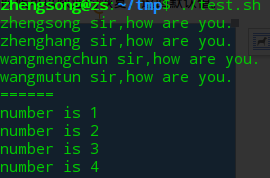
7.6.7.2 用管道链接循环语句结果和指令

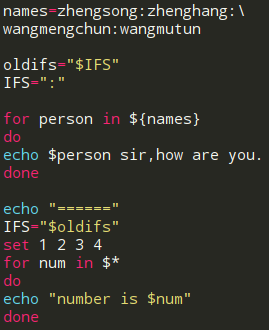


上例对for循环语句显示未排序的数字，使用（循环语句结束的）done关键字的下一个管道和sort -n命令对数字进行排序，之后字shell显示升序排列的循环语句。

7.6.9 IFS和循环语句

Shell内部的字段分隔符（IFS）评估空格、tab、newline字符，并且可以成为read、set、for等词目录解析命令的词分隔符。用户需要使用其他的分隔符时可以重置。还可以在更改值之前将IFS的初始值保存到其他的变量中，需要的时候可以快速的恢复，返回默认值。





7.7 函数

函数是命令名或者命令组名，使用函数可以将程序模块化并且提高结构效率。函数在当前的shell环境执行，且不会生成子shell。先创建函数专用文件，并且在制作脚本时为使用函数专用文件做好准备，然后才能在脚本中执行。

格式：function 函数名 {命令；命令；...}

7.7.1 解除函数设置

格式：unset -f 函数名

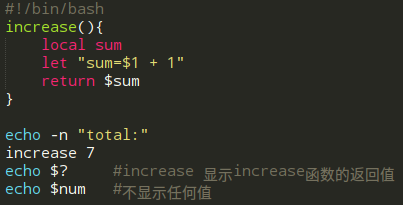
7.7.2 导出函数

格式：export -f 函数名； #向子shell导出函数就可以在子shell中使用函数。

7.7.3函数参数和返回值

因为在当前的shell内执行函数，所以变量在函数和shell内均可以使用。

1. 参数：使用位置参数可以将参数传递给函数
2. 内置local函数：局部变量只能用于函数内部，使用local函数定义局部变量。终止函数就能够删除局部变量。
3. 内置return函数：可以在为终止函数而调用的函数位置上使用return命令返回程序管理。若不使用return命令指定参数，函数的返回值则成为脚本最后使用命令的终止状态值。若return命令分配的值存在，该值就被保存到“？”变量且拥有0~255的整数值，因为return命令只能返回0~256的整数值。还可以使用其他命令获取函数结果，可以使用括号内以$开始的全部函数（$(函数名)），或者传统的反单引号（``）获取命令结果并且分配到变量。



在上例中，使用内置local函数创建sum变量，这样就可以在终止increase函数的时候删除内存中的sum变量，然后使用let命令，给sum变量分配$1加1的值。调用函数时使用return命令返回sum变量值。此时return命令的参数保存于“？”变量。若是return语句未指定使用的参数，就表示函数使用的最后命令的终止状态值。