Linux SHELL中的变量详述

本篇文章来源于 Linux公社网站(www.linuxidc.com) 原文链接：<http://www.linuxidc.com/Linux/2008-01/10285p2.htm>

一，SHELL本地变量：

本地变量就如同局部变量一样，只在本SHELL中起作用。它不会影响到其他SHELL中的变量。

格式：NAME=value

1，变量的调用：在变量前加$

$ echo $HOME

/home/hbwork

$ WEEK=Satur

$ echo Today is $WEEKday

Today is

$echo Today is ${WEEK}day //若变量和其他字符组成新的字,这时就必须给变量加上大括号{}，以更加清楚的显示给shell,哪个是真正的变量，以实现字符串的合并等功能。

Today is Saturday

2，显示变量值

可以使用echo命令。需要注意的是，使用变量的时候必须在变量名前加上$符号。另外可以使用set命令，显示所有本地SHELL变量。包括SHELL中预定义了一些环境变量，且用户自己已经定义过的本地变量也会在其中显示。

3，Shell变量赋值从右到左进进行赋值(变量未赋值时，shell不报错，而是赋值为空!)

$ X=$Y Y=y

$ echo $X

y

$ Z=z Y=$Z

$ echo $Y

$

4，使用unset命令删除变量的赋值

$ Z=hello

$ echo $Z

hello

$ unset Z

$ echo $Z

$

5，有条件的命令替换 （测试变量是否已经赋值）

在Bourne Shell中可以使变量替换在特定条件下执行，即有条件的环境变量替换。 这种变量替换总是用大括号括起来的。

${variable: -value}变量替换时将使用命令行中定义的默认值，但变量的值并没有因此而改变。variable是一变量值，value是变量替换使用的默认值

例：$echo Hello $UNAME

结果显示:Hello

$echo Hello ${UNAME: -there}

结果显示:Hello there

$echo $UNAME

结果显示: (空)

$UNAME=John

$echo Hello ${UNAME: -there}

结果显示:Hello John

${variable:=value}不但使用默认值进行替换，而且将默认值赋给该变量。该形式在变量替换后同时把值value符给变量variable。

例：$echo Hello $UNAME

结果显示:Hello

$echo Hello ${UNAME:=there}

结果显示:Hello there

$echo $UNAME

结果显示:there

变量替换的值也可以是` `括起来的命令:

$USERDIR={$Mydir: -`pwd`}

${variable: +value}只有当变量已赋值时才用指定值替换，变量variable已赋值时，其值才用value替换，否则不进行任何替换。

例：$ERROPT=A

$echo ${ERROPT: +"Error tracking is acitive"}

结果显示:Error tracking is acitive

$ERROPT=

$echo ${ERROPT: +"Error tracking is acitive"}

结果显示: (空)

${variable:?message}使用错误检查的条件进行变量替换，当变量variable已设置时，正常替换。否则消息message将送到标准错误输出(若此替换出现在shell程序中,那么该程序将终止)。

例：$UNAME=

$echo $ {UNAME:?"UNAME HAS NOT BEEN SET"}

结果显示:UNAME HAS NOT BEEN SET

$UNAME=Stephanie

$echo $ {UNAME:?"UNAME HAS NOT BEEN SET"}

结果显示:Stephanie

当没有指定message时，shell将显示一条默认的消息，例如:

$UNAME=

$echo $ {UNAME:?}

结果显示:sh:UNAME:parameter null or not set

本篇文章来源于 Linux公社网站(www.linuxidc.com) 原文链接：<http://www.linuxidc.com/Linux/2008-01/10285.htm>

6，特殊设置

readonly 变量名

有时我们想要在说明一个变量并对它设置为一个特定值后就不在改变它的值时，可以用下面的命令来保证一个变量的只读性：单独执行readonly可以显示所有的只读变量。

export 变量名

　　　　在任何时候，创建的变量都只是当前Shell的局部变量，所以不能被Shell运行的其他命令或Shell程序所利用，而export命令可以将一个局部变量提供给Shell执行的其他命令使用，也可以在给变量赋值的同时使用export命令（export 变量名=变量值）。使用export说明的变量，在Shell以后运行的所有命令或程序中都可以访问到。

二，Shell参数

1、 位置参数

　　由系统提供的参数称为位置参数。位置参数的值可以用$N得到，N是一个数字，如果为1，即$1.类似C语言中的数组，在shell解释用户的命令时，将把命令行的第一个字作为命令，而其他的字作为参数。当命令对应的可执行文件为Shell程序时，这些参数将作为位置变量传送给该程序。第0号为程序名字，从1开始就表示传递给程序的参数。第一个参数记为$1,第二个为$2....第九个为$9。其中1到9是真正的参数名，"$"符只是用来标识变量的替换。

2、 内部参数

　　上述过程中的$0是一个内部变量，它是必须的，代表程序本身，而$1则可有可无。和$0一样的内部变量还有以下几个。

$# ----传递给程序的总的参数数目

　　$? ----上一个代码或者shell程序在shell中退出的情况，如果正常退出则返回0，反之为非0值。

　　$\* ----传递给程序的所有参数组成的字符串。

$- ----在Shell启动或使用set命令时提供选项

$? ----上一条命令执行后返回的值

$$ ----当前shell的进程号

$! ----上一个子进程的进程号

$@ ----所有的参数，每个都用双括号括起

$n ----位置参数值，n表示位置

$0 ----当前shell名

　　例：建立一个内容为如下的程序P1:

　　echo “Program name is $0”

　　echo “There are totally $# parameters passed to this program”

　　echo “The last is $?”

　　echo “The parameters are $\*”

　　执行后的结果如下：

　　[beichen@localhost bin]$ P1 this is a test program //传递5个参数

　　Program name is /home/beichen/bin/P1 //给出程序的完整路径和名字

　　There are totally 5 parameters passed to this program //参数的总数

　　The last is 0 //程序执行结果

The parameters are this is a test program //返回有参数组成的字符串0

三，环境变量：

shell 环境变量是所有shell 程序都会接受的参数。shell程序运行时，都会接收一组变量，这组变量就是环境变量，它作用于所有的用户进程。在Linux中，登陆进程称为父进程，shell中执行的用户程序均称为子进程。

环境变量可以在命令行中设置，但用户注销时这些值将丢失。因此最好在$HOME/目录下的.profile中定义。传统上环境变量均为大写。环境变量应用于用户进程之前，必须用export命令导出。设置方法与本地变量设置方法相同。

1，显示环境变量

使用env命令或者 printenv 命令可以察看所有环境变量

2，将变量导出到子进程

在父脚本中使用export命令可以将变量导出，这样子脚本便可以知道该变量的值。

如果你希望把你定义的变量让其他所有的shell程序都能使用，也就是定义新的环境变量，你只要使用export关键词就可以了。

　 3，常用的Shell环境变量有：

　　HOME 用于保存注册目录的完全路径名。

　　PATH 用于保存用冒号分隔的目录路径名，Shell将按PATH变量中给出的顺序搜索这些目录，找到的第一个与命令名称一致的可执行文件将被执行。

　　TERM 终端的类型。

　　UID 当前用户的识别字，取值是由数位构成的字串。

　　PWD 当前工作目录的绝对路径名，该变量的取值随cd命令的使用而变化。

PS1 主提示符，在特权用户下，默认的主提示符是#，在普通用户下，默认的主提示符是$。

附：

bash的内部命令

　　bash命令解释套装程序包含了一些内部命令。内部命令在目录列表时是看不见的，它们由Shell本身提供。常用的内部命令有：echo, eval, exec, export, readonly, read, shift, wait和点(.)。下面简单介绍其命令格式和功能。

　　1．echo

　　命令格式：echo arg

　　功能：在屏幕上显示出由arg指定的字串。

　　2．eval

　　命令格式：eval args

　　功能：当Shell程序执行到eval语句时，Shell读入参数args，并将它们组合成一个新的命令，然后执行。

　　3．exec

　　命令格式：exec命令参数

　　功能：当Shell执行到exec语句时，不会去创建新的子进程，而是转去执行指定的命令，当指定的命令执行完时，该进程（也就是最初的Shell）就终止了，所以Shell程序中exec后面的语句将不再被执行。

　　4．export

　　命令格式：export变量名或：export变量名=变量值

　　功能：Shell可以用export把它的变量向下带入子Shell，从而让子进程继承父进程中的环境变量。但子Shell不能用export把它的变量向上带入父Shell。

　　注意：不带任何变量名的export语句将显示出当前所有的export变量。

　　5．readonly

　　命令格式：readonly变量名

　　功能：将一个用户定义的Shell变量标识为不可变。不带任何参数的readonly命令将显示出所有只读的Shell变量。

　　6．read

　　命令格式：read变量名表

　　功能：从标准输入设备读入一行，分解成若干字，赋值给Shell程序内部定义的变量。

　　7．shift语句

　　功能：shift语句按如下方式重新命名所有的位置参数变量，即$2成为$1，$3成为$2…在程序中每使用一次shift语句，都使所有的位置参数依次向左移动一个位置，并使位置参数$#减1，直到减到0为止。

　　8．wait

　　功能：使Shell等待在后台启动的所有子进程结束。wait的返回值总是真。

　　9．exit

　　功能：退出Shell程序。在exit之后可有选择地指定一个数位作为返回状态。

　　10．“.”（点）

　　命令格式：. Shell程序文件名

　　功能：使Shell读入指定的Shell程序文件并依次执行文件中的所有语句。

运行Shell程序的方法

　　用户可以用任何编辑程序来编写Shell程序。因为Shell程序是解释执行的，所以不需要编译成目的程序。按照Shell编程的惯例，以bash为例，程序的第一行一般为“#！/bin/bash”，其中 # 表示该行是注释，叹号！告诉Shell运行叹号之后的命令并用文档的其余部分作为输入，也就是运行/bin/bash并让/bin/bash去执行Shell程序的内容。

　　执行Shell程序的方法有3种。

　　1．bash Shell程序文件名

　　这实际上是调用一个新的bash命令解释程序，而把Shell程序文件名作为参数传递给它。新启动的Shell将去读指定的文件，可执行文件中列出的命令，当所有的命令都执行完后结束。该方法的优点是可以利用Shell调试功能。

　　2． bash< Shell程序名

　　这种方式就是利用输入重定向，使Shell命令解释程序的输入取自指定的程序文件。

　　3．用chmod命令使Shell程序成为可执行的

　　一个文件能否运行取决于该文档的内容本身可执行且该文件具有执行权。对于Shell程序，当用编辑器生成一个文件时，系统赋予的许可权都是644(rw-r-r--)，因此，当用户需要运行这个文件时，只需要直接键入文件名即可。

　　在这3种运行Shell程序的方法中，最好按下面的方式选择：当刚创建一个Shell程序，对它的正确性还没有把握时，应当使用第一种方式进行调试。当一个Shell程序已经调试好时，应使用第三种方式把它固定下来，以后只要键入相应的文件名即可，并可被另一个程序所调用。

　bash程序的调试

　　在编程过程中难免会出错，有的时候，调试程序比编写程序花费的时间还要多，Shell程序同样如此。

　　Shell程序的调试主要是利用bash命令解释程序的选择项。调用bash的形式是：

　　bash -选择项Shell程序文件名

　　几个常用的选择项是：

　　-e 如果一个命令失败就立即退出。

　　-n 读入命令但是不执行它们。

　　-u 置换时把未设置的变量看做出错。

　　-v 当读入Shell输入行时把它们显示出来。

　　-x 执行命令时把命令和它们的参数显示出来。

　　上面的所有选项也可以在Shell程序内部用“set -选择项”的形式引用，而“set +选择项”则将禁止该选择项起作用。如果只想对程序的某一部分使用某些选择项时，则可以将该部分用上面两个语句包围起来。

　　（1）未置变量退出和立即退出

　　未置变量退出特性允许用户对所有变量进行检查，如果引用了一个未赋值的变量就终止Shell程序的执行。Shell通常允许未置变量的使用，在这种情况下，变量的值为空。如果设置了未置变量退出选择项，则一旦使用了未置变量就显示错误信息，并终止程序的运行。未置变量退出选择项为-u。

　　当Shell运行时，若遇到不存在或不可执行的命令、重定向失败或命令非正常结束等情况时，如果未经重新定向，该出错信息会显示在终端屏幕上，而Shell程序仍将继续执行。要想在错误发生时迫使Shell程序立即结束，可以使用-e选项将Shell程序的执行立即终止。

　　（2）Shell程序的跟踪

　　调试Shell程序的主要方法是利用Shell命令解释程序的-v或-x选项来跟踪程序的执行。-v选择项使Shell在执行程序的过程中，把它读入的每一个命令行都显示出来，而-x选择项使Shell在执行程序的过程中把它执行的每一个命令在行首用一个+加上命令名显示出来。并把每一个变量和该变量所取的值也显示出来。因此，它们的主要区别在于：在执行命令行之前无-v，则显示出命令行的原始内容，而有-v时则显示出经过替换后的命令行的内容。

　　除了使用Shell的-v和-x选择项以外，还可以在Shell程序内部采取一些辅助调试的措施。例如，可以在Shell程序的一些关键地方使用echo命令把必要的信息显示出来，它的作用相当于C语言中的printf语句，这样就可以知道程序运行到什么地方及程序目前的状态。

本篇文章来源于 Linux公社网站(www.linuxidc.com) 原文链接：http://www.linuxidc.com/Linux/2008-01/10285p2.htm