Linux——shell脚本基础1：变量

[简介及分类 1](#_Toc323575304)

[环境变量 1](#_Toc323575305)

[本地变量 3](#_Toc323575306)

[位置变量 4](#_Toc323575307)

[特定位置变量参数 4](#_Toc323575308)

[附：环境变量的设置及修改 5](#_Toc323575309)

##### 简介及分类

|  |
| --- |
| 变量： |
| 变量可以定制本身的工作环境。使用变量可以保存有用的信息，是系统获取用户相关的设置，变量也用于保存暂时信息。  在默认情况下，所有变量都被看做字符串并以字符串来存储，即使它们被复制为数值时也是如此  变量赋值   1. 赋值等号两边不能有空格 2. 字符串通常放在双引号中，以防止被空白字符分开，同时又允许$扩展 |
| 分类: |
| 四种变量：环境变量、本地变量、位置变量、特定变量参数  环境变量可作用于所有子进程  本地变量在用户现在的shell 生命期的脚本中使用，仅存在于当前进程  位置变量：作为程序参数  特定变量：特殊作用 |

##### 环境变量

|  |
| --- |
| 环境变量 |
| 可在命令行中设置，但是用户注销时值将丢失  最好在.profile中设置  系统管理员可在/etc/profile中设置 |
| 设置环境变量 |
| $var-Name=value  $export var-Name  【退出登陆后也丢失】  或者  $export MYNAME=”ken” |
| 显示环境变量 |
| $echo $var-Name  建议加{} |
| 查看所有环境变量 |
| $env |
| 消除环境变量 |
| $unset var-Name |
| 修改环境变量 |
| $MYNAME=”tom” |

|  |  |
| --- | --- |
| 嵌入shell变量[预留变量名，不做其他用途] | |
| CDPATH |  |
| EXINIT |  |
| HOME | 当前用户主目录【家目录】 |
| IFS | 输入域分隔符。当shell读入输入时，它给出的分割单词的一组字符。通常是空格、制表符及换行符 |
| LOGNAME | 当前用户的登录名 |
| MAIL |  |
| MAILCHECK | 每隔多少秒检查是否有新的信件 |
| MAILPATH |  |
| PATH | 决定了shell将到哪些目录中寻找命令或程序, 执行命令时所搜寻的目录  冒号分隔，用来搜索命令的目录列表 |
| PS1 | 基本命令提示符，对于root用户是#，对于普通用户是$【在命令列时的提示号】 |
| PS2 | 附属二级提示符，默认是“>”， 当命令尚未打完时,Shell 要求再输入时的提示号 |
| SHELL | 当前用户Shell类型 |
| TERMINFO |  |
| TERM |  |
| TZ | 时区 |
| EDITOR |  |
| PWD |  |
| PAGER |  |
| MAINPATH | man 指令的搜寻路径 |
| USER |  |

##### 本地变量

|  |  |  |  |
| --- | --- | --- | --- |
| 本地变量 | | | |
| 本地变量在用户现在的Shell生命周期中使用，只在 用户当前Shell生命期有意义，若在shell中启动另一进程或者退出，此值将无效  优点：用户不能对其他地Shell或进行设置此变量有效 | | | |
| 设置方式： | | | |
| $Var-name=value  或者  ${Var-name=value}  若取值包含空格，可用双引号引起  例子：  $GREAT=”12345”  $echo $GREAT  $echo ${GREAT} | | | |
| Var-Name=value | | | 赋值 |
| Var-Name+value | | | 若设置了，重新赋值，否则赋值 |
| Var-Name:?value | | | 若未设置，显示未设用户错误信息 |
| Var-Name?value | | | 若未设置，显示系统错误信息 |
| Var-Name:=value | | | 若未设置，设置 |
| Var-Name:-value | | | 同上，但取值并不设置到var-Name，可以被替换 |
| 使用declare | | | |
| declare [option] Var-name=value | | | |
| -a | to make NAMEs arrays (if supported) 数组 | | |
| -f | to select from among function names only | | |
| -F | to display function names (and line number and source file name if  debugging) without definitions | | |
| -i | to make NAMEs have the `integer' attribute 整型 | | |
| -r | to make NAMEs readonly  只读变量（值不可改变，可保证关键值不被覆盖，不仅可以标识变量，还可以标识数组，定义 declare -r | | |
| -t | to make NAMEs have the `trace' attribute | | |
| -x | to make NAMEs export 等价于export | | |
| 例子：  declare -a array  array[0]=jack  array2=(tomcat java mysql) | | | |
| 消除变量 | | | |
| $unset variable-name  例子:  $PC=”enterprise”  $unset PC  $echo $PC | | | |
| 显示所有本地shell变量 | | | |
| $set | | | |
| 结合变量值(将变量并排) | | | |
| $echo ${v1}${v2} | | | |
| 测试变量是否取值： | | | |
| $unset tom  $echo “${tom:?}” //若未设置返回系统错误信息  $echo “${tom:?”MyErrorMsg”}” //未设置返回自定义错误信息  $echo “The sky is ${COLOUR:-grey} today” //若未设置，使用grey，之后COLOUR仍未设置  $ehco “The sky is ${COLOUR:=grey} today” //若为设置，设置之，之后，COLOUR有值 | | | |
| 使用变量保存系统命令参数 | | | |
| $SOURCE=”/etc/passwd”  $DEST=”/tmp/passwd.bak”  $cp $SOURCE $DEST | | | |
| 将变量设置为只读[不能再后期改变之] | | | |
| $var-Name=value  $readonly var-Name | | | |
| bash变量操作 | | | |
| ${var#key} | | 从头到尾匹配，最短 | |
| ${var##key} | | 从头到尾匹配，最长 | |
| ${var%key} | | 从头到尾匹配删除，最短 | |
| ${var%%key} | | 从头到尾匹配删除，最长 | |
| ${var/key/newstr} | | 从头到尾匹配，替换第一个 | |
| ${var//key/newstr} | | 从头到尾匹配，替换全部 | |

##### 位置变量

|  |
| --- |
| 位置变量 |
| 执行脚本时向其传递参数，或传递给函数的  可传递多个，但是使用位置变量参数只能访问前9个, 使用shift命令可以改变这个限制.  脚本名称 A B C D E F  $0 $1 $2 $3 $4 $5 $6  $0表示当前执行的进程名,script 本身的名字,或者在正则表达式中表示整行输出 |

##### 特定位置变量参数

|  |  |
| --- | --- |
| 特定位置变量参数 | |
| 运行时脚本的一些相关信息 | |
| $0 | shell脚本名称 |
| $# | 传递到脚本的参数个数 |
| $$ | shell脚本运行的当前进程ID号【程序的PID】 |
| $! | 后台运行的最后一进程的进程ID号【执行上一个指令的PID】 |
| $\* | 以一个单字符串显示所有想脚本传递的参数，与位置变量参数不同，此项参数可超9个【程序的所有参数】，各个参数之间使用环境变量IFS的第一个字符分隔开 |
| $@ | 是$\*的一种精巧变体，不适用IFS环境变量。但是使用时加引号，并在引号中返回每个参数 |
| %@和$\*的区别在于，  $@表示真正的参数，包括含有空格的参数；以(“$1 $2...”)的形式保存所有输入的命令行参数  $\*只是表示命令行上出现的单词；以(“$1”，“$2”...)的形式保存所有输入的命令行参数  没有特别要求，使用$@ | |
| $- | 显示shell使用的当前选项，与set命令功能相同 |
| $? | 显示最后命令的退出状态，0表示成功，其他值表示出错【执行上一个指令的返回值】  用于检查上一个命令,函数或者脚本执行是否正确.(在Linux中,命令退出状态为0表示该命令正确执行,任何非0值表示命令出错.) |
| $\_ | 保存之前执行的命令的最后一个参数 |

##### 附：环境变量的设置及修改

|  |
| --- |
| 附注：环境变量的设置及修改 |
| Bash启动时读取以下文件  /etc/profile  ~/.bash\_profile  ~/.bashrc |
| 方法一：修改/etc/profile文件 |
| 作用范围：**对所有用户生效（永久的）**  适用场景：你的计算机仅仅作为开发使用时推荐使用这种方法，因为所有用户的shell都有权使用这些环境变量，可能会给系统带来安全性问题。  步骤：  (1)用文本编辑器打开/etc/profile  (2)在profile文件末尾加入所需环境变量  export JAVA\_HOME=/usr/share/jdk1.5.0\_05  export PATH=$JAVA\_HOME/bin:$PATH  (3)重新登录  注：修改文件后要想马上生效还要运行# source /etc/profile不然只能在下次重进此用户时生效  注意：   1. linux下用冒号“:”来分隔路径 2. 大小写必须严格区分 3. 在设置环境变量时特别要注意不能把原来的值给覆盖掉了，这是一种常见的错误。需引用&添加 |
| 方法二：修改**用户目录下的.bash\_profile文件中增加变量** |
| 作用范围：对单一用户生效（永久的）  适用长江：这种方法更为安全，它可以把使用这些环境变量的权限控制到用户级别，如果你需要给某个用户权限使用这些环境变量，你只需要修改其个人用户主目录下的.bashrc文件就可以了。  步骤：   1. 用文本编辑器打开用户目录下的.bashrc文件   $ cd ~ #到用户根目录下  $ ls -a #查看所有文件，包含隐藏的文件  $ vi .bash\_profile #修改用户环境变量文件  (2)在.bashrc文件末尾加入：  export JAVA\_HOME=/usr/share/jdk1.5.0\_05  export PATH=$JAVA\_HOME/bin:$PATH  (3)重新登录 |
| 方法三：直接在shell下设置变量 |
| 作用范围：只对当前shell（BASH）有效（临时的）  适用范围：临时使用。因此这种方法仅仅是临时使用，以后要使用的时候又要重新设置，比较麻烦。  只需在shell终端执行下列命令：  export JAVA\_HOME=/usr/share/jdk1.5.0\_05  export PATH=$JAVA\_HOME/bin:$PATH |