**[Linux的环境变量](http://www.cnblogs.com/Neddy/archive/2011/03/01/1968018.html)**

**一、Linux的变量种类**

      按变量的生存周期来划分，Linux变量可分为两类：

      1、永久的：需要修改配置文件，变量永久生效。

      2、临时的：使用export命令声明即可，变量在关闭shell时失效。

**二、设置变量的三种方法**

      1、在/etc/profile文件中添加变量【对所有用户生效（永久的）】

      用VI在文件/etc/profile文件中增加变量，该变量将会对Linux下所有用户有效，并且是“永久的”。

      例如：编辑/etc/profile文件，添加CLASSPATH变量

      # sudo vi /etc/profile

      export CLASSPATH=./JAVA\_HOME/lib;$JAVA\_HOME/jre/lib

(在文件最后一行输入这一句，（在vi模式下输入I进入编辑模式，加入这句后按esc退出，然后按:wq保存文件并退出）)

      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。

      2、在用户目录下的.bash\_profile文件中增加变量【对单一用户生效（永久的）】

      用VI在用户目录下的.bash\_profile文件中增加变量，改变量仅会对当前用户有效，并且是“永久的”。

      例如：编辑guok用户目录（/home/guok）下的.bash\_profile

      $ vi /home/guok/.bash.profile

      添加如下内容：

      export CLASSPATH=./JAVA\_HOME/lib;$JAVA\_HOME/jre/lib

      注：修改文件后要想马上生效还要运行$ source /home/guok/.bash\_profile不然只能在下次重进此用户时生效。

Ubuntu没有.bash\_profile文件，但是在用户目录下有.profile文件。

      3、直接运行export命令定义变量【只对当前shell（BASH）有效（临时的）】

      在shell的命令行下直接使用[export 变量名=变量值]

      定义变量，该变量只在当前的shell（BASH）或其子shell（BASH）下是有效的，shell关闭了，变量也就失效了，再打开新shell时就没有这个变量，需要使用的话还需要重新定义。

**三、PATH声明，其格式为：**

      PATH=$PATH:<PATH 1>:<PATH 2>:<PATH 3>:------:<PATH N>

      你可以自己加上指定的路径，中间用冒号隔开。环境变量更改后，在用户下次登陆时生效。

      如果想立刻生效，则可执行下面的语句：$source .bash\_profile

      需要注意的是，最好不要把当前路径”./”放到PATH里，这样可能会受到意想不到的攻击。

      完成后，可以通过$ echo $PATH查看当前的搜索路径。这样定制后，就可以避免频繁的启动位于shell搜索的路径之外的程序了。

linux PATH环境变量全解析

关于PATH的作用：  www.2cto.com

PATH说简单点就是一个字符串变量，当输入命令的时候LINUX会去查找PATH里面记录的路径。

比如在根目录/下可以输入命令ls,

在/usr目录下也可以输入ls,

但其实ls这个命令根本不在这个两个目录下。

事实上当你输入命令的时候LINUX会去/bin,/usr/bin,/sbin等目录下面去找你此时输入的命令，

而PATH的值恰恰就是/bin:/sbin:/usr/bin:……。

其中的冒号使目录与目录之间隔开。

   www.2cto.com

关于新增自定义路径：

假设新安装了一个命令在/usr/locar/new/bin下面，

如果想像ls一样在任何地方都使用这个命令，

就需要修改环境变量PATH了。

准确的说就是给PATH增加一个值/usr/locar/new/bin。

需要一行bash命令:

export PATH=$PATH:/usr/locar/new/bin

这条命令的意思为: 使PATH自增:/usr/locar/new/bin,

既PATH=PATH+":/usr/locar/new/bin";

通常的做法是：

把这行bash命令写到/root/.bashrc的末尾，

然后当你重新登陆LINUX的时候（应该是linux启动时就会执行这个文件），

新的默认路径就添加进去了。

当然，

也可以直接用命令：

source /root/.bashrc

执行这个文件重新登陆了。

可以用echo $PATH命令查看PATH的值。

   www.2cto.com

关于删除自定义路径：

如果发现新增的路径/usr/locar/new/bin已经没用了，

可以修改/root/.bashrc文件里面你新增的路径。

或者修改/etc/profile文件删除不需要的路径

修改/root/.bashrc文件，

删除相应环境变量选项，然后

￥source /root/.bashrc

即可。

或者可以利用命令。如要删除PATH里的/usr/local/del/bin:变量，则可直接在命令行里输入

$export PATH=$(echo $PATH | sed 's/:\/usr\/local\/del\/bin:/:/g')

注意：＂／＂代表转意字符

比如要把/etc/apache/bin目录添加到PATH中，方法有三：

1.

$PATH=$PATH:/etc/apache/bin

使用这种方法,只对当前会话有效，

也就是说每当登出或注销[系统](http://www.2cto.com/os/" \t "http://www.2cto.com/os/201211/_blank)以后，PATH 设置就会失效

2.

$vim /etc/profile

在适当位置添加 PATH=$PATH:/etc/apache/bin （注意：＝ 即等号两边不能有任何空格）

这种方法最好,除非你手动强制修改PATH的值,否则将不会被改变

   www.2cto.com

3.

$vim ~/.bash\_profile

修改PATH行,把/etc/apache/bin添加进去

这种方法是针对用户起作用的

NOTE：

想改变PATH，必须重新登陆才能生效，以下方法可以简化工作：

如果修改了/etc/profile，

那么编辑结束后执行

$source profile(source /etc/profile)

或 执行点命令

$ ./profile,

PATH的值就会立即生效了。

这个方法的原理就是再执行一次/etc/profile shell脚本，

注意如果用sh /etc/profile是不行的，因为sh是在子shell进程中执行的，即使PATH改变了也不会反应到当前环境中，

但是source是在当前 shell进程中执行的，所以我们能看到PATH的改变。

这样你就学会[Linux](http://www.2cto.com/os/linux/" \t "http://www.2cto.com/os/201211/_blank)系统下修改环境变量PATH路径的方法。

  www.2cto.com

补充说明

工作环境设置文件

环境设置文件有两种：系统环境设置文件 和 个人环境设置文件

1.系统中的用户工作环境设置文件：

  登录环境设置文件：/etc/profile

  非登录环境设置文件：/etc/bashrc

2.用户个人设置的环境设置文件：

  登录环境设置文件: $HOME/.bash\_profile   //这个是环境变量设置的地方

  非登录环境设置文件：$HOME/.bashrc       //这个是定义别名的地方

登录环境：指用户登录系统后的工作环境

非登录环境：指用户再调用子shell时所使用的用户环境

[Linux中profile、bashrc、bash\_profile之间的区别和联系](http://blog.csdn.net/chenchong08/article/details/7833242)

/etc/profile:此文件为系统的每个用户设置环境信息,当用户第一次登录时,该文件被执行.并从/etc/profile.d目录的配置文件中搜集shell的设置.

英文描述为：

# /etc/profile

# System wide environment and startup programs, for login setup  
# Functions and aliases go in /etc/bashrc

# It's NOT a good idea to change this file unless you know what you  
# are doing. It's much better to create a custom.sh shell script in  
# /etc/profile.d/ to make custom changes to your environment, as this  
# will prevent the need for merging in future updates.

所以如果你有对/etc/profile有修改的话必须得重启你的修改才会生效，此修改对每个用户都生效。

/etc/bashrc:为每一个运行bash shell的用户执行此文件.当bash shell被打开时,该文件被读取.

英文描述为：

# /etc/bashrc

# System wide functions and aliases  
# Environment stuff goes in /etc/profile

# It's NOT a good idea to change this file unless you know what you  
# are doing. It's much better to create a custom.sh shell script in  
# /etc/profile.d/ to make custom changes to your environment, as this  
# will prevent the need for merging in future updates.

如果你想对所有的使用bash的用户修改某个配置并在以后打开的bash都生效的话可以修改这个文件，修改这个文件不用重启，重新打开一个bash即可生效。

~/.bash\_profile:每个用户都可使用该文件输入专用于自己使用的shell信息,当用户登录时,该文件仅仅执行一次!默认情况下,他设置一些环境变量,执行用户的**.bashrc**文件.

此文件类似于/etc/profile，也是需要需要重启才会生效，/etc/profile对所有用户生效，~/.bash\_profile只对当前用户生效。  
  
~/**.bashrc**:该文件包含专用于你的bash shell的bash信息,当登录时以及每次打开新的shell时,该文件被读取.（每个用户都有一个.bashrc文件，在用户目录下）

此文件类似于/etc/bashrc，不需要重启生效，重新打开一个bash即可生效，  /etc/bashrc对所有用户新打开的bash都生效，但~/**.bashrc**只对当前用户新打开的bash生效。  
  
~/.bash\_logout:当每次退出系统(退出bash shell)时,执行该文件.   
  
另外,/etc/profile中设定的变量(全局)的可以作用于任何用户,而~/**.bashrc**等中设定的变量(局部)只能继承/etc/profile中的变量,他们是"父子"关系.  
   
~/.bash\_profile 是交互式、login 方式进入bash 运行的；  
~/**.bashrc** 是交互式 non-login 方式进入bash 运行的；  
通常二者设置大致相同，所以通常前者会调用后者。