

输入关键字



系列专栏 (<http://www.jellythink.com/archives/category/columns>) 关于果冻 (<http://www.jellythink.com/about>)



2015-10-21 分类 : Linux Shell (<http://www.jellythink.com/archives/category/linuxdevelopment/linux-shell/>) / Linux开发
(<http://www.jellythink.com/archives/category/linuxdevelopment/>) 阅读(1092) 评论(4)

那天同事在我的电脑上给我演示一个操作的步骤，使用了一个 `env` 命令，好吧，我承认我文盲，我不知道这个命令是干嘛的！！！正是由于我的无知，反倒激起了学习探索的欲望，一定要把这个 `env` 学个透彻。经过几个下班后的空闲时间，终于搞清楚了这“坨”东西，记录成文，和大家分享，如果大家有所帮助，或者能够扫除大家的一些盲点，我也就胜感欣慰。

这就是你眼中的Shell，当你在Linux中，打开Terminal的时候，出现了一个黑屏，又或一个白屏的窗口的，哦，这就是我们眼中的Shell，这种理解也许对，也许不对。

当我们打开Terminal的时候，其实运行了一个默认的Shell解释器，我们一般都是 `/bin/bash`，当我们在终端中输入各种命令时，都是由这个 `/bin/bash` 来进行解释的；也就是说Shell是运行在Terminal上的一个程序。

明白了Shell以后，我们经常会在当前的Shell中运行一个Shell脚本，当你运行这个Shell脚本的时候，你是否知道这背后在干些什么么？这又说到下一个话题：父Shell与子Shell。

如果不明白父Shell与子Shell的关系，那么这篇文章重点要讲的Shell中的变量，也就说不清了。

当我们启动Terminal的时候，会运行一个Shell进程，暂且叫做Shell进程A；当我们在这个Terminal中运行Shell脚本的时候，进程A会fork出一个新进程，从而启动另一个Shell解释器（这由脚本中第一行指定的，例如：`#!/bin/bash`），fork出来的这个新进程暂且叫做进程B。此时，进程A和进程B就是父、子进程的关系；进程B是一个子Shell，而进程A则是父Shell。一旦子Shell中的脚本执行完毕，此子Shell随即结束，返回到父进程。这就是父Shell和子Shell。

我想我应该把父Shell和子Shell说清了，明白了这层关系以后，我们继续总结。

<http://www.jellythink.com/archives/1323>

首先，你要意识到一点，就是你登陆Linux系统，打开Terminal，使用Shell的时候，系统在背后读取了一大“坨”的配置文件，这些配置文件决定了你的Shell中的变量。所以，在具体总结Shell中的变量时，我们还需要来看看读了哪些配置文件，以及读取这些配置文件的顺序是什么。

1. `/etc/profile` ：该文件为系统的每个用户设置环境信息，当用户第一次登录系统时，该文件会被执行。 并从`/etc/profile.d`目录的配置文件中读取shell的设置；
2. `~/.bash_profile` 或者 `~/.bash_login` 或者 `~/.profile` ：系统会依次寻找这三个文件，这些是针对当前用户的配置，但是需要注意的是，这三个文件一般不会同时存在，即使同时存在，系统按照这个顺序找到了一个以后，就不会再去读剩下的了；
3. `~/.bashrc` ：该文件包含了专属于当前登录用户的bash shell的bash信息，当登录以及每次打开新的shell时，该文件都会被读取；
4. `/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。

这些脚本配置文件决定了你的系统变量和环境变量等。如果你感兴趣，你可以去看看这些脚本的源码，你就会知道它们到底是怎么调用的了。当然了，如果我们需要定义一些我们经常用到的变量，比如配置JDK的时候，你可能还要编辑它们。

Shell中的变量

先不带子Shell玩，把单个Shell进程中的变量捋清了，再把子Shell加进来一起总结。

在Shell中有以下三种变量：

- 内部变量；系统定义，不能修改；
- 环境变量；系统定义，可以修改，可以利用 `export` 将用户变量转为环境变量；
- 用户变量；用户定义，可以定义，玩坏了都没事。

比如以下这些就是内部变量：

| 变量名 | 描述 |
|-------------------|--------------|
| <code>\$#</code> | 命令行参数个数 |
| <code>\$0</code> | 当前程序的名称 |
| <code>\$?</code> | 前一个命令或函数的返回码 |
| <code>\$\$</code> | 当前程序的PID |

以下这些是我们常用的一些环境变量，一般我们使用 `env` 命令查看当前用户的环境变量。

| 变量名 | 描述 |
|--------------------|-----------------------|
| <code>PATH</code> | 表示Shell将到哪些目录中寻找命令或程序 |
| <code>SHELL</code> | 当前用户的Shell类型 |
| <code>HOME</code> | 当前用户主目录 |
| <code>PS1</code> | 基本提示符 |

用户变量（本地变量）那就比较随性了，你可以自己随意定义，比如：

```
> str='Hello World'
> echo $str
```

我们使用 `set` 命令来显示当前Shell中定义的用户变量，当然了 `set` 命令也会输出环境变量。

上面说的这些你明白了么？我们继续。

带上子Shell一起玩

当我们在Shell里面再运行一个Shell脚本的时候，这个时候会fork出一个新的Shell进程，此时就会有父、子Shell两个进程了。有了父、子关系，那么父进程中的Shell变量会遗传到子进程中么？？？同时，子进程中的Shell变量会返回到父进程么？？？这些都是我们需要关注的。先来看一个例子：

父Shell定义一个变量：

```
> str='Hello World'
```

然后在父Shell中运行以下脚本：

```
#!/bin/bash
# 输出父Shell中定义的str
echo $str

# 输出环境变量
echo $HOME
echo $PATH
```

你会发现，运行以后，\$str输出空，但是\$HOME和\$PATH却可以很完美的输出。这也说明，我们在一个Shell中定义的用户变量，只能被当前Shell所使用，别人是无法访问到的，即使是子Shell也不例外；而父进程的环境变量是可以在子进程中被访问。但是有的时候，我们有这样的需求：

在子进程中访问父进程的用户变量（本地变量），这该怎么办？？？

当我们遇到这样的需求时，我就不得不说一下 export 命令了。

说说export命令

export 命令可以将用户变量设置为环境变量，从而可以在子Shell进程中访问该变量。这正好也好和 export 的中文含义相符。对于之前的例子，我们可以在父Shell中输入一下命令：

```
> export str
```

再次执行脚本时，就可以输出用户变量str的值了；这就是 export 的作用。但是，我们在父Shell中输入 export str 以后，当我们关闭父Shell以后，该环境变量将失效，如果想打开Shell就能立刻设置 export ，我们可以按照我们的需要，将 export str 写到上面总结的那一“坨”启动配置文件中，这样就不会因为关闭了父Shell而导致export失效。

父与子的另一层关系

现在我们可以访问父进程的变量了，你是否想过，在子Shell中修改父Shell的变量是否会影响父Shell中该变量的值呢？不妨做个测试。

```
#!/bin/bash
# 修改父Shell传递过来的变量str的值
$str='http://www.jellythink.com'
echo $str
```

运行脚本，发现在父Shell中，str的值并没有发生改变。其实，这又关系到Linux中关于进程的另一个知识点。当父进程中fork一个子进程时，子进程会拷贝父进程的相关变量，此时，子进程就会拥有和父进程同名同值的变量，虽然同名同值，但是却只是父进程的一个副本，对于副本的修改都和父进程无关。关于Linux进程知识，可以参考[这篇文章 \(http://www.jellythink.com/archives/900\)](http://www.jellythink.com/archives/900)。

而如果我们在子Shell中定义一个变量，反过来在父Shell中是否可以访问呢？实践告诉我们，这样是行不通的，你不可在父Shell中访问子Shell中定义的变量。如果想在父Shell中访问子Shell中定义的变量，可以借助一个临时文件，将局部变量写入临时文件，父Shell读取这个文件，从而达到访问子Shell中定义的变量的目的。

总结

这篇文章虽然总结的多而杂，但是每一部分都是息息相关、环环相扣的，对于大家整体理解Linux Shell有非常大的帮助，也有利于大家学习Linux Shell。文章有点长，需要一点耐心去把它从头到尾好好的读完。为了学习，是需要一点耐心的，你说呢？这一篇就总结到这里了，下一篇再见。

果冻想 (<http://www.jellythink.com/>)-一个原创技术文章分享网站。

2015年10月21日 于呼和浩特。

附录

这里对文中涉及到的命令以及一些相关命令进行总结一下。

| 命令 | 描述 |
|-----|----------------|
| set | 显示本地定义的Shell变量 |

| | |
|--------|---------------------|
| unset | 清除环境变量，例如：unset str |
| export | 设置一个新的环境变量 |
| env | 显示当前用户所有环境变量 |

未经允许不得转载：果冻想 (http://www.jellythink.com) » 聊聊Linux Shell (http://www.jellythink.com/archives/1323)


分享到：更多 ()

标签：Linux (http://www.jellythink.com/archives/tag/linux)Linux Shell (http://www.jellythink.com/archives/tag/linux-shell)

相关推荐

- Linux expect详解 (http://www.jellythink.com/archives/1470)
- Linux netstat命令详解 (http://www.jellythink.com/archives/1466)
- 玩玩awk (http://www.jellythink.com/archives/1252)
- awk中的函数 (http://www.jellythink.com/archives/1248)
- 玩玩Sed (http://www.jellythink.com/archives/1145)
- Linux C语言时间操作总结 (http://www.jellythink.com/archives/1073)

评论 4



你的评论可以一针见血

提交评论

昵称


昵称 (必填)

邮箱

邮箱 (必填)

网址

网址



好久不来了，发现内容更新的蛮多的，主题也换了，dux1.3漂亮了很多

#3

eliteYang (http://www.cppfans.org) 7个月前 (10-22)



这些内容太初级了……随便 man env 即可。

#2

御宅暴君 (http://acgtyrant.com) 7个月前 (10-22)



有些内容真不是man手册能提供的。

果冻想 (http://www.jellythink.com) 7个月前 (10-23)



不错

#1

jimmyyang 2周前 (05-16)

在这里玩技术，享受技术带来的疯狂

捐赠名单 (http://www.jellythink.com/donations/)

关于果冻 (http://www.jellythink.com/about/)

