**0、前言**

项目环境部署是自己的薄弱项，一些简单的服务，照着搜索来的步骤能凑合部署成功，但并不明白其中的原理，偶尔遇到一些ERR，就抓瞎，无从下手。项目环境部署，需要一些基础服务，如java环境，tomcat环境、mysql等等；服务部署成功后，还需要给其配置环境变量；要配置环境变量，就要用到什么bashrc或bash\_profile文件，可是这文件有啥区别，一直不明白，今天看到这篇文章，结合自己实践，分享给各位。

**1、bash的startup文件**

Linux shell是用户与Linux系统进行交互的媒介，而bash作为目前Linux系统中最常用的shell，它支持的startup文件也并不单一，甚至让人感到费解。本文以CentOS 7系统为例，对bash的startup文件进行一些必要的梳理和总结。  
（1）先来看看bash手册上的描述

/etc/profile  
The systemwide initialization file, executed for login shells。系统初始化文件，在login shells时执行  
/etc/bash.bash\_logout  
The systemwide login shell cleanup file, executed when a login shell exits。系统的登录shell清理文件，当一个登录shell退出时执行。  
~/.bash\_profile  
The personal initialization file, executed for login shells。个人初始化文件，为登录shell执行。  
~/.bashrc  
The individual per-interactive-shell startup file。每个交互式shell启动文件。  
~/.bash\_logout  
The individual login shell cleanup file, executed when a login shell exits。单个登录shell清理文件，当一个登录shell退出时执行。

此外，bash还支持~/.bash\_login和~/.profile文件，作为对其他shell的兼容，他们与~/.bash\_profile文件的作用是相同的。

备注：Debian系统会使用~/.profile文件取代~/.bash\_profile文件，相关细节上也会和Cent OS略有不同。

**2、“profile”与“rc”系列**

通过名字的不同，我们可以直观地将startup文件分为“profile”与“rc”两个系列，其实他们的功能都很类似，但是使用的场景不同，这也是大家最容易忽略的地方。

所谓的不同场景，其实就是shell的运行模式。我们知道运行中的bash有“交互”和“登陆”两种属性，而执行“profile”系列还是“rc”系列，就与shell的这两个属性有关。  
原理上讲，“登陆shell”启动时会加载“profile”系列的startup文件，而“交互式非登陆shell”启动时会加载“rc”系列的startup文件。

**3、“profile”系列的执行场景**

根据bash手册上的描述：

When bash is invoked as an interactive login shell, or as a non-interactive shell with the –login option, it first reads and executes commands from the file /etc/profile, if that file exists. After reading that file, it looks for ~/.bash\_profile, ~/.bash\_login, and ~/.profile, in that order, and reads and executes commands from the first one that exists and is readable. The –noprofile option may be used when the shell is started to inhibit this behavior.  
When a login shell exits, bash reads and executes commands from the files ~/.bash\_logout and /etc/bash.bash\_logout, if the files exists.

翻译成中文为：

当bash被作为交互式登录shell调用时，或者作为一个非交互式shell使–login选项时，如果该文件存在，它将首先从文件/etc/profile中读取和执行命令。读了这个文件后，它查找~/.bash\_profile、~/.bash\_login和~/.profile，在该顺序中，并从存在且可读的第一个命令中读取和执行命令。shell可以使用–noprofile选项禁止该行为。  
当一个登录shell退出时，bash将从文件中读取和执行命令~/.bash\_logout和/etc/bash.bash\_logout，如果文件存在。

“profile”系列的代表文件为~/.bash\_profile，它用于“登录shell”的环境加载，这个“登录shell”既可以是“交互式”的，也可以是“非交互式”的。  
通过–noprofile选项可以阻止系统加载“profile”系列的startup文件。

**4、对于交互式的登陆shell而言，CentOS规定了startup文件的加载顺序如下：**

登陆过程：  
1. 读取并执行/etc/profile文件；  
2. 读取并执行~/.bash\_profile文件；  
- 若文件不存在，则读取并执行~/.bash\_login文件；  
- 若文件不存在，则读取并执行~/.profile文件；

登出过程：  
1. 读取并执行~/.bash\_logout文件；  
2. 读取并执行/etc/bash.bash\_logout文件；

为了完成实验，我新建了一些系统默认没有提供的startup文件，例如/etc/bash.bash\_logout。然后在每个文件中打印了文件名，并将它们之间的显式调用语句注释掉，例如~/.bash\_profile对~/.bashrc的显式调用。

“交互式登陆shell”的实验结果如下：

[root@localhost ~]# su - chen

Last login: Tue Apr 18 17:15:08 CST 2017 from 192.168.161.1 on pts/2

execute /etc/profile

execute ~/.bash\_profile

-bash-4.2$ exit

logout

execute ~/.bash\_logout

execute /etc/bash.bash\_logout

[root@localhost ~]#

我们看到，因为执行了~/.bash\_profile文件，所以优先级更低的~/.bash\_login和~/.profile文件并没有被执行。  
我们可以删除~/.bash\_profile和~/.bash\_login文件，这样系统就会找到并执行~/.profile文件：

[root@localhost ~]*# mv /home/chen/.bash\_profile /home/chen/.bash\_profile.bak*

[root@localhost ~]*# mv /home/chen/.bash\_login /home/chen/.bash\_login.bak*

[root@localhost ~]*# su - chen*

Last login: Tue Apr 18 17:27:21 CST 2017 on pts/1

execute /etc/profile

execute ~/.profile

-bash-4.2$ exit

logout

execute ~/.bash\_logout

execute /etc/bash.bash\_logout

[root@localhost ~]*#*

**5、非交互式登陆shell**

对于非交互式的登陆shell而言，CentOS规定了startup文件的加载顺序如下：

登陆过程：  
1. 读取并执行/etc/profile文件；  
2. 读取并执行~/.bash\_profile文件；  
- 若文件不存在，则读取并执行~/.bash\_login文件；  
- 若文件不存在，则读取并执行~/.profile文件；

我们注意到，与“交互式登陆shell”相比，“非交互式登陆shell”并没有登出的过程，实验也证实了这一点：

-bash-4.2$ bash --login -c "uname -r"

execute /etc/profile

execute ~/.bash\_profile

3.10.0-514.el7.x86\_64

-bash-4.2$ *# 此时非交互式shell已退出*

**6、“rc”系列的执行场景**

根据bash手册上的描述：

When an interactive shell that is not a login shell is started, bash reads and executes commands from ~/.bashrc, if that file exists. This may be inhibited by using the –norc option. The –rcfile file option will force bash to read and execute commands from file instead of ~/.bashrc.

“rc”系列的代表文件为~/.bashrc，它用于“交互式非登录shell”的环境加载。

通过–norc选项可以阻止系统加载“rc”系列的startup文件；通过–rcfile选项可以使用指定的文件替代系统默认的~/.bashrc文件。

**7、交互式非登陆shell**

对于交互式的非登陆shell而言，CentOS规定了startup文件的加载顺序如下：  
1. 读取并执行~/.bashrc或–rcfile选项指定的文件

这里需要说明，其实“rc”系列startup文件还包括/etc/bashrc。但是系统并不直接调用这个文件，而是通过~/.bashrc文件显式地调用它。

为了完成实验，我在每个startup文件中打印了文件名，并将它们之间的显式调用语句注释掉，例如~/.bashrc对/etc/bashrc的显式调用。

“交互式非登陆shell”的实验结果如下：

[root@localhost ~]*# su chen*

execute ~/.bashrc

bash-4.2$ exit

exit

[root@localhost ~]*#*

**8、startup文件的默认调用关系**

细心的用户会发现，startup文件的加载并不像上面所述的那样简单。这是因为在CentOS中，startup文件之间还存在着默认的显式调用关系，它们是：  
1. ~/.bash\_profile显式调用~/.bashrc文件；  
2. ~/.bashrc显式调用/etc/bashrc文件；

再看startup文件  
分别打开/etc/profile和/etc/bashrc两个文件，我们可以看到：

[root@localhost ~]# head /etc/profile

# /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.

[root@localhost ~]# head /etc/bashrc

# /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.

由此可见，“profile”系列文件的主要目的在于为“登录shell”设置环境变量和启动程序；而“rc”系列文件的主要目的在于设置功能和别名。

顺便提一句，Linux中“rc”是英文“run command”的缩写，表示文件中存放需要执行的命令。其实这也非常符合逻辑，设置功能就要执行shopt命令，而设置别名要执行alias命令。与“rc”系列互补，“profile”系列用来设置环境变量，它不会去调用这两个命令，但却经常需要使用export语句。不信你可以看一看这两个文件。

另外值得一提的是，这两个文件同时提到了一个位置：/etc/profile.d目录。这个目录用于存放个性化配置脚本，你可以把自己需要的全局配置放入以.sh结尾的文件中，系统在执行/etc/profile和/etc/bashrc文件时，都会择机调用它们。这样做最大的好处是便于维护，而且相对更加安全。

这些文件的编写方法，可以参考目录下已有的文件：

[root@localhost ~]# ls /etc/profile.d/\*.sh

/etc/profile.d/256term.sh /etc/profile.d/colorls.sh /etc/profile.d/less.sh

/etc/profile.d/colorgrep.sh /etc/profile.d/lang.sh /etc/profile.d/which2.sh

**9、总结**

对于“登录shell”而言，“交互式”执行“登陆”和“登出”相关的“profile”系列startup文件，“非交互式”只执行“登陆”相关的“profile”系列startup文件；对于“非登陆shell”而言，“交互式”执行“rc”系列的startup文件，而“非交互式”执行的配置文件由环境变量BASH\_ENV指定。

Linux中startup文件区分全局和个人：全局startup文件放在/etc目录下，用于设置所有用户共同的配置，除非你清楚地知道你在做的事情，否则不要轻易改动它们；个人startup文件放在~目录下，用于设置某个用户的个性化配置。

~/.bash\_profile会显式调用~/.bashrc文件，而~/.bashrc又会显式调用/etc/bashrc文件，这是为了让所有交互式界面看起来一样。无论你是从远程登录（登陆shell），还是从图形界面打开终端（非登陆shell），你都拥有相同的提示符，因为环境变量PS1在/etc/bashrc文件中被统一设置过。

下面我来对startup文件进行一个完整的总结：  
  
备注：  
1. “直接执行”表示此文件被系统直接调用，它的执行是无条件的；  
2. “条件执行”表示此文件被系统调用是有先决条件的（没有优先级更高的文件可用）；  
3. “引用执行”表示此文件不是被系统直接调用的，而是被其他文件显式调用的；  
4. 后面的数字表示文件被调用的顺序，数字越大调用越靠后；  
5. “非交互非登陆”shell的配置文件可以由BASH\_ENV环境变量指定；

最后我想说的是，知道startup文件何时被执行并不是关键，关键是要理解自己的情况应该去修改哪个startup文件。

如果你想对bash的功能进行设置或者是定义一些别名，推荐你修改~/.bashrc文件，这样无论你以何种方式打开shell，你的配置都会生效。而如果你要更改一些环境变量，推荐你修改~/.bash\_profile文件，因为考虑到shell的继承特性，这些更改确实只应该被执行一次（而不是多次）。针对所有用户进行全局设置，推荐你在/etc/profile.d目录下添加以.sh结尾的文件，而不是去修改全局startup文件。