Linux系统中的Shell环境配置

目录

目录

为什么需要Shell的配置文件?

```
[root@test ~]# export NAME="tester"
[root@test ~]# echo $NAME
tester
[root@test ~]# unset NAME
[root@test ~]# echo $NAME

[root@test ~]#
```

- 在讲解Shell配置文件之前，我们先来回顾一下之前讲解的自定义环境变量。定义一个环境变量需要使用export命令，例如左图：将NAME定义为环境变量。这里为了与环境变量的命名风格保持一致，变量名称将全部采用大写字母表示。将NAME定义为环境变量之后，在当前Shell的子Shell中就可以直接使用这个变量了。如果要撤销我们所自定义的环境变量，那么可以使用unset命令。例如，撤销自定义环境变量NAME。unset命令同样也可以用于撤销用户自定义变量。

- 通过刚才这种方式所定义的环境变量，其作用范围也仅限于当前 Shell及其子Shell，如果我们切换到另一个终端，仍然无法使用当前设置的变量。而且，即使在当前Shell中切换到另一个用户的身份，也是无法使用这个自定义的环境变量的。另外，通过这种方式定义的环境变量是临时性的， 当用户退出登录或者系统重启之后， 自定义的变量都会失效。**如何才能定义一个对于所有终端或者所有用户永久有效的环境变量呢？这就需要修改Shell的配置文件。**

- **这里以常用的Bash为例，Linux环境下的Bash配置文件如下：**

/etc/profile	/etc/bashrc	~/.bash_profile	~/.bashrc	~/.bash_logout

详解Bash配置文件

- **/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的目录中搜集shell的设置。所以，如果你对/etc/profile有修改，需要重启才能生效，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。如果你想对所有使用bash的用户修改某个配置并在以后打开的bash都生效，可以修改这个文件，重新打开一个bash即可生效。

- **~/.bash_profile**

每一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，它设置一些环境变量，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。

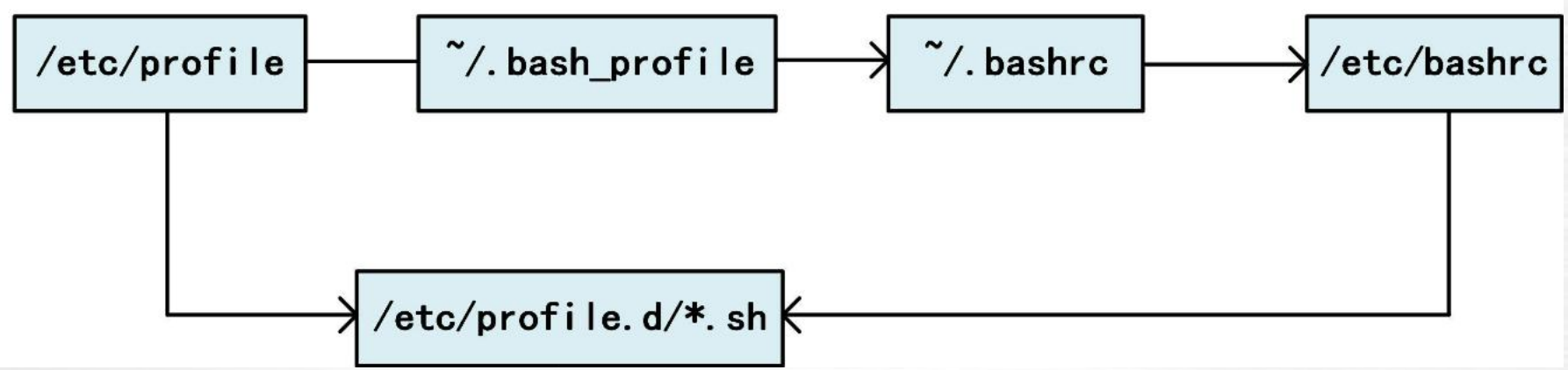
- **~/.bash_logout**

当每次退出系统时，执行该文件。

Bash配置文件 – 读取顺序

Shell配置文件分为系统级别的配置文件和用户级别的配置文件。任何一种Shell都有用户级别的配置文件，以及对应的系统级别的配置文件。

- **系统级别的配置文件位于/etc/下，这些配置会应用于所有用户：/etc/profile, /etc/bashrc。**
- **用户级别的配置文件位于用户目录~下，通常会加一个.来隐藏：~/.bashrc。**
- **在shell启动时，会首先执行系统级别的配置文件，再执行用户级别的配置文件。也就是说~/.bashrc中的配置会覆盖/etc/bashrc中的配置。**



- **profile: 统一设置系统范围的环境和启动程序，只在用户登录时执行一次。**
- **bashrc: 是专门给bash做初始化设置用的，不但在用户登录时会执行，而且每当用户打开新的shell或者创建子shell时也会执行。**

Bash配置文件 – 实例演示1

➢ profile类文件

首先，我们仍以之前的定义环境变量为例。我们希望可以将NAME定义为一个在所有终端上对所有用户都永久有效的环境变量，那么就可以修改/etc/profile文件，系统中很多内置的环境变量就是由这个文件来定义的。打开该文件之后，可以发现这其实是一个Shell 脚本文件，里面有大量的代码，我们将光标移到文档末尾，在文档最后插入一行语句来定义环境变量。由于在Bash配置文件中定义的设置不会立即生效，因此我们可以使“source”或 “.”命令在当前Shell中重新加载配置文件，使其生效。

```
[root@test ~]# echo "export NAME=tester" >> /etc/profile
[root@test ~]# source /etc/profile
[root@test ~]#
```

```
[root@test ~]# echo $NAME
tester
[root@test ~]#
```

然后，打开一个新的终端，并以root用户身份登录，可以发现NAME变量仍然有效。而且即使切换到其它用户，这个变量也仍然有效。修改/etc/profile文件之所以可以达到这样的效果，是因为系统中的所有用户在登录shell时都会自动加载并执行这个文件中的设置。另外，除修改/etc/profile文件之外，我们也可以在/etc/profile.d目录中新建一个后缀为".sh"的脚本文件，并在这个文件中写入要自定义的环境变量，这可以达到相同的效果。

```
[root@test ~]# echo "export NAME=tester" >> /etc/profile.d/name.sh
[root@test ~]# source /etc/profile.d/name.sh
[root@test ~]# echo $NAME
tester
[root@test ~]#
```

更推荐后一种方式，将一个大配置文件中的设置项按照类别提取出来，并分散存储在各个小配置文件中，这种模块化的设计思路备受推崇。

Bash配置文件 – 实例演示2

➢ bashrc类文件

我们就了解了 profile 类配置文件，那么 bashrc类文件又是做什么的呢？其实这两类文件的作用是类似的，都可以用于定义用户在登录 Shell 时要自动执行的操作。它们的区别：profile 类配置文件只在用户登录 Shell 时会被加载执行；而 bashrc 类配置文件除在用户登录 Shell 时会被加载执行之外，每当用户打开新的 Shell 或者子 Shell 时，都会被加载执行。

```
[root@test ~]# echo "echo 'hello world'" >> /etc/profile
[root@test ~]# echo "echo '你好！ '" >> /etc/bashrc
[root@test ~]# su - test
Last login: Tue Feb 14 14:45:51 CST 2023 on pts/0
hello world
你好！
[test@test ~]$ bash
你好！
[test@test ~]$
```

```
Last login: Tue Feb 14 14:29:02 2023 from 192.168.95.1
hello world
你好！
[root@test ~]#
```

在理解所有这些Bash配置文件的特性之后，我们可以发现它们的用途还是非常广泛的，**通过修改这些配置文件，就可以指定用户在登录Shell时自动执行某些操作。**

目录

命令别名 - alias

➢ 命令别名 – alias 作用是简化命令输入

命令别名即把一个命令名称定义成另一个名称，在使用时，可以使用命令本身，也可以使用命令的别名。

定义命令别名时shell的特性，只在当前终端生效，当用户退出当前终端时，所定义的别名就会失效。就算是同一个用户再次打开一个shell，其也不会生效，也就是说，在shell中定义的命令别名仅仅在当前shell生命周期中有效。别名的有效范围为当前shell进程，如果需要永久生效，则需要通过修改bash相关的配置文件来实现。

当前用户生效的命令别名，仅在当前用户下使用，全局生效的命令别名则在系统所有用户下都可使用。

■ 定义命令别名 – alias

✓ 定义命令别名语法：

```
alias 自定义别名=<命令>
```

此方式为临时定义命令别名。

```
[root@test ~]# alias cdmnt='cd /mnt'
[root@test ~]# cdmnt
[root@test mnt]# pwd
/mnt
[root@test mnt]#
```

```
[root@test ~]# alias mountiso='mount -o loop /root/CentOS-7-x86_64-DVD-1810-v7.iso /mnt'
[root@test ~]# mountiso
mount: /dev/loop0 is write-protected, mounting read-only
[root@test ~]# ls /mnt
boot.cat         EFI    GPL     isolinux  Packages  RPM-GPG-KEY-CentOS-7          TRANS.TBL
CentOS_BuildTag  EULA   images  LiveOS    repodata  RPM-GPG-KEY-CentOS-Testing-7
[root@test ~]#
```

命令别名 - alias

■ 删除命令别名 – unalias

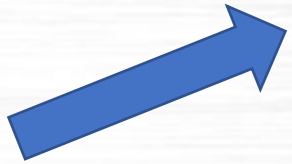
✓ 删除命令别名语法：

```
unalias <命令别名>
```

```
[root@test ~]# unalias cdmnt
[root@test ~]# cdmnt
bash: cdmnt: command not found...
[root@test ~]# unalias mountiso
[root@test ~]# mountiso
bash: mountiso: command not found...
[root@test ~]#
```

■ 设置命令别名永久生效（分为两种生效范围）

1. 设置当前用户命令别名永久生效
– 仅仅当前用户使用有效。



```
[root@test ~]# vim /root/.bashrc
[root@test ~]# cat /root/.bashrc
# .bashrc

# User specific aliases and functions

alias rm='rm -i'
alias cp='cp -i'
alias mv='mv -i'

# Source global definitions
if [ -f /etc/bashrc ]; then
        . /etc/bashrc
fi
alias mountiso='mount -o loop /root/CentOS-7-x86_64-DVD-1810-v7.iso /mnt'
alias cdmnt='cd /mnt'
[root@test ~]#
```

```
[root@test ~]# source /root/.bashrc
你好！
[root@test ~]# ls /mnt
[root@test ~]# mountiso
mount: /dev/loop0 is write-protected, mounting read-only
[root@test ~]# ls /mnt
boot.cat         EFI   GPL     isolinux  Packages  RPM-GPG-KEY-CentOS-7
CentOS_BuildTag  EULA  images  LiveOS    repodata  RPM-GPG-KEY-CentOS-Testing-7
[root@test ~]#
```

命令别名 - alias

2. 设置全局使用命令别名永久生效 – 系统所有用户使用有效。

```
[root@test ~]# vim /etc/bashrc
[root@test ~]# cat /etc/bashrc |grep alias
# System wide functions and aliases
alias mountiso='mount -o loop /root/CentOS-7-x86_64-DVD-1810-v7.iso /mnt'
alias cdmnt='cd /mnt'
[root@test ~]# source /etc/bashrc
你好！
[root@test ~]# su - test
Last login: Tue Feb 14 16:14:11 CST 2023 on pts/1
hello world
你好！
[test@test ~]$ cdmnt
[test@test mnt]$ pwd
/mnt
[test@test mnt]$ mountiso
mount: only root can use "--options" option
[test@test mnt]$
```

```
[root@test ~]# su - test1
Last login: Tue Feb 14 16:39:44 CST 2023 on pts/1
hello world
你好！
[test1@test ~]$ cdmnt
[test1@test mnt]$ pwd
/mnt
[test1@test mnt]$ mountiso
mount: only root can use "--options" option
[test1@test mnt]$
```

备注：左边两幅图已经证明，系统中不同的用户，都可以直接使用全局命令别名进行操作。

命令别名 - alias

- **查看所有的命令别名 – 直接输入alias，回车就可以看到该用户下的定义的所有命令别名。**

```
[root@test ~]# alias
alias cdmnt='cd /mnt'
alias cp='cp -i'
alias egrep='egrep --color=auto'
alias fgrep='fgrep --color=auto'
alias gcc='gcc -std=gnu99'
alias grep='grep --color=auto'
alias l.='ls -d .* --color=auto'
alias ll='ls -l --color=auto'
alias ls='ls --color=auto'
alias mountiso='mount -o loop /root/CentOS-7-x86_64-DVD-1810-v7.iso /mnt'
alias mv='mv -i'
alias perlll='eval `perl -Mlocal::lib`'
alias rm='rm -i'
alias which='alias | /usr/bin/which --tty-only --read-alias --show-dot --show-tilde'
[root@test ~]#
```

```
[root@test ~]# su - test
Last login: Tue Feb 14 17:09:10 CST 2023 on pts/1
hello world
你好！
[test@test ~]$ alias
alias cdmnt='cd /mnt'
alias egrep='egrep --color=auto'
alias fgrep='fgrep --color=auto'
alias gcc='gcc -std=gnu99'
alias grep='grep --color=auto'
alias l.='ls -d .* --color=auto'
alias ll='ls -l --color=auto'
alias ls='ls --color=auto'
alias mountiso='mount -o loop /root/CentOS-7-x86_64-DVD-1810-v7.iso /mnt'
alias perlll='eval `perl -Mlocal::lib`'
alias vi='vim'
alias which='alias | /usr/bin/which --tty-only --read-alias --show-dot --show-tilde'
[test@test ~]$
```

```
[root@test ~]# su - test1
Last login: Tue Feb 14 17:14:39 CST 2023 on pts/1
hello world
你好！
[test1@test ~]$ alias
alias cdmnt='cd /mnt'
alias egrep='egrep --color=auto'
alias fgrep='fgrep --color=auto'
alias gcc='gcc -std=gnu99'
alias grep='grep --color=auto'
alias l.='ls -d .* --color=auto'
alias ll='ls -l --color=auto'
alias ls='ls --color=auto'
alias mountiso='mount -o loop /root/CentOS-7-x86_64-DVD-1810-v7.iso /mnt'
alias mycd='cd /etc'
alias perlll='eval `perl -Mlocal::lib`'
alias vi='vim'
alias which='alias | /usr/bin/which --tty-only --read-alias --show-dot --show-tilde'
[test1@test ~]$ mycd
[test1@test etc]$ pwd
/etc
[test1@test etc]$
```

问题：怎么定义不同用户下的不同命令别名呢?

谢 谢