16. shell基础（上）

笔记本： 优秀笔记
创建时间： 2018/12/26 11:08 **更新时间：** 2018/12/29 15:52
作者： 306798658@qq.com

一、Shell介绍

- Shell是一个命令解释器，提供用户和机器之间的交互，你敲命令，它出结果。
- Shell支持特定语法，比如逻辑判断、循环；
- 每个用户都可以有自己特定的shell；
- CentOS7默认shell为bash (Bourne Agin Shell)；
- 还有zsh、ksh等。

二、命令历史

我们使用过的命令，即命令历史，会存放在用户家目录下/root/.bash_history。history命令,可以查看使用过的命令历史，这个命令历史数量限制为1000条。

```
[root@aming01 ~]# echo $HISTSIZE
1000
```

history -c命令，可以将当前内存的命令历史清空，但是不会清空/root/.bash_history里面的命令

```
  996  ls -l /var/cache/yum/x86_64/7/updates/packages/
  997  echo $HISTSIZE
  998  history
[root@aming01 ~]# history -c
[root@aming01 ~]# history
    1  history
```

注意：刚刚敲过的命令，只有当用户正常退出终端时，才会保存到.bash_history文件中去。

环境变量HISTSIZE在/etc/profile中定义，可以vi /etc/profile，找到HISTSIZE并修改参数，然后执行命令# source /etc/profile，使修改马上生效

```
[root@aming01 ~]# echo $HISTSIZE
1000
[root@aming01 ~]# source /etc/profile
[root@aming01 ~]# echo $HISTSIZE
2000
```

修改命令历史格式，可以更加方便我们能直观的查看

```
HISTTIMEFORMAT=”%Y/%m/%d %H:%M:%S”
[root@aming01 ~]# history
    1  history
    2* v
    3  echo %HISTSIZE
    4  echo $HISTSIZE
    5  source /etc/profile
    6  echo $HISTSIZE
    7  history
[root@aming01 ~]# HISTTIMEFORMAT="%Y/%m/%d %H:%M:%S"
[root@aming01 ~]# echo $HISTTIMEFORMAT
%Y/%m/%d %H:%M:%S
[root@aming01 ~]# history
    1  2018/06/01 17:25:40history
    2* 2018/06/01 17:26:08v
    3  2018/06/01 17:26:58echo %HISTSIZE
```

```
    4  2018/06/01 17:27:13echo $HISTSIZE
    5  2018/06/01 17:27:24source /etc/profile
    6  2018/06/01 17:27:25echo $HISTSIZE
    7  2018/06/01 17:29:14history
    8  2018/06/01 17:29:22HISTTIMEFORMAT="%Y/%m/%d %H:%M:%S"
    9  2018/06/01 17:29:35echo $HISTTIMEFORMAT
   10  2018/06/01 17:29:38history
```

该格式只适用于当前终端，若想环境变量生效，需要# vi /etc/profile，在配置文件中添加

```
HISTTIMEFORMAT=”%Y/%m/%d %H:%M:%S”，然后source /etc/profile，打开其他终端，也会生效
HOSTNAME=`/usr/bin/hostname 2>/dev/null`
HISTSIZE=2000
HISTTIMEFORMAT="%Y/%m/%d %H:%M:%S"

if [ "$HISTCONTROL" = "ignorespace" ] ; then
    export HISTCONTROL=ignoreboth
else
```

想命令历史永久保存并不被别人删除，运行chattr +a ~/.bash_history（只能追加，不能删除）

```
[root@aming01 ~]# lsattr .bash_history
---------------- .bash_history
[root@aming01 ~]# chattr +a ./.bash_history
[root@aming01 ~]# lsattr .bash_history
-----a---------- .bash_history
```

显示历史命令执行时间：
设置环境变量：HISTTIMEFORMAT= “%Y/%m/%d %H:%M:%S” 可以显示历史命令的执行时间。

```
[root@aming01 ~]# HISTTIMEFORMAT="%Y/%m/%d %H:%M:%S"
[root@aming01 ~]# history
    1  2018/06/05 22:07:59history
    2  2018/06/05 22:08:10history -d2
    3  2018/06/05 22:08:12history
    4  2018/06/05 22:09:11cat ~/.bash_history
    5  2018/06/05 22:40:46ehco histsiz
```

当然前面的设置只是当前终端生效，如果重新登录或者重启系统就会失效，可以将命令写入~/.bash_profile或者/etc/profile(全局登录生效)文件中来永久生效。

```
[root@aming01 ~]#  echo "HISTTIMEFORMAT='%Y/%m/%d %H:%M:%S'" >> ~/.bash_profile
```

!! 表示执行最后一条命令

```
[root@aming01 ~]# ls
666  anaconda-ks.cfg
[root@aming01 ~]# !!
ls
666  anaconda-ks.cfg
```

!n 表示运行第几条命令（n代表数字）

```
[root@aming01 ~]# history
    1  2018/06/01 17:25:40history
    2* 2018/06/01 17:26:08v
    3  2018/06/01 17:26:58echo %HISTSIZE
    4  2018/06/01 17:27:13echo $HISTSIZE
    5  2018/06/01 17:27:24source /etc/profile
    6  2018/06/01 17:27:25echo $HISTSIZE
    7  2018/06/01 17:29:14history
    8  2018/06/01 17:29:22HISTTIMEFORMAT="%Y/%m/%d %H:%M:%S"
    9  2018/06/01 17:29:35echo $HISTTIMEFORMAT
```

```
   10  2018/06/01 17:29:38history
   11  2018/06/01 17:30:23vim /etc/profile
   12  2018/06/01 17:30:52source /etc/profile
   13  2018/06/01 17:31:07vim /etc/profile
   14  2018/06/01 17:31:57lsattr .bash_history
   15  2018/06/01 17:32:10chattr +a ./.bash_history
   16  2018/06/01 17:32:12lsattr .bash_history
   17  2018/06/01 17:32:29ls
   18  2018/06/01 17:32:54history
[root@aming01 ~]# !17
ls
666  anaconda-ks.cfg
```

!echo 表示会在命令历史里面，最近一次执行以 echo 开头的命令

```
[root@aming01 ~]# !echo
echo $HISTTIMEFORMAT
%Y/%m/%d %H:%M:%S
```

三、命令补全和别名

按一次tab可以补全一个命令、一个路径或者是一个文件名；连续按两次tab键，则把所有的命令或者文件名都列出来。在centos7里支持参数自动补全，需要安装安装包bash-completion；然后重启下系统（reboot），才会生效

```
[root@aming01 ~]# yum install -y bash-completion
已加载插件：fastestmirror
Loading mirror speeds from cached hostfile
```

我们可以通过alias把一个常用且很长的指令另取名一个简单易记的指令，如果不想用了，也可以使用unalias取消别名，直接执行alias命令，则会看到系统中所有的别名。

```
[root@aming01 ~]# alias
alias cp='cp -i'
alias egrep='egrep --color=auto'
alias fgrep='fgrep --color=auto'
alias grep='grep --color=auto'
alias l.='ls -d .* --color=auto'
alias ll='ls -l --color=auto'
alias ls='ls --color=auto'
alias mv='mv -i'
alias rm='rm -i'
alias which='alias | /usr/bin/which --tty-only --read-alias --show-dot --show-tilde'
```

alias存放：

第一个在用户家目录下.bashrc文件下
第二个在 /etc/profile.d 目录下的 colorgrep.sh 、colorls.sh 这些脚本中定义的

使用unalias可以取消别名。命令格式： unalias alias_name

四、通配符

通配符*，表示零个或多个字符

```
[root@aming01 ~]# ls *.txt
1.txt  2.txt
[root@aming01 ~]# ls txt*
txt.1
[root@aming01 ~]# ls *txt*
1.txt  2.txt  txt.1
```

通配符？，表示任意一个字符

```
[root@aming01 ~]# ls *txt
1.txt  2.txt  aa.txt
[root@aming01 ~]# ls ?.txt
1.txt  2.txt
```

中括号 []，ls [0-9].txt表示0-9范围内的任意.txt文件

```
[root@localhost ~]# ls
1.txt  2.txt  4.txt  aa.txt  anaconda-ks.cfg  a.txt  b.txt  txt.1
[root@localhost ~]# ls [0-4].txt
1.txt  2.txt  4.txt
[root@localhost ~]# ls [0-4a-bA-Z].txt
1.txt  2.txt  4.txt  a.txt  b.txt
[root@localhost ~]#
[root@localhost ~]# ls [12a].txt
1.txt  2.txt  a.txt
[root@localhost ~]#
```

花括号 {}，ls {1,2,3}.txt表示括号内任意.txt文件

```
[root@localhost ~]# ls
1.txt  2.txt  4.txt  aa.txt  anaconda-ks.cfg  a.txt  b.txt  txt.1
[root@localhost ~]# ls {1,2}.txt
1.txt  2.txt
[root@localhost ~]# ls {1,2,b}.txt
1.txt  2.txt  b.txt
[root@localhost ~]# ls {1,2,3,4,b}.txt
ls: 无法访问3.txt: 没有那个文件或目录
1.txt  2.txt  4.txt  b.txt
[root@localhost ~]#
```

五、输入输出重定向

cat 1.txt > 2.txt，大于号>表示将前面的命令输出，直接输入到后面的文件里面去，这里是指将1.txt的内容重定向到2.txt里面去，之前2.txt的内容会被删除。

```
[root@aming01 ~]# echo "111">>1.txt
[root@aming01 ~]# cat 1.txt
111
[root@aming01 ~]# echo "11111111">>2.txt
[root@aming01 ~]# cat 2.txt
11111111
[root@aming01 ~]# cat 1.txt>2.txt
[root@aming01 ~]# cat 2.txt
111
```

cat 1.txt >> 2.txt，两个大于号>>表示追加，不会删除2.txt的内容，将1.txt的内容追加到2.txt里面去。

```
[root@aming01 ~]# cat 1.txt>>2.txt
[root@aming01 ~]# cat 2.txt
111
111
```

ls cdjdj 2> a.txt，2大于号表示将命令产生的错误信息输入到一个文件里去

```
[root@localhost ~]# cdjdj 2> a.txt
[root@localhost ~]# cat a.txt
-bash: cdjdj: 未找到命令
[root@localhost ~]# asdd
-bash: asdd: 未找到命令
[root@localhost ~]# asdd 2> a.txt
[root@localhost ~]# cat a.txt
-bash: asdd: 未找到命令
[root@localhost ~]#
```

ls afhvjdh 2>>a.txt，表示错误信息追加重定向

```
[root@localhost ~]# cat a.txt
-bash: asdd: 未找到命令
[root@localhost ~]# fhvjdh 2>> a.txt
[root@localhost ~]# cat a.txt
-bash: asdd: 未找到命令
-bash: fhvjdh: 未找到命令
[root@localhost ~]#
```

ls {1,2}.txt bb.txt &> a.txt 2>b.txt， 表示将正确和错误的输出信息都输入到a.txt中

```
[root@localhost ~]# ls
1.txt  2.txt  4.txt  aa.txt  anaconda-ks.cfg  a.txt  b.txt  txt.1
[root@localhost ~]# ls {1,2}.txt bb.txt &> a.txt
[root@localhost ~]# cat a.txt
ls: 无法访问bb.txt: 没有那个文件或目录
1.txt
2.txt
[root@localhost ~]#
```

ls {1,2}.txt bb.txt &>> a.txt ,表示将正确和错误的输出信息都追加到a.txt中

```
[root@localhost ~]# ls {1,2}.txt bb.txt &>> a.txt
[root@localhost ~]# cat a.txt
1.txt
2.txt
ls: 无法访问bb.txt: 没有那个文件或目录
1.txt
2.txt
[root@localhost ~]#
```

ls {1,2}.txt bb.txt > a.txt 2>b.txt;表示将正确的输出信息都输入到a.txt中；错误的输出信息都输入到b.txt中；

```
[root@localhost ~]# ls
1.txt  2.txt  4.txt  aa.txt  anaconda-ks.cfg  a.txt  b.txt  txt.1
[root@localhost ~]# ls {1,2}.txt bb.txt > a.txt 2>b.txt
[root@localhost ~]# cat a.txt
1.txt
2.txt
[root@localhost ~]# cat b.txt
ls: 无法访问bb.txt: 没有那个文件或目录
[root@localhost ~]#
```

wc -l < 1.txt ， 表示把1.txt文件内容输入重定向到命令wc -l 中去， （左边必须要是命令，不能由文件到文件，很少用，了解即可）

```
[root@localhost ~]# wc -l < 1.txt
1
[root@localhost ~]# 2.txt < 1.txt
-bash: 2.txt: 未找到命令
[root@localhost ~]#
```

六、管道符和作业控制

管道符|，用于将前一个指令的输出作为后一个指令的输入

```
cat 1.txt |wc -l ;cat 1.txt |grep ‘aaa’
```

命令wc -l： 统计文件数量

命令grep： 是用来过滤指定关键词的命令，只要文件中含有关键词，就会把这一行过滤出来

作业控制

Ctrl+z 暂停一个正在执行的任务

```
[root@localhost ~]# vi 10.txt.bak

[1]+  已停止               vi 10.txt.bak
[root@localhost ~]#
```

然后使用fg（foreground）命令恢复

```
[1]+  已停止               vi 10.txt.bak
[root@localhost ~]# fg
vi 10.txt.bak
[root@localhost ~]#
```

如果暂停多个任务，可以用jobs查看在后台运行的任务

```
[root@localhost ~]# vi 1.txt

[1]+  已停止               vi 1.txt
[root@localhost ~]# jobs
[1]+  已停止               vi 1.txt
[root@localhost ~]# vi 10.txt.bak

[2]+  已停止               vi 10.txt.bak
[root@localhost ~]# jobs
[1]-  已停止               vi 1.txt
[2]+  已停止               vi 10.txt.bak
[root@localhost ~]#
```

如果想重新调回，需要输入fg [id]，同理也可以使用bg（background）命令放到后台运行去，用法bg [id]
bg[id] =background把任务调到后台运行
fg[id] =foreground把任务调到前台运行

```
[root@localhost ~]# fg 1
vi 1.txt
[root@localhost ~]# fg 2
vi 10.txt.bak
[root@localhost ~]# jobs
[root@localhost ~]#
```

```
[root@localhost ~]# jobs
[1]+  已停止               vi 10.txt.bak
[root@localhost ~]# bg 1
[1]+ vi 10.txt.bak &
[root@localhost ~]#
```

sleep 命令；sleep 100=意思相当于让机器休眠静止100秒

```
[root@localhost ~]# sleep 100
^Z
[3]+  已停止               sleep 100
[root@localhost ~]# jobs
[1]   已停止               vi 10.txt.bak
[2]-  已停止               sleep 100
[3]+  已停止               sleep 100
[root@localhost ~]# bg 2
[2]- sleep 100 &
[root@localhost ~]# bg 3
[3]+ sleep 100 &
[root@localhost ~]# jobs
[1]+  已停止               vi 10.txt.bak
[2]   运行中               sleep 100 &
[3]-  运行中               sleep 100 &
[root@localhost ~]#
```

“&” 命令：后面加&相当于直接把任务调到后台运行

```
[root@localhost ~]# sleep 100 &
[2] 2341
[root@localhost ~]# jobs
[1]+  已停止               vi 10.txt.bak
[2]-  运行中               sleep 100 &
[root@localhost ~]#
```

七、shell变量

env命令，查看系统常用的环境变量

```
[root@aming01 ~]# env
XDG_SESSION_ID=2
HOSTNAME=aming01
SELINUX_ROLE_REQUESTED=
TERM=xterm
SHELL=/bin/bash
HISTSIZE=2000
SSH_CLIENT=192.168.0.107 54933 22
SELINUX_USE_CURRENT_RANGE=
SSH_TTY=/dev/pts/0
USER=root

01;36:*.axa=01;36:*.oga=01;36:*.spx=01;36:*.xspf=01;36:
MAIL=/var/spool/mail/root
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/tmp/done:/root/bin:/tmp/done:/tmp/done
PWD=/root
LANG=zh_CN.UTF-8
SELINUX_LEVEL_REQUESTED=
HISTCONTROL=ignoredups
SHLVL=1
HOME=/root
LOGNAME=root
SSH_CONNECTION=192.168.0.107 54933 192.168.0.115 22
LESSOPEN=||/usr/bin/lesspipe.sh %s
XDG_RUNTIME_DIR=/run/user/0
_=/usr/bin/env
```

set命令，不仅可以查看系统内置的环境变量，还可以查看用户自定义的变量

自定义变量

```
[root@aming01 ~]# a=123
[root@aming01 ~]# echo $a
123
[root@aming01 ~]# set |grep 123
_=123
a=123
```

变量名规则

变量名可以是字母、数字和下划线，首位不能为数字

```
[root@aming01 ~]# a1=1
[root@aming01 ~]# echo $a1
1
[root@aming01 ~]# a_1=2
[root@aming01 ~]# echo $a_1
2
[root@aming01 ~]# 1a=3
-bash: 1a=3: 未找到命令
```

变量值有特殊符号时需要用单引号（脱义）括起来

```
[root@aming01 ~]# a='a b c '
[root@aming01 ~]# echo $a
a b c
```

表示式复杂的时候，用双引号引起来，变量的累加

```
[root@localhost ~]# a=1
[root@localhost ~]# b=2
[root@localhost ~]# echo $a$b
12
[root@localhost ~]# a='a$bc'
[root@localhost ~]# echo $c

[root@localhost ~]# echo $a
a$bc
[root@localhost ~]# echo $a$b
a$bc2
[root@localhost ~]# c="a$bc"
[root@localhost ~]# echo $c
a
[root@localhost ~]# c="a$b"c
[root@localhost ~]# echo $c
a2c
[root@localhost ~]# c='a$b'c
[root@localhost ~]# echo $c
a$bc
[root@localhost ~]# c=a"$b"c
[root@localhost ~]# echo $c
a2c
[root@localhost ~]#
```

获取变量的值并插入到字符串中间

```
[root@long01 test]# echo "e$abc"   #这样的写法获取不到变量的值。
e
[root@long01 test]# echo "e${a}${b}c"  #用大括号将变量名括起来就可以获取到变量的值了。
ea b1 2c
```

全局变量

当打开多个终端时，使用命令echo $SSH_TTY，可以查看当前所在终端

```
[root@aming01 ~]# echo $SSH_TTY
/dev/pts/0
[root@aming01 ~]#
```

```
[root@aming01 ~]# echo $SSH_TTY
```

```
/dev/pts/1
[root@aming01 ~]#
```

在终端1定义一个变量，在终端2中是查看不到的

```
[root@aming01 ~]# echo $a
a b c
[root@aming01 ~]# echo $a
1
```

bash是shell的一个子bash，可以通过pstree查看在哪，如果没有该命令，使用如下命令安装：

```
[root@aming01 ~]# yum install psmisc
已加载插件：fastestmirror
Loading mirror speeds from cached hostfile

[root@aming01 ~]# pstree
systemd─┬─NetworkManager───2*[{NetworkManager}]
        ├─VGAuthService
        ├─auditd───{auditd}
        ├─crond
        ├─dbus-daemon───{dbus-daemon}
        ├─firewalld───{firewalld}
        ├─login───bash
        ├─lvmetad
        ├─master─┬─pickup
        │        └─qmgr
        ├─polkitd───5*[{polkitd}]
        ├─rsyslogd───2*[{rsyslogd}]
        ├─sshd─┬─sshd───bash───pstree
        │      └─sshd───bash
```

现在是在第二个bash下，第一个bash所设定的环境变量在第二个bash下不生效，如何让第一个bash的自定义环境变量在第二个bash下生效，利用全局变量：

```
[root@aming01 ~]# export b=111
[root@aming01 ~]# bash
[root@aming01 ~]# echo $b
111
```

取消变量

```
[root@aming01 ~]# unset b
[root@aming01 ~]# echo $b
```

八、环境变量配置文件

环境变量配置文件可分为：系统层次配置文件和用户层次配置文件

系统层次配置文件（/etc下的文件）：
1）/etc/profile 用户环境变量，交互，登录才执行
2）/etc/bashrc 用户不用登录，执行shell就生效

用户层次配置文件（用户家目录下的文件）：
1）~/.bashrc
2）~/.bash_profile

一般我们不要编辑系统层次的配置文件，在有需要时，可以编辑用户层次的配置文件.bash_profile

source .bash_profile和. .bash_profile作用一样，加载配置文件里的配置

- ~/.bash_history，用来记录命令历史
- ~/.bash_logout ，用来定义用户退出的时候需要做的一些操作

ps1是在/etc/bashrc里面定义的，用于定义命令左边的字符串显示，如下

```
[root@aming01 ~]#
1
```

修改W为w，发现变成绝对路径

```
[root@localhost ~]# echo $PS1
[\u@\h \W]\$
[root@localhost ~]# PS1='[\u@\h \w]\$'
[root@localhost ~]#cd /etc/lo
locale.conf     localtime       login.defs      logrotate.conf  logrotate.d/
[root@localhost ~]#cd /usr/local/src/
[root@localhost /usr/local/src]#
```

修改方括号，这里的普通用户显示普通用户显示，root用户显示#

```
[root@localhost /usr/local/src]#PS1='<\u@\h \w>\$'
<root@localhost /usr/local/src>#
```

颜色显示

```
[root@localhost /usr/local/src]#PS1='[\033[01;32m]\u@\h[\033[00m]:[\033[01;36m]\w[\033[00m
]\$ '
[]root@localhost[]:[]/usr/local/src[]#
```