**SHELL环境及特性**

**什么是shell？**

**在linux内核与用户之间的解释器程序**

**——通常指/bin/bash**

**——负责向内核翻译及传达用户/程序指令**

**——相当于操作系统的 “外壳“**

**shell与bash的区别：shell是一个概念；而bash则是一个明确的产品**

**常见的shell程序种类**

**如何切换shell环境**

**——通过usermod、chsh更改登录shell**

**——手动执行目标shell程序**

**[root@room8pc205 桌面]# cat /etc/shells**  
**/bin/sh                             ###多数Unix默认的shell**  
**/bin/bash                        ###多数Linux默认使用的shell**  
**/sbin/nologin                 ###非登录shell**

**[root@svr7 ~]# yum -y install ksh               ###若缺少ksh请先安装ksh包**  
**[root@svr7 ~]# cat /etc/shells**  
**/bin/sh**  
**/bin/bash**  
**/sbin/nologin**  
**/bin/ksh                                ### 确认当前系统已识别ksh**  
**/bin/rksh**

**[root@svr7 ~]# ksh            ###进入ksh环境下**  
**# exit                                     ###退出ksh环境**  
**[root@svr7 ~]# usermod -s /bin/tcsh zhangsan         ###永久改变zhangsan的登录shell为/bin/tcsh**  
===============================================================  
**历史命令**

**默认记录1000条**

**——保存位置：～/.bash\_history**

**——控制历史命令的数量: /etc/profile**

**[root@room8pc205 桌面]# grep ^HISTSIZE /etc/profile**  
**HISTSIZE=1000              ###全局设置的记录个数**

**清空已记录的历史命令**  
**[root@svr7 ~]# history -c                      ###清空自己的历史命令**  
**[root@svr7 ~]# > ~/.bash\_history       ###清空记录文件**

==========================================================================================================

**别名**

**别名设置一般存放在用户的.bashrc文件内：**

**[root@svr7 ~]# cat ~/.bashrc**  
**alias rm='rm -i'**  
**alias cp='cp -i'**  
**alias mv='mv -i'**

**取消别名**

**取消单个别名：[root@svr7 ~]# unalias 别名**

**取消所有别名：[root@svr7 ~]# unalias -a**  
==========================================================================================================  
**重定向标准输入/输出/错误输出**  
**标准输入（stdin），描述号为0；标准输出（stdout），描述号为1；标准错误（stderr），描述号为2。**

**重定向标准错误:对于命令执行出错的信息，使用 > 无法保存，仍然会输出到屏幕。**

**比如，可使用ls命令同时查看两个对象（其中nb并不存在），重定向输出：**

**[root@svr7 ~]# ls /etc/passwd nb.txt > my.log**  
**ls: 无法访问nb.txt: 没有那个文件或目录**

**使用 2> 可重定向错误信息:**

**[root@svr7 ~]# ls /etc/passwd nb.txt 2> my.log      ###把错误的提示信息重定向输入到my.log里**  
**/etc/passwd**  
  
**[root@svr7 ~]# ls /etc/passwd nb.txt &> my.log    ####把所有的提示信息重定向输入到my.log里**  
**[root@svr7 ~]# cat my.log**  
**ls: 无法访问nb.txt: 没有那个文件或目录**  
**/etc/passwd**  
  
**[root@svr7 ~]# ls /etc/passwd nb.txt > 1.log 2> 2.log      ###把正确的提示信息重定向输入到1.log；错误的输入到2.log**  
**[root@svr7 ~]# cat 1.log**  
**/etc/passwd**  
**[root@svr7 ~]# cat 2.log**  
**ls: 无法访问nb.txt: 没有那个文件或目录**i  
  
**案例1：未利用“重定向“的脚本：**

**[root@svr7 ~]# vim test.sh**

**#! /bin/bash**  
**for i in {1..10}**  
**do**  
**useradd user$i**  
**echo '123' | passwd --stdin user$i**  
**done**  
**案例2：利用了重定向后的脚本：**

**[root@svr7 ~]# vim test.sh**

**#! /bin/bash**  
**rm -rf error.log**  
**for i in {1..10}**  
**do**  
**useradd user$i 2>> error.log**  
**echo '123' | passwd --stdin user$i > /dev/null**  
**done**  
**案例1 与 案例2 的区别：案例1，未利用重定向与rm，会导致所有的结果显示在当前屏幕；而案例2则不会，错误的提示会在文件error.log里！！！**

**重定向标准输入：  "<"         默认输入是键盘。**

**[root@svr7 ~]# mail -s haha root < user.txt     ###把user.txt的内容输入到邮件内容**

**案例：简单Shell脚本的设计**

**1，在屏幕上输出一段文字“Hello World”**

**[root@svr7 ~]# vim test.sh**

**#! /bin/bash**  
**echo 'hello world'**  
**[root@svr7 ~]# sh test.sh**  
**hello world**

**2，能够为本机快速配好Yum仓库,并为本机快速装配好vsftpd服务**

**[root@svr7 ~]# vim test.sh**  
**#! /bin/bash**  
**rm -rf /etc/yum.repos.d/\***  
**echo "[id]**  
**name=rh7**  
**baseurl=http://192.168.4.254/rh7dvd**  
**gpgcheck=0" > /etc/yum.repos.d/rh7.repo**  
**yum clean all**  
**yum repolist**  
**yum -y install vsftpd                            ###自动安装vsftpd服务**  
**systemctl restart vsftpd                     ###重启vsftpd服务**  
  
**Shell脚本的执行方式**

**方法一，作为“命令字”：指定脚本文件的路径，前提是有 x 权限**  
**[root@svr7 ~]# .  /test.sh                                ###指定相对路径**  
**[root@svr7 ~]#  /root/test.sh                         ###指定绝对路径**

**方法二，作为“参数”：使用bash、sh、source来加载脚本文件**

**[root@svr7 ~]# bash  /test.sh               ###开启子进程**  
**[root@svr7 ~]# sh   /test.sh                   ###开启子进程**

**[root@svr7 ~]# source   /test.sh          ###不开启子进程**

**开启子进程 与  不开启 的区别（只对少数的脚本产生不同的影响）**

**[root@svr7 ~]# vim +46 /etc/profile           ###光标直接调到46行**  
**HISTSIZE=50                                               ###修改历史命令数为50条**  
**[root@svr7 ~]# history |wc -l                      ###查看当前历史命令数**  
**144**  
**[root@svr7 ~]# sh /etc/profile                    ###用sh 重新加载/etc/profile**  
**[root@svr7 ~]# history |wc -l                     ###查看**  
**146**  
**[root@svr7 ~]# source /etc/profile           ###用source 重新加载/etc/profile**  
**[root@svr7 ~]# history |wc -l                    ###查看**  
**50**  
  
**使用shell变量**

**1） 新建/赋值变量**

**只能使用 “字母、数字、\_ 这三个来表示；不能用数字开头 ”**

**新建变量SCHOOL，赋值“Taren IT GROUP” , 通过set命令可以检查变量设置：**

**[root@svr7 ~]# SCHOOL="Taren IT"**  
**[root@svr7 ~]# set | grep SCHOOL**  
**SCHOOL='Taren IT'**  
**2)      查看变量**

**通过 echo $变量名  可输出变量值：**

**[root@svr7 ~]# echo $SCHOOL**  
**Taren IT**  
**3)     撤销自定义变量**

**使用unset命令：**

**[root@svr7 ~]# unset SCHOOL**

**使用为值变量与预定义变量**  
**[root@svr7 ~]# vim test.sh**

**#! /bin/bash**  
**echo $0**  
**echo $1**  
**echo $2**  
**echo $3**  
**echo $\***  
**echo $#**  
**echo $$**

**echo $?**

**[root@svr7 ~]# sh test.sh aa 1 c 99 tt**  
**test.sh          ###等于$0（脚本名称）**  
**aa                 ###等于$1（第一个参数）**  
**1                   ###等于$2（第二个参数）**  
**c                    ###等于$3（第三个参数）**  
**aa 1 c 99 tt            ###等于$\*（所有参数）**  
**5                    ###等于$#** **（参数的数量）**  
**20755            ###等于$$(shell 的进程ID号，即PID)**

**0                   ###等于$?——>判断上一个程序命令是否成功（为0是成功；非0则是不成功）**

**案例：创建账户与修改密码的脚本**

**[root@svr7 ~]# vim test1.sh**  
**#! /bin/bash**  
**useradd $1**  
**echo "$2" | passwd --stdin $1**  
**[root@svr7 ~]# sh test1.sh aa sb**  
**更改用户 aa 的密码 。**  
**passwd：所有的身份验证令牌已经成功更新。**  
  
**变量的扩展应用**

**1） 三种引号对变量赋值的影响**

**——双引号：界定一个完整字符串。**

**——单引号：界定一个完整字符串，并可以实现屏蔽特殊符号的功能。**

**双引号 与  单引号的区别：**

**[root@svr7 ~]# note="当前用户是:$USER"  
[root@svr7 ~]# echo $note  
当前用户是:root  
[root@svr7 ~]# note='当前用户是:$USER'  
[root@svr7 ~]# echo $note  
当前用户是:$USER**

**——反撇号 ` 或  $() 时，可以将命令执行的标准输出作为字符串存储，因为成为命令替换。**

**[root@svr7 ~]# tar -czf  log-`date +%Y%m%d`.tar.gz  /var/log**  
**[root@svr7 ~]# ls**  
**log-20180301.tar.gz**  
**[root@svr7 ~]# tar -czf log-`date +%F`.tar /var/log**  
**[root@svr7 ~]# ls**  
**log-2018-03-01.tar**

**2)使用read命令从键盘读取变量值**

**——read基本用法：执行后会等待并接受用户输入，并赋值给变量;**

**为了不让用户不知所措，使用结合 -p  选项给出友好提示：**

**[root@svr7 ~]# vim test2.sh**

**#! /bin/bash**  
**read -p "请输入用户名：" user**  
**read -p "请输入密码：" passwd**  
**useradd $user**  
**echo $passwd | passwd --stdin $user**  
**[root@svr7 ~]# sh test2.sh**  
**请输入用户名：abc**  
**请输入密码：123**  
**更改用户 abc 的密码 。**  
**passwd：所有的身份验证令牌已经成功更新。**

**——stty终端显示控制 ： 将回显功能关闭（stty -echo）;**

**将回显功能恢复（stty echo）。**

**[root@svr7 ~]# vim test2.sh**

**#! /bin/bash**  
**read -p "请输入用户名：" user**

**stty -echo**  
**read -p "请输入密码：" passwd**

**stty echo**  
**useradd $user**  
**echo $passwd | passwd --stdin $user**  
**[root@svr7 ~]# sh test2.sh**

**请输入用户名：hahe**

**请输入密码：                  ###密码是不明文显示的**

**更改用户 hahe 的密码 。**

**passwd：所有的身份验证令牌已经成功更新。**  
  
**3）使用export发布全局变量**

**——默认情况下，自定义的变量为局部变量，只在当前Shell环境中有效，而在子Shell环境中无法直接使用。若希望定义的变量能被子进程使用，可以使用export命令将其发布为全局变量。使用export发布时，只需指定变量名（可以有多个）即可，也可以通过export命令直接设置新的全局变量。**

**[root@svr7 ~]# export t=111**  
**[root@svr7 ~]# echo $t**  
**111**  
**[root@svr7 ~]# bash**  
**[root@svr7 ~]# echo $t**  
**111**