还原三台虚拟机classroom server desktop

[root@server0 ~]# rht-vmctl reset classroom

[root@server0 ~]# rht-vmctl reset server

[root@server0 ~]# rht-vmctl reset desktop

确认虚拟机server Yum是否可用

[root@server0 ~]# yum clean all

[root@server0 ~]# yum repolist

[root@server0 ~]# yum -y install vsftpd

######################################################

一块硬盘的“艺术”之旅

• 识别硬盘 => 分区规划 => 格式化 => 挂载使用

总结:

1.lsblk 识别本机的硬盘

2.fdisk 进行分区

3.partprobe 进行刷新

4.mkfs.xfs 格式化

5.mount 手动挂载使用

6./etc/fstab 开机自动挂载使用

####################################################

Shell脚本编程

shell语言

脚本: 一个可以执行的文件,执行实现某种功能

案例1:

请用 shell语言实现一个脚本,运行结果在屏幕输出hello world

[root@server0 ~]# vim /root/hello.sh

echo hello world

[root@server0 ~]# chmod +x /root/hello.sh #赋予所有人执行权限

[root@server0 ~]# /root/hello.sh #绝对路径运行

########################################################

规范Shell脚本的一般组成

• #! 环境声明

• # 注释文本

• 可执行代码

########################################################

案例2:

编写一个面世问候 /root/hello.sh 脚本

– 显示出一段话“Hello World!!”

1)输出当前红帽系统的版本信息

2)输出当前使用的内核版本

3)输出当前系统的主机名

[root@server0 ~]# vim /root/hello.sh

#!/bin/bash

echo hello world

cat /etc/redhat-release

uname -r

hostname

ifconfig | head -2

######################################################

简单脚本技巧

重定向输出

>: 将前面命令的正确输出,进行重定向

2>: 将前面命令的错误输出,进行重定向

&>: 将前面命令的正确与错误输出,进行重定向

[root@server0 ~]# echo 123 > /opt/1.txt

[root@server0 ~]# cat /opt/1.txt /etc/

[root@server0 ~]# cat /opt/1.txt /etc/ > /mnt/a.txt

[root@server0 ~]# cat /mnt/a.txt

[root@server0 ~]# cat /opt/1.txt /etc/ 2> /mnt/a.txt

[root@server0 ~]# cat /mnt/a.txt

[root@server0 ~]# cat /opt/1.txt /etc/ &> /mnt/a.txt

[root@server0 ~]# cat /mnt/a.txt

##################################################

案例3:

创建用户zhangsan的脚本,并且设置密码123

[root@server0 ~]# vim /root/user.sh

#!/bin/bash

useradd zhangsan

echo 123 | passwd --stdin zhangsan

[root@server0 ~]# chmod +x /root/user.sh

[root@server0 ~]# /root/user.sh

黑洞设备:/dev/null

为增加灵活度,为了适用多变的环境--------->变量

变量:会变化的量, 类似于容器,以不变的名字,存储可以变化的值

– 变量名=变量值

[root@server0 ~]# vim /root/user.sh

#!/bin/bash

abc

useradd $abc &> /dev/null

echo 用户$abc创建成功

echo 123 | passwd --stdin $abc &> /dev/null

echo 用户$abc密码设置成功

[root@server0 ~]# /root/user.sh

为了脚本使用难度降低,方便用户随意输入参数 --------> 产生交互

read -p 记录用户在键盘上的输入,并且传递值赋予变量

[root@server0 ~]# vim /root/user.sh

#!/bin/bash

read -p '请输入您要创建的用户名:' abc

useradd $abc &> /dev/null

echo 用户$abc创建成功

echo 123 | passwd --stdin $abc &> /dev/null

echo 用户$abc密码设置成功

######################################################

定义/赋值变量

• 设置变量时的注意事项

– 若指定的变量名已存在,相当于为此变量重新赋值

– 等号两边不要有空格

– 变量名只能由字母/数字/下划线组成,区分大小写

– 变量名不能以数字开头,不要使用关键字和特殊字符

基本格式

– 引用变量值:$变量名

– 查看变量值:echo $变量名、echo ${变量名}

[root@server0 ~]# a=rhel

[root@server0 ~]# echo $a

[root@server0 ~]# echo $a7

[root@server0 ~]# echo ${a}7

$[] : 运算

+ - \* /

%(取余运算)

^(指数运算)

[root@server0 ~]# echo $[1+3]

[root@server0 ~]# echo $[1-3]

[root@server0 ~]# echo $[2\*3]

[root@server0 ~]# echo $[10/3]

[root@server0 ~]# echo $[10%3]

[root@server0 ~]# echo 1+1 | bc

[root@server0 ~]# echo 2\*3 | bc

[root@server0 ~]# echo 2^3 | bc

数学上: 余数一定小于除数

$()与 ` ` 反撇号:将命令的输出结果作为参数

[root@server0 opt]# cd /opt

[root@server0 opt]# date

[root@server0 opt]# date +%F

[root@server0 opt]# mkdir $(date +%F)

[root@server0 opt]# ls

[root@server0 opt]# mkdir nsd-$(date +%F)

[root@server0 opt]# mkdir MySQL-$(date +%F)

[root@server0 opt]# ls

[root@server0 opt]# mkdir `hostname`-`date +%F`

[root@server0 opt]# ls

' ': 取消特殊字符的意义

##################################################

变量的种类

环境变量 变量名一般都大写,由系统定义完成,直接使用即可

USER:永远储存当前登陆的用户名

位置变量 由系统定义完成,直接使用即可

1 2 3 4 5 6 7 ........

[root@server0 /]# vim /root/2.sh

#!/bin/bash

cat -n $1 | head -$2

[root@server0 /]# chmod +x /root/2.sh

[root@server0 /]# /root/2.sh /etc/passwd 3

预定义变量 由系统定义完成,直接使用即可

$# 已加载的位置变量的个数(求和所有位置变量个数)

$\* 所有位置变量的值

[root@server0 /]# vim /root/2.sh

#!/bin/bash

cat -n $1 | head -$2

echo $#

echo $\*

[root@server0 /]# /root/2.sh /etc/passwd 4 a b c d

$? 程序退出后的状态值,0表示正常,其他值异常,判断上面命令是否执行成功

#####################################################

条件测试

• 检查文件状态

-e:文档存在为真

-d:文档存在,且为目录为真

-f:文档存在,且为文件为真

[root@server0 /]# [ -e /etc ]

[root@server0 /]# echo $?

0

[root@server0 /]# [ -d /etc/passwd ]

[root@server0 /]# echo $?

1

[root@server0 /]# [ -f /etc/passwd ]

[root@server0 /]# echo $?

0

[root@server0 /]# [ -f /etc ]

[root@server0 /]# echo $?

1

• 比较整数大小 greater(大于) little(小于) equal(等于)

-gt: 大于

-ge: 大于等于

-eq: 等于

-ne: 不等于

-lt: 小于

-le: 小于等于

[root@server0 /]# [ 1 -gt 1 ]

[root@server0 /]# echo $?

[root@server0 /]# [ 1 -ge 1 ]

[root@server0 /]# echo $?

[root@server0 /]# [ 1 -le 2 ]

[root@server0 /]# echo $?

[root@server0 /]# [ 1 -eq 3 ]

[root@server0 /]# echo $?

• 字符串比对

==: 字符串相等为真

!=:字符串不相等为真

[root@server0 /]# [ root == redhat ]

[root@server0 /]# echo $?

1

[root@server0 /]# [ root != redhat ]

[root@server0 /]# echo $?

0

[root@server0 /]# [ root != root ]

[root@server0 /]# echo $?

1

[root@server0 /]# [ redhat == user ]

[root@server0 /]# echo $?

######################################################

if选择结构

if双分支

if [条件测试];then

命令序列xx

else

命令序列yy

fi

案例4

请书写脚本,用户输入一个IP地址,进行判断

如果IP地址可以ping通 ,则输出 可以通信

如果IP地址不可以ping通 ,则输出 不可以通信

[root@server0 /]# vim /root/ip.sh

#!/bin/bash

read -p '请输入您测试的IP地址:' ip

ping -c 2 $ip &> /dev/null

if [ $? -eq 0 ];then

echo 可以通信

else

echo 不可以通信

fi

案例5:

计算机随机产生一个0~9的随机数字

用户输入一个0~9的随机数字

判断:如果 用户输入 与 计算机随机产生的数字一致,输出 恭喜您猜对了

如果 用户输入 与 计算机随机产生的数字一致,输出 猜错误了

RANDOM:储存随机数的环境变量

[root@server0 /]# vim /root/num.sh

#!/bin/bash

num1=$[$RANDOM%10]

read -p '请您输入一个0~9之间的数字:' num2

if [ $num1 -eq $num2 ];then

echo 恭喜您猜对了

else

echo 猜错误了

echo 正确的是$num1

fi

#####################################################

if多分支处理

if [条件测试1];then

命令序列xx

elif [条件测试2];then

命令序列yy

elif [条件测试3];then

命令序列aa

......

else

命令序列zz

fi

案例6:

用户输入成绩,判断

如果大于等于90,则 输出 优秀

如果大于等于80,则 输出 良好

如果大于等于70,则 输出 合格

如果大于等于60,则 输出 多努力

以上条件均不满足,则 输出 在牛的肖邦,也弹不出哥的悲伤

[root@server0 /]# vim /root/if.sh

#!/bin/bash

read -p '请输入您的成绩:' num

if [ $num -ge 90 ];then

echo 优秀

elif [ $num -ge 80 ];then

echo 良好

elif [ $num -ge 70 ];then

echo 合格

elif [ $num -ge 60 ];then

echo 多努力

else

echo 在牛的肖邦,也弹不出哥的悲伤

fi

#################################################

案例7:

计算机随机产生一个0~9的随机数字

用户输入一个0~9的随机数字

判断:

如果 用户输入 与 计算机随机产生的数字一致,输出 恭喜您猜对了

如果 用户输入 与 计算机随机产生的数字不一致,如果用户的数字大于计算机产生的数字输出猜大了,如果用户的数字小于计算机产生的数字输出猜小了.

[root@server0 /]# vim /root/num.sh

#!/bin/bash

num1=$[$RANDOM%10]

read -p '请您输入一个0~9之间的数字:' num2

if [ $num1 -eq $num2 ];then

echo 恭喜您猜对了

elif [ $num2 -gt $num1 ];then

echo 猜大了

echo 正确的是$num1

else

echo 猜小了

echo 正确的是$num1

fi

案例8:

在案例7的基础上,允许用户可以有三次机会

for循环:

作用反复执行重复性的操作

列表式循环:

for 变量名 in 值列表

do

命令序列

done

[root@server0 /]# vim /root/for.sh

#!/bin/bash

for i in {1..10}

do

useradd stu$i

echo stu$i创建成功

done

Linux造数: {起始..结束} 例:{3..10}

案例8:

在案例7的基础上,允许用户可以有三次机会

[root@server0 /]# vim /root/num.sh

#!/bin/bash

num1=$[$RANDOM%10]

for i in {1..3}

do

read -p '请您输入一个0~9之间的数字:' num2

if [ $num1 -eq $num2 ];then

echo 恭喜您猜对了

exit

elif [ $num2 -gt $num1 ];then

echo 猜大了

else

echo 猜小了

fi

done