**Engineer02**

shell 脚本基础

脚本:可以执行的一个文件,实现某种功能

提前设计可执行语句,用来完成特定任务的文件

-解释型程序

-顺序、批量执行

案例 编写一个面试问候 /root/hello.sh 脚本

显示出一段话 Hello World

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

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

[root@server0 ~]# ls -l /root/hello.sh

-rwxr-xr-x. 1 root root 17 12月 10 09:41 /root/hello.sh

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

规范shell脚本的一般组成

#!环境声明 编辑语音环境 #!/bin/bash

#注释文本

可执行代码

重定向输出

>:只收集前面命令的正确输出

2>:只收集前面命令的错误输出

&>:只收集前面命令的正确输出与错误输出

案例:书写一个创建用户的脚本,并且设置密码

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

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

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

#!/bin/bash

useradd nsd01 &> /dev/null

echo 用户nsd01创建成功

echo 123 | passwd --stdin nsd01 &> /dev/null

echo 用户nsd01密码创建成功

**黑洞设备:/dev/null**

为了脚本的灵活度,适应多变的环境,使用变量

变量=容器 以不变的名称,存放可以变化的值

定义变量:

变量名=存放的值

#!/bin/bash

a=nsd03

useradd $a &> /dev/null

echo 用户$a创建成功

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

echo 用户$a密码创建成功

定义/赋值变量

• 设置变量时的注意事项

– 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

– 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

查看/引用变量

• 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

rhel

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

rhel

[root@server0 ~]# echo $a7

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

rhel7

[root@server0 ~]#

补充:

单引号’’:取消特殊字符的意义,原样输出

反撇号``或$():将命令的输出,直接参与其他命令的运行

[root@server0 ~]# rm -rf /opt/\*

[root@server0 ~]# mkdir /opt/nsd-`date +%F`

[root@server0 ~]# ls /opt/

[root@server0 ~]# mkdir /opt/abc-`date +%F`

[root@server0 ~]# ls /opt/

[root@server0 ~]# mkdir /opt/`hostname`-$(date +%F)

[root@server0 ~]# ls /opt/

根据变量的用途不同区分

**环境变量:变量名一般都大写,用来设置用户/系统环境**

**由系统定义完成,由系统赋值完成**

USER=当前系统登录的用户名

**位置变量:bash内置,储存执行脚本时提供的命令行参数**

**由系统定义完成,由系统赋值完成**

表示为$n,n为序号

$1 $2 ..... ${10} ${11} ......

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

#!/bin/bash

a=$1

useradd $1 &> /dev/null

echo 用户$1创建成功

echo 123 | passwd --stdin $1 &> /dev/null

echo 用户$1密码创建成功

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

用户natasha创建成功

用户natasha密码创建成功

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

用户harry创建成功

用户harry密码创建成功

[root@server0 ~]# vim /root/1.sh

[root@server0 ~]# cat /root/1.sh

#!/bin/bash

cat -n $1 | head -$2

[root@server0 ~]# /root/1.sh /etc/passwd 3

-bash: /root/1.sh: 权限不够

[root@server0 ~]# chmod +x /root/1.sh

为了降低脚本的使用难度,提高用户的体验度,采用交互式传递参数的方法

**read作用:**

**1.产生交互**

**2.自动记录用户在键盘上的输入**

**3.自己将用户在键盘上的输入,赋值给一个变量存储**

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

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

#!/bin/bash

read -p '请输入您要创建的用户名' a

useradd $a &> /dev/null

echo 用户$a创建成功

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

echo 用户$a密码创建成功

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

**预定义变量:bash内置,可直接调用的特殊值,不能直接修改**

**由系统定义完成,由系统赋值完成**

$#已加载的位置变量的个数

$\*所有位置变量的值

$?程序退出后的状态值,0表示正常,其他值异常

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

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

[root@server0 ~]# cat /root/2.sh

#!/bin/bash

echo $1

echo $2

echo $3

echo 您一共输入了$#个位置变量

echo 他们的值分别为$\*

[root@server0 ~]# /root/2.sh dc tc dz tz

dc

tc

dz

您一共输入了4个位置变量

他们的值分别为dc tc dz tz

**自定义变量:用户资助设置、修改及使用**

条件测试及选择

条件测试

[ 测试表达式 ] #每一部分之间都要有空格

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

-r：文档存在，且对文档具备读取权限为真

-w：文档存在，且对文档具备写权限为真

-x：文档存在，且对文档具备执行权限为真

比较整数大小

**-gt:大于**

**-ge:大于等于**

**-eq:等于**

**-ne:不等于**

**-lt:小于**

**-le:小于等于**

字符串比对

• 字符串比对

== :相等一致为真

!= :不相等为真

**if选择结构**

if双分支处理

**if [条件测试];then**

**命令序列xx**

**else**

**命令序列yy**

**fi**

[root@server0 ~]# vim /root/if01.sh

[root@server0 ~]# chmod /root/if01.sh

[root@server0 ~]# cat /root/if01.sh

#!/bin/bash

if [ $1 -eq $2 ];then

echo 进入

else

echo 禁止入内

fi

案例：书写一个判断IP通信的脚本

用户输入IP地址，利用read -p 方式获得用户输入的IP地址

判断如果本机可以与该IP地址通信，则输出 可以通信

判断如果本机不可以与该IP地址通信，则输出 无法可以通信

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

#!/bin/bash

read -p '请输入您要测试的IP地址：' ip

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

if [ $? -eq 0 ];then

echo $ip可以通信

else

echo $ip无法通信

fi

[root@server0 /]#

**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：书写一个成绩脚本

用户输入考试成绩，利用 read -p 手法进行获得用户成绩

如果成绩 大于等于 90 ，则输出 优秀

如果成绩 大于等于 80 ，则输出 良好

如果成绩 大于等于 70 ，则输出 一般

如果成绩 大于等于 60 ，则输出 合格

如果以上条件均不满足，则输出 再牛的肖邦也弹不出哥的悲伤！

[root@server0 /]# vim /root/if04.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 一般

elif [ $num -ge 60 ];then

echo 合格

else

echo '再牛的肖邦，也弹不出哥的悲伤！'

fi

**for循环结构:可以重复执行一个操作**

for 变量名(献血车) in 值列表(lisi )

do

命令序列(抽血)

done

**[root@server0 ~]# chmod /root/for01.sh**

**[root@server0 ~]# chmod +x /root/for01.sh**

**[root@server0 ~]# cat /root/for01.sh**

**#!/bin/bash**

**for a in zhangsan lisi wangwu dc tc dz tz**

**do**

**useradd $a &> /dev/null**

**echo $a 创建成功**

**done**

for循环中,变量的值可不参与循环执行的操作

造数工具:{起始点..结束点} {1..50}生成1到50之间的所有数字

**[root@server0 ~]# cat /root/for01.sh**

**#!/bin/bash**

**for a in zhangsan {1..15}**

**do**

**echo hello world**

**done**