|  |
| --- |
|  |

### shell基础

脚本：可以执行的一个文本文件，可以实现某种功能。

规范shell脚本的一般组成

#！环境声明（以下代码用哪一个程序来解释,默认为/bin/bash）

#注释文本

可执行代码 赋予x权限

重定向输出

>:只收集正确输出

2>:之收集错误输出

&>:正确输出和错误输出都收集

$[]:代表运算 []里面写运算表达式，

运算符：+ - \* / %

‘ ’：(单引号)取消所有字符特殊意义

$():将命令的输出，当作参数参与其他命令的执行

()里面放入可执行输出的命令

创建设备，并创建密码

黑洞设备：/dev/null

变量：

变量名要求：

变量名不能有空格

变量名只能由字母、数字、下划线组成，严格区分大小写

变量名不能由数字开头，不能使用特殊字，尽量不要用特殊字给变量命名

变量的引用：

$变量名

${变量名}

${变量名}123 以{}界定易混淆的

环境变量：有系统定义完成，并且赋值完成，用户直接使用

USER=永远储存当前登陆的用户名

位置变量：由系统定义完成，并且赋值完成，用户直接使用

作用：在执行脚本时提供命令行的参数

表示：$n,n为序号

$1,$2,$3...${10}

编写脚本代码：Vim /mnt/xiao.sh

cat -n ${1} | tail -${2}

修改执行权限：Chmod +x /mnt/xiao.sh

执行脚本：/mnt/xiao.sh /etc/passwd 5 #cat -n ${1} | tail -${2}

预定义变量：有系统定义完成，并且赋值完成，用户直接使用

$#以加载的位置变量的个数，球和

$\* 所有位置参数的值

脚本xioa.sh： #!/bin/bash

echo ${1}

echo ${2}

echo $#

echo $\*

运行脚本：/mnt/xiao.sh 1 2 3 4

输出：1

2

4

1 2 3 4

$? 程序执行后的状态值，0为执行成功，其他值异常

以上面的xiao.sh为例：echo $?

输出：0

增加用户体验程度，降低用胡使用难度

read：产生交互，捕捉用户在键盘上的输入，并且赋值给一个变量

-p：‘屏幕提示信息’

read -p ‘屏幕提示信息’ a

条件表达及机选择判断

条件表达式：

每一部分都要有空格

检查文件状态：

-e：文档存在为真

-d：存在且为目录为真

-f：存在且为文件为真

比较整数大小：

-gt：> (greater than)

-ge:>= (greater equal)

-eq:= (equal)

-ne:!= (not equal)

-lt:< (less than)

-le:<= (less equal)

[ a -gt b ] 逻辑测试句

字符串比对

==：等于为真0

!=：不等于为真0

[ a == b ] 逻辑测试句

### if语句：

单分支：

if 条件测试;then

命令序列xx

fi

双分支：

if 条件测试;then

命令序列xx

else

命令序列yy

fi

例：#!/bin/bash

read -p '请输入IP地址：' a

ping -c 2 $a > /dev/null

b=$?

if [ $b -eq 0 ];then

echo ping通了

else

echo ping不通

fi

多分支：

if [ ]

随机数

1. 内部系统变量（$RANDOM）
2. $[$RANDOM%10] #10以内的随即数

$():将命令的输出，当作参数参与其他命令的执行

echo 123 >&2 # ‘>&2’ 将前面的‘123’输出作为错误输出

exit 2 # 退出并返回2

**案例：**

当运行/root/foo.sh redhat，输出为fedora

当运行/root/foo.sh fedora，输出为redhat

当没有任何参数或者参数不是 redhat 或者 fedora时，其错误输出产生以下信息： /root/foo.sh redhat|fedora

答：#!/bin/bash

a=fedora

b=redhat

if [ $# -eq 0 ];then

echo '/xiao/if01.sh redhat|fedora' >&2

exit 4

elif [ $1 == $a ];then

echo redhat

elif [ $1 == $b ];then

echo fedora

else

echo '/xiao/if01.sh redhat|fedora' >&2

exit 2

fi

### for语句

循环的列表值，可以不参与循环的命令序列

for 变量名 in 列表值

do

命令序列

done

造数工具：{起始值..结束值} # {1..50}

$():将命令的输出，当作参数参与其他命令的执行

echo 123 >&2 # ‘>&2’ 将前面的‘123’输出作为错误输出

exit 2 # 退出并返回2

## while循环

While [条件判断] # [条件判断]可以是一条命令，运行成功为真，反之为假；

do

循环执行的语句

done

**expect预期交互**

expect可以为交互式过程（比如FTP、SSH等登录过程）自动输送预先准备的文本或指令，而无需人工干预。触发的依据是预期会出现的特征提示文本。

Spawn ftp 192.168.4.100

Expect << EOF

Expect 提示 {send “交互内容”}

Expect 提示 {send “交互内容”}

Expect 提示 {send “交互内容”}

Expect 提示 {send “交互内容”}

EOF

\033[42m 033[0m