**shell day 02**

**=============================================**

**一，运算**

**1，expr 运算并输出结果**

**2，$[ ]或者$(( )) 运算 ，如果要直接看**

结果，可以配合echo命令

echo $[1+1]

echo $[2-1]

echo $[2\*2]

echo $[4/2]

echo $[4%2]

a=10

b=20

echo $[a+b]

1. **使用let命令，不输出结果，专用于变量的创建**

**或者变量的自增减**

let a=1+1 //创建变量

变量的自增减：

常规写法 主流写法

let a=a+1 let a++ 变量a加1

let a=a-1 let a-- 变量a减1

let a=a+10 let a+=10 变量a加10

let a=a-10 let a-=10 变量a减10

let a=a\*2 let a\*=2 变量a乘以2

let a=a/2 let a/=2 变量a除以2

let a=a%3 let a%=3 变量a除以3取余数

**4，使用bc，可以进行小数计算**

echo "10/3" | bc

echo "scale=2;10/3" | bc

---------------------------------------------------

**二，条件测试，可以赋予脚本智能判断的效果**

**语法格式 1, test 表达式 2, [ 表达式 ]**

**1，对字符串进行测试**

== 两边是否相等，相等时条件测试算成功

!= 两边是否不等，不相等时条件测试算成功

test a == a //方式一：test ，使用常量判断，再用echo $?测试结果

test a == b

[ a == a ] //方式二：中括号，使用常量判断，再用echo $?测试结果

[ a == b ]

a=abc

a=xyz

[ $a == $b ] //使用变量判断两个变量的值是否相等

[ $a != $b ] //使用变量判断两个变量的值是否不相等

[ "$c" == $b ] //如果变量为空，有可能报错，加双引号可以避免

-z

[ -z $a ] //判断变量是否为空，为空时条件测试算成功

[ ! -z $a ] //判断变量是否非空，非空时条件测试算成功

**2，逻辑组合**

**条件 && 指令 条件成功才执行指令**

**条件 || 指令 条件失败才执行指令**

[ root == $USER ] || exit //判断当前用户名是否为root

如果判断失败，则执行exit

[ $USER != root ] && exit //判断当前用户名是否不为root

如果判断成功，则执行exit

------------------------------------------------------------

**当多个逻辑符号组合使用时**

**A && B A、B任务都成功算成功**

**A || B A、B 任务有一个成功算成功**

touch a b c

ls a && ls b && ls c //结果是都显示

ls a || ls b || ls c //结果是显示a

ls a && ls b || ls c //结果是显示a和b

ls a || ls b && ls c //结果是显示a和c

[ root != $USER ] && echo "非管理员不能执行该脚本" && exit //之前

的语句可以改进成这个方式

**3，数字**

-eq 是否相等 -ne 是否不等 -gt 是否大于

-ge 是否大于等于 -lt 是否小于 -le 是否小于等于

编写脚本，每2分钟检查服务器的用户数量，如果发生变化

则发邮件通知管理员

编写脚本前 echo "测试" | mail -s test root 测试下邮件功能是否正常

yum -y install mailx postfix //如果不正常就安装软件包

systemctl start postfix //然后启动服务

#!/bin/bash

x=$(cat /etc/passwd | wc -l) //将目前用户数量赋值给变量x

[ $x -gt 29 ] && echo "用户数量发生变化，服务器可能被入

侵" | mail -s test root //如果目前用户数量大于29，就

发邮件给管理员，29是之前查看的用户数量

chmod u+x test01.sh //然后给脚本加x权限

crontab -e //编写计划任务

\*/2 \* \* \* \* /opt/test01.sh //定义每2分钟执行脚本

-------------------------------------

1. **文件**

-e 判断文件是否存在，不关心类型

-f 判断文件是否存在，必须是普通文件

-d 判断文件是否存在，必须是目录

-r 判断当前用户对文件是否有读权限，对root无效

-w 判断当前用户对文件是否有写权限，对root无效

-x 判断当前用户对文件是否有x权限

[ -e abc ] //判断当前目录下的abc是否存在，用echo $?查结果

[ -f /opt/abc ] //判断opt下有没有abc这个普通文件

[ -d /opt/abc ] //判断opt下有没有abc这个目录

mkdir xyz

[ -d /opt/xyz ] //判断opt下有没有xyz这个目录

[ -r /opt/abc ] //判断当前用户对abc是否有读权限

[ -w /opt/abc ] //判断当前用户对abc是否有写权限

[ -x /opt/abc ] //判断当前用户对abc是否有x权限

---------------------------------------------------------------------

== != -z ! -z

&& ||

-eq -ne -gt -ge -lt -le

-e -f -d -r -w -x

---------------------------------------

**三，if分支**

1. **单分支**

if 条件测试;then //满足条件的话就执行下面指令

执行指令

fi

#!/bin/bash

if [ $UID -eq 0 ];then

echo "我是管理员"

echo ok

fi

1. **双分支**

if 条件测试;then //满足条件的话就执行下面指令

执行指令A //指令可以是一条，也可以是多条

else //否则，不满足条件的话就执行下面指令

执行指令B

fi

#!/bin/bash

if [ $UID -eq 0 ];then

echo "我是管理员"

echo ok

else

echo "我不是管理员"

echo no

fi

------------------------------------------------

ping -c 3 -i 0.2 -W 1 192.168.2.254

// -c 定义ping的次数 -i 定义ping的间隔时间(秒)

-W ping失败时多久反馈结果

-------------------------------------------------

使用if双分支编写ping脚本

#!/bin/bash

ping -c 3 -i 0.2 -W 1 $1 &> /dev/null //使用$1位置变量

更方便

if [ $? -eq 0 ];then

echo "通了！"

else

echo "不通！"

fi

**3，多分支**

if 条件测试;then //满足条件的话就执行下面A指令

执行指令A //指令可以是一条，也可以是多条

elif 条件测试;then //满足条件的话就执行下面B指令

执行指令B

elif 条件测试;then //满足条件的话就执行下面C指令

执行指令C

else //否则，不满足条件的话就执行下面指令

执行指令

fi

#!/bin/bash

read -p "请输入月考成绩：" x

[ $x -lt 0 ] && echo "别闹" && exit

if [ $x -ge 90 ];then

echo "优秀！"

elif [ $x -ge 80 ];then

echo "良好！"

elif [ $x -ge 60 ];then

echo "及格。。"

else

echo "晚上没饭了！"

fi

-------------------------------------

**四，循环，可以反复执行某任务**

**for循环，可以定义循环次数**

for 变量名称 in 值1 值2 值3 。。。。 //此处变量名可以自定义，通常习惯

用i，值的多少决定了下面do与done之间的任务执行多少次，每个值之间有空

格，这里是有3个值，所以就循环执行指令3次

do

任务

done

----------------------------------

#!/bin/bash

for i in a b c //循环3次

do

ping

echo nb

echo $i

done

-----------------------------------

#!/bin/bash

for i in {1..100} //循环100次

do

ping

echo nb

echo $i

done

-------------------------------------

编写脚本，可以测试192.168.2.1~192.168.2.10

#!/bin/bash

for i in {1..10}

do

ping -c 3 -i 0.2 -W 1 192.168.2.$i &> /dev/null

if [ $? -eq 0 ];then

echo "通了"

else

echo "不通"

fi

done

--------------------------------------

改良版，可以统计最终通了和不通的数量

#!/bin/bash

x=0

y=0

for i in {1..10}

do

ping -c 3 -i 0.2 -W 1 192.168.2.$i &> /dev/null

if [ $? -eq 0 ];then

echo "192.168.2.$i 通了"

let x++ //如果通了一台就把变量x+1

else

echo "192.168.2.$i 不通"

let y++ //如果不通了一台就把变量y+1

fi

done

echo "$x台通了，$y台不通"

------------------------------------------------------------------------

练习：

1，列出常见的整数值比较操作，并说明各自作用。

2，运用条件测试操作，检查当前的用户是否为root。

3，绘图描述if双分支结构的执行流程。

4，[ -z $abc ] && echo "yes" || echo "no" 可以实现什么测试效果。

参考答案

1，列出常见的整数值比较操作，并说明各自作用。

-eq 等于

-ne 不等于

-ge 大于等于

-le 小于等于

-gt 大于

-lt 小于

2，运用条件测试操作，检查当前的用户是否为root。

[ $USER == "root" ] && echo "yes" || echo no

3，绘图描述if双分支结构的执行流程。

if双分支结构判断一次条件，当条件成立时执行分支1、若不成立则执行分支2，如图-1所示。

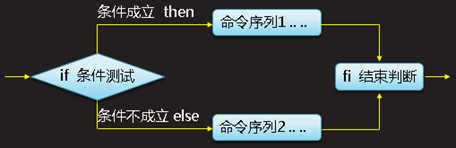


图-1

4，[ -z $abc ] && echo "yes" || echo "no" 可以实现什么测试效果。

如果变量abc是空则显示yes，否则显示no