回顾

bash的优点，快捷键，tab键，历史记录，别名，管道，重定向

编写脚本的格式

1. 声明解释器
2. 注释
3. 执行指令

变量

1，自定义变量 变量名称=变量的值

自定义变量的名称可以使用数字、字母、下划线，但不能以数字开头，不能使用特殊符号

a=10

echo ${a}XXXX

2，环境变量

USER UID HOME HOSTNAME SHELL PWD PATH PS1 PS2

1. 预定义变量和位置变量

$\* $# $$ $?

$1 $2 $3 ….

' ' " " `` $()

read stty -echo stty echo

练习一：运算

方式1：expr + - \* / %

方式2：$[ ] 或 $(( ))

echo $[1+1]

echo $[2-1]

echo $[2\*2]

echo $[4/2]

echo $[5%2]

a=10

b=20

echo $[a\*b] 支持变量运算，且变量名前无需再写$符号

方式3：let，运算结果不显示，通常用来创建变量，或者做变量的自增减

let 1+1 没有结果显示

let a=10+20 创建变量，变量的值是运算的结果

let a=a+1 或 let a++ 变量的自增减,要使用后者主流方式

let a=a+2 或 let a+=2

let a=a-2 或 let a-=2

let a=a\*2 或 let a\*=2

let a=a/2 或 let a/=2

let a=a%3 或 let a%=3

方式4，使用bc工具，可以计算小数

[root@svr7 opt]# echo "1.1+1" | bc

[root@svr7 opt]# echo "scale=3;10/3" | bc

练习二：条件测试，可以使脚本拥有智能判断的工具

基本格式： test 表达式 或者 [ 表达式 ]

1. 字符串

== 是否相等 != 是否不等 -z 判断变量是否为空

test abc == abc 或 [ abc == abc ]

echo $? 查询结果，0是成功，非0是失败

test abc == xyz 或 [ abc == xyz ]

echo $? 查询结果

[ root == $USER ] 判断当前用户是否为root

[ root != $USER ] 判断当前用户是否不为root

[ abc == "$o" ] 在使用一个空变量的时候会出现语法报错，加双引号可以避免

[ -z $o ] 判断变量o是否为空

[ ! -z $o ] 判断变量o是否非空

1. 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

[ 100 -eq 100 ] 判断100是否等于100

[ 100 -ne 100 ] 判断100是否不等于100

1. 逻辑组合

&& 之前任务成功才执行之后任务

|| 之前任务失败才执行之后任务

[ 0 -eq $UID ] && echo ok 如果当前用户是管理就执行echo命令

[ 0 -eq $UID ] || exit 如果当前用户不是管理员就执行exit命令

如果使用2个逻辑符号，那么第2个逻辑符号后面的任务是否执行，需要依靠第一个逻辑符号来判断

如果是&&，那么两边的任务都成功才算成功

如果是||， 那么两边的任务只要一个成功就算成功

[ 0 -ne $UID ] && echo "你不是管理员，无法执行该脚本" && exit 如果是管理则不会有信息提示，如果不是管理员则出现信息提示并退出脚本

1. 文件

-e 判断是否存在，不关心文件类型

-f 判断是否存在，必须是普通文件

-d 判断是否存在，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执行权限

== != -z ! -z -eq -ne -gt -ge -lt -le && || -e -f -d -r -w -x

练习三：if语句

1. 单分支

if 条件测试;then 当条件测试成功

执行指令 才会执行此处指令，指令可以有很多

fi

1. 双分支

if 条件测试;then

执行指令1

else

执行指令2

fi

ping -c 指定ping的次数 -i 指定ping间隔时间(秒) -W 当目标主机不通时，多久反馈信息

利用双分支编写脚本，实现查看ping结果

#!/bin/bash

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

if [ $? -eq 0 ];then

echo "通了"

else

echo "不通"

fi

1. 多分支

if 条件测试1;then

执行指令1

elif 条件测试2;then

执行指令2

else

执行指令n

fi

#!/bin/bash

read -p "请输入期末考试成绩：" n

if [ $n -ge 90 ];then

echo "优秀！"

elif [ $n -ge 80 ];then

echo "良好！"

elif [ $n -ge 60 ];then

echo "加油！"

else

echo "晚上没饭了！"

fi

练习四：循环，当某任务需要反复执行时可以用到的工具

1，for循环 可以定义循环次数，有限的循环

for 变量名 in 值1 值2 值3 。。。。。

do

循环任务

done

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

#!/bin/bash

for i in {1..10} 循环10次的便捷写法

do

echo nb

echo $i

done

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

#!/bin/bash

a=10

for i in `seq $a` 支持变量的多次循环写法

do

echo nb

echo $i

done

编写脚本，使for循环嵌套if语句实现对主机的批量ping测试

#!/bin/bash

for i in {1..10}

do

ping -c 3 -i 0.2 -W 1 192.168.4.$i &> /dev/null

if [ $? -eq 0 ];then

echo "192.168.4.$i 通了"

else

echo "192.168.4.$i 不通"

fi

done

2，while 循环 可以实现无限循环

while 条件测试

do

循环任务

done

#!/bin/bash

while : 使用冒号可以实现无限循环

do

echo abc

sleep 0.1 无限循环的脚本可能会大量消耗cpu资源，导致服务器崩溃，可以使用sleep命令缓解

done