回顾

条件测试

== != -z ! -z

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

&& ||

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

if语句

单分支、双分支、多分支

循环

for

while

练习一：编写猜数脚本

#!/bin/bash

y=0

x=$[RANDOM%10]

while :

do

let y++

read -p "请输入一个数字(0-9)" n

if [ $x -eq $n ];then

echo "恭喜，猜对了！你猜了$y次"

exit

elif [ $x -lt $n ];then

echo "猜大了！"

else

echo "猜小了！"

fi

done

练习二：使用case分支

1，case基本使用

类似if，但不如if强大，但语句比if精简

case 变量 in

模式1)

执行指令;;

模式2)

执行指令;;

模式N)

执行指令;;

\*)

执行指令

esac

2，编写部署nginx服务的脚本

#!/bin/bash

tar -xf nginx-1.12.2.tar.gz

cd nginx-1.12.2

yum -y install gcc pcre-devel openssl-devel

./configure

make

make install

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

systemctl stop httpd 关闭httpd服务，避免端口冲突

/usr/local/nginx/sbin/nginx 运行nginx服务

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

netstat -ntulp | grep nginx 查询nginx服务状态

netstat命令可以查看系统中启动的端口信息，该命令常用选项如下：

-n以数字格式显示端口号

-t显示TCP连接的端口

-u显示UDP连接的端口

-l显示服务正在监听的端口信息，如httpd启动后，会一直监听80端口

-p显示监听端口的服务名称是什么（也就是程序名称）

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

systemctl stop firewalld 临时关闭防火墙

然后在真机使用浏览器访问linux虚拟机，可以看到nginx欢迎界面

/usr/local/nginx/sbin/nginx -s stop 关闭nginx服务

编写方便控制nginx服务的脚本

#!/bin/bash

case $1 in

start|kai)

/usr/local/nginx/sbin/nginx;;

stop|guan)

/usr/local/nginx/sbin/nginx -s stop;;

restart|cq)

/usr/local/nginx/sbin/nginx -s stop

/usr/local/nginx/sbin/nginx;;

\*)

echo "start|kai stop|guan restart|cq"

esac

再经过优化，grep -q 也可以让查询的信息不输出，这里可以实现与&> /dev/null相同效果

#!/bin/bash

netstat -ntulp | grep nginx &> /dev/null

x=$?

case $1 in

start|kai)

[ $x -eq 0 ] && echo "服务正在运行..." && exit

/usr/local/nginx/sbin/nginx;;

stop|guan)

[ $x -ne 0 ] && echo "服务已经关闭..." && exit

/usr/local/nginx/sbin/nginx -s stop;;

restart|cq)

[ $x -ne 0 ] && /usr/local/nginx/sbin/nginx && exit

/usr/local/nginx/sbin/nginx -s stop

/usr/local/nginx/sbin/nginx;;

status)

[ $x -eq 0 ] && echo "服务已开启" || echo "服务未开启";;

\*)

echo "start|kai stop|guan restart|cq"

esac

echo -e "\033[32mABCD\033[0m"

练习三，函数，可以将公共的语句块定义成函数名称，方便反复重复调用，达到精简脚本的目的

#!/bin/bash

a(){ 定义函数

echo -e "\033[$1m$2\033[0m"

}

a 31 ABCD 调用函数，31对应函数中$1，ABCD对应函数中的$2

a 32 WERW 可以反复调用

使用函数编写高效率ping工具

#!/bin/bash

mping(){

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

if [ $? -eq 0 ];then

echo "$1 通了"

else

echo "$1 不通"

fi

}

for i in {1..254}

do

mping 192.168.4.$i &

done

wait

这里的&符合是可以将命令扔到后台运行，然后立刻开始下一次for循环，wait的作用是等待那些扔到后台的任务都执行完毕后再退出脚本

练习四：

循环的控制

exit 可以终止循环，但整个脚本也被终止

break 终止循环，继续运行循环之后的任务

continue 终止当前循环，继续下一次循环

编写脚本，可以帮助用户进行整数求和，如果用户输入的是0，则结束求和，并输出结果

#!/bin/bash

x=0

while :

do

read -p "请输入一个整数求和（0是结束）" n

[ -z $n ] && continue 当给的数字是空，就从来

[ $n -eq 0 ] && break 当给的数字是0，就跳出循环

let x+=n 每次将给的数字与x相加

done

echo "总和是$x"

练习五：字符串的处理

1. 字符串的截取

编写脚本，实现获取10个8位长度的随机字符串

${变量的名称:截取的位置:截取的长度}

#!/bin/bash

x=abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789

for j in {1..10}

do

a=

for i in {1..8}

do

n=$[RANDOM%62]

p=${x:n:1}

a=$a$p

done

echo $a

done

2, 字符串的替换

a=123

[root@svr7 opt]# echo ${a/2/8} 替换1个2为8

183

[root@svr7 opt]# a=112233

[root@svr7 opt]# echo ${a//2/8} 使用两个/可以替换所有2为8