shell day 03

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

**一，while循环，可以根据条件决定循环次数，也可以实现无限循环**

while 条件测试 //根据条件的结果决定是否要执行任务，条件测试成

功的话就执行，如果失败立刻结束循环

do

任务

done

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

#!/bin/bash

while : //冒号代表永远正确

do

echo abc

sleep 0.1 //休息0.1秒

done

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

#!/bin/bash

n=10

while [ $n -ge 5 ] //可以根据条件决定是否要循环

do

echo abc

sleep 0.1

let n-- //每次循环将n-1

done

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

**二，case分支，功能类似if，编写时语句比if精简**

case 调用的变量名 in

模式1)

指令;;

模式2)

指令;;

\*)

指令

esac

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

#!/bin/bash

case $1 in

t) //如果$1是t就执行touch任务

touch $2;;

m) //如果$1是m就执行mkdir任务

mkdir $2;;

r) //如果$1是r就执行rm任务

rm -rf $2;;

\*)

echo "请输入t或者m或者r"

esac

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

**nginx 是搭建网站的服务**

**编写源码安装nginx脚本：**

tar -xf lnmp\_soft.tar.gz //在家目录释放软件包，这个软件包是从真机获取的

cp lnmp\_soft/nginx-1.17.6.tar.gz /opt //拷贝nginx到opt下

cd /opt //回opt写下列脚本

#!/bin/bash

yum -y install gcc make pcre-devel openssl-devel //安装依赖软件

tar -xf nginx-1.17.6.tar.gz //释放nginx的tar包

cd nginx-1.17.6 //进入目录

./configure //配置

make //编译

make install //安装

脚本写完后，检查yum必须可用！

运行完脚本后，查看 ls /usr/local/nginx 目录可以看到

4个目录则成功 conf html logs sbin

systemctl stop httpd //关闭httpd避免端口冲突

/usr/local/nginx/sbin/nginx //开启nginx服务

systemctl stop firewalld //关闭防火墙

使用浏览器访问虚拟机的网站页面<http://192.168.2.5/>

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

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

netstat 命令可以查询系统启动的端口信息

-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）

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

**使用case分支编写控制nginx服务的脚本：**

#!/bin/bash

case $1 in

start|kai) //输入start或者kai 就开启nginx

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

stop|STOP|s|guan) //输入stop或STOP或s或guan就关闭nginx

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

re|cq) //输入re或者cq就重启nginx

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

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

cx|status) //输入cx或者status就查询nginx状态

netstat -ntulp | grep -q nginx //查询nginx，但不输出结果 -q选项是保持沉默

[ $? -eq 0 ] && echo "nginx正在运行" || echo "nginx未开启";; //判断，如果上述

任务成功，说明nginx服务开启，就显示"nginx正在运行"，如果上述任务失败，说

名nginx服务关闭，就显示"nginx未开启"

\*)

echo "请输入start或stop"

esac

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

echo -e "\033[34mABCD\033[0m" //使用echo输出不同颜色的内容

**三，函数，可以利用一个名称存储公共的语句块，实现**

**精简脚本方便后期调用的目的**

函数名(){

指令

}

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

a(){ //创建函数

echo abc

echo xyz

}

a //调用函数

a //可以反复调用函数

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

使用函数修改字符颜色

#!/bin/bash

a(){ //创建函数，名字叫a

echo -e "\033[$1m$2\033[0m" //输出带颜色的字符

}

a 31 ABCD //调用函数时，后面跟第一个位置变量定义颜色

第二个位置变量定义内容

a 32 xyzZ

a 33 EGSD

a 34 ERTY

a 35 BJYQ

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

**利用函数完善之前的脚本，使输出信息时有不同颜色**

#!/bin/bash

a(){

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

}

case $1 in

start|kai)

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

stop|STOP|s|guan)

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

re|cq)

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

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

cx|status)

netstat -ntulp | grep -q nginx

[ $? -eq 0 ] && a 32 "nginx正在运行" || a 31 "nginx未开启";;

\*)

echo "请输入start或stop"

esac

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

**四，循环的控制**

**exit 可以终止循环，但脚本也终止**

**break 可以终止循环，继续循环后的任务**

**continue 可以终止当前循环，继续下一次循环**

**编写脚本，帮用户进行整数求和，如果用户输入0**

**就结束求和，并输出结果**

#!/bin/bash

x=0

while : //while循环后面写冒号代表永远正确可以无限循环

do

read -p "请输入一个整数求和(0是结束并输出结果):" n

[ -z $n ] && continue //如果n是空值则重新进行循环任务

[ $n -eq 0 ] && break //如果n是0则退出循环执行循环后任务

let x+=n //不断的将n的值保存在x里

done

echo "总和是$x"

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

**五，字符串的处理**

1. **字符串的截取**

**${变量名称:截取位置:截取长度}**

[root@proxy opt]# a=abcd

[root@proxy opt]# echo ${a:1:2} //从第二位截取两位

bc

[root@proxy opt]# echo ${a:0:2} //从头截取两位

ab

**编写脚本，可以生成8位随机字符用作密码**

#!/bin/bash

x=abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789

for i in {1..8}

do

n=$[RANDOM%62] //得到0~61随机数存在变量n中

a=${x:n:1} //截取到1个随机字符，存在变量a中

c=$a$c //不断往c中追加，此时c相当于一个袋子

done

echo $c //最后喊出，得到8位长度随机字符串

1. **字符串替换**

**${变量名/旧/新}**

a=1234

echo ${a/2/6} //将1个字符2换成6

a=112233

echo ${a/2/6} //将1个字符2换成6

echo ${a//2/6} //将所有字符2换成6

a=1122233

echo ${a/22/66} //换2个

echo ${a/11/} //将11替换成空，相当于删除

1. **字符串的删除，可以删除两边**

${变量名#要删除的内容} 掐头

${变量名%要删除的内容} 去尾

a=abcdef //创建变量，作为素材

echo ${a#abc} //掐头,删除到abc

echo ${a%def} //去尾,删除到def

echo ${a#abcde} //掐头,删除到abcde

a=abcdefghijk //创建变量，作为素材

echo ${a#abcdefghi} //掐头,删除到i

echo ${a#\*i} //效果同上，精简写法

echo ${a%defghijk} //去尾,删除到d

echo ${a%d\*} //效果同上，精简写法

**编写脚本，可以批量修改扩展名**

#!/bin/bash

for i in $(ls \*.txt) //找到所有的txt文件交给for循环

do

n=${i%.\*} //用去尾的方法删除扩展名

mv $i $n.doc //再将源文件扩展名修改为doc

done

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

练习：

1，简述case分支结构。

2，简述定义一个Shell函数的方法。

3，简述Shell环境常见的中断及退出控制指令。

4，使用 while 循环,统计 1+2+3+4...+100的结果。

提示:可以用一个独立的变量a存放求和的值。

5，编写脚本,通过 3 个 read 命令读取用户输入的三个任意数字,脚本对输入的三个数字求和输出。

6， 编写脚本,测试当前用户对/etc/passwd 文件是否具有读、写、执行的权限,让脚本执行结果类似下面的效果。

参考答案：

1，简述case分支结构。

case 变量名 in

模式1)

执行指令 ;;

模式2)

执行指令 ;;

\*)

执行指令

esac

2，简述定义一个Shell函数的方法。

函数名() {

执行指令

}

3，简述Shell环境常见的中断及退出控制指令。

break：跳出当前所在的循环，执行循环之后的语句。

continue:跳过循环内余下的语句，执行下一次循环。

exit:退出脚本

4，使用 for 循环,统计 1+2+3+4...+100的结果。

提示:可以用一个独立的变量a存放求和的值。

#!/bin/bash

a=0

for i in {1..100}

do

let a+=i

done

echo $a

5，编写脚本,通过 3 个 read 命令读取用户输入的三个任意数字,脚本对输入的三个数字求和输出。

#!/bin/bash

read -p "请输入数字" num1

read -p "请输入数字" num2

read -p "请输入数字" num3

echo $[num1+num2+num3]

6， 编写脚本,测试当前用户对/etc/passwd 文件是否具有读、写、执行的权限,让脚本执行结果类似下面的效果。

当前用户对/etc/passwd 有读权限

当前用户对/etc/passwd 有写权限

当前用户对/etc/passwd 无执行权限

#!/bin/bash

file=/etc/passwd

[ -r /etc/passwd ] && echo "当前用户对$file 有读权限" || echo "当前用户对$file 没有读权限"

[ -w /etc/passwd ] && echo "当前用户对$file 有写权限" || echo "当前用户对$file 没有写权限"

[ -x /etc/passwd ] && echo "当前用户对$file 有执行权限" || echo "当前用户对$file 没有执行权限"