shell day 03

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

**while循环，可以根据条件测试决定循环次数，可以实现无限循环**

**语法结构：**

while 条件测试 **//如果条件测试成功，就执行下面的指令，然后再回**

**来继续看条件测试能否成功，如果成功就继续执行指令，且可以实现无限**

**循环，一旦发现条件测试失败了，就立刻终止循环**

do

执行指令

done

**实例：**

#!/bin/bash

while [ 1 -eq 1 ]

do

echo abc

sleep 0.1 //如果系统执行任务消耗cpu比较多，可以每次稍微休息0.1秒

done

**另外如果想故意创造一个无限循环可以在while后面写冒号即可**

while : //冒号代表测试永远正确

do

echo abc //循环任务

sleep 0.1 //如果系统执行任务消耗cpu比较多，可以每次稍微休息一下

done

**练习：**

#!/bin/bash

x=1

while [ $x -le 5 ] //通过条件测试还可以决定执行任务的次数

do

echo abc

sleep 0.1

let x++ //将x+1

done

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

**循环的嵌套**

#!/bin/bash

for i in a b

do

for j in x y

do

echo $i

echo $j

done

done

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

**case分支，功能类似if，不如if强大，语句比if精简**

**基本语法格式：**

case 调用变量名 in //如果调用的变量内容与下面某个模式一致，就执行模式下面的指令

模式1) //这里的模式可以有很多

执行指令;; //指令需要用双分号结尾，如果一个模式有多个指令，那只需在该模式的最后一条指令后加双分号即可

模式2)

执行指令;;

…

\*) //如果上述模式都没有被匹配，那就匹配这个ZZ

执行指令

esac

**实例：**

#!/bin/bash

case $1 in

a)

echo aaa;;

b)

echo bbb;;

\*)

echo "请输入a或者b"

esac

**练习：**

**测试case分支的实际应用**

#!/bin/bash

case $1 in //使用执行脚本后的第1个位置变量作为匹配对象

t|T|tt) //如果$1是t或者T或者tt，都可以算匹配

touch $2;; //此处是创建文件的命令，后面是第二个位置变量的参数

m)

mkdir $2;;

r)

rm -rf $2;;

\*)

echo "t|m|r"

esac

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

部署nginx服务，**httpd是之前使用过的网站服务，除此之外nginx也可以实现搭建网站的任务**

1. **将lnmp\_soft.tar.gz软件包从真实主机拖拽到虚拟机的管理员家**

**目录，然后释放**

[root@svr7 ~]# tar -xf lnmp\_soft.tar.gz //释放到当前目录

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

cd /opt

**编写成部署nginx服务的脚本**

#!/bin/bash

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

tar -xf nginx-1.17.6.tar.gz //释放tar包

cd nginx-1.17.6 //进入nginx目录

./configure //配置

make //编译

make install //安装

上述脚本运行完毕之后

systemctl stop httpd //关闭httpd

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

systemctl stop firewalld //关闭防火墙

使用真实主机的火狐浏览器打开192.168.4.7可以看到nginx的

欢迎界面

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

**安装nginx脚本的思路：**

1. 安装依赖包
2. 释放nginx的tar包
3. 进入释放后的目录
4. ./configure
5. make
6. make install

**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）

netstat -ntulp | grep nginx //查询nginx服务是否开启

yum -y install psmisc //安装killall软件包

killall nginx //如果nginx服务异常不受控制，可以杀死重新开启

#!/bin/bash

case $1 in //使用执行脚本后的第1个位置变量作为匹配对象

s|start|kai) //如果$1是s或start或kai ，那么就执行以下指令

netstat -ntulp | grep -q nginx //查询有没有开启nginx服务, -q是不输出查询结果

[ $? -eq 0 ] && echo "nginx已经开启" && exit //判断如果开了nginx就退出

/usr/local/nginx/sbin/nginx;; //如果没开nginx就开启

stop|guan) //如果$1是stop或guan，那么就执行以下指令：

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

restart|cq) //如果$1是restart或cq，那么就执行以下指令

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

/usr/local/nginx/sbin/nginx;; //开启nginx，此处相当于重启nginx

status|zt) //如果$1是status或zt，那么就执行以下指令

netstat -ntulp | grep -q nginx //查询有没有开启nginx服务, -q是不输出查询结果

[ $? -eq 0 ] && echo "nginx正在运行。。" || echo "ngxin未开启";; //根据查询结果输出nginx正在运行或者nginx未开启的提示

\*)

echo "请输入start|stop|restart" //如果没有匹配任何模式就是喊出使用该脚本的提示，告诉使用者$1应该敲啥，而不能随意敲。

esac //结尾，固定语法，不能少

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

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

**函数，可以将公共的语句块用一个函数名定义，实现精简**

**脚本且方便后期调用的效果，还增加了可读性**

#!/bin/bash

a() { //定义函数

echo abc

echo xyz

}

a //调用函数，相当于执行上述两个echo任务

a //可以反复调用

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

#!/bin/bash

a() { //定义函数

echo -e "\033[$1m$2\033[0m" //输出不同颜色的文本内容，并加入位置变量

}

a 31 ABCD //调用时函数后面可以写位置变量的

内容，31就是$1 ABCD是$2

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

**将函数与修改颜色的语句整合到之前的脚本中**

#!/bin/bash

a(){ //首先定义函数

echo -e "\033[$1m$2\033[0m" //函数内容是方便输出不同内容不同颜色的字符

}

case $1 in

s|start|kai)

netstat -ntulp | grep -q nginx

[ $? -eq 0 ] && echo "nginx已经开启" && exit

/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;;

status|zt)

netstat -ntulp | grep -q nginx

[ $? -eq 0 ] && a 32 "nginx正在运行。。" || a 31 "ngxin未开>启";; //调用函数

\*)

echo "请输入start|stop|restart"

esac

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

**循环的控制：**

**exit 可以终止循环，但同时也终止整个脚本**

**break 可以终止循环，继续循环之后的任务**

**continue 可以终止当前循环，继续下一次循环**

**编写脚本，可以为用户进行整数求和，如果用户输入**

**0则终止脚本并输出之前整数之和**

#!/bin/bash

x=0

while :

do

read -p "请输入整数求和(0是结束并输出结果)" n

[ -z $n ] && continue //如果n是空值，就重新循环

[ $n -eq 0 ] && break //如果n是0，就退出循环

let x+=n //将x+n

done

echo "整数之和是$x"

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

字符串的处理

1. 字符串的截取

${变量名:截取的起始位置:截取的长度}

a=abcdef //创建变量，作为截取素材

echo ${a:1:2} //截取bc，位置是从0开始计算，所以要从1开始

才能截取第二个字符b

echo ${a:1:1} //从第2个字符截取，截取1位

下课休息 17:25回