Linux系统中权限，默认是三个rwx（读、写、执行）权限对象分为ugo（拥有者、拥有组和其他人），权限还有特殊权限位，sst（setUID、setGID、粘滞位），如果s出现在u的权限位称为setUID，如果出现在g的权限位称为setGID，如果在o的权限位出现t称为粘滞位

set权限代表在执行的时候是临时的获得了拥有者或拥有组的身份

ifconfig eth0 可以查看网卡的IP地址信息

ifconfig eth0 1.1.1.1 可以临时修改网卡的IP地址，重启网络失效，这个命令，只有管理员有权限修改IP，普通用户只能查看

chmod u+s `which ifconfig` 管理员为ifconfig命令增加一个setUID权限位，任何人在执行ifconfig命令的时候都是临时获得了管理员的身份，普通用户执行ifconfig一样可以修改IP地址

setUID权限属于一种变相的管理员管理权限的下放，允许普通用户以管理员的身份执行某些特定命令，系统中默认passwd命令有该配置项，sudo也是一种管理员管理权限下放的方法允许普通用户以root身份执行某些命令，setUID不能控制某些用户某些权限，一旦设置了setUID权限，任何可以执行该命令的用户都是以root身份执行，sudo可以方便控制某些用户某些权限，权限的管理建议使用sudo方式

setGID一般对文件夹做，可以统一文件夹下创建的子对象的属组，任何人在该目录下创建的文件或文件夹属组统一为父目录的组

chmod g+s 文件夹名称

t粘滞位，为了防止用户恶意删除，一般也是文件夹上配置，配置了t权限的文件夹只允许拥有者和root删除，其他用户不允许删除

chmod o+t 文件夹名称

查找系统中新增set位权限的文件或文件夹（set位权限比较危险，可能会导致系统出现异常）

find / -perm /6000 寻找当前系统中有set位的文件或文件夹，6000就是setUID（4）或setGID（2）

先建立一份基准文件（含有正常情况下有set位文件或文件夹）

find / -perm /6000 > setID.txt 当前系统中的set位文件或文件夹，建立基准文件

vim setID.sh

#!/bin/bash

#查询系统中新增的set位权限的文件或文件夹

for i in `find / -perm /6000 2> /dev/null`

do

grep ^$i$ setID.txt &> /dev/null

[ $? -ne 0 ] && echo "系统中新增set位权限的是: `ls -ld $i`"

done

chmod +x setID.sh

./setID.sh 默认不会显示任何内容，没有新增set位文件

touch a;chmod u+s a

./setID.sh 这时候会显示新增的set权限的文件了

while循环，当条件的成立不停的执行直到条件不成立为止，可以按照某一个规律性执行

使用while循环求和1--100（之前有for循环）

vim num-while.sh

#!/bin/bash

#这是使用while循环求和1--100

i=1

sum=0

while [ $i -le 100 ] -le是小于等于，如果是lt是小于

do

sum=`expr $sum + $i`

i=`expr $i + 1` 如果改为expr $i +2代表只计算奇数的求和

done

echo $sum

chmod +x num-while.sh

./num-while.sh

猜数字小游戏脚本：

vim caishuzi.sh

#!/bin/bash

#这是一个猜数字的脚本，提示用户输入数字，根据用户输入的数字来判断大小进行输出

NM=`echo $RANDOM | cut -c-2`

num=1 统计战绩，猜测的次数

while : while没有写条件直接使用:代表条件永远为真，不停执行

do

read -p "请输入一个数字（1--100）:" nm

if [ $nm -lt $NM ]

then

echo "数字太小了..."

elif [ $nm -gt $NM ]

then

echo "数字太大了..."

else

echo "恭喜你，答对了..."

echo "你尝试的次数:$num"

fi

num=`expr $num + 1` 每循环一次增加1

done

99乘法表脚本使用while循环即可

#!/bin/bash

#这是一个while循环写9x9乘法表脚本

a=1

while [ $a -le 9 ]

do

b=1

while [ $b -le $a ]

do

echo -n "${b}\*${a}=`expr $a \\* $b` " -n不换行，默认情况下每一个循环就是单独一行

if [ $b -eq $a ]

then

echo -e '\n' 屏幕输出换行符，一旦b和a一样大直接换行

fi

b=`expr $b + 1`

done

a=`expr $a + 1`

done

case语句，不同分支执行不同任务，系统中很多服务启动脚本都是case语句编写

vim key.sh

#!/bin/bash

#这是一个判断用户输入字符类型的脚本

read -p "请输入一个字符:" KEY

case $KEY in

[a-z]|[A-Z])

echo "你所输入的是一个字母类型"

;; 每一个分支必须有;;代表结束判断，没有;;会继续后面判断

[0-9])

echo "你所输入的是一个数字类型"

;;

\*)

echo "输入错误，你所输入的既不是数字也不是字母"

;;

esac

系统中/etc/init.d目录有有大量服务启动脚本，服务启动脚本基本上都含有case语句，通过case语句来进行服务启动的判断，如果以后需要安装源码包的软件，但是希望能够像rpm软件包一样执行，可以手动写脚本使用case语句放置在/etc/init.d目录下即可

脚本中可以使用函数来减少重复代码量，可以将重复性用到的代码直接写成函数多次调用即可

vim fun.sh

#!/bin/bash

#这是一个函数样例脚本

function show\_system {

echo "当前系统时间为:`date`"

echo "当前登录用户为:$USER"

}

show\_system

show\_system

show\_system

break是跳出循环，当循环在执行过程中，出现某个条件可以直接结束循环，不再继续循环

vim caishuzi.sh

#!/bin/bash

#这是一个猜数字的脚本，提示用户输入数字，根据用户输入的数字来判断大小进行输出

NM=`echo $RANDOM | cut -c-2`

num=1

while :

do

read -p "请输入一个数字（1--100）:" nm

if [ $nm -lt $NM ]

then

echo "数字太小了..."

elif [ $nm -gt $NM ]

then

echo "数字太大了..."

else

echo "恭喜你，答对了..."

echo "你尝试的次数:$num"

break 一旦执行到else将会自动调用break结束循环，跳出循环

fi

num=`expr $num + 1`

done

默认break只跳出当前的循环，如果是多个循环的嵌套可以使用break 数字，来跳出多层循环

continue是跳出这个循环继续其他循环

vim con.sh

#!/bin/bash

#这是一个continue的语句

for i in `seq 1 5`

do

echo $i

if [ $i -eq 3 ]

then

continue

fi

echo $i

done

可以尝试将continue改为break再执行脚本，比较两个脚本执行差异，continue继续后面的循环，可以继续显示4和5，但是break将会结束循环，4和5将不会在屏幕上显示出来