## 变量运算

**变量运算都是整数运算，不存在小数**

shell中所有变量的赋值都是字符串，要想让数值进行运算，必须用let、expo等命令，另外，shell不能处理浮点小数，处理必须用bc计算器（但是我试了不行）

expr 命令：进行数值间和运算，并打印结果

例：

a=10

b=2

expr a+b

12

另一种运算方法：

$(())

echo $((a+b))

echo $((a-b))

echo $((a\*b))

echo $((a/b))

echo $((a%b))

echo $((a\*\*b)) a的b次方



let运算：

let 运算



let 变量=运算

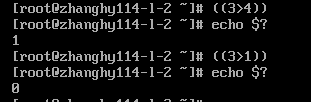
echo $变量

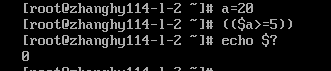
$[]运算：echo $[运算]



以及数值比较：

(()) ((1>=9)) ((2=4)) ((2<=0))





$?返回上一个执行结果，1为假，0为真

数值比较2：

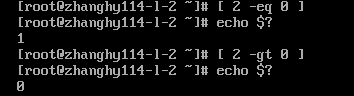
-eq equal等于

-gt great than 大于

-lt less than 小于

-ge great than and equal 大于等于

-le less than and equal 小于等于



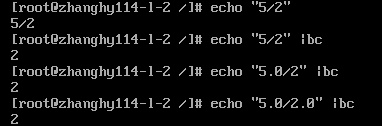
linux计算器：

bc交互式

输入bc命令进入交互式计算器界口

但是我不可以进行小数运算

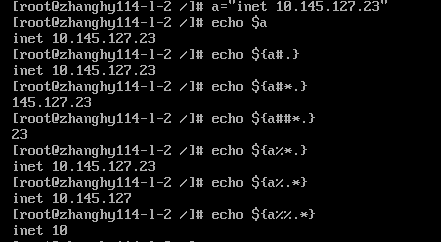
yum install bc –y(安装计算器)



## 2.变量内容的编辑

${}对内容编辑用{}

将一个变量内容进行删除打印，**但是不会改变变量内容**



echo ${a#\*.} 表示从前往后删，从第一个.之前的内容全部删除，\*代表.之前内容

echo ${a##\*.} 表示从前往后删，从最后一个.之前的内容全部删除，\*代表.之前内容

echo ${a%.\*} 表示从后往钱删，从第一个.之前的内容全部删除，\*代表.之后内容

echo ${a%%.\*} 表示从后往钱删，从后往前最后一个.之前的内容全部删除，\*代表.之后内容

打印变量的字符个数

echo ${#a}

如何切割变量内容(切片操作)

echo $(a:num:num) 第一个num表示第几个字符，第二个表示从这个字符开始有几个字符



内容替换

echo ${a/10/88} 第一个表示要替换的内容，第二个表示替换后的内容

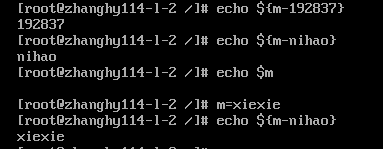


完美

变量值完全替代

echo ${a-283743}

当变量a没有值时，用283743代替，有值的话则不代替



以上操作均没有赋值或者修改变量的值，只是打印修改

接下来时无论变量有没有值，均用后面值代替

echo ${a+283743}

## 3.检测脚本运行过程

bash –vx 脚本名字

编写一个脚本，如果/root下面有app目录没有则提示没有，那么创建，



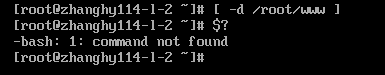
## 4.条件测试

[ 命令，这是一个命令，里面放置条件判断，所以命令之间必须用空格隔开



结束用]结束

接下来测试是否有www目录



test + 条件语句，用来测试，真返回0，假返回1

判断数值：

test 1 –eq 3

判断文件：

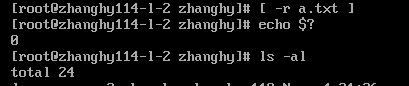
test –d 文件

判断权限：

test –r 文件 判断用户是否有读权限



[ 条件 ]可以判断也可以



### if 语句

（语句可以是条件判断[ 内容 ]，也可以是单纯的命令ls –al a.txt）

模块

if 语句；then

内容

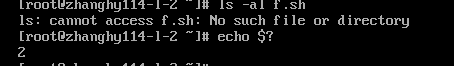
fi

例子1：

判断是否存在文件a.sh

如果不存在就创建，存在就提示已经存在，创建完记得提示已经创建完

方法1：执行命令，然后if [ $? –eq 1 ]



$?不为0即为错

方法2：if ls –al a.sh;then

## 5.文件权限

本来是要和4一起讲的，但是我觉得真的很重要，就单独拿出来讲了

什么用户创建了这个文件，这个文件的前3个就是针对这个用户，中三个是针对这个用户的用户组的，最后三个是其他用户。



可以看到用户和用户组权限是root，rw 和r。

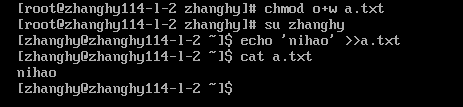
那么我们将这个文件传给其他用户



可以看到其他用户只有r读权限，试试其他用户反应？

果然只能查看，写内容或者执行（不是脚本，没必要执行）都不可以

我们修改权限为rw，让他可以读和写



很简单的权限

### 修改权限

chmod 命令

修改规则：

u代表用户

g代表用户组

o代表其他用户

u+x 用户增加可执行权限

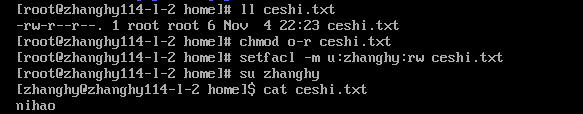
a+x 所有增加可执行权限

chmod u=rw- a.txt (将a.txt的用户权限修改为rw-)

### 2.ACL权限

当一个文件/目录，是root用户文件，我想让alice用户可以访问，但是，其他用户不能访问，这就需要增加ACL权限

cat /etc/passwd 我创建了二个用户zhanghy和zhanghy114



增加了用户zhanghy的rw权限，发现可以访问

但是切换zhanghy114，则没有权限



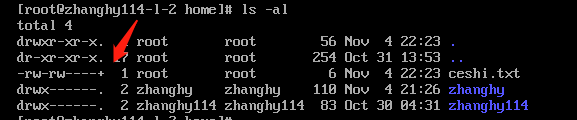
用法：

 setfacl -m u:alice:rw /home/test.txt //增加用户alice权限

setfacl –m u:用户名:权限 文件目录

 setfacl -m o::rw /home/test.txt

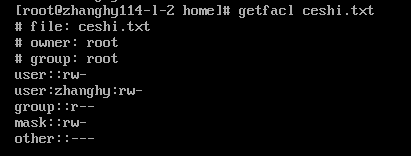
查看这个文件的ACL权限



+表示有ACL权限

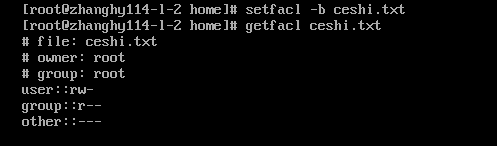
查看命令：

getfacl +文件/目录



删除所有用户（owner除外）权限

setfacl –b 文件



### 3.chattr

常用于锁定某个文件，拒绝修改

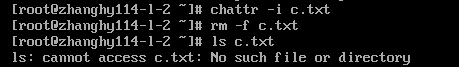
**chattr +i 文件/目录 文件无法修改，删除，添加内容，但如果是目录则无法在目录下创建和删除文件，但是可以修改其中子文件**

**chattr +a 文件 仅仅可以对文件追加内容，无法修改删除内容**

chattr +A 无法修改文件访问的时间



恢复 chattr –i –a –A等操作即可



### 4．sudo

通过sudo提权，但是需要输入root密码

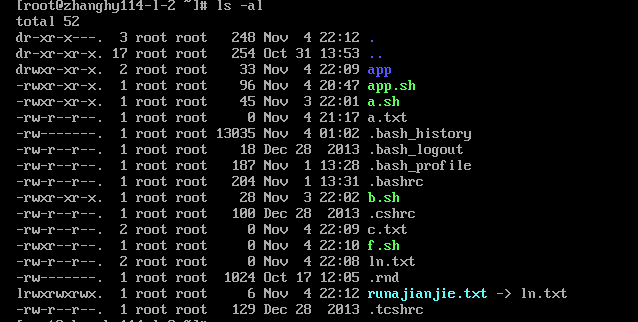
## 6.文件类型

d 目录文件（蓝色）  
b 设备文件（块设备）存储设备硬盘，U盘 /dev/sda, /dev/sda1  
c 设备文件（字符设备）打印机，终端 /dev/tty1  
l 链接文件（淡蓝色）  
s 套接字文件  
p 管道文件

脚本文件必须是可执行才有颜色（绿色）



其中app是蓝色，为目录，a.sh是脚本，因为加了可执行才有颜色，runlianjie.txt是软连接文件，其他为普通文件



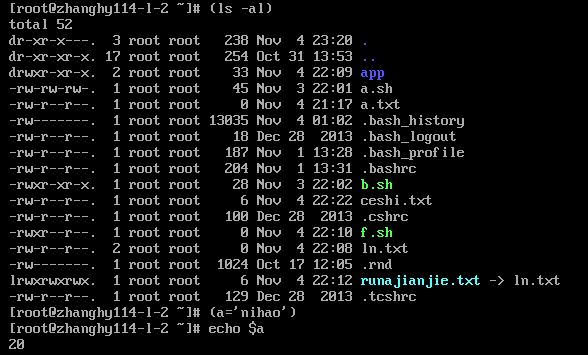
只要记住这三种颜色就够了

## 7.括号() {} []区别 （后续补充）

() {} []说真的,这三种符号快把我折磨死了

### ()

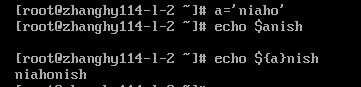
1.一般用在创建子shell时候运行一段命令，运行完关闭



2数组创建 array =(a,d,v,w)

### {}

1. 当变量打印时为了防止字符串与变量混淆



2.${}对变量内容编辑，详细见第二章

### []

1. 变量运算

echo $[运算结果]

eg: echo $[2+3]

1. shell命令

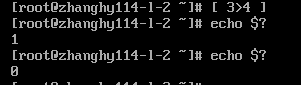
which [

用于条件判断

但是二边都要有空格，内部操作符和内部变量之间也要有空格



1. 用于数值判断

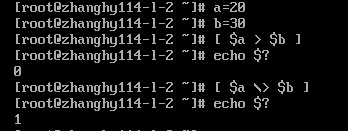




4．用于字符串比较（字符串和数值也可以）

**由于变量值是字符串**

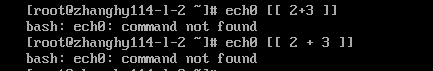
所以字符串比较也是可以的，但是一定要转义< 或>，\>和\<才可以

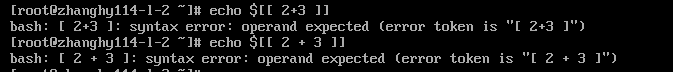


可以看见，没有转义是不可以执行的，执行也是错误，而且没有大于等于转义

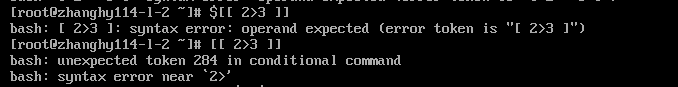
### [[]]

1.与[]区别在于，不可以进行运算

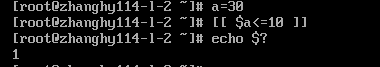




1. 他是关键字，但是也要二边都要有空格
2. 也不可以进行数值比较



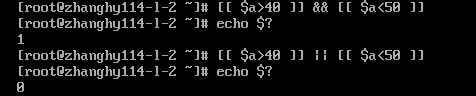
1. 但他可以字符串比较（字符串和数值），而且不用转义，而且执行小于等于都可以



1. 甚至可以进行&&和||运算，这是[]不可以实现的

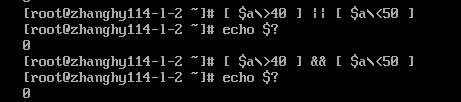
我们先测试[[]]进行逻辑运算

已知a=30



所以语句执行完美。

但是用[]语句就完了



他执行不出来结果

### (())

双小括号

用于数值运算

echo $((运算结果))



## 8.定时

定时命令是个好东西

### sleep 延迟命令

sleep 1000

1000代表1000s，表示延迟1000s

sleep 1m 表示延迟1分钟

1h 延迟1小时

配合sleep 1 就是1s

sleep 1m 延迟1分钟

1s 延迟1s

1h 延迟1小时

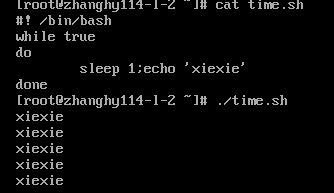
1d 延迟1天

例子1：2秒后执行一个命令：



例子2：

每秒打印一句话：



写一个while循环即可，但是true是小写，大写True不可以，和python不一样

### crontab

安装crontab

yum install cronie

crond和crontab

crond是进程，可以用ps –ef | grep crond查看

也可以关掉这个服务

systemctl stop crond

crontab则是命令

crontab –e 输入定时程序

crontab –l 列出所有定时

其中有针对系统和用户的，这里我们就针对用户吧



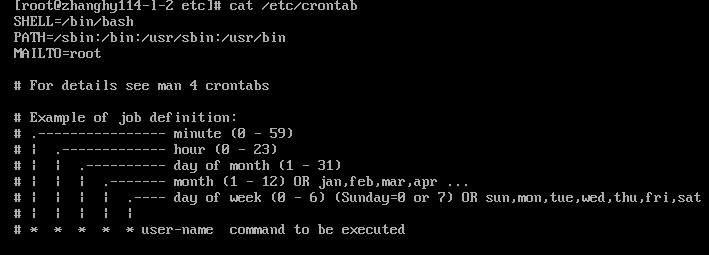


\*\*\*\*\* 命令

,用于不同时间段，比如2，3，4二点三点四点

-用于时间区，2-4，2点到4点

\n 这个用于每隔 \*\2 \*\*\*\*，每隔2分钟执行一次



eg:

1. 设置每五分钟执行一次打印内容到文本

crontab –e

加入内容：

# 5minute echo to /root.txt

\*/5 \* \* \* \* echo ‘nimabi’ >> /root/a.txt

注意：如果格式错误是无法保存的

用crontab –l 可以查看编辑的情况

2.每周五下午3点执行一次

crontab –e

加入内容：

# 说明内容

0 15 \* \* 5 echo ‘nimabi’ >> /root/a.txt

1. 每天上午8点半和下午5点半执行一次（如果是8点和5点半就要写二个定时了）

# 说明内容

30 8,17 \* \* \* echo ‘nimabi’ >> /root/a.txt

1. 每个月第一天上午8点执行命令

# 说明内容

0 8 1 1-12 \* echo ‘nimabi’ >> /root/a.txt

4.每周1和周五上午8点执行

# 说明内容

0 8 \* \* 1-5 echo ‘nimabi’ >> /root/a.txt

5.每天8点执行多条命令

# 说明内容

0 8 \* \* \* echo ‘nimabi’ >> /root/a.txt;sleep 10000

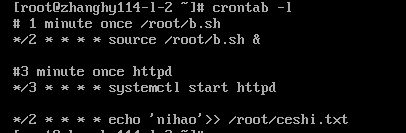
1. 每年执行一次任务

@yearly /scripts/script.sh

5.系统重启时执行任务

@reboot /scripts/script.sh

服务测试：





发现完全成功



删除定时任务：

直接在crontab -e上面删除

直接crontab –r全部清除

备份crontab文件：

.将所有的 cron 命令备份到文本文件当中  
这是一个当我们丢失了cron命令后方便快速的一个恢复方式。  
下面是利用这个方式恢复cron的一个小例子。（看看就行~）  
首先：检查当前的cron

# crontab -l

MAIL=rahul

0 2 \* \* \* /script/backup.sh

然后：备份cron到文件中

# crontab -l > cron-backup.txt

# cat cron-backup.txt

MAIL=rahul

0 2 \* \* \* /script/backup.sh

接着：移除当前的cron

# crontab –r 删除所有定时记录

# crontab -l

no crontab for root

恢复：从text file中恢复

删除定时任务：

直接 crontab –e进去删最好

crontab –r命令是删除用户所有定时

crontab的权限控制

/etc/cron.allow 和 /etc/cron.deny 文件被用来限制对 cron 的使用。

这两个控制文件的格式都是每行一个用户。两个文件都不允许有空格。

如果控制文件被修改了，cron守护进程 crond 不必重启。控制文件在每次用户添加或删除一项 cron 任务时都会被读取。

/etc/cron.allow 的优先级高于 /etc/cron.deny。

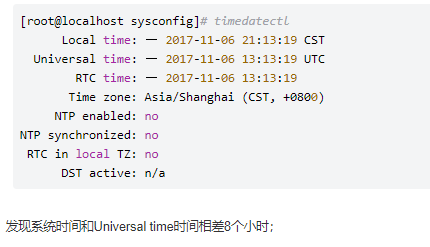
如果 cron.allow 文件存在，只有其中列出的用户才被允许使用 cron，并且cron.deny 文件会被忽略。

如果 cron.allow 文件不存在，所有在cron.deny 中列出的用户都被禁止使用 cron。

## 9.调整时间

。1.查看系统时间

timedatectl



如果不是：

yum install ntp //安装ntp服务

systemctl enable ntpd //开机启动服务

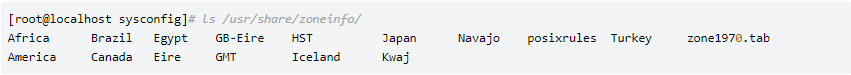
systemctl start ntpd //启动服务

timedatectl set-timezone Asia/Shanghai //更改时区

2、查看系统时区：

ls /usr/share/zoneinfo/

会发现有很多时区时间



3、删除当前系统所处的分区

rm –f /etc/localtime

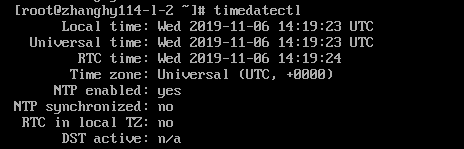
[root@localhost sysconfig]# sudo rm /etc/localtime

4、从/usr/share/zoneinfo/中创建软连接以替换当前的时区信息，直接选择Universal：

ln -s /usr/share/zoneinfo/Universal /etc/localtime

1. 再次查看时间

datetimectl



已经恢复了，设置时间对于定时来说非常重要

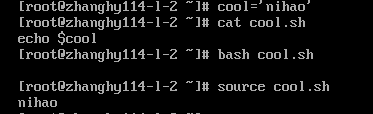
## 10．关于子shell的理解

子shell，bash运行就会自动开启子shell，后台也是开启子shell

但我们bash a.sh，其实是开启子脚本来运行这个脚本。运行完就关闭这个子shell。

可以实验，因为子shell无法继承shell变量，除非环境变量

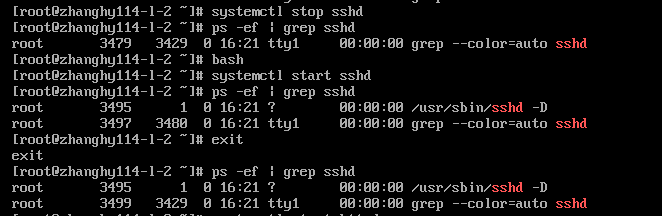
### 这个命令告诉bash \*.sh 和source \*.sh的区别



一个是需要开启子shell，一个直接运行，当作命令运行

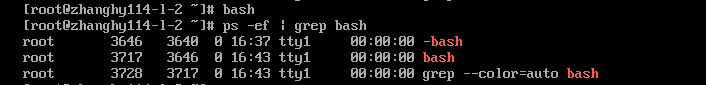
那么是否可以通过子shell来运行一个程序呢？

当然可以

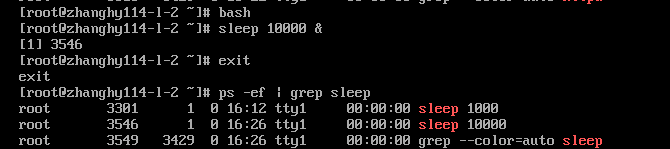


但是变量就不可以，哪怕你设置export，进入shell一样不行

查看子shell



在子shell中，可以通过运行程序到后台来让进程运行

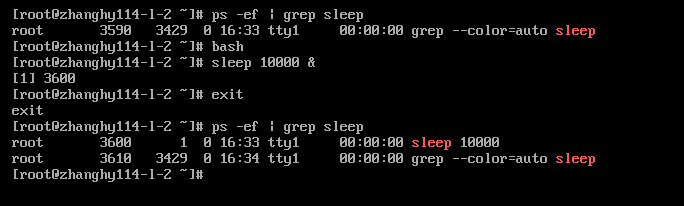


打开脚本，如果是bash脚本，里面有设置进程，那么设想，如果bash 脚本关闭脚本，会不会退出进程？

肯定会，就算在主shell命令中，你执行sleep 10000

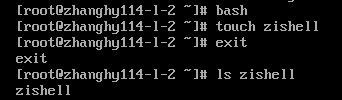
然后再ctrl +c一样会关闭sleep，除非你加入后台

所以，在编写脚本时候，如果想让进程一直运行，最好加上&，这样即使关停了脚本，也不会关停进程，因为有一个子shell在运行这个进程

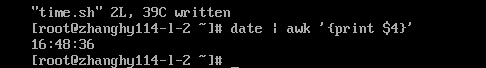


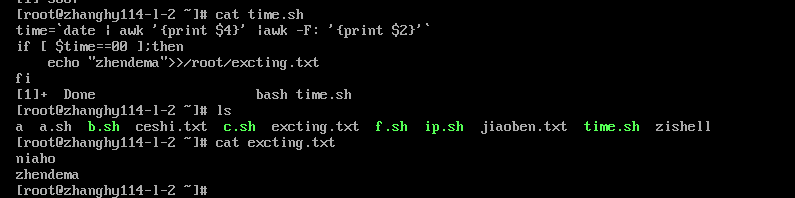
所以子shell其实可以看做是主shell，这里面操作系统文件其实都成立（linux都是文件嘛），所以在外面一样能实现

我在子shell在root下面修改创建文件，进入主shell一样成立，命令也一样，命令也是文件



### 实战：我写了一个定时脚本，在一定时间就会运行





但是要一直运行脚本

这里我用了后台运行

bash time.sh &即可一直运行脚本了

这个例子充分说明，子shell后台启动的作用