第一章 bash变量
一、变量基础
1.变量名的长度不能超过255个字符。
2.在bash中，变量的默认类型都是字符串类型。
3.bash变量的分类：
用户自定义变量：用户自定义。
环境变量：变量的名字是固定的，作用也是固定的
位置参数变量：主要向脚本传递参数的，变量名不能自定义，作用也是固定的。
预定义变量：是bash已经定义好的变量，变量名不能自定义，作用也是固定的。
二、用户自定义变量
1.定义变量：
等号两边不能有空格，否则会报错。
x=5;
x= "qwe"; 错误
注：这是由于系统命令是有空格的，这个便于区分。如果变量值中有空格，则用双引号括起来。
2.变量调用：
$变量名
例：echo $变量名
3.变量的叠加：
x ="$x"456
x=${x}789
4.变量查看
set 查看变量
-u 如果设定此选项，调用未声明变量时会报错（默认无任何提示）
5.变量的删除
unset 变量名
注：不需要加$符号。
三、环境变量
1.与自定义变量的区别：
环境变量是全局变量而自定义变量是局部变量。
环境变量在当前shell以及当前shell的子shell中都是有效的，而自定义变量只在当前shell中有效。
2.对系统生效的环境变量名和作用都是固定的。
3.设置环境变量：
export 变量名=变量值
4.查看环境变量
set ： 可以查看所有变量
env：只用来查看环境变量
5.删除环境变量
unset 变量名
注：不加$符号。
6.常用的环境变量
建议将环境变量设置为大写。

常用环境变量

- HOSTNAME：*主机名*
- SHELL：*当前的shell*
- TERM：*终端环境*
- HISTSIZE：*历史命令条数*
- SSH_CLIENT：*当前操作环境是用ssh连接的，这里记录客户端ip*
- SSH_TTY：*ssh连接的终端时pts/1*
- USER：*当前登录的用户*

PATH环境变量：
系统搜索命令路径。

PS1环境变量

- PS1变量：命令提示符设置
  - \d：显示日期，格式为“星期 月 日”
  - \H：显示完整的主机名。如默认主机名“localhost.localdomain”
  - \t：显示24小时制时间，格式为“HH:MM:SS”
  - \A：显示24小时制时间，格式为“HH:MM”
  - \u：显示当前用户名
  - \w：显示当前所在目录的完整名称
  - \W：显示当前所在目录的最后一个目录
  - \$：提示符。如果是root用户会显示提示符为“#”，如果是普通用户会显示提示符为“$”

7.语系变量
locale：查询当前的语系；
LANG:定义系统主语系的变量
LC_ALL：定义整体语系的变量
locale –a：查看所支持的所有语系
cat /etc/sysconfig/i18n 查看系统默认环境。

四、位置参数变量
1.定义：

位置参数变量	作　　用
$n	n为数字，$0代表命令本身，$1-$9代表第一到第九个参数，十以上的参数需要用大括号包含，如${10}.
$*	这个变量代表命令行中所有的参数，$*把所有的参数看成一个整体
$@	这个变量也代表命令行中所有的参数，不过$@把每个参数区分对待
$#	这个变量代表命令行中所有参数的个数

五、预定义变量

预定义变量	作　　用
$?	最后一次执行的命令的返回状态。如果这个变量的值为0，证明上一个命令正确执行；如果这个变量的值为非0（具体是哪个数，由命令自己来决定），则证明上一个命令执行不正确了。
$$	当前进程的进程号（PID）
$!	后台运行的最后一个进程的进程号（PID）

2.接受键盘输入
read [选项] [变量名]
选项：
–p 提示信息：在等待read输入时，输出提示信息
–t 秒数：指定等待的时间
–n 字符数：read只接受指定的字符数
–s 隐藏输入数据，用于机密信息的输入

第二章 运算符
一、declare命令
1.作用：用于声明变量的类型
2.declare [+/–][选项] 变量名
选项：
–：给变量设定类型属性
+：取消变量的类型属性
–a：将变量声明为数组型
–i：声明为整数型
–x：声明为环境变量
–r：声明为只读变量
–p：显示指定变量的被声明类型。
3、例：
声明整型：
declare –i cc=$aa+$bb
声明数组：
movie[0]=1
movie[1]=2
declare -a movie[2]=3

用不用declare都可以声明数组。
查看数组：
${movie}：不加调用数组的第一个值
${movie[1]}
${movie[*]}列出数组中所有内容
声明环境变量：
declare –x a=1
export实际上是declare–x
声明变量是只读属性：
只读属性会让变量不能修改，不能删除，甚至不能取消只读属性。
二、数值运算
1.declare声明数值型变量进行运算。
2.expr或let数值运算工具
dd=$(expr $aa + $bb)
注：+两侧必须有空格
3.“$((运算符))”或$[运算符]
注：$(命令)，先运算括号里面的命令，再将值赋予变量。
gg=$(date)
三、运算符

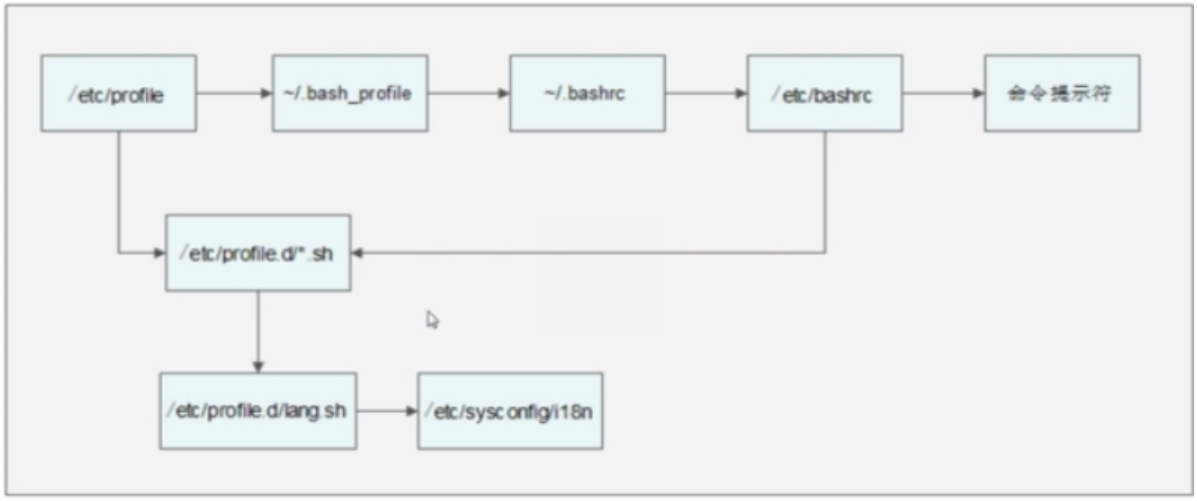
优先级	运算符	说明
13	-，+	单目负、单目正
12	!，~	逻辑非、按位取反或补码
11	*，/，%	乘、除、取模
10	+，-	加、减
9	<<，>>	按位左移、按位右移
8	<=，>=，<，>	小于或等于、大于或等于、小于、大于
7	==，!=	等于、不等于
6	&	按位与
5	^	按位异或
4	|	按位或
3	&&	逻辑与
2	||	逻辑或
1	=，+=，-=，*=，/=，%=，&=，^=，|=，<<=，>>=	赋值、运算且赋值

四、变量测试

变量置换方式	变量y没有设置	变量y为空值	变量y设置值
x=${y-新值}	x=新值	x为空	x=$y
x=${y:-新值}	x=新值	x=新值	x=$y
x=${y+新值}	x为空	x=新值	x=新值
x=${y:+新值}	x为空	x为空	x=新值
x=${y=新值}	x=新值 y=新值	x为空 y值不变	x=$y y值不变
x=${y:=新值}	x=新值 y=新值	x=新值 y=新值	x=$y y值不变
x=${y?新值}	新值输出到标准错误输出（就是屏幕）	x为空	x=$y
x=${y:?新值}	新值输出到标准错误输出	新值输出到标准错误输出	x=$y

1.变量测试在脚本优化中使用。
测试x=${y-新值}

第三章 环境变量配置文件
1.source命令：
source 配置文件名 或 . 配置文件名
2.环境变量配置文件简介：主要定义对操作系统环境生效的系统默认环境变量，如PATH等。
3.常用的登录时起作用的环境变量配置文件：
/etc/profile
/etc/profile.d/*.sh
~/.bash_profile
~/.bashrc
/etc/bashrc



登录过程分为两种：
第一种是输入用户名和密码：先通过/etc/profile 加载 /etc/profile.d/*.sh 加载lang.sh和i18n 接着加载~/.bash_profile
~/.bashrc
/etc/bashrc 登录成功。
第二种是在一个用户下切换到另一个用户，不需要输入用户名和密码：
先通过/etc/bashrc 加载/etc/profile.d/*.sh 加载lang.sh和i18n 登录成功。

4./etc/profile
a.作用：

- USER变量：
- LOGNAME变量：
- MAIL变量：
- PATH变量：
- HOSTNAME变量：
- HISTSIZE变量：
- umask：
- 调用/etc/profile.d/*.sh文件

注：umask：
查看系统默认权限
文件最高权限是666；
目录最高权限是777；
权限不能使用数字进行切换，必须使用字母。

umask是系统权限中准备丢弃的权限。

5.~/.bash_profile
a.作用：调用了~/.bashrc文件
在PATH变量后面加入了“:$HOME/bin”这个目录
6.~/.bashrc
a.作用：
变量别名
7./etc/bashrc
a.作用：PS1变量，umask，PATH变量，调用/etc/profile.d/*.sh

8.其他环境变量配置文件
a.注销时生效的环境变量配置文件
~/.bash_logout
b.~/.bash_history 历史命令是存放在硬盘上的。
用history查看的命令，是保存在内存中的，不会即时写入文件，待系统用户退出时，会将命令写入文件。
c.终端欢迎信息
/etc/issue

转义符	作　　用
\d	显示当前系统日期
\s	显示操作系统名称
\l	显示登录的终端号，这个比较常用。
\m	显示硬件体系结构，如i386、i686等
\n	显示主机名
\o	显示域名
\r	显示内核版本
\t	显示当前系统时间
\u	显示当前登录用户的序列号

注：上述文件只对本地终端起作用，对远程的是不起作用的。
远程终端欢迎信息：/etc/issue.net
注：转义符在/etc/issue.net文件中不能使用
是否显示此欢迎信息，由ssh的配置文件/etc/ssh/sshd_config决定，加入"Banner /etc/issue.net"才能显示

d.登录后显示欢迎信息：/etc/motd
这个文件不管是本地终端还是远程终端都可以显示。

第四章 正则表达式
1.定义：主要是用来对字符串进行分割、匹配、查找及替换操作。
2.通配符：
  –* 匹配任意内容
  –? 匹配任意一个内容
  –[] 匹配括号中的一个字符
通配符用来匹配符合条件的文件名，通配符是完全匹配。ls find cp 这些命令不支持正则表达式，所以只能使用shell自己的通配符来进行匹配。
正则表达式用来在文件中匹配符合条件的字符串，正则式包含匹配。grep awk sed 等命令可以支持正则表达式。
注：正则表达式与通配符的区别：
1.正则是用来匹配文件中的字符串的而通配符是用来匹配文件名的。
2.正则式包含匹配而通配符是完全匹配。
3.搜索文件内容的命令支持正则，搜索文件名的命令支持通配符。

3.基础正则表达式

元字符	作　　用
*	前一个字符匹配0次或任意多次。
.	匹配除了换行符外任意一个字符。
^	匹配行首。例如：^hello会匹配以hello开头的行。
$	匹配行尾。例如：hello&会匹配以hello结尾的行。
[]	匹配中括号中指定的任意一个字符，只匹配一个字符。 例如：[aoeiu] 匹配任意一个元音字母，[0-9] 匹配任意一位数字，[a-z][0-9] 匹配小写字和一位数字构成的两位字符。
[^]	匹配除中括号的字符以外的任意一个字符。例如：[^0-9] 匹配任意一位非数字字符，[^a-z] 表示任意一位非小写字母。
\	转义符。用于取消讲特殊符号的含义取消。
\{n\}	表示其前面的字符恰好出现n次。例如：[0-9]\{4\} 匹配4位数字，[1][3-8][0-9]\{9\} 匹配手机号码。
\{n, \}	表示其前面的字符出现不小于n次。例如：[0-9]\{2,\} 表示两位及以上的数字。
\{n, m\}	表示其前面的字符至少出现n次，最多出现m次。例如：[a-z]\{6,8\} 匹配6到8位的小写字母。

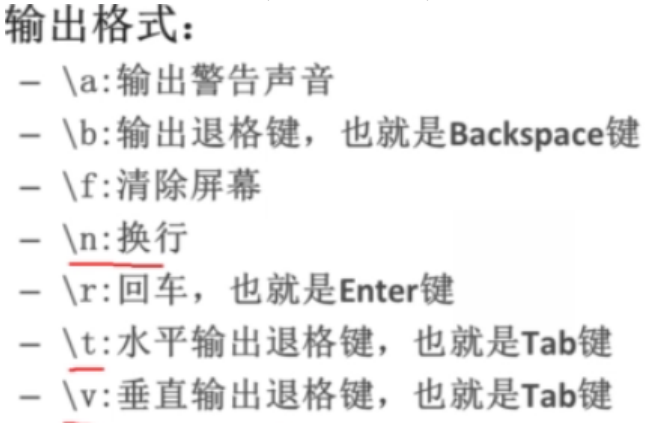
1.* 匹配前面的字符0次或任意次
注：*前面加一个字符是没有意义的，至少要加两个字符。
a* 匹配所有内容，包括空白行
aa* 匹配以a开头的任意字符
aaa*匹配以aa开头的任意字符
2. . 匹配除了换行符外的任意字符
注：a*和 。*都可以匹配整篇文档的内容，但是a*是错误的写法。
3. ^ 匹配行首，$匹配行尾
注：^*匹配空白行
4.[] 匹配中括号中指定的任意一个字符，只匹配一个字符。
s[oa]id
匹配要么是o，要么是a
[0-9]
匹配任意一个数字
^[a-z]
匹配以任意字母开头的字符
5.[^] 匹配除中括号的字符以外的任意一个字符
^[^a-z] 匹配不以小写字母开头的字符
^[^a-zA-Z]匹配不以字母开头的字符
6.\ 转义符
\.$匹配用.结束的文件内容
7.\{n\}标示其前的字符必须出现n次
a\{3\}匹配出现三次字母a的内容
需要加定界符：
xa\{3\}x
这样才会起作用。
8.\{n,\}匹配恰好出现n次以上的内容
9.\{n,m\}匹配其前的字母最少出现n次，最多出现m次。
注：如果\{n\}、\{n,\}、\{n,m\}如果没定义好开始和结尾，则匹配得到的内容都是相同的。

4.正则表达式的实例
a.匹配日期：
[0-9]\{4\}-[0-9]\{2\}-[0-9]\{2\}
b.匹配IP地址
[0-9]\{1,3\}\.[0-9]\{1,3\}\.[0-9]\{1,3\}\.[0-9]\{1,3\}

5.字符截取命令
grep命令：在文件中提取符合条件的一行。
1.cut命令：
cut [选项] 文件名

-f 列号，提取第几列
-d 分隔符，按照指定分隔符分割列。默认使用tab符号
注：grep是行提取命令，cut是列提取命令。
　　cut默认使用tab作为分隔符。
例如：
cut -f 2 student.txt
cut -f 2,4 student.txt
cut -f 1,3 -d ":" student.txt

cut命令的局限：
只要cut找不到分隔符，cut会认为所有内容均在一列。
cut一般适用于比较规律的文件，对于用空格分隔的文件，支持的不是很好。
2.printf命令：
printf '输出类型输出格式' 输出内容
输出类型：
-%ns: 输出字符串。n是指输出几个字符
-%ni:输出整数。n是数字指代输出几个数字
-%m.nf:输出浮点数。m和n是数字，指代输出的整数位数和小数位数。
如：%8.2f 代表共输出8位数，其中2位是小数，6位是整数。



注：单引号不能省略。
printf不支持数据流。
printf是标准输出命令，不会自动加换行符，需要手动加入。
3.awk命令：
awk '条件1{动作1} 条件2{动作2}...' 文件名
条件：
-一般使用关系表达式作为条件
-x>10 判断变量x是否大于10
-x>=10
-x<=10
动作：
-格式化输出
-流程控制语句
例：
awk '{printf %2 "\t"%4"\n"}' student.txt
print和printf的区别：print会在输出的内容后自动加换行符。
awk 'BEGIN{print "test"}'{print $2"\t"$4} student.txt
awk 'END{print "test"}'{print $2"\t"$4} student.txt
FS内置变量：
awk 'BEGIN{FS=":"}'{print $2"\t"$4} student.txt
用于设置分隔符。

关系运算符：
awk '$4 >=70{print %2"\n"}' student.txt

sed命令：
字符替换命令，主要是用来将数据进行选取、替换、删除和新增。

sed [选项] '[动作]' 文件名
选项：
-n:一般sed命令会把所有数据都输出到屏幕，如果加入此选择则只会把经过sed命令处理的行输出到屏幕。
-e:允许对输入数据应用多条sed命令编辑
-i:用sed的修改结果直接修改读取数据的文件，而不是由屏幕输出。

动作：

- a ：追加，在当前行后添加一行或多行
- c ：行替换，用c后面的字符串替换原数据行
- i ：插入，在当期行前插入一行或多行。 d：删除，删除指定的行
- p：打印，输出指定的行。
- s：字串替换，用一个字符串替换另外一个字符串。格式为“行范围s/旧字串/新字串/g”（和vim中的替换格式类似）。

行数据操作：
sed '2p' student.txt
sed -n '2p' student.txt 输出第二行

sed '2,4d' student.txt
删除第二行到第四行的数据

```
# sed  ‘s/旧字串/新字串/g’  文件名

# sed '3s/60/99/g' student.txt
#在第三行中，把60换成99
#sed -i '3s/60/99/g' student.txt
#sed操作的数据直接写入文件

# sed -e 's/fengj//g ; s/cang//g' student.txt
#同时把“fengj”和“cang”替换为空
```

字符处理命令：
1.sort命令：

sort [选项] 文件名

选项：

- -f：忽略大小写
- -n：以数值型进行排序，默认使用字符串型排序
- -r：反向排序
- -t：指定分隔符，默认是分隔符是制表符
- -k n[,m]：按照指定的字段范围排序。从第n字段开始，m字段结束（默认到行尾）

```
sort -t ":" -k 3,3 /etc/passwd
#指定分隔符是“:”，用第三字段开头，第三字段结尾排序，就是只用第三字段排序

sort -n -t ":" -k 3,3 /etc/passwd
```

注：sort命令支持管道数据流。
2.统计命令：wc
wc [选项] 文件名
选项：
-l:只统计行数
-w:只统计单词数
-m:只统计字符数
注：字符是会统计换行符的。

第五章 流程控制语句
一、条件判断语句
1.按文件类型进行判断

测试选项	作　用
-b 文件	判断该文件是否存在，并且是否为块设备文件（是块设备文件为真）
-c文件	判断该文件是否存在，并且是否为字符设备文件（是字符设备文件为真）
-d 文件	判断该文件是否存在，并且是否为目录文件（是目录为真）
-e 文件	判断该文件是否存在（存在为真）
-f 文件	判断该文件是否存在，并且是否为普通文件（是普通文件为真）
-L 文件	判断该文件是否存在，并且是否为符号链接文件（是符号链接文件为真）
-p 文件	判断该文件是否存在，并且是否为管道文件（是管道文件为真）
-s 文件	判断该文件是否存在，并且是否为非空（非空为真）
-S 文件	判断该文件是否存在，并且是否为套接字文件（是套接字文件为真）

判断格式：
test -e /root/install.log
[-e /root/install.log] 推荐使用
注：测试使用：
test -e install.log && echo yes || echo no
中括号两边需要有空格，否则会报错。
2.按文件权限进行判断

测试选项	作　用
-r 文件	判断该文件是否存在，并且是否该文件拥有读权限（有读权限为真）
-w文件	判断该文件是否存在，并且是否该文件拥有写权限（有写权限为真）
-x 文件	判断该文件是否存在，并且是否该文件拥有执行权限（有执行权限为真）
-u 文件	判断该文件是否存在，并且是否该文件拥有SUID权限（有SUID权限为真）
-g 文件	判断该文件是否存在，并且是否该文件拥有SGID权限（有SGID权限为真）
-k 文件	判断该文件是否存在，并且是否该文件拥有SBit权限（有SBit权限为真）

注：上述判断并不会涉及所有者是否有相应权限，只要有一个就为真。
3.两个文件之间的比较

测试选项	作　用
文件1 -nt 文件2	判断文件1的修改时间是否比文件2的新（如果新则为真）
文件1 -ot 文件2	判断文件1的修改时间是否比文件2的旧（如果旧则为真）
文件1 -ef 文件2	判断文件1是否和文件2的Inode号一致，可以理解为两个文件是否为同一个文件。这个判断用于判断硬链接是很好的方法

4.两个整数之间的比较

测试选项	作　用
整数1 -eq 整数2	判断整数1是否和整数2相等（相等为真）
整数1 -ne 整数2	判断整数1是否和整数2不相等（不相等位置）
整数1 -gt 整数2	判断整数1是否大于整数2（大于为真）
整数1 -lt 整数2	判断整数1是否小于整数2（小于位置）
整数1 -ge 整数2	判断整数1是否大于等于整数2（大于等于为真）
整数1 -le 整数2	判断整数1是否小于等于整数2（小于等于为真）

注：由于存在上述的运算符，shell会自动将左右两个字符串转换为整型，从而可以进行比较。
5.字符串的判断

测试选项	作　　用
-z 字符串	判断字符串是否为空（为空返回真）
-n 字符串	判断字符串是否为非空（非空返回真）
字串1 ==字串2	判断字符串1是否和字符串2相等（相等返回真）
字串1 != 字串2	判断字符串1是否和字符串2不相等（不相等返回真）

6.多重条件判断

测试选项	作　　用
判断1 -a 判断2	逻辑与，判断1和判断2都成立，最终的结果才为真
判断1 -o 判断2	逻辑或，判断1和判断2有一个成立，最终的结果就为真
！判断	逻辑非，使原始的判断式取反

二、单分支if语句

```
 if [条件判断式] then;
    语句
fi
if[条件判断式]
then
   语句
fi
```

注：中括号和条件判断式之间一定要有空格。

判断分区使用率：

```
#!/bin/bash

#check the / is full
 dev=df -lhlgrep '[/]$'lawk '{print $5}'lcut -d '%' -f 1
if [ '$dec' -ge '90' ]
then
 echo 'the / is full'
fi
```

三、双分支if语句

```
if [ 条件 ]
  then
     语句
  else
     语句
fi
```

四、多分支if语句

```
if [ condition ]
```

```
    then
       ...
elif
    then
     ...
     ...
else
      ....
if
```

多分支语句实现计算器：
1.是否是是数值的判断：
$(echo $num_1|sed 's/[0-9]//g')
只要判断上面的值是否为空就可以。为空则表明为数值型数据。

五、多分支case语句

```
case $var in
        "值1")
        ；；
         "值2")
         ；；

          *)
          ;
esac
```

六、for循环

```
for i in $var
        do

        done
for ((i=1;i<100;i=i+1))
        do

        done
```

批量解压缩文件；

passwd --stdin 为某用户设置初始密码

while循环和until循环

```
while [ 条件 ]
          do

          done
until [ 条件 ]
          do

          done
```

主控脚本：
场景脚本：提取Linux操作系统信息；获取操作系统运行状态；分析应用状态；应用日志分析

应用状态监控脚本：
命令：
利用操作系统命令：
网络命令：ping、nslookup、nm-tool, tracertroute,dig,telnet,nc,curl
监控进程：ps、netstat、pgrep
第三方工具包：nginxstatus、nagios-libexec

服务端接口支持：
nginx–http_stub_status_module
nutcracker监控集群状态；

nginx监控脚本：
curl –m 5 –s –w %{http_code} http://www.baidu.com/nginx_status –o /dev/null

mysql主从复制监控：
show slave status\G;
slave_io_running:io线程是否连接到主服务器上；
seconds_behind_master:主从同步的延时时间；

nc –z –w2 ip port
通过$?判断是否连接成功；
shell直接通过函数名来调用函数；
awk '{if($2 != Yes){print 123;exit 1}}'

系统信息监控脚本
if [$# –eq 0]
    then

fi
获取操作系统信息：
操作系统类型：uname -o
发行版本：cat /etc/issue | grep -v Kernel
cpu架构：uname -m
内核版本：uname -r
主机名：uname -n
内网ip：hostname -I
公网ip：curl -s http://ipecho.net/plain
dns:cat /etc/resolv.conf | grep -E "\<nameserver[ ]+"|awk "{printf $2}"
网络是否已经连接：ping -c 2 www.baidu.com &> /dev/null && echo 'ok' || echo 'no ok'

操作系统内存：
系统使用内存 = total – free
应用使用内存 = total – （free + cache +buffer）
注：
内存中cache和buffer的区别：

	功能	读取策略
Cache	缓存主要用于打开的文件	最少使用原则（LRU）
Buffer	分缓存主要用于目录项、inode等文件系	先进先出策略

cache比较大表明文件读取比较频繁，缓存比较大；
buffer比较大表明inode缓存比较大；
操作系统使用内存：
 awk '/MemTotal/{total=$a}/MemFree/{free=$2}END{printf (total–free)/1024}' /proc/meminfo
应用使用内存：
awk '/MemTotal/{total=$2}/MemFree/{free=$2}/^Cached/{cached=$2}END{printf (total–free–buffer–cached)/1024}'
/proc/meminfo

操作系统负载：
top -n 1 -b |grep 'load average'
硬盘容量：

df -hPlgrep -vE 'file system'lawk

shell高亮显示：
基本格式：
echo -e 终端颜色 显示内容 结束后颜色
tput sgr0 重置终端
例：
echo -e "\e[1;30m" hello world $(tput sgr0)

shell中的关联数组：
声明关联数组：declare –A array
array[index]=value

ls –l 'text' ./
列出除text文件外的所有文件；