一篇文章学懂Shell脚本

程序君 **Linux编程** 2 days ago

Linux编程
点击右侧关注，免费入门到精通！

作者丨关玮琳linSir
https://www.jianshu.com/p/71cb62f08768

Shell脚本,就是利用Shell的命令解释的功能，对一个纯文本的文件进行解析，然后执行这些功能，也可以说Shell脚本就是一系列命令的集合。

Shell可以直接使用在win/Unix/Linux上面，并且可以调用大量系统内部的功能来解释执行程序，如果熟练掌握Shell脚本，可以让我们操作计算机变得更加轻松，也会节省很多时间。

⛵Shell应用场景

⛵Shell能做什么

将一些复杂的命令简单化(平时我们提交一次github代码可能需要很多步骤，但是可以用Shell简化成一步)

可以写一些脚本自动实现一个工程中自动更换最新的sdk(库)

自动打包、编译、发布等功能

清理磁盘中空文件夹

总之一切有规律的活脚本都可以尝试一下

⛵Shell不能做什么

需要精密的运算的时候

需要语言效率很高的时候

需要一些网络操作的时候

总之Shell就是可以快速开发一个脚本简化开发流程，并不可以用来替代高级语言

⛵Shell的工作原理

Shell可以被称作是脚本语言，因为它本身是不需要编译的，而是通过解释器解释之后再编译执行，和传统语言相比多了解释的过程所以效率会略差于传统的直接编译的语言。

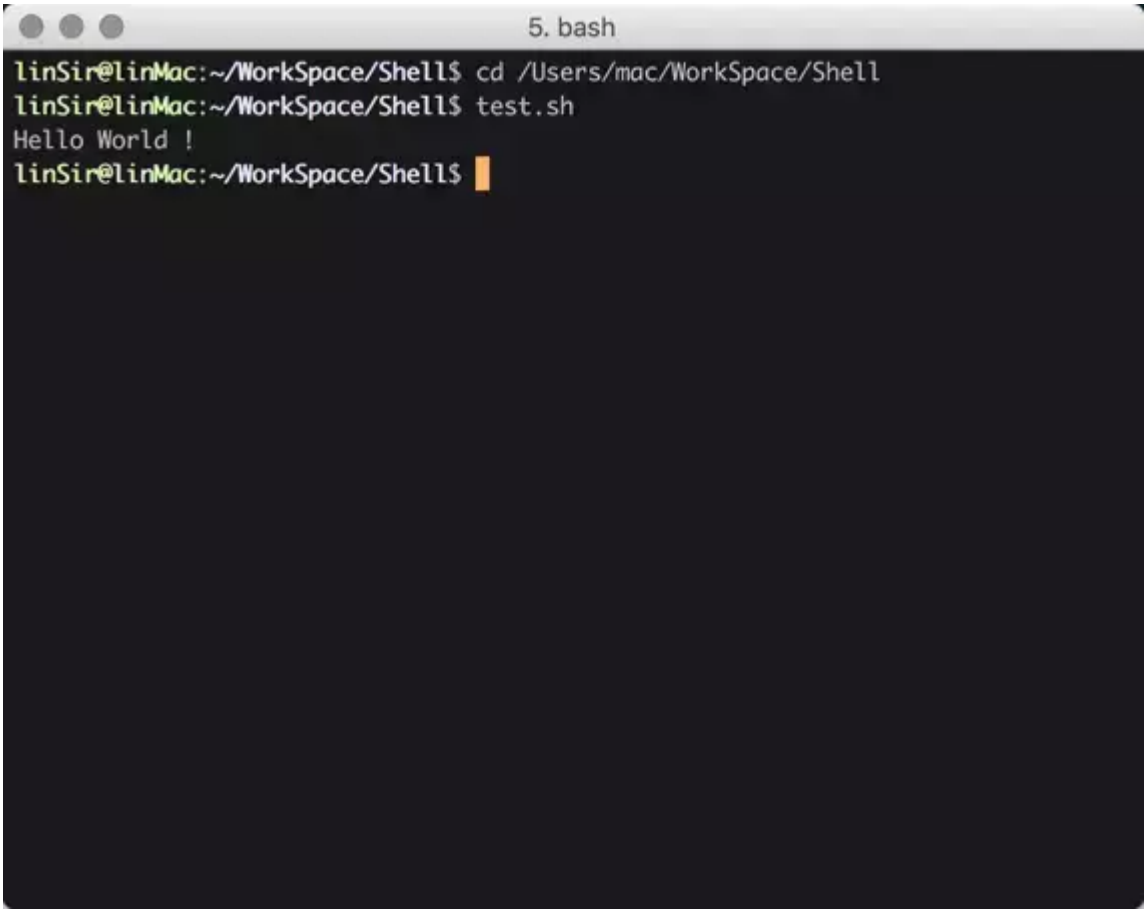
⛵最简单的脚本:

```
#!/bin/bash
echo "Hello World"
```

只需要打开文本编辑工具，编辑成以上的样子,然后保存成test.sh

⛵运行该脚本：

1. cd 到该目录下
2. chmod +x ./test.sh #给脚本权限
3. ./test.sh #执行脚本



这样我们便写出来了第一个最简单的脚本，下面我们可以尝试着写一些复杂的脚本。

⛵Shell中的变量

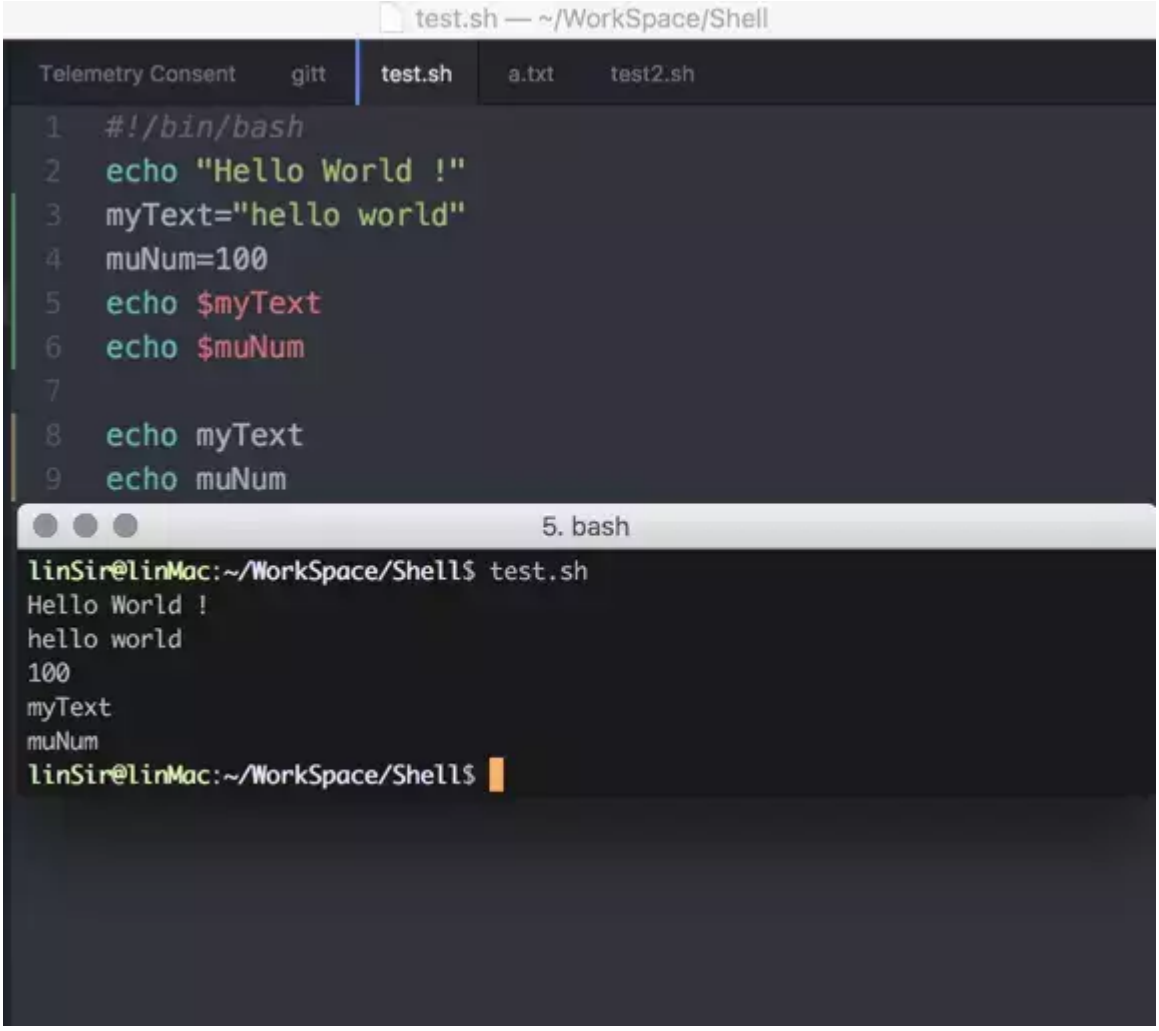
```
myText="hello world"
muNum=100
```

这里面需要注意的就是，“=”前后不能有空格，命名规则就和其它语言一样了。

⛵访问变量

```
myText="hello world"
muNum=100
echo $myText
echo muNum
```

当想要访问变量的时候，需要使用$，否则输出的将是纯文本内容，如下图所示。



Shell中的四则运算

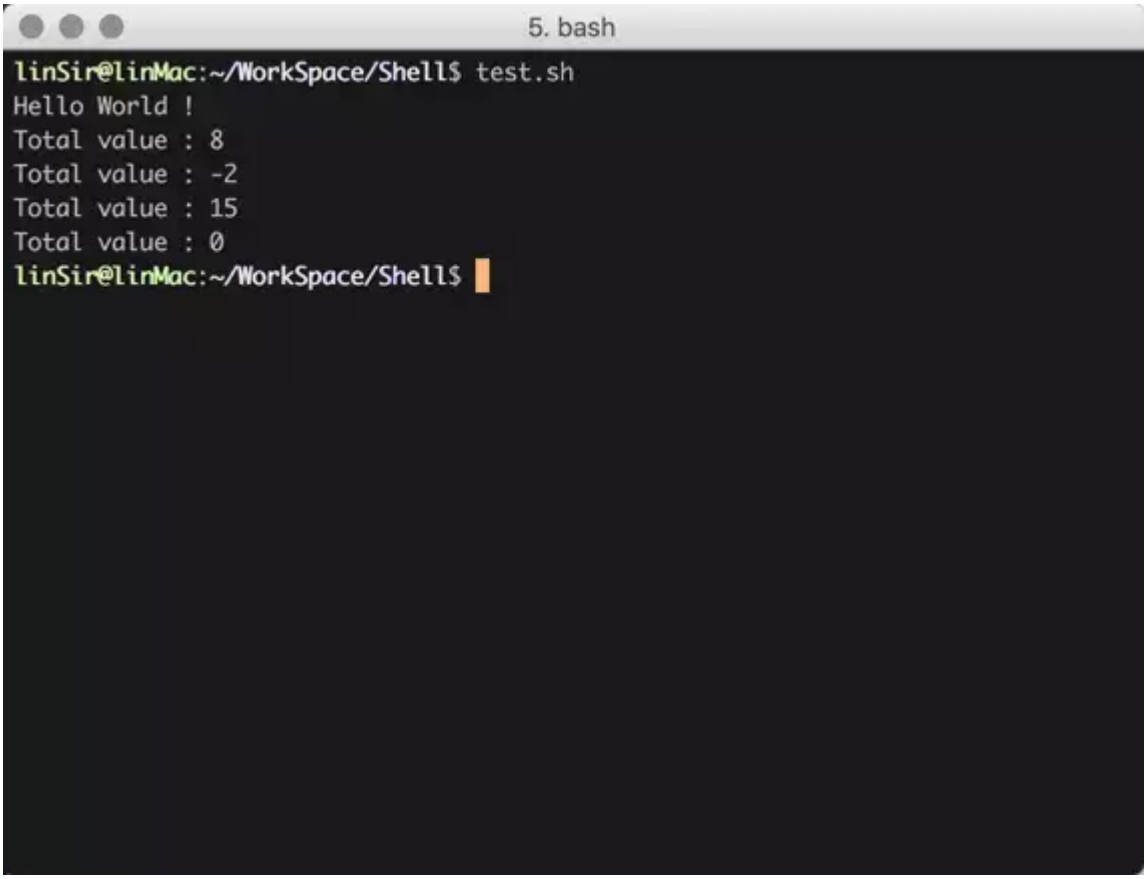
运算符	含义
+	加法运算
-	减法运算
*	乘法运算
/	除法运算

例子程序

```
#!/bin/bash
echo "Hello World !"
a=3
b=5
val=`expr $a + $b`
```

```
echo "Total value : $val"

val=`expr $a - $b`
echo "Total value : $val"

val=`expr $a \* $b`
echo "Total value : $val"

val=`expr $a / $b`
echo "Total value : $val"
```

这里面需要注意的就是，定义变量的时候“=”前后是不能有空格的，但是进行四则运算的时候运算符号前后一定要有空格，乘法的时候需要进行转义。

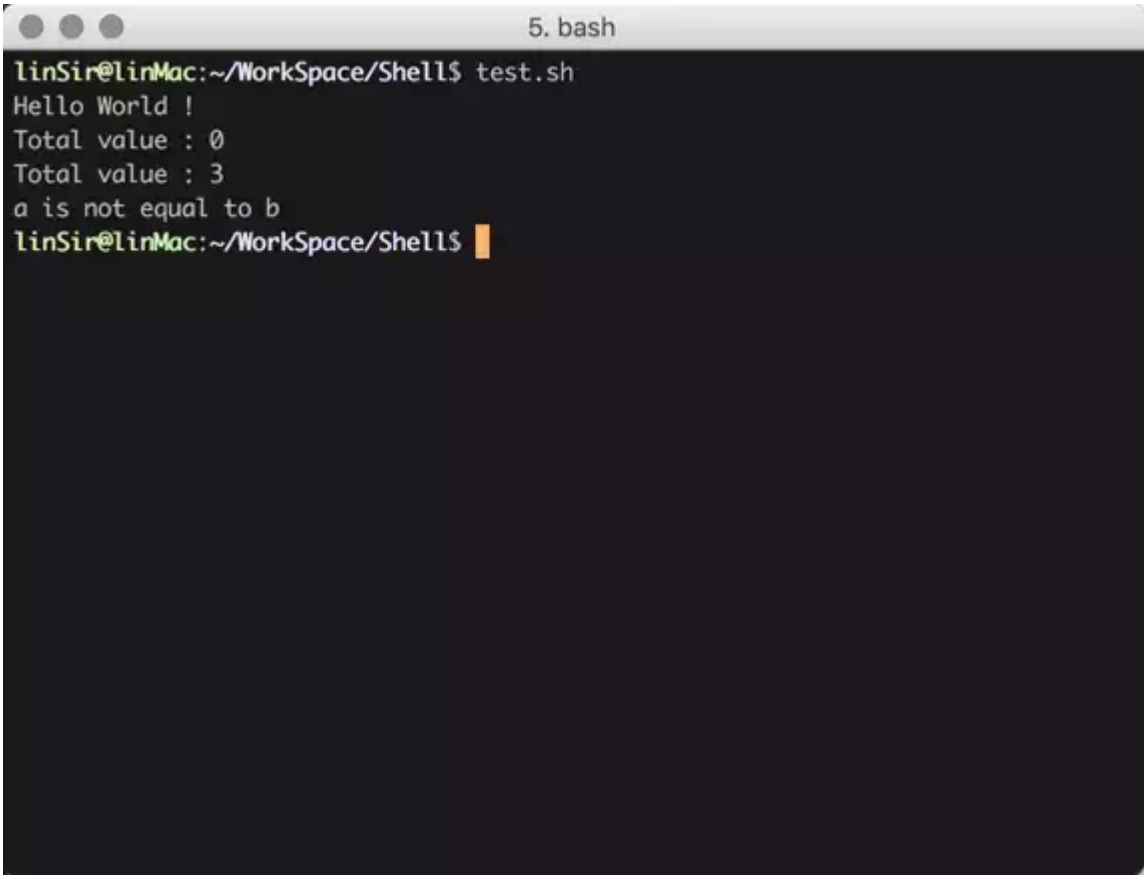


其它运算符 =、==、!=、 ！、-o、-a

运算符	含义
%	求余
==	相等
=	赋值
!=	不相等
!	非
-o	或
-a	与

例子程序

```
a=3
b=5
val=`expr $a / $b`
echo "Total value : $val"

val=`expr $a % $b`
echo "Total value : $val"

if [ $a == $b ]
then
   echo "a is equal to b"
fi
if [ $a != $b ]
then
   echo "a is not equal to b"
fi
```



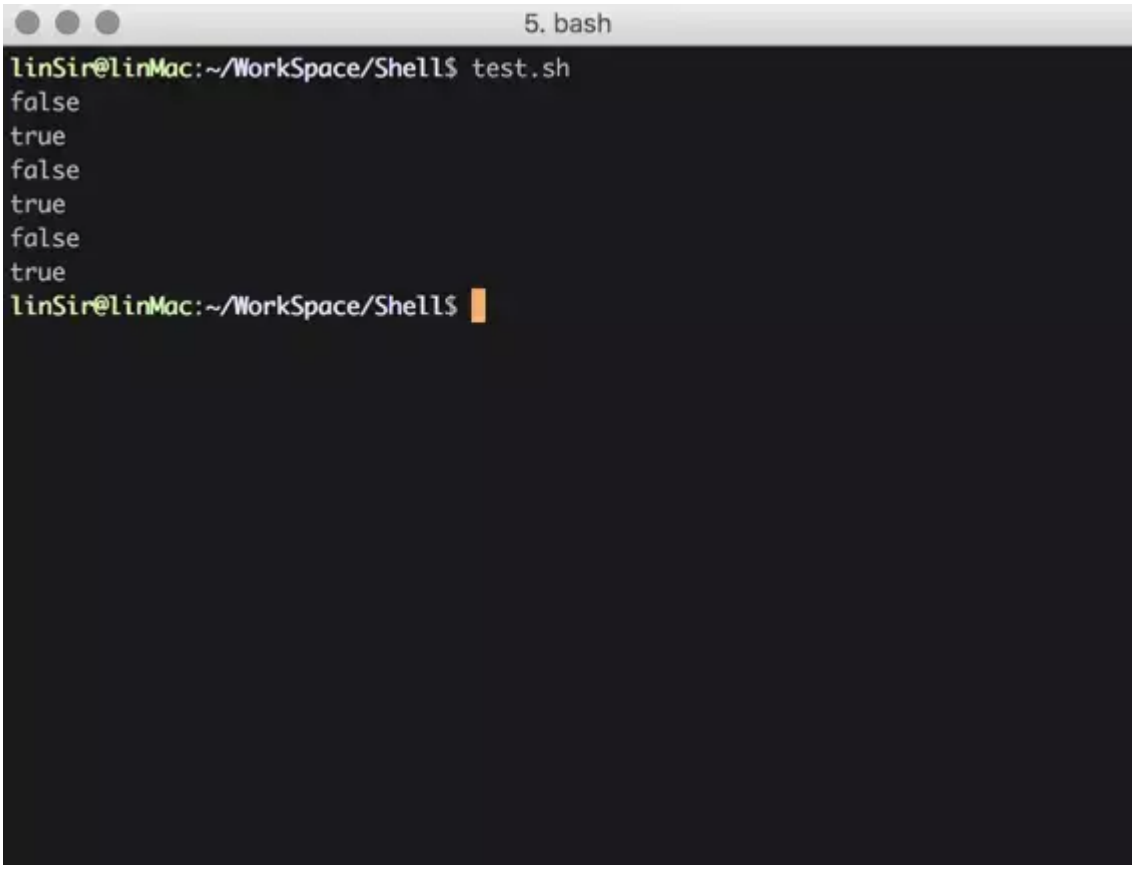
关系运算符

运算符	含义
-eq	两个数相等返回true
-ne	两个数不相等返回true
-gt	左侧数大于右侧数返回true
-lt	左侧数小于右侧数返回true
-ge	左侧数大于等于右侧数返回true
-le	左侧数小于等于右侧数返回true

例子程序

```
#!/bin/sh
a=10
b=20
if [ $a -eq $b ]
then
   echo "true"
```

```
else
   echo "false"
fi

if [ $a -ne $b ]
then
   echo "true"
else
   echo "false"
fi

if [ $a -gt $b ]
then
   echo "true"
else
   echo "false"
fi

if [ $a -lt $b ]
then
   echo "true"
else
   echo "false"
fi

if [ $a -ge $b ]
then
   echo "true"
else
   echo "false"
fi

if [ $a -le $b ]
then
   echo "true"
else
   echo "false"
fi
```



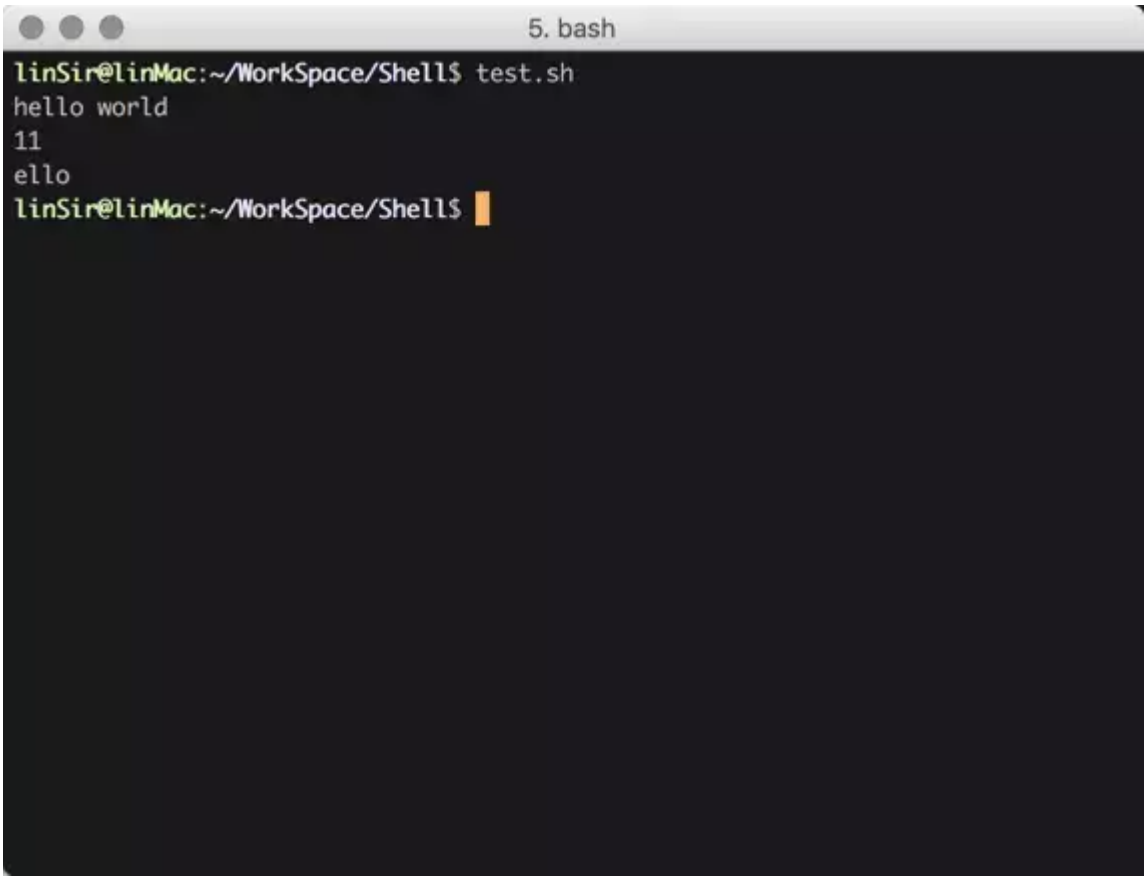
字符串运算符

运算符	含义
=	两个字符串相等返回true
!=	两个字符串不相等返回true
-z	字符串长度为0返回true
-n	字符串长度不为0返回true

运算符	含义
-d file	检测文件是否是目录，如果是，则返回 true
-r file	检测文件是否可读，如果是，则返回 true
-w file	检测文件是否可写，如果是，则返回 true
-x file	检测文件是否可执行，如果是，则返回 true
-s file	检测文件是否为空（文件大小是否大于0，不为空返回 true
-e file	检测文件（包括目录）是否存在，如果是，则返回 true

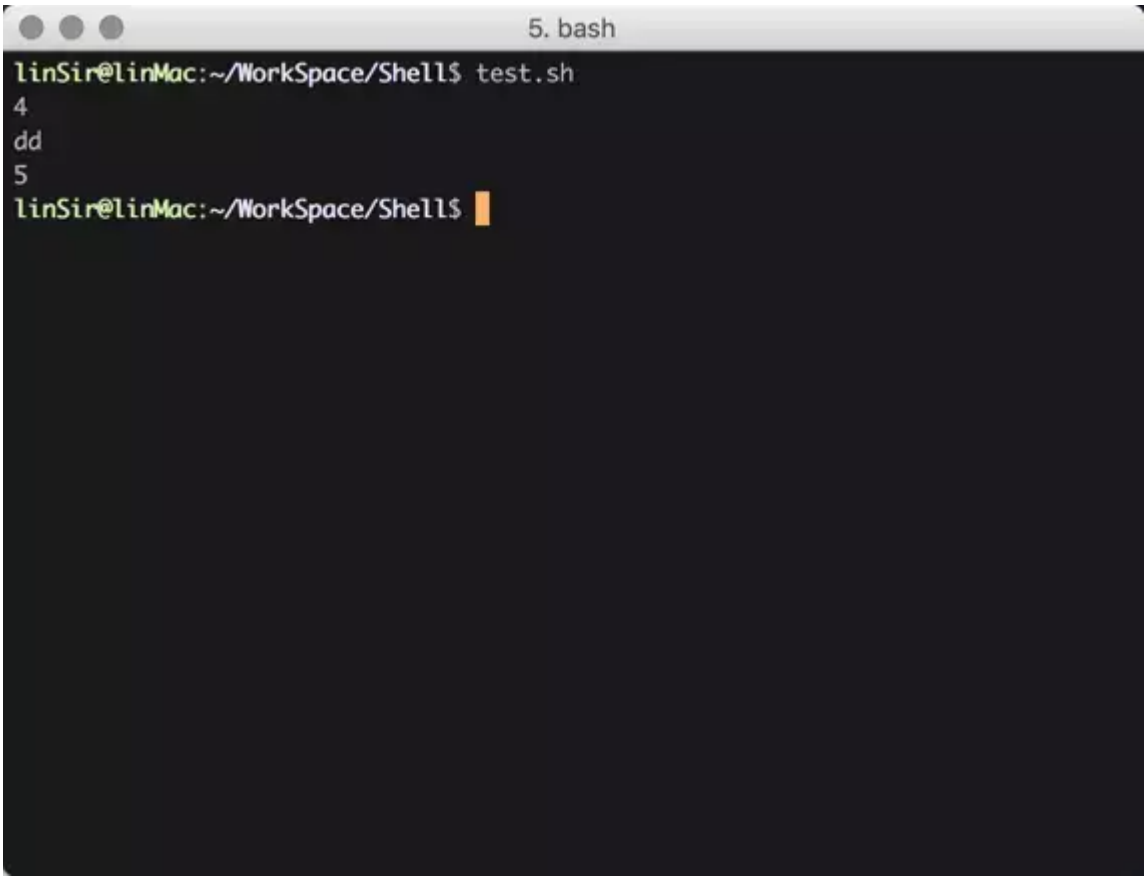
字符串

```
#!/bin/sh
mtext="hello"  #定义字符串
mtext2="world"
mtext3=$mtext" "$mtext2  #字符串的拼接
echo $mtext3  #输出字符串
echo ${#mtext3}  #输出字符串长度
echo ${mtext3:1:4}  #截取字符串
```



数组

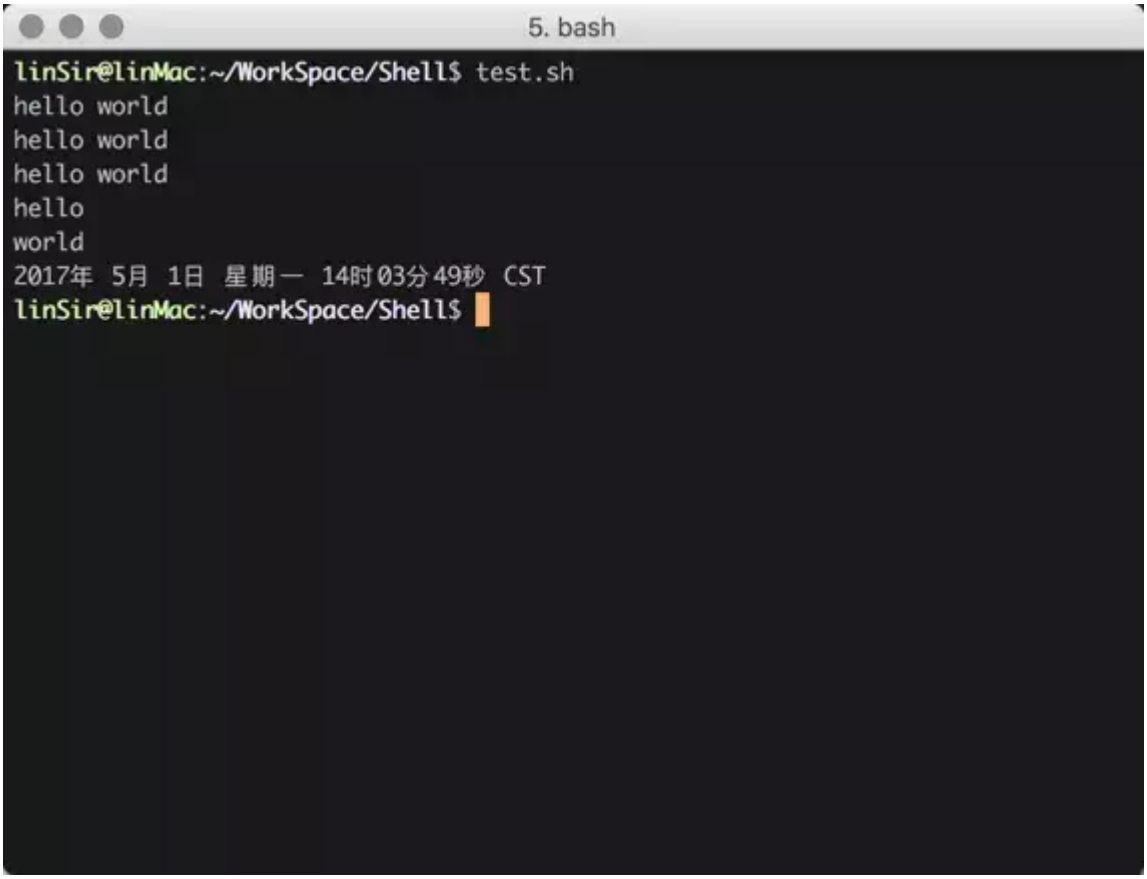
```
#!/bin/sh
array=(1 2 3 4 5)  #定义数组
array2=(aa bb cc dd ee)  #定义数组
value=${array[3]}  #找到某一个下标的数，然后赋值
echo $value  #打印
value2=${array2[3]}  #找到某一个下标的数，然后赋值
echo $value2  #打印
length=${#array[*]}  #获取数组长度
echo $length
```



⛵输出程序

echo

```
#!/bin/sh
echo "hello world"
echo hello world

text="hello world"
echo $text

echo -e "hello \nworld"  #输出并且换行

echo "hello world" > a.txt  #重定向到文件

echo `date`  #输出当前系统时间
```



printf

同c语言，就不过多介绍了

⛵判断语句

if

if-else

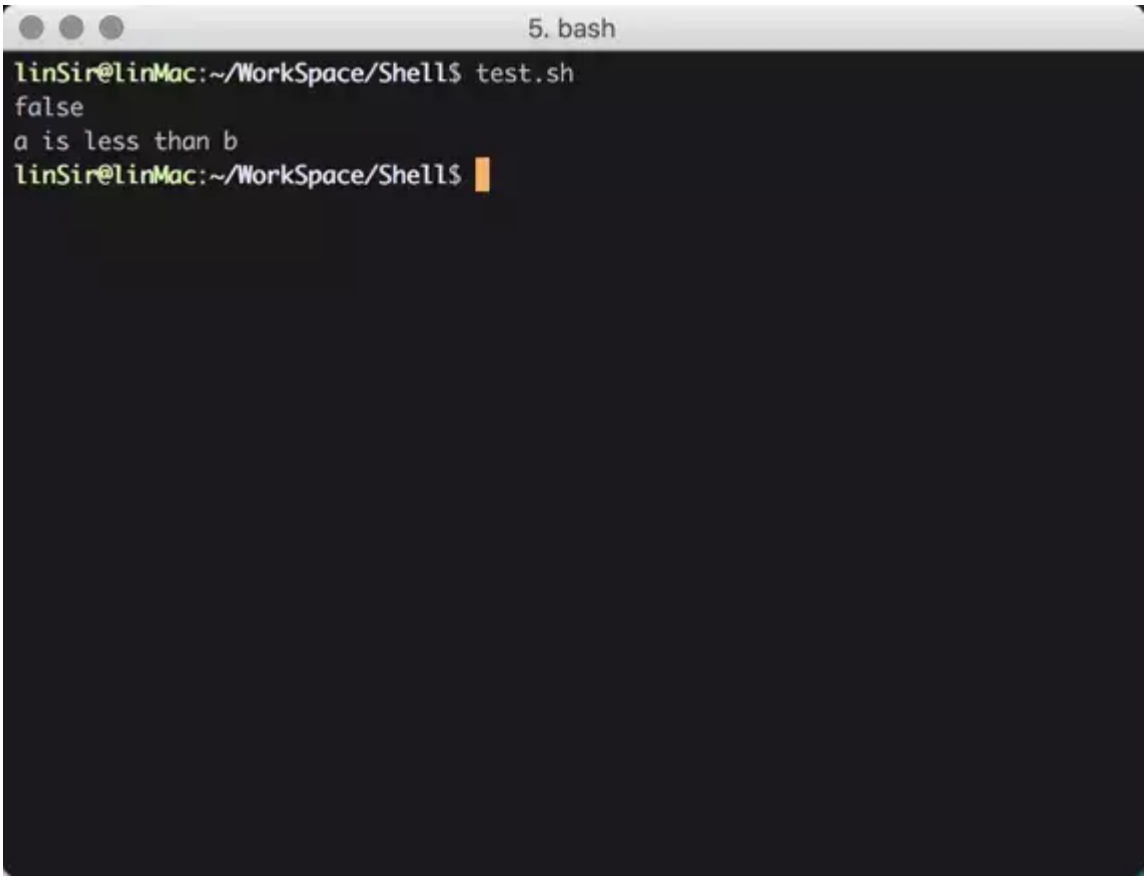
if-elseIf

case

```
#!/bin/sh
a=10
b=20
if [ $a == $b ]
then
   echo "true"
fi
```

```
if [ $a == $b ]
then
   echo "true"
else
   echo "false"
fi
```

```
if [ $a == $b ]
then
   echo "a is equal to b"
elif [ $a -gt $b ]
then
   echo "a is greater than b"
elif [ $a -lt $b ]
then
   echo "a is less than b"
else
   echo "None of the condition met"
fi
```



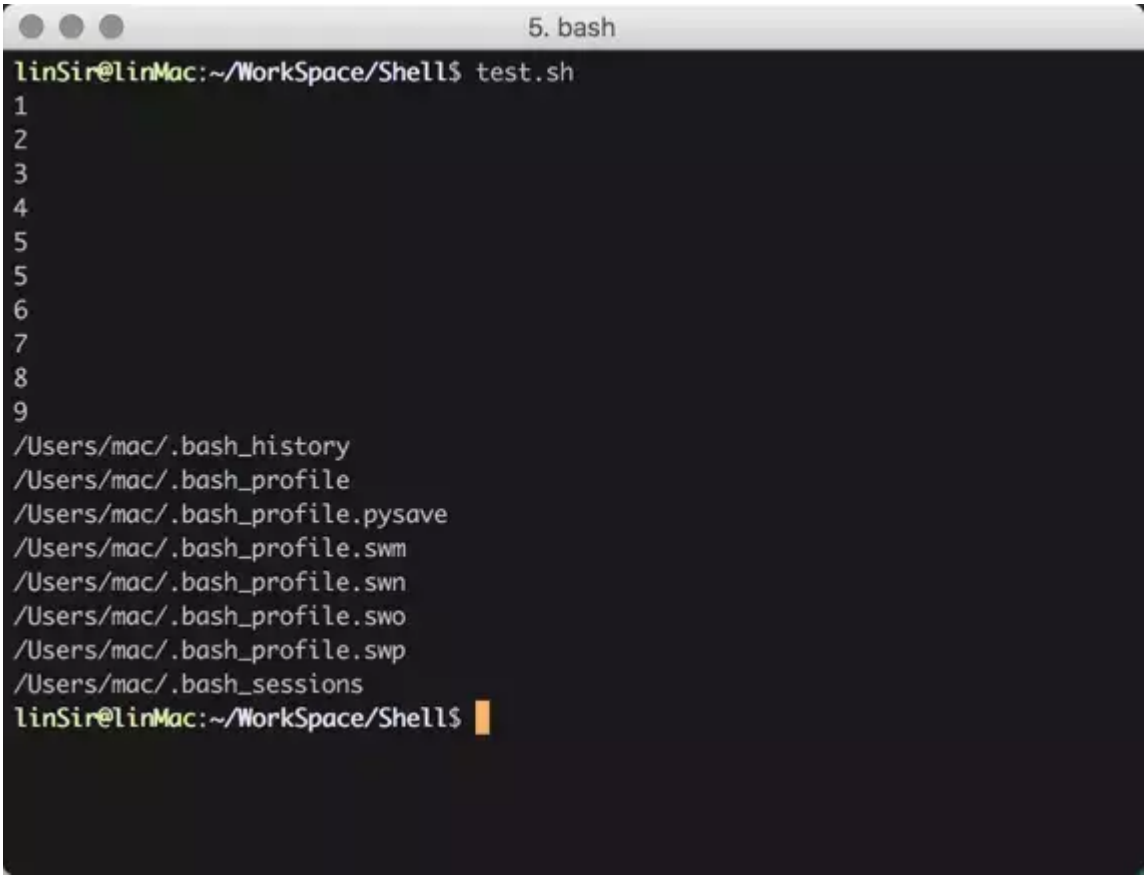
test命令

```
test $[num1] -eq $[num2]  #判断两个变量是否相等
test num1=num2  #判断两个数字是否相等
```

参数	含义
-e file	文件存在则返回真
-r file	文件存在并且可读则返回真
-w file	文件存在并且可写则返回真
-x file	文件存在并且可执行则返回真
-s file	文件存在并且内容不为空则返回真
-d file	文件目录存在则返回真

for循环

```
#!/bin/sh

for i in {1..5}
do
    echo $i
done


for i in 5 6 7 8 9
do
    echo $i
done


for FILE in $HOME/.bash*
do
   echo $FILE
done
```



while循环

```
#!/bin/sh
COUNTER=0
while [ $COUNTER -lt 5 ]
do
    COUNTER=`expr $COUNTER + 1`
    echo $COUNTER
done

echo '请输入。。。'
echo 'ctrl + d 即可停止该程序'
while read FILM
do
    echo "Yeah! great film the $FILM"
done
```

以上是while循环的两种用法，第一种是比较常规的，执行循环，然后每次都把控制的数加1，就可以让while循环有退出的条件了。

第二种是用户从键盘数据，然后把用户输入的文字输出出来。

跳出循环

```
break  #跳出所有循环
break n  #跳出第n层f循环
continue  #跳出当前循环
```

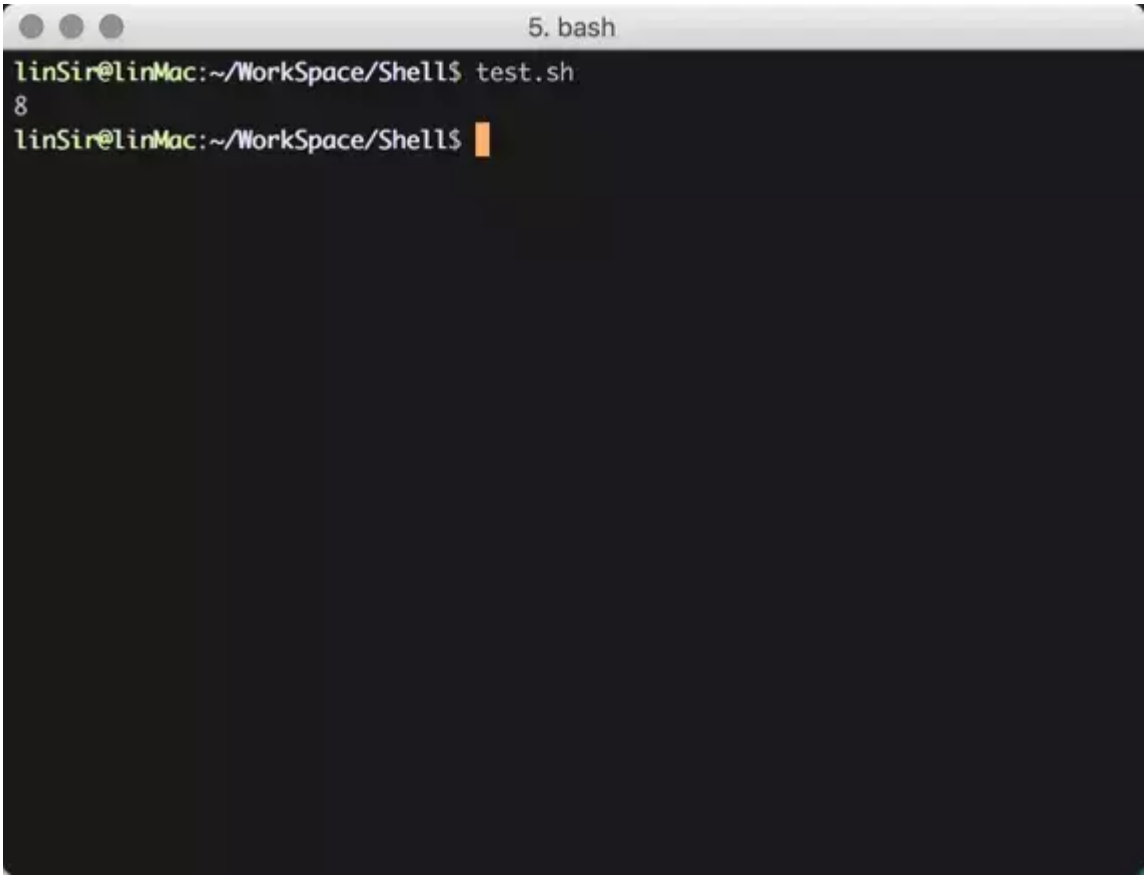
函数

```
#!/bin/sh

sysout(){
    echo "hello world"
}

sysout
```

定义一个没有返回值的函数，然后调用该函数

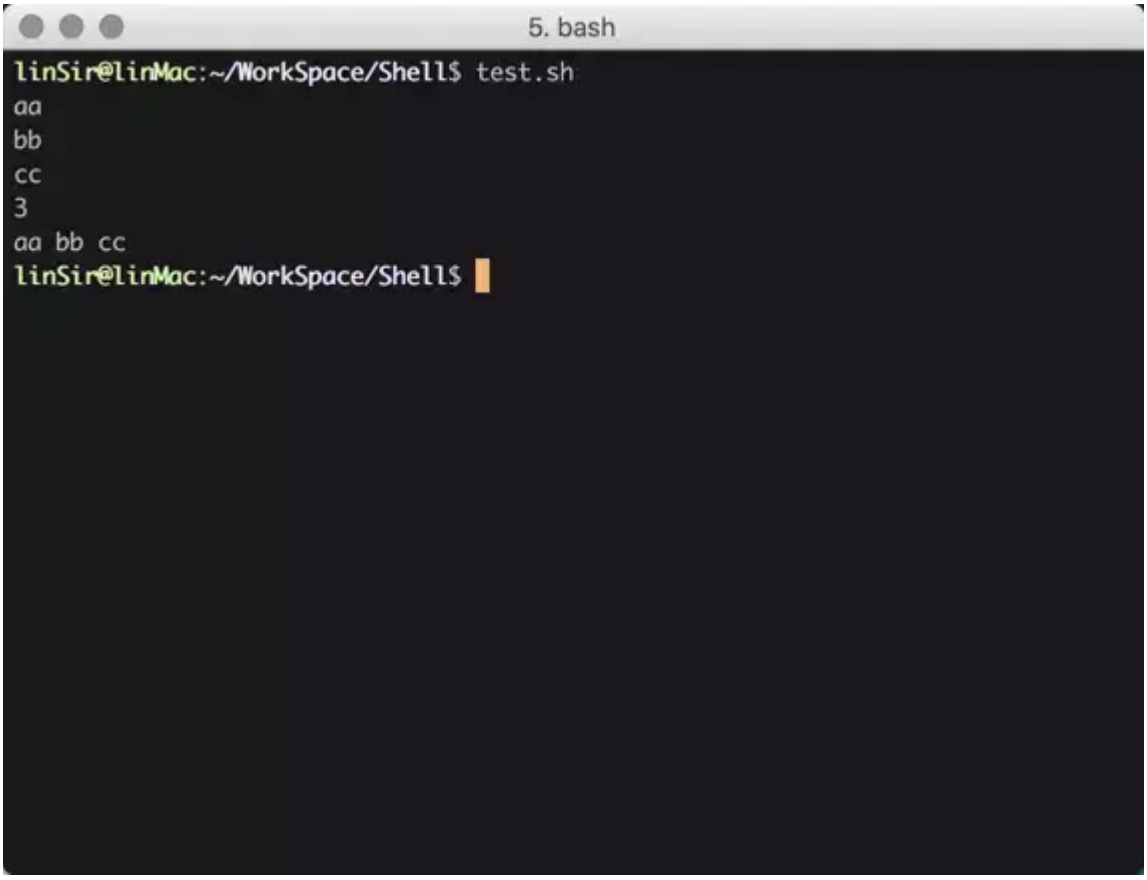
```
#!/bin/sh

test(){

    aNum=3
    anotherNum=5
    return $(($aNum+$anotherNum))
}
test
result=$?
echo $result
```

定义一个有返回值的函数，调用该函数，输出结果




```
#!/bin/sh

test(){
    echo $1  #接收第一个参数
    echo $2  #接收第二个参数
    echo $3  #接收第三个参数
    echo $#  #接收到参数的个数
    echo $*  #接收到的所有参数
}

test aa bb cc
```

定义了一个需要传递参数的函数



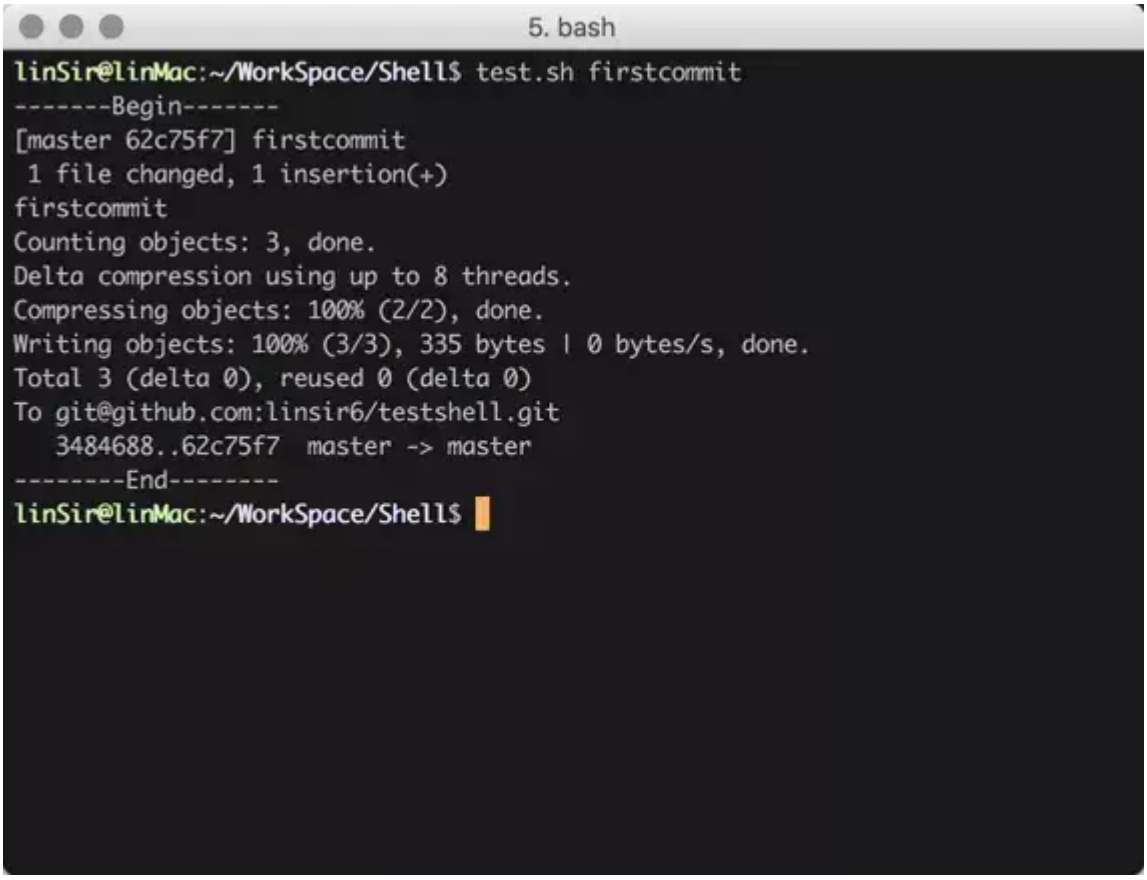
重定向

```
$echo result > file  #将结果写入文件，结果不会在控制台展示，而是在文件中
$echo result >> file  #将结果写入文件，结果不会在控制台展示，而是在文件中
echo input < file  #获取输入流
```

写一个自动输入命令的脚本

自动提交github仓库的脚本

```
#!/bin/bash
echo "-------Begin-------"
git add .
git commit -m $1
echo $1
git push origin master
echo "--------End--------"
```


5. bash
linSir@linMac:~/WorkSpace/Shell$ test.sh firstcommit
-------Begin-------
[master 62c75f7] firstcommit
1 file changed, 1 insertion(+)
firstcommit
Counting objects: 3, done.
Delta compression using up to 8 threads.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 335 bytes | 0 bytes/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To git@github.com:linsir6/testshell.git
3484688..62c75f7 master -> master
--------End--------
linSir@linMac:~/WorkSpace/Shell$

推荐↓↓↓

长 按 关 注

👉【16个技术公众号】都在这里！

涵盖：程序员大咖、源码共读、程序员共读、数据结构与算法、黑客技术和网络安全、大数据科技、编程前端、Java、Python、Web编程开发、Android、iOS开发、Linux、数据库研发、幽默程序员等。

万水千山总是情，点个 “好看” 行不行

Read more