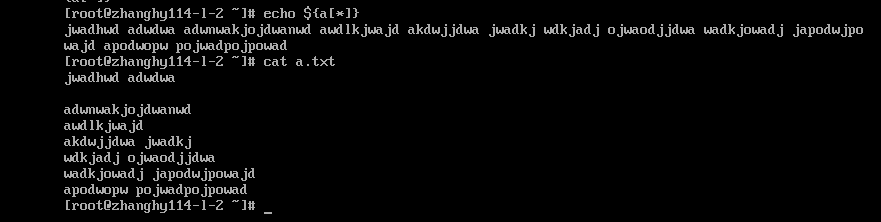
# 1.数组遍历

遍历整个数组

a=(`cat a.txt`)

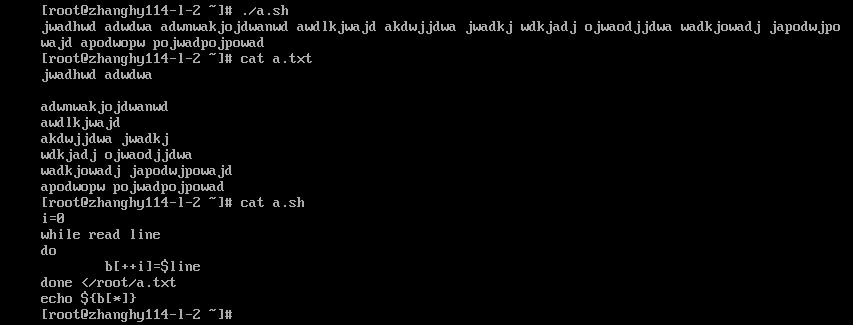
的确能够将文件每一行赋值给数组每一个值。但是如果文件有空格，他会将空格也当作分隔符

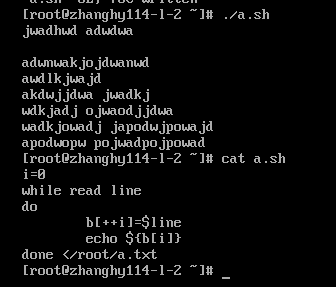


手动编写脚本将值赋值给数组，而且是一行一个值？这就需要循环了。

但是有一个问题，我运行这个脚本还必须是. a.sh

因为在子脚本中赋值变量是无法在shell中生效的。



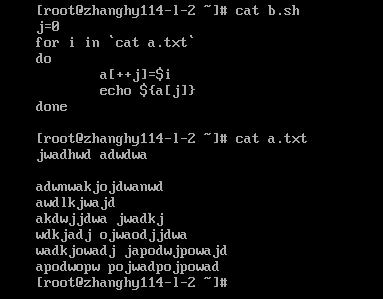


可以清楚得看见，这是以行为标准。

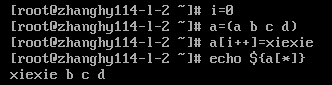
将一行为标准赋值给数组

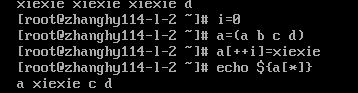
接下里我们已for为标准。

其实for是以空格作为分隔符号的。



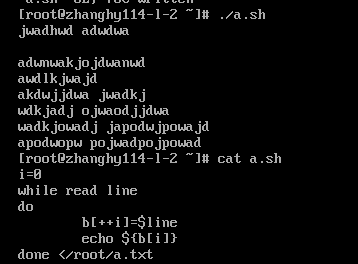
注意：数组遍历，注意的是a[++j]和a[j++].

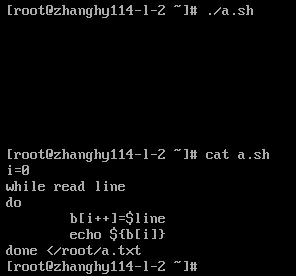




我当然知道二者区别。

但是问题是：为什么遍历必须用++j,而用j++就无效





我来教大家为啥只能用++i不能用i++

就以上面图片为例：

b[i++]=$line

是将b[0]=$line

然后我们打印 ${b[i]}这是打印b[1]的值。很遗憾，我还没有给b[1]赋值。

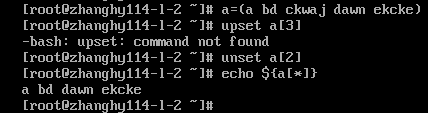
然后接下里我给b[1]赋值了，但是我打印的是b[2]

所以要打印全部，只有从do done 后面放置信息

## 数组的删

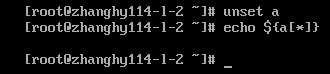
删除单个值

unset array[2]



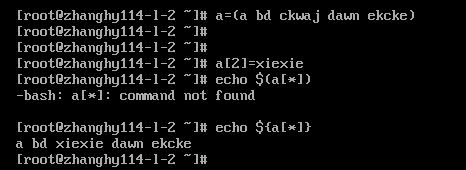
删除整个数组：

unset array



## 数组的改：

直接array[num]=值

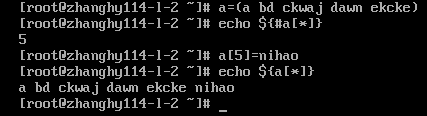


数组的增：

a=${#array[\*]}

先打印长度

array[a]=新增的值（因为数组的值是从0开始）



针对关联数组，这个还要定义

declare –A a (前提这个数组不存在)

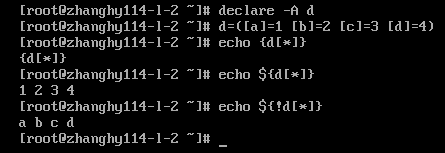
a=([a]=1 [b=2 [c]=3 [d]=4])

查看数组索引

echo ${!a[\*]}

查看数组值

echo ${a[\*]}



## 数组：功能：统计列表的值

eg:

|  |  |  |
| --- | --- | --- |
| Name | Age | Sex |
| Zhang | 12 | M |
| Ua | 12 | W |
| Dawd | 32 | W |
| Adwdw | 22 | M |
| Wda | 33 | W |
| dwa | 11 | x |

## 其实还要讲到一个功能let：

let 变量++

1. 变量不需要加$
2. 变量不需要赋值0，默认就是0

啥意思

比如

i=0

for I in seq 1 12

do

let m++

done

echo $m

这个是成立的

虽然m没定义，虽然没写$m,但是就是这样

结果 12

let 可以做很多运算

let i++

let a=a+2

let a\*=2

但是let进行运算很完美，但是没有输出

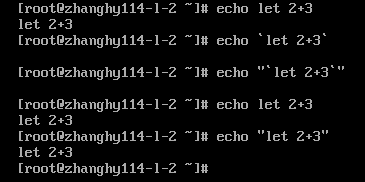
我们可以将输出给一个变量，然后打印 eg let a\*=10 echo $a

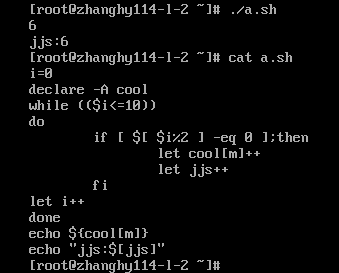
绝对绝对不要用echo let 运算

就算是echo `let 2+3`

echo “let 2+3”

都不行

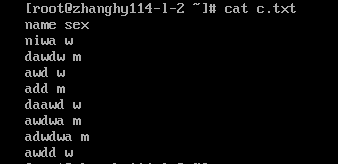




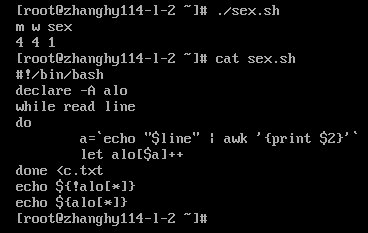
那么开始数组，统计上面表格男人数量和女人数量

eg:sex=([man]=12 [woman]=23)

怎么实现



就像这份表



我成功的统计了信息

# 2.函数

功能：方便代码复用化

函数必须先定义才可以使用

fangfu() {

内容

}

fangfu

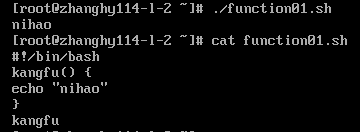
例如：

fangfu() {

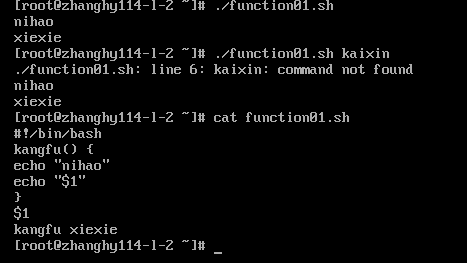
echo “nihao”

}

fangfu



函数的位置参数，如果函数里面有$!，注意这不是脚本的位置参数，这就是函数的。



传递方法，函数名 参数

## shell的C语言循环写法

shell写法：

#!/bin/bash

i=0

**while (($i<**=10))

do

let i++

done

其实也可以用c语言风格

for ((i=1;$i<=10;i++))

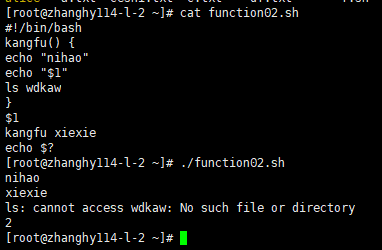
do

done

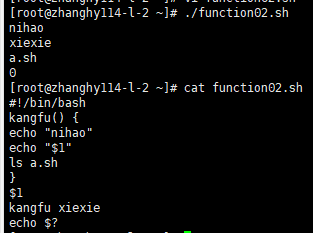
## 函数返回值

返回的是函数最后一条命令的值。

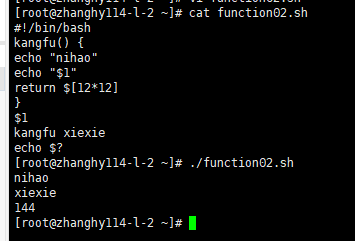
我故意将函数最后一条命令写错



我将函数最后一条命令写对

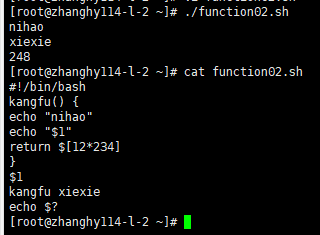


当然，手动定义返回值是最完美的



我return $[12\*12]，返回了144

但是要是$[12\*234]

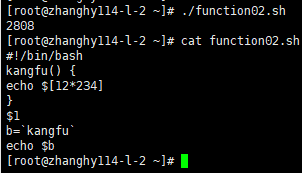


为什么返回值是248，因为返回值最大就是255

返回字符串可以吗？不可以

那怎么办？

将函数的结果打印出来：



传递位置参数：

函数传递位置参数和脚本传递位置参数用法一样，但是传递一定要注意，传递给脚本的实参无法传递给函数

if [ $# -ne 3 ];then

fi

传递参数，其实可以将数组值传递给函数。

num=(1 2 3 4 5 6)

cool() {

for I in $\*

do

done

}

cool ${num[\*]}

接下来演示结果：

其实就是

函数 ${a[\*]}

将数组的所有值当作函数的位置参数，而函数则使用 $i 来即可 ，这是不对的

因为$i代表数组的值 ,而$1代表的是数组位置，二者不可混淆

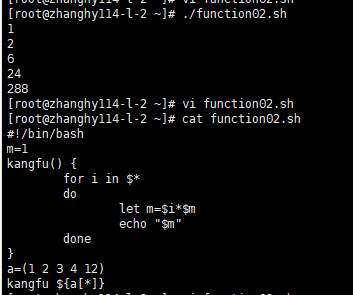
应该是 $\* 代表所有位置参数，

for I in $\*

do

echo $i 就可以遍历所有的位置参数了

done



# 3．循环扩展

break

这个我之前说过了，只要使用，就会跳出循环

但是在嵌套循环中，内循环跳出 就用break，但若是要一起跳出呢

break 2就是跳出2层循环

shift

位置参数每次向左移动一个，直到没有

啥意思呢？

比如我想传递给一个脚本位置参数，但是我自己都不知道要传递多少个，我就写一个函数，用shift将位置参数每次移动一个，第一次是$1,移动之后就是$2

那么演示：

脚本：

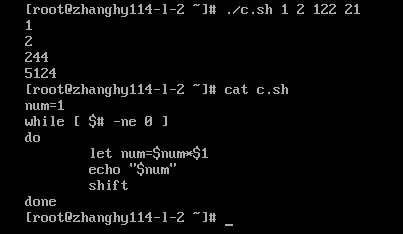
while [ $# -ne 0 ] $#表示位置参数个数

do

let num=$num\*$1

echo $num

done



如下图，可以传递任意参数，每次执行shift 就会删除最左边一个位置参数。

如果不加shift，这就是一个死循环。因为 $#就是代表参数，参数值不等于0就会一直循环

# 正则表达式

用于匹配特定字符

元字符是最重要的概念