昨日作业

```
1.
#!/bin/bash

echo "$1 $2"
var1=$1
var2=$2
temp=$var1
var1=$var2
var2=$temp
echo $var1
echo $var2
2.
#!/bin/bash

cd $1
touch 1.txt 2.txt

var1="hello world"
var2="你好世界"

echo $var1>1.txt
echo $var2>2.txt
```

shell脚本里的数组

shell脚本里面存在数组，但是只存在一维数组

1.数组定义格式

数组名=（成员列表）

2.单个成员的赋值

数组名=（[下标]=数值。。。）

```
#!/bin/bash

#定义一个数组
```

```
arr=(hello nihao hahha)
#指定下标进行数组赋值
arr1=([0]=你好 [2]=hahha [1]=dcjbzf)
#访问数组单个成员使用${数组名[下标]]}}
echo ${arr1[0]} ${arr1[1]} ${arr1[2]}
```

3.数组的性质

a.访问数组单个成员使用${数组名[下标]]}

```
echo ${arr1[0]} ${arr1[1]} ${arr1[2]}
```

b.访问数组所有成员个数使用${数组名[@]}/${数组名[*]}

c. 计算数组成员的个数使用${#数组名[@]}${#数组名[*]}

```
#!/bin/bash

#定义一个数组
arr=(hello nihao hahha)
#指定下标进行数组赋值
arr1=([0]=你好 [2]=hahha [1]=dcjbzf)
#访问数组所有成员
echo ${arr[@]}
echo ${arr1[*]}
#计算数组成员个数
echo arr成员个数为：${#arr[@]}
echo arr1成员个数为：${#arr1[*]}
```

d.计算数组单个成员的长度：${#数组名[下标]}

e.单独修改数组某个成员的数值：

数值名[下标]=数值

```
#!/bin/bash

#定义一个数组
arr=(hell nihao hahhaa)
#指定下标进行数组赋值
arr1=([0]=你好 [2]=hahha [1]=dcjbzf)
#计算数组单个成员的长度
echo ${#arr[0]}
echo ${#arr[1]}
```

```
echo ${#arr[2]}
#单独修改数组某个成员数值
arr[2]=世界
echo ${arr[@]}
```

f.shell脚本里数组的追加和清空

```
#!/bin/bash

#定义一个数组
arr=(hell nihao hahhaa)
echo ${arr[@]}
#数组成员的追加
arr=(${arr[@]} 你好 世界)
echo ${arr[@]}
#数组的清空
unset arr
echo ${arr[@]}

```

练习

在命令行传一些参数进入数组，要求输出数组成员的个数，打印数组所有成员，把数组第一个成员的数值修改为“hello world”,输出数组最后一个成员的长度。

```
#!/bin/bash

#定义一个数组接收命令行参数
arr=($*)
echo ${arr[@]}
#修改第一个成员
arr[0]="hello world"
echo ${arr[@]}
#输出数组最后一个成员的长度
echo ${#arr[$#-1]}
```

shell脚本里的数据运算

shell脚本里面的变量或者数组成员的数值都是字符串，无法实现直接的算数运算，假设想去做算术运算，得需要特定的标识符来帮助我们完成：

1. （（））：只能实现算数运算
2. $[]:只能实现算数运算
3. expr:既可以用来进行算数运算，也可以实现字符串相关的操作

（（））

例子：

```
#!/bin/bash

var1=100
var2=111
#算数运算使用(())
var3=$((($var1+$var2))
echo $var3
```

性质

1.使用(())进行算数运算时，内部变量可以加$，也可以不加，运算符前后可以+空格，也可以不加
2.如果想要把算术运算的结果赋值给一个变量，需要在(())前加$
3.变量可以进行自运算，在进行自运算时，（（））前不用加$
4.变量=$（（表达式1，表达式2，表达式3）），这种写法每一个表达式都会执行，但是取最右边表达式的运算结果赋值给变量

```
#!/bin/bash

var1=100
var2=111
#算数运算使用(())
var3=$((++var1,var2++,var1+var2))
echo $var3
```

5.(())内部可以进行比较复杂的运算，比如循环实现

```
#!/bin/bash

i=0
sum=0
for((i=1;i<=100;i++))
{
        ((sum+=i))
```

```
}
echo $sum
```

练习：

脚本文件里使用一个数组存放10个城市的名字，通过for循环依次输出10个城市的名字

```
#!/bin/bash

i=0
city=(济南 临沂 泰安 济宁 聊城 德州 淄博 威海 枣庄 烟台)
for((i=0;i<10;i++))
{
        echo ${city[$i]}
}
```

$[]

例子：

```
#!/bin/bash

var1=100
var2=111
var3=$[var1+var2]
echo $var3
```

性质

1.$[]进行算数运算时，内部变量可以加$，也可以不加，运算符前后可以+空格，也可以不加

2.在$[]进行算数运算的时候一定要有变量来接收运算结果或者直接输出，否则报错

```
#!/bin/bash

var1=100
var2=111
var3=$[$var1+$var2]
echo $var3
echo $[11+20]
```

3.在$[]内部不可以进行复杂的运算，比如循环

4.在$[]内部进行自加运算时，不在在变量前加$,不然会报错

5.变量=$[表达式1，表达式2，表达式3]，这种写法每一个表达式都会执行，但是取最右边表达式的运算结果赋值给变量

```
#!/bin/bash
var1=100
var2=111
var3=$[var1++,++var2,var1+var2]
echo $var3
```

expr

是一个命令，可以把命令后面运算的结果直接输出在终端

```
#!/bin/bash

var1=100
var2=111

expr $var1 + $var2
```

性质

1.expr使用时在运算符两边一定要加空格，不然无法返回运算的结果

```
expr $var1 + $var2
```

2.expr使用时，变量前一定要加$

3.命令置换符``的使用:会将一个命令的输出结果赋值给一个变量

4.想要把运算结果反馈给一个变量，需要使用命令置换符

```
#!/bin/bash



var=100
var2=11

var1=`expr $var + $var2`
echo $var1
```

5.expr在进行算数运算时，如果是乘法运算，不可以直接用*，如果用了会报错，可以先给‘*’转义再使用

```
#!/bin/bash



var=100
var2=11
#需要对*进行转义，否则解析器会当作通配符处理，进而脚本报错

var1=`expr $var \* $var2`
echo $var1
```

6.使用expr求字符串长度：

expr length $变量名

```
var=helloworld
expr length $var
注意：目标字符串不可以带空格，否则expr语法错误
```

7.使用expr在字符串中查找指定的字符

格式：expr index $变量名 “目标字符”

```
var="helloworld"
expr index $var "l"
```

成功找到对应字符会返回字符在字符串中的位置，否则返回0

8.字符串的裁剪

格式：expr substr $变量名 起始位置 裁剪长度

```
var="helloworld"
expr substr $var 6 5
```

9.字符串的匹配

格式：expr match $变量名 “目标字符串”

```
var="helloworld"
expr match $var "hello"
```

匹配原理：从第一个字符开始按照位置进行匹配，任意一个字符没匹配上就返回0，匹配结束如果成功返回匹配的字符个数

shell脚本里的输入和输出

输入

格式：read 参数 变量名

1. read 变量名：在终端输入一个字符串给指定的变量

2. read 变量列表：在终端给多个变量输入数值

```
、#!/bin/bash



read var var1 var2
echo $var
echo $var1
echo $var2
注意：在进行多个变量数值输入时，会以空格区分字符串，如果最后输入的字符串个数大于变量的个数，会把剩下的字符串都放在最后一个变量里面
```

3. read -p "打印提示信息" 变量名:在输入数值之前打印提示信息

```
#!/bin/bash

read -p "请输入变量数值:" var
echo $var
```

4.read -a 数组名：给数组输入数据

```
#!/bin/bash

read -p "请输入数组数值:" -a arr
echo ${arr[0]}
echo ${arr[*]}
```

5.read -n 数字 变量名：在终端输入指定长度的字符串给变量

```
read -n 5 var
echo $var
```

6.read -s 变量名：输入数据给变量时数值不显示

```
read -s  var
```

```
echo $var
```

7.read -t n 变量名：设置输入超时检测，n秒内不输入程序向下执行

```
#!/bin/bash

read  -t 3 var
echo hello
echo $var
```

参数总结：

-p:提示信息

-a:给数组输入数值

-n:设置数值长度

-s:设置输入回显

-t:设置超时检测

输出

格式：echo 参数 输出目标

参数：

-n:设置输出不换行

```
read  -t 3 var
echo hello
echo -n $var
```

-e:识别转义符

```
echo -e "hello\nhahah\nniaho"
```

shell脚本里的条件判断

补充：

$?:可以存储上一次指令的执行结果过着执行状态

test 判断条件：对指定的条件进行判断

注意：在进行判断时，判断符号两边一定要有空格

1.对于数字的判断

-eq 等于
-ne 不等于
-gt 大于
-ge 大于等于

-lt 小于

-le 小于等于

```
test 12 -ge 11
echo $?
```

2.逻辑判断

逻辑与：-a
逻辑或：-o
逻辑取反！

```
read var1 var2 var3
test $var1 -gt $var2 -o $var1 -gt $var3
echo $?
```

3.判断字符串

-z 判断字符串是否为空 //为空条件满足

```
var1=hello
#判断是否为空
test -z $var1
echo $?
unset var1
test -z $var1
echo $?
```

-n 判断字符串是否不为空 //不为空条件满足

```
#!/bin/bash

var1=hello
#判断是否不为空
#必须使用""把字符串括起来，否则不判断
test -n "$var1"
echo $?
unset var1
test -n "$var1"
echo $?
```

注意：使用-n时把字符串用""括起来，否则逻辑不执行
==或者= 判断字符串是否相等
！=不等于
\> 大于
/< 小于

```
var1=hello
var2=hello
test $var1 \> $var2
echo $?
```

4.文件类型的判断

-e 判断文件是否存在 //存在条件满足
-s 判断文件是否存在，并是否为空 //不为空满足
-b 判断文件是否存在，并是否为块设备文件
-c 判断文件是否存在，并是否是字符设备文件
-d 判断文件是否存在，并是否是目录文件
-f 判断文件是否存在，并是否是普通文件
-L 判断文件是否存在，并是否是链接文件
-S 判断文件是否存在，并是否是套接字文件
-p 判断文件是否存在，并是否是管道文件
判断文件类型时在终端传文件名首先判断是否传过来，再判断类型

```
test -d /home/ubuntu/3/1
echo $?
```

5.对文件权限的判断

-r:判断是否有读权限 //有权限条件满足
-w:是否有写权限
-x:是否有执行权限

```
test -x /home/ubuntu/3/1.c
echo $?
```

shell脚本里的if语句

格式1：

#if test 判断条件
if [判断条件]
then
    语句1
else
    语句2
fi
如果条件满足执行语句1，否则执行语句2

```
#!/bin/bash

#if test -x /home/ubuntu/3/1.c
```

```
if [ -x /home/ubuntu/3/1.c ]
then
        echo 1.c具有执行权限
else
        echo 1.c不具有执行权限
fi
```

格式2：

if [判断条件1]
then
    语句1
elif [判断条件2]
then
    语句2
    …
else
    语句n
fi

满足哪个判断条件就进入哪个语句块

```
#!/bin/bash

#if test -x /home/ubuntu/3/1.c
if [ -d /home/ubuntu/3/1.c ]
then
        echo 1.c是目录文件
elif [ -f /home/ubuntu/3/1.c ]
then
        echo 1.c是普通文件
else
        echo 1.c是其他文件
fi
```

任务

1.复习今日内容

2.在终端输入一个文件名，判断文件是否为空，如果不为空，判断文件是否是普通文件，如果是普通文件，就判断是否具有写权限，没有写权限，就加上写权限，并在文件最后追加一行”hello world “

3.在家目录/home/ubuntu 下创建两个目录文件 file_dir 和 dir_dir,如果家目录下有这两个文件夹，就不需要创建，询问用户是否要对目录清空[Y/N], 输入一个文件路径，判断这个文件路径是否存在，如果存在，把这个目录下的目录文件复制到 dir_dir 中，如果是其他文 件，复制到 file_dir 下，统计复制的文件和目录文件的个数，并打印出来