

## 2 实验二

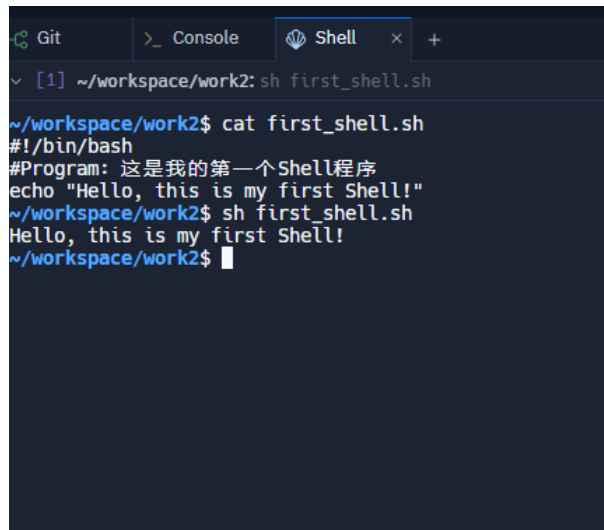
### 2.1 实验内容

- 1、调试 Shell 快速入门中的程序。
- 2、利用 shell 编程，实现 1-10 数字的求和。
- 3、编写一个小程序，实现九九乘法表。
- 4、编写一个 Shell 程序，使得程序在系统时间分钟值为 5 的倍数的时，自动备份此目录下的.png 文件到 backup 中。

### 2.2 实验步骤

#### 2.2.1 调试 Shell 快速入门的程序

- 1、第一个 shell 程序



```
Git  Console  Shell x +
[1] ~/workspace/work2: sh first_shell.sh

~/workspace/work2$ cat first_shell.sh
#!/bin/bash
#Program: 这是我的第一个Shell程序
echo "Hello, this is my first Shell!"
~/workspace/work2$ sh first_shell.sh
Hello, this is my first Shell!
~/workspace/work2$
```

图 14: 第一个 shell 程序

- 2、测试 if 语句, 判断输入是否是正数

```

work2 > >_ test_if.sh

1  #!/bin/bash
2  #Program:测试if语句,判断输入是否是正数
3  #History:2013/10/22 Version1.0
4  read -p "请输入一个数:" temp
5  #if test $temp -gt 0 #测试tset
6  if [ $temp -gt 0 ] #测试[]
7  then
8      echo "您输入是正数"
9  else
10     echo "您输入是非正数"
11 fi
12
13

~/workspace/work2$ cat first_shell.sh
#!/bin/bash
#Program: 这是我的第一个Shell程序
echo "Hello, this is my first Shell!"
~/workspace/work2$ sh first_shell.sh
Hello, this is my first Shell!
~/workspace/work2$ cat test_if.sh
#!/bin/bash
#Program:测试if语句,判断输入是否是正数
#History:2013/10/22 Version1.0
read -p "请输入一个数:" temp
#if test $temp -gt 0 #测试tset
if [ $temp -gt 0 ] #测试[]
then
    echo "您输入是正数"
else
    echo "您输入是非正数"
fi
~/workspace/work2$ sh test_if.sh
请输入一个数:-1
您输入是非正数
~/workspace/work2$
    
```

图 15: 测试 if 语句

### 3、测试 for 语句，输出 1 到 10

```

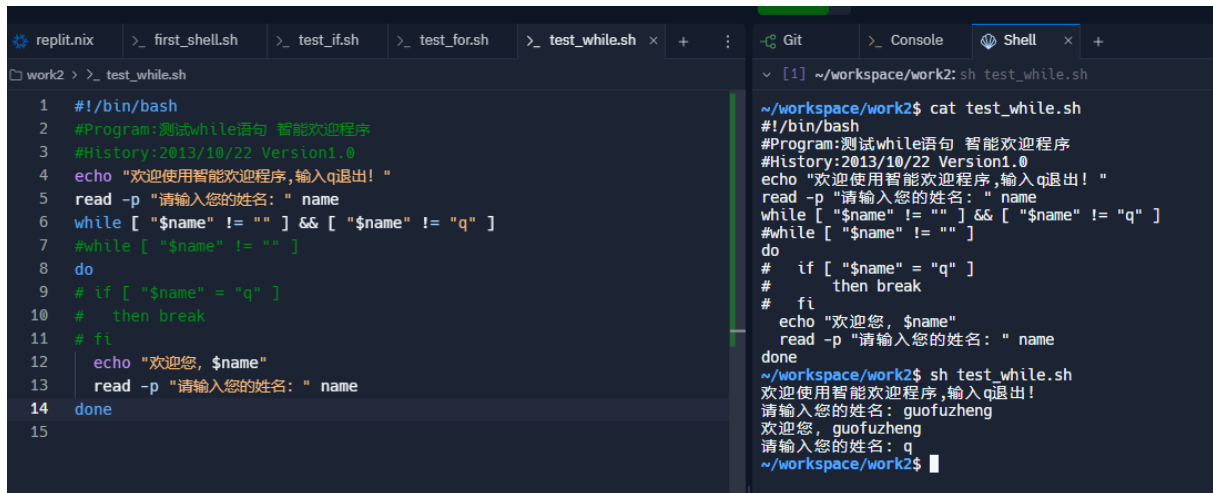
work2 > >_ test_for.sh

1  #!/bin/bash
2  #Program:测试for语句 输出1到10
3  #History:2013/10/22 Version1.0
4  for i in $(seq 10)
5  #for i in $* #运行的时候带参数
6  do
7      echo $i
8  done
9
10

~/workspace/work2$ cat test_for.sh
#!/bin/bash
#Program:测试for语句 输出1到10
#History:2013/10/22 Version1.0
for i in $(seq 10)
#for i in $* #运行的时候带参数
do
    echo $i
done
~/workspace/work2$ sh test_for.sh
1
2
3
4
5
6
7
8
9
10
~/workspace/work2$
    
```

图 16: 测试 for 语句

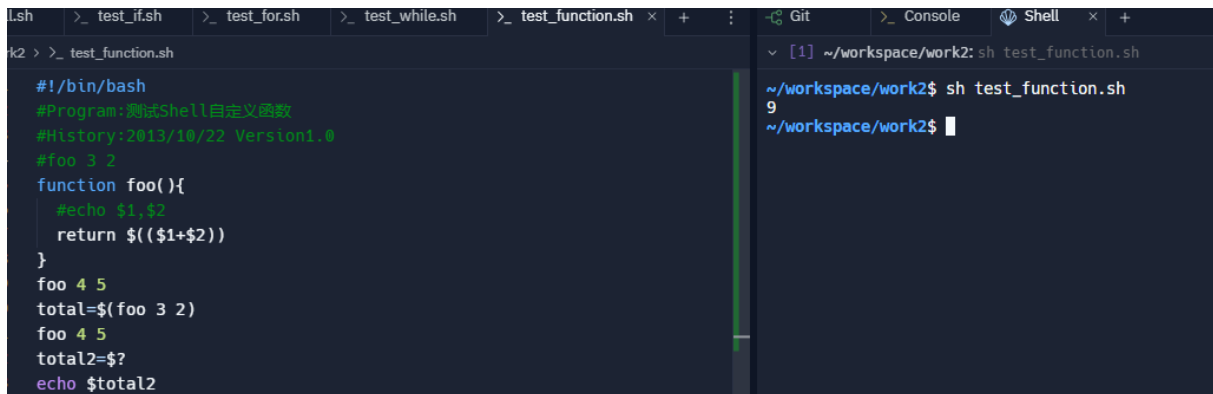
### 4、测试 while 语句，智能欢迎程序



```
replit.nix  >_ first_shell.sh  >_ test_if.sh  >_ test_for.sh  >_ test_while.sh  ×  +  :  - Git  >_ Console  Shell  ×  +  
work2 > >_ test_while.sh  
1  #!/bin/bash  
2  #Program:测试while语句 智能欢迎程序  
3  #History:2013/10/22 Version1.0  
4  echo "欢迎使用智能欢迎程序,输入q退出!"  
5  read -p "请输入您的姓名: " name  
6  while [ "$name" != "" ] && [ "$name" != "q" ]  
7  #while [ "$name" != "" ]  
8  do  
9  # if [ "$name" = "q" ]  
10 # then break  
11 # fi  
12 echo "欢迎您, $name"  
13 read -p "请输入您的姓名: " name  
14 done  
15  
~/workspace/work2$ cat test_while.sh  
#!/bin/bash  
#Program:测试while语句 智能欢迎程序  
#History:2013/10/22 Version1.0  
echo "欢迎使用智能欢迎程序,输入q退出!"  
read -p "请输入您的姓名: " name  
while [ "$name" != "" ] && [ "$name" != "q" ]  
#while [ "$name" != "" ]  
do  
# if [ "$name" = "q" ]  
# then break  
# fi  
echo "欢迎您, $name"  
read -p "请输入您的姓名: " name  
done  
~/workspace/work2$ sh test_while.sh  
欢迎使用智能欢迎程序,输入q退出!  
请输入您的姓名: guofuzheng  
欢迎您, guofuzheng  
请输入您的姓名: q  
~/workspace/work2$
```

图 17: 测试 while 语句

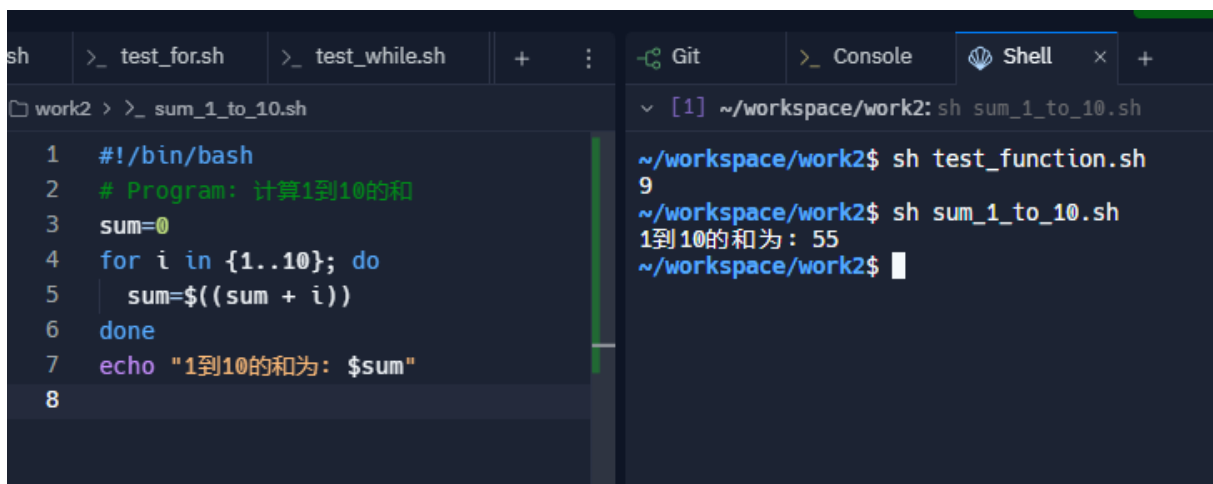
## 5、shell 自定义函数的测试



```
Lsh  >_ test_if.sh  >_ test_for.sh  >_ test_while.sh  >_ test_function.sh  ×  +  :  - Git  >_ Console  Shell  ×  +  
k2 > >_ test_function.sh  
#!/bin/bash  
#Program:测试Shell自定义函数  
#History:2013/10/22 Version1.0  
#foo 3 2  
function foo(){  
#echo $1,$2  
return $((($1+$2))  
}  
foo 4 5  
total=$(foo 3 2)  
foo 4 5  
total2=$?  
echo $total2  
~/workspace/work2$ sh test_function.sh  
9  
~/workspace/work2$
```

图 18: 自定义函数的测试

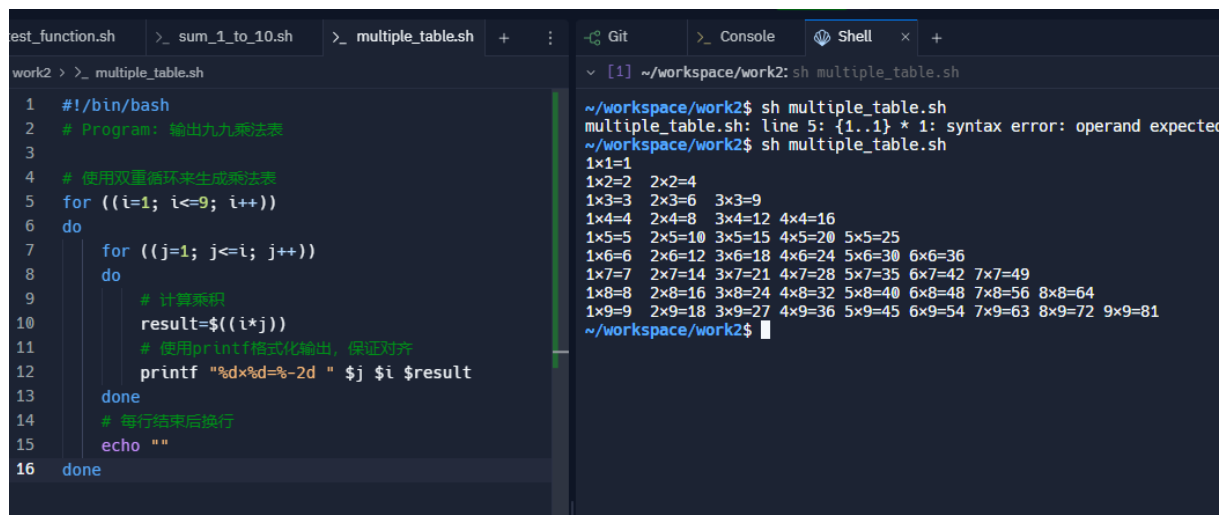
### 2.2.2 shell 实现 1 到 10 的求和



```
sh  >_ test_for.sh  >_ test_while.sh  +  :  - Git  >_ Console  Shell  ×  +  
work2 > >_ sum_1_to_10.sh  
1  #!/bin/bash  
2  # Program: 计算1到10的和  
3  sum=0  
4  for i in {1..10}; do  
5  sum=$((sum + i))  
6  done  
7  echo "1到10的和为: $sum"  
8  
~/workspace/work2$ sh test_function.sh  
9  
~/workspace/work2$ sh sum_1_to_10.sh  
1到10的和为: 55  
~/workspace/work2$
```

图 19: 1 到 10 的求和

### 2.2.3 shell 实现九九乘法表

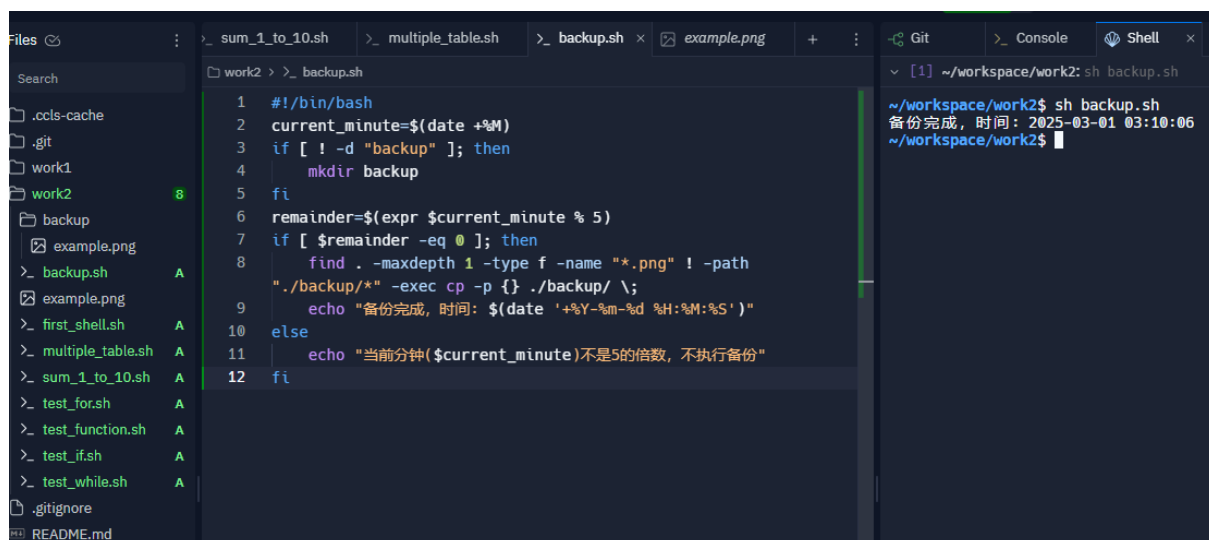


```
test_function.sh > _sum_1_to_10.sh > _multiple_table.sh + : Git > Console Shell x +
work2 > _multiple_table.sh
1  #!/bin/bash
2  # Program: 输出九九乘法表
3
4  # 使用双重循环生成乘法表
5  for ((i=1; i<=9; i++))
6  do
7      for ((j=1; j<=i; j++))
8      do
9          # 计算乘积
10         result=$((i*j))
11         # 使用printf格式化输出, 保证对齐
12         printf "%d×%d=%-2d " $j $i $result
13     done
14     # 每行结束后换行
15     echo ""
16 done

~/workspace/work2$ sh multiple_table.sh
multiple_table.sh: line 5: {1..1} * 1: syntax error: operand expected
~/workspace/work2$ sh multiple_table.sh
1×1=1
1×2=2 2×2=4
1×3=3 2×3=6 3×3=9
1×4=4 2×4=8 3×4=12 4×4=16
1×5=5 2×5=10 3×5=15 4×5=20 5×5=25
1×6=6 2×6=12 3×6=18 4×6=24 5×6=30 6×6=36
1×7=7 2×7=14 3×7=21 4×7=28 5×7=35 6×7=42 7×7=49
1×8=8 2×8=16 3×8=24 4×8=32 5×8=40 6×8=48 7×8=56 8×8=64
1×9=9 2×9=18 3×9=27 4×9=36 5×9=45 6×9=54 7×9=63 8×9=72 9×9=81
~/workspace/work2$
```

图 20: 99 乘法表

### 2.2.4 shell 实现自动备份



```
Files < > _sum_1_to_10.sh > _multiple_table.sh > _backup.sh x example.png + : Git > Console Shell x +
Search
.oclscache
.git
work1
work2
  backup
    example.png
  > backup.sh
  > example.png
  > first_shell.sh
  > multiple_table.sh
  > sum_1_to_10.sh
  > test_for.sh
  > test_function.sh
  > test_if.sh
  > test_while.sh
.gitignore
README.md

work2 > _backup.sh
1  #!/bin/bash
2  current_minute=$(date +%M)
3  if [ ! -d "backup" ]; then
4      mkdir backup
5  fi
6  remainder=$((expr $current_minute % 5))
7  if [ $remainder -eq 0 ]; then
8      find . -maxdepth 1 -type f -name "*.png" ! -path
9      "./backup/*" -exec cp -p {} ./backup/ \;
10     echo "备份完成, 时间: $(date '+%Y-%m-%d %H:%M:%S')"
11 else
12     echo "当前分钟($current_minute)不是5的倍数, 不执行备份"
13 fi

~/workspace/work2$ sh backup.sh
备份完成, 时间: 2025-03-01 03:10:06
~/workspace/work2$
```

图 21: 系统时间为 5 的整数时自动备份 png 文件

## 2.3 实验体会

通过这次实验, 我对 Shell 编程有了更深刻的理解, 也体会到了 Linux 系统中脚本编写的强大与便捷。从调试 Shell 快速入门程序开始, 我逐步熟悉了 Shell 的基本语法和逻辑结构, 无论是 if 语句的条件判断、for 循环的迭代输出, 还是 while 循环的动态交互, 每一步都让我感受到 Shell 脚本在处理简单任务时的效率。在调试第一个 Shell 程序时, 我发现细节非常重要, 例如变量赋值时的空格处理或命令格式的正确性, 一个小小的错误就可能导致程序无法运行, 这让我意识到编程中严谨性的必要。逐步测试 if、for、

while 语句和自定义函数的过程中，我不仅掌握了这些结构的基本用法，还学会了如何通过错误信息定位问题并修正代码，这种实践经验比单纯阅读文档更加宝贵。

实现 1 到 10 的求和让我进一步体会到 Shell 处理数值运算的能力，虽然语法上与 C 语言有所不同，但通过变量累加和循环结构的结合，我成功完成了任务，这让我对 Shell 的灵活性有了新的认识。编写九九乘法表时，我第一次尝试用嵌套循环解决问题，刚开始有些生疏，但通过反复调整 and 测试，最终看到整齐的乘法表输出时，成就感油然而生，这个过程让我明白了逻辑设计的重要性。自动备份程序是实验中最具挑战性的部分，我需要结合系统时间判断和文件操作命令，通过不断查阅资料和试验，我成功实现了每当分钟数为 5 的倍数时自动备份 png 文件到 backup 目录的功能，这让我深刻感受到 Shell 脚本在自动化任务中的实用价值，同时也体会到编写脚本需要耐心和对系统命令的熟悉。