

# 实验报告二：Shell 脚本编写与运行实验

B23041504 何奕洋

## 一、实验目的

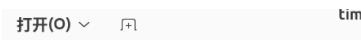
- 掌握 Linux 系统中 Shell 脚本的基础语法体系，包括变量定义与使用、条件判断（if-elif-else、case）、循环结构（for、while）及函数定义与调用的核心逻辑。
- 熟练运用 Linux 文本编辑器（如 vim、nano）编写 Shell 脚本，掌握`chmod`命令修改文件权限的操作，理解文件权限对脚本运行的影响。

## 二、实验环境

- 操作系统：Linux 系统（本次实验采用 Ubuntu 20.04 LTS，通过 VMware 虚拟机搭建）
- 工具软件：终端（Terminal）、文本编辑器（vim）
- 核心命令：`vim`（脚本编写）、`chmod`（权限修改）、`bash`（脚本运行）、`ls -l`（权限查看）、`expr`（整数运算）

## 三、实验内容

- 编写时段判断脚本，通过获取系统当前小时数，结合多条件判断逻辑，输出对应的上午、下午或晚上问候语。



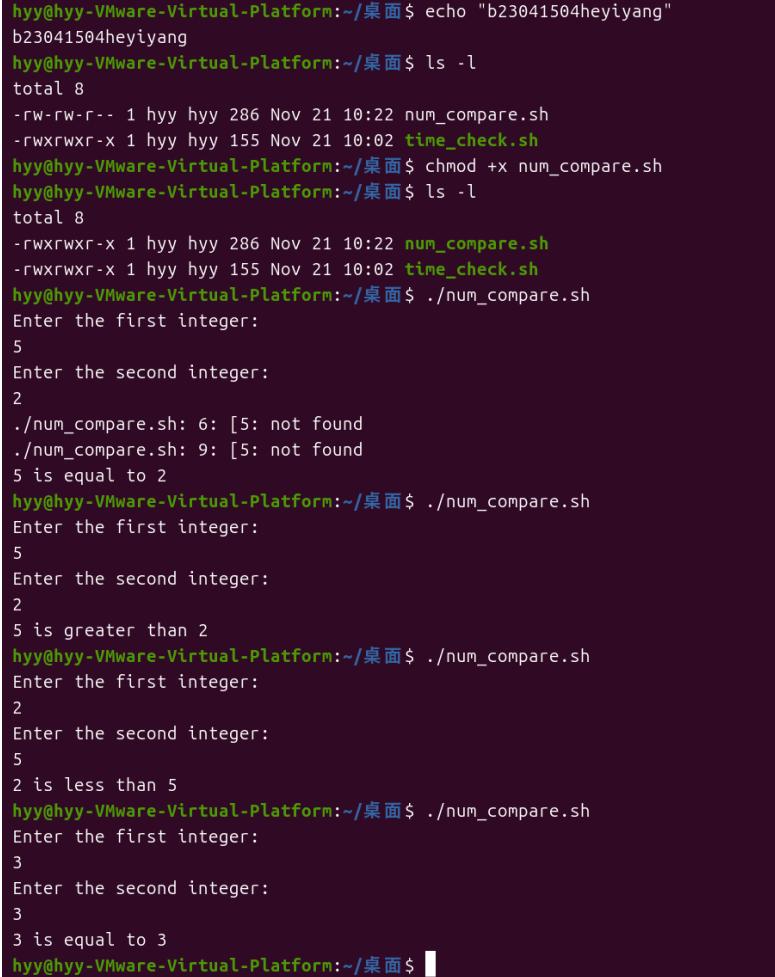
```
#!/bin/bash
hour=`date+%-H`
case $hour in
0[1-9]|1[01])
echo "Good morning !!";;
1[234567])
echo "Good afternoon !!";;
*)
echo "Good evening !!";;
esac
```



```
hyy@hyy-VMware-Virtual-Platform:~/桌面$ echo "b23041504heyiyang"
b23041504heyiyang
hyy@hyy-VMware-Virtual-Platform:~/桌面$ ls
time_check.sh
hyy@hyy-VMware-Virtual-Platform:~/桌面$ ./time_check.sh
bash: ./time_check.sh: 权限不够
hyy@hyy-VMware-Virtual-Platform:~/桌面$ chmod +x time_check.sh
hyy@hyy-VMware-Virtual-Platform:~/桌面$ ls _l
ls: cannot access '_l': No such file or directory
hyy@hyy-VMware-Virtual-Platform:~/桌面$ ls -l
total 4
-rwxrwxr-x 1 hyy hyy 155 Nov 21 10:02 time_check.sh
hyy@hyy-VMware-Virtual-Platform:~/桌面$ ./time_check.sh
Good evening !!
hyy@hyy-VMware-Virtual-Platform:~/桌面$
```

2. 编写整数比较脚本，设计交互输入逻辑，提示用户依次输入两个整数，通过条件判断语句比较两数大小并输出结果。

```
#!/bin/sh
echo "Enter the first integer:"
read first
echo "Enter the second integer:"
read second
if [ "$first" -gt "$second" ]
then
    echo "$first is greater than $second"
elif [ "$first" -lt "$second" ]
then
    echo "$first is less than $second"
else
    echo "$first is equal to $second"
fi
```



```
hyy@hyy-VMware-Virtual-Platform:~/桌面$ echo "b23041504heyiyang"
b23041504heyiyang
hyy@hyy-VMware-Virtual-Platform:~/桌面$ ls -l
total 8
-rw-rw-r-- 1 hyy hyy 286 Nov 21 10:22 num_compare.sh
-rwxrwxr-x 1 hyy hyy 155 Nov 21 10:02 time_check.sh
hyy@hyy-VMware-Virtual-Platform:~/桌面$ chmod +x num_compare.sh
hyy@hyy-VMware-Virtual-Platform:~/桌面$ ls -l
total 8
-rwxrwxr-x 1 hyy hyy 286 Nov 21 10:22 num_compare.sh
-rwxrwxr-x 1 hyy hyy 155 Nov 21 10:02 time_check.sh
hyy@hyy-VMware-Virtual-Platform:~/桌面$ ./num_compare.sh
Enter the first integer:
5
Enter the second integer:
2
./num_compare.sh: 6: [5: not found
./num_compare.sh: 9: [5: not found
5 is greater than 2
hyy@hyy-VMware-Virtual-Platform:~/桌面$ ./num_compare.sh
Enter the first integer:
5
Enter the second integer:
2
5 is less than 5
hyy@hyy-VMware-Virtual-Platform:~/桌面$ ./num_compare.sh
Enter the first integer:
3
Enter the second integer:
3
3 is equal to 3
hyy@hyy-VMware-Virtual-Platform:~/桌面$
```

3. 编写最小值查找脚本，预设固定数字列表，通过循环遍历列表元素，结合条件判断筛选出列表中的最小值并输出。

```

#!/bin/bash
smallest=10000
for i in 8 2 18 0 -3 87
do
if test $i -lt $smallest
then
smallest=$i
fi
done
echo $smallest

hyy@hyy-VMware-Virtual-Platform:~/桌面$ echo "b23041504heyiyang"
b23041504heyiyang
hyy@hyy-VMware-Virtual-Platform:~/桌面$ ls -l
total 12
-rw-rw-r-- 1 hyy hyy 122 Nov 21 10:32 find_min.sh
-rwxrwxr-x 1 hyy hyy 290 Nov 21 10:25 num_compare.sh
-rwxrwxr-x 1 hyy hyy 155 Nov 21 10:02 time_check.sh
hyy@hyy-VMware-Virtual-Platform:~/桌面$ bash./find_min.sh
bash: bash./find_min.sh: 没有那个文件或目录
hyy@hyy-VMware-Virtual-Platform:~/桌面$ bash find_min.sh
-3
hyy@hyy-VMware-Virtual-Platform:~/桌面$

```

4. 编写可执行文件统计脚本，遍历当前目录下所有文件，通过权限判断筛选出具有可执行权限的文件，利用计数器统计其总数并输出。

```

#!/bin/bash
count=0
for i in *
do
if test -x $i
then
count=`expr $count + 1`
fi
done
echo Total of $count files executable

```

```

hyy@hyy-VMware-Virtual-Platform:~/桌面$ echo "b23041504heyiyang"
b23041504heyiyang
hyy@hyy-VMware-Virtual-Platform:~/桌面$ ls -l
total 16
-rw-rw-r-- 1 hyy hyy 124 Nov 21 10:43 count_exec.sh
-rw-rw-r-- 1 hyy hyy 122 Nov 21 10:32 find_min.sh
-rwxrwxr-x 1 hyy hyy 290 Nov 21 10:25 num_compare.sh
-rwxrwxr-x 1 hyy hyy 155 Nov 21 10:02 time_check.sh
hyy@hyy-VMware-Virtual-Platform:~/桌面$ bash count_exec.sh
Total of expr $count +1 files executable
hyy@hyy-VMware-Virtual-Platform:~/桌面$ bash count_exec.sh
Total of 2 files executable
hyy@hyy-VMware-Virtual-Platform:~/桌面$

```

5. 编写质数判断脚本，定义质数判断函数，通过命令行传参接收待判断数字，调用函数后输出该数字是否为质数的结果。

```

#!/bin/bash
prime()
{
    flag=1
    j=2
    while [ $j -le `expr $1 / 2` ]
    do
        if [ `expr $1 % $j` -eq 0 ]
        then
            flag=0
            break
        fi
        j=`expr $j + 1`
    done
    if [ $flag -eq 1 ]
    then
        return 1
    else
        return 0
    fi
}

prime $1
if [ $? -eq 1 ]
then
    echo "$1 is a prime!"
else
    echo "$1 is not a prime!"
fi

```

```

hyy@hyy-VMware-Virtual-Platform:~/桌面$ echo "b23041504heyiyang"
b23041504heyiyang
hyy@hyy-VMware-Virtual-Platform:~/桌面$ ls -l
total 20
-rw-rw-r-- 1 hyy hyy 125 Nov 21 10:45 count_exec.sh
-rw-rw-r-- 1 hyy hyy 122 Nov 21 10:32 find_min.sh
-rwxrwxr-x 1 hyy hyy 290 Nov 21 10:25 num_compare.sh
-rw-rw-r-- 1 hyy hyy 298 Nov 21 10:57 prime_check.sh
-rwxrwxr-x 1 hyy hyy 155 Nov 21 10:02 time_check.sh
hyy@hyy-VMware-Virtual-Platform:~/桌面$ bash prime_check.sh
prime_check.sh: 行 23: prime: 未找到命令
is not a prime!
hyy@hyy-VMware-Virtual-Platform:~/桌面$ bash prime_check.sh
expr: syntax error: unexpected argument '2'
prime_check.sh: 第 6 行: [: 2: 需要一元运算符
is a prime!
hyy@hyy-VMware-Virtual-Platform:~/桌面$ bash prime_check.sh 9
9 is not a prime!
hyy@hyy-VMware-Virtual-Platform:~/桌面$ bash prime_check.sh 7
7 is a prime!
hyy@hyy-VMware-Virtual-Platform:~/桌面$ 

```

## 四、实验总结

### (一) 错误类型及原因

- 语法错误：变量赋值时等号两侧添加空格（Shell 脚本语法要求等号两侧无空格）、case 语句结尾关键字大小写错误（需小写`esac`）、函数定义时括号两侧添加空格、运算符两侧缺少空格（`expr` 运算要求运算符前后必须空格）、引号使用不匹配（中文引号与英文引号混用）

及函数调用时拼写错误（如将`prime`误写为`prine`）。

2. 权限错误：脚本编写完成后未添加执行权限，直接运行时出现“Permission denied”提示，原因是 Linux 系统中新建文件默认无执行权限，需手动授权。
3. 逻辑错误与运行异常：部分脚本缺少参数校验，未传参时导致变量为空，引发运算错误；变量大小写使用不一致，导致变量无法正确解析；循环结构中变量自增逻辑错误，导致循环无法正常终止或判断结果异常；边界值处理缺失，如质数判断脚本未考虑 0、1 等非质数边界值，导致判断结果错误。

## （二）错误解决方法

1. 语法错误解决：严格遵循 Shell 脚本语法规范，编写时注意关键字大小写、括号与引号的正确使用，运算符号两侧预留空格；脚本编写完成后，可先通过`bash -n 脚本名.sh`命令检查语法正确性，提前规避语法错误。
2. 权限错误解决：使用`chmod +x 脚本名.sh`命令为脚本添加执行权限，授权后通过`ls -l 脚本名.sh`命令查看权限字段是否含`x`，确认授权成功后再运行脚本。
3. 逻辑错误与运行异常解决：运行脚本时确保按要求传参，避免变量为空；统一变量命名规范，确保变量大小写一致；检查循环结构中变量自增逻辑，确保循环能够正常终止；补充边界值处理逻辑，针对特殊值（如 0、1、负数）单独设计判断条件；函数调用前仔细核对函数名拼写，避免因拼写错误导致函数无法调用。

## （三）实验体会

本次实验通过实际开发 5 个 Shell 脚本，深入掌握了 Shell 脚本的基础语法与应用技巧，同时对 Linux 系统的权限管理、命令行操作有了更深刻的理解。实验过程中遇到的各类错误，让我认识到 Shell 脚本对语法规范性的严格要求，以及逻辑严谨性对程序运行结果的重要影响。