



中国海洋大学

## 系统开发工具基础实验报告

上课时间:	周五 1-4 节
姓 名:	张仕达
学 号:	23010022094
指导老师:	周小伟

# 1 实验内容

## 1.1 主题 1: shell 编程: 自定义无参带返回值函数

### 1.1.1 内容

定义一个函数, 求两个数的最大值

脚本内容:

```
[root@localhost ~]# cat batch.sh
#!/bin/bash
sum()
{
    read -p "enter the first number:" n2
    read -p "enter the second number:" n1
    return $((n1+n2))
}
sum
echo "the sum of the two numbers is $?"
```

### 1.1.2 结果

```
the sum of the two numbers is 35
[root@localhost ~]# sh batch.sh
enter the first number:5
enter the second number:30
the sum of the two numbers is 35
```

## 1.2 主题 2: shell 编程: 自定义有参函数

### 1.2.1 内容

定义一个函数, 返回指定参数

```
#!/bin/bash
funParam()
{
    echo "the first parameter is: $1"
    echo "the third parameter is: $3"
    echo "there are $# parameters"
}
funParam 2 5 9 7 4 6
```

### 1.2.2 结果

```
batch.sh 22) 1070 1070000
[root@localhost ~]# sh batch.sh
the first parameter is: 2
the third parameter is: 9
there are 6 parameters
```

## 1.3 主题 3: shell 工具: cut

### 1.3.1 内容

切割提取指定列数据

文本内容:

cu1.txt:

11 22 333

aa bb ccc

44 55 666

aa bb ccc

命令: 按空格来分割切取第一列和第三列的内容

cut cu1.txt -d " " -f 1,3

### 1.3.2 结果

```
[root@localhost ~]# cat cu1.txt
11 22 333
aa bb ccc
44 55 666
aa bb ccc
[root@localhost ~]# cut cu1.txt -d " " -f 1,3
11 333
aa ccc
44 666
aa ccc
```

## 1.4 主题 4: shell 工具: sed

### 1.4.1 内容

向指定行号前或后添加数据

文本内容:

cu1.txt:

11 22 333

aa bb ccc

44 55 666

aa bb ccc

命令：向第三行前添加 qqg”

sed "3iqqq" cu1.txt

3 表示行号，i 表示向前添加

命令：向第三行后添加”qqg”

sed "3aqqg" cu1.txt

3 表示行号，a 表示向后添加

### 1.4.2 结果

```
[root@localhost ~]# sed "3aqqg" cu1.txt
11 22 333
aa bb ccc
44 55 666
qqg
aa bb ccc
```

```
[root@localhost ~]# sed "3iqqq" cu1.txt
11 22 333
aa bb ccc
qqg
44 55 666
aa bb ccc
```

## 1.5 主题 5: shell 工具: awk

### 1.5.1 内容

awk 是一种文本分析工具，默认按照每行空格切割数据

命令：

```
# echo "aaa 11 22 ds" | awk '{print $1"&"$2"&"$3}'
```

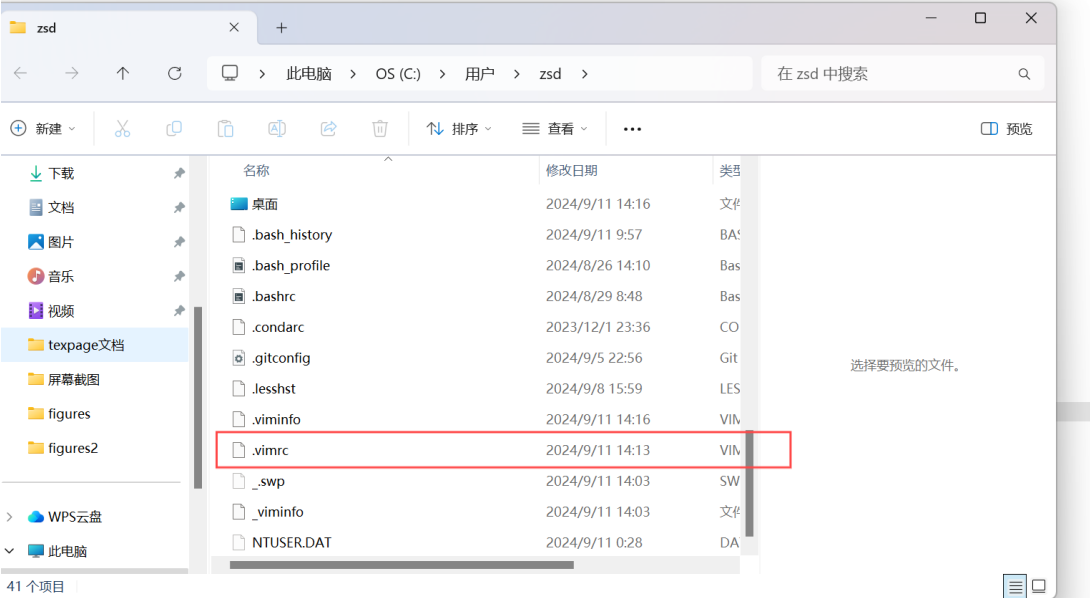
### 1.5.2 结果

```
[root@localhost ~]# echo "aaa 11 22 ds" | awk '{print $1"&"$2"&"$3}'
aaa&11&22
```

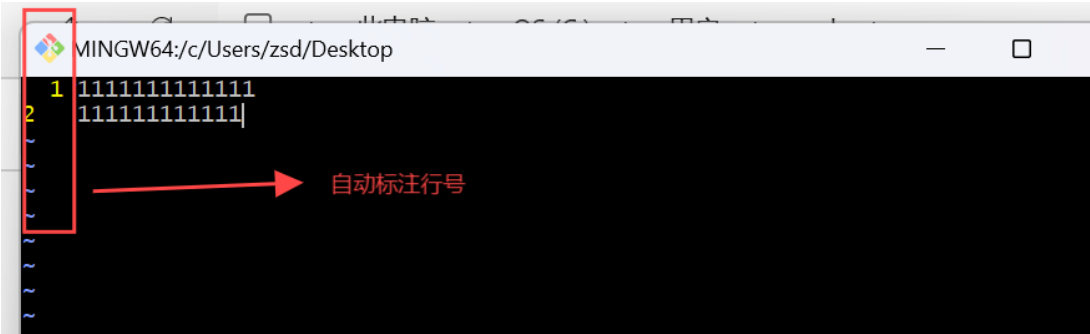
1.6 主题 6: vim: vimrc 的配置

1.6.1 内容

将.vimrc 文件放在用户的目录下面，就会对 vim 进行配置，配置后 vim 的功能将更加完善。



1.6.2 结果



1.7 主题 7: 数据处理

1.7.1 内容

查询文件中含有”aa” 的行数据

文本内容:

cu1.txt:

11 22 333

aa bb ccc

44 55 666

aa bb ccc

命令:

sed -n '/aa/p' cu1.txt

### 1.7.2 结果

```
[root@localhost ~]# cat cu1.txt
11 22 333
aa bb ccc
44 55 666
aa bb ccc
[root@localhost ~]# sed -n '/aa/p' cu1.txt
aa bb ccc
aa bb ccc
```

## 1.8 主题 8：数据处理：查询一个文件（file.txt）中空行所在的行号

### 1.8.1 内容

脚本实现

1.file.txt 数据准备

test test

111 111

222 222

333 333

2. 脚本实现

1. 检查 file.txt 文件是否存在

2. 使用 grep 查找空行，并打印行号

```
[root@localhost ~]# cat batch.sh
#!/bin/bash
if [! -f "file.txt"];then
    echo "Error: file.txt does not exist."
    exit 1
fi
grep -n '^$' file.txt
```

### 1.8.2 结果

```
[root@localhost ~]# cat file.txt
file file

111 111
222 222

333 333
[root@localhost ~]# sh batch.sh
batch.sh: line 2: [! : command not found
2:
5:
```

## 1.9 主题 9：数据处理：对一个文件中的每一行第一个数字进行排序

### 1.9.1 内容

#### 1.file.txt 数据准备

```
[root@localhost ~]# cat file.txt
1 2
9 8
8 7
3 6
6 4
4 8
7 1
2 9
5 3
```

#### 2. 脚本实现

检查 file.txt 文件是否存在

使用 awk 提取数字行，并通过 sort 命令进行排序，最后输出结果

假设每行都是数字或者包含数字（但只关注第一个数字）

```
[root@localhost ~]# cat batch.sh
#!/bin/bash
if [! -f "file.txt"];then
    echo "Error: file.txt does not exist."
    exit 1
fi
awk '{print $1}' file.txt | sort -n
```

### 1.9.2 结果

```
[root@localhost ~]# sh batch.sh
batch.sh: line 2: [!: command not found
1
2
3
4
5
6
7
8
9
```

## 1.10 主题 10：数据处理：查找 root 下所有包含“111”的文件名称

### 1.10.1 内容

脚本实现

1. 使用 find 命令来遍历根目录
2. 使用 chmod +x 脚本名命令给予该脚本执行权限。

```
"batch.sh" 4L, 75C written
[root@localhost ~]# cat batch.sh
#!/bin/bash
find / -type f -name "*111*" > files_with_111.txt
echo "done"
```

### 1.10.2 结果

```
[root@localhost ~]# sh batch.sh
done
[root@localhost ~]# cat files_with_111.txt
/sys/kernel/debug/tracing/trace_stat/function111
/root/files_with_111.txt
/usr/lib/modules/3.10.0-957.el7.x86_64/kernel/drivers/i2c/busses/i2c-amd8111.ko.xz
/usr/lib/modules/3.10.0-957.el7.x86_64/kernel/drivers/media/usb/dvb-usb-v2/dvb-usb-mxl111sf.ko.xz
/usr/lib/modules/3.10.0-957.el7.x86_64/kernel/drivers/media/usb/dvb-usb-v2/mxl111sf-demod.ko.xz
/usr/lib/modules/3.10.0-957.el7.x86_64/kernel/drivers/media/usb/dvb-usb-v2/mxl111sf-tuner.ko.xz
/usr/lib/modules/3.10.0-957.el7.x86_64/kernel/drivers/net/ethernet/amd/amd8111e.ko.xz
/usr/lib/firmware/moxa/moxa-1110.fw
/usr/lib64/gconv/IBM1112.so
[root@localhost ~]#
```



## 1.11 主题 11：数据处理：输出文件内长度大于 3 的单词

### 1.11.1 内容

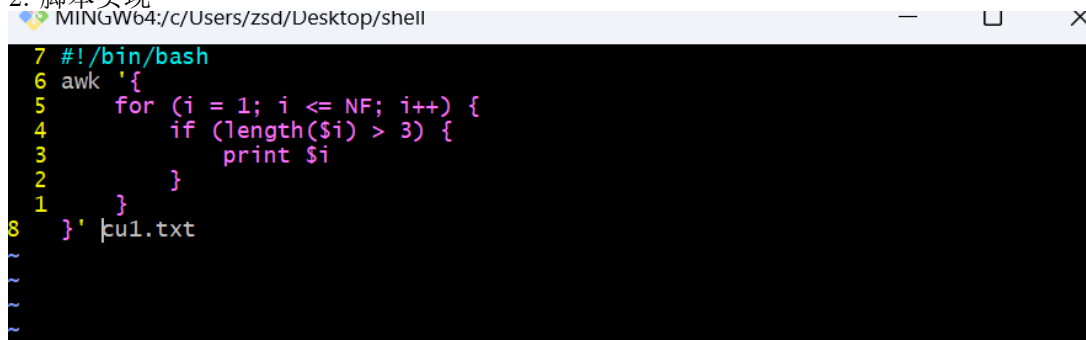
#### 1. 数据准备

aaar aaas asd

r

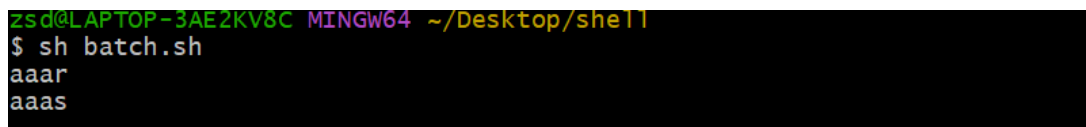
aaf as

#### 2. 脚本实现



```
7 #!/bin/bash
6 awk '{
5     for (i = 1; i <= NF; i++) {
4         if (length($i) > 3) {
3             print $i
2         }
1     }
8 }' pu1.txt
```

### 1.11.2 结果

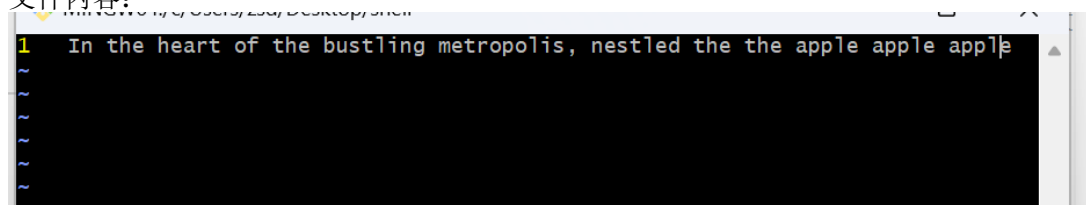


```
zsd@LAPTOP-3AE2KV8C MINGW64 ~/Desktop/shell
$ sh batch.sh
aaar
aaas
```

## 1.12 主题 12：数据处理：单词去重排序

### 1.12.1 内容

文件内容：



```
1 In the heart of the bustling metropolis, nestled the the apple apple apple
```

脚本内容：

```
#!/bin/bash
awk '{
    for (i = 1; i <= NF; i++) {
        # 使用gsub函数移除单词中的标点符号，转换为小写以统一处理
        word = tolower($i)
        gsub(/[[[:punct:]]/], "", word)
        # 累加单词计数
        count[word]++
    }
}
END {
    for (word in count) {
        print word, count[word]
    }
}' cu1.txt
```

### 1.12.2 结果

```
$ sh batch.sh
nestled 1
metropolis 1
apple 3
of 1
the 4
in 1
heart 1
bustling 1
```

## 1.13 主题 13: shell 脚本编程：批量生成文件

### 1.13.1 内容

```
#!/bin/bash
read -t 30 -p "please enter the num of the files you want to creat:" n
test=$(echo $n | sed 's/[0-9]//g')
if [ -n "$n" -a -z "$test" ]
then
    for ((i=0;i<$n;i++))
    do
        name=$(date +%N)
        [ ! -d ./temp ] && mkdir -p ./temp
        touch "./temp/$name"
        echo "creat $name success!"
    done
else
    echo "false"
    exit 1
fi
```

### 1.13.2 结果

```
[root@localhost ~]# ll temp
ls: cannot access temp: No such file or directory
[root@localhost ~]# sh batch.sh
please enter the num of the files you want to creat:5
creat 693683758 success!
creat 695365585 success!
creat 696487289 success!
creat 697548152 success!
creat 698479625 success!
[root@localhost ~]# ll temp
total 0
-rw-r--r--. 1 root root 0 Sep  6 23:41 693683758
-rw-r--r--. 1 root root 0 Sep  6 23:41 695365585
-rw-r--r--. 1 root root 0 Sep  6 23:41 696487289
-rw-r--r--. 1 root root 0 Sep  6 23:41 697548152
-rw-r--r--. 1 root root 0 Sep  6 23:41 698479625
```

## 1.14 主题 14: shell 脚本编程: 猜数字游戏

### 1.14.1 内容

脚本内容:

```
1 #!/bin/bash
2 num=$((RANDOM%100+1))
3 echo $num
4 # 使用 read 提示用户猜数字
5 # 使用 if 判断用户猜数字的大小关系:-eq(等于),-ne(不等于),-gt(大于),-ge(大于等于),
6 # -lt(小于),-le(小于等于)
7 while :
8 do
9   read -p "计算机生成了一个 1-100 的随机数,你猜: " cai
10  if [ $cai -eq $num ]
11  then
12    echo "恭喜,猜对了"
13    exit
14  elif [ $cai -gt $num ]
15  then
16    echo "Oops,猜大了"
17  else
18    echo "Oops,猜小了"
19  fi
20 done
```

### 1.14.2 结果

```
$ sh batch.sh
计算机生成了一个 1-100 的随机数,你猜: 54
Oops,猜小了
计算机生成了一个 1-100 的随机数,你猜: 65
Oops,猜小了
计算机生成了一个 1-100 的随机数,你猜: 88
Oops,猜大了
计算机生成了一个 1-100 的随机数,你猜: 77
恭喜,猜对了
```

## 1.15 主题 15: shell 脚本编程: 输入三个数并进行升序排序

### 1.15.1 内容

```
12 #!/bin/bash
11 read -p "请输入一个整数:" num1
10 read -p "请输入一个整数:" num2
9 read -p "请输入一个整数:" num3
8 tmp=0
7
6 if [ $num1 -gt $num2 ];then
5 tmp=$num1
4 num1=$num2
3 num2=$tmp
2 fi
1 # 如果 num1 大于 num3,就把 num1 和 num3 对调,确保 num1 变量中存的
13
1 if [ $num1 -gt $num3 ];then
2 tmp=$num1
3 num1=$num3
4 num3=$tmp
5 fi
6 if [ $num2 -gt $num3 ];then
7 tmp=$num2
8 num2=$num3
9 num3=$tmp
10 fi
11 echo "排序后数据(从小到大)为:$num1,$num2,$num3"
```

### 1.15.2 结果

```
$ sh batch.sh
请输入一个整数:10
请输入一个整数:30
请输入一个整数:80
排序后数据(从小到大)为:10,30,80
```

## 1.16 主题 16: shell 脚本编程: 石头、剪刀、布游戏

### 1.16.1 内容

```
#!/bin/bash
game=(石头 剪刀 布)
num=$((RANDOM%3))
computer=$((game[num]))
echo "请根据下列提示选择您的出拳手势"
echo "1. 石头"
echo "2. 剪刀"
echo "3. 布"
read -p "请选择 1-3:" person
case $person in
1)
if [ $num -eq 0 ]
then
echo "平局"
elif [ $num -eq 1 ]
then
echo "你赢"
else
echo "计算机赢"
fi;;
2)
if [ $num -eq 0 ]
then
echo "计算机赢"
elif [ $num -eq 1 ]
then
echo "平局"
else
echo "你赢"
fi;;
3)
if [ $num -eq 0 ]
then
echo "你赢"
elif [ $num -eq 1 ]
then
echo "计算机赢"
else
echo "平局"
fi;;
*)
echo "必须输入 1-3 的数字"
esac
```

### 1.16.2 结果

```
$ sh batch.sh
请根据下列提示选择您的出拳手势
1. 石头
2. 剪刀
3. 布
请选择 1-3:2
你赢
```

## 1.17 主题 17: shell 脚本编程: 打印乘法口诀表

### 1.17.1 内容

```
#!/bin/bash
for i in `seq 9`
do
for j in `seq $i`
do
echo -n "$j*$i=${i*j} "
done
echo
done
```

### 1.17.2 结果

```
$ sh batch.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
```

## 1.18 主题 18: shell 脚本编程: 对 1 到 100 进行求和

### 1.18.1 内容

```
#!/bin/bash
sum=0
for i in `seq 100`
do
    sum=$((sum+i))
done
echo "总和是:$sum"
```

### 1.18.2 结果

```
$ sh batch.sh
总和是:5050
```

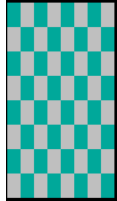
## 1.19 主题 19: shell 脚本编程: 打印国际象棋棋盘

### 1.19.1 内容

```
#!/bin/bash
# 打印国际象棋棋盘
# 设置两个变量,i 和 j,一个代表行,一个代表列,国际象棋为 8*8 棋盘
# i=1 是代表准备打印第一行棋盘,第 1 行棋盘有灰色和蓝色间隔输出,总共
# 为 8 列
# i=1,j=1 代表第 1 行的第 1 列;i=2,j=3 代表第 2 行的第 3 列
# 棋盘的规律是 i+j 如果是偶数,就打印蓝色色块,如果是奇数就打印灰色色
# 块
# 使用 echo -ne 打印色块,并且打印完成色块后不自动换行,在同一行继续输
# 出其他色块
for i in {1..8}
do
    for j in {1..8}
    do
        sum=$((i+j))
        if [ $((sum%2)) -eq 0 ];then
            echo -ne "\033[46m \033[0m"
        else
            echo -ne "\033[47m \033[0m"
        fi
    done
    echo
done
```

### 1.19.2 结果

```
$ sh batch.sh
```



## 1.20 主题 20: 判断用户输入字符类型

### 1.20.1 内容

```
#!/bin/bash
# 判断用户输入的数据类型(字母、数字或其他)
read -p "请输入一个字符:" KEY

# 使用正则表达式检查输入
if [[ "$KEY" =~ ^[a-zA-Z]$ ]]; then
    echo "字母"
elif [[ "$KEY" =~ ^[0-9]$ ]]; then
    echo "数字"
else
    echo "空格、功能键、其他控制字符或不是单个字符的输入"
fi
```

### 1.20.2 结果

```
sd@LAPTOP-3AE2KV8C MINGW64 ~/Desktop/shell
$ sh batch.sh
请输入一个字符:3
数字

sd@LAPTOP-3AE2KV8C MINGW64 ~/Desktop/shell
$ sh batch.sh
请输入一个字符:j
字母

sd@LAPTOP-3AE2KV8C MINGW64 ~/Desktop/shell
$ sh batch.sh
请输入一个字符:
空格、功能键、其他控制字符或不是单个字符的输入
```



## 2 解题感悟

shell 编程的学习让我对文件的操作更加熟练了，再加上 vim 的使用，我现在能够更加方便的对各种文件进行修改。类似编程、功能却更加偏实用的脚本则让我对于代码的兴趣更加的浓厚。这次课程的学习让我受益匪浅。

[github 地址](#) [点击]