南昌大学实验报告

姓名：Qing Liu

学号：6130116184

邮箱地址：1119637652@qq.com

专业班级：Class 164 of Computer Science and Technology

实验日期：THU. April 11st, 2019

课程名称：Cloud Computing Technology Experiments

实验项目名称

Basic Interface and Programming

实验目的

- To write shell scripts to solve problems
- To implement some standard Linux-kernel utilities such as ls,cp,etc using system calls
- Understanding the idea of multi-programming (or multiplexing) and threading
- Learn to read an English poem
- Learn to read instructions, carefully

实验基础

- **Hardware**: Lenovo Ideapad 700 - 15ISK
- **Software**: Windows 10 Home Edition, VMware Workstation 15 PRO, Ubuntu 18.04 LTS

实验步骤

Write shell scripts to solve problems

1. Write a Shell script that accepts a filename, starting and ending line numbers as arguments and displays all the lines between the given line numbers.

```
#!/bin/bash
# 1.sh
```

```
FILENAME=$1
START=$2
END=$3

if [ $FILENAME = "" ]; then
        echo "Please input the filename"
        exit 1
fi

if [ $START = "" ]; then
        echo "Please input the start line number"
        exit 1
fi

if [ $END = "" ]; then
        echo "Please input the end line number"
        exit 1
fi

if [ $END -lt $START ]; then
        echo "The end line number is less than the start line number"
        exit 1
fi

SUM=`wc -l < $FILENAME`

M=$((SUM-END+1))
N=$((END-START+1))
tail -$M $FILENAME | head -$N
exit 0
```

2. Write a Shell script that deletes all lines containing a specified word in one or more files supplied as arguments to it.

```
#!/bin/bash
# 2.sh

if [ $# = 0 ]; then
        echo "Please input the specific word"
        exit 1
fi

if [ $# = 1 ]; then
        echo "Please input at least one filename"
        exit 1
fi

WORD=$1

while [ -n $2 ]; do
        FILENAME=$2
```

```
        ALIAS=`echo "$FILENAME.bak"`
        grep -v $WORD < $FILENAME > $ALIAS
        rm -f $FILENAME
        mv $ALIAS $FILENAME
        shift
done

exit 0
```

3. Write a Shell script that displays list of all the files in the current directory to which the user has read, write and execute permissions.

```
#!/bin/bash
# 3.sh

for file in `ls`; do
        if [ -x $file ] && [ -r $file ] && [ -x $file ]; then
                echo $file
        fi
done
```

4. Write a Shell script that receives any number of file names as arguments checks if every argument supplied is a file or a directory and reports accordingly. Whenever the argument is a file, the number of lines on it is also reported.

```
#!/bin/bash
# 4.sh

if [ $# -eq 0 ]; then
        echo "Arguments Error, please input at least 1 file or directory name"
        exit 1
fi

while [ -n $1 ]; do
        if [ -d $1 ]; then
                echo "$1 is a directory!"
        else
                line=`wc -l <$1`
                echo "$1 is a file which contains $line lines"
        fi
        shift
done
```

5. Write a Shell script that accepts a list of file names as its arguments, counts and reports the occurrence of each word that is present in the first argument file on other argument files.

```
#!/bin/bash
# 5.sh

FIRSTFILE=$1
declare -a words
declare -a files
declare -a counts
shift

# the number of other files
FILENUM=0
# the number of words in first file
WORDNUM=0


# save file names except the first file into array files
while [[ -n $1 ]]; do
        files[$FILENUM]=$1
        shift
        FILENUM=$((FILENUM+1))
done


# save every word into array words in the first file without repeating
BAK=$IFS
IFS=$(echo -en "\ \n\t\b\.\?,")
for word in `cat $FIRSTFILE`; do
        flag=0
        for((i = 0; i < $WORDNUM; i++)); do
                if [ $word = ${words[$i]} ]; then
                        flag=1
                        break
                fi
        done

        if [ $flag = 1 ]; then
                continue
        fi

        words[$WORDNUM]=$word

        WORDNUM=$((WORDNUM+1))
done

# count the words in the first file
for((i = 0; i < $WORDNUM; i++)); do
        counts[$i]=0
        for((j = 0; j < $FILENUM; j++)); do
                for w in `cat ${files[$j]}`; do
                        if [ $w = ${words[$i]} ]; then
                                c=${counts[$i]}
                                c=$((c+1))
                                counts[$i]=$c
```

```
                        fi
                done
        done
done

IFS=$BAK

# output
for((i = 0; i < $WORDNUM; i++));do
        echo "${words[$i]}: ${counts[$i]}"
done
```

6. Write a Shell script to list all of the directory files in a directory.

```
#!/bin/bash
# 6.sh

echo -n "Please input the directiry name: "
read DIR

if [ ! -d $DIR ]; then
        echo "$DIR is not a directry file!"
else
        for d in `ls $DIR`; do
                if [ -d "$DIR/$d" ]; then
                        echo $d
                fi
        done
fi
```

7. Write a Shell script to find factorial of a given integer.

```
#!/bin/bash
# 7.sh

echo -n "Please input an integer(>=0):"
read NUM
if [ NUM -lt 0 ]; then
        echo "The number you give is a negative number!"
elif [ NUM -eq 0 ];then
        echo "The factorial of 0 is 1"
else
        result=1
        for((i = 2; i <= $NUM; i++)); do
                result=$((result*i))
        done
        echo "The factorial of $NUM is $result"
fi
```

Test for the written scripts

Here is the test code file.

```
#!/bin/bash
# test

# add execute premission to every sub script
for((i = 1; i <= 7; i++)); do
        chmod +x $i.sh


while true; do
        clear
        echo "================================================="
        echo "***              Test for 1 to 7 .sh            ***"
        echo "                  1 - Test for 1.sh                "
        echo "                  2 - Test for 2.sh                "
        echo "                  3 - Test for 3.sh                "
        echo "                  4 - Test for 4.sh                "
        echo "                  5 - Test for 5.sh                "
        echo "                  6 - Test for 6.sh                "
        echo "                  7 - Test for 7.sh                "
        echo "                  0 - Back                         "
        echo "================================================="
        echo -n "Please input the number of the script:"
        read NUM
        case $NUM in
                # display 2 - 6 lines of the file poem
                1)./1.sh poem 2 6;;
                # delete the lines contain string
                # "hello" in file test1 and test2
                2)./2.sh hello test1 test2;;
                # display list of all files in the
                # current directory which the user has
                # read, write and execute premisions
                3)./3.sh ;;
                # check dir, 1.sh, 2.sh 3.sh
                # dir/file1 is file or directory
                4)./4.sh dir 1.sh 2.sh 3.sh dir/file1;;
                # counts and reports the occurrence
                # of each word that is present in poem
                # on test1 and test2
                5)./5.sh poem test1 test2;;
                # list all of the directory files in dir
                6)./6.sh dir;;
                # find factorial of 6
                7)./7.sh;;
                0)
                        clear
                        echo "BACKING..."
                        sleep 2;;
```

```
                *)
                        echo "Error number of script $NUM"
                        sleep 2;;
        esac
done
```

实验数据或结果

All the file contents for test

- file `test1`

```
A apple is on the desk.
If you want to eat it, you have to take it by yourself.
int function(){ int x = 5; return x + 3; }
hello world
which
thread safe though reentrant
reentrant reentrant though reentrant
```

- file `test2`

```
saved x it reentrant tough
depends global
hello world
v fjdk fjkgj
gjtr git
```

- file `poem`

```
A reentrant function,
if interrupted,
will return a result,
which is not perturbed.
int global_int;int is_not_reentrant(int x){ int x = x; return global_int + x; };
depends on a global variable,
which may change during execution.
int global_int;int is_reentrant(int x){ int saved = global_int; return saved + x
;},
mitigates external dependency,
it is reentrant, though not thread safe.
```

The result of test for all scripts and the screen shots

- **Test number 1**

  In the test code, it displays 2 to 6 line of the file `poem` .

```
========================================================
***            Test for 1 to 7 scripts              ***
                 1 - Test for 1.sh
                 2 - Test for 2.sh
                 3 - Test for 3.sh
                 4 - Test for 4.sh
                 5 - Test for 5.sh
                 6 - Test for 6.sh
                 7 - Test for 7.sh
                 0 - BACK
========================================================
Please input the number of the script:1
depends on a global variable,
which may change during execution.
int global_int;int is_reentrant(int x){ int saved = global_int; return saved + x
;},
mitigates external dependency,
```

- **Test number 2**

  Delete the lines contain string "hello" in file `test1` and `test2`.

```
========================================================
***            Test for 1 to 7 scripts              ***
                 1 - Test for 1.sh
                 2 - Test for 2.sh
                 3 - Test for 3.sh
                 4 - Test for 4.sh
                 5 - Test for 5.sh
                 6 - Test for 6.sh
                 7 - Test for 7.sh
                 0 - BACK
========================================================
Please input the number of the script:2
```

  There is no any output information. But we can use `cat` to display the content of file `test1` and `test2`

```
cleo@vm-ubuntu:~/lab2$ cat test1
A apple is on the desk.
If you want to eat it, you have to take it by yourself.
int function(){ int x = 5; return x + 3; }
which
thread safe though reentrant
reentrant reentrant though reentrant
cleo@vm-ubuntu:~/lab2$ cat test2
saved x it reentrant tough
depends global
v fjdk fjkgj
gjtr git
cleo@vm-ubuntu:~/lab2$
```

  We can see that the line contains "hello" has been deleted.

- **Test number 3**

Display list of all files in the current directory which the user has read, write and execute premisions.

```
============================================================
***             Test for 1 to 7 scripts                ***
                  1 - Test for 1.sh
                  2 - Test for 2.sh
                  3 - Test for 3.sh
                  4 - Test for 4.sh
                  5 - Test for 5.sh
                  6 - Test for 6.sh
                  7 - Test for 7.sh
                  0 - BACK
============================================================
Please input the number of the script:3
1.sh
2.sh
3.sh
4.sh
5.sh
6.sh
7.sh
dir
test0.sh
test.sh
============================================================
```

We can compare to the screen shot of command `ls -l`

```
cleo@vm-ubuntu:~/lab2$ ls -l
total 52
-rwxr-xr-x 1 cleo cleo  474 4月  11 20:40 1.sh
-rwxr-xr-x 1 cleo cleo  331 4月  11 11:14 2.sh
-rwxr-xr-x 1 cleo cleo  111 4月  11 21:14 3.sh
-rwxr-xr-x 1 cleo cleo  283 4月  11 14:41 4.sh
-rwxr-xr-x 1 cleo cleo 1031 4月  11 18:12 5.sh
-rwxr-xr-x 1 cleo cleo  215 4月  11 14:51 6.sh
-rwxr-xr-x 1 cleo cleo  317 4月  11 15:01 7.sh
drwxr-xr-x 5 cleo cleo 4096 4月  11 18:44 dir
-rw-r--r-- 1 cleo cleo  387 4月  11 17:10 poem
-rwxr-xr-x 1 cleo cleo 1181 4月  11 20:49 test0.sh
-rw-r--r-- 1 cleo cleo  195 4月  11 20:58 test1
-rw-r--r-- 1 cleo cleo   65 4月  11 20:58 test2
-rwxr-xr-x 1 cleo cleo 1147 4月  11 20:38 test.sh
```

- **Test number 4**

  Check dir, 1.sh, 2.sh 3.sh dir/file1 is file or directory

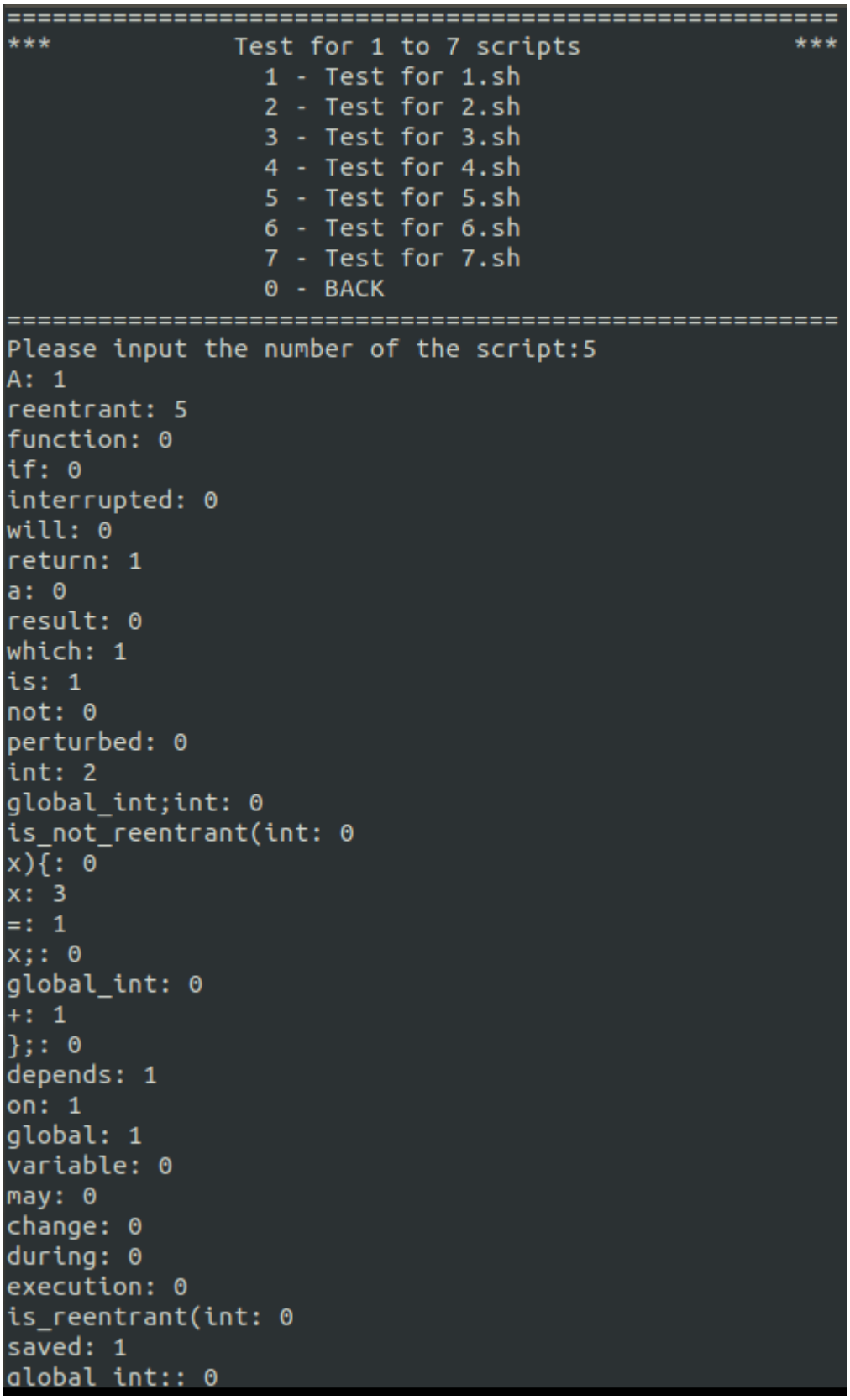
```
==========================================================
***            Test for 1 to 7 scripts                ***
                 1 - Test for 1.sh
                 2 - Test for 2.sh
                 3 - Test for 3.sh
                 4 - Test for 4.sh
                 5 - Test for 5.sh
                 6 - Test for 6.sh
                 7 - Test for 7.sh
                 0 - BACK
==========================================================
Please input the number of the script:4
dir is a directory!
1.sh is a file which contains 32 lines
2.sh is a file which contains 24 lines
3.sh is a file which contains 7 lines
dir/file1 is a file which contains 0 lines
==========================================================
```

We can compare to this picture:

```
cleo@vm-ubuntu:~/lab2$ wc -l 1.sh
32 1.sh
cleo@vm-ubuntu:~/lab2$ wc -l 2.sh
24 2.sh
cleo@vm-ubuntu:~/lab2$ wc -l 3.sh
7 3.sh
cleo@vm-ubuntu:~/lab2$ wc -l dir/file1
0 dir/file1
cleo@vm-ubuntu:~/lab2$ ls -ld dir
drwxr-xr-x 5 cleo cleo 4096 4月  11 18:44 dir
cleo@vm-ubuntu:~/lab2$
```

- **Test number 5**

  Counts and reports the occurrence of each word that is present in poem on test1 and test2. I just set the seperater as `\`, `\n`, `\t`, `.`, `,`, so there are maybe some non-word characters.



- **Test number 6**

  List all of the directory files in dir.

```
==========================================================
***            Test for 1 to 7 scripts                ***
                 1 - Test for 1.sh
                 2 - Test for 2.sh
                 3 - Test for 3.sh
                 4 - Test for 4.sh
                 5 - Test for 5.sh
                 6 - Test for 6.sh
                 7 - Test for 7.sh
                 0 - BACK
==========================================================
Please input the number of the script:6
Please input the directiry name: dir
subdir1
subdir2
subdir3
==========================================================
```

We can use `ls -l dir` to compare the results.

```
cleo@vm-ubuntu:~/lab2$ ls -l dir
total 12
-rw-r--r-- 1 cleo cleo    0 4月  11 12:49 file1
-rw-r--r-- 1 cleo cleo    0 4月  11 18:44 file2
drwxr-xr-x 2 cleo cleo 4096 4月  11 14:49 subdir1
drwxr-xr-x 2 cleo cleo 4096 4月  11 14:49 subdir2
drwxr-xr-x 2 cleo cleo 4096 4月  11 18:44 subdir3
```

- **Test number 7**

  Find factorial of 6.

```
==========================================================
***            Test for 1 to 7 scripts                ***
                 1 - Test for 1.sh
                 2 - Test for 2.sh
                 3 - Test for 3.sh
                 4 - Test for 4.sh
                 5 - Test for 5.sh
                 6 - Test for 6.sh
                 7 - Test for 7.sh
                 0 - BACK
==========================================================
Please input the number of the script:7
Please input an integer(>=0):6
The factorial of 6 is 720
```

实验思考

It is this term when we are learning Linux OS. We were tought Shell Program at 2nd week but we didn't get practices about it. I think this is a good practice and I became more familiar with the Shell Program.

Here are some contents I didn't know when I look through the homework page and I can figure it now.

- **The meaning of "reentrant kernel"**

  A re-entrant kernel enables a process (and it’s threads) to give away the CPU while in kernel mode. They do not hinder other processes from also entering kernel mode. This behavior allows CPU to be shared among multiple processes.

  A typical use case is IO wait. The process wants to read a file. It calls a kernel function for this. Inside the kernel function, the disk controller is asked for the data. Getting the data will take some time and the function is blocked during that time.

  With a re-entrant kernel, the scheduler will assign the CPU to another process (kernel thread) until an interrupt from the disk controller indicates that the data is available and our thread can be resumed. This process can still access IO (which needs kernel functions), like user input. The system stays responsive and CPU time waste due to IO wait is reduced.

  All Unix kernels are reentrant. This means that several processes may be executing in Kernel Mode at the same time. Of course, on uniprocessor systems, only one process can progress, but many can be blocked in Kernel Mode when waiting for the CPU or the completion of some I/O operation.

  If a hardware interrupt occurs, a reentrant kernel is able to suspend the current running process even if that process is in Kernel Mode. This capability is very important, because it improves the throughput of the device controllers that issue interrupts.

- **Understanding for "reentrant function"**

  A function is said to be reentrant if there is a provision to interrupt the function in the course of execution, service the interrupt service routine and then resume the earlier going on function, without hampering its earlier course of action. Reentrant functions are used in applications like hardware interrupt handling, recursion, etc.

  The function has to satisfy certain conditions to be called as reentrant:

  1. It may not use global and static data. Though there are no restrictions, but it is generally not advised. because the interrupt may change certain global values and resuming the course of action of the reentrant function with the new data may give undesired results.

  2. It should not modify it’s own code. This is important because the course of action of the function should remain the same throughout the code. But, this may be allowed in case the interrupt routine uses a local copy of the reentrant function every time it uses different values or before and after the interrupt.

  3. Should not call another non-reentrant function.

- **Thread safety and Reentrant functions**

  Reentrancy is distinct from, but closely related to, thread-safety. A function can be thread-safe and still not reentrant. For example, a function could be wrapped all around with a mutex (which avoids problems in multithreading environments), but if that function is used in an interrupt service routine, it could starve waiting for the first execution to release the mutex. The key for avoiding confusion is

that reentrant refers to only one thread executing. It is a concept from the time when no multitasking operating systems existed.

- **Questions**

  When I run the `test.sh`, there is an error, and I don't know what happend. I just edit it in vim. I have to edit the file again named `test0.sh`

```
cleo@vm-ubuntu:~/lab2$ ./test.sh
./test.sh: line 43: syntax error: unexpected end of file
```

参考资料

- https://devhints.io/bash
- https://linuxconfig.org/bash-scripting-tutorial-for-beginners
- https://www.quora.com/What-is-meant-by-a-reentrant-kernel
- https://www.geeksforgeeks.org/reentrant-function/