

浙江大学实验报告 2

课程名称: Linux 应用技术基础

实验类型: 综合型

实验项目名称: 实验 2 程序设计

学生姓名: 王俊 专业: 海洋工程与技术 学号: 3170100186

电子邮件地址: 596954871@qq.com

实验日期: 2021 年 1 月 12 日

一、实验环境

计算机配置:

处理器 1.4 GHz Intel Core i5

内存 16G

显卡 Intel Iris Plus Graphics 645 1536 MB

操作系统环境: 版本: macOS Mojave 10.14.6

Linux 版本: ubuntu-16.04, kernel: 4.15.0-129-generic

二、实验内容和结果及分析

//实验程序源代码（包括注释），程序必要的文档或说明；上机实验输入输出显示的结果截图

1. (15 分) 编写一个 shell 脚本程序，它带一个命令行参数，这个参数是一个文件名。如果这个文件是一个普通文件，则打印文件所有者的名字和最后的修改日期。如果程序带有多个参数，则输出出错信息。

Code:

```
#!/bin/bash
#To ensure the input with one parameter
if test $# -ne 1;
then
    echo "Please just input one parameter!!!"
    exit 1
fi

filename=$1
if test -f "$filename" #check the file
then #if it is a normal file
    set -- $(ls -l $filename)
    echo "The filename is $filename." #print the filename
    echo "The owner is ${3}." #print the owner
    echo "The last edit time is $6 $7 $8." #print the last edit time
    exit 0
fi

#if it is not a normal file
echo "$filename is not a normal file."
exit 1
```

输入一个参数

```
gakiara@ubuntu:~/Lab2$ bash 1.sh 1.txt
The filename is 1.txt.
The owner is gakiara.
The last edit time is Jan 12 16:39.
```

输入两个参数

```
gakiara@ubuntu:~/Lab2$ bash 1.sh 1.txt 2.txt
Please just input one parameter!!!
```

输入一个非普通文件的参数

```
gakiara@ubuntu:~/Lab2$ bash 1.sh ~/tmp1
/home/gakiara/tmp1 is not a normal file.
```

2. (15 分) 编写 shell 程序，统计指定目录下的普通文件、子目录及可执行文件的数目，统计该目录下所有普通文件字节数总和，目录的路径名字由参数传入。

```
#!/bin/bash
#To ensure the input with one parameter
if test $# -gt 1;
then
    echo "Please just input one parameter!!!"
    exit 1
fi

directory=$1

#If the parameter is empty, the parameter is current directory
echo -en "Directroy: "
if test $# -lt 1; then
    pwd
else
    echo $directory
fi

#Count the number of different types of files in the directory, including hidden files.
#the first character is the file type, '-' is a normal file, and 'd' is a directory.
echo -en "Normal files: "
ls -Al $directory | grep -c "^-"
echo -en "Sub-directory: "
ls -Al $directory | grep -c "^d"

#If there is x or s in the file permissions, the file is executable.
echo -en "Executable: "
ls -Al $directory | grep -c "^-[rw-]*[sx]"

# Count the total number of bytes of all ordinary files in this directory.
declare -i sum=0
#Enter the directory
cd $directory

for x in $(ls)
do
    if test -f "$x"
    then
        set -- $(ls -Al $x)
        ((sum=sum + "$5"))
    fi
done
echo "Total size (in byte): $sum"

#exit the program
exit 0
```

输入两个参数:

```
gakiara@ubuntu:~/Lab2$ bash 2.sh ~ ~/tmp
Please just input one parameter!!!
```

查找主目录, 输入一个参数

```
gakiara@ubuntu:~/Lab2$ bash 2.sh ~
Directroy: /home/gakiara
Normal files: 24
Sub-directory: 31
Executable: 7
Total size (in byte): 735879930
```

缺省默认当前目录

```
gakiara@ubuntu:~/Lab2$ bash 2.sh
Directroy: /home/gakiara/Lab2
Normal files: 6
Sub-directory: 0
Executable: 0
Total size (in byte): 735879930
```

3. (15 分) 编写一个 shell 脚本，输入一个字符串，忽略（删除）非字母后，检测该字符串是否为回文(palindrome)。对于一个字符串，如果从前向后读和从后向前读都是同一个字符串，则称之为回文串。例如，单词“mom”，“dad”和“noon”都是回文串。

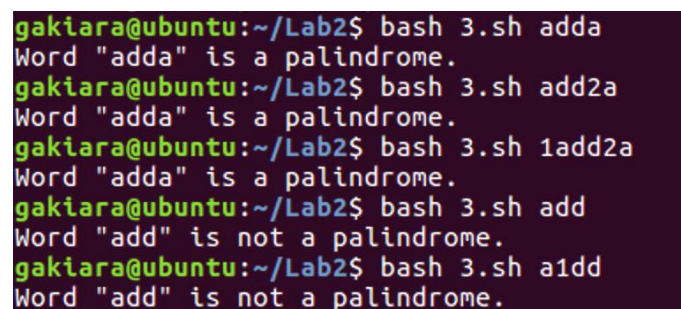
```
#!/bin/bash
# To ensure the input with one parameter
if test $# -ne 1
then
    if test $# -lt 1 ; then
        echo -n "Missing argument! "
    elif test $# -gt 1 ; then
        echo -n "Too many arguments! "
    fi
    exit 1
fi

str=${1//[!a-zA-Z]}          # Only keep alphabetic characters
revstr=`echo $str | rev`     # Generate the reversed string of the string

# If the original string = reversed string, output "Yes", otherwise output "No"
if [[ "$str" = "$revstr" ]]
then
    echo "Word \"$str\" is a palindrome."
else
    echo "Word \"$str\" is not a palindrome."
fi

#exit the program
exit 0
```

输入回文字符串和非回文字符串并在其中夹杂着数字，结果如图所示：



```
gakiara@ubuntu:~/Lab2$ bash 3.sh adda
Word "adda" is a palindrome.
gakiara@ubuntu:~/Lab2$ bash 3.sh add2a
Word "adda" is a palindrome.
gakiara@ubuntu:~/Lab2$ bash 3.sh 1add2a
Word "adda" is a palindrome.
gakiara@ubuntu:~/Lab2$ bash 3.sh add
Word "add" is not a palindrome.
gakiara@ubuntu:~/Lab2$ bash 3.sh a1dd
Word "add" is not a palindrome.
```

4. (15 分) 编写一个 shell 脚本，把当前目录下文件大小大于 100K 的文件全部移动到~/tmp/ 目录下。

```
#!/bin/bash
# Traverse the files in the current directory and find the size which is greater 100KB
for Filename in $(ls -l | awk '$5 > 102400 {print $9}')
do
    # remove the file to the ~/tmp
    mv $Filename ~/tmp
done
# tell the user it is finished
echo "Done!"
```

输入查找当前比 100K 大的文件，运行后可以看到文件全部移动到了 ~/tmp/

```

gakiara@ubuntu:~/Lab2$ ls -l | awk '$5 > 102400{print $9}'
myfs
test2
gakiara@ubuntu:~/Lab2$ ls -l ~/tmp | awk '$5 > 102400{print $9}'
gakiara@ubuntu:~/Lab2$ sudo bash 4.sh
Done!
gakiara@ubuntu:~/Lab2$ ls -l ~/tmp | awk '$5 > 102400{print $9}'
myfs
test2

```

5. (30 分) 编写一个实现文件备份和同步的 shell 脚本程序 dirsinc。程序的参数是两个需要备份同步的目录，如：

`dirsinc ~\dir1 ~\dir2` # ~\dir1 为源目录，~\dir2 为目标目录

dirsinc 程序实现两个目录内的所有文件和子目录（递归所有的子目录）内容保持一致。程序基本功能如下。

- 1) 备份功能：目标目录将使用来自源目录的最新文件，新文件和新子目录进行升级，源目录将保持不变。dirsinc 程序能够实现增量备份。
- 2) 同步功能：两个方向上的旧文件都将被最新文件替换，新文件都将被双向复制。源目录被删除的文件和子目录，目标目录也要对应删除。
- 3) 其它功能自行添加设计。

```

#!/bin/bash
usage() # To show the correct format
{
    echo "Usage: $0 <mode:backup|sync> source_dir target_dir"
}

usage #To tell the userh how to use it

# If the number of input parameters is not 3, show error!
if [ $# -ne 3 ] ; then
    if [ $# -lt 3 ] ; then
        echo -n "Missing argument(s)! "
    elif [ $# -gt 3 ] ; then
        echo -n "Too many arguments! "
    fi
    usage
    exit 1
fi

# If the first parameter is not "backup" or "sync", an error will be reported
if [ "$1" != "backup" -a "$1" != "sync" ] ; then
    echo -n "Mode not recognized. "
    usage
    exit 1
fi

# The input source directory does not exist, an error is reported
if [ ! -d "$2" ] ; then
    echo "$0: Source dir '$2': No such directory"
    exit 1
fi

```



```

mode=$1
src=$2; tgt=$3
# Accept two parameters: source folder, target folder, recursively delete files in the target folder that do not
exist in the source folder
rmExtraFiles()
{
    for file in $2/* ; do # Traverse all files in the target folder
        if [ ! -e "${file/$2/$1}" ] ; then # If the file does not exist in the source folder, delete it
            rm -rf "$file"
        elif [ -d "$file" ] ; then #If the file is a directory, recursively perform a check delete operation
            rmExtraFiles "${file/$2/$1}" "$file"
        fi
    done
}

sync() # Synchronize function
{
    [ ! -d "$2" ] && echo "Target directory '$2' does not exist. Creating target dir..." && mkdir -p "$2"
    rmExtraFiles "$1" "$2" # First remove the extra files in the target folder
    # Update the destination folder with the files from the source folder (-u: Updated files will only be copied)
    cp -Trup "$1" "$2"
    # Update the source folder with files from the destination folder
    cp -Trup "$2" "$1"
}

case $mode in
    backup) # backup -function
        echo "Backing up $src -> $tgt..." # If the destination directory does not exist, create a new directory
        [ ! -d "$tgt" ] && echo "Target directory '$tgt' does not exist. Creating target dir..." && mkdir -p "$tgt"
        # copy SRC directory recursively (-r), attribute reserved (-p) to TGT directory (-t: treat DEST as a normal
        file), and when the source file update/destination directory does not have this file to copy (-u), from this achieve
        backup function
        cp -Trup "$src" "$tgt" && echo "Backup from $src to $tgt completed!" || (echo "$0: Backup failed"; exit 1)
        ;;
    sync) # Synchronize function
        echo "Syncing $src <-> $tgt..."
        sync "$src" "$tgt" && echo "Sync $src and $tgt completed!" || (echo "$0: Sync failed"; exit 1)
        ;;
    *)
        echo -n "$0: Mode error! "
        usage
        exit 1
        ;;
esac
exit 0

```

备份功能的

测试 1:

可以看到 dir1 里的 2.txt 是最新的，然后备份后 dir2 里的就变成最新的了

```

gakiara@ubuntu:~/Lab2$ sudo bash cmpNew.sh ./dir1/2.txt ./dir2/2.txt
./dir1/2.txt is newer than ./dir2/2.txt
gakiara@ubuntu:~/Lab2$ sudo bash 5.sh backup ./dir1 ./dir2
Backing up ./dir1 -> ./dir2...
Backup from ./dir1 to ./dir2 completed!
gakiara@ubuntu:~/Lab2$ sudo bash cmpNew.sh ./dir1/2.txt ./dir2/2.txt
./dir2/2.txt is newer than ./dir1/2.txt

```

测试 2:

可以看到 dir1 里的 2.txt 是最新的，然后备份后 dir2 里的就不需要更新了

```

gakiara@ubuntu:~/Lab2$ sudo bash cmpNew.sh ./dir1/2.txt ./dir2/2.txt
./dir2/2.txt is newer than ./dir1/2.txt
gakiara@ubuntu:~/Lab2$ sudo bash 5.sh backup ./dir1 ./dir2
Backing up ./dir1 -> ./dir2...
Backup from ./dir1 to ./dir2 completed!
gakiara@ubuntu:~/Lab2$ sudo bash cmpNew.sh ./dir1/2.txt ./dir2/2.txt
./dir2/2.txt is newer than ./dir1/2.txt

```

同步功能的

测试:

Dir1 中的 1.txt 是最新的; hello

Dir2 中的 2.txt 是最新的: hello

可以看到同步后, 两个本来没有内容的旧文件被更新成 hello 了

```
gakiara@ubuntu:~/Lab2$ sudo bash cmpNew.sh ./dir1/2.txt ./dir2/2.txt
./dir1/2.txt:
./dir2/2.txt:
hello
./dir2/2.txt is newer than ./dir1/2.txt
gakiara@ubuntu:~/Lab2$ sudo bash cmpNew.sh ./dir1/1.txt ./dir2/1.txt
./dir1/1.txt:
hello
./dir2/1.txt:
./dir1/1.txt is newer than ./dir2/1.txt
gakiara@ubuntu:~/Lab2$ sudo bash cmpNew.sh ./dir1/2.txt ./dir2/2.txt
./dir1/2.txt:
./dir2/2.txt:
hello
./dir2/2.txt is newer than ./dir1/2.txt
gakiara@ubuntu:~/Lab2$ sudo bash 5.sh sync ./dir1 ./dir2
Syncing ./dir1 <-> ./dir2...
Sync ./dir1 and ./dir2 completed!
gakiara@ubuntu:~/Lab2$ sudo bash cmpNew.sh ./dir1/1.txt ./dir2/1.txt
./dir1/1.txt:
hello
./dir2/1.txt:
hello
./dir2/1.txt is newer than ./dir1/1.txt
gakiara@ubuntu:~/Lab2$ sudo bash cmpNew.sh ./dir1/2.txt ./dir2/2.txt
./dir1/2.txt:
hello
./dir2/2.txt:
hello
./dir2/2.txt is newer than ./dir1/2.txt
```

提示: 不能使用现有的备份或同步程序, 如: /usr/bin/rsync

选做题 (4 分, 如果平时成绩没有满分, 可计入平时成绩) :

用 C 语言写一个名字为 myls 程序, 实现类似 Linux 的 ls 命令, 其中 myls 命令必须实现 -a、-l、-i 等选项的功能。要求 myls 程序使用系统调用函数编写, 不能使用 exec 系统调用或 system() 函数等调用 ls 命令来实现。命令 man ls 可以得到更多 ls 选项的含义。有用的系统调用: stat(), opendir(), readdir() 和 getcwd() 等。

(完成本题的有关知识请参考教材第 5 章)

Myls.c:

The print function

```
#include <sys/stat.h>
#include <stdio.h>
#include <string.h>
#include <dirent.h>
#include <time.h>
#include <pwd.h>
#include <grp.h>

void print(struct stat *st){
    switch (st->st_mode & S_IFMT) {
        case S_IFBLK: printf("b"); break; //块设备文件
        case S_IFCHR: printf("c"); break; //字符设备文件
        case S_IFDIR: printf("d"); break; //目录文件
        case S_IFIFO: printf("p"); break; //管道文件
        case S_IFLNK: printf("l"); break; //符号链接文件
        case S_IFSOCK: printf("s\n"); break; //socket
        default: printf("-"); break; //普通文件
    }

    if (st->st_mode & S_IRUSR) printf("r"); //文件所有者具可读取权限
    else printf("-");
    if (st->st_mode & S_IWUSR) printf("w"); //文件所有者具可写入权限
    else printf("-");
    if (st->st_mode & S_IXUSR) printf("x"); //文件所有者具可执行权限
    else printf("-");
    if (st->st_mode & S_IRGRP) printf("r"); //所有者所在的用户组具可读取权限
    else printf("-");
    if (st->st_mode & S_IWGRP) printf("w"); //所有者所在的用户组具可写入权限
    else printf("-");

    if (st->st_mode & S_IXGRP) printf("x"); //所有者所在的用户组具可执行权限
    else printf("-");
    if (st->st_mode & S_IROTH) printf("r"); //其他用户组具可读取权限
    else printf("-");
    if (st->st_mode & S_IWOTH) printf("w"); //其他用户组具可写入权限
    else printf("-");
    if (st->st_mode & S_IXOTH) printf("x"); //其他用户组具可执行权限
    else printf("-");

    printf(" %3ld ", (long)st->st_nlink); //链接数
    struct passwd *pwd = getpwuid(st->st_uid); //用户名
    printf(" %6s ", pwd->pw_name);
    struct group *grp = getgrgid(st->st_gid); //组名
    printf(" %6s ", grp->gr_name);
    printf(" %6ld ", (long)st->st_size); //大小
    printf(" %s", ctime(&st->st_mtime)); //最后修改时间
}
```

Main function:


```

int main(int argc, char* argv[]){
    DIR *dir;
    struct dirent *item;
    struct stat buf;
    int i, count=0;
    dir = opendir("./"); //打开当前目录并建立一个目录流
    if(strcmp(argv[1], "ls") != 0) { //如果输入参数不是 ls
        printf("cannot recognize the command!\n");
        return 1;
    }
}

```

With no option:

```

//no option
if(argc == 2){ // 如果输入参数仅有一个 ls
    item = readdir(dir); count++;
    while(item != NULL){
        //去除文件.和..
        if(strcmp(item->d_name, ".")==0 || strcmp(item->d_name, "..")==0 ){
            item = readdir(dir);
            continue;
        } //左对齐占用 10 格的形式输出文件名
        printf("%-10s ", item->d_name); count++;
        if(count%5 == 1) printf("\n"); //一行输出 5 个
        item = readdir(dir); //获取下一个文件
    }
    printf("\n");
}

```

With option like -a -i -l:

-a:

```

// with option
if(argc == 3){
    // -a 与无参数 ls 类似，但是要输出.和..文件
    if(strcmp(argv[2], "-a") == 0){
        item = readdir(dir); count++;
        while(item != NULL){
            printf("%-10s ", item->d_name);
            count++;
            if(count%5 == 1) printf("\n");
            item = readdir(dir); //获取下一个文件
        }
        printf("\n");
    }
    // -i
}

```

-i:

```
// -i
if(strcmp(argv[2], "-i") == 0){
    item = readdir(dir); count++;
    while(item != NULL){
        if(strcmp(item->d_name, ".")==0 || strcmp(item->d_name, "..")==0 ){
            item = readdir(dir);
            continue;
        }
        //输出 inode 以及文件名
        printf("%ld %-10s ", item->d_ino, item->d_name);
        count++;
        if(count%4 == 1) printf("\n");
        item = readdir(dir); //获取下一个文件
    }
    printf("\n");
}
```

-l:

```
// -l
if(strcmp(argv[2], "-l") == 0){
    item = readdir(dir);
    while(item != NULL){
        if(strcmp(item->d_name, ".")==0 || strcmp(item->d_name, "..")==0 ){
            item = readdir(dir); //读取下一个, 不进行 print
            continue;
        }
        lstat(item->d_name, &buf); //通过文件名查找文件的信息
        printf(" %-10s ", item->d_name); //先打印文件名
        print(&buf); //调用自定义的打印函数, 依次打印文件的信息
        item = readdir(dir); //获取下一个文件
    }
}
```

-Test:

```
gakiara@ubuntu:~/Lab2$ gcc -o myls myls.c
gakiara@ubuntu:~/Lab2$ ./mysls ls -a
5.sh      1.txt      1.sh      ..          3.sh
.          myls       2.txt     dir2        dir1
2.sh      myls.c     cmpNew.sh 4.sh
gakiara@ubuntu:~/Lab2$ ./mysls ls -i
3685944 5.sh      3686631 1.txt      3686629 1.sh      3686639 3.sh
3685936 myls       3686636 2.txt      3686555 dir2     3686633 dir1
3670316 2.sh      3685928 myls.c     3685957 cmpNew.sh 3686626 4.sh
gakiara@ubuntu:~/Lab2$ ./mysls ls -l
5.sh      -rw-rw-r-- 1 gakiara gakiara 2959 Thu Jan 14 00:52:40 2021
1.txt     -rw-rw-r-- 1 gakiara gakiara 0 Tue Jan 12 16:39:14 2021
1.sh      -rw-rw-r-- 1 gakiara gakiara 514 Tue Jan 12 16:43:38 2021
3.sh      -rw-rw-r-- 1 gakiara gakiara 606 Tue Jan 12 16:54:50 2021
mysls     -rwxrwxr-x 1 gakiara gakiara 13320 Thu Jan 14 01:21:11 2021
2.txt     -rw-rw-r-- 1 gakiara gakiara 0 Tue Jan 12 16:39:18 2021
dir2      drwxrwxr-x 2 gakiara gakiara 4096 Thu Jan 14 01:12:18 2021
dir1      drwxrwxr-x 2 gakiara gakiara 4096 Thu Jan 14 01:12:18 2021
2.sh      -rw-rw-r-- 1 gakiara gakiara 1059 Wed Jan 13 21:59:01 2021
mysls.c   -rw-rw-r-- 1 gakiara gakiara 4647 Thu Jan 14 00:17:20 2021
cmpNew.sh -rw-rw-r-- 1 gakiara gakiara 213 Thu Jan 14 01:18:06 2021
4.sh      -rw-rw-r-- 1 gakiara gakiara 253 Wed Jan 13 23:49:26 2021
```

三、 讨论、心得 (必填) (10 分)

//在这里写: 实验过程中遇到的问题及解决的方法, 你做本实验体会。300字以上。

1. 书写规范要注意

当大小写不一样的时候表示的意义不一样, -f 和 -F 以及中英文输入都需要注意其中空格要特别注意

等于号, 在下面第一种情况下是无法赋值的

```
filename = "$1" #if the file is an ordinary file
```

```
parallels@parallels-Parallels-Virtual-Platform:~/lab2$ bash 1.sh 1.txt
1.sh: line 10: filename: command not found
the filename is .
the owner is -rw-r--r--.
the last edit time is parallels 474 Dec.
```

正确:

```
...
filename="$1" #if the file is an ordinary file
```

漏掉了空格后, #后面的并不能正确地编译, 作为注释, 后面会进行空格的总结

```
name"#print the owner
```

2. shell 编程中, 使用[]或者[][]包裹的表达式中, 在括号的旁边需要留出空格, 像表达式[3 -eq 5]会出现无法解析的错误。[[是 bash 程序语言的关键字。并不是一个命令, [[]] 结构比[]结构更加通用。在[[和]]之间所有的字符都不会发生文件名扩展或者单词分割, 但是会发生参数扩展和命令替换。

3. 在使用 Makefile 编译的题目上, 以前只知道这么打命令就可以编译链接, 没有仔细地去看它具体的实现过程。这次作业让我对于整个过程有了更好的理解, 同时对于以前写程序时出现的动态库丢失错误有了更好的理解。

4.在“编写 shell 程序, 统计指定目录下的普通文件、子目录及可执行文件的数目, 目录的路径名字由参数传入”, 统计目录下的普通文件、子目录的方式还可以通过 find 命令来查找, 具体命令如下:

```
echo `find $1 -type f | wc -l`
```

```
echo `find $1 -type d | wc -l`
```

```
echo `find $1 -type f -executable | wc -l`
```

5.对空格的情况做一个总结:

重定向时, 0 为 stdin, 1 为 stdout, 2 为错误信息, 写重定向的命令时, 一定要注意要写成 `1> filename` 而不是 `1 > filename`, 也就是 1 与 > 之间不能够有空格否则会出错。”。

`declare` 的赋值前后不能有空格。`declare -x age=20`。

shell 编程中, 赋值语句前后不能够有空格, 如 `a=3` 不能写成 `a = 3`, 也不能写成 `$a=3`。同时如果要将命令的执行结果赋值给变量时, 需要用以下两者之一的方式: `res=$(pwd)`或 `res=`pwd``。

shell 编程中, 使用 `[]` 或 `[[]]` 包裹的表达式中, 在操作数和操作符或者括号的前后都要至少留一个空格比如 `[$# -ne 1]`或`[3 -eq 5]`, 而`[3 -eq 5]`则会报错找不到命令。`[]`与 `test` 的功能是相同的, 双方括号`[[expr]]`命令中的表达式

`expr` 可以使用标准的字符串比较, 也能够使用正则表达式。

双括号命令允许在比较过程中使用高级表达式, 形式为`((expr))`, 如`(($val1 ** 2 > 90))`, 但可以不加空格如`((b =2**2))`

`let` 语句中, 若表达式有空格, 则使用引号, 如:

```
let "a = 8" "b = 13" let c=a+b
```

`expr` 命令运算符两边都需要保留空格, 如 `expr 1 + 2` , 如果没保留空格如 `expr 1+2` 则会输出 `1+2`

④ `gcc -c test.c` 将生成 `test.o` 的目标文件, `gcc -o target test.c` 将生成可执行文件

`target`, `gcc -c a.c -o a.o` 与 `gcc -c a.c` 等价, 而 `gcc -o out a.o` 将直接生成可执行文件

`out`。 `gcc -c` 可以跟多个文件, 如 `gcc -c main.c input.c compute.c`

`gcc -o` 有两种写法:

```
---gcc main.o input.o compute.o -o power -lm
```

```
---gcc -o power main.o input.o compute.o -lm
```

`#include<pthread.h>`时需要加`-lpthread`

通过本次实验，我初步掌握了 shell 程序设计的一般方法，对 shell 的程序设计思想有所体会。shell 虽然是一门脚本语言，但其接近操作系统底层，有很多与其他脚本语言不同之处。shell 由于有管道和重定向功能，其对文本的处理能力非常强大。但 shell 对数值的处理能力较弱，只支持整数的计算，且编写复杂，效率较低，在编程的过程中应减少使用。