EzShell

黄铭昊 191830058

目录

1

需求分析

需求分析

实现类Shell，支持以下功能：

运行本地Shell（.sh文件）

命令	参数
cp	-r –i --help
cmp	-b –l --help
wc	-w –c –l –m –L --help
cat	-n –b –s –E --help
man	--help
pwd	--help
cd	
ls	-a –A –l –t -r --help
mkdir	--help
touch	--help
rmdir	-p --help
rm	-i –r --help
echo	
exit	

2

结构设计

结构设计

分析问题：

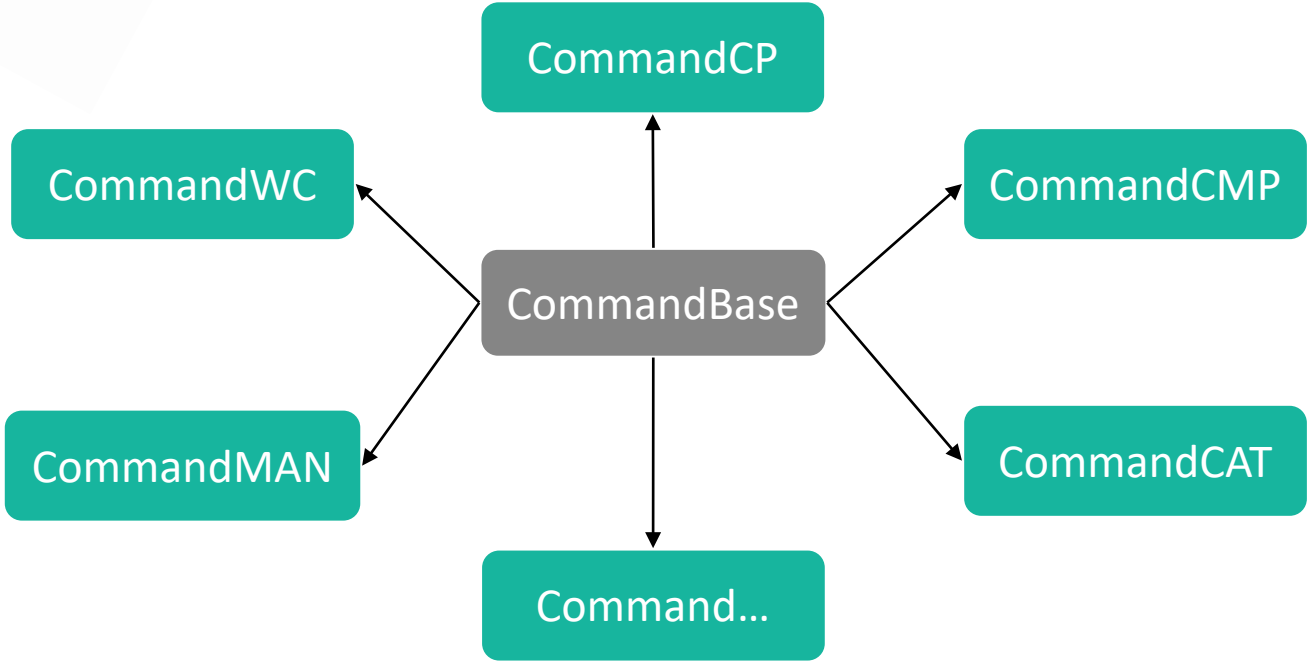
所有的具体命令都有共通的步骤：
（1）运行
（2）解析命令（分离参数，分离“文件”）
（3）解析参数
（4）获取当前的路径

能否用共用的操作来减少代码的重复？

运用面向对象，设计一个CommandBase基类！

结构设计

以CommandBase为基类，其他命令继承CommandBase



结构设计

CommandBase类提供虚函数void run()，实现命令的运行
命令子类重载void run()，实现多态

```
class CommandBase {
public:
    CommandBase(string name, string str, DirHelper *dirHelper);
    ~CommandBase();
    virtual void run();
protected:
    string strSrc, name, help;
    DirHelper *dirHelper;
    vector<string> command;
    vector<string> files;
    vector<string> opt;
    unordered_map<string, bool*> mapOpt;
    void splitCommand();
    int analyzeOpt();
};
```

dirHelper用来存储当前的路径

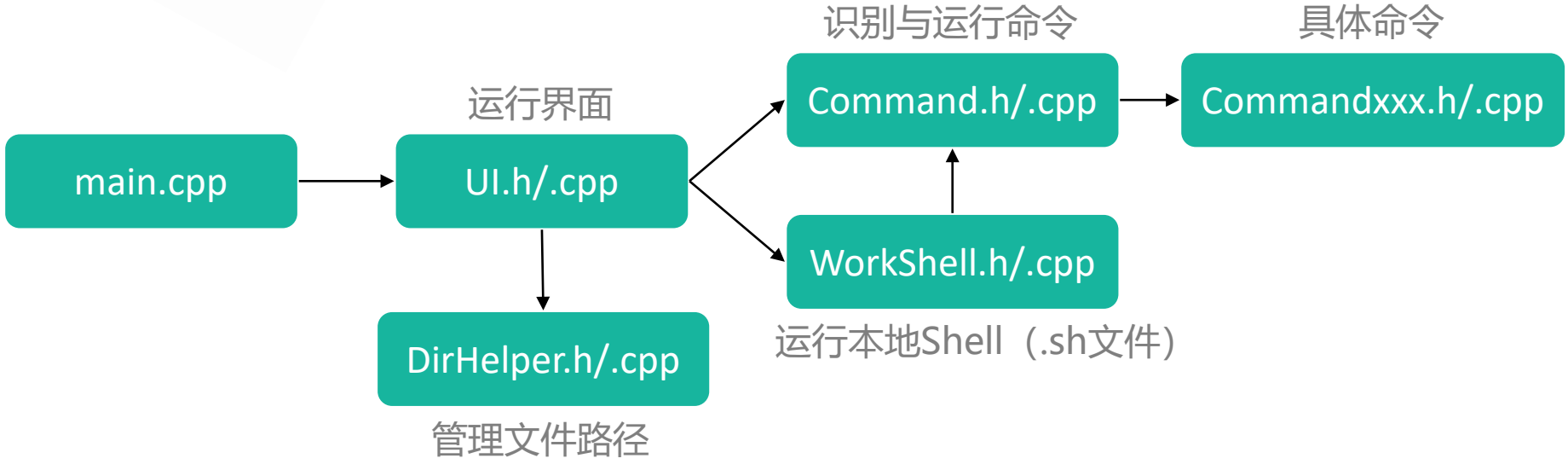
分析参数
用unordered_map
存储参数和对应字符
串的关系

分割输入的命令
command：按照空格、tab初步划分
opt：存储参数
files：存储除参数外的内容

3

模块划分

模块划分


识别与运行命令
具体命令
运行界面
Command.h/.cpp
Commandxxx.h/.cpp
main.cpp
UI.h/.cpp
WorkShell.h/.cpp
DirHelper.h/.cpp
运行本地Shell（.sh文件）
管理文件路径

UI.h/.cpp

```
void UI::show() {
    DirHelper *dirHelper = new DirHelper();
    dirHelper->initPath();
    command = new Command(dirHelper);
    string str;
    cout << "Welcome!" << endl;
    cout << "$ ";
    while (getline(cin, str)) {
        int k = command->find(str);
        if (!k) {
            cout << "Unknown command." << endl;
        } else if (k == -1) {
            break;
        } else if (k == -3) {
            WorkShell *shell = new WorkShell(str, dirHelper);
            shell->run();
        } else if (k != -2) {
            command->run();
        }
        cout << "$ ";
    }
}
```

循环输入命令
调用Command的void find()查找命令
调用Command的void run()运行命令
调用WorkShell的void run()运行本地Shell

Command.h/.cpp

识别命令
给CommandBase command赋予相应的具体命令

```
int Command::find(string str) {
    string s = string();
    for (int i = 0; i < str.size(); ++i) {
        if ((str[i] == ' ' || str[i] == '\t') && s.empty()) continue;
        if (str[i] != ' ' && str[i] != '\t') s.push_back(str[i]);
        else break;
    }
    command = NULL;
    if (s == "cp") command = new CommandCP(str, dirHelper);
    if (s == "cmp") command = new CommandCMP(str, dirHelper);
    if (s == "wc") command = new CommandWC(str, dirHelper);
    if (s == "cat") command = new CommandCAT(str, dirHelper);
    if (s == "man") command = new CommandMAN(str, dirHelper);
    if (s == "echo") command = new CommandECHO(str, dirHelper);
    if (s == "ls") command = new CommandLS(str, dirHelper);
    if (s == "pwd") command = new CommandPWD(str, dirHelper);
    if (s == "cd") command = new CommandCD(str, dirHelper);
    if (s == "mkdir") command = new CommandMKDIR(str, dirHelper);
    if (s == "exit") return -1;
    if (s.empty()) return -2;
    if (s.size() >= 2 && s[0] == '.' && s[1] == '/') return -3;
    return command != NULL;
}
```

运行命令
直接调用CommandBase的void run()
方便，简洁!

```
void Command::run() {
    command->run();
}
```

WorkShell.h/.cpp

WorkShell本质上还是和UI一样进行多个命令的处理，只不过增加了一步文件读入

```
void WorkShell::run() {
    int p = 0;
    while (str[p] == ' ' || str[p] == '\t') p++;
    p += 2;
    if (p >= str.size()) {
        cout << "Please input file name!" << endl;
        return;
    }
    fileName.clear();
    for (int i = p, flag = 0; i < str.size(); ++i) {
        if (str[i] == ' ' && !flag) {
            break;
        }
        flag = 0;
        if (str[i] == '\\') {
            flag = 1;
        } else {
            fileName.push_back(str[i]);
        }
    }
    ifstream in(dirHelper->getPath()+"/"+fileName);
    if (in) {
        string s;
        Command *command = new Command(dirHelper);
        while (getline(in, s)) {
            int k = command->find(s);
            if (!k) {
                cout << "Unknown command." << endl;
            } else if (k == -1) {
                break;
            } else if (k == -3) {
                WorkShell *shell = new WorkShell(s, dirHelper);
                shell->run();
            } else if (k != -2) {
                command->run();
            }
        }
        in.close();
    } else cout << fileName << ": No such file or directory" << endl;
}
```

识别文件名（处理空格“\ ”）

读入文件

执行文件中的命令（与UI.h/.cpp相似）

DirHelper.h/.cpp

```
class DirHelper {
public:
    DirHelper();
    ~DirHelper();
    string getPath();
    void initPath();
    void setPath(string str);
    string back(string str);
    string backToUser();
private:
    string path;
};
```

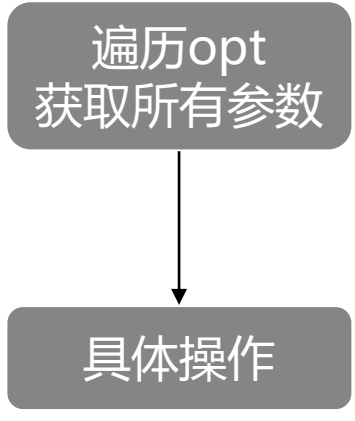
void getPath()：获取路径
void initPath()：获取一开始的路径
void setPath(string str)：设置路径
string back(string str)：获取str的上一级路径
string backToUser()：获取user文件路径

4

功能实现

功能实现

通用的框架：



功能实现

cp：复制文件：用二进制读取srcFile，用二进制写入destFile
复制文件夹：递归复制，调用复制文件函数

cmp：用二进制读取file1，file2读入到char *buffer1，buffer2
比较buffer1和buffer2

wc：用二进制读取file读入到char *buffer，统计各种数目

cat：每行读取file，边读边输出

man：根据要查看的命令读取help/命令文件，边读边输出

功能实现

如何遍历文件夹以及其下的子文件/夹?

用递归实现!

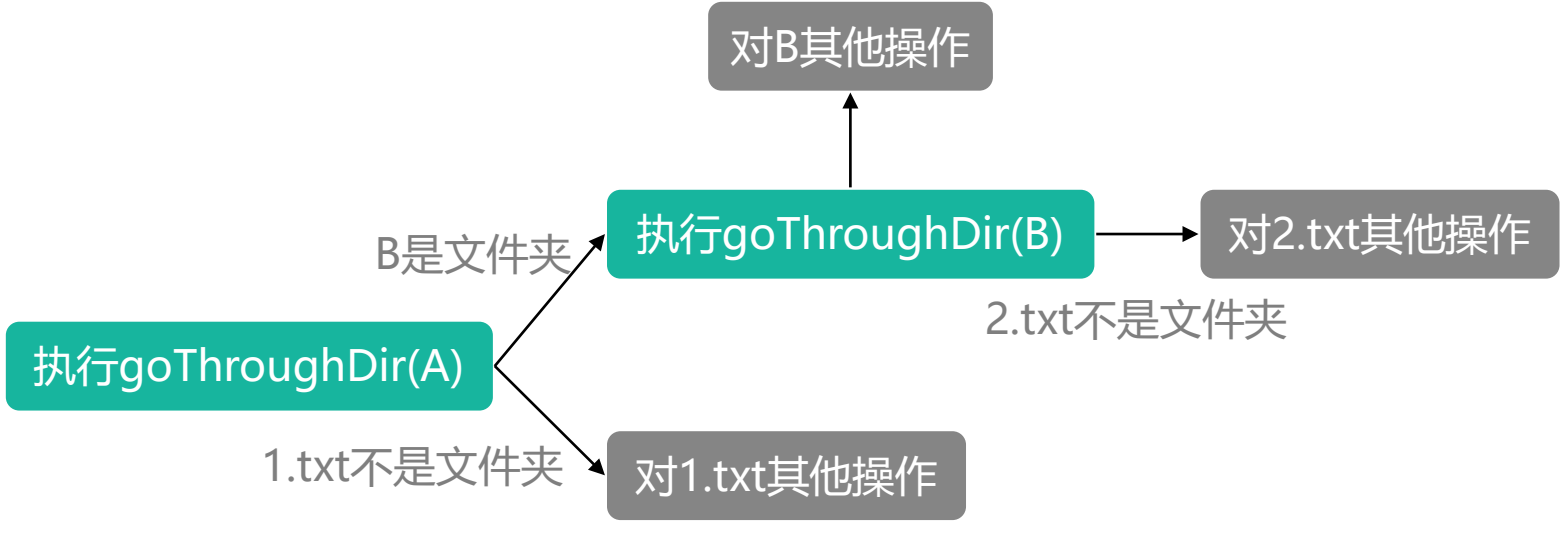
```
void goThroughDir(string path) {
    DIR *dir = opendir(path.c_str()); //打开文件夹
    if (dir != NULL) {
        dirent *file;
        while (file = readdir(dir)) { //遍历文件夹下所有文件
            if (file->d_type == DT_DIR && strcmp(file->d_name, ".") && strcmp(file->d_name, "..") {
                //如果是file是文件夹并且不是当前目录也不是上级目录
                goThroughDir(path+”/”+file->d_name);
                //TODO：其他操作
            } else {
                //如果file不是文件夹
                //TODO：其他操作
            }
        }
        closedir(dir);
    }
}
```

功能实现

如何遍历文件夹以及其下的子文件/夹?

举个例子.....

有一文件夹：A –B　　-2.txt
　　|-1.txt



功能实现

pwd：直接输出dirHelper->getPath()

cd：解析路径到vector<string> path里，逐个尝试进入path[i]

ls：遍历目录下文件

mkdir：运用mkdir()函数

echo：输出command[i]

功能实现

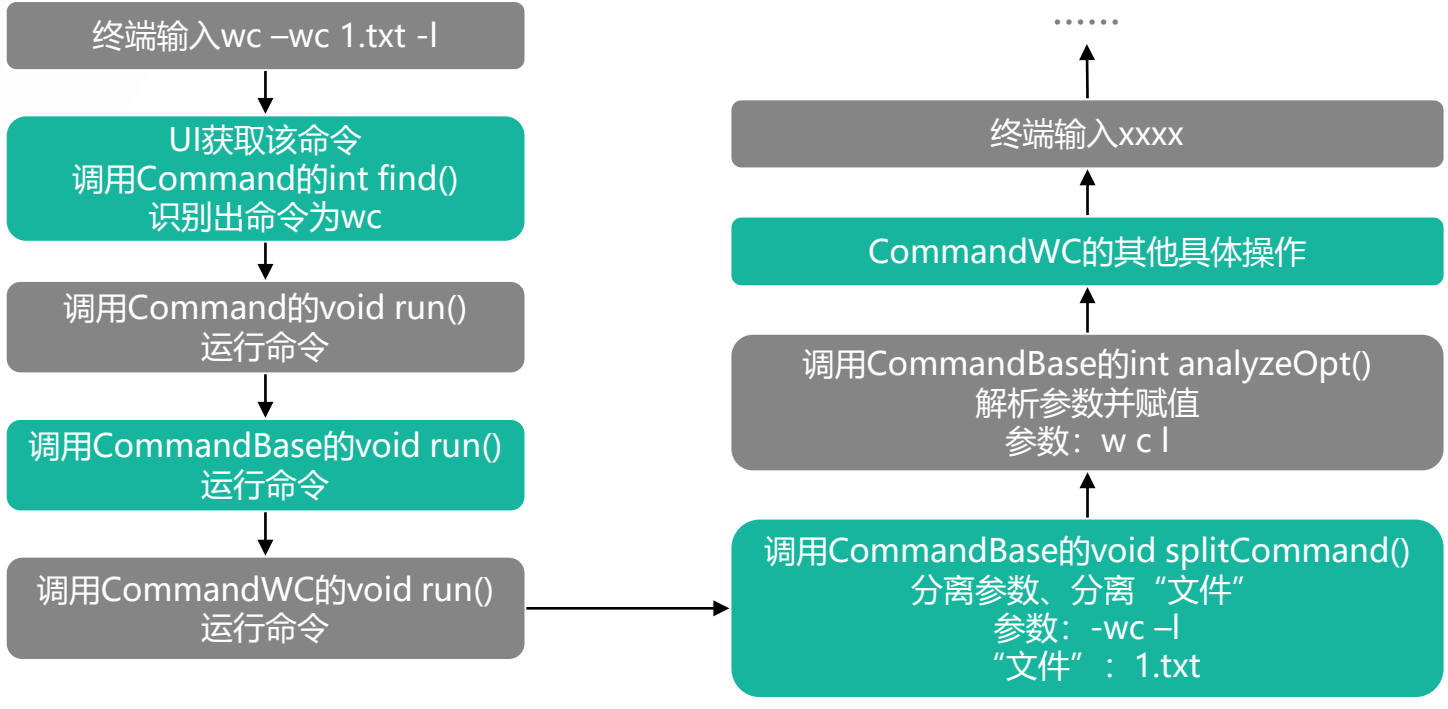
touch：若文件不存在，则新建一个
若文件存在，用二进制读一遍再写一遍以修改其修改时间

rmdir：调用rmdir()函数

rm：删除文件：调用remove()函数
删除文件夹：递归删除

功能实现

举个例子……

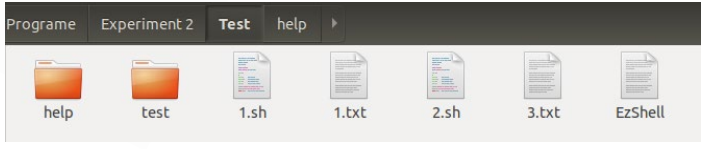

终端输入wc –wc 1.txt -l
UI获取该命令
调用Command的int find()
识别出命令为wc
调用Command的void run()
运行命令
调用CommandBase的void run()
运行命令
调用CommandWC的void run()
运行命令
调用CommandBase的void splitCommand()
分离参数、分离“文件”
参数：-wc –l
“文件”：1.txt
调用CommandBase的int analyzeOpt()
解析参数并赋值
参数：w c l
CommandWC的其他具体操作
终端输入xxxx
……

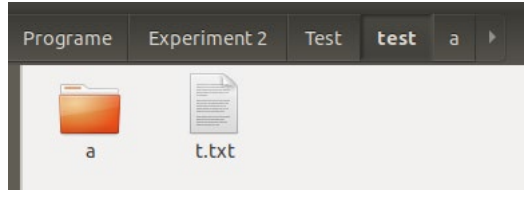
5

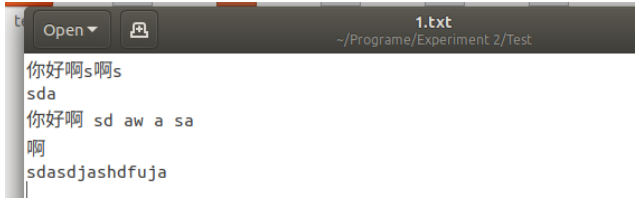
功能展示

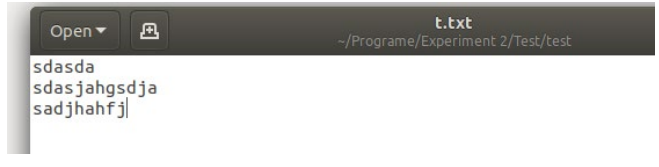
功能展示

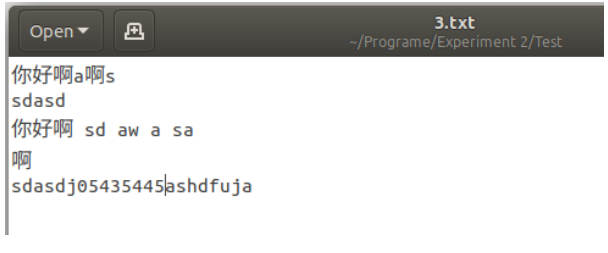
分别用EzShell和Shell运行1.sh

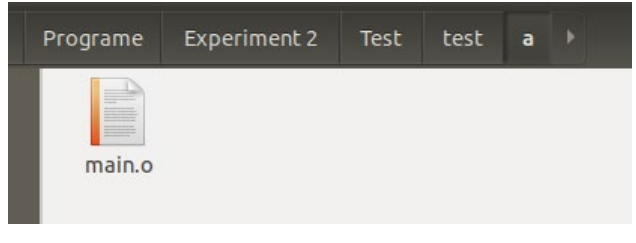












功能展示

Open ▾ 1.sh ~/Programe/Experiment 2/Test

```
echo Hello!
cp 1.txt 2.txt
cmp 1.txt 2.txt
cmp 1.txt 3.txt
mkdir test1
cd test1
pwd
cd ..
pwd
./2.sh
cat 1.txt
cp 1.txt 2.txt -i
wc -wc -m -Ll 1.txt
cat 3.txt -b -E 1.txt
cat test1
cp 1.txt test1
cp -r test test2
ls test2
cmp -b -l test1/../test1/1.txt 3.txt
ls -lAtr
wc --help
touch 4.txt
cd test2
rm t.txt
mkdir b
ls
cd ..
rm -r test2
echo done!
```

Open ▾ 2.sh ~/Programe/Experiment 2/Test

```
echo this is 2.sh
```

功能展示

EzShell运行结果：

```
sea@sea-virtual-machine:~/Programe/Experiment 2/Test$ ./EzShell
Welcome!
$ ./1.sh
Hello!
1.txt 3.txt differ: byte 10, line 1
/home/sea/Programe/Experiment 2/Test/test1
/home/sea/Programe/Experiment 2/Test
this is 2.sh
你好啊s啊s
sdasd
你好啊 sd aw a sa
啊
sdasdjashdfuja

cp: overwrite '/2.txt'? y
6 9 46 62 17 1.txt
1 你好啊a啊s$
2 sdasd$
3 你好啊 sd aw a sa$
4 啊$
5 sdasdj05435445ashdfuja$
1 你好啊s啊s$
2 sdasd$
3 你好啊 sd aw a sa$
4 啊$
5 sdasdjashdfuja$
$
cat: test1: Is a directory
a t.txt
10 163 s 141 a
53 141 a 60 0
54 163 s 65 5
55 150 h 64 4
56 144 d 63 3
57 146 f 65 5
58 165 u 64 4
59 152 j 64 4
60 141 a 65 5
61 12 ^J 141 a
62 12 ^J 163 s
cmp: EOF on test1/../test1/1.txt after byte 62
```

```
cmp: EOF on test1/../test1/1.txt after byte 62
drwxr-xr-x 2 sea sea 4096 Apr 3 6:29 help
drwxr-xr-x 3 sea sea 4096 Apr 5 5:17 test
-rwxr--r-- 1 sea sea 18 Apr 5 5:19 2.sh
-rw-r--r-- 1 sea sea 62 Apr 5 5:24 1.txt
-rw-r--r-- 1 sea sea 69 Apr 5 5:24 3.txt
-rwxr--r-- 1 sea sea 350 Apr 7 4:2 1.sh
-rwxrwxr-x 1 sea sea 263248 Apr 7 4:3 EzShell
-rw-r--r-- 1 sea sea 62 Apr 7 4:6 2.txt
drwxr-xr-x 3 sea sea 4096 Apr 7 4:6 test2
drwxr-xr-x 2 sea sea 4096 Apr 7 4:6 test1

Usage: wc [OPTION]... [FILE]...
Print newline, word, and byte counts for each FILE, and a total line if
more than one FILE is specified.  A word is a non-zero-length sequence of
characters delimited by white space.

The options below may be used to select which counts are printed, always in
the following order: newline, word, character, byte, maximum line length.
  -c, --bytes            print the byte counts
  -m, --chars            print the character counts
  -l, --lines            print the newline counts
  -L, --max-line-length  print the maximum display width
  -w, --words            print the word counts
      --help     display this help and exit
a b
done!
$
```

```
$ exit
sea@sea-virtual-machine:~/Programe/Experiment 2/Test$ ls -R
.:
1.sh  1.txt  2.sh  2.txt  3.txt  4.txt  EzShell  help  test  test1

./help:
cat  cmp  cp  echo  ls  man  mkdir  pwd  wc

./test:
a  t.txt

./test/a:
main.o

./test1:
1.txt
```

功能展示

Shell运行结果：

```
sea@sea-virtual-machine:~/Programe/Experiment 2/Test$ sh
$ ./1.sh
Hello!
1.txt 3.txt differ: byte 10, line 1
/home/sea/Programe/Experiment 2/Test/test1
/home/sea/Programe/Experiment 2/Test
this is 2.sh
你好啊s啊s
sdasd
你好啊 sd aw a sa
啊
sdasdjashdfuja

cp: overwrite '2.txt'? y
 6  9 46 62 17 1.txt
     1  你好啊a啊s$
     2  sdasd$
     3  你好啊 sd aw a sa$
     4  啊$
     5  sdasdj05435445ashdfuja$
     6  你好啊s啊s$
     7  sdasd$
     8  你好啊 sd aw a sa$
     9  啊$
    10  sdasdjashdfuja$
$
cat: test1: Is a directory
a  t.txt
10 163 s    141 a
53 141 a     60 0
54 163 s     65 5
55 150 h     64 4
56 144 d     63 3
57 146 f     65 5
58 165 u     64 4
59 152 j     64 4
60 141 a     65 5
61  12 ^J   141 a
62  12 ^J   163 s
cmp: EOF on test1/../test1/1.txt after byte 62
```

```
total 296
drwxr-xr-x 2 sea sea   4096 Apr  3 06:29 help
drwxr-xr-x 3 sea sea   4096 Apr  5 05:17 test
-rwxr--r-- 1 sea sea     18 Apr  5 05:19 2.sh
-rw-r--r-- 1 sea sea     62 Apr  5 05:24 1.txt
-rw-r--r-- 1 sea sea     69 Apr  5 05:24 3.txt
-rwxr--r-- 1 sea sea    350 Apr  7 04:02 1.sh
-rwxrwxr-x 1 sea sea 263248 Apr  7 04:03 EzShell
drwxr-xr-x 3 sea sea   4096 Apr  7 04:08 test2
drwxr-xr-x 2 sea sea   4096 Apr  7 04:08 test1
-rw-r--r-- 1 sea sea     62 Apr  7 04:08 2.txt
Usage: wc [OPTION]... [FILE]...
  or:  wc [OPTION]... --files0-from=F
Print newline, word, and byte counts for each FILE, and a total line if
more than one FILE is specified.  A word is a non-zero-length sequence of
characters delimited by white space.

With no FILE, or when FILE is -, read standard input.

The options below may be used to select which counts are printed, always in
the following order: newline, word, character, byte, maximum line length.
  -c, --bytes            print the byte counts
  -m, --chars            print the character counts
  -l, --lines            print the newline counts
      --files0-from=F    read input from the files specified by
                           NUL-terminated names in file F;
                           If F is - then read names from standard input
  -L, --max-line-length  print the maximum display width
  -w, --words            print the word counts
      --help     display this help and exit
      --version  output version information and exit

GNU coreutils online help: <http://www.gnu.org/software/coreutils/>
Full documentation at: <http://www.gnu.org/software/coreutils/wc>
or available locally via: info '(coreutils) wc invocation'
a  b
done!
$
```

```
$ exit
sea@sea-virtual-machine:~/Programe/Experiment 2/Test$ ls -R
.:
1.sh  1.txt  2.sh  2.txt  3.txt  4.txt  EzShell  help  test  test1

./help:
cat  cmp  cp  echo  ls  man  mkdir  pwd  wc

./test:
a  t.txt

./test/a:
main.o

./test1:
1.txt
```

谢谢