# 《Linux 操作系统设计实践》

**实验报告**

## 实验 4：文件操作

**院 系： 数学与计算机科学学院**

**专 业： 计算机科学与技术**

**年 级： 2016 级计算机 5 班**

**学 号： 031602507**

**姓 名： 陈俞辛**

### 一、实验环境：Ubuntu Kylin 14.04

### 二、实验内容：

**（一）实验代码**

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

int main()

{

FILE \*input;

FILE \*output;

char ch;

char str[100];

int strlen = 0;

char word[100];

int wordlen = 0;

input = fopen("input.txt","wt+"); //以读写方式打开或者建立一个文本文件

if(input == NULL) //打开文件失败

{

printf("Cannot open file strike any key exit!");

exit(1);

}

else // 打开文件成功

{

printf("input a string:\n"); //输入一个字符串，换行结束

ch = getchar();

while(ch != '\n')

{

fputc(ch,input);

ch = getchar();

}

rewind(input); //rewind() 用于把 input 所指文件的内部位置指针移到文件头

while(ch != EOF) //逐个字符读取文件，存入字符数组

{

ch = fgetc(input);

str[strlen++] = ch;

}

fclose(input); //关闭文件

output = fopen("output.txt","wt+"); //以读写方式打开或者建立一个文本文件

for(int i = strlen-1;i >= 0;i--) //将先前保存的字符数组处理之后写入文件

{

if((str[i] >= 'a' && str[i] <= 'z') || (str[i] >= 'A' && str[i] <= 'Z'))

{

word[wordlen++] = str[i];

}

else if(str[i] == ' ')

{

for(int j = wordlen-1;j >= 0;j--)

{

fputc(word[j],output);

}

wordlen = 0;

fputc('/',output);

}

else

{

fputc(str[i],output);

}

}

for(int j = wordlen-1;j >= 0;j--)

{

fputc(word[j],output);

}

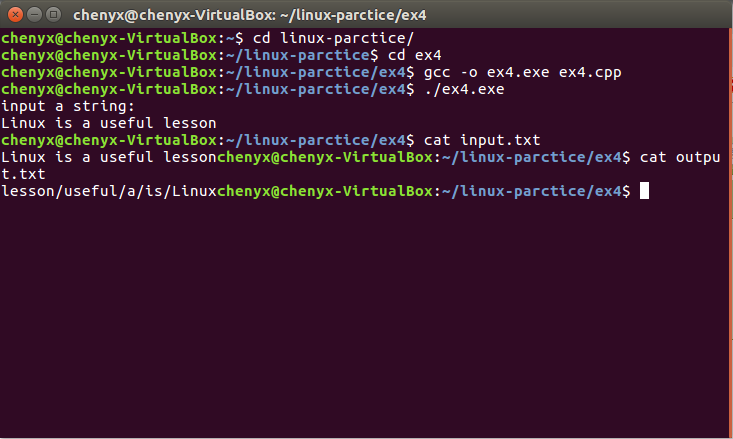
fclose(output); //关闭文件

}

return 0;

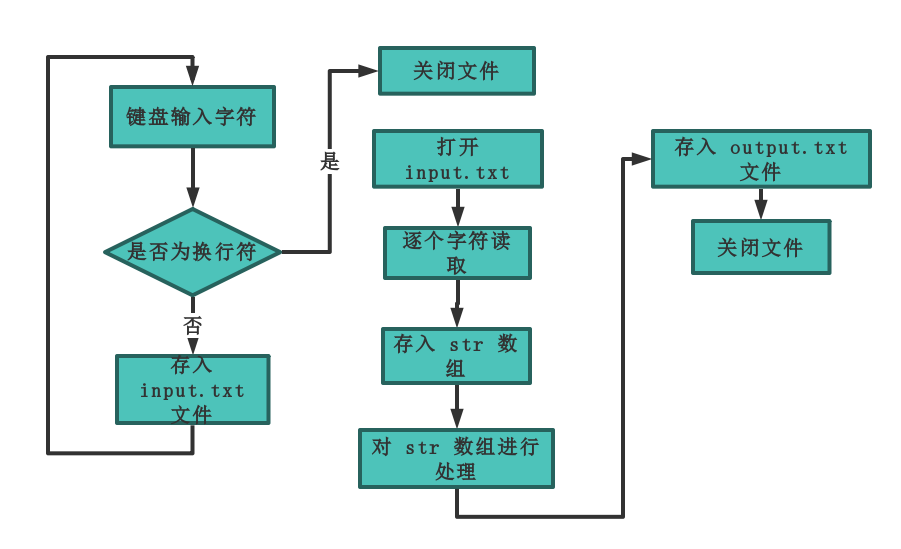
}

**（二）实验结果**



**三、 实验总结**

**（一）编程思想**

****

程序的流程如上图所示。运行程序，用户可以输入一串字符，以换行结束输入。用户每输入一个字符就存入 input.txt 文件，直至换行符。然后打开 input.txt 文件将内容取出存入 str 数组。对 str 数组进行处理，将文本中的单词倒序输出至 output.txt 文件，并且用 / 替代原来文本中的空格。

**（二）相关知识**

**1、fopen()**

创建一个新的文件或者打开一个已有的文件，这个调用会初始化类型 FILE 的一个对象，类型 FILE 包含了所有用来控制流的必要的信息

函数原型如下：

FILE \*fopen( const char \* filename, const char \* mode );

filename 是字符串，用来命名文件，访问模式 mode 的值可以是下列值中的一个：

|  |  |
| --- | --- |
| **模式** | **描述** |
| r+ | 打开一个文本文件，允许读写文件。 |
| w+ | 打开一个文本文件，允许读写文件。如果文件已存在，则文件会被截断为零长度，如果文件不存在，则会创建一个新文件。 |
| a+ | 打开一个文本文件，允许读写文件。如果文件不存在，则会创建一个新文件。读取会从文件的开头开始，写入则只能是追加模式。 |

如果处理的是二进制文件，则需使用下面的访问模式来取代上面的访问模式：

**"rb", "wb", "ab", "rb+", "r+b", "wb+", "w+b", "ab+", "a+b"**

**2、fputc()**

把参数 c 的字符值写入到 fp 所指向的输出流中。如果写入成功，它会返回写入的字符，如果发生错误，则会返回 EOF

函数原型如下：

int fputc( int c, FILE \*fp );

**3、rewind()**

用于将文件指针重新指向文件的开头，同时清除和文件流相关的错误和EOF标记，相当于调用fseek(stream, 0, SEEK\_SET)

函数原型如下：

void rewind( FILE \* stream );

**4、fgetc()**

从 fp 所指向的输入文件中读取一个字符。返回值是读取的字符，如果发生错误则返回 EOF。

函数原型如下：

int fgetc( FILE \* fp );

**5、fclose()**

用于关闭文件，如果成功关闭文件，fclose( ) 函数返回零，如果关闭文件时发生错误，函数返回 EOF。这个函数实际上，会清空缓冲区中的数据，关闭文件，并释放用于该文件的所有内存。

函数原型如下：

int fclose( FILE \*fp )