操作系统实验报告

姓名：任杰文

学号：09013430

东南大学计算机科学与工程学院

School of Computer Science & Engineering

Southeast University

二0一六年三月十三日

实验一

一、 实验内容：

使用系统调用，用 C 或 C++写一个程序，实现如下功能：从一个文件中读出数据，写入另一个文件中。

要求：

1. 具有良好的交互性

   使用者可输入源文件和目的文件的路径和文件名。

2. 具有完善的错误处理机制

   针对可能出现的各种错误，要有相应的错误提示输出，并作相应处理。

3. 在 Windows 和 Linux 操作系统上调试并运行

二、 实验目的：

1. 通过实验，加深对系统调用概念的理解，了解其实现机制以及使用方式。
2. 通过在 Linux 操作系统上编写和调试简单程序，进一步熟悉 Linux 操作系统的使用，初步掌握 linux 环境下的 C 或 C++编译和调试工具，为进一步理解和学习 Linux 操作系统的内核结构和核心机制作准备。

三、 设计思路及流程图

1. 设计思路：

（1） 提示并输入源文件和目标文件名；

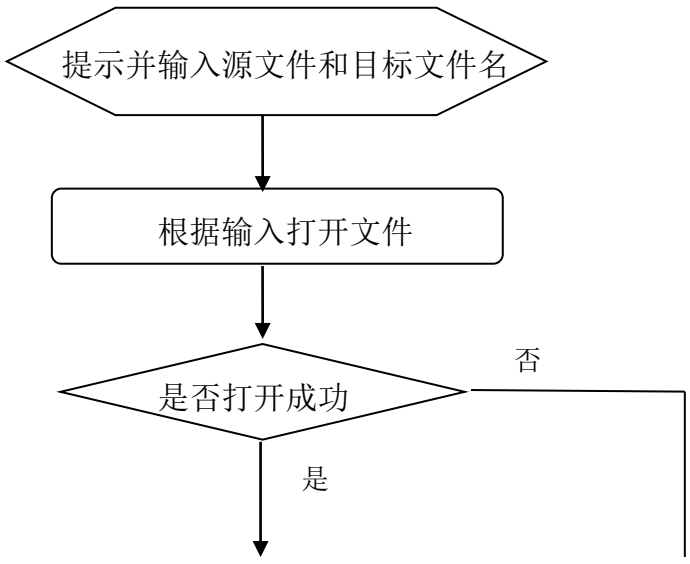
（2） 根据输入打开文件，并判断是否打开成功；

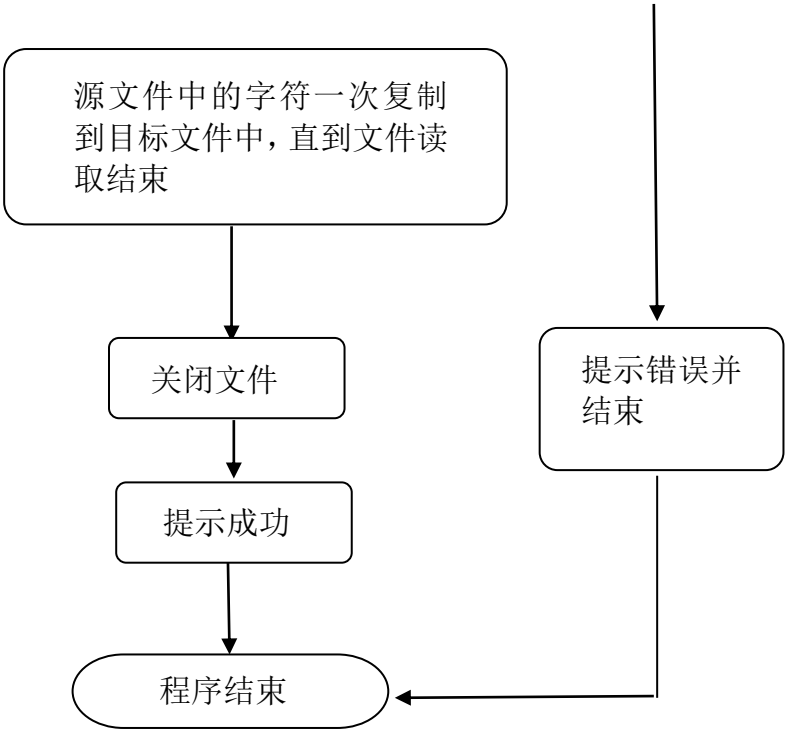
（3） 若文件打开成功，将源文件中的字符一次复制到目标文件中，直到文件读取结束；若打开失败，提示错误，结束程序；

（4） 传输结束后关闭文件；

（5） 输出成功提示；

2. 流程图：





四、 源程序

1. Windows 源程序：

```
/*
 * copyFile.cpp
 *
 *  Created on: 2016年3月20日
 *      Author: rjw
 */

#include "windows.h"
#include <iostream>
#include <string>
using namespace std;
int main()
{
  //创建句柄
  HANDLE hFileRead;
  HANDLE hFileWrite;
  //数据长度
  DWORD dwDataLen = 100;
  DWORD written = 0;
  //数据缓存
  char buffer[101];

  //提示并输入源文件和目标文件名
```

```
string inputFile;
string outputFile;
cout<<"Please input the file name of source file:" <<endl;
cin>>inputFile;
cout<<"Please input the file name of destination file:" <<endl;
cin>>outputFile;

//打开文件
if ((hFileRead=CreateFile(inputFile.c_str(),
      GENERIC_READ,
      FILE_SHARE_READ,
      NULL,
      OPEN_EXISTING,
      FILE_ATTRIBUTE_NORMAL,
      NULL)) == INVALID_HANDLE_VALUE )
{
      cout<<"The temp to open file makes error! "<<endl;
      return 0;
}
if (( hFileWrite=CreateFile(outputFile.c_str(),
      GENERIC_WRITE,
      FILE_SHARE_WRITE,
      NULL,
      OPEN_ALWAYS,
      FILE_ATTRIBUTE_NORMAL,
      NULL) )== INVALID_HANDLE_VALUE)
{
      cout<<"The temp to open file makes error! "<<endl;
      return 0;
}

while( dwDataLen != 0 )
{
      //读取数据
      ReadFile(hFileRead,buffer,99,&dwDataLen,NULL);
      //标记结束位置
      buffer[dwDataLen]='\0';
      SetFilePointer(hFileWrite, 0, NULL, FILE_END);
      WriteFile(hFileWrite,buffer,dwDataLen,&written,NULL);//写入文件
}
//关闭句柄
CloseHandle(hFileRead);
CloseHandle(hFileWrite);
//输出成功提示
```

```
    cout<<"The operation of copying has terminated normally!"<<endl;
    return 0;
}
```

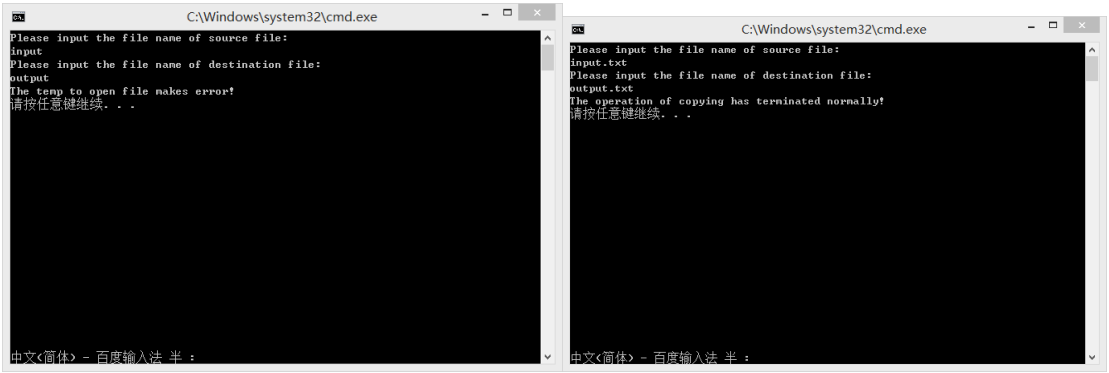
2. linux 源程序：

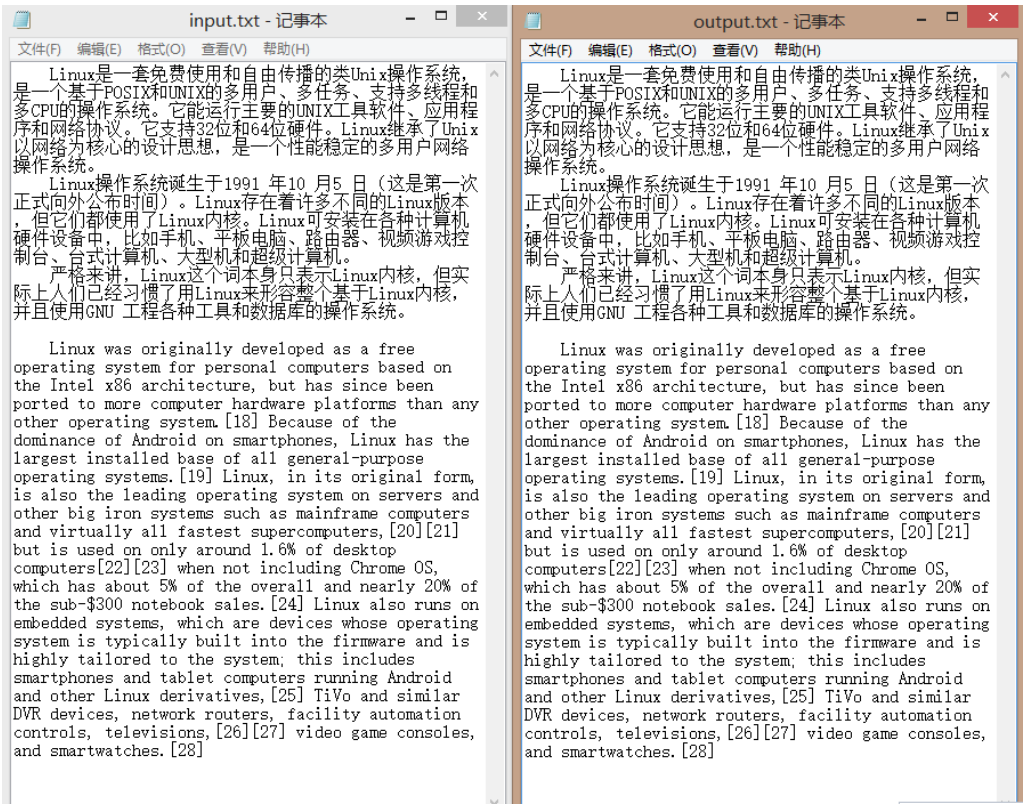
```
/*
 * copyFile.cpp
 *
 *  Created on: 2016年3月20日
 *      Author: rjw
 */
#include <string.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <unistd.h>
#include <fcntl.h>
using namespace std;
#define BUFFER_SIZE 1024
#define FILENAME_SIZE 10
int main() {
    int file_len = 0;
    int from_fd = -1;
    int to_fd = -1;
    int rest = 1;
    char buffer[BUFFER_SIZE];
    char inputFile[FILENAME_SIZE], outputFile[FILENAME_SIZE];
    char *ptr;
    //输入提示
    cout << "请输入源文件名：" << endl;
    cin >> inputFile;
    cout << "请输入目标文件名：" << endl;
    cin >> outputFile;
    //打开文件
    if ((from_fd = open(inputFile, O_RDONLY | O_CREAT)) == -1) {
        cout << "inputFile open error!" << endl;
        exit(1);
    }
    if ((to_fd = open(outputFile, O_WRONLY | O_CREAT)) == -1)
    {
        cout << "outputFile open error!" << endl;
        exit(1);
    }
    //检查文件长度
    file_len = lseek(from_fd, 0L, SEEK_END);
```

```
    lseek(from_fd, 0L, SEEK_SET);
    cout << "文件长度为：" << file_len<<endl;
    //文件复制
    while (rest) {
        rest = read(from_fd, buffer, BUFFER_SIZE);
        if (rest == -1) {
            cout<<"Read error!"<<endl;
            exit(1);
        }
        write(to_fd, buffer, rest);
        file_len -= rest;
        bzero(buffer, BUFFER_SIZE);
    }
    cout<<"there are "<<file_len<<" byte(s) data left without copy\n";
    //关闭文件
    close(from_fd);
    close(to_fd);
    return 0;
}
```

五、 实验截图

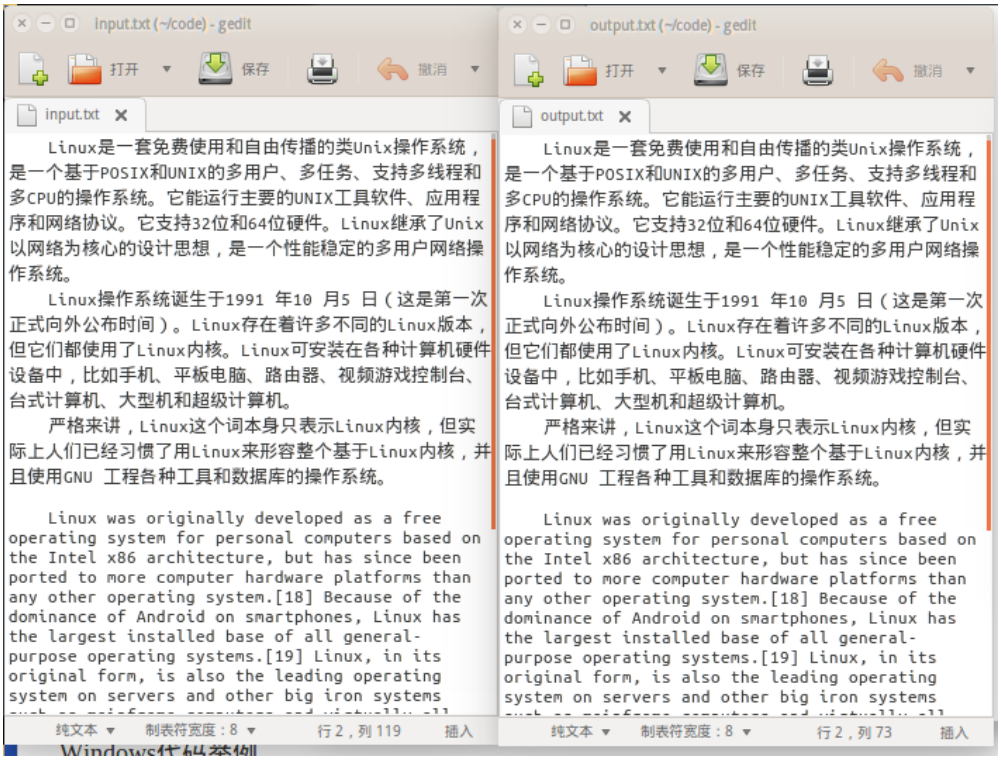
Windows:（左：文件打开失败提示；右:文件打开成功提示）





Linux：

```
请输入源文件名：
input.txt
请输入目标文件名：
output.txt
文件长度为：1758
there are 0 byte(s) data left without copy
```



六、 实验体会

1. 实验中用到的系统调用机器原型

（1） Linux 程序中用到的系统调用及调用的功能

Linux system call	功能
read	由已打开的文件读取数据
open	打开文件
create	创建新文件
lseek	用于文件位置定位
close	关闭文件
exit	中止进程
write	将数据写入已打开的文件内

函数原型：

1） open

```
int open(const char *pathname, int flags);
int open(const char *pathname, int flags, mode_t mode);
```

open 函数有两个形式，其中 pathname 是我们要打开的文件名(包含路径名称，缺省是认为在当前路径下面)，flags 可以去下面的一个值或者是几个值的组合：

标志	含义
O_RDONLY	以只读的方式打开文件
O_WRONLY	以只写的方式打开文件

O_RDWR	以读写的方式打开文件
O_APPEND	以追加的方式打开文件
O_CREAT	创建一个文件
O_EXEC	如果使用了 O_CREAT 而且文件已经存在，就会发生一个错误
O_NOBLOCK	以非阻塞的方式打开一个文件
O_TRUNC	如果文件已经存在，则删除文件的内容

O_RDONLY、O_WRONLY、O_RDWR 三个标志只能使用任意的一个。

mode 可以是以下情况的组合：

标志	含义
S_IRUSR	用户可以读
S_IWUSR	用户可以写
S_IXUSR	用户可以执行
S_IRWXU	用户可以读、写、执行
S_IRGRP	组可以读
S_IWGRP	组可以写
S_IXGRP	组可以执行
S_IRWXG	组可以读写执行
S_IROTH	其他人可以读
S_IWOTH	其他人可以写
S_IXOTH	其他人可以执行
S_IRWXO	其他人可以读、写、执行
S_ISUID	设置用户执行 ID
S_ISGID	设置组的执行 ID

2) read、write

```
int read(int fd, const void *buf, size_t length);
int write(int fd, const void *buf, size_t length);
```

其中参数 buf 为指向缓冲区的指针，length 为缓冲区的大小（以字节为单位）。函数 read()实现从文件描述符 fd 所指定的文件中读取 length 个字节到 buf 所指向的缓冲区中，返回值为实际读取的字节数。函数 write 实现将把 length 个字节从 buf 指向的缓冲区中写到文件描述符 fd 所指向的文件中，返回值为实际写入的字节数。

以 O_CREAT 为标志的 open 实际上实现了文件创建的功能，因此，下面的函数等同 creat()函数：

```
int open(pathname, O_CREAT | O_WRONLY | O_TRUNC, mode);
```

3) lseek

```
int lseek(int fd, offset_t offset, int whence);
```

lseek()将文件读写指针相对 whence 移动 offset 个字节。操作成功时，返回文件指针相对于文件头的位置。参数 whence 可使用下述值：

SEEK_SET：相对文件开头

SEEK_CUR：相对文件读写指针的当前位置

SEEK_END：相对文件末尾

4) close

```
int close(int fd);
```

fd 是我们要关闭的文件描述符

（2） Windows 中与这些系统调用相对应的 Windows32 API 及函数原型。

Windows32 API	功能
ReadFile	读文件
WriteFile	写文件
CreateFile	打开\创建文件
CloseHandle	关闭文件
SetFilePointer	移动文件指针

函数原型：

1) CreateFile

```
HANDLE CreateFile(LPCTSTR lpFileName,
DWORD dwDesiredAccess,
DWORD dwShareMode,
LPSECURITY_ATTRIBUTES lpSecurityAttributes
DWORD dwCreationDisposition
DWORD dwFlagsAndAttributes
HANDLE hTemplateFile);
```

参数：lpFileName:是以空值结尾的字符串的指针，包含要创建、打开或截取的文件、管道、通信资源、磁盘设备或控制台的名称。
dwDesAccess:指定文件的输出类型。
dwShareMode:确定是否且如何共享这个文件。
lpSecurityAttributes:是指向 SECURITY_ATTRIBUTES 结构的指针，指定了目录的安全属性，但要求文件系统支持如 NTFS 的格式。Windows 98 不支持此属性，在函数调用时应设置为 NULL。

dwCreationDisposition:确定文件存在或不存在时所采取的动作。
dwFlagsAndAttributes：指定文件的属性和标志。
hTemplateFile：用于存取模板文件的句柄，模板文件为正在创建的文件提供扩展属性。
返回值
如果函数调用成功则返回打开文件的句柄。如果调用前文件已经存在，且 dwCreationDisposition 参数使用 CREATE_ALWAYS 或 OPEN_ALWAYS，则返回 ERROR_ALREADY_EXISTS。函数调用失败则返回 INVALID_HANDLE_VALUE。

2) ReadFile

```
BOOL ReadFile(HANDLE hFile, //文件指针
LPVOID lpBuffer, //数据缓冲
DWORD nNumberOfBytesToRead, //读取的字节数
LPDWORD lpNumberOfBytesRead, //接收要读取的字节数
LPOVERLAPPED lpOverlapped //覆盖缓冲)
```

参数 hFile: 是指向要打开文件的指针。
lpBuffer: 是接收来自文件数据缓冲区的指针。
nNumberOfBytesToRead: 指从文件中读取的字节数。
lpNumberOfBytesRead: 用于接收要读取的字节数。
lpOverlapped: 是指向 OVERLAPPED 结构的指针，如果 hFile 所指向的文件是用 FILE_FLAG_OVERLAPPED 创建的，则需要用到此结构。
返回值：如果函数调用成功则返回值为 TRUE,否则为 FALSE。

3) WriteFile

```
BOOL WriteFile (HANDLE hFile,
LPCVOID lpBuffer,
DWORD nNumberOfBytesToWrite,
LPDWORD lpNumberOfBytesWritten,
LPOVERLAPPED lpOverlapped);
```

其参数设置与读取文件函数 ReadFile 大同小异，只需要将读取改成写入即可，返回值也很相似，在这里就不多介绍了。

4) CloseHandle

```
BOOL CloseHandle(HANDLE hObject);
```

5) SetFilePointer

```
DWORD SetFilePointer(
HANDLE hFile, // 文件句柄
LONG lDistanceToMove, // 偏移量(低位)
PLONG lpDistanceToMoveHigh, // 偏移量(高位)
DWORD dwMoveMethod)
```

dwMoveMethod 取值：基准位置 FILE_BEGIN:文件开始位置 FILE_CURRENT:文件当前位置 FILE_END:文件结束位置
说明:移动一个打开文件的指针

2. 第一次使用 linux，在实验过程中学习了安装 linux 操作系统，配置 c++编译环境，编写系统调用程序。参考了很多网上的资料，最终初步熟悉了 linux 操作系统及其编译环境。对系统调用有了更深的理解。