



院 系 数据科学与计算机学院

学号 19335174

姓名 施天予

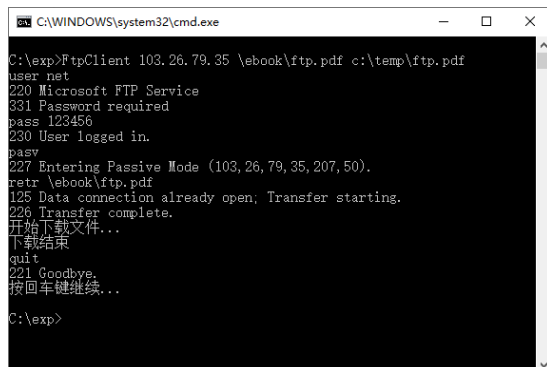
## 【实验题目】文件传输实验

【实验目的】学习利用套接字传送文件

## 【实验内容】

1、(FtpClient)写一个Ftp 客户端程序，用FTP 协议下载多个文件到指定目录。

### (1) 参考运行截屏：



注意，其中的命令不是键入的而是编程把字符串直接输出给服务器的。  
经验证，下载的文件可以打开。

### (2) 老师用的函数(仅作参考):

```
strchr(); strrchr(); sprintf(buf, "retr %s\r\n", filename);
```

```
sscanf(st, "%d,%d,%d,%d,%d,%d", &ip1, &ip2, &ip3, &ip4, &port1, &port2);
```

```
SOCKET dataConn() { ... }
```

建立数据连接

```
unsigned __stdcall recvFromNet(void *){...}
```

接收服务器的消息（线程函数）

```
void sendmsg(SOCKET sock, char *msg){ ... }
```

发送控制消息

```
int saveFile(SOCKET sock, char * fileName) { ... }
```

接收数据并保存为文件

### (3) 源代码:

```
#include <stdio.h>
#include <stdlib.h>
#include <io.h>
#include <iostream>
#include <winsock2.h>
#include <string.h>
#include <windows.h>
#include <math.h>
#include <ws2tcpip.h>
#include <process.h>
using namespace std;

#define LEN 2000
#define WSVERS MAKEWORD(2, 0)
#pragma comment(lib, "ws2_32.lib")
```



```
char* host;
int finish = 0;
bool download = false;
int dport = 0;

struct Sub {
    SOCKET control;
    string dest;
    Sub(SOCKET& s, string f) {
        control = s;
        dest = f;
    }
};

unsigned __stdcall fun(void* ptr) {
    char* ptr2;
    char buf[LEN];
    SOCKET s = reinterpret_cast<Sub*>(ptr)->control;
    FILE* file = NULL;
    memset(buf, 0, LEN);
    while (1) {
        int cc = recv(s, buf, LEN, 0);
        if (cc == SOCKET_ERROR || cc == 0) {
            printf("Error\n");
            return 1;
        }
        else {
            buf[cc] = '\0';
            printf("%s", buf);
            if (!strncmp(buf, "227", 3)) {
                string p(buf);
                size_t pos1 = p.find_last_of(',');
                string port1 = p.substr(pos1 - 3, pos1 - 1);
                string port2 = p.substr(pos1 + 1, pos1 + 3);
                dport = _strtol_l(port1.c_str(), &ptr2, 10, NULL) * 256 +
                _strtol_l(port2.c_str(), &ptr2, 10, NULL);
                SOCKET dsocket;
                struct sockaddr_in dsin;                /* an Internet endpoint address
                */
                WSADATA wsadata;
                WSAStartup(WSVERS, &wsadata);                //加载winsock
                library。WSVERS为请求的版本, wsadata返回系统实际支持的最高版本
                dsocket = socket(PF_INET, SOCK_STREAM, IPPROTO_TCP);                //创建套接
                字, 参数: 因特网协议簇(family), 流套接字, TCP协议
                memset(&dsin, 0, sizeof(dsin));
                dsin.sin_family = AF_INET;
```



---

```
dsin.sin_addr.s_addr = inet_addr(host);
dsin.sin_port = htons((u_short)dport);
connect(dsocket, (struct sockaddr*)&dsin, sizeof(dsin));
download = true;
while (1) {
    string x = reinterpret_cast<Sub*>(ptr)->dest;
    while (_access(x.c_str(), 0) == 0);
    file = fopen(x.c_str(), "wb");
    if (file == NULL) {
        printf("File can not find!\n");
        printf("Please press any key to continue...");
        getchar();
        getchar();
        exit(1);
    }
    printf("Start downloading...\n");
    while (1) {
        int cc2 = recv(dsocket, buf, LEN, 0);
        int f = fwrite(buf, 1, cc2, file);
        if (cc2 <= 0) break;
    }
    printf("Finish!\n");
    fclose(file);
    finish++;
    memset(buf, 0, LEN);
}
download = false;
}
}
return 0;
}

int main(int argc, char* argv[]) {
    host = argv[1];          /* server IP to connect */
    const char* service = "21"; /* server port to connect */
    string source;
    string dest;
    source = argv[2];
    dest = argv[3];
    struct sockaddr_in sin;    /* an Internet endpoint address */
    SOCKET sock;              /* socket descriptor */

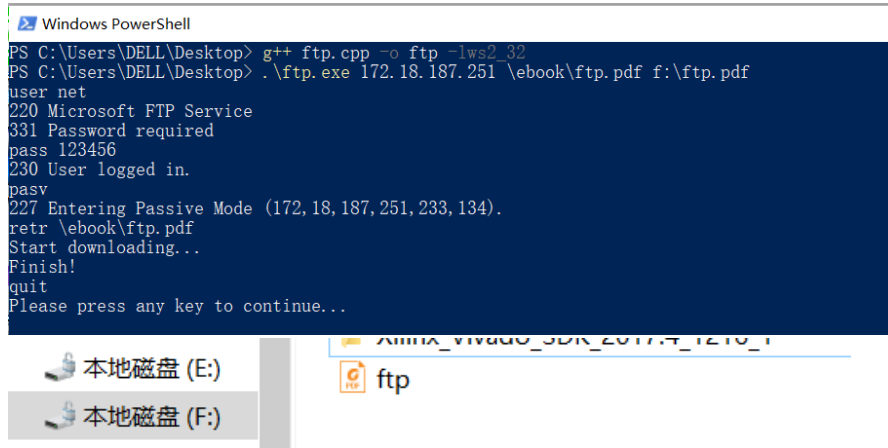
    WSADATA wsadata;
    WSStartup(WVSERS, &wsadata);
    sock = socket(PF_INET, SOCK_STREAM, IPPROTO_TCP);
```



```
memset(&sin, 0, sizeof(sin));
sin.sin_family = AF_INET;
sin.sin_addr.s_addr = inet_addr(host);
sin.sin_port = htons((u_short)atoi(service));
connect(sock, (struct sockaddr*)&sin, sizeof(sin));
Sub s(sock, dest);
HANDLE thread = (HANDLE)_beginthreadex(NULL, 0, &fun, (void*)&s, 0, NULL);
string* command = new string[5];
command[0] = "user net\r\n";
command[1] = "pass 123456\r\n";
command[2] = "pasv\r\n";
command[3] = "retr " + source + "\r\n";
command[4] = "quit\r\n";
for (int j = 0; j < 5; j++) {
    if (command[j].substr(0, 4) == "retr")
        while (!download);
    if (command[j].substr(0, 4) == "quit")
        while (!finish);
    printf("%s", command[j].c_str());
    send(sock, command[j].c_str(), command[j].length(), 0);
    Sleep(50);
}
CloseHandle(thread);
closesocket(sock);

WSACleanup();
printf("Please press any key to continue...");
getchar();
return 0;
}
```

#### (4) 运行截图及说明:



文件 [ftp.pdf](#) 被正确下载到了 F 盘下!



2、(P2pChat)通过建立 TCP 连接实现一对一的聊天和传输文件功能。

(1) 功能说明:

在客户和服务端之间建立 TCP 连接之后,任何一方可以输入并执行命令:

>rdir d:\test	设置接收文件的目录 d:\test
>chat hello	向对方发送聊天字符串“hello”或者 >hello (非命令即可)
>send c:\temp\ftp.pdf	向对方发送文件 ftp.pdf,接收方对重名文件加编号。
>quit	退出程序

(2) 数据包设计(仅作参考):

- 一个包由三部分构成:结构 1,结构 2(多种类型),数据  
其中:结构 1 和结构 2 分别包含数据类型和数据长度。
- 在接收结构 1 之后,通过数据类型确定结构 2,不同数据类型的结构 2 可以不同,再通过结构 2 中的数据长度接收数据。
- 接收结构和数据均要根据该结构或数据的长度接收。要累计已接收的字节数,直到全部接收完毕,再接收下一部分。

(3) 老师用的函数(仅作参考):

```
// 获得文件大小
__int64 getFileSize(char * fileName);

// 从带文件名的路径中提取文件名
char * getFileName(char *pathName);

// 从filePathName提取长度为len的字符串到fileFullName (不包含扩展名的文件名)
int mystrcpy(char * fileFullName, int len, char * filePathName);

// 获得唯一文件名,即下载的重名文件加序号
void getUniqueName(char *newFileName, char *filePathName);

// 接收len个字节存放到buf中
int recvData(SOCKET sock, char *buf, int len);

// 接收长度为fileSize的文件存放用fileName到路径path中
int recvFile(SOCKET sock, char * fileName, int fileSize, char *path);

// 从网上接收数据(线程函数)
unsigned __stdcall myrecv(void *p);

// 读出文件srcFileName发送到网上
int sendFile(SOCKET sock, char *srcFileName);

// 发送聊天包:发送结构1(包含数据类型),发送结构2(包含数据长度),发送聊天数据
void sendChatPacket(SOCKET sock, char *chatData);

// 发送文件包:发送结构1(包含数据类型),发送结构2(包含数据长度),发送文件
void sendFilePacket(SOCKET sock, char * fullFileName);

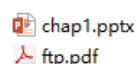
// 执行命令: quit,exit,rdir,chat,file。返回值: 0-正常, -1-退出
int executeCmd(char * cmdLine);

// 建立连接,循环: 输入命令和执行命令
void main(int argc, char *argv[]);
```

(4) 参考程序: P2PServer.exe, P2PClient.exe

(5) 参考运行情况:

起始时, c:\test 为空目录, c:\temp 有两个文件





运行:

```
C:\exp>P2PServer
等待连接...
连接成功!
>rdir c:\test
>file c:\temp\ftp.pdf
收到文件: ftp.pdf
>file c:\temp\chap1.ppt
文件不存在!
>file c:\temp\chap1.pptx
>file c:\temp\ftp.pdf

C:\exp>P2PClient
正在连接...
连接成功!
>rdir c:\test2
收到文件: ftp.pdf
>file c:\temp\ftp.pdf
收到文件: chap1.pptx
收到文件: ftp.pdf
```

\* 数字为输入或显示次序

运行后: c:\test 有一个文件, c:\test2 有三个文件。它们均可以正常打开。



## (5) 源代码:

服务器

```
#include <stdio.h>
#include <stdlib.h>
#include <io.h>
#include <iostream>
#include <winsock2.h>
#include <string.h>
#include <windows.h>
#include <math.h>
#include <ws2tcpip.h>
#include <process.h>
using namespace std;

#define LEN 2000 // 接收缓冲区大小
#define WSVERS MAKEWORD(2, 0) // 指明版本2.0
#pragma comment(lib, "ws2_32.lib") // 使用winsock 2.0 Llibrary

struct MyFile {
    char name[LEN];
    long size;
};

WSADATA wsadata;
struct sockaddr_in fsin; // an Internet endpoint address */
HANDLE thread;

int cc; // recv character count */
SOCKET msock, sock; // socket descriptor */
char path[LEN];
```



```
char in[LEN]; //输入的命令
char* ss; //命令类型

unsigned __stdcall fun(void* x) {
    FILE* file = NULL;
    char buf[LEN];
    memset(buf, 0, LEN);
    while (1) {
        char type;
        cc = recv(sock, &type, 1, 0);
        if (type == 1) {
            MyFile f;
            cc = recv(sock, (char*)&f, sizeof(MyFile), 0);
            char* ptr = strrchr(f.name, '\\');
            string x = path;
            string y = ptr;
            x += y;
            int count = 1;
            while (access(x.c_str(), 0) == 0) {
                string s = "(0)";
                s[1] += count;
                int pos = x.rfind('.');
                string pathname = x.substr(0, pos);
                string type = x.substr(pos, x.length() - pos);
                pos = x.rfind('(');
                pathname = pathname.substr(0, pos);
                pathname += s;
                pathname += type;
                x = pathname;
                count++;
            }
            if (fopen_s(&file, x.c_str(), "wb") != NULL) {
                printf("文件不存在! \n");
                getchar();
                getchar();
                exit(1);
            }
            long sum = 0;
            while (f.size - sum >= LEN) {
                cc = recv(sock, buf, LEN, 0);
                int len = fwrite(buf, 1, cc, file);
                sum += len;
            }
            cc = recv(sock, buf, f.size - sum, 0);
            fwrite(buf, 1, cc, file);
            if (cc <= 0) {
```



```
        printf("Error: %d.\n", GetLastError());
        return 1;
    }
    printf("收到文件: %s\n", ptr + 1);
    fclose(file);
}
else {
    MyFile f;
    cc = recv(sock, (char*)&f, sizeof(MyFile), 0);
    cc = recv(sock, buf, f.size, 0);
    if (cc <= 0) {
        printf("Error: %d.\n", GetLastError());
        return 1;
    }
    buf[cc] = '\0';
    printf("%s\n", buf);
}
}
return 0;
}

void main(int argc, char* argv[]) {
    const char* service = "50500";           /* server port to connect */

    struct sockaddr_in sin;                  //an Internet endpoint
    address
    WSStartup(WSVERS, &wsadata);             //加载winsock library。WSVERS
    为请求的版本，wsadata返回系统实际支持的最高版本
    msock = socket(PF_INET, SOCK_STREAM, IPPROTO_TCP); //创建套接字，参数：因特网协
    议簇(family)，流套接字，TCP协议

    FILE* file = NULL;
    char* fname = new char[LEN];

    memset(&sin, 0, sizeof(sin));            // 从&sin开始的长度为
    sizeof(sin)的内存清0
    sin.sin_family = AF_INET;                 // 因特网地址簇(INET-Internet)
    sin.sin_addr.s_addr = INADDR_ANY;        // 监听所有(接口的)IP地址。
    sin.sin_port = htons((u_short)atoi(service)); // 监听的端口号。atoi一把ascii
    转化为int，htons--主机序到网络序(16位)
    bind(msock, (struct sockaddr*)&sin, sizeof(sin)); // 绑定监听的IP地址和端口号
    listen(msock, 5);                          // 等待建立连接的队列长度为10

    int alen = sizeof(struct sockaddr);       // 取到地址结构的长度
    sock = accept(msock, (struct sockaddr*)&fsin, &alen); // 如果有新的连接请求，返回连
    接套接字，否则，被阻塞。fsin包含客户端IP地址和端口号
```





---

```
thread = (HANDLE)_beginthreadex(NULL, 0, &fun, NULL, 0, NULL);  
printf("服务器已连接\n");
```

```
while (1) {  
    printf(">>");  
    gets_s(in);  
    if (strcmp(in, "")) {  
        ss = strtok(in, " ");  
        if (strcmp(ss, "rdir") == 0) {  
            char* x = strtok(NULL, " ");  
            string a = x;  
            memcpy(path, a.c_str(), a.length());  
        }  
        else if (strcmp(ss, "chat") == 0) {  
            char* m = strtok(NULL, " ");  
            char type = 2;  
            MyFile f;  
            memset(f.name, 0, LEN);  
            f.size = strlen(m);  
            send(sock, &type, 1, 0);  
            send(sock, (char*)&f, sizeof(MyFile), 0);  
            send(sock, m, strlen(m), 0);  
        }  
        else if (strcmp(ss, "send") == 0) {  
            fname = strtok(NULL, " ");  
            MyFile f;  
            char buf[LEN];  
            int len;  
            if (fopen_s(&file, fname, "rb") != NULL) {  
                printf("文件未找到! \n");  
                break;  
            }  
            char type = 1;  
            strcpy(f.name, fname);  
            fseek(file, 0, SEEK_END);  
            f.size = ftell(file);  
            fseek(file, 0, SEEK_SET);  
            send(sock, &type, 1, 0);  
            send(sock, (char*)&f, sizeof(MyFile), 0);  
            while ((len = fread(buf, 1, LEN, file)) >= LEN)  
                send(sock, buf, len, 0);  
            send(sock, buf, len, 0);  
        }  
        else if (strcmp(ss, "quit") == 0) {  
            printf("客户端已退出\n");  
            break;  
        }  
    }  
}
```



```
    }
    else {
        printf("错误的命令! \n");
    }
}

}

CloseHandle(thread);
shutdown(sock, SD_BOTH);
WSACleanup(); // 卸载winsock library
printf("按回车键继续...");
getchar(); // 等待任意按键
}
```

客户端

```
#include <stdio.h>
#include <stdlib.h>
#include <io.h>
#include <iostream>
#include <winsock2.h>
#include <string.h>
#include <windows.h>
#include <math.h>
#include <ws2tcpip.h>
#include <process.h>
using namespace std;

#define LEN 2000 // 接收缓冲区大小
#define WSVERS MAKEWORD(2, 0) // 指明版本2.0
#pragma comment(lib, "ws2_32.lib") // 使用winsock 2.0 Library

struct MyFile {
    char name[LEN];
    long size;
};

WSADATA wsadata;
struct sockaddr_in fsin; /* an Internet endpoint address */
HANDLE thread;
int cc; /* recv character count */
SOCKET sock; /* socket descriptor */
char path[LEN];
char in[LEN]; // 输入的命令
char* ss; // 命令类型

unsigned __stdcall fun(void* x) {
    FILE* file = NULL;
    char buf[LEN];
```



```
memset(buf, 0, LEN);
while (1) {
    char type;
    cc = recv(sock, &type, 1, 0);
    if (type == 1) {
        MyFile f;
        cc = recv(sock, (char*)&f, sizeof(MyFile), 0);
        char* ptr = strchr(f.name, '\\');
        string x = path;
        string y = ptr;
        x += y;
        int count = 1;
        while (access(x.c_str(), 0) == 0) {
            string s = "(0)";
            s[1] += count;
            int pos = x.rfind('.');
            string pathname = x.substr(0, pos);
            string type = x.substr(pos, x.length() - pos);
            pos = x.rfind('(');
            pathname = pathname.substr(0, pos);
            pathname += s;
            pathname += type;
            x = pathname;
            count++;
        }
        if (fopen_s(&file, x.c_str(), "wb") != NULL) {
            printf("文件不存在! \n");
            getchar();
            getchar();
            exit(1);
        }
        long sum = 0;
        while (f.size - sum >= LEN) {
            cc = recv(sock, buf, LEN, 0);
            int len = fwrite(buf, 1, cc, file);
            sum += len;
        }
        cc = recv(sock, buf, f.size - sum, 0);
        fwrite(buf, 1, cc, file);
        if (cc <= 0) {
            printf("Error: %d.\n", GetLastError());
            return 1;
        }
        printf("收到文件: %s\n", ptr + 1);
        fclose(file);
    }
}
```



```
else {
    MyFile f;
    cc = recv(sock, (char*)&f, sizeof(MyFile), 0);
    cc = recv(sock, buf, f.size, 0);
    if (cc <= 0) {
        printf("Error: %d.\n", GetLastError());
        return 1;
    }
    buf[cc] = '\0';
    printf("%s\n", buf);
}
}
return 0;
}

void main(int argc, char* argv[]) {
    const char* host = "127.0.0.1";
    const char* service = "50500";           /* server port to connect */

    struct sockaddr_in sin;                  //an Internet endpoint
    address
    WSStartup(WSVERS, &wsadata);             //加载winsock library。WSVERS
    为请求的版本，wsadata返回系统实际支持的最高版本
    sock = socket(PF_INET, SOCK_STREAM, IPPROTO_TCP); //创建套接字，参数：因特网协
    议簇(family)，流套接字，TCP协议

    FILE* file = NULL;
    char* fname = new char[LEN];

    memset(&sin, 0, sizeof(sin));             // 从&sin开始的长度为
    sizeof(sin)的内存清0
    sin.sin_family = AF_INET;                  // 因特网地址簇
    sin.sin_addr.s_addr = inet_addr(host);     // 服务器IP地址(32位)
    sin.sin_port = htons((u_short)atoi(service)); // 服务器端口号
    connect(sock, (struct sockaddr*)&sin, sizeof(sin)); // 连接到服务器

    thread = (HANDLE)_beginthreadex(NULL, 0, &fun, NULL, 0, NULL);
    printf("客户端已连接\n");

    while (1) {
        printf(">>");
        gets_s(in);
        if (strcmp(in, "")) {
            ss = strtok(in, " ");
            if (strcmp(ss, "rdir") == 0) {
                char* x = strtok(NULL, " ");
            }
        }
    }
}
```



```
        string a = x;
        memcpy(path, a.c_str(), a.length());
    }
    else if (strcmp(ss, "chat") == 0) {
        char* m = strtok(NULL, " ");
        char type = 2;
        MyFile f;
        memset(f.name, 0, LEN);
        f.size = strlen(m);
        send(sock, &type, 1, 0);
        send(sock, (char*)&f, sizeof(MyFile), 0);
        send(sock, m, strlen(m), 0);
    }
    else if (strcmp(ss, "send") == 0) {
        fname = strtok(NULL, " ");
        MyFile f;
        char buf[LEN];
        int len;
        if (fopen_s(&file, fname, "rb") != NULL) {
            printf("文件未找到! \n");
            break;
        }
        char type = 1;
        strcpy(f.name, fname);
        fseek(file, 0, SEEK_END);
        f.size = ftell(file);
        fseek(file, 0, SEEK_SET);
        send(sock, &type, 1, 0);
        send(sock, (char*)&f, sizeof(MyFile), 0);
        while ((len = fread(buf, 1, LEN, file)) >= LEN)
            send(sock, buf, len, 0);
        send(sock, buf, len, 0);
    }
    else if (strcmp(ss, "quit") == 0) {
        printf("客户端已退出\n");
        break;
    }
    else {
        printf("错误的命令! \n");
    }
}

CloseHandle(thread);
shutdown(sock, SD_BOTH);
WSACleanup();
printf("按回车键继续...");
```

// 卸载winsock library



```
getchar(); // 等待任意按键  
}
```

## (6) 服务器运行截屏:

C:\Users\DELL\Desktop\P2Pserver.exe

```
服务器已连接  
>>hi  
chat hello  
>>send e:\sty.txt  
>>rdir e:\  
>>收到文件: hhh.txt
```

## (7) 客户端运行截屏:

C:\Users\DELL\Desktop\P2Pclient.exe

```
客户端已连接  
>>chat hi  
>>hello  
rdir f:\  
>>收到文件: sty.txt  
send f:\hhh.txt
```

经检验, 发送的文件能正确收到!

## (8) 与同学互测 (选做)

同学的 IP 地址 172.26.30.163

我的客户端

E:\VSsource\ConsoleApplication1\Release\1.exe

```
客户端已连接  
>>chat hi  
>>hey  
send f:\ftp.pdf  
>>rdir e:\  
>>收到文件: chapter.ppt
```

本地磁盘 (D:)	WinHex	2021/3/19 11:10	文件夹
本地磁盘 (E:)	迅雷下载	2021/3/6 10:37	文件夹
本地磁盘 (F:)	chapter	2021/4/11 23:26	Microsoft Power... 2,192 KB

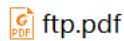
## 我的服务器

D:\Wetchatfiles\WeChat Files\wxid\_8qhonsckx54e22\FileStorage\File\2021-04\P2Pserver.exe

```
服务器已连接  
>>hi  
chat hey  
>>rdir D:\test  
>>收到文件: ftp.pdf  
send D:\cnexp3\ebook\chapter.ppt  
>>
```



摘要 > Data (D:) > test



ftp.pdf

## 【完成情况】

是否完成以下步骤? (√完成 ×未做)

1 [√] 2 [√]

第 2(8)步互测 (选做) 的同学的学号姓名: 19335177 孙奥远

## 【实验体会】

本次实验是在之前几个实验的提升, 结合了前几次实验所学的内容。在做 ftp 实验时, 我一开始没仔细看, 以为 ip 地址是 103.26.79.35, 后来想起来应该是 172.18.187.251, 导致我总是连接失败, 找不出问题, 太粗心了! 在做 P2P 实验时, 我在接收部分一开始不知道怎么辨别是接收数据还是文件, 后来发现可以用两次套接字 send 函数, 第一次判断类型, 第二次发送数据或文件, 就能成功实现了。因为这次的两个实验涉及大量的字符串操作, 所以我就使用了 C++ 里的一些函数, 省去了很多麻烦。

传输所有类型的文件而不仅仅是文本文件, 需要使用二进制方式打开文件, 也就是 “rb” 和 “wb”。文本文件也视作二进制文件。如果只是以文本方式打开文件的话, 那么在处理二进制文件时就可能发生意外情况。

总而言之, 这次实验虽然耗费了我大量的精力, 但也让我学到了很多, 受益匪浅。

## 【交实验报告】

- (1) 每位同学单独完成本实验内容并填写实验报告。
- (2) 上交网址: <http://172.18.187.251/netdisk/default.aspx?vm=19net> 实验上交/编程实验
- (3) 截止日期 (不迟于): 2021 年 4 月 16 日 23:00 (周五)。

上传文件: 学号\_姓名\_文件传输实验.doc

学号\_姓名\_文件传输实验.rar