



院 系 数据科学与计算机学院 班 级 18 计算机 学号 18340215 姓名 张天祯

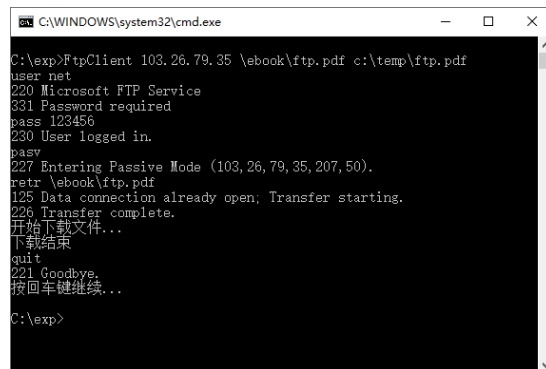
【实验题目】文件传输实验

【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 *p) {...}
```

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:

```
#pragma warning( disable : 4996)
```

```
#include <stdlib.h>
```

```
#include <stdio.h>
```

```
#include <winsock2.h>
```

```
#include <string.h>
```

```
#include <process.h>
```

```
#include <ws2tcpip.h>
```

```
#define WSVERS AKEWORD(2, 0)
```

// 指明版本 2.0

```
#pragma comment(lib,"ws2_32.lib")
```

// 使用 winsock 2.0 Llibrary

```
#define BUFLen (10 << 24)
```

// 缓冲区大小

```
#define FTP_MSG_1 "user net\r\n"
```

```
#define FTP_MSG_2 "pass 123456\r\n"
```

```
#define FTP_MSG_3 "pasv\r\n"
```

```
#define FTP_MSG_4 "quit\r\n"
```



```
#define FTP_SG "227"
```

```
char argv_3[100];
```

```
char buff[BUFLEN + 1];
```

```
char data_buff[BUFLEN + 1];
```

```
unsigned __stdcall rec_2(void* sock)
```

```
{
```

```
    int cc;
```

```
    SOCKET* ssock = (SOCKET*)sock;
```

```
    FILE* file = fopen(argv_3, "wb");
```

```
    fclose(file);
```

```
    //printf("%s\n", ((agmt*)ag)->argv_3);
```

```
    puts("开始下载文件...");
```

```
    while (1)
```

```
    {
```

```
        cc = recv(*ssock, data_buff, BUFLen, 0);           // cc 为接收到的
        字符的个数(>0)或对方已关闭(=0)或连接出错(<0)
```

```
        //printf("%s\n", data_buff);
```

```
        if (cc == SOCKET_ERROR || cc == 0)
```

```
        {
```

```
            //printf("Error: %d.\n", GetLastError());      //出错。其后必须
            关闭套接字 sock。
```

```
            break;
```

```
        }
```

```
        else if (cc > 0) {
```

```
            FILE* file = fopen(argv_3, "ab");
```

```
            if (!file) puts("FILE OPEN ERROR!\n");
```

```
            fwrite(data_buff, 1, cc, file);
```

```
            fclose(file);
```

```
        }
```

```
    }
```

```
    puts("下载结束");
```

```
    return 0;
```

```
}
```

```
void dataConn()
```

```
{
```

```
    char* pos1 = strchr(buff, '(');
```

```
    int ip1, ip2, ip3, ip4, port1, port2;
```

```
    sscanf(pos1 + 1, "%d,%d,%d,%d,%d,%d", &ip1, &ip2, &ip3, &ip4, &port
1, &port2);
```

```
    char rst[100] = {};
```

```
    sprintf(rst, "%d.%d.%d.%d", ip1, ip2, ip3, ip4);
```

```
    struct addrinfo* host;
```

```
    char port[100] = {};
```



```
itoa(port1 * 0x100 + port2, port, 10);
getaddrinfo(rst, port, NULL, &host);
SOCKET sock = socket(host->ai_family, host->ai_socktype, host->ai_protocol);
if (connect(sock, host->ai_addr, host->ai_addrlen)) puts("Date Connect Error!");
//printf("%s\n", ag.argv_3);
HANDLE thr = (HANDLE)_beginthreadex(NULL, 0, &rec_2, (void*)&sock, 0, NULL);
WaitForSingleObject(thr, INFINITE);
CloseHandle(thr);
shutdown(sock, SD_BOTH);
closesocket(sock);
freeaddrinfo(host);
}

unsigned __stdcall rec_1(void* sock)
{
    int cc;
    SOCKET* ssock = (SOCKET*)sock;
    while (1)
    {
        cc = recv(*ssock, buff, BUFLen, 0); // cc 为接收到的
        // 字符的个数(>0)或对方已关闭(=0)或连接出错(<0)
        if (cc == SOCKET_ERROR || cc == 0)
        {
            //printf("Error: %d.\n", GetLastError()); // 出错。其后必须
            // 关闭套接字 sock。
            break;
        }
        else if (cc > 0) {
            buff[cc] = '\0'; // ensure null-termination
            int pos = 0;
            while (pos < cc)
            {
                printf("%c", buff[pos]);
                pos++;
            }
            if(strchr(buff, '(')) dataConn();
        }
    }
    return 0;
}
```



```
int main(int argc, char** argv)
{
    if (argc != 4) return 0;
    WSADATA wsadata;
    WSStartup(MAKEWORD(2, 0), &wsadata);
    struct addrinfo* host;
    getaddrinfo(argv[1], "ftp", NULL, &host);
        //默认 ftp
    SOCKET sock = socket(host->ai_family, host->ai_socktype, host->ai_protocol);
    if (connect(sock, host->ai_addr, host->ai_addrlen)) printf("Connect Error!\n");
    strcpy(argv_3,argv[3]);
    HANDLE thr = (HANDLE)_beginthreadex(NULL, 0, &rec_1, (void*)&sock, 0, NULL);

    char buf[100] = FTP_MSG_1;
    printf("%s", buf);
    send(sock, buf, strlen(buf), 0);
    strcpy(buf, FTP_MSG_2);
    printf("%s", buf);
    send(sock, buf, strlen(buf), 0);
    strcpy(buf, FTP_MSG_3);
    printf("%s", buf);
    send(sock, buf, strlen(buf), 0);
    sprintf(buf, "retr %s\r\n", argv[2]);
    printf("%s", buf);
    send(sock, buf, strlen(buf), 0);
    strcpy(buf, FTP_MSG_4);
    printf("%s", buf);
    send(sock, buf, strlen(buf), 0);

    WaitForSingleObject(thr, INFINITE);
    CloseHandle(thr);
    shutdown(sock, SD_BOTH);
    closesocket(sock);
    freeaddrinfo(host);
    WSACleanup();
    return 0;
}
```

(4) 运行截图及说明:



```
user net
pass 123456
pasv
retr \ebook\ftp.pdf
quit
220 Microsoft FTP Service
331 Password required
230 User logged in.
227 Entering Passive Mode (103,26,79,35,250,38).
150 Opening ASCII mode data connection.
开始下载文件...
下载结束
226 Transfer complete.
221 Goodbye.

C:\Users\E480\Desktop\计网\实验\实验五\MY\FtpClient\
Debug\FtpClient.exe (进程 8604) 已退出, 代码为 0。
```

计网 > 实验 > 实验五 > MY

搜索"MY"

名称	修改日期	类型	大小
 ftp.pdf	2020/6/10 15:56	Foxit PhantomP...	84 KB

经验证可正常打开。

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:

在客户和服务器之间建立 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);

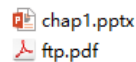
// 执行命令: quit, exit, rdir, chat, file。返回值: 0-正常, -1-退出
int executeCmd(char * cmdLine);

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

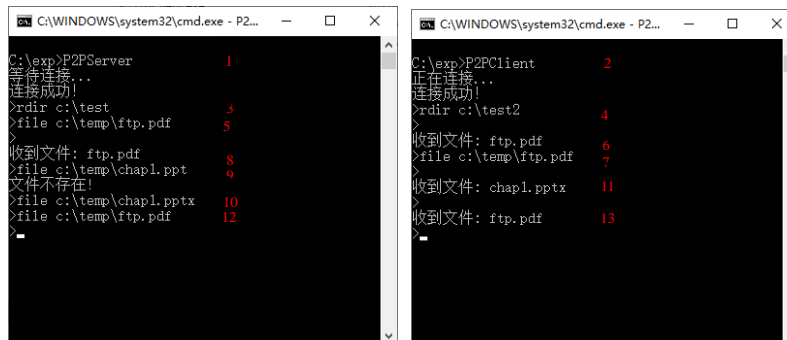
(4) 参考程序: P2PServer.exe, P2PClient.exe

(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:



* 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:

```
#pragma warning(disable : 4996)
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <winsock2.h>
#include <time.h>
#include <process.h>
#include <sys/stat.h>
#include "conio.h" // 仅能用于Dos类操作系统
#pragma comment(lib, "ws2_32.lib") // 使用winsock 2.2 library
#define WSVERS MAKEWORD(2, 2) // MAKEWORD函数创建一个符合网络传输(小端?)的2B的word, 这里的参数表示后面的WSAStartup将使用版本号2.2(换成2.0也行)
#define SERVICE "50500"
```



```
#define QUSIZE 2
#define BUFFSIZE (10 << 24)
#define RDIR "rdir"
#define CHAT "chat"
#define SEND "send"
#define QUIT "quit"
#define MAX_FILE_NAME 200
#define MAX_FILE_TYPE 10
#define PDF "pdf"
#define TXT "txt"
#define JPG "jpg"
#define PNG "png"
#define PPTX "pptx"
#define PDF_Struct int
#define TXT_Struct int
#define JPG_Struct int
#define PNG_Struct int
#define PPTX_Struct int

typedef struct {
    char fileName[MAX_FILE_NAME];
    char fileType[MAX_FILE_TYPE];
}FileStruct;

char buf[BUFFSIZE];
char file_name_head[MAX_FILE_NAME];
char data_buf[BUFFSIZE];

void app_str(char* str, int num)
{
    strcpy(str, "(");
    char tem[10];
    itoa(num, tem, 10);
    strcat(str, tem);
    strcat(str, ")");
}

void add_str(char* str, char* front_app)
{
    int len = strlen(front_app);
    char tem[100] = {};
    strcpy(tem, front_app);
    tem[len] = '\\';
    tem[len + 1] = '\\0';
    strcpy(tem + len + 1, str);
    strcpy(str, tem);
}
```



```
}
```

```
void send_file(SOCKET ssock, char* file_name)
{
    int len = 0;
    FileStruct tem;
    strcpy(tem.fileName, file_name);
    char* pos = strchr(file_name, '.');
    strcpy(tem.fileType, pos + 1);
    memcpy(data_buf, &tem, sizeof(FileStruct));
    len += sizeof(FileStruct);
    add_str(file_name, file_name_head);
    long long sz = 0;
    if (!strcmp(tem.fileType, PDF) || !strcmp(tem.fileType, TXT)
        || !strcmp(tem.fileType, JPG) || !strcmp(tem.fileType, PNG)
        || !strcmp(tem.fileType, PPTX))
    {
        PDF_Struct size;
        struct stat statbuf;
        if (stat(file_name, &statbuf) != 0)
        {
            printf("File Size Error!\n");
            return;
        }
        size = statbuf.st_size;
        sz = size;
        memcpy(data_buf+len, &size, sizeof(PDF_Struct));
        len += sizeof(PDF_Struct);
    }
    else {
        printf("File Type Error!\n");
        return;
    }
    send(ssock, data_buf, len, 0);
    FILE* file = fopen(file_name, "rb");
    if (!file)
    {
        printf("FILE OPEN ERROR!\n");
        return;
    }
    while (sz>0)
    {
        fread(data_buf, 1, BUFFSIZE>sz?sz:BUFFSIZE, file);
        sz -= send(ssock, data_buf, BUFFSIZE > sz ? sz : BUFFSIZE, 0);
    }
    fclose(file);
}
```




```
}
```

```
unsigned __stdcall rec(void* ssock)
{
    int cc;
    SOCKET* sock = (SOCKET*)ssock;
    while (1)
    {
        cc = recv(*sock, buf, BUFFSIZE, 0);
        if (cc == SOCKET_ERROR || cc == 0)
        {
            printf("Error: %d.\n", GetLastError());    //出错。其后必须关闭套接字sock。
            break;
        }
        else if (cc > MAX_FILE_NAME) {
            FileStruct tem;
            memcpy(&tem, buf, sizeof(FileStruct));
            char file_name[MAX_FILE_NAME];
            strcpy(file_name, tem.fileName);
            add_str(file_name, file_name_head);
            long long sz = 0;
            if (!strcmp(tem.fileType, PDF) || !strcmp(tem.fileType, TXT)
                || !strcmp(tem.fileType, JPG) || !strcmp(tem.fileType, PNG)
                || !strcmp(tem.fileType, PPTX))
            {
                PDF_Struct size;
                memcpy(&size, buf + sizeof(FileStruct), sizeof(PDF_Struct));
                sz = size;
            }
            FILE* file;
            int num = 2;
            while (1)
            {
                file = fopen(file_name, "rb");
                if (file)
                {
                    fclose(file);
                    char app[10] = {};
                    app_str(app, num);
                    num++;
                    char* pos;
                    pos = strrchr(file_name, '(');
                    if (!pos) pos = strchr(file_name, '.');
                    strcpy(pos, app);
                    strcat(file_name, ".");
                }
            }
        }
    }
}
```



```
        strcat(file_name, tem.fileType);
    }
    else {
        break;
    }
}
file = fopen(file_name, "wb");
if (!file)
{
    printf("FILE OPEN ERROR!\n");
    break;
}
while (sz)
{
    cc = recv(*sock, buf, BUFFSIZE, 0);
    fwrite(buf, 1, BUFFSIZE, file);
    sz -= cc;
}
fclose(file);
}
else{
    buf[cc] = '\0'; // ensure null-termination

    if (!strcmp(buf, QUIT)) break;
    printf("\r%s\n>", buf); // 显示所接收的字符串
}
}

return 0;
}
```

```
int main(int argv, char** argc)
{
    WSADATA wsadata;
    WSStartup(WSAVERS, &wsadata);
    SOCKET msock, ssock;
    msock = socket(PF_INET, SOCK_STREAM, IPPROTO_TCP);
    struct sockaddr_in sin;
    memset(&sin, 0, sizeof(sin));
    sin.sin_family = AF_INET;
    sin.sin_addr.s_addr = INADDR_ANY;
    sin.sin_port = htons((u_short)atoi(SERVICE));
    bind(msock, (struct sockaddr*) & sin, sizeof(sin));
    listen(msock, QUSIZE);
}
```



```
printf("服务器已启动! \n\n");
struct sockaddr_in fsin;
int hd_sz = sizeof(fsin);
ssock = accept(msock, (struct sockaddr*) &fsin, &hd_sz);
HANDLE thr = (HANDLE)_beginthreadex(NULL, 0, &rec, (void*)&ssock, 0, NULL);
char comm[MAX_FILE_NAME];

while (1)
{
    printf(">");
    scanf_s("%[^\n]s", comm, MAX_FILE_NAME);
    getchar();
    if (!strcmp(comm, RDIR, 4)) {
        strcpy(file_name_head, comm + 5);
    }
    else if (!strcmp(comm, CHAT, 4)) {
        send(ssock, comm+5, strlen(comm)-5, 0);
    }
    else if (!strcmp(comm, SEND, 4)) {
        send_file(ssock, comm + 5);
    }
    else if (!strcmp(comm, QUIT)) {
        send(ssock, comm, strlen(comm), 0);
        break;
    }
    else {
        send(ssock, comm, strlen(comm), 0);
    }
}

CloseHandle(thr);
closesocket(ssock);
closesocket(msock);
WSACleanup();

return 0;
}
```

// 关闭套接字
// 关闭监听套接字
// 卸载winsock library

客户端:

```
#pragma warning( disable : 4996)
#include <stdlib.h>
#include <stdio.h>
#include <winsock2.h>
#include <string.h>
#include <process.h>
```



```
#include <sys/stat.h>

#pragma comment(lib, "ws2_32.lib")

#define WSVERS MAKEWORD(2, 0)
#define SERVICE "50500"
#define BUFLen (10 << 24)
#define RDIR "rdir"
#define CHAT "chat"
#define SEND "send"
#define QUIT "quit"

#define MAX_FILE_NAME 200
#define MAX_FILE_TYPE 10
#define PDF "pdf"
#define TXT "txt"
#define JPG "jpg"
#define PNG "png"
#define PPTX "pptx"

#define PDF_Struct int
#define TXT_Struct int
#define JPG_Struct int
#define PNG_Struct int
#define PPTX_Struct int

typedef struct {
    char fileName[MAX_FILE_NAME];
    char fileType[MAX_FILE_TYPE];
}FileStruct;

char buf[BUFLen];
char file_name_head[MAX_FILE_NAME];
char data_buf[BUFLen];

void app_str(char* str, int num)
{
    strcpy(str, "(");
    char tem[10];
    itoa(num, tem, 10);
    strcat(str, tem);
    strcat(str, ")");
}

void add_str(char* str, char* front_app)
{
    int len = strlen(front_app);
    char tem[100] = {};
    strcpy(tem, front_app);
```



```
tem[len] = '\\';
tem[len + 1] = '\\0';
strcpy(tem + len + 1, str);
strcpy(str, tem);
}

void send_file(SOCKET ssock, char* file_name)
{
    int len = 0;
    FileStruct tem;
    strcpy(tem.fileName, file_name);
    char* pos = strchr(file_name, '.');
    strcpy(tem.fileType, pos + 1);
    memcpy(data_buf, &tem, sizeof(FileStruct));
    len += sizeof(FileStruct);
    add_str(file_name, file_name_head);
    long long sz = 0;
    if (!strcmp(tem.fileType, PDF) || !strcmp(tem.fileType, TXT)
        || !strcmp(tem.fileType, JPG) || !strcmp(tem.fileType, PNG)
        || !strcmp(tem.fileType, PPTX))
    {
        PDF_Struct size;
        struct stat statbuf;
        if (stat(file_name, &statbuf) != 0)
        {
            printf("File Size Error!\n");
            return;
        }
        size = statbuf.st_size;
        sz = size;
        memcpy(data_buf + len, &size, sizeof(PDF_Struct));
        len += sizeof(PDF_Struct);
    }
    else {
        printf("File Type Error!\n");
        return;
    }
    send(ssock, data_buf, len, 0);
    FILE* file = fopen(file_name, "rb");
    if (!file)
    {
        printf("FILE OPEN ERROR!\n");
        return;
    }
    while (sz > 0)
    {
```




```
        pos = strrchr(file_name, '(');
        if(!pos) pos = strchr(file_name, '.');
        strcpy(pos, app);
        strcat(file_name, ".");
        strcat(file_name, tem.fileType);
    }
    else {
        break;
    }
}
file = fopen(file_name, "wb");
if (!file)
{
    printf("FILE OPEN ERROR!\n");
    break;
}
while (sz>0)
{
    cc = recv(*sock, buf, BUFLen, 0);
    fwrite(buf, 1, cc, file);
    sz -= cc;
}
fclose(file);
}
else {
    buf[cc] = '\0'; // ensure null-termination

    if (!strcmp(buf, QUIT)) break;
    printf("\r%s\n", buf); // 显示所接收的字符串
}
}

return 0;
}
int main(int argc, char* argv[])
{
    char tem1[20] = "127.0.0.1";
    //char tem1[20] = "103.26.79.35";
    //char tem1[20] = "111.230.210.193";

    char* host = tem1;
    char tem2[6];
    strcpy(tem2, SERVICE);
    char* service = tem2;
    struct sockaddr_in sin;
    SOCKET sock;
```



```
WSADATA wsadata;
WSAStartup(WSAVERS, &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));
if(connect(sock, (struct sockaddr*) & sin, sizeof(sin))) printf("Connect Error!\n");
HANDLE thr = (HANDLE)_beginthreadex(NULL, 0, &rec, (void*)&sock, 0, NULL);
char comm[100];
while (1)
{
    printf(">");
    scanf_s("%[^\\n]s", comm, 100);
    getchar();
    if (!strncmp(comm, RDIR, 4)) {
        strcpy(file_name_head, comm + 5);
    }
    else if (!strncmp(comm, CHAT, 4)) {
        send(sock, comm + 5, strlen(comm) - 5, 0);
    }
    else if (!strncmp(comm, SEND, 4)) {
        send_file(sock, comm + 5);
    }
    else if (!strcmp(comm, QUIT)) {
        send(sock, comm, strlen(comm), 0);
        break;
    }
    else {
        send(sock, comm, strlen(comm), 0);
    }
}

CloseHandle(thr);
closesocket(sock);
WSACleanup();
return 0;
}
```

// 关闭监听套接字
// 卸载winsock library

(6) 服务器运行截屏:

```
服务器已启动!
>kon!
k on!
>rdir C:\Users\E480\Desktop\计网\实验\实验五\MY\P2PServer\file
>send ftp.pdf
>send ftp.pdf
>send kon.txt
>quit
C:\Users\E480\Desktop\计网\实验\实验五\MY\P2PServer\Debug\P2PServer.exe
```




> 实验五 > MY > P2PServer > file

名称

fate.jpg

ftp.pdf

kon.txt

(7) 客户端运行截屏:

```
kon!
>k on!
>dir C:\Users\E480\Desktop\计网\实验\实验五\MY\P2PClient\file
>send fate.jpg
>quit
C:\Users\E480\Desktop\计网\实验\实验五\MY\P2PClient\Debug\P2PClient.exe
```

> 实验五 > MY > P2PClient > file

名称

fate.jpg

ftp(2).pdf

ftp(3).pdf

ftp.pdf

kon.txt

经验证均能正常打开。

(8) 与同学互测 (选做)

【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: _____

【实验体会】

写出实验过程中遇到的问题, 解决方法和自己的思考; 并简述实验体会 (如果有的话)。

坑: 创建线程是时错误地使用了套接字的指针, 但在新的套接字被创建后, 原套接字可能被 kill 掉了, 无法创建数据连接。出现 Error: 10038。

解决办法: 直接传套接字本身创建线程。使用 `WaitForSingleObject(thr, INFINITE)`;。使用全局变量。

不足: 为符合老师对于数据包的设计, 该程序暂时仅能发送 pdf, jpg, png, txt, pptx 共 5 种文件, 并且他们的数据大小都用 int 来传输。但增加其它类型的传输也是简单的, 只要用宏规定该类型文件数据大小的类型, 并在文件类型检查处增加条件即可。

【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://103.26.79.35/netdisk/default.aspx?vm=18net> 实验上交/编程实验

(3) 截止日期 (不迟于): 2020 年 6 月 13 日 23:00 (周六)。

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