**华中科技大学**

**网络空间安全学院**

**《计算机通信与网络》课程设计报告**

姓 名 齐玮

班 级 密码2101

学 号 U202112150

联系方式 18602712617

分 数

**实验报告及代码和设计评分细则**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 评 分 项 目 | | | 满分 | 得分 | 备注 |
| 文档格式（段落、行间距、缩进、图表、编号等） | | | 10 |  |  |
| 感想（含思政） | | | 10 |  |  |
| 意见和建议 | | | 10 |  |  |
| 验收时间 | | | 10 |  |  |
| Socket编程 | 代码可读性 | | 10 |  |  |
| 注释 | | 10 |  |
| 软件体系结构 | | 30 |  |
| 问题描述及解决方案 | | 10 |  |
| 实验报告总分 | | | 100 |  |  |
| 教师签名 | |  | | 日 期 |  |

目录

**[《计算机通信与网络》课程设计报告 I](#_Toc8050)**

[一、 实验概述 1](#_Toc17103)

[1.1 实验名称 1](#_Toc30814)

[1.2 实验目的 1](#_Toc3629)

[1.3 实验环境 1](#_Toc10493)

[1.4 实验内容 1](#_Toc1226)

[1.5 实验要求 1](#_Toc27097)

[二、 实验过程 1](#_Toc21422)

[2.1 系统结构设计 1](#_Toc21762)

[2.2 详细设计 3](#_Toc32727)

[2.3 代码实现 6](#_Toc23796)

[三、 实验测试与分析 10](#_Toc15214)

[3.1 系统测试及结果说明 10](#_Toc24250)

[3.2 遇到的问题及解决方法 15](#_Toc20428)

[3.3 设计方案存在的不足 16](#_Toc21992)

[四、实验总结 17](#_Toc27358)

[4.1 实验感想 17](#_Toc20639)

[4.2 意见和建议 17](#_Toc18159)

**Socket编程实验**

## **实验概述**

## **实验名称**

Socket编程实验。

## **实验目的**

通过socket程序的编写、调试，了解计算机网络可靠传输协议，熟悉基于UDP协议的socket编程方法，掌握如何开发基于TCP/UDP的网络应用。

## **实验环境**

操作系统：Windows/Linux

编程语言：C，C++

## **实验内容**

完成一个TFTP协议客户端程序，实现一下要求：

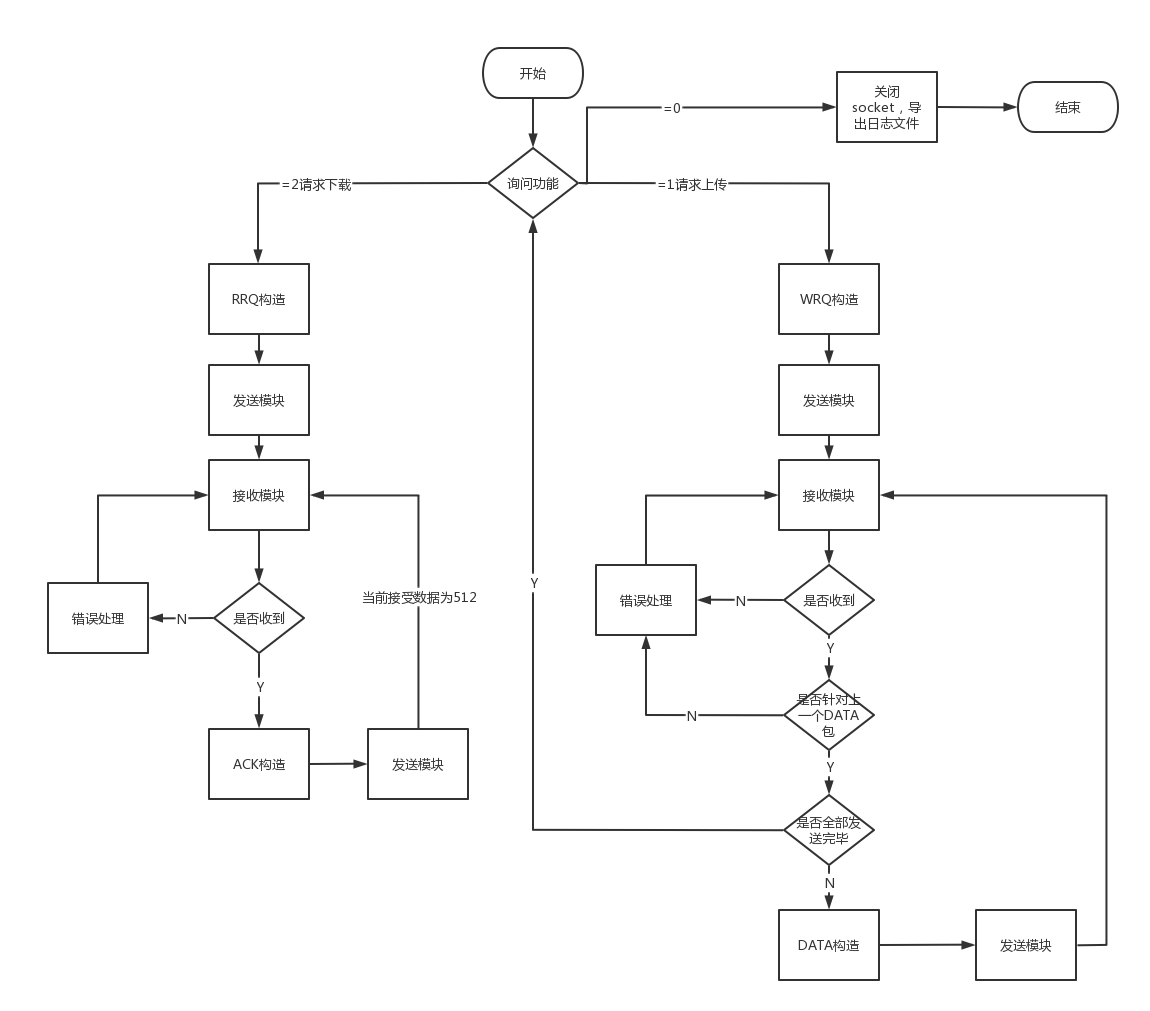
1. 严格按照TFTP协议与标准TFTP服务器通信；
2. 能够实现两种不同的传输模式netascii和octet；
3. 能够将文件上传到TFTP服务器；
4. 能够从TFTP服务器下载指定文件；
5. 能够向用户展现文件操作的结果：文件传输成功/传输失败；
6. 针对传输失败的文件，能够提示失败的具体原因；
7. 能够显示文件上传与下载的吞吐量；
8. 能够记录日志，对于用户操作、传输成功，传输失败，超时重传等行为记录日志；
9. 人机交互友好（图形界面/命令行界面均可）；
10. 额外功能的实现，将视具体情况予以一定加分。

## **实验要求**

1. 必须基于Socket编程，不能直接借用任何现成的组件、封装的库等；
2. 提交实验设计报告和源代码；实验设计报告必须包括程序流程图，源代码必须加详细注释。
3. 实验设计报告需提交纸质档和电子档，源代码、编译说明需提交电子档。
4. 基于自己的实验设计报告，通过实验课的上机试验，将源代码编译成功，运行演示给实验指导教师检查。

## **实验过程**

1. **系统结构设计**
2. **模块框图**

****

1. **模块功能说明**

RRQ构造：构造一个RRQ数据包

发送模块：客户端将构造的数据包发送给TFTP服务器端

接收模块：接受来自TFTP客户端的数据包

错误处理：当接收到的数据包出错或者未收到来自服务器的数据包时，采取相应的措施

ACK构造：构造一个ACK数据包

WRQ构造：构造一个WRQ数据包

DATA构造：构造一个DATA数据包

1. **模块之间的接口说明**

1）RRQ构造：

接口函数为char\* RequestDownloadPack(char\* content,int\* datalen,int type)

参数：content为存放需要下载的文件的文件名的字符串指针

datalen为文件名长度的指针

type为传输方式

功能：返回指向构造出的RRQ包的指针

2）WRQ构造：

接口函数为char\* RequestUpdownloadPack(char\* content,int\* datalen,int type)

参数：content为存放需要上传的文件的文件名的字符串指针

datalen为文件名长度的指针

type为传输方式

功能：返回指向构造出的WRQ包的指针

3）ACK构造:

接口函数为char\* AckPack(short\* no)

参数：no为期待收到的下一个数据包编号的指针

功能：返回指向构造出的ACK包的指针

4）DATA构造：

接口函数为char\* MakeData(short\* n,FILE\* f,int\* datalen)

参数：n为DATA数据包的编号

f为指向需要上传的文件的指针

datalen为数据包长度

功能：返回指向构造出的DATA包的指针

5）发送模块：

接口函数为int sendto (SOCKET s, char \* buf, int len ,int flags, struct sockaddr\_in \* to, int tolen);

参数：s

[in] 一个SOCKET(可能已连接)。

buf

[in] 指向要传输的数据的缓冲区的指针。

len

[in] buf的长度。

flags

[in] 指定函数调用的方式。一般取0。

to

[in] 指向目标地址结构体的指针。

tolen

[in] 目标地址结构体的长度。

功能：函数执行成功返回发送的字节数，失败则返回SOCKET\_ERROR

6）接收模块：

接口函数为int recvfrom (SOCKET s, char \* buf, int len ,int flags, struct sockaddr\_in \* from, int \* fromlen);

参数：s

[in] 一个已经绑定的SOCKET。

buf

[out] 指向接收数据的缓冲区的指针。

len

[in] buf的长度。

flags

[in] 指定函数调用的方式。一般取0。

from

[out] 指向源地址结构体的指针。

fromlen

[in/out] 源地址结构体的长度。

功能：函数执行成功返回发送的字节数（可能小于len），失败则返回SOCKET\_ERROR

7）错误处理：

当出现超时时，调用sendto函数重传

当其他错误时，不调用函数，而是显示错误类型

1. **详细设计**

1.请求上传模块详细设计伪代码

char\* sendData = RequestUpdownloadPack();//构造WRQ包

int res = sendto();//将WRQ包发送给TFTP服务器

while(发送失败)

{

    if(Killtime <= 10)//失败次数<10

    {

        重传;

        Killtime++;

    }

    else break;

    if(Killtime > 10)   放弃;

}

FILE\* f 指向需要上传的文件;

while(1)

{

    res = recvfrom();*//监听来自服务器的数据包*

    if(没有收到数据)

    {

        重传上一个数据包;

        if(超时)

        {

            重传数据包,但是最多重传10次;10次仍超时,则放弃;

        }

        如果重传10次仍没有收到回应,则放弃

    }

    if(收到了服务器的回应数据包)

    {

        获取数据包的类型编号flag;

        if(ACK包)

        {

            if(文件上传完毕)

            {

                计算传输时间,编辑日志文件;

                退出循环;

            }

            否则

            {

                从f中向后取出512字节的数据来制作下一个DATA包

                更新缓冲区中的内容为当前DATA包

                发送DATA包

            }

        }

        if(错误包)

        {

            显示错误并退出;

        }

    }

}

1. 请求下载模块详细设计伪代码

**char**\* sendData = RequestDownloadPack();*//构造RRQ包*

**int** res = sendto();*//将RRQ包发送给TFTP服务器*

while(发送失败)

{

    if(Killtime <= 10)*//失败次数<10*

    {

        重传;

        Killtime++;

    }

    else break;

    if(Killtime > 10)   放弃;

}

FILE\* f 指向需要上传的文件;

while(1)

{

    获取分配的端口号server;

    res = recvfrom();*//监听来自服务器的数据包*

    if(没有收到数据)

    {

        if(超时)

        {

            重传数据包,但是最多重传10次;10次仍超时,则放弃;

        }

        重传上一个数据包;

        如果重传10次仍没有收到回应,则放弃

    }

    if(收到了服务器的回应数据包)

    {

        获取数据包的类型编号flag;

        if(DATA包)

        {

            获取该DATA包的编号;

            制作ack = Ackpack(no) 包;

            发送ACK包sendlen = sendto(ack);

            if(sendto != 4)  ack包sendto出错,重传;

            if(no == want\_recv) 更新缓冲区;

            if(当前接受的数据包长度<512) 传输结束,退出;

            want\_recv++;

        }

        if(错误包)

        {

            显示错误并退出;

        }

    }

}

**2.3 代码实现**

以下为上传文件模块代码实现

**int** datalen;

**char**\* sendData = RequestUpdownloadPack(name, &datalen, type);

            buflen = datalen;

            NumbertoKill = 1; *//numbertokill变量表示一个数据包recv\_from超时的次数*

            memcpy(commonbuf, sendData, datalen);

*//commonbuf是一个专门用于数据重传的缓冲区，有可能被重传的数据都会统一的放进commonbuf中*

*//重传机制会直接从commonbuf中获得数据*

*//第一次发送WRQ包*

**int** res = sendto(socket, sendData, datalen, 0, (sockaddr\*)&addr, sizeof(addr));

            start = clock();*//开始计时*

            print\_time(fp);

            fprintf(fp, "send WRQ for file:%s\n", name);

            Killtime = 1; *//表示sendto的超时的次数，与recv\_from超时分开来计算*

            while (res != datalen) *//如果sendto函数失败了，则立即重新sendto，在成功或者到达上限次数之前不会进行其他操作*

            {

                printf("send WRQ failed %d times\n",Killtime);

                if (Killtime <= 10) *//10次为上限*

                {

                    res = sendto(socket, sendData, datalen, 0, (sockaddr\*)&addr, sizeof(addr));

                    Killtime++;

                }

                else

                    break;

            }

            if (Killtime > 10)

                continue;

            free(sendData);

            FILE\* f = fopen(name, "rb"); *//f指向目标文件*

            if (f == NULL)

            {

                printf("File %s open failed!\n", name);

                continue;

            }

*//开始传输文件*

**short** block = 0;*//文件编号*

            datalen = 0;

**int** RST = 0; *//记录重传次数*

**int** Fullsize = 0; *//记录文件的总大小*

            while (1) *//开始传输*

            {

**char** buf[1024];

                sockaddr\_in server; *//从server反馈的数据包中活得分配的端口号*

**int** len = sizeof(server);

                res = recvfrom(socket, buf, 1024, 0, (sockaddr\*)&server, &len); *//监听服务器的数据包*

                if (res == -1) *//如果没有收到数据*

                {

                    printf("%d ", NumbertoKill);

                    if (NumbertoKill > 10) *//如果连续10次没有收到回应*

                    {

                        printf("No acks get. trainsmission failed\n");

                        print\_time(fp);

                        fprintf(fp, "Upload file: %s failed.\n", name);

                        break;

                    }

**int** res = sendto(socket, commonbuf, buflen, 0, (sockaddr\*)&addr, sizeof(addr));

*//重传上一个数据包*

                    RST++;

                    printf("resend last blk\n");

                    Killtime = 1; *//同上处理sendto超时的情况*

                    while (res != buflen)

                    {

                        printf("Resend last blk failed %d times\n",Killtime);

                        if (Killtime <= 10)

                        {

                            res = sendto(socket, commonbuf, buflen, 0, (sockaddr\*)&addr, sizeof(addr));

                            Killtime++;

                        }

                        else

                            break;

                    }

                    if (Killtime > 10)

                        break;

                    NumbertoKill++;

                }

                if (res > 0) *//收到服务器的回应数据包*

                {

**short** flag;

                    memcpy(&flag, buf, 2);*//获取数据包的类型编号*

                    flag = ntohs(flag);

                    if (flag == 4)*//收到的是ACK包*

                    {

**short** no;

                        memcpy(&no, buf + 2, 2);

                        no = ntohs(no);

                        if (no == block)

                        {

                            addr = server;

                            if (feof(f) && datalen != 516)*//如果上传文件已经全部上传完毕*

                            {

                                printf("upload finished!");

                                end = clock();

                                runtime = (**double**)(end - start) / CLOCKS\_PER\_SEC;

*//计算传输时间*

                                print\_time(fp);*//写入日志文件*

                                printf("Average trainsmission rate: %.2lf kb/s\n", Fullsize / runtime / 1000);

                                fprintf(fp, "Upload file: %s finished.resent times:%d.Fullsize:%d\n", name, RST, Fullsize);

                                break;

                            }

                            block++; *//否则，制作下一个DATA包*

                            sendData = MakeData(&block, f, &datalen);

                            buflen = datalen;

                            Fullsize += datalen - 4;

*//fullsize要去除数据包中头部的长度*

                            NumbertoKill = 1;*//重置当前数据包的重发次数*

                            memcpy(commonbuf, sendData, datalen);*//更新commonbuf中的内容，准备下一次可能的重传*

                            if (sendData == NULL)

                            {

                                printf("File reading mistakes!\n");

                                break;

                            }

**int** res = sendto(socket, sendData, datalen, 0, (sockaddr\*)&addr, sizeof(addr));

                            Killtime = 1;

                            while (res != datalen)

                            {

                                printf("send block %d failed\n", block);

                                if (Killtime <= 10)

                                {

                                    res = sendto(socket, commonbuf, buflen, 0, (sockaddr\*)&addr, sizeof(addr));

                                    Killtime++;

                                }

                                else

                                    break;

                            }

                            if (Killtime > 10)

                                continue;

                            printf("Pack No = %d\n", block);

                        }

                    }

                    if (flag == 5)*//处理错误包*

                    {

**short** errorcode;

                        memcpy(&errorcode, buf + 2, 2);

                        errorcode = ntohs(errorcode);

**char** strError[1024];*//继续拆解并获得错误详细信息*

**int** iter = 0;

                        while (\*(buf + iter + 4) != 0)

                        {

                            memcpy(strError + iter, buf + iter + 4, 1);

                            ++iter;

                        }

                        \*(strError + iter + 1) = '\0';

                        printf("Error %d %s\n", errorcode, strError);*//输出错误类型*

                        print\_time(fp);

                        fprintf(fp, "Error %d %s", errorcode, strError);

                        break;

                    }

                }

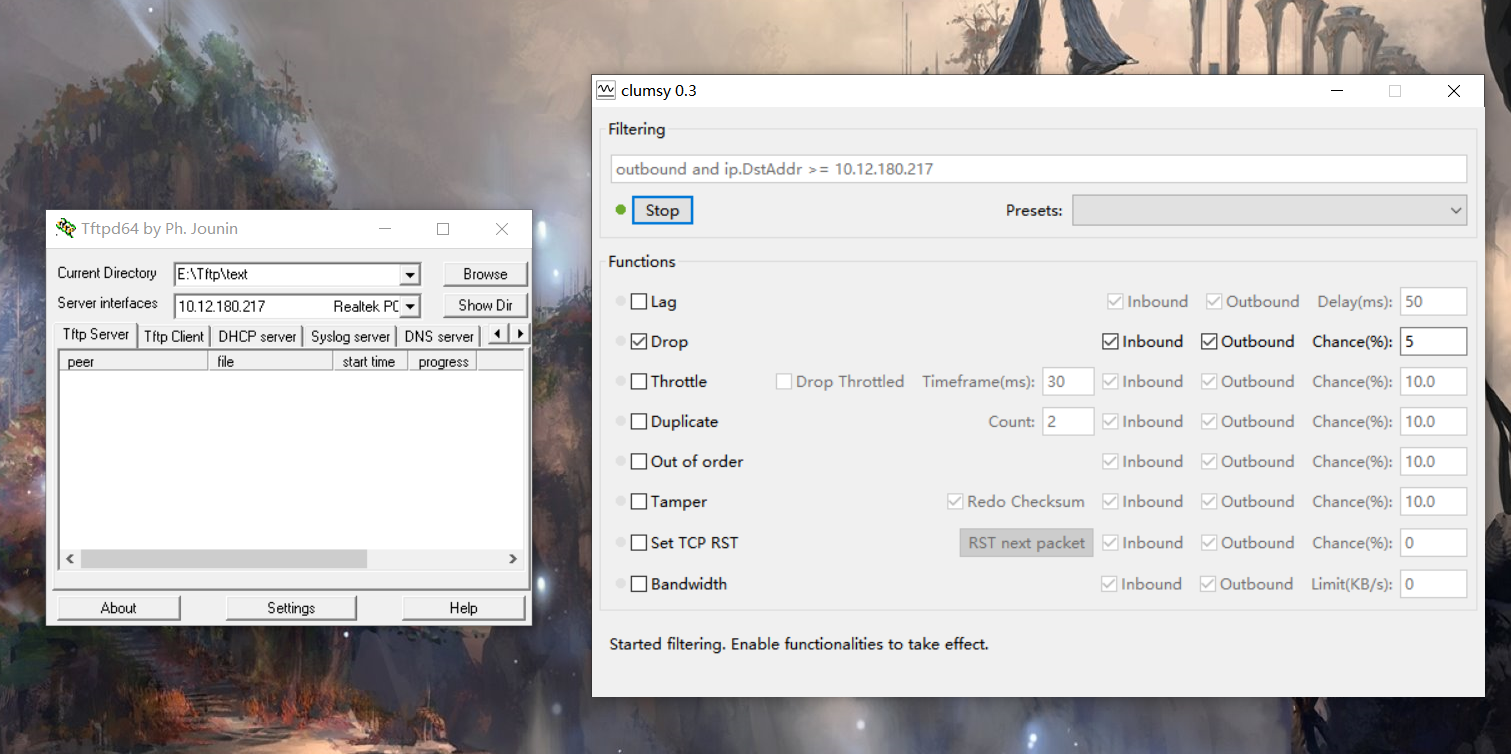
            }

            fclose(f);*//关闭文件*

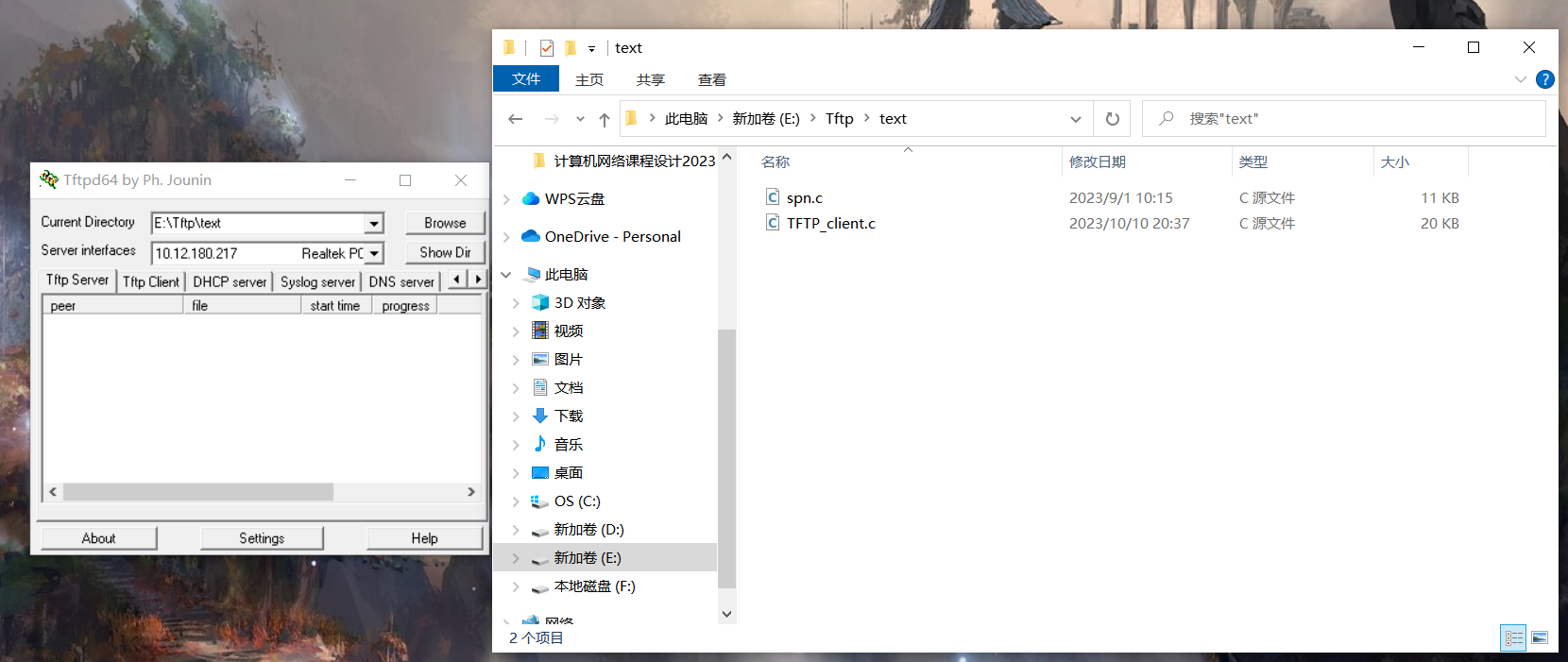
## **实验测试与分析**

## **系统测试及结果说明**

**使用tftpd64模拟TFTP服务器，使用clumsy模拟弱网。**

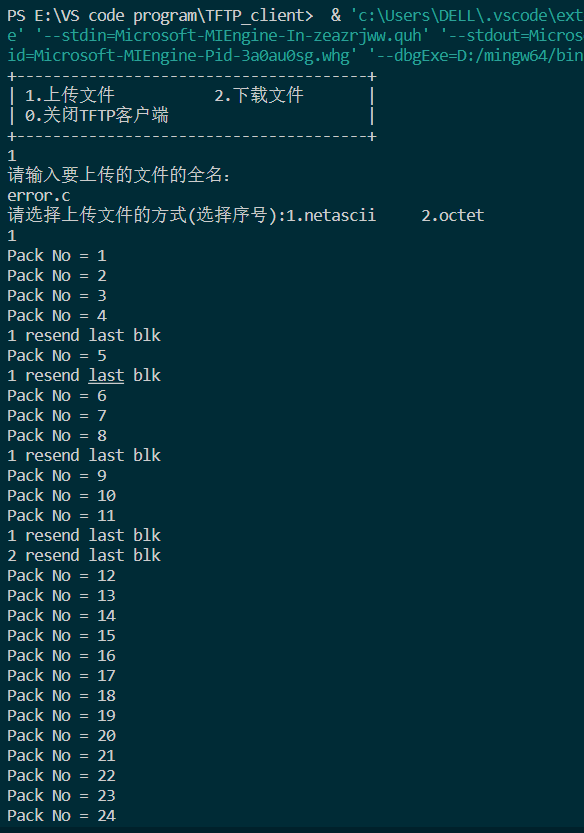


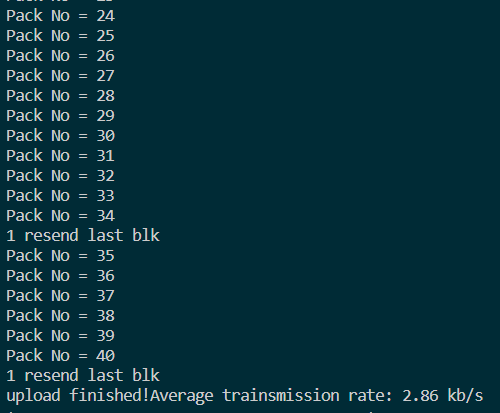
Tftpd64以及clumsy设置



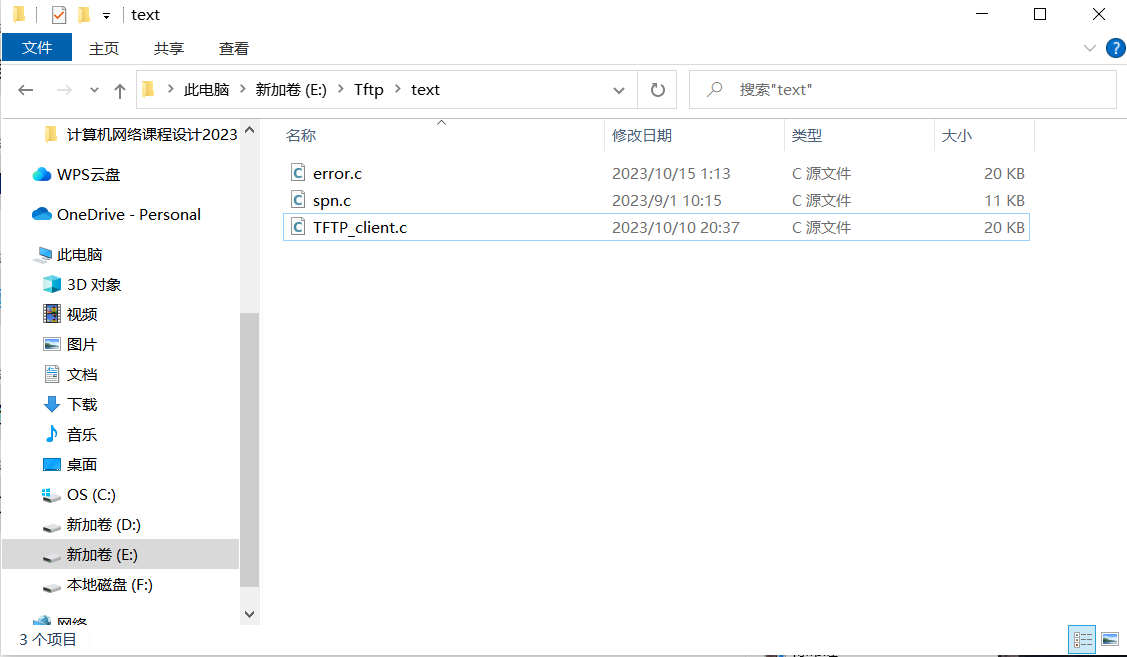
Tftpd64根目录

**1.运行源程序测试上传文件error.c**





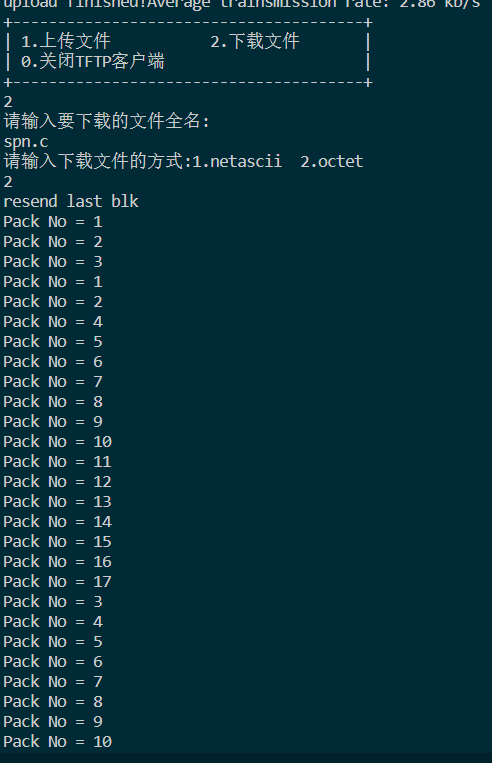
源程序运行上传文件结果

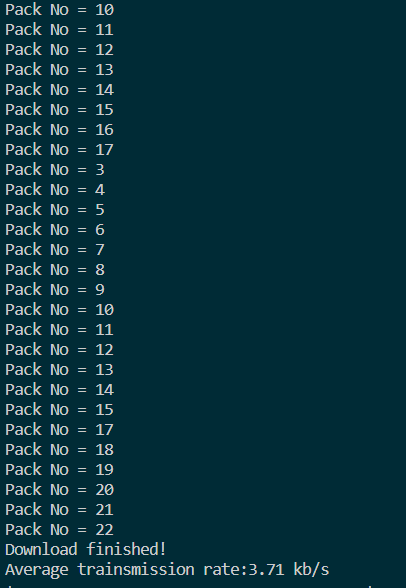


上传后的tftp根目录

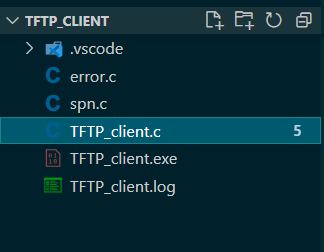
根据运行截图，文件error.c文件成功上传到了TFTP服务器中，且丢包重传功能正常，上传完成后显示了传输速率为2.86kb/s，测试成功！

1. **运行源程序测试下载文件spn.c**



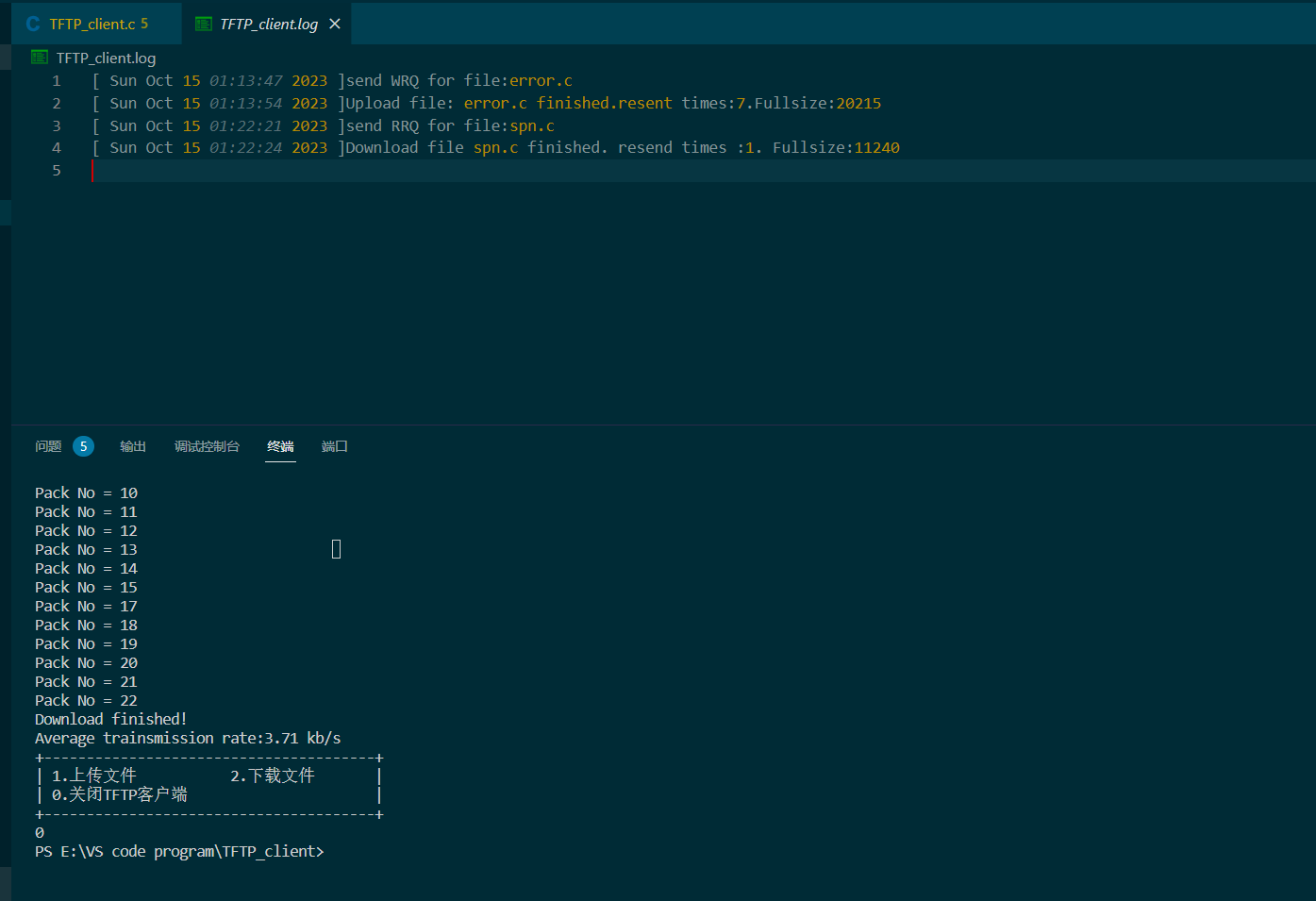


源程序运行下载文件结果



下载文件后源程序目录

根据测试截图，进行下载操作后，spn.c文件成功下载到了源程序目录中，下载完成后显示传输速率为3.71kb/s，且出现了超时重传情况，测试成功！



关闭客户端

关闭客户端后生成日志文件TFTP\_client.log，记录的操作以及操作时间均正确。

## **遇到的问题及解决方法**

1. **VScode配置**

在VScode中进行socket编程需要修改tasks.json文件，仅仅在源程序中#include是不够的，需要添加ws2\_32.lab库，而且winsock2.h需要包含在windows.h之前。

1. **头文件winsock2.h**

头文件winsock2.h中有很多方便的函数可以直接调用，但需要一定时间去熟悉各类函数的用法，我印象比较深的是htons()和ntohs()两个函数，当时看代码视频的时候不知道这个函数的功能，之后上网查了资料才理解网络字节序和主机字节序的区别。另外，在进行socket编程时，需要用到许多函数进行初始化和建立连接，如WSAstartup函数初始化网络环境。

1. **结构体sockaddr\_in**

结构体sockaddr\_in困扰了我一段时间，因为实在不理解它的含义，以至于getaddr函数是照着示例代码写的，之后调用的时候心里也没什么底气，重复测试了许多次。

1. **数据包的构造**

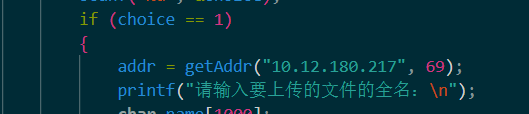
这是我第一次按照协议格式构造数据包，最初感觉很别扭，对字符串不够敏感，第一次写的时候不知道怎么设置操作码，DATA数据包不知道怎么放入数据。参照示例代码看了RRQ包的构造后，其余数据包的构造也就相对简单了，重点在与用好memcpy函数和注意每一部分的字长。

1. **错误处理**

因为每次传输一个数据包都要监视回应包，也已位置没传一个数据包过去都有可能出发错误处理，所以要设置一个缓冲区，保存当前传输的数据包，以备超时或丢包情况出现后，可以进行重传。刚开始写重传的时候不清楚如何判断丢包或者超时，看了示例代码后发现它用了两个函数sendto()和recvfrom()，了解过后明白了可以通过他们的返回值判断是否出错。其次是对ACK包和DATA包的操作，需要从数据包中取出有用的信息，这些细节的地方正是我需要强化注意的。

## **设计方案存在的不足**

需要手动修改IP



addr = getAddr("10.12.180.217", 69);

## 有两行中的IP 需要手动修改为TFTP服务器的IP。

日志文件需要关闭TFTP客户端才能生成。

没办法保证每个文件都能上传/下载，因为重传此处超过10次后程序会直接放弃上传/下载

## **四**、**实验总结**

## **4.1 实验感想**

做这个socket编程的时候我是很痛苦的，因为第一次接触按着协议规定的格式进行构造数据包，没有经验，对字符串的各种操作不熟练，因此导致构造数据包时心里没有底，而且对相关函数了解很少，对丢包，超时等情况的判断以及解决方法都只停留在理论阶段，从没有实践过，等到了要写程序去实现的时候才发现无从下手，不过也很庆幸能通过这次实验暴露自己的这些问题。

通过本次实验，我对网络协议的了解更深刻了，也对socket的工作模式有了更为直观的了解，学会了构造数据包的方法，模拟了客户-服务器的工作模式，虽然很痛苦，但收获很大！

## **4.2 意见和建议**

|  |
| --- |
| 原创性声明 |
| 本人郑重声明本报告内容，是由作者本人独立完成的。有关观点、方法、数据和文献等的引用已在文中指出。除文中已注明引用的内容外，本报告不包含任何其他个人或集体已经公开发表的作品成果，不存在剽窃、抄袭行为。  已阅读并同意以下内容。  判定为不合格的一些情形：  （1） 请人代做或冒名顶替者；  （2） 替人做且不听劝告者；  （3） 实验报告内容抄袭或雷同者；  （4） 实验报告内容与实际实验内容不一致者；  （5） 实验代码抄袭者。  **作者签名：** |