|  |
| --- |
| 上海海事大学 |
| 组网与网络设计报告 |
|  |
|  |

题目：用socket编程实现网上聊天和大文件传输

姓 名 徐艺芸 芸

学 号 201910121213 3

指导教师 高军 啊

班 级 网络191 啊

学 院 信息工程学院 1

完成时间 2021年12月14日 星

目录

[摘要 4](#_Toc90333385)

[Abstract 6](#_Toc90333386)

[1.概述 8](#_Toc90333387)

[1.1 课题的背景和研究意义 8](#_Toc90333388)

[1.2 国内外研究现状 9](#_Toc90333389)

[1.3 项目实施过程与进度说明 10](#_Toc90333390)

[2.项目实现基本原理 11](#_Toc90333391)

[2.2实现思想概述 14](#_Toc90333392)

[2.2.1系统处理信息的实现 14](#_Toc90333393)

[2.2.2对文件进行拆包传输的实现 15](#_Toc90333394)

[2.2.3 聊天功能实现的具体流程 17](#_Toc90333395)

[2.2.4 聊天中双方实现并发通信的实现 17](#_Toc90333396)

[3.开发环境及所用技术简介 21](#_Toc90333397)

[3.1开发环境 21](#_Toc90333398)

[3.2所用技术简介 22](#_Toc90333399)

[3.3 socket相关背景知识简介 22](#_Toc90333400)

[3.3.1、socket的分类 23](#_Toc90333401)

[3.3.2、Socket通信工作原理图 23](#_Toc90333402)

[3.4、相关库函数介绍 24](#_Toc90333403)

[3.4.1、socket函数 24](#_Toc90333404)

[3.4.2、gethostbyname函数 25](#_Toc90333405)

[3.4.4、bind函数 26](#_Toc90333406)

[3.4.5、listen函数 27](#_Toc90333407)

[3.4.6、accept函数 27](#_Toc90333408)

[3.4.7、send函数 28](#_Toc90333409)

[3.4.8、recv函数 29](#_Toc90333410)

[3.4.9、socket文件描述符 29](#_Toc90333411)

[3.4.10、服务端程序绑定地址 29](#_Toc90333412)

[3.4.11、服务端程序绑定的通信端口 30](#_Toc90333413)

[3.4.12、客户端程序指定服务端的ip地址 30](#_Toc90333414)

[3.4.13、客户端程序指定服务端的通信端口 30](#_Toc90333415)

[3.4.14、服务端有两个socket 30](#_Toc90333416)

[4.关键代码展示： 31](#_Toc90333417)

[（1）客户端连接服务器代码 31](#_Toc90333418)

[（2）客户端选择功能代码段（界面显示） 32](#_Toc90333419)

[（3）服务器创建套接字流程： 33](#_Toc90333420)

[（4）服务端处理信息代码段： 34](#_Toc90333421)

[（5）客户端聊天室代码： 36](#_Toc90333422)

[（6）客户端接受文件代码（如何组装包） 36](#_Toc90333423)

[（7）服务端发送文件代码： 37](#_Toc90333424)

[(8)发送成功后选择 38](#_Toc90333425)

[5.功能展示： 40](#_Toc90333426)

[6、结论与展望 45](#_Toc90333427)

[6.1结论 45](#_Toc90333428)

[6.2 展望 46](#_Toc90333429)

[参考文献 47](#_Toc90333430)

# 摘要

随着互联网的高速发展，网上聊天和通过网络进行文件传输功能也逐步发展完善，两者的应用极大的方便了人们的生活，通过网上聊天，人们可以足不出户的与信任的人进行通信和交流，减少了面对面交流所消耗的时间、空间的成本，通过网络进行文件传输可以让文件被人们及时的得到利用，实现资源共享，这两个方面的应用和发展极大的改善了人们的生活，基于这两方面的应用也可以延申出很多功能的实现：比如电子邮件的发明，聊天系统软件的实现，网络资源存储和共享等等，掌握好两个技术有利于我们更好的理解计算机网络的组成原理，更好的帮助我们理解和学习计算机网络前沿技术，因此笔者将利用现学的socket编程方法，用C语言在windows系统下实现简单的网上聊天功能和文件传输功能，对现学的计算机网络知识进行巩固加深其影响，也可以用来其交流和学习。

本项目利用已有的计算机资源，开发出一种能有效进行网上聊天兼文件传输的软件，分为客户端和服务器端，主要完成了如下工作：

1. 服务器端开启，对客户端进行监听，接受到连接请求并建立连接；
2. 客户端打开，自动连接上预设的服务器端，双方都打开时，两者就可以开始通信；
3. 客户端界面打印功能选项，客户端可通过命令选择要进行的功能,如键“a”进入聊天室，键“b”进行文件传输，键入“c”退出聊天室；
4. 服务端根据客户端发送的信息判断进行发送信息还是传送文件。
5. 服务端在聊天过程中可以输入“close”命令主动关闭于客户端的连接。
6. 客户端在聊天过程中可以输入“close”命令关闭与客户端的连接；输入“sendfile”命令请求发送文件。
7. 服务端成功发送文件后，客户端可选择继续聊天。
8. 客户端可向服务端申请发送大于100M以上的大文件。
9. 双方的通信支持并发功能，即单方可以同时发送多条数据或者接受多条数据而不发生堵塞。

关键词：socket编程，网上聊天，多线程并发聊天，大文件传输，C语言，Windows开发环境

# Abstract

With the rapid development of the Internet, the functions of online chat and file transmission through the network have gradually developed and improved. Their applications have greatly facilitated people's life. Through online chat, people can communicate and communicate with trusted people at home, reducing the cost of time and space consumed by face-to-face communication, File transmission through the network can make files used by people in time and realize resource sharing. The application and development of these two aspects have greatly improved people's life. The application based on these two aspects can also extend the realization of many functions: such as the invention of e-mail, the realization of chat system software, network resource storage and sharing, etc, Mastering the two technologies will help us better understand the composition principle of computer network and help us better understand and learn the cutting-edge technology of computer network. Therefore, the author will use the current socket programming method to realize the simple online chat function and file transmission function in C language under Windows system, consolidate the current computer network knowledge and deepen its influence, It can also be used for communication and learning.

The project uses the existing computer resources to develop a software that can effectively carry out online chat and file transmission, which is divided into client and server. The main work is as follows:

（1） The server is turned on to listen to the client, receive the connection request and establish a connection.

（2） When the client is opened, it will automatically connect to the preset server. When both sides are opened, the two can start communication

（3） Print function option on the client interface. The client can select the function to be performed through commands, such as "a" to enter the chat room and "B" to transfer files.

（4） The server determines whether to send information or transfer files according to the information sent by the client.

（5） During the chat, the server can enter the "close" command to actively close the connection to the client.

（6） During the chat, the client can enter the "close" command to close the connection with the client; Enter the "sendfile" command to terminate the conversation and request to send the file.

（7） After the server successfully sends the file, the client can choose to continue chatting.

（8） The client can apply to the server for sending large files larger than 100m.

（9）The communication between the two sides supports the concurrent function, that is, one party can send multiple data or accept multiple data at the same time without blocking.

【Key words】: socket programming, online chat, multithreaded concurrent chat, large file transfer, C language, windows development environment

# 1.概述

## 1.1 课题的背景和研究意义

 Socket是应用层与TCP/IP协议族通信的中间软件抽象层，它是一组接口。在设计模式中，Socket其实就是一个门面模式，它把复杂的TCP/IP协议族隐藏在Socket接口后面，对用户来说，一组简单的接口就是全部，让Socket去组织数据，以符合指定的协议。

本次课程设计的研究方向是便是利用socket编程技术，实现在互联网上两个计算机之间的通信，实现大文件传送功能，兼客户端和服务端相互之间的并发通信聊天的功能，研究这一方面的意义是如果项目开发成功，用户可以充分利用计算机网络的优势，优化解决在互联网上客户端和服务端同时实现实时并发进行聊天以及大文件传送问题，也排除了因为借助第三方软件实现其功能所带来的安全隐患问题，因为存在若双方之间的通信借助第三方软件，信息很容易也被第三方窃取和攻击篡改，因此独立设计研发一款用于发送文件兼聊天功能的软件极为重要。基于上述问题和设计初衷，笔者就打算结合所学的计算机网络的相关知识以及和C语言基础独立开发一款实时聊天和大文件传送软件的软件。

设计这款软件的难点在于：

1. 如何实现文件传输。
2. 如何保证文件传输过程中不会出错。
3. 如何解决大文件传输慢的问题，超大文件传输不会出错的问题。
4. 如何将聊天功能和文件传输功能结合起来，当运行程序时，双方功能不会相互干扰和影响，系统稳定。
5. 如何设计出界面美观，用户体验良好的界面。
6. 如何实现客户端和服务端双方并发通信问题。

## 1.2 国内外研究现状

1. 国内研究现状

目前，随着互联网的发展，网络越来越影响着我们生活的方方面面，介于这样的发展趋势，国内很多互联网公司成立起来，网络编程这项技术发展迅速，目前在我国对于实现文件传输功能兼聊天功能这一方面做得比较好并具有代表性的主要是腾讯旗下的QQ和微信。例如qq实现的功能非常丰富，他支持在线聊天、[视频聊天](https://baike.baidu.com/item/%E8%A7%86%E9%A2%91%E8%81%8A%E5%A4%A9/2451484)以及[语音聊天](https://baike.baidu.com/item/%E8%AF%AD%E9%9F%B3%E8%81%8A%E5%A4%A9)、点对点[断点续传](https://baike.baidu.com/item/%E6%96%AD%E7%82%B9%E7%BB%AD%E4%BC%A0)文件、[共享文件](https://baike.baidu.com/item/%E5%85%B1%E4%BA%AB%E6%96%87%E4%BB%B6/3638070)、[网络硬盘](https://baike.baidu.com/item/%E7%BD%91%E7%BB%9C%E7%A1%AC%E7%9B%98)、自定义面板、[远程控制](https://baike.baidu.com/item/%E8%BF%9C%E7%A8%8B%E6%8E%A7%E5%88%B6)、[QQ邮箱](https://baike.baidu.com/item/QQ%E9%82%AE%E7%AE%B1)、传送[离线文件](https://baike.baidu.com/item/%E7%A6%BB%E7%BA%BF%E6%96%87%E4%BB%B6)等多种功能，并可与多种通讯方式相连。

此外QQ还具有与手机聊天、视频通话、语音通话、点对点[断点续传](https://baike.baidu.com/item/%E6%96%AD%E7%82%B9%E7%BB%AD%E4%BC%A0/764723)传输文件、传送离线文件、共享文件、QQ邮箱、网络收藏夹、发送贺卡等，储存文件等功能。QQ不仅仅是简单的[即时通信软件](https://baike.baidu.com/item/%E5%8D%B3%E6%97%B6%E9%80%9A%E4%BF%A1%E8%BD%AF%E4%BB%B6)，它与全国多家寻呼台、移动通信公司合作，实现传统的[无线寻呼网](https://baike.baidu.com/item/%E6%97%A0%E7%BA%BF%E5%AF%BB%E5%91%BC%E7%BD%91/22498405)、GSM移动电话的短消息互联，是国内最为流行功能最强的即时通信（IM）软件。腾讯QQ支持在线聊天、即时传送视频、语音和文件等多种多样的功能。同时，QQ还可以与移动通讯终端、IP电话网、无线寻呼等多种通讯方式相连，使QQ不仅仅是单纯意义的网络虚拟呼机，而是一种方便、实用、超高效的即时通信工具。QQ可能是在中国被使用次数最多的通讯工具。QQ状态分为不在线，离线，忙碌，请勿打扰，离开，隐身，在线，**Q**我吧。还可以自定义QQ状态。

微信和qq面向的用户不同，但其功能非常相似，它支持跨通信[运营商](https://baike.baidu.com/item/%E8%BF%90%E8%90%A5%E5%95%86/4530550)、跨[操作系统](https://baike.baidu.com/item/%E6%93%8D%E4%BD%9C%E7%B3%BB%E7%BB%9F/192)平台通过网络快速发送免费（需消耗少量网络流量）[语音短信](https://baike.baidu.com/item/%E8%AF%AD%E9%9F%B3%E7%9F%AD%E4%BF%A1/8430295)、[视频](https://baike.baidu.com/item/%E8%A7%86%E9%A2%91/321962)、[图片](https://baike.baidu.com/item/%E5%9B%BE%E7%89%87/372416)和文字，同时，也可以使用通过共享[流媒体](https://baike.baidu.com/item/%E6%B5%81%E5%AA%92%E4%BD%93/98740)内容的资料和基于位置的社交插件“[摇一摇](https://baike.baidu.com/item/%E6%91%87%E4%B8%80%E6%91%87/14079911)”、“[朋友圈](https://baike.baidu.com/item/%E6%9C%8B%E5%8F%8B%E5%9C%88/17341022)”、”[公众平台](https://baike.baidu.com/item/%E5%85%AC%E4%BC%97%E5%B9%B3%E5%8F%B0/3274590)“、”[语音记事本](https://baike.baidu.com/item/%E8%AF%AD%E9%9F%B3%E8%AE%B0%E4%BA%8B%E6%9C%AC)“等服务插件。

1. 国外研究现状

目前，国外的社交媒体软件做的也非常出色，不同于我国的社交软件的设计，国外软件实现的功能都比较单一但独具特色。在国外聊天和文件传输功能这一方面做得比较好的有WhatsApp、Line、Skype、Signal、BatChat等，使用最多也最为典范的是WhatsApp，它发布于2009年智能手机刚刚起步的时候，此后被Facebook收购，再加上与苹果公司合作，让它成为了世界第一聊天软件。WhatsApp比微信更加简洁，更加注重聊天工具的本质，没有朋友圈，没有游戏平台，也没有表情商店。它的功能简单，程序小，使用起来非常流畅，和手机自带的短信一样快。且WhatsApp和手机通讯录深度整合，用户手机号码进行短信验证后，就自动将通讯录联系人。

总体来看实现在互联网上实现聊天兼文件传送功能的软件大部分做得都非常的成熟，很值得借鉴与学习，但是还是有很多功能有待优化和改进。在开发过程中还是存在以下问题：

1. 功能单一

一个系统只为了解决一个具体的问题而开发的。各个相关系统间没有紧密、有效的联系，影响了信息的重用和共享。如只能实现聊天功能或者只能实现文件传输功能，两者的功能不能有机的相结合。

1. Tcp粘包问题

发送方发送的若干包数据到达接收方时粘成了一包，从接收缓冲区来看，后一包数据的头紧接着前一包数据的尾，出现粘包的原因是多方面的，可能是来自发送方，也可能是来自接收方。

1. 在网上文件传送速度慢。
2. 在发送文件的实现机制没有优化和创新。
3. 大文件无法进行传输。

## 1.3 项目实施过程与进度说明

此项目由网路191 的徐艺芸独立完成，于2021年9月确定目标，10月设计思路并开发，11月中旬软件大致完成，11月底完成系统的全方位的测试和改进，总耗时3个月，由于笔者编程技术有限，和不太扎实的计算机网络知识储备，期间遇到了很多的问题和困难，但在本人的不懈努力和老师同学的鼎力帮助，最终实现了该功能。

以下是笔者大致如何此项项目的进度流程安排：

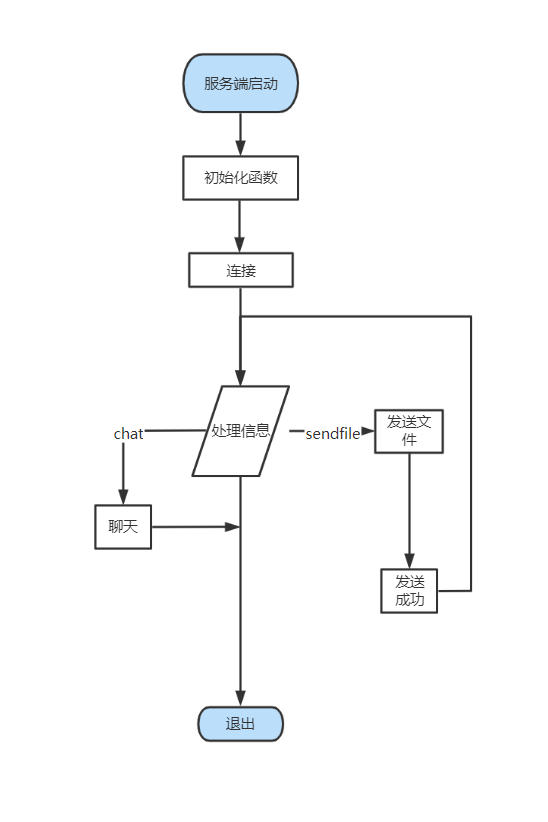
1. 9月份，在网上学习socket编程知识，选定一门开发语言定向练习；
2. 10月初跟着网上上教程学习socket聊天软件的开发；
3. 10月初跟着网上教程学习socket文件传输功能软件项目的开发；
4. 10月中旬对所学的知识进行整理，构思如何将文件传输功能和聊天功能结合起来；
5. 11月初将文件传输功能和聊天功能结合实现成功；
6. 11月中旬应用所学的操作系统的知识把并发机制加入到聊天系统中；
7. 11月底软件界面优化；
8. 11月底软件全方面测试。

# 2.项目实现基本原理

2.1流程图

由于此次项目的应用对象涉及到客户端和服务端，本次项目开发了两个应用程序，一方作为服务端，另一端作为客户端，下面是双方的工作流程图：

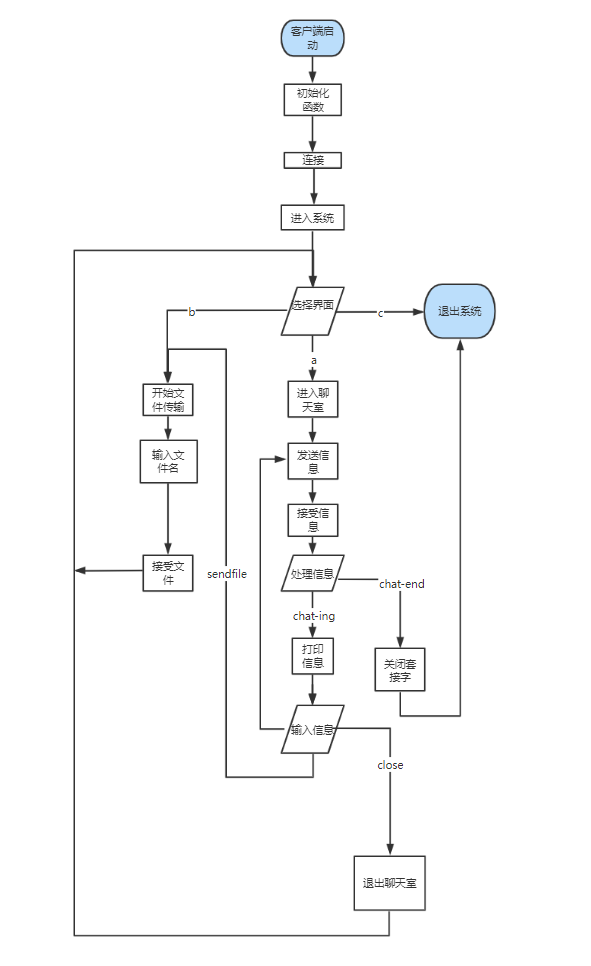
服务端：

****

**图2.1-1**

**服务端工作流程图**

客户端：

****

**图2.1-2**

**客户端工作流程图**

## 2.2实现思想概述

处理信息是此次开发过程的难点和突破点，为了实现聊天和发送文件合并在一个程序中，特设置标志位用来标志发送信息的属性，将发送的信息用一个结构体封装，双方在进行通信时根据发送信息的头部信息进行处理，本次项目中，充分利用开发工程中函数封装的思想，服务端只需调用一个处理信息函数就可以运行，客户端根据用户选择的需要分别独立设计了发送文件的处理函数和聊天的处理函数。这样做的好处是使程序的实现更具有条理性，开发更为简单和具体化，后期也便于维护、优化和改进。。

### 2.2.1系统处理信息的实现

由于发送缓存区大小容量的限制，使得大文件传输较为困难，但在实际应用环境中，简单的文件传输并不能满足其用户的需求，如果传送文件的方式是一次性的，系统往往会因为分配内存不足而导致文件传送失败或者信息的丢失。为了解决这一问题，本课程设计采取的思路是：服务端对要进行发送的文件进行拆分，少量多次的传送数据，客户端对传来数据的片段进行拼接，少量多次进行写入，直到接受信息完成。

以下是该系统处理信息的大致思想：

处理发送文件信息流程：

1、客户端输入文件名发送给客户端，标志位置1；

2、服务端接受到文件名，查找有无文件，有则发送文件大小给客户端，标志位置2，同时开辟发送缓存的内存空间；若无则发送无文件的消息，标志位置6；

3、客户端接受到文件大小就根据文件大小开辟内存空间并发送接受请求，标志位置3；接受到无文件信息继续输入文件名并发送，标志位置1；

4、服务端接受到接受请求后开始发送文件，标志位置4；发送完毕后再发送成功发送信号，标志位置5；关闭连接。

5、客户端接受文件，收到发送成功信号后停止接受，关闭连接。

### 2.2.2对文件进行拆包传输的实现

拆包思想概要：

发送大文件采用拆包的思想，并用标志位对发送的信息进行处理。双方根据包的头部信息判断下一步该做什么操作。

以下是客户端和服务段发送信息协议的具体格式：



图2.2.2 设计的协议格式

为了更加方便的理解其实现原理，现附上代码进行补充说明。

代码实现：

|  |
| --- |
| enum MSGTAG  **{**  MSG\_FILENAME **=** 1**,//文件名**  MSG\_FILESIZE **=** 2**,//文件大小**  MSG\_READY\_READ **=** 3**,//告诉客户端准备读**  MSG\_SENDFILE **=** 4**,//文件发送中**  MSG\_SUCCESSED **=** 5**,//文件发送成功**  MSG\_OPENFILE\_FAILD **=** 6**,//打开文件失败**  ENDSEND**=**7**,//终止发送文件**  // CHAT\_REQUSET **=** 8**,**  //CHAT\_START **=** 9  CHAT\_END **=** 10，//聊天结束  CHAT\_ING**=**11//聊天开始  **};**  #pragma pack(1)  #define PACKET\_SIZE (10240 -sizeof(int)\*3)//定义包的长度  struct MsgHeader  **{**  enum MSGTAG msgID**;**//当前消息标记  union MyUnion  **{**  struct  **{**  char fileName**[**256**];**  int fileSize**;**  **}**fileInfo**;** //260  struct  **{**  int nStart**;**  int nsize**;**//该包的数据大小  char buf**[**PACKET\_SIZE**];**  **}**packet**;**  struct  **{**  char chatbuf**[**BUFSIZ**];**  **}**CHAT**;**  **};**  **};**  #pragma pack() |

为了便于理解其构造的函数数据结构，下面具体描述其服务端拆包细节：

1. 包大小（PCKETSIZE）：10204字节，其中标志位占一个字节，文件起始位置占一个字节，包数据大小占一个字节，所以发送的文件信息占（10240-sizeof(int)）个字节；特别强调，发送文件内容（标志位为4）时，不必发送文件名和文件大小，因此不占用包的空间。
2. 用nstart来记录待发送文件信息的位置，即发送下一个包时，文件信息的起始位置。
3. 用nsize记录包数据大小。
4. 如果nstart+nsize+1>f-size,则说明这个包是最后一个包了，发送方根据这个条件判断发送完成情况。

### 2.2.3 聊天功能实现的具体流程

客户端选择聊天功能后，便调用聊天室功能的函数，以下是聊天室功能函数的流程分析：

1. 创建两个线程，一个是用于发送信息的发送线程，另一个是用于接受信息的接受线程。
2. 在发送线程中，用if判断语句对要发送的信息进行判断，聊天要发送的信息太长提醒用户重新输入，如果发送信息的内容是发送文件的命令则调用接受文件的函数对文件进行接受文件，如果是关闭系统的命令，就直接退出程序，在以上情况都不满足的条件下，发送聊天信息，完成后，继续等待用户输入信息进行发送。
3. 在接受线程中，也同时对发来的信息进行判断，如果信息标志是结束聊天就关闭程序，如果是正在聊天，则打印对方发来的信息，继续接受。

### 2.2.4 聊天中双方实现并发通信的实现

通过的简单调用堵塞型send(),recv()函数，并不能满足实际的开发的需要，为了使得双方通信时可以连续的发送信息和接受信息，笔者在实现这一功能时考虑了两种方案：

方法一：客户端定时连续发送，服务端定时连续接受；实现失败，难点在于：发送的信息无法预测下一个信息是否会发送，如果设置缓存区，实时性无法保证，并发聊天失去意义。

方法二：创建多线程，发送线程和接受线程同时进行。技术难点：send(),reciv()如何封装？

两种方法尝试后，笔者最终在方法二进行突破，参考基于window系统下相关线程并发的资料，决定利用操作系统的并发机制，考虑到程序运行环境在Windows操作系统中，且用的是C语言编写，查阅资料可知，使用Win32提供的在 C/C++运行时库中的\_beginthreade，可以来完成用户线程的创建，该API的原型为：

|  |
| --- |
| uintptr\_t \_beginthreadex(  　void \*security,//Pointer to a SECURITY\_ATTRIBUTES structure  　unsigned stack\_size,  　unsigned ( \_\_stdcall \*start\_address )( void \* ),  　void \*arglist,  　unsigned initflag,//Initial state of new thread (0 for running or CREATE\_SUSPENDED for suspended);  　unsigned \*thrdaddr  ); |

WaitForSingleObject是一种Windows API函数。当等待仍在[挂起状态](https://baike.baidu.com/item/%E6%8C%82%E8%B5%B7%E7%8A%B6%E6%80%81/9359976)时，句柄被关闭，那么函数行为是未定义的。该句柄必须具有 SYNCHRONIZE 访问权限。因此调用 WaitForSingleObject()函数等待其线程的结束，该API的原型为

|  |
| --- |
| DWORD WaitForSingleObject**(**  HANDLE hHandle**,**  DWORD dwMilliseconds  **);** |

于是笔者将利用这两个基本函数实现对收发功能线程的创建，下面便是其实现函数

客户端创建线程的代码展示：

|  |
| --- |
| HANDLE hThread**[**2**];**//声明两个线程  unsigned threadID**[**2**];**  hThread**[**0**]** **=** **(**HANDLE**)**\_beginthreadex**(NULL,** 0**,** p\_send**,** **&**serfd**,** 0**,** **&**threadID**[**0**]);**//多线程send;  hThread**[**1**]** **=** **(**HANDLE**)**\_beginthreadex**(NULL,** 0**,** p\_recv**,** **&**serfd**,** 0**,** **&**threadID**[**1**]);**//多线程recv;  WaitForSingleObject**(**hThread**[**0**],** INFINITE**);**//等待线程结束  WaitForSingleObject**(**hThread**[**1**],** INFINITE**);**  CloseHandle**(**hThread**[**0**]);**//关闭线程  CloseHandle**(**hThread**[**1**]);** |

根据客户端和服务端会根据其信息的类型所作出不同响应，对发送信息和接受信息处理过程分别用p\_send()和p\_recv()函数进行封装，这样封装可以更好的调用用\_beginthreade函数完成用户收发线程的创建

下面仅列举客户端收发函数封装过程，由于服务端和客户端采取的机制一致，就不再累述：

客户端实现并发发送函数：

|  |
| --- |
| unsigned \_\_stdcall p\_send**(**void**\*** pfd**)** **{**  SOCKET fd**;**  fd **=** **\*((**SOCKET**\*)**pfd**);**  struct MsgHeader Chat3**;**//用于发送消息  int flag **=** 0**;**//用于标记是否退出  **while** **(**flag**==**0**)** **{**  printf**(**"send>"**);**  gets\_s**(**Chat3**.**CHAT**.**chatbuf**,** BUFSIZ**);**  //如果输入的内容是close，就关闭  **if** **(**strcmp**(**Chat3**.**CHAT**.**chatbuf**,** "close"**)** **==** 0**)**  **{**  Chat3**.**msgID **=** CHAT\_END**;**  send**(**fd**,** **(**char**\*)&**Chat3**,** **sizeof(**struct MsgHeader**),** 0**);**  printf**(**"exit chatroom,bye~bye~\n"**);**  flag **=** 1**;**//标志结束  closesocket**(**fd**);**  **return** **false;**  **}**  //如果输入sendfile，就发送文件  **if** **(**strcmp**(**Chat3**.**CHAT**.**chatbuf**,** "sendfile"**)** **==** 0**)**  **{**  downloadFileName**(**fd**);**  **}**  Chat3**.**msgID **=** CHAT\_ING**;**  **if** **(**SOCKET\_ERROR **==** send**(**fd**,** **(**char**\*)&**Chat3**,** **sizeof(**struct MsgHeader**),** 0**))**  **{**  printf**(**"server had closed!\n"**);**  **return** **false;**  **}**  memset**(&**Chat3**,** 0**,** **sizeof(**Chat3**.**CHAT**.**chatbuf**));**  **}**  **}** |

客户端实现并发接受函数：

|  |
| --- |
| unsigned \_\_stdcall p\_recv**(**void**\*** pfd**)** **{**  int flag **=** 0**;**//用于标记退出  **while** **(**flag**==**0**)** **{**  SOCKET fd**;**  fd **=** **\*((**SOCKET**\*)**pfd**);**  int nRes **=** recv**(**fd**,** g\_recvBuf**,** 1024**,** 0**);**  **if** **(**0 **<** nRes**)** **{**  struct MsgHeader**\*** msg **=** **(**struct MsgHeader**\*)**g\_recvBuf**;**  **switch** **(**msg**->**msgID**)**  **{**  **case** MSG\_OPENFILE\_FAILD**:**  printf**(**"send file failed ,please try again\n"**);**  downloadFileName**(**fd**);**  **break;**  **case** MSG\_FILESIZE**:**  readyread**(**fd**,** msg**);**  **break;**  **case** MSG\_READY\_READ**:**  writeFile**(**fd**,** msg**);**  **break;**  **case** MSG\_SUCCESSED**:**  printf**(**"send file success!\n"**);**  //进入选项窗口  **break;**  **case** CHAT\_ING**:**  printf**(**"recv:%s\n"**,** msg**->**CHAT**.**chatbuf**);**  memset**(**msg**,** 0**,** **sizeof(**msg**));**//清空缓存防止溢出  **break;**  **case** CHAT\_END**:**  printf**(**"the client had exit this system!\n"**);**  closeSocket**();**  flag **=** 1**;**  **break;**  **}**  **}**  **}**  **}** |

# 3.开发环境及所用技术简介

## 3.1开发环境

以下是本系统相关开发环境和所用技术说明：

|  |  |
| --- | --- |
| 编程工具 | Visual Studio 2019 |
| 编程语言 | C语言 |
| 演示系统 | Windows·10 家庭版 |
| 电脑配置 |  |

## 3.2所用技术简介

1、Visual Studio 2019简介

Microsoft Visual Studio（简称VS）是美国[微软公司](https://baike.baidu.com/item/%E5%BE%AE%E8%BD%AF%E5%85%AC%E5%8F%B8)的[开发工具](https://baike.baidu.com/item/%E5%BC%80%E5%8F%91%E5%B7%A5%E5%85%B7/10464557)包系列产品。VS是一个基本[完整](https://baike.baidu.com/item/%E5%AE%8C%E6%95%B4/32785)的开发工具集，它包括了整个[软件生命周期](https://baike.baidu.com/item/%E8%BD%AF%E4%BB%B6%E7%94%9F%E5%91%BD%E5%91%A8%E6%9C%9F/861455)中所需要的大部分工具，如[UML](https://baike.baidu.com/item/UML/446747)工具、代码[管控](https://baike.baidu.com/item/%E7%AE%A1%E6%8E%A7/24264804)工具、[集成开发环境](https://baike.baidu.com/item/%E9%9B%86%E6%88%90%E5%BC%80%E5%8F%91%E7%8E%AF%E5%A2%83/298524)(IDE)等等。所写的[目标代码](https://baike.baidu.com/item/%E7%9B%AE%E6%A0%87%E4%BB%A3%E7%A0%81/9407934)适用于微软支持的所有平台，包括[Microsoft Windows](https://baike.baidu.com/item/Microsoft%20Windows)、[Windows Mobile](https://baike.baidu.com/item/Windows%20Mobile/749381)、[Windows CE](https://baike.baidu.com/item/Windows%20CE)、[.NET Framework](https://baike.baidu.com/item/.NET%20Framework)、[.NET Compact Framework](https://baike.baidu.com/item/.NET%20Compact%20Framework/1926409)和Microsoft [Silverlight](https://baike.baidu.com/item/Silverlight) 及[Windows Phone](https://baike.baidu.com/item/Windows%20Phone/9227600)。

Visual Studio是最流行的Windows平台应用程序的[集成开发环境](https://baike.baidu.com/item/%E9%9B%86%E6%88%90%E5%BC%80%E5%8F%91%E7%8E%AF%E5%A2%83/298524)。最新版本为 Visual Studio 2019 版本，基于.NET Framework 4.8 。

2、c 语言简介

C语言是一门[面向过程](https://baike.baidu.com/item/%E9%9D%A2%E5%90%91%E8%BF%87%E7%A8%8B/9957246)的、[抽象化](https://baike.baidu.com/item/%E6%8A%BD%E8%B1%A1%E5%8C%96/10844295)的通用程序设计语言，广泛应用于[底层开发](https://baike.baidu.com/item/%E5%BA%95%E5%B1%82%E5%BC%80%E5%8F%91/6448268)。C语言能以简易的方式[编译](https://baike.baidu.com/item/%E7%BC%96%E8%AF%91/1258343)、处理低级[存储器](https://baike.baidu.com/item/%E5%AD%98%E5%82%A8%E5%99%A8/1583185)。C语言是仅产生少量的[机器语言](https://baike.baidu.com/item/%E6%9C%BA%E5%99%A8%E8%AF%AD%E8%A8%80/2019225)以及不需要任何[运行环境](https://baike.baidu.com/item/%E8%BF%90%E8%A1%8C%E7%8E%AF%E5%A2%83/6555199)支持便能运行的高效率程序设计语言。尽管C语言提供了许多低级处理的[功能](https://baike.baidu.com/item/%E5%8A%9F%E8%83%BD/10346898)，但仍然保持着[跨平台](https://baike.baidu.com/item/%E8%B7%A8%E5%B9%B3%E5%8F%B0/8558902)的特性，以一个标准规格写出的C语言程序可在包括类似[嵌入式处理器](https://baike.baidu.com/item/%E5%B5%8C%E5%85%A5%E5%BC%8F%E5%A4%84%E7%90%86%E5%99%A8/9242667)以及[超级计算机](https://baike.baidu.com/item/%E8%B6%85%E7%BA%A7%E8%AE%A1%E7%AE%97%E6%9C%BA/5373711)等作业平台的许多[计算机平台](https://baike.baidu.com/item/%E8%AE%A1%E7%AE%97%E6%9C%BA%E5%B9%B3%E5%8F%B0/2606037)上进行[编译](https://baike.baidu.com/item/%E7%BC%96%E8%AF%91/1258343)。

## 3.3 socket相关背景知识简介

为便于理解该项目的工作原理，特补充其课题实现的背景和原理。

### 3.3.1、socket的分类

socket提供了流（stream)和数据报（datagram)两种通信机制，即流socket和数据报socket。

流socket基于TCP协议，是一个有序、可靠、双向字节流的通道，传输数据不会丢失、不会重复、顺序也不会错乱。

数据报socket基于UDP协议，不需要建立和维持连接，可能会丢失或错乱。UDP不是一个可靠的协议，对数据的长度有限制，但是它的效率比较高。

某些应用层协议，出于历史原因，受当时技术和网络条件限制，选择了基于UDP实现，其选择的理由现在很可能已经不再成立了。

实时的音视频聊天可能采用的是UDP，这种业务可以接受数据的丢失且不必重传。

### 3.3.2、Socket通信工作原理图

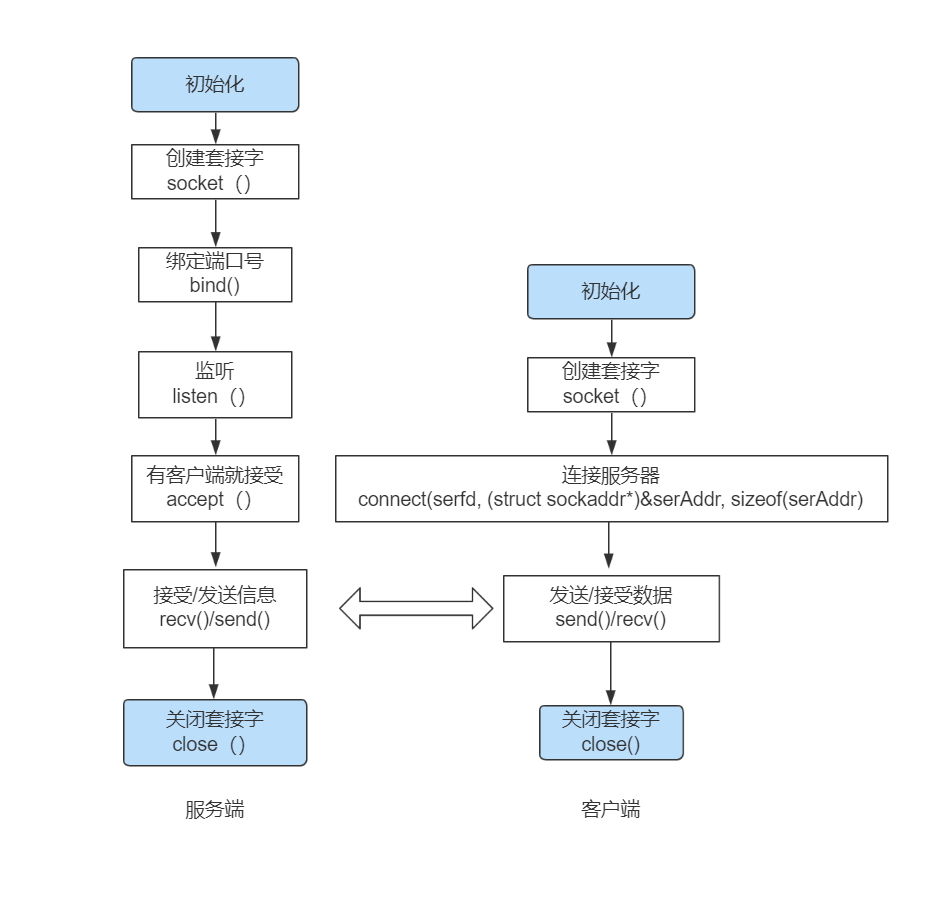


图3.3 Socket通信工作原理图

服务端的工作流程

1）创建服务端的socket。

2）把服务端用于通信的地址和端口绑定到socket上。

3）把socket设置为监听模式。

4）接受客户端的连接。

5）与客户端通信，接收客户端发过来的报文后，回复处理结果。

6）不断的重复第5）步，直到客户端断开连接。

7）关闭socket，释放资源。

客户端的工作流程

1）创建客户端的socket。

2）向服务器发起连接请求。

3）与服务端通信，发送一个报文后等待回复，然后再发下一个报文。

4）不断的重复第3）步，直到全部的数据被发送完。

5）第4步：关闭socket，释放资源。

## 3.4、相关库函数介绍

本程序运行的环境是在Windows下进行的，进行socket编程函数调用的使用的库是：WinSock2.h，下面是本程序调用该库各个函数的简介：

### 3.4.1、socket函数

socket函数用于创建一个新的socket，也就是向系统申请一个socket资源。socket函数用户客户端和服务端。

函数声明：

|  |
| --- |
| int socket(int domain, int type, int protocol); |

参数说明：

* domain：协议域，又称协议族（family）。常用的协议族有AF\_INET、AF\_INET6、AF\_LOCAL（或称AF\_UNIX，Unix域Socket）、AF\_ROUTE等。协议族决定了socket的地址类型，在通信中必须采用对应的地址，如AF\_INET决定了要用ipv4地址（32位的）与端口号（16位的）的组合、AF\_UNIX决定了要用一个绝对路径名作为地址。
* type：指定socket类型。常用的socket类型有SOCK\_STREAM、SOCK\_DGRAM、SOCK\_RAW、SOCK\_PACKET、SOCK\_SEQPACKET等。流式socket（SOCK\_STREAM）是一种面向连接的socket，针对于面向连接的TCP服务应用。数据报式socket（SOCK\_DGRAM）是一种无连接的socket，对应于无连接的UDP服务应用。
* protocol：指定协议。常用协议有IPPROTO\_TCP、IPPROTO\_UDP、IPPROTO\_STCP、IPPROTO\_TIPC等，分别对应TCP传输协议、UDP传输协议、STCP传输协议、TIPC传输协议。第一个参数只能填AF\_INET，第二个参数只能填SOCK\_STREAM，第三个参数只能填0。除非系统资料耗尽，socket函数一般不会返回失败。
* 返回值：成功则返回一个socket，失败返回-1，错误原因存于errno 中。

### 3.4.2、gethostbyname函数

把ip地址或域名转换为hostent结构体表达的地址。

函数声明：

|  |
| --- |
| struct hostent \*gethostbyname(const char \*name); |

* 参数name，域名或者主机名，例如"192.168.1.3"、"www.freecplus.net"等。
* 返回值：如果成功，返回一个hostent结构指针，失败返回NULL。

gethostbyname只用于客户端。

gethostbyname只是把字符串的ip地址转换为结构体的ip地址，只要地址格式没错，一般不会返回错误。失败时不会设置errno的值。

#### 3.4.3、connect函数

向服务器发起连接请求。

函数声明：

|  |
| --- |
| int connect(int sockfd, struct sockaddr \* serv\_addr, int addrlen); |

* 函数说明：connect函数用于将参数sockfd 的socket 连至参数serv\_addr 指定的服务端，参数addrlen为sockaddr的结构长度。
* 返回值：成功则返回0，失败返回-1，错误原因存于errno 中。
* connect函数只用于客户端。

如果服务端的地址错了，或端口错了，或服务端没有启动，connect一定会失败。

### 3.4.4、bind函数

服务端把用于通信的地址和端口绑定到socket上。

函数声明：

|  |
| --- |
| int bind(int sockfd, const struct sockaddr \*addr,socklen\_t addrlen); |

* 参数sockfd，需要绑定的socket。
* 参数addr，存放了服务端用于通信的地址和端口。
* 参数addrlen表示addr结构体的大小。
* 返回值：成功则返回0，失败返回-1，错误原因存于errno 中。
* 如果绑定的地址错误，或端口已被占用，bind函数一定会报错，否则一般不会返回错误。

### 3.4.5、listen函数

listen函数把主动连接socket变为被动连接的socket，使得这个socket可以接受其它socket的连接请求，从而成为一个服务端的socket。

函数声明：

|  |
| --- |
| int listen(int sockfd, int backlog); |

* 参数sockfd是已经被bind过的socket。socket函数返回的socket是一个主动连接的socket，在服务端的编程中，程序员希望这个socket可以接受外来的连接请求，也就是被动等待客户端来连接。由于系统默认时认为一个socket是主动连接的，所以需要通过某种方式来告诉系统，程序员通过调用listen函数来完成这件事。
* 参数backlog，这个参数涉及到一些网络的细节，比较麻烦，填5、10都行，一般不超过30。当调用listen之后，服务端的socket就可以调用accept来接受客户端的连接请求。
* 返回值：成功则返回0，失败返回-1，错误原因存于errno 中。listen函数一般不会返回错误。

### 3.4.6、accept函数

服务端接受客户端的连接。

函数声明：

|  |
| --- |
| int accept(int sockfd,struct sockaddr \*addr,socklen\_t \*addrlen); |

* 参数sockfd是已经被listen过的socket。
* 参数addr用于存放客户端的地址信息，用sockaddr结构体表达，如果不需要客户端的地址，可以填0。
* 参数addrlen用于存放addr参数的长度，如果addr为0，addrlen也填0。
* accept函数等待客户端的连接，如果没有客户端连上来，它就一直等待，这种方式称之为阻塞。
* accept等待到客户端的连接后，创建一个新的socket，函数返回值就是这个新的socket，服务端使用这个新的socket和客户端进行报文的收发。
* 返回值：成功则返回0，失败返回-1，错误原因存于errno 中。

accept在等待的过程中，如果被中断或其它的原因，函数返回-1，表示失败，如果失败，可以重新accept。

### 3.4.7、send函数

send函数用于把数据通过socket发送给对端。不论是客户端还是服务端，应用程序都用send函数来向TCP连接的另一端发送数据。

函数声明：

|  |
| --- |
| ssize\_t send(int sockfd, const void \*buf, size\_t len, int flags); |

* sockfd为已建立好连接的socket。
* buf为需要发送的数据的内存地址，可以是C语言基本数据类型变量的地址，也可以数组、结构体、字符串，内存中有什么就发送什么。
* len需要发送的数据的长度，为buf中有效数据的长度。
* flags填0, 其他数值意义不大。

函数返回已发送的字符数。出错时返回-1，错误信息errno被标记。

注意，就算是网络断开，或socket已被对端关闭，send函数不会立即报错，要过几秒才会报错。

如果send函数返回的错误（<=0），表示通信链路已不可用。

### 3.4.8、recv函数

recv函数用于接收对端socket发送过来的数据。

recv函数用于接收对端通过socket发送过来的数据。不论是客户端还是服务端，应用程序都用recv函数接收来自TCP连接的另一端发送过来数据。

函数声明：

|  |
| --- |
| ssize\_t recv(int sockfd, void \*buf, size\_t len, int flags); |

* sockfd为已建立好连接的socket。
* buf为用于接收数据的内存地址，可以是C语言基本数据类型变量的地址，也可以数组、结构体、字符串，只要是一块内存就行了。
* len需要接收数据的长度，不能超过buf的大小，否则内存溢出。
* flags填0, 其他数值意义不大。

函数返回已接收的字符数。出错时返回-1，失败时不会设置errno的值。

如果socket的对端没有发送数据，recv函数就会等待，如果对端发送了数据，函数返回接收到的字符数。出错时返回-1。如果socket被对端关闭，返回值为0。

如果recv函数返回的错误（<=0），表示通信通道已不可用。

### 3.4.9、socket文件描述符

在UNIX系统中，一切输入输出设备皆文件，socket()函数的返回值其本质是一个文件描述符，是一个整数。

### 3.4.10、服务端程序绑定地址

如果服务器有多个网卡，多个IP地址，socket通信可以指定用其中一个地址来进行通信，也可以任意ip地址。

1. 指定ip地址的代码

|  |
| --- |
| m\_servaddr.sin\_addr.s\_addr = inet\_addr("192.168.149.129"); // 指定ip地址 |

1. 任意ip地址的代码

|  |
| --- |
| m\_servaddr.sin\_addr.s\_addr = htonl(INADDR\_ANY); // 本主机的任意ip地址 |

在实际开发中，采用任意ip地址的方式比较多。

### 3.4.11、服务端程序绑定的通信端口

|  |
| --- |
| m\_servaddr.sin\_port = htons(5000); // 通信端口 |

### 3.4.12、客户端程序指定服务端的ip地址

|  |
| --- |
| struct hostent\* h;  if ( (h = gethostbyname("118.89.50.198")) == 0 ) // 指定服务端的ip地址。  { printf("gethostbyname failed.\n"); close(sockfd); return -1; } |

### 3.4.13、客户端程序指定服务端的通信端口

|  |
| --- |
| servaddr.sin\_port = htons(5000); |

### 3.4.14、服务端有两个socket

对服务端来说，有两个socket，一个是用于监听的socket，还有一个就是客户端连接成功后，由accept函数创建的用于与客户端收发报文的socket。

#### 3.4.15、程序退出时先关闭socket

socket是系统资源，操作系统打开的socket数量是有限的，在程序退出之前必须关闭已打开的socket，就像关闭文件指针一样，就像delete已分配的内存一样，极其重要。

值得注意的是，关闭socket的代码不能只在main函数的最后，那是程序运行的理想状态，还应该在main函数的每个return之前关闭。

# 4.关键代码展示：

由于版面限制，以下仅展示其程序的关键代码。

## （1）客户端连接服务器代码

为了连接服务器，客户端要先创建一个代表服务器的套接字，然后使用connect函数连接。

|  |
| --- |
| void connectToHost**()**  **{**  //创建server socket套接字 地址，端口号  //INVALID\_SOKET SOCKET\_ERROR  SOCKET serfd **=** socket**(**AF\_INET**,** SOCK\_STREAM**,** IPPROTO\_TCP**);**  **if** **(**serfd **==** INVALID\_SOCKET**)**  **{**  printf**(**"socket faild :%d\n"**,** WSAGetLastError**());**  **return;**  **}**  //给socket绑定ip和端口号  struct sockaddr\_in serAddr**;**  serAddr**.**sin\_family **=** AF\_INET**;**//必须和创建的socket指定的一样  serAddr**.**sin\_port **=** htons**(**SPORT**);**  serAddr**.**sin\_addr**.**S\_un**.**S\_addr **=** inet\_addr**(**"127.0.0.1"**);**//监听本机所有网卡  //链接到服务器  **if** **(**0 **!=** connect**(**serfd**,** **(**struct sockaddr**\*)&**serAddr**,** **sizeof(**serAddr**)))**  **{**  printf**(**"connnect faild :%d\n"**,** WSAGetLastError**());**  **return;**  **}**  printf**(**"conect success!"**);** |

## （2）客户端选择功能代码段（界面显示）

|  |
| --- |
| //开始处理消息,选择聊天室功能还是文件传输功能：  char ch**;**  **while** **(true)** **{**  printf**(**"chat or sendfile?\nif chat press a,if sendfile press b，else you will exit this systm!\n"**);**  printf**(**"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n"**);**  printf**(**" a: chat\n"**);**  printf**(**" b: sendfile\n"**);**  printf**(**" c:exit this window\n"**);**  printf**(**"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n"**);**  printf**(**"Enter your choice (a or b or c): "**);**  //如果输入信息不正确，继续输入  **do{**  ch **=** **(**char**)**\_getch**();**  **}** **while** **(**ch **!=** 'a' **&&** ch **!=** 'b' **&&** ch **!=** 'c'**);**  system**(**"cls"**);** //清除控制台显示的信息  //进入聊天室  **if** **(**ch **==** 'a'**)**  **{**  printf**(**"welcome to yy's chatroom!\n"**);**  chatroom**(**serfd**)；**  **}**  //进行文件传输  **if** **(**ch **==** 'b'**)** **{**  printf**(**"welcome to download file!\n"**);**  //获取要发送文件名称  downloadFileName**(**serfd**);**  **while** **(**processMsg**(**serfd**)){}**  **}**  //退出系统  **if** **(**ch **==** 'c'**)** **{**  printf**(**"see you next time!\n"**);**  closesocket**(**serfd**);**  **return** **false;**  **}**  printf**(**"\nPress Any Key To Continue:"**);**  \_getch**();**  system**(**"cls"**);**  **}**  **return** **true;**  **}** |

## （3）服务器创建套接字流程：

|  |
| --- |
| void listenToClient**()**  **{**  printf**(**"listen to client....\n"**);**  //创建server socket套接字 地址，端口号  //INVALID\_SOKET SOCKET\_ERROR  SOCKET serfd **=** socket**(**AF\_INET**,** SOCK\_STREAM**,** IPPROTO\_TCP**);**  **if** **(**serfd **==** INVALID\_SOCKET**)**  **{**  printf**(**"socket faild :%d\n"**,** WSAGetLastError**());**  **return;**  **}**  //给socket绑定ip和端口号  struct sockaddr\_in serAddr**;**  serAddr**.**sin\_family **=** AF\_INET**;**//必须和创建的socket指定的一样  serAddr**.**sin\_port **=** htons**(**SPORT**);**  serAddr**.**sin\_addr**.**S\_un**.**S\_addr **=** ADDR\_ANY**;**//监听本机所有网卡  //绑定套接字  **if** **(**0 **!=** bind**(**serfd**,** **(**struct sockaddr**\*)&**serAddr**,** **sizeof(**serAddr**)))**  **{**  printf**(**"bind faild :%d\n"**,** WSAGetLastError**());**  **return;**  **}**  //监听客户端连接  **if** **(**0 **!=** listen**(**serfd**,** 10**))**  **{**  printf**(**"listen faild :%d\n"**,** WSAGetLastError**());**  **return;**  **}**  //有客户端连接，就接受  struct sockaddr\_in cliAddr**;**  int len **=** **sizeof(**cliAddr**);**  SOCKET clifd **=** accept**(**serfd**,** **(**struct sockaddr**\*)&**cliAddr**,** **&**len**);**  **if** **(**INVALID\_SOCKET **==** clifd**)**  **{**  printf**(**"accept faild :%d\n"**,** WSAGetLastError**());**  **return;**  **}**  printf**(**"new client connect success....\n"**);** |

## （4）服务端处理信息代码段：

|  |
| --- |
| bool processMsg**(**SOCKET clifd**)**  **{**  int nRes **=** recv**(**clifd**,** g\_recvBuf**,** 10240**,** 0**);**  //成功接收消息，返回接受到的字节数，接受失败返回0  **if** **(**nRes **<=**0**)**  **{**  printf**(**"客户端下线。。。。%d\n"**,** WSAGetLastError**());**  **return** **false;**  **}**  //将接受到的缓存信息强转为结构体类型，便于之后的信息判断  struct MsgHeader**\*** msg **=** **(**struct MsgHeader**\*)**g\_recvBuf**;**  //在内部无法定义结构体，只能在外面定义，用于消息的发送  struct MsgHeader exitmsg**;**  **switch** **(**msg**->**msgID**)**  **{**  **case** MSG\_FILENAME**:**  printf**(**"要发送的文件是：%s\n"**,** msg**->**fileInfo**.**fileName**);**  readFile**(**clifd**,** msg**);**//查找文件  **break;**  **case** MSG\_SENDFILE**:**  sendFile**(**clifd**,** msg**);**//发送文件  **break;**  **case** MSG\_SUCCESSED**:**  exitmsg**.**msgID **=** MSG\_SUCCESSED**;**  **if** **(**SOCKET\_ERROR **==** send**(**clifd**,** **(**char**\*)&**exitmsg**,** **sizeof(**struct MsgHeader**),** 0**))**  **{**  printf**(**"send faild :%d\n"**,** WSAGetLastError**());**  **return** **false;**  **}**  printf**(**"客户端已成功接受！~\n"**);**  **break;**  **case** ENDSEND**:**  printf**(**"客户端下线，bye~bye~\n"**);**  closesocket**(**clifd**);**  **return** FALSE**;**  **break;**  **case** CHAT\_ING**:**  printf**(**"recv:%s\n"**,** msg**->**CHAT**.**chatbuf**);**  memset**(**msg**,** 0**,** **sizeof(**msg**));**//清空缓存防止溢出  HANDLE hThread**[**2**];**//创建线程句柄  unsigned threadID**[**2**];**  hThread**[**0**]** **=** **(**HANDLE**)**\_beginthreadex**(NULL,** 0**,** p\_send**,** **&**clifd**,** 0**,** **&**threadID**[**0**]);**//多线程send;  hThread**[**1**]** **=** **(**HANDLE**)**\_beginthreadex**(NULL,** 0**,** p\_recv**,** **&**clifd**,** 0**,** **&**threadID**[**1**]);**//多线程recv;  WaitForSingleObject**(**hThread**[**0**],** INFINITE**);**  WaitForSingleObject**(**hThread**[**1**],** INFINITE**);**  CloseHandle**(**hThread**[**0**]);**  CloseHandle**(**hThread**[**1**]);**  **break;**  **case** CHAT\_END**:**  printf**(**"客户端不想聊天啦\n"**);**  closesocket**(**clifd**);**  memset**(**msg**,** 0**,** **sizeof(**msg**));**  **return** **false;**  **break;**  **}**  **return** **true;**  **}** |

## （5）客户端聊天室代码：

|  |
| --- |
| bool chatroom**(**SOCKET serfd**)**  **{**  HANDLE hThread**[**2**];**//声明两个线程  unsigned threadID**[**2**];**  hThread**[**0**]** **=** **(**HANDLE**)**\_beginthreadex**(NULL,** 0**,** p\_send**,** **&**serfd**,** 0**,** **&**threadID**[**0**]);**//多线程p\_send;  hThread**[**1**]** **=** **(**HANDLE**)**\_beginthreadex**(NULL,** 0**,** p\_recv**,** **&**serfd**,** 0**,** **&**threadID**[**1**]);**//多线程p\_recv;  WaitForSingleObject**(**hThread**[**0**],** INFINITE**);**//等待线程结束  WaitForSingleObject**(**hThread**[**1**],** INFINITE**);**  CloseHandle**(**hThread**[**0**]);**//关闭线程  CloseHandle**(**hThread**[**1**]);**  **}** |

## （6）客户端接受文件代码（如何组装包）

|  |
| --- |
| bool writeFile**(**SOCKET serfd**,** struct MsgHeader**\*** pmsg**)**  **{**  //如果g\_fileBUf为空，则分配内存失败  **if** **(**g\_fileBuf **==** **NULL)**  **{**  **return** FALSE**;**  **}**  //将指针移到开始读的位置  int nStart **=** pmsg**->**packet**.**nStart**;**  //获取包长度  int nsize **=** pmsg**->**packet**.**nsize**;**  //拼接  memcpy**(**g\_fileBuf**+**nStart**,**pmsg**->**packet**.**buf**,** nsize**);**  //判断文件是否完整，服务器是否发完了数据，接受到最后一个包时开始写入  **if(**nStart **+** nsize **>=** g\_fileSize**)**  **{**  FILE**\*** pwrite **=** fopen**(**g\_fileName**,** "wb"**);**  **if** **(**pwrite **==** **NULL)**  **{**  printf**(**"write file error..\n"**);**  **return** **false;**  **}**  fwrite**(**g\_fileBuf**,** **sizeof(**char**),** g\_fileSize**,** pwrite**);**  //写完释放缓存，关闭读指针  fclose**(**pwrite**);**  free**(**g\_fileBuf**);**  g\_fileBuf **=** **NULL;**  //发送写入成功消息  struct MsgHeader msg**;**  msg**.**msgID **=** MSG\_SUCCESSED**;**  send**(**serfd**,** **(**char**\*)&**msg**,** **sizeof(**struct MsgHeader**),** 0**);**  **}**  **return** **true;**  **}** |

## （7）服务端发送文件代码：

|  |
| --- |
| bool sendFile**(**SOCKET clifd**,** struct MsgHeader**\*** pmsg**)**  **{**  //告诉客户端准备接受文件了  //如果文件的长度大于每个数据包能传送的大小(packet\_size)那么就分快  struct MsgHeader msg**;**  msg**.**msgID **=** MSG\_READY\_READ**;**    **for** **(**size\_t i **=** 0**;** i **<** g\_fileSize**;** i **+=** PACKET\_SIZE**)**  **{**  msg**.**packet**.**nStart **=** i**;**  //判断是否为最后一个包，获取最后一个包的长度  **if** **(**i **+** PACKET\_SIZE**+**1 **>**g\_fileSize**)**  **{**  msg**.**packet**.**nsize **=** g\_fileSize **-** i**;**  **}**  **else**  **{**  msg**.**packet**.**nsize **=** PACKET\_SIZE**;**  **}**  memcpy**(**msg**.**packet**.**buf**,** g\_fileBuf **+** msg**.**packet**.**nStart**,** msg**.**packet**.**nsize**);**  **if** **(**SOCKET\_ERROR **==** send**(**clifd**,** **(**char**\*)&**msg**,** **sizeof(**struct MsgHeader**),** 0**))**  **{**  printf**(**"文件发送失败%d\n"**,**WSAGetLastError**());**  **return** **false;**  **}**  **}**    **return** TRUE**;**  **}** |

## (8)发送成功后选择

|  |
| --- |
| **case** MSG\_SUCCESSED**:**  printf**(**"send file success!\n"**);**  //进入选项窗口  char key**;**  **while** **(true)**  **{**  printf**(**"if you want to continue to download,print 1,if you want to chat,print 2,else press another key:\n "**);**  printf**(**"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n"**);**  printf**(**" 1: download\n"**);**  printf**(**" 2: chat\n"**);**  printf**(**" 3:exit this window\n"**);**  printf**(**"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n"**);**  printf**(**"Enter your choice (1 or 2 or 3): "**);**  **do** **{**  key **=** **(**char**)**\_getch**();**  **}** **while** **(**key **!=** '1'**&&** key **!=**'2' **&&** key **!=** '3'**);**  system**(**"cls"**);** //清除控制台显示的信息  **if** **(**key **==** '1'**)** **{**  downloadFileName**(**serfd**);**  **while** **(**processMsg**(**serfd**)){}**  **}**  **if** **(**key **==**'2'**)** **{**  struct MsgHeader Chat**;**  Chat**.**msgID **=** CHAT\_ING**;**  printf**(**"welcome to yy's chatroom!\n"**);**  chatroom**(**serfd**);**  **break;**  **}**  **if** **(**key **==**'3'**)** **{**  struct MsgHeader file**;**  file**.**msgID **=** ENDSEND**;**  **if** **(**SOCKET\_ERROR **==** send**(**serfd**,** **(**char**\*)&**file**,** **sizeof(**struct MsgHeader**),** 0**))**  **{**  err**(**"send"**);**  **return;**  **}**  printf**(**"关闭系统，欢迎下次再见~\n"**);**  closesocket**(**serfd**);**  **return** **false;**  **}**  **}**  **break;** |

# 5.功能展示：

1、连接客户端

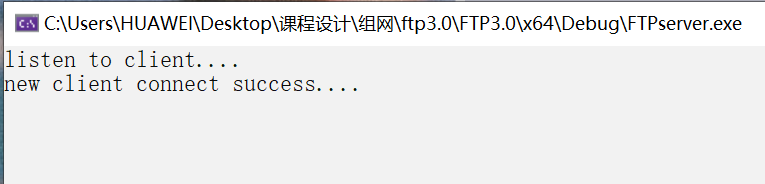


图5.1 服务器连接客户端成功显示图

2、选择菜单

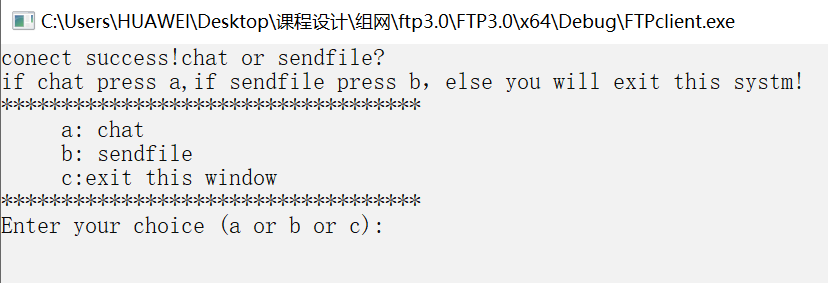


图5.2 客户端选择功能图

3、选择a,进入聊天室：

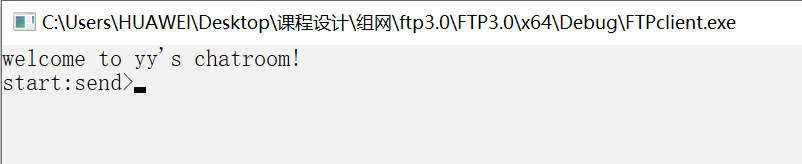


图3.1 客户端进入聊天室图

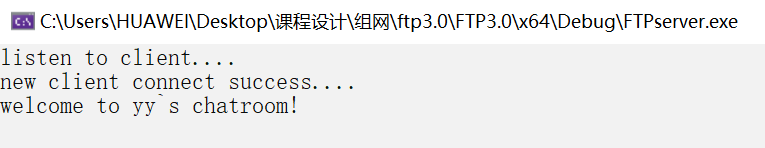


图3.2 服务端进入聊天室图



图3.3 客户端收发功能并发示意图



图3.4 服务端收发功能并发示意图

4、客户端在聊天室中输入sendfile命令，申请发送文件

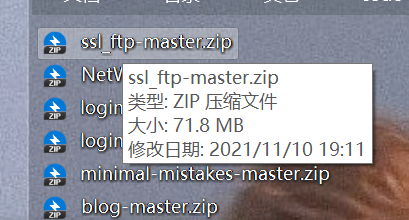
****

图4.1 测试文件大小示意图

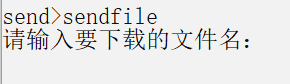
****

图4.2 sendfile命令示意图

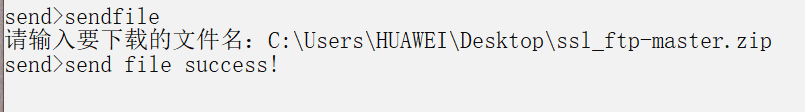
****

图4.3 客户端申请发送文件接受成功示意图

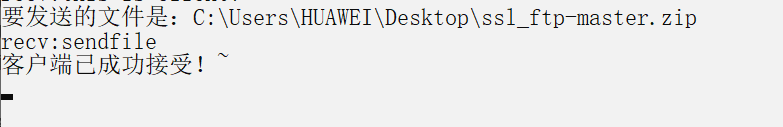
****

图4.4服务端发送文件成功示意图

****

图4.5 文件接受成功图

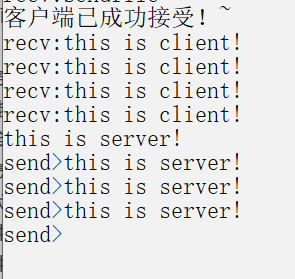
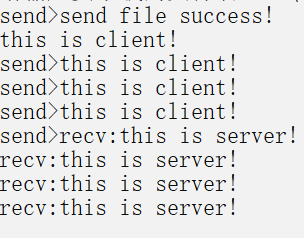
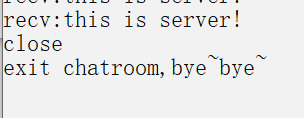
** **

图4.6 发送完毕继续聊天示意图

****

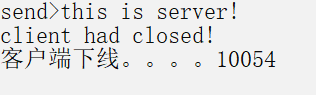
****

图4.7 客户端输入“close”命令效果图

5、选择b,发送文件

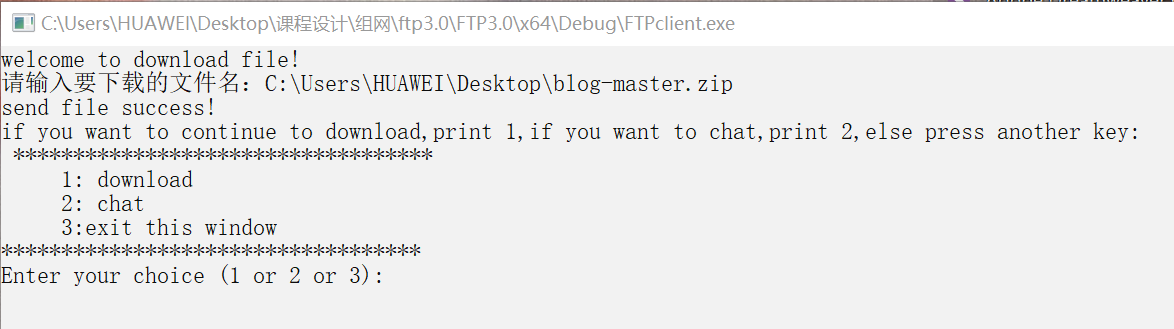


图5.1 发送文件成功图

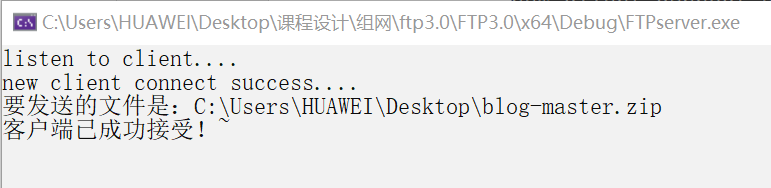


图5.2 服务端发送文件成功示意图

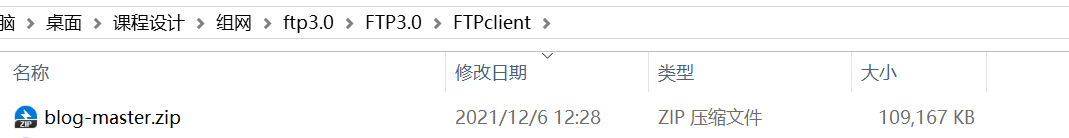


图5.3 发送成功的文件示意图

6、选择c,退出系统

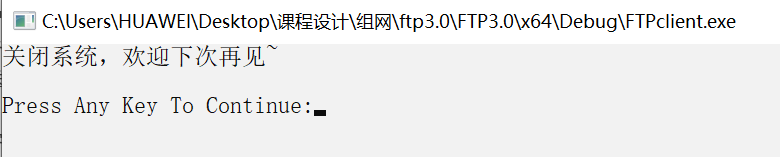


图6.1 客户端退出系统示意图

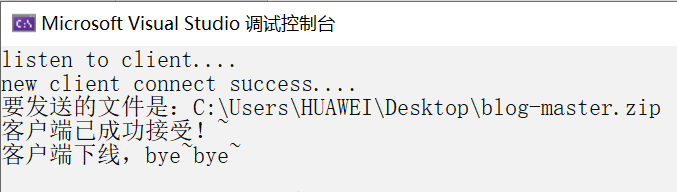


图6.2 服务端退出系统示意图

# 6、结论与展望

## 6.1结论

利用socket编程实现网上聊天和文件传输系统的基本已经实现。虽然时间短暂，但却是对着以学期学习计算机网路中理论知识和实践相结合的一次综合检验。

本课程设计参考了大量的socket编程的实例，但在网上的实例中Windows平台下用C语言写的实现聊天系统兼大文件传输系统的功能案例很少，由于C语言涉及底层，大部分都是基于c#、java、python等语言进行开发，或者在Linux系统环境下进行开发，因此在实际开发过程中遇到了很多问题，开发难度大进展较为缓慢，比如实现文件传输功能的第一个问题就是服务端如何根据用户发送的文件名对其进行检索；第二就是当待发送文件时超过包的大小如何对其进行拆包，客户端对收到的包如何拼接，第三是如何将聊天的功能和文件传输的功能进行有机的结合而不产生冲突等等，第四是如何处理发送信息缓存区溢出问题等等，在实际开发过程中，每一个环节都出现了很多问题，导致程序运行出现问题，而这些正是难点也是在做课程设计最大的突破点。

本次课程设计的完成得利于老师和同学点播和帮助，可以这么说，没有他们的帮助，此次课程设计并不会做到这个程度，特别感谢我的计算机网络的老师高军老师对此次课程设计的耐心指导和点播，以及操作系统老师毕坤老师，他所传授的线程同步的相关课程知识对自己实现并发通信功能提供了很多的灵感。

总而言之，此次课程设计收获满满，不仅是对我本学期所学知识的综合应用，也是对自己考验！

## 6.2 展望

通过这次组网的课程设计，收到了比理论课程还要大的知识和经验，但由于时间仓促，同时笔者也缺乏大量的实践经验，系统还有很多不尽人意的地方：比如1、客户端和服务端进入聊天室后，客户端发送信息的“send>”标志符会因为多次收到信息而发生错位2、该程序还还未实现图形化界面；3、代码的重用性不够高；4、一些细节的问题还没有解决。因此该软件需要大量的优化和升级；后期的项目也将着重围绕这几点进行改进。希望在以后的工作学习中能有进一步的提高。

# 参考文献

1. 谢希仁. 计算机网络[M]. 第七版. 电子工业出版社, 2017年1月.
2. c语言Plus. [socket/网络编程]C语言实现ftp文件传输服务器——超简单，一学就会[EB/OL]. [2021.12.5]. <https://www.bilibili.com/video/BV1tX4y1V7cd/?spm_id_from=333.788.recommend_more_video.14>.
3. c语言编程入门. C/C++基础教程：详细讲解socket网络编程[EB/OL]. [2021.9.25]. <https://www.bilibili.com/video/BV1m44y1B7UR?spm_id_from=333.999.0.0>
4. 龙征天. 网络Socket编程指南[EB/OL]. [2021.9.25]. https://blog.csdn.net/weixin\_43347376/article/details/105887260.