

**滨江学院**

课 程 实 践

SOCKET编程实现聊天程序

报

告

学年学期 2020-2021学年第一学期

课程名称 计算机网络实践

院 系 计算机学院

专 业 软件工程

姓名学号 毛济洲20102344050

指导教师 朱节中

**二Ｏ二一 年 一 月 一 日**

目录

[一、课程实践完成情况概述 - 3 -](#_Toc61098834)

[二、课程实践内容的实践报告 - 3 -](#_Toc61098835)

[实践目的 - 3 -](#_Toc61098836)

[实践内容及过程 - 3 -](#_Toc61098837)

[设计内容 - 3 -](#_Toc61098838)

[设计过程 - 3 -](#_Toc61098839)

[三、项目报告 - 4 -](#_Toc61098840)

[问题陈述 - 4 -](#_Toc61098841)

[背景知识 - 4 -](#_Toc61098842)

[WebSocket 简介 - 4 -](#_Toc61098843)

[TCP/IP、UDP - 4 -](#_Toc61098844)

[Socket - 4 -](#_Toc61098845)

[常用函数 - 4 -](#_Toc61098846)

[动机 - 5 -](#_Toc61098847)

[实现语言 - 6 -](#_Toc61098848)

[所作的工作 - 6 -](#_Toc61098849)

[系统描述 - 7 -](#_Toc61098850)

[源码 - 8 -](#_Toc61098851)

[服务器端： - 8 -](#_Toc61098852)

[客户端： - 10 -](#_Toc61098853)

[运行结果 - 12 -](#_Toc61098854)

[打算进行的工作或改进 - 12 -](#_Toc61098855)

[四、课程实践总结 - 12 -](#_Toc61098856)

# 一、课程实践完成情况概述

初步完成基于c++的SOCKET编程实现的聊天程序，使用Win32 Socket 函数实现聊天程序，能相互对发文本消息，由服务器端和客户端组成。

# 二、课程实践内容的实践报告

## 实践目的

（1）掌握网络应用程序的开发方法；

（2）掌握Client/ Server结构软件的设计与开发方法

（3）掌握Socket机制的工作原理

## 实践内容及过程

### 设计内容

使用Win32 Socket 函数实现聊天程序：能相互对发文本消息。

### 设计过程

1. 下载安装visual studio2019
2. 安装开发环境
3. 进行代码编写
4. 调试运行

# 三、项目报告

## 问题陈述

通过TCP,Socket实现多对一的局域网聊天室可以实现多个客户端连接服务器，服务器接收到信息就会把信息广播到所有的客户端

## 背景知识

### WebSocket 简介

WebSocket 是 HTML5 开始提供的一种在单个 TCP 连接上进行全双工通讯的协议。  
WebSocket 使得客户端和服务器之间的数据交换变得更加简单，允许服务端主动向客户端推送数据。在 WebSocket API 中，浏览器和服务器只需要完成一次握手，两者之间就直接可以创建持久性的连接，并进行双向数据传输。  
在 WebSocket API 中，浏览器和服务器只需要做一个握手的动作，然后，浏览器和服务器之间就形成了一条快速通道。两者之间就直接可以数据互相传送。

### TCP/IP、UDP

 TCP/IP（Transmission Control Protocol/Internet Protocol）即传输控制协议/网间协议，是一个工业标准的协议集，它是为广域网（WANs）设计的。  
         UDP（User Data Protocol，用户数据报协议）是与TCP相对应的协议。它是属于TCP/IP协议族中的一种。

### Socket

 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。

### 常用函数

#### 1. socket()

int socket(int family, int type, int protocol);

　　socket()打开一个网络通讯端口，如果成功的话，返回一个文件描述符，应用程序可以像读写文件一样用read/write在网络上收发数据，如果socket()调用出错则返回-1。对于IPv4，family参数指定为AF\_INET。对于TCP协议，type参数指定为SOCK\_STREAM，表示面向流的传输协议。如果是UDP协议，则type参数指定为SOCK\_DGRAM，表示面向数据报的传输协议。protocol参数的介绍从略，指定为0即可。

#### 2.   bind()

int bind(int sockfd, const struct sockaddr \*myaddr, socklen\_t addrlen);

　　服务器程序所监听的网络地址和端口号通常是固定不变的，客户端程序得知服务器程序的地址和端口号后就可以向服务器发起连接，因此服务器需要调用bind绑定一个固定的网络地址和端口号。bind()成功返回0，失败返回-1。bind()的作用是将参数sockfd和myaddr绑定在一起，使sockfd这个用于网络通讯的文件描述符监听myaddr所描述的地址和端口号。前面讲过，struct sockaddr \*是一个通用指针类型，myaddr参数实际上可以接受多种协议的sockaddr结构体，而它们的长度各不相同，所以需要第三个参数addrlen指定结构体的长度

#### 3． listen()

int listen(int sockfd, int backlog);// backlog取值0~5.

　　典型的服务器程序可以同时服务于多个客户端，当有客户端发起连接时，服务器调用的accept()返回并接受这个连接，如果有大量的客户端发起连接而服务器来不及处理，尚未accept的客户端就处于连接等待状态，listen()声明sockfd处于监听状态，并且最多允许有backlog个客户端处于连接待状态，如果接收到更多的连接请求就忽略。listen()成功返回0，失败返回-1。

#### 4.accept()

int accept(int sockfd, struct sockaddr \*cliaddr, socklen\_t \*addrlen);

　　三方握手完成后，服务器调用accept()接受连接，如果服务器调用accept()时还没有客户端的连接请求，就阻塞等待直到有客户端连接上来。cliaddr是一个传出参数，accept()返回时传出客户端的地址和端口号。addrlen参数是一个传入传出参数（value-result argument），传入的是调用者提供的缓冲区cliaddr的长度以避免缓冲区溢出问题，传出的是客户端地址结构体的实际长度（有可能没有占满调用者提供的缓冲区）。如果给cliaddr参数传NULL，表示不关心客户端的地址。

## 动机

1、对“聊天室”概要设计的阶段任务成果形成文档，以便阶段验收、评审，最终的文档验收。

2、对需求阶段的文档再次确认过程，对前一阶段需求没有做充分或错误的提出修改。

3、明确整个系统的功能框架和数据库结构，为下一阶段的详细设计、编码、和测试提供参考依据。

4、明确编码规范和命名规范，统一程序界面。

## 实现语言

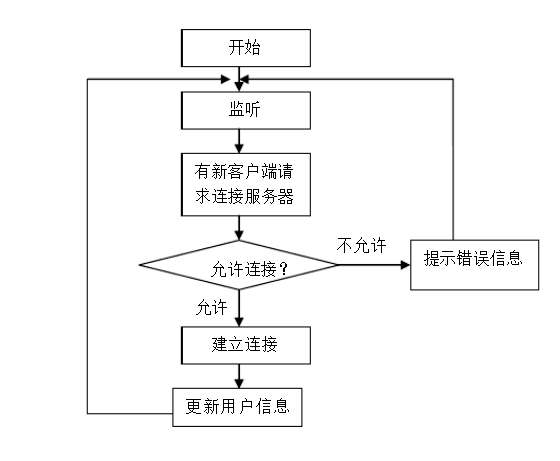
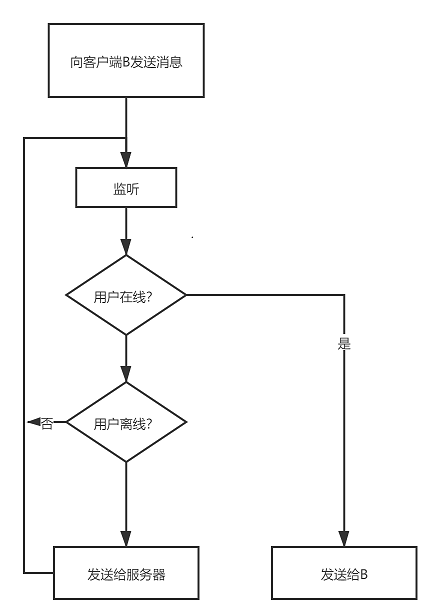
使用C++通过Visual Studio 2019进行win32开发

服务器操作系统：Windows10

客户端操作系统：Windows10

## 所作的工作

1. 进行项目规划
2. 对项目进行分析
3. 编写源代码
4. 进行调试、纠错
5. 生成exe文件



## 系统描述

在同一主机上、局域网内、互联网上进行文字聊天，windows操作系统上用TCP协议

客户端和服务器之间通信

多客户端通信：

1.服务器要接收多个客户端的连接

2.多线程来解决多客户端通信问题

2.1 有客户端连接到服务器，启动一个线程来和这个客户端通信

2.2 线程里：

2.2.1 接收客户端发来的数据

2.2.2 把数据广播给当前连接到服务器的所有客户端

## 源码

### 服务器端：

#include <stdio.h>

#include <windows.h>

#pragma comment(lib, "ws2\_32.lib")

SOCKADDR\_IN cAddr = { 0 };

int len = sizeof cAddr;

SOCKET clientSocket[1024];

int count = 0;

void tongxin(int idx) {

char buff[1024];

int r;

while (1) {

r = recv(clientSocket[idx], buff, 1023, NULL);

if (r > 0) {

buff[r] = 0;

printf("%d:%s\n", idx, buff);

//广播数据

for (int i = 0; i < count; i++) {

send(clientSocket[i], buff, strlen(buff), NULL);

}

}

}

}

int main() {

//1 请求协议版本

WSADATA wsaData;

WSAStartup(MAKEWORD(2, 2), &wsaData);

if (LOBYTE(wsaData.wVersion) != 2 ||

HIBYTE(wsaData.wVersion) != 2) {

printf("请求协议版本失败!\n");

return -1;

}

printf("请求协议成功!\n");

//2 创建socket

SOCKET serverSocket = socket(AF\_INET, SOCK\_STREAM, IPPROTO\_TCP);

if (SOCKET\_ERROR == serverSocket) {

printf("创建socket失败!\n");

WSACleanup();

return -2;

}

printf("创建socket成功!\n");

//3 创建协议地址族

SOCKADDR\_IN addr = { 0 };

addr.sin\_family = AF\_INET;//协议版本

addr.sin\_addr.S\_un.S\_addr = inet\_addr("10.2.17.225");//用自己的ip

addr.sin\_port = htons(10086);//0 - 65535 10000左右

//os内核 和其他程序 会占用掉一些端口

//4 绑定

int r = bind(serverSocket, (sockaddr\*)&addr, sizeof addr);

if (-1 == r) {

printf("bind失败!\n");

closesocket(serverSocket);

WSACleanup();

return -2;

}

printf("bind成功!\n");

//5 监听

r = listen(serverSocket, 10);

if (-1 == r) {

printf("listen失败!\n");

closesocket(serverSocket);

WSACleanup();

return -2;

}

printf("listen成功!\n");

//6 等待客户端连接 阻塞 尾生抱柱

//客户端协议地址族

while (1) {

clientSocket[count] = accept(serverSocket, (sockaddr\*)&cAddr, &len);

if (SOCKET\_ERROR == clientSocket[count]) {

printf("服务器宕机了!\n");

//8 关闭socket

closesocket(serverSocket);

//9 清除协议信息

WSACleanup();

return -2;

}

printf("有客户端连接到服务器了：%s!\n", inet\_ntoa(cAddr.sin\_addr));

CreateThread(NULL, NULL, (LPTHREAD\_START\_ROUTINE)tongxin,

(char\*)count, NULL, NULL);

count++;

}

return 0;

}

### 客户端：

#include <stdio.h>

#pragma warning(disable:4996)

//#include <windows.h>

#include <graphics.h> //easyX

#pragma comment(lib, "ws2\_32.lib")

SOCKET clientSocket;

HWND hWnd;

int count = 0;

void jieshou() {

char recvBuff[1024];

int r;

while (1) {

r = recv(clientSocket, recvBuff, 1023, NULL);

if (r > 0) {

recvBuff[r] = 0;

outtextxy(0, count \* 20, recvBuff);

count++;

}

}

}

int main() {

//初始化界面

hWnd = initgraph(300, 400, SHOWCONSOLE);

//1 请求协议版本

WSADATA wsaData;

WSAStartup(MAKEWORD(2, 2), &wsaData);

if (LOBYTE(wsaData.wVersion) != 2 ||

HIBYTE(wsaData.wVersion) != 2) {

printf("请求协议版本失败!\n");

return -1;

}

printf("请求协议成功!\n");

//2 创建socket

clientSocket = socket(AF\_INET, SOCK\_STREAM, IPPROTO\_TCP);

if (SOCKET\_ERROR == clientSocket) {

printf("创建socket失败!\n");

WSACleanup();

return -2;

}

printf("创建socket成功!\n");

//3 获取服务器协议地址族

SOCKADDR\_IN addr = { 0 };

addr.sin\_family = AF\_INET;//协议版本

addr.sin\_addr.S\_un.S\_addr = inet\_addr("10.2.17.225");//用自己的ip

addr.sin\_port = htons(10086);//0 - 65535 10000左右

//os内核 和其他程序 会占用掉一些端口 80 23

//4 连接服务器

int r = connect(clientSocket, (sockaddr\*)&addr, sizeof addr);

if (r == -1) {

printf("连接服务器失败!\n");

return -1;

}

printf("连接服务器成功!\n");

//5 通信

char buff[1024] = { 0 };

CreateThread(NULL, NULL,

(LPTHREAD\_START\_ROUTINE)jieshou,

NULL, NULL, NULL);

while (1) {

memset(buff, 0, 1024);

printf("你想说啥:");

scanf("%s", buff);

send(clientSocket, buff, strlen(buff), NULL);

}

return 0;

}

## 运行结果

截图里有图片

描述已自动生成

## 打算进行的工作或改进

1. 改进用户界面
2. 添加发送接收图片、语音功能
3. 优化代码

# 四、课程实践总结

通过这次聊天程序的编写，基本上了解了socket的CS构架原理，通信流程也清楚了，体会了TCP/IP连接的稳定性，学会了WinSock编程。  
今后的工作就是增加多线程处理，添加多个用户，实现真正的多人聊天。同时可以增加的功能就是文件互传，语音和视频聊天。  
还有一个问题就是优化代码，提高执行效率，使用更强的容错处理。