



# 计算机网络课程设计报告

系（院）： 计算机科学学院

专业班级： 计科 11402

姓 名： 程志平

学 号： 201403616

指导教师： 代江华

设计时间： 2017. 6. 19 - 2017. 6. 30

设计地点： 12 教 2 楼机房

# 目录

一、课程设计的目的和意义.....	1
二、设计题目与要求.....	1
2.1 设计要求.....	1
2.2 设计题目.....	2
三、设计内容.....	2
3.1 需求分析.....	2
3.2 概要设计.....	3
3.3 详细设计.....	4
四、运行结果及测试.....	5
五、设计成果以及心得.....	8
六、附录.....	9

# 基于 TCP 协议的通讯录

## 一. 课程设计的目的和意义

计算机网络课程设计的目的，是让学生更深入地掌握计算机网络的核心内容，实现理论与实践相结合。让学生用具体的实践成果，体现对理论知识的掌握程度。有利于学生提高计算机网络的实践能力，加深对计算机网络理论知识的理解。其基本目的是：

(1) 培养学生理论联系实际的设计思想，训练综合运用所学的基础理论知识，结合生产实际分析和解决网络应用中问题的能力，从而使基础理论知识得到巩固和加深。

(2) 学习掌握网络应用工程的一般设计过程和方法。

## 二. 设计题目与要求

### 2.1 设计要求

- (1) 编写程序，实现系统的基本功能，鼓励自行增加新功能；
- (2) 要有用户界面：要求至少采用文本菜单界面；鼓励采用图形菜单界面；
- (3) 写课程设计报告，内容包括：
  - 封面
  - 需求分析：以无歧义的陈述说明程序设计的任务，强调的是程序要做什么？给出功能模块图和流程图。同时明确规定：输入的形式和输出值的范围；输出的形式；程序所能够达到的功能；测试数据，包括正确的输入及其输出结果和含有错误的输入及其输出结果。
  - 概要设计：包括程序设计组成框图，程序中使用的存储结构设计说明（如果指定存储结构请写出该存储结构的定义）。
  - 详细设计：包括模块功能说明（如函数功能、入口及出口参数说明，函数调用关系描述等），每个模块的算法设计说明（可以是描述算法的流程图）。其中源程序要按照写程序的规则来编写，结构清晰，重点函数的重

点变量，重点功能部分要加上清晰的程序注释。

- 运行结果：包括典型的界面、输入和输出数据等；
- 总结：包括课程设计中遇到的问题，解决问题的过程及体会、收获、对课程设计的认识与思考等。
- 附录：包括主要程序清单，要有适当的注释，使程序容易阅读。

(4) 课程设计报告书写规范参见附录 II，不按照规范书写的，成绩不能评为“优”或“良”。

(5) 无论在校外、校内，都要严格遵守学校和所在单位的学习和劳动纪律、规章制度，学生有事离校必须请假。课程设计期间，无故缺席按旷课处理；缺席时间达四分之一以上者，其成绩按不及格处理。

## 2.2 设计题目

(1) 题目 2：基于 TCP 协议的通讯录

(2) 设计目标：

1. 了解 Socket 通信的原理，在此基础上编写一个基于 tcp 的通讯录程序；
2. 理解 TCP 通信原理；

(3) 基本原理

此 TCP 课程设计实现了基于 TCP 的客户/服务器通信程序，需要实现以下一些基本功能：

1. 客户端连接服务器；
2. 消息发送：客户端发送消息给服务器。
3. 消息接收：客户端接收到服务器发送给他的消息。
4. 可以有多个客户端同时连接
5. 回复功能：根据用户发送的消息内容，给予相应功能的回复。

# 三、设计内容

## 3.1 需求分析

(1) 通信功能：客户端和服务端连接于与通信

1. 客户端发送请求，并接收服务器数据
2. 服务器接收请求，并发送相应数据

(2) 客户端能够对存储在服务器端的通讯录中的各项信息进行浏览，添加，删除，等操作

1. 浏览：客户端发送浏览联系人请求，并将服务器发送的联系人信息显示在界

面上以方便浏览

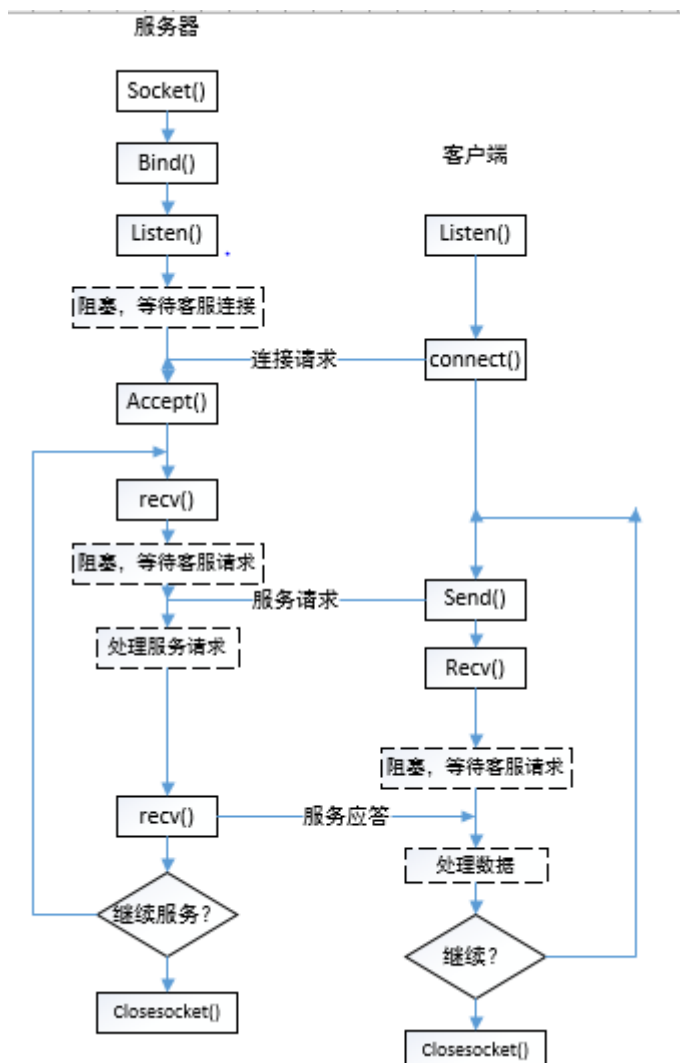
2. 添加客户端手动输入联系人的基本信息,通过添加请求能将联系人加入服务器端的通讯录,服务器端能稳定的将客户端 ip 加入通讯录的 ip 字段。
3. 修改
4. 删除:能够通过输入联系人姓名或电话来删除联系人,同时弹出对话框给予警告确认。
- (3) 退出系统功能:能够退出系统并保存数据。
- (4) 能够对输入的联系人信息,如输入值的类型,大小,字符串长度等进行正确性的检查,对不合法的输入提示错误类型,并等待重新输入。
- (5) 具有一定的健壮性,不会因为用户的操作或输入错误导致程序运行错误。

## 3.2 概要设计

### (1) 模块划分

- 1.第一层:本地管理模块。
- 2.第二层:网络管理模块。
- 3.第三层:客户端模块。

### (2) 处理流程



### 3.3 详细设计

…。(每个模块的功能说明和算法设计说明: 每个模块的功能说明(函数功能、入口及出口参数说明, 函数调用关系描述); 每个模块的算法设计说明(算法流程图); 原则上不贴代码(只贴关键部分: 如题目 2 的文件操作, IP 获取)

模块一: 本地管理通讯录信息; 可以进行本地的管理, 添加删除操作。其中有主界面的实现, 以及存储结构体的定义, 有姓名, 电话, 性别, 部门和 ip 地址, 当本地没有存储的文本时, 立即在本地创建一个名为 contact 的文本文件用来存储通讯录信息, 所以程序一开始先将文本文件读取出来, 如不存在时创建, 存在时读取并保存在结构体数组中去, 用来共系统和用户调用系信息。用户进入主界面时, 按照需要进行的操作来选择序号, server 中有对所按的数字键进行执行对应的操作, 具体用的 switch 语句, 显示, 删除, 添加都有特定的函数实现他的功能, 还有一个就是遍历的函数, 用来辅助实现显示和添加功能的, 退出程序时, 保存信息到文本文件中去, 防止信息的丢失。本模块的头文件附加如下:

```
Status AddInformation(SOCKET *server = NULL, Status SEND(SOCKET *server, Contacts
cont) = 0);//添加联系人
Status Send(SOCKET *client, Contacts cont);//后台数据发送
Status AllAddress(SOCKET *client = NULL, Status SEND(SOCKET *client, Contacts) = 0);//
后台数据遍历
Status Del(char *name, SOCKET *client = NULL, Status SEND(SOCKET *client, Contacts)
= 0);
Status Search(Contacts &cont, SOCKET *client, Status SEND(SOCKET *client, Contacts)
= 0);//后台数据的查找
Status Insert(Contacts cont);//后台数据的插入
```

模块二: 用户端的实现; 用户端有显示添加功能。显示时用户和服务器有同样的函数, 只是实现了不同的结果, 在于两个端口有俩个不同的参数, 使得在执行显示和添加时, 两个端口显示的结果不同, 用户端显示的是完整的结果而服务器端只有用户进行的操作号, 两个端口的区别, 代码如下:

```
if (ret > 0 && atoi(revData) > 0 && isdigit(revData[0]))
{
    revData[ret] = 0x00;
    printf("客户端请求的指令为%s\n", revData);
    if (atoi(&revData[0]) == 1)
    {
        memset(sendData, '0', MAX_DATA_LEN);
        strcpy(sendData, "1\n");
        send(s_client, sendData, strlen(sendData), 0);
        int retn = AllAddress(&s_client, &Send);
        if (retn == NO_ADDRESSEXIST)
        {
            memset(sendData, '0', MAX_DATA_LEN);
```

```

        Contacts tmp1 = { "",0,0,"", "",0 };
        send(s_client, (char *)&tmp1, sizeof(tmp1), 0);
        strcpy(sendData, "0 服务端没有通讯录!\n");
        send(s_client, sendData, strlen(sendData), 0);
    }
    if (retn == OK)
    {
        memset(sendData, '0', MAX_DATA_LEN);
        Contacts tmp1 = { "",0,0,"", "",0 };
        send(s_client, (char *)&tmp1, sizeof(tmp1), 0);
        memset(sendData, '0', MAX_DATA_LEN);
        strcpy(sendData, "0 通讯录发送完毕!\n");
        send(s_client, sendData, strlen(sendData), 0);
    }
}

```

这样实现了不同的结果，避免增加服务器端的任务。本模块的实验头文件如下：

```
Status AddInformation(SOCKET *server = NULL, Status SEND(SOCKET *server, Contacts
cont) = 0);//添加联系人
```

```
Status Send(SOCKET *client,Contacts cont);//后台数据发送
```

```
Status AllAddress(SOCKET *client = NULL ,Status SEND(SOCKET *client,Contacts) = 0);//
后台数据遍历
```

```
Status Del(char *name, SOCKET *client = NULL, Status SEND(SOCKET *client, Contacts)
= 0);
```

```
Status Search(Contacts &cont, SOCKET *client, Status SEND(SOCKET *client, Contacts)
= 0);//后台数据的查找
```

```
Status Insert(Contacts cont);//后台数据的插入
```

## 四、运行结果及测试

### (1) 主界面展示

服务器端主界面见图 1



图 1 客户端主界面

客户段主界面见图 2

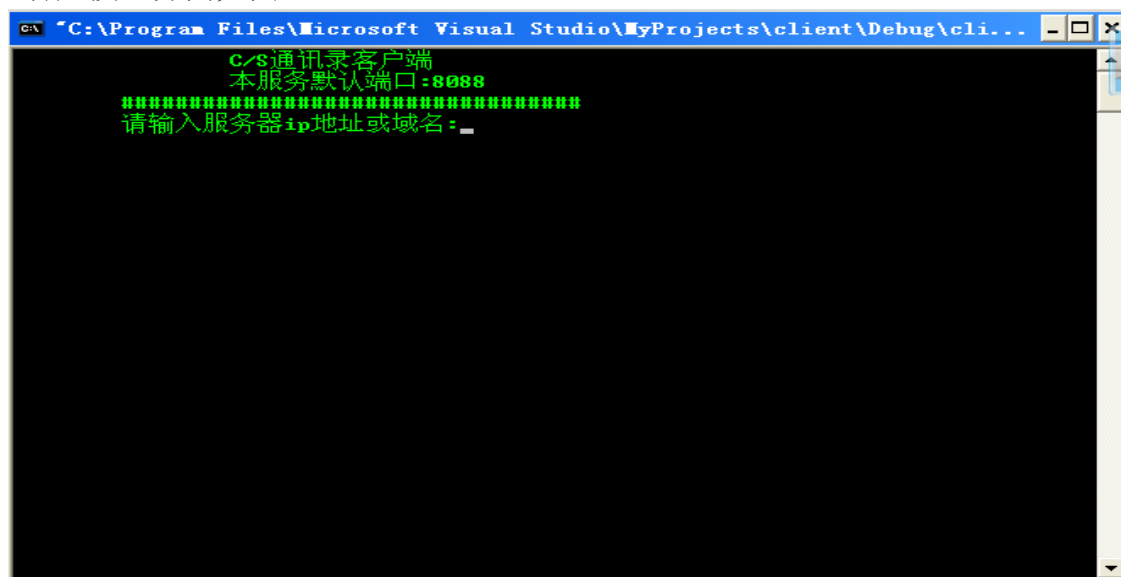


图 2 客户端主界面

(2) 功能运行结果（输入，中间数据和输出结果）

服务器端进行本地的显示，添加和删除见图 3.图 4 图 5

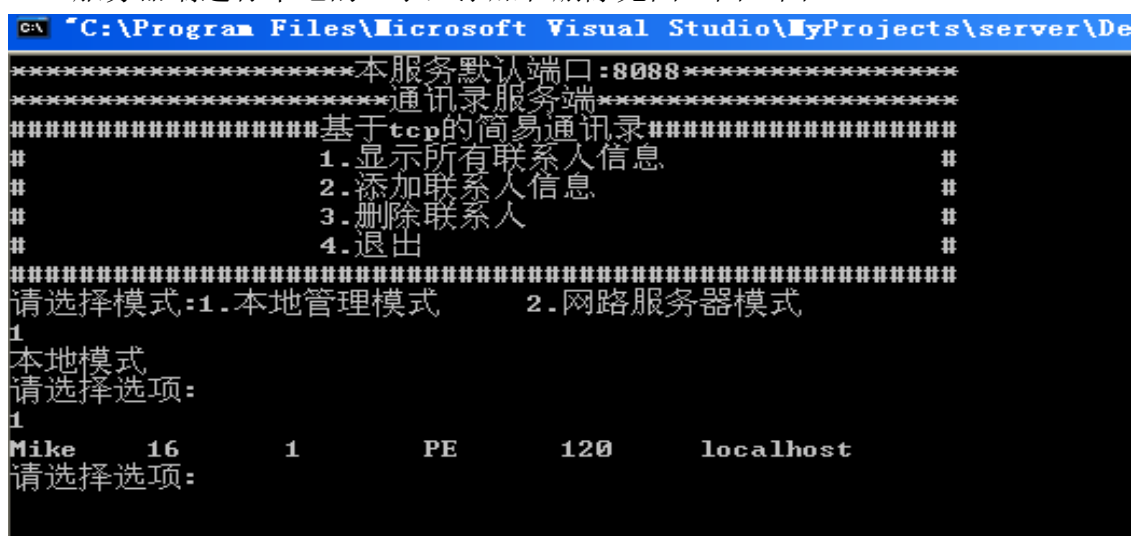


图 3 服务器端显示通讯录信息



```

C:\Program Files\Microsoft Visual Studio\MyProjects\server\Debug\ser...
*****本服务默认端口:8088*****
*****通讯录服务端*****
*****基于tcp的简易通讯录*****
#
#       1.显示所有联系人信息       #
#       2.添加联系人信息           #
#       3.删除联系人               #
#       4.退出                       #
#
*****
请选择模式:1.本地管理模式      2.网路服务器模式
1
本地模式
请选择选项:
1
Mike    16      1      PE      120      localhost
请选择选项:
2
请依次输入姓名,年龄,性别<男:1;女:0>,单位,电话<以回车隔开>
Jones 25 1 IT 119
Jones    25      男      IT      119      localhost
*****信息录入完毕*****
请选择选项:

```

图 4 服务器端添加信息

```

C:\Program Files\Microsoft Visual Studio\MyProjects\server\Debug\ser...
*****通讯录服务端*****
*****基于tcp的简易通讯录*****
#
#       1.显示所有联系人信息       #
#       2.添加联系人信息           #
#       3.删除联系人               #
#       4.退出                       #
#
*****
请选择模式:1.本地管理模式      2.网路服务器模式
1
本地模式
请选择选项:
1
Mike    16      1      PE      120      localhost
请选择选项:
2
请依次输入姓名,年龄,性别<男:1;女:0>,单位,电话<以回车隔开>
Jones 25 1 IT 119
Jones    25      男      IT      119      localhost
*****信息录入完毕*****
请选择选项:
3
请输入你要删除联系人的名字:Jones
删除的信息为:Jones    25      1      IT      119      localhost
请选择选项:

```

图 5 服务器端删除信息

客户端进行显示和添加操作见图 6, 图 7

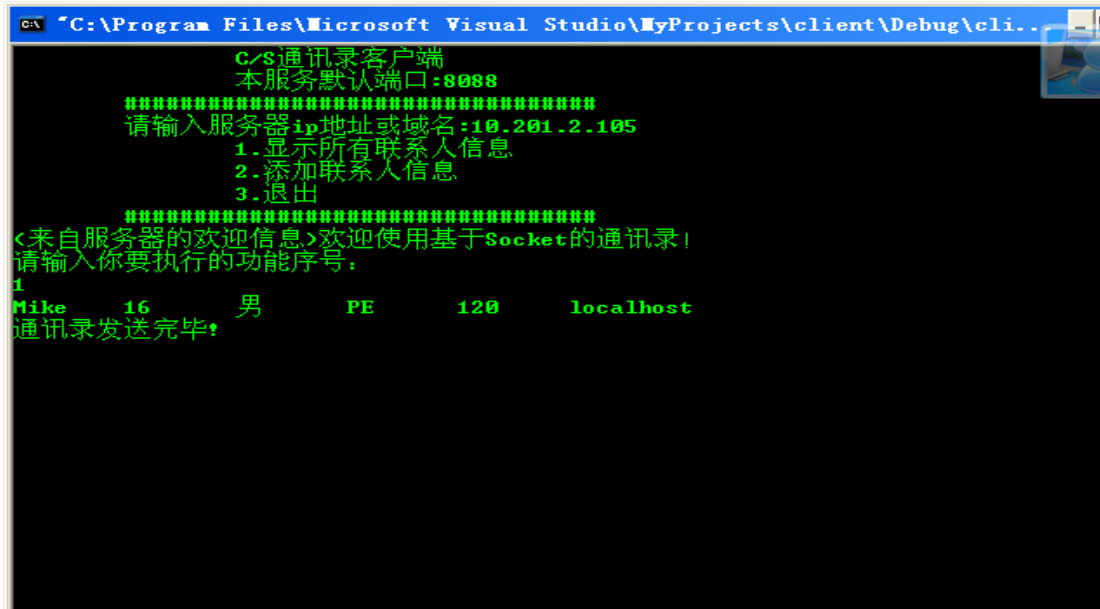


图 6 客户端显示信息

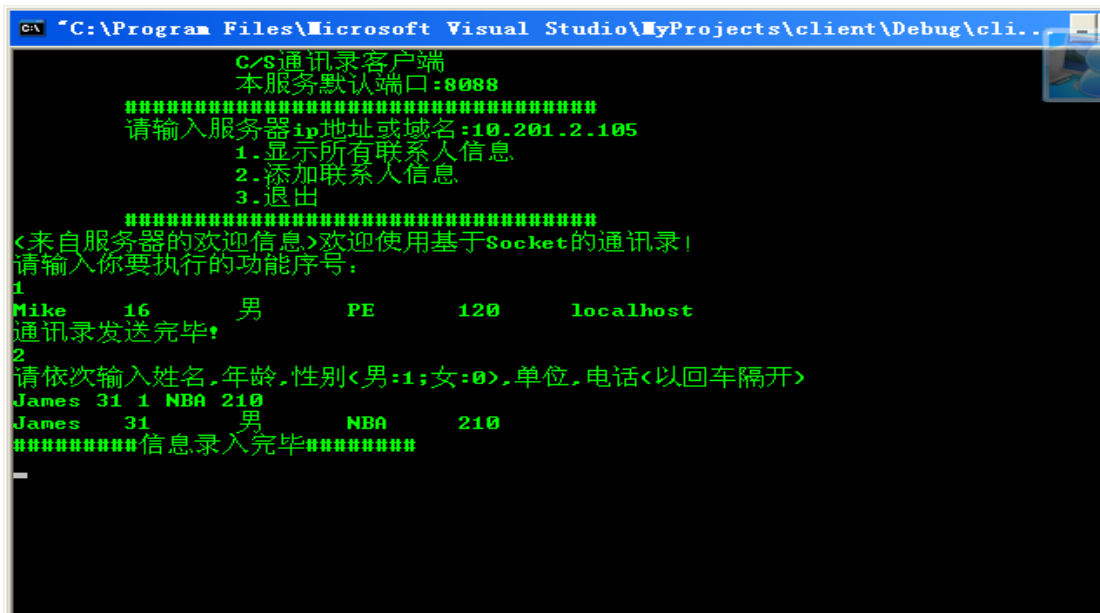


图 7 客户端添加信息

## 五、设计成果以及心得

### 1、已完成功能总结:将任务分为服务器端和客户端

#### ①服务器端有本地管理模式和网络服务器模式

I. 第一个就是本地管理模式，在此模式下可以进行通讯录的显示信息，添加信息，删除信息，退出程序。删除信息只有在服务器端可以操作，在客户端模式是看不见此操作的。

II. 第二个就是网络服务器模式，在此模式下，客户端输入自己的 ip 地址与服务器通讯，并在服务器端显示客户进行的操作号。

#### ②客户端可以进行通讯录的显示，添加。进行显示时从本地读取文件中的内容

显示在屏幕上，删除信息时输入要删除信息的名字，删除时有确认提示，避免用户过失删除记录。

2、主要运用的技术：使用到的技术有 C/S 结构，Windows Socket 编程，TCP 套接字编程

①C/S: Client 在需要服务时间向 server 提出申请；Server 等待 client 提出请求并给予相应的回应；Server 始终运行，监听网络接口；受到 client 请求启动服务进程相应客户，同时继续监听服务窗口，保证后续的 client 也能及时得到服务。

②Socket：一个本地的应用程序创建的，操作系统控制的接口。通过该接口应用程序可以从另一个应用进程发送或者接受消息，常用接口：

- I. 创建套接字：socket()
- II. 绑定本地地址：bind()
- III. 侦听连接：listen()
- IV. 建立套接字连接：accept(), connect()
- V. 面向连接的数据传输：send(), recv()
- VI. 关闭套接字：closesocket()

③TCP 套接字编程：服务器做好准备；服务器与服务器的联系；服务器收到客户端连接请求后，创建新的 socket 用来与客户端通信；TCP 协议提供了客户端和服务器之间可靠的按次序的字节流传送。

3、在课程设计中遇到的问题有：Socket 套接字编程流程的理解：在网上搜索资料和老师给予的 ppt 学习后，大概了解了 Socket 的基本流程，可以进行后面的工作；读取文件时对文件的几种操作类型区分，了解资料后对每个操作类型都有知道了他的作用；在客户端连接了服务器端后进行同一个操作名时，要求在两个端由不同的显示，为了完成此功能询问了通讯和网络论坛后，大致的解决了这个棘手的问题。

4、通过本次实验，让我更加深入的了解套接字编程实现的细节，对 TCP 协议原理也有了更深刻的理解。同时对 C 编程的熟悉和巩固流程，在实验编码中，遇到了不少错误，但都不断改进调试最后成功，还有 hostname 只用本地 ip 最直接，当然，也可以换别的主机的地址，在今后的学习和工作中，不能放下这个技能，只有每天加以复习和学习才能把新学习的东西掌握，作为一个学生，要保持一个学习的心，当今社会更新很快，只有保持学习才能进步，学如逆水行舟，不进则退。

## 六、附录

主要代码清单（包含适当的注释）

1. Funciton. cpp //功能模块

```
#include"function.h"
```

```
#include "string.h"
```

```
#include "datagram.h"
```

```
Status AddInformation(SOCKET *server,Status SEND(SOCKET *server,Contacts cont))//
```

```
添加联系人
```

```
{
```

```

int tmp;
char name[20];
int age;
bool sex;
char unit[30];
char tel[15];
char ip[20];
printf("请依次输入姓名,年龄,性别(男:1;女:0),单位,电话(以回车隔开)\n");
scanf("%s", name);
scanf("%d", &age);
scanf("%d", &tmp);
if (tmp == 1)
    sex = 1;
else
    sex = 0;
scanf("%s", unit);
scanf("%s", tel);
strcpy(ip, "localhost");
Contacts cont;
memcpy(cont.name, &name, sizeof(name));
cont.age = age;
cont.sex = sex;
memcpy(cont.unit, &unit, sizeof(unit));
memcpy(cont.tel, &tel, sizeof(tel));
memcpy(cont.ip, &ip, sizeof(ip));
if (SEND == 0)
{
    //##### 调试用#####
    printf("%s\t%d\t%s\t%s\t%s\t%s\n", cont.name, cont.age, cont.sex ? "男" : "
女", cont.unit, cont.tel, cont.ip);
    //#####
}
else
    SEND(server, cont);
Insert(cont);
printf("#####信息录入完毕#####\n");
return OK;
}

```

```

Status AllAddress(SOCKET *client, Status SEND(SOCKET *client, Contacts))//遍历联系人
{
    FILE *filein = fopen("contacts.dat", "r");
    Contacts tmp1;
    if (!filein)
    {
        return NO_ADDRESSEXIST;
    }
}

```

```

    }
    while (!feof(filein))
    {
        int ret = fread(&tmp1, sizeof(Contacts), 1, filein);
        if (ret>0)
        {
            if (SEND == 0)
            {
                //printf("%d\n", ret);
                printf("%s\t%d\t%d\t%s\t%s\t%s\n",    tmp1.name,    tmp1.age,
tmp1.sex, tmp1.unit, tmp1.tel, tmp1.ip);
            }
            else
                SEND(client,tmp1);
        }
    }
    fclose(filein);
    return OK;
}

```

Status Del(char \*name, SOCKET \*client, Status SEND(SOCKET \*client, Contacts))//删除联系人

```

{

    Contacts tmp,tmp1;
    int ret;
    //备份通讯录
    FILE *filein = fopen("contacts.dat", "rb");
    if (!filein)
        return NO_ADDRESSEXIST;
    FILE *fileout = fopen("contacts.dat.bak", "wb");
    while (!feof(filein))
    {
        if (fread(&tmp, sizeof(Contacts), 1, filein) > 0)
            fwrite(&tmp, sizeof(Contacts), 1, fileout);
    }
    fclose(filein);
    fclose(fileout);
    //#####end#####
    filein = fopen("contacts.dat.bak", "rb");
    if (!filein)
        return NO_ADDRESSEXIST;
    fileout = fopen("contacts.dat", "wb");
    while (!feof(filein))
    {
        ret = fread(&tmp1, sizeof(Contacts), 1, filein);

```

```

        if (ret>0)
        {
            if (SEND == 0 && !strcmp(tmp1.name, name))
            {
                printf("删除的信息为:%s\t%d\t%d\t%s\t%s\t%s\n", tmp1.name,
tmp1.age, tmp1.sex, tmp1.unit, tmp1.tel, tmp1.ip);
                continue;
            }
            else if((SEND != 0 && !strcmp(tmp1.name, name)))
            {
                Send(client, tmp1);
                continue;
            }
            fwrite(&tmp1, sizeof(Contacts), 1, fileout);
        }
    }
    fclose(filein);
    fclose(fileout);
    return OK;
}

```

Status Search(Contacts &cont, SOCKET \*client, Status SEND(SOCKET \*client, Contacts))//后台数据的查找

```

{
    Contacts tmp1;
    FILE *filein = fopen("contacts.dat", "rb");
    if(!filein)
        return NO_ADDRESSEXIST;
    while(!feof(filein))
    {
        int ret = fread(&tmp1, sizeof(Contacts), 1, filein);
        if (ret>0)
        {
            cont = tmp1;
            if (SEND == 0 && !strcmp(tmp1.name, cont.name))
            {
                printf("%s\t%d\t%d\t%s\t%s\t%s\n",    tmp1.name,    tmp1.age,
tmp1.sex, tmp1.unit, tmp1.tel, tmp1.ip);
            }
            else if ((SEND != 0 && !strcmp(tmp1.name, cont.name)))
                Send(client, tmp1);
        }
    }
    struct Contacts tmp = {"0", 0, true, "", "", 0};
    cont = tmp;
    fclose(filein);
}

```

```

        return NO_SUCHNAME;
    }
    Status Insert(Contacts cont)//后台数据的插入
    {
        FILE *fileout = fopen("contacts.dat", "ab");
        if (!fileout)
            return NO_ADDRESSEXIST;
        fwrite(&cont, sizeof(Contacts), 1, fileout);
        fclose(fileout);
        return OK;
    }

    Status Send(SOCKET *client, Contacts cont)//发送联系人至 client
    {
        send(*client, (char *)&cont, sizeof(cont), 0);
        return OK;
    }

```

2.net.cpp //连接模块  
#include "function.h"

```

int Net()
{
    WORD sockVersion = MAKEWORD(2, 2);//初始化 WSA
    WSADATA wsaData;
    if (WSAStartup(sockVersion, &wsaData) != 0)
    {
        return 0;
    }
    SOCKET server = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP);//创建套接字
    if (server == INVALID_SOCKET)
    {
        printf("socket error!");
        return 0;
    }
    //绑定通信端口
    sockaddr_in sin;
    sin.sin_family = AF_INET;
    sin.sin_port = htons(8088);
    sin.sin_addr.S_un.S_addr = INADDR_ANY;
    if (bind(server, (LPSOCKADDR)&sin, sizeof(sin)) == SOCKET_ERROR)
    {
        printf("bind error!");
    }
    //开始监听
    if (listen(server, 5) == SOCKET_ERROR)

```

```

{
    printf("listen error!");
    return 0;
}
printf("等待消息接收!\n");
//等待接收数据
SOCKET s_client;
sockaddr_in destaddr;
int nAddrlen = sizeof(destaddr);
char *sendData;
char *revData;

while (1)
{
    s_client = accept(server, (SOCKADDR *)&destaddr, &nAddrlen);
    if (s_client == INVALID_SOCKET)
    {
        printf("accpet error!\n");
        continue;
    }
    sendData = (char *)malloc(MAX_DATA_LEN);
    memset(sendData, '0', MAX_DATA_LEN);
    strcpy(sendData, "0(来自服务器的欢迎信息)欢迎使用基于 Socket 的通讯
录! \n 请输入你要执行的功能序号: \n");
    send(s_client, sendData, strlen(sendData), 0);
    printf("接收到一个连接:%s\n", inet_ntoa(destaddr.sin_addr));
    revData = (char *)malloc(MAX_DATA_LEN);
    while (1)
    {
        memset(revData, '0', MAX_DATA_LEN);
        int ret = recv(s_client, revData, MAX_DATA_LEN, 0);
        revData[ret] = 0x00;
        if (ret > 0 && atoi(revData) > 0 && isdigit(revData[0]))
        {
            revData[ret] = 0x00;
            printf("客户端请求的指令为%s\n", revData);
            if (atoi(&revData[0]) == 1)
            {
                memset(sendData, '0', MAX_DATA_LEN);
                strcpy(sendData, "1\n");
                send(s_client, sendData, strlen(sendData), 0);
                int retn = AllAddress(&s_client, &Send);
                if (retn == NO_ADDRESSEXIST)
                {
                    memset(sendData, '0', MAX_DATA_LEN);
                    Contacts tmp1 = { "", 0, 0, "", "", 0 };

```



```

        send(s_client, (char *)&tmp1, sizeof(tmp1), 0);
        strcpy(sendData, "0 服务端没有通讯录!\n");
        send(s_client, sendData, strlen(sendData), 0);
    }
    if (retn == OK)
    {
        memset(sendData, '0', MAX_DATA_LEN);
        Contacts tmp1 = { "", 0, 0, "", "", 0 };
        send(s_client, (char *)&tmp1, sizeof(tmp1), 0);
        memset(sendData, '0', MAX_DATA_LEN);
        strcpy(sendData, "0 通讯录发送完毕!\n");
        send(s_client, sendData, strlen(sendData), 0);
    }
}
if (atoi(&revData[0]) == 2)
{
    Contacts cont;
    memset(sendData, '0', MAX_DATA_LEN);
    strcpy(sendData, "2 请录入信息:\n");
    send(s_client, sendData, strlen(sendData), 0);
    memset(revData, '0', MAX_DATA_LEN);
    while ((ret = recv(s_client, (char *)&cont, MAX_DATA_LEN, 0)) < 0);
    if (ret > 0)
    {
        strcpy(cont.ip, inet_ntoa(destaddr.sin_addr));
        Insert(cont);
    }
}
if (atoi(&revData[0]) == 3)
{
    memset(sendData, '0', MAX_DATA_LEN);
    strcpy(sendData, "0(服务端确认)客户端主动关闭通信!\n");
    printf("%s", &sendData[1]);
    send(s_client, sendData, strlen(sendData), 0);
    break;
}
//信息确认接收
char *tmp;
tmp = (char*)malloc(MAX_DATA_LEN);
recv(s_client, tmp, MAX_DATA_LEN, 0);
free(tmp);
//###end###
continue;
}
else if (ret > 0)
{

```

```

        revData[ret] = 0x00;
        printf("%s\n", revData);
        memset(sendData, '0', MAX_DATA_LEN);
        strcpy(sendData, "0 你好!\n");
        send(s_client, sendData, strlen(sendData), 0);
    }
}
closesocket(s_client);
}
closesocket(server);
if (!revData)
    free(revData);
if (sendData)
    free(sendData);
WSACleanup();
return 0;
}

```

### 3. main.cpp 服务器端主界面

```

int main()
{
    char no[2];
    printf("*****本服务默认端口:8088*****\n");
    printf("*****通讯录服务端*****\n");
    printf("#####基于 tcp 的简易通讯录#####\n");
    printf("#          1.显示所有联系人信息          #\n");
    printf("#          2.添加联系人信息              #\n");
    printf("#          3.删除联系人                  #\n");
    printf("#          4.退出                        #\n");
    printf("#####\n");
    printf("请选择模式:1.本地管理模式      2.网路服务器模式\n");
}

```

### 3.client.cpp//客户端界面

```

#include <winsock2.h>
#include <stdio.h>

#include "../server/datagram.h"
#define MAX_DATA_LEN 1024
char destip[20];
struct hostent *hp = NULL;

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

int main()

```

```

{
    system("color 0a");
    printf("\t\tC/S 通讯录客户端\n");
    printf("\t\t本服务默认端口:8088\n");
    printf("\t\t#####\n");

    printf("\t 请输入服务器 ip 地址或域名:");
    scanf("%s",destip);

    WORD sockVersion = MAKEWORD(2, 2);
    WSADATA data;
    if (WSAStartup(sockVersion, &data) != 0)
    {
        return 0;
    }

    SOCKET c_server = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP);
    if (c_server == INVALID_SOCKET)
    {
        printf("invalid socket !");
        return 0;
    }

    sockaddr_in serAddr;
    if ((hp = gethostbyname(destip)) != NULL)
    {
        memcpy(&(serAddr.sin_addr),hp->h_addr,hp->h_length);
        serAddr.sin_family = hp->h_addrtype;
    }
    else
    {
        printf("请输入正确的地址!");
        Sleep(1000);
        return 0;
    }

    //serAddr.sin_family = AF_INET;
    serAddr.sin_port = htons(8088);
    //serAddr.sin_addr.S_un.S_addr = inet_addr(destip);
    if (connect(c_server, (sockaddr *)&serAddr, sizeof(serAddr)) == SOCKET_ERROR)
    {
        printf("connect error !");
        closesocket(c_server);
        return 0;
    }
}

```

```

printf("\t\t1.显示所有联系人信息\n");
printf("\t\t2.添加联系人信息\n");
printf("\t\t3.退出\n");
printf("\t#####\n");

char *sendData;
char *recData;

Sleep(1000);
recData = (char *)malloc(MAX_DATA_LEN);
sendData = (char *)malloc(MAX_DATA_LEN);
memset(recData, 0, sizeof(recData));
int ret = recv(c_server, recData, MAX_DATA_LEN, 0);
if (ret > 0)
{
    recData[ret] = 0x00;
    printf(&recData[1]);
}
while (1)
{
    memset(sendData, 0, sizeof(sendData));
    scanf("%s", sendData);
    send(c_server, sendData, strlen(sendData), 0);
    memset(recData, 0, sizeof(sendData));
    int ret = 0;
    ret = recv(c_server, recData, MAX_DATA_LEN, 0);
    if (ret > 0 && atoi(&recData[0]) == 0)
    {
        recData[ret] = 0x00;
        printf(&recData[1]);
    }
    if (ret > 0 && atoi(&recData[0]) == 1)
    {
        Contacts tmp;
        ret = 0;
        while (ret = recv(c_server, (char *)&tmp, sizeof(Contacts), 0) > 0)
        {
            if (strcmp(tmp.name, ""))
            {
                printf("%s\t%d\t%s\t%s\t%s\t%s\n",    tmp.name,    tmp.age,
tmp.sex?"男":"女", tmp.unit, tmp.tel, tmp.ip);
            }
            else
                break;
        }
        memset(recData, 0, MAX_DATA_LEN);
    }
}

```

```

ret = recv(c_server, recData, MAX_DATA_LEN, 0);
if (ret > 0)
{
    recData[ret] = 0x00;
    printf(&recData[1]);
}
}
if (ret > 0 && atoi(&recData[0]) == 2)
{
    int tmp;
    char name[20];
    int age;
    bool sex;
    char unit[30];
    char tel[15];
    int ip;
    printf("请依次输入姓名,年龄,性别(男:1;女:0),单位,电话(以回车隔
开)\n");
    scanf("%s", name);
    scanf("%d", &age);
    scanf("%d", &tmp);
    if (tmp == 1)
        sex = 1;
    else
        sex = 0;
    scanf("%s", unit);
    scanf("%s", tel);
    ip = 0;
    Contacts cont;
    memcpy(cont.name, &name, sizeof(name));
    cont.age = age;
    cont.sex = sex;
    memcpy(cont.unit, &unit, sizeof(unit));
    memcpy(cont.tel, &tel, sizeof(tel));
    //#####调试用#####
    printf("%s\t%d\t%s\t%s\t%s\n", cont.name, cont.age, cont.sex ? "男" : "
女", cont.unit, cont.tel);
    //#####
    send(c_server, (char *)&cont, sizeof(Contacts), 0);
    printf("#####信息录入完毕#####\n");
}
if (ret > 0 && atoi(&recData[0]) == 3)
{
    memset(sendData, 0, MAX_DATA_LEN);
    strcpy(sendData, "3\n");
    send(c_server, sendData, sizeof(sendData), 0);
}

```

```

        memset(recData, 0, MAX_DATA_LEN);
        recData[recv(c_server, recData, MAX_DATA_LEN, 0)] = 0xff;
        printf("%s", recData);
        break;
    }
    //###发送确认接收消息###
    memset(sendData, 0, MAX_DATA_LEN);
    strcpy(sendData, "OK\n");
    send(c_server, sendData, strlen(sendData), 0);
    //#####
}
closesocket(c_server);
WSACleanup();

if (!sendData)
    free(sendData);
if (!recData)
    free(recData);

return 0;
}

```

指导老师意见：

成绩：\_\_\_\_\_

教师签名：\_\_\_\_\_

年 月 日