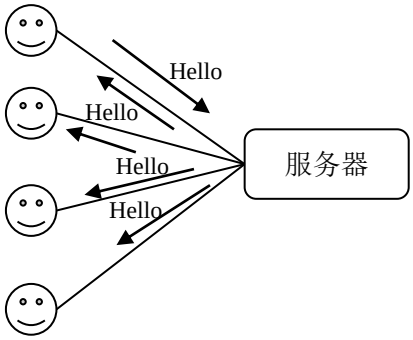
实验报告

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

【实验题目】**多人聊天编程实验**

【实验目的】掌握套节字的多线程编程方法。

【实验介绍】利用客户/服务器(Client/Sever 或 CS)模式实现一个多人聊天(群聊)程序。其功能是每个客户发送给服务器的消息都会传送给所有的客户端。



【实验环境】

(1)采用 Windows VC++控制台编程，建立新项目的方法：选择菜单“文件/新建项目/Win32 控制台应用程序/空项目”，然后增加 CPP 空文件，拷贝源程序。集成环境使用 VS2012 或更高版本。

(2) 采用 linux 编程

【参考资料】

- https://docs.microsoft.com/en-us/windows/desktop/WinSock/getting-started-with-winsock （套接字）
- https://www.cnblogs.com/hgwang/p/6074038.html （套接字）
- https://www.jb51.net/article/37410.htm （字符串）
- https://docs.microsoft.com/en-us/cpp/c-runtime-library/stream-i-o?view=vs-2017 （字符串）
- https://docs.microsoft.com/en-us/cpp/c-runtime-library/reference/crt-alphabetical-function-reference?view=vs-2017#s （字符串）
- http://www.runoob.com/cprogramming/ （字符串）

【注意事项】

依步骤完成以下实验内容，尽量多完成一些步骤。把出现的问题、解决方法和一些思考和体会写在**实验体会**部分；对典型的运行情况的客户和服务器控制台进行截屏并放在**实验结果**部分；截屏用按键(Ctrl+Alt+PrintScreen)单独截取控制台窗口。截屏应尽量 windows 绘图程序缩小尺寸(能看清就行)后粘入。

端口号采用 50500

字符串可以采用函数 scanf 或 gets_s 输入。

【参考资料】

1、 例程“_beginthreadex” （创建线程）

2、 例程“TCPServer 和 TCPClient” （传送服务器时间）

3、 课件“套接字并发编程.PDF”

4、 Chat 实验的课件

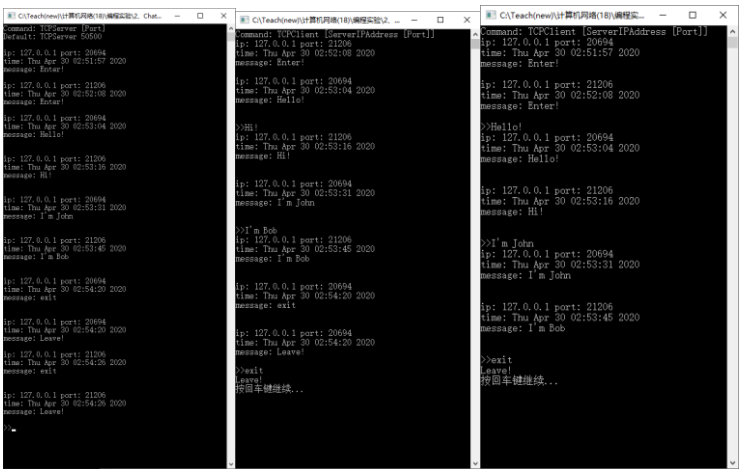
【实验内容】

先阅读课件“套接字并发编程.PDF”。重点是读懂课件中“chat 并发编程(服务器)”和“chat 并

发编程(客户端)”的流程图。 然后，完成下面步骤（截屏要同时显示服务器和至少两个客户端）：

注意保存每一步完成后的源码。

第 4 步完成后的参考截屏：



服务器　　客户端 1　　客户端 2

(1)编写多人聊天程序，要求客户端和服务器都采用多线程方式进行编程。每个客户端都采用 TCP 协议连接服务器并保持连接。服务器同时与所有客户端建立和保持连接。每个客户端输入的消息都会通过服务器转发给所有客户。

客户端程序：

```
/* TCPClient.cpp */
#pragma warning( disable : 4996)

#include <stdlib.h>
#include <stdio.h>
#include <winsock2.h>
#include <string.h>
#include <process.h>


#define BUFLEN      2000                    // 缓冲区大小
#define WSVERS      MAKEWORD(2, 0)          // 指明版本 2.0
#pragma comment(lib,"ws2_32.lib")           // 使用 winsock 2.0 Llibrary

unsigned __stdcall rec(void* ssock)
{
    char    buf[BUFLEN + 1];                /* buffer for one line of t
ext  */
    int cc;
    SOCKET sock = (SOCKET) ssock;
    while (1)
    {
```

```
        cc = recv(sock, buf, 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) {
            buf[cc] = '\0';                          // ensure null-termi
nation
            printf("\r%s\n>>", buf);                          // 显示所接
收的字符串
        }
    }

    return 0;
}

/*-------------------------------------------------------------------
---
 * main - TCP client for TIME service
 *-------------------------------------------------------------------
---
 */
void
main(int argc, char* argv[])
{
    char tem1[20] = "127.0.0.1";
    char* host = tem1;      /* server IP to connect         */
    char tem2[6] = "50500";
    char* service = tem2;       /* server port to connect       */
    struct sockaddr_in sin;             /* an Internet endpoint address
 */
    SOCKET  sock;                       /* socket descriptor
 */

    WSADATA wsadata;
    WSAStartup(WSVERS, &wsadata);                           //加载
winsock library。WSVERS 为请求的版本，wsadata 返回系统实际支持的最高版本

    sock = socket(PF_INET, SOCK_STREAM, IPPROTO_TCP);      //创建套接字，参
数：因特网协议簇(family)，流套接字，TCP 协议
                                                           //返回：要监听套
接字的描述符或 INVALID_SOCKET
```

```
    memset(&sin, 0, sizeof(sin));                    // 从&sin 开始
的长度为 sizeof(sin)的内存清 0
    sin.sin_family = AF_INET;                        // 因特网地址簇
    sin.sin_addr.s_addr = inet_addr(host);           // 服务器 IP 地
址(32 位)
    sin.sin_port = htons((u_short)atoi(service));    // 服务器端口
号
    connect(sock, (struct sockaddr*) & sin, sizeof(sin));  // 连接到服务
器
    unsigned int thrid;
    HANDLE thr = (HANDLE)_beginthreadex(NULL, 0, &rec, (void*) sock, 0,
 &thrid);

    char buffer[BUFLEN];
    memset(buffer, 0, BUFLEN);
    int cc = 0;
    while (1)
    {
        printf(">>");
        scanf("%[^\n]", buffer);
        getchar();
        cc = send(sock, buffer, strlen(buffer), 0);
    }

    WaitForSingleObject(thr, INFINITE);
    CloseHandle(thr);
    closesocket(sock);                               // 关闭监听套接字
    WSACleanup();                                    // 卸载
winsock library

    printf("按回车键继续...");
    getchar();                                        // 等待任意按键
}
```

服务器端程序：

```
/* TCPServer.cpp - main */
#pragma warning( disable : 4996)
#include <stdlib.h>
#include <stdio.h>
#include <winsock2.h>
#include <time.h>
#include <process.h>
#include "conio.h"
#define CLLEN 100
#define QLEN 5
#define BUFFSIZE 100
#define WSVERS  MAKEWORD(2, 0)
```

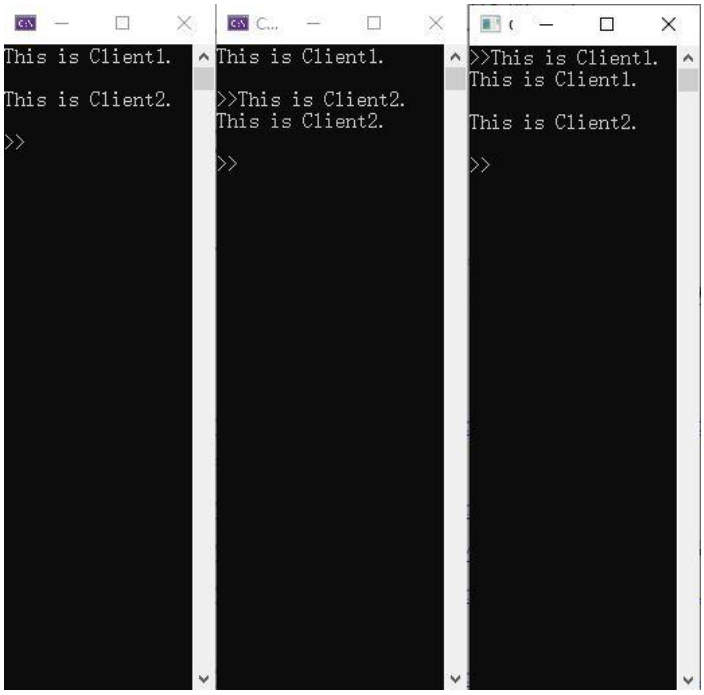
```
#pragma comment(lib,"ws2_32.lib")  //使用 winsock 2.2 library
CRITICAL_SECTION cs;                    // 临界区。每个时刻只有一个线程可以进入临界区
SOCKET arr[CLLEN];

unsigned __stdcall client(void* p)
{
    int pos = (*(int*)p)-1;

    char buffer[BUFFSIZE];
    memset(buffer, 0, BUFFSIZE);
    int cc;
    while (1)
    {
        cc = recv(arr[pos], buffer, BUFFSIZE, 0);
        if (cc == SOCKET_ERROR){                            // 出错。其后必须关闭套接字 sock
            printf("Error: %d.\n", GetLastError());
            break;
        }else if (cc == 0) {                                // 对方正常关闭
            printf("Server closed!", buffer);
            break;
        }
        else if (cc > 0)
        {
            buffer[cc] = '\0';                              // ensure null-termination
            char tem[BUFFSIZE];
            memset(tem, 0, BUFFSIZE);
            //printf("success!");
            sprintf(tem, "%s\n", buffer);
            EnterCriticalSection(&cs);      // 等待进入临界区
            printf("\r%s\n>>", tem);                                    //这里暂时会有一个 bug，当字符串仅一个字符时会多出现一个'>'
            LeaveCriticalSection(&cs);      // 离开临界区
            for (int i = 0; i < CLLEN; i++)
            {
                if (arr[i] != NULL)
                {
                    cc = send(arr[i], tem, strlen(tem), 0);
                }
            }
        }
    }
```

```
    fflush(stdout);
    (void)closesocket(arr[pos]);                                    // 关闭连
接套接字
    arr[pos] = NULL;
    return 0;
}

unsigned __stdcall server(void* p)
{
    char buffer[BUFFSIZE];
    memset(buffer, 0, BUFFSIZE);
    int cc;
    while (1)
    {
        printf(">>");
        scanf("%[^\n]", buffer);
        if (!strcmp(buffer, "")) break;
        getchar();
        printf("%s\n", buffer);
        for (int i = 0; i < CLLEN; i++)
        {
            if (arr[i] != NULL)
            {
                cc = send(arr[i], buffer, strlen(buffer), 0);
            }
        }
    }

    return 0;
}

int main(int argc, char* argv[])
/* argc: 命令行参数个数， 例如: C:\> TCPServer 8080
                    argc=2 argv[0]="TCPServer",argv[1]="8080" */
{
    struct  sockaddr_in fsin;       /* the from address of a client   *
/
    SOCKET  msock;          /* master & slave sockets         */
    WSADATA wsadata;
    char tem[6] = "50500";
    char* service = tem;
    struct  sockaddr_in sin;        /* an Internet endpoint address
 */
    int     alen;                   /* from-address length
 */
    char* pts;                  /* pointer to time string          */
```

```
    time_t  now;                    /* current time
 */


    WSAStartup(WSVERS, &wsadata);                           // 加载
winsock library。WSVERS 指明请求使用的版本。wsadata 返回系统实际支持的最高版本
    msock = socket(PF_INET, SOCK_STREAM, IPPROTO_TCP);  // 创建套接字，参
数：因特网协议簇(family)，流套接字，TCP 协议
                                                        // 返回：要监听套
接字的描述符或 INVALID_SOCKET

    memset(&sin, 0, sizeof(sin));                           // 从&sin 开始的
长度为 sizeof(sin)的内存清 0
    sin.sin_family = AF_INET;                               // 因特网地址簇
(INET-Internet)
    sin.sin_addr.s_addr = INADDR_ANY;                       // 监听所有(接口
的)IP 地址。
    sin.sin_port = htons((u_short)atoi(service));       // 监听的端口号。
atoi--把 ascii 转化为 int，htons--主机序到网络序(16 位)
    bind(msock, (struct sockaddr*) & sin, sizeof(sin));  // 绑定监听的 IP
地址和端口号

    listen(msock, QLEN);                                    // 等待建立连
接的队列长度为 5


    InitializeCriticalSection(&cs);   // 临界区初始化
    HANDLE thr[CLLEN];
    unsigned int thrid[CLLEN];
    int pos[CLLEN];
    unsigned int ct = 1;
    for (int i = 0; i < CLLEN; i++)
    {
        thr[i] = NULL;
        pos[i] = i;
    }
    thr[0] = (HANDLE)_beginthreadex(NULL, 0, &server, (void*)pos, 0, th
rid);
    while (!_kbhit()) {                                     // 检测是否
有按键
        alen = sizeof(struct sockaddr);                 // 取到地址结构
的长度
        arr[ct-1] = accept(msock, (struct sockaddr*) & fsin, &alen); //
 如果有新的连接请求，返回连接套接字，否则，被阻塞。fsin 包含客户端 IP 地址和端口
号
```

```
        thr[ct] = (HANDLE)_beginthreadex(NULL, 0, &client, (void*)(pos+
ct), 0, thrid+ct);
        ct++;
        //(void)time(&now);                                    // 取得
系统时间
        //pts = ctime(&now);                                    // 把
时间转换为字符串
        //(void)send(ssock, pts, strlen(pts), 0);               // 把缓
冲区(指针,长度)的数据发送出去
        //printf("%s", pts);                                    // 显
示发送字符串
    }

    for (int i = 0; i < CLLEN; i++)
    {
        if(thr[i]!=NULL) WaitForSingleObject(thr[i], INFINITE);
    }

    DeleteCriticalSection(&cs);  // 删除临界区
    for (int i = 0; i < CLLEN; i++)
    {
        if (thr[i] != NULL) CloseHandle(thr[i]);
    }
    (void)closesocket(msock);                                  // 关闭监
听套接字
    WSACleanup();                                              // 卸载
winsock library
}
```

运行截屏：



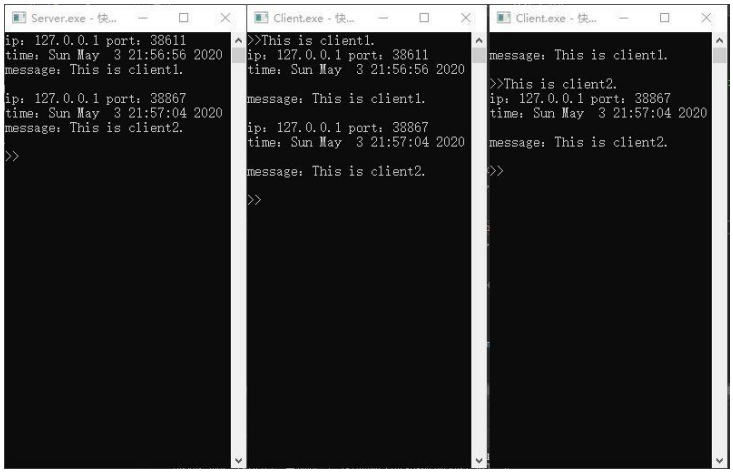
(2)服务器程序转发某个客户端发来的消息时都在消息前面加上该客户端的 IP 地址和端口号以及服务器的当前时间。要求服务器程序把转发的消息也显示出来。

服务器程序(修改部分)：

```
// 主要新增结构体用于传参，输出格式也改变了一下
typedef struct {
    int pos;
    sockaddr_in from;
}argm;
unsigned __stdcall client(void* p)
{
    int pos = (*(argm*)p).pos;
    sockaddr_in from = (*(argm*)p).from;
    char buffer[BUFFSIZE];
    memset(buffer, 0, BUFFSIZE);
    int cc;
    while (1)
    {
        cc = recv(arr[pos], buffer, BUFFSIZE, 0);
        if (cc == SOCKET_ERROR){                          // 出错。其后必
须关闭套接字 sock
            printf("Error: %d.\n", GetLastError());
            break;
        }else if (cc == 0) {                              // 对方正常关闭
            printf("Server closed!", buffer);
            break;
        }
        else if (cc > 0)
        {
            buffer[cc] = '\0';                           // ensure null-t
ermination
            char tem[BUFFSIZE];
            memset(tem, 0, BUFFSIZE);
            time_t now = time(0);
            char* curr = ctime(&now);

            sprintf(tem, "ip: %s port: %d\ntime: %smessage: %s\n",
                inet_ntoa(from.sin_addr), from.sin_port, curr, buffer);

            EnterCriticalSection(&cs);      // 等待进入临界区
            printf("\rip: %s port: %d\ntime: %smessage: %s\n\n>>",
                inet_ntoa(from.sin_addr), from.sin_port, curr, buffer);
            LeaveCriticalSection(&cs);      // 离开临界区

            for (int i = 0; i < CLLEN; i++)
            {
```

```
                if (arr[i] != NULL)
                {
                    cc = send(arr[i], tem, strlen(tem), 0);
                }
            }
        }
    }

    fflush(stdout);
    (void)closesocket(arr[pos]);                                    // 关闭连
接套接字
    arr[pos] = NULL;
    return 0;
}
    // 用结构体中变量给线程传参
    argm ag[CLLEN];
    unsigned int ct = 0;
    for (int i = 0; i < CLLEN; i++)
    {
        thr[i] = NULL;
        ag[i].pos = i;
    }
    thr[0] = (HANDLE)_beginthreadex(NULL, 0, &server, (void*)&sin, 0, t
hrid);
    while (!_kbhit()) {                                             // 检测是否
有按键
        alen = sizeof(struct sockaddr);                         // 取到地址结构
的长度
        arr[ct] = accept(msock, (struct sockaddr*) & (ag[ct].from)  , &
alen); // 如果有新的连接请求，返回连接套接字，否则，被阻塞。fsin 包含客户端 IP
地址和端口号
        thr[ct+1] = (HANDLE)_beginthreadex(NULL, 0, &client, (void*)(ag
 + ct), 0, thrid + ct);
        ct++;
    }
```

运行截屏：



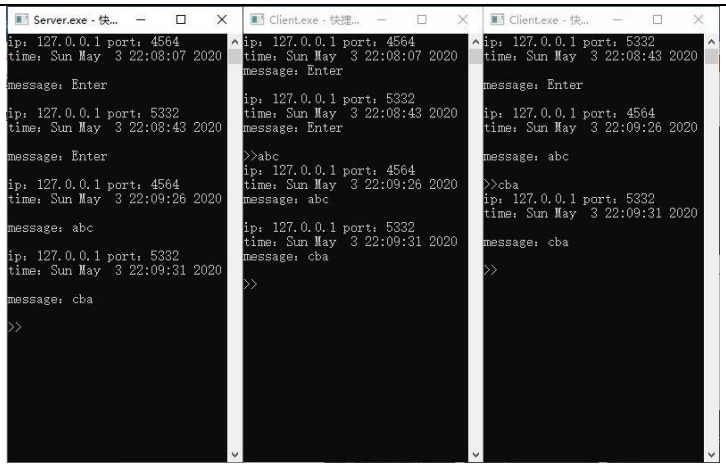
(3)新客户刚连接时服务器端把“enter”消息（包含客户端 IP 地址和端口号）发送给所有客户端。

服务器程序(修改部分)：

```
unsigned __stdcall client(void* p)
{
    int pos = (*(argm*)p).pos;
    sockaddr_in from = (*(argm*)p).from;
    char buffer[BUFFSIZE] = "Enter";
    int cc;
    char tem[BUFFSIZE];
    memset(tem, 0, BUFFSIZE);
    time_t now = time(0);
    char* curr = ctime(&now);
    sprintf(tem, "ip: %s port: %d\ntime: %smessage: %s\n",
        inet_ntoa(from.sin_addr), from.sin_port, curr, buffer);

    EnterCriticalSection(&cs);      // 等待进入临界区
    printf("\rip: %s port: %d\ntime: %smessage: %s\n\n>>",
        inet_ntoa(from.sin_addr), from.sin_port, curr, buffer);
    LeaveCriticalSection(&cs);      // 离开临界区

    for (int i = 0; i < CLLEN; i++)
    {
        if (arr[i] != NULL)
        {
            cc = send(arr[i], tem, strlen(tem), 0);
        }
    }
    while (1)
    {
        cc = recv(arr[pos], buffer, BUFFSIZE, 0);
        if (cc == SOCKET_ERROR){                            // 出错。其后必
须关闭套接字 sock
            printf("Error: %d.\n", GetLastError());
            break;
```

```
        }else if (cc == 0) {                                    // 对方正常关闭
            printf("Server closed!", buffer);
            break;
        }
        else if (cc > 0)
        {
            buffer[cc] = '\0';                              // ensure null-t
ermination

            now = time(0);
            curr = ctime(&now);

            sprintf(tem, "ip: %s port: %d\ntime: %smessage: %s\n",
                inet_ntoa(from.sin_addr), from.sin_port, curr, buffer);

            EnterCriticalSection(&cs);      // 等待进入临界区
            printf("\rip: %s port: %d\ntime: %smessage: %s\n\n>>",
                inet_ntoa(from.sin_addr), from.sin_port, curr, buffer);
            LeaveCriticalSection(&cs);      // 离开临界区

            for (int i = 0; i < CLLEN; i++)
            {
                if (arr[i] != NULL)
                {
                    cc = send(arr[i], tem, strlen(tem), 0);
                }
            }
        }
    }

    fflush(stdout);
    (void)closesocket(arr[pos]);                                // 关闭连
接套接字
    arr[pos] = NULL;
    return 0;
}
```

运行截屏：



(4)客户端输入 exit 时退出客户端程序（正常退出），或者客户端直接关闭窗口退出（异常退出），服务器都会把该客户 leave 的消息广播给所有客户。

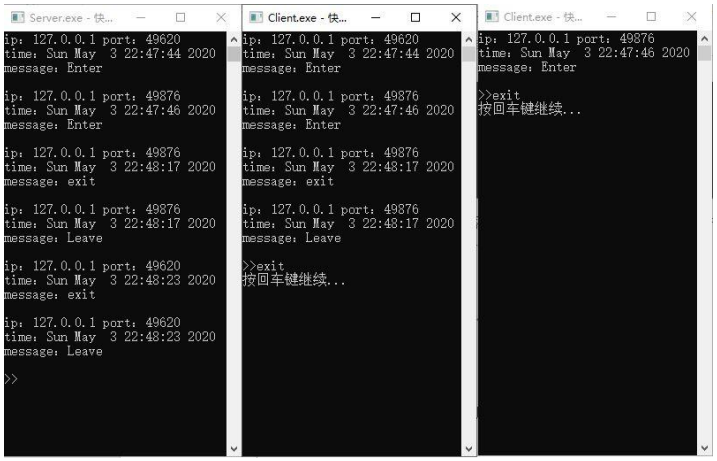
服务器程序(修改部分)：

```
unsigned __stdcall client(void* p)
{
    int pos = (*(argm*)p).pos;
    sockaddr_in from = (*(argm*)p).from;
    char buffer[BUFFSIZE] = "Enter";
    int cc;
    char tem[BUFFSIZE];
    memset(tem, 0, BUFFSIZE);
    time_t now = time(0);
    char* curr = ctime(&now);
    sprintf(tem, "ip: %s port: %d\ntime: %smessage: %s\n",
        inet_ntoa(from.sin_addr), from.sin_port, curr, buffer);

    EnterCriticalSection(&cs);      // 等待进入临界区
    printf("\rip: %s port: %d\ntime: %smessage: %s\n\n>>",
        inet_ntoa(from.sin_addr), from.sin_port, curr, buffer);
    LeaveCriticalSection(&cs);      // 离开临界区

    for (int i = 0; i < CLLEN; i++)
    {
        if (arr[i] != NULL)
        {
            cc = send(arr[i], tem, strlen(tem), 0);
        }
    }
    while (1)
    {
        cc = recv(arr[pos], buffer, BUFFSIZE, 0);
        if (cc == SOCKET_ERROR){                                    // 出错。其后必
须关闭套接字 sock
            //printf("Error: %d.\n", GetLastError());
```

```
            break;
        }else if (cc == 0) {                                    // 对方正常关闭
            printf("Server closed!", buffer);
            break;
        }
        else if (cc > 0)
        {
            buffer[cc] = '\0';                          // ensure null-t
ermination

            now = time(0);
            curr = ctime(&now);

            sprintf(tem, "ip: %s port: %d\ntime: %smessage: %s\n",
                inet_ntoa(from.sin_addr), from.sin_port, curr, buffer);

            EnterCriticalSection(&cs);      // 等待进入临界区
            printf("\rip: %s port: %d\ntime: %smessage: %s\n\n>>",
                inet_ntoa(from.sin_addr), from.sin_port, curr, buffer);
            LeaveCriticalSection(&cs);      // 离开临界区

            for (int i = 0; i < CLLEN; i++)
            {
                if (arr[i] != NULL)
                {
                    cc = send(arr[i], tem, strlen(tem), 0);
                }
            }
        }
    }
    strcpy(buffer, "Leave");
    sprintf(tem, "ip: %s port: %d\ntime: %smessage: %s\n",
        inet_ntoa(from.sin_addr), from.sin_port, curr, buffer);

    EnterCriticalSection(&cs);      // 等待进入临界区
    printf("\rip: %s port: %d\ntime: %smessage: %s\n\n>>",
        inet_ntoa(from.sin_addr), from.sin_port, curr, buffer);
    LeaveCriticalSection(&cs);      // 离开临界区

    for (int i = 0; i < CLLEN; i++)
    {
        if (arr[i] != NULL)
        {
            cc = send(arr[i], tem, strlen(tem), 0);
        }
    }
```

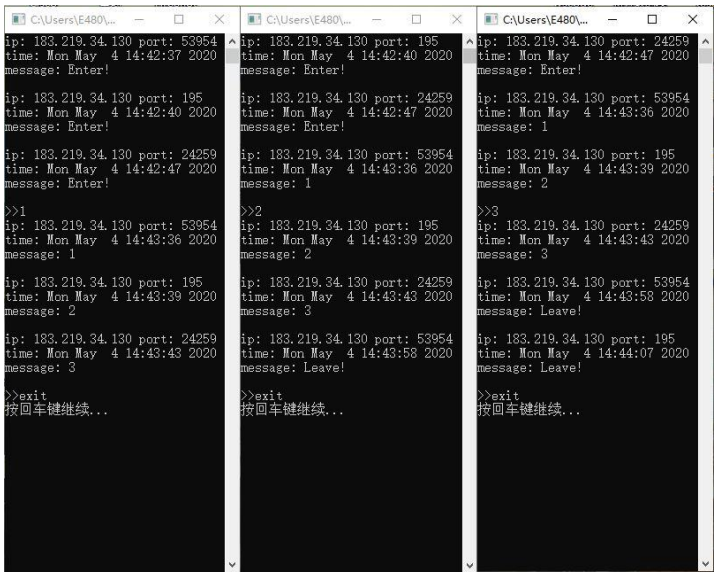
```
    fflush(stdout);
    (void)closesocket(arr[pos]);                                    // 关闭连
接套接字
    arr[pos] = NULL;
    return 0;
}
```

运行截屏：



(5)运行客户端程序测试与老师的服务器程序的连接（103.26.79.35:50500）。

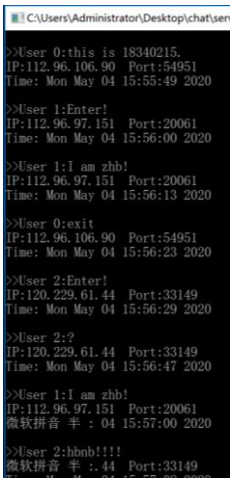
运行截屏（客户端）：



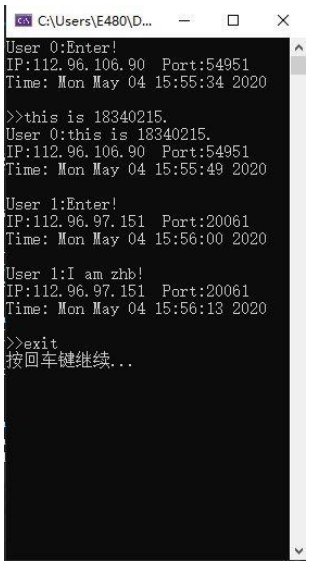
(6) （选做）与同学的程序进行相互测试，一个人可以与多人测试，截屏选择其中一个。

同学的学号姓名（可以多人）：

作为服务器运行截屏：



作为客户端运行截屏：



【完成情况】

是否完成了这些步骤？（√完成 ×未做或未完成）

(1) [√] (2) [√] (3) [√] (4) [√] (5) [√] (6) [√]

【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）。

1,编译运行老师例程 Server 时,VS 编译报错 MSB802。解决方法：项目->属性->常规->平台工具集。改为最新版即可。

2，关于 fflush(stdout):第一次接触该函数，查阅相关资料了解到 fflush(stdin)主要用来清除输入缓冲区的多余字符（如 scanf()后余留’\n’、’ ’）,而 fflush(stdout)用来查看输出缓冲区是否有多余字符未输出，主要用来检查错误。

https://blog.csdn.net/Veniversum/article/details/62048870?utm_medium=distribute.pc_relevant.none-task-blog-BlogCommendFromBaidu-8&depth_1-utm_source=distribute.pc_relevant.none-task-blog-BlogCommendFromBaidu-8

3，一个找了比较久的 bug 是数组下标错误，在服务器程序中，用来在服务器端广播的是线程 0，它调用的是单独写的一个函数，关于它的变量都是单独处理，但在编程过程中为了方便，我将一些信息放在了一些数组下标 0 的地方，后面修改时，忘记修改客户端内相关信息的下标，导致连接失败。

4，和老师的服务器联机的时候，每次都会多发一个”>>”，思考后认为可能老师的程序是将“\n>>”等格式化信息一起作为传输信息的一部分传输，但我的程序中认为这些都是固化的格式，所以直接在客户端和服务器端约定好即可，不用传输。对自己的程序稍加修改，完成了和老师服务器的格式化对话。

5，不论是和老师的程序联机，还是和同学多人联机，仅需要修改 ip 和端口号，体会到了互联网的神奇。

5，实验一较简单，做实验一的时候还不明白老师的分步实验设置。实验二有了十分清晰的体会，刚开始面对这样一个新问题，将问题拆解为小步，一步一步升级，很快就能成功。

【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/2、Chat 实验

(3) 截止日期（不迟于）：5 月 6 日（周三） 23:00。

上传文件：(1)学号_姓名_chat 实验报告.doc (最好用 doc 文件)

(2)学号_姓名_chat 实验要求.rar (打包源程序文件和可执行文件)