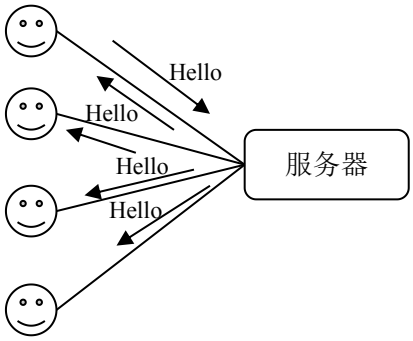
实验报告

院 系 数据科学与计算机学院 班 级 17 学号 17341097 姓名 廖永滨

【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】

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。

【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 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）：

(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 */

#include <stdlib.h>
#include <stdio.h>
#include <winsock2.h>
#include <string.h>
#include "conio.h"
#include <windows.h>
#include <process.h>
#include <math.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

/*-------------------------------------------------------------------------
 * main - TCP client for TIME service
 *-------------------------------------------------------------------------
 */
SOCKET    sock;                         /* socket descriptor        */

unsigned __stdcall SumAll(void * p)     // 用结构变量或全局变量带入参数
{
    char buf[BUFLEN];
    memset(buf, '\0', BUFLEN);
    while (1) {
        int cc = recv(sock, buf, BUFLEN, 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-termination
            printf("%s", buf);                          // 显示所接收的字符串
        }
    }
    return 0;
}
```

```
void
main(int argc, char *argv[])
{
    char *host = "172.18.187.9";        /* server IP to connect         */
    char *service = "50500";        /* server port to connect       */
    struct sockaddr_in sin;                 /* an Internet endpoint address */
    char buf[BUFLEN+1];                 /* buffer for one line of text  */
    HANDLE hThread;
    int cc;                                 /* recv character count         */
    WSADATA wsadata;
    WSAStartup(WSVERS, &wsadata);
    sock = socket(PF_INET, SOCK_STREAM, IPPROTO_TCP);
    memset(&sin, 0, sizeof(sin));
    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
    hThread = (HANDLE)_beginthreadex(NULL, 0, &SumAll, NULL, 0, NULL);
    while (1) {
        memset(buf, '\0', BUFLEN+1);
        scanf_s("%s", buf, BUFLEN + 1);
        if (buf[0] == 'e'&&buf[1] == 'x'&&buf[2] == 'i')
            if (buf[3] == 't'&&buf[4] == '\0')
                break;
        send(sock, buf, strlen(buf)+1, 0);                      //送内容
    }
    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：

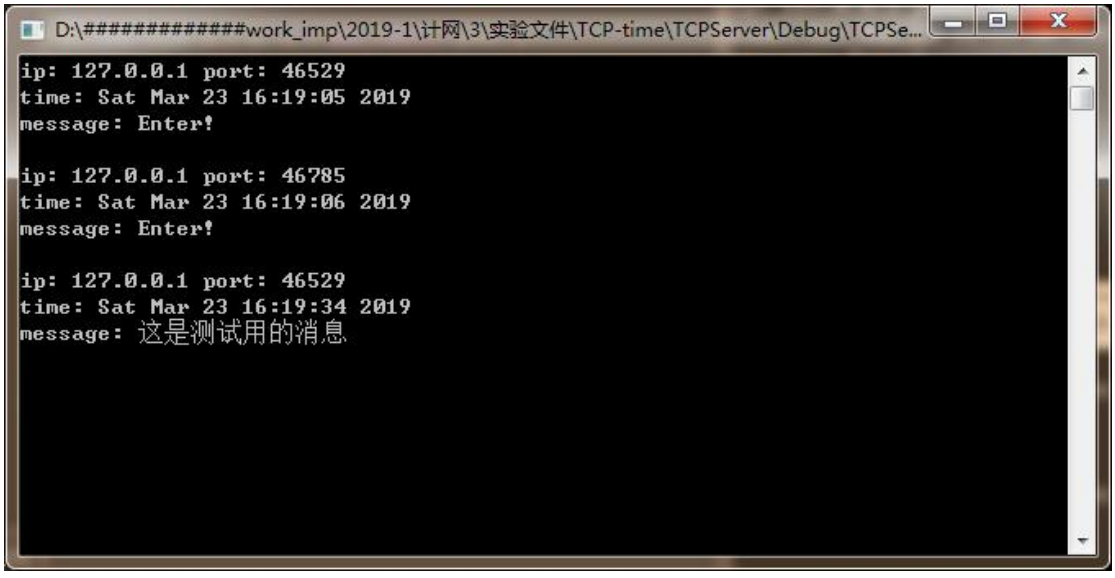
```
#include <stdlib.h>
#include <stdio.h>
#include <winsock2.h>
#include <time.h>
#include "conio.h"
#include <windows.h>
#include <process.h>
#include <math.h>

#define QLEN     5
#define  WSVERS    MAKEWORD(2, 0)
```

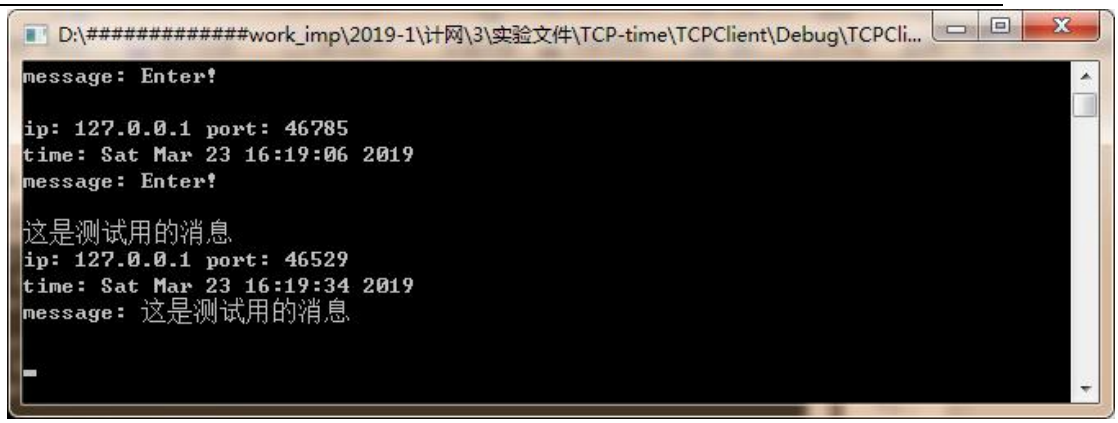
```
#define MAX_CHATS  100

#pragma comment(lib,"ws2_32.lib")  //使用 winsock 2.2 library
SOCKET ssock[MAX_CHATS];
bool is_used[MAX_CHATS];
struct   sockaddr_in fsin[MAX_CHATS];       /* the from address of a client   */
char *pts;                      /* pointer to time string          */
time_t   now;                   /* current time                        */

struct struparam {              /* 用于带入参数 */
    int threadno;              /* 线程编号 */
    int index;
}param[MAX_CHATS];

unsigned __stdcall SumAll(void * p)    // 用结构变量或全局变量带入参数
{
    struparam * p1 = (struparam *)p;   // 参数类型转换
    char buf[2000];
    char buf2[2000];
    char addr_ip[300];
    sprintf(addr_ip, "%d.%d.%d.%d", fsin[p1->index].sin_addr.S_un.S_un_b.s_b1,
fsin[p1->index].sin_addr.S_un.S_un_b.s_b2, \
            fsin[p1->index].sin_addr.S_un.S_un_b.s_b3,
fsin[p1->index].sin_addr.S_un.S_un_b.s_b4);
    (void)time(&now);                                   // 取得系统时间
    pts = ctime(&now);                                  // 把时间转换为字符串
    sprintf(buf, "ip: %s port: %d\ntime: %smessage: Enter!\n", addr_ip,
        fsin[p1->index].sin_port,pts);
    printf("%s\n", buf);
    for (int i = 0; i < MAX_CHATS; i++) {
        if (is_used[i])
            send(ssock[i], buf, strlen(buf), 0);                    //送内容
    }
    (void)time(&now);                                   // 取得系统时间
    pts = ctime(&now);                                  // 把时间转换为字符串
    while (1) {
        int dd = recv(ssock[p1->index], buf2, 2000, 0);
        (void)time(&now);                                   // 取得系统时间
        pts = ctime(&now);
        if (dd == SOCKET_ERROR) {
            printf("Error: %d.\n", GetLastError());
            sprintf(buf, "ip: %s port: %d\ntime: %smessage: Leave!\n", addr_ip,
                fsin[p1->index].sin_port, pts);
            printf("%s\n", buf);
            is_used[p1->index] = false;
            for (int i = 0; i < MAX_CHATS; i++) {
```

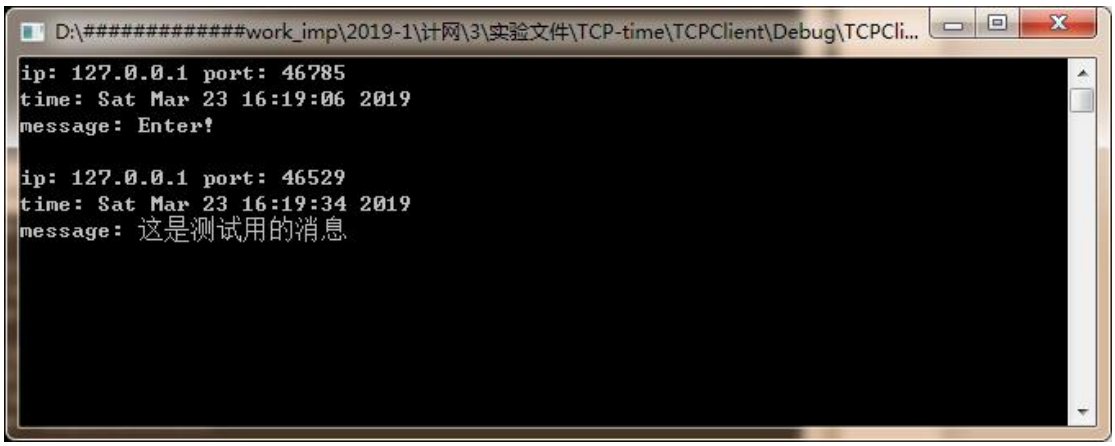
```
                    if (is_used[i])
                        send(ssock[i], buf, strlen(buf), 0);
                }
                (void)closesocket(ssock[p1->index]);                    //
                break;
            }
            else if (dd == 0) {                             // 对方正常关闭
                printf("Server closed!");
                sprintf(buf, "ip: %s port: %d\ntime: %smessage: Leave!\n", addr_ip,
                    fsin[p1->index].sin_port, pts);
                printf("%s\n", buf);
                for (int i = 0; i < MAX_CHATS; i++) {
                    is_used[p1->index] = false;
                    if (is_used[i])
                        send(ssock[i], buf, strlen(buf), 0);
                }
                (void)closesocket(ssock[p1->index]);                    //
                break;
            }
            sprintf(buf, "ip: %s port: %d\ntime: %smessage: %s\n", addr_ip,
                fsin[p1->index].sin_port, pts,buf2);
            printf("%s\n", buf);
            for (int i = 0; i < MAX_CHATS; i++) {
                if(is_used[i])
                    send(ssock[i], buf, strlen(buf), 0);
            }
        }
        (void)closesocket(ssock[p1->index]);
        is_used[p1->index] = false;
        return 0;
    }
    void
    main(int argc, char *argv[])
    {
        SOCKET   msock;           /* master & slave sockets       */
        WSADATA wsadata;
        char *service = "50500";
        struct  sockaddr_in sin;       /* an Internet endpoint address     */
        int      alen;                     /* from-address length              */

        HANDLE hThread[MAX_CHATS];
        struparam *ptr;
        WSAStartup(WSVERS, &wsadata);
        msock = socket(PF_INET, SOCK_STREAM, IPPROTO_TCP);
        memset(&sin, 0, sizeof(sin));
        sin.sin_family = AF_INET;
```

```
sin.sin_addr.s_addr = INADDR_ANY;
sin.sin_port = htons((u_short)atoi(service));
bind(msock, (struct sockaddr *)&sin, sizeof(sin));
for (int i = 0; i < MAX_CHATS; i++) {
    is_used[i] = false;
}
listen(msock, 5);
int i = 0;
while(!_kbhit()){                                    // 检测是否有按键
   alen = sizeof(struct sockaddr);                    // 取到地址结构的长度

   for (i = 0; i < MAX_CHATS; i++) {
       if (!is_used[i])
           break;
   }
   is_used[i] = true;
   ssock[i] = accept(msock, (struct sockaddr *)&fsin[i], &alen);
   param[i].index = i;
   param->threadno = i;
   ptr = &param[i];
   hThread[i] = (HANDLE)_beginthreadex(NULL, 0, &SumAll, (void *)ptr, 0, NULL);
   i++;
}
(void) closesocket(msock);
WSACleanup();
}
```

运行截屏：（从上到下依次为服务器端、客户端 1、客户端 2）







(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)：

```
sprintf(addr_ip, "%d.%d.%d.%d", fsin[p1->index].sin_addr.S_un.S_un_b.s_b1,
fsin[p1->index].sin_addr.S_un.S_un_b.s_b2,fsin[p1->index].sin_addr.S_un.S_un_b.s_b3,
    fsin[p1->index].sin_addr.S_un.S_un_b.s_b4);
sprintf(buf, "ip: %s port: %d\ntime: %smessage: %s\n", addr_ip,
    fsin[p1->index].sin_port, pts,buf2);
    //主要由上述两个语句完成，就是将消息 buf2 与 ip 地址、端口号、时间拼在一起
```

运行截屏：同（1）

(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)：

```
sprintf(buf, "ip: %s port: %d\ntime: %smessage: Enter!\n", addr_ip,
    fsin[p1->index].sin_port,pts);
printf("%s\n", buf);
for (int i = 0; i < MAX_CHATS; i++) {
    if (is_used[i])
        send(ssock[i], buf, strlen(buf), 0);                    //送内容
    }
```

//进来线程函数时就发送相应消息给所有客户端

运行截屏：同（1）

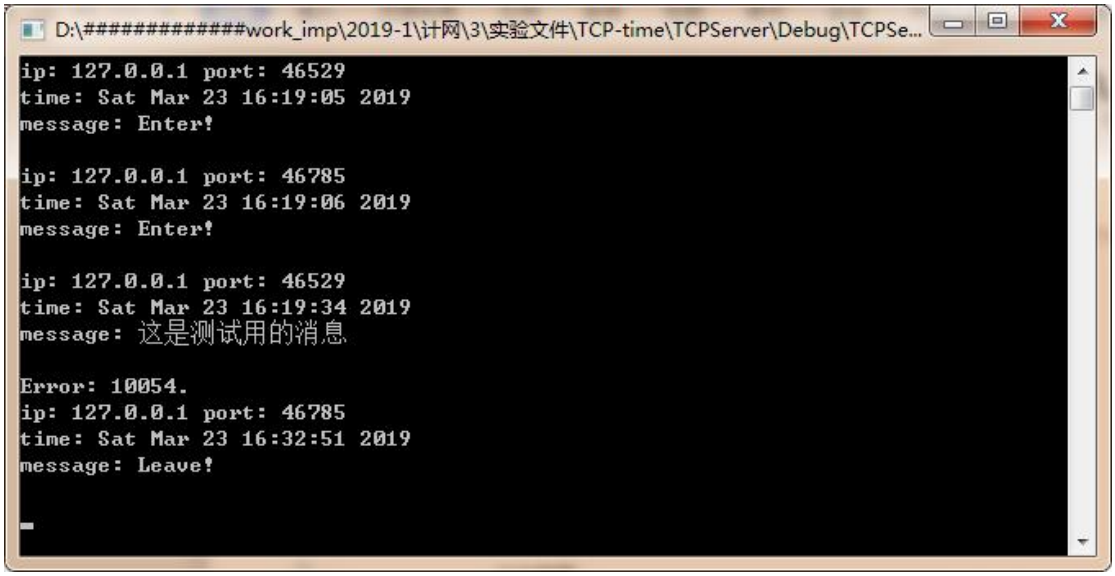
(4)客户端输入 exit 时退出客户端程序（正常退出），或者客户端直接关闭窗口退出（异常退出），服务器都会把该客户 leave 的消息广播给所有客户。

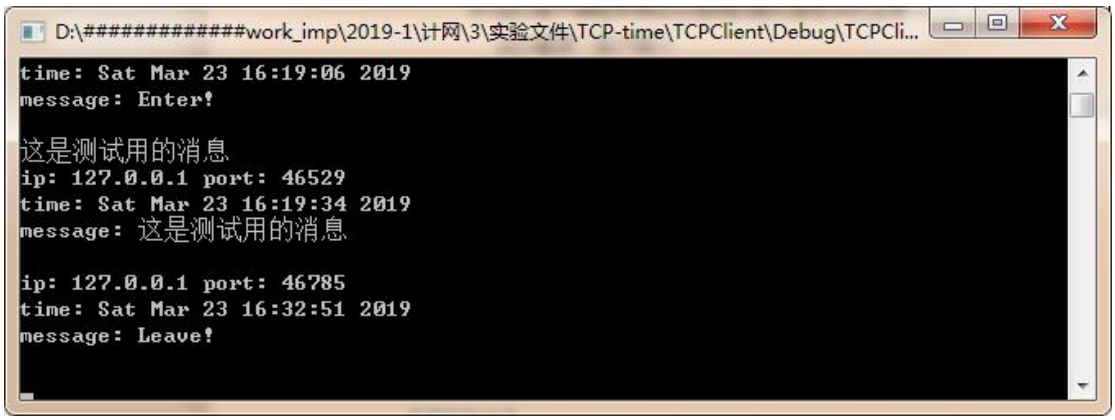
服务器程序(修改部分)：

```
sprintf(buf, "ip: %s port: %d\ntime: %smessage: Leave!\n", addr_ip,
    fsin[p1->index].sin_port, pts);
printf("%s\n", buf);
for (int i = 0; i < MAX_CHATS; i++) {
    is_used[p1->index] = false;
    if (is_used[i])
        send(ssock[i], buf, strlen(buf), 0);
    }
```

//recv 函数出错时就发送 leave 消息给所有客户端

运行截屏：（从上到下依次为服务器、未退出的客户端）

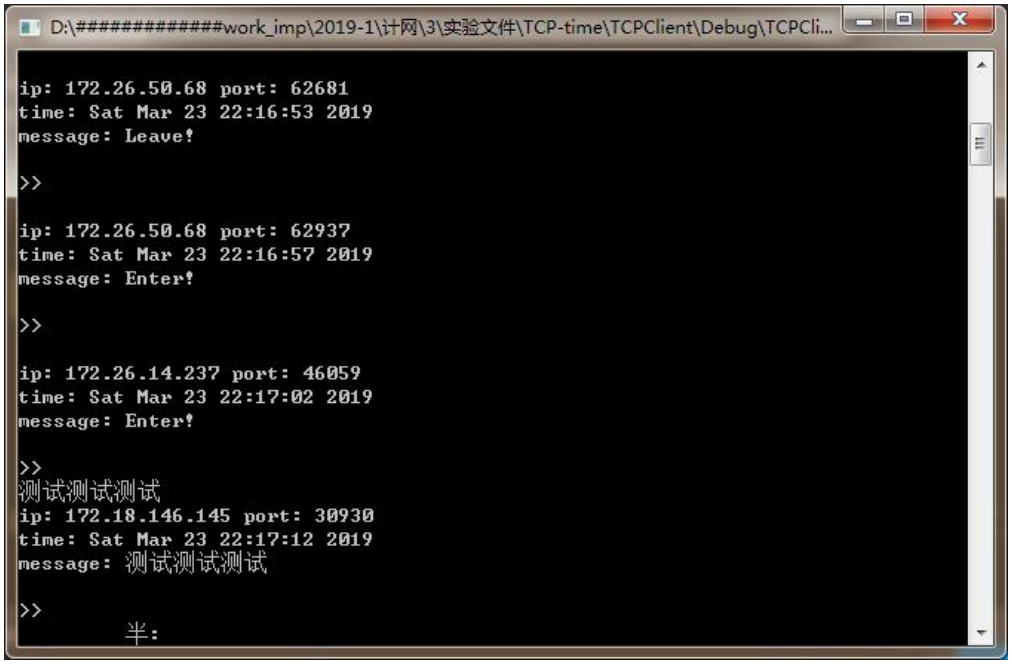




//可以看到，服务器将 leave 信息传给了所有客户端

(5)运行客户端程序测试与老师的服务器程序的连接（172.18.187.9:50500）。

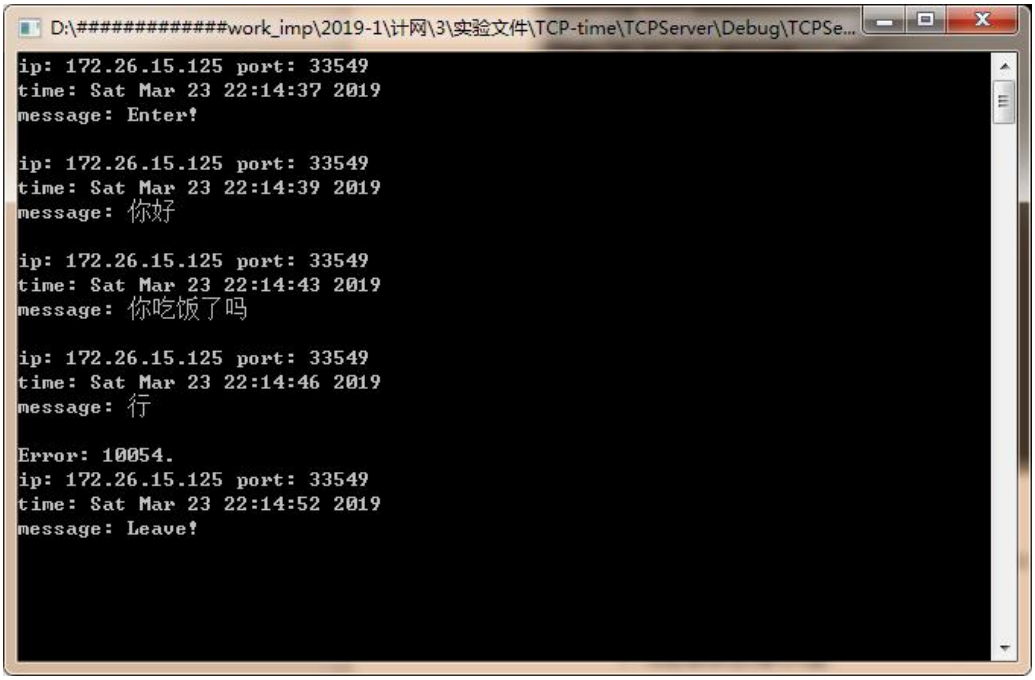
运行截屏（客户端）：



(6)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）:17341106 刘锋

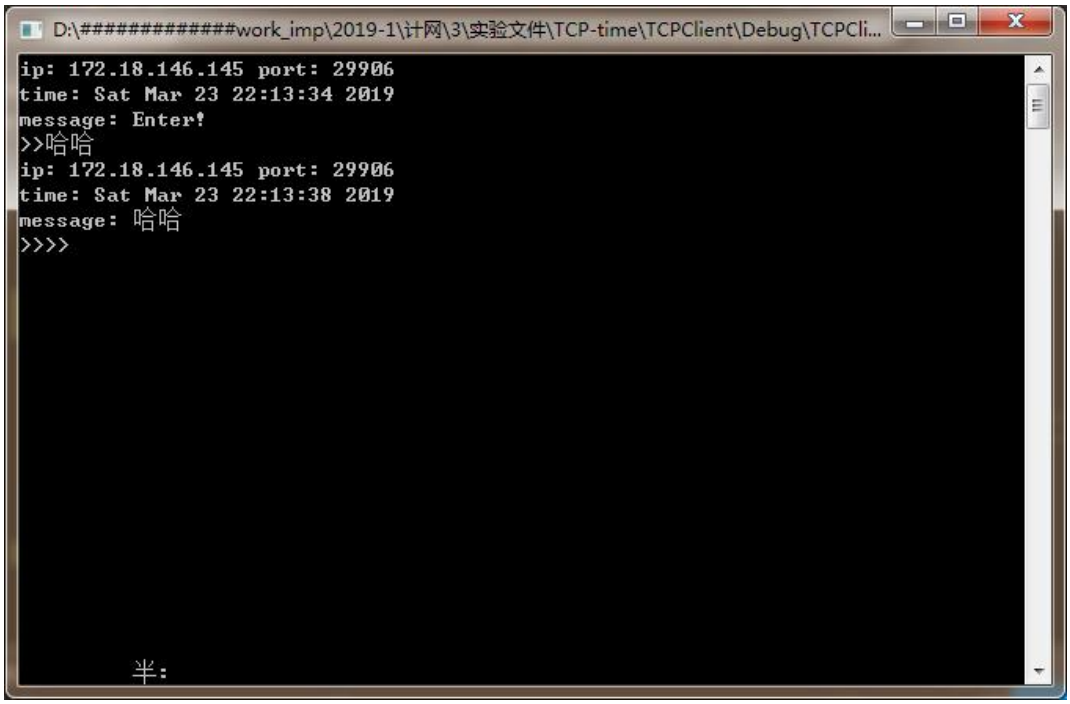
作为服务器运行截屏：



作为客户端运行截屏：

中山大學 SUN YAT-SEN UNIVERSITY

实验报告



【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】

遇到的问题：

1、线程调用时的编号问题

毕竟一个服务器要与多个客户端连接，依次要处理好不同线程之间的命名逻辑，要管理好套接字数组的使用，不能用重复了，不然会出错。

2、字符串的长度问题

多发送（接收）一个字符或者少发送（接收）一个字符都会带来各类 bug，出现字符串乱码的情况，解决方法也简单，就是长度那里+1 或者随便调一调，反正要把‘\0’传过去，而且要记得把字符串都初始化，memset 一遍。

3、线程函数参数的传输问题

在大量尝试之后，我发现像老师一样通过结构体传参是一种选择，但是也容易出现各种编译问题。最后我选了一种最简单的方法，就是把用到的参数声明成全局变量，就容易很多了。

4、群发问题

和线程处理问题一样，反正都是搞个数组标记一下，然后 for 循环一遍，挨个发送

实验体会：

计网实验真有趣，还能搞群聊

【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://172.18.187.9/netdisk/default.aspx?vm=17net

实验上交/编程实验/3、Chat 实验

(3) 截止日期（不迟于）：2019 年 3 月 23 日（周六） 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)