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】Echo 实验

【实验目的】掌握套节字的基本使用方法。

【实验说明】

- 把源程序和可执行文件放在相应的**上交源码**目录中。
- **截屏**用按键(Ctrl+Alt+PrintScreen)截取当前窗口

【参考资料】

- 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）
- https://www.runoob.com/cprogramming/c-function-sscanf.html (sscanf)
- sprintf 可以用于合并多个字符串和整数等：

```
char  buffer[50];
char *test = "Hello world!";
sprintf(buffer,"The length of %s is %d",test,strlen(test));/*赋予数值*/
```

【实验环境】

- **172.18.187.251 为校园网内的服务器地址，需要先用 VPN 连入校园网才能访问。**
- Windows + VS 2012 http://172.18.187.251/netdisk/default.aspx?vm=18net
- 对于 VS2015 和 VS2017 默认使用安全周期检查，如果不关闭 VS 的安全周期检查，很多字符串函数都不能用。
- Linux + gcc

【实验内容】

先尝试运行文件夹“TCP”中的程序：先运行 Server 程序(打开 TCPServer.sln，然后执行)再运行 Client 程序(打开 TCPClient.sln，然后执行)。这两个程序的功能是客户端从服务器获取当前时间。

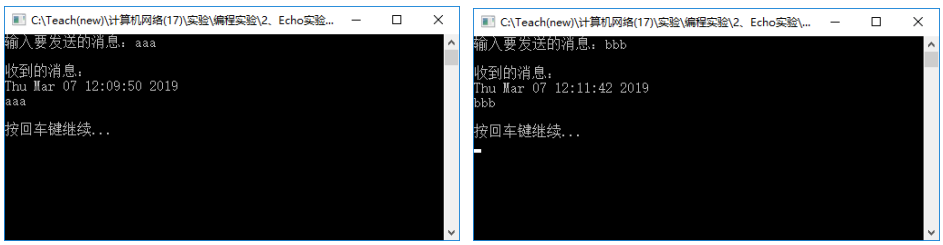
(1)编写 TCP Echo 程序

- **实验要求：**

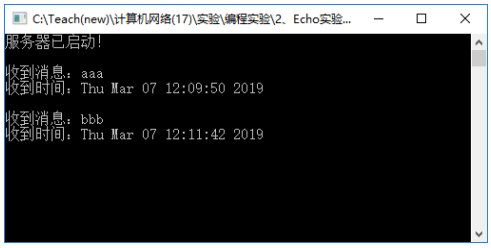
  服务器把客户端发送来的任何消息都返回给客户端，返回的消息前面要加上服务器的当前时间。**客户端**把返回的消息显示出来。客户端每输入一条消息就建立 TCP 连接，并把消息发送给服务器，在收到服务器回应后关闭连接。

- **参考运行截屏：**

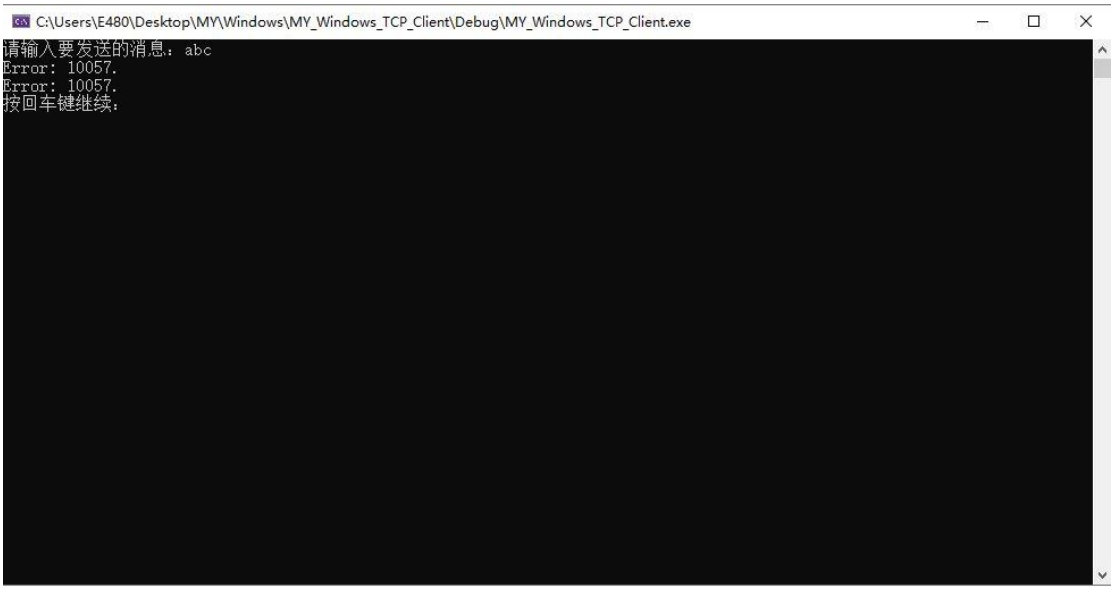
  客户端（两次运行）



  服务器：



- **只运行客户端程序而不运行服务器程序会出现什么错误，截屏并说明原因。**



Socket error 10057 - Socket is not connected

因为服务器未打开，所以实际上 connect 失败了，socket 没有连接。

- **服务器如何可以退出循环？**

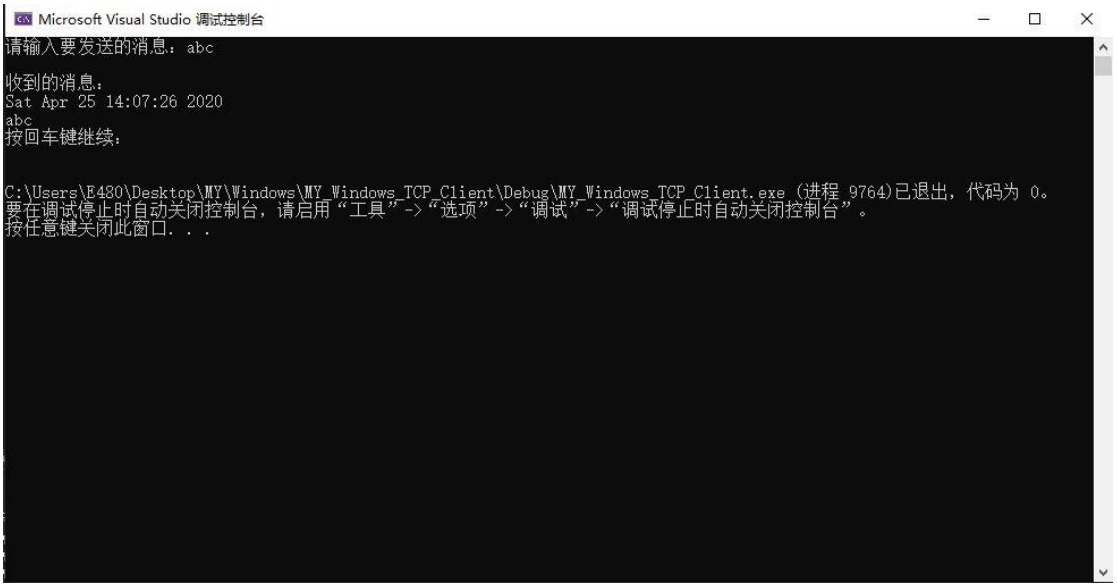
需要在运行客户端程序并输入字符串前，在服务器程序中键入任意字符，再在客户端上正常输入字符串，回车后，服务器就会结束。

因为 accept 是阻塞函数，只要运行服务器程序，就会一路运行至 accept 处，若此时键入字

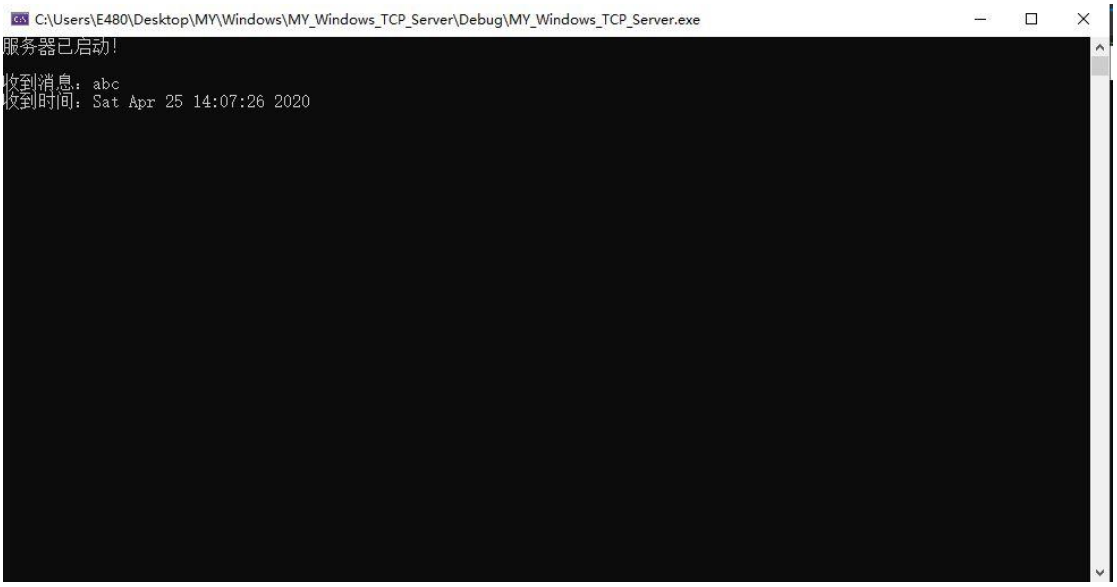
符，字符不会显示，只会被留在缓冲区。然后正常执行客户端程序，服务器正常结束一次循环，检查缓冲区是否键入字符，确实有，结束循环，程序结束。

- **截屏（ctrl+alt+PrintScreen）服务器和客户端的运行结果（注明客户端和服务器）：**

客户端：



服务器：（仍未结束）



- **服务器的全部源代码（或自选主要代码）：**

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <winsock2.h>
#include <time.h>
#include "conio.h"                  // 仅能用于 Dos 类操作系统
#pragma comment(lib,"ws2_32.lib")   // 使用 winsock 2.2 library
#pragma warning(disable : 4996)
#define WSVERS  MAKEWORD(2, 2)
```

```
// MAKEWORD 函数创建一个符合网络传输（小端？）的 2B 的 word，这里的参数表示后面
// 的 WSAStartup 将使用版本号 2.2（换成 2.0 也行）
#define SERVICE "50500"
#define QUSIZE 5
#define BUFFSIZE 100

int main(int argv,char** argc)
{

    WSADATA wsadata;
    WSAStartup(WSVERS, &wsadata);
    // 加载 winsock library，SVERS 指明请求使用的版本。wsadata 返回系统实际支
    // 持的最高版本

    SOCKET  msock, ssock;
    // master & slave sockets，为并发服务提供支持，否则只能 1 对 1
    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（这个换
    // 成 0 也行）
    // 返回：要监听套接字的描述符或 INVALID_SOCKET
    // sockaddr_in 与 sockaddr 二者的占用的内存大小是一致的，因此可以互相转化，
    // 从这个意义上说，他们并无区别
    struct  sockaddr_in sin;
    // 但 sockaddr_in 更方便程序员操作，而 sockaddr 更通用。后面相关 size 操作实
    // 际是要 sockaddr 的大小，但为方便直接取了 sockaddr_in 的大小
    // 惯例上会将先操作 sockaddr_in，再转化为 sockaddr
    memset(&sin, 0, sizeof(sin));                    // 将 sin 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。htons--主机序到网络序(host to network，s-short 16 位)，
    // 类似 MAKEWORD
    bind(msock, (struct sockaddr*) & sin, sizeof(sin));
    // 绑定监听的 IP 地址和端口号

    listen(msock, QUSIZE);
    // 进行监听，第二个参数为监听队列长度，下方循环是监听的具体操作，也是核心
    printf("服务器已启动！\n\n");
    while (!_kbhit())
    // 定义在 conio.h，若有按键，结束循环
    {
        struct  sockaddr_in fsin;
        int hd_sz = sizeof(fsin);
        ssock = accept(msock, (struct sockaddr*) & fsin, &hd_sz);
```

```
// 建立连接，创建新套接字，fsin 存放的是客户端的 sockaddr，hd_sz 为大小的指针

        char buffer[BUFFSIZE];
        memset(buffer, 0, BUFFSIZE);
        int cc = recv(ssock, buffer, BUFFSIZE, 0);
        if (cc == SOCKET_ERROR)
        // 出错。其后必须关闭套接字 sock
            printf("Error: %d.\n", GetLastError());
        else if (cc == 0) {
        // 对方正常关闭
            printf("Server closed!", buffer);
        }
        else if (cc > 0) {
            buffer[cc] = '\0';
        // ensure null-termination
            printf("收到息：%s\n", buffer);
        // 显示所接收的字符串
        }
        char tem[BUFFSIZE];
        memset(tem, 0, BUFFSIZE);
        time_t now = time(0);
        char* curr = ctime(&now);
        printf("收到时间：%s\n", curr);
        strcpy(tem, curr);
        int len = strlen(tem);
        tem[len - 1] = '\n';
        tem[len] = '\0';
        strcat(tem, buffer);
        cc = send(ssock, tem, strlen(tem), 0);
// 第二个参数指向发送缓冲区，第三个参数为要发送的字节数，第四个参数一般置 0，返
// 回值：>=0 实际发送的字节数，0 对方正常关闭，SOCKET_ERROR 出错。

        closesocket(ssock);                                    // 关闭套接字
    }

    closesocket(msock);                                        // 关闭监听套接字

    WSACleanup();
    // 卸载 winsock library

    return 0;
}
```

- **客户端的全部源代码（或自选主要代码）：**

```
#include <stdio.h>
#include <stdlib.h>
```

```
#include <string.h>
#include <winsock2.h>
#include <time.h>
#include "conio.h"                  // 仅能用于 Dos 类操作系统
#pragma comment(lib,"ws2_32.lib")  // 使用 winsock 2.2 library
#pragma warning(disable : 4996)
#define WSVERS  MAKEWORD(2, 2)
// MAKEWORD 函数创建一个符合网络传输（小端？）的 2B 的 word，这里的参数表示后面
// 的 WSAStartup 将使用版本号 2.2（换成 2.0 也行）
#define SERVICE "50500"
#define HOST "127.0.0.1"
#define BUFFSIZE 100
int main(int argc, char** argv)
{
    WSADATA wsadata;
    WSAStartup(WSVERS, &wsadata);
    // 加载 winsock library，SVERS 指明请求使用的版本。wsadata 返回系统实际支
    // 持的最高版本

    SOCKET sock;
    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（这个换
    // 成 0 也行）
    // 返回：要监听套接字的描述符或 INVALID_SOCKET
    // sockaddr_in 与 sockaddr 二者的占用的内存大小是一致的，因此可以互相转化，
    // 从这个意义上说，他们并无区别
    struct  sockaddr_in sin;
    // 但 sockaddr_in 更方便程序员操作，而 sockaddr 更通用。后面相关 size 操作实
    //  际是要 sockaddr 的大小，但为方便直接取了 sockaddr_in 的大小
    // 惯例上会将先操作 sockaddr_in，再转化为 sockaddr
    memset(&sin, 0, sizeof(sin));                       // 将 sin 清 0
    sin.sin_family = AF_INET;
    // 因特网地址簇(INET-Internet)
    sin.sin_addr.s_addr = inet_addr(HOST);              // 设置 IP 地址
    sin.sin_port = htons((u_short)atoi(SERVICE));
    // 监听的端口号。htons--主机序到网络序(host to network，s-short 16 位)，
    // 类似 MAKEWORD
    int ret = connect(sock, (struct sockaddr*) & sin, sizeof(sin));
    // 连接到服务器，第二个参数指向存放服务器地址的结构，第三个参数为该结构的大
    //  小，返回值为 0 时表示无错误发生，返回 SOCKET_ERROR 表示出错，应用程序可
    //  通过 WSAGetLastError()获取相应错误代码。

    char buffer[BUFFSIZE];
    memset(buffer, 0, BUFFSIZE);
    printf("请输入要发送的消息：");
    scanf("%s", buffer);
```

```
    getchar();
    int cc = send(sock, buffer, strlen(buffer), 0);
    if (cc == SOCKET_ERROR)
    // 出错。其后必须关闭套接字 sock
        printf("Error: %d.\n", GetLastError());
    else if (cc == 0) {                                  // 对方正常关闭
        printf("Server closed!", buffer);
    }

    cc = recv(sock, buffer, BUFFSIZE, 0);
    // 第二个参数指向缓冲区，第三个参数为缓冲区大小(字节数)，第四个参数一般设置
    // 为 0，返回值:(>0)接收到的字节数,(=0)对方已关闭,(<0)连接出错
    if (cc == SOCKET_ERROR)
    // 出错。其后必须关闭套接字 sock
        printf("Error: %d.\n", GetLastError());
    else if (cc == 0) {                                  // 对方正常关闭
        printf("Server closed!", buffer);
    }
    else if (cc > 0) {
        buffer[cc] = '\0';
    // ensure null-termination
        printf("\n 收到的消息:
\n%s\n", buffer);                              // 显示所接收的字符串
    }
    printf("按回车键继续: \n");
    getchar();

    closesocket(sock);                                   // 关闭监听套接字

    WSACleanup();
    // 卸载 winsock library
    return 0;
}
```

(2)编写 TCP Echo 增强程序

- **实验要求：**

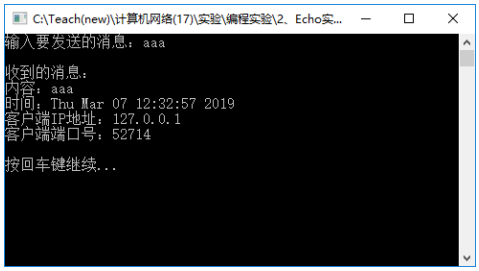
  在(1)的基础上，**服务器**在收到客户端的消息时显示服务器的当前时间、客户端的 IP 地址、客户端的端口号和客户端发来的信息，并把它们一并返回给客户端。

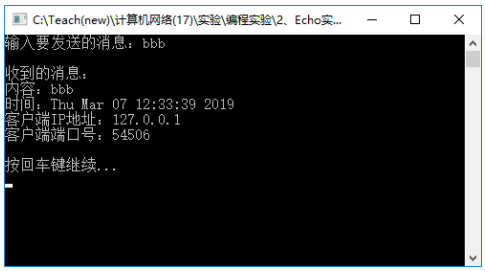
  客户端在发送消息后把服务器发回给它的消息显示出来。*客户端程序与(1)同，不用修改

  要求**服务器**直接从 accept()的参数 fsin 中得到客户端的 IP 地址和端口号。注：服务器获取 IP 地址后要求直接使用 s_un_b 的四个分量得到 IP 地址，不能使用函数 inet_ntoa()转换 IP 地址。

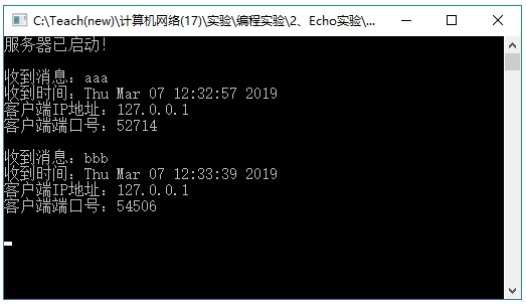
- **参考运行截屏：**

  客户端（两次运行）





服务器

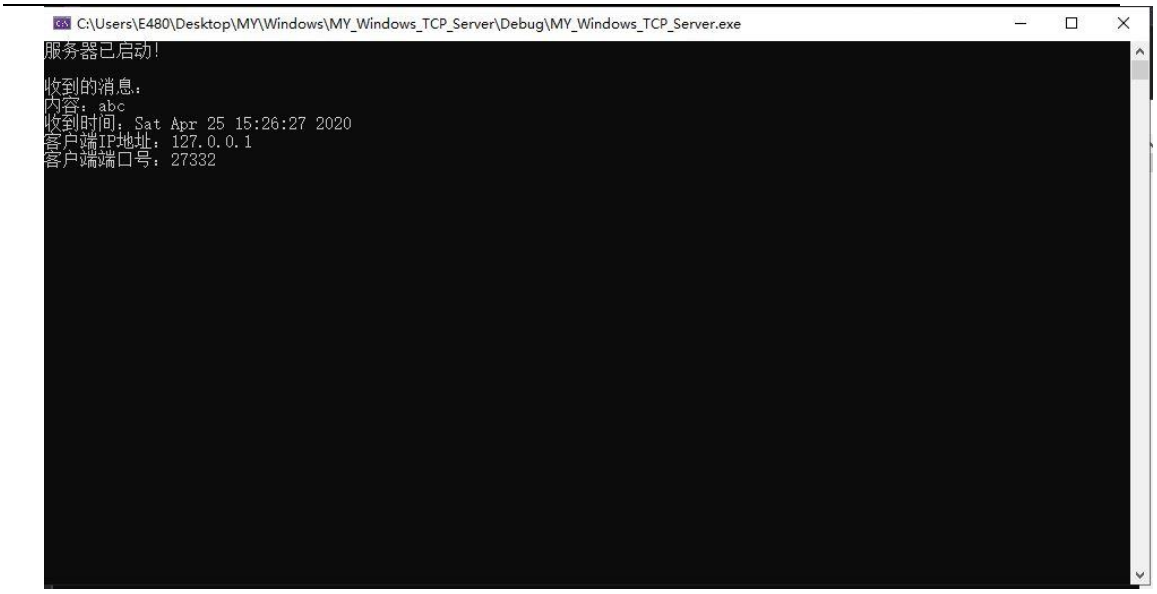


▪ 截屏服务器和客户端的运行结果(注明客户端和服务器)：

客户端：

服务器：

实验报告



▪ **服务器的全部源代码（或自选主要代码）:**

与上一问唯一不同的只是循环中对字符串的处理，注意这里的宏定义 BUFFSIZE 改为了 200（客户端也修改了）

```
while (!_kbhit())
// 定义在 conio.h，若有按键，结束循环
    {
        struct  sockaddr_in fsin;
        int hd_sz = sizeof(fsin);
        ssock = accept(msock, (struct sockaddr*) & fsin, &hd_sz);
// 建立连接，创建新套接字，fsin 存放的是客户端的 sockaddr，hd_sz 为大小的指针

        char buffer[BUFFSIZE];
        memset(buffer, 0, BUFFSIZE);
        int cc = recv(ssock, buffer, BUFFSIZE, 0);
        if (cc == SOCKET_ERROR)
// 出错。其后必须关闭套接字 sock
            printf("Error: %d.\n", GetLastError());
        else if (cc == 0) {                                  // 对方正常关闭
            printf("Server closed!", buffer);
        }
        else if (cc > 0) {
            buffer[cc] = '\0';
// ensure null-termination


            char tem[BUFFSIZE];
            memset(tem, 0, BUFFSIZE);
            time_t now = time(0);
            char* curr = ctime(&now);
            unsigned int ip = fsin.sin_addr.S_un.S_addr;
```

```
            printf("收到的消息：\n 内容：%s\n 收到时间：%s 客户端 IP 地
址：%d.%d.%d.%d\n 客户端端口号：%d\n",
                buffer, curr, (ip << 24) >> 24, (ip << 16) >> 24, (ip <
< 8) >> 24, ip >> 24, fsin.sin_port);
            sprintf(tem, "收到的消息：\n 内容：%s\n 收到时间：%s 客户端 IP 地
址：%d.%d.%d.%d\n 客户端端口号：%d\n",
                buffer, curr, (ip << 24) >> 24, (ip << 16) >> 24, (ip <
< 8) >> 24, ip >> 24, fsin.sin_port);
            cc = send(ssock, tem, strlen(tem), 0);
// 第二个参数指向发送缓冲区，第三个参数为要发送的字节数，第四个参数一般置 0，返
回值：>=0 实际发送的字节数，0 对方正常关闭，SOCKET_ERROR 出错。
        }
```

(3)编写 UDP Echo 增强程序

- **实验要求：**

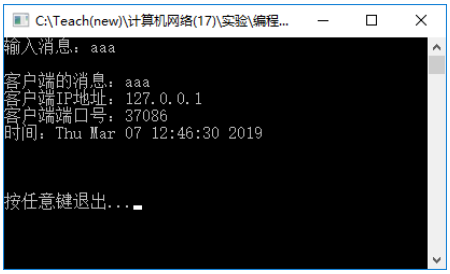
  修改 UDP 例程，完成 Echo 功能，即当客户端发来消息时，服务器显示出服务器的当前时间、客户端的 IP、客户端的端口号和客户发来的信息，并把它们一并发回给客户端，客户端然后把它们显示出来。

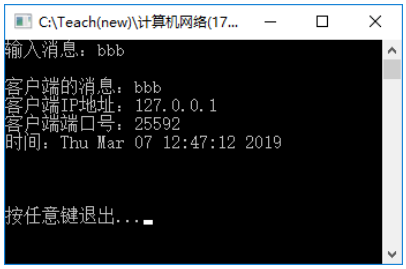
  服务器可以直接从 recvfrom()的参数 from 中得到客户端的 IP 地址和端口号，并且服务器用 sendto()发回给客户端消息时可以直接用该参数 from 作为参数 toAddr。可以使用 inet_ntoa()转换客户端 IP 地址。

  客户端程序的 recvfrom()可以直接使用原来 sendto 使用的 sock。该 sock 已经绑定了客户端的 IP 地址和端口号，客户端可以直接用来接收数据。

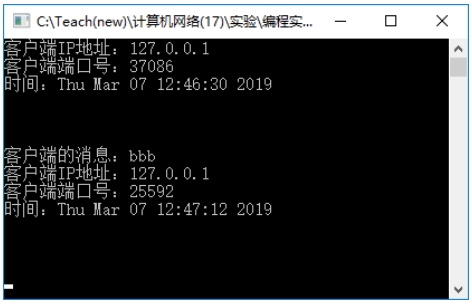
- **参考运行截屏：**

  客户端（两次运行）





  服务器：



- **只运行客户端程序而不运行服务器程序会出现什么错误，截屏并说明原因。**

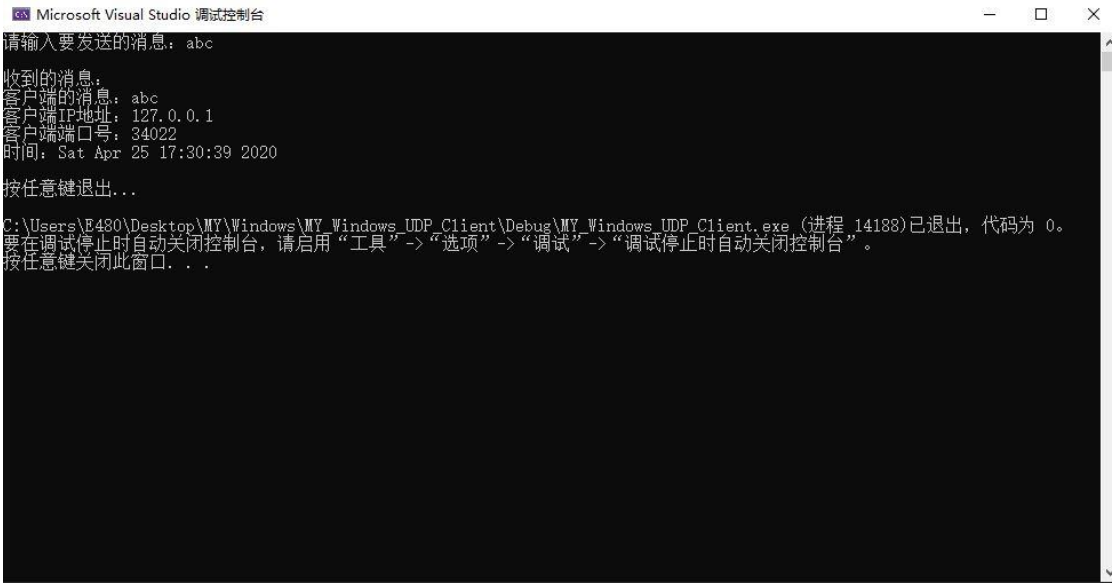
实验报告

Socket error 10054 – connection reset by peer //连接被重置

该错误常发生在服务器出现问题时。和 TCP 不同的是，没有在信息发送时显示错误，而是在接受信息的地方发生错误。

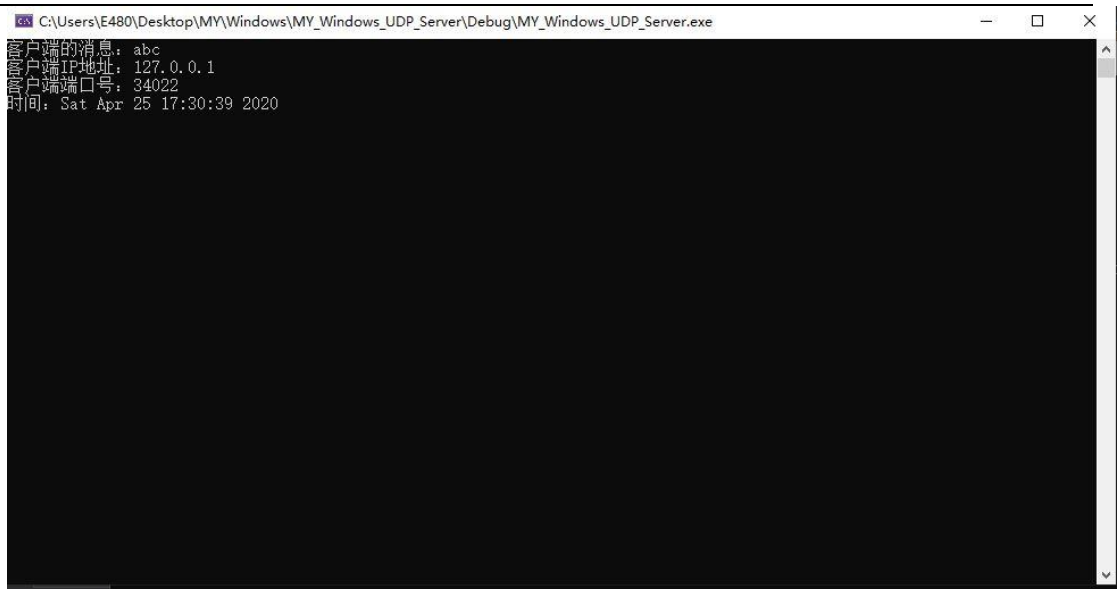
- **截屏服务器和客户端的运行结果（注明客户端和服务器）：**

客户端：



服务器：

实验报告



- 服务器的全部源代码（或自选主要代码）：

```
/* UDPServer.cpp */

#include <stdlib.h>
#include <stdio.h>
#include <winsock2.h>
#include <string.h>
#include <time.h>
#include "conio.h"
#pragma warning(disable : 4996)

#define BUFLEN      2000                    // 缓冲区大小
#define WSVERS      MAKEWORD(2, 2)          // 指明版本 2.2
#pragma comment(lib,"ws2_32.lib")           // 加载 winsock 2.2 Llibrary

/*-------------------------------------------------------------------
---
 * main - TCP client for DAYTIME service
 *-------------------------------------------------------------------
---
 */
void
main(int argc, char* argv[])
{
    char str1[BUFLEN] = "127.0.0.1";
    char* host = str1;      /* server IP Address to connect */
    char str2[BUFLEN] = "50500";
    char* service = str2;       /* server port to connect       */
    struct sockaddr_in sin;         /* an Internet endpoint address */
    struct sockaddr_in from;        /* sender address               */
    int    fromsize = sizeof(from);
```

```
    char   buf[BUFLEN + 1];          /* buffer for one line of text  */
    SOCKET  sock;                    /* socket descriptor            */
    int cc;                          /* recv character count         */

    WSADATA wsadata;
    WSAStartup(WSVERS, &wsadata);   /* 加载 winsock library，WSVERS 为请求版本,wsadata 返回系统实际支持的最高版本。     */
    sock = socket(PF_INET, SOCK_DGRAM, IPPROTO_UDP); // 创建 UDP 套接字，参数：因特网协议簇(family)，数据报套接字，UDP 协议号， 返回：要监听套接字的描述符或 INVALID_SOCKET
    memset(&sin, 0, sizeof(sin));
    sin.sin_family = AF_INET;
    sin.sin_addr.s_addr = INADDR_ANY;                       // 绑定(监听)所有的接口。
    sin.sin_port = htons((u_short)atoi(service));           // 绑定指定接口。atoi--把 ascii 转化为 int，htons -- 主机序(host)转化为网络序(network)，为 short 类型。
    bind(sock, (struct sockaddr*) & sin, sizeof(sin));      // 绑定本地端口号（和本地 IP 地址)

    while (!_kbhit()) {                                     //检测是否有按键
        cc = recvfrom(sock, buf, BUFLEN, 0, (SOCKADDR*)&from, &fromsize); //接收客户数据。返回结果：cc 为接收的字符数，from 中将包含客户 IP 地址和端口号。
        if (cc == SOCKET_ERROR) {
            printf("recvfrom() failed; %d\n", WSAGetLastError());
            break;
        }
        else if (cc == 0)
            break;
        else {
            buf[cc] = '\0';
            char tem[BUFLEN];
            memset(tem, 0, BUFLEN);
            time_t now = time(0);
            char* curr = ctime(&now);

            printf("客户端的消息：%s\n 客户端 IP 地址：%s\n 客户端端口号：%d\n 时间：%s",
                buf, inet_ntoa(from.sin_addr), from.sin_port, curr);
            sprintf(tem, "客户端的消息：%s\n 客户端 IP 地址：%s\n 客户端端口号：%d\n 时间：%s",
                buf, inet_ntoa(from.sin_addr), from.sin_port, curr);
            cc = sendto(sock, tem, strlen(tem), 0, (SOCKADDR*)&from, fromsize);
```

```
        }
    }
    closesocket(sock);
    WSACleanup();                       /* 卸载某版本的 DLL */
    getchar();
}
```

▪ 客户端的全部源代码（或自选主要代码）：

```
/* UDPClient.cpp */

#include <stdlib.h>
#include <stdio.h>
#include <winsock2.h>
#include <string.h>
#include <time.h>
#pragma warning(disable : 4996)
#define BUFLEN      2000                    // 缓冲区大小
#define WSVERS      MAKEWORD(2, 2)          // 指明版本 2.2
#pragma comment(lib,"ws2_32.lib")           // 加载 winsock 2.2 Llibrary

void
main(int argc, char* argv[])
{
    char str1[BUFLEN] = "127.0.0.1";
    char* host = str1;      /* server IP Address to connect */
    char str2[BUFLEN] = "50500";
    char* service = str2;       /* server port to connect       */
    struct sockaddr_in toAddr;          /* an Internet endpoint address
 */
    int toAddrsize = sizeof(toAddr);
    char    buf[BUFLEN + 1];            /* buffer for one line of text
 */
    SOCKET  sock;                       /* socket descriptor
 */
    int cc;                             /* recv character count
 */
    char* pts;                      /* pointer to time string       */
    time_t  now;                        /* current time
 */

    WSADATA wsadata;
    WSAStartup(WSVERS, &wsadata);       /* 启动某版本 Socket 的
DLL        */

    sock = socket(PF_INET, SOCK_DGRAM, IPPROTO_UDP);
```

```
    memset(&toAddr, 0, sizeof(toAddr));
    toAddr.sin_family = AF_INET;
    toAddr.sin_port = htons((u_short)atoi(service));    //atoi：把 ascii 转化为 int. htons：主机序(host)转化为网络序(network), s--short
    toAddr.sin_addr.s_addr = inet_addr(host);           //如果 host 为域名，需要先用函数 gethostbyname 把域名转化为 IP 地址


    memset(buf, 0, BUFLEN);
    printf("请输入要发送的消息：");
    scanf("%s", buf);
    getchar();

    cc = sendto(sock, buf, BUFLEN, 0, (SOCKADDR*)&toAddr, sizeof(toAddr));
    if (cc == SOCKET_ERROR) {
        printf("发送失败，错误号：%d\n", WSAGetLastError());
    }
    else {
        //printf("发送成功!\r\n");
    }

    cc = recvfrom(sock, buf, BUFLEN, 0, (SOCKADDR*)&toAddr, &toAddrsize);          // 第二个参数指向缓冲区，第三个参数为缓冲区大小(字节数)，第四个参数一般设置为 0，返回值:(>0)接收到的字节数,(=0)对方已关闭,(<0)连接出错
    if (cc == SOCKET_ERROR)                                 // 出错。其后必须关闭套接字 sock
        printf("Error: %d.\n", GetLastError());
    else if (cc == 0) {                                 // 对方正常关闭
        printf("Server closed!", buf);
    }
    else if (cc > 0) {
        buf[cc] = '\0';                         // ensure null-terminati
on
        printf("\n 收到的消息:
\n%s\n", buf);                          // 显示所接收的字符串
    }
    closesocket(sock);
    WSACleanup();                       /* 卸载某版本的 DLL */

    printf("按任意键退出...");
    getchar();

}
```

【完成情况】

是否完成以下步骤？（√完成 ×未做）

(1) [√] (2) [√] (3) [√]

【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）。

第一次接触套接字编程，刚开始比较困难，直接运行老师的程序后，看到了一些结果，很神奇。在做第一个实验时以为老师的程序仅是参考，不能复制部分代码，又是假期，所以自己在 MOOC 上学习了计网的内容，第一个程序基本上是自己重写的。这次实验其实不是很困难，只要熟悉了套接字编程的套路和一些相关结构体的知识，就能把问题转化为简单的字符串处理问题。第二个实验和第三个实验是在老师给的参考例程基础上直接修改得到的。

阻塞的概念曾在 Verilog 编程中接触过，也不算太难理解。

【交实验报告】

每位同学单独完成本实验内容并填写实验报告。

交作业地点：http://172.18.187.251/netdisk/default.aspx?vm=18net

编程实验

截止日期： 开学第二周的周五 23:00

上传文件： 学号_姓名_Echo 实验报告.doc

学号_姓名_Echo 实验源码.rar (源程序和可执行程序)