实验报告

院 系 数据科学与计算机学院 班 级 19 学 号 19335174 姓 名 施天予

【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。

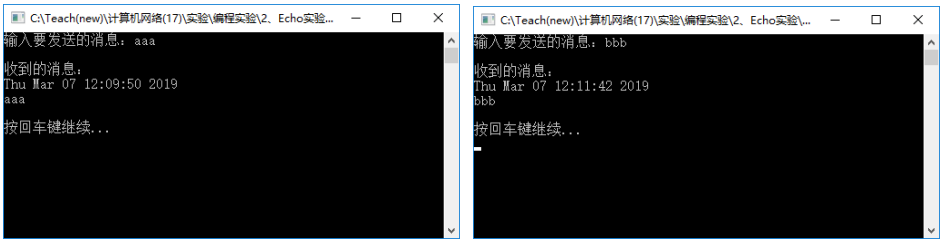
(1)编写 TCP Echo 程序

- **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。

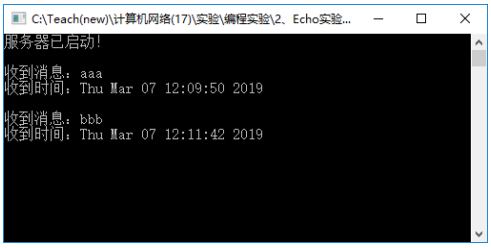
- **参考运行截屏：**

  客户端（两次运行）

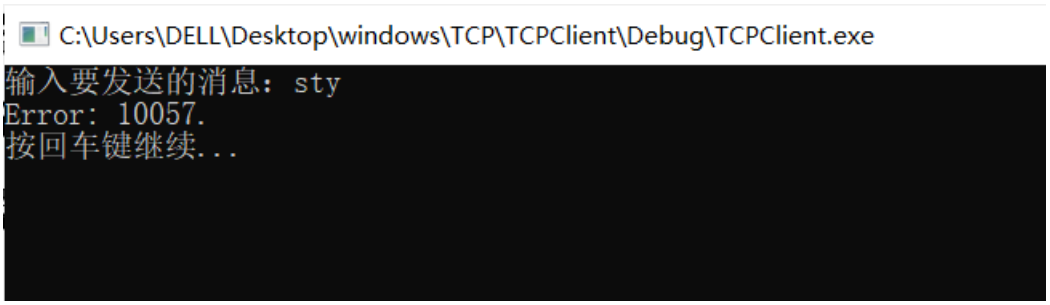


  服务器：

实验报告



- **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。**

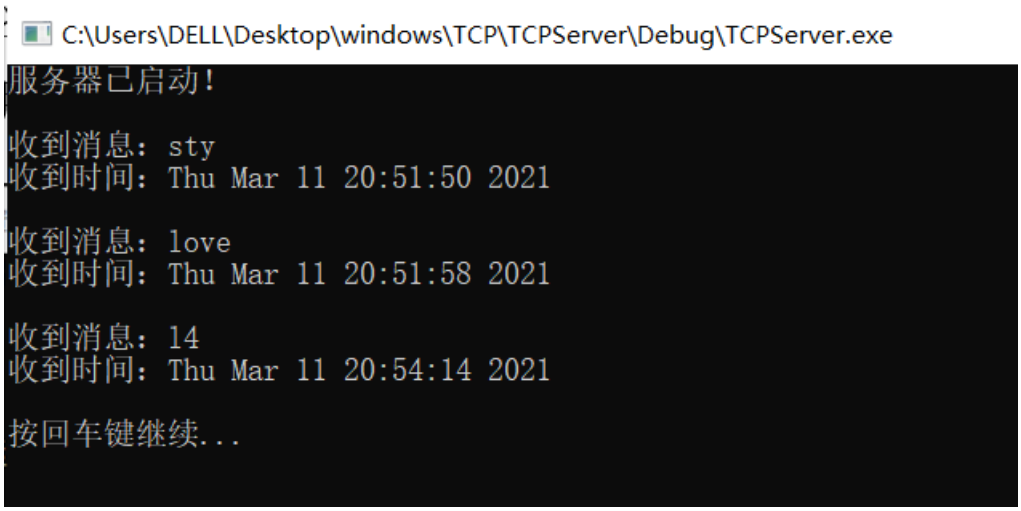


错误码 10057 表示套接字未连接

- **服务器如何可以退出循环？**

```
while(!_kbhit()){                                    // 检测是否有按键,如果没有则进入循环体执行
```

循环条件为 while(!_kbhit())，即有键盘输入时跳出循环。当我在 TCPserver.exe 中尝试键入时，跳出了“按回车键继续…”



而 printf(“按回车键继续…”);在 while 循环外

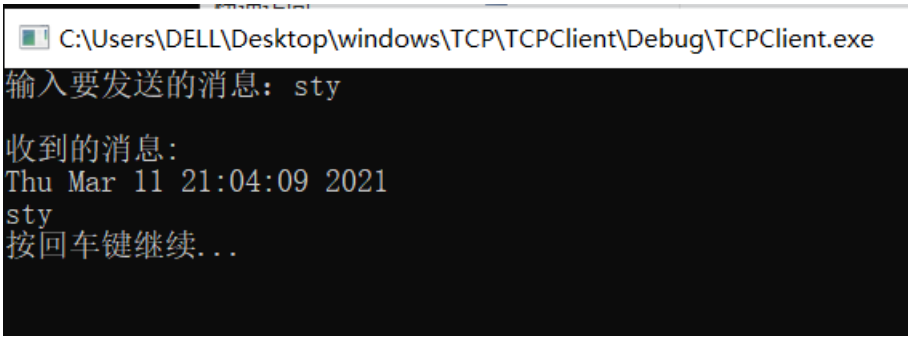
```
}
(void) closesocket(msock);                           // 关闭监听套接字
WSACleanup();                                        // 卸载winsock library
printf("按回车键继续...");
```

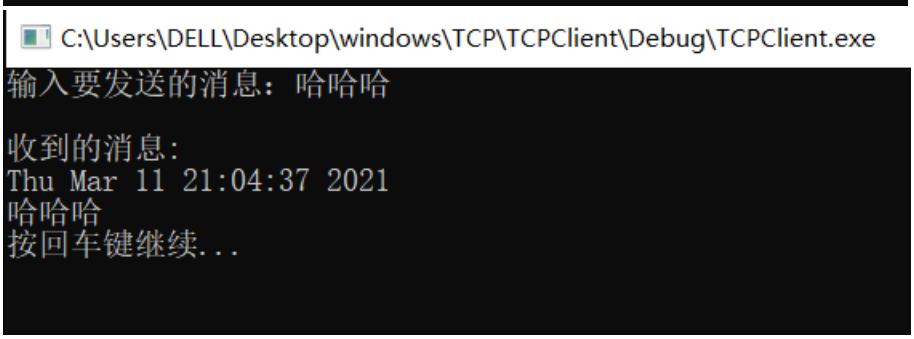
所以此时服务器已退出循环

实验报告

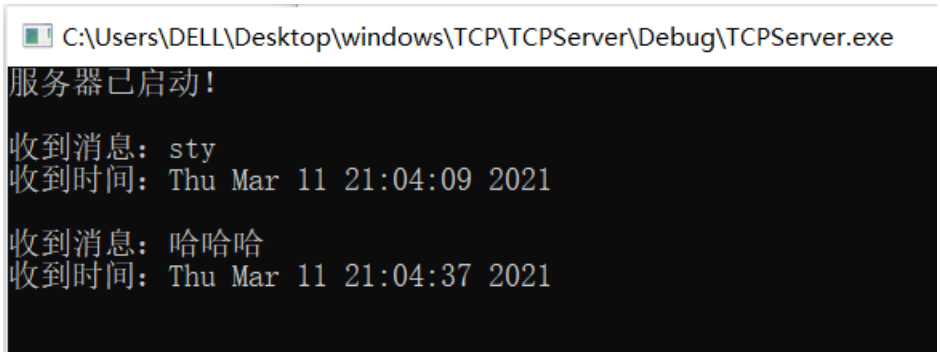
- **截屏（ctrl+alt+PrintScreen）服务器和客户端的运行结果（注明客户端和服务器）：**

客户端（两次运行）





服务器：



- **服务器的全部源代码（或自选主要代码）：**

```
/* TCPServer.cpp - main */

#include <stdlib.h>
#include <stdio.h>
#include <winsock2.h>
#include <time.h>
#include "conio.h"

#define  WSVERS   MAKEWORD(2, 0)
#define BUFLEN 2000
#pragma comment(lib,"ws2_32.lib")  //使用winsock 2.2 library
/*------------------------------------------------------------------------
 * main - Iterative TCP server for TIME service
 *------------------------------------------------------------------------
 */
void
```

```
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, ssock;         /* master & slave sockets        */
   WSADATA wsadata;
   char  *service = "50500";
   struct  sockaddr_in sin;        /* an Internet endpoint address     */
    int      alen;                    /* from-address length             */
   char  *pts;                     /* pointer to time string          */
   time_t    now;                  /* current time                        */
   char  buf[BUFLEN + 1];

   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(host to network，s-short 16位)
   bind(msock, (struct sockaddr *)&sin, sizeof(sin));  // 绑定监听的IP地址和端口号

   listen(msock, 5);                                          // 建立长度为5的连接请求队列，并把到来的连接请求加入队列等待处理。
   printf("服务器已启动！\n\n");

    while(!_kbhit()){                                         // 检测是否有按键,如果没有则进入循环体执行
       alen = sizeof(struct sockaddr);                        // 取到地址结构的长度
      ssock = accept(msock, (struct sockaddr *)&fsin, &alen); // 如果在连接请求队列中有连接请求，则接受连接请求并建立连接，返回该连接的套接字，否则，本语句被阻塞直到队列非空。fsin包含客户端IP地址和端口号
      int ccc = recv(ssock, buf, BUFLEN, 0);
      printf("收到消息：%s\n", buf);
       (void) time(&now);                                     // 取得系统时间
       pts = ctime(&now);                                     // 把时间转换为字符串
      printf("收到时间：%s\n", pts);
```

```
        sprintf(pts, "%s%s", pts, buf);
        int cc = send(ssock, pts, strlen(pts), 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。
        //printf("%s", pts);                                    // 显示发送的字符串
        (void) closesocket(ssock);                              // 关闭连接套接字
      }
     (void) closesocket(msock);                                 // 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
     getchar();
}
```

客户端的全部源代码（或自选主要代码）：

```
/* TCPClient.cpp */

#include <stdlib.h>
#include <stdio.h>
#include <winsock2.h>
#include <string.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
 *-------------------------------------------------------------------
 */
void main(int argc, char *argv[])
{
    char  *host = "127.0.0.1";      /* 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  */
    SOCKET     sock;                        /* socket descriptor        */
    int   cc;                              /* recv character count         */

    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);           // 设置服务器IP地址(32位)
    sin.sin_port = htons((u_short)atoi(service));    // 设置服务器端口号
    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 str[BUFLEN + 1];
    printf("输入要发送的消息：");
    scanf("%s", str);
    send(sock, str, BUFLEN, 0);

    cc = recv(sock, buf, BUFLEN, 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!",buf);
    }  else if(cc > 0) {
        buf[cc] = '\0';                             // ensure null-termination
        printf("\n收到的消息:\n%s\n", buf);          // 显示所接收的字符串
    }
    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
   getchar();
}
```

(2)编写 TCP Echo 增强程序

- **实验要求：**

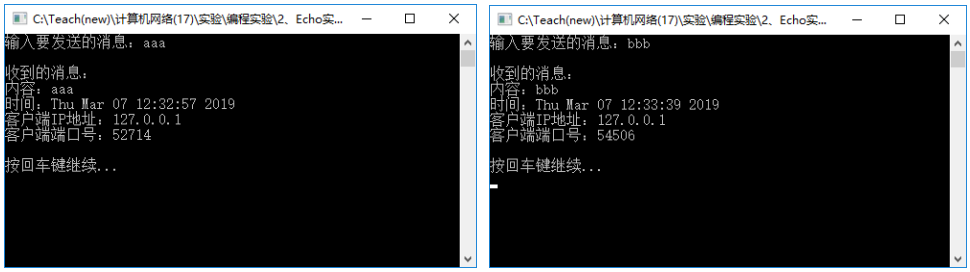
  在(1)的基础上，**服务器**在收到客户端的消息时显示服务器的当前时间、客户端的 IP 地址、客户端的端口号和客户端发来的信息，并把它们一并返回给客户端。

  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

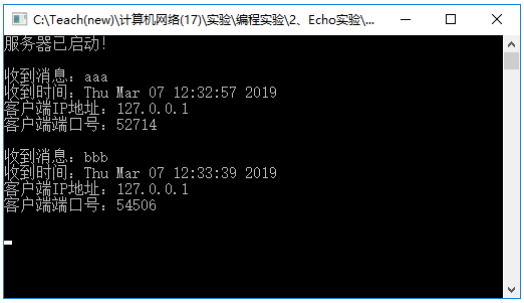
  要求**服务器**直接从 accept()的参数 fsin 中得到客户端的 IP 地址和端口号。注：服务器获取 IP 地址后要求直接使用 s_un_b 的四个分量得到 IP 地址，不能使用函数 inet_ntoa()转换 IP 地址。

- **参考运行截屏：**

客户端（两次运行）

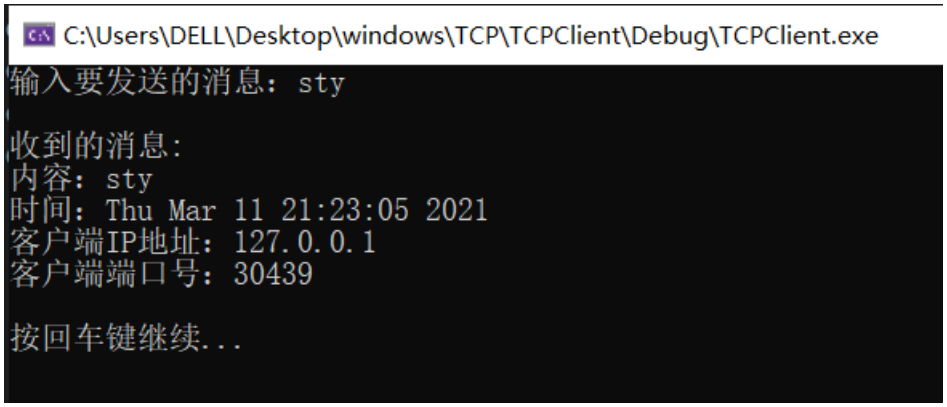


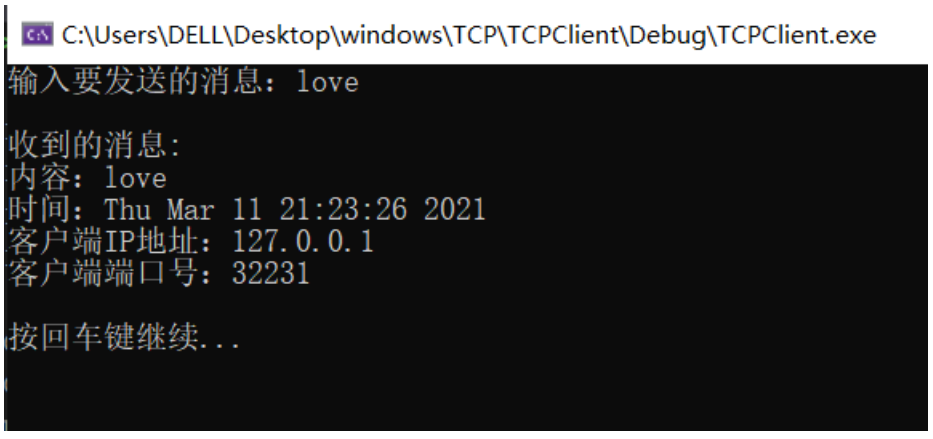
服务器



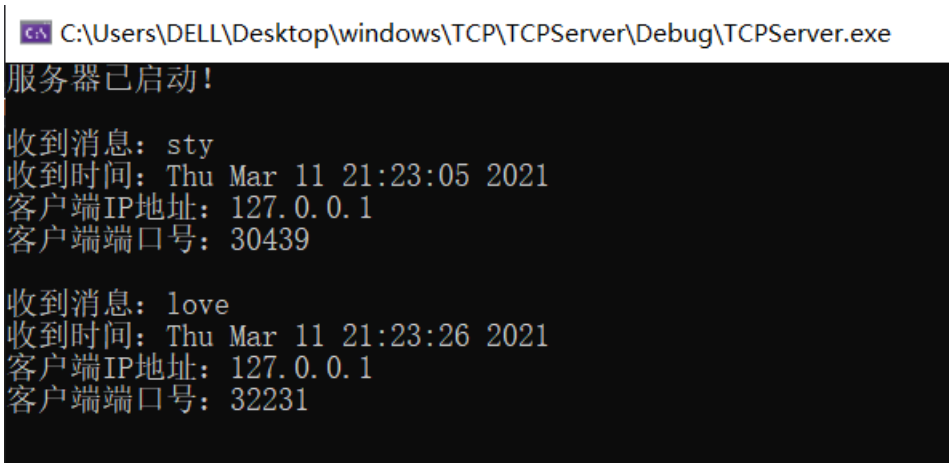
- **截屏服务器和客户端的运行结果(注明客户端和服务器)：**

客户端（两次运行）





服务器



▪ **服务器的全部源代码（或自选主要代码）：**

```
/* TCPServer.cpp - main */

#include <stdlib.h>
#include <stdio.h>
#include <winsock2.h>
#include <time.h>
#include "conio.h"

#define  WSVERS   MAKEWORD(2, 0)
#define BUFLEN 2000
#pragma comment(lib,"ws2_32.lib")  //使用winsock 2.2 library
/*------------------------------------------------------------------------
 * main - Iterative TCP server for TIME service
 *------------------------------------------------------------------------
 */
void 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, ssock;         /* master & slave sockets         */
    WSADATA wsadata;
    char *service = "50500";
    struct  sockaddr_in sin;       /* an Internet endpoint address     */
    int      alen;                     /* from-address length              */
    char *pts;                    /* pointer to time string           */
    time_t   now;                 /* current time                         */
    char buf[BUFLEN + 1];

    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(host to network，s-short 16位)
    bind(msock, (struct sockaddr *)&sin, sizeof(sin));    // 绑定监听的IP地址和端口号

    listen(msock, 5);                                     // 建立长度为5的连接请求队列，并把到来的连接请求加入队列等待处理。
    printf("服务器已启动！\n\n");

   while(!_kbhit()){                                      // 检测是否有按键,如果没有则进入循环体执行
       alen = sizeof(struct sockaddr);                    // 取到地址结构的长度
       ssock = accept(msock, (struct sockaddr *)&fsin, &alen); // 如果在连接请求队列中有连接请求，则接受连接请求并建立连接，返回该连接的套接字，否则，本语句被阻塞直到队列非空。fsin包含客户端IP地址和端口号
       int ccc = recv(ssock, buf, BUFLEN, 0);

       printf("收到消息：%s\n", buf);
       (void) time(&now);                                 // 取得系统时间
       pts = ctime(&now);                                 // 把时间转换为字符串
       printf("收到时间：%s", pts);
       printf("客户端IP地址：%d.%d.%d.%d\n", fsin.sin_addr.S_un.S_un_b.s_b1, fsin.sin_addr.S_un.S_un_b.s_b2, fsin.sin_addr.S_un.S_un_b.s_b3, fsin.sin_addr.S_un.S_un_b.s_b4);
       printf("客户端端口号：%d\n\n", fsin.sin_port);
       char total[BUFLEN + 1];
       sprintf(total, "内容：%s\n时间：%s客户端IP地址：%d.%d.%d.%d\n客户端端口号：%d\n", buf, pts, fsin.sin_addr.S_un.S_un_b.s_b1, fsin.sin_addr.S_un.S_un_b.s_b2, fsin.sin_addr.S_un.S_un_b.s_b3, fsin.sin_addr.S_un.S_un_b.s_b4, fsin.sin_port);
       int cc = send(ssock, total, strlen(total), 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。

       (void) closesocket(ssock);                         // 关闭连接套接字
   }
   (void) closesocket(msock);                             // 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
    getchar();
}
```

(3)编写 UDP Echo 增强程序

- **实验要求：**

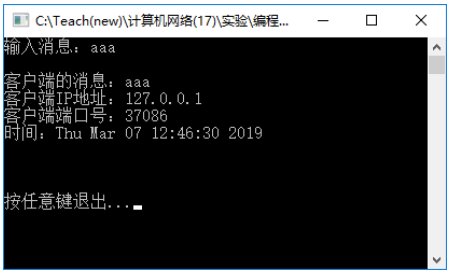
  修改 UDP 例程，完成 Echo 功能，即当客户端发来消息时，服务器显示出服务器的当前时间、客户端的 IP、客户端的端口号和客户发来的信息，并把它们一并发回给客户端，客户端然后把它们显示出来。

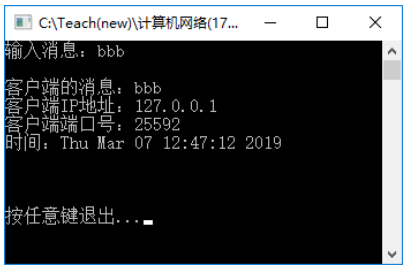
  服务器可以直接从 recvfrom()的参数 from 中得到客户端的 IP 地址和端口号，并且服务器用 sendto()发回给客户端消息时可以直接用该参数 from 作为参数 toAddr。可以使用 inet_ntoa()转换客户端 IP 地址。

  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。

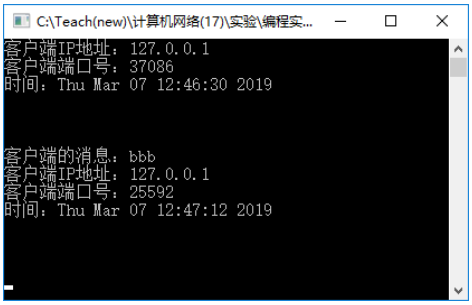
- **参考运行截屏：**

  客户端（两次运行）





  服务器：



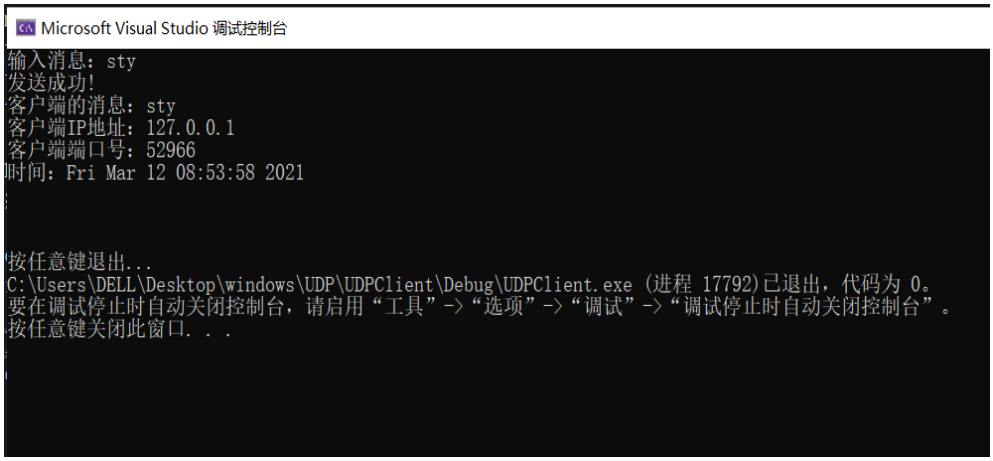
- **只运行客户端程序而不运行服务器程序会出现什么错误，截屏并说明原因。**

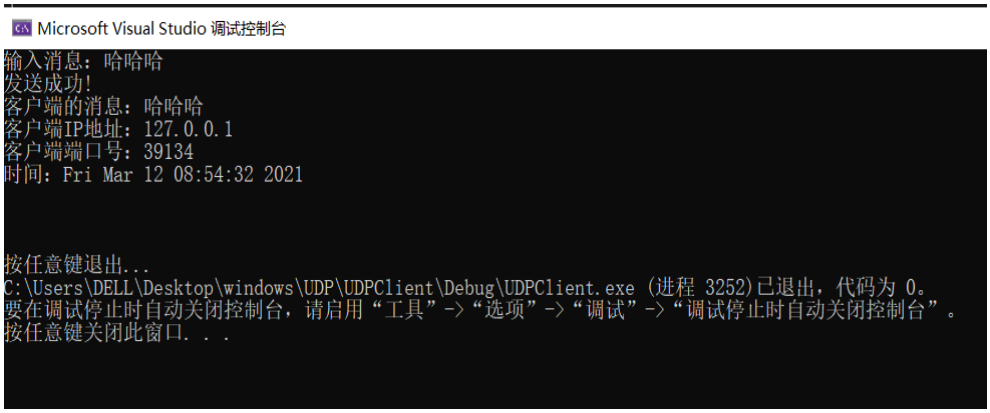
Socket error 10054 - Connection reset by peer

错误码 10054 表示“对等端重置连接”，因为是 UDP 所以发送成功，但没有运行服务器程序所以不会收到回复。

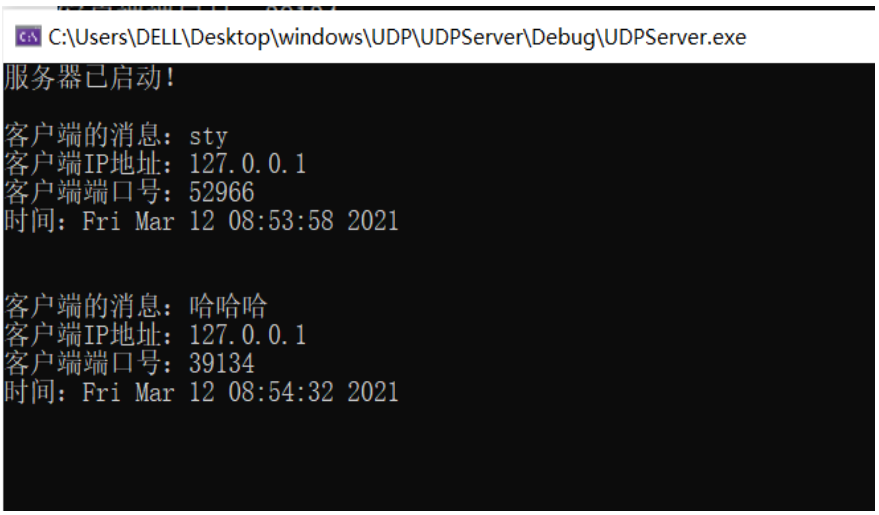
- **截屏服务器和客户端的运行结果（注明客户端和服务器）：**

客户端（两次运行）





服务器：



- **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 "conio.h"
#include <time.h>


#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   *host = "127.0.0.1";        /* server IP Address to connect */
    char   *service = "50500";         /* 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         */
    char* pts;                              /* pointer to time string      */
    time_t   now;                     /* current time                    */

    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）

    printf("服务器已启动！\n\n");
    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。
        (void)time(&now);                                // 取得系统时间
```

```
            pts = ctime(&now);                                    // 把时间转换为字符串
            if (cc == SOCKET_ERROR){
                printf("recvfrom() failed; %d\n", WSAGetLastError());
                break;
            }
            else if (cc == 0)
                break;
            else{
                buf[cc] = '\0';
                char total[BUFLEN + 1];
                sprintf(total, "客户端的消息：%s\n客户端IP地址：%d.%d.%d.%d\n客户端端口
号：%d\n时间：%s\n", buf, from.sin_addr.S_un.S_un_b.s_b1, from.sin_addr.S_un.S_un_b.s_b2,
from.sin_addr.S_un.S_un_b.s_b3, from.sin_addr.S_un.S_un_b.s_b4, from.sin_port, pts);
                printf("%s\n", total);
                sendto(sock, total, BUFLEN, 0, (SOCKADDR*)&from, sizeof(from));
            }
        }
        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>

#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  *host = "127.0.0.1";       /* server IP to connect         */
    char  *service = "50500";        /* server port to connect       */
    struct sockaddr_in toAddr;           /* an Internet endpoint address */
    char  buf[BUFLEN+1];             /* buffer for one line of text  */
    SOCKET     sock;                          /* socket descriptor        */
    int   cc;                                /* recv character count         */

    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

   printf("输入消息：");
   scanf("%s", buf);

   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");
     int size = sizeof(toAddr);
     int ccc = recvfrom(sock, buf, BUFLEN, 0, (SOCKADDR*)&toAddr, &size);
     if (ccc == SOCKET_ERROR)
         printf("接收失败，错误号：%d\n\n", WSAGetLastError());
     else
         printf("%s\n\n", buf);
   }

   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）

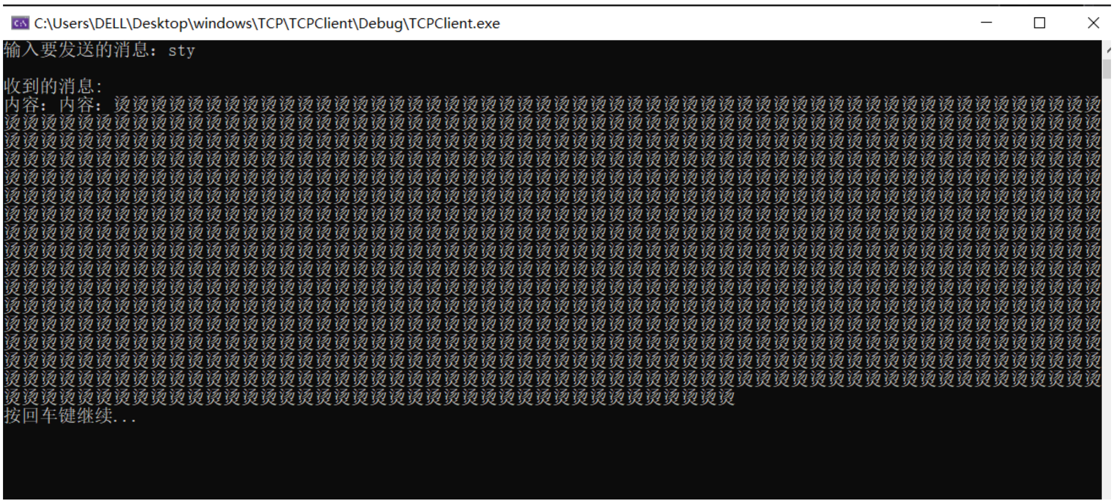
(1) [√] (2) [√] (3) [√]

【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）。

这次实验刚开始时把我吓到了，各种复杂的函数看得我头晕眼花，但仔细阅读程序之后发现其实也还好，自己要写的部分其实不是很多，关键是要读懂程序。在做 TCP Echo 增强实验时，我一开始用 sprintf 把所有信息合并到接收消息的 buf 字符串，出现了下面这种奇怪的情况，但我后来用一个新

的 total 字符串来合并消息就成功了，也不知道是什么原因。总体来说，这次实验对我受益匪浅，希望以后继续努力！



【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=19net

编程实验

截止日期： 3 月 16 日 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）