



院 系 数据科学与计算机学院 班 级 17 计科 学 号 17341097 姓 名 廖永滨

【实验题目】Echo 实验

【实验目的】掌握套接字的基本使用方法。

【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)

【实验环境】

- ♦ Windows + VS 2012 <http://172.18.187.9/netdisk/default.aspx?vm=17net>
- ♦ 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

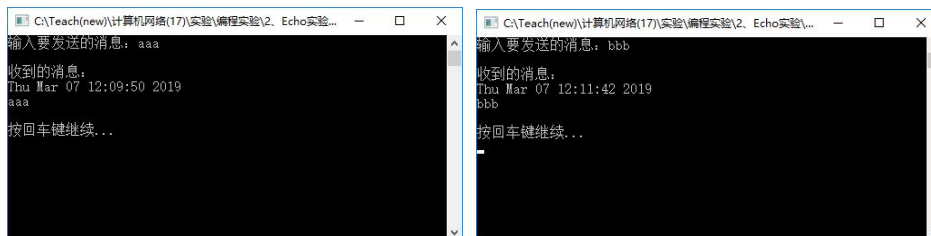
▪ 实验要求:

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。

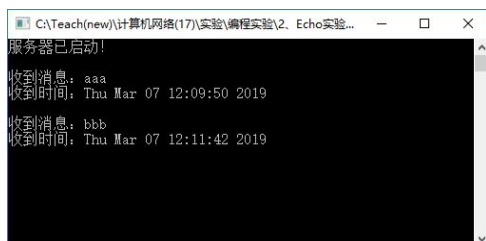
客户端把返回的消息显示出来。客户端每输入一条消息就建立 TCP 连接, 并把消息发送给服务器, 在收到服务器回应后关闭连接。

▪ 参考运行截屏:

客户端 (两次运行)

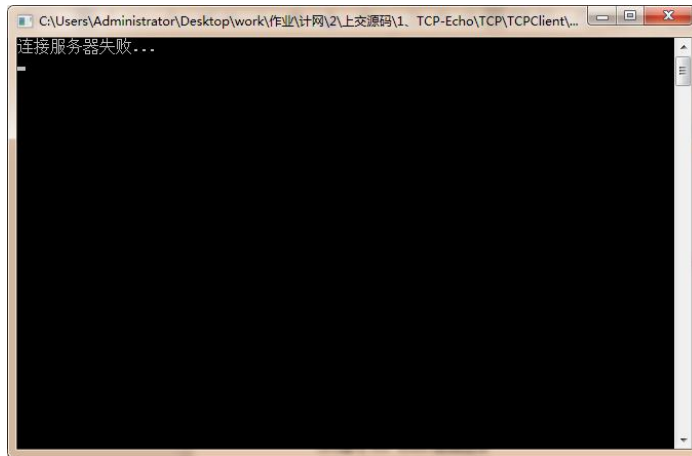


服务器:





- 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。



`int ret=connect(...)` ;语句在找不到对应服务器时或者对应服务器不处于监听状态时，则会返回 `SOCKET_ERROR`

- 服务器如何可以退出循环？

`while(!_kbhit()) {}` 语句的结束只需要任意键盘按键操作即可，但仍然需要等待整个语句的执行结束。也就是说，在任意按键操作后，等待服务器处理完当前客户端的连接请求以及接收发信息后，就会退出循环。

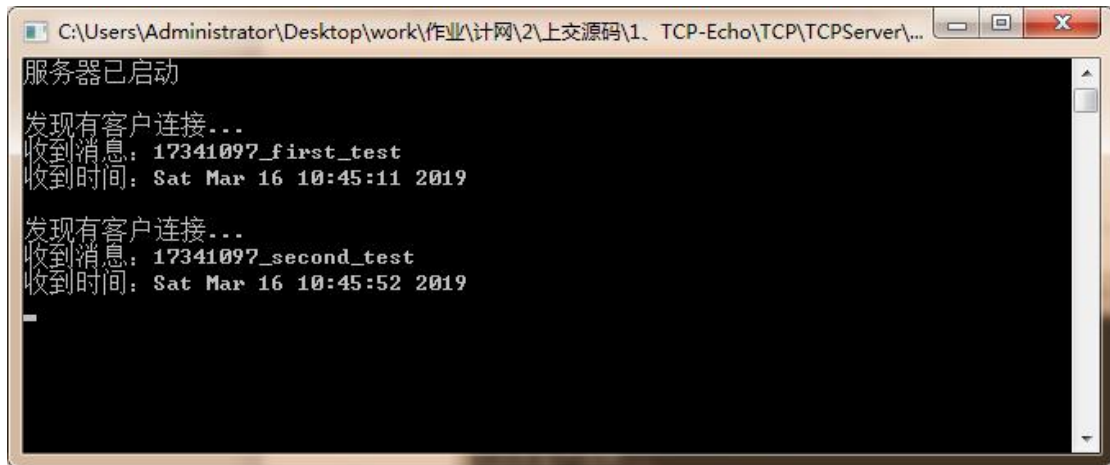
- 截屏（ctrl+alt+PrintScreen）服务器和客户端的运行结果（注明客户端和服务端）：



(客户端的第一次运行)



(客户端的第二次运行)



(服务器运行状态)

▪ 服务器的全部源代码 (或自选主要代码):

```
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
    printf("\n 发现有客户连接...\n");
    (void) time(&now); // 取得系统时间
    pts = ctime(&now); // 把时间转换为字符串
    int dd = recv(ssock, buf2, 300, 0); // 接受内容
    sprintf(buf, "%s\n", pts, buf2);
    printf("收到消息: %s\n 收到时间: %s", buf2, pts);
    dd = send(ssock, buf, strlen(buf), 0); // 送内容
    (void) closesocket(ssock); // 关闭连接套接字
}
```

//此处仅贴建立好连接后的收发送信息代码, 其余代码与老师提供的一致, 且此处为了简略, 没有用 if 语句对 recv 以及 send 语句进行错误返回判断, 原则上是要加的。

▪ 客户端的全部源代码 (或自选主要代码):

```
printf("连接服务器成功! \n 请输入要发送的信息: ");
scanf_s("%s", buf, BUFLen+1);
send(sock, buf, strlen(buf)+1, 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", buf); // 显示所接收的字符串
}
```

//此处仅贴建立好连接后的收发送信息代码，其余代码与老师提供的一致

(2) 编写 TCP Echo 增强程序

■ 实验要求：

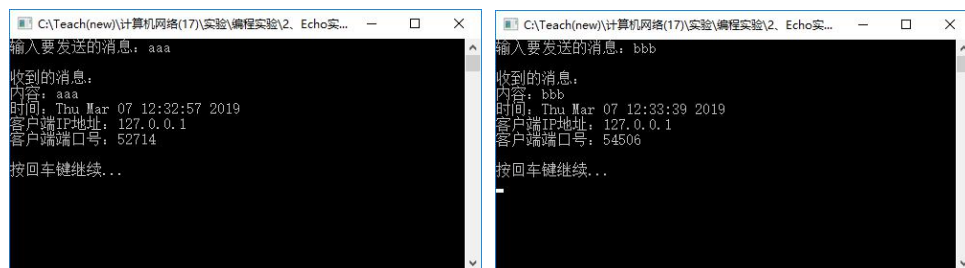
在(1)的基础上，**服务器**在收到客户端的消息时显示服务器的当前时间、客户端的 IP 地址、客户端的端口号和客户端发来的信息，并把它们一并返回给客户端。

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

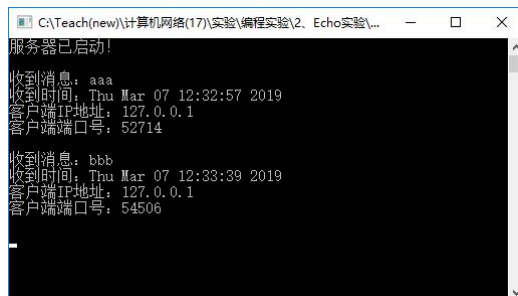
要求**服务器**直接从 accept() 的参数 fsin 中得到客户端的 IP 地址和端口号。注：服务器获取 IP 地址后要求直接使用 s_un_b 的四个分量得到 IP 地址，不能使用函数 inet_ntoa() 转换 IP 地址。

■ 参考运行截屏：

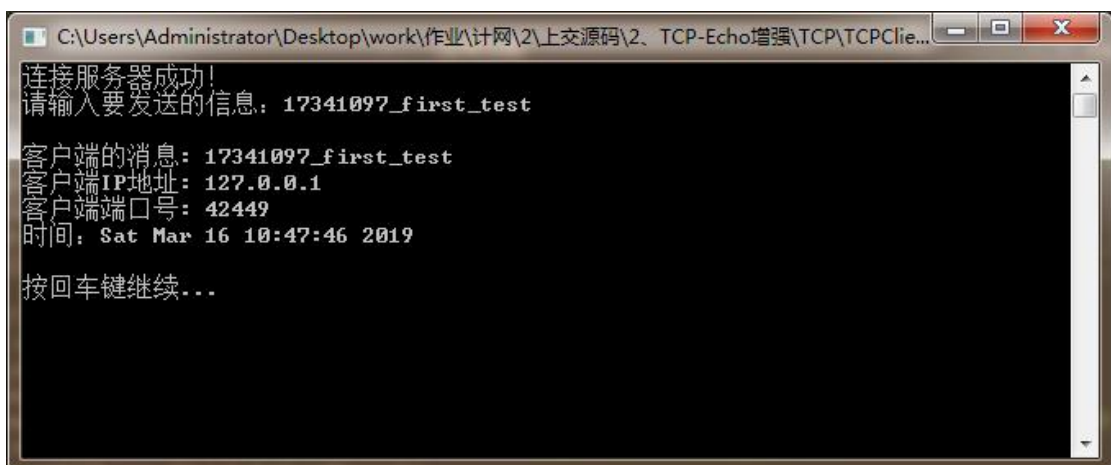
客户端（两次运行）



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的)：



（客户端的第一次运行）



```
C:\Users\Administrator\Desktop\work\作业\计网\2\上交源码\2、TCP-Echo增强\TCP\TCPClient...
连接服务器成功!
请输入要发送的信息: 17341097_second_test

客户端的消息: 17341097_second_test
客户端IP地址: 127.0.0.1
客户端端口号: 43729
时间: Sat Mar 16 10:48:27 2019

按回车键继续...
```

(客户端的第二次运行)

```
C:\Users\Administrator\Desktop\work\作业\计网\2\上交源码\2、TCP-Echo增强\TCP\TCPServer...
1
发现有客户连接...
客户端的消息: 17341097_first_test
客户端IP地址: 127.0.0.1
客户端端口号: 42449
时间: Sat Mar 16 10:47:46 2019

发现有客户连接...
客户端的消息: 17341097_second_test
客户端IP地址: 127.0.0.1
客户端端口号: 43729
时间: Sat Mar 16 10:48:27 2019
```

(客户端的运行状态)

▪ 服务器的全部源代码（或自选主要代码）:

```
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
    printf("发现有客户连接...\n");
    (void) time(&now);                                  // 取得系统时间
    pts = ctime(&now);                                  // 把时间转换为字符串
    int dd = recv(ssock, buf2, 300, 0);
    sprintf(addr_ip, "%d.%d.%d.%d", 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);
    sprintf(buf, "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\n",
buf2, addr_ip, fsin.sin_port, pts);
    printf("%s\n", buf);
```



```
send(ssock, buf, strlen(buf), 0);           //送内容
(void) closesocket(ssock);                 // 关闭连接套接字
}
```

//在文档中，换行与实际代码不一致，所以显得有点乱。这里我依旧没写错误检查，与第一个实验相比，仅仅多了 ip 地址（用 `s_un_b` 四个分量接在一起）和端口号（直接用 `sin_port` 获取）。

(3)编写 UDP Echo 增强程序

▪ 实验要求：

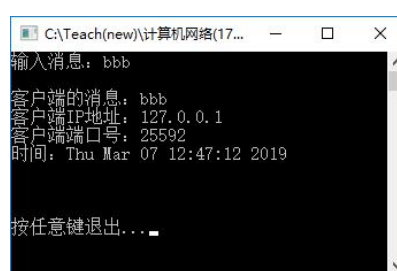
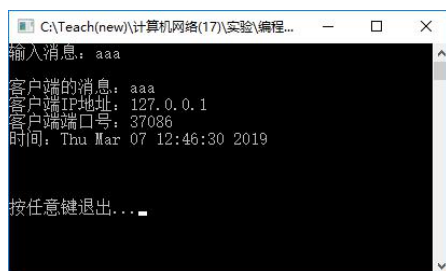
修改 UDP 例程，完成 Echo 功能，即当客户端发来消息时，服务器显示出服务器的当前时间、客户端的 IP、客户端的端口号和客户发来的信息，并把它们一并发回给客户端，客户端然后把它们显示出来。

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。

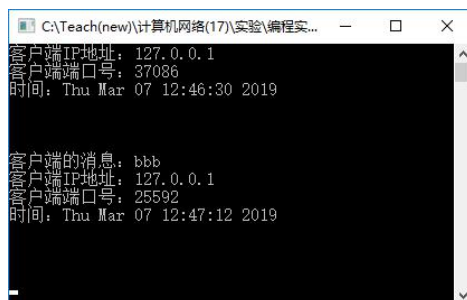
客户端程序的 `recvfrom()` 可以直接使用原来 `sendto` 使用的 `sock`。该 `sock` 已经绑定了客户端的 IP 地址和端口号，客户端可以直接用来接收数据。

▪ 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。



```
Microsoft Visual Studio 调试控制台
输入消息: ABCD
ABCD
按任意键退出...
C:\Users\Administrator\Desktop\work\作业\计网\2\上交源码\3、UDP-Echo\UDPClient\Debug\UDPClient.exe <进程 6680>已退出, 返回代码为: 0。
若要在调试停止时自动关闭控制台, 请启用“工具”->“选项”->“调试”->“调试停止时自动关闭控制台”。
按任意键关闭此窗口...
```

基本未发生错误, 但是信息会丢失, 原因是 UDP 下客户端不与服务器先 connect, 而是采用发送接收信息的方式互相交流。当端口号和 ip 地址无误时就能发送成功。但是该数据会丢失。

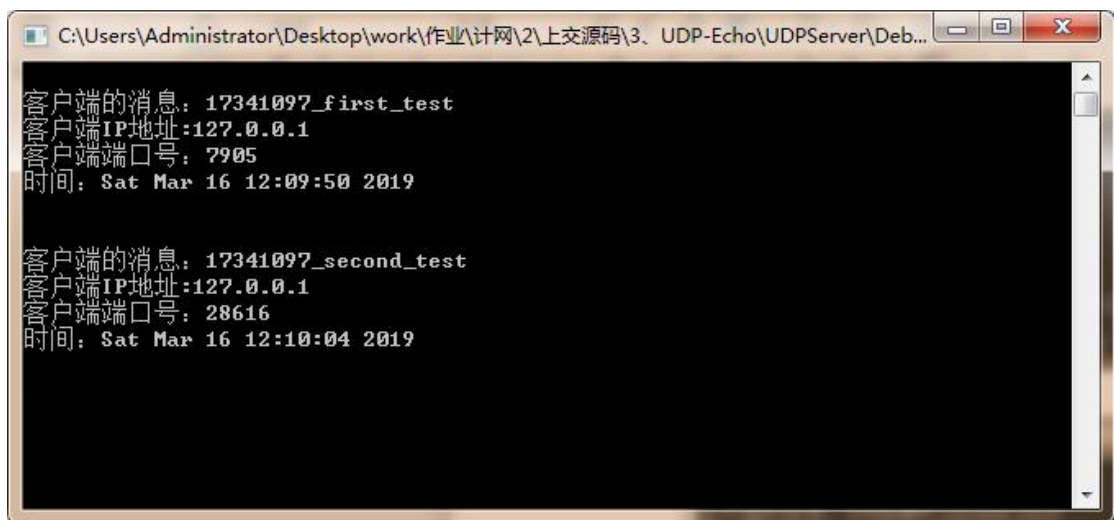
- 截屏服务器和客户端的运行结果 (注明客户端和服务):

```
Microsoft Visual Studio 调试控制台
输入消息: 17341097_first_test
客户端的消息: 17341097_first_test
客户端IP地址: 127.0.0.1
客户端端口号: 7905
时间: Sat Mar 16 12:09:50 2019
按任意键退出...
C:\Users\Administrator\Desktop\work\作业\计网\2\上交源码\3、UDP-Echo\UDPClient\Debug\UDPClient.exe <进程 14408>已退出, 返回代码为: 0。
若要在调试停止时自动关闭控制台, 请启用“工具”->“选项”->“调试”->“调试停止时自动关闭控制台”。
按任意键关闭此窗口...
```

(客户端第一次运行)

```
Microsoft Visual Studio 调试控制台
输入消息: 17341097_second_test
客户端的消息: 17341097_second_test
客户端IP地址: 127.0.0.1
客户端端口号: 28616
时间: Sat Mar 16 12:10:04 2019
按任意键退出...
C:\Users\Administrator\Desktop\work\作业\计网\2\上交源码\3、UDP-Echo\UDPClient\Debug\UDPClient.exe <进程 13620>已退出, 返回代码为: 0。
若要在调试停止时自动关闭控制台, 请启用“工具”->“选项”->“调试”->“调试停止时自动关闭控制台”。
按任意键关闭此窗口...
```

(客户端第二次运行)



```
C:\Users\Administrator\Desktop\work\作业\计网\2\上交源码\3、UDP-Echo\UDPServer\Deb...
客户端的消息: 17341097_first_test
客户端IP地址:127.0.0.1
客户端端口号: 7905
时间: Sat Mar 16 12:09:50 2019

客户端的消息: 17341097_second_test
客户端IP地址:127.0.0.1
客户端端口号: 28616
时间: Sat Mar 16 12:10:04 2019
```

(服务器端运行状态)

▪ 服务器的全部源代码 (或自选主要代码):

```
while(!_kbhit()){                                     //检测是否有按键
    (void)time(&now);                                  // 取得系统时间
    pts = ctime(&now);
    cc = recvfrom(sock, inf, 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;
    }
    sprintf(buf, "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",
    inf, inet_ntoa(from.sin_addr), from.sin_port, pts);
    cc = sendto(sock, buf, BUFLen, 0, (SOCKADDR *)&from, sizeof(from));
    if (cc == SOCKET_ERROR) {
        printf("发送失败, 错误号: %d\n", WSAGetLastError());
    }

    printf("\n%s\n", buf);
}
//此处仅贴绑定好 ip 等收发信息代码, 其余代码与老师提供的一致
```

▪ 客户端的全部源代码 (或自选主要代码):

```
printf("输入消息: ");
scanf_s("%s", buf, BUFLen+1);
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());
}

cc = recvfrom(sock, buf, BUFLen, 0, (SOCKADDR *)&from, &fromsize);
printf("\n%s\n", buf);
//此处仅贴绑定好 ip 等收发信息代码, 其余代码与老师提供的一致
```




【完成情况】

是否完成以下步骤? (√完成 ×未做)

(1) [√] (2) [√] (3) [√]

【实验体会】

1、主要问题与解决方案

由于老师已经提供相当充分的历程代码，本次实验非常简单，基本没遇到任何问题，只要在老师的基础上增加新的发送或接收函数以及一些字符串拼接工作就可以了。但是我还是遇到了一些小问题。

首先是，收发信息的阻塞，要求设计代码逻辑时，必须一方接受一方发送，所以代码顺序绝对不能乱，其次就是收与发的字符长度都要有同一的规定，不然很容易出现“烫烫烫”这种错误。

最后，udp 的端口不能为空，不然会出现 10049 错误，也就是 sendto 和 recvfrom 函数要好好写，修改老师的代码时要小心。

2、心得体会

明白了网络信息传递的主要方式，对套接字有了更进一步的理解。

【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://172.18.187.9/netdisk/default.aspx?vm=l7net>
编程实验

截止日期: 2019 年 3 月 16 日 23: 00 (周六)。

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)