实验二 UDP 通信实验

原创

时间与记忆 于 2017-07-11 15:11:31 发布 7898 收藏 31
分类专栏： Windows网络编程 文章标签： UDP 通信实验 Winsock API
版权声明：本文为博主原创文章，遵循 CC 4.0 BY-SA 版权协议，转载请附上原文出处链接和本声明。
本文链接：https://blog.csdn.net/DmxExcalibur/article/details/74974902
版权

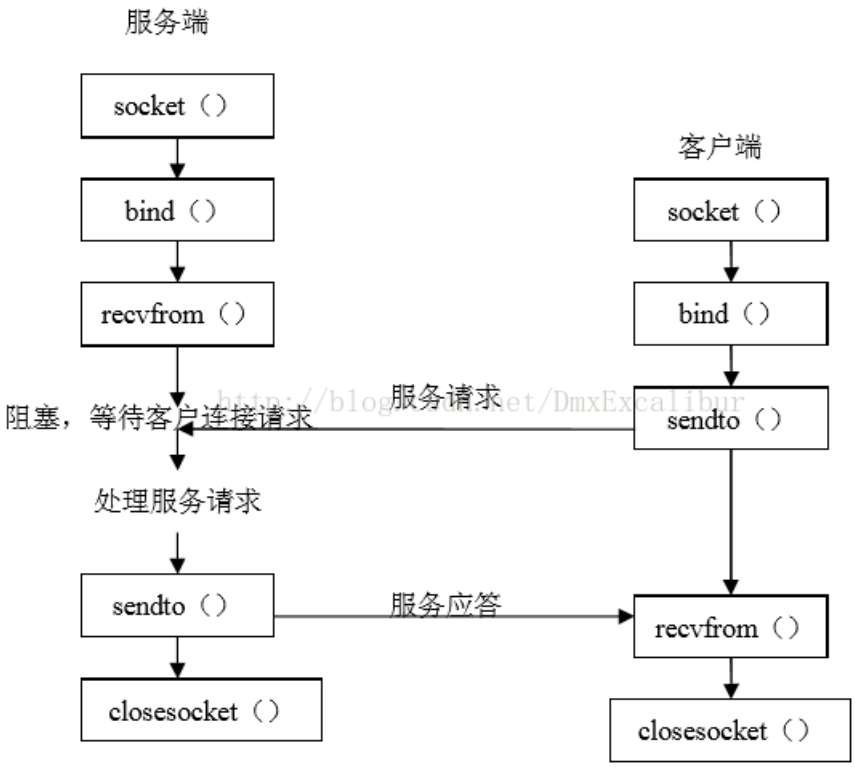
Windows网络编程 专栏收录该内容
8 篇文章 2 订阅
订阅专栏

一、实验目的

进一步理解 Winsock API 的调用方法。
了解 UDP 协议的工作原理。
掌握 UDP 服务端程序和客户端程序的编写流程。
熟悉程序的调试方法。

二、实验设计

1、背景知识
Winsock 编程模型 Winsock 编程的主要模型分为流套接字编程模型和数据报套接字编程模型两类，主要区别在于：前者提供双向的、有序的、无重复并且无记录边界的数据流服务，即采用有连接的 数据传输服务，保证数据可靠到达；后者也支持双向数据流，但不能保证数据的可靠、有序和无重复，它保留了记录边界，是一种无连接、不可靠的数据传输模型。
数据报套接字编程模型
数据报套接字使用 UDP 协议进行数据的传输，是一种无连接的数据传输模型，编程过程相对简单，采用客户/服务器（C/S）结构进行设计。
在数据报套接字编程模型中，客户端发送数据（也称发送端），服务器端接收数据（也称接收端）。实际上，由于数据报套接字编程模型也支持双向数据传递，因此，服务器端和客户端的概念已经比较模糊。为了说明数据报套接字编程模型的工作原理，这里仍然沿用这两个概念。
数据报套接字的服务进程和客户进程不需要在通信前建立连接，仅需要创建各自的套接 字，因此程序设计过程相当简单，简述如下：
接收端：(1)、创建数据报套接字；(2)、绑定本机地址和端口；(3)、等候接收数据；(4)、使用 完成后关闭套接字。
发送端：(1)、创建数据报套接字；(2)、向指定地址和端口发送数据；(3)、使用完成后关闭套接字。
数据报套接字编程使用的函数
(1) 创建套接字函数 socket() SOCKET socket（int af，int type，int protocol）；
由于采用数据报套接字进行数据传输，因此 type 参数必须设置为 SOCK_DGRAM， protocol 参数必须设置为 IPPROTO_UDP
(2) 绑定本地地址到所创建的套接字函数 bind()
int bind（SOCKET s，const struct sockaddr* name，int namelen）；
在实际编程时可以省略该函数，系统会自动绑定
(3) 接收数据函数 recvfrom()
int recvfrom（SOCKET s，char* buf，int len，int flags, struct sockaddr* from,int* fromlen）；
(4) 发送数据函数 sendto()
int sendto(SOCKET s，const char* buf，int len，int flags, const struct sockaddr* to,int* tolen）；
(5) 关闭套接字函数 closesocket()
int closesocket（SOCKET s）；
数据报套接字编程模型时序和流程
为便于理解数据报套接字模型下的编程过程，用时序图表述如下（请注意，时序图不同 于程序流程图，它只是对完成一次通信过程进行原理性描述的手段。



2、实验内容

(1)、理解数据报套接字编程模型，仔细阅读并调试运行 UDPserve.cpp 程序和 UTPClient.cpp 程序源代码，分析在服务端和客户端分别使用了哪些 Winsock API 函数；

(2)、修改 UDPServer 和 UDPClient 程序，设计一个简单的 UDP 通信程序，并达到以下要 求：

a.双方能相互发送数据，并显示接收到的数据。

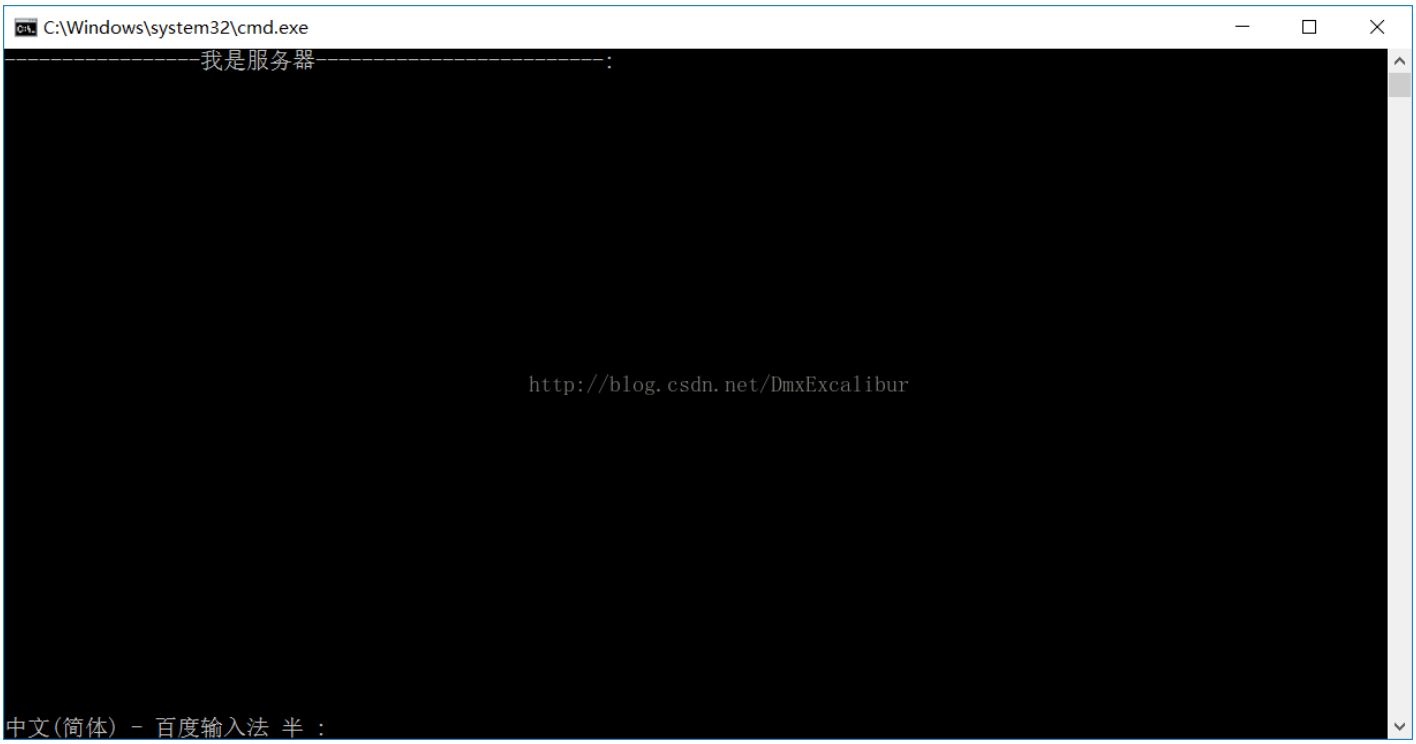
b.当收到对方的数据为“bye”时，能退出程序。

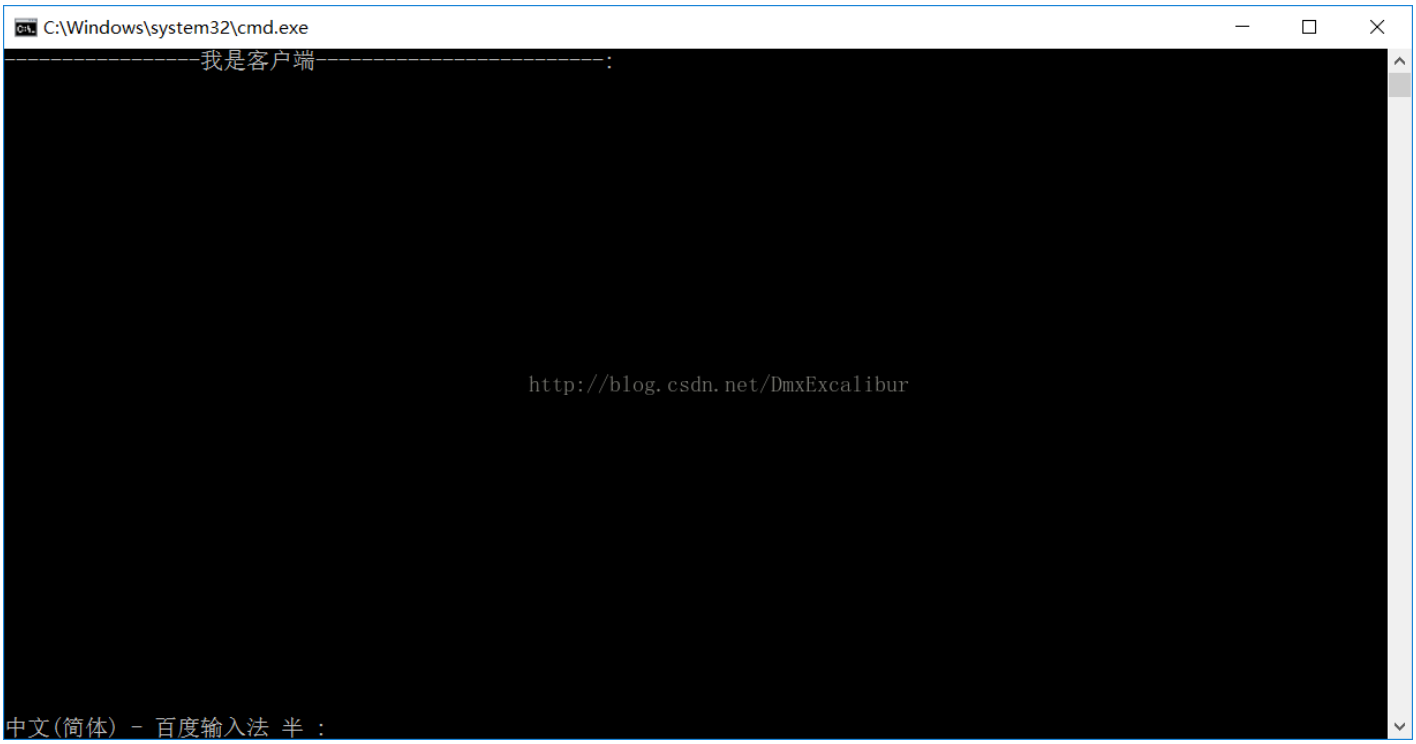
c.编程验证实验思考题中问题。

d.选做，服务器同多个客户端通信。

三、实验过程（包含实验结果）

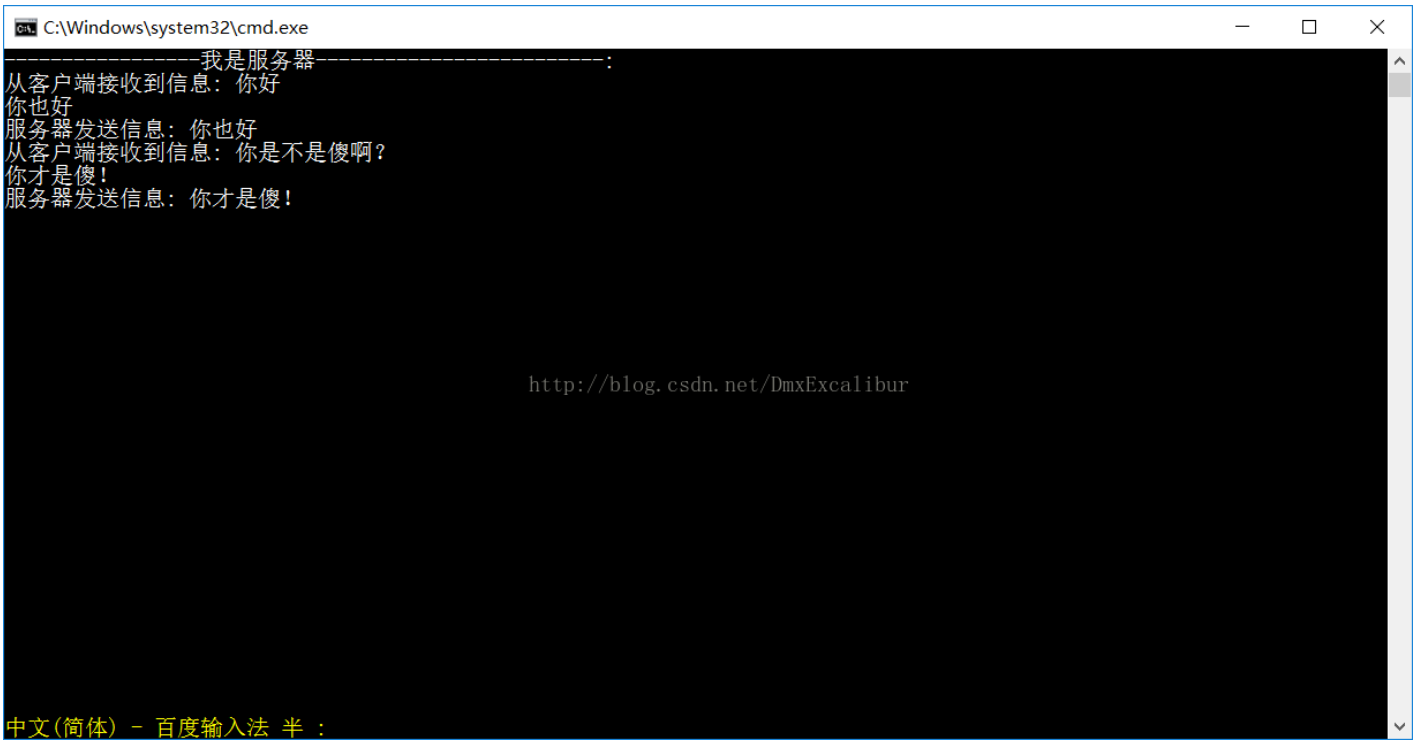
1、运行服务器和客户端程序

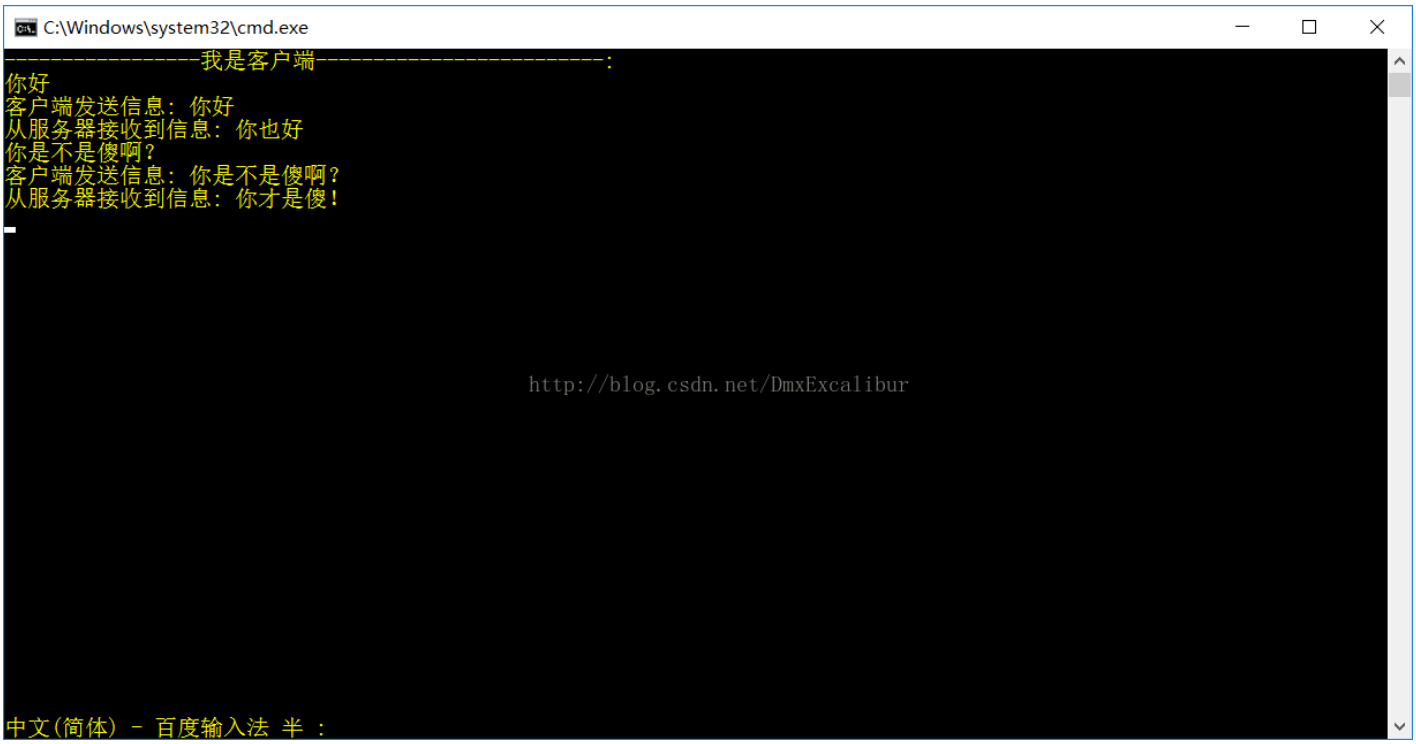




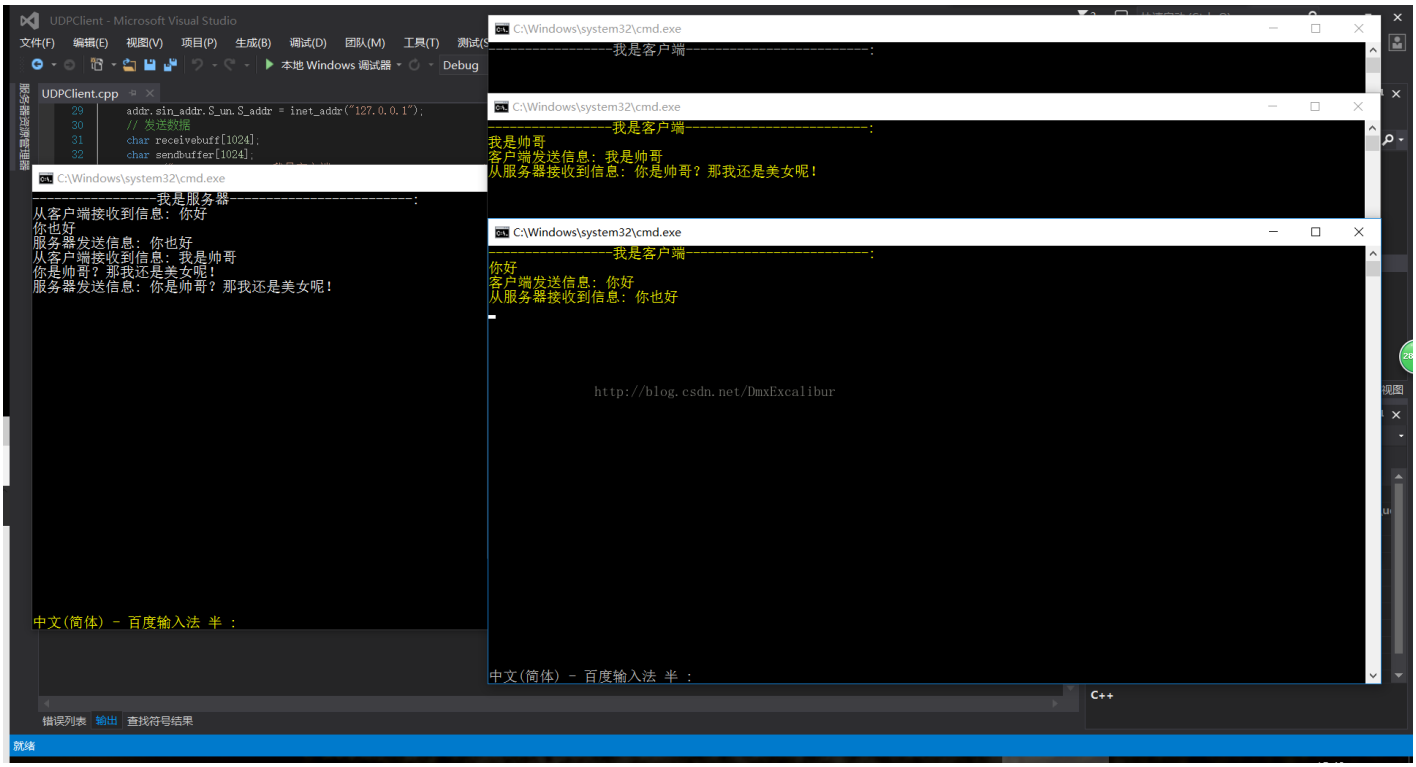
2、开始通信

服务器和客户端可以互发消息，收到消息的一方字体会变成亮黄色，用于提示，发送消息成功得而一方字体颜色恢复成白色。

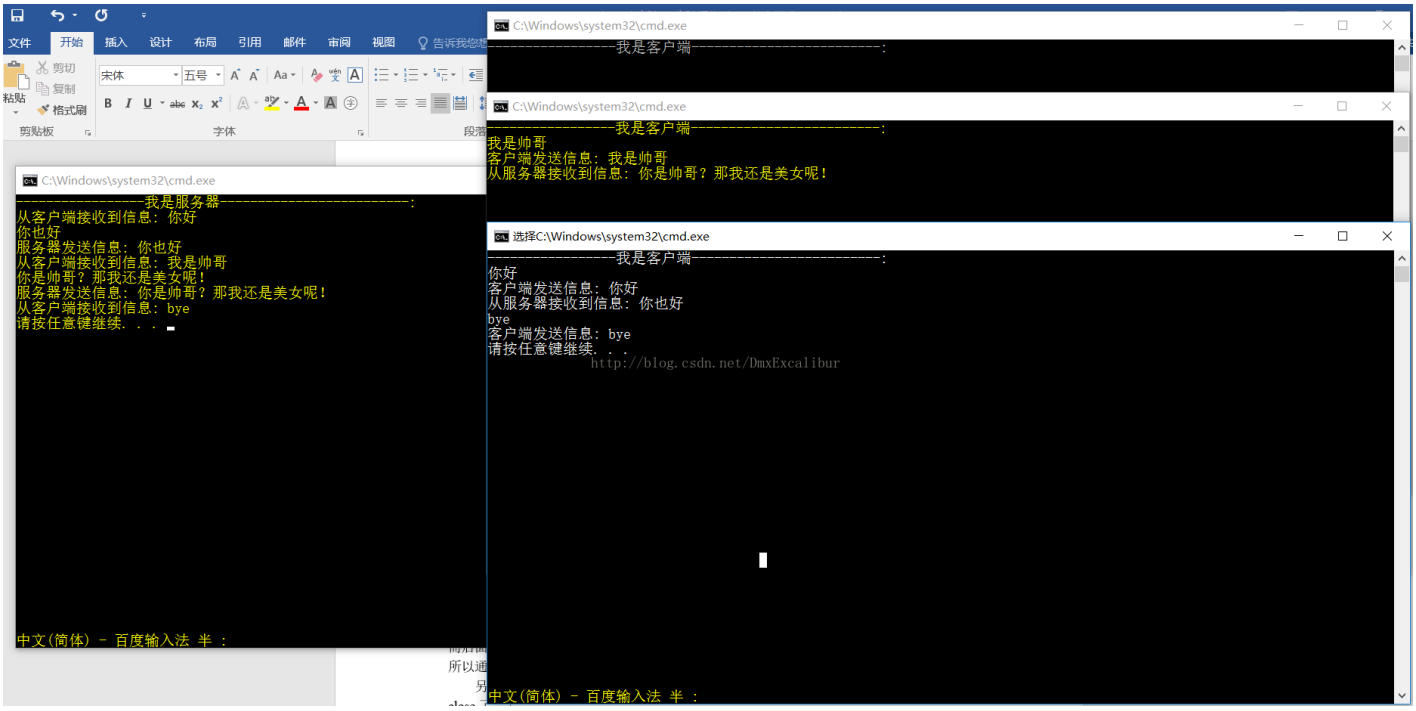

C:\Windows\system32\cmd.exe
----------我是服务器----------:
从客户端接收到信息：你好
你也好
服务器发送信息：你也好
从客户端接收到信息：你是不是傻啊？
你才是傻！
服务器发送信息：你才是傻！
http://blog.csdn.net/DmxExcalibur
中文(简体) - 百度输入法 半 ：


C:\Windows\system32\cmd.exe
----------我是客户端----------:
你好
客户端发送信息：你好
从服务器接收到信息：你也好
你是不是傻啊？
客户端发送信息：你是不是傻啊？
从服务器接收到信息：你才是傻！
http://blog.csdn.net/DmxExcalibur
中文(简体) - 百度输入法 半 ：

3、多个客户端通信



4、客户端输入“bye”，该客户端退出连接。



四、讨论与分析

1、能否在接收数据之间不进行bind（）调用？如果能，请说明可能的情况？

回答一：可以。创建套接字之后如果首先调用的是sendto()函数则可以不调用bind()函数显示的绑定本地地址，系统会自动的为程序绑定，因此今后即便是调用recvfrom()也不会失败。但是，如果创建套接字之后直接调用recvfrom()就会失败，因为套接字没有绑定。

回答二：书上都是这么说的，UDP客户端不用绑定IP和端口，操作系统会给它自动分配端口。但是虽然没有显示绑定，但是操作系统却似乎做了些隐蔽的事情。首先，在客户端，fd = socket(AF_INET, SOCK_DGRAM, 0)，然后就想在此fd下进行recvfrom是收不到对方（假设对方就是服务器吧）的消息是办不到的，其实想想也很容易明白，这是fd未和任何端口、IP产生关联要是这样都能收到消息，那真要乱套了。想要在没绑定的情况下受到服务器发来的消息，首先客户端得通过fd描述符首先向服务器发信息，然后这时在fd下进行阻塞recvfrom就能收到消息了，如果再在客户端上fd 1= socket(AF_INET, SOCK_DGRAM, 0)，想在fd1 上进行recvfrom依然收不到消息，因为fd和服务器同过信，但fd1没有，所以fd1收不到，但是从fd1向服务器发消息没问题！所一总结一下就是，只有当已通过fd向服务器发送了消息时（并且已经发通了），才能在fd处收到服务器发回来的消息，但是向服务器发送消息就不需要。所以说操作系统在此做了些隐蔽的事情，当fd首先向服务器发消息时客户端自动选折IP和一个PORT与该fd关联了起来，（我觉得相当于背后还是绑定了一样）。而后面创建的fd1和之前的fd他们

出客户端的PORT是不同的（我在服务器端检测了一下），所以通过fd向服务器发了消息但想在新建立的fd1下去recvfrom收不到消息。

另外，只能对一个socket描述符绑定一次，不能绑定多次，除非前面已经将该描述符close了。

反过来一个端口也只能被绑定到同一个socket描述符上，除非他们使用的不同的协议。

2、能否使用connect（）连接对方？为什么？

协议让UDP也可以调用connect。

(1).因为UDP可以是一对一，多对一，一对多，或者多对多的通信，所以每次调用sendto()/recvfrom()时都必须指定目标IP和端口号。通过调用connect()建立一个端到端的连接，就可以和TCP一样使用send()/recv()传递数据，而不需要每次都指定目标IP和端口号。但是它和TCP不同的是它没有三次握手的过程。

(2).还可以通过在已建立连接的UDP套接字上，再次调用connect()实现以下功能：

a.指定新的IP地址和端口号。

b.断开连接。

这也与TCP有所不同，TCP套接字只能调用一次connect()函数。

3、能否在不调用sendto()函数之前调用recvfom()函数？

答：可以。此时只需要在调用recvfrom()函数之前使用bind（）绑定地址。

五、总结

本次实验其实很简单，利用UDP协议进行通信，这样的通信并非TCP那样是可靠的传输，实验中用到最主要的值recvfrom()、sendto()两个函数，在一个循环中检测有没有收到信息，如果收到消息就将其打印出来，并将字体显示为亮黄色，字体变色是我的创新，考虑到提示的作用，所以用了亮黄色。

六、附录：关键代码

1.UDP服务器

```
#include "stdafx.h"
#include <stdio.h>
#include <winsock.h>
#include <string.h>
#include <iostream>
#pragma comment(lib,"wsock32.lib")
using namespace std;
int main()
{
 WSADATA wsaData;
 WSAStartup(MAKEWORD(2, 2), &wsaData);
 // 创建套节字
 SOCKET s = ::socket(AF_INET, SOCK_DGRAM, IPPROTO_UDP);
 if (s == INVALID_SOCKET)  {
  printf("Failed socket() \n");
  return 0;
 }
 // 填充sockaddr_in结构
 sockaddr_in sin;
 sin.sin_family = AF_INET;
 sin.sin_port = htons(4567);
 sin.sin_addr.S_un.S_addr = INADDR_ANY;
 // 绑定这个套节字到一个本地地址
 if (::bind(s, (LPSOCKADDR)&sin, sizeof(sin)) == SOCKET_ERROR)
 {
  printf("Failed bind() \n");
  return 0;
 }
 // 接收数据
 char receivebuff[1024];
```

```
 char sendbuff[1024];
 sockaddr_in addr;
 int nLen = sizeof(addr);
 printf("-----------------我是服务器-------------------------:\n");
 while (TRUE)
 {
  int nRecv = ::recvfrom(s, receivebuff, 1024, 0, (sockaddr*)&addr, &nLen);
  if (nRecv > 0)
  {    //接收数据
   system("color 0E");
   receivebuff[nRecv] = '\0';
   printf("从客户端接收到信息: %s\n", receivebuff);
   if (strcmp(receivebuff, "bye") == 0)
   {
    ::closesocket(s);
    return 0;
   }
   //发送数据
   // scanf("%s", sendbuff);
  }
  gets(sendbuff);
  ::sendto(s, sendbuff, strlen(sendbuff), 0, (sockaddr*)&addr, sizeof(addr));
  system("color 0F");
  printf("服务器发送信息: %s\n", sendbuff);
  if (strcmp(sendbuff, "bye") == 0)
  {
   ::closesocket(s);
   return 0;
  }

 }
 return 0;
}
```

2、UDP客户端

```
#include "stdafx.h"
#include <stdio.h>
#include <winsock.h>
#include <iostream>
#pragma comment(lib,"wsock32.lib")
using namespace std;
int main()
{
 WSADATA wsaData;
 WSAStartup(MAKEWORD(2, 2), &wsaData);
 // 创建套节字
 SOCKET s = ::socket(AF_INET, SOCK_DGRAM, IPPROTO_UDP);  if (s == INVALID_SOCKET)
 {
  printf("Failed socket() %d \n", ::WSAGetLastError());
  return 0;
 }
 // 也可以在这里调用bind函数绑定一个本地地址,否则系统将会自动安排
 // 填写远程地址信息  sockaddr_in addr;
 sockaddr_in addr;
 addr.sin_family = AF_INET;
```

```
 addr.sin_family = AF_INET;
 addr.sin_port = htons(4567);
 int nLen = sizeof(addr);
 // 注意，这里要填写服务器程序所在机器的IP地址，如果你的计算机没有联网，直接使用127.0.0.1即可
 addr.sin_addr.S_un.S_addr = inet_addr("127.0.0.1");
 // 发送数据
 char receivebuff[1024];
 char sendbuffer[1024];
 printf("-----------------我是客户端-------------------------:\n");
 while (TRUE)
 {
  int nRecv = ::recvfrom(s, receivebuff, 1024, 0, (sockaddr*)&addr, &nLen);
  if (nRecv > 0)
  {
   //接收数据
   system("color 0E");
   receivebuff[nRecv] = '\0';
   printf("从服务器接收到信息: %s\n", receivebuff);
   if (strcmp(receivebuff, "bye") == 0)
   {
    ::closesocket(s);
    return 0;
   }
  }
  //发送数据
  // scanf("%s", sendbuffer);
  gets(sendbuffer);
  ::sendto(s, sendbuffer, strlen(sendbuffer), 0, (sockaddr*)&addr, sizeof(addr));
  system("color 0F");
  printf("客户端发送信息: %s\n", sendbuffer);
  if (strcmp(sendbuffer, "bye") == 0)
  {
   ::closesocket(s);
   return 0;
  }
 }
 return 0;
}
```

注：本博客源代码下载地址：http://download.csdn.net/detail/dmxexcalibur/9904514