用Winsock实现语音全双工通信 PDF转换可能丢失图片或格式，建议阅读原文 https://www.100test.com/kao_ti2020/245/2021_2022__E7_94_A8Winsock_c97_245946.htm 一、引言 Windows 95作为微机的操作系统，已经完全融入了网络与通信功能，不仅可以建立纯Windows 95环境下的“对等网络”，而且支持多种协议，如TCP/IP、IPX/SPX、NETBUI等。在TCP/IP协议组中，TPC是一种面向连接的协义，为用户提供可靠的、全双工的字节流服务，具有确认、流控制、多路复用和同步等功能，适于数据传输。UDP协议则是无连接的，每个分组都携带完整的目的地址，各分组在系统中独立传送。它不能保证分组的先后顺序，不进行分组出错的恢复与重传，因此不保证传输的可靠性，但是，它提供高传输效率的数据报服务，适于实时的语音、图像传输、广播消息等网络传输。 Winsock接口为进程间通信提供了一种新的手段，它不但能用于同一机器中的进程之间通信，而且支持网络通信功能。随着Windows 95的推出。Winsock已经被正式集成到了Windows系统中，同时包括了16位和32位的编程接口。而Winsock的开发工具也可以在Borland C 4.0、Visual C 2.0这些C编译器中找到，主要由一个名为winsock.h的头文件和动态连接库winsock.dll或wsodk32.dll组成，这两种动态连接库分别用于Win16和Win32的应用程序。 本文针对话音的全双工传输要求，采用UDP协议实现了实时网络通信。使用VisualC 2.0编译环境，其动态连接库名为wsock32.dll。 二、主要函数的使用要点 通过建立双套接字，可以很方便地实现全双工网络通信。 1.套接字建立函数： SOCKET socket(int family,int type,int protocol)

对于UDP协议，写为： SOCKRET s.
s=socket(AF_INET,SOCK_DGRAM,0).
或s=socket(AF_INET,SOCK_DGRAM,IPPROTO_UDP) 为了建立两个套接字，必须实现地址的重复绑定，即，当一个套接字已经绑定到某本地地址后，为了让另一个套接字重复使用该地址，必须为调用bind()函数绑定第二个套接字之前，通过函数setsockopt()为该套接字设置SO_REUSEADDR套接字选项。通过函数getsockopt()可获得套接字选项设置状态。需要注意的是，两个套接字所对应的端口号不能相同。 此外，还涉及到套接字缓冲区的设置问题，按规定，每个区的设置范围是：不小于512个字节，大大于8k字节，根据需要，文中选用了4k字节。 2.套接字绑定函数 int bind(SOCKET s,struct sockaddr_in*name,int namelen) s是刚才创建好的套接字，name指向描述通讯对象的结构体的指针，namelen是该结构体的长度。该结构体中的分量包括：IP地址(对应name.sin_addr.s_addr)、端口号(name.sin_port)、地址类型(name.sin_family，一般都赋成AF_INET，表示是internet地址)。 (1)IP地址的填写方法：在全双工通信中，要把用户名对应的点分表示法地址转换成32位长整数格式的IP地址，使用inet_addr()函数。 (2)端口号是用于表示同一台计算机不同的进程(应用程序)，其分配方法有两种：1)进程可以让系统为套接字自动分配一端口号，只要在调用bind前将端口号指定为0即可。由系统自动分配的端口号位于1024~5000之间，而1~1023之间的任一TCP或UDP端口都是保留的，系统不允许任一进程使用保留端口，除非其有效用户ID是零(超级用户)。 (2)进程可为套接字指定一特定端口。这对于需要给套

接字分配一众所端口的服务器是很有用的。指定范围为1024和65536之间∩任意指定。 在本程序中，对两个套接字的端口号规定为2000和2001，前者对应发送套接字，后者对应接收套接字。 端口号要从一个16位无符号数(u_short类型数)从主机字节顺序转换成网络字节顺序，使用 htons()函数。 根据以上两个函数，可以给出双套接字建立与绑定的程序片断； //设置有关的全局变量 SOCKET sr,ss. HPSTR sockBufferS,sockBufferR. HANDLE hSendData,hReceiveData. DWROD dwDataSize=1024*4. struct sockaddr_in therel.there2. #DEFINE LOCAL_HOST_ADDR 200.200.200.201 #DEFINE REMOTE_HOST-ADDR 200.200.200.202 #DEFINE LOCAL_HOST_PORT 2000 #DEFINE LOCAL_HOST_PORT 2001 //套接字建立函数 BOOL make_skt(HWND hwnd) { struct sockaddr_in here,here1. ss=socket(AF_INET,SOCK_DGRAM,0). sr=socket(AF_INET,SOCK_DGRAM,0). if((ss==INVALID_SOCKET)||(sr==INVALID_SOCKET)) { MessageBox(hwnd,“ 套接字建立失败! ”，“ ”,MB_OK). return(FALSE). } here.sin_family=AF_INET. here.sin_addr.s_addr=inet_addr(LOCAL_HOST_ADDR). here.sin_port=htons(LICAL_HOST_PORT). //another socket herel.sin_family=AF_INET. herel.sin_addr.s_addr(LOCAL_HOST_ADDR). herel.sin_port=htons(LOCAL_HOST_PORT1). SocketBuffer().//套接字缓冲区的锁定设置 setsockopt(ss,SOL_SOCKET,SO_SNDBUF,(char FAR*)sockBufferS,dwDataSize).

if(bind(ss,(LPSOCKADDR)amp.here1,sizeof(here1))) { MessageBox(hwnd, “ 接收套接字绑定失败! ”， “ ” ， MB_OK). return(FALSE). } return(TRUE). } //套接字缓冲区设置 void sockBuffer(void) { hSendData=GlobalAlloc(GMEM_MOVEABLE|GMEM_SHARE,dwDataSize). if(!hSendData) { MessageBox(hwnd, “ 发送套接字缓冲区定位失败! ”， NULL, MB_OK|MB_ICONEXCLAMATION). return. } if((sockBufferS=GlobalLock(hSendData)==NULL) { MessageBox(hwnd, “ 发送套接字缓冲区锁定失败! ”， NULL, MB_OK|MB_ICONEXCLAMATION). GlobalFree(hRecordData[0]. return. } hReceiveData=globalAlloc(GMEM_MOVEABLE|GMEM_SHARE,dwDataSize). if(!hReceiveData) { MessageBox(hwnd," “ 接收套接字缓冲区定位败! ”， NULL MB_OK|MB_ICONEXCLAMATION). return. } if((sockBufferT=Globallock(hReceiveData))=NULL) MessageBox(hwnd,"发送套接字缓冲区锁定失败! ”， NULL, MB_OK|MB_ICONEXCLAMATION). GlobalFree(hRecordData[0]). return. } 100Test 下载频道开通，各类考试题目直接下载。详细请访问 www.100test.com