计 算 机 网 络

课 程 设 计 报 告

姓名：　　叶剑飞

学号：　　1005020201

班级：　　10 网络二班

指导老师：　　周新莲

湖南科技大学计算机科学与工程学院

2013 年 3 月

目录

1 网络聊天程序　　Win32 版

1.1 简介

早在 2011 年，我就和周开锋一起写了一个 TCP 协议 C/S 模式的网络聊天程序，该程序使用的是 WinSock 网络编程，因此只能运行于 Windows 系列操作系统。网络通信直接调用 WinSock API 来实现。至于图形用户界面部分，我们并没有使用各种 C++ 的图形库，而是直接调用了 Windows API 函数，使用纯 C 语言实现的图形界面。

1.2 版权说明

该网络聊天程序的服务器端的 control_clients.h 和 control_clients.c 版权所有：周开锋。客户端的 core.c 代码，版权所有：周开锋、叶剑飞。其余代码，版权所有：叶剑飞。

1.3 使用说明

该网络聊天程序支持 Windows XP 及以上版本的 Windows 操作系统。不支持 Windows 98/2000 或更低版本的操作系统，也不支持类 Unix 操作系统。该聊天软件仅支持 IPv4 地址，不支持 IPv6 地址。

首先，在一台计算机上打开该网络聊天程序的服务器端。经过一些初始化工作后，如果启动正常，应该看到如下所示的窗口：

图 1: 服务器端界面

然后，在另一台计算机上打开该网络聊天程序的客户端。如果打开正常，应该看到如下所示的窗口：

图 2: 客户端登录界面

在这个界面内键入服务器端的 IPv4 地址，即向服务端发起连接请求。连接请求成功后，显示下面这个对话框：

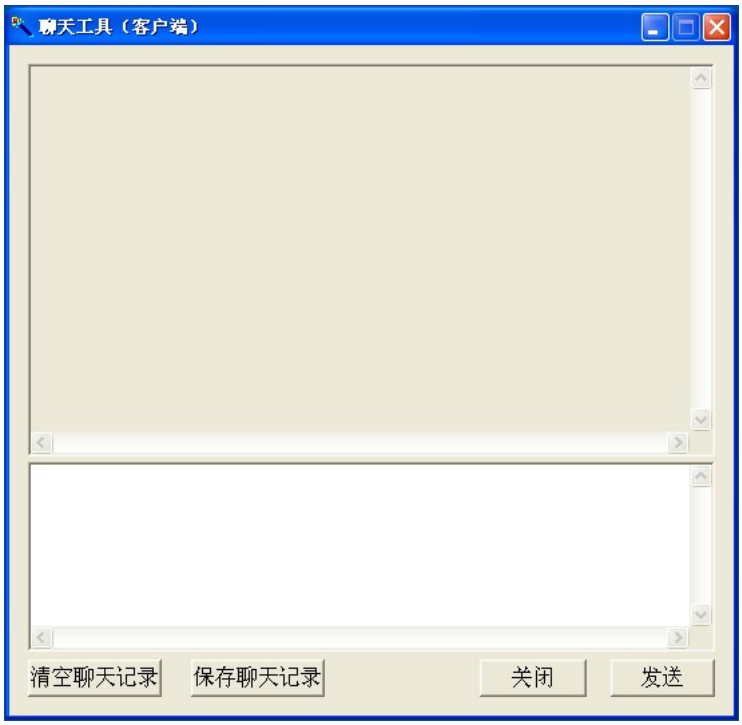


图 3: 客户端聊天界面

然后，就可以开始进行网络聊天了，其它客户端也可以继续输入服务器的 IP 地址，来加入聊天。

下面，给出几个聊天截图吧：

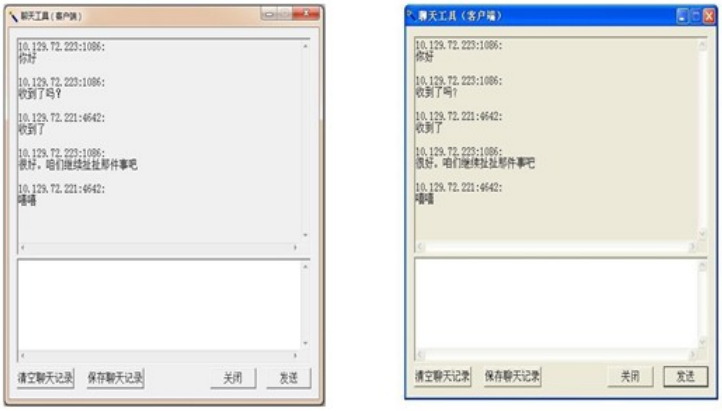


图 4: 网络聊天界面

1.4 技术细节

1.4.1 工作原理

本聊天程序采用网络通信部分用的是 WinSock 编程，图形用户界面用的是 Windows API 函数直接实现。完全是纯的 C 语言的框架。

服务器端的工作原理是：先调用 WSAStartup 函数，载入 WinSock 套接字库。使用的版本号是 WinSock 2.2 ，所以函数是 WSAStartup (MAKEWORD (2,2), &wsaData); 。然后调用 socket 函数，创建套接字。由于用的是 TCP 协议，所以代码是 socket(AF_INET, SOCK_STREAM, IPPROTO_TCP);，接下来服务端调用 bind 函数绑定套接字到服务器端的端口，该网络聊天软件用的是 2007 号端口。绑定完毕后，调用 listen 函数，开始监听端口，用 accept 函数来等待客户端的 connect 函数的连接，此时开启一个线程来接收客户端发来的 TCP 报文。并且在此时，显示主窗口。接受客户端的连接请求和接收报文在两个线程内互不干扰，这样能不断接收报文，并不断接受新的客户端。新的客户端被存入链表。而服务器端发送是遍历一整个链表来发送。当服务器端的 recv 的返回值为 0 的时候，说明客户端发送了 shutdown 信号，因此此时可以从链表中清除这个已断开连接的无效套接字。服务器端单击关闭按钮的时候，也会遍历一次链表，向每个套接字发送一个 shutdown 信号，并关闭套接字。最后调用 WSACleanup() 函数移出 WinSock 套接字库，并关闭窗口，结束进程。

客户端的工作原理是：先显示 IP 输入框，让用户输入服务器端的 IPv4 地址。与此同时，另外开个线程来做一些初始化工作。例如调用 WSAStartup 函数载入 WinSock 套接字库，并调用 socket 函数来创建客户端套接字。输入完 IPv4 地址并确定后，即可禁用 IPv4 输入框，并调用 connect 函数连接服务器。连接成功后，显示网络聊天客户端的主窗口。然后就可以互相 send 和 recv，如果 recv 的返回值为 0，说明服务器端调用了 shutdown 函数断开了连接，那么此时客户端就应该关闭套接字，并移出 WinSock 套接字库，并用 MessageBox 显示“服务器端已断开连接”的提示。然后即可结束进程。

网络聊天的过程的示意图如下所示：

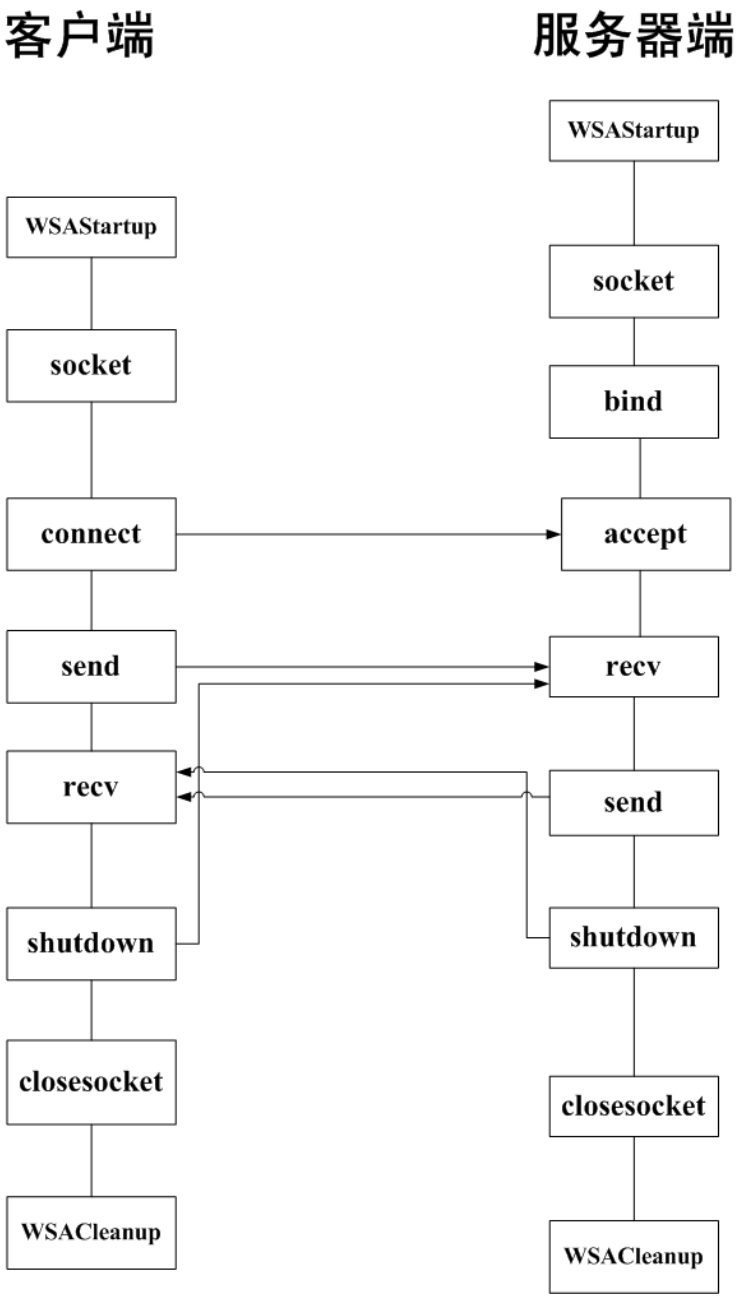


图 5: WinSock C/S 模式 TCP 协议的网络聊天程序原理示意图

1.4.2 开发过程

当时，我负责写客户端程序，而周开锋负责写服务器端程序。起初，我们是先写了单线程控制台程序。经过调试，修复了 bug 之后，改成多线程控制台程序。多线程的调试难度较大，因此我们插入了一些等待线程的一些代码，作为调试代码。而对于 sockaddr_in 之类的数据，由于内部是二进制数据，不易

阅读。为了调试这些东西，我们插入 inet_ntoa 等函数作为调试代码，将其转化成“人类可阅读”(human readable) 的形式，输出到控制台上，以便调试。后来，客户端加入了图形用户界面，而服务器端用的仍然是命令行界面。最后，我把服务器端改成了图形用户界面，所以就成为完全图形界面的了。为了把服务器端改成图形界面，我重写了部分上层代码，而周开锋写的下层代码几乎是原封不动，所以如果认真的阅读服务器端的代码还有很多命令行界面的“历史遗留问题”的痕迹，但这种痕迹已经不多了。当时，我们还打算做用户名密码登录，后来放弃了。但是 control_client.h 和 control_client.c 中还保留着这些“未来发展接口”的痕迹。服务器端经过命令行界面改图形用户界面之后，只剩下 control_clients.c 文件和 control_clients.h 还是原封不动地使用周开锋的代码，其余都是我自己写的代码。全部工程完成于 2012 年春。

1.5 程序源代码

1.5.1 服务器端和客户端共用的 C 语言源代码

```
/***
 *CommonDialog.h - declarations for Common Dialog
 *
 *        Copyright (C) 2011, Victor Jianfei Ye. All rights reserved.
 *
 *Purpose:
 *        This file declare three important functions of the common dialog.
 *
 ****/

#ifndef _MODULE_DIALOG_H
#define _MODULE_DIALOG_H

#ifndef _WINDOWS_
#include <windows.h>
#endif


#ifdef __cplusplus
extern "C" {
#endif

BOOL SaveAsDlg ( HWND hWnd , LPCTSTR pstrFilter , PTSTR pstrFilePath ,
                 PTSTR pstrFileName , LPCTSTR lpstrDefExt ) ;
BOOL OpenDlg ( HWND hWnd , LPCTSTR pstrFilter , PTSTR pstrFilePath ,
                           PTSTR pstrFileName ) ;
BOOL BrowseForFolder ( HWND hWnd , LPCTSTR pszTitle , LPTSTR pszPathName ) ;

#ifdef __cplusplus
}
#endif


#endif
```

```
/***
 *CommonDialog.c - common dialog functions
 *
 *        Copyright (C) 2011, Victor Jianfei Ye. All rights reserved.
 *
 *Purpose:
 *        defines common dialog functions
 *
 ****************************************************************/

#include <windows.h>
#include <shlobj.h>

#ifdef _DEBUG
#pragma comment(linker, "/NODEFAULTLIB:\"LIBC\"")
#else
#pragma comment(linker, "/OPT:NOREF")
#endif

/***
 *BOOL SaveAsDlg( HWND hWnd, LPCTSTR pstrFilter,
 *                PTSTR pstrFilePath, PTSTR pstrFileName,
 *                LPCTSTR lpstrDefExt) - show a "Save As" dialog
 *
 *Purpose:
 *        Show a "Save As" dialog
 *
 *Entry:
 *        HWND      hWnd        - the window that owns the dialog box
 *        LPCTSTR   pstrFilter  - The first string in each pair
 *        is a display string that describes
 *        the filter (for example, "Text Files"),
 *        and the second string specifies the
 *        filter pattern (for example, "*.TXT").
 *        e.g. TEXT("Text Files (*.txt)\0*.txt\0") \
 *                TEXT("All Files (*.*)\0*.*\0") \
 *                TEXT("\0");
 *        PTSTR       pstrFilePath  - returns the full file path
 *        PTSTR       pstrFileName  - returns the file name
 *        LPCTSTR     lpstrDefExt   - the default extension name
 *
 *Exit:
 *        Success:  TRUE
 *        Failure:  FALSE
 *
 *Exceptions:
 *
```

```
 ***************************************************************/

BOOL SaveAsDlg ( HWND hWnd , LPCTSTR pstrFilter ,
          PTSTR pstrFilePath , PTSTR pstrFileName , LPCTSTR lpstrDefExt )
{
    OPENFILENAME ofn ;
    pstrFilePath[0] = (TCHAR)'\0' ;
    pstrFileName[0] = (TCHAR)'\0' ;
    ofn.lStructSize = sizeof ( OPENFILENAME ) ;
    ofn.hwndOwner = hWnd ;
    ofn.hInstance = GetModuleHandle( NULL ) ;
    ofn.lpstrFilter = pstrFilter ;          // file filter
    ofn.lpstrCustomFilter = NULL ;
    ofn.nMaxCustFilter = 0L ;
    ofn.nFilterIndex = 0L ;
    ofn.lpstrFile = pstrFilePath ;
    ofn.nMaxFile = MAX_PATH ;
    ofn.lpstrFileTitle = pstrFileName ;
    ofn.nMaxFileTitle = MAX_PATH ;
    ofn.lpstrInitialDir = NULL ;
    ofn.lpstrTitle = NULL ;          // the title bar of the dialog box
    ofn.Flags = OFN_OVERWRITEPROMPT ;
    ofn.nFileOffset = 0 ;
    ofn.nFileExtension = 0 ;
    ofn.lpstrDefExt = lpstrDefExt ;     // the default extension
    ofn.lCustData = 0L ;
    ofn.lpfnHook = NULL ;
    ofn.lpTemplateName = NULL ;
    return GetSaveFileName ( &ofn ) ;
}

/***
 *BOOL OpenDlg ( HWND hWnd , LPCTSTR pstrFilter ,
 *              PTSTR pstrFilePath , PTSTR pstrFileName ) - show an "Open" dial
 *
 *Purpose:
 *        Show an "Open" dialog
 *
 *Entry:
 *        HWND        hWnd            - the window that owns the dialog box
 *        LPCTSTR         pstrFilter          - The first string in each pair
 *        is a display string that describes
 *        the filter (for example, "Text Files"),
 *        and the second string specifies the
 *        filter pattern (for example, "*.TXT").
 *        e.g. TEXT ("Text Files (*.txt)\0*.txt\0") \
 *                TEXT ("All Files (*.*)\0*.*\0" ) \
 *                TEXT ("\0");
 *        PTSTR         pstrFilePath    - returns the full file path
 *        PTSTR         pstrFileName    - returns the file name
```

```
 *
 *Exit:
 *         Success:  TRUE
 *         Failure:  FALSE
 *
 *Exceptions:
 *
 ******************************************************************/

BOOL OpenDlg ( HWND hWnd , LPCTSTR pstrFilter ,
                          PTSTR pstrFilePath , PTSTR pstrFileName )
{
    OPENFILENAME ofn ;
    pstrFilePath[0] = (TCHAR)'\0' ;
    pstrFileName[0] = (TCHAR)'\0' ;
    ofn.lStructSize = sizeof (OPENFILENAME) ;
    ofn.hwndOwner = hWnd ;
    ofn.hInstance = GetModuleHandle( NULL ) ;
    ofn.lpstrFilter = pstrFilter ;         // file filter
    ofn.lpstrCustomFilter = NULL ;
    ofn.nMaxCustFilter = 0L ;
    ofn.nFilterIndex = 0L ;
    ofn.lpstrFile = pstrFilePath ;
    ofn.nMaxFile = MAX_PATH ;
    ofn.lpstrFileTitle = pstrFileName ;
    ofn.nMaxFileTitle = MAX_PATH ;
    ofn.lpstrInitialDir = NULL ;
    ofn.lpstrTitle = NULL ;          // the title bar of the dialog box
    ofn.Flags = OFN_HIDEREADONLY | OFN_CREATEPROMPT ;
    ofn.nFileOffset = 0 ;
    ofn.nFileExtension = 0 ;
    ofn.lpstrDefExt = NULL ;     // the default extension
    ofn.lCustData = 0L ;
    ofn.lpfnHook = NULL ;
    ofn.lpTemplateName = NULL ;
    return GetOpenFileName ( &ofn ) ;
}

/***
 *BOOL BrowseForFolder ( HWND hWnd , LPCTSTR pszTitle , LPTSTR pszPathName )
 *
 *Purpose:
 *         Show an "browse for folder" dialog
 *
 *Entry:
 *         HWND      hWnd          - the window that owns the dialog box
 *         LPCTSTR   pszTitle      - The prompt in the browse-for-folder dialog b
 *         LPTSTR    pszPathName   - returns the full folder path
 *
 *Exit:
```

```
 *          Success:   TRUE
 *          Failure:   FALSE
 *
 *Exceptions:
 *
 ************************************************************************

BOOL BrowseForFolder ( HWND hWnd , LPCTSTR pszTitle , LPTSTR pszPathName )
{
    BROWSEINFO browseInfo ;
    LPITEMIDLIST lpItemIDList;
    pszPathName[0] = '\0' ;
    browseInfo.hwndOwner = hWnd ;
    browseInfo.pidlRoot = 0 ;
    browseInfo.pszDisplayName = pszPathName ;
    browseInfo.lpszTitle = pszTitle ;
    browseInfo.ulFlags = BIF_RETURNONLYFSDIRS ;
    browseInfo.lpfn = NULL ;
    browseInfo.iImage = 0 ;
    if ( ( lpItemIDList = SHBrowseForFolder ( &browseInfo ) ) != NULL )
    {
        if ( SHGetPathFromIDList ( lpItemIDList , pszPathName ) )
            return TRUE ;
        else
            return FALSE ;
    }
    return FALSE ;
}
```

1.5.2 服务器端程序 C 语言源代码

```
/***
 *common.h - declarations for external global varable
 *
 *          Copyright (C) 2011, Victor Jianfei Ye. All rights reserved.
 *
 *Purpose:
 *          This file declare the external global varable
 *                  szCaption in order to use in the MessageBox function.
 *
 ****/

#ifndef _COMMON_H
#define _COMMON_H

extern TCHAR szCaption[100] ;
```

```
#endif



/***
 *client_structs.h – definition of the structure of the client
 *
 *        Copyright (C) 2011, Victor Jianfei Ye. All rights reserved.
 *
 *Purpose:
 *        This file defines the structure of the client.
 *
 ****/

#ifndef _CLIENT_STRUCTS_H
#define _CLIENT_STRUCTS_H

typedef volatile struct client
{
   SOCKET sck;
   SOCKADDR_IN addr;
    volatile BOOL state;
   SYSTEMTIME starttime;
   SYSTEMTIME latesttime;
    struct client volatile *next;
} client,*pclient;

typedef struct list
{
    int total_online;
    int total;
    pclient head;
    pclient tail;
}list , *plist;


#endif



/***
 *control_clients.h – declarations for control_clients
 *
 *        Copyright (C) 2011, Zhou Kaifeng. All rights reserved.
 *
 *Purpose:
 *        This file declares some function about the clients' information.
 *
```

```
 ****/

#ifndef CONTROL_CLIENTS_H
#define CONTROL_CLIENTS_H

#include <winsock2.h>
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include "resource.h"
#pragma comment(lib,"ws2_32.lib")

#define PEMTERLEN 255 //设定命令的长度
#define BUFLEN 65535  //缓冲区的长度
#define TIME  1000       //更新的时间间隔
#define SERVERLEN  2000 //服务器方的最大发送长度

#include "client_structs.h"

int init_clients(void);          //初始化用户链表
BOOL ins_clients(client vertex) ;    //插入一个用户到链表中
int del_clients(void);                  //删除已断线的用户
BOOL dst_clients(void);                 //注销用户链表
void getsocket(client clt,SOCKET *sck);       //得到用户的socket;
void getstate(client clt,char *state);        //得到用户的在线
状态
void getaddr(client clt,char *addr_str,
                  unsigned short *port);    //得到用户的和端
口ip
pclient next(pclient bef);         //访问下一个用户
int get_online_count(void);        //得到在线用户的数量
pclient getlist_head(void);        //得到用户列表的第一个用户,
没有用户时返回为空
BOOL headmatch(const char *n,const char *m) ;      //字符串首
部匹配
void getsubstr(char *s,char *sub,int begin,int end);   //得
到指定的字串
void show(pclient clt);                 //显示用户信息
void sendall(const char *bufer) ;    //发送给所有用户信息
void operate_msg(pclient clt,char *msg);        //客户端消息处
理
//建立并初始化一个用户
void set_client(client *clt,SOCKET sck,SOCKADDR_IN addr ,
                BOOL state , SYSTEMTIME time , pclient next) ;
#endif



/***
 *control_clients.c - control_clients functions
```

```
 *
 *        Copyright (C) 2011, Zhou Kaifeng. All rights reserved.
 *
 *Purpose:
 *        defines control_clients functions
 *
 ************************************************************/

#include"control_clients.h"
#pragma warning( disable : 4090 4022 )

list clients;
const plist client_list = &clients;

BOOL ShowMessage ( LPCTSTR szMessage ) ;

int init_clients(void) // initialize users' linklist
{
    client_list->total=0;
    client_list->head=NULL;
    client_list->tail=NULL;
    return TRUE;
}

BOOL ins_clients(client vertex) // insert a user into the linklist
{
    pclient pot=(pclient)malloc(sizeof(client));
    if(pot)
    {
        (*pot) = vertex;
        pot->next=NULL;
        if(client_list->head==NULL)
            client_list->head=pot;
        if(client_list->tail!=NULL)
            client_list->tail->next=pot;
        client_list->tail=pot;
        (client_list->total_online)++;
        (client_list->total)++;
        return TRUE;
    }
    else
        return FALSE;
}

int del_clients(void) // delete offline users
{
    pclient pot=client_list->head,p,top,head;
    head=pot;
    top=pot;
    while(pot)
```

```
    {
        p=pot->next;
        if(!(pot->state)) // judge whether the user is online or not
        {
            if(pot==top)
                top=p;
            if(pot==head)
                head=p;
            else top->next=p;
            shutdown(pot->sck,SD_BOTH);
            closesocket(pot->sck);
            free(pot);
            (client_list->total_online)--;
        }
        else
            top=pot;
        pot=p;
    }
    client_list->tail=top;
    client_list->head=head;
    if(client_list->tail!=NULL)client_list->tail->next=NULL;
    return     TRUE;
}

BOOL dst_clients(void) // log off users' linklist
{
    pclient pot=client_list->head,p=NULL;
    if(client_list->head != NULL)
    {
        pot=pot->next;
        shutdown(client_list->head->sck,SD_BOTH);
        closesocket(client_list->head->sck);
        free(client_list->head);
    }
    if(pot != NULL)
    {
        p=pot->next;
        while( p != NULL )
        {
            shutdown(pot->sck,SD_BOTH);
            closesocket(pot->sck);
            free(pot);
            pot=p;
            p=pot->next;
        }
        shutdown(pot->sck,SD_BOTH);
        closesocket(pot->sck);
        free(pot);
    }
    client_list->total=0;
```

```
    client_list->total_online=0;
    return TRUE;
}
void sendall(const char *bufer)
{
    pclient pclt;
    SOCKET sck;
    pclt=getlist_head();
    while(pclt)
    {
        getsocket((*pclt),&sck);
        if ( send ( sck , bufer , strlen(bufer)+1 , 0 ) == SOCKET_ERROR )
        {
            // sending error
        }
        pclt = next(pclt);
    }
}


pclient next(pclient bef) // visit the next user
{
    if(bef)
        return bef->next;
    else return NULL;
}

// create and initialize a user

void set_client(client *clt,SOCKET sck,SOCKADDR_IN addr ,
                    BOOL state , SYSTEMTIME time , pclient next)
{
    clt->sck=sck;
    clt->addr=addr;
    clt->state=state;
    clt->starttime=time;
    clt->latesttime=time;
    clt->next=next;
}

void getaddr(client clt,char *addr_str,unsigned short *port)// get user's IP an
{
    strcpy(addr_str,inet_ntoa(clt.addr.sin_addr));
    *port=ntohs(clt.addr.sin_port);
}

pclient getlist_head(void) // get the first user in the user list.
                           // If there are no users, return NULL
{
    return (client_list->head);
```

```
}

int get_online_count(void) // get the population of the online users
{
    return client_list->total_online;
}


void getsocket(client clt,SOCKET *sck) // get user's Socket
{
    (*sck)=clt.sck;
}

void getstate(client clt,char *state) // get user's online state
{
    *state=clt.state;
}

void getOnlineUserList (char* szOnlineUserList)
{
    char addr_str[20] ="\0" ;
    char port_str[5] = "\0" ;
    unsigned short port = (unsigned short)0 ;
    pclient p=client_list->head ;
    strcpy ( szOnlineUserList , "\0" ) ;
    while(p)
    {
        if(p->state)
        {
            getaddr ( *p , addr_str , &port ) ;
            sprintf ( port_str , "%hu" ,port ) ;
            strcat(szOnlineUserList,addr_str);
            strcat(szOnlineUserList,":");
            strcat(szOnlineUserList,port_str);
            strcat(szOnlineUserList,"\r\n") ;
        }
        p=p->next;
    }
}

BOOL headmatch(const char *n,const char *m) // match the first few letters of t
{
    int i,len1=strlen(m),len2=strlen(n);
    if(len2<len1)
        return FALSE;
    else
    {
        for(i=0;i<len1;i++)
        {
            if(n[i] != m[i])
```

```
                return FALSE;
            }
        }
        return TRUE;
}
void getsubstr(char *s,char *sub,int begin,int end) // get the designated strin
{
        int i;
        if(begin<=end)
        {
            for(i=begin;i<=end;i++)
                sub[i-begin]=s[i];
            sub[i-begin]='\0';
        }
}


void operate_msg(pclient clt,char *msg) // operate the message of the client
{
        char szBufSend[BUFLEN+100]="\0",ip[16]="\0";
        char list[5000]="",adr[16];
        unsigned short port;
        getaddr ( (*clt) , adr , &port ) ;
        sprintf(szBufSend,"%s:%d:\r\n%s",adr,port,msg);
        sendall(szBufSend);
        ShowMessage ( szBufSend ) ;
}



/***
 *online_users.h - declarations the structure
 *
 *        Copyright (C) 2011, Victor Jianfei Ye. All rights reserved.
 *
 *Purpose:
 *        This file declares the structure containing the handle to the
 *        two controls of the place to show people online.
 *
 ****/

#ifndef _ONLINE_USER_H
#define _ONLINE_USER_H

typedef struct tagHOnlineUserInfo
{
    HWND hLableOnlinePeople ;
    HWND hOnlineUserList ;
} HOnlineUserInfo , *PHOnlineUserInfo ;
```

```
#endif



/***
 *online_people.c - functions that show the people online
 *
 *        Copyright (C) 2011, Victor Jianfei Ye. All rights reserved.
 *
 *Purpose:
 *        defines the functions that show the people online
 *
 *******************************************************************************

#include <windows.h>
#include <stdio.h>
#include "resource.h"
#include "online_users.h"

extern PHOnlineUserInfo phOnlineUserInfo ;
extern volatile BOOL state ;

int del_clients(void) ;
int get_online_count(void);
void getOnlineUserList (char* szOnlineUserList) ;

/***
 *void refresh ( void ) - refresh the people online
 *
 *Purpose:
 *        Refresh the people online
 *
 *Entry:
 *
 *Exit:
 *
 *Exceptions:
 *
 *******************************************************************************

void refresh ( void )
{
    char szOnlineUsers[30] = "\0" ;
    char szOnlineUserList[10000] = "\0" ;
    getOnlineUserList (szOnlineUserList) ;
    sprintf ( szOnlineUsers , "%d\0" , get_online_count () ) ;
    SendMessage ( phOnlineUserInfo->hLableOnlinePeople ,
                  WM_SETTEXT , 0 , (LPARAM)(PCSTR)szOnlineUsers ) ;
    SendMessage ( phOnlineUserInfo->hOnlineUserList ,
                  WM_SETTEXT , 0 , (LPARAM)(PCSTR)szOnlineUserList ) ;
```

```
}

/***
 *DWORD APIENTRY SetOnlinePeople ( LPVOID threadNum ) - a thread
                                that shows the people online
 *
 *Purpose:
 *        Shows the people online every six seconds.
 *
 *Entry:
 *        LPVOID  threadNum   - the parameter that pass to this thread, useless
 *
 *Exit:
 *        Return 0
 *
 *Exceptions:
 *
 *******************************************************************************/

DWORD APIENTRY SetOnlinePeople ( LPVOID threadNum )
{
    while(state)
    {
        del_clients( ) ;
        refresh ( ) ;
        Sleep ( 6000 ) ;
    }
    return 0 ;
}



#include <winsock2.h>
#include "common.h"
#include "resource.h"
#include "client_structs.h"

volatile BOOL state = FALSE ;
volatile HWND hCurDlg = NULL ;

void refresh(void) ;
int init_clients(void);     // initialize users' linklist
void set_client(client *clt,SOCKET sck,SOCKADDR_IN addr,
       char *user,char *password,char *nickname,
       int state,SYSTEMTIME time,struct client * next) ;
int ins_clients(client vertex);  // insert a user into a linklist
int dst_clients(void);           //destroy the users' linklist

DWORD APIENTRY GetMessageFromClient(LPVOID threadNum) ;
```

```
DWORD APIENTRY WinSock ( LPVOID threadNum )
{
    // call Socket library (initialize)
    WSADATA sd;
    SYSTEMTIME nowtime;
    WORD ver=0x0101;
    SOCKET svrSck,sck;
    DWORD arg;
    SOCKADDR_IN addr;
    fd_set rfd,efd;
    SOCKADDR_IN ad;
    struct timeval timeout;
    int len;
    char szMessage[1000];
    client clt;
    HINSTANCE hInstance = *(HINSTANCE *) threadNum ;

    HANDLE hThread = NULL ;
    DWORD ThreadID = 0L ;


    init_clients(); // Initialize users' list
    if(WSAStartup(ver,&sd))
    {
        LoadString ( hInstance , IDS_INIT_FAILURE ,
                            szMessage , sizeof (szMessage ) ) ;
        MessageBox ( hCurDlg , szMessage , szCaption , MB_ICONERROR ) ;
        ExitProcess ( EXIT_FAILURE ) ;
    }
    // Create Socket
    svrSck=socket(AF_INET,SOCK_STREAM,0);
    if(svrSck==INVALID_SOCKET)
    {
        WSACleanup ( );
        LoadString ( hInstance , IDS_CREATION_FAILED ,
                            szMessage , sizeof ( szMessage ) ) ;
        MessageBox ( hCurDlg , szMessage ,
                            szCaption , MB_ICONERROR ) ;
        ExitProcess ( EXIT_FAILURE ) ;
    }
    // Set Socket running mode
    arg=1;
    ioctlsocket(svrSck,FIONBIO,&arg); // Set non-blocking mode
    // bind
    addr.sin_family=AF_INET;
    addr.sin_port=htons(2007);
    addr.sin_addr.s_addr = htonl(ADDR_ANY);
    if(bind(svrSck,(SOCKADDR*)&addr,sizeof(SOCKADDR)) == SOCKET_ERROR)
    {
        shutdown(svrSck,SD_BOTH);
```

```
        closesocket(svrSck);
        WSACleanup();
        LoadString ( hInstance , IDS_BIND_FAILED , szMessage , sizeof ( szMessa
        MessageBox ( hCurDlg , szMessage , szCaption , MB_ICONERROR ) ;
        ExitProcess ( EXIT_FAILURE ) ;
    }
    // start listening
    if( listen(svrSck,10) == SOCKET_ERROR )
    {
        closesocket ( svrSck ) ;
        LoadString ( hInstance , IDS_LISTEN_FALED , szMessage , sizeof ( szMess
        MessageBox ( hCurDlg , szMessage , szCaption , MB_ICONERROR ) ;
        ExitProcess ( EXIT_FAILURE ) ;
    }

    state = TRUE ;
    hThread = CreateThread (NULL, 0, GetMessageFromClient, 0, 0, &ThreadID) ;
    while ( state ) // Wait for client connection
    {
        FD_ZERO(&rfd); // used for judging whether there is connection
        FD_ZERO(&efd); // used for judging whether there's any error
        FD_SET(svrSck,&rfd) ;
        FD_SET(svrSck,&efd) ;
        timeout.tv_sec=0 ;
        timeout.tv_usec=10000 ; // set timeout as 10 second
        if( select(0,&rfd,NULL,&efd,&timeout) )
        {
            if(FD_ISSET(svrSck,&rfd)) // if there's a connction
            {
                len = sizeof(SOCKADDR_IN);
                sck = accept(svrSck,(SOCKADDR*)&ad,&len);
                if( sck != INVALID_SOCKET )
                {
                    GetLocalTime(&nowtime);
                    set_client (&clt,sck,ad,"","","",TRUE,nowtime,NULL);
                    ins_clients(clt);
                    refresh( ) ;
                }
            }
            else if( FD_ISSET( svrSck,&efd ) ) // Something goes wrong
            {
                shutdown(svrSck,SD_BOTH);
                closesocket(svrSck);
                WSACleanup();
                LoadString ( hInstance , IDS_SOCKET_ERROR ,
                                        szMessage , sizeof ( szMessage ) ) ;
                MessageBox ( hCurDlg , szMessage , szCaption , MB_ICONERROR ) ;
                ExitProcess ( EXIT_FAILURE ) ;
            }
        }
```

```
        Sleep(10);
    }
    TerminateThread ( hThread , 0 ) ;
    CloseHandle ( hThread ) ;
    hThread = NULL ;
    dst_clients( );
    shutdown(svrSck,SD_BOTH);
    closesocket(svrSck);
    WSACleanup();
    return 0L;
}



/***
 *sock_init_dlg.c - Socket-initializing-dialog function
 *
 *        Copyright (C) 2011, Victor Jianfei Ye. All rights reserved.
 *
 *Purpose:
 *        defines the callback function of the socket initializing
 *
 ****************************************************************/

#include <windows.h>
#include "common.h"
#include "resource.h"

extern volatile HWND hCurDlg ;
extern volatile BOOL state ;
extern HANDLE hThread ;

DWORD APIENTRY WinSock ( LPVOID threadNum ) ;

/***
 *BOOL CALLBACK SocketDlgProc ( HWND hDlg , UINT uMsg ,
 *                WPARAM wParam , LPARAM lParam ) - the callback function
 *                                  of the IDD_SOCKET_INIT dialog
 *
 *Purpose:
 *        Deal with the messages that send to the IDD_SOCKET_INIT dialog.
 *
 *Entry:
 *        HWND    hDlg      - the handle to the IDD_SOCKET_INIT dialog
 *        UINT    uMsg      - the type of the message
 *        WPARAM  wParam    - the value passed as a parameter to a window
 *                                    procedure or callback function
 *        int     nShowCmd  - the 32-bit value passed as a parameter to a
 *                                  window procedure or callback function
 *
```

```
*Exit:
*         Return FALSE
*
*Exceptions:
*
*******************************************************************************

BOOL CALLBACK SocketDlgProc ( HWND hDlg , UINT uMsg , WPARAM wParam , LPARAM lP
{

    DWORD ThreadID = 0L ;
    static HINSTANCE hInstance = NULL ;
    HICON hIcon = NULL ;

    if ( state )
    {
        hCurDlg = NULL ;
        EndDialog ( hDlg , 0 ) ;
        return FALSE ;
    }

    switch ( uMsg )
    {
    case WM_INITDIALOG:
        hInstance = *(HINSTANCE *) lParam ;
        hIcon = LoadIcon ( hInstance , MAKEINTRESOURCE ( IDI_ICON ) ) ;
        SendMessage ( hDlg, WM_SETICON , TRUE , (LPARAM) hIcon ) ;
        SendMessage ( hDlg, WM_SETICON , FALSE , (LPARAM) hIcon ) ;
        hCurDlg = hDlg ;

        return FALSE ;

    case WM_SHOWWINDOW:
        hThread = CreateThread ( NULL, 0, WinSock ,
                    (LPVOID)&hInstance , 0 , &ThreadID );
        return FALSE;
    case WM_COMMAND:
        switch ( LOWORD ( wParam ) )
        {
        case IDCANCEL:
            ExitProcess(0);
            return FALSE ;
        default:
            break ;
        }
        return FALSE ;
    case WM_CLOSE:
        hCurDlg = NULL ;
        ExitProcess ( 0 ) ;
        return FALSE ;
```

```
    default:
        return FALSE ;
    }
}



/***
 *SaveMessages.c - message-saving function
 *
 *        Copyright (C) 2011, Victor Jianfei Ye. All rights reserved.
 *
 *Purpose:
 *        defines the message-saving function
 *
 *******************************************************************/

#include <stdio.h>
#include <windows.h>
#include "CommonDialog.h"

/***
 *BOOL SaveMessages ( HWND hDlg , LPCTSTR pstrBuffer ) - the function
 *                            to save messages to the text file
 *
 *Purpose:
 *        Save messages to the text file.
 *
 *Entry:
 *        HWND      hDlg          - the handle to the dialog
 *        LPCTSTR   pstrBuffer    - the constant character
 *                            pointer to the string of the messages
 *
 *Exit:
 *        Success: TRUE
 *        Failure: FALSE
 *
 *Exceptions:
 *
 *******************************************************************************

BOOL SaveMessages ( HWND hDlg , LPCTSTR pstrBuffer )
{
    FILE * fp = NULL ;
    TCHAR pstrFilePath[MAX_PATH] = TEXT("\0") ;
    TCHAR pstrFileName[MAX_PATH] = TEXT("\0") ;
    LPCTSTR pstrFilter = TEXT ("Text␣Files␣(*.txt)\0*.txt\0") \
                         TEXT ("All␣Files␣(*.*)\0*.*\0" ) \
                         TEXT ("\0") ;
```

```
    if ( SaveAsDlg ( hDlg , pstrFilter , pstrFilePath , pstrFileName , TEXT("tx
    {
        fp = fopen ( pstrFilePath , TEXT("w") ) ;
        if (fp == NULL )
            return FALSE ;
        fprintf ( fp , pstrBuffer ) ;
        fclose (fp) ;
        fp = NULL ;
    }
    return TRUE ;
}



/***
 *messages.c - messages treating functions
 *
 *        Copyright (C) 2011, Victor Jianfei Ye. All rights reserved.
 *
 *Purpose:
 *        defines messages treating functions
 *
 *******************************************************************/

#include <windows.h>
#include "resource.h"
#include "client_structs.h"
void refresh ( void );

#define BUFLEN 65500

char buf_server[BUFLEN]="\0";
extern volatile BOOL state ;

pclient getlist_head(void) ;
void operate_msg (pclient clt,char *msg) ;
void getsocket(client clt,SOCKET *sck) ;
pclient next(pclient bef);
void sendall(const char *bufer) ;
BOOL ShowMessage ( LPCTSTR szMessage ) ;

/***
 *DWORD APIENTRY GetMessageFromClient(LPVOID threadNum) - get message from clie
 *
 *Purpose:
 *        Get message from client.
 *
 *Entry:
 *        LPVOID   threadNum    - the thread number pass to the thread, useless
 *
```

```
 *Exit:
 *        Return 0
 *
 *Exceptions:
 *
 ******************************************************************************

DWORD APIENTRY GetMessageFromClient(LPVOID threadNum)
{
    pclient pclt;
    SOCKET sck;
    int bytesRecv ;
    char szBufferFromClient[BUFLEN] = "\0" ;
    while(state)
    {
        pclt = getlist_head();
        while (pclt)
        {
            getsocket((*pclt),&sck);
            bytesRecv = recv (sck,szBufferFromClient,
                    sizeof(szBufferFromClient),0) ;
            if (bytesRecv == 0 ||
                    bytesRecv == WSAECONNRESET ||
                    szBufferFromClient[0]== '\a' )
            {
                pclt->state= FALSE ;
                szBufferFromClient[0] = '0' ;
                refresh();
            }
            else if( bytesRecv != -1 )
            {
                szBufferFromClient[bytesRecv-1] = '\0' ;
                operate_msg (pclt,szBufferFromClient) ;
            }
            pclt = next(pclt);
        }
        Sleep(10);
    }
    return 0 ;
}

/***
 *void SendMessageToClient( HINSTANCE hInstance ,
 *             const char * szMessageFromServer) - send message from client
 *
 *Purpose:
 *        Send message from client.
 *
 *Entry:
 *        HINSTANCE    hInstance          - the handle to this instance
```

```
 *        const char * szMessageFromServer - the constant character
 *                                           pointer to the message from server
 *
 *Exit:
 *
 *Exceptions:
 *
 ************************************************************************/

void SendMessageToClient( HINSTANCE hInstance, const char *szMessageFromServer)
{
    char szBufferServer[BUFLEN] = "\0" ;
    LoadString ( hInstance , IDS_SERVER_PROMPT ,
                 szBufferServer , sizeof (szBufferServer) ) ;
    strcat(szBufferServer, szMessageFromServer );
    ShowMessage ( szBufferServer ) ;
    sendall(szBufferServer);
}



/***
 *main_dialog.c - main_dialog callback functions
 *
 *        Copyright (C) 2011, Victor Jianfei Ye. All rights reserved.
 *
 *Purpose:
 *        defines main_dialog callback functions
 *
 ************************************************************/

#include <windows.h>
#include <string.h>
#include <stdlib.h>
#include "common.h"
#include "online_users.h"
#include "resource.h"

PHOnlineUserInfo phOnlineUserInfo ;
extern volatile HWND hCurDlg ;
extern volatile BOOL state ;

BOOL SaveMessages ( HWND hDlg , LPCTSTR pstrBuffer ) ;
DWORD APIENTRY SetOnlinePeople ( LPVOID threadNum ) ;
void SendMessageToClient ( HINSTANCE hInstance , const char *szMessageFromServe

HWND hShowMessage = NULL ;

/***
 *BOOL ShowMessage ( LPCTSTR szMessage ) - show the messages in the edit box
```

```
 *
 *Purpose:
 *        Show the messages in the edit box
 *
 *Entry:
 *        LPCTSTR    szMessage    – the constant character pointer
 *                          to the C–style string of the message
 *
 *Exit:
 *        Success:  TRUE
 *        Failure:  FALSE
 *
 *Exceptions:
 *
 ********************************************************************************

BOOL ShowMessage ( LPCTSTR szMessage )
{
    int cchTextMax = 0 ;
    PTSTR lpszText = NULL ;
    PTSTR lpszTextTemp = NULL ;
    int MinPos =0 , MaxPos =0 ;
    cchTextMax = 1 + SendMessage ( hShowMessage , WM_GETTEXTLENGTH , 0 , 0 ) ;
    lpszText = (PTSTR) malloc ( cchTextMax * sizeof(TCHAR) ) ;
    if ( lpszText == NULL )
    {
        return FALSE ;
    }
    else
    {
        SendMessage ( hShowMessage , WM_GETTEXT , (WPARAM) cchTextMax , (LPARAM
        lpszTextTemp = (PTSTR) realloc ( lpszText ,
                  (cchTextMax + strlen (szMessage)+4) * sizeof (TCHAR) ) ;
        if ( lpszTextTemp == NULL )
        {
            free ( lpszText ) ;
            lpszText = NULL ;
            return FALSE ;
        }
        else
        {
            lpszText = lpszTextTemp ;
            strcat ( lpszText , szMessage ) ;
            strcat ( lpszText , "\r\n\r\n" ) ;
            SendMessage( hShowMessage , WM_SETTEXT , 0 , (LPARAM)(LPCTSTR)lpszT
            if ( GetScrollRange ( hShowMessage , SB_VERT , &MinPos , & MaxPos )
            {
                SetScrollPos ( hShowMessage , SB_VERT , MaxPos , TRUE ) ;
                SendMessage ( hShowMessage , WM_VSCROLL ,
                                MAKEWPARAM ( SB_BOTTOM , 0 ), (LPARAM)NULL )
```

```
                Sleep ( 30 ) ;
                free ( lpszText ) ;
                lpszText = NULL ;
                return TRUE ;
            }
            else
            {
                free ( lpszText ) ;
                lpszText = NULL ;
                return FALSE ;
            }
        }
    }
}

/***
 *BOOL CALLBACK MainDialogProc ( HWND hDlg , UINT uMsg ,
 *                    WPARAM wParam , LPARAM lParam ) - the callback
 *                                      function of the main dialog
 *
 *Purpose:
 *        Deal with the messages that send to the main dialog.
 *
 *Entry:
 *        HWND        hDlg            - the handle to the main dialog
 *        UINT        uMsg            - the type of the message
 *        WPARAM        wParam        - the value passed as a parameter to
 *                                      a window procedure or callback function
 *        int             nShowCmd    - the 32-bit value passed as a parameter t
 *                                      a window procedure or callback function
 *
 *Exit:
 *        Return FALSE
 *
 *Exceptions:
 *
 *******************************************************************************

BOOL CALLBACK MainDialogProc ( HWND hDlg , UINT uMsg , WPARAM wParam , LPARAM l
{
    static HINSTANCE hInstance = NULL ;
    HICON hIcon = NULL ;
    HWND hDlgItem = NULL ;
    int iLength = 0 ;
    PTSTR pstrBuffer = NULL ;
    TCHAR szMsgBuffer[100] = TEXT("\0") ;

    static HANDLE hThreadSetOnlinePeople = NULL ;
    static DWORD ThreadIDSetOnlinePeople = 0L ;
    static HOnlineUserInfo hOnlineUserInfo ;
```

```
    switch ( uMsg )
    {
    case WM_INITDIALOG:
        hInstance = *(HINSTANCE *) lParam ;
        hIcon = LoadIcon ( hInstance , MAKEINTRESOURCE ( IDI_ICON ) ) ;
        SendMessage ( hDlg, WM_SETICON , TRUE , (LPARAM) hIcon ) ;
        SendMessage ( hDlg, WM_SETICON , FALSE , (LPARAM) hIcon ) ;
        hCurDlg = hDlg ;

        hShowMessage = GetDlgItem ( hDlg , IDC_MESSAGES ) ;
        if ( hShowMessage == NULL )  // fail to get the handle to the message t
        {
            LoadString ( hInstance , IDS_UNKNOWN_SYSTEM_ERROR ,
                        szMsgBuffer , sizeof ( szMsgBuffer )
) ;
            MessageBox ( hDlg , szMsgBuffer , szCaption , MB_ICONERROR ) ;
            ExitProcess ( EXIT_FAILURE ) ;
        }

        hDlgItem = GetDlgItem ( hDlg , IDC_ONLINE_PEOPLE ) ;
        if ( hDlgItem == NULL )  // fail to get the handle to the online_people
        {
            LoadString ( hInstance , IDS_UNKNOWN_SYSTEM_ERROR ,
                        szMsgBuffer , sizeof ( szMsgBuffer )
) ;
            MessageBox ( hDlg , szMsgBuffer , szCaption , MB_ICONERROR ) ;
            ExitProcess ( EXIT_FAILURE ) ;
        }
        else
        {
            hOnlineUserInfo.hLableOnlinePeople = hDlgItem ;
            hDlgItem = GetDlgItem ( hDlg , IDC_ONLINE_USER_LIST ) ;
            if ( hDlgItem == NULL )  // fail to get the handle to the online_pe
            {
                LoadString ( hInstance , IDS_UNKNOWN_SYSTEM_ERROR ,
                            szMsgBuffer , sizeof ( szMsgBuffer )
) ;
                MessageBox ( hDlg , szMsgBuffer , szCaption , MB_ICONERROR ) ;
                ExitProcess ( EXIT_FAILURE ) ;
            }
            else
            {
                hOnlineUserInfo.hOnlineUserList = hDlgItem ;
                phOnlineUserInfo = &hOnlineUserInfo ;
                hThreadSetOnlinePeople = CreateThread ( NULL , 0 ,
                            SetOnlinePeople , NULL , 0 , &ThreadIDSetOnlineP
            }
        }
```

```
            return FALSE ;
        case WM_COMMAND:
            switch ( LOWORD ( wParam ) )
            {
            case IDC_CLEAR:
                SetDlgItemText ( hDlg , IDC_MESSAGES , TEXT("") ) ;
                break ;
            case IDC_SAVE:
                hDlgItem = GetDlgItem ( hDlg , IDC_MESSAGES ) ;
                if ( hDlgItem == NULL )  // fail to get the handle to the message c
                {
                    LoadString ( hInstance , IDS_UNKNOWN_SYSTEM_ERROR ,
                                                szMsgBuffer , sizeof ( szMsgBuffer
) ;
                    MessageBox ( hDlg , szMsgBuffer , szCaption , MB_ICONERROR ) ;
                }
                else    // succeed in getting the handle to the message dialog
                {
                    iLength = (int)SendMessage ( hDlgItem , WM_GETTEXTLENGTH , 0 ,
                    pstrBuffer = (PTSTR) malloc ( (iLength+1) * sizeof (TCHAR) ) ;
                    if ( pstrBuffer == NULL )
                    {
                        LoadString ( hInstance , IDS_MEMORY_FAILED,
                                                szMsgBuffer, sizeof ( szMsgBuffer
) ;
                        MessageBox ( hDlg , szMsgBuffer , szCaption , MB_ICONERROR
                    }
                    else
                    {
                        if ( GetDlgItemText ( hDlg , IDC_MESSAGES , pstrBuffer , iL
                        {
                            if ( !SaveMessages ( hDlg , pstrBuffer ) )
                            {
                                free ( pstrBuffer ) ;
                                pstrBuffer = NULL ;
                                LoadString ( hInstance , IDS_FILE_OPEN_ERROR ,
                                                    szMsgBuffer , sizeof ( szMsgBuffe
) ;
                                MessageBox ( hDlg , szMsgBuffer , szCaption , MB_IC
                            }
                            else    // fail to write the file
                            {
                                free ( pstrBuffer ) ;
                                pstrBuffer = NULL ;
                            }
                        }
                        else  // no chatting records
                        {
                            free (pstrBuffer) ;
                            pstrBuffer = NULL ;
```

```
                        LoadString ( hInstance , IDS_NO_TEXT ,
                                                szMsgBuffer , sizeof ( szMsgBuffer )
) ;
                        MessageBox ( hDlg , szMsgBuffer , szCaption , MB_ICONWA
                    }
                }
            }
            break ;
        case IDC_SEND:
            hDlgItem = GetDlgItem ( hDlg , IDC_COMMAND_LINE ) ;
            if ( hDlgItem == NULL )  // fail to get the handle to the message d
            {
                LoadString ( hInstance , IDS_UNKNOWN_SYSTEM_ERROR , szMsgBuffer
) ;
                MessageBox ( hDlg , szMsgBuffer , szCaption , MB_ICONERROR ) ;
            }
            else    // succeed in getting the handle to the message dialog
            {
                iLength = (int)SendMessage ( hDlgItem , WM_GETTEXTLENGTH , 0 ,
                pstrBuffer = (PTSTR) malloc ( (iLength+1) * sizeof (TCHAR) ) ;
                SendMessage ( hDlgItem , WM_GETTEXT , (WPARAM)(iLength+1) ,
                                                    (LPARAM)pstrBuffer ) ;
                SendMessageToClient ( hInstance , pstrBuffer ) ;
                Sleep (20) ;
                free ( pstrBuffer ) ;
                pstrBuffer = NULL ;
                SendMessage ( hDlgItem , WM_SETTEXT , (WPARAM)0 ,
                                                    (LPARAM)(LPCTSTR)"\0" ) ;
            }
            break ;
        default:
            break ;
        }
        return FALSE ;

    case WM_CLOSE:
        TerminateThread ( hThreadSetOnlinePeople , 0 ) ;
        state = FALSE ;
        CloseHandle ( hThreadSetOnlinePeople ) ;
        hThreadSetOnlinePeople = NULL ;
        EndDialog ( hDlg , 0 ) ;
        return FALSE ;
    default:
        return FALSE ;
    }
}



/***
```

```
 *main.c - WinMain function
 *
 *        Copyright (C) 2011, Victor Jianfei Ye. All rights reserved.
 *
 *Purpose:
 *        defines WinMain function
 *
 ****************************************************************/

#include <windows.h>
#include "resource.h"

HANDLE hThread = NULL ;
TCHAR szCaption[100] = TEXT("\0") ;

BOOL CALLBACK MainDialogProc ( HWND hDlg , UINT uMsg ,
                               WPARAM wParam , LPARAM lParam ) ;
BOOL CALLBACK SocketDlgProc ( HWND hDlg , UINT uMsg ,
                              WPARAM wParam , LPARAM lParam ) ;

/***
 *int WINAPI WinMain ( HINSTANCE hInstance , HINSTANCE hPrevInstance ,
 *             LPSTR lpCmdLine , int nShowCmd ) - the entrance
 *                           of this program
 *
 *Purpose:
 *        Startup this program and call the other functions,
 *             called by the Windows operating system.
 *
 *Entry:
 *        HINSTANCE    hInstance    - the handle to this instance
 *        HINSTANCE    hPrevInstance    - the handle to the previous
 *                     instance, it must be NULL under Win32 platform
 *        LPSTR        lpCmdLine    - the command-line parameter
 *        int         nShowCmd    - the startup message of
 *                          the way of the style of the program
 *
 *Exit:
 *        Success:  EXIT_SUCCESS
 *        Failure:  EXIT_FAILURE
 *
 *Exceptions:
 *
 ****************************************************************/

int APIENTRY WinMain ( HINSTANCE hInstance , HINSTANCE hPrevInstance ,
                LPSTR lpCmdLine, int nCmdShow )
{
    TCHAR szMsgBuffer[100] = TEXT("\0") ;
    LoadString ( hInstance , IDS_CAPTION , szCaption , sizeof (szCaption ) ) ;
```

```
    if ( -1 == DialogBoxParam ( hInstance , MAKEINTRESOURCE(IDD_SOCKET_INIT) ,
                NULL , (DLGPROC)SocketDlgProc , (LPARAM)&hInstance ) )
    {
        LoadString ( hInstance , IDS_DIALOGBOX_INIT_ERROR , szMsgBuffer,
                            sizeof (szMsgBuffer) ) ;
        MessageBox ( NULL , szMsgBuffer , szCaption , MB_ICONERROR ) ;
        return EXIT_FAILURE ;
    }

    if ( -1 == DialogBoxParam ( hInstance , MAKEINTRESOURCE(IDD_MAINDIALOG) ,
                NULL , (DLGPROC)MainDialogProc , (LPARAM)&hInstance ) )
    {
        LoadString ( hInstance , IDS_DIALOGBOX_INIT_ERROR , szMsgBuffer,
                            sizeof (szMsgBuffer) ) ;
        MessageBox ( NULL , szMsgBuffer , szCaption , MB_ICONERROR ) ;
        return EXIT_FAILURE ;
    }

    WaitForSingleObject ( hThread , 5000 ) ;
    TerminateThread ( hThread , 0 ) ;
    CloseHandle ( hThread ) ;
    hThread = NULL ;

    return EXIT_SUCCESS ;
}



//{{NO_DEPENDENCIES}}
// Microsoft Developer Studio generated include file.
// Used by server.rc
//
#define IDS_CAPTION                     1
#define IDS_DIALOGBOX_INIT_ERROR        2
#define IDS_MEMORY_FAILED               3
#define IDS_UNKNOWN_SYSTEM_ERROR        4
#define IDS_FILE_OPEN_ERROR             5
#define IDS_INIT_FAILURE                6
#define IDS_PROMPT                      7
#define IDS_CREATION_FAILED             8
#define IDS_BIND_FAILED                 9
#define IDS_LISTEN_FALED                10
#define IDS_SOCKET_ERROR                11
#define IDS_SERVER_PROMPT               12
#define IDS_NO_TEXT                     13
#define IDI_ICON                        102
#define IDD_MAINDIALOG                  103
#define IDD_SOCKET_INIT                 104
#define IDC_MESSAGES                    1000
```

```
#define IDC_COMMAND_LINE                1001
#define IDC_CLEAR                       1002
#define IDC_SAVE                        1003
#define IDC_SEND                        1004
#define IDC_DATA                        1005
#define IDC_ONLINE_PEOPLE               1006
#define IDC_ONLINE_USER_LIST            1007

// Next default values for new objects
//
#ifdef APSTUDIO_INVOKED
#ifndef APSTUDIO_READONLY_SYMBOLS
#define _APS_NO_MFC                     1
#define _APS_NEXT_RESOURCE_VALUE        105
#define _APS_NEXT_COMMAND_VALUE         40001
#define _APS_NEXT_CONTROL_VALUE         1008
#define _APS_NEXT_SYMED_VALUE           101
#endif
#endif
```

1.5.3 服务器端程序资源文件代码

```
//Microsoft Developer Studio generated resource script.
//
#include "resource.h"

#define APSTUDIO_READONLY_SYMBOLS
//////////////////////////////////////////////////////////////////////////////
//
// Generated from the TEXTINCLUDE 2 resource.
//
#include "afxres.h"

//////////////////////////////////////////////////////////////////////////////
#undef APSTUDIO_READONLY_SYMBOLS

//////////////////////////////////////////////////////////////////////////////
// Chinese (P.R.C.) resources

#if !defined(AFX_RESOURCE_DLL) || defined(AFX_TARG_CHS)
#ifdef _WIN32
LANGUAGE LANG_CHINESE, SUBLANG_CHINESE_SIMPLIFIED
#pragma code_page(936)
#endif //_WIN32

#ifndef _MAC
```

```
//////////////////////////////////////////////////////////////////////////////
//
// Version
//

VS_VERSION_INFO VERSIONINFO
 FILEVERSION 1,0,0,1
 PRODUCTVERSION 1,0,0,1
 FILEFLAGSMASK 0x3fL
#ifdef _DEBUG
 FILEFLAGS 0x1L
#else
 FILEFLAGS 0x0L
#endif
 FILEOS 0x40004L
 FILETYPE 0x1L
 FILESUBTYPE 0x0L
BEGIN
    BLOCK "StringFileInfo"
    BEGIN
        BLOCK "080004b0"
        BEGIN
            VALUE "Comments", "内核设计——周开锋WinSock␣\r\n\r\n␣图
形用户界面（␣GUI␣）设计——叶剑飞\0"
            VALUE "CompanyName", "周开锋、叶剑飞\0"
            VALUE "FileDescription", "这是一个聊天软件的服务
器端程序\0"
            VALUE "FileVersion", "1,␣0,␣0,␣1\0"
            VALUE "InternalName", "server\0"
            VALUE "LegalCopyright", "版权所有 (␣C␣)␣␣周开锋、
叶剑飞\0"
            VALUE "LegalTrademarks", "周开锋、叶剑飞\0"
            VALUE "OriginalFilename", "server.exe\0"
            VALUE "PrivateBuild", "\0"
            VALUE "ProductName", "聊天软件服务器端（图形用户界
面）\0"
            VALUE "ProductVersion", "1,␣0,␣0,␣1\0"
            VALUE "SpecialBuild", "\0"
        END
    END
    BLOCK "VarFileInfo"
    BEGIN
        VALUE "Translation", 0x800, 1200
    END
END

#endif    // !_MAC


#ifdef APSTUDIO_INVOKED
```

```
//////////////////////////////////////////////////////////////////////////////
//
// TEXTINCLUDE
//

1 TEXTINCLUDE DISCARDABLE
BEGIN
    "resource.h\0"
END

2 TEXTINCLUDE DISCARDABLE
BEGIN
    "#include␣""afxres.h""\r\n"
    "\0"
END

3 TEXTINCLUDE DISCARDABLE
BEGIN
    "\r\n"
    "\0"
END

#endif    // APSTUDIO_INVOKED


//////////////////////////////////////////////////////////////////////////////
//
// Icon
//

// Icon with lowest ID value placed first to ensure application icon
// remains consistent on all systems.
IDI_ICON                ICON    DISCARDABLE     "talking.ico"

//////////////////////////////////////////////////////////////////////////////
//
// Dialog
//

IDD_MAINDIALOG DIALOG DISCARDABLE  100, 70, 307, 215
STYLE DS_MODALFRAME | WS_MINIMIZEBOX | WS_POPUP | WS_CAPTION | WS_SYSMENU
CAPTION "聊天软件（服务器端）"
FONT 12, "宋体"
BEGIN
    EDITTEXT        IDC_MESSAGES,7,6,190,119,ES_MULTILINE | ES_AUTOHSCROLL |
                    ES_READONLY | ES_WANTRETURN | WS_VSCROLL | WS_HSCROLL
    EDITTEXT        IDC_COMMAND_LINE,7,128,192,55,ES_MULTILINE |
                    ES_AUTOHSCROLL | ES_WANTRETURN | WS_VSCROLL | WS_HSCROLL
    PUSHBUTTON      "清空聊天记录",IDC_CLEAR,7,194,50,14
    PUSHBUTTON      "保存聊天记录",IDC_SAVE,66,194,50,14
```

```
    PUSHBUTTON      "发送",IDC_SEND,131,194,50,14
    LTEXT           "在线人数：",IDC_STATIC,205,10,41,8
    LTEXT           "0",IDC_ONLINE_PEOPLE,253,10,35,8
    EDITTEXT        IDC_ONLINE_USER_LIST,204,35,98,173,ES_MULTILINE |
                    ES_AUTOHSCROLL | ES_READONLY | ES_WANTRETURN |
                    WS_VSCROLL
    LTEXT           "在线用户列表：",IDC_STATIC,204,24,57,8
END

IDD_SOCKET_INIT DIALOG DISCARDABLE  100, 78, 106, 69
STYLE DS_MODALFRAME | WS_POPUP | WS_CAPTION | WS_SYSMENU
CAPTION "初始化中Socket..."
FONT 10, "System"
BEGIN
    PUSHBUTTON      "退出",IDCANCEL,28,39,50,14
    CTEXT           "初始化中Socket...",IDC_STATIC,17,15,71,14,
                    SS_CENTERIMAGE
END


////////////////////////////////////////////////////////////////////////////
//
// DESIGNINFO
//

#ifdef APSTUDIO_INVOKED
GUIDELINES DESIGNINFO DISCARDABLE
BEGIN
    IDD_MAINDIALOG, DIALOG
    BEGIN
        LEFTMARGIN, 7
        RIGHTMARGIN, 302
        TOPMARGIN, 6
        BOTTOMMARGIN, 208
    END

    IDD_SOCKET_INIT, DIALOG
    BEGIN
        LEFTMARGIN, 7
        RIGHTMARGIN, 99
        TOPMARGIN, 7
        BOTTOMMARGIN, 62
    END
END
#endif    // APSTUDIO_INVOKED


////////////////////////////////////////////////////////////////////////////
//
// String Table
```

```
//

STRINGTABLE DISCARDABLE
BEGIN
    IDS_CAPTION                 "聊天软件（服务器端）"
    IDS_DIALOGBOX_INIT_ERROR    "警告：对话框初始化失败！"
    IDS_MEMORY_FAILED           "警告：内存不足，操作失败！"
    IDS_UNKNOWN_SYSTEM_ERROR    "警告：系统发生未知错误！"
    IDS_FILE_OPEN_ERROR         "警告：文件打开失败！"
    IDS_INIT_FAILURE            "Socket 初始化失败！"
    IDS_CREATION_FAILED         "创建 Socket 失败！"
    IDS_BIND_FAILED             "绑定失败！"
    IDS_LISTEN_FALED            "监听失败！"
    IDS_SOCKET_ERROR            "Socket 出现错误！"
    IDS_SERVER_PROMPT           "服务器端提示：\r\n"
    IDS_NO_TEXT                 "聊天记录为空！"
END

#endif    // Chinese (P.R.C.) resources
//////////////////////////////////////////////////////////////////////////////



#ifndef APSTUDIO_INVOKED
//////////////////////////////////////////////////////////////////////////////
//
// Generated from the TEXTINCLUDE 3 resource.
//


//////////////////////////////////////////////////////////////////////////////
#endif    // not APSTUDIO_INVOKED
```

1.5.4 客户端程序 C 语言源代码

```
/***
 *common.h - declarations for print function
 *
 *        Copyright (C) 2011, Victor Jianfei Ye. All rights reserved.
 *
 *Purpose:
 *        This file declare the print function.
 *
 ****/

#ifndef COMMON_H
```

```
#define COMMON_H

int print ( const char *msg ) ;

#endif
```

```
/***
 *core.h - declarations for WinSock functions
 *
 *        Copyright (C) 2011, Victor Jianfei Ye. All rights reserved.
 *
 *Purpose:
 *        This file declare the WinSock functions.
 *
 ****/

#ifndef CORE_H
#define CORE_H

BOOL send_msg ( HINSTANCE hInstance ) ;
BOOL receive_msg ( const char *rec_msg , HINSTANCE hInstance ) ;
DWORD WINAPI connection (LPVOID threadNum) ;
void CloseConnection ( void ) ;

#endif
```

```
/***
 *core.c - WinSock functions
 *
 *        Copyright (C) 2011, Victor Jianfei Ye and Zhou Kaifeng.
 *                            All rights reserved.
 *
 *Purpose:
 *        defines WinSock functions
 *
 *******************************************************************/

#include <stdio.h>
#include <winsock2.h>
#include "resource.h"
#include "core.h"
#include "common.h"
#pragma comment(lib,"ws2_32.lib")

volatile int status = 0 ;
```

```
volatile int bEnd = 0 ;
volatile SOCKET cltSck;

extern char IP[20] ;
extern HWND hMainDlg ;
extern char szTitle[102] ;
extern HANDLE hThread ;

/***
 *DWORD WINAPI connection ( LPVOID threadNum ) - connect to
 *                                the server by WinSock technology
 *
 *Purpose:
 *        Connect to the server.
 *
 *Entry:
 *        LPVOID   threadNum    - the pointer to the handle of this instance
 *
 *Exit:
 *        Success:  0
 *
 *Exceptions:
 *
 *******************************************************************************

DWORD WINAPI connection ( LPVOID threadNum )
{
    HINSTANCE hInstance = * ( (HINSTANCE *)threadNum ) ;
    char szConnectionMessage[50] , szRecvBuf[65500] ;
    WSADATA wsaData;
    SOCKADDR_IN clientService;
    DWORD arg = 1 ;
    HWND hGetText = NULL ;
    int MinPos , MaxPos ;
    int bytesRecv = SOCKET_ERROR ;

    // Initialize
    int iResult = WSAStartup ( MAKEWORD (2,2), &wsaData );

    if ( iResult != NO_ERROR )
    {
        LoadString ( hInstance , IDS_SOCKET_INIT_ERROR ,
              szConnectionMessage , sizeof ( szConnectionMessage ) ) ;
        print ( szConnectionMessage ) ;
        ExitProcess ( 1 ) ;
    }

    // Create socket
    cltSck = socket ( AF_INET, SOCK_STREAM, IPPROTO_TCP );
```

```
if ( cltSck == INVALID_SOCKET )
{
    WSACleanup() ;
    LoadString ( hInstance , IDS_SOCKET_CREATION_ERROR ,
           szConnectionMessage , sizeof ( szConnectionMessage ) ) ;
    print ( szConnectionMessage );
    return 1 ;
}

// Connect to server

clientService.sin_family = AF_INET ;
clientService.sin_addr.s_addr = inet_addr( IP ) ;
clientService.sin_port = htons( 2007 ) ;

while ( connect ( cltSck, (SOCKADDR*) &clientService,
              sizeof(clientService)) == SOCKET_ERROR )
{
    ;
}


// Send and receive messages
status = 1 ;

while ( bEnd == 0 )
{
    bytesRecv = recv ( cltSck, szRecvBuf, sizeof ( szRecvBuf ), 0 ) ;
    if ( bytesRecv == 0 || bytesRecv == WSAECONNRESET )
    {
        shutdown ( cltSck , SD_BOTH ) ;
        closesocket ( cltSck ) ;
        WSACleanup ( ) ;
        LoadString ( hInstance , IDS_CONNECTION_CLOSED ,
               szConnectionMessage , sizeof ( szConnectionMessage ) ) ;
        MessageBox ( hMainDlg , szConnectionMessage ,
                     szTitle , MB_ICONWARNING ) ;
        SendMessage ( hMainDlg, WM_CLOSE , 0 , 0 ) ;
        return 0 ;
    }
    else if ( bytesRecv != SOCKET_ERROR )
    {
        szRecvBuf[bytesRecv] = '\0' ;
        strcat ( szRecvBuf , "\r\n\r\n" ) ;
        receive_msg ( szRecvBuf , hInstance ) ;
        hGetText = GetDlgItem ( hMainDlg , IDC_GET_TEXT ) ;
        if ( GetScrollRange ( hGetText , SB_VERT , &MinPos , & MaxPos ) )
        {
            SetScrollPos ( hGetText , SB_VERT , MaxPos , TRUE ) ;
            SendMessage ( hGetText , WM_VSCROLL ,
```

```
                                MAKEWPARAM ( SB_BOTTOM , 0 ), (LPARAM)NULL ) ;
            }
        }
    }
    shutdown ( cltSck , SD_BOTH ) ;
    closesocket ( cltSck ) ;
    WSACleanup ( ) ;
    return 0 ;
}

/***
 *void CloseConnection ( void ) - close the connection to the server
 *
 *Purpose:
 *        Close the connection to the server.
 *
 *Exceptions:
 *
 *******************************************************************/

void CloseConnection ( void )
{
    send ( cltSck,"\a", strlen("\a") , 0 ) ;
    TerminateThread ( hThread , 0 ) ;
    shutdown ( cltSck , SD_BOTH ) ;
    closesocket ( cltSck ) ;
    WSACleanup ( ) ;
}



/***
 *main.c - main function
 *
 *        Copyright (C) 2011, Victor Jianfei Ye. All rights reserved.
 *
 *Purpose:
 *        defines WinMain function
 *
 *******************************************************************/

#include <windows.h>
#include "resource.h"
#include "main_dialog.h"
#include "server_ip.h"

char szTitle[102] ;
extern HANDLE hThread ;
extern volatile int bEnd ;
```

```
/***
 *int WINAPI WinMain ( HINSTANCE hInstance ,
 *           HINSTANCE hPrevInstance , LPSTR lpCmdLine ,
 *           int nShowCmd ) - the entrance of this program
 *
 *Purpose:
 *        Startup this program and call the other functions,
 *                    called by the Windows operating system.
 *
 *Entry:
 *        HINSTANCE   hInstance      - the handle to this instance
 *        HINSTANCE   hPrevInstance  - the handle to the previous
 *                                     instance, it must be NULL
                                       under Win32 platform
 *        LPSTR       lpCmdLine      - the command-line parameter
 *        int         nShowCmd       - the startup message of the
                                       way of the style of the program
 *
 *Exit:
 *        Success:  EXIT_SUCCESS
 *        Failure:  EXIT_FAILURE
 *
 *Exceptions:
 *
 *******************************************************************************/

int WINAPI WinMain ( HINSTANCE hInstance , HINSTANCE hPrevInstance ,
                                LPSTR lpCmdLine , int nShowCmd )
{
    char szWindowCreationFailure[102] ;
    LoadString ( hInstance , IDS_WINDOW_CREATION_FAILURE ,
            szWindowCreationFailure , sizeof ( szWindowCreationFailure ) ) ;
    LoadString ( hInstance , IDS_TITLE , szTitle , sizeof ( szTitle ) ) ;
    if ( -1 == DialogBoxParam ( hInstance ,
                    MAKEINTRESOURCE ( IDD_SEVER_IP ) , NULL ,
                    (DLGPROC) SeverIp , (LPARAM) hInstance ) )
    {
        MessageBox ( NULL, szWindowCreationFailure ,
                        szTitle , MB_ICONERROR) ;
        return EXIT_FAILURE ;
    }
    if ( -1 == DialogBoxParam ( hInstance ,
                  MAKEINTRESOURCE ( IDD_MAIN_DIALOG ) ,
                  NULL , (DLGPROC) MainDialog , (LPARAM) hInstance ) )
    {
        MessageBox ( NULL, szWindowCreationFailure ,
                          szTitle , MB_ICONERROR) ;
        return EXIT_FAILURE ;
    }
    bEnd = 1 ;
```

```
    WaitForSingleObject ( hThread , INFINITE ) ;
    return EXIT_SUCCESS ;
}



/***
 *main_dialog.h - declarations for main-dialog callback function
 *
 *        Copyright (C) 2011, Victor Jianfei Ye. All rights reserved.
 *
 *Purpose:
 *        This file declare the main-dialog callback function.
 *
 ****/

#ifndef MAIN_DIALOG_H
#define MAIN_DIALOG_H

BOOL CALLBACK MainDialog ( HWND hDlg , UINT uMsg ,
                    WPARAM wParam , LPARAM lParam ) ;

#endif



/***
 *main_dialog.c - main_dialog function
 *
 *        Copyright (C) 2011, Victor Jianfei Ye. All rights reserved.
 *
 *Purpose:
 *        defines main_dialog function
 *
 ****************************************************************/

#include <windows.h>
#include <string.h>
#include <stdlib.h>
#include "resource.h"
#include "main_dialog.h"
#include "core.h"
#include "common.h"
#include "SaveMessage.h"

volatile HWND hMainDlg = NULL ;
char *receive_text = NULL ;
extern char szTitle[102] ;
```

```
extern int bEnd ;

/***
 *BOOL CALLBACK MainDialog ( HWND hDlg , UINT uMsg ,
 *                   WPARAM wParam , LPARAM lParam ) - the callback
 *                   function of the main dialog
 *
 *Purpose:
 *         Deal with the messages that send to the main dialog.
 *
 *Entry:
 *         HWND   hDlg      - the handle to the main dialog
 *         UINT   uMsg      - the type of the message
 *         WPARAM wParam    - the value passed as a parameter
 *                            to a window procedure or
 *                            callback function
 *         int    nShowCmd - the 32-bit value passed as
 *                           a parameter to a window
 *                           procedure or callback function
 *
 *Exit:
 *         Return FALSE
 *
 *Exceptions:
 *
 *******************************************************************/

BOOL CALLBACK MainDialog ( HWND hDlg , UINT uMsg ,
                           WPARAM wParam , LPARAM lParam )
{
    static HINSTANCE hInstance = NULL ;
    HICON hIcon ;
    HWND hGetText = NULL ;
    char szMsgBuffer[100] = "\0" ;
    int iLength = 0 ;
    PTSTR pstrBuffer = NULL ;

    switch ( uMsg )
    {
    case WM_INITDIALOG :
        hInstance = (HINSTANCE)lParam ;
        hIcon = LoadIcon ( hInstance , MAKEINTRESOURCE ( IDI_ICON ) ) ;
        SendMessage ( hDlg, WM_SETICON , TRUE , (LPARAM) hIcon ) ;
        SendMessage ( hDlg, WM_SETICON , FALSE , (LPARAM) hIcon ) ;
        receive_text = (char *) malloc ( 165536 * sizeof(char) ) ;
        hMainDlg = hDlg ;
        return FALSE ;
    case WM_COMMAND :
        switch ( LOWORD (wParam) )
        {
```

```
            case IDC_SEND:
                send_msg ( hInstance ) ;
                break ;
            case IDC_CLEAR:
                SetDlgItemText ( hDlg , IDC_GET_TEXT , "" ) ;
                break ;
            case IDC_SAVE:
                hGetText = GetDlgItem ( hDlg , IDC_GET_TEXT ) ;
                if ( hGetText == NULL )  // fail to get the handle to the message d
                {
                    LoadString ( hInstance , IDS_UNKNOWN_SYSTEM_ERROR ,
                                 szMsgBuffer , sizeof ( szMsgBuffer )
) ;
                    MessageBox ( hDlg , szMsgBuffer , szTitle , MB_ICONERROR ) ;
                }
                else   // succeed in getting the handle to the message dialog
                {
                    iLength = (int)SendMessage ( hGetText ,
                                  WM_GETTEXTLENGTH , 0 , 0 ) ;
                    pstrBuffer = (PTSTR) malloc ( (iLength+1)
                                  * sizeof (TCHAR) ) ;
                    if ( pstrBuffer == NULL )
                    {
                        LoadString ( hInstance ,
                               IDS_MEMORY_FAILED ,
                               szMsgBuffer , sizeof ( szMsgBuffer )
) ;
                        MessageBox ( hDlg , szMsgBuffer ,
                               szTitle , MB_ICONERROR ) ;
                    }
                    else
                    {
                        if ( GetDlgItemText ( hDlg ,
                                IDC_GET_TEXT , pstrBuffer , iLength+1 ) )
                        {
                            if ( !SaveMessages ( hDlg , pstrBuffer ) )
                            {
                                free ( pstrBuffer ) ;
                                pstrBuffer = NULL ;
                                LoadString ( hInstance ,
                                     IDS_FILE_OPEN_ERROR ,
                                     szMsgBuffer , sizeof ( szMsgBuffer )
) ;
                                MessageBox ( hDlg , szMsgBuffer ,
                                    szTitle , MB_ICONERROR ) ;
                            }
                            else   // fail to write the file
                            {
                                free ( pstrBuffer ) ;
                                pstrBuffer = NULL ;
```

```
                    }
                }
                else  // no chatting records
                {
                    free (pstrBuffer) ;
                    pstrBuffer = NULL ;
                    LoadString ( hInstance , IDS_NO_TEXT ,
                        szMsgBuffer , sizeof ( szMsgBuffer )
) ;
                    MessageBox ( hDlg , szMsgBuffer ,
                                szTitle , MB_ICONWARNING ) ;
                }
            }
        }
        break ;
    case IDC_CLOSE:
        bEnd = 1 ;
        SendMessage ( hDlg, WM_CLOSE , 0 , 0 ) ;
        break ;
    }
    return FALSE ;
case WM_CLOSE :
    CloseConnection ( ) ;
    EndDialog ( hDlg , 0 ) ;
    return FALSE ;
}
return FALSE ;
}



/***
 *message_process.c - message_process functions
 *
 *        Copyright (C) 2011, Victor Jianfei Ye. All rights reserved.
 *
 *Purpose:
 *        defines message_process functions
 *
 *****************************************************************/

#include <windows.h>
#include <string.h>
#include "common.h"
#include "resource.h"

extern volatile HWND hMainDlg ;
extern SOCKET cltSck;
extern char szTitle[102] ;
extern char *receive_text ;
```

```
/***
 *BOOL receive_msg ( const char * rec_msg ,
 *           HINSTANCE hInstance ) - the function
 *          to receive the message from the server
 *
 *Purpose:
 *        Receive the messages from the server
 *                  and send to the IDC_GET_TEXT edit box.
 *
 *Entry:
 *        const char * rec_msg    - the string received from the server
 *        HINSTANCE    hInstance  - the handle to this instance
 *
 *Exit:
 *        Success: TRUE
 *        Failure: FALSE
 *
 *Exceptions:
 *
 *******************************************************************/

BOOL receive_msg ( const char *rec_msg , HINSTANCE hInstance )
{
    long length = 0 ;
    char *temp = NULL ;
    char szSevereWarning[15] ;
    char szMemoryError[200] ;
    if ( rec_msg[0] == '\a' )
        return FALSE ;
    if ( ( length = SendMessage ( GetDlgItem ( hMainDlg ,
            IDC_GET_TEXT ) , WM_GETTEXTLENGTH , 0 , 0 ) ) > 165536 )
    {
        temp = (char *) realloc ( receive_text ,
                        ( length + 65536 ) * sizeof(char) ) ;
        if ( temp != NULL )
            receive_text = temp ;
        else
        {
            LoadString ( hInstance , IDS_SEVERE_WARNING ,
                 szSevereWarning , sizeof ( szSevereWarning ) ) ;
            LoadString ( hInstance , IDS_MEMORY_ERROR ,
                        szMemoryError , sizeof ( szMemoryError ) ) ;
            if ( MessageBox ( hMainDlg , szMemoryError ,
                        szSevereWarning ,
                    MB_YESNO|MB_ICONWARNING ) == IDYES )
            {
                strcpy ( receive_text , "" ) ;
                free ( receive_text ) ;
                receive_text = (char *) malloc ( 165536 * sizeof(char) ) ;
```

```
            }
            else
            {
                free ( receive_text ) ;
                receive_text = NULL ;
                ExitProcess ( 0 ) ;
            }
        }
    }
    GetDlgItemText ( hMainDlg , IDC_GET_TEXT , receive_text , 100000 ) ;
    strcat ( receive_text , rec_msg ) ;
    return SetDlgItemText ( hMainDlg , IDC_GET_TEXT , receive_text ) ;
}

/***
 *BOOL send_msg ( HINSTANCE hInstance ) - the function
 *                                    of sending the message to the server
 *
 *Purpose:
 *        Send the messages to the server from the
 *                    IDC_SEND_TEXT edit box.
 *
 *Entry:
 *        HINSTANCE  hInstance - the handle to this instance
 *
 *Exit:
 *        Success: TRUE
 *        Failure: FALSE
 *
 *Exceptions:
 *
 *******************************************************************************

BOOL send_msg ( HINSTANCE hInstance )
{
    char send_text [60000] , szSendingError[50] ;
    HWND hWnd = GetDlgItem ( hMainDlg , IDC_SEND_TEXT ) ;
    SetFocus ( hWnd ) ;
    GetDlgItemText ( hMainDlg , IDC_SEND_TEXT , send_text , sizeof(send_text) ) ;
    SetDlgItemText ( hMainDlg , IDC_SEND_TEXT , "" ) ;
    if ( send ( cltSck,send_text, strlen(send_text)+1 , 0 ) == SOCKET_ERROR )
    {
        LoadString ( hInstance , IDS_SENDING_ERROR ,
                    szSendingError , sizeof ( szSendingError ) ) ;
        MessageBox ( hMainDlg , szSendingError ,
                    szTitle , MB_ICONERROR ) ;
        return FALSE ;
    }
    Sleep(10) ;
```

```
    return TRUE ;
}

/***
 *int print ( const char *msg ) - the function to show a
 *                          message box from the given string
 *
 *Purpose:
 *        Show a message box from the given string.
 *
 *Entry:
 *        const char *    msg - the pointer to the given string
 *
 *Exit:
 *        Return IDOK
 *
 *Exceptions:
 *
 *******************************************************************************

int print ( const char *msg )
{
    return MessageBox ( hMainDlg , msg , szTitle , MB_ICONERROR ) ;
}



/***
*SaveMessage.h - declarations for save-message function
*
*        Copyright (C) 2011, Victor Jianfei Ye. All rights reserved.
*
*Purpose:
*        This file declare the save-message function.
*
****/

#ifndef _SAVE_MESSAGE_H
#define _SAVE_MESSAGE_H

BOOL SaveMessages ( HWND hDlg , LPCTSTR pstrBuffer ) ;

#endif



/***
*SaveMessage.h - declarations for save-message function
*
```

```
*        Copyright (C) 2011, Victor Jianfei Ye. All rights reserved.
*
*Purpose:
*        This file declare the save-message function.
*
****/

#ifndef _SAVE_MESSAGE_H
#define _SAVE_MESSAGE_H

BOOL SaveMessages ( HWND hDlg , LPCTSTR pstrBuffer ) ;

#endif
```

```
/***
 *server_ip.h - declarations for server-ip dialog callback function
 *
 *        Copyright (C) 2011, Victor Jianfei Ye. All rights reserved.
 *
 *Purpose:
 *        This file declare the server-ip dialog callback function.
 *
 ****/

#ifndef SERVER_IP_H
#define SERVER_IP_H

BOOL CALLBACK SeverIp ( HWND hDlg , UINT uMsg ,
                WPARAM wParam , LPARAM lParam ) ;

#endif
```

```
/***
 *server_ip.c - server_ip dialog function
 *
 *        Copyright (C) 2011, Victor Jianfei Ye. All rights reserved.
 *
 *Purpose:
 *        defines the server_ip dialog function
 *
 ****************************************************************/

#include <windows.h>
#include <string.h>
#include <commctrl.h>
```

```
#include "resource.h"
#include "server_ip.h"
#include "core.h"
#pragma comment ( lib , "comctl32.lib" )

HANDLE hThread = NULL ;
char IP[20] ;

extern volatile int status ;
extern volatile HWND hMainDlg ;

/***
 *BOOL CALLBACK MainDialog ( HWND hDlg , UINT uMsg ,
 *                    WPARAM wParam , LPARAM lParam )
 *                    - the callback function of the main dialog
 *
 *Purpose:
 *         Deal with the messages that send to the main dialog.
 *
 *Entry:
 *         HWND    hDlg      - the handle to the main dialog
 *         UINT    uMsg      - the type of the message
 *         WPARAM  wParam    - the value passed as a parameter
 *                             to a window procedure or
 *                             callback function
 *         int     nShowCmd  - the 32-bit value passed as
 *                             a parameter to a window
 *                             procedure or callback function
 *
 *Exit:
 *         Return FALSE
 *
 *Exceptions:
 *
 *******************************************************************************

BOOL CALLBACK SeverIp ( HWND hDlg , UINT uMsg , WPARAM wParam , LPARAM lParam )
{
    HWND hWnd ;
    static HINSTANCE hInstance = NULL ;
    char szStaticIpText[50] ;
    char szButtonText[15] ;
    char szOKCancel[10] ;
    static DWORD ThreadID = 0 ;
    HICON hIcon ;
    hMainDlg = hDlg ;


    if ( status )
        EndDialog ( hDlg , 0 ) ;
```

```
switch ( uMsg )
{




case WM_INITDIALOG :
    hInstance = (HINSTANCE)lParam ;
    hIcon = LoadIcon ( hInstance , MAKEINTRESOURCE ( IDI_ICON ) ) ;
    SendMessage ( hDlg, WM_SETICON , TRUE , (LPARAM) hIcon ) ;
    SendMessage ( hDlg, WM_SETICON , FALSE , (LPARAM) hIcon ) ;
    InitCommonControls ( ) ;
    SetDlgItemText ( hDlg , IDC_IPADDRESS , "0.0.0.0" ) ;
    return FALSE ;
case WM_COMMAND :
    switch ( LOWORD (wParam) )
    {
    case IDC_OK_CANCEL:
        GetDlgItemText ( hDlg , IDC_OK_CANCEL , szButtonText , 18 ) ;
        LoadString ( hInstance , IDS_OK , szOKCancel , 8 ) ;
        if ( !strcmp ( szButtonText , szOKCancel ) )
        {
            GetDlgItemText ( hDlg , IDC_IPADDRESS , IP , 18 ) ;
            hWnd = GetDlgItem (hDlg , IDC_IPADDRESS );
            EnableWindow ( hWnd , FALSE ) ;
            LoadString ( hInstance , IDS_IP_CONNECTION ,
                                        szStaticIpText , 40 ) ;
            SendMessage ( hDlg , WM_SETTEXT , 0 ,
                                    (LPARAM)szStaticIpText );
            strcat ( szStaticIpText , IP ) ;
            hWnd = GetDlgItem ( hDlg , IDC_STATIC_IP ) ;
            SendMessage ( hWnd , WM_SETTEXT , 0 ,
                                   (LPARAM)szStaticIpText );
            LoadString ( hInstance , IDS_CANCEL , szOKCancel , 8 ) ;
            SetDlgItemText ( hDlg , IDC_OK_CANCEL , szOKCancel ) ;
            hThread = CreateThread ( NULL, 0, connection ,
                        (LPVOID)&hInstance , 0 , &ThreadID ) ;
        }
        else
        {
            ExitProcess ( 0 ) ;
        }
        break ;
    }
    return FALSE ;
case WM_CLOSE :
    ExitProcess ( 0 ) ;
    return FALSE ;
}
```

```
    return FALSE ;
}


//{{NO_DEPENDENCIES}}
// Microsoft Developer Studio generated include file.
// Used by client.rc
//
#define IDS_TITLE                       2
#define IDS_WINDOW_CREATION_FAILURE     3
#define IDS_SEVERE_WARNING              4
#define IDS_MEMORY                      5
#define IDS_MEMORY_ERROR                5
#define IDS_SOCKET_INIT_ERROR           6
#define IDS_SOCKET_CREATION_ERROR       7
#define IDS_IP_CONNECTION               8
#define IDS_OK                          9
#define IDS_CANCEL                      10
#define IDS_NON_BLOCKING_MODE_ERROR     11
#define IDS_CONNECTION_CLOSED           12
#define IDS_SENDING_ERROR               13
#define IDS_UNKNOWN_SYSTEM_ERROR        14
#define IDS_MEMORY_FAILED               15
#define IDS_FILE_OPEN_ERROR             16
#define IDS_NO_TEXT                     17
#define IDD_MAIN_DIALOG                 101
#define IDD_CONNECTION_DIALOG           102
#define IDI_ICON                        103
#define IDD_SEVER_IP                    104
#define IDC_GET_TEXT                    1000
#define IDC_SEND_TEXT                   1001
#define IDC_SEND                        1002
#define IDC_CLOSE                       1003
#define IDC_IPADDRESS                   1008
#define IDC_OK                          1014
#define IDC_OK_CANCEL                   1014
#define IDC_STATIC_IP                   1015
#define IDC_CLEAR                       1017
#define IDC_SAVE                        1018

// Next default values for new objects
//
#ifdef APSTUDIO_INVOKED
#ifndef APSTUDIO_READONLY_SYMBOLS
#define _APS_NO_MFC                     1
#define _APS_NEXT_RESOURCE_VALUE        106
#define _APS_NEXT_COMMAND_VALUE         40001
#define _APS_NEXT_CONTROL_VALUE         1019
#define _APS_NEXT_SYMED_VALUE           101
```

```
#endif
#endif
```

1.5.5 客户端程序资源文件代码

```
//Microsoft Developer Studio generated resource script.
//
#include "resource.h"

#define APSTUDIO_READONLY_SYMBOLS
//////////////////////////////////////////////////////////////////////////////
//
// Generated from the TEXTINCLUDE 2 resource.
//
#include "afxres.h"

//////////////////////////////////////////////////////////////////////////////
#undef APSTUDIO_READONLY_SYMBOLS

//////////////////////////////////////////////////////////////////////////////
// Chinese (P.R.C.) resources

#if !defined(AFX_RESOURCE_DLL) || defined(AFX_TARG_CHS)
#ifdef _WIN32
LANGUAGE LANG_CHINESE, SUBLANG_CHINESE_SIMPLIFIED
#pragma code_page(936)
#endif //_WIN32

//////////////////////////////////////////////////////////////////////////////
//
// Dialog
//

IDD_MAIN_DIALOG DIALOG DISCARDABLE  120, 20, 271, 251
STYLE DS_MODALFRAME | WS_MINIMIZEBOX | WS_POPUP | WS_CAPTION | WS_SYSMENU
CAPTION "聊天工具（客户端）"
FONT 12, "宋体"
BEGIN
    EDITTEXT        IDC_GET_TEXT,7,7,257,147,ES_MULTILINE | ES_AUTOVSCROLL |
                    ES_AUTOHSCROLL | ES_READONLY | ES_WANTRETURN |
                    WS_VSCROLL | WS_HSCROLL
    EDITTEXT        IDC_SEND_TEXT,7,156,257,71,ES_MULTILINE | ES_AUTOHSCROLL |
                    ES_WANTRETURN | WS_VSCROLL | WS_HSCROLL
    PUSHBUTTON      "发送",IDC_SEND,225,230,39,14
    PUSHBUTTON      "关闭",IDC_CLOSE,176,230,40,14
    PUSHBUTTON      "清空聊天记录",IDC_CLEAR,7,230,50,14
```

```
    PUSHBUTTON      "保存聊天记录",IDC_SAVE,68,230,50,14
END

IDD_SEVER_IP DIALOG DISCARDABLE  150, 100, 203, 112
STYLE DS_MODALFRAME | WS_MINIMIZEBOX | WS_POPUP | WS_CAPTION | WS_SYSMENU
CAPTION "填写服务器地址IP"
FONT 12, "宋体"
BEGIN
    DEFPUSHBUTTON   "确定",IDC_OK_CANCEL,76,80,50,14
    CTEXT           "请输入服务器的␣IP␣地址",IDC_STATIC_IP,7,28,189,8,
                    SS_CENTERIMAGE
    CONTROL         "IPAddress3",IDC_IPADDRESS,"SysIPAddress32",WS_TABSTOP,
                    53,51,100,15
END


/////////////////////////////////////////////////////////////////////////////
//
// DESIGNINFO
//

#ifdef APSTUDIO_INVOKED
GUIDELINES DESIGNINFO DISCARDABLE
BEGIN
    IDD_MAIN_DIALOG, DIALOG
    BEGIN
        LEFTMARGIN, 7
        RIGHTMARGIN, 264
        TOPMARGIN, 7
        BOTTOMMARGIN, 244
    END

    IDD_SEVER_IP, DIALOG
    BEGIN
        LEFTMARGIN, 7
        RIGHTMARGIN, 196
        TOPMARGIN, 7
        BOTTOMMARGIN, 105
    END
END
#endif    // APSTUDIO_INVOKED


#ifdef APSTUDIO_INVOKED
/////////////////////////////////////////////////////////////////////////////
//
// TEXTINCLUDE
//

1 TEXTINCLUDE DISCARDABLE
```

```
BEGIN
    "resource.h\0"
END

2 TEXTINCLUDE DISCARDABLE
BEGIN
    "#include␣""afxres.h""\r\n"
    "\0"
END

3 TEXTINCLUDE DISCARDABLE
BEGIN
    "\r\n"
    "\0"
END

#endif    // APSTUDIO_INVOKED


#ifndef _MAC
/////////////////////////////////////////////////////////////////////////////
//
// Version
//

VS_VERSION_INFO VERSIONINFO
 FILEVERSION 1,0,0,1
 PRODUCTVERSION 1,0,0,1
 FILEFLAGSMASK 0x3fL
#ifdef _DEBUG
 FILEFLAGS 0x29L
#else
 FILEFLAGS 0x28L
#endif
 FILEOS 0x40004L
 FILETYPE 0x1L
 FILESUBTYPE 0x0L
BEGIN
    BLOCK "StringFileInfo"
    BEGIN
        BLOCK "080404b0"
        BEGIN
            VALUE "Comments", "内核设计——周开锋WinSock␣\r\n\r\n␣图
形用户界面（␣GUI␣）设计——叶剑飞\0"
            VALUE "CompanyName", "周开锋、叶剑飞\0"
            VALUE "FileDescription", "这是一个聊天软件客户端\0"
            VALUE "FileVersion", "1,␣0,␣0,␣1\0"
            VALUE "InternalName", "client\0"
            VALUE "LegalCopyright", "版权所有  (␣C␣)␣␣周开锋、
叶剑飞  2011\0"
```

```
            VALUE "LegalTrademarks", "周开锋、叶剑飞\0"
            VALUE "OriginalFilename", "client.exe\0"
            VALUE "PrivateBuild", "叶剑飞\0"
            VALUE "ProductName", "聊天工具（客户端）    应用程
序\0"
            VALUE "ProductVersion", "1, 0, 0, 1\0"
            VALUE "SpecialBuild", "叶剑飞\0"
        END
    END
    BLOCK "VarFileInfo"
    BEGIN
        VALUE "Translation", 0x804, 1200
    END
END

#endif    // !_MAC


/////////////////////////////////////////////////////////////////////////////
//
// Icon
//

// Icon with lowest ID value placed first to ensure application icon
// remains consistent on all systems.
IDI_ICON                ICON    DISCARDABLE     "Victor.ico"

/////////////////////////////////////////////////////////////////////////////
//
// String Table
//

STRINGTABLE DISCARDABLE
BEGIN
    IDS_TITLE                   "聊天程序（客户端）"
    IDS_WINDOW_CREATION_FAILURE "应用程序发生异常，窗口创建失
败！"
    IDS_SEVERE_WARNING          "严重警告"
    IDS_MEMORY_ERROR            "内存已耗尽，请确认是否清空消息记
录？ \r\n\r\n警告：若不清空消息记录，则结束进程！"
    IDS_SOCKET_INIT_ERROR       "Socket 初始化失败！"
    IDS_SOCKET_CREATION_ERROR   "创建 Socket 失败！"
    IDS_IP_CONNECTION           "正在建立连接中    ..."
    IDS_OK                      "确定"
    IDS_CANCEL                  "取消"
    IDS_NON_BLOCKING_MODE_ERROR "设置非阻塞模式失败"
    IDS_CONNECTION_CLOSED       "服务器已中断连接"
    IDS_SENDING_ERROR           "消息发送失败！"
    IDS_UNKNOWN_SYSTEM_ERROR    "系统发生未知错误！"
    IDS_MEMORY_FAILED           "警告：内存不足！操作失败！"
```

```
END

STRINGTABLE  DISCARDABLE
BEGIN
    IDS_FILE_OPEN_ERROR     ”无法访问此文件。保存失败！”
    IDS_NO_TEXT             ”聊天记录为空！”
END

#endif    // Chinese (P.R.C.) resources
//////////////////////////////////////////////////////////////////////////////



#ifndef APSTUDIO_INVOKED
//////////////////////////////////////////////////////////////////////////////
//
// Generated from the TEXTINCLUDE 3 resource.
//


//////////////////////////////////////////////////////////////////////////////
#endif    // not APSTUDIO_INVOKED
```

2　基于 IP 多播的网络会议程序　　Win32 版

2.1　简介

本网络会议软件使用的是 IP 多播发送会议信息，多播地址为 233.0.0.1，占用 8000 端口，字符编码采用的是 UTF-16。

2.2　使用说明

该网络会议程序支持 Windows XP 及以上版本的 Windows 操作系统。不支持 Windows 98/2000 或更低版本的操作系统，也不支持类 Unix 操作系统。该网络会议软件仅支持 IPv4 地址, 不支持 IPv6 地址。

双击可执行文件，出现主界面，即可加入网络会议。界面如下：

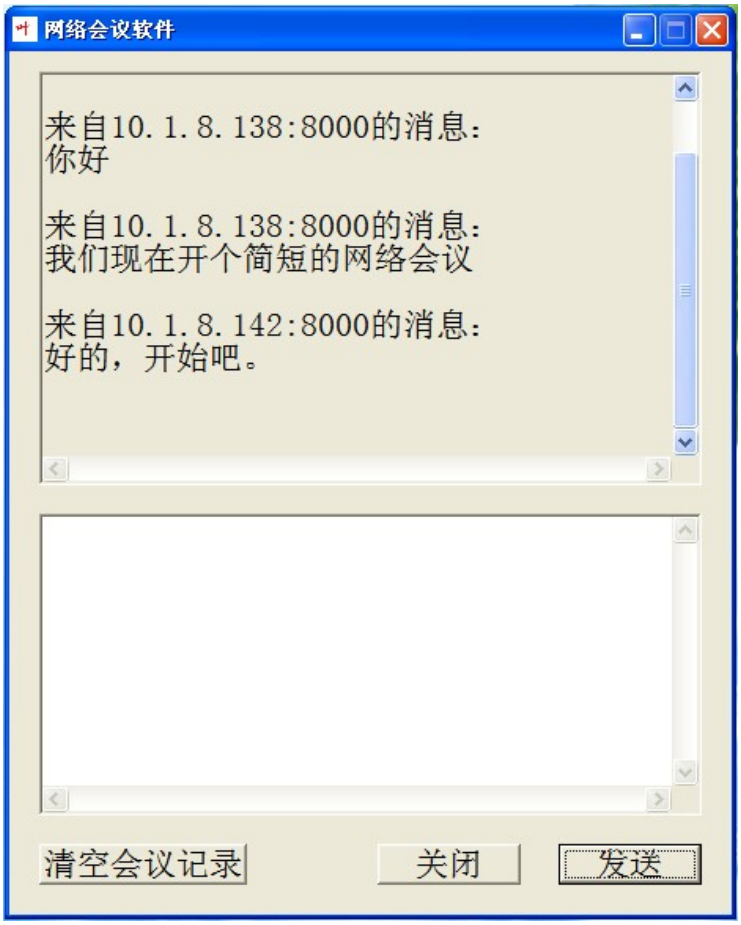


图 6: 网络会议程序主界面

在下方文本框中键入文字，单击发送即可发送文本。上方文本框用于显示接收到的文字信息。

2.3 技术细节

2.3.1 工作原理

WinSock 编程，必然要先调用 WSAStartup 函数来载入 WinSock 库。接下来创建一个 SOCK_DGRAM 类型的套接字。将此 socket 绑定在本地的一个端口上（我用的是 8000 端口），以便接收多播数据，然后调用 WSAJoinLeaf 函数加入多播组即可。然后就可以用 sendto 来发送多播数据，用 recvfrom 来接收多播数据。退出时调用 closesocket 关闭套接字。由于是 WinSock 编程，所以还要调用 WSACleanup 函数移出 WinSock 库。

2.3.2 开发过程

我先是照着我们的课程设计指导书分别敲上了发送方控制台和接受方的控制台程序，不过调试起来可不容易，因为发送方和接受方用的是同一个端口号，不能在同一台计算机上运行。当时我就远程桌面到另两台来调试。很快就调试成功了，于是我立刻就把发送方的代码整合到接受方代码中，成为收发两用的 IP 多播程序。不过这种单线程的有个严重问题——如果没有人发信息，那么 recvfrom 将阻塞，导致后面的输入和发送的代码没有机会运行；如果运行到 gets 却不输入，那么也会阻塞在那里，导致下一轮的 recvfrom 无法运行，从而不能收到多播数据。这样互相阻塞对方不是个办法，有必要改成多线程——一个线程专门用 gets 等待输入并用 sendto 发送多播数据，另一个线程专门用 recvfrom 接收多播数据，并用 puts 输出来。很快我就该好了，调试也通过了。终于到了添加图形用户界面的时候了。在 Visual Studio 中添加一个资源文件，放一个对话框，然后用 DialogBoxParam 启动这个对话框，并做一些消息处理，分别进行收发，图形用户界面版本的很快就完工。

2.4 程序源代码

2.4.1 C++ 源文件 main.cpp 代码

```
#include <tchar.h>
#include <process.h>
#include <WinSock2.h>
#include <WS2tcpip.h>
#include <stdio.h>
#include <stdlib.h>
#include <locale.h>
#include "resource.h"

#define MCASTADDR "233.0.0.1" // multacast group address
#define MCASTPORT  8000 // the local port that binds
#define BUFSIZE 1024  // the size of the buffer
#define MAX_RECEIVE_BYTES 8000000

bool running = false;
struct sockaddr_in remote, from;
SOCKET sock, sockMulticast;

int MulticastText ( HWND hSendText )
{
```

```
    LPTSTR sendbuf = NULL;
    TCHAR szShowText[BUFSIZE];
    int len = ::SendMessage( hSendText, WM_GETTEXTLENGTH, 0, 0 );
    sendbuf = (LPTSTR)malloc( (1+len) * sizeof(TCHAR) );

    if ( sendbuf == NULL )
    {
        ::closesocket( sockMulticast );
        ::MessageBox( hSendText, TEXT("Malloc Failed!!!!!"),
                  TEXT("Error"), MB_ICONERROR );
        ::ExitProcess(EXIT_FAILURE);
    }

    try
    {
        ::GetWindowText( hSendText, sendbuf, len+1 );
        if ( sendto( sockMulticast, (const char *)sendbuf,
               _tcslen(sendbuf) * sizeof(TCHAR), 0,
               (struct sockaddr *)&remote, sizeof(remote))
                 == SOCKET_ERROR )
        {
            _stprintf( szShowText, TEXT("sendto failed with: %d\n"),
                       WSAGetLastError() );
            ::free( sendbuf );
            ::closesocket( sockMulticast );
            running = false;
            ::MessageBox( hSendText, szShowText,
                    TEXT("Error"), MB_ICONERROR );
            return EXIT_FAILURE;
        }
        ::free( sendbuf );
        ::SetWindowText( hSendText, TEXT("") );
        return EXIT_SUCCESS;
    }
    catch( ... )
    {
        ::free( sendbuf );
        closesocket( sockMulticast );
        running = false;
        return EXIT_FAILURE;
    }
}

unsigned __stdcall receiver ( void * args )
{
    int n;
    int sockaddr_len = sizeof( struct sockaddr_in );
    static TCHAR recvbuf[MAX_RECEIVE_BYTES];
    LPTSTR lpszShowText = NULL;
    TCHAR szShowString[BUFSIZE];
```

```
HWND hReceiveText = (HWND)args;
int strlength;

try
{
    while ( running )
    {
        if (( n = recvfrom(sock, (char *)recvbuf,
             sizeof(recvbuf), 0, (struct sockaddr *)&from,
             &sockaddr_len)) == SOCKET_ERROR )
        {
            _stprintf( szShowString,
                  TEXT("recvfrom failed with: %d\n"),
                  WSAGetLastError() );
            closesocket(sock);
            running = false;
            ::MessageBox( hReceiveText, szShowString,
                  TEXT("Error"), MB_ICONERROR );
            return EXIT_FAILURE;
        }
        recvbuf[n/sizeof(TCHAR)] = TEXT('\0');
        if ( _tcscmp( recvbuf, TEXT("QUIT") ) == 0 )
        {
            running = false;
            return EXIT_SUCCESS;
        }
        else
        {
            recvbuf[n] = TEXT('\0');
            TCHAR szAddressString[100];
            u_long dwAddressStringLength = 20;
            WSAAddressToString( (struct sockaddr *)&from,
                sizeof(from), NULL,
                szAddressString, &dwAddressStringLength );

            strlength = 100 + n + ::SendMessage(
                 hReceiveText, WM_GETTEXTLENGTH, 0, 0);
            lpszShowText = (LPTSTR)malloc( strlength * sizeof(TCHAR));
            if ( lpszShowText == NULL )
            {
                running = false;
                ::closesocket( sock );
                ::ExitProcess(EXIT_FAILURE);
            }

            ::GetWindowText( hReceiveText, lpszShowText, strlength );
            _stprintf( lpszShowText,
                  TEXT("%s 来自%s 的消息: \r\n%s\r\n\r\n"), lpszShowText,
                szAddressString, recvbuf);
            ::SetWindowText(hReceiveText, lpszShowText );
```

```
                free(lpszShowText);
                ::SendMessage( hReceiveText, WM_VSCROLL,
                    MAKEWPARAM( SB_BOTTOM, 0 ), (LPARAM)NULL );
            }
        }
        return EXIT_SUCCESS;
    }
    catch(...)
    {
        closesocket(sock);
        running = false;
        return EXIT_FAILURE;
    }
}

BOOL CALLBACK ChatDlgProc( HWND hDlg, UINT uMsg,
                WPARAM wParam, LPARAM lParam )
{
    static HICON hIcon = NULL;
    static HANDLE hThreadReceiver = NULL;
    static HWND hSendText = NULL;
    static HWND hReceiveText = NULL;

    switch ( uMsg )
    {
    case WM_INITDIALOG:
        hIcon = ::LoadIcon( ::GetModuleHandle(NULL),
                MAKEINTRESOURCE(IDI_ICON1) );
        ::SendMessage( hDlg, WM_SETICON, ICON_BIG,
                (LPARAM)hIcon );
        ::SendMessage( hDlg, WM_SETICON, ICON_SMALL,
                (LPARAM)hIcon );
        hSendText = ::GetDlgItem( hDlg, IDC_SENDTEXT );
        hReceiveText = ::GetDlgItem( hDlg, IDC_RECVTEXT );
        hThreadReceiver = (HANDLE)::_beginthreadex(
                NULL, 0, receiver, hReceiveText, 0, NULL );
        return TRUE;
    case WM_COMMAND:
        switch( LOWORD(wParam) )
        {
        case IDC_CLEAR:
            ::SetWindowText( hReceiveText, TEXT("") );
            return TRUE;
        case IDC_SEND:
            ::MulticastText( hSendText );
            return TRUE;
        case IDC_CLOSE:
            ::EndDialog( hDlg, EXIT_SUCCESS );
            running = false;
            ::TerminateThread(
```

```
                    hThreadReceiver, EXIT_SUCCESS );
                ::CloseHandle( hThreadReceiver );
                ::DestroyIcon( hIcon );
                return TRUE;
            }
            return FALSE;
        case WM_CLOSE:
            ::EndDialog( hDlg, EXIT_SUCCESS );
            running = false;
            ::TerminateThread( hThreadReceiver, EXIT_SUCCESS );
            ::CloseHandle( hThreadReceiver );
            ::DestroyIcon( hIcon );
            return TRUE;
        }
        return FALSE;
}

int WINAPI _tWinMain( HINSTANCE hInstance,
            HINSTANCE hPrevInstance, LPTSTR lpCmdLine,
            int nCmdShow )
{
    WSADATA wsd;
    struct sockaddr_in local;
    TCHAR szError[BUFSIZE];

    setlocale(LC_CTYPE, "");

    // initialize WinSock 2.2
    if ( WSAStartup( MAKEWORD(2,2), &wsd ) != 0 )
    {
        _tprintf( TEXT("WSAStartup failed\n") );
        return EXIT_FAILURE;
    }
    if ((sock = WSASocket(AF_INET, SOCK_DGRAM, 0,
        NULL, 0, WSA_FLAG_MULTIPOINT_C_LEAF
        | WSA_FLAG_MULTIPOINT_D_LEAF |
        WSA_FLAG_OVERLAPPED)) == INVALID_SOCKET )
    {
        _stprintf( szError, TEXT("socket failed with: %d\n"),
                                    WSAGetLastError() );
        WSACleanup();
        ::MessageBox( HWND_DESKTOP, szError,
                                TEXT("Error"), MB_ICONERROR );
        return EXIT_FAILURE;
    }
    // bind a local port with a socket.
    local.sin_family = AF_INET;
    local.sin_port = htons(MCASTPORT);
    local.sin_addr.s_addr = INADDR_ANY;
    if ( bind(sock, (struct sockaddr *)&local,
```

```
            sizeof(local)) == SOCKET_ERROR )
{
    int err = WSAGetLastError();
    if ( err == 10048 )
    {
        _stprintf( szError,
           TEXT( "IP端口绑定错误，" )
           TEXT( "您不能在一台电脑上同时多次运行该软件。" ),
           WSAGetLastError() );
    }
    else
        _stprintf( szError, TEXT("bind failed with: %d" ), err );
    closesocket(sock);
    WSACleanup();
    ::MessageBox( HWND_DESKTOP, szError, TEXT("Error"), MB_ICONERROR );
    return EXIT_FAILURE;
}
// join the multicast group
remote.sin_family = AF_INET;
remote.sin_port = htons(MCASTPORT);
remote.sin_addr.s_addr = inet_addr(MCASTADDR);

if ((sockMulticast = WSAJoinLeaf(sock, (struct sockaddr *)&remote,
    sizeof(remote), NULL, NULL, NULL, NULL, JL_BOTH) ) == INVALID_SOCKET )
{
    _stprintf( szError, TEXT("WSAJoinLeaf() failed: %d\n"),
            WSAGetLastError() );
    closesocket(sock);
    WSACleanup();
    ::MessageBox( HWND_DESKTOP, szError, TEXT("Error"), MB_ICONERROR );
    return EXIT_FAILURE;
}

running = true;

::DialogBoxParam( hInstance, MAKEINTRESOURCE(IDD_DIALOG_CHAT),
      HWND_DESKTOP, ::ChatDlgProc, 0L );

try
{
    closesocket(sockMulticast);
}
catch(...) { }

try
{
    closesocket(sock);
}
catch(...) { }
```

```
    WSACleanup();

    return EXIT_SUCCESS;
}
```

2.4.2 resource.h 代码

```
//{{NO_DEPENDENCIES}}
// Microsoft Visual C++ generated include file.
// Used by resource.rc
//
#define IDD_DIALOG_CHAT                 101
#define IDI_ICON1                       102
#define IDC_RECVTEXT                    1001
#define IDC_SEND                        1002
#define IDC_CLOSE                       1003
#define IDC_SENDTEXT                    1004
#define IDC_BUTTON3                     1005
#define IDC_CLEAR                       1005

// Next default values for new objects
//
#ifdef APSTUDIO_INVOKED
#ifndef APSTUDIO_READONLY_SYMBOLS
#define _APS_NEXT_RESOURCE_VALUE        103
#define _APS_NEXT_COMMAND_VALUE         40001
#define _APS_NEXT_CONTROL_VALUE         1006
#define _APS_NEXT_SYMED_VALUE           101
#endif
#endif
```

2.4.3 资源文件代码

```
// Microsoft Visual C++ generated resource script.
//
#include "resource.h"

#define APSTUDIO_READONLY_SYMBOLS
//////////////////////////////////////////////////////////////////////////////
//
// Generated from the TEXTINCLUDE 2 resource.
//
#include "afxres.h"

//////////////////////////////////////////////////////////////////////////////
#undef APSTUDIO_READONLY_SYMBOLS

//////////////////////////////////////////////////////////////////////////////
// Chinese (P.R.C.) resources

#if !defined(AFX_RESOURCE_DLL) || defined(AFX_TARG_CHS)
```

```
LANGUAGE LANG_CHINESE, SUBLANG_CHINESE_SIMPLIFIED

#ifdef APSTUDIO_INVOKED
//////////////////////////////////////////////////////////////////////////////
//
// TEXTINCLUDE
//

1 TEXTINCLUDE
BEGIN
    "resource.h\0"
END

2 TEXTINCLUDE
BEGIN
    "#include␣""afxres.h""\r\n"
    "\0"
END

3 TEXTINCLUDE
BEGIN
    "\r\n"
    "\0"
END

#endif    // APSTUDIO_INVOKED


//////////////////////////////////////////////////////////////////////////////
//
// Dialog
//

IDD_DIALOG_CHAT DIALOGEX 0, 0, 167, 209
STYLE DS_SETFONT | DS_MODALFRAME | DS_CENTER | WS_MINIMIZEBOX | WS_POPUP | WS_C
CAPTION "网络会议软件"
FONT 16, "宋体", 400, 0, 0x0
BEGIN
    EDITTEXT        IDC_RECVTEXT,7,5,153,100,ES_MULTILINE | ES_AUTOVSCROLL | ES
    PUSHBUTTON      "发送",IDC_SEND,127,192,33,10
    PUSHBUTTON      "关闭",IDC_CLOSE,85,192,33,10
    EDITTEXT        IDC_SENDTEXT,7,112,153,73,ES_MULTILINE | ES_AUTOVSCROLL | E
    PUSHBUTTON      "清空会议记录",IDC_CLEAR,7,192,48,10
END


//////////////////////////////////////////////////////////////////////////////
//
// DESIGNINFO
//
```

```
#ifdef APSTUDIO_INVOKED
GUIDELINES DESIGNINFO
BEGIN
    IDD_DIALOG_CHAT, DIALOG
    BEGIN
        LEFTMARGIN, 7
        RIGHTMARGIN, 160
        TOPMARGIN, 7
        BOTTOMMARGIN, 202
    END
END
#endif    // APSTUDIO_INVOKED


//////////////////////////////////////////////////////////////////////////////
//
// Icon
//

// Icon with lowest ID value placed first to ensure application icon
// remains consistent on all systems.
IDI_ICON1               ICON                    "victor.ico"

//////////////////////////////////////////////////////////////////////////////
//
// Version
//

VS_VERSION_INFO VERSIONINFO
 FILEVERSION 1,0,0,1
 PRODUCTVERSION 1,0,0,1
 FILEFLAGSMASK 0x3fL
#ifdef _DEBUG
 FILEFLAGS 0x1L
#else
 FILEFLAGS 0x0L
#endif
 FILEOS 0x40004L
 FILETYPE 0x1L
 FILESUBTYPE 0x0L
BEGIN
    BLOCK "StringFileInfo"
    BEGIN
        BLOCK "080404b0"
        BEGIN
            VALUE "CompanyName", "叶剑飞"
            VALUE "FileDescription", "多播网络会议软件"
            VALUE "FileVersion", "1.0.0.1"
            VALUE "InternalName", "ipmultic.exe"
```

```
            VALUE "LegalCopyright", "版权所有 © 叶剑飞 2013"
            VALUE "OriginalFilename", "ipmultic.exe"
            VALUE "ProductName", "多播网络会议软件"
            VALUE "ProductVersion", "1.0.0.1"
        END
    END
    BLOCK "VarFileInfo"
    BEGIN
        VALUE "Translation", 0x804, 1200
    END
END

#endif    // Chinese (P.R.C.) resources
/////////////////////////////////////////////////////////////////////////////



#ifndef APSTUDIO_INVOKED
/////////////////////////////////////////////////////////////////////////////
//
// Generated from the TEXTINCLUDE 3 resource.
//


/////////////////////////////////////////////////////////////////////////////
#endif    // not APSTUDIO_INVOKED
```

3 基于 UDP 广播的网络会议程序　　跨平台版

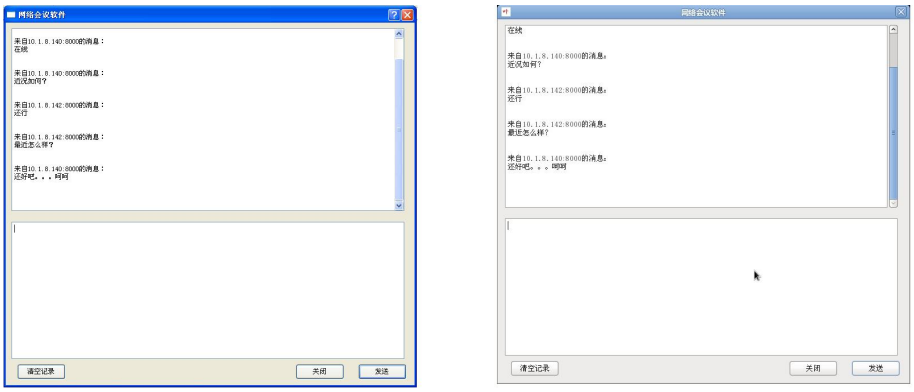
3.1 简介

本网络会议软件使用的是 UDP 广播发送会议信息，占用 8000 端口，字符编码采用的是 UTF-8。该软件用完全采用 Qt 库来写，因此具有良好的跨平台能力。

3.2 使用说明

该网络会议程序支持带 Qt4 运行库的 Linux 操作系统、Mac OS X 操作系统和 Windows XP 及以上版本的 Windows 操作系统。该网络会议程序仅支持 IPv4 地址, 不支持 IPv6 地址。

双击可执行文件，出现主界面，即可加入网络会议。界面如下：



Windows 版　　**Linux版**

图 7: 网络会议软件主界面

在下方文本框中键入文字，单击发送或者按下 Ctrl + Enter 快捷键，即可发送文本。上方文本框用于显示接收到的文字信息。

3.3 技术细节

整个敲码过程完全在 Linux 操作系统上完成。Qt 网络库封装了一个 QUdpSocket 类。这个类的构造函数的内部就是初始化 UDP 套接字，析构函数就是关闭套接字。用这个类的成员函数 bind，即可绑定本地端口。然后即可调用 QUdpSocket 的成员函数 readDatagram 和 writeDatagram 来收发广播数据。至于 Ctrl+Enter 发送，可以用 Qt 对话框封装类 QDialog 的成员函数 eventFilter 来捕获 Ctrl 键和 Enter 键。

3.4 程序源代码

3.4.1 cmulticast.h 头文件代码

```
#ifndef CMULTICAST_H
#define CMULTICAST_H
```

```
#include <QtNetwork/QUdpSocket>
class Dialog;
class QPlainTextEdit;

class CMulticast : public QUdpSocket
{
    Q_OBJECT
public:
    CMulticast( Dialog * parent = 0 );
public slots:
    void RecvDiagram( );
    void SendDiagram( const QString & txtSend );
private:
    Dialog * parent;
};

#endif // CMULTICAST_H
```

3.4.2 dialog.h 头文件代码

```
#ifndef DIALOG_H
#define DIALOG_H

#include <QtGui/QDialog>
class CMulticast;
class QPlainTextEdit;
class QPushButton;

namespace Ui {
    class Dialog;
}

class Dialog : public QDialog
{
    Q_OBJECT

public:
    explicit Dialog(QWidget *parent = 0);
    ~Dialog();
    QPlainTextEdit * getTxtRecv();

protected:
    bool eventFilter(QObject * obj, QEvent * e );

private slots:
    void on_btnClose_clicked();

    void on_btnSend_clicked();
```

```
    void on_btnClear_clicked();

private:
    Ui::Dialog *ui;
    CMulticast * multicast;
};

#endif // DIALOG_H
```

3.4.3 cmulticast.cpp 源文件代码

```
#include "cmulticast.h"
#include "dialog.h"
#include <QtGui/QPlainTextEdit>
#include <QtCore/QByteArray>
#include <QtCore/QString>
#include <QtGui/QScrollBar>

#define      MULTICAST_PORT_NUM       8000

CMulticast::CMulticast( Dialog * parent )
    : QUdpSocket( parent ), parent( parent )
{
    bind( MULTICAST_PORT_NUM, QUdpSocket::ShareAddress );
    connect( this, SIGNAL(readyRead()), this, SLOT(RecvDiagram()) );
}

void CMulticast::RecvDiagram( )
{
    QString textRecved;
    QByteArray datagram;
    QHostAddress hostAddress;
    quint16 port;

    while ( hasPendingDatagrams() )
    {
        datagram.resize( pendingDatagramSize());
        readDatagram(datagram.data(), datagram.size(),
                 &hostAddress, &port);
        textRecved = QString("来自的消息：%1:%2\n%3\n\n\n")
                 .arg(hostAddress.toString()).arg(port)
                 .arg(QString(datagram));
        parent->getTxtRecv()->appendPlainText(textRecved);
        parent->getTxtRecv()->verticalScrollBar()
                 ->setValue(parent->getTxtRecv()
                 ->verticalScrollBar()->maximumHeight());
    }
}

void CMulticast::SendDiagram( const QString & txtSend )
```

```
{
    QByteArray byteArray = txtSend.toAscii();
    writeDatagram( byteArray.data(), byteArray.size(),
            QHostAddress::Broadcast, MULTICAST_PORT_NUM );
}
```

3.4.4 dialog.cpp 源文件代码

```
#include "dialog.h"
#include "ui_dialog.h"
#include "cmulticast.h"

Dialog::Dialog(QWidget *parent) :
    QDialog(parent),
    ui(new Ui::Dialog), multicast( new CMulticast(this) )
{
    ui->setupUi(this);
    installEventFilter(this);
}

Dialog::~Dialog()
{
    delete ui;
}

bool Dialog::eventFilter(QObject * obj, QEvent * e )
{
    if (e->type() == QEvent::KeyPress)
    {
        QKeyEvent *event = static_cast<QKeyEvent*>(e);
        if (event->key() == Qt::Key_Return && (event->modifiers() & Qt::Control
        {
            emit on_btnSend_clicked();
            return true;
        }
    }
    return false;
}

QPlainTextEdit * Dialog::getTxtRecv()
{
    return ui->txtRecv;
}

void Dialog::on_btnClose_clicked()
{
    emit close();
}

void Dialog::on_btnSend_clicked()
```

```
{
    multicast->SendDiagram(ui->txtSend->toPlainText());
    ui->txtSend->clear();
    ui->txtSend->setFocus();
}

void Dialog::on_btnClear_clicked()
{
    ui->txtRecv->clear();
}
```

3.4.5 main.cpp 源文件代码

```
#include <QtGui/QApplication>
#include <QtCore/QTextCodec>
#include "dialog.h"

int main(int argc, char *argv[])
{
    QApplication a(argc, argv);
    QTextCodec * codec = QTextCodec::codecForName("UTF-8");
    QTextCodec::setCodecForLocale(codec);
    QTextCodec::setCodecForTr(codec);
    QTextCodec::setCodecForCStrings(codec);

    Dialog w;
    w.show();

    return a.exec();
}
```

3.4.6 dialog.ui 界面文件代码

```
<?xml version="1.0" encoding="UTF-8"?>
<ui version="4.0">
 <class>Dialog</class>
 <widget class="QDialog" name="Dialog">
  <property name="geometry">
   <rect>
    <x>0</x>
    <y>0</y>
    <width>719</width>
    <height>637</height>
   </rect>
  </property>
  <property name="windowTitle">
   <string>网络会议软件</string>
  </property>
  <property name="windowIcon">
   <iconset resource="resource.qrc">
    <normaloff>:/icon/victor.ico</normaloff>:/icon/victor.ico</iconset>
```

```
</property>
<widget class="QPlainTextEdit" name="txtSend">
 <property name="geometry">
  <rect>
   <x>10</x>
   <y>350</y>
   <width>691</width>
   <height>241</height>
  </rect>
 </property>
</widget>
<widget class="QPushButton" name="btnSend">
 <property name="geometry">
  <rect>
   <x>620</x>
   <y>600</y>
   <width>85</width>
   <height>27</height>
  </rect>
 </property>
 <property name="text">
  <string>发送</string>
 </property>
</widget>
<widget class="QPushButton" name="btnClose">
 <property name="geometry">
  <rect>
   <x>510</x>
   <y>600</y>
   <width>85</width>
   <height>27</height>
  </rect>
 </property>
 <property name="text">
  <string>关闭</string>
 </property>
</widget>
<widget class="QPlainTextEdit" name="txtRecv">
 <property name="geometry">
  <rect>
   <x>10</x>
   <y>10</y>
   <width>691</width>
   <height>321</height>
  </rect>
 </property>
 <property name="readOnly">
  <bool>true</bool>
 </property>
</widget>
```

```
  <widget class="QPushButton" name="btnClear">
   <property name="geometry">
    <rect>
     <x>20</x>
     <y>600</y>
     <width>85</width>
     <height>27</height>
    </rect>
   </property>
   <property name="text">
    <string>清空记录</string>
   </property>
  </widget>
 </widget>
 <layoutdefault spacing="6" margin="11"/>
 <resources>
  <include location="resource.qrc"/>
 </resources>
 <connections/>
</ui>
```

3.4.7 Qt 资源文件 resource.qrc 代码

```
<RCC>
    <qresource prefix="/icon">
        <file>victor.ico</file>
    </qresource>
</RCC>
```

3.4.8 微软资源文件 resource.rc 代码

```
IDI_ICON1           ICON        DISCARDABLE "victor.ico"
```

4 网络代理服务器　　跨平台版

4.1 简介

该网络代理服务器完全跨平台，可运行于 Windows、Linux 和 Mac OS X 操作系统，可代理任何支持代理服务器的浏览器，例如 Google Chrome、Mozilla Firefox、Internet Explorer 等等。该代理服务器只支持 HTTP 协议。

4.2 使用说明

先运行这个程序代理服务器程序，然后打开浏览器，设置代理服务器，IP 设置为代理服务器所在计算机的 IP，端口号设置为 8080。即可通过本代理服务器访问网站。该代理服务器只支持 HTTP 协议，不支持 HTTPS 协议，以及其它协议（例如 FTP 等）。该代理服务器并不支持二级代理，也不支持 SSL 加密代理。所以不要用于访问安全性需要较高安全性的网站。

4.3 技术细节

4.3.1 工作原理

先创建服务器端套接字，接受客户端（即浏览器）的连接，然后 recv 浏览器的 HTTP 请求报文，解析出其中的域名，调用 gethostbyname 来把域名解析成 IP 地址。若域名解析失败，则给客户端（即浏览器）发送 HTTP/1.0 502 Bad Gateway 的响应报文。如果域名解析成功，那酒建立客户端套接字，建立与网页服务器建立连接。连接成功后，即可转发浏览器发来的 HTTP 请求报文，并接收网页服务器发来的 HTTP 响应报文。然后把响应报文转发给浏览器。最后调用 shutdown 函数关闭 TCP 连接。然后调用 closesocket(WinSock)/close(Unix Socket) 函数关闭套接字。如果网络通信发生异常，则一律发送 HTTP/1.0 400 Bad Request 的响应报文。

本代理服务器程序软件未使用跨平台函数库，而是用预处理宏来进行条件编译。Windows 操作系统上一般预定义了 _WIN32 宏，而类 Unix 操作系统上则没有定义该宏。因此 #ifdef-#else-#endif 系列条件编译就够了。编译指定系统上的 socket API 。从而达到源代码跨平台的目的。

4.3.2 开发调试过程

当时本程序是在 Windows 平台上用 VS2010 开发，不过一开始就注意了预处理条件编译。当时用了 inet_ntop 函数，然而运行时却“无法在 ws2_32.dll 找到 inet_ntop 的函数入口点”。也就是说 VS2012 的 ws2_32.lib 静态库中有这个函数（编译链接通过了），但是 Windows XP 的 ws2_32.dll 动态库却没有这个函数，导致运行时函数调用错误。最后，我改成了一个一个字节地手动赋值。WinSock 的 struct sockaddr_in 内部是个联合体，定义了每个字节。然而 Unix Socket 却没有定义每个字节，我只好用 char* 指针手动地一个一个字节地赋值。

4.4 程序源代码

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/types.h>
```

```
#include <boost/thread.hpp>
#ifdef _WIN32
#include <WinSock2.h>
#include <WS2tcpip.h>
#pragma comment( lib, "ws2_32.lib" )
#else
#include <sys/socket.h>
#include <arpa/inet.h>
#include <netdb.h>
#include <signal.h>
typedef int SOCKET;
#define INVALID_SOCKET  (SOCKET)(~0)
#define SOCKET_ERROR            (-1)
#endif

#define LISTENING_PORT     8080
#define MAX_BUF 800000
#define MAX_CONNECTION_TRY 5

void parseDomainNameAndPortFromUrl( const char * url,
         char * domain_name, unsigned short & port )
{
    for ( int i = 0 ; url[i] != '\0'; ++ i )
    {
        if ( url[i] == '/' && url[i+1] == '/' )
        {
            for ( int j = i + 2; url[j] != '\0'; ++j )
            {
                if ( url[j] == ':' || url[j] == '/' )
                {
                    memset( domain_name, 0, strlen(url) );
                    memcpy( domain_name, url + i + 2, j - i - 2 );
                    if ( url[j] == ':' )
                    {
                        port = 0;
                        for ( int k = j+1 ; url[k] != '/' ; ++ k )
                        {
                            port *= 10;
                            port += url[k] - '0';
                        }
                    }
                    else
                        port = 80;
                    return;
                }
            }
            throw 400;
            return;
        }
    }
```

```
    throw 400;
}

struct sockaddr_in GetAddrFromHttpHeader( const char * HttpHeader )
{
    const char * method_end = strchr( HttpHeader, ' ' );
    const char * url_end = strchr( 1 + method_end, ' ' );

    char * method = new char[method_end - HttpHeader + 1 ];
    memset( method, 0, method_end - HttpHeader + 1 );
    strncpy( method, HttpHeader, method_end - HttpHeader );
    if ( !strcmp( method, "CONNECT" ) )
    {
        delete [] method;
        method = NULL;
        throw 405;
    }
    delete [] method;
    method = NULL;

    char * url = new char[url_end - method_end + 1];
    memset( url, 0, url_end - method_end + 1 );
    strncpy( url, method_end + 1, url_end - method_end - 1 );
    char * domain_name = new char[strlen(url)];
    unsigned short port;
    try
    {
        parseDomainNameAndPortFromUrl( url, domain_name, port );
    }
    catch( int errorCode )
    {
        delete [] url;
        delete [] domain_name;
        throw errorCode;
    }

    hostent * host = gethostbyname( domain_name );
    if ( host == NULL )
    {
        delete [] url;
        delete [] domain_name;
        throw 502;
    }
    struct sockaddr_in SockAddr;
    memset( &SockAddr, 0, sizeof(SockAddr) );
    SockAddr.sin_family = host->h_addrtype;
#ifdef _WIN32
    SockAddr.sin_addr.s_net = (unsigned char)(host->h_addr_list[0][0]);
    SockAddr.sin_addr.s_host = (unsigned char)(host->h_addr_list[0][1]);
    SockAddr.sin_addr.s_lh = (unsigned char)(host->h_addr_list[0][2]);
```

```
    SockAddr.sin_addr.s_impno = (unsigned char)(host->h_addr_list[0][3]);
#else
    ((unsigned char *)&(SockAddr.sin_addr))[0] =
              (unsigned char)(host->h_addr_list[0][0]);
    ((unsigned char *)&(SockAddr.sin_addr))[1] =
              (unsigned char)(host->h_addr_list[0][1]);
    ((unsigned char *)&(SockAddr.sin_addr))[2] =
              (unsigned char)(host->h_addr_list[0][2]);
    ((unsigned char *)&(SockAddr.sin_addr))[3] =
              (unsigned char)(host->h_addr_list[0][3]);
#endif
    SockAddr.sin_port = htons( port );

    printf( "Domain name: %s\n", domain_name );
    printf( "Server IP: %s\n", inet_ntoa(SockAddr.sin_addr) );
    printf( "Port: %hu\n", port );

    delete [] domain_name;
    delete [] url;

    return SockAddr;
}

void HTTP_proxy( SOCKET * pBrowserSocket )
{
    char buffer[MAX_BUF+1];
    int bytesSent, bytesRecv;
#ifdef _WIN32
    int timeout = 1000;
#else
    struct timeval timeout = {3,0};
#endif
    setsockopt( *pBrowserSocket, SOL_SOCKET, SO_RCVTIMEO,
            (char *)&timeout,sizeof(timeout));
    memset( buffer, 0, sizeof(buffer) );
    if ( ( bytesRecv = recv( *pBrowserSocket, buffer,
            MAX_BUF, 0 ) ) <= 0 )
    {
#ifdef _WIN32
        shutdown( *pBrowserSocket, SD_BOTH );
        closesocket( *pBrowserSocket );
#else
        shutdown( *pBrowserSocket, SHUT_RDWR );
        close( *pBrowserSocket );
#endif
        delete pBrowserSocket;
        return;
    }
    buffer[bytesRecv] = '\0';
    try
```

```
    {
        SOCKET clientSocketOfWebServer = socket(
                        AF_INET, SOCK_STREAM, IPPROTO_TCP );
        if ( clientSocketOfWebServer == INVALID_SOCKET )
        {
#ifdef _WIN32
            shutdown( *pBrowserSocket, SD_BOTH );
            closesocket( *pBrowserSocket );
#else
            shutdown( *pBrowserSocket, SHUT_RDWR );
            close( *pBrowserSocket );
#endif
            delete pBrowserSocket;
            return;
        }
        struct sockaddr_in WebServerSockAddr;
        WebServerSockAddr = GetAddrFromHttpHeader( buffer );
        setsockopt( clientSocketOfWebServer, SOL_SOCKET,
                SO_RCVTIMEO,(char *)&timeout,sizeof(timeout));
        int connectResult;
        for ( int i = 0 ; i < MAX_CONNECTION_TRY ; i ++ )
        {
            connectResult = connect( clientSocketOfWebServer,
                (const struct sockaddr *)&WebServerSockAddr,
                sizeof(WebServerSockAddr) );
            if ( connectResult != SOCKET_ERROR )
                break;
        }
        if ( connectResult == SOCKET_ERROR )
        {
#ifdef _WIN32
            shutdown( *pBrowserSocket, SD_BOTH );
            shutdown( clientSocketOfWebServer, SD_BOTH );
            closesocket( *pBrowserSocket );
            closesocket( clientSocketOfWebServer );
#else
            shutdown( *pBrowserSocket, SHUT_RDWR );
            shutdown( clientSocketOfWebServer, SHUT_RDWR );
            close( *pBrowserSocket );
            close( clientSocketOfWebServer );
#endif
            delete pBrowserSocket;
            return;
        }
        bytesSent = send( clientSocketOfWebServer, buffer, bytesRecv, 0 );
        if ( bytesSent <= 0 )
        {
#ifdef _WIN32
            shutdown( *pBrowserSocket, SD_BOTH );
            shutdown( clientSocketOfWebServer, SD_BOTH );
```

```
            closesocket( *pBrowserSocket );
            closesocket( clientSocketOfWebServer );
#else
            shutdown( *pBrowserSocket, SHUT_RDWR );
            shutdown( clientSocketOfWebServer, SHUT_RDWR );
            close( *pBrowserSocket );
            close( clientSocketOfWebServer );
#endif
            delete pBrowserSocket;
            return;
        }
        while ( ( bytesRecv = recv( clientSocketOfWebServer,
                    buffer, MAX_BUF, 0 ) ) > 0 )
        {
            bytesSent = send( *pBrowserSocket, buffer, bytesRecv, 0 );
            if ( bytesSent <= 0 )
                break;
        }
#ifdef _WIN32
        shutdown( clientSocketOfWebServer, SD_BOTH );
        closesocket( clientSocketOfWebServer );
#else
        shutdown( clientSocketOfWebServer, SHUT_RDWR );
        close( clientSocketOfWebServer );
#endif
    }
    catch ( int num )
    {
        const char * sendBuf = NULL;
        if ( num == 400 )
        {
            puts( "400 Bad Request" );
            sendBuf = "HTTP/1.0 400 Bad Request\r\n"
                "Content-Type: text/plain; charset=UTF-8\r\n"
                "Connection: close\r\n"
                "Content-Length: 0\r\n\r\n";
        }
        else if ( num == 502 )
        {
            puts( "502 Bad Gatway" );
            sendBuf = "HTTP/1.0 502 Bad Gateway\r\n"
                "Content-Type: text/plain; charset=UTF-8\r\n"
                "Connection: close\r\n"
                "Content-Length: 0\r\n\r\n";
        }
        else if ( num == 405 )
        {
            puts( "405 Method Not Allowed" );
            sendBuf = "HTTP/1.0 405 Method Not Allowed\r\n"
                "Content-Type: text/plain; charset=UTF-8\r\n"
```

```
                "Connection: close\r\n"
                "Content-Length: 0\r\n\r\n";
        }
        else
        {
            puts( "503 Service Unavailable" );
            sendBuf = "HTTP/1.0 503 Service Unavailable\r\n"
                "Content-Type: text/plain; charset=UTF-8\r\n"
                "Connection: close\r\n"
                "Content-Length: 0\r\n\r\n";
        }
        bytesSent = send( *pBrowserSocket, sendBuf, strlen(sendBuf), 0 );
    }
    catch( ... )
    {
        const char * sendBuf = "HTTP/1.0 503 Service Unavailable\r\n"
            "Content-Type: text/plain; charset=UTF-8\r\n"
            "Connection: close\r\n"
            "Content-Length: 0\r\n\r\n";
        bytesSent = send( *pBrowserSocket, sendBuf, strlen(sendBuf), 0 );
    }
#ifdef _WIN32
    shutdown( *pBrowserSocket, SD_BOTH );
    closesocket( *pBrowserSocket );
#else
    shutdown( *pBrowserSocket, SHUT_RDWR );
    close( *pBrowserSocket );
#endif
    delete pBrowserSocket;
}

int main()
{
#ifdef _WIN32
    WORD wVersionRequested;
    WSADATA wsaData;
    int err;
    wVersionRequested = MAKEWORD(1, 1);
    err = ::WSAStartup( wVersionRequested, &wsaData );
    if ( err != 0 )
        return EXIT_FAILURE;
    if ( LOBYTE(wsaData.wVersion) != 1 || HIBYTE(wsaData.wVersion) != 1 )
    {
        fputs( "WSAStartup failed\n", stderr );
        ::WSACleanup();
        return EXIT_FAILURE;
    }
#endif
    SOCKET proxyServerSocket = socket( AF_INET, SOCK_STREAM, IPPROTO_TCP );
    if ( proxyServerSocket == INVALID_SOCKET )
```

```
    {
        fputs( "Socket initailizing failed\n", stderr );
#ifdef _WIN32
        ::WSACleanup();
#endif
        return EXIT_FAILURE;
    }
    struct sockaddr_in proxyServerSockAddr;
    memset( &proxyServerSockAddr, 0, sizeof(proxyServerSockAddr) );
    proxyServerSockAddr.sin_family = AF_INET;
    proxyServerSockAddr.sin_addr.s_addr = INADDR_ANY;
    proxyServerSockAddr.sin_port = htons( LISTENING_PORT );
    if ( bind( proxyServerSocket, (struct sockaddr *)&proxyServerSockAddr,
        sizeof(proxyServerSockAddr) ) == SOCKET_ERROR )
    {
        fputs( "Bind failed\n", stderr );
#ifdef _WIN32
        shutdown( proxyServerSocket, SD_BOTH );
        closesocket( proxyServerSocket );
        ::WSACleanup();
#else
        shutdown( proxyServerSocket, SHUT_RDWR );
        close( proxyServerSocket );
#endif
        return EXIT_FAILURE;
    }
    if ( listen( proxyServerSocket, 1 ) == SOCKET_ERROR )
    {
        fputs( "Listen failed\n", stderr );
#ifdef _WIN32
        shutdown( proxyServerSocket, SD_BOTH );
        closesocket( proxyServerSocket );
        ::WSACleanup();
#else
        shutdown( proxyServerSocket, SHUT_RDWR );
        close( proxyServerSocket );
#endif
        return EXIT_FAILURE;
    }
#ifndef _WIN32
    struct sigaction sa;
    sa.sa_handler = SIG_IGN;
    sigaction( SIGPIPE, &sa, 0 );
#endif
    SOCKET * pBrowserSocket;
    while ( true )
    {
        pBrowserSocket = new SOCKET;
        do
        {
```

```
            *pBrowserSocket = accept ( proxyServerSocket, NULL, NULL );
            if ( *pBrowserSocket != SOCKET_ERROR )
                puts( "Browser connected" );
        } while ( *pBrowserSocket == SOCKET_ERROR );
        try
        {
            boost::thread thrd(HTTP_proxy, pBrowserSocket);
        }
        catch( ... )
        {
        }
    }
#ifdef _WIN32
    ::WSACleanup();
#endif
    return EXIT_SUCCESS;
}
```