1.简单服务器

```
/*
#include <winsock2.h>
#pragma comment(lib,"WS2_32.lib")
*/
WSADATA wsd;
static UINT port=%%1;
UINT Listen(LPVOID pParam)
{
	SOCKET sServer,sClient;
	char buf[1024];
	int retVal;
	if(WSAStartup(MAKEWORD(2,2),&wsd)!=0)
	{
		return -1;//失败
	}
	sServer=socket(AF_INET,SOCK_STREAM,IPPROTO_TCP);
	if(INVALID_SOCKET==sServer)
	{
		WSACleanup();
		return -1;//创建套接字失败
	}
	SOCKADDR_IN addrServ;
	addrServ.sin_family=AF_INET;
	addrServ.sin_port=htons((short)pParam);
	addrServ.sin_addr.s_addr=INADDR_ANY;
	retVal=bind(sServer,(LPSOCKADDR)&addrServ,sizeof(SOCKADDR_IN));
	if(SOCKET_ERROR==retVal)
	{
		closesocket(sServer);
		WSACleanup();
		return -1;//绑定套接字失败
	}
	retVal=listen(sServer,1);
	if(SOCKET_ERROR==retVal)
	{
		closesocket(sServer);
		WSACleanup();
		return -1;//开始监听失败
	}
	sockaddr_in addrClient;
	int addrClientlen=sizeof(addrClient);
	sClient=accept(sServer,(sockaddr FAR*)&addrClient,&addrClientlen);
	if(INVALID_SOCKET==sClient)
	{
		closesocket(sServer);
		WSACleanup();
		return -1;//开始接受客户端连接失败
	}
	ZeroMemory(buf,sizeof(buf));
	retVal=recv(sClient,buf,sizeof(buf),0);
	if(SOCKET_ERROR==retVal)
	{
		closesocket(sServer);
		closesocket(sClient);
		WSACleanup();
```

```
            return -1;//接收数据失败
        }
        CString %%2(buf);
        closesocket(sServer);
        closesocket(sClient);
        WSACleanup();
        return 0;
}
CWinThread *pThread=AfxBeginThread(Listen,&port);
```

2.简单客户端

```
/*
#include <winsock2.h>
#pragma comment(lib,"WS2_32.lib")
*/
WSADATA wsd;
SOCKET sHost;
SOCKADDR_IN servAddr;
char buf[1024];
int retVal;
if(WSAStartup(MAKEWORD(2,2),&wsd)!=0)
{
        return -1;//失败
}
sHost=socket(AF_INET,SOCK_STREAM,IPPROTO_TCP);
if(INVALID_SOCKET==sHost)
{
        WSACleanup();
        return -1;//创建套接字失败
}
servAddr.sin_family=AF_INET;
servAddr.sin_addr.s_addr=inet_addr(%%3);
servAddr.sin_port=htons((short)%%2);
int nServAddlen=sizeof(servAddr);
retVal=connect(sHost,(LPSOCKADDR)&servAddr,sizeof(servAddr));
if(SOCKET_ERROR==retVal) {
        closesocket(sHost);
        WSACleanup();
        return -1;//连接服务器失败
}
ZeroMemory(buf,sizeof(buf));
strcpy(buf,%%3);
retVal=send(sHost,buf,sizeof(buf),0);
if(SOCKET_ERROR==retVal)
{
        closesocket(sHost);
        WSACleanup();
        return -1;//向服务器发送数据失败
}
closesocket(sHost);
WSACleanup();
```

3.获得本机IP

```
/*
#include <winsock2.h>
#pragma comment(lib,"WS2_32.lib")
```

```
*/
WSADATA wsd;
if(WSAStartup(MAKEWORD(2,2),&wsd)!=0)
{
	return -1;//失败
}
char szHostname[100],szHostaddress[200];
if(gethostname(szHostname,sizeof(szHostname))!=SOCKET_ERROR)
{
	HOSTENT *pHostEnt=gethostbyname(szHostname);
	if(pHostEnt!=NULL){
		sprintf(szHostaddress,"%d.%d.%d.%d",
			( pHostEnt->h_addr_list[0][0]&0x00ff ),
			( pHostEnt->h_addr_list[0][1]&0x00ff ),
			( pHostEnt->h_addr_list[0][2]&0x00ff ),
			( pHostEnt->h_addr_list[0][3]&0x00ff ));
	}
}
else
return;
CString %%1(szHostaddress);
```

4.端对端通信

```
/*
#include <winsock2.h>
#pragma comment(lib,"WS2_32.lib")
*/
WSADATA wsd;
SOCKET s;
char buf[1024];
if(WSAStartup(MAKEWORD(2,2),&wsd)!=0)
{
	return -1;//失败
}
s=socket(AF_INET,SOCK_DGRAM,0);
if(s==INVALID_SOCKET)
{
	WSACleanup();
	return -1;//创建套接字失败
}
SOCKADDR_IN servAddr;
servAddr.sin_family=AF_INET;
servAddr.sin_addr.s_addr=inet_addr(%%1);
servAddr.sin_port=htons(INADDR_ANY);
if(bind(s,(SOCKADDR*)&servAddr,sizeof(SOCKADDR_IN))==SOCKET_ERROR)
{
	closesocket(s);
	WSACleanup();
	return -1;//绑定套接字失败
}
int nServAddrlen=sizeof(servAddr);
ZeroMemory(buf,sizeof(buf));
if(recvfrom(s,buf,sizeof(buf),0,(SOCKADDR*)&servAddr,&nServAddrlen)==SOCKET_ERROR)
{
	closesocket(s);
	WSACleanup();
```

```
	return -1;//接收数据失败
}
CString %%2(buf);
ZeroMemory(buf,sizeof(buf));
strcpy(buf,%%3);
SOCKADDR_IN clientAddr;
clientAddr.sin_family=AF_INET;
clientAddr.sin_addr.s_addr=inet_addr(%%4);
clientAddr.sin_port=htons((short)%%5);
int nClientlen=sizeof(clientAddr);
if(sendto(s,buf,sizeof(buf),0,(SOCKADDR*)&clientAddr,nClientlen)==SOCKET_ERROR)
{
	closesocket(s);
	WSACleanup();
	return -1;//向服务器发送数据失败
}
closesocket(s);
WSACleanup();
```

5.点对点通信

```
/*
#include <winsock2.h>
#pragma comment(lib,"WS2_32.lib")
*/
WSADATA wsd;
SOCKADDR_IN addrServ,addrServ2;
SOCKET sServer,sClient,sHost;
int retVal;
sockaddr_in addrClient;
char buf[1024];
static UINT port=%%2;
BOOL listenerRun=TRUE;
UINT Listen(LPVOID pParam)
{
	addrServ.sin_family=AF_INET;
	addrServ.sin_port=htons((UINT)pParam);
	addrServ.sin_addr.s_addr=INADDR_ANY;
	retVal=bind(sServer,(LPSOCKADDR)&addrServ,sizeof(SOCKADDR_IN));
	if(SOCKET_ERROR==retVal)
	{
		closesocket(sServer);
		WSACleanup();
		return -1;//绑定套接字失败
	}
	retVal=listen(sServer,1);
	if(SOCKET_ERROR==retVal)
	{
		closesocket(sServer);
		WSACleanup();
		return -1;//开始监听失败
	}
	int addrClientlen=sizeof(addrClient);
	sClient=accept(sServer,(sockaddr FAR*)&addrClient,&addClientlen);
	if(INVALID_SOCKET==sClient)
	{
		closesocket(sServer);
```

```
            WSACleanup();
            return -1;//接收客戶端请求失败
        }
        while(listenerRun)
        {
            ZeroMemory(buf,sizeof(buf));
            retVal=recv(sClient,buf,sizeof(buf));
            if(SOCKET_ERROR==retVal)
            {
                closesocket(sServer);
                closesocket(sClient);
                WSACleanup();
                return -1;//接收客户端数据失败
            }
            CString %%4(buf);
        }
    }
    if(WSAStartup(MAKEWORD(2,2),&wsd)!=0)
    {
        return -1;//失败
    }
    sServer=socket(AF_INET,SOCK_STREAM,IPPROTO_TCP);
    if(INVALID_SOCKET==sServer)
    {
        WSACleanup();
        return -1;//创建套接字失败
    }
    CWinThread *pThread=AfxBeginThread(Listen,&port);
    sHost=socket(AF_INET,SOCK_STREAM,IPPROTO_TCP);
    if(INVALID_SOCKET==sHost)
    {
        WSACleanup();
        return -1;//创建套接字失败
    }
    servAddr2.sin_family=AF_INET;
    servAddr2.sin_addr.s_addr=inet_addr(%%1);
    servAddr.sin_port=htons((short)%%3);
    int nServerAddrlen=sizeof(servAddr2);
    retVal=connect(sHost,(LPSOCKADDR)&servAddr2,sizeof(servAddr2));
    if(SOCKET_ERROR==retVal)
    {
        closesocket(sHost);
        WSACleanup();
        return -1;//连接失败
    }
    zeroMemory(buf,sizeof(buf));
    strcpy(buf,%%5);
    retVal=send(sHost,buf,sizeof(buf),0);
    if(SOCKET_ERROR==retVal)
    {
        closesocket(sHost);
        WSACleanup();
        return -1;//向发送数据失败
    }
    listenerRun=FALSE;
    DWORD dwExitCode;
```

```
::GetExitCodeThread(pThread->m_hThread,&dwExitCode);
pThread=null;
closesocket(sServer);
closesocket(sClient);
closesocket(sHost);
WSACleanup();
```

6.UDP对时服务器端

```
/*
#include <winsock2.h>
#pragma comment(lib,"WS2_32.lib")
*/
WSADATA wsd;
SOCKET s;
char buf[1024];
if(WSAStartup(MAKEWORD(2,2),&wsd)!=0)
{
	return -1;//失败
}
s=socket(AF_INET,SOCK_DGRAM,0);
if(s==INVALID_SOCKET)
{
	WSACleanup();
	return -1;//创建套接字失败
}
SOCKADDR_IN servAddr;
servAddr.sin_family=AF_INET;
servAddr.sin_addr.s_addr=inet_addr("127.0.0.1");
servAddr.sin_port=htons(5000);
if(bind(s,(SOCKADDR*)&servAddr,sizeof(SOCKADDR_IN))==SOCKET_ERROR)
{
	closesocket(s);
	WSACleanup();
	return -1;//绑定套接字失败
}
int nServAddrlen=sizeof(servAddr);
ZeroMemory(buf,sizeof(buf));
if(recvfrom(s,buf,sizeof(buf),0,(SOCKADDR*)&servAddr,&nServAddrlen)==SOCKET_ERROR)
{
	closesocket(s);
	WSACleanup();
	return -1;//接收数据失败
}
CString str(buf);
if(str=="TimeNow")
{
	SOCKADDR_IN clientAddr;
	clientAddr.sin_family=AF_INET;
	clientAddr.sin_addr.s_addr=inet_addr("127.0.0.1");
	clientAddr.sin_port=htons((short)2000);
	int nClientlen=sizeof(clientAddr);
	SYSTEMTIME systime;
	GetLocalTime(&systime);
	if(sendto(s,(char
*)&systime,sizeof(SYSTEMTIME),0,(SOCKADDR*)&clientAddr,nClientlen)==SOCKET_ERROR)
	{
```

```
            closesocket(s);
            WSACleanup();
            return -1;//向服务器发送数据失败
        }
    }
    closesocket(s);
    WSACleanup();
```

7.UDP对时客户端

```
    /*
    #include <winsock2.h>
    #pragma comment(lib,"WS2_32.lib")
    */
    WSADATA wsd;
    SOCKET s;
    char buf[1024];
    if(WSAStartup(MAKEWORD(2,2),&wsd)!=0)
    {
        return -1;//失败
    }
    s=socket(AF_INET,SOCK_DGRAM,0);
    if(s==INVALID_SOCKET)
    {
        WSACleanup();
        return -1;//创建套接字失败
    }
    SOCKADDR_IN servAddr;
    servAddr.sin_family=AF_INET;
    servAddr.sin_addr.s_addr=inet_addr("127.0.0.1");
    servAddr.sin_port=htons(2000);
    if(bind(s,(SOCKADDR*)&servAddr,sizeof(SOCKADDR_IN))==SOCKET_ERROR)
    {
        closesocket(s);
        WSACleanup();
        return -1;//绑定套接字失败
    }
    int nServAddrlen=sizeof(servAddr);
    ZeroMemory(buf,sizeof(buf));
    CString ss="TimeNow";
    strcpy(buf,ss);
    SOCKADDR_IN clientAddr;
    clientAddr.sin_family=AF_INET;
    clientAddr.sin_addr.s_addr=inet_addr("127.0.0.1");
    clientAddr.sin_port=htons((short)5000);
    int nClientlen=sizeof(clientAddr);
    if(sendto(s,buf,sizeof(buf),0,(SOCKADDR*)&clientAddr,nClientlen)==SOCKET_ERROR)
    {
        closesocket(s);
        WSACleanup();
        return -1;//向服务器发送数据失败
    }
    memset(buf,0,1024);
    if(recvfrom(s,buf,sizeof(buf),0,(SOCKADDR*)&servAddr,&nServAddrlen)==SOCKET_ERROR)
    {
        closesocket(s);
        WSACleanup();
```

```
        return -1;//接收数据失败
}
SYSTEMTIME systime;
memcpy(&systime,buf,16);
SetLocalTime(&systime);//设置本地与服务器时间同步。
closesocket(s);
WSACleanup();
```

8.点对点传输文件

```
CFile myFile;
if(!myFile.Open(Dlg.GetPathName(), CFile::modeRead | CFile::typeBinary))
{
        AfxMessageBox("文件不存在!",MB_OK|MB_ICONERROR);
        return;
}

CSocket sockSrvr;
sockSrvr.Create(800);

sockSrvr.Listen();
CSocket sockRecv;
sockSrvr.Accept(sockRecv);

SOCKET_STREAM_FILE_INFO StreamFileInfo;
WIN32_FIND_DATA             FindFileData;

FindClose(FindFirstFile(Dlg.GetPathName(),&FindFileData));
memset(&StreamFileInfo,0,sizeof(SOCKET_STREAM_FILE_INFO));
strcpy(StreamFileInfo.szFileTitle,myFile.GetFileTitle());

StreamFileInfo.dwFileAttributes     =       FindFileData.dwFileAttributes;
StreamFileInfo.ftCreationTime       =       FindFileData.ftCreationTime;
StreamFileInfo.ftLastAccessTime     =       FindFileData.ftLastAccessTime;
StreamFileInfo.ftLastWriteTime      =       FindFileData.ftLastWriteTime;
StreamFileInfo.nFileSizeHigh        =       FindFileData.nFileSizeHigh;
StreamFileInfo.nFileSizeLow         =       FindFileData.nFileSizeLow;

sockRecv.Send(&StreamFileInfo,sizeof(SOCKET_STREAM_FILE_INFO));

UINT dwRead=0;
while(dwRead<StreamFileInfo.nFileSizeLow)
{
        byte* data = new byte[1024];
        UINT dw=myFile.Read(data, 1024);
        sockRecv.Send(data, dw);
        dwRead+=dw;
}
myFile.Close();
sockRecv.Close();
////////////////////////////////////////////
AfxSocketInit(NULL);
CSocket sockClient;
sockClient.Create();

CString szIP;
GetDlgItemText(IDC_EDIT_IPADDRESS,szIP);
```

```
if(!sockClient.Connect((LPCTSTR)szIP, 800))
{
	AfxMessageBox("连接到对方机器失败!");
	return;
}


SOCKET_STREAM_FILE_INFO StreamFileInfo;
sockClient.Receive(&StreamFileInfo,sizeof(SOCKET_STREAM_FILE_INFO));

CFile destFile(StreamFileInfo.szFileTitle, CFile::modeCreate | CFile::modeWrite | CFile::typeBinary);

UINT dwRead = 0;
while(dwRead<StreamFileInfo.nFileSizeLow)
{
	byte* data = new byte[1024];
	memset(data,0,1024);

	UINT dw=sockClient.Receive(data, 1024);
	destFile.Write(data, dw);

	dwRead+=dw;
}

SetFileTime((HANDLE)destFile.m_hFile,&StreamFileInfo.ftCreationTime,
		&StreamFileInfo.ftLastAccessTime,&StreamFileInfo.ftLastWriteTime);
destFile.Close();
SetFileAttributes(StreamFileInfo.szFileTitle,StreamFileInfo.dwFileAttributes);
sockClient.Close();
AfxMessageBox("接收完毕!");
```

9.发送邮件

```
/*
#import <cdonts.dll>
#include "tchar.h"
#include "stdio.h"
*/
CoInitialize(NULL);
try
{
	CDONTS::INewMailPtr spNewMail(__uuidof(CDONTS::NewMail));
	spNewMail->From = _T("YourName");
	spNewMail->To = _T("zxgdata@21cn.com");
	spNewMail->Subject = _T("Testing");
	spNewMail->Body = _T("Put your message here");
	spNewMail-
>AttachFile(_variant_t(_bstr_t("C:\\tmp\\test\\mail\\mail.cpp")),_variant_t((long)DISP_E_PARAMNOTFOUND
, VT_ERROR),_variant_t((long)DISP_E_PARAMNOTFOUND, VT_ERROR));
	spNewMail->Send();
	printf("send ok");
}
catch(_com_error &ComError)
{
	printf("%s\n",ComError.Description());
}
CoUninitialize();
```

10.接收邮件
利用JMail组件快速构建邮件程序
http://www.vckbase.com/document/viewdoc/?id=684
http://www.vckbase.com/document/viewdoc/?id=712

11.多线程阻塞通信

12.多线程非阻塞通信

13.多线程文件断点续传
//Send.dsw
//Thead.h
// 线程对象封装
//
```
#ifndef _THREAD_INCLUDE_
#define _THREAD_INCLUDE_

class CThread
{
private:
	static DWORD WINAPI ThreadProc(LPVOID pVoid);
protected:
	BOOL	m_bTerminated; // 线程是否终止的标志

	virtual void Execute(void) = 0;
public:
	HANDLE  m_hThread; // 线程句柄

	CThread(void);
	~CThread(void);

	void Resume(void);
	void Terminate(void);
	HANDLE GetThreadHandle(void);
};

#endif // #ifndef _THREAD_INCLUDE_
```

//Thead.cpp
// 线程对象封装
//
```
#include "stdafx.h"
#include "Thread.h"

CThread::CThread(void)
{
	m_bTerminated = FALSE;

	DWORD dwThreadID;
	m_hThread = CreateThread(NULL, 0, ThreadProc, this, CREATE_SUSPENDED, &dwThreadID);
}

CThread::~CThread(void)
{
	CloseHandle(m_hThread);
```

```
	m_hThread = NULL;
}

DWORD CThread::ThreadProc(LPVOID pVoid)
{
	((CThread *)(pVoid))->Execute();

	return 0;
}

void CThread::Resume(void)
{
	ResumeThread(m_hThread);
}

void CThread::Terminate(void)
{
	m_bTerminated = TRUE;
}

HANDLE CThread::GetThreadHandle(void)
{
	return m_hThread;
}

m_pTcpClient = new CTcpClient(this);
m_strServerIp = "127.0.0.1";
m_nPort = 8000;
m_dwPackageSize = 1024;
m_strFileName = "d:\\a.pdf";
UpdateData(FALSE);

m_pTcpClient->SetOnSocketSendErr(OnSocketSendErr);
m_pTcpClient->SetOnSocketRecvErr(OnSocketRecvErr);
m_pTcpClient->SetOnSocketClose(OnSocketClose);
m_pTcpClient->SetOnOneNetMsg(OnOneNetMsg);
m_pTcpClient->SetOnSendFileSucc(OnSendFileSucc);
m_pTcpClient->SetOnSendFileFail(OnSendFileFail);
m_pTcpClient->SetOnSendFileRefuseRecv(OnSendFileRefuseRecv);
m_pTcpClient->SetOnSendFileCancelRecv(OnSendFileCancelRecv);
m_pTcpClient->SetOnSendFileRecvFail(OnSendFileRecvFail);
m_pTcpClient->SetOnSendFileProgress(OnSendFileProgress);
CStatic m_ctlCnnStatus;
CStatic m_ctlInfo;
DWORD   m_dwPackageSize;
CString m_strServerIp;
int             m_nPort;
CString m_strFileName;
CString m_strMsg;

void CSendDlg::OnConnect()
{
	if(!UpdateData())
		return;


	m_pTcpClient->SetAddr((char *)(LPCTSTR)m_strServerIp);
```

```
	m_pTcpClient->SetPort(m_nPort);
	m_pTcpClient->SetPackageSize(m_dwPackageSize);

	m_ctlCnnStatus.SetWindowText("请等待...");
	if(!m_pTcpClient->Connect())
		m_ctlCnnStatus.SetWindowText("连接失败!");
	else
		m_ctlCnnStatus.SetWindowText("已连接");
}

void CSendDlg::OnDisconnect()
{
	m_pTcpClient->Disconnect();
	m_ctlCnnStatus.SetWindowText("断开连接");
}

void CSendDlg::OnSendFile()
{
	if(!UpdateData())
		return;

	m_pTcpClient->SetPackageSize(m_dwPackageSize);
	if(!m_pTcpClient->SendFile((char *)(LPCTSTR)m_strFileName))
		AfxMessageBox("发生文件失败");
}

void CSendDlg::OnSendMsg(void)
{
	char s[99999];

	if(!UpdateData())
		return;


	sprintf(s, "@00000001%s", m_strMsg);
	m_pTcpClient->SendNetMsg(s, strlen(s) - 6);
}

void CSendDlg::OnSocketSendErr(void *pNotifyObj, SOCKET hSocket)
{
	CSendDlg *pSendDlg = (CSendDlg *)pNotifyObj;
	pSendDlg->m_ctlCnnStatus.SetWindowText("发送数据出错");
}

void CSendDlg::OnSocketRecvErr(void *pNotifyObj, SOCKET hSocket)
{
	CSendDlg *pSendDlg = (CSendDlg *)pNotifyObj;
	pSendDlg->m_ctlCnnStatus.SetWindowText("接收数据出错");
}

void CSendDlg::OnSocketClose(void *pNotifyObj, SOCKET hSocket)
{
	CSendDlg *pSendDlg = (CSendDlg *)pNotifyObj;
	pSendDlg->m_ctlCnnStatus.SetWindowText("断开连接");
}
```

```
void CSendDlg::OnOneNetMsg(void *pNotifyObj, char *Msg, int nMsgLen)
{
	CSendDlg *pSendDlg = (CSendDlg *)pNotifyObj;
	char s[9999];
	CString strInfo;

	strncpy(s, Msg, nMsgLen);
	s[nMsgLen] = 0;
	strInfo = s;

	pSendDlg->DispInfo(strInfo);
}

void CSendDlg::OnSendFileSucc(void *pNotifyObj, char *szPathName)
{
	CSendDlg *pSendDlg = (CSendDlg *)pNotifyObj;

	pSendDlg->DispInfo("OnSendFileSucc");
}

void CSendDlg::OnSendFileFail(void *pNotifyObj, char *szPathName)
{
	CSendDlg *pSendDlg = (CSendDlg *)pNotifyObj;

	pSendDlg->DispInfo("OnSendFileFail");
}

// 接收方拒绝接收文件
void CSendDlg::OnSendFileRefuseRecv(void *pNotifyObj, char *szPathName)
{
	CSendDlg *pSendDlg = (CSendDlg *)pNotifyObj;

	pSendDlg->DispInfo("OnSendFileRefuseRecv");
}

// 接收方拒绝文件
void CSendDlg::OnSendFileCancelRecv(void *pNotifyObj, char *szPathName)
{
	CSendDlg *pSendDlg = (CSendDlg *)pNotifyObj;

	pSendDlg->DispInfo("OnSendFileCancelRecv");
}

// 接收方取消接收
void CSendDlg::OnSendFileRecvFail(void *pNotifyObj, char *szPathName)
{
	CSendDlg *pSendDlg = (CSendDlg *)pNotifyObj;

	pSendDlg->DispInfo("OnSendFileRecvFail");
}

void CSendDlg::OnSendFileProgress(void *pNotifyObj, int nSentBytes, int nTotalBytes)
{
	CSendDlg *pSendDlg = (CSendDlg *)pNotifyObj;
	CString strInfo;

	strInfo.Format("%d / %d", nSentBytes, nTotalBytes);
```

```
	pSendDlg->DispInfo(strInfo);
}

void CSendDlg::DispInfo(CString strInfo)
{
	m_ctlInfo.SetWindowText(strInfo);
}

void CSendDlg::OnCancelSend()
{
	m_pTcpClient->CancelSendFile();
}

void CSendDlg::OnDestroy()
{
	CDialog::OnDestroy();

	m_pTcpClient->Disconnect();
}
delete m_pTcpClient;

//Recv.dsw
#include "Shlwapi.h"

// 判断文件是否存在
BOOL IsFileExists(char *pszPathName);
// 创建多层目录,成功返回TRUE,识别返回FALSE
BOOL ForceDirectories(char *pszDir);
// 扩展文件操作
BOOL DeleteFileEx(char *szPathName, BOOL bAllowUndo = FALSE);
BOOL RenameFileEx(char *szOldPathName, char *szNewPathName);
BOOL MoveFileEx(char *szSrcPathName, char *szDstPathName);
BOOL CopyFileEx(char *szSrcPathName, char *szDstPathName);
// 重新启动操作系统
BOOL RebootWindows();
// 设置程序是否在操作系统启动后自动运行
void SetAutoRun(BOOL bEnable);
BOOL ShutDownWin98();
BOOL ShutDownWinNT();

BOOL IsLegalFileName(char *szFileName);
m_pTcpServer1 = new CTcpServer(this);
m_pTcpServer1->SetBindAddr("");
m_pTcpServer1->SetPort(8000);
m_pTcpServer1->SetOnAccept(OnAccept);
m_pTcpServer1->SetOnAcceptErr(OnAcceptErr);
m_pTcpServer1->SetOnSocketConnect(OnSocketConnect);
m_pTcpServer1->SetOnSocketDisconnect(OnSocketDisconnect);
m_pTcpServer1->SetOnSocketSendErr(OnSocketSendErr);
m_pTcpServer1->SetOnSocketRecvErr(OnSocketRecvErr);
m_pTcpServer1->SetOnOneNetMsg(OnOneNetMsg);
m_pTcpServer1->SetOnRecvFileStart(OnRecvFileStart);
m_pTcpServer1->SetOnRecvFileProgress(OnRecvFileProgress);
m_pTcpServer1->SetOnRecvFileFail(OnRecvFileFail);
m_pTcpServer1->SetOnRecvFileSucc(OnRecvFileSucc);
m_pTcpServer1->SetOnRecvFileCancel(OnRecvFileCancel);
```

```
if(!m_pTcpServer1->StartAccept())
{
	AfxMessageBox("开始服务失败");
	return FALSE;
}

void CRecvDlg::OnAccept(void *pNotifyObj, SOCKET hSocket, BOOL &bAccept)
{
	CRecvDlg *pRecvDlg = (CRecvDlg *)pNotifyObj;
	CString strInfo;
	strInfo.Format("OnAccept-%d", hSocket);

	pRecvDlg->DispInfo(strInfo);
}

void CRecvDlg::OnAcceptErr(void *pNotifyObj, SOCKET hAccept)
{
	CRecvDlg *pRecvDlg = (CRecvDlg *)pNotifyObj;
	CString strInfo;
	strInfo.Format("OnAcceptErr-%d", hAccept);

	pRecvDlg->DispInfo(strInfo);
}

void CRecvDlg::OnOneNetMsg(void *pNotifyObj, char *Msg, int nMsgLen)
{
	CRecvDlg *pRecvDlg = (CRecvDlg *)pNotifyObj;
	CString strInfo;
	char s[10240];

	memcpy(s, Msg, nMsgLen);
	s[nMsgLen] = 0;
	strInfo = (LPCTSTR)s;

	pRecvDlg->DispInfo(strInfo);
}

void CRecvDlg::OnRecvFileStart(void *pNotifyObj, char *szPathName, BOOL &bRecv)
{
}

void CRecvDlg::OnRecvFileProgress(void *pNotifyObj, DWORD dwRecvedBytes, DWORD dwFileSize)
{
	CRecvDlg *pRecvDlg = (CRecvDlg *)pNotifyObj;
	CString strInfo;

	strInfo.Format("%d / %d", dwRecvedBytes, dwFileSize);
	pRecvDlg->DispInfo(strInfo);
}

void CRecvDlg::OnRecvFileSucc(void *pNotifyObj, char *szPathName)
{
	CRecvDlg *pRecvDlg = (CRecvDlg *)pNotifyObj;
	CString strInfo = "OnRecvFileSucc";

	pRecvDlg->DispInfo(strInfo);
}
```

```
void CRecvDlg::OnRecvFileFail(void *pNotifyObj, char *szPathName)
{
	CRecvDlg *pRecvDlg = (CRecvDlg *)pNotifyObj;
	CString strInfo = "OnRecvFileFail";

	pRecvDlg->DispInfo(strInfo);
}

void CRecvDlg::OnRecvFileCancel(void *pNotifyObj, char *szPathName)
{
	CRecvDlg *pRecvDlg = (CRecvDlg *)pNotifyObj;
	CString strInfo = "OnRecvFileCancel";

	pRecvDlg->DispInfo(strInfo);
}

void CRecvDlg::DispInfo(CString &strInfo)
{
	m_ctlInfo1.SetWindowText(strInfo);
}

void CRecvDlg::DispCnnCount(void)
{
	CString strCnnCount;

	strCnnCount.Format("%d", m_pTcpServer1->GetClientCount());
	m_ctlCnnCount.SetWindowText(strCnnCount);
}

void CRecvDlg::OnCancelRecv()
{
	m_pTcpServer1->CancelAllRecvFile();
}

void CRecvDlg::OnSocketConnect(void *pNotifyObj, SOCKET hSocket)
{
	CRecvDlg *pRecvDlg = (CRecvDlg *)pNotifyObj;
	CString strInfo;

	strInfo.Format("OnSocketConnect-%d", hSocket);
	pRecvDlg->DispInfo(strInfo);
	pRecvDlg->DispCnnCount();
}

void CRecvDlg::OnSocketDisconnect(void *pNotifyObj, SOCKET hSocket)
{
	CRecvDlg *pRecvDlg = (CRecvDlg *)pNotifyObj;
	CString strInfo;

	strInfo.Format("OnSocketDisconnect-%d", hSocket);
	pRecvDlg->DispInfo(strInfo);
	pRecvDlg->DispCnnCount();
}

void CRecvDlg::OnSocketSendErr(void *pNotifyObj, CServerClientSocket *pServerClientSocket)
{
```

```
	CRecvDlg *pRecvDlg = (CRecvDlg *)pNotifyObj;
	CString strInfo = "OnSocketSendErr";

	pRecvDlg->DispInfo(strInfo);
}

void CRecvDlg::OnSocketRecvErr(void *pNotifyObj, CServerClientSocket *pServerClientSocket)
{
	CRecvDlg *pRecvDlg = (CRecvDlg *)pNotifyObj;
	CString strInfo = "OnSocketRecvErr";

	pRecvDlg->DispInfo(strInfo);
}

void CRecvDlg::OnCloseCnn()
{
}

void CRecvDlg::OnDestroy()
{
	CDialog::OnDestroy();

	m_pTcpServer1->CloseAllServerClientSocket();
}
delete m_pTcpServer1;
m_pTcpServer1 = NULL;
```

14.多线程多文件断点续传

```
//Send.dsw
//Thead.h
// 线程对象封装
//
#ifndef _THREAD_INCLUDE_
#define _THREAD_INCLUDE_

class CThread
{
private:
	static DWORD WINAPI ThreadProc(LPVOID pVoid);
protected:
	BOOL    m_bTerminated; // 线程是否终止的标志

	virtual void Execute(void) = 0;
public:
	HANDLE  m_hThread; // 线程句柄

	CThread(void);
	~CThread(void);

	void Resume(void);
	void Terminate(void);
	HANDLE GetThreadHandle(void);
};

#endif // #ifndef _THREAD_INCLUDE_
```

```
//Thead.cpp
// 线程对象封装
//
#include "stdafx.h"
#include "Thread.h"

CThread::CThread(void)
{
	m_bTerminated = FALSE;

	DWORD dwThreadID;
	m_hThread = CreateThread(NULL, 0, ThreadProc, this, CREATE_SUSPENDED, &dwThreadID);
}

CThread::~CThread(void)
{
	CloseHandle(m_hThread);
	m_hThread = NULL;
}

DWORD CThread::ThreadProc(LPVOID pVoid)
{
	((CThread *)(pVoid))->Execute();

	return 0;
}

void CThread::Resume(void)
{
	ResumeThread(m_hThread);
}

void CThread::Terminate(void)
{
	m_bTerminated = TRUE;
}

HANDLE CThread::GetThreadHandle(void)
{
	return m_hThread;
}

m_pTcpClient = new CTcpClient(this);
m_strServerIp = "127.0.0.1";
m_nPort = 8000;
m_dwPackageSize = 1024;
m_strFileName = "d:\\a.pdf";
UpdateData(FALSE);

m_pTcpClient->SetOnSocketSendErr(OnSocketSendErr);
m_pTcpClient->SetOnSocketRecvErr(OnSocketRecvErr);
m_pTcpClient->SetOnSocketClose(OnSocketClose);
m_pTcpClient->SetOnOneNetMsg(OnOneNetMsg);
m_pTcpClient->SetOnSendFileSucc(OnSendFileSucc);
m_pTcpClient->SetOnSendFileFail(OnSendFileFail);
m_pTcpClient->SetOnSendFileRefuseRecv(OnSendFileRefuseRecv);
m_pTcpClient->SetOnSendFileCancelRecv(OnSendFileCancelRecv);
```

```
m_pTcpClient->SetOnSendFileRecvFail(OnSendFileRecvFail);
m_pTcpClient->SetOnSendFileProgress(OnSendFileProgress);
CStatic m_ctlCnnStatus;
CStatic m_ctlInfo;
DWORD   m_dwPackageSize;
CString m_strServerIp;
int             m_nPort;
CString m_strFileName;
CString m_strMsg;

void CSendDlg::OnConnect()
{
        if(!UpdateData())
                return;


        m_pTcpClient->SetAddr((char *)(LPCTSTR)m_strServerIp);
        m_pTcpClient->SetPort(m_nPort);
        m_pTcpClient->SetPackageSize(m_dwPackageSize);

        m_ctlCnnStatus.SetWindowText("请等待...");
        if(!m_pTcpClient->Connect())
                m_ctlCnnStatus.SetWindowText("连接失败!");
        else
                m_ctlCnnStatus.SetWindowText("已连接");
}

void CSendDlg::OnDisconnect()
{
        m_pTcpClient->Disconnect();
        m_ctlCnnStatus.SetWindowText("断开连接");
}

void CSendDlg::OnSendFile()
{
        if(!UpdateData())
                return;

        m_pTcpClient->SetPackageSize(m_dwPackageSize);
        if(!m_pTcpClient->SendFile((char *)(LPCTSTR)m_strFileName))
                AfxMessageBox("发生文件失败");
}

void CSendDlg::OnSendMsg(void)
{
        char s[99999];

        if(!UpdateData())
                return;


        sprintf(s, "@00000001%s", m_strMsg);
        m_pTcpClient->SendNetMsg(s, strlen(s) - 6);
}

void CSendDlg::OnSocketSendErr(void *pNotifyObj, SOCKET hSocket)
```

```
{
	CSendDlg *pSendDlg = (CSendDlg *)pNotifyObj;
	pSendDlg->m_ctlCnnStatus.SetWindowText("发送数据出错");
}

void CSendDlg::OnSocketRecvErr(void *pNotifyObj, SOCKET hSocket)
{
	CSendDlg *pSendDlg = (CSendDlg *)pNotifyObj;
	pSendDlg->m_ctlCnnStatus.SetWindowText("接收数据出错");
}

void CSendDlg::OnSocketClose(void *pNotifyObj, SOCKET hSocket)
{
	CSendDlg *pSendDlg = (CSendDlg *)pNotifyObj;
	pSendDlg->m_ctlCnnStatus.SetWindowText("断开连接");
}

void CSendDlg::OnOneNetMsg(void *pNotifyObj, char *Msg, int nMsgLen)
{
	CSendDlg *pSendDlg = (CSendDlg *)pNotifyObj;
	char s[9999];
	CString strInfo;

	strncpy(s, Msg, nMsgLen);
	s[nMsgLen] = 0;
	strInfo = s;

	pSendDlg->DispInfo(strInfo);
}

void CSendDlg::OnSendFileSucc(void *pNotifyObj, char *szPathName)
{
	CSendDlg *pSendDlg = (CSendDlg *)pNotifyObj;

	pSendDlg->DispInfo("OnSendFileSucc");
}

void CSendDlg::OnSendFileFail(void *pNotifyObj, char *szPathName)
{
	CSendDlg *pSendDlg = (CSendDlg *)pNotifyObj;

	pSendDlg->DispInfo("OnSendFileFail");
}

// 接收方拒绝接收文件
void CSendDlg::OnSendFileRefuseRecv(void *pNotifyObj, char *szPathName)
{
	CSendDlg *pSendDlg = (CSendDlg *)pNotifyObj;

	pSendDlg->DispInfo("OnSendFileRefuseRecv");
}

// 接收方拒绝文件
void CSendDlg::OnSendFileCancelRecv(void *pNotifyObj, char *szPathName)
{
	CSendDlg *pSendDlg = (CSendDlg *)pNotifyObj;
```

```
	pSendDlg->DispInfo("OnSendFileCancelRecv");
}

// 接收方取消接收
void CSendDlg::OnSendFileRecvFail(void *pNotifyObj, char *szPathName)
{
	CSendDlg *pSendDlg = (CSendDlg *)pNotifyObj;

	pSendDlg->DispInfo("OnSendFileRecvFail");
}

void CSendDlg::OnSendFileProgress(void *pNotifyObj, int nSentBytes, int nTotalBytes)
{
	CSendDlg *pSendDlg = (CSendDlg *)pNotifyObj;
	CString strInfo;

	strInfo.Format("%d / %d", nSentBytes, nTotalBytes);
	pSendDlg->DispInfo(strInfo);
}

void CSendDlg::DispInfo(CString strInfo)
{
	m_ctlInfo.SetWindowText(strInfo);
}

void CSendDlg::OnCancelSend()
{
	m_pTcpClient->CancelSendFile();
}

void CSendDlg::OnDestroy()
{
	CDialog::OnDestroy();

	m_pTcpClient->Disconnect();
}
delete m_pTcpClient;

//Recv.dsw
#include "Shlwapi.h"

// 判断文件是否存在
BOOL IsFileExists(char *pszPathName);
// 创建多层目录,成功返回TRUE,识别返回FALSE
BOOL ForceDirectories(char *pszDir);
// 扩展文件操作
BOOL DeleteFileEx(char *szPathName, BOOL bAllowUndo = FALSE);
BOOL RenameFileEx(char *szOldPathName, char *szNewPathName);
BOOL MoveFileEx(char *szSrcPathName, char *szDstPathName);
BOOL CopyFileEx(char *szSrcPathName, char *szDstPathName);
// 重新启动操作系统
BOOL RebootWindows();
// 设置程序是否在操作系统启动后自动运行
void SetAutoRun(BOOL bEnable);
BOOL ShutDownWin98();
```

```
BOOL ShutDownWinNT();

BOOL IsLegalFileName(char *szFileName);
m_pTcpServer1 = new CTcpServer(this);
m_pTcpServer1->SetBindAddr("");
m_pTcpServer1->SetPort(8000);
m_pTcpServer1->SetOnAccept(OnAccept);
m_pTcpServer1->SetOnAcceptErr(OnAcceptErr);
m_pTcpServer1->SetOnSocketConnect(OnSocketConnect);
m_pTcpServer1->SetOnSocketDisconnect(OnSocketDisconnect);
m_pTcpServer1->SetOnSocketSendErr(OnSocketSendErr);
m_pTcpServer1->SetOnSocketRecvErr(OnSocketRecvErr);
m_pTcpServer1->SetOnOneNetMsg(OnOneNetMsg);
m_pTcpServer1->SetOnRecvFileStart(OnRecvFileStart);
m_pTcpServer1->SetOnRecvFileProgress(OnRecvFileProgress);
m_pTcpServer1->SetOnRecvFileFail(OnRecvFileFail);
m_pTcpServer1->SetOnRecvFileSucc(OnRecvFileSucc);
m_pTcpServer1->SetOnRecvFileCancel(OnRecvFileCancel);
if(!m_pTcpServer1->StartAccept())
{
	AfxMessageBox("开始服务失败");
	return FALSE;
}

void CRecvDlg::OnAccept(void *pNotifyObj, SOCKET hSocket, BOOL &bAccept)
{
	CRecvDlg *pRecvDlg = (CRecvDlg *)pNotifyObj;
	CString strInfo;
	strInfo.Format("OnAccept-%d", hSocket);

	pRecvDlg->DispInfo(strInfo);
}

void CRecvDlg::OnAcceptErr(void *pNotifyObj, SOCKET hAccept)
{
	CRecvDlg *pRecvDlg = (CRecvDlg *)pNotifyObj;
	CString strInfo;
	strInfo.Format("OnAcceptErr-%d", hAccept);

	pRecvDlg->DispInfo(strInfo);
}

void CRecvDlg::OnOneNetMsg(void *pNotifyObj, char *Msg, int nMsgLen)
{
	CRecvDlg *pRecvDlg = (CRecvDlg *)pNotifyObj;
	CString strInfo;
	char s[10240];

	memcpy(s, Msg, nMsgLen);
	s[nMsgLen] = 0;
	strInfo = (LPCTSTR)s;

	pRecvDlg->DispInfo(strInfo);
}

void CRecvDlg::OnRecvFileStart(void *pNotifyObj, char *szPathName, BOOL &bRecv)
{
```

```
}

void CRecvDlg::OnRecvFileProgress(void *pNotifyObj, DWORD dwRecvedBytes, DWORD dwFileSize)
{
	CRecvDlg *pRecvDlg = (CRecvDlg *)pNotifyObj;
	CString strInfo;

	strInfo.Format("%d / %d", dwRecvedBytes, dwFileSize);
	pRecvDlg->DispInfo(strInfo);
}

void CRecvDlg::OnRecvFileSucc(void *pNotifyObj, char *szPathName)
{
	CRecvDlg *pRecvDlg = (CRecvDlg *)pNotifyObj;
	CString strInfo = "OnRecvFileSucc";

	pRecvDlg->DispInfo(strInfo);
}

void CRecvDlg::OnRecvFileFail(void *pNotifyObj, char *szPathName)
{
	CRecvDlg *pRecvDlg = (CRecvDlg *)pNotifyObj;
	CString strInfo = "OnRecvFileFail";

	pRecvDlg->DispInfo(strInfo);
}

void CRecvDlg::OnRecvFileCancel(void *pNotifyObj, char *szPathName)
{
	CRecvDlg *pRecvDlg = (CRecvDlg *)pNotifyObj;
	CString strInfo = "OnRecvFileCancel";

	pRecvDlg->DispInfo(strInfo);
}

void CRecvDlg::DispInfo(CString &strInfo)
{
	m_ctlInfo1.SetWindowText(strInfo);
}

void CRecvDlg::DispCnnCount(void)
{
	CString strCnnCount;

	strCnnCount.Format("%d", m_pTcpServer1->GetClientCount());
	m_ctlCnnCount.SetWindowText(strCnnCount);
}

void CRecvDlg::OnCancelRecv()
{
	m_pTcpServer1->CancelAllRecvFile();
}

void CRecvDlg::OnSocketConnect(void *pNotifyObj, SOCKET hSocket)
{
	CRecvDlg *pRecvDlg = (CRecvDlg *)pNotifyObj;
	CString strInfo;
```

```
	strInfo.Format("OnSocketConnect-%d", hSocket);
	pRecvDlg->DispInfo(strInfo);
	pRecvDlg->DispCnnCount();
}

void CRecvDlg::OnSocketDisconnect(void *pNotifyObj, SOCKET hSocket)
{
	CRecvDlg *pRecvDlg = (CRecvDlg *)pNotifyObj;
	CString strInfo;

	strInfo.Format("OnSocketDisconnect-%d", hSocket);
	pRecvDlg->DispInfo(strInfo);
	pRecvDlg->DispCnnCount();
}

void CRecvDlg::OnSocketSendErr(void *pNotifyObj, CServerClientSocket *pServerClientSocket)
{
	CRecvDlg *pRecvDlg = (CRecvDlg *)pNotifyObj;
	CString strInfo = "OnSocketSendErr";

	pRecvDlg->DispInfo(strInfo);
}

void CRecvDlg::OnSocketRecvErr(void *pNotifyObj, CServerClientSocket *pServerClientSocket)
{
	CRecvDlg *pRecvDlg = (CRecvDlg *)pNotifyObj;
	CString strInfo = "OnSocketRecvErr";

	pRecvDlg->DispInfo(strInfo);
}

void CRecvDlg::OnCloseCnn()
{
}

void CRecvDlg::OnDestroy()
{
	CDialog::OnDestroy();

	m_pTcpServer1->CloseAllServerClientSocket();
}
delete m_pTcpServer1;
m_pTcpServer1 = NULL;
```

15.截取屏幕

```
HBITMAP CopyScreenToBitmap(LPRECT lpRect)
//lpRect 代表选定区域
{
	HDC hScrDC, hMemDC;
	// 屏幕和内存设备描述表
	HBITMAP hBitmap, hOldBitmap;
	// 位图句柄
	int nX, nY, nX2, nY2;
	// 选定区域坐标
	int nWidth, nHeight;
```

```
	// 位图宽度和高度
	int xScrn, yScrn;
	// 屏幕分辨率

	// 确保选定区域不为空矩形
	if (IsRectEmpty(lpRect))
		return NULL;
	//为屏幕创建设备描述表
	hScrDC = CreateDC("DISPLAY", NULL, NULL, NULL);
	//为屏幕设备描述表创建兼容的内存设备描述表
	hMemDC = CreateCompatibleDC(hScrDC);
	// 获得选定区域坐标
	nX = lpRect- >left;
	nY = lpRect- >top;
	nX2 = lpRect- >right;
	nY2 = lpRect- >bottom;
	// 获得屏幕分辨率
	xScrn = GetDeviceCaps(hScrDC, HORZRES);
	yScrn = GetDeviceCaps(hScrDC, VERTRES);
	//确保选定区域是可见的
	if (nX <0)
		nX = 0;
	if (nY < 0)
		nY = 0;
	if (nX2 > xScrn)
		nX2 = xScrn;
	if (nY2 > yScrn)
		nY2 = yScrn;
	nWidth = nX2 - nX;
	nHeight = nY2 - nY;
	// 创建一个与屏幕设备描述表兼容的位图
	hBitmap = CreateCompatibleBitmap
		(hScrDC, nWidth, nHeight);
	// 把新位图选到内存设备描述表中
	hOldBitmap = SelectObject(hMemDC, hBitmap);
	// 把屏幕设备描述表拷贝到内存设备描述表中
	BitBlt(hMemDC, 0, 0, nWidth, nHeight,
		hScrDC, nX, nY, SRCCOPY);
	//得到屏幕位图的句柄
	hBitmap = SelectObject(hMemDC, hOldBitmap);
	//清除
	DeleteDC(hScrDC);
	DeleteDC(hMemDC);
	// 返回位图句柄
	return hBitmap;
}


得到屏幕位图句柄以后,我们
可以把屏幕内容粘贴到剪贴板上.
if (OpenClipboard(hWnd))
//hWnd为程序窗口句柄
{
	//清空剪贴板
	EmptyClipboard();
	//把屏幕内容粘贴到剪贴板上,
```

```
        hBitmap 为刚才的屏幕位图句柄
            SetClipboardData(CF_BITMAP, hBitmap);
        //关闭剪贴板
        CloseClipb
            oard();
}
```

我们也可以把屏幕内容以位图格式存到磁盘文件上.

```
int SaveBitmapToFile(HBITMAP hBitmap ,
                                LPSTR lpFileName) //hBitmap 为刚才的屏幕位图句柄
{ //lpFileName 为位图文件名
        HDC hDC;
        //设备描述表
        int iBits;
        //当前显示分辨率下每个像素所占字节数
        WORD wBitCount;
        //位图中每个像素所占字节数
        //定义调色板大小, 位图中像素字节大小 ,
        位图文件大小 , 写入文件字节数
            DWORD dwPaletteSize=0,
            dwBmBitsSize,
            dwDIBSize, dwWritten;
        BITMAP Bitmap;
        //位图属性结构
        BITMAPFILEHEADER bmfHdr;
        //位图文件头结构
        BITMAPINFOHEADER bi;
        //位图信息头结构
        LPBITMAPINFOHEADER lpbi;
        //指向位图信息头结构
        HANDLE fh, hDib, hPal,hOldPal=NULL;
        //定义文件,分配内存句柄,调色板句柄

        //计算位图文件每个像素所占字节数
        hDC = CreateDC("DISPLAY",NULL,NULL,NULL);
        iBits = GetDeviceCaps(hDC, BITSPIXEL) *
            GetDeviceCaps(hDC, PLANES);
        DeleteDC(hDC);
        if (iBits < = 1)
            wBitCount = 1;
        else if (iBits < = 4)
            wBitCount = 4;
        else if (iBits < = 8)
            wBitCount = 8;
        else if (iBits < = 24)
            wBitCount = 24;
        //计算调色板大小
        if (wBitCount < = 8)
            dwPaletteSize = (1 < < wBitCount) *
            sizeof(RGBQUAD);


        //设置位图信息头结构
        GetObject(hBitmap, sizeof(BITMAP), (LPSTR)&Bitmap);
        bi.biSize = sizeof(BITMAPINFOHEADER);
        bi.biWidth = Bitmap.bmWidth;
```

```
bi.biHeight = Bitmap.bmHeight;
bi.biPlanes = 1;
bi.biBitCount = wBitCount;
bi.biCompression = BI_RGB;
bi.biSi
        zeImage = 0;
bi.biXPelsPerMeter = 0;
bi.biYPelsPerMeter = 0;
bi.biClrUsed = 0;
bi.biClrImportant = 0;

dwBmBitsSize = ((Bitmap.bmWidth *
        wBitCount+31)/32)* 4
        *Bitmap.bmHeight ;
//为位图内容分配内存
hDib = GlobalAlloc(GHND,dwBmBitsSize+
        dwPaletteSize+sizeof(BITMAPINFOHEADER));
lpbi = (LPBITMAPINFOHEADER)GlobalLock(hDib);
*lpbi = bi;
// 处理调色板
hPal = GetStockObject(DEFAULT_PALETTE);
if (hPal)
{
        hDC = GetDC(NULL);
        hOldPal = SelectPalette(hDC, hPal, FALSE);
        RealizePalette(hDC);
}
// 获取该调色板下新的像素值
GetDIBits(hDC, hBitmap, 0, (UINT) Bitmap.bmHeight,
        (LPSTR)lpbi + sizeof(BITMAPINFOHEADER)
        +dwPaletteSize,
        (BITMAPINFOHEADER *)
        lpbi, DIB_RGB_COLORS);
//恢复调色板
if (hOldPal)
{
        SelectPalette(hDC, hOldPal, TRUE);
        RealizePalette(hDC);
        ReleaseDC(NULL, hDC);
}
//创建位图文件
fh = CreateFile(lpFileName, GENERIC_WRITE,
        0, NULL, CREATE_ALWAYS,
        FILE_ATTRIBUTE_NORMAL   FILE_
        FLAG_SEQUENTIAL_SCAN, NULL);
if (fh == INVALID_HANDLE_VALUE)
        return FALSE;
// 设置位图文件头
bmfHdr.bfType = 0x4D42; // "BM"
dwDIBSize = sizeof(BITMAPFILEHEADER)
        + sizeof(BITMAPINFOHEADER)
        + dwPaletteSize + dwBmBitsSize;
bmfHdr.bfSize = dwDIBSize;
bmfHdr.bfReserved1 = 0;
bmfHdr.bfReserved2 = 0;
bmfHdr.bfOffBits = (DWORD)sizeof
```

```
            (BITMAPFILEHEADER)
            + (DWORD)sizeof(BITMAPINFOHEADER)
            + dwPaletteSize;
        // 写入位图文件头
        WriteFile(fh, (LPSTR)&bmfHdr, sizeof
            (BITMAPFILEHEADER), &dwWritten, NULL);
        // 写入位图文件其余内容
        WriteFile(fh, (LPSTR)lpbi, dwDIBSize,
            &dwWritten, NULL);
        //清除
        GlobalUnlock(hDib);
        GlobalFree(hDib);
        CloseHandle(fh);
}
```

16.聊天室服务器端逻辑

一、服务器端所声明的类

```
class CCSocketDlg : public CDialog
{
        // Construction
public:
        CCSocketDlg(CWnd* pParent = NULL); // standard constructor
        ~CCSocketDlg();
        // Dialog Data
        //{{AFX_DATA(CCSocketDlg)
        enum { IDD = IDD_CSOCKET_DIALOG };
        CButton m_button;
        CListCtrl m_list;
        CEdit m_edit;
        //}}AFX_DATA
        // ClassWizard generated virtual function overrides
        //{{AFX_VIRTUAL(CCSocketDlg)
protected:
        virtual void DoDataExchange(CDataExchange* pDX); // DDX/DDV support
        //}}AFX_VIRTUAL
        // Implementation
protected:
        HICON m_hIcon;
        // Generated message map functions
        //{{AFX_MSG(CCSocketDlg)
        virtual BOOL OnInitDialog();
        afx_msg void OnSysCommand(UINT nID, LPARAM lParam);
        afx_msg void OnPaint();
        afx_msg HCURSOR OnQueryDragIcon();
        virtual void OnOK();
        afx_msg void OnButton1();
        //}}AFX_MSG
        DECLARE_MESSAGE_MAP()
public:
        WSADATA      wsaData;
        SOCKET clisock;
        SOCKET  sListen, sAccept;
        int addlen;
        int count,s;
        int getcount();
        void sendtoall(SOCKET,char*);
```

```
	struct sockaddr_in ser, cli; //服务器和客户的地址
	int iLen; //客户地址长度
	int iSend;//发送的数据长度
	int flag;//标志位
	char  buf[1000];//要发送给客户的信息
	void CRS();
};
UINT thread(LPVOID);
//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ will insert additional declarations immediately before the previous line.

#endif // !defined(AFX_CSOCKETDLG_H__2DFDFAF0_3473_43E6_A5CB_DBB8531B370E__INCLUDED_)
```

二、服务器端

```
// CSocketDlg.cpp : implementation file
//服务器端

#include "stdafx.h"
#include "CSocket.h"
#include "CSocketDlg.h"
#include <io.h>

class CAboutDlg : public CDialog
{
public:
	CAboutDlg();

	// Dialog Data
	//{{AFX_DATA(CAboutDlg)
	enum { IDD = IDD_ABOUTBOX };
	//}}AFX_DATA

	// ClassWizard generated virtual function overrides
	//{{AFX_VIRTUAL(CAboutDlg)
protected:
	virtual void DoDataExchange(CDataExchange* pDX);    // DDX/DDV support
	//}}AFX_VIRTUAL

	// Implementation
protected:
	//{{AFX_MSG(CAboutDlg)
	//}}AFX_MSG
	DECLARE_MESSAGE_MAP()
};

CAboutDlg::CAboutDlg() : CDialog(CAboutDlg::IDD)
{
	//{{AFX_DATA_INIT(CAboutDlg)
	//}}AFX_DATA_INIT
}

void CAboutDlg::DoDataExchange(CDataExchange* pDX)
{
	CDialog::DoDataExchange(pDX);
	//{{AFX_DATA_MAP(CAboutDlg)
	//}}AFX_DATA_MAP
}
```

```
BEGIN_MESSAGE_MAP(CAboutDlg, CDialog)
	//{{AFX_MSG_MAP(CAboutDlg)
	// No message handlers
	//}}AFX_MSG_MAP
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CCSocketDlg dialog

CCSocketDlg::CCSocketDlg(CWnd* pParent /*=NULL*/)
: CDialog(CCSocketDlg::IDD, pParent)
{
	//{{AFX_DATA_INIT(CCSocketDlg)
	// NOTE: the ClassWizard will add member initialization here
	//}}AFX_DATA_INIT
	// Note that LoadIcon does not require a subsequent DestroyIcon in Win32
	m_hIcon = AfxGetApp()->LoadIcon(IDR_MAINFRAME);
}

void CCSocketDlg::DoDataExchange(CDataExchange* pDX)
{
	CDialog::DoDataExchange(pDX);
	//{{AFX_DATA_MAP(CCSocketDlg)
	DDX_Control(pDX, IDC_BUTTON1, m_button);
	DDX_Control(pDX, IDC_LIST1, m_list);
	DDX_Control(pDX, IDC_EDIT1, m_edit);
	//}}AFX_DATA_MAP
}

BEGIN_MESSAGE_MAP(CCSocketDlg, CDialog)
	//{{AFX_MSG_MAP(CCSocketDlg)
	ON_WM_SYSCOMMAND()
	ON_WM_PAINT()
	ON_WM_QUERYDRAGICON()
	ON_BN_CLICKED(IDC_BUTTON1, OnButton1)
	//}}AFX_MSG_MAP
END_MESSAGE_MAP()
/////////////////////////////////////////////////////////////////////////////
// CCSocketDlg message handlers
//初始化对话框
BOOL CCSocketDlg::OnInitDialog()
{
	CDialog::OnInitDialog();

	// Add "About..." menu item to system menu.

	// IDM_ABOUTBOX must be in the system command range.
	ASSERT((IDM_ABOUTBOX & 0xFFF0) == IDM_ABOUTBOX);
	ASSERT(IDM_ABOUTBOX < 0xF000);

	CMenu* pSysMenu = GetSystemMenu(FALSE);

	if (pSysMenu != NULL)
	{
		CString strAboutMenu;
		strAboutMenu.LoadString(IDS_ABOUTBOX);
```

```
	if (!strAboutMenu.IsEmpty())
	{
		pSysMenu->AppendMenu(MF_SEPARATOR);
		pSysMenu->AppendMenu(MF_STRING, IDM_ABOUTBOX, strAboutMenu);
	}
}

// Set the icon for this dialog.  The framework does this automatically
//  when the application's main window is not a dialog
SetIcon(m_hIcon, TRUE); // Set big icon
SetIcon(m_hIcon, FALSE); // Set small icon
// TODO: Add extra initialization here
int count,s=1;
// char buff[100];
CDialog a;
CCSocketDlg *dlg=(CCSocketDlg*)AfxGetApp()->GetMainWnd();
count=0;
m_list.InsertColumn(0,"消息");
m_list.SetColumnWidth(0,435);
m_edit.SetLimitText(99);
dlg->sAccept=NULL;
//设定地址
dlg->ser.sin_addr.s_addr=htonl(INADDR_ANY);
dlg->ser.sin_family=AF_INET;
dlg->ser.sin_port=htons(5000);
addlen=sizeof(dlg->ser);
m_button.EnableWindow(FALSE);

//创建服务器端的套接口
dlg->sListen=socket(AF_INET,SOCK_STREAM,0);
if (dlg->sListen==INVALID_SOCKET)
{
	m_edit.SetWindowText("创建套接口失败");
	return FALSE;
}

//绑定
if (bind(dlg->sListen,(SOCKADDR*)&(dlg->ser),addlen))=SOCKET_ERROR)
{
	closesocket(dlg->sListen);
	m_edit.SetWindowText("绑定错误");
	return FALSE;
}
else{
	m_edit.SetWindowText("服务器创建成功");

	//开始侦听
	if (listen(dlg->sListen,5)==SOCKET_ERROR)
	{
		m_edit.SetWindowText("侦听失败");
		return FALSE;
	}

	CRS();
}
return TRUE;  // return TRUE  unless you set the focus to a control
```

```
}
void CCSocketDlg::OnSysCommand(UINT nID, LPARAM lParam)
{
	if ((nID & 0xFFF0) == IDM_ABOUTBOX)
	{
		CAboutDlg dlgAbout;
		dlgAbout.DoModal();
	}
	else
	{
		CDialog::OnSysCommand(nID, lParam);
	}
}
// If you add a minimize button to your dialog, you will need the code below
//  to draw the icon.  For MFC applications using the document/view model,
//  this is automatically done for you by the framework.
void CCSocketDlg::OnPaint()
{
	if (IsIconic())
	{
		CPaintDC dc(this); // device context for painting
		SendMessage(WM_ICONERASEBKGND, (WPARAM) dc.GetSafeHdc(), 0);
		// Center icon in client rectangle
		int cxIcon = GetSystemMetrics(SM_CXICON);
		int cyIcon = GetSystemMetrics(SM_CYICON);
		CRect rect;
		GetClientRect(&rect);
		int x = (rect.Width() - cxIcon + 1) / 2;
		int y = (rect.Height() - cyIcon + 1) / 2;
		// Draw the icon
		dc.DrawIcon(x, y, m_hIcon);
	}
	else
	{
		CDialog::OnPaint();
	}
}
// The system calls this to obtain the cursor to display while the user drags
//  the minimized window.
HCURSOR CCSocketDlg::OnQueryDragIcon()
{
	return (HCURSOR) m_hIcon;
}
void CCSocketDlg::OnOK()
{
	// CDialog::OnOK();
}
//发送数据
void CCSocketDlg::OnButton1()
{
	char buff[100];
	m_edit.GetWindowText(buff,99);
	m_edit.SetWindowText("");
	m_list.InsertItem(count++,buff);
	//m_list.Scroll(size);
	if (sAccept!=NULL)
		//发送
```

```
            send(sAccept,buff,100,0);
}
CCSocketDlg::~CCSocketDlg()
{
      if (sAccept!=NULL)
            send(sAccept,"Disconnected",100,0);
}
void CCSocketDlg::CRS()
{
      char buff[100];
      CCSocketDlg *dlg=(CCSocketDlg*)AfxGetApp()->GetMainWnd();
      //初始化客户地址长度参数
      iLen=sizeof(dlg->cli);
      //进入一个无限循环,等待客户的连接请求
      while(1)
      {
            dlg->sAccept=accept(dlg->sListen,(sockaddr*)&(dlg->ser),&(dlg->iLen));
            if (dlg->sAccept==INVALID_SOCKET)
            { dlg->m_edit.SetWindowText("Error accept");}
            dlg->m_list.InsertItem(dlg->count++,"连接成功");
            char *ctime( const time_t *timer );
            time_t ltime;
            time(<ime);
            dlg->m_list.InsertItem(dlg->count++,ctime( <ime ) );
            s=recv(dlg->sAccept,buff,100,0);
            dlg->SetForegroundWindow();
            if (s!=SOCKET_ERROR)
            {
                  dlg->m_list.InsertItem(dlg->count++,buff);
                  dlg->m_list.InsertItem(dlg->count++,ctime( <ime ) );

                  if (dlg->sAccept!=NULL)
                        //发送
                        send(dlg->sAccept,buff,100,0);
                  //dlg->sendtoall(dlg->sAccept,buff);
                  closesocket(dlg->sAccept);
            }
      }//end While
      closesocket(dlg->sListen);
      WSACleanup();
}
```

17.聊天室客户端逻辑

18.克隆对象

```
class Test
{
public:
      Test(int temp)
      {
            p1=temp;
      }
      Test(Test &c_t)//这里就是自定义的拷贝构造函数
      {
            cout<<"进入copy构造函数"<<endl;
            p1=c_t.p1;//这句如果去掉就不能完成复制工作了,此句复制过程的核心语句
```

```
    }
public:
    int p1;
};

void main()
{
    Test a(99);
    Test b=a;
    cout<<b.p1;
    cin.get();
}
//=========================================
#include <iostream>
using namespace std;
class Internet
{
public:
    Internet(char *name,char *address)
    {
        cout<<"载入构造函数"<<endl;
        strcpy(Internet::name,name);
        strcpy(Internet::address,address);
        cname=new char[strlen(name)+1];
        if(cname!=NULL)
        {
            strcpy(Internet::cname,name);
        }
    }
    Internet(Internet &temp)
    {
        cout<<"载入COPY构造函数"<<endl;
        strcpy(Internet::name,temp.name);
        strcpy(Internet::address,temp.address);
        cname=new char[strlen(name)+1];//这里注意,深拷贝的体现!
        if(cname!=NULL)
        {
            strcpy(Internet::cname,name);
        }
    }
    ~Internet()
    {
        cout<<"载入析构函数!";
        delete[] cname;
        cin.get();
    }
    void show();
protected:
    char name[20];
    char address[30];
    char *cname;
};
void Internet::show()
{
    cout<<name<<":"<<address<<cname<<endl;
}
```

```
void test(Internet ts)
{
	cout<<"载入test函数"<<endl;
}
void main()
{
	Internet a("中国软件开发实验室","www.cndev-lab.com");
	Internet b = a;
	b.show();
	test(b);
}
/*
RUMTIME_CLASS

运行时动态识别
RTTI
class base
{
virtual base* clone() = 0;
}
class A: public base
{
virtual base* clone() { return new A;}
}
class B: public base
{
virtual base* clone() { return new B;}
}

int main()
{
base* p1 = new A;
base* p2 = p1-> clone();
return 0;
}
*/

19.XML属性文件解析
/*
#include <string>
using namespace std;
*/
char sRead[5192];
CFile mFile(_T(%%1),CFile::modeRead);
mFile.Read(sRead,5192);
if(sRead!=null)
{
	string tmp;
	while(sRead!=null)
	{
		tmp.append(sRead);
		mFile.Read(sRead,5192);
	}
	//%%2="Logs" //%%4="ID" //%%6="Content"
	//%%3="Log" //%%5="Time"
	//%%7 code %%8 time %%9 content
```

```
	string target(%%7),globalTag("<"+%%2+">");
	string propTag1("<"+%%5+">",endTag1("</"+%%5+">");
	string propTag2("<"+%%6+">",endTag1("</"+%%6+">");
	int offset=tmp.find_first_of(globalTag);
	while(offset)
	{
		offset=tmp.find_first_of(globalTag);
		string description;
		tmp.copy(description.begin(),tmp.find_first_of("\"",offset+1)-offset);
		if(target.compare(description)==0)
		{
			string prop,prop2;
			offset=tmp.find_first_of(propTag1,offset)+strlen(%%5)+2;
			tmp.copy(prop.begin(),tmp.find_first_of(endTag1,offset)-offset,offset);
			offset=tmp.find_first_of(propTag2,offset)+strlen(%%6)+2;
			tmp.copy(prop2.begin(),tmp.find_first_of(endTag2,offset)-offset,offset);
			CString %%8(prop),%%9(prop2);
			%%10
				return 0;
		}
	}
}
else
return -1;
```

20.XML属性文件构造

```
/*
#include <string>
using namespace std;
*/
char sRead[5192];
string description;
CFile mFile(_T(%%1),CFile::modeRead);
mFile.Read(sRead,5192);
int no;
if(sRead!=null)
{
	string tmp;
	while(sRead!=null)
	{
		tmp.append(sRead);
		mFile.Read(sRead,5192);
	}
	//%%2="Logs" //%%4="ID" //%%6="Content"
	//%%3="Log" //%%5="Time"
	//%%7 code %%8 time %%9 content
	int offset=tmp.find_last_of("<"+%%3+" "+%%4)+strlen(%%3) +strlen(%%4)+4;
	tmp.copy(description.begin(),tmp.find_last_of("\"><"+%%5)- offset,offset);
	bo=atoi(description.c_str())+1;
	mFile.Close();
	tmp.insert(tmp.find_last_of("</"+%%2+">"),"<"+%%3+"
"+%%4+"=\""+itoa(no)+"\"><"+%%5+">"+%%8+"</"+%%5+"><"+%%6+">"+%%9+"</"+%%6+"
>");
	CFile file(_T(%%1),CFile::modeWrite);
	file.Write(tmp.c_str()):
	file.Flush();
	file.Close();
```

```
}
else
{
	CFile file(_T(%%1),CFile::modeWrite|CFile::modeCreate);
	file.Write("<?xml version=\"1.0\" encoding=\"gb2312\"?><"+%%2+"><"+%%3+"
"+%%4+"=\"0\"><"+%%5+">"+%%8+"</"+%%5+"><"+%%6+">"+%%9+"</"+%%6+"></"+%%3
+"></"+%%2+">");
	file.Flush();
	file.Close();
}
```

21.XML文件节点遍历操作

22.XML文件节点遍历查找

23.多线程端口监听

```
/*
#include <winsock2.h>
#pragma comment(lib,"WS2_32.lib")
*/
DWORD WINAPI ClientThread(LPVOID lpParam);

WORD wVersionRequested;
DWORD ret;
WSADATA wsaData;
BOOL val;
SOCKADDR_IN saddr;
SOCKADDR_IN scaddr;
int err;
SOCKET s;
SOCKET sc;
int caddsize;
HANDLE mt;
DWORD tid;

wVersionRequested = MAKEWORD( 2, 2 );
err = WSAStartup( wVersionRequested, &wsaData );
if ( err != 0 ) {
	printf("error!WSAStartup failed!");
	return -1;
}
saddr.sin_family = AF_INET;

//截听虽然也可以将地址指定为INADDR_ANY,但是要不能影响正常应用情况下,应该指定具体的IP,留
下127.0.0.1给正常的服务应用,然后利用这个地址进行转发,就可以不影响对方正常应用了
saddr.sin_addr.s_addr = inet_addr(argv[1]);
saddr.sin_port = htons(80);
if((s=socket(AF_INET,SOCK_STREAM,IPPROTO_TCP))==SOCKET_ERROR)
{
	printf("error!socket failed!");
	return -1;
}
val = TRUE;
//SO_REUSEADDR选项就是可以实现端口重绑定的
```

```
if(setsockopt(s,SOL_SOCKET,SO_REUSEADDR,(char *)&val,sizeof(val))!=0)
{
	printf("error!setsockopt failed!");
	return -1;
}
//如果指定了SO_EXCLUSIVEADDRUSE,就不会绑定成功,返回无权限的错误代码;
//如果是想通过重利用端口达到隐藏的目的,就可以动态的测试当前已绑定的端口哪个可以成功,就说明具备这个漏洞,然后动态利用端口使得更隐蔽
//其实UDP端口一样可以这样重绑定利用,这儿主要是以TELNET服务为例子进行攻击
if(bind(s,(SOCKADDR *)&saddr,sizeof(saddr))==SOCKET_ERROR)
{
	ret=GetLastError();
	printf("error!bind failed!");
	return -1;
}
listen(s,2);
while(1)
{
	caddsize = sizeof(scaddr);
	//接受连接请求
	sc = accept(s,(struct sockaddr *)&scaddr,&caddsize);
	if(sc!=INVALID_SOCKET)
	{
		mt = CreateThread(NULL,0,ClientThread,(LPVOID)sc,0,&tid);
		if(mt==NULL)
		{
			printf("Thread Creat Failed!");
			break;
		}
	}
	CloseHandle(mt);
}
closesocket(s);
WSACleanup();
return 0;
```

24.多线程端口扫描

```
/*
#include <winsock2.h>
#pragma comment(lib,"WS2_32.lib")
*/
port_segment port;
struct sockaddr_in dest_addr;

/* copy the struct to port */
memcpy( &port, arg, sizeof(struct port_segment) );

memset( &dest_addr, 0, sizeof(struct sockaddr_in) );
dest_addr.sin_family = AF_INET;
dest_addr.sin_addr.s_addr = port.dest.s_addr;

for ( int i = port.min_port; i <= port.max_port; ++i ) {
	dest_addr.sin_port = htons( i );

	/* do the scan with every port */
	if ( do_scan(dest_addr) < 0 )
```

```
            continue;
}

return NULL;
}
pthread_t *thread;
struct in_addr dest_ip[ IP_NUM ]; // IP_NUM ip address


if ( argc < 2 ) {
        fprintf( stderr, "usage: ./scan [ip1] [ip2] .. [ip5]\n" );
        exit ( EXIT_FAILURE );
}

/* copy all the ip address into dest_ip */
for ( int i = 1; i < argc; ++i ) {
        if ( inet_aton(argv[i], &dest_ip[i - 1]) == 0 ) {
                fprintf( stderr, "invalid ip address.\n" );
                exit ( EXIT_FAILURE );
        }
}

/* malloc THREAD_NUM thread */
thread = ( pthread_t * )malloc( THREAD_NUM * sizeof(pthread_t) );

for ( int j = 0; j < argc - 1; ++j ) {
        for ( int i = 0; i < THREAD_NUM; ++i ) {
                port_segment port;
                port.dest = dest_ip[ j ];
                port.min_port = i * SEG_LEN + 1;

                /* the last segment */
                if ( i == (THREAD_NUM - 1) )
                        port.max_port = MAX_PORT;
                else
                        port.max_port = port.min_port + SEG_LEN - 1;

                /* create threads to scan the ports */
                if ( pthread_create(&thread[i], NULL, scan, (void *)&port) != 0 )
                        my_error( "pthread_create failed" );

                /* waiting for the sub threads exit */
                pthread_join( thread[i], NULL );
        }
}

/* free the memory */
free( thread );
```

25.发送带附件的邮件

```
/*
#include <winsock2.h>
#include <string.h>
#include <stdio.h>
#pragma comment(lib,"WS2_32.lib")
*/
const int  BASE64_MAXLINE = 76;
```

```
const char EOL[] = "\r\n";
const char BASE64_TAB[] = "ABCDEFGHIJKLMNOPQRSTUVWXYZ"
"abcdefghijklmnopqrstuvwxyz0123456789+/";
const char HEADER[] =
"HELO support.com\r\n"
//"AUTH LOGIN\r\n" //+ BASE64 USER + BASE64 PASS
"MAIL FROM: chinansl@support.com\r\n"
"RCPT TO: shadowstar@support.com\r\n"
"DATA\r\n"
"FROM: chinansl@support.com\r\n"
"TO: shadowstar@support.com\r\n"
"SUBJECT: this is a test\r\n"
"Date: 2002-5-14\r\n"
"X-Mailer: shadowstar""s mailer\r\n"
"MIME-Version: 1.0\r\n"
"Content-type: multipart/mixed; boundary=\"#BOUNDARY#\"\r\n"
//"Content-Type: text/plain; charset=gb2312\r\n"
"\r\n";
const char CONTENT[] =
"\r\n--#BOUNDARY#\r\n"
"Content-Type: text/plain; charset=gb2312\r\n"
"Content-Transfer-Encoding: quoted-printable\r\n"
"\r\n"
"/*************************************************************"
" *  smtp.cpp - Use SMTP to send an eMail with an Attachment and verify  *"
" *  Copyright (C) 2001-2002 by ShadowStar.                              *"
" *  Use and modify freely.                                               *"
" *  http://shadowstar.126.com/                                           *"
" *************************************************************"
" */\r\n"
"\r\n";
const char ATT_HEADER[] =
"\r\n--#BOUNDARY#\r\n"
"Content-Type: application/octet-stream; name=smtp.exe\r\n"
"Content-Disposition: attachment; filename=smtp.exe\r\n"
"Content-Transfer-Encoding: base64\r\n"
"\r\n";

//---------------------------------------------------------------------------
int ANSIToBase64(const char *szInANSI, int nInLen, char *szOutBase64, int nOutLen);

WSADATA wsaData;
int  SockFD;
struct sockaddr_in ServAddr;
char  buf[0x100];
int     x;
FILE    *fp;
char    *aatt = new char[0x400000];
char *batt = new char[0x555556];

WSAStartup(MAKEWORD(2,2), &wsaData);

LPHOSTENT pHost = gethostbyname("172.16.234.111");
SockFD = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP);
ServAddr.sin_family = AF_INET;
ServAddr.sin_addr.s_addr = *(ULONG *)pHost->h_addr_list[0];
ServAddr.sin_port = htons(25);
```

```
connect(SockFD, (struct sockaddr *)&ServAddr, sizeof(ServAddr));
//send HEADER
send(SockFD, HEADER, strlen(HEADER), 0);
//send CONTENT
send(SockFD, CONTENT, strlen(CONTENT), 0);
//send ATT_HEADER
send(SockFD, ATT_HEADER, strlen(ATT_HEADER), 0);
//read attachment
fp = fopen(argv[0], "rb");
fseek(fp, 0, 2);
x = ftell(fp);
if (x > 0x400000)
x = 0;
rewind(fp);
fread(aatt, x, 1, fp);
fclose(fp);
x = ANSIToBase64(aatt, x, batt, 0x555556);
//send base64 attachment
send(SockFD, batt, x, 0);

send(SockFD, ".\r\n", strlen(".\r\n"), 0);   //end
send(SockFD, "QUIT\r\n", strlen("QUIT\r\n"), 0); //quit

closesocket(SockFD);
WSACleanup();

delete []aatt;
delete []batt;
return 0;
}
//---------------------------------------------------------------------------
int ANSIToBase64(const char *szInANSI, int nInLen, char *szOutBase64, int nOutLen)
{
	//Input Parameter validation
	if ((szInANSI == NULL) || (nInLen == 0) || (szOutBase64 == NULL) || (nOutLen == 0))
		return 0;
	if (nOutLen < (nInLen*4/3 + 1 + nInLen*4/3/BASE64_MAXLINE*2 + 1 + 4))
		return 0;

	//Set up the parameters prior to the main encoding loop
	int nInPos  = 0;
	int nOutPos = 0;
	int nLineLen = 0;
	int c1, c2, c3;
	int i;

	// Get three characters at a time from the input buffer and encode them
	for (i=0; i<nInLen/3; ++i)
	{
		//Get the next 2 characters
		c1 = szInANSI[nInPos++] & 0xFF;
		c2 = szInANSI[nInPos++] & 0xFF;
		c3 = szInANSI[nInPos++] & 0xFF;

		//Encode into the 4 6 bit characters
		szOutBase64[nOutPos++] = BASE64_TAB[c1 >> 2];
```

```
            szOutBase64[nOutPos++] = BASE64_TAB[((c1 << 4) | (c2 >> 4)) & 0x3F];
            szOutBase64[nOutPos++] = BASE64_TAB[((c2 << 2) | (c3 >> 6)) & 0x3F];
            szOutBase64[nOutPos++] = BASE64_TAB[c3 & 0x3F];
            nLineLen += 4;

            //Handle the case where we have gone over the max line boundary
            if (nLineLen > BASE64_MAXLINE - 4)
            {
                szOutBase64[nOutPos++] = EOL[0];
                szOutBase64[nOutPos++] = EOL[1];
                nLineLen = 0;
            }
        }

        // Encode the remaining one or two characters in the input buffer
        switch (nInLen % 3)
        {
        case 0:
            {
                szOutBase64[nOutPos++] = EOL[0];
                szOutBase64[nOutPos++] = EOL[1];
                break;
            }
        case 1:
            {
                c1 = szInANSI[nInPos] & 0xFF;
                szOutBase64[nOutPos++] = BASE64_TAB[(c1 & 0xFC) >> 2];
                szOutBase64[nOutPos++] = BASE64_TAB[((c1 & 0x03) << 4)];
                szOutBase64[nOutPos++] = ""="";
                szOutBase64[nOutPos++] = ""="";
                szOutBase64[nOutPos++] = EOL[0];
                szOutBase64[nOutPos++] = EOL[1];
                break;
            }
        case 2:
            {
                c1 = szInANSI[nInPos++] & 0xFF;
                c2 = szInANSI[nInPos] & 0xFF;
                szOutBase64[nOutPos++] = BASE64_TAB[(c1 & 0xFC) >> 2];
                szOutBase64[nOutPos++] = BASE64_TAB[((c1 & 0x03) << 4) | ((c2 & 0xF0) >> 4)];
                szOutBase64[nOutPos++] = BASE64_TAB[((c2 & 0x0F) << 2)];
                szOutBase64[nOutPos++] = ""="";
                szOutBase64[nOutPos++] = EOL[0];
                szOutBase64[nOutPos++] = EOL[1];
                break;
            }
        default:
            {
                return 0;
            }
        }
        szOutBase64[nOutPos] = 0;
        return nOutPos;
    }
```

26.接收带附件的邮件

```
/*
```

```
#include <winsock2.h>
#pragma comment(lib,"WS2_32.lib")
*/
try{
	CComPtr objMail;

	HRESULT hr;
	// make sure the DLL is registered
	hr = objMail.CoCreateInstance(CLSID_Mail);
	if(SUCCEEDED(hr))
	{
		if(hr== S_OK)
		{
			// profile name is compulsory, this is the outlook profile,
			// i used "outlook express" as configuring it is easier than
			// "MS outlook" make sure to specify the correct sender's address
			// for this profile and make sure that outlook express is
			//the default email client.

			if(m_strProfile.IsEmpty())
			{
				AfxMessageBox("Please specify email profile name ");
				return;
			}

			if(m_strTo.IsEmpty())
			{
				AfxMessageBox("Please specify recipient's email address ");
				return;
			}
			// by default, it's TestProfile, assumes that a profile with this
			//name exists in outlook
			hr= objMail->put_strProfileName((_bstr_t)m_strProfile);

			hr = objMail->put_strSubject((_bstr_t)m_strSubject);

			// this is the email or set of email addresses (separated by ,)
			// which is actually used to send email
			hr = objMail->put_strEmailAddress((_bstr_t)m_strTo);

			// recipient is just to show the display name
			hr = objMail->put_strRecipient((_bstr_t)m_strTo);

			hr = objMail->put_strAttachmentFilePath((_bstr_t)m_strAttachment);

			hr = objMail->put_strMessage((_bstr_t)m_strMessage);

			hr= objMail->Send();

			if(hr!=S_OK)
				AfxMessageBox("Error, make sure the info is correct");
		}//if
	} //if
} // try
catch(...)
{
	AfxMessageBox("Error, make sure specified info is correct");
```

```
}

27.Ping
/(
#include <windows.h>
#include <winsock2.h>
#include <iphlpapi.h>
#pragma comment (lib, "ws2_32.lib" )
#pragma comment (lib, "Iphlpapi.lib")
 */
 DWORD WINAPI PingThread(LPVOID lParam)
{
	int n = (int)(INT_PTR)lParam;
	IPAddr ip = inet_addr(%%1) + (n << 24); //"192.168.0.0"
	BYTE mac[8];
	ULONG len = sizeof(mac);
	if (SendARP(ip, 0, (PULONG)mac, &len) == NO_ERROR)
	{
		//printf("192.168.0.%d : %02X-%02X-%02X-%02X-%02X-%02X\n", n, mac[0], mac[1],
mac[2], mac[3], mac[4], mac[5]);
	}
	return 0;
}

HANDLE h[255];
for (int i=1; i<255; i++)
{
	h[i] = CreateThread(NULL, 0, PingThread, (PVOID)(INT_PTR)i, 0, NULL);
}
for (int j=1; j<255; j++)
{
	WaitForSingleObject(h[j], -1);
	CloseHandle(h[j]);
}

28.调用Web Service
/*
#include <stdio.h>
#import "msxml4.dll"
using namespace MSXML2;
#import "C:\Program Files\Common Files\MSSoap\Binaries\mssoap30.dll" \
exclude("IStream", "IErrorInfo", "ISequentialStream", "_LARGE_INTEGER", \
"_ULARGE_INTEGER", "tagSTATSTG", "_FILETIME")
using namespace MSSOAPLib30;
*/
CoInitialize(NULL);
ISoapSerializerPtr Serializer;
ISoapReaderPtr Reader;
ISoapConnectorPtr Connector;
// Connect to the service.
Connector.CreateInstance(__uuidof(HttpConnector30));
Connector->Property["EndPointURL"] =
"http://MyServer/Soap3DocSamples/DocSample1/Server/DocSample1.wsdl";
Connector->Connect();
// Begin the message.
//Connector->Property["SoapAction"] = "uri:AddNumbers";
Connector->Property["SoapAction"] = "http://tempuri.org/DocSample1/action/Sample1.AddNumbers";
```

```
Connector->BeginMessage();
// Create the SoapSerializer object.
Serializer.CreateInstance(__uuidof(SoapSerializer30));
// Connect the serializer object to the input stream of the connector object.
Serializer->Init(_variant_t((IUnknown*)Connector->InputStream));
// Build the SOAP Message.
Serializer->StartEnvelope("","","");
Serializer->StartBody("");
Serializer->StartElement("AddNumbers","http://tempuri.org/DocSample1/message/","","");
Serializer->StartElement("NumberOne","","","");
Serializer->WriteString("5");
Serializer->EndElement()
Serializer->StartElement("NumberTwo","","","");
Serializer->WriteString("10");
Serializer->EndElement();
Serializer->EndElement();
Serializer->EndBody();
Serializer->EndEnvelope();
// Send the message to the XML Web service.
Connector->EndMessage();
// Read the response.
Reader.CreateInstance(__uuidof(SoapReader30));
// Connect the reader to the output stream of the connector object.
Reader->Load(_variant_t((IUnknown*)Connector->OutputStream), "");
// Display the result.
printf("Answer: %s\n", (const char*)Reader->RpcResult->text);
CoUninitialize();
```

29.HTTP代理服务器

```
//http://sourceforge.net/projects/csproxy/
/*
Copyright (C) 2009 Chen Kaihui.
Name: main.c
Author: Chen Kaihui
E_mail: bmwthink-bd@yahoo.com.cn
Date: 01-12-08 16:58
Description: Main file. start http proxy server.

This program is free software; you can redistribute it and/or
modify it under the terms of the GNU General Public License
as published by the Free Software Foundation.
csproxyv1.3
*/
/*
#include <stdio.h>
#include <stdlib.h>
#include "server.h"
#define PORT 8080
*/
start_server(PORT);
```

30.创建启动线程

```
UINT Listen(LPVOID pParam)
{
	return 0;
}
CWinThread *pThread=AfxBeginThread(Listen,&port);
```

31.线程挂起唤醒

```
::SuspendThread(pThread->m_hThread);//
//pThread->SuspendThread(); //工作者线程时,用于子挂起,此时的线程类应该是个全局的对象;
::ResumeThread(pThread->m_hThread);//API函数的唤醒线程
//pThread->ResumeThread();
```

32.线程插入终止

```
DWORD dwExitCode;
GetExitCodeThread(pThread->m_hThread, &dwExitCode );
AfxEndThread( dwExitCode, TRUE );
```

33.HTTP多线程下载

```
// CInternetSession在遇到一些错误时会抛出异常,因此必须包起来
TRY
{
        CInternetSession  sess ;
          // 统一以二进制方式下载
          DWORD    dwFlag =
INTERNET_FLAG_TRANSFER_BINARY|INTERNET_FLAG_DONT_CACHE|INTERNET_FLAG_RELOAD ;
              CHttpFile  * pF = (CHttpFile*)sess.OpenURL(strFilename, 1, dwFlag); ASSERT(pF);
                if (!pF)
                      {AfxThrowInternetException(1);}
                    // 得到文件大小
                    CString   str ;
                      pF->QueryInfo (HTTP_QUERY_CONTENT_LENGTH, str) ;
                           int  nFileSize = _ttoi(str) ;
                             char  * p = new[nFileSize] ;
                                while (true)
                                  {
                                              // 每次下载8Kb
                                              int  n = pF->Read (p, (nFileSize < 8192) ? nFileSize : 8192) ;
                                                    if (n <= 0)
                                                                break ;
                                                          p += n ; nFileSize -= n ;
                                  }
                                  delete[] p ;
                                       delete pF ;
}
CATCH_ALL(e) {}
END_CATCH_ALL

int  n = pF->GetLength() ;
while (n)
{
        int  * p = new BYTE[n] ;
          pF->Read (p, n) ;
               delete[] p ;
                 n = pF->GetLength() ;
}
if (n == 0)
{
        DWORD  dw ;
          if (::InternetQueryDataAvailable ((HINTERNET)(*pF), &dw, 0, 0) && (dw == 0))
               {
```

```
                    // 到这里就代表文件下载成功了
            }
}
```

34.MP3播放

```
MCI_OPEN_PARMS openpa;
openpa.lpstrDeviceType="MCI_DEVTYPE_WAVEFORM_AUDIO";
openpa.lpstrElementName=%%1;
mciSendCommand(NULL,MCI_OPEN,MCI_DEVTYPE_WAVEFORM_AUDIO,(DWORD)(LPVOID)&openpa);
m_wDeviceID=openpa.wDeviceID;
m_open=true;
MCI_OPEN_PARMS playpa;
// playpa.dwCallback=(DWORD)pWnd->m_hWnd;
mciSendCommand(m_wDeviceID,MCI_SEEK,MCI_SEEK_TO_START,NULL);
mciSendCommand(m_wDeviceID,MCI_PLAY,NULL,(DWORD)(LPVOID)&playpa);
// return false;*/
mciSendCommand(m_wDeviceID,MCI_STOP,NULL,NULL);
m_open=false;
if(m_wDeviceID)
{
      mciSendCommand(m_wDeviceID,MCI_STOP,MCI_WAIT,NULL);
      mciSendCommand(m_wDeviceID,MCI_CLOSE,NULL,NULL);
}
m_wDeviceID=0;
```

35.WAV播放

```
MCI_OPEN_PARMS openpa;
openpa.lpstrDeviceType="MCI_DEVTYPE_WAVEFORM_AUDIO";
openpa.lpstrElementName=%%1;
mciSendCommand(NULL,MCI_OPEN,MCI_DEVTYPE_WAVEFORM_AUDIO,(DWORD)(LPVOID)&openpa);
m_wDeviceID=openpa.wDeviceID;
m_open=true;
MCI_OPEN_PARMS playpa;
// playpa.dwCallback=(DWORD)pWnd->m_hWnd;
mciSendCommand(m_wDeviceID,MCI_SEEK,MCI_SEEK_TO_START,NULL);
mciSendCommand(m_wDeviceID,MCI_PLAY,NULL,(DWORD)(LPVOID)&playpa);
// return false;*/
mciSendCommand(m_wDeviceID,MCI_STOP,NULL,NULL);
m_open=false;
if(m_wDeviceID)
{
      mciSendCommand(m_wDeviceID,MCI_STOP,MCI_WAIT,NULL);
      mciSendCommand(m_wDeviceID,MCI_CLOSE,NULL,NULL);
}
m_wDeviceID=0;
```