**e-mail客户端的实现**

**71Y15101 傅文涛**

**71Y15102 孟越**

**71Y15103 薛栋**

1. **实验目的**

实现一个email客户端，基于smtp,pop或imap协议，实现简单地收发邮件的功能，并对邮件进行解析（MIME格式），最好能实现收发附件的功能

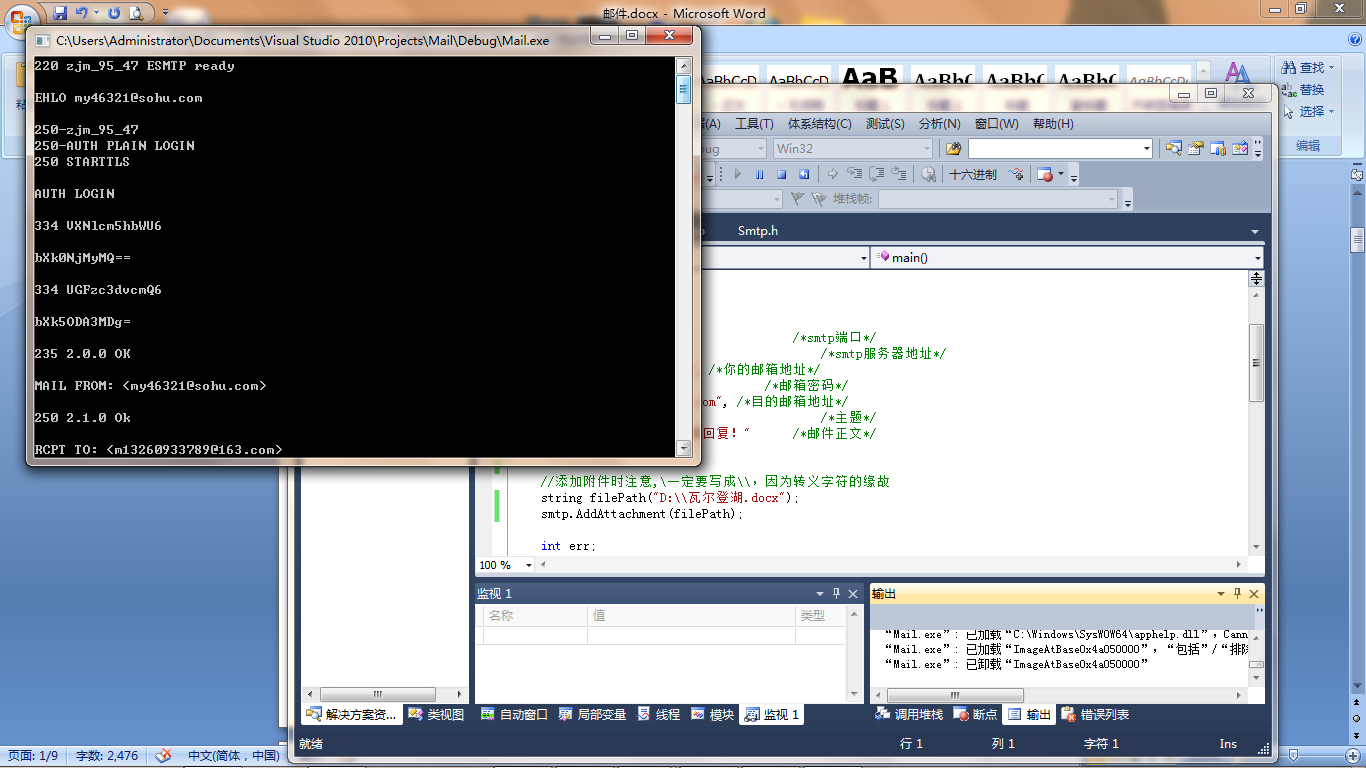
1. **实验原理**

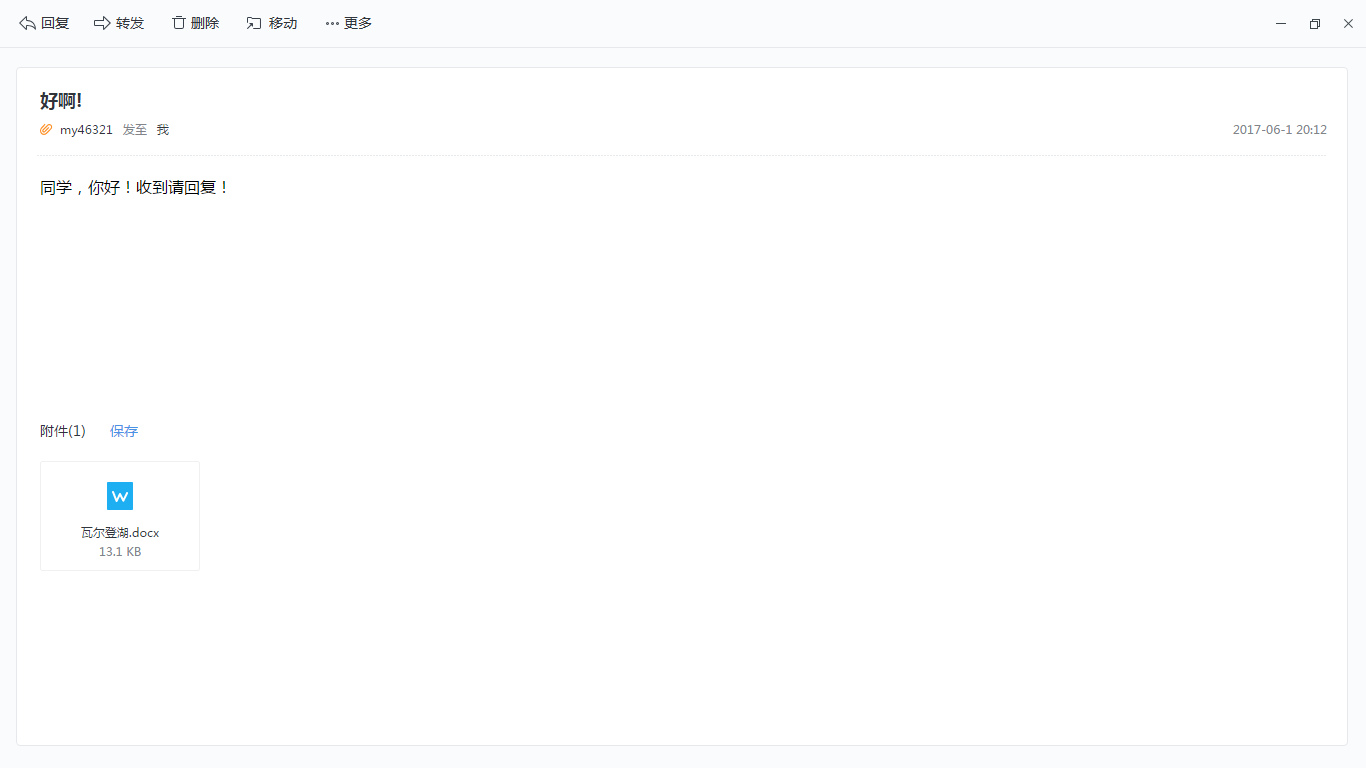
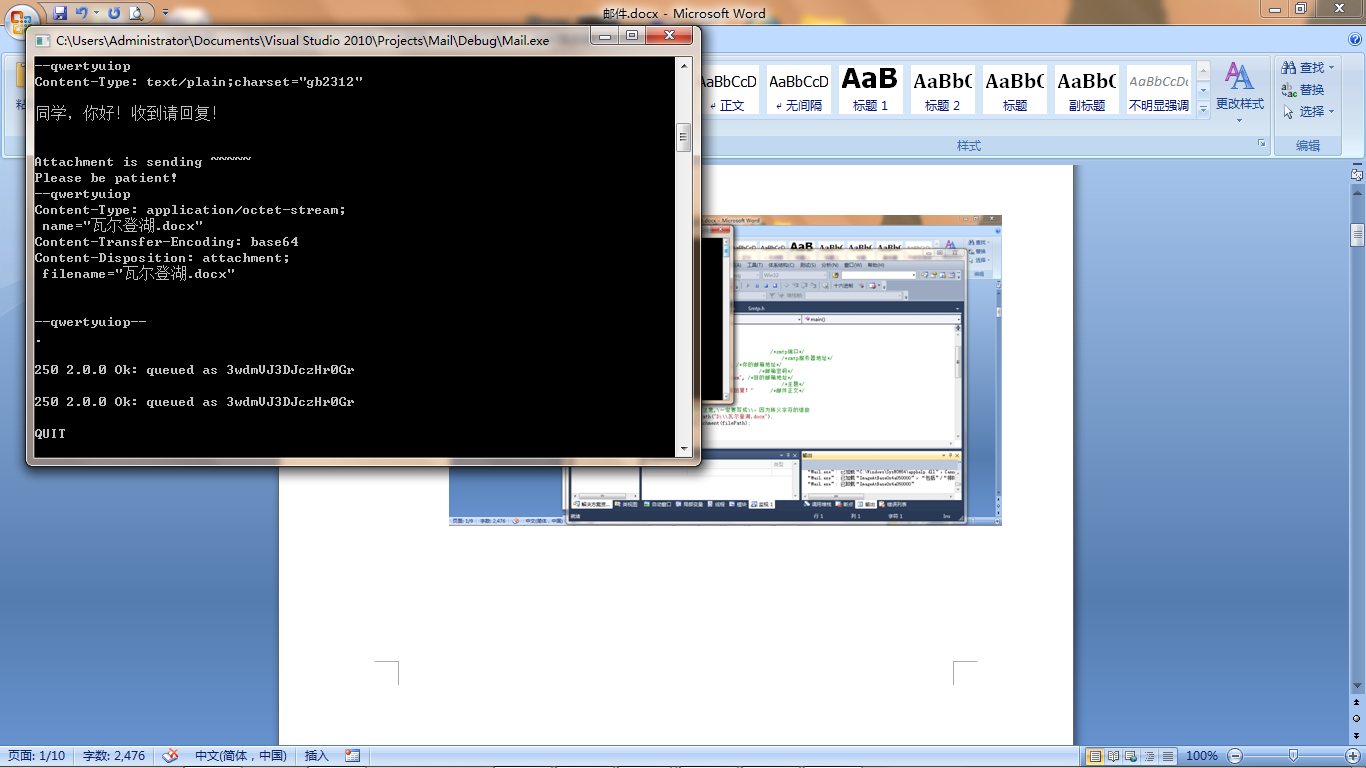
原理比较简单，我们这里使用了别人的base64编解码代码，基于smtp协议进行发送。

大致过程为：

构建smtp包->与smtp服务器建立连接

1. **实验结果**

****

****

1. **实验反思**
2. **具体实现代码（仅仅放上smtp的实现）：**

#include "Smtp.h"

#include <iostream>

#include <fstream>

#include <string>

using namespace std;

#pragma comment(lib, "ws2\_32.lib") /\*链¢¡ä接¨®ws2\_32.lib动¡¥态¬?链¢¡ä接¨®库a\*/

/\*base64采¨¦用®?别Àe人¨?的Ì?编À¨¤码?,不?过y，ê?这a不?是º?重?点Ì?，ê?重?点Ì?是º?我¨°完ª¨º成¨¦了¢?我¨°的Ì?一°?个?比À¨¨较?好?的Ì?邮®¨º件t发¤¡é送¨ª客¨ª户¡ì端?\*/

char\* CSmtp::base64Encode(char const\* origSigned, unsigned origLength)

{

unsigned char const\* orig = (unsigned char const\*)origSigned; // in case any input bytes have the MSB set

if (orig == NULL) return NULL;

unsigned const numOrig24BitValues = origLength / 3;

bool havePadding = origLength > numOrig24BitValues \* 3;

bool havePadding2 = origLength == numOrig24BitValues \* 3 + 2;

unsigned const numResultBytes = 4 \* (numOrig24BitValues + havePadding);

char\* result = new char[numResultBytes + 3]; // allow for trailing '/0'

// Map each full group of 3 input bytes into 4 output base-64 characters:

unsigned i;

for (i = 0; i < numOrig24BitValues; ++i)

{

result[4 \* i + 0] = base64Char[(orig[3 \* i] >> 2) & 0x3F];

result[4 \* i + 1] = base64Char[(((orig[3 \* i] & 0x3) << 4) | (orig[3 \* i + 1] >> 4)) & 0x3F];

result[4 \* i + 2] = base64Char[((orig[3 \* i + 1] << 2) | (orig[3 \* i + 2] >> 6)) & 0x3F];

result[4 \* i + 3] = base64Char[orig[3 \* i + 2] & 0x3F];

}

// Now, take padding into account. (Note: i == numOrig24BitValues)

if (havePadding)

{

result[4 \* i + 0] = base64Char[(orig[3 \* i] >> 2) & 0x3F];

if (havePadding2)

{

result[4 \* i + 1] = base64Char[(((orig[3 \* i] & 0x3) << 4) | (orig[3 \* i + 1] >> 4)) & 0x3F];

result[4 \* i + 2] = base64Char[(orig[3 \* i + 1] << 2) & 0x3F];

}

else

{

result[4 \* i + 1] = base64Char[((orig[3 \* i] & 0x3) << 4) & 0x3F];

result[4 \* i + 2] = '=';

}

result[4 \* i + 3] = '=';

}

result[numResultBytes] = '\0';

return result;

}

CSmtp::CSmtp(void)

{

this->content = "";

this->port = 25;

this->user = "";

this->pass = "";

this->targetAddr = "";

this->title = "";

this->domain = "";

WORD wVersionRequested;

WSADATA wsaData;

int err;

wVersionRequested = MAKEWORD(2, 1);

err = WSAStartup(wVersionRequested, &wsaData);

this->sockClient = 0;

}

CSmtp::~CSmtp(void)

{

DeleteAllAttachment();

closesocket(sockClient);

WSACleanup();

}

CSmtp::CSmtp(

int port,

string srvDomain,

string userName,

string password,

string targetEmail,

string emailTitle,

string content

)

{

this->content = content;

this->port = port;

this->user = userName;

this->pass = password;

this->targetAddr = targetEmail;

this->title = emailTitle;

this->domain = srvDomain;

WORD wVersionRequested;

WSADATA wsaData;

int err;

wVersionRequested = MAKEWORD(2, 1);

err = WSAStartup(wVersionRequested, &wsaData);

this->sockClient = 0;

}

bool CSmtp::CreateConn()

{

//为a建¡§立¢¡ésocket对?象¨®做Á?准Á?备À?，ê?初?始º?化¡¥环¡¤境3

SOCKET sockClient = socket(AF\_INET, SOCK\_STREAM, 0); //建¡§立¢¡ésocket对?象¨®

SOCKADDR\_IN addrSrv;

HOSTENT\* pHostent;

pHostent = gethostbyname(domain.c\_str()); //得Ì?到Ì?有®D关?于®¨²域®¨°名?的Ì?信?息¡é

addrSrv.sin\_addr.S\_un.S\_addr = \*((DWORD \*)pHostent->h\_addr\_list[0]); //得Ì?到Ì?smtp服¤t务?器¡Â的Ì?网ª?络?字Á?节¨²序¨°的Ì?ip地Ì?址¡¤

addrSrv.sin\_family = AF\_INET;

addrSrv.sin\_port = htons(port);

int err = connect(sockClient, (SOCKADDR\*)&addrSrv, sizeof(SOCKADDR)); //向¨°服¤t务?器¡Â发¤¡é送¨ª请?求¨®

if (err != 0)

{

return false;

//printf("链¢¡ä接¨®失º¡ì败ã¨¹\n");

}

this->sockClient = sockClient;

if (false == Recv())

{

return false;

}

return true;

}

bool CSmtp::Send(string &message)

{

int err = send(sockClient, message.c\_str(), message.length(), 0);

if (err == SOCKET\_ERROR)

{

return false;

}

cout << message.c\_str() << endl;

return true;

}

bool CSmtp::Recv()

{

memset(buff, 0, sizeof(char)\* (MAXLEN + 1));

int err = recv(sockClient, buff, MAXLEN, 0); //接¨®收º?数ºy据Y

if (err == SOCKET\_ERROR)

{

return false;

}

buff[err] = '\0';

cout << buff << endl;

return true;

}

int CSmtp::Login()

{

string sendBuff;

sendBuff = "EHLO ";

sendBuff += user;

sendBuff += "\r\n";

if (false == Send(sendBuff) || false == Recv()) //既¨¨接¨®收º?也°2发¤¡é送¨ª

{

return 1; /\*1表À¨ª示º?发¤¡é送¨ª失º¡ì败ã¨¹由®¨¦于®¨²网ª?络?错ä¨ª误¨®\*/

}

sendBuff.empty();

sendBuff = "AUTH LOGIN\r\n";

if (false == Send(sendBuff) || false == Recv()) //请?求¨®登Ì?陆?

{

return 1; /\*1表À¨ª示º?发¤¡é送¨ª失º¡ì败ã¨¹由®¨¦于®¨²网ª?络?错ä¨ª误¨®\*/

}

sendBuff.empty();

int pos = user.find('@', 0);

sendBuff = user.substr(0, pos); //得Ì?到Ì?用®?户¡ì名?

char \*ecode;

/\*在¨²这a里¤?顺3带ä?扯?一°?句?，ê?关?于®¨²string类¤¨¤的Ì?length函¡¥数ºy与®?C语®?言?中D的Ì?strlen函¡¥数ºy的Ì?区?别Àe,strlen计?算?出?来¤¡ä的Ì?长¡è度¨¨，ê?只?到Ì?'\0'字Á?符¤?为a止1,而?string::length()函¡¥数ºy实º¦Ì际¨º上¦?返¤¦Ì回?的Ì?是º?string类¤¨¤中D字Á?符¤?数ºy组Á¨¦的Ì?大ä¨®小?,你?自Á?己o可¨¦以°?测a试º?一°?下?，ê?这a也°2是º?为a什º2么¡ä我¨°下?面?不?使º1用®?string::length()的Ì?原-因°¨°\*/

ecode = base64Encode(sendBuff.c\_str(), strlen(sendBuff.c\_str()));

sendBuff.empty();

sendBuff = ecode;

sendBuff += "\r\n";

delete[]ecode;

if (false == Send(sendBuff) || false == Recv()) //发¤¡é送¨ª用®?户¡ì名?，ê?并¡é接¨®收º?服¤t务?器¡Â的Ì?返¤¦Ì回?

{

return 1; /\*错ä¨ª误¨®码?1表À¨ª示º?发¤¡é送¨ª失º¡ì败ã¨¹由®¨¦于®¨²网ª?络?错ä¨ª误¨®\*/

}

sendBuff.empty();

ecode = base64Encode(pass.c\_str(), strlen(pass.c\_str()));

sendBuff = ecode;

sendBuff += "\r\n";

delete[]ecode;

if (false == Send(sendBuff) || false == Recv()) //发¤¡é送¨ª用®?户¡ì密¨¹码?，ê?并¡é接¨®收º?服¤t务?器¡Â的Ì?返¤¦Ì回?

{

return 1; /\*错ä¨ª误¨®码?1表À¨ª示º?发¤¡é送¨ª失º¡ì败ã¨¹由®¨¦于®¨²网ª?络?错ä¨ª误¨®\*/

}

if (NULL != strstr(buff, "550"))

{

return 2;/\*错ä¨ª误¨®码?2表À¨ª示º?用®?户¡ì名?错ä¨ª误¨®\*/

}

if (NULL != strstr(buff, "535")) /\*535是º?认¨?证¡è失º¡ì败ã¨¹的Ì?返¤¦Ì回?\*/

{

return 3; /\*错ä¨ª误¨®码?3表À¨ª示º?密¨¹码?错ä¨ª误¨®\*/

}

return 0;

}

bool CSmtp::SendEmailHead() //发¤¡é送¨ª邮®¨º件t头ª¡¤部?信?息¡é

{

string sendBuff;

string temp1="MAIL FROM: <";

string temp2=">\r\n";

sendBuff=temp1+user+temp2;

if (false == Send(sendBuff) || false == Recv())

{

return false; /\*表À¨ª示º?发¤¡é送¨ª失º¡ì败ã¨¹由®¨¦于®¨²网ª?络?错ä¨ª误¨®\*/

}

sendBuff.empty();

sendBuff = "RCPT TO: <" + targetAddr + ">\r\n";

if (false == Send(sendBuff) || false == Recv())

{

return false; /\*表À¨ª示º?发¤¡é送¨ª失º¡ì败ã¨¹由®¨¦于®¨²网ª?络?错ä¨ª误¨®\*/

}

sendBuff.empty();

sendBuff = "DATA\r\n";

if (false == Send(sendBuff) || false == Recv())

{

return false; //表À¨ª示º?发¤¡é送¨ª失º¡ì败ã¨¹由®¨¦于®¨²网ª?络?错ä¨ª误¨®

}

sendBuff.empty();

FormatEmailHead(sendBuff);

if (false == Send(sendBuff))

//发¤¡é送¨ª完ª¨º头ª¡¤部?之?后¨®不?必À?调Ì¡Â用®?接¨®收º?函¡¥数ºy,因°¨°为a你?没?有®D\r\n.\r\n结¨¢尾2，ê?服¤t务?器¡Â认¨?为a你?没?有®D发¤¡é完ª¨º数ºy据Y，ê?所¨´以°?不?会¨¢返¤¦Ì回?什º2么¡ä值¦Ì

{

return false; /\*表À¨ª示º?发¤¡é送¨ª失º¡ì败ã¨¹由®¨¦于®¨²网ª?络?错ä¨ª误¨®\*/

}

return true;

}

void CSmtp::FormatEmailHead(string &email)

{/\*格?式º?化¡¥要°a发¤¡é送¨ª的Ì?内¨²容¨Y\*/

email = "From: ";

email += user;

email += "\r\n";

email += "To: ";

email += targetAddr;

email += "\r\n";

email += "Subject: ";

email += title;

email += "\r\n";

email += "MIME-Version: 1.0";

email += "\r\n";

email += "Content-Type: multipart/mixed;boundary=qwertyuiop";

email += "\r\n";

email += "\r\n";

}

bool CSmtp::SendTextBody() /\*发¤¡é送¨ª邮®¨º件t文?本À?\*/

{

string sendBuff;

sendBuff = "--qwertyuiop\r\n";

sendBuff += "Content-Type: text/plain;";

sendBuff += "charset=\"gb2312\"\r\n\r\n";

sendBuff += content;

sendBuff += "\r\n\r\n";

return Send(sendBuff);

}

int CSmtp::SendAttachment\_Ex() /\*发¤¡é送¨ª附?件t\*/

{

for (list<FILEINFO \*>::iterator pIter = listFile.begin(); pIter != listFile.end(); pIter++)

{

cout << "Attachment is sending ~~~~~" << endl;

cout << "Please be patient!" << endl;

string sendBuff;

sendBuff = "--qwertyuiop\r\n";

sendBuff += "Content-Type: application/octet-stream;\r\n";

sendBuff += " name=\"";

sendBuff += (\*pIter)->fileName;

sendBuff += "\"";

sendBuff += "\r\n";

sendBuff += "Content-Transfer-Encoding: base64\r\n";

sendBuff += "Content-Disposition: attachment;\r\n";

sendBuff += " filename=\"";

sendBuff += (\*pIter)->fileName;

sendBuff += "\"";

sendBuff += "\r\n";

sendBuff += "\r\n";

Send(sendBuff);

ifstream ifs((\*pIter)->filePath, ios::in | ios::binary);

if (false == ifs.is\_open())

{

return 4; /\*错ä¨ª误¨®码?4表À¨ª示º?文?件t打ä¨°开a错ä¨ª误¨®\*/

}

char fileBuff[MAX\_FILE\_LEN];

char \*chSendBuff;

memset(fileBuff, 0, sizeof(fileBuff));

/\*文?件t使º1用®?base64加¨®密¨¹传ä?送¨ª\*/

while (ifs.read(fileBuff, MAX\_FILE\_LEN))

{

//cout << ifs.gcount() << endl;

chSendBuff = base64Encode(fileBuff, MAX\_FILE\_LEN);

chSendBuff[strlen(chSendBuff)] = '\r';

chSendBuff[strlen(chSendBuff)] = '\n';

send(sockClient, chSendBuff, strlen(chSendBuff), 0);

delete[]chSendBuff;

}

//cout << ifs.gcount() << endl;

chSendBuff = base64Encode(fileBuff, ifs.gcount());

chSendBuff[strlen(chSendBuff)] = '\r';

chSendBuff[strlen(chSendBuff)] = '\n';

int err = send(sockClient, chSendBuff, strlen(chSendBuff), 0);

if (err != strlen(chSendBuff))

{

cout << "文?件t传ä?送¨ª出?错ä¨ª!" << endl;

return 1;

}

delete[]chSendBuff;

}

return 0;

}

bool CSmtp::SendEnd() /\*发¤¡é送¨ª结¨¢尾2信?息¡é\*/

{

string sendBuff;

sendBuff = "--qwertyuiop--";

sendBuff += "\r\n.\r\n";

if (false == Send(sendBuff) || false == Recv())

{

return false;

}

cout << buff << endl;

sendBuff.empty();

sendBuff = "QUIT\r\n";

return (Send(sendBuff) && Recv());

}

int CSmtp::SendEmail\_Ex()

{

if (false == CreateConn())

{

return 1;

}

//Recv();

int err = Login(); //先¨¨登Ì?录?

if (err != 0)

{

return err; //错ä¨ª误¨®代ä¨²码?必À?须?要°a返¤¦Ì回?

}

if (false == SendEmailHead()) //发¤¡é送¨ªEMAIL头ª¡¤部?信?息¡é

{

return 1; /\*错ä¨ª误¨®码?1是º?由®¨¦于®¨²网ª?络?的Ì?错ä¨ª误¨®\*/

}

if (false == SendTextBody())

{

return 1; /\*错ä¨ª误¨®码?1是º?由®¨¦于®¨²网ª?络?的Ì?错ä¨ª误¨®\*/

}

err = SendAttachment\_Ex();

if (err != 0)

{

return err;

}

if (false == SendEnd())

{

return 1; /\*错ä¨ª误¨®码?1是º?由®¨¦于®¨²网ª?络?的Ì?错ä¨ª误¨®\*/

}

return 0; /\*0表À¨ª示º?没?有®D出?错ä¨ª\*/

}

void CSmtp::AddAttachment(string &filePath) //添¬¨ª加¨®附?件t

{

FILEINFO \*pFile = new FILEINFO;

strcpy\_s(pFile->filePath, filePath.c\_str());

const char \*p = filePath.c\_str();

strcpy\_s(pFile->fileName, p + filePath.find\_last\_of("\\") + 1);

listFile.push\_back(pFile);

}

void CSmtp::DeleteAttachment(string &filePath) //删¦?除y附?件t

{

list<FILEINFO \*>::iterator pIter;

for (pIter = listFile.begin(); pIter != listFile.end(); pIter++)

{

if (strcmp((\*pIter)->filePath, filePath.c\_str()) == 0)

{

FILEINFO \*p = \*pIter;

listFile.remove(\*pIter);

delete p;

break;

}

}

}

void CSmtp::DeleteAllAttachment() /\*删¦?除y所¨´有®D的Ì?文?件t\*/

{

for (list<FILEINFO \*>::iterator pIter = listFile.begin(); pIter != listFile.end();)

{

FILEINFO \*p = \*pIter;

pIter = listFile.erase(pIter);

delete p;

}

}

void CSmtp::SetSrvDomain(string &domain)

{

this->domain = domain;

}

void CSmtp::SetUserName(string &user)

{

this->user = user;

}

void CSmtp::SetPass(string &pass)

{

this->pass = pass;

}

void CSmtp::SetTargetEmail(string &targetAddr)

{

this->targetAddr = targetAddr;

}

void CSmtp::SetEmailTitle(string &title)

{

this->title = title;

}

void CSmtp::SetContent(string &content)

{

this->content = content;

}

void CSmtp::SetPort(int port)

{

this->port = port;

}