上机作业二：简单的SMTP服务器程序 实验报告

1 实验要求

1. 响应客户 SMTP 命令，并将命令的交互过程和收到的邮件显示到屏幕上（注：可正常显示邮件内容和附件，附件为图片或文档）
2. 支持单用户（注：客户端为本计算机邮箱）
3. 不存储和转发收到的邮件
4. 不做错误处理
5. 界面如图：

2 实验环境

Windows 10、Visual Studio 2019 preview、Debug x86、MFC

3 实验原理

3.1 SMTP协议介绍

3.1.1 SMTP协议简介

SMTP称为简单邮件传输协议（Simple Mail Transfer Protocal），目标是向用户提供高效、可靠的邮件传输。它的一个重要特点是它能够在传送中接力传送邮件，即邮件可以通过不同网络上的主机接力式传送。通常它工作在两种情况下：一是邮件从客户机传输到服务器；二是从某一个服务器传输到另一个服务器。SMTP是一个请求/响应协议，它监听25号端口，用于接收用户的Mail请求，并与远端Mail服务器建立SMTP连接。

3.1.2 SMTP协议工作机制

SMTP通常有两种工作模式。发送SMTP和接收SMTP。具体工作方式为：发送SMTP在接收到用户的邮件请求后，判断此邮件是否为本地邮件，若是直接投送到用户的邮箱，否则向DNS查询远端邮件服务器的MX记录，并建立与远端接收SMTP之间的一个双向传送通道，此后SMTP命令由发送SMTP发出，由接收SMTP接收，而应答则反方向传送。一旦传送通道建立，SMTP发送者发送MAIL命令指明邮件发送者。如果SMTP接收者可以接收邮件则返回OK应答。SMTP发送者再发出RCPT命令确认邮件是否接收到。如果SMTP接收者接收，则返回OK应答；如果不能接收到，则发出拒绝接收应答（但不中止整个邮件操作），双方将如此反复多次。当接收者收到全部邮件后会接收到特别的序列，入伏哦接收者成功处理了邮件，则返回OK应答。

3.1.3 SMTP的连接与发送过程

1. 建立TCP连接
2. 客户端发送HELO命令以标识发件人自己的身份，然后客户端发送MAIL命令；服务器端正希望以OK作为响应，表明准备接收
3. 客户端发送RCPT命令，以标识该电子邮件的计划接收人，可以有多个RCPT行；服务器端则表示是否愿意为收件人接收邮件
4. 协商结束，发送邮件，用命令DATA发送
5. 以.表示结束输入内容一起发送出去
6. 结束此次发送，用QUIT命令退出

3.1.4 SMTP常用的相应

应答码	说明
501	参数格式错误
502	命令不可实现
503	错误的命令序列
504	命令参数不可实现
211	系统状态或系统帮助响应
214	帮助信息
220	服务器就绪
221	服务关闭
421	服务器未就绪，关闭传输信道
250	要求的邮件操作完成
251	用户非本地，将转发向
450	要求的邮件操作未完成，邮箱不可用
550	要求的邮件操作未完成，邮箱不可用
451	放弃要求的操作，处理过程中出错
551	用户非本地，请尝试
452	系统存储不足，要求的操作未执行
552	过量的存储分配，要求的操作未执行
553	邮箱名不可用，要求的操作未执行
354	开始邮件输入，以“.”结束

应答码	说明
554	操作失败

3.1.5 常用的SMTP命令

命令	描述
DATA	将之后的数据作为数据发送，以.标志数据的结尾
EXPN	验证给定的邮箱列表是否存在，扩充邮箱列表，也常被禁用
HELO	向服务器标识用户身份，返回邮件服务器身份
HELP	查询服务器支持什么命令，返回命令中的信息
MAIL FROM	在主机上初始化一个邮件会话
NOOP	无操作，服务器应响应OK
QUIT	终止邮件会话
RCPT TO	标识单个的邮件接收人；常在MAIL命令后面可有多个rcpt to：
RSET	重置会话，当前传输被取消
SAML FROM	发送邮件到用户终端和邮箱
SEND FROM	发送邮件到用户终端
SOML FROM	发送邮件到用户终端或邮箱
TURN	接收端和发送端交换角色
VRFY	用于验证指定的用户/邮箱是否存在；由于安全方面的原因，服务器常禁止此命令

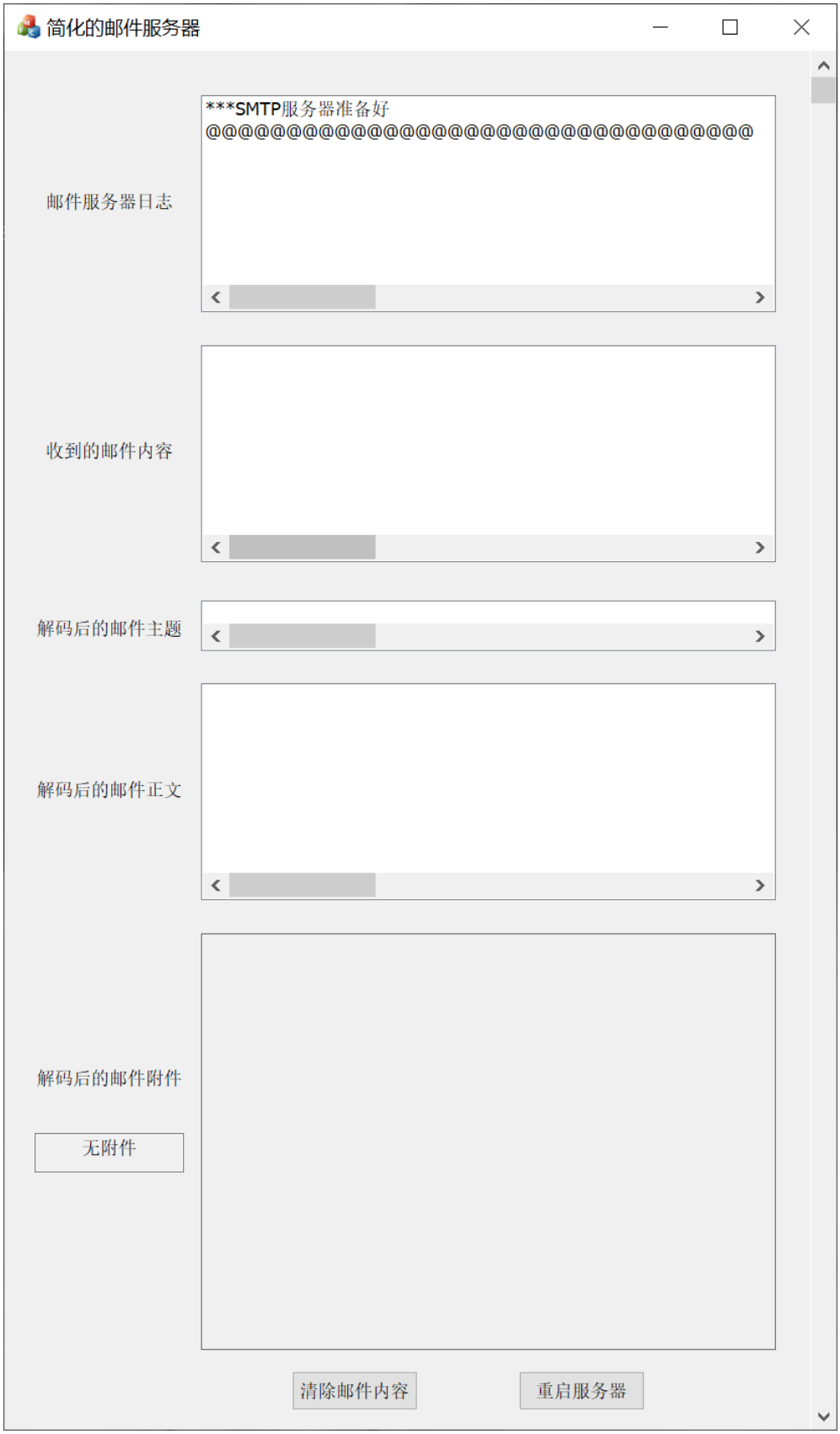
3.2 base64介绍

1. base64的编码都是按字符串长度，以每3个8bit的字符为一组。
2. 然后针对每组，首先获取每个字符的ASCII编码。
3. 然后将ASCII编码转换成8bit的二进制，得到一组3*8=24bit的字节。
4. 然后再将这24bit划分为4个6bit的字节，并在每个6bit的字节前面都填两个高位0，得到4个8bit的字节。
5. 然后将这4个8bit的字节转换成10进制，对照Base64编码表 ，得到对应编码后的字符。
6. 如果被编码字符长度不是3的倍数的时候，则都用0代替，对应的输出字符为=。

4 实现步骤

4.1 创建MFC应用程序

1. 在资源视图中构建以下界面



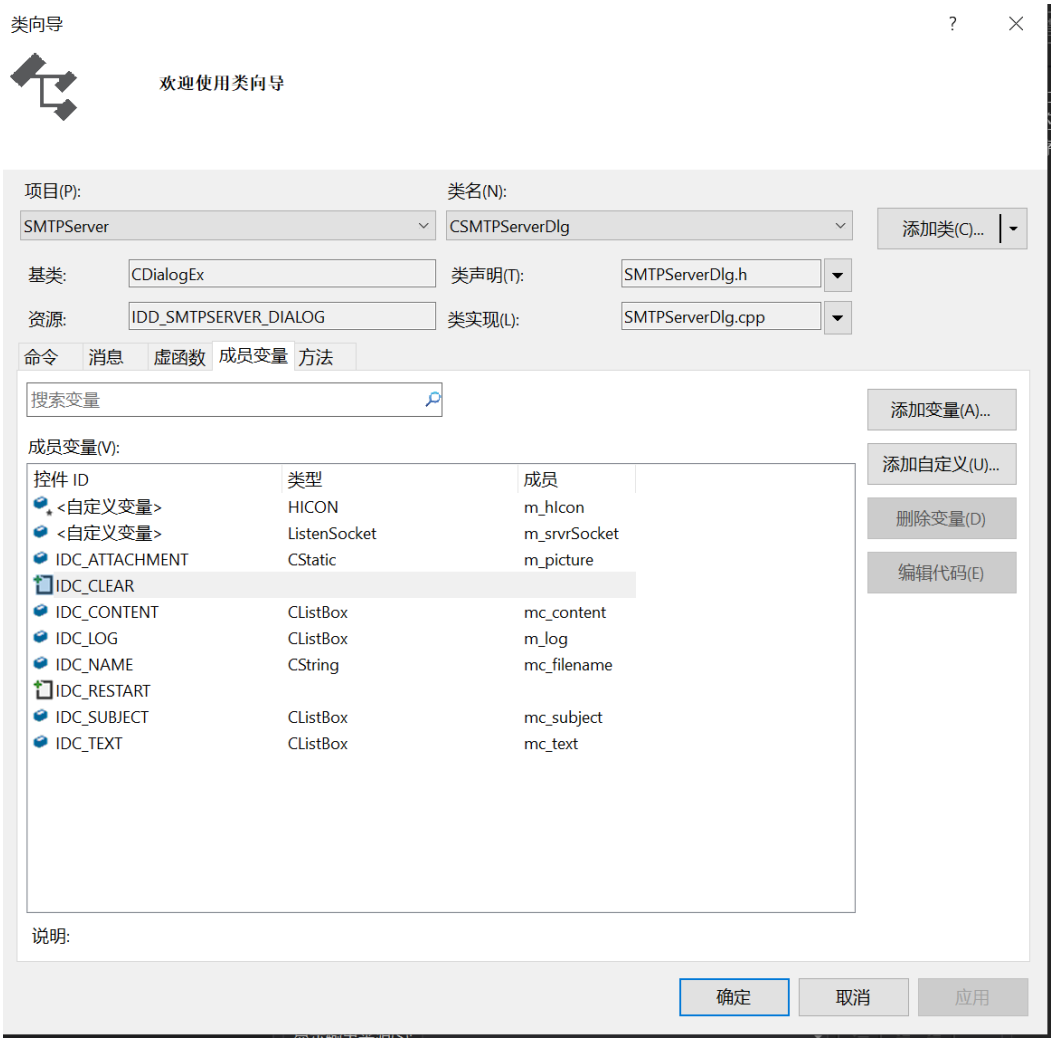
2. 修改控件的ID与其他属性值，比如，只读，滚动栏、排序等。

4.2 对话框类

对话框类主要实现初始化对话框，创建监听套接字，控件函数。

4.2.1 变量声明

使用类向导对控件的变量进行声明。在声明完成后，编译器会自动对应该初始化的变量进行初始化。



1. `m_srvrSocket` 为监听套接字类的对象，用来监听套接字。
2. `m_picture` 是加载图片的图片控件对应的变量。
3. `mc_content` 对应的是显示未解码的邮件的内容。
4. `m_log` 是保存邮件服务器运行的记录的变量，也就是与客户端进行对话的记录。
5. `mc_filename` 是保存附件名称的变量。
6. `mc_subject` 是保存解码后的邮件主题的变量。
7. `mc_text` 是保存解码后的邮件的正文的变量。

4.2.2 监听套接字

在程序启动时候就应该对套接字进行监听。并在异常时报出相应的错误。因此在`OnInitDialog()`函数中添加以下代码。具体每一步的功能，见注释。

```
// TODO: 在此添加额外的初始化代码
    //设置ListBox控件的宽度
    m_log.SetHorizontalExtent(2000);
    mc_content.SetHorizontalExtent(2000);
    mc_text.SetHorizontalExtent(2000);
    mc_subject.SetHorizontalExtent(2000);
    if (m_srvrSocket.m_hSocket == INVALID_SOCKET) {

        //创建监听套接字,激发FD_ACCEPT事件,在TCP的25端口守候,
```

```
        BOOL bFlag = m_srvrSocket.Create(25, SOCK_STREAM,
FD_ACCEPT);//失败返回-1
        if (!bFlag)//创建失败
        {
            MessageBox(_T("Socket创建失败!"), L"提醒", MB_OK |
MB_ICONSTOP);
            m_srvrSocket.Close();
            return TRUE;
        }

        m_log.AddString(_T("***SMTP服务器准备好"));

m_log.AddString(_T("@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@"));

        //监听成功,等待连接请求
        if(!m_srvrSocket.Listen())
        {
            int nErrorCode = m_srvrSocket.GetLastError();    //检
测错误信息
            if (nErrorCode != WSAEWOULDBLOCK)             //如果不
是线程阻塞
            {
                MessageBox(_T("Socket错误!"), L"提醒", MB_OK |
MB_ICONSTOP);
                m_srvrSocket.Close();
                return TRUE;
            }
        }
    }
```

4.2.3 清空邮件按钮功能实现

想要清空相应的控件，可调用该控件下相应的清空函数。也可把变量值改为空然后刷新窗口，为了保险起见，不建议用修改变量的方法实现清空。关于Picture控件，因为没有内置的清空函数，所以实现方法是先关闭再打开。

通过双击按钮即可创建相应的点击函数，代码如下。

```
void CSMTPServerDlg::OnBnClickedClear()//清空按钮
{
    // TODO: 在此添加控件通知处理程序代码
    CListBox* plb1 = (CListBox*)GetDlgItem(IDC_CONTENT);//根据ID获
取相应的控件
    plb1->ResetContent();//调用ListBox的清空函数。
    CListBox* plb2 = (CListBox*)GetDlgItem(IDC_TEXT);
    plb2->ResetContent();
    CListBox* plb3 = (CListBox*)GetDlgItem(IDC_SUBJECT);
    plb3->ResetContent();
    CStatic* plb4 = (CStatic*)GetDlgItem(IDC_ATTACHMENT);
    plb4->ShowWindow(false);//通过关闭再打开实现清空
    plb4->ShowWindow(true);
}
```

4.2.4 重启服务器按钮功能实现

这个重启其实是先获取程序运行的路径，然后新建一个进程，就把现有的挤下去了。代码如下：

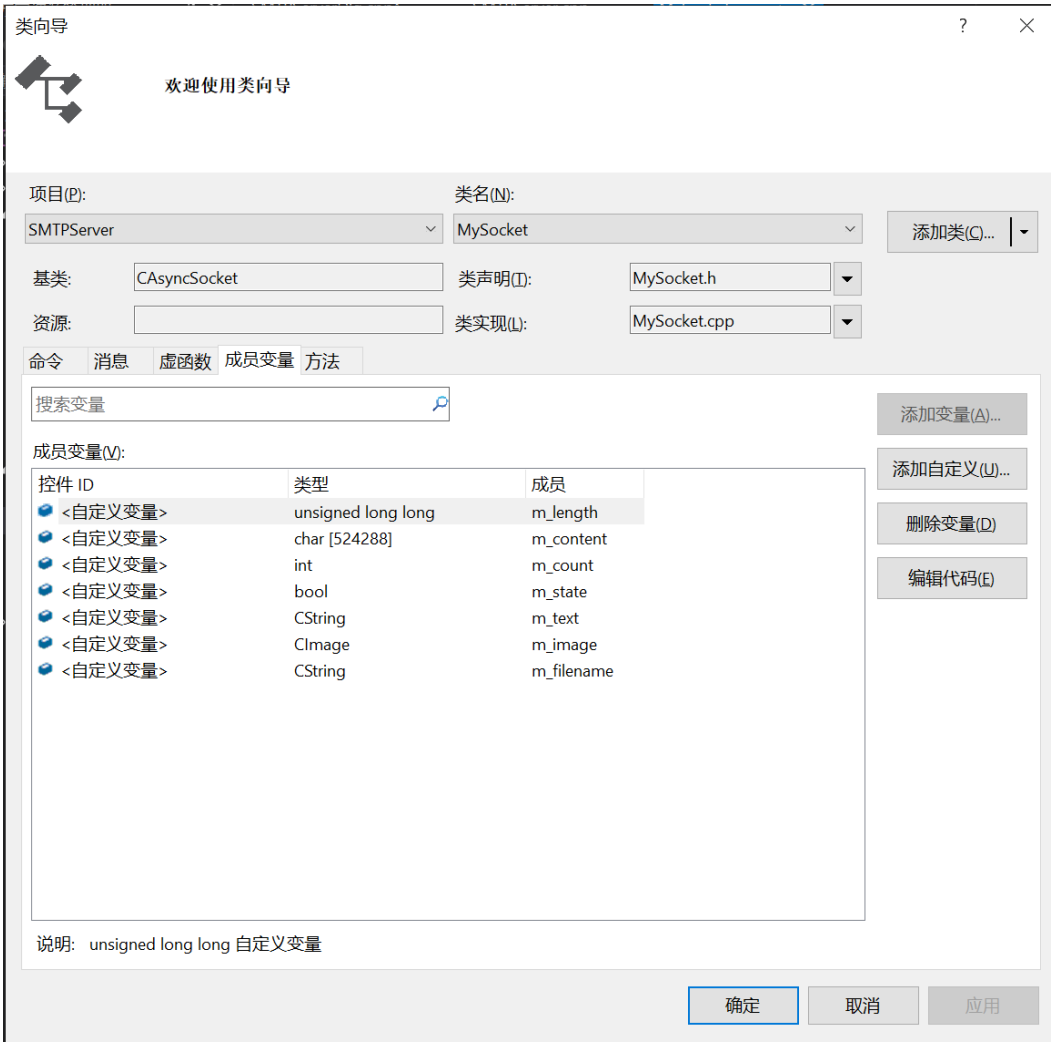
```
void CSMTPServerDlg::OnBnClickedRestart()
{
    // TODO: 在此添加控件通知处理程序代码
    ::PostMessage(AfxGetMainWnd()->m_hWnd, WM_SYSCOMMAND,
SC_CLOSE, NULL);
    //获取exe程序当前路径
    extern CSMTPServerApp theApp;
    TCHAR szAppName[MAX_PATH];
    ::GetModuleFileName(theApp.m_hInstance, szAppName, MAX_PATH);
    CString strAppFullName;
    strAppFullName.Format(_T("%s"), szAppName);
    //重启程序
    STARTUPINFO StartInfo;
    PROCESS_INFORMATION procStruct;
    memset(&StartInfo, 0, sizeof(STARTUPINFO));
    StartInfo.cb = sizeof(STARTUPINFO);
    ::CreateProcess(
        (LPCTSTR)strAppFullName,
        NULL,
        NULL,
        NULL,
        FALSE,
        NORMAL_PRIORITY_CLASS,
        NULL,
        NULL,
        &StartInfo,
        &procStruct);
}
```

4.3 MySocket类

`MySocket`类是基于`CAsyncSocket`类创建的子类。主要功能为实现与客户端的通信，以及把邮件信息发送到对话框。

4.3.1 变量声明

同样使用类向导创建变量。



1. `m_length` 是消息长度
2. `m_content[524288]` 是消息内容，未解码的。
3. `m_count` 记录进行到哪一步
4. `m_state` 标记，false：表示由Send打开，true标记由`Receive()`打开
5. `CString m_text` 是未解码的邮件正文
6. `CImage m_image` 是邮件里的解码后的图片附件
7. `CString m_filename` 是附件名称

4.3.2 初始化变量

一些变量在使用类向导创建的时候就已经初始化完成，但有些变量需要手动初始化，比如字符串数组。

```
MySocket::MySocket()
{
    m_length = 0;
    m_count = 0;
    m_text = _T("");
    m_filename =_T("");
    memset(m_content, 0, sizeof(m_content));
    m_text.Empty();
}
```

4.3.3 `OnSend`发送函数

因为与客户端进行对话分为几个步骤，所以要分情况讨论，并且在这个状态要把应答发送给客户端，并且发送到对话框类，显示出来。然后把状态标记置为true跳转到`OnReceive`函数。

```
void MySocket::OnSend(int nErrorCode)
{
    // TODO: 在此添加专用代码和/或调用基类
    CSMTPServerDlg* dlg = (CSMTPServerDlg*)theApp.m_pMainWnd;//获取对话框句柄
    if (!m_state && (m_count == 0))//服务已经准备好
    {
        char buf[100] = "220  Mail Server Is Ready For Mail\r\n";
        strcpy_s(m_content,buf);
        Send(m_content , strlen(m_content));
        dlg->m_log.AddString(_T("S: 220 Mail Server Is Ready For Mail\r\n"));
        m_state = true;
        AsyncSelect(FD_READ);//触发OnReceive函数
    }
    if (!m_state && (m_count == 1))//接收“主机域名”的请求命令成功完成
    {
        char buf[100] = "250  OK 127.0.0.1\r\n";
        strcpy_s(m_content,buf);
        Send(m_content , strlen(m_content));
        dlg->m_log.AddString(_T("S:250  OK 127.0.0.1\r\n"));
        m_state = true;
        AsyncSelect(FD_READ);//触发OnReceive函数
    }
    if (!m_state && (m_count == 2))//接收“发送者电子邮件地址”的请求命令成功完成
    {
        char buf[100] = "250  Sender OK\r\n";
        strcpy_s(m_content,buf);
        Send(m_content , strlen(m_content));
        dlg->m_log.AddString(_T("S:250  Sender OK\r\n"));
        m_state = true;
        AsyncSelect(FD_READ);//触发OnReceive函数
    }
    if (!m_state && (m_count == 3))//接收“接收者电子邮件地址”的请求命令成功完成
    {
        char buf[100] = "250  Receiver OK\r\n";
        strcpy_s(m_content,buf);
        Send(m_content , strlen(m_content));
        dlg->m_log.AddString(_T("S:250  Receiver OK\r\n"));
        m_state = true;
        AsyncSelect(FD_READ);//触发OnReceive函数
    }
    if (!m_state && (m_count == 4))//可以发送邮件内容
    {
        char buf[100] = "354  Go ahead. End with <CRLF>.<CRLF>\r\n";
```

```
        strcpy_s(m_content,buf);
        Send(m_content , strlen(m_content));
        dlg->m_log.AddString(_T("S:354  Go ahead. End with
<CRLF>.<CRLF>\r\n"));
        m_state = true;
        AsyncSelect(FD_READ);//触发OnReceive函数
    }
    if (!m_state && (m_count == 5))//接收“接收者电子邮件地址”的请求命
令成功完成
    {
        char buf[100] = "250  Message accepted for
delivery!\r\n";
        strcpy_s(m_content,buf);
        Send(m_content , strlen(m_content));
        dlg->m_log.AddString(_T("S:250  Message accepted for
delivery!\r\n"));
        m_state = true;
        AsyncSelect(FD_READ);//触发OnReceive函数
    }
    if (!m_state && (m_count == 6))//系统状态、系统帮忙应答
    {
        char buf[100] = "220 Mail Server Is Ready For Mail\r\n";
        strcpy_s(m_content,buf);
        Send(m_content , strlen(m_content));
        dlg->m_log.AddString(_T("S:220 Mail Server Is Ready For
Mail\r\n"));
        m_state = true;
        AsyncSelect(FD_READ);//触发OnReceive函数
    }
    if (!m_state && (m_count == 6))//会话结束
    {
        char buf[100] = "221 Quit, Goodbye!\r\n";
        strcpy_s(m_content,buf);
        Send(m_content , strlen(m_content));
        dlg->m_log.AddString(_T("S:221 Quit, Goodbye!\r\n"));
        m_state = true;
        AsyncSelect(FD_READ);//触发OnReceive函数
    }
    CAsyncSocket::OnSend(nErrorCode);
}
```

4.3.4 `OnReceive`函数

`OnReceive`函数则是接收客户端发来的消息并显示，在正式进行数据传输之前，就把内容显示在log里面。如果是数据，那么就存放在`m_content`中，然后放入`con_buf`进行处理。并发送给对话框显示。结束了接收数据就把数据发送给`ShowText`与`ShowImage`函数对文本与图片进行展示。

```
void MySocket::OnReceive(int nErrorCode)
{
    // TODO: 在此添加专用代码和/或调用基类
```

```
    CSMTPServerDlg* dlg = (CSMTPServerDlg*)theApp.m_pMainWnd;
    if (m_state && (m_count != 4))//开始会话
    {
        memset(m_content, 0, sizeof(m_content));
        m_length = Receive(m_content,sizeof(m_content));//接收数据
        CString cstr(m_content);

        dlg->m_log.AddString(_T("C: ")+cstr);//显示数据
        m_count++;
        m_state = false;
        AsyncSelect(FD_WRITE);//调用OnSend函数
    }
    if (m_state && (m_count == 4))//在这里接收邮件内容数据
    {
        memset(m_content, 0, sizeof(m_content));
        m_length = Receive(m_content, sizeof(m_content));//接收数
据
        dlg-
>mc_content.AddString(_T("@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@"));
        //转换换行格式显示在ListBox控件中
        char con_buf[sizeof(m_content)];
        int i_out = 0, i_in = 0;
        for (int i = 0;; i++)
        {
            memset(con_buf, 0, sizeof(con_buf));
            i_in = 0;
            while (m_content[i_out] != '\n' && m_content[i_out]
!= '\0' && m_content[i_out] != 0)
            {
                con_buf[i_in++] = m_content[i_out++];

            }
            i_out++;
            if (i_in > 1) con_buf[i_in++] = '\n';
            CString rec(con_buf, i_in);
            dlg->mc_content.AddString(rec);
            m_text = m_text + rec;
            //以“\r\n.\r\n”结束数据接收
            if ((m_content[i_out] == '.' && m_content[i_out + 1]
== '\r' && m_content[i_out + 2] == '\n'))
            {
                m_count++;
                m_state = false;
                ShowText();
                ShowImage();
                break;
            }
            if (m_content[i_out] == 0) break;
        }
        AsyncSelect(FD_WRITE);
    }
    CAsyncSocket::OnReceive(nErrorCode);
```

```
}
```

4.3.5 ShowText函数

ShowText函数主要实现对邮件内容里有用信息的提取。实现方法主要是通过CString里的Find函数找到有用信息的起始位置与结束位置，然后对这部分内容进行解码，输送到对话框类。在这其中注意的一个细节就是要先消除原字符串中的\r\n再进行查找。实现代码如下：

```
void MySocket::ShowText()
{
    CSMTPServerDlg* dlg = (CSMTPServerDlg*)theApp.m_pMainWnd;
    CString TEXT;
    CString SUBJECT;
    TEXT.Empty();
    dlg->mc_text.ResetContent();//清空列表框
    if (m_text.Find(_T("Content-Transfer-Encoding: 7bit")) != -1)
        //正文是7bit编码，无需解码
    {
        int pos1 = m_text.Find(_T("Content-Transfer-Encoding:
7bit")) + 33;
        int pos2 = m_text.Find(_T("-----") + pos1 + 1);
        TEXT = m_text.Mid(pos1, pos2 - pos1 - 2);
    }
    else
    {
        if (m_text.Find(_T("Content-Transfer-Encoding: base64"))
!= -1)
        {
            CString mailcontent;
            m_text.Remove('\r');
            m_text.Remove('\n');
            int pos1 = m_text.Find(_T("Content-Transfer-Encoding:
base64")) + 33;
            int pos2 = m_text.Find(_T("------") , pos1 + 1);
            mailcontent = m_text.Mid(pos1, pos2 - pos1);
            if (mailcontent.IsEmpty())
            {
                dlg->mc_text.AddString(L"邮件正文解析错误！");
            }
            //把CString转为string,传入解码函数
            string ccc(CW2A(mailcontent.GetString()));
            string rrr;
            rrr=base64_decode(ccc);
            TEXT = rrr.c_str();
             //把结果转为CString

        }
        if (m_text.Find(_T("Subject: =?GB2312?B?")) != -1)
        {
            CString subject;
```

```
            m_text.Remove('\r');
            m_text.Remove('\n');
            int pos1 = m_text.Find(_T("Subject: =?GB2312?B?")) +
20;
            int pos2 = m_text.Find(_T("?="), pos1 + 1);
            subject = m_text.Mid(pos1, pos2 - pos1);
            //把CString转为string,传入解码函数
            string ccc = (CW2A(subject.GetString()));
            string rrr;
            rrr = base64_decode(ccc);
            //把结果转为CString
            SUBJECT = rrr.c_str();

            dlg->mc_subject.AddString(SUBJECT);
        }
        if (m_text.Find(_T("Subject: ")) != -1&&
(m_text.Find(_T("Subject: =?GB2312?B?")) == -1))
        {
            CString subject;
            m_text.Remove('\r');
            m_text.Remove('\n');
            int pos1 = m_text.Find(_T("Subject: ")) + 9;
            int pos2 = m_text.Find(_T("X-Priority: "), pos1 + 1);
            subject = m_text.Mid(pos1, pos2 - pos1);
            dlg->mc_subject.AddString(subject);
        }

        if (!TEXT.IsEmpty())
        {
            int pos_begin = 0, pos_end = TEXT.Find('\r', 1);
            while (pos_end != -1)
            {
                CString temp;
                temp= TEXT.Mid(pos_begin, pos_end - pos_begin);
                dlg->mc_text.AddString(temp);
                pos_begin = pos_end + 4;
                pos_end = TEXT.Find('\r', pos_begin + 1);
            }

        }
    }
}
```

4.3.6 ShowImage函数

因为图片的解码与文字的解码时一样的，因为本质都是二进制，所以只需要把解码出来的二进制文件流写入与收到的附件同名的文件里面就可以实现解码了，然后再用picture控件绘制图片。捕获附件的名称与附件的base64码与正文使用的时同一个方法，在此不赘述。在绘制图片时使用了各种方法发现图片进行缩放后都会失真，因为缩放的原理是像素点与周围的点进行运算。所以我就让图片显示的区域动态变化，然后绘制图片，如果图片本身的尺寸小于绘制框，那就按原图，如果大于就以图片为准。代码如下：

```
void MySocket::ShowImage()
{
    // TODO: 在此处添加实现代码.
    CSMTPServerDlg* dlg = (CSMTPServerDlg*)theApp.m_pMainWnd;
    CString  ImageContent;  //存放正文base64编码
    m_text.Remove('\r');
    m_text.Remove('\n');
    int pos1 = m_text.Find(_T("filename"));
    if (pos1 == -1) //如果没有附件则返回
        return;
    int pos2 = m_text.Find('"', pos1 + 7);
    int pos3 = m_text.Find('"', pos2 + 1);
    m_filename = m_text.Mid(pos2 + 1, pos3 - pos2 - 1); //获取附件名称
    dlg->mc_filename = m_filename;
    dlg->UpdateData(false);                                    //显示附件名称在控件中
    int pos4 = m_text.Find(_T("------="), pos3 + 1);
    ImageContent = m_text.Mid(pos3 + 1, pos4 - pos3 - 1);
    //转化为string类进行解码
    string iii(CW2A(ImageContent.GetString()));
    string ima = base64_decode(iii);
    //以二进制的形式写入文件
    fstream of(m_filename, ios_base::out | ios_base::binary);
    of << ima << endl;
    of.close();

    if (!m_image.IsNull())
    {
        m_image.Detach();
    }

    m_image.Load(m_filename);

    if (!m_image.IsNull())
    {
        int height, width;
        height = m_image.GetHeight();
        width = m_image.GetWidth();

        CRect rect;
        CRect rect1;
        dlg->m_picture.GetClientRect(&rect);
        CDC* pDc = dlg->m_picture.GetDC();
        SetStretchBltMode(pDc->m_hDC, STRETCH_HALFTONE);//设置绘制模式
        if (width <= rect.Width() && height <= rect.Width()) //小图片，不缩放
        {
            rect1 = CRect(rect.TopLeft(), CSize(width, height));
```

```
            m_image.StretchBlt(pDc->m_hDC, rect1, SRCCOPY); //将
图片画到Picture控件表示的矩形区域
            return ;
        }
        else
        {
            float xScale = (float)rect.Width() / (float)width;
            float yScale = (float)rect.Height() / (float)height;
            float ScaleIndex = (xScale >= yScale?xScale: yScale);
            rect1 = CRect(rect.TopLeft(), CSize((int)width *
ScaleIndex, (int)height * ScaleIndex));
            m_image.StretchBlt(pDc->m_hDC, rect1, SRCCOPY); //将
图片画到Picture控件表示的矩形区域
        }
    }
}
```

4.4 `ListenSocket`类

该类主要是对`MySocket`进程进行监听，然后通知`MySocket`有连接可以接收，主要函数是`OnAccept`函数：

```
void ListenSocket::OnAccept(int nErrorCode)
{
    // TODO: 在此添加专用代码和/或调用基类
    CSMTPServerDlg* dlg = (CSMTPServerDlg*)theApp.m_pMainWnd;
    dlg->m_log.AddString(_T("***收到连接请求"));
    dlg->m_log.AddString(_T("***建立连接"));
    MySocket* pSocket = new MySocket();
    if (Accept(*pSocket))
    {
        pSocket->AsyncSelect(FD_WRITE);
    }

    CAsyncSocket::OnAccept(nErrorCode);
}
```

4.5 解码头文件

解码主要是根据base64的原理进行解码，把4个字符变为3个字符，因为需要 对位进行补充与删除，所以使用位运算。然后再根据映射表查找到相应的字符。具体代码如下：

`Bash64.h`

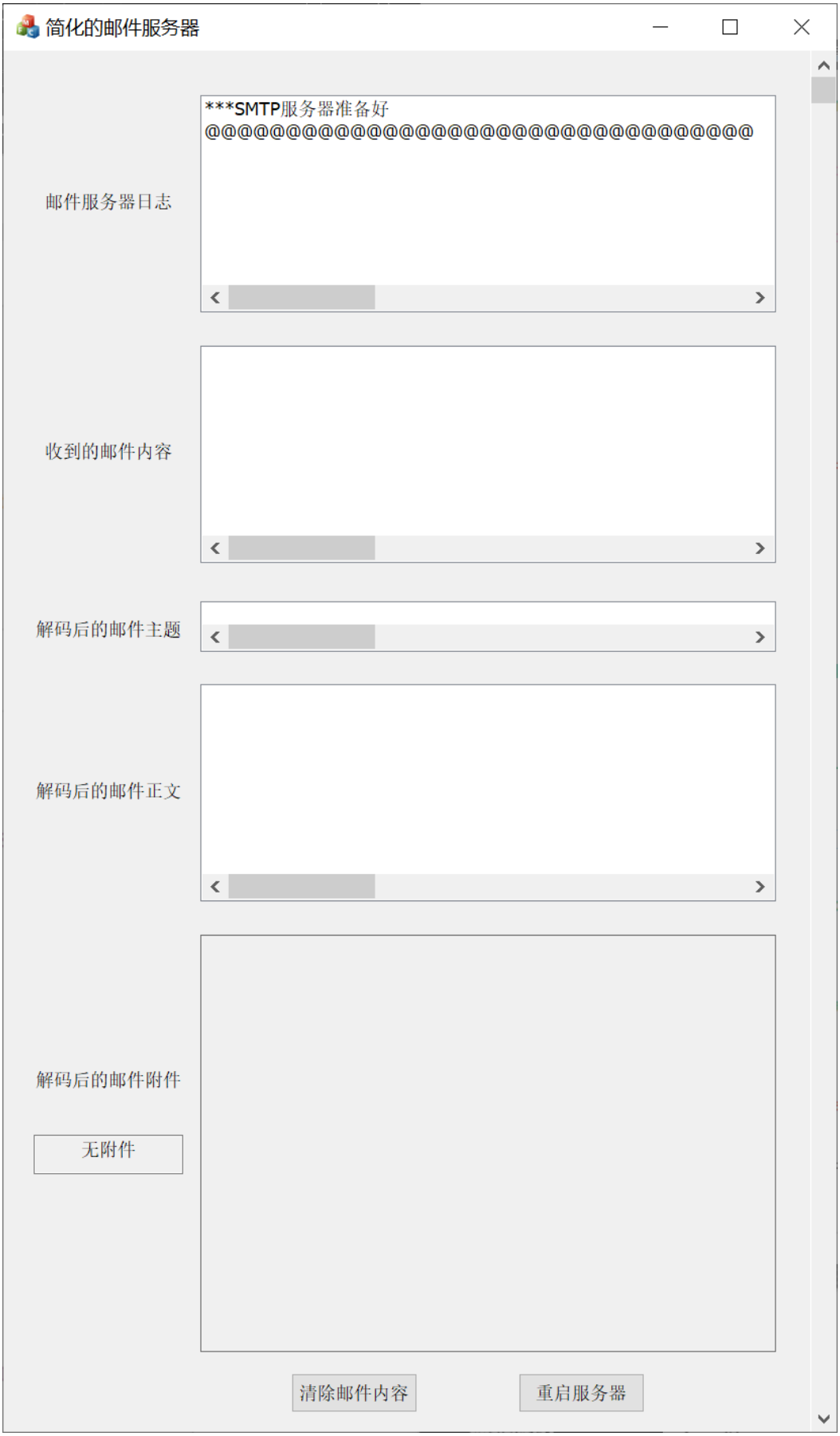
```
#pragma once


#ifndef __BASE64_H__
#define __BASE64_H__
```

```
#include <iostream>
#include <string>

static const std::string base64_chars =
"ABCDEFGHIJKLMNOPQRSTUVWXYZ"
"abcdefghijklmnopqrstuvwxyz"
"0123456789+/";

static inline bool is_base64(const char c)
{
    return (isalnum(c) || (c == '+') || (c == '/'));
}


std::string base64_decode(std::string const& encoded_string)
{
    int in_len = (int)encoded_string.size();
    int i = 0;
    int j = 0;
    int in_ = 0;
    unsigned char char_array_4[4], char_array_3[3];
    std::string ret;

    while (in_len-- && (encoded_string[in_] != '=') &&
is_base64(encoded_string[in_])) {
        char_array_4[i++] = encoded_string[in_]; in_++;
        if (i == 4) {
            for (i = 0; i < 4; i++)
                char_array_4[i] =
base64_chars.find(char_array_4[i]);

            char_array_3[0] = (char_array_4[0] << 2) +
((char_array_4[1] & 0x30) >> 4);
            char_array_3[1] = ((char_array_4[1] & 0xf) << 4) +
((char_array_4[2] & 0x3c) >> 2);
            char_array_3[2] = ((char_array_4[2] & 0x3) << 6) +
char_array_4[3];

            for (i = 0; (i < 3); i++)
                ret += char_array_3[i];
            i = 0;
        }
    }
    if (i) {
        for (j = i; j < 4; j++)
            char_array_4[j] = 0;

        for (j = 0; j < 4; j++)
            char_array_4[j] = base64_chars.find(char_array_4[j]);

        char_array_3[0] = (char_array_4[0] << 2) +
((char_array_4[1] & 0x30) >> 4);
```

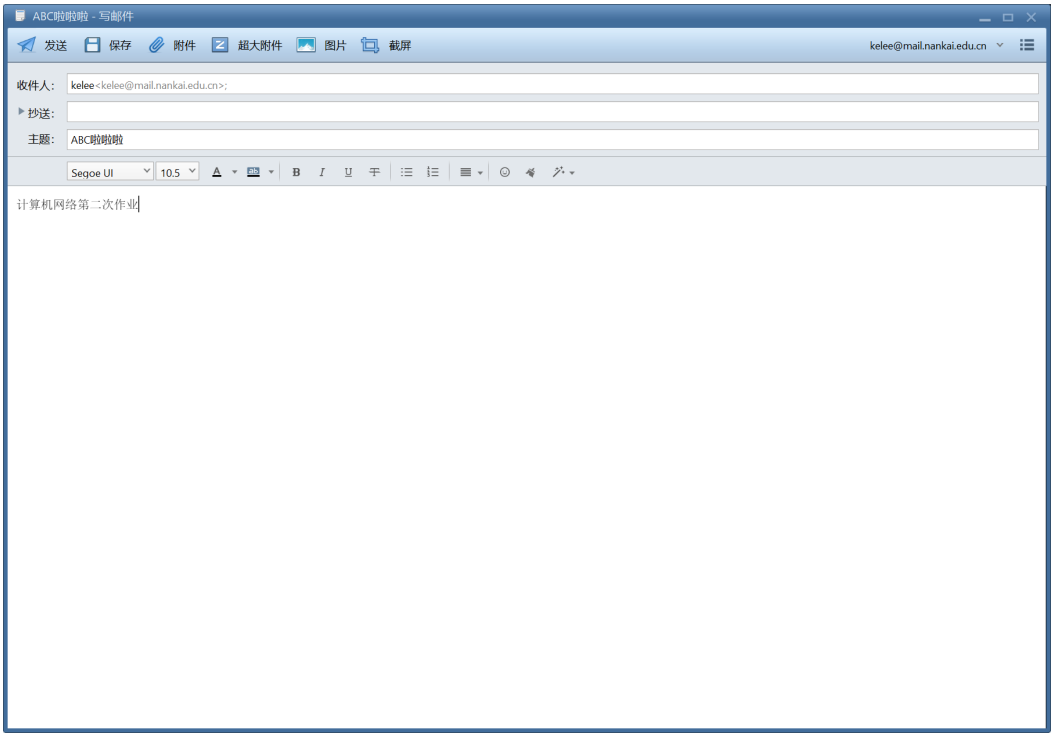
```
        char_array_3[1] = ((char_array_4[1] & 0xf) << 4) +
((char_array_4[2] & 0x3c) >> 2);
        char_array_3[2] = ((char_array_4[2] & 0x3) << 6) +
char_array_4[3];

        for (j = 0; (j < i - 1); j++) ret += char_array_3[j];
    }

    return ret;
}
#endif
```

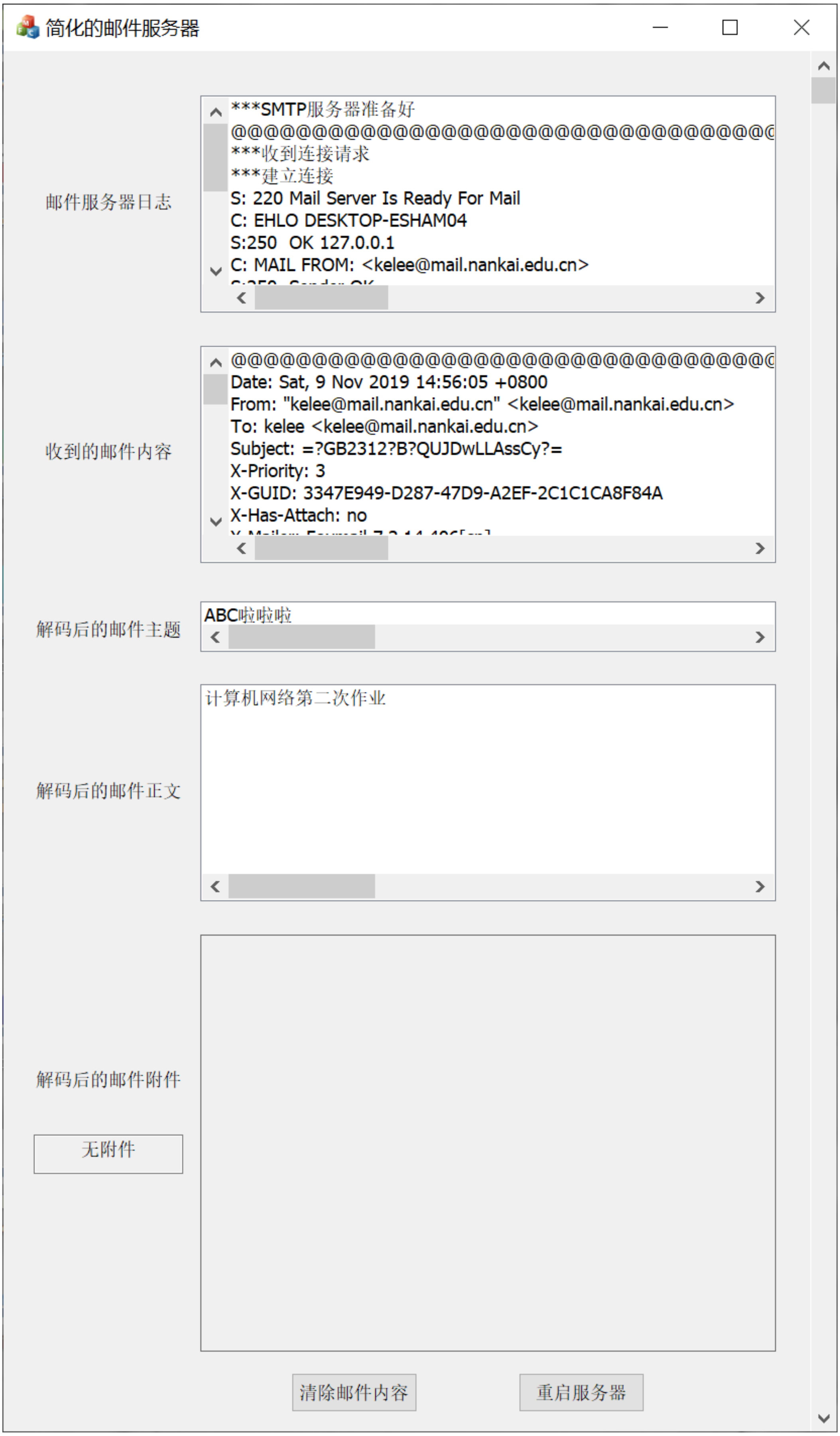
5 效果展示

5.1 程序启动


简化的邮件服务器
***SMTP服务器准备好
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
邮件服务器日志
收到的邮件内容
解码后的邮件主题
解码后的邮件正文
解码后的邮件附件
无附件
清除邮件内容
重启服务器

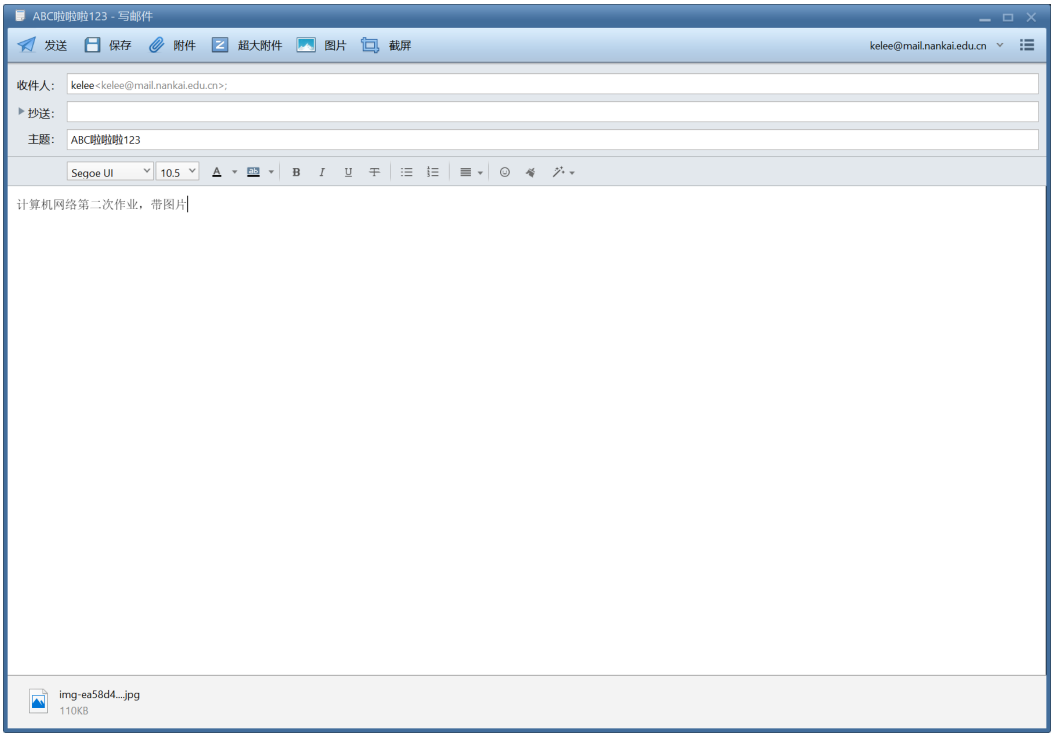
5.2 纯文本传送

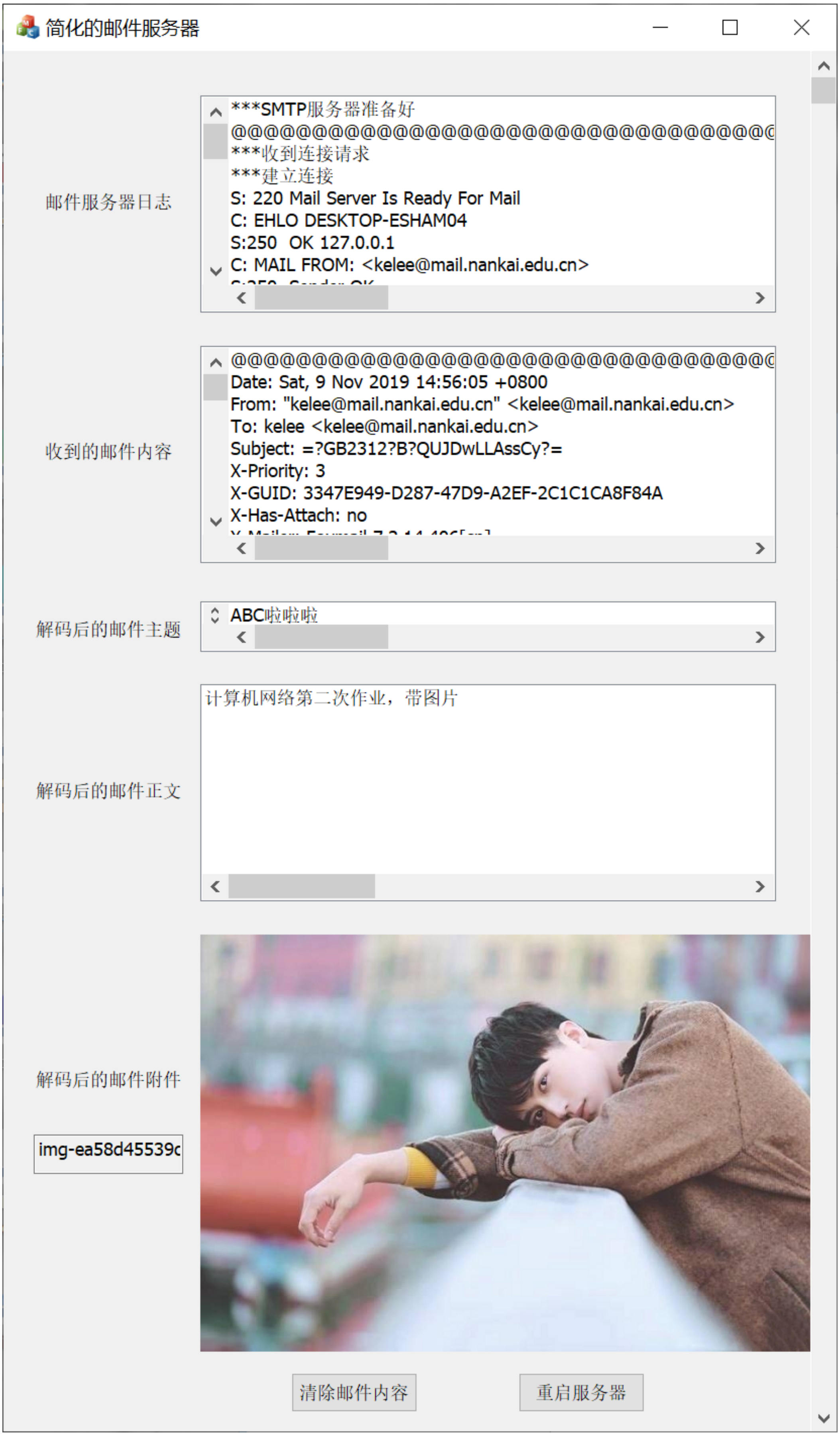

ABC啦啦啦 - 写邮件
发送
保存
附件
超大附件
图片
截屏
kelee@mail.nankai.edu.cn
收件人：
kelee<kelee@mail.nankai.edu.cn>;
抄送：
主题：
ABC啦啦啦
Segoe UI
10.5
计算机网络第二次作业



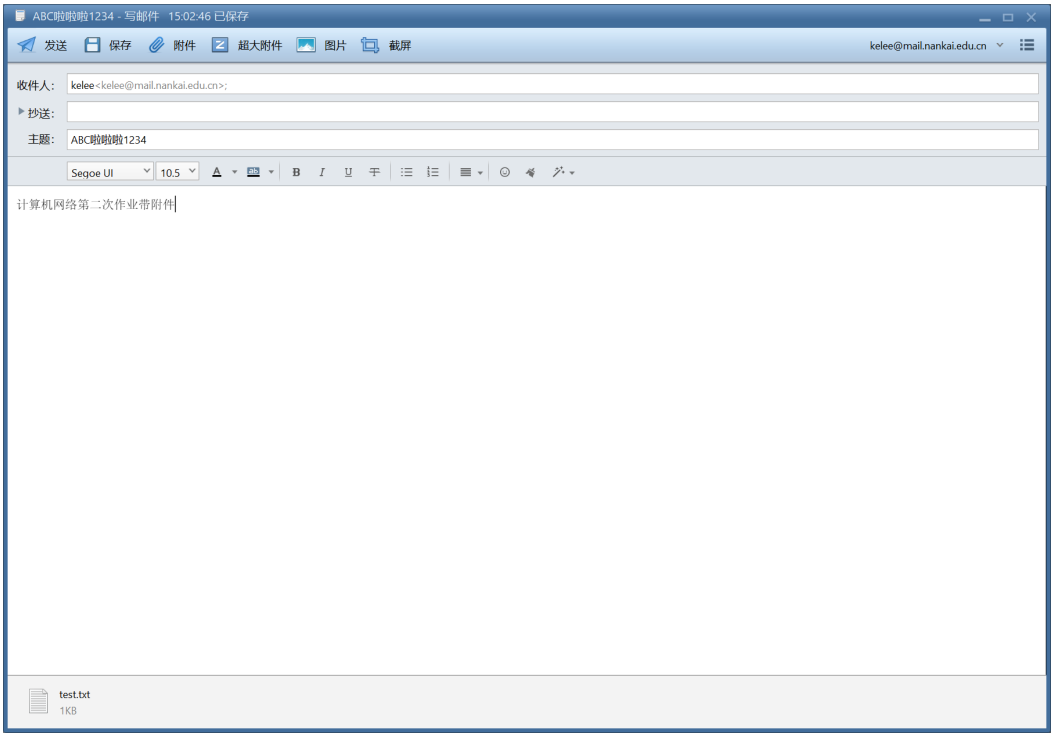
简化的邮件服务器
邮件服务器日志
***SMTP服务器准备好
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
***收到连接请求
***建立连接
S: 220 Mail Server Is Ready For Mail
C: EHLO DESKTOP-ESHAM04
S:250 OK 127.0.0.1
C: MAIL FROM: <kelee@mail.nankai.edu.cn>
收到的邮件内容
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Date: Sat, 9 Nov 2019 14:56:05 +0800
From: "kelee@mail.nankai.edu.cn" <kelee@mail.nankai.edu.cn>
To: kelee <kelee@mail.nankai.edu.cn>
Subject: =?GB2312?B?QUJDwLLAssCy?=
X-Priority: 3
X-GUID: 3347E949-D287-47D9-A2EF-2C1C1CA8F84A
X-Has-Attach: no
解码后的邮件主题
ABC啦啦啦
解码后的邮件正文
计算机网络第二次作业
解码后的邮件附件
无附件
清除邮件内容
重启服务器

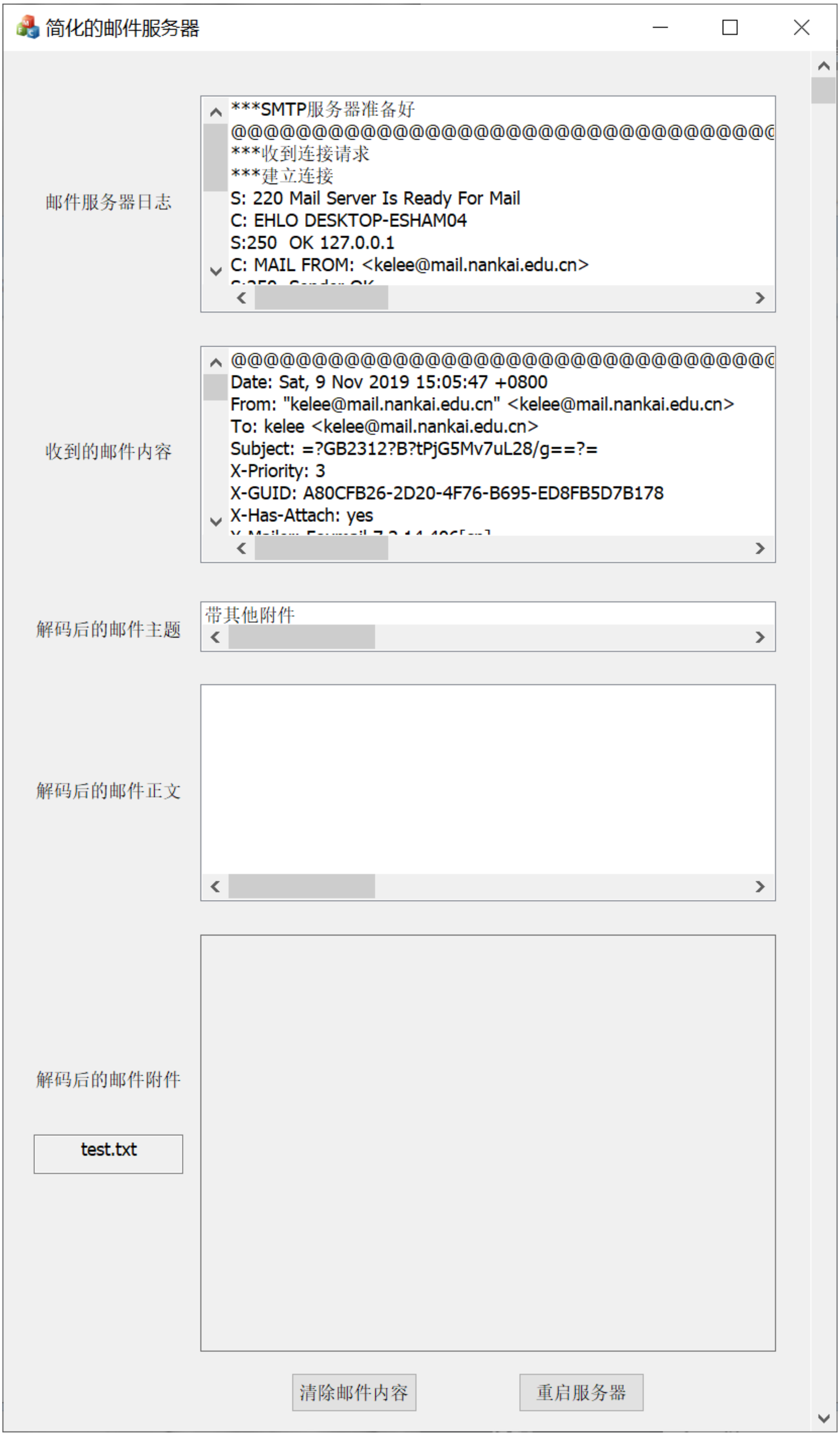
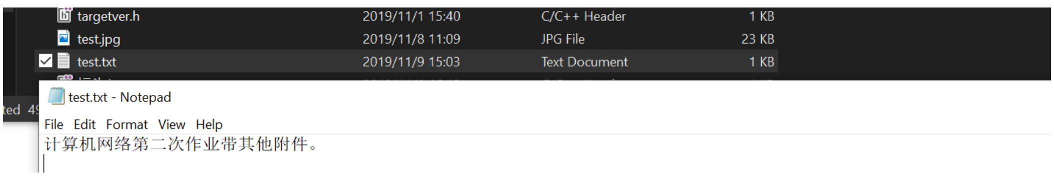
5.3 带图片传送


ABC啦啦啦123 - 写邮件
发送
保存
附件
超大附件
图片
截屏
kelee@mail.nankai.edu.cn
收件人：
kelee<kelee@mail.nankai.edu.cn>;
抄送：
主题：
ABC啦啦啦123
Segoe UI
10.5
计算机网络第二次作业，带图片
img-ea58d4....jpg
110KB

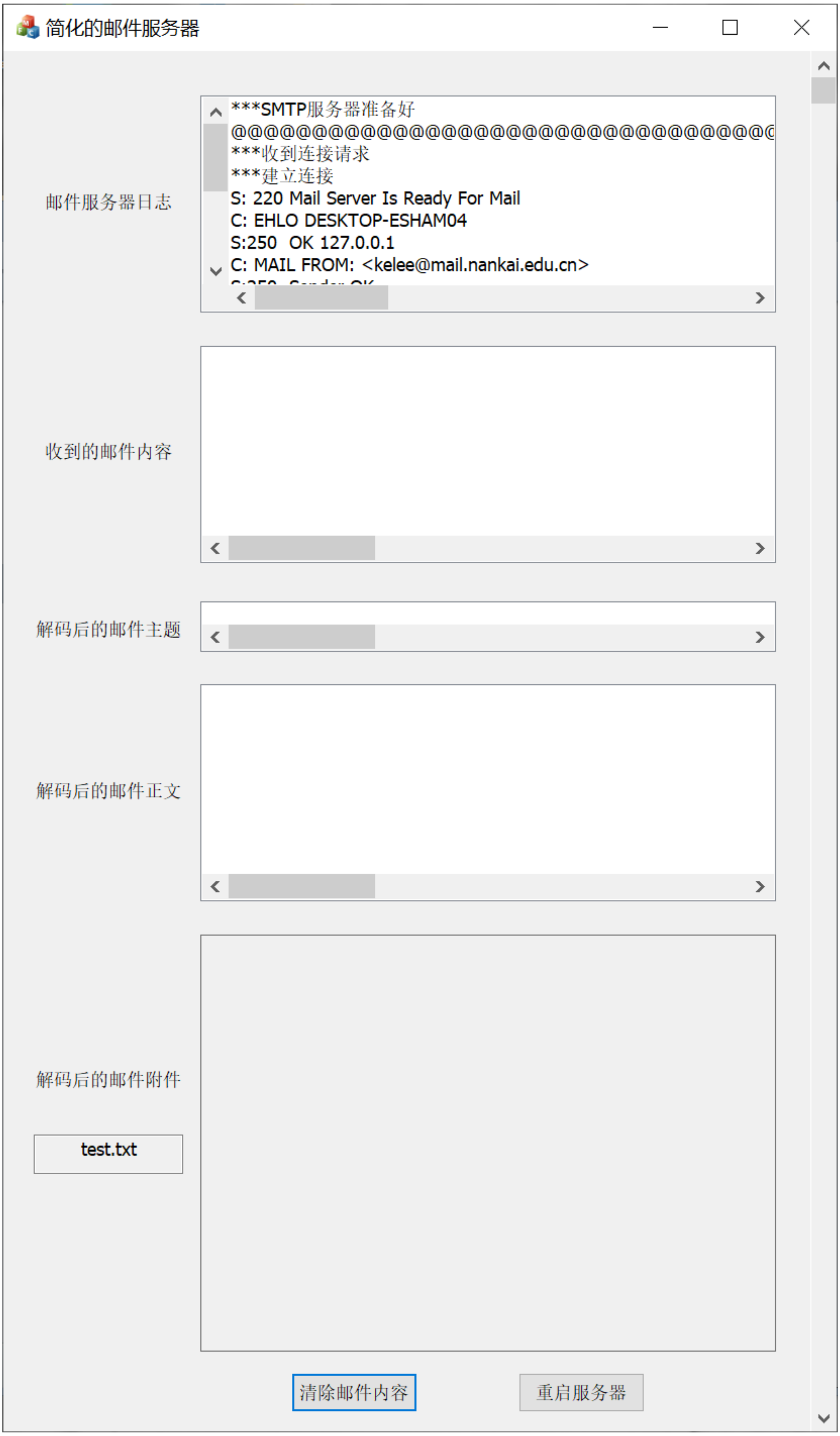

简化的邮件服务器
邮件服务器日志
***SMTP服务器准备好
***收到连接请求
***建立连接
S: 220 Mail Server Is Ready For Mail
C: EHLO DESKTOP-ESHAM04
S:250 OK 127.0.0.1
C: MAIL FROM: <kelee@mail.nankai.edu.cn>
收到的邮件内容
Date: Sat, 9 Nov 2019 14:56:05 +0800
From: "kelee@mail.nankai.edu.cn" <kelee@mail.nankai.edu.cn>
To: kelee <kelee@mail.nankai.edu.cn>
Subject: =?GB2312?B?QUJDDwLLAssCy?=
X-Priority: 3
X-GUID: 3347E949-D287-47D9-A2EF-2C1C1CA8F84A
X-Has-Attach: no
解码后的邮件主题
ABC啦啦啦
解码后的邮件正文
计算机网络第二次作业，带图片
解码后的邮件附件
img-ea58d45539c
清除邮件内容
重启服务器

5.4 带其他附件的传送

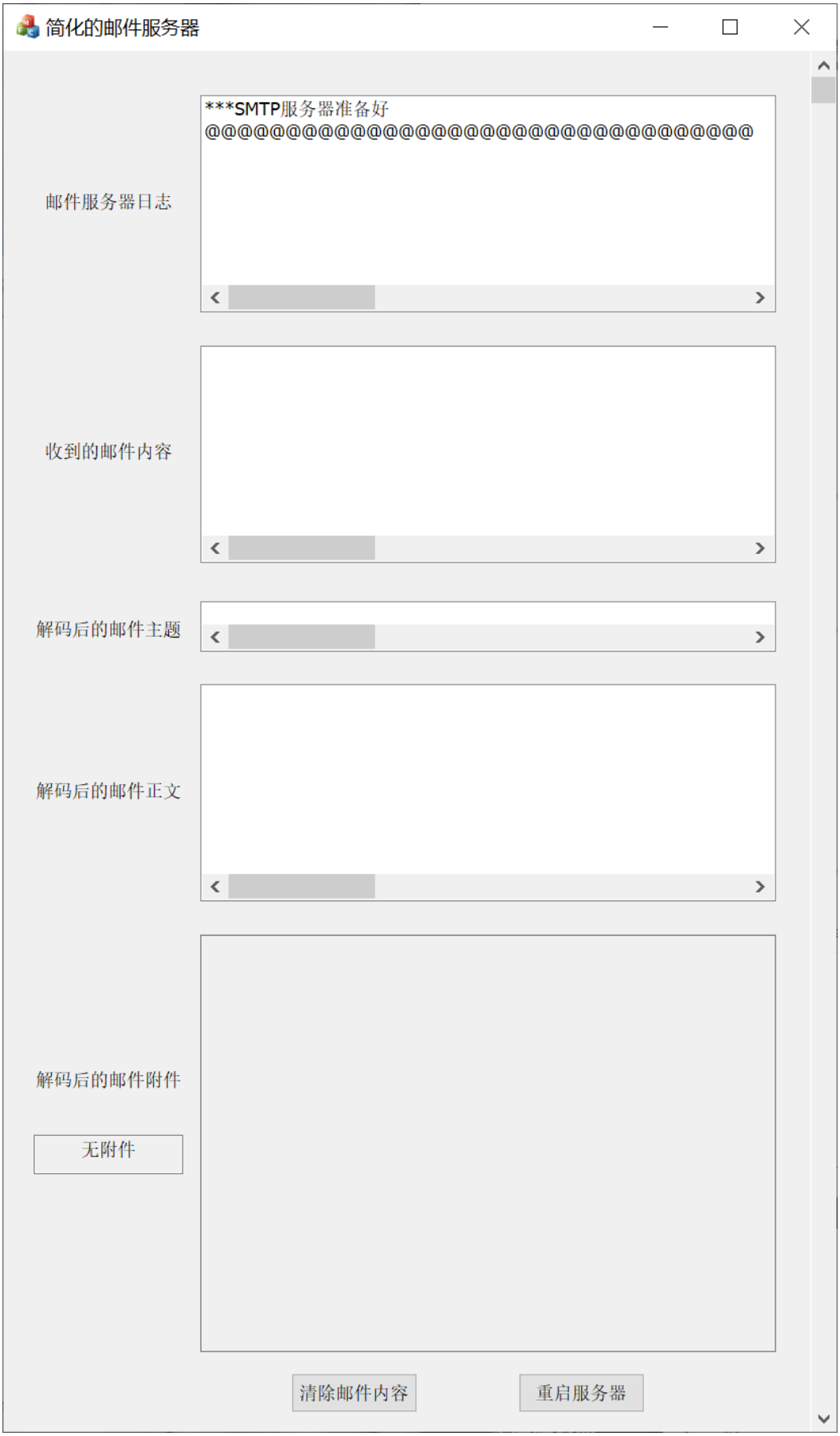

ABC啦啦啦1234 - 写邮件 15:02:46 已保存
发送
保存
附件
超大附件
图片
截屏
kelee@mail.nankai.edu.cn
收件人:
kelee<kelee@mail.nankai.edu.cn>;
抄送:
主题:
ABC啦啦啦1234
Segoe UI
10.5
计算机网络第二次作业带附件
test.txt
1KB


简化的邮件服务器
邮件服务器日志
***SMTP服务器准备好
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
***收到连接请求
***建立连接
S: 220 Mail Server Is Ready For Mail
C: EHLO DESKTOP-ESHAM04
S:250 OK 127.0.0.1
C: MAIL FROM: <kelee@mail.nankai.edu.cn>
收到的邮件内容
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Date: Sat, 9 Nov 2019 15:05:47 +0800
From: "kelee@mail.nankai.edu.cn" <kelee@mail.nankai.edu.cn>
To: kelee <kelee@mail.nankai.edu.cn>
Subject: =?GB2312?B?tPjG5Mv7uL28/g==?=
X-Priority: 3
X-GUID: A80CFB26-2D20-4F76-B695-ED8FB5D7B178
X-Has-Attach: yes
解码后的邮件主题
带其他附件
解码后的邮件正文
解码后的邮件附件
test.txt
清除邮件内容
重启服务器

targetver.h 2019/11/1 15:40 C/C++ Header 1 KB
test.jpg 2019/11/8 11:09 JPG File 23 KB
test.txt 2019/11/9 15:03 Text Document 1 KB
test.txt - Notepad
File Edit Format View Help
计算机网络第二次作业带其他附件。

5.5 清除邮件内容


简化的邮件服务器
邮件服务器日志
***SMTP服务器准备好
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
***收到连接请求
***建立连接
S: 220 Mail Server Is Ready For Mail
C: EHLO DESKTOP-ESHAM04
S:250 OK 127.0.0.1
C: MAIL FROM: <kelee@mail.nankai.edu.cn>
收到的邮件内容
解码后的邮件主题
解码后的邮件正文
解码后的邮件附件
test.txt
清除邮件内容
重启服务器

5.6 重启服务器


简化的邮件服务器
邮件服务器日志
***SMTP服务器准备好
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
收到的邮件内容
解码后的邮件主题
解码后的邮件正文
解码后的邮件附件
无附件
清除邮件内容
重启服务器