# SMTP服务器

计算机学院 1711436 皮春莹

## 实验目的

观察电子邮件应用程序与SMTP邮件服务器的命令交互过程。编写简化的SMTP服务器，响应客户SMTP命令，将命令的交互过程和收到的邮件显示到屏幕上，另外，对附件中含有base64编码的图片或文本文件进行解码并显示。拓展功能：保存收到的邮件，支持历史查看功能，支持多种图片格式（.jpg、.png、.bmp）。

## 实验环境

本程序编译环境是:Visual Studio 2017；系统环境是:Windows 10(64位)；使用了 MFC 图形化界面框架，测试用的 STMP 邮件客户端是Foxmail。

## 效果展示

#### 功能区介绍

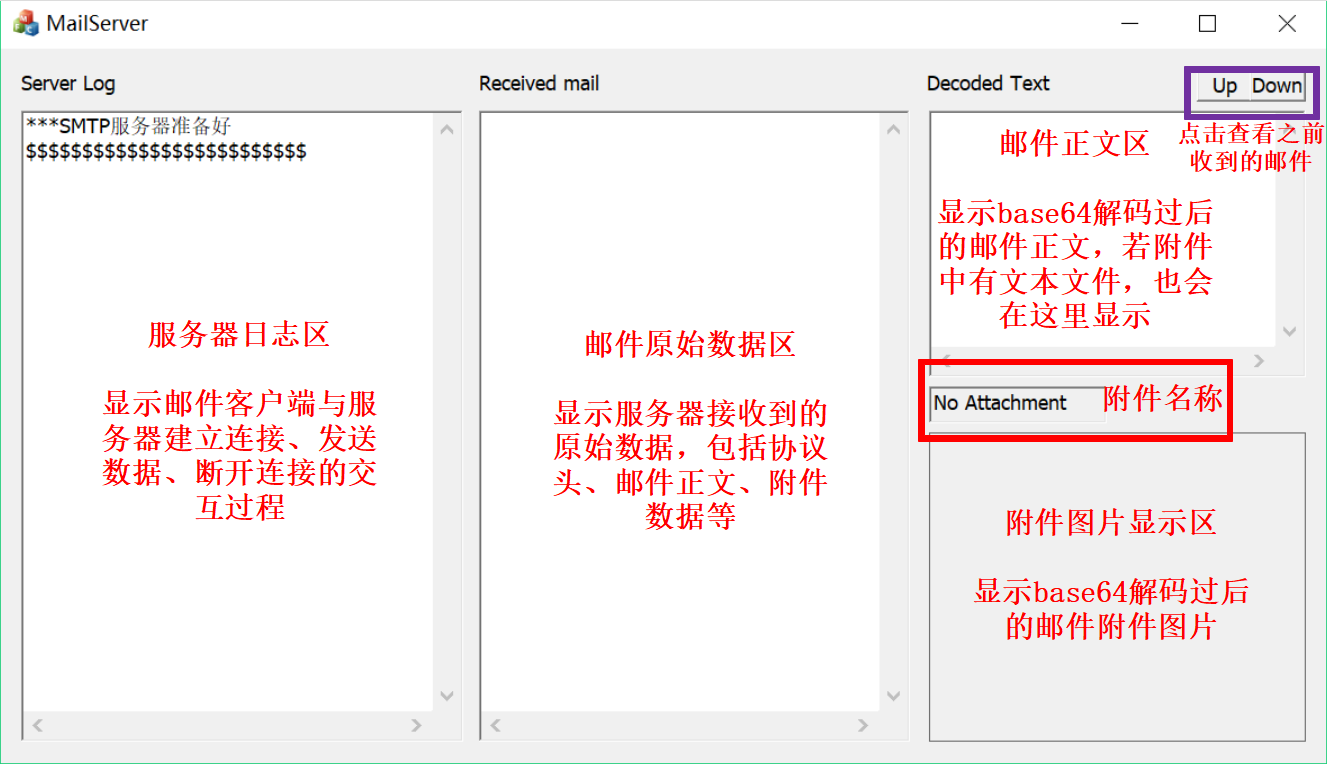


图 1 功能区介绍

#### 发送无附件的邮件

在Foxmail邮件客户端发送如下内容邮件：

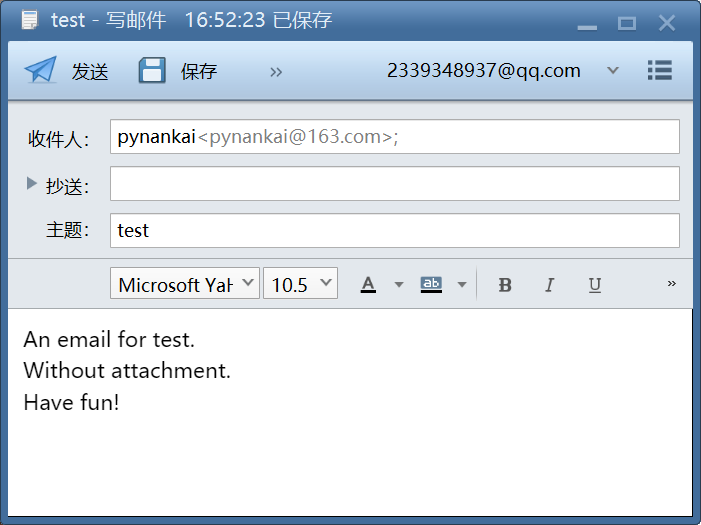


图 2 无附件邮件的Foxmail客户端

点击发送过后，在编写的简化 SMTP 邮件服务器日志上显示如下：

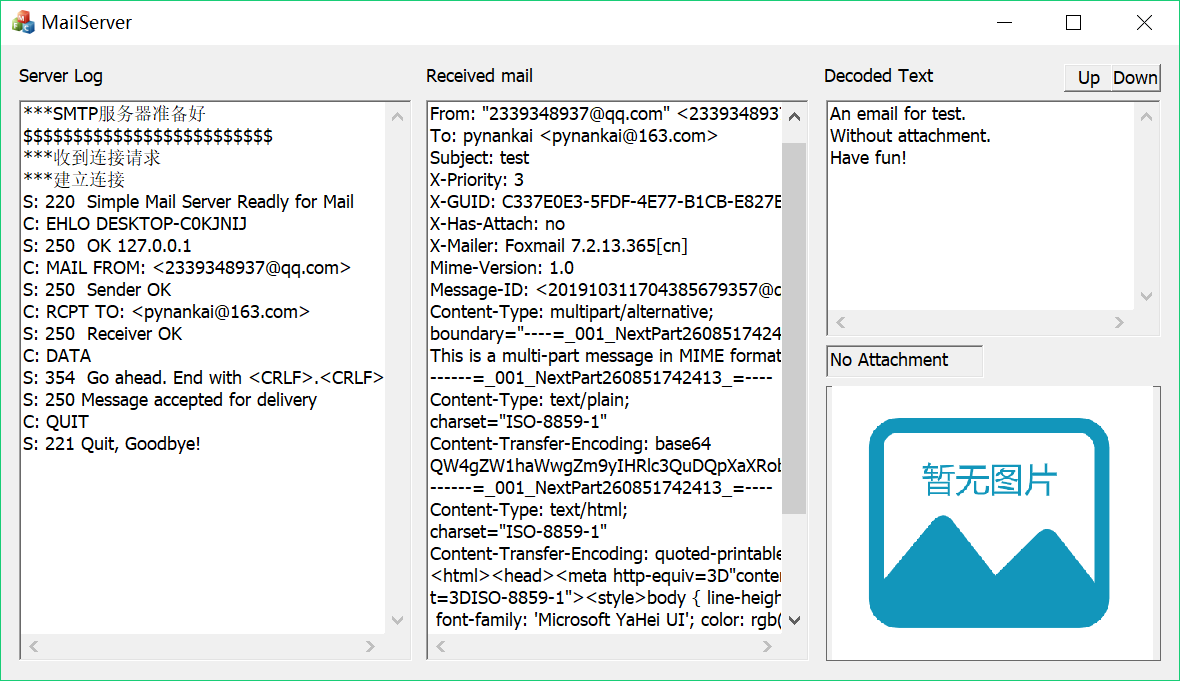


图 3 无附件邮件的服务器日志

#### 发送带有图片附件的邮件

在Foxmail邮件客户端发送如下内容邮件：

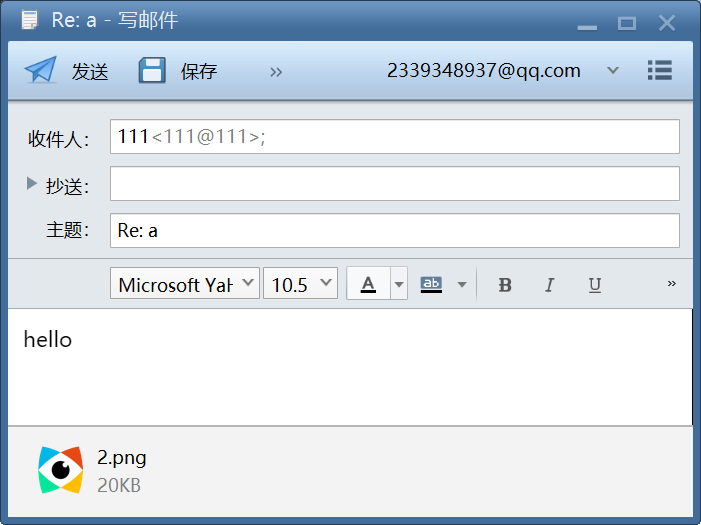


图 4 带有图片的邮件-Foxmail

点击发送过后，在编写的简化 SMTP 邮件服务器日志上显示如下：

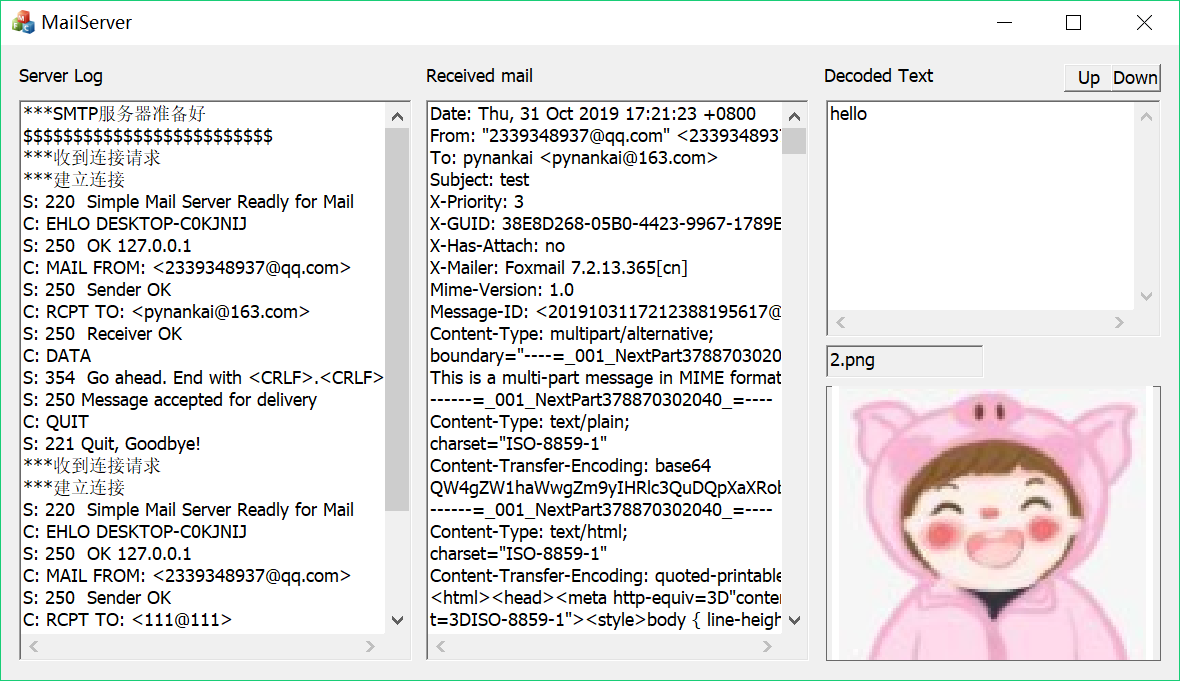


图 5 带有图片的邮件-服务器

#### 发送带有文本文件的附件的邮件

在Foxmail邮件客户端发送如下内容邮件：

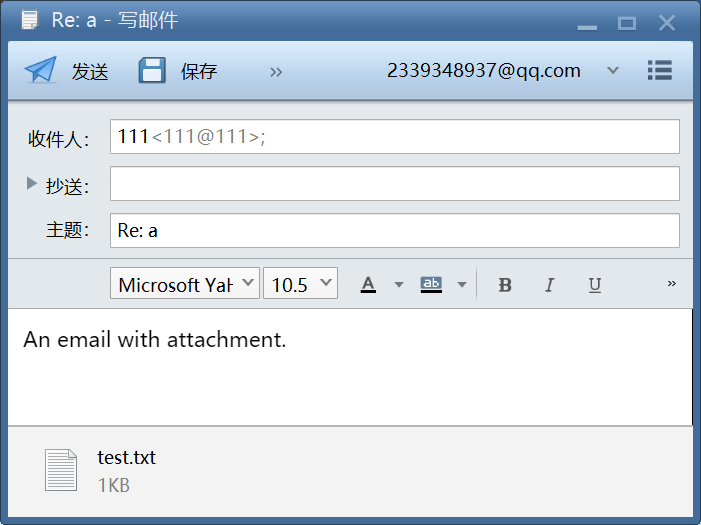


图 6 Foxmail邮件客户端

点击发送过后，在编写的简化 SMTP 邮件服务器日志上显示如下：

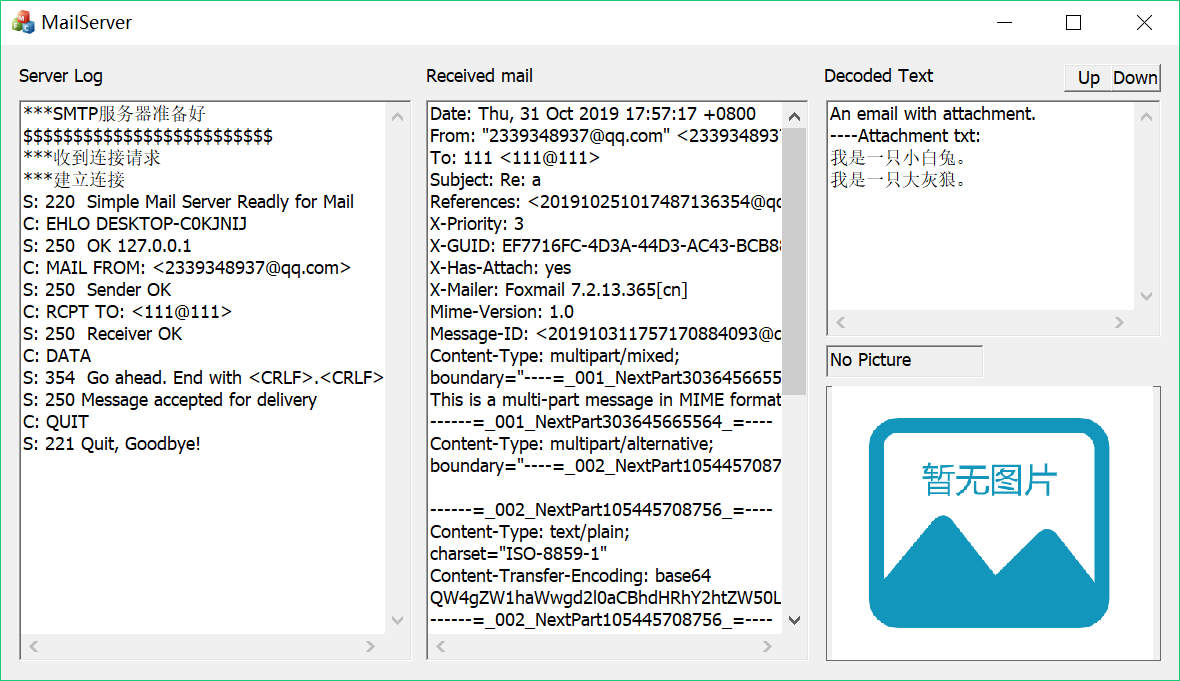


图 7 SMTP 邮件服务器日志

邮件中的内容只有“An email with attachment.”，由于这封邮件的附件中含有文本文件test.txt，因此程序会将文本文件的内容也进行解码，显示在邮件正文的下面，即“----Attachment txt:”下面的内容。

#### 查看之前接收的邮件内容

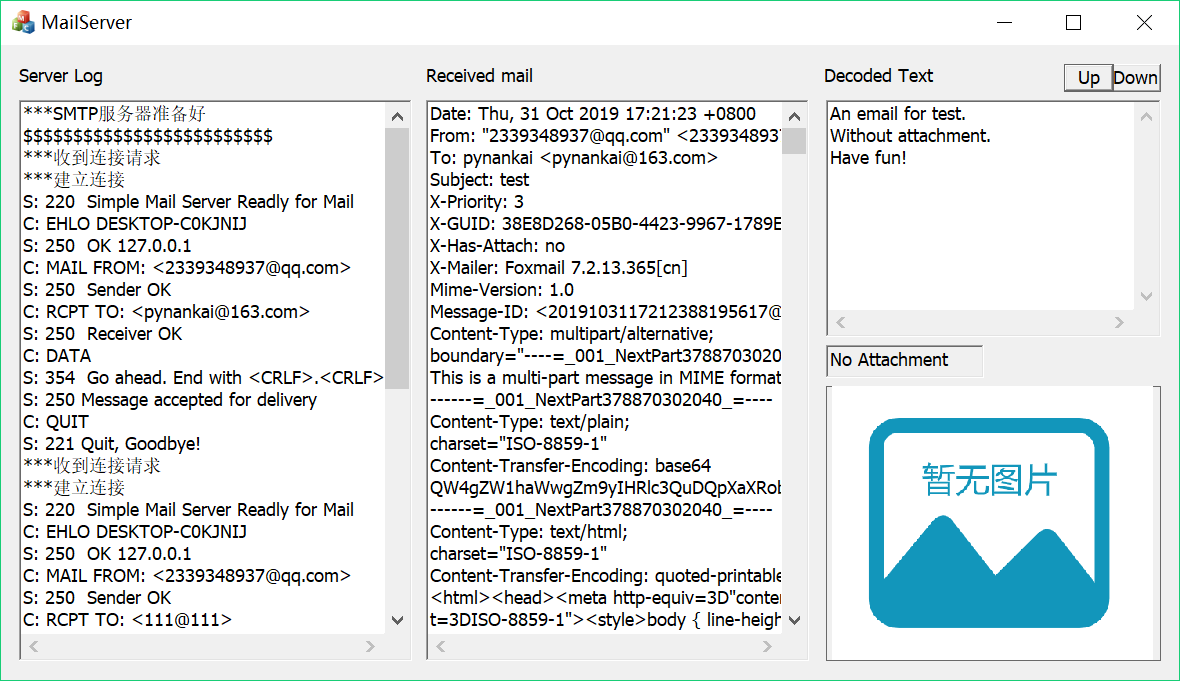


图 8 查看之前接收的历史

图8中的服务器日志区的内容与图7相同，但由于点击了两次“Up”按钮，邮件正文显示的内容以及图片解码区的内容已经切换到了上上封邮件，即与图3相同的内容。

#### 多线程测试

同时发送两封邮件，Foxmail客户端如下图所示：

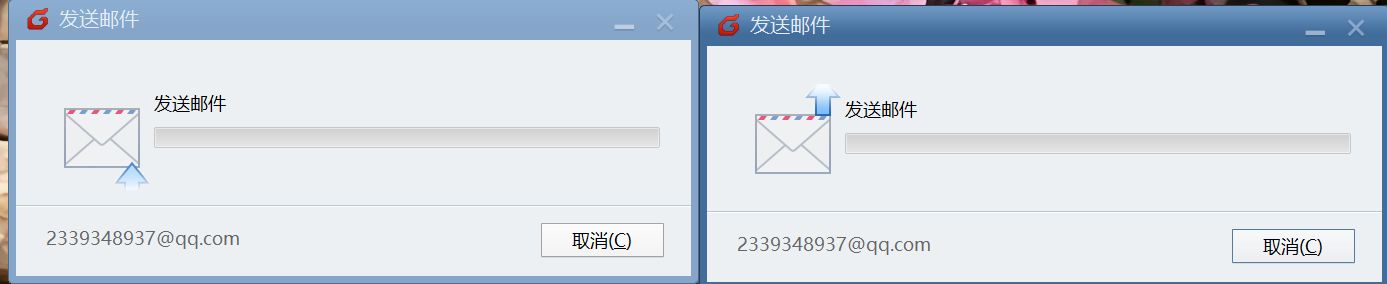


图 9 Foxmail客户端同时发送邮件

在服务器会创建两个CClientSocket来与这两个邮件数据进行交互，下图是服务器日志的截图，从图中可以看出两封邮件的数据传输是同时进行的，实现了多线程。

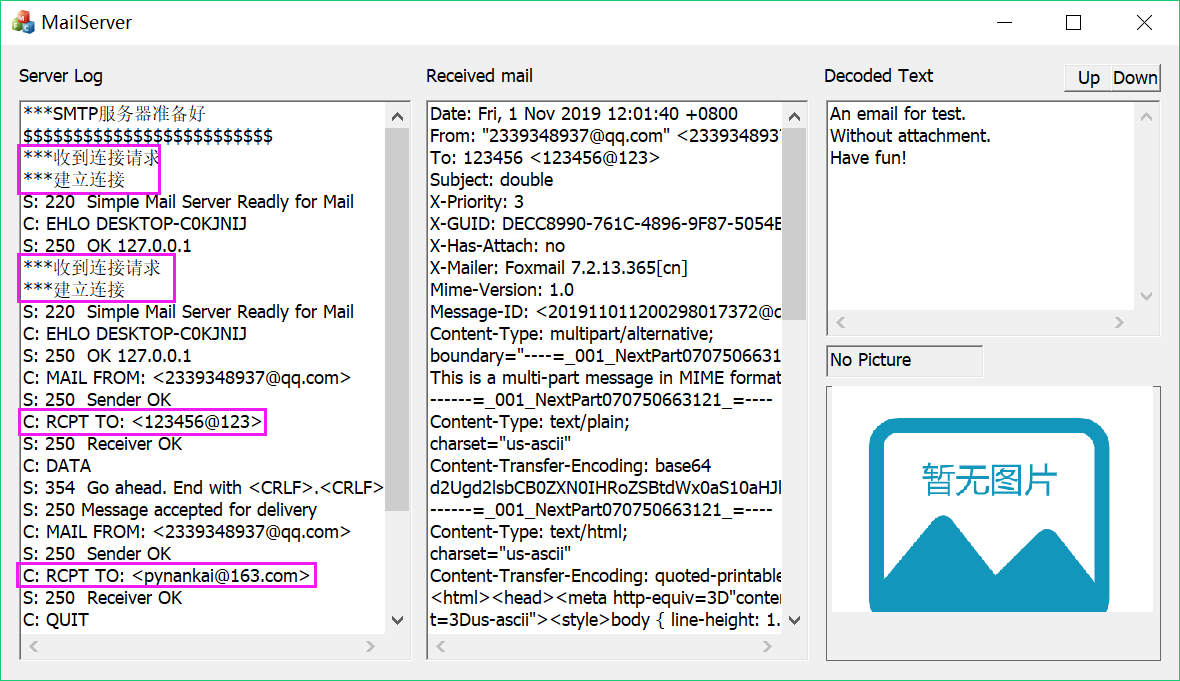


图 10 服务器多线程显示

## 实验原理

### SMTP工作原理

SMTP 服务器在 TCP 的 25 端口守候,其规定发送程序和接收程序之间的命令和应答命令和响应都是可读的 ASCII 字符串。常见的 SMTP 命令有如下：

表1 常见的 SMTP 命令

|  |  |
| --- | --- |
| **命令** | **含义** |
| HELO<主机域名> | 开始会话 |
| MAIL FROM:<发送者邮件地址> | 指出邮件发送者 |
| RCPT TO:<接收这邮件地址> | 指出邮件接收者 |
| DATA | DATA后面的数据作为发送邮件内容，以<CR><LF>.<CR><LF>结束 |
| REST | 中止当前的邮件处理 |
| NOOP | 无操作 |
| QUIT | 结束会话 |

其应答响应命令以 3 位数字开始，后面跟有该响应的具体描述，常见的应答命令有如下：

表2 常见的应答命令

|  |  |
| --- | --- |
| 命令 | 含义 |
| 220 | 服务就绪 |
| 221 | 服务关闭了传输通道 |
| 250 | 所要求的邮件动作完成，可以继续邮件对话 |
| 354 | 开始接收邮件内容输入，以<CRLF>.<CRLF>结束 |
| 500 | 命令格式错误，不可识别 |
| 502 | 命令尚未实现 |

我用Foxmail发送了一封邮件，同时用Wireshark捕获数据，如下图所示，可以直观地看到邮件客户端与发件服务器的交互过程：

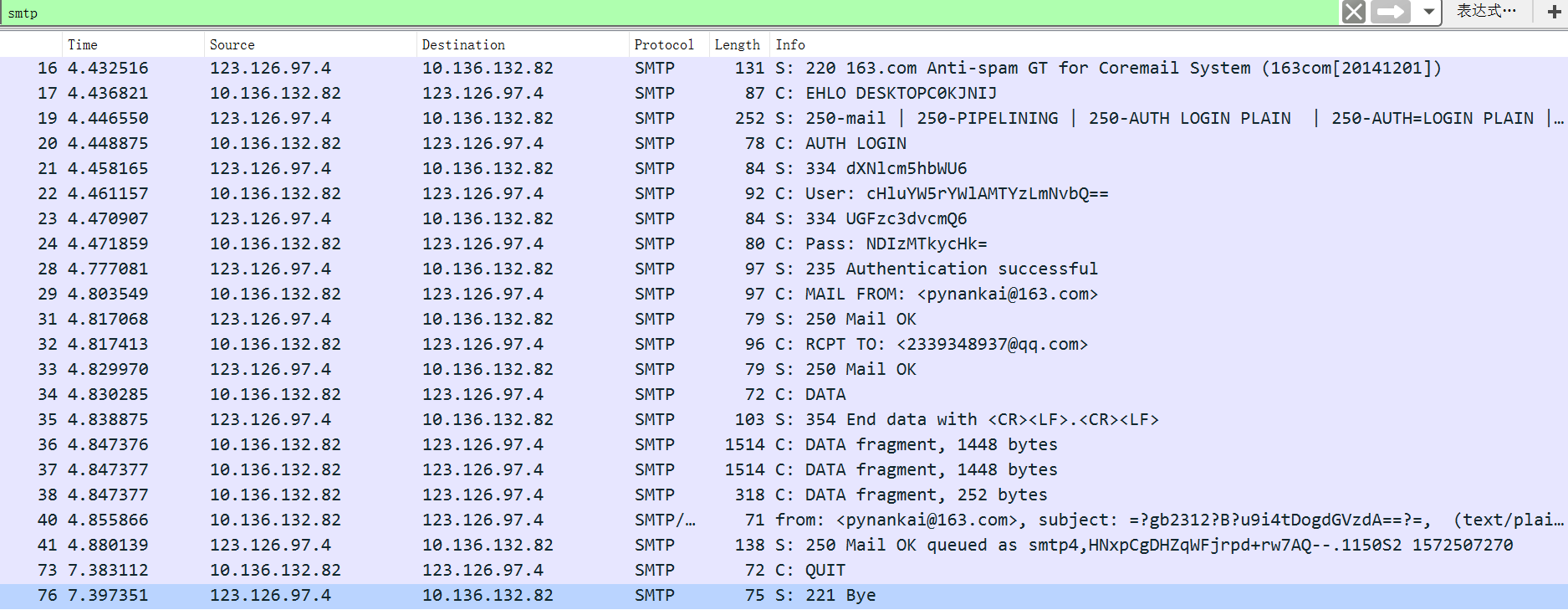


图 11 Wireshark捕获数据

客户端发送HELO报文将自己的域地址告诉给SMTP服务器，服务器接受连接请求，向客户端发送请求账号密码的报文（AUTH LOGIN）；客户端向SMTP服务器传送账号和密码，如果验证成功，可以开始报文传输；客户端使用MAIL命令将邮件发送者的名称发送给SMTP服务器，服务器做出响应；客户端使用RCPT命令发送邮件接收者地址，服务器做出响应；收到SMTP服务器的250 OK命令后，客户端使用DATA命令发送邮件数据；客户端发送QUIT命令终止连接。

在本次实验中，简化 SMTP 邮件服务器中与邮件客户端应答命令和响应命令如下：

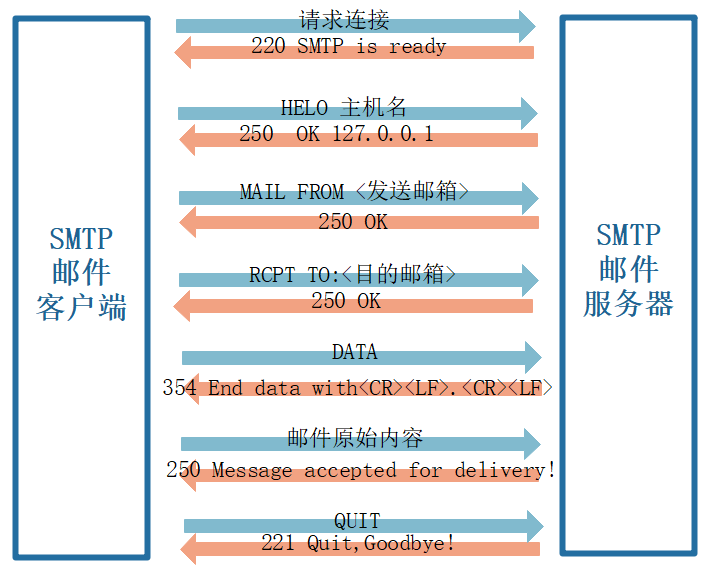


图 12 简化 SMTP服务器与客户端

### Base64编码规则

1. 将每 3 个字节做为一个整体将其划分为 4 组，每组 6 位；将 6 位的值作为索引，映射为对应的可打印 ASCII 字符。
2. 原始文件尾部处理：若剩 1 个字节，后面补 4 个比特的“0”，再分成 2 个 6 位组，映射为 2 个ASCII 字符，而后再填充两个“=”，若剩 2 个字节，后面补 2 个比特的“0”，再分成 3 个 6 位组，映射为 3 个ASCII 字符，而后再填充 1 个“=”。
3. 添加回车换行：变换后，每 76 个字符后增加一回车换行

在后面的base64.h中，按照上面的规则，反向进行base64 的解码。

## 具体实现

### 邮箱配置

在Foxmail中，将发件服务器设置为localhost或127.0.0.1，端口号为25：

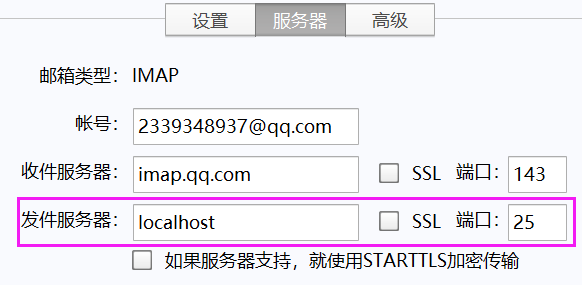


图 13 发件服务器设置

### 代码解释

#### 关于多线程

在MailServer.h中，声明一个CPtrList类型的全局变量theList，用于支持多线程，存储由控制连接ClistenSocket创建出来的一个或多个数据连接CClientSocket。

在CListenSocket::OnAccept()中使用AddTail(socket)，添加新创建的CClientSocket对象，在CClientSocket::OnClose()中，需要清理清理CClientSocket对象占用的堆空间，代码如下：

1. **void** CClientSocket::OnClose(**int** nErrorCode)
2. {
3. // TODO: 在此添加专用代码和/或调用基类
4. CPtrList &list = theApp.theList;
5. POSITION pos = list.GetHeadPosition();
6. **while** (pos)
7. {
8. **if** (list.GetAt(pos) == **this**)
9. {
10. list.RemoveAt(pos);
11. **break**;
12. }
13. list.GetNext(pos);
14. }
15. //清理对象占用的堆空间
16. **delete** **this**;
17. CAsyncSocket::OnClose(nErrorCode);
18. }

#### 用于监听端口的CListenSocket

创建CListenSocket 类继承C++中的异步 Socket 类 CAsyncSocket。重载OnAccept()函数，用于监听对应端口，默认为 25。

1. **void** CListenSocket::OnAccept(**int** nErrorCode)
2. {
3. // TODO: 在此添加专用代码和/或调用基类
4. CMailServerDlg\* dlg = (CMailServerDlg\*)theApp.m\_pMainWnd;
5. dlg->server\_log.InsertString(Count++,\_T("\*\*\*收到连接请求"));
6. dlg->server\_log.InsertString(Count++, \_T("\*\*\*建立连接"));
8. CClientSocket \*socket=**new** CClientSocket();
9. **if** (Accept(\*socket)) {
10. theApp.theList.AddTail(socket);
11. CString send = "220 SMTP is ready \r\n";
12. dlg->server\_log.InsertString(Count++, \_T("S: 220  Simple Mail Server Readly for Mail\r\n"));
13. socket->Send(send, send.GetLength());
15. }
16. **else** {
17. **delete** socket;
18. }
19. CAsyncSocket::OnAccept(nErrorCode);
20. }

在开启服务器之后，如果ClistenSocket对象监听到了客户端对服务器所在端口发出的请求， 则触发重载的 OnAccept() 函数，并创建一个CClientSocket类的对象和该客户端进行交互。每个不同客户端的请求到来时都会对应创建一个新的CClientSocket类对象，由此实现了多用户的交互。

#### 进行数据交互的CClientSocket

对于每个和客户端进行交互的 CClientSocket类对象，在收到客户端发送的消息时，就会触发重载的 OnReceive 函数。OnReceive 函数的实现，需要按照 SMTP 协议的规范，对每个客户端发来的请求回复响应的消息。

对于“HELO xxx”、“MAIL FROM xxx”、“RECT TO xxx”、“REST”以及“QUIT”等指令的响应是固定的模式，较为简单，代码行数较多，就不粘贴在这里。需要说明的是在服务器接收到“DATA”的消息之后，会将isData置为true，之后程序再接收消息时，就会进入对邮件内容的处理，代码如下：

1. **else** {//接收的是数据，在数据未接收完全时，不发送应答
2. **if** (!get.IsEmpty()) {
3. //显示在屏幕上
4. **int** pos\_begin = 0, pos\_end = get.Find('\r', 1);
5. **while** (pos\_end != -1){
6. CString temp = get.Mid(pos\_begin, pos\_end - pos\_begin);
7. dlg->rec\_mail.InsertString(rec\_line++, temp);
8. pos\_begin = pos\_end + 2;
9. pos\_end = get.Find('\r', pos\_begin + 1);
10. }
11. }
12. **int** end = get.Find("\r\n.\r\n");
13. **if** (end != -1) {
14. isData = **false**;
15. result = "250 Message accepted for delivery\r\n";
16. dlg->server\_log.InsertString(Count++, \_T("S: 250 Message accepted for delivery\r\n"));
17. t\_pic += get;
18. // find 找到base 的 begin 和 end
19. **int** pos1 = t\_pic.Find(\_T("Content-Transfer-Encoding: base64"));
20. Email mail;
21. **int** lines = 0;
22. **if** (pos1 != -1){
23. //找到base64编码的正文，进行解码
24. **int** left1 = pos1 + 33;
25. **int** pos2 = t\_pic.Find(\_T("------"), left1);
26. CString text = t\_pic.Mid(left1, pos2 - left1);
27. text.Remove('\r');
28. text.Remove('\n');
29. CString output = Text\_Decode\_Base64(text);
30. dlg->dec\_text.ResetContent();
31. **if** (!output.IsEmpty()) {
32. //显示在屏幕上
33. lines=show\_text(output,0);
34. }
35. //存入邮箱记录
36. mail.mail\_text = output;
37. }
38. **int** pos3 = t\_pic.Find(\_T("filename"));
39. **if** (pos3 != -1){
40. //如果有附件，需要显示
41. **int** left2 = pos3 + 8;
42. **int** pos4 = t\_pic.Find('"', left2);
43. **int** pos5 = t\_pic.Find('"', pos4 + 1);
44. //附件名称
45. CString filename = t\_pic.Mid(pos4 + 1, pos5 - pos4 - 1);
46. filename.Remove('\r');
47. filename.Remove('\n');
48. mail.filepath = filename;
49. **int** pos6 = t\_pic.Find(\_T("------="), pos5 + 1);
50. CString image = t\_pic.Mid(pos5 + 1, pos6 - pos5 - 1);
51. image.Remove('\r');
52. image.Remove('\n');
53. **if** (filename.Right(3) == "txt"){//附件是txt文本
54. CString outtxt = Text\_Decode\_Base64(image);
55. //显示在屏幕上
56. show\_text(outtxt,lines);
57. show\_pic("default.png");
58. dlg->UpdateData(**false**);
59. }**else**{//图片
60. Image\_Decode\_Base64(image, filename);
61. show\_pic(filename);
62. dlg->UpdateData(**false**);
63. }
64. }**else**{//未接收到附件
65. show\_pic("default.png");
66. }
67. Send(result, result.GetLength());
68. //presentID = mailID;
69. mail\_list[mailID++] = mail;
70. }**else** {t\_pic += get;}}

在数据未接收完全时，不发送应答，直到收到“\r\n.\r\n”，表示邮件正文部分接收完毕时，服务器发出响应。

接着对接收到的邮件内容进行检查。观察邮件的内容可以得知，邮件的正文部分处于“Content-Transfer-Encoding: base64”和紧接着下一个的“------ xx…xx”之间，用Find()函数找到上面两个标志的位置，再用Mid()函数提取邮件正文信息，由于base64编码的过程中，每 76 个字符后会增加一个回车换行，因此需要在这里去掉回车换行，便于后续的解码。对于处理后的邮件正文，调用base64.h中的Text\_Decode\_Base64()函数进行解码，并调用show\_text()函数将内容逐行显示在服务器日志界面上。Text\_Decode\_Base64()函数写在base64.h中，代码如下。解码时需要注意去掉末尾的“=”。

1. CString Decode\_Base64(CString input){
2. unsigned **char** \* base64 = (unsigned **char** \*)"ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/=";
3. **int** n, i, j, pad;
4. unsigned **char** \*p;
5. **static** unsigned **char** \*dst;
6. pad = 0;
7. n = input.GetLength();
8. CString src;
9. src.GetBufferSetLength(n);
10. **for** (i = 0; i<n; i++)
11. src.SetAt(i, input[i]);
12. **int** dstSize = n \* 3 / 4 + 1;
13. **while** (n>0 && src[n - 1] == '=') {
14. src.SetAt(n - 1, 0);
15. pad++;
16. n--;
17. }
18. **for** (i = 0; i<n; i++) {
19. p = (unsigned **char** \*)strchr((**const** **char** \*)base64, (**int**)src[i]);
20. **if** (!p)
21. **break**;
22. src.SetAt(i, p - (unsigned **char** \*)base64);
23. }
24. dst = (unsigned **char** \*)malloc(dstSize);
25. **for** (i = 0, j = 0; i<n; i += 4, j += 3) {
26. dst[j] = (src[i] << 2) + ((src[i + 1] & 0x30) >> 4);
27. dst[j + 1] = ((src[i + 1] & 0x0F) << 4) + ((src[i + 2] & 0x3C) >> 2);
28. dst[j + 2] = ((src[i + 2] & 0x03) << 6) + src[i + 3];
29. }**return** CString(dst).Mid(0,j-1);
30. }
32. CString Text\_Decode\_Base64(CString input){
33. CString output = Decode\_Base64(input);
34. //有的文本末尾没有换行，要稍作处理
35. **if** (output.Right(2) != "\r\n")
36. output += "\r\n";
37. **return** output;
38. }

show\_text()函数写在base64.h中，代码如下：

1. **int** show\_text(CString text,**int** i){
2. CMailServerDlg\* dlg = (CMailServerDlg\*)theApp.m\_pMainWnd;
3. **int** pos\_begin = 0, pos\_end = text.Find('\n', 1);
4. **int** k = i;
5. **if** (k)
6. dlg->dec\_text.InsertString(k++, "----Attachment txt:\r\n");
7. **while** (pos\_end != -1){
8. CString temp = text.Mid(pos\_begin, pos\_end - pos\_begin+1);
9. dlg->dec\_text.InsertString(k++, temp);
10. pos\_begin = pos\_end + 1;
11. pos\_end = text.Find('\n', pos\_begin + 1);
12. }**return** k;
13. }

除此之外，还需要在获取到的数据中，查找是否有“filename”，如果有，表明此邮件含有附件，在这里我实现的可处理的附件类型为图片或文本文件。先提取附件的内容，方法与提取邮件正文的方法相同，不再赘述。得到删除回车换行符之后的邮件附件内容，判断是图片还是文本文件。如果是文本文件，那么处理方式则与正文的解码类似；如果是图片，则调用Image\_Decode\_Base64()函数对图片内容进行解码，将结果写入磁盘文件流，然后调用show\_pic()函数，使用 CImage 类，将图片显示在SMTP服务器日志的界面上。Image\_Decode\_Base64()解码的原理与Text\_Decode\_Base64()相同，不再在这里展示，下面只展示show\_pic()的代码：

1. **void** show\_pic(CString path){
2. CMailServerDlg\* dlg = (CMailServerDlg\*)theApp.m\_pMainWnd;
3. dlg->filename = path;
4. **if** (path == "default.png")
5. dlg->filename = "No Picture";
6. CImage picture;
7. picture.Load(path);
8. **if** (!picture.IsNull()) {
9. SetStretchBltMode(dlg->rec\_pic.GetDC()->GetSafeHdc(), HALFTONE);
10. CRect dest;
11. dlg->rec\_pic.GetClientRect(&dest);
12. **float** nRatioImage = picture.GetWidth() / **static\_cast**<**float**>(picture.GetHeight());
13. **float** nRatioDest = dest.Width() / **static\_cast**<**float**>(dest.Height());          CRect rectDraw = dest;
14. **if** (nRatioImage > nRatioDest) {    //设置在空间居中显示
15. rectDraw.SetRect((dest.Width() / 2) - (rectDraw.right / 2), (dest.Height() / 2) - ((rectDraw.right / nRatioImage) / 2), (dest.Width() / 2) + (rectDraw.right / 2), (dest.Height() / 2) + ((rectDraw.right / nRatioImage) / 2));
16. }
17. **else** **if** (nRatioImage < nRatioDest) {
18. rectDraw.SetRect((dest.Width() / 2) - ((rectDraw.bottom \* nRatioImage) / 2), (dest.Height() / 2) - (rectDraw.bottom / 2), (dest.Width() / 2) + ((rectDraw.bottom \* nRatioImage) / 2), (dest.Height() / 2) + (rectDraw.bottom / 2));
19. }
20. picture.Draw(dlg->rec\_pic.GetDC()->GetSafeHdc(), rectDraw);
21. }}

#### 按钮“Up”和“Down”的绑定事件

为了实现查看过去收到的邮件的功能，需要对每次收到的邮件的主要信息进行保存，因此在Email.h中定义了Email类，定义成员变量mail\_text和filepath，分别用于保存邮件解码后的正文和附件路径。每一个Email类对象有一个唯一对应的mailID，这个对应关系我使用了map来保存。

1. **class** Email{
2. **public**:
3. Email();
4. ~Email();
5. CString mail\_text;
6. CString filepath;
7. };
8. **extern** map<**int**, Email> mail\_list;
9. **extern** **int** mailID;

每次点击“Up”或“Down”按钮时，会触发MailServerDlg中定义的CheckFormerMail()函数或CheckBackMail()函数，将当前的mailID减一或加一，找到应该显示的邮件编号，调用show\_mail()函数，将内容显示在控件上。show\_mail()函数的内容与前面介绍过的show\_text()和show\_pic()类似，不再赘述。

## 附录

**实验中遇到的问题及解决方法：**

1. 在控件中打印出来的字符串与代码中出现的顺序不一致：

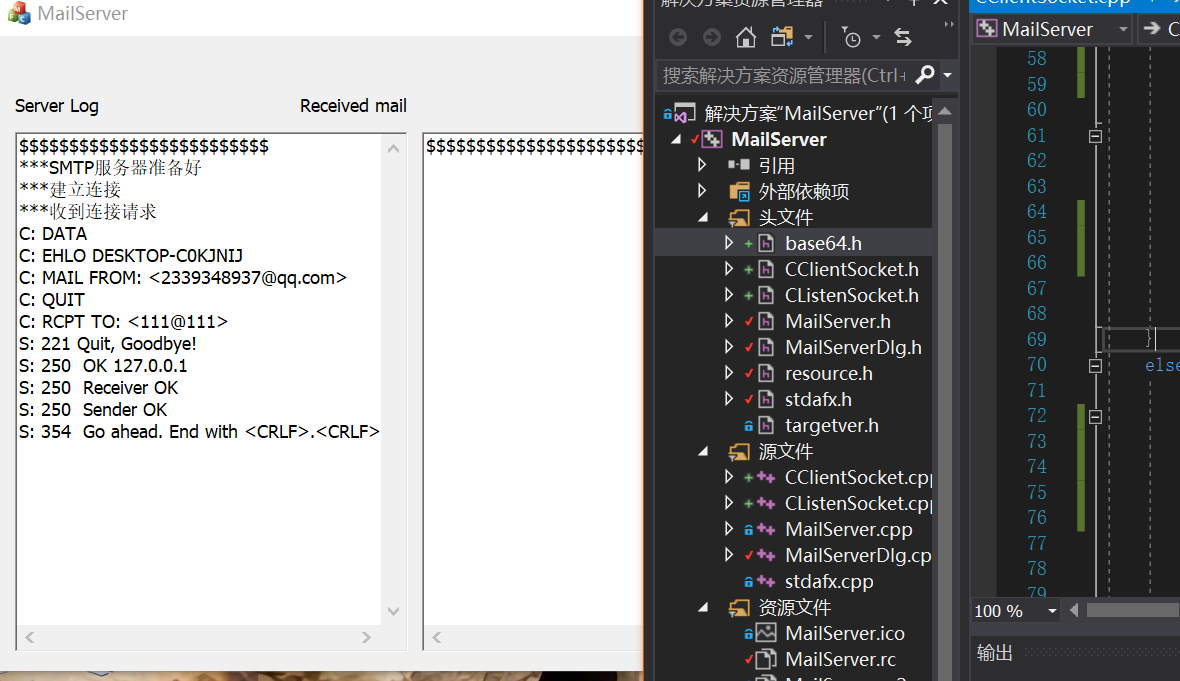


图 14 问题一

解决方法：使用一个全局变量Count记录行号，使用ClistBox控件的InsertString()函数，指定当前字符串应输出的位置。

1. 在显示解码后的邮件内容时，如果遇到双字节字符，比如中文，则可能会截断乱码：

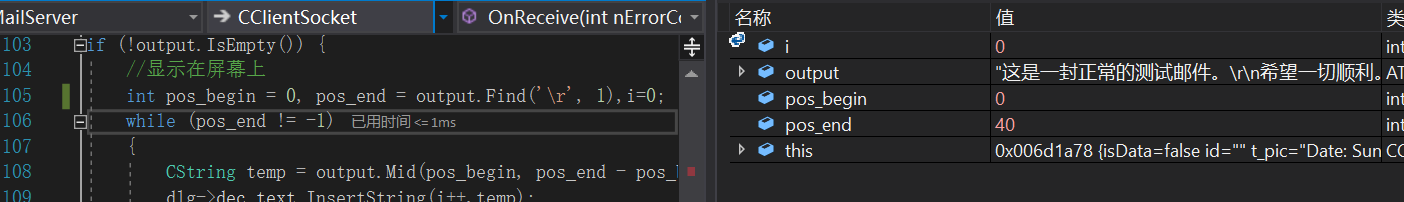


图 15 中文截断乱码

解决方法：转换查找的字符——‘\n’，正好是双字节的位置。但要注意之后查找的起始位置是pos\_end+1，而不是之前的pos\_end+2。

#### 其他

CAsyncSocket::OnAccept

Called by the framework to notify a listening socket that it can accept pending connection requests by calling the Accept member function.

CAsyncSocket::OnReceive

Called by the framework to notify this socket that there is data in the buffer that can be retrieved by calling the Receive member function.

CAsyncSocket::OnSend

Called by the framework to notify the socket that it can now send data by calling the Send member function.

char \*strchr(const char \*str, int c) 在参数str 所指向的字符串中搜索第一次出现字符c（一个无符号字符）的位置。