南京航空航天大学

计算机网络课程实验报告

工程型实验--发送邮件

学生姓名	陈明富
学　　号	161710228
学　　院	计算机科学与技术学院
专　　业	计算机科学与技术
班　　级	1617102
指导教师	燕雪峰

二〇二〇年五月

目录

实验目的与要求

要求学生掌握 Socket 编程中流套接字的技术，以及邮件的发送。
1.要求学生掌握利用 Socket 进行编程的技术；
2.不能采用现有的工具，必须自己一步一步，根据协议进行操作；
3.要求每一次操作，必须点击下一步才能继续；
4.了解邮件发送格式；
5.必须采用图形界面，可以编辑发送内容；
6.可选，建立自己的邮件服务器；
7.发送邮件可以发给自己的邮件服务器，也可以发给已知邮件服务器；
8.要求可以查看得到发送的邮件。

实验内容

电子邮件概述

电子邮件系统具有三个主要组成构件，即**用户代理、邮件服务器以及邮件协议**。主要的应用层协议就是简单邮件传送协议 SMTP(Simple Mail Transfer Protocol)、邮局协议 POP3(Post Office Protocol)以及 1993 年提出的通用互联网邮件扩充 MIME(Multipurpose Internet Mail Extensions)。

邮件服务器是电子邮件基础结构的核心，其功能是发送和接收邮件，同时还向发件人报告邮件传送的结果（已交付、被拒绝、丢失等），按照客户服务器方式工作。

邮件服务器使用两种不同的协议。一种协议用于用户代理向邮件服务器发送邮件或者在邮件服务器之间发送邮件，如本实验中的 SMTP；另一种协议用于用户代理从邮件服务器读取邮件，如邮局协议 POP3。

用户代理 UA（User Agent）是用户与电子邮件系统的接口，大多数情况下它就是运行在用户电脑中的一个程序。因此用户代理又称为电子邮件客户端软件。用户代理向用户提供一个友好的接口（目前主要是窗口界面）来发送和接收邮件。

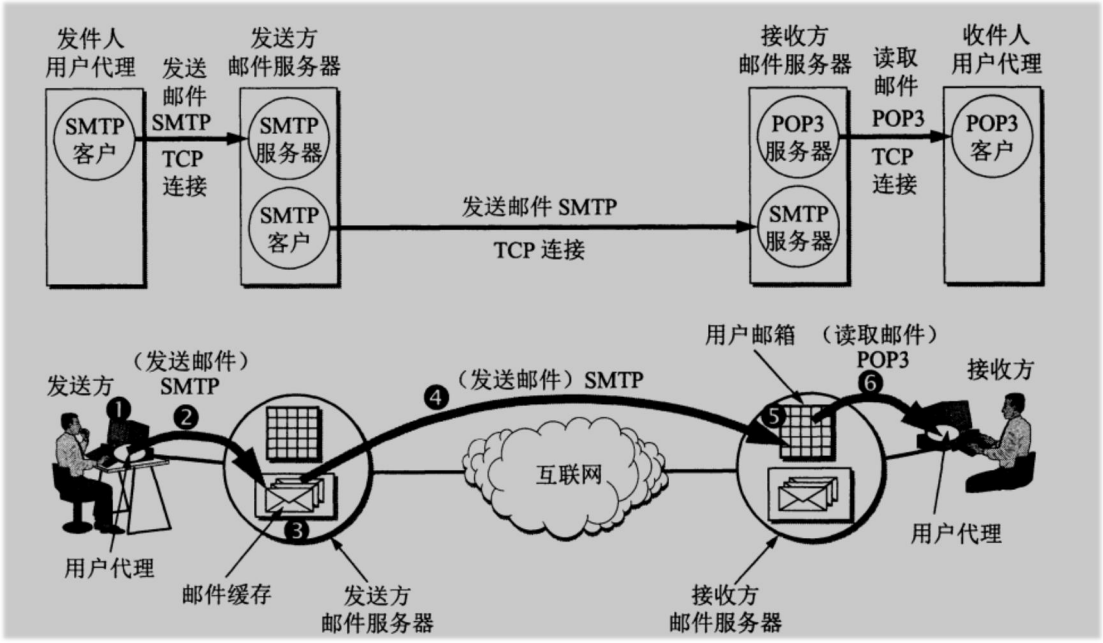
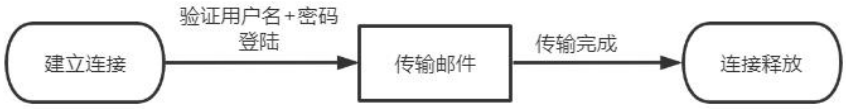


图-电子邮件最主要的组成构件

邮件发送与 SMTP 协议



SMTP 规定了在两个相互通信的 SMTP 进程之间应如何交换信息。SMTP 规定了 14 条命令和 21 种应答信息(部分可见参考与附录)。每条命令用几个字母组成，每一种应答信息一百只有一行，有 3 位数字的代码开始，SMTP 的流程可以简单抽象成上图。

通过 Windows 命令行下的 Telnet，可以简单的实现 SMTP 发送邮件的简单过程。

1.建立 TCP 连接：telnet smtp.qq.com 25 〈地址,熟知端口〉

2.客户端发送 HELO 命令标识发件人自己的身份，并得到服务器返回的状态

3.客户端 AUTH LOGIN 登陆，Base64 编码用户的邮件地址和密码

4.邮件传送是从 MAIL 命令开始的，传送邮件的主体

5.以<CR><LF>.<CR><LF>作为邮件内容的结束

6.连接释放 QUIT 命令

```
//C:Client(本地客户) S:Server(SMTP服务器)
//以下为命令行对话框
C: HELO PINK            // HELO后的内容任意，但是至少有一个空格Warning.1
S: 250-newxmesmtplogicsvrszc7.qq.com-100.77.8.215-30206205
250-SIZE 73400320
250 OK
C: AUTH LOGIN           // 验证登陆
S: 334 VXNlcm5hbWU6
C: Base64 of username   // 输入经过base64加密后的用户名
S: 334 UGFzc3dvcmQ6
C: Base64 of password   // 输入经过base64加密后的密码or授权码)
S: 235 Authentication successful
C: MAIL FROM:<youemail>     // 发件人 Warning.2
S: 250 OK
C: RCPT TO:<ReceiverEmail> // 收件人
S: 250 OK
C: DATA                     // 邮件
S: 354 End data with <CR><LF>.<CR><LF>. // 以回车换行+ . +回车换行结束
C: TO:
C: SUBJECT:
C:                          // CRLF 表示以下为邮件内容  Warning.3
C: ni,hao
C: good bye!
C: .                        // 表结束
S: 250 OK: queued as.
/*
```

图-telnet 使用 SMTP 发送电子邮件

使用上述命令时的问题(图中的 Warning)：1.很奇怪的输入问题，输入过程回退或者输入法为中文时可能会返回 502 Invalid(见错误码),输入时需注意。2.MAIL FROM 和 RCPT TO 命令的格式容易出错，使用<>(很多相关博客中并没有<>，我怀疑他们的是错的)。3.此处需要有一个 CRLF，这是 MAIL 邮件的格式。

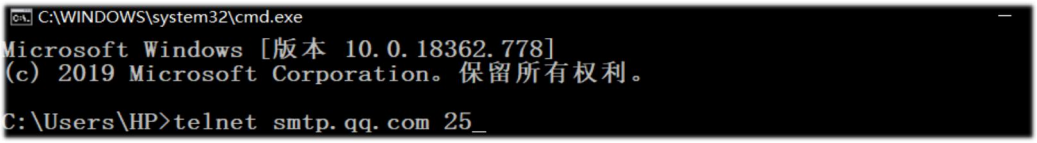


图-telnet 发送电子邮件

关于邮件读取与 pop3 协议

POP 协议是一种用于接收电子邮件的协议，发送端的邮件根据 SMTP 协议将被转发给一直处于插电状态的 POP 服务器。客户端再根据 POP 协议从 POP 服务器接收对方发来的邮件。在这个过程中，为了防止他人盗窃邮件内容，还要进行用户验证。具体过程是：邮件发送到服务器上，电子邮件客户端调用邮件客户机程序以连接服务器，并下载所有未阅读的电子邮件。这种离线访问模式是一种存储转发服务，将邮件从邮件服务器端送到个人终端机器上，一般是 PC 机或 MAC。一旦邮件发送到 PC 机或 MAC 上，邮件服务器上的邮件将会被删除。但目前的 POP3 邮件服务器大都可以“只下载邮件，服务器端并不删除”，也就是改进的 POP3 协议。

图-telnet 使用 POP3 获取电子邮件

简单来说，我们需要做的事情就是使用 socket 套接字建立连接，操作 socket 的输入流输和输出流，以达到 Telnet 的效果。

一个不恰当的比喻：整个邮件发送-传输-接收过程，用户代理(即我们要实现的

功能)向对应的STMP服务器(比如stmp.qq.com)发送邮件就像把真实邮件送到附近邮局就可以默认是发送了，邮局把这信件发送的接收者所在的邮局，中间不会停留在某个邮局，pop协议取邮件即接受者的用户代理去邮局取出邮件。

电子邮件的信息格式

图 6-18　MIME 和 SMTP 的关系

MIME，全称为“Multipurpose Internet Mail Extensions”，比较确切的中文名称为“多用途互联网邮件扩展”。它是当前广泛应用的一种电子邮件技术规范，基本内容定义于RFC 2045-2049。自然，MIME邮件就是符合MIME规范的电子邮件，或者说根据MIME规范编码而成的电子邮件。在MIME出台之前，使用RFC 822只能发送基本的ASCII码文本信息，邮件内容如果要包括二进制文件、声音和动画等，实现起来非常困难。MIME提供了一种可以在邮件中附加多种不同编码文件的方法，弥补了原来的信息格式的不足。

邮件头包含了发件人、收件人、主题、时间、MIME版本、内容的类型、内容的传输编码方式等重要信息。每条信息称为一个域，由域名后加冒号（“ : ”）和信息内容构成，可以是一行，也可以占用多行。域的首行必须顶头写(即左边不能有空白字符（空格和制表符））；续行则必须以空白字符打头，且第一个空白字符不是信息本身固有的，解码时要过滤掉。另外，邮件头中不允许出现空行。

域名	含义	添加者
Received	传输路径	各级邮件服务器
Return-Path	回复地址	目标邮件服务器
Delivered-To	发送地址	目标邮件服务器
Reply-To	回复地址	邮件的创建者
From	发件人地址	邮件的创建者
To	收件人地址	邮件的创建者
Cc	抄送地址	邮件的创建者
Bcc	暗送地址	邮件的创建者
Date	日期和时间	邮件的创建者
Subject	主题	邮件的创建者
Message-ID	消息ID	邮件的创建者
MIME-Version	MIME版本	邮件的创建者
Content-Type	内容的类型	邮件的创建者
Content-Transfer-Encoding	内容的传输编码方式	邮件的创建者

图-MIME电子邮件头部

域名	含义
Content-Type	段体的类型
Content-Transfer-Encoding	段体的传输编码方式
Content-Disposition	段体的安排方式
Content-ID	段体的ID
Content-Location	段体的位置(路径)
Content-Base	段体的基位置

图-MIME电子邮件段头

图-MIME 电子邮件主体格式

```
From: xiexiren@tsinghua.org.cn
To: xyz@163.com
MIME-Version: 1.0
Content-Type: multipart/mixed; boundary=qwertyuiop

--qwertyuiop
XYZ:
     你要的图片在此邮件中，收到后请回信。
                                                谢希仁
--qwertyuiop
Content-Type: image/gif
Content-Transfer-Encoding: base64
          ...data for the image (图像的数据)...
--qwertyuiop--
```

图-MIME 电子邮件样例

如果在邮件中要添加附件，必须定义 multipart/mixed 段；如果存在内嵌资源，至少要定义 multipart/related 段；如果纯文本与超文本共存，至少要定义 multipart/alternative 段。其实很容易察觉,MIME 邮件就像我们实体信件一样，信封注明了一些信息。

设计与实现

总体框架

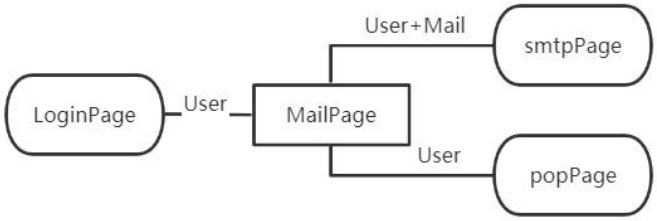
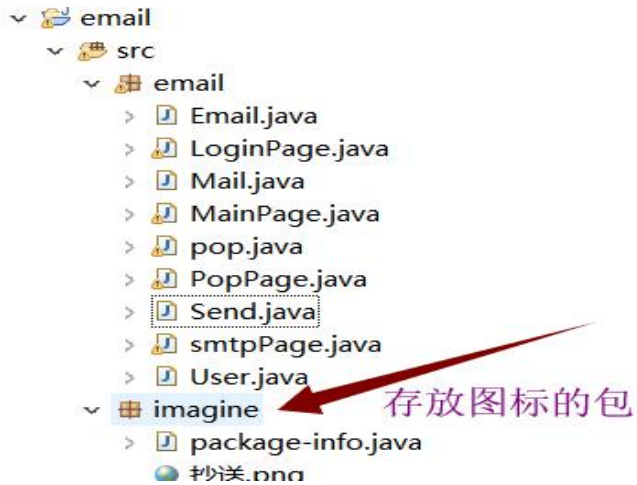


图-框架图

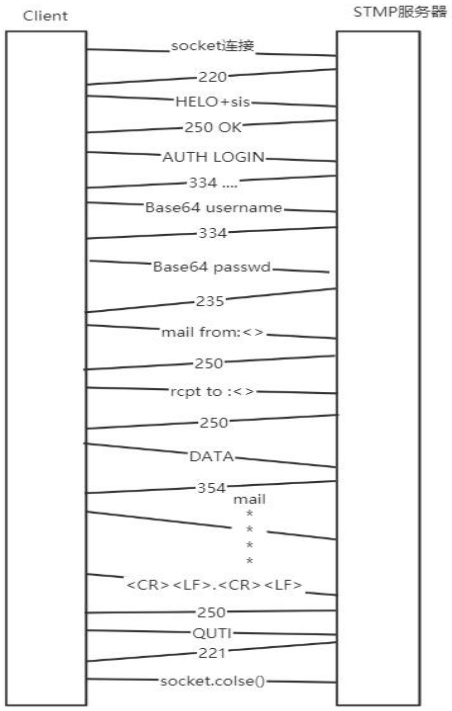


类图见附录

Java socket 编程使用

关于 java 的 socket 编程，可以看到大致流程是这样的，在建立 socket 连接后，问题转变为如何使用 smtp 和 pop 协议对流进行操作。在连接过程中通过输入输出流读取和发送指令。

```
        Socket socketSend = new Socket("SMTP." + sendUser.getServerAddr(), 25);
    //Output: message to server  发送给server的信息
    OutputStream outToServer = socketSend.getOutputStream();
    PrintWriter toServer = new PrintWriter(outToServer,true);
    //Input: message from server 接受来自server的信息
    InputStream inFromServer = socketSend.getInputStream();
    BufferedReader fromServer = new BufferedReader(new InputStreamReader(inFromServer));
```



登陆页 LoginPage

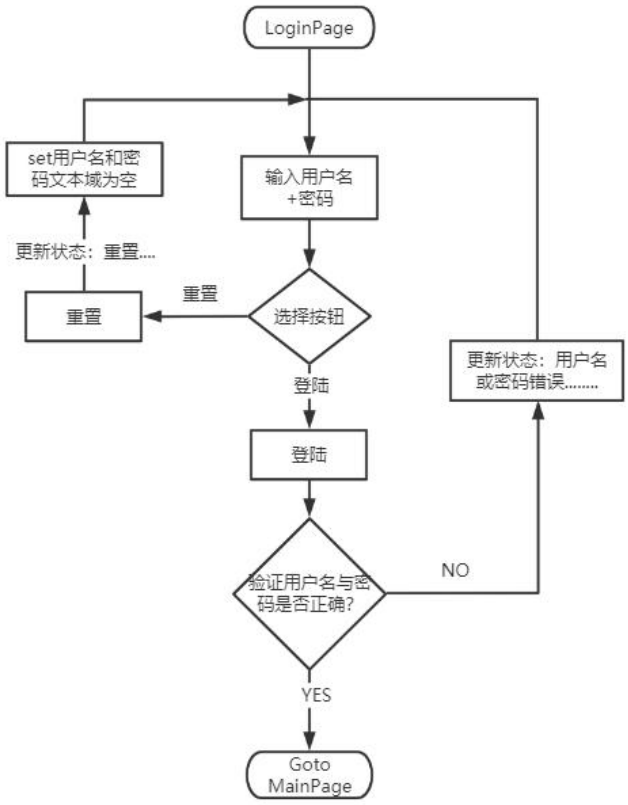


图-loginPage 登陆流程图

将登陆功能封装到 User 类中，使用 java 的 socket 连接 SMTP 服务器，并且捕获 socket 的未知主机异常(UnknownHostException)、连接异常 ConnectException、连接超时异常(SocketTimeoutException)，并且得到 socket 的输入流和输出流。

```
try {
    Socket socketLogin = new Socket("SMTP." + serverAddr,25);
    //捕获异常--socketLogin
    OutputStream outToServer = socketLogin.getOutputStream();
    PrintWriter toServer = new PrintWriter(outToServer,true);
    InputStream inFromServer = socketLogin.getInputStream();
    BufferedReader fromServer = new BufferedReader(new InputStreamReader(inFromServer));
```

根据 HELO 的返回是不是 250(不是说没服务器忙于其他)和 AUTH LOGIN 命令的返回码是不是(235 说明用户名密码正确)确定登陆是否成功，然后释放连接。

```
toServer.println("HELO CMF");
ReMessage = fromServer.readLine().toString();
status = Integer.parseInt(ReMessage.substring(0,3));
if(status!=250) {
    socketLogin.close();
    return false;
}
if(serverAddr.indexOf("qq")!=-1||serverAddr.indexOf("QQ")!=-1)
{fromServer.readLine();  由于smtp.qq.com对HELO命令返回的是三行信息，而其他smtp服务
  fromServer.readLine();器返回的都是一行指令(250 OK...)，所以需要将输出流读取到最
}                        新位置，否则就会产生流的阻塞。
toServer.println("AUTH LOGIN");
fromServer.readLine();
toServer.println(userBase64);   向socket输入流写入用户名和密码
fromServer.readLine();
toServer.println(passwordBase64);
ReMessage = fromServer.readLine().toString();
status = Integer.parseInt(ReMessage.substring(0,3));
if(status!=235) {
    socketLogin.close();
    return false;
}
```

登陆按钮的点击触发事件。

```
//  登录按钮-监听事件
loginButton.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        authLog  = new User();
        String Message = new String("");
        if(authLog.setUser(userTextField.getText().toString(),String.valueOf(pas
            {
            if(authLog.login()) {   根据User类中封装的login()功能，
                setSignLogin(true);   返回true or false确定登陆是否成功。
                Message  = "登录成功......";
            }
            else  Message  = "密码或用户名错误,请重试......";
            }
        else Message  = "用户名错误,请检查重试.......";
        printStatusLabel.setText(Message);   显示状态Label
        if(signLogin) {
            new MainPage(authLog).doMainPage(authLog);  Goto MainPage
            dispose();
        }
    }
});
```

图- 优雅的 loginPage 登陆界面图

主页面 mainPage

此模块绘制页面，主要做一些按钮的监听事件和负责显示文字。

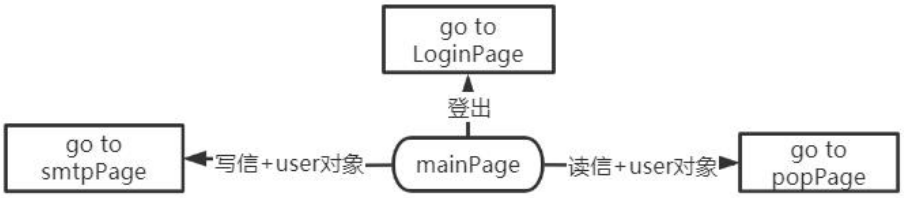


图-mainPage 页面

写信页 SmtpPage

这部分将实现 SMTP 发送邮件功能，数据流可以看主体框架得到。

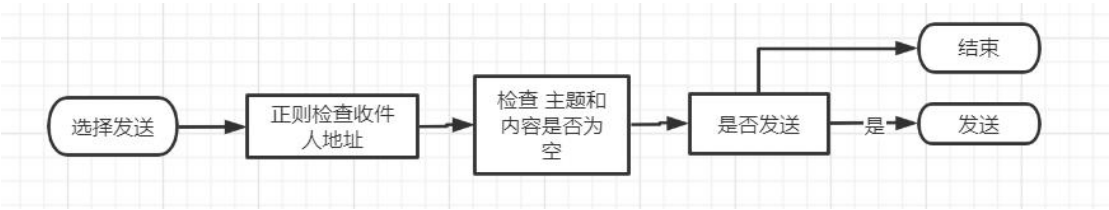


图- smtpPage 一般流程图

通过将发送邮件封装成 send 类，通过图形部分得到收件人、主题、内容、附件；在 send 类中实现邮件的发送。监听发送按钮，触发点击时，捕获各个文本域信息传递给 send 类发送。

```
sendButton.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        if(isRight(revtextField.getText())) {
            String subject = subjectTextField.getText();   // 抄送 和 主题
            String body = contentArea.getText(),Message = new String("<html><h1><font color='black'>");

            boolean signal = false;
            int Option = 0;
            if(subjectTextField.getText().length()==0) {
                Message += "主题为空, ";
                signal = true;
            }
            if(contentArea.getText().length()==0) {
                Message += "邮件内容为空, ";
                signal = true;
            }
            if(signal)  Option = JOptionPane.showConfirmDialog(null,Message+"确定不填写吗？？</font></h1></h
          if(Option==0) {  // 选择空的内容为Yes
              String showMessage = new  String();
              Mail.setMailContent(revtextField.getText(),subject, body);
              showMessage = returnConnet(new Send(auth).connect(getMail()));
              JOptionPane.showMessageDialog(null,showMessage,"消息",1);
          }
        } else
        {  // 用户名不含@
            JOptionPane.showMessageDialog(null,"用户名不正确，请检查重试.","错误",0);
        }
```

首先，由于已经使用账号登陆成功，所以接下来发送邮件不必去做检验账号和密码处理，但是 socket 的异仍需要捕获。简单的流程框架就是使用 telnet 发送邮件的流程。不过需要处理的是：对各个部分的编码、以及带附件以后采用 mixed 格式，那么将包含几个 multipart，分别由于--boundary 分割开，在每一个 multipart 中都将在此声明一个子 part-boundary 用于在此 multipart 中分割。另外需要声明此部分内容的文本格式（假如是文本的话），以及编码方式。然后，在下一个 multipart 就可以类似的加入附件，可参考以下流程图。

对于正常不带附件的邮件，那么只需要在一个 part 中描述编码的内容即可，当然也可以多个 multipart 的文本，不过对于实现来说容易，但是在 pop 解析部分变得异常艰难。

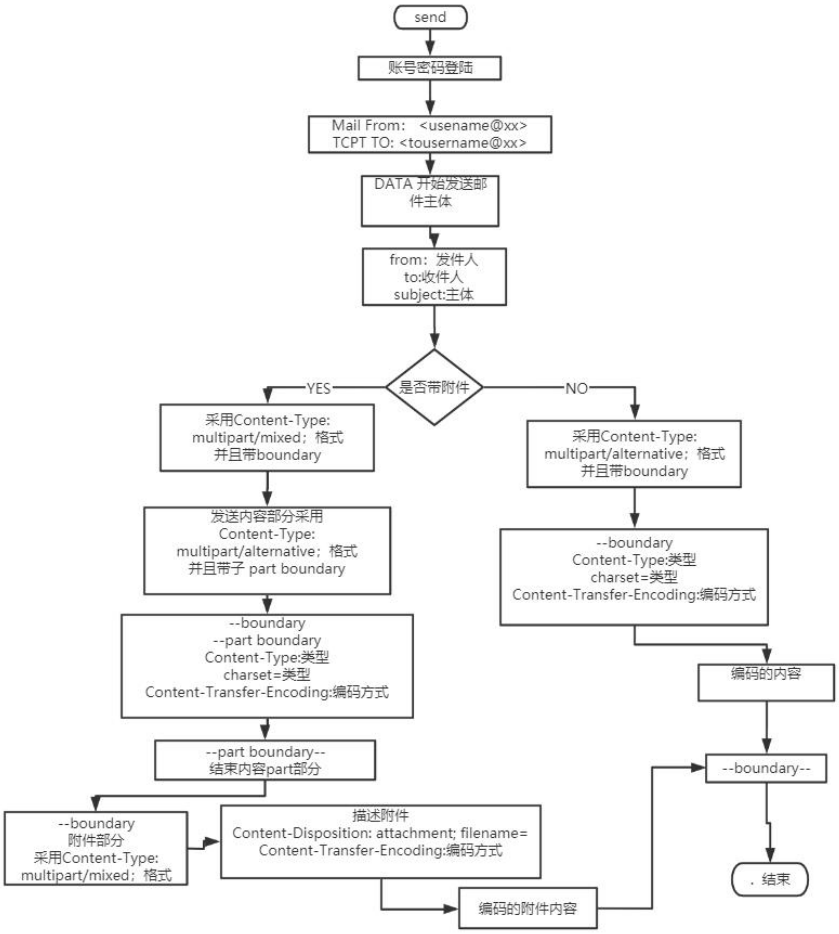


图-send 类发送的流程图

```
//邮件头部
toServer.println("From: "+ "<"+sendUser.getUserName()+">");
toServer.println("To:" + "<"+Mail.getAddressee()+">");
toServer.println("Subject: =?UTF-8?B?"+ Base64.getEncoder().encodeToString(Mail.getSubjec
if(Mail.getAnnex()) {
    toServer.println("Content-Type: multipart/mixed;boundary=\"minfu\"");
    toServer.println("MIME-Version: 1.0");
    toServer.println("\n--minfu");
    toServer.println("Content-Type: multipart/alternative; boundary=\"part_minfu\"");
    toServer.println("\n--part_minfu");
    toServer.println("Content-Type: text/plain; charset=\"UTF-8\"");
    toServer.println("Content-Transfer-Encoding: base64\n");
}else
{
    toServer.println("Content-Type: multipart/alternative; boundary=\"minfu\"");
    toServer.println("MIME-Version: 1.0");
    toServer.println("\n--minfu");
    toServer.println("Content-Type: text/plain; charset=\"UTF-8\"");
    toServer.println("Content-Transfer-Encoding: base64");
    toServer.println();
}
//正文
```

对应上述流程图的附件或正常文本格式的头部。

图- 优雅的 smtpPage 发送界面图

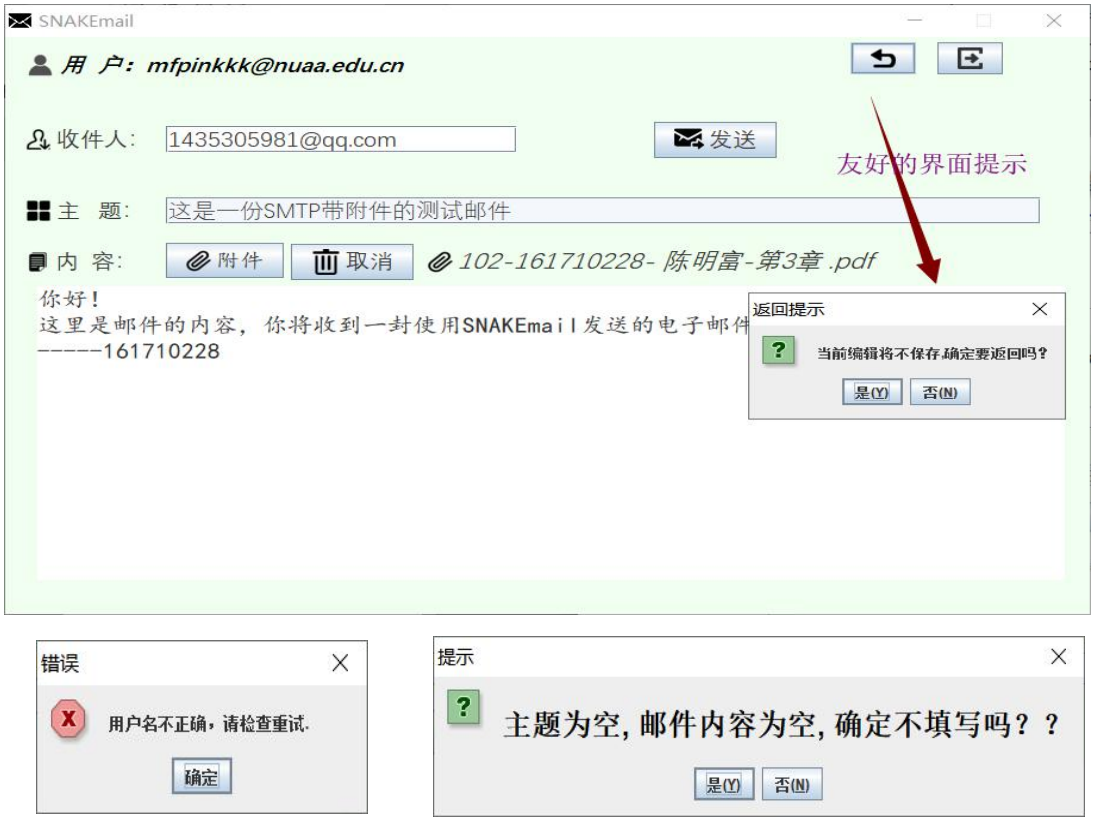
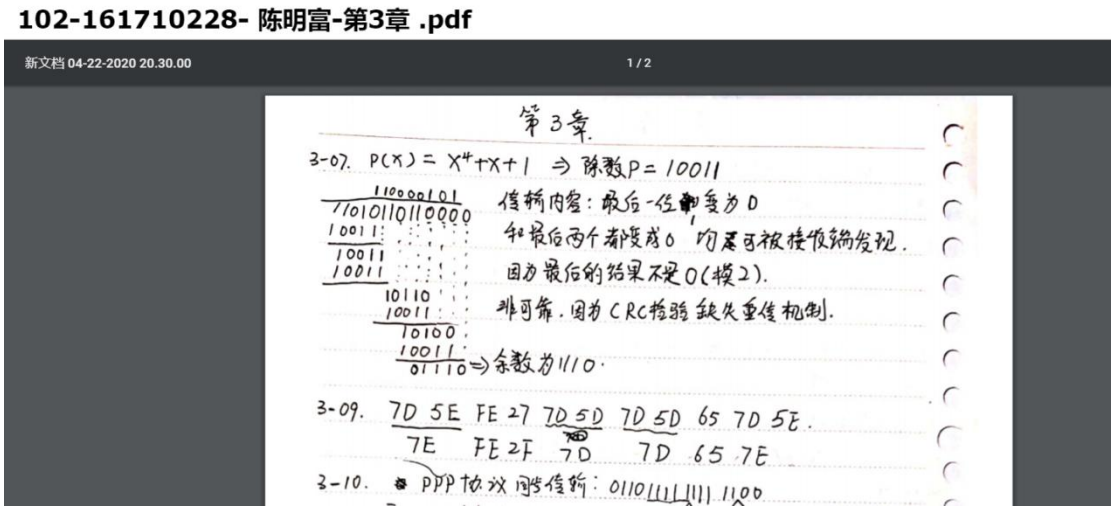


图--友好的提示

登陆 QQ 邮箱检查上面发送的邮件，并打开附件 PDF 可以看到正常，说明编码和解码没有问题。



读信页 popPage

此模块堪称为最难部分亦是最复杂部分，不亚于写一个编译词法语法分析。

首先由上述实验内容部分可以得到MIME邮件的基本格式，这其中首先要对smtp的命令关键字进行识别，然后在string中截取出文字格式、编码方式、boundary、filename、multipart-type等内容，由于邮件都是以--mainboundary--和<CR><LF>.<CR><LF>结束，可作为一切循环的终止条件，以防进入死循环。

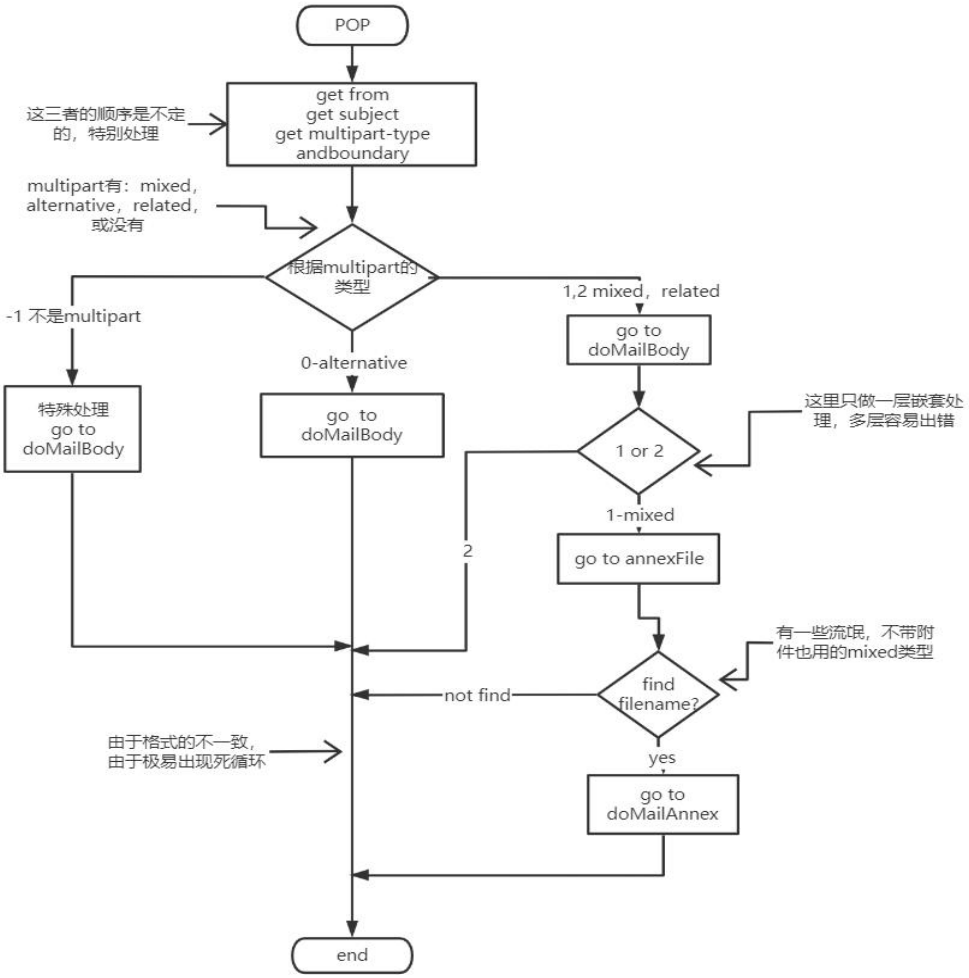


图-pop 读取解析邮件

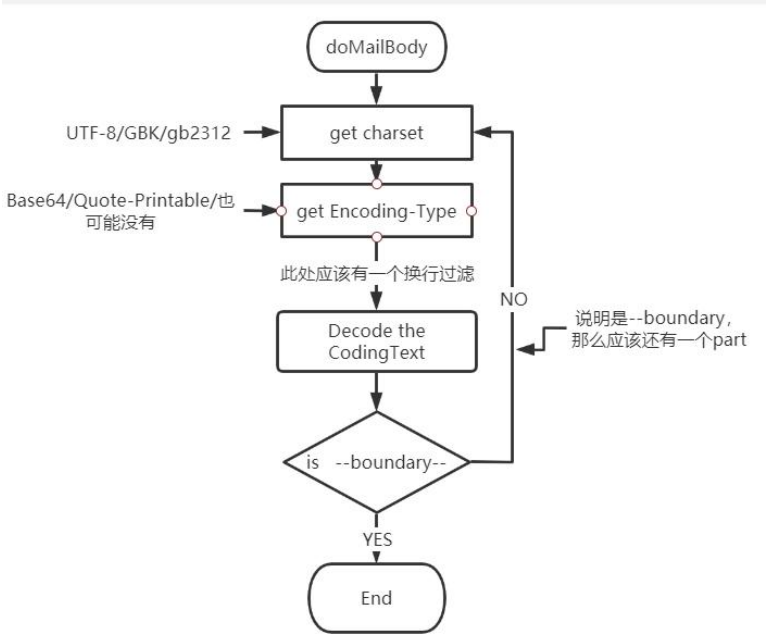


图-doMailBody 流程图

```
public boolean doMialBody(BufferedReader fromServer,String reply,String boundary) throws Exception{
    String charset = new String("");
    int encodingType = 0;
     try {
         // get charset
         //reply = fromServer.readLine();
         while(!reply.equals("--"+boundary+"--")&&!isEndLine) {
             charset = getCharset(fromServer,reply);
             if(isEndLine)  break;
             // get encoding-type
             reply = fromServer.readLine();
             encodingType = getEncodingType(fromServer,reply);
             if(isEndLine)  break;

             body +=  doContent(fromServer, boundary, charset, encodingType);
             reply = fromServer.readLine();
         }
        if(reply.indexOf("--"+boundary+"--")!=-1||isEndLine) return true;
}catch(Exception e) {
}
     return  false;
}
```

由于各种用户代理之间的差异，无可避免的存在了多种情况，如编码和未编码，编码中又有 UTF-8 和 gb2312、gbk 等，有一些是没有双引号”修饰的；未编码可能就相对简单。而其他部分也或多或少存在相似的处理。

截取 from 的 type 和 codetype 代码。

```
String fromType = reply.substring(reply.indexOf("=?")+2);
char codeType  = fromType.charAt(fromType.indexOf("?")+1);
fromType = fromType.substring(0,fromType.indexOf("?"));
reply = reply.substring(reply.indexOf(codeType+"?")+2);
String coding = reply.substring(0,reply.indexOf("?="));
```

```
public String doMialSubject(String reply)throws Exception {
    String subject = new String("");
    try {
        if(reply.indexOf("=?")!=-1) {
            /* Subject: =?subjectType?coding?coding=?=
             * subjectType: UTF-8 GBK gb2312
             * codeType:  B/Q  Base64
            */
            String subjectType = reply.substring(reply.indexOf("=?")+2);
            char subjectCoding  = subjectType.charAt(subjectType.indexOf("?")+1);
            subject = subjectType;
            subjectType = subjectType.substring(0,subjectType.indexOf("?"));
            subject = subject.substring(subject.indexOf(subjectCoding+"?")+2);
            subject = subject.substring(0,subject.indexOf("?="));
            if(subjectCoding=='b'||subjectCoding=='B') // BASE64编码
                subject = new String(Base64.getMimeDecoder().decode(subject), subjectType);
            else {
                ByteArrayOutputStream buffer = new ByteArrayOutputStream();
                qpDecoding(buffer,reply);
                subject = new String(buffer.toByteArray(), subjectType);
            }
        }
        else {//Subject: 未编码纯文本格式主题
            subject = new String(reply.substring(9).getBytes(),"UTF-8");
        }
    }catch(Exception e) {
    }
    return subject;
```

实验结果：

图-popPage 实验结果

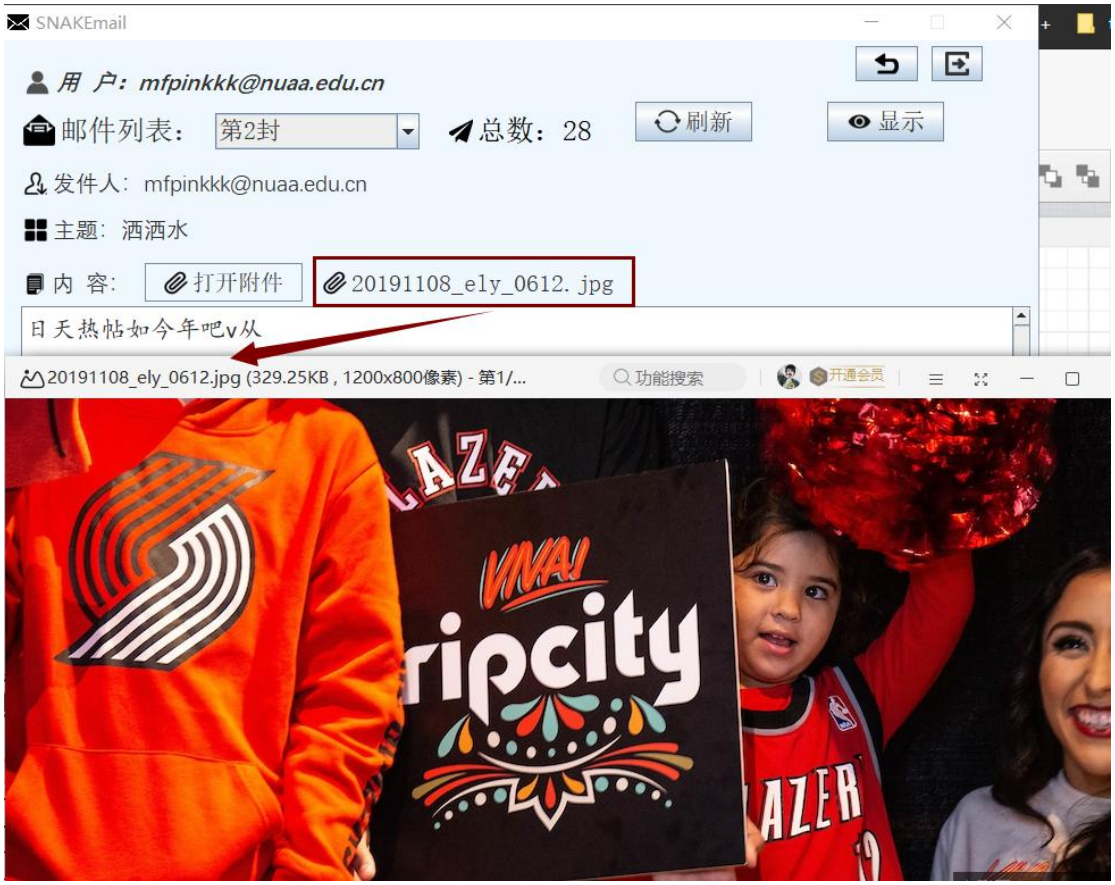


图-popPage 实验结果检验-1

邮件是可以自己给自己发送的，默认附件是自动下载的，如上打开附件没有问题。尝试着解析其他人发来的邮件，此部分堪称最难，由于各种不同的用户代理采用都是 MIME，但是很多细节格式并不一致，比如 from、subject、boundary 的位置顺序并不是固定的。实话说，解析邮件不亚于一个编译器。

以下解析其他人发来的邮件结果。

图-popPage 实验结果检验-2

图-popPage 实验结果检验-3

一封邮件的原文大致如下，不带附件的。

```
/*
删除了一些无关紧要的信息
*/
Date: Tue, 14 Apr 2020 12:49:02 +0800 (CST)
From: Fang Fang <irecruit@italent.cn>
Reply-To: fang_fang <fang_fang@trendmicro.com>
To: mfpinkkk <mfpinkkk@nuaa.edu.cn>
Message-ID: <1586839742612_19243_23175_1543.sc-10_9_63_161-inbound0@shmail.ibeisen.com>
Subject: =?utf-8?b?6LaL5Yq/56eR5oqA5qCh5Zut5oub6IGY56yU6K+V6YCa55+l?=
MIME-Version: 1.0
Content-Type: multipart/alternative; boundary="----=_Part_1024540_543389348.1586839742594"
/*
删除了一些无关紧要的信息
*/
------=_Part_1024540_543389348.1586839742594
Content-Type: text/html; charset="UTF-8"
Content-Transfer-Encoding: base64

PHA+6ZmI5piO5a+M77yMPC9wPgo8cD4mbmJzcDsgJm5ic3A7ICZuYnNwO+aCqOWlve+8jOaBreWW
nOaCqOW3sue7j+i/m+WFpei2i+WKv+enkeaKgOagoeaLm+iBjOS9jeeahOeslOivleeOr+iKgu+8
jCDor6XogYzkvY3nmoTnrJTor5Xml7bpl7TmmoLlrprku6Dbml6UxOTowMCAt
IDIwOjMw5oiWMjAyMOW5tDXmnIgxMOaXpTE5OjAwIC0gMjA6MzDjgII8L3A+CjxwPiZuYnNwOyAm
bmJzcDsgJm5ic3A75oiR5Lus5bCG6YCa6L+H55+t5L+hLeUteivneS4juaCqOi/m+S4gOatpeeh
dDsiPuS6uuWKm+i1hOa6kOmDqDwvcD4=

------=_Part_1024540_543389348.1586839742594--
```

一封带附件的邮件原文，可以观察 boundary 和 partboundary，对照着上述流程图。

```
/*删除了一些无关紧要的信息*/
From: <mfpinkkk@nuaa.edu.cn>
To:<mfpinkkk@nuaa.edu.cn>
Subject: =?UTF-8?B?5rSS5rSS5rC0?=
Content-Type: multipart/mixed;boundary="minfu"
MIME-Version: 1.0
/*删除了一些无关紧要的信息*/
--minfu
Content-Type: multipart/alternative; boundary="part_minfu"

--part_minfu
Content-Type: text/plain; charset="UTF-8"
Content-Transfer-Encoding: base64

5pel5aSp54Ot5biW5aaC5LuK5bm05ZCnduS7jg==

--part_minfu--

--minfu
Content-Type:multipart/mixed
Content-Disposition: attachment; filename="=?UTF-8?B?MjAxOTExMDhfZWx5XzA2MTIuanBn?="
Content-Transfer-Encoding: base64

6E2+GSc0OBB0W0qT2Ochv6N2bf4fYbLzdWx/epX7h2dlUGA+2VIthzjB/wDMVUTceVz+Wg9E4LGY
pI3jUH/YX8pOpi1krTJTuGKN/v4eh6kJ5ZUuEtDU4pWnFwXHL5FHDDKyWP8AQv0vqOrT6LaUWbJR
/*为了显示，删除了一部分附件的编码，实际上有成千上万行编码*/

--minfu--
```

附件解码代码。

```
if(isBase64(annaxEncoding))
{ // BASE64编码
    while (true) {
      if(reply.indexOf("--"+Mainboundary)!=-1) break;
      annexfileInput.write(Base64.getMimeDecoder().decode(reply));

      reply = fromServer.readLine();
  }
}
 else { // q编码
    ByteArrayOutputStream buffer = new ByteArrayOutputStream();
    while (true) {
         if(reply.indexOf("--"+Mainboundary)!=-1) break;
         qpDecoding(buffer,reply);
         reply = fromServer.readLine();
     }
    annexfileInput.write(buffer.toByteArray());
 }
```

心得体会

首先，我觉得应当先说明遇到的那些问题：

在实现 telent 发送邮件时，从网络上搜索参考的大部分博客在 mail from 和 rcpt to 指令上皆没有<>,导致停滞了很久。

在没有知道 Windowsbuilder 之前，我企图去画图形界面，简直就是天方夜谭；好在参考报告模板里提到了 Windowsbuilder，于是让图形化变得简单，从而专注于功能的实现。

由于之前没有 java 图形编程经验，基本一切从 0 开始，学习 java swing，容器面板、文本域、按钮、label 等的使用，如何设置图标，如何事件监听触发。

在 socket 连接中，好在 java 有各种异常的捕获，也因此减少了很多问题，还有其他的数据越界之类的异常，真的省了不少精力。

在实现邮件的解析部分真的令人崩溃，代码必须适配各种不一样的邮件格式，即使他们采用的是同一个标准，一不小心就死循环。而那些嵌套 mixed 的就更可怕了，boundary 的子 part boundary.....比起来，解析邮件就简单多。

编码部分亦是一大头痛问题，因为有些邮件原文采用 UTF-8/GBK/gb2312，或 Base64 或 qp 或明文，需要把各个词语截取下来，然而有一些会多个空格，或是没有引号”.....不禁感叹制定标准和建立标准体系多么重要，秦始皇车同轨书同文是多么伟大。

实话说，解析原件部分简直和编译的词法语法分析似曾相识，不可能兼容所有的邮件，所以有些 bug 总是要存在的。当然，经过本次工程实验的，不管是对 SMTP/POP 协议、电子邮件格式，还是 java、流程图和 eclipse 的使用上都大有收获。因为之前并没有实际上手过 java，导致一开始犹豫 C++还是 java，最后是馋了 java 的 swing 图形才敢于上手，至于最终结果也是满意的。

在没有清晰的框架概念之前，最后不要下手写代码，不然会一改再改，面目全非。现代人的世界离不开网络，我们并不是孤独在一条路上行走，固然有众多前辈留下了宝贵的经验，值得去参考和借鉴。为什么不呢？闭门造车的时代早已逝去。即使是重复造轮子，那也要尽量造的圆一些。

最后，疫情期间的网课着实不易，感谢燕老师！

参考与附录

[1].Socket 收发邮件--SMTP 和 POP3

https://blog.csdn.net/iteye_18655/article/details/81400129

[2].SMTP 邮件传输协议发送邮件和附件

https://www.cnblogs.com/sdgwc/p/3324368.html

[3].Socket 计算机网络--电子邮件

https://blog.csdn.net/NEUChords/article/details/92090423

[4].SMTP 错误码/建议解决方法

https://blog.csdn.net/chenfei_5201213/article/details/10138969

[5].SMTP errors and reply codes
https://serversmtp.com/smtp-error/
[6].电子邮件基本格式（MIME 格式）
https://blog.csdn.net/weixin_37958284/article/details/77186279
[7].MIME 格式介绍
https://www.cnblogs.com/robinhood/articles/540464.html
[8].MIME 参考手册
https://www.w3school.com.cn/media/media_mimeref.asp

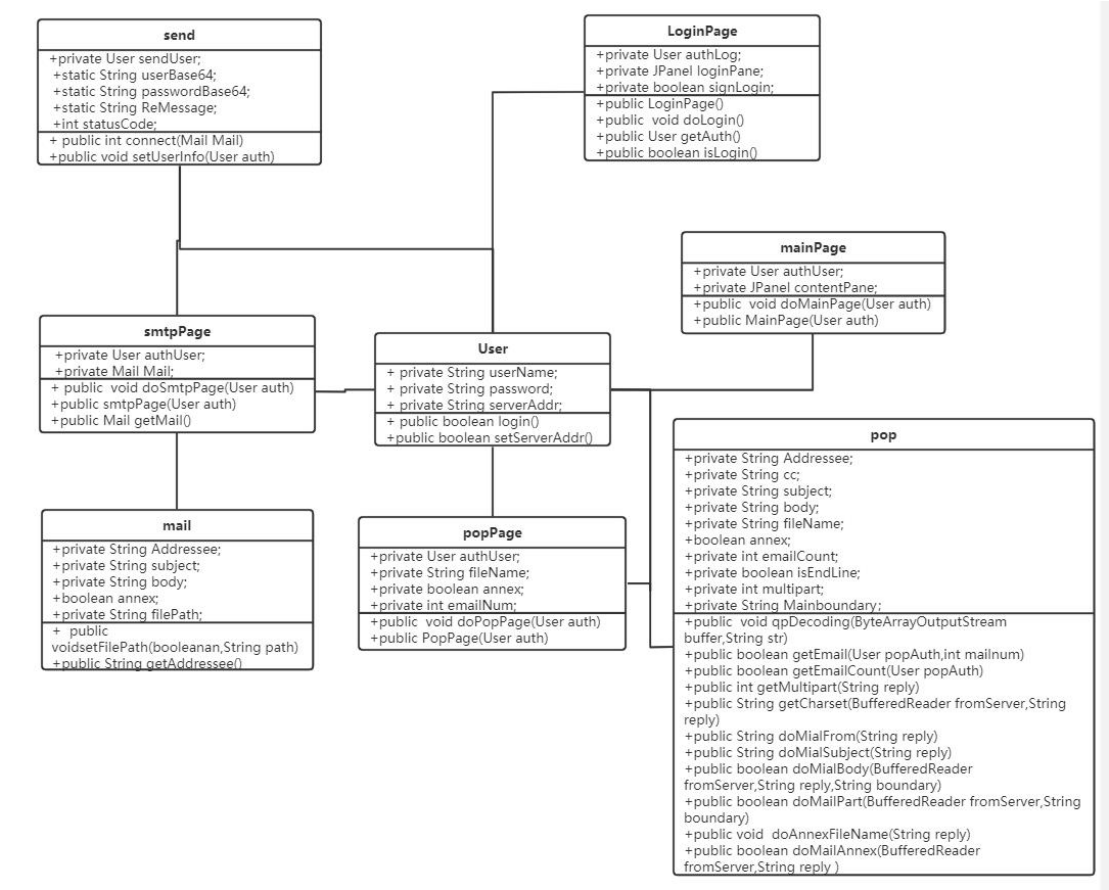


图-类图

自画类图，可能描述不一定准确。