

武汉大学计算机学院

本科生实验报告

网络系统软件与应用软件开发

专 业 名 称 : 卓越工程师班

课 程 名 称 : 计算机网络应用设计

团 队 名 称 : 白色小分队

指 导 教 师 一: 周浩 讲师

指 导 教 师 二:

团 队 成 员 一: 王一舟 (2015301500153)

团 队 成 员 二: 罗 格 (2015301500123)

团 队 成 员 三: 李文洲 (2015301500162)

团 队 成 员 四: 王致远 (2015301500132)

二〇一八年四月

郑 重 声 明

本团队呈交的实验报告，是在指导老师的指导下，独立进行实验工作所取得的成果，所有数据、图片资料真实可靠。尽我所知，除文中已经注明引用的内容外，本实验报告不包含他人享有著作权的内容。对本实验报告做出贡献的其他个人和集体，均已在文中以明确的方式标明。本实验报告的知识产权归属于培养单位。

团队成员签名： 王一丹 罗格 李文洲 王致远

日期： 2018.7.9

摘 要

实验的实验目的是完成一个邮件系统（包括 SMTP 和 POP3 客户端）。

实验设计主要遵循网络系统开发的基本方法、开发平台的使用方法、继承数据库的调用方法。

实验内容主要包括：邮件系统（包括 SMTP 和 POP3 客户端）的基础功能，接受、编写并发送、阅读、删除等几个功能。

关键词：邮件系统；SMTP；POP3

目录

1 实验目的和意义	5
1.1 实验目的	5
1.2 实验意义	5
2 实验设计	6
2.1 概述	6
2.2 实验原理	6
2.3 实验方案	10
3 结论	13
3.1 程序主要界面及结果	13
3.2 程序源程序	17
参考文献	30

1 实验目的和意义

1.1 实验目的

本实验是使学生熟悉网络规划与设计的基本知识和方法、掌握网络系统软件与应用软件开发的方法，能将所学的操作系统、数据库、软件工程、计算机网络等方面的知识集成到一起，规划、安装、调试实际网络系统、开发实际软件系统。

1.1.1 目的一：面向系统的软件开发

本实验是使学生掌握网络系统软件的开发方法、开发平台的使用、与实际数据库的集成方法。用 JAVA/ VC++/C# 完成 FTP 客户端、SMTP 客户端、POP3 客户端三个系统程序。

1.1.2 目的二：面向网络应用的软件开发

利用软件工程的方法，设计一个小规模的应用系统，与具体数据库连接起来，从用户界面设计、数据库设计到处理流程设计，最终完成系统的编程。本实验将用网上书店作为例子，要求实现基于 Web 的远程功能（Web 页面）和后端管理功能。

1.2 实验意义

该实验是理论知识和动手能力的综合体现。通过本实验，掌握网络系统软件、网络应用软件的开发方法、开发平台的使用、与实际数据库的集成方法。

2 实验设计

2.1 概述

实验运用网络编程的原理，采用 C# 进行图形界面和后台服务的编写，实现了一个具有基本功能的邮件系统（包括 SMTP 和 POP3 客户端）

2.2 实验原理

2.2.1 SMTP

简单邮件传输协议（SMTP）的目标是可靠高效地传送邮件，它独立于传送子系统而且仅要求一条可以保证传送数据单元顺序的通道。

SMTP 设计基于以下通信模型：针对用户的邮件请求，发送 SMTP 建立与接收 SMTP 之间建立一个双向传送通道。接收 SMTP 可以是最终接收者也可以是中间传送者。SMTP 命令由发送 SMTP 发出，由接收 SMTP 接收，而应答则反方面传送。

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），双方将如此重复多次。当接收者收到全部邮件后会接收到特别的序列，如果接收者成功处理了邮件，则返回 OK 应答。

在 SMTP 发送操作中有三步，操作由 MAIL 命令开始给出发送者标识。一系列或更多的 RCPT 命令紧跟其后，给出了接收者信息，然后是 DATA 命令列出发送的邮件内容，最后邮件内容指示符确认操作。

过程中的第一步是 MAIL 命令，< reverse-path >包括源邮箱。

MAIL FROM:

此命令告诉接收者新的发送操作已经开始，请复位所有状态表和缓冲区。它给出反向路径以进行错误信息返回。如果请求被接收，接收方返回一个 250 OK 应答。中不止包括了邮箱，它包括了主机和源邮箱的反向路由，其中的第一个主机就是发送此命令的主机。

过程中的第二步是发送 RCPT 命令。

RCPT TO:

此命令给出向前路径标识接收者，如果命令被接收，接收方返回一个 250 OK 应答，并存储向前路径。如果接收者未知，接收方会返回一个 550 Failure 应答。此过程可能会重复若干次。

不仅包括邮件，它是主机和目的邮箱的路由表，在其中的第一个主机就是接收命令的主机。过程中的第三步是发送 DATA 命令。

DATA

如果命令被接收，接收方返回一个 354 Intermediate 应答，并认定以下的各行都是信件内容。当信件结尾收到并存储后，接收者发送一个 250 OK 应答。因为邮件是在传送通道上发送，因此必须指明邮件内容结尾，以便应答对话可以重新开始。SMTP 通过在最后一行仅发送一个句号来表示邮件内容的结束，在接收方，一个对用户透明的过程将此符号过滤掉，以不影响正常的的数据。

2.2.2 ESMTP

ESMTP (Extended SMTP)，是对标准 SMTP 协议进行的扩展。ESMTP 最显著的地方是添加了用户认证功能。如果用户想使用 ESMTP 提供的新命令，则在初次与服务器交互时，发送的命令应该是 EHLO 而不是 HELO。EHLO 对于具体服务器，响应会不同，关键字“8BITMIME”用来说明服务器是否支持正文中传送 8 位 ASCII 码，而以“X”开头的关键字都是指服务器自定义的扩充（还没纳入 RFC 标准）。

一个 ESMTP 通信过程的命令和响应如下：

```
C: //发起连接
S: 220 antispam1 ESMTP ready
C: EHLO smtp.tom.com
S: 250-antispam1
    250 AUTH PLAIN LOGIN
C: AUTH LOGIN
S: 334 VXN1cm5hbWU6
C: // BASE64 编码的用户名
```

```
S: 334 UGFzc3dvcnQ6
C: // BASE64 编码的密码
S: 235 2.0.0 OK
C: MAIL FROM:lyokoj@tom.com
S: 250 2.1.0 Ok
C: RCPT TO:lyokoj@tom.com
S: 250 2.1.5 Ok
C: DATA
S: 354 End data with <CR><LF>.<CR><LF>
C: // 邮件内容 以<CR><LF>.<CR><LF>结束
S: 250 2.0.0 Ok: queued as 725768125E
C: QUIT
S: 221 2.0.0 Bye
```

2.2.3 POP3

邮局协议-版本 3 使工作站可以用一种比较实用的方法来访问存储于服务器上的储存邮件。通常，这意味着工作站可以从服务器上取得邮件，而服务器为它暂时保存邮件。在下文中，客户主机指的是利用 POP3 服务的主机，而服务器主机指的是提供 POP3 服务的主机。

初始时，服务器通过侦听 TCP 端口 110 开始 POP3 服务。当客户主机需要使用服务时，它将与服务器主机建立 TCP 连接。当连接建立后，POP3 发送确认消息。客户和 POP3 服务器相互（分别）交换命令和响应，这一过程一直要持续到连接终止。

POP3 命令由一个命令和一些参数组成。所有命令以一个 CRLF 对结束。命令和参数由可打印的 ASCII 字符组成，它们之间由空格间隔。命令一般是三到四个字母，每个参数却可达 40 个字符长。

POP3 响应由一个状态码和一个可能跟有附加信息的命令组成。所有响应也是由 CRLF 对结束。现在有两种状态码，“确定”（“+OK”）和“失败”（“-ERR”）。

一时 TCP 连接由 POP3 客户打开，POP3 服务器发送一个单行的确认。这个消息可以是由 CRLF 结束的任何字符。例如，它可以是：

```
S: +OK POP3 server ready
```

此时 POP3 会话就进入了“确认”状态。此时，客户必须向服务器证明它的身份。用 USER 和 PASS 命令进行确认过程，客户必须首先发送 USER 命令，如果 POP3 服务器以“确认”状态码响应，客户就可以发送 PASS 命令以完成确认，或者发送 QUIT 命令终止 POP3 会话。如果 POP3 服务器返回 “失败” 状态码，客户可以再发送确认命令，或者发送 QUIT 命令。

一旦客户向服务器成功地确认了自己的身份，服务器将锁住并打开相应的邮件，这时 POP3 会话进入“操作”状态。可以使用 STAT、LIST、RETR、DELE、NOOP 和 REST 指令。最后，客户发送 QUIT 命令，会话进入“更新”状态。

一个 POP3 通信过程的命令和响应如下：

```
C: //发起连接
```

```
S: +OK POP3 ready
```

```
C: USER lyokoj@tom.com
```

```
S: +OK
```

```
C: PASS //密码
```

```
S: +OK Logged in.
```

```
C: LIST
```

```
S: +OK 11 messages:
```

```
1 19866
```

```
2 4141
```

```
3 4496
```

```
4 13561
```

```
5 7095
```

```
6 7735
```

```
7 3445
```

```
8 5166
```

```
9 2849
```

```
10 2824
11 2883
.
C: DELE 1
S: +OK Marked to be deleted.
C: RETR 11
S: +OK 2883 octets
    // 邮件内容
C: QUIT
S: +OK Logging out, messages deleted.
```

2.3 实验方案

2.3.1 整体模块设计

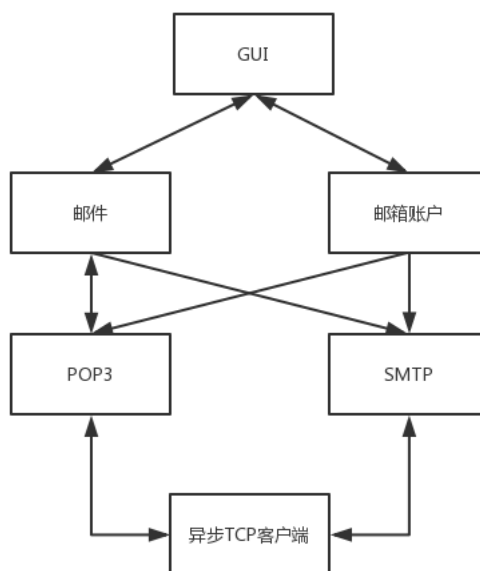


图 2.3.1 邮箱客户端模块

邮件客户端共有 6 个主要模块：

- GUI：图形界面，处理用户交互以及数据显示

- 邮件：存储邮件数据，包括发件人、收件人、主题、日期和内容等。
- 邮件账户：存储邮件账户数据，包括用户名、密码和服务器设置等。
- POP3：实现 POP3 协议，提供邮件收取和删除功能。
- SMTP：实现 SMTP 协议，提供邮件发送服务。
- 异步 TCP 客户端，实现非阻塞服务器客户端通信功能。

2.3.2 系统流程

邮件客户端主要由以下三种流程：

- 邮件收取
 - 用户由 GUI 发起收取邮件的申请。
 - POP3 模块根据用户设置的邮件账户提供的 POP3 服务器地址和端口打开一个向服务器连接的异步 TCP 客户端。
 - 根据用户名和密码，使用客户端向服务器发送登陆命令。
 - 登陆成功后，向服务器发送请求邮件个数的命令，POP3 模块解析出每封邮件的 ID 和大小。
 - 接着根据邮件 ID，分别向服务器请求每封邮件的内容
 - ◆ 邮件的头部和正文由邮件模块进行解析。
 - 所有邮件收取完毕后，向服务器发出退出命令。
 - GUI 界面更新。
- 邮件删除
 - 用户由 GUI 发起删除邮件的申请。
 - POP3 模块根据用户设置的邮件账户提供的 POP3 服务器地址和端口打开一个向服务器连接的异步 TCP 客户端。
 - 根据用户名和密码，使用客户端向服务器发送登陆命令。
 - 登陆成功后，根据 GUI 提供的邮件 ID，向服务器发送删除邮件命令。
 - 向服务器发出退出命令。
 - GUI 界面更新。

- 邮件发送

- 用户由 GUI 发起发送邮件的申请。
- 邮件模块根据 GUI 信息解析邮件头部和正文。
- SMTP 模块根据用户设置的邮件账户提供的 SMTP 服务器地址和端口打开一个向服务器连接的异步 TCP 客户端。
- 根据用户名和密码，使用客户端向服务器发送登陆命令。
- 登陆成功后，向服务器发送发件人地址。
- 服务器正确接收后，发送收件人地址。
- 服务器正确接收后，发送 DATA 以请求发送正文。
- 服务器响应正确后，发送邮件模块处理后的正文信息。
- 服务器响应正确后，向服务器发送退出命令。

上述流程中，若服务器没有正确响应，则中止流程，向服务器发送退出命令，并向用户提示错误信息。

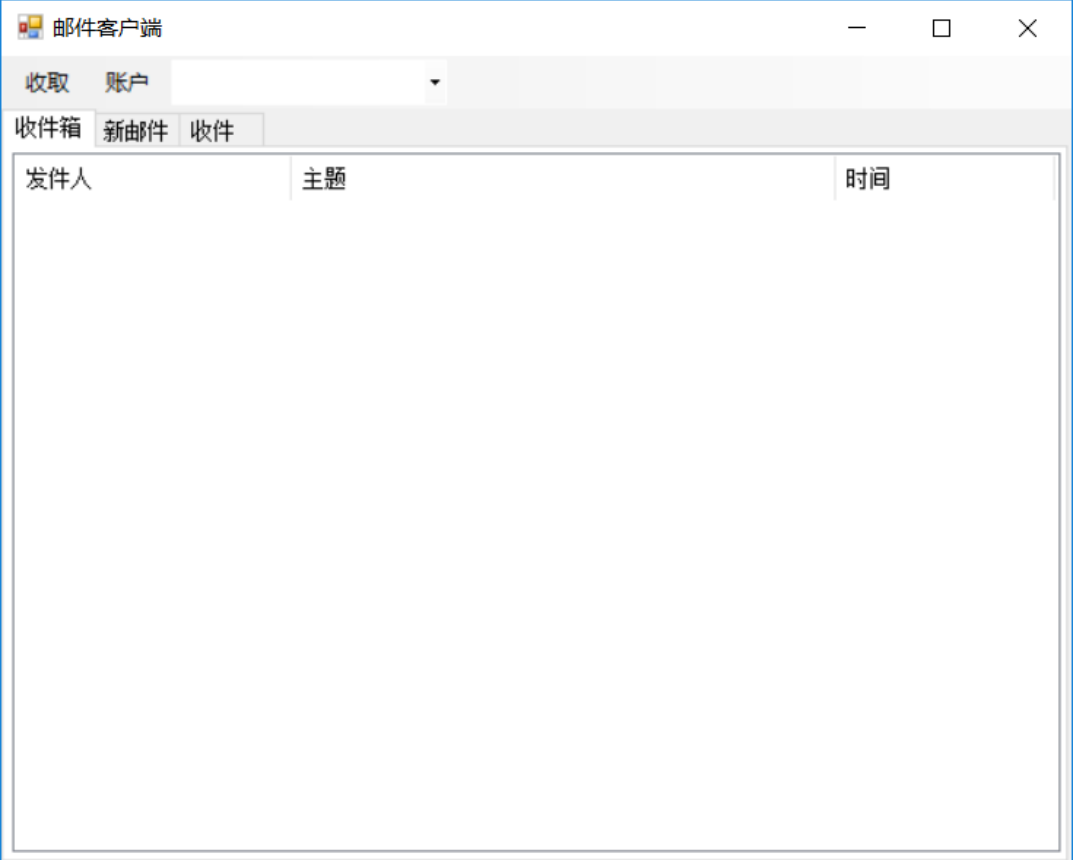
2.3.3 团队分工

- 罗格：实现邮件模块、邮件账户模块和异步 TCP 客户端。
- 王一舟：实现 SMTP 模块。
- 李文洲：实现 POP3 模块。
- 王致远：GUI 设计和编写。

3 结论

3.1 程序主要界面及结果

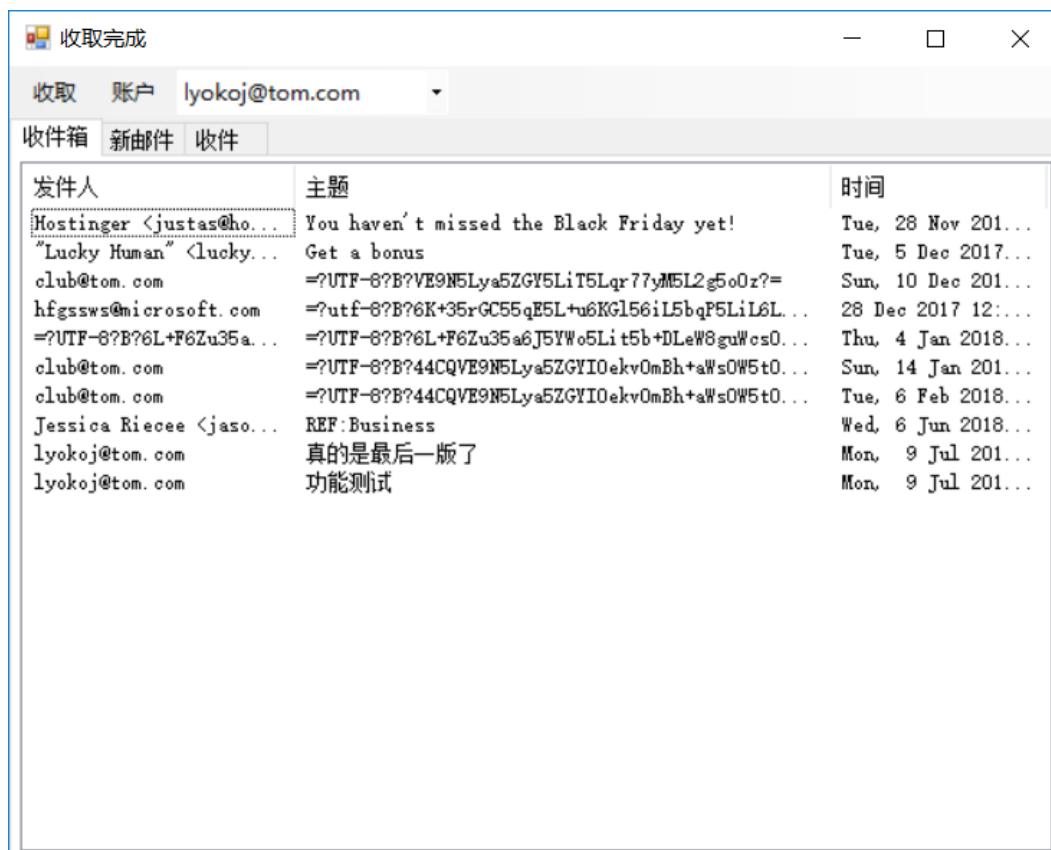
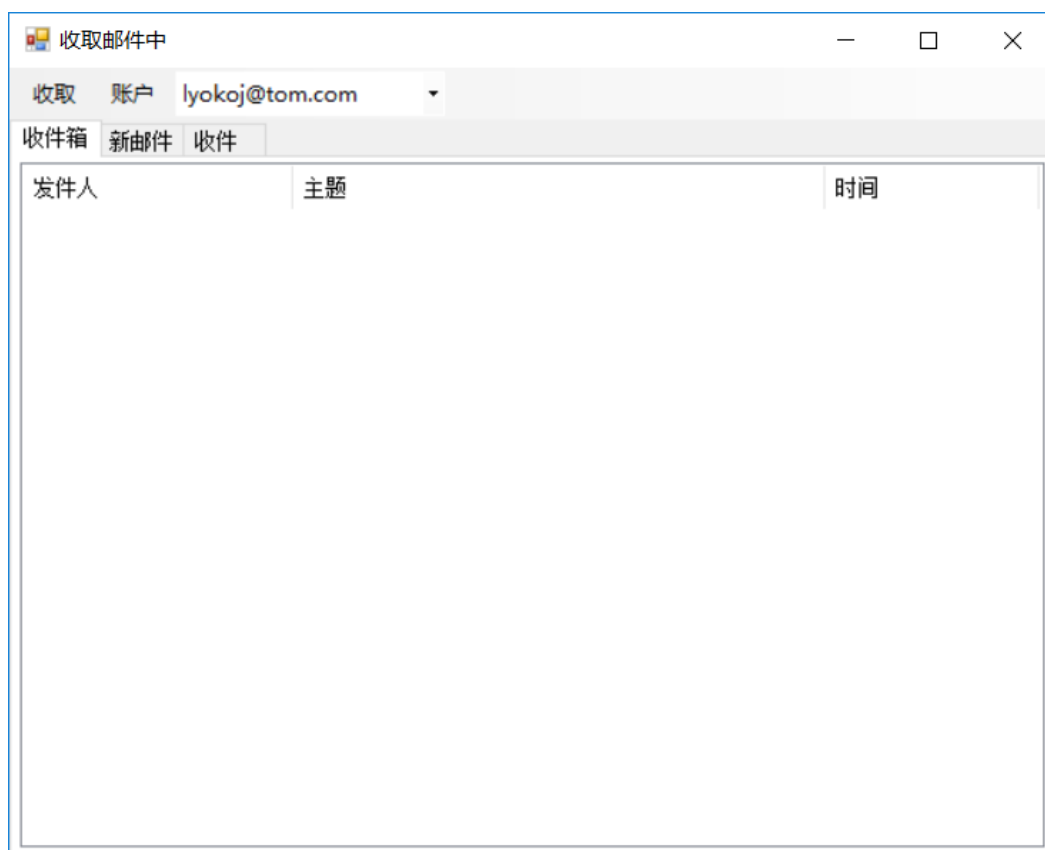
邮件系统初始界面如下：



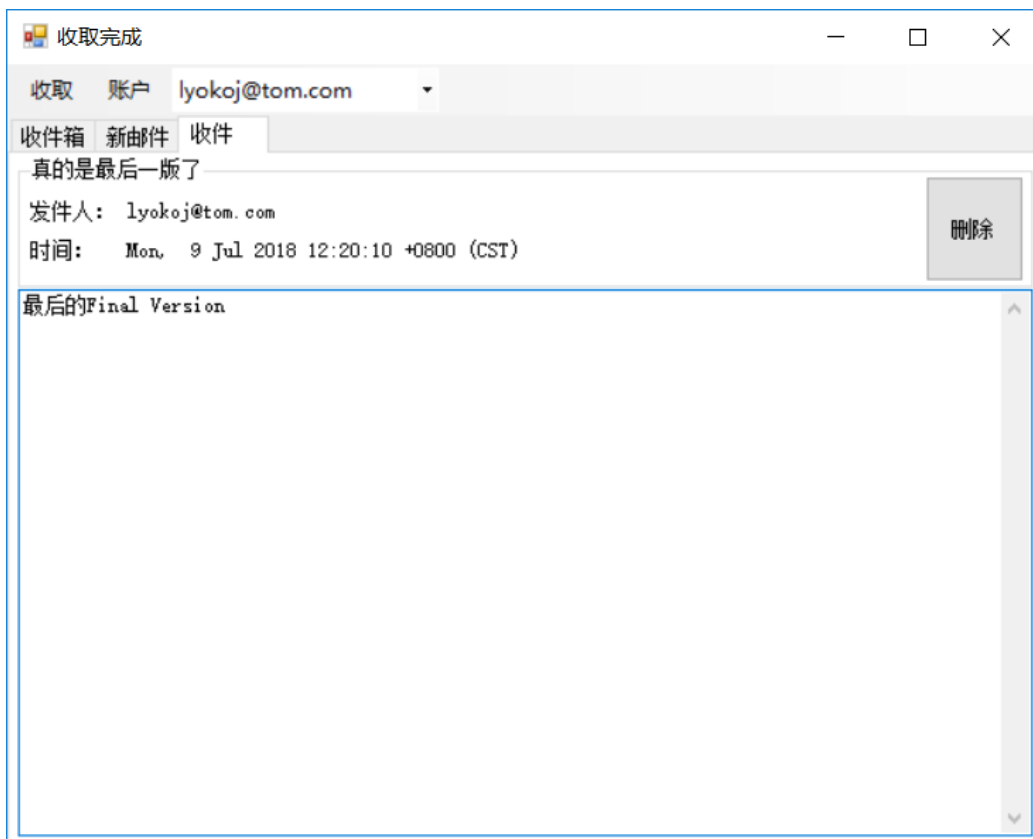
新建账户如下：



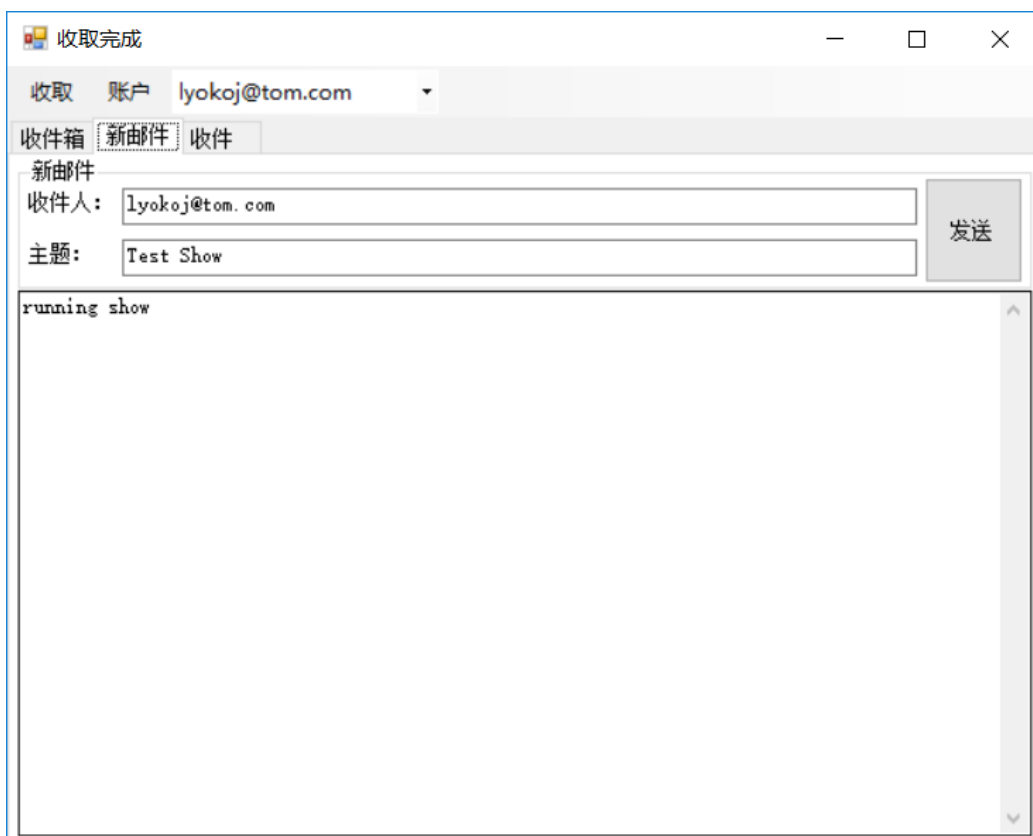
点击收取之后获得邮件：

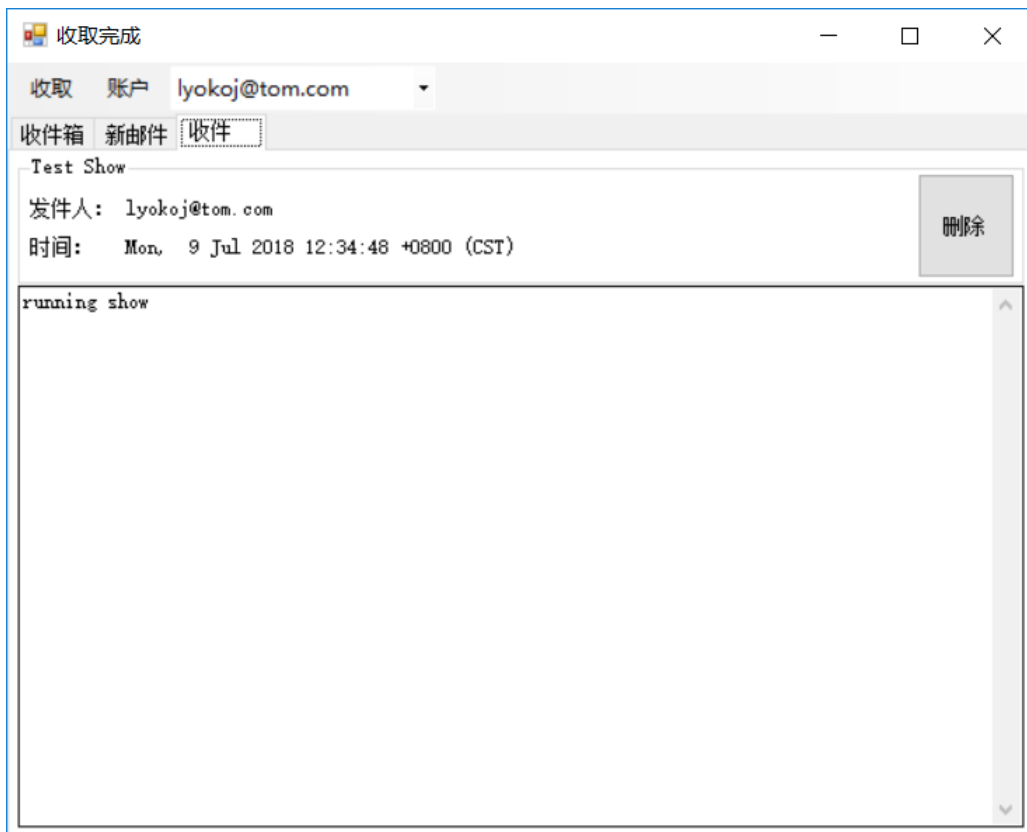


双击发件人名字进入邮件内容

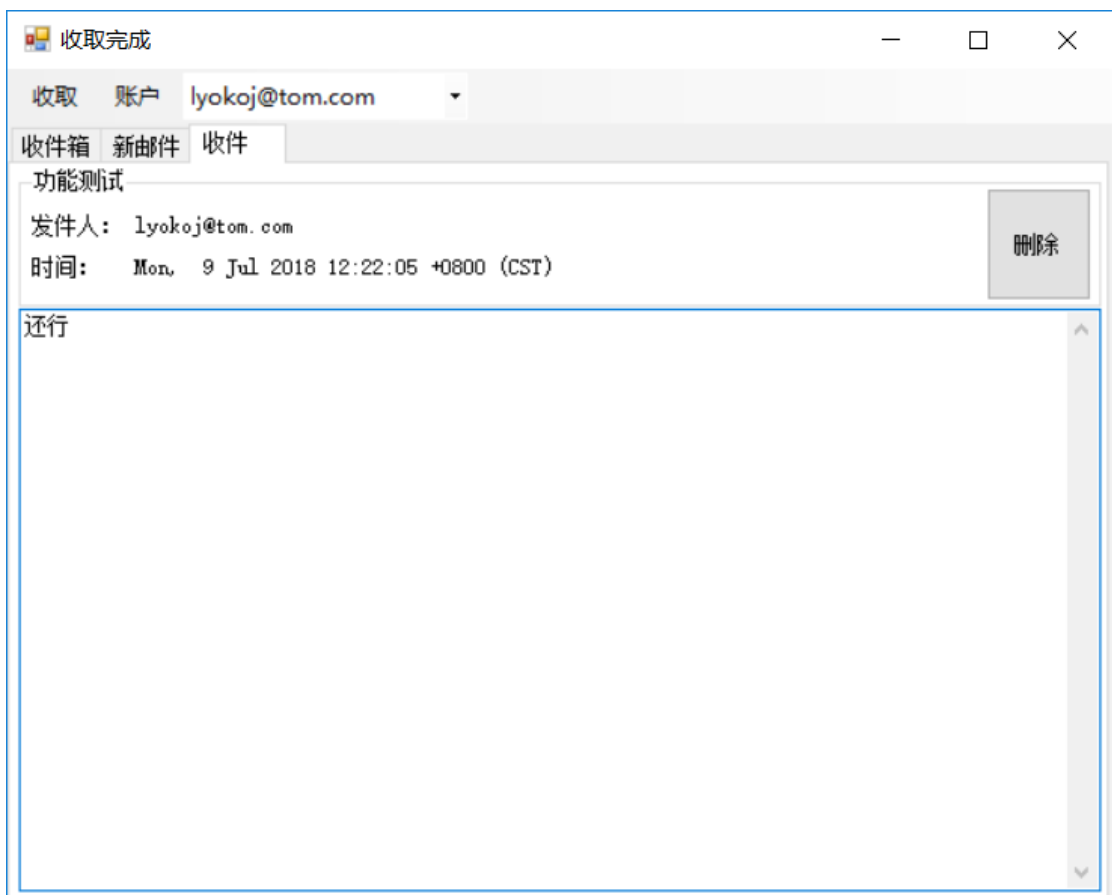


点击新邮件可以编写并发送邮件

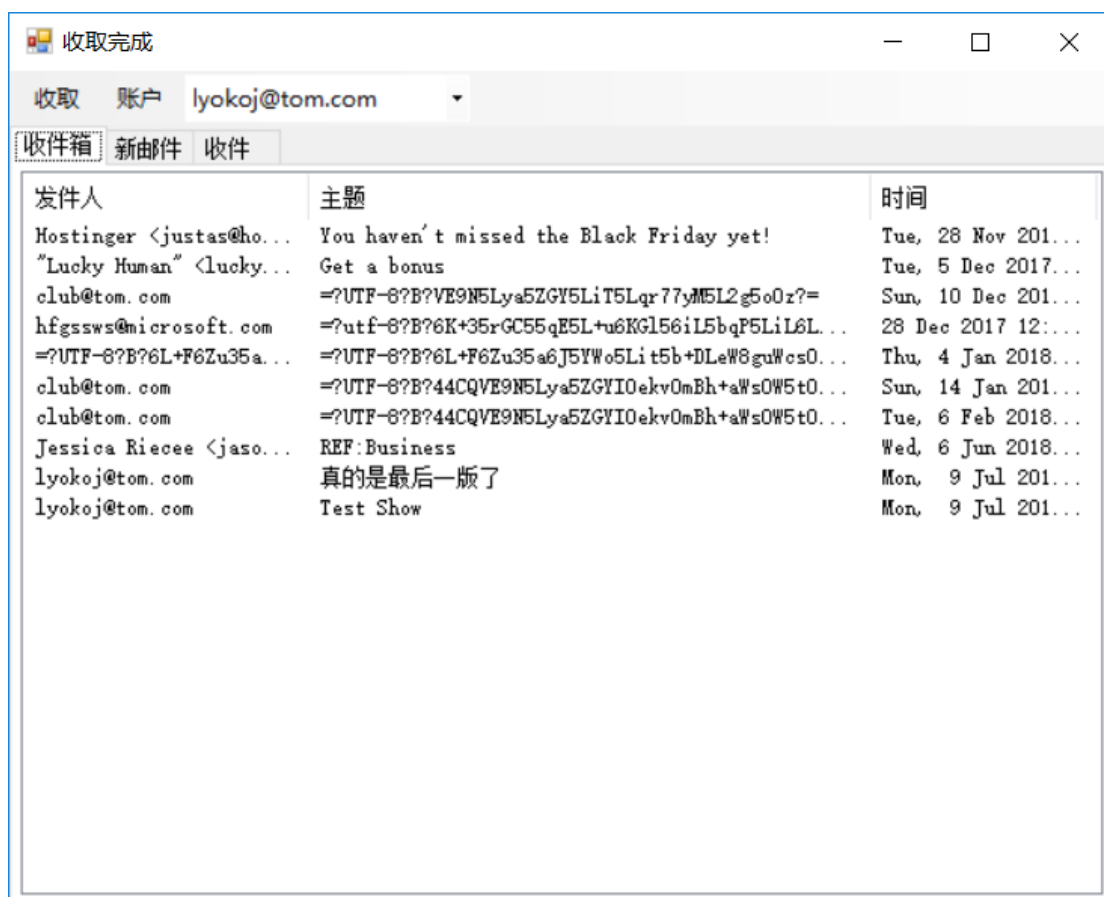
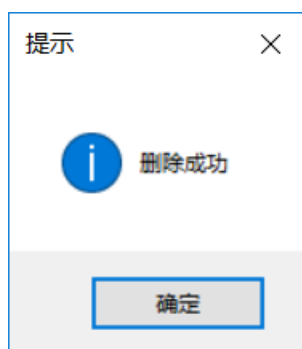




在收件里阅读邮件的时候可以删除邮件



点击删除后再次接受，将看不到主题为“功能测试”的邮件



3.2 程序源程序

1. Mail.cs: 邮件类，定义邮件的基本信息（主题，日期，来源等）

```
using System;
using System.Collections.Generic;
using System.Text;

namespace MailClient
{
    class Mail
```

```

{
    string subject;
    string date;
    string from;
    string content;
    string to;
    int id;

    public static char []cr_lf = { (char)13, (char)10 };
    public static string crlf = new string(cr_lf);
    public static string mailend = crlf + "." + crlf;

    public string Subject { get => subject; set => subject = value; }
    public string Date { get => date; set => date = value; }
    public string From { get => from; set => from = value; }
    public string Content { get => content; set => content = value; }
    public int Id { get => id; set => id = value; }
    public string To { get => to; set => to = value; }

    Mail() { }

    string getattri(string attri, string str)
    {
        string ans = "";

        int pos = str.IndexOf(attri);
        if (pos == -1)
            return ans;
        pos += attri.Length;

        while (str.Substring(pos, 2) != crlf)
            ans += str[pos++];
        return ans;
    }

    public Mail(string str)
    {
        subject = getattri("Subject: ", str);
        date = getattri("Date: ", str);
        from = getattri("From: ", str);

        int pos = str.IndexOf(crlf+crlf);
        pos += 4;
        content = "";
    }
}

```

```

        while (str.Substring(pos, 5) != crlf + "." + crlf)
            content += str[pos++];
    }

    public Mail(string to, string subject, string content)
    {
        this.subject = subject;
        this.content = content;
        this.to = to;
    }

    public string toString()
    {
        return "Subject: " + subject + crlf + content + mailend;
    }
}

```

2. MailAccount.cs: 邮件账户类，用于定义邮件的账户类型

```

using System;
using System.Collections.Generic;
using System.Text;

namespace MailClient
{
    public class MailAccount
    {
        string username;
        string password;

        string pop3add;
        int pop3port;

        string smtpadd;
        int smtpport;

        public MailAccount()
        {
        }

        public MailAccount(string username, string password,
            string pop3add, int pop3port,
            string smtpadd, int smtpport)
        {
        }
    }
}

```

```

{
    this.username = username;
    this.password = password;
    this.pop3add = pop3add;
    this.pop3port = pop3port;
    this.smtpadd = smtpadd;
    this.smtpport = smtpport;
}

public string Username { get => username; set => username = value; }
public string Password { get => password; set => password = value; }
public string Pop3add { get => pop3add; set => pop3add = value; }
public int Pop3port { get => pop3port; set => pop3port = value; }
public string Smtppadd { get => smtpadd; set => smtpadd = value; }
public int Smtpport { get => smtpport; set => smtpport = value; }
}
}

```

3. SMTP.cs: SMTP类, 用于实现SMTP服务

```

using System;
using System.Collections.Generic;
using System.Text;

namespace MailClient
{
    class SMTP
    {
        MailAccount account;
        NetService net;
        string info;
        Object infolock;

        static string []codes = { "220", "250", "334", "235", "354", "221" };
        public static string suc = "成功";

        SMTP() { }

        public SMTP(MailAccount account)
        {
            this.account = account;
            info = "";
            infolock = new Object();
        }
    }
}

```

```

void setInfo(string str)
{
    lock (infolock)
    {
        info = str;
    }
}

bool step(string cmd, string statuscode)
{
    string res = "";
    while (!net.send(cmd)) ;
    while (" " == (res = net.getmsg())) ;
    string[] strs = res.Split(new char[] { ' ', '-' });
    if (strs[0] != statuscode)
    {
        setInfo(res);
        return false;
    }
    return true;
}

bool login()
{
    string res = "";
    net = new NetService(account.Smtpadd, account.Smtpport);
    while (" " == (res = net.getmsg())) ;
    string[] strs = res.Split(new char[] { ' ', '-' });
    if (strs[0] != codes[0])
    {
        setInfo(res);
        return false;
    }

    if (!step("EHLO " + account.Smtpadd + Mail.crlf, codes[1]))
        return false;

    if (!step("AUTH LOGIN" + Mail.crlf, codes[2]))
        return false;

    string u64 =
Convert.ToBase64String(Encoding.UTF8.GetBytes(account.Username));
    if (!step(u64 + Mail.crlf, codes[2]))
        return false;
}

```

```

        string p64 =
Convert.ToBase64String(Encoding.UTF8.GetBytes(account.Password));
        if (!step(p64 + Mail.crlf, codes[3]))
            return false;

        return true;
    }

    bool send(Mail mail)
    {
        if (!step("MAIL FROM:" + account.Username + Mail.crlf, codes[1]))
            return false;

        if (!step("RCPT TO:" + mail.To + Mail.crlf, codes[1]))
            return false;

        if (!step("DATA" + Mail.crlf, codes[4]))
            return false;

        if (!step(mail.toString(), codes[1]))
            return false;

        return true;
    }

    void quit()
    {
        net.send("QUIT" + Mail.crlf);
    }

    public void sendmail(Mail mail)
    {
        if (login() && send(mail))
            setInfo(suc);
        quit();
    }

    public string getInfo()
    {
        lock (infoLock)
        {
            return info;
        }
    }

```

```

    }
}
}

```

4. POP3.cs: POP3类，用于实现POP3的客户端服务

```

using System;
using System.Collections;
using System.Collections.Generic;
using System.Text;
using System.Timers;

namespace MailClient
{
    class POP3
    {
        ArrayList mails;
        MailAccount account;
        NetService net;
        string info;
        Object infolock;

        string[] errs = { "服务器响应错误", "用户名错误", "用户名或密码错误" };
        public string suc = "成功";

        POP3() { }

        public POP3(MailAccount account)
        {
            this.account = account;
            mails = new ArrayList();
            info = "";
            infolock = new Object();
        }

        void setInfo(string str)
        {
            lock (infolock)
            {
                info = str;
            }
        }

        string step(string cmd, string err)
        {

```

```

    string res = "";
    while (!net.send(cmd)) ;
    while (" " == (res = net.getmsg())) ;
    if (res[0] != '+')
        setInfo(err);
    return res;
}

bool login()
{
    string res = "";
    net = new NetService(account.Pop3add, account.Pop3port);
    while (" " == (res = net.getmsg())) ;
    if (res[0] != '+')
    {
        setInfo(errs[0]);
        return false;
    }

    if (step("USER " + account.Username + Mail.crlf, errs[1])[0] != '+')
        return false;

    if (step("PASS " + account.Password + Mail.crlf, errs[2])[0] != '+')
        return false;

    return true;
}

bool recv()
{
    string res = "";
    if ((res = step("LIST" + Mail.crlf, errs[0]))[0] != '+')
        return false;
    while (!res.Contains(Mail.mailend))
        res += net.getmsg();

    string []lis = res.Split(Mail.cr_lf,
StringSplitOptions.RemoveEmptyEntries);
    int len = lis.Length;
    string[] id_size = lis[len - 2].Split(' ');

    int total = Convert.ToInt32(id_size[0]);
    for (int i = 1; i <= total; i++)
    {

```

```

        if ((res = step("RETR " + i.ToString() + Mail.crlf, errs[0]))[0] !=
'+' )

            return false;
        while (!res.Contains(Mail.mailend))
            res += net.getmsg();

        Mail mail = new Mail(res);
        mail.Id = i-1;
        mails.Add(mail);
        System.Threading.Thread.Sleep(1);
    }
    return true;
}

bool delete(int id)
{
    string res = "";
    if ((res = step("DELE " + id.ToString() + Mail.crlf, errs[0]))[0] != '+' )
        return false;
    return true;
}

void quit()
{
    net.send("QUIT"+Mail.crlf);
}

public void threadingRetrieve()
{
    new System.Threading.Thread(retrieve).Start();
}

void retrieve()
{
    mails.Clear();
    if (login() && recv())
        setInfo(suc);
    quit();
}

public void remove(int id)
{
    if (login() && delete(id))
        setInfo(suc);
}

```

```

        quit();
    }

    public ArrayList getMails()
    {
        lock (infoLock)
        {
            if (info == suc)
                return mails;
            else
                return null;
        }
    }

    public string getInfo()
    {
        lock (infoLock)
        {
            return info;
        }
    }
}

```

5. NetService.cs: NetService类，封装了底层的通信函数，用于TCP的异步通信

```

using System;
using System.Collections.Generic;
using System.IO;
using System.Net.Sockets;
using System.Text;
using System.Threading;

namespace MailClient
{
    class NetService
    {
        NetworkStream nwStream;
        TcpClient sock;
        string add;
        int port;
        bool quit;

        Object sendlock;
    }
}

```

```

string msgsend;
Queue<string> msgrecv;

static char[] cr_lf = { (char)13, (char)10 };
static string crlf = new string(cr_lf);

NetService() { }

public NetService(string add, int port)
{
    quit = false;
    this.add = add;
    this.port = port;
    msgsend = "";

    sock = new TcpClient();
    msgrecv = new Queue<string>();
    sendlock = new Object();
    new Thread(connect).Start();
}

void connect()
{
    sock.Connect(add, port);
    nwStream = sock.GetStream();
    Thread tsend = new Thread(new ThreadStart(send));
    Thread trecv = new Thread(new ThreadStart(recv));
    tsend.Start();
    trecv.Start();

    tsend.Join();
    trecv.Join();

    sock.Close();
    //System.Windows.Forms.MessageBox.Show("Session Closed");
}

void send()
{
    while (!quit)
    {
        lock (sendlock)
        {
            if (msgsend != "")

```

```

    {
        lock (nwStream)
        {

            if (nwStream.CanWrite)
            {
                byte[] data = Encoding.UTF8.GetBytes(msgsend);
                nwStream.Write(data, 0, data.Length);

                if (msgsend == "QUIT" + crlf)
                    quit = true;

                msgsend = "";
            }
        }
    }
    Thread.Sleep(10);
}

void recv()
{
    while (!quit)
    {
        lock (nwStream)
        {
            if (nwStream.DataAvailable)
            {
                byte[] data = new byte[1024];
                int len = 1024;
                MemoryStream msl = new MemoryStream();

                while (true)
                {
                    if (nwStream.CanRead)
                    {
                        int r = nwStream.Read(data, 0, len);
                        if (r <= 0)
                            continue;
                        msl.Write(data, 0, r);
                        if
(Encoding.Default.GetString(msl.GetBuffer()).Contains(crlf))
                            break;
                    }
                }
            }
        }
    }
}

```

```

        lock (msgrecv)

msgrecv.Enqueue(Encoding.UTF8.GetString(msl.GetBuffer()).TrimEnd('\0'));
    }
}
Thread.Sleep(10);
}
}

public bool send(string send)
{
    lock (sendlock)
    {
        if (msgsend == "")
            msgsend = send;
        return msgsend != "";
    }
}

public string getmsg()
{
    string msg = "";
    lock (msgrecv)
    {
        while (msgrecv.Count > 0)
            msg += msgrecv.Dequeue();
    }
    return msg;
}
}
}

```

参考文献

- [1] 《网络编程实用教程》（叶树华等编著，人民邮电出版社）
- [2] SIMPLE MAIL TRANSFER PROTOCOL. Jonathan B Postel. RFC 821 . 1982
- [3] Post Office Protocol-Version 3. J. Myers, M. Rose. RFC 1939 . 1996
- [4] SMTP 与 ESMTP. [DB/OL] <https://blog.csdn.net/mhfh611/article/details/9423649>

教师评语评分

评语： _____

评分： _____

评阅人：周浩

年 月 日

（备注：对该实验报告给予优点和不足的评价，并给出百分之评分。）