# 毕业设计说明书

## 电子邮件客户端软件

作　者：_____ 学 号：_____

电子与计算机科学技术学院

学院(系)：计算机科学与技术系

专　业：计算机科学与技术

指导教师：_____

_____

评 阅 人：_____

2006　年　6　月

# 电子邮件客户端软件


## 摘要

电子邮件在当今社会中扮演了一个很重要的角色。越来越多的人在使用它。而且用它的人数势必会继续增加。虽然，现在已经有很多的邮件收发软件例如著名的 FoxMail 但是对于大多数的非专业的人来说它还是有点难度稍嫌负责。因此，我们就利用 SMTP 和 Pop 协议从底层开发了这个软件。SMTP 全称是简单邮件传输协议，它专门用来发送邮件用的。Pop 全称是邮局协议，是专门用于接收邮件的。我主要是负责如何实现发送邮件功能的。MailSend 命名空间是我整个程序的核心。它包括两个类。在 SmtpMail 的类中包含了一个 SendMail 的方法，它从底层详细地实现了和服务器的交互操作。你既可以用它发送一个纯文本邮件，也可以发送一个带有附件的邮件，理所当然地，你也可以使用不同的 SMTP 服务器。经过测试，证实此软件是一个支持多收信人，多附件的群发软件。虽然它没有 FoxMail 那么强大的功能，但是它容易掌握和使用。


**关键词： SMTP，命名空间，类，附件**

# E-Mail Client SoftWare


## Abstract

E-Mail play a very important role in modern times.More and more people are using it,and the number of it will larger and larger.Though there are a lot of software for sending and receiving letters such as FoxMail which are also multifunctional,it is difficult and complicated to the Most of people who are curbstone.For this reason,we do this software with the rock-bottom protocol of SMTP and Pop. The full name of SMTP is Simple Mail Transfer Protocol.It is Used to sending letters.The full name of Pop is Post Office Protocol which is Special to receive letters.I basically take charge to how to realize the function of sending letters. A namespace which is named MailSend is the soul of my programe.It includes two classes.A method named sendmial which realize the fuction step by step belongs to the class of SmtpMail. It detailedly note the track of client exchange to the server. You can use the software to send either a text E –Mail or a textE-mail with Attachments.You also can Send a letter to many addressee. In the nature of things,you can use a different SMTP service. The software I did support multiletters and multisender after I test.It is simplier than FoxMail and other professional softwares,but it is easy to hold and use.


**Key Words:**SMTP，nameSpace，Class，Attachment

# 目 录

1 引言 …… 7
1．1 电子邮件介绍 …… 7
1．2 开发背景 …… 8
1．3 开发环境及运行环境 …… 8
2 软件架构及系统用例图 …… 9
2．1 系统架构 …… 9
2．2 系统总体用例 …… 9
2．4 发送邮件类 …… 10
2．5 附加小功能类 …… 10
3 SMTP 协议的研究 …… 10
3．1 SMTP 协议简介及工作原理 …… 11
3．2 SMTP 协议的命令和应答 …… 12
3．2．1 SMTP 协议的命令 …… 12
4 RFC822 …… 20
4．1 RFC822 简单介绍 …… 20
4．2 信件的头部 …… 20
5 命名控件 MailSend …… 26
5．1 发送邮件类 SmtpMail …… 26
5．2 AddExtra 类 …… 33
5．2．1 调用 Windows API 所需的命名空间 …… 33
5．2．3 在程序中具体的使用 …… 34
6 软件运行时的界面 …… 35
6．1 新建邮件帐号 …… 35
7 系统测试 …… 38
7．1 同一 SMTP 服务器发送邮件的测试 …… 38
7．2 利用不同的 SMTP 服务器发送邮件的测试 …… 38

8　结论 …… 40

参考文献 …… 41

致 谢 …… 42

外文文献原文

译文

**毕 业 设 计 开 题 报 告**

# 1 引言

## 1．1 电子邮件介绍

电子邮件(简称 E-mai1)又称电子信箱、电子邮政，它是一种用电子手段提供信息交换的通信方式。它是全球多种网络上使用最普遍的一项服务。这种非交互式的通信,加速了信息的交流及数据传送,它是一个简易、快速的方法。通过连接全世界的 Internet,实现各类信号的传送、接收、存贮等处理,将邮件送到世界的各个角落。到目前为止,可以说电子邮件是 Internet 资源使用最多的一种服务，E-mai1 不只局限于信件的传递,还可用来传递文件、声音及图形、图像等不同类型的信息。

电子邮件不是一种“终端到终端”的服务,是被称为“存贮转发式”服务。这正是电子信箱系统的核心，利用存贮转发可进行非实时通信，属异步通信方式。即信件发送者可随时随地发送邮件，不要求接收者同时在场，即使对方现在不在，仍可将邮件立刻送到对方的信箱内,且存储在对方的电子邮箱中。接收者可在他认为方便的时候读取信件,不受时空限制。在这里,“发送”邮件意味着将邮件放到收件人的信箱中,而“接收”邮件则意味着从自己的信箱中读取信件,信箱实际上是由文件管理系统支持的一个实体。因为电子邮件是通过邮件服务器(mai1 server)来传递档的。通常 mail server 是执行多任务操作系统 UNIX 的计算机，它提供 24 小时的电子邮件服务,用户只要向 mail server 管理人员申请一个信箱账号,就可使用这项快速的邮件服务。

电子邮件的工作原理：

1) 电子邮件系统是一种新型的信息系统,是通信技术和计算机技术结合的产物。

电子邮件的传输是通过电子邮件简单传输协议(Simple Mail Transfer Protocol,简称 SMTP)这一系统软件来完成的,它是 Internet 下的一种电子邮件通信协议。

2) 电子邮件的基本原理,是在通信网上设立“电子信箱系统”，它实际上是一个计算机系统。系统的硬件是一个高性能、大容量的计算机。硬盘作为信箱的存储介质,在硬盘上为用户分一定的存储空间作为用户的“信箱”，每位用户都有属于自己的一个电子信箱。并确定一个用户名和用户可以自己随意修改的口令。存储空间包含存放所收信件、编辑信件以及信件存盘三部分空间,用户使用口令开启自己的信箱,并进行发信、读信、编辑、转发、存档等各种操作。系统功能主要由软件实现。

3) 电子邮件的通信是在信箱之间进行的。用户首先开启自己的信箱，然后通过键入命令的方式将需要发送的邮件发到对方的信箱中。邮件在信箱之间进行传递和交

换,也可以与另一个邮件系统进行传递和交换。收方在取信时，使用特定账号从信箱提取。

## 1．2　开发背景

当前流行的各大邮件客户端软件的除了最主要的收发信件之外，功能越来越复杂，但是人们平常真正用到的功能很少,很多功能尤其对于那些计算机知识相对缺乏的人来说，更加显得太过于华丽而不太实用。有鉴于此，在了解 RFC 底层协议的基础上，我们开发了这个各种功能相对简单实用的邮件客户端程序，简化了很多不必要的功能。

## 1．3　开发环境及运行环境

### 1．3．1　开发环境

AMD Athlon(TM)，512M 内存，80G 硬盘

Microsoft® Windows™ XP Professional

Microsoft® Visual Studio 2003(C Sharp)

Microsoft® Developer Network for Visual Studio.NET 2003

### 1．3．2　运行环境

Intel® Pentium® 2 及以上处理器，32M 以上内存，4G 以上硬盘

Microsoft® Windows™ 9X/NT 操作系统

800*600 或以上的屏幕分辨率

确保机器上安装有.Net FrameWork 1.0 或者以上版本

# 2　软件架构及系统用例图

## 2．1　系统架构

软件的总体架构如图 2.1：

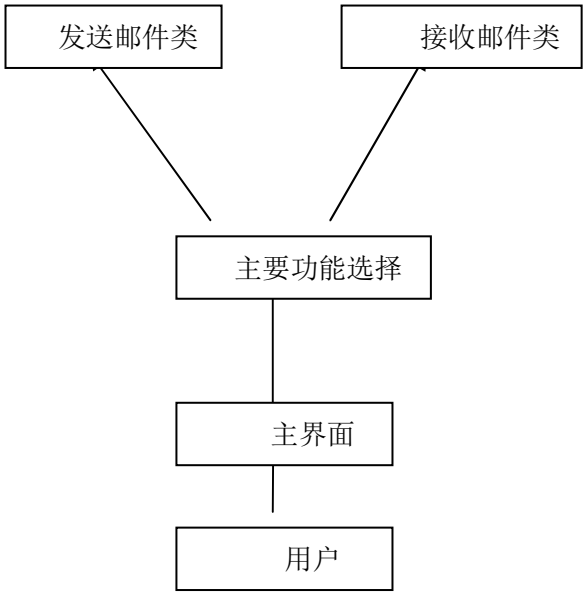



图 2.1　软件架构图

## 2．2　系统总体用例

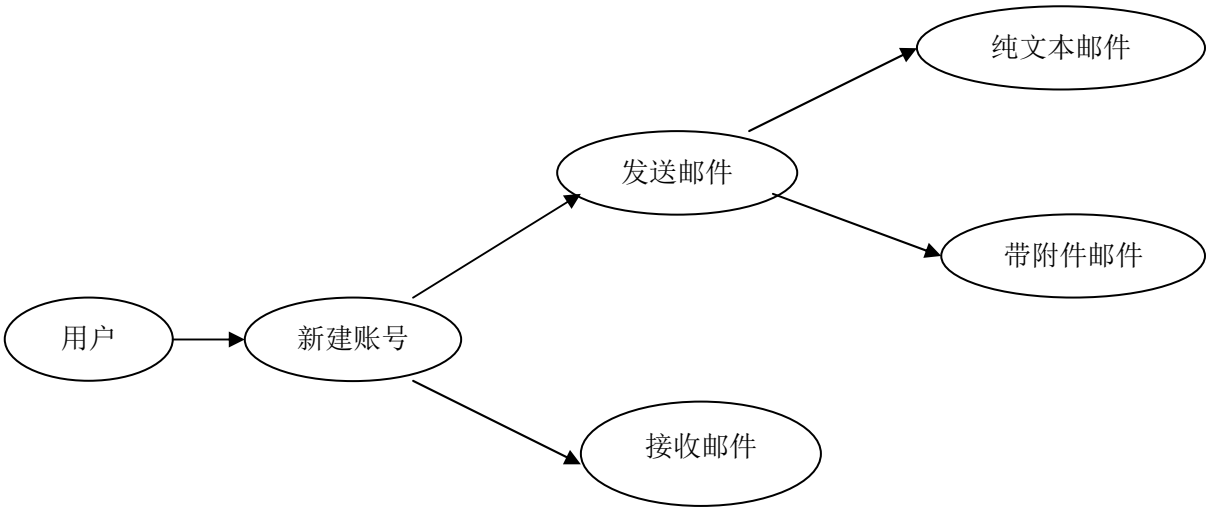



图 2.2　系统总体用例图

## 2．3　程序功能框图

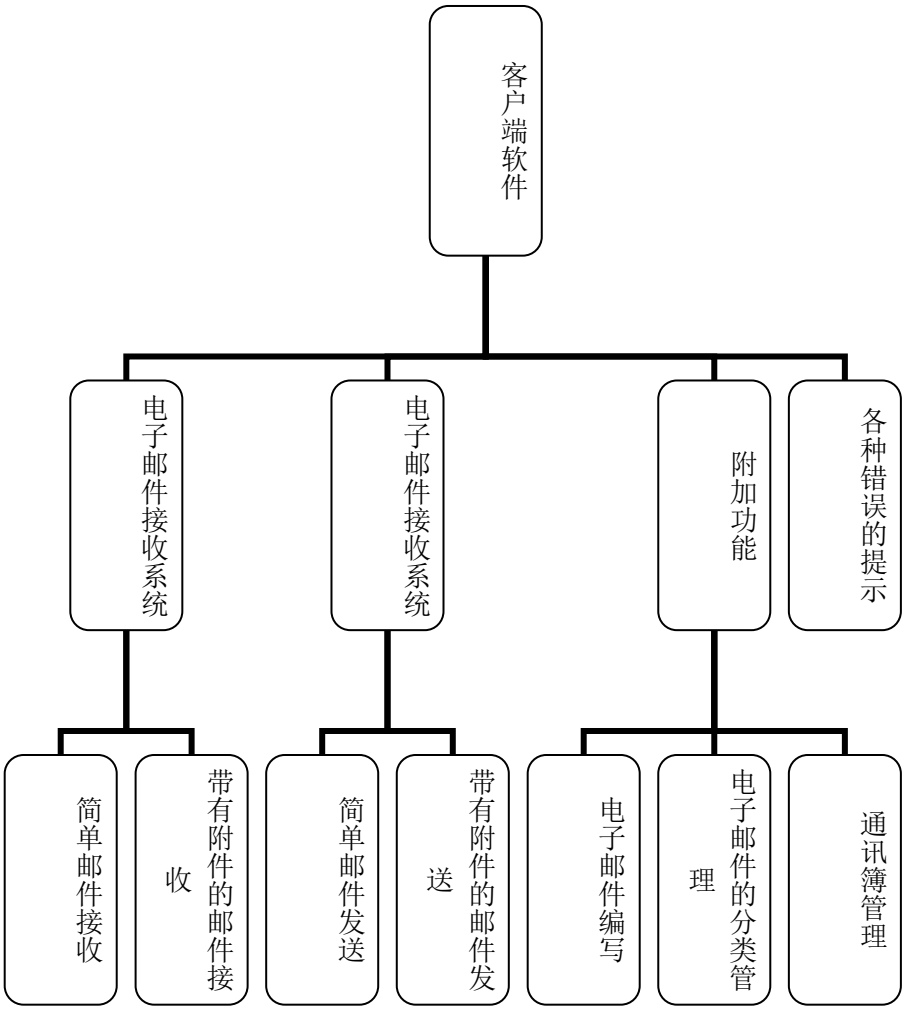



图 2.3　程序功能图

### 2．4　发送邮件类

是发送邮件的核心，类名为 SmtpMail，隶属于命名空间 MailSend。封装了发送邮件的具体实现方法，也是具体的 RFC 用代码实现的过程。而用户通过具体的操作接口，接口与 SmtpMail 类通过交互操作来实现用户发送信件的操作。

### 2．5　附加小功能类

是获取一些诸如系统时间，当前用户名，以及本机 IP 之类的类，类名为 AddExtra，隶属于命名空间 MailSend。

## 3　SMTP 协议的研究

由于要开发的是邮件客户端程序，就不得不用到 SMTP 协议和 POP 协议。而我个人负责的是邮件发送功能的实现，因此就必然会涉及到 SMTP（Simple Mail Transfer Protocol）协议。SMTP 被用来在因特网上发送邮件，该协议规定了一些基本的命令

和方法使客户端与服务器进行交互，以达到发送邮件的目的。

## 3．1　SMTP 协议简介及工作原理

### 3．1．1　介绍

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 的一个重要特点是它能够在传送中接力传送邮件，传送服务提供了进程间通信环境（IPCE），此环境可以包括一个网络，几个网络或一个网络的子网。理解到传送系统（或 IPCE）不是一对一的是很重要的。进程可能直接和其它进程通过已知的 IPCE 通信。邮件是一个应用程序或进程间通信。邮件可以通过连接在不同 IPCE 上的进程跨网络进行邮件传送。更特别的是，邮件可以通过不同网络上的主机接力式传送。

### 3．1．2　SMTP 模型

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。

SMTP 提供传送邮件的机制，如果接收方与发送方连接在同一个传送服务下时，邮件可以直接由发送方主机传送到接收方主机；或者，当两者不在同一个传送服务下时，通过中继 SMTP 服务器传送。为了能够对 SMTP 服务器提供中继能力，它必须拥有最终目的主机地址和邮箱名称。

MAIL 命令参数是回复路径，它指定邮件从何处来；而 RCPT 命令的参数是转发路径的，它指定邮件向何处去。向前路径是源路径，而回复路径是返回路径（它用于发生错误时返回邮件）。

当同一个消息要发往不同的接收者时，SMTP 遇到了向不同接收者发送同一份数据的复制品的问题，邮件命令和应答有一个比较奇怪的语法，应答也有一个数字代码。在下面，例子中可以看到哪些使用实际的命令和应答。完整的命令和应答在第四节。

命令与应答对大小写不敏感，也就是说，命令和应答可以是大写，小写或两者的混合，但这一点对用户邮件名称却不一定是对的，因为有的主机对用户名大小写是敏感的。这样 SMTP 实现中就将用户邮箱名称保留成初始时的样子，主机名称对大小写不敏感。

命令与应答由 ASCII 字母表组成，当传送服务提供 8 位字节传送通道，每 7 位字符正确传送，而最高位被填充为 0。当指定一般的命令或应答格式后，参数会由一些类似于语言的字符串表示出来，如"<string>"或"<reverse-path>"，这里尖括号表示这是一种类似于语言的变量。

## 3．2　SMTP 协议的命令和应答

### 3．2．1　SMTP 协议的命令

SMTP 命令定义了邮件传输或由用户定义的系统功能。它的命令是由<CRLF>结束的字符串。而在带有参数的情况下，命令本身由<SP>和参数分开，如果未带参数可以直接和<CRLF>连接。邮箱的语法格式必须和接收站点的格式一致。下面讨论 SMTP 命令和应答。

发送邮件操作涉及到不同的数据对象，它们由不同的参数相互连接。回复路径就是 MAIL 命令的参数，而转发路径则是 RCPT 命令的参数，邮件日期是 DATA 命令的参数。这些参数或者数据对象必须跟在命令后。这种模式也就要求有不同的缓冲区来存储这些对象，也就是说，有一个回复路径缓冲区，一个转发路径缓冲区，一个邮件内容缓冲区。特定的命令产生自己的缓冲区，或使一个或多个缓冲的内容被清除。

HELLO (HELO)

此命令用于向接收 SMTP 确认发送 SMTP。参数域包括发送 SMTP 的主机名。接收 SMTP 通过连接确认命令来向发送 SMTP 确认接收 SMTP。引命令和 OK 响应确认发送和接收 SMTP 进入了初始状态，也就是说，没有操作正在执行，所有状态表和缓冲区已经被子清除。

MAIL (MAIL)

此命令用于开始将邮件发送到一个多个邮箱中。参数域包括回复路径。返回路径中包括了可选的主机和发送者邮箱列表。当有主机列表时，它是一个回复路径源，它说明此邮箱是由在表中的主机一一传递发送（第一个主机是最后一个接收到此邮件的主机）过来的。此表也有作向发送者返回非传递信号的源路径。因为每个传递主机地址都被加在此表起始处，它就必须使用发送 IPCE 而不是接收 IPCE（如果它们不是一

个 IPCE 的话）清楚的名称。一些出错信息的回复路径可能就是空的。

此命令清除回复路径缓冲区，转发路径缓冲区和邮件内容缓冲区，并且将此命令的回复路径信息插入到回复路径缓冲区中。

RECIPIENT (RCPT)

此命令用于确定邮件内容的唯一接收者；多个接收者将由多个此命令指定。转发路径中包括一个可选的主机和一个必须的目的邮箱。当出现主机列表时，这就是一个源路径，它指明邮件必须向列表中的上一个主机发送。如果接收 SMTP 未实现邮件的传递发送，就会返回如未知本地用户（550）的信息给用户。

当邮件被传递发送时，传递主机必须将自己的名称由转发路径的开始处移至回复路径的结束处。当邮件最终到达目的地时，接收 SMTP 将以它的主机邮件格式自己的名称插入目标邮件中。例如，由传递主机 A 接收的带有如下参数的邮件时，

FROM:<USERX@HOSTY.ARPA>

TO:<@HOSTA.ARPA,@HOSTB.ARPA:USERC@HOSTD.ARPA>

将会变成如下形式：

FROM:<@HOSTA.ARPA:USERX@HOSTY.ARPA>

TO:<@HOSTB.ARPA:USERC@HOSTD.ARPA>.

此命令导致它的转发路径参数加入转发路径缓冲区中。

DATA (DATA)

接收者将跟在命令后的行作为邮件内容。此命令导致此命令后的邮件内容加入邮件内容缓冲区。邮件内容可以包括所有 128 个 ASCII 码字符。邮件内容由只包括一个句号的行结束，也就是如下的字符序列："<CRLF>.<CRLF>"，它指示了邮件的结束。

邮件内容的结束指示要求接收者现在就处理保存的邮件内容。此过程将回复路径缓冲区，转发路径缓冲区和邮件内容缓冲区的内容全部清空。如果操作成功，接收者必须返回 OK 应答；如果失败也必须返回失败应答。

当接收 SMTP 收到一条信息时，无论是用作转发还是此邮件已经到达目的地，它都必须在邮件内容的开始处加上时间戳这一行，这一行指示了接收到邮件主机和发出此邮件主机的标识，以及接收到邮件内容的时间和日期。转发的信件将有多行这样的时间戳。当接收 SMTP 作最后一站的传送时，它将返回路径信息行插入邮件中。此行包括了发送命令中的<reverse-path>的信息。在这里，最后一站的传送的意思是邮件将被送到目的用户手中，但在一些情况下，邮件可能需要更进一步的加工并由另外的邮

件系统传送。

可能在返回路径中的邮箱与实际发送的邮件不一致，这个情况可能发生在需要传送一个特定的错误处理信箱而不是信件发送者那里。上面所述说明了，最后的邮件内容由一个返回路径行，和在其后的一个或多个时间戳行构成。这些行后面是邮件内容的头和体信息。

当处理后面的邮件数据指示部分成功时就需要特定的说明。这种情况可能发生在发送 SMTP 发现当邮件需要传送给多个用户时，只能够成功地向其中的一部分发送信息这种情况下。在这种情况下，必须对 DATA 命令发送 OK 应答，而接收 SMTP 组织并发送一个"不可传递邮件"信息到信息的发送者。在此信息中或者发送一个不成功接收者的列表，或者每次发送一个不成接收者，而发送多次。所有不可传递邮件信息由 MAIL 命令发送。

返回路径和接收时间戳例子

Return-Path: <@GHI.ARPA,@DEF.ARPA,@ABC.ARPA:JOE@ABC.ARPA>

Received: from GHI.ARPA by JKL.ARPA ; 27 Oct 81 15:27:39 PST

Received: from DEF.ARPA by GHI.ARPA ; 27 Oct 81 15:15:13 PST

Received: from ABC.ARPA by DEF.ARPA ; 27 Oct 81 15:01:59 PST

Date: 27 Oct 81 15:01:01 PST

From: JOE@ABC.ARPA

Subject: Improved Mailing System Installed

To: SAM@JKL.ARPA

This is to inform you that ...

SEND (SEND)

此命令用于开始一个发送命令，将邮件发送到一个或多个终端上。参数域包括了一个回复路径，此命令如果成功就将邮件发送到终端上了。

回复路径包括一个可选的主机列表和发送者邮箱。当出现主机列表时，表示这是一个传送路径，邮件就是经过这个路径上的每个主机发送到这里的（列表上第一个主机是最后经手的主机）。此表用于返回非传递信号到发送者。因为每个传递主机地址都被加在此表起始处，它就必须使用发送 IPCE 而不是接收 IPCE（如果它们不是一个 IPCE 的话）清楚的名称。一些出错信息的回复路径可能就是空的。

此命令清除回复路径缓冲区，转发路径缓冲区和邮件内容缓冲区，并且将此命令

的回复路径信息插入到回复路径缓冲区中。

SEND OR MAIL (SOML)

此命令用于开始一个邮件操作将邮件内容传送到一个或多个终端上，或者传送到邮箱中。对于每个接收者，如果接收者终端打开，邮件内容将被传送到接收者的终端上，否则就送到接收者的邮箱中。参数域包括回复路径，如果成功地将信息送到终端或邮箱中此命令成功。

回复路径包括一个可选的主机列表和发送者邮箱。当出现主机列表时，表示这是一个传送路径，邮件就是经过这个路径上的每个主机发送到这里的（列表上第一个主机是最后经手的主机）。此表用于返回非传递信号到发送者。因为每个传递主机地址都被加在此表起始处，它就必须使用发送 IPCE 而不是接收 IPCE（如果它们不是一个 IPCE 的话）清楚的名称。一些出错信息的回复路径可能就是空的。

此命令清除回复路径缓冲区，转发路径缓冲区和邮件内容缓冲区，并且将此命令的回复路径信息插入到回复路径缓冲区中。

SEND AND MAIL (SAML)

此命令用于开始一个邮件操作将邮件内容传送到一个或多个终端上，并传送到邮箱中。如果接收者终端打开，邮件内容将被传送到接收者的终端上和接收者的邮箱中。参数域包括回复路径，如果成功地将信息送到邮箱中此命令成功。

回复路径包括一个可选的主机列表和发送者邮箱。当出现主机列表时，表示这是一个传送路径，邮件就是经过这个路径上的每个主机发送到这里的（列表上第一个主机是最后经手的主机）。此表用于返回非传递信号到发送者。因为每个传递主机地址都被加在此表起始处，它就必须使用发送 IPCE 而不是接收 IPCE（如果它们不是一个 IPCE 的话）清楚的名称。一些出错信息的回复路径可能就是空的。

此命令清除回复路径缓冲区，转发路径缓冲区和邮件内容缓冲区，并且将此命令的回复路径信息插入到回复路径缓冲区中。

RESET (RSET)

此命令指示当送邮件操作将被放弃。任何保存的发送者，接收者和邮件内容应该被抛弃，所有缓冲区和状态表应该被清除，接收方必须返回 OK 应答。

VERIFY (VRFY)

此命令要求接收者确认参数是一个用户。如果这是（已经知道的）用户名，返回用户的全名和指定的邮箱。此命令对回复路径缓冲区，转发路径缓冲区和邮件内容缓

冲区没有影响。

EXPAND (EXPN)

此命令要求接收者确认参数指定了一个邮件发送列表，如果是一个邮件发送列表，就返回表中的成员。如果这是（已经知道的）用户名，返回用户的全名和指定的邮箱。此命令对回复路径缓冲区，转发路径缓冲区和邮件内容缓冲区没有影响。

HELP (HELP)

此命令导致接收者向 HELP 命令的发送者发出帮助信息。此命令可以带参数，并返回特定的信息作为应答。此命令对回复路径缓冲区，转发路径缓冲区和邮件内容缓冲区没有影响。

NOOP (NOOP)

此命令不影响任何参数和已经发出的命令。它只是说明没有任何操作而不是说明接收者发送了一个 OK 应答。此命令对回复路径缓冲区，转发路径缓冲区和邮件内容缓冲区没有影响。

QUIT (QUIT)

此命令指示接收方必须发送 OK 应答然后关闭传送信道。接收方在接到 QUIT 命令并做出响应之前不应该关闭通信信道。发送方在发送 QUIT 命令和接收到响应之前也不应该关闭信道。即使出错，也不应该关闭信道。如果连接被提前关闭，接收方应该象接收到 RSET 命令一样，取消所有等待的操作，但不恢复原先已经做过的操作。而发送方应该象接收到暂时错误（4XX）一样假定命令和操作仍在支持之中。

TURN (TURN)

此命令指定接收方要么发送 OK 应答并改变角色为发送 SMTP，要么发送拒绝信息并保持自己的角色。如果程序 A 现在是发送 SMTP，它发出 TURN 命令后接收到 OK（250）应答，它就变成了接收 SMTP。程序 A 就进入初始状态，好象通信信道刚打开一样，这时它发送 220 准备好服务信号。如果程序 B 现在是接收 SMTP，它发出 TURN 命令后接收到 OK（250）应答，它就变成了发送 SMTP。程序 A 就进入初始状态，好象通信信道刚打开一样，这时它准备接收 220 准备好服务信号。

若要拒绝改变角色，接收方可以发送 502 应答。

对于这些命令的顺序有一定的限制。对话的第一个命令必须是 HELLO 命令，此命令在此后的会话中也可以使用。如果 HELLO 命令的参数不可接受，必须由返回一个 501 失败应答，同时接收到的 SMTP 必须保持在与刚才一致的状态下。 NOOP，

HELP,EXPN 和 VRFY 命令可以在会话的任何时候使用。MAIL，SEND，SOML 或 SAML 命令开始一个邮件操作。一旦开始了以后就要发送 RCPT 和 DATA 命令。邮件操作可以由 RSET 命令终止。在一个会话中可以有一个或多个操作。

如果在操作开始参数不可接受，必须返回 501 失败应答，同时接收到的 SMTP 必须保持在与刚才一致的状态下。如果操作中的命令顺序出错，必须返回 503 失败应答，同时接收到的 SMTP 必须保持在与刚才一致的状态下。

会话的最后一个命令必须是 QUIT 命令。此命令在会话的其它时间不能使用。

COMMAND 语法格式

命令是由命令码和其后的参数域组成的。命令码是四个字母组成的，不区别大小写。因为下面的命令的作用是相同的：

MAIL Mail mail MaIl mAIl

这对于引导任何参数值的标记也是适用的，如 TO 和 to 就是一样的。命令码和参数由一个或多个空格分开。然而在回复路径和转发路径中的参数是区别大小写的。特别是在一些主机上，"smith"和"Smith"就根本不是一个用户。

参数域由不定长的字符串组成，它由<CRLF>结束，接收方在完全接收到此序列前不会采取任何行动。方括号代表可选的参数域。如果不选择的话，系统选择默认的设置。

下面是 SMTP 命令： HELO <SP> <domain> <CRLF> MAIL <SP> FROM:<reverse-path> <CRLF>

RCPT <SP> TO:<forward-path> <CRLF>

DATA <CRLF>

RSET <CRLF>

SEND <SP> FROM:<reverse-path> <CRLF>

SOML <SP> FROM:<reverse-path> <CRLF>

SAML <SP> FROM:<reverse-path> <CRLF>

VRFY <SP> <string> <CRLF>

EXPN <SP> <string> <CRLF>

HELP [<SP> <string>] <CRLF>

NOOP <CRLF>

QUIT <CRLF>

TURN <CRLF>

### 3．2．2　SMTP 的应答码

对 SMTP 命令的响应是多样的，它确定了在邮件传输过程中请求和处理的同步，也保证了发送 SMTP 知道接收 SMTP 的状态。每个命令必须有且只有一个响应。

SMTP 响应由三位数字组成，其后跟一些文本。数字帮助决定下一个应该进入的状态，而文本对人是有意义的。三位的响应已经包括了足够的信息，不用再阅读文本，文本可以直接抛弃或者传递给用户。特别的是，文本是与接收和环境相关的，所以每次接收到的文本可能不同。在附录 E 中可以看到全部的响应码。正规的情况下，响应由下面序列构成：三位的数字，<SP>，一行文本和一个<CRLF>，或者也可以是一个多行响应。只有 EXPN 和 HELP 命令可以导致多行应答，然而，对所有命令，多行响应都是允许的。

REPLY CODES BY FUNCTION GROUPS 500 格式错误，命令不可识别（此错误也包括命令行过长）

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 <domain> 服务就绪

221 <domain> 服务关闭传输信道

421 <domain> 服务未就绪，关闭传输信道（当必须关闭时，此应答可以作为对任何命令的响应）

250 要求的邮件操作完成

251 用户非本地，将转发向<forward-path>

450 要求的邮件操作未完成，邮箱不可用（例如，邮箱忙）

550 要求的邮件操作未完成，邮箱不可用（例如，邮箱未找到，或不可访问）

451 放弃要求的操作；处理过程中出错

551 用户非本地，请尝试<forward-path>

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（例如邮箱格式错误）

354 开始邮件输入，以<CRLF>.<CRLF>结束

554 操作失败

# 4 RFC822

说道发送和接受邮件，我们就必须不得不提 RFC822 了。RFC822 的全称是“ARPA 因特网文本信件格式的标准”(Standard for the Format of ARPA Internet Text Messages)。该标准提供了邮件内容的格式和相关语义。

## 4．1 RFC822 简单介绍

RFC822 规定的电子邮件内容全部由 ASCII 字符组成，就是通常所说的文本文件，因而标准将它称为 Internet 文本信件(Internet Text Messages)。

从直观上看，信件非常简单，就是一系列由 ASCII 字符组成的文本行，每一行以回车换行符(“CRLF“，就是 ASCII 码的 13 和 10)结束。

从组织上看，信件内容结构分为两大部分，中间用一个空白行(只有 CRLF 符的行)来分隔。第一部分称为信件的头部(the header of the message)，包括有关发送方、接收方、发送日期等信息。第二部分称为信件的体部(Body of the message)，包括信件内容的正文文本。信头是必需的，信体是可选的，即信体可有可无。如果不存在信体，用作分隔的空白行也就不需要。在信体中，也可以有用作分隔的空白行。这样设计的信件便于进行语法分析，提取信件的基本信息。

在 RFC822 中规定，信件体就是一系列的向收信人表达信息的文本行，比较简单，可以包含任意文本，并没有附加的结构。信件头则具有比较复杂的结构，在下一小节中详述。

## 4．2 信件的头部

### 4．2．1 信头的一般格式

信头的结构比较复杂，信头由若干信头字段(header field)组成，这些字段为用户和程序提供了关于信件的信息。要了解信头的结构就要弄清楚各种信头字段。

所有的信头字段都具有相同的语法结构，从逻辑上说，包括四部分，字段名(field name)，紧跟冒号":" (colon)，后跟字段体(field body)，最后以回车换行符(CRLF)终止。即

信头字段 = 字段名：字段体 CRLF

字段名必须由除了冒号和空格以外的可打印 US—ASCII 字符(其值在 33 和 126 之间)组成，大多数字段的字段名称由一系列字母，数字组成，中间经常插入横线符。字段名告诉电子邮件软件如何翻译该行中剩下的内容。

字段体可以包括除了 CR 和 LF 之外的任何 ASCII 字符。但是其中的空格，加括号的注释，引号和多行字段都比较复杂，另外，字段体的语法和语义依赖于字段名，每个类型的字段有特定的格式。

RFC822 为信件定义了一些标准字段，并提供了用户自行定义非标准字段的方

### 4．2．2　结构化字段和非结构化字段

每个字段所包含的信息不同，字段大体可以分为结构化字段和非结构化字段。

结构化字段有特定的格式，由语法分析程序检测。Sender 字段就是一个很好的例子，它的字段内容是信箱， 有一个离散的结构。

非结构化的字段含有任意的数据，没有固定格式。例如，Subject 字段可以含有任意的文字，并且没有固定格式。非结构化的字段数量较少，只有 Subject、Comments、扩展字段，非标准字段、IN—Reply 和 References 等。所有其它字段都是结构化的。

### 4．2．3　信头字段的元素

尽管 Email 信件的总体结构非常简单，但一些信头字段的结构是很复杂的。下面介绍一些大多数字段共有的元素。

（1）空白符

像其它文本文件一样，空白符包括空格符(ASCII 码 32)和制表符 Tab(ASCII 码 19)。此外，行末的回车换行符 CRLF 也应算是空白符。使用空白符可以对字段进行格式化，增加它的可读性。例如，每个字段间用 CRLF 来分离，在字段内用空格来分隔字段名和字段内容。在 Subject 后面的冒号和内容之间插入空格字符，会使字段结构更加清晰。在 Email 中，空白符的使用并没有固定的规则，但应当正确地使用，仅在需要时才使用空白符，以便接收软件进行语法分析。

（2）注解

注解是由括号括起来的一系列字符，例如，(这份礼物)。注解一般用在非结构化的信头字段中，没有语法语义，仅为人提供了一些附加的信息。如果在加引号的字符串中有包括在括号中的字符，那是字符串的一部分，不是注解。在解释信件的时候，会将注解忽略，可以用一个空格字符代替它们，这样就什么也不会破坏。

（3）字段折叠

每个信头字段从逻辑上说应当是一个由字段名、冒号、字段体和 CRLF 组成的单一的行，但为了书写与显示的方便，增加可读性，也为了符合 1000/80 的行字符数的限制，可以将超过 80 个字符的信头字段分为多行，即对于比较长的字段，可以分割成几行，形成折叠。在结构化和非结构化字段中都允许折叠。通过在字段中某些点插

入 CRLF 符和至少一个或多个空白字符来实现字段的折叠，第一行后面的行称为信头字段的续行。续行都以一个空白符开始，这种方法称为折叠（folding），例如标题字段 Subject: This is a test 可以表示为：

Subject: This is a test

反之，将一个被折叠成多行的信头字段恢复到它的单行表示的过程叫做去折叠，只要简单地移除后面跟着空格的 CRLF，将折叠空白符 CRLF 转换成空格字符，就可以完成去折叠(unfolding)。在分析被折叠的字段的语法时，要把一个多行的折叠字段展开为一行，根据它的非折叠的形式来分析它的语法与语义。

（4）字段大小写

字段名称是不区分大小写的，所以 Subject、subject 或 SUBJECT 都一样。不过字段名称大小写有习惯的常用形式，如主题字段的大小写形式通常为 Subject。字段体的大小写稍微复杂点，要视情况而定。比如 Subject 后面的字段体，其中的大写可能就是缩写的专用名词，不能改动。

### 4．2．4　标准的信头字段

下面介绍 RFC822 中定义的常用的标准信头字段。

表 4.1　RFC822 常用的标准信头字段

| | 与发信方有关的信头字段 |
|---|---|
| 格式：From：mailbox | 写信人字段。说明信件的原始创建者，给出他的电子信 |

| | |
|---|---|
| 举例：From：wang@163.com | 箱地址。创建者对信件的原始内容负责。 |
| 格式：Sender：mailbox<br>举例：From：wang@163.com<br>Sender：li@sina.com | 发送者字段。说明实际提交发送这个信件的人，给出他的电子信箱地址。当发信人与写信人不一样时使用。比如，秘书替经理发信。发送者对发送负责。 |
| 格式：Reply-TO：mailbox<br>举例：From：wang@163.com<br>From：zhao@soho.com | 回复字段。指定应当把回信发到哪里。如果有此字段，回信将会发给它指定的邮箱，而不会发给 From 字段指定的邮箱。比如，发送的是经理的信，但回信应交办公室处理。 |
| 与收信方有关的信头字段 | |
| 格式：TO：mailbox list<br>举例：TO：zhang@263.com | 收信人字段。指定主要收信人的邮箱地址，可以是多个邮箱地址的列表，地址中间用逗号隔开。 |
| 格式：Cc：mailbox list<br>举例：Cc：zhang@863.com | 抄送字段。指定此信件要同时发给哪些人，也称为抄送。也可以使用邮箱地址列表，抄送给多个人。 |
| 格式：Bcc：mailbox list | 密抄字段。指定此信件要同时秘密发给哪些人，也称为密件抄送。也可以使用邮箱地址列表，密抄给多个人。 |
| 其它的信头字段 | |
| 格式：Date：date-time<br>举例：Date：Tue,04 Dec 2004 16:18:08 +800 | 日期字段：Date 字段含有电子邮件创建的日期和时间。 |
| 格式：Subject：*text<br>举例：Subject：Hello!<br>Subject：Re:Hello! | 信件主题字段。描述信件的主题。当回复信件时，通常在主题前面增加“Re:”前缀，标记为该信件为回复信件：当信件被转发时，通常在主题文字前面加上“Fw:”，“Fwd:”这样的前缀。 |
| 格式：Received:<br>["from" domain] ;发送主机<br>["by" domain] ;接收主机<br>["via" atom] ;物理路径<br>["id" msg-id] ;接收者 msg id | 接受字段。是投递信件的特定邮件服务器所作的记录。处理邮件投递的每个服务器必须给它处理的每个信头的前面加一个 Received 字段，用以描述信件到达目的地所经过的路径以及相关信息。当跟踪各个电子邮件问题时，这个信息很有帮助。 |
| 举例：Received:from wang[195.0.0.1] by li[129.5.0.4] Tue dec 2003 12:18:02 +800 | |
| 格式：Comments：*text | 注释字段。用于把一个注解添加到信件中。 |
| 格式：Resent-*<br>举例：Resent-From<br>Resent-Sender<br>Resent-date<br>Resent-Reply-To | 重发字段。当需要把收到的信件重发给另一组收信人的时候，可以保持整个原始信件不变，并简单地产生重发信件所要求的新信头字段。为避免与以前的字段相混。新添加的信头字段都加上 Resent-前缀字符串，它们的语法与未加前缀的同名字段相同。 |
| 格式：Message-ID：msg-id | 信件标识字段。用于表示一个信件唯一标识，该字段通常有 Smtp 服务器生成，这个值通常是唯一的。形式根据使用的软件而定。通常左边是标识符，右边指定电脑名 |

图 2 7 - 2 表中的关键字表明了电子邮件借用了办公室备忘录中的概念和术语：电子邮件的头部能够包含一行说明应当接收到该备忘录的接收方。象传统的办公室备忘录一样，电子邮件使用关键字 Cc 指明一个复写副本(carbon copy).电子邮件软件必须向 Cc:后面的电子邮件地址表中的每个地址发送一份消息的副本。

传统的办公室过程要求备忘录的发送方通知接收方副本是否传给其它人。有时发送方希望将备忘录的一个副本给别人而不显示出有一个副本被发送出去。一些电子邮件系统提供这样的选项，遵循传统的办公室术语，用盲复写副本(blind carbon copy)来表示。创建消息的用户

在关键字 Bcc 后给出一个电子邮件地址表，指定一个或多个盲复写副本。虽然 Bcc 在发送方出现，但当信息发送时，邮件系统将它从消息中除去。每个接收方必须检查头部的 To 和 Cc 行以决定信息是直接发送还是作为盲副本发送的(有些邮件系统在正文部分附加信息来告诉接收者它是一个盲副本)。其它接收者不知道有哪些用户接收到盲副本。

电子邮件使用与传统的办公室备忘录相同的格式和术语：头部包括与消息有关的信息，正文包括消息文本。电子邮件头部的行说明发送方、接收方、日期、主题、应当收到副本的人的列表。

（5）扩展字段

如果想在信头中加入 RFC822 中没有规定的字段，就需要创建非标准字段。方法非常简单，只要在自定义的信头字段名的前面使用 X-前缀。RFC822 将这种方法称为扩展字段。　　事实上已经有许多扩展字段被广泛应用，但没有标准定义。例如：

X-LOOP 字段

X-LOOP 字段用来防止邮件的循环传送。过滤或邮件列表处理程序，可以给它处理的每个信件增加一个 X-LOOP 字段，以后就可以根据这个字段中含有的特别值，判断一个信件是否被循环传送。如果确认邮件发生了循环，过滤或邮件列表处理程序就可以用不同的方式处理该信件。

◆X-Mailer 字段

X-Mailer 字段用于指示什么样的程序产生了这个信件，它是使用最广泛的扩展字段。产生邮件的软件可以为所有发送的信件增加合适的 X-Mailer 字段，该字段不仅含有软件的名称，还包含软件的版本号。例如软件名为 Littlefox Mailer，版本为 V1.0，可以将“X-Mailer:Littlefox Mailer V1.0”加到邮件信头中去。

图 2 7 - 2 列出了一些在因特网电子邮件中可以找到的普通关键字，以及使用它们的目的。

关键字　　含义

From　　发送方地址

To 接收方地址

Cc 复制副本地址

Date 信息创建日期

Subject 信息主题

Reply-To 回复地址

X-Charset 使用的字符集（通常为 ASCII）

X-Mailer 发送信息所使用的软件

X-Sender 发送方地址的副本

X-Face 经编码的发送方面孔的图象

整个系统的核心是收发信件的操作,因此为了方便维护,以后的升级,故将这两个最主要的操作写成类库(.dll)的形式,以组件的形式加载到主程序中，而且其它的功能如果需要的话，也可以通过这样的组件的形式增加到主程序中。这也体现了 C Sharp 这一新的微软主推语言的方便和高校。而且这样做也方便了我们小组的程序的顺利结合。

# 5 命名控件 MailSend

由于在 C Sharp 语言中，都是以命名控件来组织程序的。而所有的类都归属于一个特定的命名空间下。需要的命名空间系统本身自带了一部分，而且如果系统没有你需要的命名空间的话，就可以自己编写，本节中的这个命名空间就是由于需要而编写的。而调用某一个类中的某个变量成员的方法就是通过 命名空间名.类名.变量成员来访问的，当然在 C Sharp 中如果在程序开始通过 Using 命名空间名，就可以直接的象 C++那样来访问成员变量，可以说相当的方便，这些都会在程序中体现出来，再次不再做过多的叙述。

## 5．1 发送邮件类 SmtpMail

### 5．1．1 主要成员变量说明

1) 网络连接类及实例 TcpClient tc

为 TCP 网络服务提供客户端连接类 TcpClient 实例对象 tc。TcpClient 类提供了一些简单的方法，用于在同步阻塞模式下通过网络来连接、发送和接收流数据。而实例化的过程也是连接 SMTP 服务器的过程。它的重载方法之一的两个参数一个为服务器名称字符串，另一个为服务器的埠。

2）提供用于网络访问的基础数据流及其实例 NetworkStream ns

此类提供访问网络的基础数据流的方法。其中最基本也是最重要的两个方法就是 Write（）和 Read（）方法，至于参数不再次赘述。

3）一维字符串数组变量 FilePath

此字符串数组主要用来存放用户选择的附件的绝对路径名，并在发送带附件的邮件时用到。

4）发送邮件所需的基本参数

比如用于 ESMTP 等录检验用的用户名、密码，发送邮件需要的收信人，发信人地址以及主题等等在此不再赘述。

### 5．1．2 主要成员函数说明

1）重载的构造函数 SmtpMail（）

此函数主要用于在初始化过程中，把用户选择的附件的路径以参数的形式传给 FilePath。

2）添加附件的函数 AddAttachment

传给 FilePath 的路径，通过这样一个函数就可以循环的动态的添加到 IList 接口的一个对象中了，方便以后在具体的实现的过程中的使用。

3）得到上传的附件的文件流　GetStream

由于在网络中的操作都是以网络流的形式来实现的，因此先将上传的附件转换成文件流，然后再用 Write 的方法把这些附件的文件流写入到网络中，来完成发送附件的操作。具体实现代码如下所示：

```
private string GetStream(string FilePath)
    {
        //建立文件流对象
    System.IO.FileStream      FileStr=new
System.IO.FileStream(FilePath,System.IO.FileMode.Open);
            byte[] by=new byte[System.Convert.ToInt32(FileStr.Length)];
            FileStr.Read(by,0,by.Length);
            FileStr.Close();
            return(System.Convert.ToBase64String(by));
            }
```

4）将字符串编码为 Base64 字符串的函数　Base64Encode

由于 ESMTP 的 LOGIN 认证机制是采用 Base64 编码，当用户发出 AUTHLOGIN 的命令后，服务器返回 334 的应答码等待用户输入。如果身份确认后服务器返回 235 的应答码，否则返回失败信息。所以要将用户名和密码转换成 Base64 编码然后再发给服务器。此函数的作用就是把给定的字符串转换成相应的 Base64 编码的字符串。

5）发送 SMTP 命令的函数 SendCommand

这个函数的作用是把 SMTP 命令的字符串转换成对应的字节型值（C＃中规定的 Write 方法只能写入字节型的数据）然后写入网络中，如果操作成功就返回一个标志为真的布尔型变量，如果操作失败或者发生异常就返回标志为假的布尔型变量。具体代码如下所示：

```
private bool SendCommand(string str)
            {
                //定义一个数组
                byte[] WriteBuffer;
```

```
            //设定一个布尔类型的变量
            bool state=false;
            WriteBuffer = Encoding.Default.GetBytes(str);
            //加入防错机制，可以有效提高程序运行的效率和捕获出错信息
            try
            {
                //向网络中写入数据
                ns.Write(WriteBuffer,0,WriteBuffer.Length);
                state=true;
            }
            catch(Exception ex)
            {
                //返回出错信息
                MessageBox.Show (ex.ToString ());
                state=false;
            }
        //返回标志位
        return state;
        }
```

6) 接受服务器应答的函数 RecvResponse

它的作用就是从网络流中读取服务器返回的字节型的信息，将其转换成字符串型的变量，然后将其返回，可以通过其返回值来判断操作是否成功。具体实现代码如下所示：

```
private string RecvResponse()
        {
            int StreamSize=0;
            string ReturnValue ="";
            //定义一个字节型的数组
            byte[] ReadBuffer = new byte[1024] ;
            try
```

```
            {
                //从网络流中读取数据，并返回读取的个数
                StreamSize=ns.Read(ReadBuffer,0,ReadBuffer.Length);
            }
            catch (Exception ex)
            {
                //返回异常信息
                MessageBox.Show(ex.ToString ());
            }
            if (StreamSize!=0)
            {
        //将当前读取的信息转换成字符串型然后返回
ReturnValue= Encoding.Default.GetString(ReadBuffer).Substring(0,StreamSize);
            }
            return ReturnValue;
        }
```

7) 重载的函数 Dialog

它们的作用是与服务器交互，发送命令并接收回应。不同的是参数是字符串类型的那个函数，每次发送一条命令，并接受服务器的响应，根据响应的信息来判断交互的结果是否成功。而参数是字符串数组的函数每次发送的是一组命令，用于和服务器的交互，这个函数主要是用于 ESMTP 服务器的验证的功能，因为验证的过程是一个等待然后又输入的过程，因此将他们放在一个数组中有利于理解和操作。而他们的实现主要是通过调用上面的发送 SMTP 命令函数 SendCommand 以及接受 SMTP 服务器响应的函数 RecvResponse 来实现的。具体的代码如下所示：

```
private bool Dialog(string str,string errstr)
        {
            bool flag=false;
            if(str==null||str.Trim()=="")
            {
                flag=true;
```

```
            }
            if(SendCommand(str))
            {
                string RR=RecvResponse();
                //从返回的数据中截取前三位
                string RRCode=RR.Substring(0,3);
                //然后用这前三位与哈希表中正确的回应码比较
                if(RightCodeHT[RRCode]!=null)
                {
                    flag=true;
                }
                else
                {
                    flag=false;
                }
            }
            else
            {
                flag=false;
            }
            return flag;
        }
```

发送一组命令主要用于服务器验证的重载函数为：

```
private bool Dialog(string[] str,string errstr)
        {
            for(int i=0;i<str.Length;i++)
            {
                //循环调用单个的与服务器的交互过程
                if(!Dialog(str[i],""))
                {
```

```
                return false;
            }
        }
        return true;
    }
```

8) 邮件发送程序 SendMail

这是整个程序的核心部分。具体的实现 SMTP 协议的程序正是通过它一步一步实现并最终实现发送简单邮件甚至带附件的邮件的功能。而它的实现是调用以上给出的各个函数的结果。以下就简单的通过几个 SMTP 命令的格式来实现

```
private bool SendEmail()
    {
        //连接网络
        try
        {
            //建立一个 TCP 连接
            tc=new TcpClient(mailserver,mailserverport);
        }
        catch
        {
            MessageBox.Show ("连接失败","请确认");
            return false;
        }
        //获取当前流的资料
        ns = tc.GetStream();
        SMTPCodeAdd();
        //验证网络连接是否正确
        if(RightCodeHT[RecvResponse().Substring(0,3)]==null)
        {
            return false;
        }
```

```
string[] SendBuffer;
string SendBufferstr;
//进行 SMTP 验证
//具体的 SMTP 命令与代码的结合
if(ESmtp)
{
    SendBuffer=new String[4];
    SendBuffer[0]="EHLO " + mailserver + enter;
    SendBuffer[1]="AUTH LOGIN" + enter;
    SendBuffer[2]=Base64Encode(username) + enter;
    SendBuffer[3]=Base64Encode(password) + enter;
if(!Dialog(SendBuffer,"SMTP 服务器验证失败，请核对用户名和密码。"))
        return false;
}
else
{
    SendBufferstr="HELO " + mailserver + enter;
    if(!Dialog(SendBufferstr,""))
        return false;
}
SendBufferstr="MAIL FROM:<" + From + ">" + enter;
if(!Dialog(SendBufferstr,"发件人地址错误，或不能为空"))
    return false;
//把传过来的收件人的地址分割然后提交给服务器
string split=";";
string []address=Regex.Split (Recipient,split);
SendBuffer=new string [address.Length];
for(int i=0;i<SendBuffer.Length;i++)
{
    SendBuffer[i]="RCPT TO:<" +address[i]+">" + enter;
```

```
                }
                if(!Dialog(SendBuffer,"收件人地址有误"))
                    return false;
                SendBufferstr="DATA" + enter;
                if(!Dialog(SendBufferstr,""))
                    return false;
                SendBufferstr="From:" + FromName + "<" + From +">" +enter;
    SendBufferstr += enter + "." + enter;
                if(!Dialog(SendBufferstr,"错误信件信息"))
                    return false;
                SendBufferstr="QUIT" + enter;
                if(!Dialog(SendBufferstr,"断开连接时错误"))
                    return false;
                //关闭流对象
                ns.Close();
                //关闭连接
                tc.Close();
                FilePath=null;
                return true;
            }
```

以上即为发送不带附件的邮件 SMTP 命令用代码实现的过程。

## 5．2　AddExtra 类

这个附加的小类只是提供一些返回当前系统时间，获取主机名，主机 IP，有关帮助等小的功能，在此仅对帮助信息中的“关于”操作函数稍加说明。因为它说明了在 C Sharp 中调用 Windows API 函数所需如下几个步骤：

### 5．2．1　调用 Windows API 所需的命名空间

----using System.Runtime.InteropServices;

而调用显示关于对话框的函数 ShellAbout 还需要用到两个命名空间如下所示

---using System.Reflection;

---using System.Diagnostics ;

### 5．2．2　在程序中声明所需的 API 函数

```
[DllImport("shell32.dll")]
    static extern int ShellAbout(IntPtr hWnd, string szApp, string szOtherStuff,
IntPtr hIcon);
```

### 5．2．3　在程序中具体的使用

```
    Assembly ass=Assembly.GetExecutingAssembly();
    FileVersionInfo myVersion=FileVersionInfo.GetVersionInfo(ass.Location );
    ShellAbout(this.Handle  ," 邮 件 收 发 系 统 #"," 版 本 "+myVersion.FileMajorPart
+"."+myVersion.FileMinorPart+"." +myVersion.CompanyName ,this.Icon .Handle );
```

至此就完成了在 C Sharp 中调用 Windows API 函数的过程。

# 6　软件运行时的界面

## 6．1　新建邮件帐号

用户打开软件之后，需要新建一个邮件帐号，在这个信件帐号的过程中，需要指定 SMTP 服务器，SMTP 的端口，以及用于 ESMTP 验证的用户名和密码。指定这些发邮件的必须参数之后，再回到系统的主界面如下所示：

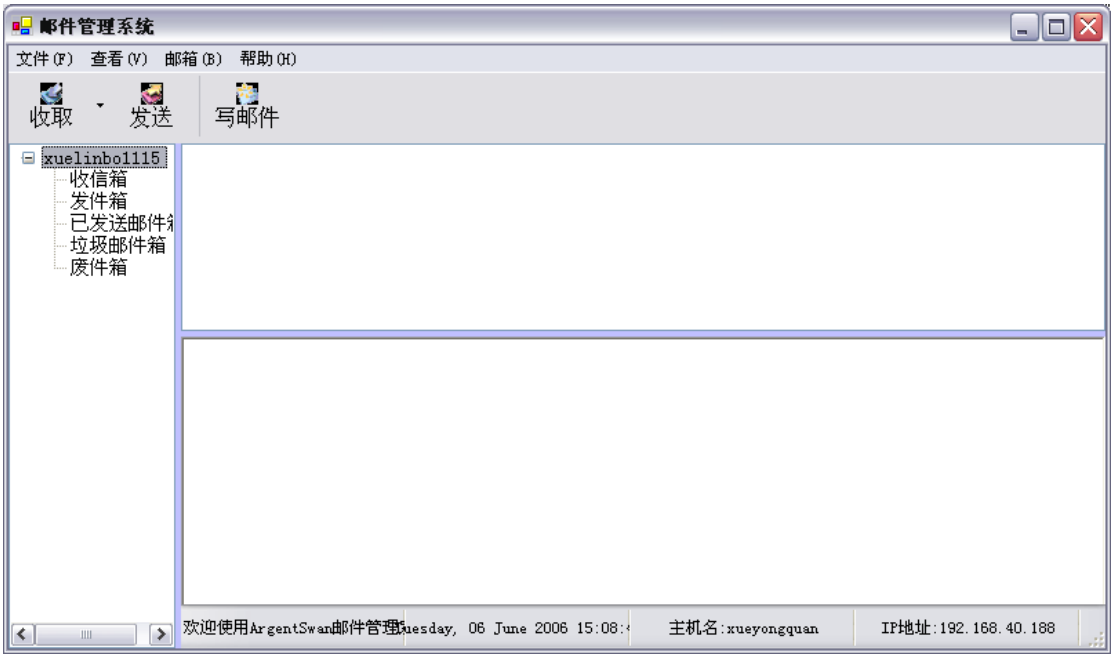



图 6.1　新建邮件帐号界面

## 6．2　发送邮件界面

### 6．2．1　发送不带附件的邮件

在新建帐号的过程中已经指定了邮件地址，和帐号名称，所以默认的以这些参数来发送邮件。通过调用参数的不同程序会自动的调用相对应的代码来执行不同的操作。发送简单的邮件运行界面如下。

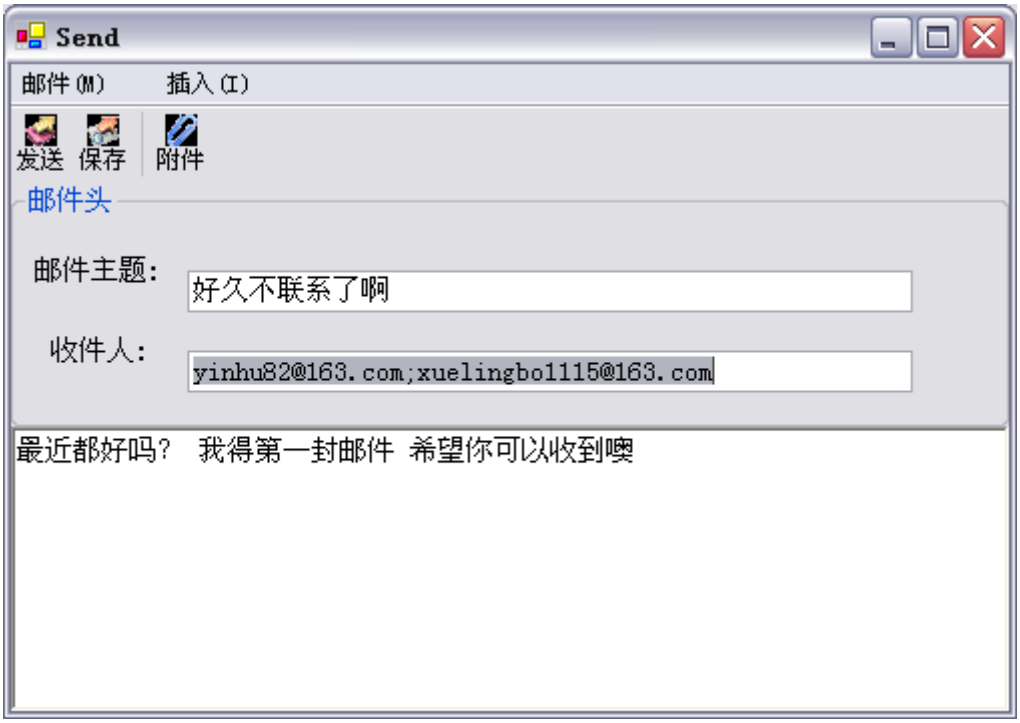



图 6.2 发送纯文本邮件

### 6．2．2 发送带附件的邮件

和简单的邮件不同之处在于多了发送附件的功能，软件模拟 FoxMail 里面发送邮件时，在程序的下面自动显示增添的附件的名称，以及图标等信息。并且邮件支持添加，删除，排列图标等功能。 运行界面如下所示：

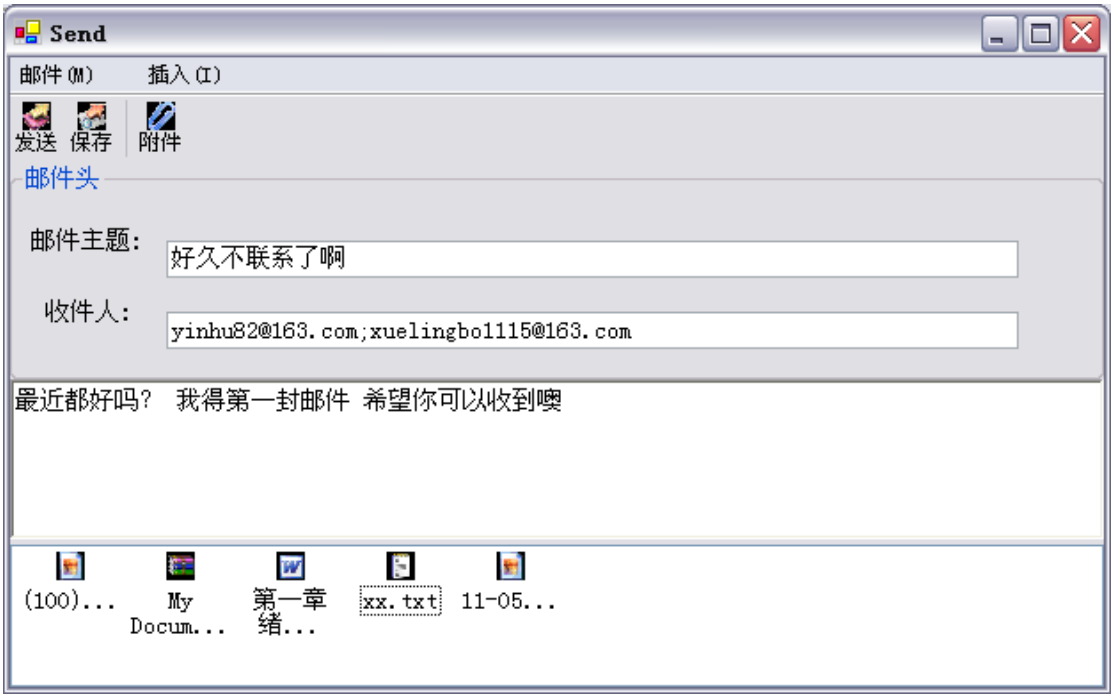



图 6.3 发送带有附件的邮件

## 6．3　验证邮件发送是否成功

邮件发送出去之后，用 FoxMail 跟踪接收之后，证明邮件和附件都可以正常接收，具体的 FoxMail 的接收界面如下所示：


图 6.4　验证发送的邮件是否成功

# 7 系统测试

我个人做的是这个软件收发系统的一个最基本也是最主要的功能之一：发送邮件。

所以主要的测试也是围绕发送邮件展开的，具体的可以分为以下几个方面。

## 7．1 同一 SMTP 服务器发送邮件的测试

这个方面的测试测的是，用户登录一个服务器（测试中用的是 163 的 SMTP 服务器）来发送一封邮件的测试。而这个测试又可以分为以下两个方面：

### 7．1．1 同一服务器，发送一封纯文本邮件的测试

1） 发送一封文本邮件给一个收信人

测试中用 163 的邮箱分别往 163 的邮箱以及新浪的邮箱发送邮件均可以用 FoxMail 正常的接收到发送的普通的纯文本文件。

2） 发送一封文本邮件给多个收件人

测试中仍然用 163 的邮箱同时发往不同的邮箱，通过 FoxMail 都可以正常的接收到。从而很好的验证了，我们的邮件发送系统支持群发的功能。

### 7．1．2 同一服务器，发送一封带附件的邮件的测试。

1） 发送一封带附件（可以是多附件）的邮件给一个收件人

测试中用 163 的邮箱分别往 163 的邮箱以及新浪的邮箱发送之外，又添加了不同的邮件类型（个数分别为等于 1，大于 1 即验证是否支持多附件的发送），用 FoxMail 接收之后，所有发送的纯文本信息，以及附件信息都正常无误。经过这些验证可以证明本软件支持对一个收件人发送多附件。由于带有多附件的信件，所以写入速度明显慢于纯文本邮件的速度。

2）发送一封带附件（可以是多附件）的邮件给多个收件人

测试中用 163 的邮箱分别往 163 的邮箱以及新浪的邮箱发送之外，又添加了不同的邮件类型（个数分别为等于 1，大于 1 即验证是否支持多附件的发送），用 FoxMail 接收之后，所有发送的纯文本信息，以及附件信息都正常无误。经过这些验证可以证明本软件支持对多个收件人发送多附件。

## 7．2 利用不同的 SMTP 服务器发送邮件的测试

这个方面的测试是指利用不同的邮箱来发送邮件，至于测试的分类雷同于利用同一服务器发送邮件的测试，所以不再此赘述。

总之，通过以上的各方面的测试，使我改正了代码中的许多不合理以及错误之处，最终也证明了，我们的软件系统是支持多种服务器，支持多附件发送的群发软件。

# 8 结论

这次编写的邮件客户端系统，我负责的是邮件的发送的功能。在熟悉了专门用于发送邮件的 SMTP 协议以及 RFC 规定的邮件的格式的基础上，运用了微软新推出的 C Sharp 这一新型的面向对象语言的便利性和灵活性，从 SMTP 协议规定的底层命令做起，一步步的与服务器进行交互操作，最终实现发送多附件多接收人的功能。其中，具体的和服务器的交互操作，都封装了在 SmtpMail.dll 这个动态链接库里面了。而为了方便最终的调用和整合，所有的有关后台操作发送邮件的类以及其他的附加功能的类，全部都归属于 MailSend 这个命名空间了。在力求达到 FoxMail 功能的同时，又加了一点个人的思想并把它体现到了这一软件上。最主要的体现就是新建帐号的提前检测这一特色上，这一功能类似于很多 Web 页面的“检测新帐号”的功能，这样就免去了用户一直到确定注册完成时，才因为帐户因为已经被使用而注册失败的麻烦。总之，通过这次的编程，使我对网络编程有了一个很好的认识和锻炼，也使我对 C Sharp 这一语言的掌握程度又上了一个新台阶，虽然编出来的软件不能和功能强大的 FoxMail 相提并论，但是相信它简单，易操作性，和 FoxMail 的很多强大但却“鸡肋”似的功能比较起来，更多了几分实用性。以后的日子，随着我技术的提高和思想的成熟，我一定会把它做的更好，更趋近于完美。

# 参考文献

[1] Simon Robinson, K.Scott Allen 等.C#高级编程. 北京:清华大学出版社, 2002,3

[2] Tom Archer. C#技术内幕. 北京:清华大学出版社, 2002,1

[3]沉舟.Microsoft.NET 编程语言 C#. 北京:希望电子出版社 2001,3

[4] 罗军舟,黎波涛,杨明等.TCP/IP 协议及网络编程技术. 北京: 清华大学出版 2004,10

[5] Tim Parker .TCP/IP 协议及网络编程技术. 北京: 机械工业出版社, 2000,7

[6] 周存杰 . Visual C#.NET 网络核心编程. 北京:清华大学出版社, 2002,11

[7] 电脑编程技巧与维护杂志社.C#编程技巧典型案例解析. 北京:中国电力出版社, 2005,8

[8] 云颠工作室. Visual C#中文版全面剖析. 北京:中国水利水电出版社, 2003,5

[9] 叶树华 《电子协议与编程》,《电子邮件格式》,《电子邮件接收》,《mime 编码解码与发送附件》

[10] MSDN 中文网站网络广播 C#设计模式纵谈 http://www.microsoft.com/china/msdn/events/webcasts/shared/Webcast/MSDNWebCast.aspx

[11] 滁州,马金虎,朱力勇. 编写基于 SMTP 网络应用程序. 电脑爱好者,2003,5:92~94

[12] 滁州,马金虎,朱力勇. 编写基于 POP3 网络应用程序. 电脑爱好者,2003,6:92~94

[13] 潘泰国. 新一代电子邮件系统. 电子技术应用. 1992,11

[14] 代继红. SMTP 认证机制模块化设计及实现. 中南民族大学学报(自然科学版), 2005,4

[15] 胡安廷. 简单实现中文邮件. 中国计算机报, 2004,11

# 致 谢

感谢我的父母和亲人，没有您们的包容和支持，就不会有我的今天。

感谢我的导师叶树华老师，为我们提供丰富的材料，指导我们完成毕业设计。

感谢我的班主任李玲老师，无论在学习上还是生活上，都给予我莫大鼓励和帮助。

感谢张建华老师，给我讲解了许多 C Sharp 的知识，使我获益匪浅。

感谢 MSDN 中文网站的全体员工们，使我可以免费获取更多的知识。

感谢 CSDN 技术论坛的人们，帮我解决了很多技术性的难题。

感谢 412 的全体室友们，我们一起走过的日子，我感觉充实又快乐。

感谢南京神州数码公司，虽然面试我没有通过，但是你们使我明白现实世界并不是如我想象的那么美好，使我看到自己的不足，也使我积累到了经验。

感谢篮球场上和我一起挥汗如雨的哥们们，无论我身在何方，我都会记得曾经一起并肩作战的你们。

感谢 NBA 中一样为了生存和梦想而奋斗者的球员们，为我带来精彩的比赛，激励着我追寻自己的梦想。

感谢我在 xxx 的四年里，所经历的一切，无论我走到哪里，我都会记得这是我生命中不可或缺的记忆。

# 外文文献原文

## SMTP Service Extension for Authentication

This document specifies an Internet standards track protocol for the Internet community, and requests discussion and suggestions for improvements. Please refer to the current edition of the "Internet Official Protocol Standards" (STD 1) for the standardization state and status of this protocol. Distribution of this memo is unlimited.



1. Introduction

This document defines an SMTP service extension [ESMTP] whereby an SMTP client may indicate an authentication mechanism to the server,perform an authentication protocol exchange, and optionally negotiatea security layer for subsequent protocol interactions. This extension is a profile of the Simple Authentication and Security Layer [SASL].

2. Conventions Used in this Document

In examples, "C:" and "S:" indicate lines sent by the client and server respectively. The key words "MUST", "MUST NOT", "SHOULD", "SHOULD NOT", and "MAY" in this document are to be interpreted as defined in "Key words for use in RFCs to Indicate Requirement Levels" [KEYWORDS].

3. The Authentication service extension

(1) the name of the SMTP service extension is "Authentication"

(2) the EHLO keyword value associated with this extension is "AUTH"

(3) The AUTH EHLO keyword contains as a parameter a space separated list of the names of supported SASL mechanisms.

(4) a new SMTP verb "AUTH" is defined

(5) an optional parameter using the keyword "AUTH" is added to the MAIL FROM command, and extends the maximum line length of the MAIL FROM command by 500 characters.

(6) this extension is appropriate for the submission protocol [SUBMIT].

4. The AUTH command AUTH mechanism [initial-response]

Arguments:

a string identifying a SASL authentication mechanism. an optional base64-encoded response

Restrictions:

After an AUTH command has successfully completed, no more AUTH commands may be issued in the same session. After a successful AUTH command completes, a server MUST reject any further AUTH commands with a 503 reply. The AUTH command is not permitted during a mail transaction.

Discussion:

The AUTH command indicates an authentication mechanism to the server. If the server supports the requested authentication mechanism, it performs an authentication protocol exchange to authenticate and identify the user. Optionally, it also negotiates a security layer for subsequent protocol interactions. If the requested authentication mechanism is not supported, the server rejects the AUTH command with a 504 reply.

The authentication protocol exchange consists of a series of server challenges and client answers that are specific to the authentication mechanism. A server challenge, otherwise known as a ready response, is a 334 reply with the text part containing a BASE64 encoded string. The client answer consists of a line containing a BASE64 encoded string. If the client wishes to cancel an authentication exchange, it issues a line with a single "*". If the server receives such an answer, it MUST reject the AUTH command by sending a 501 reply.

The optional initial-response argument to the AUTH command is used to save a round trip when using authentication mechanisms that are defined to send no data in the initial challenge.

When the initial-response argument is used with such a mechanism, the initial empty challenge is not sent to the client and the server uses the data in the initial-response argument as if it were sent in response to the empty challenge. Unlike a zero-length client answer to a 334 reply, a zero- length initial response is sent as a single equals sign ("="). If the client uses an initial-response argument to the AUTH command with a mechanism that sends data in the initial challenge, the server rejects the AUTH command with a 535 reply.

If the server cannot BASE64 decode the argument, it rejects the AUTH command with

a 501 reply. If the server rejects the authentication data, it SHOULD reject the AUTH command with a 535 reply unless a more specific error code, such as one listed in section 6, is appropriate. Should the client successfully complete the authentication exchange, the SMTP server issues a 235 reply.

The service name specified by this protocol's profile of SASL is "smtp".

If a security layer is negotiated through the SASL authentication exchange, it takes effect immediately following the CRLF that concludes the authentication exchange for the client, and the CRLF of the success reply for the server. Upon a security layer's taking effect, the SMTP protocol is reset to the initial state (the state in SMTP after a server issues a 220 service ready greeting). The server MUST discard any knowledge obtained from the client, such as the argument to the EHLO command, which was not obtained from the SASL negotiation itself. The client MUST discard any knowledge obtained from the server, such as the list of SMTP service extensions, which was not obtained from the SASL negotiation itself (with the exception that a client MAY compare the list of advertised SASL mechanisms before and after authentication in order to detect an active down-negotiation attack). The client SHOULD send an EHLO command as the first command after a successful SASL negotiation which results in the enabling of a security layer.

The server is not required to support any particular authentication mechanism, nor are authentication mechanisms required to support any security layers. If an AUTH command fails, the client may try another authentication mechanism by issuing another AUTH command.

If an AUTH command fails, the server MUST behave the same as if the client had not issued the AUTH command.

The BASE64 string may in general be arbitrarily long. Clients and servers MUST be able to support challenges and responses that are as long as are generated by the authentication mechanisms they support, independent of any line length limitations the client or server may have in other parts of its protocol implementation.

Examples:

S: 220 smtp.example.com ESMTP server ready

C: EHLO jgm.example.com

S: 250-smtp.example.com

S: 250 AUTH CRAM-MD5 DIGEST-MD5

C: AUTH FOOBAR

S: 504 Unrecognized authentication type.

C: AUTH CRAM-MD5

S: 334

PENCeUxFREJoU0NnbmhNWitOMjNGNndAZWx3b29kLmlubm9zb2Z0LmNvbT4

=

C: ZnJlZCA5ZTk1YWVlMDljNDBhZjJiODRhMGMyYjNiYmFlNzg2ZQ==

S: 235 Authentication successful.

5. The AUTH parameter to the MAIL FROM command

AUTH=addr-spec

Arguments:

An addr-spec containing the identity which submitted the message to the delivery system, or the two character sequence "<>" indicating such an identity is unknown or insufficiently authenticated. To comply with the restrictions imposed on ESMTP parameters, the addr-spec is encoded inside an xtext. The syntax of an xtext is described in section 5 of [ESMTP-DSN].

Discussion:

The optional AUTH parameter to the MAIL FROM command allows cooperating agents in a trusted environment to communicate the authentication of individual messages.

If the server trusts the authenticated identity of the client to

assert that the message was originally submitted by the supplied addr-spec, then the server SHOULD supply the same addr-spec in an AUTH parameter when relaying the

message to any server which supports the AUTH extension.

A MAIL FROM parameter of AUTH=<> indicates that the original submitter of the message is not known. The server MUST NOT treat the message as having been originally submitted by the client.

If the AUTH parameter to the MAIL FROM is not supplied, the client has authenticated, and the server believes the message is an original submission by the client, the server MAY supply the client's identity in the addr-spec in an AUTH parameter when relaying the message to any server which supports the AUTH extension.

If the server does not sufficiently trust the authenticated identity of the client, or if the client is not authenticated, then the server MUST behave as if the AUTH=<> parameter was supplied. The server MAY, however, write the value of the AUTH parameter to a log file.

If an AUTH=<> parameter was supplied, either explicitly or due to the requirement in the previous paragraph, then the server MUST supply the AUTH=<> parameter when relaying the message to any server which it has authenticated to using the AUTH extension.

A server MAY treat expansion of a mailing list as a new submission, setting the AUTH parameter to the mailing list address or mailing list administration address when relaying the message to list subscribers.

It is conforming for an implementation to be hard-coded to treat all clients as being insufficiently trusted. In that case, the implementation does nothing more than parse and discard syntactically valid AUTH parameters to the MAIL FROM command and supply AUTH=<> parameters to any servers to which it authenticates using the AUTH extension.

Examples:

C: MAIL FROM:<e=mc2@example.com> AUTH=e+3Dmc2@example.com

S: 250 OK

6. Error Codes

The following error codes may be used to indicate various conditions as described.

432 A password transition is needed

This response to the AUTH command indicates that the user needs to transition to the

selected authentication mechanism. This typically done by authenticating once using the PLAIN authentication mechanism.

534 Authentication mechanism is too weak

This response to the AUTH command indicates that the selected authentication mechanism is weaker than server policy permits for that user.

538 Encryption required for requested authentication mechanism

This response to the AUTH command indicates that the selected authentication mechanism may only be used when the underlying SMTP connection is encrypted.

454 Temporary authentication failure

This response to the AUTH command indicates that the authentication failed due to a temporary server failure.

530 Authentication required

This response may be returned by any command other than AUTH, EHLO, HELO, NOOP, RSET, or QUIT. It indicates that server policy requires authentication in order to perform the requested action.

7. Formal Syntax

The following syntax specification uses the augmented Backus-Naur Form (BNF) notation as specified in [ABNF].

Except as noted otherwise, all alphabetic characters are case- insensitive. The use of upper or lower case characters to define token strings is for editorial clarity only. Implementations MUST accept these strings in a case-insensitive fashion.

UPALPHA = %x41-5A ;; Uppercase: A-Z

LOALPHA = %x61-7A ;; Lowercase: a-z

ALPHA = UPALPHA / LOALPHA ;; case insensitive

DIGIT = %x30-39 ;; Digits 0-9

HEXDIGIT = %x41-46 / DIGIT ;; hexidecimal digit (uppercase)

hexchar = "+" HEXDIGIT HEXDIGIT

xchar = %x21-2A / %x2C-3C / %x3E-7E;; US-ASCII except for "+", "=", SPACE and CTL

xtext = *(xchar / hexchar)

AUTH_CHAR = ALPHA / DIGIT / "-" / "_"

auth_type = 1*20AUTH_CHAR

auth_command = "AUTH" SPACE auth_type [SPACE (base64 / "=")]*(CRLF [base64]) CRLF

auth_param = "AUTH=" xtext;; The decoded form of the xtext MUST be either;; an addr-spec or the two characters "<>"

base64 = base64_terminal /( 1*(4base64_CHAR) [base64_terminal] )

base64_char = UPALPHA / LOALPHA / DIGIT / "+" / "/";; Case-sensitive

base64_terminal = (2base64_char "==") / (3base64_char "=")

continue_req = "334" SPACE [base64] CRLF

CR = %x0C ;; ASCII CR, carriage return

CRLF = CR LF

CTL = %x00-1F / %x7F ;; any ASCII control character and DEL

LF = %x0A ;; ASCII LF, line feed

SPACE = %x20 ;; ASCII SP, space

8. References

[ABNF] Crocker, D. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", RFC2234, November 1997.

[CRAM-MD5] Klensin, J., Catoe, R. and P. Krumviede, "IMAP/POP AUTHorize Extension for Simple Challenge/Response", RFC 2195, September 1997.

[ESMTP] Klensin, J., Freed, N., Rose, M., Stefferud, E. and D. Crocker, "SMTP Service Extensions", RFC1869, November 1995.

[ESMTP-DSN] Moore, K, "SMTP Service Extension for Delivery Status Notifications", RFC1891, January 1996.

[KEYWORDS] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC2119, March 1997

[SASL] Myers, J., "Simple Authentication and Security Layer (SASL)", RFC2222, October 1997.

[SUBMIT] Gellens, R. and J. Klensin, "Message Submission", RFC 2476, December 1998.

[RFC821] Postel, J., "Simple Mail Transfer Protocol", STD 10, RFC 821, August 1982.

[RFC822] Crocker, D., "Standard for the Format of ARPA Internet Text Messages", STD 11, RFC822, August 1982.

9. Security Considerations

Security issues are discussed throughout this memo.

If a client uses this extension to get an encrypted tunnel through an insecure network to a cooperating server, it needs to be configured to never send mail to that server when the connection is not mutually authenticated and encrypted. Otherwise, an attacker could steal the client's mail by hijacking the SMTP connection and either pretending the server does not support the Authentication extension or causing all AUTH commands to fail.

Before the SASL negotiation has begun, any protocol interactions are performed in the clear and may be modified by an active attacker. For this reason, clients and servers MUST discard any knowledge obtained prior to the start of the SASL negotiation upon completion of a SASL negotiation which results in a security layer.

This mechanism does not protect the TCP port, so an active attacker may redirect a relay connection attempt to the submission port [SUBMIT]. The AUTH=<> parameter prevents such an attack from causing an relayed message without an envelope authentication to pick up the authentication of the relay client.

A message submission client may require the user to authenticate whenever a suitable SASL mechanism is advertised. Therefore, it may not be desirable for a submission server [SUBMIT] to advertise a SASL mechanism when use of that mechanism grants the client no benefits over anonymous submission.

This extension is not intended to replace or be used instead of end- to-end message signature and encryption systems such as S/MIME or PGP. This extension addresses a different problem than end-to-end systems; it has the following key differences:

(1) it is generally useful only within a trusted enclave

(2) it protects the entire envelope of a message, not just the message's body.

(3) it authenticates the message submission, not authorship of the message content

(4) it can give the sender some assurance the message was delivered to the next hop in the case where the sender mutually authenticates with the next hop and negotiates an appropriate security layer.

Additional security considerations are mentioned in the SASL specification [SASL].

# 译文

## SMTP 服务扩展的认证机制

这个文档详细说明了因特网团体的一个标准的协议的发展，以及对其改进和建议提出了要求。说到这，为了标准化这个协议的状态和地位，就必须提及目前最新的“Internet 官方协议的标准”(STD1)。发送这个文档是不受限制的。

版权须知



1 简介

这个文档定义了 SMTP 服务的扩展（ESMTP）并且说明了一个 SMTP 客户端可以为服务器指定一种用来执行与认证协议的交换，并且随意地穿越并发的协议之间交互的安全层的认证机制。这个扩展是“简单认证和安全层”[SASL]的一个侧面。

2 这个文档用到的协定

在以下的这些例子中，C 和 S 分别表示客户端和服务器。

诸如 MUST", "MUST NOT", "SHOULD", "SHOULD NOT", and "MAY"这些关键性的单词被可以看作和“用在 RFC 文档中用来标示必须的级别的关键字”[KEYWORDS]相同的解释。

3 认证服务的扩展

SMTP 服务扩展的名称是 "Authentication"

联合这个扩展的 EHLO 关键字的值是“AUTH“

(3)AUTH EHLO 关键字 是一个有空格间隔的被 SASL 机制支持的名字列表的参数

(4)一个新的 SMTP 动词“AUTH“定义完成。

(5)用在关键字“AUTH“的一个可选的参数被附加到 MAIL FROM 命令里，用来指定 MAIL FROM 命令一行的最大长度不能超过 500 个字符。

(6) 此扩展和委托协议兼容。

4 AUTH 命令

AUTH 机制[初始化响应]

观点：

用来标识 SASL 认证机制的一个字符串

可选的 Base64 编码的一个响应

约束：

再成功发出了一个 AUTH 命令之后，在同一时间段里不能再执行其他的 AUTH 命令。在成功执行了一个 AUTH 命令之后，服务器必须拒绝后来的 AUTH 命令并且返回一个 503 响应码。

在处理一个邮件事务期间，服务器不会再接受 AUTH 命令。

讨论：

AUTH 命令显示了一种和邮件服务器间的安全认证机制 。如果邮件服务器支持这种认证机制，它就会执行一个认证协议交互来认证并识别邮件用户。作为可选的情况，他也会忽略这以后后协议交互的一个安全层。如果服务器并不支持所需要的认证协议，就会用 504 的回答来拒绝这个 AUTH 命令。

这种认证机制的交互由一些列的服务器的响应和对认证机制来说的一些特殊的回答来组成。服务器的正确响应，不同于其他的响应的是针对文本部分采用 Base64 编码以 334 做为回应的。客户端的回应是一个包含 Base64 编码的字符串的队列。如果客户端想取消与服务器的认证交互，就执行一个单个的“*”。如果服务器接到这样一个回应，就通过发送一个 501 的响应来拒绝执行 AUTH 命令。

对 AUTH 命令来说，可选的初始化响应建议是用来在使用认证机制时保持一个往

返的回程,认证机制的定义中此建议不发送任何数据。当初始化响应部分用在这种机制时，

开始的空的发起命令不被送到客户端，并且服务器端使用的数据也好象是发送来

响应一个空的命令。它发送一个零长度的初始化回答作为一个"="符号。如果客户端

在认证机制的 AUTH 命令响应中使用初始化建议，客户端就在初始化命令中发送响应的

数据，服务器端用 535 回答来拒绝 AUTH 命令。

如果服务器不能对发送来的命令采用 Base64 解码的话，将拒绝执行 Auth 命令，并返回 501 响应。如果服务器拒绝认证的数据，服务器应该拒绝执行并返回一个 535 响应码除非有更详细的错误代码，例如在 Section 6 列出来的那个。如果客户端和服务器进行了正确的交互的操作的话，SMTP 服务器将发出一个 235 响应码。

详细说明这个 SASL 侧面的服务器的名称是”SMTP“。

如果 SASL 认证交互穿越了一个安全层，将会通过一个有用来中止认证交互的 CRLF 来产生效果，而服务器也通过一个 CRLF 做出正确的响应。在服务器的安全层生效之前，SMTP 协议被重置到初始状态（SMTP 中的状态是服务器发出了一个 220 服务的问候之后）。服务器 MUST 命令将抛弃所有的不是通过客户端而得到的认知，比如不是通过 SASL 本身而获得认知的 EHLO 命令的论点。客户端的 MUST 命令将抛弃所有的从服务器获得的认知，例如不是通过 SASL 本身而获得的 SMTP 服务扩展的队列。客户端的 SHOULD 在 SASL 商议成功之后，发出一个 EHLO 命令做为第一个命令，这些将使得安全层得到授权。

服务器不一定要求支持任何的认证机制，而认证机制也不一定要支持所有的安全层。如果一个 AUTH 命令失败了，客户端将试图执行另一个认证机制的 AUTH 命令。

一个 Base64 编码的字符串通常来说是没有长度限制的。只要由认证机制产生的受客户端和服务器支持的命令和响应，客户端和服务器端必须支持，而不依赖于服务器或者客户端的、可能存在于协议实现的某些方面的行长度的限制。

例如：

Examples:

S: 220 smtp.example.com ESMTP server ready

C: EHLO jgm.example.com

S: 250-smtp.example.com

S: 250 AUTH CRAM-MD5 DIGEST-MD5

C: AUTH FOOBAR

S: 504 Unrecognized authentication type.

C: AUTH CRAM-MD5

S: 334

PENCeUxFREJoU0NnbmhNWitOMjNGNndAZWx3b29kLmlubm9zb2Z0LmNvbT4

=

C: ZnJlZCA5ZTk1YWVlMDljNDBhZjJiODRhMGMyYjNiYmFlNzg2ZQ==

S: 235 Authentication successful.

5. AUTH 命令的参数附加到的 MAIL FROM 命令

AUTH=addr-spec

参数：

一个包含标志的被提交给传送系统的 addr-spec，或者是两个字符组成的序列"<>" ,

表明这个标志是未知的或被验明为不完成的。

讨论：

AUTH 中一个可选的参数的 MAIL FROM 命令允许一个协同工作的代理与一单独的消息就行通信在一个被信任的环境里。

如果服务器认为最初提交消息的 Addr－dec 的客户端是可信任的话，将会发出一个声明，接着服务器应当提供一个相同的 addr－dec 给任何其他支持 AUTH 扩展的用来中转消息的服务器。

如果 MAIL FROM 命令中那个可选的 AUTH 命令的参数没有得到提供的话，而客户端已经得到认证，那么服务器认为消息是由客户端提交的原始的信息，那么在中转给其他的中继服务器的时候，当前服务器就会把 addr－dec 做为 Auth 命令的可选的参数提供给其它的服务器。

如果服务器不是充分的相信客户端的身份或者客户端并没有得到认证的话，那么服务器必须自己提供 AUTH 命令的那个参数一个值。并且将这个值写入到日志文件中。

如果 AUTH 命令的可选的参数已经提供了的话，不管是明确的提出还是由于前面段落的需要，服务器应当提供这个参数给任何其他支持 AUTH 扩展的用来中转消息的服务器。

服务器将把邮件列表的扩充视为一个新的任务，AUTH 命令加入到邮件地址列表中，或者在中转这些消息到列表签署者的时候管理邮件列表。

为了一致，在一个执行很难被编码的时候，服务器将认为所有的这些客户端都是不可信任的。在这种情况下，服务器能做的仅仅就是解析有效的 AUTH 命令的参数，并把它提供给任所有使用 AUTH 扩展的认证机制的服务器，并遗弃无效的参数。

例如：

C: MAIL FROM:<e=mc2@example.com> AUTH=e+3Dmc2@example.com

S: 250 OK

6　错误代码

以下的错误代码常常用来描述和标识各种情况。

432　需要进行密码的转换

这个响应码表示，对于服务器的认证机制来讲，用户必须进行一个转换。比较由代表性的就是一旦你使用了 PLAIN 认证机制的话，就必须进行转换。

534　认证机过于简单

这个响应码表示的是选择的认证机制相对于服务器所允许的认证机制来说显得太弱了。

538　请求的认证机制需要加密

这个响应码表示的是所选的认证机制只有在 SMTP 连接是需要加密的情况下才用的着

454　暂时的认证失败

这个响应码表示的是认证失败的原因是由于服务器暂时出现问题

530　认证是必须的

除了 AUTH，EHLO，HELO，NOOP，RESET 或者 QUIT 这几个命令之外的任何一个命令，都将返回这个响应码。这表示服务器需要为了执行被请求的操作，需要一个认证。

7　正规的语法

以下的用在扩展的 BNF 符号和用在 ABNF 中的语法的规格是一样的。

除了那些被标注的以外，所有的按字母顺序排列的特征都是适合于固定场合的。排在上面的或者下面的被用来定义为有象征意义的字符串的用处仅仅是为了编辑时的便利以及清晰。执行这些必须在以固定的格式在一定的场合来接受这些字符串。

UPALPHA = %x41-5A ;; Uppercase: A-Z

LOALPHA = %x61-7A ;; Lowercase: a-z

ALPHA = UPALPHA / LOALPHA ;; case insensitive

DIGIT = %x30-39 ;; Digits 0-9

HEXDIGIT = %x41-46 / DIGIT ;; hexidecimal digit (uppercase)

hexchar = "+" HEXDIGIT HEXDIGIT

xchar = %x21-2A / %x2C-3C / %x3E-7E

;; US-ASCII except for "+", "=", SPACE and CTL

xtext = *(xchar / hexchar)

AUTH_CHAR = ALPHA / DIGIT / "-" / "_"

auth_type = 1*20AUTH_CHAR

auth_command = "AUTH" SPACE auth_type [SPACE (base64 / "=")]

*(CRLF [base64]) CRLF

auth_param = "AUTH=" xtext

;; The decoded form of the xtext MUST be either

;; an addr-spec or the two characters "<>"

base64 = base64_terminal /

( 1*(4base64_CHAR) [base64_terminal] )

base64_char = UPALPHA / LOALPHA / DIGIT / "+" / "/"

;; Case-sensitive

base64_terminal = (2base64_char "==") / (3base64_char "=")

continue_req = "334" SPACE [base64] CRLF

CR = %x0C ;; ASCII CR, carriage return

CRLF = CR LF

CTL = %x00-1F / %x7F ;; any ASCII control character and DEL

LF = %x0A ;; ASCII LF, line feed

SPACE = %x20 ;; ASCII SP, space

9　安全问题考虑

如果客户端使用这个扩展得到不加密的渠道但是通过一个不安全的网络连接到协同工作的服务器的话，客户端将被阻断而永远不能发送邮件到服务器，当服务器不能够互助地进行验证和加密的时候。否则，攻击者将会通过截断 SMTP 的连接而偷取客户端的信件，或者假装服务器不支持认证，从而导致所有的 AUTH 命令失败。

在 SASL 商议开始之前，任何协议之间的交互都可以畅通无阻的进行，但也有可能被活动着的攻击者修改。因此客户端和服务器都必须抛弃在 SASL 之前获得的所有消息。

认证机制并不保护 TCP 端口，攻击者就会通过修改中转的连接而试图连接（SUBMIT）。AUTH 命令的参数就可以阻止这种攻击。

一个消息子服务客户端可能会要求用户通过验证，在任何它得到一个合适的 SASL 的时候。

因此，对于一个声明 SASL 机制的 SUBMIT 来说并不是合算的，在使用一个同

意匿名子任务的客户端而没有任何利益的机制的时候。

这个扩展并不是有意的取代或者被用来取代端到端的消息签名在加密系统 S/MIME 或者 PGP 中。此扩展和端到端的系统有一些细微的差别，主要是以下的部分。

（1）它通常只在一个被信任的范围里有效

（2）它保护整个消息的封装替，而不仅仅是正文

（3）它鉴证消息的子任务而不是消息的内容

（4）在发信者协同验证下一个中转点并且穿越一个合理的安全层的情况下，它可以给发信者一个保证，那就是可以把消息安全地传递到下一个地点。

另外需要说明的是，安全问题的考虑在 SASL 说明里面也有提到。

# 毕 业 设 计 开 题 报 告

1．结合毕业设计课题情况，根据所查阅的文献资料，撰写2000字左右的文献综述：

## 文 献 综 述

随着计算机网络的发展，人与人之间信息传输的时间大为缩短。许多文件都是以电子邮件的形式来传送；通常使用过计算机的人，或多或少都会用到Email来传输信息。通过电子邮件，人们可以进行文字、图片、视频、声音、数据文件等的传递。随着Internet网和WWW网的广泛普及，电子邮件的使用迅速增多起来。电子邮件的使用不仅在数量上有突飞猛进的发展，其重要性日益增加。据IDC（国际文献资料中心）统计，目前全球电子邮箱总数已超过5亿。而据CNNIC（中国互联网络信息中心）的最新调查，中国网络用户拥有E-mail帐号的平均值为2.6个，用户平均每周收到电子邮件数为12.9封，发出电子邮件数为8.2封。这说明电子邮件已不只是电话的替代品，它可以而且已经在广泛地应用着。当然，电子邮件也为人们带来了不利的一面。由于其接收发送电子邮件很少受到限制，造成电脑病毒、大量的垃圾邮件盛行，甚至个人隐私及安全受到了严重的威胁。但是，电子邮件作为当今社会主要的信息传播载体，发展趋势不会因此而停止。

目前由于电子邮件广泛而频繁的使用，国内外各大网站都提供了电子邮件服务，而且有些是免费服务。人们日常使用的电子邮件有很多，但是就方式来说只有两种。第一种是在网站上使用电子邮件，也就是一般所说的Web使用方式，通常人们使用时必须首先登录到WebMail服务器(网页邮件服务器)，通过身份验证后才可以查阅收发邮件。但是用户不能总是处于登录状态，也不能每时每刻发送接收邮件。如果长时间没有对WebMail服务器进行操作，则被认为登录超时，自动退出系统。用户需要再次登录邮件服务器，才可以进行操作。由于频繁的登录填写帐户和密码，造成了用户对电子邮件的使用产生不便。这时出现一种不需要访问Web页面，只需要您在本地机器上使用电子邮件的相关软件，就可以直接收发、管理电子邮件。它既能支持全部的 Internet 电子邮件功能，又能改正WebMail服务器使用的种种缺点，方便用户发送接收邮件。目前以微软的OutLook Express和国产FoxMail为两款经典代表。两者都是著名的客户端Email软件，功能不差上下。

客户端电子邮件软件一般都比 WebMail 服务器（网页邮件服务器）提供更为全面的功能。使用客户端软件收发邮件，登陆时不用下载网站页面内容，速度更快；使用客户端软件收到的和

曾经发送过的邮件都保存在自己的电脑中，不用上网就可以对旧邮件进行阅读和管理。同时实现多用户，多邮箱帐户，多 POP3 支持，对邮件管理更加快捷方便。正是由于电子邮件客户端软的种种优点，它已经成为了人们工作和生活上进行交流必不可少的工具。

电子邮件（简称 E-mai1）又称电子信箱、电子邮政，它是一种用电子手段提供信息交换的通信方式。它是全球多种网络上使用最普遍的一项服务。这种非交互式的通信，加速了信息的交流及数据传送，它是一个简易、快速的方法。通过连接全世界的 Internet，实现各类信号的传送、接收、存贮等处理， 将邮件送到世界的各个角落。 到目前为止， 可以说电子邮件是 Internet 资源使用最多的一种服务，E-mai1 不只局限于信件的传递，还可用来传递文件、声音及图形、图象等不同类型的信息。

电子邮件不是一种“终端到终端”的服务，是被称为“存贮转发式”服务。这正是电子信箱系统的核心，利用存贮转发可进行非实时通信，属异步通信方式。即信件发送者可随时随地发送邮件，不要求接收者同时在场，即使对方现在不在，仍可将邮件立刻送到对方的信箱内，且存储在对方的电子邮箱中。接收者可在他认为方便的时候读取信件，不受时空限制。在这里，“发送”邮件意味着将邮件放到收件人的信箱中，而“接收”邮件则意味着从自己的信箱中读取信件，信箱实际上是由文件管理系统支持的一个实体。因为电子邮件是通过邮件服务器（mai1 server）来传递文件的。通常 mail server 是执行多任务操作系统 UNIX 的计算机，它提供 24 小时的电子邮件服务，用户只要向 mail server 管理人员申请一个信箱帐号，就可使用这项快速的邮件服务。

电子邮件的工作原理：

（1） 电子邮件系统是一种新型的信息系统，是通信技术和计算机技术结合的产物。

电子邮件的传输是通过电子邮件简单传输协议（Simple Mail Transfer Protocol,简称 SMTP）这一系统软件来完成的，它是 Internet 下的一种电子邮件通信协议。

（2） 电子邮件的基本原理，是在通信网上设立“电子信箱系统”，它实际上是一个计算机系统。系统的硬件是一个高性能、大容量的计算机。硬盘作为信箱的存储介质，在硬盘上为用户分一定的存储空间作为用户的“信箱”，每位用户都有属于自己的一个电子信箱。并确定一个用户名和用户可以自己随意修改的口令。存储空间包含存放所收信件、编辑信件以及信件存档三部分空间，用户使用口令开启自己的信箱，并进行发信、读信、编辑、转发、存档等各种操作。系统功能主要由软件实现。

（3） 电子邮件的通信是在信箱之间进行的。用户首先开启自己的信箱，然后通过键入命令的方式将需要发送的邮件发到对方的信箱中。邮件在信箱之间进行传递和交换，也可以与另一个邮件系统进行传递和交换。收方在取信时，使用特定帐号从信箱提取。

# 毕 业 设 计 开 题 报 告

## 2．本课题要研究或解决的问题和拟采用的研究手段（途径）：

一.功能概述

此次设计我们将以国内久负盛名的FoxMail电子邮件客户端软件为范本来主要是实

现发送邮件，接受邮件这两个最基本但是也最重要的功能，以及其他的发送附件，发送

多媒体附件，实现代理服务器登录SMTP服务器,以及实现用户可以自主编辑的通讯簿等

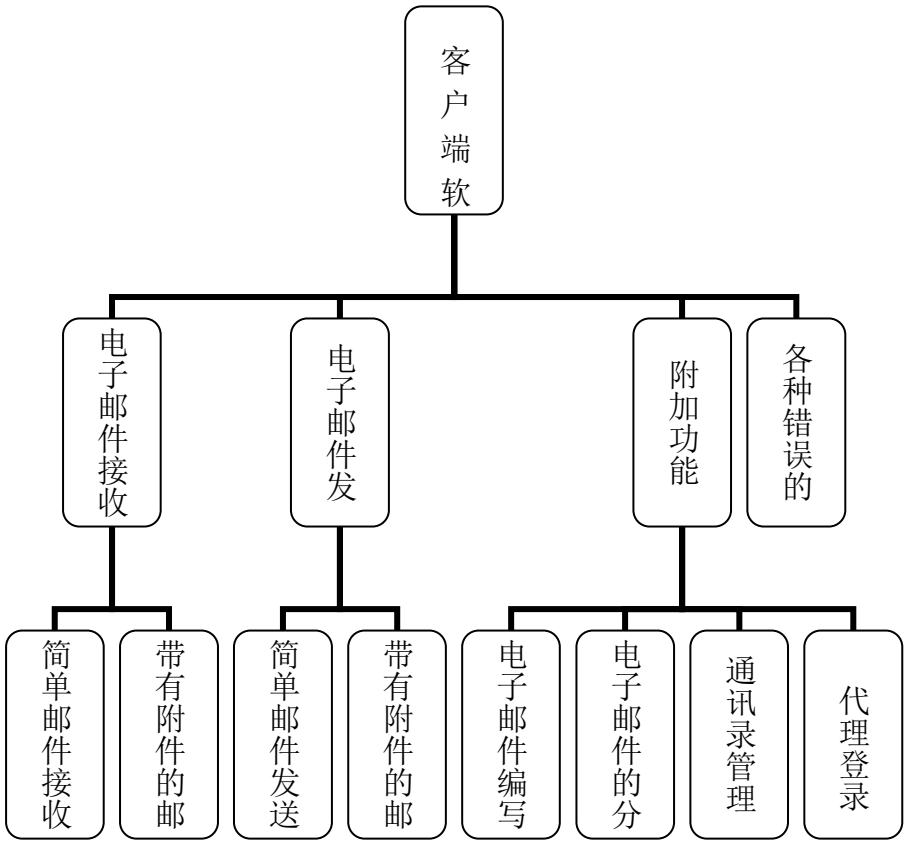
等附加功能。而我个人主要负责电子邮件的发送这个功能。

二．编程环境选择

语言选择:Visual Stdio 2003中的C#(C Sharp)

操作系统选择: WindowsXP系统

三．程序功能框图



四．功能的实现

C#语言，作为微软官方主推语言，拥有和 VB 一样的对于开发界面窗体的便利，同时又保持了 C++的高效和便利。而且作为 C 语言和 C ++的发展，它在完全兼容 C++的基础上又有乐很大的发展。C#作为一种面向对象的语言，也为以后程序的维护以及扩展提供了方便。而且最重要的是将来即使和 VC++结合的时候也不会存在问题。

对于发送邮件来说，在.NET 中的具体的实现基本上有三种方式

1) 利用 System.Web.Mail 这个名称空间.在这个名字空间下，有一个专门使用 SMTP 协议来发送邮件的类：SmtpMail,它已能满足最普通的发送邮件的需求。这个类只有一个自己的公共函数--Send()和一个公共属性—SmtpServer，您必须通过 SmtpServer 属性来指定发送邮件的服务器的名称（或 IP 地址）,然后再调用 Send()函数来发送邮件。您可以在 Send 函数的参数 MailMessage 对象中设置邮件的相关属性，如优先级、附件等等。除了以 MailMessage 对象为参数（如上述代码），Send 函数还可以简单的直接以邮件的 4 个主要信息（from，to，subject，messageText）作为字符串参数来调用。

2) 使用 CDO 组件发送邮件

CDO 是 Collaboration Data Objects 的简称，它是一组高层的 COM 对象集合，并经历了好几个版本的演化，现在在 Windows2000 和 Exchange2000 中使用的都是 CDO2.0 的版本（分别为 cdosys.dll 和 cdoex.dll）。CDOSYS 构建在 SMTP 协议和 NNTP 协议之上，并且作为 Windows2000 Server 的组件被安装，您可以在系统目录（如 c:\winnt 或 c:\windows）的 system32 子目录中找到它(cdosys.dll)。

CDO 组件相对于先前介绍的 SmtpMail 对象功能更为丰富，并提供了一些 SmtpMail 类所没有提供的功能，如通过需要认证的 SMTP 服务器发送邮件等。最后，其它的那些附加功能完全都可以以组件的形式加到主程序中，这样以来有利于代码的高效性，安全性，也为以后的升级，维护提供了方便。

3）使用 Socket 撰写邮件发送程序

如果 SmtpMail 不能满足设计的需求，CDO 又不够直截了当，那就只能自己动手了；在熟悉 Socket 编程的基础上，自己写一个发送邮件的程序并不很难。具体

如下：

1.首先，需要使用 EHLO 而不是原先的 HELO。

2.EHLO 成功以后，客户端需要发送 AUTH 原语，与服务器就认证时用户名和密码的传递方式进行协商。

3 如果协商成功，服务器会返回以 3 开头的结果码，这是就可以把用户名和密码传给服务器。

4.最后，如果验证成功，就可以开始发信了。

采用以上任何一种都可以实现发送邮件的功能了，具体的是三种方法实现的功能上

来说是越来越完善的。至于那些其他的媒体附件，实现代理服务器登录 SMTP 服务器，

以及实现用户可以自主编辑的通讯簿等等附加功能都可以以组件甚至插件的形式加到

主程序中了。以组件的形式编写这些附加功能，有利于软件以后功能的扩展和完善，也有利于软件以后的升级与维护

2006 年 2 月 20 日~2006 年 3 月 10 日　　开题报告

2006 年 3 月 15 日~2006 年 4 月 5 日　　发送功能的基本实现

2006 年 4 月中旬 ~2006 年 6 月　　其余各功能的完善翻译文献

2006 年 6 月中旬　　答辩

# 毕 业 设 计 开 题 报 告

**指导教师意见：**

指导教师：_____

年

月　日

所在系审查意见：

系主任：_____

年　月

日

## 参考文献

[1] Simon Robinson, K.Scott Allen 等.C#高级编程. 北京:清华大学出版社, 2002,3

[2] Tom Archer. C#技术内幕. 北京:清华大学出版社, 2002,1

[3]沉舟.Microsoft.NET 编程语言 C#. 北京:希望电子出版社 2001,3

[4]罗军舟,黎波涛,杨明等.TCP/IP 协议及网络编程技术. 北京: 清华大学出版 2004,10

[5] Tim Parker .TCP/IP 协议及网络编程技术. 北京: 机械工业出版社 , 2000,7

[6] 周存杰 . Visual C#.NET 网络核心编程. 北京:清华大学出版社, 2002,11

[7] 电脑编程技巧与维护杂志社.C#编程技巧典型案例解析. 北京:中国电力出版社,2005,8

[8] 云颠工作室. Visual C#中文版全面剖析. 北京:中国水利水电出版社, 2003,5

[9] 叶树华 《电子协议与编程》,《电子邮件格式》,《电子邮件接收》,《mime 编码解码与发送附件》

[10] MSDN 中文网站网络广播 C#设计模式纵谈 http://www.microsoft.com/china/msdn/events/webcasts/shared/Webcast/MSDNWebCast.aspx

[11] 滁州,马金虎,朱力勇. 编写基于 SMTP 网络应用程序. 电脑爱好者,2003,5:92~94

[12] 滁州,马金虎,朱力勇. 编写基于 POP3 网络应用程序. 电脑爱好者,2003,6:92~94

[13] 潘泰国. 新一代电子邮件系统. 电子技术应用. 1992,11

[14] 代继红. SMTP 认证机制模块化设计及实现. 中南民族大学学报(自然科学版), 2005,4

[15] 胡安廷. 简单实现中文邮件. 中国计算机报, 2004,11