一种具有身份认证功能的微型邮件引擎

王晓东，　郑连清，　沈树涛
(空军工程大学 电讯工程学院，陕西 西安　710077)

摘　要:依据SMTP协议和Base64编码规则,利用VC++编译器实现了一种具有身份认证功能的邮件引擎,程序代码少、运行速度快、可作为函数调用、能发送二进制文件,测试证明性能良好。

关键词:邮件引擎;VC++;身份认证;SMTP协议
中图分类号:TP309　　**文献标识码**:A　　**文章编号**:1009-3516(2003)04-0048-05

邮件引擎是用于发送电子邮件的专用程序,它的性能直接影响着电子邮件的发送质量和效率。目前许多软件都具有发送电子邮件的功能,如Outlook、Foxmail等。这些软件功能固然十分强大,但是软件自身庞大、使用起来比较繁琐、操作不灵活、对系统依赖性强,特别是在用户设计自己的网络软件并为其开发电子邮件投递功能时,由于邮件引擎并不是软件的主要功能部分,所以总是希望邮件引擎部分不会明显增加软件篇幅。此外,用户还会希望邮件引擎本身能够具有独立工作能力,如果按部就班地调用上述邮件发送软件,显然是不明智的[1]。为此,我们利用VC++语言设计一种具有通用性的微型邮件引擎。

1　SMTP与对话

SMTP(简单邮件传输协议)是用于电子邮件传输的基本协议。协议规定了计算机间交换电子邮件时要用到的控制信息,包括对正确连接的校验、发送者的标识、传输参数的商定及邮件的传输。SMTP是按照客户/服务器方式进行工作的,发信人的主机为客户方,收信人的邮件服务器为服务方。SMTP还定义了进行邮件传输命令以及相应的响应码,客户机与服务器按照这些命令及其响应进行对话。图1描述1个具体的SMTP对话过程。

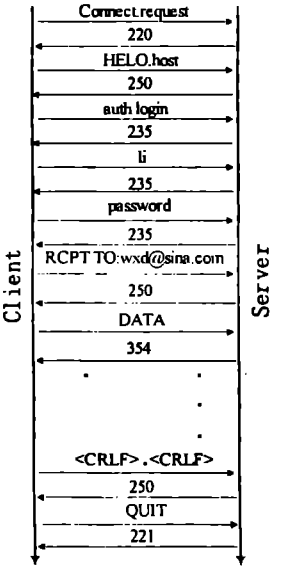

图1　SMTP对话过程

首先,邮件用户发出连接请求后,服务器用"220"做出响应;用户接着发出"HELO"命令交换标识符,如果成功,用户则发出"MAIL FROM"命令告诉服务器一项新的邮件业务开始;随后,服务器对用户进行身份认证;接着,用户发送"RCPT TO"命令给出服务器的转发路径地址,并以"DATA"命令通知下面就是报文的内容,服务器用"354" 响应,表示邮件可以开始输入;当邮件发送完毕,用户用双回车符之间的"."表示输入结束,服务器用"250"做出响应;任务完成后,双方分别使用"QUIT"和响应"221"关闭连接。如果需要向多个用户发送报文,可用多个"RCPT TO"命令进行补充说明[2]。

邮件引擎的SMTP应答工作由Smtpcmd()函数完成,该函数按照下面模式完成SMTP对话中的每条命令收发及其应答码判断。

```
send(mysock,cmd,strlen(cmd),0); //cmd为SMTP命令行
if(! getrespond(user_success))return FALSE; //接收并进行应答码判断,再根据图1对话过程的顺序依
```

收稿日期:2003-01-22
基金项目:国家自然科学基金资助项目(69631020)
作者简介:王晓东(1974-),男,陕西长安人,硕士生,主要从事信息与信号处理研究;
郑连清(1965-),男,山西侯马人,教授,主要从事信息战仿真研究.

次发送 SMTP 命令,即可完成与邮件服务器的应答功能。

在每次发送完毕 SMTP 命令以后,邮件引擎会接收到邮件服务器对 SMTP 命令返回的应答码。由于每次的应答码不尽相同,为了快速判断应答码所代表的服务器信息,本邮件引擎设计了 getrespond( ) 函数。该函数分别利用数组、枚举结构将可能发生的应答码、SMTP 回应事件一一对应(除 last_success 用于判断 SMTP 对话是否结束)起来,根据应答码与回应事件的对应关系进行判读,不但便于理解而且简化了操作过程,数组结构如下:

```
UINT errocode[ ] = //SMTP 应答码
{ 250,220,334,235,354,221 };
```

枚举结构如下:

```
enum errorresponse          //SMTP 回应事件
{ generic_success,          //请求的邮递活动已完成
    connect_success,        //(域)服务准备好
    user_success,           //用户名接收完毕
    password_success,       //用户密码正确
    data_success,           //可以开始传输数据
    quit_success,           //退出
    last_success, };        //SMTP 对话结束
```

getrespond( ) 函数的具体定义如下:

```
bool getrespond(errorresponse respond_expected)
{TCHAR receive[128];    //接受相应码的缓冲区
    ZeroMemory(receive,sizeof(receive));
    ::recv(mysock,receive,sizeof(receive) - sizeof(TCHAR),0);
     UINT uError = atoi(receive);
    if(uError! = errocode[respond_expected]) return FALSE;
    return TRUE;}
```

在 Smtpcmd( ) 函数中每次发送完 SMTP 命令后,getrespond( ) 都会被调用。调用后,getrespond( ) 清空接收数据变量 receive(不能太短,必需将 SMTP 相应内容全部读完,否则会引起溢出,从而影响邮件引擎的正常工作),利用 recv( ) 函数收听服务器发来的应答码,并将该应答码转换为十进制数与预期事件对应的应答码相比较,如果比较结果相等则代表该步 SMTP 对话顺利通过,否则失败。

2　邮件编码

受网络带宽的限制,电子邮件在初始设计时只适用于采用 ASCII 码格式的文字信息的传送(虽然目前网络带宽已经大大改观,但是电子邮件依然采用这种格式)。由于 ASCII 码是 7 位代码,因而非 ASCII 码格式的文件(二进制、Unicode 文件)不经过编、译码,在传送过程中文件信息会因 ASCII 码 7 位的限制而被分解,分解之后,收信方将无法正确解释邮件信息,导致邮件传输失败。邮件中的附件文件一般多为二进制文件,所以必须对其进行编、译码后,才可在 Internet 网络中顺利传送,否则每一字节末尾的一位将被截掉[3]。

理论上讲,只要是低于 7 位的编码都可进行网络传输,考虑到程序运行速度和通用性因素,本邮件引擎采用目前网络中通用的 Base64 编码方式,该方式实际上是将文件数据按 6 位为 1 个单位进行分组并根据这个分组的数值用 ASCII 标准字符进行替代,译码过程反之。编码程序具体过程是分批从待编码的二进制文件中读出一个字节的数据,然后切割成 6 位数的子段(余下的数据计入下一个子块);然后根据 6 位数的子数据段存储的数值大小,在规定的 2^6 个字符编码表中进行检索,替换编码前的二进制数;当编码全部结束后,如果字节数不够 4 的倍数时补“ = ”,最终完成对原数据的 ASCII 码转化,程序流程如图 2 所示。本邮件引擎的编码工作是由 Base64encode( ) 函数完成的,该函数不但可对二进制附件文件编码,而且可对邮件的正文进行快速编码,函数定义了以下关键变量:

```
char * mybuffer;      //需要编码的字符串
```

```
char * encoderesult;        //编码后的结果
const char m_sBase64[ ] =
"ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789 + /";      //编码表
int m_nMask[ ] = { 0, 1, 3, 7, 15, 31, 63, 127, 255 };      //用于进行选位
int remainbit;              //剩余位
ULONG storagebit;           //存储读取的数据
ULONG decodetemp;           //临时存储处理中的数据
```

编码的具体过程以一个 16 位的二进制数 1010110001100001 为例:初始化变量后,Base64encode( )函数首先将该二进制数的第一个字节的 8 位数 10101100 读入变量 c,并赋值给 Storagebit,之后将 Storagebit 右移位 remainbit - 6 位(即:2 位)的结果赋值给变量 decodetemp,为 101011,也就是十进制的 43,在字符编码表 m_sBase64 中进行检索为"r",记录入字符串 result,剩余值 remainbit 为 2。接着将二进制数的第二个字节的 8 位数 01100001 读入变量 c,并赋值给 storagebit,将 storagebit 右移位 remainbit - 6 位(即:4 位)的结果赋值给变量 decodetemp,为 000110,也就是十进制的 6,在字符编码表中进行检索为"G",记录入字符串 result,remainbit 的值为 4。这时 2 个字节已经读完,但仍有 4 位二进制数未完成编码,根据剩余的 4 位二进制数 0001 即十进制中的 1,在字符编码表中进行检索为"B",记录入字符串 result。补齐"="后,得出 1010110001100001 编码结果为"rGB="。Base64encode( )是利用 C 语言中的"位运算"来实现的,可供邮件引擎其它部分进行调用。经测试,上述编码程序具有短小、高速以及误码率低的特点,经其编码后的邮件附件文件可以在网络上准确无误的传输。

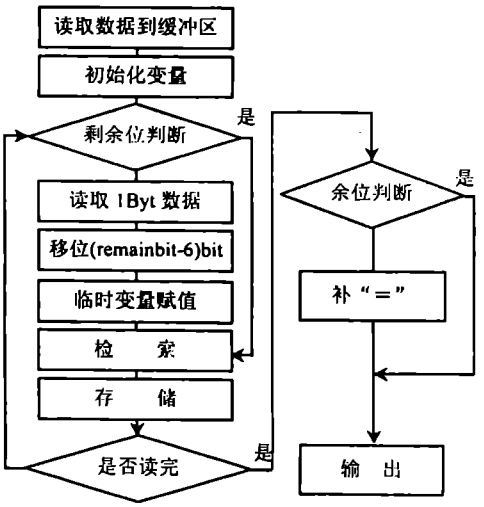

图 2 Base64 编码流程

3 邮件身份认证

由于受到邮件病毒及垃圾邮件等威胁,目前大多数邮件服务器在提供邮件服务时都添加了身份认证机制,即:在 SMTP 协议的基础上增加了一组应答对话,具体过程是:当用户向服务器申请邮件服务时,利用 auth login 命令向邮件服务器提供用户的信息(用户名、邮箱密码),这些信息经过服务器认证后用户才会得到服务。因此,设计邮件引擎时必须考虑身份认证功能,按照邮件服务器的要求把正确的邮件用户身份和密码提供给邮件服务器。这里需要注意的是:由于用户的名称与密码有可能是非 ASCII 字符,所以也要对它们进行 Base64 编码,否则即使传送的用户名、密码属于 ASCII 字符,邮件服务器也会拒绝服务。下面以用户名是 wangxiaodong1974,密码是 19780901 为例,来说明邮件引擎所完成的一个具体身份认证程序过程:

```
ZeroMemory(cmd,sizeof(cmd));
strcpy(cmd,"auth login\r\n");
::send(mysock,cmd,strlen(cmd),0);      //发送 auth login 命令
if(! getrespond(user_success)return FALSE;
ZeroMemory(cmd,strlen(cmd));
strcpy(cmd,"wangxiaodong");
Base64encode(cmd,strlen(cmd),Base64buffer)      //对用户名进行编码
CopyMemory(cmd,Base64buffer,strlen(cmd));
strcat(cmd,"\r\n");
::send(mysock,cmd,strlen(cmd),0);      //发送用户名
if(! getrespond(user_success) return FALSE;
```

```
ZeroMemory(cmd,strlen(cmd));
strcpy(cmd,"19780901");
ZeroMemory(Base64buffer,strlen(Base64buffer));
Base64encode(cmd,strlen(cmd),Base64buffer);      //对邮箱密码进行编码
CopyMemory(cmd,Base64buffer,strlen(Base64buffer));
strcat(cmd,"\r\n");
::send(mysock,cmd,strlen(cmd),0);      //发送用户密码
if(! getrespond(password_success) return FALSE;
```

该段程序在 SMTP 对话过程中进行到身份认证步骤时运行,工作流程如图 3 所示。首先邮件引擎向服务器发送"auth login"命令,标志身份认证开始,服务器用"334"做答;而后,邮件引擎向服务器发送经过编码后的用户名,服务器应答码为"334";紧接着发送编码后的邮箱密码,服务器根据所提供的用户名与密码进行查询,如果与记录相符,服务器就会及时的清除缓冲区,复位状态表并准备接收报文,并向客户端发送"235"表示通过身份认证,否则拒绝继续服务,身份认证就完成了[4]。

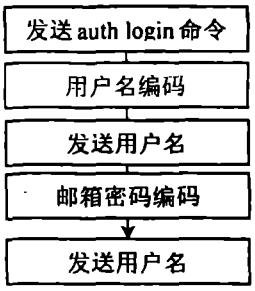

图 3　邮件引擎用户身份认证流程

4　邮件引擎的主框架

邮件系统的传送功能是由相应的 SMTP 进程完成,这样的进程包括发送者进程、接收者进程,它们之间的功能关系如图 4 所示。发送者 SMTP 进程即邮件引擎,一般运行于客户端,其功能是把信件发送到 SMTP 服务器,也就是本邮件引擎所完成的主要工作。接收者 SMTP 进程运行于服务器端,负责邮件的接收。由模型可以看出:邮件引擎的工作主要就是处理邮件用户信息、处理并发送文件系统中的文件以及与邮件服务器对话。根据上述功能要求和 SMTP 协议应答流程,利用套接字即可建立邮件引擎(流程如图 5 所示)。

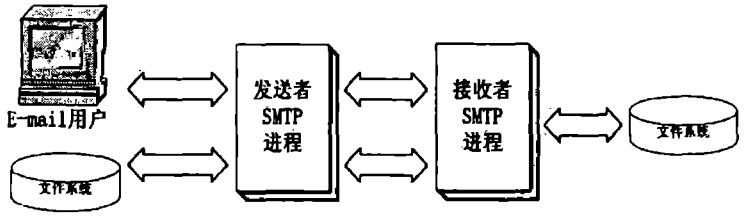

图 4　SMTP 协议邮件传输模型

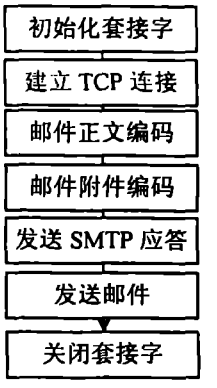

图 5　邮件引擎工作流程

邮件引擎首先建立一个 SOCKET 变量,并且在完成变量初始化后,调用 API 函数 gethostbyname(),通过网络中的 DNS 服务器获取目标邮件服务器的 IP 地址[5];在得到 IP 地址后使用该地址和缺省端口 25,利用 connect()函数与邮件服务器的 25 端口进行 TCP 连接[5];通常如果连接成功邮件服务器会主动发送 1 个问候语,接收到该问候语表示可以进行 SMTP 对话了。邮件引擎会首先对邮件正文以及附件进行编码,作好发送邮件前的准备工作;随后,邮件引擎调用 Smtpcmd()函数(函数包括身份认证过程),进行 SMTP 对话;如果对话申请到服务器的邮件服务,就可发送经过编码后的邮件及其附件了;在完成邮件投递工作之后,套接字连接被断开,清除所占用的存储空间,邮件引擎的一个程序周期就顺利完成了。

5 程序的微缩编译

虽然本邮件引擎大量使用了 API 函数来减小程序篇幅、提高运行速度,但是为了使其更加趋于微型化、编译后的可执行程序尽量的小,在使用 VC ++ 编译器时采用下述方法:选择编译方式为 Win32 - Release,并在编译器的 Project - > Settings 选项中将 Object/library modules 内无关的 lib 删除,然后更改为通用 MSVCRT. LIB kernel32. lib user32. lib, 此外,编译程序前,在 Project - > Settings 选项中的 Link 属性页的 Project Options 下添加:"/ALIGN:4096"(指定程序不是驱动程序)然后进行编译。

如果上述做法还不能满足要求,还可采用设计专有入口函数,不使用链接器默认提供的程序初始化操作代码,入口函数如下:

```
void EntryPoint( )
{ ExitProcess( WinMain( GetModuleHandle( NULL) ,
NULL, GetCommandLine( ) ,SW_SHOWNORMAL) ) ; } ,
```

程序设计者可根据自己所需,有选择的添加初始化变量以及类。这样将大大缩减生成文件的大小,经编译后的可执行文件仅有 5 kByt 左右。

6 结束语

通过互联网上的测试,本邮件引擎可以顺利通过绝大多数邮件服务器(如:搜狐、新浪、263 等常用邮件服务器)的身份认证机制并建立有效连接,发送电子邮件,且运行速度快,对操作系统的依赖性小,编译后的可执行文件仅有 7 kByt 左右。如有必要还可以对本邮件引擎进行多线程开发,使其具有执行多任务功能,不过这样有可能增加程序篇幅,但同样采用上述微缩编译手段,程序的大小可以控制在较小范围内。

参考文献:

[1] 王家俊. 亲手创建 Internet 四大服务[M]. 北京:人民邮电出版社,1999.
[2] RFC821. Simple Mail Transfer Protocol[S].
[3] 李博轩. Visual C ++ 网络及 Internet 开发指南[M]. 北京:清华大学出版社,2000.
[4] RFC1423. Algorithms,Modes and Identifiers[S].
[5] 朱友芹,张争平,郭 立,等. 新编 Windows API 参考大全[M]. 北京:电子工业出版社,2000.

(编辑:门向生)

A Subminiature E - Mail Engine with Authentication

WANG Xiao - dong, ZHENG Lian - qing, SHEN Shu - tao
(The Telecommunication Engineering Institute, Air Force Engineering University, Xi'an, Shaanxi 710077, China)

Abstract: According to SMTP protocol and the principle of Base64 encode, an E - mail engine program is designed which can pass the authentication of e - mail server with VC. Being tested on network, the engine is of subminiature, less programming code and high operating speed. Moreover, it can work as a common function and can be used for sending binary files on network.

Key words: E - mail engine; VC ++ ; authentication; SMTP protocol