实验十 POP3 协议与 BASE64 编码

10.1 实验目的

电子邮件是互联网中一种重要的应用，给人们提供了极大便利，本实验主要介绍基于 POP3 协议实现邮件的接收，并且介绍了邮件数据中常用的 BASE64 编码方案。

10.2 POP3 协议

电子邮件保存在邮件服务器上，用户通过网络协议从获取邮件内容，IMAP 协议和 POP3 协议用于实现从服务器下载邮件到客户机。POP3(Post Office Protocol 3) 即邮局协议的第 3 个版本，规定个人计算机如何连接到互联网上的邮件服务器进行收发邮件的协议。POP3 协议是 TCP/IP 协议族中的一员，由 RFC 1939 定义，POP3 协议允许用户从服务器上把邮件下载到本地主机，客户端通过向服务器发送命令删除或保存在邮件服务器上的邮件。

POP3 协议采用的是 C/S 的网络通信模式，它的默认服务端口为 110，POP3 客户向 POP3 服务器发送命令并等待响应，POP3 命令采用命令行形式，用 ASCII 码表示。服务器响应是由一个单独的命令行组成或多个命令行组成，响应第一行以 ASCII 文本 +OK 或 -ERR(OK 指成功，-ERR 指失败）指出相应的操作状态是成功还是失败。

使用 POP3 协议通信时具有三种状态：AUTHORIZATION（授权），TRANSACTION（处理），UPDATE（更新），当客户机与服务器建立连接时，客户机向服务器发送自己身份 (这里指的是账户和密码）并由服务器成功确认，即客户端由认可状态转入处理状态，在完成列出未读邮件等相应的操作后客户端发出 quit 命令，退出处理状态进入更新状态，开始下载未阅读过的邮件到计算机本地之后最后重返认证状态确认身份后断开与服务器的连接。表 10-1 例举了部分 POP3 协议用到的命令及其意义说明，服务器会根据用户的操作返回一定的响应文本。

POP3 使用 PASS 命令传送用户的密码，并以明文传送，因此具有安全隐患，另外一个命令 APOP，可以安全传输用户密码，避免了安全隐患。多数著名邮件服务都提供 POP3 服务，例如 126 邮箱，139 邮箱等，其服务器地址分别为 pop.126.com 及 pop.139.com。用户可以使用 telnet 工具来测试 pop3 协议的命令，方法是在命令行输下面的命令：

telnet pop.126.com 110

成功连接服务器后，输入 POP3 的命令进行邮件操作。

10.3 BASE64 编码

BASE64 编码是一种简单且应用广泛的字节编码方式，在 MIME 格式的电子邮件中，BASE64 将 binary 的字节序列数据编码成 ASCII 字符序列构成的文本，即只使用普通的英文

表 10-1 POP3 命令

命令	参数	状态	描述
USER	用户名	授权	与 pass 命令一起确认用户信息
PASS	密码	授权	与 user 命令一起确认用户信息
STAT	无	处理	请求服务器发回关于邮箱的统计资料，如邮件总数和总字节数
UIDL	邮件编号	处理	返回邮件标识
LIST	邮件编号	处理	返回邮件数量和邮件大小
RETR	邮件编号	处理	获取邮件内容
DELE	邮件编号	处理	将给定邮件标记为删除，在 QUIT 后执行删除
RSET	无	处理	重置删除标记，可用于消息除 DELE 命令
TOP	邮件编号	处理	返回标记邮件的头 n 行内容，n 必须为正整数
QUIT	无	更新	退出登录，服务器返回确认信息

字母来表示任意的数据。使用时，在传输编码方式中指定 base64。使用的字符包括大小写字母各 26 个，加上 10 个数字，和加号"+"，斜杠"/"，一共 64 个字符，等号"="用来作为后缀用途。BASE64 的定义由 RFC1421 和 RFC2045 说明。编码后的数据比原始数据略长，为原来的 4/3。在电子邮件中，根据 RFC822 规定，每 76 个字符，还需要加上一个回车换行。可以估算编码后数据长度大约为原长的 1.35 倍。

BASE64 编码方法是将三个 byte 的数据，先后放入一个 24bit 的缓冲区中，先来的 byte 占高位。数据不足 3byte 的话，于缓冲区中剩下的 Bit 用 0 补足。然后，每次取出 6 个 bit，按照其值选择 ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/中的字符作为编码后的输出。不断进行，直到全部输入数据转换完成。如果最后剩下两个输入数据，在编码结果后加 1 个"="；如果最后剩下一个输入数据，编码结果后加 2 个"="。

邮件中的内容与附件都采用了 BASE64 编码方式，虽然算法不复杂，开发者实现邮件编码转换却很繁琐，.NET 平台提供 SmtpClient 等类执邮件操作支持 BASE64 运算而无需用户重新实现。

10.4 实验内容

10.4.1 使用 POP3 协议检查与下载邮件

本小节实现从 126 邮件服务器定期查收邮件功能，程序主要流程如图 10-1 所示，程序使用线程循环向服务器发 POP3 命令，如果检测到新邮件到达，则下载邮件并显示邮件内容。新建一窗体应用程序，设计界面可参考图 10-2，在窗体中定义要使用的常量和变量。

```
//动态链接库引入
[DllImport("User32.dll", EntryPoint = "SendMessage")] private
static extern int SendMessage(
IntPtr hWnd, // handle to destination window int Msg,
// message
int wParam, // first message parameter
```

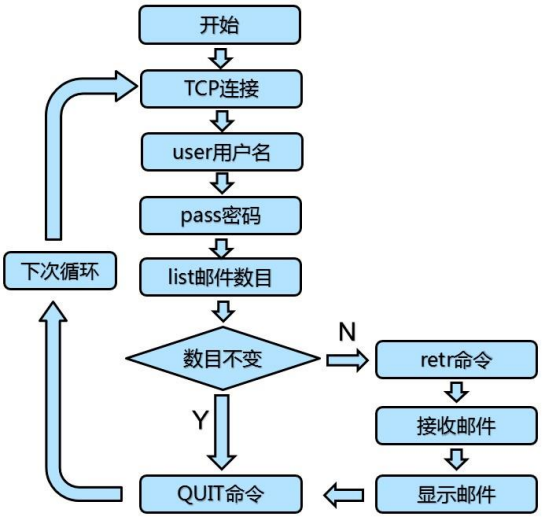

图 10-1 邮件检查与下载程序流程

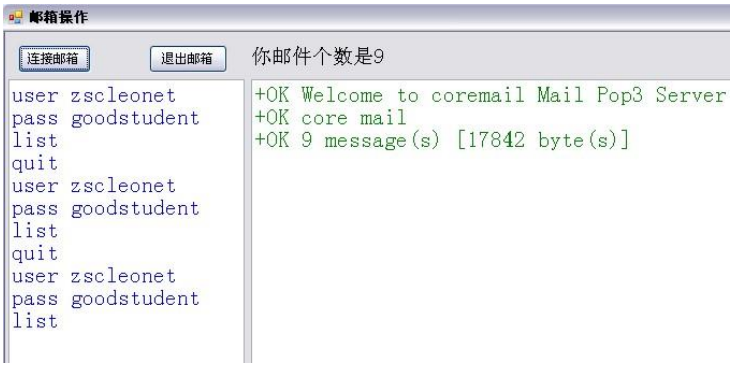

图 10-2 邮件检查与下载运行效果

```
int lParam // second message parameter
);
public static IntPtr main_wnd_handle;
public static string send_str;
public static string recv_str;
public static string mailbox_str;
public static int recv_octects = 0;
public static MemoryStream ms_recv;
//定义消息常数
public const int TRAN_SEND_INFO = 0x500;
public const int TRAN_REPLY_INFO = 0x501;
public const int TRAN_MAIL_CONTENT = 0x502;
public const int MAIL_COUNT = 0x503;
public static ManualResetEvent MRE_check_end;
//重写窗体的消息处理函数：
protected override void DefWndProc(ref System.Windows.Forms.Message m)
```

```
{ switch (m.Msg)
   {
      case TRAN_SEND_INFO:
         textBox1.Text += send_str;
         break;
      case TRAN_REPLY_INFO:
         textBox2.Text += recv_str; break;
      case TRAN_MAIL_CONTENT:
         recv_str = Encoding.UTF8.GetString(ms_recv.GetBuffer(), 0, recv_octects);
         textBox2.Text += recv_str;
         break;
      case MAIL_COUNT:
         label1.Text = mailbox_str; break;
      default:
         base.DefWndProc(ref m); break;
   }
}
```

设计工作线程体，工作线程将执行向服务器发命令，并接收服务器响应的任务。

```
//线程体
static void thread_pop_con()
{
   //线程流程
   try
   {
      IPAddress ipadd_dest = Dns.GetHostEntry("pop.126.com").AddressList[0];
      IPEndPoint remoteEP = new IPEndPoint(ipadd_dest, Int32.Parse("110"));
      //连接服务器
      // Create a TCP/IP socket.
      Socket client_sock = new Socket(AddressFamily.InterNetwork,SocketType.Stream,
      ProtocolType.Tcp);
      client_sock.SetSocketOption(SocketOptionLevel.Socket, SocketOptionName.NoDelay, 1);
      client_sock.Blocking = true;
      client_sock.Connect(remoteEP);
      try { }
      catch (SocketException se3)
      {
         MessageBox.Show(" 客户端异常" + se3.Message);
      } }
   catch (SocketException se1)
```

```
    {
        MessageBox.Show("SocketException 客户端连接不到服务器呢" + se1.Message);
    }
    catch (Exception se2)
    {
        MessageBox.Show(" 客户端异常" + se2.Message);
    }
}
```

连接服务器，向服务器发送用户名信息：

```
//连接逻辑
byte[] SendDataBuffer = new byte[1024];//数据发送缓存
byte[] ReadDataBuffer = new byte[1024];//数据接收缓存
int recv_package_len = 0;
//POP 服务器会先发送一些响应字符串到客户端，显示客户端登录正常
recv_package_len = client_sock.Receive(ReadDataBuffer, 1024, SocketFlags.None);
recv_str = Encoding.UTF8.GetString(ReadDataBuffer, 0, recv_package_len);
//通知窗体显示收到的字符串
SendMessage(main_wnd_handle, TRAN_REPLY_INFO, 100, 100);
//send CMD: user send_str = "user
zscleonet\r\n";
//通知窗体显示发出的字符串
SendMessage(main_wnd_handle, TRAN_SEND_INFO,100,100);
byte[] b_cmd = Encoding.ASCII.GetBytes(send_str);
Array.Clear(SendDataBuffer, 0, 1024);
Array.Copy(b_cmd, SendDataBuffer, b_cmd.Length);
client_sock.Send(SendDataBuffer,b_cmd.Length,SocketFlags.None);
//接收服务器响应
recv_package_len=client_sock.Receive(ReadDataBuffer, 1024, SocketFlags.None);
recv_str=Encoding.UTF8.GetString(ReadDataBuffer, 0, recv_package_len);
//通知窗体显示收到的字符串
SendMessage(main_wnd_handle, TRAN_REPLY_INFO, 100, 100);
//发送用户密码：
//send CMD: pass
send_str = "pass goodstudent\r\n";
SendMessage(main_wnd_handle, TRAN_SEND_INFO, 100, 100);
b_cmd = Encoding.ASCII.GetBytes(send_str);
Array.Clear(SendDataBuffer, 0, 1024);
Array.Copy(b_cmd, SendDataBuffer, b_cmd.Length);
client_sock.Send(SendDataBuffer, b_cmd.Length, SocketFlags.None);
```

```
//接收服务器响应
recv_package_len = client_sock.Receive(ReadDataBuffer, 1024, SocketFlags.None);
recv_str = Encoding.UTF8.GetString(ReadDataBuffer, 0, recv_package_len);
//通知窗体显示收到的字符串
SendMessage(main_wnd_handle, TRAN_REPLY_INFO, 100, 100);
//验证用户成功后，获取邮件总数：
Regex str_num;
Match m;
Int32 mail_len;
String[] result_num;
Int32 old_total_mail_len,new_total_mail_len;
send_str = "list\r\n";
SendMessage(main_wnd_handle, TRAN_SEND_INFO, 100, 100);
b_cmd = Encoding.ASCII.GetBytes(send_str);
Array.Clear(SendDataBuffer, 0, 1024);
Array.Copy(b_cmd, SendDataBuffer, b_cmd.Length);
client_sock.Send(SendDataBuffer, b_cmd.Length, SocketFlags.None);
//接收服务器响应
recv_package_len = client_sock.Receive(ReadDataBuffer, 1024, SocketFlags.None);
recv_str = Encoding.UTF8.GetString(ReadDataBuffer, 0, recv_package_len);
result_num = recv_str.Split("\r\n ".ToCharArray(), StringSplitOptions.RemoveEmptyEntries);
old_total_mail_len = Int32.Parse(result_num[2]);
```

检验新邮件使用了循环结构，其过程是反复登录服务器，读取邮件数目，如果邮件数目发生变化，表示有新邮件到达。通过对邮件编号进行对比获知新邮件编号，通过 retr 命令下载邮件内容。在读取邮件内容前使用了类 Regex，Regex 是正则表达式应用类，它能够解析特定格式的文本内容，从服务器返回的字符串中解析出邮件长度值。

```
do{
    //send CMD: quit send_str = "quit\r\n";
    SendMessage(main_wnd_handle, TRAN_SEND_INFO, 100, 100);
    b_cmd = Encoding.ASCII.GetBytes(send_str);
    Array.Clear(SendDataBuffer, 0, 1024);
    Array.Copy(b_cmd, SendDataBuffer, b_cmd.Length);
    client_sock.Send(SendDataBuffer, b_cmd.Length, SocketFlags.None);
    //接收服务器响应
    recv_package_len = client_sock.Receive(ReadDataBuffer, 1024, SocketFlags.None);
    recv_str = Encoding.UTF8.GetString(ReadDataBuffer, 0, recv_package_len);
    //检验新邮件 --需要重新 TCP 连接，否则无法接收到新邮件
    client_sock.Close();//Socket 资源已经释放，要重新构造 Socket 对象
```

```
client_sock = new Socket(AddressFamily.InterNetwork,SocketType.Stream, ProtocolType.Tcp);
client_sock.SetSocketOption(SocketOptionLevel.Socket, SocketOptionName.NoDelay, 1);
client_sock.Blocking = true;
client_sock.Connect(remoteEP);
//POP 服务器会先发送一些响应字符串到客户端，显示客户端登录正常
recv_package_len = client_sock.Receive(ReadDataBuffer, 1024, SocketFlags.None);
recv_str = Encoding.UTF8.GetString(ReadDataBuffer, 0, recv_package_len);
//通知窗体显示收到的字符串
//SendMessage(main_wnd_handle, TRAN_REPLY_INFO, 100, 100);
//send CMD: user send_str = "user
zscleonet\r\n";
//通知窗体显示发出的字符串
SendMessage(main_wnd_handle, TRAN_SEND_INFO, 100, 100);
b_cmd = Encoding.ASCII.GetBytes(send_str);
Array.Clear(SendDataBuffer, 0, 1024);
Array.Copy(b_cmd, SendDataBuffer, b_cmd.Length);
client_sock.Send(SendDataBuffer, b_cmd.Length, SocketFlags.None);
//接收服务器响应
recv_package_len = client_sock.Receive(ReadDataBuffer, 1024, SocketFlags.None);
recv_str = Encoding.UTF8.GetString(ReadDataBuffer, 0, recv_package_len);
//通知窗体显示收到的字符串
//SendMessage(main_wnd_handle, TRAN_REPLY_INFO, 100, 100);
//send CMD: pass
send_str = "pass goodstudent\r\n";
SendMessage(main_wnd_handle, TRAN_SEND_INFO, 100, 100);
b_cmd = Encoding.ASCII.GetBytes(send_str);
Array.Clear(SendDataBuffer, 0, 1024);
Array.Copy(b_cmd, SendDataBuffer, b_cmd.Length);
client_sock.Send(SendDataBuffer, b_cmd.Length, SocketFlags.None);
//接收服务器响应
recv_package_len = client_sock.Receive(ReadDataBuffer, 1024, SocketFlags.None);
recv_str = Encoding.UTF8.GetString(ReadDataBuffer, 0, recv_package_len);
//通知窗体显示收到的字符串
//SendMessage(main_wnd_handle, TRAN_REPLY_INFO, 100, 100);
send_str = "list\r\n";
SendMessage(main_wnd_handle, TRAN_SEND_INFO, 100, 100);
b_cmd = Encoding.ASCII.GetBytes(send_str);
Array.Clear(SendDataBuffer, 0, 1024);
Array.Copy(b_cmd, SendDataBuffer, b_cmd.Length);
```

```
client_sock.Send(SendDataBuffer, b_cmd.Length, SocketFlags.None);
//接收服务器响应
recv_package_len = client_sock.Receive(ReadDataBuffer, 1024, SocketFlags.None);
recv_str = Encoding.UTF8.GetString(ReadDataBuffer, 0, recv_package_len);
result_num = recv_str.Split("\r\n ".ToCharArray(), StringSplitOptions.RemoveEmptyEntries);
new_total_mail_len = Int32.Parse(result_num[2]);
mailbox_str = " 你邮件个数是" + result_num[1];
SendMessage(main_wnd_handle, MAIL_COUNT, 100, 100);
if (new_total_mail_len != old_total_mail_len)
{ old_total_mail_len = new_total_mail_len;
   //send CMD: retr 1 Must handle more than 1024 bytes
   send_str = "retr 1\r\n";
   SendMessage(main_wnd_handle, TRAN_SEND_INFO, 100, 100);
   b_cmd = Encoding.ASCII.GetBytes(send_str);
   Array.Clear(SendDataBuffer, 0, 1024);
   Array.Copy(b_cmd, SendDataBuffer, b_cmd.Length);
   client_sock.Send(SendDataBuffer, b_cmd.Length, SocketFlags.None);
   //接收服务器响应 --邮件本身大小
   //+OK 835 octects
   recv_package_len = client_sock.Receive(ReadDataBuffer, 1024, SocketFlags.None);
   recv_str = Encoding.UTF8.GetString(ReadDataBuffer, 0, recv_package_len);
   //SendMessage(main_wnd_handle, TRAN_REPLY_INFO, 100, 100);
   str_num = new Regex(@"\D+(?<num_val>\d+)\D+");
   m = str_num.Match(recv_str);
   mail_len = Int32.Parse(m.Groups["num_val"].Value);
   recv_str = "要接收的邮件长度为" + mail_len.ToString() + " 字节  \r\n";
   SendMessage(main_wnd_handle, TRAN_REPLY_INFO, 100, 100);
   //通知窗体显示收到的邮件长度
   //Mail body
   ms_recv.Seek(0, SeekOrigin.Begin);
   do
   { recv_package_len = client_sock.Receive(ReadDataBuffer, 1024, SocketFlags.None);
      ms_recv.Write(ReadDataBuffer, 0, recv_package_len);
      recv_octects += recv_package_len;
   }
   while (recv_octects < mail_len);
   //通知窗体显示收到的字符串
   SendMessage(main_wnd_handle, TRAN_MAIL_CONTENT, 100, 100);
   //Mail body }
```

```
    Thread.Sleep(5000);
} while (MRE_check_end.WaitOne(1)==false);
//循环体外要补全邮件服务的退出命令：
//send CMD: quit send_str = "quit\r\n";
SendMessage(main_wnd_handle, TRAN_SEND_INFO, 100, 100);
b_cmd = Encoding.ASCII.GetBytes(send_str);
Array.Clear(SendDataBuffer, 0, 1024);
Array.Copy(b_cmd, SendDataBuffer, b_cmd.Length);
client_sock.Send(SendDataBuffer, b_cmd.Length, SocketFlags.None);
//接收服务器响应
recv_package_len = client_sock.Receive(ReadDataBuffer, 1024, SocketFlags.None);
recv_str = Encoding.UTF8.GetString(ReadDataBuffer, 0, recv_package_len);
//通知窗体显示收到的字符串
SendMessage(main_wnd_handle, TRAN_REPLY_INFO, 100, 100);
```

程序的调试需要提供 126 邮箱的用户名和密码，运行本程序，并向指定的邮件地址发邮件，程序自动显示新邮件的内容。刚接触 POP3 协议时，首先要理解的是 POP3 的命令，在设计程序时要掌握服务器的返回信息的文本格式，掌握了客户端与服务器信息的格式，只要编写 TCP 应用程序即可实现本小节任务。

10.5 实验作业

1. 调试程序实现邮件的自动检测和下载。