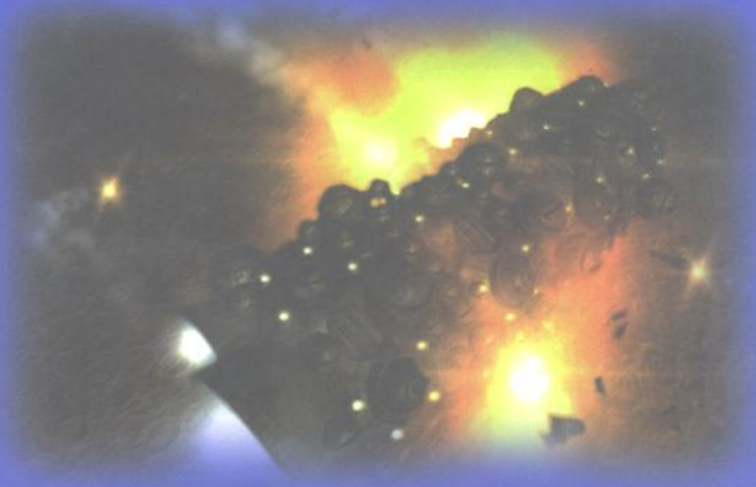
计算机开发与制作实例丛书



即插即用
即学即会

C++ Builder

网络开发实例

清汉计算机工作室　编著

机械工业出版社
China Machine Press

计算机开发与制作实例丛书

C++ Builder 网络开发实例

清汉计算机工作室　编著

机 械 工 业 出 版 社

本书采用实例的形式向读者介绍利用 C++ Builder 进行网络开发的方法和技巧。每个实例都包括主要内容、实例过程、实例详解和注释三个部分。书中实例丰富，覆盖网络开发面广，主要包括：开发浏览器类应用程序、开发发送和接收电子邮件的系统、开发上载和下载文件的应用程序、开发网上聊天系统、创建 Web 服务器应用程序、映射驱动器等。并且在讲述实例的同时，概述了有关网络的一些基础知识，有助于读者理解网络工作原理及理解本书的实例。

本书要求读者应懂得 C++，并且具有 C++ Builder 的基本知识。可供广大程序员、大专院校师生、计算机爱好者和培训班学员参考使用。

机械工业出版社（北京市百万庄大街 22 号 邮政编码 100037）
责任编辑：边　萌　何月秋　　封面设计：姚　毅
责任印制：何全君
三河市宏达印刷厂印刷·新华书店北京发行所发行
2000 年 1 月第 1 版第 1 次印刷
787mm × 1092mm 1/16　31.25 印张·768 千字
0 001—5 000 册
定价：53.00 元　（ICD，含配套书）

新出音管［1999］438 号
ISBN 7 – 980039 – 73 – 4/TP·18

前　言

随着我国计算机应用的普及与发展，各种不同用途的应用软件不断被开发出来，并迅速为人们掌握和使用。目前，除了专业软件开发人员外，许多一般的计算机用户也已经开始自己动手开发一些应用程序。计算机的应用开发工作包括很多方面，例如：图形图像处理开发、动画制作、专项应用程序开发、通用应用程序开发、游戏软件的制作等等。开发这些项目需要使用各种各样的开发工具，以便能够大大地缩短开发周期并减少开发费用。因此，很多从事计算机应用开发工作的人员迫切需要了解和掌握各种计算机编程和制作的方法。但目前市场上的这类图书主要是介绍开发工具的使用，并没有大量的应用实例与之配套，这与只有课本而没有习题集的情况类似。为适应广大读者在学习软件的同时需要参考大量实例的要求，我们特地组织了这套“计算机开发与制作实例丛书”。

这套丛书主要是针对应用程序开发者而编写的，它以各种实例的方式，向读者介绍了计算机编程和制作方面的知识，力求使读者能够轻松地学习开发的方法并熟练地掌握、使用相关的开发工具。对于需要编制应用程序或利用计算机进行开发的用户，这套丛书不仅是一个开发指南，而且还给出了其他图书很少涉及的开发技巧。通过本丛书中介绍的一个个具体的例子，用户便可以比较容易地掌握各种开发软件的功能和使用技巧。

这套丛书力求通过实例来介绍开发工具的功能，由浅入深地讲述编程和开发内容，每个实例都包括：目标和要点、实现步骤、归纳和注释、小结或练习几大部分，不仅适用于那些刚刚从事编程或开发制作的计算机使用者，并且对已经有较多开发经验的计算机用户也同样有较大的帮助。易学易用且实用性强是本丛书的特点，书中每一个例子里的程序或方法，读者都可以直接引用，相信会使广大读者受益匪浅。

这套丛书目前拟包括以下内容：

1．Office 等办公软件实例（Word、Excel、PowerPoint 和 Access、Microsoft Project98、Microsoft Money98 等）；

2．Microsoft Developer Studio 实例（包括 VC、VB、VFP、VJ、ActiveX）；

3．多媒体实例（包括 CorelDRAW、PhotoShop、3DMAX、Director、Authorware、Freehand、Illustrate 等）；

4．网页制作实例（包括 FrontPage、Pagemaker、Net Object Fusion、VBScript、JavaScript、ASP 等）；

5．图形开发实例（例如 OpenGL）；

6．数据库及网络开发实例（例如 PowerBuilder、Oracle、Sybase）。

目前本丛书共计 22 本，随着软件版本的不断更新和新型软件的出现，我们还将不断充实更新这套丛书。

清汉计算机工作室

编 者 的 话

自从第一台 IBM PC 面世以来，Internet 又成为计算机世界最大的事情。那些过去只有研究人员和硬件核心技术人员才熟悉的东西，今天已成为普通人的基本知识。在人们的日常谈话中，也时不时地说及 Internet、Jave 和冲浪等等许多网络用语。同时，各种网络应用程序非常有前途，如果读者熟悉网络编程，那么就在一个正在扩展的市场中拥有一项很有市场的技能。

C++ Builder 提供了一套由 NetMasters 公司开发的简便易用的控件。这套控件不仅提供了一种网络环境下进行编程的简单方法，而且简单易用，节省了大量查阅 API 函数和帮助文件的时间。而在此之前，希望编写 TCP/IP 程序的程序员不得不使用 Win32 下的 WinSock API 函数。

本书中不仅介绍了如何使用上述这套控件创建网络应用程序，同时也介绍了如何使用 WinSock API、WinInet API、NetBIOS 编写网络应用程序，以弥补 NetMasters 公司提供控件的不足。

本书中的实例包括：网络测试应用程序，用于测试网络接通与否；自制小电子邮件收发系统，用于收发电子邮件；网络浏览器，用于创建浏览器；用户帐号查询应用程序，用于查询服务器上的用户帐号；阅读和张贴网络新闻系统，用于浏览、阅读和张贴消息；文件传输应用程序，实现与服务器文件的发送与接收；发送和接收普通数据应用程序，通过网络发送和接收用户数据；使用代理服务器应用程序，是一个能使用代理服务器的浏览器；数据编码和解码，完成数据的编码和解码；发送用户数据报消息应用程序；创建一个接收和发送 UDP 数据的应用程序；网上对话应用程序；网上聊天应用程序，是一多用户的聊天应用程序；基本 CGI 应用程序；基本 ISAPI 应用程序；公用网关接口程序，是一能传输数据库中数据的 CGI 应用程序；使用 WININET 创建 FTP 应用程序，学习使用 WININET 创建 FTP 客户程序；获取主机 IP 地址；获取主机 MAC 地址；获取主机名；活用 TClientSocket 控件；通过调制解调器进行话音拨号；通过串口发送和接收数据；映射和断开网络驱动器等 29 个实例。

由于编者时间关系和水平有限，书中难免出错，希望读者谅解和批评指正。

编　　者

目　录

实例·1　网络测试应用程序

目标和要点

实例目标

在本实例中，利用 NetMaster 因特网控件中的 NMDayTime、NMEcho 和 NMTime 三个控件，创建一个能测试网络连通性的应用程序。

在 Windows 系统中，为测试网络的连通性，我们常常利用 Windows 系统自带的 Ping.exe 控制台应用程序。PING 命令是属于 ICMP 协议规定的，而 ICMP 是内嵌于 IP 层的，因此，可以说，PING 是网络层的命令。所以网络层是没有端口号这个概念的，端口号是 TCP 的概念。请读者千万要注意这一点。

PING 的实现过程很简单，命令将引发 IP 层发送一个简单的 IP 包，一般是 32 个字节。而目的方收到这个包之后，将源和目的地址做一下交换，重新发出这个包即可，当然还要加一些超时的机制。

图 1-1 是 Ping.exe 运行的结果。

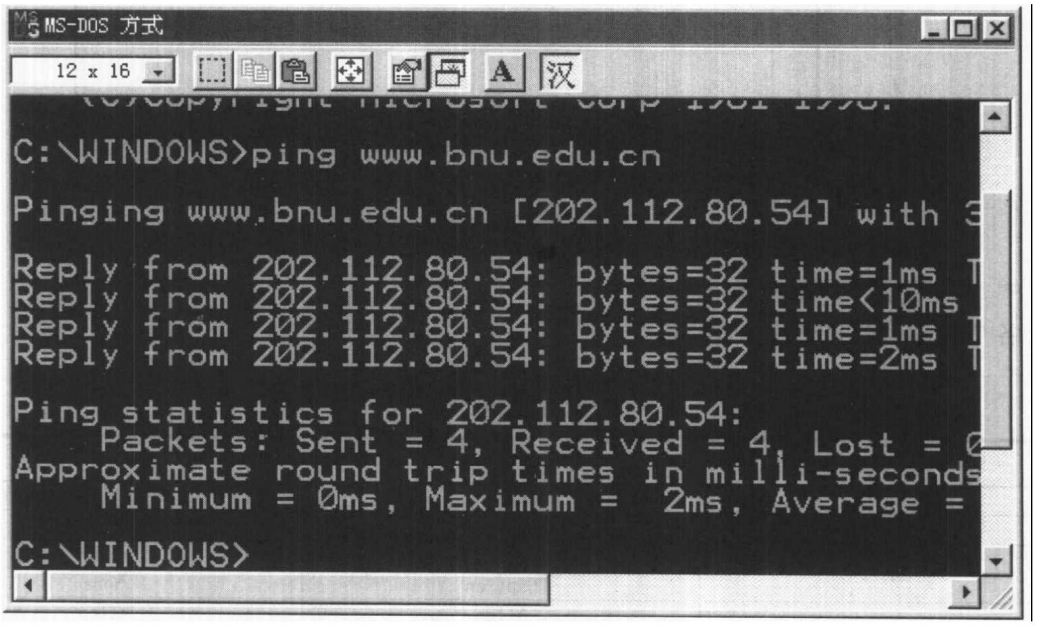


图 1-1　Ping.exe 运行的结果

技术要点

本实例的技术要点主要包括：

1．NMDayTime 控件及其使用

2．NMEcho 控件及其使用
3．NMTime 控件及其使用
4．有关网络的一些基本知识

实 现 步 骤

创建主窗体

选择 File | New Application 创建一个新的空项目文件。加入的控件及其排列如图 1-2 所示。

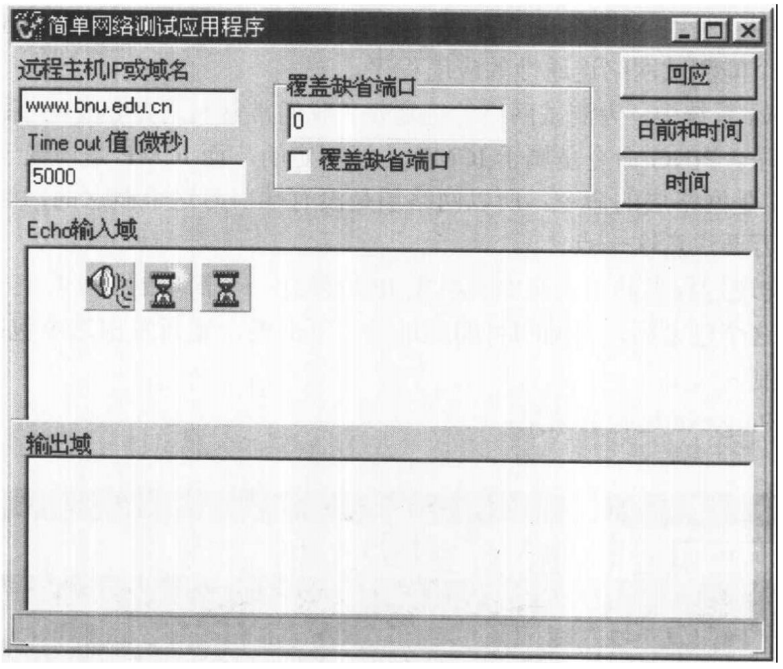


图 1-2 网络测试程序的各个控件及其排列

设定窗体和控件属性

各个控件属性的设置见表 1-1。

表 1-1 各个控件属性的设置

控 件	属 性	属 性 值
Form1	Caption	简单网络测试应用程序
GroupBox1	Caption	覆盖缺省端口
Panel1	Align	alTop
	Caption	None
Panel2	Align	alTop
	Caption	None
Panel3	Align	alClient
	None	None
StatusBar1	SimplePanel	true
Label1	Caption	远程主机 IP 或域名
Label2	Caption	Time out 值 (微秒)
Label3	Caption	Echo 输入域
Label4	Caption	输出域

（续）

控　件	属　性	属 性 值
Edit1	Name	Host
	Text	www bnu edu cn
Edit2	Name	TimeOut
	Text	50000
Edit3	Name	Port
	Text	0
CheckBox1	Name	Override
	Caption	覆盖缺省端口
RichEdit1	Name	Input
	Lines	None
RichEit2	Name	Output
	Lines	None
Button1	Name	BtEcho
	Caption	回应
Button2	Name	BtDayTime
	Caption	日前和时间
Button3	Name	BtTime
	Caption	时间

☞ **编写代码**

切换到 Unit1.cpp 中，加入如下所示的一些代码：

```
//---------------------------------------------------------------------------
#include <vcl h>
#pragma hdrstop

#include "DebugForm.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "* dfm"
TForm1 *Form1;
//---------------------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
    : TForm(Owner)
{
}
//---------------------------------------------------------------------------
void __fastcall TForm1::BtEchoClick(TObject *Sender)
{
    NMEcho1->Host = Host->Text;
    NMEcho1->TimeOut = StrToInt(TimeOut->Text);
    if(Override->Checked)
        NMDayTime1->Port = StrToInt(Port->Text);
```

```
    else
        NMDayTime1->Port = 7;
    NMEcho1->ReportLevel = Status_Basic;
    // 建立连接
    NMEcho1->Connect();
    Output->Clear();
    for(int i = 0; i < Input->Lines->Count, i++)
        Output->Text = Output->Text + NMEcho1->Echo(Input->Lines->Strings[i]);
    NMEcho1->Disconnect();
}
//---------------------------------------------------------------------------
void __fastcall TForm1::BtDayTimeClick(TObject *Sender)
{
    //设置标准的 TCP/IP 属性
    NMDayTime1->Host = Host->Text;
    NMDayTime1->TimeOut = StrToInt(TimeOut->Text);

    if(Override->Checked)
        NMDayTime1->Port = StrToInt(Port->Text);
    else
        NMDayTime1->Port = 13;
    NMDayTime1->ReportLevel = Status_Basic;
    //建立连接并接收字符串
    Output->Text = "The Current Date and Time is : "+NMDayTime1->DayTimeStr;
}
//---------------------------------------------------------------------------
void __fastcall TForm1::BtTimeClick(TObject *Sender)
{
    //设置标准的 TCP/IP 属性
    NMTime1->Host = Host->Text;
    NMTime1->TimeOut = StrToInt(TimeOut->Text),
    if(Override->Checked)
        NMTime1->Port = StrToInt(Port->Text);
    else
        NMTime1->Port = 37;
    NMTime1->ReportLevel = Status_Basic;
    //建立连接并发送字符串
    Output->Text = "The Current Date and Time is : "+NMTime1->TimeStr;
}
```

```
//---------------------------------------------------------------------------
void __fastcall TForm1::NMEcho1Connect(TObject *Sender)
{
 StatusBar1->SimpleText="Echo has connected host";
}
//---------------------------------------------------------------------------
void __fastcall TForm1::NMEcho1ConnectionFailed(TObject *Sender)
{
 ShowMessage("Connection failed");
}
//---------------------------------------------------------------------------
void __fastcall TForm1::NMEcho1Disconnect(TObject *Sender)
{
 StatusBar1->SimpleText="Echo has disconnected";
}
//---------------------------------------------------------------------------
void __fastcall TForm1: NMEcho1HostResolved(TComponent *Sender)
{
 StatusBar1->SimpleText="Host resolved";
}
//---------------------------------------------------------------------------
void __fastcall TForm1: NMEcho1InvalidHost(bool &Handled)
{
 AnsiString TmpStr;
 if(InputQuery("Invalid Host!", "Specify a new host:", TmpStr))
    {
     NMEcho1->Host = TmpStr,
     Handled = true;
    }
}
//---------------------------------------------------------------------------
void __fastcall TForm1::NMDayTime1InvalidHost(bool &Handled)
{
   AnsiString TmpStr;
   if(InputQuery("Invalid Host!", "Specify a new host:", TmpStr))
      {
       NMDayTime1->Host = TmpStr;
       Handled = true;
      }
```

```
}
//---------------------------------------------------------------------------
void __fastcall TForm1::NMDayTime1Connect(TObject *Sender)
{
 StatusBar1->SimpleText="DayTime has connected host";
}
//---------------------------------------------------------------------------
void __fastcall TForm1::NMDayTime1HostResolved(TComponent *Sender)
{
 StatusBar1->SimpleText="Host resolved",
}
//---------------------------------------------------------------------------
void __fastcall TForm1::NMDayTime1ConnectionFailed(TObject *Sender)
{
 ShowMessage("Connection failed");
}
//---------------------------------------------------------------------------
void __fastcall TForm1::NMDayTime1Disconnect(TObject *Sender)
{
 StatusBar1->SimpleText="NMDayTime1 has disconnected";
}
//---------------------------------------------------------------------------
void __fastcall TForm1::NMTime1Connect(TObject *Sender)
{
 StatusBar1->SimpleText="NMTime1 has connected host";
}
//---------------------------------------------------------------------------
void __fastcall TForm1::NMTime1ConnectionFailed(TObject *Sender)
{
 ShowMessage("Connection failed");
}
//---------------------------------------------------------------------------
void __fastcall TForm1::NMTime1Disconnect(TObject *Sender)
{
 StatusBar1->SimpleText="NMTime1 has disconnected";
}
//---------------------------------------------------------------------------
void __fastcall TForm1.::NMTime1HostResolved(TComponent *Sender)
{
```

```
  \
  StatusBar1->SimpleText="Host resolved";
}
//---------------------------------------------------------------------
void __fastcall TForm1 :NMTime1InvalidHost(bool &Handled)
{
  AnsiString TmpStr,
  if(InputQuery("Invalid Host!", "Specify a new host ", TmpStr))
     {
      NMTime1->Host = TmpStr,
      Handled = true;
     }
}
//---------------------------------------------------------------------
```

归纳和注释

本实例介绍了如何利用 NMEcho、NMDayTime 和 NMTime 三个控件，编写测试网络连通性的应用程序。本实例使用的主要技术如下：

- 利用 NMEcho 控件实现网络测试
- 利用 NMDayTime 控件实现网络测试
- 利用 NMDayTime 控件实现网络测试
- Internet 简介

利用 NMEcho 控件实现网络测试

这里是在 BtEcho 按钮控件的 OnClick 事件处理函数中实现 Echo 客户机的。在此事件处理函数中，首先从 Host 编辑框控件中读入用户输入的远程主机地址，并将此值赋给 NMEcho 控件的 Host 属性。然后将用户在 TimeOut 编辑框中输入的超时值赋给 NMEcho 控件的 TimeOut 属性。如果不使用缺省的端口值，将用户的 Port 编辑框中输入的端口值赋给 NMEcho 控件的 Port 属性，否则使用缺省的端口值 7。在设置好 NMEcho 控件的标准 TCP/IP 属性后，调用此控件的 Connect 方法，建立与 Echo 服务器的连接，将用户在 Input 控件中输入的内容发送给服务器。当服务器接收到客户发送来的数据以后，它将这些数据发送回给客户机，这时我们将这些数据显示在 Output 控件中。具体实现代码如下：

```
//---------------------------------------------------------------------
void __fastcall TForm1. BtEchoClick(TObject *Sender)
{
    //设置 NMEcho 控件的标准的 TCP/IP 属性
    NMEcho1->Host = Host->Text,
    NMEcho1->TimeOut = StrToInt(TimeOut->Text),
    if(Override->Checked)
        NMDayTime1->Port = StrToInt(Port->Text),
```

```
    else
        NMDayTime1->Port = 7;
    NMEcho1->ReportLevel = Status_Basic;
    // 建立连接
    NMEcho1->Connect();
    Output->Clear();
    for(int i = 0; i < Input->Lines->Count; i++)
        Output->Text = Output->Text + NMEcho1->Echo(Input->Lines->Strings[i]);
    NMEcho1->Disconnect();
}
//---------------------------------------------------------------------------
```

NMEcho 控件实现了 Echo 协议。当 TCP/IP 上用户建立与 Echo 服务器之间的连接后，所有发送给服务器的数据都被直接发送回给客户端。UDP 和 TCP 协议缺省的 Echo 端口都是 7。

要注意的是，在调用 NMEcho 控件的 Connect 方法时，应该确保在接收数据之前连接已经建立。如果连接失败，控件将自动抛出异常，但是由于此异常由窗体之外的事件捕捉，所以程序仍然可以正常运行。

当调用 NMEcho 控件的 Connect 方法后，如果用户输入的是域名地址，而不是 IP 地址，并且域名服务器成功地解析了此域名，将触发此控件的 OnHostResolved 事件，在此事件处理函数中，我们在状态条上显示已成功解析域名的信息给用户。具体代码如下：

```
//---------------------------------------------------------------------------
void __fastcall TForm1::NMEcho1HostResolved(TComponent *Sender)
{
 StatusBar1->SimpleText="Host resolved";
}
//---------------------------------------------------------------------------
```

如果用户输入的远程主机不正确，将触发此控件的 OnInvalidHost 事件。在此事件处理函数中，实例显示一个对话框要求用户重新输入主机的 IP 地址或域名地址，最后将 Handle 参数设置为 true，这将要求重新试图跟服务器建立连接。具体实现代码如下：

```
//---------------------------------------------------------------------------
void __fastcall TForm1::NMEcho1InvalidHost(bool &Handled)
{
 AnsiString TmpStr;
 if(InputQuery("Invalid Host!", "Specify a new host:", TmpStr))
    {
     NMEcho1->Host = TmpStr;
     Handled = true;
    }
}
```

```
//---------------------------------------------------------------------------
```

如果跟服务器建立了连接，将触发 NMEcho 控件的 OnConnect 事件。在此事件的处理函数中，我们在状态条上显示一条已经和服务器建立连接的信息给用户。具体代码如下：

```
//---------------------------------------------------------------------------
void __fastcall TForm1::NMEcho1Connect(TObject *Sender)
{
 StatusBar1->SimpleText="Echo has connected host";
}
//---------------------------------------------------------------------------
```

如果在调用 Connect 方法后，在超时时间到达时仍然没有根 Echo 服务器建立连接，这时将触发此控件的 OnConnectFailed 事件，在此事件的处理函数中，向用户显示一个连接失败的消息。具体实现代码如下：

```
//---------------------------------------------------------------------------
void __fastcall TForm1::NMEcho1ConnectionFailed(TObject *Sender)
{
 ShowMessage("Connection failed");
}
//---------------------------------------------------------------------------
```

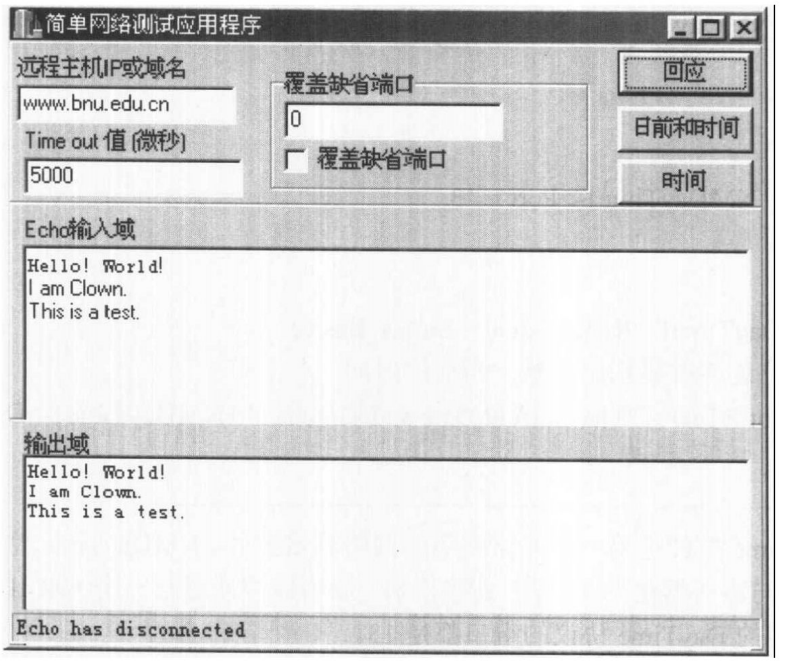


图 1-3　Echo 网络测试程序运行的结果

当 Echo 服务器和客户机建立连接后，将客户机发送来的数据重新发送回给客户机，然后断开与客户机的连接，这时将触发 Echo 控件的 OnDisconnect 事件，在此事件的处理函数中，在状态条上显示一与服务器断开连接的信息给用户。具体实现代码如下：

```
//---------------------------------------------------------------------------
void __fastcall TForm1 .NMEcho1Disconnect(TObject *Sender)
{
 StatusBar1->SimpleText="Echo has disconnected",
}
//---------------------------------------------------------------------------
```

此段代码运行的结果如图 1-3 所示。

利用 NMDayTime 控件实现网络测试

在 BtDayTime 按钮控件的 OnClick 事件处理函数中，实现从服务器上获取当前日期和时间的实现代码如下：

```
//---------------------------------------------------------------------------
void __fastcall TForm1. BtDayTimeClick(TObject *Sender)
{
    // 设置 NMDayTime 控件的标准 TCP/IP 属性
    NMDayTime1->Host = Host->Text,
    NMDayTime1->TimeOut = StrToInt(TimeOut->Text);

    if(Override->Checked)
      {
        NMDayTime1->Port = StrToInt(Port->Text),
      }
    else
       {
         NMDayTime1->Port = 13;
       }

    NMDayTime1->ReportLevel = Status_Basic,
    //建立连接并显示服务器上当前的时间
    Output->Text = "The Current Date and Time is  "+NMDayTime1->DayTimeStr,
}
//---------------------------------------------------------------------------
```

NMDayTime 控件是另一个非常有用的网络测试控件。NMDayTime 控件实现了 DayTime 协议。在客户与服务器建立 TCP/IP 连接以后，调用该控件返回一个 ASCII 字符串。对于 TCP 和 UDP，缺省的 DayTime 协议的端口都是 13。

在上述代码中，首先将用户在 Host、Port、TimeOut 编辑框控件中输入的数据拷贝给 NMDayTime 控件的 Host、Port 和 TimeOut 属性，这些属性对于所有的标准 TCP/IP 协议都是必须的；然后可连接到 DayTime 服务器，并通过访问此控件的 DayTimeStr 属性来获取时间字符串，同时将此字符串显示在 Output 控件中。在这里有些读者可能会问，我们并没有像使用其他网络控件一样，首先调用控件的 Connect 方法和服务器建立连接。对于

NMDayTime 控件和将要讲的 NMTime 控件，与服务器的连接是在使用 DayTimeStr 和 TimeStr 属性的同时自动进行的。

当客户机试图和 DayTime 服务器连接时，首先触发 NMDayTime 控件的 OnHostResolved 事件，或者如果用户输入的远程主机地址不正确将触发此控件的 OnInvalidHost 事件。如果和服务器建立了连接，将触发控件的 OnConnect 事件。当服务器将当前时间发送给客户机以后，自动和客户机断开连接，这时将触发此控件的 OnDisconnect 事件。

这些事件处理函数的实现和前面讲述的 NMEcho 控件很相似，不再赘述，仅给出如下实现代码：

```
//---------------------------------------------------------------------------
//当用户输入的主机地址不正确时将触发此事件
void __fastcall TForm1::NMDayTime1InvalidHost(bool &Handled)
{
  AnsiString TmpStr;

  if(InputQuery("Invalid Host!", "Specify a new host.", TmpStr))
    {
     NMDayTime1->Host = TmpStr,
     //将 Handle 设置为 true 会重新试图和服务器建立连接
     Handled = true,
    }
}
//---------------------------------------------------------------------------
//当和服务器建立起连接时将触发此事件
void __fastcall TForm1::NMDayTime1Connect(TObject *Sender)
{
 StatusBar1->SimpleText="DayTime has connected host";
}
//---------------------------------------------------------------------------
//当成功地将域名地址解析成 IP 地址时触发此事件
void __fastcall TForm1: NMDayTime1HostResolved(TComponent *Sender)
{
 StatusBar1->SimpleText="Host resolved",
}
//---------------------------------------------------------------------------
//当试图和 DayTime 服务器建立连接而失败时触发此事件
void __fastcall TForm1  NMDayTime1ConnectionFailed(TObject *Sender)
{
 ShowMessage("Connection failed"),
}
```