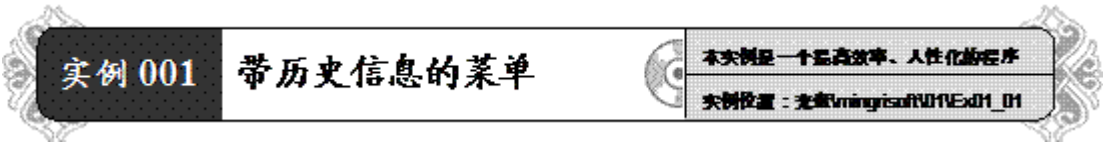
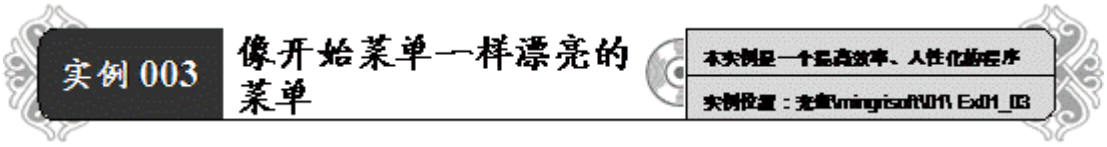
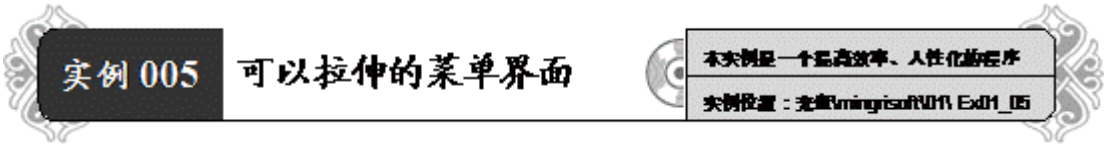
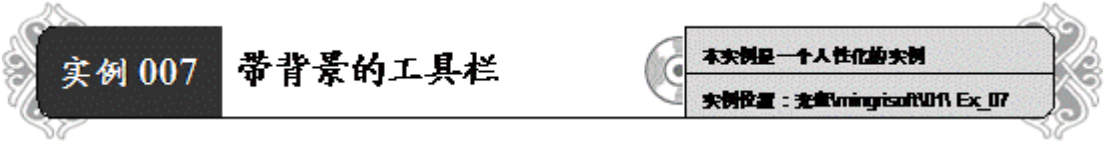
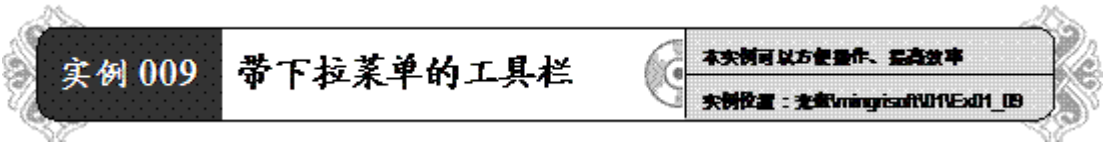
一共 470 多例 winform 界面特效的源码。

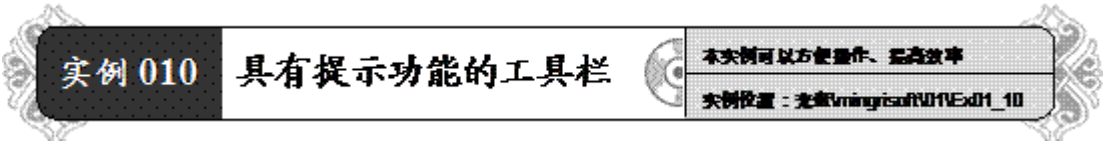


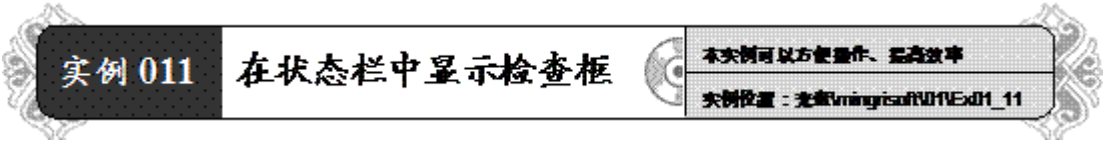


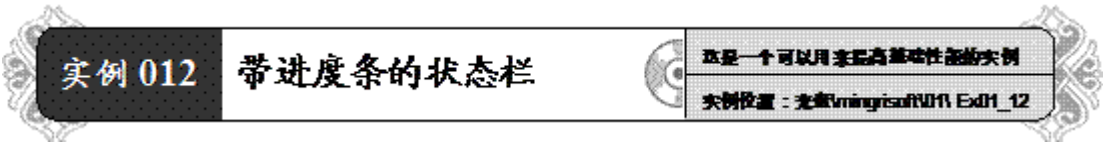


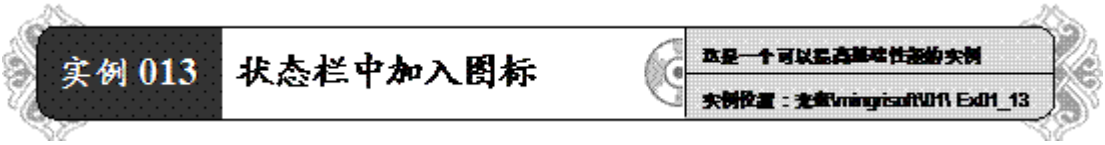


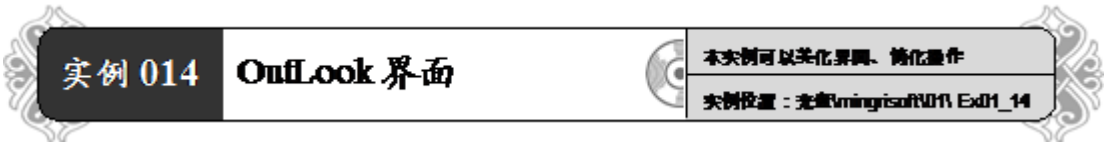


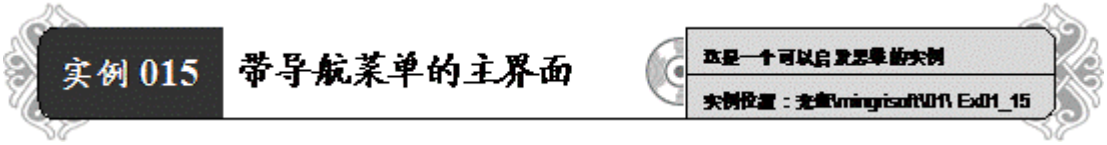


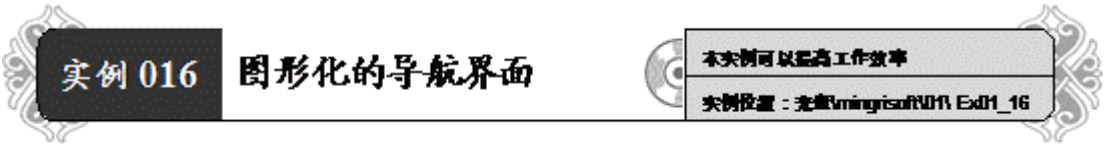


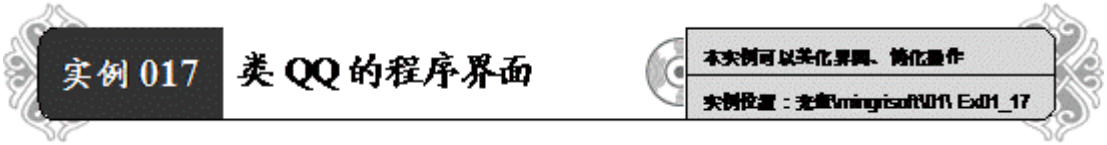


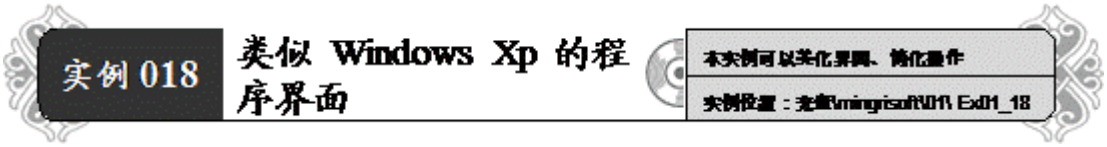


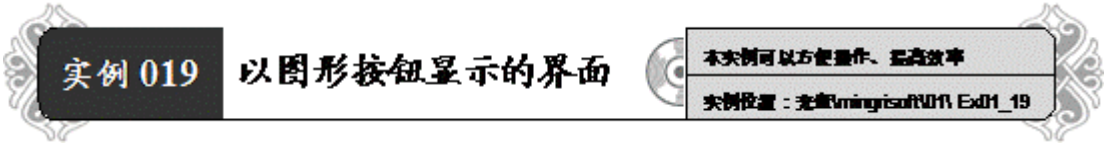


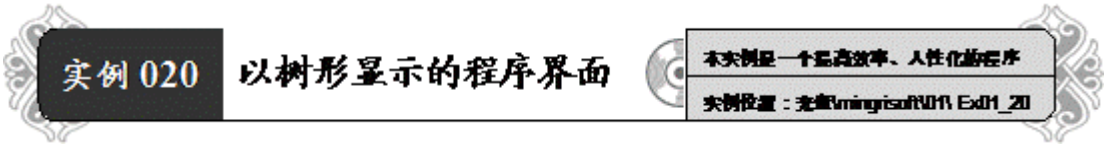


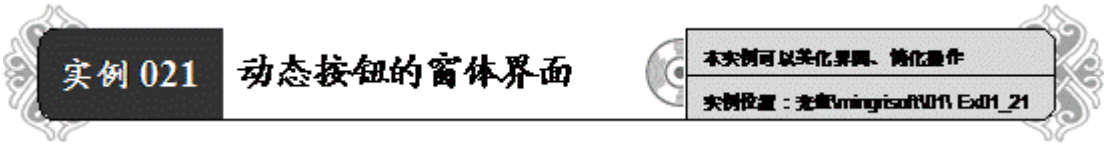


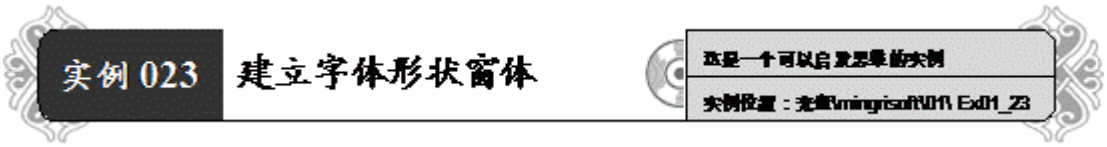


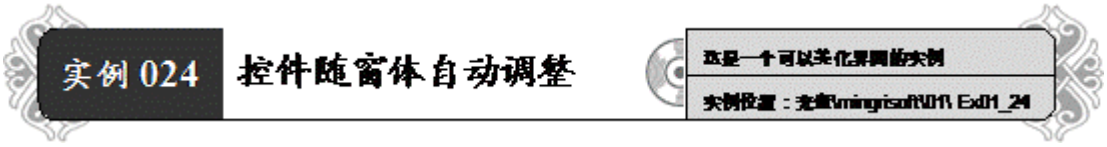


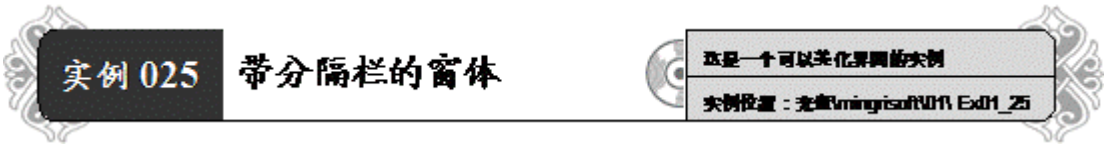


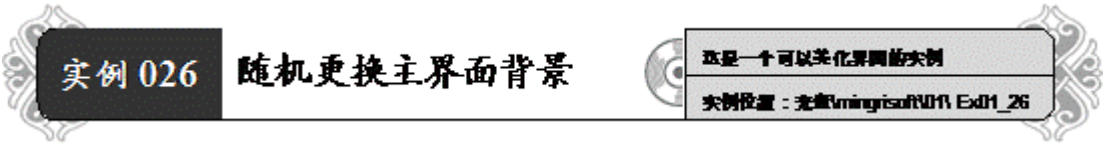


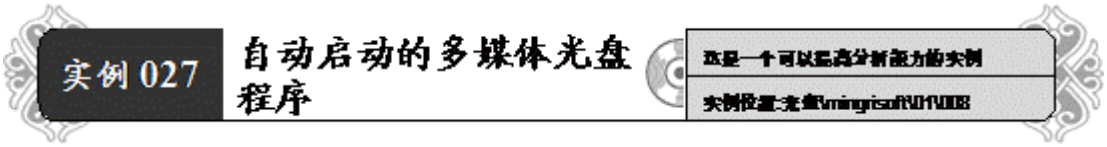


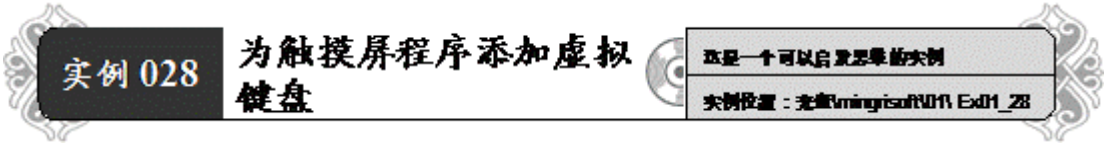


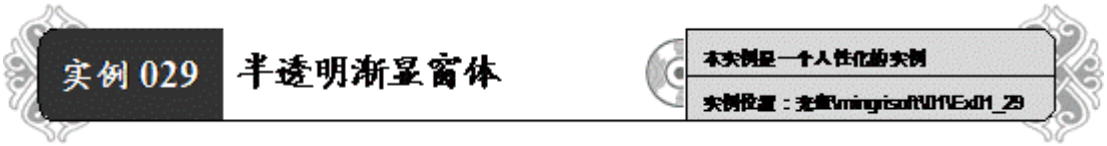


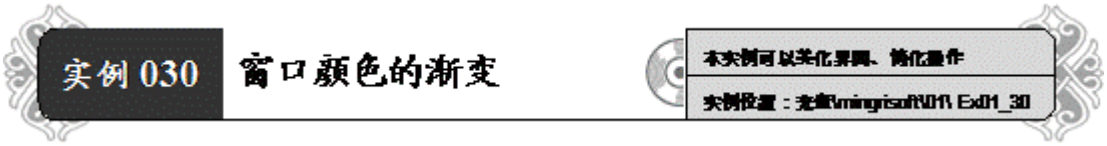


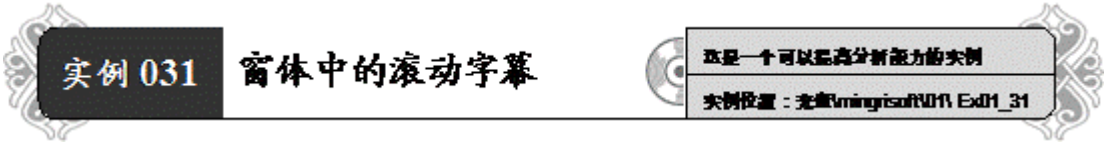


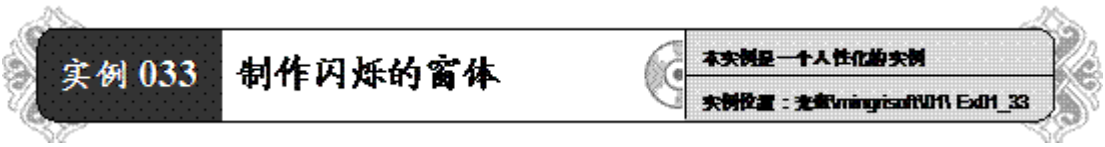


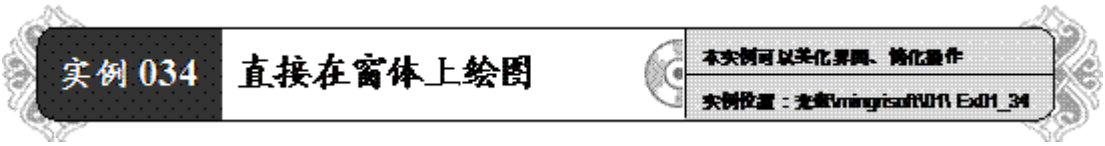


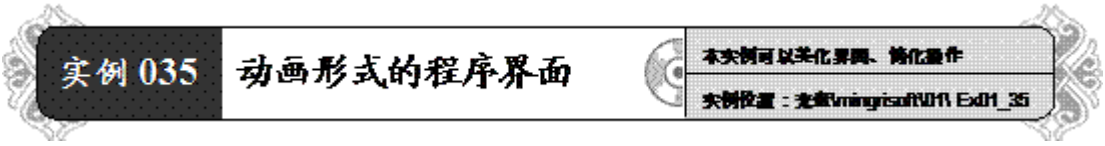


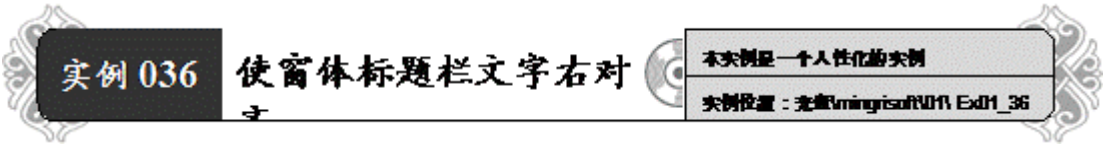


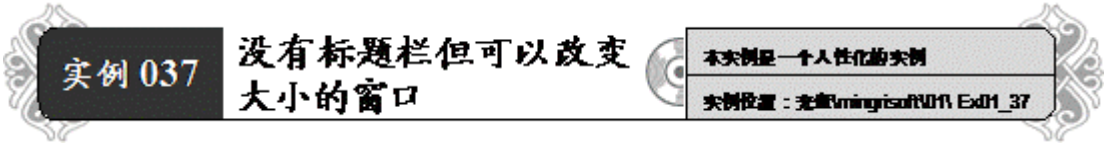


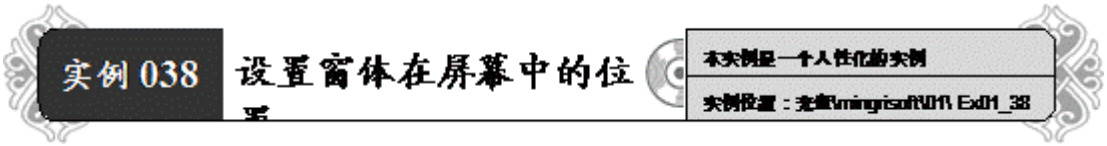


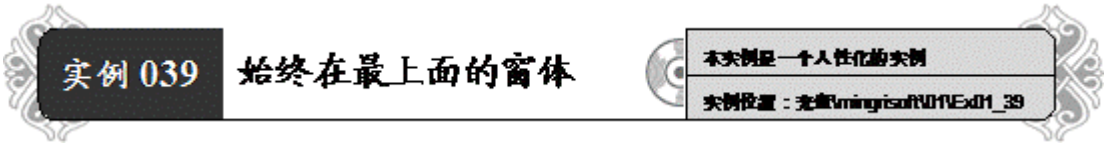


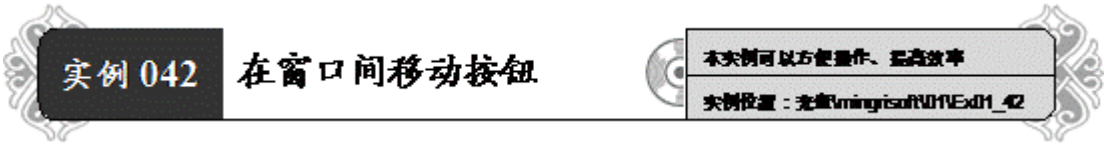


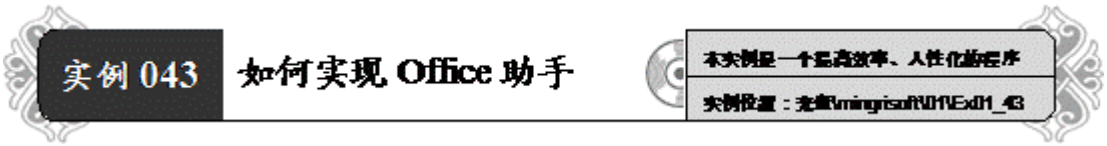


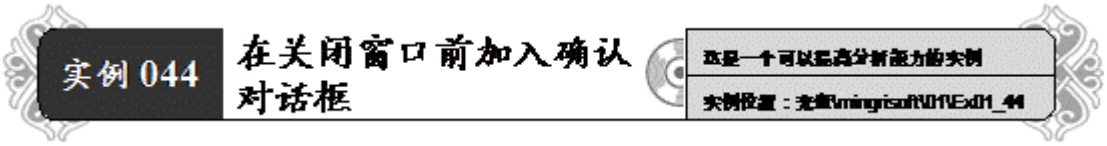


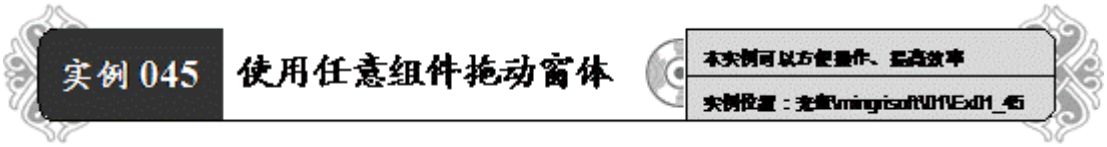


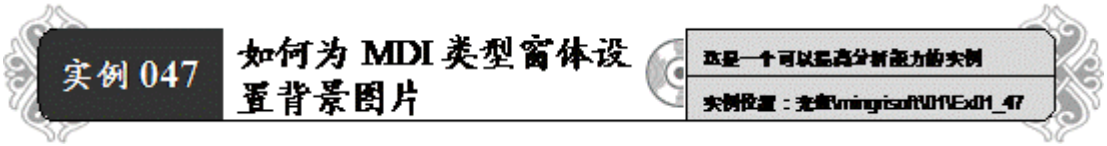


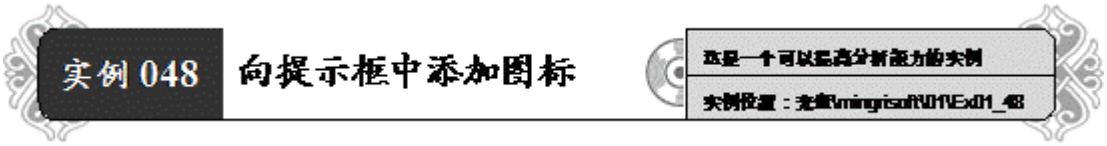


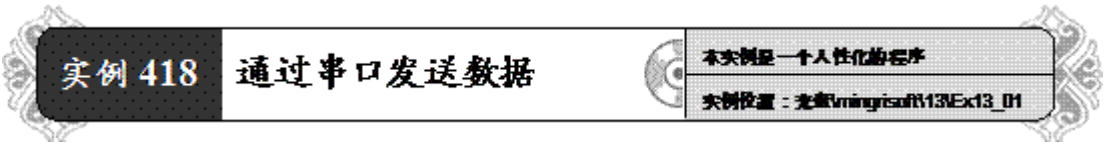


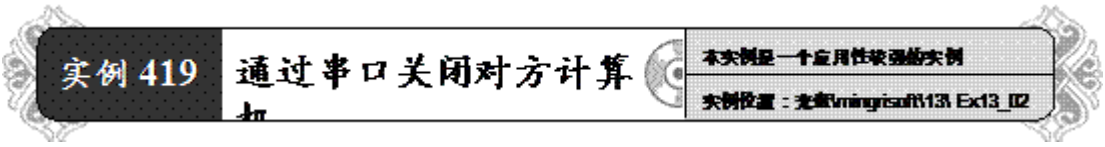


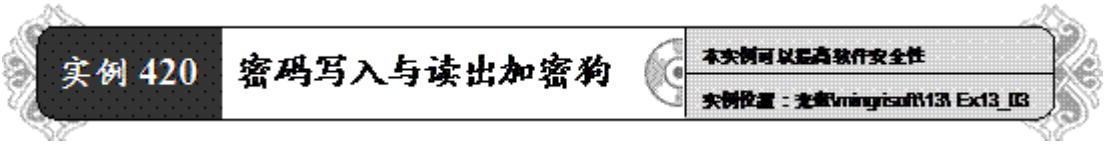


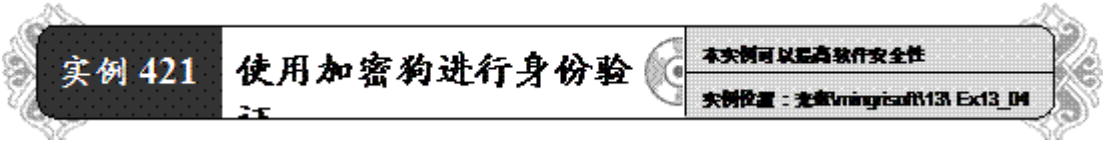


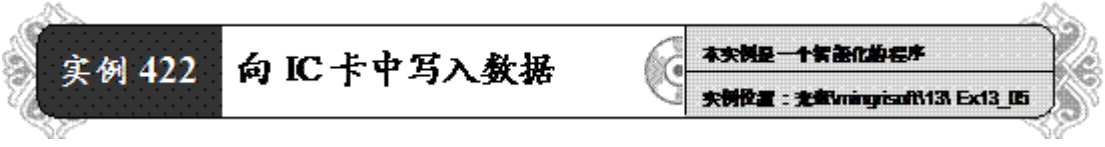


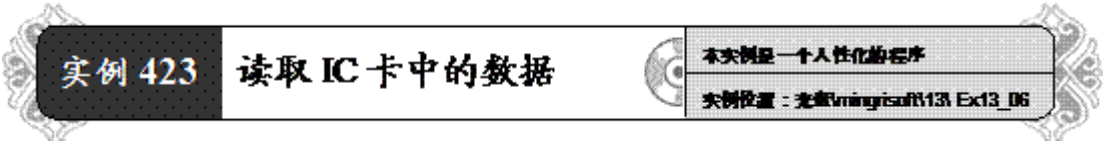


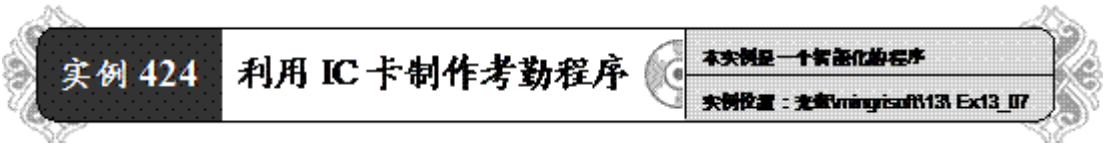


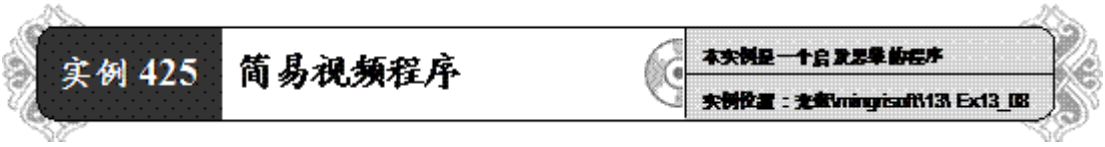


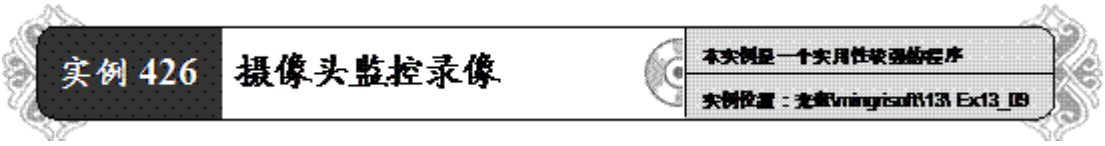


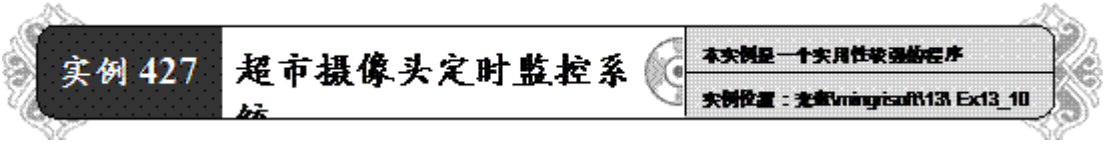


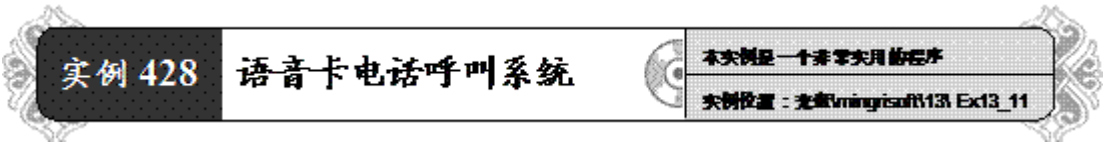


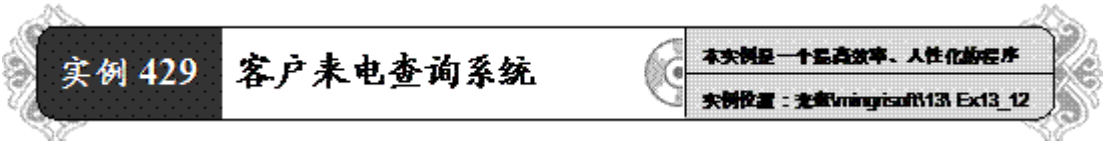


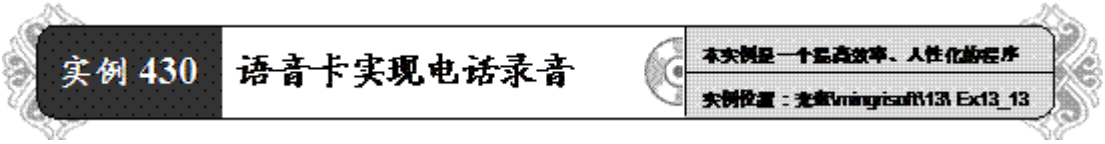


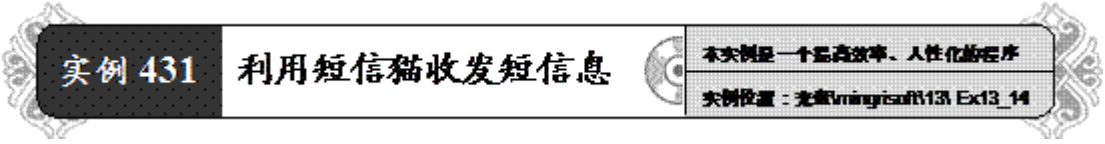


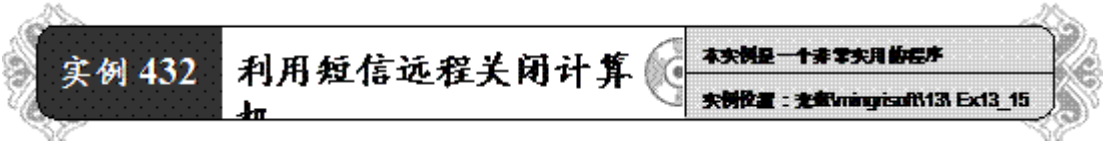


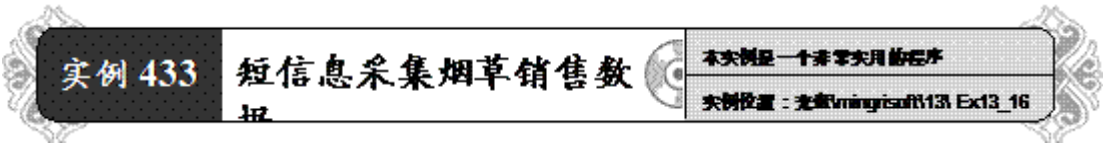


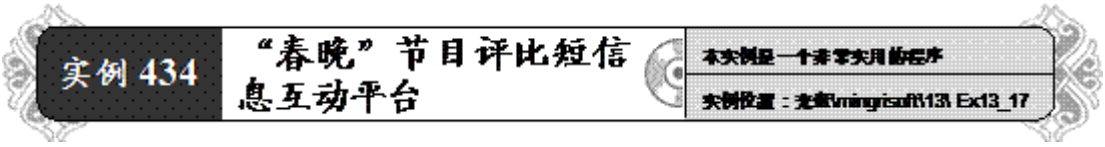


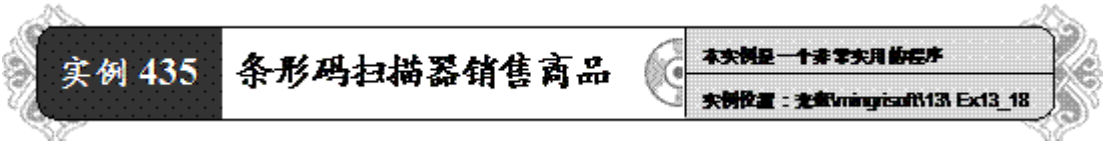


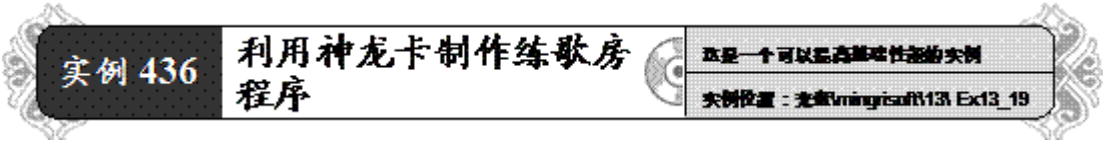


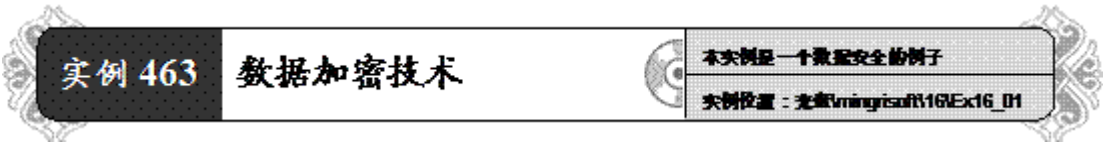


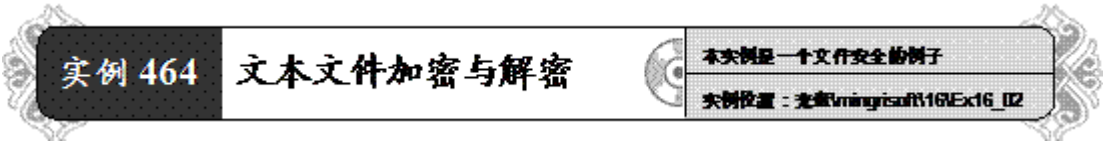


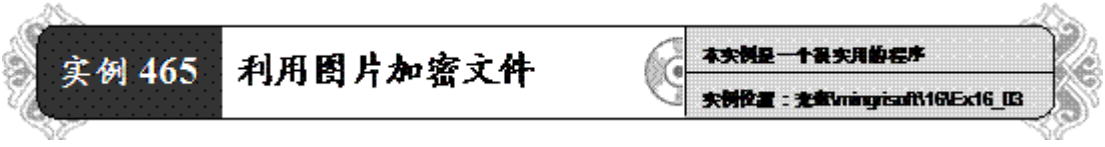


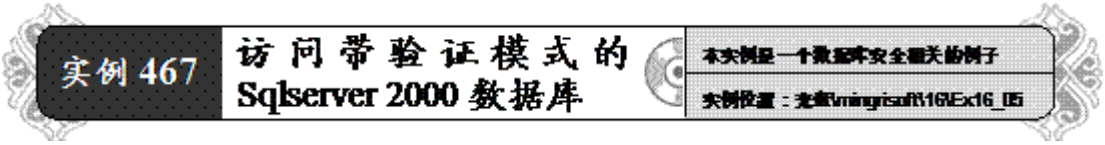


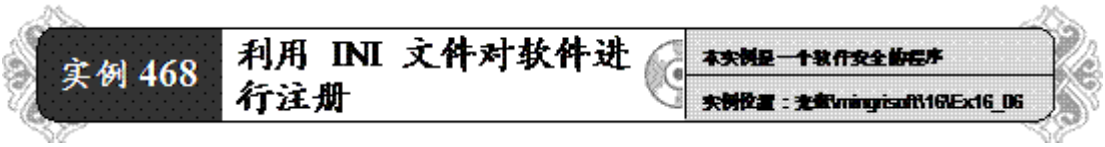


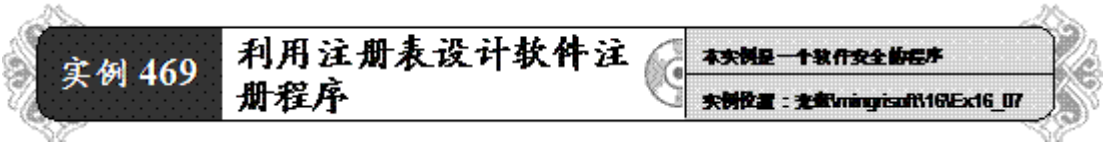


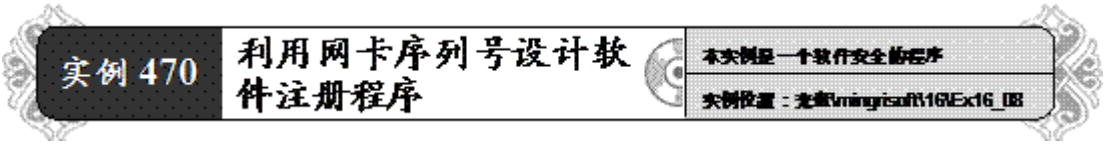


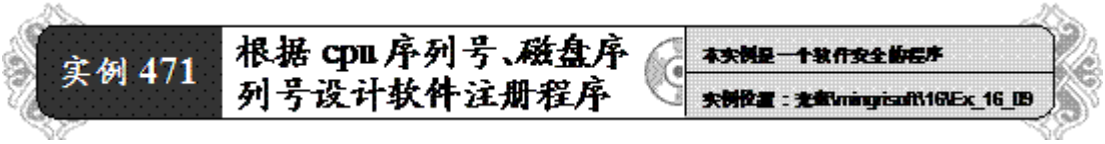












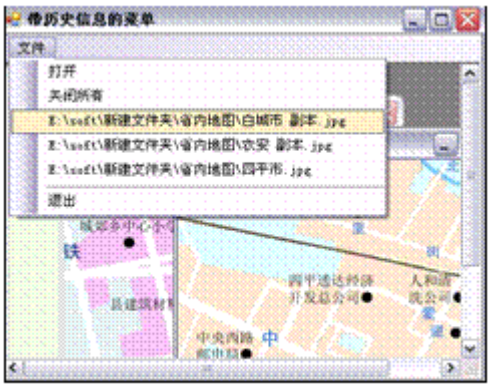
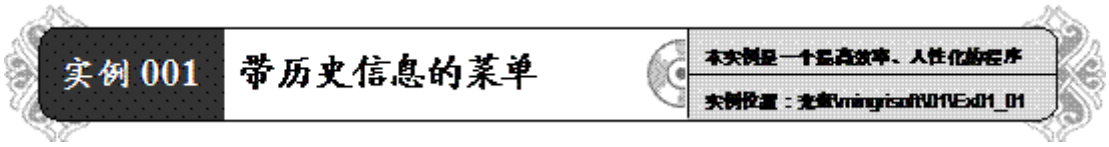


图 1.1 带历史信息的菜单

窗体与界面设计

1.1 菜单应用实例

菜单是程序开发中经常使用的界面元素，合理利用菜单不但可以使用户非常方便的操作程序的功能，更能使效率提高，适应人性化的潮流。下面通过几个应用实例，介绍菜单设计的方法和技术。



实例 001 带历史信息的菜单

实例说明

在开发图纸管理软件时，要求在菜单上记录用户最近打开的档案或图纸，以方便下次使用。如图 1.1 所示，单击“文件”菜单下的“打开文件”子菜单，打开需要查阅的图纸。下次运行该软件时，上次打开的文件名记录到“文件”菜单的历史菜单中，选择该菜单，即可打开相应的图纸文件。

技术要点

要实现保存最近打开的文件，可以将在菜单中最近打开文件的文件名和路径保存到事先建立的*.ini 文件中，系统启动时读取*.ini 中的数据建立数组菜单，即可实现显示历史菜单的功能。

注意：要建立一个带历史信息的菜单，必须首先添加一个 MenuStrip 菜单控件，并将主窗体的 IsMdiContainer 属性设为 True。

实现过程

（1）创建一个项目，将其命名为 Ex01_01，默认窗体为 Form1。

（2）从工具箱中向 Form1 窗体添加 MenuStrip 控件，同时向窗体添加 OpenFileDialog 控件。创建一个“文件”主菜单，在其下面创建打开、关闭所有、退出等菜单选项。

（3）主要程序代码。

将打开文件路径写入 INI 文件的实现代码如下：

```
private void 打开ToolStripMenuItem_Click(object sender, EventArgs e)
{
    openFileDialog1.FileName = "";
    this.openFileDialog1.ShowDialog();
    StreamWriter s = new StreamWriter(address + "\\Menu.ini", true);
    s.WriteLine(openFileDialog1.FileName);//写入 INI 文件
    s.Flush();
    s.Close();
    ShowWindows(openFileDialog1.FileName);
}
```

读取 INI 文件并将信息加入菜单的实现代码如下：

```
private void Form1_Load(object sender, EventArgs e)
{
    StreamReader sr = new StreamReader(address + "\\Menu.ini");
    int i = this.文件ToolStripMenuItem.DropDownItems.Count-2;
    while (sr.Peek()>=0)//读取 INI 文件
    {
        ToolStripMenuItem menuitem = new ToolStripMenuItem(sr.ReadLine());
        this.文件ToolStripMenuItem.DropDownItems.Insert(i, menuitem);
        i++;
        menuitem.Click += new EventHandler(menuitem_Click);
    }
    sr.Close();
```

图 1.2 动态合并菜单

```
    }
```

自定义方法 ShowWindows()用来加载背景图片并显示窗体，实现代码如下：

```
    public void ShowWindows(string fileName)
    {
        Image p = Image.FromFile(fileName);
        Form f = new Form();
        f.MdiParent = this;
        f.BackgroundImage = p;
        f.Show();
    }
```

举一反三

根据本实例，读者可以开发以下程序。

- 记录用户操作菜单日志的程序。在用户单击菜单时，把用户、菜单命令和菜单对应功能写入保存菜单日志的 INI 文件。如果需要查看日志，只需打开 INI 文件。
- 通过数据库保存菜单历史信息的程序。
- 菜单使用频率的程序。把用户使用菜单的数据信息保存到数据库中，然后统计用户使用菜单的频率，并根据此频率调整菜单的显示顺序。

实例 002 菜单动态合并

实例说明

在程序中经常使用弹出菜单，并且一个窗体中可以存在多个弹出菜单。开发过 MDI 窗体的读者可能都知道，当 MDI 子窗体最大化时，子窗体和主窗体的菜单能够自动的合并。这是如何实现的呢？本例实现了将两个弹出菜单动态的合并成一个弹出菜单的功能。实例效果如图 1.2 所示。

技术要点

C# 2.0 中已经将弹出菜单封装为 Context MenuStrip 控件，利用该控件中的 Items 对象可以操作菜单中的菜单项。该对象是 ToolStripMenuItem 类型，使用 Items.AddRange( )方法可以向弹出菜单中添加菜单项，该方法原型如下。

```
public void AddRange (
    ToolStripItem[] toolStripItems
)
```

参数说明如下。

l toolStripItems：控件的数组。

实现过程

（1）创建一个项目，将其命名为 Ex01_02，默认窗体为 Form1。

（2）从工具箱中向 Form1 窗体添加一个 MenuStrip 控件用来设计菜单；同时向窗体添加 ContextMenuStrip 控件用来设计右键菜单；选中 MenuStrip 控件创建一个“打开子窗体”主菜单，然后选中 ContextMenuStrip 控件为其添加子项。

（3）为程序添加一个窗体，默认名为 Form2，同时向窗体添加 ContextMenuStrip 控件用来设计右键菜单，然后选中 ContextMenuStrip 控件为其添加子项。

（4）主要程序代码。

```
private void 打开自窗体 ToolStripMenuItem_Click(object sender, EventArgs e)
{
    Form2 f = new Form2();
    f.MdiParent = this;
    f.Show();//显示子窗体
    f.Resize += new EventHandler(f_Resize);
}
void f_Resize(object sender, EventArgs e)
{
```

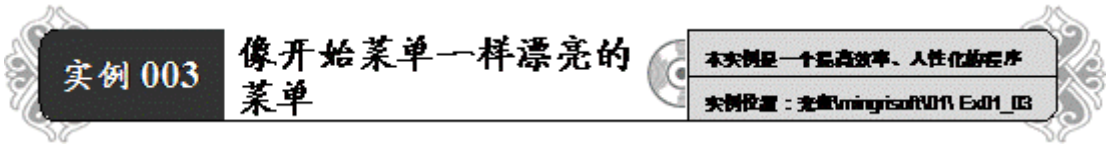
图 1.9 像开始菜单一样的菜单

```
            Form2 f = (Form2)sender;
            ToolStripMenuItem item = new ToolStripMenuItem();
            for (int i = 0; i < f.contextMenuStrip2.Items.Count; )//合并菜单
            {
                item.DropDownItems.Add(f.contextMenuStrip2.Items[i]);
            }
            this.contextMenuStrip1.Items.AddRange(new System.Windows.Forms.ToolStripI
tem[] {
            item});
        }
```

举一反三

根据本实例，读者可以实现以下功能。

- 让右键菜单在子窗体中显示。
- 让右键菜单在主窗体和子窗体中同时显示。



实例 003 像开始菜单一样漂亮的菜单

实例说明

Windows 的开始菜单非常的独特，在菜单的旁边有一条竖着的彩条，彩条中还写着文字。这种独特的菜单能够使程序的界面看起来更加的漂亮。本例中就实现了这种菜单，运行本例弹出“打开菜单”时，就会看到菜单的左边有一个紫色的彩条。实例效果如图 1.3 所示。

技术要点

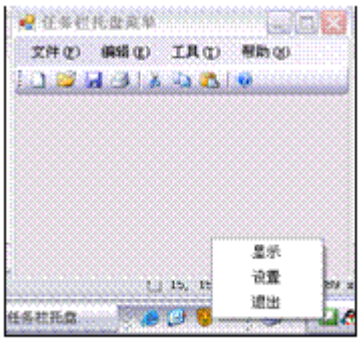


图 1.4 托盘菜单

在 C# 2.0 中，MenuStrip 控件中的子项 ToolStripMenuItem 已经包括了左侧的彩条，实现像开始菜单一样的菜单非常容易，不像在其他计算机语言的开发环境中，需要调用 API 才可以实现。如果想改变左侧竖着的彩条，只要给对应的菜单项设置相应的图片即可。

注意：如果要在左侧彩条显示文字，只要在对应的图片上加入文字即可。

实现过程

（1）创建一个项目，将其命名为 Ex01_03，默认窗体为 Form1。

（2）从工具箱中向 Form1 窗体添加 MenuStrip 控件。

（3）为 MenuStrip 控件添加相应的子项。

（4）为子项添加相应的图片。

举一反三

根据本实例，读者可以实现以下功能。

- 将菜单元设置成不同的格式（如图片、文字等）。
- 在菜单左侧播放动画。

实例 004 任务栏托盘菜单

实例说明

有一些软件通常只是在后台运行，这些进程大部分时间不显示用户界面。可通过单击任务栏状态通知区域的图标来访问的病毒防护程序就是一个示例。Windows 窗体中的 NotifyIcon 控件通常用于显示在后台运行的进程的图标，本实例利用该控件制作了一个任务栏托盘菜单。实例效果如图 1.4 所示。

技术要点

要实现程序启动时出现在系统托盘中。必须要为窗体添加 NotifyIcon 控件和 ContextMenuStrip 控件。

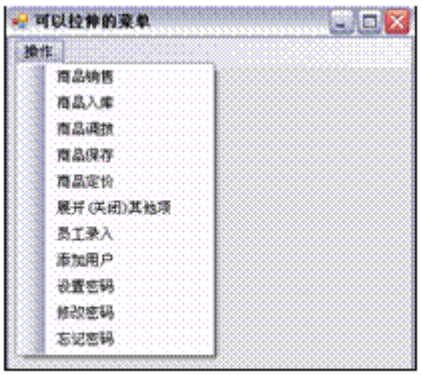


图 1.5 可以拉伸的菜单

注意：必须为 NotifyIcon 控件的 Icon 属性设置图标。

实现过程

（1）创建一个项目，将其命名为 Ex01_04，默认窗体为 Form1。

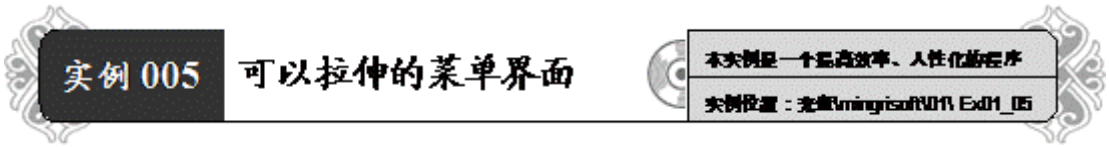
（2）向 Form1 窗体添加 NotifyIcon 控件和 ContextMenuStrip 控件，并为 ContextMenuStrip 控件添加子项。

（3）选择 NotifyIcon 控件，在其属性窗口中将 ContextMenuStrip 属性设置为添加到窗体上的 ContextMenuStrip 控件，并为 Icon 属性设置图片。

举一反三

根据本实例，读者可以开发以下程序。

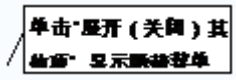
- 程序启动时不出现界面，直接出现在系统托盘中运行的后台程序。
- 程序启动时不出现在任务栏中。



实例 005 可以拉伸的菜单界面

实例说明

如果管理程序功能菜单非常多，而用户只使用一些常用菜单，这时，可以将主菜单项下的不常用菜单隐藏起来。此种显示方式类似于对菜单进行拉伸。使用时，只需单击展开菜单，即可显示相应菜单功能。运行本例，效果如图 1.5 所示。



技术要点

要实现可以拉伸的菜单，关键是要使用一个开关变量，同时调用 ShowDropDown()方法，显示操作后的结果。下面详细介绍一下该方法。

ShowDropDown()方法用来显示与此ToolStripDrop DownItem关联的ToolStripDropDownItem控件。其语法结构如下：

```
public void ShowDropDown ()
```

另外，用ShowDropDown()方法还可以显示已由 DropDown 属性设置的下拉控件。

 注意：必须设置开关变量的初值。

实现过程

（1）创建一个项目，将其命名为Ex01_05，默认窗体为Form1。

（2）从工具箱中向Form1窗体添加MenuStrip控件，选中MenuStrip控件为其添加子项。

（3）双击“展开(关闭)子项”为其双击事件添加处理代码。

（4）主要程序代码。

```
private void Form1_Load(object sender, EventArgs e)
{
   //初始设置下面的菜单隐藏
   this.设置密码 ToolStripMenuItem.Visible = false;
   this.添加用户 ToolStripMenuItem.Visible = false;
   this.忘记密码 ToolStripMenuItem.Visible = false;
   this.修改密码 ToolStripMenuItem.Visible = false;
   this.员工录入 ToolStripMenuItem.Visible = false;
}
private void toolStripMenuItem1_Click(object sender, EventArgs e)
{
   switch (i)
   {
      case 1:
      this.设置密码 ToolStripMenuItem.Visible = false;
      this.添加用户 ToolStripMenuItem.Visible = false;
      this.忘记密码 ToolStripMenuItem.Visible = false;
      this.修改密码 ToolStripMenuItem.Visible = false;
      this.员工录入 ToolStripMenuItem.Visible = false;
      i = 2;
      this.操作 ToolStripMenuItem.ShowDropDown();
      break;
      case 2:
      this.设置密码 ToolStripMenuItem.Visible = true;
```

```
            this.添加用户 ToolStripMenuItem.Visible = true;
            this.忘记密码 ToolStripMenuItem.Visible = true;
            this.修改密码 ToolStripMenuItem.Visible = true;
            this.员工录入 ToolStripMenuItem.Visible = true;
            i = 1;
            this.操作 ToolStripMenuItem.ShowDropDown();
            break;
        }
    }
```

举一反三

根据本实例，读者可以开发以下功能。

- 制作显示\隐藏工具栏。
- 合并菜单栏。

实例 006　菜级联菜单

实例说明

如果管理程序功能菜单非常多，一些功能中又包括许多子功能，这时可以使用级联菜单来组织系统的各个功能。实例运行结果如图 1.6 所示。

图 1.6　级联菜单

技术要点

制作级联菜单需要使用 MenuStrip 控件。

注意：在使用级联菜单时最好不要超过 5 层，否则用户在使用时会很不方便。

实现过程

（1）创建一个项目，将其命名为 Ex01_06，默认窗体为 Form1。

（2）在 Form1 窗体添加 MenuStrip 控件，选中 MenuStrip 控件为其添加子项和级联子项。

举一反三

根据本实例，读者可以开发以下功能。

- 大型系统的功能导航。
- 在窗体四周再增加菜单栏。

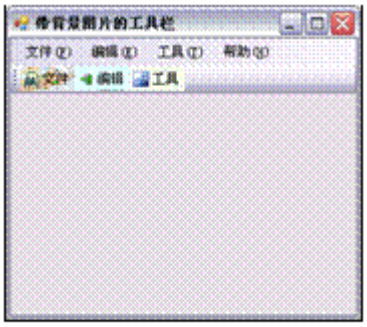
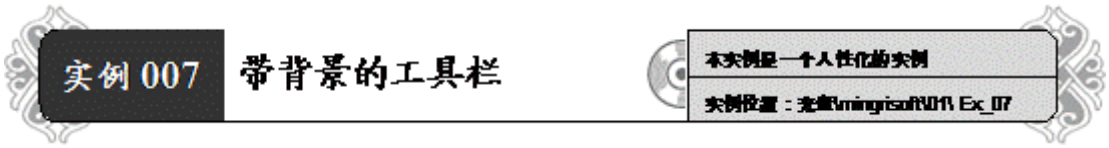


图 1.7 带背景的工具栏

1.2 工具栏设计

在菜单栏中将常用的菜单命令以工具栏按钮的形式显示，并作为快速访问方式。工具栏位于菜单栏的下方，由许多命令按钮组成，每个命令按钮上都有一个形象的小图标，以标识命令按钮的功能。由于工具栏这种直观易用的特点，使其已成为 Windows 应用程序的标准界面。



实例 007 带带背景的工具栏

实例说明

工具栏是窗体的组成部分之一，工具栏中的按钮可以设定完成一些较为常用或重要的功能，本例中设计了一个工具栏，并且为该工具栏作了一些修饰，使工具栏带有背景。背景图案可以透过按钮显示，效果如图 1.7 所示。

技术要点

工具栏中的背景是一幅图片，在运行时应该将该图片绘制到工具栏上，在.NET 2.0 中，只需将工具栏按钮的 BackGroundImage 的属性设置为对应的图片即可。

实现过程

（1）创建一个项目，将其命名为 Ex01_07，默认窗体为 Form1。

（2）从工具箱中为 Form1 窗体添加 ToolStrip 控件，并为工具栏添加相应的按钮。

（3）为工具栏的按钮设置相应的 BackGroundImage 属性，相应的的图片就会变成按钮的背景。

举一反三

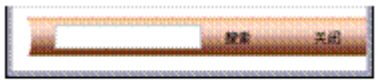


图 1.8 浮动的工具栏

根据本实例，读者可以开发以下功能。

- 制作一个带动画效果的工具栏。
- 制作一个自定义样式的工具栏。

实例 008 带浮动工具栏

实例说明

通常情况下，窗体显示在屏幕的中心。对于使用频率非常高的软件，通常放在屏幕上端以浮动工具栏形式显示。下面通过实例介绍浮动工具栏的设计方法。运行程序，程序可以停在屏幕的任何位置，当窗体失去焦点后，窗体将自动隐藏。效果如图 1.8 所示。

技术要点

窗体是否要隐藏，重要的是要判断在操作中，通过窗体的 Focused 属性，是否可以确定窗体有焦点。在窗体有焦点时，该窗体正在被操作，这时需要完全显示在屏幕当中，如果窗体没有焦点，通过设置窗体到屏幕的高度，来确定窗体的隐藏部分。下面详细介绍一下 Focused 属性。

Focused 属性用来获取一个值，该值指示控件是否有输入焦点。其语法结构如下：

```
public virtual bool Focused { get; }
```

l 属性值：如果控件有焦点，则为 True；否则为 False。

实现过程

（1）创建一个项目，将其命名为Ex01_08，默认窗体为Form1。

（2）从工具箱中为Form1窗体添加Panel控件，并为Panel控件添加相应的背景图片。

（3）在Panel上添加两个Label控件，并将其Text属性设置为“打开”和“关闭”，同时把两个Label控件的背景颜色设置为透明。

（4）主要程序代码。

```
private void timer1_Tick(object sender, EventArgs e)
{
   if (this.Focused == false)
   {
      this.Top = -30;
   }
}
private void label2_Click(object sender, EventArgs e)
{
   this.Close();
}
private void panel1_MouseClick(object sender, MouseEventArgs e)
{
   this.Top = 60;
}
```

举一反三

根据本实例，读者可以开发以下功能。

- 制作一个带动画效果的工具栏。
- 制作一个飘动的工具栏。

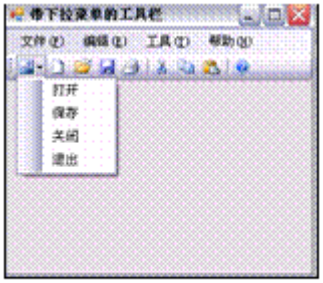


图 1.9 带下拉菜单的工具栏

实例 009 带下拉菜单的工具栏 本实例可以方便操作、提高效率 实例位置：光盘\mingrisoft\01\Ex01_09

实例 009 在带下拉菜单的工具栏

实例说明

工具栏是窗体的组成部分之一，工具栏中的按钮可以完成一些较为常用或重要的功能，本例中设计了一个工具栏，使工具栏带有下拉菜单，效果如图 1.9 所示。

技术要点

带下拉菜单的工具栏在其他计算机语言中实现比较复杂，但在.NET 2.0 中已经提供了这个功能，只需将工具栏按钮的类型设置为 DropDownButton 即可。

实现过程

（1）创建一个项目，将其命名为 Ex01_09，默认窗体为 Form1。

（2）从工具箱中为窗体添加 ToolStrip 控件，并为工具栏添加相应的按钮，在按钮的下拉选项中选择 DropDownButton 类型。

（3）为工具栏 DropDownButton 类型的按钮设置相应的下拉菜单，就可以轻松实现带下拉菜单的工具栏。

举一反三

根据本实例，读者可以开发以下功能。

- 制作一个带右键菜单的工具栏。
- 制作一个带复选框的工具栏。

实例 010 具有提示功能的工具栏 本实例可以方便操作、提高效率 实例位置：光盘\mingrisoft\01\Ex01_10

实例 010 在具有提示功能的工具栏

实例说明

图 1.10 带有提示的工具栏

在文档\视图结构的应用程序中，默认情况下，当鼠标在工具栏按钮上停留片刻，会出现一个工具提示条。本例实现了一个具有提示功能的工具栏，效果如图 1.10 所示。

技术要点

具有提示功能的工具栏在其他计算机语言中实现也许比较复杂，但在.NET 2.0 中已经提供了这个功能。只需将工具栏按钮的 ToolTipText 设置为要提示的内容即可。下面详细介绍一下该属性。

ToolTipText 属性用来获取或设置作为控件的 ToolTip 显示的文本。其语法结构如下：

```
public string ToolTipText { get; set; }
```

l 属性值：一个表示工具提示文本的字符串。

实现过程

（1）创建一个项目，将其命名为 Ex01_10，默认窗体为 Form1。

（2）从工具箱中为 Form1 窗体添加 ToolStrip 控件用来设计工具栏，并为工具栏添加相应的按钮。

（3）为相应按钮的 ToolTipTile 属性设置提示内容，就可以轻松实现具有提示功能的工具栏。

举一反三

根据本实例，读者可以开发以下功能。

- 具有提示功能的各种控件。
- 具有提示功能的窗体。

1.3 状态栏设计

状态栏是用来显示当前程序状态的。状态栏可以分为多个面板，用来显示不同状态下的内容，本节主要介绍了状态栏的用法以及如何在状态栏中添加控件。

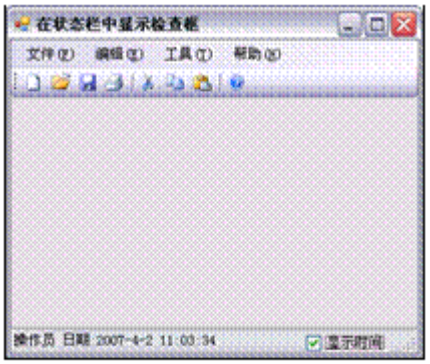


图 1.11 在状态栏中显示检查框

实例 011 在状态栏中显示检查框

实例说明

在设计程序界面时，为了规范界面，可以将一些控件放置在状态栏中，这样既能起到控制程序的作用，又能使界面和谐、美观。运行程序，在窗体的状态栏中加入了显示时间检查框。效果如图 1.11 所示。

技术要点

在状态栏中添加检查框比较容易，只需先将状态栏加入窗体，然后将检查框从工具箱中拖入状态拦即可。

实现过程

（1）创建一个项目，将其命名为 Ex01_11，默认窗体为 Form1。

（2）从工具箱中为 Form1 窗体添加 StatusStrip 控件，并从工具箱中为状态栏添加 CheckBox 控件。

（3）主要程序代码。

```
private void checkBox2_CheckedChanged(object sender, EventArgs e)
{
    if (this.checkBox2.Checked)
    {
        statusStrip1.Items[1].Text = "日期:" + DateTime.Now.ToString();
    }
    else
    {
        statusStrip1.Items[1].Text = "";
    }
```

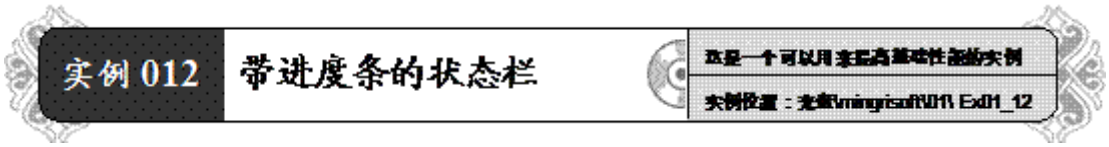
图 1.12 带进度条的状态栏

```
}
```

举一反三

根据本实例，读者可以开发以下功能。

- 运行时设置控件的位置。
- 动态控制控件的显示。



实例 012 带进度条的状态栏

实例说明

上网浏览网页的读者都用过 IE 浏览器，读者是否注意到该浏览器的状态栏，在打开网页的过程中，浏览器下边的状态栏中有一个进度条显示当前网页的载入进度，这样的状态栏使界面显得更加丰富多彩，并且非常实用。本例将设计一个带进度条的状态栏，并且在程序运行当中进度条可以显示其进度，该实例运行结果如图 1.12 所示。

技术要点

带进度条的状态栏在别的开发环境下实现相对比较复杂，但在.NET 2.0 中已经提供了这个功能，只需将状态栏的按钮类型设置为 ProgressBar 即可。通过设置 ProgressBar 的 Step 属性指定一个特定值用以逐次递增 Value 属性的值，然后调用 PerformStep 方法来使该值递增，就可以实现带进度条的状态栏。

实现过程

（1）创建一个项目，将其命名为 Ex01_12，默认窗体为 Form1。

（2）从工具箱中为 Form1 窗体添加 StatusStrip 控件，并为状态栏添加相应的按钮，在按钮的下拉选项中选择 ProgressBar 类型。

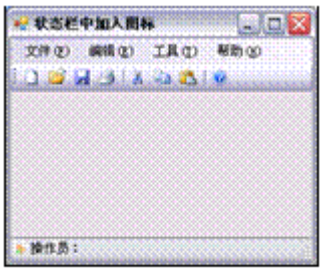


图 1.13 状态栏中加入图标

（3）设置 ToolStripProgressBar1 的 Value 属性、Maximum 属性和 Step 属性。

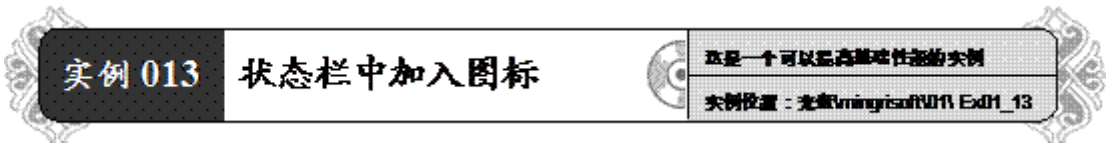
（4）主要程序代码。

```
private void Form1_Load(object sender, EventArgs e)
{
    while (toolStripProgressBar1.Value < toolStripProgressBar1.Maximum)
    {
        this.toolStripProgressBar1.PerformStep();
    }
}
```

举一反三

根据本实例，读者可以开发以下功能。

- 在状态栏中显示时间。
- 改变进度条的颜色。



实例 013 状态栏中加入图标

实例说明

状态栏已经成为主界面必不可少的部分，状态栏一般用于显示程序状态、当前日期等，在状态栏中添加一张图片会使程序的主界面更有特色。运行本例，效果如图 1.13 所示。

技术要点

状态栏中加入图标在.NET 2.0 中实现非常容易，只要将对应状态栏面板的 Image 属性设置为要显示的图片即可。

实现过程

（1）创建一个项目，将其命名为 Ex01_13，默认窗体为 Form1。

（2）从工具箱中为 Form1 窗体添加 StatusStrip 控件，并为状态栏添加相应的按钮，设置添加的按钮的 Image 属性为要显示的图片。

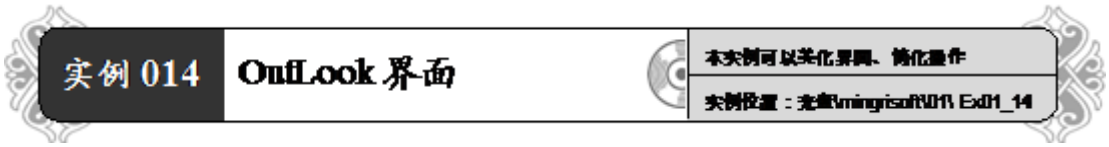
举一反三

根据本实例，读者可以实现以下功能。

- 将其他控件放置在状态栏中，如进度条。
- 将其他控件放置在状态栏中，如复选框。

1.4 导航菜单界面

对于一些应用工具软件，界面不但要求人性化、漂亮，还要突出界面功能、使用方便，这样才能吸引用户使用。本节主要介绍了常用的几种菜单界面。



实例 014 OutLook 界面

实例说明

程序主界面包括菜单栏、工具栏、状态栏和树状视图。OutLook 界面美观、友好，是一个很实用的程序主界面，并且菜单栏和工具栏是可移动的。运行本例效果如图 1.14 所示。

图 1.14 Out Look 界面

技术要点

一般程序的菜单栏和工具栏是不可移动的，但是只要将 MenuStrip 和 ToolStrip 控件的 AllowItemRecorder 属性设为 True 就可以移动。在本例中使用 MenuStrip 控件制作菜单栏，使用 ToolStrip 制作工具栏，使用 StatusStrip 控件制作状态栏。下面详细介绍一下这几个控件的属性。

1. ToolStrip. AllowItemReorder 属性

获取或设置一个值，该值指示是否由 ToolStrip 类私自处理拖放和项重新排序。其结构如下：

public bool AllowItemReorder { get; set; }

l 属性值：如果让 ToolStrip 类自动处理拖放和项重新排序，为 True；否则为 False。默认值为 False。

2. MenuStrip. AllowItemReorder 属性

获取或设置一个值，该值指示是否由 ToolStrip 类私自处理拖放和项重新排序。其结构如下：

public bool AllowItemReorder { get; set; }

l 属性值：如果让 MenuStrip 类自动处理拖放和项重新排序，为 True；否则为 False。默认值为 False。

3. ToolStripItem.DisplayStyle 属性

获取或设置是否在 ToolStripItem 上显示文本和图像。

public virtual ToolStripItemDisplayStyle DisplayStyle { get; set; }

l 属性值：ToolStripItemDisplayStyle 值之一。默认为 ImageAndText。

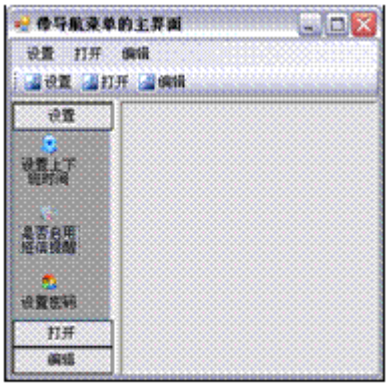


图 1.15 带导航菜单的主界面

注意：在移动菜单栏和工具栏时，需要按住“Alt”键，同时用鼠标进行拖动。

实现过程

（1）创建一个项目，将其命名为 Ex01_14，默认窗体为 Form1。

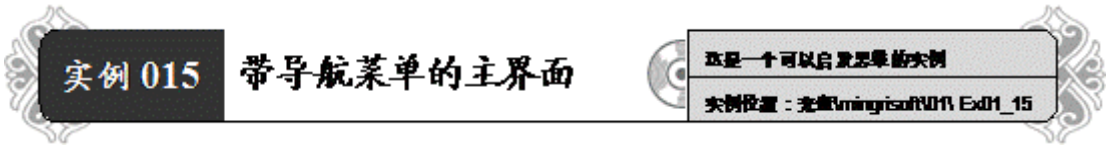
（2）在 Form1 窗体上添加 MenuStrip 控件，用来设计主菜单；添加 ToolStrip 控件，用来设计工具栏；添加 StatusStrip 控件，用来设计状态栏；添加 ImageList 控件和 TreeVew 控件，用来设计树结构。

（3）分别为 MenuStrip 控件、ToolStrip 控件、ImageList 控件和 TreeVew 控件添加子项，将 MenuStrip 控件和 ToolStrip 控件的 AllowItemRecorder 属性设为 True，并将 ToolStrip 控件的每个子项的 DisplayStyle 属性设置为“ImageAndText”。下面详细介绍这几个属性。

举一反三

根据本实例，读者可以实现以下功能。

- 制作一个系统菜单。
- 制作一个导航界面。



实例 015 带带导航菜单的主界面

实例说明

在窗体界面中，菜单栏是不可缺少的重要组成部分。本实例是用其他的控件来制作一个摸拟菜单栏。运行程序，单击窗体上面的按钮，将会在按钮的下面显示一个下拉列表。如图 1.15 所示。

技术要点

该实例中主要使用 Button 控件和 ListView 控件制作导航菜单界面。在对 ListView 控件添加菜单信息时，必需在前面写入添加语句，如 Listview.Items.Add，否则添加的菜单信息将替换前一条信息。单击相应的按钮时，应首先对 ListView 控件进行清空，否则在 ListView 控件中将继续上一次的添加菜单信息。

实现过程

（1）创建一个项目，将其命名为 Ex01_15，默认窗体为 Form1。

（2）在 Form1 窗体上添加 MenuStrip 控件设计菜单栏；添加 ToolStrip 控件设计工具栏；添加 SplitContainer 控件、ImageList 控件、3 个 Button 控件和 ListView 控件用来制作左侧的导航栏。

（3）分别为 MenuStrip 控件、ToolStrip 控件添加子项，将 3 个 Button 按钮和 ListView 控件加入 SqlitContainer1.panel 的左侧部分中。

（4）主要程序代码。

加载窗体时，设置左侧导航栏内容的实现代码如下：

```
private void Form1_Load(object sender, EventArgs e)
{
    listView1.Clear();
    listView1.LargeImageList = imageList1;
    listView1.Items.Add("设置上下班时间", "设置上下班时间", 0);
    listView1.Items.Add("是否启用短信提醒", "是否启用短信提醒", 1);
    listView1.Items.Add("设置密码", "设置密码", 2);

}
```

添加打开按钮的 ListView 控件显示内容的实现代码如下：

```
private void button2_Click_1(object sender, EventArgs e)
{
    listView1.Dock = DockStyle.None;
    button2.Dock = DockStyle.Top;
```

```
        button1.SendToBack();
        button1.Dock = DockStyle.Top;
        button3.Dock = DockStyle.Bottom;
        listView1.Dock = DockStyle.Bottom;
        listView1.Clear();
        listView1.Items.Add("近期工作记录", "近期工作记录", 3);
        listView1.Items.Add("近期工作计划", "近期工作计划", 4);
    }
```

添加编辑按钮的 ListView 控件显示内容的实现代码如下：

```
    private void button3_Click_1(object sender, EventArgs e)
    {
        listView1.Dock = DockStyle.None;
        button3.SendToBack();
        button3.Dock = DockStyle.Top;
        button2.SendToBack();
        button2.Dock = DockStyle.Top;
        button1.SendToBack();
        button1.Dock = DockStyle.Top;
        listView1.Dock = DockStyle.Bottom;
        listView1.Clear();
        listView1.Items.Add("编辑工作进度报告", "编辑工作进度报告", 5);
        listView1.Items.Add("编辑项目设计图", "编辑项目设计图", 6);
    }
```

添加设置按钮的 ListView 控件显示内容的实现代码如下：

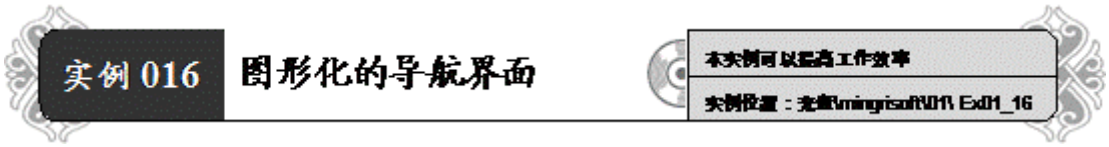
```
    private void button1_Click_1(object sender, EventArgs e)
    {
        listView1.Dock = DockStyle.None;
        button1.Dock = DockStyle.Top;
```

```
        button2.Dock = DockStyle.Bottom;
        button3.SendToBack();
        button3.Dock = DockStyle.Bottom;
        listView1.BringToFront();
        listView1.Dock = DockStyle.Bottom;
        listView1.Clear();
        listView1.Items.Add("设置上下班时间", "设置上下班时间", 0);
        listView1.Items.Add("是否启用短信提醒", "是否启用短信提醒",1);
        listView1.Items.Add("设置密码", "设置密码", 2);
    }
```

举一反三

根据本实例，读者可以实现以下功能。

- 制作一个系统菜单。
- 制作大型系统的导航界面。



实例 016　图形化的导航界面

实例说明

如果以按钮来代替菜单的功能，会使界面更具有个性化。使操作者更易于操作。下面介绍按钮显示菜单的设计方法。运行本例，效果如图 1.16 所示。

图 1.16　图形化的导航界面

技术要点

本实例主要通过设置 Button 控件的相应属性，确定其按钮的位置、文字、显示样式和要显示的图片等。下面对 Button 控件相应属性进行详细介绍。

1. Button. BackColor 属性

获取或设置控件的背景色，其方法结构如下：

```
public override Color BackColor { get; set; }
```

l 属性值：一个表示背景色的 Color 值。

2. Button. FlatStyle 属性

获取或设置按钮控件的平面样式外观。其代码如下：

```
public FlatStyle FlatStyle { get; set; }
```

l 属性值：FlatStyle 值之一。默认值为 Standard。

3. Button. TextImageRelation 属性

获取或设置文本和图像相互之间的相对位置。其代码如下：

```
public TextImageRelation TextImageRelation { get; set; }
```

l 属性值：TextImageRelation 的值之一。默认为 Overlay。

实现过程

（1）创建一个项目，将其命名为 Ex01_16，默认窗体为 Form1。

（2）在 Form1 窗体上添加 MenuStrip 控件用来设计菜单栏，添加 ToolStrip 控件用来设计工具栏，添加 Panel 控件、Button 控件用来设计图形化的导航按钮。

（3）分别为 MenuStrip 控件、ToolStrip 控件添加子项，并为 Panel 控件选择背景图片。

（4）将 Button 控件的 BackColor 属性设为“Transparent”、FlatStyle 属性设置为“Flat”、TextImageRelation 属性设置为“ImageBeforeText”。

（5）主要程序代码。

```
private void button1_Click(object sender, EventArgs e)
{
   //使子项可见
   button5.Visible = true;
```

```
        button6.Visible = true;
        button7.Visible = true;
    }
    private void button2_Click(object sender, EventArgs e)
    {
        //使子项可见
        button8.Visible = true;
        button9.Visible = true;
        button10.Visible = true;
    }
    private void button3_Click(object sender, EventArgs e)
    {
        //使子项可见
        button11.Visible = true;
        button12.Visible = true;
        button13.Visible = true;
    }
```

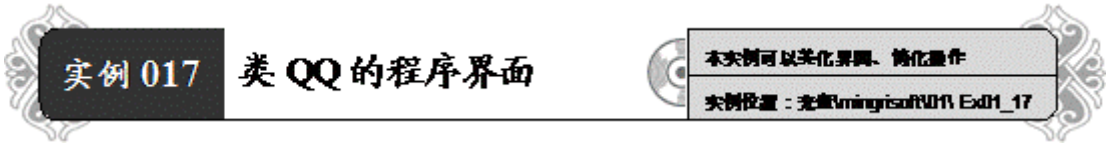
举一反三

根据本实例，读者可以实现以下功能。

- 制作动态的按钮界面。
- 制作动态的图片界面。

1.5 特色程序界面

现在有很多开发人员都将界面制作成不同类型的样式，这样可以使界面更加形象化。本节主要介绍了如何对程序界面进行特色化设计，如类似 QQ、Windows XP 的界面等。



实例 017 菜类 QQ 的程序界面

实例说明

图 1.17 QQ的程序界面

一般程序都是以菜单栏和工具栏的形式调用其他功能模块，如果以动态的类似 QQ 的程序界面来调用其他功能模块，将会给用户一种新鲜的感觉，使用户对软件更感兴趣。实例运行结果如图 1.17 所示。

技术要点

本例主要使用 Button 控件来完成布局，使用 ListView 控件来显示有图标的功能菜单。ListView 控件的常用属性及说明如下。

1. ListView.Items 属性

使用该属性可直接访问表示列表中项目的 ListItem 对象。其结构如下：

```
public ListViewItemCollection Items { get; }
```

l 属性值： ListView.ListViewItemCollection 包含 ListView 控件中所有的项。

2. ListView.Dock 属性

获取或设置哪些控件边框停靠到其父控件并确定控件如何随其父级一起调整大小。其结构如下：

```
public virtual DockStyle Dock { get; set; }
```

l 属性值：DockStyle 值之一。默认为 None。

实现过程

（1）创建一个项目，将其命名为 Ex01_17，默认窗体为 Form1。

（2）在窗体上添加 Button 控件、ListView 控件和 ImageList 控件。设置 ListView 控件的 ImageList 属性为 ImageList 控件。

（3）主要程序代码。

添加“我的好友”选项内容的实现代码如下：

```
private void button1_Click(object sender, EventArgs e)
{
    listView1.Dock = DockStyle.None;
    button1.Dock = DockStyle.Top;
    button2.Dock = DockStyle.Bottom;
```

```
        button3.SendToBack();
        button3.Dock = DockStyle.Bottom;
        listView1.BringToFront();
        listView1.Dock = DockStyle.Bottom;
        listView1.Clear();
        listView1.Items.Add("小猪", "小猪", 0);
        listView1.Items.Add("小狗", "小狗", 1);
        listView1.Items.Add("娇娇", "娇娇", 2);
    }
```

添加默认时选项内容的实现代码如下：

```
    private void Form1_Load(object sender, EventArgs e)
    {
        listView1.Clear();
        listView1.LargeImageList = imageList1;
        listView1.Items.Add("小猪", "小猪", 0);
        listView1.Items.Add("小狗", "小狗", 1);
        listView1.Items.Add("娇娇", "娇娇", 2);

    }
```

添加“陌生人”选项内容的实现代码如下：

```
    private void button2_Click(object sender, EventArgs e)
    {
        listView1.Dock = DockStyle.None;
        button2.Dock = DockStyle.Top;
        button1.SendToBack();
        button1.Dock = DockStyle.Top;
        button3.Dock = DockStyle.Bottom;
        listView1.Dock = DockStyle.Bottom;
```

```
        listView1.Clear();
        listView1.Items.Add("北风", "北风", 3);


    }
```

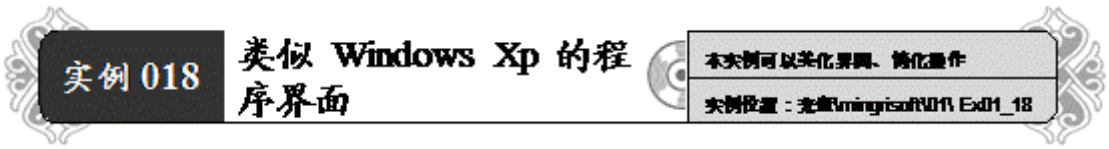
添加“黑名单”选项内容的实现代码如下：

```
    private void button3_Click(object sender, EventArgs e)
    {
        listView1.Dock = DockStyle.None;
        button3.SendToBack();
        button3.Dock = DockStyle.Top;
        button2.SendToBack();
        button2.Dock = DockStyle.Top;
        button1.SendToBack();
        button1.Dock = DockStyle.Top;
        listView1.Dock = DockStyle.Bottom;
        listView1.Clear();
        listView1.Items.Add("冰雨", "冰雨", 5);
    }
```

举一反三

根据本实例，读者可以实现以下功能。

- 根据数据库信息形成相应的功能列表。
- 制作聊天界面。



实例 018　类似 windows xp 的程序界面

实例说明

在 Windows XP 环境下打开控制面板，会发现左侧的导航界面很实用。双击展开按钮，导航栏功能显示出来，双击收缩按钮，导航按钮收缩。下面通过实例介绍此种主窗体的设计方法。运行本例，效果如图 1.18 所示。

技术要点

PictureBox 控件是一个图像显示控件，该控件主要以其中的 Image 属性存储图像数据。其详细介绍如下。

PictureBox.Image 属性用来获取或设置 PictureBox 显示的图像，其语法格式如下：

```
public Image Image { get; set; }
```

图 1.18　类似 windows xp 的程序界面

l　　属性值：要显示的 Image。

实现过程

（1）创建一个项目，将其命名为 Ex01_18，默认窗体为 Form1。

（2）在 Form1 窗体上添加 Button 控件、PictureBox 控件和 label 控件，布局如图 1.18 所示。

（3）主要程序代码。

双击“向下箭头”的实现代码如下：

```
private void pictureBox5_Click(object sender, EventArgs e)
{
    //使子项收缩
    int i ;
    i=80;
    pictureBox5.Visible = false;
    pictureBox4.Visible = false;
    label2.Visible = false;
    label3.Visible = false;
    pictureBox6.Top -= i;
    pictureBox8.Top -= i;
    label4.Top -= i;
```

```
        label5.Top -= i;
        label6.Top -= i;
        label10.Top -= i;
        label7.Top -= i;
        label8.Top -= i;
        label9.Top -= i;
        pictureBox9.Top -= i;
        pictureBox11.Top -= i;
    }
```

双击“向上箭头”的实现代码如下：

```
    private void pictureBox2_Click(object sender, EventArgs e)
    {
        //展开子项
        if (pictureBox5.Visible == false)
        {
            int i;
            i = 80;
            pictureBox5.Visible = true;
            pictureBox4.Visible = true;
            label2.Visible = true;
            label3.Visible = true;
            pictureBox6.Top += i;
            pictureBox8.Top += i;
            label4.Top += i;
            label5.Top += i;
            label6.Top += i;
            label10.Top += i;
            label7.Top += i;
```

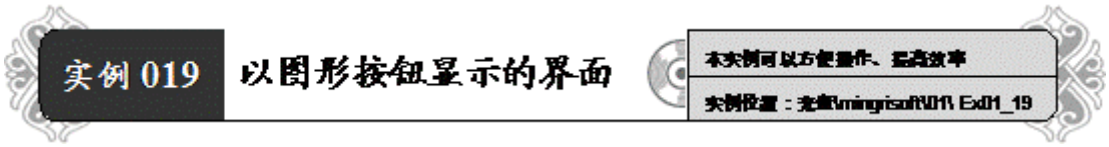
```
            label8.Top += i;
            label9.Top += i;
            pictureBox9.Top += i;
            pictureBox11.Top += i;
        }
    }
    private void Form1_Load(object sender, EventArgs e)
    {
        SetStyle(ControlStyles.SupportsTransparentBackColor,true);
    }
```

注意：在对控件的高度进行递增或递减的时候，数值不要太小。

举一反三

根据本实例，读者可以实现以下功能。

- 制作 Windows XP 控制面板。
- 制作 Windows XP 开始导航菜单。



实例 019　以图形按钮显示的界面

实例说明

菜单和工具栏虽然能方便用户操作程序的相应功能，但各有缺点。如果采用按钮式功能菜单，不但美观大方，而且操作灵活。当单击按钮时，用户区将显示相应的操作按钮组。下面介绍图形界面式菜单的设计方法。运行本例，效果如图 1.19 所示。

图 1.19　以图形按钮显示的界面

技术要点

本例中用到了 Image.FromFile 方法和 PictureBox.Image 属性，下面详细介绍一下。

（1）Image.FromFile 方法：从指定的文件创建 Image。该函数的结构为：

```
public static Image FromFile (string filename)
```

参数说明如下。

- filename：当前目录的指定路径字符串，包含要从中创建 Image 的文件的名称。
- 返回值：此方法创建的 Image。

（2）PictureBox.Image 属性：获取或设置 PictureBox 显示的图像。其属性结构为：

```
public Image Image { get; set; }
```

- 属性值：要显示的 Image。

注意：在本例中不易使窗体最大化。如最大化，会使 Label 控件不在正确的位置上。

实现过程

（1）创建一个项目，将其命名为 Ex01_19，默认窗体为 Form1。

（2）在 Form1 窗体上添加 MenuStrip 控件用来设计菜单；添加 Picture 控件、Panel 控件用来设计图形显示的界面。

（3）将 panel 的背景图片设置为图 1.19 所示，并在图片上添加 Label 控件，同时将 Label 控件的 BackColor 属性设置为 transparency。

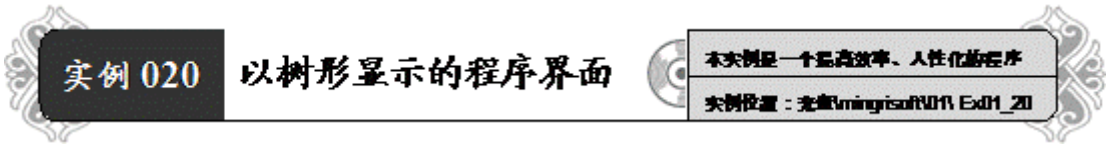
（4）主要程序代码。

```
private void label1_Click(object sender, EventArgs e)
{
  pictureBox3.Image= Image.FromFile("3.jpg");//为控件加载图像
}
```

举一反三

根据本实例，读者可以实现以下功能。

- 制作图片的动态更新。
- 制作一个图片浏览器。



实例 020　以树形显示的程序界面

实例说明

以树形来显示程序的菜单，可以更直观更快捷的对软件进行操作。树形菜单比菜单栏更加美观实用。下面介绍树形界面菜单的设计方法。运行本例效果如图 1.20 所示。

图 1.20　以树形显示的程序界面

技术要点

在对 TreeView 控件输入记录时，双击 Nodes 属性就可以对 TreeView 的节点进行设置。

可以在窗体的 Load 事件中输入下面的一条命令：

```
treeView1.ExpandAll();
```

功能：展开 TreeView 控件中所有的下级菜单。

实现过程

（1）创建一个项目，将其命名为 Ex01_20，默认窗体为 Form1。

（2）在窗体上添加 MenuStrip 控件用来设计菜单栏，添加 PictureBox 控件用来显示图片，添加 TreevVew 控件用来设计左侧树形导航界面。

（3）为 PictureBox 添加背景图片，给 MenuStrip 控件和 TreevVew 控件添加子项。

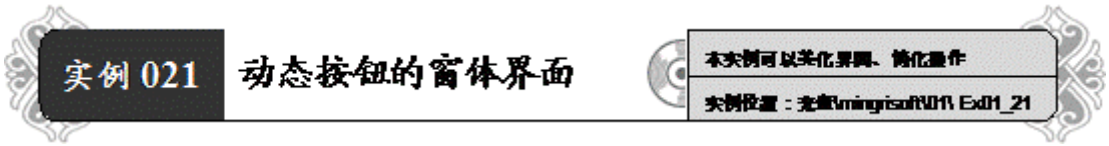
（4）主要程序代码。

```
private void Form1_Load(object sender, EventArgs e)
{
    treeView1.ExpandAll();
}
```

举一反三

根据本实例，读者可以实现以下功能。

- 制作一个动态的从数据库中读取数据的树型界面。
- 制作一个带图标的树型界面。



实例 021　动态按钮的窗体界面

实例说明

在窗体界面中，通常以按钮来代替菜单栏的功能，这种形式虽然给用户一种直观，界面风格各异的感觉，但通常按钮都是以静止的形式显示，如果使光标移到按钮时，可以使按钮上的图片和文字说明动态化，使用户快捷地找到所选按钮。这样就需要一个动态的按钮显示界面。运行本例效果如图 1.21 所示。

图 1.21　动态按钮的窗体界面

技术要点

在编辑过程中，首先在 Button 控件中的 Image 属性中添加图片，然后将 Button 控件的 ImageAlign 属性设置为 MiddleCenter，使图片居中。在设置 Button 控件的动态图片时，必须在相应控件的 MouseMove 事件中设置。

1．Button.Image 属性

获取或设置显示在按钮控件上的图像，其语法格式如下：

```
public Image Image { get; set; }
```

l 属性值：按钮控件上显示的 Image。默认值为空引用。

2．Button. ImageAlign 属性

获取或设置按钮控件上的图像对齐方式，其语法格式如下：

```
public ContentAlignment ImageAlign { get; set; }
```

l 属性值：ContentAlignment 值之一。默认值为 MiddleCenter。

 注意：当鼠标移开 Button 控件时，图片与字应及时恢复原位。

实现过程

（1）创建一个项目，将其命名为 Ex01_21，默认窗体为 Form1。

（2）在 Form1 窗体上添加 PictureBox 用来显示图片，添加 Button、Lable 控件用来设计动态按钮。

（3）为 PictureBox 控件设置背景图片，将 Lable 控件的 Text 属性设置为“明日科技有限公司”，并为每个 Button 控件设置图片和文字。

（4）主要程序代码。

```
private void button1_MouseMove(object sender, MouseEventArgs e)
{
    //鼠标移动时改变图片位置
   button1.ImageAlign = ContentAlignment.MiddleLeft;
}
private void button2_MouseMove(object sender, MouseEventArgs e)
{
```

```
        button2.ImageAlign = ContentAlignment.MiddleLeft;
    }
    private void button3_MouseMove(object sender, MouseEventArgs e)
    {
        button3.ImageAlign = ContentAlignment.MiddleLeft;
    }
    private void button4_MouseMove(object sender, MouseEventArgs e)
    {
        button4.ImageAlign = ContentAlignment.MiddleLeft;
    }
    private void button5_MouseMove(object sender, MouseEventArgs e)
    {
        button5.ImageAlign = ContentAlignment.MiddleLeft;
    }
    private void button1_MouseLeave(object sender, EventArgs e)
    {
        button1.ImageAlign = ContentAlignment.MiddleCenter;
    }
    //鼠标离开时改变图片位置
    private void button2_MouseLeave(object sender, EventArgs e)
    {
        button2.ImageAlign = ContentAlignment.MiddleCenter;
    }
    private void button3_MouseLeave(object sender, EventArgs e)
    {
        button3.ImageAlign = ContentAlignment.MiddleCenter;
    }
    private void button4_MouseLeave(object sender, EventArgs e)
```

```
{
    button4.ImageAlign = ContentAlignment.MiddleCenter;
}
private void button5_MouseLeave(object sender, EventArgs e)
{
    button5.ImageAlign = ContentAlignment.MiddleCenter;
}
private void button6_MouseMove(object sender, MouseEventArgs e)
{
    button6.ImageAlign = ContentAlignment.MiddleLeft;
}
private void button6_MouseLeave(object sender, EventArgs e)
{
    button6.ImageAlign = ContentAlignment.MiddleCenter;
}
```

举一反三

根据本实例，读者可以实现以下功能。

- 根据所给图片制作相应的窗体界面。
- 用图片代替控件制作相应的窗体界面。

1.6 特殊形状的窗体

将界面以不规则的形状显示在桌面上，可以给用户一种新鲜的感觉。本节主要对窗体以特殊的形状进行显示，如非矩形窗体和字体形窗体等。

实例 022 非矩形窗体

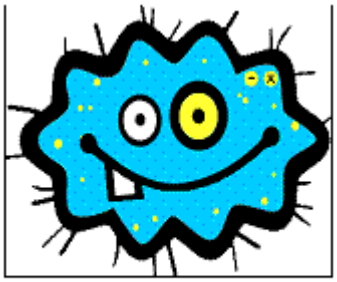


图 1.22 非矩形窗体

实例说明

大部分 Windows 窗体都是一个矩形区域，读者是否已经厌倦了这种中规中矩的矩形窗体？本例中的窗体是一个打破传统矩形的异型窗体，运行该例会看到一个非常可爱的窗体，单击【X】按钮就会使窗口关闭。实例效果如图 1.22 所示。

技术要点

以前，创建非矩形窗体是一个既费时又费人力的过程，其中涉及到 API 调用和大量的编程工作。在.NET 2.0 框架中可以不调用 API 非常轻松的实现这一功能。只要重写窗体的 OnPaint 方法，在方法中重新绘制窗体，然后用透明色将窗体设置透明即可。

（1）Form.OnPaint 方法：此成员重写 Control.OnPaint。用来重新绘制窗体图像。其结构如下：

```
protected override void OnPaint (PaintEventArgs e)
```

参数说明如下。

l PaintEventArgs：为 Paint 事件提供数据。

实现过程

（1）创建一个项目，将其命名为 Ex01_22，默认窗体为 Form1。

（2）在窗口中添加 Label 控件，并将 BackColor 属性设为透明，将 text 属性设为空。

（3）将窗体的 TransparencyKey 属性设为窗体的背景色。

（4）主要程序代码。

设置图片透明颜色的实现代码如下：

```
private void Form1_Load(object sender, EventArgs e)
{
    bit = new Bitmap("Heart.bmp");
    bit.MakeTransparent(Color.Blue);
}
```

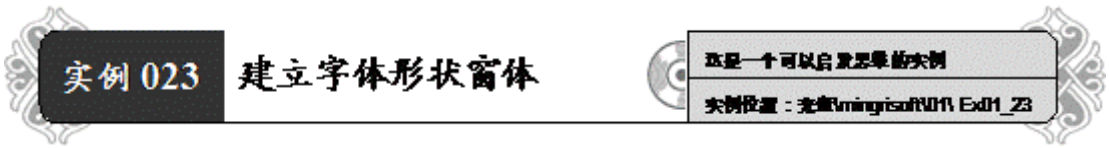
重写基类方法，具体代码如下：

```
protected override void OnPaint(PaintEventArgs e)
{
    e.Graphics.DrawImage((Image)bit, new Point(0, 0));//将图片画出
}
```

举一反三

根据本实例，读者可以开发以下程序。

- 可以把窗体制做成各种卡通图形。
- 可以将窗体制做成各种几何图形。
- 可以将窗体制做成桌面小精灵。



实例 023　建立字体形状窗体

图 1.23 建立字体形状窗体

实例说明

大家都见过不规则形状的窗体吧，那么如何制作一个文字形的窗体呢？文字形窗体一般应用在屏幕提示中，如收款机屏幕等。运行本例，效果如图 1.23 所示。

技术要点

以前，创建字体形窗体是一个既费时又费人力的过程，其中涉及到 API 调用和大量的编程工作。在.NET 2.0 框架中可以不调用 API 非常轻松的实现这一功能。只要先将字体画在一幅图上，然后重写窗体的 OnPaint 方法（方法的详细内容可以参见实例 022），在方法中用图重新绘制窗体，用背景色将窗体设置透明即可。

实现过程

（1）创建一个项目，将其命名为 Ex01_23，默认窗体为 Form1。

（2）主要程序代码。

```
namespace SpecialSharpWindows
{
    public partial class Form1 : Form
    {
        Bitmap bit;
        public Form1()
        {
            InitializeComponent();
        }
```

设置图片透明颜色的实现。代码如下：

```
        private void Form1_Load(object sender, EventArgs e)
        {
            bit = new Bitmap("1.bmp");
```

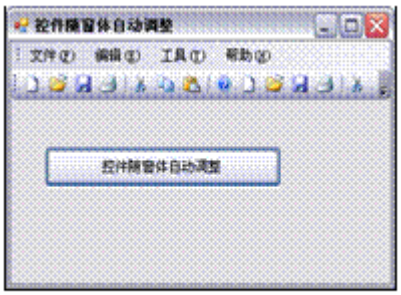


图 1.24 控件随窗体自动调整

```
            bit.MakeTransparent(Color.Blue);
        }
```

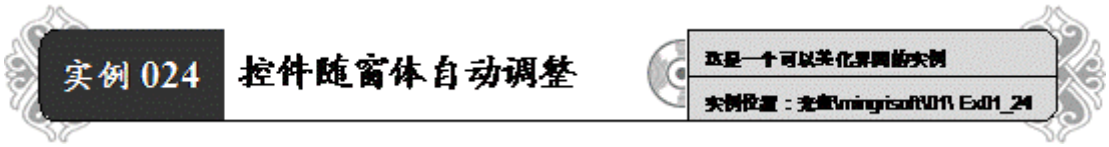
重写基类方法的实现。代码如下：

```
    protected override void OnPaint(PaintEventArgs e)
    {
        e.Graphics.DrawImage((Image)bit, new Point(0, 0));
    }
    private void label1_Click(object sender, EventArgs e)
    {
        this.Close();
    }
  }
}
```

举一反三

根据本实例，读者可以实现以下功能。

- 制作圆形的窗体。
- 制作锯齿状的窗体。



实例 024　控件随窗体自动调整

实例说明

在软件开发中，随着窗体的大小变化，界面会和设计时出现较大的差异，控件和窗体的大小会不成比例非常不美观。本例中的控件是一个可以随窗体大小变化的控件。运行该例会看到一个控件随窗体大小变化的窗体。实例效果如图 1.24 所示。

技术要点

在.NET 2.0 框架中可以非常轻松的实现这一功能。大多数控件都有 Anthor 属性，当在窗体上添加控件时设置 Anthor 属性即可。Anthor 属性是个锚定属性，

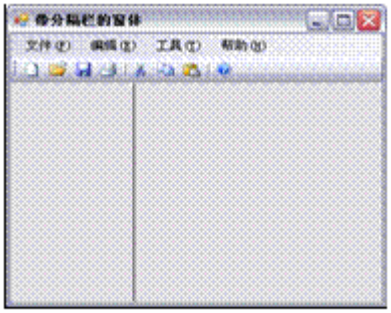


图 1.25 带分隔栏的窗体

指定了控件距容器边缘的距离。当窗体大小变化时，控件距窗体边缘的距离不变，自然大小就随窗体自动调整。

实现过程

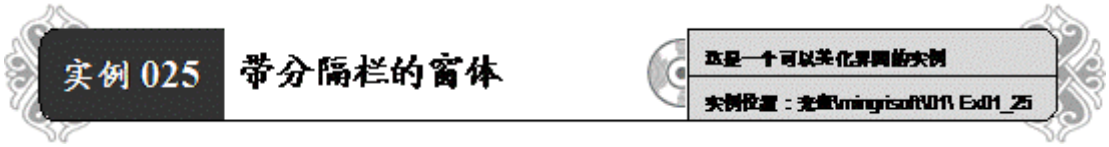
（1）创建一个项目，将其命名为 Ex01_24，默认窗体为 Form1。

（2）在窗体上添加 MenuStrip 控件、ToolStrip 控件和 Button 控件。并设置 Button 的 Anthor 属性和 Text 属性。

举一反三

根据本实例，读者可以实现以下功能。

- 图片大小随着窗体大小变化的窗体。
- 菜单栏大小随着窗体大小变化的窗体。



实例 025　带分隔栏的窗体

实例说明

在软件开发中，经常需要将界面分成几个部分，而且这几个部分又可以自由调整大小。运行本例，实例效果如图 1.25 所示。

技术要点

在.NET 2.0 框架中可以非常轻松的实现这一功能，只要在窗体中加入 SplitContainer 控件即可。SplitContainer 控件带有一个分隔栏，用来把窗体分成两部分。

实现过程

（1）创建一个项目，将其命名为 Ex01_25，默认窗体为 Form1。

（2）在 Form1 窗体上添加 MenuStrip 控件用来设计菜单栏，添加 ToolStrip 控件用来设计工具栏，添加 SplitContainer 控件用来设计分隔栏。

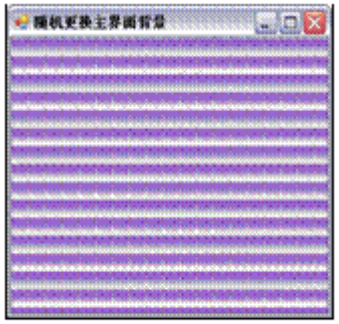
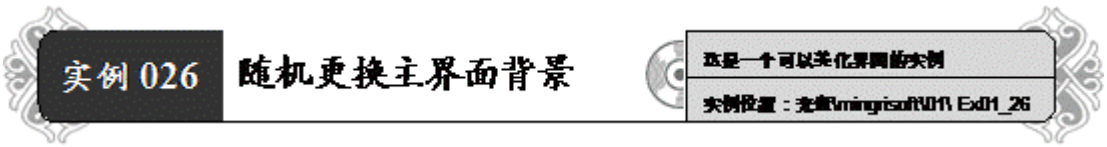


图 1.26 随机更换主界面背景的程序

举一反三

根据本实例，读者可以实现以下功能。

- 分成 3 部分的窗体。
- 分成 4 部分的窗体。



实例 026　随机更换主界面背景

实例说明

如果开发的软件用户使用频率非常高，可以为程序设计随机更换背景的程序。这样不但可以使用户心情愉快，也增加了软件的人性化设计。下面的界面就是一个随机更换主界面的例子，效果如图 1.26 所示。

技术要点

随机更换主界面背景使用了 Random 类和 ImageList 控件。首先为 ImageList 控件添加一组图片，然后实例化一个 Random 类，再用 Next()方法产生一个随机数以决定将哪个图片设为背景。

Random.Next()方法用来返回一个小于所指定最大值的非负随机数。其结构如下：

```
public virtual int Next (int maxValue)
```

参数说明如下。

l　maxValue：要生成的随机数的上界（随机数不能取该上界值）。maxValue 必须大于或等于零。

l　返回值：大于或等于零且小于 maxValue 的 32 位带符号整数，即返回的值范围包括零但不包括 maxValue。

实现过程

（1）创建一个项目，将其命名为 Ex01_26，默认窗体为 Form1。

图 1.27 自动安装的光盘程序

（2）在 Form1 窗体上添加 ImageList 控件，并为 ImageList 控件添加图片。

（3）主要程序代码。

```
private void Form1_Load(object sender, EventArgs e)
{
    Random rdn = new Random();
    int i = rdn.Next(imageList1.Images.Count);//产生一个随机数
    this.BackgroundImage = imageList1.Images[i];
}
```

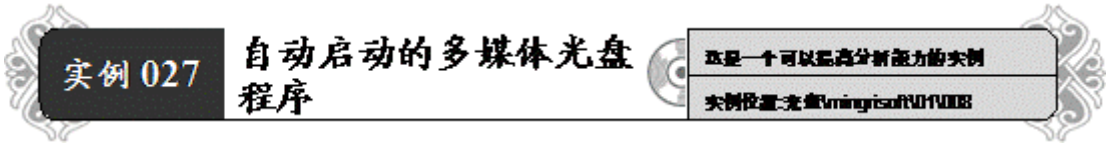
举一反三

根据本实例，读者可以开发以下程序。

- 每天更换主程序背景的主界面。
- 随机更换菜单栏、工具栏图标的主程序。

1.7 多媒体光盘

本节主要介绍了如何自动启动多媒体光盘和触摸屏的相关技术。自动启动多媒体光盘技术应用十分广泛，现在的光盘大多应用这些技术，节省了用户单击进入的时间。



实例 027 自动启动的多媒体光盘程序

实例说明

用户在安装软件时，将光盘放入光驱内，光盘会自动运行，进行安装操作，该功能是如何实现的呢？本例介绍如何制作“自动安装的光盘程序”，程序运行效果如图 1.27 所示。

技术要点

其实，实现光盘的自动运行非常简单，当用户打开自动运行的光盘时，会发现光盘中有几个特殊的文件，分别为“autorun.exe”、“run.ico”和“autorun.inf”，其中“autorun.exe”是光盘自动播放时执行的可执行文件，“run.ico”是光盘的图标，“autorun.inf”是一个 INI 文件。只要光盘中包含这些文件，那么在将光盘放入光驱时，就会自动运行。

实现过程

（1）创建一个项目，将其命名为 Ex01_27，默认窗体为 Form1。

（2）为 Form1 窗体添加背景图片 Button 控件。

（3）主要程序代码。

```
private void Form1_Load(object sender, EventArgs e)
{
    StreamWriter sw = new StreamWriter("AutoRun.inf",false);
    sw.WriteLine("[autorun]");
    sw.WriteLine("OPEN=AUTORUN.EXE");
    sw.WriteLine("ICON=run.ICO");
    sw.Close();
}
```

（4）运行程序，将可执行文件命名为“autorun.exe”，该文件就是光盘自动播放时打开的文件。

（5）选择一个图标，命名为“run.ico”，该图标在光驱读盘时显示。

（6）在刻光盘时，将上面的“run.ico”、“autorun.inf”和“autorun.exe”3 个文件刻录到光盘中。

举一反三

根据本实例，读者可以开发以下程序。

- 设计多媒体宣传光盘。

- 制作后门程序。

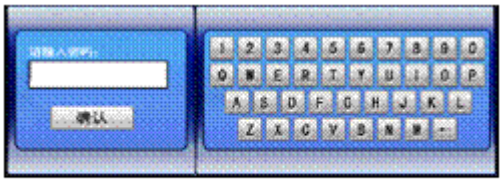
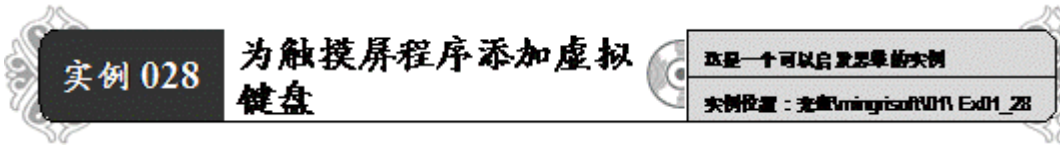


图 1.28 为触摸屏程序添加虚拟键盘



实例 028 为触摸屏程序添加虚拟键盘

实例说明

由于触摸屏没有键盘，只能利用屏幕操作。如果要输入数据或查找数据，需要制作一个虚拟键盘，以方便用户输入。本例介绍如何实现虚拟键盘的程序设计。运行本例，效果如图 1.28 所示。

技术要点

本例中用到了 Lable 控件的透明属性和字符串截取技术。主要是使用 Substring()方法。下面详细介绍一下该方法。

Substring()方法用来从此实例检索子字符串。子字符串从指定的字符位置开始且具有指定的长度。其语法结构如下：

```
public string Substring (int startIndex,int length)
```

参数说明如下。

- startIndex：子字符串起始位置的索引。
- length：子字符串中的字符数。
- 返回值：一个 String，等于此实例中从 startIndex 开始的长度为 length 的子字符串，如果 startIndex 等于此实例的长度且 length 为零，则为 Empty。

实现过程

（1）创建一个项目，将其命名为 Ex01_28，默认窗体为 Form1。

（2）在窗体上添加 2 个 Panel 控件，1 个 TextBox 控件和许多 Label 控件，每个 Label 控件对应背景图片上的 1 个按钮。

（3）为 Panel 控件添加背景图片，并将 Label 控件的 BackColor 设置为透明。

（4）主要程序代码。

```
private void Form1_Load(object sender, EventArgs e)
{
    lbl_0.Click += new EventHandler(lbl_Click);
    lbl_1.Click += new EventHandler(lbl_Click);
    lbl_2.Click += new EventHandler(lbl_Click);
    lbl_3.Click += new EventHandler(lbl_Click);
    lbl_4.Click += new EventHandler(lbl_Click);
    lbl_5.Click += new EventHandler(lbl_Click);
    lbl_6.Click += new EventHandler(lbl_Click);
    lbl_7.Click += new EventHandler(lbl_Click);
    lbl_8.Click += new EventHandler(lbl_Click);
    lbl_9.Click += new EventHandler(lbl_Click);
    lbl_Q.Click += new EventHandler(lbl_Click);
    lbl_W.Click += new EventHandler(lbl_Click);
    lbl_R.Click += new EventHandler(lbl_Click);
    lbl_E.Click += new EventHandler(lbl_Click);
    lbl_T.Click += new EventHandler(lbl_Click);
    lbl_Y.Click += new EventHandler(lbl_Click);
    lbl_U.Click += new EventHandler(lbl_Click);
    lbl_I.Click += new EventHandler(lbl_Click);
    lbl_O.Click += new EventHandler(lbl_Click);
    lbl_P.Click += new EventHandler(lbl_Click);
    lbl_A.Click += new EventHandler(lbl_Click);
    lbl_S.Click += new EventHandler(lbl_Click);
    lbl_D.Click += new EventHandler(lbl_Click);
    lbl_F.Click += new EventHandler(lbl_Click);
    lbl_G.Click += new EventHandler(lbl_Click);
    lbl_H.Click += new EventHandler(lbl_Click);
    lbl_J.Click += new EventHandler(lbl_Click);
    lbl_K.Click += new EventHandler(lbl_Click);
    lbl_L.Click += new EventHandler(lbl_Click);
    lbl_Z.Click += new EventHandler(lbl_Click);
    lbl_X.Click += new EventHandler(lbl_Click);
    lbl_C.Click += new EventHandler(lbl_Click);
    lbl_V.Click += new EventHandler(lbl_Click);
    lbl_B.Click += new EventHandler(lbl_Click);
    lbl_N.Click += new EventHandler(lbl_Click);
    lbl_M.Click += new EventHandler(lbl_Click);
    label44.Click += new EventHandler(label44_Click);
}
```

将选中的数字或字母加入 TextBox.Text 的实现代码如下：

```
void lbl_Click(object sender, EventArgs e)
{
   Label l = (Label)sender;
   textBox1.Text += l.Name.Substring(4, 1);
   textBox1.SelectionStart = textBox1.Text.Length;
}
```

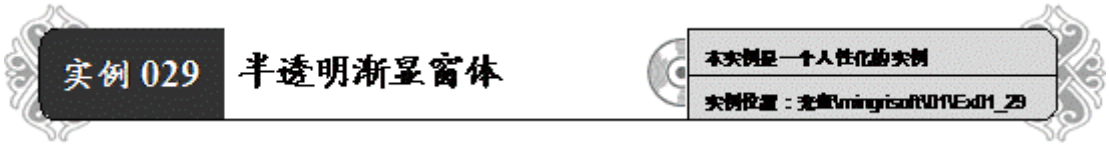
举一反三

根据本实例，读者可以实现以下程序。

- 制作注册控件的程序。
- 制作网络程序复制的程序。

1.8 窗 体 效 果

本节主要是对窗体的透明度、颜色渐变、背景及边框的相关技术进行讲解。在项目开发中窗体的设计会影响用户对软件的整体印象，所以窗体的效果要设计的美观一些。下面将介绍一些常用的效果。



实例 029　半透明渐显窗体

实例说明

很多专业软件在启动前都会显示一个说明该软件信息或用途的窗口，有的则是一个漂亮的启动界面，如 Adobe 公司的 Acrobat。该窗口使软件显得更加专业。本例将实现一个半透明的渐显窗体，运行本软件会显示一个启动画面，并且画面会将完全透明慢慢到半透明的效果显示在用户面前。效果如图 1.29 所示。

图 1.29　半透明溅显窗体

技术要点

在其他开发环境中，实现窗体的半透明渐显需要调用 API 函数，实现非常困难，但在 C# 2.0 中，窗体提供了 Opacit 属性来设置窗体的透明度。

Form.Opacit 属性用来获取或设置窗体的不透明度级别，其语法格式如下：

```
public double Opacity { get; set; }
```

l　属性值：窗体的不透明度级别。默认值为 1.00。

实现过程

（1）创建一个项目，将其命名为 Ex01_29，默认窗体为 Form1。

（2）在 Form1 窗体中设置背景图片，添加 Timer 控件用来触发渐变事件。

（3）设置 Timer 控件的 Enable 属性为 True，设置 Interval 属性为 1000。

（4）主要程序代码。

```
private void timer1_Tick(object sender, EventArgs e)
{
    this.Opacity += 0.1;
}
```

举一反三

根据本实例，读者可以开发以下程序。

- 使窗体由透明状态显现出来，可以用 Time 控件来控制窗体由透明到显示所需的时间。
- 可以使窗体为透明状态，只显示窗体上的控件。

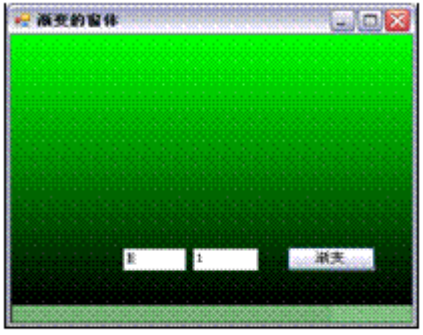
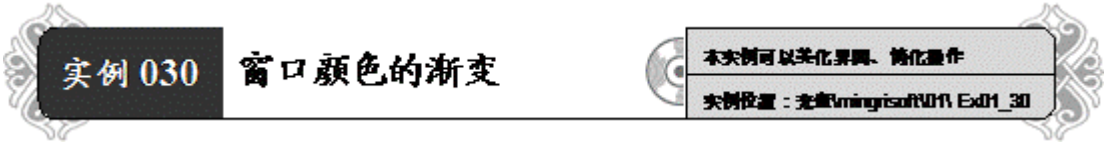


图 1.30 窗口颜色的渐变



实例 030　窗口颜色的渐变

实例说明

在程序设计时，可以通过设置窗体的 BackColor 属性来改变窗口的背景颜色。但是这个属性改变后整个窗体的客户区都会变成这种颜色，并且非常单调。如果窗体的客户区可以向标题栏一样能够体现颜色的渐变效果，那么窗体风格将会另有一番风味。本例设计了一个颜色渐变的窗体。效果如图 1.30 所示。

技术要点

C#中可以通过 Color.FromArgb(　)方法返回一种颜色，下面详细介绍一下该方法。

Color.FromArgb(　)方法用来返回 Color 的颜色值，该方法语法结构如下：

```
public static Color FromArgb (
    int red,
    int green,
    int blue
)
```

参数说明如下。

- red：新 Color 的红色分量值。有效值为从 0～255。
- green：新 Color 的绿色分量值。有效值为从 0～255。
- blue：新 Color 的蓝色分量值。有效值为从 0～255。
- 返回值：此方法创建的 Color。

该函数就是用 3 种不同的色值来返回一个颜色，而稍微的调整某一种颜色值就可以使整体的颜色发生细微的变化，在窗体中至上而下每行填充一种稍微调整

后的颜色，这样整体看来就会产生渐变的效果。可以利用窗体的 Graphics 对象对窗体进行绘图，该对象可以完全操控窗体的客户区。

 注意：颜色值在 0～255 之间。

实现过程

（1）创建一个项目，将其命名为 Ex01_30，默认窗体为 Form1。

（2）在 Form1 窗体中添加 Button 用来使颜色渐变；添加 TextBox 控件用来输入颜色 RGB 值。

（3）主要程序代码。

触发重新绘制事件的实现代码如下：

```
private void button2_Click(object sender, EventArgs e)
{
    InvokePaintBackground( );
    this.Hide( );
    this.Visible=true;
}
```

重新绘制窗体背景颜色的实现代码如下：

```
protected override void OnPaintBackground(PaintEventArgs e)
{
    int y, dy;
    y = this.ClientRectangle.Location.Y;
    dy = this.ClientRectangle.Height / 256;
    for (int i = 255; i >= 0; i--)
    {

        Color c = new Color( );
        c = Color.FromArgb(Convert.ToInt32(textBox1.Text.ToString( )), i,Convert.T
oInt32(textBox2.Text.ToString( )));
```

```
            SolidBrush sb = new SolidBrush(c);
            Pen p = new Pen(sb, 1);
            e.Graphics.DrawRectangle(p,this.ClientRectangle.X, y, this.Width,y+dy);
            y = y + dy;
        }
    }
```

举一反三

根据本实例，读者可以开发以下程序。

- 把窗体设置成单一的颜色。
- 利用 Timer 组体，使窗体动态改变颜色。

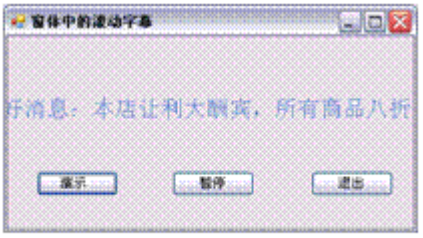
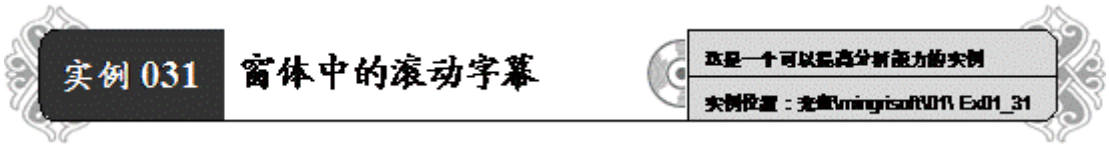


图 1.31 窗体中的滚动字幕

1.9 窗 体 动 画

本节主要对窗体进行动画设置，在窗体上添加一些动画效果，可以为操作者添加一些乐趣，下面的几个例子将详细介绍窗体动画的相关技术。



实例 031 窗体中的滚动字幕

实例说明

普通窗体中的文字位置都是固定的，一些窗体中需要让文字动起来，例如一些广告性较强的界面中需要做一些滚动的字幕。本例实现了一个具有滚动字幕效果的窗体，运行本例，单击【演示】按钮，看到窗口中的文字开始滚动。单击【暂停】按钮，可以使字幕停止滚动。本例运行效果如图 1.31 所示。

技术要点

滚动字幕的效果其实就是改变了文字的位置，在窗体中显示一串文字最好的办法就是利用 Label 控件。将 Label 控件的位置改变就可以实现文字的位置变换，

如果该控件的位置不断的向水平方向移动，就会实现文字的滚动效果。改变 Label 控件的水平位置可以通过改变 Label 控件的 Left 的值来实现。用 Timer 控件对文字的移动进行时间控制。

实现过程

（1）创建一个项目，将其命名为 Ex01_31，默认窗体为 Form1。

（2）在窗体上添加 Label 控件用来显示消息；添加 Button 控件用来控制消息的运动；添加 Timer 控件用来控制滚动速度。

（3）主要程序代码。

```
private void timer1_Tick(object sender, EventArgs e)//用 Timer 来控制滚动速度
{

   label1.Left -= 2;
   if (label1.Right < 0)
   {
      label1.Left = this.Width;
   }
}
private void button1_Click(object sender, EventArgs e)
{
   timer1.Enabled = true; //开始滚动
}
private void button2_Click(object sender, EventArgs e)
{
   timer1.Enabled = false; //停止滚动
}
```

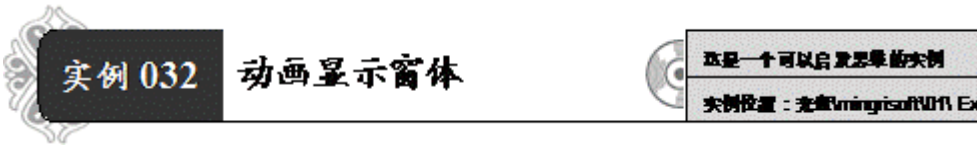

注意：要特别注意文字滚动的方向问题，向左则减，向右则加。

举一反三

图 1.32 动画显示窗体

根据本实例，读者可以开发以下程序。

- 可以在窗体中设置一个滚动的图片。
- 可以在窗体中设置一个滚动的提示信息。



实例 032　动画显示窗体

实例说明

当用户启动程序后，普通的程序窗口都是瞬间显示到屏幕上，这样未免有些生硬。如果窗口能够慢慢的展现在用户面前，将会是什么样的效果？本例设计的是一个动画显示的窗体，该程序运行后，窗体是慢慢的以拉伸的效果显示到用户的面前。当关闭时也是一样慢慢的消失。本例运行效果如图 1.32 所示。

技术要点

Windows 提供了一个 API 函数 Animate Window，该函数可以实现窗体的动画效果，AnimateWindow 函数在 C#中的声明如下。

```
[DllImportAttribute("user32.dll")]
private static extern bool AnimateWindow(IntPtr hwnd, int dwTime, int dwFlags);
```

参数说明如下。

l　hwnd：目标窗口句柄。

l　dwTime：动画的持续时间，数值越大动画效果的时间就越长。

l　DwFlags：DwFlags 参数是动画效果类型选项，该参数在 C#中的声明如下：

```
public const Int32 AW_HOR_POSITIVE = 0x00000001;
public const Int32 AW_HOR_NEGATIVE = 0x00000002;
public const Int32 AW_VER_POSITIVE = 0x00000004;
public const Int32 AW_VER_NEGATIVE = 0x00000008;
public const Int32 AW_CENTER = 0x00000010;
public const Int32 AW_HIDE = 0x00010000;
```

```
public const Int32 AW_ACTIVATE = 0x00020000;
public const Int32 AW_SLIDE = 0x00040000;
public const Int32 AW_BLEND = 0x00080000;
```

DwFlags 参数可选值含义如表 1.1 所示

表 1.1　　参数说明

标　志	描　述
AW_SLIDE	使用滑动类型。缺省则为滚动动画类型。当使用 AW_CENTER 标志时，这个标志就被忽略
AW_ACTIVE	激活窗口。在使用了 AW_HIDE 标志后不要使用这个标志
AW_BLEND	使用淡入效果。只有当 hWnd 为顶层窗口的时候才可以使用此标志
AW_HIDE	隐藏窗口，缺省则显示窗口
AW_CENTER	若使用了 AW_HIDE 标志，则使窗口向内重叠；若未使用 AW_HIDE 标志，则使窗口向外扩展
AW_HOR_POSITIVE	自左向右显示窗口。该标志可以在滚动动画和滑动动画中使用。当使用 AW_CENTER 标志时，该标志将被忽略
AW_HOR_NEGATIVE	自右向左显示窗口。当使用了 AW_CENTER 标志时该标志被忽略
AW_VER_POSITIVE	自顶向下显示窗口。该标志可以在滚动动画和滑动动画中使用。当使用 AW_CENTER 标志时，该标志将被忽略
AW_VER_NEGATIVE	自下向上显示窗口。该标志可以在滚动动画和滑动动画中使用。当使用 AW_CENTER 标志时，该标志将被忽略

实现过程

（1）创建一个项目，将其命名为 Ex01_32，默认窗体为 Form1。

（2）在窗体上添加 PictureBox 控件。

（3）设置 PictureBox 控件的 Image 属性。

（4）主要代码如下。

```
public Form1( )
{
   InitializeComponent( );
   AnimateWindow(this.Handle, 300, AW_SLIDE + AW_VER_NEGATIVE);//开始窗体动画
}
private void Form1_FormClosed(object sender, FormClosedEventArgs e)
{ //结束窗体动画
```

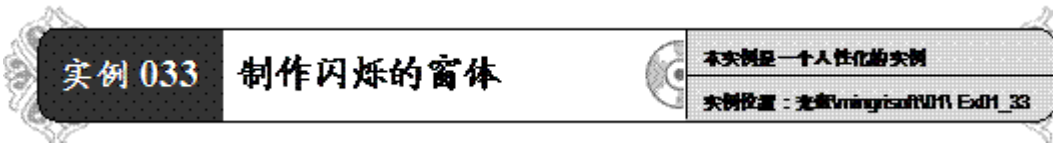
```
        AnimateWindow(this.Handle, 300, AW_SLIDE + AW_VER_NEGATIVE + AW_H
IDE);
    }
```

举一反三

根据本实例，读者可以开发以下程序。

- 实现窗体的淡入淡出。
- 实现窗体从中间扩散显示。

图 1.33 制作闪烁的窗体



实例 033 制作闪烁的窗体

实例说明

Windows 系统中，当程序在后台运行时，如果某个窗口的提示信息需要用户浏览，该窗口就会不停的闪烁，这样就会吸引用户的注意。同样，如果在自己的程序中使某个窗口不停的闪烁就会吸引用户的注意。本例设计了一个闪烁的窗体，运行程序，单击【开始闪烁】按钮，窗体就会不停的闪烁，单击【停止】按钮，窗体就会停止闪烁。本例运行效果如图 1.33 所示。

技术要点

Windows 提供了一个 API 函数 FlashWIndow，该函数可以使窗体闪烁一下。FlashWIndow 函数在 C#中声明如下：

```
[System.Runtime.InteropServices.DllImportAttribute("user32.dll")]
public static extern bool FlashWindow(IntPtr handle, bool bInvert);
```

参数说明如下。

- handle：表示将要闪烁的窗体。
- bInvert：是否恢复状态。

利用该函数只能使窗体闪烁一下，如果让窗口不停地闪烁，就需要用一个 Timer 控件每隔一段时间就调用该函数使窗体闪烁。

实现过程

（1）创建一个项目，将其命名为 Ex01_33，默认窗体为 Form1。

（2）在窗体上添加 PictureBox 控件用来显示窗体；添加 Button、Timer 控件用来开始和停止闪烁。

（3）设置 PictureBox 控件的 Image 属性。

（4）主要程序代码。

timer1 的 Tick 事件处理代码如下：

```
private void timer1_Tick(object sender, EventArgs e)
{
   FlashWindow(this.Handle,true);
}
```

【开始闪烁】按钮的单击事件，用来启动窗体闪烁：

```
private void button1_Click(object sender, EventArgs e)
{
   timer1.Enabled = true;
}
```

【停止】按钮的单击事件，用来停止窗体的闪烁：

```
private void button2_Click(object sender, EventArgs e)
{
   timer1.Enabled = false;
}
```

举一反三

根据本实例，读者可以开发以下程序。

- 利用 Visible 属性制作一个闪烁的图片。
- 制作一个闪烁的按钮。

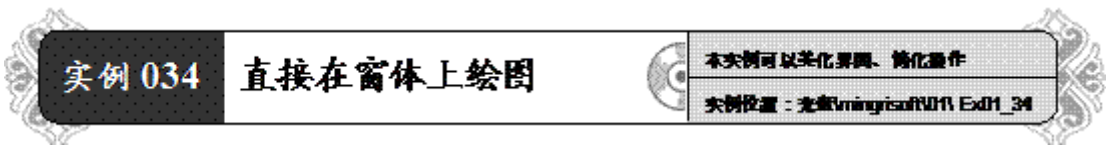


图 1.34 直接在窗体上绘图

实例 034 直接在窗体上绘图

实例说明

含有 Graphics 对象的控件都能够在其上进行绘图，很多软件就是通过 Graphics 对象来美化程序的主界面，因为窗体中含有 Graphics 对象，所以可以将窗体看作一个大画板，一个可以在上面绘图的特殊控件。本例设计了一个简单的绘图软件，该软件就利用了在窗体上绘图的方法，运行本软件可以在窗体上进行绘图。实例效果如图 1.34 所示。

技术要点

窗体中含有 Graphics 对象，使用该对象就能够完成大部分绘图功能，Graphics 对象已经对 Windows 底层的一些绘图 API 进行了封装，使用起来比较方便。下面介绍 Graphics 对象的常用方法。

Graphics.DrawLine 绘图方法用来绘制一条连接由坐标对指定的两个点的线条。其语法结构如下：

```
public void DrawLine (Pen pen,int x1,int y1,int x2,int y2)
```

参数说明如下。

- pen：Pen 对象，确定线条的颜色、宽度和样式。
- x1：第一个点的 x 坐标。
- y1：第一个点的 y 坐标。
- x2：第二个点的 x 坐标。
- y2：第二个点的 y 坐标。

实现过程

（1）创建一个项目，将其命名为 Ex01_34，默认窗体为 Form1。

（2）向 Form1 窗口中添加 GroupBox 控件，用作 RadioButton 控件的容器；添加 Button 控件用来推出程序。

（3）主要程序代码。

在窗体单元的 private 中添加变量如下：

```
int startX,startY;
```

```
Graphics g;
```

单击鼠标事件。具体代码如下：

```
private void Form1_MouseDown(object sender, MouseEventArgs e)
{
   startX=e.X;
   startY = e.Y;
}
```

鼠标在窗体中的移动事件。具体代码如下：

```
private void Form1_MouseMove(object sender, MouseEventArgs e)
{
    g = this.CreateGraphics( );
   Pen p = new Pen(Color.Black, 1);
   if(radioButton2.Checked==true)
   {
      g.DrawRectangle(p, e.X, e.Y, 1, 1);
   }
}
```

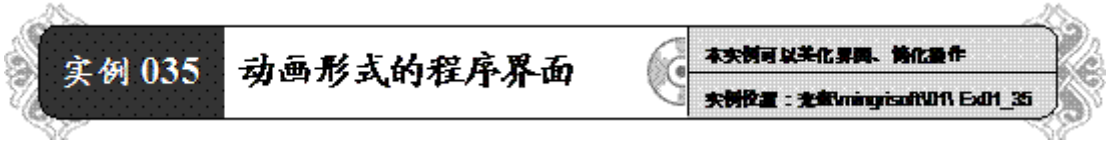
鼠标抬起事件。具体代码如下：

```
private void Form1_MouseUp(object sender, MouseEventArgs e)
{
   g = this.CreateGraphics( );
   Pen p = new Pen(Color.Black, 2);
   if (radioButton1.Checked == true )
   {
      g.DrawLine(p, startX, startY, e.X, e.Y);
   }
}
```

举一反三

根据本实例，读者可以实现以下功能。

- 改变画笔的颜色。
- 在窗体上绘制矩型。



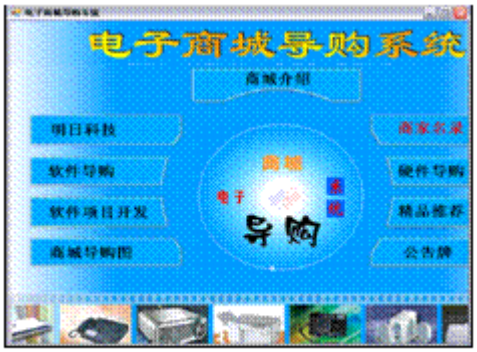


图 1.35 动画形式的程序界面

实例 035 动画形式的程序界面

实例说明

在很多的程序界面中，都是以菜单或工具栏的形式显示窗体界面，这种显示方式是以静止状态显示的，界面不够生动。下面介绍一个以动画显示窗体界面的设计方法。运行本例，效果如图 1.35 所示。

技术要点

在该实例中用到了 Microsoft Animation Control 6.0（SP4）COM 组件，所以要从工具箱“选择项”中将该组件添加到工具箱，然后继续将该组件从工具箱添加到窗体即可。下面介绍本例中用到的相关方法。

AxAnimation.open 方法用来播放动画文件。其结构如下：

```
Public void AxAnimation.open(string bstrFilename)
```

参数说明如下。

l bstrFilename：将要播放的文件名。

注意：因为使用了 AxAnimation 类，所以要添加对 WMPLib 命名空间的引用。

实现过程

（1）创建一个项目，将其命名为 Ex01_35，默认窗体为 Form1。

（2）在 Form1 窗体添加 PictureBox 控件用来显示图片，添加 Microsoft Animation Control 6.0 （SP4）COM 组件用来播放动画。

（3）主要程序代码。

```
private void Form1_Load(object sender, EventArgs e)
{
    axAnimation1.Open("Electron.avi");
    axAnimation2.Open("zybiao.avi");
    axAnimation3.Open("gd.avi");
}
```

图 1.36 使窗体标题栏文字右对齐

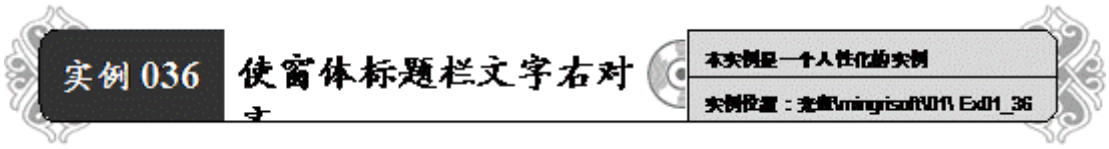
举一反三

根据本实例，读者可以实现以下功能。

- 制作摸拟网页。
- 制作动画播放器。

1.10 标题栏窗体

本节主要是对窗体的标题栏进行设置，标题栏是一个显著的位置，在这个位置添加按钮或进行一些个性化的设置，都会给人一种新奇的感觉。通过以下实例的学习，读者将掌握此技术。



实例 036 使窗体标题栏文字右对齐

实例说明

窗口标题栏中的文字是窗口的重要说明，该文字可以标示窗口的功能、状态或名称等信息，一般该文字是居左显示的，在本例中设计一个标题栏文字右对齐的窗口。本实例运行结果如图 1.36 所示。

技术要点

在 C# 2.0 中实现这一功能非常容易，只需将窗体的 RightToLeft 属性设置为 Yes 即可。

Form. RightToLeft 属性用来获取或设置一个值，该值指示是否将控件的元素对齐以支持使用从右向左的字体的区域设置，其语法结构如下：

```
public virtual RightToLeft RightToLeft { get; set; }
```

l 属性值：RightToLeft 值之一。默认为 Inherit。

实现过程

（1）创建一个项目，将其命名为 Ex01_36，默认窗体为 Form1。

图 1.37 没有标题栏改变大小的窗口

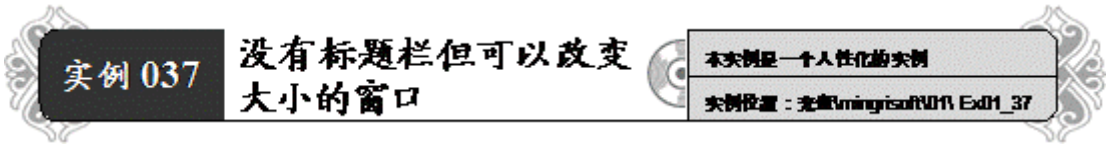
（2）为 Form1 窗体添加背景图片。

（3）设置 RightToLeft 属性为 Yes。

举一反三

根据本实例，读者可以开发以下程序。

- 利用 Timer 控件使窗体标题栏的文字进行左右闪动。
- 制作窗体标题栏滚动播放图片的窗体。



实例 037　没有标题栏可义改变大小的窗口

实例说明

隐藏 Windows 窗口的标题栏之后，窗口只剩下一个客户区域，有点像 Panel 控件在窗口中的样子，而这样的窗口通常是不能够改变大小的。因为屏蔽其标题栏之后，窗口默认将边框也去除了，本例将用特殊的方法建立一个没有标题栏但是可以改变其大小的窗体。实例运行效果如图 1.37 所示。

技术要点

窗口的样式是在窗口建立时确定的，在 C#中实现窗体没有标题栏但是可以改变大小的窗口，有一个巧妙的方法就是将窗体的 Text 属性设为空，同时将 ControlBox 属性设为 False。下面介绍一下相关的属性。

ControlBox 属性用来获取或设置一个值，该值指示在该窗体的标题栏中是否显示控件框，其语法结构如下：

```
public bool ControlBox { get; set; }
```

l　属性值：如果该窗体在窗体的左上角显示控件框，则为 True；否则为 False。默认为 True。

实现过程

（1）创建一个项目，将其命名为 Ex01_37，默认窗体为 Form1。

图 1.38 设置窗体在屏幕中的位置

（2）在 Form1 窗口中添加 Label、Button 控件，用来设计界面。

（3）主要程序代码。

```
private void Form1_Load(object sender, EventArgs e)
{
    ControlBox = false;
}
```

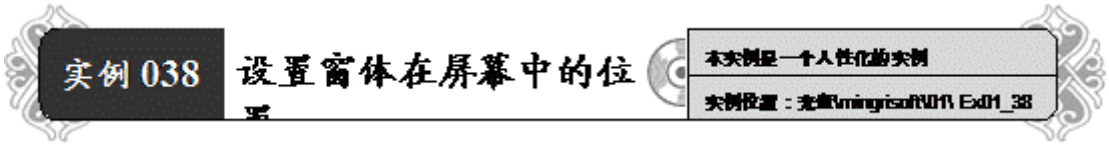
注意：必须将窗体的 Text 属性设为空。

举一反三

根据本实例，读者可以开发以下程序。

- 在窗体显示时最小化。
- 在窗体显示时最大化。**1.11 设置窗体位置**

在许多的软件中，都会对窗体的大小、位置和移动进行限定。在不同分辨率的显示器中如何正确显示窗体、如何设置窗体始终在最上面，这些都需要本节的技术。



实例 038 设置窗体在屏幕中的位置

实例说明

在窗体中可以设置窗体居中显示，本例通过设置窗体的 Left 属性和 Top 属性可以准确设置窗体的位置。运行本例，效果如图 1.38 所示。

技术要点

设置窗体在屏幕中的位置，可以通过设置窗体的属性来实现。窗体的 Left 属性表示窗体距屏幕左侧的距离，Top 属性表示窗体距屏幕上方的距离。

图 1.39 始终在最上面的窗体

实现过程

（1）创建一个项目，将其命名为 Ex01_38，默认窗体为 Form1。

（2）在窗体上添加 Label 控件；添加 TextBox 控件用来输入距屏幕的距离；添加 Button 控件用来设置窗体在屏幕上的位置。

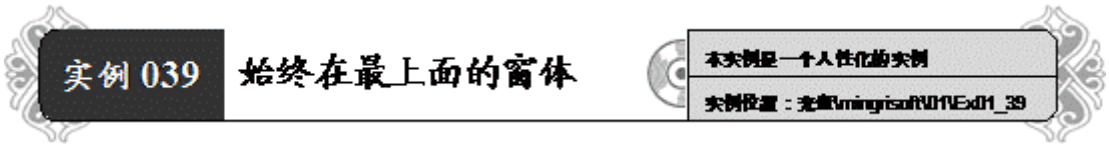
（3）主要程序代码。

```
private void button1_Click(object sender, EventArgs e)
{
    this.Left = Convert.ToInt32(textBox1.Text);
    this.Top = Convert.ToInt32(textBox2.Text);
}
```

举一反三

根据本实例，读者可以开发以下程序。

- 根据分辨率的变化动态设置窗体位置。
- 用 Timer 控件实时显示窗体位置。



实例 039　始终在最上面的窗体

实例说明

Windows 桌面上允许多个窗体同时显示，但是只有一个窗体能够得到焦点，当一个窗体得到焦点后在其上面的窗体会被得到焦点的窗体遮挡，得到焦点的窗体会显示在最上层，这样被覆盖的窗体就不能完全的显示给用户，如果该窗体中具有实时性和比较重要的信息时，需要该窗口始终在最上层。本例就实现了此功能，运行本例后，主窗体会始终在桌面的最上面。实例效果如图 1.39 所示。

技术要点

在其他开发环境中实现窗体始终在最上面比较复杂，但在 C# 2.0 中实现非常简单，只要将 TopMost 属性设为 True 即可。下面介绍一下 TopMost 属性。

Form.TopMost 属性用来获取或设置一个值，指示该窗体是否应显示为最顶层窗体。其结构如下：

```
public bool TopMost { get; set; }
```

l 属性值：如果将窗体显示为最顶层窗体，则为 True；否则为 False。默认为 False。

实现过程

（1）创建一个项目，将其命名为 Ex01_39，默认窗体为 Form1。

（2）为 Form1 窗体添加背景图片，并设置窗体 TopMost 属性为 True。

举一反三

根据本实例，读者可以开发以下程序。

- 可以将设为最上层的窗体设置成为一个电子表，以便观看时间。
- 可以将设为最上层的窗体设置成为一个工作计划表，以便随时提醒自己。

1.12 设置窗体大小

用户打开软件后首先看到的就是窗体和窗体上的控件，如何设置窗体的大小及合理的设置窗体和控件的关系就变得十分重要，下面的实例将介绍这方面的知识。

图 1.40 限制窗体大小

实例 040 限制窗体大小

实例说明

Windows 窗体是可以随意改变大小的，然而对于一些要求严格的窗体，开发人员不希望用户随意的改变其大小，例如，定位准确的地图和游戏软件等。遇到这种情况必须对窗口的大小进行一些限制。本例设计一个限制了大小的窗

体，用户虽然可以改变其大小，但是，大小的范围是受到限制的。实例效果如图1.40所示。

技术要点

在此C#中实现限制大小非常方便，只要设置窗体的最大和最小范围即可。下面介绍一下相关属性。

Form.MinimumSize属性用来获取或设置窗体可调整到的最小大小，其语法格式如下：

```
public override Size MinimumSize { get; set; }
```

l 属性值：Size，表示该窗体的最小大小。

Form.MaximumSize属性用来获取或设置窗体可调整到的最大大小，其语法格式如下：

```
public override Size MaximumSize{ get; set; }
```

l 属性值：Size，表示该窗体的最大大小。

实现过程

（1）创建一个项目，将其命名为Ex01_27，默认窗体为Form1。

（2）主要程序代码。

```
private void Form1_Load(object sender, EventArgs e)
{
    MinimumSize = new Size(200, 200);
    MaximumSize = new Size(400, 400);
}
```

举一反三

根据本实例，读者可以开发以下程序。

- 在窗体显示时规定其大小。
- 在窗体运行时规定其大小。

图 1.41 获取桌面大小

实例 041 获取桌面大小

实例说明

获取桌面分辨率可以使用 API 函数 GetDeviceCaps，但 API 函数参数较多，使用不方便，如何更方便的获取桌面分辨率呢？在本例中，通过读取 Screen 对象的属性，来获取桌面分辨率信息，以像素为单位。运行本例，效果如图 1.41 所示。

技术要点

C#中提供了 Screen 对象，在该对象中封装了屏幕相关信息。可以通过读取 Screen 对象的相关属性，来获取屏幕的信息，Screen.PrimaryScreen.WorkingArea.Width 用于读取桌面宽度；Screen.PrimaryScreen.WorkingArea.Height 可以读取桌面的高度。下面介绍一下相关属性。

Screen.PrimaryScreen.WorkingArea 属性用于获取显示器的工作区。工作区是显示器的桌面区域，不包括任务栏、停靠窗口和停靠工具栏。其结构如下：

```
public Rectangle WorkingArea { get; }
```

l 属性值：一个 Rectangle，表示显示器的工作区。

实现过程

（1）创建一个项目，将其命名为 Ex01_41，默认窗体为 Form1。

（2）在 Form1 窗体上添加一个 Button 控件，用来获取桌面大小；添加两个 TextBox 控件，用来输出所获取的桌面大小。

（3）主要程序代码。

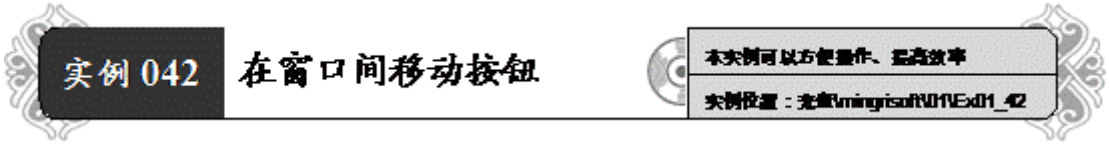
```
private void button1_Click(object sender, EventArgs e)
{
    textBox2.Text = Screen.PrimaryScreen.WorkingArea.Height.ToString( );
```

```
        textBox1.Text = Screen.PrimaryScreen.WorkingArea.Width.ToString( );
    }
```

举一反三

根据本实例，读者可以开发以下程序。

- 根据显示器的分辨率信息设置窗体大小及位置。
- 根据显示器的分辨率信息调整窗体界面。



实例 042　在窗口间移动按扭

实例说明

窗体中每个可视控件都有所有者和父对象两个重要属性，所有者是控件建立时指定的所属对象，该对象可以是不可视控件，而父对象必须是可视控件。因此可以通过窗体中可视控件的 Parent 属性来判断控件是否在这个窗体中，还可以用 Form.Controls.Add(　)方法为窗体添加控件。本例以一个可以在窗口间移动的按钮来演示父对象改变后的运行效果。运行本例，在窗口中单击按钮，按钮就会移动到另外一个窗口中。实例效果如图 1.42 和图 1.43 所示。

图 1.42　在窗口间移动按钮

图 1.43　在窗口间移动按钮

技术要点

可视控件包含一个 Parent 属性，该属性表示控件的父对象。一般将此属性设置为一个窗口。通过该属性可以控制所属窗体。

实现过程

（1）创建一个项目，将其命名为 Ex01_42，默认窗体为 Form1。

（2）添加一个窗体，默认窗体的 Name 属性为 Form2。

（3）在 Form1 窗口中添加一个 Button 控件。并为 Form1 和 Form2 设置背景图片。

（4）主要程序代码。

单击按钮在两个窗体之间移动，具体代码如下：

```
private void button1_Click(object sender, EventArgs e)
{
   if (button1.Parent == this)
   {
      f.Controls.Add(this.button1);
      this.button1.Text = "返回原地";
   }
   else
   {
      this.Controls.Add(button1);
      this.button1.Text = "开始移动";
   }
}
```

Form1 窗体加载时同时显示 Form2 窗体，具体代码如下：

```
private void Form1_Load(object sender, EventArgs e)
{
   f = new Form2( );
   f.Show( );
}
```

举一反三

根据本实例，读者可以开发以下程序。

- 试做从一个窗体将控件拖到另一个窗体。
- 试做用一个窗体控制另一个窗体。

图 1.44 如何实现 Office 助手

实例 043 如何实现 Office 助手

实例说明

用过 Office 的人都知道，Office 助手是一个非常漂亮的小工具，有了它，即使对 Office 不太熟悉的用户也可以操作自如。本实例使用 C#制作了一个类似 Office 助手的程序，实例效果如图 1.44 所示。

技术要点

要实现 Office 助手效果，需要使用 Microsoft 提供的第 3 方控件。在工具箱中单击“选择项”，从弹出的对话框中选择 COM 组件选项卡中的 Microsoft Agent Control 2.0 组件并加入工具箱中，然后再添加到窗体中。

实现过程

（1）创建一个项目，将其命名为 Ex01_43，默认窗体为 Form1。

（2）在 Form1 窗体上添加一个 ListBox 控件用来让用户选择人物的动作。

（3）主要程序代码。

声明成员变量及字符串数组，具体代码如下：

```
IAgentCtlCharacterEx ICCE;
IAgentCtlRequest ICR;
string[] ws = new string[10] { "Acknowledge", "LookDown", "Sad", "Alert", "LookDownBlink", "Search", "Announce", "LookUp", "Think", "Blink"};
```

为 ListBox 添加选项的实现代码如下：

```
private void Form1_Load(object sender, EventArgs e)
{
    for (int i = 0; i < 10; i++)
    {
```

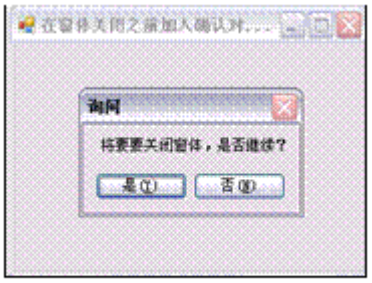


图 1.45 在关闭窗口前加入确认对话框

```
            listBox1.Items.Add(ws[i]);
        }
        ICR = axAgent1.Characters.Load("merlin", "merlin.acs");
    ICCE = axAgent1.Characters.Character("merlin");
    ICCE.Show(0);
}
```

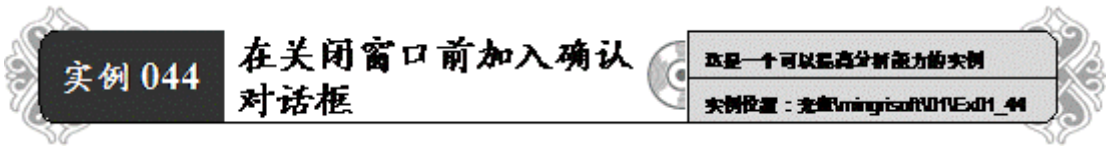
随着选项改变 Office 表情的实现代码如下：

```
private void listBox1_SelectedIndexChanged(object sender, EventArgs e)
{
    ICCE.StopAll("");
    ICCE.Play(ws[listBox1.SelectedIndex]);
}
```

举一反三

根据本实例，读者可以实现以下程序。

- 瑞星助手。
- 在自己的程序中加入 Office 助手。



实例 044　在关闭窗口前加入确认对话框

实例说明

用户对程序进行操作时，难免会有错误操作的情况，例如不小心关闭程序，如果尚有许多资料没有保存，那么损失将非常严重，所以最好使程序具有灵活的交互性。人机交互过程一般都是通过对话框来实现的，对话框中有提示信息，并且提供按钮让用户选择，例如【是】或【否】。这样用户就能够对所做的动作进行确认。正如前面所说的不小心关闭程序，如果在关闭程序之前提示用户将要关

闭程序，并且提供用户选择是否继续下去，这样就大大减少了误操作现象。本例程序中的窗口在关闭时会显示一个对话框，该对话框中有两个按钮【是】与【否】代表是否同意关闭程序操作。实例运行结果如图 1.45 所示。

技术要点

窗口正要关闭但是没有关闭之前会触发 FormClosing 事件，该事件中的参数 FormClosingEventArgs e 中包含 Cancel 属性，如果设置该属性为 True，窗口将不会被关闭。所以在该事件处理代码中可以提示用户是否关闭程序，如果用户不想关闭程序，则设置该参数为 True。利用 MessageBox 参数的返回值可以知道用户所选择的按钮。下面详细介绍一下相关属性。

CancelEventArgs.Cancel 属性用来获取或设置指示是否应取消事件的值。该属性结构如下：

```
public bool Cancel { get; set; }
```

l 属性值：如果应取消事件，则为 True；否则为 False。

实现过程

（1）创建一个项目，将其命名为 Ex01_44，默认窗体为 Form1。

（2）主要程序代码。

```
private void Form1_FormClosing(object sender, FormClosingEventArgs e)
{
    if (MessageBox.Show("将要要关闭窗体，是否继续？", "询问", MessageBoxButtons.YesNo) == DialogResult.Yes)
    {
        e.Cancel = false;
    }
    else
    {
        e.Cancel = true;
    }
}
```

举一反三

根据本实例，读者可以实现以下程序。

- 使窗体的关闭按钮无效。
- 使窗体关闭出现在托盘中。

图 1.46 使用任意组件拖动窗体

实例 045 使用任意组件拖动窗体

实例说明

通常将鼠标按住窗口的标题栏才能够拖动窗口，但是，在没有窗口标题栏的情况下如何拖动窗体呢？本例将会利用窗口中的控件拖动窗口，将鼠标放在按钮上然后按住鼠标左键移动鼠标即可拖动窗体。实例效果如图 1.46 所示。

技术要点

通过控件移动窗体时，需要判断用户的鼠标动作。用户准备拖动窗体时必须在控件上按住鼠标左键，所以应该在鼠标 MouseDown 事件处理过程中来实现窗体的拖动。当用户在按钮上将鼠标左键按下时，触发 MouseDown 事件，在该事件处理代码中，MouseEventArgs e 的 Button 属性记录了当前按下的鼠标按钮，如果按键是鼠标左键，则表示可以移动窗口，鼠标移动时，窗体就可以跟着移动了。

实现过程

（1）创建一个项目，将其命名为 Ex01_45，默认窗体为 Form1。

（2）在 Form1 窗体上添加两个 Button 控件，分别用来拖动窗体和关闭窗体。然后设置窗体的背景颜色。

（3）主要程序代码。

声明记录鼠标按下时初始位置的变量，具体代码如下：

```
private int startX, StartY;
```

鼠标按下事件处理代码，具体代码如下：

```
private void button1_MouseDown(object sender, MouseEventArgs e)
{
    if (e.Button == MouseButtons.Left)
    {
        startX = e.X;
        StartY = e.Y;
    }
```

图 1.47 修改提示字体及颜色

```
        }
```

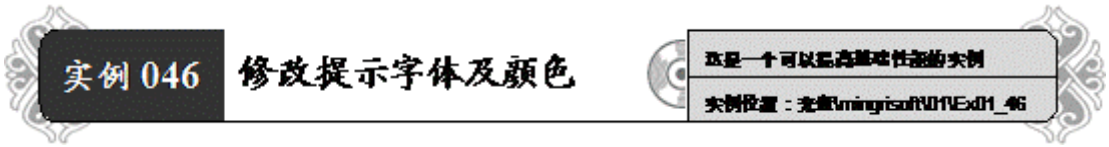
鼠标移动事件处理代码，具体代码如下：

```
        private void button1_MouseMove(object sender, MouseEventArgs e)
        {
            if (e.Button == MouseButtons.Left)
            {
                this.Left += e.X - startX;
                this.Top += e.Y - StartY;
            }
        }
```

举一反三

根据本实例，读者可以开发以下程序。

- 可以用窗体的用户区拖动窗体。
- 不可以拖动的窗体。



实例 046　修改提示字体及颜色

实例说明

如果设置了控件的 ToolTip 属性，当鼠标移到该控件后，会提示相关的文本，但没有提供对提示字体及颜色的设置属性，如何改变提示文本的样式和字体呢？本例可以设置提示文本的字体及颜色。运行本例，效果如图 1.47 所示。

技术要点

C# 2.0 中提供了 ToolTip 控件，可以指定关联控件并为每个控件提供提示文本，其中 ToolTipTitle 属性指定文本提示盒中的文本。下面介绍相关的属性和方法。

（1）SetToolTip 方法

使工具提示文本与指定的控件相关联。其语法结构如下：

```
public void SetToolTip (Control control,string caption)
```

参数说明如下。

l control：要将工具提示文本与其关联的 Control。

l caption：指针位于控件上方时要显示的工具提示文本。

（2）ToolTip.ToolTipTitle 属性

获取或设置工具提示窗口的标题。其语法结构如下：

```
public string ToolTipTitle { get; set; }
```

l 属性值：包含窗口标题的 String。该标题在窗口中作为一行粗体文本显示在标准的工具提示控件说明文本的上方。通常，标题只用于区分窗体上不同类别的控件，或作为较长控件说明的简介。

实现过程

（1）创建一个项目，将其命名为 Ex01_46，默认窗体为 Form1。

（2）在 Form1 窗体上添加 Button 控件用来在其上方显示提示文本；添加 ToolTip 控件用来设计提示文本。

（3）主要程序代码。

设置提示文本，及提示文本的关联控件，具体代码如下：

```
private void Form1_Load(object sender, EventArgs e)
{
    this.toolTip1.OwnerDraw = true;
    this.toolTip1.SetToolTip(this.button1,"设置提示的字体及颜色");
    this.toolTip1.Draw += new DrawToolTipEventHandler(toolTip1_Draw);
}
```

设置文本的提示样式，具体代码如下：

```
void toolTip1_Draw(object sender, DrawToolTipEventArgs e)
{
   // throw new Exception("The method or operation is not implemented.");
    e.DrawBackground( );
    e.DrawBorder( );
```

```
        using (StringFormat sf = new StringFormat( ))
        {
            sf.Alignment = StringAlignment.Center;
            sf.LineAlignment = StringAlignment.Center;
            sf.HotkeyPrefix = System.Drawing.Text.HotkeyPrefix.None;
            sf.FormatFlags = StringFormatFlags.NoWrap;
            using (Font f = new Font("宋体", 12))
            {
                e.Graphics.DrawString(e.ToolTipText, f,
                    SystemBrushes.ActiveCaptionText, e.Bounds, sf);
            }
        }
    }
```

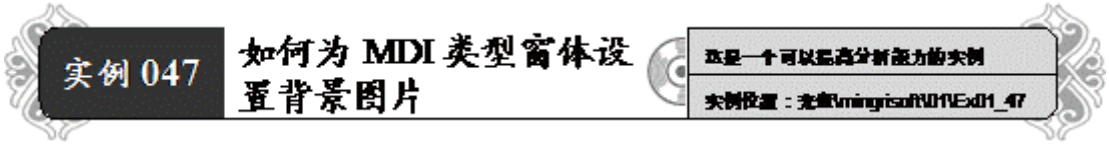
举一反三

根据本实例，读者可以实现以下功能。

- 修改任意控件提示文本的样式。
- 提示时加提示的声音。

1.14 其 他 技 术

本节主要介绍了如何创建和关闭 MDI 窗体。在大型项目和产品的开发中常常将系统设计为 MDI 界面。



实例 047　如何为 MDI 类型窗体设置背景图片

实例说明

图 1.48 给 MDI 窗体加背景

MDI 窗体是一种应用非常广泛的窗体类型，在一个主窗体内包含多个子窗体，子窗体永远不会显示在主窗体的外面。当子窗体不能完全的显示在主窗体中时，主窗体会显示滚动条来调整可视范围，在其他开发环境中为 MDI 窗体添加背景图片十分困难。但在 C# 2.0 中实现非常容易。在本例中实现了一个具有背景的 MDI 窗体。实例效果如图 1.48 所示。

技术要点

在 C# 2.0 中直接提供了 BackgroundImage 属性，该属性可以直接设置窗体的背景图片。设置 IsMdiContainer 属性为 True 可以使窗体成为 MDI 主窗体。下面详细介绍一下相关属性。

（1）BackgroundImage 属性

获取或设置在控件中显示的背景图像。其语法结构如下：

```
public virtual Image BackgroundImage { get; set; }
```

l 属性值：一个 Image，表示在控件的背景中显示的图像。

（2）Form.IsMdiContainer 属性

获取或设置一个值，该值指示窗体是否为多文档界面（MDI）子窗体的容器。其语法结构如下：

```
public bool IsMdiContainer { get; set; }
```

l 属性值：如果该窗体是 MDI 子窗体的容器，则为 True；否则为 False。默认为 False。

此属性将窗体的显示和行为更改为 MDI 父窗体。当此属性设置为 True 时，该窗体显示具有凸起边框的凹陷工作区。所有分配给该父窗体的 MDI 子窗体都在该父窗体的工作区内显示。

实现过程

（1）创建一个项目，将其命名为 Ex01_47，默认窗体为 Form1。

（2）添加一个窗体，默认窗体的 Name 属性为 Form2。

（3）为 Form1 窗体中添加背景图片。

（4）设置 Form1 窗体的 IsMdiContainer 属性为 True，该窗口作为 MDI 主窗体。

图 1.49 向提示框中添加图标

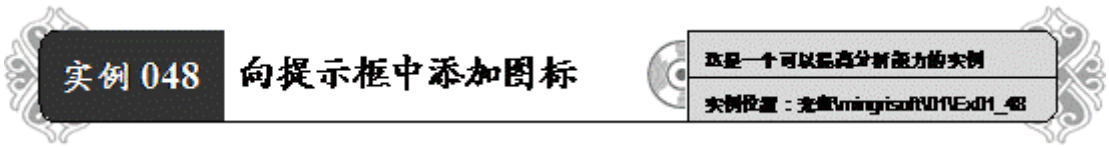
（5）主要程序代码。

```
private void Form1_Load(object sender, EventArgs e)
{
    Form2 f = new Form2( );
    f.MdiParent = this;
    f.Show( );
}
```

举一反三

根据本实例，读者可以开发以下程序。

- 为非 MDI 窗体制作背景。
- 为 MDI 子窗体设定显示区域。



实例 048　向提示框中添加图标

实例说明

在开发程序时，为了让用户熟悉操作，经常使用一些提示框，显示提示信息。默认情况下，提示信息框只包含提示信息，未免有些单调，如果在提示信息框中显示一个图标，程序或许就别具风格了。本实例实现了在提示框中添加图标的功能，实例运行结果如图 1.49 所示。

技术要点

要修改提示信息框的风格，首先需要了解 C#中提示信息框的设计原理。在 C#中，提示信息框是用 ToolTip 控件来实现的。ToolTip 控件的 ToolTipIcon 属性可以设置提示时显示的图片，下面详细介绍一下该属性。

ToolTip.ToolTipIcon 属性用来获取或设置一个值，该值定义要在工具提示文本旁显示的图标的类型。其语法结构如下：

```
public ToolTipIcon ToolTipIcon { get; set; }
```

l 属性值：System.Windows.Forms.ToolTipIcon 枚举值之一。

实现过程

（1）创建一个项目，将其命名为 Ex01_48，默认窗体为 Form1。

（2）在 Form1 窗口中添加 3 个 Label 控件，用来显示文字。

（3）在窗体上添加 ToolTip 控件用来显示提示内容和提示样式。

（4）主要程序代码。

```
. private void Form1_Load(object sender, EventArgs e)
  {
     toolTip1.SetToolTip(label1,"人生格言");
     toolTip1.SetToolTip(label2, "人生格言");
  }
```

举一反三

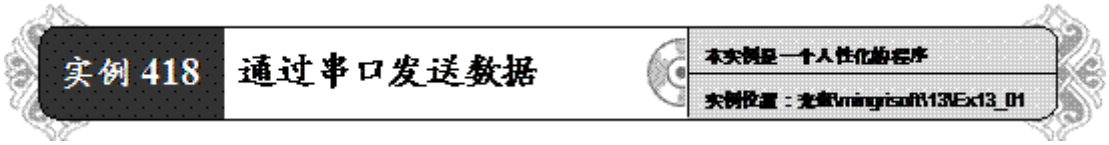
根据本实例，读者可以实现以下功能。

- 自定义提示信息框。
- 制作气泡样式提示信息框。

13.1 串 口 控 制

图 13.1 通过串口传递数据

串行口是计算机的标准接口，现在的 PC 机（个人电脑）一般至少有两个串行口 COM1 和 COM2。串行口应用广泛，在数据通信、计算机网络以及分布式工业控制系统中，经常采用串行通信来交换数据和信息。本节通过几个实例，介绍串口应用的技术和方法。



实例 418　通过串口发送数据

实例说明

现在大多数硬件设备均采用串口技术与计算机相连，因此串口的应用程序开发越来越普遍。例如，在计算机没有安装网卡的情况下，将本机上的一些信息数据传输到另一台计算机上，那么利用串口通信就可以实现。运行本程序，在“发送数据”文本框中输入要传送的数据，单击【发送】按钮，将传送的数据发送到所选择的端口号中；单击【接收】按钮，传递的数据被接收到“接收数据”文本框中。如图 13.1 所示。

技术要点

在.NET Framework 2.0 中提供了 SerialPort 类，该类主要实现串口数据通信等。下面主要介绍该类的主要属性（表 13.1）和方法（表 13.2）。

表 13.1　　SerialPort 类的常用属性

名　　称	说　　明
BaseStream	获取 SerialPort 对象的基础 Stream 对象
BaudRate	获取或设置串行波特率
BreakState	获取或设置中断信号状态
BytesToRead	获取接收缓冲区中数据的字节数
BytesToWrite	获取发送缓冲区中数据的字节数
CDHolding	获取端口的载波检测行的状态
CtsHolding	获取“可以发送”行的状态

DataBits	获取或设置每个字节的标准数据位长度
DiscardNull	获取或设置一个值，该值指示 Null 字节在端口和接收缓冲区之间传输时是否被忽略
DsrHolding	获取数据设置就绪 (DSR) 信号的状态
DtrEnable	获取或设置一个值，该值在串行通信过程中启用数据终端就绪 (DTR) 信号
Encoding	获取或设置传输前后文本转换的字节编码
Handshake	获取或设置串行端口数据传输的握手协议
IsOpen	获取一个值，该值指示 SerialPort 对象的打开或关闭状态
NewLine	获取或设置用于解释 ReadLine( )和 WriteLine( )方法调用结束的值
Parity	获取或设置奇偶校验检查协议

续表

名　　称	说　　明
ParityReplace	获取或设置一个字节，该字节在发生奇偶校验错误时替换数据流中的无效字节
PortName	获取或设置通信端口，包括但不限于所有可用的 COM 端口
ReadBufferSize	获取或设置 SerialPort 输入缓冲区的大小
ReadTimeout	获取或设置读取操作未完成时发生超时之前的毫秒数
ReceivedBytesThreshold	获取或设置 DataReceived 事件发生前内部输入缓冲区中的字节数
RtsEnable	获取或设置一个值，该值指示在串行通信中是否启用请求发送 (RTS) 信号
StopBits	获取或设置每个字节的标准停止位数
WriteBufferSize	获取或设置串行端口输出缓冲区的大小
WriteTimeout	获取或设置写入操作未完成时发生超时之前的毫秒数

表 13.2　　SerialPort 类的常用方法

方 法 名 称	说　　明
Close	关闭端口连接，将 IsOpen 属性设置为 False，并释放内部 Stream 对象
Open	打开一个新的串行端口连接
Read	从 SerialPort 输入缓冲区中读取
ReadByte	从 SerialPort 输入缓冲区中同步读取一个字节
ReadChar	从 SerialPort 输入缓冲区中同步读取一个字符
ReadLine	一直读取到输入缓冲区中的 NewLine 值
ReadTo	一直读取到输入缓冲区中指定 value 的字符串
Write	已重载。将数据写入串行端口输出缓冲区
WriteLine	将指定的字符串和 NewLine 值写入输出缓冲区

注意：用跳线使串口的第 2、3 针连接，可以在本地计算机上实现串口通信，所以，通过串口的第 2、3 针的连接可以对程序进行检测。串口截面图如图 13.2 所示。

图 13.2　串口截面图

实现过程

（1）新建一个项目，命名为 Ex13_01，默认窗体为 Form1。

（2）在 Form1 窗体中，主要添加两个 Button 控件，分别用于执行发送数据和接受数据，添加两个 TextBox 控件，用于输入发送数据和显示接收数据。

（3）主要程序代码。

```
private void button1_Click(object sender, EventArgs e)
{
    serialPort1.PortName = "COM1";
    serialPort1.BaudRate = 9600;
    serialPort1.Open();
    byte[] data = Encoding.Unicode.GetBytes(textBox1.Text);
    string str = Convert.ToBase64String(data);
    serialPort1.WriteLine(str);
    MessageBox.Show("数据发送成功！","系统提示");
}
private void button2_Click(object sender, EventArgs e)
{
    byte[] data = Convert.FromBase64String(serialPort1.ReadLine());
    textBox2.Text = Encoding.Unicode.GetString(data);
    serialPort1.Close();
    MessageBox.Show("数据接收成功！","系统提示");
```

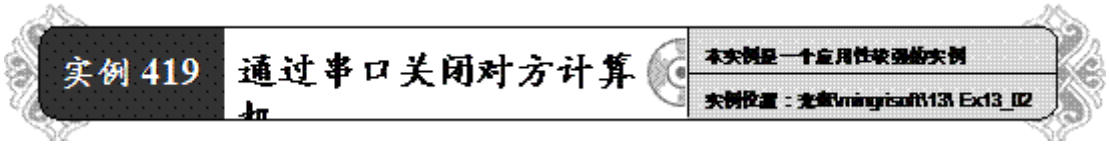
```
}
```

图 13.3 控制对方计算机关闭

举一反三

根据本实例，读者可以实现以下功能。

- 远程监控对方计算机屏幕。
- 下位机控制程序。



实例 419　通过串口关闭对方计算机

实例说明

在网络应用程序中，主要通过网卡实现数据的传输，因此可以利用套接字技术实现远程关闭计算机。如果计算机中没有安装网卡，该如何实现远程关闭计算机呢？本例实现了利用串口关闭对方计算机，程序运行结果如图 13.3 所示。

技术要点

本实例使用 SerialPort 类的属性和方法，请参见实例“通过串口发送数据”。下面主要介绍 SerialPort 类的 DataReceived 事件，DataReceived 事件为本实例的主要使用技术。DataReceived 事件表示将处理 SerialPort 对象的数据接收事件的方法。串行接收事件可以由 SerialData 枚举中的任何项引起，是否引发此事件由操作系统决定，所以不一定会报告所有奇偶校验错误。

注意：本实例从开发到测试，都是由本地计算机完成的，用户只需要使用跳线将串口的第 2、3 针连接，可以在本地计算机上实现串口通信。跳线连接请参见图 13.2。

实现过程

（1）新建一个项目，命名为 Ex13_02，默认窗体为 Form1。

（2）在 Form1 窗体中，主要添加两个 Button 控件，分别用于打开通信串口和关闭对方计算机。

（3）主要程序代码。

```
    private void button1_Click(object sender, EventArgs e)
    {
        //打开串口
        serialPort1.PortName = "COM1";
        serialPort1.Open();
        button1.Enabled = false;
        button2.Enabled = true;
    }  //数据接收事件，等待接收关机命令
    private void serialPort1_DataReceived(object sender, SerialDataReceivedEventArg
s e)
    {
        byte[] data = Convert.FromBase64String(serialPort1.ReadLine());
        string str = Encoding.Unicode.GetString(data);
        serialPort1.Close();
        if (str == "关机")
        {
            Process p = new Process();
            p.StartInfo.FileName = "cmd.exe";
            p.StartInfo.UseShellExecute = false;
            p.StartInfo.RedirectStandardInput = true;
            p.StartInfo.RedirectStandardOutput = true;
            p.StartInfo.RedirectStandardError = true;
            p.StartInfo.CreateNoWindow = true;
            p.Start();
            p.StandardInput.WriteLine("shutdown /s");
            p.StandardInput.WriteLine("exit");
        }
    }  //发送关机命令
```

```
private void button2_Click(object sender, EventArgs e)
{
    if (button2.Text == "关闭计算机")
    {
        //发送关机命令数据
        byte[] data = Encoding.Unicode.GetBytes("关机");
        string str = Convert.ToBase64String(data);
        serialPort1.WriteLine(str);
        button2.Text = "取消关机";
    }
    else
    {
        button2.Text = "关闭计算机";
        button1.Enabled = true;
        button2.Enabled = false;
        //取消关机
        Process p = new Process();
        p.StartInfo.FileName = "cmd.exe";
        p.StartInfo.UseShellExecute = false;
        p.StartInfo.RedirectStandardInput = true;
        p.StartInfo.RedirectStandardOutput = true;
        p.StartInfo.RedirectStandardError = true;
        p.StartInfo.CreateNoWindow = true;
        p.Start();
        p.StandardInput.WriteLine("shutdown /a");
        p.StandardInput.WriteLine("exit");
    }
}
```

图 13.4 将密码写入加密狗

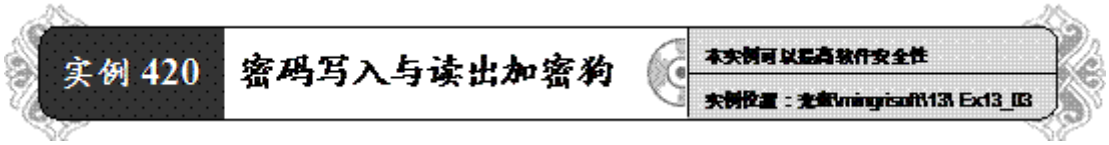
举一反三

根据本实例，读者可以实现以下功能。

- 远程控制对方计算机操作。
- 定时控制对方计算机关闭。

13.2 加 密 狗

一些商务管理软件，为了防止盗版，经常使用加密狗将软件加密。下面的两个实例将介绍如何将密码写入加密狗及利用加密狗来设计加密程序。



实例 420 密码写入与读出加密狗

实例说明

在使用加密狗时，需要向加密狗中写入或读取数据。例如，将密码写入或读取加密狗，如何实现呢？运行本例，在文本框中设置密码后，单击【写入】按钮，即可将设置的密码写入加密狗，成功写入后，单击【读出】按钮，即可将写入的密码读出并显示在文本框中。如图 13.4 所示。

技术要点

在购买加密狗时，厂家通常会附带有开发手册和一张光盘。开发手册中介绍了加密狗的使用方法和开发资料。本例使用赛孚耐信息技术有限公司的加密狗产品，该产品提供了.NET 中非托管的类库，来完成加密狗的数据读写功能。下面介绍有关加密狗的类库中的读写函数。

- DogWrite 函数

该函数将 pdogData 指向的数据写入加密狗中，从 DogAddr 地址开始写入，到 DogBytes 地址停止。

函数声明如下：

```
[DllImport("Win32dll.dll", CharSet = CharSet.Ansi)]
```

public static unsafe extern uint DogWrite(uint idogBytes, uint idogAddr, byte* pdogData);

参数说明如下。

l idogAddr：对软件狗读写操作时用户区中的首地址。取值范围为 0～99。

l IdogBytes：对软件狗读写操作时的字节长度。读写时取值范围为 1～100，并且与 idogAddr 之和不能超过 100。

l pdogData：指针型变量。指向读写操作或变换的数据缓冲区。

l 返回值：0 表示操作成功，其他值是错误码。

- DogRead 函数

该函数从加密狗中的 idogAddr 开始的存储区读出数据，存入 pdogData 指定的缓冲区，读出字节数为 idogBytes。切记，缓冲区大小要足够长。

函数声明如下：

```
[DllImport("Win32dll.dll", CharSet = CharSet.Ansi)]
public static unsafe extern uint DogRead(uint idogBytes, uint idogAddr, byte* pdogData);
```

参数说明如下。

l idogAddr：对软件狗读写操作时用户区中的首地址。取值范围为 0～99。

l idogBytes：对软件狗读写操作时的字节长度。读写时取值范围为 1～100，并且与 idogAddr 之和不能超过 100。

l pdogData：指针型变量。指向读写操作或变换的数据缓冲区。

l 返回值：0 表示操作成功，其他值是错误码。

 注意以下几点。

在使用这个函数之前，必须将随加密狗附带的安装程序安装完整，并将安装目录下的 Win32dll.dll 文件复制到系统目录下。例如：

在 Windows 2003 下将安装目录下的“\SafeNet China\SoftDog SDK V3.1\Win32\Win32dll\HighDll\ Win32dll.dll”文件复制到“C:\WINDOWS\system32\”文件夹中。

实现过程

（1）新建一个项目，命名为 Ex13_03，默认窗体为 Form1。

（2）在 Form1 窗体中，主要添加两个 Button 控件，用于执行向加密狗数据的写入与读出数据，添加两个 TextBox 控件，分别用于填写向加密狗中写入的数据和显示读取加密狗中的数据。

（3）主要程序代码。

设置加密狗类，并且完善加密狗的读写功能，代码如下：

```
[StructLayout(LayoutKind.Sequential)]
//这个类用于读写加密狗
public unsafe class Dog
{
   public uint DogBytes, DogAddr;  //设置加密狗字节长度和起始地址
   public byte[] DogData;  //设置数据的长度
   public uint Retcode;
   [DllImport("Win32dll.dll", CharSet = CharSet.Ansi)]
   public static unsafe extern uint DogRead(uint idogBytes, uint idogAddr, byte* pdo
gData);
   [DllImport("Win32dll.dll", CharSet = CharSet.Ansi)]
   public static unsafe extern uint DogWrite(uint idogBytes, uint idogAddr, byte* pdo
gData);
   public unsafe Dog(ushort num)
   {
      DogBytes = num;
      DogData = new byte[DogBytes]; //设置数据的长度
   }
   public unsafe void ReadDog()
   {
      fixed (byte* pDogData = &DogData[0])
      {
         Retcode = DogRead(DogBytes, DogAddr, pDogData);     //将数据读出加密狗
      }
   }
   public unsafe void WriteDog()
   {
      fixed (byte* pDogData = &DogData[0])
      {
         Retcode = DogWrite(DogBytes, DogAddr, pDogData);  //将数据写入加密狗
      }
   }
}
```

调用加密狗类，进行加密狗的读写功能，代码如下：

```
    private void button1_Click_1(object sender, EventArgs e)
    {
        Dog dog = new Dog(100);
        dog.DogAddr = 0;
        dog.DogBytes = 10;
        string str = textBox1.Text;
        for (int i = 0; i < str.Length; i++)
        {
            dog.DogData[i] = (byte)str[i];
        }
        dog.WriteDog();
        MessageBox.Show("密码已成功写入加密狗！", "成功提示！", MessageBoxButto
ns.OK, MessageBoxIcon.Information);
        textBox1.ReadOnly = true;
        button1.Enabled = false;
        button2.Enabled = true;
    }
    private void button2_Click_1(object sender, EventArgs e)
    {
        Dog dog = new Dog(100);
        dog.DogAddr = 0;
        dog.DogBytes = 10;
        dog.ReadDog();
        if (dog.Retcode == 0)  //开始读加密狗数据
        {
            char[] chTemp = new char[textBox1.Text.Length];
            for (int i = 0; i < textBox1.Text.Length; i++)
            {
                chTemp[i] = (char)dog.DogData[i];
            }
            String str = new String(chTemp);
            textBox2.Text = str;
        }
        else
        {
            textBox2.Text = "2:" + dog.Retcode;
        }
        textBox1.ReadOnly = false;
        button2.Enabled = false;
        button1.Enabled = true;
    }
```

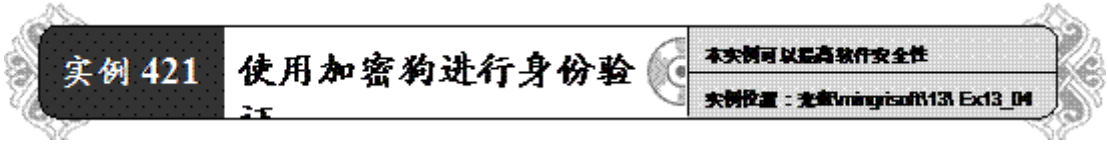
图 13.5 使用加密狗进行身份验证

注意：本程序所使用的代码为不安全代码，正常编译是无法通过的，那么需要设置开发环境允许运行不安全代码，设置步骤为：在菜单栏中选择"项目"/"属性"/"生成"子菜单，在"生成"选项卡中选中"允许不安全代码"选项即可。

举一反三

根据本实例，读者可以开发以下程序。

- 利用加密狗加密自己的软件。
- 利用加密狗设计控制计算机使用的程序。



实例 421　使用加密狗进行身份验证

实例说明

在程序开发过程中，对于一些机密的数据，开发人员需要将其有效的保护起来。例如，对于用户的密码，如果从数据库中验证用户密码，很容易被非法人员发现甚至破解。本例实现了利用加密狗进行身份验证。实例运行结果如图 13.5 所示。

技术要点

本例的关键是从加密狗中读取数据，可以使用 ReadDog 函数实现。有关该函数的介绍请参考实例“密码写入与读出加密狗”中的“技术要点”部分。

实现过程

（1）新建一个项目，命名为 Ex13_03，默认窗体为 Form1。

（2）在 Form1 窗体中，主要添加两个 Button 控件，用于数据验证和退出程序，添加两个 TextBox 控件，分别用于输入用户名称和密码。

（3）主要程序代码。

```
private void button1_Click(object sender, EventArgs e)
{
```

```
        Dog dog = new Dog(100);
        dog.DogAddr = 0;
        dog.DogBytes = 6;
        dog.ReadDog();
        if (dog.Retcode == 0)
        {
            char[] chTemp = new char[6];
            for (int i = 0; i < 6; i++)
            {
                chTemp[i] = (char)dog.DogData[i];
            }
            String str = new String(chTemp);
            if (textBox2.Text==str)
            {
                MessageBox.Show("OK");
            }
            else
            {
                MessageBox.Show("error");
            }
        }
    }
```

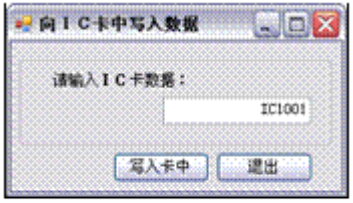


图 13.6 向 IC 卡中写入数据

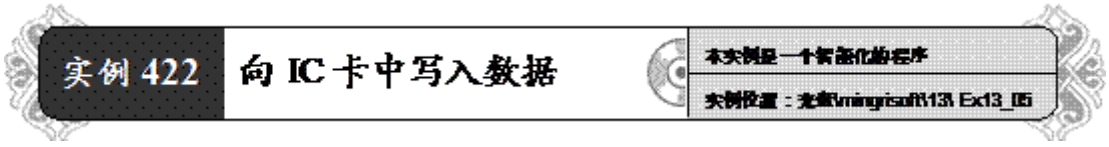
举一反三

根据本实例，读者可以实现以下功能。

- 利用加密狗设计加密软件。
- 使用加密狗控制用户使用权限。

13.3 IC 卡应用

IC（Integrated Circuit）卡，也被称作智能卡（Smart Card），具有写入数据和存储数据的功能，IC 卡内存储器的内容可以根据需要有条件地供外部读取，完成信息处理和判定。由于其内部具有集成电路，不但可以存储大量信息，具有极强的保密性能，并且还具有抗干扰、无磨损、寿命长等特性。因此在各个领域中得到广泛应用。下面通过两个实例介绍 IC 卡的简单应用。



实例 422　向 IC 卡中写入数据

实例说明

IC 卡是携带应用信息和数据的媒体，空白 IC 卡是不能立即使用的，必须对 IC 卡应用系统进行初始化，写入系统 IC 卡和个人密码，个人专用信息和应用数据。下面介绍如何向 IC 卡中写入数据。运行本例，在“数据”文本框中输入要存入 IC 卡中的数据，单击“写数据”按钮，即可将输入的数据写入 IC 卡中。如图 13.6 所示。

技术要点

本例使用的是深圳明华生产的明华 IC 卡读写器，用户在使用时将驱动程序安装完毕后，即可正常使用本系统。

本例通过调用 Mwic_32.dll 链接库，进行 IC 卡的读写工作。下面介绍与 IC 卡写操作相关的几个函数。

（1）auto_init 函数

该函数用于初始化 IC 卡读卡器。语法如下：

```
public static extern int auto_init(int port, int baud);
```

参数说明如下。

l port：标识端口号，Com1 对应的端口号为 0；Com2 对应的端口号为 1，依此类推。

l baud：标识波特率。

l 返回值：如果初始化成功，返回值是 IC 卡设备句柄；如果初始化失败，返回值小于零。

（2）setsc_md 函数

该函数用于设置设备密码模式。语法如下：

```
public static extern int setsc_md(int icdev, int mode);
```

参数说明如下。

l icdev：标识设备句柄，通常是 auto_init 函数的返回值。

l mode：标识设备密码模式，如果为 0，设备密码有效，设备在加电时必须验证设备密码才能对设备进行操作。如果为 1，设备密码无效。

l 返回值：如果函数执行成功返回值为零，否则小于零。

（3）get_status 函数

该函数用于获取设备的当前状态。语法如下：

```
public static extern Int16 get_status(int icdev, Int16* state);
```

参数说明如下。

l icdev：标识设备句柄，通常是 auto_init 函数的返回值。

l state：用于接收函数返回的结果。如果为 0 表示读卡器中无卡，为 1 表示读卡器中有卡。

l 返回值：如果函数执行成功返回值为零，否则小于零。

（4）csc_4442 函数

该函数用于核对 IC 卡密码。语法如下：

```
public static extern Int16 Csc_4442(int icdev, int len, [MarshalAs(UnmanagedType.LP
Array)] byte[] p_string);
```

参数说明如下。

l icdev：标识设备句柄，通常是 auto_init 函数的返回值。

l len：标识密码长度，其值为 3。

l p_string：标识设置的密码。

l 返回值：如果函数执行成功返回值为零，否则小于零。

（5）swr_4442 函数

该函数用于向 IC 卡中写入数据。语法如下：

```
public static extern int swr_4442(int icdev, int offset, int len, char* w_string);
```

参数说明如下。

l icdev：标识设备句柄，通常是 auto_init 函数的返回值。

l offset：标识地址的偏移量，范围是 0～255。

l len：标识字符串长度。

l w_string：标识写入的数据。

（6）ic_exit 函数

该函数用于关闭设备端口。语法如下：

```
public static extern int ic_exit(int icdev);
```

参数说明如下。

l icdev：标识设备句柄，通常是 auto_init 函数的返回值。

（7）dv_beep 函数

该函数使读卡器嗡鸣。语法如下：

```
public static extern int dv_beep(int icdev, int time);
```

参数说明如下。

l icdev：标识设备句柄，通常是 auto_init 函数的返回值。

l time：标识嗡鸣持续的时间，单位是 10 毫秒。

实现过程

（1）新建一个项目，命名为 Ex13_05，默认窗体为 Form1。

（2）在 Form1 窗体中，主要添加两个 Button 控件，用于执行向卡中写入数据和退出程序的操作，添加一个 TextBox 控件，将 TextBox 中数据写入 IC 卡中。

（3）主要程序代码。

将程序所使用的操作 IC 卡的函数，封装在类 IC 中。代码如下：

```
[StructLayout(LayoutKind.Sequential)]
public unsafe class IC
{
    //对设备进行初始化
    [DllImport("Mwic_32.dll", EntryPoint = "auto_init", SetLastError = true, CharSet =
 CharSet.Ansi, ExactSpelling = true, CallingConvention = CallingConvention.StdCall)]
    public static extern int auto_init(int port, int baud);
    //设备密码格式
    [DllImport("Mwic_32.dll", EntryPoint = "setsc_md", SetLastError = true, CharSet =
 CharSet.Ansi, ExactSpelling = true, CallingConvention = CallingConvention.StdCall)]
    public static extern int setsc_md(int icdev, int mode);
    //获取设备当前状态
    [DllImport("Mwic_32.dll", EntryPoint = "get_status", SetLastError = true, CharSet
= CharSet.Ansi, ExactSpelling = true, CallingConvention = CallingConvention.StdCall)]
    public static extern Int16 get_status(int icdev, Int16* state);
    //关闭设备通讯接口
    [DllImport("Mwic_32.dll", EntryPoint = "ic_exit", SetLastError = true, CharSet = C
harSet.Ansi, ExactSpelling = true, CallingConvention = CallingConvention.StdCall)]
    public static extern int ic_exit(int icdev);
    //使设备发出蜂鸣声
    [DllImport("Mwic_32.dll", EntryPoint = "dv_beep", SetLastError = true, CharSet =
 CharSet.Ansi, ExactSpelling = true, CallingConvention = CallingConvention.StdCall)]
    public static extern int dv_beep(int icdev, int time);
    //向 IC 卡中写数据
    [DllImport("Mwic_32.dll", EntryPoint = "swr_4442", SetLastError = true, CharSet
= CharSet.Ansi, ExactSpelling = true, CallingConvention = CallingConvention.StdCall)]
    public static extern int swr_4442(int icdev, int offset, int len, char* w_string);
```

```
   //核对卡密码
   [DllImport("Mwic_32.dll", EntryPoint = "csc_4442", SetLastError = true, CharSet =
 CharSet.Auto, ExactSpelling = true, CallingConvention = CallingConvention.Winapi)]
   public static extern Int16 Csc_4442(int icdev, int len, [MarshalAs(UnmanagedType.
LPArray)] byte[] p_string);
}
```

下面代码主要用于将 TextBox 中数据写入到 IC 卡中。代码如下：

```
      private void button1_Click(object sender, EventArgs e)
      {
         //初始化
         int icdev = IC.auto_init(0, 9600);
         if (icdev < 0)
            MessageBox.Show("端口初始化失败,请检查接口线是否连接正确。","错误提示",
MessageBoxButtons.OK,MessageBoxIcon.Information);
         int md = IC.setsc_md(icdev, 1); //设备密码格式
         unsafe
         {
            Int16 status = 0;
            Int16 result = 0;
            result = IC.get_status(icdev, &status);
            if (result != 0)
            {
               MessageBox.Show("设备当前状态错误！");
               int d1 = IC.ic_exit(icdev);  //关闭设备
               return;
            }
            if (status != 1)
            {
```

```
            MessageBox.Show("请插入ＩＣ卡");
            int d2 = IC.ic_exit(icdev);  //关闭设备
            return;
        }
    }
    unsafe
    {
        //卡的密码默认为 6 个 f（密码为：ffffff），1 个 f 的 16 进制是 15,两个 f 的 16
进制是 255
        byte[] pwd = new byte[3] { 255, 255, 255 };
        //byte[] pwd = new byte[3] { 0xff, 0xff, 0xff };
        //char[] pass=new ch{0xff,0xff,0xff};
        Int16 checkIC_pwd = IC.Csc_4442(icdev, 3, pwd);
        if (checkIC_pwd < 0)
        {
            MessageBox.Show("ＩＣ卡密码错误！");
            return;
        }
        char str = 'a';
        int write=-1;
        for (int j = 0; j < textBox1.Text.Length; j++)
        {
            str = Convert.ToChar(textBox1.Text.Substring(j, 1));
            write = IC.swr_4442(icdev, 33 + j, textBox1.Text.Length, &str);
        }
        if (write == 0)
        {
            int beep = IC.dv_beep(icdev, 20);  //发出蜂鸣声
```

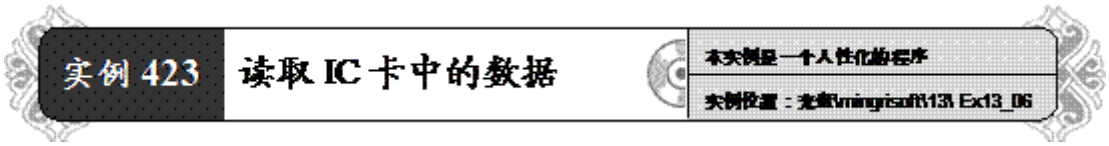
图 13.7 读取 IC 卡中的数据

```
                MessageBox.Show("数据已成功写入ＩＣ卡中！");
            }
            else
                MessageBox.Show("数据写入ＩＣ卡失败！");
        }
        int d = IC.ic_exit(icdev);  //关闭设备
    }
```

举一反三

根据本实例，读者可以实现以下功能。

- 在图书借阅中使用 IC 卡。
- 利用 IC 卡控制上网。



实例 423　读取 IC 卡中的数据

实例说明

向 IC 卡写入数据后，就可以进行读卡操作了。运行本例，将写入数据的 IC 卡插入读卡器，单击【读卡】按钮，IC 卡中的数据将显示在文本框中。如图 13.7 所示。

技术要点

本例中主要调用 srd_4442 函数读取 IC 卡中的数据，相关函数介绍请参考实例“向 IC 卡中写入数据”中的“技术要点”部分。这里只介绍读卡函数。

q　srd_4442 函数

该函数用于读取 IC 卡中的数据。语法如下：

```
public static extern int srd_4442(int icdev, int offset, int len, char* r_string);
```

参数说明如下。

- icdev：标识设备句柄，通常是 auto_init 函数的返回值。
- offset：标识地址的偏移量，范围是 0～255。
- len：标识字符串长度。
- r_string：用于存储返回的数据。

实现过程

（1）新建一个项目，命名为 Ex13_06，默认窗体为 Form1。

（2）在 Form1 窗体中，主要添加两个 Button 控件，用于读取卡中的数据和退出程序，添加一个 TextBox 控件，显示卡中的数据。

（3）主要程序代码。

```
private void button1_Click(object sender, EventArgs e)
{
    //初始化
    int icdev = IC.auto_init(0, 9600);
    if (icdev < 0)
        MessageBox.Show("端口初始化失败,请检查接口线是否连接正确。", "错误提示
", MessageBoxButtons.OK, MessageBoxIcon.Information);
    int md = IC.setsc_md(icdev, 1); //设备密码格式
    int i = IC.dv_beep(icdev, 10);  //发出蜂鸣声
    unsafe
    {
        Int16 status = 0;
        Int16 result = 0;
        result = IC.get_status(icdev, &status);
        if (result != 0)
        {
            MessageBox.Show("设备当前状态错误！");
```

```
            int d1 = IC.ic_exit(icdev);   //关闭设备
            return;
        }
        if (status != 1)
        {
            MessageBox.Show("请插入ＩＣ卡");
            int d2 = IC.ic_exit(icdev);   //关闭设备
            return;
        }
    }
    unsafe
    {
        char str = 'a';
        int read = -1;
        for (int j = 0; j < 6; j++)
        {
            read = IC.srd_4442(icdev, 33 + j, 1, &str);
            textBox1.Text = textBox1.Text + Convert.ToString(str);
        }
        if (read == 0)
            MessageBox.Show("ＩＣ卡中数据读取成功！");
    }
    int d = IC.ic_exit(icdev);  //关闭设备
}
```

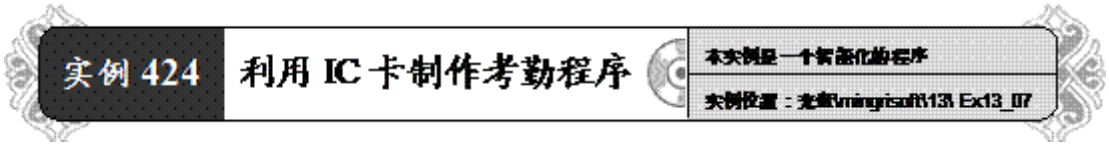
举一反三

根据本实例，读者可以开发以下程序。

- 读取 IC 卡电话系统。

图 13.8 利用 IC 卡制作考勤程序

- 公交车刷卡系统。



实例 424　利用 IC 卡制作考勤程序

实例说明

IC 卡广泛应用于各行业，包括银行卡、公交车刷卡系统、读书卡等。下面介绍使用 IC 卡制作简单的公司考勤系统。运行本例，单击【刷卡】按钮，即可对员工进行考勤。实现效果如图 13.8 所示。

技术要点

有关 IC 卡的操作函数请参考实例“向 IC 卡中写入数据”和“读取 IC 卡中的数据”中的“技术要点”部分。

下面主要介绍通过 IC 卡如何实现员工考勤。主要将写入 IC 卡中的卡号读取出来，然后从数据表中查询员工信息。具体代码请参考实现过程。

实现过程

（1）新建一个项目，命名为 Ex13_07，默认窗体为 Form1。

（2）在 Form1 窗体中，主要添加 5 个 TextBox 控件和 6 个 Label 控件，用途如图 13.7 所示，添加一个 Button 控件，执行刷 IC 卡命令。

（3）主要程序代码。

```
private void button1_Click(object sender, EventArgs e)
{
    //初始化
    int icdev = IC.auto_init(0, 9600);
    if (icdev < 0)
        label6.Text = "端口初始化失败,请检查接口线是否连接正确。";
    unsafe
```

```
        {
            Int16 status = -1;
            Int16 result = IC.get_status(icdev, &status);
            int md = IC.setsc_md(icdev, 1);  //设备密码格式
            if (result < 0)
            {
                int d1 = IC.ic_exit(icdev);  //关闭设备
                return;
            }
            else if ((result == 0) && (status == 0))
            {
                int d2 = IC.ic_exit(icdev);  //关闭设备
                label6.Text = "请插入 IC 卡";
                return;
            }
        }
        unsafe
        {
            char str = 'a';
            int read = -1;
            string ic = "";
            for (int j = 0; j < 6; j++)
            {
                read = IC.srd_4442(icdev, 33 + j, 1, &str);
                ic = ic + Convert.ToString(str);
            }
            textBox1.Text = ic;
            if (read == 0)
```

```
            label6.Text = "刷卡成功！";
            int beep = IC.dv_beep(icdev, 20);  //发出蜂鸣声
            int d3 = IC.ic_exit(icdev);   //关闭设备
        }
        int d = IC.ic_exit(icdev);  //关闭设备
        //根据卡号，查找相应数据
        OleDbConnection con = new OleDbConnection("Provider=Microsoft.Jet.OLEDB.
4.0;Data Source=" + "price.mdb" + ";Persist Security Info=False");
        OleDbDataAdapter dap = new OleDbDataAdapter("select * from worker whe
re ICID='"+textBox1.Text+"'", con);
        DataSet ds = new DataSet();
        dap.Fill(ds, "table");
        if (ds.Tables.Count > 0)
        {
            textBox2.Text = ds.Tables[0].Rows[0][0].ToString();
            textBox3.Text = ds.Tables[0].Rows[0][1].ToString();
            textBox4.Text = ds.Tables[0].Rows[0][2].ToString();
            textBox5.Text = ds.Tables[0].Rows[0][3].ToString();
        }
        else
        {
            label6.Text = "不存在该用户！";
        }
    }
```

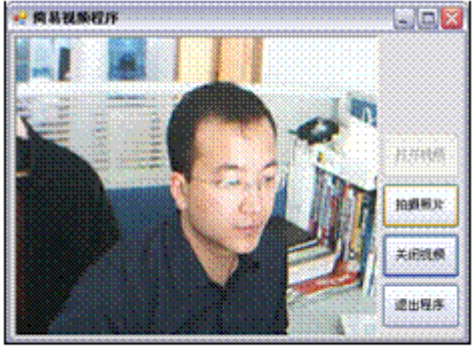


图 13.9 简易摄像头监控程序

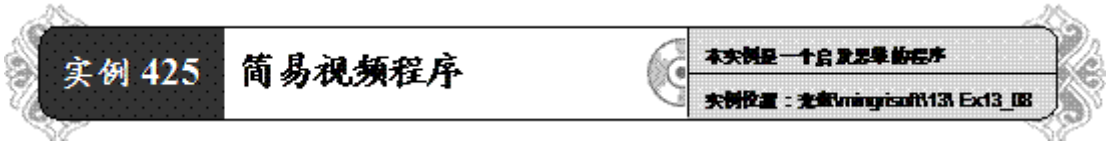
举一反三

根据本实例，读者可以开发以下程序。

- 代金卡系统。
- 工资发放系统。

13.4 监　　控

在一些银行、大型商场、办公楼、升降电梯中，为了保障公有财产、商品、办公设备、资料、人身等的安全，都设有监控系统。在出现问题时，用户可以通过监控系统查找原因。下面的几个实例分别实现了摄像头监控与定时监控的功能。



实例 425　简易视频程序

实例说明

利用普通的简易摄像头，通过 C#语言即可开发成简易视频程序。本实例利用市场上购买的普通摄像头，利用 VFW 技术，实现单路视频监控系统。运行程序，窗体中将显示舰体摄像头采集的视频信息。如图 13.9 所示。

技术要点

本实例主要使用了 VFW（Video for Windows）技术。VFW 是 Microsoft 公司为开发 Windows 平台下的视频应用程序提供的软件工具包，提供了一系列应用程序编程接口（API），用户可以通过这些接口很方便地实现视频捕获、视频编辑及视频播放等通用功能，还可利用回调函数开发比较复杂的视频应用程序。该技术的特点是播放视频时不需要专用的硬件设备，而且应用灵活，可以满足视频应用程序开发的需要。Windows 操作系统自身就携带了 VFW 技术，系统安装时，会自动安装 VFW 的相关组件。

VFW 技术主要由六个功能模块组成，下面进行简单说明。

l AVICAP32.DLL：包含执行视频捕获的函数，给 AVI 文件的 I/O 处理和视频，音频设备驱动程序提供一个高级接口。

l MSVIDEO.DLL：包含一套特殊的 DrawDib 函数，用来处理程序上的视频操作。

l MCIAVI.DRV：包括对 VFW 的 MCI 命令解释器的驱动程序。

l AVIFILE.DLL：包含由标准多媒体 I/O（mmio）函数提供的更高级的命令，用来访问.AVI 文件。

l ICM：压缩管理器，用于管理的视频压缩/解压缩的编译码器。

l ACM：音频压缩管理器，提供与 ICM 相似的服务，适用于波形音频。

其中 13.4 节所有的实例主要使用 AVICAP32.DLL 中的函数和 USER32.DLL 中的函数，函数语法及结构如下。

（1）capCreateCaptureWindow 函数

该函数用于创建一个视频捕捉窗口。语法如下：

```
[DllImport("avicap32.dll")]
public static extern IntPtr capCreateCaptureWindowA(byte[] lpszWindowName, int dwStyle, int x, int y, int nWidth, int nHeight, IntPtr hWndParent, int nID);
```

参数说明如下。

l lpszWindowName：标识窗口的名称。

l dwStyle：标识窗口风格。

l x、y：标识窗口的左上角坐标。

l nWidth、nHeight：标识窗口的宽度和高度。

l hWnd：标识父窗口句柄。

l nID：标识窗口 ID。

l 返回值：视频捕捉窗口句柄。

（2）SendMessage 函数

用于向 Windows 系统发送消息机制。

```
[DllImport("User32.dll")]
private static extern bool SendMessage(IntPtr hWnd, int wMsg, int wParam, int lParam);
```

参数说明如下。

l hWnd：窗口句柄。

l wMsg：将要发送的消息。

l wParam、lParam：消息的参数，每个消息都有两个参数，参数设置由发送的消息而定。

实现过程

（1）新建一个项目，命名为 Ex13_08，默认窗体为 Form1，添加 1 个类文件（.CS），用于编写视频类。

（2）在 Form1 窗体中，主要添加 1 个 PictrueBox 控件，用于显示视频；添加 4 个 Button 控件，用于打开视频、关闭视频、拍摄照片和退出程序。

（3）主要程序代码。

视频类中主要实现打开视频、关闭视频以及通过视频拍摄照片的功能。代码如下：

```
public class VideoAPI  //视频 API 类
{
    // 视频ＡＰＩ调用
    [DllImport("avicap32.dll")]
    public static extern IntPtr capCreateCaptureWindowA(byte[] lpszWindowName, int dwStyle, int x, int y, int nWidth, int nHeight, IntPtr hWndParent, int nID);
    [DllImport("avicap32.dll")]
    public static extern bool capGetDriverDescriptionA(short wDriver, byte[] lpszName, int cbName, byte[] lpszVer, int cbVer);
    [DllImport("User32.dll")]
    public static extern bool SendMessage(IntPtr hWnd, int wMsg, bool wParam, int lParam);
    [DllImport("User32.dll")]
    public static extern bool SendMessage(IntPtr hWnd, int wMsg, short wParam, int lParam);
```

```
    //  常量
    public const int WM_USER = 0x400;
    public const int WS_CHILD = 0x40000000;
    public const int WS_VISIBLE = 0x10000000;
    public const int SWP_NOMOVE = 0x2;
    public const int SWP_NOZORDER = 0x4;
    public const int WM_CAP_DRIVER_CONNECT = WM_USER + 10;
    public const int WM_CAP_DRIVER_DISCONNECT = WM_USER + 11;
    public const int WM_CAP_SET_CALLBACK_FRAME = WM_USER + 5;
    public const int WM_CAP_SET_PREVIEW = WM_USER + 50;
    public const int WM_CAP_SET_PREVIEWRATE = WM_USER + 52;
    public const int WM_CAP_SET_VIDEOFORMAT = WM_USER + 45;
    public const int WM_CAP_START = WM_USER;
    public const int WM_CAP_SAVEDIB = WM_CAP_START + 25;
}
public class cVideo     //视频类
{
    private IntPtr lwndC;       //保存无符号句柄
    private IntPtr mControlPtr; //保存管理指示器
    private int mWidth;
    private int mHeight;
    public cVideo(IntPtr handle, int width, int height)
    {
        mControlPtr = handle; //显示视频控件的句柄
        mWidth = width;      //视频宽度
        mHeight = height;    //视频高度
    }
    /// <summary>
```

```
        /// 打开视频设备
        /// </summary>
        public void StartWebCam()
        {
            byte[] lpszName = new byte[100];
            byte[] lpszVer = new byte[100];
            VideoAPI.capGetDriverDescriptionA(0, lpszName, 100, lpszVer, 100);
            this.lwndC = VideoAPI.capCreateCaptureWindowA(lpszName, VideoAPI.WS_CH
ILD | VideoAPI.WS_VISIBLE, 0, 0, mWidth, mHeight, mControlPtr, 0);
            if (VideoAPI.SendMessage(lwndC, VideoAPI.WM_CAP_DRIVER_CONNECT, 0,
0))
            {
                VideoAPI.SendMessage(lwndC, VideoAPI.WM_CAP_SET_PREVIEWRATE, 100,
 0);
                VideoAPI.SendMessage(lwndC, VideoAPI.WM_CAP_SET_PREVIEW, true, 0);
            }
        }
        /// <summary>
        /// 关闭视频设备
        /// </summary>
        public void CloseWebcam()
        {
            VideoAPI.SendMessage(lwndC, VideoAPI.WM_CAP_DRIVER_DISCONNECT, 0,
0);
        }
        ///  <summary>
        ///  拍照
        ///  </summary>
```

```
    ///   <param   name="path">要保存 bmp 文件的路径</param>
    public void GrabImage(IntPtr hWndC, string path)
    {
        IntPtr hBmp = Marshal.StringToHGlobalAnsi(path);
        VideoAPI.SendMessage(lwndC, VideoAPI.WM_CAP_SAVEDIB, 0, hBmp.ToInt32
());
    }
  }
```

Form1 窗体中通过调用视频类中的方法来实现相应的功能。

在【打开视频】按钮的 Click 事件中添加如下代码：

```
    private void button1_Click(object sender, EventArgs e)
    {
        btnPlay.Enabled = false;
        btnStop.Enabled = true;
        btnPz.Enabled = true;
        video = new cVideo(pictureBox1.Handle, pictureBox1.Width, pictureBox1.Heig
ht);
        video.StartWebCam();
    }
```

在【关闭视频】按钮的 Click 事件中添加如下代码：

```
    private void b_stop_Click(object sender, EventArgs e)
    {
        btnPlay.Enabled = true;
        btnStop.Enabled = false;
        btnPz.Enabled = false;
        video.CloseWebcam();
    }
```

在【拍摄照片】按钮的 Click 事件下添加如下代码：

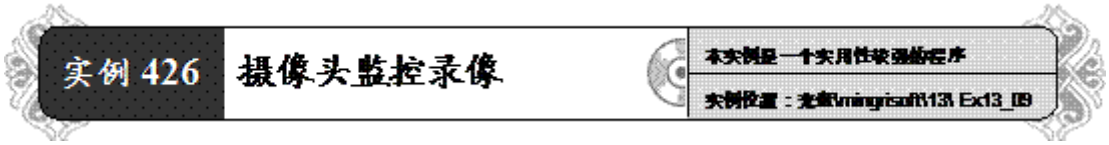
图 13.10 监控录像

```
    private void btnPz_Click(object sender, EventArgs e)
    {
        video.GrabImage(pictureBox1.Handle, "d:\\a.bmp");
    }
```

举一反三

根据本实例，读者可以开发以下程序。

- 无人值班视频实时监控系统。
- 车库安全实时监控系统。



实例 426　摄像头监控录像

实例说明

本例是为通过摄像头来实现监控录像的程序。运行本例后，单击【开始监控】按钮，程序将自动开始录像，录像文件（1x.avi）将保存在 D 盘根目录下。运行程序，效果如图 13.10 所示。

技术要点

在实例“简易视频程序”的技术要点中，使用的技术和相关函数已经介绍过。在这里主要介绍如何将捕获的视频制作成 .AVI 媒体文件。实现技术为主要通过 SendMessage 函数发送 Windows 消息机制，消息值 WM_CAP_FILE_SET_CAPTURE_FILEA 和 WM_CAP_SEQUENCE，分别用来设置视频捕捉的文件名称和初始化视频流，捕捉视频信息到文件：

```
    private const int WM_USER = 0x400;
    private const int WM_CAP_START = WM_USER;
    private const int WM_CAP_SEQUENCE = WM_CAP_START + 62;
    private const int WM_CAP_FILE_SET_CAPTURE_FILEA = WM_CAP_START + 2
0;
```

实现关键代码如下：

```
IntPtr hBmp = Marshal.StringToHGlobalAnsi(path);
SendMessage(hWndC,WM_CAP_FILE_SET_CAPTURE_FILEA,0, hBmp.ToInt32
());
SendMessage(hWndC, WM_CAP_SEQUENCE, 0, 0);
```

实现过程

（1）新建一个项目，命名为 Ex13_09，默认窗体为 Form1，添加一个类文件（.CS），用于编写视频类。

（2）在 Form1 窗体中，主要添加一个 PictrueBox 控件，用于显示视频；添加 4 个 Button 控件，用于开始监控、停止监控和监控程序。

（3）视频类中主要程序代码如下：

```
/// <summary>
/// 开始录像
/// </summary>
/// <param name="path">要保存录像的路径</param>
public void StarKinescope(string path)
{
   IntPtr hBmp = Marshal.StringToHGlobalAnsi(path);
   SendMessage(hWndC,WM_CAP_FILE_SET_CAPTURE_FILEA,0, hBmp.ToInt32
());
   SendMessage(hWndC, WM_CAP_SEQUENCE, 0, 0);
}
/// <summary>
/// 停止录像
/// </summary>
public void StopKinescope()
{
```

```
            SendMessage(hWndC, WM_CAP_STOP, 0, 0);
        }
```

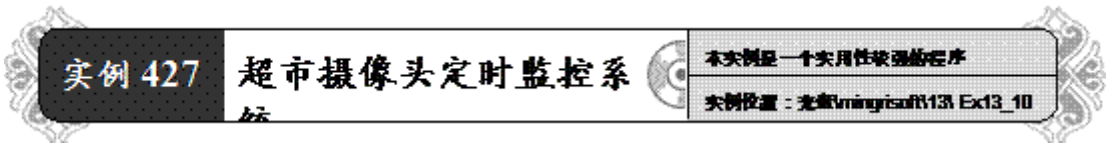
（4）Form1 窗体中主要程序代码如下：

```
//开始录像
        private void button1_Click(object sender, EventArgs e)
        {
            btnStar.Enabled = false;
            btnStop.Enabled = true;
            video.StarKinescope(@"d:\lx.avi");
        }
//停止录像
        private void button2_Click(object sender, EventArgs e)
        {
            btnStar.Enabled = true;
            btnStop.Enabled = false;
            video.StopKinescope();
        }
```

举一反三

根据本实例，读者可以开发以下程序。

- 小区视频监控录像系统。
- 公司财务室视频监控系统。



实例 427　超市摄像头定时监控系统

实例说明

本实例实现超市摄像头定时监控系统。运行本例后，在“定时监控设置”处设置监控的星期及时间，单击【保存】按钮，将“定时设置”参数数据保存到数据库中。系统在运行到定时时间后，程序将自动进行监控。如图 13.11 所示。另外，监控的录像文件和图片文件保存在 D 盘根目录中，命名格式为系统当前日期。

图 13.11　超市摄像头定时监控

技术要点

相关技术要点请参见实例“摄像头监控录像”。另外，本实例利用 Timer 控件中的定时执行功能，进行数据的定时录像工作。

实现过程

（1）新建一个项目，命名为 Ex13_10，默认窗体为 Form1，添加一个类文件（.CS），用于编写视频类。

（2）在 Form1 窗体中，主要添加一个 PictrueBox 控件，用于显示视频；其他控件的添加如图 13.11 所示。

（3）主要程序代码。

```
private void timer1_Tick(object sender, EventArgs e)
{
    string strTime="";
    //星期一
    if (chk1.Checked && Convert.ToInt32(DateTime.Now.DayOfWeek)==1)
    {
        strTime = DateTime.Now.ToString("HH:mm");
        DateTime date = Convert.ToDateTime(mtxt1.Text);
        if (strTime == date.ToString("HH:mm"))
            video.StarKinescope(@"d:\" + DateTime.Today.Month.ToString() + Date
Time.Today.Day.ToString() + DateTime.Now.Hour.ToString() + DateTime.Now.Minute.
ToString() + DateTime.Now.Second.ToString() + ".avi");
    }
    //星期二
    if (chk1.Checked && Convert.ToInt32(DateTime.Now.DayOfWeek) == 2)
    {
        strTime = DateTime.Now.ToString("HH:mm");
        DateTime date = Convert.ToDateTime(mtxt2.Text);
```

```
            if (strTime == date.ToString("HH:mm"))
                video.StarKinescope(@"d:\" + DateTime.Today.Month.ToString() + Date
Time.Today.Day.ToString() + DateTime.Now.Hour.ToString() + DateTime.Now.Minute.
ToString() + DateTime.Now.Second.ToString() + ".avi");
        }
        //星期三
        if (chk1.Checked && Convert.ToInt32(DateTime.Now.DayOfWeek) == 3)
        {
            strTime = DateTime.Now.ToString("HH:mm");
            DateTime date = Convert.ToDateTime(mtxt3.Text);
            if (strTime == date.ToString("HH:mm"))
                video.StarKinescope(@"d:\" + DateTime.Today.Month.ToString() + Date
Time.Today.Day.ToString() + DateTime.Now.Hour.ToString() + DateTime.Now.Minute.
ToString() + DateTime.Now.Second.ToString() + ".avi");
        }
        //星期四
        if (chk1.Checked && Convert.ToInt32(DateTime.Now.DayOfWeek) == 4)
        {
            strTime = DateTime.Now.ToString("HH:mm");
            DateTime date = Convert.ToDateTime(mtxt4.Text);
            if (strTime == date.ToString("HH:mm"))
                video.StarKinescope(@"d:\" + DateTime.Today.Month.ToString() + Date
Time.Today.Day.ToString() + DateTime.Now.Hour.ToString() + DateTime.Now.Minute.
ToString() + DateTime.Now.Second.ToString() + ".avi");
        }
        //星期五
        if (chk1.Checked && Convert.ToInt32(DateTime.Now.DayOfWeek) == 5)
        {
```

```
            strTime = DateTime.Now.ToString("HH:mm");
            DateTime date = Convert.ToDateTime(mtxt5.Text);
            if (strTime == date.ToString("HH:mm"))
                video.StarKinescope(@"d:\" + DateTime.Today.Month.ToString() + Date
Time.Today.Day.ToString() + DateTime.Now.Hour.ToString() + DateTime.Now.Minute.
ToString() + DateTime.Now.Second.ToString() + ".avi");
        }
        //星期六
        if (chk1.Checked && Convert.ToInt32(DateTime.Now.DayOfWeek) == 6)
        {
            strTime = DateTime.Now.ToString("HH:mm");
            DateTime date = Convert.ToDateTime(mtxt6.Text);
            if (strTime == date.ToString("HH:mm"))
                video.StarKinescope(@"d:\" + DateTime.Today.Month.ToString() + Date
Time.Today.Day.ToString() + DateTime.Now.Hour.ToString() + DateTime.Now.Minute.
ToString() + DateTime.Now.Second.ToString() + ".avi");
        }
        //星期日
        if (chk1.Checked && Convert.ToInt32(DateTime.Now.DayOfWeek) == 7)
        {
            strTime = DateTime.Now.ToString("HH:mm");
            DateTime date = Convert.ToDateTime(mtxt7.Text);
            if (strTime == date.ToString("HH:mm"))
                video.StarKinescope(@"d:\" + DateTime.Today.Month.ToString() + Date
Time.Today.Day.ToString() + DateTime.Now.Hour.ToString() + DateTime.Now.Minute.
ToString() + DateTime.Now.Second.ToString() + ".avi");
        }
```

图 19.12 语音卡电话呼叫系统

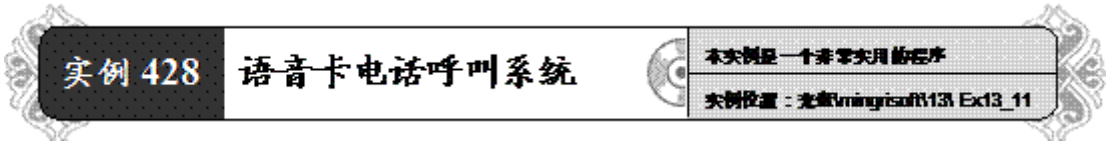
举一反三

根据本实例，读者可以开发以下程序。

- 车站定时监控系统。
- 公司定时安防系统。

13.5 语音卡控制

随着语音技术的不断发展，语音卡在通信行业应用非常广泛。本节通过几个典型实例介绍语音卡程序的开发。



实例 428 语音卡电话呼叫系统

实例说明

随着科学技术的不断发展，语音卡被广泛地应用于商业软件中。本例实现了利用语音卡实现电话呼叫的功能。实例运行结果如图 13.12 所示。

技术要点

本例采用东进公司开发的 8 路模拟语音卡，该卡采用灵活的模式化设计，可按需配置外线、内线两种模块。该语音卡可实现坐席、会议、FSK 数据收发、语音合成等多种功能，并提供 SDK 开发工具包。

在安装完驱动程序后，相应的动态链接库（NewSig.dll 和 Tc08a32.dll 文件）会复制到 Windows 的系统目录下。在语音卡的开发过程中，主要通过调用 NewSig.dll 和 Tc08a32.dll 来实现相应的功能。下面介绍这两个动态库中的主要使用函数。

（1）LoadDRV 函数

该函数用于加载动态链接库。语法如下：

```
[DllImport("Tc08a32.dll", CharSet = CharSet.Auto)]
```

```
public static extern long LoadDRV();
```

返回值：返回值为 0 表示成功；-1 表示打开设备驱动程序错误。-2 表示在读取 TC08A-V.INI 文件时发生错误；-3 表示 INI 文件的设置与实际的硬件不一致时发生错误。

（2）FreeDRV 函数

该函数用于关闭驱动程序。语法如下：

```
[DllImport("Tc08a32.dll", CharSet = CharSet.Auto)]
public static extern long EnableCard(short wusedCh, short wFileBufLen);
```

（3）EnableCard 函数

该函数用于初始化语音卡硬件，并为每个通道分配语音缓冲区。语法如下：

```
[DllImport("Tc08a32.dll", CharSet = CharSet.Auto)]
public static extern long EnableCard(short wusedCh, short wFileBufLen);
```

参数说明如下。

- wUsedCh：标识通道数量。
- WFileBufLen：标识分配的缓冲区大小。

（4）CheckValidCh 函数

该函数检测在当前机器内可用的通道总数。语法如下：

```
[DllImport("Tc08a32.dll", CharSet = CharSet.Auto)]
public static extern short CheckValidCh();
```

- 返回值：通道总数量。

（5）CheckChType 函数

该函数用于测试某个通道的类型。语法如下：

```
[DllImport("Tc08a32.dll", CharSet = CharSet.Auto)]
public static extern short CheckChType(short wChnlNo);
```

参数说明如下。

- wChnlNo：标识通道号。
- 返回值：为 0 表示内线；为 1 表示外线；为 2 表示悬空。

（6）PUSH_PLAY 函数

该函数用于维持文件录放音的持续进行，需在处理函数的大循环中调用。语法如下：

```
[DllImport("Tc08a32.dll", CharSet = CharSet.Auto)]
public static extern void PUSH_PLAY();
```

（7）SetBusyPara 函数

该函数用于设置要检测的挂机忙音的参数。语法如下：

```
[DllImport("Tc08a32.dll", CharSet = CharSet.Auto)]
public static extern void SetBusyPara(short BusyLen);
```

参数说明：

l BusyLen：标识忙音的时间长度，单位为毫秒。

（8）RingDetect 函数

该函数用于测试外线是否振铃或内线是否提机。语法如下：

```
[DllImport("Tc08a32.dll", CharSet = CharSet.Auto)]
public static extern bool RingDetect(short wChnlNo);
```

参数说明如下。

l wChnlNo：标识通道号。

返回值：如果为 1，对于外线表示有振铃信息；对于内线，表示提机。如果为 0，对于外线，表示无振铃信息；对于内线，表示挂机。

（9）OffHook 函数

该函数用于外线提机。对于内线，不起作用。语法如下：

```
[DllImport("Tc08a32.dll", CharSet = CharSet.Auto)]
public static extern void OffHook(short wChnlNo);
```

参数说明如下。

l wChnlNo：标识外线通道。

（10）HangUp 函数

该函数用于外线挂机。对于内线，不起作用。语法如下：

```
[DllImport("Tc08a32.dll", CharSet = CharSet.Auto)]
public static extern void HangUp(short wChnlNo);
```

参数说明如下。

l wChnlNo：标识外线通道。

（11）Sig_Init 函数

该函数用于完成信号音检测的初始化工作。语法如下：

```
[DllImport("NewSig.dll", CharSet = CharSet.Auto)]
public static extern void Sig_Init(int Times);
```

参数说明如下。

l wPara：缺省值为 0，不起作用。

（12）Sig_CheckBusy 函数

清空忙音检测的缓冲区以及内部计数。当检测对方挂机的忙音后，必须调用本函数。语法如下：

```
[DllImport("NewSig.DLL", CharSet = CharSet.Auto)]
public static extern void Sig_ResetCheck(short wChlNo);
```

参数说明如下。

l wChNo：标识通道号。

l 返回值：为 1 表示检测到忙音；为 0，表示没有检测到忙音。

（13）Sig_ResetCheck 函数

该函数用于清空忙音检测的缓冲区以及内部计数。当检测对方挂机的忙音后，必须调用本函数。语法如下：

```
[DllImport("NewSig.DLL", CharSet = CharSet.Auto)]
public static extern void Sig_ResetCheck(short wChlNo);
```

参数说明如下。

l wChNo：标识通道号。

（14）Sig_StartDial 函数

该函数用于拨打电话号码。开始某通道的呼出过程。该函数只是设置通道的呼出缓冲区，真正的呼出过程需要循环调用 Sig_CheckDial 函数来逐步完成。语法如下：

```
[DllImport("NewSig.dll", CharSet = CharSet.Auto)]
```

```
public static extern int Sig_StartDial(short wChNo, [MarshalAs(UnmanagedType.
LPArray)] byte[] DialNum, [MarshalAs(UnmanagedType.LPArray)] byte[] PreDialNum,
short wMode);
```

参数说明如下。

- wChNo：标识通道号。
- DialNum：标识呼出号码。
- PreDialNum：标识前导号码。
- wMode：呼出检测的模式。

（15）Sig_CheckDial 函数

该函数用于检测呼出结果。

在调用函数 Sig_StartDial 启动拨号过程后，就可以循环调用 Sig_CheckDial 函数维持拨号过程，并检测呼出的结果，直至得到结果为止。

拨号的一般过程如下。

1．如果参数 PreDialNum 不为空，则延迟 1 秒后拨出 PreDialNum，如果参数 PreDialNum 为空，则直接进入步骤 3 。

2．检测 PreDialNum 是否已发完。如已发完转至步骤 3 。

3．检测是否有拨号音，如拨号音长度达到配置项 DialToneAfterOffHook 的数值，则发送 DialNum 码串，并转至步骤 4 。如在此步骤已等待配置项 NoDialToneAfterOffHook 定义的时间长度仍未检测到拨号音，则返回 0x10。

4．检测 DialNum 码串是否发完，如已发完则延迟 StartDelay 配置项的时间长度后进入步骤 5 。

5．如果从进入此步骤起已经过配置项 RingLen 定义的时间长度，拨号音仍未停止则返回 0x10；如果在此步骤已等待配置项 NoRingLen 定义的时间长度仍未检测到回铃音则返回 0x10；如果检测到占线忙音数达到配置项 BusySigCount 定义的数字，则返回 0x21；如果检测到对方摘机，则返回 0x14；如果进入此步骤已经过配置项 Ringback_NoAnswerTime 定义的时间长度，并且已检测到回铃音，则返回 0x13；其他情况返回 0x10。

注意：在进行呼出结果检测之前必须调用函数 StartSigCheck 启动信号音采集过程，并且在进行呼出结果检测时，要循环调用 FeedSigFunc 函数维持信号音采集过程。

语法如下：

```
[DllImport("NewSig.dll", CharSet = CharSet.Auto)]
public static extern int Sig_CheckDial(short wChNo);
```

参数说明如下。

- wChNo：标识通道号。
- 返回值包括以下几种情况。
- 16(0x10)：尚未得出结果。
- 15(0x0F)：没有拨号音。
- 33(0x21)：检测到对方占线的忙音。
- 20(0x14)：对方摘机，可以进行通话。
- 19(0x13)：振铃若干次，无人接听电话。
- 21(0x15)：没有信号音。

注意：关于语音卡其他函数语法请参见光盘中的本实例文件 D161A.CS，该文件给出大部分语音卡的函数语法。

实现过程

（1）新建一个项目，命名为 Ex13_11，默认窗体为 Form1。

（2）在 Form1 窗体中，主要添加两个 Button 控件，用于执行电话拨号和电话挂机，添加一个 DataGridView 控件，显示语音卡各通道及通道状态，添加 Timer 组件实现电话的呼出过程，添加一个 TextBox 控件，用于输入呼出电话号码。

（3）主要程序代码。

在窗体装载事件中，主要进行初始化语音卡驱动程序，并且检测通道总数及状态，为每一条通道分配语音缓冲区。代码如下：

```
private void Form1_Load(object sender, EventArgs e)
{
```

```
//初始化驱动程序
long load = DJ160API.LoadDRV();
//检测通道总数,并为每个通道分配语音缓冲区
short wuseCh = DJ160API.CheckValidCh();
short wFileBufLen = 16 * 1024;
long card = DJ160API.EnableCard(wuseCh, wFileBufLen);
//设置表格通道的行数
dataGridView1.RowCount = wuseCh;
//检测每个通道类型
short chanelTpye = 0; //定义通道类型变量
string strType = "";
for (short i = 0; i < wuseCh; i++)
{
   chanelTpye = DJ160API.CheckChType(i);
   dataGridView1[0, i].Value = i;
   switch (chanelTpye)
   {
      case 0:
         strType = "内线";
         break;
      case 1:
         strType = "外线";
         break;
      case 2:
         strType = "悬空";
         break;
   }
   dataGridView1[1, i].Value = strType;
```

```
            dataGridView1[2, i].Value = "空闲";
        }
    }
```

在 DataGridView 控件中选择一个外线空闲通道，单击【拨号】按钮，进行电话拨号，并且将拨号过程中的状态显示在相应的 DataGirdView 表格中。代码如下：

```
    private void button1_Click(object sender, EventArgs e)
    {
        short wuseCh = DJ160API.CheckValidCh();
        short wFileBufLen = 16 * 1024;
        long card = DJ160API.EnableCard(wuseCh, wFileBufLen);
        DJ160API.Sig_Init(chanel);
        //检查（外线）是否有振铃信号或（内线）是否有提机
        bool ring = DJ160API.RingDetect(chanel);
        //外线提机
        DJ160API.OffHook(chanel);
        byte[] ss =new byte[textBox1.Text.Length];
        byte[] s ={ 0 };
        for (int i = 0; i < textBox1.Text.Length; i++)
        {
            ss[i] = Convert.ToByte(textBox1.Text.Substring(i, 1));
        }
        DJ160API.Sig_StartDial(chanel, ss, s, 0);
        timer1.Enabled = true;
        dataGridView1[2, chanel].Value = "拨号中...";
        dataGridView1[3, chanel].Value = textBox1.Text;
    }
```

单击【挂机】按钮，实现电话挂机功能。代码如下：

```
private void button2_Click(object sender, EventArgs e)
{
    DJ160API.HangUp(chanel);
    DJ160API.Sig_ResetCheck(chanel);
    DJ160API.StartSigCheck(chanel);
    timer1.Enabled = false;
    dataGridView1[2, chanel].Value = "空闲";
    dataGridView1[3, chanel].Value = "";
}
```

Sig_StartDial 函数用于拨打电话号码。开始某通道的呼出过程。该函数只是设置通道的呼出缓冲区，真正的呼出过程需循环调用 Sig_CheckDial 函数来逐步完成。代码如下：

```
private void timer1_Tick(object sender, EventArgs e)
{
    DJ160API.Sig_CheckDial(chanel);
}
```

单击 DataGridView 控件的相应行记录相应的通道号，代码如下：

```
private void dataGridView1_CellClick(object sender, DataGridViewCellEventArgs e)
{
    chanel = (short)e.RowIndex;
}
```

举一反三

根据本实例，读者可以开发以下程序。

- 实现电话自助服务系统。
- 实现电话自动报警系统。

实例 429　客户来电查询系统

实例说明

随着市场竞争的加剧，企业越来越重视客户服务和市场反馈。本例实现了电话来电的显示支持功能。当有客户打入电话时，会读取客户的电话号码，根据电话号码可以提取客户的相关信息，方便客服人员有针对性地进行服务。实例运行结果如图 13.13 所示。

图 13.13　客户来电查询

技术要点

其他相关函数介绍请参见实例“语音卡电话呼叫系统”，本实例主要介绍 GetCallerIDStr 函数，该函数用于获取主叫号码。语法如下：

```
[DllImport("Tc08a32.dll", CharSet = CharSet.Auto)]
public static extern short GetCallerIDStr(short wChnlNo, byte[] IDStr);
```

参数说明如下。

- wChnlNo：标识通道号。
- IDStr：用于接收读取的号码。
- 返回值：为 0，表示未收到任何信息；为 1，表示正在接收头信息；为 2 表示正在接收 ID 号码；为 3 表示接收完毕，校验正确；为 4 表示接收完毕，校验错误。

在调用 GetCallerIDStr 函数时，只有返回值为 3 或 4 才表示已经正确接收了主机号码。

实现过程

（1）新建一个项目，命名为 Ex13_12，默认窗体为 Form1。

（2）在 Form1 窗体中，主要添加一个 DataGridView 控件，显示语音卡各通道和通道状态，并在来电时显示来电号码；添加一个 Timer 控件用于时刻检测来电信息；添加其他控件及用途如图 13.13 所示。

（3）主要程序代码。

```
private void timer1_Tick(object sender, EventArgs e)
    {
        byte[] ss = new byte[100];
        for (short i = 0; i < 8; i++)
        {
            DJ160API.StartSigCheck(i);
            if(open_close==false)
                DJ160API.ResetCallerIDBuffer(i);
            if (DJ160API.RingDetect(i))
            {
                open_close = true;
                //获取来电号码
                result = DJ160API.GetCallerIDStr(i, ss);
                if (result == 3 || result == 4)
                {
                    string str = Encoding.UTF8.GetString(ss);
                    txtTel.Text = str;
                    txtTel.Text = txtTel.Text.Substring(txtTel.Text.Length - 8, 8);
                    dataGridView1[2, i].Value = txtTel.Text;
                    //查询客户资料
                    this.getMessage(txtTel.Text);
```

```
            }
        }
    }
}
private void getMessage(string str)
{
    OleDbConnection con = new OleDbConnection("Provider=Microsoft.Jet.OLEDB.
4.0;Data Source=" + "db_csell.mdb" + ";Persist Security Info=False");
    OleDbDataAdapter dap = new OleDbDataAdapter("SELECT * FROM 个人名录
表 WHERE 电话='" + str + "'", con);
    DataSet ds = new DataSet();
    dap.Fill(ds);
    if (ds.Tables[0].Rows.Count > 0)
    {
        txtName.Text = ds.Tables[0].Rows[0]["姓名"].ToString();
        txtDuty.Text = ds.Tables[0].Rows[0]["职务"].ToString();
        txtAddress.Text = ds.Tables[0].Rows[0]["地址"].ToString();
        txtMobile.Text = ds.Tables[0].Rows[0]["手机"].ToString();
        txtCompany.Text = ds.Tables[0].Rows[0]["公司名称"].ToString();
        txtPostId.Text = ds.Tables[0].Rows[0]["邮编"].ToString();
    }
    else
    {
        labStatus.Text = "非本单位会员客户。。。。";
    }
}
```

举一反三

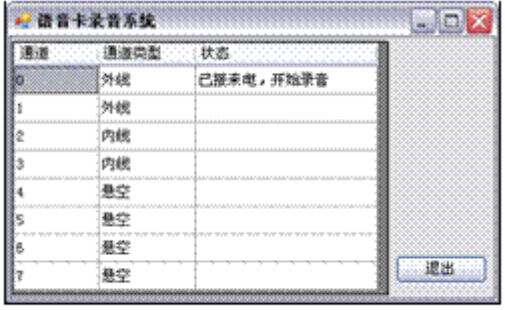
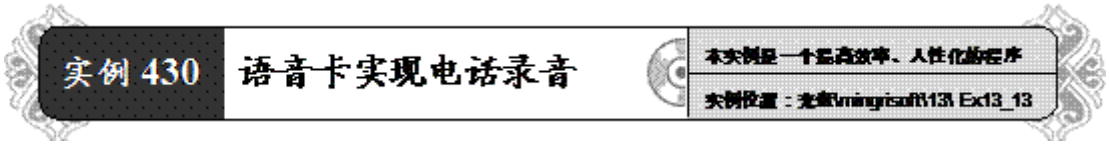


图 13.14 利用语音卡实现电话录音

根据本实例，读者可以开发以下程序。

- 用语音卡实现客户某项费用的查询。
- 用语音卡实现客户在某个时间的留言信息。
- 用语音卡实现客户购买物品的查询。
- 实现客户反馈电话录音系统。



实例 430　语音卡实现电话录音

实例说明

如今的许多电话都具有电话录音的功能。本例实现了该功能，当有电话打入时，即刻将双方的对话信息进行录音。实例运行结果如图 13.14 所示。

技术要点

其他相关函数介绍请参见实例“语音卡电话呼叫系统”，本实例主要介绍 StartRecordFile 函数和 StopRecordFile 函数。

（1）StartRecordFile 函数用于开始文件录音。停止该方式的录音一定要用 StopRecordFile 函数。检查录音是否结束，用 CheckRecordEnd 函数。StartRecordFile 函数语法如下：

```
[DllImport("Tc08a32.dll", CharSet = CharSet.Auto)]
public static extern bool StartRecordFile(short wChnlNo, byte[] FileName, long dwRecordLen);
```

参数说明如下。

- wChnINo：标识录音的通道号。
- FileName：标识录音的文件名。
- dwRecordLen：标识文件大小。

（2）StopRecordFile 函数用于停止录音。该函数语法如下：

```
[DllImport("Tc08a32.dll", CharSet = CharSet.Auto)]
```

```
public static extern void StopRecordFile(short wChnlNo);
```

参数说明如下。

l wChnINo：标识要停止的录音通道。

（3）CheckRecordEnd 函数检查指定通道录音是否结束（缓冲区已满）。

```
[DllImport("Tc08a32.dll", CharSet = CharSet.Auto)]
public static extern int CheckRecordEnd(int ChannelNo);
```

参数说明如下。

l wChnINo：标识录音的通道号。

l 返回值：0 表示未结束；1 代表结束。

实现过程

（1）新建一个项目，命名为 Ex13_13，默认窗体为 Form1。

（2）在 Form1 窗体中，主要添加一个 DataGridView 控件，显示语音卡各通道和通道状态，并在来电时显示来电号码；添加一个 Timer 控件用于实时检测来电信息，如果来电，程序将自动摘机并且实现录音；添加其他控件及用途如图 13.14 所示。

（3）主要程序代码。

```
private void timer1_Tick(object sender, EventArgs e)
{
    //维持文件录音持续执行
    DJ160API.PUSH_PLAY();
    for (short i = 0; i < 8; i++)
    {
        DJ160API.StartSigCheck(i);
        if (open_close == false)
            DJ160API.ResetCallerIDBuffer(i);
        if (DJ160API.RingDetect(i))
        {
```

```
                open_close = true;
                //摘机
                DJ160API.OffHook(i);
                DJ160API.StartSigCheck(i);
                //是否挂机
                if (DJ160API.ReadCheckResult(i, 2) != 33)
                {
                    bool bl = DJ160API.StartRecordFile(i, Encoding.UTF8.GetBytes(@"D:\l
y.001"), 600 * 1024);
                    dataGridView1[2, i].Value = "已接来电，开始录音";
                }
                else
                {
                    DJ160API.StopRecordFile(i);
                    open_close = false;
                    DJ160API.Sig_ResetCheck(i);
                    dataGridView1[2, i].Value = "";
                }
                if (DJ160API.CheckRecordEnd(i)==1)
                {
                    DJ160API.StopRecordFile(i);
                    open_close = false;
                    dataGridView1[2, i].Value = "";
                }
            }
        }
    }
```

图 13.15 利用短信猫收发短信息

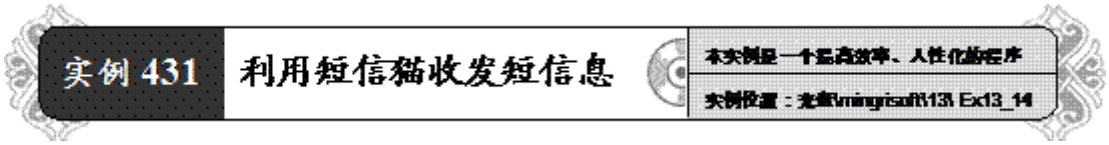
举一反三

根据本实例，读者可以实现以下功能。

- 用语音卡实现电话点播歌曲，通过按键的选择点播自己喜爱的歌曲。
- 用语音卡实现产品报价，通过按键实现某些产品的报价。
- 利用语音卡实现产品报价。

13.6 手机程序开发

如今手机已成为大众的交流工具。有关手机的程序开发越来越广泛，本节通过几个典型实例介绍如何利用短信猫发送、接收短信、远程控制计算机、业务员销售数据采集和短信息娱乐互动平台。



实例 431 利用短信猫收发短信息

实例说明

短信猫是利用 SIM 卡发送短信的硬件设备，通过串口或 USB 接口（根据设备型号而定）与计算机相连。在程序中可以利用短信猫发送或接收短信。本例实现了利用短信猫收发短信息的功能。实例运行结果如图 13.15 所示。

技术要点

本例使用的是北京人大金仓信息技术有限公司的串口短信猫。在购买短信猫时会附带包括 SDK 的开发包，其中提供了操作短信猫的函数（封装在 dllforvc.dll 动态库中）。下面介绍操作短信猫的主要函数。

（1）GSMModemGetSnInfoNew 函数

该函数获取短信猫注册需要的信息，代码如下：

```
[DllImport("dllforvc.dll",
    EntryPoint = "GSMModemGetSnInfoNew",
    CharSet = CharSet.Ansi,
```

```
        CallingConvention = CallingConvention.StdCall)]
    public static extern string GSMModemGetSnInfoNew(string device, string baudrat
e);
```

参数说明如下。

- device：通信端口，为 null 时系统会自动检测。
- baudrate：通讯波特率，为 null 时系统会自动检测。

（2）GSMModemGetDevice 函数

该函数获取当前的通讯端口，代码如下：

```
    [DllImport("dllforvc.dll",
        EntryPoint = "GSMModemGetDevice",
        CharSet = CharSet.Ansi,
        CallingConvention = CallingConvention.StdCall)]
    public static extern string GSMModemGetDevice();
```

（3）GSMModemGetBaudrate 函数

该函数获取当前的通讯波特率，代码如下：

```
    [DllImport("dllforvc.dll",
        EntryPoint = "GSMModemGetBaudrate",
        CharSet = CharSet.Ansi,
        CallingConvention = CallingConvention.StdCall)]
    public static extern string GSMModemGetBaudrate();
```

（4）GSMModemInitNew 函数

该函数用于初始化短信猫。语法如下：

```
    [DllImport("dllforvc.dll",
        EntryPoint = "GSMModemInitNew",
        CharSet = CharSet.Ansi,
        CallingConvention = CallingConvention.StdCall)]
    public static extern bool GSMModemInitNew(
        string device,
```

```
    string baudrate,
    string initstring,
    string charset,
    bool swHandshake,
    string sn);
```

参数说明如下。

- device：标识通信端口，如果为 NULL，系统会自动检测。
- baudrate：标识通讯波特率，如果为 NULL，系统会自动检测。
- initstring：标识初始化命令，为 NULL 即可。
- charset：标识通讯字符集，为 NULL 即可。
- swHandshake：标识是否进行软件握手，为 False 即可。
- sn：标识短信猫的授权号，需要根据实际情况填写。

（5）GSMModemSMSsend 函数

该函数用于发送手机短信。语法如下：

```
[DllImport("dllforvc.dll",
    EntryPoint = "GSMModemSMSsend",
    CharSet = CharSet.Ansi,
    CallingConvention = CallingConvention.StdCall)]
public static extern bool GSMModemSMSsend(
    string serviceCenterAddress,
    int encodeval,
    string text,
    int textlen,
    string phonenumber,
    bool requestStatusReport);
```

参数说明如下。

- serviceCenterAddress：标识短信中心号码，为 NULL 即可。
- encodeval：标识短信息编码格式，如果为 8，表示中文短信编码。

l text：标识短信内容。

l textlen：标识短信内容的长度。

l phonenumber：标识接收短信的电话号码。

l requestStatusReport：标识状态报告。

（6）GSMModemSMSReadAll 函数

该函数取得所有短信息，包括 SIM 卡和手机中的短信息。返回的短信内容格式为电话号码 1|短信内容 1||电话号码 2|短信内容 2||：

```
//接收短信息返回字符串格式为:手机号码|短信内容||手机号码|短信内容||
//RD_opt 为 1 表示接收短信息后不做任何处理，为 0 表示接收后删除信息
[DllImport("dllforvc.dll",
    EntryPoint = "GSMModemSMSReadAll",
    CharSet = CharSet.Ansi,
    CallingConvention = CallingConvention.StdCall)]
public static extern string GSMModemSMSReadAll(int RD_opt);
```

参数说明如下。

l RD_opt：对读取后的短信息进行处理，0 表示删除，1 表示不做处理。

实现过程

（1）新建一个项目，命名为 Ex13_14，默认窗体为 Form1。

（2）在 Form1 窗体中，主要添加 TextBox 控件和 Label 控件，控件的数量及用途如图 13.15 所示，添加两个 Button 控件，分别用于发送短信息和接收短信息。

（3）主要程序代码。

将所使用的函数封装在 GMS 类中。代码如下：

```
class GMS
{
    //初始化 gsm modem,并连接 gsm modem
    [DllImport("dllforvc.dll",
```

```
        EntryPoint = "GSMModemInitNew",
        CharSet = CharSet.Ansi,
        CallingConvention = CallingConvention.StdCall)]
    public static extern bool GSMModemInitNew(
        string device,
        string baudrate,
        string initstring,
        string charset,
        bool swHandshake,
        string sn);
    //获取短信猫新的标识号码
    [DllImport("dllforvc.dll",
        EntryPoint = "GSMModemGetSnInfoNew",
        CharSet = CharSet.Ansi,
        CallingConvention = CallingConvention.StdCall)]
    public static extern string GSMModemGetSnInfoNew(string device, string baudrat
e);
    //获取当前通讯端口
    [DllImport("dllforvc.dll",
        EntryPoint = "GSMModemGetDevice",
        CharSet = CharSet.Ansi,
        CallingConvention = CallingConvention.StdCall)]
    public static extern string GSMModemGetDevice();
    //获取当前通讯波特率
    [DllImport("dllforvc.dll",
        EntryPoint = "GSMModemGetBaudrate",
        CharSet = CharSet.Ansi,
        CallingConvention = CallingConvention.StdCall)]
```

```
public static extern string GSMModemGetBaudrate();
//断开连接并释放内存空间
[DllImport("dllforvc.dll",
    EntryPoint = "GSMModemRelease",
    CharSet = CharSet.Ansi,
    CallingConvention = CallingConvention.StdCall)]
public static extern void GSMModemRelease();
//取得错误信息
[DllImport("dllforvc.dll",
    EntryPoint = "GSMModemGetErrorMsg",
    CharSet = CharSet.Ansi,
    CallingConvention = CallingConvention.StdCall)]
public static extern string GSMModemGetErrorMsg();
//发送短信息
[DllImport("dllforvc.dll",
    EntryPoint = "GSMModemSMSsend",
    CharSet = CharSet.Ansi,
    CallingConvention = CallingConvention.StdCall)]
public static extern bool GSMModemSMSsend(
   string serviceCenterAddress,
   int encodeval,
   string text,
   int textlen,
   string phonenumber,
   bool requestStatusReport);
//接收短信息返回字符串格式为:手机号码|短信内容||手机号码|短信内容||
//RD_opt 为 1 接收短信息后不做任何处理， 0 为接收后删除信息
[DllImport("dllforvc.dll",
```

```
        EntryPoint = "GSMModemSMSReadAll",
        CharSet = CharSet.Ansi,
        CallingConvention = CallingConvention.StdCall)]
    public static extern string GSMModemSMSReadAll(int RD_opt);
}
```

在装载 Form1 窗体时，获取设备信息。代码如下：

```
    private void Form1_Load(object sender, EventArgs e)
    {
        //机器号码，当参数为空时，函数自动获取设备信息
        txtJQHM.Text = GMS.GSMModemGetSnInfoNew(txtCOM.Text, txtBTL.Text);
        txtCOM.Text = GMS.GSMModemGetDevice();      //ＣＯＭ１
        txtBTL.Text= GMS.GSMModemGetBaudrate();      //波特率
    }
```

发送短信息。代码如下：

```
    private void btnSend_Click(object sender, EventArgs e)
    {
          if(txtSJHM.Text == "")
          {
        MessageBox.Show("手机号码不能为空！","提示", MessageBoxButtons.OK);
             txtSJHM.Focus();
             return;
          }
          if(txtContent.Text=="")
          {
        MessageBox.Show("短信内容不能为空！", "提示", MessageBoxButtons.OK);
             txtContent.Focus();
             return;
          }
```

```
            //连接设备
            if(GMS.GSMModemInitNew(txtCOM.Text, txtBTL.Text, null, null, false, txtPo
wer.Text)==false)
            {
                MessageBox.Show("设备连接失败!" + GMS.GSMModemGetErrorMsg(),"提
示", MessageBoxButtons.OK);
                return;
            }
            // 发送短信
            if (GMS.GSMModemSMSsend(null, 8, txtContent.Text, Encoding.Default.GetB
yteCount(txtContent.Text),txtSJHM.Text, false) == true)
                MessageBox.Show("短信发送成功!", "提示", MessageBoxButtons.OK);
            else
                MessageBox.Show("短信发送失败!" + GMS.GSMModemGetErrorMsg(), "提
示", MessageBoxButtons.OK);
        }
```

接收短信息。代码如下：

```
        private void btnResvice_Click(object sender, EventArgs e)
        {
            //1)连接设备
            if (GMS.GSMModemInitNew(txtCOM.Text, txtBTL.Text, null, null, false, txtPow
er.Text) == false)
            {
                MessageBox.Show("连接失败!" + GMS.GSMModemGetErrorMsg(), "提示", M
essageBoxButtons.OK);
                return;
            }
            //2)接收短信
```

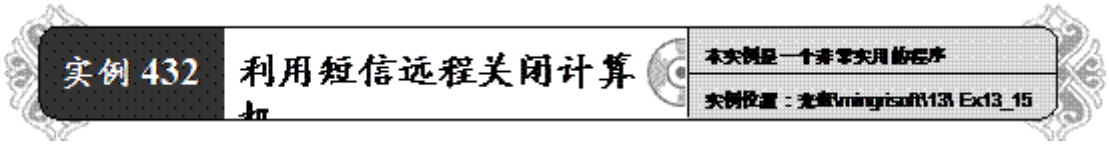
图 13.16 利用短信远程关闭计算机

```
            txtContent.Text = GMS.GSMModemSMSReadAll(1);
            txtSJHM.Text = txtContent.Text.Substring(0, 13);
            txtContent.Text = txtContent.Text.Substring(13, txtCo
ntent.Text.Length-13);
        }
```

举一反三

根据本实例，读者可以开发以下程序。

- 利用短信猫群发短信。
- 办公自动化系统，办公短信通知、短信日程提醒、应急信息短信发布和短信审批等。



实例 432　利用短信远程关闭计算机

实例说明

本例实现了利用短信远程关闭计算机的功能。运行程序，首先，进行关机信息设置；然后，开启服务；最后，通过手机向短信猫发送“关机”数据。片刻之后，指定的计算机将会自动关机。程序如图 13.16 所示。

技术要点

相关函数请参见实例“利用短信猫收发短信息”中的技术要点。

实现过程

（1）新建一个项目，命名为 Ex13_15，默认窗体为 Form1。

（2）在 Form1 窗体中，主要添加 TextBox 控件和 Label 控件，控件的数量及用途如图 13.16 所示，添加一个 Button 控件，用于开启或停止远程关闭计算机服务。

（3）主要程序代码。

```
        private void Form1_Load(object sender, EventArgs e)
        {
          //机器号码
            txtJQHM.Text = GMS.GSMModemGetSnInfoNew(txtCOM.Text, txtBTL.Text);
            txtCOM.Text = GMS.GSMModemGetDevice();      //ＣＯＭ１
            txtBTL.Text = GMS.GSMModemGetBaudrate();      //波特率
            labStatus.Text = "服务关闭中。。。。。";
        }
        private void Close_Windows()
        {
            try
            {
                //指定生成 WMI 连接所需的所有设置
                ConnectionOptions op = new ConnectionOptions();
                op.Username = txtUser.Text;  //远程计算机用户名称
                op.Password = txtPWD.Text;         //远程计算机用户密码
                //设置操作管理范围
         ManagementScope scope = new ManagementScope("\\\\" + txtIP.Text + "\\ro
ot\\cimv2", op);
                scope.Connect();  //将此 ManagementScope 连接到实际的 WMI 范围。
                ObjectQuery oq = new ObjectQuery("SELECT * FROM Win32_OperatingSy
stem");
            ManagementObjectSearcher query = new ManagementObjectSearcher(scope,
 oq);
                //得到 WMI 控制
                ManagementObjectCollection queryCollection = query.Get();
                foreach (ManagementObject obj in queryCollection)
                {
```

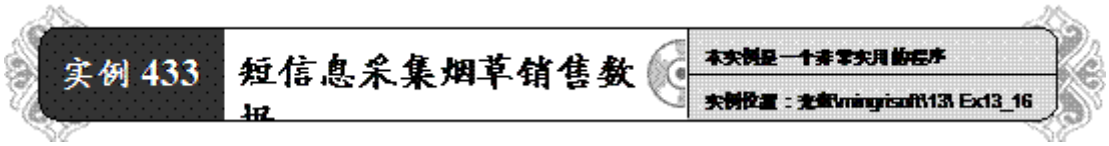
```
            obj.InvokeMethod("ShutDown", null); //执行关闭远程计算机
        }
    }
    catch(Exception ex)
    {
        Process p = new Process();
        p.StartInfo.FileName = "cmd.exe";
        p.StartInfo.UseShellExecute = false;
        p.StartInfo.RedirectStandardInput = true;
        p.StartInfo.RedirectStandardOutput = true;
        p.StartInfo.RedirectStandardError = true;
        p.StartInfo.CreateNoWindow = true;
        p.Start();
        p.StandardInput.WriteLine("shutdown /s");
        p.StandardInput.WriteLine("exit");
    }
}
private void timer1_Tick(object sender, EventArgs e)
{
    //连接设备
    if (GMS.GSMModemInitNew(txtCOM.Text, txtBTL.Text, null, null, false, txtPow
er.Text) == false)
    {
        MessageBox.Show("连接失败!" + GMS.GSMModemGetErrorMsg(), "提示", M
essageBoxButtons.OK);
        return;
    }
    //接收短信
```

```
        string str = GMS.GSMModemSMSReadAll(1);
        if (str==null)
            return;
        if (str.Substring(str.IndexOf("|")+1, 2) == "关机")
        {
            this.Close_Windows();
        }
    }
    private void button1_Click(object sender, EventArgs e)
    {
        if (button1.Text == "开启服务")
        {
            timer1.Enabled = true;
            labStatus.Text = "关机命令采集中。。。。。";
            button1.Text = "停止服务";
        }
        else
        {
            timer1.Enabled = false;
            button1.Text = "开启服务";
            labStatus.Text = "服务关闭中。。。。。";
        }
    }
```

举一反三

根据本实例，读者可以实现以下功能。

- 利用短信实现客户资料查询。
- 保险行业中：保单查询、续费提醒、客户生日提醒和保费计算等。



实例 433　短信息采集烟草销售数据

实例说明

在各类销售行业中，产品销售数据量是企业不可缺少的一项数据。当销售人员在外地出差并且在没有计算机的情况下，如何及时的将销售数据汇报到公司中呢？

本例实现利用短信息采集烟草销售数据的功能。销售人员根据规定的格式编辑短信发送到短信息猫中即可。运行程序，单击【烟草销售信息采集】按钮采集烟草销售数据，然后单击【统计】按钮，将销售数据整理出来。如图 13.17 所示。

技术要点

相关函数请参见实例“利用短信猫收发短信息”中的技术要点。

另外，程序规定的编辑短信息格式为，以冒号“：”分隔并结束。例如“张三：业务员：12：长春：3400：”。其中，主要使用 String.Split(　)方法将信息数据拆分。

图 13.17　短信息采集烟草销售数据

String.Split(　)方法：返回包含此实例中的子字符串（由指定 Char 数组的元素分隔）的 String 数组。

语法：

```
public string[] Split (
    params char[] separator
)
```

参数说明如下。

l　separator：分隔此实例中子字符串的 Unicode 字符数组，不包含分隔符的空数组或空引用。

l　返回值：一个数组，其元素包含此实例中的子字符串，这些子字符串由 separator 中的一个或多个字符分隔。

实现过程

（1）新建一个项目，命名为 Ex13_16，默认窗体为 Form1。

（2）在 Form1 窗体中，主要添加 TextBox 控件和 Label 控件，控件的数量及用途如图 13.17 所示，添加 3 个 Button 控件，分别用于发送短信息、采集销售数据和整理采集数据，添加两个 DataGridView 表格，分别用于显示短信息内容和整理后的销售数据。

（3）主要程序代码。

单击【烟草销售信息采集】按钮，接受短信息并保存到数据库中。代码如下：

```
        private void btnResvice_Click(object sender, EventArgs e)
      {
         //连接设备
         if (GMS.GSMModemInitNew(txtCOM.Text, txtBTL.Text, null, null, false, txtPow
er.Text) == false)
         {
            MessageBox.Show("连接失败!" + GMS.GSMModemGetErrorMsg(), "提示", M
essageBoxButtons.OK);
            return;
         }
         //接收短信
         string content = GMS.GSMModemSMSReadAll(0);
         if (content ==null)
         {
            this.getMessage();
            return;
         }
         content= content.Replace("||", "|"); // 替换||为|
         string[] str_sp = content.Split('|');// 进行分离
         int k=0;
         dataGridView1.ColumnCount = 2;
         dataGridView1.RowCount = str_sp.Length / 2;
         dataGridView1.Columns[0].HeaderText = "手机号码";
         dataGridView1.Columns[1].HeaderText = "短信息";
         for (int i = 0; i < str_sp.Length / 2; i++)
         {
            for (int j = 0; j < 2; j++)
            {
```

```
                dataGridView1[j, i].Value = str_sp[k];
                if (k % 2 != 0)
                    this.InsertMessage("insert into RecivedBox (Mobile,Content,reciveTime)
values('" + Convert.ToString(dataGridView1[0, i].Value) + "','" + Convert.ToString(dat
aGridView1[1, i].Value) + "','" + DateTime.Now.ToString() + "') ");
                k++;
            }
        }
        this.getMessage();
    }
```

自定义方法 getSplitMessage（）用来拆分短信息并且整理为正规数据。代码如下：

```
    private void getSplitMessage()
    {
        string content = "";
        for (int i = 0; i < dataGridView1.Rows.Count; i++)
        {
            content = content + Convert.ToString(dataGridView1["短信息", i].Value);
        }
        string[] str_sp = content.Split('：');// 进行分离
        int k = 0;
        dataGridView2.ColumnCount = 5;
        dataGridView2.RowCount = str_sp.Length/5 ;
        dataGridView2.Columns[0].HeaderText = "姓名";
        dataGridView2.Columns[1].HeaderText = "职务";
        dataGridView2.Columns[2].HeaderText = "月份";
        dataGridView2.Columns[3].HeaderText = "销售地区";
        dataGridView2.Columns[4].HeaderText = "销售数量";
```

```
        for (int i = 0; i < str_sp.Length / 5; i++)
        {
            for (int j = 0; j < 5; j++)
            {
                dataGridView2[j, i].Value = str_sp[k];
                k++;
            }
        }
    }
```

自定义 InsertMessage（）方法，将接受的短信息保存到数据库中。代码如下：

```
    private void InsertMessage(string strSql)
    {
        //将短信息内容添加到数据库中
        OleDbConnection con = new OleDbConnection("Provider=Microsoft.Jet.OLEDB.4.0;Data Source=" + "message.mdb" + ";Persist Security Info=False");
        con.Open();
        OleDbCommand cmd = new OleDbCommand(strSql, con);
        cmd.ExecuteNonQuery();
        con.Close();
    }
```

自定义 getMessage（）方法，获取数据库中所有的短信息数据。代码如下：

```
    private void getMessage()
    {
        OleDbConnection con = new OleDbConnection("Provider=Microsoft.Jet.OLEDB.4.0;Data Source=" + "message.mdb" + ";Persist Security Info=False");
```

```
        OleDbDataAdapter dap = new OleDbDataAdapter("select mobile as 手机号码,
content as 短信息,reciveTime as 日期 from RecivedBox", con);
        DataSet ds = new DataSet();
        dap.Fill(ds);
        dataGridView1.DataSource = ds.Tables[0].DefaultView;
    }
```

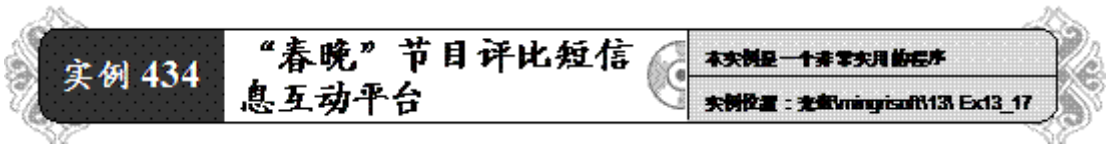
调用自定义 getSplitMessage（）方法，将整理后的销售数据显示在 DataGridView 表格中。代码如下：

```
    private void btnFind_Click(object sender, EventArgs e)
    {
        if(dataGridView1.Rows.Count>0)
        this.getSplitMessage();
    }
```

举一反三

根据本实例，读者可以实现以下功能。

- 利用短信实现销售业绩查询。
- 利用短信实现订购商品。



实例 434 “春晚”节目评比短信息互动平台

实例说明

在观看娱乐电视节目中，主持人经常带动场外的电视观众参与到活动当中。例如，在春节联欢晚会中，通过编辑短信来选取观众最喜欢的春晚节目，那么本实例将实现为“春晚”节目评比的短信息互动平台。电视观众根据规定的格式编辑短信发送到短信息互动平台进行节目评比。运行程序，单击【获取参与观众短信】按钮，即可得到观众的投票，如图 13.18 所示。

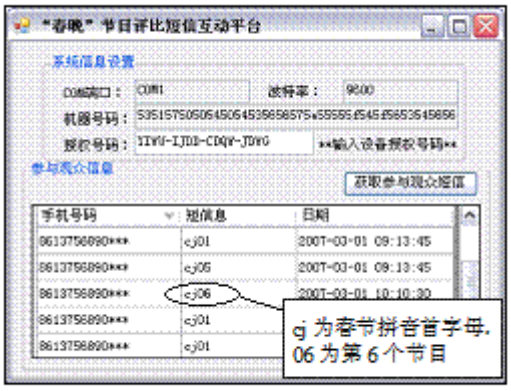


图 13.18 短信息采集节目评价数据

技术要点

相关函数请参见实例“利用短信猫收发短信息”中的技术要点。

实现过程

（1）新建一个项目，命名为 Ex13_17，默认窗体为 Form1。

（2）在 Form1 窗体中，主要添加 TextBox 控件和 Label 控件，控件的数量及用途如图 13.18 所示，添加一个 Button 控件，用于获取参与的观众短信息，添加一个 DataGridView 控件，用于显示观众的投票信息。

（3）主要程序代码。

```
private void btnResvice_Click(object sender, EventArgs e)
{
    //连接设备
    if (GMS.GSMModemInitNew(txtCOM.Text, txtBTL.Text, null, null, false, txtPower.Text) == false)
    {
        MessageBox.Show("连接失败!" + GMS.GSMModemGetErrorMsg(), "提示", MessageBoxButtons.OK);
        return;
    }
    //接收短信
    string content = GMS.GSMModemSMSReadAll(0);
    if (content == null)
    {
        this.getMessage();
        return;
    }
    content = content.Replace("||", "|"); // 替换||为|
```

```
        string[] str_sp = content.Split('|');// 进行分离
        int k = 0;
        string[,] str_Sp = new string[2, str_sp.Length / 2];
        for (int i = 0; i < str_sp.Length / 2; i++)
        {
            for (int j = 0; j < 2; j++)
            {
                str_Sp[j, i] = str_sp[k];
                if (k % 2 != 0)
                    this.InsertMessage("insert into RecivedBox (Mobile,Content,reciveTime)
values('" + Convert.ToString(str_Sp[0, i]) + "','" + Convert.ToString(str_Sp[1, i]) + "',
'" + DateTime.Now.ToString() + "') ");
                k++;
            }
        }
        this.getMessage();
    }
    private void getMessage()
    {
        OleDbConnection con = new OleDbConnection("Provider=Microsoft.Jet.OLEDB.
4.0;Data Source=" + "message.mdb" + ";Persist Security Info=False");
        OleDbDataAdapter dap = new OleDbDataAdapter("select mobile as 手机号码,
content as 短信息,reciveTime as 日期 from RecivedBox", con);
        DataSet ds = new DataSet();
        dap.Fill(ds);
        dataGridView1.DataSource = ds.Tables[0].DefaultView;
    }
    private void InsertMessage(string strSql)
```

图 13.19 利用条形码扫描器销售商品

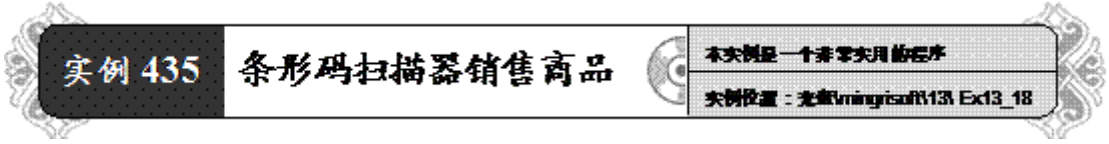
```
        {
            //将短信息内容添加到数据库中
            OleDbConnection con = new OleDbConnection("Provid
er=Microsoft.Jet.OLEDB.4.0;Data Source=" + "message.mdb"
+ ";Persist Security Info=False");
            con.Open();
            OleDbCommand cmd = new OleDbCommand(strSql, con);
            cmd.ExecuteNonQuery();
            con.Close();
        }
```

举一反三

根据本实例，读者可以实现以下功能。

- 手机短信息平台订购商品。
- 定时向手机发送天气预报

13.7 其 他 程 序



实例 435 条形码扫描器销售商品

实例说明

如今，许多超市都利用条形码销售商品。微机操作员利用扫描器在商品的条形码处进行扫描，商品的详细信息就会显示在屏幕中。本例实现了利用条形码销售商品的功能。效果如图 13.19 所示。

技术要点

当利用扫描器扫描条形码时，条形码数据会显示在当前获得焦点的窗口控件中。例如，如果当前编辑框获得焦点，那么条形码数据会显示在 TextBox 文本框中。然后会向 TextBox 文本框发送回车键按下时的消息。

在程序中只要触发 TextBox 文本框的 KeyDown 事件，判断当前按键是否是回车键，如果是，读取 TextBox 文本框中的条形码数据，并从数据表中根据条形码查询商品信息，将其显示在 DataGridView 列表中。

实现过程

（1）新建一个项目，命名为 Ex13_18，默认窗体为 Form1。

（2）在 Form1 窗体中，主要添加 TextBox 控件，用于接收条形码；添加一个 DataGridView 控件，用于显示扫描器扫描条形码的商品销售信息。

（3）主要程序代码。

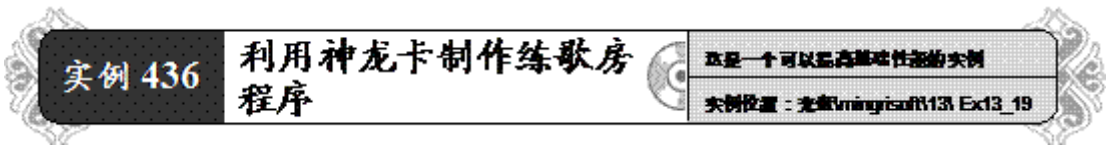
```
private void textBox1_KeyDown(object sender, KeyEventArgs e)
{
    if (e.KeyValue == 13)
    {
        OleDbConnection con = new OleDbConnection("Provider=Microsoft.Jet.OLEDB.4.0;Data Source=" + "price.mdb" + ";Persist Security Info=False");
        OleDbDataAdapter dap = new OleDbDataAdapter("select * from MerchandiseInfo where barcode='" + textBox1.Text + "'", con);
        DataSet ds = new DataSet();
        dap.Fill(ds);
        if (ds.Tables[0].Rows.Count == 0)
        {
            MessageBox.Show("该商品不存在!", "系统提示", MessageBoxButtons.OK, MessageBoxIcon.Information);
            return;
        }
```

```
            for (int i = 0; i < dataGridView1.RowCount; i++)
            {
                if (Convert.ToString(dataGridView1[0, i].Value) == "")
                {
                    dataGridView1[0, i].Value = ds.Tables[0].Rows[0][0].ToString();
                    dataGridView1[1, i].Value = ds.Tables[0].Rows[0][1].ToString();
                    dataGridView1[2, i].Value = ds.Tables[0].Rows[0][2].ToString();
                    dataGridView1[3, i].Value = ds.Tables[0].Rows[0][3].ToString();
                    return;
                }
            }
        }
    }
```

举一反三

根据本实例，读者可以开发以下程序。

- 超市条形码扫描系统。
- 公司工具条形码扫描系统。



实例 436　利用神龙卡制作练歌房程序

实例说明

在开发酒店、宾馆的点歌系统时，使用神龙 DVD 解压卡可以方便地进行媒体控制。神龙 DVD 解压卡（以下简称神龙卡）是一款专门针对中国大陆市场而开发出来的 DVD 硬解压卡，神龙卡与好莱坞卡的基本功能相近，两卡比较具有以下几个区别。

l 神龙卡可播放中国大陆全区码 DVD 碟，好莱坞卡可播放全球 1～6 区码影碟，可无数次解区码。

l 神龙卡支持 1～5M 码流播放，好莱坞卡可支持 1～15M 的视频流播放。

l 神龙卡支持全屏播放，好莱坞卡支持全屏及窗口（即可缩放窗口）播放。

l 神龙卡支持 Win 9x 下工作环境，好莱坞卡可支持 Win 9x 及 WinNT 下工作环境。

本例利用神龙卡实现了音乐的控制功能。运行程序，结果如图 13.20 所示。

图 13.20 利用神龙卡制作练歌房程序

技术要点

本程序主要通过一个第三方 NNSREALmagicCtrl.ocx 控件实现。在.NET 下使用第三方控件，首先，需要进行 Windows 注册，注册命令为“REgsvr32 路径\NNSREALmagicCtrl.ocx”；其次，将注册成功的控件添加到 Microsoft Visual Studio 2005 开发环境中，实现步骤为：选择菜单“工具”/“选择工具箱”，弹出“选择工具箱”窗口，在该窗口中选择“COM 组件”选项卡，在列表中选择注册的第三方控件，单击【确定】按钮即可，如图 13.21 所示。

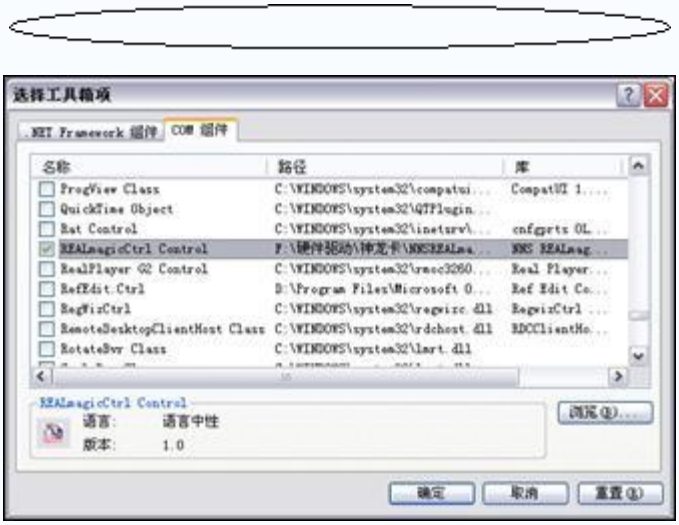


图 13.21　添加第三方控件

实现过程

（1）新建一个项目，命名为 Ex13_19，默认窗体为 Form1。

（2）在 Form1 窗体中，主要添加 DataGridView 控件，用于选择播放影音；添加其他控件及用途如图 13.20 所示。

（3）主要程序代码。

```
private void button1_Click(object sender, EventArgs e)
{
    if (strFileName == "")
    {
        MessageBox.Show("请在列表中选择播放文件！","系统提示");
        return;
    }
    axREALmagicCtrl1.Filename = strFileName;  //指定播放文件
    axREALmagicCtrl1.Play(); //播放
}
private void btnPause_Click(object sender, EventArgs e)
{
    axREALmagicCtrl1.Pause();  //暂停播放
```

```
    }
    private void btnStop_Click(object sender, EventArgs e)
    {
        axREALmagicCtrl1.Stop();     //停止播放
    }
    private void btnSpeed_Click(object sender, EventArgs e)
    {
        axREALmagicCtrl1.CurrentFrame = axREALmagicCtrl1.CurrentFrame + 125;  //快进
    }
    private void btnRecede_Click(object sender, EventArgs e)
    {
        axREALmagicCtrl1.CurrentFrame = axREALmagicCtrl1.CurrentFrame - 125;  //快退
    }
    private void rdoLeftTrack_Click(object sender, EventArgs e)
    {
        axREALmagicCtrl1.AudioChannel = NNSREALmagicCtrl.TAudChannel.acLEFT; //左声道
    }
    private void rdoRightTrack_Click(object sender, EventArgs e)
    {
        axREALmagicCtrl1.AudioChannel = NNSREALmagicCtrl.TAudChannel.acRIGHT;  //右声道
    }
    private void rdoStereo_Click(object sender, EventArgs e)
    {
```

```
        axREALmagicCtrl1.AudioChannel = NNSREALmagicCtrl.TAudChannel.acSTEREO;
  //立体声
      }
      private void tbVolume_Scroll(object sender, EventArgs e)
      {
         axREALmagicCtrl1.Volume = tbVolume.Value;        //音量
      }
      private void rdoTV_Click(object sender, EventArgs e)
      {
         axREALmagicCtrl1.DisplayDevice = NNSREALmagicCtrl.TDisDev.ddTV;  //TV 输
出模式
      }
      private void rdoVGA_Click(object sender, EventArgs e)
      {
         axREALmagicCtrl1.DisplayDevice = NNSREALmagicCtrl.TDisDev.ddVGA;  //VGA
输出模式
      }
      private void Form1_Load(object sender, EventArgs e)
      {
         if (!axREALmagicCtrl1.OpenDriver())  //打开驱动
         {
            MessageBox.Show("打开驱动失败！！", "系统提示");
            this.groupBox1.Enabled = false;
            this.groupBox2.Enabled = false;
            this.groupBox3.Enabled = false;
            this.groupBox4.Enabled = false;
            return;
         }
```

```
        OleDbConnection con = new OleDbConnection("Provider=Microsoft.Jet.OLEDB.
4.0;Data Source=" + "DataBase.mdb" + ";Persist Security Info=False");
        OleDbDataAdapter dap = new OleDbDataAdapter("select G_name as 影音名
称,G_YC as 原唱,G_wjlx as 文件格式 from g_music_name", con);
        DataSet ds = new DataSet();
        dap.Fill(ds); //显示影音文件的相关属性
        dataGridView1.DataSource = ds.Tables[0].DefaultView;
    }
    private void dataGridView1_CellClick(object sender, DataGridViewCellEventArgs
e)
    { //选择播放的影音
        strFileName = @"\vod\" + dataGridView1[0, e.RowIndex].Value.ToString() +
"." + dataGridView1[2, e.RowIndex].Value.ToString();
    }
```

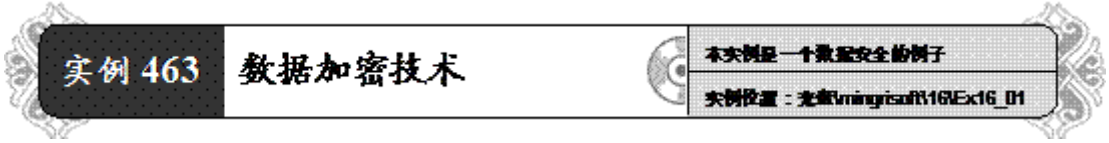
举一反三

根据本实例，读者可以开发以下程序。

- 练歌房卡拉 OK 系统。
- 单机卡拉 OK 系统。

16.1 数据加密与解密

在企业的计算机中，往往存有大量的机密文件，这些机密文件与企业的发展有紧密联系，如果这些机密文件保管不善，将会使企业遭受巨大的损失。本节通过几个典型实例详细介绍一下 C#中的加密与解密技术。



实例 463　数据加密技术

图 16.1 数据加密

实例说明

本实例实现对文件的机密数据进行加密的功能。运行程序，在文本框中输入要加密的数据，单击【加密】按钮，对数据进行加密，并将加密后的数据显示在“加密后的字符”文本框中。实例运行结果如图 16.1 所示。

技术要点

实现本实例功能主要用到了 System.Security.Cryptography 命名空间下的 MD5Crypto- ServiceProvider 类的 ComputeHash( )方法、System.Text 命名空间下的 ASCIIEncoding 类的 ASCII 属性、GetBytes( )方法和 GetString( )方法。下面分别进行介绍。

（1）System.Security.Cryptography 命名空间

System.Security.Cryptography 命名空间提供加密服务（包括安全的数据编码和解码）以及许多其他操作，例如散列法、随机数字生成和消息身份验证。

（2）MD5CryptoServiceProvider 类

此类使用加密服务提供程序（CSP）提供的实现，计算输入数据的 MD5 哈希值。

语法格式为：

```
public sealed class MD5CryptoServiceProvider : MD5
```

注意：MD5CryptoServiceProvider 类的哈希大小为 128 位。

（3）ComputeHash( )方法

此方法计算指定字节数组的哈希值。

语法格式为：

```
public byte[] ComputeHash (byte[] buffer)
```

参数说明如下。

- buffer：要计算其哈希代码的字节数组。
- 返回值：计算所得的哈希代码。

（4）System.Text 命名空间

表示 ASCII、Unicode、UTF-7 和 UTF-8 字符编码的类；是用于将字符块转换为字节块和将字节块转换为字符块的抽象基类。

（5）ASCIIEncoding 类

此类表示 Unicode 字符的 ASCII 字符编码。

语法格式为：

```
public class ASCIIEncoding : Encoding
```

（6）ASCII 属性

此属性获取 ASCII（7 位）字符集的编码。

语法格式为：

```
public static Encoding ASCII { get; }
```

l 属性值：ASCII（7 位）字符集的 Encoding。

（7）GetBytes( )方法

此方法将指定的 String 中的所有字符编码为一个字节序列。

语法格式为：

```
public virtual byte[] GetBytes (string s)
```

参数说明如下。

l s：包含要编码的字符的 String。

（8）GetString( )方法

此方法将指定字节数组中的所有字节解码为一个字符串。

语法格式为：

```
public virtual string GetString (byte[] bytes)
```

参数说明如下。

l bytes：包含要解码的字节序列的字节数组。

l 返回值：包含指定字节序列解码结果的 String。

 注意：使用 MD5CryptoServiceProvider 类必须引用 System.Security.Cryptography 命名空间。

实现过程

图 16.2 文本文件加密与解密

（1）新建一个 Windows 应用程序，将其命名为 Ex16_01，默认窗体为 Form1。

（2）在 Form1 窗体中，主要添加两个 TextBox 控件，用来输入和显示字符串；添加一个 Button 控件，用来执行加密操作。

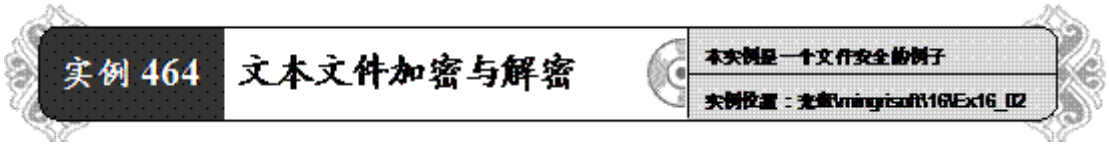
（3）主要程序代码。

```
private void button1_Click(object sender, EventArgs e)
{
    if (textBox1.Text == "")
    { MessageBox.Show("请输入加密数据"); return; }
    MD5CryptoServiceProvider M5 = new MD5CryptoServiceProvider();
    textBox2.Text = ASCIIEncoding.ASCII.GetString(M5.ComputeHash(ASCIIEncodi
ng.ASCII.GetBytes(textBox1.Text)));
}
```

举一反三

根据本实例，读者可以实现以下功能。

- 对机密的文件夹进行加密与解密。
- 在数据传输中使用，以便保证传输信息不被泄露。



实例 464　文本文件加密与解密

实例说明

本实例实现对计算机中的一些非常机密的文本文件进行加密与解密操作。运行程序，单击【选择文件】按钮，选择要进行加密或解密的文本文件（.txt 格式的文件），单击【加密】或【解密】按钮，即可完成对文本文件的加密或解密操作。实例运行结果如图 16.2 所示。

技术要点

实现本实例功能主要用到了System.Security.Cryptography命名空间下的RijndaelManaged类的CreateDecryptor( )方法、CreateEncryptor( )方法、CryptoStream类的Write( )方法、FlushFinalBlock( )方法、Close( )方法、System.IO命名空间下的FileStream类、StreamReader类的ReadToEnd( )方法、StreamWriter类的Write( )方法、Flush( )方法和BinaryReader类的ReadBytes( )方法。System.Security.Cryptography命名空间在第16章实例463中已经做过详细介绍，这里不再详细描述，下面对本实例中用到的其他知识进行详细介绍。

（1）RijndaelManaged类

此类表示Rijndael对称加密算法的所有实现必须从其继承的基类中获得。

语法格式为：

```
public abstract class Rijndael  SymmetricAlgorithm
```


注意：此算法支持128、192或256位的密钥长度。

（2）CreateDecryptor( )方法

此方法使用指定的Key和初始化向量（IV）创建对称的Rijndael解密器对象。

语法格式为：

```
public override IcryptoTransform  CreateDecryptor (byte[] rgbKey,byte[] rgbIV)
```

参数说明如下。

- rgbKey：用于对称算法的机密密钥。
- rgbIV：用于对称算法的IV。
- 返回值：对称的Rijndael解密器对象。

（3）CreateEncryptor( )方法

此方法使用指定的Key和初始化向量（IV）创建对称的Rijndael加密器对象。

语法格式为：

public override ICryptoTransform CreateEncryptor (byte[] rgbKey,byte[] rgbIV)

参数说明如下。

l rgbKe：用于对称算法的机密密钥。

l rgbIV：用于对称算法的 IV。

l 返回值：对称的 Rijndael 加密器对象。

（4）CryptoStream 类

此类定义将数据流链接到加密转换的流。

语法格式为：

public CryptoStream (Stream stream,ICryptoTransform transform,CryptoStreamMode mode)

参数说明如下。

l stream：对其执行加密转换的流。

l Transform：要对流执行的加密转换。

l Mode：CryptoStreamMode 值之一。CryptoStreamMode 值及明说如表 16.1 所示。

表 16.1 CryptoStreamMode 值及说明

值	说　明
Read	对加密流的读访问
Write	对加密流的写访问

（5）CryptoStream 类的 Write( )方法

此方法将一个字节序列写入当前 CryptoStream 类中，并从当前位置写入指定的字节数。

语法格式为：

public override void Write (byte[] buffer,int offset,int count)

参数说明如下。

l buffer：字节数组。此方法将 count 个字节从 buffer 复制到当前流。

l offset：buffer 中的字节偏移量，从此偏移量开始将字节复制到当前流。

l count：要写入当前流的字节数。

（6）FlushFinalBlock( )方法

此方法用缓冲区的当前状态更新基础数据源或储存库，随后清除缓冲区。

语法格式为：

```
public void FlushFinalBlock ()
```

（7）Close( )方法

关闭当前流并释放与之关联的所有资源（如套接字和文件句柄）。

语法格式为：

```
public virtual void Close ()
```

（8）System.IO 命名空间

System.IO 命名空间包含允许读写文件和数据流的类型以及提供基本文件和目录支持的类型。

（9）FileStream 类

此类公开以文件为主的 Stream，既支持同步读写操作，也支持异步读写操作。

语法格式为：

```
public FileStream (string path,FileModemode,FileAccessaccess)
```

参数说明如下。

l path：当前 FileStream 类对象封装文件的相对路径或绝对路径。

l Mode：FileMode 常数，确定如何打开或创建文件。FileMode 常数的值及说明如表 16.2 所示。

l Access：FileAccess 常数，确定 FileStream 对象访问文件的方式。这将获取 FileStream 对象的 CanRead 和 CanWrite 属性。如果 path 指定磁盘文件，则 CanSeek 为 True。FileAccess 常数的值及说明如表 16.3 所示。

表 16.2 FileMode 常数的值及说明

常 数 值	说 明
Append	打开现有文件并查找到文件尾，或创建新文件。FileMode.Append 只能同 FileAccess.Write 一起使用。任何读尝试都将失败并引发 ArgumentException
Create	指定操作系统应创建新文件。如果文件已存在，它将被改写。这要求 FileIOPermissionAccess.Write 和 System.IO.FileMode.Create 等效于这样的请求：如果文件不存在，则使用 CreateNew；否则使用 Truncate
CreateNew	指定操作系统应创建新文件。此操作需要 FileIOPermissionAccess.Write。如果文

	件已存在，则将引发 IOException
Open	指定操作系统应打开现有文件。打开文件的能力取决于 FileAccess 所指定的值。如果该文件不存在，则引发 System.IO.FileNotFoundException
OpenOrCreate	指定操作系统应打开文件（如果文件存在）；否则，应创建新文件。如果用 FileAccess.Read 打开文件，则需要 FileIOPermissionAccess.Read。如果文件访问为 FileAccess.Write 或 FileAccess.ReadWrite，则需要 FileIOPermissionAccess.Write。如果文件访问为 FileAccess.Append，则需要 FileIOPermissionAccess.Append
Truncate	指定操作系统应打开现有文件。文件一旦打开，就将被截断为零字节大小。此操作需要 FileIOPermissionAccessWrite。试图从使用 Truncate 打开的文件中进行读取将导致异常

表 16.3　　FileAccess 常数的值及说明

常　数　值	说　　明
Read	对文件的读访问。可从文件中读取数据。同 Write 组合即构成读写访问权
ReadWrite	对文件的读访问和写访问。可从文件读取数据和将数据写入文件
Write	文件的写访问。可将数据写入文件。同 Read 组合即构成读/写访问权

（10）StreamReader 类

此类实现一个 TextReader，使其以一种特定的编码从字节流中读取字符。

语法格式为：

```
public StreamReader (Stream stream)
```

参数说明如下。

l　stream：要读取的流。

（11）ReadToEnd(　)方法

此方法从流的当前位置到末尾读取流。

```
public override string ReadToEnd ()
```

l　返回值：字符串形式的流的其余部分（从当前位置到末尾）。如果当前位置位于流的末尾，则返回空字符串（“”）。

（12）StreamWriter 类

此类实现一个 TextWriter，使其以一种特定的编码向流中写入字符。

语法格式为：

```
public StreamWriter (Stream stream)
```

参数说明如下。

l　stream：要写入的流。

（13）StreamWriter 类的 Write(　)方法

此方法将字符写入流。

语法格式为：

```
public override void Write (char value)
```

参数说明如下。

l　value：要写入文本流中的字符。

（14）Flush( )方法

此方法清理当前编写器的所有缓冲区，并使所有缓冲数据写入基础流。

语法格式为：

```
public override void Flush ()
```

（15）BinaryReader 类

此类用特定的编码将基元数据类型读作二进制值。

语法格式为：

```
public BinaryReader (Stream input)
```

参数说明如下。

l Input：流。

（16）ReadBytes( )方法

此方法从当前流中将 count 个字节读入字节数组，并使当前位置提升 count 个字节。

语法格式为：

```
public virtual byte[] ReadBytes (int count)
```

参数说明如下。

l count：要读取的字节数。

l 返回值：包含从基础流中读取的数据的字节数组。如果到达了流的末尾，则该字节数组可能小于所请求的字节数。

实现过程

（1）新建一个 Windows 应用程序，将其命名为 Ex16_02，默认窗体为 Form1。

（2）在 Form1 窗体中，主要添加一个 TextBox 控件，用来显示文件路径；添加一个 OpenFileDialog 控件，用来选择要加密或解密的文件；添加 3 个 Button 控件，用来执行加密、解密和选择文件操作。

（3）主要程序代码。

加密文本文件的实现代码如下：

```
private void button5_Click(object sender, EventArgs e)
```

```
    {
        if (textBox1.Text == "")
        { MessageBox.Show("请选择要加密的文件"); }
        else
        {
          try{
            string strPath = textBox1.Text;//加密文件的路径
            int intLent=strPath.LastIndexOf("\\")+1;
            int intLong = strPath.Length;
        //要加密的文件名称
            string strName = strPath.Substring(intLent,intLong-intLent);
            int intTxt = strName.LastIndexOf(".");
            int intTextLeng = strName.Length;
           //取出文件的扩展名
            string strTxt = strName.Substring(intTxt,intTextLeng-intTxt);
            strName = strName.Substring(0,intTxt);
            //加密后的文件名及路径
            string strOutName = strPath.Substring(0, strPath.LastIndexOf("\\") + 1) +
 strName + "Out" + strTxt;
//加密文件密钥
            byte[] key = { 24, 55, 102, 24, 98, 26, 67, 29, 84, 19, 37, 118, 104, 8
5, 121, 27, 93, 86, 24, 55, 102, 24, 98, 26, 67, 29, 9, 2, 49, 69, 73, 92 };
            byte[] IV ={ 22, 56, 82, 77, 84, 31, 74, 24, 55, 102, 24, 98, 26, 67, 2
9, 99 };
            RijndaelManaged myRijndael = new RijndaelManaged();
            FileStream fsOut = File.Open(strOutName, FileMode.Create, FileAccess.Writ
e);
            FileStream fsIn = File.Open(strPath, FileMode.Open, FileAccess.Read);
```

```
            //写入加密文本文件
            CryptoStream csDecrypt = new CryptoStream(fsOut, myRijndael.CreateEnc
ryptor(key, IV), CryptoStreamMode.Write);
            //读加密文本
            BinaryReader br = new BinaryReader(fsIn);
            csDecrypt.Write(br.ReadBytes((int)fsIn.Length), 0, (int)fsIn.Length);
            csDecrypt.FlushFinalBlock();
            csDecrypt.Close();
            fsIn.Close();
            fsOut.Close();
            if (MessageBox.Show(strOutName, "提示:加密成功!加密后的文件名及路径为:
\n"+"是否删除源文件", MessageBoxButtons.YesNo) == DialogResult.Yes)
            {
               File.Delete(strPath);
               textBox1.Text = "";
            }else
            { textBox1.Text = ""; }
         }
         catch (Exception ee)
         {
            MessageBox.Show(ee.Message);
         }
         }
      }
```

解密文本文件的实现代码如下：

```
  private void button4_Click(object sender, EventArgs e)
      {
         if (textBox1.Text == "")
```

```
        {
            MessageBox.Show("请选择要解密的文件路径");
        }
        else
        {
            string strPath = textBox1.Text;//加密文件的路径
            int intLent = strPath.LastIndexOf("\\") + 1;
            int intLong = strPath.Length;
            //要加密的文件名称
            string strName = strPath.Substring(intLent, intLong - intLent);
            int intTxt = strName.LastIndexOf(".");
            int intTextLeng = strName.Length;
            strName = strName.Substring(0, intTxt);
            if (strName.LastIndexOf("Out") != -1)
            {
                strName = strName.Substring(0, strName.LastIndexOf("Out"));
            }
            else
            {
                strName = strName + "In";
            }
            //加密后的文件名及路径
            string strInName = strPath.Substring(0, strPath.LastIndexOf("\\") + 1) +
strName + ".txt"; //解密文件密钥
            byte[] key = { 24, 55, 102, 24, 98, 26, 67, 29, 84, 19, 37, 118, 104, 8
5, 121, 27, 93, 86, 24, 55, 102, 24, 98, 26, 67, 29, 9, 2, 49, 69, 73, 92 };
            byte[] IV ={ 22, 56, 82, 77, 84, 31, 74, 24, 55, 102, 24, 98, 26, 67, 2
9, 99 };
```

```
            RijndaelManaged myRijndael = new RijndaelManaged();
            FileStream fsOut = File.Open(strPath, FileMode.Open, FileAccess.Read);
            CryptoStream csDecrypt = new CryptoStream(fsOut, myRijndael.CreateDec
ryptor(key, IV), CryptoStreamMode.Read);
            StreamReader sr = new StreamReader(csDecrypt);//把文件读出来
            StreamWriter sw = new StreamWriter(strInName);//解密后文件写入一个新
的文件
            sw.Write(sr.ReadToEnd());
            sw.Flush();
            sw.Close();
            sr.Close();
            fsOut.Close();
            if (MessageBox.Show(strInName, "提示:解密成功!解密后的文件名及路径为:"
 + "是否删除源文件", MessageBoxButtons.YesNo) == DialogResult.Yes)
            {
               File.Delete(strPath);
               textBox1.Text = "";
            }
            else
            {
               textBox1.Text = "";
            }
         }
      }
```

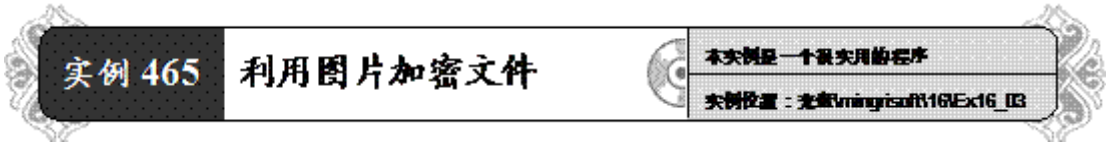
举一反三

根据本实例，读者可以实现以下功能。

- 对重要公文进行加密。

图 16.3　利用图片加密文件

- 对网络中传输的文件进行加密与解密。



实例 465　利用图片加密文件

实例说明

本实例中，利用图片生成密钥，然后对文本文件进行加密和解密操作。运行程序，单击【打开图片】按钮，选择密钥图片，然后单击【打开文本文件】按钮，选择要加密或解密的文件，单击【加密】或【解密】按钮完成文本文件的加密或解密操作。解密时的密钥图片要与加密时的密钥图片相同，否则解密不能成功。实例运行结果如图 16.3 所示。

技术要点

实现本实例功能主要用到了 System.Security.Cryptography 命名空间下的 RC2CryptoServiceProvider 类的 CreateDecryptor( )方法、CreateEncryptor( )方法、CryptoStream 类的 Write( )方法、FlushFinalBlock( )方法、Close( )方法、System.IO 命名空间下的 FileStream 类、BinaryReader 类的 ReadBytes( )方法、BinaryWriter 类的 Write( )方法、File 类的 Delete( )方法和 Copy( )方法。以上大部分知识在第 16 章实例 464 中已经做过详细介绍，这里不再详细讲解。下面主要对 RC2CryptoServiceProvider 类、BinaryWriter 类的 Write( )方法、File 类的 Delete( )方法和 Copy( )方法进行介绍。

（1）RC2CryptoServiceProvider 类

此类定义访问 RC2 算法的加密服务提供程序（CSP）实现的包装对象。

（2）BinaryWriter 类

此类以二进制形式将基元类型写入流，并支持用特定的编码写入字符串。

语法格式为：

```
public BinaryWriter (Streamoutput)
```

参数说明如下。

l output：输出流。

（3）BinaryWriter 类的 Write( )方法

此方法将一个无符号字节写入当前流，并将流的位置提升 1 个字节。

语法格式为：

```
public virtual void Write (byte value)
```

参数说明如下。

l value：要写入的无符号字节。

（4）File 类

此类提供用于创建、复制、删除、移动和打开文件的静态方法，并协助创建 FileStream 对象。

（5）Delete( )方法

此方法删除指定的文件。如果指定的文件不存在，则引发异常。

语法格式为：

```
public static void Delete (string path)
```

参数说明如下。

l path：要删除的文件的名称。

（6）Copy( )方法

此方法将现有文件复制到新文件，不允许改写同名的文件。

语法格式为：

```
public static void Copy (string sourceFileName,string destFileName)
```

参数说明如下。

l sourceFileName：要复制的文件。

l destFileName：目标文件的名称，不能是一个目录或现有文件。

实现过程

（1）新建一个 Windows 应用程序，将其命名为 Ex16_03，默认窗体为 Form1。

（2）在 Form1 窗体中，主要添加一个 TextBox 控件，用来显示加密或解密文件的路径；添加一个 OpenFileDialog 控件，用来选择要加密或解密的文件和打开密钥的图片；添加 4 个 Button 控件，用来执行加密、解密、打开文件和打开图片操作；添加一个 PictureBox 控件，用于显示密钥图片。

（3）主要程序代码。

利用图片加密文本文件的实现代码如下：

```
private void button3_Click(object sender, EventArgs e)
    {
       try
       {
          if (pictureBox1.ImageLocation==null)
          { MessageBox.Show("请选择一幅图片用于加密"); return; }
          if (textBox1.Text == "")
          { MessageBox.Show("请选择加密文件路径"); return; }
          //图片流
          FileStream fsPic = new FileStream(pictureBox1.ImageLocation, FileMode.O
pen, FileAccess.Read);
          //加密文件流
          FileStream fsText = new FileStream(textBox1.Text, FileMode.Open, FileAcc
ess.Read);
          //初始化 Key IV
          byte[] bykey = new byte[16];
          byte[] byIv = new byte[8];
          fsPic.Read(bykey, 0, 16);
          fsPic.Read(byIv, 0, 8);
```

```
            //临时加密文件
            string strPath = textBox1.Text;//加密文件的路径
            int intLent = strPath.LastIndexOf("\\") + 1;
            int intLong = strPath.Length;
            string strName = strPath.Substring(intLent, intLong - intLent);//要加密的文
件名称
            string strLinPath = "C:\\" + strName;//临时加密文件路径
            FileStream fsOut = File.Open(strLinPath, FileMode.Create, FileAccess.Writ
e);
            //开始加密
          RC2CryptoServiceProvider desc = new RC2CryptoServiceProvider();//desc 进行
加密
            BinaryReader br = new BinaryReader(fsText);//从要加密的文件中读出文件
内容
            CryptoStream cs = new CryptoStream(fsOut, desc.CreateEncryptor(bykey,
byIv), CryptoStreamMode.Write);//写入临时加密文件
            cs.Write(br.ReadBytes((int)fsText.Length), 0, (int)fsText.Length);//写入加密
流
            cs.FlushFinalBlock();
            cs.Flush();
            cs.Close();
            fsPic.Close();
            fsText.Close();
            fsOut.Close();
            File.Delete(textBox1.Text.TrimEnd());//删除原文件
            File.Copy(strLinPath, textBox1.Text);//复制加密文件
            File.Delete(strLinPath);//删除临时文件
            MessageBox.Show("加密成功");
```

```
            pictureBox1.ImageLocation = null;
            textBox1.Text = "";
        }
        catch (Exception ee)
        {
            MessageBox.Show(ee.Message);
        }
    }
```

利用图片解密文本文件的实现代码如下：

```
 private void button4_Click(object sender, EventArgs e)
    {
        try
        {
            //图片流
     FileStream fsPic = new FileStream(pictureBox1.ImageLocation, FileMode.Open,
FileAccess.Read);
            //解密文件流
        FileStream fsOut = File.Open(textBox1.Text, FileMode.Open, FileAccess.Rea
d);
            //初始化 Key IV
            byte[] bykey = new byte[16];
            byte[] byIv = new byte[8];
            fsPic.Read(bykey, 0, 16);
            fsPic.Read(byIv, 0, 8);
            //临时解密文件
            string strPath = textBox1.Text;//加密文件的路径
            int intLent = strPath.LastIndexOf("\\") + 1;
            int intLong = strPath.Length;
```

```
    string strName = strPath.Substring(intLent, intLong - intLent);//要加密的文件名称
            string strLinPath = "C:\\" + strName;//临时解密文件路径
           FileStream fs = new FileStream(strLinPath, FileMode.Create, FileAccess.Writ
e);
            //开始解密
        RC2CryptoServiceProvider desc = new RC2CryptoServiceProvider();//desc 进行
解密
         CryptoStream csDecrypt = new CryptoStream(fsOut, desc.CreateDecryptor(byk
ey, byIv), CryptoStreamMode.Read);//读出加密文件
            BinaryReader sr = new BinaryReader(csDecrypt);//从要加密流中读出文件内
容
            BinaryWriter sw = new BinaryWriter(fs);//写入解密流
            sw.Write(sr.ReadBytes(Convert.ToInt32(fsOut.Length)));//
            sw.Flush();
            sw.Close();
            sr.Close();
            fs.Close();
            fsOut.Close();
            fsPic.Close();
            csDecrypt.Flush();
            File.Delete(textBox1.Text.TrimEnd());//删除原文件
            File.Copy(strLinPath, textBox1.Text);//复制加密文件
            File.Delete(strLinPath);//删除临时文件
            MessageBox.Show("解密成功");
            pictureBox1.ImageLocation = null;
            textBox1.Text = "";
        }
        catch (Exception ee)
```

图 16.4 修复 Access 数据库

```
            {
                MessageBox.Show(ee.Message);
            }
        }
```

举一反三

根据本实例，读者可以实现以下功能。

- 利用各种图片加密文件。
- 使用图片批量解密文件。

16.2 Access 数据库安全

由于 Access 数据库操作简单，使用方便，所以一些中小型的应用软件经常采用 Access 数据库。为了保证数据库的安全，经常要加密或锁定数据库，如果数据库由于某种原因遭到破坏，就需要对数据库进行修复。下面的几个实例分别实现了加密、锁定和修复 Access 数据库的功能。

实例 466 如何编程修复 Access 数据库

实例说明

Access 数据库操作简单，使用方便，是中小型企业经常采用的数据库，但 Access 数据库容易遭到破坏，并随着时间的增加，数据库文件会变得非常大，该如何解决这些问题呢？本实例通过压缩数据库的方法重新组织修复数据库，减少了数据库占用的空间。运行程序，单击【打开】按钮，找到要修复的数据库，单击【开始修复】按钮，即可完成修复数据库操作。实例运行结果如图 16.4 所示。

技术要点

实现本实例功能主要用到了 JRO 命名空间下 JetEngineClass 对象的 CompactDatabase( )方法、System.IO 命名空间下 File 类的 Copy( )方法和 Delete( )方法。下面分别进行介绍。

（1）CompactDatabase( )方法

CompactDatabase( )方法压缩并回收本地数据库中的浪费空间。

语法格式为：

```
CompactDatabase(strng SourceConnection, string DestConnection)
```

参数说明如下。

- SourceConnection：字符串值，指定与要压缩的源数据库的连接。
- DestConnection：字符串值，指定与要通过压缩创建的目标数据库的连接。

 注意：必须引用 C:\Program Files\Common Files\System\ado\msjro.dll，该 DLL 包含 JRO 命名空间。

（2）Copy( )方法

此方法将现有文件复制到新文件，不允许改写同名的文件。

语法格式为：

```
public static void Copy (string sourceFileName,string destFileName)
```

参数说明如下。

- sourceFileName：要复制的文件。
- destFileName：目标文件的名称，不能是一个目录或现有文件。

（3）Delete( )方法

此方法删除指定的文件。

语法格式为：

```
public static void Delete (string path)
```

参数说明如下。

- path：要删除的文件的名称。

实现过程

（1）新建一个 Windows 应用程序，将其命名为 Ex16_04，默认窗体为 Form1。

（2）在 Form1 窗体中，主要添加一个 TextBox 控件，用来显示修复数据库文件的路径；添加一个 OpenFileDialog 控件，用来选择要修复的数据库文件；添加 3 个 Button 控件，用来执行修复、退出和打开数据库文件操作。

（3）主要程序代码。

开始压缩数据库的实现代码如下：

```
string strPathMdb = null;//数据库路径
private void button2_Click(object sender, EventArgs e)
{
    if (!File.Exists(strPathMdb)) //检查数据库是否已存在
    {
        MessageBox.Show("目标数据库不存在,无法压缩","操作提示");
        return;
    }
    //声明临时数据库的名称
    string temp = DateTime.Now.Year.ToString();
    temp += DateTime.Now.Month.ToString();
    temp += DateTime.Now.Day.ToString();
    temp += DateTime.Now.Hour.ToString();
    temp += DateTime.Now.Minute.ToString();
    temp += DateTime.Now.Second.ToString() + ".bak";
    temp = strPathMdb.Substring(0, strPathMdb.LastIndexOf("\\") + 1) + temp;
    //定义临时数据库的连接字符串
    string temp2 = "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=" + temp;
    //定义目标数据库的连接字符串
    string strPathMdb2 = "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=" + str
PathMdb;
    //创建一个 JetEngineClass 对象的实例
    JRO.JetEngineClass jt = new JRO.JetEngineClass();
```

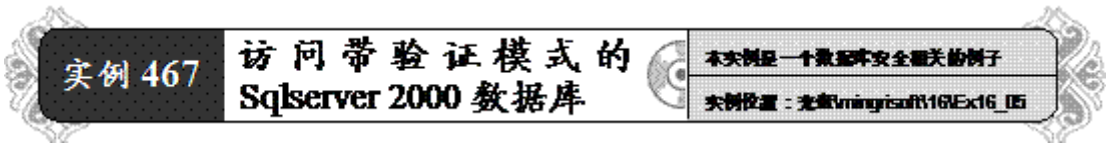
图 16.5　访问带验证模式的 Sqlserver 2000 数据库

```
        //使用 JetEngineClass 对象的 CompactDatabase 方法压缩修复数据库
        jt.CompactDatabase(strPathMdb2, temp2);
        //复制临时数据库到目标数据库(覆盖)
        File.Copy(temp, strPathMdb, true);
        //最后删除临时数据库
        File.Delete(temp);
        MessageBox.Show("修复完成");
    }
```

举一反三

根据本实例，读者可以实现以下功能。

- 定时数据库压缩。
- 定时数据库备份。



实例 467　访问带验证模式的 Sqlserver 2000 数据库

实例说明

本实例实现了访问带验证模式的 SQL Server 2000 数据库的功能。用户登录数据库时，必须输入用户名和密码。运行程序，输入计算机的名称或地址、访问数据库的用户名、密码和数据库名称，单击【登录】按钮，即可登录数据库。实例运行结果如图 16.5 所示。

技术要点

实现本实例功能主要用到了 ADO.NET 的 SqlConnection 对象的 Open(　)方法、Close(　)方法和 ConnectionState 枚举。下面分别进行介绍。

（1）SqlConnection 对象

此对象表示 SQL Server 数据库的一个打开的连接。

语法格式为：

```
public SqlConnection (string connectionString)
```

参数说明如下。

l connectionString：用于打开 SQL Server 数据库的连接。

（2）Open( )方法

此方法使用 ConnectionString 所指定的属性设置打开的数据库连接。

语法格式为：

```
public override void Open ()
```

（3）Close( )方法

此方法关闭与数据库的连接。这是关闭任何打开连接的首选方法。

语法格式为：

```
public override void Close ()
```

（4）ConnectionState 枚举

此枚举描述与数据源连接的当前状态。

语法格式为：

```
public enum ConnectionState
```

ConnectionState 枚举值及说明如表 16.4 所示。

表 16.4 ConnectionState 枚举值及说明

枚 举 值	说 明
Broken	与数据源的连接中断。只有在连接打开之后才可能发生这种情况。可以关闭处于这种状态的连接，然后重新打开
Closed	连接处于关闭状态
Connecting	连接对象正在与数据源连接
Executing	连接对象正在执行命令
Fetching	连接对象正在检索数据
Open	连接处于打开状态

 注意：使用 SqlConnection 对象必须引用 System.Data.SqlClient 命名空间。

实现过程

（1）新建一个 Windows 应用程序，将其命名为 Ex16_05，默认窗体为 Form1。

（2）在 Form1 窗体中，主要添加 4 个 TextBox 控件，用于输入登录信息；添加 3 个 Button 控件，用来执行登录、断开连接和退出操作。

（3）主要程序代码。

登录数据库的实现代码如下：

```
public SqlConnection con = null;//实义数据库连接对象
    private void button1_Click(object sender, EventArgs e)
    {
        if (textBox1.Text == "")
        {
            MessageBox.Show(textBox1.Tag.ToString()+"不能为空","提示");
            textBox1.Focus();
            return;
        }
        if (textBox2.Text == "")
        {
            MessageBox.Show(textBox2.Tag.ToString() + "不能为空", "提示");
            textBox2.Focus();
            return;
        }
        if (textBox4.Text == "")
        {
            MessageBox.Show(textBox4.Tag.ToString() + "不能为空", "提示");
            textBox4.Focus();
            return;
        }
```

```
        try
        {
            string strcon = "server='" + textBox1.Text.Trim() + "';uid='" + textBox2.
Text.Trim() + "';pwd='" + textBox3.Text + "';database='" + textBox4.Text.Trim() +
"'";
            con = new SqlConnection(strcon);//实例 SqlConnect 对象
            con.Open();
            MessageBox.Show("登录成功");
        }
        catch (Exception ee)
        {
            MessageBox.Show(ee.Message);
        }
    }
```

断开连接退出数据库登录的实现代码如下：

```
    private void button2_Click(object sender, EventArgs e)
    {
        if (con.State == ConnectionState.Open)
        {
            con.Close();
            MessageBox.Show("连接已断开");
        }
        else
        {
            MessageBox.Show("还没有连接数据库");
        }
    }
```

图 16.6 软件注册

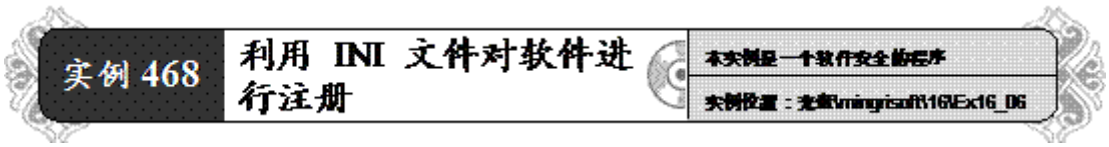
举一反三

根据本实例，读者可以实现以下功能。

- 对数据库添加用户。
- 对数据库添加用户权限。

16.3 软件注册与加密

为了使开发的软件能被更广泛地使用，开发者希望更多的用户能试用软件，而另一方面，又不想让用户长时间免费使用未经授权的软件，这就需要设计软件注册程序。下面通过几个典型实例介绍保护软件安全的方法。



实例 468 利用 INI 文件对软件进行注册

实例说明

本实例实现使用 INI 文件对软件的用户信息进行注册的功能。运行程序，输入登录名称、登录口令和注册码，单击【注册】按钮进行注册，如果注册成功，则给出提示；如果信息已注册，系统给出提示信息。实例运行结果如图 16.6 所示。

技术要点

实现本实例功能主要用到 API 函数 WritePrivateProfileString 和 GetPrivateProfileString 函数。下面分别进行介绍。

（1）WritePrivateProfileString 函数

此函数实现对 INI 文件的写操作。

函数声明如下。

```
[ DllImport ( "kernel32" ) ]
  private static extern long WritePrivateProfileString ( string section ,string key , string val , string filePath ) ;
```

参数说明如下。

l section：INI 文件中的段落。

l key：INI 文件中的关键字。

l val：INI 文件中关键字的数值。

l filePath：INI 文件完整的路径和名称。

（2）GetPrivateProfileString 函数

此函数实现对 INI 文件的读操作。

函数声明如下。

```
[DllImport("kernel32")]
    private static extern int GetPrivateProfileString(string section, string key, string def, StringBuilder retVal, int size, string filePath);
```

参数说明如表 16.5 所示。

表 16.5　　参数说明

参　数　值	说　　明
section	INI 文件中的段落名称
key	INI 文件中的关键字
def	无法读取时候的缺省数值
retVal	读取数值
size	数值的大小
filePath	INI 文件的完整路径和名称

 注意：使用 API 函数必须引用 System.Runtime.InteropServices 命名空间。

实现过程

（1）新建一个 Windows 应用程序，将其命名为 Ex16_07，默认窗体为 Form1。

（2）在 Form1 窗体中，主要添加 3 个 TextBox 控件，用于输入注册信息；添加两个 Button 控件，用来执行注册和退出操作。

（3）主要程序代码。

注册用户信息的实现代码如下：

```
private void Form1_Load(object sender, EventArgs e)
    {
```

```
        FileStream c = new FileStream("C:\\desck.ini",FileMode.OpenOrCreate,FileAcc
ess.Write);
    }
    [ DllImport ( "kernel32" ) ]
    private static extern long WritePrivateProfileString ( string section ,string key ,
string val , string filePath ) ;
    [DllImport("kernel32")]
    private static extern int GetPrivateProfileString(string section, string key, string
def, StringBuilder retVal, int size, string filePath);
    private void button1_Click(object sender, EventArgs e)
    {
        StringBuilder temp = new StringBuilder(200);
        string FileName = "C:\\desck.ini";//NI 文件的完整的路径和名称。
        foreach (object ct in Controls)
        {
            if (ct.GetType().ToString() == "System.Windows.Forms.TextBox")
            {
                TextBox tx = (TextBox)ct;
                if (tx.Text == "")
                {
                    MessageBox.Show(tx.Tag.ToString()+"不能为空");
                }
            }
        }
        string section = textBox3.Text;//INI 文件中的段落
        string key = textBox1.Text;//INI 文件中的关键字
        string keyValue = textBox2.Text;//INI 文件中的关键字
```

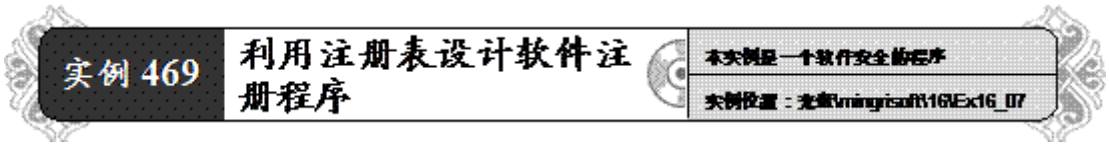
图 16.7 动态变化的软件注册程序

```
            int i = GetPrivateProfileString(section, key, "无法读取对应数值！",
temp, 200, FileName);//判断是否注册过
            if (temp.ToString() == "无法读取对应数值！")
            {
                WritePrivateProfileString(section, key, keyValue, FileName);
                MessageBox.Show("注册成功写入 INI 文件！", "信息");
            }
            else
            {
                MessageBox.Show("此信息已注册过了");
            }
        }
```

举一反三

根据本实例，读者可以实现以下功能。

- 对 INI 文件加密保存注册信息。
- 对组合 INI 文件加密保存注册信息。



实例 469　利用注册表设计软件注册程序

实例说明

大多数应用软件会将用户输入的注册信息写进注册表中，程序运行过程中，可以将这些信息从注册表中读出。本实例主要实现在程序中对注册表进行操作的功能，运行程序，单击【注册】按钮，会将用户输入的信息写入注册表中。实例运行结果如图 16.7 所示。

技术要点

实现本实例功能主要用到了Microsoft.Win32命名空间下的Registry类的CurrentUser属性、RegistryKey类的OpenSubKey( )方法、GetSubKeyNames( )方法、SetValue( )方法和CreateSubKey( )方法。下面分别进行介绍。

（1）Microsoft.Win32命名空间

Microsoft.Win32命名空间提供两种类型的类：处理由操作系统引发的事件的类和操作系统注册表的类。

（2）RegistryKey类

此类表示Windows注册表中的项级节点，此类是注册表封装。

语法格式为：

```
public sealed class RegistryKey : MarshalByRefObject, IDisposable
```


注意：要获取RegistryKey实例，需要使用Registry类的静态成员之一。

（3）Registry类

此类提供表示Windows注册表中的根项的RegistryKey对象，并提供访问项/值对的static( )方法。

语法格式为：

```
public static class Registry
```

（4）CurrentUser属性

此属性包含有关当前用户首选项的信息，该字段读取Windows 注册表中的HKEY_ CURRENT_USER注册表项。

语法格式为：

```
public static readonly RegistryKey CurrentUser
```


注意：存储在此项中的信息包括环境变量的设置和有关程序组、颜色、打印机、网络连接和应用程序首选项的数据，此项使建立当前用户的设置更容易。在此项中，软件供应商存储要在其应用程序中使用的当前用户特定的首选项。

（5）OpenSubKey( )方法

此方法检索指定的子项。

语法格式为：

public RegistryKey OpenSubKey (string name,bool writable)

参数说明如下。

l name：要打开的子项的名称或路径。

l writable：如果需要项的写访问权限，则设置为 True。

l 返回值：请求的子项；如果操作失败，则为空引用。

（6）CreateSubKey( )方法

此方法创建一个新子项或打开一个现有子项以进行写访问。字符串 subkey 不区分大小写。

语法格式为：

public RegistryKey CreateSubKey (string subkey)

参数说明如下。

l subkey：要创建或打开的子项的名称或路径。

l 返回值：RegistryKey 对象，表示新建的子项或空引用。如果为 subkey 指定了零长度字符串，则返回当前的 RegistryKey 对象。

（7）GetSubKeyNames( )方法

此方法检索包含所有子项名称的字符串数组。

语法格式为：

public string[] GetSubKeyNames ()

l 返回值：包含当前项的子项名称的字符串数组。

（8）SetValue( )方法

此方法设置指定的名称/值对。

语法格式为：

public void SetValue (string name,Object value)

参数说明如下。

l name：要存储的值的名称。

l value：要存储的数据。

注意：对注册表操作使用 RegistryKey 类和 Registry 类时，必须引用 Microsoft.Win32 命名空间。

实现过程

（1）新建一个 Windows 应用程序，将其命名为 Ex16_07，默认窗体为 Form1。

（2）在 Form1 窗体中，主要添加 3 个 TextBox 控件，用于输入注册信息；添加两个 Button 控件，用来执行注册和退出操作。

（3）主要程序代码。

```
private void button1_Click(object sender, EventArgs e)
 {
    if(textBox1.Text=="")
    {
       MessageBox.Show("公司名称不能为空"); return;
    }
       if(textBox2.Text=="")
       { MessageBox.Show("用户名称不能为空"); return; }
       if (textBox3.Text == "")
       { MessageBox.Show("注册码不能为空"); return; }
       //实例 RegistryKey 类对象
    Microsoft.Win32.RegistryKey retkey1 = Microsoft.Win32.Registry.CurrentUser.
OpenSubKey("software").OpenSubKey("ZHY").OpenSubKey("ZHY.INI", true);
    foreach (string strName in retkey1.GetSubKeyNames())//判断注册码是否过期
    {
       if (strName == textBox3.Text)
       {
          MessageBox.Show("此注册码已经过期");
          return;
       }
```

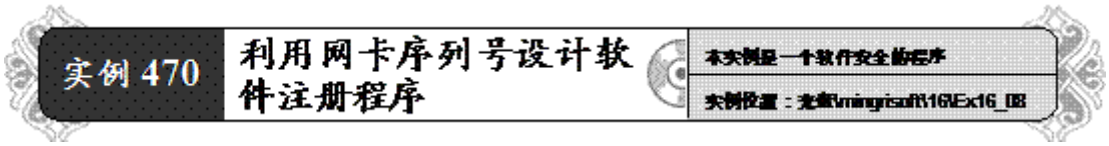
图 16.8 软件注册

```
        }//开始注册信息
        Microsoft.Win32.RegistryKey retkey = Microsoft.Win32.Registry.Cu
rrentUser.OpenSubKey("software", true).CreateSubKey("ZHY").CreateSubKey("ZHY.INI").
CreateSubKey(textBox3.Text.TrimEnd());
            retkey.SetValue("UserName", textBox2.Text);
            retkey.SetValue("capataz", textBox1.Text);
            retkey.SetValue("Code", textBox3.Text);
            MessageBox.Show("注册成功，您可以使用本软件");
            Application.Exit();
        }
```

举一反三

根据本实例，读者可以实现以下功能。

- 注册信息加密后存入注册表。
- 记录用户试用次数的注册程序。



实例 470　利用网卡序列号设计软件注册程序

实例说明

本实例实现了利用本机网卡序列号生成软件注册码的功能。运行程序，自动获得本机网卡序列号，单击【生成注册码】按钮，生成软件注册码，将注册码依次输入下面的文本框，单击【注册】按钮实现软件注册功能。实例运行结果如图 16.8 所示。

技术要点

实现本实例功能主要用到了 Microsoft.Win32 命名空间下的 Registry 类的 CurrentUser 属性、RegistryKey 类的 OpenSubKey( )方法、GetSubKeyNames( )

方法、SetValue( )方法、CreateSubKey( )方法、System.Management 命名空间下的 ManagementClass 类的 GetInstances( )方法、ManagementObjectCollection 类和 ManagementObject 类。Microsoft.Win32 命名空间下的类和方法在第 16 章实例 469 中已经做过介绍，这里不再详细说明，下面主要对 System.Management 命名空间及该命名空间下的类进行详细介绍。

（1）System.Management 命名空间

提供对大量管理信息和管理事件集合的访问，这些信息和事件是与根据 Windows 管理规范 （WMI）结构对系统、设备和应用程序设置检测点有关的。

（2）ManagementClass 类

表示公共信息模型（CIM）管理类。管理类是一个 WMI 类，如 Win32_LogicalDisk 和 Win32_Process，前者表示磁盘驱动器，后者表示进程（如 Notepad.exe）。

语法格式为：

```
public class ManagementClass : ManagementObject
```

（3）GetInstances( )方法

返回该类的所有实例的集合。

语法格式为：

```
public ManagementObjectCollection GetInstances ()
```

l 返回值：表示该类实例的 ManagementObject 对象的集合。

（4）ManagementObjectCollection 类

基于指定的查询检索管理对象的集合。此类是用于检索管理信息的较为常用的入口点之一。例如，可以用于枚举系统中的所有磁盘驱动器、网络适配器、进程及更多管理对象，或者用于查询所有处于活动状态的网络连接以及暂停的服务等。

（5）ManagementObject 类

表示 WMI 实例。

实现过程

（1）新建一个 Windows 应用程序，将其命名为 Ex16_08，默认窗体为 Form1。

（2）在 Form1 窗体中添加 4 个 TextBox 控件、3 个 Button 控件和 6 个 Label 控件。其中，TextBox 控件用输入注册码，Button 控件用来执行注册、退出和生成注册码操作，Label 控件用于显示计算机名称、网卡序列号、软件注册码和一些提示信息等。

（3）主要程序代码。

获得网卡序列号和计算机名称的实现代码如下：

```
private void Form1_Load(object sender, EventArgs e)
    {
        label2.Text = Environment.MachineName.ToString();//得到计算机名
        label4.Text = GetNetCardMacAddress();//得到网卡信息
    }
    //获得网卡信息函数
    public string GetNetCardMacAddress()
    {
        ManagementClass mc = new ManagementClass("Win32_NetworkAdapterConfiguration");
        ManagementObjectCollection moc = mc.GetInstances();
        string str = "";
        foreach (ManagementObject mo in moc)
        {
            if ((bool)mo["IPEnabled"] == true)
                str = mo["MacAddress"].ToString();
        }
        return str;
    }
```

生成注册码的实现代码如下：

```
string[] strLanCode = new string[12];// 网卡信息存储
```

```
        string[] strkey ={ "Q", "W", "7", "E", "D", "F", "2", "G", "R", "T", "Y", "8", "P",
 "N", "B", "V", "C", "X", "Z", "0", "9", "I", "8", "6", "U", "O", "P", "M", "5", "4", "3",
 "1", "A", "S", "H", "J", "K", "L" };
        //生成注册码
        public int intRand = 0;//判断随机生成次数
        private void button1_Click(object sender, EventArgs e)
        {
            //把网卡信息转换成字符串
            string strCode = GetNetCardMacAddress();//调用函数获取网卡信息
            strCode = strCode.Substring(0, 2) + strCode.Substring(3, 2) + strCode.Subst
ring(6, 2) + strCode.Substring(9, 2) + strCode.Substring(12, 2) +strCode.Substring(1
5, 2);
            string strb = strCode.Substring(0, 4) + strCode.Substring(4, 4) + strCode.Su
bstring(8, 4);//网卡信息存储
            for (int i = 0; i < strLanCode.Length; i++)//把网卡信息存入数组
            {
                strLanCode[i] = strb.Substring(i, 1);
            }
            Random ra = new Random();
            switch (intRand)//随机生成注册码的顺序
            {
                case 0:
                    label5.Text = strCode.Substring(0, 4) + "-" + strCode.Substring(4, 4)
+ "-" + strCode.Substring(8, 4) + "-" + strkey[ra.Next(0, 37)].ToString() + strkey[ra.
Next(0, 37)].ToString() + strkey[ra.Next(0, 37)].ToString() + strkey[ra.Next(0, 37)].T
oString();
                    intRand = 1;
                    break;
```

```
            case 1:
                label5.Text = strCode.Substring(0, 4) + "-" + strCode.Substring(4, 4)
+ "-" + strLanCode[ra.Next(0, 11)] + strLanCode[ra.Next(0, 11)] + strLanCode[ra.Ne
xt(0, 11)] + strLanCode[ra.Next(0, 11)] + "-" + strkey[ra.Next(0, 37)].ToString() +
strkey[ra.Next(0, 37)].ToString() + strkey[ra.Next(0, 37)].ToString() + strkey[ra.Next
(0, 37)].ToString();
                intRand = 2;
                break;
            case 2:
                label5.Text = strCode.Substring(0, 4) + "-" + strLanCode[ra.Next(0, 1
1)] + strLanCode[ra.Next(0, 11)] + strLanCode[ra.Next(0, 11)] + strLanCode[ra.Next
(0, 11)] + "-" + strLanCode[ra.Next(0, 11)] + strLanCode[ra.Next(0, 11)] + strLanC
ode[ra.Next(0, 11)] + strLanCode[ra.Next(0, 11)] + "-" + strkey[ra.Next(0, 37)].ToSt
ring() + strkey[ra.Next(0, 37)].ToString() + strkey[ra.Next(0, 37)].ToString() + strke
y[ra.Next(0, 37)].ToString();
                intRand = 3;
                break;
            case 3:
                label5.Text = strLanCode[ra.Next(0, 11)] + strLanCode[ra.Next(0, 11)]
+ strLanCode[ra.Next(0, 11)] + strLanCode[ra.Next(0, 11)] + "-" + strLanCode[ra.Ne
xt(0, 11)] + strLanCode[ra.Next(0, 11)] + strLanCode[ra.Next(0, 11)] + strLanCode[r
a.Next(0, 11)] + "-" + strLanCode[ra.Next(0, 11)] + strLanCode[ra.Next(0, 11)] + st
rLanCode[ra.Next(0, 11)] + strLanCode[ra.Next(0, 11)] + "-" + strkey[ra.Next(0, 3
7)].ToString() + strkey[ra.Next(0, 37)].ToString() + strkey[ra.Next(0, 37)].ToString()
+ strkey[ra.Next(0, 37)].ToString();
                intRand = 0;
                break;
        }
```

```
    }
```

注册软件的实现代码如下。

```
  private void button2_Click(object sender, EventArgs e)
    {
        if (label5.Text == "")
        { MessageBox.Show("请生成注册码"); }
        else
        {
            string strNameKey = textBox1.Text.TrimEnd() + textBox2.Text.TrimEnd()
+ textBox3.Text.TrimEnd() + textBox4.Text.TrimEnd();
            string strNumber = label5.Text.Substring(0, 4) + label5.Text.Substring(5,
4) + label5.Text.Substring(10, 4) + label5.Text.Substring(15, 4);
            if (strNameKey == strNumber)
            {
                Microsoft.Win32.RegistryKey retkey1 = Microsoft.Win32.Registry.Current
User.OpenSubKey("software").OpenSubKey("ZHY").OpenSubKey("ZHY.INI", true);
            foreach (string strName in retkey1.GetSubKeyNames())//判断注册码是否过
期
                {
                    if (strName == strNameKey)
                    {
                        MessageBox.Show("此注册码已经过期");
                        return;
                    }
                }//开始注册信息
                Microsoft.Win32.RegistryKey retkey = Microsoft.Win32.Registry.CurrentU
ser.OpenSubKey("software",true).CreateSubKey("ZHY").CreateSubKey("ZHY.INI").Create
SubKey(strNumber.TrimEnd());
```

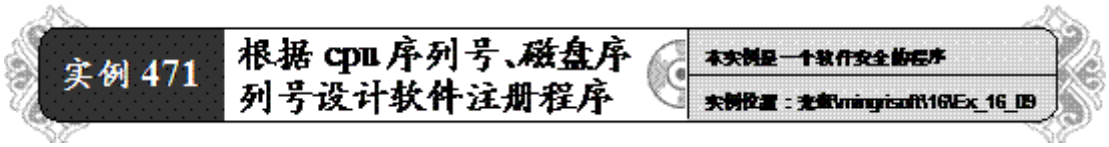
图 16.9 软件注册程序

```
                retkey.SetValue("UserName", "明日科技");
                MessageBox.Show("注册成功!", "提示");
                Application.Exit();
            }
            else
            { MessageBox.Show("注册码输入错误"); }
        }
    }
```

举一反三

根据本实例，读者可以实现以下功能。

- 应用组件的注册使用。
- 销售的软件产品进行授权。



实例 471 根据 cpu 序列号、磁盘序列号设计软件注册程序

实例说明

本实例根据计算机的 cpu 号和硬盘序列号经过简单的计算自动生成一组无规律的注册码来实现应用程序的注册。运行程序，单击【生成机器码】按钮，生成 24 位的机器码，单击【生成注册码】按钮，根据生成的机器码自动转换出 24 位注册码，将注册码输入文本框中，单击【注册】按扭，完成软件注册功能。实例运行对果如图 16.9 所示。

技术要点

实现本实例功能主要用到了 Microsoft.Win32 命名空间下的 Registry 类的 CurrentUser 属性、RegistryKey 类的 OpenSubKey( )方法、GetSubKeyNames( )方法、SetValue( )方法、CreateSubKey( )方法、System.Management 命名空

间下的 ManagementClass 类的 GetInstances( )方法、ManagementObjectCollection 类和 ManagementObject 类、Char 字符、Random 类的 Next( )方法。Microsoft.Win32 和 System.Management 命名空间下的类和方法在第 16 章实例 469 和 470 中已经做过介绍，这里不再详细讲解。下面对本实例中用到的其他知识进行详细介绍。

（1）Char 字符

Char 类型的常数可以写成字符、十六进制换码序列或 Unicode 表示形式，用户也可以显式转换整数字符代码。

（2）Random 类

表示伪随机数生成器，一种能够产生满足某些随机性统计要求的数字序列的设备。

（3）Next 方法

返回一个指定范围内的随机数。

语法格式为：

```
public virtual int Next (int minValue,int maxValue)
```

参数说明如下。

- minValue：返回的随机数的下界（随机数可取该下界值）。
- maxValue：返回的随机数的上界（随机数不能取该上界值）。maxValue 必须大于或等于 minValue。

1 返回值：一个大于或等于 minValue 且小于 maxValue 的 32 位带符号整数，即返回的值范围包括 minValue 但不包括 maxValue。如果 minValue 等于 maxValue，则返回 minValue。

实现过程

（1）新建一个 Windows 应用程序，将其命名为 Ex16_08，默认窗体为 Form1。

（2）在 Form1 窗体中，主要添加一个 TextBox 控件，用来输入注册码；添加 4 个 Button 控件，用来执行注册、退出、生成注册码和生成机器码操作；添加 3 个 Label 控件，用于显示软件注册码和机器码等信息。

（3）主要程序代码。

获得 CPU 序列号和硬盘序列号的实现代码如下：

```
public string GetDiskVolumeSerialNumber()取得设备硬盘的卷标号
{
    ManagementClass mc = new ManagementClass("Win32_NetworkAdapterConfi
guration");
    ManagementObject disk = new ManagementObject("win32_logicaldisk.devicei
d=\"d:\"");
    disk.Get();
    return disk.GetPropertyValue("VolumeSerialNumber").ToString();
}
public string getCpu()获得 CPU 的序列号
{
    string strCpu = null;
    ManagementClass myCpu = new ManagementClass("win32_Processor");
    ManagementObjectCollection myCpuConnection = myCpu.GetInstances();
    foreach( ManagementObject myObject in myCpuConnection)
    {
        strCpu = myObject.Properties["Processorid"].Value.ToString();
        break;
    }
    return strCpu;
}
```

生成机器码的实现代码如下：

```
private void button1_Click(object sender, EventArgs e)
{
  label2.Text = getCpu() + GetDiskVolumeSerialNumber();//获得 24 位 CPU 和硬盘
序列号
```

```
        string[] strid = new string[24];
        for (int i = 0; i < 24; i++)//把字符赋给数组
        {
            strid[i] = label2.Text.Substring(i, 1);
        }
        label2.Text = "";
        Random rdid = new Random();
        for (int i = 0; i < 24; i++)//从数组随机抽取 24 个字符组成新的字符生成机器码
        {
            label2.Text += strid[rdid.Next(0, 24)];
        }
    }
```

生成注册码的实现代码如下：

```
public int[] intCode = new int[127];//用于存密钥
    public void setIntCode()//给数组赋值小于 10 个的随机数
    {
        Random ra = new Random();
        for (int i = 1; i < intCode.Length;i++ )
        {
            intCode[i] = ra.Next(0, 9);
        }
    }
    public int[] intNumber = new int[25];//用于存机器码的 AscII 值
    public char[] Charcode = new char[25];//存储机器码字
    //生成注册码
    private void button2_Click(object sender, EventArgs e)
    {
```

```
        if (label2.Text != "")
        {
            //把机器码存入数组中
            setIntCode();//初始化 127 位数组
            for (int i = 1; i < Charcode.Length; i++)//把机器码存入数组中
            {
                Charcode[i] = Convert.ToChar(label2.Text.Substring(i - 1, 1));
            }
        for (int j = 1; j < intNumber.Length; j++)//把字符的ASCII值存入一个整数组
中
            {
    intNumber[j] = intCode[Convert.ToInt32(Charcode[j])] + Convert.ToInt32(Charcod
e[j]);
            }
            string strAsciiName = null;//用于存储机器码
            for (int j = 1; j < intNumber.Length; j++)
            {
                //MessageBox.Show((Convert.ToChar(intNumber[j])).ToString());
           if (intNumber[j] >= 48 && intNumber[j] <= 57)//判断字符 ASCII 值是否在
0～9 之间
                {
                    strAsciiName += Convert.ToChar(intNumber[j]).ToString();
                }
       else if (intNumber[j] >= 65 && intNumber[j] <= 90)//判断字符 ASCII 值是否在
A～Z 之间
                {
                    strAsciiName += Convert.ToChar(intNumber[j]).ToString();
                }
```

```
        else if (intNumber[j] >= 97 && intNumber[j] <= 122)//判断字符 ASCII 值是否
在 a～z 之间
                {
                    strAsciiName += Convert.ToChar(intNumber[j]).ToString();
                }
                else//判断字符 ASCII 值不在以上范围内
                {
                    if (intNumber[j] > 122)//判断字符 ASCII 值是否大于 z
                    { strAsciiName += Convert.ToChar(intNumber[j] - 10).ToString(); }
                    else
                    {
                        strAsciiName += Convert.ToChar(intNumber[j] - 9).ToString();
                    }
                }
                label3.Text = strAsciiName;//得到注册码
            }
        }
        else
        { MessageBox.Show("请选生成机器码","注册提示"); }
    }
```

举一反三

根据本实例，读者可以实现以下功能。

- 获取 CPU 信息。
- 进行远程软件产品的注册。