Delphi 开发实用技巧十例

廖为民 （武汉石化常压车间 430082）

摘要：本文总结了在用 Delphi 开发应用程序中的一些实用技巧。

自 Borland 公司推出 Delphi 以来，Delphi 便以其优良的性能及诸多的优点得到了广大程序员的喜爱。现在越来越多的人开始使用 Delphi 进行应用程序的开发。以下是进行 Delphi 程序编制时一些十分有用的技巧。

1. **程序外的 Drag & Drop 动作**

有时我们会需要将程序外的目标拖拽至应用程序，应用程序随即作出响应。例如 Win95 中的垃圾箱，只须将目标拖拽至其中即可删除目标。

在程序编制过程中可以通过调用 DragAcceptFiles 函数来实现上述功能。DragAcceptFiles 是 Windows 的一个 API 函数，它决定一个窗口是否可以接受 WM-DROPFILES 事件(在应用程序窗口释放鼠标左键时会导致发送此消息)。其形式为：

```
VOID DragAcceptFiles(
HWND hWnd, //handle of the registering window
BOOL fAccept //acceptance option
);
```

下面我们就一个实例程序来谈谈程序外 Drag&Drop 具体如何实现。

首先建立一个新项目，在上面放置一个标签对象 Label1(用来显示拖拽过来的文件名)，然后按以下步骤进行。

(1)在 TForm1.FormCreate 中加入如下代码。

```
DragAcceptFiles(Handle, True);
```

(2)在 TForm1 的 Private 段声明如下消息管理过程，用以响应 WM-DROPFILES 消息。

```
Procedure DragDropFiles(var msg:TMessage);
          message WM-DROPFILES;
```

(3)加入 DragDropFiles 的实现部分，代码如下：

```
procedure TForm1.DragDropFiles(var msg: Tmessage);
Var     Char1:array[1..128] of Char;
    Char2:PChar;
Begin
    Char2:=@CHar1; //转换 PChar
    DragQueryFile(msg.wParam, i, Char2, 128);
    Labell.Caption:=StrPas(Char2);
    DragFinish(msg.wParam); //释放内存
End;
```

DragacceptFIles、DragQueryFile、DragFinish 是 Windows 的 API 函数，详细说明可参考《Windows API 手册》或 Delphi 开发包中的 WIN API 帮助。

2. **如何在运行时拖拽和移动元件**

有时用户需要在应用程序中定制自己的使用界面，这就要求元件在运行时能够被拖动。下面的一个例子演示了如何使元件 Panel1 在运行时能被用户移动。

```
Procedure TForm1.PanellMouseDown(Sender:TObject;
    Button: TMouseButton; Shift: TShiftState; X, Y: Integer);
Const
  SC-DragMove= $F012;
Begin
    ReleaseCapture;
    (Sender as TWinControl).Perform(WM-SysCommand, SC-DragMove, 0);
End;
```

Perform 是 TControl 类中说明的方法，参数说明可参看 Delphi 帮助。ReleaseCapture 是 Windows API 函数，它将鼠标事件的发生主体约束释放，恢复正常的鼠标输入处理。必须注意的是：例子的元件 Panell 不能接受 OnClick 及 OnDBClick 等事件，若需要接受这些事件需另外编制相应程序代码，逻辑描述如下：

```
if 条件 1{例如 CheckBox1.Checked=True} then
  Begin
  正常的鼠标输入处理

  End
Else if 条件 2{例如 CheckBox1.Checked=False} then
```

```
Begin
  ReleaseCapture;
  (Sender as TWinControl).Perform(WM-SysCommand,
SC-DragMove,0);
End;
```

3. 如何定制用户光标

为使应用程序更富有个性，我们有时需要使用自己定制的光标，在 Delphi 中将非常容易。首先建立资源源文件 MYRESFILE.RES，然后为用户光标定义一个正整数索引，接着在主窗口的 OnCreate 事件响应过程中调用 LoadCursor 函数取得资源文件中索引名为 MyCursor-1 的自定义光标，代码如下：

```
{$R MyResFile.Res} {指定资源文件名}
Const crMyCursorNO = 1;
Procedure TMainForm.OnCreate(Sender:TObject);
Begin
  Screen.Cursors[crMyCursorNO]:=
    LoadCursor(hInstance,PChar('MyCursor-1'));
  Screen.Cursor:=crMyCursorNO;
End;
```

4. 隐藏窗口的标题栏

有的时候，我们并不希望窗口的标题栏出现，需要将其隐藏起来。其实你只需在想隐藏标准栏的窗口的 OnCreate 事件中加入以下代码即可：

```
Procedure TForm1.FormCreate(Sender;TObject);
Begin
  SetWindowLong(Handle, GWL-STYLE, GetWindowsLong(
    Handle, GWL-STYLE) AND NOT WS-CAPTION);
  ClientHeight:=Height;
End;
```

SetWindowLong 是一个 Windows API 函数，用法请参考 Delphi 的 WIN API 帮助。

5. 如何获得程序的命令行参数

如果你需要在你的程序中使用命令行参数，就象 Office95 中的 OSA.EXE 一样，那么可以研究一下 ParamStr 和 ParamCount 函数。ParamCount 可以获得参数的数目，ParamStr(Index:Integer)用来取得参数(ParamStr(0)用来取得执行文件的完整路径)。如：

```
For I:=0 to ParamCount do
  Labell.Caption:=ParamStr(I);
```

6. 修改显示器分辨率

在 Windows 95 及 NT 中可以修改显示器分辨率而不重新启动，其实这是利用了 32 位 Windows API 中的两个函数，EnumDisplaySettings()和 ChangeDisplaySettings()。在 Delphi 中我们用这两个函数能方便的实现这一看似神奇的功能。

```
Function ChangeResolution(X, Y:word):BOOL;
Var
  lpDevMode:TDeviceMode;
Begin
  Result:=EnumDisplaySettings(nil,0,lpDevMode);
  If Result then
    Begin
      lpDevMode.dmFields:=DM-PELSWIDTH OR
                          DM-PELSHELGHT;
      lpDevMode.dmPelsWidth:=X;
      lpDevMode.dmPelsHeight:=Y;
      Result:=ChangeDisplaySettings(lpDevMode,0)=
              DISP-CHANGE-SUCCESSFUL;
    End;
End;
```

参数 X 传递显示器横向分辨率，Y 传递纵向分辨率。需注意的是此方法只能用于 32 位的 Windows，

7. 如何 Pack 一个 DBASE 表

尽管 Delphi 的 Table 元件可以用来方便地操作数据库的表单，但 Table 元件没有提供一个方法对表单进行 Pack 操作。这对于 Pardox 或许没有什么影响，但是在没有进行 Pack 操作的 DBASE 表单中若有以被删除的记录，Table 的 RecordCount 便会大于表单实际记录数，有时会引起很大的麻烦。幸好我们可以使用 BDE API 中的 DBIPackTable 函数来补救。

具体代码如下：

```
User DBITYPES,DBIPROCS,DBIERRS;
...
DBIPackTable(Table.DbHandle,Table.Handle,
    'TABLENAME.DBF',szDBASE,TRUE);
```

要使用 DBIPackTable 函数必须在 Uses 语句中加入 DBITYPES、DBIPROCS 和 DBIERRS 三个单元说明。

8. Delphi 与 C 间的数据类型比较

在用 Delphi 开发应用程序的时候经常会遇到一些有关数据类型转换的问题，比如调用不同语言编写的 DLL 库，调用 Windows API 函数，这就需要对不同语言间的数据类型做比较。以下是 Delphi 与 C 语言之间的对比表。

C 语言	Delphi
Unsigned char	byte
Char	char
char A[size]	array[0..size-1] of char
Int	integer
Unsigned int	word
Long	longint
Unsigned Long	longint
Float	single
Double	double
char far *	pchar
char *	pchar
Struct	record
Union	record with variants

9. 利用流式对象实现文件拷贝

在 Delphi 中没有提供直接拷贝文件的过程或函数，但在 Delphi 的帮助中有一个利用 BlockRead 与 BlockWrite 函数实现文件拷贝的例子程序。然而用这种方式实现的拷贝函数程序长而不宜阅读，并且在执行过程中常会出现一些意想不到的错误。如果用流式对象来做文件拷贝就不存在这些顾虑了。Delphi 很好地继承了 PASCAL 原先的对象式数据管理的优点，并对之进行了不少的扩充，下面就是一个利用流式对象进行文件拷贝的例子。

```
Delp
Procedure FileCopy ( SourceFileName, TargetFileName: String);
Var
  S, T: TFileStream;
Begin
  S: = TFileStream. Create ( SourceFileName, fmOpenRead);
  try
    T: = TFileStream. Create ( TargetFileName, fmOpenWrite or fmCreate);
    try
      T. CopyFrom(S, S. Size);
      finally
      T. Free;
    end;
    finally
    S. Free;
  end;
End;
```

此程序思想明确、结构清晰，所以在此不做解释。

10. 将菜单项移到窗口的最右边

有些应用程序的某些菜单项，比如 Help 菜单，放到窗体的最右边，会显得十分引人注目。这一点在 Delphi 中如何实现呢？

如果你用 Delphi.0 开发 16 位应用程序，那么你只需将菜单项的 Caption 进行如下赋值即可达到这一目的。

```
MenuItem1. Caption: = #8 + MenuItem1. Caption;
```

如果你用 Delphi2 或 Delphi3 开发 32 位应用程序，就需要用一个 API 函数 ModifyMenu，具体实现过程可参考以下函数调用实例。

```
ModifyMenu( MainMenu1. Handle, 1, mf-ByPositio OR mf-Popup OR mf-Help, MenuItem1. Handle, '&Help');
```

（录稿时间：1998 年 2 月）