Windows3.x 下进程间数据共享的一种方式

袁小坊 彭楚武 傅恪 (湖南大学电气系 410082)
胡锦敏 (上海交通大学 200052)

摘要:本文给出在 16 位 Windows 环境下实现进程间数据共享的一种较简单的方式,并利用 Delphi 编程实现,文中对 Windows 的消息结构和 Delphi 的各种消息响应机制作了详尽的分析,同时给出了实例。

关键词:进程 消息 数据共享 Windows

一、引言

Windows 是多任务环境,多个进程可以同时运行,进程间可以进行数据传递实现数据共享,而实现的办法有多种,最常用的是采用 DDE(动态数据交换)方式,但 DDE 方式所牵系到的 Windows API 函数较复杂难用。我们在最近开发的"温湿度自动监控系统"中采用了一种较简洁实用的方法,其编程实现简单直接。其主要步骤有以下几步:

1.在两进程中注册同一用户消息,利用此消息传递信息。

2.发数据的进程在全局堆中申请一块共享内存,用来存放要传递的数据。

3.发数据的进程将共享内存的句柄或地址指针用消息传给接收数据进程。

4.接收数据的进程将接收到的内存句柄转化为指针,利用此指针读取数据。

本"温湿度自动监控系统"有两个进程,一进程在后台运行专门负责通过 COM 口与下位机通信获取数据,另一进程负责数据处理、图形分析、系统管理等。之所以这样分是由于下位机数目较多,通信任务很重,如果只用一个进程来处理的话,系统将忙于通信而不能及时响应用户事件。本系统用 DELPHI1.0 编写,采用上述方法实现两进程间的数据共享和相互信息传递。

二、编程实现

1.注册用户消息

Windows 是一个消息驱动式系统。对每一个输入事件 Windows 都会产生一个对应的消息,并把这些消息收集在系统队列中,然后将它们放入应用程序队列。应用程序亦能产生消息,并将它们送入其他应用程序队列,这就使得应用程序间的通信成为可能。VC++,BC++,Turbo Pascal for Windows 以及 Delphi 都支持在程序中定义用户消息。

Windows 的消息是一个包含一些重要字段的数据记录。这些字段中最为有用的是一个整数消息标识符。Delphi 的 Message 单元已对所有在 Windows 中预定义消息的标识符进行了声明。Windows 的消息标识符值分以下几类,每类的范围如下:

(1) 0 到 WM-USER(0x4000) 间为 Windows 保留的消息值即系统预定义的消息。

(2) WM-USER 到 0x7fff 间的值被应用程序用作某一窗口类的私有消息,某些构件如 BUTTON, EDIT, COMBOX, LISTBOX 可能已用了此范围中的某些值。在一个应用程序中传递窗口消息就是用这一范围的值。

(3)0X8000 到 0xbfff 间的值为 Windows 保留以作将来之用。

(4)0xc000 到 0xffff 间的值可以由应用程序在运行时通过 RegisterWindowMessage(PCHAR)函数获得,如各应用程序对一相同的字符串用此函数,它们将获得一相同的系统中唯一的消息标识符值,此消息标识符值并不是一固定值,每次运行时值会不一样。

由上可知,利用 RegisterWindowMessage(PCHAR)函数是获得在系统中唯一的用户消息标识符值的最佳办法。

在一条消息中其他有用的信息来自一个 16 位参数字段,一个 32 位参数字段及一个结果字段的内容。TMessage 对象的声明是一种常用的声明形式,它是 Delphi 的通用消息记录类型:

```
Tmessage = record
    MSG: word;
    Wparam: word;
    Lparam: longint;
    Result: longint;
end;
```

我们可以利用该记录的 Lparam 项传递共享内存地址指针。

消息的发送可用 PostMessage 函数，此函数发送完消息后不等待目标进程处理该消息而立即返回。

2. 申请共享储存区

16 位 Windows 中，每一个进程有自己的局部堆，所有进程共享一个全局堆，所以进程间可以用全局堆作为共享数据的办法。相关的 API 函数有 GlobalAlloc, GlobalLock, GlobalFree。GlobalAlloc；返回所申请到的内存块的句柄，如 memhndl: = GlobalAlloc(GMEM-SHARE , 60) 在全局堆中申请一块大小为 60 字节的全局共享内存块；pmem: = GlobalLock(memhndl)锁住这块内存，并返回一个指向它的第一字节的指针。GlobalFree(memhndl)释放这块内存。

实际上 16 位 Windows 环境下 DDE 也是利用全局堆来进行数据传递。

3. 在 Delphi 中响应消息

Delphi 是一种完全的面向对象的编程语言，彻底而又完整地实现了分类继承及封装的概念。Delphi 的可视部件库(VCL)对象体系是它与之打交道的“世界”的封装。VCL 基本上是建立在 Windows 的 API 基础上，并且是围绕着 Windows 窗口化系统及用户界面模式设计的，具有对所有 Windows 消息的响应能力。利用已有部件封装的强大功能，稍加改变可以很方便实现以前传统 API 编程所能实现的功能。

在 Delphi 中有三种方式可方便地实现进程对用户消息的响应：

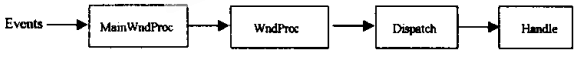

图 1

(1)重载 WndProc 过程。Delphi 通过为一个应用程序中的所有窗口化部件类型注册一个名为 MainWndProc 的方法作为窗口过程来处理消息。其处理流程如图 1 所示：

相关部件的继承体系如图 2 所示：

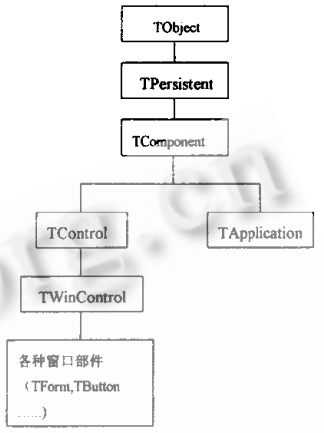

图 2

MainWndProc 为 TWincontrol 的一个方法，Twincontrol 是所有窗口化控件类型的父类。MainWndProc 包含一个异常处理块，它用来将消息记录从 Windows 传递到一个称为 WndProc 的虚拟方法中，并且通过调用该应用程序对象的 HandleExeception 方法来处理任何一个异常。WndProc 为 Tcontrol 部件的一个方法，而 Tcontrol 是所有可视控件的父类。

MainWndProc 不对任何一种特定的消息作专门地处理。鉴于每个窗口化部件都继承了这种虚拟方法且可以为自己的目的重载它，因而可以对 WndProc 的消息处理进行定制，使之响应用户消息。

WndProc 处理的消息标识符值只能在 1 到 0xbfff (49151)之间。

具体作法如下例：

```
.....
type
  TForm1 = class(TForm)
        Button1: TButton;
        procedure Button1Click(Sender: TObject);
        .....
  protected
        procedure wndproc(var message: tmessage); override; {重载 wndproc}
```

```
  end;
const
  Tstmsg = WM-USER + 10;
.....
procedure TForm1.Button1Click(Sender: TObject);
begin
  PostMessage(form1.handle, Tstmsg, 0, 0);
end;
procedure Tform1.wndproc(var message:tmessage);
  begin
    if message.msg = Tstmsg then
    showmessage('ok');
    inherited WndProc(Message); {必须继承原有方法}
  end;
end.
```

(2)利用 message 编译指令。实际上，Tcontrol .WndProc 方法调用了一个从 Tobject 继承来的方法:Dispatch 来判定调用何种方法去处理某一消息，Tobject 是所有 Delphi 部件的父类。Dispatch 利用消息标识符来判定如何发送一条特定消息,如果该部件定义了一个用于此消息的特定事件处理程序，Dispatch 将调用其消息方法。该消息处理程序是一个部件的、用 message 编译指令声明的方法。如果该部件没有为这个消息定义一个处理程序，Dispatch 将调用 DeFaultHandle。

message 编译指令声明的消息标识符值亦只能在 1 到 0xbfff(49151)之间。即用 RegisterWindowMessage (PCHAR)函数获得的用户消息标识符值是不能用 message 声明的。

一个窗口实际就是一个部件，当然可以用 message 编译指令为她声明一个响应特定消息的方法。具体作法如下例：

```
interface
.............
const
  shshmsg = WM-USER + 10;
type
  TForm1 = class(TForm)
  ..........
  procedure FormCreate(Sender: TObject);
  private
  { Private declarations }
   procedure echoshshmsg ( cmsg: Tmessage ); message
shshmsg;
public
  { Public declarations }
end;
.........
procedure Tform1.EchoShshmsg(var cmsg:TMessage);
begin
  ....
  showmessage('message ok');
end;
.......
```

(3).用 Application.OnMessage 事件。在 DELPHI 中一个 TApplication 构件对应着一个进程，Tapplication 包裹了一套消息处理机制，在接收到任何投递的 Windows 消息时会激发它的 OnMessage 事件，TApplication.OnMessage 的定义如下：

```
TMessageEvent = procedure (var Msg: TMsg; var
Handled: Boolean) of object;
property OnMessage: TMessageE ent;
```

TmessageEvent 是 OnMessage 事件的类型，它指向一个处理所接收到的消息的方法。

在应用程序中为 OnMessage 事件创建一个句柄就能调用其他过程来响应此消息。如果应用程序没有响应此消息的句柄，消息就会被传给 Windows，由 Windows 处理。所以 OnMessage 事件句柄使得应用程序赶在 Windows 之前处理接收到的消息。而 Tappliction 是不可见的，所以不可能用 Object Inspector 来为它的事件编写处理语句。可以通过以下步骤为 TApplication 的 OnMessage 事件创建事件句柄：

①编写一事件句柄作为此工程主窗口的一方法。

②在主窗口的类定义中声明此方法。

③在主窗口的 oncreate 事件中，将此方法名赋给 TApplication 的 OnMessage 事件。

Application.OnMessage 只响应用 PostMessage 投递的消息。

Application.OnMessage 可以响应用 RegisterWindowMessage(PCHAR)函数获得的用户消息标识符值。

我们在开发“温湿度自动监控系统”时用的正是这种

方法。

具体实现过程如下：

消息发送进程：

```
.......
var
  ......
  Tsmsg:word;
  memglb:word;
  pstr:pchar;
.........
procedure TForm1.FormCreate(Sender: TObject);
begin
  Tsmsg:= RegisterWindowMessage('Tstmsg'); {注册用户消息}
  memglb:= GlobalAlloc(GMEM-SHARE ,60); {获取共享内存句柄}
  pstr:= GlobalLock(memglb); {获取指针}
end;
procedure TForm1.Button1Click(Sender: TObject);
var
  winhnd:word;
  begin
    winhnd:= FindWindow(nil,'formrc'); {获取接收数据进程窗口句柄}
PostMessage;(winhnd, Tsmsg, 0, Longint(pstr)); {发送消息}
end;
.........
```

消息接收进程：

```
.........
TFormrc = class(TForm)
......
private
    { Private declarations }
procedure Appmsg(var Msg: TMsg; var Handled: Boolean); {声明一个消息响应过程作为窗口 的私有方法}
......
var
Tsmsg:word ;
pstr:pchar;
.....
procedure TFormrc.FormCreate(Sender: TObject);
begin
  Tsmsg:= RegisterWindowMessage('Tstmsg');
  Application.OnMessage:= Appmsg; {将消息响应过程赋给 Application.OnMessage 事件}
end;
procedure TFormrc.Appmsg;(var Msg: TMsg; var Handled: Boolean);
begin
    if msg.message = Tsmsg then
      pstr:= pointer(msg.Lparam);
end;
.....
```

4. **启动、关闭通信进程**

在主进程启动时用 API 函数 WinExec 启动通信进程，在主进程结束时用 PostMessage 向通信进程发 WM-QUIT 消息结束通信进程。

三、结论

这种实现进程间数据交换的方式简单实用，利用 Delphi 强大的功能可以很方便地编程实现。文中给出的在 Delphi 中响应用户消息的方法在 16 位和 32 位 Windows 环境下都适用。我们在开发时考虑到用户的设备档次不高，因此采用 Delphi1.0 在 16 位 Windows 环境下开发。在 32 位 Windows 下一个进程可拥有多线程，因而可以创建一个线程专门用于通信，这样就能更好地解决问题。另外在 32 位 Windows 下每个进程拥有一个堆，进程的地址空间是分离的，已无全局堆的概念，不能采用 GlobalAlloc 与 GlobalLock 使进程间共享数据，只能采用内存映射文件方式。

参考文献

[1] Delphi For Windows Power Toolkit [美] Harold. Davis

[2] Windows 95，WindowsNT3.5 高级编程技术 [美] Jeffrey Richter

（来稿时间：1997 年 12 月）