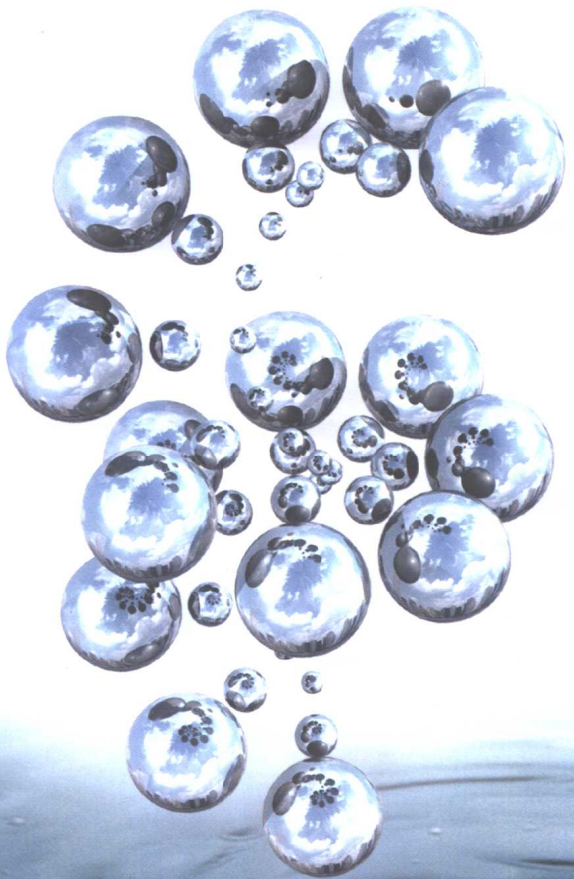
万水计算机实用编程技术系列

Delphi 6

高级编程技术

阶梯创作室　编著



中国水利水电出版社
www.waterpub.com.cn

万水计算机实用编程技术系列

Delphi 6 高级编程技术

阶梯创作室　编著

中国水利水电出版社

内 容 提 要

Delphi 作为 Windows 平台下最主要的 RAD 工具之一，同时具有方便高效和功能强大的优点。支持在 Windows 平台下进行包括一般 Windows 应用、数据库应用、分布式应用、网络通信应用、Internet/Web 应用等方方面面的开发工作。Borland 最新的 Delphi 6 较以前的版本又有了重大的改进，增加了 BizSnap、WebSnap、DataSnap 三项最新技术，将 Delphi 应用开发带入一个全新的领域。加上全新的 CLX，使 Delphi 开始具有跨平台的开发能力。

本书分为十章，介绍了用 Delphi 进行包括多线程、网络通信、Web 应用、分布式技术（包括 COM、MTS/COM+、CORBA、SOAP 等）和自定义组件等。本书的特点在于着重介绍一些 Delphi 6 特有的新技术，如进行 Web 应用开发的利器 WebSnap 技术，最新的基于 XML 的分布式技术 SOAP/WebService 等，对一些已经成熟的重要技术，如 COM、网络通信等也有一些独到的介绍，对一些较以往有重大改进的技术也有特别的说明，如 CORBA 等。本书以大量实例为基础，相信读者通过阅读示例程序，能够对 Delphi 6 的高级应用开发有较为详细的了解。

本书是一本针对 Delphi 6 开发人员编写的开发指南，其中的许多内容具有相当的深度和超前性，适合于有一定经验的 Delphi 开发人员，部分具有普遍性的内容，对其他语言的开发者也有一定的参考价值。

图书在版编目（CIP）数据

Delphi 6 高级编程技术 / 阶梯创作室编著. —北京：中国水利水电出版社，2002

（万水计算机实用编程技术系列）

ISBN 7-5084-1121-8

Ⅰ.D… Ⅱ.阶… Ⅲ.DELPHI 语言—程序设计 Ⅳ.TP312

中国版本图书馆 CIP 数据核字（2002）第 037584 号

书　　名	Delphi 6 高级编程技术
作　　者	阶梯创作室　编著
出版、发行	中国水利水电出版社（北京市三里河路 6 号　100044） 网址：www.waterpub.com.cn E-mail：mchannel@public3.bta.net.cn（万水） sale@waterpub.com.cn 电话：（010）68359286（万水）、63202266（总机）、68331835（发行部）
经　　售	全国各地新华书店
排　　版	北京万水电子信息有限公司
印　　刷	北京市天竺颖华印刷厂
规　　格	787×1092 毫米　16 开本　17.75 印张　388 千字
版　　次	2002 年 6 月第一版　2002 年 6 月北京第一次印刷
印　　数	0001—4000 册
定　　价	28.00 元

前　言

IT 业一向是一个在变革中求发展的行业。2001 年，全球 IT 业经历了一次大的动荡，各公司为了生存和发展，推出了让人眼花缭乱的各种新技术，原有的成熟技术也大多有了重大的改进。作为最主要的开发工具软件开发商的 Borland 公司（原 Inprise 公司）也在这一年里推出了多项重要的新产品如 Kylix 1、2，JBuilder 5、6，BAS 等，而其中最重要的产品之一便是 Delphi 的最新版本 Delphi 6。

Delphi 6 可以说是自 Delphi 诞生以来划时代的一个版本。因为相比以前的各个版本，Delphi 6 有了非常大的变化。其中最重要的就是从 Delphi 6 开始使用了一个全新的 Application Framework—CLX（Borland Component Library for Cross-Platform），CLX 是 Borland 专门开发的一个全新的组件库，它支持跨平台的应用开发（需要 Kylix），并且 CLX 与过去版本的 Delphi 所用的 VCL（Visual Component Library）很相似。此外 Delphi 6 还有很多的新特性，如有一个全新的 IDE：改进的 Object Inspector、Code Insight、Data Module 等；新增的 Object Treeview，根据使用 VCL 还是 CLX 而自动变化的组件栏等；改进的 Object Pascal 编译器；增强了对 COM 的支持；增加了对 IDL2PAS 支持的 CORBA 应用开发；增加了对 XML 的支持；增强了网络方面的开发；增加了一种新的跨平台数据库访问方式 dbExpress；特别是 Borland 主推的三项技术：BizSnap、WebSnap、DataSnap 更是 Delphi 6 的最大亮点。

软件开发界有一个很流行的说法："真正的程序员用 C++，聪明的程序员用 Delphi"。这样评价 Delphi 是恰如其分的，因为它支持在 Windows 平台下进行包括一般 Windows 应用、数据库应用、分布式应用、网络通信应用、Internet/Web 应用等方方面面的开发工作，同时还具备方便高效的可视化快速应用开发能力。正因为 Delphi 是如此的博大精深，要想在一本书里把 Delphi 进行应用开发的所有方面都一一说明几乎是不可能的，即使是像 Delphi 的作者之一的 Charlie.Calvert 也没有完全做到。本书也没有打算成为一本不全的所谓"大全"，只是想针对 Delphi 6 的一些新特性和一些有较大改进的技术做个介绍，当然其中也有一部分内容是作为基础知识来介绍的，以适应读者的需要。

技术的发展速度即使用"日新月异"都几乎不足以形容了，就在本书编写的过程中 Borland 公司发布了 Kylix 2 和 JBuilder 5，并发表了 Delphi 6 的第一个补丁程序；Microsoft 的.Net 也出了好几个测试版；其他如 IBM、SUN、ORACLE 都有各自的代表最新技术的产品问世。就在本书刚刚完成的时候，Borland 发表了 InterBase 6.5，并宣布了 2002 年度的 C++开发工具发布计划。本书中所介绍的新技术也许很快就会有更新的发展，所以各位读者们还需要时时关注来自 Internet 的最新信息。

本书由谢乐健策划，参与编写的人员有张海鹰，王玉馨，马文刚、董瑞熙在此书编写

过程中还得到薛青峰，虞磊廷，管知时，闫绪峰，朱海明，陈逸帆，沈玮，吴华雅的支持和帮助。

感谢中国水利水电出版社的孙春亮和石永峰先生，他们一丝不苟的工作态度鼓舞了我们，他们尽心尽力的工作使本书能与广大读者见面。

限于作者水平，加上出版时间仓促，书中难免有疏漏和不妥之处，敬请读者批评指正。

阶梯创作室

2002.4 于上海交通大学

目　　录

第一章　多线程编程

1.1　多线程技术

多线程是随操作系统发展产生的新概念，一个应用程序的一次运行就是一个进程。早期的 DOS 操作系统是单任务操作系统，在同一时刻只能运行一个任务（进程）。而现在的 Windows 操作系统是一个多任务操作系统，在同一时刻可以有多个进程。从 Windows 95 开始支持多线程，每个进程可以拥有多个线程。一个进程至少要包含一个线程，称为主线程。单 CPU 计算机不可能同时执行多个线程，而是由操作系统动态地将 CPU 的工作划分成时间片，分配给每个进程中的每个线程，从宏观上看好像是多个线程在同时运行。多线程技术通常用于几个操作必须同时进行不能等待的情况，或者在进行一些费时的操作时，用另一个线程来处理，以免用户等待。因为级别高的线程可以获得更多的系统时间，所以完成任务所需的时间更少。在目前的很多应用程序中都采用了多线程技术例如信息采集、多媒体处理等。

虽然，多线程技术能减少用户等待进程运算结果的时间。但是，多线程技术不能滥用于进程中，因为线程的切换要占用一些资源，过多地使用会对系统造成不良影响。另外，对于像 Windows 95/98/ME 等操作系统，对线程的支持不是非常好，当线程达到一定数量时，会造成系统的不稳定，甚至崩溃。还有就是在多线程编程时需要认真处理临界资源的同步和互斥的问题，否则可能造成程序的运行不正常，甚至出现灾难性的后果。读者在使用多线程前必须先考虑清楚，以免造成不必要的损失。

要编写多线程程序，首先要知道进程和线程的区别。简单地说，进程就是一个调入内存执行的应用程序。微软的 32 位操作系统（Windows 95 及以上版本）为每个进程单独分配一个虚拟的地址空间和单独的消息队列。但是进程并不直接运行代码，而是通过控制线程的开始与结束来运行。每个进程中至少有一个线程，这个线程被成为主线程。一个进程可有多个线程。所有的线程都共享进程的虚拟地址空间，但是所有的线程都必须由一个主线程控制开始与结束。线程之间还可以共享进程的全局数据和资源。

1.2　创建多线程应用

多线程编程大致分为以下四个步骤：一是创建线程，二是设置线程的优先级，三是挂起和唤醒，四是结束线程。

一种创建线程的方法是用 Windows API 函数，建议使用 System 单元中定义的 BeginThread

函数。调用 BeginThread 函数除了能够创建一个线程并返回线程外，还能自动将 System 单元中声明的全局变量 lsMultiThread 设为 True，这样，Delphi 的堆管理器就知道当前有多个线程在运行，从而防止多个线程同时修改它的内部结构，这是 Delphi 的一种安全机制。调用 BeginThread 函数还可以在创建一个新的线程时，同时为这个线程建立一个异常框架。

BeginThread 格式为：

```
function BeginThread(SecurityAttributes: Pointer; StackSize: LongWord; ThreadFunc:
TThreadFunc; Parameter: Pointer; CreationFlags: LongWord; var ThreadId: LongWord): Integer;
```

描述：

Security Attributes 参数用于设置线程的访问权限（或称安全属性），如果设为 NULL 表示用默认的安全属性，只有 Windows NT/2000 支持此属性，对于 Windows 9x，只能使用默认的安全属性。StackSize 参数用于设置分配给线程的栈空间大小，每一个线程都有自己独立的栈空间。ThreadFunc 参数指定一个线程要执行的函数，Parameter 参数是它的参数。CreationFlags 参数指定线程创建后是否立即执行，设为 0 表示立即执行，设为 CreateSuspended 表示处于挂起状态，除非调用 ResumeThread 函数将它唤起。ThreadID 是一个变量参数，BeginThread 函数返回线程的 ID，Delphi 会保证每一个线程的 ID 是唯一的。如果 ID 值为 0，表示线程没有创建成功，可以调用 Windows API 的 GetLastError 函数分析错误的原因。

线程函数执行到函数的 End 处结束，返回值就是线程的退出码。因此，线程往往被设置成无限循环。如果线程函数中调用了 EndThread 过程，线程就会被终止，但是线程对象并没有被从内存中删除，要删除线程对象，还要调用 CloseHandle 函数，这个函数同时释放线程分配的堆栈和资源。

线程和进程一样，也有优先级，级别越高的线程越是优先执行，只有当优先级高的线程被挂起时，较低的线程才能有执行的机会。当一个应用程序被执行时，其进程默认的优先级是 NORMAL_PRIORITY_CLASS 值，通过调用 API 函数 SetPriorityClass 函数重新指定线程的优先级别。

NORMAL_PRIORITY 有两种设置，这是为了前台运行的进程能更快地响应用户输入。线程默认的优先级就是进程的优先级，可以通过 SetThreadPriority 重新设置优先级，不过此优先级是相对于进程的。

如果用户要在一个进程中动态创建多个线程，则必须先考虑这个进程中的其他线程以及其他进程。调用 GetThreadPriority 可以返回线程的优先级。

通过 API 函数 SuspendThread 可以挂起线程，通过 API 函数 ResumeThread 可以唤醒线程。SuspendThread 和 ResumeThread 函数的格式如下：

```
Function SuspendThread(hThread: Thandle): DWORD;
Function ResumeThread(hThread: Thandle): DWORD;
```

每调用一次 SuspendThread，就将线程挂起的次数加一。而每调用一次 ResumeThread，就将线程挂起次数减一。如果调用成功，返回值就是被挂起的总次数，不成功就返回

0xFFFFFFFF。

要同步线程，可以使用 API 函数 WaitForSingleObject 函数和 WaitForMulitpleObjects 函数。

虽然 API 函数调用的方式比较直接，但是需要学习复杂的 API 函数。Delphi 提供了另一种方法来实现多线程，这就是 TThread 类。早期的 Delphi 的 VCL 有一个缺陷，就是有很多类和控件不支持多个线程同时访问，而 TThread 可以在一定程度上弥补这个缺陷，TThread 通过 Synchronize 方法可以让多个线程互斥地访问 VCL。从 Delphi 4 开始，新的 VCL 中的很多类和控件（主要是非可视控件）已经可以用于多线程，特别是 TCanvas 等类增加了 Lock/Unlock 方法，很好地解决了它们在多线程中出现的问题。

TThread 是一个抽象类，不可以直接用于生成线程实例，要建立一个线程，必须从 TThread 派生一个类。在一个进程中创建一个 TThread 的子类代表创建了一个新的线程类型，而每一个这种类的实例都是一个这种类型的线程。Delphi 为多线程建立了向导，使编程非常容易。选择 File 菜单栏中的 New 子菜单中的 Other...命令，弹出 New Items 对话框，然后双击 Thread Object 图标，之后会弹出如图 1-1 所示对话框，可输入所要建立的线程对象的类名。

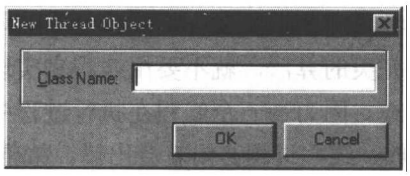


图 1-1　New Thread Object 对话框

Delphi 为这个线程对象建立了程序框架，如下所示：

```
unit Unit2;
interface
uses
  Classes;

type
  TMyThread = class(Tthread)
  private
    { Private declarations }
  protected
    procedure Execute; override;
  end;

implementation

procedure Classes.Execute;
begin
  { Place thread code here }
```

```
end;
end.
```

被重载的方法 Execute 是线程的入口。当线程建立后，有两种方法来执行它。一是使线程立即执行，二是使线程暂停等待条件满足后再执行。当然，在建立线程对象的时候，可以使用 Create 来控制。以下是 TThread.Create 的定义：

```
constructor Create(CreateSuspended : Boolean);
```

当 CreateSuspended 为 True 时，建立一个暂时停止的线程。如果为 False，建立一个立即执行的线程。如果使用自定义线程类时，一般要重载这个构造函数。

Synchronize 过程解决了 Delphi 不能同时访问 VCL 的缺陷。

Synchronize 格式为：

```
Procedure Synchronize(Method:TThreadMethod);
```

Method 参数指定了一个方法，用这个方法去访问 VCL。线程本身不调用这个方法，而是通知主线程调用这个方法，主线程一次只能收到一个通知，从而避免了因为对 VCL 并发访问而造成的冲突。

读者在编写线程时必须先考虑您在做什么，不要将过多的垃圾代码夹杂在线程中。所以编者认为，如果没有一个较为完美的算法，就不要用线程来实现。同时在线程中不要用传址方式传递参数，而改用传值方式。因为当一个线程在执行过程中时，很可能主线程已经将此地址处的参数所占内存空间释放，线程过程访问就会出错。希望读者先巩固好本节的知识点，再完成下面的多线程编程的学习。

1.3 线程类 TThread

TThread 是线程对象类，以下是其基本属性和方法。

- Handle

格式：property Handle: Integer;

描述：线程的句柄。

- Priority

格式：property Priority: TThreadPriority

描述：线程的优先级。

- ThreadID

格式：property ThreadID: THandle;

描述：线程的标识号。

- Suspended

格式：property Suspended: Boolean;

描述：返回线程是否处于挂起状态，也可以设置布尔值控制线程的状态。

- FreeOnTerminate

格式：property FreeOnTerminate: Boolean;

描述：线程结束时否自动删除线程对象。

- ReturnValue

格式：property ReturnValue: Integer

描述：设置线程的退出码。

- Create

格式：constructor Create(CreateSuspended: Boolean);

描述：建立线程，当参数 CreateSuspended 为 True 时，创建后处于挂起状态；值为 False 时，线程创建后处于唤醒状态。

- Execute

格式：procedure Execute; virtual; abstract;

描述：线程执行代码，由于它是虚拟和抽象方法，所以派生的线程类必须要重载它，重载后的 Execute 就是线程执行的入口。

- DoTerminate

格式：procedure DoTerminate; virtual;

描述：线程结束后自动调用它，之后调用 OnTerminate 事件进行处理。

- WaitFor

格式：function WaitFor: LongWord;

描述：等待线程结束后返回。如果线程调用 Synchronize 过程，不能在主函数之中调用 WaitFor 过程，这样会导致死循环。

- Suspend

格式：procedure Suspend;

描述：挂起线程。

- Resume

格式：procedure Resume;

描述：唤醒线程。如果线程调用被多次挂起，那么也要唤醒相应的次数。

- Terminate

格式：procedure Terminate;

描述：终止线程，并将属性 Terminate 设为 True。在 Execute 过程中应一直检测它，当 Terminated 为 True 时，结束线程。

- Destroy

格式：destructor Destroy; override;

描述：析构函数，终止线程。如果线程正在执行，那么调用 Terminate 方法将 Terminated 设为 True，并调用 WaitFor 函数等待线程真正终止，最后删除线程句柄。

➢ OnTerminate

格式：property OnTerminate: TNotifyEvent;

描述：当线程的 Execute 执行到 End 时调用这个事件。

➢ Sychronize

格式：procedure Synchronize(Method: TThreadMethod);

描述：解决访问 VCL 冲突的方法。

Execute 方法就是在线程中实现的过程，它和应用程序的主线程共享一个进程空间。在编写线程时必须先考虑以下两个问题：

- 如何保证线程使用的内存空间拒绝被其他子线程并发使用。
- 不同的线程之间如何共享一个内存空间，并且通过共享的内存空间使不同的线程之间通信。

1.4 协调线程

在线程中调用的 VCL 对象，在运行时往往它们的属性和方法会发生冲突。因此在设计多线程时，还应考虑到怎样捕捉这些属性和方法产生的 Windows 消息。对于这个问题，往往采用一种主线程技术，就是将所有 VCL 对象的属性和方法都在这个线程中实现，通过这个线程完成 Windows 消息的接收。

在实现多线程的同时，将会使用到线程的局部变量。通过线程的局部变量，可以轻松地控制线程的运行，以及利用线程采集信息。使用线程的局部变量，就可以轻松解决线程之间的通信问题。

往往一个线程的开始与结束关系到整个进程的运行安全，但是就是因为往往忽略这个问题，产生了类似中断冲突及内存地址无法找到等较为严重的错误，在调试线程时往往以重新启动来结束一个进程。因此多数的程序员放弃使用线程技术。其实，可以通过其他的监视线程检查和控制线程的运行。例如，当外在条件被满足后，调用线程的 Terminate 方法，从而设置属性 Terminated 为 True 达到终止线程的目的。

当线程结束后，必须释放线程独享的内存空间。所以，可以使用 TThread 的 OnTerminate 事件处理释放线程独享的内存空间。但是必须注意两点：

- 在触发 OnTerminate 事件后，部分线程局域变量已经被释放。因此，在 OnTerminate 中不能访问线程的局部变量。
- 在触发 OnTerminate 事件后，OnTerminate 中的函数在主线程中执行，因此，不必担心和别的线程发生冲突。

在并发执行线程时会遇到以下两个较为突出的线程互斥问题：

- 如何避免多个线程同时访问一个全局变量。
- 当一个线程结束后，如何自动唤醒另外一个线程。

如果要解决这两个问题，只需控制当某个线程启动或挂起时，其他的线程就自动挂起或唤醒。在 VCL 中已经提供了三种解决方法：

- TThreadList（线程列表对象）有 LockList 方法，能够阻止其他线程同时对它访问，直到 UnLockList 方法被调用，它才可以被其他的线程访问。这种方法称为锁对象。
- 设置临界段，是针对对象不提供 Lock 的方法而设计的。临界段就像一个门，同一时刻内只允许一个线程被访问。它是通过创建一个 TCriticalSection 的全局实例来实现的。TcriticalSection 可以 Acquire（锁定）和 Release（放开）。注意，只有所有线程都是通过设置临界段访问与之相连的全局变量，否则这种方法无法起到作用。
- 使用多线程时往往是对全局变量进行读操作，而不是写操作。因此不会造成错误。TMultiReadExclusiveWriteSynchronize 对象来实现多个线程同时读全局变量，而只允许一个线程更改这个全局变量。注意，只有所有线程都是通过设置临界段访问与之相连的全局变量，否则这种方法无法起到作用。

如果当您的线程必须要等待其他线程的任务结束才能够继续，那么必须解决线程同步的问题。可使用以下两种方法：

- 等待其他线程结束：可以调用 WaitFor 方法实现。
- 等待一个作业完成：在设计多线程时往往等待的是一个操作的结束，而不是整个线程的结束。因此，可以使用 TEvent 对象，但是这个对象必须作为全局变量使用。当一个线程完成了设计中的任务后，就调用 TEvent.SetEvent 方法，SetEvent 打开一个标志，其他的线程可以检查这个标志，从而得知需要的任务已完成。如果要关掉这个标志就调用 ResetEvent 方法。

1.5　多线程编程实例

通过以上的学习，读者已经掌握了多线程编程原理和技术，那么如何使用多线程编写一个实实在在的程序呢？下面通过实例展开练习。

当运行建立的实例后，一个窗体被打开，如图 1-2 所示。

图 1-2　打开窗体

由于对比度较小因此读者有可能无法清楚地看到本实例的运行过程，图中有一段文字在指定的范围内运行，并不断地更换其中的内容。可以通过设定消息窗口完成对其中内容的修改。如图 1-3 所示。

编辑的每一行文字将被显示在预览滚动条中。点击 Ok 完成编辑，并更新主程序中的消息列表。

通过以上的演示，可以发现在编辑信息的同时主程序并没有暂停过，这就是多线程同步