

吴志辉 著

(附光盘)

用 Delphi 开发共享软件



化学工业出版社

吴志辉 著

用 Delphi 开发共享软件



化学工业出版社

· 北京 ·

本书通过开发串口通讯程序、万能图形编辑器、通用表格编辑器、逻辑梯形图编辑器、CMOS 数据读写程序、多声卡音频自动播出系统、报表打印与动态连接库、坦克大战游戏、系统编程——键盘监视、服务程序——定时关机、进程隐藏——监视系统键盘、局域网编程、再谈 Win32 SDK 编程 13 个编程技巧, 以及工作任务管理器、桌面提示器、邮件群发程序 3 个软件编程实例使读者学习和掌握 Delphi 开发应用程序的技巧和方法。

本书主要读者对象是有志开发共享软件的在校学生、程序员以及编程爱好者, 也可以作为计算机软件专业教学第二课堂的培训教材。

图书在版编目 (CIP) 数据

用 Delphi 开发共享软件/吴志辉著. —北京: 化学工业出版社, 2008.11
ISBN 978-7-122-03710-7

I. 用… II. 吴… III. 软件工具-程序设计 IV. TP311.56

中国版本图书馆 CIP 数据核字 (2008) 第 140278 号

责任编辑: 王湘民
责任校对: 边 涛

文字编辑: 朱 恺
装帧设计: 张 辉

出版发行: 化学工业出版社 (北京市东城区青年湖南街 13 号 邮政编码 100011)

印 刷: 大厂聚鑫印刷有限责任公司

装 订: 三河市万龙印装有限公司

787mm×1092mm 1/16 印张 16¼ 字数 414 千字 2009 年 1 月北京第 1 版第 1 次印刷

购书咨询: 010-64518888 (传真: 010-64519686)

售后服务: 010-64518899

网 址: <http://www.cip.com.cn>

凡购买本书, 如有缺损质量问题, 本社销售中心负责调换。

定 价: 49.00 元 (附光盘)

版权所有 违者必究

前 言

本人1990年毕业于北京大学，是一个学化学的研究生，其后在一所学院教书，业余喜欢捣鼓计算机。经过6年自低级到高级的辛勤钻研，自学成“菜”之后便下海了。在市场经济的这块“商海”中游历了十余年后，因某种原因又重新回到了学校，小康目标没怎么实现，只是多了些人生的经历和教训，也不算后悔吧，直到现在还偶尔写点程序。回想自Windows 95出来后，在DOS环境下靠C/C++写程序的日子基本宣告结束，随后便选择了Delphi作为Windows下的开发工具进行着程序开发的工作。多年的经历，使我总想写一本书。加之以前的很多学生，也有经常问及一些Delphi开发方面的问题，更促进了我的写作愿望。但还是担心自己的能力，怕误人子弟，故迟迟未敢下笔。

岁月不饶人，如今写程序已不再是首要的事情了，因而决定要写这本书。关于Delphi的书籍，书市上确实很多了，从入门到提高、实战、高级、大师、深度、黑客等，以至于给书本取个名字都很难啊。干脆取个俗名吧，就叫“Delphi宝典”，就像武林江湖一样，软件开发行业中，程序员也各有自己的拿手好戏，诚然没有哪门工夫可以独霸天下，“葵花宝典”也一样(照样被独孤九剑所败)，“Delphi宝典”充其量也不过是散打水平，学好了总能抵挡一些猫脚工夫，可以行走江湖吧。后来考虑到“宝典”实在太多，便取名“用Delphi开发共享软件”，因为我自己多年来，也一直用Delphi编写共享软件并维持家庭部分开销的。

本书是一本中高级的程序设计实用书籍，以提高软件设计能力为目标，特别对有志成为共享软件的开发者很有帮助。它用专业的共享软件或已商品化的软件作为实用例子步步引导读者走入Delphi程序设计的更高境界。十多个精心挑选的应用实例，覆盖了Windows下编程的许多方面，让Delphi的已入门者，不仅感受到Delphi的强大功能，更重要的是学到程序编写的经验技巧，同时充分领略面向对象程序设计的技术优越性和强大功能。

本书主要读者对象是有志开发共享软件的在校学生和程序员，也可以作为计算机软件专业教学第二课堂的培训教材。随书所附光盘中还收集了作者多年来积累的一些第三方组件和自己开发的几十个组件，以及和本书技术相关的作者开发的商业软件。该光盘也使得本书真正物有所值。

本书编写过程中，得到了不少同志的帮助，没有他们的支持，难有此书。在此表示诚挚的感谢。疏漏和不妥之处在所难免，请读者多提宝贵意见。

吴志辉

2008.8

目 录

第1章 绪言	1
1.1 程序员开发之路	1
1.2 本书学习方法	1
1.3 Delphi 开发分类	2
1.4 对共享软件的看法	3
1.5 支持源代码开源	3
1.6 关于系统分析师	3
1.7 关于计算机语言	3
第2章 开发串口通讯程序	5
2.1 串口通讯组件的结构	5
2.1.1 事件对象的使用	6
2.1.2 接收数据线程	8
2.1.3 发送数据线程	8
2.2 在 Delphi IDE 环境开发组件	9
2.3 使用线程类 TThread	11
2.4 组件的调试	12
2.5 代码分析	12
2.6 应用程序设计	23
2.7 总结	28
习题	28
第3章 万能图形编辑器	29
3.1 功能分析	29
3.2 知识准备	30
3.3 TMyShape 源代码分析	30
3.4 在应用程序中动态创建可视组件	45
3.5 总结	52
习题	52
第4章 通用表格编辑器	53
4.1 .INI 文件的使用	53
4.2 .INI 文件的结构	54
4.3 .INI 文件的读写	54
4.4 .INI 文件的高级应用	56
4.5 .INI 文件的优缺点	60
4.6 总结	60
习题	60

第5章 逻辑梯形图编辑器	61
5.1 学习目标	61
5.2 结构化存储复合文件的结构	61
5.3 .STG 文件的读写	62
5.4 .STG 文件的优缺点	66
5.5 .INI 和 .STG 文件的性能测试比较	66
5.6 “简易逻辑梯形图编辑器”的开发	68
5.6.1 逻辑梯形图的数据结构	68
5.6.2 类 TMyLogic 的实现	70
5.6.3 类 TLGLink 的实现	72
5.6.4 类 TTestLG 的实现	73
5.7 关于逻辑梯形图路径的遍历	79
5.8 总结	80
习题	80
第6章 CMOS 数据读写程序	81
6.1 编写组件访问 IO 内存地址	81
6.2 组件源代码分析	81
6.3 TMyWinIO 的使用	86
6.4 总结	88
习题	89
第7章 多声卡音频自动播出系统	90
7.1 需求分析	90
7.2 知识准备	90
7.3 混音器组件简介	91
7.4 混音器组件源代码分析	91
7.4.1 混音器类	91
7.4.2 混音通道类	93
7.4.3 混音效果管理类	95
7.5 节目表编辑器简介	96
7.6 自动播出软件简介	99
7.7 总结	109
习题	109
第8章 报表打印与动态连接库	110
8.1 任务分析	110
8.2 如何显示 .DLL 中的窗体	111

8.3 创建包含窗体的动态连接库	112	13.4 建立简单的通讯联络方式	207
8.4 如何模拟显示报表和打印报表	115	13.5 取得远程计算机的屏幕图像	207
8.5 总结	121	13.6 取得远程计算机的进程信息	210
习题	122	13.7 改进思路	211
第9章 坦克大战游戏程序编写	123	13.8 完整代码	211
9.1 编写实时游戏程序的要求	123	习题	228
9.2 Tank War 游戏简介	124	第14章 再谈 Win32 SDK 编程	229
9.3 素材库程序编写	125	14.1 再谈使用 Win32 API 的必要性	229
9.4 地图编辑程序编写	133	14.2 混音器编程中使用 API	229
9.5 游戏主程序编写	141	14.2.1 如何确定相关 API 之间的关系	230
9.5.1 精灵基类 TSprite 的实现	141	14.2.2 正确使用数据结构	233
9.5.2 精灵引擎 TSpriteEngine 的实现	144	14.3 使用 API 的经验总结	234
9.5.3 坦克类的实现	145	14.4 用组件封装 API	235
9.5.4 主控程序编写	152	习题	235
9.6 总结	159	第15章 软件编程实例	236
习题	159	15.1 实例——工作任务管理器	236
第10章 系统编程——键盘监视	160	15.1.1 需求目的	236
10.1 任务分析	160	15.1.2 设计构思	237
10.2 建立键盘监视.DLL	160	15.1.3 主要 API 函数介绍	238
10.3 键盘监视组件的编写	169	15.1.4 主要工作流程	239
10.4 创建应用程序“键盘监视”	179	15.1.5 习题	243
第11章 服务程序编写——定时关机	180	15.2 实例2——桌面提示器	243
11.1 需求分析	180	15.2.1 需求发现	243
11.2 如何编写服务程序	181	15.2.2 设计思路	245
11.3 源代码分析	182	15.2.3 改变桌面墙纸	245
11.4 安装和使用服务程序	189	15.2.4 习题	248
习题	190	15.3 实例3——邮件群发程序	249
第12章 进程隐藏——监视系统键盘	191	15.3.1 非功能性要求	249
12.1 进程和线程	191	15.3.2 使用 Indy9 组件发送邮件	251
12.2 创建进程隐藏程序	193	15.3.3 使用多线程方式发送邮件	251
12.3 动态连接库的编写	198	15.3.4 共享软件改进建议	256
习题	203	习题	257
第13章 局域网编程	204	附录	258
13.1 学习任务	204	光盘使用说明	258
13.2 选择通讯协议	205	开发包的安装	258
13.3 获取局域网中的工作组和计算机	206		

第1章 绪 言

1.1 程序员开发之路

程序员走过的路，可能有四个阶段：学了几门编程语言后开始从事初级的编程工作；有些经验后，可依靠“技巧”进行更高些层次的编程；滚打四五年后，有些经验总结，凭借“技巧”和经验能解决一些实际问题；在实际开发过程中，不断总结领域内开发经验，系统学习设计分析理论和工具，最后达到需求捕获、分析问题、解决问题的能力，即所谓专家。这本书的读者对象应该定位在接近第三阶段的水平上。

1.2 本书学习方法

Delphi 像 Basic 一样，是一门入门容易提高难的编程语言。很多程序员便选择了用它来编写数据库应用程序，于是难度降低了许多。本书不想涉及这个领域，而是讲一些基本的“实用”技巧，对于“入门容易提高难”这一说法，可能是因为 VCL 非常完善，满足了我们大多数应用程序开发的要求，所以对 Windows 核心技术的深入和 OOAD（面向对象的分析设计）的学习，根本就难得理会了。而 VC 则有所不同，VC 程序员不得不做一些深入学习来研究 MFC，其实也就是要去学习大量的 Win32 API。如果你想成为高手，必须一样去深入学习！本书也不是一本入门书籍，所以还希望您是一个有一定入门基础的编程人员。先学好 Pascal 语法，然后读几本我国台湾编程大师李维的书。

学习是一门扎扎实实的事情，需要付出艰辛的劳动和时间。作者第一次写非可视串口通讯组件，我打印了所有不明白的 API 函数，足足有密密麻麻的 100 多页，还有 20 多页的源代码（为什么要打印出来？根据一些研究报道，人脑在阅读时，只能对最近的 3~4 页内容有较好的印象，如果您不具有非常强的记忆力的话，建议打印阅读资料，尤其是超过 100 页的资料）。作者花了足足两个月的时间来阅读理解。

“入门容易提高难”的另外一个原因，可能与教学有关。课堂上学习了 Pascal 语法和一些面向对象的编程思想，却很少有实践的机会，特别是参与到实际的应用系统的开发中，就算作了一个小课题，也是没有什么实际价值的。本书使用了十几个实用的或商用例子，每章讲一个主题，作为学习的扩展，希望展现怎样用学到的知识来完成实际的有点价值的应用。作者一直认为：“天才不多，多看、多思考、多练习，才是提高实战能力的最好途径，三者缺一不可，并且承前启后。”多看，才能思路开阔，才能发现自己知识的狭隘。多写的前提是需要多想，盲目地写，价值很低。我们常听人说：看别人的程序，不如自己重新写一个！其实不然，真正看懂和思考专家写程序的思路和方法，胜过您自己写 100 遍无聊的代码。当然，没有实际在计算机上实现“学到”的东西，最后还是不能成为自己的知识。本书的特点之一，就是在部分关键源代码中，作了详细的说明和注释，试图让读者能感受到编写代码的思路，必要时，还会绘制程序流程图，节约读者时间。另外，每章

后面，都附有几道习题。老师当多了，总想出点题目。题目难度应该在程序员水平。光盘给出了一种答案。

有关 Delphi 的书籍确实很多，但真正教会您真枪实弹地开发设计过能成为商品的软件几乎没有。我们能从每本书中吸取一点有用的知识，就已经感到很满足了。本书试图打破这种格局。本书中的许多“共享软件”，你加以改进和扩充，就可以拿来作为商品使用了。这就是共享软件！

1.3 Delphi 开发分类

“聪明的程序员用 Delphi，真正的程序员用 VC”，这句常挂在程序员口头的谬论，应该修正。语言只是一门工具。只是由于 VC 是 Windows 操作系统开发者的产品，似乎与操作系统结合紧密，好像这样就变成了真正的程序员。作为一门真正面向对象的编程语言（OOPL），Delphi 有它独到的优势；其实，微软 C# 的语法，与 Delphi 非常相似：不提倡使用指针类的单继承，异常机制（C# 首席设计师就是 Borland 的大师、Pascal 编译器创始人，即 Delphi 之父 Anders Hejlsberg；我们在 C# 中看到了许多 Delphi 的影子）。这里不讨论这些问题；只想谈一下个人对使用 Delphi 开发方向分类的看法。应该说，Delphi 在目前流行的 RAD 编程工具中，是“无所不能”的，网上也报道了用它编写驱动程序的例子。以下几个方面的应用，作者认为它应该是 Delphi 的拿手好戏。

（1）数据库信息系统的开发，特别是在多层分布式系统开发中。据称像“速达”、“用友”等许多的优秀软件就是用 Delphi 开发或改进的。Delphi 天生就是数据库信息系统开发的超级优秀工具。

（2）Web Service 应用系统。它是目前最热门的系统开发软件；Delphi 对它提供了非常好的支持，仅数数它提供的服务组件，就让人感觉它是大师级的专业工具。目前的 Delphi 2006（也称 BDS2006: Borland Developer Studio 2006）对该领域的支持，简直是天衣无缝：在 IDE 中可视化建立符合 UML 的企业逻辑模型，自动生生成服务代码框架和后台数据库。

（3）游戏程序设计。以往总认为只有 VC 才是 Windows 游戏程序编写的源生工具。可是现在大量的网络游戏，却很多是用 Delphi 开发的。甚至，许多局域网游戏，也改用 Delphi 编写了。

（4）组件及组件库设计。由于 Delphi 编写 VCL 组件或 COM 组件都非常的方便和出色，出现了专门生产组件的程序员和公司。

（5）共享软件[shared software]，即通用/专用软件的应用系统，大多为桌面应用程序。共享软件是一种先试用、后购买的软件，并不是免费软件。它的特点是实用性很强，但软件本身规模并不大。像 Windows 优化工具、迅雷、Winrar、各种媒体播放器、小型游戏程序、桌面秀等，范围非常广；这里罗列一些应用领域：计算机自动控制、电视监控、多媒体制作、教学软件、电子地图、网络工具、计算机安全系统管理、图文处理、各类解压缩算法等。

应用领域太广，似乎让人不知所措。读者应根据自己所在领域，有所重点发展方向。笔者主要从事计算机自动控制软件系统和多媒体软件的设计开发，欢迎同行一起交流心得。

1.4 对共享软件的看法

由于本书涉及共享软件开发，这里提一下本人对共享软件开发的几个观点。

(1) 不要指望一夜暴富。根据 2006 年国外共享软件网站的一些报道，目前的共享软件大约有 50% 的是用 VC 编写的，Delphi 大约占 45%。尽管共享软件作者中有不少买奔驰宝马，但大部分是很寒酸的（很多人都是业余爱好者，还不能真正靠它生存）。只有您的软件被广大用户接受，您才能致富。中国目前也有好几万共享软件作者，大部分需要好几年的时间，月收入才能达到万元。在中国，能达到 5000 元也就很好了。

(2) 不要以为软件技术高，就能卖好价钱，就会有很多用户；也不是功能越多就越能有市场。共享软件突出实用性、专业性，能很好解决某个问题就能赢得市场。很多调查表明，一些不很实用又不完善的功能，往往导致销售量下降：所谓的高版本不如低版本。

(3) 共享软件要短小精悍。很少有容量大于 10M 的软件，除非功能非常全面的一些系统。所以共享软件适合于个人开发（但也有软件公司开发共享软件的），开发周期一般在三个月左右。

(4) 软件质量和功能是生存的基本前提。可以说，您能想到的每一个领域，都已经有人在开发了。要想占有一席之地，只能靠更好的质量和功能。当然，易用性也很重要，因为共享软件的使用者大部分不是计算机专业人士，更不是软件开发人员。

1.5 支持源代码开源

虽然，程序员靠编写程序养家糊口，都不想砸了自己的饭碗，但知识是人类共同财富，我们的绝大部分知识（应该超过 95% 吧），来源于对前人知识的积累。我个人认为，Linux 的开发规则，应该值得提倡。除非是自己开发的源代码，否则都应该公布，修改的版本也一样。只是目前程序员们不愿意把自己的代码公布。尽管如此，作者还是提倡开源；本书所有章节的源代码和组件源代码都公布不漏。希望读者能发扬共享精神。

1.6 关于系统分析师

真正要达到能独立开发系统的能力，仅靠会编写程序是不够的。我个人认为，程序员应该往系统分析师的方向迈进。本书，不涉及系统分析师的广泛领域，但建议读者阅读一些系统分析师常用的关于分析设计方法的书籍；实际上，程序员都在无意识中向着系统分析师方向迈进。

1.7 关于计算机语言

计算机软件技术和语言发展迅速，让 IT 行业的程序员学习新技术感到力不从心。常常感觉到刚学习入门，还没有应用开发过像样的系统，又要花时间去学习新东西，结果一事无成。我感觉，中国的计算机程序员资格和水平考试是很务实的，掌握基本的知识，才能以不变应万变。如果您专门搞学问研究，站在技术前沿是对的，可如果您从事应用系统设计开发（就像共享软件作者一样），那千万别冒技术风险，因为那可能让您一败涂地。成熟稳定的系统才是满足用户要求的基本前提。计算机语言从汇编到 Basic，到 C/C++、

第2章 开发串口通讯程序

就算您能用 Delphi 写大型的应用程序（超过 50 万行自编源代码），但如果不能编写出自己的组件（Component）去完成一些特定功能，我个人认为，那还不够完美。Delphi 强大的组件开发功能，使得专业的组件开发商应运而生。然而，这些开发商生产的商品化组件，价格较昂贵；所以，必要时，我们自己要动手编写组件！特别是共享软件作者，本来就是要靠编写程序赚钱的，不太可能再花钱去购买第三方组件（除非技术太难做不到）。

开发组件的目的很多：①代码重用；②逻辑功能封装；③满足特殊要求。

我们的串口通讯应用程序较为简单，但主要精力却放在编写串口通讯组件上（Serial Communication Component）。许多智能仪器、仪表和高级昂贵的设备，都配置了 RS232 串口，并用它与计算机交换信息，甚至允许您对其进行控制操作。从 Internet 下载了许多相关组件，不是功能不全，就是使用不方便，还有许多“虫”，让您不敢把它应用到重要的应用程序中。而使用没有源代码的 ActiveX 控件更是不敢。于是，下决心改进它们。

Internet 网上的东西，应该都是免费的。作者遵循自由软件的原则，改进的版本一律公布源代码。本组件的源代码，您可以在光盘中找到。

在本章节，您可以学到：

- (1) 编写多线程组件；
- (2) Windows 中线程的消息处理；
- (3) 线程同步，事件内核对象的使用；
- (4) 编写组件的一般方法；
- (5) 提高学习 Windows API 的能力；
- (6) Delphi 中多线程编程的通用方法。

2.1 串口通讯组件的结构

作为编写的第一个组件，其源代码和注释介绍较多，稍显累赘。其中的英文注释为原组件作者标注（本组件借鉴了一些通讯组件的一些精华），不做改动。

组件结构：由控制界面、接收数据线程（ReadThread）和发送数据线程（WriteThread）三部分组成。组件控制界面提供设置通讯端口参数、启动通讯、发送数据、停止通讯等功能。还提供设置接收数据消息处理函数的功能。线程对象的具体功能在稍后介绍。

图 2-1 是用 UML^①描述该组件的简略类图。Delphi 的类声明，其实已经非常接近 UML 对类的描述要求，这点比起 Java 来要清晰得多，尽管 Java 提供了专门的工具来产生，Pascal

① UML 是一种适应广泛应用领域的记号系统。也是程序设计师特别是系统分析师在软件工程中使用的重要建模工具（但不是建模方法）。可以说是面向对象方法学的标准行话。如果不懂它，给人的感觉就是外行或业余爱好者，尽管爱好者也可以写出优秀的软件（但很难保证是高质量的软件）。推荐参考书：

① 《面向对象软件工程》，Bernd Bruegge, Allen H. Dutoit 著，叶俊民，汪望珠译，清华大学出版社出版；

② 《UML 用户指南》，Grady Booch, James Rumbaugh, Ivar Jacobson 著，邵维忠，麻志毅，马浩海，刘辉等译，人民邮电出版社出版。



图 2-1 通讯组件的 UML 类图及关联

语言中的先定义后使用的语法，保证了数据和类的正确使用，比 C++ 和 Java 中随意定义变量和类的自由方式要严谨得多，应该提倡。

2.1.1 事件对象的使用

可能您还没有在实际的项目中使用过 Windows 内核对象“事件”，这里开始学习它。熟悉的读者可略过。

事件对象 (Event) 是 Windows 操作系统中的同步对象，主要用来向其他线程发送信号以表示某个操作已经完成。创建事件对象的 Windows API 函数^①是：

```

Handle CreateEvent(LPSECURITY_ATTRIBUTES lpEventAttributes, // security attributes
    BOOL bManualReset, // flag for manual-reset event
    BOOL bInitialState, // flag for initial state
    LPCTSTR lpName); // pointer to event-object name
  
```

各参数的意义^②如下。

lpEventAttributes: 指向 a SECURITY_ATTRIBUTES 数据结构，它决定了能否被子进程继承。为 nil 时，返回的事件对象句柄不能被继承。

bManualReset: 指明事件对象是人工复位还是自动复位。如果为 TRUE，我们必须使用 ResetEvent 函数来手动复位到无信号态 (State of Nonsignaled)。如果为 FALSE，当等待信号的线程释放时，Windows 自动复位事件对象到无信号态。

bInitialState: 事件对象的初始状态。如果为 True，创建时为信号态。

lpName: 指向事件对象的名字。可以没有。如果有，大小写是有区别的。如果名字和已存在的信号量 (Semaphore)、互斥 (Mutex) 或事件 (Event) 的名字相同，函数调用失败。使用 API 函数 GetLastError 将返回 ERROR_INVALID_HANDLE。因为这些内核对象都共享

① Windows API 函数介绍时，使用 C 语言的语法规则。在源代码介绍时，使用 Object Pascal 语法。

② 提高学习 Win32 API 能力的方法之一就是要透彻理解函数各参数和返回值的意义。MSDN 提供了详细的开发资料，也是 Windows 下程序设计的基础，MSDN 涉及了 Windows 下编程的方方面面：图形图像处理、文字处理、消息、窗口、网络接口、通讯、多媒体等。实际上，正确使用 MSDN 已经是衡量一个程序员实际能力的标志之一。

同一命名空间。

函数成功调用，返回创建事件对象的句柄。

我们的组件中创建了如下四种事件对象。

(1) 关闭通讯事件

`hCloseEvent := CreateEvent( nil, True, False, nil );` // 人工复位的无信号态事件对象

使用 Windows API 函数 `SetEvent( hCloseEvent)` 来激活信号。读写线程检查到其信号态时，立即关闭通讯。

(2) 读数据事件

`overlappedRead.hEvent := CreateEvent( nil, True, True, nil);` //信号态事件对象

当串口有数据到达时，该事件对象激活到信号态。程序中用自定义函数 `SetupReadEvent` 来设置读数据处理过程。

(3) 处理通讯事件

`overlappedCommEvent.hEvent := CreateEvent( nil, True, True, nil);`

程序中用自定义函数 `SetupCommEvent` 来设置通讯处理过程。

(4) 处理发送数据事件

`overlappedWrite.hEvent := CreateEvent( nil, True, True, nil );`

事件对象任何时候都处于有信号态(Signaled)或者无信号态(Nonsignaled)之一。使用 `WaitForSingleObject` 或 `WaitForMultipleObjects` 函数来等待事件对象。当有信号时，操作系统会唤醒等待它的一个线程。在等待到有信号态后，线程会自动将其设置为无信号态，其他线程需继续等待。这里介绍 `WaitForMultipleObjects` 的使用：

当任何一个内核对象有信号或规定的时间等待完毕时，该函数返回，否则一直阻塞等待。

```
DWORD WaitForMultipleObjects(  
    DWORD nCount,                // number of handles in the object handle array  
    CONST HANDLE *lpHandles,     // pointer to the object-handle array  
    BOOL bWaitAll,               // wait flag  
    DWORD dwMilliseconds );      // time-out interval in milliseconds
```

各参数的意义如下。

① `nCount`: 指明参数 `lpHandles` 中内核对象的数量。最多为 `MAXIMUM_WAIT_OBJECTS`。

② `lpHandles`: 内核对象数组。可以包含不同类型的内核对象。

③ `bWaitAll`: 等待类型。如果为真 (TRUE)，只有当所有的内核对象都为信号态时，函数才返回；否则，任何一个对象为信号态时，函数就返回。返回的值就指明了处在信号态的内核对象。

④ `dwMilliseconds`: 指明函数等待的超时时间 (ms)。只要等待时间一过，不管 `bWaitAll` 参数如何，函数返回。该值为零时，函数检查各对象的状态后立即返回。如果为无穷大 (INFINITE)，则只有对象有信号时才能返回 (与 `bWaitAll` 参数有关)，否则一直阻塞等待。

返回值 (Return Values):

① 调用失败，返回值为 `WAIT_FAILED`。可使用 `GetLastError` 获取详细错误信息。

② 成功调用，返回值指明了信号态的对象位置。

a. `WAIT_OBJECT_0` to (`WAIT_OBJECT_0` + `nCount`-1): 如果 `bWaitAll` 为真，所有对象都为信号态。如果 `bWaitAll` 为假，返回值减掉 `WAIT_OBJECT_0` 就是信号态对象在

lpHandles 数组中的索引位置；如果同时有多个对象在信号态，则返回值是最小索引位置的那一个。

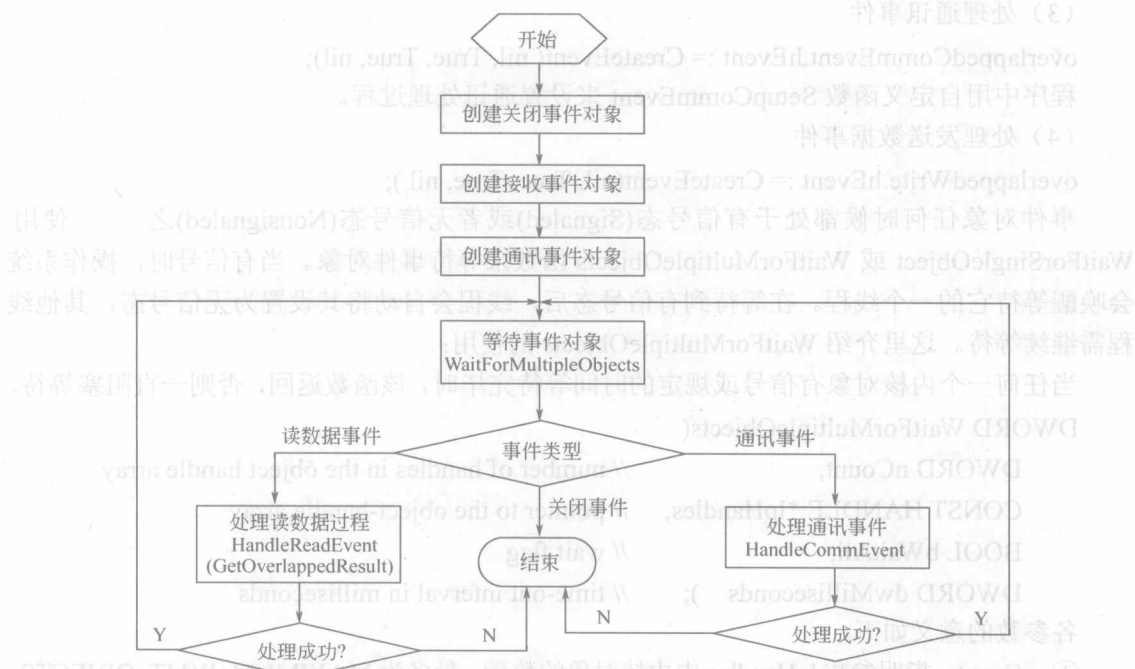
b. 如果 bWaitAll 为真，所有对象都为信号态，并且其中至少有一个是被遗弃的互斥对象。如果 bWaitAll 为假，返回值减掉 WAIT_OBJECT_0 就是信号态对象在 lpHandles 数组中的索引位置；如果同时有多个对象在信号态，则返回值是最小索引位置的那一个。

c. WAIT_TIMEOUT：没有满足 bWaitAll 的条件，并且等待时间超时。

具体使用例子，在代码介绍部分做详细注释性说明。

2.1.2 接收数据线程

使用 WaitForMultipleObjects 函数来监视是否有接收通讯数据事件的发生，如果有，把数据读到内存，并调用数据处理函数处理（如果指定了处理函数）。下面是其主要流程^①（即 ReadThread 的执行函数 Execute）：



其中处理读数据过程的流程如下：



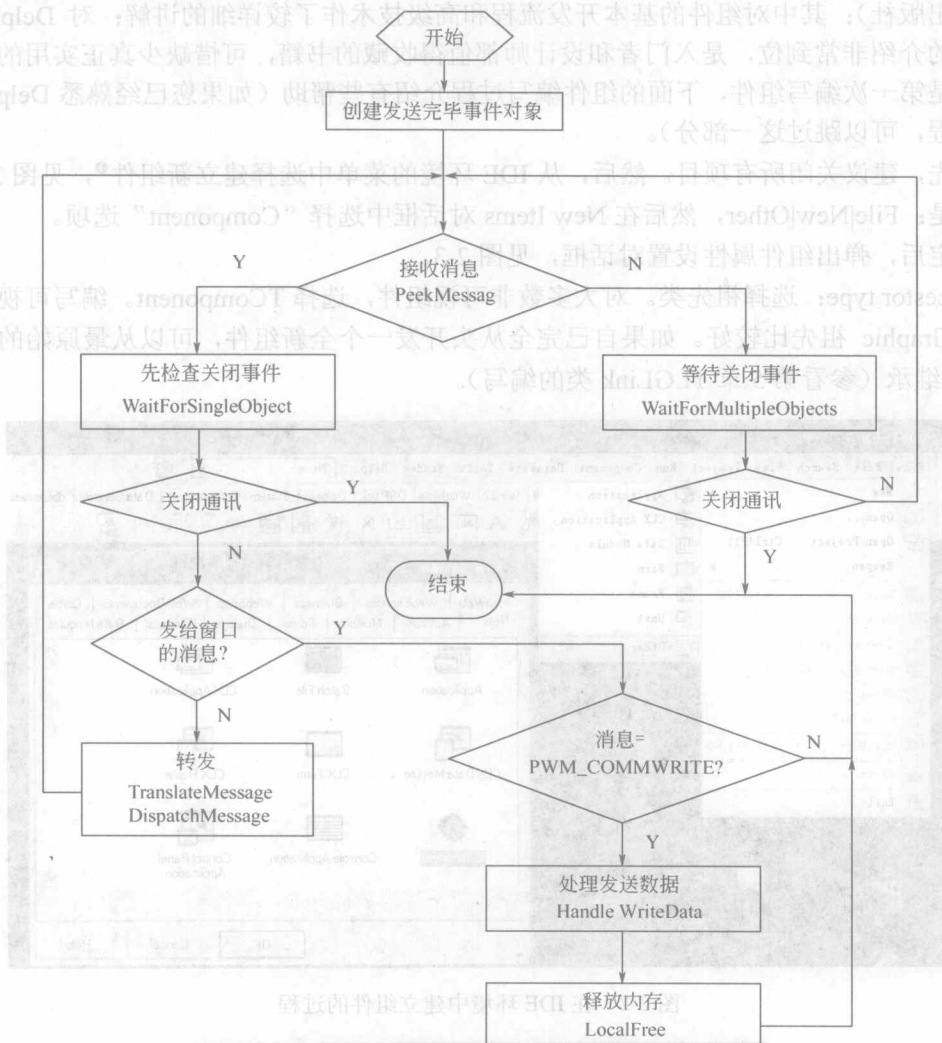
在这些过程中，使用了大量的 Win32 API 函数。在上面的流程图中，我们指明了过程的函数名，并列出了其中使用的主要 API 函数，方便读者在阅读程序代码时把握主要方向。

2.1.3 发送数据线程

发送数据是调用 WriteCommData 函数来主动请求数据发送的，处理流程和接收数据不一样。WriteCommData 函数是通过往“写线程”发送消息的方式来完成发送请求的，数据被包含在消息数据中。“写线程”接收到自定义的发送数据消息 PWM_COMMWRITE 后，处理消

① 程序流程图仍然是局部编程使用的主要方法，不管你是使用结构化编程思想还是面向对象编程思路。

息中的数据，具体的主要流程如下：



其中最核心过程是发送数据函数 `HandleWriteData`，主要是调用 API 函数 `WriteFile` 来完成发送数据。但由于 Windows 是多任务操作系统，中途可能出现“意外”：如程序结束、通讯关闭或中途被其他线程中断。必须不断调用 `GetOverlappedResult` 来判断已经发送的字节数，没有发送完毕的话，需要继续调用 `WriteFile` 来完成发送，直到所有数据发送完毕。阅读源代码时，请读者最好画一下简单的流程图。其中有一个无限循环的过程。

到此，读者或许明白了，为什么要采用线程来完成通讯任务的好处了。不管接收还是发送大量的数据，应用程序都不会被阻塞“死机”。

2.2 在 Delphi IDE 环境开发组件

编写组件，对初学者来说，是一项激动人心的事。组件开发可以独立出来，已经作为一个领域在发展。对于软件工程来讲，特别是 RAD 开发，构件重用是提高软件企业竞争力的核心技术之一，要注重长远利益，重视构件（组件）的开发。实际上，由于 Delphi 对组件开发的完美支持，设计难度并不是我们开始想象的那样困难。虽然复杂，却有章可循。如果您

从未开发过组件，强烈建议您先阅读一本很有价值的书籍：《参透 Delphi/Kylix》（乔林著，中国铁道出版社）；其中对组件的基本开发流程和高级技术作了较详细的讲解；对 Delphi 的各个方面的介绍非常到位，是入门者和设计师都值得收藏的书籍，可惜缺少真正实用的例子。如果您是第一次编写组件，下面的组件编写过程介绍有些帮助（如果您已经熟悉 Delphi 组件开发过程，可以跳过这一部分）。

首先，建议关闭所有项目。然后，从 IDE 环境的菜单中选择建立新组件^❶，见图 2-2。菜单路径是：File|New|Other，然后在 New Items 对话框中选择“Component”选项。

确定后，弹出组件属性设置对话框，见图 2-3。

Ancestor type: 选择祖先类。对大多数非可视组件，选择 TComponent。编写可视组件，选择 TGraphic 祖先比较好。如果自己完全从头开发一个全新组件，可以从最原始的祖先类 TObject 继承（参看第 5 章 TLGLink 类的编写）。

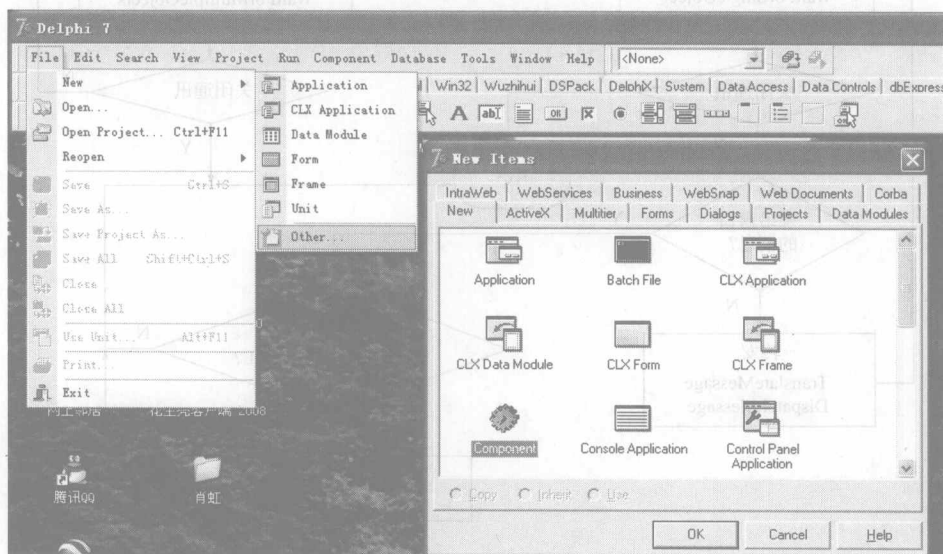


图 2-2 在 IDE 环境中建立组件的过程

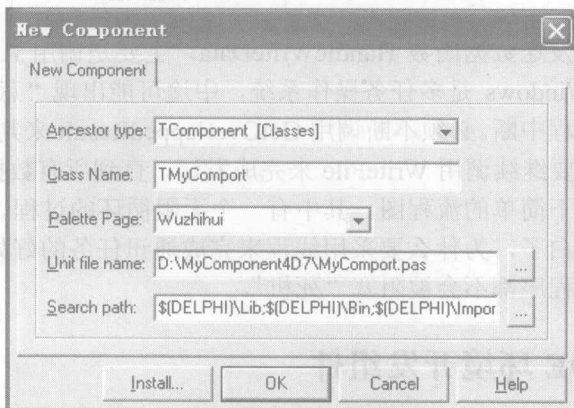


图 2-3 组件属性设置对话框

^❶ 当然，如果您对 Delphi 各种文件的意义了解很清楚的话，直接建立一个单元文件(.Pas)也可以，在单元文件中直接编写组件代码。

Class Name: 输入类名。最好取一个能代表功能或意义的名字，尽管以后还可以修改。

Palette Page: 如果组件安装到 IDE 环境，指定安装的组件面板名字。建议：自己开发的组件统一放在自己命名的面板上。

Unit file name: 组件对应的 Pascal 源文件。建议：自己开发的组件统一放在一个目录下。

Search path: 如果您的组件源文件放置的目录不在该指定的目录列表内，按“...”填加该路径。否则，安装到组件面板后，无法使用：系统提示找不到该组件。

Install: 马上安装组件到您指定的组件面板上，方便我们在 IDE 下测试。

填写完毕。Delphi 向导根据您刚才填写的数据自动生成一个源文件框架，见图 2-4。

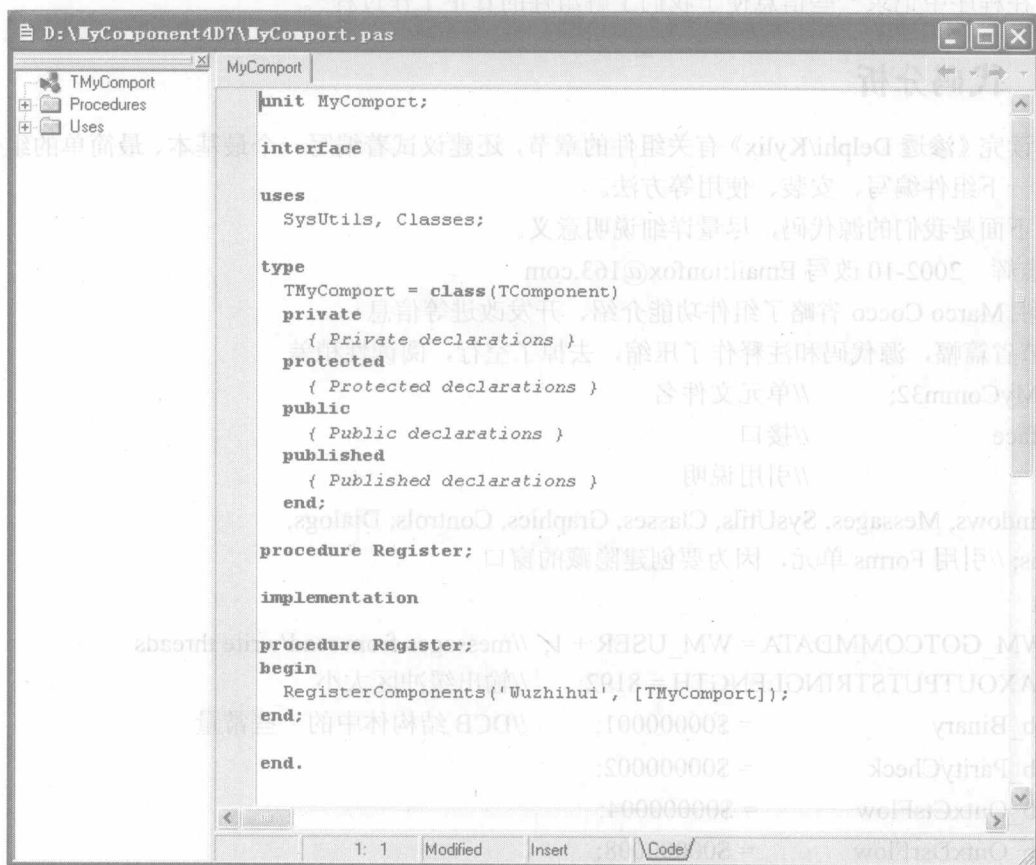


图 2-4 Delphi 向导自动生成的源文件框架

可以看出，它是一个“纯”单元文件，无对应的.dmf 表单文件。它是一个文本文件，我们可以任意修改它，只要符合 Pascal 语法规则就行。剩下的事情就是我们往其中填加代码了。

注：尽管 Borland 公司提议，一个单元文件最好只实现一个类；但实际上，为了方便，我们可以在其中实现多个类，并且非常常见。在后面的源代码介绍中，我们实现了 3 个类。

2.3 使用线程类 TThread

另外，对 Delphi 封装的线程类 TThread 有所认识。Delphi IDE 环境下，提供了建立线程对象的向导（与刚才介绍的组件建立向导相似）；并且提供了一个在线程中进行数组排序的例