电脑编程技巧与维护月刊

2001 年第 6 期　　总第 84 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
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

网上征订：
http //www.8848.net
http //www.gotoread.com
http //www.dangdang.com

2001 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 年合订本 98 元（包括挂号邮寄费）

（本刊在今年年底推出 2000 年全年 12 期源代码光盘，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　2000 年单本：7.80 元/期

2001 年单本：7.80 元/期

（凡订阅 2001 年全年的用户，随 2001 年第 12 期刊物赠送 2001 年全年源代码光盘。）

同期软盘　1998 年：20 元/季；1999 - 2001 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. 2001 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. 本刊光盘不单独出售。
4. 需要开发票者，请在汇款附言栏中注明。
5. 凡在杂志社订阅者可享受 10%的优惠。
6. 网上征订请查询：http://www.8848.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

诚聘 IT 人才

北京飞天诚信科技有限公司成立于 1994 年，是一家高速发展的股份制高新技术企业。公司现有员工 40 余人，公司员工中 40% 是技术研发人员。公司长期致力于软件加密、信息安全产品的研制与开发，并成为这一领域内的佼佼者。由于业务发展的需要诚聘如下人才：

软件开发工程师

1. 能非常熟练地使用 VC、C + + 能独立完成完整的软件编程。 编程经验 5000 行以上）

2. 精通或对 32 位汇编有一定程度了解，独立编写过 WIN32 下的设备驱动程序者优先考虑。

3. 对 IC 卡编程及读卡器设计和 PKI 体系结构比较熟悉者优先考虑。

4. 有在 MACOS 下开发经验的优先考虑。

5. 对软件加密、网络安全有较深程度了解，并对其工作有兴趣和热情者。

学历、性别不限、提供宿舍及面试费用。确有真才实学。

硬件开发工程师

有多种单片机开发能力和经验，熟练掌握汇编、C 语言等，并具有电路设计能力；编写过 32 位下的驱动程序，有 USB 或 IC 卡硬件开发经验者优先考虑。本科以上学历，计算机或电子专业毕业，年龄 30 岁以下。

硬件维修员

技术要求：熟悉数字电路、模拟电路和单片机系统。工作任务：维修公司开发的硬件产品 .

学历不限、对其工作有兴趣和热情者 .

销售工程师

要求：计算机本科学历，2 年以上软件编程经验、熟悉多种编程语言；良好的沟通、协调技巧，良好的语言表达能力。

办事处主任 广州、上海

负责指定区域客户的市场销售、推广和服务并负责办事处的日常运营了解客户需求；为客户提供技术服务；技术培训。

任职条件：大学本科，计算机或电子类专业；熟悉常用编程语言及开发工具；英语四级，熟练的读写能力和一定的听说能力；耐心、细致，对工作认真负责；良好的人际沟通能力和团队协作精神。

单位地址 :北京市海淀区学院路蓟门饭店五号楼三层 100088

Tel 010 - 82081138. FAX 82070027. www. FTsafe. com

Email FEITIAN@PUBLIC3. BTA. NET. CN 联系人 :黄先生

Oracle 和 Sun 共创世界首个 3TB TPC - H 基准测试新纪录

世界最大的电子商务解决方案供应商 Oracle 公司和网络计算领袖 Sun 公司日前共同发布了世界第一个 3TB 级的 TPC - H 基准测试结果，达到了 10 764. 7QphH@3000GB，相应的性能价格比为 1222 美元 /QphH@3000GB。这个 3TB TCP - H 基准测试的规模和范围已经大大超过了过去由 IBM 保持的 1TB TPC - H 基准测试的记录，创下了 TPC - H 基准测试新的世界纪录。

这一创纪录的基准测试结果是在 2 台 Sun Enterprise 10000 服务器 也称为 Starfire 上运行 Oracle9i 数据库实现的，采用 Solaris 8 操作环境，在位于俄勒冈州 Beaverton 市的 Sun 大规模计算中心进行的。TPC - H 基准测试程序是模拟检查大量数据、执行高复杂度查询的决策支持系统，并模拟一个系统有效执行特别复杂的查询能力，测试系统快速响应的能力。

SmartLink 公司最新推出兼容 Linux 的 modem 插卡

日前，世界领先的通讯解决方案提供商斯玛林 SmartLink 公司宣布其 SmartRISER56 tm 系列产品已可与 Linux 系统兼容。至此，全球范围内的 Linux 用户将能享受到 Smart Link 公司 SmartRISER56 技术的良好稳定性及高性能。Smart Link 公司产品线涵盖包括 AMR、CNR 和 ACR 在内的各种技术标准，与 Linux 系统的兼容将有助于进一步扩大市场，获得更大的市场份额。在 PC 市场上，internet 应用是不断增长的一部分，而 Linux 是这一部分用户优先选用的操作系统。Linux modem 解决方案将为这一特定的对价格比较敏感的市场提供了高可靠性而低价格 Internet 接入方式。

Sun 公司推出分布式计算软件

日前，Sun 微系统公司宣布推出对等网络 peer - to - peer 软件或称分布式计算 distributed computing 软件，尽管该公司在公布这一消息时有些低调，但这件事表明就连超大型企业也在涉足分布式计算领域。这家目前正在与 IBM 公司展开激烈的服务器大战的计算机硬件生产商早在今年 2 月份就宣布要实施一项名为 “Project JXTA” 的计划 JXTA 的发音为

“juxta”，其目的是使人们更加容易地接入“扩容后的网络”上的同类及资源。这个基于 Java 平台的 JXTA 软件可以使多个网络和不同平台之间的信息采集更加顺畅。这表明从未涉足过对等计算领域的 Sun 公司已改变其经营思路。对等技术的理念是把网络基础设施和用户设备的计算能力加以汇合，使它们的资源得到充分的利用。

微软宣布 5 月 31 日上市“Visio 2002”

日前，微软宣布，绘图软件“Visio 2002”已进入产品制造阶段 RTM：release to manufacturing 。Vision 本来是微软于 2000 年 1 月份收购的美国 Visio 的产品。Visio 2002 将成为微软公司开发的第一个版本。该公司已于 3 月 6 日发布了该软件的 β 版。Visio 2002 的特点是与 WWW 的整合、与“Microsoft Office”等其它微软公司产品的紧密结合以及对业界标准的全力支持。由于 Visio 2002 支持 XML Extensible Markup Language 和 COM Component Object Model 插入，所以 ISV 及解决方案提供商使用基于 Visio 2002 的开发平台就可以在短时间内轻松完成客户定制产品。

中国推出第一台卫星导航掌上电脑

中国第一台卫星导航掌上电脑近日在“海尔科技节”上首次亮相。中国最大的家电企业海尔集团今天在此间首次推出具有自主知识产权的中国第一台卫星导航掌上电脑。这是目前中国唯一能把大容量电子地图数据装进掌上电脑，并能够投入实际应用的卫星导航掌上电脑。据称，这种掌上电脑集 GPS、GIS、数据库和掌上电脑技术于一体，解决了在掌上电脑环境下运行 GPS、GIS 软件和处理大容量电子地图数据等国际公认的技术难题。

Adobe Acrobat 5. 0 加强安全功能

Adobe Acrobat，这种流行软件的最新版本可为内容创作者提供更多的安全性能，并可对网页文档进行操控。对于大多数人来说，他们认为 Adobe Acrobat 就是一种免费的程序，可以使你在网页游览器上显示出超格式化文档。并且它可以对可移植文档格式 PDF 进行精确输出打印。但是要使用 Acrobat 的全版本，你就必须得支付费用，这对于文档分类来说是一种功能强大的商用工具，可以在公司内办公使用，也可以用于个人用户。最新的 Acrobat 5. 0，可安装在以 windows 为操作系统的计算机和苹果公司生产的麦金托什机上，但从早期版本上升级到这个最新版，这两种不类型的计算机用户需要各交纳 220 美元和 89 美元。这个最新版本增加了一种新的分类功能。主要是给文档创作者对他们的工作有了更多的可控性，有了它，这些用户就可以将电子文档更容易地贴到互联网上。

索尼将推出 Linux 版 PS2 游戏机

Linux 是时下最流行的操作系统，而 PlayStation2 则是最受欢迎的游戏机。在许多用户看来，如果能将这两者融为一体将是最好不过的事情。如今，他们的这个愿意终于可以实现了。近日，索尼公司在其日本网站上张贴了有关 PlayStation2 - Linux 测试版即将发布的信息，并透露其价格将为 25，000 日元，约合 200 美元左右。据悉，PlayStaion2 - Linux 将包括一个 DVD 软件、40GB 的硬盘以及键盘和鼠标等。其测试版将于今年 6 月份问世。在此之前，曾有 6000 多名用户于今年初联名上书索尼公司，希望该公司能尽快实现这个计划。

IBM 采用纳米材料制成分子大小的晶体管

据报道，IBM 的科学家使用只有几个分子大小的新材料制作出极小的晶体管，这使得制造出比今年更快和更小的计算机成为可能。这种新材料名为碳纳米管，在未来，它制造的晶体管可以只有现在的五百分之一大。虽然 IBM 公司对其投入商用表示谨慎态度，但许多科学家都认为这是一个巨大的发现，并称其未来计算机的发展迈出了重要的一步。

飞利浦推出世界上内置电源式最薄的高精度液晶显示器

近日，飞利浦在欧洲隆重推出了两款最新的顶尖 15 英寸商用型和专业型平板液晶显示器。电源内置于 61 毫米的底座，无疑使这两款显示器成为世界上内置电源式最薄的显示器。另外，其嵌入式 AC 电源免去了顾客在安装和连接上的麻烦。在诸多革新的基础上，飞利浦这两款液晶显示器的诸多可互换的特征为用户提供了最灵活的应用模式，针对不同客户的不同需求做出了各种选择。

C + + Builder 系列讲座（一）

夏祖柱

这一讲，以导师信息管理系统中部分内容为例，介绍了 C + + Builder 编程的基本概念、基本技术，然后讲述了数据库编程和报表制作的技巧与方法，力图通过这一讲能让读者掌握 C + + Builder 编程的基础知识，并能进行简单应用系统的开发。

第一讲 基础编程篇

一、C + + Builder 应用开发概述

Borland C + + Builder 是 Borland 公司继 Delphi 之后推出的又一个极优秀的应用开发工具。它天才地实现了 C + + 语言和可视化开发的完美结合，从而既具有可视化随心所欲、快速高效开发的优点，又兼顾了 C + + 的灵活与强劲，并且还将二者的功能进行了极大的扩展和提高。它的出现使得开发人员不必在高效的底层控制与轻松的程序设计间作出艰难的抉择。新版本中组件编程技术的引入，更使 C + + Builder 大放光彩，成为当今应用开发工具中最优秀的一员。本文介绍均以 C + + Builder5 为准。

1. C + + Builder 的主要特点。

1）全新的 IDE（集成开发环境），用户界面更友好、更易于使用，包括良好的代码编写环境。

2）更强的调试功能，支持多线程和远程调试功能。

3）增强的 VCL 功能，对原 VCL（可视化组件库）作了补充与修改。

4）对已有的 C + + 资源有较好的支持，改进了对 Borland C + + OWL 和 Microsoft MFC 的兼容性。

5）增强的数据库功能，提供了支持 Oracle、SQL Server 等各大型数据库的高速驱动程序，新版本中 ADO Express 与 InterBase Express 组件的引入，极大地改进与增强了数据库的访问功能。

6）快速开发 Web 及 Internet 应用程序。

7）出色的分布式计算开发，对 COM/COM + 、DCOM、CORBA 都有较好的支持。

2. C + + Builder 与 Delphi 和 VC + + 的比较

有人说“C + + Builder 是 Delphi 的 C + + 版本”。这种说法有一定道理，因为 C + + Builder 实现可视化的核心是 Delphi 的 VCL 可视组件库，而 VCL 又确为 Object Pascal 编写。但两者是有本质差别的，C + + Builder 是真正的 C + + 语言开发工具，C + + 语言的优势使 C + + Builder 在很多方面比 Delphi 更出色。应该说，C + + Builder 的应用领域较 Delphi 更加广泛，能力更强。

不可否认，VC + + 在开发应用程序方面具有强大的功能，它对 C + + 的支持和指针的灵活应用是其他开发工具所不可比拟的，在底层控制和服务器端开发方面也是最佳的选择。但是，难以想象具有大量前端界面的客户程序或很少涉及底层的一般应用系统用 VC + + 开发，将会花费多少精力与时间，而用可视化开发工具 Delphi、C + + Builder 将会极大提高开发效率，实现的效果不会丝毫逊色。

3. 三种应用程序

用 C + + Builder 开发的应用程序有三种类型：多文档应用程序、多窗体应用程序、多页面应用程序。所谓多文档应用程序（MDI 程序）是指类似 MS - Word2000 一样的程序，它的一个主窗体中包括多个子窗体，子窗体可以共用主程序的菜单，也可以有自己的菜单。多窗体应用程序是指一个窗体中包含多个独立的窗体，窗体之间无主次之分，并且可以在屏幕上任意移动，一般我们开发的都是多窗体应用程序。多页面应用程序只要一个窗体，但窗体上有多个不同的活页，每个活页上可以有自己不同的组件和内容。

根据这三种应用程序的不同特点，我们在开发时可以选择合适的程序类型来满足需要。多文档应用程序的实现要将父窗体的 FormStyle 属性设置为 fsMDIForm，将子窗体的 FormStyle 属性设置为 fsMDIChild，多窗体应用程序的实现则要将所有窗体的 FormStyle 属性都设成 fsNormal，而多页面应用程序是用 PageControl 组件来实现的，该组件可在 Win32 组件栏中找到。

4. VCL 探讨与界面开发

进行可视化编程不可不了解 VCL（可视组件库）。VCL 是一个封装了丰富的 Windows 编程元素的类库，VCL 的出现使程序界面的设计真正变成为一件“所见即所得”的事。每一种开发工具都提供了上百种标准组件，如窗体元素、对话框等，这样，只需向窗体（Form）中添加各组件，就可开发出所想要的界面。添加组件的方法很简单，只要在组件面板上单击需要的组件，然后在窗体的相应位置单击鼠标即可。

下面大家会看到，我们的实例用到了大量的 VCL 组件。这里我不打算逐一讲解每一个组件的特性及用法，不仅是因为篇幅不允许，而且组件只有在使用中才能真正理解掌握。通过组件在下面所讲的各实例中的应用，您将会逐渐掌握各常用组件的用法。

5. C + + Builder 集成开发环境简介

C + + Builder 集成开发环境主要包括五个部分：

●窗体 (Form) 窗体是整个程序设计的焦点，是包含其他组件的组件。

●组件面板 (Component Palette) 组件面板中根据组件特性分门别类地放置了多种常用组件。

●对象检测器 (Object Inspector) 用以改变组件对象的属性值和事件行为模式。

●程序编辑器 (Code Editor) 存取及编辑本项目包含的所有程序文件。

●加速条 (SpeedBar) 提供按钮加快程序的操作。

各部分如图 1 所示：

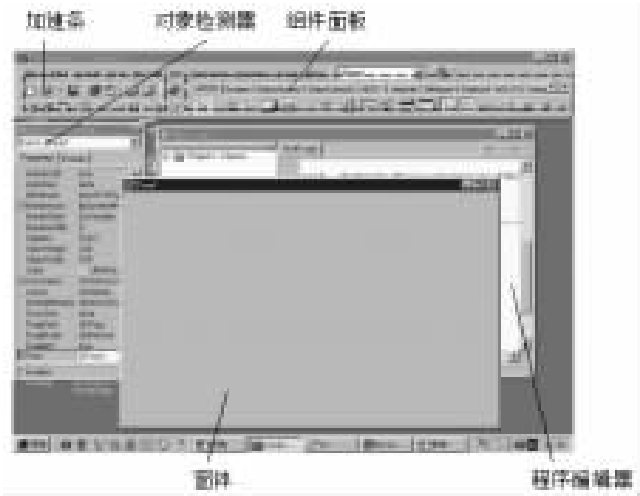


图 1 C + + Builder 集成开发环境

二、C + + Builder 编程基础技术

1. 项目的创建与保存

C + + Builder 中是以项目来组织程序的，一个项目包括多个程序单元。C + + Builder 中也只能运行项目，所以我们的程序单元只有放入一个项目中才能运行。一般打开 C + + Builder 开发工具，就自动新建了一个项目，默认名为 Project1。也可以通过 “File ” 菜单中的 “new application ” 命令来新建一个项目。

项目的保存，选择 “File ” 菜单中的 save、save all 命令都可以，在保存项目的同时，也将提示你保存程序单元。

2. 窗体的创建与关联

打开 C + + Builder 开发工具，在自动创建新项目的同时也自动创建了一个新窗体 Form1，如果我们要在当前项目中添加一个新窗体，选中 “File ” 菜单中的 “New Form ” 命令即可。

窗体的创建有两种方式：自动创建和动态创建。自动创建的窗体，在程序初始化时就创建并分配了内存空间，直到整个程序运行结束它才释放内存。动态创建的窗体，就是到使用时才创建，用完之后就释放资源。窗体创建的方式可通过 “项目选项 ” (Project Option) 对话框中的 Form 选项卡来进行设置。选择 Project 菜单中的 Option 命令就可打开 “项目选项 ” 对话框。

根据不同的创建方式，窗体之间关联方式相应有两种。例如有两个 Form：Form1 和 Form2，现在要单击 Form1 上的 Button1 组件创建 Form2，则除了在 Form1 的程序文件 Unit1. cpp 的头部要加入 “#include “Unit2. h ” ” 之外，根据 Form2 不同的创建方式，在 Button1 的 OnClick 事件中还应写入如下程序代码：

方法一： Form2 - >ShowModal ( ) ; / / 直接使用 showModal 来创建 Form2

方法二： TForm2 * form2 / / 动态创建 Form2

```
form2 = new TForm2 ( Application )
form2 - >ShowModal ( )
delete form2
```

3. 项目的运行与调试

项目的运行，用鼠标单击加速栏上的 “Run ” 按钮即可。也可通过选择 “Run ” 菜单中的 “Run ” 命令来执行。项目的调试有多种方法，比较常用的是设置断点、单步执行和跟踪调试。分别通过选择 “Run ” 菜单中的 Add Breakpoint、Step Over、Trace Into 来进行相应的调试。

4. 组件的属性设置与事件触发

我不介绍每一个组件的具体用法，但在如何灵活使用组件上还是有一定方法可寻的。当我们将组件拖到窗体上后，可仔细审查对象检测器 (Object Inspector) 中该组件的每一个属性 (在 Properties 栏中) 和事件模式 (在 Events 栏中)，根据其英文名大致可知每一个属性或事件所代表的意思。往往通过这样的观察，可以找到实现我们所需功能的方法。也可以在代码的编写中，根据 Code Completion 工具所列出的组件的可用属性和方法来进行选择。为了激活该工具，只需输入对象名如 Label1，然后键入 “- >” 或 “. ”，等待片刻即可。

三、数据库编程

之所以迅速切入数据库编程，是因为在一般应用系统如 MIS (信息管理系统) 中，数据库编程是核心，掌握了数据库编程，再加上简单轻松的界面开发，那么我们就能开发一般简单的应用系统。这里我以一个导师信息管理系统为例，来讲解各知识点。

1. 数据库表与数据库别名的创建

1) 数据库表的创建

C + + Builder 中数据库表的创建和修改是通过 DataBase Desktop 来完成的。步骤如下：

①选择 “Tools ” 菜单项中的 “Database Desktop ” 命令，打开该工具。如图 2 所示。

②在“Database Desktop”中选择“File”菜单的“new”命令，然后再选择“Table”选项。

③在弹出的对话框中选择表的类型，默认的表类型是Paradox7。

④选择好表类型后，按“ok”按钮，则出现表结构设计窗口，设计表结构。

⑤保存表，按“save as”键。

如果我们要添加或修改记录，则可用其菜单命令打开一个表，再点击工具栏上的“Edit data”按钮即可。这里我们创建了四个表：导师基本情况表、论文表、在研项目表、教学情况表，并将四个表都保存在路径为 c:\test\data 的文件夹下。

图 2　Database Desktop 窗口

2） 数据库别名的建立

C++ Builder 中对数据表的访问是通过数据库别名来实现的。数据库别名代表了数据表的访问路径，在访问数据表时，我们只需设置其数据库别名即可，而无需输入全部路径。数据库别名的建立有三种方式：利用 BDE Administrator 工具、使用 DataBase Desktop 工具和 ODBC Administrator。这里我介绍第二种方法。步骤如下：

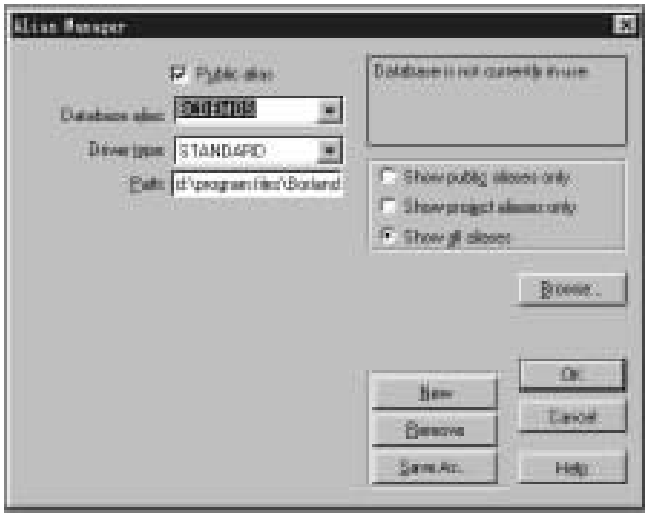


图 3　　别名管理器对话框

①在“Database Desktop”窗口中，选择“tools”菜单中的“Alias Manager”命令。弹出图 3 的别名管理器对话框。

②单击“new”按钮，新建数据库别名。

③在“Database alias”列表框中输入你想要建立的数据库别名。这里我们输入 dshxxk 作为数据库别名。

④选择数据库驱动程序。如果是 Paradox7 数据库，则就是默认的 STANDARD。我们选择 STANDARD。

⑤在“Path”文本框中输入数据库表的路径。这里我们输入 c:\test\data 即可，也可通过“browse”按钮来选择路径。

⑥单击“keep new”按钮保存数据库别名。

2. 基本数据库组件

基本的数据库组件包括数据库访问组件和数据库控制组件两大类，这两类组件封装了 C++ Builder 的所有数据库功能，用户只需略加设置就可以实现对数据库的访问和编辑。

1） 数据库访问组件

数据库访问组件在 DataAccess 组件栏中，它们是不可见的，主要功能是实现程序与数据库之间的数据交换，连接着数据库和数据库控制组件，起着一种数据交换通道的作用。常用的数据访问组件有 DataSource、Table 和 Query，这三个组件基本上能满足绝大部分开发的要求。

2） 数据库控制组件

数据库控制组件在 Data Controls 组件栏中，主要用于显示和编辑数据，是直接与用户发生联系的组件，也是设计用户接口中最重要的部分。它首先必须与数据库访问组件发生联系，通过数据库访问组件传递消息操纵数据库表。

我们要实现的导师信息管理系统中有一个核心的窗体，窗体名为 dsjbqkform，如图 4，这个窗体通过一些标准组件和基本数据库组件的应用将导师基本情况表的数据显示出来了。

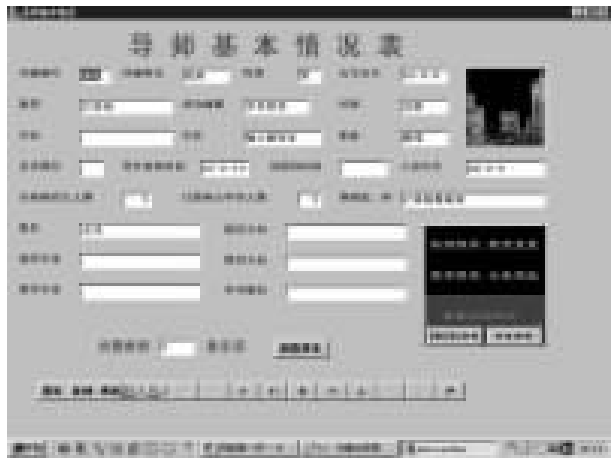


图 4　导师基本情况窗体

要实现这个窗体，首先需要往窗体中添加组件。窗体中各主要组件及属性见下表：

组件	属性	值
Label1	Caption	导师编号
DBEdit1	DataSource	DataSource1
	DataField	导师编号
DataSource1	DataSet	Table1
Table1	DatabaseName	dshxxk
	TableName	导师基本情况表
	Active	True
Query1	DatabaseName	dshxxk
	Active	False
DBimage1	DataSource	DataSource1
	DataField	导师照片
DBNavigator1	DataSource	DataSource1
Dbmemo1	DataSource	DataSource1
	DataField	教育背景

注：另外的一些组件如 label2、Label3……DBEdit2、DBEdit3…… 与 label1 和 DBEdit1 的设置方式一样，除属性值不同外，其他的组件设置比较简单，与我们所讨论的主要组件关联不大，故不一一介绍。

小技巧：导师基本情况表有 20 多个字段，要将这些字段值显示在屏幕上也需要 20 多个相应的 Label 和 DBEdit 组件，并要一一进行设置，很烦琐。有一种方法我们可一次性将 20 多个字段相应的 Label 和 DBEdit 组件拖到 FORM 上，且相应属性值也设置好了。步骤为：选中 Table1 - >单击鼠标右键 - >选中 “Fields Editor” 命令 - >在弹出的字段编辑器中单击右键 - >选择 “Add all fields” 命令 - >在对话框中选中所有字段拖到窗体上。

此时，如果我们运行程序将会看到导师的基本信息已显示在屏幕上，且能利用 DBNavigator1 组件来对数据表中记录进行添加或修改。

3. 几种查询方式的探讨

毫无疑问，查询是数据库应用系统开发中最常用的功能之一，我将作重点介绍。在 C + + Builder 中可以通过三种方式来实现查询功能，即利用 TTable 组件的查找方法查询、利用 SQL 语法查询、利用存储过程查询。

1）使用 TTable 组件设计数据库查找功能

TTable 组件提供了六种查找数据的方法。假设要按 ‘导师姓名’ 字段查找姓名为 ‘张三’ 的导师基本情况，则用不同的查询方法实现如下：

① Lookup 方法 Lookup 方法不需设定索引字段，下面代码的意思是按 ‘导师姓名’ 字段查找到 ‘张三’ 这行记录时，返回 ‘职称’ 字段的数据。

Variant Lookupresults ; //Variant 是一种复杂的数据类型，可接受多种数据类型

Lookupresults = Table1 - >Lookup “导师姓名”, “张三”, ‘职称’ ;

② find 方法 find 方法必须事先设定好索引键字段，找到后将指针也移至相应的记录，这是与 Lookup 方法不同的。这里假定我们已将导师编号设为索引键字段，程序代码为：

```
Table1 - >Close();
Table1 - >IndexFieldNames = "导师姓名";
Table1 - >Open();
Table1 - >FindKey(new TVarRec("张三"), 0);
```

③ Filter 方法 这个方法是利用 Table 组件的 Filtered 属性和 OnFilterRecord 事件来完成查询操作的。先将 Filtered 属性设成 true 然后在 OnFilterRecord 事件中加入如下代码：String temp = ‘张三’ //这里必须使用 String 型变量 否则数据类型易错

Accept = Table1 - >FieldValues[“导师姓名”] = = temp

另外还有 Locate 方法、Goto 方法和 Range 方法，因为这些都不常用，所以不一一介绍。事实上，用 Table 的方法实现查询功能有很大的弊端，它的查询效率极低。因为 Table 的语法都必须翻译为 SQL 语法后才能被后端数据库识别，并且，使用任何 TTable 的方法来存取数据库，首先都必须先 Open 这个数据库即把整个数据库表读出来，这对于追求效率的用户来说是无法忍受的，大多情况下我们还是用 TQuery 组件来实现查询功能。

2）使用 SQL 语法实现查询

C + + Builder 提供了 TQuery 组件，它可以让你直接使用 SQL 语句编写数据库程序。使用 Query 组件必须将其 DatabaseName 设为相应的数据库别名，这里我们设为 dshxxk，要实现上面的查询，我们可用两种方法实现，一种是传递参数的方法，程序写法为 ：

```
Edit1 - >Text = '张三';
Query1 - >ParamByName("temp") = Edit1 - >Text;
Query1 - >SQL - >Add( " select * from 导师基本情况表 where 导师姓名 = :temp");
Query1 - >Active = true;
```

一种是用聚集的方法，程序写法为：

```
Query1 - >SQL - >Add("select * from 导师基本情况表");
Query1 - >SQL - >Add( " where 导师姓名 = ' " + Edit1 - > Text + " ' ");
Query1 - >Active = true;
```

这两种方法各有优劣，聚集方法直接方便，但容易把数据类型搞错；参数方法不会弄错数据类型，但写起来稍有麻烦。我们可视个人习惯而定。

3）利用存储过程 （Stored Procedure）实现查询

这种方式是将 SQL 程序放在后端数据库的系统数据表中，我们执行同样的 SQL 语句时直接去执行后端数据库上的存储过程即可。这样，就不必浪费传送 SQL 程序和分析 SQL 程序的时间。在 C/S 客户机/服务器）模式中，使用存储过程还可将数据筛选工作放在 Server 端，从而极大降低了网络流量。关于存储过程的方法，我不作详细介绍。

四、报表的制作

C + + Builder 提供了强大的报表制作功能，一般有两种方法制作报表：一种是用 C + + Builder 的 QReport 组件栏中的组件来制作报表，另一种是实现 Word 或 Excel 的自动化来制作报表。这里我们讨论用第一种方法来实现复杂报表的制作。导师信息管理系统要生成两个报表，其中一个如图 5，该报表反映了导师的基本情况和导师的教学情况。导师的基本情况表中包括图片 Graphic 字段和备注 Memo 字段，备注字段存放着导师的教育背景。教学情况是动态地反映在报表上的，根据导师编号将该导师的代课情况反映在报表上，有多少条记录则可显示多少条。

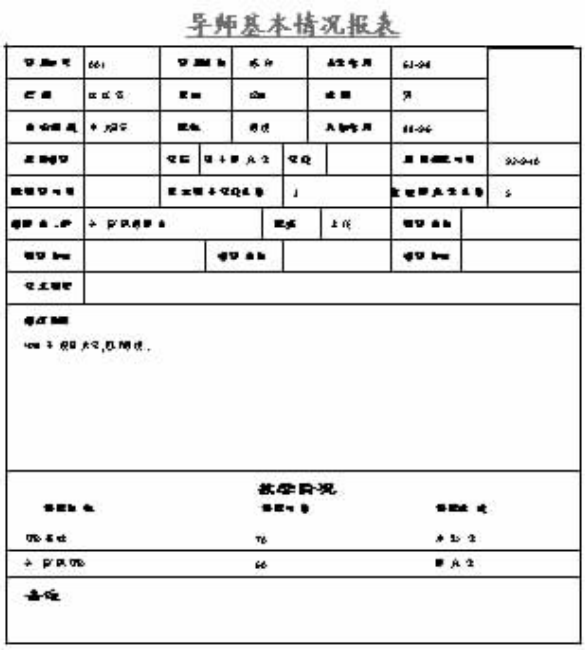


图 5

制作步骤如下：

1. 新建一个 Form，命名为 jbqkdyform，在 jbqkdyform 中依次放入如下组件，并设定其属性，各组件及其重要属性见下表：

组件	属性	值
QuickRep1	Name	QuickRep1
QRBand1	BandType	rbTitle
QRBand2	BandType	rbDetail
QRBand3	BandType	GroupHeader
QRSubDetail1	HeaderBand	QRband3
QRBand4	BandType	rbSummary
Table1	DatabaseName	dshxxk
	TableName	导师基本情况表 . DB
Table2	DatabaseName	dshxxk
	TableName	教学情况表 . DB
Query1	DatabaseName	dshxxk
Query2	DatabaseName	dshxxk

2. 在各 QRBand 上依次放入相关的 TQRLable 和 TQRDBText 组件，并将各 TQRDBText 的 DataSet 属性设为相应的 Table 引用 table 主要是方便 DataField 的设置），DataField 设为相应的表字段，最后再将 DataSet 改为相应的 TQuery 部件。例如，QRDBText1 的 DataSet 为 Query1， DataField 为导师编号。其中 TQRmemo 组件暂不做设置，在程序中设置。

3. 用 RShape 组件进行划线，将它的 Shape 属性设为 qrsRectangle 矩形 或 qrsVertLine 竖线）或 qrsHorLine 横线），并进行适当调整就可划出所想要的报表线条。各 QRBand 的边线可以通过它的 Frame 属性进行设置。最后的格式如图 6。

图 6

4. 将该单元保存为 jbqkdybiaodan. cpp，在窗体的创建事件即 jbqkdyform 的 OnCreate 事件中加入程序代码。程序代码如下：

```
void __fastcall Tjbqkdyform: : FormCreate(TObject * Sender)
    { / * dsjbqkform 是导师基本情况表名，是生成 jbqkdyform 的窗体，dsjbqkform - >DBEdit1 - >Text 为当前记录导师的编号，这里将 temp 赋值为当前记录导师的编号，作为查询的关键字 * /
 String temp = dsjbqkform - >DBEdit1 - >Text;
/ * 将导师的教育背景 DBMemo1 的值赋值到 jbqkdyform 的 QRMemo1 中 * /
this - >QRMemo1 - >Lines - >Assign(dsjbqkform - >DBMemo1 - >Lines);
this - >Query1 - >SQL - >Clear();
this - >Query1 - >SQL - >Add(″select * from 导师基本情况表 where 导师编号 = ′″ + temp + ″′″);
this - >Query2 - >SQL - >Add(″select * from 教学情况表 where 导师编号 = ′″ + temp + ″′″);
this - >Query1 - >Active = true;
this - >Query2 - >Active = true;
this - >QuickRep1 - >PreviewModal(); //预览报表
}
```

为了报表预览完毕后能正常关闭窗口，还必须编写 Quick-

弥补中文字库汉字不全的实用办法

雷 超 李 明 陈 超

一、 前言

现今汉字及汉字系统在计算机系统中已得到广泛应用。各种招生考试系统、银行实名制的实施、公安系统的户籍管理、网上缉拿凶手等，都要求能满足所用汉字的录入、查询和打印等功能。但现今使用的汉字系统为 GB2312 - 80，此标准包含的一、二级字库只有 6763 个。这和当今仍在使用的超过 5 万汉字相比，的确相差甚远。现在许多人的姓名和地名这些重要汉字因字库的不全而无法方便录入、查询。使得许多软件系统因此而显得残缺。这一问题已引起了各方面的高度重视，为此也采取了增加造字程序等办法。但这当中也存在使用不便、移植困难、网络系统难以应用等问题。今年新制订的国家标准将字库增至 3 万多个，不少人就此认为，这一老大难问题已得到解决，但事实并非如此，因为这一标准并未包含所有汉字；并且人们熟悉的五笔等音形输入法将会受到新标准的挑战。为此，我们以汉字在中文编辑系统（如：WORD2000）中造字的诸多办法，以及在数据库系统中的造字应用等问题作了较全面、系统的分析和实用方法的介绍。

二、 中文编辑系统中 ‘造字 ”

方案一

利用造字程序进行，点 ‘开始 / 程序 / 附件 / 造字程序 ” 进入， 若没有安装，可在添加删除程序中选 “Windows 安装程序 ”，安装附件中的 ‘造字程序 ” 即可），将出现 ‘造字 ” 窗口，如图 1 所示。

下面我们以 “〇 ” 字为例，步骤如下：

（）创建所造字字符

① 如图 1 所示，在 ‘文件 ” 菜单中单击 ‘字体链接 ”，

图 1

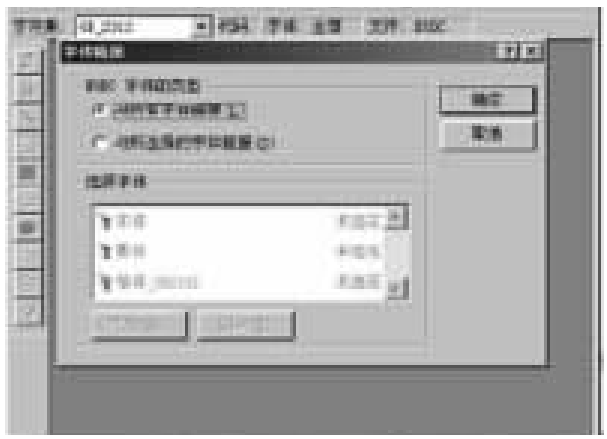


图 2

出现 ‘字体链接 ” 窗口，如图 2 所示。这一点的设置相当重要，否则调用时很有可能是一个空白字，因为我们所使用的汉字都是以点阵方式存储的，在此请选用 “与所有字体链接（） ” 更好一些。

Rep 的 AfterPreview 事件。程序代码如下：

```
/ * 预览完毕后正常关闭, 必须调用 close( ) 函数 * /
void _fastcall Tjbqkdyform: : QuickRep1AfterPreview(TObject
 * Sender)
{
this – >Close( );
}
```

通过此讲，我们了解了 C + + Builder 的一些基本概念和编程基本技术，熟悉了它的开发环境，懂得了数据库的创建和数据库别名的建立，由此进一步学习了如何实现前端界面与后台数据库的连接及常规操作，最后又介绍了报表的制作。在讲解中我以导师信息管理系统的部分内容为实例，力求让大家有一个感性的认识。读完此章，我们应能开发一般简单的数据库应用系统了。

（收稿日期：2001 年 2 月 28 日）

②在‘编辑”菜单中单击‘选定代码”，出现‘选定代码”窗口。该窗口主要由三部分组成：代码选择区、代码属性区和预览区。代码选择区即为当前新造字代码区。代码属性区指出了当前字符的代码、字体和关联文件 建议采用默认 EUDC ，另外指出造字程序的代码有效范围为 AAA1 - AFFE 和 F8A1 - FEPE，其它区域无效。预览区显示当前新选代码的字符形状。这里我们选择 AAA1，然后单击‘确定”。

③对所选代码进行编辑。

在‘编辑”菜单中单击‘调用”项，出现“复制字符”窗口。在‘字体”窗口中选中‘宋体”，在‘形态”窗口中输入‘炬”字，单击‘确定”。

然后在‘窗口”菜单中单击‘引用”项，出现‘引用”窗口。在‘字体”窗口中也选用‘宋体”， 最好与编辑窗口所选用字体相同，注：若你需其它字体的这个汉字，则必须按此方法再造这个汉字，只是在选用字体时进行改变即可，因为它都是把此图形按矢量点阵方式存入系统）。在‘形态”窗口中输入‘橘”字，单击‘确定”，出现图 3 所示编辑和引用窗口。

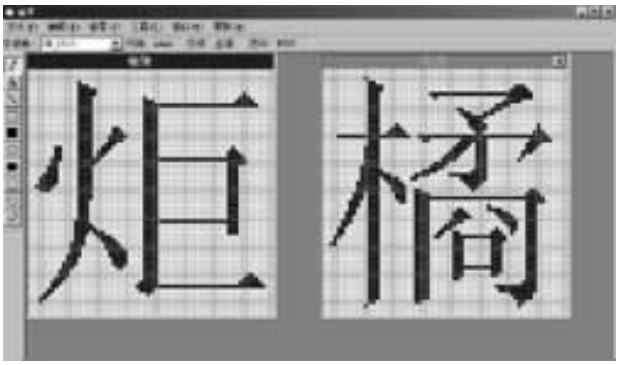


图 3

图 4

图 5

保留‘编辑”窗口中‘炬”字的左半部分“ ”旁，用任意形态选择工具选择‘引用窗”中‘橘”字的右半部分“ ”字，并拖至‘编辑”窗中，调整位置后，就组成了“ ”字，‘关闭引用”窗口如图 4 所示。

④在‘编辑”菜单上单击‘保存字符 V ”，保存对造字字符所做的修改。如果想以其它代码编号来保存现有的字符，请单击‘将字符另存为 A ”。选择其它代码，则该字符保存在该代码中，所保存的字体文件 EUDC. TTE 位于 Windows 文件夹中。此时就可以通过区位输入法直接调用，只要在区位码输入状态下键入 AAA1 即可。

（）设置造字字符的外码

我们可以编辑造字字符之后设置造字字符的外码。当编辑完一个字符后，在‘编辑”菜单中选择‘输入法链接”，选择“hWindows hCommand h”文件夹；若需五笔字型，则输入“Wb. TBL 微软拼音输入法，则输入“Weipin. TBL”；双拼输入法，则选择“Shanpin. tbl”；全拼输入法，则选择“Pinyin. tbl”；其它类同 。在弹出的‘输入法链接”窗口中再输入外码。如我们给“ ”字输入五笔字型的外码“OCMK”，如图 5 所示，再单击‘注册”，则定义的字符自动加入到五笔字型的输入法中。以后仅需在五笔字型输入法下输入‘OCMK 方可。

（）造字字符的可移植性

所造的字符如果要在别的微机上应用，就必须进行可移植性处理。只需将本机上 Window 文件夹中的 EUDC. EUF 和 EUDC. TTE 这两个文件复制到新机上，将新机 Windows 文件夹中的 EUDC. EUF 和 EUDC. TTE 这两个文件覆盖，重新启动微机后，即可进行调用，即在所使用的地方用区位码输入“AAA1”，若要使用五笔字型或微软拼音输入法，则需将“hWindows hCommand h”下的“Wb. TBL”或“Weipin. TBL”文件拷贝到新机的“hWindows hCommand h”目录下方可。

方案二

利用 Windows 的画图工具进行画图生成所需的生僻汉字。点‘开始/程序 /附件 /画笔”或直接运行 MSPAINT，首先必须以透明方式新建文档，可插入一些汉字，再用画点、线、移动及其它工具进行描点，很方便地就能生成所需生僻汉字的图片，最后放入需要的位置调整后即可。此方案也可将其拷贝后移植到另一台计算机中使用。缺点是操作有一定的技术难度。

方案三

利用常用的汉字处理软件 如 Word、Notepad 或北大方正等）进行调整。我就以 Word 造“ ”字为例；①先找他的某一部分进行录入，在此录入‘吉”；②再录入他另一些内容，在此录入‘力”；③最后选择这两个字，点‘格式/字体/字符间距”，选择合适的缩放比例，如 50% 即可得‘吉力”字，其它与此类同。用这种方法即可很快、很实时地得到想要的汉字。缺点是操作较复杂，可移植性不强。

方案四

此方案要利用外围设备，可将要用的字用笔 最好用钢笔或铅笔）把他写在一张白纸上，如“ ”字，再用手置式小型扫描仪或平板式扫描仪进行扫描后，将图片插入到需要的位置，稍作调整即可达到效果。缺点是成本较高，有条件的用户值得一试。

附注：也可采用摄像机、数码相机、视频捕捉卡等进行

相应的录入与播放计算机系统未提供的生僻汉字。总之，条条道路通罗马，都能达到同一目的。

三、 数据库系统中 ‘造字 ”

1. 设计需求

数据库系统中诸如姓名、地址等重要信息，若因系统字库的不完善而无法得出，会造成查询、打印出现严重的缺陷。此问题一直困扰着数据库系统的设计者。在单机版数据库系统中，也有一些不太完善的应对方法，但考虑到网络时代的需要，原有的诸如造字等方法，难以做到数据的共享。经过多种方案的反复比较，我们认为将 ‘生僻字 ”设计为图片，放在服务器上是最为简单可行的办法。为此以 VF6. 0 作为开发工具，进行了尝试和应用。

2. 数据结构

服务器端

总表

字段名	类型	长度	索引	null
xm	字符型	8		. f.
id1	字符型	16	upper	. t.
py1	字符型	5	upper	. t.
id2	字符型	16	upper	. t.
py2	字符型	5	upper	. t.
id3	字符型	16	upper	. t.
py3	字符型	5	upper	. t.
id4	字符型	16	upper	. t.
py4	字符型	5	upper	. t.

字库表

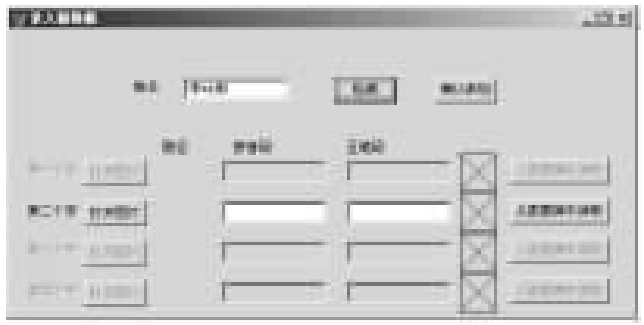
id	字符型	16	upper primary	. f.
wb	字符型	4	upper	. t.
tp	image			. f.

客户端

字段名	类型	长度	索引	null
xm	字符型	8		. f.
id1	字符型	16	upper	. t.
tp1	通用型			. t.
py1	字符型	5	upper	. t.
wu1	字符型	4	upper	. t.
ch1	逻辑型			
id2	字符型	16	upper	. t.
tp2	通用型			. t.
py2	字符型	5	upper	. t.
wu2	字符型	4	upper	. t.
ch1	逻辑型			
id3	字符型	16	upper	. t.
tp3	通用型			. t.
py3	字符型	5	upper	. t.
wu3	字符型	4	upper	. t.
ch1	逻辑型			
id4	字符型	16	upper	. t.
tp4	通用型			. t.
py4	字符型	5	upper	. t.
wu4	字符型	4	upper	. t.

3. 程序结构

（）新纪录录入



在输入框中输入姓名 特殊字以 * & 表示 点击 ‘检测 ’后，自动对输入进行判断，对应的选择框变为可用。可以选择 ‘打开图片 ’或是从数据库中读取。

A 若是 ‘打开图片 ’，则请输入 ‘拼音码 ’和 ‘五笔码 ’，以便日后查询。

B 若是 ‘从数据库中读取 ’ 则打开另一个表单。

选择好纪录后， ‘确定 ’。系统自动会将图片和拼音码、五笔码输入数据库中。

同时关闭此表单，回到上一个表单。可以继续录入或更改。

（）记录的打印

实际数据为：陈 * & 希。打印结果为 陈 + ‘图片 ”+ 希。

4. 部分代码

（）检测特殊字符

```
private cname, pos1, i
cname =alltrim(thisform. text1. value) * text1 为输入姓名的文本框.
for i =1 to 4
pos1 =right(left(cname, i * 2), 2)
if pos1 =″ * & ″
    do case
        case i =1
thisform. command11. enabled =. t. * 控件的后缀为 11 和 12 的是第一个字后面的项目, 同理递推 *
            thisform. command12. enabled =. t.
            thisform. label11. enabled =. t.
            thisform. label12. enabled =. t.
            thisform. text11. enabled =. t.
            thisform. text12. enabled =. t.
        case i =2
            thisform. command21. enabled =. t.
            thisform. command22. enabled =. t.
            thisform. label21. enabled =. t.
            thisform. label22. enabled =. t.
            thisform. text21. enabled =. t.
            thisform. text22. enabled =. t.
        case i =3
            thisform. command31. enabled =. t.
            thisform. command32. enabled =. t.
            thisform. label31. enabled =. t.
            thisform. label32. enabled =. t.
            thisform. text31. enabled =. t.
            thisform. text32. enabled =. t.
        case i =4
            thisform. command41. enabled =. t.
            thisform. command42. enabled =. t.
            thisform. label41. enabled =. t.
            thisform. label42. enabled =. t.
            thisform. text41. enabled =. t.
            thisform. text42. enabled =. t.
    endcase
endif
endfor
```

（）id 的生成

利用 vb 写一个 activex 控件，其中调用 Win api 函数，得到本机的网卡 mac 地址 （2 位，一定唯一），当把图片加入数据库时，若是从服务器上读取的，则将原来的 id 同时加入到纪录中；若是新加入一个文件的话，会在发送数据时计算 id （以 mac 地址加上 4 位形成）。

用 VB 编写 activex 的源代码：

´vb 设定为生成一个 activex 控件, 下面代码为添加的模块中的部分. 为控件添加一个方法程序. 在 vf 中调用. 1

```
Option Explicit
Public Const SOCKET_ERROR        As Long  =  -
Public Type HOSTENT
  hName      As Long
  hAliases   As Long
  hAddrType  As Integer
  hLen       As Integer
  hAddrList  As Long
End Type
Public Declare Function WSAGetLastError Lib ″WSOCK32. DLL″ () As Long
Public Declare Function WSAStartup Lib ″WSOCK32. DLL″ (ByVal wVersionRequired As Long, lpWSADATA As WSADATA) As Long
Public Declare Function WSACleanup Lib ″WSOCK32. DLL″ () As Long
Public Declare Function gethostname Lib ″WSOCK32. DLL″ (ByVal szHost As String, ByVal dwHostLen As Long) As Long
Public Declare Function gethostbyname Lib ″WSOCK32. DLL″ (ByVal szHost As String) As Long
Public Declare Sub CopyMemory Lib ″kernel32″ Alias ″RtlMoveMemory″(hpvDest As Any, ByVal hpvSource As Long, ByVal cbCopy As Long)
Public Function GetIPAddress() As String
  Dim sHostName    As String * 256
  Dim lpHost     As Long
  Dim HOST       As HOSTENT
  Dim dwIPAddr   As Long
  Dim tmpIPAddr() As Byte
  Dim i          As Integer
  Dim sIPAddr    As String
  If Not SocketsInitialize() Then
    GetIPAddress = ″″
    Exit Function
  End If
 If gethostname(sHostName, 256) = SOCKET_ERROR Then
    GetIPAddress = ″″
      MsgBox ″Windows Sockets error ″ & Str$(WSAGetLastError()) & ″ has occurred. Unable to successfully get Host Name. ″
    SocketsCleanup
    Exit Function
  End If
  sHostName = Trim$(sHostName)
  lpHost = gethostbyname(sHostName)
  If lpHost = 0 Then
    GetIPAddress = ″″
  MsgBox ″Windows Sockets are not responding. ″ & ″Unable to successfully get Host Name. ″
    SocketsCleanup
    Exit Function
  End If
  CopyMemory HOST, lpHost, Len(HOST)
  CopyMemory dwIPAddr, HOST. hAddrList, 4
  ReDim tmpIPAddr(1 To HOST. hLen)
  CopyMemory tmpIPAddr(1), dwIPAddr, HOST. hLen
  For i = 1 To HOST. hLen
    sIPAddr = sIPAddr & tmpIPAddr(i) & ″. ″
  Next
  GetIPAddress = Mid$(sIPAddr, 1, Len(sIPAddr) - 1)
```

```
  SocketsCleanup
End Function
Public Sub SocketsCleanup()
   If WSACleanup() <> ERROR_SUCCESS Then
      MsgBox "Socket error occurred in Cleanup."
   End If
End Sub
Public Function SocketsInitialize() As Boolean
 Dim WSAD As WSADATA
  Dim sLoByte As String
  Dim sHiByte As String
   If WSAStartup(WS_VERSION_REQD, WSAD) <> ER-
ROR_SUCCESS Then
MsgBox "The 32 – bit Windows Socket is not responding."
      SocketsInitialize = False
      Exit Function
  End If
  If WSAD.wMaxSockets < MIN_SOCKETS_REQD Then
      MsgBox "This application requires a minimum of " & _
CStr(MIN_SOCKETS_REQD) & " supported sockets."
       SocketsInitialize = False
       Exit Function
   End If
  If LoByte(WSAD.wVersion) < WS_VERSION_MAJOR Or
(LoByte(WSAD.wVersion) = WS_VERSION_MAJOR And
HiByte(WSAD.wVersion) < WS_VERSION_MINOR) Then
      sHiByte = CStr(HiByte(WSAD.wVersion))
      sLoByte = CStr(LoByte(WSAD.wVersion))
      MsgBox "Sockets version " & sLoByte & "." &
sHiByte & "is not supported by 32 – bit Windows Sockets."
      SocketsInitialize = False
      Exit Function
  End If
  'must be OK, so lets do it
  SocketsInitialize = True
End Function
```

以上为获得本机 ip 地址的程序，以后为获得 mac 地址的程序。

```
Public Const NCBNAMSZ As Long = & H16
Public Const NCBRESET As Long = & H32
Public Const NCBASTAT As Long = & H33
Public Const HEAP_ZERO_MEMORY As Long = & H8
Public Const HEAP_GENERATE_EXCEPTIONS As Long = &
H4
Public Type NET_CONTROL_BLOCK 'NCB
  ncb_command   As Byte
  ncb_retcode   As Byte
  ncb_lsn       As Byte
  ncb_num       As Byte
  ncb_buffer    As Long
  ncb_length    As Integer
  ncb_callname  As String * NCBNAMSZ
  ncb_name      As String * NCBNAMSZ
  ncb_rto       As Byte
  ncb_sto       As Byte
  ncb_post      As Long
  ncb_lana_num  As Byte
  ncb_cmd_cplt  As Byte
  ncb_reserve(9) As Byte ' Reserved, must be 0
  ncb_event     As Long
End Type
Public Type ADAPTER_STATUS
  adapter_address(5) As Byte
  rev_major         As Byte
  reserved0         As Byte
  adapter_type      As Byte
  rev_minor         As Byte
  duration          As Integer
  frmr_recv         As Integer
  frmr_xmit         As Integer
  iframe_recv_err   As Integer
  xmit_aborts       As Integer
  xmit_success      As Long
  recv_success      As Long
  iframe_xmit_err   As Integer
  recv_buff_unavail As Integer
  t1_timeouts       As Integer
  ti_timeouts       As Integer
  Reserved1         As Long
  free_ncbs         As Integer
  max_cfg_ncbs      As Integer
  max_ncbs          As Integer
  xmit_buf_unavail  As Integer
  max_dgram_size    As Integer
  pending_sess      As Integer
  max_cfg_sess      As Integer
  max_sess          As Integer
  max_sess_pkt_size As Integer
  name_count        As Integer
End Type
Public Declare Function Netbios Lib "netapi32.dll" (Pncb As
NET_CONTROL_BLOCK) As Byte
Public Type NAME_BUFFER
  name        As String * NCBNAMSZ
  name_num    As Integer
  name_flags  As Integer
End Type
Public Type ASTAT
  adapt         As ADAPTER_STATUS
  NameBuff(30)  As NAME_BUFFER
End Type
Public Declare Function GetProcessHeap Lib "kernel32" ()
As Long
Public Declare Function HeapAlloc Lib "kernel32" (ByVal
hHeap As Long, ByVal dwFlags As Long, ByVal dwBytes As
Long) As Long
Public Declare Function HeapFree Lib "kernel32" (ByVal
hHeap As Long, ByVal dwFlags As Long, lpMem As Any)
As Long
Public Function GetMACAddress(cip As String) As String
  Dim ret As Byte
  Dim tmp As String
  Dim pastat As Long
```

```
Dim ncb As NET_CONTROL_BLOCK
Dim ast As ASTAT
ncb. ncb_command = NCBRESET
Call Netbios(ncb)
ncb. ncb_callname = cip
ncb. ncb_command = NCBASTAT
ncb. ncb_lana_num = 0
ncb. ncb_length = Len(ast)
pastat = HeapAlloc(GetProcessHeap(), HEAP_GENERATE_
EXCEPTIONS Or HEAP_ZERO_MEMORY, ncb. ncb_length)
If pastat = 0 Then
   Debug. Print "memory allocation failed!"
   Exit Function
End If
ncb. ncb_buffer = pastat
Call Netbios(ncb)
CopyMemory ast, ncb. ncb_buffer, Len(ast)
tmp = Format$(Hex(ast. adapt. adapter_address(0)), "00")
& "" & Format$(Hex(ast. adapt. adapter_address(1)), "00")
& "" & Format$(Hex(ast. adapt. adapter_address(2)), "00")
& "" & Format$(Hex(ast. adapt. adapter_address(3)), "00")
& "" & Format$(Hex(ast. adapt. adapter_address(4)), "00")
& ""& Format$(Hex(ast. adapt. adapter_address(5)), "00")
HeapFree GetProcessHeap(), 0, pastat
GetMACAddress = tmp
End Function
```

（3）姓名的打印

在打印时每一个姓名放置 8 个控件（4 个文本框，4 个图片）。文本框的打印条件为对应的 ch 为 . f. ，其值为对应的那一个字。图片的打印条件与之相反。

5. 经验总结

（1）姓名录入时使用的特殊字符应该是两位的，而且不能是同样的字符，如& & 之类。而必须是不同的字符，如我们选择的 * & 。

（2）id 必须唯一

为了不至于发生数据冲突。必须找到一个能产生唯一 id 的方法。想到网卡的 Mac 地址是唯一的，所以使用获得 Mac 地址的方法。而获得 Mac 地址需要使用 Windows api 函数，可惜 vf 对要使用的函数不支持，只好使用 VB 编写成控件由 vf 调用。

（3）打印姓名

实际应用中一定会将姓名打印输出。设计时将图片和文本放在一起，打印条件设置为同时仅打印文本和图片中的一项，即打印正确的汉字或图片，而不会打出 * & 的符号来。

（收稿日期：2001 年 1 月 2 日）

瑞星：无忧服务

对国外杀毒厂商而言，在中国市场上，他们并未建立有效而完善的服务网络，以致无论在技术上、人力上都无从达到迅急应变的要求，在传递到客户的过程中往往是耗费了大量人力、物力与时间而收效甚微，从而往往是给企业造成巨大的损失。

从国内的反病毒市场来看，北京瑞星公司堪称业界龙头企业。在技术上，瑞星率先实现了网络病毒防治领域内最新的功能突破，真正实现了网络安全管理的全网远程化、自动化与智能化维护，在 2000 年 11 月初的国内杀毒软件评测中先声夺人，获得了网络病毒防治产品的第一名。

首先，瑞星的服务是多样化的。早在 1999 年，瑞星于反病毒界首家开通了 24 小时 800 免费服务电话，为用户提供优惠、快捷的技术支持与升级咨询服务；同时，用户可通过邮局、电子邮件、瑞星呼叫中心服务系统及其销售代理服务网等渠道向瑞星公司总部发出求助信息，可由瑞星公司合作的小红帽上门服务方式获取最新的软件升级程序等，以便得到尽快的技术服务反馈；互联网上，也有瑞星提供的可供用户免费下载的升级程序；2000 年，公司在上海、广州、沈阳、西安、成都以及深圳、福州、长沙、郑州、昆明、乌鲁木齐等国内 11 大城市正式成立了共 17 家瑞星授权客户服务站，这样，各地用户可通过特约的瑞星服务网进行产品的更新、升级等工作……。

从升级途径看，瑞星也为用户提供了相对完善、连续的服务：1. 用户只需到瑞星网站注册，就可下载升级（http ://www. rising. com. cn）；2. 可从瑞星 BBS 站下载升级；（电话：010 - 62641700）3. 通过邮局，向瑞星公司汇邮费 25 元，可索取升级盘进行自升级；4. 携带原盘和用户服务卡到瑞星公司技术支持部也可免费升级；5. 在当地瑞星授权服务站或购买处都可以进行升级。

2001 年，瑞星还将在提供技术与产品培训的基础上建立客户信息管理系统，将公司所有的客户资源建成特殊客户群，提供产品测试、免费试用、及时病毒疫情通报等服务，这一举措可视为瑞星让公司真正融入客户、体现服务价值的关键一步，同时这也会为附加值服务的实现提供有益的思路。到年底为止，瑞星还将在全国建立 100 家技术服务分中心或技术服务站，而这样的服务专门体系一定会提高服务能力的深度与广度，解决企业级服务深入的技术、过程、方案、地域等系列问题。

无可置疑的，在今天的网络时代，服务已成为衡量反病毒软件企业能力与实力的重要标志。瑞星产品能够在国内的杀毒软件统一评测中夺得双项第一，凭借的也正是其益加先进、完善的服务力量。软件的生命在于及时的升级，而服务的生命也就在于不断地有所创新，有所改革，将市场关注、产品研发的目光锁定在用户身上，始终设身处地从用户角度出发去考虑产品与服务的问题。而在未来，反病毒市场的进一步拓展更将随着用户需求的变化而得以更深意义上的改善，换句话说，任何产品的市场推广都将围绕用户来进行，以求得更加长远的发展。

利用 VBA 建立 AutoCad2000 与 Excel 通信

许春培

一、 Excel 的 ActiveX 对象模型

1. WorkBooks 集合对象

一个 WorkBook 对象实际上就是一个 Excel 文件，Excel 应用程序可以同时打或创建多个文件，它们被保存在 WorkBooks 集合对象中，可以通过索引号或名称访问集合中的任何一个工作簿，如下语句所示：

该语句激活 WorkBooks 集合中的第一个工作簿，使其成为当前工作簿

WorkBooks 1 . Activate

该语句激活 WorkBooks 集合中的 Mybook. xls 工作簿，使其成为当前工作簿

WorkBooks “Mybook. xls” . Activate

2. Worksheets 对象

每个工作簿对象上可以有多个工作表 WorkSheet。在默认情况下 Excel 的当前工作簿上有名为 Sheet1、Sheet2、Sheet3 三个工作表，并且 Sheet1 为当前工作表。如果想使 Sheet2 成为当前工作表，则可使用下列语句

ExcelApp. Worksheets “Sheet2” . Activate

3. Range 对象

该对象用来指定工作表上的区域。将单元格 A1 的值赋给单元格 A5 的语句说明如下：

Worksheets “sheet1” . range “A5” . value = worksheets “Sheet1” . range “A1” . value]

上述语句将 Sheet1 工作表上的 A1 第 1 行第 1 列）单元格中的值，赋给 Sheet1 工作表上的 A5 第 5 行第 1 列）单元格。

再看下面的语句：

'将单元格 A1 和 D26 构成的区域选中

worksheets “Sheet1” . range “a1 d26” . select

这条语句中的 Select 方法所产生的效果，与我们平时用鼠标在屏幕上将 A1：D26 区域上的单元格进行刷黑选择是一样的。Rnge 对象的另一个重要方法是 Sort，该方法用来对工作表上选定的区域进行排序，它带有许多参数，下面我们看一下该方法的语法格式：

Expression. sort(Key1, Order1, Key2, Type, Order2, Key3, Order3, Header, OrderCustom, MatchCase, Orientation, SortMethod, IgnoreControlCharacters, IgnoreDiacritics, IgnoreKashide)

其中：

expression：必选参数。该表达式返回 Rang 对象选定的区域。

Key1：Variant 类型，可选参数。第一个排序字段，主要是 Rang 对象返回的区域或由工作表对象的 Columns 属性指定的列。

Order1：Variant 类型，可选参数。可为下例 xlSortOrder 内置常量之一 xlAscending 或 xlDescending。用 xlAscending 表示以升序排列 Key1。用 xlDescending 表示以降序排列 Key1。默认值为升序 xlAscending。

Key2：Variant 类型，可选参数。第二个排序字段，主要是 Rang 对象返回的区域或由工作表对象 Columns 的属性指定的列。如果省略本参数，则没有第二个排序字段。对数据透视表排序时不用。

Type：Varoant 类型，可选参数。指定参与排序的要素。可为下列 xlSortType 常量之一：xlSortValues 或 xlSortLabels。仅用于对数据透视表的排序。

Order2：Variant 类型。可选参数。可为下列 XlSortOrder 常量之一：xlDescending 或 xlDescending。用 xlAscending 表示以升序排列 Key2。用 xldescending 表示以降序排列 Key2。默认值为 xlAscending。对数据透视表排序时不用。

Key3：Variant 类型，可选参数。第三个排序字段，主要是 Rang 对象返回的区域或由工作表对象的 Columns 属性指定的列。如果省略本参数，则没有第三个排序字段。对数据透视表排序时不用。

Order3：Variant 类型，可选参数。可为下列 xlSortOrder 常量之一：xlAscending 或 xlDescending。用 xlAscending 表示以升序排列 Key3，用 xlDescending 表示以降序排列 Key3。默认值为 xlAscending。对数据透视表排序时不用。

Heard：Variant 类型，可选取参数。指定第一行是否包含标题。可为下列 xlYesNoGuess 常量之一：xlYes、xlNo 或 xlGuess。如果首行包含标题 (不对首行排序），就指定 xlYes。如果首行不包含标题 对整个区域排序），就指定 xlNo。若指定为 xlGuess，将由 Microsoft Excel 判断是否有标题及标题位于何处。默认值为 xlNo。对数据透视表排序时不用。

OrderCustom：Variant 类型，可选参数。以从 1 开始的整数指定在自定义排序顺序列表中的索引号。如果省略本参数，就使用不着 1 （'常规：”）。

MatchCase：Variant 类型，可选。若指定为 True，则进行区分大小写的排序 若指定为 False，则排序时不区分大小写。对数据透视表排序时不用。

Orientation：Variant 类型，可选参数。如果指定为 xlTopToBottom，排序将从上到下 按行）进行。如果指定为 xlLeftToRight，

排序将从左到右 按列)进行。

SortMethod：Variant 类型，可选参数。排序方式。可为下列 xlSortMethod 常量之一：xlSyllabary 按发音排序）或 xlCodePage 按代码页排序）。默认值为 xlSyllabary。

IgnoreControlCharacters：Variant 类型，可选参数。不用于美国英语版的 Microsoft Excel 中。

IgnoreDiacritics Variant 类型，可选参数，不用于美国英语版的 Microsoft Excel 中。

IgnoreKashida：Variant 类型，可选参数。不用于美国英语版的 Microsoft Excel 中。

下面语句是有关使用 Sort 方法的 2 个示例。

示例 1：对工作表 "Sheet1" 上的单元格区域 A1：C20 进行排序，用单元格 A1 作为第一关键字，用单元格 B1 作为第二关键盘字。排序是按行以升序 默认）进行的，没有标题。

```
Worksheets "Sheet1". range "A1 c20". sort key1 = worksheets "Sheet1". range "A1" key2 = Worksheets "Sheet1". range "B1"
```

示例 2 对工作表 "Sheet1" 上包含单元格 "A1" 的当前区进行排序，按第一列中的数据进行排序，并且自动判断是否存在标题行。Sort 方法将自动判断当前区。

```
Worksheets "Sheet1". Range "A1". Sort Key1 = Worksheets "Sheet1". Columns "A" Header = xlGuess
```

4. Cells 属性

工作表对象中的 Cells 属性，在单元格的选择方面可以达到与 Rang 相同的效果，它是以行 Row 和列 Gol 作为参数的，如下语句所示：

```
'将单元格 A1 的值赋给单元格 A5
Worksheets "Sheet1". Cells 5 1 . Value = Worksheets "Sheet1". Cells 1 1 . Value
```

上面语句即将第 1 行第 1 列 (A1）单元格内的值，赋给第 5 行第 1 列 (A5）单元格。Cells 属性的优点是，对于行和列的选择可以采用变量，如下语句所示：

```
Worksheets(″Sheet1″). Activate
For theYear = 1 to 5
Cells(1, theYear + 1). Value = 1990 + theYear
Next theYear
```

上述语句将在当前工作表的第一行的第 2、3、4、5、6 列，分别添上 1992、1993、1994、1995 和 1996 的值。注意，由于第 1 条语句已将 Sheet1 设为当前工作簿，所以 Cells 属性可以不必显示指定工作表。

5. GetObject 和 CreateObject 函数

二、在 AutoCad 创建 Excel 应用程序

1. 打开 AutoCad 的 VBA 编辑器

2. 选择 "工具" h "引用" 项，在弹出的 "引用" 对话框的 "可使用的引用" 列表框内，选择 "Microsoft Excel 8. 0 Object Library" 项

3. 单击 "确定" 按钮

4. 接下来使用下列代码就可创建完整的应用程序对象实例：

```
Dim ExcelApp as Excel. Application
'激活要与之通信的 Excel 应用程序
On Error Resume Next
Set ExcelApp = GetObject( , ″Excel. Application″)
If Err <>0 Then
Set ExcelApp = CreateObject(″Excel. Applicationn″)
End If
```

注意 GetObject 和 CreateObject 函数的区别。当 Excel 程序已经在运行时，前者可以马上创建 Excel 应用程序的实例，这样不会出现 2 个 Excel 应用程序对象实例，这将有效地节省系统资源的开销。如果当前 Excel 没有运行，GetObject 函数将出错，紧接着 Err 将捕获错误，并运行 CreateObject 函数创建一个 Excel 应用程序实例。所以在具体使用时，这 2 个函数最好都不要省略。

三、将明细表做成一个 Excel 报表

1. 运行 AutoCad2000 程序

2. 打开 AutoCad2000 主运行文件夹下的 "hSample hActives hExtAtt hattrib. dwg" 文件。该文件的右上角有一明细表，该明细表的每一行都是一个插入的块引用，显示的文字就是块的属性文本或标签 生要用于标题）

3. 创建成下面的过程及代码，并运行之

```
Sub BlkAttr_Extract()
Dim Excel As Excel. Application
Dim ExcelSheet As Object
Dim ExcelWorkbook As Object
'创建 Excel 应用程序实例
On Error Resume Next
Set Excel = GetObject(, ″Excel. Application″)
If Err <> 0 Then
  Set Excel = CreateObject(″Excel. Application″)
End If
'创建一个新工作簿
Set ExcelWorkbook = Excel. Workbooks. Add
'确保 Sheet1 工作表为当前工作表
Set ExcelSheet = Excel. ActiveSheet
'将新创建的工作簿保存为 Excel 文件
ExcelWorkbook. SaveAs ″属性表 . xls″
'令 Excel 应用程序可见
Dim RowNum As Integer
Dim Header As Boolean
Dim blkElem As AcadEntity
Dim Array1 As Variant
Dim Count As Integer
RowNum = 1
Header = False
'遍历模型空间, 查找明细表的每个块引用表行
For bEach blkElem In ThisDrawing. ModelSpace
  With blkElem
'当
```

下转第 26 页

浅谈用 VC 进行应用程序背景的设置

胡朝晖

一、 引言

通常来说，我们的应用程序可以分为基于对话框、基于单文档和基于多文档三种。最通常的换肤就是改变应用程序的背景。改变背景可以有两种形式，一种是简单地改变背景的颜色，另外一种就是用一个图片作为背景。

二、关于对话框背景的改变

1. 改变对话框背景的颜色

对话框窗口中有一个消息处理函数为 OnCtlColor，该消息处理函数返回的参数是刷子句柄，该刷子就是该对话框所用来对背景进行填充的颜色 通俗一点说就象我们用刷子刷墙壁一样，刷子上是什么颜色，墙上也就是什么颜色了 。相应的代码如下：

```
HBRUSH CYourDlg: : OnCtlColor(CDC * pDC, CWnd * pWnd, UINT nCtlColor)
{
  return m_YourBrush;
}
```

这个 m_YourBrush 实际上就是我们自己定义的刷子。一个比较好但不是唯一的定义刷子的地方是在 CYourDlg OnInitDialog 中。

下面简单描述以下方法 OnCtlColor 的作用：

根据微软的技术说明，OnCtlColor 必须返回一个刷子句柄用来作为画控件背景的刷子。实际上，方法 OnCtlColor 一般应用在基于对话框的应用中，用来控制对话框和对话框上各种子控件的处理。

OnCtlColor 包括三个参数：

pDC：指向子窗口的显示设备上下文的指针

pWnd：代表请求颜色的控件的指针

nCtlColor：系统指定了一些值，用来指定控件的类型

CTLCOLOR_BTN	按纽控件
CTLCOLOR_DLG	对话框控件
CTLCOLOR_EDIT	编辑框控件
CTLCOLOR_LISTBOX	列表框控件
CTLCOLOR_MSGBOX	消息对话框控件
CTLCOLOR_SCROLLBAR	滚动条控件
CTLCOLOR_STATIC	静态文本控件

当需要画一个子控件的时候，窗口框架调用该函数，许多控件发送这个消息给它们的父窗口 优其在一个对话框中），让它准备好 pDC 并使用正确的颜色来画该控件。

2. 用位图作为对话框的背景

用位图做背景的时候，我们需要重点处理的是方法 OnEraseBkgnd，需要注意的是，在 classWizard 中，关于 Dialog 类的消息 OnEraseBkgnd 处理是不存在的。所以用户需要手工进行添加。

（1）关于 CWnd 类中的方法 OnEraseBkgnd 的说明。OnEraseBkgnd 包含一个参数：pDC，这实际上传递的是设备上下文对象的指针。当 CWnd 对象的背景需要擦除 比如窗口尺寸改变）的时候，应用框架会调用该成员函数。调用该成员函数的目的是让它准备好一块需要绘制的区域。系统缺省的实现是使用窗口类背景刷子 该刷子是由 Window 类结构中的 hbrBackground 来指定的）来擦除背景。在继承了 CWnd 的类对象的实例中，需要重载该方法的时候，需要返回一个非零值。这意味着以后不再用窗口类背景刷子来重新油漆该背景。

（2）建立一个和位图相关的类对象，这里可以建立一个类，我们不妨定义该类为 CMyBitmap，建立一个该类的实例，比如在对话框对象初始化的时候，通过该对象的实例装载具体的位图，示例的代码如下：

```
m_bmpBackground. LoadResource(uResource)
```

这里的参数 uResource 实际上就是我们要作为背景的位图的 ID 号。类 CMyBitmap 中方法 LoadResource 的作用就是确定类 CMyBitmap 要操作的位图是什么并存放起来。

3. 关于 CMyBitmap 的定义和实现

具体定义如下：

```
 class CMyBitmap : public CBitmap
{
public:
  BOOL LoadResource(UINT ID);  //装载资源
  //在指定的设备上下文和指定的位置画位图
 void DrawBMP(CDC * pDC, int x, int y, int width, int height);
  void DrawBMP(CDC * pDC, int x =0, int y =0);
  CMyBitmap();
  virtual ~CMyBitmap();
  //得到位图的大小
 int    GetWidth();
  int   GetHeight();
private:
  BOOL m_hasBMP;  //位图是否已经装载
};
```

具体实现如下

```
CMyBitmap: : CMyBitmap()
{
  m_hasBMP =FALSE;
}
BOOL CMyBitmap: : LoadResource(UINT ID)
{
```

```
    m_hasBMP = this ->LoadBitmap(ID);
    return m_hasBMP;
}
int CMyBitmap::GetWidth()
{
 BITMAP bm;
    ASSERT(m_hasBMP);
    GetBitmap(&bm);
    return bm.bmWidth;
}
int CMyBitmap::GetHeight()
{
 BITMAP bm;
    ASSERT(m_hasBMP);
    GetBitmap(&bm);
    return bm.bmHeight;
}
void CMyBitmap::DrawBMP( CDC * pDC, int x, int y, int
width, int height )
{
 ASSERT(pDC);
    CDC memDC;
    CPalette * pOldPal = 0;
 CBitmap *oldMemBitmap;
    memDC.CreateCompatibleDC(pDC);
 oldMemBitmap =memDC.SelectObject(this);
  pDC ->StretchBlt( x, y, width, height, &memDC, 0, 0,
GetWidth(), GetHeight(), SRCCOPY );
 memDC.SelectObject(oldMemBitmap);
}
void CMyBitmap::DrawBMP( CDC* pDC, int x, int y ) {
    DrawBMP( pDC, x, y, GetWidth(), GetHeight() );
}
```

4. 在确定了位图并和类 CMyBitmap 相关联以后，我们就要重载对话框类（注意对话框类实际上继承了类 CWnd）的方法 OnEraseBkgnd，在 OnEraseBkgnd 中，利用传递进来的参数 pDC，和我们定义的类 CMyBitmap 的实例 m_bmpBackground，描绘对话框的背景：

如果需要位图在对话框中进行拉升以和对话框大小相一致的话，采用如下的代码：

```
    m_bmpBackground.DrawBMP(pDC, left, right, Width,
Height);
```

如果只需要在对话框中以原图大小画出位图大小的话，只需要调用

```
    m_bmpBackground.DrawBMP(pDC, left, right);
```

需要注意的是 DrawBMP 是类 CMyBitmap 中的一个方法，其中 pDC 代表设备上下文对象句柄的指针（得到这个指针以后，也意味着我们得到了可以向对话框背景画图的前提），后四个参数决定了如何在对话框中放置选择的位图。Left、right 表示位图的左右角的坐标，Width、Height 表示位图的宽度和高度。

如果要把位图在对话框中进行平铺的话，类似的代码如下：

注意下面代码中的 x y 实际上是在对话框中画位图的左上角的坐标

```
    while(y < rc.Height()) {  //rc 表示对话框的大小尺寸
        while(x < rc.Width()) {
  m_bmpBackground.DrawBMP(pDC, x, y);
x += m_bmpBackground.GetWidth(); //得到位图的宽度
        }
  x = 0;
y += m_bmpBackground.GetHeight();  //得到位图的高度
```

三、基于单文档应用（SDI）的背景的操作

1. 改变应用背景的颜色

具体还是通过对方法 OnEraseBkgnd 进行重载而实现的。具体代码如下：

在 CMySkinView 中 OnEraseBkgnd 写入如下的代码：

```
 CBrush * oldBrush;
  //m_brush 可以在我们的类 CMySkinView 中进行定义
 //并在 CMySkinView 初始化的时候创建该刷子
  oldBrush =pDC ->SelectObject(&m_brush);  //选择新的
刷子作为刷应用背景的刷子
 CRect rcClip;
pDC ->GetClipBox(&rcClip);  //得到背景需要刷新的区域
 //用新的刷子刷背景区域
  pDC ->PatBlt(rcClip.left, rcClip.top, rcClip.Width( ), rc-
Clip.Height(), PATCOPY);
  pDC ->SelectObject(oldBrush);
  return TRUE;
```

2. 用位图作为应用程序的背景

这个实现过程基本和用位图作为对话框的背景一样，所以不再赘述。

四、基于多文档应用（MDI）的背景的操作

用 VC 的 Appwizard 建立 MDI 框架的时候，会建立一个类，该类名称为 CMainFrame，我们需要重载该类中的方法 On-CreateClient，相关的代码如下：

```
BOOL CMainFrame::OnCreateClient(LPCREATESTRUCT
lpcs, CCreateContext * pContext)
{
    COLORREF m_hWndMDIClient;
  m_hWndMDIClient =RGB(128, 0, 0);  //MDI 背景的颜色
  if (CMDIFrameWnd::OnCreateClient(lpcs, pContext))
  {
//m_hWndMDIClient 是类 CMDIFrameWnd 定义的成员变量，
包含了 MDI 客户窗口的句柄.
    //而类 CMainFrame 继承了类 CMDIFrameWnd.
    // m_wndMdiClient 是自定义类 CMdiClient 的实例，我们
将在下面做详细介绍
//下面一行代码的意思是把发送给窗口 m_hWndMDIClient 的
消息都先由窗口 m_wndMdiClient 来处理，也就是说 m_hWnd-
MDIClient 窗口处理过程变成先处理类 m_wndMdiClient 包含
的窗口过程，然后在处理 m_hWndMDIClient 自己的窗口过程，
这个主要是通过 CWnd 成员函数 SubclassWindow 来实现的.
```

实际上就是一个窗口子类化过程.

```
m_wndMdiClient. SubclassWindow(m_hWndMDIClient);
//设定我们 MDI 程序背景的颜色为 m_clientBkgndRGB
m_wndMdiClient. SetBackColor(m_clientBkgndRGB);
   return TRUE;
      }
      else
      return FALSE;
 }
```

关于 CMdiClient 类的介绍

CMdiClient 将用来处理本来由 m_hWndMDIClient 窗口处理的很多消息，比如 OnEraseBkgnd 消息、OnSize 消息等等。很明显，为了使 CMdiClient 能够处理窗口消息，它必须继承类 CWnd。然后它必须能够处理 OnEraseBkgnd、OnSize 等消息。下面是关于该类的一个定义：

```
class CMdiClient :  public CWnd
{
      DECLARE_DYNCREATE(CMdiClient)
public:
      CMdiClient();
      virtual  ~CMdiClient();
public:
      COLORREF SetBackColor(COLORREF nBackColor);
private:
      COLORREF      m_nBackColor;  //设置背景颜色
protected:
      //{{AFX_MSG(CMdiClient)
      afx_msg BOOL OnEraseBkgnd(CDC* pDC);
      //}}AFX_MSG
      DECLARE_MESSAGE_MAP()
};
```

关于 CMdiClient 类的实现

在类的构造函数中，我们需要初始化变量 m_nBackColor，具体代码如下：

```
CMdiClient: : CMdiClient()
{
            m_nBackColor  = CLR_INVALID;  //  表示我们对 m_nBackColor 初始值的设置
}
```

然后我们需要通过函数 SetBackColor 来设置应用程序的背景颜色 其基本代码如下

```
COLORREF  CMdiClient: :  SetBackColor(COLORREF  nBack-Color)
{
      LockWindowUpdate();
      COLORREF cr  =  m_nBackColor;
      m_nBackColor  =  nBackColor;
      UnlockWindowUpdate();
      return cr;
}
```

在上面这段代码中 LockWindowUpdate 和 UnlockWindowUpdate 的作用是在 LockWindowUpdate 以后，暂时就不能对该窗口中的内容进行刷新、清除或更新，需要等到 UnlockWindowUpdate 以后，才能够对该窗口重新开放操作，有点象操作系统中互斥的概念。

当然最重要的还是重载方法 OnEraseBkgnd，其实现代码如下：

```
BOOL CMdiClient: : OnEraseBkgnd(CDC *  pDC)
{
   CRect rect;
   GetClientRect(& rect);
   BOOL result  =  TRUE;
    if (m_nBackColor ! = CLR_INVALID)
   pDC -->FillSolidRect(& rect,  m_nBackColor);
   else
   result  =  CWnd: : OnEraseBkgnd(pDC);
   return result;
}
```

到现在为止，用户利用上面的代码，就可以建立一个具有不同颜色背景的 MDI 应用程序了。下面我们需要讨论的是如何用位图作为 MDI 的背景。

有了上面的基础以后，用位图作为 MDI 的背景就显得很简单。

首先我们定义一个成员变量 m_bmpBackground，它是类 CMyBitmap 的一个实例。

其次我们在类 CMdiClient 中增加一个方法，称为 SetBitmap

```
void SetBitmap  COLORREF nBackColor
```

其实现代码为：

```
BOOL CMdiClient: : SetBitmap(UINT nID)
{
      LockWindowUpdate();
   m_bmpBackground. LoadResource(nID);
UnlockWindowUpdate();
}
```

然后，改写成员函数 OnEraseBkgnd，具体代码如下 作为示例，我们这里只考虑图片平铺的情况）：

```
BOOL CMdiClient: : OnEraseBkgnd(CDC * pDC)
{
   CRect rc;
   GetClientRect(& rc);
   BOOL result  =  TRUE;
   int x, y;
   x = 0; y = 0;
   while(y < rc. Height())
   {
         while(x < rc. Width())
         {
               m_bmpBackground. DrawBMP(pDC, x, y);
         x = x + m_bmpBackground. GetWidth();
    };
   x = 0;
   y = y + m_bmpBackground. GetHeight()
   }
   return TRUE;
}
```

用 VC 实现树控件背景颜色的改变

胡朝晖

一、实现的方法分析

默认情况下树控件背景颜色是白色的，要改变树控件背景的颜色，我们需要重载方法 OnPaint 因为树控件没有 DrawItem 这样的成员函数提供给我们）。为了说明问题的方便，我们直接对 VC 中自带的例子程序 CmnCtrl1 进行了修改。最重要的是通过 ClassWizard 重载方法 OnPaint。

为了改变树控件的背景，这里就需要谈一点图形转换方面的知识。首先我们来看正常的一个树控件的图形，如图 1 所示：

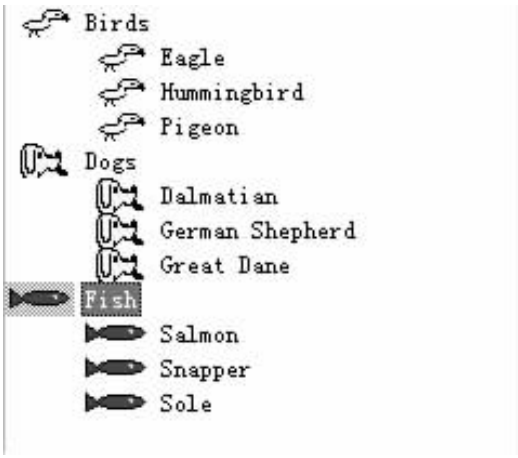


图 1

我们需要的可以带自定义背景颜色的树控件的图形，如下所示：

为了完成这个转化，我们需要三个中间的图形，首先要生成一个只有黑白两色树控件位图，如图 2 所示：

这个通过下面这段代码可以实现上面的这幅位图：

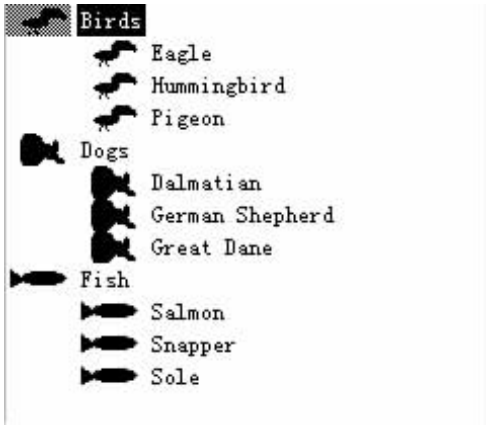


图 2

```
CDC maskDC;
maskDC. CreateCompatibleDC(& dc);
CBitmap maskBitmap;
// 用 CBitmap 类中的成员函数 CreateBitmap 创建一个位图，函数原形如下
//BOOL CreateBitmap( int nWidth, int nHeight, UINT nPlanes, UINT nBitcount, const void * lpBits );
//相关参数说明
//nWidth 表示位图的宽度
//nHeight 表示位图的高度
//nPlanes 表示位图的色彩平面数目
//nBitcount 表示每一点显示的颜色位数，也就是说每一个点，需要用多少位来显示
//lpBits 指向一个数组，该数组包含了最初的位图位值，如果是 NULL 的话，新的位图就不被初始化
//所以下面一行代码创建了一个单色的位图
maskBitmap. CreateBitmap( rcClient. Width(), rcClient. Height(), 1, 1, NULL );
maskDC. SelectObject( & maskBitmap ); //maskDC 选择该位图
```

最后，我们当然需要对类 CMainFrame 中的方法 OnCreateClient 进行重写，相关代码为：

```
BOOL CMainFrame: : OnCreateClient(LPCREATESTRUCT lpcs, CCreateContext * pContext)
{
COLORREF m_hWndMDIClient;
    if (CMDIFrameWnd: : OnCreateClient(lpcs, pContext))
    {
  m_wndMdiClient. SubclassWindow(m_hWndMDIClient);
// IDB_BMPBACKGROUND 为我们在资源中定义的一幅背景位图
  m_wndMdiClient. SetBitmap(IDB_BMPBACKGROUND);
  return TRUE;
  }
  else
  return FALSE;
}
```

到现在为止，只要你选择了一个好的背景位图，你的 MDI 应用程序就有了漂亮的位图作为背景了。

收稿日期：2000 年 4 月 24 日）

```
memDC. SetBkColor( : : GetSysColor( COLOR_WINDOW));
//设置 DC 的背景颜色
```

// 把内存 DC 中的内容拷贝到 mask DC 中去，现在 maskDC 中保存的位图和内存 DC 中的位图内容是一模一样，只是颜色只有黑白两色，能够实现这个两色图的前提是原来的位图的背景应该是白色的，我们可以看到树控件原来的背景颜色确实是白色的. 如果树控件原来的背景颜色不是白色的话，就只能创建一个全黑的位图（而没有黑白两色，这就不能达到我们的目的了）

```
maskDC. BitBlt( 0, 0, rcClient. Width( ), rcClient. Height( ),
& memDC, rcClient. left, rcClient. top, SRCCOPY );
```

然后我们需要实现一个有彩色图标和黑色背景的位图，如图 3 所示：

图 3

图 3 是通过对图 1 加上黑色背景（RGB（0，0，0））和图 2 相与以后得到的，具体代码如下所示：

```
memDC. SetBkColor(RGB(0, 0, 0));
memDC. SetTextColor(RGB(255, 255, 255));
memDC. BitBlt(rcClient. left, rcClient. top, rcClient. Width( ),
rcClient. Height( ), & maskDC, rcClient. left, rcClient. top,
SRCAND);
```

我们来仔细分析上面三句代码的作用，第一行和第二行代码实际上不是为 memDC 所设置的，它们主要还是为 maskDC 服务的，这两行代码告诉系统，当我们用 BitBlt 进行图象的转换操作的时候，先要把 maskDC 的位图的背景颜色的 RGB 设置成 RGB 0 0 0 ，把 maskDC 位图的前景颜色的 RGB 设置成 RGB 255 255 255 ，然后在把 maskDC 中的位图和 memDC 中的位图点点相与得到最后的位图结果。

然后，我们需要得到一幅具有用户自定义背景和黑色图标的位图，如图 4：

这个只需要我们把图 2 和一个具有用户自定义背景图象相与就可以得到，具体代码如下：

```
dc. FillRect(rcClient, & CBrush(RGB(249, 230, 196)) );
//创建用户自定义背景的图象
dc. SetBkColor(RGB(255, 255, 255)); //因为 maskDC 中位图的背景本来就是白色的，所以这行代码也可以不要
```

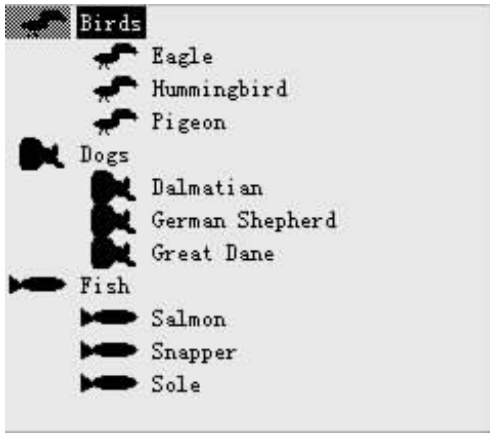


图 4

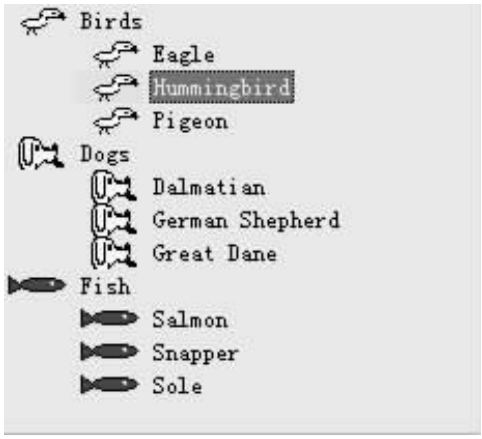


图 5

```
dc. SetTextColor(RGB(0, 0, 0)); //因为 maskDC 中位图的前景本来就是黑色的，所以这行代码也可以不要
dc. BitBlt(rcClient. left, rcClient. top, rcClient. Width( ), rc-
Client. Height( ), & maskDC, rcClient. left, rcClient. top, SR-
CAND);
```

然后把图 3 和图 4 相或，就可以得到图 5，即我们最终所需要的树控件位图的形状，相关代码如下：

```
dc. BitBlt(rcClient. left, rcClient. top, rcClient. Width( ), rc-
Client. Height( ), & memDC, rcClient. left, rcClient. top, SR-
CPAINT);
```

这里初学者容易犯的一个毛病是以为如下的代码就可以实现背景的设置：

```
memDC. SetBkColor(RGB(249, 230, 196));
dc. BitBlt(rcClient. left, rcClient. top, rcClient. Width( ), rc-
Client. Height( ), & memDC, rcClient. left, rcClient. top, SR-
CCOPY);
```

这主要是因为没有正确理解 CDC 类中 SetBkColor 的作用而引起的。事实上，单单一句 SetBkColor 是并不能改变对位图背景的设置，位图颜色、背景的变化只有通过两幅位图的与、或等操作 用 BitBlt 函数 才能实现。

二、OnPaint 的完整的代码

```
void CMyTreeCtrl: : OnPaint( )
{
```

```
    CPaintDC dc(this);
    // 创建一个和显示 DC 相容的内存 DC
    CDC memDC;
    memDC. CreateCompatibleDC( & dc );
    CRect  rcClient;
  //得到控件的尺寸大小
    GetClientRect(& rcClient);
 // 创建一个和显示 DC 配置一样的内存 DC
    CBitmap bitmap;
bitmap. CreateCompatibleBitmap( & dc, rcClient. Width( ), rcClient. Height( ) );
    memDC. SelectObject( & bitmap );
    // 在内存中创建需要刷新部分的区域
    CRgn rgn;
    rgn. CreateRectRgnIndirect( & rcClient);
    memDC. SelectClipRgn(& rgn);
    rgn. DeleteObject();
    // 让控件作完它缺省要绘制的东西
CWnd: : DefWindowProc( WM_PAINT, (WPARAM)memDC. m_hDC, 0 );
// 创建一个 MASK DC
    CDC maskDC;
    maskDC. CreateCompatibleDC(& dc);
    CBitmap maskBitmap;
//下面一行代码创建了一个单色的位图
    maskBitmap. CreateBitmap( rcClient. Width( ), rcClient. Height( ), 1, 1, NULL );
    maskDC. SelectObject( & maskBitmap ); //maskDC 选择该位图
memDC. SetBkColor( : : GetSysColor( COLOR_WINDOW ));
//设置 DC 的背景颜色
// 把内存 DC 中的内容拷贝到 mask DC 中去, 现在 maskDC 中保存的位图和内存 DC 中的位图内容是一模一样, 只是颜色只有黑白两色
maskDC. BitBlt( 0, 0, rcClient. Width( ), rcClient. Height( ), & memDC, rcClient. left, rcClient. top, SRCCOPY );
    // 用客户指定的颜色来填充控件背景
    dc. FillRect(rcClient, & CBrush(RGB(49, 70, 96)) );
    memDC. SetBkColor(RGB(0, 0, 0));   //背景黑色
    memDC. SetTextColor(RGB(255, 255, 255));
memDC. BitBlt(rcClient. left, rcClient. top, rcClient. Width( ), rcClient. Height( ), & maskDC, rcClient. left, rcClient. top, SRCAND);
    dc. SetBkColor(RGB(255, 255, 255));
    dc. BitBlt(rcClient. left, rcClient. top, rcClient. Width( ), rcClient. Height( ), & maskDC, rcClient. left, rcClient. top, SRCAND);
    //把前景和背景相结合, 完成树控件中内容的显示
    dc. BitBlt(rcClient. left, rcClient. top, rcClient. Width( ), rcClient. Height( ), & memDC, rcClient. left, rcClient. top, SRCPAINT);
}
```

实际上 我们也可以采用如下的代码来实现 它的效果与上面的相差无几 但是代码更加简洁.

```
void CMyTreeCtrl: : OnPaint( )
{
  CPaintDC dc(this);
  CDC memDC;
  memDC. CreateCompatibleDC( & dc );
  CRect  rcClient;
   GetClientRect(& rcClient);
  CBitmap bitmap;
  bitmap. CreateCompatibleBitmap( & dc, rcClient. Width( ), rcClient. Height( ) );
  memDC. SelectObject( & bitmap );
  CRgn rgn;
  rgn. CreateRectRgnIndirect( & rcClient);
  memDC. SelectClipRgn(& rgn);
  rgn. DeleteObject();
CWnd: : DefWindowProc( WM_PAINT, (WPARAM)memDC. m_hDC, 0 );
CDC maskDC;
  maskDC. CreateCompatibleDC(& dc);
  CBitmap maskBitmap;
maskBitmap. CreateBitmap(rcClient. Width( ), rcClient. Height( ), 1, 1, NULL );
  maskDC. SelectObject( & maskBitmap ); //maskDC 选择该位图
memDC. SetBkColor( : : GetSysColor( COLOR_WINDOW ));
//设置 DC 的背景颜色
 maskDC. BitBlt( 0, 0, rcClient. Width( ), rcClient. Height( ), & memDC, rcClient. left, rcClient. top, SRCCOPY );
  memDC. SetBkColor(RGB(249, 230,196));  // 直接设置 maskDC 在转化时的背景颜色, 因为 memDC 中的背景是白色, 所以 RGB 值为(255, 255, 255), 当它和 maskDC 中的背景颜色相与(SCRAND) 的时候, 背景就被设定为用户定义的背景颜色
  memDC. SetTextColor(RGB(255, 255255));  // 设置 maskDC 在转化时的前景颜色为白色, 这样在与操作的时候, 主要看 memDC 的前景图案是什么结果就是什么了
memDC. BitBlt(rcClient. left, rcClient. top, rcClient. Width( ), rcClient. Height( ), & maskDC, rcClient. left, rcClient. top, SRCAND);
  dc. BitBlt(rcClient. left, rcClient. top, rcClient. Width( ), rcClient. Height( ), & memDC, rcClient. left, rcClient. top, SRCCOPY);
}
```

最后需要注意的是，为了减少对背景的擦除，需要重载 OnEraseBkgnd 方法，具体代码如下：

```
BOOL CMyTreeCtrl: : OnEraseBkgnd(CDC * pDC)
{
  return TRUE;
  //return CTreeCtrl: : OnEraseBkgnd(pDC);
}
```

以上讨论实现了对树控件的背景颜色的改变，读者可以自己做一下尝试。

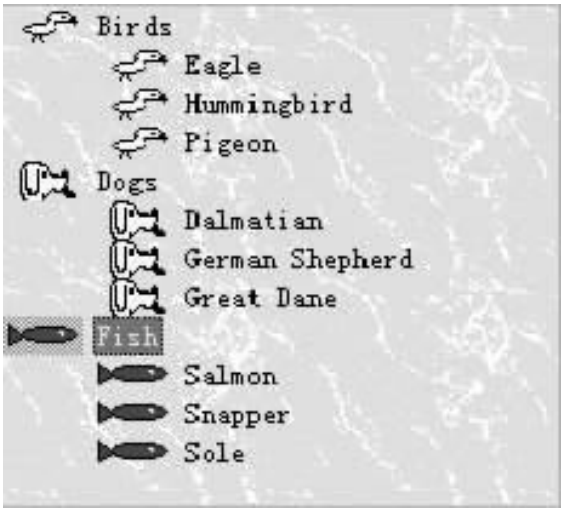


图 6

三、关于如何在树控件中实现把位图作为背景的讨论

一幅具有位图背景的树控件如图 6 所示。

具体实现的步骤如下：

1. 在类 CmyTreeCtrl 中增加几个成员变量，它们的定义如下：

```
CBitmap m_bitmap    //我们要作为背景的位图类
int m_cxBitmap  m_cyBitmap   //位图的大小
```

2. 增加一个成员方法用来设置作为背景的位图，定义如下：

```
BOOL SetBkImage  UINT nIDResource     //这里 nIDResource 为我们在资源中放置的位图的 ID 号。
```

实现代码如下

```
BOOL CMyTreeCtrl::SetBkImage(UINT nIDResource)
{
//是否已经存在对背景位图的定义，如果存在的话，就删除它
    if( m_bitmap.m_hObject != NULL )
        m_bitmap.DeleteObject();
//装载位图，返回位图句柄以便我们可以根据该句柄操作位图
    HBITMAP hBmp = (HBITMAP)::LoadImage( AfxGetInstanceHandle(), (LPCTSTR)nIDResource, IMAGE_BITMAP, 0, 0, LR_CREATEDIBSECTION );
if( hBmp == NULL )
        return FALSE;
//位图类绑定位图句柄，我们就可以方便的在类中操作该位图
    m_bitmap.Attach( hBmp );
    BITMAP bm;
    m_bitmap.GetBitmap( &bm );
    //得到位图的高和宽并保存到类的成员变量中
    m_cxBitmap = bm.bmWidth;
    m_cyBitmap = bm.bmHeight;
    return TRUE;
}
```

3. 重载方法 OnPaint 相关的代码如下

```
void CMyTreeCtrl::OnPaint()
{
CPaintDC dc(this);
CRect rcClip, rcClient;
CDC tempDC;
CDC imageDC;
CDC maskDC;
 CBitmap * pOldMemBitmap;
 CBitmap * pOldMaskBitmap;
CBitmap * pOldTempBitmap;
CBitmap * pOldImageBitmap;
//得到窗口中需要刷新的部分
dc.GetClipBox( &rcClip );
//得到窗口客户区域整个的大小
GetClientRect(&rcClient);
// 创建一个和显示 DC 相容的内存 DC
CDC memDC;
memDC.CreateCompatibleDC( &dc );
// 使内存 DC 的配置和显示 DC 保持一致
CBitmap bitmap, bmpImage;
bitmap.CreateCompatibleBitmap( &dc, rcClient.Width(), rcClient.Height() );
pOldMemBitmap = memDC.SelectObject( &bitmap );
// 让控件完成缺省应该做的绘制工作
CWnd::DefWindowProc( WM_PAINT, (WPARAM) memDC.m_hDC, 0 );
 // 如果我们定义了背景的位图的话，下面的判断式为真
if( m_bitmap.m_hObject != NULL )
{
// 创建一个只有黑白两色的 DC，这个 DC 包含的位图很有意思，它具有黑色的图象和白色的背景。注意黑色的 RGB 为 (0, 0, 0)，白色的 RGB 为 (255, 255, 255)。所以如果把这幅图和一幅和它一样大小，但只有背景位图的图象相与的话，就能产生一幅具有背景位图但图象为黑色的位图。把它和一幅有同样内容但有黑色背景和彩色图形的位图相与的时候就得到一个具有黑色背景但有彩色图形的位图。所以这个 DC 中的位图起到了图形颜色背景转化中的类似 agent 的作用。
        maskDC.CreateCompatibleDC(&dc);
        CBitmap maskBitmap;
        // 为 maskDC 创建相应的位图
maskBitmap.CreateBitmap(rcClient.Width(), rcClient.Height(), 1, 1, NULL );
pOldMaskBitmap = maskDC.SelectObject( &maskBitmap );
memDC.SetBkColor( ::GetSysColor( COLOR_WINDOW ) );
// 使 maskDC 中位图的内容和 memDC 保持一致，但注意 maskDC 中位图只有黑白两色
maskDC.BitBlt( 0, 0, rcClient.Width(), rcClient.Height(), &memDC, rcClient.left, rcClient.top, SRCCOPY );
//创建一个临时性的 tempDC，该 DC 的位图就是我们自定义的背景位图
tempDC.CreateCompatibleDC(&dc);
pOldTempBitmap = tempDC.SelectObject( &m_bitmap );
CBitmap bmpImage;
//创建一个和显示 DC 配置一样的 imageDC
imageDC.CreateCompatibleDC( &dc );
```

```
bmpImage. CreateCompatibleBitmap( & dc, rcClient. Width(),
rcClient. Height() );
pOldImageBitmap = imageDC. SelectObject( & bmpImage );
CRect rcRoot;
GetItemRect( GetRootItem(), rcRoot, FALSE );
rcRoot. left = -GetScrollPos( SB_HORZ );
//下面循环的目的就是把我们自定义的位图背景平铺到 imageDC 上去
for( int i = rcRoot. left; i < rcClient. right; i += m_cxBitmap )
for( int j = rcRoot. top; j < rcClient. bottom; j += m_cyBitmap )
imageDC. BitBlt( i, j, m_cxBitmap, m_cyBitmap, & tempDC, 0, 0, SRCCOPY );
//创建一个具有黑色背景和彩色图标(文字)的树控件位图
  memDC. SetBkColor(RGB(0, 0,0)); //设置 maskDC 背景为黑色
memDC. SetTextColor(RGB(255, 255, 255));
//设置 maskDC 前景为白色
memDC. BitBlt(rcClip. left, rcClip. top, rcClip. Width( ), rcClip. Height(), & maskDC,
rcClip. left, rcClip. top, SRCAND);
//可替换代码段
//创建一个具有我们自己定义的位图背景和黑色图标(文字)的树控件位图
//imageDC. SetBkColor(RGB(255, 255, 255)); 因为 maskDC 背景本来就是白色的
//imageDC. SetTextColor(RGB(0, 0, 0)); 因为 maskDC 前景本来就是黑色的
imageDC. BitBlt(rcClip. left, rcClip. top, rcClip. Width( ), rcClip. Height(), & maskDC,
rcClip. left, rcClip. top, SRCAND);
//把前景彩色背景黑色的位图和前景黑色背景彩色的两幅相或,就得到一幅前景和背景都彩色的位图,这就是我们要显示的树控件位图
imageDC. BitBlt(rcClip. left, rcClip. top, rcClip. Width( ), rcClip. Height(), & memDC, rcClip. left, rcClip. top, SRCPAINT);
// 上面的代码使 imageDC 有了我们所需要的位图,现在把它拷贝到显示 DC 上,这样就能在应用程序中看到该幅位图了
dc. BitBlt(rcClip. left, rcClip. top, rcClip. Width(), rcClip. Height(), & imageDC, rcClip. left, rcClip. top, SRCCOPY );
//可替换代码段
      }
      else
      {
//如果没有定义位图背景的话,简单拷贝 memDC 的内容到显示 DC 上就 OK
    dc. BitBlt( rcClip. left, rcClip. top, rcClip. Width(), rcClip. Height(), & memDC, rcClip. left, rcClip. top, SRCCOPY );
      }
//所有在本程序段中创建的 DC 依然回到以前的配置上去,保证操作系统能够正常的删除这些 CDC 变量,以便有效利用 stack 空间,防止内存溢出.
   memDC. SelectObject(pOldMemBitmap);
   maskDC. SelectObject(pOldMaskBitmap);
   tempDC. SelectObject(pOldTempBitmap);
   imageDC. SelectObject(pOldImageBitmap);
  }
```

实际上我们可以采用另一种方法实现两幅位图的叠加 可以通过把可替换代码段之间的代码用如下的代码来实现

```
    imageDC. BitBlt(rcClip. left, rcClip. top, rcClip. Width(), rcClip. Height(), & memDC, rcClip. left, rcClip. top, SRCINVERT);
    ImageDC. BitBlt(rcClip. left, rcClip. top, rcClip. Width(), rcClip. Height( ), & maskDC, rcClip. left, rcClip. top, SRCAND);
  imageDC. BitBlt(rcClip. left, rcClip. top, rcClip. Width(), rcClip. Height( ), & memDC, rcClip. left, rcClip. top, SRCINVERT);
dc. BitBlt(rcClip. left, rcClip. top, rcClip. Width( ), rcClip. Height( ), & imageDC, rcClip. left, rcClip. top, SRCCOPY );
```

其原理说明如下

背景操作为

后面的位图 XOR 前面的位图 AND RGB 255 255 255 XOR 前面的位图

= = 后面的位图 XOR 前面的位图 XOR 前面的位图

= = 结果 等于后面位图

前景操作为

后面的位图 XOR 前面的位图 AND RGB 0 0 0 XOR 前面的位图

= = RGB 0 0 0 XOR 前面的位图

= = 结果 等于前面位图

注意这里的后面的位图就是我们自己定义的背景位图,前面的位图就是树控件的位图

4. 对消息 OnItemexpanding、OnVScroll、OnHScroll 进行重载处理,目的是为了保证更好的图形显示效果。具体的代码如下:

```
void CMyTreeCtrl: : OnItemexpanding(NMHDR * pNMHDR, LRESULT * pResult)
  {
NM_TREEVIEW * pNMTreeView = (NM_TREEVIEW * ) pNMHDR;
  if( m_bitmap. m_hObject ! = NULL )
        InvalidateRect(NULL);
   * pResult = 0;
  }
void CMyTreeCtrl: : OnVScroll(UINT nSBCode, UINT nPos, CScrollBar * pScrollBar)
  {
  if( m_bitmap. m_hObject ! = NULL )
  InvalidateRect(NULL);
  CTreeCtrl: : OnVScroll(nSBCode, nPos, pScrollBar);
  }
void CMyTreeCtrl: : OnHScroll(UINT nSBCode, UINT nPos, CScrollBar * pScrollBar)
```

```
{
    if( m_bitmap.m_hObject != NULL )
        InvalidateRect(NULL);
    CTreeCtrl::OnHScroll(nSBCode, nPos, pScrollBar);
}
```

5. 为了减少对背景的擦除而引起的图象的闪烁 需要重载 OnEraseBkgnd 方法 具体代码如下

```
BOOL CMyTreeCtrl::OnEraseBkgnd(CDC* pDC)
{
    if( m_bitmap.m_hObject != NULL )
        return TRUE;
    return CTreeCtrl::OnEraseBkgnd(pDC);
    //return CTreeCtrl::OnEraseBkgnd(pDC);
}
```

6. 为了方便说明，我们直接在类的构造函数中加入函数 SetBkImage，当然实际上也可以由使用该类的代码在外部调用该方法来设置树控件背景位图。这里我们只是为了说明的方便而已。具体代码如下：

```
CMyTreeCtrl::CMyTreeCtrl()
{
    ..........
 SetBkImage(IDB_BITMAP1); //设置背景位图
}
```

收稿日期 2000 年 12 月 21 日

液晶显示器走进家庭

著名显示器生产厂商 LG 电子日前在京宣布，其在中国显示器市场上推出的两款液晶显示器已全面降价，降价幅度高达 2000 元以上。如此大幅度的降价行为，在国内显示器市场还是非常少见的。此举预示着液晶显示器将走进寻常百姓家中。

据悉，此次 LG 电子降价活动所涉及到的两款液晶显示器分别为普通型 570LS 原价 8500 元降至 6999 元和高档型 880LC 原价 26000 元降至 19999 元，这两款显示器在中国市场上都有着非常突出的表现，尤其是 880LC，在全球液晶显示器高端产品市场上，都占据着非常重要的位置。据 LG 电子中国有限公司 IT 产品营销总部市场部官员称，此次 LG 液晶显示器降价的目的是希望能够推动液晶显示器在各类用户，尤其是普通用户中的普及，通过 LG 电子的努力，让更多的人享受到液晶显示器的优越性能和高贵品质，让液晶显示器真正进入寻常百姓的家里。

作为全球液晶显示器销量第一的厂商，LG 电子此次降价，完全是从满足用户对液晶显示器强烈需求的目的出发的。LG 电子发展的宗旨是利用先进的数码产品来推动社会的发展，实现高质量的现代生活。因此 LG 电子不仅从产品技术和质量上苦下功夫，而且在保证自身利益的情况下，最大限度的向广大用户让利，希望以此来推动液晶显示器行业的发展，让更多的用户能够有机会使用液晶显示器，从而实现 LG 电子所倡导的精彩的数码生活。

上接第 17 页

```
一个块引用表行被找到后，检查它是否有属性
If StrComp(.EntityName, "AcDbBlockReference", 1) = 0 Then
        '如果有属性
        If . HasAttributes Then
        '提取块引用中的属性
        Array1 = .GetAttributes
        '这一轮循环用来查找标题，如果有填在第 1 行
        For Count = LBound(Array1) To UBound(Array1)
'如果还没有标题
If Header = False Then
'作为标题的明细行其块属性常设为 Constant 类型
If Array1(Count).Constant Then
ExcelSheet.Cells(RowNum, Count + 1).Value _
= Array1(Count).TextString
                    End If
                End If
            Next Count
'从第 2 行开始，填写其它的明细表行内容
RowNum = RowNum + 1
For Count = LBound(Array1) To UBound(Array1)
ExcelSheet.Cells(RowNum, Count + 1).Value _
            = Array1(Count).TextString
            Next Count
            Header = True
        End If
    End If
End With
Next blkElem
'对填入当前表单的内容，按第 1 列进行排序，
'范围是从 A1 单元格开始的整个工作表
Excel.Worksheets("Sheet1").Range("A1").Sort _
  key1:=Excel.Worksheets("Sheet1").Columns("A"), _
  Header:=xlGuess
'显示 Excel 工作表中的结果
Excel.Visible = True
'该语句用来等待查看显示结果
MsgBox "按 '确定' 键将关闭 Excel 的运行!"
'保存传过来的数据
ExcelWorkbook.Save
'关闭 Excel 应用程序
Excel.Application.Quit
'删除 Excel 应用程序实例
Set Excel = Nothing
End Sub
```

运行上述代码后，将在 "My Documents" 文件夹下生成一 '属性表.xls" 文件。由于在 attrib.dwg 文件中，其明细表中第一行标题的文字不是块属性，而是文本对象，所以在 '属性表.xls" 文件中的第 1 行为空。不过在 Excel 界面下要编写一行标题是非常容易的。在多数情况下，作为标题的明细表行是不希望随便改动的，所以标题行的块属性往往被设成固定不变 Constamt 类型。在 ActiveX 中的 Attribute 和 AttributeRef 对象，都有一个 Constsnt 属性，用来判断某个块或块引用中的属性值类型，它是一个布尔类型的值，其值若为 True，表示块属性值为 Constsnt 类型。

本代码在 Windows95 hAutoCadR2000 上运行通过。

收稿日期：2001 年 4 月 24 日）

用 VC 编制函数图形的显示类

武学师　石江涛

摘　要　本文提出了利用 VC 编制函数图形显示类的原理、实现方法并给出了实现过程中的一些程序。

关键词　函数图形显示，VC，OPEN GL，面向对象技术

一、引言

在某些工程应用系统中经常要对已知表达式求出其图形，由于要求运算速度等原因，一般不能调用象 Matlab 等绘图工具，如在课题“微波分析系统”中，主程序能模拟动态的微波仿真，而其中的一个分枝要提供某些已知函数的显示，因为某些原因系统不能调用 Matlab，只能利用 VC 对这些函数进行二维或三维的显示。于是在课题中我们将显示的图形进行类的编制，使得画图的工作变得很轻松。下面介绍函数图形类的编制。

二、二维显示要求的函数显示类的编制

一些函数要求显示的最终结果为二维的曲线，这些函数可以有一个或多个变量，因为是要求显示二维的曲线，因此可以简化为只有一个变量型的函数，如：y = f x 。

这些类型的实现的原理是将给出的函数用一线性映射到物理设备坐标上，并求出逆映射，由物理设备坐标调用实际逻辑坐标进行计算绘制二维或三维图线。

可以利用 VC 的 GDI 的 CDC 类来实现，并可以构造一个类 MyCDC 继承基类 CDC，所说明类似如下：

```
class MyCDC : public CDC{
public:
//.....
 //数据成员,函数成员的说明
 ..............
//说明自己的函数
void DrawChart(float( * func)(float), float a, float b, float c);
 .........
 //其他
 .....
}
```

函数 DrawChart 利用了函数指针将形如如下的形式 假设显示平方函数的图形 ：

```
float func(float x)
 {
   return x * x;
 }
```

的函数作为变量传给 DrawChart 其中参数 a b 为要显示的区间 C 为纵坐标要放大的倍数 因此只要知道函数的近似表达式 就很容易显示出结果。

函数 DrawChart 的关键程序如下

```
y = (int)(c * (func(a)));
y = 300 - y;  //调整坐标
MoveTo(100, y); //建立映射起点 100, 300
for(i = 102; i < 500; i = i + 2){       //利用逆映射计算并绘图
        yx = a + (i - 100. 0) * (b - a)/400. 0;
        y = (int)(c * func(yx) );
        y = 300 - y;
       LineTo(i, y);   //画函数图线
   }
    for(i = -250; i < = 250; i + = 50){
          xx = ((float)i/(c));
          x = (int)(xx * 1000);
          wsprintf(s, "%d", x);
        TextOut(100, 300 - i, s); //绘制坐标
    }
```

上段程序很容易以下列形式调用：

```
MyCDC dc  this
dc. DrawChart  func  a  b  c  ;
```

以下调用的形式基本类似，故省略。

以上只是在直角坐标系下 很容易将其扩充到极坐标系 如下的画方向图的函数

```
void DrawDirection  float  * func   float   float c
```

同样 func 仍为函数指针 C 为放大倍数 其内部操作大致为

```
   max = 0. 0; // max 为一变量记录最大值
  r = c * (func(0));
  if(max < r)max = r;
  x = int(fabs(r));
  y = 0;
  x = 300 + x;
  y = 300 - y;
 MoveTo(x, y);
 //从 0 度画到 360 度
for(i = 1; i < 360; i + + ){
     s = (float)i * 3. 1416/180. 0;
    r = c * (func(s)); r = fabs(r);
    if(max < r)max = r;
    x1 = r * cos(s); y1 = r * sin(s);
    x = (int)x1; y = (int)y1;
    x = 300 + x;
     y = 300 - y;
    LineTo(x, y);
 }
```

```
 r = max/4; s = 0;
//画出坐标的分度
 for(i = 1; i < = 4; i + + ){
       s = s + r;
       R = (int)s;
     Arc(300 - R, 300 - R, 300 + R, 300 + R, 300 + R, 300,
300 - 1 + R, 300 + 1);
 }
```

图 1 是利用上述程序绘制用 Hallen 积分方程求线长为 L 的阵子的电流振幅分布的图示，图 2 是绘制一激励天线阵子 L = λ/5 的电场 E 的方向图 极坐标 。

三、三维显示要求的函数显示类的编制

还有一些函数要求显示的最终结果为三维的曲线，这些函数可以有二个或多个变量，因为是要求显示三维的曲线，因此可以简化为只有一个变量型的函数，如：y = f z x 。

显示三维的图形就要用到调用 OPEN GL 接口，首先将 VC 中的 View 类进行修改，使之能够调用并显示 OPEN GL 接口函数，关于这方面的技术请参考有关的文章。

可在修改过的视类中定义一 Draw3D 函数：

```
void Draw3D float * func  float  float a1 float b1  float a2 float
b2 float c
```

函数内部的实现过程如下：

```
for(i = 0; i < = (b1 - a1) * N; i + + ){ //利用逆映射计算并绘图
          glBegin(GL_LINE_STRIP);
          for(j = 0; j < = (b2 - a2) * N; j + + ){
               x = (GLfloat)( a1 + i/N);
               z = (GLfloat)( a2 + j/N);
          y = (GLfloat)(func(x, y));
               glVertex3f(x, y, z);
          }
          glEnd();
     }
          for(j = 0; j < = (b2 - a2) * N; j + + ){
          glBegin(GL_LINE_STRIP);
          for(i = 0; i < = (b1 - a1) * N; i + + ){
               x = (GLfloat)( a1 + i/N);
               z = (GLfloat)( a2 + j/N);
          y = (GLfloat)(func(x, y));
               glVertex3f(x, y, z);
          }
          glEnd();
```

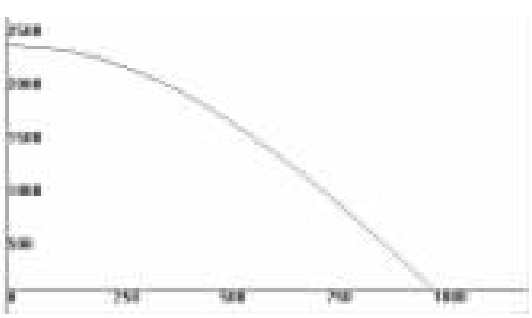


图 1

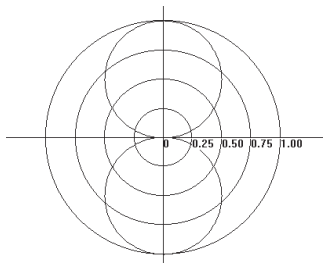


图 2

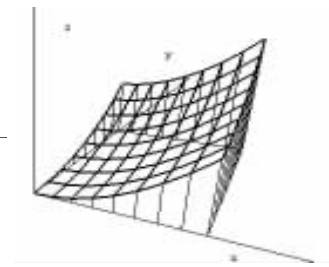


图 3

```
}
```

图 3 是利用上述程序实现函数

$z = x^2 + y^2$

在区间 x 0..1 y 0..1 z 0..1 上的图线 .

四、消隐技术的实现

从图 3 看来，并没有实现图形的消隐，这样画出的图形很难识别清楚，如果在原图形中加入消隐技术，图形将十分清楚。

在 OpenGL 中，可以利用以下的技术实现消隐：

每次求显示的数据时求出函数的两列 如果速度要求很快的话，将所求的数据放到二维数组中 ，在课题中我们将数据放到了数组中,这样在求数值时便不用什么顺序了。

显示时要求顺序,每次读数组中的两列数值,这样一次可得到 4 个点，将这 4 个点用面进行连接，便形成一个消隐面，然后在面上用粗线进行绘制，这样从原理上便能隐藏视线后的点，最后关键的一步启动光源，当画到屏幕上时便自然形成了消隐的图形。 程序略

图 4 为一个波源场的消隐图形。

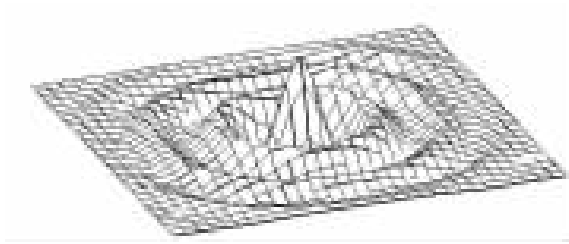


图 4

五、 总结

利用 VC 很容易编写出很精确的二维和三维函数图形显示的类，本文只是抛砖引玉，相信会给读者带来启示。

参考文献

1. Visual C + + 技术内幕 . 清华大学出版社
2. Open GL 从入门到精通 . 电子工业出版社

收稿日期：2000 年 12 月 25 日）

编程安装 Windows 9x 的汉字输入法

许泽民

摘 要 本文探讨了 Windows 9x 的多语言机制并提出一种通过编程安装汉字的键盘输入法的方法。

关键词 Unicode，多语言，汉字输入法，注册表

一、VB Windows 9x 的多语言机制

为了适应高速发展的信息技术的需要，Microsoft Windows 从 3. x 版升级至 95 版，实现了历史性的飞跃，使它从多语言分立的、以 DOS 为平台的 16 位操作环境逐渐向多语言同一界面的、16 位与 32 位兼容的独立操作系统过渡，这一过渡至它的 98 版本已得到基本完善，最近发行的它的最新版本 Windows. Me 则主要加强了网络和多媒体方面的功能。Windows 实现这一过渡靠的主要是引入了 Unicode 和语言及国家 / 地域的概念。

Unicode 是美国的 Unicode 公司开发的、经国际标准化组织 (ISO) 认可的字符编码方案。众所周知，还在不久以前，当计算机主要被应用于欧美地区时，它用以处理字符的编码遵循 ANSI (美国国家标准委员会) 标准，即 ASCII (美国标准信息交换码)。这是一种单字节的七位字符编码方案，主要用于英语地区。ASCII 通过扩展后成了八位，涵盖了欧洲拉丁语系各种语言和日常应用的字符，但数量最多只能表示 256 个，称为单字节字符集 (SBCS)。随着信息技术的发展和计算机在全世界的逐渐普及，SBCS 显然远远不够用，特别是对于东亚各国和地区 (如中国、日本等使用象形文字的地方来说更是这样。于是就出现了所谓的多字节字符集 (MBCS)，而东亚地区用的是双字节字符集 (DBCS)。DBCS 用数字 0 - 128 表示 ANSI 字符，其它大于 128 的数字被用作前导字节字符，但它并不是真正的字符，只是简单地表明下一个字符属于非拉丁字符集。在 DBCS 中，ASCII 字符的长度是一个字节，而一般的东亚字符长度是 2 个字节。DBCS 主要用于 Windows 的 16 位编码部分 (注意：Windows 9x 是一个 16 与 32 位兼容的操作系统)。为了更好地实现国际化及向 32 位过渡，Windows 最终引入了 Unicode。在 Unicode 中，每一个字符 (包括 ANSI 字符) 都用两个字节表示，所以它能表示的字符最大数量是 65536 个，基本上涵盖了世界上所有语言的所有字符，而 ISO 也为这些字符作了具体的、唯一的编码，且有一定的位置剩余。在所有 32 位版本的 Windows 中，它的部件对象模型 (COM) 都是使用 Unicode 的，也就是说 Windows 的底层支持 Unicode。但是，ANSI、DBCS 和 Unicode 是完全不同的三种字符编码方案，为了用户编程和处理字符的方便，以 Windows 为平台的多种编程语言 (如 VB) 已嵌入了一定程度的自动转换机制，在适当时候在底层自动进行这三种编码的互相转换。

Windows 成为多语言操作系统所依赖的另一重要概念是语言和国家 / 地域，具体说来就是地域标识或语言标识，即 LocaleID 或 LCID，类似于以前用于 DOS 中的代码页 (CP 或 Code Page，只为 ANSI 所用，已被 Unicode 弃用)。LCID 表示一个特定地理区域内使用的一种语言及语言习惯。例如，大中华地区通用的都是汉语，但由于习惯有所区别，就被 Windows 分成三个 LCID，即大陆地区、台湾地区和香港地区，如果加上也使用汉语的新加坡，中文就有了四个 LCID。LCID 由唯一的 16 位十六进制数表示，如前述四个地区的 LCID 分别是 0x0804、0x0404、0x0c04 和 0x1004。迄今为止，Windows 安装或支持的 LCID 已达 100 多个，且其数量还在增加。

二、Windows 9x 中的汉字输入法

Unicode 和 LCID 为 Windows 处理多语言提供了基础。但是，要从键盘上输入各种字符，还得借助于另一个概念或技术，这就是键盘布局 (Keyboard Layout)。显然，不同的语言要用不同的键盘布局输入其字符。与 LCID 相应，Windows 也用具有唯一性的键盘布局标识来表示不同的键盘布局，即 Keyboard Layout ID 或 KLID。KLID 是一个双字的字符串，其低位字是 LCID 的十六进制数字值字符串，而高位字则是所谓的 “设备标识符”。于是，正如人们可能会猜到的那样，同一种语言 (地域) 的不同输入法就体现在这个 “设备标识符” 上了。这种情形全部反映在 Windows 各种 32 位版本的注册表中，可以用 REGEDIT 等查看。例如，在 HKEY_LOCAL_MACHINE hSystem hCurrentControlSet hControl hKeyboard Layouts 键下，我们可以看到中文中国大陆地区的 “全拼输入法” 的 KLID 是 E0010804、“双拼输入法” 为 E0020804，美国英语为 00000409 等等。

三、Windows 9x 中汉字键盘输入法的安装

在 Windows 9x 中安装汉字输入法最简单的方法莫过于用

它本身提供的“输入法生成器”了。但是，如果要开发一个作商业用途的或即使不作商业用途而是在朋友间互相交流的输入法，难道每次都要用它来安装吗？这显然是不方便的。为此目的，笔者经过分析研究，找到了编程安装汉字输入法的一种方法，下面以较简单的编程语言 Visual Basic 为例加以说明。但是，在此有两个问题得预先澄清一下：一是 VB 很难称得上是“纯粹”的编程语言，用它开发的程序运行时需要很多附带的库文件，而且往往还要有一个安装过程，对程序的小型化和便于安装不利。为了尽量避免这种情况，本文所谈的程序例是用 VB 5 在 Windows 98SE 上）开发的，只做成一个较小的可执行文件 （Setup. exe）加上两个输入法文件，可以装在一张 3. 5 英寸软盘上。但即使这样，它也只能在 Windows 98 而不能在没有装上 VB 或其运行时库文件的 Windows 95 /97 上运行。二是两个输入法文件是用 Windows 的“输入法生成器”制作的，由于 Windows 95 /97 与 98 两者的“输入法生成器”有所不同，所以，用前者制成的输入法文件可以在后者中使用，但反过来却不行，读者在试验时请注意。此外，笔者所安装的输入法是按照王永民先生发明的五笔字型输入法的编码原则自行辑录词条编成的 具体方法见“输入法生成器”的“帮助”说明），两个输入法文件的名字分别为 Winwb. ime 和 Winwb. mb，输入法的名称为“许辑五笔”。下面是具体的编程思路概述。

据分析，要把输入法安装成功，得经过四个步骤：

1. 在注册表的 HKEY_LOCAL_MACHINE hSystem hCurrentControlSet hControl hKeyboard Layouts 键下创建一个 KLID，并设置相应各值。三个值的值名、值类型 均为以 Null 结尾的字符串，即源程序中的常量 REG_SZ 所表示）和值数据分别见于源程序中；

2. 在注册表的 HKEY_CURRENT_USER hkeyboard layout h preload 键下获取一个未被占用的键盘布局预加载编号 从 1 开始的顺序数字字符），创建此键并设置一个值使它与上一步刚刚创建的 KLID 联系起来；

3. 把两个输入法文件复制到 Windows 的系统目录 （system）中；

4. 加载键盘布局 即刚安装的输入法）。

由此可见，编程安装汉字输入法的整个过程的很大一部分都是对注册表的操作。具体过程见附后的源程序和源程序中较为详细的注释。 注：源程序中有一进度条 ProgressBar1，应通过 VB“工程”菜单中的“部件”把“Microsoft Windows Common Controls 5. 0”添加到工程中。）

另附已编译好的安装程序、两个输入法文件和源程序文件。用 WinZip 解压缩后可在任何同一目录或同一软盘上运行。源程序文件经解压缩后保存在与输入法文件相同的目录中，在装有 VB 的电脑中双击它并重建“工程”文件、重构进度条后即可直接运行。）

附：源程序代码 （Form1. frm）

```
VERSION 5. 00
Object = " {6B7E6392 – 850A – 101B – AFC0 – 4210102A8DA7}#1. 3#0"; "COMCTL32. OCX"
Begin VB. Form Form1
   Caption = "许辑五笔输入法安装程序"
   ClientHeight = 5595
   ClientLeft = 60
   ClientTop = 345
   ClientWidth = 9000
   ControlBox = 0 'False
   Icon = "WINWB. frx": 0000
   LinkTopic = "Form1"
   LockControls = – 1 'True
   ScaleHeight = 5595
   ScaleWidth = 9000
   StartUpPosition = 2 'CenterScreen
   Begin VB. Frame Frame1
      BeginProperty Font
         Name = "楷体 _GB2312"
         Size = 14. 25
         Charset = 134
         Weight = 400
         Underline = 0 'False
         Italic = 0 'False
         Strikethrough = 0 'False
      EndProperty
      ForeColor = & H00C00000&
      Height = 3495
      Left = 1253
      TabIndex = 3
      Top = 600
      Width = 6495
      Begin ComctlLib. ProgressBar ProgressBar1
         Height = 255
         Left = 375
         TabIndex = 10
         Top = 2280
         Width = 5760
         _ExtentX = 10160
         _ExtentY = 450
         _Version = 327682
         Appearance = 1
      End
      Begin VB. Label Label3
         AutoSize = – 1 'True
         Caption = "还需 25 秒"
         ForeColor = & H00404040&
         Height = 180
```

```
    Left            =   360
    TabIndex        =   6
    Top             =   2760
    Visible         =   0   'False
    Width           =   720
  End
  Begin VB.Label Label2
    AutoSize        =   -1  'True
    Caption         =   "辑录人 许泽民"
    BeginProperty Font
      Name          =   "楷体_GB2312"
      Size          =   12
      Charset       =   134
      Weight        =   400
      Underline     =   0   'False
      Italic        =   0   'False
      Strikethrough =   0   'False
    EndProperty
    ForeColor       =   &H00C00000&
    Height          =   240
    Left            =   4200
    TabIndex        =   5
    Top             =   2880
    Width           =   1560
  End
  Begin VB.Label Label1
    BeginProperty Font
      Name          =   "楷体_GB2312"
      Size          =   18
      Charset       =   134
      Weight        =   400
      Underline     =   0   'False
      Italic        =   0   'False
      Strikethrough =   0   'False
    EndProperty
    ForeColor       =   &H00C00000&
    Height          =   2655
    Left            =   360
    TabIndex        =   4
    Top             =   480
    Width           =   5775
  End
End
Begin VB.CommandButton Command1
  Caption           =   "继续"
  Height            =   615
  Left              =   1493
  TabIndex          =   2
  Top               =   4440
  Visible           =   0   'False
  Width             =   1215
End
Begin VB.CommandButton Command2
  Caption           =   "退出"
  Height            =   615
  Left              =   3953
  TabIndex          =   1
  Top               =   4440
  Visible           =   0   'False
  Width             =   1215
End
Begin VB.CommandButton Command3
  Caption           =   "说明"
  Height            =   615
  Left              =   6293
  TabIndex          =   0
  Top               =   4440
  Visible           =   0   'False
  Width             =   1215
End
Begin VB.Label Label6
  AutoSize          =   -1  'True
  Caption           =   "说明"
  Height            =   180
  Left              =   8040
  TabIndex          =   9
  Top               =   3960
  Width             =   360
End
Begin VB.Label Label5
  AutoSize          =   -1  'True
  Caption           =   "退出"
  Height            =   180
  Left              =   8040
  TabIndex          =   8
  Top               =   3480
  Width             =   360
End
Begin VB.Label Label4
  AutoSize          =   -1  'True
  Caption           =   "继续"
  Height            =   180
  Left              =   8040
  TabIndex          =   7
  Top               =   3000
  Width             =   360
End
End
Attribute VB_Name = "Form1"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = False
Attribute VB_PredeclaredId = True
```

```
Attribute VB_Exposed = False
    ' - - - - - - - - - - - - - - - - - - - - - - - - - - - -
    ' 本程序设计用来在 Windows 98 的多语言版本上安装一种
    ' 汉字输入法
    ' - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Option Explicit
Option Compare Text
    ' - - - - - - - - - - - - - - - - - - - - - - - - - - - -
    ' 本程序用到的 Win32 API 函数
    ' - - - - - - - - - - - - - - - - - - - - - - - - - - - -
    '函数: 用于获取 Windows 的安装目录
Private Declare Function GetWindowsDirectory Lib "kernel32" Alias "GetWindowsDirectoryA" (ByVal lpBuffer As String, ByVal nSize As Long) As Long
'函数: 用于在注册表中创建新键
Private Declare Function RegCreateKeyEx Lib "advapi32.dll" Alias "RegCreateKeyExA" (ByVal hKey As Long, ByVal lpSubKey As String, ByVal reserved As Long, ByVal lpClass As String, ByVal dwOptions As Long, ByVal samDesired As Long, lpSecurityAttributes As SECURITY_ATTRIBUTES, phkResult As Long, lpdwDisposition As Long) As Long
    '类型: 用于 RegCreateKeyEx 函数
Private Type SECURITY_ATTRIBUTES
    nLength As Long
    lpSecurityDescriptor As Long
    bInheritHandle As Long
End Type
'函数: 用于在注册表中设置键值
Private Declare Function RegSetValueEx Lib "advapi32.dll" Alias "RegSetValueExA" (ByVal hKey As Long, ByVal lpValueName As String, ByVal reserved As Long, ByVal dwType As Long, lpData As String, ByVal cbData As Long) As Long
'函数: 用于关闭注册表已打开的键
Private Declare Function RegCloseKey Lib "advapi32.dll" (ByVal hKey As Long) As Long
'函数: 用于打开注册表的键
Private Declare Function RegOpenKeyEx Lib "advapi32.dll" Alias "RegOpenKeyExA" (ByVal hKey As Long, ByVal lpSubKey As String, ByVal ulOptions As Long, ByVal samDesired As Long, phkResult As Long) As Long
'函数: 用于枚举注册表的键
Private Declare Function RegEnumKeyEx Lib "advapi32.dll" Alias "RegEnumKeyExA" (ByVal hKey As Long, ByVal dwIndex As Long, ByVal lpName As String, lpcbName As Long, ByVal lpReserved As Long, ByVal lpClass As String, lpcbClass As Long, lpftLastWriteTime As FILETIME) As Long
'类型: 用于 RegEnumKeyEx 函数
Private Type FILETIME
    dwLowDateTime As Long
    dwHighDateTime As Long
End Type
'函数: 用于查询注册表键的设置值
Private Declare Function RegQueryValueEx Lib "advapi32.dll" Alias "RegQueryValueExA" (ByVal hKey As Long, ByVal lpValueName As String, ByVal lpReserved As Long, lpType As Long, ByVal lpData As String, lpcbData As Long) As Long
'函数: 用于获取 Windows 版本
Private Declare Function GetVersionEx Lib "kernel32" Alias "GetVersionExA" (lpVersionInformation As OSVERSIONINFO) As Long
    '常数: 出错代码(成功)
Const ERROR_SUCCESS = 0&
'常数: 注册表各主键
Const HKEY_CLASSES_ROOT = &H80000000
Const HKEY_CURRENT_USER = &H80000001
Const HKEY_LOCAL_MACHINE = &H80000002
Const HKEY_USERS = &H80000003
Const HKEY_PERFORMANCE_DATA = &H80000004
'常数: 用于 LoadKeyboardLayout 函数
Const KLF_ACTIVATE = 0&
'常数: 用于操作 Windows 注册表
Const SYNCHRONIZE = &H100000
Const STANDARD_RIGHTS_READ = &H20000
Const STANDARD_RIGHTS_WRITE = &H20000
Const STANDARD_RIGHTS_EXECUTE = &H20000
Const STANDARD_RIGHTS_REQUIRED = &HF0000
Const STANDARD_RIGHTS_ALL = &H1F0000
Const KEY_QUERY_VALUE = &H1
Const KEY_SET_VALUE = &H2
Const KEY_CREATE_SUB_KEY = &H4
Const KEY_ENUMERATE_SUB_KEYS = &H8
Const KEY_NOTIFY = &H10
Const KEY_CREATE_LINK = &H20
Const KEY_READ = (( STANDARD_RIGHTS_READ Or KEY_QUERY_VALUE Or KEY_ENUMERATE_SUB_KEYS Or KEY_NOTIFY) And (Not SYNCHRONIZE))
Const KEY_WRITE = ((STANDARD_RIGHTS_WRITE Or KEY_SET_VALUE Or KEY_CREATE_SUB_KEY) And (Not SYNCHRONIZE))
Const KEY_EXECUTE = (KEY_READ)
Const KEY_ALL_ACCESS = ((STANDARD_RIGHTS_ALL Or KEY_QUERY_VALUE Or KEY_SET_VALUE Or KEY_CREATE_SUB_KEY Or KEY_ENUMERATE_SUB_KEYS Or KEY_NOTIFY Or KEY_CREATE_LINK) And (Not SYNCHRONIZE))
Const REG_SZ = &H1
Const REG_CREATED_NEW_KEY = &H1
Const REG_OPTION_NON_VOLATILE = &H0
'常数: Windows 版本
Const VER_PLATFORM_WIN32s = 0
Const VER_PLATFORM_WIN32_WINDOWS = 1
Const VER_PLATFORM_WIN32_NT = 2
'类型: 用于 GetversionEx 函数
```

```
Private Type OSVERSIONINFO
    dwOSVersionInfoSize As Long
    dwMajorVersion As Long
    dwMinorVersion As Long
    dwBuildNumber As Long
    dwPlatformId As Long
    szCSDVersion As String * 128
End Type
'函数: 用于加载键盘布局
Private Declare Function LoadKeyboardLayout Lib "user32" Alias "LoadKeyboardLayoutA" (ByVal pwszKLID As String, ByVal flags As Long) As Long
'函数: 用于启动键盘布局
Private Declare Function ActivateKeyboardLayout Lib "user32" (ByVal HKL As Long, ByVal flags As Long) As Long
'函数: 用于获取键盘布局的名称
Private Declare Function GetKeyboardLayoutName Lib "user32" Alias "GetKeyboardLayoutNameA" (ByVal pwszKLID As String) As Long
'常数: 用于 ActivateKeyboardLayout 函数
Const HKL_NEXT = 1
'变量: 注册表操作出错信息
Dim strRegError As String
'变量: 将要安装的键盘布局的标识符
Dim strKeyLID As String
'变量: 本汉字输入法的简介
Dim strNote As String
'变量: 启动画面显示信息
Dim strShowup As String
'变量: Windows 的安装目录
Dim strWindowsDir As String
'变量: 当前目录(安装程序所在目录)
Dim strCurDir As String
'变量: Windows 的系统目录(system)
Dim strSystem As String
'变量: 安装成功信息
Dim strSuccess As String
'变量: 用于新旧输入法文件的比较
Dim strNewer As String
Dim strOlder As String
Dim strSame As String
Dim datSource As String
Dim datDest As String
'变量: 文件操作出错信息
Dim strFileError As String
'变量: 正在复制文件信息
Dim strFileCopy As String
'变量: 不安装退出信息
Dim strVain As String
    Private Sub Command1_Click()
    '- - - - - - - - - - - - - - - - - - - - - - - - -
    '按钮: '继续 "( 安装)
    '- - - - - - - - - - - - - - - - - - - - - - - - -
    '变量: 用于文件比较
    Dim lngNewer As Long
    Dim strPreFile As String
    Dim strProFile As String
    '变量: 键盘布局预见加载( preload) 编号
    Dim strKLNumber As String
    On Error GoTo GenError
    '自定义过程/函数: 检查 Windows 的版本
    CheckVersion
    '自定义过程/函数: 检查将被复制的输入法文件的完整性
    If CheckFile = 0 Then
MsgBox strFileError, vbOKOnly + vbExclamation, "找不到文件"
End
    End If
    '自定义过程/函数: 获取 Windows 安装目录
    strWindowsDir = GetWinDir
    '自定义过程/函数: 检查本输入法是否已被安装过;
    '获取一个空闲的 KLID 或已安装的此输入法的 KLID
    strKeyLID = GetLayoutID
    If Right$(strKeyLID, 1) = "*" Then
        '已安装的此输入法的 KLID 后带有一个星号
        '如果是带有星号的 KLID 则进行文件比较
strPreFile = strWindowsDir & "\System\WinWb. MB"
        strProFile = strCurDir & "\WinWb. MB"
        '自定义过程/函数: 文件比较
        lngNewer = CompFile(strPreFile, strProFile)
        If lngNewer = 0 Then
    '将复制的输入法文件与已安装的输入法文件日期相同
If MsgBox(strSame, vbYesNo + vbDefaultButton2, "文件比较") = vbNo Then Exit Sub
ElseIf lngNewer = 1 Then
 '将复制的输入法文件比已安装的输入法文件日期新
If MsgBox(strNewer, vbYesNo + vbDefaultButton1, "文件比较") = vbNo Then Exit Sub
ElseIf lngNewer = 2 Then
 '将复制的输入法文件比已安装的输入法文件日期旧
If MsgBox(strOlder, vbYesNo + vbDefaultButton2, "文件比较") = vbNo Then Exit Sub
End If
 '把 KLID 后的星号去除
strKeyLID = Left$(strKeyLID, Len(strKeyLID) - 1)
 '自定义过程/函数: 关闭已启动的同一输入法
DeactivateKL strKeyLID
'自定义过程/函数: 获取一个键盘布局预加载编号并设置其值
GetLayoutNumber strKeyLID
    Else
'自定义过程/函数: 如果是一个新 KLID 则设置其值
SetKeyValue strKeyLID
```

```
    ´自定义过程/函数:获取一个键盘布局预加载编号并设置其值
        GetLayoutNumber strKeyLID
    End If
´显示文件复制过程中必要的信息
    Label1. Caption = strFileCopy
    With ProgressBar1
        . Visible = True
        . Value = 15
    End With
With Label3
 . Caption = ″还需 25 秒″
 . Visible = True
    End With
    Label2. Visible = False
    DoEvents    ´等待信息显示完毕后才进行文件复制
    ´自定义过程/函数: 复制输入法文件一
CopyWinWb strCurDir & ″\WinWb. ime″, strWindowsDir & _
        ″\System\WinWb. ime″
    ´设置进度条及相应信息
    ProgressBar1. Value = 30
    With Label3
        . Caption = ″还需 20 秒″
        . Refresh
    End With
    DoEvents    ´等待信息显示完毕后再复制输入法文件二
 CopyWinWb strCurDir & ″\WinWb. mb″, strWindowsDir & _
  ″\System\WinWb. mb″
    ProgressBar1. Value = 100
    ´自定义过程/函数: 加载新安装的输入法
    LoadKL strKeyLID
    ´显示安装成功信息
    MsgBox strSuccess, vbOKOnly + vbInformation, ″许辑五笔安装成功″
    ´复位各种显示信息及控件
  Command1. Enabled = False  ´安装成功后使此按钮无效
    Label3. Visible = False
    Label1. Visible = True
    Label1. Caption = strShowup
    Label2. Visible = True
    ProgressBar1. Visible = False
    Exit Sub
    GenError:
    MsgBox ″程序在运行过程中发现错误, 出错的原因是″ & Err. Description & ″. 请检查并排除了错误后才安″ & ″装本输入法. ″, vbOKOnly + vbExclamation, ″安装出错″
    End
    End Sub
Private Sub Command2_Click()
    ´ - - - - - - - - - - - - - - - - - - - - -
    ´ 按钮: ″退出″(安装成功或不安装)
    ´ - - - - - - - - - - - - - - - - - - - - - -
    If Command1. Enabled Then
    MsgBox ″安装程序的本次运行没有把许辑五笔输入法安装到您的电脑中. ″ & Chr$(10) & ″如果您今后需要安装此输入法, 可重新运行此安装程″ & ″序. ″, vbOKOnly + vbInformation, ″退出″
    End If
   End
End Sub
Private Sub Command3_Click()
    ´ - - - - - - - - - - - - - - - - - - - - - -
    ´ 按钮: ″说明″(显示简介信息)
    ´ - - - - - - - - - - - - - - - - - - - - - - -
MsgBox strNote, vbOKOnly + vbInformation, ″安装说明″
    End Sub
Private Sub Form_Load()
    ´ - - - - - - - - - - - - - - - - - - - - - -
    ´ 过程/函数: 窗体加载
    ´ - - - - - - - - - - - - - - - - - - - - - - -
    ´获取当前操作目录
    strCurDir = App. Path
    If Right$(strCurDir, 1) = ″\″ Then
        strCurDir = Left$(strCurDir, Len(strCurDir) - 1)
    End If
    ´以下标签用于创建″平面″按钮
    With Label4
        . Left = 1920
        . Top = 4650
    End With
   With Label5
        . Left = 4380
        . Top = 4650
    End With
    With Label6
        . Left = 6720
        . Top = 4650
    End With
    ProgressBar1. Visible = False
    ´安装程序用到的各种信息
    Frame1. Caption = ″许辑五笔输入法安装程序″
    Me. Caption = ″许辑五笔输入法安装程序″
strShowup = ″  此安装程序将把本人在日常工作中积累和辑录″ & ″的近四万词条的五笔字型输入法安装到您的电脑中. 此输入法″ & ″只供个人自用或教学之用, 不得把它用于商业目的, 否则责任自″ & ″负. 与此有关的更详细说明请按 '说明 '按钮查看. ″
    Label1. Caption = strShowup
strRegError = ″安装程序在读写注册表时发现错误, 安装过程就此中″ & ″止. 请检查注册表是否正常, 或检查一下您是否有权访问注册″ & ″表. 如果需要, 请在纠正错误后再重新安装本输入法. ″
 strNote = Space(20) & ″ «许辑五笔输入法»的安装说明″
```

& vbCrLf & vbCrLf & ″ 本输入法是辑录者按照五笔字型输入″ & ″法的发明人王永民先生所创立的编码原则根据平时的使用体会和″ & ″需求而专门辑录供自己使用的五笔字型输入法. 它的总字词条目接″ & ″近四万, 除了一般的五笔字型输入法具有的字词条目以外, 它还″ & ″有一些地方性词条以及 GBK 字符集编码的字(如粪、堃、″ & ″姮等), 这些字都是一般的五笔字型输入法所无法输入的. 但″ & ″是, 本五笔字型输入法只供个人用户自用或学校教学使用, 不″ & ″得作商业用途, 否则, 后果自负. ″ & vbNewLine

strNote = strNote & ″ 这一输入法主要供中文 Windows 98 作平台″ & ″的电脑安装使用, 并不适用于 Windows 95/97, 原因是它们都″ & ″是较早的版本, 并不支持一些较新的功能, 所以, 无法运行″ & ″安装程序. 本输入法没有在中文 Windows 2000 和 Windows NT 上″ & ″安装过, 估计也无法安装, 原因是据说它们的输入法安装方法″ & ″与个人用户常用的 Windows 9x 不同. ″ & vbNewLine & ″用″ & ″户在使用过此输入法以后有什么意见或建议, 可通过如下 email″ & ″地址向此输入法的辑录者提出, 以便今后改进. ″ & vbNewLine & ″ 辑录者 email 地址: fsxmin@netease.com″

strSuccess = ″许辑五笔输入法已经安装成功, 按 '确定 '键退出以后″ & ″您马上就可以使用它了. ″

strFileError = ″缺乏安装许辑五笔输入法所必需的文件, 无法继续″ & ″进行安装. 请按 '确定 '按钮退出. ″

strVain = ″这次运行程序要安装的输入法并没有安装. 如果需要, 您″ & ″以后可以重新运行此程序进行安装. ″

strFileCopy = vbCrLf & vbCrLf & ″ 正在复制文件, 请稍候. . . . ″

End Sub

Private Function GetWinDir() As String

```
    ' - - - - - - - - - - - - - - - - - - - - - - - - - - - -
    ' 自定义过程/函数: 获取 Windows 的安装目录
    ' - - - - - - - - - - - - - - - - - - - - - - - - - - - -
    Dim intDirLen As Integer
    Dim strWinDir As String
    Dim lngRetVal As Long
    intDirLen = 255
    strWinDir = String$(intDirLen, 0)
lngRetVal = GetWindowsDirectory(strWinDir, intDirLen)
strWinDir = Left$(strWinDir, lngRetVal)
If Right$(strWinDir, 1) = "\" Then
        GetWinDir = Left$(strWinDir, Len(strWinDir) - 1)
    Else
        GetWinDir = strWinDir
    End If
End Function
Private Sub SetKeyValue(ByVal WinWbID As String)
    ' - - - - - - - - - - - - - - - - - - - - - - - - - - - -
    ' 自定义过程/函数: 创建注册表的键并给它设置各值
    ' WinWbID 即 KLID
    ' - - - - - - - - - - - - - - - - - - - - - - - - - - - -
    '变量: 用于 RegCreateKeyEx 函数
    Dim strTmpName As String        '键值的名称
    Dim strTmpFile As String        '键值的数据
    Dim lngDispos As Long
    Dim lngResult As Long           '已打开的键的句柄
    Dim strSubkey As String         '子键(即 KLID)
    Dim typSecurityAttr As SECURITY_ATTRIBUTES
    On Error GoTo SetValueError
    strSubkey = "System\CurrentControlSet\Control\Keyboard" & "Layouts\" & WinWbID
'创建一个子键(即 KLID)
RegCreateKeyEx HKEY_LOCAL_MACHINE, strSubkey, 0&, REG_SZ, REG_OPTION_NON_VOLATILE, KEY_ALL_ACCESS, typSecurityAttr, lngResult, REG_CREATED_NEW_KEY
    '设置此键的各值
    strTmpName = "IME file"
    strTmpFile = "WINWB.IME"
RegSetValueEx lngResult, strTmpName, 0&, REG_SZ, ByVal strTmpFile, Len(strTmpFile)
strTmpName = "layout text"
    strTmpFile = "许辑五笔"
RegSetValueEx lngResult, strTmpName, 0&, REG_SZ, ByVal strTmpFile, Len(strTmpFile)
strTmpName = "layout file"
    strTmpFile = "kbdus.kbd"
RegSetValueEx lngResult, strTmpName, 0&, REG_SZ, ByVal strTmpFile, Len(strTmpFile)
        '完成后关闭此键
    RegCloseKey lngResult
    Exit Sub
    SetValueError:
    MsgBox strRegError, vbOKOnly + vbExclamation, "注册表读写出错"
    End
End Sub
Private Function GetLayoutNumber(ByVal KLID As String) As String
    ' - - - - - - - - - - - - - - - - - - - - - - - - - - - -
    ' 自定义过程/函数: 获取一个键盘布局预
    ' 加载编号并设置其值
    ' - - - - - - - - - - - - - - - - - - - - - - - - - - - -
    Dim lngReturn As Long
    '变量: 用于 RegEnumKeyEx 函数
    Dim lngIndex As Long            '键的索引号, 从 0 起
    Dim strKeyName As String        '键的名称
    Dim lngKeyLen As Long           '键名的长度
Dim lngReserved As Long     '变量: 与 RegQueryValueEx 共用
    Dim strClass As String          '此键的类名
    Dim lngClassLen As Long         '类名的长度
    Dim typLastWrite As FILETIME    'FILETIME 结构(类型)
    '变量: 用于 RegQueryValueEx 函数
    Dim lngHdl As Long              '键的句柄
```

```
    Dim strData As String   ′键的数据值
    Dim lngData As Long     ′数据值的长度
    ′变量: 用于 RegOpenKeyEx 函数
    Dim strSubkey As String
    Dim lngOption As Long
    Dim lngKeyHandle As Long
    On Error GoTo GetNumberError
    ′打开子键以寻找一个空闲的键盘布局预加载编号
    strSubkey = ″Keyboard Layout\Preload″
    lngReturn = RegOpenKeyEx(HKEY_CURRENT_USER, strSubkey, lngOption, KEY_ALL_ACCESS, lngKeyHandle)
If lngReturn <> ERROR_SUCCESS Then GoTo GetNumberError
    ′枚举已预加载的键盘布局的编号, 从索引 0 开始
    lngIndex = 0
    Do
        lngKeyLen = 10
        strKeyName = String$(lngKeyLen, 0)
        lngReserved = 0
        strClass = vbNullString
        lngClassLen = 0
lngReturn = RegEnumKeyEx(lngKeyHandle, lngIndex, strKeyName, lngKeyLen, lngReserved, strClass, lngClassLen, typLastWrite)
        lngIndex = lngIndex + 1
        If lngReturn = ERROR_SUCCESS Then
            ′如果枚举成功则打开此键(编号)以检查同一 KLID
            ′是否已被占用(即此输入法以前已被安装)
RegOpenKeyEx HKEY_CURRENT_USER, strSubkey & ″\″ & Trim(Str(lngIndex)), lngOption, KEY_ALL_ACCESS, lngHdl
            lngData = 255
            strData = String(lngData, 0)
            ′检查其默认值的设置
RegQueryValueEx lngHdl, vbNullString, lngReserved, REG_SZ, ByVal strData, lngData
            If Left$(strData, lngData - 1) = KLID Then
′如果同一输入法的 KLID 已被安装则返回此值并退出本函数
                GetLayoutNumber = Trim(Str$(lngIndex))
                RegCloseKey lngKeyHandle
                Exit Function
            End If
        End If
    Loop While lngReturn = ERROR_SUCCESS
    ′如果此输入法未曾安装过, 在此得到它的预加载编号
    GetLayoutNumber = Trim(Str$(lngIndex))
    ′关闭已打开的各键
    RegCloseKey lngKeyHandle
    RegCloseKey lngHdl
    ′设置刚获得的编号(键)的值
    strKeyName = Trim(Str$(lngIndex))
    SetKLNumber strKeyName, KLID
    Exit Function
    GetNumberError:
    RegCloseKey lngKeyHandle
    MsgBox strRegError, vbOKOnly + vbExclamation, ″注册表读写出错″
    End
End Function
Private Function CheckVersion() As String
    ′ - - - - - - - - - - - - - - - - - - - - - - - - - - - -
    ′ 自定义过程/函数: 检查操作系统的版本以确定是否适于
    ′ 安装此输入法
    ′ - - - - - - - - - - - - - - - - - - - - - - - - - - - -
    Dim typWinVer As OSVERSIONINFO
    ′定义类型(结构)的大小
    typWinVer.dwOSVersionInfoSize = 148
    GetVersionEx typWinVer
If typWinVer.dwPlatformId = VER_PLATFORM_WIN32s Then
CheckVersion = ″本输入法不能安装在 Windows 9x 以下的版本″ & ″中!请按 '确定 '按钮退出安装程序.″
        End
 ElseIf typWinVer.dwPlatformId>=VER_PLATFORM_WIN32_NT Then
    CheckVersion = ″本输入法不能安装在 Windows NT 或 Windows ″ & ″2000 中!请按 '确定 '按钮退出安装程序.″
        End
    End If
End Function
Private Function CheckFile() As Long
    ′ - - - - - - - - - - - - - - - - - - - - - - - - - - - -
    ′ 自定义过程/函数: 检查安装所必需的文件是否存在
    ′ - - - - - - - - - - - - - - - - - - - - - - - - - - - -
    If Dir(strCurDir & ″\WinWb.ime″) = ″″ Or Dir(strCurDir & ″\WinWb.mb″) = ″″ Then
        CheckFile = 0       ′Files do not exist
    Else
        CheckFile = 1       ′Files exist
    End If
    End Function
Private Function GetLayoutID() As String
 ′ - - - - - - - - - - - - - - - - - - - - - - - - - - - -
 ′ 自定义过程/函数: 检查本输入法是否已安装, 如果没有安装
    ′ 则获取一个 KLID
    ′ 返回值: KLID
    ′ - - - - - - - - - - - - - - - - - - - - - - - - - - - -
    Dim i As Integer
    Dim j As Integer
    ′变量: 用于 RegOpenKeyEx 函数
    Dim strSubkey As String
    Dim lngDesired As Long
    Dim lngKeyHandle As Long
    Dim lngOption As Long
```

```
    Dim lngReturn As Long
    lngOption = 0
    lngDesired = 0
    ´变量: 用于 RegQueryValueEx 函数
    Dim strValueName As String
    Dim strData As String
    Dim lngReserved As Long
    Dim lngDataLen As Long
i = 1
    Do
        i = i + 1
        ´ 先预设一个 KLID, 形式如 E0?00804
        ´ ? 从 2 起
        strSubkey = "System\CurrentControlSet\Control\Keyboard Layouts\" & "E0" & Trim$(Str$(i)) & "00804"
            ´打开这一预设的键
        lngReturn = RegOpenKeyEx(HKEY_LOCAL_MACHINE, strSubkey, lngOption, KEY_ALL_ACCESS, lngKeyHandle)
    ´如果此预设键能被成功打开, 检查它是否是已安装过的本输入法
        If lngReturn = ERROR_SUCCESS Then
            strValueName = "layout text"
            lngReserved = 0
            lngDataLen = 255
            strData = String$(lngDataLen, 0)
            For j = 0 To 3
lngReturn = RegQueryValueEx(lngKeyHandle, strValueName, lngReserved, REG_SZ, strData, lngDataLen)
        If lngReturn = ERROR_SUCCESS Then Exit For
            Next j
´如果它是已安装的本输入法则在其后加上一个星号并返回它
If Left$(strData, lngDataLen - 1) = "许辑五笔" Then
GetLayoutID = Mid$(strSubkey, CharFromBack(strSubkey, "\") + 1) & "*"
                RegCloseKey lngKeyHandle
                Exit Function
            End If
        Else
´如果这一预设键不能被打开则把它当作本输入法的 KLID 并返回它
            GetLayoutID = Mid $ (strSubkey, CharFromBack (strSubkey, "\") + 1)
            RegCloseKey lngKeyHandle
            Exit Function
        End If
    Loop
End Function
Private Function CompFile(ByVal InstalledFile As String, ByVal FileToInstall As String) As Long
    ´ - - - - - - - - - - - - - - - - - - - - - - - - - - - -
    ´ 自定义过程/函数: 已安装和将安装的输入法文件比较
    ´ 返回: 关于比较结果的文字表述
    ´ - - - - - - - - - - - - - - - - - - - - - - - - - - - -
´These two values are to be used by MsgBoxes showing the
    ´results of comparison
    ´They are variables used throughout the program
datDest = Format(FileDateTime(InstalledFile), "long date")
datSource = Format(FileDateTime(FileToInstall), "long date")
    ´比较文件的日期
    If Format(datSource, "long date") < Format(datDest, "long date") Then
        CompFile = 1
    ElseIf Format(datSource, "long date") > Format(datDest, "long date") Then
        CompFile = 2
    ElseIf Format(datSource, "long date") = Format(datDest, "long date") Then
        CompFile = 0
    End If
´返回的有关比较结果的文字表述
    strNewer = "  您的电脑中已经安装了许辑五笔输入法, 它的版本日" & "期是:" & vbNewLine & Space(20) & Format(datDest, "long date") & vbCrLf & Space(12) & "您将要安装的输入法的版本日期是:" & vbCrLf & Space(20) & Format(datSource, "long date") & vbCrLf & "即将安装的输入法版本日期比您机中现有的输入法版本日期新, 如果" & vbCrLf & "要安装新的输入法, 将会替换原有的输入法, 是否真的要安装?"
strOlder = "  您的电脑中已经安装了许辑五笔输入法, 它的版本日" & "期是:" & vbCrLf & Space(20) & Format(datDest, "long date") & vbCrLf & Space(12) & "您将要安装的输入法的版本日期是:" & vbCrLf & Space(20) & Format(datSource, "long date") & vbCrLf & "即将安装的输入法版本日期比您机中现有的输入法版本日期旧, 您" & vbCrLf & "似乎没有必要安装新的输入法. 如果要安装新的输" & "入法, 将会替换" & vbCrLf & "原有的输入法, 是否真的要安装?"
    strSame = "  您的电脑中已经安装了许辑五笔输入法, 它的版本日" & "期是:" & vbCrLf & Space(20) & Format(datDest, "long date") & vbCrLf & Space(12) & "您将要安装的输入法的版本日期是:" & vbCrLf & Space(20) & Format(datSource, "long date") & vbCrLf & "即将安装的输入法版本" & "日期与您机中现有的输入法版本日期相同. 如" & vbCrLf & "果要安装新的输入法, 将会替换原有的输入法, 是否真的要安装?"
End Function
Private Sub CopyWinWb(ByVal SourceName As String, ByVal DestName As String)
    ´ - - - - - - - - - - - - - - - - - - - - - - - - - - - -
    ´ 自定义过程/函数: 输入法文件复制
    ´ - - - - - - - - - - - - - - - - - - - - - - - - - - - -
    On Error GoTo CopyError
```

```
    ´If such files already exist set their attribute to Normal
    ´and delete them
    If Dir(DestName) <>"" Then
        SetAttr DestName, vbNormal
        Kill DestName
    End If
        ´Copy the files
    FileCopy SourceName, DestName
    Exit Sub
  CopyError:
MsgBox "安装程序在读写文件时发现错误,错误的原因是" & Err.Description & ".现在请按'确定'按钮退" & "出,把错误纠正后重新进行安装.如果错误是由" & "于电脑中已经装有本输入法而此次开机后又启动" & "过它的话,请重新启动电脑后再安装.", vbOKOnly + vbExclamation, "文件操作出错"
    End
End Sub
Private Sub LoadKL(ByVal KLID As String)
    ´ - - - - - - - - - - - - - - - - - - - - - - - - - - - -
    ´ 自定义过程/函数:安装成功后加载键盘布局
    ´ - - - - - - - - - - - - - - - - - - - - - - - - - - - -
    LoadKeyboardLayout KLID, KLF_ACTIVATE
End Sub
Private Sub DeactivateKL(ByVal LayoutName As String)
    ´ - - - - - - - - - - - - - - - - - - - - - - - - - - - -
    ´ 自定义过程/函数:关闭已启动的本输入法
    ´ - - - - - - - - - - - - - - - - - - - - - - - - - - - -
        ´Variables for GetKeyboardLayoutName
    Dim strKLID As String
        GetKeyboardLayoutName strKLID
        If strKLID = LayoutName Then
        ActivateKeyboardLayout HKL_NEXT, 0
    End If
End Sub
Private Sub SetKLNumber(ByVal KLNumber As String, ByVal KLID As String)
 ´ - - - - - - - - - - - - - - - - - - - - - - - - - - - -
 ´自定义过程/函数:创建键盘布局预加载编号的键并设置其值
 ´ - - - - - - - - - - - - - - - - - - - - - - - - - - - -
    ´Variables used by RegCreateKeyEx
    Dim strSubkey As String
    Dim typSecurityAttr As SECURITY_ATTRIBUTES
    Dim lngKeyHdl As Long
    Dim lngDispos As Long
     On Error GoTo SetNumError
    ´Keyboard layout preload number (the sub key
    ´under the registry main key HKEY_CURRENT_USER
    strSubkey = "Keyboard Layout\Preload\" & KLNumber
    ´创建此子键
    RegCreateKeyEx HKEY_CURRENT_USER, strSubkey, 0&, REG_SZ, REG_OPTION_NON_VOLATILE, KEY_ALL_ACCESS, typSecurityAttr, lngKeyHdl, lngDispos
    ´设置此子键的值(即 KLID)
    RegSetValueEx lngKeyHdl, vbNullString, 0&, REG_SZ, ByVal KLID, Len(KLID) + 1
    Exit Sub
    SetNumError:
    MsgBox strRegError, vbOKOnly + vbExclamation, "注册表读写出错"
    End
    End Sub
Private Sub Form_MouseMove(button As Integer, shift As Integer, x As Single, y As Single)
    ´ - - - - - - - - - - - - - - - - - - - - - - - - - - - -
    ´ 过程/函数:用于"平面"按钮
    ´ - - - - - - - - - - - - - - - - - - - - - - - - - - - -
    If ((x > Command1.Left) And (x < (Command1.Left + Command1.Width))) And ((y > Command1.Top) And (y < (Command1.Top + Command1.Height))) Then
        Command1.Visible = True
    Else
        Command1.Visible = False
    End If
    If ((x > Command2.Left) And (x < (Command2.Left + Command2.Width))) And ((y > Command2.Top) And (y < (Command2.Top + Command2.Height))) Then
        Command2.Visible = True
    Else
        Command2.Visible = False
    End If
    If ((x > Command3.Left) And (x < (Command3.Left + Command3.Width))) And ((y > Command3.Top) And (y < (Command3.Top + Command3.Height))) Then
        Command3.Visible = True
    Else
        Command3.Visible = False
    End If
End Sub
Private Function CharFromBack(ByVal InString As String, ByVal StrToSearch As String) As Long
    ´ - - - - - - - - - - - - - - - - - - - - - - - - - - - -
    ´ Position a character for its first occurence from the
    ´ back of a string, similar to the InStrRev function in
    ´ VB 6.0
    ´ 自定义过程/函数:模拟 VB6 的 InStrRev
    ´ - - - - - - - - - - - - - - - - - - - - - - - - - - - -
    Dim i As Integer
    Dim j As Integer
    For i = 1 To Len(InString)
        If Mid(InString, i, 1) = StrToSearch Then j = i
    Next
    CharFromBack = j·
  End Function
```

收稿日期 2001 年 1 月 17 日

Linux 平台下创建和使用用户函数库

韦建明

摘 要 本文介绍了 Linux 平台下有关库的概念 描述了利用 Linux 下一个高效的编译器 Gcc 创建共享库的方法 并以创建数值计算中特殊函数库为例 着重阐述了在实际应用中创建库应注意的问题和处理方法。

一、Linux 下有关函数库的概念

Linux 下的库实际上就是一些事先经过编译的函数的目标代码文件。使用库的一个好处是，由于库中的函数事先经过编译，所以在程序开发时可以直接加以利用，而不需要重新编译，这样就节省了开发人员的时间和精力，同时也降低了开发的难度。

按照使用时的不同特征，Linux 下的库有三种形式：静态库、共享库和动态库。静态库的代码在编译时就已连接到应用程序中，而共享库只是在程序运行时才载入，在编译时，只是简单地指定需要的库函数。动态库是共享库的另一种变化形式。动态库也是在程序运行时载入，但与共享库不同的地方是，动态库的函数不是在程序运行开始，而是在程序的语句需要使用该函数时才载入。显然，由于共享库和动态库并没有在程序中包含库函数的内容，只是包含了对库函数的引用，因此代码的体积比较小。这一点与 Windows 下动态链接库 dll 和静态链接库 lib 之间的区别有一定的相似之处。

Linux 下的库文件的命名是由前缀 lib 和库的名称以及后缀组成。 库的类型不同，相应的后缀名也不一样。共享库的形式是：libname. so. major. minor，其后缀名由 . so 及版本号组成； name 是库的名称，用来唯一标识这个库，name 可以是一个单字，几个字符，甚至一个字母。如某个数学共享库的库名为 libm. so. 5；静态库的一般形式为 libname. a，其后缀名为 . a， name 是库名，含义与共享库的一致。

二、用 Gcc 创建函数库

Linux 系统已经为用户提供了一些函数库。 比如标准的输入输出库、数学库等等。这样这些库中的函数如 scanf、printf 等在编程时就可以直接使用了。然而在某些情况下 用户也需要创建自己的函数库。比如在 Linux 下编写数值计算程序时，就可以把一些比较特殊的函数算法做成库的形式，在需要的时候加以调用，这无疑会给编程工作带来较大的方便。下面就介绍一下用 Linux 下一个高效编译器 Gcc 来创建用户函数库的方法。

假设用户的源函数为 myfun. c，要生成的库为 libmy. so，下面的语句可以产生一个共享库

```
gcc - shared - fPIC - W myfun. c - o libmy. so
```

有关参数的含义如下

```
-shared ----编译产生共享库
-fPIC -----在编译产生共享库时作一些优化, 可以不加
-W -----输出警告信息, 可以不加
-o -----设定输出文件名(库名)
```

如果要在另一个程序 假设为 myfun1. c 中引用刚才生成的库 libmy. so 可以利用下面的命令

```
gcc myfun1. c - o myfun1 - lmy
```

选项 - l 表示设定要连接的库。

这里还有一个要说明的地方就是库的存放路径问题。一般说来，Linux 下大部分共享库能在 /lib， /usr /lib 目录下找到。有一些库也可能存放在特定的目录中，在 /etc /ld. so. conf 配置文件中有这些目录的列表。Gcc 在缺省情况下，将只在 /etc /ld. so. conf 中指定的目录下寻找库。当然，用户也可以把库放在自己指定的目录下，然后在引用时使用 Gcc 的 - Ldir 选项来指定该目录。

下面是一个简单的创建库和引用库的例子。这里我们把一个简单的 printf 函数做成库的形式，然后在另一个程序中引用。产生库的代码 fun1. c 如下

```
#include <stdio. h>
void prt(char *str)
{
 printf("%s\n", str);
}
```

用下面的语句生成共享库 libmy. so 并把它拷贝到 /lib 目录下

```
gcc -shared -fPIC fun1. c -o libmy. so
```

要引用这个库的程序 fun2. c 为

```
extern void prt(char *);
main()
{
 prt("I love you, China");
 }
```

用下面的语句产生可执行代码

```
gcc fun2. c -o fun2 -lmy
```

执行 . /fun2 后输出 I love you China.

三、创建一个用于数值计算的特殊函数库

以上介绍了用编译器创建用户函数库的一般方法 并举了一个简单的例子。不过 实际应用时情况会复杂一些 引用形式上也会有些变化。 我们还是以创建一个数值计算中的特殊函数库为例 来作具体的描述。

我们要创建的这个库包含了用于数值计算的一些特殊函数 如各类 Bessel 函数、Beta 函数、Gamma 函数等等。 因为函数的数目不止一个 所以把对它们的说明放在一个头文件中是很自然的想法。这样 在其他程序要引用库中某个或某几个函数的时候 只要在程序中把这个头文件包含进去就可以了。

我们把有关特殊函数算法的代码放一个程序中 假设程序名为 mymath. c ，其形式示意如下

```
#include <math. h>
float bessi0(float x)
{
 // 具体代码;
 }
float beta(float x, float y)
 {
 // 具体代码;
 }
float gammln(float x)
 {
 // 具体代码;
 }
 .
 .
 .
```

相应的的头文件(mymath. h)为:

```
float bessi0(float);
float beta(float, float);
float gammln(float );
 .
 .
 .
```

实际创建库的时候还可能碰到一个情况，就是库中的某个函数调用了库里另一个或几个函数。比如，在 Beta 函数的计算中要调用 Gamma 函数，遇到这种情况一定要在程序里对被调用的函数作出说明，如下所示

```
float beta(float z, float w)
{
  extern float gammln(float );
feturn exp(gammln(z) +gammln(w) -gammln(z +w));
}
```

说明的格式是 extern float gammln float 。如果引用了几个函数 也要分别作出说明。

完成了代码的编写工作后，用下列语句创建函数库 libmymath. so

```
gcc - shared - fPIC mymath. c - o libmymath. so - lm
```

注意这里要引用 Linux 下的数学库。

要在另外一个程序中引用 libmymath. so 中的函数，应在程序中包含头文件 mymath. h。一个简单的示例如下

```
#include <stdio. h>
#include <mymath. h>
main()
{
 float a, b, c, d ;
 a = bessi0(5);
 b = bessj(4, 4);
 c = bessk(4, 3. 5);
 d = beta(2. 3, 1. 3);
 printf(″the values of a, b, c, d are %f, %f, %f, %f\n″, a, b, c, d);
}
```

编译如上程序，产生可执行代码后，可得到 a、b、c、d 的值。

当然，这个库里的函数还是很少的，不过可以通过不断地往里面添加新的函数来加以完善。把新函数加入 mymath. c 后，重新编译创建 libmymath. so，再用新的库覆盖掉旧的库就可以了。当然，头文件 mymath. h 中也要增加对新函数的说明。

四、小结

本文介绍了 Linux 下有关库的概念，以及用编译器来创建用户函数库的方法，着重描述了在实际应用中创建库应注意的问题和处理方法。从本文的叙述中不难看出以下两点： 一 Gcc 作为 Linux 下一个免费的编译器，其功能是强大的，值得我们充分加以利用； 二 了解和掌握 Linux 下函数库的创建方法，建立自己的函数库，会节省许多开发程序的时间，降低编程的难度，从而给自己的工作带来很多方便。

参考文献

1. Gcc - Howto http / / gcc. gnu. org
2. 王华 . Linux 编程和网络应用 . 冶金工业出版社 1999

收稿日期：2001 年 1 月 3 日）

所有 KV3000 用户的又一大喜迅

江民公司为了单机上网用户的安全，隆重地为 KV3000 用户增补 JGAH 单机黑客入侵监测软件。

KV3000 系列反病毒软件 + JGAH 单机黑客入侵监测软件软件，等于反病毒加反黑客，即 1 + 1 e = 2。

KV3000 系列反病毒软件可查出 5 万多种病毒和数千种潜伏在计算机中的黑客和后门程序，并可将其清除。但是，反病毒软件不能防备黑客主动入侵攻击行为，特别是 WINDOWS95 / 98 系统，最容易被黑客攻击。现在，有了 JGAH 单机黑客入侵监测软件与 KV3000 软件结合使用效果更佳，可大大减少用户上网和聊天时出现死机、蓝屏、掉线和重起的情况。

所有购买过 KV3000 老用户都可免费使用，这是江民公司回报用户的又一次大增值服务。

老 KV3000 用户可在我网站上下载安装文件到硬盘，执行后就会自动安装 没有 KV3000 的用户将不能安装）。

目前，JGAH 单机黑客入侵监测软件暂时只能安装在 WINDOWS95 / 98 / ME 系统中。

Fortran 与 AutoCAD 间数据通信的几种方法

费璟昊　李俊杰　齐同军

摘　要　本文通过对 dxf 文件格式的论述以及利用 Fortran PowerStation 形成动态链接库，并结合 AutoCAD 的 ActiveX Automation 技术，以 VB 及 VBA 为有效的开发工具，建立 Fortran 与 AutoCAD 之间的联系，为广大工程研究人员和二次开发人员提供了开发模式。

关键词　VB VBA dxf 文件格式 动态链接库 AutoCAD，Automation

一、综述

Fortran 语言和 AutoCAD 是许多行业的工程技术人员所必备的工具，将两者有机地结合起来却并非易事。长期以来，大部分 PC 机用户使用的 Fortran 源代码，都是基于 Fortran77 基础之上的，其数值运算之外的其他功能十分有限。虽然 Microsoft Fortran PowerStation 的几个版本相继出现，但其与 AutoCAD 之间直接进行数据通讯仍旧十分困难。

AutoCAD 为美国 Autodesk 公司研制的计算机辅助绘图软件，其功能强大，在工程制图中有着广泛的应用。目前，对 AutoCAD 进行二次开发的工具大致有 AutoLISP、ADS、ARX 等，随着 AutoCAD2000 新增工具 VBA 的出现，Visual Basic 和 VBA 也成为了 AutoCAD 平台下有力的开发工具。

笔者在隧洞断面及施工方量测量系统及大型储液罐 CAD 辅助设计课题的研究过程中，对 Fortran、VB 及 AutoCAD 之间数据通信方式做了一系列研究，终于探明了将 Fortran 源代码与 AutoCAD 有效地结合起来的几种方式，为广大工程研究人员提供了方向和开发模式。

二、可能的数据通信方式

1. 将 Fortran 的计算结果输出为 dxf 格式，利用 AutoCAD 将其打开即可看到计算的结果图形。Dxf 文件为普通文本模式保存的文件，利用记事本等文本编辑软件即可将其打开而进行编辑。

现举例说明这一方法：用 Fortran 在数据文件中写入如下文本，绘制一条直线和一个圆：

0↙SECTION↙2↙ENTITIES↙0↙3DLINE↙8↙1↙10↙0↙20↙0↙30↙0↙11↙100↙21↙100↙31↙100↙□□0↙ENDSEC↙□□0↙SECTION↙□□2↙ENTITIES↙□□0↙CIRCLE↙□□5↙2B↙□□8↙0↙□10↙186.3318027970312↙□20↙188.41635335252289↙□30↙0.0↙□40↙50.892176719288003↙0↙ENDSEC↙□□0↙EOF

其中，第一个 0 为起始标志，SECTION 是分段标志；第一个 2 是必须输入的，ENTITIES 是实体标志；第二个 0 是实体起始标志，3DLINE 标志是一条直线；10、20、30、11、21、31 分别是起始点的 X、Y、Z 与末点的 X、Y、Z 坐标；紧接其后的是 X、Y、Z 值的大小；对于圆来说，10、20、30、40 则表示圆心的三轴坐标及半径；ENDSEC 为段结束标志，EOF 为结束标志。

注：↙表示回车，□表示空格。

2. 通过 Visual Basic 或 VBA 实现 Fortran 与 AutoCAD 之间的数据通信。本文将着重介绍这种方法，步骤如下：

利用 Fortran 源程序编译成为 dll 文件

☆ 创建步骤

本文利用 Microsoft PowerStation 4.0 开发环境，步骤为：选择 Dynamic Link Library 创建 dll 项目空间）——>调整调用约定输入 Fortran 源代码——> 选择 build ＊.dll 进行编译形成＊.dll 文件。

☆ 调用约定

调用约定将确定命名规范、怎样调用过程以及参数如何传递。

①命名规范 Fortran 中的过程名是外部程序所需知道的内容，本文仅介绍用 stdcall （standard call 的缩写）属性来指定命名约定。如果 Fortran 中过程名与外部调用方某保留关键字冲突，则可利用 Alias 属性使之可以在外部另设名字。

②调用过程 通过 stdcall 指定约定为标准方式。

③传递参数 通过 VALUE （传递数值）和 REFERENCE （传递地址）来设置参数传递方式，从左到右依次传递参数。

④声明 在调用方 （VB 或 VBA）中声明一个 dll 过程，需要在代码窗口顶部的声明部分增加 Declare 语句，声明过程如下：

<Public 或 Private>Declare Sub subname Lib “Libname” <Alias “subdllname”> arg.....

<Public 或 Private>Declare Function Funname Lib “Libname” <Alias “subdllname”> arg..... as Datatype

< > 不同情况下使用与否

subname VB 或 VBA 中的过程名 名称区分大小写）

Libname 动态库名

Subdllname 如动态库中过程与调用方的关键字、常数、变量重名或不符合动态库的命名规范，则需用 Alias 作动态库中的子过程名

arg. . . . . 传递的参数列表

as Datatype VB 或 VBA 中的数据类型声明

相应的，应在 Fortran 中进行如下设置：

利用! MS$ATTRIBUTES 关键字

! MS$ATTRIBUTES attributes variants

（MS$ATTRIBUTES reference variants）

（） 如果变量以传递地址的方式，则写下括号内的语句

attributes 属性表

variants 参数列表

具体参数参见本文提供实例。

☆数据类型介绍：

在 Fortran 中的数据类型大致有 integer、real、double precision 及 characteric 等，在 VB 或 VBA 中则有 integer、long、single、double 及 string 等与之对应，其对应关系如下表：

需要强调的是：仅知道以上数据类型的对应规则还不够，调用往往不能取得成功 某些调用失败对于调用方来说甚至是致命的）。通常在 Fortran 中定义的 integer 2 ，integer 4 ，real 4 ，real 8 ，double pricision 等单个数据类型的传输应以传递参数值为主，如不需要改变参数值就应避免使用默认的传递参数地址方式，这在传递大量参数时尤为重要。具有 C 语言知识的人对数组的存储方式并不陌生，VB 与 Fortran 只是将数组的指针属性进行了封装，实际上，对数组的传递仍是以指针 地址）形式进行的。因此，将数组作为参数传递时，应以默认 即 byref）方式传递，将第一个元素作为参数即可。字符串与数组类似，也是通过传递地址实现的，但关键字应使用 ByVal，而且一定要声明字符串的长度 长度必须足够大，如在调用方更改字符串时超出声明范围，则可能在内存中引起错误）。通过地址传递的数据在动态库中值的变化将能够直接反映到调用方中。

3. 用 AutoCAD 中的 VBA 实现 dll 调用

实例 1：

目的：测试整型、实型的传输，在 AutoCAD 中用对话框将结果显示出来。

内容：testdll. f90

```
REAL(4) FUNCTION TESTDLL(INTS, REALS)
! MS $ATTRIBUTES STDCALL, DLLEXPORT, – ALIAS:´
TESTDLL´: : TESTDLL
! MS$ATTRIBUTES VALUE: : INTS, REALS
      INTEGER(2) INTS
      REAL(4) REALS
      REAL(4) RESULTS
      RESULTS =REALS * REAL(INTS)
      TESTDLL =RESULTS
END FUNCTION
```

在 AutoCAD 中插入一个用户窗体 UserForm1，插入一个按钮 CommandButton1，在声明区写下：

```
Private Declare Function TESTDLL Lib ″D: \fjh\ testdll. dll″
(ByVal ints As Integer, ByVal reals As Single) As Single
```

如果将动态库置于 C hWindows hSystem 下，则可省略动态库存放路径 D hfjh h；由于 testdll. f90 中函数名称为大写，故 VB 中过程名也必须大写。

CommandButton1 的 Click 事件加入以下代码：

```
Private Sub CommandButton3_Click()
Dim ints As Integer
Dim reals As Single
Dim results As Single
ints = 100
reals = 200#
results = TESTDLL(ints, reals)
MsgBox str(results)
End Sub
```

实例 2：

目的： 数组与字符串的传输，并在 AutoCAD 中将字符串显示出来。

内容：teststr. f90

```
SUBROUTINE TESTSTR(POIN, MSGSTR)
! MS $ATTRIBUTES STDCALL, DLLEXPORT, – ALIAS:´
TESTSTR´: : TESTSTR
! MS$ATTRIBUTES REFERENCE: : MSGSTR
REAL(8) POIN(1: 3)
CHARACTER(40 ) MSGSTR
POIN(1) =0. 0
POIN(2) =8000. 0
POIN(3) =0. 0
 MSGSTR =″你们好! 这是第一个标出字体程序! ″
END SUBROUTINE
```

在 AutoCAD 中添加如下声明:

```
Private Declare Sub TESTSTR Lib ″D: \fjh\testdll. dll″ (ByRef
poin As Double, ByVal MsgStr As String)
```

在 CommandButton1 的 Click 事件加入以下代码

```
Private Sub CommandButton1_Click()
Dim tObj As AcadText
Dim MsgStr As String * 40
Dim bObj As AcadBlock
Dim poin(1 To 3) As Double
Dim ObjFontStyle As AcadTextStyle
Call TESTSTR(poin(1), MsgStr)
Set ObjFontStyle = ThisDrawing. TextStyles. Add – (″Style-
ofME″)
ObjFontStyle. fontFile = ″ C: \ Windows\ Fonts – \ sim-
fang. ttf″
Set tObj = ThisDrawing. ModelSpace. AddText – (MsgStr,
poin, 800)
tObj. StyleName = ″StyleofME″
tObj. Alignment = acAlignmentMiddle ´中心对齐方式
tObj. TextAlignmentPoint = poin ´给定对齐点
ThisDrawing. SendCommand ″_zoom″ & Chr(13)
ThisDrawing. SendCommand ″all″ & Chr(13)
```

下转第 45 页

Java 程序与 Servlet、ASP、PHP、CGI 等的通信

徐迎晓

Servlet、ASP、PHP、CGI 等程序实际上是运行在 Web 服务器上的程序，其程序中一般使用一些参数，用户端通过浏览器或程序向其传递参数的值，Servlet、ASP、PHP、CGI 等程序读取这些参数值后作相应的处理，并向用户端反馈信息。

我们常见的搜索引擎就是将用户填写的关键字通过参数传递给服务器上的 Servlet、ASP、PHP、CGI 等程序的，这些程序搜索相应的数据库，从而将搜索结果反馈给用户。

Java 程序通过 URLConnection 对象可以和 Servlet、ASP、PHP、CGI 等通信，通信的方式有 get 和 post 等。

一、 Get 方式

对于 get 方式，用户向服务器上的程序传递少量信息，而得到大量的反馈信息。如一般的搜索引擎使用的就是这种方式。

浏览器或程序在传输时会将 Servlet、ASP、PHP、CGI 等程序的 URL 和执行程序的参数组成一个字符串将其传送出去。编程的基本步骤是

1. 生成传输字符串

其格式为 CGI 程序 变量 1 = 值 1& 变量 2 = 值 2& 变量 3 = 值 3 . . .

如在 http / /202. 120. 127. 219 /cgi - bin /有一个 CGI 程序 query，为了给其中的参数 xx 赋值 “my test”，可以使用如下字符串：

String s = http / /202. 120. 127. 219 /cgi - bin /query xx = + URLEncoder. encode my test

变量的值如果含有空格或中文等，一般需要通过 URLEncoder. encode () 方法进行编码。有多个参数则使用 “& ”隔开。如：

```
s =″http: //202. 120. 127. 219/cgi –bin/query?xx =″ + encode(″my test″) + ″& yy =″ + URLEncoder. encode(″my test2″);
```

2. 创建 URLConnection 对象

利用前面的传输字符串使用 new URL 传输字符串 即可创建 URL 对象，然后执行其 openConnection 方法，即可生成 URLConnection 对象。并执行服务器上对应的程序。

如对前面的字符串 s 可以使用如下语句：

URLConnection con = new URL s . openConnection

3. 打开输入流，读取输入流

执行 URLConnection 对象的 getInputStream 方法可以获得输入流，进而可以读取服务器程序的反馈。

如下面的程序在一台连入 Internet 的机器上执行的结果为：

```
<H1>Query Results </H1>You submitted the following name/value pairs: <p>
<ul>
<li> <code>xx = my test </code>
<li> <code>yy = my test2 </code>
</ul>
```

```
import java. io. *;
import java. net. *;
public class e1{
  public static void main(String a[]) throws Exception{
     String s;
      s =″ http: //202. 120. 127. 219/cgi –bin/query? xx =″ +URLEncoder. encode(″my test″) + ″& yy =″ + URLEncoder. encode(″my test2″);
    URLConnection con =(new URL(s)). openConnection();
     BufferedReader in = new BufferedReader(
       new InputStreamReader(con. getInputStream()));
     while ( (s =in. readLine() )! =null){
        System. out. println(s);
     }
 } }
```

该例子使用的是网上已有的 CGI 程序，使用 Java，我们可以自己编写 Java Servlet 或 JSP 程序，起到 CGI 程序的功能。

Java Servlet 程序基本格式为：

```
import javax. servlet. *;
import javax. servlet. http. *;
import java. io. *;t
  public class mys extends HttpServle
  {
  public void doGet(HttpServletRequest rq, HttpServletResponse rp)
throws ServletException, IOException
 {    //设置反馈格式为 HTML 格式
       rp. setContentType(″text/html″);
// 获取输出流, 通过其 println( )方法可以向用户端反馈信息
       ServletOutputStream out =rp. getOutputStream();
 // 获取用户端传递来的 xx 参数值(根据需要可以获取多个各种参数)
       String value1 = rq. getParameter(″xx″);
       String value2 = rq. getParameter(″yy″);
      // 根据需要利用参数值可以做各种事情
      // . . . .
      // 反馈信息
       out. println(″Your input is ″ +value1);
       out. println(″Your input is ″ +value2);
  }
}
```

下面给出了一个类 myp，其中定义了一个简化但通用的方

法 getinfo ()，它将一组参数及其值放在两个数组中，以 get 方式发向 URL 为 servleturl 的 Servlet 中。该方法返回一个 BufferedReader 流对象，可以从流中读出 Servlet 的反馈信息。

```
import java. io. *;
import java. net. *;
class myp{
public static BufferedReader getinfo(String servleturl,
        String query[], String value[])  throws Exception
    {  URLConnection con;
       String s;
       s =servleturl +"?";
       for(int i =0; i <query. length; i + +){
         if(i>0) s + ="& ";
      s + =query[i] +" =" +URLEncoder. encode(value[i]);
       }
       con =(new URL(s)). openConnection();
       con. setDoInput(true);
       con. setDoOutput(false);
       con. connect();
       return(new BufferedReader(
       new InputStreamReader(con. getInputStream())));
    }
}
```

下面的程序是使用上面类 myp，通过搜索引擎 Yahoo 的 CGI 程序搜索信息。中文 Yahoo 搜索引擎的 URL 为：http //cn. search. yahoo. com/search/cn，其中有两个参数：p 代表要搜索的字符串，g 的值一般可以用 "y"。

```
import javax. servlet. *;
import javax. servlet. http. *;
import java. io. *;
class e2{
  public static void main(String a[]) throws Exception{
       String s;
       String q[] ={"p", "y"};
       String v[] ={"徐迎晓", "g"};
BufferedReader  in =myp. getinfo("http: //cn. search. yahoo
. com/search/cn", q, v);
       while ( (s =in. readLine() )! =null){
         System. out. println(s);
     }
  }
}
```

编译后运行 java e2 可以得到输出结果，也可运行 java e2> re. html，将输入结果重定向到 re. html 文件中，用浏览器打开看其输出结果。

二、Post 方式

对于 post 方式，用户向服务器上的程序传递大量信息，而得到少量的反馈信息。如通过 Web 方式填写登记表一般采用的就是这种方式。

浏览器或程序在传输时会传输 Servlet、ASP、PHP、CGI 等程序的 URL，然后将程序的参数组成一个字符串传送出去。编程的基本步骤是

1. 生成 URL 字符串

如 String s =http //202. 120. 127. 219/cgi - bin/post - query，其中不含参数及参数值。

2. 创建 URLConnection 对象，并允许输出

利用前面的查找字符串使用 new URL 查找字符串 即可创建 URL 对象，执行其 openConnection 方法，即可生成 URL-Connection 对象。执行其 setDoOutput () 方法可允许向其输出信息。

如对前面的字符串 s 可以使用如下语句：

```
URLConnection con =  new URL  s  . openConnection
    con. setDoOutput  true
```

3. 打开输出流，向服务器端程序输出参数

执行 URLConnection 对象的 getOutputStream 方法可以获得输出流，进而可以使用输出流的 println () 方法向服务器程序的输出参数。如：

```
        PrintStream  out =new  PrintStream
(con. getOutputStream());
     String  ss =" xx" +" =" +URLEncoder. encode( " my
test1");
    out. println(ss);
```

有多个参数时使用多个语句，后面的参数开头必须加"& "符号。

4. 打开输入流，读取输入流

执行 URLConnection 对象的 getInputStream 方法可以获得输入流，进而可以读取服务器程序的反馈。

下例为完整的程序：

```
import java. io. *;
import java. net. *;
public class e3{
  public static void main(String a[]) throws Exception{
String   s =" http: //202. 120. 127. 219/cgi -bin/post -
query";
URLConnection con =(new URL(s)). openConnection();
    con. setDoOutput(true);
PrintStream out =new PrintStream(con. getOutputStream());
String ss ="xx" +" =" +URLEncoder. encode("my test1");
    out. println(ss);
    ss ="& yy" +" =" +URLEncoder. encode("my test2");
    out. println(ss);
     BufferedReader  in =new  BufferedReader(new  Input-
StreamReader(con. getInputStream()));
    while ( (s =in. readLine() )! =null){
       System. out. println(s);
    }
  }
}
```

如果自己使用 Java Servlet 编写服务器端的程序，其方法和前面的 get 方式相同，只要将其中的如下代码：

```
public void doGet(HttpServletRequest rq, HttpServletRe-
sponse rp)
```

throws ServletException, IOException

改为：

public void doPost(HttpServletRequest rq, HttpServletResponse rp)
throws ServletException, IOException

即可。

同样可以用在前面的 mys 类中加入如下通用的方法通过 post 方式发送数据：

```
public BufferedReader postinfo(String servleturl,
String query[], String value[]) throws Exception
{   String s;
URLConnection con = (new URL(servleturl))
. openConnection();
        con. setDoOutput(true);
PrintStream out = new PrintStream(con. getOutputStream());
for(int i = 0; i < query. length; i + +){
          s = i>0?"&":"";
s + = query[i] + "=" + URLEncoder. encode(value[i]);
          out. println(s);
        }
        out. close();
        return(new BufferedReader(
new InputStreamReader(con. getInputStream())));
    }
```

收稿日期 2001 年 1 月 12 日

上接第 42 页

```
End Sub
```

这样，就出现了 AutoCAD 界面上就出现了 Fortran 中输入的文字。另外，需要特别注意，虽然字符串是以传递地址的方式传输的 在 Fortran 中声明是用 REFERENCE），但其在 VB 中声明却用 Byval 关键字，这是与其它变量不同之处，这与 Fortran PowerStation 传输字符串的格式有关，传递字符串时，除字符串的内容之外，还以传递数值的方式传输了字符串的长度变量。测试地址传输与数值传输的区别并不难，只需在调用动态库之后查看原来的参数有无变化即可。

4. 利用 VB 实现 AutoCAD 与 Fortran 之间的通信

AutoCAD 中的 VBA 模块可以嵌入至图形中，从而成为图形的一部分，但只能在包含该模块的图形中才能执行它，而且其源代码无法进行外部封装，以致无法形成自己的安装包以适应广泛移植的需要。VB 是通过 AutoCAD ActiveX Automation 自动机）接口来建立与 AutoCAD 对象之间的联系。建立在 COM 对象模型上的 ActiveX 是一个标准通信协议，它使得 ActiveX 应用程序比 ADS 或 AutoLisp 应用程序运行速度更快。

本文仅利用 AutoCAD 提供的与菜单相关对象中的 MenuBar、MenuGroup 6）在 AutoCAD 平台上创建自己的菜单，实现动态库的调用。

实例 3：

目的：调用 AutoCAD 中的画线命令和利用 VB 直接在 AutoCAD 界面上绘制一条直线，以显示区别。

在 VB 中加入一个窗体，取默认名为 Form1，在声明部分加入以下代码：

```
Private Declare Sub GetLine Lib "D: \fjh\GetLine. dll" (ByRef
BeginXY As Double, ByRef EndXY – As Double)
Private AcadApp As Object, AcadDoc As Object, – MoSpace
As Object
```

在 Form_Load 事件中加入：

```
On Error Resume Next
Set AcadApp = GetObject("AutoCAD. Application")
If Err Then
  Err. Clear
Set AcadApp = CreateObject("AutoCAD. – Application")
End If
Set AcadDoc = AcadApp. ActiveDocument
Set MoSpace = AcadApp. Modelspace
```

在窗体上加入一个按钮 Command1 加入以下代码

```
Private Sub Command1_Click()
Dim MenuGroup As Object, NewMenu As Object
Dim MenuItem As Object, LineObj As Object
Dim CommandMacro As String
Dim BeginXY(1 To 3) As Double, EndXY(1 To 3) – As
Double
On Error Resume Next
Set MenuGroup = AcadApp. MenuGroups. Item(0)
Set NewMenu = MenuGroup. Menus. Add("自己的菜单")
NewMenu. InsertInMenuBar (AcadApp. MenuBar. Count + 1)
CommandMacro = Chr(3) & Chr(3) & Chr(95) & "line"
& Chr(32)
Set MenuItem = NewMenu. AddMenuItem(1, "画一条线",
CommandMacro)
GetLine BeginXY(1), EndXY(1)
Set LineObj = AcadDoc. Modelspace. AddLine – (BeginXY,
EndXY)
AcadApp. Visible = True
End Sub
```

相应的动态库源文件代码为：GetLine. f90

```
SUBROUTINE GetLine (BEGINXY, ENDXY)
! MS$ATTRIBUTES STDCALL, DLLEXPORT, – ALIAS: ' Get-
Line ': : GetLine
DOUBLE PRECISION BEGINXY(1: 3), ENDXY(1: 3)
BEGINXY(1) = 0; BEGINXY(2) = 0; BEGINXY(3) = 0
ENDXY(1) = 100; ENDXY(2) = 100; ENDXY(3) = 100
END SUBROUTINE
```

三、 结束语

本文通过对 AutoCAD 的菜单及图形操作为例，阐述了利用 VB 或 VBA 实现 AutoCAD 与 Fortran 之间的数据通信，从而实现了对 AutoCAD 的二次开发，为读者提供了可利用的开发模式。读者可根据实际需要进行深层次的设计和研究。

收稿日期：2001 年 1 月 11 日）

如何用 VC 实现 COBOL 系统的数据到 SQL Server 的迁移

刘东升

摘 要 本文介绍了如何用 VC 解决从 COBOL 系统下的数据向关系型数据库迁移的方法及具体实现。

关键词 VC,SQL Server,COBOL 组织

一、引言

COBOL 作为一代流行的编程语言，曾广泛地应用于政府、银行、运输等领域的信息系统中，在 80 年代，它是商业应用程序的首选语言，所以也产生了许多基于 COBOL 的数据文件，但是 COBOL 作为一种非可视化的第三代编程语言，有其固有的缺点，随着 Windows 图形用户界面在桌面计算机上统治地位的确定，如何实现将基于 COBOL 的数据安全地迁移到新的系统中去是许多以前采用 COBOL 系统的组织所面临的问题，我们就在一个项目开发中遇到了这样的问题，通过研究我们找到了一种解决方案，用 VC 得到了圆满的解决。

二、问题分析

余杭广电的收费系统是 80 年代用 COBOL 开发的，是适应 DOS 时代的产物，已越来越不能与现代的基于 Windows 的可视化界面相适应，原先的系统已到了生命的终点。他们便想将原先的基于 DOS 的软件迁移到 Windows 上来，新系统采用 PB 开发前台可视化界面，后台采用 Microsoft SQL Server 数据库，由于老系统已经积累了十几年的历史数据，不可能采用重新输入的办法，这时便需要将原先的数据安全地迁移到新的系统中来。

通过分析 COBOL 系统下的数据文件 ict 文件，我们发现了一些数据记录的规律，我们用 UltraEdit - 32 打开某一个数据文件后，发现每条记录都是定长的且都以 “hn ”结束，同时又发现了各个数据项的具体含义，这样我们的问题便成了如何将二进制文件的内容转化为平面文件的问题。例如在一个记录流水交费的数据文件中 dst00. ict 中，我们发现每行的长度为 347，且每个数据项的意义及长度如下：编号 3、区域 4、单位码 4、姓名 8、电话 30、帐号 10、单位名称 30、住址 30、幢号 3、室号 3、入网日期 6、建设费 4、收视费 4、详细地址 20、屏幕数 2、终端数 2、集体数 2、开户性质 2、欠费 8，另外还有在新系统工程中不需要的 324 位。知道了具体的含义和长度后，我们只要读出具体的数据，然后再转换就可以，同时对于以二进制存储的数据在写到平面文件时需要转换成普通的字符型存储。

三、功能设计

通过以上的分析，我们要实现的功能便是将 COBOL 系统下的 ict 文件转为平面文件，然后再将数据导到 SQL Server 数据库中，这中间的技术关键是编写将 ict 文件的内容按一定格式转成 txt 文件的程序。基于以上问题的分析，我们用 VC 编程实现上述功能，程序显示界面窗口客户区分为三部分，左边显示文件的路径、位置，右上部分显示原文件内容，右下部分显示经过处理的平面文件的内容，同时对平面文件作了存储处理，为简化程序设计，我们没做打印处理。

四、编程实现

为简化内容，我们只取了关键一部分程序内容，详细内容见文档。采用 MFC 的 SDI 单文档界面）。由于在一屏内不可能显示整个文件内容，所以选择视类的基类为 ScrollView。ict 文件的读出与处理在文档类中完成，文件的显示与滚动由视类来完成。

1. 使用 MFC AppWizard 向导产生一应用框架。在 Project name 编辑框中输入 “data ” 在 Step 1 中选 Single Document 在 Step 4 中取消 “Printing and Print Preview ”，在 Step 6 中选 CsrollView 作为 BaseClass 其余默认，点击 finish 完成应用框架的定制。

2. 在文档类 CdataDoc 中增加文件的读出和处理工作

2. 1 定义文档的成员变量，作好初始化及清理工作

1 打开 dataDoc. h 文件，增加公共变量。

```
public:
    CFile * m_pHexFile;
   LONG m_lFileLength;
```

2 打开 dataDoc. cpp 文件，在构造函数中增加下列初始化代码：

```
m_pHexFile = NULL;
 m_lFileLength = 0L;
```

在析构函数中增加如下清理代码：

```
if (m_pHexFile ! = NULL)
{
     m_pHexFile ->Close();
     delete m_pHexFile;
```

```
        m_pHexFile = NULL;
      }
```

3 重载 OnOpenDocumnet ，在其中打开文档。

```
    if (m_pHexFile ! = NULL)
    {
        m_pHexFile ->Close();
        delete m_pHexFile;
    }
    char * lpszPathName = "dst00. ict";
    m_pHexFile = new CFile(lpszPathName,
  CFile: : modeRead | CFile: : typeBinary);
    if (! m_pHexFile)
    {
        AfxMessageBox("该文件打开错");
    }
    m_lFileLength = m_pHexFile - >GetLength
```

4 在菜单中增加按扭，在其函数中增加转换功能，并将原文件写入到平面文件中去。

```
FILE * stream;
stream = fopen( "filename", "w" );   //filename 为在对话框中输入的文件名字
int fi = 0;
int nRet;    //假定每行的长度为 347, 以 nRet 为计量每行中字符的走动
    do
    {
      int nRet;
        //读编号 bh
      byte bh[3];
      int  bh2 = 0;
      nRet = m_pHexFile ->Read(bh, 3);
    bh2 = (int ) bh[2] + (int ) bh[1] * 16 * 16 + (int ) bh[0] * 256 * 256;  //原来的为 float 型存储, 需转换
   //写到平面文本文件
    //编号
    int i;
    int tmp, tmp2;
  tmp = bh2;
    //char * p;
    i = 0;
    do {
        tmp2 = tmp%10;
        tmp2 = tmp2 + 48;
        bh[i] = tmp2;
        i + +;
        tmp = tmp/10;
    }while(tmp! = 0);
    int k = i;
    for(i = k - 1; i> = 0; i - - )
     fprintf(stream, "%c", bh[i]);
        if(k < 5)
        for(i = k; i < 5; i + + )
      fprintf(stream, "%c", ' ');
}
}
    fclose( stream );
```

2.2 在视中增加显示文件内容以及处理滚动工作

1 在 dataView. h 中增加变量定义及初始化。

```
    CDATADoc * GetDocument();
```

2 在 dataView. cpp 的 OnDraw 函数中增加显示功能。

```
  CDATADoc * pDoc = GetDocument();
ASSERT_VALID(pDoc);
Cstring sline; //用于显示文本行
Csize ScrolledSize; //窗口客户区的范围
int iStartLine;    //当前屏第一行显示的行的索引号
int nHeight;  //输出时文本行的高度
Crect ScrollClient;
    nHeight = pDC - >DrawText(sLine, - 1,  &  ScrollRect, DT_TOP|DT_NOPRIFIX|DT_SINGLELINE);
ScrollRect. top + = nHeight;
```

在该函数中选择 CDC DrawText 函数也非常关键，比使用 CDC TextOut 函数能起到事半功倍的作用，有更大的优点。

五、结束语

对该工程进行编译、连接后，形成编译文件 data. exe ，经过测试运行，生成 txt 文件，然后利用 SQL Server 自带的工具顺利地将数据迁移到了 SQL Server 数据库中，达到了预定目的。

通过上面的介绍，用 VC 实现该程序并不复杂，关键在于用 UltraEdit - 32 分析原理复杂的 ict 文件，找出其中的规律，进而编程实现转换。最后，用这种方法也可以对二进制文件进行查看、转换，以及其它文件的转换。

参考文献

1. 希望图书创作室译． VC + + 6. 0 技术内幕．北京：北京希望电子出版社

2. 黄惠菊、张捷等译．轻松掌握用 VC + + 6 对数据库编程．北京：电子工业出版社

3. 袁鹏飞． SQL Server7. 0 数据库系统管理与应用开发．北京： 人民邮电出版社

（收稿日期：2001 年 1 月 15 日）

网络扬名 :卡西欧电子琴签约金仕达多媒体

卡西欧是世界著名的电子产品厂商，电子琴产品在中国可谓家喻户晓。近日，卡西欧电子与金仕达多媒体公司签订网站开发合同，为旗下重点产品——卡西欧电子琴制作营销推广网站。

随着网络的发展，上网人口的急剧增加，制作专门的产品推介网站已经成为企业产品推广的重要手段。从最初的 IT 产品开始，现在有越来越多的大众消费品也走入网络营销的行列。

作为中国最早涉足网站开发领域的软件公司之一，金仕达多媒体不仅已经具备了开发大型电子商务网站的能力，在网络营销方面已经获 得了许多成功经验。此前，曾经开发的房产营销网站——古北小区、教育商务营销网站——恩哺动力幸福网站等都取得了非常好的效果。

利用中断异常技术实现 Windows 9x 下硬盘序列号的读取

郑沛峰 谢瑞和

摘 要 本文提出了一种通过中断异常技术使 32 位的应用程序从 Ring3 级转入 Ring0 级，执行被 Windows 系统屏蔽的特权指令的方法，直接访问硬盘控制器，获取硬盘的序列号。

关键词 Windows 中断异常，Ring0，特权指令，硬盘序列号

一、 引言

硬盘序列号是硬盘厂商生产时给每一块硬盘分配的全世界唯一的标志号，存在于硬盘的控制芯片内，几乎不能修改，更不会随硬盘的各种使用方式如分区、格式化等而改变。它与硬盘分区后随机生成的序列号有着根本的不同。利用硬盘序列号的这种唯一性和只读性，可以应用于软件的防拷贝以及加密方面。这种利用物理存储介质进行软件加密的关键就是如何去获取硬盘的序列号，根据这个唯一的标志数据产生的密钥或者使用序列号，可以有效地防止软件的任意拷贝，实现用户的唯一授权使用。

在以前 DOS 操作系统中，可以直接读写硬盘控制器，很容易地获取硬盘序列号。可是，在 Windows 系统下，对硬盘控制器的读写操作属于特权指令，被系统屏蔽，应用程序是不能访问的。这种系统级的禁令使得很多想采用这种防拷贝技术的软件开发者颇为头疼，如何突破这种系统屏蔽是获取硬盘序列号的关键。下面介绍一种通过中断异常技术直接在 Visual C + + 中编程的实现方法。

二、 中断异常

在 Windows9x 系统中有两个特权级——Ring 0 和 Ring 3。运行在 Ring 0 的软件能够对系统进行任何操作，而只在 Ring 3 运行的应用程序会受到一些限制，许多对硬件端口的读写访问操作被禁止。因此，解决问题的关键就是如何将应用程序从 Ring3 级转入 Ring0 级。

通常的作法是编写一个专门的虚拟设备驱动程序（VxD）。VxD 不但可以执行所有特权指令，而且可以调用 VMM 虚拟机管理器）和其他 VxD 提供的系统级服务。但是，它体系复杂，涉及到操作系统内核，对操作系统依赖性强，开发难度大。为此，我们专门研究了 Windows 下的中断异常技术，希望借此实现在单一的可视化开发平台下直接执行特权级指令。

8086 /8088 把中断分为内部中断和外部中断两大类。为了支持多任务和虚拟存储器等功能，80386 把外部中断称为“中断”，把内部中断称为“异常”。这里，我们主要讨论内部中断即异常。异常是 80386 在执行指令期间检测到不正常的或非法的条件所引起的，如在程序中借助于软中断指令“INT n 和“INTO 就可以产生异常事件。在响应中断或者处理异常时，80386 根据中断向量号转向对应的处理程序进行处理。在保护模式下，80386 不使用实模式下的中断向量表，而是使用中断描述符表 IDT，它把中断向量号作为中断描述符表 IDT 中描述符的索引，而不再是中断向量表中的中断向量的索引。这样，在保护模式下，我们可以在应用程序中使用非特权指令 sidt 直接获取中断描述符表 IDT 的基地址，然后再修改 IDT，增加一个中断门安置自己特定的中断异常服务，一旦在 Ring3 程序中产生此中断，虚拟机管理器（VMM）就会调用此服务程序，而此时的中断或异常处理程序恰好是工作在 Ring0 级，它们可以执行任何指令，包括直接访问硬盘控制器。目前流行的 CIH 病毒具有非常大的杀伤力的原因也就是基于此。下面是具体的产生中断异常进入 Ring0 级的实现代码：

```
DWORD OldInterruptAddress;
DWORDLONG IDTR;
void _stdcall ReadIdeSerialNumber()
{
    _asm
    {
      push eax
     //获取修改的中断的中断描述符(中断门)地址
     sidt IDTR
          mov eax, dword ptr [IDTR +02h]
     add eax, 3 * 08h +04h//使用 3 号中断
     cli
     //保存原先的中断入口地址
     push ecx
     mov ecx, dword ptr [eax]
     mov cx, word ptr [eax -04h]
     mov dword ptr OldInterruptAddress, ecx
     pop ecx
    //设置修改的中断入口地址为新的中断处理程序入口地址
     push ebx
     lea ebx, InterruptProcess
          mov word ptr [eax -04h], bx
     shr ebx, 10h
```

```
        mov word ptr [eax+02h], bx
        pop ebx
        //执行中断, 转到 Ring 0(类似 CIH 病毒原理)
                int 3h
        //恢复原先的中断入口地址
        push ecx
        mov ecx, dword ptr OldInterruptAddress
        mov word ptr [eax-04h], cx
        shr ecx, 10h
        mov word ptr [eax+02h], cx
        pop ecx
        sti
        pop eax
       }
}
```

三、 硬盘序列号的读取

我们通过在 Ring0 级的中断服务程序中使用 CPU 的 I/O 指令直接访问硬盘控制器的方法来读取硬盘的序列号。读取的方法如下面的程序所示：

```
WORD serial[256]; //保存硬盘控制器的信息
static unsigned int WaitIde()
{
      BYTE xx;
Waiting:
      _asm{
            mov dx, 0x1f7
            in al, dx
            cmp al, 0x80
            jb Endwaiting
            jmp Waiting:
          }
Endwaiting:
           _asm{
                 mov xx, al
               }
      return xx;
  }
void _declspec( naked )InterruptProcess(void)// 中断服务程序
{
      int  xx;
int  i;
WORD temp;
//保存寄存器值
    _asm
     {
      push eax
      push ebx
      push ecx
      push edx
      push esi
     }
    WaitHardDiskIde(); //等待硬盘空闲状态
    //命令端口 1f6, 选择驱动器 0
     _asm{
        mov dx, 0x1f6
        mov al, 0xa0
        out dx, al
      }
   xx = WaitHardDiskIde (); //若直接在 Ring3 级执行等待命令, 会进入死循环
   if ((xx& 0x50)!=0x50)
   {
         _asm{
       pop esi
       pop edx
       pop ecx
       pop ebx
       pop eax
         iretd
         }
   }
         _asm{
         mov dx, 0x1f6 //命令端口 1f6, 选择驱动器 0
         mov al, 0xa0
         out dx, al
         inc dx
         mov al, 0xec
         out dx, al  //发送读驱动器参数命令
   }
   xx = WaitHardDiskIde ();
   if ((xx& 0x58)!=0x58)
    {
         _asm{
       pop esi
       pop edx
       pop ecx
       pop ebx
       pop eax
         iretd
         }
   }
    //读取硬盘控制器的全部信息
   for (i=0; i<256; i++) {
         _asm{
               mov dx, 0x1f0
               in ax, dx
               mov temp, ax
         }
         serial[i] = temp;
   }
         _asm{
       pop esi
       pop edx
       pop ecx
       pop ebx
       pop eax
         iretd
    }
}
```

说明：硬盘控制器的所有信息都保存在 serial 数组中，硬

用 ATL 模板库创建实现 FTP 功能的 COM 组件

章 兵 刘瑞祥

一、概述

FTP 协议是网络世界中重要的数据传输协议之一，它为网络用户异地存取文件提供了很大的方便。除此之外，FTP 协议在文档管理软件中还扮演着很重要的角色。一个公司或企业通常拥有自己的局域网，所有的科室和部门都通过网络进行文档的调阅和查询，管理部分用数据库实现，文档的异地调阅、审批、修改则由 FTP 模块完成，FTP 功能模块是这类软件的重要组成部分。

一个采用 COM 规范开发的对象具有良好的可重用性，可以方便地在任何支持 COM 技术的开发工具中使用，Windows 本身就是由大量的组件堆积而成的。早期用 VC 开发 COM 组件比较繁琐，程序员必须亲自与 IUnknow，IDispatch 接口交互，每一个功能的实现都要不厌其烦的一次次地与这些接口打交道。Microsoft 推出的 ATL 活动模板库使得现在开发 COM 组件变得非常容易，它隐藏了接口的实现细节，并自动生成接口所需的代码。使用 ATL 可以生成独立、快速、可以被任何语言直接调用的 COM 组件。

本文介绍的就是使用 COM 组件技术实现的可被其他开发工具重用的 FTP 组件。组件代码的开发环境是 Windows NT4 + SP6，开发工具是 Visual C + + 6.0 。

二、接口定义

1. IDL 语言简介

一个 COM 组件并不需要让使用者知道功能实现的细节（通常使用者也不想知道），它只需要提供给使用者一些方法（Method）和属性（Property）用于实现某些功能，这些方法和属性可以称之为接口。这些方法和属性对于 C/C + + 语言来讲就是函数和变量，由于不同的开发工具对于函数和变量的定义各不相同，如何让其它支持 COM 技术的开发工具明白这些接口的定义呢？就像两个讲不同语言的人可以通过翻译进行交流一样，我们也使用“翻译”将这些 C/C + + 声明解释为其他开发工具能理解的接口定义，这个“翻译”就是“接口定义语言”（IDL）。

IDL 语言使用一些特殊的标记让所有支持 COM 技术的开发工具明白哪些是属性，哪些是输入输出行参，哪些是返回值：

[in]：表示这是接口方法的一个输入参数。COM 支持多种数据类型，但是为了最大限度地兼容其他语言，我们通常用 Long 表示整数（即使你用了 BOOL 类型的参数也会被 IDL 解释为 Long 型），BSTR 表示字符串。COM 使用的字符串是 Unicode，也就是 wchar_t，与 Ansi C 的字符串定义不同，它的一个字符宽度为 16 位并且一个字符串允许存在多个‘h0’。

[out]：表示这是接口方法的一个输出参数，在 ATL 中它只能以指针的形式存在。使用这种方式输出形参很难被其他语言使用。

[out retval]：表示这个接口方法是一个有返回值的函数，其返回值由本标记所定义。ATL 为每个函数添加了 HRESULT 作为返回值，这个返回值只在组件内部是可见的，作为 COM

盘序列号位于此数组 10 至 20 单元中，在使用时需要将高低字节颠倒一下，得到的才是完整的序列号。

四、 结束语

本文通过中断异常技术使运行于 Ring3 的应用程序转入 Ring0 级，执行被 Windows 系统屏蔽的特权指令，直接访问硬盘控制器，获取了硬盘的序列号。这种方法无需编制复杂的 VxD，可以在单一的开发平台下编译运行，本文采用的是 Visual C + + ，因为关键的指令均采用汇编语言，稍作修改，就可以在其他的开发环境中编译。这种通过中断异常改变特权级的方法可以应用于许多其它的特殊场合，它的意义远不止仅仅去读取硬盘序列号，这里以它为例，旨在给大家抛砖引玉。

最后，本文的缺陷在于这种方法仅适用于 Windows9x 系统，不能在 Windows NT 工作；而且，虽然绝大多数硬盘如希捷、昆腾、IBM、Maxtor 等厂家生产的硬盘都有序列号，但是，还有一小部分硬盘（如三星公司生产的部分硬盘）根本就不存在序列号。

参考文献

1. Microsoft. MSDN Library Visual Studio 6.0 release. 1998

2. Matt Poetrek. Windows 95 系统编程奥秘. 电子工业出版社 1996

3. 杨亮. Windows 深入剖析 - 内核篇. 清华大学出版社，1997

4. Scott Stanfield Ralph Arvesen 著 华译工作室译. Visual C + + 4 开发人员指南. 机械工业出版社，1997

（收稿日期：2001 年 1 月 9 日）

对外提供的方法仍然是没有返回值的过程（procedure）。只能用[out retval]标记方法的返回值，很显然一个方法中最多只能有一个[out retval]标记。

2. 组件属性定义

（1） URL

定义为 BSTR 类型的属性，URL 用于描述 FTP 连接的 FTP 服务器的 IP 地址，也可以是主机名。

（2） UserName

定义为 BSTR 类型的属性，UserName 用于描述 FTP 登录到服务器所用的用户名，匿名登录通常使用 "Anonymous"。

（3） Password

定义为 BSTR 类型的属性，Password 用于描述当前用户登录使用的密码，当使用匿名方式登录时可以任意输入一个 E-mail 地址（甚至可以是不存在的），如 "123@123.123"，也可以为空。

（4） TimeOut

定义为 Long 类型的属性，TimeOut 用于描述网络连接中的超时设置，当网络操作等待超过 TimeOut 时就返回，防止无限制的堵塞。

（5）Port

定义为 Long 类型的属性，Port 用于描述 FTP 服务器实现 FTP 协议的端口，通常是 21，但某些特殊功能的服务器也使用其它端口。

（6） ErrorCode

组件通常关心的是功能的实现，至于与用户的交互应由使用这个组件的应用程序来完成。程序运行通常会发生意外错误（实现网络功能的程序更是如此），由于组件的封闭性，应用程序不知道出错的原因，也就无法给用户提供正确的错误警告或参考。定义 Long 类型的属性 ErrorCode 就是用来描述出错的原因，当错误发生时，应用程序检查 ErrorCode，得到出错原因并采取相应的措施。

（7） NotifyWnd

Long 类型的属性，NotifyWnd 是提示窗口的句柄，通常是 Edit 或 Static 控件的窗口句柄（HWND），当需要提示用户当前文件传输完成百分比和网络传输速率，可以将 Edit 或 Static 控件的窗口句柄赋值给 NotifyWnd 就可以了。

属性定义按以下步骤实现。使用 VC 的 ATL COM AppWizard 生成一个名为 TyFtp 的项目，选择组件类型为 Dynamic Link Library，不使用 MFC 支持（那样会使组件变得庞大并且依赖于 MFC 的动态链接库）。打开 Insert 菜单执行 New ATL Object 菜单命令，在弹出的 ATL COM AppWizard 对话框中选择 Simple Object 图标，然后单击 Next 按钮，在属性对话框中输入 COM 组件的 C++ 包括类的 ShortName 为 "TyxxFtp"，其他属性取默认值，单击'确定"完成操作。

这时在 WorkSpace 的 ClassView 标签下就有 TyxxFtp 组件的包括类和接口定义两个分支。在 ITyxxFtp 上点右键，在弹出的菜单上可以看到 Add Property 命令，重复执行该命令完成所有属性的添加，注意 ErrorCode 属性是只读的。

添加完属性后还要在 CTyxxFtp 类中为每个属性添加存储变量。打开 TyxxFtp.h，添加以下代码：

```
TCHAR m_szUrl[256];
TCHAR m_szUserName[32];
TCHAR m_szPassword[32];
HWND  m_NotifyWnd;
long  m_lTimeOut;
long  m_lErrorCode;
long  m_Port;
```

3. 方法的定义

（1） Connect

Connect 接口定义为：HRESULT Connect [out retval] BOOL * pRtnVal，对属性赋值后就可以调用 Connect 进行网络连接，返回值是 TRUE 表示连接成功。

（2） DownLoad

DownLoad 接口定义为：HRESULT DownLoad [in] BSTR bsRemoteFile [in] BSTR bsLocalFile [out retval] BOOL * pRtnVal，执行 Connect 方法后就可以使用 DownLoad 方法从服务器下载文件。输入参数[in] BSTR bsRemoteFile 可以是文件在服务器的全路径，也可以是文件名，函数内部根据 bsRemoteFile 参数第一个字符是不是 'h' 或 '/' 来判断 bsRemoteFile 是全路径还是文件名，如果是文件名则默认目录是当前目录。[in] BSTR bsLocalFile 是另存为本地的文件名。

（3） UpLoad

UpLoad 方法用来向 FTP 服务器上传一个文件，其接口定义为：HRESULT UpLoad [in] BSTR bsLocalFile [in] BSTR bsRemoteFile [out retval] BOOL * pRtnVal，bsLocalFile 是上传的本地文件名，bsRemoteFile 是服务器端文件名，它的格式与 DownLoad 相同。

（4） DeleteFile

DeleteFile 方法用来从服务器上删除一个文件，传入参数是需要删除的文件名，参数 bsRemoteFile 的用法与 DownLoad 和 UpLoad 相同，接口方法定义为 HRESULT DeleteFile [in] BSTR bsRemoteFile [out retval] BOOL * pRtnVal。

（5） Disconnect

Disconnect 方法切断与 FTP 服务器的连接，Disconnect 与 Connect 是一对开关操作，必须成对使用。Disconnect 的定义为：HRESULT Disconnect。

为组件添加方法与添加属性的方法相似，在 WorkSpace 的 ClassView 标签下的 ItyxxFtp 分支上点右键，在弹出的菜单上可以看到 Add Method 命令，重复执行该命令完成所有属方法的添加。

三、功能的实现

1. 本文用到的 FTP 命令

FTP 协议通过应答命令来确保在文件传输过程中请求和动作的同步，并且保证用户过程总能知道服务器的状态。每个命令至少会产生一个应答，也可以有多个应答。FTP 应答由三个数字组成，随后一个是空格，然后是文字，最后是 < Ctrl>。

USER 和 PASS：当连接到 FTP 服务器后，需要向 FTP 服务器提供用户帐号和密码才能完成登录，USER 和 PASS 命令分别用来发送用户帐号和密码，其格式分别为：USER username < Ctrl>和 PASS password < Ctrl>。

SIZE：SIZE 用来取得文件的大小，命令格式为 SIZE filename < Ctrl>。

PORT：PORT 命令指定主机 IP 地址和数据端口，这个命令的参数是 32 位的因特网主机地址和 16 位的 TCP 端口地址的串连。PORT 命令的格式为：PORT h1、h2、h3、h4、p1、p2 < Ctrl>，h1 是主机地址的高 8 位，p1 是 TCP 端口地址的高 8 位。

STOR 和 RETR：STOR 和 RETR 用来向服务器上传和下载文件，命令格式为：STOR filename < Ctrl>和 RETR filename < Ctrl>。

2. 实现接口方法

因为组件应突出其功能性且尽量少地依赖界面，所以本文介绍的组件没有界面窗口，在使用 SOCKET 函数时我们选择 Unix 风格的 Socket 函数而不是使用 WinSock 函数。

(1) Connect

Connect 完成网络连接和用户登录两个任务，首先调用 WSAStartup 初始化网络环境，然后用 ConnectorBind 函数与服务器建立连接，如果连接成功就调用 DoCommand 函数发送用户帐号和密码，登录成功返回 TRUE，否则返回 FALSE。ConnectorBind 函数用于与服务器建立连接或绑定某个端口；bIsConnect 为 TRUE 时表示网络连接；FALSE 表示绑定端口；DoCommand 函数用于向服务器发送命令，等待并处理服务器应答码，当参数 lpszCommand 赋值为 NULL 时仅仅用来接受服务器应答。

(2) DownLoad

DownLoad 函数完成文件下载工作，下面仅就几段代码作一些说明：

```
wsprintf(szCommand, "SIZE %s\r\n", szFileName);
if((m_lErrorCode = DoCommand(m_sktControl, szCommand, m_lTimeOut, (LPVOID)& iFileSize)) != E_NOERROR)
{
    if(m_lErrorCode == E_NETFTPCOMMANDNOTSUPPORT)
        iFileSize = -1;
    else
    {
        closesocket(m_sktListen);
        *pRtnVal = FALSE;
        return S_FALSE;
    }
}
```

以上这段代码用于得到文件的大小，有时返回错误只是因为服务器不支持 SIZE 命令，所以多做了一个判断。

```
m_hEvtListen = ::CreateEvent(NULL, FALSE, FALSE, NULL);
......
DWORD ThreadID;
HANDLE m_hListenThread = ::CreateThread(NULL, 0, (LPTHREAD_START_ROUTINE)ListenProc,
    (LPVOID)& paratmp, 0, & ThreadID);
MSG msg;
while(WAIT_OBJECT_0 != ::WaitForSingleObject(m_hEvtListen, 0))
{
    ::GetMessage(& msg, NULL, 0, 0);
    ::TranslateMessage(& msg);
    ::DispatchMessage(& msg);
}
```

启动工作线程来完成下载工作。为了与线程同步需要当前线程处于等待状态，但是当前线程属于调用组件的应用程序的主线程，当文件比较大时会长时间堵塞主线程消息循环，导致应用程序如死机一样。为了避免这种情况，在等待工作线程时使用循环转发消息，从而减缓了消息堵塞。

(3) UpLoad

UpLoad 的工作流程与 DownLoad 一样，只是调用的命令不同。

(4) DeleteFile

DeleteFile 从服务器删除一个文件，参数可以指定文件的完整路径，也可以只是文件名，下面这段代码在程序中多次出现，用来区分字符串参数是完整文件路径还是只包含文件名，如果是完整路径则分离出文件名。

```
if((strRemoteName[0] == _T('/')) ||(strRemoteName[0] == _T('\\')))
    _tcscpy(szPathName, (TCHAR *)strRemoteName);
else
{
    _tcscpy(szPathName, _T("/"));
    _tcscat(szPathName, (TCHAR *)strRemoteName);
}
p = szPathName + _tcslen(szPathName) - 1;
while((*p != '/') && (*p != '\\'))
    p--;
_tcscpy(szFileName, p + 1);
*p = _T('\0');
```

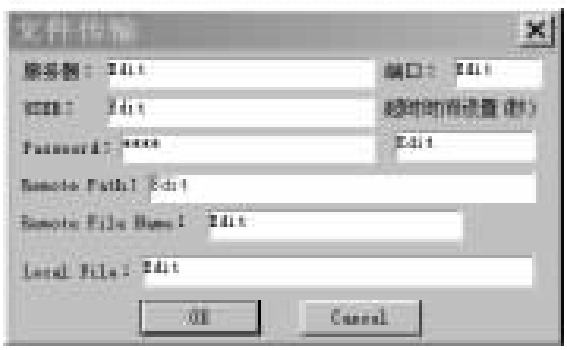
(5) Disconnect

Disconnect 向服务器发送 QUIT 命令，退出 FTP 对话，关闭所有打开的 Socket，并调用 WSACleanup 完成网络环境的清理工作。

四、测试程序

用 Visual C++ 6.0 生成该组件的一个基于对话框的简单

测试程序，项目名称为 Test，界面如下图所示。



对应的部分代码如下：

```
void CTestDlg: : OnBtnDownload( )
{
CFtpOptDlg dlg;
HRESULT hr;
hr = CoInitialize(NULL);
if(FAILED(hr))
return;
ITyxxFtp * pCom = NULL;
hr = CoCreateInstance(CLSID_TyxxFtp, NULL, CLSCTX_ALL,
IID_ITyxxFtp, (void * * )& pCom);
    . . . . . .
CoUninitialize( );
}
```

在运行测试程序之前应先用 regsvr32. exe 注册组件。具体到本文中应该先注册 TyFtp. dll 组件，然后运行 test. exe 进行测试。

参考文献

汪蒲阳 . 因特网应用编程 . 清华大学出版社，1999

收稿日期：2001 年 1 月 17 日）

北京彩虹天地信息技术有限公司

我们的未来——DRM 技术展望

DRM 是英文 ‘数字版权保护 ” 的缩写，它保护的对象是所有带版权的数字内容，包括声音、图象、文件、影像、数字文件、程序等等。对合法的使用者来说，他们感觉不到这种方法的影响；而对于非法用户，进行复制和传播将非常困难。DRM 就是这种方法的统称。在一切都渐渐数字化的今天，DRM 无疑会越来越重要。它已经成为数字化内容发展的基础和保证。

软件是最早受到人们重视并有保护需求的数字产品。‘加密狗 ” 就是在这样的社会背景和需求下应运而生的防止软件被盗版的产品。 ‘加密狗 ” 的原理是利用一个计算机上附加的硬件设备和开发商的应用程序进行交互，以确定该程序是不是合法。技术上主要集中在防止破解者的跟踪和对硬件的仿制，因此使用了诸如噪声、迷宫等技术手段，这也是当时反跟踪和身份认证的主要方法。类似技术的优点是：开发商操作简单、加密强度高、技术体系实用并易于理解。

互联网和新的数字产品的丰富，为 DRM 的发展提供了几乎无限的发展空间：我们的保护对象由一般的程序，发展到所有使用数字格式存储的数据；我们提供服务的目标，从国内的软件厂商发展到全世界的数字内容提供商。

我们必须使用一种新的 DRM 保护方法，它首先是低成本的，不会包含硬件 因此也会更稳定、更方便使用）；它与互联网技术紧密结合，这些技术包括 COM、ActiveX、ASP、Server / Client 等等；最后，它要能支持所有的数字存储形式，能够保护包括应用程序在内的各种数字内容，即它必须是通用的。

数字内容提供商将使用新的形式来使用我们的保护服务：由专业公司提供给内容提供商一整套开发工具和解决方案，由提供商建立自己的认证服务器。提供商发布的数字内容必须经认证服务器认证后才能够正常使用，而且这种认证是基于计算机的，即认证后的内容不能用在其它的计算机上。

这就是新 DRM 产品，它极大地扩展了我们的发展空间，并预示着今后几年的技术方向。当互联网的发展遭遇寒冬，人们逐渐意识到了对自主版权的数字内容进行保护的重要性。无疑，新 DRM 的未来会更好。

论选购防火墙的十大策略

防火墙为网络安全体系的基础和核心控制设备，其切断受控网络通信主干线，对通过受控干线的任何通信行为进行安全处理，如控制、审计、报警、反应等，同时，也承担了繁重的通信任务。由于自身处于网络中的敏感位置，还要面对针对自身的各种安全威胁。但目前防火墙功能都相差不大无非就是包过滤 地址转换 应用层过滤 攻击检测等等。那么用户采购防火墙时应考虑那些因素呢 有以下几点 1. 安全性 2. 稳定性 3. 高效性 4. 功能灵活性 5. 配置方便性 应用方便性）配置方便性 6. 管理方便性 7. 抗拒绝服务攻击 8 可靠性 9. 是否可针对用户身份进行过滤 10. 可扩展和可升级性。

以上就是防火墙的采购方法，选购防火墙时，如大抵能做到以上几点，防火墙的选购就不会成为难题了。

北京天融信网络安全技术有限公司

利用 Visual C + + 6.0 的 APPWIZARD 实现代码重用

亓 波

摘 要 本文论述了如何利用 Visual C + + 6.0 的 Appwizard 实现代码重用，通过该方法来设计程序有效地利用了以前程序的代码，大大节省了程序开发的时间。

关键字 Appwizard 代码重用

一、引言

在程序变得日益复杂的今天，重用已经变得尤为重要。Visual C + + 向用户提供了重用以前程序员的工作的方法，例如：Appwizard Classwizard 和 MFC。在一个程序中有好多部分可以重用，项目中可以重用的部分包括：

- 设计。
- 接口资源。用户可以重用控制、图标、菜单、工具栏或者整个对话框。
- 项目设置。无论是隐藏的链接器设置，还是工具栏的布置，用户的工作环境必须适合于自己的需要。用户可以快速地获得适合于正在完成的每一个项目的设置，因为用户可以重用上一次的设置方案。
- 文档。例如标准命令 FILE 等的帮助文件。

二、生成要重用的项目

在应用程序开发过程中尤其是应用程序的界面开发过程中，良好的界面会使程序增色不少。用 VC 经过一系列的工作做出了一个很好的界面 比如工具条的热点图象等等），而你又想在以后开发程序过程中也有同样的界面，这时便可以考虑创建一个定制的 Appwizard。

生成要重用：

1）利用 MFC Appwizard (exe）创建一个项目，把该项目命名为 Example，单击 "step1 " 中的 "Finish " 按钮，接受所有的 Appwizard 缺省值。

2）将 res 子目录下的 toolbar. bmp 拷贝为 toolbar1. bmp，然后将其作为一个位图资源插入资源中，并取其 ID 为 IDB_MAINFRAME。

3 修改 IDR_MAINFRAME 工具条，用很淡的颜色 例如淡灰色）来取代它原来的颜色。如下所示：

4）修改 CMainFrame OnCreate LPCREATESTRUCT lpCreateStruct 函数。 函数见文后）。

5）编译链接，生成应用程序。

程序运行时界面如图 1 所示：

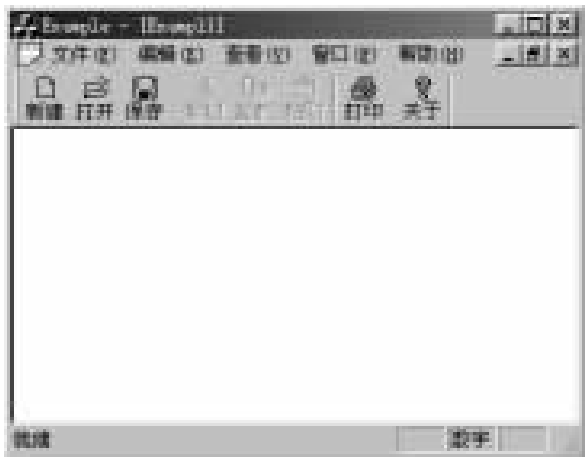


图 1 将鼠标在工具栏上移动时，工具栏的颜色将变成彩色

三、定制 Appwizard 实现重用

1）选择 "File " "New " 单击 "Project " 选项卡。选择 "Custom Appwizard "，取 Project Name 为 HotImageWizard。

2）选择 "An existing project "，不需要编辑这个 Wizard 的名字，点击 "Next "。

3 浏览 Example 的项目文件 Example. dsp 单击 "Finish "。

4 编译，在 Microsoft Visual Studio hCommon hMSDev98 h Template 下生成 HotImageWizard. awx 和 HotImageWizard. pdb 文件。

现在就完成了定制的 Appwizard - HotImageWizard Appwizard。

四、使用定制的 Appwizard

选择 "File "，再选择 "New "，然后单击 "Projects " 选项卡。HotImageWizard Appwizard 已被添加到左侧的选择列表

中 如图 2 所示 。使用它如使用 MFC AppWizard 一样简单。要想删除该 Wizard，只要删除 HotImageWizard. awx 和 HotImageWizard. pdb 文件即可。也可以把 HotImageWizard. awx 和 HotImageWizard. pdb 存在另外一个目录里，需要恢复 HotImageWizard 时只要把这两个文件拷回 Microsoft Visual Studio h Common hMSDev98 hiTemplate 即可。

五、总结

创建一个定制的 Appwizard 通常有三种方法：使用现存的 AppWizard 作为创建开始点、使用现存的一个项目作为创建开始点，或者完全的从头做起。本文只介绍了使用现存的一个项目作为创建开始点创建 Appwizard 的过程，有兴趣的读者可以详细查看 VC 的帮助。也欢迎与我联系共同探讨，我的 E - MAIL 是：qibo2000@263. net；通讯地址：中国科学院光电技术研究所一室　成都市双流 350 信箱） 610209

注：本文所有程序均在 Win98　Visual C + + 6. 0 下调试通过。

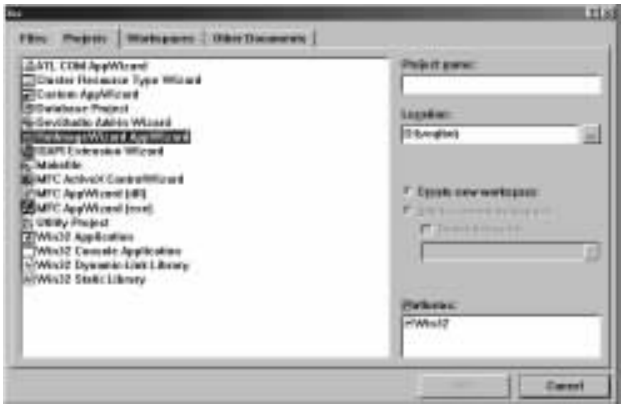


图 2　HotImageWizard AppWizard 已被添加到左侧的选择列表中

参考资料

Visual C + + 6 开发使用手册 . 机械工业出版社

附 ： CMainFrame　OnCreate LPCREATESTRUCT lpCreateStruct 函数：

```
int CMainFrame: : OnCreate(LPCREATESTRUCT lpCreateStruct)
{
  if(CMDIFrameWnd: : OnCreate(lpCreateStruct) = = -1)
    return -1;
   if(!m_wndToolBar. CreateEx(this) ||
    !m_wndToolBar. LoadToolBar(IDR_MAINFRAME))
    {
    TRACE0(″Failed to create toolbar\n″);
    return -1;      // fail to create
    }
   CImageList img;
if(!img. Create(IDB_MAINFRAME, 16, 0, RGB(128, 128, 128)))
    {
    TRACE0(″Failed to load hot images\n″);
    return -1;
    }
m_wndToolBar. GetToolBarCtrl(). SetHotImageList(& img);
img. Detach();
if(!m_wndStatusBar. Create(this) ||
      !  m_wndStatusBar. SetIndicators(indicators,  sizeof
(indicators)/sizeof(UINT)))
    {
    TRACE0(″Failed to create status bar\n″);
    return -1;      // fail to create
    }
// TODO: Remove this if you don´t want tool tips or a resizeable toolbar
m_wndToolBar. SetBarStyle(m_wndToolBar. GetBarStyle() |
    CBRS_GRIPPER | CBRS_BORDER_3D | CBRS_TOOLTIPS
| CBRS_FLYBY | CBRS_SIZE_DYNAMIC);
// Add text to each button
for(int i = 0; i < m_wndToolBar. GetCount(); i + + )
{;
    UINT id = m_wndToolBar. GetItemID(i);
    CString s;
    if(!s. LoadString(id)) continue;
    int j = s. Find(_T(´\n´));
    if(j < 0) continue;
    s = s. Right(s. GetLength() - j - 1);
    m_wndToolBar. SetButtonText(i, s)
}
// Add drop - down arrow to File|New button
CRect rect;
// Adjust sizes to include text CRect rect;
m_wndToolBar. GetItemRect(0, & rect);
m_wndToolBar. SetSizes(rect. Size(), CSize(16, 15));
// TODO: Delete these three lines if you don´t want the toolbar to be dockable m_wndToolBar. EnableDocking(CBRS_ALIGN_ANY);
m_wndToolBar. EnableDocking(CBRS_ALIGN_ANY);
EnableDocking(CBRS_ALIGN_ANY);
DockControlBar(& m_wndToolBar);
return 0;
}
```

收稿日期 2000 年 12 月 11 日

五年磨一剑，锋从磨砺出

金仕达多媒体通过国家软件企业认证

日前，根据国务院 54 号文件和上海市 18 号文件精神，金仕达多媒体公司顺利通过国家软件企业认证，成为第二批通过国家认证的软件企业，将享有税收减免、人才引进、公司上市改制等一系列优惠政策。

金仕达多媒体成立于 1996 年，曾是中国最早和最大的专业从事多媒体软件产品开发的软件企业，出品过 毁灭者》、神秘岛》、仆大名著》等有较高知名度的家庭知识娱乐产品。近年来，随着经营规模的不断扩大，金仕达多媒体先后涉足电子商务、网站建设、企业管理系统和本地化领域，先后与上海证券交易所、Intel 公司、罗氏制药等国内外著名公司机构展开合作，已经成为上海软件行业知名度较高的公司之一。

近年来，国家将软件等高科技产业作为国家未来的产业战略方向进行了重点扶持，随着相关认证和政策的推出，将对金仕达多媒体等国内先进软件生产厂商的发展起到重要的推动作用。到目前为止，上海地区只有 80 余家软件企业通过该认证。

在 VC + + 中实现对话框及其元素的等比缩放

叶德谦 马勤勇

摘 要 本文给出了在 VC + + 中用一个对话框基类实现对话框及其元素的等比缩放方法。使用此基类的对话框界面可适应不同分辨率显示和按用户的要求进行任意缩放，对话框内各个元素均按对话框大小进行等比变化。

关键词 分辨率，对话框，等比缩放，元素

引言

显示器可以设置成几种不同的显示分辨率。但是同样的元素如控件、字、图等在不同分辨率下显示的大小不同[1]，比如当分辨率为 640 x 480 时元素显示的较大，而在分辨率为 1024 x 768 时则较小。随着大尺寸、高分辨率显示器的日益普及，使用户在一个屏幕上同时观察多个完整界面内容成为可能。为达此目的，程序必须能实现界面的等比缩放功能。本文给出一种实现此功能的方法和程序实例。

该方法的基本思想是设计一个对话框缩放基类 CDialogEx，由此类派生的对话框可实现等比缩放功能：随着对话框大小的调整，对话框上的所有元素的大小都按比例变化。另外还能记忆用户调整出的对话框大小和位置，以便下次显示时能恢复到原来状态。

一、 获得当前可执行文件所在目录

存储状态的文件可保存在当前可执行文件所在的目录下。首先可使用 GetModuleFileName 函数得到当前可执行文件的路径名[2]，再去掉此路径名尾部的文件名。下面用一个声明的函数来进行处理：

```
CString CDialogEx: : GetCurDir( )
{
TCHAR sFilename[_MAX_PATH];
  TCHAR sDrive[_MAX_DRIVE];
  TCHAR sDir[_MAX_DIR];
  TCHAR sFname[_MAX_FNAME];
  TCHAR sExt[_MAX_EXT];
    GetModuleFileName(AfxGetInstanceHandle( ), Filename,
_MAX_PATH);
_tsplitpath(sFilename, sDrive, sDir, sFname,  sExt);
  CString homeDir(CString(sDrive)  + CString (sDir));
  int nLen = homeDir. GetLength( );
  if (homeDir. GetAt(nLen – 1) ! = _T(´\\´))
    homeDir + = _T(″\\″);
  return homeDir;
}
```

二、恢复对话框主窗口的位置

当对话框结束时，应将其当前位置保存下来。可使用 CWnd GetWindowRect 来获得对话框所在位置，取得的值为 RECT 结构。

一个程序中可能有多个对话框，这样就要由 CDialogEx 派生出多个对应的对话框的类，因而需要在基类 CDialogEx 中能区分出此时显示的是哪一个对话框。 由于在一个程序中不同的对话框的窗口标题不同，因此可通过 GetWindowText 来获得窗口的标题。可采用 MFC 的 CMap 来存储对话框的位置，用对话框的标题字符串来映射窗口位置。声明如下的类成员变量：

```
typedef  CMap < CString, LPCSTR, CRect, CRect & > CMapC-
StringToCRect;
CMapCStringToCRect mapStrToRect;
```

在 CDialogEx OnDestroy 中添加下列代码，以便当对话框退出时将其位置数据保存到文件 dlgStatus. sav 中去：

```
CDialog: : OnDestroy( );
CRect rectDlg;
GetWindowRect(& rectDlg);
CString strWindow;
GetWindowText(strWindow);
mapStrToRect. SetAt(strWindow, rectDlg);
CFile mFile;
mFile. Open(GetCurDir( ) + ″dlgStatus. sav″,
CFile: : modeCreate | CFile: : modeWrite);
CArchive ar(& mFile, CArchive: : store);
mapStrToRect. Serialize(ar);
ar. Close( );
mFile. Close( );
```

在对话框初始化时，从 dlgStatus. sav 文件中读出保存的数据，从中获得当前对话框的位置。 在 CDialogEx OnInitDialog 中加入下面的代码来实现此功能：

```
CFile mFile;
if(mFile. Open(theApp. GetCurDir(  ) + ″ dlgStatus.  sav″,
```

```
CFile: : modeRead) & & mFile. GetLength()>0)
{
  CArchive ar(& mFile, CArchive: : load);
  mapStrToRect. Serialize(ar);
  ar. Close();
  mFile. Close();
  CString strWindow;
  GetWindowText(strWindow);
  CRect rectDlg;
  if(mapStrToRect. Lookup(strWindow, rectDlg))
   MoveWindow(rectDlg);
}
return TRUE;
```

三、实现对话框及其元素的等比缩放

将对话框 Styles 属性中的 Border 设置为 Resizing 后，就可以拉动对话框的边缘进行缩放。为使各元素的位置随着对话框的大小按比例进行变化，应知其原始位置和对话框此时的缩放比例。因此在初始化时应将对话框及所有元素的位置保存下来。对此需要在此类中添加一个成员变量：

CList < CRect CRect> listRect

在 CDialogEx OnInitDialog 中加入下列代码，将对话框及所有元素的位置保存在 listRect 中：

```
CRect rectWnd;
GetWindowRect(& rectWnd);
listRect. AddTail(& rectWnd);
CWnd * pWndChild = GetWindow(GW_CHILD);
while (pWndChild)
{
  pWndChild - >GetWindowRect(& rectWnd);
  listRect. AddTail(& rectWnd);
  pWndChild = pWndChild - >GetNextWindow ();
}
```

当调整对话框的大小时，系统将向此对话框发送 WM_SIZE 消息。在 CDialogEx OnSize UINT nType int cx int cy 中进行各元素位置和大小的处理：

```
CDialog: : OnSize(nType, cx, cy);
if(listRect. GetCount()>0)
{
  CRect rectDlgNow;
  GetWindowRect(& rectDlgNow);
  POSITION mP = listRect. GetHeadPosition();
  CRect rectDlgSaved;
  rectDlgSaved = listRect. GetNext(mP);
  ScreenToClient(rectDlgSaved);
  ScreenToClient(rectDlgNow);
  FloatfRateScaleX = (float)(rectDlgNow. right - rectDlgNow.
left) / (rectDlgSaved. right - rectDlgSaved. left);
  float fRateScaleY = (float) (rectDlgNow. bottom - rectDlg-
Now. top) / (rectDlgSaved. bottom - rectDlgSaved. top;
ClientToScreen(rectDlgSaved);
ClientToScreen(rectDlgNow);
LOGFONT stFont;
: : GetObject ( ( HFONT ) GetStockObject ( DEFAULT_GUI_
FONT), sizeof(stFont), & stFont);
strcpy(stFont. lfFaceName, "Times New Roman");
stFont. lfHeight = (long)(fRateScaleY * 14);
stFont. lfWidth = (long)(fRateScaleX * 7);
CFont * m_pCFont;
m_pCFont = new CFont;
m_pCFont - >CreateFontIndirect(& stFont);
CRect rectChildSaved;
CWnd * pWndChild = GetWindow(GW_CHILD);
while(pWndChild)
{
    rectChildSaved = listRect. GetNext(mP);
    rectChildSaved. left = rectDlgNow. left + (int)
  (rectChildSaved. left - rectDlgSaved. left) * fRateScaleX);
    rectChildSaved. top = rectDlgNow. top + (int)
((rectChildSaved. top - rectDlgSaved. top) * fRateScaleY);
 rectChildSaved. right = rectDlgNow. right + (int) ((rectChild-
Saved. right - rectDlgSaved. right) * fRateScaleX);
 rectChildSaved. bottom = rectDlgNow. bottom + (int)((rect-
ChildSaved. bottom - rectDlgSaved. bottom) * fRateScaleY);
   }
   ScreenToClient(rectChildSaved);
   pWndChild - >MoveWindow(rectChildSaved);
   pWndChild - >SetFont(m_pCFont);
 pWndChild = pWndChild - >GetNextWindow();
}
Invalidate();
```

要使对话框及其元素位置自动适应各种分辨率，可用 GetSystemMetrics 获取当前屏幕的分辨率来确定它们应该缩放的比例系数。实现方法是在 CDialogEx OnInitDialog 中调用 MoveWindows 时，对话框显示前乘上此比例系数，而在 CDialogEx OnDestroy 保存窗口矩形之前除以此系数。

四、 结束语

本文给出了在 VC + + 中实现对话框界面及其元素等比缩放的编程方法和完整程序。在一个德国物业管理工程项目中笔者采用了此技术，实践证明该方法简便易行，显示效果良好。

参考文献

1 David J. Krulinski. Inside Visual C + +. Microsoft Press 1999

2 Viktor Toth. Programming Windows 98 / NT Unleashed. Publishing House of Electronics Industry 1999

收稿日期：2000 年 12 月 27 日）

在 VFP 中应用模型库快速生成数据库框架的方法

张夏林

关键字 数据库,模型库,CASE 技术,数据模型,表结构继承性

数据模型是一个数据库系统的核心和基础，也是建立一个数据库系统要做的第一项重要工作。系统设计人员把设计好的数据模型文档交给你，如果使用 VFP 建系统，你就该用熟悉的表设计器 (下图）开始单调而漫长的数据表结构设计工作。小的系统还好，要是建一个大型的数据库，这会是一项繁琐而枯燥的任务，远没有编程痛快。笔者曾经负责一个大型的石油勘探数据库系统的开发，要建的数据表有 158 个，字段共 2856 个，预计要一个星期才能完成。能不能找一种方法，快速地完成这些简单、重复的表结构设计工作呢？编程实现也许是个好方法。经过反复尝试，终于找到了效率极高的好方法，一周的工作一天全部完成。这里把该方法介绍给大家，希望能帮助你提高建表效率。

图 1 用表设计器进行数据表结构设计工作

首先从数据模型文档着手 表 1 示），设法将其改造为数据表的模型库。方法是，把系统设计所得的数据模型文档先存为 EXCEL 文件，再导入 VFP6. 0 成为数据库表，修改字段名，增加一列数据表文件名 - tablename C 12 ，存储数据表的名称，要保证一个表的所有字段的 tablename 值相同；不同表有不同的 tablename 值。Tablename 可以用程序在每行写入一个字符串加一个序号作为不同表的文件名，程序根据 no 的值来判断一个表的开始和结束，完成后，模型库就建好了 表 2 示）。如果你没有数据模型文档，直接按表 2 的样式，在数据表中用复制、拷贝的方法 “建 ”表，也比用表设计器快得多。

表 1 AW01 地质调查工作量数据模型 部分）

序号	数据项名称	代码	类型	宽度	小数位	单位	主键	外键	空值
1	单位	DW	VARCHAR	3			★	★	★
2	统计年度	TJND	VARCHAR	4			★		★
3	构造单元代码	GZDYDM	VARCHAR	9					
4	构造单元名称	GZDYMC	VARCHAR	30				★	★

表 2 数据表模型库 部分）

tablename	no	ccaption	fieldname	ftype	nlength	nprecision	key	forkey	cnull
AW01	1	单位	DW	VARCHAR	3		★	★	★
AW01	2	统计年度	TJND	VARCHAR	4		★		★
AW01	3	构造单元	GZDYDM	VARCHAR	9				
~	~	~	~	~	~	~	~	~	~
AW02	1	单位	DW	VARCHAR	3		★	★	★

由表 2 中可以看出，模型库中保存了关于系统数据模型 即要建立的所有数据表）的全部基本信息。

然后，就该编个小程序，从模型库中提取各个数据表的基本信息，快速自动生成数据库表。程序 CreateTable. PRG 源码见所附清单。

最后，用表设计器打开各个已经生成的数据表，稍加检查修改就大功告成了。

应用意义，该方法属于软件辅助开发 CASE 技术 的范畴，给数据表结构带来可继承性。如果有设计规范的数据库模型文档 如表 1），使用以上方法生成数据库系统的框架是省时、省事的高效方法，不必再用表设计器一个一个地设计表；如果没有设计规范的数据库模型文档，你可以建一个空的模型库 如表 2），在模型库中 “设计 ”表，设计完后一次全部生成，这也比用表设计器一个一个地设计表快速、高效。同时，这样的模型库 如表 2）只要稍做修改还可以被用在相似系统的构建中，数据表结构有了可继承性。

注意：每次运行以下的数据表生成程序时，必须保证。1. 当前活动数据库中包含模型库表 (即程序中的 TableStru. dbf）；2. 用于保存将生成数据表的文件夹中没有与将生成的数据表同名的文件存在 如程序中 D hOil hTableStru h文件夹中不能有 AW01. DBF）；3. 模型库中同一个表的字段必须互不相同；4. 模型库中 nLength (字段宽度）列可以为 “NULL ”，但不能为 0。

附：程序清单

```
* * * Create Table With Fields in a Model table * * *
PROCEDURE CreateTable
```

下转第 65 页

运用面向对象的设计思想开发分层数据库应用程序

金旭亮

提　要　本文主要讨论如何在实际的程序开发实践中以面向对象的理论作指导，具体设计并实现一个分层的数据库应用程序，强调了理论指导在程序设计中的重要作用，并给出了一个具体的程序开发范例以说明如何运用面向对象的设计思想来开发分层数据库应用程序。

关键词　面向对象，分层，数据库，软件体系结构

一、面向对象技术与分层的软件体系结构

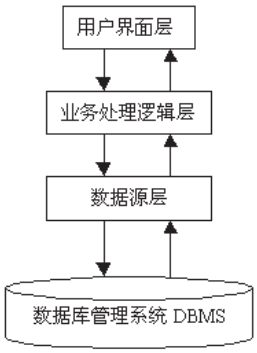
面向对象技术已成为当前软件开发技术的主流，面向对象的设计方案将数据与操作封装在同一个对象结构中，并引入了多态、重载、动态联编等特性，使开发出的软件易于维护，易于扩充。

软件开发者在应用面向对象的设计方法过程中，逐步意识到合理的软件体系结构的重要性，在实践的基础上总结出了不少行之有效的软件体系结构模式，针对特定的问题选择最合适的软件体系结构，就成为软件项目开发中极其重要和关键的一步。

目前，面向对象的开发工具已有不少，比如微软的 Visual Studio 系列，以及 Borland 公司的 C + + Builder 等，都可以用来进行面向对象的软件开发。与此同时，在软件体系结构上，人们也已经提出了不少模式，如客户机/ 服务器模式、过滤器模式、分层模式等。

因此，现在的问题是如何针对特定的实际问题选择合适的软件体系结构，并采用适当的面向对象技术去实现它。这个问题很大，也很难有一个完美的回答。下面本文将对应用广泛的数据库程序开发领域的一个范例进行分析，以此作为一个可行但不一定是最优的解答。

目前，在数据库的应用程序中，采用分层的软件体系结构具有相当多的优越性。一个典型的分层数据库应用程序体构可以图示如下：



上图中，各种数据存储于数据库中；用户界面负责与用户交互，接收用户的输入并将其请求传给业务处理逻辑层；业务处理逻辑层负责处理用户请求，比如查询某客户下的订单之类，它向数据源层提出数据请求，然后从数据源层提取所需的数据并进一步进行加工处理，然后，返回给用户界面并显示出来；数据源层是真正与数据库打交道的地方，它接收上层提出的数据请求，从数据库中获取数据，再向上层递交结果。

每一层都只向下一层提出请求，下一层向上一层提供服务，不允许跨层调用。

这种分层的软件体系结构具有结构清晰、易于维护、易于组织开发的优点。同时，各个层次之间是一个相互提供服务的关系，在开发中每一层可以再细分为若干对象，每一对象完成相应的功能，在对系统进行维护或二次开发中，通过加入新的对象，或修改已有的对象，可以将对软件其它部分的影响减少到最低限度。

这种分层的软件体系结构与面向对象开发工具的结合，是目前开发基于数据库的应用程序的理想策略。

二、分层数据库软件设计开发实例

考虑这么一个数据库软件需求：

用户希望能对数据库的内容加密，而不对数据库文件本身加密。

首先，采用什么体系结构？这需要认真仔细地比较和权衡，最终我们选择了分层的数据库软件体系结构；

其次，采用什么开发工具。在实践中，VB 被广泛用来开发数据库应用程序，但 VB 其实是一种“基于对象”而不是真正的“面向对象”的开发工具，因为它缺乏像多态、重载、继承等面向对象技术的关键特性，正因为如此，许多程序员在用 VB 进行开发时，往往只注重了对软件功能的实现，不注意以面向对象的思想来编写代码。但是不是必须要用 VC 或 C + + Builder 之类来实现呢？当然可以，但我想指出的是，只要开发人员用面向对象的设计思想来考虑问题，用 VB 同样可以开发出面向对象的程序。本文就以 VB 为范例。当然，其功能同样可以由 C + + 来实现，而且可以完成得更好。

第三，考虑具体用什么数据库技术来实现软件需求。

在 VB 中，利用各种绑定的数据控件和 ADO 数据引擎直接访问数据库是许多程序员喜欢采用的技术，然而这种技术在带来方便的同时也带来了麻烦，它实际上是将用户界面层直接与数据库相连，从而破坏了分层的体系结构，这种方式开发很小的程序是合适的，但对于上了一定规模的应用程序而言，将会使软件项目的开发工作变得极难管理，各种功能的代码全都混杂在一起，最后会成为一团叫任何程序员看了都头痛的乱麻，牵一发而动全身。

另外，采用这种技术还有一个问题。比如，当用户在一个绑定到数据库的 DataGrid 控件中修改了某字段的值，然后，他移到了另外一条记录，这时，DataGrid 控件与 ADO 数据引擎就自动地把数据库中的内容更新了（这里以访问本地数据库为情况讨论）。而用户希望对输入的数据先进行某些处理，比如判断有效性，再加密之后才将更改后的记录集写入数据库。要达到这个目的，程序员不得不对数据绑定控件的许多事件编写代码，而且这些代码高度依赖于各种具体的数据绑定控件和窗体，控件种类一变，将会导致大量的代码需要重写。

所以，采用这种技术是不可取的。解决方法是采用面向对象的设计思想，将访问数据库的功能抽取出来，封装成一个或多个独立的对象，由这些对象封装所有对数据库的访问请求，并在其内部实现对数据的加密和解密，这些对象构成了数据源层。这样，业务处理逻辑层就无需考虑加解密的问题了。

从具体的实现上来说，采用 VB 可以很容易地解决这个问题，只需定义一个类模块，利用 ADO 内存记录集，让记录集对象与数据库表的连接断开，在内存中把数据处理完毕后再将其写回数据库中。

现在具体分析一下这个类应提供的功能：

1. 必须能提供一个可绑定的记录集对象，供数据绑定控件作为数据源，以方便程序员开发；

2. 这个记录集对象应与数据库断开，这样，程序可以在内存中对其进行加解密工作，再将结果写回数据库；

3. 这个类应能执行标准的 SQL 命令，并提供一些常用的操作，比如删除一个表。

4. 这个类应能通过改变一两个属性就连接到不同种类的数据库；

5. 这个类生成的对象可以反复使用而不必生成类的多个实例，以减少内存资源的消耗。

针对上述分析，设计了一个类模块 clsDataBase，可以方便地实现上述功能。它提供的基本接口如下：

属性：

1. clsConnectStr　用户必须先定义数据库连接字串，才能使用此类

2. clsSQLCommand　用户定义要使用的 SQL 命令，常用作生成数据源，也可以是 Delete 等进行数据操作但不返回记录集的 SQL 命令

方法：

1. 设置完以上两个属性后，用 clsGetRecordset 可以获取一个内存记录集对象；

2. 用 ExecSQLCommand 执行不返回记录集的 SQL 命令；

3. 在内存中对记录集进行修改以后，用 clsSaveRecord 将记录集保存入数据库中；

4. 如果需要从多个表中读取字段，但不需保存，则可以用 SQL 命令直接实现，如 Select 表 1 字段名 1，表 2 字段名 2， from 表 1，表 2，但如果需要保存，则必须把相关表的全部有关记录复制到内存，然后由程序来实现多表字段显示。调用 clsSaveRecordset 方法时，一定要保证内存记录集中包括表中所有的字段；

5. CopyTable　将一个表中所有的数据全部备份到另一个表中。应设置 clsConnectStr 属性；

6. DeleteTable　将一个表从数据库中删除；

7. ExecSQLCommand　用于执行一个 SQL 命令 不返回记录集；

8. IsTableExist　用于判断一个表是否存在。

使用示例：

● 先定义对象变量

```
Dim tempcls As clsDataBase
Dim rst As ADODB. Recordset
```

● 生成数据库对象并设置连接数据库连接字串属性 以 VB 提供的 Biblio. mdb 为例，采用 Jet OLEDB 4. 0 数据引擎）：

```
set tempcls = new clsDataBase
tempcls. clsConnectStr = "Provider = Microsoft. Jet. OLEDB. 4. 0
Data Source = C:\Program Files\Microsoft Visual Studio\VB98\
BIBLIO. MDB Persist Security Info = False"
```

这个属性其实是指定了要连接的数据库，更改这个字串，就可以连接到不同类型的数据库。

● 设定数据源 假定从 publishers 表中读取记录）

```
tempcls. clsSQLCommand = "Select * from publishers"
```

● 返回内存记录集，在 clsGetRecordset 函数中调用 Decrypt 对文本类型的字段进行了解密处理。

```
Set temprs = tempcls. clsGetRecordset
```

这个记录集来自 publishers 表，但现在与表完全断开了，除此之外，它同一般的 Recordset 对象没有区别；

● 在程序中用记录集进行各种处理 比如统计、分类、计算）

● 将修改过的结果写回数据库 在 clsSaveRecordset 过程中对文本类型的字段调用 Encrypt 函数进行加密处理

```
if tempcls. clsSaveRecordset( temprs) = -1 then
  msgbox "在保存记录时出错!"
else
```

msgbox "记录已成功保存"
end if

这时，只要保证 clsConnectStr 和 clsSQLCommand 属性的值与打开 temprs 时的值一样，就可以安全地将其写入数据库中。

使用这个类模块生成的对象可以复用，只需更改它的 clsConnectStr 和 clsSQLCommand 属性，再用 clsGetRecordset 方法就可以生成多个内存记录集。这样，一个窗体或一个模块就只需要一个 clsDataBase 就可以了。

当需要用其它的加密解密算法时，只需简单地将 Encrypt 和 Decrypt 两个函数替换就行了。这两个函数可以用 VB 编写，也可以用C++ 编写，然后编译成动态链接库供 VB 使用。

当需要增加功能，只需扩充类模块的接口，只要不更改接口的名字，更改类模块内部访问数据库的方法，并不影响外部用到此类对象的代码。

仔细研究文后所附的代码清单就会发现，对于一个熟练的 VB 程序员，实现这些功能是多么的简单，重要的是程序员必须能想到把这些功能封装成一个对象，能否主动地应用面向对象的设计思想解决实际开发中遇到的问题成为了关键。

笔者认为，面向对象的理论和技术目前已趋于成熟，尤其是国外，大型的软件几乎都是建立在面向对象技术的理论基础之上的，对于许许多多从事具体软件开发的程序员而言，应该努力地汲取国外这种先进的设计思想与设计理论，并将其融入自己的开发实践中，这种努力的结果，必将有利于中国软件开发水平的提高。

附：clsDataBase. cls 源文件部分源代码

注：必须将类的 DataSourceBehavior 属性设为 1 - vbDataSource，才能被数据绑定控件绑定。

```
'数据访问类模块清单
'开发者: 金旭亮
'最新修改时间: 2000 年 12 月 26 日
* * * * * * * * * * * * * * * * * * * * * * * * * * * * *
    '此模块没有用到 ADO 模型中的 COMMAND 对象
Option Explicit
Private m_ConnectStr As String '定义私有连接字串
Private m_SQLCommand As String '定义私有 SQL 命令
Private m_Connection As ADODB. Connection '定义私有 Connection 对象
Private m_RecordSet As ADODB. Recordset '定义私有记录集
Public Property Get clsConnectStr() As String
'实现 clsConnectStr 属性
clsConnectStr = m_ConnectStr
End Property
Public Property Let clsConnectStr(ByVal vNewString As String)
m_ConnectStr = vNewString
End Property
Public Property Get clsSQLCommand() As String
'实现 clsSQLCommand 属性
clsSQLCommand = m_SQLCommand
End Property
Public Property Let clsSQLCommand(ByVal vNewValue As String)
m_SQLCommand = vNewValue
End Property
Public Function clsGetRecordset() As ADODB. Recordset
'此函数返回一个记录集, 出错时返回 Nothing
If m_ConnectStr = "" Or m_SQLCommand = "" Then
 MsgBox "没有设置连接字串或 SQL 命令", vbCritical + vbOKOnly, "程序内部错误"
 Set clsGetRecordset = Nothing
        Exit Function
End If
'避免独占系统
DoEvents
Call OpenConnect '打开连接对象
Call OpenRecordset '打开记录集
Set clsGetRecordset = CopyRecordSet(m_RecordSet) '复制记录集到内存
Call CloseDataBase '清除所有连接和记录集对象
End Function
Public Function clsSaveRecordset(ByVal rsMemory As ADODB. Recordset) As Integer
'此过程将内存记录集保存入数据库表 strTable 中,
'出错时, 返回 -1, IsEmptyRecordset 是笔者设计的一个函数, 用于判断是否是空记录集
If rsMemory Is Nothing Or IsEmptyRecordset(rsMemory) Then
MsgBox "没有传送有效的内存记录集供保存", vbCritical + vbOKOnly, "出错"
ClsSaveRecordset = -1
Exit Function
End If
'防止独占全机器
DoEvents
Call OpenConnect
'更改 SQL 命令, 用 SQL 命令来删除记录集, Exchange() 是自定义过程, 用于将字串中的某个单词用另一个单词替换
m_SQLCommand = UCase(m_SQLCommand)
'将 SQL 命令中的 SELECT 用 DELETE 替换
        m_SQLCommand = Exchange(m_SQLCommand, "SELECT", "DELETE")
'删除原数据库中的记录
Call ExecSQLCommand
'换回原来的 SQL 命令
      m_SQLCommand = Exchange(m_SQLCommand, "DELETE", "SELECT")
Call OpenRecordset
Dim j As Integer
'开始复制记录到数据库表中
With rsMemory
   . MoveFirst
```

```
    While Not . EOF
        m_RecordSet. AddNew
    For j = 0 To . Fields. Count - 1 '需要复制每一个字段
'只对值不为 NULL 的字段复制若值为 NULL(在表中可能是自动编号字段),则不应设置,由 Access 自动生成
        If . Fields(j). Name <> "ID" Then '除去自动编号字段
If (Not IsNull(. Fields(j). value)) And trim(. Fields(j). value) <> "" Then
If rsMemory. Fields(j). Type = 202 Then '对文本字段加密
m_RecordSet. Fields(j). value = Encrypt(rsMemory. Fields(j). value)
    Else
m_RecordSet. Fields(j). value = Trim(. Fields(j). value)
                End If
            End If
        End If
        Next j
        On Error GoTo ErrorHandle
        m_RecordSet. Update
        On Error GoTo 0
        . MoveNext
    Wend
End With
Call CloseDataBase '关闭所有对象
clsSaveRecordset = 1
Exit Function
ErrorHandle:
MsgBox Err. Description & "," 在更新数据库记录时出错,没有更新数据表", vbCritical + vbOKOnly, "出错"
Err. Clear
clsSaveRecordset = -1
on error goto 0
Exit Function
End Function
Private Sub Class_GetDataMember(DataMember As String, Data As Object)
'用于提供数据绑定功能
If m_RecordSet Is Nothing Then
  Call OpenConnect
  Call OpenRecordset
End If
Set Data = m_RecordSet
End Sub
Private Sub Class_Initialize()
'初始化
m_ConnectStr = ""
m_SQLCommand = ""
Set m_Connection = Nothing
Set m_RecordSet = Nothing
End Sub
Public Function CopyRecordSet(ByVal rstSource As ADODB. Recordset) As ADODB. Recordset
'此函数将一个记录集复制到内存中,在此过程中实现对数据记录的解密,出错返回 Nothing
If rstSource Is Nothing Then
    Set CopyRecordSet = Nothing
    Exit Function
End If
'此过程较花时间,防止中断其它响应
DoEvents
Dim rstMemory As New ADODB. Recordset
Dim i, j As Integer
With rstMemory
    '复制记录集结构
    For i = 0 To rstSource. Fields. Count - 1
        . Fields. Append rstSource. Fields(i). Name, rstSource. Fields(i). Type, rstSource. Fields(i). DefinedSize, adFldIsNullable
    Next i
  '设置记录集的属性
    . CursorType = adOpenKeyset
    . LockType = adLockOptimistic
    On Error GoTo ErrorHandle '可能出错
    . Open
    On Error GoTo 0
     '记录集是否为空
  If IsemptyRecordset(rstSource) Then
 Set CopyRecordSet = rstMemory '为空则返回一个空的记录集
      Exit Function
  End If
  '记录集不空
  With rstSource
    . MoveFirst
    While Not . EOF
     rstMemory. AddNew
     For j = 0 To . Fields. Count - 1
If (Not IsNull(. Fields(j). value)) And . Fields(j). value <> "" Then
If rstSource. Fields(j). Type = 202 Then '解密,只针对文本类型
rstMemory. Fields(j) . value = Decrypt(rstSource. Fields(j). value)
                Else
rstMemory. Fields(j). value = . Fields(j). value
                End If
                End If
        Next j
            On Error GoTo ErrorHandle
            rstMemory. Update
            On Error GoTo 0
            . MoveNext
        Wend
        End With
        . MoveFirst
        Set CopyRecordSet = rstMemory
End With
Exit Function
```

```
ErrorHandle:
 MsgBox ″在复制生成内存记录集时出现错误! –在 CopyRecordSet()中″, vbCritical + vbOKOnly, ″错误提示″
      Set CopyRecordSet = Nothing
End Function
Private Sub OpenConnect()
′此函数按连接字串打开连接对象
If m_ConnectStr = ″″ Then
MsgBox ″数据源连接字串未设置!″, vbCritical + vbOKOnly, ″出错″
Exit Sub
End If
On Error GoTo ErrorHandle
If m_Connection Is Nothing Then
      Set m_Connection = New ADODB. Connection
      m_Connection. Open m_ConnectStr
Else
      ′若连接对象不为空,则先关闭再打开一次连接对象
      m_Connection. Close
m_Connection. Open m_ConnectStr
End If
On Error GoTo 0 ′及时关闭错误捕获
Exit Sub
ErrorHandle:
 MsgBox ″在打开由:″ & m_ConnectStr & ″指定的数据库连接时发生错误:″ & Err. Description, vbCritical + vbOKOnly, ″连接数据源 –在 OpenConnect()″
Err. Clear
On error goto 0
 Exit Sub
End Sub
Private Sub OpenRecordset()
′此过程打开记录集对象
If m_SQLCommand = ″″ Then
MsgBox ″没有指定 SQL 命令!″, vbCritical + vbOKOnly, ″出错″
End If
′若没有打开连接对象
If m_Connection Is Nothing Then
          Call OpenConnect
End If
      If m_RecordSet Is Nothing Then
          Set m_RecordSet = New ADODB. Recordset
End If
      On Error GoTo ErrorHandle
′m_Recordset 已有对实际对象的引用,此时可打开记录集
With m_RecordSet
      Set . ActiveConnection = m_Connection
      . CursorType = adOpenKeyset
      . LockType = adLockOptimistic
      . CursorLocation = adUseClient
      . Open m_SQLCommand, , , , adCmdText
End With
Exit Sub
′错误处理
ErrorHandle:
MsgBox Err. Description, vbCritical + vbOKOnly, ″程序出错″
Err. Clear
      On error goto 0
      Exit Sub
End Sub
Private Sub CloseDataBase()
′并闭记录集对象
On Error Resume Next
If Not (m_RecordSet Is Nothing) Then
m_RecordSet. Close
      Set m_RecordSet = Nothing
End If
′并闭连接对象
If Not (m_Connection Is Nothing) Then
      m_Connection. Close
      Set m_Connection = Nothing
End If
Exit Sub
End Sub
Public Sub CopyTable(strTable As String, strTableBak As String)
′将一个表中所有的数据全部备份到另一个表中.
′代码略
End Sub
Public Sub DeleteTable(strTable As String)
′将一个表从数据库中删除
′代码略
End Sub
Public Sub ExecSQLCommand()
′此方法执行由 clsSQLCommand 指定的 SQL 命令,但不返回记录集
If m_SQLCommand = ″″ Then
      Exit Sub
End If
On Error GoTo ErrorHandle
Call CloseDataBase
If m_Connection Is Nothing Then
      Call OpenConnect
End If
m_Connection. Execute m_SQLCommand
On Error GoTo 0
Exit Sub
ErrorHandle:
MsgBox Err. Description
Err. Clear
On error goto 0
End Sub
Public Function IsTableExist(strTableName As String) As Boolean
′此函数在本数据库中查找由 strTablename 指定的表名
′代码略
End Function
```

收稿日期 2000 年 12 月 27 日

SQL Server7.0 数据库导入数据时出现的符号问题及其解决方法

程 骏

最近我刚刚完成一个数据迁移项目。在项目过程中，我遇到了一些问题。主要原因是 Microsoft SQL Sever7.0 认为某些字符不是数字，而当我想把某些字符转化成数字时，SQL Server 表现的却很倔强。这样一来，问题就出现了。下面我就给您介绍一下完成该项目的过程。

数据准备

我先要写一个程序来导入一大批数据，然后再依次导入该批数据的修改记录。为了保证导入的次序不会搞错，每个导入文件都含有一个记录该文件最后修改时间的时间戳。

我选择 bcp、T-SQL 存储过程和 SQL 代理任务来完成这项工作。我先用 DTS 测试了项目的原型，证实导入是可行的。但考虑到要依据时间戳按次序导入数据，我还是觉得用 T-SQL 比为 DTS 包写 VBScript 来得方便。

我决定先将数据文件导入一个临时数据库，它只包含当前导入的数据。每当新的数据导入时，旧的数据就被清除。临时数据库的作用就是针对每一批数据，通过运行 T-SQL 存储过程或者 SQL 语句来检查数据的质量。如果发现错误数据，可以删除它，或者拷到别的表中以手工修复。

临时数据库另一个用处就是所有对临时数据库的操作都不会影响实际的生成数据库。您在操作中不需要备份临时数据库的 log，也没有什么误操作的危险性。但如果您直接将数据导入生成数据库的话，您就必须时时备份 log，以防止出现致命错误。但如果您使用临时数据库，您可以设定 SQL Server 自动控制 log 大小，也不必去备份临时数据库的 log，这样可以减少很多麻烦的操作。

在将数据导入临时数据库后，我对数据进行了测试，然后将他们导入生成数据库。开始时一切顺利，测试数据的表现很好，但一旦导入大批数据后，系统 down 掉了。检查后我发现，问题源于导入后出现了很多不正确的数据。

错误数据

为了将原始数据导入临时数据库，我设定临时数据库的每一列都是 varchar 类型。这样 bcp 程序就不会给出那些讨厌的错误提示了。我设计的临时表非常宽：有 184 列，这样足够容纳导入后每行超 8000 字节的数据。

但当我一次导入上万行数据时，数据就变的有些疯狂了。大多数的错误我都有办法对付，但其中一些却有些麻烦。表中有几列原先是字符类型，需要将他们转换成数字类型。下面是这些字符型数据的几个例子：

```
3.0
＊＊3＊＊
3.
3,000
3%
3
.
＊3/1.5%
$3,000
```

您可能已经注意到倒数第三个数据就是一个点。为什么会出现这么一些奇奇怪怪的数据不是我应该调查的问题。我的任务就是设法转换它们，并且导入生成数据库。

客户告诉我不必去管那些包含在“＊”内的数据，那么我的重点就在于转换那些真正的数字。

SQL Server 有一个函数 ISNUMERIC（）用于检验一个字符的值是不是数字。如果参数的值是数字类型，函数返回 1，否则返回 0。如果利用这个函数，我只要将每个字符送入该函数检验一下，是数字就进行转换。

但问题不那么简单。什么是数字类型呢？举个例子说明：

```
SELECT ISNUMERIC('3.0')
```

系统返回 1（是数字）

```
SELECT ISNUMERIC('＊＊3＊＊')
```

系统返回 0（不是数字）

这些结果都很正常，但试一下这个

```
SELECT ISNUMERIC('.')
```

系统返回 1，“.”是数字吗！！！

既然系统认为是，我们假设“.”代表 0。那么我试图转换它。可我发现几次尝试都失败了。

```
SELECT CONVERT(int, '.')
SELECT CONVERT(float, '.')
SELECT CONVERT(numeric(10,2), '.')
```

同样的问题出现在转换“3 000”的时候。

```
SELECT CONVERT(int, '3,000')
SELECT CONVERT(float, '3,000')
SELECT CONVERT(numeric(10,2), '3,000')
```

但是 SELECT ISNUMERIC('3,000')

系统返回的是 1。为什么系统认为是数字的字符却无法转换呢。可无论如何我要解决这个问题。

解决方法

当我在屋子里面转了几圈，下定决心写一个复杂的 CASE

语句，将 “. 从字符中剔除出去。在这之前，我打算再看一下 Books Online。

在 Books Online 中是这样定义 ISNUMERIC ( ) ，它对所有可以转换成数字类型的字符类型返回 1，包括浮点类型、货币类型和十进制数。

货币类型，我一直都没有试过，它能解决问题吗？我决定试一下。

```
SELECT CONVERT( money, '.')
```

系统返回 0. 结果令人满意.

```
SELECT CONVERT( money, '3, 000')
```

系统返回 3, 000. 00

可我并不打算使用货币类型来存储数据。这样写就解决问题了。

```
SELECT CONVERT (numeric (10, 2) , CONVERT (monery, '. )
```

有趣的是，如果您用 “$$ ”、 “. 和 代入 ISNUMERIC 试验时，结果都是 0。可见该函数认为带有一个 “. 、 “, ”或者货币符号的字符串都是数字，但不可以直接转换成数字类型，却可以直接转换成货币类型。

下面我用图表的形式总结一下我的发现：

	'.'	'.n'	'n.'	','	'n, nnn'	'$'	'$n'	'$, .'
Int	N	N	N	N	N	N	N	N
Float	N	Y	Y	N	N	N	N	N
Numeric	N	Y	Y	N	N	N	N	N
Money	Y	Y	Y	Y	Y	Y	Y	Y

注：表中 n 代表数字字符，N 代表 no，Y 代表 yes。

虽然把字符先转化成货币类型，再转换成数字类型看起来有些别扭，但这确实管用，或许您有更好的方法，请告诉我。

收稿日期：2000 年 12 月 27 日）

上接第 58 页

```
IF !USED('TableStru') && TableStru 是模型库的文件名
    USE D:\Oil\TableStru\TableStru.dbf IN 0 && 打开 TableStru 表, D:\Oil\TableStru\为路径
ENDIF
SELECT TableStru
GO TOP
SCAN
    SELECT TableStru
    cNewTableName =TableName && 生成数据表的文件名称，也是循环的控制条件
TableStruInfo = '' && 定义存放一个字符串，存放表结构信息
    nCount = 0 && 记数变量
DO WHILETableName = cNewTableName && 每个表处理一次
        IF ISNULL(nLength)
            cLength = "8"
        ELSE
            cLength = STR(nLength)
        ENDIF
        DO CASE && 字段的类型
        CASE ftype = "NUMERIC"
            cFieldtype = "N"
        CASE ftype = "DATE"
            cFieldtype = "D"
        OTHER
            cFieldtype = "C"
        ENDCASE
IF cType ="NUMERIC". AND. ! ISNULL(nprecision) && 小数位数
            cPrecision = "," + STR(nprecision)
         ELSE
            cPrecision = ""
        ENDIF
IF ISNULL(cNull) && 是否允许空值
  cFieldcNUll = " NULL"
        ELSE
            cFieldcNUll = " NOT NULL"
        ENDIF
     TableStruInfo = TableStruInfo + FieldName +" " + cFieldtype + "(" + cLength + cPrecision + ")" + cFieldcNUll + ","
        nCount = nCount + 1
    DIMENSION cFieldCaption(nCount) && 动态定义一个数组，存储各字段的文字标题
    cFieldCaption(nCount) = cCaption
    SKIP
  ENDDO
  IF !EOF()
      SKIP
  ENDIF
  TableStruInfo = LEFT(TableStruInfo, LEN(TableStruInfo) – 1)
  CNewTableStruRule = " D:\ Oil\ TableStru\" + cNewTableName + "(" + TableStruInfo + ")"
  CREATE TABLE & CNewTableStruRule && 使用模型中的数据生成新表
nNumber = AFIELDS(aStru) && nNumber 每个表的字段数
FOR t = 1 to nNumber && Caption (标题)的自动提取
SetValue = """ + cNewTableName + "." + aStru (t, 1) + """ + ", 'FIELD', 'Caption','" + cFieldCaption(t) + "'"
      DBSETPROP(& SetValue)
  ENDFOR
ENDSCAN
ENDPRO
```

收稿日期 2000 年 11 月 20 日

利用 ADO 的 Data shape 访问数据库

谢立冬

摘 要 Data shape 是开发人员访问数据库的强有力的工具。本文通过对 TreeView 控件的填充，介绍了 ADO 的 Data shape 的结构、基本语法，并与现有方法进行了比较。

主题词 ADO Data shape，记录集，TreeView 控件

Data Shape 是在 ADO 2.0 中开始出现的，自从 ADO 2.0 推出以来，Data shape 在数据访问中就一直占据很大的地位。它和数据源间的连接简单，不需要为多个记录集间的逻辑关系编写代码，也不需要复杂的过滤器，因为它本身就是一个过滤器。它可以有效地防止无用数据在网上的流动，并为 Count 之类的集合函数提供了更多的灵活性。

本文通过创建两个简单的父/子关系的记录集来介绍 ADO 的 Data shape。

Data shape 的结构

Data shape 是一种父/子关系（或称为层次关系）的数据模型：在两个记录集（本文提到的记录集是指 ADO 中的 Recordset 对象）之间建立一个一对多或多对多的关系，并为记录集提供分组和集合函数等功能。例如，你可以在客户记录集和订单记录集之间建立关系，这样，对于客户记录集中的每一个客户，订单记录集记录了与之相对应的一系列订单。

这种层次关系有点类似于 TreeView 控件，记录集相当于 TreeView 控件中一层具有相同父节点的所有节点的集合。

我们有什么理由需要使用 Data Shape

考虑这样一种问题，假如你想了解某一个客户到底有多少订单，以发现有价值的客户。我们使用 Microsoft Access 自带的 Northwind 数据库，为了解决这个问题，我们需要利用两个表，Orders 表和 Customers 表。Customers 表中存储了客户的详细信息，Orders 表中存储了所有客户的详细订单情况。在传统的方法中，我们需要使用 JOIN 或两个独立的记录集，而利用 Data shape，我们只需要设置 Customers 表为父表，Orders 表为子表即可。显然，利用 Data shape 要相对简单得多。

另外，可以利用它来替换 SQL 字句中的 GROUP BY 字句。使用 GROUP BY 字句，记录集只能获得统计数据，而不能获得详细数据，对于层次关系的记录集，你能在父记录集中获得统计数据，同时在子记录集中仍然维持着详细数据。

基本语法

Data shape 使用标准的 SQL 语句以及三个主要的关键字：SHAPE，APPEND 和 RELATEl 来创建层次关系的记录集。SHAPE 用来定义父记录集，它的语法形式如下：

```
SHAPE {Parent – SQL – Statement} [AS Parent – Record-Set – Alias]
```

可用下面的 SQL 语句从 Customers 表中提取客户信息作为父记录集：

```
SHAPE {SELECT CustomerID, CompanyName, ContactName FROM Customers}
```

APPEND 子句用于定义并附加一个子记录集在父记录集上。APPEND 的语法和 SHAPE 类似，为了定义并附加订单记录集在客户记录集上，增加如下代码：

```
APPEND ({SELECT OrderID, shipname FROM orders WHERE shipname = ″″} AS oRSOrder
```

RELATE 字句标识子对象和父对象之间的关系，它的语法如下：

```
RELATE parent - column to child - column) [AS chapter - alias]
```

因此，完整的 SQL 语句应该是：

```
SHAPE {SELECT CustomerID, CompanyName, Contact-Name FROM Customers}
APPEND ({SELECT OrderID, shipname, CustomerID
FROM orders WHERE shipname = ″″}
AS oRSOrder
RELATE CustomerID To CustomerID)
```

ADO 提供了三种类型的 Data shape，刚才我们讨论的是一种基于关系的 Data shape，另外，还有基于参数和基于分组的 data shape，基于参数的 data shape 与基于关系的 Data shape 基本相同，不同点在于前者只有在应用程序需要数据的时候才从数据库中提取数据，而后者一开始就提取了所有需要的数据。

基于分组的 data shape 提供了诸如 SQL 语句中的 Count，sum，avg 的集合功能，父记录集包含了一些分组和集合信息，而子记录集包含了详细的信息。它的语法和基于关系的 Data shape 基本相同，但有一些重要区别，看如下 SHAPE 语句：

```
SHAPE {SELECT OrderID, CustomerID, Freight FROM Orders} As rs1
COMPUTE SUM(rs1.Freight) As SumFreight, rs1 By CustomerID
```

COMPUTE 关键字产生或定义一个父记录集，SHAPE 定义了子记录集。父记录集可以包括子记录集（如 rs1）、计算字段（如 SUM rs1.Freight ）及子记录集中的任何字段（如 Cus-

tomerID）。BY 关键字类似于 SQL 语句中的 GROUP BY，指定了按子记录集中的哪些字段分组。

应用实例：在 TreeView 控件中列出客户和他的订单信息

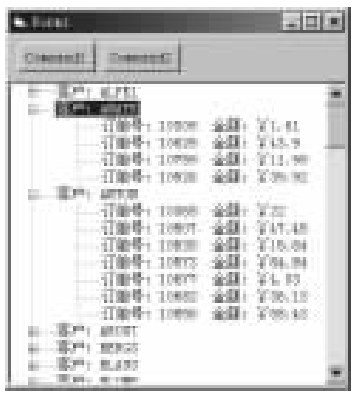
在这里，我们利用 MS ACESS 自带的 Northwind 数据库填充 Windows 的 TreeView 控件。建立一个标准的 VB 工程，从菜单上选择工程|组件并添加 Mircsoft Windows Common Controls 组件，点击 OK 就可以在 IDE 的工具箱中看到 TreeView 控件。在工程的缺省窗体中添加一个 TreeView 控件，并设置它的 LineStyle 属性为 1 - tvwRootLines，再添加两个 CommandButton 控件。下一步，我们选择菜单条上的工程|引用添加对 Microsoft ActiveX Data Object 2. 0 Libray 或更高版本 的引用，单击 OK 关闭对话框。在 Command1 按钮的 Click 事件中添加如下代码：

```
Private Sub Command1_Click()
    Dim oConn As ADODB. Connection
    Dim oRS As ADODB. Recordset
    Dim oRSChild As ADODB. Recordset
    Dim oparnode As Node
    TreeView1. Nodes. Clear
     Set oConn = CreateObject(″ADODB. Connection″)
    Set oRS = CreateObject(″ADODB. Recordset″)
    oConn. Provider = ″MSDataShape″
     oConn. Open ″Data Provider = Microsoft. jet. oledb. 3. 51;
Data Source = Nwind. mdb; ″
     oRS. Open ″SHAPE {SELECT CustomerID, Company-
Name, ContactName FROM Customers} ″ & ″APPEND
({SELECT OrderID, Freight, CustomerID FROM orders} AS
orsOrder ″& ″RELATE CustomerID To CustomerID)″, oConn
        Do While Not oRS. EOF
         Set oparnode = TreeView1. Nodes. Add(Text: = ″客
户: ″ & oRS. Fields(″CustomerID″))
        Set oRSChild = oRS. Fields(″orsOrder″). Value
        Do While Not oRSChild. EOF
           TreeView1. Nodes. Add relative: = oparnode. Index,
relationship: = tvwChild, Text: = ″订单号: ″ &
oRSChild. Fields(″OrderID″) & ″金额: ¥″ &
oRSChild. Fields(″Freight″)
            oRSChild. MoveNext
        Loop
        oRS. MoveNext
    Loop
    oRS. Close
    oConn. Close
    Set oRS = Nothing
    Set oConn = Nothing
End Sub
```

这些代码用基于关系的 Data shape 在 TreeView 控件列出了客户信息和他的订单信息。

为了在客户信息中显示统计信息，如显示他的所有订单的金额总和，在 “Command2 ”按钮的 Click 事件中添加同样的代码，其中阴影部分改为：

```
oRS. Open ″SHAPE {SELECT OrderID, CustomerID, Freight
FROM Orders} As rs1 ″ & ″COMPUTE SUM(rs1. Freight)
As SumFreight, rs1 By CustomerID″, oConn
    Do While Not oRS. EOF
    Set oparnode = TreeView1. Nodes. Add(Text: = ″客户: ″
& oRS. Fields(″CustomerID″) & ″ 总金额: ″ & oRS. Fields
(″SumFreight″))
    Set oRSChild = oRS. Fields(″rs1″). Value
    Do While Not oRSChild. EOF
 TreeView1. Nodes. Add relative: = oparnode. Index, relation-
ship: = tvwChild, Text: = ″订单号: ″ & oRSChild. Fields(″Or-
derID″) & ″ 金额: ¥″ & oRSChild. Fields(″Freight″)
        oRSChild. MoveNext
    Loop
    oRS. MoveNext
Loop
```



按[F5]运行，分别点击 “Command1 ”和 “Command2 ”按钮，结果如图。

为了利用 ADO 的 data shape 特性，必须设置 Conneect 对象的 Provider 属性为 MSDataShape，如下所示

```
oConn. Provider = “MSDataShape ”
```

该行代码告诉 ADO 去使用 Data Shape 引擎解释 SQL 语句。在引用子记录集时，必须明确引用域的值：

```
Set oRSChild = oRS. Fields(″oRSOrder″). Value
```

参考文献

Microsoft MSDN Online

附：源程序 略），在 VB6. 0 下运行通过。

收稿日期：2001 年 1 月 11 日）

Windows 下的 Visual Basic 通讯

肖慎平

摘 要 本文简要介绍了 PC 机 RS - 232 串行通讯的基本知识、方法与技巧，并通过一个实例说明如何实现通信。

关键字 RS - 232 串行通讯，UNIX，WinAPI32，Visual Basic

一、 引言

Visual Basic 以其简单易学、面向对象程序设计的支持，倍受广大计算机开发者的青睐，已广泛地应用于各个领域，它在实时监测系统中串行端口通信又是一项基础功能。采用 Visual Basic 开发串行通信程序 一般有两种方法：一是利用 Windows 的通信 API 函数；另一种是采用 VB 标准控件 Mscomm 来实现。本人曾使用 VB6 采用标准控件 Mscomm 编写过一个实时程控电话交换机话单自动采集系统，并将采集数据处理后同时发送给 UNIX 小型机和两台打印机的软件项目。本文就 Windows 98 环境中采用 VB6 设计串行端口通信程序进行较为详细的讨论且给予实例。

1. 串口通信基本知识

一般说来，计算机都有一个或多个串行端口，它们依次为 Com1、Com2 等等，这些串口还提供了外部设备与 PC 进行数据传输和通信的通道，这些串口在 CPU 和外部设备之间充当解释器的角色。当字符数据从 CPU 发送给外部设备时，这些字符数据将被转换成串行比特流数据；当接收数据时，比特流数据被转换为字符数据传递给 CPU。

在 Windows 操作系统下，是运用通信驱动程序 Comm. DRV 调用 API 函数发送和接收数据。当用通信控件或声明调用 API 函数时，它们由 Comm. DRV 解释并传递给设备驱动程序。若要编写通信程序，则只需知道通信控件提供给 Windows 通信 API 函数的接口，即只需设定和监视通信控件的属性和事件。

2. Windows API 方法

Windows API 主要提供了三个动态连接库 Kernel32. DLL、User32. DLL、Gdi. DLL 供开发人员调用，其中 Kernel32. DLL 主要包括一些底层操作函数，完成一些资源管理、任务、内存等操作；User32. DLL 包含了一些与 Windows 管理有关的函数，如通讯、菜单、消息、光标、插入符、计时器以及绝大多数非显示函数；Gdi. DLL 图形设备接口库，主要内容为与设备输出有关的函数。

与串口通讯有关联的函数 BuilidCommDCB 向 DCB 中传送设备定义字符串 、ClearCommBreak 恢复字符传输)、ClearCommError 允许出错后进行通信)、CommConfigDialog 通信端口设置会话）、EscapeCommFunction 发送扩展 COMM 函数）、GetCommConfig 返回通信端口设置）、GetCommMask 返回 COMM 事件屏蔽）、GetCommModemStatus 返回调制解调器控制登录值）、GetCommProperties 返回 COMM 设备属性）、GetCommState 返回 COMM 设备控制块）、GetCommTimeouts 返回 COMM 设备超时特性值）、SetCommBreak 挂起字符传送）、SetCommConfig 通信端口设置）、SetCommMask 设置通信事件屏蔽）、SetCommState 设置通信设备状态）、SetCommTimeouts 设置通信读写时间范围）、TransmitCommChar 向传输队列中加入字符）等均在 Windows 目录下的 System 子目录下的 Kernel32. DLL 动态连接库中。

利用 Win API 编写串口通信程序相对来说比较复杂，需要掌握一定的通信知识，其优点是实现功能上可作得更好、应用面更广泛，更适合于编写较为复杂的底层次通信程序等。

3. VB Mscomm 方法

Mscomm 控件全面地提供了使用 RS - 232 串行通讯上层开发的所有细则。Mscomm 控件具有两种处理方式 ①事件驱动方式 由 Mscomm 控件的 OnComm 事件捕获并处理通信错误及事件。②查询方式 通过检查 CommEvent 属性的值来判断事件和错误。Mscomm 控件的通信功能实现实际上是调用了 API 函数，而 API 函数是由 Comm. DRV 解释并传给设备驱动程序执行的，对于 VB 程序开发者只需知道 Mscomm 控件的属性和事件的用法即可以实现串口的操作。有关 Mscomm 控件的主要属性和方法在这里不作详细介绍。

二、 程序源码样例

本人的项目是：一台 PC 机实时采集程控电话交换机的话单，处理完毕同时将话单数据帧实时发送给 UNIX 小型机，并实时将话单送到一台本地打印机和一台远程打印机上打印。

工作原理：PC 机首先从通广北电交换机上采集计费数据，通过分析处理后送到 EECO System 的 UNIX 小型机上，并同时在本地打印机和远程打印机同时实时打印。

通信特点：

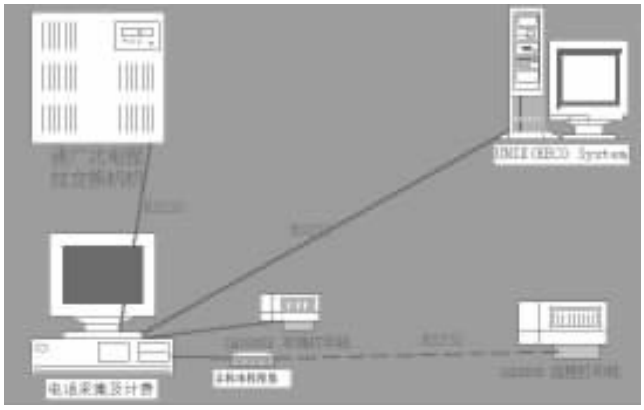
1 PC 机与交换机的底层通信协议：波特率 1200、无奇偶校验、7 个数据位、1 个停止位；PC 机与 EECO System 的底

层通信协议：波特率 1200、无奇偶校验、8 个数据位、1 个停止位；

2　PC 机向 EECO System 发送联机数据帧格式为：chr 42 + chr 64 + chr 37 + chr 13 的字符数据，3 秒钟后 PC 机接收联机应答数据帧，若第一字符数据必为 ACK 信号 chr 6 ，表示联机成功，否则重来；话单数据帧格式为：Chr 2 + Space 2 + ext_id 4Bit + space 1 + csn 6Bit + Space 1 + sec 10Bit + space 1 + Type 1Bit + space 1 + called_id 24Bit + space 1 + start_time 8Bit + space 1 + charge 8Bit + chr 13 ；

3　PC 机与对远程并行打印机的通信，通过一个并行口转串行口 PS 转换器将数据传输给远程 Epson LQ300K 打印机。

其拓朴结构图如下：



由于该项目的软件源代码较长，本人只拿出与串口通讯和并口通讯有关的程序片段来供大家参考。

1. 系统初始设置代码

……前段代码省略

```
´COM1 通讯端口，接收通广北电 Option 11 系列 PABX 传送的数据
MSCommReceive. CommPort = 1　　´通信端口 1
MSCommReceive. Settings = ″1200, N, 7, 1″　´设置控件端口通讯: 1200, N, 7, 1
MSCommReceive. InputLen = 0　　´清除从接收缓冲区读取的字符数
If Not MSCommReceive. PortOpen Then MSCommReceive. PortOpen = True ´打开通讯端口
´COM2 通讯端口，向 UNIX 小型机，香港 EECO System 传送数据
MSCommSend. CommPort = 2　　´通信端口 2
MSCommSend. Settings = ″1200, N, 8, 1″　´设置控件端口通讯: 1200, N, 8, 1
MSCommSend. InputLen = 0 ´清除从接收缓冲区读取的字符数
If Not MSCommSend. PortOpen Then MSCommSend. PortOpen = True ´打开通讯端口
´EECO System 联机通信参数 = = = => * @ % +chr(13)
MSCommSend. Output = Chr(42) & Chr(64) & Chr(37) & Chr(13) ´初始化 EECO System
MPP_Delay (3)　　´延时 3 秒钟
tmp = MSCommSend. Input
If InStr(1, tmp, Chr(6)) = 0 Then ´检查是否 UNIX 系统应答，是否收到 ACK 信号
Beep
MsgBox ″EECO System 未应答，请先重新登录 EECO System 后，并启动程控电话采集及计费系统！″, vbCritical
End
End If
```

……后段代码省略……

2. 系统接收及其分析处理代码

```
……前段代码省略……
Const Min_MsgLine_Len = 78　　´最短话单长度
Const Max_MsgLine_Len = 104　　´最长话单长度
´普通 N 话单一行信息结束标志：一个回车，一个换行，加六个 chr(0)
bill_flag = Chr(13) & Chr(10) & Chr(0) & Chr(0) & Chr(0) & Chr(0) & Chr(0) & Chr(0)
While MSCommReceive. InBufferCount >= Min_MsgLine_Len - Len(MVS_Receive)-
MSCommReceive. InputLen = MSCommReceive. InBufferCount ´接收缓冲区中等待的字符数
inp = MSCommReceive. Input　　´删除接收缓冲区中的数据流
If IsNull(inp) Or Trim(inp) = ″″ Then Exit Sub ´如未读到信息，则退出
MVS_Receive = MVS_Receive & inp　´将读到的信息拼接到内存缓冲区
i = InStr(1, MVS_Receive, ″N″ & Chr(32)) ´查找话单头位置 (N +chr(32))
If i = 0 Then MVS_Receive = ″″: Exit Sub ´如无话单头，则无用信息甩掉，并返回
MVS_Receive = Mid$(MVS_Receive, i) ´截取话单
i = InStr(1, MVS_Receive, bill_flag)　´查找话单尾位置
If i = 0 Then　　´有话单头，无话单尾
If Len(MVS_Receive) >= Max_MsgLine_Len Then MVS_Receive = Mid$(MVS_Receive, i + Len(bill_flag)) ´超出最长话单长度，截取掉话单头
Exit Sub
Else　　´话单头位置小于话单尾位置
If i - 1 < Min_MsgLine_Len - 2 * Len(bill_flag) Or i - 1 > Max_MsgLine_Len - 2 * Len(bill_flag) Then ´话单长度超出正常范围，废弃该话单
MVS_Receive = Mid$(MVS_Receive, i + Len(bill_flag))
Exit Sub
Else　　´正常话单
p_table = Mid $ (MVS_Receive, 1, i - 1) ´p_table 不含话单头尾的 bill_flag
MVS_Receive = Mid$(MVS_Receive, i + Len(bill_flag))
´分割话单，判断合法行，处理话单并写库
If MFB_AnalyseFolios(p_table) Then MPP_AnalyseTable
```

```
        End If
    End If
Wend
……后段代码省略……
……前段代码省略……
Private Sub MPP_SendBills_LPT1_PRT(PVS_Bill As String)
´Function:  通过 LPT1 口将一个分析处理好的话单送到 LPT1 打印,本地打印机,此台打印机安装时要设为并行口打印机,其打印格式为普通 A4 连打纸,隔行打印
    On Error GoTo err_prt_LPT1
    Open "C: \LPT1" For Output As #1
    Print #1, PVS_Bill;                ´打印
    Print #1, Chr(13) & Chr(10)        ´控制走纸一行
    Close #1
End Sub
Private Sub MPP_SendBills_LPT2_PRT(echobill As String)
´Function:  通过 LPT2 口将一个分析处理好的话单送到 LPT2 打印,远程打印(通过 PS 转换器,),此台打印机安装时要设为并行口打印机,其打印格式为专用打印小纸条
    On Error GoTo err_LPT2_PRT
    Open "C: \LPT2" For Output As #1
    Print #1, Chr(13) & Chr(10)        ´控制走纸一行
    Print #1, Chr(13) & Chr(10)
    Print #1, echobill;                ´打印
    Print #1, Chr(13) & Chr(10)
    Print #1, Chr(13) & Chr(10)
    Print #1, Chr(13) & Chr(10)
    Print #1, Chr(13) & Chr(10)
    Print #1, Chr(13) & Chr(10)
    Print #1, Chr(13) & Chr(10)
    Close #1
End Sub
Private Sub MPP_SendBill_COM(PVU_tele As PTABLE)
´Function:  通过 COM2 口将一个分析处理好的话单送到 EECO SYSTEM
    Dim flag As String
    Dim tmp As String
    ´主叫号码
    tmp = Space(2) & Format(Trim(PVU_tele.caller), "@@@@")
    ´流水号
    tmp = tmp & Space(1) & Format((MSFGrid.Rows - 1), "@@@@@@")
    ´通话时长-秒
    tmp = tmp & Space(1) & Format(Trim(PVU_tele.tele_secs), "@@@@@@@@@@")
    ´话单类型
    If Left(Trim(PVU_tele.called), 2) = "00" Then
        flag = Chr(73)
´IDD Call--------I  国际长途电话标志
    ElseIf Left(Trim(PVU_tele.called), 1) = "0" Then
        flag = Chr(76)
´DDD Call--------L  国内长途电话标志
    Else
flag = Chr(68)     ´Local Call-----D  本地网电话标志
    End If
    tmp = tmp & Space(1) & Format(flag, "@")
    ´被叫号码
 tmp = tmp & Space(1) & Format(Trim(PVU_tele.called), "@@@@@@@@@@@@@@@@@@@@@@@@@@")
    ´通话起始时间
    flag = Format(Trim(PVU_tele.start_date), "mmdd") & Format(Trim(PVU_tele.start_time), "hhmm")
    tmp = tmp & Space(1) & Format(flag, "@@@@@@@@")
    ´总话费
 tmp = tmp & Space(1) & Format(PVU_tele.total_charge, "@@@@@@@@@@")
    ´向 UNIX 小型机发送 a telephone bill
MSCommSend.Output = Chr(2) & tmp & Chr(3)
End Sub
Public Sub MPP_Delay(l As Long)
´Function:  延时秒数
Dim s As Single
s = Timer
While Timer - s < l
Wend
End Sub
……后段代码省略
```

三、 结束语

本文提供了一个较为详细的计算机与其外围设备进行信息交互的例子，为程序员开发有关计算机与串行通信端口和并行打印端口方面的程序提供了一个很好的例子，只需稍加修改接收处理程序和屏蔽与 UNIX 系统的通信，就可应用于各种程控电话交换机话单采集和计费。

本软件项目在 AMDK6CPU、32M 内存、配设两个串行通信端口和两个并行打印端口的计算机、一台 Epson LQ 300K 打印机和一台 Epson LQ 1600K 打印机等硬件环境；Pwin98、VB6.0 英文版软件环境下运行通过。

参考文献

1. Joe campbell. 串行通信编程指南. 北京科海培训中心，1990.10

2. 苏玉北等. 计算机 32 位串行数据通信接口的设计与实现. 计算机工程与应用，1999.12 (9)

3. 孙文全、王保平. 利用 Visual Basic 开发通讯程序的方法. 电脑编程技巧与维护，2000.3 (0)

收稿日期：2000 年 11 月 9 日)

ASP 中 E- mail 的自动发送

徐洪雷

摘 要 本文主要介绍了一种利用 asp 来实现自动发送 E - mail 的方法,代码简单、可移植性好。
关键词 asp,E - mail,自动发送

在网页制作中,若能为自己的个人主页或个人网站加上 E - mail 自动发送功能,无疑是锦上添花,即使一些比较大的网站也都在使用 E - mail 自动发送,像自动发送电子贺卡、密码查询等。

本文所介绍的方法主要是利用了 IIS 所提供的 CDONTS. NewMail 对象 (对应文件是:cdonts. dll),若找不到,则可以对其进行添加,而 PWS 则不能用这种方法来实现自动发送。下面是在我的个人俱乐部 - 中国龙网俱乐部的注册网页中实现 E - mail 自动发送功能的主要源程序。

```
<%
usname =Request. Form(″usname″)
pwd =Request. Form(″pwd″)
trname =Request. Form(″trname″)
sex =Request. Form(″sex″)
email =Request. Form(″to″)
b =len (email)
a =instr (email, ″@ ″)
′寻找标记@,若缺少@,则 a =0
′若@ 位于开头处,像@ 263. net 则属于信箱输入有错误,此时 a =1
′若@ 位于末尾,像 haha@,也属于信箱有问题,此时 a =b
If a>1 and a <b and usname <>″″ and pwd <>″″ and trname <>″″ and sex <>″″ Then
  ′所有项目都要填写,此处可随意更改
  . . . . . ′此处省略了对数据库的操作
  Set mail = Server. CreateObject( ″CDONTS. NewMail″ )
  mail. To = email
  mail. From = ″yangtianxiao75@ sina. com″ ′输入发件人信箱,此处用笔者信箱代替
  mail. Subject = ″注册回复″ ′E – mail 的主题
  body =trname& ″,您好:″& vbcrlf ′body 是信件内容
  body =body& ″您在中国龙网俱乐部的用户名是″& usname
  body =body& ″,密码是″& pwd& ″!″& vbcrlf
  mail. Body = body& ″会长 yangtianxiao 携全体会员期待着您再次光临. ″
  mail. attachfile Server. MapPath( ″introduction. htm″) ′此处是发送附件,属宣传性质
  mail. Send
Else
%>
<HTML>
<BODY>
<H3><center>欢迎加入中国龙网俱乐部 <HR><center></H3>
<FORM action =″Mail. asp″ method =″POST″> ′此处 action 的设置很关键
<TABLE Border =1 align =center>
  <TR><TD bgcolor =″#FFCC00″>用& nbsp; 户& nbsp; 名</TD> ′bgcolor 设置背景颜色
  <TD><INPUT type =″ text″ name =usname></TD></TR>
  <TR><TD bgcolor =″#FFCC00″>密& nbsp; & nbsp; & nbsp; & nbsp; 码 </TD>
  <TD><INPUT type =″text″ name =pwd></TD></TR>
  <TR><TD bgcolor =″#FFCC00″>真实姓名 </TD>
  <TD><INPUT type =″ text″ name =trname></TD></TR>
  <TR><TD bgcolor =″#FFCC00″>性& nbsp; & nbsp; & nbsp; & nbsp; 别 </TD>
  <TD><INPUT type =″text″ name =sex></TD></TR>
  <tr><td align =center bgcolor =″ # FFCC00″>E – mail </td>
  <td><INPUT type =″text″ name =to></td></tr>
</TABLE><p align =center>
<INPUT type =″submit″ value =″提交″ name =submit1>  & nbsp; & nbsp; & nbsp;
<INPUT type =″reset″ value =″取消″ name =reset1></p>
</FORM>
</BODY>
</HTML>
<%
End If
%>
```

值得注意的是,该程序将通常情况下的注册静态网页(html 文件)和后台处理文件(asp 文件)二者合并为一个 ASP 文件了,只要在 Vid 中将其保存为 . asp 文件即可。若已有类似的注册文件,则只需将其中的自动发送部分粘贴到自己的 . asp 文件中,稍加修改就行了。

(收稿日期:2000 年 12 月 5 日)

使用 SNMP 建立对 TCP 连接的监控

冯志林

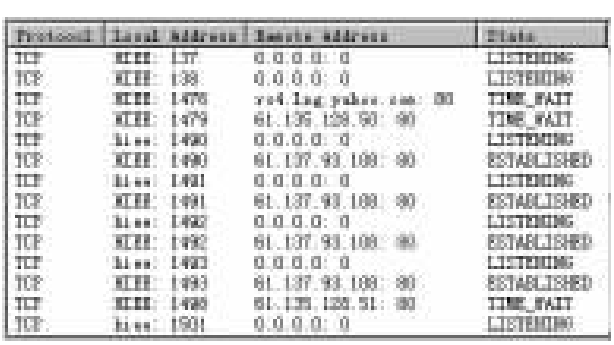
摘 要 本文介绍了利用 SNMP 接口来获取网络中的 TCP 连接状况的信息，并给出了一个监控 TCP 连接的具体实现过程。

关键词 SNMP，MIB，TCP

一、引言

在日常的网络管理中为了监测和控制网络运行的性能，需要对网络中的 TCP 连接状况进行统计，通过分析连接成功和失败的统计数据便可以判断出该网络运行的性能。为了实现这一功能，我们可以采用 SNMP Single Network Management Protocol 所提供的接口来获取 TCP 连接的信息。SNMP 网络管理协议是基于 TCP/IP 的 Internet 网的一个标准的网络管理协议，目前这种网络管理协议应用很广，几乎所有的网络厂商推出的网络管理系统都支持 SNMP 协议。SNMP 中的管理信息库 MIB 是全部被管理对象的结构集合，MIB 的定义语法采用了 ASN.1 语法的一个子集，是一种树状结构的定义，通过定义使得所有被管理对象都有它自己的对象标识符 OID。如果想要获得某设备的信息，只需提供该设备的 OID，然后通过 SNMP 提供的接口便可获得该设备的状态信息。

在 Windows 98/NT/2000 中，INETMIB1.DLL 库提供了 98/NT/2000 环境下的 SNMP 接口，这个接口通过 IOCTL 命令调用 TCP/IP 的核心驱动程序 TCP.SYS，从而获得 TCP 连接的状态信息。因此，我们只需提供 TCP 的 OID 值，然后调用 SNMP 接口函数，就可以访问 MIB 中 TCP 的信息。MIB 为 TCP 定义的 OID 为 1.3.6.1.2.1.6.13.1.1，这样根据 TCP 的 OID，通过 SNMP 提供的查询接口便可获得关于 TCP 连接的信息，从而实现对 TCP 连接的监控。在下图中显示了对主机 HIEE 所进行的 TCP 连接状态的监控。



Protocol	Local Address	Remote Address	State
TCP	HIEE: 137	0.0.0.0: 0	LISTENING
TCP	HIEE: 138	0.0.0.0: 0	LISTENING
TCP	HIEE: 1478	ns4.lmg.pdnx.com: 80	TIME_WAIT
TCP	HIEE: 1479	61.135.128.50: 80	TIME_WAIT
TCP	hiee: 1480	0.0.0.0: 0	LISTENING
TCP	HIEE: 1490	61.137.93.109: 80	ESTABLISHED
TCP	hiee: 1481	0.0.0.0: 0	LISTENING
TCP	HIEE: 1491	61.137.93.109: 80	ESTABLISHED
TCP	hiee: 1482	0.0.0.0: 0	LISTENING
TCP	HIEE: 1492	61.137.93.109: 80	ESTABLISHED
TCP	hiee: 1483	0.0.0.0: 0	LISTENING
TCP	HIEE: 1493	61.137.93.109: 80	ESTABLISHED
TCP	HIEE: 1498	61.135.128.51: 80	TIME_WAIT
TCP	hiee: 1501	0.0.0.0: 0	LISTENING

二、TCP 连接监控的制作过程

2.1 定义 TCP 连接监控中使用的结构和调用函数

TCP 连接监控中使用的结构和调用函数共有 7 项，其中 TcpState 包含了进行 TCP 连接时可能出现的状态，如连接的关闭、建立、超时等；TCPINFO 结构用于实现一个包含所有 TCP 连接的链表结构，该结构包含了进行 TCP 连接的本地 IP 地址、本地端口号、远程 IP 地址、远程端口号以及指向上一个和下一个 TCP 连接的指针；定义 pSnmpExtensionInit 函数指针是为了使用 SNMP 扩展库中的 SnmpExtensionInit 函数，该函数用于初始化 SNMP 扩展库；定义 pSnmpExtensionQuery 函数指针是为了使用 SNMP 扩展库中的 SnmpExtensionQuery 函数，该函数用于执行 MIB 对象状态的查询；GetPortName 用于获得端口号的文本值；GetIpHostName 用于将 IP 地址转换成名字解析型的格式，即 210.32.33.1 型的 IP 地址格式；LoadInetMibEntryPoints 用于装入 SNMP 扩展 dll，同时从 dll 中导出所需的 SNMP 接口函数。

1 定义 TCP 连接时可能出现的状态

```
static char TcpState[][32] = {
        "CLOSED",
        "LISTENING",
  "SYN_SENT",
  "SEN_RECEIVED",
  "ESTABLISHED",
  "FIN_WAIT",
  "FIN_WAIT2",
  "CLOSE_WAIT",
  "CLOSING",
  "LAST_ACK",
  "TIME_WAIT"
};
```

2 定义 TCP 连接链表结构

```
typedef struct _tcpinfo {
struct _tcpinfo    * prev;        //指向上一个 TCP 连接
struct _tcpinfo    * next;        //指向下一个 TCP 连接
UINT               state;         //连接状态
UINT               localip;       //TCP 连接的本地 IP 地址
UINT               localport;     //TCP 连接的本地端口号
UINT               remoteip;      //TCP 连接的远程 IP 地址
UINT               remoteport;    //TCP 连接的远程端口号
} TCPINFO, * PTCPINFO;
```

3 定义获取 SNMP 扩展库中 SnmpExtensionInit 函数的指针 pSnmpExtensionInit：

```
BOOL (_stdcall * pSnmpExtensionInit) (IN DWORD dwTimeZeroReference, OUT HANDLE * hPollForTrapEvent,
```

```
OUT AsnObjectIdentifier * supportedView);
```

4 定义获取 SNMP 扩展库中 SnmpExtensionQuery 函数的指针 pSnmpExtensionQuery

```
BOOL (_stdcall * pSnmpExtensionQuery)(IN BYTE requestType, IN OUT RFC1157VarBindList * variableBindings, OUT AsnInteger * errorStatus, OUT AsnInteger * errorIndex);
```

5 定义获取端口号文本值的函数

```
char * GetPortName( UINT port, char * proto, char * name, int namelen )
{
        struct servent * psrvent;
if( psrvent = getservbyport( htons( (USHORT) port ), proto )) {
            strcpy( name, psrvent -> s_name );
    } else sprintf(name, "%d", port);
        return name;
}
```

6 定义转换 IP 地址显示格式的函数 用于获得主机名

```
char * GetIpHostName( BOOL local, UINT ipaddr, char * name, int namelen )
{
    struct hostent          * phostent;
    UINT                    nipaddr;
    nipaddr = htonl( ipaddr ); // 将主机的无符号长整形数转换成网络字节顺序
    if( !ipaddr ) {
        if( !local ) {
            sprintf( name, "%d. %d. %d. %d",
                (nipaddr >> 24) & 0xFF,
                (nipaddr >> 16) & 0xFF,
                (nipaddr >> 8) & 0xFF,
                (nipaddr) & 0xFF);
        } else {
gethostname(name, namelen); //获取目前使用者使用的主机的名称
        }
    } else if( ipaddr == 0x0100007f ) {
        if( local ) {
            gethostname(name, namelen);
        } else {
            strcpy( name, "localhost" );
        }
} else if( phostent = gethostbyaddr( (char * ) & ipaddr,
        sizeof( nipaddr ), PF_INET )) {
        strcpy( name, phostent -> h_name );
    } else {
        sprintf( name, "%d. %d. %d. %d",
            (nipaddr >> 24) & 0xFF,
            (nipaddr >> 16) & 0xFF,
            (nipaddr >> 8) & 0xFF,
            (nipaddr) & 0xFF);
    }
    return name;
}
```

7 定义装入 SNMP 扩展库，同时导出所需的 SNMP 接口函数的 LoadInetMibEntryPoints 函数

```
BOOLEAN LoadInetMibEntryPoints()
{
  HINSTANCE    hInetLib;
  if( !(hInetLib = LoadLibrary( "inetmib1. dll" ))) return FALSE;
  if( !(pSnmpExtensionInit = (void * ) GetProcAddress( hInetLib,
            "SnmpExtensionInit" )) ) return FALSE;
  if( !(pSnmpExtensionQuery = (void * ) GetProcAddress( hInetLib,
            "SnmpExtensionQuery" )) )    return FALSE;
  return TRUE;
}
```

2. 2 定义 TCP 连接监控程序的入口点 main 函数

```
#include "windows. h"
#include "stdio. h"
#include "snmp. h"
#include "winsock. h"
#define HOSTNAMELEN 256      // 定义主机名的最大长度
#define PORTNAMELEN 256// 定义端口的最大长度
#define ADDRESSLEN HOSTNAMELEN + PORTNAMELEN
// 定义地址的最大长度
TCPINFO          TcpInfoTable; // 定义 TCP 连接列表
void main( int argc, char * argv[] )
{
HANDLE    hTrapEvent;  // 定义自陷事件句
AsnObjectIdentifier  hIdentifier; // 定义 ASN. 1 的对象标识符
RFC1157VarBindList   bindList; // 定义待查询的绑定列表
RFC1157VarBind  bindEntry; // 定义待查询的绑定
  UINT      tcpidentifiers[ ] = { 1, 3, 6, 1, 2, 1, 6, 13, 1, 1};
// SNMP 中 TCP 的 OID 值
AsnInteger  errorStatus, errorIndex;   // 定义出错状态值
TCPINFO   * currentEntry, * newEntry;   // 定义 TCP 连接链表指针
UINT   currentIndex;          // 定义 TCP 连接列表的操作号
WORD wVersionRequested;       // 定义 winsock 的版本号
WSADATA   wsaData;        // 定义 WSADATA 结构
        char    localname[HOSTNAMELEN], remotename[HOSTNAMELEN];
char remoteport[PORTNAMELEN], localport[PORTNAMELEN];
char localaddr[ADDRESSLEN], remoteaddr[ADDRESSLEN];
  // 初始化 winsock
  wVersionRequested = MAKEWORD( 1, 1 );
  if( WSAStartup( wVersionRequested, & wsaData ) ) {
        printf("Could not initialize Winsock. \n");
        return 1;
  }
  // 装入 inetmib1. dll 并导出所需的 SNMP 接口函数
  if( !LoadInetMibEntryPoints()) {
        printf("Could not load extension DLL. \n");
        return 1;
  }
  // 初始化 SNMP 扩展库
  if(    !    pSnmpExtensionInit(    GetCurrentTime(    ), & hTrapEvent, & hIdentifier )) {
```

```
        printf(″Could not initialize extension DLL. \n″);
        return 1;
    }
    // 初始化 SNMP 查询结构
    bindEntry. name. idLength = 0xA;
    bindEntry. name. ids = tcpidentifiers;
    bindList. list = & bindEntry;
    bindList. len = 1;
    // 设置 TCP 连接列表的头节点,形成双链表
    TcpInfoTable. prev = & TcpInfoTable;;
    TcpInfoTable. next = & TcpInfoTable
currentIndex = 1;          // 设置 TCP 连接列表的操作号
currentEntry = & TcpInfoTable;      // 将 currentEntry 指向链表的开始处
    // 开始枚举 TCP 连接
    while(1) {
// 设置 SNMP 接口函数 SnmpExtensionQuery 的请求类型为 ASN_RFC1157_GETNEXTREQUEST,从而// 允许 SNMP 获取下一个请求,实现全部 TCP 连接的枚举
        pSnmpExtensionQuery(
ASN_RFC1157_GETNEXTREQUEST, & bindList, & errorStatus,
& errorIndex );
    // 当列举完所有的 TCP 连接后,退出循环
    if( bindEntry. name. idLength < 0xA ) break;
    // 如果读到的是 SNMP 中下一个请求对象的信息,则回到 TCP 连接链表的开
// 始处,并设置 TCP 连接链表的操作号
if( currentIndex ! = bindEntry. name. ids[9] ) {
currentEntry = TcpInfoTable. next;          // 将 currentEntry 指向 TCP 连接链表的开始处
currentIndex = bindEntry. name. ids[9];  // 设置 TCP 连接链表的操作号
        }
// 根据 TCP 连接链表的操作号,对该链表进行相应的操作
    switch (currentIndex ) {
    case 1:
// 当 TCP 连接链表的操作号为 1 时,为每一个 TCP 连接分配一个 TCPINFO 链表节点,通过不断// 的枚举从而形成包含所有 TCP 连接的 TCP 连接链表
newEntry = (TCPINFO * ) malloc( sizeof(TCPINFO ));
newEntry ->prev = currentEntry;
newEntry ->next = & TcpInfoTable;
currentEntry ->next = newEntry;
currentEntry = newEntry;
currentEntry ->state = bindEntry. value. asnValue. number;
            break;
        case 2:
// 当 TCP 连接链表的操作号为 2 时,设置每一个 TCP 连接链表节点的 localip 值
currentEntry ->localip = * (UINT * )bindEntry. value. asnValue. address. stream;
            currentEntry = currentEntry ->next;
            break;
        case 3:
// 当 TCP 连接链表的操作号为 3 时,设置每一个 TCP 连接链表节点的 localport 值
currentEntry ->localport = bindEntry. value. asnValue. number;
            currentEntry = currentEntry ->next;
            break;
        case 4:
  // 当 TCP 连接链表的操作号为 4 时,设置每一个 TCP 连接链表节点的 remoteip 值
   currentEntry ->remoteip = * (UINT * ) bindEntry. value. asnValue. address. stream;
            currentEntry = currentEntry ->next;
            break;
        case 5:
//当 TCP 连接链表的操作号为 5 时,设置每一个 TCP 连接链表节点的 remoteport 值
currentEntry ->remoteport = bindEntry. value. asnValue. number;
            currentEntry = currentEntry ->next;
            break;
        }
  }
  // 输出 TCP 连接链表信息
  printf( ″%7s % -30s % -30s %s\n″, ″Proto″, ″Local″, ″Remote″, ″State″ );
  currentEntry = TcpInfoTable. next;   // 将 currentEntry 指向 TCP 连接链表的第一个连接节点
  while( currentEntry ! = & TcpInfoTable ) {
        sprintf( localaddr, ″%s: %s″,
  GetIpHostName( TRUE, currentEntry ->localip, localname, HOSTNAMELEN),
  GetPortName( currentEntry ->localport, ″tcp″, localport, PORTNAMELEN ));
        sprintf( remoteaddr, ″%s: %s″,
  GetIpHostName( FALSE, currentEntry ->remoteip, remotename, HOSTNAMELEN),
  currentEntry ->remoteip ? GetPortName( currentEntry ->remoteport, ″tcp″, remoteport,
PORTNAMELEN ): ″0″ );
        printf(″%7s % -30s % -30s %s\n″, ″TCP″, localaddr, remoteaddr,
            TcpState[currentEntry ->state]);
  currentEntry = currentEntry ->next; // 将 currentEntry 指向 TCP 连接链表中的下一个节点
  }
}
```

当然,限于篇幅仅提供控制台程序代码,如果读者感兴趣,也可以编写 GUI 界面的 TCP 连接监控程序,这样一个属于你自己的 TCP 连接监控程序便轻松实现了。

参考文献

1. David Bennett 等著 徐军等译. Visual C + + 5 开发人员指南. 北京 机械工业出版社 1999

2. David Zeltserman 著 潇湘工作室译. SNMPv3 与网络管理. 北京 人民邮电出版社 2000

收稿日期:2000 年 12 月 26 日)

用内存设备上下文实现图像选中区域的判断

童登金

摘　要　在交互式图像处理中，常常要通过鼠标对图像区域进行选取，然后对选中的区域进行处理，借用内存设备上下文可以把对选中区的判断过程交给 MFC 去完成。

关键词　显示设备上下文，内存设备上下文

一、 引言

设备上下文（DC）是一种 Windows 数据结构，它包含了比如显示设备和打印机绘图属性信息的描述。Windows 中所有的绘图功能都是通过 DC 对象即 CDC 类实现的，CDC 类封装了一些 API 函数来实现绘图功能。设备上下文实现了 Windows 中设备无关性的图形，可用来对显示器、打印机及内存元文件进行绘图。在内存中实现的设备上下文，就是内存设备上下文。

在交互式图像处理中，常常要通过鼠标对图像区域进行选取，然后对选中的区域进行处理。通常鼠标选取的区域很不规则，选中区的判断比较复杂。而借用内存设备上下文可以把判断过程交给 MFC 去做。

二、 如何实现内存上下文及交互的实现原理

使用内存设备上下文来处理图像，首先要在内存中建立一个和当前设备上下文相匹配的设备上下文，然后用所建的内存设备上下文选择位图加载。之后，对图像的处理就可以在此内存设备上下文上进行了。下面的代码演示如何使用 VC 的 MFC 基本类库来实现内存设备上下文。

```
CDC * CImageProcView: : PaintBegin(CDC *pDC)
{
    m_pMemDC =new CDC;
    m_pMemDC ->CreateCompatibleDC(pDC); // pDC 为当前设备上下文的一个指针
    if(hdib = =NULL)
        return NULL;
    LPBYTE lpDIB =(LPBYTE)GlobalLock(hdib); //hdib 为所用位图的句柄
    if(lpDIB = =NULL)
        return NULL;
    GlobalUnlock(hdib);
    m_hBitmap =DIBToDIBSection(lpDIB);
    if(m_hBitmap = =NULL)
        return NULL;
    m_pBitmap =new CBitmap;
    m_pBitmap ->Attach(m_hBitmap);  //m_hBitmap 为所用位图的 HBITMAP 结构
    m_pPaletteTmp =m_pMemDC ->SelectPalette
(pPalette, TRUE);
    m_pMemDC ->RealizePalette();
    //在内存设备上下文实现所用位图的调色板
    m_pBitmapTmp =(CBitmap *)m_pMemDC ->SelectObject
(m_pBitmap);
    //在内存设备上下文中加载位图
    return m_pMemDC;
}
```

使用完内存设备上下文后要把它释放掉，代码如下：

```
void CImageProcView: : PaintEnd()
{
    m_pMemDC ->SelectObject(m_pBitmapTmp);
m_pMemDC ->SelectPalette(m_pPaletteTmp, TRUE);
        delete m_pMemDC;
    ..... //其他代码
    }
```

三、用内存设备上下文实现交互式图像绘制

采用内存设备上下文可以实现对图像的交互式处理，并保存处理结果。在实现时，同时创建两个设备上下文，一个是默认设备上下文，即显示设备上下文；另一个就是和当前显示设备上下文兼容的内存设备上下文，它要选择待处理图像加载。可以先在内存设备上下文中实现对图像的交互式处理，然后用 CDC 的 StretchBlt 函数在显示设备上下文上显示。也可以在交互式操作时，两个设备上下文同时工作，显示设备上下文根据交互操作在显示设备上对图像处理并显示，内存设备上下文在内存中对图像进行修改处理；两个设备上下文保持对图像处理的一致，就实现了对图像的交互式操作。操作结束，将内存设备上下文的图像保存下来。下面是一段画线程序。

```
CClientDC dc(this); 显示设备上下文
CPen pen(PS_SOLID, 1, RGB(0, 0, 0));
CPen * pOldMemPen = pMemDC ->SelectObject( &pen); //内存设备上下文
CPen *pOldPen =dc. SelectObject(& pen);
dc. MoveTo(m_oldPoint); //保存的前一点
dc. LineTo(point);
m_oldPoint =point;              //保存当前点
dc. SelectObject(pOldPen);
//下面这段程序实现坐标的对准
CPoint refPoint = GetScrollPosition(); //本程序是 ScrollView 成员函数
```

```
CPoint newPoint = point;
newPoint.x += refPoint.x;
newPoint.y += refPoint.y;
//
pMemDC->MoveTo(m_oldDibPoint); //保存的前一点
pMemDC->LineTo(newPoint);
m_oldDibPoint = newPoint;
pMemDC->SelectObject(pOldMemPen);
```

上面的程序中提到了坐标对准的问题，在此加以说明。显示设备上下文在本程序中为 CClientDC 它的坐标区为显示窗口的客户区。而内存设备上下文载入的图像在卷屏时，内存设备上下文的坐标和显示设备上下文的坐标并不相等。它们之间需建立对应关系，这是通过用 GetScrollPosition 这个函数来实现的。GetScrollPosition 返回卷动窗口的卷动位置。用显示设备上下文加上 GetScrollPosition 得到的坐标，就得到了对应的内存设备上下文的坐标。

四、用内存设备上下文实现图像选中区域的判断

在编程时，在显示设备上下文中根据鼠标移动位置，记下坐标点集的坐标 再按此点集绘制选择区域的多边形边界以便显示观察，多边形的绘制用显示设备上下文的 MoveTo 、LineTo 和 Polyline 函数。而在内存设备上下文中先选入原图像的一个副本，建立副本的过程在此不加详述。先把图像副本中的所有像素值设为某个标志值 V1，在鼠标移动时记下坐标点集的对应坐标，然后根据此点集用内存设备上下文的 Polygon 函数绘制多边形区域，绘制前用另一个不同的标志值 V2 设置画刷工具，Polygon 函数会将选择区域中的像素点设为新的标志值 V2。在这一过程中，同样要正确建立内存设备上下文坐标和显示设备上下文坐标之间的对应关系，方法同前。

在 OnMouseMove UINT nFlags、 CPoint point 函数中，实现屏幕绘线和坐标点集的记录，代码如下：

```
void CImageProcView::OnMouseMove(UINT nFlags, CPoint point)
{
    CPoint refPoint = CImageProcView::GetScrollPosition(); //用于实现坐标的对准
    //point.x += refPoint.x;
    //point.y += refPoint.y;
    if(bSelectPoly) //如果当前操作是勾勒选择区域
    {
        if(GetCapture()!=this)
            return;
        CClientDC dc(this);
        CPen pen(PS_DOT, 1, RGB(0, 0, 0));
        CPen *pOldPen = dc.SelectObject(&pen);
        dc.MoveTo(m_oldPointS); dc.LineTo(point);
//在屏幕上绘制选择区域的多边形边界
        dc.SelectObject(pOldPen);
        m_pointS[m_nPoints].x = point.x + refPoint.x;
        m_pointS[m_nPoints].y = point.y + refPoint.y; //记下内存设备上下文中的边界点
        m_pointSL[m_nPoints].x = point.x;
        m_pointSL[m_nPoints].y = point.y; //记下显示设备上下文中的边界点
        m_oldPointS = point; //记下前一点，以便下次在屏幕上绘制边界时用
        m_nPoints++; //记下点的总数
    }
    CScrollView::OnMouseMove(nFlags, point);
}
```

在 OnLButtonUp UINT nFlags、 CPoint point 函数中，完成屏幕绘线并实现在内存上下文中对选择区域的处理，代码如下：

```
void CImageProcView::OnLButtonUp(UINT nFlags, CPoint point)
{
    CPoint refPoint = CImageProcView::GetScrollPosition();
    if(bSelectPoly){
        if(GetCapture()!=this)
            return;
        CClientDC dc(this);
        CDC *pDCS = PaintBeginS(&dc);
        CPen penS(PS_SOLID, 0, RGB(255, 255, 255));
//决定内存设备上下文中多边形边界象素的值 V3，可根据需要设值
        CBrush brushS(RGB(0, 0, 0));
// brushS 将内存设备上下文中选择区域内象素的值设为 RGB(0, 0, 0)，即前面所述的 V2;
    CBrush *pOldbrushS = pDCS->SelectObject(&brushS);
    CPen *pOldPenS = pDCS->SelectObject(&penS);
    CPen pen(PS_DOT, 1, RGB(0, 0, 0));
    CPen *pOldPen = dc.SelectObject(&pen);
    if((point.x + refPoint.x == m_pointS[0].x)&&
        (point.y + refPoint.y == m_pointS[0].y))
      //在最后一点和起始点重合的情况闭合选中区
        {;
            dc.Polyline(m_pointS, m_nPoints)
            dc.SelectObject(pOldPen);
            pDCS->Polygon(m_pointS, m_nPoints);
          //填充内存设备上下文中选择区域多边形
            pDCS->SelectObject(pOldPenS);
            pDCS->SelectObject(pOldbrushS);
            CScrollView::OnLButtonUp(nFlags, point);
            PaintEndS();
            ReleaseCapture();
            return;
        }
    //在最后一点和起始点不重合闭合选中区
        dc.MoveTo(point);
        dc.LineTo(m_pointSL[0].x, m_pointSL[0].y);
        m_pointSL[m_nPoints].x = m_pointSL[0].x;
        m_pointSL[m_nPoints].y = m_pointSL[0].y;
        dc.Polyline(m_pointSL, m_nPoints+1);
        dc.SelectObject(pOldPen);
        pDCS->Polygon(m_pointS, m_nPoints);
 //填充内存设备上下文中选择区域多边形
```

VC + +中用 OpenGL 编制 CHzGL 类实现旋转立体汉字

陈　忠

摘　要　用 OpenGL 绘制各种立体汉字给软件起到较好的包装效果，本文叙述了用自编的类 CHzGL 实现旋转立体汉字，给出全部 CHzGL 类源代码，并介绍应用 CHzGL 类编制示例程序过程，使一般用户较容易使用 OpenGL。

主题词　OpenGL，VC + +，立体汉字

一、前 言

在 Win32 环境下，一般用 DirectX 编制多媒体、游戏方便的软件，而用 OpenGL 可编制出绘制各种动态三维立体图形软件。OpenGL 是一个图形硬件的软件接口，用它可编制出逼真的在光线照射下三维立体图形，贵刊在前面几期文章中都涉及到该项内容，然而，作为一般用户，使用比较麻烦。当自己的应用程序主要功能编制完成后，经常需要包装，如在软件的启动封面、对话框中显示软件名称等文字，用旋转的立体汉字给人有耳目一新的感觉，用 OpenGL 编制是最好的选择。在贵刊 2000 第 4 期中提到该项内容。由于 MFC 未对 OpenGL 进行封装形成类，在VC + + 环境使用，要在不同的地方调用大量 API 函数，给使用者感到头绪很乱，形成不便。正是由于这个原因，笔者开发了用 OpenGL 绘制立体汉字较为简明、实用的类 CHzGL。它有如下特点：1. 面向对象编程，将烦琐的 OpenGL API 函数封装在一个类中，用户只涉及三个可调用函数，移植功能强。可将此类形成 dll 动态连接库，也可直接写入自己开发的应用程序中，分别可在对话框或窗口当中显示旋转的立体汉字，极大地方便了一般用户使用 OpenGL。2. 对 OpenGL 调用的函数作了最简化处理，代码量大大减少。3. 可同时显示两排按不同方向旋转的汉字，其中一排汉字字体颜色各不一样。适合不同要求。为方便读者使用 CHzGL 类，本文简单叙述 OpenGL 相关的 API 函数，并给出全部 CHzGL 类源代码，最后介绍应用 CHzGL 类编制示例程序过程。希望给大家的应用软件包装起到作用。

二、OpenGL 相关 API 函数

OpenGL 提供了不少 API 函数调用，笔者在开发的 CHzGL 类已作了最简化处理，由于在 CHzGL 类中未提供对话框对字体、颜色等参数的设置（因不便在自己的应用程序中调用），在实际应用中，又需改变这些参数，所以需对一些相关函数作一了解。

1. ChoosePixelFormat 函数

功能：选择与设备描述符（DC）接近的像素格式。

原型：int ChoosePixelFormat

```
HDC hdc    / / 设备描述符句柄
CONST PIXELFORMATDESCRIPTOR * ppfd
/ /最佳匹配的像素格式
```

2. SetPixelFormat 函数

功能：设置与设备描述符（DC）接近的像素格式。

原型：BOOL SetPixelFormat

```
HDC hdc    / / 设备描述符句柄
int iPixelFormat  / / 像素格式索引值
CONST PIXELFORMATDESCRIPTOR * ppfd
/ /最匹配的像素格式
```

3. wglCreateContext 函数

```
        pDCS - >SelectObject(pOldPenS);
        pDCS - >SelectObject(pOldbrushS);
        CScrollView: : OnLButtonUp(nFlags, point);
        PaintEndS();
        ReleaseCapture();
    }
}
```

以上代码中假设图像副本中已进行了 V1 值的设定。

对选中区进行判断时，对于图像中的某一点，如果它在内存设备上下文中对应点的值为 V2，则表示这一点在选择区内；为 V3 表示在区域边界上；否则表示点在选择区外。然后就可以根据需要对不同区域的点进行相应的处理。

五、结束语

上述代码在 VC + + 6. 0 上编译通过，笔者用上述方法实现了绝缘材料中电树图像的交互处理，效果良好。

（收稿日期：2000 年 12 月 12 日）

功能：创建 OpenGL 图形操作描述符 rendering context HGLRC、HGLRC 与 Win32 API 下 Handle to a device context DC - 设备描述符句柄意思相近。

原型：HGLRC wglCreateContext

HDC hdc // 设备描述符句柄

4. wglMakeCurrent 函数

功能：指定图形操作描述符。

原型：BOOL wglMakeCurrent

HDC hdc // 设备描述符句柄

HGLRC hglrc //图形操作描述符

5. glEnable 函数

功能：允许 OpenGL 一些性能，如颜色浓度、光照、纹理。

原型：void glEnable

GLenum cap

6. glClearColor 函数

功能：指定颜色为清除缓冲区一般设置显示底色。

原型：void glClearColor

GLclampf red //红色值 0 - 1 之间，为小数

GLclampf green //绿色值 0 - 1 之间，为小数

GLclampf blue // 蓝色值 0 - 1 之间，为小数

GLclampf alpha

7. wglUseFontOutlines 函数

功能：此函数为显示 3D 字体最重要函数，它建立一个显示字体轮廓列表，在使用该函数前，需选定好字体类型，所选字体必须为 TrueType（微软和 Apple 公司共同研的字型标准），需注意，字体名需是 hWINDOWS hFONTS 下所列出文件名，可参照字体属性对话框（Word、Excel 中都有字体对话框）中文字体下列表中选项。

原型：BOOL wglUseFontOutlines

HDC hdc // 所选字体的设备描述符

DWORD first // 要转换为显示列表的第一个字符

DWORD count // 转换为显示列表的字符个数

DWORD listBase // 指定的显示列表基数

FLOAT deviation // 指定的最大偏离值，一般为 0

FLOAT extrusion // 它是立体字体具体表现，Z 轴（纵深方向）方向的值，若为 0 则显示一个二维平面字体

int format // 显示的字体轮廓线格式，WGL_FONT_LINES 则轮廓线为线段 - 空心，WGL_FONT_POLYGONS 轮廓线是多边形 - 实心

LPGLYPHMETRICSFLOAT lpgmf

// 接受字符特性 字形的位置和方向 的地址

由于西文字符与汉字在计算机内编码方式是不同的，所以在调用该函数时注意 first 的值，西文字符采用单字节编码 - ASCII 码（美国信息交换标准码），如 A = 65 z = 122 当采用单字母调用时，可直接将字母的 ASCII 值赋给 first，而汉字比字母要多得多，采用双字节才能容纳常用的汉字，编码采用 GB - 2312，它是按区位码顺序，第 8 位为 1 的方式编排，如区位码第一个汉字“啊”编码为 B0A1（45217）当 first = 45217 时，则显示“啊”字，汉字的双字节在机内以区位码补码形式存放，则汉字编码计算公式为：

汉字编码 =（256 + 第一字节内码）* 256 + 256 + 第二字节内码

加上 256 得出原码，乘 256 求出编码的前八位，此计算公式可适合所有汉字编码的计算。

8. glTranslated 函数

功能：将当前矩阵转换成目标矩阵，三维图形显示，涉及到较深数学方便的知识，一般用矩阵运算实现立体坐标的转换。用此函数可确定字体显示的位置和大小。

原型：void glTranslated

GLdouble x //决定显示字体的 X 位置

GLdouble y //决定显示字体的 Y 位置

GLdouble z //决定显示字体的大小

9. glRotated 函数

功能：将当前矩阵转换成旋转矩阵，可将字体按不同方向旋转。

原型：void glRotated

GLdouble angle // 旋转角度，在 0—360 之间

GLdouble x //指定按 X 轴方向旋转

GLdouble y //指定按 Y 轴方向旋转

GLdouble z //指定按 Z 轴方向旋转

只按一个方向旋转，选取 X、Y、Z 其中之一的值大于 0，若按不同方向旋转，如各种屏幕保护程序中三维图形的任意旋转，可按 X、Y、Z 大小组合确定旋转方向。

10. glColor3dv 函数

功能：设置字体显示的颜色。

原型：void glColor3dv

const GLdouble * v

通常 v 设置成：static GLdouble v[3] = { red green blue } 它按照红、绿、蓝顺序设定颜色值。

11. glListBase 函数

功能：设置显示列表的基数。

原型：void glListBase

GLuint base

base 为列表的基数值，可任设置一整数值，它必须与 wglUseFontOutlines 中的 listBase 值一致。

三、CHzGL 类程序代码

1. CHzGL 类的 . H 文件

```
// HzGL.h: interface for the CHzGL class.
//
#if !defined(AFX_HZGL_H__FC9C6828_CEB8_11D4_9034_
F00952C10000__INCLUDED_)
#define AFX_HZGL_H__FC9C6828_CEB8_11D4_9034_
F00952C10000__
INCLUDED_
#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000
class CHzGL
{
public:
    CHzGL();
    ~CHzGL();
    bool Render() ;
    bool Create(CWnd * pWnd);
    OnResize(int cx, int cy); //窗口大小发生变化时调用
private:
        spin(); //旋转函数
    bool OnInit() ;
    CString m_theString ;
    CString m_String;
    BYTE FTextList[128];
    void Create3DFont() ;
    CDC * m_pdc;
    // 包含字形特性的位置和方向 . . .
    GLYPHMETRICSFLOAT m_agmf1[128];
    GLYPHMETRICSFLOAT m_agmf2[40] ;
    //用于 OPenGL 显示的字
    CFont fontOpenGL1, fontOpenGL2; 体
    CSize m_angle; //旋转角度 0 -360
    HGLRC m_hrc ; //OpenGL Rendering Context
    int m_iDisplayListStart1;
    int m_iDisplayListStart2;
};
#endif // !defined(AFX_HZGL_H__FC9C6828_CEB8_11D4_9034_
F00952C10000__INCLUDED_)
```

2. CHzGL 类的 . CPP 文件

```
// HzGL.cpp: implementation of the CHzGL class.
//
//////////////////////////////////////
#include "stdafx.h"
#include "HzGL.h"
#include <gl\gl.h>\\OpenGL 头文件
#include <gl\glu.h>
#ifdef _DEBUG
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#define new DEBUG_NEW
#endif
//////////////////////////////////
// Construction/Destruction
//////////////////////////////////
CHzGL::CHzGL()
{
  m_iDisplayListStart1 = 1000;
  m_iDisplayListStart2 = 2000;
  m_angle.cx = 0 ; //向下旋转角度初值
  m_angle.cy = 0 ; //左右旋转角度初值
  m_theString = "电脑编程技巧" ;
  m_String = "电脑编程";
}
bool CHzGL::Create(CWnd * pWnd)
{/
  int iPixelType = PFD_TYPE_RGBA;
DWORD dwFlags = PFD_DOUBLEBUFFER |/  使用双缓冲区
            PFD_SUPPORT_OPENGL | //使用 OpenGL
            PFD_DRAW_TO_WINDOW; // 窗口像数格式
  //描述图形表面像数格式结构
  PIXELFORMATDESCRIPTOR pfd ;
  memset(& pfd, 0, sizeof(PIXELFORMATDESCRIPTOR)) ;
  pfd.nSize = sizeof(PIXELFORMATDESCRIPTOR);
  pfd.nVersion = 1 ;        // 版本号
  pfd.dwFlags = dwFlags ;     //像数缓冲区属性
  pfd.iPixelType = iPixelType ;
  pfd.cColorBits = 24 ;          // 24 位颜色
pfd.cDepthBits = 32 ;      // 32 -bit 颜色浓度缓冲区
  pfd.iLayerType = PFD_MAIN_PLANE ;  // Layer type
  m_pdc = new CClientDC(pWnd);
//创建 TrueType 字体, 根据实现需要, 可用楷体、琥珀等字体
  LOGFONT lf;
  m_pdc->GetCurrentFont()->GetLogFont(& lf);
  lf.lfCharSet = 134;
  strcpy(lf.lfFaceName, "黑体") ;
  fontOpenGL1.CreateFontIndirect(& lf);
  strcpy(lf.lfFaceName, "宋体") ;
  fontOpenGL2.CreateFontIndirect(& lf);
  //选择与设备描述符(DC) 接近的像素格式
int nPixelFormat = ChoosePixelFormat(m_pdc->m_hDC, &
pfd);
  if (nPixelFormat == 0)
  {
AfxMessageBox("ChoosePixelFormat Failed %d\r\n", Get-
LastError()) ;
  return false;
  }
  // 设置与设备描述符(DC) 接近的像素格式
  BOOL bResult = SetPixelFormat(m_pdc->m_hDC, nPix-
elFormat, & pfd);
  if (!bResult)
  {
AfxMessageBox("SetPixelFormat Failed %d\r\n", GetLastEr-
ror()) ;
  return false;
  }
```

```
// 创建 OpenGL 绘图上下文  图形操作描述符(rendering context)
    //
    m_hrc = wglCreateContext(m_pdc->m_hDC);
    if (!m_hrc)
    {
AfxMessageBox("wglCreateContext Failed %x\r\n", GetLastError());
    return false;
    }
    //指定 rendering context
    if(!wglMakeCurrent(m_pdc->m_hDC, m_hrc))
    {
AfxMessageBox("wglMakeCurrent Failed %x\r\n", GetLastError());
        return false;
    }
    //调用成员私有函数—OpenGL 初始化
    if(OnInit()) return true;
    else return false;
}
bool CHzGL::OnInit()
{
    // 允许颜色浓度
    glEnable(GL_DEPTH_TEST);
    // 设置前颜色为材料颜色
glColorMaterial(GL_FRONT,
        GL_AMBIENT_AND_DIFFUSE);
    glEnable(GL_COLOR_MATERIAL);
    // 设置光线位置
    float fLightPosition[3] = {-1.0f, -1.0f, 1.0f};
    glLightfv(GL_LIGHT0, GL_POSITION, fLightPosition);
    // 允许计算光源
    glEnable(GL_LIGHTING);
    glEnable(GL_LIGHT0);
    // 设置屏幕底色
glClearColor(0.0f, 0.0f, 0.0f, 0.0f);//设置底色为黑色
//glClearColor(1.0f, 1.0f, 1.0f, 0.0f);//设置底色为白色
    Create3DFont();
    return true;
}
void CHzGL::Create3DFont()
{
    int i=0;
CFont * pOldFont=(CFont *) m_pdc->SelectObject(&fontOpenGL1);
    // 产生显示列表.)
    unsigned int j=0;
    i=0;
    j=0;
    while(i< m_theString.GetLength())
    {
if (IsDBCSLeadByte(m_theString[i])){ //判断是否为 双字节
        wglUseFontOutlines(
        m_pdc->m_hDC, // 所选字体的设备描述符
    (256+m_theString[i]) * 256+(256+m_theString[i+1]), // 要转换为显示列表的第一个字符
        1, // 转换为显示列表的字符个数
        m_iDisplayListStart1+j, // 指定的显示列表基数
        0.0, // 指定的最大偏离值,一般为 0
0.15f, // 它是立体字体具体表现,Z 轴(纵深方向)方向的值
WGL_FONT_POLYGONS, // 显示的字体轮廓线格式,多边形-实心
        &m_agmf1[j]); // 接受字符特性
        FTextList[j]=j;
        j++;
        i++;i++;
    }
        else{
        wglUseFontOutlines(
        m_pdc->m_hDC, // 所选字体的设备描述符
        m_theString[i], //转换为显示列表的第一个字符
        1, // 转换为显示列表的字符个数
        m_iDisplayListStart1+j, // 指定的显示列表基数
        0.0, // 指定的最大偏离值,一般为 0
0.15f, // 它是立体字体具体表现,Z 轴(纵深方向)方向的值
WGL_FONT_POLYGONS, // 显示的字体轮廓线格式,多边形-实心
        &m_agmf1[j]); // 接受字符特性
        FTextList[j]=j;
        i++;
        j++;
    }
}
    m_pdc->SelectObject(&fontOpenGL2);
    i=0;
    j=0;
     while(i<m_String.GetLength())
    {
if (IsDBCSLeadByte(m_String[i])){ ///判断是否为双字节
        wglUseFontOutlines(
        m_pdc->m_hDC, // 所选字体的设备描述符
    (256+m_theString[i]) * 256+(256+m_theString[i+1]), // 要转换为显示列表的第一个字符
        1, // 转换为显示列表的字符个数
        m_iDisplayListStart2+j, // 指定的显示列表基数
        0.0, // 指定的最大偏离值,一般为 0
0.15f, // 它是立体字体具体表现,Z 轴(纵深方向)方向的值
WGL_FONT_POLYGONS, // 显示的字体轮廓线格式,多边形-实心
        &m_agmf2[j]); // 接受字符特性
        FTextList[j]=j;
        j++;
        i++;i++;
    }
        else{
        wglUseFontOutlines(
        m_pdc->m_hDC, // 所选字体的设备描述符
        m_theString[i], //转换为显示列表的第一个字符
        1, // 转换为显示列表的字符个数
        m_iDisplayListStart2+j, // 指定的显示列表基数
        0.0, // 指定的最大偏离值,一般为 0
0.15f, // 它是立体字体具体表现,Z 轴(纵深方向)方向的值
```

```
WGL_FONT_POLYGONS, // 显示的字体轮廓线格式,多边形-实心
            & m_agmf2[j]); // 接受字符特性
            FTextList[j] =j;
            i + +;
            j + +;
      }
}
      if (pOldFont) m_pdc - >SelectObject(pOldFont) ;
      }
CHzGL: : ~CHzGL()
{
      if (m_hrc)
      {
            // 删除图形操作描述符
            if (m_hrc = = wglGetCurrentContext())
                  wglMakeCurrent(NULL, NULL) ;
            wglDeleteContext(m_hrc) ;
            m_hrc = NULL ;
      }
      delete m_pdc;
      }
bool CHzGL: : Render()
{
      // Make the HGLRC current
      wglMakeCurrent(m_pdc - >m_hDC, m_hrc);
      //第一个字符串颜色
      static GLdouble purple[3] = {2, 1, 1 } ;
      //第二个字符串四个字体不同颜色
      static GLubyte byPink[3] = {196, 0, 196} ;
      static GLubyte byTeal[3] = {0, 153, 168} ;
      static GLubyte byBlue[3] = {42, 38, 215} ;
      static GLubyte byGreen[3] = {0, 171, 82} ;
      // Clear the color and depth buffers
      glClear(GL_COLOR_BUFFER_BIT |
GL_DEPTH_BUFFER_BIT);
      //显示第一行汉字
      glMatrixMode(GL_MODELVIEW);
      glLoadIdentity();
glTranslated( -3. 10, 0. 9, -7. 0 ); //字体位置 X, Y, 字体大小
      glRotated(m_angle. cx, 1, 0, 0); // 旋转角度 X, Y, Z
      glColor3dv(purple) ;
      glListBase(m_iDisplayListStart1);
      glCallLists(m_theString. GetLength( ),
GL_UNSIGNED_BYTE, & FTextList);
      //显示第二行汉字
      double widthMN = (m_agmf2[0] . gmfCellIncX +
m_agmf2[3]. gmfCellIncX) * 0. 5 ;
      glLoadIdentity();
      glTranslated(0. 0, -0. 65, -3. 0 );
      glRotated(m_angle. cy, 0. 0, 1. 0, 0. 0);
      glTranslated( -widthMN, 0. 0, 0. 0) ;
      glScaled(0. 5, 0. 5, 0. 5) ;
      glColor3ubv(byPink) ;
      glCallList(m_iDisplayListStart2) ;
      glColor3ubv(byTeal) ;
      glCallList(m_iDisplayListStart2 +1) ;
      glColor3ubv(byBlue) ;
      glCallList(m_iDisplayListStart2 +2) ;
      glColor3ubv(byGreen) ;
      glCallList(m_iDisplayListStart2 +3) ;
      glFlush() ;
      GdiFlush() ;
      SwapBuffers(m_pdc - >m_hDC) ;
      GdiFlush() ;
      spin(); //调用成员私有函数—旋转增量
      return true;
})
BOOL CHzGL: : OnResize(int cx, int cy
{
      GLdouble gldAspect = (GLdouble) cx/ (GLdouble) cy;
      glMatrixMode(GL_PROJECTION);
      glLoadIdentity();
      //设置透视发射矩阵
      gluPerspective(30. 0, gldAspect, 1. 0, 10. 0);
      glViewport(0, 0, cx, cy);
      return TRUE ;
}
BOOL CHzGL: : spin()//旋转函数
{ ; ;
      m_angle. cx + = 10 ;
      m_angle. cy + = 10 ;
      if (m_angle. cx >= 360) m_angle. cx = 0
      if (m_angle. cy >= 360) m_angle. cy = 0
}
```

3. ChzGL 类提供应用程序可调用的函数

ChzGL 类有 5 个公有成员函数提供应用程序使用（实际只需关心三个函数调用）：

1 构造函数 CHzGL ：确定显示的字符串、变量初始化，在该类声明处调用，或动态指针申请 pHzGl = new CHzGL 时调用。

2 析构函数 ~ CHzGL ：删除图形操作描述符 m_hrc，释放申请的空间，在声明函数撤消或申请指针释放时（delete pHzGl）调用。注意析构函数是隐式调用。

3 初始化函数 Create CWnd * pWnd ：用于确定显示的字体、建立图形操作描述符 m_hrc、选择光线、底色等参数，计算显示字体轮廓列表，以及用于 OpenGL 初始化操作。当在对话框中显示立体汉字时，应在对话框类 Cdialog 的虚函数 OnInitDialog 中调用初始化函数；在窗口中显示立体汉字时，可在视类 CView 的虚函数 OnCreate LPCREATESTRUCT lpCreateStruct 调用该函数。它们调用方式都是 Create this 无论是对话框，还是窗口都是从 CWnd 类继承的。

4 视口调整函数 OnResize int cx int cy ：设置透视发射矩阵，根据实际显示尺寸调整比例，它可在对话框或视类的 OnSize UINT nType int cx int cy 中调用。

5 显示、旋转字体函数 Render ：根据字体颜色、字体大小、旋转角度显示立体字体。它可在 OnDraw CDC * pDC 或定时器响应函数 OnTimer UINT nIDEvent 中调用。

四、示例程序编制

为了简单说明 CHzGL 类的应用效果，现编制一个单文档窗口示例程序。

1. 建立工程文件 Show3DHz

打开 VisualC + + 6.0，利用 AppWizard 新建名为 Show3DHz 工程文件，选中单文档，其余为缺省值。在视类建立‘旋转字体”弹出菜单，包括‘开始旋转”和‘结束旋转”两项菜单项。

2. ChzGL 类代码编制

利用 Insert - > new class 功能插入无父类的 CHzGL 新类，形成两个文件 HzGL. h、HzGL. cpp，将以上 CHzGL 类全部代码写入这两个文件中，并在工程设置的 Link 属性页中加入 opengl32. lib、glu32. lib 、glaux. lib 三个导入库文件。

3. ChzGL 类应用代码编制

在视类 CShow3DhzView 的头文件中加入包含文件

```
# include   HzGL. h
```

添加数据成员：

```
private
    CHzGL m_GL
    UINT Time1   //定时器
```

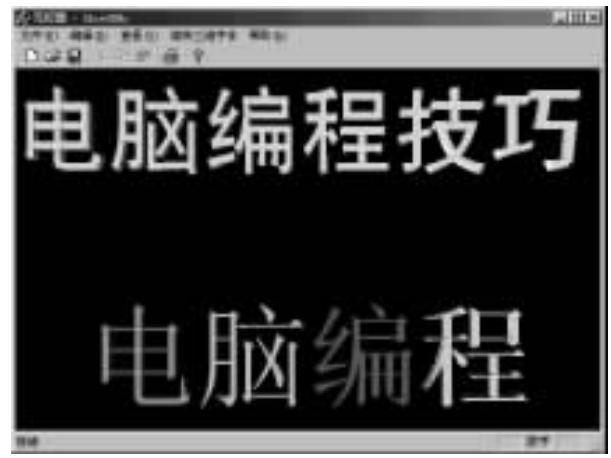
用 ClassWizard 映射 WM_CREATE、WM_SIZE、WM_TIMER 消息和菜单‘开始旋转”、‘结束旋转”菜单响应函数，并添加消息处理函数，其代码如下：

```
//消息 WM_CREATE –用于 OpenGL 初始化
int CShow3DHzView: : OnCreate(LPCREATESTRUCT lpCreat-
eStruct)
{
    if (CView: : OnCreate(lpCreateStruct) = = –1)
        return –1;
    BOOL bResult = m_GL. Create(this) ;
    if (bResult)
        return 0 ;
    else
        return –1;
}
//消息 WM_SIZE –立体汉字自动适应窗口大小
void CShow3DHzView: : OnSize(UINT nType, int cx, int cy)
{
    CView: : OnSize(nType, cx, cy);
    m_GL. OnResize(cx, cy);
    }
//消息 WM_TIMER—用定时器增加字体旋转功能
void CShow3DHzView: : OnTimer(UINT nIDEvent)
{
    Invalidate();
    //CView: : OnTimer(nIDEvent);
}
//此处实现立体汉字显示、旋转
void CShow3DHzView: : OnDraw(CDC * pDC)
{
  CShow3DHzDoc * pDoc = GetDocument();
  ASSERT_VALID(pDoc);
  m_GL. Render() ;
}
//‘开始旋转’菜单响应函数
void CShow3DHzView: : OnBegin()
{
SetTimer(Time1, 40, NULL);
}
//‘结束旋转’菜单响应函数
void CShow3DHzView: : OnStop()
{
KillTimer(Time1);
}
```

为了使旋转字体时，屏幕不闪烁，应增加 WM_ERASEBKGND 消息响应函数，使生成的窗口为透明窗口；应将源代码修改如下：

```
BOOL CShow3DHzView: : OnEraseBkgnd(CDC * pDC)
{
  return TRUE ;
  //return CView: : OnEraseBkgnd(pDC);
}
```

以上工作完成并编译后即可运行，本文示例已在 Windows98 VC + + 6.0 环境下调试通过，其执行结果如下图：



点击‘旋转字体”的‘开始旋转”菜单，在屏幕上看到上、下两排汉字以不同方向旋转。再点击‘旋转字体”的‘结束旋转”菜单，两排汉字停止旋转。

根据应用要求不同，用 CHzGL 类也可建一个动态链接库 DLL，在 CHzGL 类前加入 AFX_EXT_CLASS，编译形成 . DLL 和 . LIB 两个文件，并拷入到应用程序中一同连编。要在对话框中显示立体字体，可参照示例程序编制。

参考文献

Microsoft MSDN Library Visual Studio 6.0 版

收稿日期：2000 年 12 月 26 日）

在 ObjectARX 程序中动态添加和删除 AutoCAD 菜单命令

刘良华　袁英战

摘　要　本文利用 AutoCAD 2000 提供的类型库，编写了一个 ARX 例程。该例程在加载时将应用程序相关的菜单项添加到指定的菜单中，而在卸载时又将这些定制的菜单项全部删除。本文为 AutoCAD 主程序界面与应用程序功能紧密地结合在一起提供了新的方法和思路。

关键字　AutoCAD 二次开发，定制菜单，COM，ObjectARX

菜单是 Windows 应用程序标准的用户界面元素，其可视化的操作界面让使用者操作起来既简洁又方便。一般而言，AutoCAD 二次开发人员都有这样一种体会，即对于开发完成的大量应用程序模块，往往通过定制相应的菜单命令来执行之。通过菜单命令执行应用程序模块，用户不必牢记烦琐的文本命令。与以前的版本一样，AutoCAD 2000 也提供了定制菜单的功能，即允许用户编写自己的菜单定义文件。如果定制的菜单（或菜单命令）需要挂接在 AutoCAD 本身提供的菜单中，那么只能在原有菜单文件的基础上进行必要的修改和添加。如果应用程序卸载了，其相应的菜单命令尽管无法执行，但依然存留在当前的菜单中，显得很不方便。

通过 AutoCAD 2000 提供的 COM 接口动态添加和删除菜单命令，而无需修改原有的菜单文件。当加载时，该例程为注册的程序模块添加相应的菜单命令项，而当例程卸载时，又将相应的菜单项全部删除。

本文例程使用 ObjectARX 2000 类库，采用 Visual C++ 6.0 开发，在 AutoCAD 2000 上调试通过。

一、建立工程

AutoCAD 2000 提供了引用其 COM 对象的类型库（type library），即 acad.tlb。该文件既可以在 AutoCAD 2000 的安装目录下找到，也可以在 ObjectARX 2000 安装目录的 hInc h子目录中找到，并且两者是一样的。在 Visual C++ 6 中调用 COM 对象有两种途径：一是使用 MFC，从而可以利用 Visual C++ 6 提供的 ClassWizard（类向导）工具输入类型库；二是不使用 MFC，而直接采用 Win32 编程的方法引用类型库。本文以第二种方法为例，即不使用 MFC 的方式调用 AutoCAD 2000 提供的类型库。

按照 AutoCAD2000 ARX 开发的方法和步骤，在 Visual C++ 6 中使用 File | New 菜单命令创建一个空白工程（关于开发的方法和步骤，详见有关书籍）。选择工程类型为 Win32 Dynamic - Link Library（Win32 动态连接库），并且指定工程名称为 CCOMAddMenu。如图 1 所示。

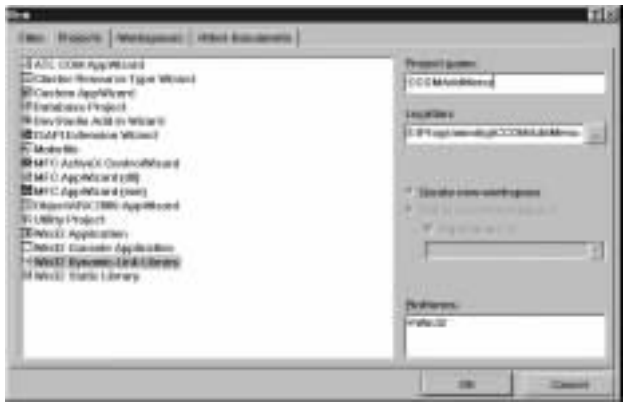


图 1　新建一个 Win32 动态库工程

接下来，使用 Project | Settings 菜单命令设置工程参数如下（其它参数保持默认值）：

1　C/C++ 参数：选择 Category（分类）为 Code Generation，确保 Use run - time library（运行库）列表项为：Multithreaded DLL。

2　Link 参数：选择 Category 为 General，将输出 DLL 文件的后缀更改为 .arx；选择引用的库文件为 acad.lib、rxapi.lib、acedapi.lib 和 acrx15.lib，库文件名之间以空格相隔。

最后，确保 Visual C++ 6 的头文件和库文件引用路径是正确的，否则，使用 Tools | Options | Directories 命令进行设置。

二、添加实现代码

向当前的工程项目添加三个文件，即头文件（CCOMAddMenu.h）、工程实现文件（CCOMAddMenu.cpp）以及模块定义文件（CCOMAddMenu.def）。

1. 打开 CCOMAddMenu.h 头文件，并添加如下代码：

```
/* CCOMAddMenu.h ——头文件 */
```

```
#include <aced.h>          // ARX 头文件
#include <rxregsvc.h>
#include <stdio.h>         // C/C++标准头文件
#include <tchar.h>         //
// 引入 AutoCAD 2000 类型库
#import "acad.tlb" no_implementation raw_interfaces_only named_guids
// 函数原型定义
void addOrDeleteMenuItem(void);          // 该函数动态添加/删除指定的菜单命令
void HelloWorld(void);                   // 该函数定义一个命令模块
// 头文件定义结束
```

注意：输入 COM 类型库是采用#import 预编译指令进行的，它可以附带一些参数 有关参数的含义，详见 Visual C++6 的联机帮助）。类型库文件名可以采用相对路径，也可以采用绝对路径。如果是相对路径，则必须保证其位于 Visual C++6 的库文件引用路径上。

2. 打开 CCOMAddMenu.cpp 文件，并添加如下代码：

```
/* CComAddMenu.cpp ——执行文件 */
#include "CComAddMenu.h"
// 该函数定义一个命令模块
void HelloWorld(void)
{
  acedAlert("Hello World!");
  return;
}
// 通过 AutoCAD 提供的 COM 接口，该函数动态添加 删除指定的菜单命令
void addOrDeleteMenuItem(void)
{
  // 静态变量，指示菜单命令是否已经添加
  static Adesk::Boolean bIsMenuLoaded = Adesk::kFalse;
  AutoCAD::IAcadPopupMenu * pPopUpMenu;
  VARIANT index;          // 临时变量，作索引用
// 通过 QueryInterface()方法获得 AutoCAD 2000 的实例对象
  AutoCAD::IAcadApplication * pAcad;
  HRESULT hr = NOERROR;
      CLSID clsid;
  LPUNKNOWN pUnk = NULL;
  LPDISPATCH pAcadDisp = NULL;
hr = ::CLSIDFromProgID(L"AutoCAD.Application", &clsid);
  if (SUCCEEDED( hr ))
  {
   if(::GetActiveObject(clsid, NULL, &pUnk) == S_OK)
     {
if(pUnk->QueryInterface(IID_IDispatch, (LPVOID *) &pAcadDisp) != S_OK)
                return;
            pUnk->Release();
        }
     }
  if( SUCCEEDED(pAcadDisp->QueryInterface(AutoCAD::IID_IAcadApplication, (void**)&pAcad)) )
pAcad->put_Visible(true);      //使 AutoCAD 2000 可见
    else
  {
      acedPrompt("\nQueryInterface 出错!");
      return;
  }
  // 获得当前加载的菜单组
  AutoCAD::IAcadMenuGroups * pMenuGroups;
  pAcad->get_MenuGroups(& pMenuGroups);
pAcad->Release();    // AutoCAD 2000 实例对象已不需要
 // 获得第一个菜单组，通常为 ACAD，即 AutoCAD 2000 的主菜单组
  VariantInit(& index);
  V_VT(& index) = VT_I4;
  V_I4(& index) = 0;
  AutoCAD::IAcadMenuGroup * pMenuGroup;
  pMenuGroups->Item(index, & pMenuGroup);
  pMenuGroups->Release();
  // 获得 ACAD 包含的各下拉子菜单
  AutoCAD::IAcadPopupMenus * pPopUpMenus;
  pMenuGroup->get_Menus(& pPopUpMenus);
  pMenuGroup->Release();
// 获取指定的用于添加菜单项的菜单，此处为 ACAD 的 Window 菜单
  long numbers;
  pPopUpMenus->get_Count(& numbers);
    for( int num = 0; num < numbers; num++ )
  {
    VariantInit(& index);
    V_VT(& index) = VT_I4;
    V_I4(& index) = num;
      pPopUpMenus->Item(index, & pPopUpMenu);
    if(pPopUpMenu == NULL)
    {
      acedPrompt("\n 无法获取指定菜单!");
      return;
    };
BSTR menuName;
      TCHAR tString[256];
      pPopUpMenu->get_Name(& menuName)
     WideCharToMultiByte(CP_ACP, 0, menuName, -1, tString, 256, NULL, NULL);
if( !stricmp(tString, _T("&Window")))// 如果是 Window 菜单
          break;
    }
  pPopUpMenus->Release();
    if( num >= numbers || pPopUpMenu == NULL)
// 如果没有找到合适的菜单
      return;
AutoCAD::IAcadPopupMenuItem * pPopUpMenuItem;
  WCHAR wstrMenuItemName[256], wstrMenuItemMacro[256];
if ( bIsMenuLoaded == Adesk::kFalse)          //如果菜单未添加，则添加
  {
    // 添加菜单项及其宏命令，宏命令串末尾添加一个空格
```

```
        VariantInit(& index);
        V_VT(& index) = VT_I4;
        V_I4(& index) = 0;        // 添加的菜单项为第一项
        MultiByteToWideChar(CP_ACP, 0, ″& Hello World″,
-1, wstrMenuItemName, 256);
MultiByteToWideChar(CP_ACP, 0, ″LLH_COMMANDS_
HELLO ″, -1, wstrMenuItemMacro, 256);
        pPopUpMenu->AddMenuItem(index, wstrMenuItem-
Name, wstrMenuItemMacro, & pPopUpMenuItem);
MultiByteToWideChar(CP_ACP, 0, ″单击该菜单命令执行\″
Hello World!\″例程.″, -1, wstrMenuItemName, 256);
pPopUpMenuItem->put_HelpString(wstrMenuItemName);
// 附加帮助提示
            pPopUpMenuItem->Release();
 V_I4(& index) = 1;        // 添加的菜单项为第二项
pPopUpMenu->AddSeparator(index, & pPopUpMenuItem);
// 添加分隔符
            pPopUpMenuItem->Release();
       acedPrompt(″\n 菜单命令添加在 Window 菜单中.″);
            bIsMenuLoaded = Adesk::kTrue;
          }
       else    // 如果菜单已经添加,则删除之
       {
// 删除菜单项,先删除第二项,后删除第一项.注意删除的顺序.
        VariantInit(& index);
        V_VT(& index) = VT_I4;
        V_I4(& index) = 1;        // 删除第二个菜单项
        pPopUpMenu->Item(index, & pPopUpMenuItem);
        pPopUpMenuItem->Delete();
        V_I4(& index) = 0;        // 删除第一个菜单项
        pPopUpMenu->Item(index, & pPopUpMenuItem);
        pPopUpMenuItem->Delete();
            acedPrompt(″\n 菜单命令已经删除!″);
            bIsMenuLoaded = Adesk::kFalse;
          }
        pPopUpMenu->Release();
    return;
}    // addOrDeleteMenuItem()函数定义结束
// 加载应用程序的初始化函数
void initApp()
{
addOrDeleteMenuItem(); //第一次执行,添加指定菜单命令
    /* 向 AutoCAD 注册一个定制的命令 */
          acedRegCmds->addCommand(″ASDK_LLH_COMMANDS″,
″ADSK_LLH_COMMANDS_HELLO″, ″LLH_COMMANDS_HELLO″,
ACRX_CMD_MODAL, HelloWorld);
    return;
}
// 卸载应用程序的清理函数
void unloadApp()
{
 addOrDeleteMenuItem(); //第二次执行,删除指定菜单命令
acedRegCmds->removeGroup(″ASDK_LLH_COMMANDS″);
    return;
}
// ARX 程序入口函数
extern ″C″ AcRx::AppRetCode acrxEntryPoint(AcRx::
AppMsgCode msg, void* pkt)
{
   switch (msg) {
   case AcRx::kInitAppMsg:
      acrxDynamicLinker->unlockApplication(pkt); // 允许
应用程序中途能够卸载
      acrxRegisterAppMDIAware(pkt);
      initApp();        //初始化应用程序
      break;
   case AcRx::kUnloadAppMsg:
      unloadApp();      //应用程序退出前进行必要的清理
      break;
   case AcRx::kLoadDwgMsg:
      break;
   }
   return AcRx::kRetOK;
}
```

函数 addOrDeleteMenuItem 向 AutoCAD 2000 的 Windows 菜单添加（或删除）定制的菜单命令（“Hello World”）和分隔符。函数的主要过程是：首先，通过一系列调用 QueryInterface 函数，获得 AutoCAD 2000 的运行实例；其次，获得 ACAD 菜单组的 Windows 菜单（此处假定当前加载的菜单组是 ACAD，即 AutoCAD 2000 的标准菜单）；最后，根据静态变量 bIsMenuLoaded 的值决定，是添加菜单项还是删除菜单项。

事实上，通过 COM 接口访问 AutoCAD 2000，关键是通过一系列 QueryInterface 函数调用获得其实例对象。在获得实例对象之后，余下的工作就是应用程序本身需要实现的功能逻辑了。

不过需要注意的是，在进行字符串的有关操作时（如菜单名的比较、添加指定名称的菜单项等），需要事先进行字符代码的转换，否则，将出现错误结果。例程中字符代码的转换，是使用 WideCharToMultiByte 函数或 MultiByteToWideChar 函数来进行的。

3. 最后，向模块定义文件 CCOMAddMenu.def 添加定义代码如下：

```
;/* * * * * * CCOMAddMenu.def * * * * * */
LIBRARY      CCOMAddMenu
DESCRIPTION  'ARX program for AutoCAD 2000'
EXPORTS
   acrxEntryPoint      PRIVATE
   _SetacrxPtp         PRIVATE
   acrxGetApiVersion   PRIVATE
```

注意：英文分号（;）开头的行为模块定义文件的注释行。至此，源代码编辑完毕。

三、编译和运行应用程序

编译整个工程，得到 CCOMAddMenu.arx 文件。如果按照上述步骤进行的话，那么整个编译过程不会出现任何警告或错

误信息。

在 AutoCAD 2000 下加载 CCOMAddMenu. arx，可以看到，在加载的时候，定制的菜单项 即 “Hello World ”命令和一个分隔符）已经添加完成，如图 2 所示。当光标移到 Hello

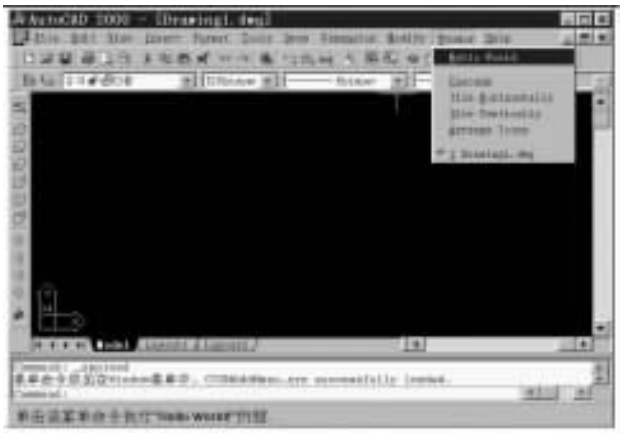


图 2 加载程序时添加指定的菜单命令

World 菜单命令时，状态栏显示定制的帮助提示。如果单击 Window | Hello World 菜单命令，则弹出一个 “Hello World ”信息框。

卸载应用程序时，定制的菜单命令则被删除，如图 3 所示。实际上，通过 COM 接口添加的菜单项不会修改 AutoCAD

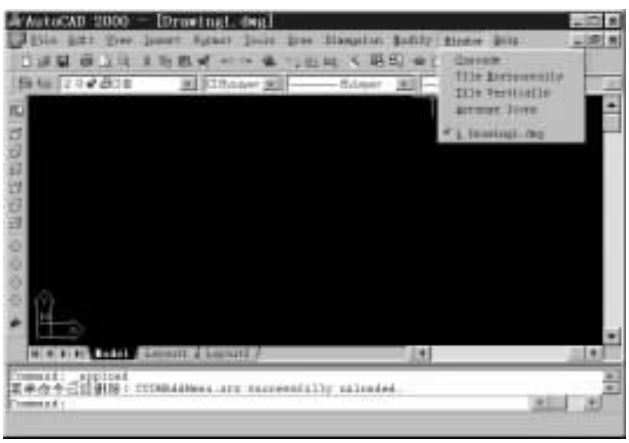


图 3 卸载程序时删除指定的菜单命令

2000 的菜单定义文件，它们只在当前的运行期间有效。当下一次启动 AutoCAD 2000 时，这些菜单项不会出现，除非应用程序再次添加。

四、结束语

本文介绍的例程通过 AutoCAD 2000 提供 COM 接口，在其原有菜单命令的基础上提供定制的菜单命令，并且避免了菜单定义文件的修改。此外，例程可以在程序加载时动态添加必要的菜单命令，而在卸载时又可以删除已不需要的菜单命令，这样既方便又灵活。本文为 AutoCAD 主程序界面与应用程序功能紧密地结合在一起，提供了新的方法和思路。当然，本文目的仅在于抛砖引玉，欢迎广大 AutoCAD 二次开发人员提出更多、更好的应用方法。

参考资料

1. 刘良华、朱东海著 . AutoCAD2000 ARX 开发技术 . 北京 清华大学出版社，2000

2. 杨秀章译 . COM 技术内幕——微软组件对象模型 . 北京 清华大学出版社，1999

收稿日期：2001 年 1 月 12 日）

注册方式软件加密

海迅达注册控件及注册管理程序

以前软件的加密大多采用软件狗的方式，但该方式存在着一些明显的特点：

- 成本过高，对于千元以下产品不合适；
- 对软硬件环境有较高限制，容易产生硬件冲突；
- 可采用共享器等方式解密；
- 对于盗版狗难以防范；
- 软件免损坏后更换麻烦。

鉴于以上缺点，软件加密的另一种方式—注册方式被越来越广泛地采用。例如，大量的共享软件和 MS Office 等都采用了注册方式加密。

注册方式加密具有保密性强、适用面广，价格低廉、方便管理等众多优点，是软件加密的发展方向。

但注册方式加密也不是轻而易举的事情，开发者需周全考虑用户使用方便、防解密、防盗版注册管理方便、能否适用于各种操作系统和硬件环境等复杂问题。

最近，由北京海迅达科技有限公司推出的 “海迅达注册控件及注册管理程序 ”为广大开发人员提供了一种便利的注册方式加密手段。

“海迅达注册控件及注册管理程序 ”开发包括一个标准的 ActiveX 控件 用于产品中实现注册加密功能 和一套完整的用户注册管理程序，并提供 VC + + 和 VB 使用示例。

该产品具有以下特点：

- 价格低廉，批量越大，平均每件的价格越低；
- 提取用户微机的软硬件信息加密，无法仿冒；
- 采用 ActiveX 控件方式加密，难以解密；
- 适用于 VC + + 、VB、Delphi 等开发环境；
- 适用于 Windows95 - 2000 的全系统操作系统；
- 对用户的硬件环境没有硬性要求，不存在硬件冲突；
- 用户注册管理程序保留有全面的用户信息，用户管理方便；
- 用户注册管理程序提供产品序列号打印功能；
- 由公司决定允许用户注册次数；
- 开放用户注册数据库，开发商可自行处理及自行开发注册查询、打印软件。

Windows CE 应用程序中的密码技术

丁胜昔　范慧琴

摘　要　本文描述了 Microsoft 密码系统，介绍了密码 API 及其在 Windows CE 应用程序中的实现，还概述了密码服务提供者的开发与签名的方法。

关键词　密码 API（CAPI），密码服务提供者（CSP）

密码技术提供了一种以加密代码或密码形式发布文件的方法，这些文件只能由期望的接收者读取。密码技术通过维护保密性和保证数据完整性可以在基于 Windows CE 的应用程序中获得安全通信。

加密是将数据编码为密文的过程，一个密码系统的加密过程与密钥长度、加密方法、密文模式、数字签名、初始化向量及要素值等相关。解密是将编码的数据转换为明文的过程，是加密的逆过程。

一、Microsoft 密码系统

Microsoft 密码系统可以分为三个可执行部件：应用程序、操作系统（OS）和密码服务提供者（CSP）。应用程序通过密码 API（CAPI）与操作系统通信，操作系统通过密码服务提供者接口（CSPI）与 CSP 通信。它们之间的关系如图 1 所示：

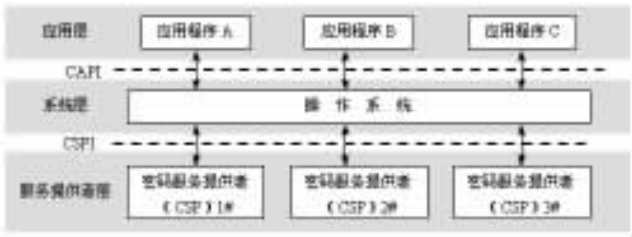


图 1　Microsoft 密码系统

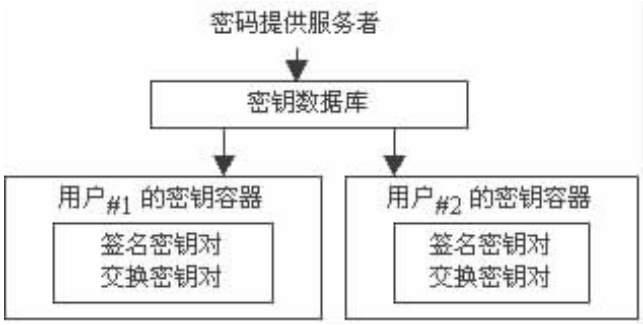


图 2　密钥数据库与 CSP、密钥容器之间的关系

CSP 负责创建和删除密钥，并用密钥执行各种密码操作。CSP 通过 Coredll. dll 与应用程序进行通信。每个 CSP 都具有名字和类型，目前 Windows CE 包含的 CSP 名字为 Microsoft 基础密码提供者 1. 0 版本，类型为 PROV_RSA_FULL，CSP 的名字是唯一的，但类型可以不唯一。

这里介绍两个概念；密钥数据库和密钥 BLOB。

每个 CSP 具有一个存储它的持续密码密钥的密钥数据库，该数据库包含一个或多个密钥容器，每个密钥容器包含属于特定用户的所有密钥对。图 2 说明了 CSP、密钥数据库和密钥容器之间的关系。

密钥二进制对象（密钥 BLOB）提供了一种在 CSP 之外存储密钥的方法，它用作将密钥从一个提供者安全地传送到另一个提供者的媒体。它有三种形式：简单、公有和私有。

二、密码 API 的 Windows CE 实现

在基于 Windows CE 的应用程序中，可以用 CAPI 完成的密码操作包括：连接 CSP、产生密码密钥、交换密码密钥、加密和解密数据、创建数字签名等，使用 CAPI 的数据加密都由对称算法执行。

2. 1 连接 CSP

表 1 列出了应用程序可以用于连接 CSP 的函数，这些函数还使应用程序能够用名字选择特定的 CSP 或者用需要的功能类获得 CSP。

表 1　用于连接 CSP 的函数及其功能描述

函数	描述
CryptAcquireContext	获取当前用户在特定 CSP 内的密钥容器句柄
CryptGetProvParam	获取 CSP 的属性
CryptReleaseContext	释放 CryptAcquireContext 函数获得的句柄
CryptSetProvider	选择特定 CSP 类型的用户缺省 CSP
CryptSetProvParam	指定 CSP 的属性

每次运行应用程序时，应用程序调用的第一个 CAPI 函数是 CryptAcquireContext，它返回给应用程序特定 CSP 的句柄，该句柄指定 CSP 内的特定密钥容器。在 C + + 中可以这样调用这些函数：

```
// 获取一个缺省 PROV_RSA_FULL 提供者句柄
HCRYPTPROV hProv = 0;
CryptAcquireContext(& hProv, NULL, NULL, PROV_RSA_FULL, 0);
// 读取缺省 CSP 的名字 .
BYTE pbData[1000];
DWORD cbData = 1000;
CryptGetProvParam(hProv, PP_NAME, pbData, & cbData, 0);
```

```
// 读取缺省密钥容器的名字 .
CryptGetProvParam(hProv, PP_CONTAINER, pbData, & cb-
Data, 0);
// 执行密码操作
...
// 释放提供者句柄
CryptReleaseContext(hProv, 0);
```

2.2 产生密码密钥

应用程序中可以用 CryptDeviceKey 和 CryptGenKey 函数创建密钥，其中 CryptDeviceKey 函数产生从口令推导出的密钥，CryptGenKey 产生随机密钥。通过调用 CryptGenKey 函数并指定 AT_KEYEXCHANGE 或 AT_SIGNATURE 可以创建交换密钥对和签名密钥对。

```
HCRYPTKEY hKey = 0;
CryptGenKey(hProv, AT_KEYEXCHANGE, CRYPT_EXPORTA-
BLE, & hKey);
```

2.3 交换密码密钥

当用户需要保存密钥以便应用程序以后使用，或者将密钥发送给其他人的时候，需要将密钥从 CSP 的安全环境输出到密钥 BLOB 中，表 2 列出了可用于创建、配置、删除密码密钥以及可用于其它用户交换密钥的函数。

表 2 创建、配置、删除密码密钥的函数及其功能描述

函数	描述
CryptDestroyKey	删除密钥
CryptExportKey	在应用程序内存空间从 CSP 向密钥 BLOB 输出密钥
CryptGenRandom	产生随机数据
CryptGetKeyParam	获取密钥参数
CryptGetUserKey	得到交换密钥或者签名密钥的句柄
CryptImportKey	从密钥 BLOB 向 CSP 输入密钥
CryptSetKeyParam	指定密钥参数

2.4 加密和解密数据

应用程序可以通过 CryptEncrypt 函数用指定的加密密钥对一部分明文进行编码，通过 CryptDecrypt 函数用指定的解密密钥对一部分密文进行解码。

在对数据进行加密或者解密之前需要有加密密钥，加密算法在创建密钥时指定，也可以用 CryptSetKeyParam 函数指定附加的加密参数。

下面的一段代码从文件 lpszSource 中读取数据，用 RC2 块密文编码数据，然后将被编码的数据写到文件 lpszDestination 中。为节省篇幅，代码没有考虑错误异常，解码的方法与它类似。

```
BOOL EncryptFile (LPTSTR lpszSource, LPTSTR lpszDesti-
nation, LPTSTR lpszPassword)
{
 FILE * hSrcFile = NULL, * hDestFile = NULL;
 HCRYPTPROV hProv = 0;
 HCRYPTHASH hHash = 0;
 HCRYPTKEY hKey = 0, hXchgKey = 0;
 PBYTE pbBuffer = NULL, pbKeyBlob = NULL;
 BOOL bEOF = 0, bReturn = FALSE;
 DWORD dwCount, dwKeyBlobLen;
 // 打开源文件
 hSrcFile = _wfopen (lpszSource, TEXT("rb"));
 // 打开目标文件
 hDestFile = _wfopen (lpszDestination, TEXT("wb"));
 // 获得缺省提供者的句柄
  CryptAcquireContext ( & hProv, NULL, NULL,
PROV_RSA_FULL, 0);
 if (lpszPassword == NULL) { // 使用随机会话密钥
加密文件
  // 创建随机会话密钥
  CryptGenKey (hProv, CALG_RC2, CRYPT_EXPORTABLE,
& hKey);
// 获得交换公共密钥的密钥句柄
CryptGetUserKey (hProv, AT_KEYEXCHANGE, & hX-
chgKey);
  // 确定密钥 BLOB 的大小并为其分配内存
  CryptExportKey (hKey, hXchgKey, SIMPLEBLOB, 0,
NULL, & dwKeyBlobLen);
  pbKeyBlob = (PBYTE)malloc(dwKeyBlobLen);
  // 将会话密钥输出到简单密钥 BLOB
  CryptExportKey (hKey, hXchgKey, SIMPLEBLOB, 0,
pbKeyBlob, & dwKeyBlobLen);
  // 将密钥 BLOB 的大小写到目标文件
  fwrite ( & dwKeyBlobLen, sizeof (DWORD), 1, hDest-
File);
  // 将密钥 BLOB 写到目标文件
  fwrite (pbKeyBlob, 1, dwKeyBlobLen, hDestFile);
 }
 else { // 使用由密码产生的会话密钥加密文件
  // 创建散列对象
  CryptCreateHash (hProv, CALG_MD5, 0, 0, & hHash);
  CryptHashData (hHash, (PBYTE) lpszPassword, wcslen
(lpszPassword), 0);
  // 从散列对象获取会话密钥
  CryptDeriveKey (hProv, CALG_RC2, hHash, 0, & hKey);
 }
 // 分配内存
 pbBuffer = (PBYTE)malloc(dwKeyBlobLen);
 // 对源文件加密并将结果写入目标文件
 do {
  // 从源文件读取数据
  dwCount = fread (pbBuffer, 1, BLOCK_SIZE, hSrcFile);
  bEOF = feof (hSrcFile);
  // 加密数据
  CryptEncrypt (hKey, 0, bEOF, 0, pbBuffer, & dwCount,
BUFFER_SIZE);
  // 向目标文件写入数据
  fwrite (pbBuffer, 1, dwCount, hDestFile);
 } while (!bEOF);
 bReturn = TRUE;
 // 关闭文件，释放内存，删除会话密钥及句柄
 if (hSrcFile) fclose (hSrcFile);
 if (hDestFile) fclose (hDestFile);
 if (pbKeyBlob) free (pbKeyBlob);
 if (hKey) CryptDestroyKey (hKey);
```

```
  if (hXchgKey)     CryptDestroyKey (hXchgKey);
  if (hHash)     CryptDestroyHash (hHash);
  if (hProv)     CryptReleaseContext (hProv, 0);
  return bReturn;
}
```

2.5 创建数字签名

如果要从报文创建数字签名，需要从报文创建散列值，然后用签名者的私有密钥对散列值签名；如果要验证数字签名，首先，必须按照与创建签名时相同的方法从报文创建散列值，然后，用签名者的公共密钥验证散列值和签名，如果二者匹配，则确定报文是初始被签名的报文，没有被篡改过。表 3 列出了可用于计算数据的安全文摘以及创建和验证数字签名的函数。

表 3 用于计算数据的安全文摘以及创建和验证数字签名的函数

函数	描述
CryptCreateHash	创建一个空的散列对象
CryptDestroyHash	删除散列对象
CryptGetHashParam	获取散列对象参数
CryptHashData	散列数据块，将它加到指定的散列对象中
CryptHashSessionKey	散列会话密钥，将它加到指定的散列对象中
CryptSetHashParam	设置散列对象参数
CryptSignHash	对指定的散列对象签名
CryptVerifySignature	验证数字签名，给定被签名的散列对象的句柄

三、CSP 的开发与签名

如果 CSP 中包含的加密算法和数据格式已经确定和实现，那么创建 CSP 显得比较简单，可以按照以下步骤进行：

（1）创建输出所有 CSPI 函数的 DLL；

（2）为 CSP 编写安装程序，安装程序创建适当的注册表项；

（3）用 CSP 开发工具包和 Windows CE Platform Builder 测试 CSP；

（4）由 Microsoft 对 CSP 签名；

（5）再次测试 CSP。

为了使操作系统能够识别，每个 CSP 必须由 Microsoft 数字签名，操作系统周期性的验证签名以确保 CSP 没有被篡改过。

参考文献

1. 微软公司 . Microsoft Windows CE 通信指南 . 1999
2. Micorsoft Corporation. Platform Builder Library. 2000

（收稿日期：2001 年 1 月 2 日）

科见通公司的未来蓝图

拥有“财务长”商标权的科见通软件开发有限公司成立于 1999 年，总部设在香港。是一家资金和实力均雄厚的软件开发供应商。

从 98 年开始，中国电子商务的交易额不断攀升。面对这种情况，科见通公司率先完成了从财务软件供应商向企业管理软件开发商的转型，拓展了新的网上服务业务，实现企业自身运作的电子商务化。成功的建立了基于 INTERNET 和电子商务的解决方案，制定了电子商务的发展计划及方向。未来一段时间，公司将致力于：

1. 专业化的电子商务核心技术。科见通公司将帮助企业建立电子商务平台。

2. 本地化的电子商务支撑服务体系。将结合企业自身的特点，对客户进行最周到的服务。

3. 全球化的电子商务枢纽。将结合国际的形势，建立符合中国企业的电子商务。这样将极大的降低成本，优化企业内部资源，扩大采购资源，增加销售对象。

科见通软件开发有限公司营销及市场总监 Donal 说：“公司在向电子商务发展时，要把握以下几种策略：

1. 渐进策略。一步一个脚印，扎扎实实。在产品方面，要贯穿“思想是软件的灵魂”的理念，以品质满足客户需求，以品质赢得客户的忠诚。

2. 用户参与策略。充分给用户主动权，充分吸收用户意见，与用户共同推进企业电子商务的实施。

3. 先进，实用的策略。走在历史的前端，从局部计算机向 INTERNET 计算机技术的转变。在产品方向上，力求国际化，应用的一体化。

对于未来，科见通充满了信心；对于现在，科见通公司已发展成为一个跨地区的集团，已拥有包括北京，厦门，上海、香港等地的多家分支机构。企业的 SCM 财务系统、固定资产管理、人力资源管理、生产管理、POS 系统等多种财务及企业管理系统的客户已遍步东南亚，香港，台湾和大陆。

科见通公司的财务，进销存，生产制造管理，分支机构管理化一体化的整合设计使各部门结合起来，并可对企业分部门，班组，按责任中心进行核算，可对集团各分公司，办事处的业务单据、凭证、报表等进行及时数据交换、汇总、合并对比分析，各类业务可实现日清日结，企业信息随时掌握，使之在完成财务工作的同时，形成一整套的分析数据，帮助企业进行管理，进行数据决策，反映现代化的管理思想，并在实践中体现出来，完成整个经济活动中所有信息数据接收，流转以及处理。科见通公司通过对客户的调查，发现售后服务方面，用户最关心的是培训。根据这一情况，科见通公司完善了售后服务体系。目前，科见通公司的售后服务包括：

1. 对客户进行培训，对客户建立友好的合作关系。

2. 设置特别顾问及热线技术支持，并提供网上服务。

3. 专门的客户实施代表为客户提供服务。

4. 可以根据用户特殊要求在原有的系统上提供二次开发服务，使系统更完善和先进。

目前，科见通公司开发的 CRM 系统即将面市，此系统功能较为完善，其中包括：营销管理功能、培训管理功能、质量管理功能、销售管理功能、市场管理功能、客户管理功能等。并且，可以支持多种语言，支持 Web 界面和多种平台，系统的规模可以无限扩充，满足不同规模企业的需要，可以进行远程查询，嵌入第三方软件。通过分析企业电子商务的数据，产生可操作性的情报、报表等，帮助企业制定和执行营销计划。

不管是现在还是将在，科见通都将永远本着“客户为中心”的宗旨，并在此基础上完善技术，做用户最好的朋友。

使用 Winsock 控件制作网络版加密软件

王道守

摘 要 本文提出了利用 Winsock 控件 编写基于 TCP / IP 协议上的客户机 / 服务器程序的网络版加密软件的方法。

关键词 Winsock 控件,TCP / IP 协议,客户机 / 服务器,软件狗

一、网络版加密软件的思路

目前保护计算机软件版权的计算机技术方法常见的有两种方法：一是采用软盘磁道加密软件的方法，每次运行软件时，都先读软盘加密磁道，正确才准予运行，缺点是读软盘加密磁道的时间较长；二是采用加软件狗 软件保护卡）的方法，一机一 ‘狗 ”，加密效果好，但是一机一 ‘狗 ”费用较大。

笔者提出可以将软件分别设计在客户机/ 服务器上，使用 Winsock 控件建立网络连接，通过传输控制协议 TCP）进行数据交换，在服务器端并行口上安装一个软件狗，使该 ‘狗 ”加密服务器软件，没有 ‘狗 ”则服务器上的该软件无法运行，所有的客户机都无法进行数据交换，达到保护计算机软件版权的目的。主要优点是利用了现有的网络，只用一只 ‘狗 ”， 达到保护计算机软件版权的目的，降低了开发软件的费用，易于控制扩展客户机。

运行过程：首先运行服务器上的软件，各客户机定时向服务器发出联系密码信号→服务器接收联系信号，并判断信号及客户机要求提供的服务项目→读取并行口软件狗中的加密程序→服务器向客户机提供其要求的服务项目程序、图片及联系密码信号→客户机接收联系密码信号及程序，并判断信号是否正确→循环进行，服务器上的软件一旦停止，客户机程序都自动提示并停止运行。

二、Winsock 控件的通讯过程与主要属性、方法和事件说明

1. Winsock 控件的通讯过程

Winsock 控件对用户来说是不可见的，它提供了访问 TCP 和 UDP 网络服务的方便途径，通过使用 Winsock 控件可方便地编写出基于 TCP / IP 协议上的客户/ 服务器程序，支持网络中计算机之间的通讯和双向交换数据。其全称为 Microsoft Winsock control 设计使用前将此控件通过 “工程 ”菜单→ ‘部件 ”→选中 ‘Microsoft Winsock Control 6. 0 ”调入工具箱之中。

TCP 数据传输协议允许创建和维护与远程计算机的连接。使用 TCP 协议，应分别设计服务器端应用程序和客户端应用程序。

创建客户端应用程序，必须知道服务器计算机名或者 IP 地址 即指定 RemoteHost 属性），还要知道进行 ‘侦听 ”的端口 即指定 RemotePort 属性），然后调用 Connect 方法向服务器发送连接请求。

创建服务器应用程序，应设置一个收听端口 即指定 LocalPort 属性），调用 Listen 方法进入侦听状态。当侦听到客户计算机的连接信号时就会产生 ConnectionRequest 事件，调用 ConnectionRequest 事件内的 Accept 方法接受客户计算机的连接请求，得到 RequestID。

建立连接后，任何一方计算机都可以收发数据。为了发送数据，可调用 SendData 方法。当接收数据时会发生 DataArrival 事件。调用 DataArrival 事件内的 GetData 方法就可获取数据。

2. Winsock 控件的主要属性、方法和事件

属性：	Protocol = sckTCPProtocol	使用 TCP 协议进行连接；
	RemoteHost	准备连接的服务器 IP 地址或计算机名
	RemotePort	连接服务器的 IP 端口号
	LocalPort	本地机监听 IP 端口号
方法：	Connect	申请连接远程机
	Listen	设置监听
	Accept	建立实际连接
	Senddata	发送数据
	Getdate	接受数据
	Close	关闭连接
事件：	Connectionrequest	一方请求连接时另一方产生
	Connect	一方接受连接时另一方产生
	Close	一方关闭连接时另一方产生
	DataArrival	一方发送数据另一方接受时产生
	Error	请求连接失败时产生

三、服务器端程序设计

1. 新建一窗体，属性设置如下：Name 为 frmservers，Caption 设置为 ‘服务器 ”，visible 为 True。

2. 在 frmservers 窗体上添加两个 Winsock 控件。第一个 Winsock 控件作用是监听来自客户机的连接信号。属性设置如

下：Name 为 Winsocka，Protocol 设置为 0 - sckTCPProtocol。第二个 Winsock 控件 它的作用是传送与接收信息。属性设置如下：Name 为 Winsockb，Index 设置为 0，创建一个控件数组，Protocol 设置为 0 - sckTCPProtocol。

3. 在 frmservers 窗体上再加一个 CommanButtond 控件，它的作用是在程序运行时单击该命令按钮可以关闭服务器，Name 为 Command1，Caption 属性为 “关闭”。

4. 在 frmservers 窗体上再加一个 TextBox 控件 Name 为 Tex1 Text 属性为 “b”。用于测试客户机的连接信号。

5. 代码编写：

```
Option Explicit
Const maxs = 50
Dim User(maxs) As Boolean
Private Sub Winsockb_DataArrival(index As Integer, ByVal
bytesTotal As Long)
Dim str As String
Dim i As Long
Winsockb(index).GetData str
Text1.Text = Text1.Text & str
  For i = 1 To maxs
   If User(i) And Text1.Text = "b5" Then
        Beep  '音响是为了实验检验效果
        Winsockb(i).SendData ("B")
   End If
   Text1.Text = "b"
  Next i
End Sub
Private Sub Command1_Click()
  End
End Sub
Private Sub Form_Load()
  Winsocka.LocalPort = 80
  Winsocka.Listen
  Text1.Text = "b"
End Sub
Private Sub Winsocka_ConnectionRequest(ByVal requestID
As Long)
Dim i As Long
  For i = 1 To maxs
    If Not User(i) Then
      User(i) = True
      Exit For
    End If
  Next i
  If i > maxs Then
     Exit Sub
  End If
  Load Winsockb(i)
  Winsockb(i).Accept requestID
  Text1.Text = "b"
  Winsockb(i).SendData ("B")
End Sub
Private Sub Winsockb_Close(index As Integer)
   Winsockb(index).Close
   Unload Winsockb(index)
   User(index) = False
End Sub
```

四、客户端程序设计

1. 新建一窗体，属性设置如下：Name 为 frmclient，Caption 设置为 “客户机”，visible 为 True。

2. 在 frmclient 窗体上添加一个 Winsock 控件。它的作用是传输连接信号和接收来自服务器的通讯密码和数据。属性设置如下：Name 为 Winsockc，Protocol 设置为 0 - sckTCPProtocol。

3. 在 frmclient 窗体上再加两个 CommanButtond 控件。第一个 CommanButtond 控件的作用是在程序运行时单击该命令按钮开始发送 Connect 信号，申请连接服务器，Name 为 Fasong，Caption 属性为 “发送”。 第二个 CommanButtond 控件的作用是在程序运行时单击该命令按钮关闭客户机，退出运行，Name 为 tuichu，Caption 属性为 “退出”。

4. 在 frmclient 窗体上再加一个 TextBox 控件 Name 为 Tex1 Text 属性为 。用于测试服务器的连接信号。

5. 在 frmclient 窗体上再加两个 Timer 控件。第一个 Timer 控件的作用是在程序运行时定时发送 （$enddata）信号，与服务器联系，Name 属性为 Timer1，Enabled 属性为 True，Interval 属性为 30000。第二个 Timer 控件的作用是在程序运行时，定时测试服务器的连接信号，如连接信号错，则强行关闭客户机，退出运行，Name 属性为 Timer2，Interval 属性为 40000，Enabled 属性为 True。

6. 代码编写如下：

```
Option Explicit
Private Sub Winsockc_Close()
  Dim Xinghao1 As Integer
  Xinghao1 = MsgBox("服务器关闭或网络连接出错!程序结
束!", vbOKOnly + vbExclamation, "系统提示")
    If Xinghao1 = vbOK Then
      Timer1.Enabled = False
      Timer2.Enabled = False
      Text1.Text = ""
      fasong.Enabled = False
       Winsockc.Close
      Unload Frmclient
      End
    End If
End Sub
Private Sub winsockc_Connect()
  Timer1.Enabled = False
  Timer2.Enabled = False
  Winsockc.SendData "5"
   Timer2.Enabled = True
End Sub
Private Sub Winsockc_DataArrival(ByVal bytesTotal As Long)
 Dim str1 As String
 Dim Answer2 As Integer
  Winsockc.GetData str1
  Text1.Text = str1
  If Text1.Text <> "B" Then
   Answer2 = MsgBox("服务器关闭或网络通讯出错!程序结
```

```
束!″, vbOKOnly + vbExclamation, ″系统提示″)
    If Answer2 = vbOK Then
       Winsockc. Close
       Text1. Text = ″″
       Timer1. Enabled = False
       Timer2. Enabled = False
       Unload Frmclient
    End If
  Else
    Text1. Text = ″″
    Timer1. Enabled = True
    Timer2. Enabled = False
  End If
End Sub
Private Sub Form_Load()
 Winsockc. Protocol = sckTCPProtocol
 Winsockc. RemoteHost = ″100. 100. 100. 28″
′这儿请输入您的服务器的 IP 地址, 或计算机名
 Winsockc. RemotePort = 80
End Sub
Private Sub Fasong_Click()
  On Error GoTo myerror
    fasong. Enabled = True
    Text1. Text = ″″
    If Winsockc. State = 0 Then
      Winsockc. Connect
    End If
    Timer1. Enabled = True
    Exit Sub
myerror:
MsgBox ″程序出错!″, vbOKOnly + vbQuestion, ″系统提示″
   Exit Sub
End Sub
Private Sub Timer1_Timer()
  If Winsockc. State = 0 Then
    Winsockc. Connect
  Else
    Winsockc. SendData ″5″
  End If
    Timer2. Enabled = True
End Sub
Private Sub Timer2_Timer()
  Dim Answer3 As Integer
  If Text1. Text <> ″B″ Then
    Answer3 = MsgBox(″服务器关闭或网络通讯出错!程序
结束!″, vbOKOnly + vbExclamation, ″系统提示″)
    If Answer3 = vbOK Then
       Winsockc. Close
       Text1. Text = ″″
       Timer1. Enabled = False
       Timer2. Enabled = False
       Unload Frmclient
    End If
  Else
    Text1. Text = ″″
    Timer1. Enabled = True
  End If
End Sub
Private Sub Tuichu_Click()
  End
End Sub
```

五、结束语

以上程序在 VB6. 0 中文企业版、中文 Win98、P Ⅱ400、内存 32M 环境中编译通过。在集线器为 ACCTON、网卡 ACCTON1688、41 台微机组成的局域网上能正常通讯运行。如需扩充客户机，可适当增加 TIMER. Interval 值和 Maxs 值。本实例为了能方便演示客户机/服务器间的通讯控制，省去了软件狗，并显式运行，在实际与应用软件结合运用时，只需再加一些代码，与应用软件“绑定”，并将 Form. Visible 值设为 false，可实现隐式加密效果。

参考文献

1. 陈华生等 . Visual Basic 程序设计教程 . 苏州大学出版社

2. 汪保华 . 用 VB 实现聊天讨论室和点对点对话 . 电脑编程技巧与维护 1999 (0)

收稿日期：2001 年 2 月 5 日）

畅通无阻 III》——开拓你的网上新世界

金洪恩公司推出影响中国近千万网民网络生涯的 畅通无阻》的完全升级版—— 畅通无阻 III》，带给大家最前沿的 Internet 的知识，让您全方位学习网页设计与制作，让你的 Internet 观念和水平一起升级。

继承洪恩软件高效实用的人性化设计特点，独创实用的‘边学边练”，让知识迅速转化为您工作和效率，让您更充分地利用网络上的丰富资源。

畅通无阻 III》从最基本的浏览网站讲起，详细而又通俗易懂地讲解了电子邮箱、搜索引擎、信息查询、软件下载、网上聊天、网上寻呼、网上论坛 (BBS）、网址精粹等诸多网络的基本用法。

对于上网已经有一段时间的网民，则讲解了深度较高的网络安全、新闻组、游戏娱乐、网上炒股、电子商务 网上购物、网上拍卖、网上支付）、网络电话等知识。

畅通无阻 III》详细而又深刻地讲解了网页制作。从 DreamWeaver 和 FrontPage 的网页设计与制作 (上篇）的实例讲起，详解这两大工具的最新版本的使用方法、特点和技巧。

在掌握网页基础制作的基础上，教您使用 Fire-Works、Photoshop 等美术设计工具。

网页制作的高级应用，可以让您迅速成为网络工程师。如何编写 HTML 源代码，如何巧妙利用 CSS，如何在网页中使用 JavaScript 特效。另外还介绍了动态网页制作 CGI / ASP / PHP / JSP 的概念。最后讲解了申请免费个人主页空间和上传主页的过程，使您的努力在 Internet 上大放光彩。

博士信箱 QUESTION&ANSWER

微机系统内存的合理使用与常见问题解答

在实际使用中对微机系统内存的基本要求是什么？

计算机内的存储器按其用途可分为主存储器（Main Memory，简称主存）和辅助存储器（Auxiliary Memory，简称辅存），主存储器又称内存储器（简称内存），辅助存储器（简称外存）。

内存实质上是一组或多组具备数据输入输出和数据存储功能的集成电路。内存按存储信息的功能可分为只读存储器（Read Only Memory）、可改写的只读存储器EPROM（Erasable Programmable Memory）和随机存储器RAM（Random Access Memory）。我们平常所说的内存是指RAM，其主要作用是存放各种输入、输出数据和中间计算结果，以及与外部存储器交换信息时作缓冲作用。由于CPU只能直接处理内存中的数据，所以内存的速度和大小对计算机性能的影响是相当大的。

在实际使用中，首先系统内存的速度必须不小地系统总线也就是CPU的外频的速度，否则轻则电脑工作不稳定，重则根本无法启动。正是由于内存的速度相同于CPU来说太慢，所以内存厂商都在努力推出速度更快的新产品。目前最被看好的是RDRAM和DDR RAM。前者速度很快且得到了Intel大力支持（配合Pentium 4的i850芯片组就支持RDRAM），但价格太高，后者相对便宜，性能比RDRAM稍差。其次系统内存容量要尽可能的大。一旦内存数量太少，Windows就会使用速度比内存还要慢上一个数量级的硬盘做虚拟内存，将会使系统的整体速度大大降低，所以内存一般都应该配置128MB。

SDRAM有什么特点

SDRAM Synchronous DRAM，即同步DRAM，顾名思义就是与系统时钟同步工作的动态存储器。系统时钟除了控制CPU外，还要控制SDRAM的速度。理论上讲，它可以实现与CPU同步工作。SDRAM采用Multiple - Memory - Banks设计，存取效率成倍提高。当前市场上出售的SDRAM内存主要有两种：一种是工作速度66MHz的产品（即PC - 66），另一种是工作速度为100MHz的产品（即PC - 100），自1998年下半年以后，PC - 100已占主导地位。SDRAMⅡ是在SDRAM的基础上发展起来的新型内存产品，正式名称应该是DDR SDRAM（DDR是Double - Data - Rate双倍数据速度）的缩写。作为SDRAM的第二代产品，它遵循的是Jedec标准。单从内核看，它与SDRAM并无根本区别，所不同的主要有两点：一是它使用了更为高级的同步电路，并允许数据在Rising（上升）与Falling（下降）两个边缘触发，因而其频带宽度是SDRAM的两倍。二是采用了DLL（Delay Locked Loop，即延时锁定回路）技术。SDRAMⅡ更适合在100MHz系统上运行。目前已有100MHz（200Mbps）、125MHz 250Mbps、133MHz 266Mbps三种产品进入市场，1999年中期实用化产品大规模投放市场。当然，要使用DDR SDRAM，主板芯片组能否支持是个先决条件。

SLDRAM有什么特点

这里的SL是Synchronize Link（同步链）的缩写。SLDRAM在增加了更为先进同步电路的同时，还改进了逻辑控制电路，因而其总体表现与RDRAM不相上下。令人欣慰的是，Micron公司已推出了一种容量为64MB的商品化产品，它采用0.25微米技术制造，频率达到400MHz，据称数据传输率可达到1.6GB/s。

RDEAM有什么特点

RDEAM Rambus DRAM是美国RAMBUS公司开发的一种存储器。它的管道存储结构支持交叉存取，同时可执行四条指令。单从封装形式上看，它与SDRAM没有什么不同，在发热量方面它与PC - 100的SDRAM大致相当。为加速其成为工业标准，该公司将核心技术转让给了日本的东芝、富士通及NEC三家公司。如今。除了上述三家外，又有三星、日立、Kingston、现代、IBM等取得了该产品的授权。如今，RDEAM已形成两个分支：一是Concrrent RDRAM；另一个是Direct RDRAM。前者的传输率为600MB/s，后者峰值传输率可达到1.6GB/s。

CDRAM有什么特点

CDRAM Cached DRAM是三菱公司的专利技术，它实际上是指在DRAM芯片的外部插针和内部DRAM之间，插入一个SRAM作为二级Cache使用。虽然几乎所有的CPU都装有一级Cache来提高效率，但随着CPU时钟频率的成倍提高，Cache命中率低对系统性能产生的影响将会越来越大。而Cached DRAM所提供的二级Cache正好用以补充CPU L1 Cache的不足，因此能极大地提高CPU效率。

何为快闪存储器

快闪存储器Flash Memory是一种内存芯片。它是一种非易失性的存储介质，加电时才能进行存储和擦写。这一点明显不同于普通内存，普通内存一旦失去电源供应，便会在瞬间失去所有数据，而快闪存储器必须加大电压才能进行擦除，否则，即便没有电源的支持，数据也会存储在介质之中。快闪存储器非常适合耗电量小的设备

需求，如 PDA 等掌上产品。

快闪存储器的另一个特点就是：它是一种超大规模集成电路，不是简单的磁介质或者光学存储器。这决定了它能够在很小的空间内存储大量的信息。换句话说，它的信息密度很高，体积非常小，通常我们能够看到的 4MB、8MB 的产品，往往只有信用卡那么大，而且仅有几毫米厚。

当然，我们还看中快闪存储器的坚固性。以 KingMax 快闪存来说，它能够承受高达 2000GB 的过载冲击。这种耐高温的坚固结构（使用环境通常是 45 ~85 ，大大高于硬盘的安全使用温度）使得闪存卡非常适合携带，无论是户外数据采集还是恶劣气候工作，甚至是野战环境，它都能高效而安全地完成数据存储。

PDA、MP3 随身听和掌上电脑、数码相机都是快闪存储器主要的“客户”。

衡量内存条技术的主要指标是什么

衡量内存条技术的一个重要指标是 DRAM 芯片的存取时间（通常用纳秒 ns 表示）数值越小，速度越快。因此内存条产品的更新往往是通过其芯片技术的革新来缩短存取时间和提高内存访问周期效率的。FP DRAM（快页模式）与 EDO DRAM 占据原 PC 内存条的大部分市场，但随着电脑进入奔腾时代，SDRAM 已作为内存条最新技术要求取代了前两者在新型号电脑上的位置。SDRAM 即同步 DRAM，也就是同 CPU 定时同步的 DRAM 技术，它可以高达 100MHz 的速度传递数据，是标准 DRAM 的 4 倍，其性能也比 EDO 内存条提高 30%。随着电脑科技的发展，SDRAM 不久将成为内存条上的主要产品，以适应如多媒体、服务器、数字、ATM 转换器与其他需要高带宽与快速传输率的网络化、通信软件的需求。

在内存条模块生产技术上，新型的 168 线 DIMM 内存条模块成为新时期内存条的主流。DIMM 指双在线模块，与 SIMM 在线模块有很大区别，其中 168 线 64 位 DIMM 模块在长度增加不多的情况下将模块的总线宽度增加一倍。

SDRAM 内存条比 EDO DRAM 内存条有何改进

由于 SDRAM 采用的是双存储体结构，这使得其存储效率有成倍提高。有资料表明，SDRAM 作内存的运行速度比 EDO DRAM 内存至少快 13%。从价格上看，同为 32MB 的内存条，SDRAM 的性能价格比更高一些。要注意的是，SDRAM 有单面条与双面条之分。前者是只有一面有存储芯片，后者是两面都有存储芯片。按规律，16MB、64MB 都为单面条，而 32MB、128MB 都为双面条。

内存条上的 SPD 的作用是什么

SPD 是位于 FCB（印刷电路板）上的一个约 4mm 见方的小芯片，它实际上是一个 256 字节的 EEPROM。里面保存着内存条有关的数据（如一些设置、模块周期信息），同时负责自动调整主板上的内存条速度。如果主板支持 SPD，那么开机后就可以在 BIOS 的 DRAM 设定时看到有关 SPD 设置。当设为 AUTO 时，一些相关设置（包括 CAS 等待时间）就会由 SPD 来设定。选用了带有 SPD 的 SDRAM，我们无需通过主板的 BIOS 进行手工设定。不过还是建议不使用 SPD 进行设定，原因是个别厂商为了降低内存条的生产成本，根本不装 SPD ，或焊上一片空的 SPD。这样一来，有可能导致 100MHz 以上外频不能正常工作，而降频使用则会造成浪费。这对于喜欢超频的用户，只要可超性好，SPD 中的数据并不重要。值得注意的是，有个别厂商的主板一定要 BIOS 检测到 SPD 中的数据才能工作，对此选购时要留意。

内存条由哪几部分组成

一个内存条由内存芯片和印刷电路板及其他的一些元件组成，因此芯片的质量只是内存质量的一部份，印刷电路板的质量也是一个重要的决定因素。同是“现代”的内存，大家却会觉得质量大不一样，这就是印刷电路板质量不同的原因。内存芯片是整条内存的核心。

内存条的 CAS 是什么意思

CAS 等待时间的定义是，一个读命令在时钟上升沿发出数据到输出端可以提供的时延。这个值一般设为 2 或 3 个时钟周期。在同等工作频率下，内存条 CAS 为 2 时要比为 3 时的速度快。一般意义上讲，CAS 等待时间决定了 SDRAM 内存条的性能，并对系统工作速度有很大影响。

电脑系统内存的主要品牌有哪些

主要品牌如下：

（1）现代系列

目前市场上的内存条共有 3 种，一种编号为 T7J，这是一种 PC100 内存，它的 CL 数值可以达到 2，速度为 10 纳秒；另一种编号为 TJK，它的性能和 T7J 差不多，只是 CL 为 3；T75 则是一种 PC133 内存，速度为 7.5 纳秒，CL 为 3，现代内存条的兼容性以及电气性能都不错，现代的 T75 PC133 内存就是理想的选择。

（2）KINGMAX 内存

KINGMAX 内存以其优秀的超频性能享有很好的口碑，KINGMAX 内存的芯片采用 TINY BGA 封装，它可以提高内存的稳定性，减少电信号干扰。这种内存工艺独特，所以基本没有假货。目前市场上销售的 KINGMAX 内存条全部为 PC133 标准。

（3）其他内存条产品

在市场上，现代和 KINGMAX 内存条几乎包办了整个内存市场，除它们之外，我们还可以见到一些樵风内存条、三星原装条、Acer 原装条、小影霸超频内存条、KTI 内存条等产品。樵风内存条很有特色，，它采用 BLP 封装形式，外形和 KINGMAX 的产品差不多，可以稳定地运行在 143MHz 频率下。三星和 Acer 内存条都是号称原厂生产，不过价格都比较贵。小影霸超频条是磐英公司推出的一种发烧型内存条，采用三星的芯片，CL 为 3，专门用于超频。而 KTI 内存条比较独特，它是由生产 KINGMAX 内存条的厂家 OEM 生产的，速度为 8纳秒，是 PC100 标准的，据说可以跑到很高的频率。