电脑编程技巧与维护月刊

2001 年第 2 期　　总第 80 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 62178333
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
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

本刊 2000 年优秀作者

冉林仓　郑州铁路分局机电设备厂
江天送　桂林电子工业学院 990121 班
江　华　吉林市师范学校
江　龙　湖北浠水师范学校微机室
邓双成　北京石油化工学院机电教研室
元凯宁　天津市河东区程林庄路 63 号 182 信箱
李震宇　合肥市经济信息中心计算机通讯部
余海燕　北方交通大学自动化所

本刊 2000 年热心读者

刘佩军　湖南省常德华南光电仪器厂退休办
何春华　浙江省海宁市人民路 169 号内海宁佳和电子工程有限责任公司
张　钟　重庆谢家湾建设工业　集团）公司摩研所
王　涛　黄金堂路 22 号湖北莉州市职业中专
江　帆　福建省光泽华桥国有林场
吴树鹏　天津市河东区万新村七区防暴支队
孙　凯　湖北工学院 73#
田伟蒙　陕西西安市金花北路 4#西安工业学院测量与控制研究所
华　磊　安徽合肥市中国科学技术大学东区 222 - 115
肖忠良　长沙电力学院化学系
唐翊国　山东省东营石油大学炼制系炼油教研室
张鸿斌　北京市 5102 信箱 81 号
蒋东焱　北京市 9200 信箱 74 分箱 19#
宋曜利　西北工业大学 754 信箱
姜忠奎　辽宁东港市前阳镇建筑安装总公司
陈　强　青海省格尔木市水文分站
徐东文　山西省万家寨引黄工程电信站
陈　峰　山大新校 64#
陆　涛　宁夏青铜峡市汉坝小学四楼计算机室
邹增理　浙江大学华家池校区环建 961 班
陈红根　河南省职业技术学院计算机系
李淑琼　深圳市华城暨南大学中旅学派 2#
郑　宏　福建省平和一中高三　仁）班
由世洲　黑龙江水运规划设计院
周京生　北京 5131 信箱 12 号北京跟踪与测量总体技术研究所八室
熊　伟　江苏南京市 83582 部队服务中心
李　均　江西省景德镇供电局
刘红强　西北工业大学 112 信箱

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	盖章 年　月　日		
单价	7.80 元/期	份数		合计			
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

读者意见征询卡

尊敬的读者,请您抽空填写此表,并邮寄或传真至我社。

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

个人资料:

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

5. **电话:**_____　6. **传真:**_____　7. **单位:**_____

8. **地址:**_____　9. **邮政编码:**_____　10. ICQ_____

11. E - mail:_____　12. **个人主页:**_____

13. **文化程度:**□**中专以下**　□**大专**　□**本科**　□**研究生以上**

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

15. **您主要买哪些电脑类报刊(请按喜欢顺序填写)**

报纸:1 _____　2 _____　3 _____

杂志:1 _____　2 _____　3 _____

16. **您购买或订阅本刊的时间:**

□**两年以上**　□**一年以上**　□**半年以上**　□**三期**　□**两期**　□**第一次**

17. **您第一次购买本刊的原因是:**

□**朋友推荐**　□**刊名吸引**　□**内容吸引**　□**价格吸引**

18. **您获得本刊的渠道:**□**邮购**　□**订阅**　□**购买**

19. **除了您阅读本刊外是否还传阅他人?**　□**是**　□**否**

20. **您认为本刊跟其他刊物相比的特点是**_____

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

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

23. **您希望本刊增加的内容是:** _____

_____

24. **您对本刊的内容和发行您还有什么建议?** _____

_____

新加坡开发出模拟亚洲人眼球软件

新加坡学者最近开发出了一套软件，它可模拟亚洲人的眼球给医学院的学生作手术之用。该软件的问世解决了医学院学生想作眼部手术而又找不到足够的动物眼球的麻烦。南洋技术大学的一个小组开发了这套软件，目的是希望医学院的大三学生利用软件虚拟现实的功能，在人眼的图象上作手术练习。该大学的才易愚 音译 教授说，传统上，医学院的学生是用动物的眼球代替人眼球来作练习之用的，但动物眼球价格昂贵，手术一旦作坏又得换新的眼球；而且动物的眼病与人的眼病也有差异，所以动物的眼球并不是首选。据悉，在美国，类似的眼部手术训练软件早被广泛使用，但模拟的眼球不是亚洲人的眼球。新加坡学者开发的这套软件模拟的则是亚洲人的眼球。

微软在机器辅助翻译领域取得重大突破

日前，微软中国研究院透露，该院致力于新一代自然语言处理技术，特别是中文信息处理的研究，并且在机器辅助翻译领域取得了重大突破，这将给不谙英语者上网带来福音。据介绍，该项技术不仅局限于单字或单词的翻译，而是准确地提供短语级译文。它甚至可以对文本进行总结，给出摘要和近似的翻译，从而使用户不需要阅读、翻译整篇文章便可以了解大概内容。对于具备一定英文基础的使用者，该技术能有效地帮助他们发现并改正写作中的语法错误，同时，系统自动提供的范文及例句也能帮助作者润色文章，拓展写作思路。另外，在中文校对系统方面，自然语言组也实现了技术上的重要进展，他们提出了 “全局求优的语言模型”，针对每个输入的汉字进行混淆级扩展识别，分辨出错误的输入，并自动进行修改。

英特尔称部分 Linux 无法在奔 4 电脑上安装

日前，英特尔公司的人员指出，一些已上市的基于 Linux 的软件无法安装于奔腾 IV 系统的计算机，出现这一现象的原因是软件无法正确识别奔腾 IV 处理器。英特尔公司认为，这是由于个别 Linux 软件公司没有及时更新操作系统所致。现在只有红帽子 Red Hat Linux7. 0 和 Turbo linux 公司的 Turbo Linux 6 能与英特尔的新芯片兼容。其它一些重要 Linux 发布商，如：Caldera、Corel、MandreketSoft 和 SuSE 等公司则没有在其软件内的 ‘芯片 ID 库”中加入识别奔腾 IV 的代码。

微软推出 “Audio 8”和 “Video 8”软件

美国微软于近日公布了 “Windows Media Audio 8”以及 “Windows Media Video 8”的测试版。Windows Media Audio 8 以及 Windows Media Video 8 是声音 / 图像数据流发行技术体系 Windows Media Technologies 中的压缩扩张技术 Codec 。据说在该公司实施的 Bench Mark 测试中，实现了到目前为止的最高品质以及压缩率。 ‘在 Video 8 中，压缩率比 Video 7 提高了 30% ” 微软 。为了实现高压缩率及扩张性能，Video 8 在 500Kbps 的因特网接驳速度下也可以发布接近 DVD 品质的视频信息数据流。Audio 8 则可在 48Kbps 的接驳速度下播放接近 CD 品质的音频信息。

微软将发表新版 Windows Media 软件

微软 Microsoft 将发表数种影音产品升级版 强调更快速的网络连结以及适用于口袋型电子产品上。产品的推出象征着微软持续向竞争对手 RealNetworks 施压。微软将在加州圣荷西的一项贸易展中发表展示其 Windows Media 的更新版。引述微软话说 Windows Media 更新版转换速度达能每秒 500 千位 KB 画质几可媲美 DVD 只读型数字多功能光盘片 部分宽频服务的用户已经得以使用此软件。微软表示 最新的 Windows Media Video and Audio 8 软件能够储存高音质录音档 所需档案空间仅为广受欢迎的 MP3 的三分之一 下载速度也较 MP3 快 60%。

Oracle 公布 WWW 服务开发技术框架

美国 Oracle 公司于近日公布了 WWW 服务开发技术框架 “Oracle9i Dynamic Services”，同时还宣布从即日起开始销售该技术产品。由于该产品可以立刻使用，所以它被称为 “. NOW ”。Oracle9i Dynamic Services 将同

“Oracle9i”数据库现行版本一起免费地提供给用户。在此框架下，开发者利用数据库服务器“Oracle9i Application Server”的功能，可以开发门户网站、无线因特网、内部网络等多种WWW服务。Oracle强调，Oracle9i Dynamic Services的独到之处是采用了XML和Java等最新的标准软件格式。今后，当UDDI Universal Description Discovery and Integration和SOAP Simple Object Access Protocol的最终格式完成以后，他们还准备采用这些格式。

Visa国际组织推出电子支付新方案

Visa国际组织近日宣布推出全球性新型方案Visa商务Visa Commerce。这是Visa长期战略不可或缺的组成部分，旨在为其会员机构提供安全的企业间电子支付解决方案。它将填补目前市场上的空白，实现企业间严密、安全的在线电子支付。Visa同时针对高额交易推出一种企业间互联网支付产品，该产品能够促进金融机构、企业和门户在电子商务领域进行严密、安全的电子支付。按照新的Visa商务战略，明年将向Visa会员银行推出一系列新产品以补充开放式帐户OpenAccount解决方案。据悉，Visa将与VCHEQ共同推出Visa商务解决方案。VCHEQ的支付网关将处理新型Visa商务开放式帐户产品。

IBM推出新一代功能强大的芯片CMOS9S

IBM日前宣布生产出用于服务器、通讯产品以及普及运算设备的新一代功能强大的芯片，这些芯片采用的是目前世界上最先进的芯片制造技术。这种新技术的名称为CMOS9S，它首次结合了IBM创新的铜芯片、绝缘硅片SOI晶体管以及具有低K值导电性的绝缘体的众多优势，生产出只有0.13微米的芯片电路，这个尺寸比人的头发丝还要细800倍。超微型的电路以及经过改进的材料能够将许多处理能力压缩在单一的芯片上，可以胜任从计算机到手机等电子产品对计算性能的急切需要，特别是在语音识别、指纹认证以及无线视频等方面的应用。目前多种采用IBM新制造工艺的芯片已经开始生产了，预计将在2001年早期面市。

新一代“生命电脑”今天出生

它可以为你朗读新闻、电子邮件，可以早晨7时叫你起床，可以帮你找到网上的朋友，因为它是一个有生命的精灵，而不只是台电脑。今天，联想发布“生命电脑”和第一代因特网电脑相比，这次发布的第二代电脑充满“生命化”设计，用语音技术、多媒体技术、人工智能技术建立了一个网络社区。用户在获得电脑的同时，就获得了一个网络世界，既可以在线交友、聊天、炒股，还可发送多媒体邮件、进行语音交流，甚至通过语音合成直接“听网”，通过摄像镜头和电视卡录制并编辑自己的节目，在线玩游戏、点歌。

德州仪器公司推出手掌大小的数字音响

德克萨斯州仪器公司TI近日推出了一种新型全数字化的音响系统，其电路板只有手掌大小。据称该设备尽管价格非常便宜但能为各种小型设备如便携式CD播放机、个人电脑和汽车上的立体声系统提供质量上乘的音响。该公司称这种新产品的关键部分是该公司开发出的一种数字化放大器，这种放大器可以取代那些既笨重而且噪音重的类似功能的放大器。德州仪器公司称，他们将这种新型数字放大器与现有的数字信号处理器DSP相结合，这种信号处理器也是德州仪器公司的核心产品，它能够将现实世界中的声音和图像转换成电脑能够处理和反馈的数字形式。全世界用在移动电话、便携式摄像机和调制解调器中的数字信号处理器大约有一半是由德州仪器公司提供的。

美国品尼高公司推出七款视频编辑软件

美国品尼高公司日前在国内推出七款视频编辑软件Studio PCTV、Studio PCTVUSB，可以让电脑用户在自己的计算机上看电视、编电影。Studio PCTV不仅可以接收电视节目、发送视频电子邮件，还可以进行视频信号的采集。通过易学易用的Studio视频编辑软件制作自己的影片，您可以轻松地对采集下来的影片进行剪辑、添加字幕、后期配音等。除了可以生成AVI文件格式外，还能生成MPGE文件格式和Realplay格式，使制作出的影片能够刻成VCD或进行网上传输。Studio PCTVUSB使用更方便，无须插卡，使利用个人电脑收看电视或者制作VCD轻松实现。拥有了该软件用户可以随时随地收看电视、编制精彩节目、捕捉精彩瞬间、创作出具有自己独特风格的作品不再是梦想。

新ATA规格——Serial ATA草案1.0版公开

负责制订与存贮设备连接接口规格“Serial ATA”的业界团体Serial ATA Working Group，于近日公布了该规格的草案1.0版。该标准可以在Serial ATA Working Group的WWW网站www.serialata.org上下载。Serial ATA Working Group早在去年11月21日就宣布完成了草案标准1.0版，并且开始向会员企业公布。此次则是首次向外界一般开发人员公布，以此来促进Serial ATA的普及。Serial ATA为串行接口的ATA规格。是用于将硬盘、DVD驱动器以及CD-RW驱动器等存贮设备连接到个人电脑上的接口。Serial ATA Working Group将其定位于取代ATA并行接口的新一代接口。Serial ATA的数据传输速度最初为1.5Gbit/s“Ultra SATA/1500”，以后将逐步扩充到倍速以及4倍速。

VC + + 系列讲座（二）

徐 亮 张博强

第二讲 MFC 入门

本讲将介绍 MFC 的知识，介绍 MFC 常用的类和宏，重点学习 MFC 的消息机制。

一、 VC + + 的核心—MFC

类库是一个可以在应用中使用的相互关联的 C + + 类的集合。微软基础类库（MFC：Microsoft Foundation Class）是微软为 Windows 程序员提供的一个面向对象的 Windows 编程接口，它大大简化了 Windows 编程工作。该层次结构包容了 Windows API 中的用户界面部分，并使你能够很容易地以面向对象的方式建立 Windows 应用程序。这种层次结构适用于所有版本的 Windows，并彼此兼容。

基础类库的核心是以 C + + 形式封装了大部分的 Windows API。类库表示窗口、对话框、设备上下文、公共 GDI 对象如画笔、调色板、控制框和其他标准的 Windows 部件。这些类提供了一个面向 Windows 中结构的简单的 C + + 成员函数的接口。

MFC 可分为两个主要部分：基础类、宏和全程函数。

1. MFC 基础类

MFC 中的类按功能可划分为以下几类：

基类

Cobject MFC 在部分类都是由 Cobject 类派生的

应用程序框架类

又分为：

●应用和线程支持类：包括

CwinApp 封装了初始化、运行和结束应用的代码。可由它派生应用类。

CwinThread 所有线程的基类。

●ISAPI 应用类：

●同步对象类：

命令处理类

文档 / 视类

包括

Cdocument 特定应用程序文档的基类。

可视对象类

Cview 所有视的基类。

CeditView 基于 Windows 编辑框控件的类。

CrecordView 在控制中显示 ODBC 数据库记录的表单视图类。

框架窗口类

包括

CMDIFrameWnd MDI 应用程序的主框架窗口的基类。

CMDIChildWnd MDI 应用程序的文档框架窗口的基类。

CframeWnd SDI 应用程序主框架窗口的基类。

对话框类

包括

Cdialog 所有模式或非模式对话框的基类。

控制类

菜单类

设备描述表

输出类和绘画对象类

包括

CwindowDC 用于整个窗口的显示器描述。

Cpen 封装了 GDI 画笔，可作为设备环境的当前画笔用来被绘制图形对象的边线。

Cfont 封装了 GDI 字体，可作为设备环境的当前字体来选择。

文件 I / O 类

文件 I / O 类给传统磁盘文件、内存文件、活动流和 Windows 套接字提供了接口。所有由 Cfile 派生的类可以被 Carchive 对象用于执行串行化。又分为：

●Cfile 类

●其他文件 I / O 类

异常类

Cexception 创建一个 Cexception 对象。

集合类

CArray 构造任意数组的模板类。

数据库类

简单数据类型

2. 宏和全程函数

若某个函数或变量不是某个类的一个成员，那么它是一个全程函数或变量。Microsoft 基本宏和全程函数提供以下功能：

数据类型

运行时刻对象类型服务

诊断服务

异常处理

CString 格式化及信息框显示

消息映射

应用消息和管理

对象连接和嵌入 （OLE）服务

标准命令和 Windows IDs

全局函数以 “Afx ”为前缀，所有全局变量都是以 “afx ”为前缀，宏不带任何特别前缀，但是全部大写。常见的全局函数和宏有：AfxGetApp、AfxGetMainWnd、AfxMessageBox、DEBUG_NEW 等。

从继承关系来看，又可将 MFC 中的类分成两大类：大多数的 MFC 类是从 CObject 继承下来；另外一些类则不是从 CObject 类继承下来，这些类包括：字符串类 CString、日期时间类 CTime、矩形类 CRect、点 CPoint 等，它们提供程序辅助功能。

由于 MFC 中大部分类是从 CObject 继承下来的，CObject 类描述了几乎所有的 MFC 中其他类的一些公共特性。 CObject 类为派生类提供了下述服务：

对象诊断

MFC 提供了许多诊断特性，它可以

输出对象内部信息：CDumpContext 类与 Cobject 的成员函数 Dump 配合，用于在调试程序时输出对象内部数据。

对象有效性检查：重载基类的 AssertValid 成员函数，可以为派生类的对象提供有效性检查。

运行时访问类的信息：

MFC 提供了一个非常有用的特性，它可以进行运行时的类型检查。如果从 Cobject 派生出一个类，并使用了以下三个宏 IMPLEMENT_DYNAMIC，IMPLEMENT_ DYNCREATE 或 IMPLEMENT_SERIAL 之一，就可以：

运行时访问类名

安全可靠的把通用的 Cobject 指针转化为派生类的指针

比如，我们定义一个主窗口类

```
CMyFrame: public CFrameWnd
{
. . . . . .
}
```

然后我们使用这个类：

```
CMyFrame ＊pFrame = (CMyFrame ＊)AfxGetMainWnd();
pFrame –>DoSomeOperation();
```

AfxGetMainWnd 是一个全局函数，返回指向应用程序的主窗口的指针，类型为 CWnd ＊，因此我们必须对它进行强制类型转换，使用 CObject 的 IsKindOf 成员函数检查 pFrame 的类型，用法如下：

```
ASSERT(pFrame –>IsKindOf(RUN_TIMECLASS
(CMyFrame)));
```

将上一语句插入到 pFrame - > DoSomeOperation 之前，就可以在运行时作类型检查，当类型检查失败时，引发一个断言 ASSERT ，中断程序执行。

二、MFC 对消息的管理

Windows 消息的管理包括消息发送和处理。为了支持消息发送机制，MFC 提供了三个函数：SendMessage、PostMessage 和 SendDlgItemMessage。而消息处理则相对来说显得复杂一些。MFC 采用了一种新的机制取代 C 语言编程时，对 Windows 消息的 Switch / Case 分支简化了 Windows 编程，使程序可读性、可维护性大大提高。

MFC 对消息的处理

MFC 不使用 switch / case 语句，而采用一种消息映射机制来决定如何处理特定的消息。这种消息映射机制包括一组宏，用于标识消息处理函数、映射类成员函数和对应的消息等。其中，用 afx_msg 放在函数返回类型前面，用以标记它是一个消息处理成员函数。类若至少包含了一个消息处理函数，那么还需要加上一个 DECLARE_MESSAGE_MAP 宏，该宏对程序执行部分所定义的消息映射进行初始化。清单 2. 3 演示了消息处理函数的例子：

```
class CMainFrame: CFrameWnd{
public:
CMainFrame();
protected:
//{{AFX_MSG(CMainFrame)
afx_msg int OnCreate(LPCREATESTRUCT lpCreateStruct);
afx_msg void OnEditCopy();
afx_msg void OnClose();
//}}AFX_MSG
DECLARE_MESSAGE_MAP()
};
```

成员函数 OnCreate、OnEditCopy、OnClose 分别用来处理消息 WM_CREATE、 ID_EDIT_COPY 和 WM_CLOSE。其中，WM_CREATE 和 WM_CLOSE 是系统预定义消息，包含在 Windows. h 中。而 ID_EDIT_COPY 是菜单 Edit - >Copy 的标识，也就是用户选择 Edit - >Copy 菜单项时产生的消息，一般在资源文件头文件中定义。在类的实现部分给出这三个成员函数的定义，以及特殊的消息映射宏。上面的例子的消息映射宏定义如下：

```
BEGIN_MESSAGE_MAP(CMainFrame, CFrameWnd)
ON_WM_CREATE()
ON_COMMAND(ID_EDIT_COPY, OnEditCopy)
ON_WM_CLOSE()
END_MESSAGE_MAP()
```

消息映射宏由 BEGIN_MESSAGE_MAP 和 END_MESSAGE_MAP 。其中，BEGIN_MESSAGE_MAP 宏包含两个参数 CMainFrame 类和 CFrameWnd，分别代表当前定义的类和它的父类。在 BEGIN_MESSAGE_MAP 和 END_MESSAGE_MAP 之间，包含了主窗口要处理的各个 Windows 消息的入口。在本例中，包含三个消息。其中 ON_ WM_CREATE 被用来指定缺省的成员函数 OnCreate 与 WM_CREATE 相对应。在 MFC 中，包含了大量的预定义消息映射宏，用来指定各种成员函数与各种

形式如 WM_XXXX 消息相对应。如 ON_WM_CLOSE 宏指定了 WM_CLOSE 消息的处理成员函数为 OnClose。这时侯，只需要写出要处理的消息就够了，不必再写出处理函数。消息映射宏 ON_COMMAND 则被用来将菜单项和用户自定义的命令同它们的处理成员函数联系起来。在上例中，用户选择 Edit - >Copy 菜单项时，系统执行 OnEditCopy 函数。ON_COMMAND 宏的一般定义形式如下：

ON_COMMAND command command_function

其中，command 为菜单消息或用户自定义消息，command_function 为消息处理函数。MFC 允许用户自定义消息，常量 WM_USER 和第一个消息值相对应，用户必须为自己的消息定义相对于 WM_USER 的偏移值，偏移范围在 0 ~ 0x3FFF 之间，这对绝大多数程序来说都是够用的。用户可以利用#define 语句直接定义自己的消息 如：

```
#define WM_USER1 (WM_USER +0)
#define WM_USER2 (WM_USER +1)
#define WM_USER3 (WM_USER +2)
```

为了说明如何使用用户自定义消息，我们看一个例子：

```
#include <afxwin. h>
#define CM_APPLE (WM_USER +0)
#define CM_ORANGE (WM_USER +1)
class CMainFrame: CFrameWnd{
public:
CMainFrame();
protected:
afx_msg int OnCreate(LPCREATESTRUCT lpCreateStruct);
afx_msg void OnClose();
//handle user select apple
afx_msg LRESULT CMApple(WPARAM wParam, LPARAM lParam);
//handle user select orange
afx_msg LRESULT CMOrange(WPARAM wParam, LPARAM lParam);
DECLARE_MESSAGE_MAP()
};
```

相应的消息映射如下

```
BEGIN_MESSAGE_MAP(CMainFrame, CFrameWnd)
ON_WM_CREATE()
ON_MESSAGE(CM_APPLE, CMApple)
ON_MESSAGE(CM_ORANGE, CMOrange)
ON_WM_CLOSE()
END_MESSAGE_MAP()
```

第一个 ON_MESSAGE 宏用于指定 CM_APPLE 命令消息的处理成员函数为 CMApple ，而第二个 ON_MESSAGE 宏用于指定 CM_ORANGE 命令消息的处理函数为 CMOrange。

消息的发送

Windows 应用程序允许应用程序向自己发送消息、向其他应用程序发送消息，甚至可以向 Windows 操作系统本身发送消息 比如要求关闭操作系统或重新启动操作系统）。Windows 提供了三个 API 函数用于发送消息，这三个函数是：SendMessage、PostMessage 和 SendDlgItemMessage。

SendMessage 用于向窗口发送消息，该函数说明如下：

```
LRESULT SendMessage(
HWND hWnd, //消息要发往的窗口的句柄
UINT Msg, //要发送的消息
WPARAM wParam, //消息的第一个参数
LPARAM lParam //消息的第二个参数
);
```

其中，hWnd 为接收消息窗口的句柄，参数 Msg 指定发送的消息，参数 wParam 和 lParam 依赖于消息 Msg。该函数调用目标窗口的窗口函数，直到目标窗口处理完该消息才返回。

PostMessage 函数同 SendMessage 类似，它把消息放在指定窗口创建的线程消息队列中，然后不等消息处理完就返回，而不象 SendMessage 那样必须等到消息处理完毕才返回。目标窗口通过 GetMessage 或 PeekMessage 从消息队列中取出并处理。PostMessage 函数说明如下：

```
BOOL PostMessage(
HWND hWnd, //消息发往的窗口
UINT Msg, //要发送的消息
WPARAM wParam, //消息的第一个参数
LPARAM lParam //消息的第二个参数
);
```

其中，参数 hWnd 为接收消息的窗口的句柄，参数 Msg 指定所发送的消息，参数 wParam 和 lParam 依赖于消息 Msg。

SendDlgItemMessage 函数用于向对话框的某个控制发送消息，函数声明如下：

```
LONG SendDlgItemMessage(
HWND hDlg, //对话框句柄
int nIDDlgItem, //对话框控件的 ID
UINT Msg, //要发送的消息
WPARAM wParam, //消息的第一个参数
LPARAM lParam //消息的第二个参数
);
```

其中，hDlg 为包含目标控制的对话框的窗口句柄，参数 nIDDlgItem 为接收消息的对话框控制的整数标识符，参数 Msg 指定了所发送的消息，参数 wParam 和 lParam 提供附加的特定消息的信息。

MFC 将这三个函数封装为 CWnd 类的成员函数，隐藏了窗口句柄和对话框句柄。这三个成员函数用于向本窗口发送消息，函数的说明如下：

```
LRESULT SendMessage( UINT message, WPARAM wParam = 0, LPARAM lParam = 0 );
BOOL PostMessage( UINT message, WPARAM wParam = 0, LPARAM lParam = 0 );
LRESULT SendDlgItemMessage( int nID, UINT message, WPARAM wParam = 0, LPARAM lParam = 0 );
```

消息映射结构只能用于 MFC。掌握它和如何在你的代码中应用它是很重要的。MFC 使用消息映射来解决虚拟函数的基本

问题。所有的消息映射到类的成员函数，这种直接消息到方法的映射对所有的消息都适用。它通过宏来实现消息到成员函数的映射,而且这些函数不必是虚拟的成员函数,这样不需要为消息映射函数生成一个很大的虚拟函数表 V表 ，就节省了内存和 CPU 的负担。

经过以上的学习，你的心中一定有了 MFC 概念，本讲最后我们来分析一个简单的例子，来加深对 MFC 的理解。程序清单如下：

```
//hello. cpp
#include <afxwin. h>
// 说明应用程序类
class CHelloApp : public CWinApp
{
  public:
    virtual BOOL InitInstance();
};
// 建立应用程序类的实例
CHelloApp HelloApp;
// 说明主窗口类
class CHelloWindow : public CFrameWnd
{
   CStatic * cs;
   public:
      CHelloWindow();
};
// 每当应用程序首次执行时都要调用的初始化函数
BOOL CHelloApp: : InitInstance()
{
   m_pMainWnd = new CHelloWindow();
   m_pMainWnd -> ShowWindow(m_nCmdShow);
   m_pMainWnd -> UpdateWindow();
   return TRUE;
}
// 窗口类的构造函数
CHelloWindow: : CHelloWindow()
{
   // 建立窗口本身
   Create(NULL,
         "Hello World!",
         WS_OVERLAPP
   // 建立静态标签 EDWINDOW,
         CRect(0, 0, 200, 200));
   cs = new CStatic();
    cs -> Create( " hello world", WS_CHILD | WS_VISIBLE |
SS_CENTER, CRect(50, 80, 150, 150), this);
}
```

上面的代码看过一遍，可以得到一个整体印象。该程序由六部分组成，每一部分都起到很重要的作用。

首先，该程序包含了头文件 afxwin. h 第 2 行 。该头文件包含有 MFC 中所使用的所有的类型、类、函数和变量。它也包含了其它头文件，如 Windows API 库等。

第 3 至 8 行从 MFC 说明的标准应用程序类 CWinApp 继承出了新的应用程序类 CHelloApp。该新类是为了要重载 CWinApp 中的 InitInstance 成员函数。InitInstance 是一个应用程序开始执行时要调用的可重载函数。

在第 10 行中，说明了应用程序作为全局变量的一个事例。该事例是很重要的，因为它要影响到程序的执行。当应用程序被装入内存并开始执行时，全局变量的建立会执行 CWinApp 类的缺省构造函数。该构造函数会自动调用在 18 至 26 行定义的 InitInstance 函数。

在第 11 至 17 中，CHelloWindow 类是从 MFC 中的 CFrameWnd 类继承来的。CHelloWindow 是作为应用程序在屏幕上的窗口。建立新的类以便实现构造函数、析构函数和数据成员。

第 18 至 26 行实现了 InitInstance 函数。该函数产生一个 CHelloWindow 类的事例，因此会执行第 27 行至 41 行中类的构造函数。它也会把新窗口放到屏幕上。

第 27 至 41 实现了窗口的构造函数。该构造函数实际是建立了窗口，然后在其中建立一个静态文本控制。

要注意的是，在该程序中没有 main 或 WinMain 函数，也没有事件循环。然而我们从上一讲在执行中知道它也处理了事件。窗口可以最大或最小化、移动窗口等等。所有这些操作都隐藏在主应用程序类 CWinApp 中，并且我们不必为它的事件处理而操心，它都是自动执行、在 MFC 中不可见的。

用 MFC 建立的每个应用程序都要包括一个单一从 CWinApp 类继承来的应用程序对象。该对象必须被说明成全局的 第 10 行 ，并且在你的程序中只能出现一次。

从 CWinApp 类继承的对象主要是处理应用程序的初始化，同时也处理应用程序主事件循环。CWinApp 类有几个数据成员和几个成员函数。在上面的程序中，我们只重载了一个 CWinApp 中的虚拟函数 InitInstance。第 3 至 8 行的代码建立了一个称为 CHelloApp 的类，它是从 CWinApp 继承来的 包含一个新的 InitInstance 函数。在重载的 InitInstance 函数内部，第 18 至 26 行，程序使用 CHelloApp 的数据成员 m_pMainWnd 来建立并显示窗口。InitInstance 函数返回 TRUE 表示初始化已成功的完成，如果返回了 FALSE，则表明应用程序会立即终止。

程序中 MFC 还定义了两个类型的窗口 1 框架窗口，它是一个全功能的窗口，可以改变大小、最小化、最大化等等 2 对话框窗口，它不能改变大小。框架窗口是典型的主应用程序窗口。

在代码 11—17 行中，从 CFrameWnd 中继承了一个新的类 ChelloWindow，它包括一个新的构造函数，同时还有一个指向程序中所使用的唯一用户界面控制的数据成员。

一个典型的应用程序将有一个主应用程序窗口。因此，CHelloApp 应用程序类定义了一个名为 m_pMainWnd 成员变量来指向主窗口。为了建立该程序的主窗口，InitInstance 函数

第 18 至 26 行 建立了一个 CHelloWindow 事例，并使用 m_pMainWnd 来指向一个新的窗口。但只建立一个简单的框架窗口是不够的。还要确保窗口能正确地出现在屏幕上。首先，代码必须要调用窗口的 ShowWindow 函数以使窗口出现在屏幕上 第 23 行 。其次，程序必须要调用 UpdateWindow 函数来确保窗口中的每个控制和输出能正确地出现在屏幕上 第 24 行 。其中，ShowWindow 和 UpdateWindow 函数是 CWnd 的成员函数，而 CFrameWnd 是从 CWnd 类继承来的，所以在这里我们可以使用它们。

代码第 22 行是初始化窗口。它调用 new 函数分配内存。这时，程序会调用 CHelloWindow 的构造函数。该构造函数在每次带类的事例被分配时都要调用。在窗口构造函数的内部，窗口通过调用 CFrameWnd 的 Create 成员函数 第 31 行 来建立。

程序中的最后一个对象是静态文本控件，程序在从 CFrameWnd 类中继承 CHelloWindow 类时 第 11 至 17 行 ，说明了一个成员类型 CStatic 及其构造函数。CStatic 构造函数是在为其分配内存时调用的，然后就调用了 Create 函数来建立 CStatic 控制的窗口。Create 函数所使用的参数与窗口建立函数所使用的参数是类似的 第 31 行 。第一个参数指定了控制中所要显示的文本内容。第二个参数指定了类型属性。类型属性在下一讲中将详细介绍。在我们使用的是子窗口类型 即在别的窗口中显示的窗口 ，它是可见的，文本的显示位置是居中的。第三个参数决定了控制的大小和位置。第四参数表示该子窗口的父窗口。已经建立了一个静态控制，它将出现在应用程序窗口上，并显示指定的文本。

这个程序很简单，利用 Wizard 几分钟就可以创建出该项目，它具有完整的结构，在以后的各讲中我们将进一步学习 MFC。

小结

通过本讲的学习，读者应掌握的知识：

1. 掌握 MFC 的概念，对于给出的一些常用的类，如 Cview、Cdocument、Cframewnd 等，应记忆。

2. 对 Cobject 类提供析服务要熟悉。

3. 要理解 MFC 的消息映射，它也是 Windows 编程的重要机制，是程序设计中的核心内容。

参考资料

1. Marshall Brain 著 张圣华译 . Visual C + + MFC 简明教程

2. 王晖等编著 . 精通 Visual C + + 6. 0. 电子工业出版社

http / / vcdynasty. yeah. net hArtical hVC + + _artical hvc1

收稿日期：2000 年 11 月 2 日）

纯软件反盗版概念

SentinelLM 许可证管理系统

SentinelLM 是由北京彩虹天地公司引进的一种纯软件方式的许可证软件保护系统。这套软件的最大特点就是引入了反盗版与软件发布相结合的新概念。目前，这套系统已经被世界上许多著名软件厂商实际采用。

SentinelLM 能够采集终端用户的计算机硬件信息，并将开发商的产品绑定在特定的计算机上。这样，只有经过授权的计算机可以运行指定的产品。SentinelLM 提供了工具软件 SentinelShell 和 API 函数，帮助开发商迅速地将许可证认证机制加入单机版或网络版的 Windows 或 UNIX 应用程序中。SentinelLM 也是目前行业中唯一支持 Sun Java client 的许可证管理系统。

利用 SentinelLM 不仅可以使开发商软件免受盗版侵害，同时它还会给开发商带来其它收益。通过 SentinelLM，开发商可以生成全功能的演示程序，实现 “先试后买 ” 销售模式，从而大大增加其软件产品的销售量。该销售模式首先为最终用户提供在一定时间范围内可以运行的演示软件。如果用户对该软件满意，付款后就可以得到永久许可证，其演示软件即变为正式软件，整个操作无需对软件本身进行重新安装。SentinelLM 通过提供电子软件销售和电子软件许可证授权功能使开发商从电子商务中获益。根据软件开发商软件的特点和最终用户的需要，开发商可以定义多种许可证类型。利用这套系统中提供的 SentinelLM - Shell，开发商可以迅速地把许可证机制加入 Windows 32 位应用程序执行文件中，无需对源程序做任何修改。同时，如果开发商选择 SentinelLM 的 API 调用，利用帮助文档，实现许可证机制同样简单快捷。开发商只要把彩虹天地提供的样本程序中相应的代码剪切下来后粘贴到自己的应用程序中，在一天之内就可以在其应用程序中实现许可证机制。因此，彩虹天地也是目前唯一不仅提供许可证 API 调用而且提供能够跟踪最终用户软件使用状况，生成使用报告工具的供应商。SentinelLM 支持所有的主要 UNIX 和 PC 平台，并支持基于 Java 的解决方案。

SentinelLM 集成了先进的反盗版技术和软件发行方式于一身，为开发商和用户同时带来方便和收益。对于开发商而言，使用这套系统，灵活、可靠、适用环境宽松、价格低廉；对于最终用户来讲，用户在选购软件时自由度更大，可以得到全功能的演示软件，购买成本更低，并且可以体验 “试用——购买——使用 ” 的全新销售途径。

在 VC + +中实现多国语言切换

唐文忠 吴志高

摘 要 本文详细介绍了在 VC + +中,用动态资源库(DLL)实现多国语言切换技术。利用这种技术可以使软件具有多种语言界面,扩大了软件的应用领域。

一、提出问题

在升级开发一个用 VC + + 开发的软件的过程中遇到这样问题:已有的软件是在英文 Windows 操作系统下开发,其应用界面是英文,现在升级开发需要改成中文界面,但又想保留原有的英文界面,使应用既有英文界面又有中文界面,即让用户可以按自己的喜好选择英文界面或中文界面。如何才能达到这个目的呢?解决这个问题的实质就是实现多国语言切换。

随着全球经济一体化和信息技术的发展,应用软件的本地化也具有了实际需求。一个应用软件如果能够根据不同的用户群提供不同文化的用户界面,无疑这个软件能占领更大的市场。因此在应用软件中实现多国语言切换也具有较普遍的意义。

二、解决方案

如何解决上述提出的问题?在 Windows 中一个应用的所有可见资源(如菜单、文字串、对话框、图标、位图等)均可集中存放在一个动态库(DLL)中,在应用运行的过程中可以动态地加载(用 AfxLoadLibrary())、指向(用 AfxSetResourceHandle())和释放(用 AfxFreeLibrary())资源库。这就可能使应用响应用户的不同要求,选择加载不同的动态资源库,并指向和调用这个库,从而实现多国语言切换。

三、具体实现

假定在 VC + + 6.0 环境下,把一个仅有英文界面的多文档 MDI 应用 Test,改变成具有中文/英文界面可切换的多文档应用。

1. 建立一个 MfC AppWizard(exe)应用工程 Test

进入 VC + +,选择菜单 File/New,选表单 Projects,选工程项 MfC AppWizard(exe),设置工程位置 x:\h(这里 x 为逻辑盘符),键入工程名 Test,点按钮 OK 进入 MFC AppWizard 导向。在 MFC AppWizard 导向的第 1 步中选择英文语言资源,点按钮 Finish - >OK 完成工程建立。

从工程 Test 的工作区(Workspase)中选表单 ResourceView,在菜单资源 IDR_TESTTYPE 和 IDR_MAINFRAME 的下拉菜单 View 中添加一个菜单项,设置其 ID 均为 ID_RES_SWITCH,其 Caption 均为 Chinese。

选择菜单 File/Save Workspace 保存工程 Test。

2. 在工程 Test 的目录下,添加一个新的 MfC AppWizard(dll)工程 EnglishRes

在工程 Test 中,选择菜单 Project/Add To Project/New,选表单 Projects,选工程项 MfC AppWizard(dll),选新建工作区,设置工程位置 x:\hTest,键入工程名 EnglishRes,点按钮 OK,进入 AppWizard 导向。在 AppWizard 导向的第 1 步中选择 Regular Dll,点按钮 Finish - >OK 完成工程建立。

从工程 EnglishRes 的工作区中选表单 FileView,删除工程 EnglishRes 下的所有文件项(保留文件夹)。选择菜单 File/Save Workspace 保存工程 EnglishRes。

3. 在工程 Test 的目录下,添加一个新的 MfC AppWizard(dll)工程 ChineseRes

选择菜单 Project/Add To Project/New,选表单 Projects,选工程项 MfC AppWizard(dll),选新建工作区,设置工程位置 x:\hTest,键入工程名 ChineseRes,点按钮 OK,进入 AppWizard 导向。在 AppWizard 导向的第 1 步中选择 Regular Dll,点按钮 Finish - >OK 完成工程建立。

从工程 ChineseRes 的工作区中选表单 FileView,删除工程 ChineseRes 下的所有文件项(保留文件夹)。选择菜单 File/Save Workspace 保存工程 ChineseRes。关闭工程 ChineseRes。

4. 修改资源

从系统下删除目录 x:\hTest\hEnglishRes 和目录 x:\hTest\hChineseRes 下的.h、.cpp、.rc、.def、.clw 和.txt 文件,并删除目录 x:\hTest\hEnglishRes\hres 和目录 x:\hTest\hChineseRes\hres 下的全部文件。

将目录 x:\hTest 下的 Test.rc 和 Resource.h 文件拷贝到目录 x:\hTest\hEnglishRes 和目录 x:\hTest\hChineseRes 下。将目录 x:\hTest\hres 下全部文件拷贝到目录 x:\hTest\hEnglishRes\hres 和目录 x:\hTest\hChineseRes\hres 下。

5. 编译资源

()打开工程 EnglishRes,从工作区中选表单 FileView,选菜单 Project/Add To Project/Files,从插入文件到工程对话框中选目录 x:\hTest\hEnglishRes 下的文件 Test.rc 和 Resource.h 插入到工程中,再选目录 x:\hTest\hEnglishRes\hres 下的全部文件插入到工程中。

选菜单 Project/Settings。选表单 Link,在输出文件名编辑框 Output file name 中键入../Debug/EnglishRes.dll,使输出

文件 EnglishRes. dll 输出到目录 x hTest hDebug 中；在工程选项列表 Project Options 的末尾添加选项 /NOENTRY，告诉连接器这是一个仅有资源的 DLL，没有入口点。选表单 Resources，在语言下拉列表 Language 中选英语；在附加资源包含目录编辑框中加入 MFC 的通用英文资源包含路径，例如：C hProgram Files hMicrosoft Visual Studio hVC98 hMFC hInclude，删除预处理器定义编辑框中的 _AFXDLL 选项，这两项设置使工程在指定的位置收集通用资源（储如打印预览等资源）。点按钮 OK 完成工程设置。

选菜单 Build / Build EnglishRes. dll，编译生成 EnglishRes. dll 并输出到 x hTest hDebug 中。

(2) 打开工程 ChineseRes，从工作区中选表单 FileView，选菜单 Project / Add To Project / Files，从插入文件到工程对话框中选目录 x hTest hChineseRes 下的文件 Test. rc 和 Resource. h 插入到工程中，再选目录 x hTest hChineseRes hres 下的全部文件插入到工程中。

在工程 ChineseRes 中，将英文资源翻译成中文资源，包括菜单、版本、图标、位图等资源。把菜单项 ID_RES_SWITCH 的 Caption 改为：英文。对于像本文这样的示例，大可不必去把英文逐一翻译成中文。可再建立一个与工程 Test 同名，且除工程的语言资源选择中文外，其他选项都相同的临时工程，把这个临时工程目录下的文件 Test. rc 和 Resource. h 拷贝到目录 x hTest hChineseRes 下，临时工程目录中 hres 下的全部文件拷贝到目录 x hTest hChineseRes hres 下即可。这样省去了翻译的工作量。

选菜单 Project / Settings。选表单 Link，在输出文件名编辑框 Output file name 中键入 . . / Debug / ChineseRes. dll，使输出文件 ChineseRes. dll 输出到目录 x hTest hDebug 中；在工程选项列表 Project Options 的末尾添加选项 /NOENTRY。选表单 Resources，在语言下拉列表 Language 中选中文；在附加资源包含目录编辑框中加入 MFC 的通用中文资源包含路径，例如：C hProgram Files hMicrosoft Visual Studio hVC98 hMFC hInclude hl. chs，删除预处理器定义编辑框中的 _AFXDLL 选项。点按钮 OK 完成工程设置。

选菜单 Build / Build ChineseRes. dll，编译生成 ChineseRes. dll 并输出到 x hTest hDebug 中。

至此得到了需要的动态资源库 EnglishRes. dll 和 ChineseRes. dll，并放到 x hTest hDebug h下。下面我们将要在应用 Test 中切换调用这两个库。

6. 添加切换处理函数

(1) 打开工程 Test，选择菜单 View / ClassWizard，选表单 Message Maps，从下拉列表 Class Name 中选类 CMainFrame，在列表 Object IDs 中选消息响应标识 ID_RES_SWITCH，双击消息列表 Messages 中的 COMMAND，添加消息响应函数 OnResSwitch。

(2) 在类 CTestDoc 中重载 OnOpenDocument 和 OnCloseDocument。从下拉列表 Class Name 中选类 CTestDoc，在列表 Object IDs 中选类 CTestDoc，双击消息列表 Messages 中的 OnOpenDocument，重载 OnOpenDocument；双击 OnCloseDocument，重载 OnCloseDocument。点按钮 OK 退出 ClassWizard。

(3) 在工作区中选表单 FileView，打开文件夹 Header Files 中的文件 MainFrm. h，添加下面的黑体字：

```
class CMainFrame : public CMDIFrameWnd
{
. . . . . .
public:
HINSTANCE m_hRes;              // resource DLL handle
CMenu * m_pMenu;               // menu for main frame
char m_iRes;                   // resource identifier
. . . . . .
// Operations
public:
void LoadMenu();
. . . . . .
}
```

(4) 打开文件夹 Source Files 中的文件 MainFrm. cpp，添加下面的黑体字：

```
CMainFrame: : CMainFrame()
{
. . . . . .
m_iRes = ´ ´;
m_hRes = AfxLoadLibrary(_T(˝ChineseRes. dll˝));
if(m_hRes)
      {
AfxSetResourceHandle(m_hRes);
m_iRes = ´C´;
      }
      else
      {
m_hRes = AfxLoadLibrary(_T(˝EnglishRes. dll˝));
      if(m_hRes)
      {
            AfxSetResourceHandle(m_hRes);
            m_iRes = ´E´;
      }
}
m_pMenu = NULL;
}
CMainFrame: : ~CMainFrame()
{
. . . . . .
if(m_hRes)
      {
AfxFreeLibrary(m_hRes);
m_hRes = NULL;
}
if(m_pMenu)
delete m_pMenu;
. . . . . .
}
void CMainFrame: : OnSwitchRes()
```

```
{
//Load the language resource DLL
if(m_iRes = = ´C´)
{
AfxFreeLibrary(m_hRes);
m_hRes = AfxLoadLibrary(_T(˝EnglishRes. dll˝));
if(m_hRes)
{
AfxSetResourceHandle(m_hRes);
m_iRes = ´E´;
LoadMenu();
}
}
else if(m_iRes = = ´E´)
{
AfxFreeLibrary(m_hRes);
m_hRes = AfxLoadLibrary(_T(˝ChineseRes. dll˝));
if(m_hRes)
{
AfxSetResourceHandle(m_hRes);
m_iRes = ´C´;
LoadMenu();
}
}
}
```

添加一个新函数 LoadMenu

```
void CMainFrame: : LoadMenu()
{
//得到已打开的文档数
int nDocCount = 0;
POSITION posDocTemplat = AfxGetApp( ) ->GetFirst-
DocTemplatePosition();
while( posDocTemplat ! = NULL )
{
CDocTemplate * pDocTemplate
= AfxGetApp() ->GetNextDocTemplate(posDocTemplat);
POSITION posDoc = pDocTemplate ->GetFirstDocPositio();
while( posDoc ! = NULL )
{
CDocument * pDoc = pDocTemplate ->GetNextDoc(posDoc);
nDocCount + +;
}
}
if(m_pMenu)
delete m_pMenu;
m_pMenu = new CMenu();
if(nDocCount>0)          //有打开的文档, 加载应用菜单
m_pMenu ->LoadMenu( IDR_TESTTYPE );
else                     //无打开的文档, 加载空的框架菜单
m_pMenu ->LoadMenu( IDR_MAINFRAME );
AfxGetMainWnd() ->SetMenu(m_pMenu);
}
```

5）**打开文件夹** Source Files **中的文件** Test. cpp，**添加下面的黑体字：**

```
BOOL CTestApp: : InitInstance()
{
. . . . . .
pMainFrame ->LoadMenu();
return TRUE;
}
```

6 **打开文件夹** Source Files **中的文件** TestDoc. cpp **添加下面的黑体字**

```
. . . . . .
#include ˝MainFrm. h˝
. . . . . .
BOOL CTestDoc: : OnNewDocument(LPCTSTR lpszPath-
Name)
{
. . . . . .
((CMainFrame * )AfxGetMainWnd()) ->LoadMenu();
return TRUE;
}
BOOL CTestDoc: : OnOpenDocument(LPCTSTR lpszPath-
Name)
{
. . . . . .
((CMainFrame * )AfxGetMainWnd()) ->LoadMenu();
return TRUE;
}
void CTestDoc: : OnCloseDocument()
{
. . . . . .
((CMainFrame * )AfxGetMainWnd()) ->LoadMenu();
}
```

7. **编译应用**

（1）**选中文件夹** Source Files **中的文件** Test. rc，**选择菜单** Edit / Cut，**从工程** Test **中剪除文件** Test. rc。

（2）**选择菜单** Build / Build Test. exe，**编译生成** Test. exe。

至此得到了应用 Test **所需要的三个文件** Test. exe、EnglishRes. dll、ChineseRes. dll。**只要这三个文件在同一个目录下，后**

图 1

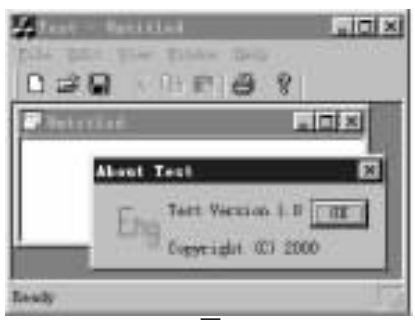


图 2

可靠地避免应用程序运行多个实例

董文军

摘 要 本文通过对一些常用的避免多实例运行方法的分析，指出它们都不能保证 100% 的可靠，同时进一步分析了其中的原因，提出了一种利用操作系统互斥量作为标志来避免多实例运行的方法，具有实现简单可靠性相当高的特点。

关键词 操作系统，互斥量，内核，多实例

一、问题的提出

Windows 的多任务特性使得同一个 Windows 程序可以多次被运行，但是在很多情况下是不允许的，因为这不但占用了系统资源还会造成应用程序运行的不稳定。例如同一个应用程序可能对同一个对象进行操作，如果在编程时没有考虑到这种情况，则会造成系统死锁等不良情况。因此我们应该使编写的程序只能运行一个实例，相关的方法很多（包括微软的 MSDN），但是这些方法都有不完善的地方，不能 100% 地保证只有一个实例被运行。

二、一般常用方法的缺陷

绝大部分方法都使用 FindWindow（LPCTSTR classname，LPCTSTR caption） API 函数来寻找前一个实例，但是如果不正确使用此 API 函数，在你调用它之后，很可能使你的程序被挂起。此函数有两个参数，但是如果你不使用 AfxRegisterClass 注册你的 Windows 类你就得不到 classname。所以很多程序忽略第一个参数，而使用第二个参数，即通过寻找应用程序窗口的标题来判断，但这有很多的限制，例如你的程序不能是多文档结构的，而且窗口的标题必须是固定的。更为糟糕的是，此函数在执行时先调用 EnumWindows 找到窗口再通过 GetWindowText 向窗口查询标题，即通过 WM_GETTEXT 传递消息来实现的。SendMessage 可以被其他程序阻塞，所以即使你的程序设计正确也会因这个原因被挂起，使用超时可以解决这个问题，但又无法保证只能一个实例被运行。

另一种改进的方法是使用 FindWindow 函数的第一个参数，通过 GUIDGEN 得到一个唯一的 classname 标识，再使用 AfxRegisterClass 注册你的 Windows 类，于是可以通过第一个参数很快捷地找到程序的前一个实例。而且不必担心因其他程序将消息阻塞而造成的死锁。 这种方法的确比较有效，绝大多数情况下都可以避免多个实例的运行，可是在一种特殊的情况下（竞争状态）还是无法避免多个实例的运行。想像这样一种情况：在第一个实例没有成功注册你的 Windows 类之前第二个实例又被运行，则两实例都会被执行。这种情况还是有可能发生的，如 Win98 被设置成为单击运行而大家又都习惯于双击运行，或者因为鼠标的设置或本身故障的原因等。

当然还有一些更糟糕的方法，如在内存、注册表或其他什么地方设置一个运行的标识，一旦程序因为某种原因没有将标识位清除，则无法再次运行该程序。

三、一种相当可靠的基于系统内核的解决办法

我们知道 Windows 操作系统是支持互斥量的，CreateMutex 可以保证不出现问题，因为这是一个原子操作，不论两个进程之间的相对时间如何，只要是创造同一个互斥量，严格地只有一个进程能够产生互斥量而另一个进程将得到 ER-

---

动 Test.exe 选择应用 Test 的菜单查看（英文）或 View/Chinese），就可以实现中/英文可切换的应用界面（如图 1、图 2）。

四、小结

多国语言切换技术的关键在于利用了 Windows 可将语言资源集中在动态资源库内，并能动态加载资源库的能力。使应用程序响应用户的不同选择而加载不同的资源库，从而实现多国语言切换。使用这种技术即使不重新编译应用程序也可以实现应用本地化，因而该技术具有一定的实用价值。

本文详细介绍了在多文档中实现中/英文语言切换的技术。在单文档（SDI）中实现这种技术更为简单，不需要修改上述的文件 Test.cpp 和 TestDoc.cpp 中的内容，文件 MainFrm.cpp 的添加的内容也更少，有兴趣的读者可试一试。

参考文献

（美）斯坦菲尔得（Stanfield S.）、（美）阿维森（Arvesen R.）著，华译工作室译 Visual C++ 开发人员指南 - 北京：机械工业出版社 1997.6

收稿日期：2000 年 7 月 20 日）

ROR_ALREADY_EXISTS 的错误。正因为这是一种基于操作系统核心的操作，可以得到保证在一个 Mutex 被创造之前另一个过程不能被创造，即使出现竞争状态的情况也只能有一个实例被运行。

首先重载函数 InitInstance 创建一个互斥量，如果成功，则没有另一个实例存在；如果失败，则原来已有一个实例被运行。一旦发现程序已经被运行则需要接着找到前一个实例的窗口句柄，我们定义一系统回调函数，通过发送自定义的消息 UWM_ARE_YOU_ME 来进行询问，判断是否是要找的窗口。

在 MFC 的 CMainFrame 类中通过以下方法加入自己的消息处理函数：

```
static const UINT UWM_ARE_YOU_ME = RegisterWindowMessage(_T("UWM_ARE_YOU_ME"));
//定义自己的消息
BEGIN_MESSAGE_MAP(CMainFrame, CFrameWnd)
    //{{AFX_MSG_MAP(CMainFrame)
ON_REGISTERED_MESSAGE( UWM_ARE_YOU_ME, OnAreYouMe )
//注册自己的消息
    //}}AFX_MSG_MAP
END_MESSAGE_MAP()
在类成员中增加自己的处理函数
afx_msg LRESULT OnAreYouMe(WPARAM, LPARAM);
处理代码
LRESULT CMainFrame::OnAreYouMe(WPARAM, LPARAM)
{
    return UWM_ARE_YOU_ME;
} // CMainFrame::OnAreYouMe
```

四、一个实例

首先使用向导生成一个单文档程序，将自己定义的消息按前述方法加入，然后再重载以下函数（为节省篇幅只写增加的代码）：

```
BOOL CSInstanApp::InitInstance()
{
  bool AlreadyRunning;
  HANDLE hMutexOneInstance = ::CreateMutex( NULL, FALSE, T( " MYAPPNAME -088FA840 -B10D -11D3 -BC36 -006067709674"));
    AlreadyRunning = ( ::GetLastError( ) == ERROR_ALREADY_EXISTS || ::GetLastError( ) == ERROR_ACCESS_DENIED);
  // The call fails with ERROR_ACCESS_DENIED if the Mutex was created in a different users session because of passing NULL for the SECURITY_ATTRIBUTES on Mutex creation);
  if ( AlreadyRunning )
  { /* kill this */
      HWND hOther = NULL;
    EnumWindows(searcher, (LPARAM)& hOther);
//call the CALLBACK function searcher and retrive the handle of previous instance by hOther
    if ( hOther != NULL )
    { /* pop up */
      ::SetForegroundWindow( hOther );
      if ( IsIconic( hOther ) )
      { /* restore */;
        ::ShowWindow( hOther, SW_RESTORE )
      } /* restore */
    } /* pop up */
    return FALSE; // terminates the creation
  } /* kill this */
  // ... continue with InitInstance
  return TRUE;
}
```

系统回调函数用于查询前一个实例的窗口句柄。

```
BOOL CALLBACK CSInstanApp::searcher(HWND hWnd, LPARAM lParam)
{
  DWORD result;
 LRESULT ok = ::SendMessageTimeout(hWnd, UWM_ARE_YOU_ME, 0, 0, SMTO_BLOCK |SMTO_ABORTIFHUNG, 200, & result);
  if(ok == 0)
    return TRUE; // ignore this and continue
  if(result == UWM_ARE_YOU_ME)
  { /* found it */
    HWND * target = (HWND *)lParam;
    *target = hWnd;
    return FALSE; // stop search
  } /* found it */
  return TRUE; // continue search
}
```

收稿日期 2000 年 8 月 28 日

编程实现 VFP6.0 操纵 Word 灵活打印特殊格式的报表

张夏林 汪新庆 王 兴

VFP6.0 作为优秀的桌面型数据库系统，提供了强大的报表、标签输出功能。但是在实际应用中，其提供的报表输出格式常常不符合要求，有时会碰到一些复杂格式的表格，有时要多报表同文档输出，用 VFP6.0 的报表设计功能难于实现。

在实际应用中笔者尝试编程实现用 VFP6.0 操纵 Word 灵活打印各式报表，取得了良好的效果，在此介绍这种方法。众所周知，Microsoft 的 Word 是一个功能强大的字处理软件，它可以用自身带有的宏编辑器编写宏来实现绝大多数菜单功能，也可以用 VBA 编写程序完成很多任务。编程实现 VFP6.0 操纵 Word 灵活打印特殊格式的报表，是用 VFP6.0 编写嵌有类似于 VBA 宏的程序，执行 Word 的命令打印报表、报告。所以理论上，用这种方式可以实现任意格式报表的输出。

问题的关键在于：一、正确地在 VFP6.0 中调用 Word。对于在 VFP6.0 中调用 Word 也可理解为客户/服务器的请求和应答关系），使用语句 loWord = etcominstance word.application 调用后，Word 就可视为 VFP 的一个对象，为了使用其变量、方法、事件和过程，必须在程序开头时包含 Word 的一个头文件，语句为：#INCLUDE wdolb.h（Offices97 中的 Word） 或 #INCLUDE msolb09.h（Offices2000 中的 Word）。二、正确地使用 Word 能执行的编排版式的语言。使用这种方式打印报表的主要目的是满足特殊的报表格式，而编排这些格式是依靠 Word 能运行的宏代码，这些宏代码又要在 VFP 中被执行，而宏在 Word 中与在 VFP 中形式大体相近，但书写格式不尽相同。例如要插入一个两行两列的表格，Word 中宏格式为：

ActiveDocument.Tables.Add Range = Selection.Range NumRows = 2 NumColumns = 2 而在 VFP 中应该为：.Activedocument.Tables.Add .Selection.Range 2 2 ；再如输出一行文本“电脑编程” Word 中宏格式为：Selection.TypeText Text = ‘电脑编程”，而在 VFP 中应该为：Selection.TypeText （‘电脑编程”）等。在后面的例程中可以看出这些差别。当不知道该如何实现某个特殊格式时，最便捷的做法是在 Word 中刻录该格式的 Word 宏，拷贝到 VFP 中做适当修改即可。三、正确处理 VFP6.0 宿主语句与 Word 宏语句的镶嵌关系。这种报表打印方式的优越之处，不光在于单份特殊格式的报表输出，而且在于多份在 VFP 意义上的报表的同文档打印，甚至用一些信息挖掘手段从几十、上百个数据表中提取信息，打印出一个大体完整的既有表格又有自由文字的报告，这些是很诱人的。所以，一般这样的报表打印程序除了满足格式上的要求外，通常还要实现一定的数据筛选，信息挖掘的功能，或者多表记录关联在同一报表输出，多报表的同文档输出等等，所以除了 Word 宏，程序中还有大量 VFP 自身的处理语句，有时两者是相互嵌套的，要注意两类语句排列顺序，有时顺序的错误会导致程序的错误。例如，对 Word 对象的控制语句要放在“WITH . . . . . ENDWITH”中，这样，在使用 VFP 的循环语句、分支语句时一定要注意不要嵌套错误。编程中如果出现奇怪的错误，最好是检查一下语句的排列顺序。

下面举一个简单的例子，介绍如何在 VFP6.0 中调用 Word、用其代码控制、以报表形式打印输出满足条件的记录。这种打印结果脱离了 VFP6.0，可以保存为独立电子文档，也可做进一步的编辑。

该示例是假定在学生信息管理系统中，要打印满足学号为特定值的学生信息。

首先，新建一个目录 WriteReport 作为当前 VFP6.0 默认路径，在其中建一个数据库表文件 Student，形式为 Student（name C 8，gender C 2，borntime D 8，photo G 4，studentno C 12，stime D 8，dpartment C 20，major C 18，drom C 20，dtelenumber C 15，address C 20，htelenumber C 15，hobby M 4），其中设置字段 studentno 为该表主关键字，并录入几条记录；然后建立一个名为 WriteReport 的过程 程序清单附后）；最后建立一个如图 1 所示的表单调用 WriteReport 过程，其中“打印”按钮的代码为：

图 1 执行表单

```
cStuNumb = ThisForm.List1.Value
IF EMPTY(cStuNumb)
MESSAGEBOX(″请先用鼠标在列表框中单击一个学生的姓名!″,0+64+0,″提示信息″)
ELSE
  Do WriteReport WITH cStuNumb
ENDIF
```

执行该过程即可在 word 中打印出如图 2 所示的报表.

WriteReport 程序清单:

```
* PROCEDURE WriteReport  Write Student Document from database table By Word!
PARAMET cStudentNumber
#INCLUDE wdolb.h
#DEFINE True .T.
```

学生信息报表

照片	姓名	高志雄	性别	男	学号	0202000002
	出生年月	1980-3-25	入学时间		1999-1-9	
	所在院系	资源学院	所学专业		资源管理	
学校住址	学生公寓 50045		寝室电话		87432222	
家庭住址	武汉市汉阳区 202 号		联系电话		027-85858585	
个人特长	擅长打爵士鼓，喜欢踢足球，爱好文学。					

图 2. 学生信息报表样式

```
#DEFINE False . F.
IF !USED(´student´)  && Open the tableo
    USE student IN 0 ORDER studentno
ENDIF
SELE student
IF PARA() =0  && When there is no parameter been pasted, write the first student in the table
    cStudentNumber  =  student. studentn
ENDIF
SEEK cStudentNumber
IF FOUND()
    LOCAL loWord, loTable, lnRow, lnColumn
    loWord =Getcominstance( ˝ word. application˝)    && Call Word application
    loWord. Visible =. t.
    cString  =  ˝学生信息报表˝
    WITH loWord
        . Documents. Add(˝Normal˝, False)
        . Selection. MoveDown(wdLine, 100)
WITH . Selection    && Information on the top of the  report
    . Font. Name  =  ˝宋体˝    && Format of the String
    . Font. Size  =  20
    . ParagraphFormat. Alignment =wdAlignParagraphCenter
        . TypeText(cString)
        . TypeParagraph        && Add a blank line
    ENDWITH
    . Selection. Sections(1) . Footers(1) . PageNumbers. Add (wdAlignPageNumberCenter, True)
    . Selection. Font. Size  =14
    . Activedocument. Tables. Add(. Selection. Range, 1, 7)
    loTable =. Activedocument. Tables(1)
    WITH  loTable
        . Cell(1, 1). Width =80
        . Cell(1, 2). Width =45
        . Cell(1, 3). Width =80
        . Cell(1, 4). Width =45
        . Cell(1, 5). Width =30
        . Cell(1, 6). Width =45
        . Cell(1, 7). Width =90
        . Cell(1, 1). Range. Insertafter(˝照片˝)
        . Cell(1, 2). Range. Insertafter(˝姓名˝)
        . Cell(1, 4). Range. Insertafter(˝性别˝)
        . Cell(1, 6). Range. Insertafter(˝学号˝)
        . Cell(1, 3). Range. Insertafter(name)
        . Cell(1, 5). Range. Insertafter(gender)
        . Cell(1, 7). Range. Insertafter(studentno)
        ENDWITH
    . Selection. MoveDown(wdLine, 100)
    . Activedocument. Tables. Add(. Selection. Range, 2, 5)
    loTable  =  . Activedocument. Tables(1)
    WITH  loTable
        FOR j =2 TO 3
        . Cell(j, 1). Width =80
        . Cell(j, 2). Width =70
        . Cell(j, 3). Width =100
        . Cell(j, 4). Width =70
        . Cell(j, 5). Width =95
        ENDFOR
        . Cell(2, 2). Range. Insertafter(˝出生年月˝)
        . Cell(2, 3). Range. Insertafter(borntime)
        . Cell(2, 4). Range. Insertafter(˝入学时间˝)
        . Cell(2, 5). Range. Insertafter(stime)
        . Cell(3, 2). Range. Insertafter(˝所在院系˝)
        . Cell(3, 3). Range. Insertafter(department)
        . Cell(3, 4). Range. Insertafter(˝所学专业˝)
        . Cell(3, 5). Range. Insertafter(major)
    ENDWITH
. Selection. SelectColumn    && Select the three columns
. Selection. Cells. Merge      && unite columns
. Selection. MoveDown(wdLine, 100)
. Activedocument. Tables. Add(. Selection. Range, 2, 4)
loTable =. Activedocument. Tables(1)
  WITH  loTable
  FOR j =4 TO 5
  . Cell(j, 1). Width =80
  . Cell(j, 2). Width =170
  . Cell(j, 3). Width =70
  . Cell(j, 4). Width =95
    ENDFOR
. Cell(4, 1). Range. Insertafter(˝学校住址˝)
. Cell(4, 2). Range. Insertafter(drom)
. Cell(4, 3). Range. Insertafter(˝寝室电话˝)
. Cell(4, 4). Range. Insertafter(dtelenumber)
. Cell(5, 1). Range. Insertafter(˝家庭住址˝)
. Cell(5, 2). Range. Insertafter(address)
. Cell(5, 3). Range. Insertafter(˝联系电话˝)
. Cell(5, 4). Range. Insertafter(htelenumber)
ENDWITH
. Selection. MoveDown(wdLine, 100)
. Activedocument. Tables. Add(. Selection. Range, 1, 2)
    loTable  =  . Activedocument. Tables(1)
    WITH lotable
        . Cell(6, 1). Width =80
  . Cell(6, 1). Height =60
  . Cell(6, 2). Width =335
  . Cell(6, 1). Range. Insertafter(˝个人特长˝)
  . Cell(6, 2). Range. Insertafter(hobby)
    ENDWITH
. Selection. Sections(1)  . Footers(1)  . PageNumbers. Add (wdAlignPageNumberCenter, True)  && Insert Pagenumber
  ENDWITH
ENDIF
RETURN
```

该程序在 VFP6. 0 中调试通过。 注意：要求同机装有 Offices 的 Word 软件，程序才能正常运行）

收稿日期：2000 年 10 月 8 日）

应用程序中在不重新启动系统方式下切换桌面

胡春松

摘 要 本文介绍了在 Windows 平台下如何实现通过不重新启动系统而在应用程序中切换桌面的编程技术,通过一个 Visual C++ 的例子介绍实现过程。

关键词 桌面，应用程序，Visual C++

在 Windows NT WorkStation / Server 或 Windows 2000 中，有时我们需要切换到另外的桌面，一般情况下通过关闭系统重新登陆的方式实现。是否能够不用重新启动机器而只需输入某个“热键”或单击某个“按键”来实现呢？在网络上我们可以找到一些小软件可以帮助实现这一功能，但是在我们开发的应用程序中也需要这一功能时又该如何实现呢？本文就这一技术问题进行了阐述，同时给出了一个 C++ 类 CSwitchDesktop，在 CSwitchDesktop 中封装了一些 Windows API 函数来来实现切换桌面的功能。通过此类我们可以方便地创建、销毁和在多个桌面之间进行切换。

首先，我们给出类 CswitchDesktop 的定义如下：

```
class CSwitchDesktop
{;
public:
    CSwitchDesktop(); //constructor
    virtual ~CSwitchDesktop(); //deconstructor
public:
    BOOL Open(LPCTSTR lpszName);
    void Close(void);
    BOOL Switch(void) const;
    static BOOL SwitchToDefault(void)
    static BOOL SwitchTo(LPCTSTR lpszName);
protected:
    HDESK m_hDesktop;
    static BOOL CALLBACK _CloseWindowProc(HWND
hWnd, LPARAM lParam);
};
```

下面我们逐个介绍各个成员函数。Open 函数一般是第一个被调用的成员函数，它接受一个以 NULL 空字符标志结束的字符串指针为输入参数，使用该字符串来标志每一个桌面。该参数需要注意以下两点：

该字符串参数对大小写敏感；

该字符串参数不能包含反斜杠字符[h]。

可以使用任何满足以上两点要求的字符串作为桌面的标志。该函数的具体实现代码如下：

```
BOOL CSwitchDesktop: : Open(LPCTSTR lpszName)
{
    DWORD dwRights = DESKTOP_READOBJECTS
        |DESKTOP_CREATEWINDOW
        |DESKTOP_CREATEMENU
      |DESKTOP_HOOKCONTROL
      |DESKTOP_JOURNALRECORD
      |DESKTOP_JOURNALPLAYBACK
      |DESKTOP_ENUMERATE
       |DESKTOP_WRITEOBJECTS
      |DESKTOP_SWITCHDESKTOP
      |STANDARD_RIGHTS_REQUIRED
      |READ_CONTROL
      |WRITE_DAC
      |WRITE_OWNER;
m_hDesktop = : :  CreateDesktop(const_cast <LPTSTR>
(lpszName), NULL, NULL,
DF_ALLOWOTHERACCOUNTHOOK, dwRights, NULL);
  return (NULL! = m_hDesktop);
}
```

可以看出在 Open 中封装了 SDK 函数 CreateDesktop 访问权限标志采用多个值的组合提高了此 C++ 类的灵活性。该类的成员变量 m_hDesktop 保存了 CreateDesktop 调用的返回值，它为后面的桌面切换操作提供了一个桌面句柄。

桌面通过类成员函数 Close 进行销毁，该函数除了关闭指定的桌面外还同时销毁此桌面上所有的窗口。关闭桌面上的所有窗口是为了防止此被关闭桌面上的窗口跳转到缺省的桌面上。因为所有的顶层（top - level）窗口均把桌面作为其“父”窗口，如果应用窗口的“父”窗口被销毁后，操作系统将自动把这些窗口移动到下一个桌面上。其实现函数如下：

```
void CSwitchDesktop: : Close(void)
{,
  SwitchToDefault();
  if(NULL! = m_hDesktop)
  : :  EnumDesktopWindows(m_hDesktop
_CloseWindowProc, 0L);
  : : CloseDesktop(m_hDesktop);
  m_hDesktop = NULL;
}
```

函数调用 SwitchToDefault 隐藏了关闭所有桌面窗口的细节，通过调用 EnumDesktopWindows 枚举出桌面上所有的窗口，通过递归调用静态回调函数 _CloseWindowProc 来关闭桌面上的窗口，最后由 SDK 函数 CloseDesktop 销毁桌面。

函数 _CloseWindowProc 实现代码如下：

```
BOOL CSwitchDesktop: : _CloseWindowProc(HWND hWnd,
LPARAM lParam)
{
  DWORD dwProcessID;
```

```
    : : GetWindowThreadProcessId(hWnd, & dwProcessID);
    if(: : GetCurrentProcessId( ) = = dwProcessID)
        : : PostMessage(hWnd, WM_CLOSE, 0, 0L);
    else
        : : PostMessage(hWnd, WM_QUIT, 0, 0L);
    return TRUE;
}
```

在上面的函数中通过判断桌面上的窗口进程是否属于当前进程来选择两种方式之一来关闭窗口。如果属于当前进程则通过发送 WM_CLOSE 消息来关闭窗口，否则发送 WM_QUIT 消息来关闭窗口并且终止进程。

通过成员函数 SwitchToDefault 来切换到缺省的桌面，它只是简单地调用静态成员函数 SwitchTo 并以“保留”桌面名字“default”作为参数。代码为：

```
BOOL CSwitchDesktop: : SwitchToDefault(void)
{
    return CSwitchDesktop: : SwitchTo(_T(″Default″));
}
```

成员函数 SwitchTo 以需要激活的桌面名作为参数，调用 SDK 函数 OpenDesktop 获取指定激活桌面的句柄参数，该临时句柄作为 SwitchDesktop 的参数来实现切换到指定的桌面，然后调用 CloseDesktop 清除临时桌面句柄。

```
BOOL CSwitchDesktop: : SwitchTo(LPCTSTR lpszName)
{
    HDESK hDesktop = : : OpenDesktop(const_cast < LPTSTR>(lpszName),
    DF_ALLOWOTHERACCOUNTHOOK, FALSE, DESKTOP_SWITCHDESKTOP);
    if(NULL = = hDesktop)
        return FALSE;
    BOOL bReturn = : : SwitchDesktop(hDesktop);
    : : CloseDesktop(hDesktop);
    return bReturn;
}
```

一般我们调用成员函数 Switch 进行桌面间的切换，由于它使用内部桌面句柄 m_hDesktop 进行桌面切换，通过封装 SDK 函数 SwitchDesktop ，所以执行速度比成员函数 SwitchTo 要快许多。其代码为：

```
BOOL CSwitchDesktop: : Switch(void) const
{
    return : : SwitchDesktop(m_hDesktop);
}
```

这就是整个类 CSwitchDesktop 的实现代码。这个类提供了实现创建复杂界面应用的简单方法，只要系统资源允许，可以用来创建任意数目的桌面。

下面我们以一个基于 MFC 的对话框应用程序作为例子来说明类 CSwitchDesktop 的使用方法。在下面的例子中我们将在两个桌面之间进行切换。首先利用 AppWizard 创建一个基于对话框的 Project 采用缺省参数便可），名为 Desktop。然后将类 CSwitchDesktop 的声明和实现文件加入到此工程中。在 CDesktopDlg 类中加入一个类 CSwitchDesktop 的实例 m_sdtSample 作为类成员变量，同时将头文件包含语句#include SwitchDesktop. h 加到 CDesktopDlg 类的声明头文件中。现在我们已经将类 CSwitchDesktop 集成到了例子工程 Desktop 中了。

在 CDesktopDlg OnInitDialog 中加入如下代码

```
BOOL CDesktopDlg: : OnInitDialog( )
{
    . . .
// TODO: codes added for switching desktop demo
m_sdtSample. Open(_T(″SampleDesktop″));
: : RegisterHotKey(m_hWnd, DEFAULT_DESKTOP_ID, MOD_CONTROL, VK_F12); : : RegisterHotKey(m_hWnd, SAMPLE_DESKTOP_ID, MOD_CONTROL, VK_F11);
    . . .
}
```

这段代码打开一个名为“SampleDesktop”的桌面，同时接着注册两个组合热键“Ctrl + F11”和“Ctrl + F12”分别对应切换到“SampleDesktop”桌面和缺省“Default”桌面。当用户按下以上两个热键之一便会触发 WM_HOTKEY 消息发送到主窗口，在 WM_HOTKEY 消息处理函数中，如果用户按下热键“Ctrl + F12”则切换到缺省“Default”桌面；若按下热键“Ctrl + F11”则切换到“SampleDesktop”桌面，同时启动程序“Explorer. exe”。代码如下：

```
LRESULT CDesktopDlg: : OnHotKey(WPARAM wParam, LPARAM lParam)
{
  static bStartExplorer = TRUE;
  if(DEFAULT_DESKTOP_ID = = wParam)
  CSwitchDesktop: : SwitchToDefault( );
  else if(SAMPLE_DESKTOP_ID = = wParam)
  {
  m_sdtSample. Switch( );
  if(TRUE = = bStartExplorer)
  {
  bStartExplorer = FALSE;
  PROCESS_INFORMATION pi;
  ZeroMemory(& pi, sizeof(PROCESS_INFORMATION));
  STARTUPINFO si;
  ZeroMemory(& si, sizeof(STARTUPINFO));
  si. cb = sizeof(STARTUPINFO);
  si. lpDesktop = _T(″SampleDesktop″);
  : : CreateProcess(NULL, _T(″explorer. exe″), NULL, NULL, TRUE, CREATE_DEFAULT_ERROR_MODE|CREATE_SEPARATE_WOW_VDM, NULL, NULL, & si, & pi);
  : : CloseHandle(pi. hProcess);
  }
  }
  return 1;
}
```

最后在 CDesktopDlg OnDestroy 注销热键，同时关闭“SampleDesktop”桌面。代码如下：

```
void CDesktopDlg: : OnDestroy( )
{
```

（下接第 24 页）

用 VC 扩展资源管理器菜单

冉林仓

摘 要 本程序通过一个简单的例子 提供一个扩展资源管理器的框架 通过这个框架 从而实现资源管理器、我的电脑以及标准文件浏览框的鼠标右击功能,借助于右击菜单实现文件打印。

关键词 资源管理器,上下文菜单,文件打印,外壳扩展

一、 引言

随着 Windows 9x 和 Windows NT 的流行，资源管理器对计算机用户来说肯定不会陌生，在资源管理器 包括我的电脑和其它的标准文件对话框），用户可以很方便地通过鼠标对文件和文件夹进行操作，尤其是那无处不在的鼠标右击极大地方便了用户，它提供的删除、更名、编辑、查看、粘贴、复制功能，使用户再也不用记那罗嗦的热键，也不用拖动那走不动的老鼠标了。细心的用户还会发现弹出式菜单中还会经常出现类似于 winzip 、virus scaners、image viewers、play in winamp 等菜单项，无疑这些菜单项大大缩短了程序与用户的距离，提高了用户对程序的认知度和亲和力，就像某一种扩展名已被某一软件注册，成为该软件的独家专利一样，你甚至经常会发现几个软件为了某一扩展名兵戈相见、水火不容，如 IE 和 NC 之争、WINZIP 和 WINRAR 较劲等等。资源管理器将像一块风水宝地，如果你不想染指的话，至少大家无法见识你的软件的大家风范和专业色彩。

有关资源管理器上下文菜单的资料可谓凤头麟角，出奇的少。微软公司出版的 msdn 提供了一个 SHELLEX 的例子，本文在参考这个例子的基础上，提供了一个制作资源管理器菜单的框架，并简要说明从调试到发行有关的注意事项，希望能够给读者提供一定的帮助。

二、 实现原理

还是让我们先看一下在资源管理器中鼠标右击系统都做了些什么：

首先系统 也就是 explorer. exe 外壳文件 确认你选中了那些文件和文件夹，查找注册文件表 HKEY_CLASS_ROOT 下的子键，查找所选文件对应的外壳扩展 CLSID 标识，然后通过查找 CLSID 子键寻找对应标识 CLSID 所对应 InprocServer32 默认值即动态连接库文件名，当外壳上下文菜单显示时，向动态连接库发送 QueryContextMenu 命令，通过动态连接库创建部分菜单，当你选中这些菜单项时，外壳又会向动态连接库发送 InvokeCommand 命令，通过动态连接库执行你的操作，包括创建对话框、创建菜单等等。

结合程序来说，借助于 COM 的魔力，利用上述的注册项，Explorer 利用这个 dll 创建了一个带有 IContextMenu / IShellExtInit 接口的对象的实例，这个对象就是 CShellExt 类。扩展的指定菜单项是在 Explorer 调用 CShellExt QueryContextMenu 时添加，而帮助命令文本是在 CShellExt GetCommandString 产生，选择文件的列表是在 CShellExt Initialize 中生成。

三、程序使用说明

1. 在您使用这个程序框架的时候，您必须生成您自己的 GUID 存储在注册编辑表中为 CLSID，您可以通过 GUIDGEN 生成，对于 VC6 且安装于 C：盘，这个文件一般位于 C hProgram Files hMicrosoft Visual Studio hCommon hTools 文件目录，生成 GUID 后把它粘贴在 CtxMenu. h 文件中。用于标识唯一特定的外壳扩展 COM 对象。

2. 每次您创建这个动态连接库时，或者您执行 REGSVR32. EXE /S /C CTXMENU. DLL 时，RegSvr32 都会调用外壳扩展中的 DLLRegisterServer 函数，在系统注册表中登记一些项目，通过这些项目 Explorer 能够知道哪些文件选中右击时外壳上下文菜单应该激活。通过修改 RegisterFileMenu 函数中的字符串 ShellExtReg. cpp ，你可以限制激活动态连接库的扩展名。DLLUnregisterServer 负责去除些登记项目。

手工注册这个动态连接库插件是件十分罗嗦的事情，你可以通过修改工程设置在编译连接时自动完成注册，修改 Project / Settings / Post - Build Command，加入 regsvr32. exe / s /c $ OUTDIR hCtxMenu. dll，VC 就会在编译连接后自动帮你注册。你可以通过单击打开按钮，选择一个或多个文件立竿见影感受到添加菜单插件的效果。

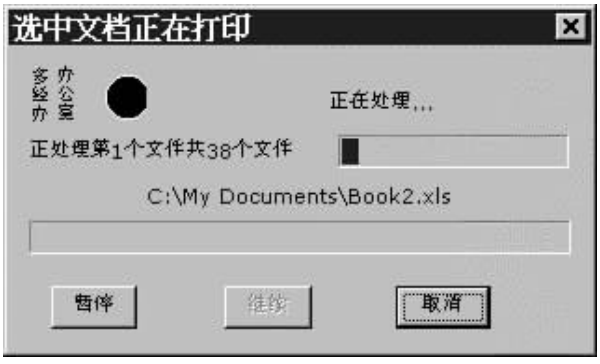
3. 程序使用了多线程编程，采用多线程编程的好处是不言而喻的，但是多线程编程会引发线程共享数据冲突等问题，为了保证这些共享数据的安全存取，程序提供了两个模板 GET_SAFE 和 SET_SAFE 用于这些共享数据变量的赋值和读取，这两个模板都是借助于多线程的同步机制实现的。在程序中建立一个共享关键区域，保证一次只有一个线程进入关键区域，其它试图进入关键区域的线程处于等待状态，直至进入关键区域的线程离开为止。为了大大缩小死锁的可能性，你应该尽可能地使用这两个模板存取这些共享数据，最好把这些共享数据的值赋值到一个局部变量，尽快离开关键区域，以便其它线程进入，本线程仅负责处理局部变量

另外对于像打印机这样的串行设备，我们无法指望它同时打印多个文件 鬼知道同时打印两个源程序打印的结果会是什么样子，反正不是我们希望看到的），只能指望它一次只能打印一个文件，形成一个打印队列。为了保证打印队列有秩

序地向前移动，我们又在程序中建立互斥信号量，第一个获得互斥量的线程获得互斥量的拥有权，其它线程因不具有拥有权被迫等待，获得拥有权的线程负责打印指定的文件，打印完毕后释放互斥信号量，等待的线程方可请求获得信号互斥量的拥有权。你在使用这个程序框架的时候，如果你的文件处理支持并行操作如磁盘处理，不一定非要建立互斥信号量，建立互斥信号量反而会影响你的运行效率。

4. 程序在 CShellExt InvokeCommand 的开始建立了一个进度取消对话框，通过这个对话框用户可以随时中断和暂停打印，注意由于 Windows 的任务机制和打印机自带打印缓冲区的影响，你可能发现暂停和取消只能管理这个对话框，并不能真正地使打印机停止操作，除非是你关掉打印机电源或者是打印机脱机，然后再按‘取消”，注意如果打印机没有联机，系统会反复地监测打印机状态，以便打印某一字符。打印方法采用直接输出到打印端口，操作系统自身的打印管理器，不会把它加入到操作系统的打印队列，也就是说你无法通过打印任务管理器暂停清除这个打印任务。打印的文件格式仅支持纯文本和打印机能够识别的二进制文件 即采用打印到文件生成的文件）。其它文件打印的结果不可预测。

5. 每次 CShellExt InvokeCommand 被调用时 程序就会循



环地处理选择的每一个文件。对每一个文件程序都要建立一个工作线程，线程执行完毕自动销毁。线程建立后执行 FileProcessThreadFunc 线程函数你可以随意地中断和暂停恢复线程运行。线程通过一个名叫 ThreadInfo 结构传递参数，你可以根据你自己的需要添加结构成员。注意要使用 GET_SAFE / SET_SAFE 模板存取结构成员，或者使用你自己的同步机制确保一次只有一个线程存取共享变量，通过线程结构传递窗口句柄并不是一个好主意，因为从一个线程到另一个线程无法保证窗口句柄是否已发生变化，也无法证明原来的窗口句柄是否有效。

6. 调试外壳扩展程序比一般的应用程序需要付出更多的工作，这是因为外壳程序就像一个插件，它不是一个独立的进程，它的运行离不开 Explorer，所以调试外壳扩展程序你必须强制 Explorer 运行在 DevStudio 中，幸运的是，VISUAL C + + STUDIO 具备这样的条件。

首先你必须在 VISUAL C + + STUDIO 中定制 tools 菜单，添加两个命令 Explorer 和 pview，分别负责启动资源管理器和管理用户进程。Explorer. exe 位于 windows 目录，pview95. exe 位于 C hProgram Files hMicrosoft Visual Studio hCommon hTools h Win95 目录。如果你使用 windows nt 强烈推荐你修改下列注册项值为 ‘0’：

HKEY_LOCAL_MACHINE h SOFTWARE h Microsoft h Windows NT hCurrentVersion hWinlogon hAutoRestartShell，修改会禁止 Explorer 的自动启动，否则你无法把 Explorer. exe 纳入到 VISUAL C + + STUDIO 环境下运行，你无法杀死已存在的这个进程并启动你自己的 explorer. exe 进程，当然你可在调试完成把它恢复原状。

还有一种比较安全的做法是，在调试程序之前先启动一个 DOS 窗口。这样即便你杀死了 explorer 进程，而你的 VISUAL C + + STUDIO 没有响应，你还可通过 DOS 框通过键入 EXPLORER. EXE 启动一份新的 Windows 外壳，你不至于陷入一个没有任务栏、没有桌面图标、没有开始菜单的尴尬局面。

调试的时候，你应该首先关闭所有的浏览器窗口，利用你刚刚建立 PVIEW 菜单杀死 EXPLORER 进程，这时候你的任务栏和桌面图标都应该消失。然后运行你的程序，指定调试会话的可执行文件为上述的 EXPLORER. EXE ，VISUAL C + + STUDIO 告诉你可执行文件没有调试信息，确认后继续。这时候 VISUAL C + + STUDIO 会启动 EXPLORER. EXE 进程，你会看到一大串 DLL 加载的信息，屏幕出现了桌面和任务栏。通过创建的菜单执行 EXPLORER. EXE 启动资源管理器窗口，在你的程序 CShellExt InvokeCommand 开始设置断点，然后右击资源管理器中的任何一个文件，你就会发现程序运行到断点处停了下来，这时候你就可以象调试其它程序一样调试这个动态连接库了。结束调试过程你可以先关闭所有资源管理器窗口，然后在选择 DEBUG /STOP Debugging 即可结束调试。调试完成后，如果你希望启动一个正常的 EXPLORER 会话以便你能够使用桌面和任务栏，你只需通过 TOOLS 菜单选择 EXPLORER。

7. 如果你希望在上下文菜单扩展中只处理文件夹，你需要做出如下变化：

In RegisterFileMenu and UnregisterFileMenu, you will need to change the line

REGSTRUCT OtherShExEntries[] = {
HKEY_CLASSES_ROOT, TEXT(″ * \\ shellex\\ ContextMenuHandlers\\″ SHELLEXNAME), NULL, TEXT(″%s″),

改为

REGSTRUCT OtherShExEntries[] = {
HKEY_CLASSES_ROOT, TEXT(″ Folder\\ shellex\\ ContextMenuHandlers\\″ SHELLEXNAME), NULL, TEXT(″%s″),

8. 服务器注册

如果你要发行一个外壳扩展程序，你必须保证安装程序能够注册这个动态连接库，幸运的是大部分的安装程序生成工具都有此项功能。

（收稿日期：2000 年 10 月 13 日）

用 VB 实现“Word 助手”动画效果

元晋豫

众所周知，Word 中有一个以动画效果呈现的‘助手’，调皮的‘大眼夹’能帮助用户及时查找所需的帮助信息，如果我们在自己开发的软件中也能实现类似的帮助动画，其效果必然新颖动人，使软件更具吸引力。笔者经过仔细琢磨，用 VB 实现了此动画，读者只要稍作修改，即可很容易地添加到自己的软件中，本程序运行后如图一所示，下面介绍实现过程。

图 1

编程原理

该动画的编程原理较简单，主要是事先建立好信息提示窗体和‘大眼夹’动画窗体，然后在需要帮助时用‘推镜头’的效果显示动画窗体，接着显示信息提示窗体。信息提示窗体包括圆角矩形窗体、三角形窗体，它们的建立和动画窗体的‘推镜头’效果需要依靠 Windows 提供的有关 API 函数（随后介绍）。

技术要点和相关 API 函数简介

建立圆角矩形窗体、三角形窗体

这里主要依靠下列几个 Windows 提供的 API 函数（具体声明和参数见文后源代码，以下同）：

CreateRoundRectRgn：建立圆角矩形区域，其参数分别为左上角及右下角坐标，还有圆角直径等（当圆角直径接近或超过矩形的长度时，将呈现为圆或椭圆形）；

CreatePolygonRgn：建立多边形区域，参数为描述该区域的边界点坐标以及边界点个数，本程序用来建立三角形窗体；

SetWindowRgn：完成一个特殊形状窗口的设置（具体形状由上面两个函数指定）。

本程序中的信息提示窗体就是用上面列出的 API 函数实现的。

‘大眼夹’窗体的‘推镜头’效果

用到两个 API 函数：

SetRect：指定一个矩形区域，该区域的左上、右下坐标由参数传入；

DrawAnimatedRects：完成窗体从源区域到目标区域的动画，需要进行动画的窗体和源区域、目标区域由参数指定，

‘大眼夹’窗体就是在用户的控制下，用这两个函数从指定地方开始动画，直到完全显示出来。

设置最前窗体

程序运行后，无论用户怎么调动其他窗体，‘大眼夹’窗体和信息提示窗体始终在所有窗体的最前面，实现这一功能需要下面的 API 函数：

SetWindowPos：设置指定窗体的位置，由 hWndInsertAfter 参数指定该窗体始终靠前。

‘透明’按钮

在信息提示窗体中有一个‘确定’按钮，本来可以引入 VB 提供的 CommandButton 控件，但笔者为了更加美观，自制了‘透明’按钮。具体是先建立一个背景色为窗体背景的 Label 控件，然后在其四周建立四个 Line 控件，由 Label 控件的 Click 事件来决定 Line 控件的颜色，造成可以‘凸起’和‘凹下’的效果。

实现过程

建立信息提示窗体

启动 VB5，新建一‘标准 EXE’工程，将 Forn1 窗体的背景设为黄色，名称设为 frmMsg，在其上添加一个圆角矩形的 Shape 控件，如图一建好其他 Image 控件、Label 控件、line 控件，并如前所述建立‘确定’按钮，其中 Label 控件的名称设为 LabYes，四周 Line 控件的颜色，左、上为淡色，右、下为深色。在代码窗口中加入下列代码：

```
Public ClrLgt As Long
Public ClrDrk As Long
´ = = =| 载入信息提示窗体 | = = =
Private Sub Form_Load()
Dim lngRgn As Long, lngWdn As Long
Dim lngCtrX As Long, lngCtrY As Long, lngStepX As Long, lngStepY As Long
    ClrLgt = Line2.BorderColor
    ClrDrk = Line4.BorderColor
```

```
    lngRgn = CreateRoundRectRgn(10, 25, Me.ScaleWidth
– 2, Me.ScaleHeight + 11, 42, 42)
    lngwnd = SetWindowRgn(Me.hWnd, lngRgn, True)
    ´设置窗体的中心坐标
    lngCtrX = Screen.Width / Screen.TwipsPerPixelX / 2
    lngCtrY = Screen.Height / Screen.TwipsPerPixelY / 2
– 100
    lngStepX = 200
    lngStepY = 110
Me.Move (lngCtrX – lngStepX) * Screen.TwipsPerPixelX,
(lngCtrY – lngStepY) * Screen.TwipsPerPixelX, 2 * lng-
StepX * Screen.TwipsPerPixelX, 2 * lngStepY *
Screen.TwipsPerPixelY
    SetWindowPos Me.hWnd, HWND_TOPMOST, 0, 0, 0,
0, SWP_NOMOVE Or SWP_NOSIZE
End Sub
´ = = = | 单击‘确定’按钮 | = = =
Private Sub labYes_Click()
Animate_Unload_Assistant lngCurX, lngCurY
blnFirstDisp = True
End Sub
Private Sub labYes_MouseDown(Button As Integer, Shift As
Integer, X As Single, Y As Single)
SetLineClr ClrDrk, ClrLgt
End Sub
´ = = = | 设置‘确定’按钮的边缘颜色 | = = =
Private Sub SetLineClr(ByVal Clr1 As Long, ByVal Clr2 As
Long)
Line2.BorderColor = Clr1
Line3.BorderColor = Clr1
Line4.BorderColor = Clr2
Line5.BorderColor = Clr2
End Sub
Private Sub labYes_MouseUp(Button As Integer, Shift As In-
teger, X As Single, Y As Single)
SetLineClr ClrLgt, ClrDrk
End Sub
```

然后添加一窗体，名称设为 frmMsgEx，在窗体上添加两个 Line 控件，排列形状如图一中信息提示窗体下部的“尖角”，在代码窗口中加入下列代码：

```
Private Sub Form_Load()
Dim rgnPoint(1 To 3) As Point
Dim lngRgn As Long, lngWdn As Long, X As Long, Y As
Long
rgnPoint(1).X = 58
rgnPoint(1).Y = 27
rgnPoint(2).X = 77
rgnPoint(2).Y = 27
rgnPoint(3).X = 77
rgnPoint(3).Y = 66
lngRgn = CreatePolygonRgn(rgnPoint(1), 3, 1)
lngWdn = SetWindowRgn(Me.hWnd, lngRgn, True)
DeleteObject (lngRgn)
X = frmMsg.Left + frmMsg.lneSpc.X1 * Screen.TwipsPer-
PixelX – 870
Y = frmMsg.Top + frmMsg.lneSpc.Y1 * Screen.TwipsPer-
PixelY – 120
Me.Move X, Y
SetWindowPos Me.hWnd, HWND_TOPMOST, 0, 0, 0, 0,
SWP_NOMOVE Or SWP_NOSIZE
End Sub
```

建立“大眼夹”窗体

在工程中添加一窗体，名称设为 frmAssistant，在其上添加三个 Image 控件，名称都为 ImgAst，Index 属性分别设为 0、1、2，Picture 属性中分别添加三副可以连成动画的图像（当然任意三副图像都可以），载入图像后这三个 Image 控件的大小要调整至相等，然后将它们重叠后放置于窗体的左上角，调整窗体大小直至刚好显示满 Image 控件中的图像（如图一所示）；添加一个定时器控件，Interval 属性设为 500，在代码窗口中加入下列代码：

```
Public lngDispFlag As Long
Private Sub Form_Load()
lngDispFlag = 0
SetWindowPos Me.hWnd, HWND_TOPMOST, 0, 0, 0, 0,
SWP_NOMOVE Or SWP_NOSIZE
End Sub
Private Sub Form_Unload(Cancel As Integer)
Animate_Unload_Assistant lngCurX, lngCurY
blnFirstDisp = True
End Sub
Private Sub Timer1_Timer()
imgAst(lngDispFlag).Visible = True 0
imgAst((lngDispFlag + 1) Mod 3).Visible = False
imgAst((lngDispFlag + 2) Mod 3).Visible = False
lngDispFlag = lngDispFlag + 1
If lngDispFlag = 3 Then lngDispFlag =
End Sub
```

建立主控窗体

如前所述在工程中添加一窗体，名称设为 frmMain，由于篇幅有限在此只介绍主要控件，其它提示性的控件均为 Label 控件，读者可根据图一具体安排）。如图一所示添加两个上下排列的 Frame 控件，Caption 分别为‘动画产生于”、“动画也可产生于用户指定位置”；在上面的 Frame 控件中添加九个 CommandButton 控件，排列为三行三列，名称全部设为 cmdDisp，按照从上到下、从左到右的顺序将它们的 Index 属性依次设为 0、1、2、...7、8，Caption 属性依次设为‘屏幕左上角”、‘屏幕正左边”、‘屏幕左下角”、‘屏幕正上方”、‘按钮本身”、‘屏幕正下方”、‘屏幕右上角”、‘屏幕正右边”、‘屏幕右下角”；在下面的 Frame 控件中添加两个上下排列的 TextBox 控件，名称分别为 txtSetX、txtSetY，在右边添加一 CommandButton 控件，名称设为 cmdSet，Caption 设为‘产生动画”；在窗体左边添加一 Label 控件，名称设为 LabScrRsl，背景为黑色，前景色为绿色。在代码窗口中加入下列代码：

```
'====| 单击按钮 |====
Private Sub cmdDisp_Click(Index As Integer)
Dim i As Long
If Not blnFirstDisp Then Animate_Unload_Assistant lngCurX, lngCurY
blnFirstDisp = False
If Index <>4 Then
    '计算屏幕边缘八个方位的起始坐标
    lngCurX = (Index \ 3) * lngScrRslX / 2
    lngCurY = (Index Mod 3) * lngScrRslY / 2
Else
lngCurX = (cmdDisp(4).Left + Me.Left + cmdDisp(4).Width / 2) / Screen.TwipsPerPixelX + 200
lngCurY = (cmdDisp(4).Top + Me.Top + cmdDisp(4).Height / 2) / Screen.TwipsPerPixelY + 50
End If
For i = -5000 To 5000
    DoEvents
Next i
Show_Msg
Animate_Disp_Assistant lngCurX, lngCurY
End Sub
'====| 自定义产生动画 |====
Private Sub cmdSet_Click()
Dim i As Long
If IsNumeric(txtSetX.Text) And IsNumeric(txtSetY.Text) Then
Else
    MsgBox "输入坐标中有非法字符!"
    txt_init
    Exit Sub
End If
If Val(txtSetX.Text) < 0 Or Val(txtSetX.Text) > lngScrRslX Or Val(txtSetY.Text) < 0 Or Val(txtSetY.Text) > lngScrRslY Then
    MsgBox "输入的坐标值不符合显示器的分辨率!"
    txt_init
    Exit Sub
End If
If Not blnFirstDisp Then Animate_Unload_Assistant lngCurX, lngCurY
blnFirstDisp = False
lngCurX = Val(txtSetX.Text)
lngCurY = Val(txtSetY.Text)
For i = -5000 To 5000
    DoEvents
Next i
Show_Msg
Animate_Disp_Assistant lngCurX, lngCurY
End Sub
'====| 载入窗体 |====
Private Sub Form_Load()
blnFirstDisp = True
lngScrRslX = Screen.Width / Screen.TwipsPerPixelX
lngScrRslY = Screen.Height / Screen.TwipsPerPixelY
labScrRsl.Caption = CStr(lngScrRslX) + "X" + CStr(lngScrRslY)
Me.Move 3000, 400
Dest_Left = lngScrRslX / 2
Dest_top = lngScrRslY * 13 / 24
End Sub
'====| 卸载窗体 |====
Private Sub Form_Unload(Cancel As Integer)
Unload frmAssistant
Unload frmMsg
Unload frmMsgEx
End Sub
'====| 分辨率文本框初始化 |====
Private Sub txt_init()
txtSetX = "0"
txtSetY = "0"
End Sub
```

建立一个标准模块

在当前工程中添加一“标准模块”，名称设为 modCtl，在其代码窗口中加入下列代码：

```
Public Type Point
    X As Long
    Y As Long
End Type
'声明显示椭圆型窗体和三角型窗体所需的 API 函数
Public Declare Function CreateRoundRectRgn Lib "gdi32" (ByVal X1 As Long, ByVal Y1 As Long, ByVal X2 As Long, ByVal Y2 As Long, ByVal X3 As Long, ByVal Y3 As Long) As Long
Public Declare Function SetWindowRgn Lib "User32" (ByVal hWnd As Long, ByVal hRgn As Long, ByVal bRedraw As Boolean) As Long
Public Declare Function CreatePolygonRgn Lib "gdi32" (lpPoint As Any, ByVal nCount As Long, ByVal nPolyFillMode As Long) As Long
Public Declare Function DeleteObject Lib "gdi32" (ByVal hObject As Long) As Long
Public Const HWND_TOPMOST = -1
Public Const SWP_NOSIZE = &H1
Public Const SWP_NOMOVE = &H2
Public Declare Sub SetWindowPos Lib "User32" (ByVal hWnd As Long, ByVal hWndInsertAfter As Long, ByVal X As Long, ByVal Y As Long, ByVal cx As Long, ByVal cy As Long, ByVal wFlags As Long)
'声明窗体移动动画所需的数据结构和 API 函数
Public Type RECT
    Left As Long
    Top As Long
    Right As Long
    Bottom As Long
End Type
Public Const IDANI_CAPTION = &H3
Public Declare Function SetRect Lib "User32" (lpRect As RECT, ByVal X1 As Long, ByVal Y1 As Long, ByVal X2 As
```

```
Long, ByVal Y2 As Long) As Long
Public Declare Function DrawAnimatedRects Lib "User32"
(ByVal hWnd As Long, ByVal idAni As Long, lprcFrom As
RECT, lprcTo As RECT) As Long
'自定义助手图像窗体移动的有关常量和变量
Public Const ASSISTANT_WIDTH = 127    '目标区域的宽度
Public Const ASSISTANT_HEIGHT = 113   '目标区域的高度
Public Dest_Left As Long    '目标区域的左、上坐标
Public Dest_top As Long
Public Assistant_Dest As RECT
Public Assistant_Source As RECT
Public lngCurX As Long    '当前水平坐标
Public lngCurY As Long    '当前垂直坐标
Public blnFirstDisp As Boolean
'自定义屏幕的分辨率变量
Public lngScrRslX As Long
Public lngScrRslY As Long
'=====| 显示提示信息窗体 |=====
Public Sub Show_Msg()
    frmMsg.Show
    frmMsgEx.Show
End Sub
'===| 动画显示助手窗体 |===
Public Sub Animate_Disp_Assistant(ByVal Start_X As Long,
ByVal Start_Y As Long)
SetRect Assistant_Source, Start_X, Start_Y, Start_X, Start_Y
SetRect Assistant_Dest, Dest_Left, Dest_top, Dest_Left +
ASSISTANT_WIDTH, Dest_top + ASSISTANT_HEIGHT
DrawAnimatedRects frmAssistant.hWnd, IDANI_CAPTION,
Assistant_Source, Assistant_Dest
    frmAssistant.Move Dest_Left * Screen.TwipsPerPixelX,
Dest_top * Screen.TwipsPerPixelX, ASSISTANT_WIDTH *
Screen.TwipsPerPixelX, ASSISTANT_HEIGHT *
Screen.TwipsPerPixelY
    frmAssistant.Show
End Sub
'===| 动画卸载助手窗体 |===
Public Sub Animate_Unload_Assistant(ByVal End_X As Long,
ByVal End_Y As Long)
    SetRect Assistant_Dest, End_X, End_Y, End_X, End_Y
    SetRect Assistant_Source, Dest_Left, Dest_top, Dest_Left
+ ASSISTANT_WIDTH, Dest_top + ASSISTANT_HEIGHT
        DrawAnimatedRects frmAssistant.hWnd, I-
DANI_CAPTION, Assistant_Source, Assistant_Dest
    frmAssistant.Move End_X * Screen.TwipsPerPixelX,
End_Y * Screen.TwipsPerPixelX, 0, 0
    frmMsg.Hide
    frmMsgEx.Hide
    frmAssistant.Hide
End Sub
```

操作说明

程序运行后，鼠标单击上面 Frame 控件中的任一按钮，“Word 助手”动画即从相应的位置显示，用户也可在下面 Frame 控件中指定 X 坐标、Y 坐标，然后单击‘产生动画”，注意输入的坐标必须符合左边显示的分辨率 在程序启动时自动获取 。

本程序在 Win98 / VB5 中通过，效果颇佳。

收稿日期：2000 年 10 月 17 日）

邮件病毒遭遇克星

一种能对电子邮件病毒进行实时监控并自动清除的反病毒软件在我国研制成功并开始投放市场。这是我国反病毒产品首次具备这一重要功能。

我国著名的反病毒软件公司——北京瑞星公司去年 12 月 26 日在北京宣布，其最新一代反病毒产品瑞星杀毒软件新品 2001 版实现了多项重大技术突破和功能提升，其中包括有效地预防和杀灭邮件病毒的重要功能。当用户在发送、接收和打开电子邮件时，瑞星杀毒软件新品 2001 版能自动对电子邮件及其附件进行监控，一旦发现病毒将自动清除，确保用户的邮件无毒并且可用。

在不久前由公安部进行的所有杀毒软件评测中，瑞星领先众多国内外的优秀反病毒软件，名列第一。其产品后又被确认为一级品。在瑞星杀毒软件新品 2001 版中，还增加了多项业内首创的新特性和新功能，例如，用户只需要点击软件界面上的一个按钮，就可以实现软件的自动升级，而不必去厂商的网站上下载升级程序包，也免去了许多复杂、繁琐的升级操作，极大地提升了反病毒软件的易用性。

瑞星杀毒软件是瑞星公司针对各种流行于国内外危害较大的恶性计算机病毒和‘黑客’程序，自主研制开发的病毒检测和清除工具。瑞星产品先后荣获‘国家级科技成果奖”、‘国家重点新产品’奖，并被国家科委列入 1998 年国家级‘火炬计划’项目。

上接第 18 页）

```
    : : UnregisterHotKey(m_hWnd, DEFAULT_DESKTOP_ID);
    : : UnregisterHotKey(m_hWnd, SAMPLE_DESKTOP_ID);
    m_sdtSample.Close();
}
```

通过以上的阐述，我们介绍了如何在不重新启动系统的情况下，在桌面间进行切换的实现技术。给出了一个 C++类，同时以一个 Visual C++ 的示例说明了该类的使用方法。希望能够对在开发应用中需要实现这一功能的程序员有所帮助和启发。

参考文献

1. David Bennett 著，贺军等译 . Visual C++ 5 开发人员指南[M] . 北京：机械工业出版社，1999. 03

2. David J. Kruglinski 等著 . Programming Visual C++6. 0 技术内幕 第五版）[M]. 北京：希望电子出版社，1999. 05

收稿日期：2000 年 10 月 12 日）

ASP 程序中利用 MsChart 实现 WEB 数据库中数据的复杂图表显示

王志强 杨 进

摘 要 本文以一简单的例子说明了如何在 ASP 程序中利用 MSCHART 实现数据库中数据的复杂图表显示，MsChart 是微软制作的功能强大的图表工具，用它可以很方便地建立各种图表。制作各种 3 维、2 维的直方图、折线图。

关键词 WEB 数据库，ASP，MSCHART

一、 设计背景

本人在进行基于 WEB 的学校评估管理 MIS 系统开发过程中，碰到要把数据库中对学校的评分结果数据进行图表显示，我利用 mschart 这个 Activex 控件方便实现了各种图表的显示，如各种 2 维、3 维直方图、折线图等。本文以程序中以学校教学条件得分为例，用 mschart 实现图表的显示 为了简单起见，起抛砖引玉作用，程序中省去原程序中的一些功能。

二、 程序所用数据库数据表的结构

Access2000

学校名称	文本
评估项目	文本
评估员姓名	文本
得分	数字

三、 MSChart 主要属性说明

1. TitleText 属性：图表显示的标题
2. ColumnCount 属性：图表显示的总列数
3. RowCount 属性：图表显示的总行数
4. Row 属性：图表当前所在的行数
5. Column 属性：图表当前所在的列数
6. RowLabel 属性：图表当前所在的行标题
7. ColumnLabel 属性：图表当前所在的列标题
8. chartType 属性：图表显示方式，具体值如下：1：二维直方图；0：三维直方图；3：二维折线图；2：三维折线图；5：二维面积图；4：三维面积图；14：饼图。
9. data 属性：设置在行和列的数据。

四、 程序设计过程

1. 建立 web 数据库的联结

```
Set Conn =Server. CreateObject(″ADODB. Connection″)
Connstr =″ DBQ =″ +server. mappath( ″ cqdb. mdb″) +″;
DRIVER ={Microsoft Access Driver ( * . mdb)};″
Conn. Open connstr
session ‘pgxxmc ’= ‘重庆计算机学校 ’设定要查寻和显示的学校名称
```

2. 提取数据中所需要的数据

```
dim pgxm(10) ′评估项目数组
dim pgname(10)′评估员姓名数组
dim pgdata(10, 10)′评估得分存放每一评估项目不同评估员的评分值
set rs =server. createobject(″adodb. recordset″)
set rs1 =server. createobject(″adodb. recordset″)
sqlstr1 =″select [评估项目] from 评分表 where 评估学校 =′″& session(″pgxxmc″) & ″′group by [评估项目]″
sqlstr =″select [评估员姓名] from 评分表 where 评估学校 =′″& session(″pgxxmc″) & ″′group by [评估员姓名]″
rs. open sqlstr, conn, 1, 3
colcount =rs. recordcount ′图表列数就是评估员姓名名数
rs1. open sqlstr1, conn, 1, 3
rowcount =rs1. recordcount′图表行数也就是评估项目名数
i =1
do while not rs. eof ′获取评估员姓名并存放于数组之中
  pgname(i) =rs(0)
  i =i +1
 rs. movenext
loop
rs. close
i =1
do while not rs1. eof ′获取评估项目并存放于数组中
  pgxm(i) =rs1(0)
  rs1. movenext
  i =i +1
  loop
for j =1 to rowcount′保存每一评估项目每一评估员的得分 变量 i 为项目数下标, j 为评估员数下标
   for i =1 to colcount
sqlstr =″select * from 评分表 where 评估员姓名 =′″ & pgname(i) & ″′and 评估项目 =′″ & pgxm(j) & ″′and 评估学校 =′″& session(″pgxxmc″) & ″′″
     rs. open sqlstr, conn, 1, 3
     if not rs. eof then
```

```
pgdata(j, i) =rs("得分") '获取每评估项目,不同评估员的评分值
      else
      pgdata(j, i) =0 '若不存在为 0
     end if
     rs. close
   next
 next
 rs1. close
 %>
```

说明：因为评估员的姓名数量是动态的，所以要对数据库中的评估员进行分组获得当前评估员的数目和具体的姓名，评估项目也是相同的原因所以要进行分组。

3. 生成图表所需数据的隐藏表单

```
<html>
<head>
<meta http -equiv = "Content -Type" content = "text/html; charset =gb2312">
<meta name = " GENERATOR" content = " Microsoft Front-Page 4. 0">
<meta name = " ProgId" content = " Front-Page. Editor. Document">
<title>New Page 1 </title>
</head>
<body bgcolor = "#FFCCFF">
<form name =form1 id =form1>
 <input type = " hidden" value = <% =colcount%> name = col>
 <input type = "hidden" value = <% =rowcount%> name = row>
 <% for i =1 to colcount%>
 <input type = " hidden" value = <% =pgname(i)%> name =pgname>
 <%next%>
 <% for i =1 to rowcount%>
 <input type = " hidden" value = <% =pgxm(i)%> name = pgxm>
 <%next%>
 <% for i =1 to rowcount
    for j =1 to colcount %>
    <input type = " hidden" value = <% =pgdata(i, j)%> name =pgdata>
  <% next
 next %>
</form>
```

说明：本部分生成的隐藏数据表单，用于向图表传递相应数据库中提取出的数据。

4. 图表的显示

```
<p align = "center">
<object 此处插入 mschart activex 控件></object>
</p>
<SCRIPT ID =clientEventHandlersJS LANGUAGE = javascript>
<! - -
MSChart1. TitleText = "教学条件评估得分图表显示"
MSChart1. ShowLegend =true
MSChart1. ColumnCount =document. form1. col. value '设定图表的列数
MSChart1. RowCount =document. form1. row. value'设定图表的行数
k =0
for(i =0; i <document. form1. row. value; i + +) '设定每一行列的标题和数据
{MSChart1. row =i +1
for(j =0; j <document. form1. col. value; j + +)
 { MSChart1. Column =j +1
   MSChart1. Data =document. form1. pgdata(k). value
       MSChart1. ColumnLabel =document. form1. pgname(j). value
     k =k +1
 }
MSChart1. RowLabel =document. form1. pgxm(i). value
}
'下面子程序用于显示不同的图表类型
function select1_onchange() {
MSChart1. chartType =select1. value
MSChart1. Plot
}
// - ->
</SCRIPT>
<p><b><font color = "#000080" size = 2">选择图形显示方式 </font></b>
<SELECT id =select1 name =select1 LANGUAGE = javascript onchange = "return select1_onchange()">
<OPTION value =1>二维直方图 </OPTION>
<OPTION value =0>三维直方图 </OPTION>
<OPTION value =3>二维折线图 </OPTION>
<OPTION value =2>三维折线图 </OPTION>
<OPTION value =5>二维面积图 </OPTION>
<OPTION value =4>三维面积图 </OPTION>
<OPTION value =14>饼图 </OPTION></SELECT></p>
</body>
 <% conn. close %>
</html>
```

说明 以上原码在 pwin98 + pws 下调试通过，在实际运行时客户端需注册 Mschart Activex 控件 我采用的是 mschart20. ocx Microsoft chart control version6. 0 oledb ，把其拷贝至 windows 的 system 目录下 在 dos 下用 regsvr32 对控件进行注册。

五、小结

在掌握 Mschart Activx 控件的使用方法之后，就可以在此基础上实现更加复杂的功能，这一定会让你的程序增色不少！

收稿日期：2000 年 10 月 18 日）

借助 VB 制作拥有大量词组的五笔输入法

张昌华

本人从会打字起就一直使用五笔字型输入法，几乎用尽了所有流行的五笔字型输入法，在这万码奔腾的时代，选来选去，本人还是觉得王码五笔和陈桥智能五笔相对好用。其中王码五笔与 Windows 的兼容性很好，而且不需注册，但词组较少，输入特殊符号不方便，很多常用词组都没有；陈桥智能五笔功能强大，词汇量堪称一流，有 4 万 7 千多条，但要注册方能正常使用，在使用过程中还经常出现这样那样的错误　至少本人使用时是这样）。因此本人使用陈桥五笔很长时间后，只好再次回到王码五笔的身边。鉴于王码的词汇量少，本人决定给它增加大量词组，经过仔细摸索，本人终于采用本文的方法如愿以偿。

为了说明本方法的可取之处，我们先分析一下王码五笔增加词组的几种普通方法：

方法 1：在线造词

在输入汉字时按 crtl +　，当王码输入法左边的图标变成“词”时，输入自定义词组，再按 crtl +　实现自定义词组的添加。

方法 2：手工造词

在王码输入法上点右键，选“手工造词”，然后在弹出的造词窗口中实现自定义词组的添加。

方法 3：批量造词

打开“输入法生成器”，选“批量造词”→点击“打开文件”打开存放要添加词组的 TXT 文件　(文件内容格式为每行一个词组）→在“目标输入法”中选王码五笔→点击“自动编码”对 TXT 中的词组进行编码→点击“全选”→点击“插入词组”→关闭输入法生成器。便可完成词组的添加。

以上三种方法都比较方便，但其不足之处在于：方法 1、2 对一般人来说添加的词组量是有限的，方法 3 虽然可一次添加很多词组，但输入法生成器一次只能接受大约 1 千条的词组，采用方法 3 要想添加数量巨大的词组是很费时费力的。而且方法 1、2、3 添加的词组均存放在与码表文件同名的 *. emb 文件里，利用输入法生成器的逆转换功能也不能将其还原到码表原文件里。而且当词组添加到一定量时，会出现乱码，导致添加的词组完全损坏，本人在使用王码 4. 0 和 86 版时经常添加到几十条时就出现乱码，很伤脑筋。

因此本人采用了另一方法——“生成自己的码表文件，重新创建五笔输入法”。

首先我们将王码五笔输入法 86 版的码表文件　(Windows h system hwinwb86. mb）还原为 TXT 原文件：利用输入法生成器的“逆转换”功能，打开王码的码表文件，然后进行逆转换便可得到对应的码表原文件　(winwb86. txt）。打开码表原文件，我们便可看到其内容包括：[Description]段、[Rule]段、[Text]段，所有词条及编码均放在[Text]段里。因此我们只需在[Text]段里添加大量词组和对应编码，然后再根据码表原文件创建输入法，便可大功告成。另外说明一下，王码五笔 4. 0 版及 4. 5 版好象均不能采用逆转换功能得到码表原文件，故本人选用了王码 86 版。

为达目的，我们可按以下步骤进行：

一、得到五笔基本字符集及编码

将逆转换得到的王码码表原文件　假设为 winwb86. txt）里的[Text]段里的内容　(不含“[Text]”本身）保存在 wmwb. txt 中，其中包含有基本字符集及编码。

二、生成词组文件

为了得到大量的词组，本人将陈桥智能五笔 4. 5 的 chencyzk. chh 文本文件　在陈桥安装目录里）插入到一个空白 word 文档里，其词组格式为“蓝本　增加　正常　钻研 ……”，即每个词组间用分号+ 空格隔开，用 word 的替换功能将分号 + 空格替换成人工换行符，然后另存为纯文本文件，假设为 phrase. txt。到此便得到了 47900 多个词组。没有陈桥五笔的可到陈桥主页 wb. 126. com 去下载一个。当然读者也可通过其它手段得到词组，如将微软拼音 2. 0 的码表文件逆转换为 txt 文件，然后从其[Text]段中筛选出大量词组，其词组量达 3 万多条，远多于王码词组。

三、利用 VB 生成码表原文件

我们可利用 VB 及 Access2000 将 phrase. txt 中词组编码，再将其与 wmwb. txt 中的单字符集合并，然后根据五笔码表原文件格式生成自己的码表原文件，假设为 mywb. txt。

为了生成码表原文件 mywb. txt，请按如下步骤进行：

1. 在 Access 2000 中建立一个数据库，名称设为 wbcode. mdb，在其中建两个表，一个表为 wbsingle，一个字段为 word，文本类型，长度为 2，一个字段为 code，文本类型，长度为 4，将 word 字段设为主键，索引设为“有　(无重复）”，code 字段的索引设为“无”。再建一个表为 wbmulti，一个字段为 word，文本类型，长度为 50，一个字段为 code，文本类型，长度为 9，不要建立主键，将 word 字段索引设为“无”，code 字段的索引设为“无”，然后在工具栏上点“索引”按钮，建立名称为 code，以 code + word 为升序的组合索引，code 索引设置如下图：

请读者建库时一定要严格按上述要求，以保证程序的顺利运行。

2. **建立 VB 应用程序：在 VB 中新建一工程，在 form1 表单上添加一按钮，将 Caption 设为 “开始处理”，然后点菜单 “工程”→ “引用”，将 “Microsoft DAO 3. 6 Object Library” 引用打上√，再将如下代码复制到代码窗：**

```
Option Explicit
Private Sub Command1_Click()
  '请先用 '工程→引用 '菜单引用 Microsoft DAO 3. 6 Object Library
  Dim dbWb As Database
  Dim rstSingle As Recordset, rstMulti As Recordset
    Set dbWb = OpenDatabase(App. Path & "\ wbcode. mdb")
  '清空表中记录, 多次运行本程序时以免受上次影响
  dbWb. Execute "delete * from wbsingle"
  dbWb. Execute "delete * from wbmulti"
  Set rstSingle = dbWb. OpenRecordset("wbsingle")
  rstSingle. Index = "PrimaryKey" '主键(word)索引
  '①打开王码五笔 TXT, 将形如 '工 a aaaa '的表达式中
  '汉字( 工) 及其完全编码( aaaa) 取出入库(wbsingle 表中)
  Print "正在提取王码单字及编码入库, 请等待……"
  Dim strWordCode As String, strWord As String
  Open App. Path & "\wmwb. txt" For Input As #2
  Do Until EOF(2) ' 循环至文件尾.
    Line Input #2, strWordCode ' 读入一行数据并将其赋予变量.
    strWord = ""
    Do While True
      If Asc(strWordCode) < 0 Or Asc(strWordCode) > 127 Then
        '如果是非字母符号, 即为汉字
        strWord = strWord & Left(strWordCode, 1)
        strWordCode = Mid(strWordCode, 2)
      Else '已到字母符号, 则开始分离处理
        If Len(strWord) = 1 Then
          '仅筛选出单字, 得到基本字符集及其编码即可
          dbWb. Execute "Insert Into wbsingle values ('" _
            & strWord & "','" & Trim(Right(strWordCode, 4)) & "')"
        End If
        Exit Do
      End If
    Loop
  Loop
  Close #2
  Print "王码单字及编码入库处理完毕. "
  '②打开外来词组文件(phrase. txt), 将其词组入库(wbmulti 表中)
  Print "正在将外来词组入库, 请等待……"
  Open App. Path & "\phrase. txt" For Input As #2
  Do Until EOF(2) ' 循环至文件尾.
    Line Input #2, strWord ' 读入一行数据并将其赋予变量.
    dbWb. Execute "Insert Into wbmulti (word) values('" & strWord & "')"
  Loop
  Close #2
  Print "陈桥词组入库处理完毕. "
  '③对外来词组进行编码处理
  Print "正在对外来词组进行编码处理, 请等待……"
  Dim strCode As String, lenWord As Integer
  Set rstMulti = dbWb. OpenRecordset("wbmulti")
  Do Until rstMulti. EOF
    strWord = rstMulti! word
    lenWord = Len(strWord)
    strCode = ""
    '在 wbsingle 表中找出词组中相应的字, 并按词组编码规则取得相应编码
    Dim J As Integer, sWord As String
    For J = 1 To IIf(lenWord > 4, 4, lenWord)
      sWord = IIf(lenWord > 4 And J = 4, Right(strWord, 1), Mid(strWord, J, 1))
      rstSingle. Seek "=", sWord
      If rstSingle. NoMatch Then
        If MsgBox(sWord & " 在字符集中找不到, 终止处理吗?", 36, "提示") = vbYes Then Exit Sub
      Else
        '编码: 二字词为一二一二, 三字词为一一一二
        '      四字词为一一一一, 多字词为一一一(末) 一
        strCode = strCode _
          & Left(rstSingle! code, IIf(lenWord = 2, 2, IIf(lenWord = 3 And J = 3, 2, 1)))
      End If
    Next J
    rstMulti. Edit
    rstMulti! code = strCode
    rstMulti. Update
    rstMulti. MoveNext
  Loop
  Print "对外来词组编码完毕. "
  '④将单字符及其所有编码方式合并到词组表(wbmulti) 中
  Print "正在将单字与词组合并, 请等待……"
  Open App. Path & "\wmwb. txt" For Input As #2
  Do Until EOF(2) ' 循环至文件尾.
    Line Input #2, strWordCode ' 读入一行数据并将其赋予变量.
    strWord = ""
    Do While True
      If Asc(strWordCode) < 0 Or Asc(strWordCode) > 127 Then
        '如果是非字母符号, 即为汉字
        strWord = strWord & Left(strWordCode, 1)
```

```
        strWordCode = Mid(strWordCode, 2)
      Else
        '已到字母符号,则开始分离处理
        If Len(strWord) = 1 Then
          '仅筛选出单字,得到基本字符集及其五笔编码即可
          dbWb.Execute "Insert Into wbmulti values ('" _
              & strWord & "','" & strWordCode & "')"
        End If
        Exit Do
      End If
    Loop
  Loop
  Close #2
  Print "单字与词组合并完毕."
  '⑤根据合并后的所有词条生成输入法码表原文件(mywb.txt)
  Print "正在生成码表原文件,请等待……"
  rstMulti.Index = "code" '所有词条以编码升序索引排列
  ' 打开码表原文件,若不存在则自动创建,存在则会覆盖
  Open App.Path & "\mywb.txt" For Output As #1
  '生成码表文件前一部分,注意中间不要有任何空格
  Print #1, "[Description]"
  Print #1, "Name=五笔"
  Print #1, "MaxCodes=4"
  Print #1, "MaxElement=1"
  Print #1, "UsedCodes=abcdefghijklmnopqrstuvwxy"
  Print #1, "WildChar=z"
  Print #1, "NumRules=3"
  Print #1, "[Rule]"
  Print #1, "ca4=p11+p21+p31+n11"
  Print #1, "ce2=p11+p12+p21+p22"
  Print #1, "ce3=p11+p21+p31+p32"
  Print #1, "[Text]"
  '将合并后的所有词条加入
  rstMulti.MoveFirst
  Do Until rstMulti.EOF
      Print #1, rstMulti!word & rstMulti!code
      rstMulti.MoveNext
  Loop
  Close #1
  rstMulti.Close
  Print "码表原文件生成完毕."
  MsgBox "恭喜您,已成功生成码表原文件 mywb.txt."
  dbWb.Close
End Sub
```

3. 运行程序：将程序保存到某一目录，文件名随便取，再将步骤 1 生成的 wmwb.txt 和步骤 2 生成的 phrase.txt 文件与程序放于同一目录。然后运行程序，几分钟后运行完毕，将在该目录下产生一码表原文件 mywb.txt，至此一个拥有 58000 多词条的五笔码表原文件便诞生了。

四、创建自己的五笔输入法

打开输入法生成器，在“创建输入法”选项卡内，指定码表原文件为 mywb.txt，“输入法码表文件名”最好改为“c:\windows\system\mywb.mb”，然后点“转换”按钮转换生成码表文件 mywb.mb，最后点“创建”按钮创建自己的五笔输入法，在后面弹出的窗口中一直按确定，最后一个窗口提示“输入法已生成，是否安装？”，选“是”。然后关闭输入法生成器，马上就可用自己的五笔输入法了。

有兴趣的读者还可将特殊符号 如“◆”等 按自己喜欢的方式进行编码，然后将其加入到 wmwb.txt 中，再按上述方法生成自己的输入法，你的五笔就可直接输入特殊符号了，本人已成功应用，因这不是本文重点，故略。

本程序在 Win9X，VB6.0 下通过，本方法产生的五笔输入法经多人使用，反应很好。只要将 mywb.txt 码表原文件保存好，可随时在任何 Win9X 系统中生成输入法，真是很方便。若不愿自己制作的朋友，也可 Email TO js_zch@sina.com 向本人索求已制作好的 mywb.txt。该方法稍加变通还可创建自己的表形码等有码表文件的输入法。

（收稿日期：2000 年 10 月 19 日）

金洪恩牵手 McAfee 杀毒之星登陆中国

日前，国内著名教育软件开发商金洪恩公司与世界上最大的网络安全公司——美国网络联盟公司（NAI）正式结盟，于近日，联合推出一款带有全新防病毒理念的反病毒软件——McAfee 杀毒之星 5.13 版。

NAI 是全球第五大独立软件公司，也是世界上最大的致力于网络安全的公司。NAI 公司出品的 McAfee 杀毒软件是全球防病毒第一品牌，在全球 150 多个国家拥有七千万用户，据 2000 年 11 月 9 日 IDC 的最新统计表明：McAfee 杀毒之星在全球市场占有率已达 47%，《财富》排名前 1000 位的企业有 80% 都采用 McAfee 作为自己的反病毒保护神，堪称世界杀毒权威。

McAfee 杀毒之星具有强大的病毒查杀能力，其查杀病毒种类在 57 000 种以上，100% 查杀已知病毒；它采用先进的双重智能扫描引擎及时发现新病毒、可疑病毒和未知病毒，在病毒发作之前有效隔离感染文件；PSDE 引擎则高效查杀各种变形病毒；采用新一代扫描引擎 Virtran 定点扫描技术使扫描速度提高 70 - 90%。

为配合中文版 McAfee 杀毒之星 5.13 版的上市，金洪恩公司专门制作了一款防病毒教育软件——洪恩毒盾，详细介绍各种病毒、防病毒知识以及 McAfee 杀毒之星的使用方法。同时 一向注重服务的金洪恩公司还将开通一专门服务网站——mcafee.hongen.com，并在国内建立 300 多个升级服务站。

金洪恩与国际杀毒权威 NAI 结盟，把先进杀毒技术、遍布全国的软件零售渠道与完善的服务体系进行整合，必将对国内杀毒软件市场格局产生重大冲击。

多用户环境下实现 Excel 的文件管理

李激扬 任秋宇

摘　要　Excel 的文件通常是一个单位管理数据的重要基石，在多用户环境下对文件进行加密和方便地打开是单位管理重要环节。本文用 VBA 通过面向登录用户的个人工作表的形式来解决多用户环境下 Excel 的文件管理。在用户新建文件时提示保存格式，并将用户保存的文件信息回送至个人工作表。在保存有文件信息的个人工作表上双击文件名区域模拟资源管理器方式直接打开文件。

主题词　Excel，VBA，多用户，文件管理，自运行宏

在多用户的环境下，保护 Excel 文件不被非法打开的最好方法便是通过保存时用密码方式进行加密。但加密文件打开的时候，便有烦人的密码输入框提示你输入密码。当一个加密后的文件若干天以后打开的话又会有忘记密码的困惑。而对于文件的加密又不能老用单一的一个密码加密所有的文件，这样一旦密码泄漏时，加密文件的安全性得不到保证。

这个程序通过 Excel 的工作表形式管理 Excel 内的文件。让用户在多用户的状态下通过密码登录，在面向登录用户个人的工作表中管理文件。新建 Excel 文件时，提示用户可用自定义密码或随机密码保存文件，并将文件信息回送至个人工作表，当用户需要打开在个人工作表上保存的文件时只需双击文件名即可打开文件，而无需键入密码。

一、 系统开发前的准备

1. 新建 “测试多用户输入界面 . xls ” 文件；

2. 保留 “Sheet 1 ” 工作簿，删除多余 “Sheet 2 ~ Sheet n ” 工作簿；

3. 插入表名为 “登录用户资料 ” 工作表，在单元格 A1 和 B1 中分别键入 “姓名 ” 和 “登录密码 ”。

二、 用户登录界面的开发

1. 用户登录界面时实现的功能

当启动 Excel 时，通过用户登录界面上下拉列表框中用户登录名的选择，进入各用户特定的工作表；也可通过新建用户的方式添加新用户进入系统。各用户进入系统时必须提供密码，当输入密码三次错误时，系统退出。

2. 用户登录界面窗体设定

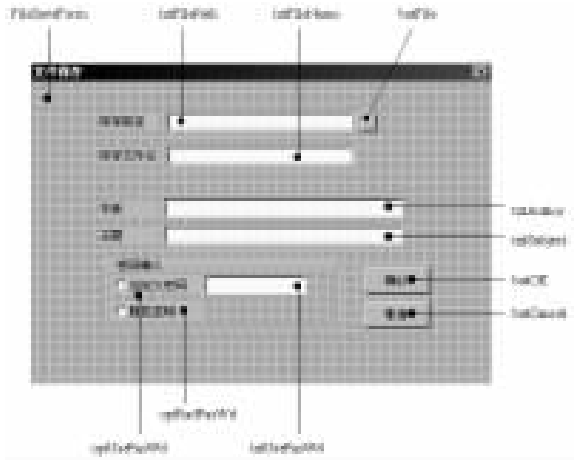
在 “测试多用户输入界面 ” 文件中打开 Visual Basic 编辑器，插入下图所示两个窗体并设定各控件名称。

3. 用户登录界面窗体程序设计

3. 1 UserSelectForm 表单的程序设计

3. 1. 1 确定按钮 ComfirmButton 的代码

通过下拉列表中用户的选择，系统在登录用户资料工作表中查询是否存在该用户。为此设定义变量 uNewCell 初始指向登录用户资料工作表 Range “A2 ” 登录用户姓名区域，而后通过 OFFSET 属性使 uNewCell 变量指向下一用户姓名，通过 LOOP 循环进行比较用户名。



```
Private Sub ComfirmButton_Click()
'通过下拉列表中用户的选择, 进入用户密码确认表单
'如果通过键入字符的方式登录用户时, 系统进行判别是否系统存在用户, 如果不是系统存在用户, 显示警示框。
'如果在下拉列表框失控时, 系统显示必须选择用户警示。
Dim uNewCell As Range, uNextNewCell As Range
Set uNewCell = Worksheets("登录用户资料").Range("A2")
If UserSelectForm.UserSelectComboBox.Value = "" Then
MsgBox prompt:="请选择登录用户名!", Title:="警示"
Else
Do While Not IsEmpty(uNewCell)
If UserSelectForm.UserSelectComboBox.Value = uNewCell.Value Then
UserPaswdForm.Show
Exit Sub
End If
Set uNextNewCell = uNewCell.Offset(1, 0)
Set uNewCell = uNextNewCell
Loop
```

```
MsgBox prompt: =″您输入的用户名不存在于系统中!″ & _
Chr(13) & Chr(13) & ″请选择添加新用户按钮!″, Title: =″
警示″, Buttons: =vbExclamation
End If
  End Sub
```

3.1.2 新用户按钮 NewUserButton 的代码

```
  Private Sub NewUserButton_Click()
´用户选择添加新用户时,判断用户名是否已存在于系统中。
´若用户名不在系统中,为新用户名,建议输入三位以上用户
登录密码,并进行密码确认。
    ´添加新的用户个人工作表,并设置工作表。
Dim NewUserName As String
Dim NewPasWd As String, NewPasWdAgan As String
Dim NewCell As Range, NextNewCell As Range
Dim NewWokset As Worksheet
Const MyTitle = ″新用户登录″
Set NewCell = Worksheets(″登录用户资料″).Range(″A2″)
InputName:
NewUserName = InputBox(prompt: =″请输入新用户姓名″,
Title: =MyTitle)
If NewUserName = ″″ Then Exit Sub
 Do While Not IsEmpty(NewCell)
 If NewUserName = NewCell.Value Then
MsgBox prompt: =″您输入的用户名已经存在于系统中!″ & _
      ″请核查后重新输入!″, Title: =″警示″
      GoTo InputName
   End If
   Set NextNewCell = NewCell.Offset(1, 0)
   Set NewCell = NextNewCell
  Loop
  NewPasWd = InputBox(prompt: =″请输入超过三个字符
用户密码″, Title: =MyTitle)
If NewPasWd = ″″ Then Exit Sub
Do While Len(NewPasWd) < 3
MsgBox prompt: =″请输入超过三个字符的登录密码!″, Ti-
tle: =″警示″
NewPasWd = InputBox(prompt: =″请输入超过三个字符用
户密码″, Title: =MyTitle)
Loop
NewPasWdAgan = InputBox(prompt: =″请输入确认密码″,
Title: =MyTitle)
Do While NewPasWd <> NewPasWdAgan
   MsgBox ″请输入正确密码!″, Title: =″警示″
   NewPasWd = InputBox(prompt: =″请重新输入用户密
码″, Title: =MyTitle)
  Do While Len(NewPasWd) < 3
MsgBox prompt: =″请输入超过三个字符的登录密码!″, Ti-
tle: =″警示″
NewPasWd = InputBox(prompt: =″请输入超过三个字符用
户密码″, Title: =MyTitle)
   Loop
NewPasWdAgan = InputBox(prompt: =″请输入确认密码″,
Title: =MyTitle)
Loop
NewCell.Offset(0, 1).Value = NewPasWdAgan
UserSelectForm.UserSelectComboBox.Value = NewUser-
Name
NewCell.Value = NewUserName
Application.ScreenUpdating = False
Set NewWokset = Worksheets.Add ´新增工作表
NewWokset.Move after: =Sheets(Sheets.Count) ´新增工
作表移至所有表后
NewWokset.Name = NewUserName ´新增工作表表名为新
建用户名
´设置新增工作表表头
NewWokset.Range(″A1″).Value = ″序号″
NewWokset.Range(″B1″).Value = ″文件名″
NewWokset.Range(″C1″).Value = ″作者″
NewWokset.Range(″D1″).Value = ″主题″
NewWokset.Range(″E1″).Value = ″保存路径″
NewWokset.Range(″F1″).Value = ″有无密码″
NewWokset.Range(″G1″).Value = ″密码″
´设置工作表列大小
NewWokset.Columns(″A:A″).ColumnWidth = 4.38
NewWokset.Columns(″B:B″).ColumnWidth = 27.63
´将 B 列单元各设置成蓝色字符加下划线格式
Columns(″B:B″).Select
   Selection.Font.ColorIndex = 5
   Selection.Font.Underline = xlUnderlineStyleSingle
´恢复 B 列表头格式
Range(″A1″).Select
Selection.Copy
Range(″B1″).Select
Selection.PasteSpecial Paste: =xlFormats, Operation: =xl-
None, SkipBlanks: = _
False, Transpose: =False
Application.CutCopyMode = False
 NewWokset.Columns(″C:C″).ColumnWidth = 7.75
NewWokset.Columns(″D:D″).ColumnWidth = 24.75
NewWokset.Columns(″E:E″).ColumnWidth = 31.88
NewWokset.Columns(″F:F″).ColumnWidth = 8.13
NewWokset.Columns(″G:G″).ColumnWidth = 9.5
Cells.Select
With Selection
   .HorizontalAlignment = xlCenter
End With
Range(″A1″).Select
UserSelectForm.Hide
ActiveWindow.WindowState = xlMaximized
Application.ScreenUpdating = True
 Call InitializeApp
End Sub
```

3.1.3 取消按扭 CancelButton 的代码

```
Private Sub CancelButton_Click()
´按取消按钮时 UserSelectForm 表单隐藏
  UserSelectForm.Hide
End Sub
```

3.1.4 表单 UserForm 激活时的代码

```
Private Sub UserForm_Activate()
´当程序启用时,工作表″登录用户资料″激活.
```

```
'并将下拉列表中的值设定为"登录用户资料"表中 a2: a30 的值.
'a2: a30 的范围可根据用户多少来设定范围
    Worksheets(Sheet & "登录用户资料"). Activate
    UserSelectComboBox. RowSource = "a2: a30"
End Sub
```

3. 1. 5 下拉列表框 UserSelectComboBox 改变时的代码

```
Private Sub UserSelectComboBox_Change()
'当下拉列表值改变时, 激活确认按钮
    ComfirmButton. SetFocus
End Sub
```

3. 2 UserPaswdForm 表单的程序设计

3. 2. 1 确定按钮 UserPaswdCButton 的代码

```
Private Sub UserPaswdCButton_Click()
Dim currcell As Range, nextcell As Range
Dim username As String
Static CautionFlag As Integer
'设定静态变量 CautionFlag, 用于记录不正确密码登陆的次数
Set currcell = Worksheets("登录用户资料"). Range("A2")
'将选择的用户名和密码同"登录用户资料"工作表中的已有用户名和登录密码比较, 如果吻合进入相应用户工作表
    Do While Not IsEmpty(currcell)
 username = currcell. Value
If username = UserSelectForm. UserSelectComboBox. Value Then
'用户密码的比较
If CStr(UserPaswdForm. UserPaswdCTextBox. Value) = CStr(currcell. Offset(0, 1). Value) Then
Worksheets(Sheet & username). Visible = xlSheetVisible '显示特定用户工作表
Worksheets(Sheet & "登录用户资料"). Visible = xlSheetVeryHidden
UserPaswdForm. Hide
UserSelectForm. Hide
ActiveWindow. WindowState = xlMaximized
Else
CautionFlag = CautionFlag + 1
MsgBox prompt: ="请输入正确的用户登录密码!"
UserPaswdForm. UserPaswdCTextBox. SetFocus
If CautionFlag = 3 Then
MsgBox prompt: ="不要再做无为的密码输入了, 这不该是你登录的用户名!" & Chr(13) & Chr(13) & "请咨询系统管理员!", Title: ="警示", Buttons: =vbExclamation
 ThisWorkbook. Close
End If
 End If
Exit Sub
End If
    Set nextcell = currcell. Offset(1, 0)
Set currcell = nextcell
  Loop
 End Sub
```

3. 2. 2 取消按扭 UserCancelButton 的代码

```
Private Sub UserCancelButton_Click()
    UserPaswdForm. Hide
End Sub
```

3. 2. 3 密码修改按扭 UserPaChButton 的代码

```
Private Sub UserPaChButton_Click()
Dim NameCell As Range, NextNameCell As Range
Dim username As String
Dim OldUserPasWd As String, NewUserPasWd As String, NewUserPasWdAgn As String
'设定静态变量 CautionFlag, 用于记录不正确密码登陆的次数
Static CautionFlag As Integer
Const MyTitle = "用户登录密码修改"
Set NameCell =Worksheets("登录用户资料"). Range("A2")
    Do While Not IsEmpty(NameCell)
       username = NameCell. Value
If username = UserSelectForm. UserSelectComboBox. Value Then '用户修改老密码时, 必须提供老密码, 如果提供密码三次不正确, 关闭该程序
        OldUserPasWd = InputBox(prompt: ="请输入用户旧密码", Title: =MyTitle)
    If CStr(OldUserPasWd) = CStr(NameCell. Offset(0, 1). Value) Then
PasswordCheck:
      NewUserPasWd = InputBox(prompt: ="请输入超过三个字符用户新密码", Title: =MyTitle)
            Do While Len(NewUserPasWd) < 3
    MsgBox prompt: ="请输入超过三个字符的登录密码!", Title: ="警示"
    NewUserPasWd = InputBox(prompt: ="请输入超过三个字符用户密码", Title: =MyTitle)
            Loop
    NewUserPasWdAgn = InputBox(prompt: ="请输入用户确认密码", Title: =MyTitle)
          If NewUserPasWd <> NewUserPasWdAgn Then
    MsgBox prompt: ="您输入了两次不同的密码, 请重新输入!", Title: ="警示"
              GoTo PasswordCheck
          End If
    NameCell. Offset(0, 1). Value = NewUserPasWdAgn '将新的用户密码回送到登录用户资料工作表中
            UserPaswdForm. Hide
        Else
            CautionFlag = CautionFlag + 1
      MsgBox prompt: ="请输入正确的用户登录密码!"
            If CautionFlag = 3 Then
    MsgBox prompt: ="不要再做无为的密码输入了, 这不该是你登录的用户名!" & Chr(13) & Chr(13) & "请咨询系统管理员!", Title: ="警示", Buttons: =vbExclamation
              '密码三次尝试错误时退出该程序
              ThisWorkbook. Close
            End If
        End If
      Exit Sub
      End If
      Set NextNameCell = NameCell. Offset(1, 0) '指向下一单元格
      Set NameCell = NextNameCell
   Loop
```

End Sub

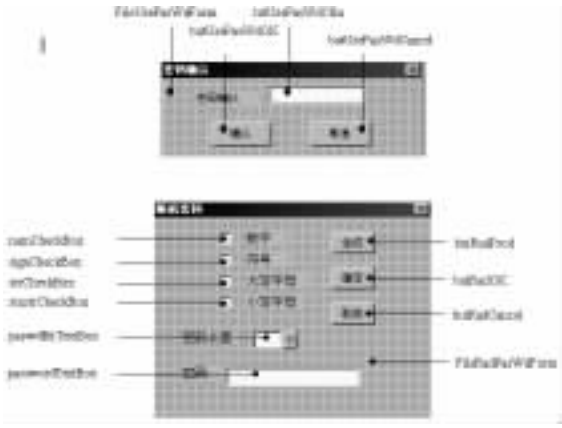
三、文件保存及其附属界面的开发

1. 文件保存及其附属界面实现的功能

在本程序运行的情况下，文件新建后调用此窗体。提供文件保存路径、文件名、自定义密码、随机密码、文件主题的输入，并将此文件信息保存到如下格式的登录用户特定工作表。

序号	文件名	作者	主题	保存路径	有无密码	密码

2. 文件保存及其附属界面窗体设定



3. 文件保存及其附属界面窗体程序设计

3.1 FileSaveForm 表单的程序设计

3.1.1 文件保存对话框按钮 butFile 的代码

通过对此文件保存按钮 FileSaveForm 的按取来调用 Excel 内置的标准 “另存为”对话框，以此获取用户完整的文件名（包括保存的路径）。调用 FullName2BookName 程序，对获取的完整文件名进行分离出不带路径的文件名。通过 InStr 函数得出不带路径的文件名字符串在完整文件名中最先出现的位置，调用 Left 函数得到用户保存的路径。对所得到的文件名和文件路径回送至相应文字框。

```
Private Sub butFile_Click()
Dim FileSaveName As String, FullName As String
'通过调用标准的“另存为”对话框，获取包含路径说明的用户文件名
FileSaveName = Application.GetSaveAsFilename(fileFilter:="Excel Files (*.xls), *.xls")
'从包含路径说明的用户文件名中提取单一文件名
FullName = FullName2BookName(FileSaveName)
'将文件名和文件保存路径回送至文件名文字框和文件路径文字框
txtFileName.Value = FullName
txtFilePath.Value = Left(FileSaveName, (InStr(FileSaveName, FullName) - 1))
End Sub
```

3.1.2 确定按钮 butOK 的代码

确定按钮 butOK_Click 主要是完成对用户新建文件的保存，并将文件资料保存到用户个人的工作表中去。对于用户选择自定义密码的选项调用自定义密码对话框。

```
Private Sub butOK_Click()
Dim LogginName As String
Dim SpCell As Range, SpNextCell As Range
Application.ScreenUpdating = False
LogginName = UserSelectForm.UserSelectComboBox.Value
txtAuthor.Value = LogginName '经获取的登录用户名缺省的回送至作者文字框
'如果用户建议保存的文件名在保存目录下一已存在建议重新设置文件名
Do While txtFileName = Dir(txtFilePath & txtFileName)
    MsgBox prompt:="您建议保存的文件名已经在该目录下存在!" & Chr(13) & Chr(13) & "请重新定义文件名!", Title:="警示", Buttons:=vbExclamation
    butFile.Value = True
Loop
If optUsePasWd.Value Then '检查用户自定义密码的复选框状态
    If Me.txtUsePasWd.Value <> "" Then
FileUsePasWdForm.Show'如存在用户密码调用密码确认窗体
    Else
NewFileCreat LogName:=LogginName'调用保存文件子程序
    End If
Else
    FileRadPasWdForm.Show'用户随机密码的窗体显示
End If
End Sub
```

3.1.3 取消按钮 butCancel 的代码

```
Private Sub butCancel_Click()
    FileSaveForm.Hide
End Sub
```

3.1.4 自定义密码 optUsePasWd 与随机密码按钮 optRadPasWd 的代码

```
Private Sub optUsePasWd_Click()
    Me.txtUsePasWd.Enabled = True
    Me.txtUsePasWd.BackColor = &H80000005
    Me.txtUsePasWd.TabStop = True
End Sub
Private Sub optRadPasWd_Click()
'当随机密码复选框被选择时，使自定义密码文字框屏蔽
    Me.txtUsePasWd.Enabled = False
    Me.txtUsePasWd.BackColor = &H8000000F
    Me.txtUsePasWd.TabStop = False
End Sub
```

3.1.5 表单 FileSaveForm 激活时的代码

```
Private Sub UserForm_Activate()
'当该窗体激活时，缺省的点击保存文件按钮和用户自定义密码复选框
'作者姓名的缺省为登录用户名
butFile.Value = True
optUsePasWd.Value = True
txtAuthor.Value = UserSelectForm.UserSelectComboBox.Value
End Sub
```

3.2 FileUsePasWdForm 表单的程序设计

3.2.1 确定按钮 butUsePasWdOK 的代码

```
Private Sub butUsePasWdOK_Click()
```

```
Application. ScreenUpdating = False
'比较在 FileSaveForm 窗体中键入的密码与 FileUsePasWdForm 窗体中键入的密码是否相同
If FileSaveForm. txtUsePasWd. Value = FileUsePasWdForm. txtUsePasWdCfm. Value Then
    NewFileCreat LogName: =UserSelectForm. UserSelectComboBox. Value, PasWd: =FileUsePasWdForm. txtUsePasWdCfm. Value
Else
    MsgBox prompt: ="您必须输入一个同上一次相同的密码!", Buttons: =vbOKOnly, Title: ="密码确认"
    FileUsePasWdForm. txtUsePasWdCfm. SetFocus
End If
End Suby
```

3.2.2 取消按钮 butUsePasWdCancel 的代码

```
Private Sub butUsePasWdCancel_Click()
  FileUsePasWdForm. Hide
End Sub
```

3.3 FileRadPasWdForm 表单程序设计

大多数同我一样在公司一机多用户上工作的同事会有相同的经历，有的文件涉及项目机密或个人隐私，对文件加密码吧，又由于思维的定式想来想去老是这几个，造一个复杂点的又想不起来，于是在这个程序中加了设计随机密码生成的窗体。在这个窗体中，用户可选择数字、大小写字母、或符号四种密码构成方式，密码的长度在一位至十位自选，这样的话对于重要文件的保护就大大增强了。

3.3.1 生成按钮 butRadProd 的代码

```
Private Sub butRadProd_Click()
'用户选择确定生成随机密码，调用生成随机密码子程序
  UserRandom
End Sub
```

3.3.2 确定按钮 butRadOK 的代码

```
Private Sub butRadOK_Click()
'用户选择确定
'屏幕显示冻结，生成随机密码，并将随机密码保存到工作簿及用户特定工作表内
UserRandom
Application. ScreenUpdating = False
'如果用户建议保存的文件名在保存目录下一已存在建议重新设置文件名
Do While FileSaveForm. txtFileName =Dir(FileSaveForm. txtFilePath & FileSaveForm. txtFileName)
  MsgBox prompt: ="您建议保存的文件名已经在该目录下存在!" & Chr(13) & Chr(13) & "请重新定义文件名!", Title: ="警示", Buttons: =vbExclamation
  butFile. Value = True
Loop
If FileRadPasWdForm. passwordTextBox. Value <> "" Then
NewFileCreat LogName: = UserSelectForm. UserSelectComboBox. Value, PasWd : = FileRadPasWdForm. passwordTextBox. Value
End If
End Sub
```

3.3.3 取消按钮 butRadCancel 的代码

```
Private Sub butRadCancel_Click()
'用户选择取消，随机密码显示表单隐藏
  FileRadPasWdForm. Hide
End Sub
```

3.3.4 复选框 numCheckBox、signCheckBox、strCheckBox、smstrCheckBox 的代码

```
Private Sub numCheckBox_click()
'当用户改变数值复选框时，对应微调项值和文本框值随之变化
  Text_SpinValue
End Sub
Private Sub signCheckBox_Click()
  Text_SpinValue
End Sub
Private Sub strCheckBox_Click()
  Text_SpinValue
End Sub
Private Sub smstrCheckBox_Click()
  Text_SpinValue
End Sub
```

3.3.5 滚动条 paswdBitSpinButton 的代码

```
Private Sub paswdBitSpinButton_Change()
'随机密码文字框值随随机密码滚动条变化而变化
  paswdBitTextBox. Value = paswdBitSpinButton. Value
End Sub
```

3.4 FileSaveForm 、FileUsePasWdForm 、FileRadPasWdForm 表单子程序设计

```
Function FullName2BookName(filename As String) As String
'要取得完整带路径文件名中的单一文件名，必须要"\"的位置，而"\"的位置较难确定(每个不同路径的"\"的位置不同)
'因此先将整个文件名倒转，然后确定第一次"\"的位置，分离出"\"之前的字符，再将分离后字符反转取的单一文件名
Dim tName As String, k As Long
'调用 ReverserStr 子程序，将完整的用户完整的用户文件名每个字符倒转
tName = ReverserStr(filename)
'k 返回字符倒转后文件名中"\"的位置
k = InStr(tName, "\")
'字符倒转后文件名"\"的位置前的字符
If k = 0 Then
  FullName2BookName = filename
Else
  tName = Left(tName, k - 1)
End If
'得到单一不带路径的文件名
FullName2BookName = ReverserStr(tName)
End Function
Function ReverserStr(str As String) As String
Dim k As Long, tstr As String
'将字符中逐个字母反转并连接后保存至 tstr
tstr = ""
For k = 1 To Len(str)
  tstr = Mid(str, k, 1) & tstr
Next k
'得到与原字符串反相的字符串
```

```
ReverserStr = tstr
End Function
Function IsBookOpen(ByVal BName As String) As Boolean
´测试用户键入的文件名是否正在使用中,若正打开则返回TRUE
Dim aBook As Object
IsBookOpen = False
BName = Trim(BName)
For Each aBook In Workbooks
    If StrComp(aBook.Name, BName, vbTextCompare) = strequal Then
        IsBookOpen = True
        Exit For
    End If
Next aBook
End Function
Function NewFileCreat(ByVal LogName As String, Optional ByVal PasWd As String)
´LogName 参数按值传递登录用户名,PasWd 传递保存文件的密码
        FullUserFileName = FileSaveForm.txtFilePath.Value & FileSaveForm.txtFileName.Value ´带路径文件名
UserFilename = FileSaveForm.txtFileName.Value ´文件名
´检查文件路径文本框中是否为空
        If FileSaveForm.txtFilePath.Value = "" Then
MsgBox prompt:="您没有输入了文件保存的路径!" & Chr(13) & Chr(13) & "请重新输入.", Title:="警示", Buttons:=vbExclamation
            Exit Function
        End If
´检查输入文件名是否在用户保存目录下已存在
        If IsBookOpen(UserFilename) Then
MsgBox prompt:="您输入了一个已经在工作簿中打开的文件名!" & Chr(13) & Chr(13) & "请重新输入文件名.", Title:="警示", Buttons:=vbExclamation
            Exit Function
        End If
´将新建文件的作者、主题等信息保存到新建文件中去
    ActiveWorkbook.Subject = FileSaveForm.txtSubject.Value
    ActiveWorkbook.Author = FileSaveForm.txtAuthor.Value
´按用户格式保存新建文件
    ActiveWorkbook.SaveAs filename:=FullUserFileName, FileFormat:=xlNormal, Password:=PasWd, WriteResPassword:="", ReadOnlyRecommended:=False, CreateBackup:=False
´***——以下为将新建文件的信息保存到登录用户的个人工作表
´标示密码的记号
        If PasWd = "" Then
            PasWDSign = "×"
        Else
            PasWDSign = "√"
        End If
        Windows("测试多用户输入界面.xls").Activate
´查询登录用户个人工作表的第一个非空单元格位置
        Set SpCell = Worksheets(LogName).Range("A1")
        Flag = 0
        Do While Not IsEmpty(SpCell)
            Set SpNextCell = SpCell.Offset(1, 0)
            Set SpCell = SpNextCell
            Flag = Flag + 1
        Loop
        If Flag = 1 Then
            SpNextCell.Value = 1 ´单元格内填入序号 1
        Else
        SpNextCell.Value = SpNextCell.Offset(-1, 0).Value + 1 ´依次在单元格内填入序号
        End If
´在用户的个人工作表通过 Offset 的偏移使 SpNextCell 向行方向记录新建文件的信息
SpNextCell.Offset(0, 1).Value=FileSaveForm.txtFileName.Value
SpNextCell.Offset(0, 2).Value=FileSaveForm.txtAuthor.Value
        SpNextCell.Offset(0, 3).Value = FileSaveForm.txtSubject.Value
        SpNextCell.Offset(0, 4).Value = FileSaveForm.txtFilePath.Value
        SpNextCell.Offset(0, 5).Value = PasWDSign
        SpNextCell.Offset(0, 6).Value = PasWd
       ActiveWindow.WindowState = xlMinimized
        Windows(FileSaveForm.txtFileName.Value).Activate
        ActiveWindow.WindowState = xlMaximized
        If FileRadPasWdForm.passwordTextBox = "" Then
            If PasWd = "" Then
                Unload FileSaveForm
            Else
                Unload FileUsePasWdForm
                Unload FileSaveForm
            End If
        Else
            Unload FileRadPasWdForm
            Unload FileSaveForm
        End If
      Application.ScreenUpdating = True
       End Function
unction UserRandom()
´检测数字、符号、大写字母、小写字母四个复写框的值来得到随机密码的构成,并和用户选定的密码长度一起调用子程序
If FileRadPasWdForm.numCheckBox And Not FileRadPasWdForm.signCheckBox And Not FileRadPasWdForm.strCheckBox And Not FileRadPasWdForm.smstrCheckBox Then
    FileRadPasWdForm.passwordTextBox = numPassWord(FileRadPasWdForm.paswdBitTextBox.Value)
ElseIf Not FileRadPasWdForm.numCheckBox And FileRadPasWdForm.signCheckBox And Not FileRadPasWdForm.strCheckBox And Not FileRadPasWdForm.smstrCheckBox Then
    FileRadPasWdForm.passwordTextBox = signPassWord
```

```
(FileRadPasWdForm. paswdBitTextBox. Value)
ElseIf Not FileRadPasWdForm. numCheckBox And Not FileRadPasWdForm. signCheckBox And FileRadPasWdForm. strCheckBox And Not FileRadPasWdForm. smstrCheckBox Then
    FileRadPasWdForm. passwordTextBox = strPassWord(FileRadPasWdForm. paswdBitTextBox. Value)
ElseIf Not FileRadPasWdForm. numCheckBox And Not FileRadPasWdForm. signCheckBox And Not FileRadPasWdForm. strCheckBox And FileRadPasWdForm. smstrCheckBox Then
    FileRadPasWdForm. passwordTextBox = smstrPassWord(FileRadPasWdForm. paswdBitTextBox. Value)
ElseIf FileRadPasWdForm. numCheckBox And FileRadPasWdForm. signCheckBox And Not FileRadPasWdForm. str
CheckBox And Not FileRadPasWdForm. smstrCheckBox Then
FileRadPasWdForm. passwordTextBox = num_signPassWord(FileRadPasWdForm. paswdBitTextBox. Value)
ElseIf FileRadPasWdForm. numCheckBox And Not FileRadPasWdForm. signCheckBox And FileRadPasWdForm. strCheckBox And Not FileRadPasWdForm. smstrCheckBox Then
FileRadPasWdForm. passwordTextBox = num_strPassWord(FileRadPasWdForm. paswdBitTextBox. Value)
ElseIf FileRadPasWdForm. numCheckBox And Not FileRadPasWdForm. signCheckBox And Not FileRadPasWdForm. strCheckBox And FileRadPasWdForm. smstrCheckBox Then
FileRadPasWdForm. passwordTextBox = num_smstrPassWord(FileRadPasWdForm. paswdBitTextBox. Value)
ElseIf Not FileRadPasWdForm. numCheckBox And FileRadPasWdForm. signCheckBox And FileRadPasWdForm. strCheckBox And Not FileRadPasWdForm. smstrCheckBox Then
FileRadPasWdForm. passwordTextBox = sign_strPassWord(FileRadPasWdForm. paswdBitTextBox. Value)
ElseIf Not FileRadPasWdForm. numCheckBox And FileRadPasWdForm. signCheckBox And Not FileRadPasWdForm. strCheckBox And FileRadPasWdForm. smstrCheckBox Then
FileRadPasWdForm. passwordTextBox = sign_smstrPassWord(FileRadPasWdForm. paswdBitTextBox. Value)
ElseIf Not FileRadPasWdForm. numCheckBox And Not FileRadPasWdForm. signCheckBox And FileRadPasWdForm. strCheckBox And FileRadPasWdForm. smstrCheckBox Then
FileRadPasWdForm. passwordTextBox = str_smstrPassWord(FileRadPasWdForm. paswdBitTextBox. Value)
ElseIf FileRadPasWdForm. numCheckBox And FileRadPasWdForm. signCheckBox And FileRadPasWdForm. strCheckBox And Not FileRadPasWdForm. smstrCheckBox Then
FileRadPasWdForm. passwordTextBox = num_sign_strPassWord(FileRadPasWdForm. paswdBitTextBox. Value)
ElseIf FileRadPasWdForm. numCheckBox And Not FileRadPasWdForm. signCheckBox And FileRadPasWdForm. strCheckBox And FileRadPasWdForm. smstrCheckBox Then
FileRadPasWdForm. passwordTextBox = num_str_smstrPassWord(FileRadPasWdForm. paswdBitTextBox. Value)
ElseIf FileRadPasWdForm. numCheckBox And FileRadPasWdForm. signCheckBox And Not FileRadPasWdForm. strCheckBox And FileRadPasWdForm. smstrCheckBox Then
FileRadPasWdForm. passwordTextBox = num_sign_smstrPassWord(FileRadPasWdForm. paswdBitTextBox. Value)
ElseIf Not FileRadPasWdForm. numCheckBox And FileRadPasWdForm. signCheckBox And FileRadPasWdForm. strCheckBox And FileRadPasWdForm. smstrCheckBox Then
FileRadPasWdForm. passwordTextBox = sign_str_smstrPassWord(FileRadPasWdForm. paswdBitTextBox. Value)
ElseIf FileRadPasWdForm. numCheckBox And FileRadPasWdForm. signCheckBox And FileRadPasWdForm. strCheckBox And FileRadPasWdForm. smstrCheckBox Then
FileRadPasWdForm. passwordTextBox = num_sign_str_smstrPassWord(FileRadPasWdForm. paswdBitTextBox. Value)
Else
'若在复选框内没有设定则给出警示
    MsgBox prompt: = "您必须在以上复选框内作出选择!", Title: = "警示"
End If
End Function
Function numPassWord(ByVal paswdBit As Integer) As String
    Dim I As Integer
'循环的次数根据用户设定密码长度 paswdBit 而定
    For I = 1 To paswdBit
        Randomize '初始化随机数生成器
        numPassWord = Int(10 * Rnd) & numPassWord '得到 0 - 9 的随机数并连接字符串
    Next I
  End Function
Function signPassWord(ByVal paswdBit As Integer) As String
    Dim I As Integer, Flag As Integer
'在标准打印 ASCII 码中,符号在 chr(33) - chr(47)、chr(58) - chr(64)、chr(91) - chr(96)、chr(123) - chr(126)中
'为了生成某个范围内的随机整数,可使用以下公式
'Int((随机数范围的上限 - 随机数范围的下限 + 1) * Rnd + 随机数范围的下限)
'对于返回 chr(33) - chr(47)中的随机符号则 Int((47 - 33 + 1) * Rnd + 33) = Int(15 * Rnd + 33)
    For I = 1 To paswdBit
        Randomize
        Flag = Int(4 * Rnd + 1) '通过返回随机的 1 - 4 来决定随机符号选自上述四个区域的那个区
        Select Case Flag
        Case 1
            Randomize
'返回在 chr(33) - chr(47)区域的随机符号并连接字符串
 signPassWord = Chr(Int(15 * Rnd + 33)) & signPassWord
```

```
        Case 2
            Randomize
  '返回在 chr(58) – chr(64)区域的随机符号并连接字符串
signPassWord = Chr(Int(7 * Rnd + 58)) & signPassWord
        Case 3
Randomize
  '返回在 chr(91) – chr(96)区域的随机符号并连接字符串
signPassWord = Chr(Int(6 * Rnd + 91)) & signPassWord
Case 4
Randomize
  '返回在 chr(123) – chr(126)区域的随机符号并连接字符串
signPassWord = Chr(Int(4 * Rnd + 123)) & signPass-
Word
End Select
Next I
  End Function
Function strPassWord(ByVal paswdBit As Integer) As String
    Dim I As Integer
       For I = 1 To paswdBit
        Randomize
'返回在 chr(65) – chr(90)区域的大写字母并连接字符串
strPassWord = Chr(Int(26 * Rnd + 65)) & strPassWord
Next I
End Function
Function smstrPassWord(ByVal paswdBit As Integer) As
String
    Dim I As Integer
   For I = 1 To paswdBit
        Randomize
  '返回在 chr(97) – chr(122)区域的小写字母并连接字符串
smstrPassWord = Chr(Int(26 * Rnd + 97)) & smstr-
PassWord
Next I
End Function
Function num_signPassWord(ByVal paswdBit As Integer) As
String
    Dim I As Integer, Flag As Integer
'数字与符号共同构成随机密码,数字与符号在每个字符位置上
出现的概率由随机数生成
    For I = 1 To paswdBit
      Randomize
      Flag = Int(2 * Rnd + 1)
      If Flag = 1 Then
num_signPassWord = numPassWord(1) &
num_signPassWord '调用数字随机密码生成子程序
      Else
num_signPassWord = signPassWord(1) &
num_signPassWord '调用符号随机密码生成子程序
      End If
    Next I
  End Function
Function num_strPassWord(ByVal paswdBit As Integer) As
String
    Dim I As Integer, Flag As Integer
      For I = 1 To paswdBit
      Randomize
      Flag = Int(2 * Rnd + 1)
      If Flag = 1 Then
num_strPassWord = numPassWord(1) & num_strPassWord
      Else
num_strPassWord = strPassWord(1) & num_strPassWord
      End If
    Next I
  End Function
Function num_smstrPassWord(ByVal paswdBit As Integer)
As String
    Dim I As Integer, Flag As Integer
   For I = 1 To paswdBit
      Randomize
      Flag = Int(2 * Rnd + 1)
      If Flag = 1 Then
            num_smstrPassWord = numPassWord(1) &
num_smstrPassWord
      Else
            num_smstrPassWord = smstrPassWord(1) &
num_smstrPassWord
      End If
    Next I
   End Function
Function sign_strPassWord(ByVal paswdBit As Integer) As
String
    Dim I As Integer, Flag As Integer
      For I = 1 To paswdBit
      Randomize
      Flag = Int(2 * Rnd + 1)
      If Flag = 1 Then
sign_strPassWord = signPassWord(1) & sign_strPassWord
      Else
sign_strPassWord = strPassWord(1) & sign_strPassWord
      End If
    Next I
   End Function
Function sign_smstrPassWord(ByVal paswdBit As Integer)
As String
    Dim I As Integer, Flag As Integer
  For I = 1 To paswdBit
      Randomize
      Flag = Int(2 * Rnd + 1)
      If Flag = 1 Then
            sign_smstrPassWord = signPassWord(1) &
sign_smstrPassWord
      Else
            sign_smstrPassWord = smstrPassWord(1) &
sign_smstrPassWord
      End If
    Next I
   End Function
Function str_smstrPassWord(ByVal paswdBit As Integer) As
```

```
String
Dim I As Integer, Flag As Integer
  For I = 1 To paswdBit
    Randomize
    Flag = Int(2 * Rnd + 1)
    If Flag = 1 Then
                str_smstrPassWord = strPassWord(1) & str_smstrPassWord
    Else
                str_smstrPassWord = smstrPassWord(1) & str_smstrPassWord
    End If
   Next I
  End Function
Function num_sign_strPassWord(ByVal paswdBit As Integer) As String
   Dim I As Integer, Flag As Integer
 For I = 1 To paswdBit
Randomize
    Flag = Int(3 * Rnd + 1)
 If Flag = 1 Then
          num_sign_strPassWord = numPassWord(1) & num_sign_strPassWord
    ElseIf Flag = 2 Then
          num_sign_strPassWord = signPassWord(1) & num_sign_strPassWord
    Else
          num_sign_strPassWord = strPassWord(1) & num_sign_strPassWord
    End If
   Next I
  End Function
Function num_str_smstrPassWord(ByVal paswdBit As Integer) As String
   Dim I As Integer, Flag As Integer
  For I = 1 To paswdBit
    Randomize
    Flag = Int(3 * Rnd + 1)
    If Flag = 1 Then
          num_str_smstrPassWord = numPassWord(1) & num_str_smstrPassWord
    ElseIf Flag = 2 Then
          num_str_smstrPassWord = strPassWord(1) & num_str_smstrPassWord
    Else
          num_str_smstrPassWord = smstrPassWord(1) & num_str_smstrPassWord
    End If
   Next I
  End Function
Function num_sign_smstrPassWord(ByVal paswdBit As Integer) As String
   Dim I As Integer, Flag As Integer
  For I = 1 To paswdBit
    Randomize
    Flag = Int(3 * Rnd + 1)
    If Flag = 1 Then
       num_sign_smstrPassWord = numPassWord(1) & num_sign_smstrPassWord
    ElseIf Flag = 2 Then
       num_sign_smstrPassWord = signPassWord(1) & num_sign_smstrPassWord
    Else
       num_sign_smstrPassWord = smstrPassWord(1) & num_sign_smstrPassWord
    End If
   Next I
  End Function
Function sign_str_smstrPassWord(ByVal paswdBit As Integer) As String
   Dim I As Integer, Flag As Integer
  For I = 1 To paswdBit
    Randomize
    Flag = Int(3 * Rnd + 1)
    If Flag = 1 Then
          sign_str_smstrPassWord = numPassWord(1) & sign_str_smstrPassWord
    ElseIf Flag = 2 Then
          sign_str_smstrPassWord = signPassWord(1) & sign_str_smstrPassWord
    Else
         sign_str_smstrPassWord = smstrPassWord(1) & sign_str_smstrPassWord
    End If
   Next I
  End Function
Function num_sign_str_smstrPassWord(ByVal paswdBit As Integer) As String
   Dim I As Integer, Flag As Integer
  For I = 1 To paswdBit
    Randomize
    Flag = Int(4 * Rnd + 1)
    If Flag = 1 Then
      num_sign_str_smstrPassWord = numPassWord(1) & num_sign_str_smstrPassWord
    ElseIf Flag = 2 Then
      num_sign_str_smstrPassWord = signPassWord(1) & num_sign_str_smstrPassWord
ElseIf Flag = 3 Then
       num_sign_str_smstrPassWord = strPassWord(1) & num_sign_str_smstrPassWord
    Else
      num_sign_str_smstrPassWord = smstrPassWord(1) & num_sign_str_smstrPassWord
    End If
   Next I
  End Function
Function CountCheckbox() As Integer
```

```
´CountCheckbox 返回检测用户选择的复选框个数
CountCheckbox = Abs(CInt(FileRadPasWdForm. numCheck-
Box. Value)) + Abs(CInt(FileRadPasWdForm. signCheckBox.
Value)) Abs(CInt(FileRadPasWdForm. strCheckBox. Value))
+ Abs(CInt(FileRadPasWdForm. smstrCheckBox. Value))
End Function
Function Text_SpinValue()
´通过用户选择的复选框个数来设定密码文本框和密码滚动条
的值
Select Case CountCheckbox
Case 1
FileRadPasWdForm. paswdBitTextBox. Value = 1
FileRadPasWdForm. paswdBitSpinButton. Value = 1
Case 2
FileRadPasWdForm. paswdBitTextBox. Value = 2
FileRadPasWdForm. paswdBitSpinButton. Value = 2
Case 3
FileRadPasWdForm. paswdBitTextBox. Value = 3
FileRadPasWdForm. paswdBitSpinButton. Value = 3
Case 4
FileRadPasWdForm. paswdBitTextBox. Value = 4
FileRadPasWdForm. paswdBitSpinButton. Value = 4
End Select
    End Function
```

四、 面向工作簿的开发

1. 面向工作簿实现的功能及解决的方法

通过上述三步的系统开发，测试多用户输入界面 . xls 程序已经具备用户登录时的选择、新建文件时的系统提供可选密码的保存，并将文件信息保存到个人工作表中去的功能。现在面向工作簿的问题是如何在用户使用该程序时出现登录对话框、新建文件时显示文件保存对话框。

为了方便用户打开已经保存文件簿，尤其是带有密码的工作簿，用户通常会有这样的经历——对于原本熟悉的密码，在几次错误输入后变得十分模糊，以至于一些重要的文件无法及时地打开。因此对于用户特定的工作表，还必须在工作表文件名显示区域通过双击文件名来打开文件， 在程序设计时不采用文件链接的方式，而采用双击文件名的模式，是因为链接的方式不提供直接的密码输入，需要用户自己键入）而在打开过程中不提示键入密码。因为有带密码显示的工作表是面向登录用户的，不用担心文件会被第二人非法打开。

当用户退出该程序时必须隐藏用户当前登录时的工作表。对于工作簿而言，打开时显示登录对话框可用 Workbook 对象的 Open 事件，在 Open 事件中调用登录对话框窗体。在工作表文件名显示区域通过双击文件名来打开文件，用 Workbook 对象的 SheetBeforeDoubleClick 事件。SheetBeforeDoubleClick 的语法如下

Private Sub object_SheetBeforeDoubleClick ByVal Sh As Object ByVal Target As Range ByVal Cancel As Boolean

其中：

object Application 对象或 Workbook 对象。

Sh 代表该工作表的 Worksheet 对象。

Target 当双击事件发生时最靠近鼠标指针的单元格。

Cancel 当事件产生时为 False。如果该事件过程将本参数设为 True，则该过程执行结束之后不执行默认的双击操作。

对于用户退出该程序时必须隐藏用户当前登录时的工作表，可用 Workbook 对象的 BeforeClose 事件。

对于实现新建文件时显示文件保存对话框的功能，必须在每一次新建文件时调用对话框。要用 Application 对象的 NewWorkbook 事件，当新建一个工作簿时产生此事件。NewWorkbook 事件的语法如下：

Private Sub object_NewWorkbook ByVal Wb As Workbook

object 在类模块中带有事件声明的 Application 类型对象。

Wb 新工作簿。

在对 Application 对象使用事件之前，必须新建一个类并定义一个带有事件声明的 Application 类型对象。例如，应用程序中新建了一个 EventClassModule 类模块，在该模块中包含下列代码。

Public WithEvents App As Application

进行完该对象的事件声明之后，该对象将出现在类模块的‘对象”下拉式列表框中，并可以为此新对象编写事件过程。

在‘对象”框中选取该新对象时，‘过程”下拉式列表框中就会自动显示该对象的有效事件）。

在运行过程之前，必须将类模块中定义的对象连接到 Application 对象。在模块中可以通过下列代码实现。

```
Dim X As New EventClassModule
Sub InitializeApp
  Set X. App = Application
End Sub
```

在运行 InitializeApp 过程之后，类模块中的 App 对象将指向 Microsoft Excel Application 对象，并且当事件发生时将执行相应的事件过程。

2. 面向工作簿的程序设计

2. 1 Open 事件的程序设计

```
    Private Sub Workbook_Open()
´打开工作簿时, 将产生本事件.
ThisWorkbook. Activate ´激活当前文件
ActiveWindow. WindowState =xlMinimized ´当前文件最小化
UserSelectForm. Show
´只有在登录用户选择用户名和键入密码时才调用 NewWork-
book 事件
If UserSelectForm. UserSelectComboBox. Value <> ”” And
UserPaswdForm. UserPaswdCTextBox. Value <> ”” Then
   Call InitializeApp
End If
End Sub
```

2. 2 SheetBeforeDoubleClick 事件的程序设计

Private Sub Workbook_SheetBeforeDoubleClick(ByVal Sh As

```
Object, ByVal Target As Range, Cancel As Boolean)
'当双击任何工作表时产生此事件
Dim OpenFileNamePath As String, OpenFileName As String
Dim FullOpenFileName As String, OpenFilePassword As String
'指定 xls 字符串在双击单元格中最先出现的位置
strCompValue = InStr(1, Range(Target.Address).Value, ".xls")
'在用户工作表中获取文件的文件名、路径、保存密码
OpenFileNamePath = Range(Target.Address).Offset(0, 3).Value
OpenFileName = Range(Target.Address).Value
FullOpenFileName = OpenFileNamePath & OpenFileName
OpenFilePassword = Range(Target.Address).Offset(0, 5).Value
'检测文件的密码标记并检测是否
'如果不是双击在文件名区域的话, Offset(0, 4).Value 不会等于 "√"
If Range(Target.Address).Offset(0, 4).Value = "√" Then
    If Dir(FullOpenFileName) = OpenFileName Then
Range(Target.Address).Font.ColorIndex = 7 '双击过的文件名呈粉红色
Workbooks.Open filename:=FullOpenFileName, Password:=OpenFilePassword '加密码的形式打开文件
Else
'如果不是双击在文件名区域的话给出警示
MsgBox prompt:="您选择的" & OpenFileName & "文件" & Chr(13) & Chr(13) & "不在" & OpenFileNamePath & "此路径内!" & Chr(13) & Chr(13) & "请重新确认后选择!", Title:="警示"
End If
Else
If strCompValue <> 0 Then
  If Dir(FullOpenFileName) = OpenFileName Then
 Range(Target.Address).Font.ColorIndex = 7
Workbooks.Open filename:=FullOpenFileName '不加密码的形式打开文件
Else
MsgBox prompt:="您选择的" & OpenFileName & "文件" & Chr(13) & Chr(13) & "不在" & OpenFileNamePath & "此路径内!" & Chr(13) & _
Chr(13) & "请重新确认后选择!", Title:="警示"
    End If
 Else
MsgBox prompt:="请选择有文件名的区域!", Title:="警示"
  End If
End If
Cancel = True
End Sub
```

2.3 BeforeClose 事件的程序设计

```
Private Sub Workbook_BeforeClose(Cancel As Boolean)
'关闭工作簿时先产生此事件
Dim currentcell As Range, nextcell As Range
ActiveWindow.WindowState = xlMinimized '最小化当前程序窗口
Application.ScreenUpdating = False '关闭屏幕更新功能
    Columns("b:b").Select
      Selection.Font.ColorIndex = 5 '恢复双击过文件名区域的颜色,呈蓝色
    Range("b1").Font.ColorIndex = 0
    Range("a1").Select
Worksheets(Sheet & "登录用户资料").Visible = xlSheetVeryHidden
Set currentcell = Worksheets(Sheet & "登录用户资料").Range("A2")
'使登录用户的个人工作表隐藏
  Do While Not IsEmpty(currentcell)
    username = currentcell.Value
 'xlSheetVeryHidden 的工作表格式不能通过菜单项的取消隐藏来显示工作表
Worksheets(Sheet & username).Visible = xlSheetVeryHidden
Set nextcell = currentcell.Offset(1, 0)
Set currentcell = nextcell
Loop
'保存当前程序
If Me.Saved = False Then Me.Save
Application.ScreenUpdating = True '打开屏幕更新功能
End Sub
```

2.4 NewWorkbook 事件的程序设计

插入一个模块 Module1，键入：

```
Dim X As New EventClassModule
Sub InitializeApp()
  Set X.App = Application
End Sub
```

添加一个类模块 EventClassModule，键入：

```
Option Explicit
Public WithEvents App As Application
Private Sub app_NewWorkbook(ByVal Wb As Excel.Workbook)
FileSaveForm.Show '调用文件保存对话框
End Sub
```

五、 结束语

程序至此已基本开发完成，为安全起见应对整个 VBAProject 工程加密，以保证程序的安全。

对于开发好的程序可以把文件 "测试多用户输入界面.xls" 对于 Office97 放在 Microsoft Office hOffice hXLStart 目录下面；对于 Office2000 放在 Windows hApplication Data hMicrosoft hExcel hXLStart 目录下面；这样当用户打开 Excel 时程序便会自动运行，使 Excel 总是工作在单一用户登录的状态下。

参考文献

1. VBA 5 开发人员指南 Matthew Harris 机械工业出版社
2. Office97、Office2000 帮助文档
3. Microsoft MSDN Library

收稿日期：2000 年 10 月 23 日）

在 C + + Builder 中如何使用游戏操纵杆

赵晓辉

在 Windows 环境下通过编程来操纵鼠标、键盘是一件再简单不过的事了，不过大家有没有想过要尝试一下另一种比较常见的输入工具——游戏操纵杆呢？在某些情况下，尤其是象编制一些小型的游戏软件的时候，加入对游戏操纵杆的支持可以给使用者提供更为友好的人机界面，极大地提高游戏软件的可玩性。

C + + Builder 中没有专门控制操纵杆函数　其实在常见的编程语言中基本上都没有），因此要增加对游戏操纵杆的支持，就要和 Windows 的 MCI API 函数打交道，这里我们首先介绍一些在读取操纵杆的属性、状态、位置和按钮信息时要用到的 API 函数、常量及数据结构。

相关常量：

```
#define MM_JOY1MOVE          0x3A0    /* 用以传递操纵杆当前状态的一些消息 */
#define MM_JOY2MOVE          0x3A1
#define MM_JOY1ZMOVE         0x3A2
#define MM_JOY2ZMOVE         0x3A3
#define MM_JOY1BUTTONDOWN    0x3B5
#define MM_JOY2BUTTONDOWN    0x3B6
#define MM_JOY1BUTTONUP      0x3B7
#define MM_JOY2BUTTONUP      0x3B8
#define JOY_BUTTON1          0x0001   /* 用以表明当前操纵杆的状态 */
#define JOY_BUTTON2          0x0002
#define JOY_BUTTON3          0x0004
#define JOY_BUTTON4          0x0008
#define JOY_BUTTON1CHG       0x0100
#define JOY_BUTTON2CHG       0x0200
#define JOY_BUTTON3CHG       0x0400
#define JOY_BUTTON4CHG       0x0800
/* 游戏操纵杆错误返回值 */
#define JOYERR_BASE          160
#define JOYERR_NOERROR       (0)              /* 正常 */
# define JOYERR_PARMS        (JOYERR_BASE +5)    /* 参数错误 */
#define JOYERR_NOCANDO       (JOYERR_BASE +6)    /* 无法正常工作 */
# define JOYERR_UNPLUGGED    (JOYERR_BASE +7)
/* 操纵杆未连接 */
/* 操纵杆标识号 */
#define JOYSTICKID1          0
#define JOYSTICKID2          1
```

相关函数：

```
WINMMAPI UINT WINAPI joyGetNumDevs(void);
```

获取设备标识号.

```
MMRESULT WINAPI joyGetDevCaps(UINT uJoyID, LPJOYCAPS pjc, UINT cbjc);
```

获取操纵杆属性信息，以结构体 JoyCaps 接收.

```
WINMMAPI MMRESULT WINAPI joyGetPos(UINT uJoyID, LPJOYINFO pji);
```

获取操纵杆位置和按钮状态，以结构体接收.

```
WINMMAPI MMRESULT WINAPI joyGetThreshold(UINT uJoyID, LPUINT puThreshold);
```

读取操纵杆移动阈值.

```
WINMMAPI MMRESULT WINAPI joyReleaseCapture(UINT uJoyID);
```

结束对操纵杆信息的接收.

```
WINMMAPI MMRESULT WINAPI joySetCapture(HWND hwnd, UINT uJoyID, UINT uPeriod,
    BOOL fChanged);
```

设置接收某一操纵杆的信息的窗口以及将何种频度接收.

```
WINMMAPI MMRESULT WINAPI joySetThreshold(UINT uJoyID, UINT uThreshold);
```

设置操纵杆移动阈值.

相关结构体：

```
typedef struct joyCaps{
    WORD wMid;                  /* 制造商标识 */
    WORD wPid;                  /* 生产编号 */
    char szPname[MAXPNAMELEN];  /* 产品名称 */
    UINT wXmin;                 /* X 轴最小值 */
    UINT wXmax;                 /* X 轴最大值 */
    UINT wYmin;                 /* Y 轴最小值 */
    UINT wYmax;                 /* Y 轴最大值 */
    UINT wZmin;                 /* Z 轴最小值 */
    UINT wZmax;                 /* Z 轴最大值 */
    UINT wNumButtons;           /* 按钮数 */
UINT wPeriodMin;     /* 最小调用间隔时间(单位 毫秒) */
UINT wPeriodMax;     /* 最大调用间隔时间(单位 毫秒) */
} JOYCAPS, * PJOYCAPS, NEAR * NPJOYCAPS, FAR * LPJOYCAPS;
typedef struct joyInfo{
    UINT wXpos;                 /* x 轴位置 */
    UINT wYpos;                 /* y 轴位置 */
    UINT wZpos;                 /* z 轴位置 */
    UINT wButtons;              /* 按钮状态 */
} JOYINFO, * PJOYINFO, NEAR * NPJOYINFO, FAR * LPJOYINFO;
```

以上这些定义存储在 mmsystem. h 文件中，所以程序要包含这个头文件。

程序需要首先检查游戏操纵杆的存在，这包括了检查驱动程序支持和确认操纵杆已与系统相连的两项工作。joyGetNumDevs 调用检查系统是否配置了游戏端口和驱动程序。如果返回值为零，表明不支持操纵杆功能。如果 joyGetNumDevs 返回值不为零，则说明系统支持游戏操纵杆功能。但 joyGetNumDevs 并不能确定操纵杆是否已被连接上了，通过调用可以完成这些工作，并检查是否有错误发生。

如果有游戏端口，joyGetNumDevs 返回值通常为 16。

一旦确认了操纵杆已连上，就可以接受发来的消息。joySetCapture 通知 Windows 操纵杆消息应发送到哪里，即发送的频率如何。

joySetCapture 中的第一个参数通知 Windows 谁将得到消息，第二个参数确定程序将从哪个操纵杆接收消息。第三个参数时表示希望以怎样的频度接受 JM_MOVE 消息　单位为毫秒），无论操纵杆是否移动，都将以这个频度接受 JM_MOVE 消息。joySetCapture 的四个参数允许程序当操纵杆移动一定的距离后才接受消息。该距离由 joySetThreshold 设置。

joySetCapture 被调用后，窗口将接受操纵杆事件。MM_JOYXMOVE（X = 操纵杆号）事件已 joySetCapture 定义的时间间隔发生。只有当操纵杆的按钮被按下时，MM_JOYXBUTTONUP 和 MM_JOYXBUTTONDOWN 事件才发生。操纵杆时间出发句柄，改变相应的标签状态信息。移动消息也同时通知程序在新的位置重画操纵杆标志。调用 joyReleaseCapture 通知 Windows 已结束操纵杆的调用。

在实际编制程序时，应首先在 Form1 的头文件 Form1. h 中加入对 mmsystem. h 的引用，再加入一些相关的消息映射即对 MM_JOYXMOVE、MM_JOYXBUTTONUP 和 MM_JOYXBUTTONDOWN 事件的响应函数说明。

```
#include <mmsystem. h>
//- - - - - - - - - - - - - - - - - - - - -
class Tform1: public TForm
{
_published:
. . .
. . .
private:
. . .
TPoint Position; //用于存储操纵杆的坐标位置.
. . .
public:
MESSAGE_HANDLER(MM_JOY1BUTTONDOWN, TMessage, JMButonUpdate)
MESSAGE_HANDLER(MM_JOY1BUTTONUP, TMessage, JMButonUpdate)
MESSAGE_HANDLER(MM_JOY1MOVE, TMessage, JMMove)
END_MESSAGE_MAP(TForm)
};
```

在 Form1 的 OnCreate 事件中加入以下代码用以检测操纵杆。

```
void _fastcall TForm1: : FormCreate(TObject * Sender)
{
 DriverCount = joyGetNumDevs();
 Connected = false;
 MMRESULT JoyResult;
 JOYINFO JoyInfo;
 //检查系统是否配置了游戏端口和驱动程序.
 if(DriverCount ! = 0)
 {
  //仍需调用 joyGetPos 进行检测，如果返回 JOYERR_NOERROR 则表示操纵杆连接正常.
  //测试第一个操纵杆.
  JoyResult = joyGetPos(JOYSTICKID1, & JoyInfo);
  if(JoyResult = = JOYERR_NOERROR )
   {
    Connected = true;
    JoystickID = JOYSTICKID1;
   }
  //如果发生 INVALIDPARAM 错误，则退出.
  else if(JoyResult = = MMSYSERR_INVALPARAM)
   Application - >MessageBox(″An error occured while calling joyGetPos″, ″Error″, MB_OK);
  // 如果第一个操纵杆为连接，则检查第二个操纵杆.
  else if((JoyResult =joyGetPos(JOYSTICKID2, & JoyInfo))
= = JOYERR_NOERROR)
  {
   Connected = true;
   JoystickID = JOYSTICKID2;
  }
 }
}
```

在确定操纵杆已正确连接之后就可以读取操纵杆的设备信息。

```
void TForm1: : ShowDeviceInfo(void)
{
    joyGetDevCaps(JoystickID, & JoyCaps, sizeof(JOYCAPS));
  Label1 - >Caption = ″Number of joysticks supported by driver = ″ + IntToStr(DriverCount);
  Label2 - >Caption = ″Current Joystick ID = ″ + IntToStr(intJoystickID);
Label3 - >Caption =″Manufacturer ID =″ + IntToStr(JoyCaps. wMid);
 Label4 - >Caption = ″Product ID = ″ + IntToStr(JoyCaps. wPid);
  Label5 - >Caption = ″Number of buttons = ″ + IntToStr(JoyCaps. wNumButtons);
  .
  .
  .
 // 设置当前窗口接收操纵杆信息.
 if(Connected)
joySetCapture(Handle, JoystickID, 2 * JoyCaps. wPeriodMin, FALSE);
 //计算操纵杆活动范围和屏幕范围的比率，在后面绘制操纵杆标志时会用到.
 XDivider = (JoyCaps. wXmax - JoyCaps. wXmin) / Width;
YDivider = (JoyCaps. wYmax - JoyCaps. wYmin) / Height;
}
```

读取操纵杆位置信息和按钮状态：

```
void TForm1: : ShowStatusInfo(void)
{
 if(Connected)
 {
```

```
    JOYINFO JoyInfo;
    TPoint Position;
    joyGetPos(JoystickID, & JoyInfo);
    Position.x = JoyInfo.wXpos;
    Position.y = JoyInfo.wYpos;
  //显示操纵杆的 X、Y 轴位置.
     Label6->Caption = "X Position = " + IntToStr(int
(JoyInfo.wXpos));
     Label7->Caption = "Y Position = " + IntToStr(int
(JoyInfo.wYpos));
  //判断某按钮是否被按下,这里只是指按钮初始的状态.)
    if(JoyInfo.wButtons & JOY_BUTTON1
      Label8->Caption = "Button 1 = Pressed";
    else
      Label8->Caption = "Button 1 = Not Pressed";
   }
}
```

下面可以编写用以响应当初在头文件中定义的事件 JMMove、JMButtonUpdate 的代码

```
void __fastcall TForm1::JMMove(TMessage & msg)
{
  /*当操纵杆位置发生变化时会自动调用本函数.在本函数中
经常是根据操纵杆当前的位置来绘制操纵杆在屏幕上显示的标
志 并擦去原来的标志.这里只是简单的改变 Image 的坐标位置
来表示操纵杆为的移动.*/
  Position.x = msg.LParamLo;
  Position.y = msg.LParamHi;
  //计算新的坐标.
  ScreenX = (Position.x - JoyCaps.wXmin)/XDivider - Im-
ageList1->Width/2;
  ScreenY = (Position.y - JoyCaps.wYmin)/YDivider - Im-
ageList1->Height/2;
  //显示新位置的 X、Y 值.
    Label6->Caption = "X Position = " + IntToStr(int
(Position.x));
    Label7->Caption = "Y Position = " + IntToStr(int
(Position.y));
  //移动 Image 的位置.
  Image1->Top = ScreenY;
  Image1->Left = ScreenX;
}
void __fastcall TForm1::JMButtonUpdate(TMessage & msg)
{
  //当程序接收到 JM_BUTTONDOWN 和 JM_BUTTONUP 消
息时,即某一按钮的状态发生改变时,都会调用本函数.
  if(msg.WParam & JOY_BUTTON1)  //判断按钮 1 是否被
按下
    Label8->Caption = "Button 1 = Pressed";
  else
    Label8->Caption = "Button 1 = Not Pressed";
}
```

最后在程序退出的时候要记得关闭对操纵杆的调用，即在 FormDestroy 事件中加入 joyReleaseCapture(JoystickID)。

```
void __fastcall TForm1::FormDestroy(TObject *Sender)
{
if(Connected)
  joyReleaseCapture(JoystickID);
}
```

以上就是使用操纵杆的常用方法，通过这些方法基本上就可以完成对操纵杆的一般操作，而且在了解了操纵杆的消息传递机制之后，我们还可以编写出一些用操纵杆模拟鼠标或是用鼠标/键盘来模拟操纵杆的程序以及通过编程让一般的手柄也象那种专用于格斗游戏的操纵杆一样具有可以记忆组合键的功能。不过要想更快速、更全面地操作操纵杆就需要用到 Windows 下高版本 DirectX 中的 DirectInput 技术了，鉴于篇幅以及使用比较繁琐的关系，这里就不予介绍了。

最后说明一下，本程序在 CBuilder4/PWin98 SE 环境下通过，在 WindowsNT 下用到的 API 函数将会与本程序中介绍的函数有所不同，详细区别请参阅 Windows API 函数手册。另外在程序测试中，我仅使用了最一般的接在声卡上的那种普通 4 键模拟手柄。针对其他的操纵杆及新型 USB 手柄/操纵杆，还希望有条件的朋友自己去测试一下。

（收稿日期：2000 年 10 月 16 日）

BMC 关注亚洲市场 从产品国际化做起

BMC 软件公司在为中国、日本、韩国和台湾市场进行的 PATROL® 国际化过程中取得关键性胜利

电子商务系统管理领域的领先供应商 BMC 软件有限公司日前宣布，在为其 PATROL®产品系列国际化的努力进程中取得了关键性胜利。根据这一宣布，PATROL 系列产品可在简体中文（大陆）、繁体中文（台湾）、韩语和日语环境下的流行操作平台和应用中运行。通过将 PATROL 国际化，BMC 软件公司正投入更多的关注，以便对亚洲太平洋市场上飞速增加的系统管理解决方案作出及时反应。

作为整个国际化进程的一部分，PATROL 新添了语言识别功能，以便支持简体/ 繁体中文、韩语和日语环境。通过这一功能，现在 PATROL 的代码独立于这些亚洲地区并可在本地化语言环境中正常运行。另外，本地化特定功能例如时间、日期等的格式也得到良好的支持。

现在一个亚洲国际化版的 PATROL 产品的子系列可交付使用，其中包括 PATROL for Windows 2000 Server，PATROL for Unix，PATROL for Oracle，PATROL for Lotus Dmino，PATROL for Internet Server Management，PATROL for Exchange Server 和 PATROL for Microsoft SQL Server。所有这些国际化产品在亚州均有质量保证。另外，术语的解释翻译遵循本地亚洲标准。

VxD 编程及其应用

乔敏灿 李克奇

摘 要 本文用实例介绍了使用 VToolsD 开发虚拟设备驱动程序的基本方法，展示了 VxD 编程的强大功能，本文介绍的实例具有一定的实用价值

关键词 VxD 程序，ring 0 层，ring 3 层，DR0 寄存器

目前，在微机操作系统中，Win9x 装机率占绝对优势，应用程序的开发也都建立在 Win9x 平台上。由于 Win9x 十分庞大，其构成也十分复杂，通常编程都是在应用层 ring 3 层 开发应用程序 Application ，但有时候我们不能不考虑如何深入到操作系统内部，编写与操作系统平起平坐的代码，如软件的加密与解密、反病毒、扩展操作系统的功能等。所兴的是，Win9x 提供了用户加载虚拟驱动程序 VxD 的功能 ，而 VxD 正是工作在 ring 0 层，具有与操作系统相同的优先级。虽然 Microsoft 提供了用于开发 VxD 的工具 MS DDK，但 MS DDK 十分庞大，内容也十分丰富，全面掌握 MS DDK 也很困难。这里介绍 Vireo Software 公司推出的 VToolsD for Win95 用其开发的 VxD 同样适用于 Win98 。VToolsD 提供了 C 语言接口，并支持 C + + 类库，使得 VxD 的开发十分方便。

VToolsD 最重要的部分是 Quick VxD，Quick VxD 提供了简单的可视化编程环境，可以快速地创建 VxD 程序框架，它包括 C/C + + 头文件、C/C + + 源程序和 C/C + + 工程文件，在这个框架中封装了 VxD 控制消息的接收、分配和处理。当使用 C + + 编程框架时，VToolsD 提供了许多类，其内容涉及开发 VxD 程序的方方面面，其中 VDevice 类是最重要的基类，每个 VxD 都须从 VDevice 派生一个自己的类，用于处理系统发来的控制消息。本文通过几则应用来展示 VxD 程序的魅力所在。

一、CIH 病毒免疫

由于 Win9x 的缺陷，运行于 ring3 的应用程序可以进入 ring0 层，取得对机器的绝对控制权。CIH 病毒正是利用这一点得以进入 ring0 层，实现了病毒的复制、发作等功能。CIH 病毒与 DOS 病毒有许多不同之处，病毒运行标志就是其中之一，CIH 病毒的运行标志设置在 DR0 寄存器中，被 CIH 病毒感染的程序在执行时首先要测试 DR0 的值，以决定是否将自身驻留于内存。根据这一原理，我们只要先于染毒文件运行之前将 DR0 设置成 CIH 病毒的运行标志，就可以达到免疫 CIH 病毒的目的，在运行于 ring3 层的应用程序中使用一般技术设置 DR0 的值是不能实现的，当然，可以使用与 CIH 病毒一样的手段 修改一个中断门的入口并激发该中断 进入 ring0 层，但编程较复杂，难以调试。使用 VxD 编程却十分简单，以下是我们给出的 nocih. vxd 的源程序：

```
// NOCIH. h – include file for VxD NOCIH
#include <vtoolscp. h>
#define DEVICE_CLASS        NocihDevice
#define NOCIH_DeviceID          UNDEFINED_DEVICE_ID
#define NOCIH_Init_Order    UNDEFINED_INIT_ORDER
#define NOCIH_Major         1
#define NOCIH_Minor         0
class NocihDevice : public VDevice
{
public:
  virtual BOOL OnDeviceInit(VMHANDLE hSysVM, PCHAR pszCmdTail);
};
// NOCIH. cpp – main module for VxD NOCIH
#define DEVICE_MAIN
#include "nocih. h"
Declare_Virtual_Device(NOCIH)
#undef DEVICE_MAIN
BOOL NocihDevice:: OnDeviceInit(VMHANDLE hSysVM, PCHAR pszCmdTail)
{
    _asm{
    push eax
    mov eax, 3243948036
    db 0fh, 23h, 0c0h   //mov dr0, eax
    pop eax
    }
  return TRUE;
}
```

在以上的源程序中，我们从 VDevice 类中派生了 NocihDevice 类，并重新定义了成员函数 OnDeviceInit，在该函数中我们只用了几条汇编语句就达到了目的。

二、网络自动登录

在计算机网络中，有时要求有的工作站能自动启动并登录，如用于数据采集的计算机；有时网络口令并无太多意义，如中小学电脑教室中联网的计算机。如何让计算机自动登录呢？显然在 ring3 层编程是无法实现的，因为登录网络前无法运行任何应用程序 Application 。借助于 VxD 编程很容易实现这一功能，

```
// AUTOLOG. h – include file for VxD AUTOLOG
・include <vtoolscp. h>
#define DEVICE_CLASS        AutologDevice
#define AUTOLOG_DeviceID        UNDEFINED_DEVICE_ID
#define AUTOLOG_Init_Order UNDEFINED_INIT_ORDER
```

```
#define AUTOLOG_Major          1
#define AUTOLOG_Minor          0
class AutologDevice : public VDevice
{
public:
virtual BOOL OnDeviceInit(VMHANDLE hSysVM, PCHAR pszCmdTail);
};
class Timer : public VGlobalTimeOut
{
public:
Timer(DWORD msec);
VOID handler(VMHANDLE hVM, THREADHANDLE th, PCLIENT_STRUCT pRegs, DWORD lag);
};
class title : public VAppyTimeEvent
{
public:
        title(char * pdw);
        virtual VOID handler(PVOID ref, DWORD flags);
};
// AUTOLOG.cpp - main module for VxD AUTOLOG
#define DEVICE_MAIN
#include "autolog.h"
Declare_Virtual_Device(AUTOLOG)
#undef DEVICE_MAIN
WORD count=0;
 char tit[128];
//口令按键的扫描码,此时为口令为 222(2 按下和释放的扫描码分别是 0x03 和 0x83), 0x1c, 0x9c 分别是回车键按下和释放的扫描码
char  keys[16]={0x03, 0x83, 0x03, 0x83, 0x03, 0x83, 0x1c, 0x9c};
BOOL AutologDevice::OnDeviceInit(VMHANDLE hSysVM, PCHAR pszCmdTail)
{
   VEvent::initEvents();
 Timer * ptimer = new Timer(600); // every 0.6 second
ptimer->Set();      // 计时生效
return TRUE;
}
title::title(char * pdw): VAppyTimeEvent((PVOID) pdw, CAAFL_TIMEOUT, 0) { }
VOID title::handler(PVOID ref, DWORD flags)
{
DWORD dd;
struct{
 WORD cb;
 DWORD sztext;
 DWORD hwnd;
 }gettext;
 if(IsAppyTimeAvailable())
{
PVOID la;
//获得当前窗口句柄
dd=CallDLL("USER", "GETACTIVEWINDOW", 0, NULL);
//分配 128 Byte 内存,用于保存窗口标题
gettext.sztext=LocalAlloc(LMEM_ZEROINIT, 128, la, NULL);
gettext.cb=64;
gettext.hwnd=dd;
//获得当前窗口标题,保存于 gettext.sztext
CallDLL( " USER", " GETWINDOWTEXT", sizeof(gettext), &gettext);
//将窗口标题复制到 ref,(gettext.sztext 和 la 指向同一地址)
strcpy((char *)ref, (char *)la);
LocalFree(gettext.sztext);
}
}
Timer::Timer(DWORD msec) : VGlobalTimeOut(msec) {}
VOID Timer::handler(VMHANDLE hVM, THREADHANDLE th, PCLIENT_STRUCT pRegs,
        DWORD lag)
{
static int i=0;
//创建 title 类实例,并调用 schedule()函数,返回时当前窗口标题在 tit 中
(new title(tit))->schedule();
// if(!strncmp(tit, "请输入", 6))//用于 Microsoft 网络用户
 if(!strncmp(tit, "欢迎使用", 8)) //用于 Microsoft 友好登录
 {
 if(i<8)                 //延时
  {i++;
  Set();
  }
 else
  {VKD_Force_Keys(keys, 8); //将口令写入键盘缓冲区
  }
 }
 else
 Set();
}
```

在这一程序中,我们定义了三个类 AutologDevice、Timer、title,分别派生于 VDevice、VGlobalTimeOut、VAppyTimeEvent,VGlobalTimeOut 是全局定时器类,VAppyTimeEvent 是异步事件类,在该类的事件处理函数中,可以调用应用程序级的函数 ring3 API ,

三、定时关机

目前的微机有很多方式来自动启动,如定时启动、MODEMO/网络唤醒等,但却不能自动关机。若想定时关闭计算机,也可以用 VxD 编程来实现。只需对上例中的 Timer handler 函数作如下修改:

```
VOID Timer::handler(VMHANDLE hVM, THREADHANDLE th, PCLIENT_STRUCT pRegs, DWORD lag)
{;
DWORD tim, dat;
tim=VTD_Get_Date_And_Time(&dat)
 tim=tim/60000;
 if(tim>=18*60+30)   //18:30 关机
  (new shutdown())->schedule();
 }
```

如何在 PowerBuilder 中实现 Windows 的资源管理器

程海鹰

摘 要 Windows 操作系统的资源管理器是一个有效地管理和浏览数据的工具。本文将结合一实例说明如何在 PowerBuilder 中实现 Windows 的资源管理器。

关键字 PowerBuilder，TreeView，ListView，拖放

Windows 操作系统的资源管理器是一个有效地管理和浏览数据的工具。在其左侧的列表中以层次结构列出了驱动器、目录以及子目录，用户通过单击层次结构中的加号（+）、减号（-）来展开或折叠目录，可使用户对数据的层次关系一目了然。在其右侧的列表中，用户可采用几种不同的方式对目录和文件进行浏览。此外，用户还可对目录和文件进行拖放操作，操作方法简单，符合人们的思维习惯。

笔者在用 PB 进行应用程序开发时，常常会遇到这种数据分层结构 若在 PB 中能实现资源管理器，将使用户以熟知的界面和操作方法对数据进行维护，起到事半功倍的效果。而在笔者所见的大多数实例中，都只是按照资源管理器的模式，实现了其中简单的一小部分功能，主要是实现了 TreeView 控件的数据浏览功能和拖放功能，而没有将 TreeView 控件和 ListView 控件进行有效地结合，实现复杂的数据浏览和拖放功能，这就大大地降低了程序的通用性和灵活性。

本文将结合一实例 如图 1 所示 说明如何在 PB 中实现 Windows 的资源管理器。读者在今后开发具有数据分层结构的应用时 只需根据实际情况对程序中的某些部分进行少量修改 即可完成此部分的开发任务。因此，程序的通用性强。

一、建立实验数据库和数据窗口对象

1. 创建一个本地 SQL Anywhere 数据库作为实验数据库

2. 创建表

本例在数据库中建立以下两个表。 （读者要根据实际情况创建表和其中的字段，本例旨在说明实现过程）

图 1 资源管理器演示程序

① employee 表 包括职工的具体信息

包括的字段为 职工姓名 emp_name char

职工号 emp_id char

职工性别 emp_sex char

职工所属部门号 dept_id char

② dept 表 表明部门的所属关系

包括的字段为 部门号 dept_id char

部门名称 dept_name char

上级部门号 last_dept char

3. 建立两个数据窗口对象

① d_emp_dept / /作为提取指定部门下的职工信息的数据源 显示风格设为 Grid 数据源为 QuickSelect ，选择全部字段

类 shutdown 派生于 VAppyTimeEvent，用来在 shutdown: : handler 内调用关机函数 EXITWINDOWS。Shutdown 类的定义和实现如下：

```
class shutdown :  public VAppyTimeEvent
{
public:
        shutdown();
        virtual VOID handler(PVOID ref,  DWORD flags);
};
shutdown: : shutdown(): VAppyTimeEvent(0){}
VOID shutdown: : handler(PVOID ref,  DWORD flags)
{
struct{
  UINT  ureserved;
  DWORD dwreserved;
  }shut;
 shut. ureserved = 0;
 shut. dwreserved = 0;
CallDLL(″USER″, ″EXITWINDOWS″, sizeof(shut), & shut);
}
```

VxD 编程的应用还有很多，如：利用 Hotkey 类可以做“一键上网”程序 笔者所做的免费程序名为< 一键通> 可到 http / / www. chinaprogrammer. com / softshow / tools / index. asp 下载 ，用系统定时器和 VToolsD 提供的 C 运行函数库可以做 Internet 上的各种 “泡点”程序 笔者已开发完成 。

收稿日期：2000 年 10 月 16 日）

可根据实际需求选择字段及其顺序 并设置提取条件为

WHERE employee . dept_id = dept_id

② d_dept_level //作为提取指定部门的下级部门信息的数据源

显示风格设为 Grid 数据源为 QuickSelect 选择全部字段，并设置提取条件为

WHERE dept . last_dept = last_dept。

三、创建定制可视用户对象 uo_tree_list_view

1. 在用户对象画笔工作区中加入 TreeView 和 ListView 控件

调整两控件位置如图 1 所示

左侧为 TreeView 控件：tv_1

右侧为 ListView 控件：lv_1

据实际需要设置两控件的属性。注意要选中 tv_1 属性表中 DragAuto 项，而不选中 Disable Drag Drop 项。在 lv_1 的属性表中选中 Drag Auto。选择合适的图片设置 tv_1 和 lv_1 的各图片项。

2. 声明实例变量

```
DataStore ids_file, ids_data
//分别用来提取指定部门下的部门项和职工项信息
Transaction it_trans//表明程序使用的事物对象
Long  il_empcolumn_count
//表明 ids_data 的数据窗口对象的字段数
Long  il_son_count//表明下一级部门数
char  ic_sourcetype//标志拖动源类型
TreeViewItem itvi_source, itvi_target
//分别表明 TreeView 的拖放源项和目标项
ListViewItem ilvi_source, ilvi_target
//分别表明 ListView 的拖放源项和目标项
long il_sourcehandle,  il_targethandle
//分别表明 TreeView 的拖放源项和目标项句柄
long il_sourceindex //表明 ListView 的拖放源项索引
long il_oldhandle//进行剪切操作时，当前 TreeView 项的句柄
```

3. 浏览功能的实现

1 定义以下用户对象函数

① uf_isnot_child //判断 TreeView 项是否有子项

入口参数 string parent_value

返回值：boolean

② uf_set_treeitem //生成下一级子项

入口参数 long parent_handle string parent_value

返回值 无

③ uf_initial //完成初始化功能

入口参数 string ls_file string ls_data

transaction lt_trans string ls_argu

返回值 无

函数体

```
. . . . . //初始化实例变量 ids_file, ids_data, it_trans
uf_set_treeitem(0, ls_argu) //生成 TreeView 的第一级项目
//为 ListView 在 Report 视图模式下增加各列
long li
il_empcolumn_count = long
(ids_data. object. datawindow. column. count)
string ls_column_name
for li = 1 to il_empcolumn_count
ls_column_name = ids_data. describe(´#´ + string(li) + ´. name´)
lv_1. AddColumn(ids_data. describe(ls_column_name + ´_t. text´),
& left!, long(ids_data. describe(ls_column_name + ´_t. width´)))
next
```

④ uf_find_treeposition

```
//找到 ListView 项对应于 TreeView 项的句柄
入口参数: string ls_value
返回值: long
函数体:
long ll_cur_handle, ll_handle
TreeViewItem ltvi_tree
//找到当前 TreeView 项的句柄
ll_cur_handle = tv_1. FindItem(CurrentTreeItem!, 0)
if tv_1. GetItem(ll_cur_handle, ltvi_tree) = -1 then return -1
tv_1. ExpandItem(ll_cur_handle)
ll_handle = tv_1. FindItem(ChildTreeItem!, ll_cur_handle)
if tv_1. GetItem(ll_handle, ltvi_tree) = -1 then return -1
if ltvi_tree. data <>ls_value then
  do
   //找到当前 TreeView 项的下一个子项的句柄
    ll_handle = tv_1. FindItem(NextTreeItem!, ll_handle)
    if tv_1. GetItem(ll_handle, ltvi_tree) = -1 then  return -1
  loop until ltvi_tree. data = ls_value
end if
//返回对应于 TreeView 项的句柄
return ll_handle
```

2 tv_1 的 itemexpanded 事件

删除已有的子项 重新生成下一级子项

3 tv_1 的 selectionchanged 事件脚本

```
TreeViewItem ltvi
if tv_1. GetItem(newhandle, ltvi) = -1 then return
ListViewItem llvi_dept, llvi_emp
long li, lj, ll_son_employeecount
il_son_count = ids_file. retrieve(ltvi. data) //下一级部门数
//该部门拥有的职工数
ll_son_employeecount = ids_data. retrieve(ltvi. data)
lv_1. DeleteItems()
if il_son_count >0 then
. . . . . //设置下一级部门的 ListView 项
end if
//设置该部门下职工的 ListView 项
if ll_son_employeecount>0 then
for li = 1 to ll_son_employeecount
  // 根据实际情况设置 llvi_emp 的属性
  llvi_emp. label = ids_data. GetItemString(li, 1)
  //在职工号前加上字符´*´来区分职工项和部门项
  llvi_emp. data = ´*´ + ids_data. GetItemString(li, 2)
  llvi_emp. PictureIndex = 2
  if il_empcolumn_count>1 then
  //设置 ListView 在 Report 视图下要显示的职工的详细信息
    for lj = 2 to il_empcolumn_count
```

```
 llvi_emp. label = llvi_emp. label + ´ ~t´ + ids_data. GetItemStrin
(li, lj)
    next
  end if
  lv_1. AddItem(llvi_emp)
next
end if
```

说明：

区分部门项和职工项的方法：由于在绝大多数情况下各种编号不会出现字符 ‘＊’，因此在设置 ListView **职工项的** data **属性时，在其职工号前加上字符 ‘＊’ 以指明该项为职工，而不是部门。若在实际应用中不能满足要求，可选择其它区分方法。**

4 lv_1 **的** doubleclicked **事件脚本**

通过调用函数 uf_find_treeposition **展开相应的** TreeView **项。**

4. 拖放功能的实现

1 **定义以下用户对象函数**

① uf_isnot_right //**判断拖放的部门项是否正确**

入口参数 long ll_source long ll_target

返回值 boolean

函数体

```
//判断源项与目标项是否相同
if ll_source = ll_target then
   messagebox(´错误´, ´源项与目标项相同, 不能移动´, Stop-
Sign!)
   return true
end if
//判断源项是否已经为目标项的下一级部门
ll_source = tv_1. FindItem(ParentTreeItem!, ll_source)
if ll_source = il_targethandle then
      messagebox(´错误´, ´源部门已是目标部门的下一级子
部门, 不能移动´, StopSign!)
      return true
end if
//判断源项是否为目标项的上级
long li, ll_min
if itvi_target. level>itvi_source. level then
ll_min = itvi_target. level  -  itvi_source. level
for li = 1 to ll_min
      ll_target = tv_1. FindItem(ParentTreeItem!, ll_target)
next
if ll_target = il_sourcehandle then
 messagebox(´错误´, ´目标部门是源部门的下属部门, 不能移
动´, StopSign!)
 return true
end if
end if
return false
```

② uf_update_empdept //**更新数据库，修改职工的所属部门**

入口参数 string emp_id string dept_id

返回值 无

③ uf_update_lastdept //**更新数据库 修改部门所属的上级部门**

入口参数 string dept_id string last_dept

返回值 无

④ uf_dragdrop //**处理部门项的拖放事件**

入口参数：无

返回值：integer

函数体：

```
long ll_newhandle, ll_cur_handle
//判断拖放是否正确, 不正确返回
if uf_isnot_right(il_sourcehandle, il_targethandle)then return -1
if messagebox(´注意´, ´你确定将´ + itvi_source. label + ´转
到´ + itvi_target. label + ´下吗?´, Question!, YesNo!) = 2 then
return -1
ll_newhandle = tv_1. InsertItemLast(il_targethandle, itvi_source)
tv_1. DeleteItem(il_sourcehandle)
tv_1. CollapseItem(ll_newhandle)
//更新数据库
uf_update_lastdept(itvi_source. data, itvi_target. data)
if ic_sourcetype = ´l´ then//拖放源项为 ListView 项
   ll_cur_handle = tv_1. FindItem(CurrentTreeItem!, 0)
   if  ll_cur_handle = il_oldhandle then)
       lv_1. DeleteItem(il_sourceindex)
     end if
else
   tv_1. ExpandItem(il_targethandle
   tv_1. SelectItem(ll_newhandle)
end if
return 1
```

2 lv_1 tv_1 **的** begindrag **事件**

主要用于标志拖放源类型 确定拖放源项。

3 lv_1 tv_1 **的** dragdrop **事件**

主要功能可通过调用函数 uf_dragdrop **来实现**

四、建立测试窗口 w_test

1. **粘贴用户对象** uo_tree_list_view **即** uo_1

2. **在窗口** w_test **的** open **事件中写上如下脚本**

```
//完成初始化功能
uo_1. uf_initial(´d_dept_level´, ´d_emp_dept´, SQLCA, ´0´)
```

说明：除了以上实现的主要功能外，还可创建一快捷菜单实现查看、剪切、粘贴、删除等多种功能。由于篇幅有限，文中只给出部分程序代码，读者不难在此基础上对其它部分进行扩充。若读者对此程序兴趣，可与笔者联系 E_mail xieycheng@263. net **，愿将全部程序代码给出。**

以上程序已在中文 Windows 98 **和** PowerBuilder 6. 0 **的环境下测试通过**

参考文献

[1]**张长富、李匀等 .** PowerBuilder 6. 0 **开发人员指南 . 北京希望电脑公司**

收稿日期：2000 **年** 10 **月** 30 **日)**

在 C + + Builder 中绘制中文数据报表

刘素芳 汤子东

提 要 本文介绍了从数据文件中读取数据，创建 DBF 数据库，利用 C + + Builder 中的 Quick Report 组件生成具有表格线的中文数据报表。程序中可以定义报表的页面属性，自动计算表格宽度。

一、前言

Quick Report 组件是挪威的 QuSoft AS 公司专门为 C + + Builder 设计的用于制作报表的一组控件，它具有很强的访问数据库的能力，利用它可以方便地制作形式多样的数据报表。

由于 Quick Report 组件是外国人研制的，所以报表一般没有表格线。这不符合中国人的习惯。若要将数据库中字段的所有记录都生成到绘制好表格线的报表中就要自己编写程序了。

在“参考文献 1”中的第 329 页介绍了如何制作中文表格。本文就是根据其原理经过改编而成。

本程序可以设置报表的页面属性 包括纸张大小、纸张方向、页面边距、字体大小等），通过读取数据文件创建数据库，把数据库的内容以中文数据报表的形式显示出来，可以打印并自动分页，可以根据纸张大小、方向和数据库的字段数自动计算表格线的位置。

二、 建立工程文件

在 C + + Builder 中建立新的工程文件 CHREPORT. BPR，窗体文件命名为 RPT_UNIT. CPP，存放到当前目录下的 SOURCE 目录中，其上增加三个 TButton 按钮，分别为 SetButton、DrawButton、ExitButton，其 Caption 属性分别定义为设置报表、绘制报表、退出系统，建立对应的 OnClick 事件。其他的定义详见程序。

在工程中增加 Tform ，命名为 ReportForm ，窗体文件命名为 MYREPORT. CPP，上面放置一个 TQuickRep 控件 QuickRep1 和 TTable 控件 Table1，QuickRep1 的 DataSet 属性设为 Table1。在 QuickRep1 上建立 BeforePrint 事件 QuickRep1BeforePrint。

在 QuickRep1 上放置两个 TQRBand 控件，QRBand1 的 BandType 属性改为 rbPageHeader，QRBand2 的 BandType 属性改为 rbDetail，QRBand2 的 Frame 属性中的 DrawLeft、DrawTop、DrawRight 改为 true。

在 QRBand1 上放置 TQRSysData 控件 QRSysData1，其 Data 属性改为 qrsReportTitle，AlignToBand 属性改为 true，Alignment 属性改为 taCenter。这样控件 QRSysData1 就显示在 QRBand1 控件的中间。

在工程中增加 TForm ，命名为 SetRepAttDialogForm ，窗体文件命名为 SET_DLG. CPP，上面有一些控件用于设置报表的页面属性，读者可以加入自己需要的属性。窗体中的具体内容详见程序。

这样工程文件就建成了，下面讲述程序的具体实现。

三、 程序的具体实现

在 RPT_UNIT. CPP 文件中，注意创建数据库函数中的这个语句：

```
rTable - >FlushBuffers
```

主要用来清空缓冲区中的内容，以免造成追加数据时出错。

在 DrawButtonClick 函数中，首先创建数据库，然后创建 ReportForm 窗口，用

```
ReportForm - >SetPagePara(RepPaperSize, RepPaperDirection, RepFontSize, RepPageTop, RepPageBottom, RepPageLeft, RepPageRight);
```

语句设置报表的页面属性，用

```
ReportForm - >SetParaInfo(CurrentDirectory, PathFile, TableName);
```

语句设置报表的数据库信息。然后设置报表标题。

```
ReportForm - >QuickRep1 - >ReportTitle = TitleStr;
```

最后预览打印

```
ReportForm - >QuickRep1 - >Preview();
```

在 MYREPORT. CPP 中 QuickRep1 的 BeforePrint 事件是绘制表格线的关键：

```
//- - - - - - - - - - - - - - - - - - - - - - - - - - - -
// 根据数据库中的内容显示每个字段的内容并绘制表格线
void __fastcall TReportForm: : QuickRep1BeforePrint (TCustomQuickRep * Sender, bool & PrintReport)
{
  SetReportPage();
  TQRShape * Vert_Band[FIELD_COUNT];
  TQRDBText * Data_Text[FIELD_COUNT];
  TQRShape * Bottom_Band;
  Table1 - >Active = true;
  int fcount = Table1 - >FieldCount; // 数据库的字段数
  if(fcount>FIELD_COUNT)
  {
    MessageBeep(MB_OK);
ShowMessage(″FIELD_COUNT = ″ + IntToStr(FIELD_COUNT)
```

```
+"定义的太小,请与编写程序的人员联系");
    return;
  }
  float PageWidthMm = QuickRep1->Page->Width; // 页面的宽度(毫米制)
  int ReportWidthPixel = QuickRep1->Width;    // 页面的宽度(象素制)
   float PixelMm = (float) ReportWidthPixel/PageWidthMm;
// 象素和毫米的比值
  int LieWidth; // 页面每列的宽度
       LieWidth = floor((QuickRep1->Page->Width - QuickRep1->Page->LeftMargin -
QuickRep1->Page->RightMargin) * PixelMm/(float)fcount);
  Font = QuickRep1->Font;
  int fontwidth = Canvas->TextWidth("好")/2;
  int fontheight = Canvas->TextHeight("好");
  int i = Table1->Fields->Fields[0]->Size * fontwidth;
  if(i>LieWidth)
  {
    MessageBeep(MB_OK);
    ShowMessage("每列的宽度小于需要的宽度,某些数据可能显示不出来,请扩大纸张的宽度");
  }
  int H_dd = 4;
  int left1;
  DetailBand1->Height = fontheight + H_dd;
  left1 = -fontwidth/2;
  // 显示数据库中每个字段的内容
  for(i = 0; i < fcount; i++)
  {
    Data_Text[i] = new TQRDBText(this)
    Data_Text[i]->Parent = DetailBand1;
    Data_Text[i]->Top = H_dd/2;
    Data_Text[i]->Width = LieWidth;
    Data_Text[i]->Height = DetailBand1->Height;
    // 注意: 下面这句话一定要有
    Data_Text[i]->AutoSize = false;
    Data_Text[i]->Left = left1 + LieWidth * i;
    Data_Text[i]->DataSet = Table1;
      Data_Text[i]->DataField = Table1->FieldDefs->Items[i]->Name;
    Data_Text[i]->Alignment = taRightJustify;
  }
  left1 = LieWidth/2;
  // 绘制垂直线条
  // 因为 TQRShape 的 Left 以 DetailBand1->Left 为左边界,形状为 qrsVertLine,
  // 例如,若宽度是 10,则在 5 处绘制一个垂直线
  // 最后一列的垂直线不用绘制, QRBand2 控件中已经绘制了
  for(i = 0; i < fcount - 1; i++)
  {
    Vert_Band[i] = new TQRShape(this);
    Vert_Band[i]->Parent = DetailBand1;
    Vert_Band[i]->Shape = qrsVertLine
    Vert_Band[i]->Brush->Style = bsClear;
    Vert_Band[i]->Width = LieWidth;
    Vert_Band[i]->Height = DetailBand1->Height;
    Vert_Band[i]->Left = left1 + LieWidth * i;
  }
  // 绘制水平线
  Bottom_Band = new TQRShape(this);
  Bottom_Band->Parent = DetailBand1;
  Bottom_Band->Shape = qrsHorLine;
  Bottom_Band->Brush->Style = bsClear;
  Bottom_Band->Width = DetailBand1->Width;
      Bottom_Band->Top = floor((float) (DetailBand1->Height)/2.0 + 0.5);
  Bottom_Band->Left = 0;
  Bottom_Band->Height = DetailBand1->Height;
}
```

注意函数中加粗倾斜的语句，这些是关键语句。

函数中首先调用设置报表页面属性的函数，再根据页面宽度、数据库字段数计算出每个字段在报表上的宽度 LieWidth，接着显示数据库中每个字段的内容：

```
    Data_Text[i]->DataSet = Table1;
      Data_Text[i]->DataField = Table1->FieldDefs->Items[i]->Name;
    Data_Text[i]->Alignment = taRightJustify;
```

字段内容是右对齐，具体如何对齐自己可以自由设定。读者可以在设置报表页面属性时加上如何对齐的选项。

然后是绘制垂直线和水平线。 程序中有注释，也可以参阅“参考文献 1”）

四、 结语

本文提供了一种制作中文数据报表的方法，读者可以根据自己的需要修改本程序，以适应自己的要求。比如显示打印时间、页号等。本程序中不论字段的宽度是多少，绘制的表格线都是等宽的，读者可以根据数据库字段的宽度求出每个字符在报表中的宽度，再求出不同宽度的字段在报表中的宽度，这样绘制的表格线就不是等宽的了。另外，还可以在 SET_DLG. CPP 中加入自己需要的一些属性，比如字段内容的对齐方式等。

本程序已被作者应用在工作中，并取得了良好效果。

希望本文能对大家有所帮助。

本程序在 C++ Builder 5.0 企业版中调试通过 （C++ Builder 安装在 D:\BORLAND\CBUILDER5 目录中），操作系统环境为中文 Windows 2000 专业版 在 Windows 95/98 中也能应用）。

参考文献

1. 李智慧、秦成 . C++ Builder 4.0 从入门到精通 . 清华大学出版社，1999 年 8 月

收稿日期：2000 年 10 月 17 日）

Windows API 拦截技术及实现

马 翔 余 矾

摘 要 本文主要介绍并分析了 Windows 下几种常用的 API 拦截技术，并以一具体实例详细说明了其主要技术和实现过程。

API 拦截，也有人称之为钩子，截取或挂接，是一个在编程中可能会遇到的问题。当我们需要对 API 的功能加以修改或替换时，就要会用到该技术。其最典型的应用就是鼠标取词。要实现 Windows 下 API 的拦截比较常用的技术有以下三种：

一种最简单通用的方法是替换系统的 DLL。即自己做一个与系统 DLL 同名的 DLL，而将原来的系统 DLL 改名。在自己编写的 DLL 中导出原系统 DLL 中所有的导出函数。这样就可以用自己的代码替换需要拦截的 API；而对于不要求处理的 API，可以简单地把参数传递给原系统 DLL 相应的 API 来处理 之后返回其结果。实际上自己编写的 DLL 充当了调用者与被调用 DLL 之间的桥梁，使其能在 API 真正执行前取得控制权。笔者曾用此法替换了 Windows95 下的动态库 wsock32. dll，实现了对其中 send 和 recv API 函数的拦截。此法虽简单，但是它不能动态加截或取消拦截功能，而且它需要对系统 DLL 文件进行替换，不便于系统进行升级。特别是对于已加载到内存中的系统 DLL，只能在系统重启后才可以进行替换。

另一种方法是修改模块的输入节。由于动态链接库在每次装载时可能要重定位，从而使其基址发生变化，导致 DLL 中的导出函数的地址也发生变化。而在 PE 格式的可执行文件的头部包含了模块的输入节信息，也就是输入函数的入口地址表。它在 DLL 加载后由系统填入，这使得模块可以不受 DLL 基址变化的影响而正确运行。模块的输入节包含一组该模块运行时需要的 DLL，它还包含该模块从每个 DLL 中输入的符号列表。当模块调用一个输入函数时，线程实际上要从模块的输入节中捕获需要的输入函数地址，然后转移到该地址。模块对 DLL 中函数的调用就通过该输入节以间接寻址方式来实现的。如果修改了该输入节，使欲拦截的 API 的入口地址指向自己的代码，就可以获取控制权，实现拦截。《Windows95 系统编程奥秘》一书中就用该方法实现了一个 API SPY 程序。很明显，此方法仅能拦截特定的模块对 API 的调用，也就是我们修改了输入节的那些模块。如果不作特殊处理，例如枚举所有进程并修改输入节，并不能拦截到所有模块对其的调用。而且对于通过 GetProcAddress 来获取的 API 调用，若不加处理也不能直接进行拦截。

还有一种方法是直接修改 API 代码。这一方法被普遍使用。很多的应用程序都使用了它 如金山词霸、RichWin、中文之星等。其实现原理是先找到要拦截的 API 函数在内存中的地址，并保存该函数的头几个字节的指令，用一个 JMP 或 INT 汇编指令改写该函数的头几个字节，该指令会直接或通过中断方式转移到你的替换函数上执行。这样，当一个线程调用该函数时，指令实际上将转移到你的替换函数。当然如果在替换函数中需要完成对原 API 函数调用 那就要先恢复原函数入口的那些字节 等调用完成后 再修改回来，以便下次拦截。这种方法由于是直接动态修改代码，所以不存在以上两种方法的局限。但也会出现新问题，在抢占式多线程环境中，如果当代码被改写时，另一个线程又试图调用该函数，此时可能会发生严重错误。只有通过一定的方法才能避免该情况发生。

以上是常见的几种 API 拦截方法。第一种较为简单，在此不再赘述。第二种在一些书中都有详尽介绍及实现代码。如《Windows 核心编程》《Windows95 系统编程奥秘》等，也不多作分析。下面，我们来重点讨论第三种方法。并以一简单实例，说明用中断调用和内存共享来挂接系统的 API 调用的具体实现过程。

用代码修改法进行 API 拦截首先遇到的问题是如何进行代码修改。我们知道在 Windows 中代码段是只读的，通常并不能自由的改写。在 Windows 3. 1 中，对于 NE 格式的 EXE 程序，系统是可以移动代码段的。这就是利用了其未公开的函数 AllocCStoDSAlias，将一个代码段选择变为数据段选择，然后就可以写了。这仅能在 Win31 中起作用。在 Win95 / 98 中虽然也可以通过编写 16 位代码完成对系统核心 16 位 API 函数的调用的拦截，但是对于 32 位的 API 调用的拦截，它就无能为力了。虽然有时也可以通过 Thunk 技术，VXD 技术或 16 位 API 提供的 LoadLibraryEx32W，GetProcAddress32W CallProc32W 以及 32 位动态库 wow32. dll 中提供的 WOWCallback16，WOWGetVDMPointer 等 API 来来实现 16 位与 32 位代码间的调用，但比较繁琐。在 Win95 / 98 中要想能直接修改代码，一种可行的办法就是强迫系统进入 RING 0，这样就可以对代码进行编辑了。关于该技术在本刊的第六期中有文章提过两种方法 即通过修改中断向量表 IDT 法和安装调用门 LDT 法。在本实例中，采用的是修改中断向量 IDT 的方法进入 RING0，从而完成代码的修改。但我们是把其入口代码改为 INT 指令，而不是 JMP。其原因就在于用 JMP 来实现一个远跳转时，至少要用到 5 个字节，也就是说我们要保存原函数入

口的 5 个字节。而当一个 API 与其相邻的下一个 API 之间的地址偏移少于 5 字节时，我们就没有办法对其拦截了。因为这样会影响到与其相邻的 API 的调用，导致非法错误。但是用 INT 指令一般只要 2 个字节就行了，对于 INT3 等指令只要 1 个字节，而对一个 API 入口而言，它与下一个 API 入口偏移一般都会大于 2 个字节。并且我们可以将所有要拦截的 API 均共享一个中断向量，并不会增加其实现的复杂性。因而这种方法有更强的适应性。

其次要解决的问题是代码注入的问题。也就是如何把自己的代码插入到其它进程中去。我们知道每个进程都拥有它自己的系统资源。对于内存而言，每个进程都具有它自己私有的 4GB 虚拟地址空间。 它包括程序的 EXE 映像，所加载的非系统的 DLL、程序的全局数据、内存映射文件等等。在 Windows 95 / 98 系统中采用页式内存管理，把进程私有页面和共享页面放在不同地址空间。只有地址空间最底部的 2GB 0 - - 0X7FFFFFFF 才是真正私有的 而顶部的 2GB 对于所有的进程都是相同的 被所有的进程共享 这使得进程之间的内存共享简单快速。它最顶部的 1GB 包括 Windows 系统内核，虚拟设备驱动程序 VxDs 和文件系统代码等；另外 1GB 存放 Windows 系统的 DLL、内存映射文件等。为了能使自己编写的替代 API 代码能被正确运行就要求该代码要与调用进程位于同一进程空间内，而这恰恰违背了独立性原则。也正因为进程间的独立性，给 API 拦截带来一定麻烦。这个问题常见的解决方案有 DLL 注入和内存共享等方法。DLL 注入是把替换的 API 函数代码写入一个 DLL，当原进程调用要拦截的 API 时，通过一次进程上下文转换，把自己编写的 DLL 映射到调用进程空间中去，从而使其在同一内存空间。我们可以用 Windows 提供的钩子函数来实现这一功能，安装一个全局钩子，如 WH_GETMESSAGE，用 SetWindowsHookEx 函数可把钩子与所有线程相联。这样系统就可以为我们完成 DLL 的映射过程了。内存共享是利用各进程的顶部 2GB - 3GB 的共享地址空间来存放替换的 API 函数代码，因而该段代码存在于所有进程空间中，对任意进程都是可见的。利用 CreateFileMapping 来创建一块共享内存区，并用 MapViewOfFile 映射到内存即可。此外，还有些别的方法，如用远程线程插入代码等。本例中使用了内存共享的方法。

最后所涉及的问题是如何完成拦截接口。在此我们结合本例来讲述该接口的实现原理。首先，当然是要更改中断向量表，使该用于 API 拦截的中断挂接到自己的代码上去。接着，就要更改欲拦截 API 的入口指令，用 INT 来替换。这样当发生 API 调用时，系统就会产生中断进入 RING0。这时，我们通过获取栈中的返回地址并修改该地址，可以知道是调用哪一个 API 时产生的中断，并能依此在中断完成后转到不同入口进行处理。本例中为了简单起见，只拦截了一个 API 调用，省去了分支跳转处理过程。进入处理代码后，要恢复原 API 入口指令，而且为了能在跳转到原 API 执行后仍返回到自己的处理代码上来，需要更改栈中的返回地址并保存原返回地址。我们可以移动当前调用栈得到空间，把原返回地址保存在该空间中去。这样在完成对原 API 调用后，仍会返回到处理代码上。此时再次将 API 的入口指令改为 INT 就可以了。最后是从栈中得到原调用进程的返回地址并返回到调用进程。这就实现了拦截接口。由于此过程要过多地涉及到栈操作，为了方便在此用了一些汇编，可能不便于阅读。但本例旨在说明其原理以及实现的可行性，有兴趣的读者可以参考代码中的注释，加以改进。在此不再对实例作更详尽的分析说明。

该程序运行后，首先会显示一个经过拦截处理的对话框，其标题已被改变。在确认后，又出现一对话框，此时可以切换到其它进程来测试其对 MessageBox 的拦截。比如简单地在 Windows 开始菜单的运行对话框中输入一个不存在的命令，我们可以看到系统出现的错误对话框的标题也被更改了。这就说明了该程序可以拦截其它进程对 API 的调用。该程序在 VC6 中调试运行通过。

总之 API 拦截是一项涉及到操作系统内核的技术，它需要对系统有深入的了解。以上这些也只是笔者对 API 拦截技术的一点粗浅认识，希望能与广大读者交流。

收稿日期：2000 年 10 月 13 日）

新世纪第一拍

——英文顶级域名‘黄河’、‘长江’被拍卖

新世纪到来的 2001 年 1 月 1 日 0 时，在中华文化上具有无比珍贵价值的英文国际顶级域名 ‘黄河’、‘长江’由中国域名专业门户网站易域网联合著名拍卖专业网站嘉德在线共同拍卖。

此次易域网拍卖的 ‘中国系列’域名共有 24 个英文顶级域名，这些英文顶级域名大部分带有 “China”或 “cn”，如中国媒体 （chinamdeia. com）、中国商店 （chinashop . com）、中国邮政 （chinapost. com）、中国书籍 （china - book. com）、中国婚庆 （china marry. com）、中国出口 （exportschina. com）、中国古玩 （chinacurio. com）、中国航运 （chinaship. com）、中国专利 （chinamonopoly. com）、中国太太 （Mrschina. com）等。其中最引人注目的就包括黄河 （yelloweriver. com）、长江 yangtseriver. com 这两个国际顶级域名。

新世纪是以电子商务为重要特征的新经济的世纪，而域名作为电子商务时代最重要的互联网上品牌标识将具有极高的政治、经济和文化价值。在新世纪到来之时启动 ‘黄河’、 ‘长江’这两个在中华文化上具有无比珍贵价值的英文国际顶级域名和其他具有极重要经济价值的域名拍卖，堪称 ‘世纪第一拍’。

Microsoft Agent 的 MFC 使用技术

丁有和

Microsoft Agent 是一个能在应用程序用户界面中显示指定的人物造型的 ActiveX 控件，其动画人物最显著的特点就是造型美观，它一般有两部分：一是精灵本身，它可以通过‘听觉”或者由程序控制摆出各种不同的姿势，如听、看、读、写等；另一部分是被称作 Balloon 的语言提示部分，它能在精灵上方显示出 Agent 所说的话。Microsoft 已经为最新的 Agent 2.0 提供了 4 种标准人物造型，它们是 Genie、Merlin、Robby、Peedy，下载地址是：http //msdn.microsoft.com/workshop/imedia/agent/agentdl.asp。

Agent 的另一大特色是语音合成和语音识别功能。只要在系统中装入了 L&H TruVoice Text-To-Speech TTS Engine 和 Microsoft Speech Recognition Engine 4.0，就能在应用程序中用相当完美的语速朗读文字。

由于在 Visual C++[1]中使用 Microsoft Agent 控件常因为一些细微的错误而导致许多用户另择他法，因此本文从创建、放映、讲话等几个方面说明用 MFC 程序控制的方法和技巧。

一、创建

在 Visual C++中，既可用对话框模板来创建，也可在程序中进行动态创建。

首先以基于对话框程序为例说明 Agent 控件利用对话框模板创建的方法，其步骤如下：

1 创建一个基于对话框的项目 AgentDemo20。

2 选择 Project | Add To Project | Components and Controls 菜单，弹出 Components and Controls 对话框。

3 在此对话框中选择 Registered ActiveX Controls，将 Microsoft Agent Control 2.0 组件相关的‘类”插入。

4 打开 IDD_IMGDEMO_DIALOG 对话框资源模板，并添加 Microsoft Agent Control 2.0，保留其缺省的 ID 号。

5 在 ClassWizard 的 Member Variables 选项卡中，为刚才添加的 Agent 控件添加相关联的成员变量 m_aAgent，并为对话框添加 WM_INITDIALOG 的消息处理，增加下列代码：

```
BOOL CAgentDemo20Dlg::OnInitDialog()
{
    CDialog::OnInitDialog();
    ...
    // TODO: Add extra initialization here
    COleVariant vPath(_T("C:\\WINDOWS\\Msagent\\CHARS\\Merlin.acs"));
    m_aAgent.GetCharacters().Load("Merlin", vPath);
    m_charMerlin = m_aAgent.GetCharacters().Character("Merlin");
    return TRUE;  // return TRUE unless you set the focus to a control
}
```

6 为 CAgentDemo20Dlg 类添加 CAgentCtlCharacterEx 类型的成员变量 m_charMerlin。

7 在 AgentDemo20Dlg.h 的源文件前面添加相关 Agent 控件的包含头文件：

```
//{{AFX_INCLUDES()
#include "agentctlex.h"
#include "AgentCtlCharacters.h"
#include "AgentCtlCharacterEx.h"
#include "AgentCtlRequest.h"
#include "AgentCtlBalloonEx.h"
//}}AFX_INCLUDES
```

若用户在窗口类中进行动态创建，则可依照下面的过程以 SDI 应用程序为例 ：

1 创建一个 SDI 应用程序项目 AgentSdi。

2 选择 Project | Add To Project | Components and Controls 菜单，弹出 Components and Controls 对话框。

3 在此对话框中选择 Registered ActiveX Controls，将 Microsoft Agent Control 2.0 组件相关的‘类”插入。

4 在 AgentSdiView.h 源文件的前面添加相关 Agent 控件的包含头文件：

```
#include "agentctlex.h"
#include "AgentCtlCharacters.h"
#include "AgentCtlCharacterEx.h"
#include "AgentCtlRequest.h"
#include "AgentCtlBalloonEx.h"
```

5 在 CAgentSdiView 类中添加所要的数据成员：

```
public:
  CAgentCtlEx * m_pAgent;
  CAgentCtlCharacterEx m_charMerlin;
```

6 用 ClassWizard 为 CAgentSdiView 类添加 OnInitialUpdate 消息，并增加下列代码：

```
void CAgentSdiView::OnInitialUpdate()
{
  CView::OnInitialUpdate();
      // TODO: Add your specialized code here and/or call the base class
  m_pAgent = new CAgentCtlEx;
  ASSERT_VALID(m_pAgent);
  m_pAgent->Create(_T("Agent Demo"), WS_CHILD | WS_VISIBLE, CRect(0, 0, 20, 20), this, 1100);
COleVariant vPath(_T("C:\\WINDOWS\\Msagent\\CHARS\\Merlin.acs"));
m_pAgent->GetCharacters().Load("Merlin", vPath);
m_charMerlin = m_pAgent->GetCharacters().Character("Merlin");
}
```

7 在构造和析构函数中添加相关代码：

```
CAgentSdiView: : CAgentSdiView()
{
// TODO: add construction code here
m_pAgent = NULL;
}
CAgentSdiView: : ~CAgentSdiView()
{
if (m_pAgent)
    delete m_pAgent;
}
```

这样，整个框架代码就搭好了，下面就可对 Agent 进行编程控制。

二、 编程控制

添加到应用程序项目的 Agent 控件的类共有十几个，分别用于角色、文字提示、命令、属性以及语音等方面的控制。一般情况下，常常需要对角色进行控制，如显示、播放、讲话、隐藏等。需要说明的是，在 Agent 提供的函数中，有许多参数是 VARIANT 数据类型。该类型广泛用于 ActiveX 中，使用时要注意设置 VARIANT 变量的具体类型，例如：

```
VARIANT fast;
fast. boolVal = TRUE;
fast. vt = VT_BOOL; // 设置具体的数据类型为 BOOL
m_charMerlin. Show(fast);
```

但对于 VARIANT 的字符串型变量来说，一般不能直接对其赋值，这时可以用 MFC 的 COleVariant 进行自动转换，如前面的：

```
COleVariant vPath(_T( ″ C: \\ WINDOWS\\ Msagent\\ CHARS\\Merlin. acs″));
```

1 显示和隐藏

CAgentCtlCharacterEx 类的 Show 和 Hide 分别用来显示和隐藏指定的动画人物。需要说明的是，Show 函数并不能将动画人物的图像显示出来，而必须在其后调用 Play “Show” 才能如愿；同样，在调用 Hide 之前，也最好调用 Play “Hide” 来强调隐藏的动画效果，如下面的代码：

```
VARIANT fast;
fast. boolVal = TRUE;
fast. vt = VT_BOOL;
m_charMerlin. Show(fast)
m_charMerlin. Play(″Show″);
. . .
m_charMerlin. Play(″Hide″);
m_charMerlin. Hide(fast);
```

2 移动

默认情况下，动画人物是在屏幕的左上角开始显示的，若觉得这个位置不理想，可使用 CAgentCtlCharacterEx 的 SetTop 和 SetLeft 来改变它，或调用 MoveTo 将动画人物移到指定的位置，如下面的代码：

```
VARIANT speed;
speed. lVal = 50;
speed. vt = VT_I4;
m_charMerlin. MoveTo(100, 100, speed);
```

其中 speed 指定移动的速度，单位为毫秒。

3 播放动画

动画人物在设计后都有自己特定的动作，如听、看、读、写、发怒、困惑等数十种动画形态，可以调用 Play 来播放动画动作。值得一提的是，Agent 2. 0 中的 4 种标准人物造型的动作名称基本一样，例如 Merlin 中的 “Lookup”，其他三个也有；更有甚者，Office 2000 所使用的十几种动画人物也有相似的动作名称。

需要说明的是，动画形态中有一些是循环动画，必须用调用 StopAll 或 Stop 来终止它才能进行下一个动画的播放。例如：

```
m_charMerlin. Play (″Hearing_1″);
COleVariant vEmpty;
m_charMerlin. StopAll(vEmpty);
m_charMerlin. Play (″Suggest″);
```

4 讲话

让动画人物根据文本内容进行讲话是 Agent 一个非常富于情趣的特性，用户可通过 CAgentCtlCharacterEx Speak 来实现。如下面的代码：

```
COleVariant url(_T(″″));
COleVariant str(_T(″Good Morning!″));
m_charMerlin. Speak(str, url);
```

需要说明的是，上述 Speak 方法将自动加载相应的 TTS 引擎，若用户需要指明一个特定的 TTS 引擎，可通过 CAgentCtlCharacterEx SetTTSModeID 来加载。

除了上述功能外，Agent 还具有语音识别、文字提示、命令、属性等方面的功能，这里不再论述。

三、 实例

根据前面的论述，可在应用程序中加入 Agent 功能，下图就是一个很好的例子 具体的代码略 。

本文仅讨论 MFC 对 Agent 控件进行编程控制的一般方法和技巧，帮助用户减少错误的产生，起到一个抛砖引玉的作用。

参 考 文 献

1. 丁有和编著. Visual C + + 程序员基础教程. 青岛出版社 1999. 10

收稿日期：2000 年 10 月 17 日）

实现文件对话框的图像预览功能

徐志波

很多图像处理方面的软件都支持文件对话框的图片预览功能 极大地方便了我们选择所需的图片 . 那么如何在自己的软件中加入这一功能呢

我们知道 MFC 中的 CFileDialog 类封装了文件对话框的功能，它的成员 m_ofn 是 OPENFILENAME 类型的结构，我们要想定制文件对话框，只需从 CFileDialog 中派生一个新类，然后如下设置 m_ofn 的内部成员

```
m_ofn. Flags | = ( OFN_EXPLORER | OFN_ENABLETEM-
PLATE);
m_ofn. lpTemplateName = MAKEINTRESOURCE(IDD_FILE-
OPENPREVIEW);
```

IDD_FILEOPENPREVIEW 是我们创建的对话框资源的 ID 包含了我们想要在定制的文件对话框中加入的控件和它们的定位信息。设置了 OFN_EXPLORER 属性后 系统将创建一个标准文件对话框的子对话框 其中不仅包含了原先的标准控件还加入了我们在 IDD_FILEOPENPREVIEW 中包含的控件 并且它们两者的位置是由 IDD_FILEOPENPREVIEW 中的定位信息所决定的。这个定位信息就是一个 ID 为 stc32 的静态文本框 . stc32 是系统预定义的标号 用来代表标准文件对话框中的控件 我们以它为参照物来自由安排自己想添加的控件的位置。

设置了对话框资源后 还需重载如下函数来响应消息。

重载 OnFileNameChange 来处理用户从文件列表中选择了一个新的文件或目录的情形 重载 OnFolderChange 来处理用户打开一个新目录的情形 重载 OnTypeChange 来处理用户选择了一种新文件类型的情形。

实际演示如下。

1. 首先创建一个单文档程序 BmpPreviewDlg. 接下来我们要先创建一个位图类 CDib。由于本程序的演示性质 实现的这个位图类功能比较简单 仅有图像的读取和显示功能。方法 BOOL Load LPCTSTR lpszPathName 载入参数表示的文件 void Draw HDC CRect * pDstRect CRect * pSrcRect = NULL 显示位图 BOOL IsEmpty 返回位图是否已装载。具体的代码附于文后。

2. 然后我们创建一个能显示位图的图像框类 CBmpStatic。选择 “Insert hNew class ” 加入新类 CBmpStatic 基类为 CStatic。在头文件 “BmpStatic. h ” 中加入如下包含语句

```
#include ″Dib. h″
```

加入如下成员

```
private:
   CDib m_Dib;
// Operations
public:
   BOOL ShowImage(LPCTSTR lpszPathName);
   BOOL HideImage();
   void DoPaint(BOOL bImage);
```

函数 ShowImage 显示位图 lpszPathName:

```
BOOL CBmpStatic: : ShowImage(LPCTSTR lpszPath-
Name)
{
BOOL bSuccess = m_Dib. Load(lpszPathName);
 DoPaint(bSuccess);
   return TRUE;
}
```

函数 HideImage 清除图像的显示

```
BOOL CBmpStatic: : HideImage()
{
  DoPaint(FALSE);
  return TRUE;
}
```

辅助函数 DoPaint 当参数 bImage 为真时，显示位图 为假时用白色填充。

```
void CBmpStatic: : DoPaint(BOOL bImage)
{
CRect ClientRect;
GetClientRect(& ClientRect);
CClientDC dc(this);
  //若 bImage 为真, 绘制位图;
  if(bImage)
  {
m_Dib. Draw(dc. GetSafeHdc(), & ClientRect);
  }
  else //否则, 填充白色;
  {
CBrush * pBrush;
pBrush = CBrush: : FromHandle((HBRUSH) GetStockObject
(WHITE_BRUSH));
    dc. FillRect(& ClientRect, pBrush);
          }
}
```

重载 WM_PAINT 消息的响应函数

```
void CBmpStatic: : OnPaint()
{
  CPaintDC dc(this); // device context for painting
  DoPaint(! m_Dib. IsEmpty());
  }
```

3. 打开资源编辑器，插入一个对话框资源 IDD_FILEOPENPREVIEW。在它的属性对话框中单击 “Styles ” 标签，设置对话框的类型 style 为 Child，边界 Border 为 None，选择 Clip siblings 属性。单击 “More Styles ” 标签，选择 “Visible ” “3D - look ” “Control ” 属性。接下来删除 “OK ” “CANCEL ” 按钮 添加一个静态文本框 static text 令其 ID 为 stc32。在其右边添加图像框 Picture IDC_IMAGE，

类型选择为 Bitmap，选择 “Sunken” “Center image”属性。结果如图 1 所示

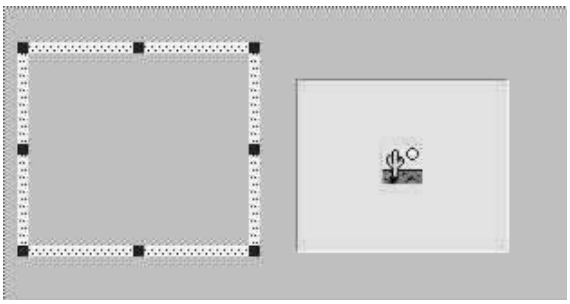


图 1

4. 选择 “Insert hNew Class”菜单加入新类 CPreviewFileDlg，其基类为 CFileDialog。在头文件 “PreviewFileDlg. h”中加入包含语句

```
#include "BmpStatic. h"
```

加入成员变量

```
protected:
  CBmpStatic m_BmpStaticCtrl;
```

在 CPreviewFileDlg 的构造函数中加入如下语句:

```
m_ofn. Flags | = ( OFN_EXPLORER | OFN_ENABLETEM-
PLATE);
m_ofn. lpTemplateName = MAKEINTRESOURCE(IDD_FILE-
OPENPREVIEW);
```

添加 WM_INITDIALOG 的响应函数

```
BOOL CPreviewFileDlg: : OnInitDialog()
{
CFileDialog: : OnInitDialog();
m_BmpStaticCtrl. SubclassDlgItem(IDC_IMAGE, this);
return TRUE;
}
```

重载虚拟函数 OnFileNameChange

```
void CPreviewFileDlg: : OnFileNameChange()
{;
CFileDialog: : OnFileNameChange()
m_BmpStaticCtrl. ShowImage(GetPathName());
}
```

重载虚拟函数 OnFolderChange

```
void CPreviewFileDlg: : OnFolderChange()
{
CFileDialog: : OnFolderChange();
m_BmpStaticCtrl. HideImage();
}
```

5. 在类 CBmpPreviewDlgApp 的实现文件 BmpPreviewDlg. cpp 中加入包含语句

```
#include "PreviewFileDlg. h"
```

添加 ID_FILE_OPEN 的响应函数 OnFileOpen

```
void CBmpPreviewDlgApp: : OnFileOpen()
{
    CString szFilter = "BMP Files( * . bmp) | * . bmp| |";
    CString szDefExt = ". bmp";
    CPreviewFileDlg dlg(TRUE, szDefExt, NULL,
OFN_HIDEREADONLY, szFilter, NULL);
  if(dlg. DoModal() = =IDOK)
  {
  OpenDocumentFile(dlg. GetPathName());
  }
}
```

6. 在类 CBmpPreviewDlgDoc 的头文件 BmpPreviewDlgDoc. h 中添加包含语句

```
#include "Dib. h"
```

在类 CBmpPreviewDlgDoc 中添加公开成员

```
CDib m_Dib;
```

重载虚拟函数 OnOpenDocument

```
BOOL CBmpPreviewDlgDoc: : OnOpenDocument(LPCTSTR
lpszPathName)
{
  BOOL bSuccess;
  BeginWaitCursor();
  bSuccess = m_Dib. Load(lpszPathName);
  EndWaitCursor();
  if(bSuccess)
  {
    SetPathName(lpszPathName);
    SetModifiedFlag(FALSE);  //开始时置未修改标记;
   }
  return bSuccess;
}
```

7. 如下处理 CBmpPreviewDlgView 的 OnDraw 函数

```
void CBmpPreviewDlgView: : OnDraw(CDC * pDC)
{
  CBmpPreviewDlgDoc * pDoc = GetDocument();
  ASSERT_VALID(pDoc);
  CDib * pCurDib = & pDoc - >m_Dib;
  if(! pCurDib - >IsEmpty())
  {
  CRect rect;
  GetClientRect(& rect);
  pCurDib - >Draw(pDC - >GetSafeHdc(), & rect);
  }
}
```

编译、运行程序。选择 “文件 h打开”菜单 是不是出现了如图 2 所示的图像预览文件对话框

图 2

本文仅仅演示了如何定制文件对话框来实现图像预览功能，其实定制文件对话框的用处是很广泛的，但不管你定制的对话框形状如何，采用的方法都是一样的。大家可以根据自己的需要任意定制文件对话框。

本文演示程序在 VC6.0 Win98 下运行通过。

附 CDib 类的源代码

```
//Dib.h
#ifndef _DIB_H
#define _DIB_H
#define WIDTHBYTES(bits)    (((bits) + 31) / 32 * 4)
class CDib : public CObject
{
DECLARE_DYNAMIC(CDib)
// Constructors
public:
    CDib();
// Attributes
protected:
   WORD m_wWidth;
    WORD m_wHeight;
    WORD m_wColorNum;
    BOOL m_bEmpty;
    LPBYTE m_pBits;
    LPBITMAPINFO m_pBi;
public:
    CPalette * m_pPal;
    // Operations
public:
void Draw(HDC, CRect * pDstRect, CRect * pSrcRect =
NULL);
    BOOL Load(LPCTSTR lpszPathName);
    BOOL IsEmpty()    { return m_bEmpty; }
      // Implementation
public:
    virtual ~CDib();
protected:
    BOOL Read(CFile& file);
    WORD NumColors(LPBITMAPINFOHEADER lpBih);
    BOOL CreatePalette();
    void Destroy();
public:
#ifdef _DEBUG
virtual void Dump(CDumpContext& dc) const;
#endif
};
#endif
//Dib.cpp
#include "stdafx.h"
#include "Dib.h"
IMPLEMENT_DYNAMIC(CDib, CObject)
CDib::CDib()
{
    m_wWidth=0;
    m_wHeight=0;
    m_wColorNum=0;
    m_bEmpty=TRUE;
    m_pBi = NULL;
    m_pBits = NULL;
    m_pPal = NULL;
}
CDib::~CDib()
{
  Destroy();
}
void CDib::Destroy()
{
  m_wWidth=0;
  m_wHeight=0;
  m_wColorNum=0;
  m_bEmpty=TRUE;
  if (m_pBits)
  {
  GlobalFree((HANDLE)m_pBits);
  m_pBits = NULL;
  }
  if (m_pBi)
  {
  GlobalFree((HANDLE)m_pBi);
  m_pBi = NULL;
  }
  if (m_pPal)
  {
  delete m_pPal;
  m_pPal = NULL;
  }
}
void CDib::Draw(HDC hDC, CRect * pDstRect, CRect *
pSrcRect)
{
  HPALETTE hPal = NULL;
  HPALETTE hOldPal = NULL;
  if (m_pPal != NULL)
  {
  hPal = (HPALETTE) m_pPal->m_hObject;
  hOldPal = SelectPalette(hDC, hPal, TRUE);
  }
  //若源矩形为空,则取整个位图;
  if(pSrcRect==NULL)
  {
    CRect srcRect;
    srcRect.left=srcRect.top=0;
    srcRect.right=m_wWidth-1;
    srcRect.bottom=m_wHeight-1;
    pSrcRect=&srcRect;
  }
  ::SetStretchBltMode(hDC, COLORONCOLOR);
  if ((pDstRect->Width() == pSrcRect->Width()) &&
    (pDstRect->Height() == pSrcRect->Height()))
  SetDIBitsToDevice(hDC, pDstRect->left, pDstRect->top,
pDstRect->Width(), pDstRect->Height(),
pSrcRect->left, pSrcRect->top, 0, m_wHeight, m_pBits,
m_pBi, DIB_RGB_COLORS);
  else
StretchDIBits(hDC, pDstRect->left, pDstRect->top,
pDstRect->Width(), pDstRect->Height(),
pSrcRect->left, pSrcRect->top, pSrcRect->Width(), pSr-
cRect->Height(),
m_pBits, m_pBi, DIB_RGB_COLORS, SRCCOPY);
if (hOldPal != NULL)
  {
```

```
SelectPalette(hDC, hOldPal, TRUE);
      }
}
BOOL CDib: : CreatePalette( )
{
// 颜色数不为零时,创建调色板;
  if (m_wColorNum ! = 0)
  {
HANDLE hLogPal = GlobalAlloc(GHND, sizeof(LOGPALETTE)
 + sizeof(PALETTEENTRY) * m_wColorNum);
      if (hLogPal = = NULL)
      return FALSE;
      LPLOGPALETTE lpPal = (LPLOGPALETTE) GlobalLock
(hLogPal);
      lpPal - >palVersion = 0x300;
      lpPal - >palNumEntries = m_wColorNum;
      for (int i = 0; i < (int)m_wColorNum; i + + )
      {
lpPal - >palPalEntry[i]. peRed = m_pBi - >bmiColors[i]. rgbRed;
      lpPal - >palPalEntry[i] . peGreen = m_pBi - >bmiColors[i]
. rgbGreen;
lpPal - >palPalEntry[i]. peBlue = m_pBi - >bmiColors[i]. rgbBlue;
lpPal - >palPalEntry[i]. peFlags = 0;
      }
      if (m_pPal)
      {
      delete m_pPal;
      }
      m_pPal = new CPalette;
      BOOL bResult = m_pPal - >CreatePalette(lpPal);
      GlobalUnlock((HGLOBAL) hLogPal);
      GlobalFree((HGLOBAL) hLogPal);
      return bResult;
}
 return TRUE;
}
//计算颜色数的函数;
WORD CDib: : NumColors(LPBITMAPINFOHEADER lpBih)
{
   WORD wColorNum;
      WORD wBitCount;
      if(lpBih - >biClrUsed! =0)
      wColorNum = (WORD)lpBih - >biClrUsed;
      else
      {
  wBitCount = (WORD)lpBih - >biBitCount;
   if(wBitCount < =8)
      wColorNum = 1 < <wBitCount;
        else
           wColorNum = 0;
       }
      return wColorNum;
}
BOOL CDib: : Load(LPCTSTR lpszPathName)
{
      CFile file;
      if(!file. Open(lpszPathName, CFile: : modeRead))
      return FALSE;
   Destroy( );
   Read(file);
   CreatePalette( );
   m_bEmpty = FALSE;
   file. Close( );
   return TRUE;
}
BOOL CDib: : Read(CFile& file)
{
   BITMAPFILEHEADER bfh;
   BITMAPINFOHEADER bih;
   if (file. Read((LPSTR)& bfh, sizeof(bfh)) ! = sizeof(bfh))
   return FALSE;
   if (bfh. bfType ! = 0x4d42)
   return FALSE;
   if(file. Read((LPSTR)& bih, sizeof(bih))! = sizeof(bih))
   return FALSE;
   m_wWidth = (WORD)bih. biWidth;
   m_wHeight = (WORD)bih. biHeight;
   m_wColorNum = NumColors(& bih);
   DWORD dwSizeBi = sizeof(BITMAPINFOHEADER) +
m_wColorNum * sizeof(RGBQUAD);
   DWORD dwSizeImage = WIDTHBYTES(m_wWidth *
bih. biBitCount) * m_wHeight;
   m_pBi = (LPBITMAPINFO)GlobalAlloc(GPTR, dwSizeBi);
   if (m_pBi = = NULL)
         return FALSE;
    file. Seek(sizeof(BITMAPFILEHEADER), CFile: : begin);
   if (file. Read(m_pBi, dwSizeBi) ! = dwSizeBi)
   {
   GlobalFree((HANDLE)m_pBi);
   m_pBi = NULL;
   return FALSE;
   }
   m_pBits = (LPBYTE)GlobalAlloc(GPTR, dwSizeImage);
   if (m_pBits = = NULL)
   {
   GlobalFree((HANDLE)m_pBi);
   m_pBi = NULL;
   return FALSE;
   }
if(file. ReadHuge(m_pBits, dwSizeImage) ! = dwSizeImage)
   {
   GlobalFree((HANDLE)m_pBi);
   m_pBi = NULL;
   GlobalFree((HANDLE)m_pBits);
   m_pBits = NULL;
   return FALSE;
   }
   return TRUE;
}
#ifdef _DEBUG
void CDib: : Dump(CDumpContext& dc) const
{
CObject: : Dump(dc);
}
#endif
```

收稿日期 2000 **年** 9 **月** 4 **日**

MFC 的 RTTI 技术设计与实现

钱俊彦

摘 要 程序员在编程时,可能在程序执行过程中想知道某个对象属于哪一个类?本文将首先介绍实现 RTTI 的类别型录网和 CRuntimeClass 结构,然后设计与实现 RTTI 技术,最后处理检测某个对象是否属于某个类的 IsKindof 函数。

关键词 结构,RTTI,宏,MFC

一、 概述

程序员在编程时,有可能在程序执行过程中想知道某个对象属于哪一个类?这就是 VC++ 中称的动态类型识别(Runtime Type Information RTTI)的能力。RTTI 就是希望我的类库具备 IsKindOf 的能力,运行时来检测某个对象是否"属于某个类"。

当我们用 VC++ 的 AppWizard 产生一个应用程序框架的时候,会产生一组令人费解的宏——DECLARE_DYNAMIC 和 IMPLEMENT_DYNAMIC。有的书中仅仅说明这是用来动态识别的,那动态识别是什么呢?又是如何实现的呢?

二、 类别型录网与 CRuntimeClass 结构

怎么设计 RTTI 呢?当你持有一种产品,想知道型号,不查型录行吗?就象你想知道祖先是谁,不查家谱行吗?要让程序做到 RTTI 的能力,必须在类构造起来的时候,记录类的必要信息来建立型录。为了在程序中便于比较,型录一般最好用链表来实现。

型录的链表元素在 MFC 用一个结构 CRuntimeClass 来描述(图 1):

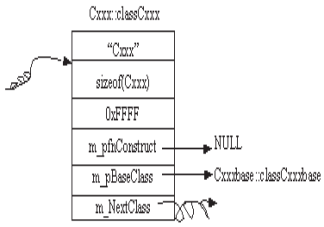


图 1

```
struct CRuntimeClass
{
   LPCSTR m_lpszClassName; //存储类名
   int m_nObjectSize; //存储类的内存长度
   UINT m_wSchema; //序列号
  void (PASCAL * m_pfnConstruct)( void *p); // 实现 RTTI 是不使用
   CRunTimeClass * m_pBaseClass; //指向基类的的指针
  static CRunTimeClass * pFirstClass; //头指针
CRunTimeClass * m_pNextClass; //指向下一个表元素的指针
}
```

希望每一个类都拥有 CRuntimeClass 成员变量,并且希望对这个 CRuntimeClass 结构有一定的命名规则,如在类名前面加上一个 class.

三、DECLARE_DYNAMIC 和 IMPLEMENT_DYNAMIC 宏

MFC (Microsoft Foundation Class)为了隐藏复杂性和简化源代码,使用了一组宏 (DECLARE_DYNAMIC 和 IMPLEMENT_DYNAMIC)来实现 RTTI。

怎样才能把 CRuntimeClass 对象放入到类中,并定义一个可以捕捉到该对象地址的函数?使用 DECLARE_DYNAMIC 宏就可以完成。MFC 中定义 DECLARE_DYNAMIC 宏如下:

```
#define DECLARE_DYNAMIC(class_name) \
public: \
   static CRuntimeClass class##class_name; \
   virtual CRuntimeClass * GetRuntimeClass() const;
```

##告诉编译器把两个字符串捆在一起

怎样才能把 CRuntimeClass 对象的内容指定并且连接起来,而且做的神不知鬼不觉呢 那就是使用 IMPLEMENT_DYNAMIC 宏,MFC 定义 IMPLEMENT_DYNAMIC 宏如下:

```
# define IMPLEMENT_DYNAMIC(class_name, base_class_name) IMPLEMENT_RUNTIMECLASS(class_name, base_class_name, 0xFFFF, NULL)
```

其中 _IMPLEMENT_RUNTIMECLASS 又是一个宏。

```
#define IMPLEMENT_RUNTIMECLASS(class_name, base_class_name, wSchema, pfnNew) static char _lpsz##class_name[] = #class_name; \
CRuntimeClass class_name::class##class_name = { _lpsz##class_name, sizeof(class_name), wSchema, pfnNew, RUNTIME_CLASS(base_class_name), NULL }; \
 static AFX_CLASSINIT init_##class_name ( & class_name::class##class_name); \
  CRuntimeClass * class_name::GetRuntimeClass( ) const \
     { return & class_name::class##class_name; }
```

其中 RUNTIME_CLASS 宏定义如下

```
# define RUNTIME_CLASS(class_name) ( & class_name::class##class_name)
```

IMPLEMENT_DYNAMIC 宏除了指定初值以外,还使用了一个结构 AFX_CLASSINIT 做连接工作,形成一张网

```
struct AFX_CLASSINIT
{   //构造函数(C++中 struct 和 class 都有构造函数)
    AFX_CLASSINIT(CRuntimeClass * pNewClass);
};
AFX_CLASSINIT::AFX_CLASSINIT(CRuntimeClass * pNewClass)
{   //用来链表(linked_list)的连接工作.
pNewClass->m_pNextClass = CRuntimeClass::pFirstClass;
    CRuntimeClass::pFirstClass = pNewClass;
}
```

例如：

```
DECLARE_DYNAMIC(CView)  //在头文件中
IMPLEMENT_DYNAMIC(CView, CWnd)  //在源文件中
```

上面可以展开为

```
//在头文件中
  public:
    static CRuntimeClass classCView;
    virtual CRuntimeClass * GetRuntimeClass() const;
//在源文件中
static char _lpszCView[] = "CView";
CRuntimeClass CView::classCView ={
     _lpszCView, sizeof(CView), 0xFFFF, NULL, &CWnd::classCWnd, NULL};
static AFX_CLASSINIT _init_CView(&CView::classCView);
CRuntimeClass * CView::GetRuntimeClass() const
     { return &CView::classCView; }
```

这样，就可以用两个宏 DECLARE_DYNAMIC Cxxx **和** IMPLEMENT_DYNAMIC Cxxx Cxxxbase **来完成链表的构造工作（图** 2**）。**

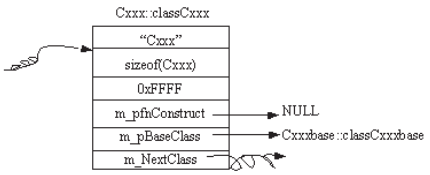


图 2

四、 最基本类 CObject

处理链表的头指针相对比较困难，不能套用现成的宏 DECLARE_DYNAMIC **和** IMPLEMENT_DYNAMIC**，怎样生成链表的头指针？必须特别设计如下（图** 3**）：**

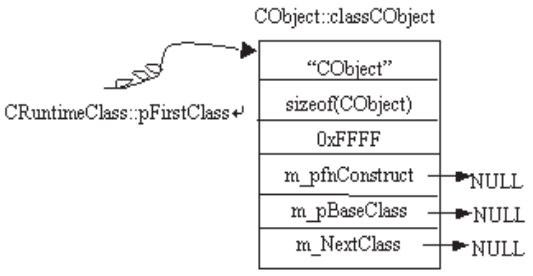


图 3

```
class CObject //在头文件中
{
  public:
    virtual CRuntimeClass * GetRuntimeClass() const;
  public:
    static CRuntimeClass classCObject;
}
//源文件
static char szCObject[] ="CObject";
struct CRuntimeClass CObject::classCObject = {
    szCObject, sizeof(CObject), 0xFFFF, NULL, NULL};
static AFX_CLASSINIT init_CObject(&CObject::classCObject);
CRuntimeClass * CRuntimeClass::pFirstClass = NULL;
CRuntimeClass * CObject::GetRuntimeClass() const
{ return &CObject::classCObject; }
```

五、IsKindOf 函数

当我们在 CCmdTarget**、**CWndThread**、**CWinApp**、**CWnd **的头文件使用** DECLARE_DYNAMIC **宏，源文件中使用** IMPLEMENT_DYNAMI **宏建立了类别型录网（图** 4**）。那实现动态类型识别，用什么函数来做呢？用** IsKindOf **函数，如何实现** IsKindOf **函数？实际上就是某个** CRuntimeClass **对象与类别型录中的元素一一比较。成功为** TRUE**，否则为** FALSE**：**

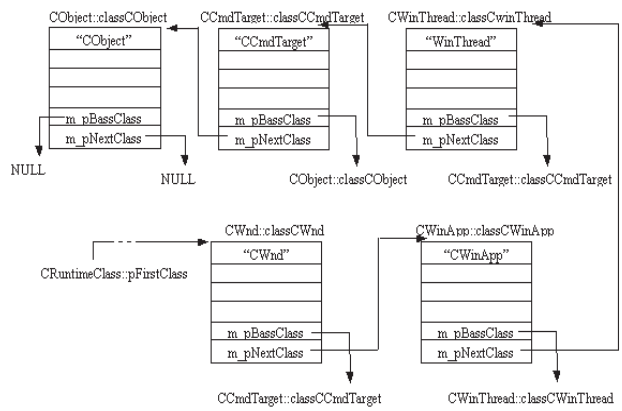


图 4

```
class Cobject //在头文件中
{
  public:
    BOOL IsKindOf(const CRuntimeClass * pClass) const;
};
BOOL CObject::IsKindOf(const CRuntimeClass * pClass) const //在源文件中
{
  CRuntimeClass * pClassThis = GetRuntimeClass();
  while( pClassThis != NULL){
      if ( pClassThis == pClass)
          return TRUE;
      pClassThis = pClassThis->m_pBaseClass;
   }
   return FALSE; //不匹配
}
```

例如：

VC + +与 Matlab 混合编程方法讨论

钱 云 夏祖明

一、前言

Matlab 其强大的数据处理能力和丰富的工具箱，使得它的编程极为简单，可以极大地缩短应用程序开发周期，提高编程效率和缩短理论方案研制周期。对于纯理论方案来说，Matlab 语言是优势较多。但由于其执行效率低，对于对实时性或速度要求较高的场合来说，就不太适应了。其对底层硬件的控制能力很差，所以对于半实物仿真和偏工程化的产品来说，Matlab 并不是一个很好的语言。对于发布软件公司来说，也希望发布的是一个可执行应用软件，而不是一个 Matlab 源代码的产品。所以我们通过把 Matlab 下的 . m 文件转化为 VC 可调用动态链接库（DLL），来达到脱离 Matlab 环境，生成可执行文件并提高执行效率，以消除 Matlab 不利因素。

二、Matlab 与 C 语言混合编程方法简介

Matlab 与 C 语言混合编程有三种方法：

A. 采用 Matlab 与 C 的接口规范来编程。

B. 用 Matlab 引擎来编程。

C. 用 Matlab 下的 . m 文件转化为 VC 可调用动态链接库（DLL）。

由于采用方法 A 和方法 B 都脱离不了 Matlab 运行环境，这里不作介绍，下文介绍方法 C 。

三、把 Matlab 下的 . m 文件转化为 VC 可调用的动态链接库

3. 1 用 . m 文件创建一个 VC 可调用的 DLL 文件

3. 1. 1 编辑一个 . m 文件

基于说明问题起见，用 Matlab 的 edit 编了一个简单的 myfunc. m 文件，程序如下所示：

```
function y = myfunct(x, b)
if b = = 1
  x = mod(x, 360) * pi/180;
  else
  x = mod(x, 2 * pi);
end
x1 = linspace(0, 2 * pi, 100);
y1 = sin(x1);
y = interp1(x1, y1, x, ´spline´);
return;
```

3. 1. 2 Matlab 编译前的准备工作

对 Matlab 编译环境进行设置。

a. 在 Matlab 环境中运行 mex - setup，按屏幕提示要求选择编译器类型，位置等有关信息。

b. 在 Matlab 环境中运行 mbuild - setup，设置方法与上面基本相同。

3. 1. 3 用 mcc 将 myfunc. m 转换为 matlab 可调用的 DLL

在 Matlab 环境中运行 mcc - t - h - L C - W lib ppp - T link lib myfunc. m。

编译完成后，将生成如下一些文件 ppp. exp、ppp. lib、myfunc. c、ppp. c、ppp. exports、myfunc. h、ppp. dll、ppp. h。其中有用的文件有三个，分别是 ppp. h、ppp. lib、ppp. dll，它们需要被添加到 VC 程序中去。

3. 1. 4 用 MFC 编译一个 VC + + 6. 0 可执行文件

a. 由于 dll 不能独立运行，所以要用 VC6. 0 创建一个 EXE 可执行程序。在 VC6. 0 中创建一个基于对话框的 MFC 工程，名为 mytest，具体过程参见一般的 VC 教程。在本例中 mytest 工程路径为 e hmat2c hmytest。

b. 对 VC 编程环境进行设置。选择 VC 编译器主菜单下 Tools - >options - >directories 选择 Show Directories for 列表框，分别把 Matlab 的包含文件路径（如 c hmatlab hertern hinclude），库文件（如 C hmatlab hextern hlib）路径添加到 VC 路径中去。

```
  CView * pView = new CView;
  pView -> IsKindOf(RUNTIME_CLASS(CWinApp)); //将返回 FALSE
  pView -> IsKindOf(RUNTIME_CLASS(CWnd)); //将返回 TRUE
```

六、结束语

RTTI 就是希望程序运行时能够检测某个对象是否‘属于某个类”。它为动态生成（动态的获得一个类的名字，产生一个对象）提供了基础，也为 MFC 中 Serialize 函数读文件恢复到原来的状态提供了基础。

参考文献

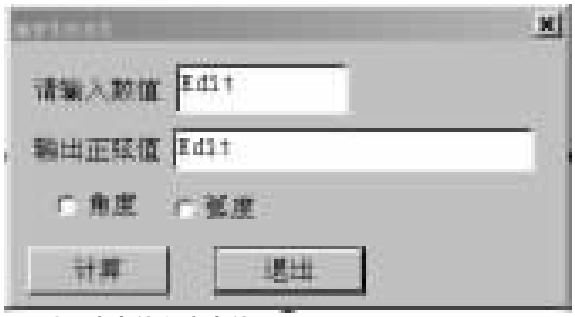
1. 侯俊杰著 . 深入浅出 Windows MFC 程序设计 . 华中理工大学出版社，1998. 4

2. Microsoft MSDN

收稿日期：2000 年 10 月 17 日）

c. 把 ppp. h、ppp. lib、ppp. dll 三个复制到 e hmat2c h mytest 目录下，以便 VC 调用 DLL 时能找到这三个文件。

d. 对对话框资源进行编辑，如下图所示



e. 引入头文件和库文件

在 mytestdlg. cpp 的头部加入一行：#include “ppp. h”，在 project>settings - >link 下的 object / libray modules 下加入 ppp. lib、libmx. lib、libmat. lib、libmatlb. lib、libmmfile. lib 目的是让 VC 能调用 ppp. dll，引入 libmx. lib、libmat. lib、libmatlb. lib 和 libmmfile. lib 的目的是让 VC 能调用 matlab 的数学库函数和一些功能性函数编译 VC 与 matlab 的代码接口和作者直接用 C 写的一些代码。

f. 对 ppp. h 做一点改动

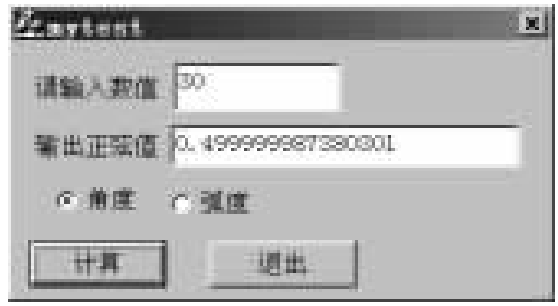
在#include “matlab. h”语句之后，加入一行 extern “C”{，在最后一行#endif 之前加入一行 }。

g. 对 VC 与 Matlab 接口进行编程

对“计算”按钮消息处理函数编程如下

```
void CMytestDlg: : OnOnCalculate()
{
// TODO: Add your control notification handler code here
   UpdateData(TRUE);
   pppInitialize();
   static double a[1] = { 0. 0 };
   static double b[1] = { 0. 0 };
     a[0] = m_din;
     b[0] = m_select + 1;
       mxArray * A = mclGetUninitializedArray();
   mxArray * B = mclGetUninitializedArray();
   mxArray * C = mclGetUninitializedArray();
   mlfAssign(& A, mlfDoubleMatrix(1, 1, a, NULL));
   mlfAssign(& B, mlfDoubleMatrix(1, 1, b, NULL));
   mlfAssign(& C, mlfMyfunc(A, B));
   double * md = mxGetPr(C);
     m_dout = md[0];
 mxDestroyArray(A);
   mxDestroyArray(B);
   mxDestroyArray(C);
     pppTerminate();
     UpdateData(FALSE);
}
```

h. 作完以上工作后，DLL 就已被成功链入 VC，再经 VC 编译器编译链接即可生成可执行文件，运行程序，在对话框中输入 30，选择角度选项，按计算按钮即可得到结果。



注意：libmx. lib、libmatlb. lib、libmmfile. lib、libmat. lib 文件并不是 Matlab 自带的，Matlab 只提供了 libmx. dll、libmatlb. dll、libmmfile. dll、libmat. dll 用户需要自己编译，把 VC 的 bin 目录下的 vcvars32. bat 拷贝的 C 盘根目录下，运行 msconfig 将 vcars32. bat 添加到 Auoexec. bat 中去。重新启动计算机，回到 MS_DOS 方式下在 VC 的目录下 extern hinclude 运行。

```
lib /def: libmat. def /machine: ix86 /out: libeng. lib
lib /def: libmatlb. def /machine: ix86 /out: libmatlab. lib
lib /def: libmmfile. def /machine: ix86 /out: libmmfile. lib
lib /def: libmx. def /machine: ix86 /out: libmx. lib
```

把生成的 libmx. lib、libmatlb. lib、libmmfile. lib、libmat. lib 文件复制到 c hmatlab hextern hlib 也就是添加到 VC 的编译路径中去）。

本文中的文件路径可能跟读者计算机中的路径有所不同，请参照修改。

收稿日期：2000 年 9 月 4 日）

建设人才高地，浦东表彰“优才”

金仕达总经理马平荣获“浦东开发建设贡献奖”

去年 12 月 8 日，2000 年度“浦东开发建设杰出人才表彰大会”在浦东新区办公中心隆重举行，该奖项是浦东新区政府为表彰浦东开发建设中做出突出贡献的杰出人才，弘扬创业、敬业精神而设立，体现了新区政府依托高层次优秀人才，加快浦东开发速度与质量的建设方针。此次表彰会上，上海金仕达多媒体有限公司总经理马平等荣膺“浦东开发建设贡献奖”。

金仕达多媒体公司在总经理马平的带领下，依靠科技创新，通过开发、汉化国外著名多媒体产品成为中国五大多媒体开发商之一。在此基础上，马平以树立电子商务专家形象，立足新世纪新经济为目标，致力于公司的业务范围的开拓，逐渐发展成为集电子商务、系统集成、多媒体等业务为一体，年营业额过千万的高科技公司。

浦东几年来的改革开放给金仕达多媒体的跳跃式发展提供了动力，同时，金仕达多媒体在前进中也为众多浦东企事业单位提供各种信息化服务，在浦东信息化进程中扮演了工程师的角色。五年来，金仕达多媒体曾先后与上海证券交易所、上海期货交易所、罗氏药业等浦东核心企事业单位、政府机关进行合作，已经成为浦东信息化建设的生力军。

在 VC 中用 DB - Library 快速访问 SQL Server 数据库技术

王 勇

一、 引言

SQL Server 是微软推出的一个中小型网络数据库管理系统，是现在使用最多的 DBMS 之一。DB - Library 是微软为 SQL Server 数据库提供的一套应用程序接口（API），通过这些 API 函数可以编写和 SQL Server 交互的应用程序。C 语言下提供了一套丰富的 DB - Library 函数，它们用于打开与数据库的连接、格式化查询语句、发送批量查询并接收查询结果、使用游标、两阶段事务处理、执行存储过程、批拷贝等，同时，DB - Library 也支持多用户并发环境。

DB - Library 实际上是比 ODBC、DMO、DAO、RDO 等数据库技术较低级的底层接口；但正是因为 DB - Library 是较 ODBC 等数据库引擎更底层，用它实现 Client / Server 模式的应用程序就更灵活、快捷。但随着面向对象编程的普及和深入，传统的 API 编程已经越来越少的被使用，大家总希望能将相关的功能尽可能地封装在一个类中，然后直接使用这个类提供的函数进行相应的操作。本文一方面说明了 DB - Library 的编程环境，同时对 DB - Library 函数在 VC 下进行了封装，提供了一个 SQL Server 数据库连接类（CDBLib）和一个数据库游标类（CDBCursor）。通过这两个封装类，可以非常方便地实现 SQL Server 数据库的连接和数据记录的读取以及对数据库中数据的存取和控制。

二、 DB - Library 编程环境

SQL Server 需要运行在 Windows NT 操作系统上，其客户端可以安装在 Windows 95 /98 或 NT 平台上，要用 DB - Library 进行编程必须有 ntwdblib. dll 动态链接库；此外，程序中需要有宏 #define DBNTWIN32，说明是 95 或 NT 下运行的 32 为程序）和相关的头文件（windows. h，sqlfront. h，sqldb. h，其包含顺序不能改变）。DB - Library 通过一个 DBPROCESS 结构和 SQL Server 交互，DBPROCESS 代表了客户和 SQL Server 的连接，绝大多数的 DB - Library 函数在调用时都会用到它；同一个应用中可以有多个 DBPROCESS，即可以与 SQL Server 建立多个连接。另外一个结构是 LOGINREC，该结构中包含了典型的用户登录信息，如用户名和用户口令。

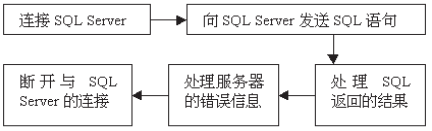


图 1

用 DB - Library 编程的基本过程可由图 1 表示：

三、 封装 DB - Library 函数

（1）数据库连接类（CDBLib）的封装

封装后的 CDBLib 类主要功能有：建立 / 断开与数据库服务器的连接、执行一个给定的 SQL 语句（无返回的结果集，对于有返回结果集的情况用下面的游标类处理）等。该类的对象模型如图 2 所示：

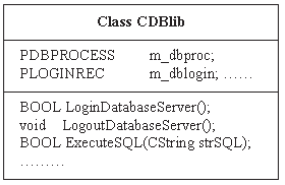


图 2

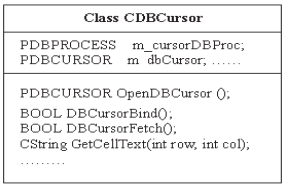


图 3

（2） 数据库游标类（CDBCursor）的封装

通过对游标函数的封装（对象模型如图 3），CDBCursor 类主要用来获取数据库中的记录。在调用打开游标函数、绑定数据并调用获取结果集函数后，那么此时中展现出来的是一张清晰的二维表；表格左上角是（0，0），即第一条记录的第一列，（0，1）表示第一条记录的第二列，其余类推。该类中用 GetCellText int row int col 函数来取得每个表格单元中的值，该值均已被转化为一个 CString 对象。此外，用户在打开游标的同时可以指定每移动一次游标所涵盖的记录数，以实现批量数据的读取。

（3） 代码实现

a. CDBLib 类：

```
BOOL CDBLib: : LoginDatabaseServer( ) /* 登录服务器函数 */
{
  if(m_dbproc ! = NULL) return TRUE; /* 已经登录 */
  else dbinit( ); /* 初始化 DB – Library */
  m_dblogin = dblogin( ); /* 创建一个 LOGINREC 结构 */
  DBSETLUSER(m_dblogin, m_userID); /* 登录用户名 */
  DBSETLPWD(m_dblogin, m_password); /* 用户密码 */
  DBSETLAPP(m_dblogin, "DBlib");
  ASSERT(dbsetlogintime(DEFAULT_LOGIN_TIME) = =
SUCCEED);
  /* 建立于数据库的连接, 得到指向 DBPROCESS 结构的指针 */
  m_dbproc = dbopen(m_dblogin, m_serverName);
  dbuse(m_databaseName); /* 设置当前使用的数据库 */
  return m_dbproc ! = NULL;  }
```

```
void CDBLib: : LogoutDatabaseServer( ) / * 断开与服务器的连接 * /
{     dbfreelogin(m_dblogin); / * 释放 PLOGINREC 结构 * /
 m_dblogin  = NULL ;
 dbclose(m_dbproc);  / * 关闭与数据库的连接 * /
 m_dbproc  = NULL ;
      dbexit( );  }
BOOL CDBLib: : ExecuteSQL(CString strSQL) / * 执行无放回结果集的 SQL 语句 * /
{
ASSERT(m_dbproc ! = 0);      dbfreebuf(m_dbproc);  / * 清除原有的 SQL 缓冲区 * /
      int retcode  = dbfcmd(m_dbproc, strSQL);
retcode  = dbsqlexec (m_dbproc);  / * 执行 SQL 语句 * /)
      if (dbresults (m_dbproc)  = = SUCCEED
 {  while (dbnextrow (m_dbproc) ! = NO_MORE_ROWS)
   { }    }
return retcode  = = 1;
}
```

b. CDBCursor 类

```
PDBCURSOR CDBCursor: : OpenDBCursor(LPCSTR strSQL, int scrollopt,    int concuropt,  int nrows ) / * 打开数据库游标函数 * /
{     if(m_cursorDBProc  = = NULL) / * 如果没有与数据库连接则返回空值 * /
            return NULL;
      m_pStatus  = new DBINT[nrows];
      m_cursorRows  = nrows; / * 每次取出的记录数 * /
      PDBCURSOR pCursor;
      pCursor  = dbcursoropen ( m_cursorDBProc, (LPCSTR) strSQL,  / * 打开游标 * /
      scrollopt, concuropt, (UINT)nrows, m_pStatus );
      m_dbCursor  = pCursor;
/ * 获取打开的数据库表中的记录数和列数,并把结果保存在相应的数据域中 * /
if(pCursor) dbcursorinfo (m_dbCursor , & m_KeysetCols , & m_KeysetRows );
return pCursor ! = NULL;
}
BOOL CDBCursor: : DBCursorBind( ) / * 绑定数据列 * /
{ / * 申请存放结果集的缓冲区 * /
m_ppResultCharData  =  new  char[  m_KeysetCols  *  AX_FIELD_LENGTH * m_cursorRows];
memset(m_ppResultCharData,  0  ,  sizeof(char)  *  m_KeysetCols * MAX_FIELD_LENGTH * m_cursorRows);
m_ppResultCharLength  =  new  DBINT[m_KeysetCols  *  m_cursorRows];
memset(m_ppResultCharLength,  0  ,  sizeof(DBINT)  *  m_KeysetCols * m_cursorRows);
      int retcode  = 0;
for (int i = 1; i  < = m_KeysetCols ; i + + ) / * 绑定数据 * /
{ retcode  =  dbcursorbind (m_dbCursor  ,  i  ,  NTBSTRING-BIND, (long) (MAX_FIELD_LENGTH), m_ppResultCharLength + (m_cursorRows * (i – 1)), (BYTE * )& (m_ppResultCharData[ (i – 1) * m_cursorRows *  MAX_FIELD_LENGTH]));
 } return (retcode ! = 0);
}
BOOL  CDBCursor: :  DBCursorFetch(int  fetchtype,  int rownum ) / * 取结果集 * /
{
   int retcode  = 0;
   m_fetchCount  + + ;  / * 私有变量,记录游标滚动次数 * /
retcode = dbcursorfetch ( m_dbCursor , fetchtype, rownum );
   return (retcode ! = FAIL);
}
/ * 得到结果集的列(字段名)信息并存入 m_colInfoList 中 * /
BOOL  CDBCursor: :  GetCursorColNameInfo(CStringList  &  colNameList)
{
   colNameList. RemoveAll( );
   char * col_name = new char[MAX_FIELD_NAME_LENGTH];
   int retcode;
   for (int i  = 1 ; i < = m_KeysetCols ; i + + )
   {
   retcode  =  dbcursorcolinfo(m_dbCursor,  (int) i,  col_name, NULL, NULL, NULL);
         colNameList. AddTail(col_name);
   }
   delete col_name;
   return retcode ! = 0;
}
CString CDBCursor: : GetCellText(int row,  int col) / * 获取指定表格单元的字符串 * /
{
   CString CellText;
   int k  = row  –  (m_fetchCount – 1) * m_cursorRows;
   CellText  = CString( & (m_ppResultCharData[
 ( col  *  m_cursorRows * MAX_FIELD_LENGTH)  +  ( k  * MAX_FIELD_LENGTH) ] ));
   return CellText;
}
```

四、实例与结论

```
   CDBLib  dblib( ″ sa″, ″ mypassword″, ″ mySqlSrvName″, ″ pubs″);
   if(dblib. LoginDatabaseServer( ))
   printf( ″Login Successed\n″);
   else  {
         printf( ″Login Failed\n″);
         return;
         }
   CDBCursor dbCursor;  / * 构造数据库游标对象 * /
   dbCursor. SetDBCursorProc(dblib. GetDBProc( ));  / * 将数据库连接指针赋给游标对象 * /
   CString strSQL  = ″ select title_id, title, type from titles ″;
/ * 要执行的 SQL 语句 * /
   UINT CursorRows  = 2; / /每次取出 2 条记录
 dbCursor. OpenDBCursor(strSQL, CUR_KEYSET, CUR_READ-
```

谈谈在 LOTUS NOTES R5 中访问关系数据库的方法

郝永峰

1989 年 12 月 6 日，莲花发展公司发表了 LOTUS NOTES，一种能极大改进公司内部、与其他公司及客户之间的环球通讯、协同工作和协调一致的软件产品。由于它有完备的电子邮件系统，领先的全文检索和复制功能，强大而灵活的复合文档数据库，还具有极强的安全措施和与 Internet / Intranet 的无缝集成，使之成为业界首选的软件平台，并得到广泛地应用。尽管 Lotus Notes 具有强大的功能和丰富的应用开发环境，但在联机事物的处理问题上，如：数据统计、分析、图表生成等方面上还是很薄弱的，这时就需要传统的关系型数据库管理系统来实现。如何把现有的关系型数据库信息转换到 Notes 数据库中，以便利用 Lotus Notes 中的许多良好性能共享数据库，是本文要讨论的问题。

这里我们采用基于 Microsoft 的开放数据库互连技术——ODBC，来达到数据库的共享，ODBC 是微软开放的服务结构中有关数据库的一个组成部分，它规定“以统一的 API 存取异构数据库信息”是对 SQL ACCESS GROUP 的 CLI 标准的一种实现，得到了世界上领先的数据库和应用程序开发商的广泛支持。通过使用这种统一的 API 建立的应用程序，对数据库的操作不依赖于任何数据库管理系统，不直接与任何 DBMS 数据库管理系统 打交道，从而直接实现应用程序对不同 DBMS 的共享。采用 ODBC 技术，应用程序只需关心数据的处理而不必考虑数据的存取，编程人员不必了解具体的 DBMS，从而极大地减少了软件开发人员的工作量，缩短了开发周期，提高了效率和软件的可靠性。

LOTUS NOTES 使用 ODBC 标准存取异种数据库信息。通过 Notes 里内嵌的公式或 Script 语言，您可以在 Notes 文档中引入非 Notes 信息，可把现成的数据转换成 Notes 数据库。在存取外部数据之前，您必须先定义一个数据源，以便让 ODBC 驱动程序管理器知道怎样存取数据。这些都通过控制面板中的“添加 ODBC 数据源”来完成。

LOTUS NOTES 提供了公式语言和 LotusScript 两种方法访问异种数据库：

在公式中提供了三个函数访问异种数据库，并返回一个值或列表值：

●@DBCOLUMN ODBC 返回表格中一列的全部数组，或者全部不同的数值；

●@DBLOOKUP （ODBC）返回在表格中一列通过匹配关键字选定的数值；

●@DBCOMMAND （ODBC）传递一个命令到外部 DBMS 并返回结果；

其中@DBCOLUMN、@DBLOOKUP 都只能提取数据，它们不能增加、删除、修改数据，或执行其他操作 @DBCOMMAND 能恢复数据或发送其他可以改变数据的 SQL 语句

下面给出它们的标准形式：

```
@ DbColumn( " ODBC": " NoCache"; data_source; user_ID; password; table; column: null_handling; "Distinct": sort)
@ DbLookup("ODBC": "NoCache"; "data_source"; "user_ID"; "password"; "table"; "column"; "key_column": "key"; "Dis-
```

```
ONLY, (UINT)CursorRows);
    dbCursor. DBCursorBind(); /* 绑定数据 */
int retcode = dbCursor. DBCursorFetch( FETCH_FIRST, 0 );
/* 第一次移动游标 */
    printf("%s \n", dbCursor. GetCellText(0, 0)); /* 第一行第一列 */
    printf("%s \n", dbCursor. GetCellText(0, 1)); /* 第一行第二列 */
    printf("%s \n", dbCursor. GetCellText(0, 2));
    printf("\n");
    printf("%s \n", dbCursor. GetCellText(1, 0)); /* 第二行第一列 */
    printf("%s \n", dbCursor. GetCellText(1, 1)); /* 第二行第二列 */
    printf("%s \n", dbCursor. GetCellText(1, 2));
    printf("\n");
    dbCursor. DBCursorFetch( FETCH_NEXT, 0 ); /* 第二次移动游标 */
    printf("%s \n", dbCursor. GetCellText(2, 0)); /* 第三行第一列 */
    . . . . . .
    dbCursor. CloseDBCursor(); /* 关闭游标 */
    dblib. LogoutDatabaseServer();
```

上述代码在 Visual C + + 6. 0 环境中，NT Server4. 0 平台和 SQL Server7. 0 / 6. 5 下调试通过。使用上面两个封装的 DB - Library 函数类，可以方便、快速地实现对 SQL Server 数据库的访问和管理。

参考文献

Microsoft SQL Server7. 0 Books On - line

收稿日期：2000 年 10 月 17 日）

tinct″: sort)
@ DbCommand(″ ODBC″: NoCache; data_source; user_ID; password; command_string)

注意：如果用户的 NOTES. INI 文件有下列语句：

NoExternalAPP = 1

则所有公式被禁止，并且看不到任何错误信息，公式不能执行。

在 LotusScript 中，ODBCConnection、ODBCQuery、ODBCResultSet 三个类为 Notes 提供了用 ODBC 标准存取外部数据库的属性和操作。记住，您必须把下面的语句放置在 GLOBAL 对象的 OPTIONS 事件中才能访问 ODBC 类：

USELSX “* LSXODBC”

● **ODBCCONNECTION 类 代表了与数据源连接的 ODBC DATA ACCESS 特性；**

● **ODBCQUERY 类 代表定义一个 SQL 语句的 ODBC 数据库存取特性，一个查询被使用或者有效确认之前，必须从属与一个有效的连接；**

● **ODBCRESULTSET 类 代表在集合上执行操作的 ODBC 数据存取特性。**

下面以 Notes 访问 SQL Server 数据库为例，介绍 Notes 访问异种数据库的一种实现方法。已知 ios 数据库中存在三个表 分别为 sale 销售表 、item 商品表 、itbatch 帐目表 。其结构如下：

item 表	sale 表	itbatch 表
it_id 商品编号	sa_id 销售编号	ib_id 帐目编号
it_name 名称	sa_date 日期	ib_it_id 商品编号
it_unit 单位	sa_it_id 商品编号	ib_b_amt 进货数
it_madein 产地	sa_amt 数量	ib_ytd_amt 现存数

我们想知道商品当天的销售数、库存量及其他信息。基本编程思路是：按被查询字段基本结构相应在 Notes 数据库里建立一个同样结构的表单，以便把 SQL server 某个字段的信息经转换后存在 Notes 表单相应字段中，建立一个操作，用 Lotus Notes 语言编写转换程序，在视图中运行，以实现外部数据库信息向 Notes 数据库转换。具体实现步骤如下：

1. 在 NOTES 中新建一个数据库，取名为 ODBC. NSF，在这个数据库里创建一个表单，取名为 Item_Info。

其域名内容如下： it_id it_name it_unit it_madein sa_amt ib_ytd_amt ib_b_amt。

在 ODBC. NSF 数据库中建立一个标准视图，取名为 Today_Info，定义该视图选项为 SELECT form = ”Item_Info” ；

同时创建一个操作，标题为 ‘当日商品销售信息”，在 CLICK 事件中作如下编程：

```
Sub Click(Source As Button)
  Dim session As New notessession
  Dim db As notesdatabase
  Dim doc As notesdocument
  Dim view As notesview
  Dim Nowdate As String
  Dim con As New odbcconnection
  Dim qry As New odbcquery
  Dim result As New odbcresultset
  Nowdate = Format(Now, ″yyyy – mm – dd″))
  Set db = session. currentdatabase
  Set view = db. getview(″Today_Info″
  If con. connectto(″ios″) Then
     Set qry. connection = con
     ,发出查询请求
     qry. sql = ″ select it_id, it_name, it_unit, it_madein, sa_amt, ib_ytd_amt, ib_b_amt from sale, item, itbatch where ib_it_id = it_id and sa_it_id = it_id and sa_date = ’”& Nowdate & ”’″
     Set result. query = qry
     If Not result. execute( ) Then
     Messagebox( ″ error: ″ & result. geterrormessage (db_lasterror))
     End If
     columns = result. numcolumns
     Do
        Call result. nextrow( )
        Set doc = New notesdocument(db)
        Doc. form = “Item_Info”
        ,取出结果集并存入相应字段中
        doc. it_id = result. getvalue(1)
        doc. it_name = result. getvalue(2)
        doc. it_unit = result. getvalue(3)
        doc. it_madein = result. getvalue(4)
        doc. sa_amt = result. getvalue(5)
        doc. ib_ytd_amt = result. getvalue(6)
        doc. ib_b_amt = result. getvalue(7)
     Loop Until result. isendofdata
     ,与数据库断开联系
     Call con. disconnect( )
  Else
     Messagebox(″could not connect to server″)
  End If
End Sub
```

2. 在视图中点击 ‘当日商品销售信息”按钮，就可以将 SQL Serve 中数据库的信息一次转换到 NOTES 数据库了。

以上方法已在机器上调试通过，运行环境为： Windows98，LOTUS NOTES R5，PII/350，64M **内存。**

收稿日期：2000 年 10 月 17 日）

北京飞天诚信 ‘软件加密锁 ’走向世界

飞天诚信公司软件 IV 型加密锁，以其优异的性能和良好的服务给中国软件业稳稳的加了一把安全的锁，因而越来越受到市场的欢迎。2000 年下半年，产品更是声名远扬，现已出口欧洲、美国、日本、韩国、台湾等地，并建立了代理渠道以更好的服务国外市场。最近，国外订单纷至沓来，飞天生产部近几月一直加班工作，最大限度满足海外客户需求；质检部门也毫不松懈，绝不让一个不合格产品流出公司。

深入 Java 编程——Java 的字节代码

邵 刚

Java 程序员很少注意程序的编译结果。事实上，Java 的字节代码向我们提供了非常有价值的信息。特别是在调试排除 Java 性能问题时，编译结果让我们可以更深入地理解如何提高程序执行的效率等问题。其实 JDK 使我们研究 Java 字节代码变得非常容易。本文阐述怎样利用 JDK 中的工具查看解释 Java 字节代码，主要包含：

- Java 类分解器——javap
- Java 字节代码是怎样使程序避免程序的内存错误
- 怎样通过分析字节代码来提高程序的执行效率
- 利用第三方工具反编译 Java 字节代码

一、Java 类分解器——javap

大多数 Java 程序员知道他们的程序不是编译成本机代码的。实际上，程序被编译成中间字节代码，由 Java 虚拟机来解释执行。然而，很少程序员注意一下字节代码，因为他们使用的工具不鼓励他们这样做。大多数的 Java 调试工具不允许单步的字节代码调试。这些工具要么显示源代码，要么什么都不显示。

幸好 JDK 提供了 Java 类分解器 javap，一个命令行工具。javap 对类名给定的文件 （class）提供的字节代码进行反编译，打印出这些类的一个可读版本。在缺省情况下，javap 打印出给定类内的公共域、方法、构造函数，以及静态初始值。

1. javap 的具体用法

语法： javap ＜选项＞ ＜类名＞. . .

其中选项包括：

2. 应用实例

让我们用一个例子来进一步说明如何使用 javap。

```
// Imports
import java. lang. String;
public class ExampleOfByteCode {
  // Constructors
  public ExampleOfByteCode() { }
  // Methods
  public static void main(String[] args) {
    System. out. println(″Hello world″);
  }
}
```

编译好这个类以后，可以用一个十六进制编辑器打开 . class 文件，再通过虚拟机说明规范来解释字节代码的含义，但这并不是好方法。利用 javap，可以将字节代码转换成人们可以阅读的文字，只要加上 - c 参数：

```
javap - c ExampleOfByteCode
```

输出结果如下：

```
Compiled from ExampleOfByteCode. java
public class ExampleOfByteCode extends java. lang. Object {
    public ExampleOfByteCode();
    public static void main(java. lang. String[]);
}
Method ExampleOfByteCode()
   0 aload_0
   1 invokespecial #6 <Method java. lang. Object()>
   4 return
Method void main(java. lang. String[])
   0 getstatic #7 <Field java. io. PrintStream out>
   3 ldc #1 <String ″Hello world″>
   5 invokevirtual #8 <Method void println(java. lang. String)>
   8 return
```

从以上短短的几行输出代码中，可以学到关于字节代码的许多知识。在 main 方法的第一句指令是这样的：

```
0 getstatic #7 <Field java. io. PrintStream out>
```

开头的初始数字是指令在方法中的偏移，所以第一个指令的偏移是 0。紧跟偏移的是指令助记符。在本例中，getstatic 指令将一个静态字段压入一个数据结构，我们称这个数据结构为操作数堆栈。后续指令可以通过此结构引用这个字段。紧跟 getstatic 指令后面的是压到哪个字段中去。这里的字段是“#7 ＜Field java. io. PrintStream out＞”。如果直接察看字节代码，这些字段信息并没有直接存放到指令中去。事实上，就象所有 Java 类使用的常量一样，字段信息存储在共享池中。在共享池中存储字段信息可以减小字节代码的大小。这是因为指令仅仅需要存储的是整型索引号，而不是将整个常量存储到常量池中。本例中，字段信息存放在常量池的第七号位

置。存放的次序是由编译器决定的，所以看到的是“#7”。通过分析第一行指令，我们可以看出猜测其它指令的含义还是比较简单的。“ldc”（载入常量）指令将常量“Hello World.”压入操作数堆栈。“invokevirtual”激发 println 方法，此方法从操作数堆栈中弹出两个参数。不要忘记象 println 这样的方法有两个参数：明显的一个是字符串参数，加上一个隐含的“this”引用。

二、Java 字节代码是怎样使程序避免程序的内存错误

Java 程序设计语言一直被称为 Internet 的安全语言。从表面上看，这些代码象典型的 C++代码，安全从何而来？安全的重要方面是避免程序的内存错误。计算机罪犯利用程序的内存错误可以将他们的非法代码加到其它安全的程序中去。Java 字节代码是站在第一线抵御这种攻击的。

1. 类型安全检测实例

以下的例子可以说明 Java 具体是怎样做的。

```
public float add(float f, int n) {
return f + n;
}
```

如果你将这段代码加到第一个例子中去，重新编译，运行 javap，分析情况如下：

```
Method float add(float, int)
  0 fload_1
  1 iload_2
  2 i2f
  3 fadd
  4 freturn
```

在 Java 方法的开头，虚拟机将方法的参数放到一个被称为局部变量表的数据结构中。从名字就可以看出，局部变量表包含所有声明的局部变量。在本例中，方法从三个局部变量表实体开始，这些是 add 方法的三个参数。位置 0 保存该方法返回类型，位置 1 和 2 保存浮点和整型参数。

为了真正操纵变量，它们必须被装载（压）到操作数堆栈。第一条指令 fload_1 将浮点参数压到操作数堆栈的位置 1。第二条指令 iload_2 将整型参数压到操作数堆栈的位置 2。有趣的是这些指令的前缀是以“i”和“f”开头的，这表明 Java 字节代码的指令按严格的类型划分的。如果参数类型与字节代码的参数类型不符合，虚拟机将拒绝不安全的字节代码。更妙的是，字节代码被设计成仅执行一次类型安全检查——当加载类的时候。

2. Java 中的类型安全检测

类型安全是怎样增强系统安全性的呢？如果攻击者可以让虚拟机将整型变量当成浮点变量，或更严重，很容易预见计算的崩溃。如果计算是发生在银行账户上的，牵连的安全问题是很明显的。更危险的是欺骗虚拟机将整型变量编程一个对象引用。在大多数情况下，虚拟机将崩溃，但是攻击者只要找到一个漏洞即可。不要忘记攻击者不需要手工查找——更好且容易的办法是写一个程序产生大量变换的坏的字节代码，直到找到一个可以危害虚拟机的。

另一种字节代码保护内存安全的是数组操作。“aastore”和“aaload”字节代码操作 Java 数组，而它们一直要检查数组的边界。当调用者超越数组边界时，这些字节代码将产生数组溢出错误（ArrayIndexOutOfBoundsException）。也许所有应用中最重要的检测是分支指令，例如，以“if.”开始的字节代码。在字节代码中，分支指令在同一个方法中只能跳转到另一条指令。向方法之外传递控制的唯一办法是返回，产生一个异常，或执行一个唤醒（invoke）指令。这不仅关闭了许多易受攻击的大门，也防止由伴随引用和堆栈的崩溃导致程序错误。如果你曾经用系统调试器打开过代码中随机定位的程序，你对这些程序错误会很熟悉。

需要着重指出的是：所有的这些检测是由虚拟机在字节代码级上完成的，不仅仅是编译器。其它编程语言的编译器象 C++，可以防止一些我们在上面讨论过的内存错误，但这些保护是基于源代码级的。操作系统将读入执行任何机器代码，而不管这些代码是由小心翼翼的 C++编译器还是由邪恶的攻击者产生的。简单地说，C++是在源程序级上面向对象的，而 Java 的面向对象特性扩展到已经编译好的字节代码上。

三、怎样通过分析字节代码来提高程序的执行效率

不管你注意它们与否，Java 字节代码的内存和安全保护都客观存在，那为什么还要那么麻烦去看字节代码呢？其实，就如在 DOS 下深入理解汇编就可以写出更好的 C++代码一样，了解编译器怎样将你的代码翻译成字节代码可帮助你写出更有效率的代码，有时候甚至可以防止不知不觉的程序错误。

1. 为什么在进行字符串合并时要使用 StringBuffer 来代替 String，我们看以下代码：

```
//Return the concatenation str1 +str2
   String concat(String str1, String str2) {
      return str1 + str2;
   }
    //Append str2 to str1
   void concat(StringBuffer str1, String str2) {
      str1.append(str2);
   }
```

试想一下每个方法需要执行多少函数。编译该程序并执行 javap，输出结果如下：

```
Method java.lang.String concat(java.lang.String, java.lang.String)
  0 new #6 <Class java.lang.StringBuffer>
  3 dup
  4 aload_1
  5 invokestatic #14 <Method java.lang.String valueOf(java.lang.Object)>
```

```
    8 invokespecial # 9 <Method java.lang.StringBuffer
(java.lang.String)>
   11 aload_2
   12 invokevirtual #10 <Method java.lang.StringBuffer ap-
pend(java.lang.String)>
   15 invokevirtual #13 <Method java.lang.String toString()>
   18 areturn
      Method void concat(java.lang.StringBuffer, java.lang.
String)
    0 aload_1
    1 aload_2
    2 invokevirtual #10 <Method java.lang.StringBuffer ap-
pend(java.lang.String)>
    5 pop
    6 return
```

第一个 concat 方法有五个方法调用：new、invokestatic、invokespecial 和两个 invokevirtual。这比第二个 cacat 方法多了好多工作，而第二个 cacat 只有一个简单的 invokevirtual 调用。String 类的一个特点是其实例一旦创建，是不能改变的，除非重新给它赋值。在我们学习 Java 编程时，就被告知对于字符串连接来说，使用 StringBuffer 比使用 String 更有效率。使用 javap 分析这点可以清楚地看到它们的区别。如果你怀疑两种不同语言架构在性能上是否相同时，就应该使用 javap 分析字节代码。不同的 Java 编译器，其产生优化字节代码的方式也不同，利用 javap 也可以清楚地看到它们的区别。以下是 JBuilder 产生字节代码的分析结果：

```
Method java.lang.String concat(java.lang.String, ja-
va.lang.String)
    0 aload_1
    1 invokestatic # 5 <Method java.lang.String valueOf
(java.lang.Object)>
    4 aload_2
    5 invokestatic # 5 <Method java.lang.String valueOf
(java.lang.Object)>
    8 invokevirtual # 6 <Method java.lang.String concat
(java.lang.String)>
   11 areturn
```

可以看到经过 JBuilder 的优化，第一个 concat 方法有三个方法调用：invokestatic、两个 invokevirtual。这还是没有第二个 concat 方法简洁。

不管怎样，熟悉即时编译器 JIT Just - in - time 。因为当某个方法被第一次调用时，即时编译器将对该虚拟方法表中所指向的字节代码进行编译，编译完后表中的指针将指向编译生成的机器码，这样即时编译器将字节代码重新编译成本机代码，它可以使你进行更多 javap 分析没有揭示的代码优化。除非你拥有虚拟机的源代码，你应当用性能基准来进行字节代码分析。

2. 防止应用程序中的错误

以下的例子说明如何通过检测字节代码来帮助防止应用程序中的错误。首先创建两个公共类，它们必须存放在两个不同的文件中。

```
public class ChangeALot
{
  // Variable
   public static final boolean debug = false;
   public static boolean log = false;
}
public class EternallyConstant
{
  // Methods
   public static void main(String [] args)
  {
       System.out.println("EternallyConstant beginning ex-
ecution");
      if (ChangeALot.debug)
         System.out.println("Debug mode is on");
      if (ChangeALot.log)
         System.out.println("Logging mode is on");
   }
}
```

如果运行 EternallyConstant 类，应该得到如下信息：

```
EternallyConstant beginning execution.
```

现在我们修改 ChangeALot 文件，将 debug 和 log 变量的值都设置为 true。只重新编译 ChangeALot 文件，再运行 EternallyConstant，输出结果如下：

```
EternallyConstant beginning execution
Logging mode is on
```

在调试模式下即使设置 debug 为 true，“Debug mode is on”还是打印不出来。答案在字节编码中。运行 javap 分析 EternallyConstant 类，可看到如下结果：

```
Compiled from EternallyConstant.java
public class EternallyConstant extends java.lang.Object
{
   public EternallyConstant();
   public static void main(java.lang.String[]);
}>
Method EternallyConstant()
   0 aload_0
   1 invokespecial #1 <Method java.lang.Object()>
   4 return
Method void main(java.lang.String[])
   0 getstatic #2 <Field java.io.PrintStream out>
   3 ldc #3 <String "EternallyConstant beginning execution">
   5 invokevirtual #4 <Method void println(java.lang.String)>
   8 getstatic #5 <Field boolean log>
  11 ifeq 22
  14 getstatic #2 <Field java.io.PrintStream out>
  17 ldc #6 <String "Logging mode is on"
     19 invokevirtual # 4 <Method void println
(java.lang.String)>
  22 return
```

很奇怪吧！由于有“ifep”检测 log 字段，代码一点都不检测 debug 字段。因为 debug 字段被标记为 final，编译器知道

debug 字段在运行过程中不会改变。所以 “if”语句被优化，分支部分被移去了。这是一个非常有用的优化，因为这使你可以在引用程序中嵌入调试代码，而设置为 false 时不用付出代价，不幸的是这会导致编译混乱。如果改变了 final 字段，记住重新编译其它引用该字段的类。这就是引用有可能被优化的原因。Java 开发工具不是每次都能检测这个细微的改变，这些可能导致临时的非常程序错误。在这里，古老的 C + + 格言对于 Java 环境来说一样成立：‘每当迷惑不解时，重新编译所有程序”。

四、利用第三方工具反编译 Java 字节代码

以上介绍了利用 javap 来分析 Java 字节代码，实际上，利用第三方工具，可以直接得到源代码。这样的工具有很多，其中 NMI s Java Code Viewer NJCV 是其中使用起来比较方便的一种。

1. NMI s Java Code Viewer 简介

NJCV 针对编译好的 Java 字节编码，即 . class 文件、. zip 或 . jar 文件。. jar 文件实际上就是 . zip 文件。利用 NJCV 这类反编译工具，可以进一步调试、监测程序错误，进行安全分析等等。通过分析一些非常优秀的 Java 代码，我们可以从中学到许多开发 Java 程序的技巧。

NMI s Java Code Viewer 的最新版本是 4. 8. 3，而且只能运行在以下 Windows 平台：

- Windows 95 /98
- Windows 2000
- Windows NT 3. 51 /4. 0

2. NMI s Java Code Viewer 应用实例

我们以前面列举到的 ExampleOfByteCode. class 作为例子。打开 File 菜单中的 open 菜单，打开 Java 字节代码文件，Java class files 中列出了所有与该文件在同一个目录的文件。选择要反编译的文件，然后在 Process 菜单中选择 Decompile 或 Dissasemble，反编译好的文件列在 Souce - code files 一栏。用 NMI s Java Code Viewer 提供的 Programmer s File Editor 打开该文件，瞧，源代码都列出来了。

```
// Processed by NMI´s Java Code Viewer 4.8.3 (c) 1997 – 2000 B. Lemaire
// Website: http://njcv.htmlplanet.com E – mail: info@ njcv.htmlplanet.com
// Copy registered to Evaluation Copy
// Source File Name: ExampleOfByteCode.java
import java.io.PrintStream;
public class ExampleOfByteCode
{
    public ExampleOfByteCode() {
    }
    public static void main(String args[])
      {
       System.out.println(˝Hello world˝);
    }
    public float add(float f, int n)
   {
       return f + (float)n;
    }
    String concat(String str1, String str2)
   {
       return str1 + str2;
    }
    void concat(StringBuffer str1, String str2) {
       str1.append(str2);
    }
}
```

NMI s Java Code Viewer 也支持直接从 jar /zip 文件中提取类文件。反编译好的文件缺省用 . nmi 扩展名存放，用户可以设置 . java 扩展名。编辑源文件时可以使用 NJCV 提供的编辑器，用户可以选择自己喜欢的编辑器。其结果与原文件相差不大，相信大家会喜欢它。

五、结束语

了解一些字节代码可以帮助从事 Java 程序编程语言的程序员们编程。javap 工具使查看字节代码变得非常容易，第三方的一些工具使代码的反编译易如反掌。经常使用 javap 检测代码，利用第三方工具反编代码，对于找到特别容易忘记的程序错误、提高程序运行效率、提高系统的安全性和性能来说，其价值是无法估量的。

随着 Java 编程技术的发展，Java 类库不断完善，利用 Java 优越的跨平台性能开发的应用软件也越来越多。Oracle 用 Java 编写了 Oracle 8i 的 Enterprise Manager，以及其数据库的安装程序；Inprise 公司的 Borland JBuilder 3. 5 也用 Java 写成；一些 Internet 电话也使用了 Java 技术，如 MediaRing、DialPad 的网络电话采用了 Java 的解决方案；甚至以上提到的 NMI s Java Code Viewer 也是用 Java 写成的。Java2 已使 Java 运行性能基本接近 C + + 程序的执行速度，结合 Enterprise JavaBean、Servlet 以及 COBRA、RMI 技术，Java 的功能会越来越强大，其应用也将日益广泛。

参考文献

1. Think in Java Prentice Hall Bruce Eckel
2. Sun Java Web Site - JDC Tech Tips
3. Java in a Nutshell O’eilly and Assoc. Mike Loukides ed.
4. Just Java 2 Prentice Hall Peter van der Linden
5. The Java Virtual Machine Specifications Addison Wesley Tim Lindholm and Frank Yellin

PROGRAMMING INSIDE JAVA - JAVA BYTE CODE

收稿日期：2000 年 9 月 11 日）

用 VB 实现网页一键通

虞东海

摘 要 本文详细讨论了用 Visual Basic 6.0 实现网页一键通的原理及技术细节 而且还涉及了有关绿色软件的开发问题 最后在文中还给出了一个简单的实现网页一键通的程序的源程序.

关键词 热键，消息处理过程，ini 文件，Visual Basic

一、引言

在互联网发达的今天，我们如果经常上某个门户网站，为了方便，我们可以通过控制面板中的“Internet”选项将其设为主页，这样在我们每次启动 IE 时它将自动连接到该网站。但是要是我们还经常上其它的网站呢？还有其它的方法吗

记得在几年前，国内某著名电脑厂商在宣传其某新产品时 总是特别强调它的一个新增功能：内置数个上网快捷键，实现一键上网。不过它是通过硬件方式来实现，今天我们就通过软件的方式来模拟该功能以实现网页一键通。

二、实现原理

在 DOS 的时代，要通过按下某个快捷键而自动执行某个程序 我们常常会通过调用 DOS 的中断来拦截中断向量的方式来实现。

在 Windows 时代，我们要实现此功能，可以在某个窗口中用 Windows API 中的 RegisterHotKey 函数定义一个 组）快捷键，此后不管用户在执行什么程序时 只要按下了该快捷键，系统就会发送一条 WM_HOTKEY 消息给待接收该消息的窗口，而当该窗口收到 WM_HOTKEY 消息时，只需判断是否为该程序所定义的快捷键被按下，如果是就立即对该事件作出响应，自动连接到预设置的网址。

三、技术细节

以上我们提出了实现原理 但是仍有许多技术细节需要实现

1. 如何定义系统热键

在上面我们说过可以通过 Windows API 中的 RegisterHotKey 函数来定义热键，下面是该函数的有关说明：

声明：Public Declare Function RegisterHotKey Lib “user32” Alias “RegisterHotKey” ByVal hwnd As Long ByVal id As Long ByVal fsModifiers As Long ByVal vk As Long As Long。

参数说明：

hwnd 参数→接收该热键的窗口的句柄

id 参数→热键的标识，可以设为 1

fsModifiers 参数→定义辅助键 alt shift control 等 的组合

vk 参数→定义热键

以上我们定义了系统热键，要注意的是在程序结束之前要使用 Windows API 中的 UnRegisterHotkey 函数将系统热键取消，否则将会出现错误。

声 明 ： Public Declare Function UnregisterHotKey Lib “user32” Alias “UnregisterHotKey” ByVal hwnd As Long ByVal id As Long As Long。

各参数意义同上。

2. 如何保存用户设置

在每次程序启动时 如何读出上次用户所设定的快捷键及相对应的网址，对这个问题一般有两个解决方法 其一是通过读取注册表的方式。即在每次程序启动时，读取注册表中的有关设置 并在每次程序结束时，又重新将当前设置写入注册表中；其二是通过读取 ini 文件的方式。这项虽然是 Windows 3.0 时代的技术却功能强大。由于通过注册表的方法不符合绿色软件的要求，同时对注册表的操作有一定的危险性 因此本文拟将采用第二种方法。

下面是读取 ini 文件的 GetPrivateProfileString 函数的说明

声 明 ： Public Declare Function GetPrivateProfileString Lib “kernel32” Alias “GetPrivateProfileStringA” ByVal lpApplicationName As String ByVal lpKeyName As Any ByVal lpDefault As String ByVal lpReturnedString As String ByVal nSize As Long ByVal lpFileName As String As Long。

参数说明：

lpApplicationName →欲在其中读取的小节名称，该字串不区分大小写。如设为 vbNullString，就在 lpReturnedString 缓冲区内装载这个 ini 文件所有小节的列表

lpKeyName →欲获取的项目名 该字串也不区分大小写。如设为 vbNullString，就在 lpReturnedString 缓冲区内装载指定小节所有项的列表

lpDefault →当指定的条目没有找到时返回的默认值。可设为空 （“”）

lpReturnedString →指定一个字串缓冲区，长度至少为 nSize

nSize →指定装载到 lpReturnedString 缓冲区的最大字符数量

下面是保存 ini 文件的 WriteProfileString 函数的说明：

声 明 ： Public Declare Function WriteProfileString Lib

"kernel32" Alias "WriteProfileStringA" ByVal lpszSection As String ByVal lpszKeyName As String ByVal lpszString As String As Long。

参数说明：

lpszSection →指定要写入新串的小节 如果该小节不存在，会创建这个小节。这个字串不区分大小写

lpszKeyName →要设置的项目名，这个字串不区分大小写。用 vbNullString 可删除这个小节的所有设置项

lpszString →指定在这个项中写入的字串值。用 vbNullString 表示删除这个项现有的字符串。

3. 如何处理 WM_HOTKEY 消息

在讨论这个问题之前，我们先要对 Windows 的消息系统和窗口过程进行了解。

有一定 C + + 语言编程基础的人都知道：Windows 系统与窗口之间的通讯是通过消息完成的。消息是由 Windows 系统或应用程序产生的 系统对每一个输入事件都要产生消息 并把消息发送给窗口过程。在 Windows 系统中每个窗口都有一个相应的消息过程 用于处理发送或投递到该窗口的所有消息的函数。在默认情况下，应用程序的窗口过程把它不处理的任何消息传给默认窗口过程函数 DefWindowProc 函数进行处理。

由于 Windows 系统的默认窗口过程函数 DefWindowProc 对 WM_HOTKEY 消息并不作处理，同时我们无法对其进行修改以处理该消息，只能采用一种类似 DOS 环境下截获中断的方法来自行处理消息：

1 在程序启动后，立即调用 Windows API 中的 GetWindowLong 函数来记录该窗口的默认过程处理函数的地址并将其保存在一个全局变量中。

声明：Public Declare Function GetWindowLong Lib "user32" Alias "GetWindowLongA" ByVal hwnd As Long ByVal nIndex As Long As Long。

说明：该函数用来从指定窗口的结构中取得信息 对我们有用的是 GWL_WNDPROC 。

参数说明：

hwnd 参数→想从中获取信息的窗口的句柄

nIndex 参数→指定要取得的信息 下面是各值及其对应的含义 对我们有用的是 GWL_WNDPROC ：

GWL_EXSTYLE →扩展窗口样式

GWL_STYLE →窗口样式

GWL_WNDPROC →该窗口的窗口函数的地址

GWL_HINSTANCE →拥有窗口的实例的句柄

GWL_HWNDPARENT→该窗口的父窗口的句柄

GWL_ID →该窗口的标识符

GWL_USERDATA →含义由应用程序规定

2 编写自己的消息处理过程，并用 Windows API 中的 SetWindowLong 函数将自己编写的处理过程设置为该窗口的默认过程处理函数。

消息处理过程函数的参数及其类型必须严格按照下面的格式：

Public Function wndproc(ByVal hwnd As Long, ByVal Msg As Long, ByVal wParam As Long, ByVal lParam As Long) As Long

hwnd 参数→ 窗口句柄，它决定消息发送到哪一个窗口；

Msg 参数→ 消息标识，它是一个命名的常量，由它来标明消息的目的；

wParam lParam 参数→ 消息参数，它指定窗口过程在处理消息时所用的数据或数据的位置。

下面是 SetWindowLong 函数的声明：Public Declare Function SetWindowLong Lib "user32" Alias "SetWindowLongA" ByVal hwnd As Long ByVal nIndex As Long ByVal dwNewLong As Long As Long。

说明：该函数用来在窗口结构中为指定的窗口设置信息。

参数说明：

hwnd 参数和 nIndex 参数含义同上

dwNewLong 参数→由 nIndex 指定的窗口信息的新值，在这里为自己的消息处理过程的地址，我们可以用 VB 的求地址运算符 AddressOf 来得到它。

3 在程序结束时，重新让 Windows 的默认消息处理过程去处理消息。方法有二个，第一个就是利用上面的 SetWindowLong 函数，第二个就是利用 CallWindowProc 函数。要注意的是这一步很重要 否则会造成 Windows 消息处理机制混乱而出现非法操作的情况，严重的甚至可能死机。下面是 CallWindowProc 函数的声明：Public Declare Function CallWindowProc Lib "user32" Alias "CallWindowProcA" ByVal lpPrevWndFunc As Long ByVal hWnd As Long ByVal Msg As Long ByVal wParam As Long ByVal lParam As Long As Long。

参数说明：

lpPrevWndFunc 参数应为默认消息处理过程的地址，其余的参数含义同上。

4. 如何确定辅助键

在说到定义系统热键时我们知道 RegisterHotKey 函数中的 fsModifiers 参数是定义辅助键的组合的，辅助键共有三个 Alt、Shift、Ctrl 在程序中如何根据用户设定来确定 fsModifiers 参数也是一个值得讨论的问题。

第一个方法是通过在 if 语句中嵌套多重 if 语句来作判断，该方法是最直接的方法也是最繁锁的方法，而且会给操作 ini 文件带来不便。

第二种方法就是将 Ctrl、Alt、Shift 键的选中状态转化为数字，当用户选定后，将其值写入 ini 文件 在定义热键时再从 ini 文件中读取该值并将其转化为相应的各键的状态即可。要注意的是在转化过程中必须保证数字与各键状态的一一对应性。

5. 如何连接到预定的网址

在 wndproc 函数中，我们必须对正确的消息作出响应——连接到预定的网址 这可以通过两种方法实现：

1 运用 VB 中的外部程序执行命令 Shell 和 Start 指令

shell ″start http://www.scu.edu.cn″ ′连接到四川大学的网址

2 调用 Windows API 中的 ShellExecute 函数 下面是该函数的说明

Public Declare Function ShellExecute Lib ″shell32.dll″ Alias ″ShellExecuteA″(ByVal hwnd As Long, ByVal lpOperation As String, ByVal lpFile As String, ByVal lpParameters As String, ByVal lpDirectory As String, ByVal nShowCmd As Long) As Long

参数说明：

hwnd 参数→ 可以设置为当前窗口的句柄

lpOperation 参数→ 操作方式，可以是 open print explore

lpFile 参数→ 可以设为指定网址

lpParameters，lpDirectory，nShowCmd 参数→ 均可设为 0。

6. 如何在系统托盘中创建图标

为了使窗体在最小化时在任务栏上不占空间 我们可以将程序以小图标的方式放到系统托盘中 就象 netants 那样），而当用户双击小图标时再弹出窗体，具体的编程方法见于各大电脑杂志，其主要原理与本文类似 在这里我就使用 SysTray 控件 该控件可以通过编译 VB 光盘上 hTools hvb hUNSUPPRT h Systray 目录下的 Systray.vbp 工程文件得到）。

实例：

以上我们提出了实现网页一键通的原理和主要技术细节在下面我们将给出一个简单的能实现该功能的应用程序．为简单其见，在下面的程序中，我们仅定义了一组系统热键，而且为了避免与其它程序的系统级热键发生冲突，我们仅选 F9、F10、F11、F12 这四个非常用键作为待选热键。

程序的运行界面如图所示

部分控件的属性如下

Form1.MaxButton = False
Combo1.Text = ″F9″
Text1.Text = ″http://″
Check1.Caption = ″Shift″
Check2.Caption = ″Ctrl″
Check3.Caption = ″Alt″
Command1.Caption = ″确定″
Command2.Caption = ″最小化″
Command3.Caption = ″关闭″

在程序的当前目录下，需要新建 operate hoperate.ini 文件，其内容为：

[set1]
shutcut = F9 ′定义热键
url = http://www.scu.edu.cn ′定义默认连接网址
keys = 0 ′定义辅助键

下面是源程序:

′以下为模块 Module1.bas 部分

Public Declare Function ShellExecute Lib ″shell32.dll″ Alias ″ShellExecuteA″(ByVal hwnd As Long, ByVal lpOperation As String, ByVal lpFile As String, ByVal lpParameters As String, ByVal lpDirectory As String, ByVal nShowCmd As Long) As Long
Public Declare Function SetWindowLong Lib ″user32″ Alias ″SetWindowLongA″(ByVal hwnd As Long, ByVal nIndex As Long, ByVal dwNewLong As Long) As Long
Public Declare Function GetWindowLong Lib ″user32″ Alias ″GetWindowLongA″(ByVal hwnd As Long, ByVal nIndex As Long) As Long
Public Declare Function CallWindowProc Lib ″user32″ Alias ″CallWindowProcA″(ByVal lpPrevWndFunc As Long, ByVal hwnd As Long, ByVal Msg As Long, ByVal wParam As Long, ByVal lParam As Long) As Long
Public Declare Function RegisterHotKey Lib ″user32″(ByVal hwnd As Long, ByVal id As Long, ByVal fsModifiers As Long, ByVal vk As Long) As Long
Public Declare Function UnregisterHotKey Lib ″user32″(ByVal hwnd As Long, ByVal id As Long) As Long
Public Declare Function GetPrivateProfileString Lib ″kernel32″ Alias ″GetPrivateProfileStringA″(ByVal lpApplicationName As String, ByVal lpKeyName As Any, ByVal lpDefault As String, ByVal lpReturnedString As String, ByVal nSize As Long, ByVal lpFileName As String) As Long
Public Declare Function WritePrivateProfileString Lib ″kernel32″ Alias ″WritePrivateProfileStringA″(ByVal lpApplicationName As String, ByVal lpKeyName As Any, ByVal lpString As Any, ByVal lpFileName As String) As Long
′定义常数
Public Const WM_HOTKEY = &H312
Public Const MOD_ALT = &H1
Public Const MOD_CONTROL = &H2
Public Const MOD_SHIFT = &H4
Public Const GWL_WNDPROC = (-4)
Public preWinProc As Long
Public modifiers As Long, uVirtKey As Long, idHotKey As Long
Private Type Stru1
  p As Long
End Type
Private Type Stru2
  lWord As Integer
  hword As Integer
End Type
Public Function wndproc(ByVal hwnd As Long, ByVal Msg As Long, ByVal wParam As Long, ByVal lParam As Long) As Long

```
Dim ret As Long
If Msg = WM_HOTKEY Then
If wParam = idHotKey Then ´判断消息是否为要处理的消息
    Dim lp As Stru1, s As Stru2
    lp.p = lParam
    LSet s = lp
If (s.lWord = modifiers) And s.hword = uVirtKey Then
    ´连接到预设定的网址
ret = ShellExecute(hwnd, "open", Form1.Text1.Text, 0, 0, 0)
    End If
  End If
End If
´将消息重新交给默认消息处理过程处理
  wndproc = CallWindowProc(preWinProc, hwnd, Msg, wParam, lParam)
End Function
Public Function cut()
Dim keys As String
Dim buff As String
buff = String(255, 0)
ret = GetPrivateProfileString( " set1", " keys", "", buff, 256, "operate/operate.ini")
keys = buff
´将数字转化为对应的各辅助键状态
Select Case keys
Case 0
Check1.Value = 0
Check2.Value = 0
Check3.Value = 0
Case 1
Check1.Value = 1
Check2.Value = 0
Check3.Value = 0
Case 2
Check1.Value = 0
Check2.Value = 1
Check3.Value = 0
Case 3
Check1.Value = 1
Check2.Value = 1
Check3.Value = 0
Case 4
Check1.Value = 0
Check2.Value = 0
Check3.Value = 1
Case 5
Check1.Value = 1
Check2.Value = 0
Check3.Value = 1
Case 6
Check1.Value = 0
Check2.Value = 1
Check3.Value = 1
Case 7
Check1.Value = 1
Check2.Value = 1
Check3.Value = 1
End Select
End Function
Public Function duqu()
´从 ini 文件中读取用户设置
Dim ret As Long
Dim buff1 As String
Dim buff2 As String
buff1 = String(255, 0)
ret = GetPrivateProfileString("set1", "shutcut", "", buff1, 256, "operate/operate.ini")
Form1.Combo1.Text = buff1
buff2 = String(255, 0)
ret = GetPrivateProfileString("set1", "url", "", buff2, 256, "operate/operate.ini")
Form1.Text1.Text = buff2
End Function
´以下是主程序部分:
Public Sub Form_Load()
Combo1.AddItem "F9"
Combo1.AddItem "F10"
Combo1.AddItem "F11"
Combo1.AddItem "F12"
Call duqu
Call cut
Dim ret As Long
´记录窗口消息处理过程的地址
preWinProc = GetWindowLong(Me.hwnd, GWL_WNDPROC)
´将 wndproc 定义为默认消息处理过程
  ret = SetWindowLong(Me.hwnd, GWL_WNDPROC, AddressOf wndproc)
 idHotKey = 1
´读取用户所设置的热键
If Combo1.Text = "F9" Then
uVirtKey = vbKeyF9
End If
If Combo1.Text = "F10" Then
uVirtKey = vbKeyF10
End If
If Combo1.Text = "F11" Then
uVirtKey = vbKeyF11
End If
If Combo1.Text = "F12" Then
uVirtKey = vbKeyF12
End If
´读取用户所设置的辅助键
Dim a, b, c, d As Integer
If Check1.Value = 0 Then
   a = 0
   Else: a = 1
End If
If Check2.Value = 0 Then
    b = 0
    Else: b = 2
    End If
If Check3.Value = 0 Then
```

```
  c = 0
  Else: c = 4
 End If
d = a + b + c
Select Case d
Case 0
modifiers = 0
Case 1
modifiers = MOD_SHIFT
Case 2
 modifiers = MOD_CONTROL
Case 3
 modifiers = MOD_CONTROL + MOD_SHIFT
Case 4
modifiers = MOD_ALT
Case 5
 modifiers = MOD_SHIFT + MOD_ALT
Case 6
 modifiers = MOD_CONTROL + MOD_ALT
Case 7
 modifiers = MOD_SHIFT + MOD_ALT + MOD_CONTROL
End Select
' 定义热键
 ret = RegisterHotKey(Me.hwnd, idHotKey, modifiers, uVirtKey)
End Sub
Private Sub Form_Resize()
If Form1.WindowState = 1 Then ' 窗体最小化时调用Command2_Click()
Call Command2_Click
End If
End Sub
Private Sub Form_Unload(Cancel As Integer)
Dim ret As Long
' 将消息交给默认消息处理过程处理,注消热键
ret = SetWindowLong(Me.hwnd, GWL_WNDPROC, preWinProc)
 Call UnregisterHotKey(Me.hwnd, uVirtKey)
 End Sub
Public Sub Command1_Click()
Dim ret As Long
Dim key As Integer
Dim keys As String
' 读取用户当前设置并将其保存到 ini 文件中
If Check1.Value = 0 Then
  a = 0
  Else: a = 1
End If
If Check2.Value = 0 Then
   b = 0
   Else: b = 2
   End If
If Check3.Value = 0 Then
  c = 0
  Else: c = 4
 End If
 key = a + b + c
keys = CStr(key)
ret = WritePrivateProfileString(″set1″, ″shutcut″, Combo1.Text, ″operate/operate.ini″)
ret = WritePrivateProfileString(″set1″, ″url″, Text1.Text, ″operate/operate.ini″)
ret = WritePrivateProfileString(″set1″, ″keys″, keys, ″operate/operate.ini″)
MsgBox ″程序需要通过关闭来更新设置,请关闭后重新运行!″
Call Command2_Click
End Sub
Private Sub Command2_Click()
cSysTray1.InTray = True ' 图标在托盘中可见
Form1.Visible = False ' 窗体不可见(在 taskbar 中消失)
Form1.WindowState = 1
End Sub
Private Sub Command3_Click()
Dim ret As Long
' 将消息交给默认消息处理过程处理,注消热键
ret = SetWindowLong(Me.hwnd, GWL_WNDPROC, preWinProc)
 Call UnregisterHotKey(Me.hwnd, uVirtKey)
 End
End Sub
Private Sub cSysTray1_MouseDblClick(Button As Integer, id As Long)
' 当双击图标时引发该事件
cSysTray1.InTray = False
Form1.WindowState = 0
Form1.Visible = True
End Sub
```

六、结语

绿色软件是当前自由软件的主潮流 在本程序中综合运用了 ini 文件的操作技术和 Windows 特定消息处理技术 既实现了网页一键通的目的，又符合了绿色软件的要求．由于通过软件实现网页一键通的功能屏蔽了用硬件实现该功能的缺陷 如自主性差 因此应用文中提出的原理和技术开发出的应用软件具有很大的应用价值．

参考文献

1. 《WIN32 程序员参考大全 一 ——窗口管理和图形设备接口》 Microsoft Corporation 著．清华大学出版社

2. 《WIN32 程序员参考大全 三 ——函数 A～G》 Microsoft Corporation 著．清华大学出版社

3. 《WIN32 程序员参考大全 四 ——函数 H～Z》 Microsoft Corporation 著．清华大学出版社

4. 《中文 Visual Basic5.0 程序员指南》 Microsoft Corporation 著．微软[中国]有限公司译．科学出版社，龙门书局

收稿日期：2000 年 10 月 23 日）

利用 Java 的多线程技术实现并行多任务的管理

杨绍方

多线程是编程社会中一个相当新的结构，它非常强大，可以提高程序的运行效率。Java 虚拟机允许一个应用程序同时运行多个线程，Java 编程环境和 Runtime 库最关键的一个特征就是多线程结构，并且 Java 是第一个在语言的核心中支持线程的编程语言。利用多线程编程技术，可以在 Java 中方便地实现任务的并行处理。限于篇幅，本文对于 Java 中与线程有关的类和接口的构造器和方法没有深入地讨论，演示程序中所有代码都有详尽的注释。

一、线程（Thread）基础

所谓线程，就是进程中一个简单的运行流。而进程是一个在自己的地址空间中执行的程序，线程是一个代码序列，它运行在一个进程的上下文中。实际上，线程不能在自己的地址空间中执行，它们要求一个父进程先运行。对于我们所提及的、在机器上运行的每一个进程，无疑是由一些正在运行的线程组成的，不管怎样，线程总是与一个特定的进程相联系。

每一个线程都有一个优先级，在 Java 中，缺省情况下线程的优先级为 5（即：NORM_PRIORITY），最高的优先级为 10（即：MAX_PRIORITY），最低的优先级为 1（即：MIN_PRIORITY）。优先级高的线程先执行，优先级低的线程后执行，线程可以设置为守护线程（daemon），当线程中运行的代码创建一个新的线程对象时，这个新线程拥有与创建它的线程一样的优先级。

二、线程的状态

线程的行为完全依赖于线程所处的状态，线程的状态有下面四种。

1. New

当线程被创建并还未调用 start 方法时，线程处于“new”状态。

2. Runnable

对于新创建的线程，调用 start 方法之后，会自动调用 run 方法，线程进入“funnable”状态。我们可以认为这时线程处于运行状态，因为 run 方法的执行意味着线程正在运行。但是，我们应该考虑到线程优先级，对于 CPU 而言，某一时刻只能有一个线程在运行，尽管这样，我们仍然认为是处于“funnable”状态，因为所有的线程都在共享系统资源。

3. Not Running

由于某些原因，有的线程会被临时暂停，则进入“not running”状态。处于这种状态的线程，对于用户而言仍然有效，仍然可以重新进入“funnable”状态。在 Java 中，下面几种事件会造成线程临时被暂停：

* 调用 sleep 方法；
* 调用 wait 方法；
* 线程由于 I/O 而阻塞（block）。

下面这些事件，会让处于“not running”状态线程重新进入“funnable”状态：

* 如果线程睡眠，sleep 方法所指定的时间已过去；
* 如果线程处于等待，拥有条件变量的对象调用了 notify 或 notifyAll 方法；
* 如果线程由于 I/O 而阻塞，该 I/O 操作已完成。

4. Dead

如果线程不再需要则进入“dead”状态，死亡的线程不能再被恢复和执行。线程进入“dead”状态可以有下面两种方法：

* 调用 exit 方法，并且安全管理器允许执行 exit 操作。
* 所有线程都不是守护线程并已死亡，或者从调用的 run 方法返回，或者在 run 方法中抛掷了异常。

三、与线程有关的类

Java 编程语言通过一个接口和少数几个类提供对线程的支持。这些接口和类分别是 Runnable、Thread、ThreadDeath、ThreadGroup 和 Object。所有这些类都是 java.lang 包的一部分。

1. Runnable 接口

Java 不直接支持多重继承，即从多个父类派生出一个类。如果一个类已经从其它类派生而来，那么，可以使用 Runnable 接口使其支持线程。

2. Thread 类

Thread 类是负责向其它类提供线程功能的最主要的类，为了向一个类增加线程功能，我们可以简单地从 Thread 类派生一个类，并重载 run 方法。run 方法是线程发生的地方，它常常被称为线程体。必须注意，Thread 类中的 stop、suspend 和 resume 方法已不鼓励使用（deprecation），主要原因是这些方法不安全并易于死锁。

3. ThreadDeath 类

ThreadDeath 错误类提供了一种机制，允许我们清理被异步中断的线程，我们称其为错误类是因为它从 Error 类派生而来，提供对错误的报告和处理。

4. ThreadGroup 类

ThreadGroup 类常常用于管理一组线程，一个线程组代表了一组线程，一个线程组还可以包含其它的线程组。线程组仅仅允许本组内的线程访问有关线程组的信息。

5. Object 类

Object 类虽然不是一个严格的线程支持类，但是它提供了三种方法，对 Java 线程结构至关重要。这些方法是 wait、notify 和 notifyAll。wait 方法让线程在睡眠状态中等待，直到通知它继续；同样，notify 方法通知等待的线程继续执行；notifyAll 方法与 notify 相同，但是它通知所有等待的线程。这三个方法仅仅能被 synchronized 方法调用。

四、创建线程

在开发大型的软件项目时，通过将任务分割给不同的线程来并行运行，可以充分利用系统资源，提高程序的运行效率，这也是使用线程的根本目的，因此，我们可以将需要以线程方式执行的功能独立地放入一个 Thread 类的派生类或 run 方法中，并行执行。

在 Java 中，有两种方法创建线程。

1. Extend Thread

如果我们的类不需从别的类派生而来，可以直接从 Thread 类派生而来，例如：

```
public class ThreadMe extends Thread
{
  public run()
  {
    // 需要以线程方式运行的代码
  }
}
```

非常简单，为了在程序中实际地使用和设置线程的行为，我们可以简单地创建一个对象并调用 run 方法，例如：

```
ThreadMe me = new ThreadMe();
me.start();
```

start 方法自动调用 run 方法来执行线程的具体处理。线程将一直运行，直到退出 run 方法。

2. Implement Runnable

如果我们的类需要从其它类而不是 Thread 类派生而来，由于 Java 不允许类的多重继承，这时，只能使用 Runnable 接口。在类中实现 Runnable 接口的方法如下：

```
public class ThreadYou extends Applet implements Runnable
{
  public run()
  {
    // 需要以线程方式运行的代码
  }
}
```

启动线程有两种方法：

```
ThreadYou you = new ThreadYou();
Thread t = new Thread(you);
t.start();
```

或者：

```
ThreadYou you = new ThreadYou();
new Thread(you).start();
```

五、多线程的分组管理

多线程编程对于许多程序员来说是一件头疼的事，特别是当同时运行的线程数量很多时，管理这些线程是一件相当困难的事，在 Java 中，提供了线程组（java.lang.ThreadGroup 类）来管理线程，它使我们同时改变大量线程的运行状态成为可能。

将线程分组有时非常有用，它允许我们将多个线程作为一个单独的实体来控制。在大多数应用程序中，线程根据实际情况可以划分给不同的组。

在 Java 的 Runtime 系统中的每一个线程都属于一个线程组，如果我们创建一个线程时没有指定线程组，那么，该线程被增加到当前线程所属的线程组中，当前线程就是创建新线程的线程。

我们将线程和线程组联系在一起，我们只有在创建线程时指定它属于哪个线程组，之后不能再更改。

在创建线程之前，我们可以创建一个 ThreadGroup 对象。下面的代码创建一个线程组名为 ThreadGroup02 的线程组 group02，并在其中加入由线程类 Transaction 实现的两个线程 transaction01 和 transaction02，这两个线程的线程名分别为 Transaction01 和 Transaction02。

```
ThreadGroup group02 = new ThreadGroup("ThreadGroup02");
Thread transaction01 = new Thread(group02, new Transaction(), "Transaction01");
Thread transaction02 = new Thread(group02, new Transaction(), "Transaction02");
transaction01.start();
transaction02.start();
```

当线程组被创建以后，每一个线程被创建并传递给 ThreadGroup 对象，成为线程组中的成员，每个线程可以各自调用 start 方法启动，ThreadGroup 类并不提供一次启动所有线程的 start 方法。与 Thread 类一样，ThreadGroup 类中的 stop、suspend 和 resume 方法已不鼓励使用（deprecation），因此，对线程组的挂起或运行操作建议通过优先级的调整和让线程睡眠（sleep）来实现。

六、并行处理的概念

并行处理（concurrent processing）含义由下面三方面组成：

1. 同时执行一道以上程序的能力，其效果是充分利用高速的中央处理器；

2. 用一部分硬件交错分时工作，同时处理两个以上的独立任务；

3. 计算机在内存中同时存储两个以上的程序，并分时控制，并行执行这些程序。

对于多处理器的计算机，Java 虚拟机将所有的 CPU 作为一个 CPU 池，这样，当使用 Java 编写的、具有并行处理能力的程序递交给 Java 虚拟机运行时，程序中的线程所面对的不是单个的 CPU，而是 CPU 池中的所有 CPU，显然，这时多任务的并行处理能力会明显提高。

七、实例

MthreadDemo. java 演示如何使用 Java 的多线程技术开发一个多任务的并行处理程序。假设 MthreadDemo. java 运行在一个服务器上，并且不考虑并行处理时数据的同步问题。

在程序中，我们定义了两个线程类，分别代表完成两种不同的任务，假设 Query 类实现对数据库的查询功能，Transaction 类实现某项实时的商业交易。

在实际应用中，经常会出现在同一时刻到达对同一种类型任务的多个请求，例如，会同时有多个查询请求和多个交易请求，为了响应这些请求，必须及时合理地创建多个线程。在程序中，我们假设同时运行有 3 个查询线程和 2 个交易线程，它们同时运行，共享系统资源。

对于处理相同类型任务的线程，可以将它们集中放入到一个线程组中集中管理，这样，一种类型的任务就对应一个线程组，根据任务的重要程度和对响应时间的要求，可以分别设置对应线程组不同的优先级。在程序中，我们将交易类线程放在线程组 group02 中，将查询类的线程放在线程组 group01 中，group02 的优先级高于 group01 的优先级。这样，线程的管理就变得简单多了。

另外，对于同一线程组中的处理同种类型任务的线程，即使在优先级相同的情况下，也不能独占系统资源（例如 CPU），必须轮流执行，共享系统资源。在程序中，我们使用了 sleep 方法让同一线程组中的线程轮流睡眠一个随机的毫秒数来模拟这种情况。

MthreadDemo. java 的完整源程序如下。对于交易类的线程，在 group02 线程组中线程在开始运行的前两次不睡眠，独占一段时间的系统资源。

```
class MThreadDemo
{
 static public void main(String[] args)
 {
  //创建一个新的线程组 group01, 名称为 ThreadGroup01
  ThreadGroup group01 = new ThreadGroup( " ThreadGroup01");
    group01. setMaxPriority(4);
    //创建新线程, 并加入到线程组 group01 中
    Thread query01 = new Thread(group01, new Query(), "Query01");
    Thread query02 = new Thread(group01, new Query(), "Query02");
    Thread query03 = new Thread(group01, new Query(), "Query03");
    //创建一个新的线程组 group02, 名称为 ThreadGroup02
     ThreadGroup group02 = new ThreadGroup( " ThreadGroup02");
    //创建新线程, 并加入到线程组 group02 中
     Thread transaction01 = new Thread(group02, new Transaction(), "Transaction01");
     Thread transaction02 = new Thread(group02, new Transaction(), "Transaction02");
    //向屏幕输出当前所有线程组的信息
    System. out. println("= = = = = = = = = =");
    System. out. println("线程组 group01 中的线程有: " );
    group01. list(); //显示线程组 group01 中的所有线程
    System. out. println("线程组 group02 中的线程有: " );
    group02. list(); //显示线程组 group01 中的所有线程
    System. out. println("= = = = = = = = = =");
    query01. start();    //启动线程
    query02. start();
    query03. start();
    transaction01. start();
    transaction02. start();
 }
}
class Query implements Runnable
{
  public void run()      //在此处放置处理实际任务的代码
  {
        //让线程总共睡眠 3 次
        for(int cnt = 1; cnt < 6; cnt + + )
        {
              try
```

```
            {
                //睡眠一个随机的毫秒数
    Thread.currentThread().sleep((int)(Math.random() * 100));
                }catch(InterruptedException e){}
                //显示当前线程的信息
System.out.println(Thread.currentThread() + "运行,次数是:" + cnt);
        }
    }
}
class Transaction implements Runnable
{
    public void run()//在此处放置处理实际任务的代码
    {
        //让线程总共睡眠3次
        for(int cnt = 1; cnt < 6; cnt++)
        {
            try
                {
                if(cnt>2)
                    //睡眠一个随机的毫秒数
Thread.currentThread().sleep((int)(Math.random() * 100));
                }
                catch(InterruptedException e){}
                //显示特定当前的信息
    System.out.println(Thread.currentThread() + "运行,次数是:" + cnt);
        }
    }
}
```

MThreadDemo.java 使用 JDK 1.3 在 Windows 98/NT 下编译通过，在命令行上运行的命令为：

```
e:\jdk1.3\bin\java MThreadDemo
```

建议读者多运行几次，您会发现每一次运行的结果几乎都不一样，这时由于让线程睡眠一个随机的毫秒数造成的。下面是某一次运行的结果：

```
= = = = = = = = = = = = = = = = = = = = = =
线程组 group01 中的线程有：
java.lang.ThreadGroup[name=ThreadGroup01 maxpri=4]
  Thread[Query01 4 ThreadGroup01]
  Thread[Query02 4 ThreadGroup01]
  Thread[Query03 4 ThreadGroup01]
线程组 group02 中的线程有：
java.lang.ThreadGroup[name=ThreadGroup02 maxpri=10]
  Thread[Transaction01 5 ThreadGroup02]
  Thread[Transaction02 5 ThreadGroup02]
= = = = = = = = = = = = = = = = = = = = = =
Thread[Transaction01 5 ThreadGroup02]运行，次数是：1
Thread[Transaction01 5 ThreadGroup02]运行，次数是：2
Thread[Transaction02 5 ThreadGroup02]运行，次数是：1
Thread[Transaction02 5 ThreadGroup02]运行，次数是：2
Thread[Transaction02 5 ThreadGroup02]运行，次数是：3
Thread[Query03 4 ThreadGroup01]运行，次数是：1
Thread[Query02 4 ThreadGroup01]运行，次数是：1
Thread[Transaction01 5 ThreadGroup02]运行，次数是：3
Thread[Query01 4 ThreadGroup01]运行，次数是：1
Thread[Transaction01 5 ThreadGroup02]运行，次数是：4
Thread[Transaction02 5 ThreadGroup02]运行，次数是：4
Thread[Query02 4 ThreadGroup01]运行，次数是：2
Thread[Query03 4 ThreadGroup01]运行，次数是：2
Thread[Query03 4 ThreadGroup01]运行，次数是：3
Thread[Query03 4 ThreadGroup01]运行，次数是：4
Thread[Query01 4 ThreadGroup01]运行，次数是：2
Thread[Transaction02 5 ThreadGroup02]运行，次数是：5
Thread[Transaction01 5 ThreadGroup02]运行，次数是：5
Thread[Query02 4 ThreadGroup01]运行，次数是：3
Thread[Query02 4 ThreadGroup01]运行，次数是：4
Thread[Query03 4 ThreadGroup01]运行，次数是：5
Thread[Query01 4 ThreadGroup01]运行，次数是：3
Thread[Query02 4 ThreadGroup01]运行，次数是：5
Thread[Query01 4 ThreadGroup01]运行，次数是：4
Thread[Query01 4 ThreadGroup01]运行，次数是：5
```

在输出结果中，线程 Thread[Transaction01 5 ThreadGroup02]的含义为：Transaction01 为当前运行的线程，5 为当前运行线程（Transaction01）的优先级，ThreadGroup02 为当前运行线程（Transaction01）的线程组。其它线程与此相似。

我们可以看到，由于线程组 ThreadGroup01 的优先级为 4，低于优先级为 5 的线程组 ThreadGroup02，因此，虽然 query01、query02 和 query03 三个线程先启动，但是 ThreadGroup02 中的线程 Transaction01 和 Transaction02 先运行，从第 3 次 for 循环开始，程序中让线程睡眠一个随机的毫秒数，这样，在 ThreadGroup02 中的线程睡眠时，ThreadGroup01 中的线程（query01、query02 和 query03）得以运行。一旦 ThreadGroup02 中线程的睡眠时间结束，就马上优先运行，ThreadGroup01 中的线程进入“hot running”状态。

另外，由于线程组 ThreadGroup02 的优先级比 ThreadGroup01 的高，虽然每次运行的结果都不同，但是每一次 ThreadGroup02 中线程总是先运行结束。

（收稿日期：2000 年 10 月 24 日）

纯 Java 网页的编程技巧

曾玉珠

摘　要　本文以网上购物网页为例介绍了如何用 Java 实现网页元素的定位和超链接及数据库的访问等编程技巧。

关键字　布局管理器，事件监听器，JDBC

在网页制作中，人们一般只把 Java 编程应用于网页的交互、动态等部分，至于整个网页如何完全用 Java 来生成，本人感觉也是值得探讨的。

一、布局的实现

网页是由文本、图像及 GUI 组件等组成，定位这些元素是生成网页的第一步。文本及图像可通过坐标定位，在 Java 中 GUI 组件是由布局管理器来定位的。Java 提供 5 个布局管理器，即 FlowLayout（流布局）、GridLayout（网格布局）、GridBagLayout（网格包布局）、BorderLayout（边界布局）及 CardLayout（卡片布局）。网格包布局管理器使用最灵活，它使用单元矩形网格，一个组件可以占据一个或多个单元，而且单元的宽度和高度不需要相同，每个单元由它的坐标来定位。用 Java 制作过网页的朋友可能都曾纳闷：为何组件的位置会偏了？原因是一个单元的大小不是固定的，它跟页面的大小及各行、列中的单元总数有关。为了使组件固定在指定的位置上，可行的方法是使每单元的大小固定即在页面大小不变的情况下，固定各行、列的单元总数，而这可通过在网页四周添加一些不影响网页内容的空标签来实现 例中用方法 addlabel 实现 。下面图一的网页由二个标签、一个按钮、两个图像和一段文本组成。为了定位这些元素，本例在网页四周添加空标签以便把页面划分为 20 行 30 列的网格，若组件的位置更复杂，可把网格进一步细分。

图 1

实现图 1 网页布局程序清单如下：

```
  //类 Class1. java
import  java. net.  * ;  import  java. applet.  * ;  import  java. awt. * ;
public class Class12 extends Applet
{   Image art;   URL codeBase;
   String name[ ] = {″bt33. gif″, ″yz10. gif″};
   int xy[ ] = {70, 15, 90, 100};
   GridBagLayout g = new GridBagLayout( );
    GridBagConstraints c = new GridBagConstraints( );
     Label label1, label2;
     public void init( )
     {int i, j, m = 20, n = 30; codeBase = getCodeBase( ); //取得基地址
setBackground(Color. white); setForeground(Color. blue);
       Font f1 = new Font(″宋体″, Font. BOLD, 12);
    Font f2 = new Font(″宋体″, Font. PLAIN, 13);
      Font f3 = new Font(″宋体″, Font. PLAIN, 14);
      setLayout(g); c. fill = GridBagConstraints. CENTER;
      addlabel(m, n);
      Button bb = new Button(″购　买″);
      location(16, 10, 4, 1, bb); bb. setFont(f2);
      bb. setBackground(Color. yellow);
      bb. setForeground(Color. red);
      label1 = new Label(″选购商品″);
      location(12, 3, 4, 1, label1); label1. setFont(f3);
      label2 = new Label(″其它服务″);
      location(16, 3, 4, 1, label2); label2. setFont(f3);
     };
public void paint(Graphics g)
{int i,  x, y, l = 0;
for(i = 0; i < 2; i + + )
{art = getImage(codeBase, name[i])
  g. drawImage(art, xy[2 * i], xy[2 * i + 1], this);
}
      String ss = ″本产品是信记家具公司最新推出的套装椅. 它是真皮精制而成, 原价为 2000 元人民币, 现九折优惠出售, 欢迎选购! ″;
setFont(f2); g. setColor(Color. orange);
disps(280, 120, 10, 2, ss, 15, g);
}
//在网页的四周添加空标签, 以便把网页划分为 m 行 n 列
public void addlabel(int m, int n)
{ for (i = 0; i < m; i + + )
 for(j = 0; j < n; j + + )
   {Label label = new Label(″ ″);
if(j = = n - 1)c. gridwidth = GridBagConstraints. REMAINDER;
     location(j, i, 1, 1, label);
     if (i> = 1& & i < m - 1)j = j + n - 2;
   }}
//location( )方法用于把组件 b 添加到坐标为(x, y)的单元中, 宽度为 gx, 高度为 gy
  public void location(int x, int y, int gx,  int gy, Component b)
{  c. gridx = x; c. gridy = y; c. gridwidth = gx; c. gridheight = gy;
add(b);       g. setConstraints(b, c);
}
//disps( )方法用于把字符串 s 按每行 n 个字, 首行缩进 k 个字显示在左上角为(x0, y0)的矩形区域中,
   public void disps(int x0, int y0, int n, int l, String s, int k, Graphics g)
    {int x, y, a = 0, c = 0;
for (int j = 0; j < s. length( ); j + + )
```

```
{ x = x0 + l * k; y = y0 + c * k;
g. drawString(s. substring(j, j + 1), x, y);    l + +;
if ((l - a) % n = = 0)
//行首遇到指定的半角符号或全角符号如″, ″和″. ″会自动排版
if (s. charAt(j + 1) ! = ′, ′ & & ! s. substring(j + 1, j + 2). equals
(″. ″)) {c + +; l = 0; a = 0; }
else      a = 1;
  }    } }
```

说明：启动该 Applet 程序时，先用 init 方法初始化网页中的组件，然后执行 paint 方法绘制网页中的图像及文本段。

二、超链接的实现

对于超链接的元素 一般要做两件事，其一，当鼠标指向它时 鼠标变为手形 若是文本的链接 文本的颜色要发生变化；这是个鼠标移动事件，由软接口 MouseMotionListener 处理；其二，当鼠标指向它并按下鼠标时，要实现链接。这是个鼠标事件，由软接口 MouseListener 处理。使用某软接口时，要在类定义中用关键字 implements 加以说明，而且必须在类中实现该软接口中的所有方法。软接口 MouseMotionListener 有两个方法，即拖动鼠标的处理方法 mouseDragged 和移动鼠标的方法 mouseMoved 。软接口 MouseListener 有五个方法，即 mouseEnter 、mouseExied 、mousePressed 、mouseReleased 和 mouseClicked 。

1. 组件的链接：

如：标签 选购商品 、标签 其它服务 、按纽 购买 的链接

要实现图一网页的超链接应在 Class1. java 类中另外加上如下内容：

① 引入处理事件的软件包

```
import java. awt. event. *;
```

②在类定义中加上所使用软接口的说明

```
public class Class1 extends Applet implements MouseMotionListener, MouseListener
```

③添加成员

```
URL theURL;        String url;
Cursor cur = new Cursor(Cursor. HAND_CURSOR); //手形鼠标
    Cursor cur2 = new Cursor(Cursor. DEFAULT_CURSOR);
//默认形的鼠标
```

④在方法 init 中添加如下内容

```
//当鼠标指向组件时光标变为手形
bb. setCursor(cur); label1. setCursor(cur);
label2. setCursor(cur);
    //注册鼠标事件监听器及鼠标移动监听器
  addMouseListener(this); addMouseMotionListener(this);
    //给标签注册鼠标事件监听器及鼠标移动监听器
    label1. addMouseListener(this);
    label1. addMouseMotionListener(this);
    label2. addMouseListener(this);
    label2. addMouseMotionListener(this);
```

⑤实现软接口 MouseListener 的方法

```
    public void mouseEntered(MouseEvent e){ }
    public void mouseExited(MouseEvent e){}
    public void mouseClicked(MouseEvent e){}
    public void mouseReleased(MouseEvent e){}
    public void mousePressed(MouseEvent e)
      { //当鼠标按下的组件为 label1 或 label2 时，链接到 Class2. htm 或 Class4. htm
if (e. getComponent(). equals(label1))  url = ″Class2. htm″;
if (e. getComponent(). equals(label2))  url = ″Class4. htm″;
LinkTo(url);
}
```

⑥实现软接口 MouseMotionListener 的方法

```
public void mouseMoved(MouseEvent e)
{ //当鼠标指向标签 label1 或 label2 时，相应组件的前景色变为红色，否则为蓝色
    x = e. getX(); y = e. getY();
    if (e. getComponent(). equals(label1))
      label1. setForeground(Color. red);;
    if (e. getComponent(). equals(label2))
     label2. setForeground(Color. red)
    if (! label1. contains(x, y))
     label1. setForeground(Color. blue);
    if (! label2. contains(x, y))
     label2. setForeground(Color. blue);
}
public void mouseDragged(MouseEvent e){}
```

⑦处理没注册鼠标监听器的组件按下鼠标事件的另外一种方法

```
public boolean action(Event evt, Object arg)
 {  if (arg = = ″购 买″)      url = ″Class3. htm″;
 LinkTo(url);  return true;
 }
```

⑧实现链接的方法 LinkTo

```
public void LinkTo(String url)
 { try
 {theURL = new URL(getDocumentBase(), url); }
catch(MalformedURLException e) {}
if (theURL! = null)
  getAppletContext(). showDocument(theURL);
}}
```

说明：处理鼠标按下事件可有如下两种方法：①给组件注册鼠标监听器，然后把处理方法加在方法 mousePressed 中。②只给 Applet 注册鼠标监听器，然后把处理方法加在方法 action 中。

2. 图像的链接

图像的链接可通过坐标控制。当鼠标指向图像区域时，鼠标应为手形，按下时实现链接，如图 2 网页中的。

若该图的左上角坐标为 80 5 右下角坐标为 130 55 则处理鼠标移动和鼠标按下事件的程序段见下面的 Class2. java 清单。

3. 文本的链接

独立的文本可用标签组件处理，否则与图像相类似，可用坐标加以控制。不同之处在于指向文本时一般要改变颜色。

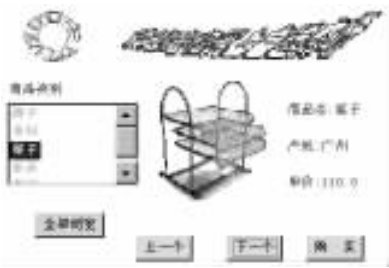


图 2

三、数据库的访问

在 Java 中要对数据库进行操作可采用 JDBC Java Database Connectivity 方法，JDBC 是用于执行 SQL 语句的 Java 应用程序接口。通过 JDBC 与 Java 结合，用户可以很容易地把 SQL 语句传送到任何关系数据库中，程序员用它编写的数据库应用软件，可在各种数据库系统上运行。JDBC 的驱动程序可用 JDBC - ODBC 桥 它可到 http //sun. java. com 下载）等。

编写 JDBC 应用程序一般有下面步骤：

★建立数据源：

我的电脑 →控制面板 →ODBC 数据源 → . . . . . .

★注册数据库驱动器：

Class. forName “Driver” 其中 Driver 为驱动器类

如：Class. forName “com. ms. jdbc. odbc. JdbcOdbcDriver”

★与数据库建立连接：

Connection con;

con = DriverManager. getConnection(String url , String user, String password);

其中 url 的语法：jdbc <子协议> <数据源>

如 ： con = DriverManager. getConnection “jdbc odbc wssc” “zyz” “120562”

★发送 SQL 常用语句：

①创建一个 Statement 对象，它用于执行 SQL 语句，获取执行结果

如：Statement stmt = con. createStatement

②执行 SQL Select 语句，返回包含满足指定条件的记录组成的结果集

如： ResultSet rs = stmt. executeQuery select * from spk

图 2 是个浏览商品的网页，商品的信息已存储在数据库 db2. mdb 的 spk 表中，并建立了与 Access 的数据库 db2. mdb 相连接的数据源 wssc。通过列表框的选项选择不同类别的商品，通过按钮选择商品和购买商品。完整的程序清单如下：

```
//Class2. java
import java. sql. *; import java. awt. event. *;
import java. applet. * ; import java. awt. * ; import java. net. *;
public class Class2 extends Applet implements ItemListener, MouseMotionListener, MouseListener
{URL codeBase, theURL; Image myImage, art1, art2;
 GridBagLayout g = new GridBagLayout();
 GridBagConstraints c = new GridBagConstraints();
 String url2 = "jdbc: odbc: wssc";
 Connection con; ResultSet rs;  Statement stmt, stmt2;
 //确定选择不同类别商品时的查询条件
String query1 = "select * from spk where splb = 'YZ'";
String query2 = "select * from spk where splb = 'BL'";
String query3 = "select * from spk where splb = 'JZ'";
String query4 = "select * from spk where splb = 'JJ'";
String query5 = "select * from spk where splb = 'DH'";
String query0 = "select * from spk ";
String query = "select * from mid ";
String mspbh, url; float mdj;
Cursor cur = new Cursor(Cursor. HAND_CURSOR);
Cursor cur2 = new Cursor(Cursor. DEFAULT_CURSOR);
Button b[] = new Button[4]; List ll;
int i, j, k, x, y, n, m = 1, ch = 0, ch2 = 0, z, kk = 0, bj = 0;
public void init()
{int mm = 20, nn = 30;  codeBase = getCodeBase();
Font f = new Font("TimesRoman", Font. PLAIN, 11);
setBackground(Color. white);  setForeground(Color. blue);
setFont(f); setLayout(g); c. fill = GridBagConstraints. BOTH;
addMouseListener(this); addMouseMotionListener(this);
 art1 = getImage(getDocumentBase(), "ff. gif");
 art2 = getImage(getDocumentBase(), "fy. gif");
 addlabel(mm, nn);
String ss[] = {"全部浏览", "上一个", "下一个", "购 买"};
int xy[] = {3, 9, 2, 1, 12, 11, 3, 1, 17, 11, 3, 1, 22, 11, 3, 1};
for (i = 0; i < = 3; i + + )
  {b[i] = new Button(ss[i]); b[i]. setCursor(cur);
   if (i = = 0)b[i]. setBackground(Color. orange);
   else b[i]. setBackground(Color. green);
   location(xy[4 * i], xy[4 * i + 1], xy[4 * i + 2], xy[4 * i + 3], b[i]);
   b[i]. addMouseListener(this);
 }
Label bb = new Label("商品类别"); location(2, 5, 4, 1, bb);
String name[] = {"椅子", "布料", "架子", "家具", "其它"};
ll = new List(5, false);
for(i = 0; i < 5; i + + ){ll. addItem(name[i]); }
ll. setForeground(Color. pink);  location(2, 6, 6, 1, ll);
ll. addItemListener(this);
try
 { Class. forName("com. ms. jdbc. odbc. JdbcOdbcDriver"); }
catch(java. lang. ClassNotFoundException e){};
try
{con = DriverManager. getConnection(url2, "zyz", "110962"); }
catch(SQLException ex){};
}
public void paint(Graphics g)
{g. drawImage(art1, 80, 5, 50, 50, this);
g. drawImage(art2, 210, 5, 320, 50, this);
try
 {stmt = con. createStatement();
  switch(ch)
    {case 1: rs = stmt. executeQuery(query1); break;
     case 2: rs = stmt. executeQuery(query2); break;
     case 3: rs = stmt. executeQuery(query3); break;
     case 4: rs = stmt. executeQuery(query4); break;
     case 5: rs = stmt. executeQuery(query5); break;
     case 0: rs = stmt. executeQuery(query0);
```

```
    }
//得到确定数据库特性的对象
ResultSetMetaData rsmd = rs.getMetaData();
//得到查询结果的行数
int numberOFColumns = rsmd.getColumnCount();
boolean bb; j = 1;
while ((bb = rs.next()) && (j < m)) j++; // 方法 rs.next()
用于取出下一行的数据
if (bb)
  {z = 1; g.setColor(Color.white);
   g.fillRect(250, 90, 290, 130); //清除显示商品图像及信息
的区域
g.setColor(Color.blue);
for(int i = 1; i <= numberOFColumns; i++)
{String columnName = rsmd.getColumnName(i); //得到第 i
列的字段名
String columnValue = rs.getString(i); //得到当前行第 i 列的
数据
 if (i == 1) mspbh = columnValue; //存储欲选购商品的编号
 if (i == 1||i == 3) continue; //spk 表中的第 1, 3 列的数据
不显示
if (i == numberOFColumns)   //最后一列存放商品的图像
{myImage = getImage(getDocumentBase(), columnValue);
g.drawImage(myImage, 250, 90, 130, 130, this); }
else
{ g.setColor(Color.blue); y = 40 + 30 * i;
if (i == 2) y += 30;
g.drawString(columnName + ": " + columnValue, 420, y);
}
//存储欲选购商品的单价
  if (i == 5) mdj = Float.valueOf(columnValue).floatValue();
 } }
if (!bb && kk == 3){z = 0; m--; }
}
catch(SQLException ex)
{g.drawString( " SQLException: " + ex.getMessage( ), 20,
380); }
}
public void itemStateChanged(ItemEvent event)
{//得到所选商品类别的序号以便采用不同的查询条件
   ch = ll.getSelectedIndex() + 1; m = 1; repaint();
}
public void update(Graphics g)
{//第一次绘画时为整个页面, 重画时区域为 205, 90, 550,
220)
if (bj == 1)      g.clipRect(250, 90, 550, 220);
else         g.clipRect(0, 0, 600, 400);
paint(g);
}
public void mouseEntered(MouseEvent e){ }
public void mouseExited(MouseEvent e){}
public void mouseClicked(MouseEvent e){}
public void mouseReleased(MouseEvent e){}
public void mousePressed(MouseEvent e)
{  if (e.getComponent().equals(b[0]))// 按下"全部浏览"
按钮时
   {m = 1; ch = 0; kk = 1; bj = 1; repaint(); }
   if (e.getComponent().equals(b[1]))// 按下"上一个"按
钮时
     {m = m - 1; kk = 2; bj = 1;
       if (m == 0) m = 1;  repaint();
     }
  if (e.getComponent().equals(b[2]))//按下"下一个"按钮
时
      {m = m + z; kk = 3; bj = 1;  repaint(); }
 if (e.getComponent().equals(b[3]))//按下"购 买"按钮时
  {try
     { ch2 = 1; bj = 1; repaint();
       //确定购买后把该商品的编号及单价存入 mid 表中,
以便其它网页使用
       stmt.executeUpdate("delete from mid");
       stmt.executeUpdate("insert into mid " + "values('" +
mspbh + "', '" + mdj + "')");
       stmt.close();   con.close();
   }
   catch(SQLException ex){};
String url = "Class3.htm"; LinkTo(url);
   }
//当鼠标在图像区域内按下时, 实现链接
x = e.getX(); y = e.getY(); //获取当前鼠标所在的坐标
 if (y>5& & y<55& & x>80& & x<130)
  {  url = "Class1.htm";  LinkTo(url);   }
}
public void mouseMoved(MouseEvent e)
{    //当鼠标指向该图像时, 鼠标变为手形, 否则为默认形
     x = e.getX(); y = e.getY();  setCursor(cur2); k = 0;
     if (y>3& & y<55& & x>80& & x<130)   setCursor(cur);
    }
public void mouseDragged(MouseEvent e){}
}
```

说明：

①本例中用到的方法 addlabel 、location 、LinkTo 均与 Class1.java 中的同名方法相同。

②在本类中增加一个项目事件监听器 ItemListener 用它来处理选择不同类别商品的事件。软接口 ItemListener 只有一个方法，即 itemStateChanged 。

③浏览商品是通过 repaint 方法重画实现的 为避免重画时 网页不必要的闪烁 本使用了 update 方法来限制重画的区域。

④本例用成员 m 的变化来选择不同的商品。

四、结束语

建立大型网站时，可先建立辅助制作网页的软件包把一些常用的类、方法等添加其中，然后引用以免重复编写。

参考文献

1. Arthur Griffith 著．Java 编程大全．电子工业出版社 1998. 12

2. Ashton Hobbs 著．自学 JDBC 数据库编程．清华大学出版社 1998. 8

收稿日期：2000 年 10 月 23 日）

用 OpenGL 和 MFC 创建三维动画

陈文英 马光宇

一、绪言

OpenGL 最初是由 SGI 为其高性能工作站开发的。由于其高性能图形和交互式视景的处理能力，很快得到了 Microsoft、SGI、ATT 公司的 UNIX 软件实验室、DEC、IBM、SUN、HP 等世界著名计算机公司的认可。随着计算机技术的发展，工作站与 PC 机的性能日趋接近。Microsoft 和 SGI 等公司，不失时机地将 OpenGL 在 Windows NT 、Windows95 平台上予以实现。Microsoft 在 Visual C + + 2.0 以上版本中内置了 OpenGL，这更为 OpenGL 在微机上的应用创造了条件，使广大微机用户能够享受 OpenGL 所带来的强大图形功能。

OpenGL 实际上是一种图形与硬件的接口。它包括了 100 多个图形函数，开发者可以利用这些函数方便地实现二维和三维的高级图形技术，在性能上表现得异常优越。它包括建模、变换、光线处理、色彩处理、动画以及更先进的能力，如纹理影射、物体运动模糊等。OpenGL 的这些能力为实现逼真的三维绘制效果、建立交互的三维场景提供了优秀的软件工具。

二、实现技术

OpenGL 是一个与操作系统无关的图形库，它支持复杂的三维着色。在三维图形设计中，OpenGL 完成了诸如加亮、设置阴影、裁剪等等的所有困难工作，所以可以在场景中自由地布置对象而不必关心着色的细节。因为 OpenGL 还可以快速地绘制场景，所以可以通过快速而连续的绘制和显示获得动画效果。

使用 OpenGL 的原语需要以一特定的次序对 OpenGL 的元素进行初始化，还要以一定的次序清理资源。此外，因为 OpenGL 库与操作系统无关，所以它有自己独特的设计，而这种设计和 GDI 模型及多数 MFC 应用程序所建立方法吻合得不好。我们都知道，在 Windows NT 平台下，GDI 是原始窗口的图形接口，能够把数据绘制到屏幕、内存、打印机等。GDI 实现这些是通过一个设备上下文（Device Context 简称 DC）来实现的。它的调用均传送给 DC，由 DC 来实现具体操作。就像 GDI 需要建立设备上下文以绘制图形一样，OpenGL 需要建立着色上下文（Rending Context 简称 RC）来绘制图形。与每个 GDI 调用均需要为其指定一个 DC 不同，OpenGL 引入了当前 RC 的概念，所有的操作都是针对当前 RC 的。RC 不能直接完成绘图，必须与特定的 DC 相联系，在绘图时 OpenGL 将绘图参数传递给 RC，RC 在将其经过一定的变换后传递给它所关联的 DC，完成具体的绘图工作。

本文例示一个简单的单文档界面（SDI）的应用程序，初步探讨 OpenGL for Windows NT 程序设计的方法和特点。该应用程序使用 OpenGL 在一个 Cview 窗口中绘制了一个动画场景。场景中包含了两个对象：一个圆筒和围绕圆筒旋转的球形光源。

三、实现步骤

1. 用 MFC AppWizard exe 创建一个新项目 ANIMATE

选项和默认类名如下（选择单文档选项并重新命名文档和视窗类）：

创建的类：

Application CANIMATEAPP 在 ANIMATE.h 和 ANIMATE.cpp 中

Frame CMainFrame 在 MainFrm.h 和 MainFrm.cpp 中

Document CANIDoc 在 ANIDoc.h 和 ANIDoc.cpp 中

View CANIView 在 ANIView.h 和 ANIView.cpp 中

特征：

+ No toolbar or status bar

+ 3D Controls

+ Use shared DLL implementation

2. 修改项目设置

修改链接器的选项使之包含 OPENGL32.LIB 和 GLU32.LIB，这两个文件是 OpenGL 应用程序所必需的。

3. 用 ClassWizard 给 CANIView 添加消息处理程序

为下述对象添加消息处理程序：

对象 ID	函 数	消 息
CANIView	PreCreateWindow	
CANIView	OnInitialUpdate	
CANIView	OnCreate	WM_CREATE
CANIView	OnDestroy	WM_DESTROY
CANIView	OnSize	WM_SIZE
CANIView	OnTimer	WM_TIMER
CANIView	OnSetCursor	WM_SETCURSOR

4. 编辑文件 ANIView.h

在类声明中增加下述受保护的方法和成员变量：

```
void DrawScene();
void DrawLight();
void PolarRotate(GLdouble dRadius, GLdouble dTwist,
GLdouble dLatitude, GLdouble dLongitude);
HGLRC m_hrc;
GLfloat m_fLatitude, m_fLongitude;
GLdouble m_dRadius;
GLUquadricObj * m_pSphere;
GLUquadricObj * m_pCylinder;
```

在该文件顶部增加两个包含文件：

```
#include <GL/gl.h>
```

```
#include <GL/glu.h>
```

5. 编辑类 CANIView 的构造函数和析构函数

从这一步开始要编辑同一个文件 ANIView.cpp。首先在文件的顶部定义几个常量：

```
#define NEARPLANE 3.0
#define FARPLANE 7.0
#define FOV 40.0
```

定位 ANIView.cpp 中的构造函数，然后像下面那样修改这个构造函数完成对成员变量的初始化：

```
m_pSphere = NULL;
m_pCylinder = NULL;
float fMaxObjectSize = (float)NEARPLANE;
m_dRadius = NEARPLANE + fMaxObjectSize/2.0;
m_fLatitude = 0.0f;
m_fLongitude = 0.0f;
```

有 OpenGL 分配给光源和圆筒的内存要在析构函数中释放，定位析构函数加入下列语句：

```
gluDeleteQuadric(m_pSphere);
gluDeleteQuadric(m_pCylinder);
```

6. 编写 PreCreateWindow 方法的代码

OpenGL 只对具有 WS_CLIPSIBLINGS 和 WS_CLIPCHILDREN 风格位的窗口有效，所以在 PreCreateWindow 方法中加入下面的代码，强行设置这两个风格位。

```
cs.style|=WS_CLIPSIBLINGS|WS_CLIPCHILDREN;
```

7. 编写 OnCreate 方法的代码

我们在 OnCreate 方法中创建一个着色上下文。首先要为 OpenGL 指定一个合适的像素格式，然后创建实际的着色上下文并将它和窗口的设备上下文关联起来。

每个 OpenGL 显示设备都支持一种指定的像素格式号。像素格式含有设备界面的属性，这些属性包括绘图界面是用 RGBA 模式还是颜色表模式，像素缓存是用单缓存还是双缓存，以及颜色位数、深度缓存和模板缓存所用的位数，还有其它一些属性。像素格式一般用一个名为 PIXELFORMATDESCRIPTOR 的结构表示，这个结构包含 26 个属性信息。PIXELFORMATDESCRIPTOR 中每个变量的具体含义和设置可以参考有关资料，本例中 PIXELFORMATDESCRIPTOR 初始化如下：

```
//定义像素格式结构
static PIXELFORMATDESCRIPTOR pfd =
{/
  sizeof(PIXELFORMATDESCRIPTOR),        /  结构尺寸
  1,                                    //结构版本
  PFD_DRAW_TO_WINDOW |
  PFD_SUPPORT_OPENGL |
     PFD_DOUBLEBUFFER,                  //特性标志
  PFD_TYPE_RGBA,                        //RGBA 模式
  24,                                   //24 位颜色
  0, 0, 0, 0, 0, 0,                     //不涉及这些属性
  0, 0, 0, 0, 0, 0, 0,                  //没有 alpha 缓存和累积缓存
  32,                                   //32 位深度缓存
  0, 0,                                 //没有模板缓存和辅助缓存
  PFD_MAIN_PLANE,                       //主层类型
  0,                                    //保留结构数
  0, 0, 0                               //不支持结构数
};
```

在上面定义的像素格式结构中，第三个变量 dwFlags 的值是 PFD_DRAW_TO_WINDOW | PFD_SUPPORT_OPENGL | PFD_DOUBLEBUFFER，表明应用程序使用 OpenGL 函数绘制窗口，且支持双缓冲机制 第四个表明当前采用 RGBA 颜色模式，第五个采用 24 位真彩色；因为没有使用 alpha 缓存和累积缓存，所以从变量 cAlphaBits 到 cAccumAlphaBits 都设置为 0；深度缓存 cDepthBits 设置为 32 位，这个缓存能解决三维场景的消隐问题；变量 cAuxBuffers 设置为 0，在 Windows NT 下不支持辅助缓存；Windows NT 下针对 OpenGL 变量 ilayerType 只能设置为 PFD_MAIN_PLANE，但其它平台也许支持 PFD_OVERLAY 或 PFD_UNDERLAY_PLANE；接下来 bReserved 变量只能设置为 0，而最后三个变量 Windows NT 都不支持，故全设置为 0。

下面的代码用于创建实际的着色上下文并将它和窗口的设备上下文关联起来。

```
CClientDC clientDC(this); //得到一个应用窗口客户区的设备描述表
int pixelFormat = ChoosePixelFormat(clientDC.m_hDC, &pfd); //选择一个像素格式
BOOL success = SetPixelFormat(clientDC.m_hDC, pixelFormat, &pfd); //设置像素格式
DescribePixelFormat(clientDC.m_hDC, pixelFormat, sizeof(pfd), &pfd);
m_hrc = wglCreateContext(clientDC.m_hDC); //建立着色上下文
```

在 OnCreate 方法的最后设置定时器，定时器事件每隔 100ms 触发时都会使客户区以强制发生重画事件。

```
SetTimer(1, 100, NULL);
```

8. 编写 OnDraw 方法的代码

在 OnDraw 方法中主要完成图形的显示，为了增强程序的易读性，显示部分用一个子程序 DrawScene 来实现 下面是加入该方法的代码：

```
wglMakeCurrent(pDC->GetSafeHdc(), m_hrc); //使 RC 有效
DrawScene(); //绘制场景
SwapBuffers(wglGetCurrentDC());
wglMakeCurrent(pDC->GetSafeHdc(), NULL); //使 RC 无效
```

在以上程序中，响应 WM_CREATE 消息的函数 OnCreate 仅仅只创造了一个临时的 RC，当函数返回时，它自动删除。因此，在这里不可能自动启用 RC。当应用程序的窗口内容得到更新后，MFC 就调用 OnDraw 函数，在此启用着色上下文 wglMakeCurrent 有效。然后 OpenGL 调用绘图函数 DrawScene ，则程序就可进行任意的 OpenGL 图形操作了。在本程序中采用了双缓冲机制，在这里使用 SwapBuffers wglGetCurrentDC 语句交换当前可见的显示缓冲区和刚着色过的缓冲区。这种双缓冲机制通过隐藏着色过程提高了性能。当绘图结束后，再调用 wglMakeCurrent ，但第二个参数设为 NULL，意思是关闭着色上下文。

9. 编写 OnInitialUpdate 方法的代码

在 OnInitialUpdate 函数中加入下面的代码，在显示之前建立一个着色上下文，包括配置光源属性、设置背景颜色和创建场景的三维对象。

```
glClearColor(0.0f, 0.0f, 0.0f, 1.0f);
glClearDepth(1.0);
glEnable(GL_DEPTH_TEST);
glEnable(GL_LIGHTING);
GLfloat light_ambient[] = {0.3f, 0.3f, 0.3f, 1.0f};
GLfloat light_diffuse[] = {0.8f, 0.8f, 0.8f, 1.0f};
GLfloat light_specular[] = { 1.0f, 1.0f, 1.0f, 1.0f };
glLightfv(GL_LIGHT0, GL_AMBIENT, light_ambient);
glLightfv(GL_LIGHT0, GL_DIFFUSE, light_diffuse);
glLightfv(GL_LIGHT0, GL_SPECULAR, light_specular);
glLightModelf(GL_LIGHT_MODEL_TWO_SIDE, 1.0f);
glEnable(GL_LIGHT0);
m_pCylinder = gluNewQuadric();
gluQuadricDrawStyle(m_pCylinder, GLU_FILL);
m_pSphere = gluNewQuadric();
gluQuadricDrawStyle(m_pSphere, GLU_FILL);
GetParentFrame( ) ->SetWindowPos(NULL, 0, 0, 200, 200,
SWP_NOMOVE|SWP_NOZORDER|SWP_NOACTIVATE);
```

10. 编写 OnDestroy 方法的代码

在撤消视窗时要删除着色上下文并撤消定时器，在该函数中须加入下面的代码：

```
wglDeleteContext(m_hrc);          //删除着色上下文
KillTimer(1);                     //撤消定时器
```

11. 编写 OnSize 方法的代码

每次改变窗口的大小时要重新配置着色上下文，包括设置新的纵横比和视口尺寸。在 OnSize 函数中加入下面的代码：

```
CClientDC clientDC(this);
wglMakeCurrent(clientDC.m_hDC, m_hrc); //启动着色上下文
GLsizei nWidth = (GLsizei)cx;
GLsizei nHeight = (GLsizei)cy;
GLdouble dAspect = (GLdouble)nWidth/(GLdouble)nHeight;
glMatrixMode(GL_PROJECTION);
glLoadIdentity();                 //初始化为单位矩阵
gluPerspective (FOV, dAspect, NEARPLANE, FARPLANE);
//建立透视投影
glViewport(0, 0, nWidth, nHeight);          //视口变换
glMatrixMode(GL_MODELVIEW);
```

12. 编写 OnTimer 方法的代码

当 100ms 定时器时间到时，简单地将重画场景的客户区置无效。在 OnTimer 函数中加入代码：

```
Invalidate(FALSE);
```

13. 编写 OnSetCursor 方法的代码

如果光标在 CANIView 上，每次重画场景时光标都会闪烁。为此，如果光标在视窗的客户区则关闭光标。在 OnSetCursor 函数中加入代码

```
if(nHitTest = =HTCLIENT)
{
    SetCursor(NULL);
    return TRUE;
}
else
{
  return CView: : OnSetCursor(pWnd, nHitTest, message);
}
```

14. 编写 Drawscene 方法的代码

这是本程序的中心内容。每次窗口需要重画时，都要调用 Drawscene 方法。下面是加入 drawscene 函数的代码：

```
static GLfloat vflGreenAmbient[] = {0.1f, 0.3f, 0.1f, 1.0f};
static GLfloat vflGreenDiffuse[] = {0.0f, 1.0f, 0.0f, 1.0f};
static GLfloat vflBlueAmbient[] = {0.1f, 0.1f, 0.3f, 1.0f};
static GLfloat vflBlueDiffuse[] = {0.0f, 0.0f, 1.0f, 1.0f};
static GLfloat vflBlueSpecular[] = {0.0f, 0.0f, 1.0f, 1.0f};
static GLfloat vflLightPosition[] = {1.0f, 1.0f, 1.0f, 1.0f};
static GLfloat flAngle = 0.0f;
glEnable(GL_DEPTH_TEST);
glClear(GL_COLOR_BUFFER_BIT|GL_DEPTH_BUFFER_BIT);
glPushMatrix();
m_fLatitude + = 4.0f;
m_fLongitude + = 2.5f;
PolarRotate(m_dRadius, 0.0f, m_fLatitude, m_fLongitude);
glPushMatrix();
flAngle + = 6.0f;
glRotatef(flAngle, 1.0f, 0.0f, 1.0f);
glTranslatef(0.0f, 1.5f, 0.0f);
glLightfv(GL_LIGHT0, GL_POSITION, vflLightPosition);
DrawLight();
glPopMatrix();
glPushAttrib(GL_LIGHTING_BIT);
glMaterialfv(GL_BACK, GL_AMBIENT, vflGreenAmbient);
glMaterialfv(GL_BACK, GL_DIFFUSE, vflGreenDiffuse);
glMaterialfv(GL_FRONT, GL_AMBIENT, vflBlueAmbient);
glMaterialfv(GL_FRONT, GL_DIFFUSE, vflBlueDiffuse);
glMaterialfv(GL_FRONT, GL_SPECULAR, vflBlueSpecular);
glMaterialf(GL_FRONT, GL_SHININESS, 50.0f);
glPushMatrix();
glRotatef(30.0f, 1.0f, 0.5f, 1.0f);
gluCylinder(m_pCylinder, 0.3, 0.5, 0.8, 15, 5);
glPopMatrix();
glPopAttrib();
glPopMatrix();
glFinish();
```

15. 编写最后两个方法

这两个方法由 DrawScene 互斥地使用：

```
void CANIView: : PolarRotate(GLdouble dRadius, GLdouble
dTwist, GLdouble dLatitude, GLdouble dLongitude)
{
  glTranslated(0.0f, 0.0f, -dRadius);
  glRotated( -dTwist, 0.0f, 0.0, 1.0f);
  glRotated( -dLatitude, 1.0, 0.0, 0.0);
  glRotated(dLongitude, 0.0, 0.0, 1.0);
}
void CANIView: : DrawLight( )
{
  glPushAttrib(GL_LIGHTING_BIT);
  glDisable(GL_LIGHTING);
  glColor3f(1.0f, 1.0f, 1.0f);
  gluSphere(m_pSphere, 0.1, 10, 10);
  glPopAttrib();
```

(下转第 92 页)

AUTO　CAD 图形的比例尺转换

刘　太

摘　要　本文通过对 AutoCAD 的二次开发，利用 AutoCAD 提供的 ActiveX　Automation 技术，调用 VisualVasic 的资源，给出了一个改变 AutoCAD 图形比例尺的实现方法。

关键词　Visual　Basic，Auto CAD，比例尺转换

一、引言

随着信息技术及 GIS 技术的飞速发展，传统的地形图表达方式和图形缩放速度已经不能满足快节奏地形图使用的需要。尤其是在城市规划设计和工业厂区的规划设计中。为了满足不同层次的需要，同一个地区可能同时需要几套不同比例尺的地形图，这对于传统的手工成图来说，无异于将图纸所包含的地面信息进行二次测绘。

由于上述现象的产生，几年前就有人提出了用计算机来绘制地形图，但由于当时计算机技术的发展水平较低、需要投入的成本又太高，所以用计算机绘制地形图也一直没有形成较大规模的应用；随着近几年来计算机技术的迅猛发展，硬件成本的进一步降低，用计算机来绘制地形图已成了一个不可扭转的发展方向，但随之而来也产生了一些新的问题。

由于应用领域较小，导致计算机绘制地形图的专业成图软件的价格居高不下，因此，许多测绘单位都不约而同地瞄准了使用方便、简单的图形处理软件 Auto　CAD，由于该软件的实用性，加之具有足够的开放性，能够进行二次开发来满足图形绘制的需要，所以该软件在测绘界的应用也在不断的扩大。

我们知道，地形图是按照一定的比例尺进行测绘而得到的，而按照一定比例尺测绘的地形图，其中的符号和文字注记也是固定大小的，这样，当我们在 Auto　CAD 上将图形绘制完成后，即可进行图形的输出。当图形的比例尺需要改变时，可以通过 AutoCAD 中的“Print / Plot　Configuration”对话框中“Scale　Rotation　and　Origin”选项中的“Plotted　MM = Drawing　Units”来方便地达到比例尺变换的目的，但这样的比例尺变换，就如同复印机一样对图形中的所有内容进行缩放，使得图形中符号和注记文字的大小发生变化，在输出的图形中出现不协调的现象。

针对上述现象，本人在分析了地形图的特点以后，利用 VB 开发了一个用上述方法进行比例尺变换时，对图形中的符号和文字进行处理的方法。

二、解决方案

地形图中的内容主要分为三部分，符号、线型和文字注记。而地形图图形中的符号基本上全是用 AutoCAD 中的“块”来实现，因此，基本想法是在 Auto CAD 图形找到这些符号和文字，根据进行变换的比例尺参数对这些对象的大小进行调整，即可得到适当比例尺的新的地形图。

三、应用程序的设计

根据以上思路进行程序的设计，具体步骤如下：

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

工程名为 ChangeScale，工程文件名为 ChangeScale. vbp，窗体名为 FrmMain. frm。程序界面设计见下图：

窗体中的控件名称和属性见下表：

控件名称	属性	内容	类别	备注
Lable1	Caption	程序说明，见上图文字	标签框	
Frame1	Caption	当前图形比例尺	框架控件	
Frame2	Caption	目的比例尺		
OptA i i = 0 - 3	Caption	1　500	单选按钮控件	控件数组
		1　1000		
		1　2000		
		其他		
OptB i i = 0 - 3	Caption	1　500	单选按钮控件	控件数组
		1　1000		
		1　2000		
		其他		
CmdOK	Caption	确　定	命令按钮	
CmdCancel	Caption	取　消		
TxtA	text	TxtA	文本框	源比例尺输入
TxtB	text	TxtB		目的比例尺输入

然后将下述的代码加入相应控件的相应事件中，编译成可执行文件 ChangeScale. EXE。

四、程序的使用

将上述可执行文件 ChangeScale. exe 拷贝至 AutoCAD 支持的查找目录（如：C:\AutoCAD\Support）中，修改 AutoCAD 安装目录中的 Support 子目录下的 ACAD. PGP 文件，在该文件中的外部命令定义中增加下面一行：

ChangeScale, ChangeScale, 5, 图形比例尺变换! 按任意键继续!

启动 AutoCAD 后（如果 AutoCAD 已经启动，可以用 REINIT 命令重新初始化 AutoCAD 的 ACAD. PGP 文件），在命令行敲入 ChangeScale 后回车，提示‘图形比例尺变换! 按任意键继续!’，然后选择您的源比例尺和目的比例尺，鼠标单击‘确定’按钮，等到程序提示‘比例尺转换完毕！’，即可完成图形中的符号和文字的处理。

以上程序，虽然比较笨拙，但比逐个去处理图形中的文字和符号还是要省去不少的工夫，当我们需要改变某图形的比例尺时，只需在程序的执行开始后，倒上一杯茶水，小憩片刻，图形的处理也就大功告成了，接下来我们只需将图形的比例尺注记改为实际比例尺，即可进行新图形的输出了。

五、说明

这种方法，只是处理地形图比例尺的一种简单方法，可以满足一般的需要，对于特殊要求的比例尺变换，还需要将变换方法进行完善，希望此法能够对测绘界的同行们有所帮助，也能给在 AutoCAD 上进行二次开发的其他计算机爱好者以启发。

本程序在 VB5. 0 企业版、Windows98 中文版、AutoCAD R14 英文版和 VB6 英文版、Windows98SE 中文版、AutoCAD R14 英文版、AutoCAD2000 英文版下调试通过。

附程序源代码：

MdlTest. bas

```
Option Explicit
'本函数用来检查文本框的输入是否为数字
'调用参数为: 提示标题、提示字符串、需检查的内容
Public Function blnCheckInput(strTitle As String, strPrompt As String, strText) As Boolean
    Dim strTmp As String          '定义临时字符串
    Dim intI As Integer           '定义循环变量
    If strText = "" Then          '如果字符串为空, 则提示
        MsgBox strPrompt & "不能为空!", vbOKOnly + vbExclamation, strTitle
        blnCheckInput = False
        Exit Function
    Else '如果字符串不为空, 则检查其中是否含有非数字内容
        For intI = 1 To Len(strText)
            strTmp = Mid(strText, intI)
            If (Asc(strTmp) <> 46) And (Asc(strTmp) < 48 Or Asc(strTmp) > 57) Then
                MsgBox strPrompt & "不能含有字符!", vbOKOnly + vbExclamation, strTitle
                blnCheckInput = False
                Exit Function
            End If
        Next intI
    End If
    strTmp = Left(strText, 1) '如果检查的内容中没有非法字符
    If (strTmp = ".") And (Len(strText) = 1) Then '则判断字符串的第一位是否为小数点
        MsgBox strPrompt & "的数值不能只有小数点!", vbOKOnly + vbExclamation, strTitle
        blnCheckInput = False
        Exit Function
    End If
    blnCheckInput = True '检查内容合法, 则返回检查标志: TRUE
End Function
```

窗体中的代码

```
Option Explicit
Dim Acadapp As Object     '定义 AutoCAD 对象
Dim AcadDoc As Object     '定义 AutoCAD 的当前图形
Dim MoSpace As Object     '定义 AutoCAD 的模型空间对象
Dim intA As Integer       '定义源比例尺选择标志
Dim intB As Integer       '定义目的比例尺选择标志
Private Sub cmdCancel_Click()
    End
End Sub
Private Sub cmdOK_Click()
    Dim aItem As Object
    Dim i As Long                 '循环变量
    Dim lngMosObject As Long      '图形中实体的数量
    Dim InsPt As Variant          '定义文字插入点
    Dim dblTextH As Double        '定义图形中原文字的高度
    Dim dblScaleA As Double       '定义原图形比例尺
    Dim dblScaleB As Double       '定义目的图形比例尺
    Dim strScaleA, strScaleB As String '定义输入比例尺字符串
    Dim intFlag As Integer        '定义给定的比例尺是否为 10 的倍数标志
    Dim blnScaleFlag As Boolean   '定义检查比例尺输入检查标志
    Dim strPrompt As String       '比例尺输入检查提示
    Dim strTitle As String        '检查提示的标题
    Dim dblBlockXscale As Double  '定义块的 X 比例
    Dim dblBlockYscale As Double  '定义块的 Y 比例
    Dim InsBlockPt As Variant     '定义块的插入点
    Select Case intA              '源比例尺的选择
        Case 1                    '源比例尺为 500
            dblScaleA = 500#
            Select Case intB
                Case 1
                    MsgBox "您选择的源比例尺和目的比例尺相同, 不必转换!", vbInformation + vbOKOnly, "比例尺转换"
                    End
                Case 2
                    dblScaleB = 1000#
                Case 3
                    dblScaleB = 2000#
```

```
            Case 4
                strScaleB = TxtB.Text
                strPrompt = "目的比例尺"
                strTitle = "目的比例尺"
blnScaleFlag = blnCheckInput(strTitle, strPrompt, strScaleB)
                If Not blnScaleFlag Then
                    TxtB.SetFocus
                    Exit Sub
                End If
                dblScaleB = CDbl(strScaleB)
        End Select
    Case 2          '源比例尺为 1000
        dblScaleA = 1000#
        Select Case intB
            Case 1
                dblScaleB = 500#
            Case 2
                MsgBox "您选择的源比例尺和目的比例尺相同,不必转换!", vbInformation + vbOKOnly, "比例尺转换"
                End
            Case 3
                dblScaleB = 2000#
            Case 4
                strScaleB = TxtB.Text
                strPrompt = "目的比例尺"
                strTitle = "目的比例尺"
blnScaleFlag = blnCheckInput(strTitle, strPrompt, strScaleB)
                If Not blnScaleFlag Then
                    TxtB.SetFocus
                    Exit Sub
                End If
                dblScaleB = CDbl(strScaleB)
        End Select
    Case 3          '源比例尺为 2000
        dblScaleA = 2000#
        Select Case intB
            Case 1
                dblScaleB = 500#
            Case 2
                dblScaleB = 1000#
            Case 3
                MsgBox "您选择的源比例尺和目的比例尺相同,不必转换!", vbInformation + vbOKOnly, "比例尺转换"
                End
            Case 4
                strScaleB = TxtB.Text
                strPrompt = "目的比例尺"
                strTitle = "比例尺转换"
blnScaleFlag = blnCheckInput(strTitle, strPrompt, strScaleB)
                If Not blnScaleFlag Then
                    TxtB.SetFocus
                    Exit Sub
                End If
                dblScaleB = CDbl(strScaleB)
        End Select
    Case 4          '源比例尺为自定义
        strScaleA = TxtA.Text
        strPrompt = "源比例尺"
        strTitle = "比例尺转换"
blnScaleFlag = blnCheckInput(strTitle, strPrompt, strScaleA)
        If Not blnScaleFlag Then
            TxtA.SetFocus
            Exit Sub
        End If
        dblScaleA = CDbl(strScaleA)
        Select Case intB
            Case 1
                dblScaleB = 500#
            Case 2
                dblScaleB = 1000#
            Case 3
                dblScaleB = 2000#
            Case 4
                strScaleB = TxtB.Text
                strPrompt = "目的比例尺"
                strTitle = "目的比例尺"
blnScaleFlag = blnCheckInput(strTitle, strPrompt, strScaleB)
                If Not blnScaleFlag Then
                    TxtB.SetFocus
                    Exit Sub
                End If
                dblScaleB = CDbl(strScaleB)
        End Select
    End Select
    '判断自定义比例尺是否为 10 的倍数
    If (Fix(dblScaleA / 10#) <> (dblScaleA / 10#)) Or (Fix(dblScaleB / 10#) <> dblScaleB / 10#) Then
        intFlag = MsgBox("您给出的比例尺不是 10 的倍数,是否继续?", vbYesNo + vbExclamation, "比例尺转换")
        If intFlag = vbNo Then
            Exit Sub
        End If
    End If
    'frmMian.Hide
    '当转换的比例尺合乎要求后,进行转换
    '因该程序是直接在 Auto CAD 下使用的,所以 CAD 当然是启动的
    '因此不必进行 CAD 是否已经启动的判断
    Set Acadapp = GetObject(, "AutoCAD.application")
    Set AcadDoc = Acadapp.ActiveDocument
    Set MoSpace = AcadDoc.ModelSpace
    '得到当前图形中的实体数量
    lngMosObject = MoSpace.Count - 1
    For i = 0 To lngMosObject
        Set aItem = MoSpace.Item(i)
        If aItem.EntityType = acText Or aItem.EntityType = acMtext Then        '查询文字
            InsPt = aItem.insertionPoint    '得到文字的插入点
            dblTextH = aItem.Height         '得到文字的高度
            dblTextH = dblTextH * (dblScaleB / dblScaleA)
'对文字高度进行转换
            aItem.Height = dblTextH     '使文字具有新的高度
            aItem.insertionPoint = InsPt'保持文字的插入点不变
```

(下转第 92 页)

用 IC 卡在 NT 网中实现附加身份认证

许 勇

一、引言

随着 Internet 的迅速发展，网络已成为一个国家的政治、经济、军事和教育的重要资源，各部门、团体甚至个人相继建立起了自己的网站和内部网络（Intranet），同时，网络的安全性也成为了一个日益严重的问题，网络攻击时有发生，许多重要网站都有被攻击和入侵的记录。在众多的攻击目标和攻击方法中，攻击者千方百计最想得到的是服务器系统管理员（NT 中的 administrator UNIX 中的 root）的帐户名和口令，从而冒充系统管理员，完全得到网络的控制权，从事他不应该从事的活动。

在一般情况下，对于一个网络用户是否具有系统管理员身份这个问题，是靠他所知道的管理员的帐户名和口令与服务器上的口令文件中的记录是否一致来进行验证的，但只要口令的输入、验证过程以及口令文件没有和攻击者在物理上隔离，帐户名和口令是可以通过多种方法得到的，如强行攻击进行猜测，用特洛伊木马 Trojan，网络监听（Sniff）技术进行盗窃等。

为改善基于口令的身份验证所带来的不安全性，通常采用一些其它方法来进行辅助识别，如一次性口令、第三方认证、生物特征鉴别、管理员是否拥有某件不能伪造的物品等。这些方法可以使系统达到较高的安全性，但实现起来都很复杂和昂贵，需要专门的设备和系统，这里介绍的方法可操作性强，但不失为提高安全性的一种有效方法。

IC 卡的全称是 Integrated Circuit Card 集成电路卡，它具有国际标准化（Standard）、灵巧智能化（Smart）、安全性（Security）以及价格低廉的特点，目前在银行、邮电、交通、保安甚至军事等领域广泛使用。IC 卡具有很高的加密强度，安全性好，成功地非法复制和改写需要付出很高的代价，因而广泛地被用来作为身份验证的物品。IC 卡有多种规格，有的配合相应的设备专门用作身份验证，这里综合价格和性能上的考虑，从一般用户都可以实现的角度出发，采用常用的逻辑加密串行传输存储卡，这种存储卡一般是电可擦除（EEPROM）的，可通过读卡机和 PC 机的串口或 USB 接口）进行通讯。

二、基本原理

逻辑加密型存储卡 Smart Card with Security Logic 主要由 EEPROM 存储单元阵列和密码控制逻辑单元组成，存储器分为主存储器、保护存储器和加密存储器，加密存储器的第 1、2、3 字节为“参照字”存储区，其内容被称为可编程加密代码（PSC），在对这种 IC 卡进行读写操作时，需输入一个 3 字节长的“效验字”与 PSC 进行比较，只有在比较成功时方可对加密存储器的第 1、2、3 字节进行读写操作和对其它存储区进行改写操作，能否读出正确的“参照字”内容和是否能对存储区改写可作为鉴别 IC 卡的手段。3 个字节可组成 16777216 种代码，并且每张 IC 卡一般只允许 3 次比较失败，在不知道“效验字”的情况下复制某 IC 是不可能的，因而用具有某“效验字”的 IC 卡作为一种身份的代表具有很高的安全性。IC 卡本身是半导体电子元件，输入和输出的都是电信号，将电平适当转变，控制好时序后可直接与微机进行通讯。

身份验证过程如下：

1. 在 NT 服务器上打开安全日志（Unix 上也有类似日志），并启动一守护进程，监视网络中管理员的登录事件，当发现有用户以管理员身份登录后，即要求登录的工作站对用户是否拥有所要求的含有特殊“参照字”的 IC 卡作认证，如果收不到认证信息则断开与该工作站的连接；

2. 用户在工作站上登录后，即运行验证程序，检查本机的串口上是否有所要求的 IC 卡存在，如验证成功，则向服务器发认证成功的消息。

这种方法可靠性很高。首先来看“参照字”的安全性。IC 卡上的“参照字”是 IC 卡和工作站上的验证程序共同“知道”的密钥，如果不知道“参照字”的具体内容，加密存储器的“参照字”是不可读写的，从 IC 卡上获取密钥和复制 IC 卡都不可能；通过对工作站上的验证程序反编译后有可能泄露密钥的内容，但反编译不是一件容易的事，而且，攻击者很难从网上获得程序。其次来看认证过程的安全性，认证过程由验证程序对 IC 卡的鉴别和守护进程对验证程序的鉴别两部份组成。验证程序读加密存储器的内容或改写存储器时，显然只有对管理员所拥有的 IC 卡进行操作时才可能成功。在守护进程和验证进程的通讯过程中，虽然传输信道被认为是不安全的，但现有多种基于密码学的协议实现一个进程通过不安全网络对另一进程进行鉴别（Authentication），例如查问-应答（Challenge-response）协议、密钥交换协议（Diffie-Hellman key exchange）、KDS 鉴别协议等，这些协议在所应

用领域都是非常安全的，这里可酌情选择一种。

三、技术规范

1. 引脚功能：接触式串行接口 IC 卡共有 6 - 8 个触点（C1 到 C8），分别接到集成电路芯片的 8 个引脚上，主要厂商是德国 SIEMENS 公司和美国 ATMEL 公司，以西门子 SLE442 为例，各触点的功能定义如下，保留使用的引脚因公司和型号的不同而有不同的用途。

触点编号	功能	触点编号	功能
C1	VCC（电源）	C5	GND（地）
C2	RST（复位信号）	C6	保留使用
C3	CLK（时钟）	C7	I/O（数据输入/输出端）
C4	保留使用	C8	保留使用

2. 电气特性：这种 IC 卡自带内部电压提升电路，采用 5V 单一电源供电，最大电流约 200mA，可由 PC 机提供，其它信号高电平为 5V，也可由串口驱动。

3. 接口：IC 卡的外观形状有多种，配备不同的读写设备，这里推荐使用通用的内置式 ID - 1 型读写器，这种读写器符合 ISO7816 国际标准，外观和 3.5 英寸磁盘驱动器一致，与 PC 机串口相连，同时也可外置使用，IC 卡尺寸 mm 为 85.6 × 53.98 × 0.76，不需额外供电。各种读写设备一般都配有在各种操作系统 如 DOS、Windows、UNIX 下的软件开发环境和 C、FOXPRO、FOXBASE 等下的接口函数。有兴趣的读者还可自制读写设备，象串口上的 DB - 9 连接器，IC 卡芯片的接触器件等在市面上皆有出售，汇编编程时需调用 ROM BIOS 的 INT 14 对串口进行读写，在 Windows API 和 UNIX 环境下需打开相应的设备文件，如 COM、 /etc/getty 等。

4. 指令格式：数据传输采用两线连接协议，按 ISO7816 标准，有复位、命令、输出和处理四种模式。复位可在任何期间进行，芯片复位响应后进入待命状态，根据命令的内容进入数据输出 （读数据）或数据处理模式 （比较或修改数据）。命令含 3 个字节，格式为：

控制码	地址码	数据
$B_7B_6B_5B_4B_3B_2B_1B_0$	$A_7A_6A_5A_4A_3A_2A_1A_0$	$D_7D_6D_5D_4D_3D_2D_1D_0$

传送顺序为 B0 - B7，A0 - A7，D0 - D7，最后附加一个 CLK 信号将 I/O 端置高。

四、编程要点

1. IC 卡的比较操作：

比较操作的流程如图 1 所示，如果是直接对芯片操作，以 SLE4442 芯片为例，命令格式如下：

功能	控制码	地址码	数 据	说　　明
读加密存储器	31H	无	无	读出加密存储区的 4 个字节
比较效验数据	33H	1 - 3	输入数据	处理模式
修改加密存储器	39H	0 - 3	输入数据	处理模式

图 1

但读写器一般带有高层的读写接口和友好的编程环境，具体命令和读写器有关。

2. 守护进程对验证进程的鉴别：

这里仅需要守护进程对验证进程进行单向鉴别，所以可以简化标准的查询 - 应答协议，如图 2 所示。

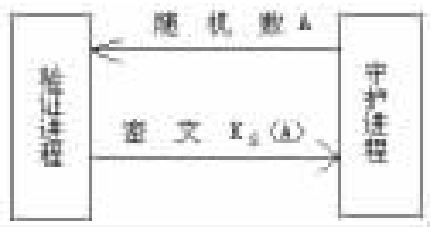


图 2

守护进程先向验证进程发送一个随机数 A，验证进程用一种特殊的方法 KS 对随机数进行转换后将 KS A 传回对方，由于“特殊的方法”KS 是守护进程和验证进程共知的秘密，所以可以确定来自验证进程的消息的真伪，K 指加密算法，S 指密钥。

加密的方法非常多，如 DES 数据加密标准的加密算法，RAS 公开密钥加密算法，MD5 等，这些算法都非常安全，但也非常复杂，可根据需要选用，这里采用一般的加密方法，这些方法也是加密的基本原则。加密的基本方法是替换、变位和添加冗余信息。替换是将某一信息固定地替换为另一信息，如 A 变为 E 等；变位是将信息的顺序按一定规律变为另一种顺序，添加冗余是在有用的信息中添加无用的信息以增加破译的难度。如下例所示：

8	5	2	8	4	6	2	7	1
3	9	0	5	7	8	3	8	9
6	6	4	2	9	8	7	6	0
7	0	5	4	8	6	5	0	4

密钥为 852846271，守护程序向验证程序发送的随机数为

390578389705486504，将明文按先行后列的顺序排列，第二行为随机的冗余信息，密文按先列后行的顺序输出，列的顺序按密码由小到大，先左后右，这样密文为904045375798960886860367524，再将前两个 9 替换为 4，这时密文为 404045375748960886860367524。

如需提高加密强度，可采用增加密钥、随机数、冗余信息长度，并将以上密文再次变换等方法。

3. 守护程序及网络通讯：守护程序为监视用户的登录事件 需以系统管理员的身份在域用户管理器中打开系统安全日志 调用以下 Windows API 函数 OpenEventLog LPCTSTR lpUNCServerName LPCTSTR lpSourceName 打开事件记录句柄 ReadEventLog HANDLE hEventLog DWORD dwReadFlags DWORD dwRecordOffset LPVOID lpBuffer DWORD nNUMBEROfBytesToRead DWORD * pnBytesRead DWORD * pnMinNumberOfBytesNeeded 读取事件以及 NotifyChangEventLog HANDLE hEventLog HANDLE hEvent 发送消息给守护进程，在收不到验证进程的认证信息时调用 WnetCancelConnecttion2 LPTSTR lpName DWORD dwFlags BOOL fForce 函数中断该网络连接。在 Windows 平台下通讯可使用 NetBEUI NetBIOS 协议，但 TCP/IP 下的套接字 Sockets 的方式更通用和方便。

收稿日期：2000 年 8 月 10 日）

上接第 86 页）

```
}
```

16. 创建和测试应用程序

我们可以在窗口中看到一个白色球体围绕圆筒转动，而圆筒在屏幕中心旋转，圆筒的内表面为绿色，外表面为蓝色。球体代表光源的位置，它只是一个参照物。

四、总结

由于篇幅的限制，本文仅给出了一个 OpenGL for Windows NT 编程的框架，以此为出发点，可以加入众多的 OpenGL 函数 实现复杂的图形功能。本文作者一直从事三维视景的研究，成功地使用 OpenGL 在 Windows NT 下开发了某型模拟器的三维视景仿真系统，取得了良好的效果。

参考文献

1. OpenGL 三维图形设计 星球地图出版社 1996
2. 3D Graphics Programming with OpenGLTM QUE
3. OpenGL Programming Guide
4. OpenGL Reference Manual
5. VISUAL C++5 开发人员参考手册 机械工业出版社 1998

收稿日期：2000 年 11 月 24 日）

上接第 89 页）

```
            aItem.Update                    '刷新文字
        Else
         If aItem.EntityType = acBlockReference Then'查询块
            '得到块的插入点
            InsBlockPt = aItem.insertionPoint
            '得到块的比例系数
            dblBlockXscale = aItem.XScaleFactor
            dblBlockYscale = aItem.YScaleFactor
            '得到块的新比例系数
dblBlockXscale = dblBlockXscale * (dblScaleB / dblScaleA)
dblBlockYscale = dblBlockYscale * (dblScaleB / dblScaleA)
            aItem.insertionPoint = InsBlockPt
            '更新块的比例系数
            aItem.XScaleFactor = dblBlockXscale
            aItem.YScaleFactor = dblBlockYscale
            '刷新块
            aItem.Update
          End If
        End If
    Next i
    MsgBox "比例尺转换完毕!", vbOKOnly + vbInformation, "比例尺"
    End
End Sub
Private Sub Form_Load()
    '选项初始化
    OptA(1).Value = True
    OptB(1).Value = True
    TxtA.Text = 5000
    TxtB.Text = 5000
    TxtA.Enabled = False
    TxtB.Enabled = False
End Sub
Private Sub OptA_Click(Index As Integer)
    '源比例尺选项
    intA = Index + 1
    If intA = 4 Then
        TxtA.Enabled = True
    Else
        TxtA.Enabled = False
    End If
End Sub
Private Sub OptB_Click(Index As Integer)
    '目的比例尺选项
    intB = Index + 1
    If intB = 4 Then
        TxtB.Enabled = True
    Else
        TxtB.Enabled = False
    End If
End Sub
```

收稿日期 2000 年 10 月 18 日

博士信箱 QUESTION&ANSWER

电脑硬盘系统使用与维护常见问题解答

用 FAT16 格式对硬盘分区有何缺点

它采用 16 位的文件分配表，能支持的最大分区为 2GB，是目前应用最为广泛和获得操作系统支持最多的一种磁盘分区格式，几乎所有的操作系统都支持这一种格式，从 DOS 、Windows 3. x、Windows 95、Windows 97 到现在的 Windows 98、Windows NT、Windows 2000，甚至 Linux 都支持这种分区格式。

但是 FAT16 分区格式有一个最大的缺点，那就是硬盘的实际利用效率低。因为在 DOS 和 Windows 系统中，磁盘文件的分配是以簇为单位的，一个簇只分配给一个文件使用，不管这个文件占用整个簇容量的多少。而且每簇的大小由硬盘分区的大小来决定，分区越大，簇就越大。例如 1GB 的硬盘若只分一个区，那么簇的大小是 32KB，也就是说，即使一个文件只有 1 字节长，存储时也要占 32KB 的硬盘空间，剩余的空间便全部闲置在那里，这样就导致了磁盘空间的极大浪费。FAT16 支持的分区越大，磁盘上每个簇的容量也就越大，造成的浪费也越大。所以随着当前主流硬盘的容量越来越大，这种缺点变得越来越突出。为了克服 FAT16 的这个弱点，微软公司在 Windows 97 操作系统中推出了一种全新的磁盘分区格式 FAT32。

用 FAT32 格式对硬盘分区的优点在哪里

这种格式采用 32 位的文件分配表，使其对磁盘的管理能力大大增强，突破了 FAT16 对每一个分区的容量只有 2GB 的限制，运用 FAT32 的分区格式后，用户可以将一个大硬盘定义成一个分区，而不必分为几个分区使用，大大方便了对硬盘的管理工作。而且，FAT32 还具有一个最大的优点，就是在一个不超过 8GB 的分区中，FAT32 分区格式的每个簇容量都固定为 4KB，与 FAT16 相比，可以大大地减少硬盘空间的浪费，提高了硬盘的利用率。

目前支持这一磁盘分区格式的操作系统有 Windows 97、Windows 98 和 Windows 2000。但是，这种分区也有它的缺点，首先是采用 FAT32 格式分区的磁盘，由于文件分配表的扩大，运行速度比采用 FAT16 格式分区的硬盘要慢；另外，由于 DOS 系统和某些早期的应用软件不支持这种分区格式，所以采用这种分区格式后，就无法再使用老的 DOS 操作系统和某些旧的应用软件了。

用 NTFS 格式对硬盘分区的特色是什么

NTFS 分区格式是网络操作系统 Windows NT 的硬盘分区格式，使用 Windows NT 的用户必须同这种分区格式打交道。其显著的优点是安全性和稳定性极其出色，在使用中不易产生文件碎片，对硬盘的空间利用及软件的运行速度都有好处。它能对用户的操作进行记录，通过对用户权限进行非常严格的限制，使每个用户只能按照系统赋予的权限进行操作，充分保护了网络系统与数据的安全。但是，目前支持这种分区格式的操作系统不多，除了 Windows NT 外，刚刚上市的 Windows 2000 支持这种硬盘分区格式。

不过与 Windows NT 不同的是，Windows 2000 使用的是 NTFS 5. 0 分区格式。NTFS 5. 0 的新特性有 “磁盘限额” 管理员可以限制磁盘使用者能使用的磁盘空间； “加密” 在从磁盘读取和写入文件时，可以自动加密和解密文件数据等。随着 Windows 2000 的普及，广大电脑用户会逐渐熟悉这种分区格式的。

Linux 操作系统对硬盘分区的格式与其他操作系统有何不同

Linux 操作系统为自由软件，几乎不用花钱就能装入电脑，所以赢得了许多用户。它的磁盘分区格式与其他操作系统完全不同，共有两种格式：一种是 Linux Native 主分区，一种是 Linux Swap 交换分区。这两种分区格式的安全性与稳定性极佳，使用 Linux 操作系统后，死机的机会大大减少，能摆脱 Windows 常常崩溃的现象。但是目前支持这一分区格式的操作系统只有 Linux。

使用磁盘高速缓存应注意哪些问题

一般情况下 磁盘高速缓存都是用系统 RAM 来对磁盘进行缓存的。每当 CPU 从磁盘读数据时 磁盘高速缓存程序就将数据所在的扇区及邻近若干个扇区的数据都读入它所开辟的内存缓存区中，CPU 下次访问的数据落在这些扇区内的概率是比较大的 这样 CPU 就可以从 RAM 而不是硬盘获得需要的数据了；另一方面 高速缓存程序也可对 CPU 写往磁盘的数据进行缓存，办法是先将写往磁盘的数据存入内存高速缓存区，等到系统不忙或过段时间后，才真正向磁盘上写数据。

使用磁盘高速缓存应注意的问题是

1 不要随便关机。目前所有磁盘高速缓存都有一个共同无法克服的问题，即断电时的写缓存处理。如上所述 CPU 在写磁盘时，由于高速缓存程序的作用，数据实际上并没有真正被写到磁盘上去，而是滞留在缓存区中，如果在这时突然关机，对数据来说将是致命的。这一点对于 DOUBLESPACE 压缩盘来说尤为重要。所以每次关机时，都要注意所有数据是否真正存盘。

2 正确设置磁盘高速缓存的大小。磁盘高速缓存是在内

存中开辟的 这就带来了应该开辟多大的缓存区才算合适的问题。太小不能充分发挥高速缓存的作用 太大一方面用不着另一方面减少了内存空间。以 SmartDrive 为例 它专门对 2MB 内存进行优化 所以当内存超过 2MB 时 使用 SmartDrive 的缺省设置尺寸是正好合适的。

怎样安装外置活动硬盘

外置活动硬盘和外置光驱一样，它的安装分硬件安装和软件安装：

（）硬件安装

活动硬盘外盒上有两个接口：输出到电脑的并行口和转接到打印机的输出口。把硬盘盒螺丝拧开，打开盖，将 IDE 硬盘放到盒中，然后把硬盘盒内的电源线和数据线连接到硬盘上，使用螺丝固定硬盘，关上盖子，上紧螺丝。再把硬盘盒和电脑的并行口连线，把原来连接到电脑的打印机数据线连接到硬盘盒上，用电源线连接硬盘盒与电源插座，此时硬件安装过程完毕。

（）软件安装

执行软盘中相应的 Setup 安装程序就可以了。以 Windows95 /98 为例，把标有 Windows95 /98 的软盘插入软驱，单击 “开始”、 “运行”，键入 A Setup，单击 “确定”就可以安装了。安装完成后按提示重新引导，即可完成安装。

（）附属软件的安装

附属软件包括各种系统下的磁盘、光驱和并行口实用程序。用于硬盘分区、格式化、备份数据、使用光驱音频输出、检测并行口设备等等。可以把自己设计的应用软件制作成安装程序，拷贝到活动硬盘上，然后在其他的电脑上安装活动硬盘，并执行安装程序安装。

目前，活动硬盘的驱动程序支持 DOS、Windows 3. 2、Windows95 /98 /NT 、Linux、Xenix、Unix 等系统。

（）安装使用中的问题处理

按照说明安装成功后，可能无法看到活动硬盘上的逻辑驱动器。这可能有几种情况：

1）活动硬盘与系统上的其他设备冲突，可检查 “控制面板”、 “设备管理器”中的设备，看活动硬盘 通常以产品名为记录名）是否有黄色问号或感叹号。如果有，可动手解决资源占用冲突；

2）活动硬盘与电脑不兼容，需要改变一些设置。比如在联想电脑上安装不成功，需要执行 “OnSpec OnIDE”菜单中的 SwissKnife 软件，选择 “Device ”、 “Device Setting”、 “Turn off Burst mode”来解决问题；

3）硬盘没有分区、格式化。硬盘盒内部安装 IDE 硬盘应该是事先格式化的才能使用。有两种格式化方法：一是把硬盘安装到硬盘盒之前就在电脑上单独分区、格式化；另外的办法是在安装后执行 SwissKnife 软件，其中有分区和快速格式化功能。硬盘没有分区、格式化的最直接的现象是在 “Attach”时，硬盘盒上的硬盘指示灯稍亮一下，格式化过的硬盘在此时要多亮一会儿，使该灯显得特别亮。

活动硬盘支持 LPT1、LPT2、LPT3、LPT4 四个并行口，可以支持并行口内的任何设置，包括 EPP、ECP、SPP、ECP + EPP 和特殊的工作模式。按它们的数据传输速度由高到低排列如下：

Enhanced Parallel Port EPP

Enhanced Capabilities Port ECP

Bi - Directional IBM PS /2 Micro channel or Toshiba 兼容

Bi - Directional printer port

Standard AT Uni - Directional printer port SPP

驱动程序可以自动识别 EPP 接口，即使 BIOS 中设置的是低速的。驱动软件可以跳过主板和 Windows 9x 中关于并行口工作模式的设置，直接使用最高速度模式工作。活动硬盘支持 IEEE 1284. 3 标准，与 HP Laser Jet 4 5 系列全兼容。

如果安装后打印机不能工作，可能是该外置活动硬盘没有通电，不通电时该转接功能失效。

怎样处理在 BIOS 中检测不到硬盘的问题

从以下几个方面检查：

检查 IDE 接口与硬盘间的电缆线是否连接好；

检查 IDE 电缆线接头处接触是否出现断裂；

检查硬盘是否接上电源或者电源接头是否插牢。

如果检测时硬盘灯亮了几下，但 BIOS 仍然报告没有发现硬盘，则可能是硬盘电路板上某个配件损坏；或主板 IDE 接口及 IDE 控制器出现故障；或接在同一个 IDE 接口上的两个设备都设成主设备或从设备了。首先确认各种连线是否有问题，接下来应用替换法确定问题所在。

怎样处理在 BIOS 自检时报告 “HDD Controller Failure”错误

如果 BIOS 在自检时等待很长时间后出现 “HDD Controller Failure”错误信息提示，可能是因为 IDE 电缆线接触不良或者接反了。如果在自检时硬盘出现 “喀、喀、喀”之类的周期性噪音，则表明硬盘的机械控制部分或传动臂有问题，或者盘片有严重损伤。

如何处理开机找不到硬盘的故障

系统找不到硬盘故障的原因可能有两种：一是主板电池问题；二是硬盘问题。一些比较旧的主板直接将电池焊在主板上，这类主板在电池损坏的情况下经常会出现 CMOS 数据丢失的警告信息，而常见的现象就是无法检测到硬盘。此类现象一般发生在每天第一次开机时。开机后，电源会自动对焊接在主板上的电池充电，电池具有一定电量后就可以正常开机了。另外，如果排除主板原因 将硬盘拿到其他电脑上使用，看是否依然存在该现象），这也是硬盘即将损坏的一种预警。可用 Scandisk 对你的硬盘作一次全面检测，并备份好你的重要数据。