与 新 世 纪 同 行

送走了2000年，迎来了新世纪的开端，新的开始带给我们新的希望，美好的憧景，也同时带来了更重的责任，更大的压力。《电脑编程技巧与维护》杂志自1994年7月创刊以来，回首她所经历的风风雨雨，几多感慨，几多欣慰。她所走过的道路虽然坎坷，但锻炼了我们的意志，增强了我们服务好读者的信心和一定办好刊物的决心；回首她所走过的路程，我们更深深地感到《电脑编程技巧与维护》这株小苗从小到大，从稚嫩到成熟，每走一步，每出一期都饱含着作者和广大读者的关爱和支持，都饱含着各级领导和广大同仁的爱护和指导，在这新世纪之初，我们杂志社的全体同仁向所有爱护、支持和指导我们杂志的朋友们致以最崇高的敬意和衷心的感谢。

创刊的时候，我们就提出“买一台电脑并不难，难的是用好电脑”，它体现了我们杂志的“实用第一，智慧密集”的办刊宗旨。不管计算机软硬件技术发展多么快，新的软件产品推出的多么快，我们一直紧紧跟踪计算机软硬件的发展和应用的趋势，将刊物的读者定位在电脑编程人员、软件开发人员以及电脑软硬件系统维护人员。为使广大编程人员和电脑爱好者真正从该刊物受益，在稿件内容的选用上紧紧扣住了“编程”这个主题，重点报导国内外计算机软硬件发展的趋势、最新技术、最新产品，详细介绍软件开发、应用以及维护的经验、方法与技巧。在刊物的制作上从组稿、初审、定稿、一校、二校、三校、终审，直到印刷装订到发行，层层把关，严格管理。多年来上下团结一致，发扬团队精神，一切为着读者，一切服务于读者，不断为提高刊物的质量而努力。在市场经济条件下，由于刊物定位准确、内容新颖、丰富、实用性强、信息量大，深受广大电脑编程人员的一致好评，正像许许多多的读者来信、来函、来电指出的该刊物“专业性强，深入程序内部，技术含量高，内容新颖实用，贴近读者，……”。这些评价或评论一方面是广大读者对我们刊物的肯定，另一方面更是对我们工作的鼓励和鞭策。我们只有一种想法，把我们这个刊物办得更实用、更贴近读者，为读者提供更好的服务。

作为电脑编程技术性很强，特点鲜明的刊物，《电脑编程技巧与维护》为了能够给广大读者奉献新颖实用的内容，对每一期刊物都本着精益求精的精神，百尺竿头、再进一步。创刊以来，根据计算机软硬件产品的不断更新换代；根据计算机应用的不断深入，根据广大读者的建议与要求，我们对刊物的栏目进行了多次调整。1999年取消了“多媒体和Windows编程”栏目，增添了“编程与应用起步”栏目，该栏目面向有一定

电脑基础，但尚不能熟练使用各种开发语言和开发工具进行开发工作的初学编程的电脑爱好者。2000 年增加了“专家论谈”栏目，针对最新推出的操作系统平台、软件开发工具编程技巧和经验作深入的探讨，将专家们的智慧奉献给广大读者。为了不断提高刊物的质量和品味，2000 年应广大读者的要求，为了加大信息量，增加更多、更实用的文章，刊物的页码由原来的 84 页增加到 100 页；印刷用纸提高了一个档次；在版式设计上正文的文字由原来的五号字体改为小五号，大大增加了信息的含量；为了更快、更及时地与广大读者沟通与交流，从今年的 7 月我们开通了网站（http：//www. comprg. com. cn），不但及时将当月的期刊的源代码和精品文章上网，同时将创刊以来的所有文章目录都上了网，供读者选用；为了回报广大订户，方便对刊物中的源代码的使用，经过精心的准备。制作了 2000 年源代码光盘并附许多实用的共享软件赠送给广大的用户。所有这些变化为的是更好地体现“实用第一，智慧密集”的创刊宗旨。所有的工作都围绕着一个主题，只要是有用的，对读者有切实的帮助，我们就不遗余力地去努力做，而且尽力做好。

从过去，到现在，再到未来，《电脑编程技巧与维护》都在变化，变化是绝对的。迈入了 21 世纪，电脑仍将以其自身的规律迅速发展。紧跟电脑技术的发展和应用的步伐是我们的心愿，不断提高刊物的品味和质量是我们永远追求的目标。在 2001 年刊物栏目设置总体格局不作大的变动，但在“编程语言”栏目增加 Linux 技术开发和应用方面的文章并不断加大分量；在“编程与应用起步”栏目中，从选稿、定稿上更多地照顾初学编程人员，其选用文章的数量加大一倍；在“网络技术”栏目中加重网络实用技术和网络应用力度；在“计算机维护”栏目中增加计算机硬件的维护方法、经验方面的文章。网站的建设上，根据读者的建议，我们进一步充实一些栏目内容，增加一些新栏目，我们的目标把网站办成一个内容充实、信息量大、使用方便、新颖美观、高技术含量、交互功能强，多方面服务读者的实用网站。

2001 年 新千年开始的一年，我们依然把“实用第一，智慧密集”的宗旨贯彻到工作的各个环节，体现到每篇文章中，力争把最需要的东西及时送到读者手中，把广大作者的智慧结晶及时、快速地传播出去，为作者和读者们做更多、更踏实的工作，以更好的质量和更高的品味回报读者，为我们电脑事业应用与发展作出应有的贡献。面向新世纪让我们继续与广大读者携手一路同行。

《电脑编程技巧与维护》杂志社

电脑编程技巧与维护月刊

2001年第1期　总第79期　1994年7月创刊）　每月3日出版

编辑出版　《电脑编程技巧与维护》杂志社
法律顾问　佟秋平 亿中律师事务所
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

新技术追踪

编程与应用起步

编程语言

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 号湖北荆州市职业中专
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（七）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

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？**____________________________________

____

2001年《电脑编程技巧与维护》杂志订单

<table>
<tr><td colspan="2">订购单位</td><td colspan="4"></td><td colspan="2">收件人</td><td></td><td>经办人</td><td></td></tr>
<tr><td colspan="2">通讯地址</td><td colspan="7"></td><td>邮　编</td><td></td></tr>
<tr><td colspan="2">订户银行
及 账 号</td><td colspan="4"></td><td rowspan="3">收
款
单
位</td><td colspan="4" rowspan="3">盖章
年　月　日</td></tr>
<tr><td>单
价</td><td>7.80
元/期</td><td>份
数</td><td></td><td>合
计</td><td></td></tr>
<tr><td colspan="2">大写金额</td><td colspan="4"></td></tr>
</table>

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

英国对 Media Player 7 的漏洞提出警告

网络安全性软件供应商 GFI 于日前宣布，发现了 Windows ME Millennium Edition 中附带的 Windows Media Player 7 中存在着安全漏洞“WMS Script Execution”，并且通知了微软公司。当 Windows ME 的用户使用 Windows Media Player 7 浏览 Web 网站或者打开用 HTML 书写的电子邮件时，存在着允许起动恶意用户编写的有害代码的可能性。微软也承认了这一事实，并且在该公司的安全公告板 Microsoft Security Bulletin 上向用户发出了警告 编号 MS00 - 090。其中还涉及到对 Windows Media Player 6. 4 和 7 都有影响的“ASX Buffer Overrun”问题 。

微软推出 Beta 2 版 FrontPage 10

日前，微软推出了 Beta 2 版 FrontPage 10，其中含有新的图形和内容工具，并进一步增强了与其他软件的协作性。FrontPage 10 集成以下工具：Photo Gallery 照片图库 ，用于快速而方便地建立用户定制图库；Automatic Web Content 自动网站内容 ，可在用户的站点添加 MSNBC 的新闻和天气预报、Expedia 的地图、bCentral 的小型企业服务工具等微软站点内容。Microsoft SharePoint，一种基于浏览器模式的工具，用于建立联合网站，供内联网或互联网用户储存和共享信息。用户还可利用 FrontPage 10 的管理工具建立专门的 HTML 或 Excel 报告以跟踪网站访客。Beta 2 版 FrontPage 10 是 Beta 2 版 Office 10 的一个组件，现已被分发到约 10 000 名测试者手中。微软计划在 2001 年上半年正式推出 Office 10 套装软件，届时也有独立版本的 FrontPage 10 上市。

威盛发布两款新型 Cyrix 处理器

威盛公司近日宣布，将有两款新的 Cyrix III 处理器上市。这两款处理器主要面对低端个人电脑，工作频率分别为 650Mhz 和 667Mhz，是目前 Cyrix 最快的产品。严格说来，这两款 Cyrix III 与赛扬 766Mhz、毒龙 800Mhz 的市场并不完全重合。并且这两款芯片支持 133Mhz 外部总线，兼容 MMX 以及 3D Now！指令集。Cyrix III 650Mhz、667Mhz 处理器采用 0. 18 微米技术，带有 128K 缓存，使用 Socket 370 插脚。据称，这种处理器芯片的尺寸只有 75 平方毫米，是目前最小的。Cyrix 650Mhz 千颗采购单价为每颗 55 美元，667Mhz 的千颗采购单价为 60 美元。

IBM 研制成功新一代加密技术

IBM 日前宣布研制成功新一代加密技术，且运行速度非常之快，所需时间仅相当于目前最快的加密技术的一半。IBM 称这项技术对移动通信非常适合，因为它对处理器的要求非常低。IBM 声称，它开发的加密技术还包括认证功能，这在加密技术的研制中尚属开创之举。不过，部分电脑安全技术专家对此不以为然。他们指出，将加密技术与认证技术合二为一并非 IBM 的首创，而且也没有什么实际意义。有学者指出，在九十年代中期，澳大利亚蒙那什大学的研究人员就开展了加密认证技术的研究。他们还指出，即使 IBM 开发的技术有应用价值 且不说要验证其应用价值需要数年甚至数十年的时间 ，它的运行速度也只是快两倍而已。

AMD 预采用‘超导硅”为其高频芯片降温

据知情者透露，在过去近 12 个月的时间中，AMD 公司一直在悄悄使用一种纯度更高的硅，用来制造测试用的微处理器。据称，这种高纯度的硅可以帮助 1. 7GHz 的‘巴洛米诺马”解决棘手的发热问题。美国加州的 Isonics 公司开发出一种同位素纯净硅，这种硅比天然硅具有更好的导热性能。但 Isonics 公司没有证实它是否在和 AMD 公司合作，它只是说：“一家主要的微处理器制造商已经试用了同位素纯净硅晶圆，并告诉我们，他们的 1GHz 微处理器的最高温度下降了 35 摄氏度。”这种复杂的冷却系统虽然可以增强现有微处理器设计方面的性能，但是，它的造价却又高得惊人，甚至要高于微处理器本身的价格。

可卷式‘软盘”ThinDisc 问世

美国 ThinDisc 媒体公司的创始人之一，电子工程师马修·里克花费了四年的时间，研制成功一种非常薄的磁盘——ThinDisc。它可能将给予“软盘”这个词以新的含意。ThinDisc 的厚度只有一张普通 DVD 或者 CD 光盘厚度的五分之一。它还非常柔软，可以包在一个可乐罐外，或者卷入一本杂志中，而不用担心会破裂。里克表示，由于材料和包装更为便宜，ThinDisc 的成本大概只有普通光盘成本的一半左右。它出现后，企业将得以低成本邮寄数字式多媒体广告、软件补丁，以及音乐和电影的免费样品，ThinDisc 的首批客户将来自媒体行业、出版商、广告商和娱乐公司。不过，里克还希望将 ThinDisc 的运用拓展至其它一些不显著的领域。他说：“通过使用 ThinDisc，商家可以把有关安全使用的视频内容包在一瓶处方药或者电力工具上。”

美国公司开发出供电线 Modem 芯片组

Adaptive Networks 开发出传输速度为 3. 55Mbit / s 整体达到 20Mbit / s 的供电线 Modem 芯片。该芯片具有不间断传输数据流视频数据的 QOS quality of service 功能和强有力的纠错功能，可将误码率降低到 1×10^{-9}。调制方式采用频谱扩散方式，扩散带宽为 5MHz ~ 40MHz。该公司已经向推动供电线 Modem 标准化的 CEA R7. 3 提交了该芯片方案。目前 CEA R7. 3 正在评价由各公司提出的方式，据说计划在 2001 年第 3 季度完成草案。Adaptive Networks 在美国有线电视专业展示

会“The Western Show 2000”上的技术展台“Cable Net”上公开了使用该芯片组的样机。据说芯片组的价格为 50 美元，而设计的参考价格将在 3000 美元左右。

可使 Java 处理速度提高 2.5 倍的技术问世

日本的 Aplix 公司开发出了可以将搭载在手机等移动设备上的 Java VM Virtue Machine，虚拟机 提高至 2.5 倍速的技术“KFTT”。该公司表示，‘手机厂商已经决定将该技术用于 2001 年上市的手机上。”该公司所开发的 VM，遵循用于手机等移动终端的 Java VM 规格“CLDC”，并与美国公司 Sun Microsystems Inc. 开发的虚拟机——“KVM”相互兼容。这种与 KVM 相互兼容的 VM，如果快的话将配置在 NTT DoCoMo 预定于年内上市的手机上。另外，J - Phone 也计划在 2001 年上市的手机上使用 Aplix 公司的 Java 技术“Jblend”，其中便采用了 KFTT 技术。KDDI 集团对于搭载 Java 技术的手机，将同 J - Phone 的制式保持一致。此次 Aplix 公司所开发的技术，是以该公司开发的 Java VM 的高速化技术“FTT3”为基础的，并且使用了高速化技术专利。

索尼推出 9.1GB 的 5.25 英寸 MO 光驱

索尼宣布，将于 2000 年 12 月份上市对应双面容量为 9.1GB 的 5.25 英寸光磁盘的驱动器“SMO - F561”。在基于现行 ISO 规格的 5.25 英寸型光磁盘中，索尼实现了每张最大可容纳 5.2GB 数据的光盘产品化。该公司于 1999 年 4 月份宣布将其容量进一步提高到 9.1GB。驱动器采用磁气超解像读取技术——CAD center aperture detection 方式的 MSR magnetic super resolution 技术，可以读出比激光光束点还要小的记录点。在光磁盘中则采用 LandGlove 方式提高了面记录密度。另外，在此之前索尼还提出过在一张 5.25 英寸型光磁盘上最多记录 40GB 数据、相当于 9.1GByte 下一代版本的可改写光磁盘规格“UDO Ultra Density Optical 规格”方案。

iPlanet 发布首个智能通讯平台

日前，Sun / Netscape 的联盟 iPlanet 推出了智能化通讯平台的概念，该可扩展的智能平台包括最新的 iPlanet 通信服务器与日历服务器和目录服务器，它能够使用户快速的通过语音、有线与无线网络进行沟通，允许互联网服务商、电信公司与企业充分的发挥其创造力，从而使其服务产生革命性的变化。iPlanet 是第一家能够提供全面的 Internet 服务开展平台的公司，其软件环境能够在目前常用的各种操作系统和网络硬件设施上运行。IPlanet 智能化通讯平台是市场上领先的产品，包括 iPlanet 信息服务器、日历服务器、无线服务器和目录服务器。

微软和多媒体公司推出能上网的 TV

日前，微软和汤姆森 Thomson 多媒体公司推出了一款能上网的模拟 TV，确保它们在转向数字电视前，能在欧洲数十亿欧元的电视市场上占一席之地。这款纯平电视机将于今年的二月份开始在法国销售，它可以通过模拟电话线提供增强的电视内容、互联网访问和交互式服务，并且与数字服务兼容。微软把自己定位为数字式电视的先锋。按双方的计划，Tak 电视机将占领欧洲巨大的模拟电视机市场，欧洲的数字电视发展还相当地缓慢。据专家估计，到 2005 年，全球将有 1.79 亿个家庭使用网络电视服务，届时法国的这一市场将达到 30 亿欧元。分析家们表示，尽管欧洲正在快速地转向数字化电视，但推出一种‘补缺”式的产品仍不失为一种聪明的市场行为。

内存新革命——不怕掉电的 MRAM 内存诞生

启动电脑时再没有漫长的等待，办公室掉电时再不会有数据丢失。这就是 IBM 磁记录随机存储芯片带给我们的。从 1974 年开始，IBM 就开始了磁存储技术的研究。经过辛勤的探索，被称为 MRAM magnetic random - access memory 的新一代内存即将走出实验室，进行大批量生产。MRAM 采用了夹心饼干一样的技术，两边是金属，中间是一层磁性材料。二进制数据就存储在磁性材料里。与我们目前使用的内存不同，磁性内存可以在没有电的情况下将数据记录下来。MRAM 既可以起到传统内存的作用，又能够替代 Flash 的位置。虽然 IBM 已经和德国 Infineon 公司达成协议，共同生产这一产品。但 MRAM 真正成为主流产品仍需一些时间。

VC++系列讲座（一）

张博强　徐　亮

编者按

本系列讲座的目的是帮助读者学习 VC++，达到入门和提高。讲座共分五讲：第一讲学习VC++ 相关的基础知识；第二讲介绍 MFC 的基础类和消息机制；第三讲设计多文档界面应用；第四讲对话框应用；第五讲 VC++的高级的编程。

第一讲　学习 VC++相关的基础知识

在本讲中将主要介绍 Windows 编程及面向对象语言的特点，并学习面向对象程序设计语言 C++的相关基础知识，结合例子分析指针和类用法。

一、进入 VC++

Visual C++作为一个功能非常强大的可视化应用程序开发工具，是计算机界公认的最优秀的应用开发工具之一，被广泛的应用于 Windows 32 位平台基础应用程序的开发。Microsoft 的基本类库 MFC 使得开发 Windows 应用程序比以往任何时候都要容易；功能强大的 App Wizard 程序生成器大大减轻程序员的工作量，减少了程序出错的几率。

1. Windows 程序设计的特点

Mirosoft Windows 以一种全新的思维方式进行程序设计，主要表现为以下几点：

事件驱动的程序设计

事件驱动程序设计是一种全新的程序设计方法，它不是由事件的顺序来控制，而是由事件的发生来控制，而这种事件的发生是随机的、不确定的，并没有预定的顺序，这样就允许程序的用户用各种合理的顺序来安排程序的流程。对于需要用户交互的应用程序来说，事件驱动的程序设计有着过程驱动方法无法替代的优点。它是一种面向用户的程序设计方法，它在程序设计过程中除了完成所需功能之外，更多的考虑了用户可能的各种输入，并针对性的设计相应的处理程序。它是一种‘被动”式程序设计方法，程序开始运行时，处于等待用户输入事件状态，然后取得事件并作出相应反应，处理完毕又返回并处于等待事件状态。

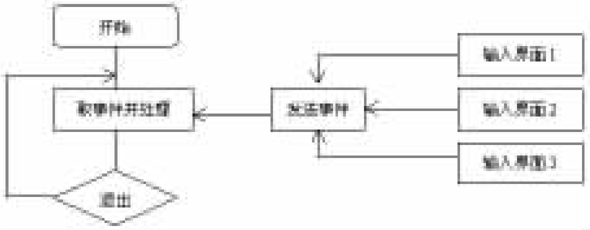


图 1　事件驱动程序模型

消息循环与输入

事件驱动围绕着消息的产生与处理展开。事件驱动是靠消息循环机制来实现的。

消息是一种报告有关事件发生的通知。消息类似于 DOS 下的用户输入，但比 DOS 的输入来源要广，Windows 应用程序的消息来源有以下四种：

（1）输入消息：包括键盘和鼠标的输入。这一类消息首先放在系统消息队列中，然后由 Windows 将它们送入应用程序消息队列中，由应用程序来处理消息。

（2）控制消息：用来与 Windows 的控制对象，如列表框、按钮、检查框等进行双向通信。当用户在列表框中改动当前选择或改变了检查框的状态时，发出此类消息。这类消息一般不经过应用程序消息队列，而是直接发送到控制对象上去。

（3）系统消息：对程序化的事件或系统时钟中断作出反应。一些系统消息，象 DDE 消息　动态数据交换消息）要通过 Windows 的系统消息队列，而有的则不通过系统消息队列而直接送入应用程序的消息队列，如创建窗口消息。

（4）用户消息：这是程序员自己定义并在应用程序中主动发出的，一般由应用程序的某一部分内部处理。

Windows 操作系统包括三个内核基本元件：GDI，KERNEL，USER。其中 GDI 图形设备接口 负责在屏幕上绘制像素、打印硬拷贝输出，绘制用户界面包括窗口、菜单、对话框等。系统内核 KERNEL 支持与操作系统密切相关的功能：如进程加载，文本切换、文件 I/O，以及内存管理、线程管理等。USER 为所有的用户界面对象提供支持，它用于接收和管理所有输入消息、系统消息并把它们发给相应的窗口的消息队列。消息队列是一个系统定义的内存块，用于临时存储消息；或是把消息直接发给窗口过程。每个窗口维护自己的消息队列，并从中取出消息，利用窗口函数进行处理。

图形输出

Windows 程序在输出上有如下特点：

（1）Windows 的每一个应用程序对屏幕的一部分进行处理，由操作系统来统一管理屏幕输出；每个窗口要输出内容时，必须首先向操作系统发出请求 GDI 请求 ，由操作系统完成实际的屏幕输出工作。

（2）Windows 程序的所有输出都是图形。Windows 提供了丰富的图形函数用于图形输出，这对输出图形是相当方便的，但是由于字符也被作为图形来处理，输出时的定位要复杂一些。

（3）Windows 下的输出是设备无关的。如对于不同类型的

打印机 GDI 会自动将图数据转换成不同的请求形式，以适应打印机完成对同一个图形的输出，但在用户看来并没有什么区别。

GDI 的图形输出是面向窗口的，面向窗口包含两层含义：

(1) 每个窗口作为一个独立的绘图接口来处理，有它自己的绘图坐标。当程序在一个窗口中绘图时，首先建立缺省的绘图坐标，原点 (0, 0) 位于窗口用户区的左上角。每个窗口必须独立的维护自己的输出。

(2) 绘图仅对于本窗口有效，图形在窗口边界会被自动裁剪，也就是说窗口中的每一个图形都不会越出边界。即使想越出边界，也是不可能的，窗口会自动的防止其他窗口传过来的任何像素。这样，你在窗口内绘图时，就不必担心会偶然覆盖其他程序的窗口，从而保证了 Windows 下同时运行多个任务时各个窗口的独立性。

资源共享

Windows 要求应用程序必须以一种能允许它共享 Windows 资源的方式进行设计，它的基本模式是这样的：

(1) 向 Windows 系统请求资源；

(2) 使用该资源；

(3) 释放该资源给 Windows 以供别的程序使用。

因此在应用程序设计时一般不要直接访问内存或其他硬件设备，如键盘、鼠标、计数器、屏幕或串口、并口等。Windows 系统要求绝对控制这些资源，以保证向所有的应用程序提供公平的不中断的运行。如果确实要访问串并口，应当使用通过 Windows 提供的函数来安全的访问。

2. Windows 应用程序组成

前面介绍了 Windows 应用程序的特点，现在让我们看看编写一个 Windows 程序需要做哪些工作。编写一个典型的 Windows 应用程序，一般需要：

(1) C CPP 源程序文件：源程序文件包含了应用程序的数据、类、功能逻辑模块 包括事件处理、用户界面对象初始化以及一些辅助例程) 的定义。

(2) H HPP 头文件：头文件包含了 CPP、C 源文件中所有数据、模块、类的声明。当一个 CPP、C 源文件要调用另一个 CPP、C 中所定义的模块功能时，需要包含那个 CPP、C 文件对应的头文件。

(3) 资源文件：包含了应用程序所使用的全部资源定义，通常以 .RC 为后缀名。注意这里说的资源不同与前面提到的资源，这里的资源是应用程序所能够使用的一类预定义工具中的一个对象，包括：字符串资源、加速键表、对话框、菜单、位图、光标、工具条、图标、版本信息和用户自定义资源等。

在 Windows 程序设计过程中，象菜单、对话框、位图等可视的对象被单独分离出来加以定义，并存放在资源源文件中，然后由资源编译程序编译为应用程序所能使用的对象的映象。资源编译使应用程序可以读取对象的二进制映象和具体数据结构，这样可以减轻为创建复杂对象所需要的程序设计工作。

我们已经了解了 Windows 程序，但学好 VC 还应该先具备良好的 C++ 的基础。虽然C++ 不是 VC，但 '磨刀不误砍柴工" 最开始接触 VC 不应急于 Windows 程序的开发 而应该先熟悉面向对象语言的特征，了解并能熟练运用 C++ 语言的特性。所以，在此有必要先进行面向对象语言和 C++ 知识的学习。

二、支持面向对象程序设计的语言 C++

C++语言是在 C 语言基础上的吸收了其它一些语言的精华而逐渐发展起来的一种通用目的程序设计语言。C++对 C 的兼容这一要求使 C++不仅很好的既持面向对象的程序设计方法，也支持 C 所支持的面向过程的程序设计方法。

在 C 中一个很重要的概念就是指针，它也是 C++中重要的部分。

指针

指针是一种值，能在函数之间传递或被函数返回，能在表达式中被求值。存储指针值的对象叫做指针对象。在指针中值存储的是另一个对象的抽象的存储地址，这样就在程序中建立了指针对象与另一个对象之间的联系，使我们能够通过指针对象去访问被该指针所指向的对象。

声明指针对象语法是：

类型表达式 * 标识符；

例如：

int * pointer；

与指针对象有关的一个基本操作符是 '& "，这是一个一元操作符。它操作一个对象，得到这个对象的存储单元的地址。指针值就是对这种地址值的抽象。例如，在定义指针 'pointer" 的程序中，也定义了一个变量：

int var；

则表达式& var 的基类型是 int * 其值指向对象 var，所以可使用下面的语句初始或更新指针对象。

pointer = & var；

或用下面的方法对指针赋初始值：

int pointer & var ；

由于指针对象的具体值不是我们所关心的，我们所关心的是指针指向的对象。所以 C++为我们提供了 "* " 操作符，用于表达被指针所指向的对象。为了便于理解我们可以看一段程序：

```
#include < iostream.h >
int main()
{
  int var( 10 );
  int *pointer;
  pointer = & var;
  *pointer = *pointer + 10;
  count < <"var = " < <var < <endl;
```

}

程序的输出结果为：

var = 20

使用赋值语句

pointer = & var；

更新指针对象 pointer，然后在语句

＊pointer = ＊pointer + 10

通过指针对象 pointer 去操作 var 对象 。

但在 C++ 的学习过程中，我们必须清楚面向对象才是它真正想要支持的。在 C++ 中最高级的程序单元是函数和类，函数是支持面向过程的程序设计的语言构造。类是 C++ 中运行面向对象的程序设计的语言构造。

C++ 类的结构

C++ 类的结构可以分为两部分，一部分描述对象外部视图，另一部分是对内部数据的操作。为了便于理解，我们看一个划线的类：

```
 class CLine
{
private:
    int m_nX1;
    int m_nY1;
    int m_nX2;
    int m_nY2;
public:
    // 构造函数
    CLine();
    CLine(int x1, int y1, int x2, int y2);
    // 析构函数
    ~CLine();
    // 初始数据
    void SetPoints(int x1, int y1, int x2, int y2);
    // 画线
    void Draw();
}
```

其中，class 后的 CLine 是类名，类名通常以“C”为前缀，成员变量通常以“m_”为前缀。关键字 private 和 public 用于定义在类中声明的标识符能否被其它程序单元引用。public 以后声明的标识符是公有标识符，这些标识可被任一程序单元引用，关键字 private 以后声明的标符是私有标识符，这些标识符只能在该类的实现代码中被引用。无论是 public，还是 private 定义的标识符都称作是类的成员，如果是与数据结构相关的成员称作数据成员，例如，m_nX1 m_nX2 …… 如果是与操作有关的成员，称作成员函数，例如，Draw 。

在类中不带参数与类同名的成员函数 CLine ，叫作构造函。它将在对象创建的时候被执行，完成一些初始工作；与其相对的是析构函数，即 ~CLine ，它将在对象被删除之前完成一些清理工作。如果你没有定义构造函数或析构函数，编译器会自动生成无返回类型的空操作的构造函数或析构函数，如

```
CLine :: CLine ()
{
}
```

或

```
CLine :: ~CLine ()
{
}
```

其中“::”操作符是作用域运算符，

返回类型 类名 ::成员函数名 (参数声明)

{

语句

}

是类实现成员函数的一般形式。

例如我们要实现 SetPoints int x1 int y1 int x2 int y2 成员函数，可以使用类似于下面的形式：

```
void CLine::SetPoints(int x1, int y1, int x2, int y2)
{
    m_nX1 =x1;
    m_nX2 =x2;
    m_nY1 =y1;
    m_nY2 =y2;
    return;
}
```

前面已经讲过，public 所定义的标识符，能够被作任一程序单元所引用，它将是类对外的接口，程序员通过这些接口实现类的功能。下面的一些例子说明了类 CLine 使用情况：

```
CLine *pMyLine;
CLine Myline1;
MyLine2 = new CLine(0, 0, 10, 10);
(*pMyLine). Draw();
pMyLine->Draw();
```

其中，CLine ＊pMyLine 是声明了一个指向 CLine 类的指针，它的类型为 CLine ＊，而 CLine Myline1 是调用了 CLine 的构造函数，应加以区别。MyLine2 = new CLine 0 0 10 10 是调用了成员函数 CLine int x1 int y1 int x2 int y2 ，关键字 new 是用来创建新的 C++ 对象，如果你用过 C，它类似于 C 中的 malloc，在 new 之后，要使用 delete 释放空间，这类似于 C 中的 free。

```
MyLine2 = new CLine(0, 0, 10, 10);
```

也可以写成

```
CLine(0, 0, 10, 10) Myline2;
```

语句 ＊pMyLine . Draw 和 pMyLine->Draw 作用是相同的，它们都是调用 Draw 员函数。“->”操作符使语义更直观，要使用第一种形式，＊pMyLine 一定要用括号括起来，因为“＊”操作低于“.”操作。

既然是一种面向对象程序设计的语言，它必然具有面向对象编程方法的特征。

面向对象的编程方法具有四个基本特征：

1. 抽象：

抽象就是为完成特定的任务面建立相对独立的程序单元，这样调用者不需要知道这个程序单元是怎样完成它的工作的，只需知道这个单元的功能是什么，以便更充分地注意与当前目

标有关的方面。C++语言中抽象包括控制抽象、过程抽象和数据抽象。

控制抽象定义了用于排列任意动作顺序的一种方法，它允许使用清楚的可理解的控制流编写面向命令的程序。这一层中，有所谓的结构化的程序设计的构造，如简单的顺序、选择、迭代等。

过程抽象是对一组输入对象的操作动作及产生的结果，在实现时表现为一组语句。在使用者眼中它可被看作是一个具有单独功能的实体，用户只关心它能够做什么。

数据抽象是一种更高级的抽象。高级语言将只关心数据能对它做什么，而不关心数据是怎样在计算机上表示的，是如何实现对数据进行操作的。如对于 2、2.1 我们只需知道它们分别为整数和实数，而不用管它们在计算机内是如何用二进制表示的；对于 1+2 的操作我们也不必考虑它们是如何在计算机上实现的。

2. 继承：

类继承（简称继承）是对象的一个新类可以从现有的类中派生，新类继承了原始类的特性，新类称为原始类的派生类（子类），而原始类称为新类的基类（父类）。派生类可以从它的基类那里继承方法和实例变量，并且类可以修改或增加新的方法使之更适合特殊的需要。继承机制建立了两个类之间的联系，使派生类能通过重用基类中的代码很快地实现，它提供了一种明确表述共性的方法。继承性很好的解决了代码的可重用性问题。比如说，所有的 Windows 应用程序都有一个窗口，它们可以看作都是从一个窗口类派生出来的。但是有的应用程序用于文字处理，有的应用程序用于绘图，这是由于派生出了不同的子类，各个子类添加了不同的特性。

3. 封装：

封装是面向对象的特征之一，是对象和类概念的主要特性。封装是把过程和数据包围起来，对数据的访问只能通过类中预留的接口来实现。面向对象计算始于这个基本概念，即现实世界可以被描绘成一系列完全自治、封装的对象，这些对象通过一个受保护的接口访问其他对象。一旦定义了一个对象的特性，则有必要决定这些特性的可见性，即哪些特性对外部世界是可见的，哪些特性用于表示内部状态。在这个阶段定义对象的接口。通常，应禁止直接访问一个对象的实际表示，而应通过操作接口访问对象，这称为信息隐藏。事实上，信息隐藏是用户对封装性的认识，封装则为信息隐藏提供支持。封装保证了模块具有较好的独立性，使得程序维护修改较为容易。对应用程序的修改仅限于类的内部，因而可以将应用程序修改带来的影响减少到最低限度。

4. 多态性：

多态性是指允许不同类的对象对同一消息作出响应。比如同样的加法，把两个时间加在一起和把两个整数加在一起肯定完全不同。又比如，同样的选择编辑-粘贴操作，在字处理程序和绘图程序中有不同的效果。多态性包括参数化多态性和包含多态性。多态性语言具有灵活、抽象、行为共享、代码共享的优势，很好的解决了应用程序函数同名问题。

面向对象程序设计具有许多优点：

·开发时间短，效率高，可靠性高，所开发的程序更强壮。由于面向对象编程的可重用性，可以在应用程序中大量采用成熟的类库，从而缩短了开发时间。

·应用程序更易于维护、更新和升级。继承和封装使得应用程序的修改带来的影响更加局部化。

要真正掌握 C++，并不是朝夕间就能做到的，不仅要经常编写一些程序以便熟悉语言特性，还应该多看书，学会看别人的程序。在具有了一定的 C++基础后，我们就可以开始正式的 Visual C++学习了。

小结

通过本讲的学习，我们应当掌握

1. Windows 编程的特点及 Windows 程序的组成；

2. 掌握面向对象程序设计的一些基本概念和特点，了解封装、继承、重载、多态性；

3. 掌握面向对象语言C++ 的基础知识，掌握指针的使用，理解类、构造函数、析构函数和成员函。

参考资料

张国峰编著. C++语言及基程序设计教程. 电子工业出版社

周伯生、童长忠等译. 面向对象式软件构造. 北京航空航天大学出版社

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

（收稿日期：2000 年 11 月 2 日）

iWay 开启企业信息管理之门

日前，BMC 软件公司、SUN 以及东大阿尔派三家 IT 著名公司在京联合举行“iWay 开启企业信息管理之门”新闻发布会，宣布将由东大阿尔派实施 SUN Enterprise 服务器产品和 BMC PATROL 企业管理方案捆绑销售计划，同时由其强大的营销服务队伍为上述产品提供售后服务支持。

三方的合作将使运行于 Sun Enterprise 强大 U-NIX 环境下的 BMC PATROL 应用管理软件在企业信息应用、监控、管理、优化等方面的性能表现得更加淋漓尽致。这种合作使用户可以在分布式或异构环境中轻松、方便地管理整个企业，开启企业信息管理之门。

在 Delphi 中实现 PACK 功能

陈 彬

Delphi 具有强大的数据库功能，用 Delphi 编写过数据库程序的朋友们，不知注意到没有，使用 Delphi 的数据集元件的“Delete”方法删除表中的记录之后，表的大小并没有改变，即删除的记录没有被物理删除。可是 Delphi 的数据集元件没有提供物理删除记录的方法，类似于 FoxPro 中的 PACK 函数功能。是不是 Delphi 没有提供实现 PACK 功能的函数或过程呢？其实不然，在 Delphi 中可以通过调用 BDE 函数：DbiPackTable 或 DbiDoRestructure 来实现 PACK 功能。下面是一个实现 PACK 功能的例子。

首先，打开 Delphi 集成开发环境，新建一个工程 Pack. dpr。在 Form1 对应的单元文件 Pack1. pas 中的 USE 部分加入 BDE 单元，并在 Form1 上放置如下几个元件并设置元件的属性：

```
Button1: TButton;
Button2: TButton;
Table1: TTable;
DataSource1: TDataSource;
DBGrid1: TDBGrid;
```

元件的属性设置如下：

```
Form1. Caption: =´Pack Table´;
Button1. Caption: =´Delete Record´;    Button2. Caption: =´Pack Table´;
Table1. DatabaseName: =´ . \´;    Table1. TableType: =ttDBase;
Table1. TableName: =´products. dbf´;
DataSource1. DataSet: =Table1;    DBGrid1. DataSource: =DataSource1;
```

第二步，双击 Form1 在 Form1 的 FormCreate 事件中添加如下代码：

```
procedure TForm1. FormCreate(Sender: TObject);
begin
  Table1. Active: =True;
end;
```

第三步，双击 Button1 在 Button1 的 Button1Click 事件中添加如下代码：

```
procedure TForm1. Button1Click(Sender: TObject);
begin
 Table1. Delete;                    //删除表中的当前记录
end;
```

第四步，双击 Button2 在 Button2 的 Button2Click 事件中添加如下代码：

```
procedure TForm1. Button2Click(Sender: TObject);
var
 hDb: hDBIDb;
 TableDesc: CRTblDesc;
begin
 if Table1. TableType =ttDBase then          //DBase 表
 with Table1 do
 begin
   active: =false;
   Exclusive: =true;
   active: =true;
    check(DbiPackTable(DBHandle, Handle, nil, szDBase, True));  //PACK 数据库
   active: =false;
   exclusive: =false;
   active: =true;
  end;
  if Table1. TableType =ttParadox then          //Paradox 表
  with Table1 do
  begin
   active: =false;
   exclusive: =true;
   active: =true;
   FillChar(TableDesc, sizeof(TableDesc), 0);
     DbiGetObjFromObj(hDBIObj(Handle), objDATABASE, hDBIObj(hDb));
   StrPCopy(TableDesc. szTblName, TableName);
   StrPCopy(TableDesc. szTblType, szPARADOX);
   TableDesc. bPack := True;
   active: =false;
   Check(DbiDoRestructure(hDb, 1, @ TableDesc, nil, nil, nil, False));     //PACK 数据库
   active: =true;
  end;
end;
```

在运行程序之前，先打开资源管理器，查看并记住 products. dbf 文件的大小，然后运行程序，products. dbf 表中的数据就显示在 DBgrid1 中。连续按几下“Delete Record”按钮，删除表中的几条记录，再在资源管理器中查看 products. dbf 文件的大小，可以发现文件大小没有改变，然后回到程序，按一下“Pack Table”按钮，再打开资源管理器，查看 products. dbf 文件的大小，可以发现文件变小了，表明第一次删除的记录已被物理删除了。

最后说明一点：对于 DBase 表，使用 DbiPackTable 函数来 PACK 数据库 而对于 Paradox 表，则使用 DbiDoRestructure 函数，用法也不一样。对于 SQL、Access 数据库，这两个函数都是不适用的。使用 Delphi 开发数据库程序的朋友们，不妨试试，在你开发的数据库程序中加入 PACK 功能，给数据库“减减肥”。以上程序在 Delphi 4 和 Delphi 5 开发环境，中文 Windows 98 操作系统下调试通过。

收稿日期：2000 年 8 月 2 日）

窗口快速重绘的虚拟窗口实现法

肖华勇

一、实现窗口重绘的三种基本技术比较

Windows 程序比起 DOS 程序的一大优越之处就是允许你在屏幕上打开许多窗口 并在各窗口之间可以自由切换。但这同时涉及到一个窗口重绘问题 当窗口 A 被窗口 B 覆盖或部分覆盖后 移去窗口 B 时 如何重新显示窗口 A 原先的内容？通常 窗口被覆盖后的重绘方法有三种，方法 1：当窗口内容是用某些计算方法来创建的 可以再次产生输出来实现窗口重绘。方法 2：可以预先保存窗口显示事件的记录 当窗口重绘时再使这些事件重新发生。方法 3：可以建立一个与显示窗口对应的虚拟窗口 每次向显示窗口写内容时 同时也向虚拟窗口写 这样当接到 WM_PAINT 消息需要重绘窗口时 就简单的将虚拟窗口的内容向显示窗口中复制就行了。

方法 1 实现起来最简单 对计算简单且快速的计算方法可以采用这种方法来实现窗口重绘 比如需要重新输出一行文字 重绘一个简单的图形不妨采用这种方法。但对计算复杂的一大段代码 或者计算量很大时 就不能采用这种方法了。试想 每次重绘都重新执行一大段代码 或者重新进行复杂的计算 是不是会使窗口响应很慢 用户没有耐心等待呢？方法 2 需要每次记录显示事件 这同样适用于比较简单的输出 对复杂的图形和文字混合内容记录起来则比较复杂 显示时还得从记录中选取那些需要重新显示的内容 使编程变得复杂 也缺乏通用性。因此最好的且通用的方法还是第三种 这种方法每次把要显示的内容同时写入一个与实际的物理窗口相联系的虚拟窗口之中 相当于给窗口内容作一个备份 覆盖后要重显只需从备份中拷贝一份过来就行了 不需要任何计算。当窗口中只是部分甚至很小一部分被遮住需要重显时 为加快速度还可以只重绘被遮住部分 没有遮住的部分则不需要花时间去重显。Windows 提供几个专门的函数用于实现这种虚拟窗口技术。这也是所有 Windows 程序窗口重绘的最常用技术。当你编制一个好的软件时 对窗口重绘工作不要忘了使用这种技术。

二、虚拟窗口技术的实现技巧

为了便于从底层根本上理解虚拟窗口实现技术，文中通过 Windows 的 API 函数来实现该技术，其实现过程分为三步：创建虚拟窗口；向虚拟窗口输出；虚拟窗口输出到显示窗口。

1. 创建虚拟窗口

首先创建一个与物理设备描述表相兼容的虚拟设备描述表 它需要在程序执行最初接收到 WM_CREATE 消息时就创建 该兼容的设备描述表在程序的全部执行过程中都将存在 . 具体过程如下 1 采用 GetDC 函数通过当前窗口句柄获得当前的设备描述表 hdc 2 使用 CreateCompatibleDC 在内存中创建兼容的设备描述表 此设备描述表的句柄保存在一个全局变量 memDC 中 3 使用 CreateCompatibleBitmap 创建一个与当前设备描述表 hdc 兼容的位图 hbit 这将在虚拟窗口与物理窗口之间建立一个一对一的的映射。位图大小是全屏幕尺寸 这是为了确保只要窗口不比屏幕大 位图就能够恢复；4 将该位图 hbit 采用 SelectObject 选入虚拟窗口 memDC 中； 5 创建一个画刷 如用于填充的白色画刷 并选入虚拟窗口 memDC 中。上面 5 步的实现参见程序 repaint. cpp 中的 case：WM_CREATE 部分。

2. 向虚拟窗口输出

做好上面的初始化准备工作后 在程序中任何部分每当向显示窗口输出内容 不管是文本还是图形 时 就必须向虚拟窗口输出同样内容 这实现起来很简单 只需将输出的窗口句柄由当前设备描述表 hdc 换为 memDC 就可以了。

3. 虚拟窗口输出到显示窗口

被覆盖窗口需要重绘时 需要接收到系统消息 WM_PAINT 才能执行。因此程序在接收到 WM_PAINT 消息时 就使用位图拷贝函数 BitBlt 将虚拟窗口内容向物理窗口设备中复制 为加快重绘速度 复制的区域只是被覆盖了的内容部分。

三、程序实现

下面是实现高效的窗口重绘的虚拟窗口实现技术。源程序为 repaint. cpp。运行该程序 当点击菜单 ‘画图 - >作图”，则在屏幕客户窗口中逐步绘制一幅彩色的漂亮晚霞图 当点击最大化窗口 则该图自动充满屏幕 当新开一个窗口 如记事本窗口 覆盖部分图形 移开后原图形自动被重绘 速度很快 几乎感觉不到重绘过程 就象图形本来就存在那里。如果通过计算重新画图 可以在清屏后查看 其绘图过程是比较慢的。

1. 执行程序 rcpaint. cpp

```
#include <windows. h>
#include ″wmenu. h″
LRESULT CALLBACK WindowFunc(HWND, UINT, WPARAM, LPARAM);
 char szWinName[] = ″RcPaint″;
  int WINAPI WinMain(HINSTANCE hThisInst, HINSTANCE hPrev, LPSTR lpCmdLine, int nShowMode)
 {
    WNDCLASSEX wcl;
    MSG msg;
   HWND hwnd;
```

```
   HMENU hMenu;
   HACCEL hAccel;
   wcl. cbSize =sizeof(WNDCLASSEX);
   wcl. hInstance =hThisInst;
   wcl. cbClsExtra =0;
   wcl. cbWndExtra =0;
        wcl. hbrBackground = (HBRUSH)      GetStockObject
(WHITE_BRUSH);
   wcl. hCursor =LoadCursor(NULL, IDC_ARROW);
   wcl. hIcon =LoadIcon(NULL, IDI_WINLOGO);
   wcl. hIconSm =NULL;
   wcl. lpfnWndProc =WindowFunc;
   wcl. lpszClassName =szWinName;
   wcl. lpszMenuName =NULL;
   wcl. style =NULL;
    if(! RegisterClassEx(& wcl)) return 0;
     hMenu =LoadMenu(hThisInst, ″mymenu″);
       hwnd =CreateWindow(szWinName, ″窗口重绘演示″,
WS_OVERLAPPEDWINDOW,
CW_USEDEFAULT,  CW_USEDEFAULT,  CW_USEDEFAULT,
CW_USEDEFAULT,
          HWND_DESKTOP, hMenu, hThisInst, NULL);
      ShowWindow(hwnd, nShowMode);
      UpdateWindow(hwnd);
     hAccel =LoadAccelerators(hThisInst, ″myAccel″);
     while(GetMessage(& msg, hwnd, 0, 0))
     {
      if(! TranslateAccelerator(hwnd, hAccel, & msg))
       {
        TranslateMessage(& msg);
        DispatchMessage(& msg);
        }
       }
     return msg. wParam;
 }
  LRESULT  CALLBACK  WindowFunc(HWND  hwnd,  UINT
message, WPARAM wParam, LPARAM lParam)
 {
  HDC hdc;
  static HDC memDC;
  static HBITMAP hBitmap;
  HBRUSH hBrush;
  PAINTSTRUCT paintstruct;
  static int maxX, maxY;
  int response;
  switch(message)
  {
  case WM_CREATE:
    maxX =GetSystemMetrics(SM_CXSCREEN);
    maxY =GetSystemMetrics(SM_CYSCREEN);
    hdc =GetDC(hwnd);  //得到当前设备描述表
    memDC =CreateCompatibleDC(hdc);
//得到兼容的内存设备描述表
     hBitmap =CreateCompatibleBitmap(hdc,  maxX,  maxY);
//创建兼容位图
    SelectObject(memDC, hBitmap);  //将位图选入内存设备
描述表
hBrush =(HBRUSH) GetStockObject(WHITE_BRUSH); //得
到画刷
SelectObject(memDC, hBrush); //将画刷选入内存设备描述
表
PatBlt(memDC, 0, 0, maxX, maxY,PATCOPY); // 用当前画
刷填充
     ReleaseDC(hwnd, hdc); //释放当前设备描述表
     break;
  case WM_PAINT:
  hdc =BeginPaint(hwnd, & paintstruct);
 BitBlt(hdc, paintstruct. rcPaint. left, paintstruct. rcPaint. top,
   paintstruct. rcPaint. right -paintstruct. rcPaint. left,
   paintstruct. rcPaint. bottom -paintstruct. rcPaint. top,
 memDC, paintstruct. rcPaint. left, paintstruct. rcPaint. top, SR-
CCOPY);
//对需要重绘区域进行重画
     EndPaint(hwnd, & paintstruct);
     break;
  case WM_COMMAND:
     switch(LOWORD(wParam))
     {
     case IDM_DRAW:
   hdc =GetDC(hwnd);
   int x, y, width, height;
   int red, green, blue;
   width =GetSystemMetrics(SM_CXFULLSCREEN); //得到
客户区宽度与高度
    height =GetSystemMetrics(SM_CYFULLSCREEN);
    for(x =0; x <width; x + +)
     for(y =0; y <height; y + +)
    {
    red =x * 255/width; ;
    green =y * 255/height;  //得到不同坐标下的颜色值
blue = ((x * 255)/width +(height -y) * 255/height)/2;
 SetPixel(hdc, x, y, RGB(red, green, blue)); //向物理窗口作
图
     SetPixel(memDC, x, y, RGB(red, green, blue)); //输出到
虚拟窗口
     }
   ReleaseDC(hwnd, hdc);
     break;
     case IDM_CLEAR:
       hdc =GetDC(hwnd);
PatBlt(hdc, 0, 0, maxX, maxY, PATCOPY); //清屏物理窗口
PatBlt(memDC, 0, 0, maxX, maxY,PATCOPY); // 清屏虚拟
窗口
      ReleaseDC(hwnd, hdc);
       break;
     case IDM_EXIT:
      response =MessageBox(hwnd, ″真的要退出吗?″, ″退
出″, MB_YESNO);
       if(response = =IDYES) PostQuitMessage(0);
       break;
     case IDM_HELP:
   MessageBox(hwnd, ″绘制一屏晚霞图, 并演示窗口快速重
绘.″, ″帮助″, MB_OK);
      break;
```

MFC 程序中类之间变量的互相访问

陆宏伟

摘　要　本文主要讨论基于 Doc / View 结构的 MFC 程序中类之间变量的互相访问问题。利用全局 API 函数、各个类的成员函数、静态 static 变量和外部 extern 变量，可以很方便地实现类之间变量的互相访问。

Microsoft 基本类库 MFC 应用框架是一种类库的超集。一般的类库只是一种可以用来嵌入到任何程序中去的孤立的类的集合。但应用框架却定义了程序的结构，它所产生的应用类使用了标准的结构。大多数应用除了需要包含应用和框架窗口类外，一般还要包含文档类和视类。这种文档——视结构可以将数据从用户对数据的观察中分离出来。文档基类通常和 File Open 及 File Save 菜单项相关联，而派生文档类则完成实际的读写操作。视类常用来和文档类联系，负责应用的显示。而类库则协调着文档、视、框架窗口以及应用对象之间的相互作用关系。MFC 通过隐藏 WinMain 函数及构造消息——控制机制来使编程得到简化。MFC 这种封装 Windows 消息处理的机制可以增强程序的安全性和减小程序员工作量，因此 MFC 使得进行 Windows 应用程序的开发工作变得轻松自如。

基于文档——视结构的应用程序，数据一般放在文档类中，而应用的显示由视类负责。以显示图像为例，图像的数据存放在文档类中，要在视类中显示图像则要获得文档类中的数据。如果利用 AppWizard 创建的应用程序则在视类中有一个 GetDocument 的成员函数可以获得文档类的指针，利用文档类的指针就可以获得文档类中的数据。但是 MFC 程序中类之间变量的互相访问并不仅限于此，以多文档界面应用程序为例，程序中主要包含视类、文档类、MDI 子框架类、MDI 父框架类和应用类，另外还有用户生成的其它对话框、对话条等类，如何实现这些类之间的变量的互相访问就是本文讨论的问题。

类之间的变量的互相访问一般采用的方法是利用特定的全局 API 函数和各个类提供的成员函数，如获得应用类指针的 AfxGetApp 函数，获得主框架类指针的 AfxGetMainWnd 函数，获得父类指针的 GetParent 函数等等。另外可以在类中定义静态变量，或者用外部变量来实现类之间的变量的互相访问。下面就具体介绍各种方法。

一、常用的方法

类之间的变量的互相访问一般采用的方法是利用特定的全局 API 函数和各个类提供的成员函数，来获得需要访问变量所在类的指针，通过该指针便可以访问到其中的变量。这里要说明一点：要访问的变量应该是 public 变量。下面针对多文档应用程序来介绍类之间变量的访问。为清楚起见，工程名取为 ClassCom，视类名为 CClassComView，文档类名为

```
    }
    break;
  case WM_DESTROY:
        DeleteDC(memDC);  //释放内存设备描述表
        DeleteObject(hBitmap); //释放位图句柄
        PostQuitMessage(0);
      break;
  case WM_QUIT:
    PostQuitMessage(0);
    break;
 default:
   return DefWindowProc(hwnd, message, wParam, lParam);
   }
  return 0;
}
```

2. 资源文件 resource. rc

```
 #include  <windows. h>
#include ˝wmenu. h˝
MyMenu MENU
{
 POPUP ˝画图(& D)˝
   {
    MENUITEM ˝作 图(& S)\t F2˝, IDM_DRAW
    MENUITEM ˝清 屏(& C)\t F3˝, IDM_CLEAR
    MENUITEM ˝退  出(& X)\t Ctrl +X˝, IDM_EXIT
     }
  POPUP  ˝帮助(& H)˝
    {
     MENUITEM ˝帮助\t F1˝, IDM_HELP
     }
  }
  MyAccel ACCELERATORS
  {
  VK_F2, IDM_DRAW, VIRTKEY
  VK_F3, IDM_CLEAR, VIRTKEY
  ˝^  X˝, IDM_EXIT
  VK_F1, IDM_HELP, VIRTKEY
  }
```

3. 头文件 wmenu. h

```
#define IDM_DRAW    100
#define IDM_EXIT    101
#define IDM_CLEAR   102
#define IDM_HELP    103
```

收稿日期　2000 年 8 月 15 日

CClassComDoc，子框架类名为 CChildFrame，主框架类名为 CMainFrame，应用类名为 CClassComApp。

在获得特定类指针的函数中最主要的两个函数是：获得应用类指针的 AfxGetApp 函数和获得主框架类指针的 AfxGetMainWnd 函数。只要灵活地运用好这两个函数，便基本上可以实现类之间变量的互相访问。具体方法如下所述：

1. 主框架类访问其它类变量的方法

可用函数 AfxGetApp 获得应用类指针，代码如下：

```
CClassComApp * m_lpApp = (CClassComApp * ) AfxGetApp();
```

为了获得当前活动的子框架类指针，主框架类提供了两个函数：MDIGetActive()和 GetActiveFrame()．两种函数调用的代码如下：

```
CChildFrame * m_lpChildFrm1 = (CChildFrame * ) MDIGetActive();
CChildFrame * m_lpChildFrm2 = (CChildFrame * ) GetActiveFrame();
```

获得了子框架类指针，就可以用子框架类提供的两个成员函数 GetActiveDocument()和 GetActiveView()函数来获得当前活动的文档类和视类的指针．代码如下：

```
CClassComDoc * m_lpDoc = (CClassComDoc * ) m_lpChildFrm1 ->GetActiveDocument();
CClassComView * m_lpView = (CClassComView * ) m_lpChildFrm1 ->GetActiveView();
```

2. 应用类访问其它类变量的方法

由上面的介绍可知，只要获得了主框架类的指针，其余子框架类、文档类和视类的指针就可以获得了。一种获得主框架类的指针的方法是用 AfxGetMainWnd 函数，代码如下：

```
CMainFrame * m_lpMainFrm1 = CMainFrame * AfxGetMainWnd
```

另外我们知道应用类中有一个 CWnd * 类型的 m_pMainWnd 变量，用来存放应用程序主框架的指针。因此利用它同样也可以获得主框架类的指针，代码如下：

```
CMainFrame * m_lpMainFrm2 = CMainFrame * m_pMainWnd
```

获得其它类指针的方法同上，这里就不赘述了。

3. 子框架类访问其它类的方法

获得应用类和主框架类指针可以用全局函数 AfxGetApp 和 AfxGetMainWnd，获得文档类和视类的指针可用其成员函数 GetActiveDocument 和 GetActiveView，具体代码同上。

4. 视类访问其它类的方法

获得应用类和主框架类指针可以用全局函数 AfxGetApp 和 AfxGetMainWnd，获得子框架类指针可用其成员函数 GetParent 来完成，代码如下：

```
CChildFrame * m_lpChildFrm1 = (CChildFrame * ) GetParent();
```

获得文档类指针可以间接用子框架类的成员函数 GetActiveDocument 来进行，但是从效率的角度出发最好用视类的成员函数 GetDocument 来完成。

5. 文档类访问其它类的方法

获得应用类和主框架类指针可以用全局函数 AfxGetApp 和 AfxGetMainWnd，获得子框架类指针可通过主框架类成员函数 GetActiveFrame 来间接完成。在许多程序中获得视类的变量是比较重要的，如保存鼠标的位置，选择区域的大小及位置等。由于一个文档类可能对应多个视类的缘故，因此文档类有一个 CPtrList 类型的变量 m_viewList 来保存多个视类的指针，并且提供了两个函数：GetFirstViewPosition 和 GetNextView 来访问对应视类的指针。

下面就给出三种获得视类指针的方法。

方法一：间接用子框架类中的成员函数 GetActiveView 来完成。

方法二：用文档类 GetFirstViewPosition 和 GetNextView 来完成，代码如下

```
POSITION pos = GetFirstViewPosition(); // 获得保存第一个视类指针的位置
CClassComView * m_lpView1 = (CClassComView * ) GetNextView(pos);
//获得第一个视类的指针，位置变量 pos 移向下一个视类指针的位置
```

如果是个单视类的情况，则调用一次 GetNextView 即可获得视类的指针，若是多个视类则可用 while 循环语句配合 MFC 基类 CObject 中的 IsKindOf 来识别出所需要的视类指针。下面给出的一段代码演示了如何获得所需视类指针。

```
CWantView * m_lpWantView; // 想要获得视类的指针变量
CView * lppView;        // 临时的视类指针变量
POSITION pos = GetFirstViewPosition(); // 获得保存第一个视类指针的位置
  while( pos ! = NULL )
  {
 lpView = GetNextView( pos); // 获得第一个视类的指针，
 //位置变量 pos 移向下一个视类指针的位置
 if( pView ->IsKindOf(RUNTIME_CLASS(CWantView)) )
 // 判断是否为指向 CWantView 类的指针
 {
 m_lpWantView = (CWantView * )lpView; // 获得想要的视类指针
 }
  }
```

方法三：上述的两种方法一般都要在需要的时候使用，肯定会降低程序运行的速度。下面给出一种有效的方法。在文档类中定义所需的视类的指针，在相应视类的 ON_WM_CREATE 消息响应函数中对其赋值。代码如下：

```
int CClassComView: : OnCreate(LPCREATESTRUCT lpCreateStruct)
{
 if (CView: : OnCreate(lpCreateStruct) = = -1)
      return -1;
      CClassComDoc * pDoc = GetDocument();
 ASSERT_VALID(pDoc);
 pDoc ->m_lpView = this;
 return 0;
```

}

由于视类在响应 WM_CREATE 消息的时候，窗体已经建立，只是窗体仍是不可见的。这时获得文档类指针的 GetDocument 已经可以使用。因此就可以对文档类中定义所需的视类指针进行赋值。这样在文档类中就可以用该指针获得所需视类中的变量。

提醒一点要在文档类定义头文件中类定义的前面，增加 CVIEW 类的预先说明 class CClassComView 如果没有预先说明 CVIEW 类，编译时会出麻烦，因为不这样做，会引起循环包含。类说明避免不必要的#include 指令，但是它仅适合类指针。

5. 其它类访问上述类的方法

许多常见的应用程序中有导游图，鼠标移动位置图像信息对话条等应用，同样需要文档类的数据，视类鼠标位置、图像滚动大小等信息。综上所述，只要用全局函数 AfxGetApp 和 AfxGetMainWnd 获得了应用类和主框架类指针，就可获得活动的子框架类指针，进而文档类和视类的指针也就可以方便地获得了。下面给出一个运用上面方法的示例，我们知道对于一个多文档的应用程序，如果打开的子窗口比较多，而有时候想把这些窗口一下全部关闭，利用主框架类的 MDIGetActive 函数就可以达到目的。具体的实现方法为：添加一个菜单，将消息映射函数映射到主框架类中，在该消息映射函数中加入如下的代码就可以实现将子窗口全部关闭的功能。

```
CMDIChildWnd *pChild;
while((pChild = MDIGetActive()) != NULL)
{
    pChild->SendMessage(WM_CLOSE);
};
```

上面就是一些类之间变量互相访问常用的方法，注意定义指针变量一定要把相应的类定义头文件包含进来，否则编译视类会出错。

二、利用静态和外部变量实现类间数据互相访问

在 C 语言中，与变量联系的标识符至少有两个属性：存储类和类型。类型也就是常说的数据类型，如 int doulbe 等，而存储类用来决定给对象分配内存的位置 数据段、寄存器、堆或栈 以及它的持续时间，也称为生存期。存储类主要有四个：auto 自动，缺省的 、register 寄存器对象 、static 静态 和 extern 外部 。上述的存储类说明符要放在对象类型说明符之前。

静态 static 变量仅在它所在的函数内部或它本身所处的文件中是永久性变量，在任何函数体外声明的静态变量是全局静态变量，在函数内部声明的静态变量是局部静态变量。外部 extern 变量不仅在自身所处的程序单元中被识别，而且在程序中其它的单元也能被识别。外部变量的存储空间在程序开始运行分配，在程序的整个生存期内一直保留，直到程序结束才消失。因此可以将要访问的变量声明为静态变量或外部变量。

下面就详细介绍实现的方法。

1. 利用外部变量来实现

MFC 应用程序在应用类的实现文件中 cpp 文件中 声明了一个应用类的对象，CClassComApp theApp

因此可以在要访问应用类中变量的 cpp 文件中加入如下对应用类对象的外部声明就可以了。

extern CClassComApp theApp

这样就可以很方便访问应用类中的变量或调用应用类中的成员函数了。如果程序中许多文件中都要访问应用类的变量，也可以将上面的声明放在应用类的头文件中应用类定义之后，因为该应用类的头文件被大多数的文件所包含。这样就不用再在访问应用类中的变量函数所在的文件中声明了，同时程序也比较简洁。在包含应用类头文件的任何函数中可以使用应用类的对象来访问其中的变量。

另外对于多文档应用程序，有时也许会在应用类的 OnIdle 消息响应函数中更新各个视类窗口的显示，因此也可以采用在应用类中定义一个视类指针类型的指针集合类，如有序列表类 CObList 和 CPtrList 指针列表类。在视类的构造函数中将该类的 this 指针加入到定义的集合类的对象中，这样就可以实现全部视类窗口显示内容的更新。具体实现的方法如下：

首先在应用类的 CPP 文件中定义下面的变量：

CTypedPtrList <CObList CClassComView*> GlobalViewList

然后在应用类头文件中将其定义为外部变量：

```
class CClassComView; //预先说明 CClassComView 类
extern CTypedPtrList<CObList, CClassComView *> GlobalViewList;
```

在 CClassComView 类的构造函数中加入如下的代码:

```
GlobalViewList.AddTail(this);
```

在 CClassComView 类的构造函数中加入如下的代码:

```
POSITION pos = GlobalViewList.GetHeadPosition(), oldPos;
while( pos != NULL )
{;
oldPos = pos;
if( GlobalViewList.GetNext(pos) == this )
    GlobalViewList.RemoveAt(oldPos)
}
```

这样在任何想访问视类的变量的地方，就可以用 GlobalViewList 中收集的视类指针来完成了。

2. 利用静态变量来实现

如果在程序中用一个对话条实现了类似 Photoshop 的 Navigator 导游图功能，那么如果图像数据改变了，就要更新对话条中的导游图。由于一般对话条是主框架类的成员变量，因此要获得主框架的指针来访问到对话条对象来完成更新。但是如果将该对话条对象定义为静态变量，那么就可以直接访问该变量了。实现的方法比较简单，首先在对象声明前加入 static 存储类说明符 然后在主框架类的 CPP 文件类实现代码前加入如下的声明。

下转第 19 页)

在 VFP 中调用 MSTTS 技术实现英文语音输出的方法

徐雨明

摘　要　本文介绍了一种在 VFP 中调用微软的 TTS 技术的方法，并通过一个发音实例，说明其设计步骤。

关键词　VFP，MSTTS，OLE 容器控件

一、前言

金山词霸 2000 中即可进行查字典时同时读出相应的英文单词，也可进行全文英文朗读功能，这大大方便了用户的使用，提高了金山词霸的实用性。其实，金山词霸 2000 中就是直接利用了微软的 MSTTS（Text To Speech）技术，即文本语音发声技术。而作为一个程序设计者，你一定也想在自己的程序中加入类似的功能吧，因为那将让自己的程序增色不少。

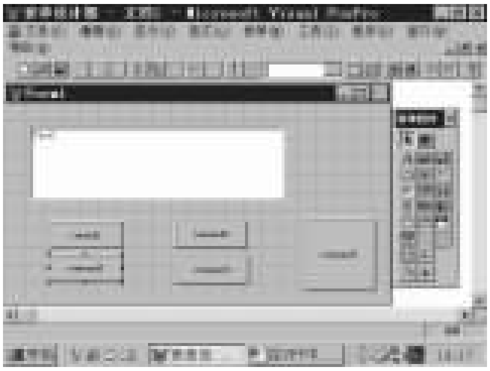
下面，介绍一种在 VFP6.0 中调用微软的 MS Speech API 接口，实现英语单词发音的实例，供有兴趣的电脑爱好者参考。

二、具体实现方法

第一，要在您的电脑中安装 MSTTS 引擎。在金山词霸 2000 的安装过程中，将自动安装 MSTTS；我们也可以利用金山词霸 2000 安装光盘中的 MSTTS.EXE 文件单独安装，在金山词霸 2000 光盘上 hciba 子目录下运行 mstts.exe（MS TTS engine）并安装 spchapi（MS Speech API）。安装成功后，在您系统的 Windows 目录下将有一个 Speech 子目录，其中安装了相应的支持文件。

第二，进入 VFP6.0 执行菜单“文件/新建”，然后单击“文件类型”框中的“项目”选项，然后单出“新建文件”按钮，在弹出的对话框中“项目文件”输入框中输入一个项目文件名，如 MSTTS，最后单击“保存”按钮就完成了第一步工作。这时屏幕上会出现一个“项目管理器”窗口。

第三，在“项目管理器”窗口中选择“文档”中的“表单”，然后单击“新建”按钮后选择“新建表单”按钮。这时会出现一个表单设计器窗口，这时便可开始进行程序界面的设计，建立五个 VFP 常规控件、五个命令按钮和一个文本框。具体如下图：



之后分别修改五个控件的主要属性：

Form1：Caption 属性的值为“微软 TTS 技术的一个例子”

Text1：Value 属性的值为“Hello my friends!”

Command1：

Caption 属性的值为“开始发音”

Enabled 属性的值为 .T.

Command2：

Caption 属性的值为“停止发音”

Enabled 属性的值为 .F.

Command3

Caption 属性的值为“暂停发音”

Enabled 属性的值为 .F.

Command4

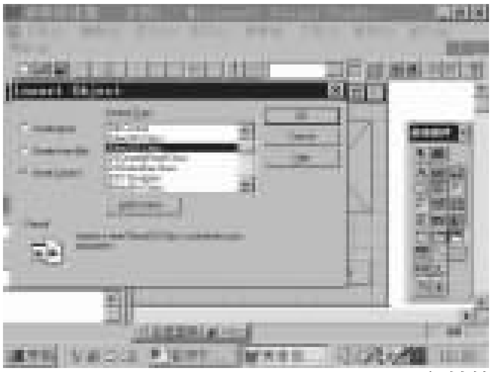
Caption 属性的值为“继续发音”

Enabled 属性的值为 .F.

Command5 Caption 属性的值为“退出系统”

第四，加入 MSTTS 发音 OLE 对象，具体如下：

在表单控件工具框中，选择“OLE 容器控件”，并在表单的 TEXT1 文本框的右边将其放置，这时可能需要一段时间等待（视计算机速度），之后将在屏幕上出现一个“Insert Object”对话框，在对话框中选择“Insert Control”选项（也可能需要一段时间等待），然后将在屏幕上出现一个“Control Type”选择框，在其中选择“DirectSS Class”（最后单击“OK”按钮完成 MSTTS 发音 OLE 对象的插入。如下图所示：



（下转第 24 页）

跟我学用 DELPHI 创建 CGI 应用程序
——通过网页查询 SQL 数据库的实例

宋正荣

关键词　CGI，URL，HTML，SQL link

在互连网刚出现时，它几乎只含有静态网页。这意味着当你选择了一个 URL 时，服务器将返回对应于那个 URL 的 HTML 文档。HTML 也可以含有其他网页的 URL。这样的应用适合传递静态信息，不适合需要交互的对象。例如需要查询某单位的电话号码，在静态方式下需要将所有的电话号码列在网页上，这显然不是一个好办法。使用动态 HTML 结合数据库可以很好地满足需要交互的应用，首先在网页上输入需要查询的单位名称，然后提交给服务器，服务器根据请求到数据库中查找数据，然后根据结果返回数据到客户。

动态 HTML 的关键在于所有处理操作均在服务器上完成，服务器不再只是返回文件的内容，而是根据动态请求定制代码来确定给客户发送什么内容。

DELPHI 可以支持四种类型的服务器进程来产生动态 HTML，分别是 ISAPI、NSAPI、CGI、WIN - CGI，它们各有特点。所幸的是 DELPHI 可以帮助我们维持一个通用的代码基础，将工程编译到 CGI、WIN - CGI、ISAPI、NSAPI 中去。这样我们可以利用 CGI 的便于调试的优点来调写程序，在调试通过后编译成 ISAPI 或 NSAPI 来提高服务器性能。

实例调试环境，一台 NT 服务器安装有 IIS 服务、SQL7 服务、DELPHI5 和 SQL links，并且要求数据库服务器名为 SERVER；创建数据库别名 XXXX，并指向数据库 PUBS；创建 SQL 用户 pt，设置权限为可以查询 PUBS 库数据，无口令。注意，SQL links 别名要在 NT 服务器上建立。

我们可以用 DELPHI 中 sql explorer 工具来创建数据库别名 XXXX，具体步骤如下：

1. 启动 sql explorer
2. 按 <ctrl> + N，添加数据库别名，此时弹出一个 new database alias 窗口，选择 database driver name 为 MSSQL，确定；
3. 将别名 MSSQL1 改为 XXXX；
4. 在 definition 中将 database name 改为 PUBS，将 server name 改为 server，将 user name 改为 pt
5. 点 apply 保存。

这样 DELPHI 就可以通过 Sql links 访问别名为 XXXX 的 SQL 数据库 PUBS 了。

环境建立好以后，可以编程了。

先来看看怎样建立 CGI 应用程序。

1. 启动 DELPHI；
2. 选择 File | New 引出 New Irems 对话框；
3. 在 New 标签中，选择 WEB Server Application，并单击 OK；
4. 此时将提示我们选择类型，选择 CGI stand alone executable 并单击 OK。

这样一个新的 CGI 工程就产生了，下一步给应用程序添加定制逻辑。步骤如下：

1. 将 data access 中 database 和 query 控件放入 components 中，右击 Webmodule1 | Action，弹出一菜单，选择 Add item，添加一个动作；
2. 单击该新行，在 Object inspector / Events 中双击 OnAction 事件，为事件处理器创建原形，然后可以在原形中添加调用 WEB 服务器时应该执行的代码。

```
procedure TWebModule1.WebModule1WebActionItem1Action(Sender: TObject; Request: TWebRequest; Response: TWebResponse; var Handled: Boolean);
begin
  //此处添加代码
end;
```

3. 保存工程，并将工程编译到 exam. exe。
4. 将 exam. exe 放到 WEB 服务器可执行文件目录中，对于 NT4 的 IIS 来说，默认的目录是 c hinetpub hscripts。

至此，一个完整的 CGI 应用程序已经创建完毕，但它是空的，没有功能。现在我们需要给它加上相应的功能。我们假定需要查询 PUBS 数据库中 authors 表中作者电话的数据，查询者通过浏览网页，输入作者名字 au_lname，然后点击查询按钮，将数据提交给 WEB 服务器，WEB 服务器根据浏览者提交的数据完成提取参数、查询数据、返回动态 HTML 的操作。

参数是通过网页提交到 WEB 服务器的，我们还得制作一个查询网页，本文给出了一个简单的 HTML 页面，将产生一个输入框和一个按钮。将它命名为 exam. htm，并保存在服务器的默认目录 c hinetpub hwwwroot 中。

HTML 源代码：

```
<html>
<head>
<meta http-equiv="Content-Type"
content="text/html; charset=gb_2312-80">
<meta name="GENERATOR" content="Microsoft Front-
```

```
Page Express 2.0">
<title></title>
</head>
<body>
<p align="center"><font color="#0080C0" size="6"
face="楷体_GB2312"><b>跟我学用 DELPHI 写 CGI 应用程
序</b></font></p>
<form  action="/scripts/exam.exe"  method="  GET"
name="exam">
  <p  align="center"> 作 者 名 字 : <input  type=" text"
size="11"
  name="lname"></p>
  <p align="center"><input type="submit" name="B1"
  value="查询"></p>
</form>
</body>
</html>
```

在这个例子中，lname 变量将被提交给服务器，其中含有所键入的数据。然后将数据传递给 ACTION 中指定的 CGI 服务器应用程序。上例是传递给 /scripts/exam.exe。

数据传递到 CGI 应用程序后，我们需要从中提取我们需要的数据，在提取数据时，需要区别对待不同的数据传递方法，在 exam.htm 中采用的是 GET 方法，那么在取数据时也要用 GET 方法。将以下代码填入，代码行意义见注释，然后重新编译。

```
var
 lname: string;  //输入数据存放的变量
 rtlname, rtphone: string;  //存放 SQL 查询操作返回的结果
begin
 //配置数据环境, 利用 SQL LINKS 访问数据库 PUBS
 database1.connected: =false;     //关闭数据库连接
database1.aliasname: ='xxxx';  //设置数据库别名为 XXXX
database1.databasename: ='pubs';  //设置要访问的数据库
database1.loginprompt: =false;  //将登录方式设成程序登录
 database1.params.clear;      //配置登录需要的参数
 database1.params.Add('USER NAME=pt');
 database1.params.Add('PASSWORD=');
 //根据不同的传递方法用不同的方法取数据
 case request.methodtype of
  mtPost:  //POST 方法从 request.contentfields 取数据, 因
为在 exam.htm 中, 我们定义了编辑框的名字为 lname, 提交参
数时格式是 lname=XXXXXX 这样的, 所以我们从第 7 个字符
开始取(总长度 -6) 个字符. 如果编辑框的名字改变了, 这些参
数要做相应调整.
   begin
     lname: =copy(request.contentfields.strings[0],  7,
length(request.contentfields.strings[0]) -6);
   end;
  mtGet: //GET 方法从 request.QueryFields 取数据
   begin
     lname: =copy(request.QueryFields.strings[0],  7,
length(request.QueryFields.strings[0]) -6);
   end;
 end;
 database1.connected: =true; //连接数据库
 query1.databasename: ='pubs';  //设置 query 从 pubs 中
查找数据
 query1.close;  //先关闭 query, 并清除 SQL 语句
 query1.sql.clear;
  query1.sql.add( ' select  au_lname,  phone  from  authors
where au_lname="' +lname+'"'); //将所需要的 SQL 语
句输入
 query1.open; //执行 SQL 查询
   rtlname: =trimright(Query1.FieldByName(  '  au_lname')
.AsString); //取得查询结果
    rtphone: =trimright(Query1.FieldByName(   '   phone')
.AsString);
 if rtlname  = lname then
   begin
       response.content: =' <HTML>' +rtlname+':' +rt-
phone+' <HTML>'; //返回动态 HTML, 这个例子中没有修
饰成分, 返回结果的页面比较简单
   end
 else
   response.content: =' <HTML>' +' 无 此 部 门!' +' </
HTML>';
 query1.close; //关闭 query 和 databaase 连接
 database1.connected: =false;
end;
```

在编译通过后，将 exam.exe 拷贝到 IIS 默认的 c hinet-pub hscripts 目录中，再将 exam.htm 拷贝到 IIS 默认的 c hinet-pub hwwwroot 目录中，就可以从浏览器中查询作者的电话号码了。

参考文献

1. 《DELPHI3 自学通》. 机械工业出版社
2. 《DELPHI3 编程指南》. 宇航出版社

（收稿日期：2000 年 10 月 8 日）

（上接第 16 页）

int CMainFrame m_MainStatic

由于全局静态变量要在程序开始运行时分配，因此要在 CPP 文件中进行上面的声明。注意该被访问的变量一定要是 public 变量，否则其它类将无权访问。另外如果在多文档程序中的子框架类、文档类和视类中定义静态变量时，应该明白的是：对于多个类的对象，该变量实际上对应的是同一块内存，即该变量对于多个类的对象是共享的、唯一的。

利用静态和外部变量实现类间数据互相访问的方法，对于一些数据较多的应用程序实现数据的合理存放、管理和访问极为有效。

上面简单介绍了 MFC 程序中类之间变量互相访问的的方法，相信只要灵活运用 MFC 函数和 C++ 语言，就一定能编制出高效率的 Windows 应用程序。

（收稿日期：2000 年 8 月 18 日）

用 OOP 技术实现一类不可预测的分形屏保技巧

李夕海

摘 要 根据分形理论的一个分支—复动力系统图形生成的特点，本文利用面向对象的编程 OOP 语言 VC++实现了对它的编程，产生了一类具有不可预测性的分形屏保，总结出了这一类屏保的编程技巧。

关键词 分形，复动力系统，OOP

一、引言

随着计算机的日益普及，人们越来越感觉到计算机已成为生活中不可缺少的部分，而屏幕保护（简称屏保）也逐渐成为人们感兴趣的话题。绝大部分人都想拥有一个令自己赏心悦目的屏保，一方面可以避免长时间显示静止画面而损坏荧光屏，另一方面也能使自己从中得到乐趣。但是目前我们所看到的绝大多数屏保在短时间内都不可避免地重复原有的东西，时间稍久一点，就觉得有点枯燥。能不能编出一类屏保，使其出现的图形人为不可预测呢。本文考虑了分形这一新兴学科的一个分支——复动力系统，根据它能产生一类分形图形——Julia 集可具有不可预测性的特点，利用面向对象的编程语言 Vc++6.0，实现了这类屏幕保护程序的编程，总结出了一些技巧。

二、实现原理

分形是新发展起来的数学分支，是研究自然界中复杂现象（即没有特征长度而又具有自相似性的形状和现象）的强有力工具。复动力系统是它的一个分支，下面首先了解其原理。

本文所用到的复动力系统是在复平面中，其迭代式为 $Z_{n+1}=Z_n^2+c$，其中，Z 和 C 都是复数，由各自的实部和虚部组成。其中，Z 的实部和虚部为：

$$x_{n+1}=x_n^2-y_n^2+c$$

$$y_{n+1}=2x_ny_n+c_y$$

c=a+ib 这个迭代将会导致我们所要生成的 Julia 集。具体产生过程为：在迭代的过程中，复数 C 是固定的，让 Z 作为原始点通过迭代而生成。选定一个 C 值，就可以在复平面上生成一幅不同的 Julia 集的计算机图像。若让 C 在复平面的某个区域内随机变化，则每一个 C 值，就相应产生一幅 Julia 分形图形。由于 C 取值是随机的，因此，对于下一次出现的图形是不可预测的。

三、实现技术

根据上述分形复动力系统产生分形图形的原理，本文利用 VC++6.0 实现了对它的屏保编程。由于 VC++的 MFC 已经封装了屏幕保护程序，因此下面介绍用 MFC 来开发这一类屏保的技巧。

3.1 创建一个工程（project）

使用 AppWizard 创建一个基于对话框的名为 FractalScreenSaverDlg 的应用程序。打开对话框编辑器，加入一静态文本框，用于预览生成的分形图形。

3.2 增加 CDrawFractalWnd 类用以生成随机分形图形

打开 Insert 菜单，利用 New Class 项来插入一新类 CDrawFractalWnd，其主要功能是处理一些典型的消息，并完成 Julia 集的绘制。

1 插入新类时，要选定 MFC Class 并把 generic CWnd 作为基类。

2 在头文件中，加入以下变量和函数声明

```
////////////////变量声明////////////
// Attributes
Public:
static LPCTSTR m_lpszClassName; //指向类名的指针
double x[200], y[200];
   int Total;
   float Bx, By;
   double p, q; //复动力系统常数 C 的实部和虚部
   float  scale, m;
   int  drawtrue, value;
   DWORD m_pcolor; //用于获得颜色
   double mag;
   CRect m_nSize;
////////////////函数声明///////////
void Draw(CDC & dc); //用于生成分形图形
double MapRand(double nMax); //用于产生随即数
```

3 在源文件中加入以下头文件及常量声明

```
#include <stdlib.h>
#include <time.h>
#include <math.h>
#include <stdio.h>
#define attract  0.0001   //吸引子灵敏度
#define MaxColor 16 //最大颜色数
#define  zoom 2.0
#define MaxY  800
```

4 在源文件的构造函数前及内部加入以下初始化变量

```
LPCTSTR CDrawWnd::m_lpszClassName=NULL;
CDrawFractalWnd::CDrawFractalWnd()
{
          p=0.576; //复常数初值
```

```
            q =0. 09;
            value =50;
}
```

5 在头文件和源文件中重载 Creat 函数如下

在头文件中：

```
virtual BOOL Create(DWORD dwExStyle, DWORD dwStyle, const RECT& rect, CWnd * pParentWnd, UINT nID, CCreateContext * pContext =NULL );
```

在源文件中

```
BOOL CDrawWnd: : Create(DWORD dwExStyle, DWORD dwStyle, const RECT& rect, CWnd * pParentWnd, UINT nID, CCreateContext * pContext)
{     // Register a class with no cursor
if (m_lpszClassName = = NULL)
{
m_lpszClassName =AfxRegisterWndClass(CS_HREDRAW | CS_VREDRAW, : : LoadCursor(AfxGetResourceHandle( ), MAKEINTRESOURCE(IDC_EMPTYCURSOR)));
}
m_nSize =rect;
return CreateEx(dwExStyle, m_lpszClassName, _T(""), dwStyle, rect. left, rect. top, rect. right - rect. left, rect. bottom - rect. top, pParentWnd ->GetSafeHwnd( ), NULL, NULL ); }
```

其中 IDC_EMPTYCURSOR 为自定义空光标。

6 增加 WM_PAINT 和 WM_TIMER 消息处理函数和产生随机数函数

```
void CDrawFractalWnd: : OnPaint(CDC & dc)//绘制分形图形
{
CPaintDC dc(this); // device context for painting
        int i, j; int k;
     double x1 =MapRand(2);
          double x2 =MapRand(2);
          scale =float(2. 0* zoom/MaxY); //坐标转换
     for (i =0; i <MaxY; i + +)
          for(j =0; j <MaxY; j + +)
                 {
     x[0] =float(scale * i -zoom); //初始值变换
     y[0] =float(zoom -scale * j);
     k =0;
     mag =0;
      while((mag <100)& & (k <MaxColor * 2))
{
     x[k +1] =x[k] * x[k] -y[k] * y[k] +p; //复数的平方
     y[k +1] =2 * x[k] * y[k] +q;
     mag = x[k +1] * x[k +1] +y[k +1] * y[k +1];
//复数模的平方
          k =k +1;
          }
    if((k = =MaxColor * 2)||(mag>100))
//发散速度不同用不同的颜色绘制
    dc. SetPixel(i , (j -150), RGB(int(k * value/x1), int(k * value), int(k * value/x2)));
     }}
void CDrawFractalWnd: : OnTimer(UINT nIDEvent) //利用定时器来随机产生常数 C 的值
{
if(MapRand(1) < =0. 5)
    {
      p =double(MapRand(2)/1. 034);
        if(MapRand(1) <0. 5)
              q =double(MapRand(2)/1. 0834);
        else if(MapRand(1)>=0. 5)
        { q =double(MapRand(2)/1. 0345);
         q = -q; }
    }
    else if(MapRand(1)>0. 5)
    {     p =double(MapRand(2)/1. 034);
       p = -p;
        if(MapRand(1) <0. 5)
        q =double(MapRand(2)/1. 0834);
        else if(MapRand(1)>=0. 5)
        { q =double(MapRand(2)/1. 0345);
         q = -q; }
    }}
Invalidate();
CWnd: : OnTimer(nIDEvent);
}
double CDrawFractalWnd: : MapRand(double nMax)
//产生随机数
{
 int nRand  =rand();
 double fMap =double(nRand)/RAND_MAX;
 double nRetVal =fMap * nMax +0. 5f;
 return  nRetVal; }
```

3. 3 创建新类 CscreenEventWnd

新类继承 CdrawFractakWnd 类，主要负责处理当检测到鼠标、键盘消息时，关闭当前屏保。其编程和其它类型的屏保一样,这里不再赘述,只给出必要的源代码。

```
BOOL CScreenEventWnd: : Create()
{
CRect rect(0, 0, : : GetSystemMetrics(SM_CXSCREEN), : : GetSystemMetrics(SM_CYSCREEN));
return CDrawFractalWnd: : Create(WS_EX_TOPMOST, WS_VISIBLE|WS_POPUP, rect, NULL, 0, NULL);
}
void CScreenEventWnd: : OnActivate(UINT nState, CWnd * pWndOther, BOOL bMinimized)
{if(nState = =WA_INACTIVE) PostMessage(WM_CLOSE);
CDrawWnd: : OnActivate(nState, pWndOther, bMinimized);
}
void CScreenEventWnd: : OnActivateApp(BOOL bActive, HTASK hTask)
{
  if(! bActive) PostMessage(WM_CLOSE);
  CDrawWnd: : OnActivateApp(bActive, hTask);
}
void CScreenWnd: : OnDestroy()
{
PostQuitMessage(0);
CDrawFractalWnd: : OnDestroy();
}
void CScreenEventWnd: : OnKeyDown(UINT nChar, UINT nRepCnt, UINT nFlags)
{
  PostMessage(WM_CLOSE);
```

```
CDrawFractalWnd::OnKeyDown(nChar, nRepCnt, nFlags);
}
void CScreenEventWnd::OnLButtonDown(UINT nFlags,
CPoint point)
{
    PostMessage(WM_CLOSE);
    CDrawFractalWnd::OnLButtonDown(nFlags, point);
}
void CScreenEventWnd::OnMouseMove(UINT nFlags,
CPoint point)
{
    if(m_ptLast==CPoint(-1, -1))m_ptLast=point;
    else if (m_ptLast!=point)PostMessage(WM_CLOSE);
    CDrawFractalWnd::OnMouseMove(nFlags, point);
}
BOOL CScreenEventWnd::OnNcActivate(BOOL bActive)
{
    if(!bActive) return FALSE;
    return CDrawFractalWnd::OnNcActivate(bActive);
}
void CScreenEventWnd::OnRButtonDown(UINT nFlags,
CPoint point)
{
    PostMessage(WM_CLOSE);
    CDrawFractalWnd::OnRButtonDown(nFlags, point);
}
BOOL CScreenEventWnd::OnSetCursor(CWnd * pWnd,
UINT nHitTest, UINT message)
{
    SetCursor(NULL);
  return CDrawFractalWnd::OnSetCursor(pWnd, nHitTest,
message);
}
void CScreenEventWnd::OnSysCommand(UINT nID,
LPARAM lParam)
{
if ((nID == SC_SCREENSAVE) || (nID == SC_CLOSE))
    return;
    CDrawFractalWnd::OnSysCommand(nID, lParam);
}
void CScreenEventWnd::OnSysKeyDown(UINT nChar, UINT
nRepCnt, UINT nFlags)
{
    PostMessage(WM_CLOSE);
CDrawFractalWnd::OnSysKeyDown(nChar, nRepCnt, nFlags);
}
```

3.4 实现预览功能

在 FractalScreenSaverDlg.h 处增加 # include "DrawFractalWnd.h"，以支持预览。

接着修改初始化函数 OnInitDialog，增加对预览窗口的显示。修改的代码如下：

```
    CRect rect;
   GetDlgItem(IDC_PREVIEW)->GetWindowRect(&rect);
    ScreenToClient(&rect);
m_wndPreview.Create(NULL, WS_VISIBLE | WS_CHILD,
rect, this, NULL);
CenterWindow();
```

3.5 最后一击

修改 CFractalScreenSaverApp 类，使其支持我们所加功能。

1 增加#include 语句：

```
#include "DrawFactalWnd.h"
#include"ScreenEventWnd.h"
```

2 增加辅助成员函数，用于打开屏保的设置对话框，代码如下：

```
BOOL CFractalScreenSaverApp::MatchOption(LPTSTR lpsz,
LPTSTR lpszOption)
{}
  if (lpsz[0] == '-' || lpsz[0] == '/') lpsz++;
  if (lstrcmpi(lpsz, lpszOption) == 0
   return TRUE;
  return FALSE; }
void CFractalScreenSaverApp::DoConfig()
{ CScreenSaveProgrameDlg dlg;
m_pMainWnd = &dlg;
dlg.DoModal() ;
}
```

3 修改 InitInstance 函数，修改后的代码如下：

```
  ////代码开始//////////
SetRegistryKey(_T("MFC Fractal Screen Savers Inc."));
if (__argc == 1 || MatchOption(__argv[1], _T("c")))
  DoConfig();
  else if (MatchOption(__argv[1], _T("p")))
  {
  CWnd * pParent = CWnd::FromHandle((HWND) atol
(__argv[2]));
  ASSERT(pParent != NULL);
  CDrawFractalWnd* pWnd = new CDrawWnd();
  CRect rect;
  pParent->GetClientRect(&rect);
pWnd->Create(NULL, WS_VISIBLE | WS_CHILD, rect,
pParent, NULL);
      m_pMainWnd = pWnd;
      return TRUE;
  }
  else if (MatchOption(__argv[1], _T("s")))
  {
  CScreenEventWnd* pWnd = new CScreenEventWnd;
      pWnd->Create();
      m_pMainWnd = pWnd;
      return TRUE;
  }
```

四、结束语

至此，编程工作已完成，但由于屏幕保护程序的后缀为 .scr，因此，打开 Project 菜单的 Setting 项，在 Link 选项将 Output file name 的后缀改为 .scr。编译、连接程序，就可产生分形屏保了。若想欣赏这难以预测的分形屏保，只需把执行文件 FractalScreenSaverDlg.scr 复制到 Windows 的 System 中，在显示中选定自己的屏保就可以了。

收稿日期 2000 年 10 月 10 日)

利用可视化方法定制数据窗口格式

孙爱东　张治坤

一、 引言

PowerBuilder 是一个功能强大的客户/服务器型第四代语言开发环境 它提供了丰富的对象和控件。其中 用户自定义对象是拓展 PowerBuilder 功能的最灵活的对象之一。利用自定义用户对象 我们既可以扩展系统原有对象的功能 增加新的使用方法 又能够创建出高度可重用的自定义事件 在一个或多个应用程序中反复试用。PowerBuilder 的用户对象分可是用户对象和类用户对象 可视用户对象可完成应用程序和用户之间的交流 而 PowerBuilder 中的数据窗口表现风格丰富 可操作能力强 因此通过用户自定义对象和数据窗口的结合 可以进一步丰富应用程序界面以及应用程序功能。

本文介绍了用户自定义对象在表格显示格式方面的应用 使用户能自由取舍表的某些列 并且列在数据窗口的位置及其大小都可以通过可视的方式由用户自由定义 做到 '所见即所得" 提高了界面的友好性。

二、 问题的提出

在工资管理信息系统中 很多表的列是需要用户自己定义的 因此应用系统中大部分数据窗口需要动态生成。工资表可能包含的项目很多 并不一定需要输出全部列。列的取舍和在数据窗口中的排列顺序都要由用户定义。用表的方法由用户定义固然可行 但是那样用户是用数字的大小顺序等定义工资条表达方式 如果能用可视的方法实现对数据窗口的定义 将大大提高易操作性。

三、实现方法

定义一个窗口，在窗口中放一个 datawindow 该 datawindow 列出工资条可以包含的列 通过拖拉的方式 从该 datawindow 中把选取的列放到窗口中 为了标识拖放的位置 在窗口上放置一个矩形框 窗口的 dragdrop 事件中包含如下代码

```
li_debug = OpenUserObjectWithParm(uo_place1, parm , i, j)li_debug = OpenUserObjectWithParm(uo_place2, parm , i, j + 70)
```

其中的 uo_place1 和 uo_place2 是可视的用户自定义对象 uo_1 的两个实例，分别代表该列在数据窗口中的列名称 (Text）和列对象 (Column），该对象的属性包含了用户选择的列的一些属性 包括代表该列的自定义对象的位置、该列所代表的列名或者表头文本的名字，该对象在窗口的实例变量：自定义对象数组中的位置 index 。这些参数是通过一个结构 u_dyn_struct 的实例 parm 传递给新建的自定义对象的。

定制的用户自定义对象 uo_1 包含了一个三个静态文本控件：st_text、st_horizontal、st_vertical。其中的 st_text 用来显示该对象所代表的列名称或者列对象标识，另两个静态文本控件用来实现对象的大小调节，用户通过按住鼠标左键拖动可以改变静态文本控件 st_text 和自定义对象本身的高度和宽度。uo_1 创建时从 parm 得到它所代表的列标题文本或列对象的属性，作为自己的实例变量，还有两个实例变量用来保存用户定义的该对象的宽度和高度。另外，当用户点击了该对象时，它便显示边框，成为当前操作对象，用户可以移动、改变大小，因此在 uo_1 的自定义事件 lbuttondown 中写如下代码：

```
iw_parent. iu_being_dragged = this
this. Border = True
```

uo_1 的实例变量 iw_parent 就是放置对象实例的窗口，它在自定义对象实例创建时通过结构 parm 传递过来，parmiu_being_dragged 是窗口的实例变量，它是 uo_1 的实例。

为了使用户能通过鼠标拖动对象或者通过键盘箭头键移动对象，在窗口的 DragDrop 事件中要加如下代码：

```
ldo_drag = DraggedObject()
if TypeOf(ldo_drag) = Userobject! then
iu_being_dragged. il_x = iu_being_dragged. X - r_1. X
  iu_being_dragged. il_y = iu_being_dragged. Y - r_1. Y
  li_height = iu_being_dragged. Height
  li_width = iu_being_dragged. Width
  iu_being_dragged. Move(i - li_width/2, j - li_height/2)
end if
```

在窗口的 Key 事件里加如下代码 以按向左箭头为例）：

```
if KeyDown(KeyLeftArrow!) and TypeOf(iu_being_dragged)
 = UserObject! Then
  li_x = iu_being_dragged. X
  li_y = iu_being_dragged. Y
  iu_being_dragged. il_x = li_x - r_1. X - 5
  iu_being_dragged. Move(li_x -5, li_y)
```

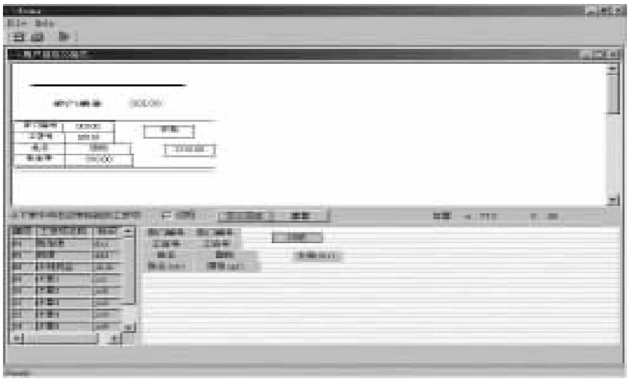
以上两段代码将用户改变后的自定义对象的位置及大小保存到对象的实例变量中去。代码中的 r_1 是为了定位各对象的位置而设。

到这里，用户已经定义好了所需要显示的列和列在数据窗口中的显示位置分布，下一步就是如何将这些代表列标题文本对象和列对象的用户自定义对象的一些性质反应到数据窗口中去。该数据窗口是自由格式的，置所有列对象和文本对象的 visible 属性为 0，使之不可见，然后根据上述用户选择的列置各个对象的属性。这些功能通过如下代码实现：

定义窗口的实例变量数组 iu_obj_name[ ] iu_obj_format[ ]，它们是自定义对象 uo_1 的实例，分别代表列标题文本对象数组和列对象数组。

```
ll_col_selected = UpperBound(iu_obj_name[ ])        //对列名称文本对象的调整
For ll_col_index = 1 to ll_col_selected
    ls_modify = iu_obj_name[ll_col_index] . is_as_string + ´_t. Text = ˝ ´ &
 + iu_obj_name[ll_col_index]. is_object_name + ´˝ ´          &
 + iu_obj_name[ll_col_index]. is_as_string + ´_t. X = ˝ ´  &
 + string(iu_obj_name[ll_col_index]. il_x) + ´˝ ´          &
 + iu_obj_name[ll_col_index]. is_as_string + ´_t. Y = ˝ ´  &
 + string(iu_obj_name[ll_col_index]. il_y) + ´˝ ´          &
 + iu_obj_name[ll_col_index]. is_as_string + ´_t. Width = ˝ ´ &
 + string(iu_obj_name[ll_col_index]. il_w) + ´˝ ´&
 + iu_obj_name[ll_col_index]. is_as_string + ´_t. Height = ˝ ´&
 + string(iu_obj_name[ll_col_index]. il_h) + ´˝ ´        &
 + iu_obj_name[ll_col_index]. is_as_string + ´_t. Visible = ˝1˝ ´ &
ls_modify_error[ll_col_index] = dw_1. Modify(ls_modify)
Next
ll_col_selected = UpperBound(iu_obj_format[])     //对列对象的调整
For ll_col_index = 1 to ll_col_selected
ls_modify = iu_obj_format[ll_col_index]. is_as_string + ´. X = ˝ ´ &
 + string(iu_obj_format[ll_col_index]. il_x) + ´˝ ´&
 + iu_obj_format[ll_col_index]. is_as_string + ´. Y = ˝ ´ &
 + string(iu_obj_format[ll_col_index]. il_y) + ´˝ ´          &
 + iu_obj_format[ll_col_index]. is_as_string + ´. Width = ˝ ´ &
 + string(iu_obj_format[ll_col_index]. il_w) + ´˝ ´          &
 + iu_obj_format[ll_col_index]. is_as_string + ´. Height = ˝ ´ &
 + string(iu_obj_format[ll_col_index]. il_h) + ´˝ ´          &
 + iu_obj_format[ll_col_index]. is_as_string + ´. Visible = ˝1˝ ´ &
  ls_modify_error[ll_col_index] = dw_1. Modify(ls_modify)
Next
```

自定义对象的实例变量 is_as_string 是列名，il_x、il_y、il_w、il_h 分别代表该对象在窗口中的位置坐标和宽度和高度，这样就能按照用户的意图改变数据窗口中这些对象的属性。用户所得到的数据窗口就是他所定义的内容。



收稿日期：2000 年 10 月 10 日)

上接第 17 页)

第五， 编写程序代码：

1. 双击 ‘开始发音”控件，在弹出的 Command1_Click 代码窗口中输入如下代码：

```
* 如果 text1 中有汉字  将有错误
* 开始朗读
thisform. olecontrol1. speak(thisform. text1. text)
thisform. command1. enabled =. t.
thisform. command2. enabled =. t.
thisform. command3. enabled =. t.
thisform. command4. enabled =. f.
```

2. 双击 ‘停止发音”控件，在弹出的 Command2_Click 代码窗口中输入如下代码：

```
*停止发音
thisform. olecontrol1. AudioReset
thisform. command1. enabled =. t.
thisform. command2. enabled =. f.
thisform. command3. enabled =. f.
thisform. command4. enabled =. f.
```

3. 双击 ‘暂停发音”控件，在弹出的 Command3_Click 代码窗口中输入如下代码：

```
*暂停发音
thisform. olecontrol1. AudioPause
thisform. command1. enabled =. t.
thisform. command2. enabled =. f.
thisform. command3. enabled =. f.
thisform. command4. enabled =. t.
```

4. 双击 ‘继续发音”控件，在弹出的 Command3_Click 代码窗口中输入如下代码：

```
*继续发音
thisform. olecontrol1. AudioResume
thisform. command1. enabled =. t.
thisform. command2. enabled =. t.
thisform. command3. enabled =. t.
thisform. command4. enabled =. f.
```

5. 双击 ‘退出系统”控件，在弹出的 Command4_Click 代码窗口中输入如下代码：

```
thisform. release
```

至此，已完成所有工作，点击 VFP 表单的执行表单按钮，运行上述程序，并在 Text1 中输入英文单词或句子 然后单击 ‘开始发音”按钮 即可实现朗读。

三、结束语

微软的 MSTTS 发音技术至目前还不支持中文发音（TTS），所以本文介绍的方法，在 Text1 文本输入框中只能输入英文单词或句子，不能输入中文，否则将发生错误，希望大家注意。

注 本程序在 Win98、VFP6. 0 环境中调试通过。

收稿日期 2000 年 9 月 25 日)

LZW 数据无损压缩算法的 C + + 实现

王俊蛟

摘　要　本文简要介绍了一种非常巧妙且非常简洁易读的数据无损压缩算法 LZW，并采用 C + + / OOP 编程技术完整实现了该压缩算法，提供了多种函数调用接口。本文代码采用了哈希表检索、函数闭包、自释放动态数组等多种编程技术，代码执行速度很快，而且思想清晰明了，很容易读懂，不仅可以帮助读者迅速理解该压缩算法的原理，还可以作为读者在 C + + 编程方面的参考资料。

关键词　压缩，算法，C + +，LZW

一、引言

目前，数据压缩技术已广泛应用于各种软件产品、技术规范、影像格式等，如我们每天都可能要使用 Winzip / ZipMagic / Arj 等压缩解压缩工具，浏览 gif / jpeg 压缩格式图片，或听 MP3 压缩音乐，甚至在您自己开发的产品中也包含了压缩技术，可见压缩技术应用是非常广泛的，特别是目前因特网如此拥挤的情况下，压缩技术更是必不可少；试想当我们下载未经压缩的 800 * 600 - 24b - BMP 位图影像，将不得不下载 1. 37MB 数据，如果将其压缩为 jpg 格式，我们也许只需要下载 100KB 左右数据即可。看来，如果没有广泛采用压缩技术，眼前的因特网不知会阻塞到何种程度

二、LZW 压缩算法

目前较成熟的数据压缩技术有许多种，主要分为无损压缩和有损压缩。其中较典型的无损压缩技术有 LZ77、LZ78 及其变种 LZO、LZH、LZW、ZLIB 等，有损压缩技术的代表则有 JPEG、MPEG、MP3 等。

本文将要介绍的是 LZW 数据无损压缩技术。LZW 压缩技术也是目前应用较广的一种压缩技术，著名的图像压缩格式 GIF98a 就是采用了 LZW 算法。该算法可以说是目前压缩算法中最简洁的一种，压缩算法描述仅 12 行即可，解压缩的算法描述也仅需要 15 行！这真令人吃惊。

LZW 压缩算法描述如下：

```
STRING = get input character
WHILE there are still input characters DO
    CHARACTER = get input character
    IF STRING + CHARACTER is in the string table then
        STRING = STRING + CHARACTER
    ELSE
        Output the code for STRING
        add STRING + CHARACTER to the string table
        STRING = CHARACTER
    END OF IF
END OF WHILE
output the code for STRING
```

LZW 解压缩算法描述如下：

```
read OLD_CODE
output OLD_CODE
WHILE there are still input character DO
  read NEW_CODE
  IF NEW_CODE is not in the translation table THEN
      STRING = get translation of OLD_CODE
      STRING = STRING + CHARACTER
  ELSE
      STRING = get translation of NEW_CODE
  END OF IF
  output STRING
  CHARACTER =first character in STRING
  add OLD_CODE + CHARACTER to the translation table
  OLD_CODE = NEW_CODE
END OF WHILE
```

根据以上算法，不难看出，LZW 压缩数据时，将由程序维护一个字节串表，该表用于存储各不相同的字节串，其中每个字节串都由一个代码标识，当输入流中再次出现前面已出现过的字节串时，便可用相应代码来替代这个字节串，从而实现数据压缩；当解压缩数据时，则执行相反的过程，将输入流中的代码翻译为实际字节串，从而实现解压缩。

LZW 压缩算法虽然寥寥十几行，但真要完整实现 LZW 算法，其代码量却会远远超出十几行，比想象中的要复杂一些，这种复杂性主要来自于数据流输入 / 输出 Stream I / O 、字节串表 String Table 的有效存储及提高字节串表检索速度方面的复杂性。本文将重点分析实现 LZW 算法中的几个难点，同时本文所附源代码涉及到 C + +类 class 、类模板 Class Templates 及位元级操作 Bit Level ，因此，本文假设读者已具备了一定的 C + +编程经验及初步的位元级编程能力。

三、LZW 压缩算法分析

LZW 压缩算法很简洁，但要完整实现其算法，尚需要解决以下方面的问题，才能使该算法真正有效。

1. 字节串表 String Table 的设计与实现

由 LZW 算法描述不难得知，需要一个字节串表 Table 来存储压缩 / 解压缩过程中的字节串，其容量为：

（1 < <lzw_bits）[其中 lzw_bits 代表压缩输出代码的位元数]

因为每输入一个字节，都需要在字节串表中查找其是否已存在；按 LZW 算法推算，也不难发现添加到表中的每个字节串长度是不固定的，至少是 2 个字节，也有可能是 3 个、4 个... 可见要直接存储这个变长字节串表是不易实现的，而且动态内存分配管理过程会明显降低程序执行效率，因此必须想办法进行简化。

经分析 LZW 算法，发现这些字节串有一个共性，即每个字节串都是由前一个代码所表示的字节串与刚输入的一个字节组成，为此，我们可以根据这一特点将不定长的字节串表转化为固定长度的元素表，即元素{prefix_code character}组成的表，同时将算法描述稍作简化即可。

字节串表现已设计成功，当需要查找时，便可很简便地检索一个线性表即可。

2. 字节串 /位串的输入与输出操作

当进行压缩 / 解压缩时，我们可以采用流式输入 / 输出操作，一次处理一个编码，这很符合 LZW 算法的特点。但压缩前输入的代码长度为 8 位，压缩后的数据将会有用更多位的代码 如 9 位 ... 16 位 进行输出；解压缩时的情况恰好相反。

由于输入 / 输出的过程不象整个字节输入 / 输出那样简单，必须提供位元级的操作函数才能较好分离出输入 /输出模块，为了使算法本身清晰明了，本文代码中将这两个过程均封装成 C + + 类 TInputBuffer 和 TOutpuBuffer ，并提供了简洁的接口。这种封装在一定程度上降低了执行效率，但经实践发现，这对程序整体效率影响不算大，但在程序可读性方面得到了较大提高。

3. 如何有效提高字节串表的检索速度

当 lzw_bits 设置为 12，字节串表的长度为 1 < < 12 = 4096 ，当直接检索这个字节串表时，发现代码执行速度极慢，其时间复杂度为 O n2 ，简直无法忍受。为此，代码中不得不引入哈希表，经实践，证明确实能有效提高字节串表的检索效率及整体执行效率，但这需要付出 1 < < lzw_bits ＊2 字节的额外内存开销。

哈希表的容量同字节串表的容量，均为 1 < < lzw_bits ，表中每个元素是长度不固定的代码数组，这个数组用于存放哈希值相同的字节串代码，实际上对于一个较好的哈希函数产生的重复值极少。这样，检索字节串所经过的比较次数大幅减少，时间复杂度几乎接近于 O n ，从而有效提高了代码的执行效率。

值得一提的是哈希函数的选择也极其重要，不合适的哈希函数不仅不能提高执行效率，反而会降低执行效率；本文所附源代码中写入了多种哈希函数，读者可以亲自动手试验一下，看看它如何影响程序的执行效率。

4. 如何得到字节串的首字节

由于在将变长字节串表转化为固定长度字节串表过程中，用编码替代了前面的字节串，这将引起不能直接访问整个字节串的首字节。为了解决这个问题，程序在递归输出过程中巧妙地在堆栈 stack 中记录了字节串的首字节，避免了再次重复递归搜索首字节，这也对程序执行效率作出了不小的贡献。

5. 为压缩数据流添加头部信息

由于压缩时有可能采用不同长度位元的编码，所以解压时也必须采用相同的位元长度进行流输入，否则将出错。为此，我们在压缩流的前面添加了 3 个字节的信息，以记载压缩算法类型、版本及位元数等。这样在解压缩时，便可根据输入流自动判别有关参数，避免手工去确认这些参数，从而使解压缩函数调用接口更为简单。

四、LZW 压缩算法的 C + + 代码

本文源代码由 Borland C + + Builder5. 0 进行编译调试，唯一依赖于编译器的地方就是压缩过程中的流式输入输出代码，如果您需要移植代码到其它平台或其它开发环境 如 VC + + ，只需要修改流式输入输出等少量代码即可。当然如果将其编译成 DLL 动态链接库，则不必修改流式输入输出代码，只需要从 DLL 输出自已需要的接口即可。

另外，为了得到最高的执行效率，编译时应指示 C + + 编译器按 ‘速度最大 ”方式去优化编译代码，如果没有编译器的优化帮忙，程序执行效率会打折扣的。

五、小结

LZW 压缩算法较简洁，实现起来相对较容易，较适合于图像文件等冗余较多的文件类型，但文件长度不能过长，一般 1M 内效果较佳。

LZW 压缩算法本身也具有一定的局限性，比如当输入流内容冗余较少或输入流尺寸太大，这将导致内部维护的字节串表 StringTable 很快充满，致使它很快就会丧失压缩能力，一旦字节串表被充满，继续压缩则会引起反作用，并逐渐抵消已经得到的压缩率，更严重时则会使文件尺寸反而增大；另外，由于 LZW 压缩率还依赖于位元数 lzw_bits 的设置，当输入为任意类型的数据流时，压缩率不一定随位元数增加而增加，也不一定随位元数的减少而减少，也就是说 LZW 压缩率与位元数的具体关系基本上是不确定的；当输入流内容相对确定时，为了得到较好的压缩效果，可通过设置位元数 lzw_bits 反复试验，才能得到最满意的效果。

当然，LZW 压缩算法的缺陷并不是完全无法避免的，有些 LZW 变种压缩算法 如 ARC 通过在适当时候清空字节串表，来避免字节串被充满，从而能够有效提高压缩率。

本文仅起抛砖引玉作用，由于本人水平所限，文中可能有考虑不周，甚至包含错误；另外，本文源代码并未提供错误校验方面的代码，如果读者感兴趣，可以自行提供校验代码和添加输入流中原始数据的 CRC32 或 ADLER32 检验值，同时，本人也非常希望能和大家能一起探讨数据压缩技术 王俊蛟 junjiao@126. com http / /junjiao. yeah. net 。

VC + + 中状态栏的动态编程

兰 帆

摘 要 本文介绍了一种在 Visual C + + 6.0 中动态改变状态栏内的字符串并在状态栏中动态显示图标的方法。

关键词 Visual C + + ,状态栏,CStatusBar

一、引言

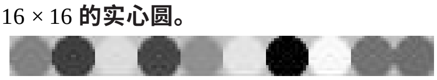
在应用程序的状态栏上显示一个不断变化的字符串，例如，当程序运行时在状态栏显示当前的系统时间，如再在状态栏上加上动画式图标将为您的程序增色不少。有一些文章也介绍过 VC 中状态栏的动态编程，可在具体实现的时候并不能让人满意。经过一定的研究，我这里提供一种动态改变状态栏的方法，可以很好地实现状态栏的动态改变，并在状态栏中加入动画图标。

二、实现原理

首先，在 CMainFrame 类中添加 InitStatusBar 函数用来对我们的状态栏进行初始化，利用 CMainFrame 类中的 CStatusBar 类成员变量 m_wndStatusBar 来实现对状态栏的控制。在 InitStatusBar 函数中还将初始化一个 CImageList 类变量 m_BarImage，用它在状态栏中显示图标。还要在 CMainFrame 类中添加 OnUpdateMyStatus 函数用以响应状态栏更新消息，还可在 CMainFrame 类中添加 OnTimer 函数响应 WM_TIMER 消息，并在 OnTimer 函数体内改变将要显示的字符串和图标。

三、具体实现

1. 用 Visual C + + 6.0 创建一个单文档工程 Test，确认在创建过程中选中了 Initial status bar 选项。在资源中插入一个新位图资源 IDB_BITMAP1，用十种不同的颜色在位图中画十个大小为 16 × 16 的实心圆。



2. 打开 MainFrm. h，添加如下变量定义：

```
// MainFrm. h : interface of the CMainFrame class
/////////////////////////////////////////
class CMainFrame : public CFrameWnd
{
  . . .  . . .
// Implementation
public:
  CString TimeTextOld; //存储当前状态栏显示时间的变量
   CString TimeText; // 存储当前时间的变量
   int m_ImageNo; //存储状态栏显示的图标序号的变量
   BOOL InitStatusBar(UINT *pIndicators, int nSize);
//初始化状态栏函数
   int m_MyPane; //状态栏中我们自己添加的窗格号
   virtual ~CMainFrame();
#ifdef _DEBUG
   virtual void AssertValid() const;
   virtual void Dump(CDumpContext& dc) const;
#endif
protected:  // control bar embedded members
   CStatusBar  m_wndStatusBar;
   CToolBar    m_wndToolBar;
// Generated message map functions
protected:
   CImageList m_BarImage;
   //{{AFX_MSG(CMainFrame)
   afx_msg int OnCreate(LPCREATESTRUCT lpCreateStruct);
   afx_msg void OnUpdateMyStatus(CCmdUI * pCmdUI);
//更新 MyPane 窗格的函数
   afx_msg void OnTimer(UINT nIDEvent); //WM_TIMER 响
```


参考文献

1. 《数据结构- - C + + 语言描述》西安交通大学出版社，1999 年，赵文静 著

2. 《Data Structures and Algorithm Analysis》电子工业出版社 1998 年 Clifford A. Shaffer 编著 《数据结构与算法分析》张铭、刘晓丹译

3. 《资料压缩——原理与实务》台湾松岗电脑图书资料股份有限公司，1994 年，张真诚、蔡文辉编著

4. 《Success With C + +》电子工业出版社，1996 年，[美]Kris Jamsa PH. D 编著 《C + +成功使用秘诀》张嵩、郝成江、张海、张虹翼等译

（收稿日期：2000 年 10 月 10 日）

```
应函数
	//}}AFX_MSG
	DECLARE_MESSAGE_MAP()
};
 ・・・ ・・・
////////////////////////////////////////
```

3. 在 CMainFrame 类中添加 BOOL CMainFrame InitStatusBar UINT ＊pIndicators int nSize ，在 InitStatusBar 函数中加入如下代码：

```
BOOL CMainFrame::InitStatusBar(UINT ＊ pIndicators, int nSize)
{
	//在状态栏指示器中增加 ID_INDICATOR_MY 窗格
	m_MyPane = nSize++;
	pIndicators[m_MyPane] = ID_INDICATOR_MY;
	//用位图资源 IDB_BITMAP1 创建 CImageList 对象，IDB_BITMAP1 可以是一堆位图，笔者就是用十种颜色在 IDB_BITMAP1 中画了十个大小为 16×16 的实心圆
	m_BarImage.Create (IDB_BITMAP1, 16, 16, RGB(192, 192, 192));
//在 InitStatusBar 函数返回前用 SetTimer(1, 300, NULL)安装定时器
	SetTimer(1, 300, NULL);
   return m_wndStatusBar.SetIndicators(pIndicators, nSize);
}
```

4. 在 CMainFrame 类中添加 WM_TIMER 消息响应函数 OnTimer UINT nIDEvent ，在 OnTimer 函数中加入：

```
void CMainFrame::OnTimer(UINT nIDEvent)
{
// TODO: Add your message handler code here and/or call default
//获得当前时间并转换成 CString 变量 TimeText，在更新状态栏时将 TimeText 字符串显示在状态栏上
	CTime time = CTime::GetCurrentTime();
	TimeText = time.Format(″ %Y 年%m 月%d 日 %A 当前时间 %X″);
	//显示的位图序号为秒数的个位数字
	m_ImageNo = time.GetSecond()%10;
	//CFrameWnd::OnTimer(nIDEvent);
}
```

5. 在 CMainFrame 类中添加 ID_INDICATOR_MY 的消息响应函数：OnUpdateMyStatus CCmdUI＊ pCmdUI ，在 OnUpdateMyStatus CCmdUI＊ pCmdUI 中添加：

```
void CMainFrame::OnUpdateMyStatus(CCmdUI＊ pCmdUI)
{
// TODO: Add your command update UI handler code here
//TimeTextOld 是当前状态栏中所显示的字符串，if 条件语句是为防止状态栏被不必要地更新，这样只有在时间发生改变时才更新状态栏可避免闪烁
		if(TimeText! = TimeTextOld)
	{
		//更新 MyPane 窗格中显示的字符串
		pCmdUI->SetText(TimeText);
		TimeTextOld = TimeText;
		CDC＊ pDC = m_wndStatusBar.GetDC();
		RECT m_PaneRect;
		CPoint m_PanePoint;
	//GetItemRect 函数用来获得状态栏的位置信息
	 m_wndStatusBar.GetItemRect(m_MyPane, (LPRECT)&m_PaneRect);
//将 m_PaneRect 矩形区域的左上角坐标赋给 m_PanePoint
		m_PanePoint.x = m_PaneRect.left;
		m_PanePoint.y = m_PaneRect.top;
	//在状态栏 m_PanePoint 处显示第 m_ImageNo 号位图
	m_BarImage.Draw(pDC, m_ImageNo, m_PanePoint, ILD_NORMAL);
	}
}
```

6. 在 CMainFrame 类的 CMainFrame OnCreate LPCREATESTRUCT lpCreateStruct 函数中添加代码如下：

```
int CMainFrame::OnCreate(LPCREATESTRUCT lpCreateStruct)
{
 ・・・ ・・・
//用 GetPaneInfo 和 SetPaneInfo 函数重新设置状态栏 ID_INDICATOR_MY 的宽度为 300 象素 nID，nStyle 和 nWidth 为临时变量用来接收 GetPaneInfo 返回的数据
	UINT nID, nStyle;
	int nWidth;
	 m_wndStatusBar.GetPaneInfo(m_MyPane, nID, nStyle, nWidth);
m_wndStatusBar.SetPaneInfo(m_MyPane, nID, nStyle, 300);
	return 0;
}
```

四、结束语

编译并运行应用程序，在程序的状态栏中可以看到一个不断变化的图标和当前的实时时间　见下图）。

收稿日期：2000 年 5 月 29 日）

PB 环境下一个高效录入界面的设计与实现

窦 燕

摘 要 本文介绍了一个在 PB 环境下基于数据窗口技术的高效可视化录入界面的设计与实现。
关键词 PB,可视化,界面,代码表

一、引言

用户界面是系统与用户之间最直接的交互界面，用户界面的友好性体现了软件设计的质量，并在很大程度决定了软件设计是否成功。任何一个管理信息系统的开发都免不了要为用户提供录入原始数据和维护原始数据的界面。当用户有大量的数据需要输入时，良好的数据录入界面将减少了系统应用的难度，极大地影响数据录入的速度和质量。

PowerBuilder 是一个面向对象的基于客户机/ 服务器模式的数据库前端开发工具，它给程序设计者提供了若干设计数据录入界面的方法。单行文本框用来录入一些无规律的信息，如姓名、通信地址等；多行文本框用来录入文字多的信息，如个人简历等；掩码框能为用户提供固定模式的数据录入形式；下拉式列表框让设计人员可以预先为用户提供一组固定信息，供用户选择录入，如一个单位的部门、一个学校的院系、专业等。充分利用 PB 提供的这些数据录入方法，可以设计出良好的数据数据录入界面。

二、问题的提出

尽管 PB 提供了许多数据录入的格式，但作者在实际开发一个学生成绩管理信息系统录入界面时，还是觉得有不尽人意的地方。如学生的学号，尽管每个学生的学号都不相同，但有一部分学生的学号是相连的，因此在成批录入学生学号时，能不能只录入一部分学生的学号，而让与它相连学生的学号自动生成。再如一部分学生所在院系、所在专业、所在班级是相同的，能否将上一个学生的信息自动带入到下一个学生的信息中。另外，有时用户想选择一个专业时，只在某院系的专业中选择。这些录入要求，PB 是没有现成的模式可用，因此只有用户自己根据需要开发出符合实际的数据录入界面。

因此本文要解决的问题是：

1. 对于学号字段的录入，如何让新记录的字段值在上一个字段值的基础上自动加 1，而在一个新院系开始时用户再重新输入；

2. 对于所在院系、所在专业、所在班级字段，如何让新记录的字段值保持上一个对应字段的值；

3. 对于所在专业字段，如何在选择所在院系字段值的前提下，只显示出该院系的所有专业。

三、问题的分析与解决方案

1. 对于学号，其编排是有规律的。例如：99 年入学的学生学号的前两位是 99，所属院系为机械工程学院机电专业，该学院该专业的编号为 012，则学号的中间三位号为 012，该专业该年招生为 120 人，该生为第 19 号，则该生的学号为 99012019。

因此，若输入这 120 名学生的信息，只要输入第一个学生的学号，则其它人的学号在前一个学生的学号基础上只要顺序加上 1 即可。

2. 姓名无规律可循，须单独输入。

3. 性别只有两种，在这里选择单选框，而不是用下拉列表用以提供用户的选择。

4. 所在院系，通过前面对学生学号的编排的分析，一个院系一次可能招 200 人，则只要输入第一个学生的所在院系，其它 119 人的所在院系保留上一个学生的所在院系即可，不用再重新输入。

5. 选择一个学生的所在专业，只希望在本院系的专业中选择，这样所在专业字段的值应只显示出某个院系的专业，若院系改变，则专业字段值也随之改变。

6. 对于所在班级，则可以用 PB 提供。

四、数据库的设计

1. 首先设计出要录入数据的数据库表的结构。如表 1 所示：

表 1 学生信息表 stu_xx 的结构

NAME	TYPE	WIDTH	DEC	中文解释
id	C	8		ID 卡号
name	C	8		姓名
xb	C	1	0	性别
yx_dm	N	2	0	院系代码
zy_dm	N	2	0	专业代码
class	N	1	0	所在班级

2. 建立院系代码表。如表 2 所示。

表 2　院系代码表 yx_code 的结构

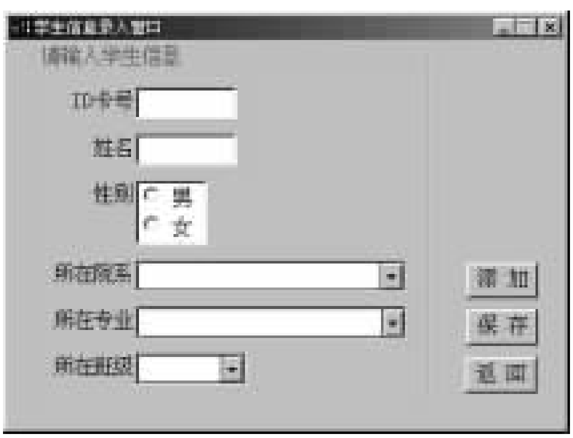
NAME	TYPE	WIDTH	DEC	中文解释
yx_dm	N	2	0	院系代码
yx_mc	C	20		院系名称

3. 建立专业代码表。如表 3 所示。

表 3　专业代码表 zy_code 的结构

NAME	TYPE	WIDTH	DEC	中文解释
zy_dm	N	2	0	专业代码
zy_mc	C	20		专业名称
yx_dm	N	2	0	院系代码

五、界面的设计与具体的实现方法



该录入界面最主要的控件为数据窗口控件，对于它的详细解释参见下面对本文提出的第三个问题的解决方法。

1. 对于第一、第二个问题的实现都比较简单。

先添加一条空记录，对于学号，先把上一个学号值转换为 integer 型，加 1，结果转换成 string 即可。在命令按钮“添加”的单击事件中编写如下的脚本：

```
//添加一个空行
dw_1. InsertRow (0)
//将新添行显示在当前窗口
integer new_row
new_row = dw_1. rowcount( )
dw_1. ScrollToRow(new_row)
//计算新行的部分值
//学号值为前一个学生学号值加 1
string new_id_str
long new_id_int
new_id_str = dw_1. GetItemString( new_row - 1, "id")
new_id_int = long(new_id_str) + 1
new_id_str = string(new_id_int)
dw_1. setitem(new_row, 1, new_id_str)
//所在院系值为前一个学生的所在院系
string new_dep
new_dep = dw_1. GetItemString( new_row - 1, "yx_dm")
dw_1. setitem(new_row, 4, new_dep)
//所在专业值为前一个学生的所在专业
string new_major
new_major = dw_1. GetItemString( new_row - 1, "zy_dm")
dw_1. setitem(new_row, 5, new_major)
//所在班级值为前一个学生的所在班级
integer new_class
new_class = dw_1. Getitemnumber( new_row - 1, "class")
dw_1. setitem(new_row, 6, new_class)
```

2. 对于本文提出的第三个问题，实现起来要稍稍麻烦些。其实现的步骤如下：

(1) 为 yx_code 表建一个包含其所有数据的 Quick Select 数据源和 Grid 显示风格的无标题的数据窗口对象 d_dddw_yx；

(2) 为 zy_code 表建一个包含 yxname 检索参数的 Quick Select 数据源和 Grid 显示风格的无标题的数据窗口对象 d_dddw_zy；

(3) 为 stu_xx 表建一个包含其所有数据的 Quick Select 数据源和 Grid 显示风格的数据窗口对象 d_dddw_xsxx；

(4) 数据窗口对象 d_dddw_xsxx 中还有两个数据窗口对象，说明如表 4。

表 4　数据窗口说明

下拉式数据数据窗口	描述
d_dddw_yx	包含 yx_code 表的所有数据。 显示列为 yx_mc，数据列为 yx_dm
d_dddw_zy	检索参数包含 yx_dm 列。 显示列为 zy_mc，数据列为 zy_dm

(5) 将 yx_dm 列的编辑风格设为下拉式数据窗口 (DDDW) 风格，数据窗口为 d_dddw_yx；

(6) 将 zy_dm 列的编辑风格设为下拉式数据窗口 (DDDW) 风格，数据窗口为 d_dddw_zy；

(7) 在数据窗口画板中，选择 Row - - > Data；

(8) 在 Data Retained on Save 对话框中单击 Insert 按钮；

(9) 单击 OK 按钮；

当选择所在院系为“数理系”时，所在班级显示出“信息与计算科学”、“理工科”、“统计学”和“文科”等专业，而当选择所在院系为“信息科学与工程学院”时，则显示“计算机科学与技术”、“电子信息工程”、“通信工程”和“电子科学与技术”专业，避免了将所有的专业都显示出来而降低了选择的速度。

上述数据窗口控件中的 ItemChanged 事件的脚本程序如下：

```
DataWindowChild zy_child    //定义一个子数据窗口对象
integer return_value, yx_id, null_value
SetNull(null_value)     //设置变量 null_value 为空
//如果当前列为´yx_dm´,取出当前的´yx_dm´以备´zy_dm´
//列的下拉式数据窗口中使用;然后得到´zy_dm 列的下拉
//式数据窗口的句柄,并以当前的´yx_dm´的值作为参数
//进行检索.
```

利用内存映射文件作快速的全文检索

江天送　王源源

在文件里查找某一字符串，在程序设计上常常要用到。若用 Delphi 写，小文件可以用 TmemoryStream 把整个文件读进内存，在内存上进行查找。但若文件比较大，上几十兆，就不能用此法，若你你强行用 TmemoryStream，硬盘因交换文件而不断的读写，效率很低。

大家对 Win32 的内存映射文件 File Mapping 一定不会陌生，用它可以方便地实现多个进程共享数据。这里介绍用内存映射文件作快速的全文检索，功能如 Windows 的查找 在某目录的文件里查找某一字符串 ，但此法比 Windows 的查找效率要高。

内存映射文件可以让我们在访问磁盘文件时如同正在访问内存中的文件一样，可以避免进行文件的输入输出操作。它先是保留一段虚拟内存地址空间，然后将磁盘文件提交给这段内存空间。我们只需要一个指向该区域的指针就可以访问整个文件的内容了。系统负责处理数据的缓存、缓冲、写入和调用以及内存的分配和释放，我们就好像在一块大的内存区域上查找字符串，效率比较高。

下面介绍具体的做法

1. 获得查找文件的文件句柄。可以用 FileOpen 打开文件。

2. 创建文件内存映射对象

无论是命名的或是无命名的内存映射文件对象，都是用 CreateFileMapping 函数创建。函数的参数如下：

```
function CreateFileMapping(hFile: THandle; //OpenFile()返回的句柄
lpFileMappingAttributes: PSecurityAttributes; //安全属性,一般为 nil
flProtect, {
PAGE_READONLY(文件只读)
PAGE_READWRITE (文件可读写)
PAGE_WRITECOPY(文件可读写,但进行写操作时,会复制修改过的页面)}
dwMaximumSizeHigh,  //文件最大尺寸的高 32 位,除非文件大于 4GB,否则为 0
dwMaximumSizeLow: DWORD; //文件最大尺寸高 32 位
lpName: Pchar//文件映射对象的名称,可含除‘/’的所有字符,若 nil 创建无名对象
): THandle; stdcall;
```

3. 映射文件视图到进程的地址空间

用 MapViewOfFile 函数,函数参数：

```
function MapViewOfFile(hFileMappingObject: THandle;
//CreateFileMapping()返回句柄
dwDesiredAccess: DWORD;
//数据访问模式,可设读写、只读、Copy-on-write
dwFileOffsetHigh, dwFileOffsetLow,
dwNumberOfBytesToMap: DWORD
//需要映射的字节数,0 为文件全部
): Pointer; //返回视图的起初地址
```

4. 在视图里查找字符串

本查找算法应用了回调函数，当找到符合的字符串时调用函数,若想停止查找,可返回 False,详见源程序。

5. 事后清洁工作

UnmapViewOfFile 解除文件视图,CloseHandle 分别关闭文件映射对象和文件内核对象。

```
IF dwo.name = 's_depart' THEN
    return_value = dw_1.GetChild("s_major", zy_child)
    IF return_value = -1 THEN
      MessageBox("错误", "这不是一个子数据窗口")
   Return
  END IF
  yx_id = integer(data)
    zy_child.settrans(sqlca)
  zy_child.retrieve(yx_id)
  this.object.s_major[row] = string(null_value)
END IF
```

六、结论

本文作者所设计的录入界面，在录入成批有规律的数据方面体现了较好的优势，有的记录甚至只用输入学生的姓名即可完成一个学生信息的输入，大大提高了数据录入的效率。如果信息录入量较大，某些字段值是有规律，而某些字段的数据是动态的，则采用本文所提供的录入界面能更好地体现出其较好的优越性，具有一定的推广应用价值，最大限度地减少了用户数据的工作量，提高了工作效率。上述程序已在 PowerBuilder6.0 下调试通过，已用于多个 MIS 的开发中，效果很好。

参考文献

1. 胡岩、王宇航、何慧 . PB 环境下模拟下拉列表框显示动态数据，计算机工程，1999 12　104~105

2. 韦明、张森 . 可视化技术在管理信息系统中的实现，计算机工作，2000 5 ，93~94

3. 张步达、杨慧、石京民 . PowerBuilder 数据窗口技术详解 电子工业出版社 2000.1

收稿日期：2000 年 7 月 28 日）

下面是演示程序窗体和代码


```
unit Unit1; interface
uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  StdCtrls;
type
  TForm1 = class(TForm)
    OpenDialog1: TOpenDialog;
    GroupBox1: TGroupBox;
    LFilename: TLabel;
    EdSt: TEdit;
    BBrowse: TButton;
    BSearch: TButton;
    ListBox1: TListBox;
    Label2: TLabel;
    Button3: TButton;
    EdText: TEdit;
    LText: TLabel;
    Cbignore: TCheckBox;
    procedure BBrowseClick(Sender: TObject);
    procedure BSearchClick(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
type FindCallback = function(pos: integer): boolean;
var
  Form1: TForm1;
implementation
{$R *.DFM}
//回调函数, pos 为找到字符串在文件的偏移量, 若返回 True 继续查找, False 停止查找
function found(pos: integer): boolean;
begin
  Form1.Listbox1.Items.Add('在偏移量: ' + inttostr(pos)
    + ' 处找到此字符串');
  result: = true;
end;
//查找函数, Fname 为查找文件名, CallBack 为回调函数(见下), ignoreCase 为是否忽略
procedure searchText(Fname: string; text: pchar;
 callback: FindCallback; ignoreCase: boolean = TRUE);
var
  FFileHandle: THandle; // 文件内核句柄
  FMapHandle: THandle;  // 文件映射句柄
  FFileSize: Integer;
  PData: PChar;       // 文件视图的地址
  textlen, i: integer;
  buf: array[0..255] of char;
begin
  if not Assigned(callback) then exit;
  //若 CallBack 过程是否有效, 否退出
  FFileHandle : = FileOpen(FName, fmOpenRead);
  //打开文件内核对象
  if FFileHandle = INVALID_HANDLE_VALUE then
    raise Exception.Create('打开文件错误');
  try
    FFileSize : = GetFileSize(FFileHandle, Nil);
    FMapHandle : = CreateFileMapping(FFileHandle, nil,
       PAGE_READONLY, 0, FFileSize, nil);
       //只读方式创建文件内存映射对象
    if FMapHandle = 0 then
      raise Exception.Create('创建文件内存映射对象错误');
  finally
    CloseHandle(FFileHandle);
    //若异常者关闭文件对象
  end;
  try
   PData : = MapViewOfFile(FMapHandle, FILE_MAP_READ, 0, 0, FFileSize); //映射文件视图
     //PData 为映射文件视图, 返回映射视图的初始地址
    if PData = Nil then
      raise Exception.Create('创建映射视图出错!');
  finally
    CloseHandle(FMapHandle); //关闭文件映射对象
  end;
  try
   textlen: = strlen(text);
   if ignorecase then //忽略大小写的处理
    begin
      strcopy(text, StrUpper(text));
      for i: = 0 to FFileSize - 1 - textlen do
       begin
        if UpCase(PData[i]) = text[0] then
          //找第一个吻合的字符
          begin
             move(Pdata[i + 1], buf[0], textlen - 1);
             if strLcomp(StrUpper(buf), pchar(@text[1]),
               textlen - 1) = 0 then //对比剩下的字符串
               if not callback(i) //回传 CallBack 函数, 若
                then break;  //返回 False 者退出查找
          end;
       end;
     end else
        begin //大小写区别的处理, 和上面的处理相近
```

Visual C++工具条编程探讨

谢经荣

摘　要　本文介绍了制作两种流行的工具条界面的原理和编程方法,并用 Visual C++给出了示范。

关键词　工具条快捷菜单,弹出式工具条

一、概述

商业软件非常注重界面的友好性，有的成熟的软件产品做得象工艺品一样精细，这是国内许多软件与国外软件的差距。本文对流行的工具条界面的编程作一探讨。

大型软件通常拥有许多的工具条，并且通常能够在查看菜单下决定哪些显示或隐藏。Word97、VC5.0、WPS2000 等更能够在工具条的空白处响应鼠标的右键后显示快捷菜单，迅速显示或隐藏某些工具条或对话条。如何实现这一功能呢？

还有一些软件，如 PhotoShop，工具条按钮“文字工具 T”按下后会显示一串相关的功能按钮供选择，这样可以减少按钮的数目。WinAmp 的‘加入”按钮操作也类似，尽管它并不是用普通的 Windows 窗口实现。这种工具条的用处在笔者的一个符号库软件中就用到，该软件符号分为点状、线状、面状三种，每种有几十个相应的符号，全部显示在工具条中显然不合适，采用这种弹出式工具条　暂这样命名）可以很好地解决问题。运行时界面如下：

二、下面是用 VC5.0 实现工具条快捷菜单的示例

首先用 Spy.exe 分析停靠工具条 ToolBar，船坞窗口 DockBar，主框架窗口 MainFrame 的关系，可以知道，船坞窗口是停靠工具条的父窗口，主框架窗口又是船坞窗口的父窗口，在工具条中响应鼠标右键消息，可以直接在 CtoolBar 派生类中实现；在船坞窗口中响应鼠标右键消息原理也是一样，但关键是要在船坞窗创建时用自己的类实例代替 MFC 中的类实例。

通过对 MFC 实现浮动工具条的源代码的分析，知道 DockBar 的创建在 CFrameWnd　DockControlBar　中实现的，所以重新写这部分代码即可实现上述功能：

```
static DWORD g_dwDockBarMap[4][2] =
{
  { AFX_IDW_DOCKBAR_TOP,      CBRS_TOP    },
  { AFX_IDW_DOCKBAR_BOTTOM,  CBRS_BOTTOM },
  { AFX_IDW_DOCKBAR_LEFT,     CBRS_LEFT   },
  { AFX_IDW_DOCKBAR_RIGHT,    CBRS_RIGHT  },
};
void CMainFrame::MyEnableDocking(DWORD dwDockStyle)
{
// must be CBRS_ALIGN_XXX or CBRS_FLOAT_MULTI only
  ASSERT((dwDockStyle & ~(CBRS_ALIGN_ANY|CBRS_FLOAT_MULTI)) == 0);
   m_pFloatingFrameClass = RUNTIME_CLASS (CMiniDockFrameWnd);
  for (int i = 0; i < 4; i++)
```

```
      for i: =0 to FFileSize - 1 - textlen do
        if PData[i] =text[0] then
          begin
            move(Pdata[i+1], buf[0], textlen - 1);
            if strLcomp(buf, pchar(@text[1]),
              textlen - 1) =0 then
              if not callback(i) then  break;
          end;
      end;
  finally
   UnmapViewOfFile(PData); //删除文件视图
  end;
end;
```

```
procedure TForm1.BBrowseClick(Sender: TObject);
begin
   if opendialog1.Execute  then
     edst.Text : =opendialog1.FileName;
end;
procedure TForm1.BSearchClick(Sender: TObject);
begin
   searchText(Edst.text, pchar(edtext.text), found, Cbignore.checked);
end;
end.
```

收稿日期　2000 年 4 月 24 日

```
        {
 if (g_dwDockBarMap[i][1] & dwDockStyle & CBRS_ALIGN_ANY)
            {
//用自己的类 CMyDockBar 代替 MFC 类 CdockBar!!
            CMyDockBar * pDock = (CMyDockBar * )GetControlBar(g_dwDockBarMap[i][0]);
                 if (pDock  = = NULL)
                 {
                      pDock  = new CMyDockBar;
              if ( ! pDock ->Create(this, WS_CLIPSIBLINGS | WS_CLIPCHILDREN | WS_CHILD | WS_VISIBLE | g_dwDockBarMap[i][1], g_dwDockBarMap[i][0]))
                      {
                           AfxThrowResourceException();
                      }
                 }
            }
        }
}
```

在自定义船坞窗口类 CMyDockBar 中，定义鼠标右键消息响应，如下

```
void CMyDockBar:: OnRButtonDown(UINT nFlags, CPoint point)
{
      CWnd: : OnRButtonDown(nFlags, point);
      AfxMessageBox("捕获到船坞窗口的鼠标右键消息.要实现快捷菜单请在这里加代码!");
}
```

至于工具条窗口中的鼠标消息很容易获取

```
void CDTb: : OnRButtonDown(UINT nFlags, CPoint point)
{
      CToolBar: : OnRButtonDown(nFlags, point);
      AfxMessageBox("捕获到工具条的鼠标右键消息.要实现快捷菜单请在这里加代码!");
}
```

这里主要介绍鼠标右键按下的消息捕获和处理，这是解决问题的关键。至于弹出快捷菜单很容易实现，这里省略。

三、下面是用 VC5.0 实现弹出式工具条的示例

工具条按钮被按下时，创建一个 Dialog Dialog 创建子窗口 ToolBarCtrl，ToolBarCtrl 的按钮按下消息再发送到消息队列中，并撤消 Dialog，这就是整个弹出式工具条的工作全过程。看似容易，但有几点还是值得注意的　一是工具条提示 ToolTip，二是工具按钮图案的切换。

首先给出工具条类的声明

```
#define  MAX_TBMAIN_COUNT    20
#define  MAX_TBSUBB_COUNT    100
class CDTb : public CToolBar
{
// - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
public:
   CBitmap *  m_pBitmap;
   CDG m_dg;
   UINT m_nTotalCount;
   UINT m_nMainCount; //工具条上的按钮数目
   UINT m_nMainID[MAX_TBMAIN_COUNT]; //工具条上的按钮 ID 号
   UINT m_nIDBitmap[MAX_TBMAIN_COUNT]; //工具条上的按钮图案
   UINT m_nSubCount[MAX_TBMAIN_COUNT]; // 工具条上的按钮的弹出窗口中子按钮数目
   UINT m_nSubBaseINDEX[MAX_TBMAIN_COUNT +1];
//当前选中的子按钮序号
   UINT m_nSubID[MAX_TBMAIN_COUNT][MAX_TBSUBB_COUNT]; //各子按钮的 ID 号
BOOL Init(UINT total, UINT mainCount, UINT * mainID, UINT * bmp, UINT * subCount, UINT * subID);
   BOOL Create(CWnd * pParent);
// - - - - - - - - - - - - - - - - - - - - - - - - - -
//......(其他类成员声明)
// Generated message map functions
public:
   //{{AFX_MSG(CDTb)
afx_msg void OnLButtonDown(UINT nFlags, CPoint point);
afx_msg void OnRButtonDown(UINT nFlags, CPoint point);
   //}}AFX_MSG
DECLARE_MESSAGE_MAP()
};
```

创建弹出式工具条分两步：首先用 Init　初始化有关参数，再调用 Create　创建窗口，如下：

```
int  CMainFrame: :  OnCreate(LPCREATESTRUCT  lpCreateStruct)
{
   if (CFrameWnd: : OnCreate(lpCreateStruct)  = =  -1)
         return  -1;
   InitToolbar();
   m_wndToolBar. Create(this);
/ *      m_wndToolBar. LoadToolBar(IDR_MAINFRAME);
   if (!m_wndToolBar. Create(this) ||
         !m_wndToolBar. LoadToolBar(IDR_MAINFRAME))
   {
         TRACE0("Failed to create toolbar\n");
         return  -1;       // fail to create
   } * /
//......
}
void CMainFrame: : InitToolbar()
{
   UINT total  = 3 * 9;
   UINT mainCount =3;
UINT mainID[]  ={ID_BBTN001, ID_BBTN011, ID_BBTN021};
   UINT bmp[] ={IDB_BMP4, IDB_BMP5, IDB_BMP6};
   UINT subCount[] ={9, 9, 9};
   UINT subID[27];
   for(UINT i =0; i <total; i + +)
```

```
    {
        subID[i] =ID_BBTN001 + i;
    }
    m_wndToolBar. Init(total, mainCount, mainID, bmp,
subCount, (UINT ∗)subID);
}
```

弹出 ToolBarCtrl 的过程上面已经叙述，实现的具体代码如下

```
void CDTb: : OnLButtonDown(UINT nFlags, CPoint point)
{
    CToolBar: : OnLButtonDown(nFlags, point);
    //...... 判断是否按到按钮上,没有则返回
    m_dg. MyCreate(this, nHitIndex, m_nIDBitmap
[nHitIndex], m_nMainID[nHitIndex], m_nSubCount
[nHitIndex], m_nSubBaseINDEX[nHitIndex], m_nSubID
[nHitIndex] );
    m_dg. ShowWindow(SW_SHOW);
    m_dg. SetCapture();
}
void CDG: : MyCreate(CWnd ∗ pParent, UINT nMainIndex,
UINT nIDBitmap, UINT nMainID, UINT nSubCount, UINT
nSubBaseINDEX, UINT nSubID[])
{
    VERIFY(Create(CDG: : IDD, pParent));
    //...... 重新设置类成员变量
    //以下创建 ToolBarCtrl
    m_dt. Create(WS_CHILD | WS_VISIBLE | CCS_TOP | TB-
STYLE_TOOLTIPS, CRect(0, 0, 300, 300), this,
IDC_TOOLBARCTRL);
    m_dt. AddBitmap(nSubCount, m_nIDBitmap);
    m_dt. AddButtons(nSubCount, Buttons);
    //设置窗口到适当位置
    //......
this ->MoveWindow(rect2. left, rect2. bottom, rect. Width()
 +6, rect. Height() +6);
    //允许工具提示
    EnableToolTips(TRUE);
}
void CDG: : OnLButtonUp(UINT nFlags, CPoint point)
{;
    CDialog: : OnLButtonUp(nFlags, point);
    ReleaseCapture();
    MapWindowPoints(& m_dt, & point, 1)
    m_dt. OnLButtonUp(nFlags, point); //调用 CToolBarC-
trl: : OnLButtonUp()
    int sub = m_dt. m_nHitIndex;
    if(sub>=0)
    ((CDTb ∗)m_pParent) ->SetButtonInfo(m_nMainIndex,
m_nSubID[sub], TBBS_BUTTON | TBSTATE_ENABLED,
m_nSubBaseINDEX + sub); //更新工具条按钮图案
    DestroyWindow();
}
void CDG: : OnMouseMove(UINT nFlags, CPoint point)
{;
  CDialog: : OnMouseMove(nFlags, point);
  MapWindowPoints(& m_dt, & point, 1)
  m_dt. OnMouseMove(nFlags, point); //调用 CToolBarC-
trl: : OnMouseMove ()
}
BOOL CDG: : OnToolTip(UINT id, NMHDR ∗ pNMHDR, LRE-
SULT ∗ pResult)
{
  //实现窗口工具提示,详可见 VC 示例 DLGCBR32 以及关
于 ToolTip 的说明
}
void CDT: : OnMouseMove(UINT nFlags, CPoint point)
{
  CToolBarCtrl: : OnMouseMove(nFlags, point);
  GetButton(m_OldHitIndex, & btn);
  CheckButton(btn. idCommand, FALSE); //恢复按钮为弹
起状态
  GetButton(m_nHitIndex, & btn);
  CheckButton(btn. idCommand, TRUE); //将按钮设为下压
状态
  m_OldHitIndex = m_nHitIndex;
}
void CDT: : OnLButtonUp(UINT nFlags, CPoint point)
{
  CToolBarCtrl: : OnLButtonUp(nFlags, point);
  TBBUTTON btn;
  int nHitIndex = m_OldHitIndex;
  if(nHitIndex >=0 & & nHitIndex <GetButtonCount() )
  {
       GetButton(nHitIndex, & btn);
       UINT nHitID = btn. idCommand;
  AfxGetMainWnd( ) ->PostMessage(WM_COMMAND,
(WPARAM)nHitID, (LPARAM)0);
  }
}
```

编程时注意将 CDT 中的两个消息响应函数 void CDT OnLButtonUp ，void CDT OnMouseMove 改为 public 类型的，以便在 CDG 类中使用。以上代码大都不完整，完整的源程序可查看软盘，或到 ftp. cug. edu. cn 的 incomint 目录下下载。

以上实现了弹出式工具条基本要求，当然这里只作了一下简单的示范，有些地方尚需要完善，比如子按钮数目为 1 即为普通工具条按钮时不弹出 ToolBarCtrl 子按钮数目大于 1 时在按钮图案右下角画一个记号 请仔细观察 Photoshop 的文字工具按钮右下的小三角形）；工具条水平停靠时 ToolBarCtrl 设置成列模式，工具条竖直停靠时 ToolBarCtrl 设置成行模式等等，这就留给读者完成吧。关于 VC 编程问题，欢迎与本人探讨。本人 Email Jrxie@263. net。

收稿日期：2000 年 8 月 24 日）

Windows NT/2000 下用 PSAPI 获取进程的信息

江天送

摘　要　Windows NT/2000 环境下用 Delphi 调用 PSAPI，获取系统进程、设备驱动程序、模块、页面的信息。

关键字　PSAPI（Process Status Helper API）、进程、模块、ToolHelp32 API

一、引言

在 Windows9x 下可以用 ToolHelp32 建立快照，列举进程、线程、模块、堆的信息，从而可以读写某进程的内存，结束某进程等功能。但 ToolHelp32 API 只适用于 Windows9x 而不能在 NT/2000 下运行，因为 ToolHelp 的某些函数违反了 NT 的健壮性的进程保护安全特性。

在 NT/2000 下怎样获取进程的信息呢？Windows 平台 SDK 提供了 PSAPI（Process Status Helper API）接口，Windows NT 的较高版本和 Windows 2000 的所有版本都有一个叫 PSAPI.DLL 的动态连接库。类似于 ToolHelp32，PSAPI 能够获取的信息包括：

1. 正在运行的进程
2. 已加载的设备驱动程序
3. 每个进程已加载的模块
4. 进程内存信息
5. 映射每个进程的文件内存

Delphi 5.0 为这个 DLL 提供了一个叫 PSAPI.pas 的接口单元，可以动态的加载所有的函数。下面介绍如何实现 PSAPI 的功能。

二、EnumProcesses（）列举进程

函数的定义：

```
function EnumProcesses(
lpidProcess: LPDWORD; //DWORD 类型足够大的数组的地址
cb: DWORD; //数组的大小
var cbNeeded: DWORD//返回写入的字节数 cbNeeded div sizeof(DWORD)即进程的个数
): BOOL; //成功调用返回 True
```

列举的方式和 ToolHelp 不同，ToolHelp 是 FindFirst 至 FindNext，这里是成功调用把所有的进程 ID 都写入 lpidProcesss。

三、EnumDeviceDrivers（）列举加载设备

函数的定义

```
function EnumDeviceDrivers(lpImageBase: PPointer; cb: DWORD;
var lpcbNeeded: DWORD): BOOL;
```

调用和 EnumProcesses（）相似，传入足够大的 Pointer 类型的数组。返回各设备的 ImageBase，后可调用 GetDeviceDriverFileName（）获取设备的文件名。

四、列举进程加载的模块

每个进程都加载了不同的模块（DLL），列举模块先要用 OpenProcess（）以只读、查询信息的方式打开进程，返回进程的句柄。后用 EnumProcessModules（）列举进程的模块，调用和 EnumProcesses（）相似，返回所有模块的句柄。可用 GetModuleFileNameEx（）获取各模块的文件名，GetModuleInformation（）获取各模块的信息（模块的基地址、大小、进入口）。

五、获取进程内存信息

调用 GetProcessMemoryInfo（）进程的内存信息，返回 PROCESS_MEMORY_COUNTERS 结构，有关于进程的各种统计数据。调用 QueryWorkingSet（）获取进程的各页面的信息，包括页面的地址、页面的类型。有了页面的地址就可调用 GetMappedFileName（）获取某地址是否包含有内存映射文件，并有返回映射文件的文件名。

下面是演示程序的界面和主要的代码：


```
unit Unit1;
interface
```


```
uses
  Windows, Messages, SysUtils, Classes, Graphics,
  Controls, Forms, Dialogs, ComCtrls, StdCtrls,
  PSAPI, ImgList;
type
  TFMain  = class(TForm)
    GroupBox1: TGroupBox;
    ListView1: TListView;
    BEnumProcesses: TButton;
    BEnumDrivers: TButton;
    BExit: TButton;
    procedure BEnumProcessesClick(Sender: TObject);
    procedure BEnumDriversClick(Sender: TObject);
    procedure ListView1DblClick(Sender: TObject);
    procedure BExitClick(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
var
  FMain: TFMain;
implementation
uses unit2;
{$R *.DFM}
//列举进程
procedure TFMain.BEnumProcessesClick(Sender: TObject);
var processIDs: array[0..$3FFF - 1] of DWORD; //进程 ID
Count: DWORD; i: integer;
ProcHand: THandle; //进程的句柄
ModHand: HMODULE; //模块的句柄
ModName: array[0..MAX_PATH] of char; //模块文件名
begin
  GroupBox1.Caption: =´正在运行的所有进程,鼠标双击有进程的详细信息´;
  ListView1.Items.Clear;
  //列举所有正在运行的进程
  if not EnumProcesses(@processIDs, SizeOf(processIDs), Count) then
    raise Exception.Create(´列举进程出错,确认是否安装了PSAPI.DLL!´);
  for i: =0 to (Count div Sizeof(DWORD)) -1 do
  begin
ProcHand: =OpenProcess(PROCESS_QUERY_INFORMATION or PROCESS_VM_READ, False, processIDs[i]); //查询方式打开进程
    if ProcHand > 0 then
      try
        //列举进程的首模块
        if EnumProcessModules(Prochand, @ModHand, sizeof(ModHand), Count) //获取模块的文件名
         then if GetModuleFileNameEx(Prochand, ModHand, ModName, SizeOf(ModName)) > 0 then
          begin
             with ListView1.Items.Add, SubItems do
              begin
               Caption : = ModName;
            Data : = Pointer(processIDs[I]); //保存进程的 ID
               Add(IntToStr(processIDs[I]));
              end;
          end;
      finally
        CloseHandle(ProcHand);
      end;
  end;
end;
//列举设备驱动程序
procedure TFMain.BEnumDriversClick(Sender: TObject);
var ImageBase: array[0..$3FFF - 1] of Pointer;
i: integer;  Count: DWORD;
DrvName: array[0..MAX_PATH] of char;
begin
  GroupBox1.Caption: =´系统加载的所有设备´;
  Listview1.Items.Clear;
   if not EnumDeviceDrivers(@ImageBase, SizeOf(ImageBase), Count) then raise Exception.Create(´列举设备出错,确认是否安装了 PSAPI.DLL!´);
  for i: =0 to (Count div Sizeof(DWORD)) -1 do
  begin
   if GetDeviceDriverFileName(imageBase[I], DrvName,
      SizeOf(DrvName)) > 0 then
     with ListView1.Items.Add do
     begin
       Caption : = DrvName;
SubItems.Add(´$´ +IntToHex(Integer(ImageBase[I]), 8));
     end;
  end;
end;
//翻译页面的类型
function MemoryTypeToString(Value: DWORD): string;
const
  TypeMask  = DWORD($0000000F);
begin
  Result : = ´´;
  case Value and TypeMask of
    1: Result : = ´Read -only´;
    2: Result : = ´Executable´;
    4: Result : = ´Read/write´;
    5: Result : = ´Copy on write´;
```

```
  else
    Result : = ´Unknown´;
  end;
  if Value and $100  <>0 then
    Result : = Result  +  ´, Shareable´;
end;
//某一进程的模块、页面等信息
procedure TFMain. ListView1DblClick(Sender: TObject);
const  AddrMask  =  DWORD($FFFFF000);
var
ProcHand: THandle; //进程的句柄
ModHand: array[0..1024] of HMODULE; //模块的句柄
ModName: array[0..MAX_PATH] of char; //模块文件名
ProcessID, Count: DWORD;
i: integer; WSPtr: Pointer;
WorkingSet: array[0..$3FFF  -  1] of DWORD;
ModInfo: TModuleInfo; //模块的信息
MapFileName: array[0..MAX_PATH] of char;
begin
 if Listview1. Selected <>nil then
   if Listview1. Selected. Data <>nil then
     with FDetail do
       begin
        MoudlesLV. items. clear; //模块的 ListView 清空
        MemoryLV. Items. Clear; //内存的 ListView 清空
        ProcessID: =DWORD(Listview1. Selected. Data);
ProcHand: =OpenProcess(PROCESS_QUERY_INFORMATION
or PROCESS_VM_READ, False, processID); //查询方式打开
进程
     LProcessID. caption: =Listview1. Selected. SubItems[0];
     LPriority. caption: =inttostr(GetPriorityClass(ProcHand));
        //进程的优先级
        EModule. Text : =Listview1. Selected. Caption;
        //列举进程所有的模块
         EnumProcessModules(Prochand, @ ModHand, sizeof
(ModHand) , Count);
        for i: =1 to (Count div Sizeof(DWORD)) -1 do
        if  (GetModuleFileNameEx(ProcHand,  ModHand[I],
ModName, SizeOf(ModName)) >0) and GetModuleInforma-
tion(ProcHand, ModHand[I], @ ModInfo, SizeOf(ModInfo))
then
          //GetModuleInformation()获取模块的信息
         with ModInfo, MoudlesLV. items. add, SubItems do
          begin
          Caption: =ModName; //模块文件名
          Add(´$´ +IntToHex(Integer(lpBaseOfDll), 8)); //模
块的基地址
        Add(inttostr(SizeofImage)); //模块的占的字节数大小
          add(´$´ +IntToHex(Integer(EntryPoint), 8));
          //模块的 Entry Point 进入口
         end;
         if QueryWorkingSet(ProcHand, @ WorkingSet, SizeOf
(WorkingSet)) then
        //获取所有内存页面信息
          for I : = 1 to WorkingSet[0] do
           begin
            WSPtr : = Pointer(WorkingSet[I] and AddrMask);
             MapFileName[0]: =#0;
       GetMappedFileName(ProcHand, WSPtr, MapFileName,
SizeOf(MapFileName));
             //获取所在页面内存映射的文件名
             with MemoryLV. items. add, subitems do
              begin
caption: =´$´ +IntToHex(Integer(WSPtr), 8); //页面的地址
add(MemoryTypeToString(WorkingSet[I])); //页面类型
               add(MapFileName); //内存映射的文件名
              end;
           end;
         CloseHandle(ProcHand);
         Show;
      end;
end;
procedure TFMain. BExitClick(Sender: TObject);
begin
  Close;
end;
   end.
```

收稿日期：2000 年 9 月 25 日）

软件保护新套装与 RG - DL 软件狗

近日，彩虹天地在现有加密狗产品原基础上，又研制、开发并最新推出一款新产品 RG - UMH 套装加密狗和一款升级产品 GS - DL 型软件狗。

新的 RG - UMH 套装有微狗、USB 狗和一根 USB 线组成，这款产品的最大特点是突破以往并口加密产品与 USB 口产品不兼容的桎梏。在使用时，用户或软件开发商无须重复安装驱动程序，即可在并口狗或 USB 狗间自由切换，体验彩虹田地以用户为中心的理念。

RG - UMH 套装产品是由 GS - MH 微狗和 USB 狗两大部分组成的。其中 GS - MH 微狗是使用在计算机并口上的用于软件的硬件产品。USB 狗是使用在计算机 USB 口上的硬件加密产品。使用 RG - UMH 套装程序，只需安装一次驱动，就可以完成对微狗与 USB 够的加密操作，从而轻而易举的满足最终客户的需求。

对于正在使用微狗或 USB 狗进行软件保护的用户，也只需简单的将现有的软件驱动升级，即可得到相互兼容的并口狗或 USB 狗。

另外，彩虹天地 RG - DL 型软件狗也是使用在计算机并口上的用于软件保护的硬件产品，具有 100 个字节的数据存储区。

RG - DL 型软件狗内部所采用的硬件不同与原 GS - DL 型软件狗，其稳定、可靠性能有了较大的提升；软件方面，RG - DL 型软件狗解决了原 DJ/DK 型软件存在的 DOS16、WIN16 模块在 WINDOWS NT/2000 下使用不稳定的问题。

彩虹天地为软件开发商提供网上免费升级服务，无论是 RG - UMH 套装还是软件狗 RG - DL，您都可以在网上获得相应的升级软件。

用 Visual C＋＋显示位图的原理与方法

王　丰

一、介绍

在 VC＋＋环境下显示位图并不是什么新技术，但本文仍然在此“老调重弹”的原因是：（1）这一技术十分重要，它是图像编程的基础，掌握不了这些基本原理也就很难独立开发出符合实际需要的应用程序；（2）许多关于 VC＋＋编程的资料都提供了显示位图（Bitmap）的实例，但遗憾的是，由于侧重点的不同使得有关调色板、设备上下文（DC）以及图形设备接口（GDI）等与位图密切相关的知识要么是很少提到，要么就是很不全面、一代而过，或者部分内容被放到了别处，显得支离破碎。这使得许多读者在模仿这些例子“克隆”出自己的应用程序后，仍感到有许多不解之处存在；（3）为了显示位图，Windows 及 MFC 提供了一些类和函数供我们利用，熟悉它们的作用对我们编程很有帮助。

二、基本概念与原理

调色板：调色板的概念必须首先介绍，它在除 24 位真彩色显示系统的其它系统中都要用到。尽管多媒体技术的发展令计算机所能显示的色彩越来越绚丽缤纷，但实际上，自然界无限种类的颜色目前仍无法在计算机上完全表达出来。现在最高级的所谓 24 位“真彩色”的显示系统也只能显示 2^{24} 即 16 777 216 种颜色，当然，这已经完全够用了，因为人眼还没有能力区分真彩色系统表现出的颜色与大自然中实际颜色的区别。在这种系统中，每一个像素的值都用红（R）、绿（G）、蓝（B）三色，每色 8 位共 24 位来表示，“24 位”显示系统的名字也由此而来，所以，其像素值就是要显示的颜色值，显然，此时只需要直接显示就行了，而不需要调色板。

但对于目前许多 4 位（16 色）或 8 位（256 色）显示系统来说，其像素值与颜色值并不一一对应，此时，调色板技术被派上了用场。

调色板的定义如下：它是在 16 色或 256 色显示系统中，由图像中出现最频繁的 16 或 256 种颜色组成的颜色表。它依靠有限种颜色通过组合来实现其它颜色。若某幅图像是使用调色板的话，那它的像素值就表示颜色在调色板查找表中的索引号，而不是颜色值。VC＋＋的 MFC 为调色板定义了相应的类 CPalette，封装了对调色板操作的一系列函数。

位图：有了调色板的概念，现在我们就可以切入正题、介绍位图了。

我们在使用计算机时要接触到大量的位图，它是指那些由一行行、一列列的像素点所组成的一个二维矩形像素点矩阵，这个矩阵存储了图像信息，组成了一幅幅的图片。可以毫不夸张地说，正是有了大量位图的存在，才使得当今的程序界面变得如此丰富多彩。

Windows 位图有两种：DDB 和 DIB。前者依赖于设备（Device Dependent Bitmap），与 MFC6.0 中的 CBitmap 类（CGdiObject 类的子类）相对应，它们在内存中的结构和位置依赖于管理它们的设备驱动程序。

DDB 不包含颜色信息，显示图像依赖于系统硬件，这使得它的优点是显示简单、迅速，但也正是由于这一特性，使得此种位图既不能通过磁盘也不能通过 Modem 将其传到其它计算机中实现共享，因此它的应用范围也就越来越小了。

与 DDB 相比，设备无关位图 DIB（Device Independent Bitmap）是一种外部图像格式，它克服了 DDB 的缺点，可以从一台主机通过网络等媒介传到其它主机上实现文件共享。DIB 之所以独立于设备，是因为它包含了一个名为 RGBQUAD 的结构，描述了 DIB 位图对应的逻辑调色板的颜色表，其像素值是该调色板中的颜色索引值，因此，当一幅 DIB 传输到另一台计算机上时，该计算机显示它时只需根据这幅 DIB 的颜色表“按图索骥”就行了，而不必关心是否与当前的硬件配置相吻合。

图 1　一幅 DIB 图像

计算机生成的所有以 .bmp 为后缀结尾的文件都代表 DIB，比如我们在 Windows 的“画图”中生成的图像（见图 1），DIB 格式有单色、16 色、256 色以及 24 位真彩色等多种。其定义如图 2 所示：

位图文件头信息(BITMAPFILEHEADER)。描述文件的类型、大小及格式
位图信息标题（BITMAPINFOHEADER）。包括信息标题主信息及调色板信息
调色板数据。描述红、绿、蓝三色分量所占比例。
实际图像数据。数据可以是压缩或不压缩的

图 2　DIB 文件结构

以上几部分在 Windows. h 当中都有相应的定义，有关 DIB 文件结构的更详细定义读者可参考有关资料。

遗憾的是，MFC 中没有专门处理 DIB 位图的类，因此，要想利用 VC++对 DIB 进行操作，那就要么直接调用 Win32 SDK 的 API 函数，要么就只能自己定义专门处理 DIB 的类。而由于相关的 API 函数也实在有限，所以一般处理位图的应用程序都采用了后一种方法。

DC：DC（Device Context）被称作设备上下文，Windows 的设备无关性使得我们无法直接对硬件进行操作，因此必须借助它来连接应用程序和设备驱动程序，以便实现诸如访问内存等功能。DC 的作用如图 3 所示：

图 3　DC 的作用

具体说来，设备上下文详尽的指定了诸如显示器、打印机、绘图机及 Modem 等物理设备的特征，是一个保存绘图界面属性的数据结构。此外，Windows 中还有一些特殊的 DC，如代表位图的设备上下文 Memory DC，只有通过它，应用程序才能在位图中绘图。

对应于 DC，VC++设计了相应的设备环境类 CDC，包括几个派生类，其相互关系及每个派生类的作用见图 4 所示。

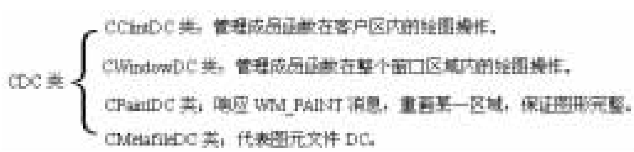


图 4　CDC 类及其派生类

GDI：最后，有关 GDI 的概念也作一下介绍。GDI 指 Windows 的图形设备接口（Graphics Device Interface），是 Windows 用来管理图像操作的一个与设备无关的模块，应用程序通过 GDI 向输出设备绘图。上面介绍的 DC 就属于 GDI 所有，其它实现位图必不可少的工具，如画笔、刷子、调色板等也都要用该模块来管理，Windows 中许多有关图像操作的 API 函数也是由 GDI 来提供的。

MFC 中的 CGdiObject 类封装了 GDI 中的部分实体（见图 5），但请注意，被封装的实体只是一部分，而并不像 CDC 类与 DC 之间有一一对应关系。

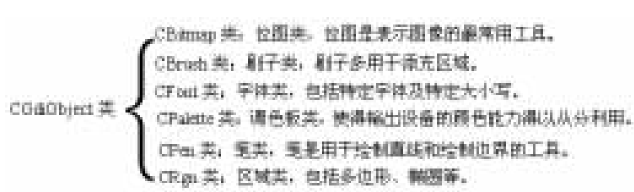


图 5　CGdiObject 类及其派生类

以上两个类共同实现了 GDI 的功能。

有了调色板、位图、DC、GDI 等几个基本概念，下面我们就可以对实现显示位图的步骤进行描述了。

三、实现方法

显示 DDB 的方法

图 6 给出了用 VC++实现显示 DDB 位图的一般方法。其中上排文字表示实现步骤，下排括号中的文字表示实现这一步骤的具体模块或函数。

图 6　显示 DDB 的一般方法

下文给出一段具体的程序代码，与图 6 相对应：

CBitmap mybitmap　　//定义 CBitmap 类的对象
CDC　mydcMemory;　　//定义 CDC 类的对象
mybitmap. LoadBitmap(IDB_MYBITMAP);　//装入 ID 号为 IDB_MYBITMAP 的位图
mydcMemory. CreateCompatibleDC(pDC);　//创建与显示 DC 相兼容的内存 DC
mydcMemory. SelectObject(&mybitmap);　//将选定的位图选入内存 DC
pDC->BitBlt(10, 10, 50, 50, &mydeMemory, 0, 0, SRCCOPY);　//从内存 DC 向显示 DC 复制. 10, 10 表示显示位置为窗口左上角, 向右向下各偏置 10; 50, 50 表示位图大小, 单位为像素; 0, 0 指出源位图的起始位置; SRCCOPY 表示直接将源位图拷贝到目的位图, 不作修改.
PDC->StretchBlt(10, 10, 50, 50, &mydeMemory, 0, 0, 100, 100, SRCCOPY);
//扩大或缩小位图. 其中 100, 100 就表示将原来 50*50 大小的位图扩大为 100*100, 其它参数与 BitBlt()中的相同.

BitBlt（）函数通过最后一个参数来表明对位图的颜色操作方式，如上例中的 SRCCOPY 意思是直接拷贝，其它还有 BLACKNESS（变为黑色）、WHITENESS（变为白色）、DESTINVERT（翻转目的位图）等多种方式，有兴趣的读者可参考相关书籍。

显示 DIB 的方法：

DIB 涉及到了调色板，所以显示 DIB 的步骤当中必须包含生成、设置以及实现调色板的内容（24 位真彩色系统除外）。

大体上来说，显示 DIB 有两种方法：

方法一　直接利用 API 函数。

法 1：可以借助 DDB 来显示 DIB，此时需要将 DIB“转换”成 DDB，因此就要先用 CreateDIBitmap（）创建一个 DDB，然后调用 SetDIBitmap（）将 DIB 拷贝到 DDB 当中，再将 DDB 选进内存 DC，最后调用 BitBlt（）函数在显示屏上显示。其总体思路与显示 DDB 的方法很类似。

法 2：还有一种方法是将位图文件直接读入内存，这时需要获得位图文件结构中的有关位图信息及显示数据（文件结构见图 2）。读入这些信息以后，如果是 24 位真彩色位图，就可以调用 StretchDIBits（）函数直接显示 DIB，而不必借助于 DDB。而如果位图不是 24 位真彩色的，那还要先处理颜色表，根据颜色表来建立位图显示调色板，再进行显示。

方法一中法 2 的实现步骤如图 7 所示。

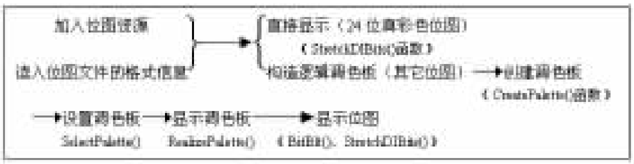


图 7　显示 DIB 的 API 方法

方法二　定制 DIB 类，用面向对象编程的方法对 DIB 进行处理。

这是最复杂也是最简单的一种办法，复杂是因为必须要建立一套完整的 DIB 处理函数库，简单是因为采用面向对象的方法在对位图进行操作时显得简单。进一步探讨定制 DIB 类不是本文的目的，许多文献中的程序实例都实现了这样的类，读者只要弄清楚了上文中描述的概念和原理，今后碰到类似的内容时就不会困惑了。

四、实例

这部分给出了一个将位图贴在按钮上的例子，因为 MFC 的 CBitmapButton 类的存在，将指定的位图与按钮联系了起来，并且封装了具体的实现代码，因此使得我们实际要做的工作很少。我们提出这样一个例子的目的是为了说明位图的一种应用。

步骤 1　在 AppWizard 下生成一个基于对话框的工程（Dialog Based Project），命名为“位图按钮”，其它全部选用默认选项。

步骤 2　生成位图资源。对于一个按钮，要生成的位图资源有两个，以便在按钮被选中以及被放开时显示不同的状态。因此，首先在“画图”程序中打开一幅自己喜欢的位图（bmp）文件，由于是用贴在按钮上，所以它不必太大（如图 8 所示），存在 BmpButton 工程下的子目录 res 下，命名为 buttonUP. bmp，作为按钮弹起时的状态；

图 8　一幅位图

对位图进行适当处理（如“图像”菜单下的“反色”算法，见图 9），以 buttonDN. bmp 为名字存在同一目录下，作为按钮弹被按下时的状态。

图 9　反色的效果

步骤 3　引入资源文件。在 VC + + 的 ResourceView 当中选择标题 Dialog Resource 并单击右键，在弹出菜单中选择 import. . . 项，然后分别从 res 目录下引入步骤二中得到的两个位图。分别将其 ID 号改为“YAHOOU”和“YAHOOD”，后者对应反色后的位图。

注意，我们这里是通过名字来标识资源，所以上述两个 ID 号一定要加引号。

步骤 4　在对话框中加入一个按钮（Button）控件，ID 号改为 IDC_YAHOO，Caption 改为 Yahoo，然后还必须将按钮的 Styles 属性（Properties）中的“Owner Draw”项选中。

步骤 5　在位图按钮 Dlg. h 的类声明部分中加入以下代码：

```
public:
    CBitmapButton m_bmpButton; //生成 CBitmapBut-
ton 类的一个对象;
```

步骤 6　在位图按钮 Dlg. cpp 的 OnInitDialog（）函数中加入以下代码：

```
VERIFY(m_bmpButton. AutoLoad(IDC_YAHOO, this));
//调用 AutoLoad()函数自动将按钮与资源联系起来; 为了在
对话框生成时完成这一工作, 这句代码加在了 OnInitDialog()
当中.
```

步骤 7　编译，运行。结果如图 10 和图 11 所示：

图 10　程序运行结果

图 11　按钮被按下去时的情形

这个按钮除了将“朴素”的外表改头换面以外，与普通按钮一样，可以实现同样的功能。

以上程序在 Visual C + + 6. 0 中调试通过。

（收稿日期：2000 年 9 月 11 日）

C++编程实现可动态创建的无限维数组

周 畅

经常用到数组变量的C++程序编写人员肯定会碰到这样的问题：当数组的维数和大小只有在程序运行后才能知道时，如何声明该数组变量呢？由于数组变量声明时必须通过显式指定其大小 如：int A[2 3]）或通过初始化 如：int A[]={{0 0} {0 0} {0 0}}）隐式指定其大小，才能使编译器知道如何为该数组分配内存，所以使用普通的数组声明无法在程序运行过程中动态创建某一事先不知道大小的数组。当然，也可以有些变通的方法，如先创建一个足够大的数组等等，但这些无疑会浪费系统资源，降低程序效率。为此，笔者利用C++构造了一个可动态创建的、维数不限的数组模板类，以解决这个问题。

一、 数组模板类 CNxArray 介绍

数组模板类CNxArray主要利用了C++ 的模板及可变数量参数这两个特性。利用模板可以使CNxArray适用于多种变量类型，如整型、浮点型等等；利用可变数量参数可以使CNxArray适用于不同维数的数组。

CNxArray有四个成员变量： m_pData、 m_nSize、m_nDimension和m_pDimension。其中m_pData指向实际数组元素占用的内存块地址；m_nSize为数组的大小，以字节为单位；m_nDimension指明数组的维数；m_pDimension相当于一个一维整型数组，其各元素代表数组模板CNxArray的各维大小。

值得注意的是，这里实际各数组元素在内存中的相对位置与普通数组完全一样。

二、 使用数组模板类 CNxArray

数组模板类CNxArray的使用格式上与普通的C语言数组稍有不同，但在方便程度上毫不逊色。使用数组模板类CNxArray时应该包含头文件NxArray.h。

首先声明一个CNxArray整型数组如下：

CNxArray <int> A DIM d1 d1

其中d1是一个一维整型数组 如：int d1[]={2 5 4}），d1的各元素代表数组A的各维大小；DIM d1 是一个宏，它计算整型数组的元素个数，在这表明A的维数。这时的A实际上是一个2*5*4的三维整型数组。然后就可以使用数组A了。CNxArray重载了符号 （）作为数组元素的索引符，我们可以用A （x，y，z）表示原来意义上的数组元素A[x][y][z]。例如下面的操作是合法的：

A （1，3，2） += 17

它相当于A[1][3][2]+=17，也就是说，我们可以为A （x，y，z）赋值或获取它的值，这样就可以把A当作普通数组那样使用了。

三、数组模板类 CNxArray 的源代码

下面是头文件NxArray.h的全部源代码 包含了数组模板类CNxArray的声明和实现）以及一个对数组模板类CNxArray实例的测试程序源代码 说明如何使用数组模板类CNxArray）。它们都已在Microsoft VC环境下调试通过。

```
///////////////////////////////////////
// File Name: NxArray.h
//////////////////////////////////////
#ifndef _NX_ARRAY_
#define _NX_ARRAY_
#ifndef ANSI
#define ANSI
#endif
#define DIM(d)    sizeof(d)/sizeof(int)
template  <class ARRAY_TYPE> class CNxArray
{
public:
        CNxArray(int nDimension, int*  pDimension);
   ~CNxArray();
protected:
        ARRAY_TYPE* m_pData;
   int     m_nSize;
   int     m_nDimension;
   int *        m_pDimension;
public:
int GetSize() { return m_nSize*sizeof(ARRAY_TYPE); };
//获取数组大小
        const int * GetDimension( int&  nDimension)
   //获取数组维数信息
                    {      nDimension  = m_nDimension;
                           return m_pDimension;       };
ARRAY_TYPE&  operator()(int d0, ...); //数组元素操作
};
///////////////////////////////////////
// Impletation of template CNxArray
///////////////////////////////////////
#include  <stdarg.h>
template <class ARRAY_TYPE>
CNxArray <ARRAY_TYPE>::CNxArray(int nDimension, int *
pDimension)
{
        m_nDimension  = nDimension;
```

```
      m_pDimension  = new int[m_nDimension +1];
      m_pDimension[m_nDimension]  = 1;
           int i  = 0;
      for(i =0, m_nSize =1; i <m_nDimension; i + +)
           {
                m_pDimension[i]  = pDimension[i];
           m_nSize  *  = m_pDimension[i];
           }
      m_pData  = new ARRAY_TYPE[m_nSize];
           for(i =0; i <m_nSize; i + +)
                m_pData[i]  = 0;
}
template <class  ARRAY_TYPE>  CNxArray <ARRAY_TYPE>: :
 ~CNxArray()
{
           delete [] m_pDimension;
      delete [] m_pData;
}
template <class ARRAY_TYPE>
ARRAY_TYPE &  CNxArray <ARRAY_TYPE>: : operator( ) (int
d0,  . . . )
{
           va_list ArrayFooter;
      int nIndex  = 0, nSubIndex  = 1, i  = 0;
      int  *  d  = new int[m_nDimension];
           d[0]  = d0;
      va_start( ArrayFooter, d0);
/ * Initialize variable arguments.  * /
           for(i =1; i <m_nDimension; i + +)
           d[i]  = va_arg( ArrayFooter, int);
    va_end( ArrayFooter ); / * Reset variable arguments. * /
           for(i =m_nDimension -1; i>=0; i - -)
           {
                nSubIndex  *  = m_pDimension[i +1];
                nIndex  + = d[i]  *  nSubIndex;
           }
      delete [] d;  / *  make sure ˝nIndex  <  m_nSize˝  * /
           return m_pData[nIndex];
}
#endif
////////////////////////////////////////
//   End of file NxArray. h
////////////////////////////////////////
//测试程序
#include  <stdio. h>
#include ˝NxArray. h˝
void main()      / *  testing CNxArray object  * /
{
      int d1[] ={2, 5, 4};
      int d2[] ={4, 3};
           int i, j, k;
           CNxArray  <int> A( DIM(d1), d1 );
      printf(˝\n CNxArray object instance A (%d Bytes): \n\
n˝, A. GetSize());
  for(i =0; i <d1[0]; i + +){
                for(j =0; j <d1[1]; j + +){
                     for(k =0; k <d1[2]; k + +){
                     A(i, j, k)  - = (i  + j  - k);
  printf(˝  A(%d, %d, %d) =% -3d˝, i, j, k, A(i, j, k));
                     }
                     printf(˝\n˝);
                }
           }
           CNxArray  <double> B( DIM(d2), d2 );
  printf(˝\n CNxArray object instance B (%d Bytes): \n\n˝,
B. GetSize());
           for(i =0; i <d2[0]; i + +){
           for(j =0; j <d2[1]; j + +){
                B(i, j)  = (i  + j  - 1. 1 )  *  1. 7;
                printf(˝  B(%d, %d) =%2. 3f˝, i, j, B(i, j));
                }
           printf(˝\n˝);
           }
  printf(˝\n\n˝);
}
```

测试程序的输出如下

```
CNxArray object instance A (160 Bytes):
A(0, 0, 0) =0    A(0, 0, 1) =1    A(0, 0, 2) =2  A(0, 0, 3) =3
A(0, 1, 0) = -1  A(0, 1, 1) =0   A(0, 1, 2) =1  A(0, 1, 3) =2
A(0, 2, 0) = -2  A(0, 2, 1) = -1 A(0, 2, 2) =0 A(0, 2, 3) =1
A(0, 3, 0) = -3  A(0, 3, 1) = -2  A(0, 3, 2) = -1 A(0, 3, 3) =0
A(0, 4, 0) = -4  A(0, 4, 1) = -3  A(0, 4, 2) = -2  A(0, 4, 3)
 = -1
A(1, 0, 0) = -1  A(1, 0, 1) =0  A(1, 0, 2) =1   A(1, 0, 3) =2
A(1, 1, 0) = -2  A(1, 1, 1) = -1 A(1, 1, 2) =0 A(1, 1, 3) =1
A(1, 2, 0) = -3 A(1, 2, 1) = -2 A(1, 2, 2) = -1 A(1, 2, 3) =0
A(1, 3, 0) = -4  A(1, 3, 1) = -3  A(1, 3, 2) = -2  A(1, 3, 3)
 = -1
A(1, 4, 0) = -5  A(1, 4, 1) = -4  A(1, 4, 2) = -3  A(1, 4, 3)
 = -2
 CNxArray object instance B (96 Bytes):
  B(0, 0) = -1. 870   B(0, 1) = -0. 170   B(0, 2) =1. 530
  B(1, 0) = -0. 170   B(1, 1) = 1. 530   B(1, 2) =3. 230
  B(2, 0) = 1. 530   B(2, 1) = 3. 230   B(2, 2) =4. 930
  B(3, 0) = 3. 230   B(3, 1) = 4. 930   B(3, 2) =6. 630
```

四、对数组模板类 CNxArray 的加强

如果有兴趣，可以容易地为数组模板类 CNxArray 或其派生类加上数组的各种操作方法，如翻转、求逆、数组的加减乘除等等，那么，它将变成一个功能强大、使用灵活、方便于各种算法研究或仿真运算的数组模板类了。

笔者已将此数组模板类 CNxArray 成功地运行于隐马尔可夫模型参数估计、伺服系统可控性研究等多个课题研究项目中，现在只是希望它也能在 C + + 编程方面起到一点抛砖引玉的作用。

收稿日期：2000 年 8 月 21 日)

在 VB 中获取目录名另一法

邓世学

笔者偶尔用 VB 编程。我们知道，在 VB 中通过交互式对话框获取目录名的方法很多，如用 DirList 控件，或用 CommonDialog 控件。但笔者最欣赏的是在 Windows 系统中如安装驱动程序指明文件路径时出现如图所示的对话框。为了获得那样的目录对话框，在 VB 中试过过许多方法，如对 CommonDialog 控件的常数等进行不同取值，用 Windows API 函数等等都无法取得，百思不得其解。

一次偶然机会，笔者打开 MicoOffice hOffice hMicos hConvert8. wiz 文件。运行文件中的宏，按下"浏览"按钮，就出现了上图中所示的对话框。于是，逐个查找按钮所对应的代码，并仔细分析后发现：之所以能够出现上图所示的对话框，是因为用了 Office 提供的 Convert8. dll 中的 SelectFolder 函数。于是根据 Convert8. wiz 的宏思路，利用自已拥有的 VB 知识，先把 Convert8. dll 拷贝到 c hwindows hsystem 目录下（注意：声明指明 Convert8. dll 的具体路径）编写获取目录对话框程序。程序的源代码如下：

```
(Attribute VB_Name = "Module1")(注: 该行代码在 VB 的模块中是不可见的)
Option Explicit
'常数取值
Global Const ONE_K_BUFF = 1024
Global Const strSEL_FLD_BUTTON_TEXT = "Select"
'SelectFolder 函数声明, 注意: 1. 文件路径根据各自位置而定; 2. 函数的变量类型
Declare Function SelectFolder Lib "c:\windows\system\Convert8.dll" (ByVal lpstrFileTitle As String, lpcbMaxFileTitle As Long, ByRef lpdwRestoreFlag As Long, ByRef lpdwX0 As Long, ByRef lpdwY0 As Long, ByVal lpstrSelectButtonText As String, ByVal hParentWnd As Long) As Long
Function SelectedPath() As String
 '获取出现目录对话框后的用户选取的目录全名
 Dim strPath As String * ONE_K_BUFF '定义目录名为定长字串
 Dim lPathSize As Long '目录名的长度
 Dim dwRestoreFlag As Long
 Dim dwX0 As Long
 Dim dwY0 As Long
 lPathSize = ONE_K_BUFF
  dwRestoreFlag = 0
 dwX0 = -1
 dwY0 = -1
 '调用定义的 Convert8. dll 中的 SelectFolder 函数
 SelectFolder strPath, lPathSize, dwRestoreFlag, dwX0, dwY0, strSEL_FLD_BUTTON_TEXT, 0
'获取目录名
SelectedPath = strPath
'去掉目录名 CHr(0)后的无用字符
SelectedPath = Mid(SelectedPath, 1, InStr(1, SelectedPath, Chr(0)) - 1)
End Function
```

Convert8. dll 文件只有 22K 多点，编写的代码也较精简，占用空间不多。有兴趣的读者不妨把它作为子过程调用，丰富 VB 程序的界面。当然因受本人知识面等限制，也许会出现上图对话框更高效、更精简、更方便的办法。

本程序在 Windows95 h98、VB5 中文版下通过。

收稿日期：2000 年 9 月 13 日）

如何编程获取 Windows NT 的性能数据

周京生

摘　要　本文较详细地介绍了如何编程获取以及计算 NT 的性能数据的方法，以处理器% Processor-Time 为例,给出了关键部分的实现源代码。

关键字　性能监视,对象

一、概述

我们知道，NT 管理工具中包括的性能监视器 Performance Monitor 是分析 Windows NT 系统性能最重要的一个工具。它是一个多用途的功能强大的监视程序，你可以用它来监视和测量本地以及网络上 NT 系统，发现系统的瓶颈。通过性能监视器，你可以了解到各种系统对象的表现情况，例如，处理器、内存、高速缓存、线程和进程等。

但在实际工程应用中，往往需要由应用程序来自动地监视系统的运行情况，以便作出相应的处理。例如，在某些分布式的网络环境中，服务器如何自动地将需要处理的任务合理地分配到若干台配置相同的工作站上？这就需要服务器了解各台工作站运行的饱和程度。对于处理器密集型的任务，工作站的处理器开销情况 Processor time 就是很重要的参数。

Windows NT 提供了两种编程获取性能数据的方法：

1 使用注册表函数访问 NT 的性能蜂巢 Performance hive HKEY_PERFORMANCE_DATA 获取性能数据。对于采集大量的性能数据以及更全面地控制采集过程的应用来说，这种方法的效率较高。Windows NT 的性能监视器就是采用这种方法实现的。

2 使用 PDH. DLL 动态链接库中的 API 函数获取性能数据。其实，PDH. DLL 中提供的 API 函数是通过实现前一种中方法来获取性能数据的，它只不过是隐藏了前种方法中很多复杂的实现细节，使其编程界面更简单、易用。对于只采集少量的性能数据时，其效率也是很高的。

二、术语

1 性能对象 ：使性能数据通过 Registry 可用的任何对象，包括 CPU、内存、高速缓存、进程等。

2 性能计数器 ：性能对象的数据采用计数器的形式表示，代表了各种测量值。例如，内存对象包含了多个计数器，分别代表自由内存量、当前正在使用的内存量等。Windows NT 共定义了两种类型的计数器：比率计数器 、原始计数器。

比率计数器：在这类计数器的名称中都包含每秒或百分号的字样，例如 % Processor Time 、Interrupts / sec 等。它们的检测都是针对一段时间进行的，至少需要两次的测量值才能计算出结果。

原始计数器：主要显示最新的测量值。例如，TCP 对象的计数器 Connections Established、Connections Active 等。

3 对象实例 ：性能对象可能会有多个实例。例如，某个系统有几个 CPU 或磁盘驱动器。为了区分相同对象的不同成员，把这些对象表示为该对象类型的不同实例。

三、访问注册表性能蜂巢获取性能数据

1. 注册表性能蜂巢

在 NT 的注册表中有一类特殊的蜂巢，它就是性能蜂巢 HKEY_PERFORMANCE_DATA。说它特殊是因为此蜂巢不同于其它的蜂巢如 HKEY_LOCAL_MACHINE、HKEY_CURRENT_USER 等，用户使用注册表编辑器无法看到此蜂巢，只能够通过 Registry API 函数来访问它。虽然 HKEY_PERFORMANCE_DATA 是一个与性能数据相关的蜂巢，但 NT 的性能数据实际上不是以硬件或软件参数的方式存储在注册表中的，程序通过使用带 HKEY_PERFORMANCE_DATA 键的 Win32 Registry API 函数使系统收集合适的系统对象管理器中的数据。

2. 收集性能数据方法

如果是收集本地系统的性能数据，可以使用带有参数 HKEY_PERFORMANCE_DATA 键的 RegQueryValueEx 函数,在完成收集工作后一定要使用 RegCloseKey 关闭。如要访问远程系统的性能数据，则必须使用带有远程系统机器名字和 HKEY_PERFORMANCE_DATA 键的 RegConnectRegistry 函数。该函数返回代表远程系统性能数据的句柄，作为随后调用 RegQueryValueEx 函数的参数 而不是使用 HKEY_PERFORMANCE_DATA 键)。

3. 性能数据结构

由 RegQueryValueEx 获取的数据基本布局如图 1 所示

・PERF_DATA_BLOCK　该结构对系统和性能数据进行描述，包括 RegQueryValueEx 返回的数据大小、被监视对象的数量、高分辨率性能计数时的频率和数值等。

・PERF_OBJECT_TYPE　对一种对象类型性能数据进行描述,包括结构的大小和跟在后面的特定数据等。

・PERF_COUNTER_DEFINITION　定义了计数器的大小、

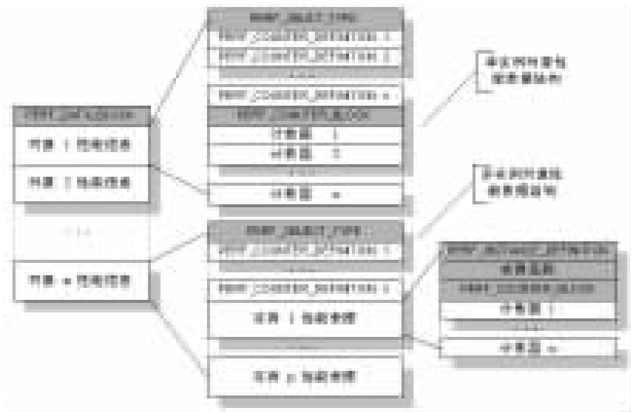


图 1 性能数据结构

类型和计数器偏移等。

·PERF_COUNTER_BLOCK 所有原始计数器的标头，定义了后跟的性能计数器的总字节长度。

·PERF_INSTANCE_DEFINITION 如果对象类型有多个实例，那么在计数器定义列表后就要跟该结构，包含了实例相关的信息。

例如：处理器对象是个多实例对象，它有 10 个计数器，包括 % Processor Time、% User Time . . . APC Bypasses / sec。在单处理器的系统中性能数据的机构如下图所示：

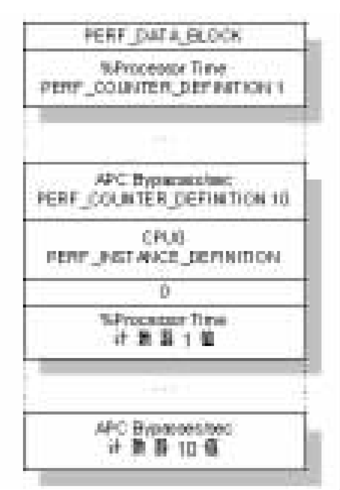


图 2 单处理器对象的性能数据块结构图

4. 程序实现

在此以获取处理器时间（% Processor Time）开销为例，演示如何通过注册表函数从注册表性能蜂巢中获取以及计算性能数据。采用这种方法需要完成下面三个步骤：

步骤一：判断是连接本地的注册表还是远程系统的注册表，如果是连接远程系统注册表要用 RegConnectRegistry 建立与其的连接。

```
#define IsLocalComputer(a , LocalComputerName) (!lstrcmp(a, LocalComputerName))
. . .
CString MachineName; //要获取性能参数的机器名
. . .
DWORD dwLength = MAX_COMPUTERNAME_LENGTH ;
::GetComputerName(LocalComputerName, &dwLength);
LPCTSTR p =MachineName.GetBuffer(MachineName.GetLength());
if(IsLocalComputer(LocalComputerName, p))
{ // 本地计算机
    hKey = HKEY_PERFORMANCE_DATA;
}
else
{ // 远程计算机
    MachineName = "\\\\" +MachineName;
LPTSTR p = MachineName.GetBuffer(MachineName.GetLength());
LONG lRet =:: RegConnectRegistry ( p, HKEY_PERFORMANCE_DATA, &hKey);
  . . .
}
```

步骤二：获取处理器对象的索引值。Windows NT 为每个性能对象及其计数器定义了一个索引值，存储在注册表的 HKEY_LOCAL_MACHINE \Software \Microsoft \Windows NT \CurrentVersion \Perlib \009 的 Counter 中。获取处理器对象的索引值，在后面调用 RegQueryValueEx 获取性能数据时使用这个索引值标明要获取的内容，就可以只得到处理器对象计数器信息数据块。本人在编程过程中发现，通过 HKEY_PERFORMANCE_DATA 也可以获取性能对象索引值。

```
char szIndex[256] = "";
XYGetIndexByName( "Processor", szIndex , hKey); // 获取 Processor 对象的索引值, hKey 步骤一中的值
   //获取性能对象索引值
   void XYGetIndexByName( char * pszCounter, char * szIndex , HKEY hkey)
   {
      char*  pszBuffer;
      char*  pszTemp;
      char  szObject[256] = "";
      DWORD  dwBytes;
      HKEY  hKeyIndex;
      int  i = 0, j = 0;
      // 得到 Counter 中值的字节长度
   RegQueryValueEx( hkey , "Counters", NULL, NULL, NULL, &dwBytes );
   pszBuffer = (char * ) HeapAlloc( GetProcessHeap(), HEAP_ZERO_MEMORY, dwBytes );
      // 获取 Counter 键中的值(索引、对象名表)
    RegQueryValueEx( hkey, "Counters", NULL, NULL, (LPBYTE)pszBuffer, &dwBytes );
      // 查找 Processor 对象的索引值
      pszTemp = pszBuffer;
      while( i != (int)dwBytes )
      {
         //得到索引值
         while ( *(pszTemp +i) != '\0')
         {
            szIndex[j] = *(pszTemp +i);
            i++;
            j++;
```

```
        }
        szIndex[j] = '\0';
        i++;
        //得到对象名
        j = 0;
        while (*(pszTemp+i) != '\0')
        {
            szObject[j] = *(pszTemp+i);
            i++;
            j++;
        }
        szObject[j] = '\0';
        i++;
        //进行比较
        if( strcmp(szObject, pszCounter) == 0 )
            break;
        //准备下一个循环
        j = 0;
        if( *(pszTemp+i) == '\0' )
        i++;
    }
    HeapFree( GetProcessHeap(), 0, (LPVOID)pszBuffer );
}
```

步骤三：根据对象的索引值获取处理器对象的% Processor Time 计数器信息，通过两次采样得到的计数器值计算处理器的开销。每次采样得到的计数器原始值并不是我们所需要的结果，要得到最终结果还需要根据计数器的类型对采样得到的原始计数器值进行计算。% Processor Time 计数器表示的是处理器执行非空闲线程所花费的时间百分比，而我们得到的计数器原始值是当前处理器执行空闲线程所花费时间的累计值（100纳秒为单位），由这个数据我们可以得到两次采样间隔之间，空闲线程执行的时间百分比。用 100% 减去空闲线程执行的时间百分比值，也就得到了我们所需要的结果。在 PERF_COUNTER_DEFINITION 结构中的 CounterType 字段中给出了每个计数器的类型，在 winperf.h 文件以及 MSDN 的帮助有具体说明。

```
COMPUTE_DATA  PreCounter; //前一次采样值
COMPUTE_DATA  CurrentCounter; //当前采样值
FLOAT PerfData;
// 第一次采样收集数据
XYGetRawData(&PreCounter, szIndex, hKey);
Sleep(1000); //采样间隔
// 进入采样收集循环
while(bStop)
{
    XYGetRawData(&CurrentCounter, szIndex, hKey);
    PerfData =XYGetPerfData(PreCounter, CurrentCounter);
//计算结果
    ...
    PreCounter.llRawData = CurrentCounter.llRawData;
    PreCounter.llData100NS =CurrentCounter.llData100NS;
    Sleep(1000); //采样间隔
}
    typedef struct _COMPUTE_DATA
    {
        LONGLONG llRawData;      // raw data
        LONGLONG llData100NS;    // time
    }COMPUTE_DATA;
    // 获取计数器的原始值函数
    void XYGetRawData(COMPUTE_DATA * pComputeData, LPTSTR  szIndex , HKEY hkey)
    {;
        PPERF_DATA_BLOCK  PerfDataBlock =NULL;
        PPERF_OBJECT_TYPE  PerfObjectType
        PPERF_INSTANCE_DEFINITION  PerfInstanceDefinition;
        PPERF_COUNTER_DEFINITION  PerfCounterDefinition;
        PPERF_COUNTER_BLOCK  PerfCounterBlock;
        DWORD BufferSize =BUFFERSIZE;
    PerfDataBlock =(PPERF_DATA_BLOCK)malloc(BufferSize);
        while( RegQueryValueEx( hkey, (LPTSTR)szIndex, NULL, NULL, (LPBYTE) PerfDataBlock, &BufferSize) == ERROR_MORE_DATA)
        {
            BufferSize += INCREMENT;
            PerfDataBlock = (PPERF_DATA_BLOCK) realloc (PerfDataBlock, BufferSize);
        }
        pComputeData->llData100NS = * (LONGLONG UNALIGNED *)(&PerfDataBlock->PerfTime100nSec);
        // 得到性能对象结构指针
        PerfObjectType =(PPERF_OBJECT_TYPE)
    ((PBYTE)PerfDataBlock +PerfDataBlock->HeaderLength);
        if(PerfObjectType->NumInstances)
    PerfCounterDefinition = (PPERF_COUNTER_DEFINITION)
((PBYTE)PerfObjectType +PerfObjectType->HeaderLength);
        //得到处理器对象实例 CPU0 结构指针
    PerfInstanceDefinition =(PPERF_INSTANCE_DEFINITION)
((PBYTE)PerfObjectType +PerfObjectType->DefinitionLength);
        PerfCounterBlock = (PPERF_COUNTER_BLOCK)
    ((PBYTE)PerfInstanceDefinition +PerfInstanceDefinition->ByteLength);
        // 得到%Processor Time 计数器原始值
        if (PerfCounterDefinition->CounterSize <= 4)
        {
    pComputeData->llRawData = * ((DWORD FAR *)
    ((PBYTE) PerfCounterBlock + PerfCounterDefinition->CounterOffset));
pComputeData->llRawData &= (LONGLONG)(0x0ffffffff);
        }
        else
pComputeData->llRawData = * ((LONGLONG UNALIGNED * )
    ((PBYTE)PerfCounterBlock + PerfCounterDefinition->CounterOffset));
        free(PerfDataBlock);
    }
    // 计算最终结果函数
    FLOAT XYGetPerfData(IN COMPUTE_DATA Pre_CouumputeData, IN COMPUTE_DATA Cur_CouumputeData)
    {
        FLOAT  eTimeInterval; //精确的采样间隔时间
        FLOAT  eDifference;  //两采样获取的时间差
        FLOAT  eCount ;
        // 两次采样的时间间隔(100 纳秒为单位)
```

```
eTimeInterval = (FLOAT)(Cur_CoumputeData.llData100NS - Pre_CoumputeData.llData100NS);
    if (eTimeInterval <= 0.0f) return (FLOAT) 0.0f;
    // 采样值差(100 纳秒为单位)
  eDifference = (FLOAT)(Cur_CoumputeData.llRawData - Pre_CoumputeData.llRawData);
    // 空闲线程执行所占时间百分比
    eCount = eDifference / eTimeInterval ;
    // 非空闲线程执行所占时间百分比
    eCount = (FLOAT) 1.0 - eCount ;
    // 去掉百分号
    eCount *= 100.0f ;
    if (eCount < 0.0f) eCount = 0.0f ;
    if (eCount > 100.0f ) eCount = 100.0f;
    return(eCount) ;
}
```

四、使用 PDH.DLL 动态链接库中的 API 函数获取性能数据

PDH (Performance Data Helper) 库的实现也是通过访问注册表获取性能参数，只不过它的 API 函数对用户屏蔽了很多的实现细节，使得编程的界面更简单、易于操作。在 VC++5.0 中只包括有 pdh.h 和 pdh.lib 而 pdh.dll 在 Win32 SDK 中有，其函数都以 Pdh 开头。使用 PDH 库获取性能数据的步骤如下：

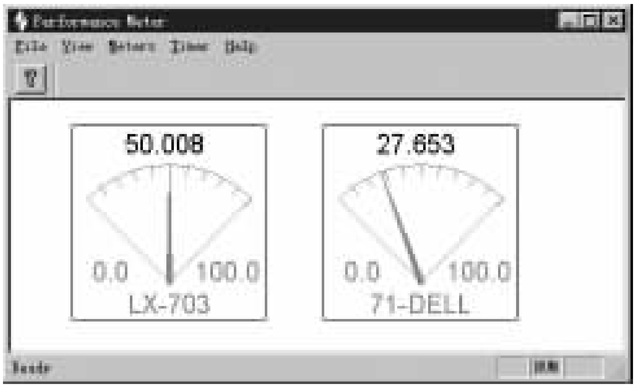
1 创建一个计数器队列，并添加要采样计数器
2 收集性能数据
3 计算、显示最终结果
4 结束性能数据收集

下面给出用 PDH 库中 API 实现上面的方法完成的获取处理器时间% Processor Time 的代码。

```
static HQUERY  hQuery = NULL;
 CString  name; //  机器名
 HCOUNTER hCounter;
 PDH_STATUS  pdhStatus;
 PDH_COUNTER_PATH_ELEMENTS  pdh_Path;
 DWORD   dwType;
 PDH_FMT_COUNTERVALUE   pValue;
 CHAR  szCounterBuffer[MAX_PATH];
 DWORD  length = MAX_PATH;
 name = "\\\\" +name;
     pdh_Path.szMachineName = name.GetBuffer(name.GetLength());
 pdh_Path.szObjectName = "Processor";
 pdh_Path.szInstanceName = "0";
 pdh_Path.szParentInstance = NULL;
 pdh_Path.dwInstanceIndex = 0;
 pdh_Path.szCounterName = "% Processor Time";
 // 产生完整的路径
 PdhMakeCounterPath ( & pdh_Path, szCounterBuffer, & length, 0);
 // 初始化打开 (1 )
 pdhStatus = PdhOpenQuery (NULL, 0, & hQuery);
 // 添加要采样的计数器
 pdhStatus = PdhAddCounter (hQuery, szCounterBuffer, 0, & hCounter);
 ...
 // 第一次采样数据(2)
 pdhStatus = PdhCollectQueryData (hQuery);
 Sleep(1000); // 采样间隔
 // 采样数据循环
 while(bStop)
 {
    // 采样
    pdhStatus = PdhCollectQueryData (hQuery);
    ...
    // 计算最终结果(3)
    // PdhGetFormattedCounterValue 函数针对最近的采样结果进行计算
    pdhStatus = PdhGetFormattedCounterValue( hCounter, PDH_FMT_DOUBLE, & dwType, & pValue);
      if(pValue.doubleValue <0)   pValue.doubleValue = (FLOAT)0.0f;
    ...
    Sleep(1000); // 采样间隔
 }
 // 结束收集 (4)
 PdhCloseQuery(hQuery);
```

五、结束语

本人用上面两种方法，在 Windows NT Server 4.0 上用 VC++5.0 实现了一个可以监视本地和网络上 NT 计算机% Processor Time 的程序，如下图所示，两个仪表可以同时监视两台计算机。在具体实现过程中，还需要注意性能数据的采样工作最好在单独的线程中完成，并且整个进程及采样线程要有优先级 (进程设为 HIGH_PRIORITY_CLASS、线程为 THREAD_PRIORITY_HIGHEST) 。在监视网络上的计算机时，要先建立与该机器的连接，即具有访问该机器共享资源的权利。



六、Demo 程序使用条件及环境

环境： Windows NT Server 4.0 VC++5.0。pdh.dll Win32 SDK 4 Demo 例程 pdh 目录中带有该动态链接库 将这个文件放在 VC 的 bin 目录中。如果在 VC+6.0 中使用，则要将 pdh 目录中的 pdh.dll、pdh.h、pdh.lib 分别放在 VC 的

实现多重条件综合查询的方法

杨晓明

摘　要　本文介绍了两种通用性十分良好的数据查询方法。第一种是在常规的数据录入、浏览模块中加入查询功能,使数据很好地定位到需要的记录。第二种是通用的数据查询模块,不加任何变动就可以加到 VFP 数据管理系统中去。两种方法都可对被查数据表的任意个字段输入条件,使用十分方便。

主题词　数据库,综合查询

任何一个管理系统都需要数据查询功能。查询一般使用在两种场合，一是在数据的录入、浏览模块中，使用查询功能使浏览的数据限制在一定的条件下，或者在录入时把数据定位在某一条记录上，把这条记录的某些有用的信息带到新的记录中去，以减少录入的工作量。这在设备管理、计量管理等系统中是十分必要的。二是查看符合一定条件的数据，从一定意义上说，这是一种真正的查询。两种查询在本质上是一样的，可以用一样的方式实现。但由于使用场合不同，在细节的处理上会有些差别。在以往看到的文章中，介绍的方法都是对数据表固定数目（两、三个）的字段输入查询条件，而不能任意选择数据表的多个字段输入查询条件进行查询。本文向大家介绍实现对数据表的任意字段进行查询的方法，供参考。

任何软件都不太可能十分完善，在管理中常常需要使用单命令方式处理一些事情，所以本人不喜欢使用汉字作字段名，对字段意义的说明，都使用称之为“结构库”的数据表，在本文中取名是在源数据表名前加“TEL”。这个表是用：COPY STRU TO ＜文件名＞ EXTE 的命令生成的。使用时修改这个表的结构，增加一个字段，字段名用“MEM”，然后用浏览命令输入各条记录的 MEM 字段的值，内容是源数据表每个字段的中文含义。在以下的介绍中也使用了这种表。对于使用汉字作字段名的表，本文的方法也是可以使用的，只需利用 AFIELDS（）、FCOUNT（）和 TYPE（）等函数，对有关的代码作一些修改。

由于本人习惯于使用全部大写字母来编写代码，所以在以下的介绍中没有遵守 VFP 的书写规则和变量命名规则，读者切勿效仿。

一、在录入、浏览中完成查询

1. 生成录查改表单 LCG. SCX

用任何一种方法生成录入、浏览的表单（不使用 GRID 控件），在表单中除了一般常有的录入、修改、退出、上一条记录、下一条记录、第一条记录和最后一条记录等命令按钮外，增加两个按钮：浏览和查询。

表单的 INIT 事件代码：

```
PARA  DB, TJ
CLOS DATA
PUBL RECC
SELE 1
USE & DB          & & 打开源数据表
SELE 2
USE TEL& DB       & & 打开对应的´结构库´
RECC = RECC( )
THISFORM. TEXT1. CONTROLSOURCE = ´& DB. . FLDM´
THISFORM. TEXT2. CONTROLSOURCE = ´& DB. . JLBH´
.
.
SELE & DB
```

bin、include 和 lib 目录中，重新编译、链接。

使用方法：先在 Meters 菜单中选择监控面板，与所要连接的机器建立连接（本地或网络），如下图示。然后在 Timer 菜单中选择 Start，就可以进行监视了。

参考文献

1. [美]Mark T Edmead 等著 金甄平译． Windows NT — 性能监视、测试和调试　电子工业出版社 1999

2. [美]Art Baker 著 科欣翻译组译． Windows NT 设备驱动程序设计指南　机械工业出版社 1997

3. [美]Peter D Hipson 著 朱友芹等译． Windows NT4 注册表专家指南　机械工业出版社 1997

4、Microsoft MSDN 帮助．

（收稿日期：2000 年 9 月 28 日）

```
SET FILT TO & TJ
GO TOP   && BOTT，视各人喜欢
```

上述代码中 DB 代表表单所要操作的源数据表表名，TJ 是查询条件。调用此表单时用：

```
DO FORM <表单名> WITH <源数据表名>,′′
```

THISFORM. TEXT1. CONTROLSOURCE = & DB..FLDM 中的"FLDM"是源数据表的第一个字段名，往下一一列出各控件的 CONTROLSOURCE 属性。如果您统一使用文本框控件，这段代码可简化成：

```
FOR I =10 TO RECC
  GO I
  NAME =TRIM(FIELD_NAME)
  NN =ALLT(STR(I))
  THISFORM. TEXT& NN. . CONTROLSOURCE =′& DB. . & NAME′
ENDFOR
SELE & DB
SET FILT TO & TJ
GO TOP
```

常用按钮的 CLICK 事件代码不在这里介绍，只列出浏览、查询和录入按钮的 CLICK 事件代码。

浏览按钮的作用是解除对源数据表的过滤，回到正常的浏览状态，它的 CLICK 事件代码：

```
SET FILT TO
FOR I =1 TO RECC
  NN =ALLT(STR(I))
  THISFORM. TEXT& NN. . READONLY =. T.
ENDFOR
```

如果不是全部使用文本框控件，FOR - ENDFOR 的整段代码不能使用，要逐个将各控件的 READONLY 属性设置为真，浏览状态一般不允许对数据进行修改。

查询按钮的作用是打开查询表单（UJCX. SCX），开始输入查询条件，它的 CLICK 事件代码：

```
SET FILT TO   && 为多次查询解除过滤
RELEAS THISFORM
CLEA EVENT
DO FORM UJCX   && UJCX 是查询表单的名称
```

录入按钮的作用是添加记录，它的 CLICK 事件代码没有什么特殊，只要在常规的代码中加入：

```
SET FILT TO
THISFORM. TEXT1. SETFOCUS
FOR I =1 TO RECC
  NN =ALLT(STR(I))
  THISFORM. TEXT& NN. . READONLY =. F.
ENDFOR
```

2. 生成查询表单（UJCX. SCX）

将 LCG. SCX 和 LCG. SCT 两个文件分别拷贝成 UJCX. SCX 和 UJCX. SCT。将原表单中的用于录入数据的控件全部改成文本框控件。删除所有的按钮，另建一个 OK 按钮，用于退出查询表单。OK 按钮的 CLICK 事件代码：

```
SET SAFE OFF
TJ =′′
SELE 2
SET EXAC OFF
FOR I = 1 TO RECC()
  NN =ALLT(STR(I))
  CTEXT& NN =THISFORM. TEXT& NN. . VALUE))))
  IF !EMPTY(CTEXT& NN)
    GO I
    LYPS& NN =FIELD_TYPE
    L& NN =FIELD_LEN
    LZD& NN =TRIM(FIELD_NAME)
    DO CASE
      CASE CTEXT& NN =″>=″
          FH =″>=″
          CTEXT& NN =SUBSTR(CTEXT& NN, 3, L& NN
      CASE CTEXT& NN =″<=″
          CTEXT& NN =SUBSTR(CTEXT& NN, 3, L& NN
          FH =″<=″
      CASE CTEXT& NN =″>″
          CTEXT& NN =SUBSTR(CTEXT& NN, 2, L& NN
          FH =″>″
      CASE CTEXT& NN =″<″
          CTEXT& NN =SUBSTR(CTEXT& NN, 2, L& NN
          FH =″<″
      OTHERWISE
          FH =″=″
    ENDCASE
IF LYPS& NN$″Cc″OR LYPS& NN$″Dd″ OR LYPS& NN$″Ll″
      CTEXT& NN =″′″ +TRIM(CTEXT& NN) +″′″
      FH =′=′      && TJ 中的符号
    ENDIF
    TJ =TJ +LZD& NN +FH +CTEXT& NN +″. AND. ″
  ENDIF
ENDFOR
SELE 1
LENGTH1 =LEN(TRIM(TJ))
IF LENGTH1>0
  TJ =SUBSTR(TJ, 1, LENGTH1 -5)   && 去掉最后一个′. AND. ′
  COUN TO S FOR & TJ
  IF S =0
    WAIT WIND ′无满足条件的记录′
  ELSE
    WAIT WIND ″共″+STR(S, 4) +′条记录！′
    RELEAS THISFORM
    CLEA EVENT
    DO FORM LCG WITH <源数据表名>, TJ
  ENDIF
ELSE
  COUN TO S
  WAIT WIND ″共″+STR(S, 4) +′条记录！′
  RELEAS THISFORM
  CLEA EVENT
  DO FORM LCG WITH <源数据表名>,′  ′
ENDIF
```

当您用 DO FORM LCG WITH <源数据表名>′ ′运行录查改表单时，点击'查询'按钮，程序就转向执行 UJCX 表

单。您可以对任意的文本框控件，即任意的字段输入条件，然后点击 OK 按钮（当然可以不输入条件，直接点击 OK 按钮），程序重新转到执行录查改表单，但此时的数据已经是经过过滤的数据。点击上一条记录、下一条记录等按钮时，只有满足查询条件的记录才出现在表单中。如果要撤销过滤，可以点击'浏览'按钮。在输入查询条件时，数值型字段可以输入大于、小于、大于等于和小于等于等符号。

二、通用查询表单

通用查询表单我们取名 ZHCX. SCX，在表单中建立若干（90 个以内）文本框控件和相同个数的标签控件，一个 GRID 控件，一个开始按钮，一个 OK 按钮和一个退出按钮。

所有文本框控件的 CONTROLSOURCE 属性和 VALUE 属性为'空'，VISIBLE 属性为'假'，WIDTH = 49，HEIGHT = 25，NAME 属性要自成体系地连续命名，并从 1 开始，否则要修改有关循环的初值和终值（下同）。

所有标签控件的 CAPTION 属性为空，VISIBLE 属性为'假'，NAME 属性也自成体系地连续命名。

开始按钮的功能主要是在每次查询、浏览后为下次的查询准备条件，本文使用形状控件 SHPAE5，它的 CLICK 事件代码为：

```
THISFORM. REFRESH
THISFORM. GRID1. VISIBLE =. F.
SELE 2      & & 打开´结构库´
FOR N =1 TO RECC()
  GO N
  NN =ALLT(STR(N))
  THISFORM. TEXT& NN. . VISIBLE =. T.
  THISFORM. TEXT& NN. . VALUE =SPAC(20)  & & 20 根据输入查询字符的多少而定
  THISFORM. ZDLABEL& NN. . VISIBLE =. T.
  THISFORM. ZDLABEL& NN. . AUTOSIZE =. T.
ENDFOR
THISFORM. GRID1. RECORDSOURCE =´& DBNAME1´
THISFORM. GRID1. COLUMNCOUNT =RECC()
FOR I =1 TO RECC()
   GO I
   MEMQ =TRIM(MEM)
   NN =ALLT(STR(I))
 THISFORM. GRID1. COLUMN& NN. . HEADER1. CAPTION =´& MEMQ´
ENDFOR
THISFORM. TEXT1. SETFOCUS
```

表单的 INIT 事件代码：

```
PARA DB1, DB2
PUBL DBNAME1, DBNAME2
DBNAME1 =DB1
DBNAME2 =DB2
SELE 1
USE & DB1    & & 被查询的源数据表
SELE 2
USE & DB2    & & 被查询的数据表的´结构表´
DO CASE
  CASE RECC() < =40
    VDTL =1
  CASE RECC()>40 AND RECC() < =60
    VDTL =2
  CASE RECC()>60 AND RECC() < =90
    VDTL =3
  OTHER
    VDTL =4
ENDCASE
IF VDTL =1
  NTOP =13
  NTOP1 =15
ELSE
  NTOP =17
  NTOP1 =20
ENDIF

FOR N =1 TO RECC()
  GO N
  NN =ALLT(STR(N))
  DO CASE    & & 以下的代码只列出了 RECC() < =90 的情况
    CASE VDTL =1 OR VDTL =2
        AAA =MOD(N, 4)
NLEFT =IIF(AAA =0, 505, IIF(AAA =3, 346, ;
IIF(AAA =2, 185, 22)))
NLEFT1 =IIF(AAA =0, 562, IIF(AAA =3, 400, ;
IIF(AAA =2, 238, 76)))
        IF VDTL =1
          NTOPC =30
        ELSE
          NTOPC =20
        ENDIF
        IF AAA =1
          NTOP =NTOP +NTOPC
          NTOP1 =NTOP1 +NTOPC
        ENDIF
        IF VDTL =1
          THISFORM. TEXT& NN. . HEIGHT =25
          THISFORM. TEXT& NN. . FONTSIZE =14
          THISFORM. TEXT& NN. . WIDTH =49
          THISFORM. ZDLABEL& NN. . HEIGHT =23
          THISFORM. ZDLABEL& NN. . FONTSIZE =14
        ELSE
          THISFORM. TEXT& NN. . HEIGHT =19
          THISFORM. TEXT& NN. . FONTSIZE =9
          THISFORM. TEXT& NN. . WIDTH =41
          THISFORM. ZDLABEL& NN. . HEIGHT =17
          THISFORM. ZDLABEL& NN. . FONTSIZE =8
        ENDIF
    CASE VDTL =3
        AAA =MOD(N, 6)
NLEFT =IIF(AAA =0, 572, IIF(AAA =5, 462, IIF(AAA =4, 352, ;
IIF(AAA =3, 242, IIF(AAA =2, 132, 22)))))
```

```
NLEFT1 =IIF(AAA =0, 615, IIF(AAA =5, 505, IIF(AAA =4, 395, ;
IIF(AAA =3, 285, IIF(AAA =2, 175, 65)))))
          NTOPC =20
          IF AAA =1
             NTOP =NTOP +NTOPC
             NTOP1 =NTOP1 +NTOPC
          ENDIF
          THISFORM. TEXT& NN. . HEIGHT =19
          THISFORM. TEXT& NN. . FONTSIZE =9
          THISFORM. TEXT& NN. . WIDTH =41
          THISFORM. ZDLABEL& NN. . HEIGHT =17
          THISFORM. ZDLABEL& NN. . FONTSIZE =8
OTHER
 * RECC()>90 时的代码, 略
   ENDCASE
   THISFORM. TEXT& NN. . VISIBLE =. T.
   THISFORM. TEXT& NN. . LEFT =NLEFT
   THISFORM. TEXT& NN. . TOP =NTOP
   THISFORM. TEXT& NN. . VALUE =SPAC(20).1
   MEMQ1 =TRIM(MEM)
   THISFORM. ZDLABEL& NN. . CAPTION =MEMQ1
   THISFORM. ZDLABEL& NN. . VISIBLE =. T
   THISFORM. ZDLABEL& NN. . TOP =NTOP1
   THISFORM. ZDLABEL& NN. . LEFT =NLEFT
   THISFORM. ZDLABEL& NN. . AUTOSIZE =. T.
ENDFOR
THISFORM. GRID1. RECORDSOURCE =´& DB1´
THISFORM. GRID1. COLUMNCOUNT =RECC()
FOR I =1 TO RECC()
    GO I
    MEMQ =TRIM(MEM)
    NN =ALLT(STR(I))
   THISFORM. GRID1. COLUMN &  NN. . HEADER1. CAPTION
 = ´& MEMQ´
ENDFOR
```

OK 按钮 (本文用形状控件 SHPAE6) 的功能是结束查询条件的输入，开始处理查询条件，按条件对数据表进行过滤，并运行 GRID 控件，它的 CLICK 事件代码：

```
TJ =´´
SELE 2
RECC =RECC()
SET EXAC OFF
FOR I =  1 TO RECC
 * 循环体内的语句和前面的一样, 略.
ENDFOR
SELE 1
 * 以下一段处理 TJ 的代码也和前面的一样, 略.).
SET FILT TO & TJ
SELE 2
FOR I =1 TO RECC()
    GO I
    NN =ALLT( STR(I))
    THISFORM. TEXT& NN. . VALUE =SPAC(20
    THISFORM. TEXT& NN. . VISIBLE =. F.
    THISFORM. ZDLABEL& NN. . VISIBLE =. F
ENDFOR
IF S>0
  SELE 1
  GO TOP
  THISFORM. REFRESH
  THISFORM. GRID1. VISIBLE =. T.
ELSE
  ENDFOR
  THISFORM. SHAPE5. CLICK  & & 相当于开始按钮
ENDIF
```

如果我们把循环中的如下一段代码改一下，就可以实现对字符字段的模糊查询。

源代码：

```
IF LYPS& NN$˝Cc˝OR LYPS& NN$˝Dd˝ OR LYPS& NN$˝Ll˝
CTEXT& NN =˝´˝ +TRIM(CTEXT& NN) +˝´˝
FH =´ =´
ENDIF
TJ =TJ +LZD& NN +FH +CTEXT& NN +˝. AND. ˝
```

改成：

```
DO CASE
CASE LYPS& NN$˝Dd˝ OR LYPS& NN$˝Ll˝
       CTEXT& NN =˝´˝ +TRIM(CTEXT& NN) +˝´˝
       FH =´ =´
TJ =TJ +LZD& NN +FH +CTEXT& NN +˝. AND. ˝ ˝ ˝
CASE LYPS& NN$˝Cc˝
CTEXT& NN =˝´˝ +TRIM(CTEXT& NN) +˝´˝
FH =´$´
TJ =TJ +CTEXT& NN +FH +LZD& NN +˝. AND.
OTHER
TJ =TJ +LZD& NN +FH +CTEXT& NN +˝. AND.
ENDCASE
```

退出按钮的作用是退出查询表单，其代码就不再介绍。

GRID 控件的属性设置：

```
Readonly =. T.     & & 在查询中一般不允许修改数据
Recordsourcetype =1 –别名
Columncount = –1
```

GRID 控件的属性必须设置正确。

三、结束语

为了节省篇幅，本文只介绍了表单设计中最基本的内容，读者可以进一步完善和美化自己的表单。通用查询模块生成后可以直接调用，调用方法：DO ZHCXWITH < 源数据表名> <对应的 结构数据表 名>。数据表的字段不能多于 90 个，否则要适当修改相关的代码，见代码中的说明。

所介绍的方法中，所有输入的查询条件均是必须满足的，即每个条件间的关系全是 ‘与’，这可以从相关的代码中看出来。由于输入查询条件的控件均是文本框控件，可以输入 ‘AND’或 ‘OR’，所以只要修改处理 ‘TJ’变量的有关代码，就可以改变查询条件的关系，做到有的条件是 ‘与’，有的是 ‘或’。在现在的情况下，输入查询条件时只能输入 ‘>’， ‘<’， ‘>=’等符号，否则将查不到满足条件的记

Java 对象系列化技术在分布式数据库中的应用

辛明海

摘　要　Java 提供了对象系列化技术，利用此技术把数据对象系列化在网络上传输，实现多层分布式数据库应用中数据对象的迁移，可以减少网络通信，提高系统性能，并且所设计的程序简洁明了，易于实现。以一个简单的三层数据库应用为例，详细描述了如何利用 Servlet 和对象系列化技术来设计多层数据库应用。对象系列化在软件工程应用中具有非常积极的意义，不仅适用于多层数据库的应用，也同样适用于其他方面的软件设计，具有广阔的应用前景。

关键字　对象系列化，分布式，Servlet

一、引言

从 JDK1.1 开始 SUN 提供了一种新的机制叫作对象系列化（Object Serialization），它可以把任何实现了 Serializable 接口的对象转换为字节流，而该字节流可以在以后还原为原来的对象，这一机制不仅可以应用于单机上，而且可以被应用到网络上，这样对象可以在任一台机器上被系列转化为字节流，通过网络传送到另一台机器上，再恢复成原来的对象以供使用。在通常的分布式三层数据库应用中，中间层负责向后台数据库提交客户端的查询请求，同时把查询结果以结果集（Resultset）的形式返回给客户端，在需要时中间层会格式化结构集再返回给客户端，比如，利用 Servlet 作中间层则客户端接收的是格式化后的 HTML 形式。利用类似方法的弊端是对结果集的处理麻烦，例如游标的定位或者图片的显示。JBUIDER3.5 作为企业级的 RAD，提供了功能强大的 DataExpress Component Library 及 JavaBeans 组件库。DataExpres 组件库具有数据库连接，数据自动感知功能，更重要的是它提供了将数据对象系列化的功能。中间层把数据对象系列化后传递给客户端，客户端反系列化，把数据对象提供给数据库 JavaBeans 组件，数据库 JavaBeans 组件能够自动绑定，不用编写任何代码即可以实现数据的显示、游标定位、查询、更新等功能。同时由于在网络上传递的是对象形式，能够减轻网络通信，提高系统性能。

二、对象系列化的实现原理

JBuiler3.5 为了实现数据对象的系列化，精心设计了 DataExpress 组件库，利用该组件库，可以轻易地实现数据对象系列化，以下对主要涉及到的组件或对象作简要说明。

1. DataSetDataSet

是一抽象类，具有对二维数据进行一般编辑、观察的功能，可以借组件的数据自动感知能力对 DATASET 的数据进行调整，修改 DATASET 的数据源属性使得 DATASET 与数据感知连接在一起，从而获得对外界的控制。不同组件的 DATASET 属性可以设置成相同的，即控制源或数据源是一样的情况下，控制发生了改变，受控制的组件必然作相应的同步变化，通称为数据绑定技术，这种绑定可以大大减少用户界面中界面随数据变化部分的开发。

2. StorageDataSet

StorageDataSet 继承于 DataSet，负责管理 DataSet 数据操作的存储、提供和还原。

3. QueryDataSet

QueryDataSet 是继承于 StorageDataSet，QueryDataSet 与 DataSet 组间协作从一远程数据库查询获得数据。

4. DataSetData

DataSetData 是实现数据对象系列化的关键。DataSetData 可分离 DataSet 的状态，从而抽取 DataSet 的数据。由于 DataSetData 实现了 Serializable 接口，因此是可系列化的，可用作数据流对象，或将作为传递给 RMI 调用的参数；DataSetData 从 DataSet 抽取数据后，一些列属性的元数据将被保存，如字段名、字段类型等。这型元数据信息是 DataSetData 从 DataSet 对象抽取数据后自动填充进自身的。DataSetData 的主要方法如下：

录。

本文中‘结构库”的 MEM 字段长度为 10，如果你在使用中超过此长度，要适当改变 NLEF＝和 NLEFT1＝ 语句中的各个值。

某些录入程序在录入数据时对数据没有多少限制，也无需提供帮助，但在使用过程中，数据表的字段却会增减，如工资管理系统、工资项目随时可能变动。这种情况下，对用户提供通用型的录入程序更便于使用和维护，利用本文介绍的设计思想，可以方便地做到这一点。

希望本文能起到抛砖引玉的作用。

收稿日期：2000 年 8 月 14 日）

i　public static DataSetData extractDataSet　com. dx. dataset. DataSet dataset

从 DataSet 抽取数据后，保存元数据信息到 datasetdata，抽取的数据按字段类型组织成数组。

ii　public void loadDataSet　DataSet dataset

从 DataSetData 中恢复数据，并将数据保存到 dataset。

5．Provider

Provider 是向 DataSet 提供数据的抽象类。DataSet 若要从某种特定的途径获得数据，可以继承 Provider 类，定制 Provider 的方法——public abstract void provideData　StorageDataSet data　boolean toOpen　来实现。

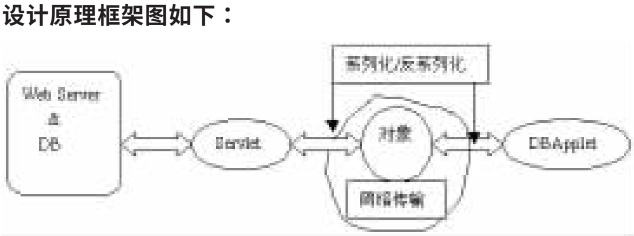
可见 DataSet 数据源的获取是非常灵活的，对于不同的数据源可以采用不同的 provideData　方法。

三、对象系列化在分布式数据库的应用

利用 Jbuilder3. 5 企业版提供的数据对象系列化技术编写分布式数据库应用，具有令开发人员意想不到的简单和快捷。以下举一个利用 Servlet 实现三层分布式应用来说明，为了清楚起见，只定制了 Provider 方法，即只能查询、浏览。利用相同的方法，定制 Resolver 方法，就能实现增、删、改。

1．程序的系统要求

在这个简单的分布式数据库应用中，客户端采用 DBApplet，能够接受 SQL 语句输入，在浏览器内显示高级复杂的用户界面；中间层采用 Servlet 接收查询语句，访问后台不同类型的数据库　这里用 ACCESS），使得用户看来是透明的，DBApplet 和 Servlet 之间实现高级的对象传输，计算由 DBApplet 和 Servlet 分担，利用 Java 提供的 URL 对象，可以打开并访问网络上的对象，其访问方式与访问本地文件系统几乎是一样的，由此实现计算的分布。主要的工作由 Servlet 来完成，设计原理框架图如下：



Servlet 接受 DBApplet 发送过来的参数，通过 JDBC 访问后台 ACCESS 数据库，将所得查询结果抽出 DataSetData 对象，然后将系列化的对象往网络上传输，网络另一端的 DBApplet 接收并反系列化对象，利用可以绑定数据的 Bean 把友好的用户界面显现给用户。由于 DBApplet 也参与了计算的协作，在某种程度上减轻了服务器的负担，并且两者以高层次的对象进行通信，避免了字节流繁琐的通信，简化了通信协议。

2．程序的主要代码实现

i）DBApplet. java

DBApplet. java 向用户提供交互界面，收集数据库查询参数，把这些参数发送给 Servlet。程序源码如下：

```
package dbapplet;
import java. awt. * ;
import java. awt. event. * ;
import java. applet. * ;
import javax. swing. * ;
import com. borland. dx. sql. dataset. * ;
import com. borland. dx. dataset. * ;
import com. borland. dbswing. * ;
import com. borland. jbcl. layout. * ;;;
public class DBApplet extends JApplet {
  boolean isStandalone = false;
  TableDataSet tableDataSet1 = new TableDataSet();
  XYLayout xYLayout1 = new XYLayout()
  JLabel jLabel1 = new JLabel();
  JTextField odbc = new JTextField();
  JLabel jLabel2 = new JLabel();
  JTextField sql = new JTextField();
  JButton jButton1 = new JButton();
  JdbNavToolBar jdbNavToolBar1 = new JdbNavToolBar();
 TableScrollPane tableScrollPane1 = new TableScrollPane();
  JdbTable jdbTable1 = new JdbTable()
  JdbStatusLabel jdbStatusLabel1 = new JdbStatusLabel();
  JLabel jLabel3 = new JLabel();
  JTextField userName = new JTextField();
  JLabel jLabel4 = new JLabel();
  JTextField password = new JTextField();
  //Get a parameter value
  public String getParameter(String key, String def) {
    return isStandalone ? System. getProperty(key, def) :
   (getParameter(key) ! = null ? getParameter(key) : def);
  }
  //Construct the applet
  public DBApplet() {
  }
  //Initialize the applet
  public void init() {
    try {
      jbInit();
    }
    catch(Exception e) {
      e. printStackTrace();
    }
  }
  //Component initialization
  private void jbInit() throws Exception {
    jLabel1. setText("ODBC");
    this. setSize(new Dimension(400, 300));
    this. getContentPane(). setLayout(xYLayout1);
    odbc. setText("Authors");
    jLabel2. setText("SQL");
    sql. setText("select * from authors");
    jButton1. setText("OK");
    jButton1. addActionListener(new java. awt. event. Action-
Listener() {
      public void actionPerformed(ActionEvent e) {
        jButton1_actionPerformed(e);
      }
```

```
  });
  jdbStatusLabel1.setText("jdbStatusLabel1");
    jdbNavToolBar1.setButtonStateInsert(JdbNavToolBar.HIDDEN);
    jdbNavToolBar1.setButtonStateDelete(JdbNavToolBar.HIDDEN);
    jdbNavToolBar1.setButtonStatePost(JdbNavToolBar.HIDDEN);
    jdbNavToolBar1.setButtonStateCancel(JdbNavToolBar.HIDDEN);
    jdbNavToolBar1.setButtonStateDitto(JdbNavToolBar.HIDDEN);
    jdbNavToolBar1.setButtonStateSave(JdbNavToolBar.HIDDEN);
    jdbNavToolBar1.setButtonStateRefresh(JdbNavToolBar.HIDDEN);
  jLabel3.setText("UserName");
  jLabel4.setText("Password");
   this.getContentPane().add(jLabel1, new XYConstraints(15, 8, 58, 16));
   this.getContentPane().add(tableScrollPane1, new XYConstraints(31, 144, 357, 129));
   this.getContentPane().add(jdbStatusLabel1, new XYConstraints(32, 277, 357, 20));
   this.getContentPane().add(jLabel2, new XYConstraints(15, 55, 49, -1));
   this.getContentPane().add(odbc, new XYConstraints(47, 9, 131, 23));
   this.getContentPane().add(sql, new XYConstraints(47, 47, 132, 24));
   this.getContentPane().add(jButton1, new XYConstraints(42, 92, 64, 40));
   this.getContentPane().add(jdbNavToolBar1, new XYConstraints(141, 91, 241, 39));
   this.getContentPane().add(jLabel3, new XYConstraints(188, 15, 67, 14));
   this.getContentPane().add(userName, new XYConstraints(262, 12, 110, 20));
   this.getContentPane().add(jLabel4, new XYConstraints(187, 52, 69, 21));
   this.getContentPane().add(password, new XYConstraints(261, 53, 113, -1));
  tableScrollPane1.getViewport().add(jdbTable1, null);
 }
 //Get Applet information
 public String getAppletInfo() {
  return "Applet Information";
 }
 //Get parameter info
 public String[][] getParameterInfo() {
  return null;
 }
 //static initializer for setting look & feel
 static {
  try {
    //UIManager.setLookAndFeel(UIManager.getSystemLookAndFeelClassName());
    //UIManager.setLookAndFeel(UIManager.getCrossPlatformLookAndFeelClassName());
   }
   catch(Exception e) {
   }
  }
  void jButton1_actionPerformed(ActionEvent e) {
   try{
   //接受客户输入的查询条件
    String queryText = "odbc=" + odbc.getText();
    queryText += "&sql=" +sql.getText();
    if (userName.getText().length() >0)
   queryText += "&username=" + userName.getText();
    if (password.getText().length() >0)
   queryText += "&password=" + password.getText();
    //设定 Provider
ClientProvider clientProvider =new ClientProvider(queryText);
    tableDataSet1.close();
    tableDataSet1.setProvider(clientProvider);
    //绑定具有数据感应能力的 Beans
    jdbStatusLabel1.setDataSet(tableDataSet1);
    jdbNavToolBar1.setDataSet(tableDataSet1);
    jdbTable1.setDataSet(tableDataSet1);
   }
   catch(DataSetException ex){
    System.out.println(ex.getMessage());
   }
  }
}
```

ii DBServlet.java

DBServlet.java 负责接受客户端发送过来的查询参数，对数据库进行查询，然后将查询结果对象系列化送回客户端的 ClientProvider 对象。程序源码如下：

```
package dbapplet;
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;
import java.util.*;
import com.borland.dx.sql.dataset.*;
import com.borland.dx.dataset.*;
public class DBServlet extends HttpServlet {
 //Initialize global variables
public void init(ServletConfig config) throws ServletException
{
  super.init(config);
 }
/**
 * 利用 JDBC 查询后台数据库,并将数据对象系列化
 * 返回 DataSetData 对象
 */
 public DataSetData provideAuthorData(String url, String username, String password, String queryText) throws DataSetException {
  Database db = new Database();
  QueryDataSet qds = new QueryDataSet();
  try {
```

```
   db.setConnection(new com.borland.dx.sql.dataset.ConnectionDescriptor(url, username, password, false, "sun.jdbc.odbc.JdbcOdbcDriver"));
     qds.setQuery(new com.borland.dx.sql.dataset.QueryDescriptor(db, queryText, null, true, Load.ALL));
     qds.open();
     DataSetData data = DataSetData.extractDataSet(qds);
     return data;
   }
   finally {
     qds.close();
     db.closeConnection();
   }
  }
  //Process the HTTP Get request
  public void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException
{
   PrintWriter out = new PrintWriter (response.getOutputStream());
   response.setContentType("text/html");
   out.println("<FONT COLOR=GREEN>");
   out.println("祝贺你!Servlet 已正常启动!");
   out.println("</FONT>");
   out.close();
  }
  //Process the HTTP Post request
  public void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException
{;
    ObjectOutputStream outputToApplet = new ObjectOutputStream(response.getOutputStream());
   String sql="", odbc="", username="", password="";
   try{
       sql = request.getParameter("sql")
       odbc = request.getParameter("odbc");
       username = request.getParameter("username");
       password = request.getParameter("password");
   }catch(Exception e){
       System.out.println("取参数失败");
   }
   System.out.println(odbc + sql);
   try{
     odbc="jdbc:odbc:" + odbc;
   DataSetData data = provideAuthorData(odbc, username, password, sql);
     outputToApplet.writeObject(data);
     outputToApplet.flush();
   }
   catch (com.borland.dx.dataset.DataSetException ex) {
    System.out.println(ex.getMessage()); // new DataSetException(ex.getMessage());
   }
   finally{
     outputToApplet.close();
   }
  }
  public DBServlet() {
    try {
      jbInit();
    }
    catch(Exception e) {
      e.printStackTrace();
    }
  }
  private void jbInit() throws Exception {
  }
}
```

iii ClientProvider.java

ClientProvider.java 接受 DBApplet 的用户输入参数，触发远程 Servlet 的 doPost，向远程 Servlet 递交参数，接受 Servlet 发送回来的结构对象，并将其反系列化，提供给 DBApplet。程序源码如下：

```
/**
 * Title:        JAVA 对象系列化在数据库中的应用<p>
 * Description:  <p>
 * Copyright:    Copyright (c) love<p>
 * Company:      hqu<p>
 * @author love
 * @version 1.0
 */
package dbapplet;
import java.io.*;
import java.net.*;
import com.borland.dx.dataset.*;
import com.borland.dx.sql.dataset.*;
/**
 *通过 URL 调用 Servlet 对象，建立流连接，
 * 传递客户参数，启动 doPost 方法
 */;
public class ClientProvider extends Provider {
  URLConnection servletcon;
  private ObjectInputStream inputFromServlet;
  public ClientProvider(String queryText) {
    String where = "http://localhost:8080/servlet/dbapplet.DBServlet";
    try{
      URL servleturl = new URL(where);
      servletcon = servleturl.openConnection();
      servletcon.setDoInput(true);
      servletcon.setDoOutput(true);
      servletcon.setUseCaches(false);
      servletcon.setDefaultUseCaches(false);
      servletcon.setAllowUserInteraction(false);
     PrintStream sqlToServlet = new PrintStream(servletcon.getOutputStream());
      sqlToServlet.println(queryText)
      sqlToServlet.flush();
      sqlToServlet.close();
    }
    catch(Exception ex){
      System.out.println(ex.getMessage());
    }
```

超联接控件的制作

罗筱波

在当今的网络时代，超联接无所不在，在各种各样的软件上我们都可以看到它的存在，这几乎成了一个必不可少的一项宣传内容。如果能在自己的程序中加入一个超联接，那么会使我们的程序带上鲜明的网络特征。但是在 VC6.0 中并没有提供一个用于制作超联接的控件，因此只有自己编写实现此功能的程序。下面是本人编写的一个实现超联接的类，希望能与大家分享。

制作过程如下：

首先从类 CStatic 派生出一个类 CsuperLinkCtrl，头文件具体代码如下：

```
////////////////////////////////////////
#if !defined(AFX_SUPERLINKCTRL1_H__EAAF218E_5962_
11D4_94CC_904165C12710__INCLUDED_)
#define AFX_SUPERLINKCTRL1_H__EAAF218E_5962_11D4_
94CC_904165C12710__INCLUDED_
#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000
// SuperLinkCtrl1.h : header file
/////////////////////////////////////////////
// CSuperLinkCtrl window
class CSuperLinkCtrl : public CStatic
{
// Construction
public:
   CSuperLinkCtrl();
     CSuperLinkCtrl(COLORREF clickedtextcor, COLORREF
moveontextcor, COLORREF ordinarytextcor);
// Attributes
public:
   COLORREF m_crText;    //超联接文本的颜色
// Operations
public:
// Overrides
   // ClassWizard generated virtual function overrides
   //{{AFX_VIRTUAL(CSuperLinkCtrl)
   //}}AFX_VIRTUAL
// Implementation
```

```
  }
/**
 * 接收 Servlet 传送来的数据对象，反系列化后提供给 DBAp-
plet 的 tableDataSet1
 */
  public void provideData(StorageDataSet dataSet, boolean
toOpen)  throws  com.borland.dx.dataset.DataSetException
{
 //TODO: implement this com.borland.dx.dataset.Provider
abstract method
    try{
inputFromServlet = new ObjectInputStream(servletcon.getIn-
putStream());
      System.out.println("clientprovider1");
   DataSetData data  =  (DataSetData)inputFromServlet.read-
Object();
      dataSet.empty();
      data.loadDataSet(dataSet);
      inputFromServlet.close();
    }{
    catch(ClassNotFoundException ex)
      System.out.println(ex.getMessage());
    }
    catch(IOException ex){
      System.out.println(ex.getMessage());
    }
    catch (DataSetException ex){
      System.out.println(ex.getMessage());
    }
  }
}
```

3. 程序的运行

程序在中文版 NT4.0，JBuilder3.5Enterprise Edition 集成环境下通过，下面介绍在 JBuilder3.5Enterprise Edition 集成环境下的运行方法。

i 在 JBuilder3.5 中选择 File|Open File 选择 DBApplet.jpr 文件。

ii 选择 Project|Project Properties，选择 RUN，选择 JSP/SHTML，设置 Start JSP file 为 DBServlet.shtml。

iii 在 Project 面板中选择 DBServlet.shtml，单击鼠标右键，选择运行启动 Servlet，一般情况下 Servlet 在本机的 8080 端口侦听，否则必须修改 ClientProvider.java 中的 where 字符串变量。

iv 在 Project 面板中选择 DBApplet.html，单击鼠标右键，选择运行启动 DBApplet，或直接双击 DBApplet.html。

四、结束语

分布式对象已得到越来越广泛的承认和应用，利用 JAVA 对象的系列化技术，可以在较高的层次上实现对象通信，增加了系统的结构性，降低了编程的难度。对于大型的分布式应用是非常有意义的。

收稿日期：2000 年 10 月 11 日）

```
public:
      BOOL  SetLinkAddress(CString linkaddress);   // 用于设置超联接的地址
      void MouseLeave(void);
      void MouseEnter(void);
      void SetLinkCursor(HCURSOR hCursor);   // 用于设置超联接的鼠标形状
      void SetTextColor(COLORREF crText);   //用于设置超联接文本的颜色
      void SetSuperLink(BOOL setlink);
      virtual ~CSuperLinkCtrl();
      // Generated message map functions
protected:
      CString m_linktext;
      BOOL m_bClicked;
      HCURSOR m_hCursor;
//    void SetTextColor(COLORREF crText);
      //{{AFX_MSG(CSuperLinkCtrl)
afx_msg HBRUSH CtlColor(CDC* pDC, UINT nCtlColor);
afx_msg void OnLButtonDown(UINT nFlags, CPoint point);
      afx_msg BOOL OnSetCursor(CWnd* pWnd, UINT nHitTest, UINT message);
 afx_msg void OnMouseMove(UINT nFlags, CPoint point);
      //}}AFX_MSG
      DECLARE_MESSAGE_MAP()
private:
      BOOL m_bcaptured;
COLORREF m_clickedtextcor;   //单击超联接后文本的颜色
      COLORREF m_moveontextcor;    //鼠标移动到超联接时的文本的颜色
COLORREF m_ordinarytextcor; //未激活超联接时文本的颜色
      BOOL m_bclicked;
      LOGFONT     m_lf;
      CFont       m_font;    //超联接上的文本的字体
};
////////////////////////////////////////
//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ will insert additional declarations immediately before the previous line.
#endif // !defined(AFX_SUPERLINKCTRL1_H__EAAF218E_5962_11D4_94CC_904165C12710__INCLUDED_)
////////////////////////////////////////
```

类的实现文件如下：

```
// SuperLinkCtrl1.cpp : implementation file
//
#include "stdafx.h"
#include "SuperLinkCtrl1.h"
#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif
////////////////////////////////////////
// CSuperLinkCtrl
CSuperLinkCtrl::CSuperLinkCtrl()
{;
   m_bClicked = FALSE;
   m_bcaptured = FALSE;
   m_clickedtextcor = RGB(0, 255, 255)
   m_moveontextcor = RGB(0, 255, 0);
   m_ordinarytextcor = RGB(0, 0, 255);
    m_crText = m_ordinarytextcor;
   ::GetObject((HFONT)GetStockObject(DEFAULT_GUI_FONT), sizeof(m_lf), &m_lf);
    m_lf.lfUnderline = TRUE;
   BOOL bCreated = m_font.CreateFontIndirect(&m_lf);
   ASSERT(bCreated);
   m_hCursor = NULL;
}
CSuperLinkCtrl::CSuperLinkCtrl(COLORREF clickedtextcor, COLORREF moveontextcor, COLORREF ordinarytextcor)
{;
   m_bClicked = FALSE;
   m_bcaptured = FALSE;
   m_clickedtextcor = clickedtextcor
   m_moveontextcor = moveontextcor;
   m_ordinarytextcor = ordinarytextcor;
    m_crText = m_ordinarytextcor;
   ::GetObject((HFONT)GetStockObject(DEFAULT_GUI_FONT), sizeof(m_lf), &m_lf);
    m_lf.lfUnderline = TRUE;
   BOOL bCreated = m_font.CreateFontIndirect(&m_lf);
   ASSERT(bCreated);
    m_hCursor = NULL;
}
CSuperLinkCtrl::~CSuperLinkCtrl()
{
   m_font.DeleteObject();
}
BEGIN_MESSAGE_MAP(CSuperLinkCtrl, CStatic)
   //{{AFX_MSG_MAP(CSuperLinkCtrl)
   ON_WM_CTLCOLOR_REFLECT()
   ON_WM_LBUTTONDOWN()
   ON_WM_SETCURSOR()
   ON_WM_MOUSEMOVE()
   //}}AFX_MSG_MAP
END_MESSAGE_MAP()
////////////////////////////////////////
// CSuperLinkCtrl message handlers
HBRUSH CSuperLinkCtrl::CtlColor(CDC* pDC, UINT nCtlColor)
{
   // TODO: Change any attributes of the DC here
   if (CTLCOLOR_STATIC == nCtlColor)
   {
        pDC->SelectObject(&m_font);
        pDC->SetTextColor(m_crText);
        pDC->SetBkMode(TRANSPARENT);
   }
       HBRUSH hBrush = ::CreateSolidBrush(GetSysColor(COLOR_3DFACE));
   // return a handle to the brush that is to be used for painting the control background.
```

```
    return hBrush;  // here must not return null
}
void CSuperLinkCtrl::SetTextColor(COLORREF crText)
{
    m_crText = crText;
    RedrawWindow();
    return;
}
void CSuperLinkCtrl::OnLButtonDown(UINT nFlags, CPoint point)
{
    // TODO: Add your message handler code here and/or call default
    m_bClicked = TRUE;
    m_crText = m_clickedtextcor;
    RedrawWindow();
    if(m_linktext! = "")
  ShellExecute(NULL, "open", m_linktext, NULL, NULL, SW_SHOWNORMAL);
    CStatic::OnLButtonDown(nFlags, point);
}
```

// **下面** function **是使** static **控件能响应消息的关键(因为通常** static **控件只能响应很少一部分消息)**

```
void CSuperLinkCtrl::SetSuperLink(BOOL setlink)
{
    if (setlink)   ///  注意:此处必须加入 ModifyStyle(),使 static 控件能响应 mouse 的消息
        ModifyStyle(0, SS_NOTIFY);
    else
        ModifyStyle(SS_NOTIFY, 0);
}
BOOL CSuperLinkCtrl::OnSetCursor(CWnd * pWnd, UINT nHitTest, UINT message)
{
    // TODO: Add your message handler code here and/or call default
    if (m_hCursor)
    {
        ::SetCursor(m_hCursor);
        return TRUE;
    }
return CStatic::OnSetCursor(pWnd, nHitTest, message);
}
void CSuperLinkCtrl::SetLinkCursor(HCURSOR hCursor)
{
        m_hCursor = hCursor;
}
void CSuperLinkCtrl::OnMouseMove(UINT nFlags, CPoint point)
{
    // TODO: Add your message handler code here and/or call default
    SetCapture();  // capture the mouse
    POINT mpoint;
    GetCursorPos(&mpoint);  // get the mouse' position
    CRect rect;
    GetWindowRect(&rect);
    if(!rect.PtInRect(mpoint)) //if the course isn't in the control'rectangle
    {
        m_bcaptured = FALSE;
        ReleaseCapture();       // release the capture
        MouseLeave();
        return;
    }
    if(m_bcaptured)  // if the mouse has already captured
        return;
    m_bcaptured = TRUE;
        MouseEnter();
    ReleaseCapture();
    CStatic::OnMouseMove(nFlags, point);
})
void CSuperLinkCtrl::MouseEnter(
{
m_crText = m_bClicked? m_clickedtextcor : m_moveontextcor;
    RedrawWindow();
})
void CSuperLinkCtrl::MouseLeave(
{
m_crText = m_bClicked? m_clickedtextcor : m_ordinarytextcor;
    RedrawWindow();
}
BOOL CSuperLinkCtrl::SetLinkAddress(CString linkaddress)
{)
    if(linkaddress.Find("http")! = -1
        m_linktext = linkaddress;
    else if(linkaddress.Find("@")! = -1)
        m_linktext = "mailto: " + linkaddress;
    else
    {
        MessageBox("Error: wrong superlink format");
        m_linktext = "";
        return FALSE;
    }
    return TRUE;
}
```

类 CsuperLinkCtrl 完成后便可在其它程序中使用了。

使用方法有两种：一是用 classwizard 使一个 static 控件与一个 CsuperLinkCtrl 变量相关联，另一种方法是定义一个 CsuperLinkCtrl 变量，然后用 SubclassDlgItem　使它与一个 static 控件相关联。

使用示例：

首先将 SuperLinkCtrl1.h 和 SuperLinkCtrl1.cpp 加入到工程的目录中，然后定义一个变量 CSuperLinkCtrl m_test　然后在对话框的 OnInitDialog　函数中加入如下代码即可。

```
m_test.SubclassDlgItem(IDC_STATICTEST, this);
m_test.SetSuperLink(TRUE);
m_test.SetLinkCursor(AfxGetApp()->LoadCursor(IDC_CURSORHAND));
m_test.SetLinkAddress("xxxx@citiz.net");
```

怎么样，赶紧用鼠标点一下吧！

收稿日期：2000 年 8 月 15 日)

应用 ASP 技术编写一个简单论坛

刘小环

一、论坛的应用技术和支持环境

论坛，即讨论组，是基于网络技术的 WEB 应用。人们可以在上面按一定的组织形式发布各自的问题和观点、进行非实时性交流。笔者在此给出一个编写简单论坛的方法，读者若有兴趣可以一试。

本论坛采用 ASP Active Server Pages 技术编写，后台数据库为简单起见，采用 Microsoft Access。在 95/98 平台上，需要安装 Microsoft Personal Web Server，在 NT 平台则使用 Internet Information Server 调试可用 Peer Server ，以此来设置你的网站的目录和名称，并能运行和调试你的 ASP 应用程序。笔者对数据库接口 ODBC 的配置是

数据库名称：forum. mdb

数据库与 ASP 应用程序在同一目录下，在此，均放在网站主目录下的子目录 /forum 中。

ODBC 中用户数据源 DSN 名为：forum，选取 forum. mdb 作为数据库。

登录名称：user

登录密码：无

配置好 ODBC 数据源后，启动你的 PWS 或 IIS，即可开始运行你的 ASP 应用程序。

二、论坛的形式

本论坛充分利用了浏览器的框架特性 frames，当然，必须浏览器支持，IE4. 0 以上即可 ，将页面划分成三部分，如下图 1 所示。该页面由顶框架 TOP、左框架 LEFT 和主框架 MAIN 三个部分组成：

图 1

●顶框架：导航栏，在整个论坛的使用和浏览期间始终不变，包括：

[返回主页]——从论坛直接退回网站主页；

[刷新内容]——对论坛进行刷新，读取最新消息；

[发　　送]——在主框架中显示发送消息表单。

●左框架：论坛目录和消息的列表，包括：

论坛目录——显示论坛组织的讨论主题，如本例中的六种，足球、篮球、排球、网球、羽毛球、乒乓球；

消息列表——按讨论主题分类显示消息，并且将消息与其答复消息组织在一起，并采用向右缩格方式表示其答复关系，如图 1 中足球主题中的一则消息“马明宇走得好！！！”，其答复消息为“RE：同意”。

●主框架：消息内容显示和发送消息区，包括：

消息发送页面——进入论坛或点击导航栏的[发送]，显示一组由文本框和选择框组成的表单，填写相应内容后，点击“发送内容”按钮发送消息。消息发送成功会显示发送成功页面；

消息内容显示页面——选择点击左框架消息列表中的一项，在主框架中会显示该消息的标题、主题、提交作者、提交人邮箱、提交日期、提交时间、提交内容。在该页面中还有一个按钮“答复”，用于答复所浏览的消息；

消息答复页面——在消息内容显示页面中点击“答复”按钮，此时显示一个表单，填写相应内容后，点击“发送内容”按钮发送消息。消息发送成功会显示发送成功页面。答复消息将显示在所答复的消息下方，如消息列表中所述。

三、论坛的程序结构和数据库设置

3. 1 论坛的文件清单

本论坛由以下几个文件 均在 forum 子目录中 组成：

myforum. html——本论坛的进入主页，仅起引导作用；

discuss. asp——论坛的主页，包括三个框架，由 discuss_top. html、discuss_left. asp 和 discuss_main. asp 组成，如图 1 所示即为 discuss. asp 页面。本 asp 预统计每个论坛主题的消息总数和 7 天内的最新消息数并存在数据库中；

discuss_top. html——论坛主页的导航栏，即图 1 中的顶框架部分；

discuss_main. asp——论坛主页的主框架，即图 1 中的右下角表单部分，用于发送消息；

discuss_left. asp——论坛主页的左框架，即图 1 中的左下角目录列表部分，用于显示论坛的讨论主题目录，以及每个主题下的消息；

disc_post. asp——发送消息处理页面。当 discuss_main. asp 中填写的表单信息被发送时，调用本 asp 处理表单信息，并返回发送是否成功的信息；

contentdisplay. asp——显示消息内容页面。当选择点击左框架中消息列表的一项时，在主框架中显示该消息的相关信息 内容见前文 2 中主框架部分 ；

disc_reply. asp——答复消息表单页面。当主框架中显示某消息内容时，若点击 ‘答复 ”按钮，则调用本 asp 显示一个表单，用于答复所浏览的消息；

disc_response. asp——答复消息处理页面。当 disc_reply. asp 中填写的答复表单信息备发送时，调用本 asp 处理表单信息，并返回发送是否成功的信息；

disc_showre. inc —— 消息发送成功包含文件。当 disc_post. asp 或 disc_response. asp 处理表单信息成功后，调用本包含文件显示消息发送成功信息；

message_show. inc —— 消息列表显示包含文件，在 disc_left. asp 中处理消息列表的显示；

opendb33. inc——包含文件，以光标类型 3 和锁定类型 3 打开主题消息表；

closedb33. inc——包含文件，执行事务处理并关闭和清除主题消息表对象；

forum. mdb——本论坛的数据库，采用 Microsoft Access，其内所有的表和域说明见后。

3. 2 论坛的程序结构

下面采用图来说明论坛的文件关系和程序结构：

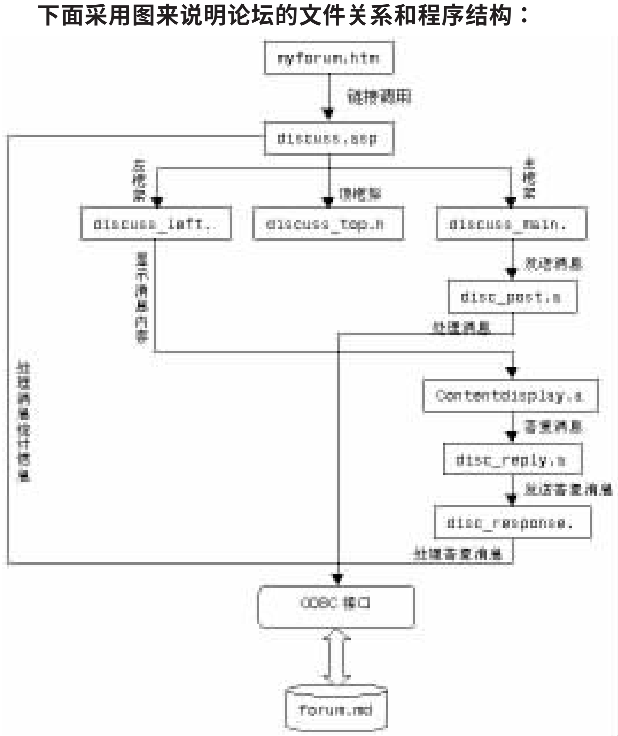


图 2　论坛程序结构图

其中黑箭头表示调用或访问关系。

3. 3 论坛的数据库设置

本论坛的数据库为 forum. mdb。此数据库内有数个表，分两类：

论坛主题表——forum_content，其结构设置如下：

表 1　论坛主题表的结构

字段名称	数据类型	字段大小	必填字段	允许空	索　引	注　释
index	自动编号	长整形			有 无重复	记录索引
contentname	文本	100	否	否	无	讨论主题
total	数字	长整形	否		无	该主题消息总数
newest	数字	长整形	否		无	最近 7 天内的新消息数

●主题消息表——在论坛主题表 forum_content 中字段 contentname 中有几个记录，就应该建立几个主题消息表，表的名称和 contentname 的值相同。

例如，本论坛中设定了六个讨论主题，即在讨论主题表 forum_content 中添加六个记录，其字段 contentname 值依次为：足球、篮球、排球、网球、羽毛球、乒乓球，则建立六个主题消息表，其表名依次为：足球、篮球、排球、网球、羽毛球、乒乓球，并且每个主题消息表的结构都是相同的。

主题消息表的结构设置如下：

表 2　论坛主题消息表的结构

字段名称	数据类型	字段大小	必填字段	允许空	索　引	注　释
index	自动编号	长整形			有 无重复	记录索引
subject	文本	100	否	否	无	消息标题
submitdate	日期/时间	长整形	否		无	消息提交时间
level	数字	长整形	否		无	该消息所处的层次，详见 4
content	备注	65 535	否	是		消息正文
next	数字	长整形	否		无	链接指针，指向该消息的下一个消息，详见 4
name	文本	50	否	否	无	提交该消息者的姓名
email	文本	50	否	是	无	提交该消息者的电子邮箱

四、论坛的关键算法说明

编写本论坛有两个需要注意的，一个是论坛主题的确定和显示，一个是消息按主题和层次显示和处理。

4. 1 论坛主题的确定和显示

可能读者已经注意到，在 3. 3 中论坛主题表 forum_content 中有一个字段域为 contentname，用来存放论坛主题。在本论坛中，采用读取该字段值输出到客户端浏览器页面的方法来显示论坛主题，这样只需要在论坛主题表中修改、添加或删减记录即可实现论坛主题的变更，而不需修改应用程序。唯一需要注意的一点是，每当发生论坛主题的变更时，应该相应地对 3. 3 中的主题消息表的表名进行变更，保证论坛主题的每个记录在数据库中都有一个以该记录中字段 contentname 的值命名的主题消息表存在。

这段处理程序在文件 discuss_left. asp 中，如下：

```
<%                    ´display discuss content 显示讨论主题
```

```
  on error resume next      ´程序出错则继续运行
  dim recordtotal
´定义变量 recordtotal 存储论坛主题表中记录总数
  dim contotal()
´定义数组 contotal 存储论坛主题表中的 contentname 的值
  set rs = server.createobject(″ADODB.recordset″)
  rs.open ″select * from forum_content″, ″DSN = forum; UID = user; PWD = ″, 3, 1
     ´以只读方式打开数据库 forum.mdb 中表 forum_content
  if not rs.eof then        ´记录不为空则继续
%>   <ul>      ´采用无序项目列表方式显示所有的讨论主题
<%
   recordtotal = rs.recordcount
                            ´获得记录总数
   redim contotal(recordtotal - 1)
                  ´重定义数组 contotal, 确定实际内存大小
   for i = 0 to recordtotal - 1
%>
     <li><a href = ″ # content <% = i%>″><% = rs( ″ contentname″)%></a>
     (共 <% = rs(″total″)%>条, <font color = ″#ff0000″>新 <% = rs(″newest″)%>条 </font>) </li>
´显示论坛主题及该主题下的消息总数和新消息数(这里为 7 天内)其统计处理程序段在 discuss.asp 中
<%   contotal( i ) = rs(″contentname″)
        rs.movenext
        next%>         ´for 循环, 直到记录尾
        </ul>
<%
   else                    ´没有讨论主题记录
     response.write ″对不起, 还没有讨论内容 . ″
     rs.close
     set rs = nothing
     response.end          ´显示相应信息, 关闭并清除 rs, 结束 asp 页面请求并显示页面结果
  end if
  rs.close
  set rs = nothing   ´有讨论主题记录操作结束时关闭并清除 rs
%>
```

程序段 1

4.2 消息按主题和层次显示和处理

本论坛中的消息按主题显示，如图 1，左框架中消息列在所属主题后。由于每个消息的内容都存在数据库中相应的主题消息表中，所以消息按主题显示易于实现。

在一个特定主题消息表中，消息的显示由两个字段域 level 和 next 决定。level 确定消息的层次，即消息处于第几层答复，next 决定消息的显示顺序。这两个字段域值的确定是本论坛消息处理和显示的关键所在，具体算法和程序实现说明如下：

●消息的显示

每个消息按其层次右缩 level * 2 个空格。非答复的消息贴子的 level 值为 1，每多一个答复层次，其 level 值在所答复的消息的 level 值的基础上加 1。

特定主题消息表中的消息的显示顺序由 next 域决定。在消息表不为空时，总是从第一个记录开始显示，然后按该记录的 next 值相对于当前记录移动光标到新的记录处，该新的记录即成为当前记录。next>0，光标前移；next < 0，光标后移。重复这个过程直到记录结束或当前记录的 next 值为 0。这样，有 n 条消息记录，则光标只需移动最多 n - 1 次，便可完全显示所有消息。

其程序在 discuss_left.asp 中实现，其引用的变量在前面程序段 1 中定义赋值。如下：

```
<%
   for i = 0 to recordtotal - 1  ´for 循环开始, 循环次数为论坛主题数
%>
<a name = ″ # content <% = i%>″></a><font color = ″ #5555aa″ size = ″3″><% = (i + 1)%>
<% = contotal( i )%></font> & nbsp; & nbsp; <a href = ″#toc″><font size = ″2″>[返回页首] </font></a>
 <br>                        ´显示论坛主题, 并加以编号
<%   set rs = server.createobject(″ADODB.recordset″)
     rs.open ″select * from ″& contotal( i ), ″DSN = forum; UID = user; PWD = ″, 3, 1
                   ´以只读打开主题消息表, 表名存于 contotal( i )
     if not rs.eof then                ´该表不为空继续
%>
     <! - - # INCLUDE VIRTUAL = ″ forum/message_show.inc″ - - >´显示首条消息记录信息
   ´消息显示处理包含文件 message_show.inc 见程序段 3
<%
 endnext = cint(trim(rs(″next″)))´获取首条记录的 next 值
      countend = 1             ´记录当前已处理消息记录数
    thiscount = rs.recordcount´获取本主题消息表的记录总数
      do until ((endnext = 0) or (countend = thiscount))
  ´do until 循环, 直到当前记录的 next = 0 或处理了所有记录
  thisnext = cint(trim(rs(″next″)))´获取当前记录的 next 值
      rs.move thisnext, 0´相对于当前记录将光标移动 next 位
        endnext = thisnext
        countend = countend + 1
%>
     <! - - # INCLUDE VIRTUAL = ″ forum/message_show.inc″ - - >
                         ´显示当前消息记录信息
<%
     loop                      ´do until 循环结束
   else                    ´该主题消息表记录为空
      response.write ″ <center><fontsize = 2> 尚无讨论提交 . </font></center><p>″
    end if
    rs.close
    set rs = nothing                   ´关闭并清除 rs
  next                     ´for 循环结束
      %>
```

程序段 2

message_show.inc 的程序如下：

```
<%      for nvspnum = 1 to cint(rs(″level″))
%>           & nbsp; & nbsp;
```

```
  <%next%>  ´按记录的 level 值右缩 level * 2 个空格
 <font size = "2">
 <a  href = " contentdisplay. asp? contentdis = <% = rs( " index") %>& discussarea = <% = contotal( i ) %>
"target = "mainFrame"><% = rs("subject") %></a>
( <% = rs("name") %><font color = "#000000">, <% = rs("
submitdate") %></font>)
´显示消息的标题等信息,并创建面向 contentdisplay. asp 的页面的超文本链接,链接内含有名为 contentdis 和 discussarea 的查询字符串变量,用于将该记录的索引值和所在的主题消息表名传给 contentdisplay. asp 处理,以显示消息内容
 <%      thediff = datevalue(now) - datevalue(rs( " submit-
date"))
       if thediff < = 7 then
%>
           <font color = "#ff0000">new </font>
 <%
       end if   ´如果该消息在 7 天内提交,其后添加 new 标记
%><br></font>
```

程序段 3

●消息的处理

当在论坛中发送消息或答复消息时，应用程序在数据库中相应的主题消息表中添加新记录，最重要的就是确定记录中 level 和 next 的值，并修改相关的记录的 next 值。

本论坛应用程序有两个文件涉及到消息的处理，即发送消息的 disc_post. asp 表单信息来自于 discuss_main. asp ，答复消息的 disc_response. asp 表单信息来自于 disc_reply. asp 。

在 disc_post. asp 中处理的消息，其 level 值肯定为 1；并且将新消息记录添加到相应主题消息表的末尾，其 next 值肯定为 0。其处理程序段如下：

```
 <%
    subject = trim(request. form("subject"))
    discussarea = trim(request. form("selectno"))
    comments = trim(request. form("comments"))
    name = trim(request. form("name"))
    email = trim(request. form("email"))
              ´获得来自 discuss_main. asp 的表单信息
    ´to create record
    on error resume next
%>
  <! - - #INCLUDE VIRTUAL = "forum/opendb33. inc" - - >
      ´以光标 3 和锁定 3 的方式打开主题消息表 discussarea
                        ´其内容见程序段 5
 <%
    truefind = true´truefind 用于判断是否找到 next = 0 的记录
    if not rs. eof then          ´主题消息表不为空继续
      rs. movefirst              ´光标移动到首条记录
      firstindex = cint(trim(rs("index")))
                  ´记录首条记录的索引值到变量 firstindex 中
      rs. movelast               ´光标移动到最后一条记录
      tonext = 1            ´变量 tonext 用于存储表中 next = 0 的记录到表尾的距离
      findit = false             ´findit 判断是否找到 next = 0 的记录或搜索完所有记录
      do until findit            ´do until 循环开始
         if cint(trim(rs("index"))) = firstindex then
           findit = true
 end if  ´如果已经搜索到首条记录,findit 赋值为真,结束循环
         if cint(rs("next")) < >0 then
                      ´next 不为 0,则将光标继续前移
           truefind = false
           rs. moveprevious
           tonext = tonext + 1
         else            ´next = 0,则找到记录
           truefind = true
           findit = true
         end if
      loop               ´do until 循环结束
      if truefind then   ´找到 next = 0 的记录,并将该条记录的 next 值赋为 tonext,这样记录的下一条显示记录即为新添加记录
         rs("next") = tonext
      end if
    else                 ´该主题消息表为空,则 tonext 应为 0
      tonext = 0
    end if
    if truefind then     ´找到 next = 0 的记录,则添加新纪录
      rs. addnew
      rs("subject") = subject
      rs("submitdate") = now
      rs("content") = comments
      rs("name") = name
      rs("next") = 0
      rs("level") = 1
      rs("email") = email
      rs. update
    else  ´没有找到 next = 0 的记录,其处理在文件 closedb33. inc 中
%>
 <! - - #INCLUDE VIRTUAL = "forum/closedb33. inc" - - >
                        ´其内容见程序段 6
```

程序段 4

opendb33. inc 的程序如下：

```
  <%
     set cnn1 = server. createobject("adodb. connection")
     cnn1. open "forum", "user", ""
     set rs = server. createobject("ADODB. recordset")
     rs. cursortype = 3
     rs. locktype = 3
     cnn1. begintrans          ´开始事务处理
rs. open discussarea, cnn1, , , 2  ´打开主题消息表 discussarea
  %>
```

程序段 5

closedb33. inc 的程序如下：

```
  <%
  cnn1. rollbacktrans´没有找到 next = 0 的记录,撤销事务处理
        response. write " <center>对不起,本讨论区出现问题,暂时停用! </center>"
        rs. close
        set rs = nothing
```

```
    cnn1.close
    set cnn1 = nothing         ´关闭并清除 rs 和 cnn1
    response.end               ´页面请求结束,显示页面
  end if                       ´if truefind then 判断语句结束
  if err = 0 then              ´程序运行无错,触发事务处理
    cnn1.committrans
  else
    cnn1.rollbacktrans  ´程序运行出错,撤销事务处理
    response.write " <center>讨论提交失败! </center>"
    rs.close
    set rs = nothing
    cnn1.close
    set cnn1 = nothing         ´关闭并清除 rs 和 cnn1
    response.end
  end if
  rs.close
  set rs = nothing
  cnn1.close
  set cnn1 = nothing                  ´关闭并清除 rs 和 cnn1
      %>
```

程序段 6

在 disc_response.asp 中处理的消息，其 level 值肯定为所答复消息记录的 level 值加 1；并且将新消息记录添加到相应主题消息表的末尾，应该将其所答复消息记录的 next 指向新记录，同时新纪录的 next 指向所答复消息记录指向的记录。其处理程序段如下：

```
<%
  subject = trim(request.form("subject"))
  discussarea = request.form("discussarea")
  comments = trim(request.form("comments"))
  name = trim(request.form("name"))
  email = trim(request.form("email"))
  levelreply = request.form("levelreply") + 1
  indexreply = cint(trim(request.form("indexreply")))
        ´获得来自 disc_reply.asp 的表单信息
  ´to create record
  on error resume next
%>
 <!--#INCLUDE VIRTUAL = "forum/opendb33.inc"-->
                ´内容见程序段 5
<%
  truefind = true
  rs.movefirst
  firstindex = cint(trim(rs("index")))
  rs.movelast
  tonext = 1
  findit = false
  do until findit              ´do until 循环开始
    if cint(trim(rs("index"))) = firstindex then
      findit = true
    end if
    if cint(rs("index")) <> indexreply then
                ´未找到所答复消息记录,光标继续前移
      truefind = false
      rs.moveprevious
      tonext = tonext + 1
    else´找到所答复消息记录,则 truefind 和 findit 赋值为真
      truefind = true
      findit = true
    end if
  loop                         ´do until 循环结束
  if truefind then´找到所答复消息记录,则存储该记录 next 到变量 thisnext 中,并修改该记录的 next 为到新添加记录的距离
    thisnext = cint(trim(rs("next")))
    rs("next") = tonext
    rs.addnew                  ´添加新记录
    rs("subject") = subject
    rs("submitdate") = now
    rs("content") = comments
    rs("name") = name
    rs("email") = email
    if thisnext <> 0 then      ´如果 thisnext 不为 0,则新记录 next 值为 thisnext - tonext
      rs("next") = thisnext - tonext
    else                       ´如果 thisnext 为 0,则新记录 next = 0
      rs("next") = 0
    end if
    rs("level") = levelreply
    rs.update
  else´没有找到所答复消息记录,其处理见 closedb33.inc
%>
 <!--#INCLUDE VIRTUAL = "forum/closedb33.inc"-->
                ´内容见程序段 6
```

程序段 7

五、程序清单

这里列出本论坛的程序清单。这只是一个非常简单的 WEB 程序，可以添加很多功能进行扩展，如添加图片，将主题显示与消息显示分成不同页面。有兴趣不妨一试。

5.1 myforum.html：

```
<HTML>
<body>
<p><p>
<center><h3>Welcome to My <a href = "discuss.asp" target = "_self">Forum </a>! </h3></center>
                ´建立到论坛的页面链接
</body>
</HTML>
```

5.2 discuss.asp：

```
<%@ language = VBScript%>
<html>
<head>
<title>讨论组 </title>
<meta http-equiv = "Content-Type" content = "text/html; charset = gb2312">
</head>
<%
 ´on error resume next
 discussarea = "forum_content"
%>
```

```
<!--#INCLUDE VIRTUAL="forum/opendb33.inc"-->
    '打开表 forum_content
<%
   if rs.recordcount=0 then   '无记录,则结束页面请求
   cnn1.rollbacktrans
   response.write "<center>对不起,讨论区没有设定内容!</center>"
   rs.close
   set rs=nothing
   cnn1.close
   set cnn1=nothing
   response.end
  end if
  for todoit=1 to rs.recordcount   '有记录,则对每个主题消息表进行统计
   set rs1=server.createobject("ADODB.recordset")
   rs1.open "select * from "&rs("contentname"),"DSN=forum;UID=user;PWD=",3,1
   newnum=0
   for todoit1=1 to rs1.recordcount
  thediff=datevalue(now)-datevalue(rs1("submitdate"))
    if thediff<=7 then
     newnum=newnum+1'7天之内发送的消息为新消息
    end if
    rs1.movenext
   next
   rs("total")=rs1.recordcount
   rs("newest")=newnum
   rs1.close
   set rs1=nothing
   rs.movenext
  next
  if err=0 then
   cnn1.committrans
  else
   cnn1.rollbacktrans
   response.write "<center>对不起,讨论区失败!</center>"
   rs.close
   set rs=nothing
   cnn1.close
   set cnn1=nothing
   response.end
  end if
  rs.close
  set rs=nothing
  cnn1.close
  set cnn1=nothing
%>
<frameset frameborder="no" border="0" framespacing="0" bordercolor="#ffffff"
rows="79,421*" cols="*">
  <frame name="topFrame" scrolling="NO" noresize src="discuss_top.html">
  <frameset frameborder="YES" border="2" framespacing="2" bordercolor="#FFCCFF"
     cols="414,379*" rows="*">
     <frame name="leftFrame" noresize src="discuss_left.asp">
    <frame name="mainFrame" src="discuss_main.asp">
   </frameset>
   <noframes size="+1">
   <body bgcolor="#FFFFFF">
   </body>
   </noframes></frameset>
 <frameset>
  <noframes>
  </noframes>
 </frameset>          '以上为框架定义语句
 </html>
```

5.3 discuss_top.html:

```
<html>
<head>
<title>讨论组</title>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312">
</head>
<body bgcolor="#FFFFFF">
<p align="center"><font face="黑体,System,楷体_GB2312">
<marquee border="0" align="middle" behavior="alternate" style="color: rgb(0,0,255)">
<kbd><font size="4">小小论坛欢迎您</font></kbd>
</marquee></font></p>
<table width="75%" border="0" align="center" height="25">
  <tr align="center" valign="middle">
   <td height="27" nowrap>
    <div align="center"><a href="myforum.html" target="_parent">[返回主页]</a></div>
   </td>
   <td height="27" nowrap>
    <div align="center"><a href="discuss.asp" target="_parent">[刷新内容]</a></div>
   </td>
   <td height="27" nowrap>
    <div align="center"><a href="discuss_main.asp" target="mainFrame">[发 送]</a></div>
   </td>
  </tr>
</table>
<p> </p>
</body>
</html>
```

5.4 discuss_main.asp:

```
<HTML>
<HEAD><TITLE>讨论组发送区</TITLE>
<script language=javascript>          '表单输入检查脚本
 function checksubmit(form)
 {
  if (form.subject.value=="")
  {
   alert("请输入您的标题");
   form.subject.focus();
```

```
    return false;
   }
  if (form.name.value = ="")
  {
    alert("请输入您的姓名");
    form.name.focus();
    return false;
   }
   if (form.comments.value = ="")
   {
      alert ("请输入讨论内容");
      form.comments.focus();
      return false;
   }
   return true;
}
</script>
</HEAD>
<BODY>
<FORM METHOD=POST ACTION=" disc_post.asp"
name=form1
onsubmit="return checksubmit(form1)">'发送表单信息到
disc_post.asp 中处理
<FONT color=#8000ff><STRONG>请逐项填写下表,用
TAB 键跳到下一项.</STRONG></FONT><p>
<STRONG>您的标题:</STRONG><br><input type=text
name="subject" tabindex=1 maxlength=100 size=20><
br>
<STRONG>您的姓名:</STRONG><br><input type=text
name="name" tabindex=1 maxlength=50 size=20><br>
<STRONG>您的信箱:</STRONG><br><input type=text
name="email" tabindex=1 maxlength=50 size=20><br>
<STRONG>选择:</STRONG><br>
<select name="selectno" tabindex=2>        '论坛主题作
为下拉列表框的项
<%
  set rs=server.createobject("ADODB.recordset")
  rs.open "select * from forum_content","DSN=forum;
UID=user;PWD=",3,1
  if not rs.eof then%>
     <option value="<%=rs("contentname")%>" select-
ed><%=rs("contentname")%>
<%  rs.movenext
    for i=2 to rs.recordcount%>
     <option value="<%=rs("contentname")%>"><%=
rs("contentname")%>
<%  rs.movenext
    next
  else%>
     <option value="nocontent">没有内容
<% end if
  rs.close
  set rs=nothing
%>
</select>
<br>
<STRONG>讨论内容:</STRONG><br><textarea rows=
5 cols=45 name="comments" tabindex=3></textarea>
<p>
<center><input type=submit value="发送内容"> 
 <input type=reset value="重填内容"></center>
</form>
</BODY>
</HTML>
```

5.5 discuss_left.asp:

```
<html>
<head>
<title>讨论区目录页</title>
<meta http-equiv=" Content-Type" content=" text/
html; charset=gb2312">
</head>
<body bgcolor="#FFFFFF">
<center><a name="#toc"></a>
<font color="#333333" size=2"> 讨论区内容目录</
font></center><br>
见程序段 1
  'display toc
见程序段 2
<%  if err<>0 then
     response.write "<center>对不起,讨论区失败!</cen-
ter>"
   end if
%>
</body>
</html>
```

5.6 disc_post.asp:

```
<HTML>
<HEAD><TITLE>讨论组发送区</TITLE>
</HEAD>
<BODY>
见程序段 4
<!--# INCLUDE VIRTUAL=" forum/disc_showre.inc"
-->  '消息处理结果显示包含文件
</BODY>
</HTML>
```

5.7 contentdisplay.asp:

```
<HTML>
<HEAD><TITLE>讨论组内容显示区</TITLE>
</HEAD>
<BODY>
<%
   'display discuss content
   contentdis=trim(request.querystring("contentdis"))
   discussarea=trim(request.querystring("discussarea"))
           '获取从左框架消息列表中发送过来的查询字符串
  if contentdis="" or discussarea="" then  '查询字符串为
空表示出错
 response.write "对不起,本讨论内容信息含错,不能浏览."
     response.end
  else                     '查询字符串不为空则进行处理
   on error resume next
   set rs=server.createobject("ADODB.recordset")
   rs.open "select * from "&discussarea,"DSN=forum;
```

```
UID=user;PWD=",3,1
    findid=false
    if not rs.eof then                ′主题消息表不为空继续
      countrs=1
      do until findid or (countrs>rs.recordcount) ′查找要显示的消息记录
        if cstr(rs("index"))=contentdis then
          findid=true
        else
          countrs=countrs+1
          rs.movenext
        end if
      loop
     end if
    if findid then                    ′找到该记录,则进行显示
      mydate=formatdatetime(trim(rs("submitdate")),1)
    mytime=formatdatetime(trim(rs("submitdate")),3)
%>
    <h2><strong>讨论区</strong></h2><hr><center>
    <form method=post action="disc_reply.asp"> ′发送隐藏表单信息给 disc_reply.asp
    <input type=submit name=submit value="答复">
    <input type=hidden name="subject1" value=<%=rs("subject")%>>
    <input type=hidden name="discussarea" value=<%=discussarea%>>
    <input type=hidden name="theindex" value=<%=rs("index")%>>
    </form>
    </center><p><h4><strong><%=rs("subject")%></strong></h4>
    <table>                           ′显示找到的消息信息
    <tr align=right><td><font color="#000000">探讨主题:</font></td>
    <td><font color="#ee3333"><%=discussarea%></font></td></tr>
    <tr align=right><td><font color="#000000">提交作者:</font></td>
    <td><font color="#ee3333"><%=rs("name")%></font></td></tr>
    <tr align=right><td><font color="#000000">提交人邮箱:</font></td>
    <td><font color="#0000ff"><a href="mailto:<%=rs("email")%>">
    <%=rs("email")%></font></td></tr>
    <tr align=right><td><font color="#000000">提交日期:</font></td>
    <td><font color="#ee3333"><%=mydate%></font></td></tr>
    <tr align=right><td><font color="#000000">提交时间:</font></td>
    <td><font color="#ee3333"><%=mytime%></font></td></tr>
    </table>
    <font color="#000000">提交内容:</font><br>
  <font color="#ee3333"><%=rs("content")%></font>
<%
  else                                ′没有找到该消息记录
    response.write "对不起,该提交内容已被删除."
    rs.close
    set rs=nothing
    response.end
  end if
 end if
%>
</BODY>
</HTML>
```

5.8 disc_reply.asp：

```
<HTML>
<HEAD><TITLE>答复</TITLE>
<script language=javascript>          ′表单输入检查脚本
 function checksubmit(form)
 {
   if (form.subject.value=="")
   {
     alert("请输入答复标题");
     form.subject.focus();
     return false;
    }
   if (form.name.value=="")
   {
     alert("请输入您的姓名");
     form.name.focus();
     return false;
  }
   if (form.comments.value=="")
  {
     alert("请输入答复内容");
     form.comments.focus();
     return false;
  }
  return true;
 }
</script>
</HEAD>
<BODY>
<%
    dim findid
    subject1=request.form("subject1")          ′获取从contentdisplay.asp传来的表单信息
    discussarea=request.form("discussarea")
    theindex=request.form("theindex")
    set rs=server.createobject("ADODB.recordset")
    rs.open "select * from "&discussarea,"DSN=forum;UID=user;PWD=",3,1
    if not rs.eof then                        ′该主题消息表不为空继续
      findid=false
      do until ((findid=true) or rs.eof)      ′按索引查找要答复的消息记录
        comp=strcomp(trim(rs("index")),trim(theindex),1)
        if comp=0 then
          findid=true
        else
          if not rs.eof then
            rs.movenext
```

```
      end if
    end if
  loop
  if findid = true then             ´找到记录,则存储该记录的 level 和 index 信息
    getlevel = rs(˝level˝)
    getindex = rs(˝index˝)
  else                              ´没有找到记录,则结束页面请求
    response. write ˝没有相关信息!˝
    rs. close
    set rs = nothing
    response. end
  end if
else                                ´该主题消息表为空,结束页面请求
  response. write ˝没有相关信息 . ˝
  rs. close
  set rs = nothing
  response. end
end if
rs. close
set rs = nothing
%>
<FORM METHOD = POST ACTION = ˝ disc_response. asp˝ name = form1 onsubmit = ˝
return checksubmit(form1)˝>                ´发送表单信息给 disc_response. asp
<FONT color = #8000ff>
<STRONG>请逐项填写下表,用 TAB 键跳到下一项. </STRONG></FONT><p>
<STRONG>答复标题: </STRONG><br>
<input type = text name = ˝subject˝ value = ˝RE: ˝& subject1 tabindex = 1 maxlength = 100 size = 20><br>
<STRONG>您的姓名: </STRONG><br>
<input type = text name = ˝name˝ tabindex = 1 maxlength = 50 size = 20><br>
<STRONG>您的信箱: </STRONG><br>
<input type = text name = ˝email˝ tabindex = 1 maxlength = 50 size = 20><br>
<input type = hidden name = ˝ levelreply˝ value = <% = getlevel%>>
<input type = hidden name = ˝ indexreply˝ value = <% = getindex%>>
<input type = hidden name = ˝ discussarea˝ value = <% = discussarea%>>
<br>
<STRONG>讨论内容: </STRONG><br>
<textarea rows = 5 cols = 45 name = ˝ comments˝ tabindex = 3></textarea><p>
<center><input type = submit value = ˝发送内容˝>& nbsp; & nbsp;
<input type = reset value = ˝重填内容˝></center>
</form>
</BODY>
</HTML>
```

5. 9 disc_response. asp:

```
<HTML>
<HEAD><TITLE>讨论组答复发送区</TITLE>
</HEAD>
<BODY>
见程序段 7
<! - - # INCLUDE VIRTUAL = ˝ forum/disc_showre. inc˝ - - >       ´消息处理结果显示包含文件
</BODY>
</HTML>
```

5. 10 disc_showre. inc:

```
<%call showresult
sub showresult()
%>
<html><head><title>内容提交成功</title></head>
<body>
<center><strong>您的讨论内容已经提交成功</strong></center><hr>
<p><font color = ˝#1111ff˝ size = ˝2˝>注意: 您必须刷新内容才能看到提交的内容! </font><p>
<p><font color = ˝#00000˝> 请您点击 <a href = ˝ discuss. asp˝ target = ˝_parent˝>[返回讨论组]</a>, 可以看到您的内容已经加入 . </font>
</body>
</html>
    <%end sub%>
```

5. 11 message_show. inc:

见程序段 3

5. 12 opendb33. inc:

见程序段 5

5. 13 closedb33. inc:

见程序段 6

收稿日期 2000 年 9 月 11 日

跨平台电子商务沟通新脉动——XML 技术及 Tamino 产品研讨会

华彩软件与德商赛克科技(Software AG)共辟业界新领域

11 月 30 日华彩软件与德商赛克科技(Software AG)于京正式发布了全球第一套以纯 XML 为基础的数据库管理系统—Tamino。此项产品是世界首先依据纯粹且标准的 XML 格式进行资料储存与读取的信息服务器。

Tamino 的优势是 XML 技术为其核心技术,XML 能够支持各种界面、资料之间的转换,整合企业资源,降低营运与管理成本,提供企业一个前所未有的机会,使您的企业在一夕之间进入电子商务和全球信息网之中。

Tamino 产品由德商赛克科技股份有限公司研发,其做为欧洲最大的系统软件制造商,同时也是全球提供数据处理服务与电子商务的科技大厂。华彩软件是华彩网络集团于 1992 年成立的子公司,经过八年的辛勤耕耘,在产品行销与渠道两方面取得卓越绩效。

随着 XML 技术的崛起,全球第一套纯 XML 技术为基础的 Tamino 产品凭借首屈一指的技术特色吸引了人们的注意力,带给业界强大的冲击力,率先引领了数据管理不可逆转的发展趋势。

Windows9x 网络底层应用程序的设计方法

陈取才　张蕴玉　胡修林

摘　要　本文介绍了 Windows9x 下的一个虚拟驱动程序 VPacket 的应用接口,通过该接口,Win32 应用程序可以直接访问安装在计算机上的网络接口控制器 (NIC) 从而可以对网络应用进行监测或者实现用户自己的网络协议栈。

关键词　VPacket,VxD,Win32API,NDIS

一、概述

出于系统资源共享和安全性的考虑,Win32API 不提供直接访问网络低层协议的支持。应用程序要想进行底层操作,就必须编制相应的客户虚拟驱动程序 (Virtual Device Driver),由虚拟驱动程序 (VxD) 来充当底层的网络接口控制器 (NIC) 和上层 Win32 应用程序之间的接口,形成如图 1 所示的结构。图中工作于 Ring3 层的 Win32 应用程序虽然不能直接访问底层的硬件资源,但它可以通过调用 VxD 来驱动和控制 NDIS,达到同 NIC 通讯的目的。P32 编程环境自带的 Vpacket. vxd 就为 Win32App 与 NIC 通信的提供了灵活的控制接口,适于用户进行底层网络应用开发。下面就具体讨论 Vpacket 的开发方法。

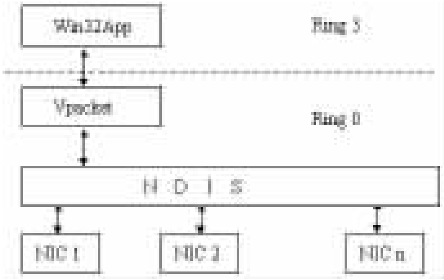


图 1

二、接口抽象层

为了使操作系统具有良好的兼容性,Win32 应用程序调用的虚拟驱动程序 (VxD) 也并不直接同网络接口控制器打交道,而是在 VxD 与网络硬件之间定义了一个接口抽象层 NDIS3. 10 它的主要作用是把软件从网络适配器的具体细节中解放出来,驱动程序能同计算机上的任意 NIC 相通信,当然前提是 NIC 是与 NDIS 兼容的,这样就简化了 VxD 的设计,缩短开发周期。

三、加载驱动程序

应用程序调用 VxD 时,是通过虚拟机管理器 (VMM) 查询 VxD 的设备描述符块 DDB Device Descriptor Block 来获得 VxD 的主入口点,VMM 利用这个主入口点将 VM 及 Windows 自身的状态通知给 VxD,然后 VxD 通过相应的工作来响应这些事件。Win32API 提供了动态加载 / 卸载 VxD 的接口函数 CreateFile .... /CloseHandle 典型的动态加载 / 卸载代码如下:

```
//动态加载
HANDLE hVxD;
hVxD = CreateFile(″\\\\.\\VPACKET.VXD,  //*.vxd 必须处于 System 或当前目录下
        GENERIC_READ|GENERIC_WRITE,
        0,
        NULL,
        OPEN_EXISTING,
FILE_ATTRIBUTE_NORMAL|FILE_FLAG_OVERLAPPED |FILE_FLAG_DELETE_ON_CLOSE, //可异步操作、关闭时 VxD 须卸载
        NULL);
if (hVxD == INVALID_HANDLE_VALUE)return SYSERR; //加载失败
else ......; //加载成功
//动态卸载
if(hVxD != INVALID_HANDLE_VALUE)CloseHandle(hVxD);
```

四、驱动程序同网络接口的绑定

如前所述,VPacket 能支持多个 NIC VPacket 加载后,要对某个具体的 NIC 进行控制,还必须把 VxD 同相应的 NIC 相绑定,方法如下:

```
int Bind(HANDLE hVxD, BYTE * inBuffer);
```

其中的第一个参数是已加载的驱动程序句柄,第二个参数是描述 NIC 的字符串,实际上是在注册表:

HKEY_LOCAL_MACHINE\ System\ CurrentControlSet\ Services\Class\Net

下该 NIC 对应的键值,如:拨号网卡对应“0000”,第一个以太网卡对应“0001”等等。

五、设备驱动程序的应用接口 (VxD API)

设备驱动程序的应用接口包括 VxD 为 Win32App 提供的服务接口和 Win32App 为 VxD 提供的服务接口两个方面,如图2所示。

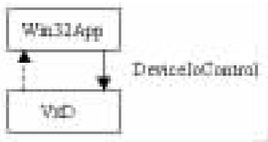


图 2

Win32 应用向 Vpacket 发起请求服务的唯一接口是：

```
DeviceIoControl(hVxD, dwIoControlCode, //Vxd 句柄、命令码
lpvInBuf, cbIn,          // 传递给 VxD 的参数区
lpvOutBuf, cbOut,        // 返回的参数区
nByteRet,                //实际返回字节数
lpOverlapped);
```

其中 dwIoControlCode 指定 Win32 应用程序希望的操作，这些操作在编写驱动程序时已经集成到 VxD 中，Vpacket 提供的命令码如下：

```
IOCTL_PROTOCOL_BIND          将 VPACKET 同指定适配器绑定
IOCTL_PROTOCOL_QUERY_OID     获取指定对象的 ID 号
IOCTL_PROTOCOL_SET_OID       设置指定对象的 ID 号
IOCTL_PROTOCOL_STATISTICS    获取指定适配器的统计结果
IOCTL_PROTOCOL_RESET         适配器复位
IOCTL_PROTOCOL_READ          接收数据包
IOCTL_PROTOCOL_WRITE         发送数据包
IOCTL_PROTOCOL_MACNAME       查询适配器的名字
......
```

利用这些命令，用户可以对网络进行监控、定义自己的传输协议栈、对接收的数据包进行分析，对一些非法用户的数据包予以抛弃，从而实现防火墙等等。

在实际运行 Windows 系统中，处于 Ring3 层的应用程序与 Ring0 层的 VxD 是同时运行的，Ring3 层的应用程序接收用户指令的同时，VxD 也时刻准备着接收随机到达的数据包，因此必须提供一种机制，使得当 VxD 接收到数据包时，VxD 能及时通知 Ring3 层相应的应用程序，然而，Window9x 并没有为 VxD 提供可以直接调用的接口函数，而是引入了一种基于事件的异步通讯机制。在 VPacket 为 Win32App 提供的接口函数 DeviceIoControl 中，Win32App 向 VPacket 传递了一个类型为 OVERLAPPED 结构的参数 lpOverlapped，该参数的成员变量 hEvent 指向事先在 Win32App 中创建的事件实例，VxD 接收到数据包或完成 Win32App 请求的服务后，激发该事件，从而通知 Ring3 层执行中或处于睡眠的应用程序完成后续操作。

六、应用开发实例

Win32 应用程序利用设备驱动程序的应用接口 IOCTL_PROTOCOL_READ 获得的数据包是 MAC 数据包，在此基础上用户可以定义自己的处理模块，象 TCP/IP 一样对此 MAC 数据包进行分析处理，生成符合应用要求的数据格式和控制结构， 发送过程类似）实现用户协议栈等底层网络应用。

以下代码是监听网络数据的实例，描述了底层网络应用程序的基本流程：

```
//加载 VPacket
hVxD = CreateFile(″\\\\.\\VPACKET.VXD″, ......);
if (hVxD == INVALID_HANDLE_VALUE){......;}// 错误处理与 NDIS3 适配器绑定
Bind(hVxD, args[1]);
//处理循环
 while (1)
 {//监听网络
   count = GetPacket( hVxD, (ULONG)IOCTL_PROTOCOL_READ, ibuf, ilen, obuf, olen);
      if (count == 0)
{  ......  ;}//接收错误 else
{......;}//显示及处理
}
 //卸载 VPacket
   CloseHandle( hVxD );
```

实验证明，利用 Vpacket 驱动，Ring3 层的应用程序可以直接截获 MAC 层的数据包，因而可以撇开 Windows 内嵌的 TCP/IP 处理过程，开发用户自己的网络应用程序。

七、结论

VPacket 为 Windows9x 下的应用程序直接访问网络接口控制器提供了一种简单的途径，在 P32 编程环境下，VPacket 可以方便地实现整个 TCP/IP 协议栈、监测网络流量、实施网络路由等一系列应用。

参考资料

1. 95DDK& 98DDK

2. 美 Karen Hazzh . Writing Windows VxDs and Device Drivers. Lawrence KS R& D Publications. 1995 - 1997

3. Chlap Christopher VPACKET VxD Source Code

收稿日期：2000 年 9 月 13 日）

利用 NetBIOS 进行 Windons 网络编程

刘安安

摘 要 本文介绍了 NetBIOS 编程的一些基本概念,并通过一个异步事件服务器和一个异步事件客户机的例子,详细说明了 NetBIOS 进行 Windows 编程的基本方法。

关键词 NetBIOS 异步事件,服务器,客户机

"网络基本输入/输出系统"(Network Basic Input/Output System,NetBIOS)是 1983 年由 Sytex 公司为 IBM 公司开发的一种标准应用程序编程接口,并被微软采用。1985 年,IBM 改进了 NetBIOS,推出了 NetBIOS 扩展用户接口(NetBIOS Extended User Interface NetBEUI)通信协议,它占用内存少,配置简单,适用于小型局域网不同计算机之间的通信,但不具有跨网段工作的能力,不支持路由机制。NetBIOS 是一种与"协议无关"的编程接口,它使应用程序不用理解网络细节,应用程序可通过 TCP/IP、NetBEUI、SPX/IPX 运行。下面我们介绍 NetBIOS 编程用到的一些重要概念及其实现方法。

一、理解 NetBIOS

1. LANA 编号

理解 LAN 适配器(LAN Adapter LANA)编号是 NetBIOS 进行网络编程的关键所在。网络的传输协议是通过 LANA 编号同 NetBIOS 对应起来,每个 LANA 编号对应于网卡及传输协议的唯一组合。因此,我们在编程时要注意,两台要进行通信计算机必须至少安装有同一种协议,并且这两台计算机通信所依赖的 LANA 编号对应的网络协议要相同,否则即使这两台计算机安装相同的协议也无法进行通信。LANA 编号范围在 0 到 9 之间,其中,LANA 0 代表默认的 LANA。

2. NetBIOS 名字

NetBIOS 名字可分为两种类型:唯一名字(Unique Name)和组名(Group Name)。

顾名思义,唯一名字只允许一台计算机注册该名字,一旦唯一名字注册成功,其他计算机如果再注册该名字,就会出现:"名字重复"的错误,微软网络中的机器名采用的就是 NetBIOS 唯一名字。组名则是一组计算机的总称,可以用来接收发给这一组计算机的数据。值得注意的是:组名可以和唯一名字同名,这会引起发送或接收数据的目的出现错误!NetBIOS 名字长度为 16 个字符,其中第 16 个字符用于区分不同的网络服务。关于计算机注册 NetBIOS 名字的信息可以利用 Nbtstat 命令查看。

3. NetBIOS 提供的服务

NetBIOS 提供两种服务:面向连接的服务和数据报服务(无连接)。

面向连接的服务为两台需要进行通信的计算机建立一个连接,并利用错误探测和恢复机制保证数据在通信的两端准确无误的传输,它适于传输比较长的消息。对于 NetBIOS,服务器在对想通过它建立通信的 LANA 编号上注册,而对于位于其他计算机上的客户机会搜索服务器注册的名字,并将它解析为机器名,然后发出进行通信的请求。

数据报服务是无连接的,因而它不能保证数据有序、正确的传输,但它可以节省建立连接的开销,它适合短消息的传输。在 NetBIOS 中,客户机只是将发送数据的目的地定义为服务器注册的进程名,而不进行任何连接。

二、NetBIOS 编程的实现

NetBIOS 的所有函数声明、常数都在头文件"Nb30.h"中定义,在编程时还须与 Netapi32.lib 库进行链接。NetBIOS 接口通过一个函数实现:

```
UCHAR Netbios (PNCB pNCB);
```

其中,参数 PNCB 指向一个网络控制块(Net Control Block NCB)指针 NCB 结构如下:

```
typedef struct _NCB {
        UCHAR   ncb_command;    // NetBIOS 命令
        UCHAR   ncb_retcode;    // 指定操作的返回代码
    UCHAR   ncb_lsn;        // 本地会话编号
    UCHAR   ncb_num;        // 本地名字编号
        PUCHAR  ncb_buffer;     // 数据缓冲区地址
        WORD    ncb_length;     // 缓冲区长度
    UCHAR   ncb_callname[NCBNAMSZ]; // 远程应用程序名
      UCHAR   ncb_name[NCBNAMSZ]; //本地应用程序名
        UCHAR   ncb_rto;        // 接收操作延时
        UCHAR   ncb_sto;        // 发送操作延时
void (CALLBACK *ncb_post)( struct _NCB * );
              // 异步命令完成后需调用的后例程地址
        UCHAR   ncb_lana_num;   // LANA 编号
        UCHAR   ncb_cmd_cplt;   // 指定操作的返回代码
        UCHAR   ncb_reserve[10]; // 保留字段
        HANDLE  ncb_event;      // Win32 事件句柄
} NCB, * PNCB;
```

此外,编程时应注意调用 NetBIOS 函数的同步和异步问题。NetBIOS 命令调用本身均为同步,即在完成指定命令之前,会一直调用 NetBIOS 模块。而在实际编程时,我们通常需

要进行异步调用，即希望允许多个客户机同时与服务器进行连接，这就需要让 NetBIOS 命令与异步标志逻辑或（OR）操作，但必须在 ncb_post 字段中指定一个后例程，或在 ncb_event 字段中指定一个事件句柄。

下面，我以实现一个异步事件服务器和一个异步事件客户机为例，具体说明 NetBIOS 的编程实现，其中，服务器接收由客户机发送的数据。

1. 异步事件服务器的实现

首先，我们进行初始化工作，列举可用的 LANA 编号，并重设：

```
if (LanaEnum(&lenum) != NRC_GOODRET)
    return 1;
if (ResetAll( & lenum, (UCHAR) MAX_SESSIONS,
(UCHAR)MAX_NAMES, FALSE) != NRC_GOODRET)
    return 1;
```

&lenum 是一个 LANA_ENUM 结构变量 其定义如下

```
typedef struct LANA_ENUM
{
    UCHAR    length;
    UCHAR    lana[MAX_LANA+1];
} LANA_ENUM, *PLANA_ENUM;
```

其中，length 指出本地计算机可用的 LANA 的数量，lana 表示由这些 LANA 编号组成的一个数组。

然后为每个 LANA 编号分配 NCB 结构，并添加服务器名字，执行异步侦听。如果有客户机与服务器连接成功，则服务器接收由客户机发送的数据。程序代码如下：

```
// 为异步事件分配一组句柄
EventArray = (HANDLE *)GlobalAlloc(GMEM_FIXED,
sizeof(HANDLE) * lenum.length);
// 为每个 LANA 编号分配一个 NCB 结构
GlobalClients = (PNCB) GlobalAlloc(GMEM_FIXED |
GMEM_ZEROINIT, sizeof(NCB) * lenum.length);
// 产生事件, 为 LANA 编号添加服务器名, 并执行异步侦听
for(i = 0; i < lenum.length; i++)
{
    EventArray[i] = GlobalClients[i].ncb_event = CreateEvent(NULL, TRUE, FALSE, NULL);
    AddName(lenum.lana[i], SERVER_NAME, &num);
    Listen( & GlobalClients[i], lenum.lana[i], SERVER_NAME);
}
// 此时若在 windows 下运行 Nbtstat -n 服务器名 , 会看到一个 NetBIOS 名字表.
// 产生事件, 为 LANA 编号添加服务器名, 并执行异步侦听
while (1)
{
    // 等待, 直到有一个连接建立
    RetEvent = WaitForMultipleObjects(lenum.length,
EventArray, FALSE, INFINITE);
    if (RetEvent == WAIT_FAILED)
    {
        printf("等待连接失败! \n");
        break;
    }
    // 遍历每个 NCB 结构, 是否有多个连接建立, 如果 ncb_cmb_plt 的值不是 NRC_PENDING, 即连接成功, 则产生接收数据的是线程, 并为它创建一个新的 NCB 结构.
    for(i = 0; i < lenum.length; i++)
    {
     if (GlobalClients[i].ncb_cmd_cplt != NRC_PENDING)
      {
pncb = (PNCB)GlobalAlloc(GMEM_FIXED, sizeof(NCB));
     memcpy(pncb, & GlobalClients[i], sizeof(NCB));
        pncb->ncb_event = 0;
     hThread = CreateThread(NULL, 0, ClientThread,
(LPVOID)pncb, 0, & ThreadId);
        CloseHandle(hThread);
        // 重设事件句柄, 进行另一个侦听
        ResetEvent(EventArray[i]);
Listen(& GlobalClients[i], lenum.lana[i], SERVER_NAME);
      }
    }
}
```

最后，关闭所有句柄，释放内存空间。

```
for(i = 0; i < lenum.length; i++)
{
    DelName(lenum.lana[i], SERVER_NAME);
    CloseHandle(EventArray[i]);
}
GlobalFree(GlobalClients);
```

2. 异步事件客户机的实现

同实现服务器一样先进行初始化。在给 LANA 编号添加客户机名字之后，进行异步连接，若连接成功，则向服务器发送数据，最后，关闭句柄，释放内存空间。这里我只给出异步连接和发送数据的部分程序代码：

```
// 产生一个事件, 并把它分配给一个相应的 NCB 结构, 并执行异步连接
for(i = 0; i < lenum.length; i++)
{;
EventArray[i] = CreateEvent(NULL, TRUE, FALSE, NULL);
    pncb[i].ncb_event = EventArray[i]
    AddName(lenum.lana[i], ClientName, & Num);
    Connect( & pncb[i], lenum.lana[i], ServerName,
ClientName);
}
// 等待, 直到至少有一个连接成功
RetEvent = WaitForMultipleObjects(lenum.length, EventArray, FALSE, INFINITE);
if (RetEvent == WAIT_FAILED)
{
    ErrorCode.Format("等待连接失败!");
    AfxMessageBox(ErrorCode);
}
else
{
// 如果有多个连接成功, 关闭多余的连接, 我们只用由 WaitForMultipleObjects 函数返回的连接; 如果没有连接成功, 则取
```

下转第 77 页

AVI 文件格式动画生成技术

陶　胜

随着多媒体技术的飞速发展，多媒体技术广泛应用于信息领域中，出现了各种各样的多媒体数据文件，AVI 文件就是常见的一种。比起 Windows 其它的资源性数据文件，AVI 文件的结构要复杂得多。本文探讨编程实现生成 256 色压缩和非压缩的 AVI 动画文件。

一、AVI 文件格式分析

AVI 文件是 Microsoft 公司制定的一种 RIFF Resource Interchange File Format 文件格式，主要由各种各样的块 chunk 构成，块包括块头和块体两部分，块头的结构如下：

```
struct chunkhead  {
  unsigned char type[4];
  unsigned long size;
 }
```

前 4 个字节标识块的类型，后一个无符号长整型数记载块体的长度；而块体可能是基本数据，也可能是一个具有块头和块体结构的子块。但最底层子块的块体必须是基本数据，称为数据块，而称非底层子块为构造块。数据块的块标识指明数据的性质，例如：

块标识	数据性质
“avih”	AVI 头
“strh”	数据流头
“strf”	数据流格式
“00db”	非压缩视频数据
“00dc”	压缩视频数据
“00dx”	压缩视频数据
“01wb”	音频数据

而构造块的标识一律都是“LIST”，故也称为 LIST 块，为进一步区别其属性，在块体的前面，用 4 个字节来标识其类型，例如：

标识字节	类型
“hdrl”	AVI 总参数头构造
“strl”	流参数构造
“movi”	流数据构造

整个 AVI 文件就是一个以“RIFF”为标识的构造块，其构造类型为“AVI ”，它包括两个必备的 LIST 块和一个可选的索引块。索引块的标识为“idx1”，其作用是在播放时可以随机地访问任一幅画面或任一段声音。第一个 LIST 块为参数块，其底层块记载视频音频格式参数，分别采用 BMP 和 WAV 文件的格式规范。限于篇幅，本文只讨论不含音频数据的 256 色 AVI 动画文件的生成。第一个 LIST 块的层次结构描述如下：

偏址	内容	长度
0xc	LIST 块头 1	8
0x14	“hdrl” LIST 块标识	4
0x18	avih 块头	8
0x20	avih 块体	0x38
0x58	LIST 块头 2	8
0x60	“strl” LIST 块标识	4
0x64	strh 块头	8
0x6c	strh 块体	0x38
0xa4	strf 块头	8
0xac	bmp 头	0x28
0xd4	调色板数据	4 * n

其中 n 为实际使用的颜色数（n≤256），LIST 块头 2 记载块体的长度为 116 + 4 * n，LIST 块头 1 记载块体的长度为 192 + 4 * n。至于 avih 块体、strh 块体、bmp 头的结构，请参阅程序。

第二个 LIST 块为记录块，其底层块记载视频数据，视频数据块的块标识可能是“00db”或“00dc”等等，而视频数据可能是非压缩的或压缩的。第二个 LIST 块的层次结构描述如下：

内容	长度
第二个 LIST 块块头	8
movi 第二个 LIST 块标识	4
第一个视频数据块块头	8
第一帧视频数据	size1
… …	…
第 N 个视频数据块块头	8
第 N 帧视频数据	sizeN

第二个 LIST 块块头记载块体的长度为 4 + 8 * N + size1 + size2 + …sizeN。

AVI 文件中还可以有一种“废块”，标识为“JUNK”，是数据块格式，其作用是便于在文件中随机地增删新旧视频音频记录，而不致于造成 AVI 文件写操作中定位的困难。本文在生成 AVI 动画时不考虑 JUNK 块。

二、RLE8 压缩方法

视频数据采用 BMP 格式，对于非压缩情形，它是以图像的左下角为起点，按照从左至右，由下而上的次序，将图像

数据存入文件，256 色图像是一个字节记录一点；采用压缩方式处理数据也是按照上述次序，而压缩方法有多种。本文介绍 Microsoft Windows 的游程编码 RLE8 Run Length Encodeing 压缩方法：对数据流中同一数据游程进行压缩，即以两个字节代表一串相同数值的数据，第一个字节表示有多少个相同的数据，第二个字节表示此相同的数据，这两个字节最多能代替 255 个连续重复出现的数据，如果连续重复出现的数据的个数超过 255 个，则必须两组或两组以上的压缩码来代表，这就是 RLE8 的计数方式；对于一串互不相同的数据，则采用识别码，识别码共有 4 种，其第一个字节必须是 0，如下定义：

●0x00 0x00 表示一行图像数据的结束。RLE8 压缩法以行为压缩单元，每行数据经压缩后，在末端加入 0x00 0x00 两个字节。

●0x00 0x01 表示所有图像数据的结束。当所有的行都压缩处理后，就加入 0x00 0x01 两个字节，作为结束标志。

●0x00 0x02 X Y 表示向右移 X 点，向下移 Y 点。

●0x00 N…表示有 N 点不同值的图像数据。如果不同值的图像数据为奇数个，则必须在数据末端加入 0x00 ，以维持压缩数据长度为偶数个字节。

例如：有一串原始数据为 0x16 0x16 0x16 0x16 0x42 0x36 0x04 ，用 RLE8 压缩数据为 0x04 0x16 0x00 0x03 0x42 0x36 0x04 0x00。

注意：当不同点的个数为 1 或 2 时，应视为相同 个数为 1）的情形来处理。

三、AVI 动画生成

本程序用 Turbo C2. 0 在标准模式下开发而成。在屏幕上显示 “AVI Animator! 的动画，然后按照 AVI 动画的文件格式生成动画文件 animator. avi。程序中 compression 的值为 0，则生成 256 色非压缩的动画文件；compression 的值为 1，则应用 Microsoft Windows 的 RLE8 压缩法压缩生成 256 色压缩格式的动画文件。生成的动画文件 animator. avi 可直接在 Windows 下播放。本程序在 Turbo C2. 0 Small 模式下编译通过。程序代码如下：

```
#include  <stdio. h>
#include  <io. h>
#include  <math. h>
#include  <alloc. h>
#include  <stdlib. h>
#include  <graphics. h>
#define widthbytes(i) ((i+31)/32*4)
int frames, colors, linebytes, color, bkcolor;
int left, top, right, bottom, bmpwidth, bmpheight;
unsigned long compression, imagebytes,
            offset, *bmpsize;
struct chunkhead
 {
  unsigned char type[4];
  unsigned long size;
 }riff, list, block, subblock;
struct aviheader
 {
  unsigned long microsecperframe,
           maxbytespersec, reserved1;
  unsigned long flags, totalframes,
           initialframes, streams;
  unsigned long suggestedbuffersize, width,
           height, reserved[4];
 }avih;
struct avistreamheader
 {,
  unsigned char type[4];
  unsigned long compression, reserved1,
           reserved2, reserved3;
  unsigned long streams, quality,
           initialframes, totalframes;
  unsigned long suggestedbuffersize
           samplesize, reserved[2];
  unsigned int width, height;
 }strh;
struct bitmapinfo
 {
  unsigned long size, width, height;
  unsigned int plane, bitsperpixel;
  unsigned long compression, imagesize,
           xpels, ypels;
  unsigned long colorused, colorimportant;
 }bmp;
struct idxchunk
 {;)
  unsigned char type[4];
  unsigned long flags, offset, length
 }idx;
void getacolor(int k, int acolor[3]
 {
  int j, color[16]={0, 1, 2, 3, 4, 5, 22, 7, 56, 57, 58,
          59, 60, 61, 62, 63};
  j=color[k];
  if(j= =22) j=20;
  outportb(0x3c7, ((unsigned char)j));
  acolor[0] =inportb(0x3c9);
  acolor[1] =inportb(0x3c9);
  acolor[2] =inportb(0x3c9);
 }
void getcolortable(int colortable[16][3])
 {;
  int i, j;
  int color[16]={0, 1, 2, 3, 4, 5, 22, 7, 56, 57, 58,
          59, 60, 61, 62, 63};
  for(i=0; i<16; i+ +)     {
   j=color[i];
   if(i= =6) j=20;
   outportb(0x3c7, ((unsigned char)j));
   colortable[i][0] =inportb(0x3c9)
   colortable[i][1] =inportb(0x3c9);
```

```
  colortable[i][2] =inportb(0x3c9);
  }
 }
void createchunkhead(FILE  * fp)
 {
  int i, colortable[16][3];
  unsigned char pal[16][4], type[4] =″    ″;
  linebytes =widthbytes(8 * bmpwidth);
  memcpy(riff. type, ″RIFF″, 4);
  riff. size =224L +4 * colors +24 * frames
    +(unsigned long)linebytes * bmpheight * frames;
  fwrite(& riff, sizeof(riff), 1, fp);
  memcpy(type, ″AVI ″, 4);
  fwrite(type, 4, 1, fp);
  memcpy(list. type, ″LIST″, 4);
  list. size =(unsigned long)192 +4 * colors;
  fwrite(& list, sizeof(list), 1, fp);
  memcpy(type, ″hdrl″, 4);
  fwrite(type, 4, 1, fp);
  memcpy(block. type, ″avih″, 4);
  block. size =0x38;
  fwrite(& block, sizeof(block), 1, fp);
  avih. microsecperframe =0x1046b;
  avih. maxbytespersec =0x640;
  avih. reserved1 =0;
  avih. flags =0x810; avih. totalframes =frames;
  avih. initialframes =0; avih. streams =1;
  avih. suggestedbuffersize =(unsigned
      long)linebytes * bmpheight;
  avih. width =bmpwidth; avih. height =bmpheight;
  avih. reserved[0] =0; avih. reserved[1] =0;
  avih. reserved[2] =0; avih. reserved[3] =0;
  fwrite(& avih, sizeof(avih), 1, fp);
  memcpy(block. type, ″LIST″, 4);
  block. size =(unsigned long)116 +4 * colors;
  fwrite(& block, sizeof(block), 1, fp);
  memcpy(type, ″strl″, 4);
  fwrite(type, 4, 1, fp);
  memcpy(subblock. type, ″strh″, 4);
  subblock. size =0x38;
  fwrite(& subblock, sizeof(subblock), 1, fp);
  memcpy(strh. type, ″vids″, 4);
  strh. compression =0; strh. reserved1 =0;
  strh. reserved2 =0; strh. reserved3 =0;
  strh. streams =1; strh. quality =15;
  strh. initialframes =0; strh. totalframes =frames;
  strh. suggestedbuffersize =(unsigned long)linebytes * bmp-
height;
  strh. samplesize =0;
  strh. reserved[0] =0; strh. reserved[1] =0;
  strh. width =bmpwidth; strh. height =bmpheight;
  fwrite(& strh, sizeof(strh), 1, fp);
  memcpy(subblock. type, ″strf″, 4);
  subblock. size =0x28 +4 * colors;
  fwrite(& subblock, sizeof(subblock), 1, fp);
  bmp. size =0x28;
  bmp. width =bmpwidth;
  bmp. height =bmpheight;
  bmp. plane =1; bmp. bitsperpixel =8;
  bmp. compression =compression;
  bmp. imagesize =(unsigned long)linebytes * bmpheight;
  bmp. xpels =0x0ece; bmp. ypels =0xec4;
  bmp. colorused =colors;
  bmp. colorimportant =colors;
  fwrite(& bmp, sizeof(bmp), 1, fp);
  getcolortable(colortable);
  for(i =0; i <colors; i + +) {
   pal[i][0] =(unsigned char)colortable[i][2] < <2;
   pal[i][1] =(unsigned char)colortable[i][1] < <2;
   pal[i][2] =(unsigned char)colortable[i][0] < <2;
   pal[i][3] =0;
   }
  fwrite(pal, 1, 4 * colors, fp);
  memcpy(list. type, ″LIST″, 4);
 list. size =4L +8 * frames + (unsigned long)linebytes * bmp-
height * frames;
  fwrite(& list, sizeof(list), 1, fp);
  memcpy(type, ″movi″, 4);
  fwrite(type, 4, 1, fp);
 }
void makeavi(FILE  * fp)
 {
  unsigned char byte;
  int x, y;
  memcpy(block. type, ″00db″, 4);
 block. size =(unsigned long)linebytes * bmpheight;
  fwrite(& block, sizeof(block), 1, fp);
  for(y =0; y <bmpheight; y + +)
   for(x =0; x <linebytes; x + +)
    if(x <bmpwidth)     {
     byte =getpixel(left +x, bottom -y)>0?
          color: bkcolor;
     fputc(byte, fp);
    }
   else
    fputc(0, fp);
 }
void makeidxchunk(FILE  * fp)
 {
  int i;
  memcpy(block. type, ″idx1″, 4);
  block. size =(unsigned long)16 * frames;
  fwrite(& block, sizeof(block), 1, fp);
  memcpy(idx. type, ″00db″, 4);
  idx. flags =0x10; idx. offset =4;
 idx. length =(unsigned long)linebytes * bmpheight;
  for(i =1; i < =frames; i + +)  {
   fwrite(& idx, sizeof(idx), 1, fp);
   idx. offset + =8 +idx. length;
   }
 }
void putRLE8code(FILE  * fp, unsigned char  * buffer, un-
signed int bytes)
 {
```

```
  unsigned int count;
  if(bytes>2)  {
   fputc(0, fp);
   fputc(bytes, fp);
   imagebytes + =2;
   for(count =0; count <bytes; count + +)  {
    fputc(buffer[count], fp);
    imagebytes + +;
   }
   if(bytes& 1)  {
    fputc(0, fp);
    imagebytes + +;
   }
  }
 else  {
  for(count =0; count <bytes; count + +)  {
   fputc(1, fp);
   fputc(buffer[count], fp);
   imagebytes + =2;
   }
  }
 }
void encodeRLE8(FILE  * fp)
 {
  unsigned char linebuf[320], buffer[640];
  unsigned int x, y, linebytes, bytes, bytecount, nonbytes;
  linebytes =bmpwidth; y =bmpheight;
  while(y - -) {
   bytecount =nonbytes =bytes =0;
   for(x =0; x <linebytes; x + +)  {
linebuf[x] =getpixel(left +x, top +y +1)>0?
         color: bkcolor;
    } =;
   while(bytecount <linebytes)      {
while(linebuf[bytecount +bytes] =
     linebuf[bytecount +bytes +1]     & &
(bytecount +bytes +1) <linebytes& & bytes <254)
    bytes + +;
    if(bytes>1)  {
     if(nonbytes)   {
      putRLE8code(fp, buffer, nonbytes)
      nonbytes =0;
      }
     bytes + +;
     fputc(bytes, fp);
     fputc(linebuf[bytecount], fp);
     bytecount + =bytes;
     bytes =0;
     imagebytes  + = 2;
     };
    else   {
     if(bytes)
      bytes - -;
     buffer[nonbytes + +] =linebuf[bytecount + +];
     if(nonbytes = =255)  {
      putRLE8code(fp, buffer, nonbytes)
      nonbytes =0;
      }
     }
    };
   if(nonbytes)
    putRLE8code(fp, buffer, nonbytes)
   fputc(0, fp); fputc(0, fp);
   imagebytes + =2;
   }
   fputc(0, fp); fputc(1, fp);
   imagebytes  + = 2;
  }
unsigned long formavi(FILE  * fp)
 {
  imagebytes =0;
  fwrite(& block, sizeof(block), 1, fp);
  encodeRLE8(fp);
  fseek(fp, -8 -imagebytes, 1);
  memcpy(block. type, "00dc", 4);
  block. size =imagebytes;
  fwrite(& block, sizeof(block), 1, fp);
  list. size + =8 +imagebytes;
  fseek(fp, imagebytes, 1);
  return(imagebytes);
 }
void formidxchunk(FILE  * fp)
 {
  int i;
  memcpy(block. type, "idx1", 4);
  block. size = (unsigned long)16 * frames;
  fwrite(& block, sizeof(block), 1, fp);
  memcpy(idx. type, "00dc", 4);
  idx. flags =0x00; idx. offset =4;
  for(i =0; i <frames; i + +)  {
   idx. length =bmpsize[i];
   fwrite(& idx, sizeof(idx), 1, fp);
   idx. offset + =8 +idx. length;
   }
 }
void showimage( )
 {
  int i, charsize =4;
  char text[ ] ="AVI Animator! ";
  FILE  * fp;
  settextstyle(TRIPLEX_FONT, HORIZ_DIR, charsize);
  left =220; top =200; right =420; bottom =280,
  compression =1; frames =21;
  bmpwidth =right -left; bmpheight =bottom -top;
  color =GREEN; bkcolor =BLACK, colors =16;
  if((fp =fopen("Animator. avi", "wb")) = =NULL) {
   printf("Can not create avi File! \n");
   exit(1);
  }
  createchunkhead(fp);
  list. size =4;
  bmpsize =malloc(frames * sizeof (unsigned long));
  setbkcolor(bkcolor);
  for(i =0; i <frames; i + +) {
```

```
   setcolor(color);
   outtextxy(right - 20 * i - 20, top + 20, text);
   if(compression)
    bmpsize[i] = formavi(fp);
   else
    makeavi(fp);
   delay(200);
   setcolor(bkcolor);
   outtextxy(right - 20 * i - 20, top + 20, text);
   cleardevice( );
  }
  if(compression)  {
   formidxchunk(fp);
   riff.size = 220L + 4 * colors + 16 * frames + list.size;
   fseek(fp, 0L, 0);
   fwrite(& riff, sizeof(riff), 1, fp);
   offset = 212L + 4 * colors;
   fseek(fp, offset, 0);
   memcpy(list.type, "LIST", 4);
   fwrite(& list, sizeof(list), 1, fp);
   }
  else
   makeidxchunk(fp);
  free(bmpsize);
  fclose(fp);
 }
main( )
 {
  int g_driver, g_mode, g_error;
  detectgraph(& g_driver, & g_mode);
  if(g_driver < 0)  {
    printf("no graphics hardware detected ! \n");
    exit(1);
   };
  if(registerbgidriver(EGAVGA_driver) < 0)
   exit(1);
 if(registerbgifont(triplex_font) ! = TRIPLEX_FONT)
   exit(1);
  initgraph(& g_driver, & g_mode, " ")
  g_error = graphresult( );
  if(g_error < 0)   {
printf("initgraph error:  %s. \n",
      grapherrormsg(g_error));
    exit(1);
   }
  cleardevice( );
  showimage( );
  closegraph( );
 }
```

参考文献

1. 李振辉、李仁和编著. 探索图像文件的奥妙. 清华大学出版社

2. 刘瑞祥等. AVI 文件格式探秘. 电子与电脑 1997 年 4 月

收稿日期：2000 年 9 月 13 日）

上接第 72 页）

消.

```
       for(i = 0; i < lenum.length; i + +)
       {
           if (i ! = RetEvent)
           {
               if (pncb[i].ncb_cmd_cplt  = = NRC_PENDING)
                   Cancel(& pncb[i]);
               else
        Hangup(pncb[i].ncb_lana_num, pncb[i].ncb_lsn);
           }
       }
       // 发送消息
       sprintf(SendBuffer, m_sendmsg);
       RetValue = Send(pncb[RetEvent].ncb_lana_num, pncb
[RetEvent].ncb_lsn, SendBuffer, strlen(SendBuffer));
       if (RetValue ! = NRC_GOODRET)
           AfxMessageBox("无法建立连接!");
           Hangup(pncb[RetEvent]   .ncb_lana_num,      pncb
[RetEvent].ncb_lsn);
      }
```

本程序在 Windows98 环境下，由 VC + + 6.0 编译通过。

参考文献

1. Anthony Jones Jim Ohlund Network Programming for Microsoft Windows 1999。

2. MSDN Library。

收稿日期：2000 年 9 月 25 日）

用 AVICap 实现视频捕获

刘 佳

一、前言

在本实验室所开发的基于 Internet 的实时视频通信系统中，本人主要负责视频的采集和压缩编码部分。其中，视频源是由摄像头直接捕获获得的。根据当前的视频采集设备的发展趋势和实验条件，我们选用的是 USB 接口，支持 VFW，纯软件采集，无硬件支持的摄像头，这种摄像头已成为目前视频采集设备中的主流。在开发应用的过程中，本人在程序中使用了 AVICap 窗口类来实现视频的采集，这部分是后续的视频压缩处理和通信传输的基础。在工作中，本人对 AVICap 的使用进行了一定的学习和尝试，由于开发过程中发现这方面的中文学习资料很少，英文帮助和例程对初学者可能比较难于入手，故想在这里将自己学到的一点知识和编程中的一些心得介绍给大家，希望能对其他读者有所帮助。

二、AVICap 简介

AVICap 窗口类是 VFW （Video for Windows）的一个重要组成部分，它的主要作用是实现视频的捕获。AVICap 为应用程序提供了一个简单的，基于消息的接口，通过该接口，程序可以访问视频和波形音频硬件并控制视频流到硬件的捕获。通过 AVICap，程序员可以轻松地将视频捕获功能加入自己的应用程序。

AVICap 支持实时视频流捕获和单帧捕获。此外，AVICap 使程序员可以控制视频源的开始和结束位置，并加入了如序列帧捕获等功能。

使用 AVICap 生成的捕获窗具有以下功能：

●将音频和视频流捕获到 AVI 文件。

●动态地连接或断开音频和视频输入设备。

●以叠加 （overlay）或预览 （preview）模式显示输入的实时视频信号。

●指定捕获所用的文件，并可将捕获文件的内容拷贝到另一个文件。

●设置捕获速率。

●显示控制视频源和视频格式的对话框。

●创建，保存和载入对话框。

●将图像和调色板拷贝到剪贴板。

●捕获和保存 DIB 格式单帧图像。

最后，需要指出的是 AVICap 所能提供的功能是硬件相关的，许多功能，如以叠加 （overlay）模式显示视频，捕获音频，以及所支持的视频采集格式是否可用等应由所选择的视频捕获设备而定。具体应用设计时要根据硬件所支持的性能来灵活处理。所幸的是，我们可以通过检查 CAPDRIVERCAPS 结构来获取捕获驱动器的能力，并通过 AVICap 提供的 Video Source Video Format Video Display 对话框来对捕获参数进行设置。如何实现，请见下部分的介绍。

三、用 AVICap 实现视频捕获

用 AVICap 实现视频捕获主要分为以下几个步骤：

1. 创建捕获窗

捕获窗是所有捕获操作的基础，其构造要调用：

```
ghWndCap = capCreateCaptureWindow(//创建捕获窗函数
           NULL,                      // 窗口名称
           WS_CHILD | WS_VISIBLE,     // 窗口风格
           0, 0, 176, 144,            // 窗口位置和大小
           hMainWin,                  //父窗口句柄
           1                          //窗口 ID
         );
if (ghWndCap = = NULL) //如创建不成功, 返回 false
    return false;
```

其中 ghWndCap 为由 HWND ghWndCap 定义的捕获窗句柄。

2. 将捕获窗与视频捕获驱动相连

```
BOOL fOK;
fOK = capDriverConnect(ghWndCap, 0);
if(!fOK)
{
//无法将指定的视频捕获驱动连接到捕获窗
//在此加入错误处理
}
```

3. 获得捕获驱动器的能力

```
capDriverGetCaps(hwndCap, & gCapDriverCaps, sizeof
(CAPDRIVERCAPS));
```

其中 gCapDriverCaps 是由 CAPDRIVERCAPS gCapDriverCaps 定义的。CAPDRIVERCAPS 结构定义了捕获驱动器的能力，如有无视频叠加能力，有无控制视频源、视频格式的对话框等。

执行 capDriverGetCaps 语句后，gCapDriverCaps 中就获得了与当前捕获窗相连的捕获驱动的各项能力，要根据该捕获驱动的能力来实现视频的显示和捕获。

4. 显示视频

上已提及，显示视频有两种模式：叠加 （overlay）或预览 （preview）模式。

由于叠加模式只被部分视频捕获卡支持，而大部分视频捕获设备都支持预览模式，我们用预览模式显示视频：

```
capPreviewRate(ghWndCap, 66);        //设置显示帧率(本例中为 66ms/帧)
capPreview(ghWndCap, TRUE);          //开始预览显示
```

要停止预览可用

```
capPreview(ghWndCap, FALSE);
```

用叠加模式显示视频的方法相似，只是所用函数应为 capOverlay。

5. 进行视频捕获

视频捕获主要有两种模式，连续视频流捕获和单帧捕获。

在进行视频捕获之前要指定捕获文件名并为捕获文件分配存储空间。

```
char szCaptureFile[] = "MYCAP.AVI";     //捕获文件名为"MYCAP.AVI"
capFileSetCaptureFile(ghWndCap, szCaptureFile);     //指定捕获文件为"MYCAP.AVI"
capFileAlloc(ghWndCap, (1024L * 1024L * 5));     //为捕获文件分配空间
```

连续视频流捕获用：

```
capCaptureSequence(ghWndCap);
```

单帧捕获使用：

```
capGrabFrame(ghWndCap);
```

要结束视频捕获可用：

```
capCaptureAbort(ghWndCap);
```

还有一种捕获方式是不将捕获结果写到文件 这时 可使用回调函数来直接处理捕获到的数据，关于这种方式，我们在第四节有详细介绍。

6. 设置视频源，视频格式和显示对话框

每个视频捕获驱动最多具有三个不同的对话框用于控制视频的数字化和捕获方法，可调用这些对话框完成对视频捕获的控制。由 (3) 中获得的 gCapDriverCaps 中的成员可判断当前的捕获驱动是否具有这些对话框。

```
if (gCapDriverCaps.fHasDlgVideoSource)  //如果具有设置视频源对话框
    capDlgVideoSource(ghWndCap);        //显示该对话框
if (gCapDriverCaps.fHasDlgVideoFormat)  //如果具有设置视频格式对话框
    capDlgVideoFormat(ghWndCap);        //显示该对话框
if (gCapDriverCaps.fHasDlgVideoDisplay) //如果具有设置显示对话框
    capDlgVideoDisplay(ghWndCap);       //显示该对话框
```

在显示的对话框中，该捕获驱动支持的视频源格式、捕获格式、显示格式都会被列出，用户可根据需要进行选取、设置。

7. 结束，将捕获窗同驱动断开连接

```
capDriverDisconnect (ghWndCap);
```

这一步在结束视频捕获程序时是必须的，否则将导致视频驱动无法释放，其它程序将不能使用捕获设备。

四、实现 AVICap 的高级功能

在实际应用中，我们有时候需要对捕获的视频流进行处理，如在视频通信系统中要进行压缩编码，这时，就需要我们在基本的视频捕获程序中加入一些其它函数，如回调函数，并在其中进行处理。而且，在实际应用中，也会遇到如当有多个捕获驱动时的检测问题，设置控制视频捕获过程的参数问题和视频数据的格式设置问题等，下面，就一些常见的问题及如何运用回调函数在视频捕获中加入视频处理进行基本的介绍。

1. 列举已安装的视频捕获驱动

当有多个视频捕获驱动已安装时，可依次获得各视频捕获驱动的名称及版本信息。

```
char szDeviceName[80];     //用于存放捕获驱动的名称
char szDeviceVersion[80];  //用于存放捕获驱动的版本号
for (UNIT wIndex = 0; wIndex < 10; wIndex++)
{
if (capGetDriverDescription (
wIndex,                   //视频捕获驱动的索引号,从 1 到 10
szDeviceName,             //视频捕获驱动的名称
sizeof (szDeviceName),
szDeviceVersion,          //视频捕获驱动的版本号
sizeof (szDeviceVersion))
        {
//将该捕获驱动的名称加入到已安装的视频捕获驱动列表
//使用户可选择使用那个驱动进行捕获
        }
)
```

2. 改变视频捕获参数

结构 CAPTUREPARMS 中包含控制视频捕获过程的各种参数，如捕获帧频、捕获时间限制、捕获时的缓存数、捕获如何被终止等。用 capCaptureGetSetup 和 capCaptureSetSetup 可完成捕获参数的获取和设置。下面的例子是将捕获帧率从省缺的 15 帧/秒设置为 15 帧/秒。其它参数可根据应用的需要进行设置，方法不变。

```
capCaptureGetSetup(ghWndCap, &gCapParms, sizeof (CAPTUREPARMS));
//获得当前视频捕获参数
float FramesPerSec = 10.0;     //待设帧率
    gCapParms.dwRequestMicroSecPerFrame = (DWORD) (1.0e6/ FramesPerSec);
//计算出每帧的时间(微秒),并赋予相应的成员
capCaptureSetSetup(ghWndCap, &gCapParms, sizeof (CAPTUREPARMS));
//设置新的视频捕获参数
```

其中，gCapParms 由 CAPTUREPARMS gCapParms 定义，CAPTUREPARMS 结构定义了控制视频流捕获过程的参数。

3. 获取并设置视频格式

前面已经讲过可用视频格式对话框来设置视频格式，在实际应用中，有时候要根据处理视频的要求在程序中自动完成格式设置，如在低码率视频压缩标准 H.263 中要处理的标准图

像格式是 QCIF YUV4 1 1，大小 176 * 144 ，这时，可用 capSetVideoFormat 完成格式设置。

需要注意的是在获取并设置视频格式所用到的参数之一是 BITMAPINFO 结构的变量， BITMAPINFO 结构是可变长结构。故应用在检索当前视频格式之前必须用 capGetVideoFormatSize 获得该结构的大小。

下面的例子先使用 capGetVideoFormatSize 获取 BITMAPINFO 结构的大小，再用 capGetVideoFormat 获取当前的视频格式，如当前格式不为 QCIF 时用 capSetVideoFormat 进行设置，若设置成功，则当前视频格式将变为 QCIF YUV4 1 1，大小 176 * 144 。

```
LPBITMAPINFO lpbi;
DWORD dwSize;
BOOL fOK;
dwSize = capGetVideoFormatSize(ghWndCap);
//获取 BITMAPINFO 结构的大小
lpbi = (LPBITMAPINFO)new unsigned char[dwSize];
//分配空间
capGetVideoFormat(ghWndCap, lpbi, dwSize); //获取当前的视频格式
if(lpbi->bmiHeader.biWidth! =176    ||    lpbi->bmiHeader.biHeight! =144||
   lpbi->bmiHeader.biCompression! =mmioFOURCC( ´I´, ´4´, ´2´, ´0´)) //若不为 QCIF 格式
     {    //设置为所要求的格式
          lpbi->bmiHeader.biWidth=176; //位图的宽度
          lpbi->bmiHeader.biHeight=144; //位图的高度
          lpbi->bmiHeader.biCompression = mmioFOURCC(´I´, ´4´, ´2´, ´0´);
//图像压缩格式设置为 I420(即 YUV4: 1: 1)
          lpbi->bmiHeader.biBitCount=12;
//每个像素所用的比特数
          lpbi->bmiHeader.biSizeImage=38016;
//该结构所用总比特数
     fOK =capSetVideoFormat(ghWndCap, lpbi, dwSize);
//设置为 QCIF 格式
          delete lpbi;
          return fOK;
//设置成功返回 TRUE, 否则返回 FALSE
     }
     else// 已为 QCIF 格式, 返回 TRUE
     {
          delete lpbi;
          return TRUE;
     }
```

由于视频格式是与设备相关的，故应用程序应检查 capSetVideoFormat 的返回值以判断该视频采集设备是否支持指定的视频格式并采取相应的措施。如该视频采集设备不支持此种格式可用其它格式进行采集再加入格式转换函数。

4. 使用回调函数

回调函数使应用能够直接使用音频和视频。

应用程序可用捕获窗来登记回调函数，在发生下列情况时，它能通知应用程序，此时被登记的回调函数将被自动调用。

●状态改变

●出错

●视频帧和音频缓存可用

●在捕获过程中，应用处于让步 (yield) 地位

下面我们举例说明回调函数的使用。

我们以 capVideoStreamCallback 为例，该回调函数是与流式采集相结合使用的，用于选择性地处理捕获的视频帧。当捕获设备标记视频缓存已装满后，捕获窗将自动调用视频流回调函数。它的使用分以下几步

首先，登记回调函数：

```
capSetCallbackOnVideoStream(ghWndCap, & StreamCallbackProc);
//回调函数 StreamCallbackProc 应已被提前声明
```

然后, 定义回调函数:

```
LRESULT CALLBACK StreamCallbackProc(HWND hWnd, LPVIDEOHDR lpVHdr)
{
if (lpVHdr->dwFlags & VHDR_DONE) //视频缓存已装满
        {
//此时 lpVHdr->lpData 中存放的就为已捕获的一帧视频数据
//可在此加入自己的处理函数
//注意要将 lpVHdr->lpData 内的数据及时保存
        }
return (LRESULT) TRUE;
  }
```

其中，VIDEOHDR 结构定义了视频数据块的头信息。

最后，取消已定义的回调函数：

```
capSetCallbackOnVideoStream(ghWndCap, NULL);
```

在我们的程序设计中，在视频捕获中采用

```
capCaptureSequenceNoFile(ghWndCap);
```

进行捕获，capCaptureSequenceNoFile 以不存入文件的方式捕获，由回调函数 StreamCallbackProc 直接处理捕获数据，进行编码等操作。

五、小结

本文介绍了使用 AVICap 窗口类开发视频捕获程序和视频通信系统的基本方法，对于其基本实现给予了示例和说明，并对一些关键问题的实现方法予以较为详细的解释，有关 AVICap 中其它函数的使用请参见 MSDN 的帮助文件和 vidcap 例程。

Video for Windows (VFW，Windows 环境下的视频服务) 软件是 Microsoft 公司开发，随 Windows 操作系统发布的，它使多媒体个人计算机 (MPC) 具有处理视频信号的能力。VFW 的一个关键思想是播放时不需要专用硬件，它能使应用程序数字化并播放从传统模拟视频源得到的视频。

收稿日期：2000 年 10 月 10 日)

AutoCAD 图像控件中动态数字的实现

熊 辉

摘 要 介绍一种在 AutoCAD 图像控件中实现动态响应用户输入数据的功能及其实现方法。

主题词 图像控件,PDB 技术,单个数字写函数,尺寸数字写函数

一、前言

AutoCAD 是目前流行的计算机辅助设计平台。用 AutoCAD 内嵌的 Autolisp 语言和 PDB 可编程对话框）技术，可以为参数化绘图提供良好的数据输入界面。为了增强视觉效果，对话框中可有图像控件。

一般情况下，图像控件中只允许出现一幅静态的幻灯画面，但在多数情况下，编程人员和用户希望图像控件能动态地响应用户输入的尺寸数值，实现所见即所得的功能。参数化绘螺钉时的数据录入对话框中对话框左边为用户输入外径和长度的编辑框，右边为图像控件。要求用户分别在列表框中输入外径和长度或在编辑框中输入长度时，右边的图像控件能动态地响应这个输入。如将螺钉长度改为 80，则图像控件中的长度将由 50 变为 80。

用 ObjectARX 用 visual C++ 语言）或 ActiveX Automation 用 Visual Basic 语言 技术开发 AutoCAD 应用程序，可很容易的用 WINDOWS 对话框来实现上述功能。但由于目前大多数二次开发的 AutoCAD 应用程序是用的 Autolisp 语言，如果对这些应用程序进行改进，则最好用 Autolisp 语言来实现上述功能。因此，开发用 Autolisp 语言及 PDB 技术在 AutoCAD 图像控件中实现这种功能的方法是很有意义的。

下面介绍一种用 Autolisp 语言及 PDB 技术在 AutoCAD 图像控件中实现这种功能的方法。本方法的实质是采用在图像控件中 ‘画尺寸” 的方法，来使尺寸数字变化，从而达到 WINDOWS 对话框的使用效果。

二、实行方法

要实现上述功能，程序员必须解决以下几个问题：

1. 由于图像控件中的图像是一幅一幅地迭加上去的，故必须清除图像控件中原有的数据。如要将数据由 50 改为 80，则必须先清除 50 才能再写 80。

2. 由于 PDB 技术中没有专门的向图像控件写数字的功能，故必须增加向图像控件写数字的功能。

3. 图像控件应立即显示用户输入的数字。

分析 AutoCAD 图像控件和其它控件的操作，以上问题可用下述方法解决：

1. 用 fill_image x1 y1 x2 y2 color 函数可实现清除图像控件中原有数据的功能。具体方法为将参数 x1、y1、x2、y2 设置成刚好覆盖欲清除的数字的座标值，将参数 color 设置成和图像控件底色相同，调用该函数即可清除图像控件中原有的数据。

2. 用 vector_image x1 y1 x2 y2 color 函数可实现向图像控件写数字的功能。具体方法为将参数 x1、y1、x2、y2 设置成欲写数字处的座标值，将参数 color 设置成前景色，调用该函数即可写出数字的一个笔画。将多个笔画组合起来即可组成单个的数字，将多个数字组合起来即可。

如要在图像控件中的座标值为 （10，20）处写出数字 “10” 可采用如下步骤：

㈠ 定义数字 “1” 的写函数 defun num1 a b vector_image + a 3 + b 0 + a 3 + b 9 0 。

式中 num1 为函数名；a 和 b 为形参。本函数的功能是向图像控件写数字 1，本函数调用了画直线函数 vector_image。被调函数的参数为 + a 3 、 + b 0 、 + a 3 、 + b 9 和 0，前者为直线 1 的始和终点座标值，后者为这条直线的颜色代码，此处取为 0 值，即前景色。用本函数定义的数字组成数字时，要求相邻两个数字其 X 座标值相差为 7，Y 座标值相差为 0；

㈡用与步骤㈠相似的方法定义数字 ‘0” 的写函数 具体函数见附录）；

㈢调用函数 num1 向座标 （0，20）处写数字 “1”，方法为 num1 10 20）。num1 为函数名，10 和 20 为实参；

㈣调用函数 num0 向座标 （7，20）处写数字 0，方法为 num1 17 20）。因为数字 10 的 1 和 0 之间的 x 座标间相差为 7；y 座标间相差为 0，故其调用时的实参 x 值相差为 7，实参 y 值相差为 0。

以上方法为将数字分解为单个数字后一个一个写的方法来实现的。用这种方法只须定义 10 个数字的写函数。如果采用直接定义数字的方法，则二位数须 100 种函数定义，三位数须 1000 种函数定义。

以此类推，可以说用直接定义数字的方法在目前是不可能实现的。

3. AutoCAD 的 PDB 技术中的 action_tile 函数可以定义对话框中某个控件被选中后的操作。我们可以将获取用户输入的数据与向图像控件中画该数据定义在同一个操作表达式

中。这样，图像控件即可立即显示用户输入的数字。例如，将长度输入编辑框的操作表达式作如下定义：

```
(action_tile ″otherlenght″ ″(draw_numlen (setq L(atoi $value)))″)
```

式中“otherlenght”为dcl文件中长度编辑框的关键字；“draw_numlen setq L atoi $value ”定义了一个向图像控件写数字的操作；setq L atoi $value 将用户输入的数字变为整数后赋给变量L。

变量L有两个作用：1）作为函数draw_numlen的实参；2）保存用户的输入以便画螺钉的子程序调用。

下面给出画螺钉的实用程序，以说明本方法的运用。

第一步，定义单个数字的写函数，这在前面已进行了较详细的说明。

第二步，定义尺寸数字的写函数 draw_num numbe 本函数的功能：

1）将用户的输入转换成单个的数字并分别保存在gw 个位）、sw 十位）、bw 百位）和qw 千位）变量中，本转换是用取子字符串的方法实现的。

2）确定清屏及各数字在图像控件中的座标，如螺钉长度的清屏及在图像控件中的座标是如下定义的：

```
(if( = flen 1)(setq fix1 117 fiy1 77 fx0 119 fx1 126 fx2 133 fx3 140 fy 77))
```

式中flen为螺钉长度的标识码；fix1和fiy1为清屏的起始座标；fx0、fx1、fx2和fx3分别为个、十、百和千位数字的x座标；fy为个、十、百和千位数字的y座标。

3）确定清屏的终点座标fix2和fiy2。其对应语句为 setq fix2 30 fiy2 17 。需要指出的是：由于图像控件中的座标是相对座标，所以在各清屏语句中fix2和fiy2不变。

4）清屏 其对应语句为 fill_image fix1 fiy1 fix2 fiy2 - 2 ，式中的 - 2为图像控件底色码；

5）调用单个数字的写函数向图像控件画尺寸。本功能是用一个重复函数来实现的。本函数的第一个语句 setq num＊ read strcat “num” chr k znum read chr k 的作用为：⑴将单个数字的写函数填上实参后赋给变量num＊；⑵给变量znum赋值。注意：单个数字的写函数的实参与znum的值相等。重复函数表达式中变量k的初值为48，此值为数字“0”的ascii值。由函数chr将数字“0”的ascii值转换为数字“0”。本函数的第二个语句 if = qw znum eval num＊ fx0 fy 的作用是：当用户输入的千位数字即qw与znum相等时，num函数便立即向x座标为fx0 y座标为fy处画出znum所表示的数字。下面的语句分别为画百位、十位和个位数字的语句。执行重复函数，将使单个数字的写函数从0到9遍历一遍。使每种数字都可写向图像控件。

6 在图像控件中产生图像，其对应语句为：

```
(setq width (dimx_tile ″ld_tx″) height (dimy_tile ″ld_tx″))
(slide_image 0 0 width height″screw1″)
```

本语句将在图像控件中产生图像。注意，⑴产生图像的幻灯片“screw1”中的各尺寸位置应与程序中的各尺寸位置协调。⑵幻灯片应在AutoCAD的库搜索路径上。 3 为了杂志编辑部演示的需要，程序中将显示幻灯的语句作了无效处理。

第三步，定义各尺寸函数。例如定义画螺钉长度函数，其对应语句为：

```
(defun draw_numlen (numn)(setq fdia 0 flen 1 )(draw_num numn))
```

式中draw_numlen为函数名，本函数由对话框驱动程序调用。numn为形参，由用户的输入即实参来取代。

setq函数将螺钉长度标识码设为可选，其余为不可选以免重复。语句 draw_num numn 用来调用尺寸数字的写函数。

第四步，定义画螺钉的对话框驱动程序。本程序中与画尺寸有关的部分前面已介绍得很详细。为了突出重点，本程序中要用到的画螺钉子程序 hld 在附录中没有提供。实际上程序员利用类似的输入界面可开发出多种应用程序。因此画图子程序可以是各种类型的子程序，如画螺母、齿轮以及画各种专用图形的子程序。

三、使用方法

画螺钉的对话框驱动程序定义了一个AutoCAD命令ddld。将该函数装入AutoCAD后只要用户在命令行键入ddld后回车，即会在屏幕上出现要求用户输入螺钉参数的两个列表框、一个编辑框和一个图像控件 由画螺钉的对话框定义文件定义的）。因为PDB技术中没有提供象Visual Basic语言或Visual C++语言环境中的编辑框键盘事件处理功能，所以用户采用编辑框输入后，只有在输入焦点移到下一个控件时才能在图像控件中反映出输入的数据。如果输入是用列表框或弹出式列表框，则只要在列表框或弹出式列表框选取一个数据，该数据会在图像控件中立即反映出来，这已与Visual Basic语言或Visual C++语言编写程序的相应功能相同。

本程序可在AutoCADR14或AutoCAD 2000环境下装入后直接运行。为提高安全性和运行速度，也可在AutoCAD 2000环境下编译成vlx或ARX文件后再运行。

该功能已应用于笔者的各种标准件 如各种螺栓、螺钉、钻套、齿轮、弹簧等 的参数化绘图和专用自动化设计软件的数据录入界面中。

四、存在问题及开发前景

1. 本方法目前仅能水平书写整数，如要垂直书写应另定义写函数。如要写小数则要另行开发。

2. 附录中仅用到了千位数字 用户可仿此编写出写任意位数数字的程序。

3. 如果图像控件采用图像按钮，则可用选择点在该控件中返回的座标值来实现直接从图像控件中选取欲编辑的数据。用这种方式，可只用一个编辑框实现编辑图像控件所有尺寸的功能。通常当图像控件中的数据个数超过5个，数据名称又难以用

文字描述时，最好采用直接从图像控件中选取欲编辑数据的方式。

```
;;;;;;;;;;;;;;;;;程序清单;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;以下为 Autolisp 语言程序;;;;;;;;;;;;;;;;;;;;
;;;;定义尺寸数字的写函数;;;;
(defun draw_num (numbe / k znum numb fix1 fix2 fiy1 fiy2
fx0 fx1 fx2 fx3 fy qw bw sw gw)
;;;;取长度数字的个(gw),十(sw),百(bw)和千(qw)位数字
(setq gw "" sw "" bw "" qw"")              ;初始化中间变量
      (setq numb(itoa numbe))
      (cond((= (strlen numb) 1)(setq gw(atoi numb)))
           ((= (strlen numb) 2)(setq gw(atoi (substr numb 2
1)) sw(atoi (substr numb 1 1))))
           ((= (strlen numb) 3)(setq gw(atoi (substr numb 3
1)) sw(atoi (substr numb 2 1))
                 bw(atoi (substr numb 1 1))))
           ((= (strlen numb) 4)(setq gw(atoi (substr numb 4
1)) sw(atoi (substr numb 3 1))
bw(atoi (substr numb 2 1)) qw(atoi (substr numb 1 1))))
      )
      (start_image "ld_tx")
             ;;;确定清屏及各数字在图像按钮中的座标
      (if(= fdia 1)(setq fix1 238 fiy1 135 fx0 224 fx1 231
fx2 238 fx3 245 fy 135))
      (if(= flen 1)(setq fix1 117 fiy1 77 fx0 119 fx1 126 fx2
133 fx3 140 fy 77))
      (setq fix2 30 fiy2 17)          ;由于图像按钮中的座标
是相对座标,fix2 和 fiy2 不变
      (fill_image fix1 fiy1 fix2 fiy2 -2)   ;清屏
             ;;;;调数目字函数
      (setq k 48)
      (repeat 10
(setq num * (read(strcat "num" (chr k))) znum(read(chr
k)))
        (if(= qw znum) ((eval num *) fx0 fy))
        (if(= bw znum) ((eval num *) fx1 fy))
        (if(= sw znum) ((eval num *) fx2 fy))
        (if(= gw znum) ((eval num *) fx3 fy))
        (setq k(+ k 1))
      )
     (setq width (dimx_tile "ld_tx") height (dimy_tile "ld_tx"))
; (slide_image 0 0 width height"screw1")
;显示幻灯片"screw1"
      (end_image)
)
;;;;以下定义画各长度函数(由画螺钉对话框驱动程序调用)
            ;;;;各长度函数要调用尺寸数字的写函数
(defun draw_numdia (numn)(setq fdia 1 flen 0 )(draw_num
numn))
(defun draw_numlen (numn)(setq fdia 0 flen 1 )(draw_num
numn))
            ;;;;;画螺钉的对话框驱动程序
(defun C:ddld(/ dcl_id)
  (setq dcl_id (load_dialog "c:/hld.dcl"))  ;装入对话框,文
件在 c 盘根目录下
  (if (not (new_dialog "dld" dcl_id)) (exit))
  (action_tile "cancel" "(done_dialog)(exit)");定义退出对话
框操作
  (action_tile "accept" "(done_dialog)") ;定义确定按钮操作
  (action_tile "otherlenght" "(draw_numlen (setq l (atoi
$value)))");定义自定长度编辑框操作
  (draw_numlen 50)
  (draw_numdia 10)
  (action_tile "ld_zj" "(draw_numdia
             (setq d (atoi (nth (atoi $value) dialist ))))")
;定义选择直径操作
  (action_tile "ld_cd" "(draw_numlen
             (setq l (atoi (nth (atoi $value) lenglist ))))")
;定义选择长度操作
            ;装入长度列表
  (setq lenglist (list "5" "6" "8" "10" "12" "14" "16" "18"
 "20" "22" "25" "28" "30" "32" "35" "38" "40" "45" "50"
"55""60""65" "70" "75" "80" "85" "90" "95" "100" "110"
"120" "130" "140" "150" "160" "170" "180" "190" "200"
"210" "220" "230" "240" "250" "260" "280" "300"))
  (start_list "ld_cd" 3)
  (mapcar 'add_list lenglist)
  (end_list)
            ;装入直径列表
(setq dialist (list "4" "5" "6" "8" "10" "12" "14" "16" "18"
    "20" "22" "24" "27" "30" "36" "42" "48"))
  (start_list "ld_zj" 3)
  (mapcar 'add_list dialist)
  (end_list)
  (start_dialog)
  (if (= d nil)(= d 10))             ;赋直径缺省值
  (if (= l nil)(= d 50))             ;赋长度缺省值
  (hld)                              ;调用画螺钉的子程序
)
            ;;;;定义单个数目字的写函数;;;;
(defun num0 (a b)
       (vector_image (+ a 4) (+ b 0) (+ a 0) (+ b 0) 0)
       (vector_image (+ a 0) (+ b 0) (+ a 0) (+ b 9) 0)
       (vector_image (+ a 0) (+ b 9) (+ a 4) (+ b 9) 0)
     (vector_image (+ a 4) (+ b 9) (+ a 4) (+ b 0) 0))
(defun num1 (a b)
    (vector_image (+ a 3) (+ b 0) (+ a 3) (+ b 9) 0))
(defun num2 (a b)
       (vector_image (+ a 0) (+ b 1) (+ a 0) (+ b 0) 0)
       (vector_image (+ a 0) (+ b 0) (+ a 4) (+ b 0) 0)
       (vector_image (+ a 4) (+ b 0) (+ a 4) (+ b 3) 0)
       (vector_image (+ a 4) (+ b 3) (+ a 0) (+ b 6) 0)
       (vector_image (+ a 0) (+ b 6) (+ a 0) (+ b 9) 0)
     (vector_image (+ a 0) (+ b 9) (+ a 4) (+ b 9) 0))
(defun num3 (a b)
       (vector_image (+ a 0) (+ b 1) (+ a 0) (+ b 0) 0)
       (vector_image (+ a 0) (+ b 0) (+ a 4) (+ b 0) 0)
       (vector_image (+ a 4) (+ b 0) (+ a 4) (+ b 9) 0)
       (vector_image (+ a 4) (+ b 4) (+ a 1) (+ b 4) 0)
       (vector_image (+ a 4) (+ b 6) (+ a 4) (+ b 9) 0)
       (vector_image (+ a 4) (+ b 9) (+ a 0) (+ b 9) 0)
     (vector_image (+ a 0) (+ b 9) (+ a 0) (+ b 8) 0))
(defun num4 (a b)
```

```
        (vector_image (+ a 3) (+ b 0) (+ a 0) (+ b 7) 0)
        (vector_image (+ a 0) (+ b 7) (+ a 4) (+ b 7) 0)
    (vector_image (+ a 3) (+ b 0) (+ a 3) (+ b 9) 0))
(defun num5 (a b)
        (vector_image (+ a 4) (+ b 0) (+ a 0) (+ b 0) 0)
        (vector_image (+ a 0) (+ b 0) (+ a 0) (+ b 4) 0)
        (vector_image (+ a 0) (+ b 4) (+ a 4) (+ b 4) 0)
        (vector_image (+ a 4) (+ b 4) (+ a 4) (+ b 9) 0)
    (vector_image (+ a 4) (+ b 9) (+ a 0) (+ b 9) 0))
(defun num6 (a b)
        (vector_image (+ a 4) (+ b 1) (+ a 4) (+ b 0) 0)
        (vector_image (+ a 4) (+ b 0) (+ a 0) (+ b 0) 0)
        (vector_image (+ a 0) (+ b 0) (+ a 0) (+ b 9) 0)
        (vector_image (+ a 0) (+ b 4) (+ a 4) (+ b 4) 0)
        (vector_image (+ a 4) (+ b 4) (+ a 4) (+ b 9) 0)
    (vector_image (+ a 4) (+ b 9) (+ a 0) (+ b 9) 0))
(defun num7 (a b)
        (vector_image (+ a 0) (+ b 1) (+ a 0) (+ b 0) 0)
        (vector_image (+ a 0) (+ b 0) (+ a 4) (+ b 0) 0)
    (vector_image (+ a 4) (+ b 0) (+ a 2) (+ b 9) 0))
(defun num8 (a b)
        (vector_image (+ a 0) (+ b 4) (+ a 4) (+ b 4) 0)
        (vector_image (+ a 4) (+ b 0) (+ a 0) (+ b 0) 0)
        (vector_image (+ a 0) (+ b 0) (+ a 0) (+ b 9) 0)
        (vector_image (+ a 0) (+ b 9) (+ a 4) (+ b 9) 0)
    (vector_image (+ a 4) (+ b 9) (+ a 4) (+ b 0) 0))
(defun num9 (a b)
        (vector_image (+ a 0) (+ b 5) (+ a 4) (+ b 5) 0)
        (vector_image (+ a 0) (+ b 0) (+ a 0) (+ b 5) 0)
        (vector_image (+ a 4) (+ b 0) (+ a 0) (+ b 0) 0)
        (vector_image (+ a 0) (+ b 8) (+ a 0) (+ b 9) 0)
        (vector_image (+ a 0) (+ b 9) (+ a 4) (+ b 9) 0)
    (vector_image (+ a 4) (+ b 9) (+ a 4) (+ b 0) 0))
;;;;;;;;;;;Autolisp 语言程序结束;;;;;;;;;;;;;;;;;;;;
      /* 以下为对话框定义文件 */
     /* 画螺钉对话框定义文件 */
dld : dialog {
   label  =″选择螺钉参数″;
   initial_focus = ″ld_zj″;
   : row{
   : boxed_column{
      label  = ″直径″;
        : list_box{
           key  = ″ld_zj″;
           value = 10;
           allow_accept = true;
                 }
               }
   : boxed_column{
      label  = ″长度″;
        : list_box{
           key  = ″ld_cd″;
           value = 50;
           allow_accept = true;
                 }
        : edit_box{
           label  = ″其它″;
           key  = ″otherlenght″;
           allow_accept = true;
                 }
               }
       : image_button{
          key  = ″ld_tx″;
          allow_accept = true;
          width = 50;
          aspect_ratio = 0.8;
          color = 7;
                 }
          }
       : row{
          fixed_width = true;
          alignment = centered;
          : ok_button{
             key = ″accept″;
             label = ″确定″;
             is_default = true;
                 }
          : spacer{width = 2;}
          : cancel_button{
             key = ″cancel″;
             label = ″取消″;
                 }
          }
   }
          /* 对话框定义文件结束 */
```

演示本程序的方法

1. 本程序应当在 AutoCADR14 环境下运行。

2. 先在 AutoCAD 的 support 子目录下任选一个 Autolisp 文件和一个对话框定义文件即 DCL 文件拷贝到 C 盘根目录下。

3. 将 Autolisp 文件原来内容清空后将上面的 Autolisp 文件内容粘贴到文件中 将文件名改为 yswj。

4. 将 DCL 文件原来内容清空后将上面的 DCL 文件内容粘贴到文件中 将文件名改为 hld。

5. 启动 AutoCADR14 用 “AP” 命令将 Autolisp 文件 yswj 装入内存。

6. 在命令行键入 “ldld” 后回车即可运行本程序。

7. 程序启动后 只要在对话框中选择一个直径或长度 图像控件中的数字即可立即反映用户的选择。

8. 注意! 因为考虑到杂志编辑部绘制及保存和调用幻灯片可能有困难 故去掉了显示螺钉幻灯片的功能。

参考文献

1. AutoCADR14 中文版 二次开发技术. 郭朝勇等编著 清华大学出版社 1999 年 4 月第一版

2. VisualLISP R14 - 2000 编程与应用. 孙江宏等编著 科学出版社 1999 年 10 月第一版

收稿日期:2000 年 8 月 14 日)

应用 Java 语言进行 AutoCAD2000 二次开发

刘良华　袁英战　朱东海

摘　要　当前，采用 C/C++、VB 进行 AutoCAD 二次开发已经很普遍。本文另辟蹊径，利用 AutoCAD2000 提供的类型库，根据 COM 技术原理编写了一个例程。通过该例程，详细介绍了应用 Java 语言进行 AutoCAD 二次开发的方法和思路。

关键字　二次开发，AutoCAD 定制，COM，Java 编程

Microsoft 的组件对象模型（COM　Component Object Model）允许基于 Windows 平台的应用程序之间可以建立联系并进行数据交换。采用组件对象模型有一个很大的好处就是，开发者并不需要了解服务程序（server application）的内部机制，只要利用其提供的接口（interface）就可以实现对服务程序内部的访问和操作。同样，AutoCAD2000 提供了它自己的 COM 接口，因此客户程序（client application）可以通过该接口来对 AutoCAD2000 进行操作，如打开 DWG 图形、添加图形对象等。

目前，支持 COM 接口的开发语言越来越多，如 C/C++、Java 等。原则上，不管采用何种开发语言，其实现原理都是相同的。实际上，采用 Java 语言来进行 AutoCAD 的二次开发是非常有意义的，因为这样可以将 AutoCAD 的二次开发应用于 Java 小应用程序（applet）上。为简单计，本文仍以普通 Win32 应用程序为例，该例程使用 Visual J++6.0 开发工具，详细展示了 Java 语言在 AutoCAD2000 二次开发中的应用方法和技巧。

一、AutoCAD2000 的类型库

类型库（type library）是一种 OLE 复合文档，通常以 .tlb 为后缀（也有其它形式的后缀，如 .odl、.ocx 等）。类型库文件包含了 OLE 服务程序提供的接口类型及 COM 对象的有关信息。具体来说，包含的内容如下：

1. 数据类型的信息，如别名（alias）、枚举、结构（struct）或联合（union）等；

2. 模块、接口、IDispatch 接口或组件对象类的描述信息，每个描述信息称为一个类型信息记录。

实际上，类型库文件提供了访问被提供的 COM 组件的细节。只有了解这些细节，应用程序才能访问其提供的 COM 组件。AutoCAD2000 提供了 COM 组件及其相应的类型库文件，文件名为 acad.tlb，存放于安装根目录下（如 D:\ACAD2000\）。

以 AutoCAD2000 为例（下同），在 Java 源文件中利用 COM 对象的方法如下：

```
import acad.*        //引入 AutoCAD2000 提供的类型库信息
IAcadApplication IApp = (IAcadApplication) new AcadApplication();
```

其中，AcadApplication 是由 acad.tlb 提供的信息生成的 Java 类，它定义了 AutoCAD2000 应用程序（COM 类）。与 C++类似，生成一个 Java 类实例可以使用 new 操作符，但不能直接使用生成的类实例，而必须通过其 COM 接口来实现。因此，下述代码尽管是合法的，但在运行过程中将导致严重错误。

```
AcadApplication App = new AcadApplication();  //错误 **
```

二、建立工程

首先，从“开始|程序”菜单中启动 Visual J++6.0，进入开发环境（缺省），如图 1 所示。

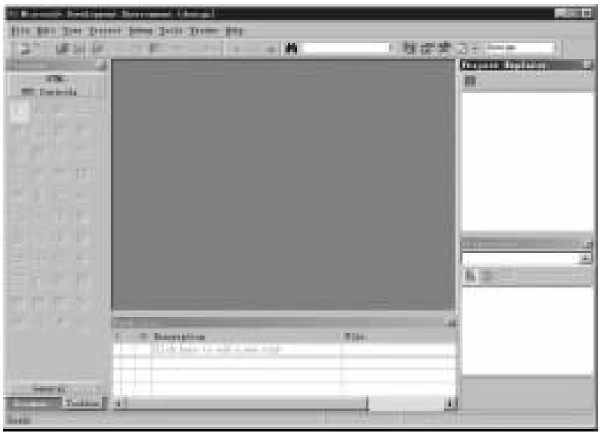


图 1　Visual J++6.0 主程序窗口

Visual J++6.0 主程序窗口除包含有菜单栏、工具栏等标准窗口元素外，还有工程资源管理器（Project Explorer）、工具箱（Toolbox）、属性窗口（Properties）以及任务列表框（Task List）等。其中，工程资源管理器的作用类似于 Visual C++6.0 的工作区窗口（Workspace），而工具箱和属性窗口则是为编辑窗体（form）等控件资源而提供的。

按照下面的步骤，新建一个工程。如图 2 所示，选择 Visual J++工程类型为 Application；然后，在右边的框中选择“Windows Application”，即新建一个 Win32 应用程序；接下来，在 Name 文本框中输入工程名称为 JCOMToACAD，并选择

适当的存放路径。最后，单击“打开”按钮。

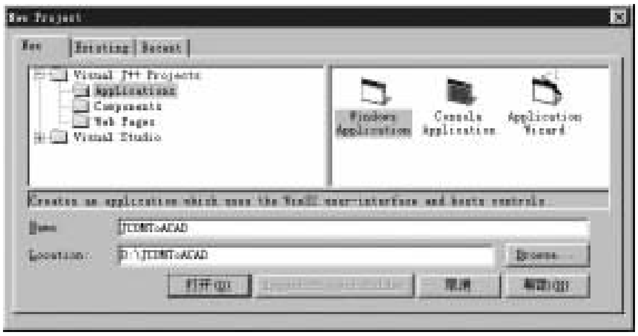


图 2　新建一个“Windows 应用程序”工程

开发环境为当前新建的工程添加了一个源程序文件 Form1. java。双击该文件，弹出一个窗体（即 Form1），它是应用程序运行的主窗口。现在，将文件 Form1. java 更名为 mainForm. java，方法是在工程资源管理器中选中该文件，然后执行“File | Rename”菜单命令，在随之出现的文本框中键入新文件名。

接下来，在工程资源管理器中双击 mainForm. java，这时弹出的是该文件的源代码窗口，而不是窗体编辑窗口。使用“Edit | Find and replace”菜单命令，将 mainForm. java 文件中的字符串 Form1 全部替换为 mainForm，然后关闭该文件的源代码窗口。之后，当再次双击 mainForm. java 文件时，首先弹出的就是 mainForm 窗体了，如图 3 所示。

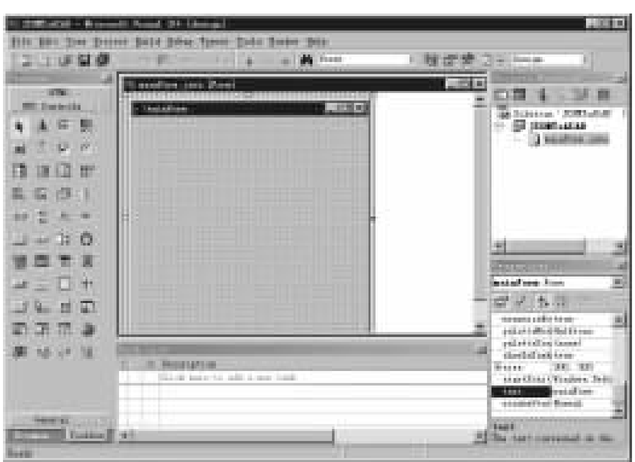


图 3　编辑窗体资源

然后编辑菜单资源

向图 3 所示的程序窗体添加一个菜单栏。为此，在工具箱中选择“MainMenu”工具，方法是在工具箱中单击该工具按钮，然后在 mainForm 窗体上部单击鼠标左键并向右拖动，这时一个菜单栏就添加好了。默认的菜单栏名称为 mainMenu1，可以在属性窗口中将其更改为新的名称（如 mainMenu），如图 4 所示。注意：编辑其它控件元素属性的方法与此类似。

接下来，向菜单栏中添加两个子菜单：File（文件）和 Draw（制图）。File 子菜单命令如图 5 所示，其各控件元素的属性见表 1。注意：添加子菜单以及菜单项的方法是，鼠标左键单击“Type here”区域，然后在随之出现的文本框中输入所需的标题即可。

图 4　编辑控件元素的属性

图 5　新添加的 File 子菜单

表 1　File 子菜单及其菜单项的属性

子菜单及菜单项标题	text 属性	name 属性（即标识符）	修改的其它属性
File（子菜单标题）	& File	ID_MENU_FILE	
Start AutoCAD	& Start AutoCAD	ID_FILE_STARTACAD	
Quit AutoCAD	& Quit AutoCAD	ID_FILE_QUITACAD	无
分隔符（separator）	-（连字符）	无	
Exit	& Exit	ID_FILE_EXIT	

添加的 Draw 子菜单提供了两个命令，如图 6 所示，其各控件元素的属性见表 2。

图 6　新添加的 Draw 子菜单

表 2　Draw 子菜单及其菜单项的属性

子菜单及菜单项标题	text 属性	name 属性（即标识符）	修改的其它属性
Draw（子菜单标题）	& Draw	ID_MENU_DRAW	
Highlight Line	& Highlight Line	ID_DRAW_LINE	无
Red Circle	& Red Circle	ID_DRAW_CIRCLE	

三、输入 AutoCAD 类型库

如需在工程中使用 AutoCAD 提供的 COM 对象，就必须向工程项目输入类型库（tlb）。选择“Project | Add COM Wrapper”菜单命令，弹出的对话框如图 7 所示，该对话框列出了可以利用的 COM 组件。如果没有列出来，那么可以单击“Browse”按钮，浏览和查找所需的 COM 组件。

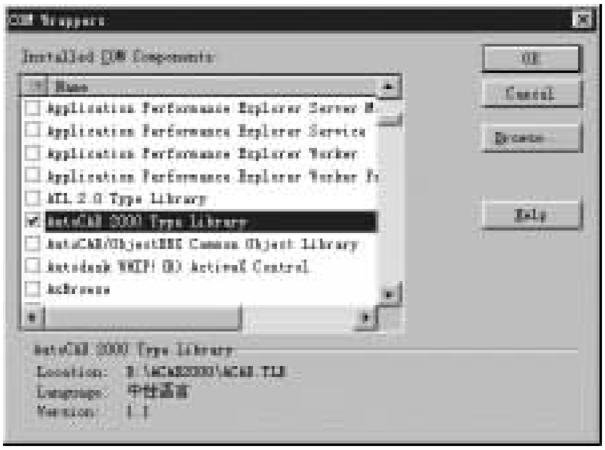


图 7 选择输入 AutoCAD2000 类型库

在列表框中选择“AutoCAD 2000 Type Library”，然后单击 OK 按钮。Visual J++6.0 在当前工程项目所在的目录中新建一子目录（hacad），其中包含了 AutoCAD2000 类型库提供的 COM 组件。注意：在存放工程的目录下不能含有 hacad 子目录 即使是空目录也不行），因为 Visual J++ 将认为当前工程已经输入了 AutoCAD2000 类型库而不再进行任何操作。

接下来，打开 mainForm.java 文件，添加如下代码 黑体部分，下同）。预编译指令 import 与 C/C++ 语言中的#include，两者的作用是类似的。

```
import com.ms.wfc.app.*;
import com.ms.wfc.core.*;
import com.ms.wfc.ui.*;
import com.ms.wfc.html.*;
import acad.*;   //引入 AutoCAD 2000 提供的类型库信息
import com.ms.com.*;
……
```

这样，就可以在 mainForm.java 源文件中使用 AutoCAD2000 类型库提供的组件了。

四、添加实现代码

编辑完菜单资源后，接下来添加程序代码，以实现各菜单命令。在控件资源编辑窗口中 如图 5、图 6 所示），双击菜单项或右击菜单项然后在弹出的菜单中选择“View Code”命令。此时，光标通常位于该控件的响应函数体内。

首先，在 mainForm 类定义的开始处定义几个 COM 对象接口指针，通过它们能够访问 AutoCAD2000。这几个变量将在菜单命令的实现中使用。

打开 mainForm.java 的代码窗口，并添加如下代码：

```
public class mainForm extends Form
{
   IAcadApplication IApp;
   IAcadDocument IDoc;
   IAcadModelSpace IMSpace;
   IAcadUtility IUtil;
   boolean boolAcadStarted = false; //指示 AutoCAD 2000 是否启动
   public mainForm()
   {
      initForm();
   }
   ……
}
```

其中，接口 IAcadApplication 用于初始化和获得 AutoCAD2000 的一个实例，只有通过它才能访问其它的 COM 接口。IAcadDocument 和 IAcadModelSpace 用于向 AutoCAD 的图形数据库中添加实体对象 如果是图形实体，同时将显示在 AutoCAD 的图形窗口中）。

IAcadUtility 用于用户输入输出 如坐标点、关键字、距离的输入）以及格式化等操作。

同样，添加菜单命令‘File|Start AutoCAD” 指的是例程应用程序的菜单命令，下同）响应函数的实现代码如下：

```
private void ID_FILE_STARTACAD_click(Object source, Event e)
{
if(boolAcadStarted == true)//如有一备分已初始化,则返回
      return;
   IApp = (IAcadApplication) new AcadApplication();
   if(IApp == null)
   {
      MessageBox.show("初始 AutoCAD 2000 失败!");
      return;
   }
   IApp.setVisible(true);
   IDoc = (IAcadDocument) IApp.getActiveDocument();
   IUtil = (IAcadUtility) IDoc.getUtility();
   IMSpace = (IAcadModelSpace) IDoc.getModelSpace();
   boolAcadStarted = true;
}
```

上述代码首先启动并初始化 AutoCAD 应用程序，获得 IAcadApplication 指针；然后通过其成员函数 setVisible 将 AutoCAD 置为可见。接下来，依次获得当前的活动文档接口及由此获得其它接口，即 IAcadUtility 和 IAcadModelSpace。

菜单命令‘File|Quit AutoCAD”的作用是使先前启动的 AutoCAD 退出，其响应函数的实现代码如下：

```
private void ID_FILE_QUITACAD_click(Object source, Event e)
{
   if(boolAcadStarted == true)
   {
      IApp.Quit();      //退出 AutoCAD2000
      boolAcadStarted = false;
   }
}
```

菜单命令‘File|Exit”的作用是退出当前的应用程序，其响应函数的实现代码如下：

```
private void ID_FILE_EXIT_click(Object source, Event e)
{
   if(boolAcadStarted == true)
   {
      IApp.Quit();      //退出 AutoCAD2000
      boolAcadStarted = false;
   }
```

```
    Application.exit();          //应用程序退出
}
```

接下来，定义 Draw 子菜单命令的实现函数。命令“Draw|Hightlight Line”的作用是在 AutoCAD 视图窗口中画一黄色直线，并高亮度显示，其响应函数的实现代码如下：

```
private void ID_DRAW_LINE_click(Object source, Event e)
{
    if(boolAcadStarted == false)    //如果 AutoCAD 未被成功初始化,则返回
        return;
    int hr = 0;
    Variant opt = new Variant();
    opt.VariantClear();
    opt.noParam();          //该参数值可选
    Variant pt1 = new Variant();
    pt1.VariantClear();
    Variant pt2 = new Variant();
    pt2.VariantClear();
    Variant prompt = new Variant();
    try
    {
        prompt.putString("第一点:");          //设置提示信息
        IUtil.InitializeUserInput(1, opt);  //初始用户输入函数
        pt1 = IUtil.GetPoint(opt, prompt);     //提示用户输入第一个点
    }
    catch(ComException except)
    {
        hr = except.getHResult();
    }
    if(hr == 0)
    {
        try
        {
            prompt.putString("第二点:");       //设置提示信息
            IUtil.InitializeUserInput(1, opt);//初始用户输入函数
pt2 = IUtil.GetPoint(pt1, prompt);//提示用户输入第二个点
        }
        catch(ComException except)
        {
            hr = except.getHResult();
        }
    }
    if(hr == 0)
    {
IAcadLine ILine = (IAcadLine) IMSpace.AddLine(pt1, pt2);
        ILine.setColor(2);          //设置黄颜色
        ILine.Highlight(true);      //设置高亮度显示
    }
}
```

上述代码首先检查 AutoCAD 是否已成功初始化，如否则退出。函数代码提示用户输入两个坐标点，并以此两点画一直线。

如前所述，IAcadUtility 接口提供了用户输入输出的成员函数。IAcadUtility.GetPoint 函数提示用户输入一个坐标点，它的功能同 ADS 函数 acedGetPoint 。其中，第一个参数是可选的，参数类型为 Variant。如需设置可选，那么则应该调用 Variant.noParam 函数。

```
Variant optional = new Variant();      //定义新变量
opt.VariantClear();                    //清空
opt.noParam();                         //置为可选参数
```

类似的，菜单命令“Draw|Red Circle”的作用是在 AutoCAD 视图窗口中画一红色的圆，其响应函数的实现代码如下：

```
private void ID_DRAW_CIRCLE_click(Object source, Event e)
{
    if(boolAcadStarted == false)   //如果 AutoCAD 未被成功初始化,则返回
        return;
    int hr = 0;
    Variant opt = new Variant();
    opt.VariantClear();
    opt.noParam();
    Variant centerPt = new Variant();
    centerPt.VariantClear();
    double radus = 0.0;
    Variant prompt = new Variant();
    try
    {
        prompt.putString("中心点:");          //设置提示信息
        IUtil.InitializeUserInput(1, opt);  //初始用户输入函数
        centerPt = IUtil.GetPoint(opt, prompt);  //提示用户输入中心点
    }
    catch(ComException except)
    {
        hr = except.getHResult();
    }
    if(hr == 0)
    {
        try
        {
            prompt.putString("半径:");        //设置提示信息
        IUtil.InitializeUserInput(3, opt);  //初始用户输入函数
            radus = IUtil.GetDistance(centerPt, prompt);
//提示用户输入半径
        }
        catch(ComException except)
        {
            hr = except.getHResult();
        }
    }
    if(hr == 0)
    {
        IAcadCircle ICirc = (IAcadCircle) IMSpace.AddCircle
(centerPt, radus);
        ICirc.setColor(1);          //设置红颜色
    }
}
```

五、编辑工程并运行

在 Visual J++6.0 中选择“Project|JCOMToACAD Properties”命令，设置工程参数。如图 8 所示，确保应用程序运行时首先加载的是 mainForm 窗体 其它保持默认值）。

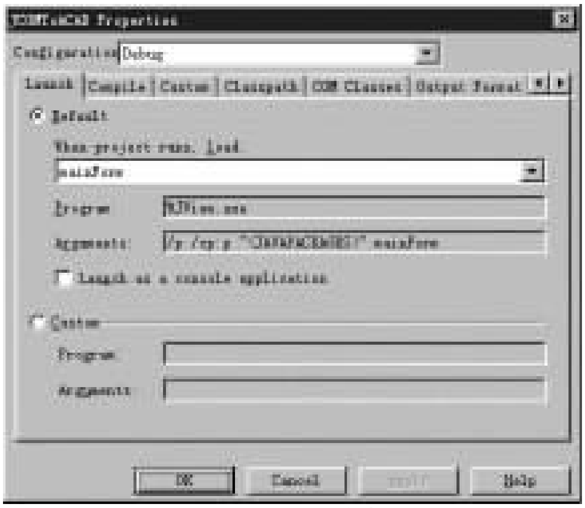


图 8　设置工程参数

选择“Build | Rebuild”菜单命令，编译整个工程。如果前述各步骤操作无误的话，那么整个编译过程将不会出现任何错误或警告信息，最后得到 JCOMToACAD. exe 文件。

在 Windows 平台下运行 JCOMToACAD. exe。如图 9 所示，这是运行应用程序的主窗口，它的窗口标题为 mainForm，并包含了一个菜单栏。

图 9　应用程序运行时的主窗口

在图 9 中，选择“File | Start AutoCAD”菜单命令，应用程序将启动 AutoCAD2000。接下来，选择“Draw | Highlight Line”菜单命令，切换到 AutoCAD2000 窗口中指定两点，程序将在这两点之间画一直线。并且，在指定第一点后拖动鼠标的过程中，在第一点和光标的当前位置之间显示一条橡皮线。

同理，选择“Draw | Red Circle”菜单命令，将可以在 AutoCAD2000 中画一红色的圆。最后的结果包括一条高亮度显示的直线和一红色的圆，如图 10 所示。

六、结束语

上述例程的目的只是在于介绍应用 Java 语言进行 AutoCAD2000 二次开发的方法和技巧，其中假定读者已经对 Java 语言以及 ActiveX 编程技术有所了解。实际上，在获得服务程序的实例后，就可以调用其提供的接口来访问其提供的 COM

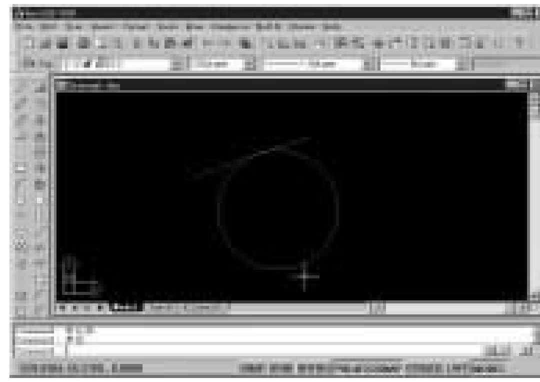


图 10　添加的直线和圆

对象了。所涉及的编程方法与其它普通应用编程几乎没有差别。

同理，在 Visual Basic 环境中同样可以使用 AutoCAD2000 提供的 COM 对象，即 AutoCAD 的 VBA 二次开发。

参考资料

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

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

（收稿日期：2000 年 10 月 11 日）

教育软件——知识到效率的转化

洪恩公司透露：《开天辟地 II》的升级产品《开天辟地 III》正在紧锣密鼓的开发中，预计春节前后就将上市。除了继承《开天辟地 II》的通俗易懂、实用简练等优点以外，更在学习方法、设计理念等方面上了一个新台阶。

《开天辟地 III》首先在内容上将跟随市场进行全新的内容更新，保证您得到的知识都是目前主流的东西。还是比较注重对初级用户的照顾，从最基础的部分键盘、鼠标讲起，步步深入，直到程序设计入门篇、知识篇，内容上补充了 Office2000、Liunx、WinMe，更新了 Internet 部分、图像处理、排版 丰富了程序设计入门知识。提供了更新的知识篇，增加趣味的、包含一定故事情节的训练游戏。继续秉承洪恩软件一贯人性化的设计理念，《开天辟地 III》在设计上还注重了交互性，这一点将比《开天辟地 II》更为先进；学习者可以在学习的过程中加入更多的练习，这样不但可以使学习者迅速掌握知识，更重要的是可以在学习中加以实践，迅速实现了从学习到应用的转化，根本上达到了教育软件的最终目的。《开天辟地 III》将知识点融会贯通到整个教学过程中，并使之按照一定规律进行重复，使学习者能通过反复学习将知识牢牢记住。除了全程语音之外，更增加了创新性的主动提示方式，向导性和主动性，使整个《开天辟地 III》在学习方式上更高出目前市场上的其他学习软件一筹。所谓的主动性，就是指到了某一个知识点该提示学习者的时候，无需用户提出请求就可以自动弹出对这个知识点的提示，非常容易使用户加深对所学知识的印象。

软件防拷贝注册机制的程序实现

王国忠

一、设计思路

软件推广与软件的防盗版往往是矛盾的：推销软件时，极力推荐试用自己的产品，同时，为了保护自己的知识产权，又要努力防止产品不被非法复制。于是出现了共享软件的注册机制：未注册时，是演示版，注册并付款后，用户才获得软件的正式版。在用户注册后，许多公司的做法是向客户邮寄或电邮正式版软件，用户收到后必须重新安装软件。

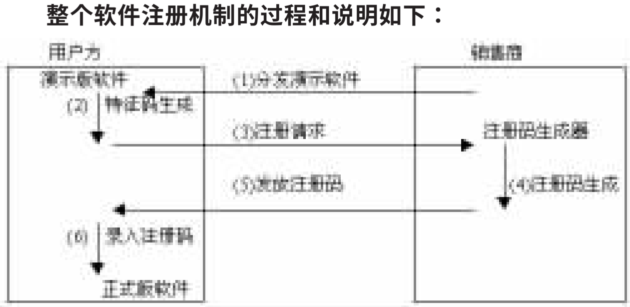
为了兼顾上述两方面，笔者的思路是：软件公司向客户提供的注册码必须结合用户所在系统的特征信息的代码，这样，即使软件被复制，由于新环境的特征与原有环境特征不同，软件内部设计好的特征判定程序会拒绝系统执行正版软件的一些功能，而只执行演示版程序的功能。

关键是客户特征如何取得并保证其唯一性。

较成功的做法是：采用用户计算机硬盘的产品序列号或网卡的 MAC 地址，因为这些代码重复的可能性极小，特别是后者，是全世界唯一的。用户在注册时，演示版程序的“注册”功能自动提取该特征信息，然后向软件销售商发送该信息，销售商获得该特征码后，用注册码生成器按一定算法对客户特征码进行变换，最后生成注册码，发送给用户，用户再用注册功能输入收到的注册码，软件自动升级为正式版。

在这种方法中，用户特征码到用户注册码的转换算法必须合理，保证转换之后的注册码也保持唯一性，同时，由于软件中也必须具有根据用户特征生成注册码并与用户从销售商处获得的注册码比较的功能，因此，转换算法必须有一定的复杂性以防止软件的注册机制被非法破解。

整个软件注册机制的过程和说明如下：



(1) 软件销售商向用户分发演示软件。

(2) 用户运行演示程序、用软件注册功能生成特征码。

(3) 用户将特征码通过 Email 等形式反馈给销售商，请求注册。

(4) 销售商用注册码生成器，根据用户反馈的特征码生成注册码。

(5) 销售商向用户通过 Email 等形式发放注册码。

(6) 用户在演示程序中输入获得的注册码，用注册功能录入注册码，演示程序根据注册码生成规则生成内部注册码，再与用户录入的注册码比较，若相等，则软件自升级为正式版，否则，继续以演示软件形式运行。

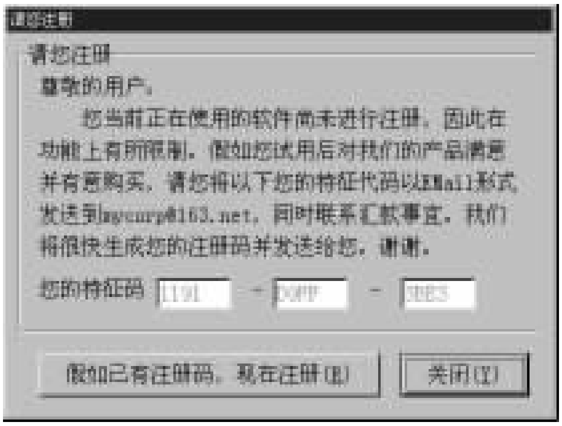
二、客户特征码的获取

在 Delphi 中，获取用户计算机的网卡 MAC 地址可通过 Windows 中 NetBIOS 的 API 调用来完成，得到地址后，作适当变换再作为用户特征码。实现这一过程的程序如下：

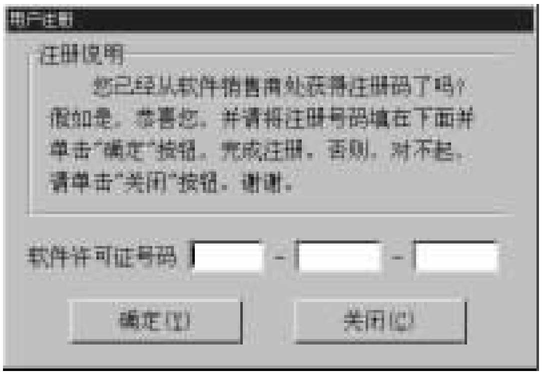
```
unit mac;
interface
uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs, NB30,  StdCtrls;
type
 TNBLanaResources  = (lrAlloc, lrFree);
type
 PMACAddress  = ^ TMACAddress;
 TMACAddress  = array[0..5] of Byte;
type
  TForm1  = class(TForm)
    Label1: TLabel;
    Edit1: TEdit;
    Button1: TButton;
    Button2: TButton;
    function GetMACAddress(LanaNum: Byte; MACAddress: PMACAddress): Byte;
    procedure Button1Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
var
  Form1: TForm1;
implementation
{$R *.DFM}
//GetMACAddress - 获取网卡 MAC 地址的函数
//其中: LanaNum 取值为 0 - n, 指哪一个网卡
//      MACAddress 为获取的 MAC 地址值
function TForm1.GetMACAddress(LanaNum: Byte; MACAddress: PMACAddress): Byte;
var
  AdapterStatus: PAdapterStatus;       {网卡状态变量}
```

```
  StatNCB: PNCB;                    {网络控制块变量}
begin
  New(StatNCB);
  ZeroMemory(StatNCB, SizeOf(TNCB));          {分配网络控制块空间}
  StatNCB.ncb_length := SizeOf(TAdapterStatus) + 255 * SizeOf(TNameBuffer);
  GetMem(AdapterStatus, StatNCB.ncb_length);
  try
    with StatNCB^ do
    begin
      ZeroMemory(MACAddress, SizeOf(TMACAddress));
      ncb_buffer := PChar(AdapterStatus);
      ncb_callname := '*               ' + #0;
      ncb_lana_num := Char(LanaNum);
      ncb_command := Char(NCBASTAT);
      NetBios(StatNCB);                   {调用 NetBIOS 功能}
      Result := Byte(ncb_cmd_cplt);
      if Result = NRC_GOODRET then{获取地址,存放到变量}
        MoveMemory(MACAddress, AdapterStatus, SizeOf(TMACAddress));
    end;
  finally
    FreeMem(AdapterStatus);               {释放空间}
    Dispose(StatNCB);
  end;
end;
procedure TForm1.Button1Click(Sender: TObject);
var
  LanaNum: Byte;
  MACAddress: PMACAddress;
  RetCode: Byte;
begin
  LanaNum := 0;
  New(MACAddress);
  try
    RetCode := GetMACAddress(LanaNum, MACAddress);
{调用函数,获取 MAC 地址}
    if RetCode = NRC_GOODRET then
    begin
      Edit1.Text := Format('%2.2x%2.2x%2.2x%2.2x%2.2x%2.2x', [MACAddress[0]+11, MACAddress[1]+12,
MACAddress[2]+13,  {作适当变换}
        MACAddress[3]+14, MACAddress[4]+15, MACAddress[5]]+16);
    end else
    begin
      Beep;
      Edit1.Text := 'Error';
      ShowMessage('GetMACAddress Error! RetCode = $' + IntToHex(RetCode, 2));
    end;
  finally
    Dispose(MACAddress);
  end;
end;
procedure TForm1.Button2Click(Sender: TObject);
begin
   form1.close;
end;
end.
```

将上述程序嵌入所发布的软件并略加修改，便可以向试用软件的用户提供注册的界面以及启动时的注册码校验。下图是注册功能的两个界面的示例：



特征码生成



完成注册

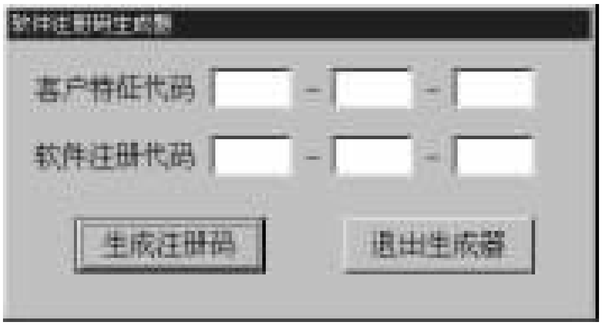
三、注册码生成

用户向销售商反馈特征码并提出注册请求之后，软件销售商用注册码生成器生成用户的注册码，其过程仅仅是对用户提交的特征码进行一系列的变换。注册码生成器的源程序如下，有编程经验的朋友可以自行改变特征码到注册码的变换过程。

```
unit main;
interface
uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  StdCtrls;
type
  TForm1 = class(TForm)
    Label1: TLabel;
    Label3: TLabel;
```

```
    Button1: TButton;
    Button2: TButton;
    Edit1: TEdit;
    Label2: TLabel;
    Edit2: TEdit;
    Label4: TLabel;
    Edit3: TEdit;
    Edit4: TEdit;
    Label5: TLabel;
    Edit5: TEdit;
    Label6: TLabel;
    Edit6: TEdit;
    procedure Button2Click(Sender: TObject);
    procedure Button1Click(Sender: TObject);
    function GenRegCode(source: string): string;
  private
    { Private declarations }
  public
    { Public declarations }
  end;
var
  Form1: TForm1;
implementation
{$R *.DFM}
procedure TForm1.Button2Click(Sender: TObject);
begin
  form1.close;
end;
procedure TForm1.Button1Click(Sender: TObject);
var
regcode: string;
begin
    if (length(trim(edit1.text))<>4) or (length(trim(edit2.text))<>4)
         or (length(trim(edit3.text))<>4) then
     begin
       application.MessageBox('客户提交的特征码无效,必须为3个4位!', '错误信息', mb_ok);
       exit;
     end;
   regcode: =genregcode(trim(edit1.text)+trim(edit2.text)+trim(edit3.text));
  edit4.text: =copy(regcode, 1, 4);
  edit5.text: =copy(regcode, 5, 4);
  edit6.text: =copy(regcode, 9, 4);
end;
function TForm1.GenRegCode(source: string): string;
var
cusstr, mystr: array [0..12] of char;        {密钥}
i: integer;
ii, j, k: byte;
begin
  strpcopy(cusstr, source);
  mystr[0]: ='T';
  mystr[1]: ='h';
  mystr[2]: ='i';
  mystr[3]: ='s';
  mystr[4]: ='a';
  mystr[5]: ='T';
  mystr[6]: ='e';                          {密钥}
  mystr[7]: ='s';
  mystr[8]: ='t';
  mystr[9]: ='1';
  mystr[10]: ='2';
  mystr[11]: ='3';
  k: =0;
  for i: =1 to 12 do
    k: =k xor ord(cusstr[i-1]);     {特征异或和}
  result: ='';
  for i: =1 to 12 do
    begin
      j: =ord(mystr[i-1]) xor ord(cusstr[12-i]); {每字节与密钥字节异或}
      j: =j xor k;            {再与累加和异或}
      j: =abs(j);             {取其绝对值}
      if j>127 then
        j: =j-128;            {限制在0-127之间}
      ii: =i;
      if j<=33 then
        j: =j+33+ii;          {限制在空格-127之间}
      result: =result+char(j);  {得到注册码}
    end;
end;
end.
```

该注册码生成器的软件界面如下：



注册码生成器

四、软件自升级

用户在输入了正确的注册码之后，软件应由原来的演示版自动升级为正式版。设计程序的朋友都知道，这很简单：将用户输入的注册码保存在文件中，如<.INI>文件，每次软件启动时读入该代码，同时，读取系统的特征码并按注册器相同的规则生成内部注册码，再与用户输入的注册码比较，若相同，则应开放系统所有功能　即自升级为正式版），否则，应按演示软件的特点屏蔽一些功能或加上时间限定。这部分的程序由于比较简单，在此不再列出。

收稿日期：2000年8月24日）

电脑硬盘系统使用与维护常见问题解答

操作系统分配给驱动器盘符原则是什么

操作系统分配给驱动器盘符规定如下：A：、B：固定分配给软盘驱动器使用，而不管机器上是否有软盘驱动器存在；C：分配给活动分区，以后按顺序分配给逻辑分区，最后分配给光盘驱动器。

如一个 6.4GB 的硬盘，如果使用 FAT32 文件系统，只建立一个基本分区，不建扩展分区，则硬盘的驱动器号就是 C：，光驱的驱动器号是 D：；也可以建立一个基本分区，一个扩展分区，各为 3GB，则基本分区的驱动器号为 C：，扩展分区的驱动器号为 D：，光驱的驱动器号为 E：。

如果使用 FAT16 文件系统，由于其支持的最大分区是 2GB，因此可以将硬盘划分为一个基本分区 2GB，一个扩展分区 4.4GB。扩展分区又划分为三个逻辑分区，分别为 2GB、1.2GB、1.2GB，启动操作系统后，基本分区将被分配为 C：，三个逻辑驱动器依次为 D：、E：、F：，光驱为 G：。

安装新硬盘要做哪些工作

硬盘安装分为硬件安装和软件安装两步，前者是固定硬盘和连线等，后者包括对硬盘分区和格式化。

（1）硬件安装

1） 准备工作

准备十字螺丝刀一把，将身上的静电放掉（用手摸一下机壳）。

2） 认识接线端口

在硬盘的一个侧面有三个端口，其中有四个铜柱的接电源线，有两排共 40 个引脚的接数据线，剩下的那个端口一般位于中间，是设置跳线用的。

3） 跳线的设置

硬盘在出厂时一般都将其默认设置为主盘，跳线连接在“Master”的一侧。如果计算机上已经有了一个做主盘的硬盘，现在要连接一个做从盘的硬盘，那么要将跳线接在“Slave”的一侧。但这种设置方法并不通用，有的硬盘在做主盘时要连接两个跳线，做从盘时不连接跳线，这要视不同的硬盘而定，可以在硬盘的某一面上找到有关跳线的具体设置方法，按照该方法去正确设置跳线。

4） 连线

硬盘的连线分电源线和数据线两种连接，两者谁先连接并不重要。对于电源线的连接，注意电源端口上的小缺口，在电源接头上也有相似缺口，这样便于正确插入。仔细一点，千万别将接头的方向弄反了（有些劣质电源的接头在反向时也可以插入硬盘的电源端口）。硬盘数据线是 40 线的，与软驱数据线有点相像，但比软驱线的 34 线宽一些，也分方向，有一端有红色标记，一般将红色的一端插入硬盘数据线插槽上标有“1”的一端，另一端插入主板 IDE 口上也标有“1”的那端。数据线插反不要紧，如果开机硬盘不转的话（听不到硬盘自举的响声），多半插反了，将其旋转 180 度后插入即可。

5） 固定

连好线后就可以用螺丝将硬盘固定在机箱上，注意有接线端口的那一个侧面向里，另一头朝向机箱面板。

（2）设置硬盘参数

安装了硬盘之后，要在 CMOS 中设置硬盘参数，包括容量、簇数、扇面等。可参照硬盘说明（一般在硬盘的背面有标示）在相应的地方填入，也可以进入 CMOS 让机器自动设置。目前微机上用得较多的 BIOS 有 AWARD 和 AMI 两家公司的产品，其中又以 AWARD 公司的最为流行，我们就以 AWARD 的 CMOS 设置为例给大家介绍其具体操作方法：

选择“IDE HDD AUTO DETECTIONG”项，系统将自动查找所有连上的可识别的硬盘，找到后将显示其识别的参数，并询问你是否正确，你选择了是（“Yes”）或否（“No”）后，系统将寻找下一个硬盘。查找顺序为：第一主硬盘——第一从硬盘——第二主硬盘——第二从硬盘。查找完毕之后，将系统配置信息自动写入 CMOS 设置中。一般情况下，这样的设置都是正确的，但如果开机无法进入系统，可考虑手动输入。

（3）低级格式化

低级格式化（简称“低格”）是对一块盲盘划分磁道和扇区、标注地址信息、设置交叉因子等操作，这个过程进行得相对很慢。由于低格很伤硬盘，所以用户轻易不要对硬盘进行这种操作。新硬盘出厂时就已经进行过低级格式化，并且低格的过程比较简单。

（4）分区

在一个新硬盘安装好后，接下来要做的是对硬盘进行分区。常见的分区软件有很多，最易上手且最易找到的是 DOS 下自带的 Fdisk.exe。下面就介绍怎样用该程序进行分区。

1） 准备工作

准备一张能启动的 3.5 英寸盘，除了 Fdisk.exe 外，还要有格式化程序 Format.com。

2） 分区操作

分区的操作是按照如下顺序进行的：

建立基本 DOS 分区；

建立扩展 DOS 分区；

在扩展分区中建立逻辑 DOS 分区。

如果在建立分区前硬盘上已经有分区，需要将原分区删除后再重新划分分区。为了便于描述分区操作，我们假设将一个 4.3G 的硬盘分为 3 个区：C 盘 1G，D 盘 1.8G，E 盘 1.5G。这里基本 DOS 分区指的是 C 盘、D 盘和 E 盘都属于扩展 DOS 分区中的逻辑分区。

3）运行 Fdisk 程序

从 A 盘启动计算机，运行 Fdisk 程序，出现 FDISK 的主菜单，分区工作只要用到前面的前两个选项即可。先按 1　或直接回车）进入建立分区的子菜单。

3）　建立基本 DOS 分区

在主菜单上选择 1 后，屏幕上会提示一段话，问你是否将全部的硬盘都分为一个基本分区，需要注意的是 FAT16 分区只能管理到 2GB 大小的硬盘，如果只分一个区，则只能得到 2GB 的容量　而不是 4.3GB）。键入“N”，系统提示输入基本 DOS 分区大小，键入“1000”即可。这时系统自动将该分区的盘符设为 C，并在屏幕上显示 C 区的信息。基本分区建好后，按 Esc 回到 FDISK 主菜单。

4）　将基本 DOS 分区激活

系统认为必须有一个分区是活动分区　即开机后能从该分区上找到启动系统的信息），一般是将基本 DOS 分区　(C 区）设为活动分区。在主菜单中按“2”进入活动分区设置，再按“1”，这时屏幕显示已经设置好的活动分区信息，与“4”略有不同的是在“Status”项目下的 C 区栏多了一个“A”，意思是 C 区为活动　(Active）分区。按 Esc 回到 FDISK 主菜单。

5）　建立扩展 DOS 分区

在 FDISK 主菜单中直接回车　即选择 1），进入建立分区菜单；按“2”建立扩展 DOS 分区。系统认为硬盘上只能有一个基本 DOS 分区和一个扩展 DOS 分区，这样剩下的容量将全部是扩展 DOS 分区。直接按回车键，进入逻辑 DOS 分区的设置。

6）　建立逻辑 DOS 分区

在 5）的基础上，屏幕会显示一些信息，询问如何建立逻辑分区。如果只建立一个逻辑分区则直接回车　在本例中将得到 2GB 容量的 D 区，而不是 3.3GB），否则键入 D 区的容量大小：1800，然后回车。这时屏幕显示在 C 区下多了一个 D 区。按照与建立 D 区相同的步骤建立 E 区。全部分区建立好后，系统将显示全部硬盘分区信息。

7）按 Esc 退出 FDISK 程序，重新启动机器。

(5）格式化

这里的格式化指的是高级格式化。分区建立好后，在 A 盘符下键入：format c　/s 对 C 盘进行高格。屏幕显示已经完成的百分比直到 100% 完成。对 D、E 盘也要进行高级格式化操作。

格式化完成后，这块硬盘就可以使用了，可以在上面进行系统软件的安装等工作。

(6）分区的删除

如果要对一块已经分过区的硬盘进行重新分区，必须先将原有的分区信息删除。删除分区也是在 A 盘中运行 FDISK 程序，进入主菜单后，键入“3”，进入删除分区子菜单。删除分区的顺序是：

删除非 DOS 分区；

删除逻辑 DOS 分区；

删除扩展 DOS 分区；

删除基本 DOS 分区。

(7）关于 FAT32

上面介绍的分区和格式化是基于 FAT16 分区方法进行的，在多年前就一直沿用这个方法。这种分区的方法对不同大小的区规定了每个簇的容量，系统在存取数据时以簇为单位，即便是 1 个字节的信息也要占用一个簇，一个 33KB 的文件在不同的区上占用的空间不等，对硬盘容量的消费也不等，如表 1 所示。可以知道，分区越大　簇的容量越大），文件越小，文件个数越多，浪费的空间也越多。

表 1　分区大小与簇容量等的对应关系

分区大小 MB	簇的容量 (KB)	33KB 文件占用簇数	浪费空间 (%）
0 - 16	4	9	8.3
16 - 128	2	17	2.9
128 - 256	4	9	8.3
256 - 512	8	5	17.5
512 - 1024	32	2	48.4

现在的硬盘越来越大，用 FAT16 分区方法在分区大于 512MB 时将会带来很大浪费，在小于 512MB 时又必须分很多区，很难记住软件存放在什么盘中。在这种情况下，微软公司在 Windows 95 OSR2　俗称 Windows 97）以上版本中加入了对 FAT32 分区方法的支持，这种分区方法支持大于 2GB 的分区，更重要的是它规定了分区小于 256MB 时簇容量为 0.5KB；分区在 256MB 到 8GB 时簇的容量为 4KB；8GB 到 16GB 时簇容量为 8KB，这样即使将一个 4.3GB 的硬盘全部分为一个区，也不会造成很大浪费。

但是 FAT32 也有缺点，首先它不支持有些基于 FAT16 的工具软件，如常用的 PCTOOLS 等。其次是不能在 FAT32 分区的硬盘上使用压缩程序，也不能使用低版本的操作系统。在 Windows 95/98 的安全模式和 MS-DOS 状态下硬盘的速度将变得慢，虽然可以通过加载 Smartdrv 磁盘缓冲程序来解决，但是占用了系统资源。究竟使用哪一种分区方法要看该计算机主要从事什么工作来决定。

若想使用 FAT32 分区，要用 Windows 98 自带的 FDISK 来启用系统文件，当然也可以用系统自带的‘驱动转换器’对硬盘进行文件的直接转换，还可以用相关工具软件 PQMAGIC 等进行。