电脑编程技巧与维护月刊

2001 年第 5 期　　总第 83 期　1994 年 7 月创刊）　　每月 3 日出版

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

新技术追踪

编程与应用起步

编程语言

专家论坛

可视化专栏

数据库

网络技术

图形图像处理

计算机维护

计算机安全

博士信箱

‘编程疑难问题解答”征稿启事

应广大读者的要求，为加强读者间的交流，及时解答电脑编程人员在应用开发编程实践中遇到的疑难问题，本刊准备新辟‘编程疑难问题解答”栏目，现开始向广大读者征稿，来稿要求：

1. 编程实践中遇到的典型问题；
2. 所提问题明确，带有普遍性；
3. 解答条理清晰，程序源代码要求调试通过；
4. 可以只提出问题，也可以又提出问题又进行解答；
5. 内容要求短小、精练。

《电脑编程技巧与维护》杂志社

Brocade 开发速度更快的数据传输交换机

发展最快的数据存储部件供应商之一 Brocade Communications Systems Inc. 正在开发一种高性能交换机，其速度比目前的产品快一倍，而价格基本相同。Brocade 近来一直向分析师介绍这个名为 SilkWorm 12000 的产品，并计划在今天的一个行业研讨会上推出其原型。这台交换机每秒可传输 2 千兆字节 Gigabyte 的数据，价格在 30 万 - 70 万美元之间。公司表示，这种新的网络交换机能够在互相不兼容的网络间传递数据，并能把数据传入产自不同生产商的存储器中。

QAD 公司展示最新电子商务解决方案

QAD 公司于 4 月 2 日 - 6 日在昆明召开了一年一度的 QAD 亚太联盟伙伴 TEAMWORK 大会和中国用户大会，并在会中展示了 QAD eQ SV 和 QAD Storefront 等一系列创新的电子商务解决方案。QAD 是全球著名的软件应用系统供应商，为全球制造商提供优秀的电子商务方案。QAD 公司是少数几个能针对行业需要，提供完整的端到端解决方案的供应商之一。大会还着重介绍了 QAD 专有的解决方案空间 solution space 技术，能帮助不同行业的用户，根据行业需要定制最适

合自己的电子商务解决方案，并能帮助企业制定明确的策略，向电子商务顺利迈进。

Linux - Mandrake 8.0 Beta 3 推出

MandrakeSoft 公司日前推出了代号为“Traktopel”、最新版本的 8.0 Beta3 Linux - Mandrake 发行版本。这个新版本包含了 Linux 2.4.2 内核，KDE 2.1.1 以及最新的 GNOME 和 Nautilus 鹦鹉螺 。新版本使用了最新的 KDE 2.1.1 和 2.0 版本相比有了很多的改进，对 2.1 也作了很多的 Bug 修补。新版本的缺省内核就是 Linux 2.4.2 了，因此你可以得到更加多的 USB 支持，更好的 SMP 对称多处理器 支持，新的 IP 和防火墙堆栈。

IA - 64 Linux 内核支持英特尔微处理器

最新版本的 IA - 64 Linux 内核已经发布，该内核增加了对英特尔公司生产的 McKinley 微处理器的支持。McKinley 是 Itanium 的后续者，预计将在今年第四季度提供给客户试用。除了支持 McKinley 之外，最新版本的 IA - 64 Linux 内核还有其他一些变化，包括和 2.4.3 版本的常规 IA - 32 Linux 内核同步等。

用鼠标也可输入汉字 每分钟可录 30 字

日前在兰州易博文软件公司获悉，一种名为 “易博汉字鼠标输入法”的软件已于今年 3 月获得国家信息产业部颁发的知识产权证书，并在近日投入批量生产。据了解， “易博汉字鼠标输入法”是国内迄今为止首次利用鼠标输入汉字的软件技术，特别适合非专业人士和网络初学者使用。最多四下，就可以完成汉字输入的简易方法。此种方法是先取汉字声母 + 该字首笔画+ 该字末笔画，再利用重码及联想功能，从选取汉字窗口中选出所需的词，因此具有简单易学、现学现用的特点。据悉，该输入法最高输入速度每分钟可达 30 字，完全可以为普通人上网提供工具性支持。

Power Paper 推出纸一样薄的电池

位于以色列的 Power Paper 公司发明了超薄、无外壳、低成本，甚至像纸一样柔韧具有灵活性的多种电池。电池仅有 0.5mm 厚，而且可以根据要求制作成任何尺寸和形状。Power Paper 电池是制作一次性微电子应用领域的理想材料。它使用锌和二氧化锰做阴极，而采用专利技术的油墨状层做阳极。阳极材料可做成薄片状，这样可印刷、粘贴以附着在任何基层上 比如纸张 。一个一英寸见方的 Power Paper 印刷电池可以提供 1.5V 15mAh 的电力。这种电池不会造成环境污染。

Macromedia 发布 FreeHand 10

Macromedia 公司日前发布了最新版本的 FreeHand 10。FreeHand 10 不仅和 Macromedia 的系列网页制作工具结合得更加紧密，而且还增加了对苹果电脑公司最新 OS X 操作系统的支持。FreeHand 10 可以帮助所有的用户都具有制作高质量图表的能力，为此，该产品增加了好几种制作工具。FreeHand 10 最重要的特色之一是采用了 Macromedia 的标准用户界面。现在，该公司软件产品中的所有工具看起来都非常相似，而且位于同一个区域。

开拓金融市场——Sybase 推出 FFS 解决方案

国际著名的电子商务解决方案供应商 Sybase 公司日前宣布，其子公司 Financial Fusion 推出了针对金融行业而设计的最新版本电子商务系统解决方案 FFS Financial Fusion Server 。该解决方案通过对金融行业内外部系统的无缝连接，简化了电子零售业务等方面的处理，为用户提供功能更加强大的零售电子银行和电子交易服务。FFS Financial Fusion Server 的推出，不但为电子金融领域提供了全面的业务支持，同时也将满足证券交易中机构性活动的需求，为 Sybase 成功开拓证券交易行业创造了有利条件。

Portal 软件公司推出 Infranet 6.1

日前，北京——Portal 软件公司近日宣布推出其市场领先的 Infranet 产品的最新版本 Infranet 6.1。新增功能旨在满足各类通信服务提供商的特殊要求，包括高性能的事件处理、先进的开发工具、改进的品牌性能、支持无线设备和无线因特网服务的用户自主维护等。Infranet 6.1 强化了 Infranet 平台的价值和威力。通过与现有庞大用户群的紧密合作，Infranet 6.1 进一步优化了强大的性能，使用户得以迅速定义、部署和管理业务。

VC + +系列讲座（五）

张博强　徐　亮

第五讲　使用 ODBC 访问数据库

ODBC 是 Open Database Connector（开放数据库互连）的简称，它最初是由 Microsoft 公司提出，是作为一个标准的、基于 SQL 的接口来实现的。在它接口的背后其实是一些针对各种不同的数据库来设计的 plug - in 插件，每种插件都是在 ODBC 函数和某种特定的数据库接口之间进行转换。使程序员用一种接口可以访问所有的数据库。在 VC + + 的 MFC 类库中集成了许多 ODBC 类，其中最重要的是 CDatabase、CRecordset 和 CRecordView。

下面我们就来创建一个带有 ODBC 支持的 SDI（单文档界面）程序。

首先，我们需要一个数据库，我使用的是 Microsoft Access 2000，这主要是考虑到 Microsoft Office 已经被普遍的使用，当然你可以使用其它的数据库。只要在数据库中建立一个名为“学生”的表，包括：学号、姓名、性别、出生年月、系/专业、班级及备注七个字段，然后使用 ODBC 管理器为该数据库配置一个 ODBC 数据源，数据源名（DSN）为“Mydb”。

接下来我们创建一个标准 SDI 风格的数据库应用程序。

首先，打开 VC + +，使用 MFC AppWizard exe 向导创建一个新的工程，命名为 Students。

然后，在 AppWizard 向导的第一步选择“Signle Document”选项，其它选项保持默认情况。在第二步中选择“Database view with file support”，并单击“Data Source”按钮，从 ODBC 数据源中选中“Mydb”，Snapshot 选项告诉 VC + + 一次下载全部整个查询，Dynaset 则使 VC + + 为每个记录创建一个实际的指针，只有实际需要时才从服务器上下载记录。选择后单击 OK 按钮，出现对话框，从其中选择表“学生”后单击 OK 按钮返回。随后的几步均保持缺省设置。在最后一步中我们可以看到 AppWizard 为我们创建了一个 CStudentsSet 类，它继承于 CRecordset 类。

完成应用程序外壳的创建，删除没有用的控件，然后，可以先运行程序看看结果。

接下来我们将使用静态文本和编辑框控件设计程序的主窗口，它将用来显示数据库中的七个字段，属性设置如下：

Static Text　Caption 学号
Edit Box　ID　IDC_EDIT_CODE
Static Text　Caption 姓名
Edit Box　ID　IDC_EIDT_NAME
Static Text　Caption 性别
Edit Box　ID　IDC_EDIT_GENDER
Static Text　Caption 出生年月
Edit Box　ID　IDC_EDIT_BIRTHDAY
Static Text　Caption 系/专业
Edit Box　ID　IDC_EIDT_DEPARTMENT
Static Text　Caption 班级：
Edit Box　ID　IDC_EIDT_CLASS
Static Text　Caption 备注：
Edit Box　ID　IDC_EIDT_REMARK

图 1　设计程序的主窗口

完成以上设计后，要把它们与数据库中的相应字段联系起来。

单击鼠标右键，从中打开 ClassWizard，并选择 Member Variable 项，确定 Class name 为 CStudentsSet。这时可以看到下面的列表中每个 Column Name 都绑定了一个变量，但这些变量的命名没有规律，我们将删除它们重新添加变量如下：

[学号]	long	m_Code
[姓名]	CString	m_Name
[性别]	CString	m_Gender
[出生年月]	CTime	m_Birthday
[系/专业]	CString	m_Department
[班级]	CString	m_Class
[备注]	CString	m_Remark

将数据库中字段与变量绑定之后，我们还需要把变量与空控件联系起来。

从当前 Member variable name 标签对话框的 Class name 下拉列表框中选择 CStudentsView 类，这时可以看到各个编辑框的控件 ID，点击 Add Variable 按钮打开对话框。从 Member varable name 的上拉列表中，你会发现我们已经加入的变量，只要选择对应于控件的变量就可以了。不过，其中也有一个

例外，那就是你无论如何也没有找到 m_pSet - >m_Birthday。这是由于在 CStudentsSet 类中，与出生年月字段绑定的变量的类型被置为 CTime，而 VC 中没有一个宏或函数可以用来处理对话框控件与 CTime 变量之间的数据交换。所以我们要对此进行一些修改。点击 OK 按钮，关闭 ClassWizard，然后在 ClassView 中扩展 Students classes 和 CStudentsSet 双击 m_Birthday 项，找到其在程序中的位置，将 CTime 类型改为 ColeDatTime 变量类型。再打开 ClassWizard 的 Member variable name 标签，会发现里面的内容已经被更正了，并且也已经可以将变量与对话框中的控件联系起来了。不过要实现变量类型的转换，我们要增加一些代码来实现此功能。

首先，我们为 IDC_EDIT_BIRTHDAY 重新定义一个变量，它的类型为 COleDateTime，变量名为 m_olebirthday。从 ClassView 扩展 CStudentsView，双击 DoDataExchange 找到该函数在程序中的位置，修改后程序如下：

```
void CStudentsView: : DoDataExchange(CDataExchange * pDX)
{
    CRecordView: : DoDataExchange(pDX);
    //下面是加入的代码
    if(pDX –>m_bSaveAndValidate = =FALSE)
        m_olebirthday = m_pSet –>m_birthday;
    //上面是加入的代码
    //{{AFX_DATA_MAP(CStudentsView)
    DDX_FieldText(pDX, IDC_EDIT_CLASS, m_pSet –>m_class, m_pSet);
    DDX_FieldText(pDX, IDC_EDIT_CODE, m_pSet –>m_code, m_pSet);
    DDX_FieldText(pDX, IDC_EDIT_DEPARTNENT, m_pSet –>m_department, m_pSet);
    DDX_FieldText(pDX, IDC_EDIT_GENDER, m_pSet –>m_gender, m_pSet);
    DDX_FieldText(pDX, IDC_EDIT_NAME, m_pSet –>m_name, m_pSet);
    DDX_FieldText(pDX, IDC_EDIT_REMARK, m_pSet –>m_remark, m_pSet);
    DDX_Text(pDX, IDC_EDIT_BIRTHDAY, m_olebirthday);
    //}}AFX_DATA_MAP
    //下面是加入的代码
    if (pDX –>m_bSaveAndValidate = =TRUE)
        m_pSet –>m_birthday = m_olebirthday;
    //上面是加入的代码
}
```

其中，m_bSaveAndValidate 是对话框数据交换（DDX）的方向指示标志，它为非 0 时，CdataExchange 对象移动数据从对话框控件到对话框类数据成员，为 0 时，则使用对话框类数据成员初始化对话框控件。重新编译运行，我们将得到一个基本的数据库应用程序，你可以检索到数据库中的记录，并允许滚动和修改记录。

我们在浏览数据的时候，经常会增加一条记录或删除某条记录，下面我们就来实现增加、删除的功能。

图 2 在 Member Varialbe 对话框中添加变量

加入新的数据记录

首先，我们要在菜单中加入选项，以便执行增加记录的功能。在 SDI 中创建菜单的方法与在 MDI 中创建菜单的方法类似。

打开 ResourceView，用菜单资源编辑器，在“记录”菜单中增加“添加记录”项，其 ID 值为“IDM_ADD”；通过工具条资源编辑器增加一个按钮，作为添加记录的按钮，打开属性对话框，从其 ID 值的下拉菜单中选择 IDM_ADD，将两者联系起来。

做完可视设计，我们来为菜单和按钮编码。打开 Class-Wizard，选择类名为 CstudentsView，Object ID 为 IDM_ADD 然后为它增加 COMMAND 事件的消息处理函数。编辑代码如下：

```
void CStudentsView: : OnAdd()
{
  CRecordset * pSet = OnGetRecordset();
//当前所在记录是否保存
if (pSet –>CanUpdate() & & !pSet –> IsDeleted ())
{
  pSet –>Edit();
  if (!UpdateData())
      return;
  pSet –>Update();
}
//当前所在记录是否保存
m_pSet –>MoveLast();
m_pSet –>AddNew();
m_pSet –>m_remark = "无";
m_pSet –>Update();
m_pSet –>Requery();
m_pSet –>MoveLast();
UpdateData(FALSE);
  }
```

在这段代码的开始，先建立了一个 CrcordsetSet 类指针，接着用 CanUpdate 和 IsDelete 判断一下当前的记录是否被更新或删除，如果被修改则将修改结果保存。否则，就移动到记录最后，添加一条新记录，在更新之后重新检索数据。

删除一条记录

要删除数据库中的一条记录，要使用 Delete 函数。不过我们首先还是要为它创建菜单项和工具条按钮，创建的方

法与前面一样。

从 ResourceView 打开菜单资源编辑器，增加“删除记录”的菜单项，其 ID 值为“IDM_DEL”；使用工具条资源编辑器添加一个与之对应的删除记录按钮。

下一步工作，打开 ClassWizard，选择 CstudentsView 类，为 IDM_DEL 增加一个 COMMAND 事件的消息处理函数 OnDel。代码如下：

```
void CStudentsView: : OnDel()
{
m_pSet –>Delete();
m_pSet –>MovePrev();
if(m_pSet –>IsBOF())
m_pSet –>MoveLast();
if(m_pSet –>IsEOF())
m_pSet –>SetFieldNull(NULL);
UpdateData(FALSE);
}
```

在用 Delete 函数删除当前记录之后，用 MovePrev 函数将指针移动到前一条记录，并通过 IsBOF 函数判断记录的位置，如果从第一条记录再向前移，IsBOF 将返回非 0 值。与此相似，通过 IsEOF 函数的返回值可以判断记录集是否为空，当记录集由最后一条记录向后滚动或当记录集为空时，IsEOF 将返回非 0 值。如果当前记录集为空，则用 SetFieldNull 函数标记出来，最后对窗体进行更新，完成了一个删除操作。不过在删除之前，我们通常会进行确认，以免执行误操作。所以我们在 OnDel 函数中再加 if 语句进行判断，修改后代码如下：

```
void CStudentsView: : OnDel()
{
if (MessageBox(″你确定要删除当前的记录吗?″, ″注意!″,
MB_YESNO|MB_ICONQUESTION) = =IDYES)
{
m_pSet –>Delete();
m_pSet –>MovePrev();
if(m_pSet –>IsBOF())
m_pSet –>MoveLast();
if(m_pSet –>IsEOF())
m_pSet –>SetFieldNull(NULL);
UpdateData(FALSE);
}
}
```

这样我们已经实现了对数据库记录的添加与删除操作，不过此时的程序在功能上还不完善，如缺乏对数据的有效性检验，而且你还可以增加对记录的查询与过滤功能，这就留给读者作为进一步的练习吧。

小结

本讲的主要内容是使用 ODBC 对数据库的一些基本操作，读者通过学习应掌握：

1. 用 AppWizard 创建带有数据库支持的 SDI 工程的方法
2. 建立变量与数据库中数据的联系
3. 在变量间进行日期型数据的交换
4. 了解 CRecordset 类，掌握对数据库的检索、添加、删除等基本操作，掌握一些基本成员函数的使用，如 AddNew、Delete、Update 等。

图 3 程序运行窗口

参考资料

1. 王晖等编著．精通 Visual C + + 6. 0. 电子工业出版社
2. 刘文智编著．Visual C + + 6. 0 教程
3. Chunk Wood 编著．Visual C + + 6. 0 数据库编程大全

收稿日期：2001 年 1 月 17 日）

建设中美商贸桥梁，金仕达携手 ChinaOnline

近日，接受美国著名商务公司 ChinaOnline 的委托，金仕达多媒体全力开发电子商务网站 ChinaAD。

ChinaOnline 是一个专门传播有关中国商业信息的英文网站，以中国为信息源，广泛地从中国众多的内容提供机构获得已公开发表的翔实可靠的商业信息，通过国际互联网提供给美国及世界工商业各界。2000 年，ChinaOnline 于金仕达多媒体公司结成战略伙伴，在业务方面进行全面的合作。

ChinaAD 隶属 ChinaOnline 旗下，提供中国广告运作、媒体宣传、品牌建立等方面的专业信息，是 ChinaOnline 中国战略的重要组成部分。金仕达多媒体将利用自己在电子商务网站开发方面的优势，为 ChinaAD 的建设提供全套解决方案。

B/S 结构下 Excel 文件的在线处理

黄 勇 孙 婕

摘 要 Excel 是最流行的图表处理软件。在网络中存在着大量以 Excel 格式传递的文件。B/S 结构是网络时代应用最多的技术，如何实现 Excel 文件在三层 B/S 结构下的实时在线处理具有很高的实用价值。本文通过一个实例说明如何使用 ASP 技术实现服务器端对 Excel 文件的在线处理，即实现 Excel 文件与数据库的信息交换。

关键词 Excel， ASP， B/S

引言

Microsoft 公司的 Office 系列软件在国内外都有很高的市场占有率，Excel 也因此成为广泛使用的图表处理软件。许多用户的数据都是以 Excel 文件格式存储。随着网络日益普及的时代，Excel 文件的在线处理也成为许多网站必不可少的任务。

本文将通过具体实例介绍使用 ASP 技术实现用户端 Excel 数据文件的提交和服务器端动态生成 Excel 文件的方法。

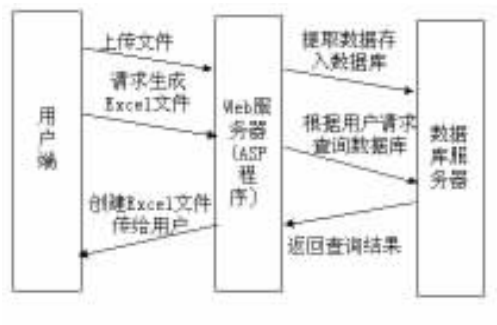
问题描述

授权销售用户需要通过 Internet 向远程服务器提交自己的销售计划，计划文件以 Excel 格式存储，其中包括产品名称，进货价，销售价，数量等信息，服务器收到用户提交的 Excel 文件后，从 Excel 文件提取数据存入相应的数据库。

授权销售用户需要通过 Internet 查询最新的价格表，价格表以 Excel 格式提供。用户可以在线随机选择不同的产品，服务器端根据用户的选择查询数据库，动态生成 Excel 格式的价格表返回给客户，客户可以另存为本机文件。为了便于程序的处理，上传和下载的 Excel 格式文件都必须满足一定的样式。

处理流程

整个过程可以用下面的图示描述：



处理流程

从图中可以看出，在三层 B/S 结构中，用户只需要提交请求，即可完成相应的功能。用户将 Excel 数据文件上传给 Web 服务器，嵌在 Web 服务器端的 ASP 程序能实现对上传文件的分析，并分离出相应数据存入数据库中。

同时，对于用户提出浏览数据的请求，ASP 程序在分析之后，通过与数据库的通信，取得数据并以 Excel 文件形式返回给用户端。这就完成了用户需要实现的整个功能。

开发平台

服务器端：由于服务器端要处理 Excel 文件，因此在服务器端必须应用 Windows 平台，我们这里应用的是 NT Server4；Option Pack4；安装有 Excel 97；同时为了实现文件上载，也须安装相应组件。

客户端：客户端需在线打开 Excel，因而也须使用 Windows 平台，并安装 Excel。

数据库结构与 Excel 文件样式

数据库中数据存储在 table 中。以前面提到的问题为例，必须建立以下几个表：用户信息表：存放用户 ID，用户名，地址信息，及其他有关信息；产品信息表：存储产品 ID，产品名及价格等信息；用户定购产品情况表：包括产品 ID，用户 ID 及交易情况。具体情况如下表所示：

用户信息

Users //表名			
Name	Data Type	Size	Allow Null
User_ID //用户代码	Varchar	20	
User_Name //用户名	Varchar	40	
Company //公司名	Varchar	50	
Telephone_NO //电话	Varchar	10	√
Address //地址	Varchar	50	√

产品信息

Products //表名			
Name	Data Type	Size	Allow Null
Product_ID //产品代码	Varchar	20	
Product_Name //产品名称	Varchar	40	
Trade_Price //批发价	Decimal	8	√
Retail_Price //零售价	Decimal	8	√

用户定购产品情况

User_Product //表名			
Name	DataType	Size	Allow Null
Product_ID //产品代码	Varchar	20	
User_ID //用户代码	Varchar	20	
Quantity //交易数量	Int	8	

上传的 Excel 文件统一采用以下样式：

产品代码	批发价	零售价	定购数量

下载 Excel 文件采用以下样式：

产品代码	产品名称	批发价	零售价

客户 Excel 文件提交处理

● 功能描述

用户将所需存入数据库的数据用 Excel 文件形式（具体样式如上）存放，并上载到服务器，服务器对文件进行分析之后，将数据导入数据库。

● 具体实现过程

● 用户上传文件代码

在客户端，为了能上传文件，必须提供给用户以相应的界面。用 ASP 描述，提交表单代码如下：

```
<form enctype = "multipart/form-data" method = "POST" name = "fm_upload" action = "upload.asp" >
<TABLE bgColor = #6666cc border = 0 cellPadding = 1 cellSpacing = 0>
<TBODY> <TR> <TD align = middle bgColor = #6666cc colSpan = 2><B>上载 Excel 文件 </B> </TD> </TR> <TR>
<INPUT TYPE = "file" size = 10 name = "filename"><BR><INPUT onclick = "Submit" TabIndex = 3 Type = Button value = "上载" ></TR></TABLE> </form>
```

● 处理 Excel 文件

用户提交表单后，由服务器端组件将上传的文件保存在服务器端，关于服务器端的上传组件在许多地方有详细介绍，在此忽略。

Upload.asp 将用户上传的文件另存在服务器，再从服务器端保存的用户上传的 Excel 文件中提取数据。为了处理 Excel 文件，在服务器端引用 Automation 对象 Excel.Sheet 来实现对 Excel 文件的操作。

Excel.Sheet 的主要用法如下：

1. 创建 Excel.Sheet 对象的引用：

```
Set ExcelSheet = CreateObject ("Excel.Sheet")
```

2. 当服务器端出现异常时，为了能够关闭对 Excel 的引用，可以手工关闭，但要加上 ExcelSheet.Application.Visible = true 以使得其在任务管理器中可见；

3. 打开 Excel 文件，假设文件存放在 C hInetPub hwwwroot huploads 文件夹中，文件名为 data.xls：

```
ExcelSheet.Application.WorkBooks.Open( " C:\InetPub\wwwroot\fc\uploads\data.xls")
```

4. 选择文件的某一页：

```
ExcelSheet.Application.Sheets("Sheet1").Select
```

5. 引用当前页的第一个单元：ExcelSheet.Application.

```
ActiveWorkBook.ActiveSheet.Cells(1,1)
```

6. 关闭对 Excel.Sheet 的引用：

```
ExcelSheet.Application.ActiveWorkbook.Close()
ExcelSheet.Application.Quit
```

由上可知，应用 Excel.Sheet 对象访问已上传到服务器上的 Excel 文件，将有用的信息提取出来组成 SQL 语句，即能将信息添加到数据库中。

实现上述功能的具体程序代码如下：（假设用户已将上传的程序存放在目录 C hInetPub hwwwroot huploads h中，并命名为 data.xls）

```
Set ExcelSheet = CreateObject ("Excel.Sheet")
ExcelSheet.Application.Visible = true
ExcelSheet.Application.WorkBooks.Open( " C:\InetPub\wwwroot\uploads\data.xls")
i = 2
ExcelSheet.Application.Sheets("Sheet1").Select
While ExcelSheet.Application.ActiveWorkBook.ActiveSheet.Cells ( i, 1 ) <> 0 and ExcelSheet.Application.ActiveWorkBook.ActiveSheet.Cells(i, 2) <>0
pid =ExcelSheet.Application.ActiveWorkBook.ActiveSheet.Cells(i, 1))
        // 从 Excel 文件中读取产品代码
 uid = Request.Form("login_id")
 // 从登录信息中取用户代码
 Sql = "Select * from User_Product where User_ID =" & uid & " and Product_ID = " & pid
 // 查询 User_Product 表中是否已有定购信息
 Set rs = conn.Execute (Sql)
 If rs.Eof Then //如果没有插入信息
   sql = "Insert Into User_Product Values (" & pid & "," & uid & "," & ExcelSheet.Application.ActiveWorkBook.ActiveSheet.Cells(i, 4) & ")"
   conn.Execute (sql)
 End If
 i = i + 1
Wend
```

服务器端动态生成 Excel 文件

● 实现功能描述

用户将需求通过表单提交给服务器，服务器接收用户信息，从数据库中取出相应数据并以 Excel 文件形式返回给客户。

● 具体实现过程

● 服务器接收用户提交表单，根据用户提交信息构造 SQL 语句

用脚本文件修改注册表

丁 健 龚 怒 李炎新 都的箭

Windows32 位系统的注册表包括注册表数据库和注册表编辑器。用注册表编辑器手工进行修改操作时有以下缺点：修改立即生效，而编辑器没有 Undo 功能，没有警告，也没有详细的帮助信息。因此手工修改很不安全。笔者通过对 Reg 注册表脚本文件、Inf 安装信息文件结构的研究，顺利地实现了用这两种脚本文件修改注册表，可以替代手工操作。

一、利用 Reg 注册表脚本文件

Reg 是注册表编辑器的脚本文件，它是一种文本文件，可以用任何文本编辑器对它进行编写。因为 Reg 文件与 Regedit 应用程序关联，我们可以用执行 REGEDIT 应用程序带 REG 文件名作为命令行参数的语句，实现注册表信息增加、更新操作。编程任务可概括为：①创建脚本文件，②将脚本文件内容合并入注册表。

REG 文件有严格的格式规定，对不符合格式的脚本文件，注册表编辑器解释时会发生错误，注册会失败。REG 文件的格式为：

1. 文件第一行是 "Regedit4 "；
2. 文件最后一行为空行；
3. 在注册表中新建一个主键值对应的语法是：[路径 h主键值名]；
4. 更改主键值默认值的语法是：@ = "默认值"；
5. 在主键下建立新串值要在相应的主键下添加："串值名" = "串值"；
6. 新建一个二进制值的语法是："二进制值名" = hex 二进制值；
7. 新建一个 DWORD 值的语法是："DWORD 值名" = dword Dword 值；
8. 凡用到应用程序或动态链接库文件路径的字符串中"h"要用 "hh"来代替；
9. 以 "；"开头的行为注释行。

我们以在 VB5.0 环境中编程实现 "HH"扩展名文件与记事本 "Notepad"应用程序关联为例，说明编程方法，程序段如下：

```
'定义变量
Dim str_filetype As String
Dim str_appname As String
'扩展文件类型
str_filetype = "hh"
```

假设用户希望返回所有批发价与零售价差值大于 10 元的产品，构造 SQL 语句：Sql = "Select * from products where Retail_Price - Trade_Price > 10"，使用 ASP 的 ActiveX Data Objects ADO 对象可以方便地实现与数据库的连接。实现这一步查询在所有的 ASP 书籍中都有详细的说明。

下面给出了与数据库的连接的程序段：

```
Set conn = Server.CreateObject("ADODB.Connection")
conn.Open "example", "sa", ""
Set rs =conn.Execute(Sql)
```

● 将查询结果按照约定的格式写入 Excel 文件中

```
i = 2
While Not rs.EOF
ExcelSheet.Application.ActiveWorkBook.ActiveSheet.Cells
(i, 1) = rs("Product_ID")
ExcelSheet.Application.ActiveWorkBook.ActiveSheet.Cells
(i, 2) = rs("Product_Name")
ExcelSheet.Application.ActiveWorkBook.ActiveSheet.Cells
(i, 3) = rs("Trade_Price")
ExcelSheet.Application.ActiveWorkBook.ActiveSheet.Cells
(i, 4) = rs("Retail_Price")
rs.MoveNext
i =i +1
Wend
```

● 将结果文件返回给用户

因为 ASP 不能往远程的客户机上写文件，只能保存在服务器本地硬盘，所以应用 Response.directe，将结果文件重新定向给用户。

```
ExcelSheet.Application.ActiveWorkbook.Save()
ExcelSheet.Application.ActiveWorkbook.Close()
ExcelSheet.Application.Quit
Response.Redirect("/results/result.xls")
//假设结果文件路径为 c:\inetpub\wwwroot\results\result.xls.
```

结论

本文给出了一种在三层 B/S 结构下 Excel 文件的在线处理方法，是基于 B/S 结构的数据库与 Excel 文件交互信息的方法。这种方法已经在实践中得到了应用，是一种比较实际的方法。

参考书目

Visual InterDev 6.0 网络编程技术. 人民邮电出版社

收稿日期：2000 年 10 月 25 日）


```
'关联的应用程序
str_appname = "D:\WIN95\NOTEPAD.EXE"
'创建 link.reg 脚本文件
Open App.Path + "\link.reg" For Output As #1
Print #1, "REGEDIT4"
Print #1, "[HKEY_CLASSES_ROOT\." + str_filetype + "]"
Print #1, "@ =" + Chr(34) + str_filetype + "file" + Chr(34)
Print #1,
Print #1, "[HKEY_CLASSES_ROOT\" + str_filetype + "file\
shell\open\command]"
Print #1, "@ =" + Chr(34) + backslash(str_appname) + " %1"
+ Chr(34)
Print #1,
Close #1
'合并信息
Shell "regedit.exe " + App.Path + "\link.reg", vbHide
```

其中 backslash 是一个自编函数，用于将一个字符串内的“h”替代成“hh”，以符合脚本内应用程序路径要求，上例中字符串“D hWIN95 hNOTEPAD.EXE”经该函数处理后变成“D hhWIN95 hhNOTEPAD.EXE”，符合格式中第 8 条要求。在 VB6.0 环境中有函数可以直接实现字符串内字符替代。

```
Function backslash(ByVal ss As String) As String
    Dim k As Integer, i As Integer, read_str_temp As String
    k = 0
    i = 0
    For i = 1 To Len(ss)
        If Mid$(ss, i, 1) = "\" Then
            k = k + 1
        End If
    Next
  read_str_temp = ss
    backslash = ""
    For i = 1 To k
      backslash = backslash + Mid$(read_str_temp, 1, InStr
(1, read_str_temp, "\")) + "\"
      read_str_temp = Mid$ (read_str_temp, InStr(1, read_str_
temp, "\") + 1, Len(read_str_temp) - InStr(1, read_str_temp,
"\"))
    Next
    backslash = backslash + read_str_temp
End Function
```

二、利用安装信息文件

Inf 文件是安装信息文件，也是文本文件，也可以用任何文本编辑器编写。右击 INF 选“安装”或按 SHIFT 键同时按功能键 F10 均可使内含的注册表操作动作起作用。类似地，编程任务可概括为：①创建脚本文件，②使文件内动作产生作用。Inf 脚本文件格式为：

1. 文件内须包含[Version]小节，位置不限，内容须为 signature = “$CHICAGO$”；

2. 包含[DefaultInstall]小节，小节内关键词 AddReg 指明动作是“增加”键或子键，小节内关键词 DelReg 指明动作是“删除”键或子键，在[DefaultInstall]小节内可以同时有多个 AddReg、DelReg 关键词；

3. 关键词的值是具体动作内容所在节的节名索引；

4. 增加、更新动作内容语法是：“根键，键路径，子键名，子键值类型，子键具体值”；

5. 删除动作内容的语法是：“根键，键路径，子键名，，”；

6. 根键用 HKCR HKEY_CLASSES_ROOT、HKCU HKEY_CLASSES_USER、HKLM HKEY_LOCAL_MACHINE 等省略写法；

7. 键路径语法是：＊＊＊h＊＊＊h＊＊＊；setupx.dll 文件内预定义有一些目录路径，如%1% 代表本 Inf 文件所在目录路径、%11% 代表 WINDOWS 的 SYSTEM 目录存在路径，其他可以通过调试获知；

8. 子键值类型：1 为二进制，2 为字符串键，其他类推；

9. 如创建缺省值，省略子键名和子键值类型；

VB5.0 启动时有时会莫名其妙地提示未知错误，然后退出，对经常用 VB 编程的用户来说，此提示相当频繁。尝试删除下面路径内所有键后，再次启动，VB 可正常运行。

```
HKEY_CLASSES_USER\Software
  \Microsoft
    \Visual Basic\5.0
        \RecentFiles
```

显然，手工操作不甚方便。可用 VB 编译一个 EXE 文件，放至桌面，备用。关键程序段为：

```
Open App.Path + "\link.inf" For Output As #1
      Print #1, "[Version]"
Print #1, "signature =" + Chr(34) + "$CHICAGO$" + Chr(34)
      Print #1, "[DefaultInstall]"
      Print #1, "DelReg = Del_VBRecentFiles.Reg"
      Print #1, "[Del_VBRecentFiles.Reg]"
      Print #1, "HKCU, %Location%, 1, ,"
      Print #1, "HKCU, %Location%, 2, ,"
      Print #1, "HKCU, %Location%, 3, ,"
      Print #1, "HKCU, %Location%, 4, ,"
      Print #1, "[Strings]"
  Print #1, "Location = ""Software\Microsoft\Visual Basic\
5.0\RecentFiles"""
Close #1
Shell "rundll.exe setupx.dll, InstallHinfSection DefaultInstall
132 " & App.Path & "\link.inf", vbHide
   Kill App.Path + "\link.inf"
```

程序运行中间过程生成如下脚本文件：

```
[Version]
signature = "$CHICAGO$"
[DefaultInstall]
DelReg = Del_VBRecentFiles.Reg
[Del_VBRecentFiles.Reg]
HKCU, %Location%, 1, ,
HKCU, %Location%, 2, ,
HKCU, %Location%, 3, ,
HKCU, %Location%, 4, ,
```

如何在 PowerBuilder 中实现 MapInfo 的 callback

杨 勇 傅家祥

摘 要 在我们的程序中集成 MapInfo 应用后，MapInfo 一般不自动向客户程序发送信息。本文提出了解决如何让 PowerBuilder 应用程序接收 MapInfo 发送的消息，使应用程序据此进行相应处理的有效方法。

一、概述

MapInfo 是一个功能强大、全面直观的桌面地理信息系统（GIS）。其功能的强大及操作的简单性，受到市场的广泛应用，但对大型的数据库应用系统程序的开发显得有点力不从心。PowerBuilder 是专门设计企业级客户 / 服务器（Client / Server）模式应用程序的开发环境，随着数据库技术在各行各业的广泛应用，作为企业级数据库前端开发工具的 PowerBuilder 日益成为开发人员的得力助手。PowerBuilder 以其开放的体系结构，友好的用户界面和简洁高效的开发环境赢得了众多程序员的喜爱，连续多年被评为美国计算机界的年度风云产品，在数据库开发工具领域占据了高达 44% 的市场份额。所以在大型的 GIS 数据库应用程序的开发，我们往往综合二者的特点，用 PB 对 MapInfo 进行二次开发。很多情况下是采用将 MapInfo 作为 OLE 对象集成到二次开发的应用程序中。开发的客户（Client）应用程序作为客户端运行在前台，直接与用户进行交互操作；MapInfo 作为服务器（Server）在后台运行，对 Client 程序的请求做出回答。一般情况下，MapInfo 不会自动地把信息传递给 Client 程序，是处于很被动的应用。

在很多情况下，当用户对 MapInfo 地图窗口的操作时，如改变窗口大小、关闭某一图层、点击某一自定义按钮等，需要应用程序据此及时地进行相应的处理，就需要 MapInfo 能够主动地发送信息。也即这时 MapInfo 要充当 Client 的角色，而原定的 Client 应用程序转为 Server 的角色。MapInfo 提供了这种自动发送信息给客户应用程序的功能——称为回调（Callback）。

二、Callback 简介

在我们的程序中集成 MapInfo 应用后，MapInfo 能够自动地向客户程序发送信息。例如，在你的应用程序中，当地图窗口发生变化时，MapInfo 将调用你的客户端程序通知你发生变化的窗口的句柄。当发生某一事件导致 MapInfo 调用你的客户端程序的这种方法称为 Callback（回调）。

在下列情况中，Callback 允许 MapInfo 发送消息给您的客户应用程序：

● 用户使用定制工具与 MapInfo 窗口进行交互

例如，用户点击并拖动鼠标在 MapInfo 的地图窗口上画线时，MapInfo 能够把用户选择的各点坐标消息发送给应用程序。

● 用户选择菜单命令

例如，如果应用程序定制了 MapInfo 的快捷菜单（即用户右击时弹出的菜单），当用户从快捷菜单中选择了定制的命令后，MapInfo 就会把这个菜单事件通知给应用程序。

● 地图窗口发生变化

当用户改变地图窗口内容（如添加或删除图层，平移地图时），MapInfo 就会将发生变化的窗口句柄告知应用程序（这类似于 MapBasic 自身的句柄过程：WinChangedHandler）。

● MapInfo 中状态栏的内容发生变化。

三、用 PB 中实现 MapInfo 的 callback

为了使用 MapInfo 的 callback，我们的应用程序必须能作为 DDE 服务器或 OLE 自动化服务器。VC + +，VB，Delphi 等开发工具对 OLE 自动化服务的支持都很好，开发起来很方便。但 PB 对 OLE 自动化服务的支持不够理想，笔者和周围的许多同行试了很多方法都未能实现 callback。是不是在 PB 里就不能使用 MapInfo 的 callback 呢？笔者出于对 PB 的偏爱及执着，坚持不懈地查阅了大量的文献，终于找到了一个方法实现了 callback。喜悦之余，不忘与大家分享。

[Strings]
Location = "Software\Microsoft\Visual Basic\5.0\RecentFiles"

在实际使用中应该注意借鉴和摸索，如上段代码中使用了字符串宏。

本文介绍的两种方法简单、方便、快捷，通俗易懂，可以实现对注册表键、子键、值的增加、更新操作，Inf 脚本文件还可以实现删除操作。API 函数可以进行的枚举操作用脚本文件不容易实现，尚须进一步编程。

（收稿日期：2000 年 11 月 23 日）

实现步骤如下：

打开 PB 的库画板，新建一个 PB 库并命名为 Ole_CallBack. pbl；定义一个立即数变量：integer handle ；在库中创建一个不可见自定义用户对象：nuo_CallBack，为其添加用户对象函数：public subroutine uf_ini integer Whnd ，无返回值。

在其中脚本中添加代码如下：

```
handle = Whnd
```

如果你要将 MapInfo 状态栏的信息反映到客户应用程序的状态栏上，则要添加一个名为 SetStatusText 用户对象函数，只有一个字符串型 （string）的参数。当每次地图窗口发生变化时，你需要 MapInfo 通知你的应用程序，就要添加一个名为 WindowContentsChanged 用户对象函数，只有一个四字节整型 （integer）的参数。当自定义菜单项或自定义按钮被使用时，你需要 MapInfo 通知你的应用程序，就要添加一个名为与你的自定义菜单项或自定义按钮的过程名相同的用户对象函数，只有一个字符串型 （string）的参数。限于篇幅，我们在这里仅添加名为 WindowContentsChanged 用户对象函数，参数为：integer as_response，无返回值。

在其中脚本中添加代码如下：

```
send(handle, 1024, 0, as_response) /* 向句柄号为 handle 的窗口发送系统消息为 pbm_custom01 的消息，并把 as_response 也发送过去 */
```

保存库文件并编译成 PB 运行时链连库 Ole_CallBack. pbd，这是关键所在。我们的第一阶段就完成了，在后面的工作中我们要使用到这个链接库。

在 PB 的库画板再新建一个名为 CallBack_Test. pbl 的 PB 库，新建一个 MDI 窗口 w_main 和一个用作 w_main 的 sheet 的窗口 w_sheet。

为 w_main 新增一个用户自定义事件：ue_MapInfoWCHnd 事件号为 pbm_custom01，用以接收 MapInfo 发来的窗口句柄。

在 w_main 的 open 事件中编写代码如下：

```
openSheet(w_sheet, w_main)
```

在 w_main 的 ue_MapInfoWCHnd 事件中编写代码如下：

```
messagebox("回调测试", "MapInfo 返回的地图窗口发生变化的窗口句柄为:" + string( as_response ) )
```

在 w_sheet 的 open 事件中编写代码如下

```
OleObject myPBOleObject
myPBOleObject =create OleObject /* 创建 PowerBuilder OLE 自动化服务对象 */
int Errcode
ErrCode = myPBOleObject. ConnectToNewObject( " PowerBuilder. Application")
If ErrCode <>0 Then
MessageBox("错误", "连接 PowerBuilder 服务器错误，退出.")
    Halt Close
end if
/* 这就是我们前面编译好的链接库 Ole_CallBack. pbd 的完整路径 */ /
myPBOleObject. LibraryList = "c: \Ole_CallBack. pbd"
myPBOleObject. MachineCode =False    * 如果你编译成的是机器码就用 True, 这里是 False */
OleObject UserOleObject /* 创建用户自定义 OLE 对象 */
UserOleObject =create OleObject
/* 链接到不可见自定义用户对象: nuo_CallBack */
  UserOleObject =myPBOleObject. CreateObject ( "nuo_CallBack")
if isNull(UserOleObject) then
messagebox("没有对象", "不能创建对象 UserOleObject ")
halt close
end if
myPBOleObject. uf_int(handle(w_main)) /* 把主窗口的句柄传递给用户自定义 OLE 对象 */
OLEObject MIOleObject
MIOleObject = Create OLEObject
/* 连接 MapInfo OLE 服务器 */
ErrCode = MIOleObject. ConnectToNewObject( " MapInfo. Application")
If ErrCode <>0 Then
  MessageBox("连接错误", "连接地图服务器错误, 在使用本程序前要安装 MapInfo 软件. ")
  Halt
End If
/* 把我们的自定义 OLE 对象传递给 MapInfo OLE 对象的 CallBack */
MIOleObject. SetCallBack(UserOleObject)
/* 重新定向 MapInfo 的地图窗口到 w_sheet 上 */
MIOleObject. Do( " Set Next Document Parent " + String (Handle(this)) + " Style 1")
MIOleObject. Do( ' Set Application Window ' +String(Handle (this)))
/* 打开 MapInfo 地图并在 w_sheet 上显示 */
MIOleObjectDo('run application"c: \maptest. wor"')
```

编译执行。改变地图窗口的大小或图层，w_main 主窗口将会接收到 MapInfo 发来的消息，将会弹出我们前面的消息窗口显示发生变化地图窗口的句柄，将它替换成你的代码就可以进行相应的处理了。

四、说明

本程序在 PowerBuilder 6. 5 下编译测试通过，在 MapInfo Professional 4. 0、MapInfo Professional 4. 5 和 MapInfo Professional 5. 0 环境下均运行良好。

五、结论

PB 对 OLE 自动化服务的支持并没有 VC + + 、VB、Delphi 等开发工具的支持强大，我们只好采用迂回策略，定制一个 PB 用户自定义 OLE 对象用来作 MapInfo 的 OLE 调用，再把自定义 OLE 对象接收到的消息发送给我们的应用程序，从而实现了 PB 与 MapInfo 的交互，进而开发出功能完善的大型 GIS 数据库应用程序。

（收稿日期：2000 年 11 月 6 日）

利用 MSXML 解析 XML 文本

胡朝晖

摘 要 主要介绍了 XML 的特点和应用，同时详细分析了 MSXML 解析器的特点，并分析了文档对象模型 DOM 的结构和应用，同时用 VC 编程语言给出了通过 MSXML 进行 XML 解析的实例。

关键词 MSXML，XML，XML 解析器，文档对象模型，接口，Windows

一、引言

当前 Web 上流行的剧本语言是以 HTML 为主的语言结构，HTML 是一种标记语言，而不是一种编程语言，主要的标记是针对显示，而不是针对文档内容本身结构描述的。也就是说，机器本身是不能够解析它的内容的，所以就出现了 XML 语言。XML eXtensible Markup Language 语言是 SGML 语言的子集，它保留了 SGML 主要的使用功能，同时大大缩减了 SGML 的复杂性。XML 语言系统建立的目的就是使它不仅能够表示文档的内容，而且可以表示文档的结构，这样在能够被人类理解的同时，也能够被机器所理解。 XML 要求遵循一定的严格的标准。XML 分析程序比 HTML 浏览器更加要挑剔语法和结构，XML 要求正在创建的网页正确的使用语法和结构，而不是象 HTML 一样，通过浏览器推测文档中应该是什么东西来实现 HTML 的显示，XML 使得分析程序不论在性能还是稳定性方面都更容易实现。XML 文档每次的分析结果都是一致的，不象 HTML，不同的浏览器可能对同一个 HTML 作出不同的分析和显示。同时因为分析程序不需要花时间重建不完整的文档，所以它们能比同类 HTML 更有效地执行其任务。它们能全力以赴地根据已经包含在文档中的那个树结构建造出相应的树来，而不用在信息流中的混合结构的基础上进行显示。XML 标准是对数据的处理应用，而不是只针对 Web 网页的。任何类型的应用都可以在分析程序的上面进行建造，浏览器只是 XML 的一个小的组成部分。当然，浏览仍旧极其重要，因为它为 XML 工作人员提供用于阅读信息的友好工具。但对更大的项目来说它就不过是一个显示窗口。因为 XML 具有严格的语法结构，所以我们甚至可以用 XML 来定义一个应用层的通讯协议，比如互联网开放贸易协议 （nternet Open Trading Protocol）就是用 XML 来定义的。从某种意义上说，以前我们用 BNF 范式定义的一些协议和格式从原则上说都可以用 XML 来定义。实际上，如果我们有足够的耐心，我们完全可以用 XML 来定义一个 C + + 语言的规范。

当然，XML 允许大量 HTML 样式的形式自由的开发，但是它对规则的要求更加严格。 XML 主要有三个要素：DTD Document Type Declaration 文档类型声明 或 XML Schema （XML 大纲）、XSL （Xtensible Stylesheet Language 可扩展样式语言）和 XLink （Xtensible Link Language 可扩展链接语言）。DTD 和 XML 大纲规定了 XML 文件的逻辑结构，定义了 XML 文件中的元素、元素的属性以及元素和元素的属性之间的关系；Namespace 名域 实现统一的 XML 文档数据表示以及数据的相互集成；XSL 是用于规定 XML 文档呈现样式的语言，它使得数据与其表现形式相互独立，比如 XSL 能使 Web 浏览器改变文档的表示法，例如数据的显示顺序的变化，不需要再与服务器进行通讯。通过改变样式表，同一个文档可以显示得更大，或者经过折叠只显示外面的一层，或者可以变为打印的格式。而 XLink 将进一步扩展目前 Web 上已有的简单链接。

二、实现 XML 解析的说明

当然，从理论上说，根据 XML 的格式定义，我们可以自己编写一个 XML 的语法分析器，但是实际上微软已经给我们提供了一个 XML 语法解析器，如果你安装了 IE5. 0 以上版本的话，实际上你就已经安装了 XML 语法解析器。可以从微软站点 （www. microsoft. com）下载最新的 MSXML 的 SDK 和 Parser 文件。它是一个叫做 MSXML. DLL 的动态链接库，最新版本为 msxml3，实际上它是一个 COM 对象库，里面封装了所有进行 XML 解析所需要的所有必要的对象。因为 COM 是一种以二进制格式出现的和语言无关的可重用对象。所以你可以用任何语言 比如 VB、VC、DELPHI、C + + Builder 甚至是剧本语言等等 对它进行调用，在你的应用中实现对 XML 文档的解析。下面的关于 XML 文档对象模型的介绍是基于微软最新的 msxml3 为基础进行的。

三、XML 文档对象 （XML DOM）模型分析

XML DOM 对象提供了一个标准的方法来操作存储在 XML 文档中的信息，DOM 应用编程接口 （API）用来作为应用程序和 XML 文档之间的桥梁。

DOM 可以认为是一个标准的结构体系用来连接文档和应用程序 也可以是剧本语言）。MSXML 解析器允许你装载和创建一个文档，收集文档的错误信息，得到和操作文档中的所有的信息和结构，并把文档保存在一个 XML 文件中。DOM 提供给用户一个接口来装载、到达和操作并序列化 XML 文档。

DOM 提供了对存储在内存中的 XML 文档的一个完全的表示，提供了可以随机访问整个文档的方法。DOM 允许应用程序根据 MSXML 解析器提供的逻辑结构来操作 XML 文档中的信息。利用 MSXML 所提供的接口来操作 XML。

实际上 MSXML 解析器根据 XML 文档生成一个 DOM 树结构，它能够读 XML 文档并根据 XML 文档内容创建一个节点的逻辑结构，文档本身被认为是一个包含了所有其他节点的节点。

DOM 使用户能够把文档看成是一个有结构的信息树，而不是简单的文本流。这样应用程序或者是剧本即使不知道 XML 的语义细节也能够方便地操作该结构。DOM 包含两个关键的抽象：一个树状的层次、另一个是用来表示文档内容和结构的节点集合。树状层次包括了所有这些节点，节点本身也可以包含其他的节点。这样的好处是对于开发人员来说，他可以通过这个层次结构来找到并修改相应的某一个节点的信息。DOM 把节点看成是一个通常的对象，这样就有可能创建一个剧本来装载一个文档，然后遍历所有的节点，显示感兴趣的节点的信息。注意节点可以有很多中具体的类型，比如元素、属性和文本都可以认为是一个节点。

微软的 MSXML 解析器读一个 XML 文档，然后把它的内容解析到一个抽象的信息容器中称为节点（NODES）。这些节点代表文档的结构和内容，并允许应用程序来读和操作文档中的信息而不需要显示知道 XML 的语义。在一个文档被解析以后，它的节点能够在任何时候被浏览而不需要保持一定的顺序。

通过 DOM 对 XML 文档进行解析的图例如下图所示：

对开发人员来说，最重要的编程对象是 DOMDocument。DOMDocument 对象通过暴露属性和方法来允许你浏览，查询和修改 XML 文档的内容和结构，每一个接下来的对象暴露自己的属性和方法，这样你就能够收集关于对象实例的信息，操作对象的值和结构，并导航到树的其他对象上去。

MSXML. DLL 所包括的主要的 COM 接口有：

1 DOMDocument

DOMDocument 对象是 XML DOM 的基础，你可以利用它所暴露的属性和方法来允许你浏览、查询和修改 XML 文档的内容和结构。DOMDocument 表示了树的顶层节点。它实现了 DOM 文档的所有的基本的方法并且提供了额外的成员函数来支持 XSL 和 XSLT。它创建了一个文档对象，所有其他的对象都可以从这个文档对象中得到和创建。

2 IXMLDOMNode

IXMLDOMNode 是文档对象模型（DOM）中的基本的对象，元素，属性，注释，过程指令和其他的文档组件都可以认为是 IXMLDOMNode，事实上，DOMDocument 对象本身也是一个 IXMLDOMNode 对象。

3 IXMLDOMNodeList

IXMLDOMNodeList 实际上是一个节点 Node 对象的集合，节点的增加、删除和变化都可以在集合中立刻反映出来，可以通过 "for. . . next" 结构来遍历所有的节点。

4 IXMLDOMParseError

IXMLDOMParseError 接口用来返回在解析过程中所出现的详细的信息，包括错误号，行号，字符位置和文本描述。

下面主要描述一个 DOMDocument 对象的创建过程，这里我们用 VC 描述创建一个文档对象的过程。

```
HRESULT hr;
IXMLDomDocument * pXMLDoc;
IXMLDOMNode * pXDN;
Hr = CoInitialize(NULL);  //COM 的初始化
//得到关于 IXMLDOMDocument 接口的指针 pXMLDOC.
   hr = CoCreateInstance(CLSID_DOMDocument,  NULL,
CLSCTX_INPPROC_SERVER,
 IID_IXMLDOMDocument, (void * * )& pXMLDoc);
//得到关于 IXMLDOMNode 接口的指针 pXDN.
      hr = pXMLDoc - >QueryInterface(IID_IXMLDOMNode,
(void * * )& pXDN);
```

在 MSXML 解析器使用过程中，我们可以使用文档中的 createElement 方法来创建一个节点装载和保存 XML 文件。通过 load 或者是 loadXML 方法可以从一个指定的 URL 来装载一个 XML 文档。Load LoadXML 方法带有两个参数：第一个参数 xmlSource 表示需要被解析的文档，第二个参数 isSuccessful 表示文档装载是否成功。Save 方法是用来把文档保存到一个指定的位置。Save 方法有一个参数 destination 用来表示需要保存的对象的类型，对象可以是一个文件，一个 ASP Response 方法，一个 XML 文档对象，或者是一个能够支持持久保存（persistence）的客户对象。下面是 save 方法使用的一个简单的例子：

```
BOOL DOMDocSaveLocation()
{
  BOOL bResult = FALSE;
  IXMLDOMDocument * pIXMLDOMDocument = NULL;
  HRESULT hr;
   try
  {
    _variant_t varString = _T(″D: \\sample. xml″);
    // 这里需要创建一个 DOMDocument 对象和装载 XML 文档, 代码省略.
     hr = pIXMLDOMDocument - >save(varString); //保存文档到 D: \\sample. xml 中去.
     if(SUCCEEDED(hr))
       bResult = TRUE;
   }
   catch(. . . )
```

```
    {
      DisplayErrorToUser();
// 这里需要释放对 IXMLDOMDocument 接口的引用,代码省略.
    }
    return bResult;
  }
```

同时，在解析过程中，我们需要得到和设置解析标志。利用不同的解析标志，我们可能以不同的方法来解析一个 XML 文档。XML 标准允许解析器验证或者不验证文档，允许不验证文档的解析过程跳过对外部资源的提取。另外，你可能设置标志来表明你是否要从文档中移去多余的空格。

为了达到这个目的，DOMDocument 对象暴露了下面几个属性，允许用户在运行的时候改变解析器的行为：

1. Async 相对于C++是两个方法，分别为 get_async 和 put_async

2. ValidateOnparse 相对于 C++ 是两个方法，分别为 get_validateOnParse 和 put_validateOnParse

3. ResolveExternals 相对于 C++ 是两个方法，分别为 get_ ResolveExternals 和 put_ ResolveExternals

4. PerserveWhiteSpace 相对于 C++是两个方法，分别为 get_ PerserveWhiteSpace 和 put_ PerserveWhiteSpace

每一个属性可以接受或者返回一个 Boolean 值。缺省的，anync validateOnParse resolveExternals 的值为 TRUE，perserveWhiteSpace 的值跟 XML 文档的设置有关，如果 XML 文档中设置了 xml space 属性的话，该值为 FALSE。

同时在文档解析过程中可以收集一些和文档有关的信息，实际上在文档解析过程中可以得到以下的信息：

1. doctype 文档类型 ：实际上是用来定义文档格式的 DTD 文件。如果 XML 文档没有相关的 DTD 文档的话，它就返回 NULL。

2. implementation 实现 ：表示该文档的实现，实际上就是用来指出当前文档所支持的 XML 的版本。

3. parseError 解析错误 ：在解析过程中最后所发生的错误。

4. readyState 状态信息 ：表示 XML 文档的状态信息，readyState 对于异步使用微软的 XML 解析器来说的重要作用是提高了性能，当异步装载 XML 文档的时候，你的程序可能需要检查解析的状态，MSXML 提供了四个状态，分别为正在状态，已经状态，正在解析和解析完成。

5. url 统一资源定位 ：关于正在被装载和解析的 XML 文档的 URL 的情况。注意如果该文档是在内存中建立的话，这个属性返回 NULL 值。

在得到文档树结构以后，我们可以操作树中的每一个节点，可以通过两个方法得到树中的节点，分别为 nodeFromID 和 getElementsByTagName。

nodeFromID 包括两个参数，第一个参数 idString 用来表示 ID 值，第二个参数 node 返回指向和该 ID 相匹配的 NODE 节点的接口指针。注意根据 XML 的技术规定，每一个 XML 文档中的 ID 值必须是唯一的，而且一个元素 （element）仅且只能和一个 ID 相关联。

getElementsByTagName 方法有两个参数，第一个参数 tagName 表示需要查找的元素 Element 的名称，如果 tagName 为 “*”的话返回文档中所有的元素 Element 。第二个参数为 resultList 它实际是指向接口 IXMLDOMNodeList 的指针，用来返回和 tagName 标签名字 相关的所有的 Node 的集合。

下面是一个简单的例子：

```
IXMLDOMDocument *pIXMLDOMDocument = NULL;
wstring strFindText (_T("author"));
IXMLDOMNodeList *pIDOMNodeList = NULL;
IXMLDOMNode *pIDOMNode = NULL;
long value;
BSTR bstrItemText;
HRESULT hr;
try
{
// 创建一个 DOMDocument 文档对象,并装载具体文档,相关代码省略
//下面的代码用来得到一个和标签名称 author 相关的所有的节点集合
hr = pIXMLDOMDocument->getElementsByTagName(
(TCHAR*)strFindText.data(), &pIDOMNodeList);
SUCCEEDED(hr) ? 0 : throw hr;
//是否正确的得到了指向 IDOMNodeList 的指针.
hr = pIDOMNodeList->get_length(&value); //得到所包含的 NODE 节点的个数
if(SUCCEEDED(hr))
{
pIDOMNodeList->reset();
for(int ii = 0; ii < value; ii++)
{
 //得到具体的一个 NODE 节点
pIDOMNodeList->get_item(ii, &pIDOMNode);
if(pIDOMNode )
{
pIDOMNode->get_text(&bstrItemText); //得到该节点相关的文本信息
::MessageBox(NULL, bstrItemText, strFindText.data( ), MB_OK);
pIDOMNode->Release();
pIDOMNode = NULL;
}
}
}
pIDOMNodeList->Release();
 pIDOMNodeList = NULL;
    }
    catch(...)
    {
      if(pIDOMNodeList)
        pIDOMNodeList->Release();
```

```
    if(pIDOMNode)
      pIDOMNode->Release();
    DisplayErrorToUser();
  }
```

最后我们讨论一下如何来创建新的节点，实际上可以通过方法 createNode 来创建一个新的节点。CreateNode 包括四个参数：第一个参数 Type 表示要创建的节点的类型，第二个参数 name 表示新节点的 nodeName 的值，第三个参数 namespaceURI 表示该节点相关的名字空间，第四个参数 node 表示新创建的节点。注意可以通过使用已经提供的类型 Type ，名称 name 和名字空间 nodeName 来创建一个节点。

当一个节点被创建的时候，它实际上是在一个名字空间范围 如果已经提供了名字空间的话）内创建的。如果没有提供名字空间的话，它实际上是在文档的名字空间范围内创建的。

四、利用 MSXML 进行 XML 文档分析的简单实例

为了说明如何在 VC 中使用 XML DOM 模型，这里我们显示了一个简单的实例程序，它是一个 Console Application。下面是主要的程序代码，本代码用来在一个 XML 文档中定位一个特殊的 Node 节点，并插入一个新的子节点。

```
#include <atlbase.h>
//下面的 .h 文件是在安装了最新的 XML Parser 以后所包含的 .h 文件.
#include "C:\Program Files\Microsoft XML Parser SDK\inc\msxml2.h"
#include <iostream>
void main()
{
    // 初始化 COM 接口
    CoInitialize(NULL);
//在程序中,我们假定我们装载的 XML 文件名称为 xmldata.xml,它缺省的和可执行文
//件在同一个目录中.该文件的内容如下:
    // <?xml version="1.0"?>
    // <xmldata>
    //   <xmlnode />
    //   <xmltext>Hello, World!</xmltext>
    // </xmldata>
    //程序将寻找名为"xmlnode"的节点,然后插入一个新的名称为"xmlchildnode"的节点,然后它去寻找一个名为"xmltest"的节点,然后提取包含在节点中的文本并显示它。最后它把新的改变过的 XML 文档保存在名称为"updatexml.xml"的文档中。
    try {
        // 通过智能指针创建一个解析器的实例.
        CComPtr<IXMLDOMDocument> spXMLDOM;
    HRESULT hr = spXMLDOM.CoCreateInstance(__uuidof(DOMDocument));
if ( FAILED(hr) ) throw "不能创建 XML Parser 对象";
if ( spXMLDOM.p == NULL ) throw "不能创建 XML Parser 对象";
// 如果对象创建成功的话,就开始装载 XML 文档
VARIANT_BOOL bSuccess = false;
hr = spXMLDOM->load(CComVariant(L"xmldata.xml"), &bSuccess);
if ( FAILED(hr) ) throw "不能够在解析器中装载 XML 文档";
if ( !bSuccess ) throw "不能够在解析器中装载 XML 文档";
  // 检查并搜索"xmldata/xmlnode"
CComBSTR bstrSS(L"xmldata/xmlnode");
CComPtr<IXMLDOMNode> spXMLNode;
//用接口 IXMLDOMDocument 的方法 selectSingleNode 方法定位该节点
hr = spXMLDOM->selectSingleNode(bstrSS, &spXMLNode);
if ( FAILED(hr) ) throw "不能在 XML 节点中定位'xmlnode' ";
if ( spXMLNode.p == NULL ) throw "不能在 XML 节点中定位'xmlnode' ";
//DOM 对象"spXMLNode"现在包含了 XML 节点<xmlnode>,所以我们可以在它下面创建一个子节点并把找到的该节点作为它的父节点。
CComPtr<IXMLDOMNode> spXMLChildNode;
//用接口 IXMLDOMDocument 的方法 createNode 方法创建一个新节点
  hr = spXMLDOM->createNode(CComVariant(NODE_ELEMENT), CComBSTR("xmlchildnode"), NULL, &spXMLChildNode);
if ( FAILED(hr) ) throw "不能创建'xmlchildnode' 节点";
if ( spXMLChildNode.p == NULL )
 throw "不能创建'xmlchildnode' 节点";
//添加新节点到 spXMLNode 节点下去.
CComPtr<IXMLDOMNode> spInsertedNode;
hr = spXMLNode->appendChild(spXMLChildNode, &spInsertedNode);
if ( FAILED(hr) ) throw "不能创建'xmlchildnode' 节点";
if ( spInsertedNode.p == NULL ) throw "不能移动'xmlchildnode' 节点";
//对新节点添加属性.
CComQIPtr<IXMLDOMElement> spXMLChildElement;
spXMLChildElement = spInsertedNode;
if ( spXMLChildElement.p == NULL )
throw "不能在 XML 元素接口中查询到'xmlchildnode' ";
//设置新节点的属性
hr = spXMLChildElement->setAttribute(CComBSTR(L"xml"), CComVariant(L"fun"));
if ( FAILED(hr) ) throw "不能插入新的属性";
//下面的程序段用来寻找一个节点并显示该节点的相关信息
// 查找"xmldata/xmltext"节点
spXMLNode = NULL; // 释放先前的节点
bstrSS = L"xmldata/xmltext";
hr = spXMLDOM->selectSingleNode(bstrSS, &spXMLNode);
if ( FAILED(hr) ) throw "不能定位'xmltext'节点";
if ( spXMLNode.p == NULL ) throw "不能定位'xmltext'节点";
        // 得到该节点包含的文本并显示它
        CComVariant varValue(VT_EMPTY);
hr = spXMLNode->get_nodeTypedValue(&varValue);
if ( FAILED(hr) ) throw "不能提取'xmltext'文本";
if ( varValue.vt == VT_BSTR ) {
// 显示结果,注意这里要把字符串从形式 BSTR 转化为 ANSI
```

```
    USES_CONVERSION;
    LPTSTR lpstrMsg = W2T(varValue.bstrVal);
    std::cout << lpstrMsg << std::endl;
    } // if
    else {
    // 如果出现错误
    throw "不能提取"xmltext"文本";
    } // else
    //保存修改过的 XML 文档到指定的文档名
hr = spXMLDOM->save(CComVariant("updatedxml.xml"));
if ( FAILED(hr) ) throw "不能保存修改过的 XML 文档";
std::cout << "处理完成..." << std::endl << std::endl;
} // try
catch(char* lpstrErr) {
    // 出现错误
std::cout << lpstrErr << std::endl << std::endl;
} // catch
catch(...) {
    // 未知错误
std::cout << "未知错误..." << std::endl << std::endl;
} // catch
// 结束对 COM 的使用
CoUninitialize();
}
```

五、小节

XML 文档因为有着比 HTML 严格得多的语法要求，所以使用和编写一个 XML 解析器要比编写一个 HTML 的解析器要容易的多。同时因为 XML 文档不仅可以标记文档的显示属性，更重要的是它标记了文档的结构和包含信息的特征，所以我们可以方便地通过 XML 解析器来获取特定节点的信息并加以显示或修改，方便了用户对 XML 文档的操作和维护。同时我们需要注意的是 XML 是一种开放的结构体系并不依赖于任何一家公司，所以开发基于 XML 的应用必然会得到绝大多数软件开发平台的支持。另外，我们也可以看到，象微软这样的软件开发主流企业也把目光定位在基于 XML + COM 的体系上，无论是微软的 OFFICE 系列、Web 服务器和浏览器还是数据库产品 SQL SERVER 都已经开始支持基于 XML 的应用。通过 XML 来定制应用程序的前端，COM 来实现具体的业务对象和数据库对象，使系统具有更加灵活的扩展性和维护性。

参考文献

1. MSDN 2000，Microsoft Company
2. David Burdett Donald E. Eastlake III Marcus Goncalves. Internet Open Trading Protocol McGraw - Hill. 北京：人民邮电出版社 2000
3. Simon St. Laurent. XML A Primer. 北京：电子工业出版社 2000
4. http //www.w3.org
5. http //www.microsoft.com
6. http //www.wdj.com
7. http //www.msj.com

收稿日期：2000 年 12 月 6 日）

用 Notes 实现基于工作流程的通用办公自动化系统

任　远　张尚玉　张仲义

摘　要　本文阐述了通用办公自动化系统 UOAS 的设计思想。并以工作流程为基础详细论述了通用办公自动化系统的组成结构。

关键字　工作流程,通用办公自动化系统,协同工作

办公自动化系统是实现企业内部之间以及内外部之间办公信息的收集与处理、流动与共享、实现科学决策具有战略意义的信息系统。它的总体目标是：以先进成熟的计算机和通信技术为主要手段，建成一个覆盖办公部门的办公信息系统，提供与其他专用计算机网络之间的信息交换，建立高质量、高效率的信息网络，为领导决策和办公提供服务，实现办公现代化、信息资源化、传输网络化和决策科学化。下面以我们开发的通用办公自动化系统（UOAS）为例进行说明。

一、Lotus Notes 与 UOAS 的关系

Lotus Notes，是目前全球最流行的“群件”产品，其所具有的众多强项是创建工作流程的理想平台，但本身并非是工作流程产品。Notes 提供了成熟、复杂的数据复制服务和 VIM 传输协议所形成的应用集成标准；并提供各种外部数据库接口，如 ODBC、Notes Pump 等。其 RTF 域又能帮助管理各种信息类型（文字处理、电子表格、影像、图形等）；Notes 还可在所有市场上流行的硬件平台中运行，包括主机、小型机、UNIX 工作站与微机；其应用开发的环境亦被公认为是最简易的软件平台之一。所有这些都为工作流程的解决方案提供了基本的要素。但就本质而言，用 Notes 来支持有效工作流程的应用环境，需要开发人员投入大量的精力。尽管 Notes 在编程方面的灵活性能缩短开发新应用系统的周期，但有经验的开发人员都会感觉到，要维护修改已有的 Notes 应用系统是相当沉重的负担。在现实生活环境中，企事业的组织结构可能会由于新的机遇或新的主管任命而有所改变，员工的工作职责可能会由于提升或调动而重新分派，工作过程可能会由于新的法规或科技发展而重新定义，所有这些都需要原有的应用流程迅速作出相应的调整。UOAS 顺应这种需求，为 Notes 用户提供了一个有效的工作管理环境。

UOAS，是在 Lotus Notes 平台上长时期开发各类关键性业务办公自动化应用系统而演化出来的经验成果。它主要由三部分组成：Designer、Manager 和 Desktop 等。UOAS 提供一个工作流生成器（Designer），让用户定义其工作流程，提供一个客户平台（Desktop），把文档按照所定义的规则进行流转，无需编程，使 Notes 在实现工作流程的机制和环境获得大幅度的延伸：提供一个管理工具（Manager），获取有关详尽的跟踪、监控及统计资料，帮助用户有效地检查工作上的瓶颈，从而作出适当的调整，为 Lotus Notes 平台建立完整工作管理模型提供进一步的增值。

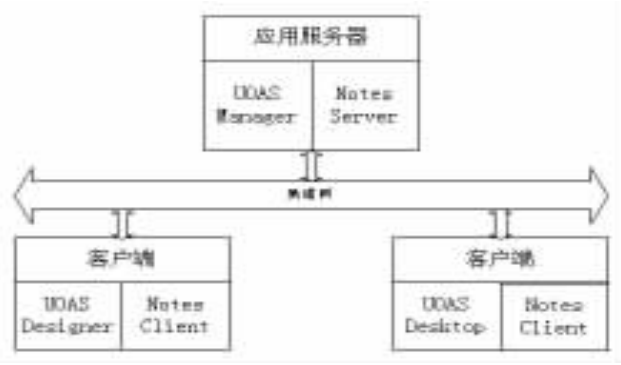


图 1

二、UOAS 以过程管理为导向

UOAS 所面对的重要对象是那些以 IRACIS（Increase Revenue：增加收入，Avoid Cost：减低成本，Improve Services：提高服务）作为最终业务目标的实质性工作管理应用，而不仅仅是工作流程技术的应用。是面对压力需要达到越来越高的效率和服务标准的企事业机构和政府机关，实现目标管理、政务分开和科学决策的重要技术手段。UOAS 是以处理过程为主导的工具。它可被视为把用户所需要的不同的应用系统、企业数据、以及采集或调用的文档等结合在一起的集成元素。它提供丰富的在办事项信息，告知用户瓶颈地带将会在哪里出现，用户可在日程安排上增加资源的投入，或基于预测对处理的环节作适当的调整，从而避免处理过程出现瓶颈。根据历史活动纪录的统计分析结果，用户可改变工作的规则，优化业务处理的模型。所有这些都对于工作规则的自动化与业务运作方式的指引提供了基本的能量。

三、过程优化与 UOAS

以提高生产效率与服务质量为核心，过程优化一般可分为以下几个阶段：

业务目标的定义是领导根据业务环境需要而议定的决策，这是完全依赖人的智能与经验，UOAS 并不能提供多大的帮助。有了业务目标，接下来是对处理过程的分析，此阶段

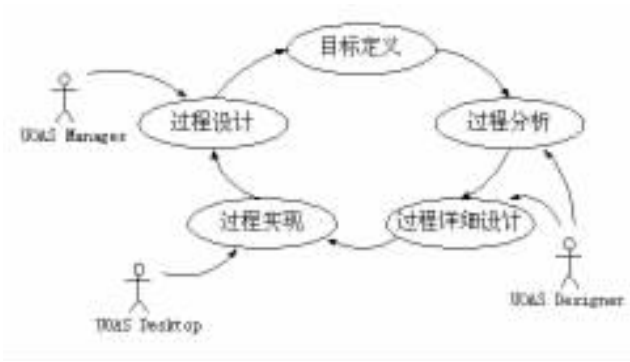


图 2

主要仍是倚赖人的智能与经验，但 UOAS 提供了可视化的过程定义环境，在反复研究分析的过程中提供了直观及易于改动的工具。分析的结果可能引至处理过程需要重新设计 再工程化 ，UOAS Designer 在此大大发挥了其用途。它不仅是一个直观的画图工具，还能把用户设计的流程正式生成。一经提交，过程能马上实现，毋需另作编程。过程处理的驱动由 UOAS 服务器负责，所有当前的任务会呈现在执行者的 UOAS Desktop 的个人工作台上，用户只需安坐办公桌，按照每一任务的提示及轻重缓急逐项处理。若用户配有查看功能，当执行某一任务时，可随时按‘查看流程”的按钮，查看当前任务的图形过程定义，当前任务是哪一个，要处理的事项是从哪一个任务传送过来的，往下的传送路径又是什么等。当处理过程实现后，UOAS 服务器端会记录此运行过程的所有有关活动，用户可利用 UOAS Manager 跟踪某事项的处理状态，监控某过程、任务、角色、参与者的工作量及逾时情况，发出催办单，并执行各种活动的统计，作为辅助决策的科学依据。领导根据统计结果，在修改业务处理目标时，能掌握真实情况，减少决策失误的可能性。

总之，UOAS 帮助企业或政府实现业务处理过程自动化，并有能力与不同的应用系统集合，同时提供广泛的信息让用户掌握业务处理的实际情况，增强管理的能力。综合起来，令用户能真正有效地优化业务处理的过程。

四、UOAS 的工作管理概念

UOAS 的工作管理概念是基于工作人员，处理过程与处理事项相互的影响与效用。参与者 参与 UOAS 工作流程的用户 以角色来分类。角色类似于工作岗位。每一任务并不直接

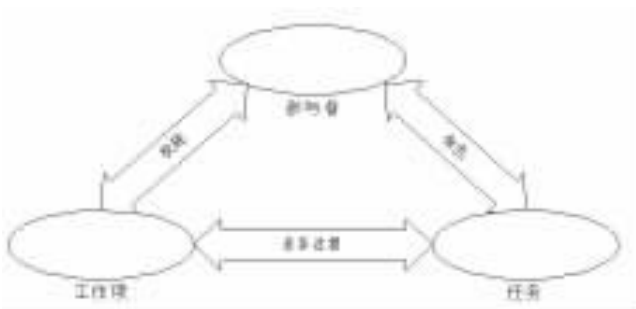


图 3 个人工作台

分配给参与者，而是分配给某一特定的角色。在人事变动的情况下，若某任务仍然存在，角色也会维持不变，只需要把需要变更的人员配以相应的角色便能反映人事变动的调整，毋需修改程序。业务过程由一系列的任务组成，而工作项则是业务过程的其中一宗事件。例如，一封人民来信便是人民来信管理过程中的其中一个工作项。参与者通过个人工作台处理所有传交给他的工作项。参与者在赋予的权限下，可以对业务过程中的所有工作项进行跟踪，监控及催办。工作项是业务过程需要处理的一宗事务，其本质可能是任何格式类型的对象，包括文稿、影像、图纸、电子表格，甚至于多媒体文件等等。参与者与角色在数据库角度来看，属于一对多或多对多的关系。每一参与者都可担当一个或多个角色，例如：办公室秘书可同时担当文件核稿人、领导日程安排人等角色。而每一角色亦可以由一个或多个参与者担当，例如：打字员可以同时由多名员工负责。UOAS 的角色由组别及角色二个层次组成，使角色具有部门的概念。在规模比较大的机构，很多部门内都有类似的岗位，角色加上组别分类更有效处理流程的设计。任务是相应的角色需要执行的事情，一个任务只能配以一个角色，但一个角色可同时担当多个任务。例如：起草发文是一个任务，整理发文样稿又是另一个任务，但一般是业务经办人同时担当这个角色。任务分类得越精细，系统的灵活性越高。例如：某局领导在当时分管四项审批工作，但因为组织机构变动，改为担当其中两项。若只用一个任务处理所有四项审批工作，在体制变动时所牵涉的改动相对大了很多。当任务完成后，流程的定义决定下一个任务是什么。UOAS 支持多种的流向，包括了直流、分流、并流、辅流、会流、子流等等。业务过程是某一类重复性活动的整体处理过程。它由一系列的任务与流程组合而成。每一组业务过程都有一个起始点及一个或多个结束点。起草文稿可以是一个流程的起始点，扫描来文也可是一个起始点，签发后归档可以是一个流程的结束点，不同意签发转交回经办人归档也可以是另一个结束点。所有业务过程处理过的活动状态都存有纪录，作为跟踪、监控与统计用途。个人工作台排列的工作项来自所有相关的业务过程。所有当时需要该参与者处理的工作项都按预定的次序排列在个人工作台上。整个概念是用户不用撤换任何屏幕，只需安坐办公桌上逐项执行所有的任务。参与者若被赋予特定的权限，可以对业务过程的活动进行跟踪、监控及统计。UOAS 记录下所有活动的详细内容，包括任务、角色、签收时间、完成时间、执行人、发送人等等。目的是让用户能实时知道某一事务处理的状态，并提供统计数据，让领导清楚业务运行环境的真实情况，作为进一步提高效率与管理的依据。

五、UOAS Manager

UOAS Manager 提供给有权限的用户对流程进行跟踪、监控与统计，是工作管理概念重要的环节。流程跟踪 用户能随时随地查询某工作项的处理状态，在哪一参与者手里，停留了

跨越域的 Cookie

赵金东

所有的网站开发者都会非常喜欢 cookie 的强大特性和易用性，它在跟踪用户信息，建设人性化、个性化的网站方面，有着强大的作用，而且，又避免了使用数据库的昂贵开销。但是，cookie 却不能跨越域传递，只有那些创建它的域才能访问；这里，我们讨论如何利用 ASP 突破这个限制。

Cookie 简介

首先，我们对 Cookie 做一个简单的介绍，说明如何利用 ASP 来维护 cookie。

Cookie 是存储在客户端计算机中的一个小文件，这就意味着每当一个用户访问你的站点，你就可以秘密地在它的硬盘上放置一个包含有关信息的文件。这个文件几乎可以包含任何你打算设置的信息，包括用户信息、站点状态等等。这样的话，就有一个潜在的危险：这些信息有可能被黑客读取。为了防止这个问题的发生，一个有效的办法就是 cookie 只能被创建它的域所存取。这就是说：比如 ytu. edu. cn 只能访问 ytu. edu. cn 创建的 cookie。通常来讲，这没有什么问题；但是，如果需要两个不同域上的两个不同站点共享保存在 cooie 中的用户信息，该如何处理呢？当然可以选择复制用户信息，但是，如果你需要用户只能在一个站点上注册，并且自动成为另外一个站点的注册用户呢？或者，两个站点共享一个用户数据库，而又需要用户自动登录呢？这时候，跨越域共享 cookie 是最好的解决方案。

这里，先看一些 ASP 处理 cookie 的代码，以便以后便于引用、参考。

'创建 cookie
```
Response. Cookies(″MyCookie″). Expires = Date + 365
Response. Cookies(″MyCookie″). Domain =″mydomain. com″
Response. Cookies(″MyCookie″)(″Username″) =strUsername
Response. Cookies(″MyCookie″)(″Password″) =strPassword
```

读写 cookie 非常简单。上面的代码创建一个 cookie 并给 cookie 设置属性：域、过期时间，以及其他一些保存在 cookie 中的值。这里，strUsername，strPassword 是在前面某个地方设置的变量。然后，通过下面的语句在 cookie 中读取。

'读取 cookie
```
datExpDate = Request. Cookies(″MyCookie″)
strDomain = Request. Cookies(″MyCookie″). Domain
strUsername =Request. Cookies(″MyCookie″)(″Username″)
strPassword =Request. Cookies(″MyCookie″)(″Password″)
```

更详细的信息，可以参考 ASP 的资料。

实现

简单地共享 cookie 的诀窍是重定向，一般过程为：

1. 一个用户点击 siteA. com。

2. 如果用户没有 siteA. com 的 cookie，就把用户重定向到 siteB. com。

3. 如果用户有 siteB. com 的 cookie，把用户连同一个特殊的标志 将在下面解释 重定向回 siteA. com，否则，只把用户重定向到 siteA. com。

4. 在 siteA. com 创建 cookie。

看起来很简单，仔细分析一下：siteA. com 和 siteB. com 共享相同的用户设置，所以，如果用户有 siteB. com 的 cookie

多久等等。流程监控 流程监控主要分为在办事务的监控与已结案事务的查询。在办事务的监控可根据参与者、任务或角色分别查询其中的总体工作项、在办工作项、逾时在办工作项、待办工作项及被催办的工作项。亦可以列出某一工作项所有已进行的活动过程。结案事务的监控可根据参与者、任务或角色分别查询其中的所有工作项及逾时工作项；以参与者或任务查询更可以列出特送的工作项。与在办事务一样，也能列出某一工作项所有已进行的活动过程。此外，在结案事务的监控中可列出所有工作项的处理周期。流程统计 UOAS 提供 15 种标准统计分析报告，主要是对工作项、角色、任务、参与者及过程进行中某一时间段内的工作量统计及比较分析。此外，用户可从系统中取出所有活动的数据，利用 Excel 或其它工具进行特定的统计，加深了解活动的情况。

六、小结

此应用系统是一个相当实用的行政办公系统，内容包括流程及非流程的应用模块，很适合国内一般的行政办公环境。用户只需略作本地化工作，便能实现行政办公自动化，缩短开发的周期，加快实现无纸办公的理想。

参考文献

1 Kuln F A. Bukhres E. general Purpose Workflow Language. Distributed and Paralle Database 1995 3 2 98

2 鲍敢峰、朱鹏、尤晋元．工作流管理技术．计算机科学，1998 25 6 30

3 Data Engineering. Specila Issue on Workflow System 1995 18 1 89

收稿日期：2000 年 12 月 18 日）

已经注册），siteA. com 能够同样读取 cookie、提供 cookie 所允许的特性。这样，访问 siteA. com 的用户就如同访问了 siteB. com。

这个检查的环节应该在 siteA. com 中的文件所包含一个 cookies. inc 中实现。让我们看一下这段代码：

1 - 1

```
´SiteA. com
´检查 cookie
If Request. Querystring(″Checked″) <> ″True″ then
If not Request. Cookies(″SiteA_Cookie″). Haskeys then
´重定向到 siteB. com
Response. Redirect(″http: //www. siteB. com/cookie. asp″)
    End if
End if
```

如果用户有一个 siteA. com 的 cookie，则不需要做任何事情了；第一个 if 语句用来消除无限的循环。让我们看一下 siteB. com 上的 cookie. asp 文件来获得进一步的理解。

1 - 2

```
´SiteB. com
´检查 cookie
If not Request. Cookies(″SiteB_Cookie″). Haskeys then
´重定向到 siteA. com
Response. Redirect(″http: //www. siteA. com/index. asp″ _
& ″?checked =True″)
Else
´获取 username
strUsername = Request. Cookies(″SiteB_Cookie″)(″User-
name″)
´将用户连同一个特殊的标志返回到 siteA. com
Response. Redirect(″http: //www. siteA. com/index. asp″ _
    & ″?checked =True& identifier = & strUsername)
End if
```

如果用户在 siteB. com 上仍没有 cookie，于是，将他送回到 siteA. com，并且通过在查询语句中提供一个叫做 “checked ”的参数让应用程序知道你已经检查过 cookie 了。否则，将用户送回到 siteB. com，并退出循环。

然而，如果用户拥有 siteB. com 的 cookie，我们需要将用户送回 siteA. com 并告诉 siteA. com。为此，我们在数据库中附加一个唯一的标志，username。所以，我们扩展 siteA. com 中的代码。

1 - 3

```
´SiteA. com
. . .
. . .
´检查标志
If Request. Querystring(″identifier″) <> ″″ then
strUsername = Request. Querystring(″identifier″)
    ´记录到数据库
Response. Cookies(″SiteA_Cookie″). Expires = Date + 365
Response. Cookies(″SiteA_Cookie″). Domain = ″siteA. com″
Response. Cookies(″SiteA_Cookie″)(″Username″) =
strUsername
End if
```

最后，我们回到 siteA. com。文件的第一部分 1 - 1 检查是否完成了 cookie 的检查，由于可以明显地知道已经完成（由语句中的 “checked ”参数表明），进行到 1 - 3 所示的程序的第二部分。如果存在特殊的标志，我们就可以在 siteA. com 创建 cookie。使用这个特殊的标志（在这里是 username），我们可以在任何需要的时候查询数据库。然后，设置 cookie，显示页面的其他部分。如果没有指定的标志，也没必要担心，只要简单地显示页面的余下部分。

这样，毫不费力地，siteA. com 拥有了和 siteB. com 一样的 cookie。我们可以传输更多的信息而不只是一个标志，并且，将网络流量控制在最小范围内。

要注意一点，即使用户拥有 siteA. com 上的 cookie，仍需要检查 siteB. com。通常来讲，这不是必须的，也会节约时间。但是，一旦用户在 siteB. com 更改个人信息？这样做，会保持所有信息的同步。

Cookie 环

要完成这些，我们需要两个文件：一个在原始站点服务器（siteA. com），完成检查；一个在参考服务器（siteB. com），验证用户。如果有一台参考服务器包含有需要的所有用户信息或 cookie，就可以增加随意多的原始服务器，所需要做的就是在所有要共享 cookie 的服务器上增加 cookie. inc 文件。

也可以以相反的次序执行，例如，如果 siteB. com 是原始服务器，而 siteA. com 包含用户信息。访问过 siteA. com 却从未访问过 siteB. com 的用户也可以登录到 siteB. com，并且拥有所有的曾经的设置。注意，如果拥有多个参考服务器，这样将会很使人迷惑，并且消耗过多的资源：因为必须将用户重定向到每一台参考服务器。

理论上讲，可以拥有一个所有站点都共享相同的用户的网络。最可行的方案就是建立共享 cookie 环。将参考服务器列表存储在一个地方（备份服务器），以便每个参考服务器可以查找并决定重定向用户的下一个站点。记住一定要通过查询字符串的意思跟踪用户是在哪个原始服务器开始。这样，信息的传输非常迅速，这个环节变得越来越可行。

这里还存在一些问题，首先是反应时间。对用户而言，他们最好不知道过程是怎样的。他所需的时间依赖于 siteA. com、siteB. com 之间的连接，有可能会比较长，在实现 cookie 环时可能会更长。

再一个主要问题，就是每一个实现者大都会面对无限的重定向。这有很多原因，例如：用户的浏览器不支持 cookie。这就需要再设计代码来监测用户浏览器的性能。

最好，还需要注意安全问题。如果有些黑客发现了其中的诀窍，他可能会得到 cookie 中的信息。最简单的防范办法就是保护参考服务器，只允许原始服务器访问 cookie. asp 文件。

让应用程序听懂你的话

—— 一个简单的语音应用程序

朱 杰

摘 要 Microsoft 公司最近推出了一套全新的语音识别系统 Microsoft Speech API5. 0。该系统采用了 COM 体系结构，全面支持中文语音的输入和合成。本文将主要介绍 SAPI5. 0 的主要体系结构，并通过一个简单的语音应用程序介绍 SAPI5. 0 的实现方法。

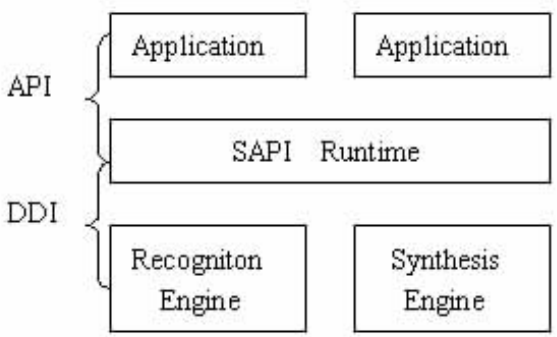
关键词 语音识别，语音合成，COM

前言

众所周知，COM Componet Object Model 作为微软的一种新型的软件体系结构，由于其可扩展性，可重用性和可再开发性等开发优势，越来越多的为人们所应用。微软在开发 Microsoft Speech SDK5. 0 中也采用了 COM 体系标准，因此为开发人员进行二次开发提供了更方便、更快捷的途径，从而大大节省了系统开发的时间，提高了系统的可靠性。

体系结构

Microsoft Speech SDK5. 0 中包含 Speech 应用程序接口 SAPI 微软的连续语音识别 MCSR 引擎、微软的连续语音合成引擎 以及一系列的用于语音开发的开发工具。对于开发人员来说我们主要需要了解的是 SAPI 5. 0 的 COM API， 下图是 SAPI 的体系结构。



语音识别中主要的 COM 接口

● ISpRecognizer 接口：这个接口对象主要用于访问语音引擎。该接口对象有两种实现方式，一种是实现共享的引擎 Shared Recognizer ，另一种是实现独占的引擎 InProc Recognize 。用前种实现方式实现的引擎对象可以在不同的应用程序中使用，而后一种实现方式实现的引擎对象只能在本应用程序中使用，其他应用程序中不能使用。此外通过该接口应用程序可以选择不同语言的引擎。

● ISpRecoContext 是语音识别中的一个主要接口。该接口主要用于接收和发送与语音消息有关的事件消息，装载和卸载 Grammars，例如任何的对话框窗口，菜单项都可以有自己的 RecoContext，但在对应窗口中都必须实现一个处理事件和消息的窗口函数。换句话说，每一个 RecoContext 都对应一个消息处理函数。一般的来说一个应用程序必须实现一个 RecoContext 对象，当然在一个应用程序中也可以实现多个 RecoContext 对象。

● IspRecoGrammar 接口：该接口主要定义了引擎需要识别什么？用于建立、载入和激活语法和规则 Grammar and Rule ，它通常表明什么类型的单词用来识别。通常有两种类型的语法 Grammar 。一种是听写语法 Dictation ，另一种是命令和控制语法 Command and Control 。听写语法允许你使引擎词典中的大量的短语，而命令和控制语法是一些自定义的非常有限的短语用来在应用程序中使用。引擎会从规则中定义的词中来最大匹配用户通过音频输入设备所说的命令。

语法结构

SAPI5. 0 语音识别的命令和控制语法格式是建立在 XML 框架基础之上。可以通过任何的文本编辑器，写我们所需要的规则，需要注意的是必须存成 Unicode 形式。例如 < DEFINE> 用来定义规则的 ID 号。每个规则都有自己唯一的 ID 号。

结束语

对于创建对用户友好的站点来说，通过共享 cookie 来共享用户数据是一种很好的办法。如今大多的站点都使用各自站点所保存的用户信息，这就致使用户不得不一遍又一遍地注册自己的信息。如果多个站点可以共享相同的用户信息，则可以大大地节约用户的时间和金钱。

收稿日期：2000 年 12 月 19 日）

实现语音应用程序的步骤

第一步：初始化 COM。在使用 SAPI5. 0 的 COM API 函数前，确信 COM 在整个应用程序的执行全过程中存在，并且已被激活。通过使用 COM 命令 CoInitialize 和 CoUninitialize 这些命令来实现 COM 初始化。

第二步：建立一个 Recognizer 对象。通过 CoCreateInstance 函数建立一个该 COM 对象。

第三步：建立一个 Recognition Context 对象。本文章中的例子只使用了一个 RecoContext 对象。通过 Recognizer 对象的 CreateRecoContext 函数建立。

第四步：设置是应用程序中的事件通知机制和消息事件，通过调用 SetNotifyWindowMessage 方法来表明与 RecoContext 相关的事件将被送到哪个窗口中，通过调用 SetInterest 方法来表明应用程序只关心某类特定的消息。注意在本文例子中我们只关心 SPEI_RECOGNITION 正确识别用户输入 和 SPEI_FALSE_RECOGNITION 不能识别用户输入 消息。

第五步：通过文本编辑器写我们所需的语法规则用于应用程序中的识别。引擎会从规则中定义的词中来最大匹配用户通过音频输入设备所说的命令。

第六步 :装载规则 Rule 。通过 RecoContext 对象的 CreateGrammar 函数建立一个 Rule 对象，再调用 LoadCmdFromFile 函数从 XML 文件中装载对应的 Rule。

第七步：将规则对象置成激活状态，只有在这种状态下，语音应用程序才开始接收音频输入并进行识别。

程序清单

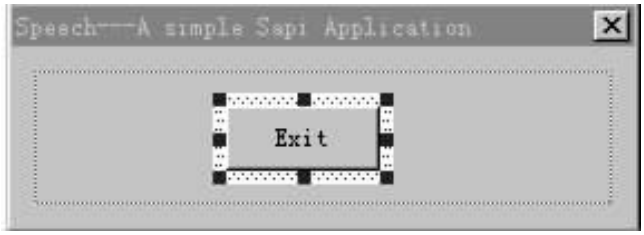
接下来我们通过一个简单的应用程序来实现基本功能。当应用程序工作时，在任何时候只要用户通过话筒说 “关闭程序 ”，该应用程序会自动关闭，而用户所说的其它话，应用程序均不做任何动作。

为了以后说明清楚我们不妨先用文本编辑器写一段规则，并存为名为 CmdCtrl. xml 的文件。在这段规则中我们定义了 “关闭程序 ” 这个词供引擎使用来识别用户所说的命令。

```
<GRAMMAR LANGID = ″804″>  //中文的 Language ID
<DEFINE>                  //定义一个 Rule ID 号
<ID NAME = ″CMD″ VAL = ″10″/>
</DEFINE>
< RULE NAME = ″COMMAND″ ID = ″CMD″ TOPLEVEL =
″ACTIVE″>//定义一个 Rule
<L> //用户自定义词的列表
<P>关闭程序 </P>  //定义一个词
</L>
</RULE>
</GRAMMAR>
```

用户可以任意在该规则中的 <L>. . . </L> 范围内加入你想定义的词作为引擎识别的词典。

接下来我们用 Visual C + + 6. 0 提供的 MFC APPWizard 生成一个名为 Speech 的对话框工程。Wizard 自动为我们生成一个 CSpeechApp 应用程序 和一个 CSpeechDlg 对话框，我们将对话框资源改成如下的界面。



在该对话框中我们加入一个 Button 控件 ID 为 IDC_EXIT，Caption 为 Exit。

按照前面的步骤我们首先在 Speech. CPP 文件中加入以下的代码。

```
BOOL CSpeechApp: : InitInstance()
{
. . . . . .
// Initializes the COM library on the current thread
        if (!SUCCEEDED(: : CoInitialize(NULL)))
  {
  exit(0);   //if failed, Exit this App
  }
CSpeechDlg dlg;
  m_pMainWnd = & dlg;
  int nResponse = dlg. DoModal();
  if (nResponse = = IDOK)
  {
  }
  else if (nResponse = = IDCANCEL)
  {
  }
// Closes the COM library on the current thread
  : : CoUninitialize();
. . . . . .
}
```

通过以上步骤我们实现了初始化 COM 库的工作，接下来我们就要实现与引擎相关的工作了。为了方便起见，我们在该工程中加入一个 CSREngine 类，用该类来实现与语音识别引擎相关的工作。

在 SREngine. h 中，加入以下代码：

```
//ATL 库头文件
#include <atlbase. h>
//SAPI5. 0 头文件
#include <sapi. h>
#include <sphelper. h>
//定义一个资源 ID 号, 这是 Commad and Control Grammar
的 ID 号
#define GID_SRCMD_CN  1234
class CSREngine
{
public:
   //Speech Variable
```

```
    CComPtr <ISpRecognizer>    m_cpRecognizer;
    CComPtr <ISpRecoContext>    m_cpRecoCtxt;
    CComPtr <ISpRecoGrammar>    m_cpCmdGrammar;
    //Audio Variable
      CComPtr <ISpAudio> m_cpAudio;
      public:
      HRESULT SetRuleState( const WCHAR * pszRuleName, const WCHAR * pszValue, BOOL fActivate );
      HRESULT LoadCmdGrammarFromFile(CString XMLFileName);
      HRESULT InitializeSapi(HWND hWnd, UINT Msg);
}
```

SREngine. cpp **中实现如下**:

```
//这段代码实现了以下功能,建立了一个 Recognizer Object,建立了一个 RecoContext,并设置了事件激励机制. 选择了默认的 Audio Device 作为语音输入设备
HRESULT CSREngine:: InitializeSapi(HWND hWnd, UINT Msg)
{
      HRESULT hr = S_OK;
   //建立了一个独占的识别引擎对
hr = m_cpRecognizer. CoCreateInstance(CLSID_SpInproc – Recognizer );
      if(FAILED(hr))
      {
MessageBox(NULL, " Error Create Recognizer", " Error", MB_OK);
      return hr;
      }
   //建立一个 RecoContext 对象
   //本应用程序中我们只使用一个 RecoContext 对象
      hr = m_cpRecognizer – > CreateRecoContext ( & m_cpRecoCtxt );
      if( FAILED(hr) )
      {
MessageBox(NULL, " Error Create RecoContext", " Error", MB_OK);
      return hr;
    }
// 设置识别的事件机制,与语音相关的消息都已 Msg 消息通知应用程序
      hr = m_cpRecoCtxt – >SetNotifyWindowMessag(hWnd, Msg, 0, 0 );
 // 应用程序关心什么类型的消息,在这里当引擎开始识别用户输入时,我们只关心正确识别和不能识别两个事件
const ULONGLONG ullInterest = SPFEI(SPEI_RECOGNITION) |SPFEI (SPEI_FALSE_RECOGNITION);
hr = m_cpRecoCtxt – >SetInterest(ullInterest, ullInterest);
if (FAILED(hr))
{
MessageBox(NULL, "Error setinterest", "Error", MB_OK);
          return hr;
      }
// 建立默认的音频输入对象(一般为声卡的声音输入设备)
   hr = SpCreateDefaultObjectFromCategoryId ( SPCAT_AUDIOIN, & m_cpAudio);
   if (FAILED(hr))
   {
   MessageBox (NULL, "Create Default Audio Object Error", "Error", MB_OK);
   return hr;
   }
//将上面建立的音频输入对象,作为引擎的音频输入源
    hr = m_cpRecognizer – >SetInput(m_cpAudio, TRUE);
   if (FAILED(hr))
   {
MessageBox(NULL, "SetInput Error", "Error", MB_OK);
   return hr;
   }
return hr;
}
//从一个 XML 文件中加载规则
HRESULT CSREngine:: LoadCmdGrammarFromFile(CString XMLFileName)
{
   HRESULT hr = S_OK ;
   if (!m_cpCmdGrammar)
   {
       //建立一个规则语法对象
   hr = m_cpRecoCtxt – >CreateGrammar(GID_SRCMD_CN, & m_cpCmdGrammar);
         if( FAILED(hr) )
         {
MessageBox(NULL, " Error CreateGrammar", " Error", MB_OK);
         return hr;
   }
   WCHAR wszXMLFile[20] = L"";
   MultiByteToWideChar(CP_ACP, 0, (LPCSTR) XMLFileName, – 1, wszXMLFile, 256);
//将 ANSI 的码转换为 Unicode 码
//从 XML 文件中加载规则
 hr = m_cpCmdGrammar – >LoadCmdFromFile(wszXMLFile, SPLO_DYNAMIC);
         if (FAILED(hr))
         {
MessageBox(NULL, " Error LoadCmdFromFile", " Error", MB_OK);
         return hr;
         }
   }
   return hr;
}
//设置规则的状态.
HRESULT CSREngine:: SetRuleState(const WCHAR * pszRuleName, const WCHAR * pszValue, BOOL fActivate)
{
   HRESULT hr = S_OK ;
```

```
    if (fActivate )
      {  //激活状态
      hr = m_cpCmdGrammar -> SetRuleState( pszRuleName,
NULL, SPRS_ACTIVE);
    }
      else
      {  //非激活状态
      hr = m_cpCmdGrammar -> SetRuleState( pszRuleName,
NULL, SPRS_INACTIVE);
      }
   return hr;
}
```

接下来我们在 CSpeechDlg 对话框类 SpeechDlg. h 中增加成员变量，并定义一个消息常量 WM_RECOEVENT，用于通知应用程序同语音识别有关的消息事件。

```
#define WM_RECOEVENT WM_USER +100
class CSpeechDlg : public CDialog
{
Public:
   CSREngine m_SREngine;
   . . . . . .
   afx_msg void OnRecoEvent(WPARAM wParam, LPARAM
lParam);
}
```

在 SpeechDlg. cpp 中我们加入如下代码:

```
BEGIN_MESSAGE_MAP(CSpeechDlg, CDialog)
{
   ON_MESSAGE(WM_RECOEVENT, OnRecoEvent)
}
```

通过以上途径我们实现了消息接收机制。

```
BOOL CSpeechDlg: : OnInitDialog()
{
   . . . . . .
   // TODO: Add extra initialization here
   //初始化语音识别引擎对象
HRESULT      hr = m_SREngine. InitializeSapi(this -> GetSafe-
Hwnd(), WM_RECOEVENT);
      if (FAILED(hr))
      {
            return FALSE;
      }
   //将名为 CmdCtrl. xml 的规则加入
        hr = m_SREngine. LoadCmdGrammarFromFile( ″CmdC-
trl. xml″);
      if (FAILED(hr))
      {
return FALSE;
      }
   //激活该规则，只有激活该规则，引擎才开始工作，识别用户的音频输入
      hr = m_SREngine. SetRuleState(NULL, NULL, TRUE);
      if (FAILED(hr))
      {
            return FALSE;
   }
}
```

设置消息机制，当用户有声音输入时会产生一些特定的消息。

```
void CSpeechDlg: : OnRecoEvent(WPARAM wParam,
LPARAM lParam)
{
   USES_CONVERSION;
   CSpEvent event;
    HRESULT hr = S_OK;
    if (m_SREngine. m_cpRecoCtxt)
    {
while(event. GetFrom(m_SREngine. m_cpRecoCtxt) = = _OK)
{
 //获得事件的 ID
 switch (event. eEventId)
 {
 case SPEI_FALSE_RECOGNITION:
 //如果没有识别成功，则忽略.
 break;
 case SPEI_RECOGNITION:
    {
  // 成功识别
 CComPtr < ISpRecoResult> cpResult;
 CSpDynamicString dstrText;
 CString strResult;
 //获得识别结果
   cpResult = event. RecoResult();
cpResult -> GetText(SP_GETWHOLEPHRASE,
SP_GETWHOLEPHRASE, TRUE, & dstrText, NULL);
 //将结果转换为 CString 类型
   strResult = W2T (dstrText);
    if (! strResult. CompareNoCase( ″关闭程序″) )
    {
  //如果用户输入为关闭程序，则退出应用程序
    OnExit();  //退出应用
   }
 //否则忽略
 }
 break;
 }}}}
```

结论

同时，Microsoft Speech SDK5. 0 **中除了** Speech Recognition **引擎外还包括了语音合成引擎。通过该引擎你可以开发出支持语音合成的应用程序。如果你需要** Microsoft Speech SDK5. 0 **你能从以下的网址免费获得** http / / www. microsoft. com / speech / SpeechSDK / sdk5. asp. **目前，语音识别的应用越来越广，希望通过该文使读者简单建立起语音应用程序的基本概念，为语音技术在实际应用中开辟更广阔的空间。**

收稿日期：2000 **年** 12 **月** 20 **日）**

Delphi 编写浏览器 URL 过滤软件

陈岳林 江天送

一、引言

现在许多家庭和学校都上了互联网，学生在互联网遨游时，可能会有意无意的浏览到一些不健康的网站或网页。如何阻止学生访问一些不健康的网页呢 前些时候看到杂志上有教编写这方面的软件，只能做到监视用户浏览过的网页的标题，比较消极，用处不大。现介绍做的是浏览器的 URL 过滤软件，可以设置一些关键字如 Sex、xxx、gril，只要用户浏览页面的 URL 包含有关键字的字符串，就重新定位 URL 到警告的页面上。这样既警告用户，又阻止了用户访问禁止页面。

二、URL 过滤的实现方法

1. 获得浏览器 URL 地址

我们常用的浏览器都有一个地址栏，可以给你输入 URL 地址，同时它会随你切换页面而改变，即使隐藏了地址栏，它还是会随页面变化。说穿了地址栏就是注册的类名为 "Edit" 的单行的文本框 Edit Controls），我们只要知道地址文本框的句柄，可以向它发 WM_GETTEXT 的消息，其参数如下：

WM_GETTEXT

wParam / / 指定最大的拷贝字符数

lParam = / / 指向返回文本的地址

即可获得它的文本，即当前的浏览器的 URL。随着用户浏览网页，地址栏是不断改变的，我们可以用 WM_GETTEXT 定时获取 URL 字符串。

那么怎样找到浏览器 URL 文本框的句柄呢？我们可以用 GetForegroundWindow 获得当前的工作窗体， 非当前工作窗体的浏览器不是用户正在浏览的，可以忽略）判断它是否是浏览器窗体，IE 可以判断它的类名，Netscape 和 Opera 可以直接判断其标题 具体看源程）。之后可以用 FindwindowEx 找 URL 文本框的句柄，由于它和浏览器主窗体的子父关系比较复杂，若用 FindwindowEx 来找会变得很麻烦且效率不高，可以用 EnumChildWindows 枚举浏览器的所有子窗体，找到类名为 "Edit" 即是 URL 文本框。

2. 重新定位浏览器的 URL

获得了浏览器的 URL 地址，只能做到监视用户浏览网页的情况，而不能禁止用户浏览某些页面。当然可在用户浏览禁止访问的网页时，弹出对话框警告或直接关掉浏览器。更好的方法是把浏览器的 URL 地址重新定位到指定的网页上，网页上可以显示警告的语句等，这样一来既警告了用户，又阻止了用户访问禁止页面。

具体的做法也很简单，只要用给 URL 的文本框发 WM_SETTEXT 的消息，改变其文本，之后再向其发 WM_KEYDOWN 和 WM_KEYUP 的消息 注意不能用 SendMessage，而要用 PostMessage 模拟键盘输入回车，即可把浏览器定位到警告网页上。

演示程序的窗体及源代码：


```
unit Unit2;
interface
uses
  Windows, Messages, SysUtils, Classes, Graphics,
  Controls, Forms, Dialogs, StdCtrls, ExtCtrls;
type
  TForm2 = class(TForm)
    Filterkeys: TListBox;
    NewUrl: TEdit;
    Label1: TLabel;
    Label2: TLabel;
    Baddkey: TButton;
    Bdelkey: TButton;
    Timer1: TTimer;
    Button3: TButton;
    Bsavetofile: TButton;
    Bloadfromfile: TButton;
    OpenDialog1: TOpenDialog;
    SaveDialog1: TSaveDialog;
    procedure Timer1Timer(Sender: TObject);
    procedure FormCreate(Sender: TObject);
```

```
    procedure BaddkeyClick(Sender: TObject);
    procedure BdelkeyClick(Sender: TObject);
    procedure Button3Click(Sender: TObject);
    procedure BsavetofileClick(Sender: TObject);
    procedure BloadfromfileClick(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
var
  Form2: TForm2;
implementation
{$R *.DFM}
function filter(url: pchar): boolean; //过滤部分
var i: integer;
s: string;
begin
     result:=false;
     s:=lowercase(strpas(url));
     with form2.filterkeys do
     for i:=0 to Items.Count-1 do
      if pos(items[i], s)>0 then
       begin //当 Url 含有过滤关键字返回真
         result:=true;
         exit;
       end;
end;
function EnumChildProc(
    hwnd: HWND ;    // 子窗体的举柄
    lParam: LPARAM  // 用户只定义的数据
   ): bool; stdcall;
var buf: array[0..250] of char;
rsize: integer;
begin
   result:=true;   //返回 True 继续枚举
   Getclassname(hwnd, buf, sizeof(buf));
   //获取子窗体的注册类名
   if strpas(buf)='Edit' then
   //类名为'Edit'即为 URl 所在的 Edit 控件
    begin
     rsize:=sendmessage(hwnd, WM_GETTEXT, sizeof(buf)
      , integer(@buf));   //发送 WM_GETTEXT 获取其文本
     if rsize>0 then
      if strpas(buf)<>form2.NewUrl.text then
       if filter(buf) then
         begin
           //Edit 的文本设为新的 NewUrl.text
            sendmessage(  hwnd,  WM_SETTEXT,  0,  integer
(form2.NewUrl.text));
           //模拟回车键
          postmessage(hwnd, WM_KEYDOWN, $D, $1c0001);
          postmessage(hwnd, WM_KEYUP, $d, $c01c0001);
        end;
      result:=false;   //返回 False 结束枚举
    end;
end;
procedure TForm2.Timer1Timer(Sender: TObject);
 //定时监视控制(时间间隔 1-3 秒)
var fwnd: Thandle;
buf2, buf: array[0..250] of char;
begin
   fwnd:=GetForegroundWindow;
    // 获取当前工作窗口的句柄
   Getclassname(fwnd, buf, sizeof(buf));
    //获取注册类名
   Getwindowtext(fwnd, buf2, sizeof(buf2));
   //获取窗口的标题
   if (strpas(buf)='CabinetWClass') or (strpas(buf)=
'IEFrame')
    //类名为 CabinetWClass 或 IEFrame 为 IE
    or (pos('Netscape', strpas(buf2))>0)
      or (pos('Opera', strpas(buf2))>0) then
     EnumChildWindows(fwnd, @enumchildproc, 0);
      //枚举浏览器窗口的所有子窗体
end;
procedure TForm2.BaddkeyClick(Sender: TObject);
var value: string;
begin
   if InputQuery('Url 过滤演示', '添加关键字', value)
    then  filterkeys.Items.Add(lowercase(value));
end;
procedure TForm2.BdelkeyClick(Sender: TObject);
begin
 if filterkeys.ItemIndex<>-1 then
   filterkeys.Items.Delete(filterkeys.ItemIndex);
end;
procedure TForm2.Button3Click(Sender: TObject);
begin
 form2.WindowState:=wsMinimized ;
end;
procedure TForm2.FormCreate(Sender: TObject);
begin
    NewUrl.Text :=ExtractfilePath(paramstr(0))+'warn-
ing.htm';
end;
procedure TForm2.BsavetofileClick(Sender: TObject);
begin
  if savedialog1.Execute then //保存过滤关键字到文件
    Filterkeys.Items.LoadFromFile(savedialog1.filename);
end;
procedure TForm2.BloadfromfileClick(Sender: TObject);
begin
   if opendialog1.Execute then //从文件加载过滤关键字
    Filterkeys.Items.loadfromfile(opendialog1.filename);
end;
end.
```

收稿日期 2000 年 11 月 28 日

在 Delphi 中利用多线程实现数据采集的方法

贾 静

一、问题的提出

我们知道，在 Windows 环境下如果要实现对硬件的数据采集，往往需要编写设备驱动程序，而设备驱动程序的编写是一项费时费力的工作，并且，Win95、Win98 的驱动程序也不尽相同，需要分别开发。所以对于通信数据量小的系统和一些实时性要求不高的地方，我们希望能有一种更简单的数据采集方法。以查询的方式从 I/O 口数据线读入数据或者采用串口进行通信都是比较简便的方法。

下面我们介绍一种利用 Windows 中的多线程来实现从 I/O 口数据线进行数据查询采集的方法。该方法用 Delphi 编写，适用于数据实时性要求不苛刻场合下使用。其优点是编程调试简单，对硬件的要求也简单。

二、线程的概念

Win32 操作系统中，进程是应用程序的执行实例，每个进程是由私有的虚拟地址空间、代码、数据和其他各种系统资源组成的。进程在运行过程中创建的资源随着进程的终止而被销毁，所以使用的系统资源在进程终止时被释放或者关闭。线程是进程内部的一个执行单元。系统创建好进程后，实际上就启动了执行该进程的主执行线程。主执行线程终止了，进程也就随之终止。

每个进程至少有一个线程，这个线程由系统自动建立。用户根据需要在应用程序中创建其它的线程。Win32 以线程为其分配 CPU 时间的基本实体，时间片的大小为 20ms 级。所以可以在单处理器上“同时”执行多个线程。利用线程的这种特性，我们可以为数据采集建立一个线程，它与主线程“同时”运行，在采集数据的同时，可以处理来自键盘、鼠标的消息。目前计算机的速度越来越快，线程的优点也越来越明显。由于一个进程中所有的线程共享该进程的虚拟地址空间并能访问全局变量和进程的系统资源，这样，线程与进程、其它线程之间的通信非常容易。

三、Delphi 中的 TThread 类

Delphi 为程序员编写线程提供了一个类 TThread，它提供了建立线程、控制线程开始、暂停、结束以及释放等功能。TThread 将 WindowsAPI 函数中关于多线程方面的函数封装到了一起。TThread 是一个抽象类，不可以直接创建它的实例，但是可以通过重载其中的虚函数 Execute，快速地建立自己所需要的线程，大大简化了编程工作。

需要注意的是，由于 Delphi 不支持多线程同时访问可视对象类库（VCL），所以在线程中调用例如更新屏幕这样的操作时，需要使用特殊的方法来解决这个问题。一种方法是使用函数 Synchronize。其原型如下：

```
type TThreadMethod = procedure of object
procedure Synchronize Method TThreadMethod
```

其中的参数 Method 是一个不带参数名的过程名称。

程序运行期间的具体过程实际上是由 Synchronize 过程来通知主线程，然后主线程在适当的时机执行 Synchronize 过程中参数所指定的过程。在多个线程的情况下，主线程将 Synchronize 过程所发出的通知放到消息队列中，然后逐个地响应这些消息。通过这种机制 Synchronize 实现了线程之间的同步。

程序中，每次采集一定数量的数据，如果需要在屏幕上显示数据的更新情况，就需利用到函数 Synchronize 来实现可视类与线程的同步。

四、访问硬件

Delphi 所提供的内嵌汇编功能（BASM）使用户可以像编写 DOS 程序一样直接对微机底层操作，这使得访问硬件工作变得非常容易。

所谓嵌入汇编，就是直接把 Intel80x86 汇编语言的代码直接写入 Delphi 的语言代码中一起进行编译。Delphi 的嵌入汇编很简单，先键入关键字 asm，编写汇编代码，结束时加上关键字 end 即可。但是有两点需要注意：

（1）在汇编代码中访问应用程序的全局变量以及过程或函数的局部变量时，寄存器的类型与变量的类型必需匹配。386 以上的计算机常用的寄存器有：32 位通用寄存器 EAX、EBX、ECX、EDX；16 位寄存器 AX、BX、CX、DX；8 位寄存器 AH、BH、CH、DH、AL、BL、CL、DL。如果变量为 integer 类型，则应选用 32 位寄存器；如果变量为 Word，则应选用 16 位寄存器；对于 Byte 类型变量，需选用 8 位寄存器。

（2）因为直接使用了寄存器，而操作系统本身也在使用这些寄存器，故两者有可能发生冲突。所以在使用寄存器之前，应该将寄存器的值进行压栈保护，在使用完这些寄存器后，一定不要忘记恢复其原值。

笔者使用内嵌汇编写了关于端口读写的几个函数。它们分别是从端口读字节（inportb）、字（inportw），写字节outportb、字outportw。通过这些函数，可以很方便地读取来自硬件端口的数据。

下面具体给出从端口读取一个字的函数：

```
function inportw(Address: Word): word;
var
 data: word;
begin
asm
push ax
push dx
mov dx, Address
in  ax, dx
mov data, ax
pop dx
pop ax
end;
result: =data;
end;
```

五、具体实例

下面笔者将介绍一个简单的实例，解释和说明前面所述的内容。

1）建立Form，在其中建立两个按钮“开始采集”和“停止采集”。

2）建立一个新的单元，在其中输入以下代码：

```
unit CollectThread;
interface
uses
  Classes, SysUtils, stdctrls;
type
   TCollectDataThread  = class(TThread)
  private
    FData:  double;
    FPreSetTime: Word;
  protected
 procedure Execute;  override;
 procedure DataSmoothProcess;
public
UsedTime: integer;
constructor Create(PresetTime: Word);
end;
implementation
{TCollectDataThread}
constructor TCollectDataThread. Create(PreSetTime: Word);
begin
 FPreSetTime:  =PreSetTime;     // 预置采集次数
 UsedTime:  =0;                          // 已经采集的次数
 FData:  =0;                             // 数据初值
 inherited Create(true);
 FreeOnTerminate : = True; //当线程终止时自动释放该线程
end;
procedure TCollectDataThread. Execute;
var
  flag: Byte;
  address: Byte;
  collectingdata: integer;
begin
  while UsedTime  <  FPresetTime do
  begin
    UsedTime:  =UsedTime +1;
  flag:  =1;
repeat
begin      // read data
flag:  =inportb($341);
flag:  =flag and $c0;
 if (flag =0) then
begin
collectingdata:  =inportw($342);
FData:  =FData +  collectingdata ;
end;
flag:  =flag and $10;
end;    /// end while not flag =0
until(flag =0);
// you can add the function Synchronize(Method: TThread-
Method) here to show the // data in form.
   end;  //while Usedtime <FPresetTime
DoTerminate;        //线程终止
end;
```

3）在Form的OnCreate事件中加入以下代码：

```
CollTestThread:  = TCollectDataThread. Create(100);
```

4）在“开始采集”按钮的OnClick事件中加入以下代码：

```
CollTestThread. Resume;
```

5）在“暂停采集”按钮的OnClick事件中加入以下代码：

```
CollTestThread. Suspend;
```

由于篇幅的限制，本例中省略了数据在窗体中的显示，如果用户需要间隔一定时间显示采集数据情况，可以利用前面所提到的Synchronize函数在TThread中来访问窗体，或者使用定时器定期刷新屏幕。

结束语

采用上面所述的方法，笔者实现了对心电谱数据的采集和显示，达到了很好的效果。本方法尤其适用于那些利用计算机I/O插槽来实现数据采集的硬件设备，串口通信中的采集方法也与上述思想相类似。鉴于笔者的水平有限，欢迎读者批评指正。

（收稿日期：2001年1月10日）

DELPHI 报表的动态生成

段智勇

摘 要 本文主要介绍如何在 DELPHI 开发环境下动态生成报表，给出公用模块，并结合具体示例分析了实现的主要技术问题，给出解决这些问题的技术细节。

关键词 DELPHI 报表，动态创建

一、引言

报表是数据库应用程序的重要部分，可是报表的生成也是数据库开发中最麻烦的一项工作。报表格式复杂多样，一直是使程序员头疼的事。DELPHI 在其 3.0 以后版本中加入了 QUICKREPORT，使这种情形有所改变。它的全部可视化编程以及设计和运行过程中都可以进行预览等特性给程序开发带来了很大的方便。我们可以通过在应用程序设计阶段往窗体中添加报表控件，再与数据库表组件 Ttable，查询组件 Tquery 等数据访问组件相关联，从而实现报表的预览与报表的输出。但在许多应用程序中，程序员往往期望从静态或者动态生成的数据库表中动态提取所需数据，进而生成报表输出。本文介绍的就是如何动态选择所需数据，动态生成报表的公用模块。这种动态方式生成的报表更加灵活，也更加容易做到报表的格式统一。

动态报表主要是在程序运行阶段利用 Creat 方法动态创建控件，设置其 Parent 属性来设置其容器控件。然后，在程序中根据所选择的数据库表中的数据字段的长度和数目，通过修改控件的大小和位置属性，来控制控件的大小和外观，如果该控件有事件，可直接把函数或过程名赋给它的相应事件名。

二、动态报表的实现

下面结合一个具体实例来说明其如何实现的方法和技术。

1. 基本思路

数据来源，我们以 DELPHI 自带的 DBDEMOS 中的 employee.db 表为例，它共有 6 个字段。

在 F_main 主窗体中 如图 1），可以自由选择所需要打印的字段。它的主要控件及属性设置如下：

① Table1 Databasename 设置为 DEDEMOS Tablename 设置为 employee.db

② Listbox1 显示所连数据库表中的全部字段

③ Listbox2 用于选择所需报表输出的字段

④ AddBitBtn 用于把所选择的字段名添加到 Listbox2 中

⑤ DeleteBitBtn 用于把 Listbox2 中的字段名去掉

⑥ PreviewBitBtn 用于报表的预览

⑦ PrintBitBtn 用于报表的输出

⑧ CloseBitBtn 用于关闭应用程序

在 F_report 窗体中，放置了以下主要控件，并设置属性，以减少程序的篇幅：

① Table1 Databasename 设置为 DEDEMOS Tablename 设置为 employee.db

② QuickRep1：papesize 属性为 A4，dataset 属性为 Table1，bands 属性中的 hascolumnheader、hasdetail、hastitle 设置为 True

图 1

显示数据库表中的全部字段。

在 F_report 的 Oncreate 事件中加入了如下代码：

```
Table1.Open;
if Table1.Active then
 Table1.GetFieldNames(Listbox1.Items); //获得数据库表中的全部字段名
 DeleteBitBtn.Enabled:=False; //在 Listbox2 中无字段时,DeleteBitBtn 变灰
```

2. 从 Listbox1 中选择字段添加到 Listbox2 中为 AddBitBtn 的 Onclick 事件加入如下代码：

```
if listbox1.Items.Count=0 then exit;  //如 Listbox1 中无可供选择的字段,则执行空操作
if listbox1.Selected[listbox1.ItemIndex] then  //在 Listbox1 中选择字段
  begin
Listbox2.Items.Add(Listbox1.Items
[Listbox1.ItemIndex]); //往 Listbox2 中增加选中的字段
Listbox1.Items.Delete(Listbox1.ItemIndex); //从 Listbox1 中删除此字段
if Listbox2.Items.Count>=1 then  //在 Listbox2 中有字段才允许执行删
```

```
  DeleteBitBtn. Enabled: =True;
 end;
```

3. 从 Listbox2 中删除不需要的字段

为 DeleteBitBtn 的 Oncreate 事件添加如下代码：

```
  if Listbox2. Items. Count =0 then exit;  //如果 Listbox2 中无字段,则执行空操作
  if listbox2. Selected[Listbox2. ItemIndex] then
//在 Listbox2 中选择字段
 begin
  Listbox1. Items. Add(Listbox2. items
[Listbox2. itemindex]);  //添加到 Listbox1 中
Listbox2. Items. Delete(Listbox2. itemindex);
//从 Listbox2 中删除此字段
 end;
 if Listbox2. Items. Count =0 then
//如果 Listbox2 中无字段,则 DeleteBitBtn 变灰
 DeleteBitBtn. Enabled: =False;
```

4. 在报表中动态添加一列的步骤

① TitleBand1 中打印的是报表的名称，这里假设为：动态报表生成示例。可以动态创建 TQrlabel 控件，把它的 Parent 属性置为 F_report. TitleBand1，使其成为 TQrlabel 控件的容器控件。

② ColumnHeaderBand1 中需要打印的是报表的列名。为了使报表的格式更加整齐，我们同时动态创建 TQrlabel 控件和 TQRshape 控件，把 F_report. ColumnHeaderBand1 设为它们的容器控件。把 Listbox2 中选择的字段名，赋给 Tqrlabel. caption，从而显示列名。

③ 在 DetailBand1 中创建 TQRDbText 控件与 TQRshape 控件，把它的 Parent 属性指向 F_report. DetailBand1，与在 ColumnHeaderBand1 中创建列名相类似。并使 TQRDbText 控件的 dataset 属性指向相应的 Ttable 或 Tquery 控件，dataField 属性指向对应的字段。

④ 在预览前根据选择字段，判断报表的总宽度是否超出宽度，以及报表的打印方向是横向还是纵向。如总宽度超出报表所限的最大值，则提示警告信息，并强制进行调整。

⑤ 另外需要考虑的问题是如何确定列的宽度，以及各列之间的相对位置。通过 Columnswidth 过程，确定所选择的字段中最大的字段长度 maxwidth。各列的打印宽度可以通过公式

该列宽度 = 字段最大长度 * 给定字体下每字节所占的宽度 + 两边所留空隙　调整。

```
 widthperbyte: =10;  //每个字节对应的像数
 columnswidth;  //计算最大列宽与总宽度
 disposecontrols;  //释放动态创建的控件
 if totalwidth * widthperbyte>1123 then
//判断总宽度是否超出最大值
  begin
Application. MessageBox('报表超宽,请调整再输出!','警告',1);
exit;//提示警告信息,并强制进行调整
  end
 else if totalwidth * widthperbyte>794 then
F_report. QuickRep1. Page. Orientation: =polandscape
//横向
 else
F_report. QuickRep1. Page. Orientation: =poPortrait;
//纵向
Heading: =TQRlabel. Create(self);  //创建 TQRlabel 控件
Heading. parent: =F_report. TitleBand1;  //设置其容器控件
Heading. Caption: ='动态报表生成示例';//报表标题
Heading. Font. Size: =16;
Heading. Font. Style: =[fsbold];  //粗体字
  Heading. Alignment: =tacenter;
  Heading. Width : = Length ('动态报表生成示例') *
(widthperbyte +4);//标题的宽度
 Heading. Left : = ( F_report. QuickRep1. Width – Heading.
width)div 2;
  Heading. Height: =F_report. TitleBand1. Height –1;
  Heading. Top: =0;
  Leftx: =(F_report. quickrep1. width –totalwidth * width-
perbyte)div 2;
  F_report. QuickRep1. Font. Size: =12;
  for i: =0 to Listbox2. items. count –1 do
//根据所选择字段的数目来动态创建
  begin
  QRShape1: =TQRSHape. Create(self);
//创建 TQRSHape 控件打印列名的格线
QRShape1. parent: =F_report. ColumnHeaderBand1;
//设置容器控件
 QRShape1. Left: =Leftx;  //与列名的格线相对齐
QRShape1. Width: =maxwidth * widthperbyte +4;
// TQRSHape 控件的宽度
QRShape1. Height: =F_report. ColumnHeaderBand1. Height;
QRShape1. top: =0;
QRLabel: =TQRLabel. Create(self);  //创建 TQRLabel 控件
QRLabel. parent: =F_report. ColumnHeaderBand1;
//设置容器控件
 QRLabel. Font. Style: =[fsbold];  //设置列名的字体为粗体字
QRLabel. Left: =Leftx +2;  //左边空 2 个像数
QRLabel. width: =maxwidth * widthperbyte;
//列名的打印宽度
QRLabel. height: =F_report. ColumnHeaderBand1. Height –2;
//空 2 个像数
QRLabel. top: =1;
QRLabel. caption: =Listbox2. Items. Strings[i];
//显示选择的字段名称
QRShape2: =TQRSHAPE. Create(self);
//创建 TQRSHAPE 控件
QRShape2. Parent: =F_report. DetailBand1;  //设置容器控件
QRShape2. Left: =Leftx;
QRShape2. Width: =maxwidth * widthperbyte +4;
          //确定格线的宽度
QRShape2. Height: =F_report. DetailBand1. Height;
QRShape2. top: =0;
QRDBText: =TQRDBText. Create(self);
```

```
//创建 TQRDBText 控件
QRDBText. parent: =F_report. DetailBand1; //设置容器控件
QRDBText. Left: =Leftx +2; //左边空 2 个像数
QRDBText. Width: =maxwidth * widthperbyte;
//设置 QRDBText 的宽度
QRDBText. Height: =F_report. DetailBand1. Height -2;
//空 2 个像数
QRDBText. Top: =1;
QRDBText. DataSet: =F_report. Table1; //连接数据表控件
QRDBText. DataField: =Listbox2. Items. Strings[i];
//连接选择的字段
Leftx: =Leftx +maxwidth * widthperbyte +4;
//设置下一列的起始位置
    end;
  F_report. Table1. Active: =true;
  F_report. QuickRep1. Preview;  //报表的预览
```

5. 为计算报表的总宽度 totalwidth 与最大列宽 maxwidth，定义了过程 columnswidth。各字段的长度可以用 Tfield 的 DataSize 属性得到。字段名的长度根据 Length 函数来获得。为了整个打印表格整齐，通过比较字段名与字段长度来确定最大列宽 maxwidth。

```
maxwidth: =0;
for i: =0 to F_main. Listbox2. items. count -1 do
  begin
if F_main. Table1. Fields. Fieldbyname(F_main. Listbox2.
items. strings[i]). datasize
 >maxwidth then
maxwidth: =F_main. Table1. Fields. Fieldbyname
(F_main. Listbox2. items. strings[i]). datasize; //确定字段中
数据的最大长度
if Length(F_main. Listbox2. items. strings[i])>maxwidth then
maxwidth: =Length(F_main. Listbox2. items. strings[i]);
//确定字段名中最大长度
  end;
  totalwidth: =0;
  for i: =0 to F_main. Listbox2. items. count -1 do
totalwidth: =totalwidth +maxwidth +4; //报表总宽
```

6. 在动态创建完控件后，我们通过过程 disposecontrols 来释放其所占的资源。由于用户可能多次点击预览键，因此我们必须在每次预览事件发生之前，释放上次动态创建的控件。

```
for i: =0 to F_report. TitleBand1. ControlCount -1 DO//取消
系统对控件的控制
F_report. TitleBand1. RemoveControl
(F_report. TitleBand1. Controls[0]);
for  i: =1  to  F_report. ColumnHeaderBand1. ControlCount
DO//取消系统对控件的控制
F_report. ColumnHeaderBand1. RemoveControl
(F_report. ColumnHeaderband1. Controls[0]);
for i: =1 to F_report. detailband1. controlcountDO// 取消系
统对控件的控制
F_report. detailband1. removecontrol
(F_report. detailband1. Controls[0]);
  F_report. Table1. active: =false;
```

三、注意事项

1. 程序员可以根据用户的实际需求，设置表格线或者取消表格线。也可以不计算最大宽度，而根据各列实际宽度打印，则无须定义 columnswidth 过程。各列宽度的计算公式则改为：该列宽度 = 该字段长度 * 给定字体下每字节所占的宽度 + 两边所留空隙。

2. 在添加 Preview 的 OnClick 事件前必须先添加 TQRLabel、TQRShape、TQRDBText 控件的系统标准引用单元 QRCtrls。

3. 在设置纸的打印方向时，必须引用 Printers。

4. 动态生成组件的宽度计算必须放在定义其字体属性完成后进行。

5. 如果要修改 QRDBText 控件的数据位置，必须先设置其 AutoSize 属性为 false，然后才能设置其 Alignment 属性为所需的左对齐、居中或者右对齐。这一点很容易被忽略。

四、结束语

以上程序在 DELPHI 中调试通过。

上述示例只是介绍了报表动态生成的核心部分，由于在不同的实际情况下，用户对报表输出的格式会有所不同，因此需要根据具体情况，更加灵活地运用报表类控件，对上述示例程序进行修改和添加，以满足不同的要求。比如：可以动态地创建数据库表，再通过数据访问组件来获取所需数据。也可以动态创建 TQRExpr 控件来实现动态报表的计算功能等等。

U_main 的程序代码：

```
unit U_main;
interface
uses
Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs, ExtCtrls, StdCtrls, Buttons, Db, DBTables, QRCtrls, printers;
type
  TF_main  = class(TForm)
    ListBox1: TListBox;
    ListBox2: TListBox;
    addBitBtn: TBitBtn;
    deleteBitBtn: TBitBtn;
    Panel1: TPanel;
    Panel2: TPanel;
    previewBitBtn: TBitBtn;
    printBitBtn: TBitBtn;
    closeBitBtn: TBitBtn;
    Table1: TTable;
    procedure FormCreate(Sender: TObject);
    procedure addBitBtnClick(Sender: TObject);
    procedure deleteBitBtnClick(Sender: TObject);
    procedure previewBitBtnClick(Sender: TObject);
```

```
    procedure printBitBtnClick(Sender: TObject);
    procedure closeBitBtnClick(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
var
  F_main: TF_main;
  maxwidth, totalwidth: integer;
  procedure columnswidth;
  procedure disposecontrols;
implementation
uses U_report;
{$R *.DFM}
procedure TF_main.FormCreate(Sender: TObject);
begin
  Table1.Open;
  if Table1.Active then
  Table1.GetFieldNames(Listbox1.Items);//获得数据库表中的全部字段名
  DeleteBitBtn.Enabled:=False; //在 Listbox2 中无字段时,DeleteBitBtn 变灰
end;
procedure TF_main.addBitBtnClick(Sender: TObject);
begin
  if listbox1.Items.Count=0 then exit; //如 Listbox1 中无可供选择的字段,则执行空操作
if listbox1.Selected[listbox1.ItemIndex] then //在 Listbox1 中选择字段
  begin
Listbox2.Items.Add(Listbox1.Items[Listbox1.ItemIndex]);
    Listbox1.Items.Delete(Listbox1.ItemIndex);
    if Listbox2.Items.Count>=1 then //在 Listbox2 中有字段才允许执行删
    DeleteBitBtn.Enabled:=True;
  end;
end;
procedure TF_main.deleteBitBtnClick(Sender: TObject);
begin
  if Listbox2.Items.Count=0 then exit; //如果 Listbox2 中无字段,则执行空操作
   if listbox2.Selected[Listbox2.ItemIndex] then //在 Listbox2 中选择字段
  begin
Listbox1.Items.Add(Listbox2.items[Listbox2.itemindex]);
    Listbox2.Items.Delete(Listbox2.itemindex);
  end;
  if Listbox2.Items.Count=0 then //如果 Listbox2 中无字段,则 DeleteBitBtn 变灰
  DeleteBitBtn.Enabled:=False;
end;
procedure columnswidth;
var i: integer;
begin
  maxwidth:=0;
  for i:=0 to F_main.Listbox2.items.count-1 do
  begin
    //确定字段中数据的最大长度
     if F_main.Table1.Fields.Fieldbyname(F_main.Listbox2.
 items.strings[i]).datasize>maxwidth then
maxwidth:=F_main.Table1.Fields.Fieldbyname
(F_main.Listbox2.items.strings[i]).datasize;
    //确定字段名中最大长度
if Length(F_main.Listbox2.items.strings[i])>maxwidth then
    maxwidth:=Length(F_main.Listbox2.items.strings[i]);
  end;
  totalwidth:=0; //报表总宽
  for i:=0 to F_main.Listbox2.items.count-1 do
  totalwidth:=totalwidth+maxwidth+4;
end;
procedure TF_main.previewBitBtnClick(Sender: TObject);
var
  i: integer;
  leftx: integer;
  widthperbyte: integer;
  Heading: TQRlabel;
  QRLabel: TQRlabel;
  QRshape1: TQRshape;
  QRshape2: TQRshape;
  QRdbtext: TQRDBtext;
begin
  widthperbyte:=10;    //每个字节对应的像数
  columnswidth;        //计算最大列宽与总宽度
  disposecontrols;     //释放动态创建的控件
  if totalwidth*widthperbyte>1123 then
    begin
      Application.MessageBox('报表超宽,请调整再输出!','警告',1); //输出对话框
      exit;
    end
  else if totalwidth*widthperbyte>794 then
F_report.QuickRep1.Page.Orientation:=polandscape //横向
 else
F_report.QuickRep1.Page.Orientation:=poPortrait; //纵向
Heading:=TQRlabel.Create(self);   //创建 TQRlabel 控件
Heading.parent:=F_report.TitleBand1;  //设置其容器控件
Heading.Caption:='动态报表生成示例';     //报表标题
   Heading.Font.Size:=16;
    Heading.Font.Style:=[fsbold];
    Heading.Alignment:=tacenter;
     Heading.Width:=Length('动态报表生成示例')*
(widthperbyte+4);
         Heading.Left:=(F_report.QuickRep1.Width-Heading.width)div 2;
    Heading.Height:=F_report.TitleBand1.Height-1;
    Heading.Top:=0;
     Leftx:=(F_report.quickrep1.width-totalwidth * width-
```

```
perbyte)div 2;
    F_report. QuickRep1. Font. Size: =12;
    for i: =0 to Listbox2. items. count -1do  // 根据所选择字段的数目来动态创建
    begin
      QRShape1: =TQRSHape. Create(self);
      QRShape1. parent: =F_report. ColumnHeaderBand1;
      QRShape1. Left: =Leftx;
      QRShape1. Width: =maxwidth * widthperbyte +4;
QRShape1. Height: =F_report. ColumnHeaderBand1. Height;
      QRShape1. top: =0;
      QRLabel: =TQRLabel. Create(self);
      QRLabel. parent: =F_report. ColumnHeaderBand1;
      QRLabel. Font. Style: =[fsbold];
      QRLabel. Left: =Leftx +2;
      QRLabel. width: =maxwidth * widthperbyte;
QRLabel. height: =F_report. ColumnHeaderBand1. Height -2;
      QRLabel. top: =1;
      QRLabel. caption: =Listbox2. Items. Strings[i];
      QRShape2: =TQRSHAPE. Create(self);
      QRShape2. Parent: =F_report. DetailBand1;
      QRShape2. Left: =Leftx;
      QRShape2. Width: =maxwidth * widthperbyte +4;
      QRShape2. Height: =F_report. DetailBand1. Height;
      QRShape2. top: =0;
      QRDBText: =TQRDBText. Create(self);
      QRDBText. parent: =F_report. DetailBand1;
      QRDBText. Left: =Leftx +2;
      QRDBText. Width: =maxwidth * widthperbyte;
      QRDBText. Height: =F_report. DetailBand1. Height -2;
      QRDBText. Top: =1;
      QRDBText. DataSet: =F_report. Table1;
      QRDBText. DataField: =Listbox2. Items. Strings[i];
      Leftx: =Leftx +maxwidth * widthperbyte +4;
    end;
  F_report. Table1. Active: =true;
  F_report. QuickRep1. Preview;
end;
procedure disposecontrols;
var i: integer;
begin
  for i: =0 to F_report. TitleBand1. ControlCount -1 do //取消系统对控件的控制
    F_report. TitleBand1. RemoveControl(F_report. TitleBand1.
Controls[0]);
for i: =1 to F_report. ColumnHeaderBand1. ControlCount DO
     F_report. ColumnHeaderBand1. RemoveControl(F_report.
ColumnHeaderband1. Controls[0]);
  for i: =1 to F_report. detailband1. controlcount do
   F_report. detailband1. removecontrol(F_report. detailband1.
Controls[0]);
  F_report. Table1. active: =false;
end;
procedure TF_main. printBitBtnClick(Sender: TObject);
begin
  F_report. QuickRep1. Print;
end;
procedure TF_main. closeBitBtnClick(Sender: TObject);
begin
  disposecontrols;
  close;
end;
end.
```

参考文献

1. 潘将一著. DELPHI4. 0 数据库与 INTERNET 开发指南. 清华大学出版社 1999. 9

2. 美）P. Thurrott G. Brent R. Bagdazian S. Tendon 著 卢庆龄、蒋全等译. DELPHI3. 0 编程参考手册. 清华大学出版社，1998. 8

收稿日期：2001 年 1 月 11 日）

用 VC + +完善 RealPlayer

林 泉

摘 要 目前流行的媒体播放软件 RealPlayer 存在着一些使用不便之处。我们为了解决这些问题而采用 VC + +6.0 编制了一个小程序。该程序能够适当弥补 RealPlayer 的不足。同时该程序所采用的很多技巧对于我们其它程序也具有相当的参考价值。

关键词 VC + +,RealPlayer

一、前言

流行的媒体压缩格式 rm 凭借其极高的压缩比率和相对较小的压缩损伤，一举成为目前流行的影音压缩格式，同时其播放软件 RealPlayer 也就顺理成章地成为几乎每台电脑必备的媒体播放软件。但是在使用 RealPlayer 中我们发现其中的一些设计缺憾给广大的使用者带来了许多不便，例如 RealPlayer 不能在播放过程中自动将本机的屏幕保护和能源保护功能关闭，于是设置了这些功能的用户就不得不在使用 RealPlayer 之前手工关闭屏幕保护，而使用完 RealPlayer 之后又需要手工开启屏幕保护。还有就是 RealPlayer 虽然支持扩展名为 .ram 的播放列表格式，但是本身不支持播放列表的编辑，导致播放列表功能形同虚设。其实通过用 VC + +6.0 制作一个小程序，就能够适当弥补 RealPlayer 的这些不足之处。同时，实现该程序所用到的许多技巧在其它的程序编制中也有着相当的实用价值。

二、程序设计思路

为了解决 RealPlayer 的上述问题，可以制作一个小程序来进行 rm 等媒体文件的搜索和排序等工作，然后在编排好播放列表以后调用 RealPlayer 播放该列表即可完成播放列表的管理与编排。而如果在程序最开始加上关闭屏幕保护和能源保护的代码，在程序退出之前执行恢复屏幕保护和能源保护的代码，就能够免去每次手动调整的麻烦。另外通过该程序还应该能直接控制 RealPlayer 的播放、暂停以及停止功能。

三、程序用到的技术

1. 屏幕保护和能源保护功能的关闭与开启

在许多的媒体播放软件中都早已实现了自动关闭屏幕保护和能源保护的功能，其实要实现这一功能非常简单。只需要使用一个 Windows 的 API 函数 SystemParametersInfo，就可以对许多系统参数进行调整，包括关闭或开启屏幕保护和能源保护。SystemParametersInfo 函数的原型如下：

BOOL WINAPI SystemParametersInfo UINT uiAction UINT uiParam PVOID pvParam UINT fWinIni

使用其关闭屏幕保护的代码为：

SystemParametersInfo(SPI_SETSCREENSAVEACTIVE, FALSE, 0, SPIF_SENDWININICHANGE);

使用其关闭能源保护的代码为：

SystemParametersInfo(SPI_SETLOWPOWERACTIVE, ALSE, 0, SPIF_SENDWININICHANGE);
SystemParametersInfo(SPI_SETPOWEROFFACTIVE, FALSE, 0, SPIF_SENDWININICHANGE);

使用其打开屏幕保护的代码为：

SystemParametersInfo(SPI_SETSCREENSAVEACTIVE, TRUE, 0, SPIF_SENDWININICHANGE);

使用其打开能源保护的代码为：

SystemParametersInfo(SPI_SETLOWPOWERACTIVE, TRUE, 0, SPIF_SENDWININICHANGE);
SystemParametersInfo(SPI_SETPOWEROFFACTIVE, TRUE, 0, SPIF_SENDWININICHANGE);

2. 调用 RealPlayer 播放 .ram 播放列表

我们知道在 Windows 下面许多类型的文档都与对应的执行文件进行了关联，只要在资源管理器里面双击已经建立了关联的文件，Windows 就会自动使用文档所关联的文件来打开该文档。而在 VC + + 中如果要调用 Windows 的默认关联程序来打开 播放）一个文件，则可以使用 ShellExecute 函数。该函数原型如下：

WINSHELLAPI HINSTANCE APIENTRY ShellExecute HWND hwnd LPCSTR lpOperation LPCSTR lpFile LPCSTR lpParameters LPCSTR lpDirectory INT nShowCmd

例如我们要用 Windows 自动调用 RealPlayer 来播放一个文件名为 hotfox.ram 的播放列表，则可以使用如下的代码：

ShellExecute 0 “open” “hotfox.ram” NULL NULL 0

3. 实现点击按钮后弹出一 Popup 菜单

在许多情况下我们要在有限的空间内放置许多功能，例如 Winamp 的 PlayList 面板里面就是这样的。为了将大量的功能分类组织在几个有限的按钮上，我们需要在用户点击对应的按钮以后能够弹出该按钮对应的 Popup 菜单。实现步骤如下：

首先，在 ResourceView 的 Menu 项目里加入需要的弹出菜

单。其中顶级的菜单标题并不需要填入。

然后，假设我们要控制的按钮的 ID 为 IDC_BUTTON_DELFILE，当点击这个按钮弹出的菜单的 ID 为 IDR_MENU_DELFILE，则应该在对应的按钮的响应函数 OnButtonDelFile 中加入如下代码：

```
void CRealFoxDlg: : OnButtonDelFile()
  {
    RECT rect; // 用来存放所点击按钮的位置
    CButton * pBtnDelFile; // 对应所点击按钮的指针
    pBtnDelFile = (CButton * )(GetDlgItem(IDC_BUTTON_
DELFILE)); // 取出按钮指针
pBtnDelFile - >GetWindowRect(& rect); // 得到按钮的位置
    CMenu menu, * pmnuContext;
    // 读入上下文菜单
    menu. LoadMenu(IDR_MENU_DELFILE);
    pmnuContext = menu. GetSubMenu(0);
    if(pmnuContext) // 弹出菜单
    pmnuContext - >TrackPopupMenu(
TPM_LEFTALIGN | TPM_LEFTBUTTON, // 菜单的弹出方向为右下
(rect. left + rect. right)/2, // 菜单的位置为按钮的中央
    (rect. top + rect. bottom)/2,
    this, // 菜单的父窗口指针
    NULL);
    }
```

这样，当我们点击这个 ID 为 IDC_BUTTON_DELFILE 的按钮时，就会在这个按钮中央弹出 ID 为 IDR_MENU_DELFILE 的菜单，我们再将这个菜单中的每个项目的消息响应函数中添加上需要的代码即可。

4. 实现 Dialog Based 程序的 ToolTip 功能

直接通过 App Wizzard 生成的 Dialog Based 程序是不具备 ToolTip 功能的，要添加这个功能也很简单。

首先，在 VC 的集成界面菜单中选择 Project - > Add To Project - > Components and Controls，然后在出现的对话框选择 Visual C + + Components - > ToolTip Support，然后点击 Insert 按钮即完成加入。

然后在程序的主对话框的 OnInitDialog 函数中，找到已经加入的 ToolTip 初始化部分，然后按照其注释里的说明，加上类似于如下的代码即可完成 ToolTip 功能：

```
    m_ToolTip. AddTool GetDlgItem IDC_BUTTON_DELFILE
"从列表中移除文件"
```

5. 利用 CFileDialog 公共对话框实现单选和多选文件

利用 CFileDialog 实现文件单项选择是很简单的，很多书籍资料上都有，这里只简单地列出其实现代码，主要用来与实现文件多项选择进行对照。

文件单选代码如下：

```
  {
    // sFileter 是文件类型过滤字符串
    CString sFileName;
    CString sFilter =
    "Real 媒体 ( * . rm, * . ra, * . au, * . avi, * . mp3, *
. mpg, * . mpeg) | * . rm; * . ra; * . au; * . avi; * . mp3;
* . mpg; * . mpeg|所有文件 ( * . * ) | * . * ||";
    CFileDialog dlgOpen(TRUE, 0, 0, OFN_FILEMUSTEXIST,
        (LPCTSTR)sFilter, this);
    if(IDCANCEL = = dlgOpen. DoModal())
    return; // 如果用户 Cancel 掉选择则返回
sFileName = dlgOpen. GetPathName(); // 取出用户所选的文件全路径
    }
```

而实现文件多选则相对复杂得多，除了在创建的时候要加上 OFN_ALLOWMULTISELECT 属性以外，还要求创建一个用来存放文件名列表的缓冲区，然后在执行 DoModal 显示对话框之前，将 CFileDialog 对象的 m_ofn. lpstrFile 和 m_ofn. nMaxFile 两个成员变量分别赋值为缓冲区指针和缓冲区的长度。最后在用户选择完文件返回以后，缓冲区里存放的是一系列连续存放的以字符 'h0' 结束的字符串。如果用户只选择了一个文件，那么缓冲区内只有一个字符串，内容是选择的那个文件的包括目录在内的完整路径；而如果用户选择了多个文件，则缓冲区内的第一个字符串是所选文件所在的路径名，而剩下的字符串都是所选文件的文件名（不含目录名）。

具体实现代码如下：

```
  {
  CString sFilter = "Real 媒体 ( * . rm, * . ra, * . au, *
. avi, * . mp3, * . mpg, * . mpeg) | * . rm; * . ra; * . au;
* . avi; * . mp3; * . mpg; * . mpeg|所有文件 ( * . * ) | *
. * ||";
  CFileDialog dlgOpen(TRUE, 0, 0, OFN_FILEMUSTEXIST |
OFN_ALLOWMULTISELECT, (LPCTSTR)sFilter, this);
  const DWORD MAXBUFFER = 8192;
// 存放文件名列表的缓冲区的长度
TCHAR achBuffer[MAXBUFFER]; // 存放文件名列表的缓冲区
  achBuffer[0] = '\0';
dlgOpen. m_ofn. nMaxFile = MAXBUFFER; //赋值缓冲区长度
dlgOpen. m_ofn. lpstrFile = achBuffer;
// 连接缓冲区到 CFileDialog 上
if(IDCANCEL = = dlgOpen. DoModal())
  return;
  INT i, nCount = 0;
  CString str, sPath = "", s, sFileName[100];
  str = achBuffer;
  i = str. GetLength() + 1;
  if( * (achBuffer + i)) // 如果用户选择了多项
  {
      sPath = str;
      if(sPath[sPath. GetLength() - 1]! = '\\')
      sPath + = '\\'; // 保证目录名最后是\号
while( * (achBuffer + i) & & i < MAXBUFFER)
```

```
// 取出用户所选的全部文件名
        {
        str = achBuffer + i;
        i + = 1 + str. GetLength();
        sFileName[nCount + + ] = sPath + str;
        }
    }
    else // 用户只选择了一项
        sFileName[nCount + + ] = str;
}
```

6. 实现目录的选择以及遍历

在本程序中，允许用户选择某一目录，然后程序会自动将该目录下的所有的媒体文件加入到自己的播放列表里面。其中，要实现只选择目录对话框，可以使用 SHBrowseForFolder 函数。其原型如下：

```
WINSHELLAPI LPITEMIDLIST WINAPI SHBrowseForFolder
(LPBROWSEINFOA lpbi);
```

另外实际使用中还要用到 SHGetMalloc 以及 SHGetPathFromIDList 函数。

具体说明很长，请参见 MSDN。选择目录以及遍历用法举例如下 由于使用了 _chdir 等函数，因此需要在程序前面加上 #include <direct. h>）：

```
void CRealFoxDlg: : OnInsFileDir()
{
  BROWSEINFO bi;
  char szBuf[MAX_PATH];
  LPITEMIDLIST pidl;
  LPMALLOC pMalloc;
  CString szStr = ″″;
  if(: : SHGetMalloc(& pMalloc) = = NOERROR)
  {
bi. hwndOwner = NULL; // 父窗口句柄
bi. pidlRoot = NULL; // 起始根目录
bi. pszDisplayName = szBuf; // 名称缓冲区
bi. lpszTitle = _T(″请选择媒体所在目录″); // 对话框标题
bi. ulFlags = BIF_RETURNFSANCESTORS |
BIF_RETURNONLYFSDIRS;
    bi. lpfn = NULL;
    bi. lParam = 0;
    if((pidl = : : SHBrowseForFolder(& bi)) ! = NULL)
    {
    if(: : SHGetPathFromIDList(pidl, szBuf))
      {
    szStr = szBuf; // 取出用户选择的目录的完整路径名
      }
      pMalloc - >Free(pidl);
    }
    pMalloc - >Release();
    }
    if(szStr = = ″″) //没选择
      return;
    // 处理目录选择
    AddDir(szStr); // 遍历目录树
    }
    void CRealFoxDlg: : AddDir(LPCTSTR lpszDir)
    {
    CFileFind ff;
    INT nOldDrive = 3;
    CString sDrive = ″″, sPath, sFileName, sExt;
    BOOL bWorking;
    sPath = lpszDir;
    if(sPath[sPath. GetLength() - 1] ! = ′\\′)
      sPath + = ′\\′; //保证最后是\号
    sDrive = GetFileName(lpszDir, 1); //取出盘驱号
    if(sDrive. GetLength() > 1) //如果有盘驱号
    {
      sDrive. MakeUpper();
      nOldDrive = _getdrive();
      _chdrive(sDrive[0] - ′A′ + 1); //切换盘驱
    }
    _chdir(lpszDir); //切换目录
    // 搜寻所有的 rm、ra、avi、mpg 文件
    bWorking = ff. FindFile( ″*. *″);
    while(bWorking)
    {
      bWorking = ff. FindNextFile();
      if(ff. IsDirectory()) //如果是目录则跳过
        continue;
      // 根据文件名来去掉不要的部分
      sFileName = ff. GetFileName();
    sExt = GetFileName(sFileName, 4); //取出文件扩展名
    sExt. MakeUpper();
    if(sExt = = ″. RM″ || sExt = = ″. RA″ || sExt = = ″
. AVI″ || sExt = = ″. MPG″) //如果是媒体文件
    {
      // 在这里加入对于找到的文件的操作代码
      }
    }
    // 搜寻所有的子目录
    bWorking = ff. FindFile( ″*. *″);
    while(bWorking)
    {
      bWorking = ff. FindNextFile();
      if(! ff. IsDirectory()) //如果是不是目录则跳过
        continue;
      if(ff. IsDots()) //如果是 . 和 .. 目录则跳过
        continue;
      // 递归
      AddDir(ff. GetFileName());
    }
    _chdir(″..″);
    if(sDrive. GetLength() > 1) //如果有盘驱号
      _chdrive(nOldDrive);
```

```
}
CString CRealFoxDlg::GetFileName(LPCTSTR lpszFilePath, INT nID)
{
    CString sPath = lpszFilePath, sFileName;
    TCHAR sDrive[MAX_PATH], sDir[MAX_PATH], sName[MAX_PATH], sExt[MAX_PATH];
    _splitpath((LPCTSTR) sPath, sDrive, sDir, sName, sExt);
    switch(nID)
    {
    case 1:
        sFileName = sDrive;
        break;
    case 2:
        sFileName = sDir;
        break;
    case 3:
        sFileName = sName;
        break;
    case 4:
        sFileName = sExt;
        break;
    default:
        sFileName = sName;
    }
    return sFileName;
}
```

7. 实现控制 RealPlayer 播放、暂停以及停止控制

要通过外部程序控制另外一个程序，其实只需要知道被控制的程序内部的消息形式，然后在控制程序中设法找到被控制程序的指针或者句柄，通过 SendMessage 或者 PostMessage 函数给被控制程序发送相应的消息就能够实现。例如当知道了点击 RealPlayer 播放按钮所产生的消息是 WM_COMMAND，并且该消息的 wParam 的值为 0x07d1，lParam 的值为播放按钮的句柄，于是现在要让程序控制 RealPlayer 的播放键，也就是给对应的 RealPlayer 窗口发送这条消息过去。

具体的实现代码如下：

```
HWND parWnd=::FindWindowEx(NULL, NULL, "PNGUIClass", NULL);
HWND conWnd=::FindWindowEx(NULL, parWnd, "PNGUIClass", NULL);
HWND playWnd=::FindWindowEx(conWnd, NULL, "Button", "Play");
::SendMessage(conWnd, WM_COMMAND, 0x07d1, (LPARAM)playWnd);
```

同理，控制播放暂停的代码如下：

```
HWND parWnd=::FindWindowEx(NULL, NULL, "PNGUIClass", NULL);
HWND conWnd=::FindWindowEx(NULL, parWnd, "PNGUIClass", NULL);
HWND playWnd=::FindWindowEx(conWnd, NULL, "Button", "Pause");
::SendMessage(conWnd, WM_COMMAND, 0x07d1, (LPARAM)playWnd);
```

控制播放停止的代码如下：

```
HWND parWnd=::FindWindowEx(NULL, NULL, "PNGUIClass", NULL);
HWND conWnd=::FindWindowEx(NULL, parWnd, "PNGUIClass", NULL);
HWND playWnd=::FindWindowEx(conWnd, NULL, "Button", "Stop");
::SendMessage(conWnd, WM_COMMAND, 0x07d1, (LPARAM)playWnd);
```

如果要控制其他的功能，只要利用 VC 自带的 Spy++工具研究被控程序的消息，在通过上述方法就能够实现。

8 RealPlayer 的 RAM 格式

这种播放列表格式很简单，其中用分号起始的行是注释行，剩下的每一行都是一个媒体文件的 URL 本地文件格式为 File ///d /xxx/xxx/xxx.xx 形式）。RealPlayer 在播放 RAM 时会依次播放其中的所有媒体文件，同时也可以控制选择播放“上一个媒体文件”、“下一个媒体文件”或者直接选择其中的某一个媒体文件播放。例如下面就是一个 RAM 格式的播放列表的例子：

```
-----乱码 OVA.ram--------
;剪辑列表标题:乱码 OVA(300KB/s)
File:///F:\乱码 ova\乱码 01.rm
File:///F:\乱码 ova\乱码 02.rm
File:///F:\乱码 ova\乱码 03.rm
File:///F:\乱码 ova\乱码 04.rm
File:///F:\乱码 ova\乱码 05.rm
File:///F:\乱码 ova\乱码 06.rm
;---------文件结束-------------
```

四、补充说明

除去上述代码以外，剩下的都是比较简单的技术，包括 ListCtrl 的控制、排序等。由于篇幅的关系，这里不详细列出全部的程序代码。因为有了以上的技术，要完成这个程序的困难已经很少了。关于上面介绍的 API 的详细说明，可以参阅 MSDN。

参考资料

1. 美 Peter Norton，Rob McGregor. MFC 开发 Windows95/NT4 应用程序

2. Advanced MFC Programing

收稿日期：2000 年 12 月 12 日）

在网上操作 VFP 6.0 数据库

杨 晔

在网上处理大量数据，离不开对数据库的操作，对数据库有各种各样的操作，其中常用的操作是查询、修改、添加、删除记录等。把 ASP Active Server Page 、ADO ActiveX Data Object 及 ODBC Open Database Connection 技术结合起来可以非常容易地在网上对诸如 access、SQL、dBase、Foxbase、Foxpro、Excel 等数据库进行上述操作。

在我国个人计算机上使用得非常普及的、历史最长的是 dBASE Ⅱ、dBASE Ⅲ、Foxbase、Foxpro 中的数据库，为能充分共享这些已有数据资源，本文主要介绍如何在网上对用 Foxpro6.0 建立的数据库进行查询、修改、添加、删除记录等操作。

一、准备工作

（一）安装 PWS （Microsoft Personal Web Server4.0）

使用 ASP 网页前必须先设计和调试，这两项工作当然最好在本地机上完成，要在你的计算机上设计出能运行可操作数据库的 ASP 网页，必须先把你的计算机变成一个 Web 服务器，如果操作系统是 Windows98，那么只要再安装一个 PWS 软件就可以把你的计算机变成一个 Web 服务器。

另外，要建立能存取服务器端数据库的 Web 应用程序，要用到 ADO 的有关对象，而 ADO 是集成在 Microsoft Personal Web Server 或 Internet Information Server 等操作平台中的。基于上述理由，安装 PWS 是必须的。

在 Windows98 光盘的 Add.ons 目录下有一个名为 pws 的目录，执行这个目录下的 setup.exe 程序就可以安装 PWS 软件。安装好后重新启动电脑，在系统任务栏中会出现一个表示 PWS 已经启动的图标。

（二）虚拟目录的设置

先在硬盘上建立一个目录，目录名可任取，本文取的目录名为 “DBWEB”，然后右击任务栏中的 PWS 图标，在弹出菜单中单击 “属性”选项，屏幕上出现一个 “个人 Web 管理器”窗口。单击该窗口中的 “高级”按钮，出现 “高级选项”框，单击 “虚拟目录”框中的 “添加”按钮，出现 “添加目录”对话框，单击 “目录”文本框，再单击 “浏览”按钮，选择先前已经建好的目录 “DBWEB”，然后在 “别名”文本框中输入虚拟目录名 “WLDB”。以后你所建立的数据库、ASP 文件、HTML 文件都要存放在该目录中。

（三）建立一个数据表

为能演示文中所列程序，用 Foxpro6.0 建立一个数据表，该表可以是自由表，也可以是非自由表，笔者建立的是自由表，文件名是 WL_DB.DBF，表中数据如下：

ZKZH	XM	KSRQ	KCMC	KSCJ	KHCJ
21100002	汤彩虹	98.11.30	计算机应用基础	77	
21100003	沈剑英	99.05.29	多媒体课件制作	86	
21100004	顾美娟	98.08.30	网页设计与网络应用	84	
21100007	李晓晨	98.11.30	计算机应用基础	90	
21100008	金莲花	99.05.29	多媒体课件制作	74	
21100009	许小宾	98.08.30	网页设计与网络应用	74	
21100010	赵冠英	99.05.29	多媒体课件制作	82	
21100011	陈银莲	98.11.30	计算机应用基础	71	

（四）设置数据源

要在网上访问数据库信息，首先要与数据源库建立连接，这种连接是利用 ADO 提供的 Connection 对象来实现的，但是只用 ADO 的对象是不能建立有效连接的，ADO 必须与 “ODBC”驱动程序结合起来才能与数据源库建立连接。

“ODBC”是 Microsoft 所定的数据库标准界面，每种数据库软件所建立的数据库的文件类型都不相同，应用程序必须通过驱动程序作为媒介，即利用 “ODBC”驱动程序来达到存取数据库的目的。数据库驱动程序使用 Data Source Name （DSN）来定位和标识特定的 “ODBC”兼容数据库，使 Web 应用程序能够操作数据库。下面说明一下如何来建立 DNS。

打开控制面板，双击 “ODBC”图标，选择 “系统 DSN”，在该选项卡中单击 “添加”按钮，在 “Select a driver for which you want to set up a data source”选项框单击 “Microsoft Visual FoxPro Driver”选项，然后单击 “完成”按钮。在接下来出现的 “ODBC Visual Foxpro Setup”对话框中的 “DATA Source Name”文本框中输入 “NETDB”；再在选项框 “Database type”中单击 “Free Table directory”单选框，如果所建立的是非自由表，则应选择 “Visual FoxPro database [.DBC]”单选框。接着单击 “Browse”按钮，找到前面所建的目录 “DBWEB”，单击 “确定”按钮。通过上述步骤后就设置好了数据源名 “NETDB”，顺便说明一下，数据源名与数据表名，虚拟目录名都无关，可以任意取。

DSN 的类型有 “用户 DSN”、 “系统 DSN”和 “文件 DSN”。 “系统 DSN”允许所有用户登录到特定的服务器上去访问数据库；而 “用户 DSN”使用适当的安全身份证明限制数据库与特定的用户连接； “文件 DSN”用于从文本文件中获取表格。本文使用的是 “系统 DSN”。

二、ASP 网页的设计

ASP 文件是文本文件，其文件名的形式是 *.asp，这种文件可用任意一个纯文本编辑软件来编写，文件中可以包含

“文本”、“HTML 标签”和 ASP 脚本命令的任意组合。所谓脚本是一系列的命令和指令，ASP 使用定界符 <%和%> 括入脚本命令，定界符括入的命令称为主脚本命令，被括入的命令必须对主脚本语言有效，在默认情况下主脚本语言是 VBscript。括在定界符中的脚本只能在服务器端执行并将执行结果发送给客户端浏览器。

利用 ADO 的 Connection、Recordset 及 Command 对象建立与数据库的传输通道后，ASP 就可用 SQL 语言对数据库进行查询、修改、添加和删除等操作。本文涉及到的 SQL 语句共有四条，现列举如下：

SELECT * FROM <数据表名> [WHERE <检索条件>]

功能：从数据表中检索数据，其中“*”号表示所有字段，如只要检索部分字段，则“*”要换成用“,”分隔的字段名表；

INSERT INTO <数据表名 字段名表 > VALUE < 数据表达式表 >

功能：在表尾添加一个包含指定字段值的记录，可将‘数据表达式表”中的值填写到由‘字段名表”规定的字段中；

UPDATE <数据表名> SET < 字段名1 = 数值表达式 1> [< 字段名 2 = 数值表达式 2>. . . . .] [WHERE <条件表达式>]

功能：用新值替换所选字段中的原有值；

DELETE FROM <数据表名> [WHERE <条件表达式>]

功能：从当前数据表中删除符合条件的记录，若无条件选项就要删除所有记录；

SQL 语句的完整格式是相当复杂的，以上仅就本文所要用到的部分作一简略介绍。

ASP 可用内建组件 Response 把数据库中的数据发送到浏览器端，也可用 VBscript、Javascript 等把从数据库中查询到的数据用 HTML 标签括起来后发送到浏览器端。利用内建组件 Request， ASP 可以接受从客户端传来的数据，从而可利用这些数据来确定查询条件，修改数据库中的数据或者增加数据库中的数据。

（一）查询数据记录

1. 利用 Connection 对象查询记录

EXAM1. ASP 程序可以把数据库中所有记录一下子都发送到客户端浏览器上。

```
EXAM1. ASP
<%
set conn = Server. CreateObject( "ADODB. Connection" )
conn. Open "netdb", "", ""
rsSQL = "select zkzh, xm, kcmc, kscj from wl_db"
set rec = conn. Execute(rsSQL)
Do Until rec. EOF
%>
<% =rec(0) %>
<% =rec(1) %>
<% =rec(2) %>
<% =rec(3) %>
<%response. write "<BR>" %>
<%rec. MoveNext
  LOOP
set rec = Nothing
set conn = Nothing %>
```

要访问数据库信息首先要和数据库源建立连接，上例中的第一句就是利用 ADO 提供的 Connction 对象建立程序和 ODBC 数据库之间的连接。接着利用 Connection 对象的 Open 方法打开连接，将数据源‘NETDB”连接到 Conn。Open 方法后有三个参数，分别是数据源名、注册名和密码，因注册名和密码未设置，故未用。然后定义一个 SQL 查询语句，再用 Connction 对象的 Execute 方法打开 WL_DB. DBF 表，建立记录集对象 rec。接下来的循环语句是把数据记录发送到客户端浏览器，在循环体中语句<% =rec i %> 的功能是把第 i 个字段的值发送到客户端，每循环一次就用 ASP 的内建对象 RESPONSE 的 WRITE 方法发送一个换行标签“〈BR〉”到浏览器，然后下移一条记录，进行下一个循环。最后一句是用来切断与数据表的联系，释放所占内存。

Exam1. asp 文件不能直接执行，必须用 HTML 文件中的表单（form）将 ASP 文件发送给 Web 服务器后才能执行。发送 Exam1. asp 的 HTML 文件见 Exam1. HTM。

```
EXAM1. HTM
<html>
<head>
<meta http – equiv = "Content – Type"
content = "text/html: charset = gb_2312 – 80">
<meta name = " GENERATOR" content = " Microsoft Front-
Page Express2. 0">
<title>网络数据库 </title>
</head>
<body>
<form method = " POST" action = " http: //ok/wldb/ex-
am1. asp">
<input name = " submit" type = " Submit" value = " 开始查
询">
</form>
</body>
</html>
```

有了以上两个文件就可以来演示一下了。首先启动 PWS4. 0，然后打开 IE 浏览器，在地址栏中输入‘http //ok/wldb/exam1. htm”后按一下回车，再单击‘开始查询”按钮就可得到如下查询结果。上述网址中的‘ok”是笔者使用的服务器名，也就是在安装 Windows98 时输入的用户名，如果不知道所使用的计算机的主机名，可以执行 Windows 目录下的 winipcfg. exe，再单击‘详细信息”按钮即可查到；‘wldb”就是用 PWS4. 0 建立的虚拟目录。

查询结果如下：

21100002 汤彩虹 计算机应用基础 77

21100003 沈剑英 多媒体课件制作 86

21100004 顾美娟 网页设计与网络应用 84

21100007 李晓晨 计算机应用基础 90

21100008 金莲花 多媒体课件制作 74

21100009 许小宾 网页设计与网络应用 74

21100010 赵冠英 多媒体课件制作 82

21100011 陈银莲 计算机应用基础 71

EXAM1. ASP 非常简单，但不能按客户要求查询，数据显示又不整齐，所以使用价值不大。可以用下面的 EXAM2. HTM 输入查询对象的准考证号，然后提交给 EXAM2. ASP 即可查到所需记录。

2. 利用 Recordset 对象按条件查询数据记录

EXAM2. HTM

```
<html>
<head>
<meta http-equiv="Content-Type"
content="text/html; charset=gb_2312-80">
<meta name="GENERATOR" content="Microsoft FrontPage Express2.0">
<title>考试成绩查询</title>
</head>
<body>
<center><font size=7 color="blue" style="font-family: 隶书"><B>自学考试成绩查询网页</B></font></center>
<form method="POST" action="http://ok/wldb/exam2.asp">
<hr>
<div align="center"><p><font size=5 color="red" style="font-family: 宋体">准考证号: </font><input id="zkz" name="zkz" value="" size=8></p>
<input name="submit" type="Submit" Value="开始查询">
<input name="reset" type="reset" Value="重输考号">
</center>
<div>
</form>
</body>
</html>
```

EXAM2. ASP

```
<%@ Language=VBScript %>
<html>
<HEAD>
<TITLE>考试成绩表</TITLE>
</HEAD>
<BODY>
<%
kh=Request("zkz")
Set Conn=Server.CreateObject("ADODB.Connection")
Conn.Open "netdb","",""
dbSQL="select zkzh, xm, kcmc, ksrq, kscj, khcj from wl_db where zkzh='"&kh&"'"
Set rec=Server.CreateObject("ADODB.Recordset")
rec.Open dbSQL, Conn
%><p><center>
<TABLE border=1><tr>
<td>准考证号</TD>
<TD>姓名</TD>
<TD>考试科目</TD>
<TD>考试日期</TD>
<TD>考试成绩</TD>
<TD>考核成绩</TD>
</tr>
<% do while not rec.Eof %>
<tr><% for i=0 to Rec.fields.count-1 %>
<td><% =Rec(i) %></td>
<%next%></tr>
<% rec.MoveNext
loop %>
</table></center>
<%set rec=Nothing
Set Conn=Nothing %>
<div align=right><a href="http://ok/wldb/exam2.htm">返回</a></div>
</p></body>
</html>
```

要按条件查询记录首先要解决如何获取用于组织查询条件的客户的输入信息，解决这一问题有赖于 ASP 的内建对象 Request，在本例中就是用语句 "kh=Request("zkz")"来获取准考证号的，并把它保存在变量 kh 中，然后再在 SQL 语句中组成查询条件。在条件表达式中一定要用 "'"&kh&"'" 的格式才能使用变量中的数据。如果条件有多个，可用 ".AND."、".OR." 等逻辑运算符把它们组织起来。

为使查询到的数据排列整齐，应用程序中使用了 HTML 语言中的表格标签 "<TABLE>.....</TABLE>"，把查到的数据放在表格内。

应用程序的第一句 <%@ Language=VBscript%> 用于设置主脚本语言，该句一定要放在 ASP 应用程序的开头。由于默认情况下 ASP 使用的主脚本语言是 VBscript，所以应用程序中如果使用的是 VBscript 的话，该句可以省略。

在浏览器的地址栏中输入 "http://ok/wldb/exam2.htm" 后按一下回车，输入准考证号后就可查询，查询结果如下：

准考证号	姓名	考试科目	考试日期	考试成绩	考核成绩
21100010	赵冠英	多媒体课件制作	99.05.29	82	

在 EXAM1. ASP 中只用了 Connection 对象来查询记录，用这一方法有一个不足之处，即在同一时刻不能有两个或两个以上的客户共享一个数据库。为解决这一问题要用到 ADO 中另一对象 Recordset。

ADO 中的 Recordset 对象代表了对数据库操作返回的整个结果集，EXAM2. ASP 除了创建了一个连接对象 Conn 外，还创建了一个 Recordset 对象 rec，打开数据表后会先将记录存储到 Rec 中，这样数据库就可以让两个以上的客户同时应用了。

另外，Recordset 对象会在所连接的数据库文件里建立一

个指向记录的指针，可以利用 Recordset 对象中的有关方法上下移动指针，指向所需记录，为操作数据库提供了许多方便。

（二）添加数据记录

以下介绍两种添加记录的方法。

1. 使用 Connection 对象添加记录

添加记录的 ASP 应用程序见 EXAM3. ASP，通过与 EXAM1. ASP 的比较可以看出只要把 EXAM1. ASP 中的 SQL 语句改成添加语句，再把其中的显示数据部分的语句去掉，即可得到 EXAM3. ASP。可用 EXAM3. HTM 把 EXAM3. ASP 发送到服务器，执行的结果可用 EXAM1. ASP 来观察。

EXAM3. ASP

```
<html>
<HEAD>
<TITLE>添加记录</TITLE>
</HEAD>
<BODY>
<%
zkz1 =request(″zkz″)
xingming1 =request(″xingming″)
rq1 =request(″rq″)
kc1 =request(″kc″)
cj1 =request(″cj″)
kh1 =request(″kh″)
Set Conn =Server. CreateObject(″ADODB. Connection″)
Conn. Open ″netdb″, ″″, ″″
sch =″insert into wl_db(zkzh, xm, ksrq, kcmc, kscj, khcj) values_
(′″& zkz1 & ″′, ′″& xingming1 & ″′, ′″& rq1 & ″′, ′″& kc1& ″′, val(′″& cj1& ″′), ′″& kh1& ″′)″
conn. Execute(sch)
set Conn =Nothing %>
<a href =″http: //ok/wldb/exam3. htm″>返回</a></div>
</p></body>
</html>
```

EXAM3. HTM

```
<html>
<head>
<meta http -equiv =″Content -Type″
content =″text/html: charset =gb_2312 -80″>
<meta name =″GENERATOR″ content =″Microsoft FrontPage Express2. 0″>
<title>网络数据库</title>
</head>
<body>
<form method =″POST″ action =″http: //ok/wldb/exam3. asp″>
<p>准考证号: <input id =″zkz″ name =″zkz″ value =″″ size =8></p>
<p>姓　名: <input id =″xingming″ name =″xingming″ value =″″ size =8></p>
<p>考试日期: <input id =″rq″ name =″rq″ value =″″ size =8></p>
<p>课程名称: <input id =″kc″ name =″kc″ value =″″ size =26></p>
<p>考试成绩: <input id =″cj″ name =″cj″ value =″″ size =4></p>
<p>考核成绩: <input id =″kh″ name =″kh″ value =″″ size =4></p>
<input name =″submit″ type =″Submit″ Value =″开始添加″>
<input name =″reset″ type =″reset″ Value =″重新输入″>
</form>
</body>
</html>
```

2. 使用 Command 对象添加记录

EXAM4. ASP

```
<%@ Language =VBScript %>
<html>
<HEAD>
<TITLE>添加记录</TITLE>
</HEAD>
<BODY>
<%
Set Conn =Server. CreateObject(″ADODB. Connection″)
Conn. Open ″netdb″, ″″, ″″
set comm =Server. CreateObject(″ADODB. Command″)
set comm. ActiveConnection =Conn
comm. CommandText =″insert into wl_db(zkzh, xm, ksrq, kcmc, kscj, khcj) values(?, ?, ?, ?, ?, ?)″
comm. Prepared =True
comm. Parameters. Append comm. CreateParameter(″x1″, 129, , 8)
comm. Parameters. Append comm. CreateParameter(″x2″, 129, , 8)
comm. Parameters. Append comm. CreateParameter(″x3″, 129, , 8)
comm. Parameters. Append comm. CreateParameter(″x4″, 129, , 26)
comm. Parameters. Append comm. CreateParameter(″x5″, 3, , 8)
comm. Parameters. Append comm. CreateParameter(″x6″, 129, , 8)
comm(″x1″) =request(″zkz″)
comm(″x2″) =request(″xingming″)
comm(″x3″) =request(″rq″)
comm(″x4″) =request(″kc″)
comm(″x5″) =request(″cj″)
comm(″x6″) =request(″kh″)
comm. Execute
set comm =Nothing
set conn =Nothing
%>
<a href =″http: //ok/wldb/exam4. htm″>返回</a>
</body>
</html>
```

ADO 中的 Command 对象可以提高对数据库的操作能力，该对象可使 SQL 语言中的可变部分的选项保持局部未定义，Command 对象中的 Prepared 属性可以用来确定在一个 Command 对象第一次执行之前是否创建一个存储在 CommandText

中的命令的编译好的语句，如果编译了，则在以后执行该命令时都使用这个编译好的语句，这样能够大大提高性能。

由于 Command 对象不能直接建立连接，所以必须用 ActiveConnection 属性来确定特定的 Command 对象执行时使用的 Connection 对象。

SQL 语句应作为一个属性值，赋给 Command 对象的属性 CommandText。

使用 Command 对象时还要用 Command 对象中的 CreateParameter 方法来创建具有合适属性设定的 Parameter 对象，并用 Append 方法将其加入到 Command 对象的 Parameters 集合中，Parameter 是 Command 对象中的一个对象，代表查询的参数或存储过程的进出参量和返回值。CreateParameter 方法的格式如下：

Command . . CreateParameter (A，B，C，D，E)，其中各参数的含义如下：

A：参数的名称

B：参数的类型 (Type)，参数类型较多，就列举以下几个

类型	Char	Integer	Numeric	Time
代码	129	3	131	134

C：方向 1、输入 2、输出 3、输入或输出 4、指返回值

D：参数的最大长度

E：参数的默认值

在把 Parameter 对象附加到 Parameters 集合之前必须先设置它的 Type 属性。

除 URL 外，EXAM4. HTM 与 EXAM3. HTM 完全相同，故省略。

删除和修改记录的例子见 EXAM5. ASP 和 EXAM6. ASP，其中主要知识在前面都已解释过，故不再赘述。所要用到的 HTM 文件分别和 EXAM2. HTM、EXAM3. HTM 相同，只要修改一下相应的 URL 即可。这两个文件执行的结果都可用 EXAM1. ASP 来查看。

三 删除记录

```
EXAM6. ASP
<%
kh =Request(″zkz″)
Set conn =Server. CreateObject(″ADODB. Connection″)
Conn. Open ″netdb″, ″″, ″″
Conn. Execute ″delete from wl_db where zkzh =″ & ″′″ & kh& ″′″
Set Conn =Nothing %>
```

(四) 修改记录

```
EXAM7. ASP
<%
zkz1 =request(″zkz″)
xingming1 =request(″xingming″)
rq1 =request(″rq″)
kc1 =request(″kc″)
cj1 =request(″cj″)
kh1 =request(″kh″)
Set conn =Server. CreateObject(″ADODB. Connection″)
Conn. Open ″netdb″, ″″, ″″
Conn. Execute ″Update WL_DB set xm =′″& xingming1& ″′,
ksrq =′″& rq1& ″′, kcmc =′″& kc1& ″′_
, kscj = val(′″& cj1 & ″′), khcj =′″ & kh1 & ″′ WHERE
ZKZH =″& ″′″& zkz1& ″′″
Set Conn =Nothing
%>
```

以上例程全都调试通过。笔者所用的计算机是奔腾 133，内存 32MB；软件环境是 Windows 98、IE 4. 0、PWS 4. 0 及 VFP 6. 0。

收稿日期：2000 年 12 月 11 日)

长城网络为深圳教育行业再助新力

近日，长城网络在教育行业连连中标的喜讯频频传来，继在广东中山教育系统、贵阳教育系统成功中标后，日前又分别在深圳龙岗区教育局、深圳宝安区教育局和深圳南山区教育局主办的 ‘校校通 ”工程中，连连夺魁！长城网络将为这些区的所有中小学建立多媒体教室，并提供全部的网络产品和服务器，据悉，该投标活动是配合深圳市建设全市中小学校多媒体教室的工程，长城网络除了提供全部的网络产品之外，还将为整个工程提供方案和技术支持。

目前，国家规定几年内省会城市的中小学校必须建成多媒体教室，以加快教育行业的网络化进程。深圳市作为全国信息化建设的先进试点，积极落实这项政策，以区为点，展开所辖范围内所有中小学校普及网络教育工程的行动，成为教育界 e 化的领跑者。此次的招标活动均由各个区教育局、电教站和财政局主办并组成评审委员会，所有的技术设备选购中采用公开招标的方式，公平、严格的选拔。参与此次投标的厂商有联想、方正等十余家，而长城网络产品以其较高的性能价格比，得到评审委员会的一致首肯，最终连夺花魁。

作为国内著名网络产品提供商的长城网络，在不长的时间里就凭其雄厚的技术实力和无可比拟的价格优势受到业界的瞩目和市场的认可，其丰富的网络产品线和成熟的解决方案满足了多方要求，特别是在其一直致力于推进网络化进程的教育行业中，更是起到了推波助澜的作用。新世纪里，以普及教育行业网络化为市场战略重点之一的长城网络将为教育界再助新力！

用 VC 编制函数图形的显示类

武学师　石江涛

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

关键词　函数图形显示，VC，OPEN GL，面向对象技术

一、引言

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

二、二维显示要求的函数显示类的编制

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

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

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

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

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

```
float func  float x
{
return x * x
}
```

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

函数 DrawChart 的关键程序如下

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

上段程序很容易以下列形式调用：

```
  MyCDC dc(this);
  dc.  DrawChart(func, a, b, c);
```

以下调用的形式基本类似，故省略。

以上只是在直角坐标系下 很容易将其扩充到极坐标系如下面的画方向图的函数

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

同样 func 仍为函数指针 C 为放大倍数 其内部操作大致为

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

```
        x = (int)x1; y = (int)y1;
        x = 300 + x;
      y = 300 - y;
        LineTo(x, y);
  }
  r = max/4; s = 0;
//画出坐标的分度
  for(i = 1; i < = 4; i + + ){
        s = s + r;
        R = (int)s;
      Arc(300 - R, 300 - R, 300 + R, 300 + R, 300 + R, 300,
300 - 1 + R, 300 + 1);
  }
```

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

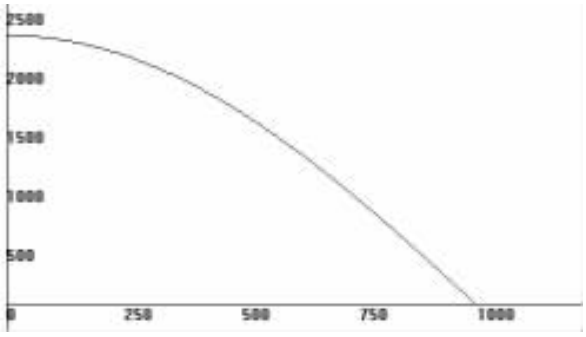


图 1

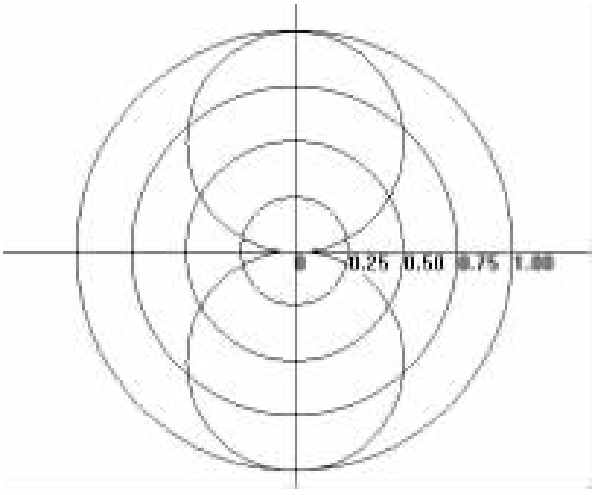


图 2

三、三维显示要求的函数显示类的编制

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

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

可在修改过的视类中定义一 Draw3D 函数：

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

函数内部的实现过程如下：

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

图 3 是利用上述程序实现函数：

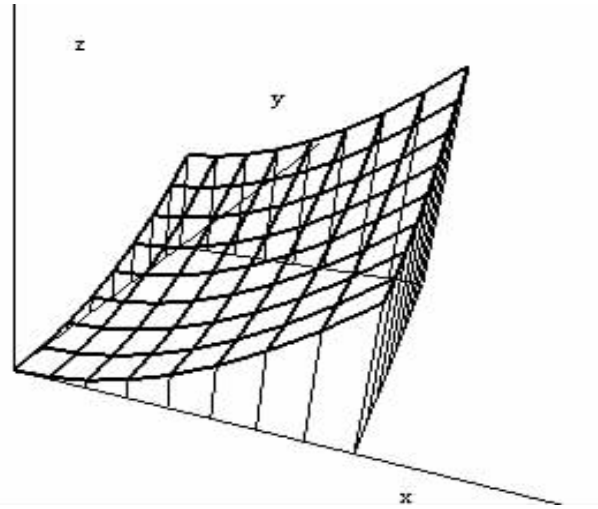


图 3

$Z = x^2 + y^2$

在区间 x 0. . 1 y 0. . 1 z 0. . 1 上的图线。

四、消隐技术的实现

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

在 OpenGL 中，可以利用以下的技术实现消隐：

每次求显示的数据时求出函数的两列 如果速度要求很快的话将所求的数据放到二维数组中 ，在课题中我们将数据放

可视化专栏 VISUAL COLUMN

再论不规则窗体的制做

——用 VC6 实现不规则对话框

王志刚

以往不规则窗体的实现，无外乎用矩形、圆、椭圆等基本区域合并而成；稍复杂一点的形状可以用 CreatePolygonRgn 函数实现。但是 CreatePolygonRgn 函数需要一个 POINT 类型的数组参数，当然可以利用“画图”勾勒出想要的形状，同时记录下坐标值形成数组。显而易见这不是个好主意，在形状较大又较复杂的情况下工作量非常大，而且不具有通用性，违反了程序设计的原则。我们应该将这些繁杂的工作交给计算机去做，因为它不怕麻烦，完成的质量又很好。

如果能够将 Dib 设备无关位图 的轮廓转换为区域，那么想象一下，我们可以尽情发挥设计出任意形状要多酷有多酷的窗体，所要做的工作就是用绘图或图形处理软件进行“艺术创作”，最后只要将“作品”保存背景为白色或黑色 甚至是其他只要前景中不包括的颜色）的位图。很自然我们需要一个函数 DIBToRgn 完成转换工作。函数的原理很简单，就是对位图逐行逐像素的扫描，每一条水平像素线都是由背景颜色的线段 也可能是点）和“前景线段”组成的，我们可以把这些“前景线段”看成是高度为 1 像素的矩形区域，当整个位图全部扫描完成后，把所有的这些小矩形区域合并在一起就构成了我们想要的形状。另外，程序中涉及堆的内存分配部分并没有使用 C + + 的 new 操作符，因为它的效率比较低，在位图较大和较复杂时速度很慢。我使用了 Win32API 的 Heap 系列函数，结果，速度比较令人满意。DIBToRgn 函数原型为：

```
HRGN DIBToRgn (
  HBITMAP hBmp,
  COLORREF BkColor = 0x00ffffff,
  BOOL Direct =TRUE
);
```

hBmp 是 Dib 位图句柄；BkColor 是位图的背景颜色值可以用 RGB 函数获得；Direct 决定是直接创建区域还是间接创建，所谓直接创建就象前面介绍原理时所说的通过直接合并小矩形区域得到；小矩形区域的数量的多少决定了速度的快慢，在背景区域的小矩形区域少于前景的小矩形区域时，可以用整个位图区域剪去背景区域从而得到前景区域，这样可以提高速度和减少资源占用，这种方法就是间接创建。代码如下：

```
HRGN CDemoDlg: : DIBToRgn(HBITMAP hBmp, COLORREF BkColor, BOOL Direct)
{
  HRGN hRgn = NULL; //用于返回的区域句柄
#define MAX_ALLOC_RECTS 100
COLORREF Tolerance =0x00101010; //颜色偏差量
if (hBmp)
  {
HDC hMemDC = CreateCompatibleDC(NULL); //创建一个存放 32 位 Dib 位图的内存 DC
  if (hMemDC)
  {
  BITMAP bm;
  GetObject(hBmp, sizeof(bm), & bm); //获得位图的尺寸
  //填充位图信息头结构
  BITMAPINFOHEADER BmpInfoh = {
  sizeof(BITMAPINFOHEADER), // biSize
  bm. bmWidth,              // biWidth;
  bm. bmHeight,             // biHeight;
  1,                        // biPlanes;
  32,                       // biBitCount
  BI_RGB,                   // biCompression;
```

到了数组中，这样在求数值时便不用什么顺序了。

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

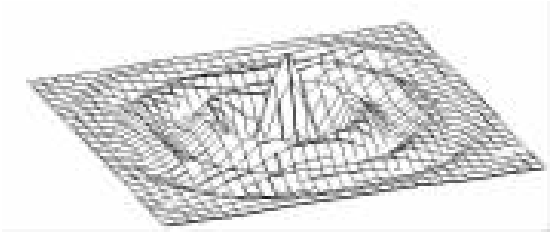


图 4

图 4 为一个波源场的消隐图形。

五、 总结

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

参考文献

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

收稿日期：2000 年 12 月 25 日）

```
    0,                          // biSizeImage;
    0,                          // biXPelsPerMeter;
    0,                          // biYPelsPerMeter;
    0,                          // biClrUsed;
    0                           // biClrImportant;
    };
    LPVOID pBit32; //定义一个指向位图 bit 的指针
    //创建一个 32 位的 Dib(设备无关位图)
HBITMAP hDib32 = CreateDIBSection( hMemDC, ( BITMAP-
INFO * )& BmpInfoh, DIB_RGB_COLORS, & pBit32, NULL, 0);
    if (hDib32)
    {
HBITMAP hOldib32 = (HBITMAP) SelectObject(hMemDC,
hDib32); //将 Dib 选进 DC
HDC hDC = CreateCompatibleDC(hMemDC);
//创建一个用于存放原图的 DC
    if (hDC)
        {
        BITMAP bm32;
GetObject(hDib32, sizeof(bm32), & bm32);
//获取新创建的 32 位 Dib 的尺寸
//确保存 32Dib 的每一行像素的字节数为 DWORD(4 字节)
的整数倍
     while (bm32. bmWidthBytes % 4)
    bm32. bmWidthBytes + +;
HBITMAP holdBmp = (HBITMAP)SelectObject(hDC, hBmp);
//将原图选进 DC
//复制原图到内存 DC
BitBlt(hMemDC, 0, 0, bm. bmWidth, bm. bmHeight, hDC,
0, 0, SRCCOPY);
 DWORD MaxRects = MAX_ALLOC_RECTS;
                  SYSTEM_INFO Sysinfo;
GetSystemInfo(& Sysinfo); //得到内存页尺寸
//创建一个可增大的堆
  HANDLE hRcData = HeapCreate(HEAP_GENERATE_EXCEP -
TIONS, Sysinfo. dwPageSize, 0);
//为堆分配内存
RGNDATA * pRcData = (RGNDATA * ) HeapAlloc(hRcData,
HEAP_ZERO_MEMORY, sizeof(RGNDATAHEADER) + sizeof
(RECT) * MaxRects); //填充 RGNDATA 结构的头
pRcData -> rdh. dwSize = sizeof(RGNDATAHEADER);
pRcData -> rdh. iType = RDH_RECTANGLES;
pRcData -> rdh. nCount = pRcData -> rdh. nRgnSize = 0;
SetRect(& pRcData -> rdh. rcBound, MAXLONG, MAXLONG,
0, 0);
BYTE hr, hg, hb, lr, lg, lb;
switch(BkColor)
    {
    case RGB(255, 255, 255): //如果背景为白色
      hr = GetRValue(BkColor);
      hg = GetGValue(BkColor);
      hb = GetBValue(BkColor);
      lr = min(0xff, hr - GetRValue(Tolerance));
      lg = min(0xff, hg - GetGValue(Tolerance));
      lb = min(0xff, hb - GetBValue(Tolerance));
      break;
   case RGB(0, 0, 0): //如果背景为黑色
 lr = GetRValue(BkColor);
lg = GetGValue(BkColor);
 lb = GetBValue(BkColor);
 hr = min(0xff, lr + GetRValue(Tolerance));
 hg = min(0xff, lg + GetGValue(Tolerance));
 hb = min(0xff, lb + GetBValue(Tolerance));
    break;
default: //如果背景为其它颜色
  Tolerance = 0x050505;
  lr = max(0, GetRValue(BkColor) - GetRValue(Tolerance));
lg = max(0, GetGValue(BkColor) - GetGValue(Tolerance));
lb = max(0, GetBValue(BkColor) - GetBValue(Tolerance));
hr = min(0xff, GetRValue(BkColor) + GetRValue(Tolerance));
hg = min(0xff, GetGValue(BkColor) + GetGValue(Tolerance));
hb = min(0xff, GetBValue(BkColor) + GetBValue(Tolerance));
   break;
   }
 //得到位图 bit 指针, 从位图的底部开始逐行扫描每一个象素
(位图在垂直方向是反向存储的)
BYTE * pBits = (BYTE * )bm32. bmBits + (bm32. bmHeight
- 1) * bm32. bmWidthBytes;
for (int y = 0; y < bm. bmHeight; y + +)
   {
   for (int x = 0; x < bm. bmWidth; x + +)
   {
   int x0 = x;
   DWORD * pColor = (DWORD * )pBits + x;
        BYTE dr, dg, db;
        while (x < bm. bmWidth)
        {
        dr = GetRValue( * pColor);
        dg = GetGValue( * pColor);
        db = GetBValue( * pColor);
if ((dr>= lr && dr < = hr)& & (dg>=lg& & dg < =hg)& &
(db>=lb& & db < =hb))
   {
        if(Direct)
        break;
        else{
        pColor + +;
        x + +;
        }
        }
        else if(Direct)
        {
        pColor + +;
        x + +;
        }
   else
        break;
        }
```

```
            if (x > x0)
            {
        // 增加一个用(x0, y, x, y + 1)创建的矩形区域
        if (pRcData -> rdh. nCount >= MaxRects)
            {
            MaxRects += MAX_ALLOC_RECTS;
          //重新为堆分配内存
pRcData = (RGNDATA * )HeapReAlloc(hRcData, HEAP_ZERO_
MEMORY, pRcData, sizeof(RGNDATAHEADER) + sizeof
(RECT) * MaxRects);
      }
RECT * pr = (RECT * )& pRcData -> Buffer;
SetRect(& pr[pRcData -> rdh. nCount], x0, y, x, y + 1);
pRcData -> rdh. rcBound. left = x0;
pRcData -> rdh. rcBound. top = y;
pRcData -> rdh. rcBound. right = x;
pRcData -> rdh. rcBound. bottom = y + 1;
pRcData -> rdh. nCount + +;
if (pRcData -> rdh. nCount = = 3000)
{
HRGN tmphRgn = ExtCreateRegion(NULL, sizeof(RGNDATA-
HEADER) + (sizeof(RECT) * MaxRects), pRcData);
        if (hRgn)
        {
        CombineRgn(hRgn, hRgn, tmphRgn, RGN_OR);
        DeleteObject(tmphRgn);
        }
        else
        hRgn = tmphRgn;
pRcData -> rdh. nCount = 0;
SetRect(& pRcData -> rdh. rcBound, MAXLONG, MAXLONG,
0, 0);
}
}
}
        // 扫描下一行
        pBits - = bm32. bmWidthBytes;
        }
HRGN tmphRgn = ExtCreateRegion(NULL, sizeof(RGNDATA-
HEADER) + (sizeof(RECT) * MaxRects), pRcData);
if (hRgn)
        {
CombineRgn(hRgn, hRgn, tmphRgn, RGN_OR);
            DeleteObject(tmphRgn);
}
else
        hRgn = tmphRgn;
//创建一个与位图等大小的矩形区域,将些区域与透明区域进
行异或合并从而得到想要的位图区域
if(! Direct)
        {
HRGN hRect = CreateRectRgn(0, 0, bm. bmWidth, bm. bm-
Height);
 if(hRect)
```


```
   {
   CombineRgn(hRgn, hRgn, hRect, RGN_XOR);
   DeleteObject(hRect);
}
    else
   return NULL;
}
//释放内存
HeapFree(hRcData, HEAP_NO_SERIALIZE, pRcData);
   SelectObject(hDC, holdBmp);
   DeleteDC(hDC);
   DeleteObject(holdBmp);
   }
   SelectObject(hMemDC, hOldib32);
       DeleteDC(hMemDC);
   DeleteObject(hOldib32);
   DeleteObject(hDib32);
   }
   else
 DeleteDC(hMemDC); //如果 hDib32 为空
   }
}
return hRgn;
}
```

下面以一个具体实例来看一下效果，启动 VC6 创建一个基于 MFC 的对话框工程名称为 Demo。删除对话框上的按钮和静态文本框。在对话框上单击鼠标右键，在弹出的菜单上选择 Properties 项，弹出属性对话框。选择 Styles 选项卡单击 Border 下拉列表框选择 None，然后关闭属性对话框。插入一幅准备好的位图 如上图 ，修改其 ID 为 IDB_BMP。为 CDemoDlg 类增加三个私有成员变量，HRGN m_hWndRgn 窗口区域句柄 、HRGN m_hClientRgn 窗口客户区域句柄）、HBITMAP hBmp (背景位图句柄）。打开 ClassWizard 为 CdemoDlg 添加 WM_ERASEBKGND、WM_NCHITTEST、WM_CLOSE、WM_LBUTTONDBLCLK 四个消息映射函数。打开 Workspace 选择 ClassView 为 CDemoDlg 类添加公有成员函数 DIBToRgn。选择 FileView 展开文件列表，双击 DemoDlg. cpp 在编辑窗口中找到 CDemoDlg 的构造函数，在末尾添加如下语句：

```
. . . . . .
m_hBmp = (HBITMAP) LoadImage( AfxGetApp () -> m_hInstance, MAKEINTRESOURCE(IDB_BMP), IMAGE_BITMAP, 0, 0, LR_CREATEDIBSECITON);
m_hClient = CreateEllipticRgn(33, 34, 200, 125);
m_hWndRng = DIBToRgn(m_hBmp, RGB(0, 255, 0), FALSE);
//背景为绿色
. . . . . .
```

在对话框的初始化函数 OnInitDialog 里添加如下代码

```
. . . . . .
// TODO: Add extra initialization here
if(m_hWndRgn)
  SetWindowRgn(m_hWndRgn, TRUE);
. . . . . .
```

找到 OnEraseBkgnd 函数添加如下代码为对话框增加位图背景

```
// TODO: Add your message handler code here and/or call default
      if(m_hBmp)
      {
    BITMAP bm;
     GetObject(m_hBmp, sizeof(bm), & bm);
     HDC hMemdc = CreateCompatibleDC(pDC ->m_hDC);
        if(hMemdc)
        {
 HBITMAP  hOldBmp = (HBITMAP) SelectObject(hMemdc, m_hBmp);
     if(hOldBMp)
           {
  BitBlt(pDC ->m_hDC, 0, 0, bm. bmWidth, bm. bmHeight, hMemdc, 0, 0, SRCCOPY);
       SelectObject(hMemdc, hOldBmp);
       DeleteDC(hMemdc);
       DeleteObject(hOldBmp);
          return TRUE;
           }
           else
         DeleteDC(hMemdc);
       }
      }
. . . . . .
```

找到 OnClose 函数添加如下代码以便于在程序退出时释放占用的内存

```
 // TODO: Add your message handler code here and/or call default
      if(m_hBmp)
        DeleteObject(m_hBmp);
      if(m_hClientRgn)
        DeleteObject(m_hClientRgn);
      if(m_hWndRgn)
        DeleteObject(m_hWndRgn);
 . . . . . .
```

找到 OnLButtonDblClk 函数添加如下代码

```
// TODO: Add your message handler code here and/or call default
  if(m_hClientRgn)
  {
      if(PtInRegion(m_hClientRgn, point. x, point. y))
        OnOK();
  }
 . . . . . .
```

这样当鼠标双击 m_hClientRgn 区域时程序将退出。

因为对话框没有标题区无法拖动它，所以我重载了 OnNcHitTest 函数添加的代码如下：

```
// TODO: Add your message handler code here and/or call default
  if(m_hClientRgn)
  {
       ScreenToClient(& point);
       if(!PtInRegion(m_hClientRgn, point. x, point. y))
         return HTCAPTION;
       ClientToScreen(& point);
  }
. . . . . .
```

经过这步处理我们就可以在除了 m_hClientRgn 区域以外的任何区域拖动窗口。在本例中 m_hClientRgn 被作为客户区，可以响应鼠标的双击事件，以此类推，我们可以构造一些按钮区域来响应鼠标事件，随着事件的不同显示不同的位图甚至是动画，这样可以实现非常酷的界面。好了按 Ctrl + F5 或者单击执行按钮来看一下我们的劳动成果，怎么样还不错吧！ 本程序在 Pwin98 + VC6. 0 下调试通过）

收稿日期：2000 年 12 月 25 日）

新中大与金川公司共同打造 ‘中国镍都 ”

新中大兰州办事处消息：日前，新中大软件股份有限公司凭借其卓越的品质，强大的产品功能，完善的售后服务，在河西走廊上力克群雄，成功与中国最大的镍钴生产基地、特大型有色冶金、化工联合企业——金川有色金属公司签约，吹响了胜利的号角。

据悉，此次签约所采用的是新中大集团财务软件 NgPower 大型版，后台采用 Window NT 操作系统和 Oracle 数据库系统。

金川公司通过新中大集团财务软件 Ngpower 所提供的先进的信息管理系统平台，能够有效地实现对各分子公司的财务集中管理和监控，提高企业的财务工作效率，控制生产经营成本，从而使集团企业的管理水平上了一个新台阶。

在支持西部大开发的大环境下，新中大公司与金川公司的成功签约，也必将对提高西部企业的管理应用水平和提升西部企业综合竞争力起到积极的推动作用。

MS SQL Server 7.0 的异类数据库链接技术

马　民　张丽艳

MS SQL Server 7.0（简称 SQL Server）是微软最新发布的数据库产品，其主要改进之一是内建了完整的对异类数据库的链接技术。链接技术应用于分布式数据库系统（DDBS）中，它通过在本地数据库中定义链接服务器对象（Linked Server Object）来实现用户对远程数据的透明访问。而大多数主流数据库系统仅支持对自身数据库产品的链接服务，链接异类数据库需要专门的应用服务器产品，如 Oracle 的 Open Gateways 以及 Sybase 的 OmniConnect 产品。因此，微软的异类数据库链接技术使 SQL Server 在异类环境下的 DDBS 中占有重要地位。

下面介绍 SQL Server 数据库链接技术的三层模型体系及其在异类环境下分布式数据库系统中的应用。

一、 SQL Server 数据库链接技术的三层模型体系

微软的数据库链接技术将关系数据库管理系统（RDBMS）由原来的客户/服务器两层体系拓展成客户/服务器/数据库服务器三层模型体系，两者的对比见图 1 所示。

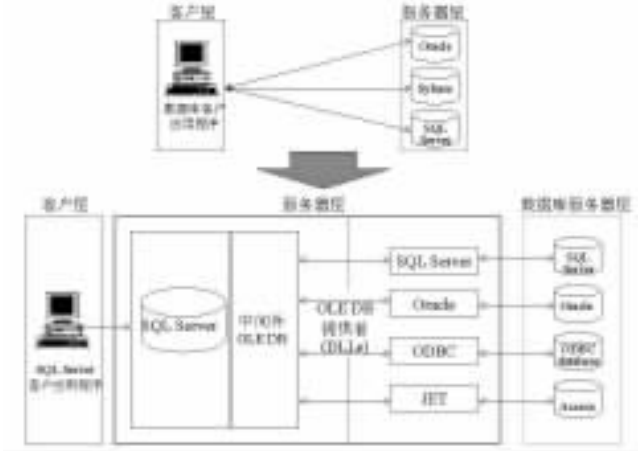


图 1　SQL Server 链接数据库的三层体系与原两层体系的对比

微软主要凭借其掌握 Windows 的内核技术并以 OLE DB 作为中间件（MiddleWare）来实现对异类数据库的链接服务。OLE DB 是微软的一种新技术，用来提供统一的数据访问（UDA），它通过组件对象模型（COM）接口的扩展集来实现对存储在不同信息源中数据的一致访问，适用于关系型和非关系型数据源。在 SQL Server 中，定义链接服务器对象需指定一个 OLE DB 提供者和一个 OLE DB 数据源。OLE DB 提供者是一组用于管理数据源并与之交互的动态链接库（DLLs），OLE DB 数据源可以是任何一种被 OLE DB 访问的数据集合，通常为数据库。

经过微软严格测试可应用于 SQL Server 的 OLE DB 提供者包括 Microsoft OLE DB provider for SQL Server、Jet、ODBC、Oracle 以及 Indexing Service。

我们以一个分布式查询为例加以说明：当客户应用程序执行一个分布式查询命令来请求链接服务器上的数据时，SQL Server 截获该命令并将请求发送给中间件 OLE DB，OLE DB 再将请求提交给关联此链接服务器的 OLE DB 提供者，由该提供者作为客户程序打开数据源进行查询，查询结果通过提供者和 OLE DB 转交给 SQL Server，最后由 SQL Server 发送给客户应用程序。

在上述分布式查询中对链接服务器上的对象（数据）的引用采用四部分命名方法：linked_server_name. catalog. schema. object_name，其依次代表链接服务器的网络名、目录名（对应一个数据库）、用户模式名（对应于对象的所有者）和对象名（如要访问的表名）。

二、 SQL Server 数据库链接技术在异类环境下分布式数据库系统中的应用

异类环境下分布式数据库系统的拓扑图如图 2 所示。图中的 DDBS 包括 Oracle、Sybase 和 SQL Server 等三种异类数据库，分布在以太网环境下由路由器连接的两个网段内，网络协议采用 TCP/IP，设备旁注明的为其 IP 地址/子网掩码、操作系统（分别为 Solaris 2.6、WinNT 4.0 以及 SCO OpenServer 5.0.5）和数据库版本。

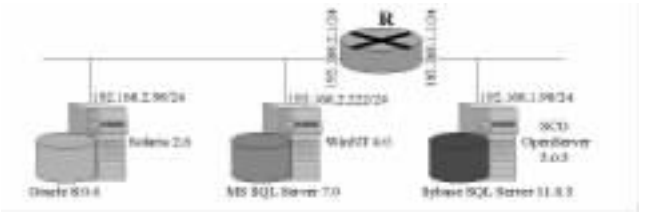


图 2　异构环境下分布数据库系统拓扑图

管理员（DBA）配置 SQL Server 以实现对异类数据库链接服务的参考步骤如下（为描述方便，各数据库中均使用缺省的管理员帐户作为登录名）：

1. 为 SQL Server 所在主机安装异类数据库的客户驱动程序

由于 OLE DB 是作为链接服务器的客户来访问异类数据库的，因此，需为 SQL Server 所在主机安装异类数据库的客户驱动程序。

① 安装 Oracle8 Client

运行 Oracle Installer→选择产品 Oracle8 Client 并按提示完成安装。

运行 Oracle Net8 Easy Config→Add new service，指定 SQL * Net 的别名为 SunOrcl→填写 Solaris 上 Oracle 实例所在主机名和端口号→以 system / manager 测试与实例的连接并完成配置。

② 安装 Sybase Open Client

运行 Sybsetup→选择 Licensed product 并按提示完成安装。运行 Dsedit→add server object→指定服务器名、选择 TCP 协议、填写 SCO OpenServer 的 IP 地址及服务器的端口号→选择 ping 测试与服务器的连接并完成配置。

需要注意的是，对于 Unix 系统平台的异类数据库，还应持有或购买基于 Windows 客户驱动程序的许可协议（License）。比如对于 Sybase，由于没有版本为 11.0.3 的基于 Windows Open Client 的 License，因此安装的是持有 License 的 11.5 PC Client for Unix Platforms HP_UX ，其中 Open Client 的版本号为 11.1.1。

2. 创建异类数据库的链接服务器对象

查询 SQL Server 的有关文档，确定异类数据库的 OLE DB 提供者和数据源并以管理员权限创建链接服务器对象。

① Oracle

使用 OLE DB provider for Oracle，指定 OLE DB 提供者名称为 MSDAORA，数据源为 Oracle 实例的 SQL * Net 别名 SunOrcl。执行系统存储过程 sp_addlinkedserver 创建链接服务器对象 SunOrc：

```
C: \>osql -Usa
Password:
1> sp_addlinkedserver ´SunOrc´, ´Oracle´, ´MSDAORA´, ´SunOrcl´
2> go
(1 row affected)
(1 row affected)
Server added.
```

使用 sp_addlinkedsrvlogin 创建由 SQL Server 登录名 sa 到 Oracle 帐户 system / manager 的映射

```
1> sp_addlinkedsrvlogin ´SunOrc´, false, ´sa´, ´system´, ´manager´
2> go
(0 rows affected)
(0 rows affected)
(0 rows affected)
(1 row affected)
```

② Sybase

使用 OLE DB provider for ODBC。由于 SQL Server 是作为 Windows Service 运行的，因此应创建 ODBC 系统数据源。选择控制面板→ODBC Data Source→System DSN 标签→add，指定数据源名为 DSNScoSYB，服务器名为 SYBASE，数据库为 master。

指定 OLE DB 提供者名称为 MSDASQL，数据源为 ODBC 系统数据源名称 DSNScoSYB，由 SQL Server 登录名 sa 映射 Sybase 登录名 sa，创建链接服务器对象 ScoSYB：

```
1> sp_addlinkedserver ´ScoSYB´, ´´, ´MSDASQL´, ´DSN-ScoSYB´
2> go
(1 row affected)
(1 row affected)
Server added.
1> sp_addlinkedsrvlogin ´ScoSYB´, false, ´sa´, ´sa´, ´manager´
2> go
(0 rows affected)
(0 rows affected)
(0 rows affected)
(1 row affected)
```

3. 进行分布式查询和远程存储过程调用测试

① 准备测试用例

在各自数据库上以所用的映射帐户创建表 ServerInfo 并插入一行有关自身信息的元组（见后查询结果）：

```
create table ServerInfo(
PLATFORM  varchar(30),
PRODUCT   varchar(30),
VERSION   varchar(30)
)
```

在 Sybase 上以映射帐户 sa 创建存储过程 sp_version

```
create proc sp_version as
select @@version VERSION
```

② 分布式查询测试

按四部分命名方法进行分布式查询测试，注意各部分的写法：

```
1> select * from SunORC..SYSTEM.SERVERINFO
2> go
 PLATFORM          PRODUCT           VERSION
 ----------  ----------  ----------
 Solaris 2.6       Oracle            8.0.4
(1 row affected)
1> select * from ScoSYB.master.dbo.ServerInfo
2> go
 PLATFORM              PRODUCT           VERSION
 ----------  ----------  ----------
 SCO OpenServer 5.0.5  Sybase SQL Server  11.0.3
(1 row affected)
```

③ 远程存储过程调用测试

配置链接服务器 ScoSYB 以允许进行 RPC（远程过程调用），其方法是：

```
1> sp_serveroption ´ScoSYB´, ´rpc out´, ´true´
2> go
(1 row affected)
1> sp_helpserver
2> go
name      network_name      status              id
 ----  ----------  ----------  ----
BT2000    BT2000            rpc, rpc out        0
ScoSYB    NULL              rpc out, data acces 2
SunOrc    NULL              data access         1
```

同样采用四部分命名法进行远程存储过程调用：

```
1> ScoSYB. . . sp_version
2> go
 VERSION
 - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
 SQL Server/11.0.3.2/P/SCO/SCO_SV r3.2v5.0.2/SWR
7575 Rollup/OPT/Mon Nov  3 23:43:28 PST 1997
```

④ 创建视图为用户提供远程数据的位置透明性

尽管 SQL Server 没有提供同义词对象，但仍可以用创建视图的方法来为用户提供远程数据的位置透明性。我们既可以为每一个远程表创建一个视图，也可以为相关的几个表创建一个视图，这里采用后一种方法创建视图 TserverInfo：

```
1> create view TServerInfo as
2> select * from ServerInfo
3> union
4> select * from ScoSYB.master.dbo.ServerInfo
5> union
6> select * from SunORC. . SYSTEM.SERVERINFO
7> go
```

这样，用户可以直接查询本地数据库的视图而无需知道数据位于 DDBS 中的位置：

```
1> select * from TServerInfo
2> go
 PLATFORM        PRODUCT          VERSION
 - - - - - -   - - - - - - - -   - - - - - - -
 WinNT 4.0       MS SQL SERVER      7.0
 Solaris 2.6     Oracle             8.0.4
 SCO OpenServer 5.0.5  Sybase SQL Server   11.0.3
(3 rows affected)
```

注意：本文采用了基于命令行 osql 的配置方法，也可以使用基于 GUI 的 Enterprise Manager 进行配置，如图 3 所示，这里不再赘述。

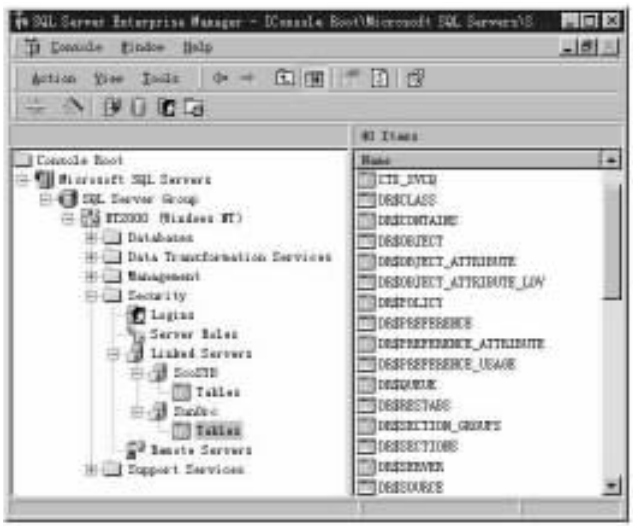


图 3　基于 SQL Server Enterprixe manager 的配置

最后，值得一提的是，在使用 SQL Server 异类数据库链接技术时，除了要考虑 DDBS 固有的优缺点外，下面两点要特别引起注意：

1. 与其它主流数据库相比，SQL Server 毕竟是低端数据库产品，在此异类环境下使用 SQL Server 的链接技术可能会降低整个 DDBS 的性能；

2. 与专门的应用服务器　连接产品）相比，SQL Server 的链接技术仅提供有限的功能，如不能在链接服务器上使用 create、alter 和 drop 语句以及操纵特殊的数据类型等等。

收稿日期：2000 年 11 月 6 日）

彩虹天地新款产品 RG - UMH 套装及软件狗 RG - UMH

作为国内软件保护行业的开拓者和领导者，多年来北京彩虹天地信息技术有限公司一直专注于保护软件开发者的权益，维护开发者的利益。近日，彩虹天地在现有加密狗产品原有基础上，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 RG - DL 型软件狗。新的 RG - UMH 套装由微狗、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，即可在并口狗与 USB 狗间自由切换，体验彩虹天地 “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保护的硬件产品。它采用了多项先进的软件保护技术 不仅具有令人信服的加密强度 更可以操作于多种操作系统之上，如 DOS WINDOWS 3.X/9X/NT/2000/ME 等操作环境。

RG - UMH 套装的另一个重要组成部分是 USB 狗。USB 狗是使用在计算机 USB 口上的硬件加密产品。与微狗相同，USB 狗也具有 200 个字节的数据存储区和可自选的加密算法。并且由于 USB 口自身具有的特点，USB 狗在兼容性与稳定性方面的表现极为突出 它不会与外设发生冲突。现在，USB 狗支持的操作系统有 win98/NT/2000/ME 用户可在这些操作系统下对任何 windows 和 dos 程序加密。

以前，软件开发商在进行软件保护时，需要将微狗与 USB 狗的驱动分别安装两次，才能满足客户端对两种端口的不同需求；而使用 RG - UMH 套装程序，只需安装一次驱动，就可以完成对微狗与 USB 狗的加密操作，从而轻而易举地满足客户端的需求。

对于正在使用微狗或 USB 狗进行软件保护的用户，也只需简单地将现有的软件驱动升级，就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，是软件狗 GS - DJ 的升级产品，是使用在计算机并口上的用于软件保护的低价位产品，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于原 GS - DJ 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 VC 编写基于 ODBC API 的数据库程序

季坚波

一、概要

开放数据库互连 ODBC（Open DataBase Connectivity）是微软开放服务体系（WOSA）的一部分，为应用程序访问关系型数据库提供了一个统一的接口。使用这一标准接口，我们可以不关心具体的数据库管理系统（DBMS）的细节，而只要有相应数据库的 ODBC 驱动程序，就可以实现对数据库的访问。ODBC 编程接口为我们提供了极大的灵活性，我们可以通过这一个接口访问不同类型的数据库。而且，通过相应的 ODBC 驱动程序，我们可以方便地实现不同数据类型之间的转换。ODBC API 使用标准的 SQL（结构化查询语言）作为其数据库访问语言。ODBC 是一种底层的访问技术，因些，ODBC API 可以使应用程序能够从底层设置和控制数据库，完成一些高层数据库技术无法完成的功能。

二、创建 ODBC API 数据库程序

用 ODBC API 编写数据库程序时，要加入 SQLEXT. H 和 ODBC32. LIB 文件。

一般地，编写 ODBC 数据库程序主要有以下几个步骤：

⑴配置数据源

①通过控制面板的“ODBC 数据源（32 位）”手动配置。

②调用 SQLConfigDataSource 函数进行动态配置，使用时要加入 SQLEXT. H 和 ODBC32. LIB 文件。

其函数格式如下：

```
BOOL SQLConfigDataSource(HWND hwndParent, WORD fRequest, LPCSTR lpszDriver, LPCSTR lpszAttributes);
```

参数：

hwndParent：父窗体句柄，若为 NULL 则不显示任何对话框，如果 lpszAttributes 提供的信息不完全，则在创建过程中会出现对话框要求用户提供信息。

fRequest：请求类型，可以设置为以下的数值之一：

ODBC_ADD_DSN：增加一个新的用户数据源。

ODBC_CONFIG_DSN：修改一个已存在的用户数据源。

ODBC_REMOVE_DSN：删除一个已存在的用户数据源。

ODBC_ADD_SYS_DSN：增加一个新的系统数据源。

ODBC_CONFIG_SYS_DSN：修改一个已存在的系统数据源。

ODBC_REMOVE_SYS_DSN：删除一个已存在的系统数据源。

lpszDriver：数据库引擎的名称。比如 Access 为“Microsoft Access Driver（*.mdb）”。

lpszAttributes：一连串的“KeyName = Value”字符串，每两个 KeyName 值之间用“\0”隔开。详细信息可参考微软的 MSDN。

⑵分配一个 ODBC 环境句柄

对于任何 ODBC 应用程序来说，第一步的工作是分配一个环境句柄。首先在程序中声明一个 SQLHENV 类型的变量，用于存放环境句柄，然后调用 SQLAllocHandle 函数初始化句柄，其函数格式如下：

```
SQLAllocHandle( SQL_SMALLINT HandleType, SQLHANDLE InputHandle, SQLHANDLE *OutputHandlePtr );
```

参数：

HandleType：句柄类型，可以设置为以下的数值之一：

SQL_HANDLE_ENV（环境句柄）

SQL_HANDLE_DBC（连接句柄）

SQL_HANDLE_STMT（SQL 语句句柄）

SQL_HANDLE_DESC（描述句柄）

InputHandle：说明新句柄由哪个句柄得出的。

OutputHandlePtr：指向要分配的新句柄。

用 ODBC 完成应用程序后，应使用 SQLFreeHandle 函数释放句柄。还有一点要注意，由于 ODBC 驱动器管理器支持不同版本的驱动程序和应用程序，而当使用不同版本的 ODBC 时，某些函数的作用也不同，这就要求在分配连接句柄前必须说明应用程序的 ODBC 的版本，通过调用 SQLSetEnvAttr 函数实现。

⑶分配一个连接句柄

分配好环境句柄后，接下来的工作是为每一个将要使用的数据源分配一个连接句柄，通过调用 SQLAllocHandle 函数来完成。

⑷连接到指定的数据源

当连接句柄分配完成后，我们可以设置连接属性，所有的连接属性都有缺省值，但是我们可以通过调用函数 SQLSetConnectAttr 来设置连接属性。

一旦完成上述工作后，便可以通过 SQLConnect 函数把该句柄与数据源连接。其函数格式如下：

```
SQLRETURN SQLConnect( SQLHDBC ConnectionHandle, SQLCHAR ServerName, SQLSMALLINT NameLength1, SQLCHAR UserName, SQLSMALLINT NameLength2, SQLCHAR *Authentication, SQLSMALLINT NameLength3 );
```

参数

ConnectionHandle：连接句柄

ServerName：数据源名称

NameLength1：数据源名称长度

UserName：用户名称

NameLength2：用户名称长度

Authentication：用户口令

NameLength3 用户口令长度

⑸执行 SQL 语句

当成功连接到数据源后，便可以通过执行 SQL 语句来完成对数据的各种操作。在执行一个语句前，首先必须通过调用 SQLAllocHandle 函数分配一个 SQL 语句句柄，然后通过调用 SQLExecDirect 函数来执行 SQL 语句。其函数格式如下：

```
SQLRETURN SQLExecDirect(SQLHSTMT StatementHandle,
SQLCHAR * StatementText, SQLINTEGER TextLength);
```

参数

StatementHandle：SQL 语句句柄。

StatementText：SQL 语句字符串。

TextLength：SQL 语句字符串的长度 在 C++ 中一般设置为 SQL_NTS）。

⑹取回查询结果

①绑定列

返回的结果存放在 SQL 语句句柄中，通常称之为结果集，一般来说，从结果集中获取数据的最好办法是提前为结果集中的列与指定的存储区绑定，可以通过调用 SQLBindCol 函数实现，其函数格式如下

```
SQLRETURN SQLBindCol(SQLHSTMT StatementHandle,
SQLUSMALLINT ColumnNumber, SQLSMALLINT Target-
Type, SQLPOINTERT argetValuePtr, SQLINTEGER Buffer-
Length, SQLINTEGER * StrLen_or_IndPtr);
```

参数：

StatementHandle：SQL 语句句柄。

ColumnNumber：列序号 从 1 开始）。

TargetType：目标变量的类型。

TargetValuePtr：目标变量。

BufferLength：目标变量所占的字节数。

StrLen_or_IndPtr：返回值的长度。

②取回一个记录

在绑定列后，通过调用 SQLFetch 函数把每一个记录的各个列的数据复制到与该列绑定的存储区中。这样便可以对存储区的数据进行处理了。其函数格式如下

```
SQLRETURN SQLFetch(SQLHSTMT StatementHandle)
```

参数

StatementHandle：SQL 语句句柄。

⑺断开与数据源的连接

当完成对数据库操作后，调用 SQLDisconnect 函数断开与数据源的连接。如下所示：

```
SQLRETURN SQLDisconnect(SQLHDBC ConnectionHandle);
```

参数：

ConnectionHandle：连接句柄。

⑻释放 ODBC 句柄

调用 SQLFreeHandle 函数释放所有句柄。

```
SQLRETURN SQLFreeHandle(SQLSMALLINT HandleType,
SQLHANDLE Handle)
```

参数：

HandleType：句柄类型。

Handle：要释放的句柄。

三、封装 ODBC API 的类

根据以上的分析，可以设计一个数据库类，取名为 CDataBase，封装了部分 ODBC API 函数，实现对数据库的一般操作。具体实现代码如下：

```
//CdataBase.h
#include "sqlext.h"
#include "ODBCinst.h"
#pragma comment(lib, "ODBC32.lib")
#pragma comment(lib, "ODBCCP32.lib")
class CDataBase //数据库操作类
{
 public:
  CDataBase();
  ~CDataBase();
  BOOL ConfigDataBase(char driver[], char attribute[]);
//配置数据源
  BOOL InitODBC();          //初始化句柄
  BOOL Connect(char ServerName[], char UserName[], char
Password[]); //连接到指定的数据库
  BOOL ExecuteSQL(char Sql[]); //执行 SQL 语句
  BOOL BindCol(int nColNum, int TargetType, void * lpBuf,
int nBufLength); //绑定一列
  int FetchRow();          //取一条记录
  void DisConnect();       //断开连接
  CString GetExecuteErr();
//返回执行 SQL 语句时可能发生的错误信息
  CString GetConnectErr();
//返回连接到数据库过程中可能发生的错误信息
  CString GetInitErr();
//返回在初始化过程中可能发生的错误信息
  CString GetBindErr();
 //返回在绑定列过程中可能发生的错误信息
  CString GetFetchErr();
 //返回取一条记录时可能发生的错误信息
 private:
  SQLHENV hOdbcEnv; //环境句柄
  SQLHDBC hDbConn;  //连接句柄
  SQLHSTMT hStmt;   //SQL 语句句柄
  int nInitErr;     //初始化错误标志，为 0 表示无错误
  int nConnectErr;  //连接错误标志，为 0 表示无错误
int nExecuteErr;  //执行 SQL 语句错误标志，为 0 表示无错误
  int nBindErr;     //绑定列错误标志，为 0 表示无错误
  int nFetchErr; //取一条记录时的错误标志，为 0 表示无错误
```

```
};
// CdataBase.cpp
#include "stdafx.h"
#include "CDataBase.h"
/*构造函数,初始化句柄和变量*/
CDataBase::CDataBase()
{
 hOdbcEnv=SQL_NULL_HANDLE;
 hDbConn=SQL_NULL_HANDLE;
 hStmt=SQL_NULL_HANDLE;
 nInitErr=0;
 nConnectErr=0;
 nExecuteErr=0;
 nBindErr=0;
 nFetchErr=0;
}
/*析构函数,断开与数据源的连接并释放已分配的句柄*/
CDataBase::~CDataBase()
{
      if (hStmt!=SQL_NULL_HANDLE)SQLFreeHandle(SQL_
HANDLE_STMT, hStmt);
 if(hDbConn!=SQL_NULL_HANDLE)
 {
  SQLDisconnect(hDbConn);  //断开与数据源的连接
  SQLFreeHandle(SQL_HANDLE_DBC, hDbConn);
//释放连接句柄
 }
  if(hOdbcEnv!=SQL_NULL_HANDLE) SQLFreeHandle(SQL_
HANDLE_ENV, hOdbcEnv);
}
/*配置数据源*/
//driver:驱动程序名称
//attribute:配置数据源所需的信息(比如 Access 需要 DSN,
DBQ 信息等)
BOOL CDataBase::ConfigDataBase(char driver[], char at-
tribute[])
{
 return SQLConfigDataSource(NULL, ODBC_ADD_SYS_DSN,
driver, attribute);
}
/*分配环境句柄、连接句柄*/
BOOL CDataBase::InitODBC()
{
 SQLRETURN sr;
sr=SQLAllocHandle(SQL_HANDLE_ENV, QL_NULL_HANDLE,
& hOdbcEnv);
 if(sr!=SQL_SUCCESS)
 {
  nInitErr=1;
  return FALSE;
 }
 sr=SQLSetEnvAttr(hOdbcEnv, SQL_ATTR_ODBC_VERSION,
(SQLPOINTER)SQL_OV_ODBC3, SQL_IS_INTEGER);
//设置 ODBC 的版本为 3.0
 if(sr!=SQL_SUCCESS)
 {
  nInitErr=2;
  if(hOdbcEnv!=SQL_NULL_HANDLE)
SQLFreeHandle(SQL_HANDLE_ENV, hOdbcEnv);
//释放先前分配的环境句柄
  return FALSE;
 }
    sr=SQLAllocHandle ( SQL_HANDLE_DBC, hOdbcEnv, &
hDbConn); //分配连接句柄
 if(sr!=SQL_SUCCESS)
 {
  nInitErr=3;
  if(hOdbcEnv!=SQL_NULL_HANDLE)
SQLFreeHandle(SQL_HANDLE_ENV, hOdbcEnv);
  return FALSE;
 }
 //设置连接数据源的超时值为 5 秒
  sr=SQLSetConnectAttr(hDbConn, SQL_ATTR_LOGIN_TIME-
OUT, (void *) 5, 0);
 if(sr!=SQL_SUCCESS)
 {
  nInitErr=4;
  if(hDbConn!=SQL_NULL_HANDLE)
SQLFreeHandle(SQL_HANDLE_DBC, hDbConn);
  if(hOdbcEnv!=SQL_NULL_HANDLE)
SQLFreeHandle(SQL_HANDLE_ENV, hOdbcEnv);
  return FALSE;
 }
 nInitErr=0;
 return TRUE;
}
/*返回在初始化过程中可能发生的错误信息*/
CString CDataBase::GetInitErr()
{
 CString ErrMsg="";
 switch(nInitErr)
 {
 case 1: ErrMsg="分配 ODBC 环境句柄出错"; break;
 case 2: ErrMsg="设置 ODBC 版本(3.0)出错"; break;
 case 3: ErrMsg="分配 ODBC 连接句柄出错"; break;
 case 4: ErrMsg="设置连接数据库的超时值出错"; break;
 }
 nInitErr=0;
 return ErrMsg;
}
/*连接到指定的数据库*/
//ServerName:数据源名称(字符串)
//UserName:用户名(字符串)
//Password:密码(字符串)
BOOL CDataBase::Connect ( char ServerName[], char
UserName[], char PassWord[])
{
 if(hDbConn==SQL_NULL_HANDLE) return FALSE;
```

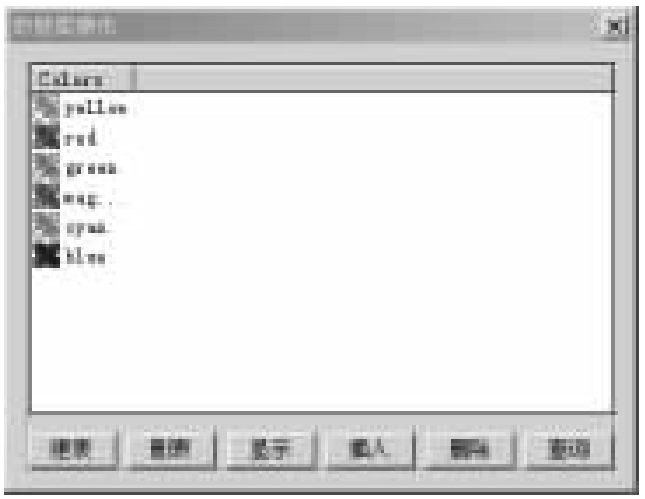
```
 SQLRETURN sr;
  sr = SQLConnect ( hDbConn, (UCHAR * ) ServerName,
SQL_NTS, (UCHAR * ) UserName, SQL_NTS, (UCHAR * )
PassWord, SQL_NTS);
 switch(sr)
 {
  case SQL_SUCCESS: nConnectErr = 0; break;
  case SQL_SUCCESS_WITH_INFO: nConnectErr = 1; break;
  case SQL_ERROR: nConnectErr = -1; break;
  case SQL_INVALID_HANDLE: nConnectErr = -2; break;
 }
 if(nConnectErr>=0) return TRUE;
 else return FALSE;
}
/*返回连接到数据库过程中可能发生的错误信息*/
CString CDataBase: : GetConnectErr()
{
 SQLCHAR SqlState[6];
 SQLINTEGER NativeError;
 SQLCHAR ErrStr[SQL_MAX_MESSAGE_LENGTH];
 int i;
 CString ErrMsg = "";
 switch(nConnectErr)
 {
  case -1: ErrMsg = "连接相应的数据库时发生错误\n";
break;
  case -2: ErrMsg = "数据库的连接句柄还没有被初始化\n";
break;
 }
 if(nConnectErr <0) //获取额外的错误信息
 {
  i = 1;
   while ( SQLGetDiagRec(SQL_HANDLE_DBC, hDbConn, i,
SqlState, & NativeError, ErrStr, sizeof(ErrStr), NULL)! =
SQL_NO_DATA)
    ErrMsg. Format ( ErrMsg + "%d. %X: %s\n", i + +,
SqlState, ErrStr);
 }
 else //取掉操作成功时所返回的附带信息
 {
  i = 1;
 while(SQLGetDiagRec(SQL_HANDLE_DBC, hDbConn, i + +,
SqlState, & NativeError, ErrStr,
sizeof(ErrStr), NULL)! = SQL_NO_DATA);
 }
 nConnectErr = 0;
 return ErrMsg;
}
/*断开连接*/
void CDataBase: : DisConnect()
{
  if (hDbConn! = SQL_NULL_HANDLE) SQLDisconnec(hDb-
Conn);
}
/*执行 SQL 语句*/
BOOL CDataBase: : ExecuteSQL(char Sql[])
{
 if(hStmt! = SQL_NULL_HANDLE)
SQLFreeHandle(SQL_HANDLE_STMT, hStmt);
//释放掉上次所分配的 SQL 语句句柄
 SQLRETURN sr;
   sr = SQLAllocHandle(SQL_HANDLE_STMT, hDbConn, &
hStmt); //分配一个新的 SQl 语句句柄
 switch(sr)
 {
  case SQL_INVALID_HANDLE: nExecuteErr = -1; break;
  case SQL_ERROR: nExecuteErr = -2; break;
  default: nExecuteErr = 0;
//SQL_SUCCESS 或 SQL_SUCCESS_WITH_INFO
 }
 if(nExecuteErr = =0)
 {
  sr = SQLExecDirect(hStmt, (UCHAR * )Sql, SQL_NTS);
//执行 SQL 语句
  switch(sr)
  {
   case SQL_ERROR: nExecuteErr = -3; break;
   case SQL_INVALID_HANDLE: nExecuteErr = -4; break;
   default: nExecuteErr = 0;
//SQL_SUCCESS 或 SQL_SUCCESS_WITH_INFO
  }
 }
 if(nExecuteErr = =0) return TRUE;
 else return FALSE;
}
/*返回执行 SQL 语句时可能发生的错误信息*/
CString CDataBase: : GetExecuteErr()
{
 SQLCHAR SqlState[6];
 SQLINTEGER NativeError;
 SQLCHAR ErrStr[SQL_MAX_MESSAGE_LENGTH];
 int i;
 CString ErrMsg = "";
 switch(nExecuteErr)
 {
  case -1: ErrMsg = "数据库的连接句柄还没有被初始化\n";
break;
  case -2: ErrMsg = "分配 SQL 语句句柄失败\n"; break;
  case -3: ErrMsg = "执行 SQL 语句失败\n"; break;
  case -4: ErrMsg = "SQL 语句句柄还没有被初始化\n";
break;
 }
 if(nExecuteErr <0) //获取额外的错误信息
 {
  i = 1;
   while ( SQLGetDiagRec ( SQL_HANDLE_STMT, hStmt, i,
SqlState, & NativeError, ErrStr, sizeof(ErrStr), NULL)! =
SQL_NO_DATA)
```

```
    ErrMsg.Format ( ErrMsg + "%d.%X:%s\n", i++, SqlState, ErrStr);
 }
 else //取掉操作成功时所返回的附带信息
 {
  i=1;
  while(SQLGetDiagRec(SQL_HANDLE_STMT, hStmt, i++, SqlState, & NativeError, ErrStr, sizeof(ErrStr), NULL)!=SQL_NO_DATA);
 }
 nExecuteErr=0;
 return ErrMsg;
}
/*绑定一列,指定从数据库返回的数据格式和存放的缓冲区*/
//nColNum:第几列,从左到右依次为1,2,3...
//TargetType:目标数据类型的标识(例如SQL_C_CHAR,可参考MSDN)
//lpBuf:目标缓冲区的起始地址
//nBufLength:目标缓冲区的大小
BOOL CDataBase::BindCol(int nColNum, int TargetType, void *lpBuf, int nBufLength)
{
 SQLINTEGER len;
 SQLRETURN sr;
   sr=SQLBindCol(hStmt, (SQLUSMALLINT) nColNum, (SQLSMALLINT) TargetType, (SQLPOINTER) lpBuf, (SQLINTEGER)nBufLength, & len);
 switch(sr)
 {;
  case SQL_SUCCESS:
  case SQL_SUCCESS_WITH_INFO: nBindErr=0; break;
  case SQL_ERROR: nBindErr=-1; break
  case SQL_INVALID_HANDLE: nBindErr=-2; break;
 }
 if(nBindErr==0) return TRUE;
 else return FALSE;
}
/*返回在绑定列过程中可能发生的错误信息*/
CString CDataBase::GetBindErr()
{
 SQLCHAR SqlState[6];
 SQLINTEGER NativeError;
 SQLCHAR ErrStr[SQL_MAX_MESSAGE_LENGTH];
 int i;
 CString ErrMsg="";
 switch(nBindErr)
 {
  case -1: ErrMsg="绑定列时出错\n"; break;
  case -2: ErrMsg="SQL 语句句柄还没有被初始化\n\n"; break;
 }
 if(nBindErr==-1) //获取额外的错误信息
 {
  i=1;
  while ( SQLGetDiagRec ( SQL_HANDLE_STMT, hStmt, i, SqlState, & NativeError, ErrStr, sizeof(ErrStr), NULL)!=SQL_NO_DATA)
    ErrMsg.Format ( ErrMsg + "%d.%X:%s\n", i++, SqlState, ErrStr);
 }
 else //取掉操作成功时所返回的附带信息
 {
  i=1;
  while(SQLGetDiagRec(SQL_HANDLE_STMT, hStmt, i++, SqlState, & NativeError, ErrStr, sizeof(ErrStr), NULL)!=SQL_NO_DATA);
 }
 nBindErr=0;
 return ErrMsg;
}
/*取一条记录*/
int CDataBase::FetchRow()
{
 SQLRETURN sr;
 sr=SQLFetch(hStmt); //取一条记录
 switch(sr)
 {
  case SQL_ERROR: nFetchErr=-1; break;
  case SQL_INVALID_HANDLE: nFetchErr=-2; break;
  case SQL_NO_DATA: nFetchErr=1; break; //取完所有记录
  default: nFetchErr=0;
//SQL_SUCCESS 或 SQL_SUCCESS_WITH_INFO
 }
 return nFetchErr;
}
/*返回取一条记录时可能发生的错误信息*/
CString CDataBase::GetFetchErr()
{
 SQLCHAR SqlState[6];
 SQLINTEGER NativeError;
 SQLCHAR ErrStr[SQL_MAX_MESSAGE_LENGTH];
 int i;
 CString ErrMsg="";
 switch(nFetchErr)
 {
  case -1: ErrMsg="取一条记录出错\n"; break;
  case -2: ErrMsg="SQL 语句句柄还没有被初始化\n"; break;
 }
 if(nFetchErr==-1) //获取额外的错误信息
 {
  i=1;
  while ( SQLGetDiagRec ( SQL_HANDLE_STMT, hStmt, i, SqlState, & NativeError, ErrStr, sizeof(ErrStr), NULL)!=SQL_NO_DATA)
    ErrMsg.Format ( ErrMsg + "%d.%X:%s\n", i++, SqlState, ErrStr);
```

```
}
else //取掉操作成功时所返回的附带信息
{
 i=1;
  while(SQLGetDiagRec(SQL_HANDLE_STMT, hStmt, i++, SqlState, & NativeError, ErrStr, sizeof(ErrStr), NULL)!= SQL_NO_DATA);
}
nFetchErr=0;
return ErrMsg;
}
```

四、使用自己编写的 CDataBase 类来编写一个简单的数据库程序

(1)利用 VC++的 App Wizard 新建一个基于对话框的工程，取名为 student，设置对话框的 Caption 为“数据库操作”，删除由 App Wizard 创建的控件。添加 1 个 List Control 控件和 6 个 Button 控件，List Control 控件的 ID 为 IDC_LIST_STUDENT，设置控件的属性：在 Style/View 选择 Report，同时选中 Single Selection，其他取默认值；最后利用 MFC Class Wizard 为该控件添加一个相关联的变量 m_ListCtrl，其类型为 ClistCtrl (Control)。添加 6 个按钮，ID 分别为 IDC_BUTTON_CREATETABLE (建表)、IDC_BUTTON_DROP-TABLE (删表)、IDC_BUTTON_DISPLAY (显示)、IDC_BUTTON_INSERT (插入)、IDC_BUTTON_DELETE (删除)、IDC_BUTTON_SELECT (查询)。如图所示：



(2)将自定义的类 (CDataBase.h 和 CDataBase.cpp 文件)加入到工程中，同时为工程添加一个全局变量 CDataBase db (定义在 StudentDlg.cpp 文件中)。在 StudentDlg.h 文件中定义一个 student 结构体如下：

```
struct Student //定义一个学生结构体
{
    char Name[11]; //姓名最多 5 个汉字, 10 个字母
    char Num[7];  //学号 6 位
    char Sex[3];  //性别取值为'男'或'女'
    long Age;     //年龄
};
```

(3)为对话框 (CStudentDlg)添加如下的成员函数：

```
void CStudentDlg::InitDataBase() //初始化数据库
{
char driver[]="Microsoft Access Driver (*.mdb)\0";
//驱动程序名称
char createdb[]="CREATE_DB=C:\\student.mdb\0";
//创建数据库文件
char attribute[]="DSN=student\0" //数据源名称
 "DBQ=student.mdb\0"
//Access 文件(*.mdb)所在的路径
 "DEFAULTDIR=C:\\"; //默认文件夹
db.ConfigDataBase(driver, createdb);
 //创建 Access 文件 student.mdb
if(!db.ConfigDataBase(driver, attribute))
//配置 Access 的数据源
  {
MessageBox("配置数据源失败.", "错误信息", MB_OK);
SendMessage(WM_CLOSE);
//连接失败则发"关闭"消息给对话框, 使之立即关闭
  return;
}
if(!db.InitODBC()) //初始化 db
{
MessageBox(db.GetInitErr(), "错误信息", MB_OK);
 return;
}
if(!db.Connect("student", "", ""))//连接到数据源 student
  {
   MessageBox(db.GetConnectErr(), "错误信息", MB_OK);
   SendMessage(WM_CLOSE);
   return;
  }
}
void CStudentDlg::InitListCtrl()
//初始化列表控制(List Control)
{
  LV_COLUMN lvcol;
  lvcol.mask=LVCF_FMT | LVCF_SUBITEM | LVCF_TEXT | LVCF_WIDTH;
   lvcol.fmt=LVCFMT_CENTER;
//标题栏各列的文字居中显示, 第一列只能是左对齐
       lvcol.pszText="姓名";
   lvcol.iSubItem=0;
   lvcol.cx=80;
   m_ListCtrl.InsertColumn(0, & lvcol); //插入第一列
  lvcol.pszText="学号";
   lvcol.iSubItem=1;
   lvcol.cx=60;
   m_ListCtrl.InsertColumn(1, & lvcol); //插入第二列
  lvcol.pszText="性别";
   lvcol.iSubItem=2;
   lvcol.cx=50;
```

```
    m_ListCtrl. InsertColumn(2, & lvcol); //插入第三列
      lvcol. pszText = "年龄";
    lvcol. iSubItem = 3;
    lvcol. cx = 50;
    m_ListCtrl. InsertColumn(3, & lvcol); //插入第四列
}
void CStudentDlg: : UpdateListCtrl()
//在列表视图中显示 student 表中的记录
{
      Student s;
      //绑定各个列
if(!db. BindCol(1, SQL_C_CHAR, s. Name, sizeof(s. Name)))
MessageBox(db. GetBindErr(), "错误信息", MB_OK);
      if( ! db. BindCol(2, SQL_C_CHAR, s. Num, sizeof
(s. Num)))
MessageBox(db. GetBindErr(), "错误信息", MB_OK);
      if(!db. BindCol(3, SQL_C_CHAR, s. Sex, sizeof(s. Sex)))
MessageBox(db. GetBindErr(), "错误信息", MB_OK);

if(!db. BindCol(4, SQL_C_SLONG, & s. Age, sizeof(s. Age)))
MessageBox(db. GetBindErr(), "错误信息", MB_OK);
      CString age;
      int i = 0, j;
      m_ListCtrl. DeleteAllItems(); //清空列表视图
 · LV_ITEM lvitem;
      while((j = db. FetchRow()) = = 0)
      {
//取一条记录, 直到取完为止(返回值为 1), 返回值为 0 表此
次操作成功
       lvitem. mask = LVIF_TEXT|LVIF_PARAM;
       lvitem. iItem = i;
       lvitem. iSubItem = 0;
       lvitem. pszText = s. Name;
       lvitem. lParam = i;
       m_ListCtrl. InsertItem(& lvitem); //姓名
       lvitem. mask = LVIF_TEXT;
       lvitem. iItem = i;
       lvitem. iSubItem = 1;
       lvitem. pszText = s. Num;
       m_ListCtrl. SetItem(& lvitem); //学号
       lvitem. iSubItem = 2;
       lvitem. pszText = s. Sex;
       m_ListCtrl. SetItem(& lvitem); //性别
       lvitem. iSubItem = 3;
       age. Format("%d", s. Age);
       lvitem. pszText = age. GetBuffer(age. GetLength());
       m_ListCtrl. SetItem(& lvitem); //年龄
       i + + ;
      }
      if(j < 0) //取一条记录时发生错误
      {
       MessageBox(db. GetFetchErr(), "错误信息", MB_OK);
       return;
      }
}
```

(4)在 CStudentDlg OnInitDialog 函数中添加如下代码：

```
BOOL CStudentDlg: : OnInitDialog()
{
  // TODO: Add extra initialization here
  InitListCtrl();
  InitDataBase();
  . . . . . .
}
```

(5)添加 2 个对话框，分别为 "查询" 对话框和 "插入记录" 对话框，如下图所示：

用 MFC ClassWizard 为 "查询" 对话框添加一个类 CselectDlg 由 CDialog 公有派生)，对应的文件为 SelectDlg. h 和 SelectDlg. cpp；为 "插入记录" 对话框添加一个类 CInsertDlg 由 CDialog 公有派生)，对应的文件为 InsertDlg. h 和 InsertDlg. cpp。

①为 "查询" 对话框添加 1 个 Static 控件，1 个 Edit Box 控件，2 个 Button 控件，分别为 IDC_BUTTON_OK 确定) 和 IDC_BUTTON_CANCEL 取消)；利用 MFC Class Wizard 为 Edit Box 控件添加一个相关联的变量 m_strName，其类型为 CString (Value)。为类添加一个成员函数 CString GetName 。具体代码如下：

```
void CSelectDlg: : DoDataExchange(CDataExchange * pDX)
```

```
{
    CDialog::DoDataExchange(pDX);
    //{{AFX_DATA_MAP(CSelectDlg)
    DDX_Text(pDX, IDC_EDIT_NAME, m_strName);
    DDV_MaxChars(pDX, m_strName, 10);
    //}}AFX_DATA_MAP
}
void CSelectDlg::OnButtonOk()
{
    UpdateData(); //将控件的内容赋给相对应的变量 m_strName
    m_strName.TrimLeft(); //去掉字符串左边的空格符
    m_strName.TrimRight(); //去掉字符串右边的空格符
    if(m_strName.IsEmpty())
    {
MessageBox("输入要查询的姓名.","警告", MB_OK | MB_ICONQUESTION);
        return;
    }
    else CDialog::EndDialog(0); //关闭对话框
})
void CSelectDlg::OnButtonCancel(
{
    m_strName.Empty(); //将字符串置空
    CDialog::EndDialog(0); //关闭对话框
}
CString CSelectDlg::GetName() //返回用户输入的姓名
{
 return m_strName;
}
```

②为“插入记录”对话框添加 4 个 Static 控件，3 个 Edit Box 控件，分别为 IDC_EDIT_NAME、IDC_EDIT_NUM 和 IDC_EDIT_AGE，1 个 Combo Box 控件，其 ID 为 IDC_COMBO_SEX，2 个 Button 控件，分别为 IDC_BUTTON_OK（确定）和 IDC_BUTTON_CANCEL（取消）。利用 MFC Class Wizard 为 3 个 Edit Box 控件分别添加相关联的变量 m_strName（Cstring）、m_strNum（Cstring）和 m_strAge（UINT）；为 Combo Box 控件添加一个相关联的变量 m_ComboCtrl（CComboBox）。为类添加一个成员变量 CString student 和一个成员函数 CString GetStudent。具体代码如下：

```
void CInsertDlg::DoDataExchange(CDataExchange * pDX)
{
    CDialog::DoDataExchange(pDX);
    //{{AFX_DATA_MAP(CInsertDlg)
    DDX_Control(pDX, IDC_COMBO_SEX, m_ComboCtrl);
    DDX_Text(pDX, IDC_EDIT_NAME, m_strName);
    DDV_MaxChars(pDX, m_strName, 10);
    DDX_Text(pDX, IDC_EDIT_NUM, m_strNum);
    DDV_MaxChars(pDX, m_strNum, 6);
    DDX_Text(pDX, IDC_EDIT_AGE, m_strAge);
    //}}AFX_DATA_MAP
}
BOOL CInsertDlg::OnInitDialog()
{
  CDialog::OnInitDialog();
  // TODO: Add extra initialization here
  //给组合框添加两项
  m_ComboCtrl.AddString("男"); //第一项
  m_ComboCtrl.AddString("女"); //第二项
  m_ComboCtrl.SetCurSel(0);
//设置组合框的当前值为第一项
  return TRUE;
// return TRUE unless you set the focus to a control
// EXCEPTION: OCX Property Pages should return FALSE
}
CString CInsertDlg::GetStudent()
{
return student; //返回插入记录的 SQL 语句
})
void CInsertDlg::OnButtonCancel(
{
  student.Empty();
  CDialog::EndDialog(0);
}
void CInsertDlg::OnButtonOk()
{
UpdateData(); //将各个控件的内容赋值给相对应的变量
m_strName.TrimLeft(); //去掉字符串左边的空格
m_strName.TrimRight(); //去掉字符串右边的空格
m_strNum.TrimLeft();
m_strNum.TrimRight();
CString msg="";
if(m_strName.IsEmpty()) msg="姓名不能为空\t\n";
if(m_strNum.IsEmpty()) msg+="学号不能为空\t\n";
if(m_strAge<10 || m_strAge>30) msg+="年龄应为 10 到 30 之间的一个整数\t";
if(!msg.IsEmpty())
{
MessageBox(msg,"警告",MB_OK|MB_ICONQUESTION);
return;
}
CString s1;
m_ComboCtrl.GetLBText(m_ComboCtrl.GetCurSel(),s1);
//获取组合框的当前值
student.Format("Insert Into student Values('%s','%s','%s',%d)",m_strName, m_strNum,s1,m_strAge);
//格式化 SQL 语句
  CDialog::EndDialog(0);
}
```

(6)在 StudentDlg.cpp 文件加入

```
#include "SelectDlg.h" //查询记录对话框的头文件
#include "InsertDlg.h" //插入记录对话框的头文件
```

(7)利用 MFC ClassWizard 为对话框 StudentDlg 的 6 个 Button 控件添加相应的单击响应函数 具体代码如下

```
void CStudentDlg::OnButtonCreatetable() //建 student 表
{
```

```
char sql[] = "Create Table student( "//创建 student 表
"name CHAR(10) NOT NULLUNIQUE, "// 姓名不为空且不重名
"num CHAR(6) NOT NULL, "//学号
"sex CHAR(2) NOT NULL, "//性别
"age INTEGER NOT NULL, "//年龄
"PRIMARY KEY(num))"; //设置主键为学号
    if(!db. ExecuteSQL(sql))
    MessageBox(db. GetExecuteErr( ), " 错 误 信 息 ", MB_OK);
                    else //建表成功
    {
    //插入四条记录
  if (!db. ExecuteSQL ( "Insert into student values('约翰', '200001', '男', 20)"))
MessageBox(db. GetExecuteErr(), "错误信息", MB_OK);
if ( !db. ExecuteSQL ( "Insert into student values ('比尔', '200002', '男', 21)"))
MessageBox(db. GetExecuteErr(), "错误信息", MB_OK);
 if ( !db. ExecuteSQL ( "Insert into student values('杰克', '200003', '男', 23)"))
MessageBox(db. GetExecuteErr(), "错误信息", MB_OK);
  if(!db. ExecuteSQL("Insert into student values('艾丽思', '200004', '女', 21)"))
MessageBox(db. GetExecuteErr(), "错误信息", MB_OK);
OnButtonDisplay(); //在列表视图中显示 student 表中的记录
    }
}
void CStudentDlg: : OnButtonDroptable() //删除 student 表
{
if(!db. ExecuteSQL("Drop Table student"))//删除 student 表
    MessageBox(db. GetExecuteErr( ), " 错 误 信 息 ", MB_OK);
                    m_ListCtrl. DeleteAllItems(); //列表视图清空
}
void CStudentDlg: : OnButtonDelete()
//删除 student 表中指定的记录
{
    CString name;
  int index = m_ListCtrl. GetNextItem( -1, LVNI_SELECTED);
//获取列表视图被选中的记录
    if(index>=0)
//index 表示被选中的记录的索引值(第一条对应为 0)
    {
name = m_ListCtrl. GetItemText(index, 0);
//获取被选中的记录的第一项(姓名)
name. Format( " Delete From student Where name = '" + name + "'"); //格式化 SQL 语句
if(!db. ExecuteSQL(name. GetBuffer(name. GetLength())))
//执行删除操作
  MessageBox(db. GetExecuteErr(), "错误信息", MB_OK);
        else OnButtonDisplay(); //更新列表视图
    }
}
void CStudentDlg: : OnButtonInsert()
//向 student 表中插入一条记录
{
CInsertDlg dlg; //定义一个查询记录对话框的对象
 dlg. DoModal(); //显示插入记录对话框
 CString sql;
 sql = dlg. GetStudent(); //取回插入一条记录的 SQL 语句
 if(sql. IsEmpty()) return;    //为空则不执行
 if(!db. ExecuteSQL(sql. GetBuffer(sql. GetLength())))
//执行插入操作
 MessageBox(db. GetExecuteErr(), "错误信息", MB_OK);
 else OnButtonDisplay();
//显示插入一条记录后 student 表中的记录
}
void CStudentDlg: : OnButtonSelect()  //查询特定的记录
{
CSelectDlg dlg; //定义一个查询记录对话框的对象
dlg. DoModal(); //显示查询记录对话框
CString name;
name = dlg. GetName(); //取得用户输入的姓名
if(!name. IsEmpty()) //不为空则根据姓名查询 student 表
{
  CString sql;
  //格式化 SQL, 采用模糊查询
sql. Format( " Select name, num, sex, age From student Where name Like '%%" + name + "%%'");
if(!db. ExecuteSQL(sql. GetBuffer(sql. GetLength())))
//执行查询操作
 MessageBox(db. GetExecuteErr(), "错误信息", MB_OK);
  else UpdateListCtrl(); //执行成功则显示结果
}
}
void CStudentDlg: : OnButtonDisplay()
//显示 student 表中的记录
{
  if ( !db. ExecuteSQL( "Select name, num, sex, age From student"))
MessageBox(db. GetExecuteErr(), "错误信息", MB_OK);

   else UpdateListCtrl();     //更新列表视图
}
```

程序在 VC 6. 0 + Win 98 / SE / ME / 2000 下运行成功

参考资料

1. 微软 MSDN Library

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

3. 王华、叶爱亮等编著 . Visual C + + 6. 0 编程实例与技巧 . 机械工业出版社

收稿日期：2000 年 12 月 25 日）

网络连通性测试与网络扫描技术

张春明 姜绍飞 张春丽

通常我们使用 Ping 命令测试网络的连通性。它的工作机理是：首先向目标主机的 IP 堆栈发出一个 ICMP（Internet 控制消息协议）回波报文 Echo，即回应请求报文。然后根据能否收到一个由目标主机返回的 ICMP 回波应答报文 Echo Reply，来判断出目标主机是否正在运行。

由于 Ping 命令是一个 DOS 程序，无法直接用在自己编写的程序中，而采用 Raw Socket 方法编写同样功能的 Windows 程序又太复杂，因此，这里给出一种简便方法，即通过调用 Windows 9x/NT 自带的 ICMP.DLL 来测试网络的连通性。此外，利用网络连通性测试很容易实现网络扫描，即查找某个网段上正在运行的所有主机 IP 地址，甚至主机名（注：安装了防火墙的主机不在本文讨论之列）。

一、编写网络连通性测试程序的一种简便方法

1. 首先介绍 ICMP.DLL 中三个相关 API 函数的用法

① IcmpCreateFile 函数用于打开一个 ICMP 句柄，以便发送 ICMP 回波请求。调用成功时将返回一个打开的句柄，否则返回 INVALID_HANDLE_VALUE，这时可以调用 GetLastError 函数获得更详细的出错信息。函数声明如下：

```
function IcmpCreateFile: THandle; stdcall; external 'ICMP.DLL';
```

② IcmpCloseHandle 函数用于关闭由 ICMPOpenFile 函数打开的一个 ICMP 句柄。关闭成功则返回 TRUE，否则返回 FALSE，这时可以调用 GetLastError 函数获得更详细的出错信息。函数声明如下：

```
function IcmpCloseHandle(
    IcmpHandle: THandle      // 待关闭的 ICMP 句柄
    ): Boolean; stdcall; external 'ICMP.DLL';
```

③ IcmpSendEcho 函数发出一个 ICMP 回波请求，并等待接收一个或多个回波应答。请求超时或回波应答缓冲区已满时，函数返回已接收并存储到 ReplyBuffer 的回应次数。返回值为零时，可以调用 GetLastError 函数获得更详细的出错信息。函数声明如下：

```
function IcmpSendEcho(
IcmpHandle: THandle; // 用 ICMPCreateFile 函数打开的 ICMP 句柄
DestinationAddress: DWord; // 目标主机地址
RequestData: Pointer;      // 回波请求所发数据的缓冲区
RequestSize: Word; // 回波请求数据缓冲区大小(以字节计)
RequestOptions: PIP_Option_Information;  // 回波请求中 IP 报头选项地址,可以为空
ReplyBuffer: Pointer;  // 用于存储回波应答数据的缓冲区
ReplySize: DWord;    // 回波应答缓冲区大小(以字节计)
Timeout: Dword       // 等待回应的时间(以毫秒计)
): DWord; stdcall; external 'ICMP.DLL';
```

其中 ReplyBuffer 回波应答缓冲区的存储结构如下：

ICMP_ECHO_REPLY 类型结构 + 附加数据

附加数据至少为 8 个字节，这是一个 ICMP 出错信息的大小。

2. 其次再介绍一下 ICMP 应用程序接口中的一些数据结构

```
TIP_Option_Information = packed record
TTL: Byte;          // 存活时间(用于路由跟踪)
TOS: Byte;          // 服务类型(通常为 0)
Flags: Byte;        // IP 头标志(通常为 0)
OptionsSize: Byte; // 附加数据大小(通常为 0,最大为 40)
OptionsData: PChar;                    // 附加数据
end;
TIcmp_Echo_Reply = packed record
Address: DWord;     // 应答的主机地址
Status: DWord;      // IP 状态码
RTT: DWord;         // 往返旅行时间(以毫秒计)
DataSize: Word;     // 回波应答数据大小(以字节计)
Reserved: Word;     // 系统保留
Data: Pointer;      // 回波应答数据指针
Options: TIP_Option_Information;  // 回波应答参数
end;
PIP_Option_Information = ^TIP_Option_Information;
PIcmp_Echo_Reply = ^TIcmp_Echo_Reply;
```

3. Ping 子程序实现方法

由于程序用到了一些 Winsock 函数，因此，必须在窗体中引用 Winsock 单元，即 uses Winsock。

在使用任何 Winsock 函数之前必须调用 WSAStartup 函数进行初始化，并在使用完毕后调用 WSACleanup 函数进行清理。

Ping 程序的工作流程如下：首先打开一个 ICMP 句柄，并

进行动态内存分配。然后发送回波请求，并等待回波应答。最后关闭打开的 ICMP 句柄，释放分配的内存空间。

```
const
  PacketSize = 32;  // 发送的数据包大小(以字节计)
TimeOut = 5000;          // 超时设定(以毫秒计)
procedure Ping(TheIPAddress: string);
var
WSAData: TWSAData;          // Winsock 数据结构
  DestAddress: DWord;   // 目标主机 IP 地址
  RequestDataBuffer: Pointer;   // 请求数据缓冲区指针
  ReplyDataBuffer: Pointer;     // 应答数据缓冲区指针
ICMPEchoReplyBuffer: PIcmp_Echo_Reply;
// ICMP 回波应答缓冲区
IPOptionInfo: TIP_Option_Information;
// 待发送数据包的 IP 选项
begin
if WSAStartup($102, WSAdata) <>0 then // 初始化 Winsock
  begin
    ShowMessage('Winsock 初始化失败!');
    Exit;
  end;
  ICMPHandle := IcmpCreateFile;    // 打开 ICMP 句柄
If ICMPHandle = INVALID_HANDLE_VALUE then//错误处理
  begin
    ShowMessage('无法获得 ICMP 句柄!');
    Exit;
  end;
  DestAddress := inet_addr(PChar(TheIPAddress));
// 将目标地址转换成网络格式
  GetMem(RequestDataBuffer, PacketSize);
// 分配请求数据缓冲区
  FillChar(RequestDataBuffer^, PacketSize, $FF);
// 填充请求数据缓冲区
  FillChar(IPOptionInfo, SizeOf(IPOptionInfo), 0);
// 填充 IP 选项数据
  IPOptionInfo.TTL := 64;
// 设置存活期
  GetMem(ReplyDataBuffer, PacketSize);
      // 分配应答数据缓冲区
// 分配回波应答结构缓冲区
 GetMem(ICMPEchoReplyBuffer, SizeOf(TIcmp_Echo_Reply)
 + PacketSize);
  ICMPEchoReplyBuffer^.Data := ReplyDataBuffer;
// 填入缓冲区指针
  IcmpSendEcho(ICMPHandle, DestAddress, // 发送回波请求,并等待回波应答
              RequestDataBuffer, PacketSize,
              @IPOptionInfo, ICMPEchoReplyBuffer,
       SizeOf(TIcmp_Echo_Reply) + PacketSize, TimeOut);
  ShowMessage('向' + TheIPAddress + // 显示测试结果
 '地址发送了' + IntToStr(PacketSize) + '字节数据,' + #
10#13 +'在' + IntToStr(ICMPEchoReplyBuffer^.RTT) +
'毫秒内从' +StrPas(inet_ntoa ( TInAddr ( ICMPEchoReply-
Buffer^.Address))) +'接收了' + IntToStr(ICMPEchoReply-
Buffer^.DataSize) + '字节.');
  FreeMem(ICMPEchoReplyBuffer); // 释放分配的内存空间
  FreeMem(ReplyDataBuffer);
  FreeMem(RequestDataBuffer);
  IcmpCloseHandle(ICMPHandle); // 关闭 ICMP 句柄
  if WSACleanup <>0 then        // 关闭 Winsock
ShowMessage('无法关闭 winsock!');
end;
```

限于篇幅，程序大部分未进行出错处理。读者可以调用 GetLastError 函数获得更详细的出错信息。常见的错误信息有不可到达的网络、主机、协议、端口和请求超时等。

二、网络扫描的实现方法

利用上述 Ping 子程序可以轻而易举地实现网络扫描，找出网络上正在运行的每个主机。具体方法是针对 IP 地址的每个字节分别从 1 到 255 循环调用 Ping 过程。例如，下面的代码完成 202. 118. 0. 1 ~ 202. 118. 0. 255 网段的扫描。

```
For i := 1 to 255 do
Ping '202. 118. 0. ' + IntToStr i
```

除了查找正在运行的主机外，还可以根据 IP 地址解析出主机名。方法如下：

首先声明一个用于名字解析的 HostEntry 结构缓冲区指针变量。注意：该缓冲区空间由 Windows Sockets 自动进行分配和释放。

```
var
  PHostEntry: PHostEnt; // 用于名字解析的 HostEntry 结构缓冲区
```

然后调用 GetHostByAddr 函数解析出主机名。

```
DestAddress := inet_addr(PChar(TheIPAddress));
PHostEntry: =GetHostByAddr(@DestAddress, 4, PF_INET);
if PHostEntry <> nil then
    begin
  HostName := PHostEntry^.h_name;
      HostIP := StrPas(inet_ntoa(TInAddr(DestAddress)));
end;
```

反过来，也可以使用 GetHostByName 函数根据主机名字解析出 IP 地址。

```
PHostEntry := GetHostByName(PChar(TheIPAddress));
if PHostEntry <> nil then
    begin
    DestAddress := LongInt(PLongInt(PHostEntry^.h_addr_
list^)^);
  HostName := PHostEntry^.h_name;
  HostIP := StrPas(inet_ntoa(TInAddr(DestAddress)));
end;
```

收稿日期 2000 年 12 月 15 日

使用 TAPI 在 Windows98 下对调制解调器编程

刘 鹏

随着计算机的普及和通信技术的飞速发展，越来越多的人通过电话线接入国际互连网，来享受信息高速公路所带来的诸如信息共享等各种好处。基于 MMX 技术的奔腾处理器的推出后，电话线上的可视电话被越来越多的人接受和认可。此外还有许多单位开始通过电话线实现工厂的无人值班、远程诊断维护和管理自动化等。所有这一切都离不开调制解调器。

在 DOS 下编制 Modem 通信程序时，我们经常使用 Hayes 兼容的 AT 命令集来生成呼叫，AT 命令就是写向已知有调制解调器相连的 COM 口的 ASCII 字符串，即用于控制调制解调器的标准语言。但由于各调制解调器生产厂家对 AT 命令集都做了各自的扩展，所以这种方法通用性不强，只能使用一部分调制解调器，而各厂家都提供相应的驱动程序，使用 TAPI 编制的应用程序则可以使用大部分的调制解调器。

Windows 为硬件的操作提供了方便的编程接口（API），Win32 的应用程序接口（API）提供的通信手段大致分为以下几类：

1. 基 TCP / IP 协议 Winsock API，可实现局域网上或互联网上的微机通信；

2. 基于进程之间的通信技术：动态数据交换（DDE）；

3. 基于直接电缆连接的通信技术，可直接操作串行口、并行口以及远红外线接口；

4. 基于电话线路的通信应用程序接口（TAPI / Telephony API），可方便地控制调制解调器。

从目前的发展状况看来，调制解调器已经成为远距离通信的一种重要工具，为此 Microsoft 及 Intel 公司联合开发了 TAPI 这样一个编程接口，而且，使用 API 函数编制的程序段既适用于 Borland C + + 编译器，同时也能插入 Visual C + + 程序中编译运行。Win32 通信 API 使得用户在使用 MODEM 和其它通信设备时可以不必关心设备的具体情况，即设备无关性。通信应用程序使用 Win32 通信 API 设置 MODEM 和传输数据，使用 Win32 TAPI 控制连接。TAPI 服务为通信应用程序提供了一种与设备无关的方法，使应用程序可以与各种通信网络进行通信，包括 ISDN 等；同时还提供了访问通信端口和设备的通用机制。这样多个应用程序就可以共享这些设备：TAPI 管理对每一个设备的访问，让程序协同使用这些设备。TAPI 的设计又使它可以扩展，从而满足未来通信网络的新要求。

使用 TAPI 过程分为以下几步：

1. 初始化线路，通过使用 lineInitialize 函数初始化 TAPI. DLL 得 TAPI 使用句柄的指针 hTapi，注意参数中回调函数的定义（所有提及函数的用法均可在 MSDN 中查到），通过调 lineOpen 函数（用到参数 hTapi）获得线路句柄 hLine，再利用 lineGetID（用到参数 hLine）获取调制解调器句柄 hModem；

2. 配置线路（可选）；

3. 拨号，呼叫方执行，使用 lineMakeCall 函数（用到 hLine 进行拨号，完成后获得呼叫句柄 hCall（呼叫方的呼叫句柄）；

4. 应答链接（应答方执行），被呼叫放的回调函数得到 LINECALLSTATE_OFFERING 消息时调用 lineAnswer 函数实现自动应答；

5. 数据通信（双方），当回调函数收到 LINECALLSTATE_CONNECTED 消息后，先清除接收缓冲区，便可以使用函数为 WriteFile 及 ReadFile 函数进行数据交换，注意参数 hFile 为调制解调器句柄 hModem；

6. 挂机（任一方），通信完毕任何一方都可以调用函数 lineDrop hCall NULL 0 来停止呼叫，该函数还发送 LINECALLSTATE_IDLE 消息给回调函数；

7. 关闭线路（双方），通信双方的回调函数在收到 LINECALLSTATE_IDLE 消息时都应该调用函数 lineDeallocateCall hCall 释放相应呼叫占用的资源；当回调函数收到 LINECALLSTATE_DISCONNECTED 消息时使用 lineClose hLine 释放由 lineOpen 分配的资源，调用 lineShutDown hTapi 释放为线路设备分配的资源。

下面是一个基于对话框的 TAPI 的应用程序 Tapi。

利用 AppWizard 在 CtapiDlg 类中加入成员变量 CTapiConnect m_tapiObj 和 BOOL m_bCallInProgress，CTapiConnect 是为使用 TAPI 定义的新类。m_bCallInProgress 是一个控制量。呼叫按钮对应的函数：

```
void CTapiDlg: : OnDial()
{
  // TODO: Add your control notification handler code here
   m_bCallInProgress = TRUE;
    GetDlgItemText ( IDC_TAPIEDIT, m_tapiObj. m_szPhone-
Number, 32);
  if (!m_tapiObj. DialCall(m_tapiObj. m_szPhoneNumber))
      OutputDebugString(˝Dial call failed. \n˝);
  //IDC_TAPIEDIT 为对话框编辑框 ID
}
```

挂起按钮函数

```
void CTapiDlg: : OnHangup()
{
  // TODO: Add your control notification handler code here
```

```
    if (m_bCallInProgress)
     m_tapiObj. HangupCall();
    EndDialog(ID_HANGUP);
}
```

在初始化对话框响应函数中加入

```
BOOL CTapiDlg: : OnInitDialog()
{
    CDialog: : OnInitDialog();
    // Add ″About. . . ″ menu item to system menu.
(Generated by ClassWizard 省略)
    // TODO: Add extra initialization here
    //Initialize TAPI line
    m_tapiObj. Create();
    return TRUE;  // return TRUE unless you set the focus to a control
}
```

我们可以大概看出 CtapiConnect 的成员函数 Create DialCall HangupCall 是分别用来完成初始化 拨号和挂起操作的。

CtapiConnect 类中的成员变量和成员函数有：

```
Public:
   BOOL Create();
   BOOL DialCall(char * szPhoneNumber = NULL);
   BOOL HangupCall();
    static void CALLBACK lineCallbackFunc(DWORD dwDevice, DWORD dwMsg, DWORD dwCallbackInstance, DWORD dwParam1, DWORD dwParam2, DWORD dwParam3);
 protected:
 // this area contains the protected members of the CTapiConnect class
   DWORD m_dwNumDevs;   // 可获得的线路设备数
   DWORD m_dwDeviceID;  // 设备 ID
    // BOOLEANS to handle reentrancy
   BOOL m_bShuttingDown;  //whether shutdown the application´s usage of line
   BOOL m_bStoppingCall;  //whether stopping the call
   BOOL m_bInitializing;  //whether initiallizing
   BOOL m_bReplyReceived;  //whether received replay
   BOOL m_bTapiInUse;  // whether TAPI is in use or not
BOOL m_bInitialized;  // whether TAPI has been initialized
 public:
   // this area contains the public members of the CTapiConnection class
   HLINEAPP m_hLineApp;  // API 句柄
   HCALL m_hCall;       // 呼叫句柄
   HLINE m_hLine;       // 线路句柄
   DWORD m_dwAPIVersion;   // the API version
   char m_szPhoneNumber[64];  // 呼叫号码
 protected:
   // Here is where I put the protected (internal) functions
   BOOL ShutdownTAPI();
   BOOL HandleLineErr(long lLineErr);D
 LPLINEDEVCAPS GetDeviceLine(DWOR  * dwAPIVersion, LPLINEDEVCAPS lpLineDevCaps);
    LPLINEDEVCAPS  MylineGetDevCaps(LPLINEDEVCAPS lpLineDevCaps, DWORD dwDeviceID, DWORD dwAPIVersion);
    LPVOID CheckAndReAllocBuffer(LPVOID lpBuffer, size_t sizeBufferMinimum);
  LPLINEADDRESSCAPS MylineGetAddressCaps ( LPLINEADDRESSCAPS lpLineAddressCaps, DWORD dwDeviceID, DWORD dwAddressID, DWORD dwAPIVersion, DWORD dwExtVersion);
    BOOL  MakeTheCall(LPLINEDEVCAPS  lpLineDevCaps, LPCSTR lpszAddress);
LPLINECALLPARAMS  CreateCallParams(LPLINECALLPARAMS lpCallParams, LPCSTR lpszDisplayableAddress);
   long WaitForReply (long lRequestID);
void  HandleLineCallState(DWORD  dwDevice,  DWORD dwMessage, DWORD dwCallbackInstance, DWORD dwParam1, DWORD dwParam2, DWORD dwParam3);
```

限于篇幅只对个别函数说明。

```
BOOL CTapiConnect: : Create()  //初始化线路
{
   long lReturn;
// If we´re already initialized, then initialization succeeds.
if (m_hLineApp)
      return TRUE;
// If we´re in the middle of initializing, then fail, we´re not done.
   if (m_bInitializing)
      return FALSE;
   m_bInitializing = TRUE;
    // Initialize TAPI
   do
   {
lReturn =: : lineInitialize(& m_hLineApp,
              AfxGetInstanceHandle(),
              lineCallbackFunc,
              ″DialIt″,
              & m_dwNumDevs);
      if (m_dwNumDevs = = 0)
      {
    AfxMessageBox ( ″There are no telephony devices installed. ″);
         m_bInitializing = FALSE;
         return FALSE;
      }
if (HandleLineErr(lReturn)) //线路错误处理函数，线路错返回零，否则非零
         continue;
      else
      {
OutputDebugString(″lineInitialize unhandled error\n″);
 m_bInitializing = FALSE;
return FALSE;
      }
```

```
    }
    while(lReturn ! = SUCCESS);
    OutputDebugString(″Tapi initialized. \n″);
    m_bInitializing = FALSE;
    return TRUE;
}
BOOL CTapiConnect: : DialCall(char * szPhoneNumber)
//建立呼叫
{;
    long lReturn;
    LPLINEDEVCAPS lpLineDevCaps = NULL
    if (m_bTapiInUse)
    {
        AfxMessageBox(″A call is already being handled. ″);
        return FALSE;
    }
    // Make sure TAPI is initialized properly
    if (!m_hLineApp)
    {
        if (!Create())//NULL
            return FALSE;
    }
//If there are no line devices installed on the machine, quit.
    if (m_dwNumDevs < 1)
        return FALSE;
    // We now have a call active. Prevent future calls.
    m_bTapiInUse = TRUE;
    // Get a phone number from the user.
    if (szPhoneNumber = = (char * ) NULL)
    {
        if (m_szPhoneNumber = = (char * ) NULL)
        {
            HangupCall();
            goto DeleteBuffers;
        }
    }
    else
        strcpy(m_szPhoneNumber, szPhoneNumber);
    // Get the line to use
     lpLineDevCaps = GetDeviceLine( & m_dwAPIVersion,
lpLineDevCaps); //用于发现线路//信息
//Need to check the DevCaps to make sure this line is usable.
    if (lpLineDevCaps = = NULL)
    {
        OutputDebugString(″Error on Requested line\n″);
        goto DeleteBuffers;
    }
     if ( ! (lpLineDevCaps - >dwBearerModes & LINEBEAR-
ERMODE_VOICE ))
    {
        AfxMessageBox( ″The selected line doesn´t support
VOICE capabilities″);
        goto DeleteBuffers;
    }
    // Does this line have the capability to make calls?
     if ( ! (lpLineDevCaps - >dwLineFeatures & LINEFEA-
TURE_MAKECALL))
    {
        AfxMessageBox( ″The selected line doesn´t support
MAKECALL capabilities″);
        goto DeleteBuffers;
    }
    // Does this line have the capability for interactive voice?
     if( ! (lpLineDevCaps - >dwMediaModes & LINEMEDI-
AMODE_INTERACTIVEVOICE))
    {
        AfxMessageBox( ″The selected line doesn´t support
INTERACTIVE VOICE capabilities″);
        goto DeleteBuffers;
    }
    // Open the Line for an outgoing call.
    do
    {
     lReturn = : : lineOpen(m_hLineApp, m_dwDeviceID, &
m_hLine, m_dwAPIVersion, 0, 0, LINECALLPRIVILEGE_NONE,
0, 0);
     if (( lReturn = =LINEERR_ALLOCATED) || (lReturn = =
LINEERR_RESOURCEUNAVAIL))
        {
HangupCall();
OutputDebugString(″Line is already in use by a non - TAPI
application ″ ″or by another TAPI Service Provider. \n″);
goto DeleteBuffers;
        }
        if (HandleLineErr(lReturn))
continue;
        else
        {
 OutputDebugString(″Unable to Use Line\n″);
 HangupCall();
 goto DeleteBuffers;
        }
    }
    while(lReturn ! = SUCCESS);
    // Start dialing the number
    if( MakeTheCall(lpLineDevCaps, m_szPhoneNumber))
        OutputDebugString(″lineMakeCall succeeded. \n″);
    else
    {
OutputDebugString(″lineMakeCall failed. \n″);
        HangupCall();
    }
DeleteBuffers:
    if (lpLineDevCaps)
        LocalFree(lpLineDevCaps);
    return m_bTapiInUse;
} /
BOOL CTapiConnect: : HangupCall() / 挂起呼叫进程
```

```
{
    LPLINECALLSTATUS pLineCallStatus = NULL;
    long lReturn;
    // Prevent HangupCall re-entrancy problems.
    if (m_bStoppingCall)
        return TRUE;
//If Tapi is not being used right now, then the call is hung up.
    if (!m_bTapiInUse)
        return TRUE;
    m_bStoppingCall = TRUE;
    OutputDebugString("Stopping Call in progress\n");
    // If there is a call in progress, drop and deallocate it.
    if (m_hCall)
    {
        pLineCallStatus = (LPLINECALLSTATUS)malloc(sizeof(LINECALLSTATUS));
        if (!pLineCallStatus)
        {
            ShutdownTAPI();
            m_bStoppingCall = FALSE;
            return FALSE;
        }
    lReturn = ::lineGetCallStatus(m_hCall, pLineCallStatus);
      // Only drop the call when the line is not IDLE.
        if (!((pLineCallStatus->dwCallState) & LINECALLSTATE_IDLE))
        {
            WaitForReply(lineDrop(m_hCall, NULL, 0));
            OutputDebugString("Call Dropped. \n");
        }!
        // The call is now idle.  Deallocate it
        do
        {
            lReturn = ::lineDeallocateCall(m_hCall);
            if (HandleLineErr(lReturn))
                continue;
            else
            {
OutputDebugString("lineDeallocateCall unhandled error\n");
                break;
            }
        }
        while(lReturn != SUCCESS);
        OutputDebugString("Call Deallocated. \n");
    }
    // if we have a line open, close it.
    if (m_hLine)
    {
      lReturn = ::lineClose(m_hLine);
      if (!HandleLineErr(lReturn))
         OutputDebugString("lineClose unhandled error\n");
       OutputDebugString("Line Closed. \n");
    }
    // Clean up.
    m_hCall = NULL;
    m_hLine = NULL;
    m_bTapiInUse = FALSE;
     m_bStoppingCall = FALSE; // allow HangupCall to be called again.
   // Need to free buffer returned from lineGetCallStatus
    if (pLineCallStatus)
        free(pLineCallStatus);
    OutputDebugString("Call stopped\n");
    return TRUE;
})
BOOL CTapiConnect::ShutdownTAPI(
//关闭 TAPI 的所有应用
{
    long lReturn;
// If we aren't initialized, then Shutdown is unnecessary.
    if (m_hLineApp == NULL)
        return TRUE;
    // Prevent ShutdownTAPI re-entrancy problems.
    if (m_bShuttingDown)
        return TRUE;
    m_bShuttingDown = TRUE;
    HangupCall();
    do
    {
        lReturn = ::lineShutdown(m_hLineApp);
        if (HandleLineErr(lReturn))
            continue;
        else
        {
OutputDebugString("lineShutdown unhandled error\n");
            break;
        }
    }
    while(lReturn != SUCCESS);
    m_bTapiInUse = FALSE;
    m_hLineApp = NULL;
    m_hCall = NULL;
    m_hLine = NULL;
    m_bShuttingDown = FALSE;
    OutputDebugString("TAPI uninitialized. \n");
    return TRUE;
}
```

TAPI 的使用相当方便，初学者会有一些困难。但它比起 AT 指令来有很大的优势，所以程序员学习使用它很有必要，还有 TAPI 的使用需要连接 tapi32.lib 否则程序不会通过。

以上只是源程序的一部分，但简单地介绍了 TAPI 的使用，如果大家有兴趣可以发 E_mail 和我联系索要源程序共同讨论。（lp_junior@263.net）

收稿日期：2000 年 12 月 15 日）

自己编程测试网络速度

陈绍溴

摘　要　利用 C + + Builder 的 TNMEcho 控件测试网络速度，充分利用上网时间，节省上网费用。
关键词　网络速度，C + + Builder 5. 0，TNMEcho 控件

为了节省上网的时间和费用，我们都希望在网络不太拥挤的时候上网，怎样判断网络是否拥挤呢？测试网络运行的速度可以达到此目的。因为网络在拥挤时段和在空闲时段的连接速度不同。

你是否也想自己编写一个测试网络速度的应用程序呢？使用 C + + Builder 可以圆你的梦，因为 C + + Builder 有强大的网络功能，使用户可以不必关心复杂通讯协议的具体实现方法，就能编写 Internet 网络程序。使用 C + + Builder5 的 FaarNet 选项卡中的 TNMEcho 控件可以实现测试网络速度的目的。该控件可用于发送文本到 internet 的远程服务器，远程服务器在收到文本后，将源地址与目的地址互换后，又将该文本返回用户，并显示文本从发送到返回共花费的时间，利用这个特性，我们就能达到测试网络的速度的目的。程序编写步骤如下：

运行 C + + Builder 打开一个新工程，将窗体的标题 Caption 命名为‘网络速度测试”，在窗体中加入面板 Panel11，初始状态条 StatusBar1，五个 Label 标签，四个 Button 按钮，四个 Edit 编辑框，和一个 NMEcho 控件，如下图所示：

各部件的属性设置为：

Label1 Caption = ‘服务器 IP 地址”；Label2 Caption = ‘端口号”；Label3 Caption = ‘发送文本内容”；Label4 Caption = “返回文本内容”；Label5 Caption = “已花费时间”；

Button1 Caption = ‘连接”；Button2 Caption = ‘断开连接”；Button3 Caption = “发送”；Button4 Caption = ‘中断”。

以上标签和按钮的内容在设计时填写。

编辑框的内容既可以在设计时填写好，也可以在程序运行后随意填写。编辑框 Edit1 用于填入选定的远程服务器的 IP 地址，比如填入 “202. 101. 231. 112”即选定江西宜春地区邮件服务器，编辑框 Edit2 填入使用的端口号，例如 “7”是 TCP / IP 协议中 TNMEcho 控件的默认端口号；编辑框 Edit3 填入要发送到远端服务器的报文内容，例如 ‘测试网络连接速度”。编辑框 dit4 不填写任何内容，用于接收从服务器返回的文本内容。

StatusBar1 用于显示应用程序的状态，‘连接按钮”可使计算机与 Internet 连接，‘断开按钮”可使计算机断开和远程服务器的连接，‘发送按钮”使计算机通过 NMEcho 控件将文本内容发送到远程服务器，然后又将文本返回计算机，并显示花费的时间，实现测试网络运行速度的目的。利用本程序测试网络运行速度的检测结果如下图所示。本人在不同时段测试网络速度，发现时间可相差好几倍。

整个程序代码和注释如下：

```
//- - - - - - - - - - - - - - - - - - -
#include  <vcl. h>
#pragma hdrstop
 #include ″Unit1. h″
//- - - - - - - - - - - - - - - - - - - -
#pragma package(smart_init)
#pragma resource ″* . dfm″
TForm1  * Form1;
//- - - - - - - - - - - - - - - - - - - -
_fastcall TForm1: : TForm1(TComponent * Owner) : TForm
(Owner)
{
}
//- - - - - - - - - - - - - - - - - - - -
void _fastcall TForm1: : Button1Click(TObject * Sender)
// 用户要求连接 Internet
{
```

```
 NMEcho1 -->ReportLevel = Status_Basic;
 NMEcho1 -->TimeOut = 20000;      //超时限制
 NMEcho1 -->Host = Edit1 -->Text;
 NMEcho1 -->Port = StrToInt(Edit2 -->Text);
 NMEcho1 -->Connect();
}
//---------------------
void __fastcall TForm1:: NMEcho1Connect(TObject * Sender)
 //与远端服务器连接成功将触发控件的 OnConnect( )方法显示 '连接成功! "
{
 StatusBar1 -->SimpleText = "已连接成功!";
}
//---------------------
void __fastcall TForm1:: NMEcho1ConnectionFailed(TObject * Sender)
// 与远端服务器连接失败将触发控件的 OnConnectionFailed( )方法显示 '连接失败! "
{
 ShowMessage("连接失败!");
}
//---------------------
//当调用 Connect( ) 方法后, 如果用户输入的是域地址而不是 IP 地址, 且域名服务器成功地解析了这个域名, 将触发控件的 OnHostResoved 事件, 将解析成功的消息在状态栏中显示给用户
void __fastcall TForm1:: NMEcho1HostResolved(TComponent * Sender)
{
 StatusBar1 -->SimpleText = "Host resolved";
}
//---------------------
//如果用户输入的远程主机地址不正确, 将触发控件的 OnInvalidHost 事件, 在此事件的处理中, 弹出对话框要求用户重新输入远程主机的 IP 地址或域名地址
void __fastcall TForm1:: NMEcho1InvalidHost(bool & Handled)
{
 AnsiString NewHost;
 if (InputQuery("主机 IP 地址无效", "请选择新的主机 IP 地址", NewHost))
 {
  NMEcho1 -->Host = NewHost;
  Handled = true;
 }
}
//---------------------
void __fastcall TForm1:: Button2Click(TObject Sender)
 //用户切断与远端服务器的连接
{
 NMEcho1 -->Disconnect();
}
//---------------------
void __fastcall TForm1:: Button3Click(TObject * Sender)
//发送文本到远程服务器
{
 Edit4 -->Text = NMEcho1 -->Echo(Edit3 -->Text);
//在编辑框 Edit4 显示返回的文本
  Label5 -->Caption = "已花费时间: "+FloatToStr(NMEcho1 -->ElapsedTime)+" milliseconds";
//显示从发送到返回花费的时间
}
//---------------------
//未联接 Internet 就点击发送按钮向与远端服务器发送文本时, 触发控件的 OnConnetionReqired( )方法, 出现首先必须连接 Internet 提示对话框
void __fastcall TForm1:: NMEcho1ConnectionRequired(bool & Handled)
{
 AnsiString BoxCaption;
 AnsiString BoxMsg;
 BoxCaption = "连接提示";
 BoxMsg = "要发送文本首必须连接 Intenet, 是否现在连接?";
  if (MessageBox(0, & BoxMsg[1], & BoxCaption[1], MB_YESNO + MB_ICONEXCLAMATION) = = IDYES)
 {
  Handled = true;
  Button1Click(this );
}
}
---------------------
void __fastcall TForm1:: Button4Click(TObject * Sender)
//用户中断向远端服务器发送文本
{
NMEcho1 -->Abort();
}
//---------------------
void __fastcall TForm1:: NMEcho1Disconnect(TObject * Sender)
 //用户切断与远程服务器的连接时触发 OnDisconnect{ }方法, 在初始状态栏显示 '连接已断开 "
{
 if (StatusBar1 ! = 0)
  StatusBar1 -->SimpleText = "连接已断开!";
}
//---------------------
void __fastcall TForm1:: NMEcho1Status(TComponent * Sender, AnsiString Status)
{
 if (StatusBar1 ! = 0)
   StatusBar1 -->SimpleText = Status;
}
//---------------------
```

收稿日期 2000 年 12 月 20 日

IP 炸弹来电显示

冯徐波

随着 INTERNET 的日益发展，上网的人越来越多。据统计我国的网民人数在 2000 万左右，而且人数呈现加速增长的势头。上网已经成为人们日常用语中一个较热的词语。

上网上多了，难免会发表一些与他人相左的意见，经常可以看到一些网友为了自己的观点争得“面红耳赤”，更有甚者会动用一些网络“武器”来攻击对方，让对方来个哑口无言 死机、蓝屏等）。这些网络“武器”通常是网上下载的一些所谓的 IP 炸弹，使用者只要知道对方的 IP 地址，就能使对方的计算机产生一些诸如死机，蓝屏等严重故障。IP 炸弹的原理大致上是利用 Windows 系统本身的漏洞来实现的。此漏洞主要表现为：①在 Windows95 下，当 135、137、139 端口有数据传送的话，即会导致系统死机 ②在 Windows98 下存在所谓的：IGMP 漏洞，当有一些残缺不全的 IP 包发送到计算机的 100 端口，即会发生死机、蓝屏 ←种系统严重故障的表现）、自动重新启动等现象。

要想免受此类 IP 炸弹的攻击，最好的方法是给 Windows 系统打补丁。打补丁的方法很简单，只需到 Microsoft 官方网站下载补丁程序，然后安装即可。

那么，是否我们只能一味的消极防御，只能眼睁睁的看着那些所谓的“黑客”耀武扬威呢？回答是否定的，下面是笔者利用 VB 自己编程实现的一段程序，该程序可以显示 IP 炸弹攻击者的 IP 地址，让攻击者无处可遁。当然前提是你的计算机已打好补丁。

该程序主要可以应付攻击 135、137、139 等端口的 IP 炸弹。过程如下：

一、 首先启动 VB6.0，选择新建“标准 EXE”。然后加入 WINSOCK 控件。

二、 在窗体上放置 WINSOCK、TIMER、RICHTEXTBOX 控件各一，且分别取名为 ProtectSock、Timer1、MsgText。

三、 然后加入如下代码：

```
Public ConnectCount As Long
Private Sub Form_Load()
On Error Resume Next
'连线计数=0
ConnectCount = 0
'设定传输协定为 TCP/IP,而监听端口为 139.
ProtectSock(0).Protocol = sckTCPProtocol
ProtectSock(0).LocalPort = 139  '若要设为其他端口则可以在此更改
ProtectSock(0).Listen          'WINSOCK 进入监听状态
'若开启监听失败,则开启 Timer 控件,一直不断的开启监听,直到取得控制权为止,否则关闭.
If ProtectSock(0).State <> 2 Then
 Timer1.Enabled = True
Else
 Timer1.Enabled = False
End If
'一般使用 WinSock 会遇到一个问题,就是只能与一条 IP 做连线动
'作.假如 IP 断线,WinSock 也会跟着中断.所以我们必需开启 WinSock
'阵列,以产生很多 Shell 可以跟很多 IP 进行连线.
End Sub
Private Sub Form_Resize()
'将 RichTextBox 设计成可以随视窗大小做 Resize
On Error Resume Next
'若视窗太小,则将文字框隐藏,以免发生错误
If (Me.Height - 370) < 0 Or (Me.Width - 370) < 0 Then
 MsgText.Visible = False
Else
 '否则将文字框显示出来,并且将其宽高 Resize
 MsgText.Visible = True
 MsgText.Height = Me.Height - 370
 MsgText.Width = Me.Width - 100
End If
End Sub
Private Sub ProtectSock_Close(Index As Integer)
'如果对方断线,则会引发这个事件.
ProtectSock(Index).Close
'对于 PortSock(0)则继续做监听的动作
ProtectSock(0).Close
ProtectSock(0).Listen
End Sub
Private Sub ProtectSock_ConnectionRequest(Index As Integer, ByVal requestID As Long)
'当有电脑企图对机器做 Port 139 的连线时,即会启动此事件.这时我们可以利用此事件来截取攻击者的 IP 地址.
'将连线计数器+1
ConnectCount = ConnectCount + 1
'载入一个 WinSock,供这条 IP 连线
Load ProtectSock(ConnectCount)
'开启连线
ProtectSock(ConnectCount).Accept requestID
'对于 PortSock(0)则继续做监听的动作
ProtectSock(0).Close
ProtectSock(0).Listen
'在文字框上显示遭到攻击的讯息
MsgText.SelStart = Len(MsgText.Text)
MsgText.SelText = "遭到来自 [" + ProtectSock(0)
```

用 VC 开发 Intranet 数据同步程序

汪 洋

摘 要 本文介绍了一种在 Internet 上实现数据同步的方法 基本介绍了 FTP 协议 还给出了一些关键部分的源代码。

关键字 VC Windows Sockets Intranet 数据同步

一、 引言

在 Internet 技术飞速发展的今天，许多企业开始采用 Internet 组网和应用技术构建企业内网 Intranet。数据库的共享对于一个企业具有重要的意义，不论是从数据共享后所省下的大量成本来看，还是从数据共享的时效性来说都有明显的好处，而这对在不同地区有着分部门的企业更是十分重要的事。对于大型的企业，可以通过从 ISP 商购买相应的服务来解决，如租用专线、建立 VPN 专用网。而这对于中小型企业很可能价格过高，难以接受，况且 ISP 商并不能给所有地区都提供完善的网络服务，很多地区只能通过并不稳定的电话线上网。本文就是要介绍一种只要使用一个程序、一条电话线就可以稳定地实现数据同步的方法。实际上，本文不只局限于 Intranet 的数据同步，对于任何需要在 Internet 上保持数据同步的情况，例如网站镜像，也具有参考意义。

二、 基本原理

本项目由两个程序组成，Mirror 和 MirrorGuide。MirrorGuide 运行于中央服务器 通常位于公司总部），用来产生每天更新的文件列表，并由 FTP 服务器端程序发布这些文件。Mirror 程序运行于子服务器 通常位于公司分部），用来根据 MirorGuide 产生的文件列表进行下载，保证数据同步。Mirror 程序的流程如图 1。

文件传输采用 FTP 协议，因为 ① FTP 协议支持多帐号、密码，可以限制不同使用者使用资源；② FTP 协议支持断点续传，这对于在线路不稳定或文件体积庞大时能顺利完成任务是至关重要的。网络通讯程序利用 Windows Sockets 简称 WinSock 来实现。虽然微软的 MFC 类库提供了支持 FTP 的类（CFtpConnection），但是在传输的速度、编程的灵活性及差错控制方面，WinSock 显然是要优于 MFC 封装后的类库。

三、 程序编制

1. MirrorGuide

本程序运行后将产生指定目录下在某段时间内更新过的文件列表，把产生的列表存成一个以当天日期命名的文件。然后启动定时程序，如 Windows 自带的 Mstask 设置每天运行 MirrorGuide 程序一次。程序编写中主要难点是利用队列和广度优先算法遍历指定目录，发现某文件的更新时间在所需范围内时便将其加入列表。

完成后的可执行程序，最好放在提供 FTP 服务的根目录。这样 MirrorGuide 只需产生当前目录下的更新文件列表，就是全部可下载文件的列表。子服务器可以先下载该列表，再作挑选。

程序编写中需要注意：在把文件名存入列表时，还需存入

```
. RemoteHostIP + "] 的电脑攻击." + Chr(10)
End Sub
Private Sub ProtectSock_DataArrival(Index As Integer, ByVal bytesTotal As Long)
'若对方有传送讯息,则会引发此事件,这时,我们可以利用此事件,截取对方传来的讯息,r
Dim GetStr As String
ProtectSock(Index). GetData GetSt
'在文字框上显示对方传来的讯息
MsgText. SelStart = Len(MsgText. Text)
MsgText. SelText = "[" + ProtectSock(0) . RemoteHostIP + "]" + GetStr + Chr(10)
End Sub
Private Sub Timer1_Timer()
'若仍无法取得监听权,则一直不断的开启,否则,若开启成功,则关闭 Timer
On Error Resume Next
If ProtectSock(0). State <> 2 Then
 ProtectSock(0). Protocol = sckTCPProtocol
 ProtectSock(0). LocalPort = 100
 ProtectSock(0). Listen
Else
 Timer1. Enabled = False
End If
End Sub
```

本程序在 VB6. 0 下调试通过。如果您有任何建议，请到笔者的网页参与讨论 （http / / msvb. go. 163. com）。

收稿日期：2000 年 12 月 25 日）

文件长度。这样有助于子服务器一次下载该文件失败后的多次下载，具体原因见后文。

2. Mirror

本程序是一个根据 FTP 协议传输文件的客户端程序 需要与 FTP 服务器端的 MirrorGuide 程序配合运行。它首先下载 MirrorGuide 产生的更新文件列表，然后按需要下载列表中的文件。程序最重要的是它的续传功能，即当线路不好时，能不断地续传以完成工作。

2. 1 FTP 编程

2. 1. 1 FTP 常用命令

FTP 协议 （RFC765）可以从网上查到。常用 FTP 命令如下表：

命令名	命令代表的意义	用 法
USER	用户名	USER + 用户名
PASS	密码	PASS + 密码
TYPE	数据表示类型	TYPE + A
CWD	改变工作目录	CWD + 目录名
PORT	监听端口的网址	PORT + IP 地址、端口号
REST	设置文件传输的起点	REST + 偏移大小
STOR	上传	STOR + 文件名
RETR	下载	RETR + 文件名
QUIT	退出	QUIT

上表中，TYPE 命令对应于不同的系统有多种选择，在 Windows 系统中用 TYPE A 就可以了；PORT 的具体用法为 PORT h1 h2 h3 h4 p1 p2，其中 h1 到 h4 为主机地址，p1、p2 是端口地址 h1 和 p1 都是地址的高 8 位 按主机字节顺序 。另外 FTP 的每一个命令都要以回车 （“lrln” 结尾。

2. 1. 2 FTP 命令的应答方式

FTP 命令的应答信号由 3 个数字和一串解释的文字组成 每个数字都有不同的含义 但通常我们只关心开头的第一个数字 它的含义如下

1xx ：肯定预备应答，命令已经初始化，还需要进一步执行，请等待下一次应答。

2xx ：肯定完成应答，命令已经成功地完成。

3xx ：肯定中间应答，命令接受，用户需发出更详细的命令参数。

4xx ：否定暂时应答，命令暂时无法完成，请重试。

5xx ：否定永久应答，命令无法完成，命令的内容是错误的。

2. 1. 3 FTP 传输文件的基本流程

在 FTP 协议中，下载文件的基本流程如下：

① 连接到 FTP 服务器的 FTP 端口上。

② 发送用户名、密码。 USER PASS

③ 发送数据表示类型。 TYPE A

④ 在客户机上建立监听端口，发送监听端口网址。 PORT

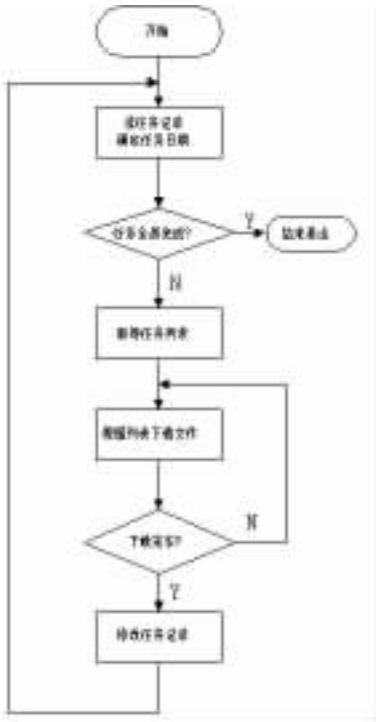


图 1

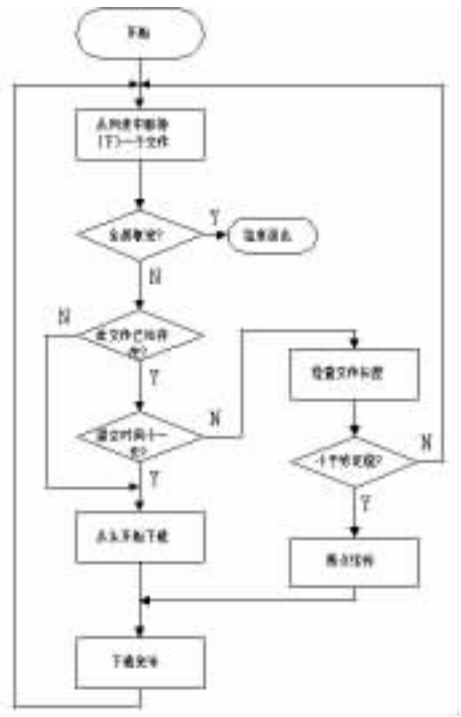


图 2

⑤ 发送偏移大小。 （REST ）

⑥ 发送下载命令。 RETR

⑦ 在监听端口上接受连接，产生数据套接字。

⑧ 在数据套接字上进行文件下载。

⑨ 退出。

FTP 上传文件的流程基本同上，只是没有 “REST” 命

令，再把“RETR”改成“STOR”就可以了。调用 WinSock 的 connect、send、recv、accept 等函数按上述流程编程就可以实现 FTP 传输文件了。

2.2 如何实现“不断地续传”

上文说过，MirrorGuide 程序每天产生一个以当天日期命名的文件，Mirror 程序应该比较容易地做到每天多次运行，直到完成当天任务为止，并且如果完不成时，第二天能够继续上一天的任务。问题是当下载一个文件，而本地已存在时如何操作？是重新下载、断点续传还是跳过它？

作者采用的方法是初次下载文件时修改文件的建立时间，即把文件的建立时间改成与当前任务是同一天。当发现本地已存在该文件时，检查它的建立时间，如果小于当前任务的时间（以天为单位），则重新下载；否则再检查它的长度，如果小于 MirrorGuide 给出的长度则断点续传（等于则跳过。其流程如图 2。

2.3 一些细节

① 程序中调用 WinSock 的函数最好单独放进一个线程。

这样，当套接字进行阻塞操作时，用户界面（UI）能保持激活状态，不至于一运行起来就象“死”了一样。VC 提供两种实现多线程的方式：工作者线程和用户界线程。本程序中，作者使用了用户界面线程，因为在这种方式下，一个线程对应于一个有消息循环的类，有助于我们以面向对象的方式进行编程。

② 由于通讯线路不很稳定，WinSock 的一些阻塞操作有可能会无限地阻塞下去，为了防止这种情况，可以“重载”send 和 recv 函数，代码如下：

```
int CMySocket::Send(const char* sToSend, const int nSize, const int nSecs)
{
    FD_SET fd = {1, m_hSocket}; // m_hSocket 是要进行 send 操作的套接字
TIMEVAL tv = {nSecs, 0}; // tv 是 nSecs 秒
if(select(0, NULL, &fd, NULL, &tv) == 0) // 套接字在 tv 内准备好?
  {
throw new CMySocketException("Send timeout"); // 超时,退出
    }
    int nBytesSent;
if ((nBytesSent = send(m_hSocket, sToSend, nSize, 0)) == SOCKET_ERROR)
  {
throw new CMySocketException("Send"); // 发送错误,退出
    }
    return nBytesSent;
}
```

重载 recv 函数的方法基本同上。

③ 在下载大量文件的过程中，当某一个文件传输错误时，我们希望能跳过这个文件，继续下去，这时便需要关闭套接字资源，重新开始。在编程中发现，closesocket 函数可能不立即释放套接字资源，这将导致以后的文件下载无法进行。原来强行关闭套接字需要增加以下代码：

```
LINGER ling1; // 一种套接字选项
ling1.l_onoff = 1;
ling1.l_linger = 0;
setsockopt(testsocket, SOL_SOCKET, SO_LINGER, &ling1, sizeof(LINGER));
// 更改 testsocket 的 SO_LINGER 选项
closesocket(testsocket); // 强行关闭
```

四、结束语

本文中介绍了一种在 Internet 上实现数据同步的方法，功能还比较单一。有需要的话还可以增加回报功能，把每天的工作结果以 Email 的方式通知管理员，这在 VC 中也比较容易实现，增加一个 SMTP 类即可，也可以在程序中增加与数据库连接的部分，实现数据库的同步。本程序已投入使用，每次运行可以正确传送千个以上的文件。希望本文有助于您建立一种稳定、自动、功能全的数据同步系统。

参考文献

1. RFC765

2. 蒋东兴编著. Windows Sockets 网络程序设计大全. 清华大学出版社，1999

3. David J. Kruglinski 著 希望图书创作室译. Visual C++ 6.0 技术内幕. 北京希望电子出版社，1999

（收稿日期：2001 年 1 月 10 日）

计算机病毒大变异 瑞星剿灭 Winux

3 月 28 日，瑞星公司向业界宣布：瑞星公司在国内率先捕获并剿灭能同时在 Windows 和 Linux 操作系统上传播、发作的病毒——Win32.Winux，此病毒是第一例可跨越两大平台发作的病毒。凡瑞星杀毒软件正版用户在升级自己的软件后均可完全防御该病毒！

Winux 病毒是世界上首例具有双重感染能力的病毒，该病毒用汇编语言编写，主要感染 Windows PE 和 Linux ELF 文件，它在 Windows 95/98/Me/2000/NT 操作系统中都具有传播能力，不仅如此，它还能够感染使用各种版本 Linux 操作系统的计算机，在用户双击感染了病毒的应用程序或者电子邮件附件时，Winux 病毒就会被激活，该病毒会自动寻找 100K 以上的应用程序并将之感染，从而使病毒迅速扩散。

近日，自称是 29A 病毒编写组织成员的 Benny 声称自己是 Winux 病毒制造者。瑞星公司提醒广大用户：买正版软件后勿忘及时注册、升级！

VC 应用程序中多背景位图动画的实现

姚 晔 胡益雄

摘 要 本文详细介绍了 VC 应用程序中实现多背景位图动画的两种方法及原理，并分析了它们的优缺点。

关键词 VC,多背景位图动画,文本位图,定时器

一、 动画基础简述

在一个应用程序中，动画的实现无疑给应用程序界面增添了一份生机和趣味。动画可被定义为变化的图形图象。随时间的变化显示不同的图形图象是生成动画的基本方法，因此，时间是动画的重要因素。在 Windows 中设置时间码，并在响应 WM_TIMER 消息时显示不同的图形图象就能产生动画效果。根据形成图形图象的方法不同 ，可简单地将动画划分为图形动画、调色板动画和位图动画。图形动画即随时间的变化使用 GDI 函数绘制不同的图形，以产生变化的图形效果 。调色板动画是先显示的特定位图随时间变化不断修改位图的调色板或每个象素的颜色值而形成动态视觉效果。位图动画是预先定义好的不同的位图序列，在不同的时间顺序显示它们而得到动画效果。随时间变化而显示预先定义好的图标也可形成动画，其方法与位图动画类似，可将其归并到位图动画。

在 VC 应用程序中，由于位图动画实现相对要简单一些，而且已有许多象 Photoshop 等一些图形处理软件的出现，将使位图动画更为生动。所以，位图动画是 VC 应用软件开发者在实现动画的过程中喜欢使用的方法。

下面，笔者根据本人在利用 VC 位图动画设计 ‘吸附式空调器测试软件系统 ” 主界面时的一些经验来谈谈 VC 应用程序中多背景位图动画实现的具体过程与步骤。为了方便起见，将应用工程命名为 Animate。

二、 多背景位图动画实现前准备：

2. 1 定制位图

该软件系统主界面背景图是引用 PhotoShop 中 Sample 目录下的 Bigsky. tif 图形，可利用 ‘另储存副本 ” 菜单选项将其转换成位图格式，取其名为 BackgroundCloudy. Bmp。位图大小是 800x600。ID 值为 IDB_BackgroundCloudy。利用 Visual Studio 中的位图编辑器制作需要在主界面背景图上移动的文本位图，文本位图的内容是 ‘欢迎使用吸附式空调器测试软件 ”，为了使文本位图更好看些，可利用 Photoshop 软件对其进行修饰。使用画笔将文本之外的部分涂成黑色，将之保存为 Text1. Bmp；ID 值为 IDB_Text1。再一次对该文本位图进行修改，将文本部分涂成黑色，文本之外部分变成白色，另存为 Text2. Bmp；ID 值为 IDB_Text2。文本位图的宽与高分别设为 rc. width.、rc. height。

2. 2 背景位图的显示

利用 AppWizard 生成虚拟函数 OnDraw，在 Animate. Cpp 中的 OnDraw 函数中实现背景位图 BackgroundCloudy. Bmp 在视图窗口中的显示。代码如下

```
CAnimateView: : OnDraw(CDC * pDC)
{CBitmap * BackgroundBmp;
BackgroundBmp = new CBitmap;
BackgroundBmp - >LoadBitmap(IDB_BackgroundCloudy);
CDC * pmendc = new CDC;
pmendc - >CreateCompatibleDC(pDC);
pmendc - >SelectObject(BackgroundBmp);
pDC - >BitBlt(0, 0, 800, 600, pmendc, 0, 0, SRCCOPY);
delete BackgroundBmp;
delete pmendc;
}
```

CDC LoadBitmap 函数负责将一个基于资源的 DIB 转换 GID 位图。由于不能直接从屏幕 DC 上选位图，所以首先要创建一个称之为 pmendc 的内存 DC，将位图选入内存中，再拷到屏幕 DC 中。CDC BitBlt 函数用于从内存设备上下文向显示设备上下文复制 BackgroundCloudy. Bmp 像素。在 VGA 显示器中，它是占据 800x600 逻辑单元矩形区域。

三、 多背景动画实现方法及原理

该软件系统主界面要求在 BackgroundCloudy 多背景图上实现文本 ‘欢迎使用吸附式空调器测试软件 ” 从左到右的移动。下面介绍两种实现方法。

3. 1 利用更新窗口某区域的方法实现

在 Animate. Cpp 中的 OnInitialUpdate 函数中加载文本位图并设置一个时间定时器，代码如下：

```
void CAnimateView: : OnInitialUpdate( )
{; ;
Text1bmp = new CBitmap;
Text2bmp = new CBitmap;
Text1bmp - >LoadBitmap(IDB_Text1)
Text2bmp - >LoadBitmap(IDB_Text2)
Startposx = 0;
Startposy = 150;
if( SetTimer( 1, 200, NULL) = = 0) AfxMessageBox( “Can not
```

```
install timer");
}
```

其中，Startposx、Startposy 是在 AnimateView. H 中定义的 public 整型变量，用来控制文本位图在屏幕中显示的位置。同时，在 AnimateView. H 的 public 中定义：CBitmap * Text1bmp，* Text2bmp

设置的定时器每隔 200 毫秒新产生 WM_TIMER 事件，现给 WM_TIMER 编代码如下：

```
void CAnimateView: : OnTimer( UINT nIDEvent)
{
Startposx = Startposx + 5;
CClientDC dc(this);
CDC * bmpText1 = new CDC;
CDC * bmpText2 = new CDC;
bmpText1 - >CreateCompatibleDC(& dc);
bmpText1 - >SelectObject(Text1bmp);
bmpText2 - >CreateCompatibleDC(& dc);
bmpText2 - >SelectObject(Text2bmp);
CRect rc(Startposx - 5, Startposy, Startposx - 5 + rc. width,
Startposy + rc. height );
InValidateRect(& rc);
UpdateWindow();
dc. BitBlt(Startposx, Startposy, rc. width, rc. height, bmp-
Text2, 0, 0, SRCAND);
dc. BitBlt(Startposx, Startposy, rc. width, rc. height, bmp-
Text1, 0, 0, SRCPAINT);
delete bmpText1, bmpText2;
}
```

要注意的是，别忘记在 Animate. Cpp 中的析构函数 ~ CAnimate ｛ 中加如下代码；

```
delete Text1bmp, Text2bmp ;
```

在文本位图移动到下一个位置之前，对文本位图上一个位置的区域 rc 利用 InValidateRect & rc 和 UpdateWindow 进行刷新。刷新的结果使该区域背景图重画，如此重复，实现文本位图的动画效果。让文本位图 Text2. Bmp 先与背景作与，由于 Text2. Bmp 文本部分全是黑色，与背景作与的结果是背景上文本部分为黑色，而 Text2. Bmp 文本之外的部分是白色，作用的结果是其周围的背景保留；再用 Text1. Bmp 与背景上刚刚被 Text2. Bmp 作用过的部分作异或，由于 Text1. Bmp 的文本部分是正常颜色，与黑色异或自然是正常颜色的文本，而 Text1. Bmp 文本周围均已涂黑，与背景异或不改变背景 。

3. 2. 利用保存背景再恢复背景的方法实现

该法需要在以上基础上作如下修改：

在 Animate. H 开始部分添加代码 HBITMAP SaveBackground (HWND hWnd)；

在 Animate. Cpp 中加入如下代码：

```
HBITMAP SaveBackground(HWND hWnd )
//保存需要保存的位图
{
CBitmap * m_background;
m_background = new CBitmap;
m_background - >LoadBitmap(IDB_BackgroundCloudy);
//装载背景位图
HDC hDC = GetDC(hWnd); //获取指定窗口的设备上下文
HBITMAP Hbackground = (HBITMAP) m_background - >Get-
SafeHandle( ); //获得背景位图的句柄
HDC hdcback = CreateCompatibleDC(hDC);
//为背景位图创建临时 DC
SelectObject(hdcback , Hbackground);
//将背景位图选入内存设上下文
HBITMAP Hbackgroundstore = CreateCompatibleBitmap(hDC,
800, rc. height );
//创建兼容位图, 用来保存某指定区域的背景位图
HDC hmendc = CreateCompatibleDC (hDC );
//创建兼容的内存设备上下文
HBITMAP Holdbitmap = (HBITMAP) SelectObject( hmendc,
Hbackgroundstore); //将兼容位图选入兼容内存设备上下文
BitBlt( hmendc, 0, 0, 800, rc. height, hdcback, 0, Startposy,
SRCCOPY);
//将指定区域背景位图拷贝到兼容的内存设备上下文
//Startposy 要重新定义, 或给定一个确定值, 在这里取 150
SelectObject( hmendc , Holdbitmap);
//恢复兼容的内存设备上下文
DeleteDC ( hmendc ); //删除创建的内存设备上下文
ReleaseDC( hWnd, hDC); //释放窗口设备上下文
delete m_background; //删除创建的位图对象
return Hbackgroundstore; //返回某区域背景位图句柄
}
```

现将 WM_TIMER 中的代码修改如下

```
void CAnimateView: : OnTimer( UINT nIDEvent)
{ HBITMAP m_hbackgroundcloudy;
Startposx = Startposx + 5;
m_hbackgroundcloudy = SaveBackground(m_hWnd);
CClientDC dc(this);
CDC * bmpText1 = new CDC;
CDC * bmpText2 = new CDC;
bmpText1 - >CreateCompatibleDC(& dc);
bmpText1 - >SelectObject( Text1bmp );
bmpText2 - >CreateCompatibleDC(& dc);
bmpText2 - >SelectObject( Text2bmp );
CDC pbackrecovery;
pbackrecovery. CreateCompatibleDC( & dc);
CBitmap * pold = pbackrecovery. SelectObject( CBitmap: :
FromHandle ( m_hbackgroundcloudy ));
dc. BitBlt(Startposx, Startposy, rc. width + Startposx,
rc. height, & pbackrecovery, 0, 0, SRCCOPY);
dc. BitBlt(Startposx, Startposy, rc. width, rc. height, bmp-
Text2, 0, 0, SRCAND);
dc. BitBlt(Startposx, Startposy, rc. width, rc. height, bmp-
Text1, 0, 0, SRCPAINT);
pbackrecovery. SelectObject ( pold );
DeleteObject( m_hbackgroundcloudy );
delete bmpText1, bmpText2;
}
```

该法基本操作流程可归纳如下：

a. 编辑动画的各子画面位图，将运动物体之外的区域变成白色，利用白色与背景色作与依然是背景色的原理屏蔽掉运

彩虹字的实现方法

曹 琦

程序介绍

窗体上放置的控件有：一个 Picture1 设置它的 AutoRedraw 为 Ture；一个 Timer1 设置它的 Interval 为 100；一个 Command 名称为 cmdSelectClipPath，设置它的 Visible 为 false。

在这个程序中我使用了几个 Api 函数。

```
Option Explicit
Private Declare Function TextOut Lib "gdi32" Alias "TextOutA" (ByVal hdc As Long, ByVal x As Long, ByVal y As Long, ByVal lpString As String, ByVal nCount As Long) As Long
Private Declare Function EndPath Lib "gdi32" (ByVal hdc As Long) As Long
Private Declare Function SelectClipPath Lib "gdi32" (ByVal hdc As Long, ByVal iMode As Long) As Long
Private Declare Function AbortPath Lib "gdi32" (ByVal hdc As Long) As Long
Private Declare Function BeginPath Lib "gdi32" (ByVal hdc As Long) As Long
Private Const RGN_COPY = 5
Private Sub DrawRainbow()
  Dim I As Integer
  For I = 0 To Picture1.ScaleHeight - 1 Step 4
    Picture1.Line (0, I)-(Picture1.ScaleWidth, I + 4), QBColor(Int(16 * Rnd)), BF
  Next I
End Sub
Private Sub cmdSelectClipPath_Click()
  Dim rc As Long
  Dim sString As String
  sString = "你好,«电脑报»软件世界!"
  rc = BeginPath(Picture1.hdc)
   rc = TextOut(Picture1.hdc, 0, 0, sString, 2 * Len(sString)) '因为这里用的是中文字符所以字符串长度*2
  rc = EndPath(Picture1.hdc)
  rc = SelectClipPath(Picture1.hdc, RGN_COPY)
  rc = AbortPath(Picture1.hdc)
  Call DrawRainbow
End Sub
Private Sub Form_Load()
  Randomize
  Me.ScaleMode = vbPixels
  Me.BackColor = vbWhite
  With Picture1
    .ScaleMode = vbPixels
    .ForeColor = vbRed
    .DrawWidth = 2
    .Font.Name = "宋体"
    .Font.Size = 36
    .Font.Bold = True
    .Font.Italic = True
  End With
End Sub
Private Sub Timer1_Timer()
cmdSelectClipPath = True
End Sub
```

此程序在 VB6.0 下测试成功。

收稿日期：2000 年 12 月 1 日）

动物体之外区域的白色；

b. 将定义好的位图装入内存并获得位图对象或句柄；

c. 设置适当的时间码；

d. 在时间处理函数中，先恢复上一字画面位图区域背景，再在新的位置显示新子画面位图，从而实现动画。

四、 两种方法比较和结论

方法 1 实现的代码简单，但是由于每隔一定时间 如 200 毫秒）就要对窗口某区域进行重画和更新，而在重画之前先要擦除文本位图字画面，所以将不可避免产生该区域的屏幕闪烁，而且重画区域越大，闪烁越厉害。方法 2 是利用背景位图恢复的办法实现位图动画，消除了方法 1 闪烁的缺陷，提高了动画的视觉效果，但是实现起来要复杂一些。

在运动文字画面大小相对小时，两种方法均可达到预期效果；而在运动文字画面大小相对大时，采用方法 2 将更好一些。

参考文献

1. Steve Rimmer 著，扬士强等译．高级多媒体程序设计．电子工业出版社，1995

2. Paul Perry 著，陈向群等译．多媒体开发指南．清华大学出版社，1995

3. Scott Stanfied 等著，华译工作室译．Visual C++4 开发人员指南．机械工业出版社，西蒙与舒斯特国际出版公司，1997

4. RobertD Thompson 著，前导工作室译．MFC 开发人员参考手册．机械工业出版社，1998

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

收稿日期：2000 年 11 月 28 日）

快速实现彩色图像的增强——削波

马 苗

一、引言

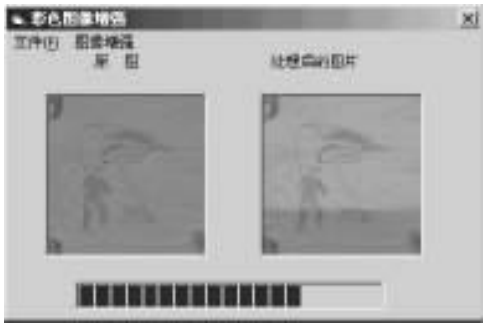
由于成像时光照不足或者光照过强 会使得整幅图像偏暗或偏亮。我们称这些情况为低对比度 即灰度都挤在一起 没有拉开。要想获得较好的视觉效果就必须进行灰度扩展即把感兴趣的灰度范围拉开，使得该范围内的像素，亮得越亮，暗的越暗，从而达到增强对比度，实现图像增强的目的。削波 是对比度扩展的一个特例 即只对图片的中间某一个范围内的灰度值进行扩展。在此我们对彩色图片进行削波处理，方法是把红、绿、蓝三种分量的值分别进行扩展，以获得更多的图像信息。

二、编程应用

以下是用 VB 6.0 编写的程序，分别在 Window98 和 Windows 2000 环境中实现。

1. 建立工程：启动 VB6.0|新建工程|标准 EXE；

2. 控件与菜单：这里我们一共用到 2 个 Microsoft Common Dialogue、1 个 Progress Bar 和 2 个 picture box、2 个 Label 和一个含有二级的菜单。选择视图 | 工具栏 | 标准，在 form1 上画出 picturebox1 picturebox2，lable1 和 label2；把鼠标移在工具箱上，单击左键，在弹出菜单中，选取 ‘部件”，在 ‘控件”上，选中 “Microsoft Common Dialog 6.0 和 “Microsoft Windows Common Controls 6.0 前面的复选框。在 “应用”时，你已经具备了增强图像的所有控件。下面我们来建立菜单：选择视图 | 工具栏 | 窗体编辑器，建立一级菜单 ‘文件”， ‘图像增强”；在 ‘文件”下，建立二级菜单 ‘打开文件”、 ‘保存文件”和 ‘退出系统”。在 ‘图像增强”下，建立二级菜单 ‘彩色削波”。



3. 调整控件后锁定，设置控件属性：

```
form1. Caption = ″彩色图像增强″
Picturebox2. ScaleMode = 3 ′Pixel;
Picturebox1. ScaleMode =3 ′Pixel
Lable1. Caption = ‘原 图”; lable2. caption = ″处理后的图像″
```

4. 程序代码：

```
Option Explicit
 Dim imagepixels(2, 1024, 1024) As Integer ′用来存储读入的图像数据
 Dim x, y As Integer ′用来记录图像的宽度和高度
 Dim picturename, picture_savename As String
I、打开文件
Private Sub open_Click()
Dim i As Integer, j As Integer
Dim red As Long, green As Long, blue As Long
Dim pixel As Long
 ′ 设置 ‘CancelError ’为 True
CommonDialog1. CancelError = True
On Error GoTo ErrHandler ′ 设置标志
CommonDialog1. Flags = cdlOFNHideReadOnly ′ 设置过滤器
CommonDialog1. filter = ″All Files (*.*)|*.*|Text Files″ & _
″(*.txt)|*.txt|pictures(*.gif)|*.gif|pictures(*.bmp)|*.bmp″ ′ 指定缺省的过滤器
CommonDialog1. FilterIndex = 4 ′ 显示 ‘打开 ’对话框
CommonDialog1. ShowOpen ′ 显示选定文件的名字
picturename = CommonDialog1. FileName
If picturename = ″″ Then Exit Sub
Picture1. Picture = LoadPicture(picturename)
Picture2. Picture = Picture1. Picture
Picture1. Refresh
Picture2. Refresh
Picture1. AutoSize = True
x = Picture1. ScaleWidth
y = Picture1. ScaleHeight
form1. Visible = False
For i = 0 To y - 1
For j = 0 To x - 1
pixel& = form1. Picture1. Point(j, i)
red = pixel& Mod 256
green = ((pixel& And &HFF00) / 256&) Mod 256&
blue = (pixel& And &HFF0000) / 65536
imagepixels(0, j, i) = red ′分别存储像素点的 GRB 值
imagepixels(1, j, i) = green
 imagepixels(2, j, i) = blue
 Next
   Next
form1. Visible = True
 form1. Show
ErrHandler:
    ′ 用户按了 ‘取消 ’按钮
Exit Sub
End Sub
II、保存文件
```

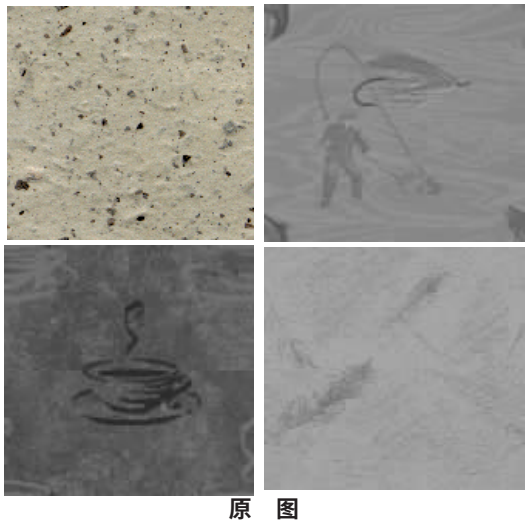
```
Private Sub save_Click()
CommonDialog2.CancelError = True ' 初始化 "CancelError" 为 True
On Error GoTo ErrHandler ' 设置标志
CommonDialog2.Flags = cdlOFNHideReadOnly ' 设置过滤器
CommonDialog2.filter = "All Files (*.*)|*.*|Text Files" & _
"(*.txt)|*.txt|pictures(*.gif)|*.gif|pictures(*.bmp)|*.bmp" ' 指定缺省的过滤器
CommonDialog2.FilterIndex = 4 ' 显示 '打开'对话框
CommonDialog2.ShowSave ' 显示选定文件的名字
picture_savename = CommonDialog2.FileName
SavePicture Picture1.Image, picture_savename
ErrHandler: ' 用户按了 '取消'按钮
Exit Sub
End Sub
```

III. **退出系统**

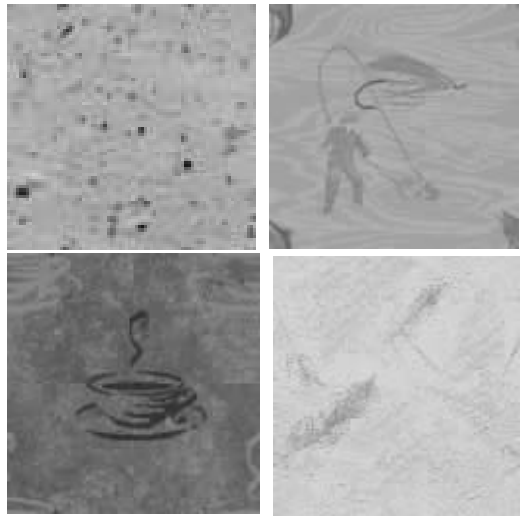
```
Private Sub exit_Click()
Unload Me
End Sub
```

IV. **实现彩色图像削波**

```
Private Sub mclipping1_Click() '对彩色图像进行削波
Dim i As Integer, j As Integer
Dim red As Long, green As Long, blue As Long
Dim gray
Dim Y_of_Img, U_of_Img, V_of_Img, redr, greeng, blueb
Dim f_rgb(10) As Single '该数组存放 RGB 向 YUV 转换时的常量系数
Dim yuv(10) As Single '该数组存放 YUV 向 RGB 转换时的常量系数. 1
f_rgb(1) = 1: f_rgb(2) = 1: f_rgb(3) =
f_rgb(4) = 0: f_rgb(5) = 0.395: f_rgb(6) = 2.032
f_rgb(7) = 1.14: f_rgb(8) = -0.581: f_rgb(9) = 0
yuv(1) = 0.299: yuv(2) = -0.148: yuv(3) = 0.615
yuv(4) = 0.587: yuv(5) = -0.289: yuv(6) = -0.515
yuv(7) = 0.114: yuv(8) = 0.437: yuv(9) = -0.1
If Picture1.Picture = 0 Then
MsgBox ("please choose an image, firstly")
Exit Sub
End If
ProgressBar1.Visible = True
For i = 0 To y - 1
    For j = 0 To x - 1
        red = imagepixels(0, j, i)
        green = imagepixels(1, j, i)
        blue = imagepixels(2, j, i) '求出亮度
gray = Int(red * yuv(1) + green * yuv(4) + blue * yuv(7))
If gray < 30 Then Picture1.PSet (j, i), RGB(0, 0, 0) '令小于某亮度的部分置为黑色
 If gray >= 30 And gray < 200 Then
Y_of_Img = Int(red * yuv(1) + green * yuv(4) + blue * yuv(7))
U_of_Img = Int(red * yuv(2) + green * yuv(5) + blue * yuv(8))
V_of_Img = Int(red * yuv(3) + green * yuv(6) + blue * yuv(9))
redr = Int((Y_of_Img * f_rgb(1) + U_of_Img * f_rgb(4) + V_of_Img * f_rgb(7)) / 200 * 256)
greeng = Int((Y_of_Img * f_rgb(2) + U_of_Img * f_rgb(5) + V_of_Img * f_rgb(8)) / 200 * 256)
blueb = Int((Y_of_Img * f_rgb(3) + U_of_Img * f_rgb(6) + V_of_Img * f_rgb(9)) / 200 * 256)
 If redr > 255 Then redr = 255: If redr < 0 Then redr = 0
If greeng > 255 Then greeng = 255: If greeng < 0 Then greeng = 0
If blueb > 255 Then blueb = 255: If blueb < 0 Then blueb = 0
Picture1.PSet (j, i), RGB(redr, greeng, blueb)
End If
 If gray >= 200 Then Picture1.PSet (j, i), RGB(255, 255, 255) '令大于某亮度的部分置为白色
Next
 Picture1.Refresh
ProgressBar1.Value = i * 100& / (y - 1)
DoEvents
   Next
MsgBox ("the colorful image is clipped!")
form1.ProgressBar1.Visible = False
End Sub
```



原 图



削波后的图像

彻底解决 Win95 - OSR2 版本的双重引导问题

王兴波

摘 要 本文介绍了一个解决 Win95 - OSR2 版本不能双重引导问题的方法。利用本文提供的方法和数据，用 C 语言编写一个小程序，就能解决 Win95 - OSR2 版本不能双重引导问题。而且 Win95 的功能不受任何影响。

关键词 操作系统，引导程序，磁盘，扇区

一、 引言

DOS 操作系统曾经是微机操作系统的主角。几乎所有的电脑操作者都有过使用 DOS 操作系统的经历。随着 Windows 系列新操作系统的问世，Windows9x 已经成为大多数微机用户不可缺少的操作系统。但是，对于一些特殊层的用户，尽管到了 WindowsME、Windows2000 时代，DOS 操作系统仍然是他们不可丢弃的。因此在 DOS 与 Windows 之间的双重引导成为这些用户关心的焦点。通过 Internet 以及一些文献资料可以看出，有关 DOS 与 Windows95 之间的双重引导，依旧是人们讨论的话题[1 - 4]。从现有介绍 DOS 与 Windows95 之间双重引导的文献资料来看，现行方法都是基于手工操作的，如文[1][2]的方法，操作烦琐，对于非计算机专业的用户无法实施，也不具备通用性。

由于早期的 Windows95 以及 Windows98 / NT 与 DOS 之间在双重引导问题上基本没有问题，本文主要解决 Win95 - OSR2 版本与 DOS 之间双重引的问题。通过对操作系统引导扇区的结构分析，笔者从理论上给出了解决 Win95 - OSR2 版本与 DOS 之间双重引问题的方法，并利用 C 语言编写一个小程序，彻底解决这个问题。经过近年来在数百台微机上的运行实践证明，本文的方法简单、实用、可靠。

二、 解决问题的方法

对于本文提出的方法是彻底修改 Win95 - OSR2 的引导扇区。为此需要了解一下引导扇区结构[5]。DOS / WindowsX 系统的引导扇区是一个 512 字节的扇区，其位置为磁盘第一个逻辑扇区 （编号 0）。引导扇区的结构如下：

起始位置	意 义
00h - 01h	JMP 指令 EB XX 两字节
02h	NOP 指令
03h - 0Ah	OEM 版本标识域 8 字节
0Bh - 23h	磁盘 IO 表 19 字节
24h	驱动器代码 0x80 为 C 盘
25h	保留字节
26h	磁盘署名字节
27h - 2Ah	磁盘卷系列号 4 字节
2Bh - 35h	磁盘卷标 11 字节
36h - 2FDh	引导记录代码
2Feh - 2FFh	引导区标记 55hAAh

其中，磁盘 IO 参数表记录着磁盘各项 IO 参数。可以看出，引导扇区里有许多内容是跟磁盘有关的。例如磁盘 IO 参数表、磁盘卷系列号、磁盘卷标等。对于不同的物理硬盘，这些参数自然不同。即使是相同的物理硬盘，这些参数也有不同的。例如，由于分区大小不一样、卷标不一样，这些参数都会不一样。有关磁盘 IO 参数表更详细的资料请参考[2]。

文[1]里已经说明， 对 Win95 标准版的引导代码稍加修改，就可以引导工作于 16 位 FAT 的 Win95 - OSR2 系统，而且笔者经过分析，发现这些代码是固定的。因此修改引导扇区的关键是保证与磁盘相关的各项参数准确。

（1） 在需要修改引导的机器上，获取 OSR2 的 Boot 区，从其中获取磁盘信息。

（2） 在任何一台机器上，获取 Win95 标准版的 Boot 区，将这个 Boot 区的数据作为一个数组。

（3） 用获取的磁盘信息替代数组里的相关内容。

（4） 修改数组里引导部分的相关内容。例如改 WINBOOT. SYS 为 JO. SYS 等。

三、结束语

当一幅自己喜欢的图片 由于成像时光照不足 或者光照过强使得整幅图像模糊不清时，你再也不用为此而沮丧了。因为只需要安装上 VB6. 0 参考上面的程序，你便可以轻轻松松 DIY，化腐朽为神奇了！

（收稿日期：2000 年 12 月 4 日）

5） 将数组写入需要修改引导的机器的引导扇区。

这样，就完成了引导扇区的修改工作。剩下的事情是在 Win95 的 MSDOS. SYS 里增加一行 BootMenu = 1 以便进行双重引导。

用 C 语言写成的修改引导的完整程序如下：

```
#include <stdio. h>
#include <dos. h>
unsigned char W95Boot[512] ={
0xeb, 0x3e, 0x90, ´W´, , ´9´ , ´5´ , ´B´ , ´O´ , ´O´ , ´T´ ,
0x20, 0x00, 0x02, 0x04, 0x01, 0x00,
0x02, 0x00, 0x02, 0x00, 0x00, 0xf8, 0xc9, 0x00, 0x3f, 0x00,
0x40, 0x00, 0x3f, 0x00, 0x00, 0x00,
0x01, 0x23, 0x03, 0x00, 0x80, 0x00, 0x29, 0x98, 0xa8,
0x53, 0x24, 0x20, 0x20, 0x20, 0x20, 0x20,
0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x46, 0x41, 0x54,
0x31, 0x36, 0x20, 0x20, 0x20, 0xf1, 0x7d,
0xfa, 0x33, 0xc9, 0x8e, 0xd1, 0xbc, 0xfc, 0x7b, 0x16, 0x07,
0xbd, 0x78, 0x00, 0xc5, 0x76, 0x00,
0x1e, 0x56, 0x16, 0x55, 0xbf, 0x22, 0x05, 0x89, 0x7e, 0x00,
0x89, 0x4e, 0x02, 0xb1, 0x0b, 0xfc,
0xf3, 0xa4, 0x06, 0x1f, 0xbd, 0x00, 0x7c, 0xc6, 0x45, 0xfe,
0x0f, 0x8b, 0x46, 0x18, 0x88, 0x45,
0xf9, 0xfb, 0x38, 0x66, 0x24, 0x7c, 0x04, 0xcd, 0x13, 0x72,
0x3c, 0x8a, 0x46, 0x10, 0x98, 0xf7,
0x66, 0x16, 0x03, 0x46, 0x1c, 0x13, 0x56, 0x1e, 0x03,
0x46, 0x0e, 0x13, 0xd1, 0x50, 0x52, 0x89,
0x46, 0xfc, 0x89, 0x56, 0xfe, 0xb8, 0x20, 0x00, 0x8b, 0x76,
0x11, 0xf7, 0xe6, 0x8b, 0x5e, 0x0b,
0x03, 0xc3, 0x48, 0xf7, 0xf3, 0x01, 0x46, 0xfc, 0x11, 0x4e,
0xfe, 0x5a, 0x58, 0xbb, 0x00, 0x07,
0x8b, 0xfb, 0xb1, 0x01, 0xe8, 0x94, 0x00, 0x72, 0x47, 0x38,
0x2d, 0x74, 0x19, 0xb1, 0x0b, 0x56,
0x8b, 0x76, 0x3e, 0xf3, 0xa6, 0x5e, 0x74, 0x4a, 0x4e, 0x74,
0x0b, 0x03, 0xf9, 0x83, 0xc7, 0x15,
0x3b, 0xfb, 0x72, 0xe5, 0xeb, 0xd7, 0x2b, 0xc9, 0xb8, 0xd8,
0x7d, 0x87, 0x46, 0x3e, 0x3c, 0xd8,
0x75, 0x99, 0xbe, 0x80, 0x7d, 0xac, 0x98, 0x03, 0xf0, 0xac,
0x84, 0xc0, 0x74, 0x17, 0x3c, 0xff,
0x74, 0x09, 0xb4, 0x0e, 0xbb, 0x07, 0x00, 0xcd, 0x10, 0xeb,
0xee, 0xbe, 0x83, 0x7d, 0xeb, 0xe5,
0xbe, 0x81, 0x7d, 0xeb, 0xe0, 0x33, 0xc0, 0xcd, 0x16, 0x5e,
0x1f, 0x8f, 0x04, 0x8f, 0x44, 0x02,
0xcd, 0x19, 0xbe, 0x82, 0x7d, 0x8b, 0x7d, 0x0f, 0x83, 0xff,
0x02, 0x72, 0xc8, 0x8b, 0xc7, 0x48,
0x48, 0x8a, 0x4e, 0x0d, 0xf7, 0xe1, 0x03, 0x46, 0xfc, 0x13,
0x56, 0xfe, 0xbb, 0x00, 0x07, 0x53,
0xb1, 0x04, 0xe8, 0x16, 0x00, 0x5b, 0x72, 0xc8, 0x81, 0x3f,
0x4d, 0x5a, 0x75, 0xa7, 0x81, 0xbf,
0x00, 0x02, 0x42, 0x4a, 0x75, 0x9f, 0xea, 0x00, 0x02, 0x70,
0x00, 0x50, 0x52, 0x51, 0x91, 0x92,
0x33, 0xd2, 0xf7, 0x76, 0x18, 0x91, 0xf7, 0x76, 0x18, 0x42,
0x87, 0xca, 0xf7, 0x76, 0x1a, 0x8a,
0xf2, 0x8a, 0x56, 0x24, 0x8a, 0xe8, 0xd0, 0xcc, 0xd0, 0xcc,
0x0a, 0xcc, 0xb8, 0x01, 0x02, 0xcd,
0x13, 0x59, 0x5a, 0x58, 0x72, 0x09, 0x40, 0x75, 0x01,
0x42, 0x03, 0x5e, 0x0b, 0xe2, 0xcc, 0xc3,
0x03, 0x18, 0x01, 0x27, 0x0d, 0x0a, 0x49, 0x6e, 0x76,
0x61, 0x6c, 0x69, 0x64, 0x20, 0x73, 0x79,
0x73, 0x74, 0x65, 0x6d, 0x20, 0x64, 0x69, 0x73, 0x6b, 0xff,
0x0d, 0x0a, 0x44, 0x69, 0x73, 0x6b,
0x20, 0x49, 0x2f, 0x4f, 0x20, 0x65, 0x72, 0x72, 0x6f, 0x72,
0xff, 0x0d, 0x0a, 0x52, 0x65, 0x70,
0x6c, 0x61, 0x63, 0x65, 0x20, 0x74, 0x68, 0x65, 0x20,
0x64, 0x69, 0x73, 0x6b, 0x2c, 0x20, 0x61,
0x6e, 0x64, 0x20, 0x74, 0x68, 0x65, 0x6e, 0x20, 0x70,
0x72, 0x65, 0x73, 0x73, 0x20, 0x61, 0x6e,
0x79, 0x20, 0x6b, 0x65, 0x79, 0x0d, 0x0a, 0x00, 0x49, 0x4f,
0x20, 0x20, 0x20, 0x20, 0x20, 0x20,
0x53, 0x59, 0x53, 0x4d, 0x53, 0x44, 0x4f, 0x53, 0x20, 0x20,
0x20, 0x53, 0x59, 0x53, 0x80, 0x01,
0x00, ´J´ , ´O´ , ´ ´ , ´ ´ , ´ ´ , ´ ´ , ´ ´ , ´ ´ , ´S´ , ´Y´ , ´
S´ , 0x00, 0x00, 0x55, 0xaa,
};
unsigned char OSRBoot[512];
void main()
{ int i;

  puts(˝Changing your OSR2 Boot˝);
  if(absread(2, 1, 0, OSRBoot)! =0)
  { puts(˝Read original boot error˝);
    return;
   }
  for(i =0x0b; i <0x36; i + +)
  W95Boot[i] =OSRBoot[i];
  if(abswrite(2, 1, 0, W95Boot)! =0)
  puts(˝Write new boot error! ˝);
  puts(˝OK! The OSR2 boot is changed. ˝);
 }
```

将上述程序用 Borland C 3. 1 编译链接成可执行文件，在需要修改的机器上 DOS 环境下运行。就能够解决问题。

参考文献

1. 李建伟 . Win95 OSR2 双重引导问题 . 计算机世界，No. 12 1997

2. 博战捷 . Windows 95 与 MS - DOS 双重引导问题解决一例 . 计算机应用研究，No. 4 1999

3. 李莹 . 兼容安装 Windows95 的两种对策 . http / / www. zhanjiang. gd. cn / home / jinyt / qd / dn / myjy / gao / ANWIN95. TXT

4. 刘崇 . Windows95 和 DOS6. 22 双平台的安装与运行 . http / / www. swm. com. cn / yingyong / rj - 98 - yy8 / 98 - y8 - yy5. htm

5. 杨迈等 . 软件加密 / 解密反跟踪实用技术 . 西安电子科技大学出版社，1993，12

收稿日期：2000 年 12 月 12 日）

AutoCAD 中表面粗糙度标注工具的二次开发

易春峰 张锡滨

摘 要 本文介绍了在用 AutoCAD 绘制机械零件工作图时，如何使用带属性的块与 AutoLisp 综合编程的方法，对表面粗糙度标注工具进行二次开发，讨论了表面粗糙度标注工具的使用技巧，提供了标注表面粗糙度的 AutoLisp 程序代码。

关键词 表面粗糙度标注，AutoLisp 编程，二次开发

一、 引言

在 AutoCAD 中绘制零件图时，标注零件表面粗糙度是一项十分繁琐的工作。虽然在 AutoCAD 中也可以用插入带属性的图块的方式来提高表面粗糙度的标注效率，但在一个复杂的机械零件工作图中，常常要标注数十个表面粗糙度符号，且必须符合机械制图标准规范，如图 1 所示，在两个特殊方位还必须使用指引线方式，那么仅仅使用插入带属性的图块的方法，是不能满足需要的。这就迫切需要一种灵活高效的表面粗糙度标注工具。因此，我们利用带属性块与 AutoLisp 综合编程方法开发出一个高效实用的表面粗糙度标注工具。它具有如下特点：自由旋转粗糙度符号并根据粗糙度符号的方位自动调整粗糙度值的方位；自动识别特殊方位并自动改用引线方式标注；同一粗糙度值可作为默认值重复标注直到更改成下一默认值；粗糙度符号循环标注直到按 Esc 键退出；粗糙度符号的大小可调。

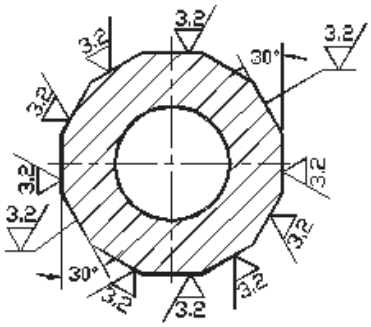


图 1

二、 表面粗糙度标注工具的开发方法

粗糙度标注工具是把 AutoCAD 的属性块与 AutoLisp 编程结合起来，用 AutoLisp 编写插入属性块的程序来实现的，这样可以保证在标注表面粗糙度时既方便直观，又高效智能化，同时也便于用属性的特点管理粗糙度。

2. 1 图块文件的制作

在编写 AutoLisp 程序之前，先在 AutoCAD 的安装目录中作好下面四个待插入的块文件：

1 粗糙度符号块文件：“CCDY. DWG”，要求必须按以下顺序绘制三条线段，从点 D 到点 C 到点 B 到点 A，如图 2 所示。线段的长度 AB = BC = CD / 2 = 10 基准点 Base point 为 C 点。

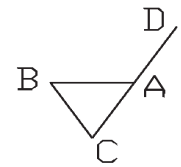


图 2

2）右对齐方式（MR）属性块文件：“CCDZ. DWG”，高度 5 基准点（Base point）为属性文字的右中点（MR 点）。

3）左对齐方式（ML）属性块文件：“CCDF. DWG”，高度 5 基准点（Base point）为属性文字的左中点（ML 点）。

4 带右对齐方式的属性的图块文件：“CCD. DWG”，如图 3 所示，线段长度与基准点同图块文件“CCDY. DWG”，属性文字高度 5。

图 3

2. 2 编写 AutoLisp 程序

因常有同一粗糙度的值重复标注多处的情况，为提高效率，编程时，用可修改的变量 ccd，记录粗糙度的临时默认值 当前值 作为待插入块的属性值，标注完所有某一个粗糙度值之后，修改变量 ccd，又可以一次性地标注完所有另一个粗糙度值，而插入块的缩放比，有两种处理方法，一是将其记录于一个在程序每次启动时可以设置，而在程序运行当中即循环插入块的过程中不在可见的变量 sc 之中，以提高粗糙度标注工具的使用灵活性，即若对粗糙度符号的大小不满意，只需再启动一次便可重新设置缩放比；二是将缩放比记录于一个在程序第一次启动时可以设置，同一图形文件中程序再次启动时不再可见，且程序运行中亦不在可见的变量 sc 之中，以提高绘图效率，但为了兼顾灵活性，可在本程序之外，单独更改缩放比变量 sc。经过实践，发现第二种处理方法效率更高，因为当图形较大、较复杂时，屏幕上只能看清图中的局部区域，往往会随时中断粗糙度的标注，然后又多次启动粗糙度标注工具，才能将图中的所有粗糙度符号标注完毕，如果采用第一种处理方法，将不得不反复设置缩放比变量 sc，而降低了绘图效率。最后，将编好的 AutoLisp 程序以文件名“CCDY. LSP”存入 AutoCAD 的安装目录（文件夹）中。

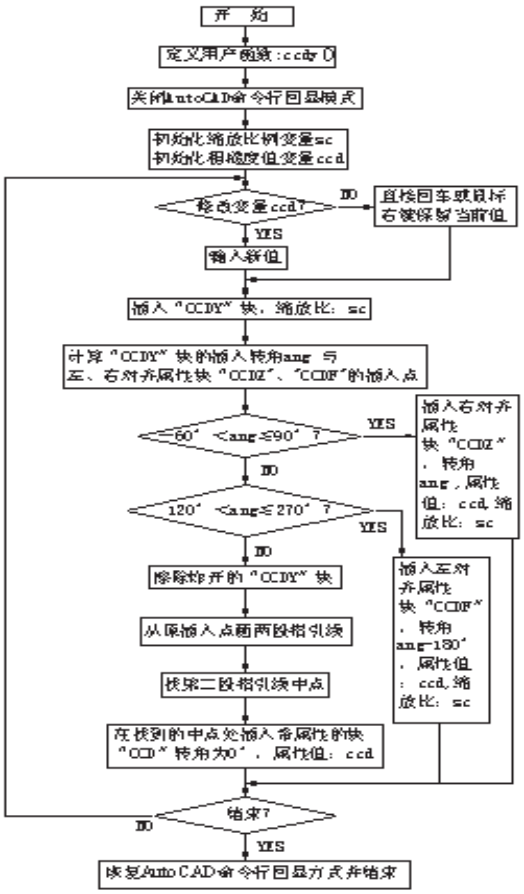


图 4

三、技术关键

本程序的关键和难点是如何测出粗糙度符号块："CCDY"的插入转角及右、左对齐属性块"CCDZ"与"CCDF"的插入点。若将图 2 视为插入后的"CCDY"图块，则只要求出 A 点和 B 点的坐标就可算出粗糙度符号块："CCDY"的插入转角，再利用插入点 C 的坐标即可推算出属性块"CCDZ"与"CCDF"的插入点。

要解决这一问题，需要利用 AutoLisp 的 (entlast) 函数，获取刚插入"CCDY"块的实体名，存入变量 a，再用 (entget a) 函数，获取块"CCDY"的实体数据表，存入变量 b，然后用 (nth n b) 函数，取块的插入点表，存入变量"XY"，再用 (adr xy) 与 (addr xy)，就可以求得插入点 C 点的 X 坐标和 Y 坐标了。而要求 A 点与 B 点的坐标，必须先炸开 (explode) "CCDY"块→ (Command "explode entLast") 然后用 (entlast) 与 entget 函数，获取直线 AB 的实体数据表,存入变量 b,再用 (nth m b) nth n b 即可获取线段 AB 的端点 A 点与 B 点的点表,进而求出它们的 X、Y 坐标。这样就可以利用所求出的 A 点与 B 点的 X、Y 坐标算出块的插入转角，利用 A 点、B 点与 C 点的坐标，算出属性块"CCDZ"与"CCDF"的插入点了。

四、 粗糙度标注工具的使用方法

1 只要在命令行键入 (load "CCDY") 回车就可以将"ccdy"按一般的 AutoCAD 命令一样，在命令行中使用。使用时循环插入粗糙度符号，直到按 ESC 键中断命令为止！

2 若要进一步提高使用效率和方便性，可以定制一个粗糙度的图标按钮，将其菜单宏定义为： C C (load "ccdy") ccdy ，以加快每次输入命令的速度。

3 在"CCDY. lsp"卸载之前，要更改图块插入的缩放比可再定制一个"更改缩放比"的图标按钮，将其菜单宏定义为： C C setq sc getreal "请输入新绘图比例："这样就可在需要更改粗糙度符号的大小时，灵活更改。而不需改时，则默认上次设置的缩放比反复工作，避免频繁设置以提高效率。

五、 结束语

AutoCAD 是一个通用的 CAD 软件平台，要想用 AutoCAD 高效绘图必须进行二次开发。表面粗糙度标注是一项繁琐的重复性任务，最有必要进行二次开发，以扩充 AutoCAD 的相关功能，而利用插入属性块与 AutoLisp 编程的方法，结合定制相应的图标按钮，是解决此类问题，提高绘图效率的有效方法。

六、附 14. 0 与 2000 版的 AutoLisp 源程序（文件名：CCDY. LSP）

```
(defun c: ccdy(); 定义 ccdy()用户函数
(setq jsf 1)
(setvar "cmdecho" 0); 关闭 AutoCAD 命令回显模式
(if (= sc nil) (setq sc (getreal "请输入缩放比例: <1>")))
(if (= sc nil) (setq sc 1.0)); 设初始缩放比
(if (= ccd nil) (setq ccd "3.2")); 设默认粗糙度值
(while (= jsf 1); 设条件循环
 (princ "请输入粗糙度的值: <当前值: ")
 (princ ccd)
 (princ ">: ")
 (setvar "cmdecho" 1)
 (setq cc (getstring " "))
 (setvar "cmdecho" 0)
 (setq ccd (if (/= cc "") cc ccd)); 设当前粗糙度默认值
 (princ "请输入插入点: ")
 (command "-osnap" "nea" "INSERT" "ccdy" "S" sc PAUSE PAUSE ); 将物体捕捉方式设为 "nea"并插入 "CCDY"块
 ;;;;;;;;;;;; 获取插入点坐标 x, y
 (setq a (entlast))
 (setq b (entget a))
 (setq xy (nth 8 b));;; 注: 若为 ACAD2000 版此处改为(setq xy (nth 10 b))
 (setq x (cadr xy))
 (setq y (caddr xy))
```

(下转第 92 页)

funlove. 4099 病毒的分析与清除

齐玉东

摘 要 本文对 funlove. 4099 病毒进行了详细地分析,并给出了清除病毒的方法。
关键词 病毒,PE,服务

前言

FunLove. 4099 是一个运行在 Win9X、WinNT 和 Win2000 环境下的文件型病毒 它感染 PE 格式的文件。除了感染本地硬盘上的所有的 PE 格式文件外,它还会周期性地检查网络上的可写的共享网络驱动器,并感染其上的文件。另外它还修改 NTOSKRNL. EXE 和 NTLDR 等 NT 文件, 这样它就能在下次系统重启动后拥有全部权限。病毒在 Win9x、Win2000 或 NT 的 SYSTEM 目录下建立一个名为 flcss. exe 的文件,大小为 4099 字节,正常文件被病毒感染后,最后一节被置入 FLCSS. EXE。当在 DOS 下运行 FLCSS. EXE 时便会显示 ~ Fun Loving Criminal ~ ,并企图重新启动计算机。

病毒采用了多线程的编程方法,在 Win9x、Win2000 或 NT 中创建一个服务进程。这样系统一旦启动完毕,病毒程序便开始运行,直到系统关闭为止。由于病毒程序是作为服务程序存在的,因此在 Win9X 下,我们在任务列表中将看不到病毒进程的存在,在 NT 下,当我们试图在服务管理器中,关闭这一服务时,将得到一错误提示,却停止不了病毒程序的执行。另外,病毒在传染时,会跳过对一些反病毒软件的感染,因此会使它更加难以发现。病毒代码使用了 API 函数的重定位技术、递归程序调用技术、Win32 服务程序编程技术以及 PE 文件生成技术。下面先对此病毒程做一分析,然后给出清除病毒的方法。

一、病毒的分析

(一) 病毒的主程序分析

病毒首先获取其基地址,并将其保存在寄存器 EBX 中,在以后的引用病毒数据区中的数据、拷贝病毒代码及生成 flcss. exe 时,都要用到它。为了能够返回宿主程序,病毒随后恢复了宿主文件的正常入口地址。恢复程序正常入口点的代码片段如下:

```
lea         esi, [HostCode @ ]
mov         edi, [esp]
sub         edi, 08
mov         [esp], edi
movsd
movsd
```

病毒感染文件时,将可执行文件的入口点处的 8 个字节保存起来,而在其中放了一条 CALL 指令。这里将原来的 8 个字节压栈,病毒主程序退出时,便将控制权转给被其感染的程序。

接下来病毒的工作是对 API 函数的重定位。由于病毒是附在 PE 文件的后面的,它没有自己的输入函数表,不能象正常 PE 文件那样调用 API 函数,因此必需采用别的方法调用 API 函数。病毒的做法是先找到 Kernel32. dll 的内存映象的基地址,然后找到 GetProcAddress 的入口地址,这样,其它的 API 函数的入口地址可以利用 GetProcAddress 得到。不同操作系统下的 Kernel32. dll 的基地址是不一样的,而且 GetProcAddress 入口处的代码也不相同。在 Win32 环境下,执行一个进程的第一条指令时,堆栈顶部存放的便是操作系统的 Kernel32. dll 的基地址。病毒就是利用了 Win32 的这个特点,获得 kernel32. dll 的基地址,然后根据不同操作系统下的 GetProcAddress 入口处代码的不同,找到这个 API 函数的入口点。接着便调用 GetProcAddress 得到病毒需要使用的 Kernel32. dll 中的函数。若是要使用其它 DLL 中的函数,只需先用 LoadLibrary 将这个 DLL 调入内存,然后使用 GetProcAddress 获得从其中引入的函数入口。

若程序在 Win9X 下运行,首先在 system 目录下创建 flcss. exe,然后调用 CreateProcess 运行它。若程序在 NT 下运行,首先在 system 目录下创建 flcss. exe,然后创建并运行这个服务进程。在创建进程时,首先调用 OpenSCManagerA 获得服务管理器的句柄,若 flcss 服务正在运行,则调用启动传染线程的过程,启动另一个传染线程,否则创建这个服务并启动服务。这部分代码如下:

```
xor         eax, eax
push        eax
push        eax
push        eax
push        eax
push        eax
lea         eax, [Buffer1 @ ] ;  -> flcss. exe
push        eax
push        01
push        02
push        20
push        edi
push        00
push        esi
```

```
    push        SCM_Handle
    call        CreateServiceA
    or          eax, eax
    jz          short CNT_Failed
CNT_Run:
        push        00
        push        00
        push        eax
        call        StartServiceA
```

（二）flcss. exe 的分析

flcss. exe 从功能上可分为两大块，一部分用来实现一个在 NT 下运行的服务进程，另一部分用来实现一个在 WIN9X 下运行的进程。

1. NT 下的病毒服务程序的实现

病毒首先重定位了从 Advapi32. dll 中所引用的函数，然后调用 StartServiceCtrlDispatcher 函数，将 flcss. exe 注册到 SCM 里，并传给 SCM 一个指向服务主函数的指针，然后跳到主函数去执行。主函数首先调用 RegisterServiceCtrlHandler 函数，该函数为注册句柄函数。RegisterServiceHandler 会返回一个句柄，当服务程序需要送消息给 SCM 时就通过这个句柄。注意，病毒的句柄函数只有一条 ret 指令，因此当我们在服务管理器中试图停止这个服务时，将得到一个来自 SCM 的一个错误提示，而不能停止病毒服务。主函数注册句柄函数后，将服务的状态设为运行，潜伏 8 秒钟后，启动传染线程。

2. WIN9X 下的病毒服务程序的实现

病毒首先注册其主窗口的类型，设为不可见的，然后调用 RegisterServiceProcess 函数，注册服务进程，这部分代码为：

```
    push        01                  ; 服务的属性
    push        eax                 ; eax 为病毒进程 ID
    call        RegisterServiceProcess
```

这里将病毒进程设为服务进程的目的是不让该进程出现在任务列表中。

3. DOS 头信息

我们知道，当在 DOS 下执行一个 WIN32 程序时，会出现“This program cannot be run in DOS mode”的信息。这条信息是放在 PE 文件的 DOS 头中的，而 flcss. exe 则将相应的信息置换为“~ Fun Loving Criminal ~”，正常的可执行文件在显示了上述信息后，会调用 INT 21H 的 4C 子功能退出程序，而 flcss. exe 则是使用了一个无限循环，造成死机，使我们不得不重新启动计算机。

（三）传染线程的分析

从上面的分析中，我们可以看到，当执行一个染毒文件时，启动了两个传染线程，一个是在染毒程序的进程空间运行，另一个在 flcss. exe 的进程空间运行。传染线程的代码为：

```
VThread   PROC            NEAR
    call        GetVS           ; 获得病毒代码的基地址
    call        InfectDrives    ; 传染本地驱动器上的文件
    push        60d * 1000d
    call        Sleep           ; 潜伏 60 秒
    call        GetRand         ; 产生随机数
    and         al, 1F
    jnz         short VThread
    call        InfectNetwork ; 传染网络驱动器上的文件
    jmp         short VThread
VThread                 ENDP
```

可见传染线程是一个无限循环的过程，从线程启动开始，一直到线程被停止，病毒总是在试图传染文件，传染一个本地驱动器上的文件后，为了更具有隐蔽性，病毒先潜伏 60 秒，然后再随机决定是传染下一个本地驱动器上的文件，还是传染网络驱动器上的文件。

1. 传染本地驱动器上的文件

病毒首先判断驱动器的类型，若是软盘，则退出，否则试图找到 ntldr 和 WINNT hSystem32 hntoskrnl. exe，并修改它们，接着调用一子过程搜索此驱动器上的文件，找到可传染的文件并传染它。最后判断是否已传染到 Z 盘，若是，则退出，否则接着传染下一个驱动器上的文件。

2. 传染网络上的文件

为了找到网络上的文件并感染它们，为此病毒构造了一个网络搜索过程。在这个过程中，病毒首先列出网络上的所有资源，若该一项资源是一个硬盘驱动器，则进行与传染本地驱动器上的文件一样的过程；若该资源是一个包含性的资源，则递归调用网络搜索过程；若一项资源既不是硬盘驱动器，也不是包含性的资源，则继续处理下一项资源。网络搜索过程的递归调用直到所有的网络资源都被处理为止。在进行网络搜索的过程中，病毒使用了从 mpr. dll 中引入的 API 函数 WnetOpenEnum、WnetEnumResource 和 WnetCloseEnum 函数。我们可以从 MSDN 中找到这些函数的用法，这里不再详细说明。

3. NT 装载和内核函数的修改和文件搜索

无论是在传染本地驱动器上的文件还是在传染网络上的文件过程中，都调用了下面这段代码：

```
        push        edi
        call        BlownAway
        push        edi
        call        FileSearch
```

其中第一个 CALL 调用，用来修改 NT / WIN2000 的系统装载程序 ntldr 和内核程序 ntoskrnl. exe。在这个子过程中，首先打开 ntldr，并将其映射到内存中，然后找到判断系统文件（包括 ntoskrnl. exe）是否正确那部分代码。以 NT 为例，这部分代码为：

```
CMP     AX, [BP + 58]
JZ      XXX
```

一旦找到它，病毒便把它修改为：

```
CMP     AX, [BP + 58]
JZ      XXX
```

这样，不管对系统文件检查结果如何，系统文件都会被加载。

修改 ntldr. sys 后，病毒会用同样的方法在 ntoskrnl. exe 中，查找检查访问权限的代码，并修改它，这样，当 NT 下次重启时，将使得 NT / Win2000 的权限检查功能失效。

第二个 CALL 调用用来搜索当前目录下的所有文件或目录，若搜索到的是一个文件，则判断是否可以传染该文件，如果这个文件可以传染，则立刻调用传染子过程来传染该文件；若找到的是一个子目录，则设此子目录为当前目录，递归调用文件搜索过程。病毒在其数据区中放置了一些反病毒软件的信息，病毒在搜索文件的过程中，若查找到一些文件名中包含有这些信息，则跳过它们，不予处理。

另外，病毒在搜索文件的过程中，还检查文件的后辍，若文件的后辍是“xco”，“fcs”，“exe”中的其中之一，则调用传染过程准备传染该文件，否则跳过这个文件。

4. 传染文件过程

首先病毒以读写模式打开准备传染的文件，并把文件的大小、创建日期等信息保存起来。然后判断该文件是否已被传染，判断过程如下：

```
push        00
push        eax ; eax 中为文件句柄
call        GetFileSize
mov         i_FileSize, eax ; 保存文件大小
cmp         al, 03
jz          IN_Exit    ; 若文件已传染, 则退出
```

判断条件是，假设文件的大小为 X 字节，若 X 除于 256 的余数等于 3，则认为该文件已被感染。

若还没有被传染，则将此文件映射到内存中，即创建一个内存映象文件，这是为了加快传染速度。若此文件是 PE 格式文件，则先把程序的入口点保存到距病毒首部偏移 BA7 处，然后把原文件最后一节数据的大小补齐至 4096 字节，接着把病毒代码附加在此文件之后，并把病毒代码补足 4096 字节，我们知道通常一个 PE 文件每一节都补齐至 4K 大小，因此经过上述操作后，一个染毒文件肯定会满足上面提到的判断条件。接着计算出病毒入口点的位置，然后在原来的程序正常入口点放置一条调用病毒入口处代码的 CALL 指令，以调用病毒代码。这样，程序一旦执行，首先就要执行这条 CALL 指令，使得病毒获得控制权。注意，因为病毒代码被附加在可执行文件的最后一节，为了其本身的一些特性，如对其自身代码的写操作和对传染速度的要求，病毒修改了最后一节的属性，进行的操作如下：

```
or          eax, 8C000000   ; eax 中是最后一节的节头中的;
                            Characteristics 域
and         eax, not 12000000
```

这里将该节数据的属性设为可写的、不进行页交换、非共享和不可丢弃的。

为了增加传染的隐蔽性，病毒在保存并关闭文件之前恢复了文件的创建时间。

二、病毒的清除

为了清除病毒，我们要进行的工作是：

- 清除附在染毒文件上的病毒代码并恢复原程序入口点
- 删除 system 目录下的 flcss. exe
- 修复 NT / Win2000 的装载程序 ntldr 和内核程序 ntoskrnl. exe
- 从 NT / Win2000 的服务数据库中删除病毒服务

在一个染毒的操作系统环境中，可能会存在多个病毒传染线程，若我们要在 Win32 环境中杀毒，则我们首先要终止病毒的的服务进程，还要找到每一个传染线程并终止它们，然后再清除每个染毒文件中的病毒代码，并且恢复原程序入口点。这样，程序实现起来较为复杂，另外，当我们终止病毒传染线程的时候，有可能病毒正在打开一个文件并在传染它，这样容易造成文件损坏。所以我们最好在 DOS 下清除病毒，这种情况下，只需修复染毒文件即可。因为 flcss. exe 在 NT / Win2000 启动时会自动做为服务被启动，因此我们也需要在 DOS 下删除此文件。为了修复 NT / Win2000 的装载程序 ntldr 和内核程序 ntoskrnl. exe，我们需要打开这两个文件，然后找到病毒所修改的地方，还原正确的代码。最后我们还要在 NT / Win2000 下，从服务数据库中删除病毒所注册的服务。下面给出在 DOS 下清除病毒的实现，然后给出删除病毒服务的方法。删除 flcss. exe 和修复 ntldr 和 ntoskrnl. exe 的程序实现这里没有给出，参看源代码。

（一）在 DOS 下，修复染毒文件

这里给出了一个函数 Remove，它的入口参数是一个文件名。在这个函数中，首先做了三个判断：

- 初步判断该文件是否一个染毒文件，判断条件见上面所述
- 距文件结尾偏移 4049 处，是否有信息 “~ Fun Loving Criminal ~ ”
- 此文件是否为 PE 文件

若这些条件都满足，则认为此文件即是一个染毒文件，先恢复原程序入口点，然后去掉病毒代码并保存文件，最后关闭文件。

函数 Remove 实现如下：

```
void Remove(char * filename)
 {
  int handle;
  int exe_flag = 0;
  long pe_offset;
  long pe_flag = 0;
  int NumberOfSections = 0;
  int SizeOfOptionalHeader = 0;
  char virus_mark[22] = {0};
  char host_code[8] = {0};
  long AddressOfEntryPoint = 0;
```

```
  long pAddressOfEntryPoint =0;
 IMAGE_SECTION_HEADER section_data; //声明一个 PE 节头类型变量
  char zero_code[4099] ={0};
handle = open(filename, O_RDWR, S_IREAD | S_IWRITE);
  if(handle = = -1)
       return ;
  long filesize =filelength(handle);
  if(filesize%256 = =3){//通过判断文件的长度,初步判断该文件是否染毒
       lseek(handle, 0L, SEEK_SET);
read(handle, & exe_flag, 2); /*判断该文件是不是一个可执行文件*/
      if(exe_flag = =0x5a4d)
        {
          lseek(handle, 0x3c, SEEK_SET);
      read(handle, & pe_offset, 4); /*定位到 PE 标志处*/
  lseek(handle, pe_offset, SEEK_SET);
      read(handle, & pe_flag, 4); /*读出 PE 标志*/
            if(pe_flag = =0x00004550)
            {
       lseek(handle, -(0x1003 -0x50), SEEK_END);
  read(handle, virus_mark, 21); /*读出病毒标志*/
  virus_mark[21] =0;
  if(strcmp(virus_mark, "~Fun Loving Criminal~") = =0)
       {
   lseek(handle, -(0x1003 -0xba7), SEEK_END); read(handle, & host_code, 8); /*读出原程序入口处的代码*/
  lseek(handle, pe_offset +6, SEEK_SET); read(handle, & NumberOfSections, 2); /*读出文件的节数*/
  lseek(handle, pe_offset +0x14, SEEK_SET);
  read(handle, & SizeOfOptionalHeader, 2);
lseek(handle, pe_offset +0x28, SEEK_SET);
  read(handle, & AddressOfEntryPoint, 2);
for(int section =0; section <NumberOfSections; section + +)
  {
lseek(handle, pe_offset +0x18 +SizeOfOptionalHeader +section*0x28, SEEK_SET);
  read(handle, & section_data, 0x28); /*读取节数据*/
if((AddressOfEntryPoint -section_data. VirtualAddress>0)
& &  (section_data. SizeOfRawData>AddressOfEntryPoint -section_data. VirtualAddress))
  {
  /*获得程序入口点在文件中的物理位置*/
        pAddressOfEntryPoint =AddressOfEntryPoint -section_data. VirtualAddress +section_data. PointerToRawData;
lseek(handle, pAddressOfEntryPoint, SEEK_SET);
write(handle, host_code, 8); /*恢复程序入口点的代码*/
lseek(handle, -4099, SEEK_END);
write(handle, zero_code, 4099); /*将病毒代码置零*/
printf("found funlove. 4099 virus in %s , and killed\n", filename);
  break;
  }
  }
  }
  }
  }
close(handle);
return ;
}
```

仁）删除病毒服务

在 NT/Win2000 下，我们先调用 OpenService 获得病毒服务句柄，然后调用 DeleteService 删除此项服务。代码片段如下：

```
  VOID DeleteVirusService()
  {
  SC_HANDLE schSCManager;
  SC_HANDLE schService;
  /*取得操作服务管理器数据库的句柄*/
  schSCManager = OpenSCManager(
NULL,                  //
  NULL,                   //
  SC_MANAGER_ALL_ACCESS);  //
if (schSCManager = = NULL)
     return;
  schService = OpenService(
                  schSCManager,
                  TEXT("FLC"), // 病毒服务的名字
                  DELETE);     // 删除该服务
       if (schService = = NULL)
            return;
       if (! DeleteService(schService) )
            return;
       else
MessageBox(NULL, "message""Delete Service SUCCESS",
MB_OK);
 CloseServiceHandle(schService);
  }
```

收稿日期 2000 年 12 月 11 日

Progress WebClient 问世 全面代替传统 GUI 仿真产品

Progress 软件公司日前宣布推出一款全新的软件——Progress& reg WebClient，作为现有的瘦客户机 GUI 仿真产品的替代方案。这款软件允许在 Web 上部署基于 Progress 的瘦客户机应用，最大限度地减少客户机上的网络流量和应用软件。通过利用现有的 GUI 客户机代码，WebClient 可以运行一种应用程序，在保留所有功能的同时，提供强大丰富的用户界面及可扩充的高性能结构。Progress WebClient 将负责运行用户界面代码，而由 Progress AppServer 完成数据库访问和其它商业逻辑和检验功能。用户将不再需要使用第三方仿真软件产品，如 Citrix 或 Tarantella，从而可以降低用户的运营成本，同时仍能在 web 或 ASP 模型中导入应用程序。

用 VB 编程实现隐藏驱动器盘符的方法

李学刚

Windows 可以通过对注册表进行修改实现在图形界面下隐藏某个磁盘驱动器（包括软驱、硬盘、光驱）的目的，其原理如下：

在注册表 HKEY_CURRENT_USER hSoftware hMicrosoft hWindows hCurrentVersion hPolicies hExplorer 中，如果有一个名为“NoDrives”的二进制值，根据 Windows 对该值的规定，相应隐藏驱动器盘符。从字面上即可看出 NoDrives 键值的作用是设置是否隐藏某个驱动器，它由四个十六进制字节构成，每个十六进制字节的值都分别对应一个磁盘驱动器的盘符，如果指定相应的值，那么资源管理器及“我的电脑”中的相应驱动器图标即会隐藏起来。NoDrives 隐藏驱动器符所对应的值如下表所示：

隐藏盘符	键值	隐藏盘符	键值	隐藏盘符	键值
A	01 00 00 00	I	00 01 00 00	Q	00 00 01 00
B	02 00 00 00	J	00 02 00 00	R	00 00 02 00
C	04 00 00 00	K	00 04 00 00	S	00 00 04 00
D	08 00 00 00	L	00 08 00 00	T	00 00 08 00
E	10 00 00 00	M	00 10 00 00	U	00 00 10 00
F	20 00 00 00	N	00 20 00 00	V	00 00 20 00
G	40 00 00 00	O	00 40 00 00	W	00 00 40 00
H	80 00 00 00	P	00 80 00 00	X	00 00 80 00
全不隐藏	00 00 00 00	Y	00 00 00 01	Z	00 00 00 02
全部隐藏	FF FF FF FF	注：表中键值具有可加性，也就是说隐藏多个驱动器时将相应键值相加即可达到目的。			
全部隐藏	键值为空				

根据自己驱动器盘符的实际情况对“NoDrives”的值进行修改，就可以达到隐藏的目的。若隐藏盘符 D 则应修改为 08 00 00 00；若光驱盘符为 K 则应修改为 00 04 00 00。若将 C 盘和 H 盘同时隐藏怎么办？按照上表备注提示，将“NoDrives”的值相加即改为 84 00 00 00 即可，这里 84 = 04 + 80。再例如，将 C 盘、F 盘和 H 盘同时隐藏掉，将“NoDrives”的值改为 A4 00 00 00 即可，这里的 A4 = 04 + 20 + 80。那位朋友该问了：04 + 20 + 80 = 104，怎么等于 A4？因为这是是十六进制加法。假如我的光驱盘符是 K，要将软盘 A、C 盘、F 盘、H 盘和光盘同时隐藏掉，将“NoDrives”的值改为 A5 04 00 00 即可。

上述内容是手工修改注册表，实际使用起来显得很麻烦，为了方便起见，利用 VB 编制了一个小程序，来实现隐藏驱动器盘符的目的。程序的原理是利用调用 API 实现对注册表的操作，即实现上述根据使用者的要求向注册表写入数据，达到隐藏驱动器盘符的目的。

软件的编制采用 VB 制作，由三部分组成，两个窗体和一个模块。

一、“窗体 1”窗体

在“窗体 1”创建三个 label，其中 label1 和 label2 用于显示标题，属性为透明，字体随便设置，内容在设计时指定。label3 用于动态显示文字，内容参看程序。然后再创建一个 Image，它的 picture 指向 pig. ico 文件。最后创建五个按钮和 26 个 CheckBox。“窗体 1”的主要作用是实现隐藏操作。

“窗体 1”的代码如下：

```
Private Sub Form_MouseMove(Button As Integer, Shift As Integer, X As Single, Y As Single)
Label3. Visible = False ′鼠标移出 label3 时, 关闭 label3 显示
End Sub
Private Sub Image1_dblClick()
′单击小猪图标浏览网页
Shell ″start http: //mrlxg. myetang. com″
End Sub
Private Sub Image1_Click()
′双击小猪图标复活节彩蛋标志为 1
Egg = 1 ′已定义为全局变量
End Sub
Private Sub Image1_MouseMove(Button As Integer, Shift As Integer, X As Single, Y As Single)
Label3. ForeColor = & HFF ′红色
Label3. Caption = ″双击这只小猪前往猪八戒乐园背媳妇″
Label3. Visible = True ′显示 label3
End Sub
Private Sub 关于按钮 _Click()
关于 . Show ′显示″关于 . Frm″窗体
End Sub
Private Sub 全显示 _Click()
′令 26 个 Check 清零
CheckA = 0: CheckB = 0: CheckC = 0: CheckD = 0
CheckE = 0: CheckF = 0: CheckG = 0: CheckH = 0
CheckI = 0: CheckJ = 0: CheckK = 0: CheckL = 0
CheckM = 0: CheckN = 0: CheckO = 0: CheckP = 0
CheckQ = 0: CheckR = 0: CheckS = 0: CheckT = 0
CheckU = 0: CheckV = 0: CheckW = 0: CheckX = 0
CheckY = 0: CheckZ = 0
End Sub
Private Sub 全隐藏 _Click()
′令 26 个 Check 为 1
CheckA = 1: CheckB = 1: CheckC = 1: CheckD = 1
CheckE = 1: CheckF = 1: CheckG = 1: CheckH = 1
CheckI = 1: CheckJ = 1: CheckK = 1: CheckL = 1
CheckM = 1: CheckN = 1: CheckO = 1: CheckP = 1
CheckQ = 1: CheckR = 1: CheckS = 1: CheckT = 1
```

```
CheckU = 1: CheckV = 1: CheckW = 1: CheckX = 1
CheckY = 1: CheckZ = 1
End Sub
Private Sub 执行按钮_Click()
'先清零,实际程序应该读取注册表获得原来的值,这里省略
A=0: B=0: C=0: D=0: E=0: F=0: G=0: H=0
I=0: J=0: K=0: L=0: M=0: N=0: O=0: P=0
Q=0: R=0: S=0: T=0: U=0: V=0: W=0: X=0
Y=0: Z=0
'以下各 CheckBox 的值对应要隐藏的驱动器盘符
If CheckA = 1 Then A = &H1 '隐藏 A:
If CheckB = 1 Then B = &H2 '隐藏 B:
If CheckC = 1 Then C = &H4 '隐藏 C:
If CheckD = 1 Then D = &H8 '隐藏 D:,以下依次类推
If CheckE = 1 Then E = &H10
If CheckF = 1 Then F = &H20
If CheckG = 1 Then G = &H40
If CheckH = 1 Then H = &H80
If CheckI = 1 Then I = &H1
If CheckJ = 1 Then J = &H2
If CheckK = 1 Then K = &H4
If CheckL = 1 Then L = &H8
If CheckM = 1 Then M = &H10
If CheckN = 1 Then N = &H20
If CheckO = 1 Then O = &H40
If CheckP = 1 Then P = &H80
If CheckQ = 1 Then Q = &H1
If CheckR = 1 Then R = &H2
If CheckS = 1 Then S = &H4
If CheckT = 1 Then T = &H8
If CheckU = 1 Then U = &H10
If CheckV = 1 Then V = &H20
If CheckW = 1 Then W = &H40
If CheckX = 1 Then X = &H80
If CheckY = 1 Then Y = &H1
If CheckZ = 1 Then Z = &H2
C1 = A+B+C+D+E+F+G+H '隐藏的盘符具有可加性
C2 = I + J + K + L + M + N + O + P
C3 = Q + R + S + T + U + V + W + X
C4 = Y + Z
SetBinaryValue "HKEY_CURRENT_USER\Software\Microsoft\
Windows\CurrentVersion\Policies\Explorer", "NoDrives",
Chr$(C1)+Chr(C2)+Chr(C3)+Chr(C4) '向注册表写二进制数据
MsgBox "按确定返回后退出,然后重新启动(或注销)即可生效",,"操作成功"
End Sub
Private Sub 退出按钮_Click()
End
End Sub
Private Sub Form_Load()
Move (Screen.Width - Width) \ 2, (Screen.Height -
Height) \ 2 '保证窗体居中
Label3.Visible = False
End Sub
```

二、 "关于"窗体

代码如下:

```
Private Sub Command1_Click()
关于.Visible = False '隐藏关于窗体面板
窗体 1.Visible = True '显示主窗体面板
'以下恢复默认值
Egg = 0
Label2.ForeColor = &H0
Label2.Caption = "程序设计: 猪八戒"
Label3.Caption = "给猪八戒抛一个媚眼"
Label4.Caption = "去猪八戒乐园背媳妇"
Command1.Caption = "讨厌"
关于.Caption = "关于钉耙乱舞"
Image2.Visible = False
Image1.Visible = True
End Sub
Private Sub Form_Load()
窗体 1.Visible = False '面板不显示
Move (Screen.Width - Width) \ 2, (Screen.Height -
Height) \ 2 '窗体居中
Label2.ForeColor = &H0
Label2.Caption = "程序设计: 猪八戒"
Label3.Caption = "给猪八戒抛一个媚眼"
Label4.Caption = "去猪八戒乐园背媳妇"
Command1.Caption = "讨厌"
关于.Caption = "关于钉耙乱舞"
Image2.Visible = False
Image1.Visible = True
End Sub
Private Sub Image1_dblClick()
'双击关于窗体中的小猪图标是否激活复活节彩蛋
If Egg = 1 Then '复活节彩蛋标志,即在窗体 1(面板)里单击了小猪图标
mrlxg = Chr(-16146) + Chr(-11865) + Chr(-18219)
Label2.ForeColor = &HFF0000
Label2.Caption = "程序设计:" + mrlxg
Label3.Caption = "给" + mrlxg + "发电子邮件"
Label4.Caption = "访问" + mrlxg + "个人主页"
Command1.Caption = "确定"
关于.Caption = "关于隐藏驱动器盘符"
Image1.Visible = False
Image2.Visible = True
Else
MsgBox "想捕获复活节彩蛋?方法不对哟"
End If
End Sub
Private Sub Label3_Click()
Shell "start mailto:MrLxg@sina.com" '发送 Email
End Sub
Private Sub Label4_Click()
Shell "start http://mrlxg.myetang.com" '浏览网页
End Sub
Private Sub Label5_Click()
Shell "start mailto:MrLxg@sina.com" '发送 Email
End Sub
```

```
Private Sub Label6_Click()
Shell "start http://mrlxg.myetang.com" '浏览网页
End Sub
Private Sub Label7_Click()
Shell "start http://go5.163.com/~mrlxg" '浏览网页
End Sub
```

三、 “主模块.bas”

“主模块.bas”的作用是API声明，及一些窗体中调用的函数，参看程序中的说明。代码如下：

```
Attribute VB_Name = "主模块"
'这是一个操作注册表的VB程序,其中包含可以建立新键值
'该程序的作用隐藏驱动器盘符
'李学刚制作
Type FILETIME
lLowDateTime As Long
lHighDateTime As Long
End Type
Declare Function RegOpenKeyEx Lib "advapi32.dll" Alias
"RegOpenKeyExA" (ByVal hKey As Long, ByVal lpSubKey
As String, ByVal ulOptions As Long, ByVal samDesired As
Long, phkResult As Long) As Long
Declare Function RegCloseKey Lib "advapi32.dll" (ByVal
hKey As Long) As Long
Declare Function RegCreateKey Lib "advapi32.dll" Alias
"RegCreateKeyA" (ByVal hKey As Long, ByVal lpSubKey As
String, phkResult As Long) As Long
Declare Function RegDeleteKey Lib "advapi32.dll" Alias
"RegDeleteKeyA" (ByVal hKey As Long, ByVal lpSubKey As
String) As Long
Declare Function RegQueryValueEx Lib "advapi32.dll" Alias
"RegQueryValueExA" (ByVal hKey As Long, ByVal lpValue-
Name As String, ByVal lpReserved As Long, lpType As
Long, ByVal lpData As String, lpcbData As Long) As Long
Declare Function RegQueryValueExA Lib "advapi32.dll"
(ByVal hKey As Long, ByVal lpValueName As String, ByVal
lpReserved As Long, lpType As Long, ByRef lpData As
Long, lpcbData As Long) As Long
Declare Function RegSetValueEx Lib "advapi32.dll" Alias
"RegSetValueExA" (ByVal hKey As Long, ByVal lpValue-
Name As String, ByVal Reserved As Long, ByVal dwType As
Long, ByVal lpData As String, ByVal cbData As Long) As
Long
Declare Function RegSetValueExA Lib "advapi32.dll" (ByVal
hKey As Long, ByVal lpValueName As String, ByVal Re-
served As Long, ByVal dwType As Long, ByRef lpData As
Long, ByVal cbData As Long) As Long
Declare Function RegSetValueExB Lib "advapi32.dll" Alias
"RegSetValueExA" (ByVal hKey As Long, ByVal lpValue-
Name As String, ByVal Reserved As Long, ByVal dwType As
Long, ByRef lpData As Byte, ByVal cbData As Long) As
Long
Const ERROR_SUCCESS = 0&
Const ERROR_BADDB = 1009&
Const ERROR_BADKEY = 1010&
Const ERROR_CANTOPEN = 1011&
Const ERROR_CANTREAD = 1012&
Const ERROR_CANTWRITE = 1013&
Const ERROR_OUTOFMEMORY = 14&
Const ERROR_INVALID_PARAMETER = 87&&
Const ERROR_ACCESS_DENIED = 5&
Const ERROR_NO_MORE_ITEMS = 259&
Const ERROR_MORE_DATA = 234&
Const REG_NONE = 0&
Const REG_SZ = 1&
Const REG_EXPAND_SZ = 2&
Const REG_BINARY = 3&
Const REG_DWORD = 4&
Const REG_DWORD_LITTLE_ENDIAN = 4&
Const REG_DWORD_BIG_ENDIAN = 5&
Const REG_LINK = 6&
Const REG_MULTI_SZ = 7&
Const REG_RESOURCE_LIST = 8&
Const REG_FULL_RESOURCE_DESCRIPTOR = 9&
Const REG_RESOURCE_REQUIREMENTS_LIST = 10&
Const KEY_QUERY_VALUE = &H1&
Const KEY_SET_VALUE = &H2&
Const KEY_CREATE_SUB_KEY = &H4&
Const KEY_ENUMERATE_SUB_KEYS = &H8
Const KEY_NOTIFY = &H10&
Const KEY_CREATE_LINK = &H20&
Const READ_CONTROL = &H20000
Const WRITE_DAC = &H40000
Const WRITE_OWNER = &H80000
Const SYNCHRONIZE = &H100000
Const STANDARD_RIGHTS_REQUIRED = &HF0000
Const STANDARD_RIGHTS_READ = READ_CONTROL
Const STANDARD_RIGHTS_WRITE = READ_CONTROL
Const STANDARD_RIGHTS_EXECUTE = READ_CONTROL
Const KEY_READ = STANDARD_RIGHTS_READ Or
KEY_QUERY_VALUE Or KEY_ENUMERATE_SUB_KEYS Or
KEY_NOTIFY
Const KEY_WRITE = STANDARD_RIGHTS_WRITE Or
KEY_SET_VALUE Or KEY_CREATE_SUB_KEY
Const KEY_EXECUTE = KEY_READ
Dim hKey As Long, MainKeyHandle As Long
Dim rtn As Long, lBuffer As Long, sBuffer As String
Dim lBufferSize As Long
Dim lDataSize As Long
Dim ByteArray() As Byte
Const DisplayErrorMsg = False
Global Egg
Function SetBinaryValue(SubKey As String, Entry As String,
Value As String)
'写二进制值到注册表函数
Call ParseKey(SubKey, MainKeyHandle)
If MainKeyHandle Then
rtn = RegOpenKeyEx(MainKeyHandle, SubKey, 0, KEY_WRITE,
hKey) '打开键名
If rtn = ERROR_SUCCESS Then '打开键名成功
lDataSize = Len(Value)
ReDim ByteArray(lDataSize)
For I = 1 To lDataSize
```

```
ByteArray(I) = Asc(Mid$(Value, I, 1))
Next
rtn = RegSetValueExB(hKey, Entry, 0, REG_BINARY, ByteArray(1), lDataSize) ´write the value
If Not rtn = ERROR_SUCCESS Then ´打开键名失败
If DisplayErrorMsg = True Then ´如果用户想显示错误
MsgBox ErrorMsg(rtn) ´显示错误
End If
End If
rtn = RegCloseKey(hKey) ´关闭键名
Else ´
If DisplayErrorMsg = True Then
MsgBox ErrorMsg(rtn)
End If
End If
End If
End Function
Function GetBinaryValue(SubKey As String, Entry As String)
Call ParseKey(SubKey, MainKeyHandle)
If MainKeyHandle Then
rtn = RegOpenKeyEx(MainKeyHandle, SubKey, 0, KEY_READ, hKey)
If rtn = ERROR_SUCCESS Then
lBufferSize = 1
rtn = RegQueryValueEx(hKey, Entry, 0, REG_BINARY, 0, lBufferSize) ´从注册表获取键值
sBuffer = Space(lBufferSize)
rtn = RegQueryValueEx(hKey, Entry, 0, REG_BINARY, sBuffer, lBufferSize) ´从注册表获取键值
If rtn = ERROR_SUCCESS Then ´如果不能返回值
rtn = RegCloseKey(hKey) ´关闭键名
GetBinaryValue = sBuffer
Else
GetBinaryValue = ″Error″
If DisplayErrorMsg = True Then ´用户想显示错误
MsgBox ErrorMsg(rtn)
End If
End If
Else ´如果键名不能打开
GetBinaryValue = ″Error″
If DisplayErrorMsg = True Then
MsgBox ErrorMsg(rtn)
End If
End If
End If
End Function
Function GetMainKeyHandle(MainKeyName As String) As Long
Const HKEY_CLASSES_ROOT = & H80000000
Const HKEY_CURRENT_USER = & H80000001
Const HKEY_LOCAL_MACHINE = & H80000002
Const HKEY_USERS = & H80000003
Const HKEY_PERFORMANCE_DATA = & H80000004
Const HKEY_CURRENT_CONFIG = & H80000005
Const HKEY_DYN_DATA = & H80000006
Select Case MainKeyName
Case ″HKEY_CLASSES_ROOT″
GetMainKeyHandle = HKEY_CLASSES_ROOT
Case ″HKEY_CURRENT_USER″
GetMainKeyHandle = HKEY_CURRENT_USER
Case ″HKEY_LOCAL_MACHINE″
GetMainKeyHandle = HKEY_LOCAL_MACHINE
Case ″HKEY_USERS″
GetMainKeyHandle = HKEY_USERS
Case ″HKEY_PERFORMANCE_DATA″
GetMainKeyHandle = HKEY_PERFORMANCE_DATA
Case ″HKEY_CURRENT_CONFIG″
GetMainKeyHandle = HKEY_CURRENT_CONFIG
Case ″HKEY_DYN_DATA″
GetMainKeyHandle = HKEY_DYN_DATA
End Select
End Function
Function ErrorMsg(lErrorCode As Long) As String
Select Case lErrorCode
Case 1009, 1015
GetErrorMsg = ″注册表遭破坏!″
Case 2, 1010
GetErrorMsg = ″错误键名″
Case 1011
GetErrorMsg = ″不能打开键名″
Case 4, 1012
GetErrorMsg = ″不能读取键名″
Case 5
GetErrorMsg = ″拒绝使用该键名″
Case 1013
GetErrorMsg = ″不能写入键名″
Case 8, 14
GetErrorMsg = ″内存溢出″
Case 87
GetErrorMsg = ″无效参数″
Case 234
GetErrorMsg = ″还有很多数据保留在缓冲区中.″
Case Else
GetErrorMsg = ″未知错误,代码为: ″ & Str$(lErrorCode)
End Select
End Function
Private Sub ParseKey(Keyname As String, Keyhandle As Long)
rtn = InStr(Keyname, ″\″)
If Left(Keyname, 5) <> ″HKEY_″ Or Right(Keyname, 1) = ″\″ Then
MsgBox ″Incorrect Format: ″ + Chr(10) + Chr(10) + Keyname
Exit Sub
ElseIf rtn = 0 Then
Keyhandle = GetMainKeyHandle(Keyname)
Keyname = ″″
Else
Keyhandle = GetMainKeyHandle(Left(Keyname, rtn – 1))
Keyname = Right(Keyname, Len(Keyname) – rtn)
End If
End Sub
```

```
Function CreateKey(SubKey As String)
Call ParseKey(SubKey, MainKeyHandle)
If MainKeyHandle Then
rtn = RegCreateKey(MainKeyHandle, SubKey, hKey)
If rtn = ERROR_SUCCESS Then
rtn = RegCloseKey(hKey)
End If
End If
End Function
Sub main()
窗体 1.Show
While DoEvents() > 0 '等到没有要显示的窗体时退出
Wend
End
End Sub
```

软件调试成功后就可以编译了，编译成功后为了减小文件尺寸，可以采用 UPX 进行压缩，压缩后只有 14K。使用这个小软件时，根据屏幕操作就可以了。软件中隐藏了一个小彩蛋，捕获的方法是在主画面中单击小猪图标，点“关于”，双击左上角的小猪图标即可，彩蛋内容是显示作者的真实姓名，小猪变成小兔子。这个小软件可以复制到其他 Win98 上运行，如复制到其他 Win95 97 不能运行，请把 Win98 hSystem h Msvbvm50.dll 复制到 Win95 97 hSystem 目录中即可。

编译好的运行文件可到 http //mrlxg.myetang.com/soft/rake.exe 或者 http //go5.163.com/~mrlxg/soft/rake.exe 下载。

后记：以上所述隐藏驱动器的做法只是把驱动器不显示在窗口中，实际上并未从硬盘分区表等处作手脚，因此不是真的隐藏，如果要进入隐藏的驱动器如 C 盘，只要打开我的电脑，在地址栏输入 C h回车就行了，当然在 MS-DOS 方式下输入 DIR C 回车也可查看 C 盘的内容。使用本软件隐藏任意一个驱动器，在“开始”上点右键，弹出的“资源管理器”和“打开”将不可用。

（收稿日期：2000 年 12 月 13 日）

上接第 83 页

```
;;;;;;;;;;;;计算插入块的转角 ang 与属性块的插入点(xc, yc)
 (command "explode" (ENTLAST))
 (setq a (entlast))
 (setq b (entget a))
 (setq x1y1 (nth 7 b));;;;;; ACAD2000 版此处改为(setq x1y1 (nth 9 b))
 (setq x1 (cadr x1y1))
 (setq y1 (caddr x1y1))
 (setq x2y2 (nth 8 b));;;;;; ACAD2000 版此处改为(setq x2y2 (nth 10 b))
 (setq x2 (cadr x2y2))
 (setq y2 (caddr x2y2))
 (setq x12 (/ (+ x1 x2) 2.0))
 (setq y12 (/ (+ y1 y2) 2.0))
 (setq x122 (- x2 (/ (- x2 x1) 5)))
 (setq y122 (- y2 (/ (- y2 y1) 5)))
 (setq xc (+ x122 (/ (- x122 x) 1.6)))
 (setq yc (+ y122 (/ (- y122 y) 1.6)))
 (setq ycz (- y2 y1))
 (setq xcz (- x2 x1))
 (command "-osnap" "off")
 (if (= xcz 0.0) (setq ang 90.0) (setq ang (* (atan (/ ycz xcz)) (/ 180.0 pi))))
 (if (< xcz 0.0) (setq ang (+ ang 180.0)))
 (if (and (= xcz 0.0) (< ycz 0.0)) (setq ang 270.0))
;;;;;;根据 ang 的大小决定插入什么属性块"ccdz"还是"ccdf"
(cond ((or (= ang 90.0) (and (< ang 90.0) (> ang -60.0)))
 (command "insert" "ccdz" "s" sc (list xc yc) ang ccd))
((or (= ang 270.0) (and (> ang 120.0) (< ang 270.0)))
(command "insert" "ccdf" "s" sc (list xc yc) (- ang 180.0) ccd))
;;;;;;;当 ang 在两个特殊角度范围内时要用指引线方式标注
(progn
(command "erase" "l" "" "erase" "l" "" "erase" "l" "")
(alert "此处要用引线方式标注!")
(princ "\n 此处要用引线方式标注!请画两段引线:")
(command "line" (list x y) pause "");画第一段指引线
(command "ortho" "on");打开正交模式
(command "line" "" pause "")
(command "ortho" "off")
;;;;;;;;;;计算第二段指引线的中点
(setq a (entlast))
(setq b (entget a))
(setq x1y1 (nth 7 b)) ;;;2000 版改为(setq x1y1 (nth 9 b))
(setq x1 (cadr x1y1))
(setq y1 (caddr x1y1))
(setq x2y2 (nth 8 b));;2000 版改为(setq x2y2 (nth 10 b))
          (setq x2 (cadr x2y2))
          (setq y2 (caddr x2y2))
          (setq x12 (/ (+ x1 x2) 2.0))
          (setq y12 (/ (+ y1 y2) 2.0))
;;;;;;;;;;;;以转角 0 度插入带属性的粗糙度符号块"ccd"
    (command "insert" "ccd" "s" sc (list x12 y12) "0" ccd)
     )
 ))
(setvar "cmdecho" 1)
);;;;;;;;;;;;;;程序结束
```

参考文献

1. 王忠伟. AutoCAD R14 培训教程. 北京：人民交通出版社，1999

2. 梁雪春等. AutoLisp 实用教程. 北京：人民邮电出版社，1998

3. 中国纺织大学工程图学教研室. 画法几何及工程制图. 上海：上海科学技术出版社，1997

（收稿日期：2000 年 12 月 25 日）

电脑硬盘系统使用与维护常见问题解答

黄建龙

怎样从硬盘故障提示信息处理硬盘故障

硬盘因使用率高，维护不当、误操作或受到强烈震动，都可能引起硬盘的系统出故障，一般常见的故障提示信息有以下几种：

（1）Non - system disk or disk error　非系统盘或磁盘错误）

Replace and strike any key when ready　换上系统盘后按任一键）

系统不能从硬盘启动，开机后系统不能从硬盘启动，会显示出上述错误信息。

（2）Invalid drive specification　　无效驱动器定义）

操作系统不能识别硬盘等，把操作系统盘插入 A 驱动器，从软盘启动，想要进入硬盘 C 时，会显示出上述错误信息。

这是硬盘的常见故障，是硬盘的主引导区部分的分区坏了，可使用 Fdisk 建立分区。故障的原因是多种多样的。建议使用和原来相同版本的系统的 Fdisk 最好。支持 FAT32 硬盘格式的 Fdisk 在执行时，首先会要你选择是否使用 FAT32 格式，建议还是选择 FAT16 格式，以使系统更加稳定可靠。

操作步骤是，使用系统软盘从 A 软驱引导系统，键入 Fdisk，选择第一项重新建立分区。

（3）Seek error reading drive C　读 C 盘时定位错）

Abort Retry Ignort

不能定位到指定磁道，读写硬盘经常出现该错误信息。

（4）Invalid media type reading drive C

使用 A 盘启动电脑，当出现 "A：h>" 后，键入 "C：回车"，想进入硬盘，出现 "C：h>" 后，键入 Dir 命令，系统提示该错误信息。

（5）No partition bootable

这是硬盘没有分区的表现。使用系统软盘从 A 软驱重新引导系统。键入 Fdisk，选择第一项，建立 DOS 分区。

（6）No operating system

如果正在重新做硬盘，可能是忘记了硬盘格式化或格式化硬盘时没有带参数 /S　可使用软盘引导系统，并使用 Format C　/S 格式化硬盘。

如果原来有系统，现在发生这样的事，可检查主板 Setup 中的硬盘参数，使用 "Auto IDE Delect" 重新检测硬盘参数。

（7）Non - system disk or disk error

使用软盘引导时可能有磁盘坏的问题，否则多是 Msdos. sys、Io. sys 和 Command. com 不完整。

解决的方法是使用系统盘引导或修复引导盘。操作步骤是：看是否把非系统软盘插入软驱了，如果是，可把该盘取出，使系统从硬盘引导。

如果目前正在从硬盘引导，可使用软盘引导，并检查 Msdos. sys、Io. sys 和 Command. com，如果缺什么文件，使用同版本的系统补上。

（8）Bad or missing command Interpreter

可能是文件被病毒破坏或杀毒后破坏了 Command. com 文件。处理方法是拷贝同版本的 Command. com 文件。把相同版本的 Command. com 文件覆盖引导盘根目录中即可。

怎样以磁盘扇区的格式保存和恢复硬盘主引导扇区的信息

在硬盘启动过程中，常出现因盘主引导信息损坏而导致硬盘无法引导、系统处于瘫痪的现象。可采用以磁盘扇区的格式解决这个问题。

用 DEBUG 将主引导扇区（0 磁头，0 柱面，1 扇区）内容读入内存，把内存中的主引导信息写入软盘的某一扇区内，在软盘上作上标记存档，待硬盘主引导扇区有问题时，把备份存档的主引导信息恢复到硬盘上。

（1）首先用 DEBUG 将其主引导内容读入内存 XXXX：0200 地址处

```
A> DEBUG
-A100
XXXX: 0100 MOV AX, 0201
XXXX: 0103 MOV BX, 0200
XXXX: 0106 MOV CX, 0001
XXXX: 0109 MOV DX, 0080
XXXX: 010C INT 13
XXXX: 010E INT 3
XXXX: 010F
-G = 100
```

（2）把一张格式过的软盘放入 A 驱（或 B 驱），将内存 XXXX：0200 处开始的主引导信息写入 A 驱 50 扇区内（逻辑扇区数）

```
-W 200 0 1001
```

其中：1：写入的扇区数

100：写入的逻辑起始扇区

0：磁盘代号 00：A 盘；01：B 盘；02：C 盘。

200：写入的内存起始地址

（3）把保存有主引导扇区内容的软盘作上标记存放备

用。

4）当发现主引导扇区损坏而无法引导系统时，取出备份盘将其恢复。

①将备份软盘插入 A 驱（或 B 驱），先将备份扇区信息调入内存：0200 处。（用 DEBUG 中的 L 命令）。

```
-L 200 0 100 1
```

②将内存的主引导信息写回到硬盘上

```
A>DEBUG
-A100
XXXX: 0100 MOV AX,
0301
XXXX: 0103 MOV BX,
0200
XXXX: 0106 MOV CX,
0001
XXXX: 0109 MOV DX,
0080
XXXX: 010C INT 13
XXXX: 010E INT 3
XXXX: 010F
-G = 100
```

怎样以文件格式保存和恢复硬盘主引导扇区的信息

可以把硬盘的主引导扇区内容写到一个文件内，待硬盘主引导扇区有误时，将软盘上的文件复原。

（）编写备份硬盘主引导扇区程序 program1. dat

```
A100
MOV AX, 0201
MOV BX, 0200
MOV CX, 0001
MOV DX, 0080
INT 13
INT 3
G = 100
D0200, 03FF
RIP
0100
RCX
0200
RBX
0000
NA: program1. dat
W 200
Q
```

2 编写恢复主引导扇区程序 program2. dat

```
NA: program1. dat
1200
D0200, 03FF
A100
MOV AX, 0301
MOV BX, 0200
MOV CX, 0001
MOV DX, 0080
INT 13
INT 3
RIP
0100
G = 100
Q
```

5）将 program1. dat、program2. dat 和 DEBUG 拷入 A 驱软盘上。

1）保存扇区时执行 A>DEBUG < program1. dat

2）恢复扇区时执行 A>DEBUG < program2. dat

怎样在无备份硬盘主引导扇区情况下恢复主引导扇区信息

既没软盘备份，也没硬盘备份，主引导扇区损坏后，可采用如下方法

先将损坏的主引导扇区用 DEBUG 读入内存，并将分区信息（四个分区）记录或打印出来，用其他机器硬盘上的主引导扇区（同版本）覆盖有误扇区，最后再将覆盖后的扇区中分区信息改写成（用 E 命令）原分区信息，系统便可正常运行。