第2章　类

到目前为止，我们只讨论了 MFC的CWnd类。在本章中，我们将讨论 MFC提供的其他重要的类，这些类可以分成以下几种：

- 访问用户界面的类，包括CWnd。
- 帮助绘图的类。
- 提供运行一个应用程序所需功能的类。
- 处理数组和数据列表的类。
- 访问数据库的类。
- 维护文件的类。
- 允许应用程序在网上或Internet上通信的类。
- 一些用来帮助同步和调试应用程序的混合类。

本章的目标不是成为 MFC参考指南，而是综合论述 MFC所能提供的功能。对于任何在本章中没有讨论的类，或有关某一特定的类的详细描述，请参考有关 MFC文献。

大部分OLE类没有在本章中论及，因为它们不在本书的讨论范围。

2.1　基类

大多数MFC类是从三个基类(Base Class)派生的： CObject、CCmdTarget和CWnd。CCmdTarget派生于CObject类，而CWnd于CCmdTarget类。从CObject派生的类，具有在运行时获得对象大小和名字的能力；从 CCmdTarget派生的类，能够处理命令消息；从 CWnd派生的类，能控制它们自己的窗口。

2.1.1　CObject

CObject类本身提供的功能较少，主要工作由六个伴生宏(　companion macros)完成。CObject和这些宏一起，允许 CObject的派生类在运行时获取类名和对象大小，创建一个类对象而不必知道类名，以及允许从文件设备中存取一个类的实例而不必知道类名。

下面的宏允许类的一个实例在运行时知道它的类名和对象大小：

```
DECLARE_DYNAMIC (CYourClass)          // in the .h file
IMPLEMENT_DYNAMIC (CYourClass, CYourBaseClass) // in the .cpp file
```

可以用CObject::GetRuntimeClass ( )获得运行时使用这些宏的类的详情。

下面这些宏包括了前面两个宏的功能，但是允许在不知道类名的情况下创建一个类的实例。

```
DECLARE_DYNCREATE (CYourClass)     // in the .h file
IMPLEMENT_DYNCREATE (CYourClass, CYourBaseClass)     // in the .cpp file
```

在不知道类名的情况下，可以用 CRuntimeClass::CreateObject ( )创建一个运用这些宏的类的实例。

下面这些宏包括了前面所有宏的功能，但还允许在不知道类名的情况下，把一个类的实例存到磁盘上。

```
DECLARE_SERIAL (CYourClass)              // in the .h file
IMPLEMENT_SERIAL (CYourClass, CYourBaseClass, schema)   // in the .cpp file
```

有关使用最后两个宏的例子，参见例 70。

2.1.2 CCmdTarget

由CCmdTarget派生的类，可以接收和处理来自应用程序菜单或工具栏的命令消息。有关CCmdTarget类的详细的讨论将在第3章中进行。

2.1.3 CWnd

这在第1章中已讨论过， CWnd的成员函数封装了 Windows API中负责维护和创建窗口的函数。CWnd是从CCmdTarget派生的，因此，也能接收和处理命令消息。

注意 在本章中，用下面的字母标明MFC类是从前述的基类中哪一个派生的。

- O表示该类是从CObject派生的。
- O/C表示该类是从CObject和CCmdTarget派生的。
- O/C/W表示该类是从CObject、CCmdTarget和CWnd派生的。

2.2 应用程序、框架、文档和视图类

使用Developer Studio的AppWizard(应用程序向导)创建MFC应用程序时，应用程序从四个基类派生：

CWinApp 是应用程序的“ 应用程序类 ”，负责初始化和运行应用程序。

CFrameWnd 是应用程序的“ 框架类 ”，负责显示和搜寻用户命令。

CDocument 是应用程序的“ 文档类 ”，负责装载和维护文档。文档可以是从文档到网络设备设置的任何内容。

Cview 是应用程序的“ 视图类 ”，负责为文档提供一个或多个视图。

注意 在本书中，我们将用应用程序类、框架类等术语表示从这四个基类派生的类。

根据所建的应用程序类型，应用程序中 AppWizard包含相应的基类。

对话框应用程序 用户界面只有一个对话框，没有框架类、文档类或视图类。对话框应用程序只用应用程序类的派生类 —— CWinApp。对话框用MFC的CDialog类创建，这将在随后部分讨论。

单文档界面应用程序 能在某个时间内装入和编辑一个文档，使用了前面提及的所有四个基类。

多文档界面应用程序 一次可以装载和编辑多个文档，并且不但使用了所有四个基类，还使用了CFrameWnd的两个派生类，即CMDIFrameWnd和CMDIChildWnd。

CDocument和CView类的派生类负责第1章中描述的文档 /视图。.使用AppWizard的一个例子，参见例2。

2.2.1 CWinApp(O/C/W)

应用程序类CWinAPP(O/C/W)是应用程序开始后创建的第一个对象，并且是在结束前最后一个执行的对象。起动时，应用程序类负责创建应用程序的其余对象。

- 对于一个对话框应用程序，应用程序类应用 CDialog创建一个对话框。
- 对于一个SDI应用程序，应用程序类创建一个或多个文档模板 (见下面的讨论)，然后用模板打开一个空文档。
- 对于一个MDI应用程序，在主框架类中，应用程序类创建一个或多个文档模板，然后用模板打开一个空文档。

在下一章中还将看到，应用程序类也同操作系统交互，作为中继，将鼠标信息和键盘输入传给应用程序的其余部分。

应用程序类是由CWinApp派生的，并且， AppWizard以CXxxApp形式进行命名，这里的Xxx是应用程序的名字。

1. 文档模板

应用程序打开一个文档时，文档模板定义创建什么样的框架、文档和视图。要创建一个文档模板，要么为 SDI应用程序创建一个 CSingleDocTemplate类，要么为MDI应用程序创建CMultiDocTemplate类，并且用三个类的指针对它进行初始化。

```
pDocTemplate=newCMultiDocTemplate (
IDR_APPTYPE,
RUNTIME_CLASS (CAppDoc),            // Your Document Class
RUNTIME_CLASS (CChildFrame),        // Your Frame Class
RUNTIME_CLASS (CAppView));          // Your View Class
```

这里的RUNTIME_CLASS宏返回一个指向类的 CRuntimeClass结构的指针，该结构用DECLARE_DYNCREATE和IMPLEMENT_DYNCREATE宏填加到类中。文档模板通过用CRuntimeClass::CreateObject ( )函数创建一个所有三个类的实例来打开一个文档。

2. 线程

CWinApp类自身是从CWinThread派生的。CWinThread类封装了系统中用来创建和维护应用程序线程的Windows API函数。实际上，可以通过创建 CWinThread类的另一个实例实现应用程序多任务。在应用程序中， CWinApp类代表主要的执行线程。

3. CFrameWnd(O/C/W)

此框架类CFrameWnd(O/C/W)是应用程序运行时创建的第二个对象，负责显示和监督用户对应用程序其余部分的命令。

对于一个 SDI应用程序，框架类是从 CFrameWnd派生的， AppWizard自动命名为：CMainFrame。

对于一个 MDI应用程序，框架类派生于 CMDIFrameWnd，AppWizard自动命名为CMainFrame。在MDI应用程序中，每个打开的文档有一个子框架类，每个子框架类是从CMDIChildWnd派生的。AppWizard自动指定子框架类名为CChildFrm。

对于一个对话框应用程序来说，没有框架类。在前面已提过，一个对话框应用程序是由一个应用程序类和一个对话框类组成的。

4. CDocument (O/C)

通常，文档类CDocument(O/C)是应用程序打开一个新文档或一个已存在的文档时创建的

对象。文档类负责将一个文档赋给它的成员变量，并允许视图类编辑这些成员变量。一个文档包括从图形文件到可编程控制的任何内容。文档类派生于 CDocumnt，AppWizard自动命名为CXxxDoc，Xxx是应用程序名。

2.2.2 CView (O/C/W)

在创建文档类的一个实例之后将创建视图类 CWiew(O/C/W)的一个实例。视图类负责描述文档类的内容，它也可以允许用户编辑文档。

窗口分区类，即CSplitterWnd，允许文档有多视图；这些视图以由相同视图类的一些实例创建，也可以由完全不同的视图类实例创建。

AppWizard允许视图类从几个基类 (包括CTreeView、CEditView、CRichEditView、CListView等)之一派生，每一个基类赋予应用程序不同的功能集。有关这些类的更多例子参见例1。所有这些类都派生于 CView类，不管选择哪个基类， AppWizard自动命名派生类为CXxxView，这里的Xxx是应用程序名。

前面已提及，可以从四个基类创建三种类型的应用程序：对话框、单文档、多文档。下面我们将详细讨论这些应用程序类型。

1. 对话框应用程序

对话框应用程序是由一个派生于 CWinApp的应用程序类，和用派生于 CDialog类创建的对话框构成的(见图2-1)。

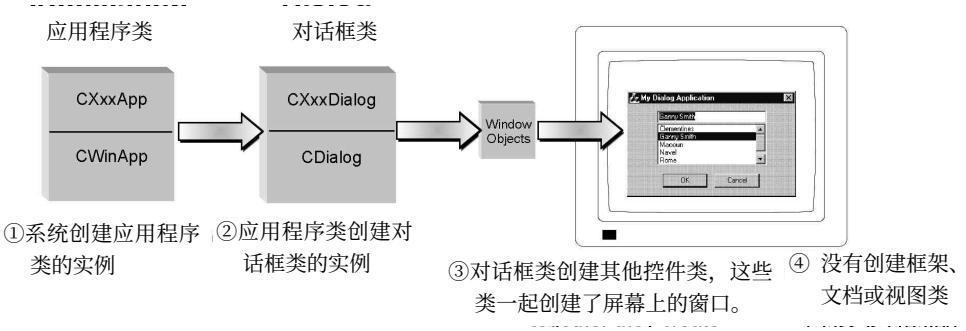

图2-1 一个对话框应用程序由一个应用程序类和一个对话框类创建

2. 单文档界面应用程序

单文档界面应用程序包括以下类：从 CWinApp派生的应用程序类、从CFrameWnd派生的框架类、从CDocument派生的文档类，以及每个文档一个或多个视图类，这些视图类是从一些派生于CView的视图类派生的(见图2-2)。

3. 多文档界面应用程序

多文档界面 (MDI)应用程序包括以下类：派生于 CWinApp的应用程序类、派生于CMDIFrameWnd的框架类、派生于 CMDIChildWnd的一个或多个子框架类、派生于CDocument的每个子框架一个文档类和派生于CView的每个文档一个或多个视图类(见图2-3)。

4. 应该选择哪一种应用程序类型

对话框应用程序一般在用户交互要求有限的情况下使用。然而就个人而言，宁愿使用一个带窗体视图(CFormView)的SDI应用程序，而不愿意使用一个对话框应用程序，这样既利用了创建对话框应用程序的简单性，又具备了单文档应用程序的文档特性。

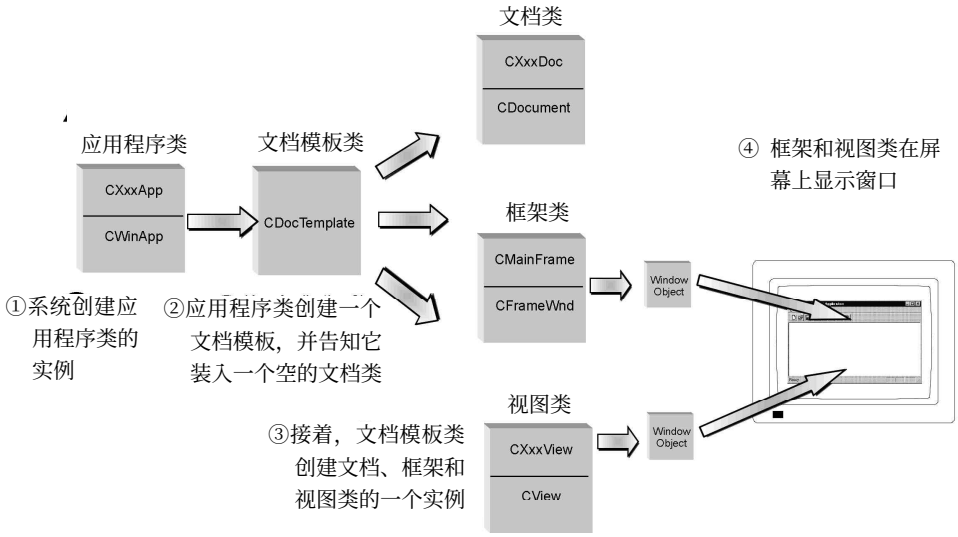

图2-2 一个SDI应用程序由应用程序、框架、文档和视图类创建

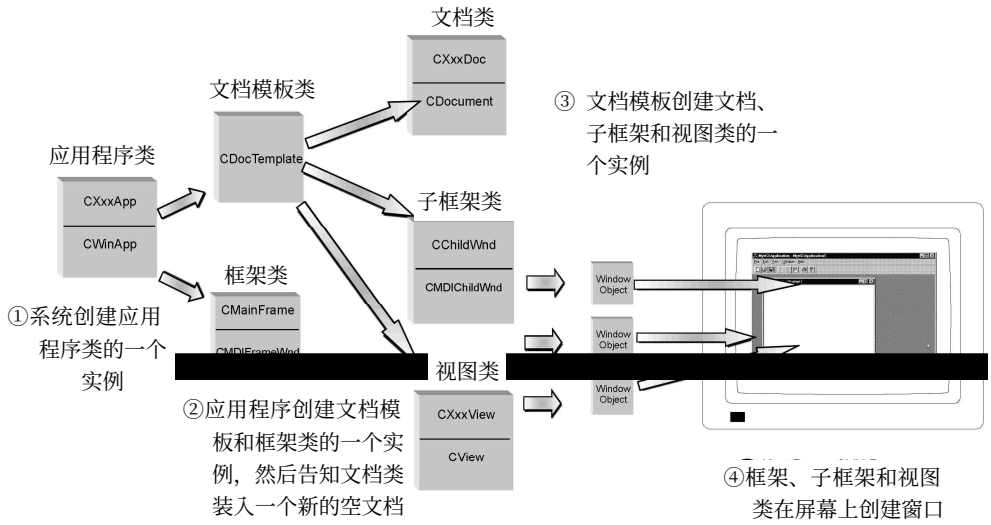

图2-3 一个MDI应用程序由应用程序、框架、子框架、文档和视图类创建

什么时候应该使用MDI应用程序，而不是SDI应用程序呢？如果在运行期间应用程序只与一个文档有关连，那么就使用SDI方法。但是，如果应用程序能产生多个文档，并对多个文档进行处理(即使这不是一次性完成的)，这时应该采取MDI的方法，即使在开始时并没有看到在同一时间修改多个文档的可能。一个MDI应用程序并不比一个SDI应用程序复杂多少，却可以带给用户在同一时间看阅多个文档的方便。

2.3 其他用户界面类

除了框架和视图类外，MFC还提供了许多其他支持用户界面的类。

控件窗口类 封装了通用控件。

菜单类 处理菜单，就如CWnd类处理窗口一样。

对话框类 封装了对话框和通用对话(Common Dialog)框。

控制条类 封装了控制条(工具栏、对话条和状态栏)。

属性类 封装了属性表(Property Sheet)和属性页(Property Page)。

2.3.1 通用控件类

通用控件类(O/C/W)封装了通用控件(如按钮、列表框等)的功能。这些类派生于CWnd,继承了诸如ShowWindow( )和MoveWindow( )之类的窗口成员函数。当这些类创建窗口时,它们使用一个通用控件窗口类。例如,用 CButton通用控件类创建一个按钮时,它就用BUTTON窗口类创建真实的窗口:

```
Create (_T ("BUTTON"), lpszCaption, dwStyle, rect, pParentWnd, nID);
```

下面表中列出了通用控件类、它们创建的控件,以及它们使用的窗口类。

通用控件MFC类和它们的窗口类

MFC类	通用控件	窗口类
CAnimateCtrl (O/C/W)	动画控件	SysAnimate32
CButton(O/C/W)	按钮控件	BUTTON
CComboBox(O/C/W)	组合框控件	COMBOBOX
CEdit(O/C/W)	编辑控件	EDIT
CHeaderCtrl(O/C/W)	标题控件	SysHeader32
CListBox(O/C/W)	列表框控件	LISTBOX
CListCtrl(O/C/W)	列表控件	SysListView32
CProgressCtrl(O/C/W)	进度指示控件	msctls_progress32
CScrollBar(O/C/W)	滚动条控件	SCROLLBAR
CSliderCtrl(O/C/W)	滑块控件	msctls_trackbar32
CSpinButtonCtrl(O/C/W)	微调按钮控件	msctls_updown32
CStatic(O/C/W)	静态文本控件	STATIC
CTreeCtrl(O/C/W)	树型控件	SysTreeView32
CTabCtrl(O/C/W)	标签控件	SysTabControl32
CDateTimeCtrl(O/C/W)	日期/时间控件	SysDateTimePick32
CMonthCalCtrl(O/C/W)	月历控件	SysMonthCal32
CHotKeyCtrl(O/C/W)	热键控件	msctls_hotkey32
CToolTipCtrl(O/C/W)	工具提示控件	tooltips_class32

不是所有的通用控件类都简单地封装一个通用控件窗口类。事实上,有三个 MFC类提供了通用控件中没有的功能。下面的表格列出了这些类、它们派生的 MFC类和它们所提供的额外功能。

派生的MFC类和它们的MFC基类

MFC类	所派生的MFC类	增加的功能
CBitmapButton	CButton	更好地支持按钮上的位图
CCheckListBox	CListBox	列表框中的复选框
CDragListBox	CListBox	列表框中用户可拖拉的工程

2.3.2 菜单类

菜单类封装了 Windows API中用来创建和维护菜单的函数。 CMenu还有两个成员函数

Attach ( )和Detach ( )，允许封装一个已建的菜单，如同 CWnd对象能够封装一个已有的窗口一样。

2.3.3 对话框类

CDialog类封装了用来创建对话框的 Windows API。对话框是弹出式窗口，在创建时可以用对话框模板中定义的控件窗口来填充。

通用对话框MFC类

MFC还有六个类封装了 Windows API用来创建通用对话框的函数。通用对话框是一些随控件预先装入的用于提示用户输入一些常用信息的对话框，如装入或存储的文件名、颜色、字体和打印参数。通用对话框不但为用户提供了一个熟悉的对话框，而且节省了书写这些对话的工作量。

下面的表格显示了通用对话框的目的、提供它们的 Windows APIs和封装这些 Windows APIs的MFC通用对话框类。

通用对话框MFC类及其用途：

通用对话框	Windows API调用	MFC类
选择颜色	::ChooseColor()	CColorDialog
打开/保存文件	::GetOpenFileName() ::GetSaveFileName()	CFileDialog
查找或替换文本	::FindText() ::ReplaceText()	CFindReplaceDialog
选择字体	::ChooseFont()	CFontDialog
设置打印页	::PageSetupDlg()	CPageSetupDialog
打印	::PrintDlg()	CPrintDialog

2.3.4 控制条类

控制条类(O/C/W)封装了为应用程序提供工具栏、状态栏、对话条和 rebar 的Windows API函数。

- CToolBar (O/C/W)和CToolBarCtrl (O/C/W)类帮助创建和维护一个工具栏。CToolBarCtrl类封装了系统提供的窗口类 ToolbarWindow32的功能，并由该类来做实际的工具栏的创建和维护工作。而 CToolBar类就是通常的 MFC。CToolBar能提供比CToolBarCtrl更多的功能，但必须以开销更多的内存为代价。
- CStatusBar (O/C/W)和CStatusBarCtrl (O/C/W)类创建和维护状态栏。CStatusBarCtrl类封装了系统提供的窗口类msctls-statusbar32，并由该类来做实际的状态栏的创建和维护工作。而CStatusBar类就是通常的MFC。CStatusBar能提供比CStatusBarCtrl更多的功能，但必须以开销更多的内存为代价来支持它。
- CDialogBar (O/C/W)类创建和维护对话条。一个对话条是对话框和工具栏的混和产物，它显示一个对话框模板，但却象一个工具栏一样浮动并停靠在主框架窗口中。
- CRebar (O/C/W)和CRebarCtrl (O/C/W)类创建和维护Rebar。Rebar是窗口控件，可以包含一个工具栏、一个对话条和任何其他子窗口。它们提供了一个移动条 (grabber bar)(左边双线)，使它们更易于被移动，也更易于在他们的背景上画一幅位图。 CReBarCtrl类

封装了由系统提供的Window类ReBarWindow32的功能，并由该类做实际的 ReBar创建和维护工作。换个角度说， CReBar类就是通常的MFC。CReBar提供比CReBarCtrl更多的功能，但以开销更多的内存为代价。

2.3.5 属性类

属性类封装了Windows API中用来为应用程序提供属性页和属性表的函数。一个或多个属性页出现在属性表中，以创建Windows用户熟悉的tabbed视图。该视图通常用来选择程序选项。

- CPropertySheet(O/C/W)类创建一个属性表，尽管CPropertySheet不是从CDialog派生的，但它们非常相似。
- CPropertyPage(O/C/W/CDialog)类创建一个属性类，该类是从CDialog派生的。

关于属性表和属性页的例子，参见例 8。

2.4 绘图类

2.4.1 设备环境类

CDC(0)类封装了Windows API中用来画图的函数。该类也维护“设备环境（ Device Context）”，设备环境是内存中的一个对象，包含用来绘制的设备的所有特征。这些特征包括设备的最大尺寸和发生作用的方式，设备可以是屏幕或打印机。有关设备环境和 CDC类的成员函数的更详细的讨论，请参看第 4章。

有四个类是从CDC类派生的，并提供了额外功能。

CClientDC 用来方便地创建和破坏一个设备环境， CClientDC通常在堆栈中创建。它的构造函数通过调用 CDC::GetDC ( )为窗口的客户区创建一个设备环境。当例程返回时，CClientDC的析构函数通过调用CDC:ReleaseDC ( )销毁该设备。既不麻烦也不复杂，更不会因忘记释放设备环境而导致资源泄漏。

CWindowDC 类维护窗口的非客户区，如CClientDC维护客户区一样。

CPaintDC 在被构造以获得设备环境时调用CWnd::BeginPaint ( )。在这种情况下，设备环境只允许在已被无效化的窗口客户区绘图，而不能在整个客户区绘图。析构时， CPaintDC类调用CWnd::EndPaint ( )。

CMetaFileDC 创建一个Microsoft元文件，元文件是一个磁盘文件，它包含绘制一幅图形时所必需的全部绘图行为和模型。可以通过打开一个元文件设备环境创建一个元文件，然后用画图工具对它进行绘制，就当它是一个屏幕或打印机设备一样。产生的文件可以在未来的某个时刻被再次读进，以创建其他设备之一的图像。想更多地了解元文件，参见第 4章。

2.4.2 图形对象类

设备环境不足以包含绘图功能所需的所有绘图特征，除了设备环境外， Windows还有其他一些图形对象用来储存绘图特征。这些附加的功能包括从画线的宽度和颜色到画文本时所用的字体。图形对象类封装了所有六个图形对象。

下面的表格列出了MFC类，它封装的图形对象，以及存储在对象中的绘图特征。

图形对象类和它们封装的句柄

MFC 类	图形对象句柄	图形对象目的
CBitmap	HBITMAP	内存中的位图
CBrush	HBRUSH	画刷特性 —— 填充某个图形时所使用的颜色和模式
CFont	HFONT	字体特性 —— 写文本时所使用的字体
CPalette	HPALETTE	调色板颜色
CPen	HPEN	画笔特性 —— 画轮廓时所使用的线的粗细
CRgn	HRGN	区域特性 —— 包括定义它的点

2.5 文件类

CFile ( )类封装了创建和维护平面文件的 Windows API，三个从CFile派生的MFC类提供附加功能：

CMemFile　允许不在磁盘而在内存里创建文件。当构造一个 CMemClass对象时，文件即被打开，并能用它的成员函数对它进行读写，就象一个磁盘文件一样。有关内存文件的例子参见例65。

CSharedFile　与CMemFile类似，但CSharedFile配置在全局堆上，这使得能用剪贴板和DDE对它进行共享。有关共享内存文件的例子参见例 76。

CStdioFile　允许对以回车控制符和换行符结束的文本串进行读写。有关使用 CStdioFile 参见例64。

还有一个感兴趣的文件类是 CFileFind (0)，用它来定位磁盘上的本地文件。 CFileFind类是另外两个类， CFtpFileFind和CGopherFileFind类的基类，这两个类用来帮助在 Internet上定位文件。这两个类将在本章后面讨论。

CArchive和串行化

在一个称为串行化的过程中， CArchive类用CFile类将文档的类对象保存到磁盘上。应用“ 串行化 ”，可以将类的成员变量和全部类对象依次保存到一个存档设备上，以便它们能以完全相同的次序被恢复。有关串行化的一些例子参见第 13章。

2.6 数据库类

MFC库具有支持两种类型数据库的类：

开放数据库连接(ODBC)　封装了大多数数据库厂商支持的 ODBC API。如果应用程序使用了MFC的ODBC类，它就可以支持任何支持ODBC标准的数据库管理系统(DBMS)。

数据访问对象(DAO)　支持一个更新的数据库 API，这已被Microsoft Jet数据库引擎有效利用。也可以通过Jet引擎访问符合ODBC的数据库系统和其他数据源。

2.6.1 ODBC类

三个主要的ODBC类：

CDatabase (0)　用ODBC API打开一个DBMS数据库。构造一个CDatabase对象后，可以用它的OpenEx ( )成员函数创建与数据库的连接。调用 CDatabase成员函数Close ( )结束连接。

CRecordset　通过数据连接来保存和检索记录。

CDBVariant 代表一个记录集中的一列，而不必考虑数据类型。
想更多了解访问ODBC数据，参见例72。

2.6.2 DAO类

DAO类包括与ODBC类相似的三个类：
CDaoDatabase (0) 打开一个DAO数据库。
CDaoRecordSet (0) 容纳记录。
COleVariant 代表记录列。
DAO还包括另外三个类：
CDaoWorkSpace (0) 管理数据库会话，允许事务被处理(存储到数据库)或取消操作(恢复)。
CDaoQueryDef (0) 代表一个查询定义。
CDaoTableDef (0) 代表一个包括域和索引结构的表定义。
想更多了解访问一个DAO数据库，参见例73。

2.7 数据集类

数据集类维护和支持数组、列表和数据对象映像。

CArray 该类及其派生类支持数据对象数组。一个数组由一个或多个相同的数据对象组成(如整数、类等)，它们在内存中相邻，可以通过简单索引访问它们。 CArray类可以动态增加或减少数组大小。有一些派生 CArray的类(如CByteArray、 CDWordArray等)，允许创建一个类型安全(type-safe)的数组。然而，还有一个 CArray <type, arg_type>模板类允许使任何数组类型安全。

CList 该类及其派生类支持数据对象的链接表。一个链接表由一个或多个相同的数据对象组成(如整型、类等)，它们在内存中不连续，但数据对象是双向连接的，以便于通过链表前后搜索数据。有些 CList的派生类(如CPtrList、CObList等)，允许创建一个类型安全的列表。然而，还有一个Clist <type, arg_type>模板类使CList对任何类型都是类型安全的。

CMap 该类及其派生类支持数据对象的字典。在一个二进制或文本关键字约束下，数据字典存储一个或多个相同数据对象 (如整型、类等)，可以用该关键字检索数据项。举例来说，因为Windows不跟踪哪个MFC CWnd对象属于哪个窗口，所以应用程序使用 CMap对象将窗口句柄与它们的CWnd对象关联。有些CMap的派生类(如CMapWordToPtr、CObToString等)，允许创建一个类型安全的字典。不过，还有一个 CMap <class Key, Class ARG_KEY, Class VALUE, Class ARG_VALUE>模板类，允许创建任何类型安全的映像。

有关上述每个类的例子，参见例 78、例79和例80。

2.8 其他数据类

MFC库还包括其他一些数据类：

- CTime和CTimeSpan类具有这样的成员函数，它们既可以读入，又可以格式化 time_t数据类型的时间串(见例81)。
- COleDateTime和COleDateSpan类具有这样的成员函数，它们既可以读入，又可以格式

化OLE DATE数据类型的时间和日期串(见例81)。

CTime或COleDateTime

应该使用哪一个，CTime还是COleDateTime？这取决于想要达到的流行程度。CTime支持的time_t数据类型易于存储和传输，因为它仅仅是个32位整型数。然而，CTime却没有COleDateTime所具有的那么好的函数，如“获得一年中的某一天”。当然time_t数据类型在21世纪将有Y2K问题(2000年问题)，因为它表示自1/1/70所经历的秒数。

- ColeCurrency类具有读入和格式化OLE CURRENCY数据类型的成员函数。
- CString类具有操纵文本串的成员函数，CString操纵的数据类型为TCHAR。
- CPoint类提供了操纵POINT结构的成员函数，该结构定义了一个点的X和Y坐标。
- CSize类提供了操纵SIZE结构的成员函数，该结构定义了尺寸的宽和高。
- CRect类提供了操纵RECT结构的成员函数，该结构定义了矩形的四个点。

2.9 通信类

MFC库包含了一些允许应用程序与网络或因特网(Internet)进行通信的类。

1. 网络类(Network Classes)

CASyncSocket (0)允许应用程序通过网络与其他应用程序进行通信。CASyncSocket封装了Window Socket API, Window Socket API是基于伯克利的California大学的BSD(伯克利软件发行中心)的UNIX Socket实现。CASyncSocket能存取网络通信中的位和字节。

CSocket是从CASyncSocket派生的类，它允许在网络上发送和接收文件而不必考虑细节问题。可以创建一个CSocketFile对象，并将它与CSocket关联；然后创建一个CArchive对象，并与CSocketFile对象关联。此时发送一个文件只是在网络上将它串行化的问题。

2. 因特网类

MFC有两组因特网类(Internet Classes)：一组允许应用程序成为一个客户(如Web浏览器)，而另一组用来增强Internet HTTP服务器软件功能(如Web站点)。

3. 客户类

MFC的因特网客户类(Client Classes)封装了Win32 Internet Extensions (WinInet)，允许用C++语言利用四种协议之一对因特网进行访问，这四种协议为：文件传输协议(FTP)，超文本传输协议(HTTP)、gopher和文件：

CInternetSession (0)类有两种方式与Web站点上的文件相互作用，对于简单的阅读或下载文件，可以用OpenURL ( )成员函数，并将Web站点上文件的URL(通用资源定位器)传给它。OpenURL ( )返回一个指向另一个MFC因特网类的指针，该类取决于用户所使用的URL。

URL前缀	返回的因特网类的指针	URL前缀	返回的因特网类的指针
File://	CStdioFile*	Gopher://	CGopherFile*
http://	CHttpFile*	ftp://	CInternetFile*

上面的每个因特网类都派生于CStdioFile，用户可以读这些文件对象，但不能写。

对一个文件进行写操作或执行一些其他特定协议的操作，必须先用CInternetSession的另一个成员函数打开一个连接。该连接仍然在另一个MFC因特网类中返回。

协　议	调用的CinternetSession成员函数	返回的因特网类指针
FTP	GetFtpConnection()	CFtpConnection*
HTTP	GetHttpConnection()	CHttpConnection*
Gopher	GetGopherConnection()	CGopherConnection*

然后，可以用CFtpConnection、CHttpConnection或CGopherConnection的成员函数与Web站点相互作用。

另外三个相关的MFC 因特网客户类是CFtpFileFind, CGopherFileFind和CGopherLocator。CFtpFileFind和CGopherFileFind派生于前面提及的CFindFile类，可以在Internet上定位文件。CGopherLocator从gopher服务器获得一个gopher“定位器”，并使定位器与CGopherFileFind联系。

4. Internet服务器扩展类

MFC有些类允许在因特网服务器中添加功能，然而，要对这些类进行工作，必须有一个ISAPI —— 兼容的HTTP服务器，而Microsoft没有用MFC对它提供支持，一个差不多与它兼容的服务器是Microsoft的因特网信息服务器。

- CHttpServer类可以被服务器创建，处理来自客户浏览器的特定请求。
- CHttpFilter类可以用来寻找并作用于那些可以通过服务器指定的通信。

2.10 其他类

1. 调试类

CMemoryState类能帮助你正确指出内存泄漏。通过创建一个 CMemoryState对象，并调用CheckPoint ( )成员函数，可获得内存的瞬态图；在稍后时刻，用另外一个 CMemoryState对象获得另外一个瞬态图，并比较结果。在应用程序的调试版， DEBUG_NEW宏利用这些类自动定位内存泄漏。

CException (0)类是MFC处理C++意外事故的设备。有11个派生于CException的类，用来处理列在下面表格中的意外事故：

MFC类	意 外 事 故
CArchiveException	使用CArchive类的错误
CDaoException	使用CDaoDatabase类的错误
CDBException	使用CDatabase类的错误
CFileException	使用CFile类的错误
CInternetException	任何Internet类的错误
CMemoryException	任何内存错误
CNotSupportedException	试图使用一个不支持的函数
CResourceException	试图定位或配置一种资源
CUserException	通常用来提示用户发生了某种例外

处理意外可以使用下面模板：

```
TRY
{
   // attempt something risky
}
```

```
CATCH(CUserException, e)
{
   //handle exception
}
AND_CATCH(CMemoryException, e)
{
   //handle another exception
}
CATCH_ALL(e)
{
   //handle all exceptions
}
END_CATCH
```

2. 同步类

在多线程应用程序中， MFC同步类用来防止数据对象被破坏。前面已提及，一个 MFC应用程序可以同时运行多个线程。如果不止一个这样的线程同时修改相同的数据对象且同时把该数据保存到相同内存地址时，便有可能破坏该数据。

在一个多线程应用程序中，四个MFC类和两个MFC辅助类帮助同步数据存取。

CMutex 用来防止多个线程同时访问同一数据对象。要启用 CMutex，先要把它添加到数据类的成员变量；接着构造另一个 MFC类，即CSingleLock，对任何访问这些成员变量的成员函数引用CMutex；然后调用CSingleLock的Lock (int timeout)成员函数。如果别的线程已在访问该数据，则Lock ( ) 函数不返回，直到该线程调用 Unlock ( )或超时时才返回。CMultiLock类允许指定多个CMutex对象，以便能同时服务多个访问 (见图2-4)。

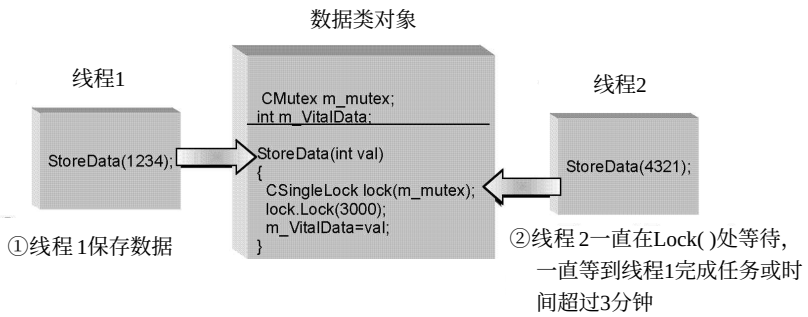

图2-4 用CMutex和CSingleLock同步数据访问

CSemaphore 与CMutex类基本相同，但是，前者允许同时存在一个或多个同时发生的访问。CSingleLock和CMultiLock用法如前。

CCriticalSection 与CMutex类基本相同，但是，前者在堆栈中被构造，并有比 CMutex更快的速度。

CEvent 基于任何事件(不仅仅是被另一线程访问)对数据访问进行同步。线程将一直等待，直到调用CEvent的SetEvent ( )和ResetEvent ( )成员函数允许它们继续为止。

3. 沙漏光标类

CWaitCursor 可以方便地将鼠标变为一个沙漏标来指示系统忙。 CWaitCursor通常在一个费时任务之前在堆栈中构造，在该任务期间禁止输入。然后，当函数返回时， CWaitCursor

自动析构，而沙漏消失(见例34)。

2.11 小结

在本章中，我们已经了解了一些MFC类库必须为应用程序提供的类，这些类包括：

- 用以创建应用程序的类。
- 创建用户界面的类。
- 帮助应用程序绘图的类。
- 处理数据集合、数据库和二维文件的类。
- 允许在网络或因特网上通信的类。

我们也简单涉及到类是怎样应用消息和 CCmdTarget类进行互相通信的，在下一章中将更进一步来探究该主题。