第一部分

Visual C++简介

第一章　Visual C++环境

使用 Visual C++环境来创建 Windows 应用程序的结果是使用 Microsoft C 编译器早期版本快得多的 Windows 应用程序开发周期。用 Visual C++ 2.0，程序员可以通过 Visual Workbench 的使用快速且简易地开发支持 32 位 Windows，MFC3.0，OLE2 及 ODBC 的应用程序。

Visual C++ 2.0 并不意味着完全代替 Visual C++1.5。相反，直到 Windows 3.x 的使用已经下降到不再灵活地开发或保持 16 位 Windows 应用程序时，Visual C++ 1.5 将仍继续作为大部分 Windows 程序员 16 位 Windows 开发所选择的平台。图 1.1 显示了 Visual C++多种版本之间的关系以及它们的关键组成部分。

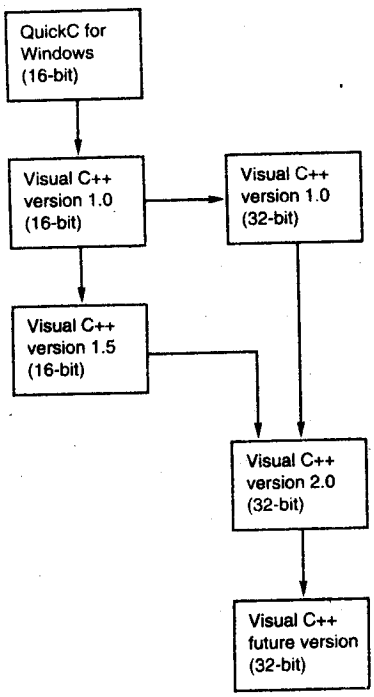

图 1.1　Visual C++的发展

图 1.1 显示了 Visual C++系列中的第一个成员是叫作 Quick C for Windows 的产品。把 Quich C for Windows 包括在此图中，是因为一些 Visual C++的特征(包括 Windows 开发环境和创建 Shell Windows 应用程序的能力)首先在 Quick C for Windows 中介绍。Quick 不提供对 C++的支持。

作为一种产品，QuickC for Windows 并没有太长的生命周期。很快就被 Visual C++1.0的个人版本所代替。当下一个版本被发行时，Visual C++产品的两个版本(个有版本和

专家版本)的创建下降。那时,Microsoft 对于 Visual C++产品的定价很高以至于几乎对这种低成本。系列化的 Visual C++版本没有什么需要和要求。

1.1 VISUAL C++各种版本之间的区别

Visual C++始终保持两个版本:16 位版本和 32 位版本。这两种产品的开发界面不一样,特别是 Visual C++ 32 位版本始终比 16 位版本提供了更好地程序员支持。

每个 Visual C++版本为 MFC 库提供支持。MFC 已经发展经历了三个主要版本(1.0,2.0 和 2.5)。对于 OLE 的支持也经走过了几个版本(OLE1 然后是 OLE2,下面将是 OLE 3.0版本)。Microsoft 已经决定去掉 OLE 名称的数字部分只称作 OLE 产品。例如,这意味着将不会出现 OLE3。

使用 MFC 的程序员将会喜欢一种没有痛苦的升级方法。Microsfot 非常小心不要通过改变或者从 MFC 库中去掉一些特征而明显地影响 MFC 用户。仅有的一些改变是那些改正早期 MFC 版本中发的错误或缺陷所必需要做的东西。

Visual C++ 2.0 为资源编辑和源代码编辑器提供了一个紧密的结合。早期的 Visual C++版本,使用 AppStudio 来编辑资源。App Studio 的名称是来源于 Visual C++的一个独立工具。在 Visual C++ 2.0 中,资源在 Visual C++的开发环境中直接编辑。这意味着在 Visual C++和 App Studio 之间没有切换(及不得不记住的保存文件)。图 1.2 显示了已装入了 Clock 工程文件且对话框已打开的 Visual C++。

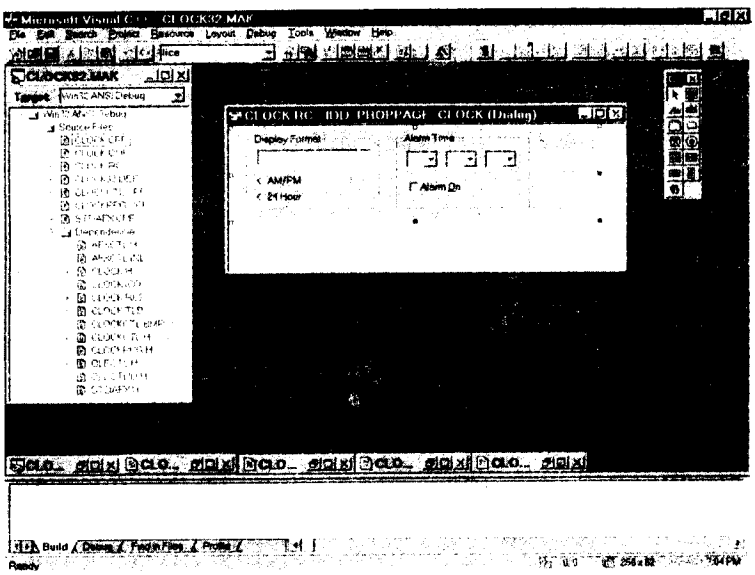

图 1.2 有对话框资源打开 Visual C++ 2.0

用户可以使用可从 Microsoft 中获得的附加 RISC 编译器,为 Windows NT 所支持的平台开发应用程序,包括 RISC(精减指令集计算机)平台。

对于创建不得不移植到不同的非 Windows 平台的应用程序开发人员,Visual C++

2.0支持对于来自于 Apple 计算机公司的计算机 Macintosh 的交叉平台开发。使用 Visual C++ 2.0,Windows 应用程序可以容易地移植到 68XXX 平台。

在过去,当程序员开发一个新的 Windows 应用程序时,要花几个月的时间来完成包容器(Shell)。过去要花几个月的时间和精力所作的事情现在只有需在几分钟内用一些键击来完成。虽然没有确定使用 AppWizard 开发 Windows 应用程序要花多长时间,但是时间不会长。本人对创建第一个 Windows 程序基本的基本包容器进行了计时——整四个月时间。为了开发第一个应用程序,使用了一些相当简陋的工具:一个简单的对话编辑器(可用来给控件定位和确定大小,除此外无其他功能),一个简单的图标编辑器和一个基于 DOS 编辑器命令行。直到 Windows3.x 的发行,不可能真正地在运行 Windows 的同时来开发 Windows 程序——没有足够的内存让编译器和链接器来运行。最好的开发系统是 Windows 和 OS/2 机器的网,其中编辑和程序的建立在 OS/2 机器上进行而实际测试在运行 Windows 的其他系统上进行。

现在,情况要好得多。使用 Windows NT 和 Widnows 95 的保护模式,同 Windows 3.0 之前的版本相比,开发工具可以访问更多的内存。另外,Windows 应用程序(包括 Visual C++的工具)在保护模式中运行,它们同早期的实模式开发工具相比已经访问了更多的内存。

得益于使用更多内存这一能力的开发工具的典型例子是 Visual Workbench 程序。最初创建对话框的程序 DLGEDIT 仅有基本功能:只能创建对话框。要创建其他的资源,比如光标和图标,就不得不使用其他的应用程序。由于 Visual C++的 Visual Workbench 可以访问许多特征,这些特征允许用户创建在 Windows 应用程序中需要的所有资源,编辑源文件、管理工程文件,通过调用编译器和链接器创建最后的程序。

今天,由于有了 Visual C++,用户可以在几分钟内开发出 Windows 程序的包容器。可以创建对话框,加上必要的 C++类,画图标,定义菜单和建立及测试应用程序,这些只需花去过去所需要的几年时间中的的一小部分就可完成。

1.2 小 结

本章介绍了 Visual C++ 2.0,它为 Windows 程序员提供了一些功能很强的特征。本章讲了 Visual C++ 2.0 和早期 Visual C++版本之间的全部差别。包括了下面的主题:

(1) Visual C++产品的起源。

(2) Microsoft Foundation Class 库的升级。

(3) OLE 的升级。

第二章　Visual C++ 2.0 的新特征

以前，需要程序员几个月的时间来完成 Windows 程序的包容器。使用 Visual C++后，现在只需用几个键在几分钟内即可完成。

在开发工具上的改进继续提高了生产力并使写 Windows 应用程序的任务更加简单。本章评述了 Visual C++ 2.0 的新特征并且描述了怎样使用它们。

2.1　Visual Workbench

Visual Workbench 是程序员同 Visual C++的界面。由于有了 Visual Workbench，程序员可以访问 C++源代码编辑器，使用内部调试器，并且创建和建立工程文件。另外还可以创建 Windows 应用程序将使用的所有资源。Visual Workbench 常被看作一个 IDE(集成开发环境或编辑器)。不过，由于 IDE 一般用作 Integrated Drive Electronics(关于硬盘驱动的界面标准)的缩写，就不在本书中使用术语 IDE。

由于有 Visual C++ 2.0，Visual Workbench 的基本用户界面已经从 Visual C++ 1.0 和 1.5 中的用户界面中有所改变。有一些主要的变化，比如找到 Visual Workbench 中的 AppWizard。不过即使有这些改变，所有的用户界面是非常相似于早期版本的。有经验的 Visual C++程序员将很快地适应和欢迎 Visual C++ 2.0 提供的新特征。

本章讲 Visual Workbench 的用户界面并且概述了 Visual Workbench 的编辑器。在本章中也有形成 Visual Workbench 每部分工具的信息：编辑器，AppWizard，AppStudio，ClassWizard，和 Browser 集成工具。

2.2　Visual Workbench 用户界面

Visual Workbench 提供了设计很好的用户界面。过去几年里 Microsoft 听取了 Visual C++用户决定在为 Widnows 编程时所想要的东西的意见，这种界面就逐渐形成了。

通过 Visual Workbench 的菜单结构和工具条程序员可以访问大量的特征。有五个工具条可用，并且用户为满足其要求可以创建新的工具条。图 2.1 显示了带有装入了 HELLO 工程文件的 Visual Workbench(取自 Visual C++ 2.0 提供的例子)。这是在建立工程文件之后的结果。在 OutPut 窗口后面，可以看见一个装入到编辑器窗口的源文件，而另一个窗口显示了源文件的头文件。

下面部分考察了在 Visual Workbench 菜单中可用的特征。

2.2.1　File 菜单

File 菜单使用户可以创建一个新的文件或者整个工程文件，选择要编辑的文件，存储当前文件(以原来名称或新名称)，并存储所有的当前打开的文件。

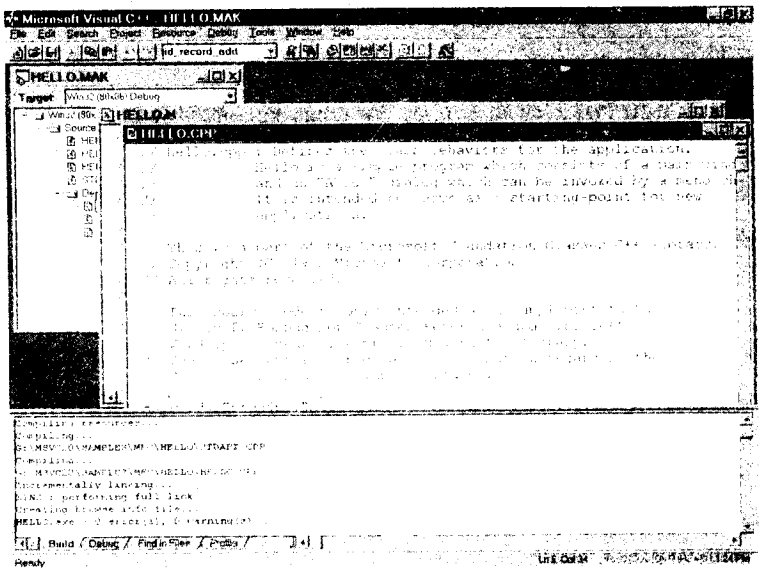

图 2.1 带有正在建立工程文件的 Visual Workbench

File|New 选项使用户可以创建一个新的 Visual Workbench 工程文件。这个菜单选项产生一个对话框以提示用户输入工程文件的名称以及该工程文件要创建的程序(程序平台)的类型(见图 2.2)。此过程由从 Project 菜单中调用 AppWizard 的 Visual C++的早期版本中派生出来的。

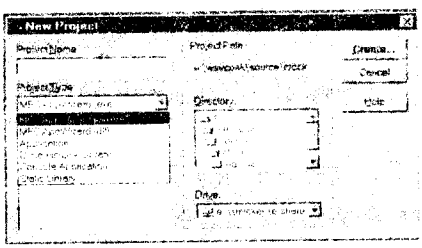

图 2.2 Visual Workbench 的 New Project Name 和 Type 对话框

表 2.1 列出对于 Visual C++有效的平台类型。要使用不是 Win32 的平台,就必须配置合适的交叉开发编译器。在 New Project Name 和 Type 对话框中,还可以选择是否对 Microsoft Foundation Classes 的支持。

表 2.1 Visual C++平台类型

程序类型	硬件类型
Win32 应用程序(.EXE)	IBM 兼容 PC
Apple Macintosh 应用程序	Apple Macintosh 68XXX 系统
Win32 应用程序	运行 Windows NT 的 Alpha 系统
Win32 应用程序	运行 Windows NT 的 MIPS 系统

正如表 2.1 中所见，Visual C++可以创建在大量不同平台下可以执行的程序。Micoroft 负责为 Visual C++ 2.0 创建新的交叉开发平台。Visual C++ 2.0 标准版本将创建 IBM 兼容(80x86)应用程序。对于 MIPS，Alpha 和 Macintosh 系统可以获得 Visual C++ 2.0 的其他版本。

在指定工程文件名称和类型之后，提供给用户又一个对话框，从中选择工程文件的类型。使用 Next 和 Back 按钮在此对话框中设置新的应用程序属性。对话框中这六个区中的每一个代表工程文件创建过程的不同方面：

(1) 可以创建 MDI，SDI 和基于应用程序的对话。

(2) 可按下列级别包含 ODBC 支持：

1) 没有 ODBC 支持

2) 只有头文件，以便后来添加 ODBC

3) 只有数据库视图，不支持数据库文件

4) 对于视图和文件的完整数据库支持

(3) 可按下列级别包含 OLE 支持：

1) 无 OEL 支持

2) 只有包容器支持

3) 只有最小服务器支持

4) 全服务器支持

5) 包容器和服务器的支持

(4) 可按下列内容包含支持：

1) Dockable 图示行

2) 初始状态行

3) 打印和打印预检查

4) 上下文敏感性帮助

5) 3D 控件(CTRL3D.DLL)的使用

6) MRU 列表中的文件数量

7) 高级选项，包括文档模板设置，MainFrame 的选项，包括分隔窗口和 MDI Child Frame 窗口使用

(5) 源程序注释，创建文件的类型以及 MFC 库的用法。

(6) 用于应用程序类的名称以及保存这些类的文件的文件名。

如果想要创建支持不具有视图支持的数据库文件的应用程序，在创建应用程序的工程文件之后，选择 Header Files Only 选项，并使用 ClassWizard 来创建 CRecordSet 类。

工程文件创建过程的最后一步，是确认工程文件选项正是所想要的内容。图 2.3 显示了 Visual Workbench 的 Project Files 对话。

典型的工程文件应该包括的文件类型列在表 2.2 中。使用 File|New 菜单选项，可以创建任何文件类型。

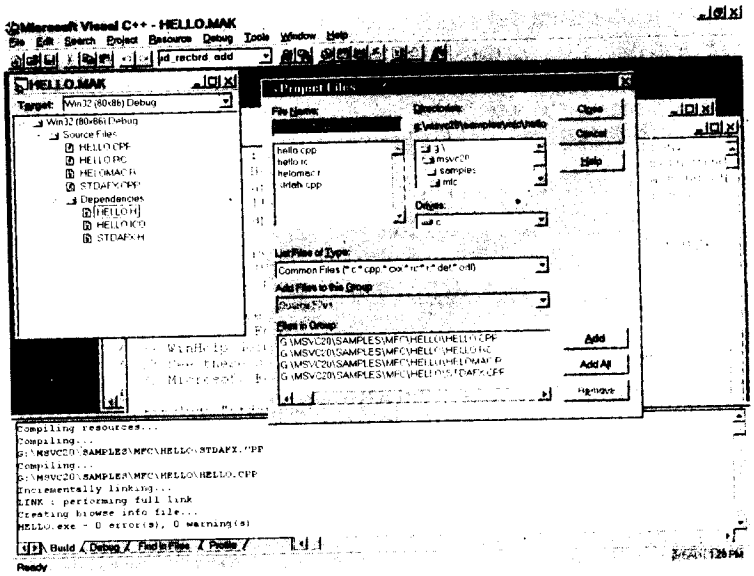

图 2.3 Visual Workbench 的 Project Files 对话框

表 2.2 典型地包含在一个工程文件中的文件

文件类型	扩展名
代码/文本文件	.C,.CPP,.CXX,.H
工程文件	.MAK
资源脚本文件	.RC
位图文件	.BMP,.DIB
图标文件	.ICO
光标文件	.CUR

并非所有的工程文件都要具备上述这些类型的文件。通常 Windows 工程文件必须具备源文件、定义文件(Visual C++自动创建)以及资源文件。Windows 工程文件可能还会有静态链接库以及/或者目标文件。

1. Page Setup

打印支持还包括 Page Setup,可用来定义清单打印输出的格式。Page 格式包括设定边界并且(可选择地)定义页标题和/或脚注。在页标题或标注中,可以通过在页标题或脚注的文本字符串中指定替代代码来选择包含入的被替代的信息。表 2.3 示出了有效的页标题和标注的式代码。

表 2.3 页面布局的页标题和脚注的格式代码

代码	描述
&f	替代要被打印的文件的文件名
&p	替代当前页号
&t	插入当前系统时间
&d	插入当前系统数据

（续表 2.3）

代码	描述
&l	使页标题或脚注左边对齐
&c	使页标题或脚注居中
&r	使页标题或脚注右边对齐

2. 打印

在 File 菜单上提供有打印支持。可以打印部分（当前选择）或者当前文件的全部内容。

3. MRU 文件列表

File 菜单上倒数第二个选项是最近使用（MRU）文件的列表，允许迅速打开最近四个打开的文件之一。这些列表分为两部分：第一部分列出最近使用的源文件，第二部分列出最近打开的工程文件。

4. Exit

使用 Exit 菜单选项终止当前 Visual C++任务。这个选项功能同按下 Alt—F4，从 System Menu 中选择 Close，或击标题条的退出按钮一样。

2.2.2 Edit 菜单

下一个 Visual Workbench 菜单是 Edit。Edit 菜单用剪贴板，Undo 和 Redo 及 Properties 对话控制 Visual C++界面。

1. Undo 和 Redo

Edit 菜单的第一部分是 Undo 和 Redo 支持。使用 Undo 可以取消对已编辑文件所做的改变。可以取消的编辑次数取决于取消缓冲区的大小，这由选择 Tools|Options|Editor...（后面描述）来设定。Redo 选项是 Undo 选项的相反功能：Redo 使用户“取消”一个取消命令，再取消命令去掉的编辑。当在认识到所进行的编辑实际是可接受的之前已经取消了大量编辑时，Redo 是非常方便的。

2. Cut，Copy，Paste 和 Delete

Edit 菜单还包括剪贴板界面。可以看到 Edit 菜单的剪贴板界面特征紧紧遵从 Widnows 标准，提供下列能力：

Edit 提供在活动编辑窗口中剪裁和废除当前被选中文本的能力。Copy 使用户能够把活动编辑窗口中当前选中文本复制到剪贴板上，从那里又可以复制到其他窗口或应用程序中去。用 Paste，可以把当前剪贴板上的内容复制到光标位置（如果剪贴板提供一个可被转化为文本的对象）。Edit 的 Del 菜单选项的功能和 Delete 键相同——删除当前选择且不把它复制到剪贴板上。

3. Select All

Edit 菜单包括一个选项可使用户在不改变当前文本光标位置的同时选择当前文件的内

容。

4. Properties

可以为窗口(可编辑窗口,而非标准输出窗口)设定属性。该菜单选项在当前活动编辑窗口的基础上调出适当的 File Properties 菜单。能够被设置的确切属性由当前活动窗口中的文件类型决定。

下面是文本窗口中可以获得的属性。对于显示资源的窗口,比如对话框,提供的属性将有所变化。

5. Properties 对话框:只读

可以为任意文件设定只读模式,只读的优点在于,当在编辑器中文件被取出时可以防止一些因粗心对文件进行的修改。所有 Visual Workbench 的标准调试窗口(Output,Watch,Locals,Registers,Memory,Call Stack,Disassembly)都是只读的。

6. Properties 对话:语言

在 Properties 对话中的这个选项允许 C(或 C++)语言程序语法的不同部分以不同颜色显视。能被显示颜色的不同语法项目包括注释,常数(数字和字符)和关键字。

编辑器的缺省是根据文件的扩展名(.C 和.CPP 扩展名使用 C 或 C++的语法着色)使语法项着色。这个对话框选择使用户能够重设缺省(指定.C 文件的 C++语法着色,或者为.CPP 文件指定 C 语法着色)或者关闭语法着色。

从作者个人角度感觉语法着色是相当有意义的。帮助程序员在不同的类型中标定错误,比如在需要整型时使用了双精度实型,或者在想要短整型时用长整型。当然,对于非 C 或 C++源代码的文件使用语法着色时并没有这些意义。

2.2.3 Search 菜单

在 Visual Wirkbench 中的下一个菜单是 Search。在 Visual C++的早期版本中,搜索功能是 Edit 菜单的一部分。

1. Find 和 Replace

Visual Workbench 编辑器使用 Find 支持所编辑文件中的文本搜索。另外,可以使用 Replace 功能代替文本。被搜索的文本可以是规则文本或者使用正则表达式(在下一节中描述)形成的文本。

2. Find in Files...

使用 Find in Files 菜单选项,可以搜索所选文件中的文本字符串。被选择的文本字符串可以是规则文本,或者包含有正则表达式。没有子目录选项,使搜索子目录树有些不便。图 2.4 显示了 Find in Files 对话。

3. Match Brace...

在 Search 中

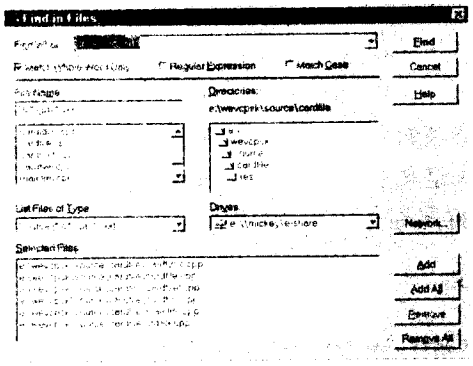

图 2.4　Find in Files 对话框

括号所在行。

4. Go To...

通过指定其行号使用 Go To... 跳向该行。要使用这个选项，必须知道想要跳向的行号。可能会发现，由于一些目的，书签功能（以后将描述）为在文件中移到指定行提供了较好的方法。Go To... 功能最常用在移向一条错误消息中的行。

5. Next Error 和 Previous Error

Serach 菜单中的这两个选项通常在一起使用。在产生编译错误或警告消息之后，Next Error 和 Previous Error 可以定位到错误或警告出现的那一行。当使用 Next Error 和 Previous Error 功能时，错误或警告信息的文本被置于 Visual Workbench 状态条的输出区域中。这样可以简单地看见有错误的行和编译器遇到的问题的描述。不妙的是对于编译器错误以外的错误，Next Error 和 Previous Errfor 信息特征是没有用的，因为只有编译器错误包含行号信息。

6. Bookmark

在 Search 菜单下的第五个选项支持书签，使用书签返回源文件中指定的行。当在一行设定书签时，该行被高亮度化，在左边界有一个小矩形框（在大部分系统中的缺省颜色是蓝青色）。在源文件中可根据各人意愿具有多少书签标记行。不过，如果标志太多行，当寻找想要跳入的行时会变得很麻烦。一般来说使用书签具有三或四个被标记行；不过有一次作者有过 10 个标记。

7. Toggle Bookmark

支持书签的第一个菜单选项是 Toggle Bookmark。当选择该项时。当前行的标记状态被触发。因此，如果当前行被标记，书签被关闭；如果当前行未被书签标记，则成为标记行。

8. Next Bookmark

下一个 Searcah 菜单选项是 Next Bookmark，向前（朝文件结尾的方向）跳向下一个被定

义的书签。如果没有被下义的书签,Next Bookmark 选项就没有作用。当到达文件结尾时,Visual Workbench 返回文件始端开始又一轮书签搜索。

9. Previous Bookmark

紧跟 Next Bookmark 选项的是 Previous Bookmark,向回(朝文件开始的方向)跳向有书签定义的下一行。如果没有定义书签,Previous Bookmark 选项没有作用。当到达文件开始时,Visual Workbench 在文件尾端开始又一轮搜索。

10. Clear All Bookmarks

最后一个书签选项是 Clear All Bookmarks,使用这个选项可以清除在当前文件中已经设定的所有书签。当发现创建了大多数的书签而并不真正有效时此选项很有用。

11. Go to Definition

Search 菜单上的 Go To Definition 选项使用户跳向被选中的符号定义的第一个位置。既可以在源文件窗口又可以在 Browse 窗口选择此符号。

12. Go to Reference

Go to Reference 选项允许跳向选中符号被引用的第一个位置。既可以在源文件窗口又可以在 Browse 窗口选择此符号。

13. Next Definition

Next Definition 选项能够跳向下一个定义或引用。Next 常用以 Go to Definition 或 Go to Reference 之后。

14. Previous Definition

Previous Definition 用于 Go to Definition 或 Go to Reference 之后,跳向前一个定义或引用。

15. Pop Context

当用在 Go to Definition 或 Go to Reference 之后时,这个选项返后最后被使用的符号。

16. Browse

Browse 选项允许移到源文件的不同部分以便能够为符号快速查找定义和引用。既可以在源窗口又可以在 Browse 窗口选择被浏览的符号。Browse 菜单选项的最后部分调用 Browse 功能。通过选择 Project Open.... 创建 Browse 窗口。

2.2.4 Project 菜单

在 Visual Workbench 中,Project 菜单在 Search 菜单之后。使用 Project 创建、修改和存储正在编程的工程文件。

有人会说什么是工程文件呢?工程文件是一种机制,它组合了一个应用程序的所有源文件的组成部分(应用程序可以是 Windows 程序,DLL,也许是 LIB 文件)。工程文件实际上包含在有 .MAK 扩展名的文件中。注意到并非所有的.MAK 文件都是 Visual C++工程文件是非常重要的。某些.MAK 文件只是被创建来提供对 NMake 实用程序的输入(那些需要只用 NMake 来使用的文件称为外部制作或外部工程文件)。不过,如果希望地话,所有的工程文件能够用 NMake 使用。如果直接带工程文件名使用 NMake,就不能获得 Visual Workench 提供的好处,比如跳向错误的能力。AppWizard 是创建工程文件的一种方法,它创建可以同 Visual Workbench 兼容的工程文件,或者只能用 NMake 使用的工程文件。

为什么创建外部制作文件呢? 对于 Visual Workbench 工程文件,必须用 Visual Workbench 来修改工程文件。这就限制了一些在工程文件中能够进行的事情。对于外部制作文件,可以用自己所想的内容编辑和改变。通过编辑 Visual Workbench 工程文件就可以把它转变为外部工程文件。用户会发现通过 App Wizard 或 Visual Workbench 创建的工程文件是简洁的和易于组织的。不过如果对它们的语法不太熟悉时会有一点儿困难。要进一步获知制作文件的信息,应该参考 NMake 实用程序的文献。

每个工程文件菜单选项被在下列的段落中讲述。

1. Files

Project 菜单的第一个选项是 Fiels,能够从当前工程文件中增加或减少文件。在 Visual C++ 1.5 和早期版本中这项功能是 Edit 菜单选项的一部分。

2. New Group

New Group 能够管理 Visual C++工程文件的层次。

3. Settings

Setttings 选项显示 Settings 对话框。使用 Settings 来改变工程文件的可修改选项,包括以下部分:

(1) General:基础类用法和目录
(2) Debug:调试环境的设置在此进行
(3) C/C++:影响编译器的设置上,比如警告级和定义,在此项设定
(4) Link:用于设定链接器的选项和库
(5) Resources:用于设定工程文件的资源选项
(6) OLE Types:用于设定 OLE 类型选项
(7) Browse:用于设定浏览选项

还可以指定应用程序的目标类型。选择包括下列内容:

(1) Win32 Release(发行)
(2) Win32 Debug(调试)
(3) Macintosh Release(发行)
(4) Macintosh Debug(调试)

另外,还能使用这些目标类型:

(1) Win32Application(应用程程序)

(2) Win32Dynamic Link Library(.DLL0)(动态链接库)

(3) Win32Static LinkLibrary(.LIB)(静态链接库)

(4) Win32Console Application(在 DOS 窗口中运行)(Console 应用程序)

4. Targets

Project|Targets 菜单选项允许改变工程文件的目标类型。要了解更多的信息参见标题"Settings"下的部分。

5. Compile <file>....

可以通过选择 Project|Complie<file>...来让 Visual Workbench 编译当前文件。这个菜单选项只编译当前文件。它不调用链接器或者任何其他的建立工具。如果当前文件是工程文件的资源脚本,将调用 Resource Compiler 而非 Visual C++编译器。

6. Bulid<file>

Build <file>....菜单选项能够编译任何被改变的源文件(源文件未被改变,并且有一个现存的目标文件是不被编译的)。在编译过程中如果没有创建错误,其他的工程文件建立工具将被调用来创建最后的工程文件。

7. Rebuild All...

Rebuild All...允许用户编译所有的源文件,不论它们是否被改变。如果编译过程中没有错误,其他的工程文件建立工具将被调用来创建最后的工程文件。

8. Batch Build...

Batch Build...打开 Batch Build 对话框,能单步重新建立不只一个工程文件。如果在主应用程序之外工程文件还包括.DLL 或.LIB 文件并且希望重新建立工程文件的所有部分,这个菜单选项是非常有用。

9. Stop Build

假设选择 Build 或 Rebuild All。编译器开始工作而用户看见一个问题,或许是关于一个头文件。因为所有的编译将会失败,所以判定继续建立过程是无价值的。或许编译只产生警告消息(不会停止建立过程),但是用户希望在完成编译之前解决产生这条警告消息的问题。可以通过选择 Profect|Stop Build 在任何位置停止建立过程。在建立过程的一些点上,不得不选择两次 Stop Build。有时在链接过程中,会发现第一次选择 Stop Build,链接器将停止,但是下一次建立步骤将开始。如果发生这些情况时,只需再次选择 Stop Build。

Microofst 已经增加 Ctrl-Break 作为热键访问 Stop Build。这很容易理解,因为一些程序员本能地按下 Ctrl-Break 来停止这个过程。

10. Execute<file>

在 Project 菜单上的下一个选项是 Execute<file>。使用 Execute<file>，可以运行工程文件的程序(假设工程文件是针对可执行程序，而不是针对链接库. DLL 或 VBX 控件)。Execute 起动程序但在 Visual Workbench 调试器下并不运行它。通常，对于用 Release 选项(见标题为"Option"选项的部分)建立的程序使用 Execute。所有不用调试选项建立的程序可以运行，但不能访问 Visual Workbench 调试选项的任一项。

要在调试器下运行一个程序，必须使用图示行的 go 按钮或选项 Debug|Go。

11. Update Dependencies

用 Project|Update Dependencies 可以扫描当前资源文件——如果它是资源脚本，C，或 C++源文件——确定这个源文件依赖于哪个包含(头)文件。然后这些依赖性被增加进包含文件的工程文件列表。如果已经增加头文件到当前源文件中，然后使用 Update Dependencies 来更新工程文件。通常当增加作为 Visual C++一部分的标准 C 和 C++头文件时，不必更新依赖性，因为这些文件不会被改变。

警告：不要改变 MSVC\INCLUDE 目录中出现的标准 C 和 C++包括文件。

如果有一个不希望包含在依赖性列表中的头文件，可以把它增加进 SYSINCL. DAT 文件中，SYSINCL. DAT 在\MSVC\BIN 目录。这里的任何文件将不会被增加进依赖性中。如果细读此文件，会发现它列出了所有的 Visual C++的包含文件。清单 2. 1 显示了一小部分 SYSTINCL. DAT。

清单 2. 1　SYSINCL. DAT 的部分清单

```
AFX. H
AFX. INL
AFXCOLL. H
AFXCOLL. INL
AFXDB. H
AFXDB. INL
AFXDB. RC
AFXDD _. H
AFXDISP. H
AFXDLGS. H
AFXDLGS. INL
AFXDLL _. H
AFXDLLX. H
AFXEXT. H
AFXEXT. INL
AFXMSG _. H
AFXODLGS. H
AFXOLE. H
AFXOLE. INL
```

AFXOLECL.RC
AFXOLESV.RC
AFXPEN.H
AFXPEN.INL
AFXPRINT.RC
AFXPRIV.H
AFXRES.H
AFXRES.RC
AFXV_DLL.H
AFXV_DOS.H
AFXVER_.H
....

12. Uupdate All Dependencies

用 Project |Update All Dependencies,可以扫描所有的源文件(.RC,.C,.CPP,和.CXX)以确定哪一个包含(头)文件是工程文件的部分,依赖地增加进工程文件向。如果已经为大量源文件更新包含文件,这个选项很有用。如果怀疑是否所有头文件被恰当地包含在工程文件中,可以使用这个选项。通常,Update All Dependencies 是很快的。作者试图时不时地运行它只是为确定已经正确地在工程文件中包括了所有的头文件。尤其对于C++代码,如果工程文件不包含所有的依赖性,会产生一些严重的错误。

13. ClassWizard

Class Wizard 是能够创建对于成员函数的创建新类和映射 Windows 消息的一个实用程序。可以使用 Class Wizard 来操作管理理对话框控件的类。

ClassWizard 实用程序包含一个制表对话框。主要的制表提供这些方面的支持(针对大部分应用程序):

(1) 消息映射:管理同 Windows 类交互的类
(2) 成员变量:管理属于给定类的一部分的变量
(3) OLE Automation:OLE automation 的管理
(4) OLE 事件:OLE 事件的管理
(5) Class Info:类的全部管理

可以添加类、成员变量和成员函数并在作为用户应用程序一部分的 MFC 类上实现大量其他的函数。

14. Close Browse Info File

Close Browse Info File 菜单选项能关闭当前打开的 Browser 信息文件。

15. Build Browse Info File

Build Browse Info File 菜单选项允许根据需要创建一个 Browser 信息文件。如果已经打开在 Project|Setting|Browse Info 下面的 Demand 选项上 Update Browse Info,File,这个菜单

选项是有用的。

2.2.5 Resource 菜单

Resource 菜单管理工程文件的资源。因为 Visual C++ 2.0 已经把原来是 AppStudio 一部分的所有功能建立在 Visual Workbench 中，现在非常容易操作 Windows 程序的资源。

1. New

Resource|New 能够创建新的资源。Visual C++可以创建的资源类型包括下列内容：

(1) 加速键表
(2) 位图
(3) 光标
(4) 对话框
(5) 图标
(6) 菜单
(7) 字符串表输入
(8) 版本表

一个好的特征是具有使用标准化的输入格式创建版本表的能力。

2. Open Binary Data

Open Binary Data 菜单能够为以二进制格式编辑的资源打开二进制编辑窗口。

3. Import

使用 Import，可以移动一个图标、光标或位图到当前资源文件。

4. Exprot

使用 Export，可以从当前资源文件移出一个图标、光标或位图到一个外部文件中去。

5. Symbols

Symbols 菜单选项打开 Symbol Browser 对话框，它列出每个资源符号、符号的值和符号是否当前被使用。可以用此对话框创建新的符号、改变现存符号和删除不使用的符号。

6. Set Includes

Set Includes 菜单选项打开 Set Include 对话框，它列出由应用程序的资源被包含的每个文件。缺省是包含 AFXRES.H，而非 MFC 应用程序通常包含 WINDOWS.H。可以在对话框中增加新的包含文件和编译时间(资源编译)指令。

2.2.6 Debug 菜单

当调试用 Visual C++开发的应用程序时，使用 Debug 菜单。Visual Workbench 集成调试器允许源代码级调试。通过使用混合的汇编/源清单选项，常常能够迅速地查出在 C/