计算机语言技术系列丛书

用 Borland C++ 编写 Windows 应用程序

赵雪泠 编著

希望

学苑出版社

计算机语言技术系列丛书

用 Borland C++编写 Windows 应用程序

赵雪泠　编著
亦　鸥　审校

学苑出版社

(京)新登字151号

内 容 简 介

本书是学习和使用 Borland C++进行 Windows 程序设计的基础读物，书中介绍了用 Borland C++进行 Windows 程序设计的有关要点。

本书对从事软件设计、开发和应用的技术人员具有重要的参考价值。

需要本书的用户，可直接与北京海淀8721信箱书刊部联系，邮码100080，电话2562329。

计算机语言技术系列丛书

用 Borland C++编写 Windows 应用程序

编 著 者：赵雪泠

审　　校：亦　鸥

责任编辑：徐建军

出版发行：学苑出版社　　**邮政编码**：100036

社　　址：北京市海淀区万寿路西街11号

印　　刷：保定列电印刷厂

开　　本：787×1092　1/16

印　　张：10.875　　**字数**：257 千字

印　　数：1～5000 册

版　　次：1994 年 7 月北京第 1 版第 1 次

I S B N：7-5077-0776-8/TP・8

本册定价：14.00 元

学苑版图书印、装错误可随时退换

目 录

第一章 Borland 高级编译器 …… 1
1.1 新功能和应用 …… 1
1.2 运行程序员平台 …… 3
1.2.1 在 MS-Windows 下运行程序员平台 …… 3
1.2.2 Borland C++命令行选择项 …… 5
1.3 Turbo Debugger 2.5 和 3.X …… 6
1.4 Turbo Assembler 2.5 和 3.X …… 7
1.4.1 影响 TASM 的 C 编译器差别 …… 8
1.4.2 MODEL 标识符 …… 10
1.4.3 虚拟段 …… 12
1.4.4 486 指令 …… 12
1.4.5 RETCODE 指令 …… 13
1.4.6 TASMX.EXE 和 Windows DPMI …… 13
1.5 Whitewater 资源开发工具 …… 13
1.6 Resourece Workshop …… 16
1.7 小结 …… 28
第二章 在 Windows 下使用 Borland C++ …… 29
2.1 Windows 编程环境 …… 29
2.1.1 Windows 是一个多任务环境 …… 30
2.1.2 Windows 是一个面向对象的环境 …… 31
2.2 为 Windows 应用程序设置 Borland C++ …… 39
2.2.1 为 RC 和 IMPLIB 设置编译器 …… 39
2.2.2 为入口、出口代码设置编译器 …… 42
2.3 Whitewater 资源开发工具(RT) …… 43
2.3.1 把 RT 设置成 Windows 程序项目 …… 44
2.3.2 示例程序 fcwin.c 需要的资源 …… 44
2.3.3 运行 RT …… 45
2.4 Borland Resourece Workshop …… 53
2.5 小结 …… 63
第三章 设计 Windows 应用程序 …… 64
3.1 设置 Windows 应用程序环境 …… 64
3.1.1 设置用户自己的 Windows 承接程序 …… 64

3.1.2 Windows 目录用法 …… 66
3.1.3 学习实践:设计 fcwin.c …… 67
3.2 建立 Windows 应用程序源文件 …… 100
3.2.1 Windows 3.X 编程环境 …… 100
3.2.2 建立模块定义文件 …… 100
3.2.3 设计程序头文件 …… 102
3.2.4 建立工程文件 …… 104
3.3 小结 …… 104
第四章 编写实用 Windows 程序 …… 106
4.1 设计 Windows 界面 …… 106
4.1.1 登录窗口类 …… 106
4.1.2 设置主消息循环 …… 109
4.1.3 写 Wndproc()函数 …… 110
4.1.4 设置对话回调函数 …… 112
4.1.5 使用 MessageBox()产生帮助和错误提示消息 …… 115
4.1.6 用 Windows 打印管理功能脱机打印 …… 117
4.2 使用动态连接库 …… 125
4.2.1 理解 DLL …… 125
4.2.2 写 DLL 应用程序 …… 126
4.3 小结 …… 130
第五章 使用 ObjectWindows …… 131
5.1 C++的 ObjectWindows …… 131
5.1.1 在 Windows 程序中使用 C++类 …… 132
5.1.2 理解 ObjectWindows 库 …… 145
5.1.3 编写第一个 ObjectWindows 程序 …… 147
5.2 基本 Windows API 函数和元素形成的界面 …… 152
5.2.1 控制应用程序 …… 152
5.2.2 控制窗口对象 …… 153
5.2.3 控制窗口消息 …… 155
5.3 小结 …… 157
附录 A FCWIN 资源的程序清单 …… 158

第一章　Borland 高级编译器

Borland C++ 2.0 和 Borland C++ 3.X 是 Borland 公司最新推出的 C++开发软件，可用来替代 Turbo C++ Professional 软件包。与 Turbo C++ Professional 类似，Borland C++包括许多增强了的功能，其中最主要的是可在保护模式运行及建立 Microsoft Windows 3.X 可执行文件的能力。有了 Borland C++ 3.X，也就有了 Turbo C++ for Windows，其 IDE 在运行时如同一个真正的 Windows 应用程序。

新增添的保护模式功能使用户可以在保护模式下运行程序员平台，这就给予了 IDE 和用户程序更多的内存空间，开发程序的速度也因此加快。

Borland C++最显著的新功能在于：它可以利用 Microsoft Windows 3.X 提供的所有功能来建立应用程序。Windows 函数对程序员是开放的，有了 Windows 函数，就可以轻松地建立使用图形用户界面(GUI)的应用程序。本书将向读者介绍 Windows 应用程序可以使用的许多新资源。

Borland C++ 2.0 软件包中，包括了 Whitewater 资源开发工具。用户使用这些工具，可以根据编程需要，对 Windows 资源进行建立和修改。Borland C++中还包括了 Turbo Debugger，Turbo Profiler 和 Turbo Assembler。这些软件包都已升级，提高了制作效率和易用性。Borland C++ 3.X 和 Turbo C++ for Windows 中，Whitewater 资源开发工具被替换成 Borland Resource Workshop(Resource Workshop)，同时，Turbo Debugger 和 Turbo Assembler 也升级为 3.X 版本。

本章将对 Borland C++所提供的新功能作一简略、概括的介绍。读者将会了解开发软件包的升级情况及它们对用户的帮助。

1.1　新功能和应用

如前所述，Borland C++ 2.0 的两个主要改进之处在于其新增添的建立 Windows 应用程序和保护模式运行的功能。这些功能可以提高编程效率，有助于生成优秀的程序。Borland C++的另一个新功能是预编译头文件。

在 Windows 3.X 下运行的程序可以分为两大类：Windows 应用程序和非 Windows 应用程序。非 Windows 应用程序可以在 DOS 和 Windows 下运行，但不能够利用 Windows 的任何功能，它和真正的 Windows 应用程序在外观上给人的感觉就不相同。

Windows 应用程序设计成在 Windows 环境下运行，因此它依赖于 Windows 提供的界面函数。在设计 Windows 应用程序时编程者不必设计和建立用户界面，只要调用合适的 Windows 函数，就可以生成专业化的易用界面。Windows 管理着运行图形用户界面所需要的所有繁琐的细节。图 1.1 给出了将在本书第二章中建立的一个 Windows 应用程序的例子。

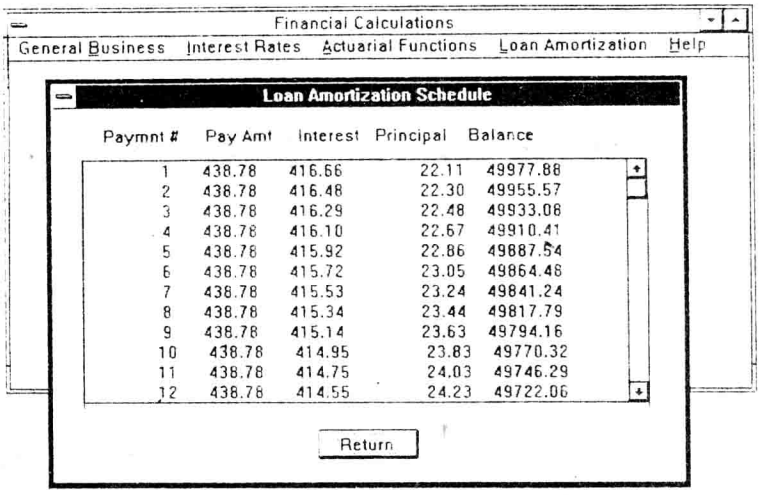

图 1.1　程序 fcwin.c 屏幕

建立 Windows 应用程序有许多益处。最主要的益处在于其统一的用户界面。另一个益处在于它的速度，用户可以很快地把一个同外界联系的复杂程序制作出来。Windows 提供了支持程序，用户可以很方便地使用计算机的视频和其他 I/O 端口。建立 Windows 应用程序时，用户可以控制启动多个程序所发生的结果。编写 Windows 程序使用户可以利用 Windows 的多任务处理能力。

第二、三、四章将介绍在 Windows 下编程。在 Windows 下编程并不太难，但如果读者从未编程设计过多任务处理环境，就需要一些练习。

Borland C++可以在保护模式下运行，这是一种建立在 80286，80386，80486 处理芯片基础上的特殊的运行方式。保护模式改变了计算机主内存的存取方式，在保护模式下用户可以使用比实模式下更多的内存。Borland C++为自身代码段和用户编写的程序占用了额外的内存，因此保护模式被激活时，Borland C++在内存中存储了更多的数据，连接程序也就可以更快地运行。

若想使用 Borland C++的保护模式版本，可运行 BCX.EXE 程序。尽管保护模式和标准模式下内部结构不同，但两者在外观和处理过程上是相同的。这两种版本的 IDE 之间的唯一区别在于它们使用计算机内存方式的不同。

读者如果已经领略过坐在计算机前等待程序编译完成的滋味，就会更加欣赏预编译头文件功能。这项功能使用户可以建立和使用预编译后的头文件。读者通过阅读下面的对头文件和编译过程的简略介绍，就可以理解这项新功能对用户的帮助。

在编译程序时，Borland C++对用户包括的头文件进行语法分析。这个过程中将建立符号表，用来记录程序中使用的说明和定义的信息。没有预编译头文件功能时，每次编译程序都将建立一个新的符号表，编译完成后抛弃。

具备了预编译头文件功能，用户就可以指定在编译过程结束后保留那些为特定头文件组建立的符号表。下次编译的程序中若使用了这组头文件，Borland C++就不再建立新的符

号表，而是从磁盘中直接读取存储的符号表。这样便节约了程序编译过程中的大量时间。

注意，在建立预编译头文件时，符号表是存储在磁盘上的。存储空间不多的情况下，要留心将建立多少符号表映像。

使用了单个预编译头文件的所有源文件中，下述条件必须相同：

1. 相同的头文件使用顺序必须相同，而且被包含的头文件必须具有相同的时间标志。这样的头文件可以被直接包含，也可以被间接包含，后一种方式是在另一个 #include 文件中以镶嵌的形式包含。

2. 宏必须被定义成具有相同的识别数值。

3. 必须使用相同的语言，也就是使用 C 或 C++，但下一次编译的使用了预编译头文件的程序中，不能改变语言。

4. 内存模式必须相同。

5. externs 中的下划线必须匹配。

6. 目标环境必须相同。在 Borland C++中，用户可以选择 DOS 标准、Windows 可执行或 Windows DLL(动态连接库)目标类型。

7. Generate Word Alignment 选择项必须按相同的方式进行设置。

8. Pascal 类型调用使用方式必须相同。

9. 若把枚举型数值以整型数值对待，那就必须始终以这种方式来处理枚举型数值。

1. 若缺省时 char 型为无符号型，在使用预编译头文件时也应如此。

11. 在编译 C++程序时可以生成小型、局部型、公共型或远型虚拟表。每次使用预编译头文件时，务必要使用相同的选择项。

1.2 运行程序员平台

本节将介绍怎样在用户选择的 DOS 或 Windows 环境下运行 Borland C++。前一部分将论述怎样在 Windows 下运行 Borland C++，后一部分将论述启动 Borland C++时可供使用的命令行选择项，这些选择项主要是程序在通常 DOS 环境下运行时使用。

1.2.1 在 MS-Windows 下运行程序员平台

Borland C++和 Turbo C++ for Windows(若后者是单独买来的)都提供了安装步骤，可以建立一个 Windows 程序组和适当的程序项目。

Borland C++并不是 Windows 应用程序，但在 Windows 下可以很容易地运行此软件包。这种运行方式需要一些解释程序。要在 Windows 下运行 Borland C++，需要建立程序信息文件(PIF)并把 Borland C++加入程序组中。PIF 是 Windows 使用的特殊文件，用来建立和运行 Windows 环境下的应用程序。

用户可以使用 Windows 内部设置的 PIF 编辑器来建立一个 PIF 文件。启动 PIF 编辑器后，挑选 New 选择项，会看到如图 1.2 所示的屏幕。

图 1.2 所示的 PIF 屏幕是两个 PIF 编辑屏幕的前者，这两个屏幕都是在 Windows 386 增强模式下运行(标准模式屏幕是增强模式屏幕的简略版本)。PIF 编辑程序的第一个区域

是程序文件名区域，在这里，输入 TC. EXE 文件的完整路径和文件名。

Windows Title 框中用于输入将在 Program Manager 屏幕上显示的标题。Optional Parameters 和 Start-up Directory 区域并不一定需要处理。用户若不在 TC. EXE 所在的目录中存储程序源代码时，就可以在这里指定目录。

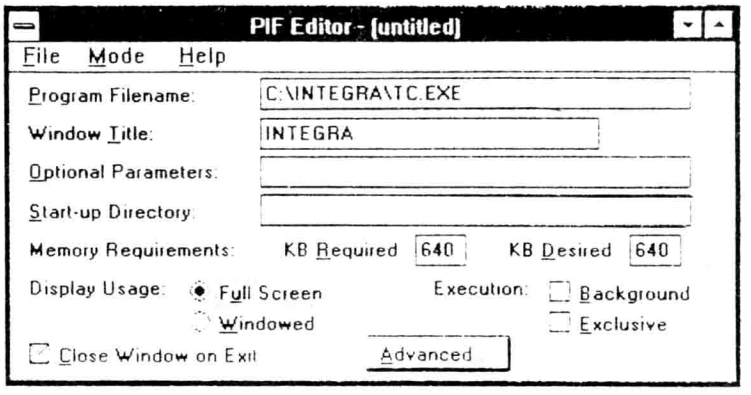

图 1.2　386 增强模式下的 PIF 编辑屏幕

Memory Requirements 区域是很重要的。若用户指定 640K 为需要的内存数量，就不必考虑 Desired memory 区域。

Borland C++的 IDE 并不是真正的 Windows 应用程序，它只能在全屏幕幅度下运行。用户不能像收缩 Windows App 窗口一样来收缩 Borland C++的窗口。

用鼠标器单击增强模式下 PIF 屏幕底部的 Advanced 按钮会弹出另一个装满选择项的菜单。图 1.3 给出了第二个 PIF 编辑器屏幕。

除非具有特殊需要，最好是不指定 Windows 的 Background 和 Foreground Priorities 这两项，使之缺省设置。

在 Memory Options 框中，用户可以选择自己的应用程序需要的内存空间的数量和种类。EMS 内存是扩充内存，XMS 内存是 386 或 486 机器上的扩展内存。用户分配给使用 Borland C++的内存数量取决于机器安装的内存数量。

若想在 Windows 下运行保护模式版本的 Borland C++，Windows 的运行模式只能为实的或标准的模式。不可能同时使用增强模式的 Windows 和保护模式的 IDE。

在 Windows 下运行 Borland C++时需要执行 TKERNEL. EXE 程序。在 DOS 环境下运行 TCX. EXE 时自动运行了 TKERNEL. EXE。但在 Windows 下运行 TCX. EXE 时，需要按如下参数运行 TKERNEL. EXE：

```
TKERNEL hi=yes Kilos=nnnn
```

上式中的 nnnn 参数告诉 TKERNEL 可以管理多少千字节的内存。Borland 推荐此参数设置为 2048。运行 TKERNEL. EXE 后，就可以在标准模式下启动 Windows，所有剩余的扩充内存都可以被 Windows 使用。

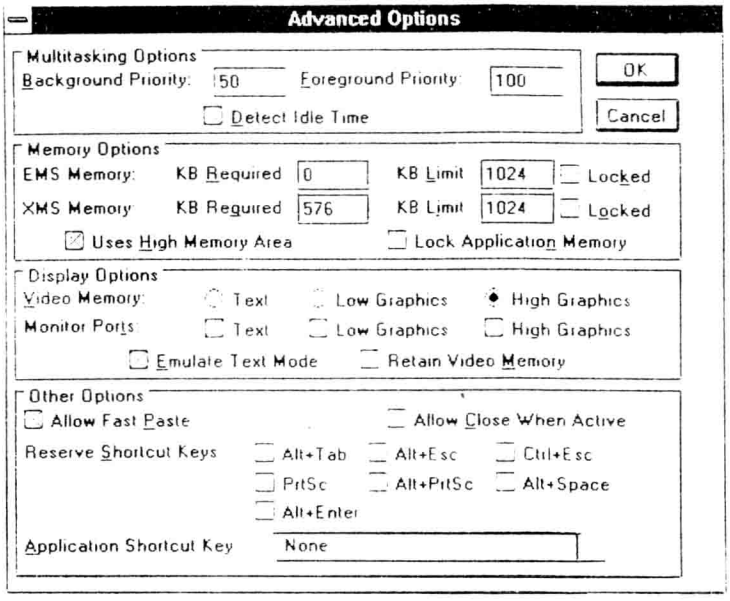

图 1.3　第二个 PIF 编辑器屏幕

为 Borland C++建立好 PIF 文件后，就可以把程序加入到用户自己的一个 Windows 程序组中。这个过程的第一步是使用鼠标来移动亮光条到 Borland C++要添加的程序组中。第二步是在 Program Manager 主菜单下选择 File|Add 菜单命令。第三步是选择 Add Program Item 无线按钮，因为这时只把程序加入到程序组中。

第四步是在下一个对话框中填写启动 Borland C++用的命令行。若 Borland C++是安装在缺省目录中，请输入下面这条命令行：

C:\BORLANDC\BIN\BC. PIF

这行命令告诉 Windows 加载 PIF 文件后执行 PIF 中描述的程序。用户还需要告诉 Windows 一个新的程序项目名，这个项目名输入在对话框的 Description 区域中。

结束输入所有信息后，Windows 把 Borland C++加入到用户指定的程序组中。下一次需要启动 Borland C++时，只要用鼠标器双击程序组名，Windows 将根据用户在 PIF 中指定的选择项来启动编译器。

1.2.2　Borland C++命令行选择项

用户启动 BC 或 BCX 时可以指定几个命令行选择项。命令行选择项的使用格式如下：

BC|BCX 选择项(可选)源名(可选)|工程名(可选)源名(可选)

使用源名或工程名可选项，用户可以指定启动 Borland C++时加载哪个源文件或工程文件。在这里指定文件名，可以省去在 IDE 中定位和打开文件(用户可以同时指定工程和文件名，以打开工程并装载文件进行编辑)。

命令行中可以指定十个可选项。下面列出了每个可选项及各自的作用。

1. /b 选择项使 Borland C++重新建立(build)工程中的所有文件。Borland C++将启动、装载当前工程或命令行中指定的工程,重新编辑并连接所有文件后返回操作系统。

2. /d 选择项用于在检测到必需的硬件时开启双监视器模式。

3. /e 选择项指示 Borland C++把数据交换到扩展内存中。通常分配内存时,Borland C++把数据交换到硬盘中去。使用/e 选择项则交换到扩展内存中去。/e 格式如下:

$/e=n_{可选}$

这里 n 是用户指定 IDE 可使用的扩展内存页面数,每个页面是 16K。

4. /x 选择项作用和/e 选择项类似,只是这时数据是交换到扩充内存中去。/x 命令使用格式如下:

$/x=r_{可选},n_{可选}$

这里 r 指定了为其他应用程序保留内存的千字节数,n 指定了分配给交换数据用的内存千字节数。

5. /h 选择项列出了所有可使用的命令行选择项。

6. /l 选择项指定启动 Borland C++时使用液晶(LCD)屏幕显示模式。

7. /m 选择项使 Borland C++制作(make)用户工程中的所有文件。这时只重新编译和连接过时了的文件。/m 和/b 的区别就在于这里。

8. /p 选择项控制着 EGA 系统调色板(palette)的数据交换。若用户程序修改了 EGA 调色板而且打开了/p 选择项,屏幕进行数据交换时都要恢复原先的调色板。

9. /r 选择项指示 Borland C++把数据交换到 RAM 盘中而不是硬盘中。/r 选择项的使用格式如下:

/rx

这里 x 是 RAM 盘所在的驱动器名。例如,/re 指示 Borland C++把数据交换到 E:RAM 盘中。

1.3 Turbo Debugger 2.5 和 3.X

Turbo Debugger 2.5 及以上版本的升级之处在于它能够调试 Windows 应用程序。升级后的 Turbo Debugger 可以和 Borland C++ 2.0 和 3.X 一起使用。

Borland 公司在 Borland C++软件包中提供了两个版本的 Turbo Debugger。第一个版本 TD.EXE,是在 DOS 下使用的普通版本。第二个版本 TDW.EXE,是在 Microsoft Windows 3.X 下运行的特殊版本。由于 Turbo Debugger for Windows 是一个 Windows 应用程序,它必须安装在 Windows 程序组中。

安装 Tubo Debugger for Windows 是很容易的。安装调试程序时,Turbo Debugger for Windows 将随 Borland C++软件包一起拷贝到硬盘中。缺省情况下,它在 BORLANDC\BIN 子目录中。

由于所有的 Turbo Debugger for Windows 文件已经存在磁盘上,唯一剩余的任务是把 Turbo Debugger 添加到一个 Windows 程序组中。这项工作通过以下三步来完成。首先,用鼠标器选择要加入 Turbo Debugger 的程序组;其次,从 Program Manager 主菜单行中选择 file|

Add 子菜单项；最后，把程序项目加入到当前程序组。加入程序项时，Windows 将给出程序在程序组中使用的名字以及用来启动程序的命令行。若 Turbo Debugger 安装在缺省目录中，可使用下行命令：

C:\BORLANDC\BIN\TDW.EXE

Turbo Debugger 是 Windows 应用程序，程序的所有图标都存储在 TDW.EXE 文件中。用户可以使用 Turbo Debugger 的特殊图标，不用 Windows 提供的图标。

使用 Turbo Debugger for Windows 和使用通常的调试程序相类似。不同之处在于 Turbo Debugger for Windows 可以显示 Windows 的特定信息。这些信息包括：

1. 用户程序窗口中传递的消息。
2. 一个完整的程序组成模块表，包括 DLL(动态连接表)模块。
3. 局部和全程堆阵。

与 DOS Turbo Debugger 相比，有一些功能 Turbo Debugger for Windows 不支持，这是由于这些功能不适用于 Windows 环境。这些功能包括：

1. 不支持硬件调试。
2. 不支持设备驱动器和 TSR 的调试。
3. 没有敲键记录选择项。
4. DOS Shell 是不需要也不适用的。
5. 用户不可退出 TDW(Turbo Debugger for Windows)后使之驻留。
6. 不支持 Table Relocate，因为无法设置基地址和符号表。
7. 使用 Ctrl-Alt-SysReq 组合键可以完成从正在调试的应用程序到 TDW 的转换。
8. 设置 Options|Path for Source 同时也设置了启动目录。

Turbo Debugger for Windows 至少需要在以 80286 为 CPU 的机器上运行。这是因为 TDW 只能运行在 Windows 标准或增强模式下，而这些模式只适用于 286,386,486 存储器。

1.4 Turbo Assembler 2.5 和 3.X

Borland C++最新软件包提供了一组新功能，Turbo Assembler 并未排除在通用功能提升之外。Turbo Assembler 2.5 和 3.X 包括了 TASM 2.0 的所有新功能，并增加了一些重要的功能。本节中就将介绍这些新功能，其内容包括：

1. 编译器的差别影响着 TASM 2.5 和 3.X 的特性。Turbo C++ 2.0 和 Borland C++ 2.0 和 3.X 编译器之间的差别影响着 C 和汇编程序的接口方式。

2. Windows 3.X 支持的 MODEL 标识符。Borland C++软件包最重要的新功能是它支持 Windows 3.X 应用程序的编程。TASM 2.5 及以上版本提供的功能使用户可以在 Windows 环境下混合开发汇编和 C 语言程序。

3. 支持虚拟段。TASM 2.5 和 3.X 从 C++中借用虚拟段注释。虚拟段通常被看作属于包围它们的段，并继承包围它们的段的特性。

4. 支持 486 指令。Intel 80486 以强劲的功能占据了 CPU 市场。TASM 2.5 和 3.X 具有新的 486 指令和汇编伪指令，以支持高效的 80486 CPU。

5. RETCODE 伪指令。这个新增添的伪指令可以产生基于当前内存模式的近或远返回指令。

1.4.1 影响 TASM 的 C 编译器差别

用户在用 C 和汇编语言混合编程时，有必要弄清 Turbo C++和 Borland C++ 2.0 和 3.X 间的一些差别。这些差别包括下面的一些新功能和需要。

在 Borland C++程序中，不要混合 TASM 1.0 直接插入汇编语句。TASM 1.0 已经过于陈旧，它不能处理 Borland C++直接插入汇编所产生的汇编语句。但要知道 Borland C++具有内部汇编程序，只有在使用 #pragma 直接插入伪指令或-B 编译伪指令来显式地调用外部汇编程序时，上面的限制才起作用。

读者已经学习了怎样从汇编返回 3 字节或大于 3 字节的结构到 C 语言中。规则表明，用户需要把这样的结构拷贝到一个全程静态区，并返回一个指向结构的指针(对于小型或大型内存模式在 AX 或 DX:AX 中)。

但也有例外的情况。Turbo C++和 Borland C++软件包中的 README 文件表明，一个返回类型为 Stuct 的函数，要求传递一个隐含指针给它，这个指针通常应该是 far 型，它指向函数拷贝结构去向的区域。若不执行拷贝，或不返回指针，用户的努力都是无效的。被指向的位置常处于堆栈段(这完全废弃了返回地址数值取得的效率收益)。

总之：若汇编模块建立的 struct 不在全程静态区域时，把它拷贝到一个这样的区域中。返回隐含参数指向区域的地址到调用处，调用处 C 代码段可能要求从原位置重新拷贝 stuct。例如在下面这段简短的 C++程序中，要求汇编语言模块建立并返回一个 strint 类对象，它也是一个 stuct。

```
#include <iostream.h>

class strint {
  int a ,b, c;
public:
  void show() {
    cout << a << ' ' << b << ' ' << c << endl;
  }
}

extern strint rsasm(); // Create and return class obj

void main()
{
  strint i;
```

```
    i=rsasm(); // Call TASM routine to build class obj
    i.show();
}
```

下面的汇编语言程序很好地完成了任务。注意其中需要 mangled 函数名。

```
                DOSSEG
                .MODEL small
ISTRUCT         STRUC
int1            dw              0
int2            dw              0
int3            dw              0
ISTRUCT         ENDS
                .DATA
classobj ISTRUCT <2,3,4> ;      Create class object / struct
                .CODE
;
;Use C++ mangled function name.
;   Note that do not use the C language spec. for the
;               PUBLIC or PROC directives so that underscores
;               will not be generated ( C++ mangled names only )
;
                PUBLIC @rsasm$qv
@rsasm$qv PROC
                push            bp
                mov             bp,sp
                lea             si,classobj     ; offset sender,DS already set
                push            [bp+6]              ; seg adrs of recv'g loc
                pop             es
                mov             di,[bp+4]       ; offset address of recv'g loc
                mov             cx,3                        ; copy 3 words
copyit:
                lodsw
                stosw
                loop copyit
                mov ax,[bp+4]   ; small model ptr is just offset
                pop             bp
                ret             ; and return to caller
```

```
@rsasm $ qv   ENDP
              END
```

在 Turbo C++ 2.0 中用户可以说明 far 型函数或指针。而在 Borland C++中用户则可以说明一个 far 型对象。说明一个 far 型对象则把它定位在它原先所在的 far 段中(就像使用的是巨型内存模式一样)。例如:

```
int far i=0;
int far j;
```

上两句中的整型变量 i 和 j 就为每个数据对象建立了单独的 far 段。用户也可以使用 extern 和 static 说明符来说明 far 型对象。

用户可以根据需要来控制 far 型的段名、类和族,这可以通过使用命令行编译器(TCC)选择项、编译伪指令 #pragma option,或通过上面两种方式的结合来实现。命令行选择项如下:

-zE 段名

把段名设置成已说明的数值。这里可以是 TASM 认为合法的任何段名。

-zF 类名

把段类设置成已说明的数值。这种说明和说明语句 SEGMENT...'类名'效果相同。

-zH 族名

定义了 far 型段属于的族。

命令行选择项只允许对一个 far 型段定义一次。若使用 C 伪指令 #pragma option,则可以把 far 型对象安置在几个不同的 far 型段之中,并可以按要求的任何方式来进行组合。例如:

```
#pragma option -zEusrseg -zHusrgroup -zFusrclass
int far i;
int far j;
#pragma option -zEappseg -zHappgroup -zFappclass
int far k;
int far p;
#pragma option -zE* -zH* -zF*
```

这段程序把 far 型对象放置在 far 型段 usrseg 中,把 far 型对象 k 和 p 放置在 far 型段 appseg 中。最后一行把 far 型段名、类、族复位为原缺省值。

1.4.2 MODEL 标识符

读者已经知道,Borland C++支持 Windows 3.X 编程环境,TASM 2.5 和 3.X 也有与之相对应的增加功能。用户特别需要注意 PROC 伪指令中的新关键字,它们用于生成与 Windows 兼容的入口和出口代码段。PROC 伪指令使用格式如下:

程序名 PROC 语言修饰符(可选)语言(可选) NEAR|FAR(可选)

上式中合法的语言修饰符可以是 NORMAL 或 WINDOWS。NORMAL 语言修饰符选择正常的入口和出口代码顺序，WINDOWS 标识符选择与 Microsoft Windows 编程环境兼容的入口与出口代码顺序。

NORMAL 和 WINDOWS 语言修饰符可以位于任何合法的语言关键字之前。NORMAL 关键字生成的入口和出口代码顺序如下：

```
;Entry/exit sequences not generated if no args or locals.
;Note that the 186 Version uses ENTER/LEAVE.
push bp
mov bp, sp
sub sp,local_size                ;Do this if locals used
[push uses registers here]
...                              ; User code goes here
[pop uses registers here]
mov sp,bp                        ;Do this if locals used
ret
```

为 Windows TASM 应用程序生成的入口和出口代码段如下：

```
push ds
pop ax
xchg ax,ax
inc bp
push bp
mov bp,sp
push ds
mov ds,ax
sub sp,local_size          ; if local variables used
[push uses registers here]
...                        ;User code goes here
[pop uses registers here]
sub pb,2                   ; if local variables used
mov sp,bp                  ; if local variables used
pop ds
pop bp
dec bp
ret
```

Windows 环境下允许建立动态连接库(DLL),只在段安排上有一些特殊的要求(可从第四章中学习有关这个主题的更多内容)。特别要注意的是,一个 DLL 模块没有属于它自身的堆栈段,它只能使用调用它的应用程序模块的堆栈段。MODEL 伪指令可以说明一个不在任何数据段之内的堆栈段,例如:

```
.MODEL SS_NE_DS LARGE ;Equivalent to TC'S large model
```

这里 SS_NE_DS 标识符和说明语句 ASSUME SS:NOTHING 作用相同,表明堆栈不是 DGROUP 的一部分。

最后还要指出,还有一个 WINDOWS.INC 汇编程序包含了用汇编语言编写 Windows 应用程序的文件。

1.4.3 虚拟段

用户可以把段说明为 VIRTUAL(虚拟)型。在 TASM 2.0 中首先引入了这个概念,这里值得重新提及。一个 VIRTUAL 段是一个公用区域,它属于包围它的段。说明一个 VIRTUAL 段的通常格式如下:

```
segname SEGMENT VIRTUAL ;MASM mode
...
ENDS

SEGMENT segment VIRTUAL ;Ideal mode
...
ENDS
```

一个 VIRTUAL 段和一个 COMMON 段类似,只是 VIRTUAL 段代表着公用区中的一组数据对象,而 COMMON 段给出的是覆盖投影数据对象。而且一个 VIRTUAL 段被认为通过它的父(包围它的)段的段寄存器来寻址。

1.4.4 486 指令

TASM 2.5 和 3.X 可支持 INTEL 80486 CPU。它增加了以下伪指令和指令:

(1).486,.486c

只有在 MASM 下,这些伪指令才可以使用包括 80486 实模式无优先权指令和浮点指令在内的汇编语言。

(2)P486N

允许使用 80486 处理器的实模式无优先权指令。

(3).486p

只有在 MASM 模式下,才允许使用所有的 CPU 和协处理器指令。

(4)P486