

Windows 程序设计

——用 Borland C++ &

Turbo C++ for Windows

第四版

章生立 董三立

北京航空航天大学出版社

Windows 程序设计

—用 Borland C++ & Turbo C++ for Windows

章生立 董三立等 编

北京航空航天大学出版社

(京)新登字 166 号

内 容 简 介

本书介绍利用 Borland C++ 和 Turbo C++ for Windows 进行 Windows 程序设计的技术和参考信息。首先介绍 Windows 的优点、概念和术语,如何控制 Windows 环境和用 C 与 C++ 编写简单的应用程序。接着讨论如何控制 Windows 窗口;如何设计图标、光标和位图;如何开发菜单和键加速器;如何编写使用对话框和字体的代码;如何绘制图形[科学图、数学图、饼形图、条形图和线图];如何设计多媒体的程序。再阐述使用 ObjectWindows 类库(OWL)开发 Windows 应用程序的基本概念和技巧;介绍了从 Windows 3.1 移植程序到 Windows NT 的方法。最后列出了 Windows API 和 ObjectWindows 类库参考和 Borland 集成环境工程管理的用法。适合程序员使用,也可供计算机专业高年级学生、研究生参考。

图书在版编目(CIP)数据

Windows 程序设计:用 Borland C++ 和 Turbo C++ for Windows/章生立等编. —北京:北京航空航天大学出版社
1995.6
ISBN 7-81012-577-X

I. W... II. 章... III. ①C 语言-程序设计②操作系统, Windows-程序设计 IV. TP312

中国版本图书馆 CIP 数据核字 (95) 第 04672 号

- 书 名: Windows 程序设计 / 用 Borland C++ & Turbo C++ for Windows
- 编 著 者: 章生立 董三立
- 责任编辑: 王鉴莉 许传安
- 出版者: 北京航空航天大学出版社
- 地 址: 北京市海淀区学院路 37 号(100083), 发行科电话: 2015720
- 印 刷 者: 朝阳科普印刷厂印刷
- 发 行: 新华书店总店科技发行所
- 经 售: 全国各地书店
- 开 本: 787×1092 1/16
- 印 张: 45
- 字 数: 1168 千字
- 印 数: 5000 册
- 版 次: 1995 年 6 月第 1 版
- 印 次: 1995 年 6 月第 1 次印刷
- 书 号: ISBN 7-81012-577-X/TP·165
- 定 价: 54.00 元

编者前言

Borland C++的前身为 Turbo C。从 2.0 版开始, Turbo C 演化为 Borland C++ 和 Turbo C++ 两个系列。Borland C++ 后来又推出了 2.0, 3.0.3.1 和 4.0; Turbo C++ 推出了 2.0 和 3.0。Borland C++ 非常庞大, 主要被专业程序员使用, 而 Turbo C++ 相对较小, 主要用于教学, 为初学者所喜爱。

从 Turbo C++ for Windows 1.0 开始, 集成环境运行于 Windows, 是一个真正的 Windows 应用程序。而 Borland C++ 从 4.0 开始, 是一个很好用的 Windows 应用程序开发环境。要特别指出的是, 新的 IDE(集成开发环境)可在 Windows 中直接调试程序, 工程管理比以前更方便。

Borland C++ 和 Turbo C++ for Windows 配备了 Turbo Debugger for Windows 用来调试设计的 Windows 应用程序, 增强了包括图标、光标、位图、对话框、菜单、热键和字体等的资源设计。特别是在 Borland C++ 4.0 中, 资源设计已得到显著增强, 用来设计 Windows 应用程序的界面。Borland C++ 在 OWL(ObjectWindows Class Library)中封装了 Windows API(Application Program Interface)包括函数、数据结构和消息等, 可以加快读者用 C++ 开发 Windows 应用程序的速度, 减轻读者用 C++ 开发 Windows 应用程序的难度。

本书适用于 Borland C++ 和 Turbo C++ for Windows 以及各种演化版本。书的最后, 还介绍了用 Borland C++ 4.0 开发 Windows NT 应用程序的技术。

书中的例子针对国内程序员的需要, 全部采用中文的菜单、对话框和提示界面, 每个程序都有很好的注释和解说, 便于理解和学习。

目 录

第一章 Windows 和 Windows NT 介绍	1
1.1 什么是 Windows	1
1.2 Windows 的功能	1
1.2.1 标准的用户界面	2
1.2.2 多任务	2
1.2.3 内存管理	2
1.2.4 队列输入	3
1.2.5 消息	3
1.2.6 设备的独立性	4
1.2.7 动态连接库	4
1.3 Windows 的特征	4
1.4 Windows 的概念和术语	5
1.4.1 一个窗口的定义	5
1.4.2 可视界面	5
1.4.3 窗口类	7
1.4.4 面向对象的程序设计	7
1.4.5 消息传递	10
1.4.6 使用 Windows 函数	13
1.4.7 Windows.h 头文件	13
1.5 创建一个 Windows 程序的步骤	14
1.6 必备的条件	15
1.6.1 关于 Windows 工具	15
第二章 资源和资源编译器	17
2.1 Windows 资源	17
2.2 使用 Borland Resource Workshop (BRW)	18
2.2.1 创建图标、光标和位图	18
2.2.2 定制图标和定制光标	21
2.2.3 怎样创建菜单	22
2.2.4 菜单结构	22
2.2.5 怎样使用对话框输入数据	25
2.2.6 对话框编辑器	26
2.3 从命令行中使用资源编译器 RC	32
2.3.1 资源语句	33
2.3.2 单行语句语法	34
2.3.3 编译资源	34

第三章 访问 Windows 环境	36
3.1 坐标系统	36
3.1.1 8 种映像模式	36
3.1.2 设备坐标	37
3.1.3 视口	37
3.1.4 MM_ISOTROPIC 和 MM_ANISOTROPIC 的说明	37
3.1.5 改变缺省坐标	38
3.2 选择初始窗口大小、位置、光标、图标和风格	38
3.3 ShowWindow 函数	47
3.4 SetClassWord 函数	48
3.5 虚拟键	49
3.6 控制和对话框	51
3.6.1 静态控制	51
3.6.2 按式按钮控制	51
3.6.3 单选按钮控制	51
3.6.4 复选框控制	51
3.6.5 编辑框	52
3.6.6 列表框	52
3.6.7 滚动条	52
3.7 系统计时器	53
3.8 内存	53
3.8.1 内存分配	54
3.8.2 内存管理	54
第四章 编写简单的 Windows 程序	57
4.1 入门	57
4.2 Windows 应用程序的基本组成	58
4.2.1 WinMain 函数	58
4.2.2 注册窗口类	58
4.2.3 创建窗口	62
4.2.4 显示及更新窗口	62
4.2.5 消息循环	62
4.3 窗口函数	63
4.4 创建模块定义文件	66
4.5 创建 MAKE 文件	67
4.6 联编	68
4.7 怎样使用 SWA 创建其它的 Windows 应用程序	69
4.8 创建 Include 文件	71
4.9 资源文件	72
4.10 MAKE 的回顾	75

第五章 控制 Windows 窗口	76
5.1 什么是滚动条	76
5.1.1 向下的含义	76
5.1.2 滚动条范围	76
5.1.3 滚动条位置	76
5.1.4 滚动条类型	77
5.2 怎样使用滚动条编写应用程序	77
5.2.1 Make 文件(SCROL. MAK)	82
5.2.2 定义文件(SCROL. DEF)	82
5.2.3 文档文件(SCROL. DOC)	83
5.2.4 应用程序文件(SCROL. CPP)	83
5.3 如何使用系统计时器	87
5.3.1 MAKE 文件(TICK. MAK)	91
5.3.2 定义文件(TICK. DEF)	91
5.3.3 消息文件(TICK. DOC)	91
5.3.4 应用程序文件(TICK. CPP)	91
5.4 如何创建一个抵押偿还表	94
5.4.1 LOAN. MAK Make 文件	100
5.4.2 LOAN. DEF 模块定义文件	101
5.4.3 LOAN. CPP 应用程序文件	101
第六章 设计图标、光标和位图	104
6.1 使用 Windows 工具创建图标、光标和位图	104
6.2 使用图标	108
6.2.1 Make 文件(ICON. MAK)	111
6.2.2 定义文件(ICON. DEF)	111
6.2.3 资源文件(ICON. RC)	111
6.2.4 应用程序文件(ICON. CPP)	111
6.3 使用光标	111
6.3.1 Make 文件及定义文件(CUR. MAK 及 CUR. DEF)	114
6.3.2 资源文件(CUR. RC)	114
6.3.3 应用程序文件(CUR. CPP)	114
6.4 使用 Borland Make	115
6.5 使用资源编译器	116
6.5.1 资源编译器语句	116
6.5.2 使用资源编译器编译	118
6.5.3 与 Borland Resource Workshop 兼容的资源	119
第七章 开发菜单和键加速器	120
7.1 菜单机制	120
7.1.1 什么是菜单	120

7.1.2	菜单关键字和选项	121
7.2	随时创建菜单	125
7.2.1	创建菜单来改变图形的尺寸	125
7.2.2	用菜单改变背景颜色	130
7.2.3	用菜单决定系统信息	138
7.2.4	用菜单查看目录表列	145
7.3	总结	155
第八章	数据输入对话框	156
8.1	对话框简介	159
8.2	Resource Workshop	160
8.2.1	为何要 Dialog Editor	160
8.2.2	使用 BorlandDialog Editor	160
8.2.3	创建一个对话框	162
8.2.4	查看.DLG 文件	163
8.3	为各种需要创建对话框	165
8.3.1	创建一个简单的 About 对话框	165
8.3.2	用一个对话框改变图形形状	172
8.3.3	通过对话框输入文本	183
8.3.4	用对话框输入整型数	191
8.3.5	用对话框输入实型的数字	201
8.4	创建消息框	209
第九章	字体的使用	216
9.1	字体的结构和定义	216
9.1.1	逻辑字体常数	216
9.1.2	TEXTMETRIC 结构	218
9.1.3	LOGFONT 结构	219
9.1.4	字符单元	219
9.2	字体附加的特性	221
9.2.1	字体宽度	221
9.2.2	自动引导和压缩字距磅	221
9.2.3	OEM 与 ANSI 字符集相比较	221
9.2.4	逻辑字体与物理字体	223
9.2.5	向量字体与光栅字体	223
9.2.6	生成字体	223
9.3	字体的类型	223
9.3.1	缺省字体	223
9.3.2	打印机字体和显示字体	223
9.3.3	Custom 字体	224
9.4	字体映射方法	224

9.5 字体编辑器的使用	224
9.5.1 如何装入字体编辑器	225
9.5.2 基本的字体编辑器窗口	225
9.5.3 如何改变字体头	226
9.5.4 如何定制字体	227
9.5.5 如何存储 Custom 字体设计	229
9.5.6 如何生成一个字体源文件	229
9.6 字体程序	230
9.6.1 CreateFont 函数	230
9.6.2 CreateFontIndirect 函数	231
9.6.3 FONT1 程序	231
9.6.4 FONT1.MAK 文件	234
9.6.5 FONT1.DEF 文件	235
9.6.6 FONT1.CPP 文件	235
9.6.7 FONT2 程序	236
9.6.8 FONT2.MAK 和 FONT2.DEF 文件	240
9.6.9 FONT2.CPP 文件	240
9.6.10 FONT3 程序	240
9.6.11 FONT3.MAK 及 FONT3.DEF 文件	244
9.6.12 FONT3.CPP 文件	244
9.6.13 FONT4 程序	244
9.6.14 FONT4.MAK 及 FONT4.DEF 文件	248
9.6.15 FONT4.CPP 文件	248
9.7 更好的字体	249
第十章 图形概念和绘图元语	250
10.1 图形设备接口	250
10.1.1 GDI 的用途	250
10.1.2 象素操作	250
10.1.3 设备信息	250
10.1.4 设备场境句柄	254
10.1.5 映射方式	254
10.2 GDI 绘图元语	255
10.2.1 图形元语	255
10.2.2 GDI 绘图元语的使用	260
10.2.3 简单的条形图	264
10.3 GDI 工具	268
10.3.1 画笔	269
10.3.2 画刷	270
10.3.3 简单的条形图再述	271

10.3.4	颜色	275
10.3.5	位图	285
第十一章	科学图和数学图的绘制	290
11.1	正弦波	290
11.2	衰减正弦波	294
11.3	一个傅立叶级数	300
第十二章	饼形图、条形图和线图的设计	311
12.1	调色板管理程序	311
12.2	饼形图	313
12.2.1	PIE.MAK 文件和 PIE.DEF 文件	322
12.2.2	PIE.H 头文件	322
12.2.3	PIE.RC 资源文件	322
12.2.4	PIE.CPP 程序	322
12.3	条形图	325
12.3.1	BAR.MAK 文件和 BAR.DEF 文件	337
12.3.2	BAR.H 头文件	338
12.3.3	BAR.RC 资源文件	338
12.3.4	BAR.CPP 程序	338
12.4	线图	340
12.4.1	LINE.MAK 文件和 LINE.DEF 文件	353
12.4.2	LINE.H 头文件	354
12.4.3	LINE.RC 资源文件	354
12.4.4	LINE.CPP 程序	354
12.5	关于三大程序的变化	356
第十三章	专用应用程序:带多媒体声音的草图、动画和屏幕保存程序	358
13.1	草图绘制:带多媒体声音的鼠标速写	358
13.1.1	SKCH 文件	367
13.1.2	SKCH.CPP 应用程序代码	368
13.2	动画:火车 1	369
13.3	动画:带多媒体声音的火车 B	374
13.4	重要的是实践	381
第十四章	了解 Borland 的 ObjectWindows 以便开发面向对象的 Windows 程序	382
14.1	三个重要的面向对象特征	382
14.1.1	抽象	382
14.1.2	封装	382
14.1.3	消息响应	383
14.2	ObjectWindows 对象	383
14.3	一个简单的 ObjectWindows 应用程序——BSOWA.CPP	385
14.4	从 BSOWA.CPP 应用程序中建立应用程序	393

14.4.1	如何画一个数学曲线	393
14.4.2	使用 Arial True Type 字体	396
14.4.3	旋转一个 Times New Roman True Type 字体	399
14.5	更高级的工作	404
第十五章	用资源文件开发 Borland C++ Object Windows 的应用程序	405
15.1	DRAW: 用户图标、光标、菜单和一组键盘加速键的开发	406
15.1.1	DRAW.DEF 和 DRAW.H 文件	411
15.1.2	DRAW.ICO 图标	411
15.1.3	DRAW.CUR 光标	411
15.1.4	DRAW.RC 菜单和键盘加速键	412
15.1.5	DRAW.CPP 应用程序代码	413
15.1.6	DRAW 应用程序的执行	415
15.2	PIE: 简报显示图形程序, 带用户图标、光标、菜单及对话框	415
15.2.1	PIE.DEF 和 PIE.H 文件	424
15.2.2	PIE.ICO 和 PIE.CUR 图标和光标	424
15.2.3	C5PIE.RC 菜单和对话框资源文件	424
15.2.4	PIE.CPP 应用程序代码	426
15.2.5	执行 PIE 应用程序	431
15.3	EDIT: 一个增强型文本编辑器	431
15.3.1	EDIT.DEF 和 EDIT.H 文件	437
15.3.2	EDIT.ICO 图标	437
15.3.3	EDIT.RC 资源文件	437
15.3.4	EDIT.CPP 应用程序代码	438
15.3.5	执行 EDIT 应用程序	441
第十六章	开发 32 位 Windows NT 程序	442
16.1	Windows NT 的不同之处	442
16.1.1	Windows NT 数据类型和结构	442
16.1.2	特殊句柄	443
16.2	一个简单的应用程序: ntswp	444
16.2.1	对 ntswp.c 的进一步检查	446
16.2.2	WinMain 函数	447
16.2.3	窗口类登记	447
16.2.4	创建一个窗口	449
16.2.5	显示并更新窗口	450
16.2.6	消息循环	450
16.2.7	窗口函数	451
16.2.8	WM_PAINT 消息	452
16.2.9	WM_DESTROY 消息	453
16.2.10	DefWindowsProc	453

16.3	展望	453
第十七章	资源与 32 位 Windows NT 程序	454
17.1	利用资源的应用程序	454
附录 A	API 函数和 ObjectWindows 类	467
附录 B	IDE 工程管理器	538
B.1	什么是工程管理器	538
B.2	建立一个工程	539
B.2.1	建立一个多任务工程	541
B.2.2	转换旧的工程	542
B.2.3	将工程转化为制作文件(makefiles)	542
B.2.4	改变工程视图	543
B.3	建立一个工程	544
B.4	编辑工程树	546
B.4.1	使用 TargetExpert 编辑任务属性	546
B.4.2	编辑节点属性	547
B.4.3	增加或删除一个节点	548
B.4.4	增加或删除任务	550
B.4.5	移动节点和任务	550
B.4.6	拷贝节点	551
B.5	使用 Source Pool	552
B.6	设置工程选项	553
B.6.1	Local Override	554
B.6.2	使用 Style Sheet	555
B.6.3	将一个 Style Sheet 与一个节点相连	555
B.6.4	生成一个 Style Sheet	555
B.6.5	编辑 Style Sheet	558
B.6.6	共享 Style Sheet	560
B.6.7	查看工程中的选项	561
B.7	转换器	562
B.7.1	安装一个转换器	562
B.7.2	使用 SpeedMenu 中的 Special 命令	565
B.7.3	安装观察器和工具	565
附录 C	Windows.h 头文件	568
附录 D	移植 3.1 OWL 1.0 程序到 4.0	698
D.1	与 Borland C++ 4.0 一起使用对象基类库	698
D.2	与 Borland C++ 4.0 一起使用 OWL 1.0	698
D.2.1	重建 OWL 库	698
D.2.2	使用新的 OWL 1.0 库	699
D.2.3	新的 OWL 1.0 MAKEFILE 的内容	700

第一章 Windows 和 Windows NT 介绍

Borland C++ 编译器以及相关的 Windows 和 Windows NT 的开发工具为 C/C++ 程序员编辑、连接 Microsoft Windows 和 Windows NT 程序提供了很大方便。用户对 Microsoft Windows NT 程序设计感兴趣的主要原因是 Microsoft 成功地基于图形的操作环境。Windows 提供了按钮式控制方式和下拉式菜单。

本书中, Windows 3.1 术语适用于只同基于 OS 环境的 Windows 有关的概念, Windows NT 术语适用于只同基于 Windows NT 环境有关的概念。相对于环境, Windows 术语的程序设计概念都会用到。

本书将集中精力介绍怎样使用 Borland C++ 编译器在 Windows 3.1 和 Windows NT 环境中开发程序。

本章目的是对有关 Windows 3.1 和 Windows NT 的重要概念做说明。

针对 16 位的 Windows 环境而编写的应用程序, 可以在 Windows 3.1、Windows NT 和 Windows 95 环境下运行。针对 Windows NT 环境编写的 32 位应用程序只能在 Windows NT 和 Windows 95 下运行。许多标准的 DOS 程序可以从命令提示符环境, 移植到这两种环境下运行。

如果要针对 16 位的 Windows 3.1 开发应用程序, 必须要有 Borland C++ 3.1 的编辑器或更新版本, 还要有有关 Windows 的工具。

针对 32 位的 Windows NT 环境开发应用程序, 应该具备 Borland C++ 4.0 的编译器和有关 Windows 的工具。

1.1 什么是 Windows

Windows 是一个基于图形的多任务窗口环境, 它可以使针对 Windows 编写的程序有始终如一的外形和命令结构。由于有这一功能, 即使是新的程序员也能轻易学会。

Windows 提供了几个内建的例程, 可以使下拉式菜单、滚动条、对话框、图标和其它的友好图形界面功能轻易完成。从 Windows 3.0 到 Windows NT, 程序员可以使用新的对话控制、菜单类型和“自画控制”。通过利用大量图形程序设计语言, 程序员可以轻易用不同的字体和间距格式化并输出文本。

Windows 将视频显示器、键盘、鼠标、打印机、串行端口和系统时钟按独立设备方式处理。这种处理方式可以使相同的程序在不同的硬件配置上运行一致。

1.2 Windows 的功能

Windows 操作环境同 MS-DOS 环境相比, 前者为用户和程序员提供了更多的有利条件。Windows 有三个主要功能: 一个是面向图形的用户界面; 一个是多任务功能; 另一个是硬件独

立性。单独说这三项功能没有一项是新的,但是将其合并到一个单独的微机操作环境中,就成为一种新思想。

1.2.1 标准的用户界面

在 Windows 提供的三项主要功能中,面向图形的用户界面最引人注目,对用户来说当然是最重要的。用户界面用与其一致的图形来表示驱动器、文件、子目录和许多操作系统命令和行为。图 1.1 显示了一典型 Windows 窗口的外观。通过标题条确定程序,通过点和单击鼠标可以在程序菜单中访问许多基本的文件操作函数。多数 Windows 程序都提供了键盘界面和鼠标界面。尽管 Windows 程序中的多数函数都能通过键盘控制,但鼠标控制对很多任务来说更为简单。

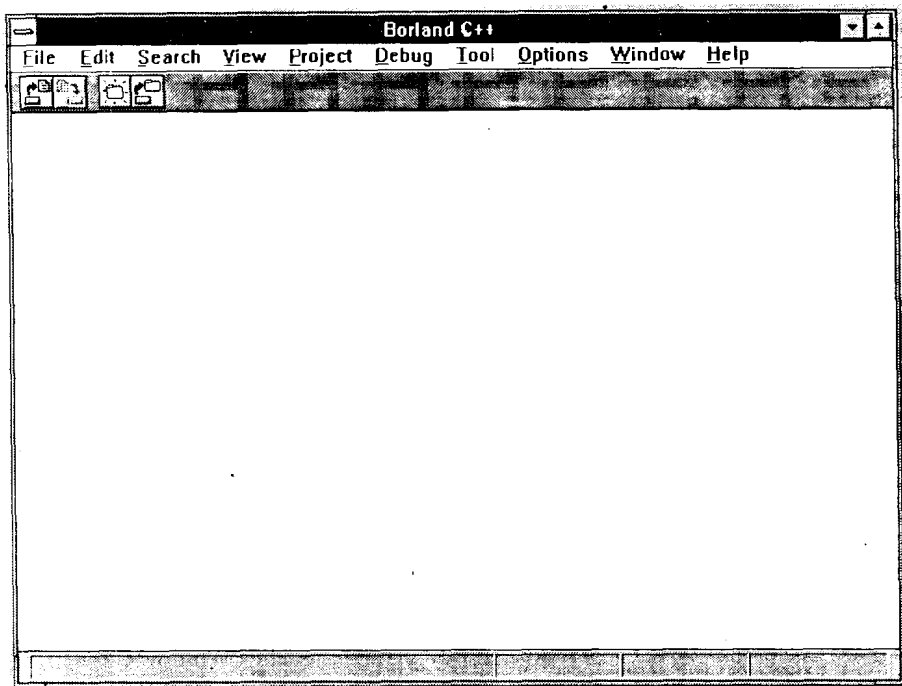


图 1.1 一个典型的 Windows 应用程序窗口

1.2.2 多任务

一个多任务操作环境允许用户同时运行几个程序或同一个程序的几个实例。现在许多用户仍然想知道多任务在微型计算机上有没有必要。图 1.2 显示了几个 Windows 程序。每个程序在屏幕上占据了一个矩形窗口。任何时候用户都能在屏幕上移动窗口,在不同的程序之间切换,改变窗口的大小,从一个窗口到另一个窗口交换信息。

1.2.3 内存管理

在 Windows 环境中,内存是一项重要的共享资源。当同时运行程序的数目超过一个时,每

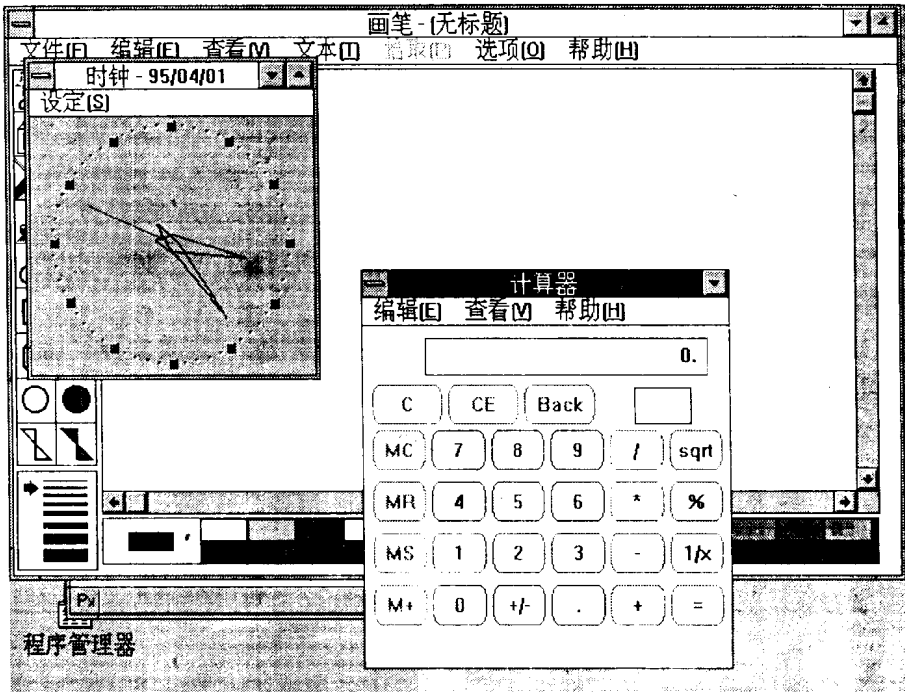


图 1.2 在 Windows 环境下多任务的几个应用程序

个程序都必须协作共享内存，目的是不浪费储备。当新程序开始运行旧程序终止时，内存可能会分裂。Windows 能够通过系统在内存中移动代码块和数据块来统一自由内存空间。

1.2.4 队列输入

另一项重要的共享资源方式是从键盘和鼠标上输入。正因为如此，不应该用 C 和 C++ 中的许多输入输出语句，例如 `getchar`，而用 Windows 中的具体函数调用代替。在 Windows 环境下，一个应用程序不能利用明确的调用从键盘或鼠标输入设备上读取信息。相反，Windows 是按系统顺序从键盘、鼠标和计时器上接受所有的输入。队列的任务是将输入信息从系统行列拷贝到程序行列，使其改道在合适的程序中。此时，当程序准备接受输入时，程序从其本身的队列中读取并向正确的窗口分发一个消息。

1.2.5 消息

多任务环境中传播数据的主要手段是 Windows 消息系统。从程序的观点看，消息以通知的形式看到，其中一些有意义的消息已经存在，它们关联(也可不)一项具体的行为。这些事件可由用户自己启动，例如单击或移动鼠标，改变窗口的大小，或选择菜单。消息也可以由程序本身产生。比如，一个基于图形的电子表格由于需要更新一个显示条图而应该进行循环。这时程序将向其自身发出一个“更新”窗口的消息。

Windows 本身也能产生消息，遇到“关闭过程”时产生。在这种情况下，Windows 将关闭的意图通知给每一个程序。最后，会从设备监视程序中产生一个额外的消息源，它指明一个重要

的化学过程已经到达了指定温度。不考虑消息源时,用户的程序必须有适当的行为。

1.2.6 设备的独立性

Windows 提供的另一项功能是硬件设备的独立性。Windows 将程序员从必须考虑监视器、打印机等不同种类的现代输入设备中解放出来。在 Windows 的环境中,每个设备——不管是视频显示器、打印机还是其它——都只需写一次,不必每个软件公司都去编写自己完整的集合,因为生产硬件的公司对自己生产的系统已经编写了驱动程序。Windows 中有许多驱动程序,其它的可以从制造商那里得到。

用户的程序是同 Windows 打交道,而不是其它的具体设备,不需要知道打印机连接的是什么。比如程序通知要画一个实心矩形,Windows 就会考虑如何完成这项任务。同样的原因,每个设备驱动程序将通过 Windows 中的一个程序来进行工作。

1.2.7 动态连接库

大多数 Windows 的功能是由新动态连接库提供的,称为 DLL。通过提供有利而又灵活的图形用户界面增强了基本操作系统。动态连接库含有预定义函数。将其装入时,它同应用程序动态地连接,而不是在 EXE 文件生成时静态地连接。

动态库的概念并非起源于 Windows。例如,对 C 编译器来说,要想实现不同系统的标准功能,对库的依赖性就很强。连接器将运行的库函数,例如 `getchar` 和 `printf`,复制到程序的执行文件中。函数库使程序员避免了把对象读入字符或格式化输出一类的一般操作都重建一个新过程的工作。程序员可以建立自己的库来包含附加的功能,例如改变字体和对齐文本。把可利用的函数当成通用工具以消除多余的设计。

当一个 Windows 程序调用一个 Windows 函数时,编译器必须为该函数的调用生成一段机器代码,该函数位于一个 Windows 库中的一代码段上。这也存在一个问题,因为在 Windows 程序中实际运行之前,Windows 函数的地址还不知道。

解决这个问题的方法用户已很熟悉——即迟后联编或动态连接。连接器允许用户调用的函数在连接时不必完全确定。只有当程序装入内存准备运行时,远程函数调用才确定。

1.3 Windows 的特征

由于使用了 Windows 3.0 版本,大大提高了计算机的运行、内存管理和多任务功能,还有许多其它的主要变化:

- 直接访问扩充内存
- 完整的内存管理
- 调色板管理程序能使一个应用程序充分利用设备的彩色功能。
- 与设备无关的彩色图标和位图
- 与设备无关的光标和位图,能从程序提供的一套预先定义的图象中自动选择合适的指定设备的图象。
- 一个增强的用户界面,能允许应用程序使用新的对话框控制、菜单类型、“自画”控制和选择更多种类的字体系。

- 一个适应性更强的安装过程。
- Windows 帮助编译器。

1.4 Windows 的概念和术语

可以将有关 Windows 的概念和术语分成两种主要类型:Windows 中可见的功能,比如菜单,图标等等;另一种是屏幕后面的操作,比如消息,函数访问等等。为使 Windows 程序开发人员之间能有效地交流,对 Windows 中的所有性能都给出一个名字和有关的使用方法。本节将介绍开发 Windows 程序所需的最少用语。

1.4.1 一个窗口的定义

对用户来说,一个 Windows 窗口就是显示器上的一个矩形部分,同时带有一个与程序无关的、始终如一的外形,这就是用户和程序产生的窗口之间的可见界面。对一个应用程序来说,窗口就是在屏幕上由这个程序控制创建并控制的一块矩形区域。该程序创建和控制主窗口中的所有功能,包括它的大小和形状。当用户启动一个程序时,就创建了一个窗口。用户每次单击一个窗口选项,程序就会做出相应的反应。关闭一个窗口,就终止了一个程序的运行。就象这种可视用户/应用程序信息交换一样,窗口在表示 Windows 系统基本的子结构中起了重要的作用。

例如,当窗口本身表示面向图形的用户界面时,它也是多任务和硬件无关性的可视表示。正是这种运行 Windows 程序的可见重叠性,才为用户提供了真正的多任务功能。通过将屏幕分成不同的窗口,用户可以利用键盘或鼠标选择一个当前运行的程序,将输入直接对着多任务环境中某个指定的程序。然后 Windows 截取用户的输入,并在需要时分配所有必要的资源(例如处理器)。

1.4.2 可视界面

Windows 程序的某些功能和行为是相同的:边界、控制框和 About 等等。这为判断各个程序的属性提供了方便。图 1.3 图示了一个窗口的主要成分。

1.4.2.1 边界

所有的 Windows 窗口周围都有边界。边界由线组成,最后形成一个窗口。边界看起来象是从别处描绘了一个程序的屏幕视图口,但是,仔细研究几个重叠的窗口程序,就会发现,其间有很大差别。边界不只是作为屏幕真实的等级界线,同时也表示了这个活动窗口。移动鼠标指针到一条边界,按下左按钮,用户就可以改变窗口的大小。

1.4.2.2 标题条

标题条在窗口的上端显示了这个应用程序的名字。标题条位于上端,相关窗口的中间处。这对帮助记忆当前是哪个程序在运行很有用处。

1.4.2.3 控制框

Windows 程序用到了一个控制框,这是一个小方框,左上角有一短划线。在控制框内单击鼠标指针就会使窗口显示系统菜单。