

序 言

美国 Microsoft 公司开发的窗口软件 Windows,从 1981 年的构想开始,到成熟上市的 Windows 3.0,可以说是“十年磨一剑”。

1985 年 11 月发布 Windows 1.0,1987 年 11 月 Windows 2.0 问世,1990 年 5 月 22 日宣布开始销售 Windows 3.0。Windows 3.0 实际上是新一代的 DOS 操作系统,最大特点是突破了传统 DOS 640K 基本内存的限制,支持 286/386 微处理器使用虚拟地址方式扩大内存空间。对 286 微机可直接访问 16M 扩充存储器,对 386 微机最大可直接访问 4GB 虚拟存储器。Windows 3.0 真正按多任务方式工作,任务间可以通过动态数据交换功能(DDE)进行数据交换。Windows 3.0 具有丰富的图形用户接口(GUI)。

Windows 3.0 发布以来,对 Windows 环境感兴趣的人盛况空前。但 Windows 使用容易编程困难,必须在充分掌握 Windows 的编程结构后才能设计出优秀的 Windows 软件。C 语言是 Windows 编程的基础,本书介绍了编写 Windows 程序各组成部分,有助于应用程序开发人员将各部分结合为有机整体。为了充分说明各种 Windows 程序的设计技巧,本书提供了五十多个完整的程序设计实例,有些实例充分说明了各部分的工作原理。同时本书还提供了几个十分有用的实用工具程序及一些 Windows 开发工具。这些软件都已做成 PC 机软盘,可提供给用户使用。

全书共有五部分内容:第一部分(一至二章)为 Windows 概念介绍,可以全面了解 Windows 的工作原理;第二部分(三至六章)为输入设备控制,对键盘、鼠标、计时器以及子窗口控制均作了详细说明;第三部分(七至十章)为系统配置资源的应用,对内存管理、图标、光标、菜单及对话框分别给出工作方式和应用方法;第四部分(十一至十五章)为图形设备接口和图形系统应用,对 GDI、绘图、图元文件、文本、字模打印等工作机制分别作了详细的原理和应用的说明;第五部分(十六至十九章)为内部数据交换及链接的方式,对剪接系统、动态数据交换(DDE)、多文件接口(MDI)及动态链接库(DLL)作了详尽的原理和使用的说明。

总之,本书为程序开发人员掌握了解 Windows 内部机制,在 Windows 环境下开发应用系统提供了良好的指南。

当读者使用书中所提供的源程序、使用编译程序或编写自己的 Windows 程序时,须具备以下软件:

- Microsoft Windows 3.0
- Microsoft Windows 软件开发工具(3.0 版本)
- Microsoft C 开发系统(Microsoft C 6.0)

一般常用的 C 编译程序可以编译某些 Windows 程序,但多数 C 编译程序不能全部胜任对 Windows C 程序的编译,需要使用 Microsoft C 编译器完成。

运行 Windows 和 Windows 软件开发工具需要以下硬件支持环境:

- 80286/80386 以上的 IBM PC 及其兼容机、20M 硬盘、640K 内存,并运行 MS-DOS 3.1 以上操作系统版本。扩展内存 2M 的 80386 机器更为理想。
- 使用与 IBM VGA 完全兼容的或更理想的图形卡。

- 使用鼠标器。Windows 多数程序可不用鼠标器,但本书提供的一些程序使用鼠标器。

Windows 3.0 版本是一个较成熟的版本,当然同其它软件一样,也还存在许多不尽人意之处,如安装启动过程较烦,色彩的丰富性和文件管理等一些方面均存在一些不足。Microsoft 公司已致力于 Windows 的开发,今后必然会作出新的改进,使版本升级。

但我们应明确相信,Windows 软件结构的基本框架和作用机制已定,今后更新的 Windows 版本编程原理上不会有本质差别。

我们的目标是用有限的代价,带给您无限的收益!

编者

目 录

序言	(1)
第一章 何谓 Windows 程序	(1)
1.1 Windows 简史	(2)
1.2 从用户观点看 Windows	(3)
1.2.1 图形用户接口(GUI)	(3)
1.2.2 GUI 概念和原理	(4)
1.2.3 一致的用户接口	(4)
1.2.4 多任务的优点	(6)
1.2.5 内存管理	(7)
1.2.6 与设备无关的图形接口	(8)
1.2.7 MS-DOS 应用程序	(9)
1.3 从程序员观点看 Windows	(10)
1.3.1 Windows 和 MS-DOS	(10)
1.3.2 Windows 的任务	(11)
1.3.3 函数调用	(11)
1.3.4 动态链接	(12)
1.3.5 面向对象的程序设计	(14)
1.3.6 消息驱动的体系结构	(14)
1.3.7 窗口过程	(15)
1.4 第一个 Windows 程序设计	(16)
1.4.1 程序的问题	(17)
1.4.2 HELLOWIN 文件	(18)
1.4.3 MAKE 文件	(23)

1.4.4	C 源程序文件	(26)
1.4.5	Windows 函数调用	(26)
1.4.6	大写标识符	(27)
1.4.7	新数据类型	(28)
1.4.8	获取句柄	(30)
1.4.9	匈牙利表示法	(30)
1.4.10	程序入口点	(32)
1.4.11	注册窗口类	(33)
1.4.12	建立窗口	(37)
1.4.13	显示窗口	(39)
1.4.14	消息循环	(40)
1.4.15	窗口过程	(43)
1.4.16	处理消息	(44)
1.4.17	WM_PAINT 消息	(45)
1.4.18	WM_DESTROY 消息	(47)
1.4.19	模块定义文件	(48)
1.5	Windows 程序设计	(49)
1.5.1	不同的调用方式	(50)
1.5.2	排队和非排队消息	(52)
1.5.3	非抢先多任务	(54)
1.5.4	学习的过程	(55)
第二章	文本画面	(57)
2.1	绘图和重画	(58)
2.1.1	WM_PAINT 消息	(58)
2.1.2	有效矩形和无效矩形	(59)
2.2	GDI 简介	(60)
2.2.1	设备环境	(61)
2.2.2	获取环境句柄(方法一)	(62)
2.2.3	绘图信息结构	(63)

2.2.4	获取环境句柄(方法二)	(66)
2.2.5	TextOut(细节)	(66)
2.2.6	系统字模	(68)
2.2.7	字符尺寸	(69)
2.2.8	文本点阵(细节)	(70)
2.2.9	格式文本	(70)
2.2.10	示范程序	(73)
2.2.11	SYSMETS1.C 窗口过程	(81)
2.2.12	空间不够	(83)
2.2.13	工作区尺寸	(84)
2.3	滚动条	(85)
2.3.1	滚动条的范围和位置	(87)
2.3.2	滚动条消息	(88)
2.3.3	滚动条程序 SYSMETS	(90)
2.3.4	组织绘图程序	(97)
2.3.5	建立最佳的滚动条	(99)
2.3.6	无鼠标操作	(108)
第三章	键盘	(109)
3.1	键盘基础知识	(109)
3.1.1	键盘驱动程序	(110)
3.1.2	舍弃某些键盘消息	(111)
3.1.3	谁是输入点	(112)
3.1.4	击键和字符	(113)
3.2	击键消息	(114)
3.2.1	系统击键与非系统击键	(114)
3.2.2	lParam 变量	(116)
3.2.3	虚拟键码	(118)
3.2.4	Shift 状态	(122)
3.2.5	使用击键消息	(123)

3.3	在 SYSMETS 中增加键盘接口	(124)
3.3.1	增加 WM_KEYDOWN 逻辑	(125)
3.3.2	发送消息	(127)
3.4	字符消息	(138)
3.4.1	WM_CHAR 消息	(141)
3.4.2	死字符消息	(142)
3.5	浏览键盘消息	(143)
3.6	插入记号(不是光标)	(152)
3.6.1	插入记号函数	(152)
3.6.2	TYPE 程序	(154)
3.7	WINDOWS 字符集	(165)
3.7.1	OEM 字符集	(166)
3.7.2	ANSI 字符集	(168)
3.7.3	OEM、ANSI 及字模	(168)
3.8	国际化问题	(169)
3.8.1	使用字符集	(170)
3.8.2	与 MS-DOS 对话	(171)
3.8.3	使用数字小键盘	(173)
第四章	鼠标	(174)
4.1	鼠标基础知识	(174)
4.1.1	快速定义	(174)
4.2	工作区的鼠标消息	(175)
4.2.1	简单鼠标处理举例	(177)
4.2.2	POINT、RECT 和 lParam	(184)
4.2.3	对 Shift 键的处理	(186)
4.2.4	鼠标的双选	(187)
4.3	非工作区的鼠标消息	(188)
4.3.1	击中测试消息	(190)

4.3.2	消息来源于消息	(191)
4.4	程序中的击中测试	(192)
4.4.1	一个假设的例子	(193)
4.4.2	程序举例	(194)
4.4.3	用键盘来模拟鼠标	(199)
4.4.4	在 CHECKER 中增加键盘接口	(202)
4.4.5	使用子窗口代替击中测试	(209)
4.4.6	CHECKER 中的子窗口	(210)
4.5	捕获鼠标	(219)
4.5.1	BLOWUP1 程序	(219)
4.5.2	改变鼠标光标形状	(225)
4.5.3	StretchBlt 调用	(227)
4.5.4	画出捕获的块	(227)
第五章	计时器	(229)
5.1	计时器基础知识	(231)
5.1.1	SYSTEM.DRV 和 Windows 计时器	(231)
5.1.2	计时消息不是异步的	(233)
5.2	使用计时器的三种方法	(234)
5.2.1	第一种方法	(235)
5.2.2	无计时器可用时如何处理	(236)
5.2.3	程序举例	(239)
5.2.4	第二种方法	(245)
5.2.5	程序举例	(248)
5.2.6	正确使用反调用函数	(252)
5.2.7	第三种方法	(255)
5.3	计时器用于状态报告	(255)
5.3.1	有创造性地使用图标	(261)
5.3.2	强制变成图标	(262)

5.3.3	保持图标为图标	(263)
5.3.4	计算可用内存	(263)
5.3.5	使用浮点运算	(264)
5.4	使用计时器作时钟	(264)
5.4.1	对弹出窗口定位及给出大小	(272)
5.4.2	获得日期和时间	(272)
5.4.3	国际化	(272)
5.5	Windows 的标准时间	(274)
第六章	子窗口控制	(276)
6.1	按钮类	(278)
6.1.1	建立子窗口	(284)
6.1.2	子窗口与父窗口对话	(286)
6.1.3	父窗口与子窗口对话	(287)
6.1.4	PUSH 按钮	(288)
6.1.5	检查框(check box)	(290)
6.1.6	收音机按钮(radio button)	(291)
6.1.7	GROUP 框	(292)
6.1.8	用户定义按钮	(292)
6.1.9	改变按钮正文	(292)
6.1.10	可见的与可用的按钮	(293)
6.1.11	按钮与输入点	(294)
6.2	控制及颜色	(295)
6.2.1	系统颜色	(296)
6.2.2	按钮颜色	(298)
6.2.3	WM_CTLCOLOR 消息	(299)
6.3	静态类(static class)	(302)
6.4	滚动条类(scrollbar class)	(304)
6.4.1	COLORS1 程序	(306)

6.4.2	自动键盘接口	(316)
6.4.3	窗口子类(windows subclassing)	(317)
6.4.4	加背景色	(318)
6.4.5	对滚动条上色	(320)
6.4.6	处理多种事例	(321)
6.4.7	将 COLORS1 作为图标	(322)
6.5	编辑类(edit class)	(322)
6.5.1	编辑类模式	(327)
6.5.2	编辑控制通知单(Edit Control Notification).....	(328)
6.5.3	使用编辑控制	(329)
6.5.4	向编辑控制发送的消息	(329)
6.6	目录框类(listbox class)	(331)
6.6.1	目录框样式	(332)
6.6.2	在目录框中放入字符串	(333)
6.6.3	选择和摘取输入	(335)
6.6.4	从目录框接受消息	(337)
6.6.5	目录框应用程序举例	(338)
6.6.6	显示文件目录	(344)
6.6.7	Windows 的头	(346)
6.6.8	2KB 的空间浪费	(355)
第七章	内存管理	(356)
7.1	INTEL 存储器的段模式	(358)
7.2	Windows 的内存组织	(359)
7.2.1	固定段与可移动段	(361)
7.2.2	可淘汰内存	(362)
7.2.3	全局内存表	(364)
7.2.4	局部内存	(365)
7.3	代码段和数据段	(367)
7.3.1	小型、中型、紧凑性、大型和巨型存储模式	(367)

7.3.2	多代码段	(369)
7.3.3	紧凑存储模式和大存储模式	(371)
7.3.4	避免移动	(373)
7.3.5	程序段属性	(374)
7.4	Windows 如何移动与重装入程序段	(377)
7.4.1	远程函数的特殊处理	(378)
7.4.2	Windows 执行程序	(382)
7.4.3	MakeProcInstance 的作用	(384)
7.4.4	动态链接库带来的差异	(386)
7.4.5	栈探测	(387)
7.4.6	扩充存储器	(388)
7.4.7	保护方式	(390)
7.5	在程序内部申请空间	(391)
7.5.1	存储块锁定	(392)
7.5.2	一个快速示例	(393)
7.5.3	全局内存函数	(394)
7.5.4	其它的内存函数	(397)
7.5.5	使用可淘汰全局内存	(399)
7.5.6	巨型全局内存块	(400)
7.5.7	局部内存分配	(403)
7.5.8	其它的局部内存函数	(406)
7.5.9	锁定自己的数据段	(407)
7.5.10	内存分配的“捷径”	(408)
7.5.11	使用 C 的内存函数	(410)
7.5.12	在保护方式下运行	(411)
第八章	图标、光标、位图和字符串	(413)
8.1	资源编译	(414)
8.2	图标与光标	(417)
8.2.1	SDKPAINT 工具	(422)

8.2.2	取得图标句柄	(426)
8.2.3	在程序中使用图标	(428)
8.2.4	使用候选光标的语句	(429)
8.3	资源与内存	(430)
8.3.1	位图:像素点图画	(431)
8.3.2	使用位图和画刷	(432)
8.4	字符串	(438)
8.4.1	使用字符串资源	(438)
8.4.2	在消息框中使用字符串	(439)
8.4.3	字符串与内存空间	(441)
8.4.4	用户自定义资源	(442)
第九章	菜单与加速键	(456)
9.1	菜单	(456)
9.1.1	菜单结构	(457)
9.1.2	菜单样本	(458)
9.1.3	应用程序中如何调用菜单	(462)
9.1.4	菜单和消息	(464)
9.1.5	范例	(468)
9.1.6	菜单格式	(478)
9.1.7	定义菜单的另一种格式	(479)
9.1.8	定义菜单的第三种途径	(481)
9.1.9	浮动的弹出式菜单	(481)
9.1.10	系统菜单的使用	(490)
9.1.11	修改菜单	(496)
9.1.12	其它菜单命令	(496)
9.1.13	一个非正规的菜单	(498)
9.2	菜单中使用位图	(506)
9.2.1	利用位图作为菜单项的两种方法	(520)
9.2.2	内存设备环境	(521)

9.2.3	用程序建立位图	(522)
9.2.4	位图的变形调整	(523)
9.2.5	完整菜单的形成	(525)
9.2.6	增加键盘操作界面	(527)
9.3	加速键	(528)
9.3.1	为什么要使用加速键	(528)
9.3.2	加速键定义的一些规则	(529)
9.3.3	加速键表	(530)
9.3.4	装入加速键表	(533)
9.3.5	加速键的转换	(533)
9.3.6	接收加速键消息	(535)
9.3.7	用菜单和加速键编写的 POPPAD 程序	(536)
9.3.8	菜单项能否执行的判别	(546)
9.3.9	菜单项的处理	(547)
第十章	会话框	(552)
10.1	标准会话框	(553)
10.1.1	建立“About”会话框	(553)
10.1.2	会话框样本	(559)
10.1.3	会话框过程	(562)
10.1.4	会话框过程的卸出	(564)
10.1.5	会话框的启动	(564)
10.1.6	进一步讨论会话框样式	(566)
10.1.7	其它会话框控制	(568)
10.1.8	一个稍复杂的会话框	(571)
10.1.9	会话框中控制的处理	(581)
10.1.10	OK 和 Cancel 按钮	(585)
10.1.11	控制的分组	(587)
10.1.12	会话框中绘图	(590)
10.1.13	在会话框中使用其它函数	(591)

10.1.14	自定义控制	(591)
10.2	消息框	(602)
10.2.1	Assertion 消息框	(604)
10.2.2	用消息框显示有关消息	(606)
10.3	POPPAD 程序的文件菜单项功能	(607)
10.3.1	OpenFile 函数	(607)
10.3.2	文件输入输出的两种方式	(611)
10.3.3	Open 和 Save 菜单项的会话框	(615)
10.3.4	DlgDirList 函数和 DlgDirSelect 函数	(627)
10.3.5	获取合法文件名	(629)
10.3.6	新版的 POPPAD 程序	(630)
10.4	非标准会话框	(652)
10.4.1	标准会话框和非标准会话框之间的区别	(653)
10.4.2	新版的 COLORS 程序	(655)
10.4.3	HEXCALC 程序是窗口还是会话框	(664)
10.4.4	控制标识符的有效使用	(675)
10.5	会话框实用程序的使用	(676)
第十一章	GDI 引言	(681)
11.1	GDI 原理	(681)
11.2	设备环境(DC)	(684)
11.2.1	获取设备环境句柄	(684)
11.2.2	获取设备环境信息	(687)
11.2.3	DEVCAPS1 程序	(688)
11.2.4	设备尺寸	(707)
11.2.5	查找色彩信息	(709)
11.2.6	设备环境属性	(711)
11.2.7	保存设备环境	(712)
11.3	映射方式	(715)

11.3.1	设备坐标和逻辑坐标	(717)
11.3.2	设备坐标	(718)
11.3.3	视口和窗口	(719)
11.3.4	使用 MM-TEXT 方式工作	(721)
11.3.5	“Metric”映射方式	(726)
11.3.6	“Roll Your Own”映射方式	(729)
11.3.7	WHATSIZE 程序	(737)
第十二章	作图	(745)
12.1	画点	(745)
12.2	画线	(746)
12.2.1	使用库存画笔	(749)
12.2.2	生成、选择、删除画笔	(751)
12.2.3	避免设备依赖性	(756)
12.2.4	在隙间填充	(757)
12.2.5	作图方式	(757)
12.2.6	ROP2 和色彩	(767)
12.3	绘制填充区域	(769)
12.3.1	有界图形	(771)
12.3.2	ARCS 程序	(777)
12.3.3	饼图的三角学	(784)
12.3.4	Polygon 函数和 Polygon 填充方式	(787)
12.3.5	涂刷图形内部	(789)
12.3.6	画刷和位图	(793)
12.3.7	生成和使用位图画刷	(795)
12.3.8	画刷的定位	(799)
12.4	矩形、区域和裁剪	(803)
12.4.1	对矩形进行工作	(804)
12.4.2	生成和绘制区域	(806)
12.4.3	对矩形和区域进行裁剪	(808)

12.4.4	CLOVER 程序	(809)
12.5	杂类 GDI 函数	(817)
12.6	永久绘图程序	(825)
第十三章	位图、位块传输和图元文件	(835)
13.1	旧位图格式	(837)
13.1.1	在程序中生成位图	(837)
13.1.2	单色位图格式	(840)
13.1.3	彩色位图格式	(841)
13.1.4	位图的大小	(842)
13.2	设备无关位图	(843)
13.2.1	DIB 文件	(843)
13.2.2	生成一个 DIB	(846)
13.3	内存设备环境	(848)
13.4	强有力的 BLT	(850)
13.4.1	patBlt 函数	(851)
13.4.2	块传递(BLT)坐标	(854)
13.4.3	使用 BitBlt 传递位	(856)
13.4.4	DrawBitmap 程序	(858)
13.4.5	使用不同的 ROP 代码	(860)
13.4.6	内存设备环境更有趣的应用	(863)
13.4.7	色彩转换	(869)
13.4.8	映射方式的转换	(871)
13.4.9	使用 StretchBlt 扩展位图	(871)
13.4.10	动画	(873)
13.5	图元文件	(880)
13.5.1	内存图元文件的简单应用	(881)
13.5.2	在磁盘上存放图元文件	(884)
13.5.3	使用已存在的图元文件	(886)

13.5.4	把图元文件作为源	(890)
13.5.5	查看图元文件	(896)
13.5.6	图元文件做什么、不做什么	(898)
第十四章	文本和字模	(901)
14.1	简单文本输出	(901)
14.1.1	文本输出函数	(901)
14.1.2	文本的设备环境属性	(904)
14.1.3	使用矢量字体	(905)
14.1.4	使字符串变灰	(907)
14.1.5	GrayString 的简单用法	(910)
14.1.6	不用 GrayString 而使字符串变灰	(910)
14.2	字体的背景知识	(911)
14.2.1	字模的类型	(911)
14.2.2	话题 1: 系列和字样	(913)
14.2.3	字体资源文件	(916)
14.2.4	话题 2: 关于“点”	(918)
14.2.5	为什么要使用逻辑英寸	(919)
14.2.6	话题 3: leading 和 spacing	(920)
14.2.7	逻辑“TWIPS”映射方式	(921)
14.3	创建、选择和删除逻辑字体	(923)
14.3.1	字体选择及定义	(939)
14.3.2	逻辑字体结构	(940)
14.3.3	字模端射算法	(946)
14.3.4	找出字模	(947)
14.4	枚举字体	(950)
14.5	文本格式化	(969)
14.5.1	单行文本对齐	(970)
14.5.2	段落处理	(972)