

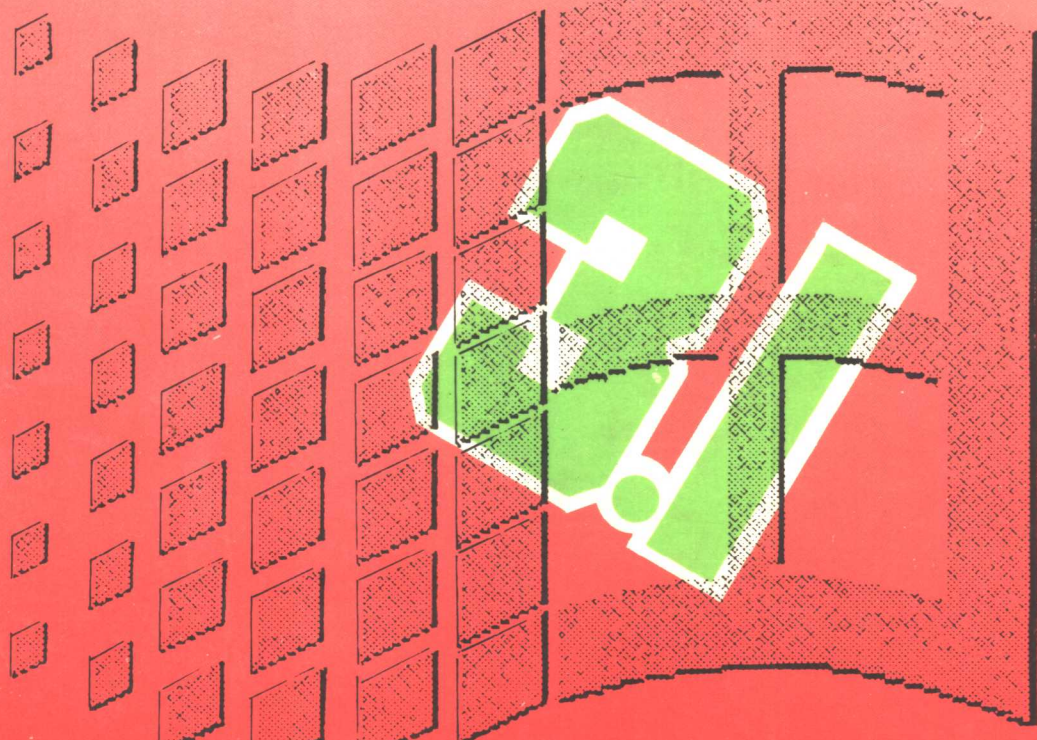
Microsoft Microsoft Microsoft

Microsoft Windows 3.1

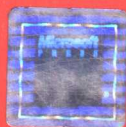
程序员参考大全(一)

—综 述

沈金发 李莉 陈建伟 李逸 译



清华大学出版社



Microsoft

Windows 3.1

程序员参考大全(一) ——综述

[美] Microsoft 公司 著
沈金发 李莉 陈建伟 李逸 译

清华大学出版社

Microsoft Windows 3.1 Programmer's Reference

Volume 1—Overview

本书由 Microsoft 公司属下的 Microsoft 出版社(Microsoft Press)出版。
版权为 Microsoft 公司所有(© Microsoft Corporation, 1987—1992)。
本书中文版由 Microsoft Press 授权© 清华大学出版社独家出版,1993。
未经出版者书面允许,不得用任何手段复制或抄袭本书部分或全部内容。

Adobe®和 PostScript®是 Adobe Systems 公司的注册商标。

Apple®和 TrueType 是 Apple Computer 公司的注册商标。

Banyan®和 VINES®是 Banyan Systems 公司的注册商标。

Hewlett-Packard®是 Hewlett-Packard Company 的注册商标。

Intel®是 Intel Corporation 的注册商标和 i486™是 Intel Corporation 的商标。

Helvetica®, Linotype®, Times®和 Times Roman®是 Linotype AG 和(或)其它附属公司的注册商标。

CodeView®, Microsoft®, MS®, MS-DOS®和 QuickC®是 Microsoft Corporation 的注册商标, QuickBasic™和 Windows™是 Microsoft Corporation 的商标。

Arial®和 Times New Roman®是 Monotype Corporation PLC 的注册商标。

NetWare®和 Novell®是 Novell 公司的注册商标。

Net/One®和 Ungermann-Bass®是 Ungermann-Bass 公司的注册商标。

Paintbrush™是 ZSoft Corporation 的商标。

(京)新登字158号

Microsoft Windows 3.1

程序员参考大全(一)——综述

沈金发 李莉 陈建伟 李逸 译

☆

清华大学出版社出版

北京 清华园

清华大学印刷厂印刷

新华书店总店科技发行所发行

☆

开本:787×1092 1/16 印张:21 字数:490千字

1993年6月第1版 1993年6月第1次印刷

印数:0001—5000

ISBN 7-302-01222-9/TP·460

定价:36.00元

引 言

《Microsoft Windows 3.1程序员参考大全(一)——综述》介绍了各种接口函数和 Microsoft Windows操作系统支持的扩展库；还包括描述 Windows 应用程序特性的应用程序注解。附录中列出了 Windows 函数的模块和库名。

第一部分“窗口管理、图形和系统服务”介绍了有关窗口管理、图形输出和系统服务的函数。窗口管理函数处理消息；创建、移动或切换窗口；产生系统输出。图形设备接口 (GDI) 函数完成与设备无关的图形操作，如在不同的输出设备上产生线、字符和位图输出。系统服务函数涉及到访问模块的数据和代码；分配和管理内存；翻译字符串以及创建、打开文件之类的操作。

第二部分“扩展库”描述了支持 Windows V3.1许多新特性的库。这些新特性包括公共对话框；简化动态数据交换 (DDE) 的管理函数、对象链接与嵌入 (OLE)；注册数据库和拖动等 Shell 增强特性；可使 Windows-hosted 工具的创建得以简化的工具帮助函数；数据解压缩函数；人为消耗系统资源并用于调试应用程序的资源紧张测试实用程序；文件安装函数；利用 80386 和 80486 处理器 32 位内存寻址能力的函数；浮点仿真函数及 Control Panel 中的屏幕节约程序。

第三部分“应用程序注解”介绍了应用程序应该用来实现一些 Windows 特性和增强功能的技术。手册的这一部分说明了如何创建 Control Panel 应用程序；如何创建和安装 File Manager 的扩展功能；如何使用 Program Manager 的动态数据交换接口；如何编制与国家、语言无关的应用程序；如何编写网络应用程序；如何在 Windows 应用程序中使用 Microsoft MS-DOS 函数；如何编写产生 Windows 前置和后续代码的编译程序；如何初始化并启动 Windows 应用程序；如何改善 Windows 应用程序的视觉效果；如何编写自装载 Windows 应用程序；以及如何与可安装驱动器进行交互。

附录中列出了每个 Windows 函数的模块名和库名。

符号约定

本手册中使用下列约定来定义语法：

约定	含 义
黑体	表示要照原样键入的词或字符，例如，资源定义语句或函数名——如 MENU 或 CreateWindows、MS-DOS 命令或命令行选择项。
斜体	表示一个位置或变量，使用时必须以实际值代替。如，语句 SetCursorPos(<i>x</i> , <i>y</i>)，就要求用实际值去替换 <i>x</i> , <i>y</i> 。
[]	可选参数。
	分隔或选择项。

(续表)

约定	含 义
...	表示前面的项可以重复。
<i>BEGIN</i>	
.	
.	表示应用程序例子的省略部分。
.	
<i>END</i>	

另外,还有些约定用来帮助你更好地理解本手册:

约 定	含 义
小号大写字符	表示键名、顺序键或组合键。如,ALT+SPACEBAR。
正常大写字母	表示文件名和路径,大部分类型、结构名(同时也是黑体)和常数。
空格	分隔代码和表示语法间隔符。

目 录

引言	IX
----------	----

第一部分 窗口管理、图形和系统服务

第1章 窗口管理	3	1.2.22 窗口创建函数	21
1.1 消息	3	1.3 显示和移动函数	21
1.1.1 消息的产生和处理	3	1.4 输入函数	22
1.1.2 翻译消息	4	1.5 硬件函数	23
1.1.3 检查消息	5	1.6 绘图	24
1.1.4 发送消息	5	1.6.1 Windows 如何管理	
1.1.5 避免消息死锁	5	显示器	24
1.1.6 消息函数	6	1.6.2 显示描述表类型	24
1.2 创建和管理窗口	7	1.6.3 显示描述表高速缓存区	26
1.2.1 窗口类	7	1.6.4 绘图次序	27
1.2.2 Windows 如何寻找		1.6.5 WM_PAINT 消息	27
窗口类	7	1.6.6 更新区域	27
1.2.3 类属	8	1.6.7 窗口背景	28
1.2.4 注册窗口类	8	1.6.8 刷对齐	28
1.2.5 共享窗口类	8	1.6.9 显示矩形区域	28
1.2.6 预定义窗口类	8	1.6.10 画图符	29
1.2.7 窗口类的元素	8	1.6.11 输出格式文本	29
1.2.8 类风格	11	1.6.12 显示灰字符	30
1.2.9 内部数据结构	12	1.6.13 非客户区显示	30
1.2.10 窗口子类化	12	1.6.14 绘画函数	31
1.2.11 重画客户区	12	1.7 对话框	31
1.2.12 类和私有显示描述表	13	1.7.1 对话框的使用	32
1.2.13 窗口过程	13	1.7.2 创建对话框	32
1.2.14 窗口风格	16	1.7.3 对话框返回值	33
1.2.15 多文本接口窗口	18	1.7.4 对话框中的控制框	33
1.2.16 标题栏	18	1.7.5 对话框的键盘接口	36
1.2.17 System 菜单	18	1.7.6 对话框函数	37
1.2.18 滚动杠	19	1.8 滚动	38
1.2.19 菜单	19	1.8.1 标准滚动杠及滚动	
1.2.20 窗口状态	20	杠控制	38
1.2.21 窗口的生命期	20	1.8.2 滚动框	38

1.8.3 滚动请求	39	2.2.2 笔的使用	56
1.8.4 处理滚动消息	39	2.2.3 指定颜色	56
1.8.5 滚动客户区	39	2.2.4 绘画工具函数	57
1.8.6 隐藏标准滚动杠	39	2.3 调色板	58
1.8.7 滚动函数	40	2.3.1 了解调色板	58
1.9 菜单函数	40	2.3.2 使用调色板	59
1.10 信息函数	41	2.3.3 调色板函数	60
1.11 系统函数	42	2.4 绘画属性	61
1.12 剪接板函数	42	2.4.1 设置颜色	61
1.13 错误函数	43	2.4.2 扩展控制	61
1.14 插入符	43	2.4.3 绘画属性函数	61
1.14.1 创建和显示插入符	43	2.5 映像模式	62
1.14.2 共享插入符	44	2.5.1 约束映像模式	62
1.14.3 插入符函数	44	2.5.2 其他映像模式	64
1.15 光标	44	2.5.3 映像函数	64
1.15.1 鼠标和光标	44	2.6 坐标函数	65
1.15.2 显示和隐藏光标	44	2.7 区域函数	66
1.15.3 定位光标	45	2.8 剪接函数	67
1.15.4 光标热点和光标限制	45	2.9 线输出	67
1.15.5 创建自定义光标	45	2.9.1 弧	68
1.15.6 光标函数	45	2.9.2 简单线	68
1.16 钩子	46	2.9.3 画线函数	68
1.16.1 过滤函数链	46	2.10 椭圆和多边形	69
1.16.2 安装过滤函数	47	2.10.1 矩形	69
1.16.3 钩子函数	47	2.10.2 边界矩形	69
1.17 特征列表	47	2.10.3 椭圆和多边形函数	69
1.17.1 使用特征列表	47	2.11 位图函数	70
1.17.2 特征函数	49	2.12 设备无关位图函数	70
1.18 矩形	49	2.13 文本函数	71
1.18.1 在 Windows 应用程序中 使用矩形	49	2.14 字体函数	72
1.18.2 矩形坐标	49	2.15 元文件	73
1.18.3 创建与使用矩形	50	2.15.1 创建元文件	73
1.18.4 矩形函数	50	2.15.2 储存元文件	74
1.19 相关内容	51	2.15.3 改变 Windows 显示元文件 的方式	74
第2章 图形设备接口	52	2.15.4 元文件函数	75
2.1 设备描述表	52	2.16 设备控制函数	75
2.1.1 访问输出设备	52	2.17 打印机函数	75
2.1.2 设备描述表属性	53	2.18 相关内容	76
2.1.3 设备描述表函数	55	第3章 系统服务	77
2.2 画图工具	55	3.1 模块管理函数	77
2.2.1 刷的使用	55	3.2 内存管理函数	77

3.3 段函数	79	3.10 通讯函数	83
3.4 操作系统中断函数	79	3.11 实用宏和函数	83
3.5 任务函数	80	3.12 文件输入输出函数	84
3.6 资源管理函数	80	3.13 调试函数	84
3.7 字符串处理函数	81	3.14 优化工具函数	85
3.8 原子管理函数	81	3.15 执行应用程序函数	85
3.9 初始化文件函数	82	3.16 相关内容	86

第二部分 扩展库

第4章 公共对话框库	89	框的对话框消息	105
4.1 使用 Color 对话框	90	4.6 定制公共对话框	106
4.1.1 Color 对话框使用的 颜色方式	91	4.6.1 恰当的和不当的 定制工作	107
4.1.2 用 Color 对话框显示 基本颜色	92	4.6.2 钩函数和用户 对话框模板	107
4.1.3 用 Color 对话框显示 用户颜色	93	4.6.3 显示定制对话框	110
4.2 使用 Font 对话框	95	4.7 在公共对话框中支持 Help 按钮	111
4.2.1 在应用程序中显示 Font 对话框	95	4.8 错误检测	112
4.3 使用 Open 和 Save As 对话框	97	4.9 相关内容	113
4.3.1 在应用程序中显示 Open 对话框	97	第5章 动态数据交换管理库	114
4.3.2 在应用程序中使用 Save As 对话框	99	5.1 基本概念	115
4.3.3 在 Open 对话框或 Save As 对 话框中监视列表框控制	101	5.1.1 客户应用程序和服务器应用 程序的交互	115
4.3.4 监视 Open 对话框或 Save AS 对话框中的文件名	101	5.1.2 事务和 DDE 回调函数	115
4.4 使用 Print 和 Print Setup 对话框	102	5.1.3 服务名、话题名 和项目名	115
4.4.1 设备驱动程序和 Print 对话框	102	5.1.4 System 话题	116
4.4.2 为默认打印机显示 Print 对话框	103	5.2 初始化	117
4.5 使用 Find 和 Replace 对话框	104	5.3 回调函数	118
4.5.1 显示 Find 对话框	104	5.4 字符串管理	119
4.5.2 显示 Replace 对话框	105	5.5 名称服务	120
4.5.3 处理 Find 或 Replace 对话 框的对话框消息	105	5.5.1 服务名注册	120
		5.5.2 服务名过滤器	121
		5.6 会话管理	121
		5.6.1 单线会话	121
		5.6.2 多线会话	124
		5.7 数据管理	126
		5.8 事务管理	128
		5.8.1 Request 事务	128

5.8.2	Poke 事务	128	6.3.13	Class Name Object 命令	160
5.8.3	Advise 事务	129	6.3.14	Links 命令	160
5.8.4	Execute 事务	130	6.3.15	关闭客户应用程序	161
5.8.5	同步和异步事务	130	6.4	服务器应用程序	161
5.8.6	事务控制	131	6.4.1	启动服务器应用程序	162
5.8.7	事务类	132	6.4.2	打开文本或对象	164
5.8.8	事务小结	132	6.4.3	服务器应用程序的 Cut 和 Copy 命令	164
5.9	错误检测	133	6.4.4	Update, Save As 和 New 命令	165
5.10	监视应用程序	133	6.4.5	关闭服务器应用程序	166
第6章	对象连接和嵌入库	137	6.5	对象处理程序	166
6.1	对象连接和嵌入的基本概念	137	6.5.1	对象处理程序的实现	167
6.1.1	复合文本	137	6.5.2	在对象处理程序中 创建对象	168
6.1.2	连接和嵌入对象	138	6.6	动态数据交换的直接使用	170
6.1.3	对象连接和嵌入 的好处	139	6.6.1	客户应用程序与动态数据 交换的直接使用	170
6.1.4	在 OLE 和 DDEML 之间 选择	140	6.6.2	服务器应用程序与动态 数据交换的直接使用	172
6.2	对象连接和嵌入中的 数据传输	142	6.6.3	会话	172
6.2.1	客户应用程序	142	6.6.4	System 话题的项目	173
6.2.2	服务器应用程序	142	6.6.5	标准项目名和 通知控制	173
6.2.3	对象处理程序	142	6.6.6	DDE 执行字符串中的 标准命令	175
6.2.4	OLE 库之间的通信	143	第7章	Shell 库	178
6.2.5	剪接板约定	143	7.1	注册数据库	178
6.2.6	注册	145	7.1.1	数据库结构	178
6.2.7	客户用户接口	147	7.1.2	注册文件的格式	180
6.2.8	服务器用户接口	149	7.1.3	类注册	181
6.2.9	对象存储格式	150	7.1.4	查询和删除数据 库条目	184
6.3	客户应用程序	152	7.2	拖动特性	185
6.3.1	启动客户应用程序	152	7.3	利用联系查找并启动 应用程序	186
6.3.2	打开复合文本	153	7.4	从可执行文件中获取图符	187
6.3.3	文本管理	153	7.5	相关内容	187
6.3.4	保存文本	153	第8章	工具帮助库	188
6.3.5	关闭文本	154	8.1	调用工具帮助函数	188
6.3.6	非同步操作	154	8.2	访问内部 Windows 列表	188
6.3.7	显示和打印对象	155			
6.3.8	打开和关闭对象	156			
6.3.9	删除对象	156			
6.3.10	客户应用程序的 Cut 和 Copy 命令	157			
6.3.11	创建对象	158			
6.3.12	Undo 命令	159			

8.2.1 查看 Windows 类列表	189	12.4 在 Windows 应用程序中使用 32比特内存	204
8.2.2 查看 Windows 模块 列表	189	12.4.1 使用32比特数据对象 ...	205
8.2.3 查看 Windows 任务 队列	189	12.4.2 在子程序库中使用32比特 代码和数据	205
8.3 获得建议性信息	189	12.4.3 在主程序中使用32比特 代码和数据	205
8.4 查看全局堆和局部堆	190	12.5 错误值	205
8.4.1 查看全局堆	190	第13章 浮点仿真库	207
8.4.2 查看局部堆	190	13.1 仿真方法	207
8.5 跟踪 Windows 堆栈	190	13.1.1 用例外处理程序 实现仿真	207
8.6 检查和修改内存内容	191	13.1.2 Windows 80x87 浮点仿真	207
8.7 安装回调函数	191	13.2 Windows 3.0版本的限制	209
8.8 控制过程执行	192	13.3 函数	209
第9章 数据解压缩库	193	13.4 结构	212
9.1 数据压缩	193	第14章 屏幕节约程序库	215
9.2 数据解压缩	193	14.1 关于屏幕节约程序	215
9.3 解压缩单个文件	194	14.2 创建屏幕节约程序	216
9.4 解压缩多个文件	194	14.2.1 处理屏幕节约程序 消息	217
9.5 从压缩文件中读字节	195	14.2.2 提供配置子程序	217
第10章 系统资源紧张测试库	196	14.2.3 创建模块定义和 资源定义文件	217
10.1 系统资源紧张测试库函数	196	14.3 安装新的屏幕节约程序	218
第11章 文件安装库	197	14.4 一个屏幕节约示范程序	218
11.1 文件安装的概念	197	14.4.1 通用说明	218
11.2 创建安装程序	197	14.4.2 消息处理	218
11.3 在文件中增加版本信息	199	14.4.3 配置对话框	220
第12章 32比特内存管理库	200	14.4.4 增加帮助功能	222
12.1 分段内存模式和平面 内存模式	200	14.4.5 输出函数	222
12.2 使用 WINMEM32.DLL 库	201	14.5 函数	222
12.3 使用32比特内存的注意事项 ...	202		
12.3.1 平面内存模式的限制 ...	203		
12.3.2 应用程序堆栈	203		
12.3.3 中断代码	204		
12.3.4 编程语言	204		

第三部分 应用程序注释

第15章 Control Panel 应用程序	233	15.2.2 初始化应用程序	236
15.1 启动 Control Panel 应用程序 ...	233	15.2.3 用户操作	237
15.2 创建 Control Panel 应用程序 ...	235	15.2.4 退出应用程序 和 DLL	237
15.2.1 创建入口点函数	235		

15.2.5	Control Panel 应用	
	程序举例	237
15.3	安装新的应用程序	238
第16章	File Manager 扩展	240
16.1	创建 File Manager 扩展	240
16.2	创建入口点函数	240
16.2.1	装入扩展	241
16.2.2	菜单选择的处理	242
16.2.3	扩展菜单的初始化	242
16.2.4	扩展菜单的更新	242
16.2.5	文件选择的处理	242
16.2.6	退出扩展 DLL	242
16.3	安装扩展	242
16.4	扩展消息	243
16.5	File Manager 扩展范例	243
16.6	增加 Undelete 命令	246
第17章	Shell 动态数据交换接口	247
17.1	PROGMAN.INI 文件	247
17.1.1	设置部分	248
17.1.2	分组部分	248
17.1.3	限制部分	248
17.2	命令字符串接口	249
17.2.1	CreateGroup	250
17.2.2	ShowGroup	250
17.2.3	DeleteGroup	251
17.2.4	Reload	251
17.2.5	AddItem	251
17.2.6	ReplaceItem	252
17.2.7	Delete Item	253
17.2.8	ExitProgman	253
17.3	查询分组信息	253
第18章	国际应用程序	254
18.1	创建国际通用应用程序	254
18.2	实现国家和语言独立性	254
18.2.1	WIN.INI 中的 国际信息	254
18.2.2	Windows 函数中的 国际信息	258
18.2.3	文件版本库的 国际使用	261
18.3	方便实现本地化	261
18.3.1	隔离可本地化信息	261
18.3.2	为字符串分配额外 的空间	261
18.3.3	处理外国语	262
第19章	网络应用程序	264
19.1	多用户共享	264
19.1.1	共享目录	264
19.1.2	共享暂存	265
19.1.3	共享文件	265
19.1.4	共享设备	265
19.2	在保护方式下调用网络软件	265
19.2.1	Microsoft Networks 和 MS-DOS 网络函数	266
19.2.2	NetBIOS 功能调用	266
19.2.3	LAN Manager Networks	266
19.2.4	Novell NetWare	267
19.2.5	Ungermann-Bass Net/One	267
19.2.6	Banyan VINES	267
第20章	Windows 应用程序与 MS-DOS 函数	268
20.1	使用 DOS 保护模式接口 函数	268
20.1.1	Windows 内核	268
20.1.2	其他应用程序 编程接口	269
20.2	MS-DOS 中断支持	269
20.2.1	不支持的 MS-DOS 中断 及功能	269
20.2.2	部分支持的 MS-DOS 中断 21h 功能	270
20.3	NetBIOS 支持	271
第21章	Windows 前导和后续代码	272
21.1	数据段初始化	272
21.1.1	输出远函数	272
21.1.2	非输出远函数	274
21.1.3	动态链接库中的输出 远函数	274
21.2	实模式下的前导代码	275
21.3	保护模式下的前导代码	276
第22章	启动 Windows 应用程序	277
22.1	启动要求	277

22.2	一个启动过程的实例	278	24.3.1	装入段	288
22.3	函数说明	279	24.3.2	重装段	288
第23章	视频技术	282	24.3.3	复位硬件	289
23.1	使用统一调色板	282	24.4	函数参考	289
23.1.1	了解系统调色板	282	第25章	可安装驱动器	293
23.1.2	创建统一调色板	283	25.1	可安装驱动器简介	293
23.2	适配不同的视频适配器 和驱动器	283	25.2	创建可安装驱动器	294
23.2.1	区分 Standard VGA 和 Super VGA	283	25.2.1	打开可安装驱动器	296
23.2.2	在不同的显示适配器上 使用统一调色板	283	25.2.2	关闭可安装驱动器	297
23.3	使用设备无关位图驱动器	283	25.2.3	配置可安装驱动器	297
23.3.1	创建驱动器显示 描述表	284	25.2.4	统计可安装驱动器 的实例	297
23.3.2	对显示设备存取位图 ...	285	25.3	更新 SYSTEM.INI 文件	297
23.3.3	修改位图	285	25.4	OEMSETUP.INF 文件 的内容	298
23.3.4	创建驱动设备描述表 ...	285	25.5	Drivers Control Panel 应用 程序	299
第24章	自装载 Windows 应用程序	287	25.5.1	安装驱动器	300
24.1	装载函数	287	25.5.2	通过 Drivers Control Panel 应用程序使用驱动器 ...	300
24.2	装载数据表	287	25.6	建立定制配置应用程序	301
24.3	装载代码	288	附录	Windows 函数的模块与库名	302

第一部分

窗口管理、图形和系统服务

第 1 章

窗 口 管 理

本章描述了 Microsoft Windows 操作系统中处理消息;创建、移动或更换窗口;或产生系统输出的一系列函数。这些函数组成了窗口管理程序接口。

1.1 消 息

消息是一个应用程序的输入。它们表明了需要应用程序做出响应的某种事件。消息由消息标识和消息参数构成,参数的内容随消息种类不同而不同。

1.1.1 消息的产生和处理

Windows 为每一个输入事件产生一个输入消息,如用户移动鼠标或按下一个按键都会产生输入消息。Windows 把输入消息收集在系统消息队列中,然后把这些消息,包括定时器和绘画消息,送到应用程序的消息队列中。应用程序消息队列遵循先进先出的原则,而定时器和绘画消息是例外的,这两类消息将保留在应用程序消息队列中直到应用程序处理完所有其它的消息。Windows 把属于特定应用程序的消息放置到相应的应用程序消息队列中。应用程序调用 **GetMessage** 函数读取消息,并用 **DispatchMessage** 函数把这些消息分发到相应的窗口过程。

Windows 可以把有些消息直接发送到相应应用程序的窗口过程,而无需先放置到应用程序消息队列中。这样的消息称之为非排队消息。一般情况下,非排队消息是只影响其窗口的消息。**SendMessage** 函数用于把消息直接发送至窗口过程。有关窗口过程的进一步情况,参见第 1.2.13 节“窗口过程”。

例如:**CreateWindow** 函数使 Windows 发送一个 **WM_CREATE** 消息到应用程序的窗口过程,并等待窗口过程处理完这条消息。也就是说,Windows 直接发送这条消息到窗口过程,并不把它放到应用程序消息队列中。

尽管消息大部分都由 Windows 产生,但应用程序本身也能产生自己的消息,并把它们放在自己或其他应用程序的消息队列中。

一般情况下,应用程序使用 **GetMessage** 函数从应用程序消息队列中移走消息,函数 **GetMessage** 在 **WinMain** 的一个循环语句中,这个循环叫做主消息环。**GetMessage** 函数在应用程序消息队列中搜索消息;如果有消息,就返回队列中最上面的消息;如果消息队列是空的,**GetMessage** 就处于等待状态。

处于等待状态的 **GetMessage** 把控制权交回 Windows,以便其它应用程序得到控制权来处理它们自己的消息。

一旦某个应用程序的 **WinMain** 函数从应用程序消息队列中获得了消息,就用函数

DispatchMessage 把消息发送到窗口过程。这个函数使 Windows 调用与消息有关窗口的窗口过程,并把消息的内容作为函数的参数;然后,窗口过程即可对消息进行处理,并对窗口实现所要求的改变。窗口过程返回后,Windows 即把控制权交给 **WinMain** 函数中的主消息环。这样主消息环即可获得队列中下一条消息。

注意: 除非特别说明,Windows 可以按任意次序发送消息;应用程序接收消息也无特定的次序。

用户每按一次键,Windows 即产生一个消息,这个消息含有一个虚键值,表明哪一个键被按过,但并不一定等于该键的字符值。而要想得到字符值,需要在 **WinMain** 函数的主消息环中调用 **TranslateMessage** 函数对虚键值消息进行翻译。此函数把另一条含有相应字符值的消息送到应用程序消息队列中,然后,这条消息即可被分发到窗口过程。

1.1.2 翻译消息

通常,WinMain 函数应使用 **TranslateMessage** 函数对每一条消息进行翻译,而不仅仅是虚键消息。尽管 **TranslateMessage** 函数对其他类型的消息没有作用,但这样做能保证准确地对键盘输入进行翻译。

下面的例子是一个典型的主消息环,**WinMain** 函数用它从应用程序消息队列中获得消息,并分发到应用程序的窗口过程。

```
int PASCAL WinMain(hInst, hPrevInst, lpCmdLine, ShowCmd)
HINSTANCE hInst;
HINSTANCE hPrevInst;
LPSTR lpCmdLine;
int ShowCmd;
{
    MSG msg;
    .
    .
    .
    while (GetMessage(&msg, NULL, 0, 0)) {
        TranslateMessage(&msg);
        DispatchMessage(&msg);
    }
    return msg.wParam;
}
```

使用加速键的应用程序必须首先用 **LoadAccelerators** 函数装入资源定义文件中的加速键表,然后用 **TranslateAccelerator** 函数把键盘消息翻译成加速键消息。有关加速键的内容,参见《Microsoft Windows 3.1 程序员参考大全(六)——编程指南》。

一个使用加速键的应用程序,其主消息环应具有下列格式:

```
while (GetMessage(&msg, NULL, 0, 0)) {
    if (TranslateAccelerator(hwnd, hAccel, &msg) == 0) {
        TranslateMessage(&msg);
        DispatchMessage(&msg);
    }
}
return msg.wParam;
```

TranslateAccelerator 函数应该放在标准的 **TranslateMessage** 和 **DispatchMessage** 函数的前面。而且,因为 **TranslateAccelerator** 自动地发送加速键消息到相应的窗口过程,所以,**TranslateAccelerator** 返回一个非零值时,就无需再调用 **TranslateMessage** 和 **DispatchMessage** 函数。

1.1.3 检查消息

应用程序可以通过 **PeekMessage** 函数检查其消息队列中的指定消息,而又不把它从队列中删除。如果队列中存在所要检索的消息,则返回一个非零值,这样就可以不必通过主消息环获取并处理这条消息。

一般情况下,当应用程序正实现一个长时操作时,如处理输入、输出时,应用程序用 **PeekMessage** 对消息进行定时检查。例如:这个函数可以用来检测操作的结束。如果消息队列中没有消息的话,**PeekMessage** 也给了应用程序一个放弃控制的机会,因为消息队列中没有消息时,函数 **PeekMessage** 可以放弃控制。

1.1.4 发送消息

应用程序用 **SendMessage** 和 **PostMessage** 函数发送消息至自己的窗口或其它应用程序的窗口。**PostAppMessage** 函数与 **PostMessage** 的不同之处是通过应用程序的模块句柄而非窗口句柄来发送消息。

PostMessage 函数使 Windows 投递消息,也就是把消息放到应用程序消息队列中。**PostMessage** 函数立即返回控制给调用应用程序;而在消息未被读出队列之前,由消息指定的任何操作都不会发生。

SendMessage 函数使 Windows 把消息直接发送到给定的窗口过程,而不通过应用程序的消息队列。窗口过程接收并处理完该消息或者调用 **ReplyMessage** 函数返回控制之前,Windows 不会将控制返回调用应用程序。

如果应用程序需要消息的返回值,那么发送该消息时,就必须用 **SendMessage** 函数;**SendMessage** 的返回值与处理消息的窗口过程的返回值相同。函数 **PostMessage** 发送消息之后就立即返回,它的返回值仅是一个布尔数,表明消息是否被成功地放进到队列中,并不表明消息被如何处理。

1.1.5 避免消息死锁

应用程序在处理由 **SendMessage** 函数从其他应用程序(或是通过 Windows)发来的消息时如放弃控制,那么就可能在 Windows 中产生一个死锁。

一般情况下,一个调用 **SendMessage** 发送消息给另一个任务的任务在接收消息的窗口过程返回之前是不能继续运行的,如果接收消息的任务放弃了控制,而发送消息的任务又不能继续运行和处理消息(因为它正在等待函数 **SendMessage** 返回),那么就出现了消息死锁。

处理消息的应用程序不能直接放弃控制,以免导致这类问题。调用下列函数之一会导致应用程序失去控制: