陆余良　陈先才
胡荣贵　许后华　编著

WINDOWS

编程短平快

南京大学出版社

WINDOWS 编程短平快

陆余良　陈先才
胡荣贵　许后华　编著

王世明　主审

南京大学出版社

1993·南京

(苏)新登字第 011 号

全新的设计风格
高级的编程技术
灵活的开发手段
完美的程序框架

WINDOWS 编程短平快

陆余良　陈先才
胡荣贵　许后华　编著
王世明　主审

*
南京大学出版社出版发行
（南京大学校内　邮编 210008）
科星电脑有限公司排版
83404 部队印刷厂印刷
*
开本 787×1092 1/16　印张 10　字数 256 千
1993 年 9 月第 1 版　1993 年 9 月第 1 次印刷
印数 1—5000
ISBN 7—305—02309—4/TP・76
定价11.00 元

快速进入 Windows 编程

Microsoft Windows 是一个以图形用户界面为基础的功能强大的软件环境。它以丰富多彩的图形取代了 MS—DOS 以字符为基础的单一形态，提供了漂亮、统一和友好的用户界面。特别是 90 年代初 Windows 3.X 的推出，在计算机领域产生了巨大而深远的影响。它以 Intel 80386 为硬件平台，充分发挥了 80386 处理器的能力，使其在图形功能、运行速度、内存管理、多任务环境、资源共享以及支持多媒体等方面都有了很大的发展。

从程序设计观点看，Windows 支持面向对象的程序设计方法，提供了多窗口处理、多任务环境以及消息驱动程序结构。Windows 强大的内存管理能力，使得在 Windows 平台上开发的软件可以不去考虑内存的管理就可以获得相当良好的运行品质。Windows 支持资源共享的能力以及它本身丰富的资源，形成了良好的软件开发环境，它的应用软件工具包 SDK 提供了各类工具、资源、函数库和数据结构。这一切，对于缩短软件开发周期，开发出高质量、用户界面良好的应用软件，都具有十分重要的意义。

Windows 3.X 推出的时间虽然不长，但已风靡全球，其销售量已达几千万套，其程序设计思想和方法正在被越来越多的软件开发人员所接受。几乎所有市场上畅销的 PC 软件都有 Windows 版本。据报导，Microsoft 已决定不再开发 DOS 版软件，而全面转向 Windows。预计 Windows 必将成为微机上压倒一切的操作环境。笔者 92 年在美国硅谷地区访问时曾听几位年青的朋友说："只要有在 Windows 环境下开发软件的经历，不愁找不到工作"。此话信然！

如何紧紧跟上这个迅猛发展的潮流，尽快地掌握在 Windows 环境下编程的技术，已经成为国内软件开发人员十分关切的问题。目前可以得到的参考资料和书籍，大多以详尽、深入地介绍 Windows 的方方面面功能的使用为目的，这就使得初学者很难在短期内把握 Windows 的程序设计要领，从而对开发 Windows 应用程序望而生畏。《Windows 编程短平快》一书正是在这种情况下应运而生的。

著名的计算机科学家，PASCAL 语言之父 N. Wirth 曾经说过："程序设计是一种构造性的技术，怎样开展这种构造性的又具有创造力的活动的教学呢？……我们所能采用的教学方法只能是仔细地选择和描述标准的例子"。该书正是以这种方式作为特点的，一开始就提出一个实例 THISBOOK，通过详细地剖析这个实例，

使读者很快掌握 Windows 应用程序的标准结构框架。然后每介绍一个新的内容，就通过修改这个框架中某些部分的方法，来形成一个新的程序实例，如此不断展开和深入。因此，读者在很短时间内就可以进入 Windows 编程。

该书的几位作者，均在计算机图形学、大型软件开发领域工作多年，特别是近几年来，专门在 Windows 环境下开发各种应用软件，积累了丰富的经验。书中的内容，都是作者们根据亲身经验和体会提炼而成，深入浅出，既全面地概括 Windows 的程序设计技术，又详细地介绍 Windows 应用程序的编程技巧。希望该书的出版，能为普及 Windows 应用软件的开发，使我国计算机产业跟上国际发展的步伐起一定的积极作用。

王世明

1993 年 9 月

目　录

第一章　全新的程序设计风格 …… 1
1.1　漂亮、统一的用户界面 …… 1
1.2　面向对象的程序设计 …… 2
1.3　消息驱动的程序结构 …… 3
1.4　多任务 …… 3
1.5　高效的内存管理 …… 4
1.6　数据交换与共享 …… 5
1.7　与设备无关的图形接口 …… 5
第二章　Windows 应用程序设计的基本方法 …… 7
2.1　Windows 应用程序开发环境与开发过程 …… 7
2.1.1　开发环境 …… 7
2.1.2　开发过程 …… 7
2.1.3　编程要点 …… 8
2.2　一个简单的 Windows 应用程序 …… 9
2.3　有关的基本概念 …… 15
2.3.1　Windows 函数调用 …… 15
2.3.2　Windows 的数据类型和数据结构 …… 16
2.3.3　句柄(HANDLE) …… 16
2.3.4　实例(INSTANCE) …… 17
2.3.5　程序入口点 …… 17
2.3.6　注册窗口类 …… 17
2.3.7　创建和显示窗口 …… 19
2.3.8　建立消息循环 …… 20
2.3.9　窗口函数 …… 21
2.3.10　对话框函数 …… 21
2.3.11　设备描述表(Device Context) …… 22
2.3.12　WM_PAINT 消息 …… 22
2.3.13　终止程序和 WM_DESTROY 消息 …… 23
2.3.14　包含文件(.H) …… 23
2.3.15　资源描述文件(.RC) …… 24
2.3.16　模块定义文件(.DEF) …… 25
2.4　如何建立自己的应用程序 …… 25
2.4.1　资源描述文件的改动 …… 26
2.4.2　包含文件的改动 …… 26
2.4.3　C 语言源文件的改动 …… 27

2.4.4 模块定义文件的改动…… 28
2.5 应用程序的编译、连接与调试 …… 29
第三章 灵活的输入手段 …… 30
3.1 键盘…… 30
3.1.1 键盘消息…… 31
3.1.2 字符消息…… 32
3.2 鼠标器…… 34
3.2.1 鼠标器消息…… 34
3.2.2 鼠标器消息的处理…… 36
3.2.3 如何捕捉鼠标器…… 37
3.2.4 用键盘仿真鼠标器…… 37
3.3 定时器…… 39
3.3.1 Windows 定时器的基本原理 …… 39
3.3.2 定时器的使用…… 40
3.4 控制窗口…… 42
3.4.1 建立控制窗口…… 43
3.4.2 使用控制窗口…… 45
3.4.3 常用控制窗口的建立和使用…… 48
3.5 文件…… 57
3.5.1 关于 OpenFile 函数…… 57
3.5.2 创建和打开文件…… 58
3.5.3 读写文件…… 58
3.5.4 重新打开文件…… 59
第四章 丰富的资源 …… 60
4.1 肖像、光标和点位图 …… 60
4.1.1 肖像(ICON) …… 60
4.1.2 光标(CURSOR) …… 62
4.1.3 点位图(BITMAP) …… 63
4.2 字符串…… 65
4.2.1 字符串资源的定义及使用…… 65
4.2.2 字符串资源与内存空间…… 66
4.3 菜单…… 66
4.3.1 菜单的基本知识…… 66
4.3.2 定义及使用菜单的一般方法…… 67
4.3.3 定义及使用菜单的另一种方法…… 68
4.3.4 菜单与消息…… 68
4.3.5 改变已有的菜单…… 70
4.3.6 使用点位图作菜单项…… 71
4.3.7 实现浮动的弹出式菜单…… 71
4.3.8 系统菜单的使用…… 72

4.3.9 与菜单有关的其它函数…… 73
4.4 键盘加速键…… 74
4.4.1 提供菜单加速键的方法…… 74
4.4.2 加速键的分配原则…… 74
4.4.3 加速键表的定义…… 75
4.4.4 装入加速键表…… 75
4.4.5 翻译加速键消息…… 76
4.4.6 关于加速键消息的几点说明…… 76
4.5 对话框…… 77
4.5.1 对话框模板…… 77
4.5.2 对话框函数…… 78
4.5.3 模式对话框与无模式对话框…… 78
4.5.4 对话框的显示与关闭…… 79
4.5.5 在对话框中使用控制窗口…… 81
4.5.6 Windows 3.1 通用对话框的使用 …… 87
4.6 内存管理与使用…… 89
4.6.1 使用内存…… 90
4.6.2 代码段和数据段…… 92
4.6.3 管理程序代码和数据应注意的问题…… 94
第五章 与设备无关的图形接口 …… 96
5.1 设备描述表句柄的获取与释放…… 96
5.2 映射方式…… 98
5.3 图形操作 …… 100
5.3.1 画笔 …… 100
5.3.2 刷子 …… 101
5.3.3 绘图模式的设定 …… 102
5.3.4 画点、画线和绘制区域图形…… 102
5.4 图元文件(Metafile) …… 104
5.5 调色板与图象显示 …… 106
5.5.1 调色板的概念 …… 106
5.5.2 与设备无关的点位图(DIB)的数据结构 …… 106
5.5.3 逻辑调色板的创建和使用 …… 108
5.5.4 图象显示 …… 112
5.6 文本与字库 …… 114
5.6.1 文本输出和颜色属性的设置 …… 114
5.6.2 系统字库的使用 …… 116
5.6.3 自定义逻辑字库的创建和使用 …… 116
5.6.4 枚举字库 …… 117
5.6.5 字库选择的通用对话框 …… 118
5.7 打印 …… 119

5.7.1 打印机设备描述表 …… 119
5.7.2 Escape 函数 …… 120
5.7.3 打印图形 …… 121
5.7.4 取消打印操作 …… 122
5.7.5 打印设置 …… 124
第六章 更高级的编程技术 …… 126
6.1 剪接板(Clipboard) …… 126
6.1.1 剪接板的数据格式 …… 126
6.1.2 如何使用剪接板 …… 127
6.1.3 使用剪接板应注意的问题 …… 128
6.2 动态数据交换(DDE) …… 129
6.2.1 DDE 的一些基本概念 …… 129
6.2.2 DDE 消息 …… 130
6.2.3 DDE 如何进行一次会话 …… 130
6.3 动态连接库(DLL) …… 132
6.3.1 如何建立一个 DLL …… 132
6.3.2 应用程序如何调用 DLL …… 137
6.3.3 编写 DLL 应注意的问题 …… 139
6.4 多文档界面(MDI) …… 140
6.4.1 MDI 窗口的组成 …… 141
6.4.2 编写 MDI 应用程序 …… 141
6.4.3 子窗口与文档数据 …… 143
6.4.4 一个 MDI 示例程序 …… 144

第一章　全新的程序设计风格

Windows 是 Microsoft 公司开发的一个图形窗口环境软件，最初是于 1983 年 11 月宣布的，两年后即 1985 年 11 月推出了第一个公开版本。此后，Windows 经历了不断的修改与更新，目前正在流行的是 Windows 3.1 版。Windows 的推出，特别是 1990 年 5 月推出 3.0 版以后，立即引起了用户的强烈反响，使得操作计算机的方法和软件开发过程发生了根本性的变化。近几年，Windows 软件每年都以数百万份的拷贝向外销售，并且已经开发出一大批 Winsows 环境下的应用软件。

Windows 运行于 MS—DOS 之上，是 MS—DOS 的扩展。它与 MS—DOS 共同管理计算机的硬件资源，MS—DOS 继续管理文件系统，而 Windows 管理其它一切，包括显示器、键盘、鼠标器、打印机和串行口，并负责内存管理和程序的执行、调度。

Windows 丰富多彩的界面改变了以往 DOS 系统那种呆板的面孔，给微机操作系统注入了新的活力。Windows 不具有强大的存储管理能力；能同时运行多个任务；可以通过多种方式进行应用程序间的数据交换，实现信息共享；能支持网络系统；在多媒体技术方面更将大显身手。1993 年 5 月 Microsoft 公司又正式推出了 Windows NT 操作系统，必将进一步推动 Windows 在全球范围内的普及。

与 DOS 环境相比，Windows 不管是对用户还是对应用程序开发人员都提供了更加强大的功能，特别是其先进的程序设计思想和方法，更是程序设计人员必须熟悉和掌握的。

1.1　漂亮、统一的用户界面

对于“用户友好接口”这一概念，大家并不陌生，这是衡量应用程序质量的一个重要方面。DOS 环境下用 C 语言或其它语言开发应用程序，程序员为了设计出“友好”的界面往往要花费很多精力和时间，而设计出的界面还是因人而异，没有统一的风格。这也给用户使用和掌握应用程序带来了困难。

Windows 为设计漂亮、统一和友好的用户界面提供了一种全新的方法。Windows 是一个图形用户界面(GUI)，各种操作对象都是以图形形式显示在屏幕上，通过键盘或鼠标器，用户可以在屏幕上直接操纵这些对象。这样，用户与程序之间的交互变得更加紧密了，不再是单一的用键盘输入信息到屏幕去显示，而是能直接与屏幕上的对象进行交互操作。

所有 Windows 应用程序的基本外观均相同，每个程序占据一个窗口。窗口是屏幕上的一个矩形区域，它是由标题条来标识的。程序中绝大多数功能均可通过菜单来执行。某些菜单会触发出对话框，通过对话框，用户可以输入较多的附加信息。

一个典型的 Windows 程序的界面如图 1－1 所示。

利用 Windows 提供的 Help 工具，可以为应用程序创建统一的联机求助系统，用户可以很方便地搜索、查询有关信息。

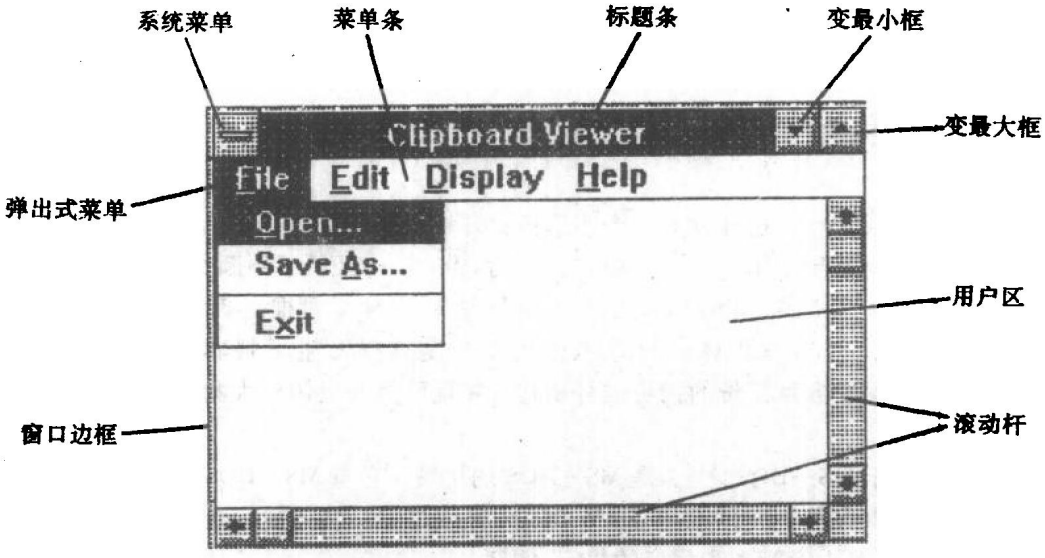

图 1—1　典型的 Windows 程序界面

此外，Windows 还提供了一种多文档界面技术，利用它在一个应用程序中可以创建多个窗口，分别用来显示和处理不同的文档。

由于用户界面的一致性，用户一旦学会了如何使用一个 Windows 应用程序，就会很快掌握其它 Windows 应用程序的使用方法。在 Windows 环境下设计统一的用户界面是较容易的事情。

1.2　面向对象的程序设计

Windows 程序设计实际上是面向对象的程序设计，这是目前最先进的程序设计方法之一。Windows 编程所涉及的对象很多，主要有窗口、菜单、对话框等。

如前所述，窗口是屏幕上的一个矩形区域，窗口通过键盘或鼠标器接收用户的输入，并将图形输出显示到屏幕上。

菜单是 Windows 应用程序接收用户输入的主要手段。一个菜单是一组可供用户查看和选择的命令列表，用户只要选择菜单项就能执行相应的功能。

对话框是一种特殊的窗口，它为用户显示更多的有关命令的信息，并能接收用户的输入信息。一个对话框可以包含一个或多个控制窗口，“控制窗口”也是一种特殊的窗口，又称为“子窗口控制”，简称“控制”。控制窗口的形式包括各种按钮、文本编辑框、列表框、组合框和滚动杆，可供用户作出选择和输入。

用户在屏幕上看到这些对象，可直接与它们进行交互。交互的方法是按一下按钮、滚动一个滚动杆或选中一个菜单项等。其实，用户所执行的这些操作是通过发送消息来告诉程序的，消息是应用程序执行命令的信号。

在面向对象的程序设计中，“对象”表示代码和数据的组合。以窗口为例，窗口是一个对象，其代码是窗口函数（该函数处理窗口的各种消息）；而数据就是那些与窗口相关的消息。

1.3 消息驱动的程序结构

Windows 消息提供了应用程序与应用程序之间和应用程序与 Windows 系统之间进行通讯的手段。

几乎每个 Windows 程序所做的事情都是为了响应发送到窗口函数的某条消息。在大多数应用程序中，很大一部分代码的内容都是用来处理这些消息的。

Windows 的消息由三部分组成：消息号、字参数和长参数。消息号由事先定义的消息名标识，字参数和长参数包含与消息号相关的值。例如，当光标在窗口用户区时，按下鼠标器的左按钮，Windows 就向该窗口的窗口函数发送一条 WM_LBUTTONDOWN 消息，其长参数含有光标的 x 和 y 坐标(分别在其低位字和高位字中)，而其字参数则包含指示各种虚拟键是否被按下的值(例如是否在按鼠标左按钮的同时，按下了键盘上的 Shift 键等等)。窗口函数接收到此消息后，就可以作相应的处理。

Windows 应用程序创建的每个窗口都要有一个窗口函数(有些特殊窗口的窗口函数由 Windows 内部提供)，用以处理发送到该窗口的消息。窗口函数由 Windows 采用消息驱动的形式直接调用，而不是由应用程序显式调用的。窗口函数处理完消息后又将控制返回给 Windows。

Windows 中有一个系统消息队列，简称系统队列。当开始执行一个 Windows 应用程序时，Windows 为该程序建立一个“消息队列”，称为应用程序队列。这个队列存放该程序可能创建的各种窗口的所有消息。程序中含有一段叫做“消息循环”的代码，用来从消息队列中检索这些消息并把它们分发到相应的窗口函数中。有些消息不必放到消息队列中而直接送给窗口函数。系统队列、应用程序队列、消息循环、窗口函数之间的关系见图 1－2。

Windows 应用程序接受用户输入的方式也不同于 DOS 环境下的应用程序。在 DOS 环境下，程序通过显式地调用一个函数(如 getchar())来读取键盘输入，该函数一般要等待用户按下一个键后，再把相应的字符返回给程序。在 Windows 环境下，Windows 接收所有来自键盘、鼠标器等设备的输入，并把它们放在相应的应用程序的消息队列中。在应用程序需要获取输入时，只不过是简单地从其消息队列中读取下一个输入消息。

从上面我们可以看到，Windows 是一种消息驱动式的系统。应用程序要实现的功能由消息来触发，并靠对消息的响应和处理来完成。

1.4 多任务

Windows 又是一种多任务系统，可以同时显示和运行多个应用程序，每个应用程序有各自的窗口。用户可以在屏幕上移动这些窗口、改变它们的尺寸、在不同的程序间进行切换，并可以在应用程序之间进行数据交换。

为标准的 DOS 环境编写的程序一般都假定它独占计算机的所有资源，这些资源包括输入输出设备、内存、系统显示器、CPU 等。而在 Windows 环境下，应用程序必须与所有当前正在运行的其它应用程序共享这些资源，为此，程序设计时要考虑多任务的特点，协调并共享资源，以避免耗尽资源。

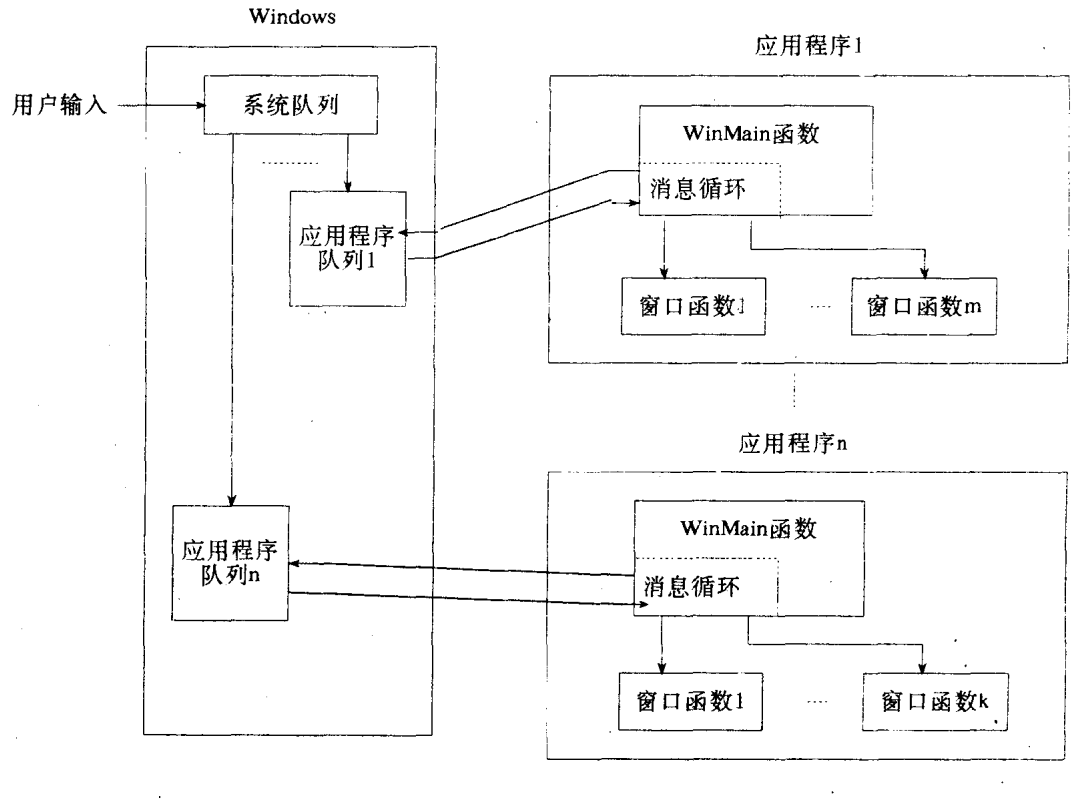

图 1－2　Windows 中对消息的处理

此外，Windows 实现多任务的方法与传统的多任务系统不同。Windows 通过在程序间进行切换来实现多任务，只有当正在运行的应用程序消息队列中没有消息（或只有 WM_PAINT 和 WM_TIMER 消息）时，才能转去执行其它有等待消息的程序。因此，它是一种非抢先的多任务，而传统的多任务系统是基于硬件时钟为每个程序分配一小块时间片来实现多任务的。

1.5　高效的内存管理

Windows 为保证多任务的运行，采取了一系列措施来提高内存的利用率，从而保证能以较少的内存资源运行较大的程序。Windows 内存管理的特点是：

1. 当 Windows 运行同一应用程序的多个拷贝（每个拷贝称为一个实例）时，每个实例使用相同的代码段和相同的资源；

2. 在 Windows 环境中分配的内存块多数是可移动的（如：可移动的代码段、可移动的数据段、可移动的资源等），从而便于 Windows 对内存块的管理和提高内存资源的利用率；

3. 代码段和程序资源通常都是视需要而被装入内存的，而且在多数情况下分配为可丢弃的内存块。在内存资源紧张时，这种内存块可被丢弃以释放所占空间。当再次需要时，Windows 能自动地从磁盘上复制相应的内容到内存中。

Windows 突破了 DOS 对内存 640KB 的限制，提供了实模式、标准模式和 386 增强模式等运行模式。标准模式和 386 增强模式统称为保护模式。

在基于 Intel 8086 CPU（或 80286、80386 及以上的 CPU，内存少于 1MB）的机器上，Windows

3.0 将以“实模式”运行。在这种模式下，Windows 系统以及 Windows 应用程序占用 640KB 常规内存的上部，而 MS—DOS、设备驱动程序以及内存驻留程序占用 640KB 常规内存的下部。Windows 3.1 不支持实模式。

在基于 80286 CPU，内存至少 1MB（或 80386 及以上 CPU，内存至少 1MB 但不大于 2MB）的机器上，Windows 3.X 将以标准模式运行。这种模式又称为 286 兼容的保护模式。在这种模式下 Windows 可使用多达 16MB 的常规内存和扩展内存。

在基于 80386 及以上 CPU，内存至少 2MB 的机器上，Windows 3.X 将以 386 增强模式运行。这种模式的主要特点是使用 CPU 的页寄存器实现虚拟内存管理。CPU 的内存页面长度规定为 4KB，Windows 可将内存页面交换到磁盘上，需要时再从磁盘上重新装入内存。页交换和重装入的过程均由 Windows 系统自动进行，在编写应用程序时不必考虑页交换的过程。

当 Windows 系统启动时，它根据机器的硬件配置自动选择适当的模式运行。用户也可以从 DOS 命令行中使用/R（实模式）、/2（标准模式）或/3（386 增强模式）参数来启动 Windows 系统，以改变 Windows 的缺省模式。如，WIN/3 命令则表示 Windows 系统在 386 增强模式下运行。

1.6 数据交换与共享

Windows 提供了多种手段来实现应用程序之间的数据交换与共享，包括：剪接板（Clipboard）、动态数据交换（DDE）和动态连接库（DLL）。

剪接板是共享内存块的管理器，应用程序既可以向其中写入数据也可以从中读出数据，利用它，能实现应用程序内部或应用程序之间的数据交换。

动态数据交换是一种更高级的数据交换手段，它是一种数据交换的消息协议，通过建立应用程序之间的数据链路，使得应用程序之间可以进行自动的数据传送，而不需要用户干预。这种功能对于开发需要连接实时数据的应用程序（如过程控制和监测等）具有十分重要的意义。

动态连接库是应用程序之间实现代码和资源共享的一种手段。在 Windows 环境下，由于可以同时执行多个任务，使用 DLL 比使用静态连接库有更多的优点。如果两个应用程序同时运行，且它们使用了某一静态库中的同一函数，那么系统中就要出现该函数的两个副本，而 DLL 却能允许若干应用程序共享某个函数的单个副本，从而节省了内存空间。此外，DLL 还可以用来实现其它资源的共享，如数据和硬件资源的共享。

Windows 所有的运行库均为 DLL，如 GDI.EXE、USER.EXE、KERNEL.EXE 等，它们是 Windows 的主要构成部分之一。标准的 Windows 设备驱动程序也是用 DLL 实现的。程序员还可以根据需要，开发出自己的 DLL。

1.7 与设备无关的图形接口

Windows 提供了丰富的、与设备无关的图形处理功能。应用程序能很方便地画出直线、矩形、圆和其它复杂图形，而不需直接与具体的输出设备打交道。由于 Windows 提供了设备无关性，因而在应用程序里可使用同一函数在打印机或显示器上输出同一种图形。

Windows 应用程序并不直接访问图形显示设备，而是通过其图形程序设计语言（又叫图形设备接口，即 GDI）来实现图形输出的。Windows 应用程序的输出可以适用于任何显示或打印设备，

只要在 Windows 环境下安装相应的设备驱动程序即可，由设备驱动程序将 GDI 图形输出请求转换为显示器、打印机等输出设备上的图形输出。图 1—3 说明了 Windows 应用程序的图形输出过程。

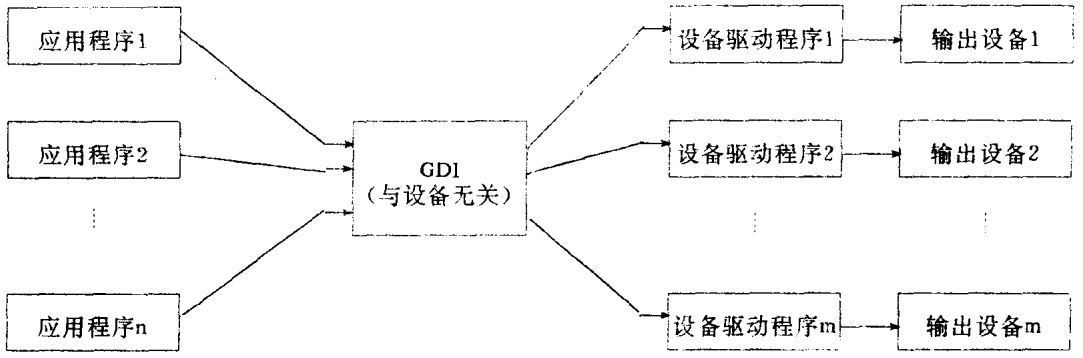

图 1—3　应用程序的图形输出过程

从前面几节的讨论我们可以看到，在 Windows 环境下开发应用程序与在一般环境（如 MS—DOS）下开发应用程序有很大区别，Windows 提供了更为强大的功能和更加先进的程序设计方法。Windows 3.X 应用软件开发工具包 SDK（Software Development Kit），为在 Windows 环境下开发出具有窗口特点和功能的应用软件提供了各类工具、资源、函数库和数据结构。使用好 Windows 及其开发工具包 SDK，对提高机器的使用率、缩短软件开发周期、开发出高质量的应用程序，都具有十分重要的意义。

当然，设计 Windows 程序也并不是非常简单的事情。对于初学者来说有许多概念需要掌握和理解，必须将程序设计思想调整到 Windows 的面向对象、消息驱动式结构上来。

第二章　Windows 应用程序设计的基本方法

本章主要介绍有关 Windows 程序设计的一些基本概念，以一个简单的应用程序为例，说明 Windows 应用程序的主要组成部分，让读者对 Windows 程序有一个初步的了解。并从初学者的角度出发，逐步引导读者熟悉 Windows 编程。

2.1　Windows 应用程序开发环境与开发过程

Windows 应用程序除了包含常规意义下的源程序外，一般还包含资源描述文件(.RC 文件)和模块定义文件(.DEF 文件)。为建立一个 Windows 应用程序，需要有一定的硬件和软件开发环境，本书中，我们主要以 Windows 3.1 SDK 以及 Microsoft C/C++ 7.0 作为开发平台来进行介绍，但书中的大多数内容同样适合于 Windows 3.0 环境。

2.1.1　开发环境

从开发应用程序的角度来看，我们建议最好使用如下硬件环境：

1. Intel 80386 或以上的处理器；
2. 4MB 以上的内存空间，120MB 以上的硬盘；
3. VGA 以上的显示卡和相适应的彩色监视器；
4. 一台辅助单色监视器和适配卡(Windows 调试器 CVW 使用)；
5. 一个 Microsoft 鼠标或兼容的鼠标。

软件工具建议使用

1. MS—DOS 5.0 版本以上；
2. Microsoft C/C++ 7.0。其中包括编译器(CL)、连接器(LINK)、程序维护工具(NMAKE)、程序员工作平台(PWB)等；
3. Windows 3.1 SDK。其中包括资源编译器(RC)、图象编辑器(IMAGEDIT)、对话框编辑器(DLGEDIT)、字体编辑器(FONTEDIT)、Windows 调试器(CVW)等工具。

编写 Windows 应用程序，除了要使用 C 语言的库函数及数据结构外，还需要使用 Windows SDK 提供的库函数及各种数据结构、语句、文件结构，所有这些内容组成了 Windows 应用程序设计接口(API)。

2.1.2　开发过程

Windows 应用程序一般由源程序文件(.C)、包含文件(.H)、资源描述文件(.RC)、模块定义文件(.DEF)组成。源程序文件中含有 WinMain 函数、窗口函数及其它源代码；包含文件中包括常量定义、公用变量说明、类型定义及函数原形说明；资源描述文件用来描述应用程序的所有资源，包括菜单、对话框等等；模块定义文件用来定义应用程序模块的属性，如段属性、堆栈大小等，另

外还要定义被应用程序引出的函数名及其序数值，这些函数将被 Windows 直接调用，称为回调函数(Callback Function)，其中包括窗口函数和对话框函数。

为建立 Windows 应用程序，必须创建上述各种文件，当然，.C 和 .H 文件可以不止一个。在创建 .RC 文件之前，还可能要使用 Windows SDK 提供的资源编辑器(如图象编辑器、对话框编辑器、字体编辑器)创建应用程序所需要的光标、肖像、点位图、对话框和字库等资源，以便在 .RC 文件中对它们进行命名、描述。

这些文件建立后，首先利用 CL 工具编译所有的 .C 文件，生成 .OBJ 文件；然后利用 LINK 工具将 .OBJ 文件与 .DEF 文件连接成 .EXE 文件；最后利用 RC 工具将 .RC 文件编译成 .RES 文件，并将它加入到可执行文件中。

为了对程序很好地进行维护，一般要建立一个 .MAK 文件，该文件列出建立一个应用程序所需的命令和文件，其中命令可编译和连接应用程序所需的各种文件。利用 Microsoft C 提供的程序维护工具 NMAKE 可以执行 .MAK 文件。

实际上，除了用资源编辑器创建资源外，其余工作均可以在 Microsoft C 提供的集成化开发工具 PWB 环境下完成。

2.1.3 编程要点

Windows 应用程序设计不同于一般 C 程序设计，读者除了要将设计思想转到面向对象、消息驱动上来以外，还必须领会和掌握以下要点：

1. 不要对 CPU 进行独占控制——它是一种共享资源。由于 Windows 是一非抢先的多任务系统，也就是说在应用程序未释放所占用的 CPU 之前，它不能从应用程序那里夺回控制，所以应用程序必须谨慎地使用好 CPU，以给其它应用程序有充分的执行机会。

2. 不要直接访问内存或其它硬件设备，如：键盘、鼠标、定时器、显示器以及串并口等。Windows 必须对这些硬件资源进行绝对的控制，以保证正在运行的应用程序都能均等地、无中断地访问它们。

3. 在 Windows 应用程序里，所有 Windows 能调用的函数必须使用 PASCAL 关键字定义，这将保证函数能正确地访问参数；另外除了 WinMain 函数以外，都必须使用 FAR 关键字定义。Windows 可调用的函数有 WinMain 函数和回调函数。

4. 每个应用程序必须具有一个 WinMain 函数，它是应用程序的入口点(类似于一般 C 程序的 main 函数)，没有它 Windows 应用程序就不能运行。该函数含有用来创建窗口的代码以及消息循环等代码。其形式如下：

```
int  PASCAL  WinMain(hInstance,hPrevInstance,lpCmdLine,nCmdShow)
HINSTANCE    hInstance;     /* 当前实例的句柄 */
HINSTANCE    hPrevInstance; /* 前驱实例的句柄 */
LPSTR        lpCmdLine;     /* 命令行的地址 */
int          nCmdShow;      /* 显示窗口的类型(打开/肖像) */
{
   ⋮
}
```

WinMain 函数声明必须带有 PASCAL 关键字，但不能使用 FAR 关键字。

5. 使用 Windows 函数时，一定要检查其返回值，否则可能会引起程序工作异常。例如，用 GlobalAlloc 函数从全局堆中分配内存块时，如果不成功则返回 NULL，此时若再按正常情况进行处理就会引发错误。