WILEY

WIN32 高级程序设计

[美] Martin Heller 著
祝远玲 冯 玉 译
王 勇 校

WITH DISK

電子工業出版社
Publishing House of Electronic Industry

Win32 高级程序设计

[美] Martin Heller 著

祝远玲 冯 玉 译

王 勇 校

電子工業出版社

(京)新登字 055 号

内 容 提 要

本书系统地介绍了 Win32 高级编程中的有关移植和程序设计等问题。第一章、第二章介绍了 16 位编程到 32 位 Windows 编程的转换；第三章是关于如何实现 C 语言到 C++的转换；第四章具体讲述 C++编程；第五章介绍了如何使用 Win32 的一些高级特征来优化 Windows NT 下的 Win32 应用程序；第六章讲述了如何通过 Win32s 在 Winodws 3.1 下运行 Win32 应用程序；第七章及第八章介绍了多媒体编程和对画笔系统的支持；第九章学习使用 Unicode；第十章概述了 OLE 2；第十一章重点讲述了进程通信和分布计算；最后在第十二章对 Windows 的新版本进行了介绍。

本书适用于软件开发人员及计算机专业有关人员。

本书的中文翻译与文字处理工作，由美国 Far East Books 公司完成。

Win32 高级程序设计

[美] Martin Heller 著

祝远玲 冯 玉 译

王 勇 校

责任编辑 胡毓坚

*

电子工业出版社出版(北京市万寿路)

电子工业出版社发行 各地新华书店经销

北京市燕山联营印刷厂印刷

*

开本：787×1092 毫米 1/16 印张：23.5 字数：572 千字

1994 年 11 月 第 1 版 1995 年 5 月 第二次印刷

印数：6000—11000 册 定价：60 元(含磁盘)

ISBN 7-5053-2681-3/TP・835

译　者　序

Microsoft 于 1993 年正式推出 Windows NT，作为一个 32 位系统，Windows NT 的许多新特性对软件开发人员来说是极具吸引力的，因而掌握 Win32 编程以更好地利用 Windows NT 的性能已成为 Windows 软件开发人员的迫切要求。

Martin Heller 的这本《Win32 高级程序设计》对 Win32 编程做了十分全面的介绍，该书的最大特点就是使用了大量的实例，使得书中的内容易于理解，不失为一本很好的 Win32 指南。本书的译出希望能够对国内的读者有所帮助，由于时间仓促和水平有限，书中难免会有不足和错误之处。敬请广大读者多提宝贵意见。

全书由王勇审校。其中，第一章至第六章由祝远玲翻译，第七章至第十二章由冯玉翻译。参加翻译和整理工作的还有张兰、王旭和马广文。曹向东、任武兰等为本书的出版作了大量录排工作，特此感谢。

译　者

一九九四年八月于北京

前　言

软件开发人员的工作似乎是永无止境的，对于在某一桌面操作系统下开发的应用软件，用户会希望在其它系统下也能够运行，但既便能够实现这种可移植性，用户又会希望它能够适用于现在迅速发展起来的各种基于微处理器的电子产品，例如：办公设备，手持设备，便携机以及功能强大的服务器等等。由于这些新型设备能够满足各种应用程序的计算需求，用户期待着开发人员不仅能够提供高效的个人产品，而且还能提供适应商业或关键任务应用的软件。

那么开发人员为适应这种需求究竟应该支持多少种不同的程序界面、操作系统和格式呢？而这样做的相应代价又如何呢？实际上最有效的方法就是尽可能地保留和利用在时间、金钱和编码上的已有投资。基于 Windows 的产品已得到相当广泛的应用，因而 Windows 应该能够提供一种有效的开发方法，使得开发人员和用户能够轻松地仅依赖于他们原有的投入和系统结构，而获得对新型设备的存取能力并开发新的应用。

但是言之易行之难，仅基于单一的操作系统来适应如今开发中的多向性是十分困难的，内存小的便携机往往注重能量损耗，而高端工作站或服务器则注重安全性、能力和可靠性，因而它们各自所使用的技术和方法之间存在着很大的差异，克服这种差异以保持统一的用户界面和程序界面将使用户受益非浅。

Microsoft 为迎接这一挑战，将它的 Windows 产品从单一产品扩充到一个产品家族，每个家族成员都具有特殊的内部实现，而它们之间又是互补的。

Windows 所采取的策略是：开发一个连续的操作系统家族，它横跨笔式电脑、笔记本、桌面系统、高端工作站以及服务器和微处理器等各种机型。统一的操作系统家族系列具有相同的用户界面，相同的应用和程序模型，从而软件开发人员和他们的用户将真正受益：

- 软件开发人员将获得更大的市场，他们能够因此节省许多时间和精力，而无需再为一种新平台而重写应用程序。他们将以更低的开销而更快地进入市场，同时不必担忧由于仅基于某种平台而限制了产品的市场，各厂商在选择他们的合作伙伴时将获得更大的自由，因为他们使用相同的操作系统语言，而且通过他们共同的操作系统知识可以减少开销；
- MIS 管理员也将受益，大多数计算机系统安装中的巨大开销是由于缺乏经验而并非硬件的原因。一个单一的操作系统家族能够减少培训的时间和管理开销；它简化并增强了管理工作，支持旧式的小主机系统，减少开销并解决了向新系统和新平台转换等令人头痛的问题；
- 终端用户也是受益者。他们能够在某种平台（例如桌面系统）上运行一个应用程序，并能以同样方式在另一平台（例如笔式或笔记本式）上运行。用户一旦掌握了某个应用软件的使用，便能够使用另一完全不同的应用，因为用户的不同应用是基于同一个操作系统工作的 。对终端用户来说，这将非常节省时间与财力，并带来高效率。这就是扩展 Windows 操作系统产品家族背后的策略。这个策略的实现是逐级处理的；
- 为 Microsoft At Work 手持式小型可移动设备而设计的 Windows 兼容操作系统；
- 为中档和未联网的 PC 机所设计的 Winodws 3.1；

- 为网络上的分组计算而设计的 Windows TM for Workgroups；
- 为功能较强的 PC 机，工作站以及客户/服务器结构的网络系统而提供的 32 位操作系统 Windows NT。

对开发人员而言，获得所有这些 Windows 产品支持的关键是使用 Win32 应用程序界面，这是一种基于 32 位的应用开发程序界面。无论是开发一个全新的应用软件，还是从其他系统，例如 UNIX 或 OS/2，向 Windows 下移植软件，如果是 32 位的应用程序，Win32 都将保证它与各 Windows 家族成员是兼容的。Win32 允许开发人员利用 Windows NT 的所有优势，Windows NT 被认为是客户/服务器计算的最为有力、最为可靠和开放的平台，大批用户开始转向客户/服务器结构来处理商务应用和关键任务应用，因而开发人员能否获得成功，将取决于他们通过 Win32 应用开发来利用 Windows NT 的能力。

如何利用 Win32 来为 Windows NT 及 Windows 家族系列开发强大的 32 位的应用软件呢？使用高级的 Win32 位编程，则已迈出了关键的一步，作者在后面的章节中力图提供一个综合的 Win32 开发方法。本书中包含许多清晰、丰富和非常有用的入门指导和实例，是一本 Win32 指南书籍。无论对从 16 位开发转向 32 位，还是从另一 32 位系统移植到 Windows NT，或者生成一个全新的应用软件，作者都进行了分别的详述。

书中内容包括：如何实现 C 语言到 C++的转换；如何优化 Windows NT 下的 Win32 应用程序；如何通过 Win32s 在 Winodws3.1 下运行 Win32 应用程序；以及如何支持多媒体和画笔系统，读者借助于本书，能够掌握网络、进程间通讯以及相关的分布式客户/服务器处理机制。读者还将学习到如何使用 Unicode 来为其应用程序生成一个通用版本。通过 Win32 应用程序，使读者得以充分地利用 Windows NT 的强大功能、可靠性和开放性。其优势表现在：

- Windows NT 的界面与 Windows 界面相同，但提供了更为有力的支持，这已为 Windows 用户所熟悉，Win32 主要通过提供先进的操作系统支持（例如：多线索的进程、同步机制、安全机制、I/O 和对象管理）而获得这样的优势，用户能够同时运行多个应用程序，由于 Windows NT 是独立于平台且是可扩缩的，用户希望能够将 Win32 应用程序运行在各种处理器上：从 Intel 886 芯片到 RISC 芯片，从 MIPS 和 Digital 到一些 30 个对称多处理器系统以至于多于 2600 种计算机和外围设备上；
- 用户需要 32 位的应用程序运行在最可靠的系统上。可靠性和安全性不再是后来的一个附加层，而在设计时已嵌入 Windows NT，其中包括对不中断电力供应的支持等性质以及新的 Windows NT 文件系统（NTFS），减少了硬件失败的可能性并保证了从任何例外失败迅速地恢复的能力。Windows NT 提供了广泛的安全性，其官方证明的安全性是 C2 级别，能够预防无意或恶意的篡改，这将促进 Windows NT 以及 Win32 应用程序打入新的市场；
- 开放性是评判 32 位操作系统的另一个重要标准，Windows NT 操作系统提供对网络多种协议的内部支持，包括 TCP/IP、NetBEUI、IPX/SPX 和 DLC。它支持大部分网络，包括 SAN、LAN Manager、NetWare、NFS、Banyan VINES 和 AppleTalk，Windows NT 支持分布式计算标准，包括：Windows Sockets、Named Pipes 和 OSF DCE——可兼容的远程过程调用（RPC），这种综合支持使得 Win32 成为分布式客户/服务器应用软件的最佳选择——它们能够无缝地存取网络上的不同主机和数据库中的信息。

实业家们也同样对 Windows NT 热心倍至，Windows NT 自从 1993 年中被引入以来，他

们有 73,000 多家购买了 Windows NT 软件开发集，使之成为最畅销的操作系统软件开发工具集之一。独立的软件厂商中正在进行开发的 Windows NT 的 32 位应用软件有 2000 多个。公司用户还开发另外 3800 种应用软件供内部使用，其中有 25%是从 UNIX 、VMS、OS/400 和其它高端系统移植而来。

欢迎进入 Win32 和 Windows NT 的世界——在您发现了所有 32 位 Windows 应用软件开发的优势后，祝您成功！

Paul Maritz
Senior Vice President
Systems Division
Microsoft Corporation

原　序

《Win32 高级程序设计》在某种意义上说是一旅行指南或路线图。它标出了软件开发人员从熟悉的 16 位 Windows 世界到 32 位的新领域所需要经历的区域。

我们为什么要放弃自己熟悉的东西呢？从 16 位编程转移到 32 位编程的优点是使人非常信服的：许多计算的速度自动加快，狭窄的 64 KB 的段大小变成了宽敞的 4 GB 线性内存空间，并且最让人烦恼的与整数范围有关的溢出问题也不存在了。在特殊情况下，速度将大幅度提高。例如，当把使用 "huge"16 位段偏移指针（如我以前的书《Windows 高级程序设计》中所用的例子 IMAGE2 中实现的 24 位彩色调整）的计算转换为使用"near"32 位指针的计算，速度可提高五倍。

除了与 32 位系统有关的优点外，Windows NT 还具有许多引人入胜的新特点。Windows NT 支持高端硬件如 MIPS R400 boxes、DEC Alpha AXP PCs 和对称多处理器，它还有一增强的图形设备界面，包括 Bézier 曲线（一种有用的类似于样条的光滑曲线），Path（一种通用的用作创建和填充复杂图形的机制），World 变换（允许显示的图形页从绘图协作系统中被旋转、绘制、反射或切割）和屏蔽（它允许一些区域排除在位图的显示之外）。它还支持执行的多线索（一种多任务的轻量方式）、多种进程通信机制和 C2 级的安全性，并且 Windows NT 还支持一些重要的网络。

Win32 是 Windows NT 的 API，很大程度上与 16 位 Windows 3.1 的 API 一致。基本的差别是：现存的 API 已扩展到 32 位上；新的 APIs 支持线索、多任务、安全性、Bézier 曲线和其它的 Windows NT 的高级特征；而与 32 位不相关的 16 位 APIs 如段操作等，则被删除。

Win32 包含的一套 DLLs 和 VxDs（ Win32s）能让 Win32 的程序在 Windows 3.1 上运行，这样就能提高 Win32 程序的销售量，它们非常有用和方便，但它们也不是治百病的灵丹妙药。Win32 集合中支持 Windows NT 高级特征的新 APIs 出现在 Win32s("s"表示"subset")中的只有返回 FALSE 的短线函数。例如，能够用 Win32s 程序来创建一个线索，但是编程时必须保证此程序既使在线索创建失败时，也能象在 Windows 3.1 下一样运行。

Win32s 也不是生成 32 位 Windows 应用程序完全通用的方法。一个主要的限制是它没有直接调用在 16 位 DLL 中函数的措施，而只能调用系统支持函数。然而 Win32 还是支持几种间接调用 16 位 DLL 函数的方法。

Windows 的未来版本，从"Chicago"开始将包括许多在 Windows NT 的新的功能。Win32s 程序在编写时充分利用当前存在的高级函数的优点，它在 Windows 的新版本上将会自动地比在 Windows 3.1 上工作得更好、更快。若想为"Chicago"作准备，现在就应开始学习 Win32。

我是因为自己的需要才决定写《Win32 高级程序设计》的：我想强迫自己使用 Windows NT 的新功能，并从 C 编程转换到 C++编程。我将尽力保存一本很好的关于从 Windows 向 Windows NT 世界迁移的旅行日志，它可以作为这种旅程的一个路线标志图。

我希望《Win32 高级程序设计》是一本有用的指南，无论是把残存的 16 位代码移植到 32 位 Windows 上或编写与两者都兼容的新代码，还是直接跳到 32 位 Windows 编程，无论你是新手还是老手，我都希望这里的观点和程序的摘录能在您的工作中发挥作用。我建议读者阅读一下我以前的一本书《Windows 高级程序设计》，它将会帮助您从熟悉 Windows SDK 入

手，直到能学会编写有价值的多模块 Windows 程序。

为了有效地使用《Win32 高级程序设计》中的资料，应该具备一台运行 Windows NT 的计算机、Win32 的有关文献、Windows NT 的 32 位 C 和 C++编译器、运行 Windows 3.1 的计算机和 Win32s 库的文献。

在第一章，学习了进行从 16 位编程到 32 位 Windows 编程的转换；第二章，我们把在《Windows 高级程序设计》中开发的例子 Image2 迅速地移植到 Win32 中；在第三章，我们学习从 C 编程语言到 C++的转换；在第四章，我们不再使用移植的代码，然后在第五章学习使用 Win32 的一些高级特征。

在第六章，我们回过头来使用 Win32s，在 Windows NT 的高级特征以及与 Windows 3.1 的兼容性之间做一权衡。在第七章，我们开始运用一些多媒体编程，在第八章，我们将学习如何支持 Pens、Ink 和 Tablets。

在第九章中，学习使用 Unicode 并做了一些国际化工作，第十章研究了 OLE 2，甚至于 OLE 3。第十一章，学习了进程通信和分布计算，最后在第十二章，展望未来。另外，为了查找方便，我们还在附录中描述了一些 Win32 开发的资料和工具。

本书中有许多资料，不必担心很难学，我们会慢慢开始，并且还有大量例子来辅助学习。

没有许多人的帮助，我是不可能写出这本书的。首先，我要感谢我的编辑，Diane Cerra，是他督促我真正写下这本书。

我还要感谢我的书评作者：Ed Adams，Roger Grossman，Timothy Larson，Chris Marriott，John Ruley，William VanRyper 和 Bjarne Stroustrup。他们花了许多时间来阅读，替我发现错误，使我能够做得更好。我万分感谢他们的努力。

本书的漂亮设计和布局是 Desktop 工作室的 Clair Stone Spellman，在 Frank Grazioli 和他的工作小组帮助下的作品。我是用 Microsoft Word for Windows 写的这本书，Claire 为这本书付出了辛苦的劳动。

最后，我还要感谢我的妻子 Claudia 和女儿 Tirzah。和作家一起生活非常不易，我的家庭更使我确信了这一点。我将永远不会忘记 Clandia 特别忍受了更多的痛苦，但她最终赢得了胜利；当我写完这本书之后但还没有出版时，我们的第三个孩子出生了，我就将这本书献给我的孩子。

目　录

第一章　从 Win16 到 Win32 的移植

从美国的东部去往西部，今天的人们可以乘坐飞机或者驱车前往，而在一百五十年前，那些西部拓荒者们则是驾着马车上路的。他们心中祈祷着能够尽快地穿过沙漠，越过高山，并能躲过土著的袭击。

许多人丧生于漫长的旅途上，没有好的方向指引，很容易迷失在沙漠和高山中。所以艰难的旅程中，方向是最为珍贵的。

第一章可以看作是一张粗略的地图：它并不能指出最佳的路线，但它却标明了高山和沙漠的位置，我们通过它可以找到如何避免大部分的危险。

我们首先考察的是关于字长(word size)的变化，这是一个很不起眼的问题，但也确实有不少人在上面犯错误。

1.1　字长的烦恼

关于字长对编程的影响问题众说纷纭。乐观人士认为大字长将自动使程序运行得更快并更为精确，忽视能力和应用范围的保守人士则认为现在的字长对于例外处理问题实在太小了。

一个 16 位有符号整数的表示范围是-32,768 到 32,767(32K-1)；一个 16 位无符号整数的表示范围是 0 到 65535(64K-1)。32 位有符号整数的表示范围是-2,147,483,648 到 2,147,483,647(2G-1)，32 位无符号整数的表示范围是 0 到 4,294,967,295(4G-1)。

16 位 Windows 编程中 32K 和 64K 的局限性日益严重，有时是直接地由于整数的大小引起的，有时则是间接地由于内存片段的最大尺寸的问题。有时这种局限性可以通过小心地编码来避免，而有些时候这种局限性将导致程序运行极为缓慢，下面将逐一举例说明：

```
#define n 10
int x[n],y[n],xmid=0,ymid=0;

for(i=0;i<n;i++) {
      xmid += x[i];
      ymid += y[i];
      }
xmid /= n;
ymid /= n;
```

此代码在 1024×768 的 Super-VGA 显示屏上仍是可运行的，但如果要在一台每英寸 300 个点的激光打印机上打印时会出现什么情况呢？8.5 英寸×11 英寸的页具有 2550×3300 的点坐标，很有可能 10 个坐标相加起来会超过 32,767，其总和就会溢出，结果是一个很大的负数，这时就会将中心点错打在页尾上，有时甚至错打在另一页上。显然，当 n 大于 10 时，错误的发生就会更频繁。

这种情况下，可以不借助于 long 型变量就可重写一更可靠一些的代码，尽管相比之下

具有较低的效率和精度。

```
#define n 10
int x[n],y[n],xmid=0,ymid=0;

for(i=0;i<n;i++) {
      xmid += (x[i] / n);
      ymid += (y[i] / n);
      }
```

另一种选择是使 xmid 和 ymid 成为 long 型变量——即使用 32 位的工作变量。在一个 16 位体系结构中，这意味着编译器必须生成代码将 X[i]和 Y[i]的值从 16 位扩展到 32 位，并能够生成或调用一种代码，该代码能够使用 16 位寄存器做 32 位数字的加法。这种方法代替了那种通过在循环中附加代码来生成加法指令的方法①。

下面来考虑一种较严重的情况。这是 IMAGE2 中 PROCESS 模型的代码，它是《Windows 高级程序设计》一书中的例子程序：

```
int dr,dg,db;
double r,g,b,dh,ds,dv,dl,h,s,v,l;
LPBITMAPINFOHEADER lpbi;
RGBQUAD FAR *pRgb;
unsigned char huge *pixels;
unsigned char huge *pb;
unsigned char huge *pb1;
unsigned char huge *pb2;
int colors,i;
DWORD il;
//NOTE: BOUND is a macro that limits the first variable
//  to the range of the second and third variables

lpbi=(VOID far *)GlobalLock(hdibCurrent);
pixels = (unsigned char huge *)(lpbi + lpbi->biSize);
switch(ColorModel) {
      case IDD_RGB:
        for(il=0;il<lpbi->biSizeImage;il+=3) {
             pb=pixels+il;
             pb1=pixels+il+1;
             pb2=pixels+il+2;
             *pb  = (BYTE)BOUND(*pb +db,0,255);
             *pb1 = (BYTE)BOUND(*pb1+dg,0,255);
             *pb2 = (BYTE)BOUND(*pb2+dr,0,255);
             }
        break;
...
```

此例用来改变 24 位图象的颜色，其中值得注意的问题很多。首先，由于逻辑上有错误，对某些图象的处理可能会失败，这主要表现在：应该按扫描线循环处理像点，而不是在对整个图象线性地处理每个像点。因为扫描线的末尾能被自动填满以致于下一条线落在双字的长度范围内，则无法以 3 来计数。更糟的是，Windows 甚至于没有为扫描线末尾的空字节分

① 考虑 c=a*b，其中所有的变量都声明为长整型。在生成代码时，16 位的 Microsoft 编译器将产生一个调用，它要包括 5 次乘法和 2 次加法，但任何的 32 位编译器都只生成包含一次乘法的代码。（感谢 Chris Marriott 指出了这一点。）

配内存，一旦碰上这样的故障，往往很难弄清其原因。我也曾在很长一段时期内将这种故障错误地归结为是 24 位设备驱动器的问题。

第二个需要注意的问题比较重要。在这个颜色校正代码中，指针变量 pixels、pb、pb1 和 pb2 都是 huge 型，而索引变量 i1 是一个 DWORD 变量，这意味着位于循环体内的二级指针 pb、pb1 和 pb2 的计算是十分低效率的：因为每次都必须重新计算选择器(selector)和偏移量，以确保偏移量不会超过 64K。

在 16 位 Microsoft 编译器中，一个 near 型指针是通过预先装入的 DS 寄存器来存取缺省数据段的。而一个 far 型指针是通过 ES 寄存器来存取所指定的数据段的，ES 寄存器则是在使用该指针之前由一个存储选择器即时装入的，一个 huge 指针能访问一个内存块中偏移量大于 64K 的地址，它对这种大地址的存取是通过一组选择器来实现的。其算法主要是通过不断地从选择器组中选取正确地选择器，使偏移量相对于该选择器是正常的来实现的。

这就是 64K 的分段内存的局限性，如果将这种 huge 16 位段指针的代码与使用 near 型的 32 位指针的代码组比较，结果是 32 位的版本运行时间要快 5 倍，当然这个比例不同情况也将不同——这里仅是针对 24 位图象处理而言①。由于速度得以提高，24 位的图象和 24 位显示都将越来越普遍，因而提高这种代码的速度是十分重要的，实际上这也是促使我开发 32 位 Windows 技术的原因之一。

对这种代码来说，平板(flat)内存模型与分段的内存模型相比要有明显的优势。但在调试时，平板内存模型比较麻烦：它不能象使用保护段那样有效地使硬件陷入错误。在 80x86 上段保护方式下的代码中，若有一个错误指针或数组没有结束符，则会导致 Trap D 来指示错误，而平板内存模型下同样的情形，并不会指示出这种错误，这时就很难发现问题究竟出自哪里，这时就要求程序员必须具有较高的编程技术②。

显然，32 位编程很有效地提高了速度，这必然导致了对从其它平台向 Windows NT 移植的需求，但也同时要求在编码时比 16 位 Windows 下更为谨慎些，因为这时候我们丧失了能捕获多种错误的段级硬件保护，所以只能注意编码使大部分错误尽可能在编译时被发现。

1.2 Windows 向 Win32 移植指南

首先回顾一下 Microsoft 关于从 Windows 3.1 向 Win32 移植的指南，若希望了解更多详细的情况，可参阅 Microsoft Win 32 SDK for Windows NT 的 Programming Techniques 一卷。

Microsoft 建议在从 16 位 Windows 向 Win32 移植时使用自顶向下的方法。它与自底向上方法的区别在于：自底向上方法需要极大的耐心。自底向上方法首先要从当前的 Windows 内存模型转移到 Win32 下，用 C 对所有的函数重新编码，然后仔细察看所有的代码，最后在 Win32 下重编译。

---

① 也许差别并不很明显，实际上，任何对大块内存的操作，通过使用 near 型的 32 位指针来代替 far 型的 16 位指针，其速度会得到很大提高。但例如 256 种颜色的独立于设备的位图(DIBs)中使用了调色板，调色板比较小，因而处理颜色的代码不需要使用很大的图象数据。24 位的 DIBs 中没有调色板，图象中的每个像点有 3 字节的颜色信息。

② 在 Windows NT 系统下的保护级别一般是页级的，其粒度为 4K，对指针的使用如果足够注意的话，硬件也会处理这种问题，另一方面，每个过程都有它自己的地址空间，因而用户不必担心被其它程序越界写，如果每次都初始化变量，则数组大小错误在平板内存模式下也能被发现。