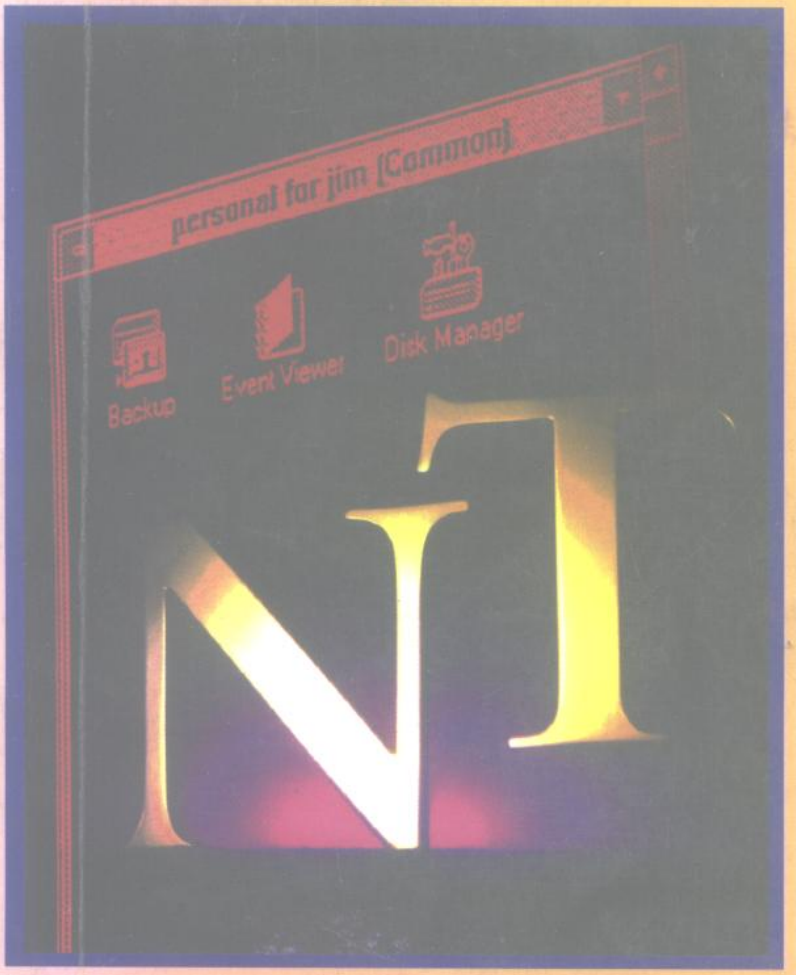
# 精通Windows NT编程技术

## Mastering

# *Windows NT*

## Programming




**Brian Myers**
**Eric Hamer**

李成辉　周长缨　译
刘锦德　校

SYBEX

電子工業出版社

Mastering Windows NT Programming

Brian Myers Eric Hamer

# 精通Windows NT编程技术

李成辉 周长缨 译
刘锦德 校

電子工業出版社

（京）新登字055号

## 内 容 提 要

本书是为计算机软件编程人员所写的有关32位操作系统Windows NT的专著。全书共分十七章，主要涉及以下内容：Windows NT的内部结构和其如何工作；建立Windows NT程序的过程；Windows NT编程中的关键成分：线程、进程、同步、管道、异常处理、内存管理等；WINDOWS NT对过去Windows API方面的增强与扩充，包括GDI、宏文件、位图和调色板、动态链接和文件入/出命令等。

本书适用于具有C语言和Windows编程初步知识的计算机程序设计人员、软件工作者及大专院校师生。



**精通Windows NT编程技术**

[美] Brian Myers Eric Hamer 著

李成辉 焦长缨 译

刘锦德 校

责任编辑 陆伯雄

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

北京顺义李史山胶印厂

开本：787×1092毫米 1/16 印张：62.125 字数：1490千字

1994年10月第1版 1994年10月第1次印刷

印数：5000 册 定价：90.00 元

ISBN 7-5053-2341-5/TP.661

# 出 版 说 明

计算机科学技术日新月异。为了引进国外最新计算机技术，提高我国计算机应用与开发的水平，中国电子工业出版社与美国万国图文有限公司合资兴办的北京美迪亚电子信息有限公司取得了美国SYBEX公司的独家版权代理。SYBEX公司授权本公司通过电子工业出版社等出版机构全权负责在中国大陆出版该公司的中文版和英文版图书。现在与广大读者见面的是最近推出的第一批图书。今后我们还将陆续推出SYBEX公司的最新计算机图书和软件，为广大读者提供更好的服务，传递更多的信息。

美国SYBEX公司是世界著名的计算机图书出版商，该公司自1976年创办开始，其宗旨就是通过出版有效的、高质量的图书向计算机用户介绍实用技巧。我们优选翻译出版的图书是SYBEX公司的最新计算机图书，并采用了该公司提供的电子排版文件，从而提高质量并大大缩短了图书的出版时间，从根本上改变了以往翻译版图书要落后原版书较长的“时差”现象，这在电子技术日新月异的时代具有深远意义。

北京美迪亚电子信息有限公司

1993年11月

# 引　言

作为本书的开始，我们首先自问：Windows NT这一新型操作系统有哪些特色成份具有强大的功能，而对于那些从16位Windows环境转移到32位Windows平台上工作的程序员来说又是最陌生的？本书的目录实际上已经给出了这个问题的答案，即这些特色成份是：线程、进程、同步、管道、结构化异常处理、虚存管理、内存映射文件、增强型元文件、世界坐标系转换、系统安全性和Unicode。然而，我们的目标并非只是对这些新特色成份作知识性的传授。全书在内容上着力之处是：使得那些新命令能让读者更容易理解。如果读者还未曾在其它系统中接触过管道或执行过异步I/O操作，那么与它们初识时就会感到它们的用意何在使人不太清楚，至于它们的实现细节更是令人糊涂。所以，本书在说明问题时追求条理分明地把握住细节，如某一特色成份所适用的种种情况，造成外表不一致的原因；并以较大篇幅来将每一个“为什么”引向各个“怎么干”。

本书各个章节的组织结构，反映了我们在传授时重点放在说理透彻上。书中大部分章节均由三个部分构成：即“概念”、“命令”和“代码”。首先是介绍基本概念，然后讨论相关的Windows函数，最后给出实际的程序例子。这样安排各章内容，可以使读者更容易找出对自己最合适的阅读进程。为了对整个系统有个全局印象，读者只需要阅读各章的“概念”一节。然后，就可以去细读与自己的工作最为有关的章节。如果读者已经熟悉结构化异常处理，那么就可以跳过其概念部份而直接去阅读相关的命令；如果读者急于生成自己的代码，也可以首先从“代码”一节开始，当遇到疑难问题时才回过头来略读一下相应的说理内容。

## 本书读者对象

本书是为那些已经掌握了C语言并且具有一定的 Windows 编程经验的程序员所编写的。我们假定读者已经具备了一些基本概念，如消息驱动程序、对话框和设备环境等。这些特色成份自早期的Windows版本开始就未曾改变过，而且读者在需要帮助时可以从其它有关书籍中找到它们。所以，本书一开始就将直接介绍 Windows NT 主要的新特色成份，如线程和进程等。如果读者正想往32位的Windows平台转移，并且需要对其新的编程格局有较全面的了解，那么本书将是最合适的了。（如果读者需要了解 Windows编程技术的基本概念和结构，建议参阅Brian Myers和Chris Doner合著的《Windows 3.1程序员导引》，该书由SYBEX公司于1992年出版。）

## 硬件环境要求

要运行Windows NT系统，需要一台具有32位处理器的机器。NT可以运行在基于

Intel 80386/25 CPU的计算机上，也可以运行在具有某些RISC CPU（例如，MIPS R4000和DEC Alpha）的机器上。这时还需要一个分辨率至少为VGA的图形适配器和显示器、一个硬盘（至少有70MB自由空间）、一个高密软盘驱动器（只用于Intel型机器）以及8MB RAM空间。为了进行编程，还需要一支鼠标、一个SCSI CD-ROM、两倍的内存空间以及至少再提供30MB额外的磁盘空间。同时，带有Windows NT程序设计库的32位的C编译器也是必须具备的。

第十章中的调色板代码的一些特色成份要求视频适配器必须支持256色，否则将看不到什么。第十六章中的声音程序可在任何硬件平台上运行，但如果没有语音卡（例如SoundBlaster），就没有什么效果。

## Borland用户注意事项

本书附带的盘中的源代码，大部份都带有适用于Borland C++4.0 编译器的make文件。在本书付印之时，Borland开发工具的第一个测试版本还未将所有Win32特色成份并入，所以我们无法测试某几个程序。不过，在得到Borland工具的新版本后，读者就可以按照我们提供的实例make文件，编译上述余下的几个程序。

## Windows NT概貌

Microsoft Windows NT 是一个真正的32位操作系统，所以它能充分利用先进的CPU芯片的能力，而这是DOS所永远办不到的。与早期Windows版本不同，NT不再运行于另一个系统之上，所以它不必像以前的版本那样总要费力地解决旧的基础层所带来的限制。Windows NT仍然保持了与16位Windows相同的图形用户界面（GUI），但是其内部实现方法已大不相同。其中，新的特色成份包括：

- 抢占式多任务
- 32位操作
- 虚存和受保护的地址空间
- 规模可变，以便能够运行在不同的平台上，后者包括RISC工作站和多处理器计算机
- 安全系统，可以用来识别用户、保护系统资源和检查用户行为
- 内藏网络功能，可实现共享文件和共享打印机
- 支持多种网络（例如Banyan、Novell）和网络传输协议（NetBEUI、TCP/IP）.
- 具有多个子系统，它们能分别运行 DOS程序、16位Windows程序、POSIX程序和基于字符的OS/2程序
- 新的文件系统NTFS，允许使用长文件名，并具有更好的安全性
- 新的从属性应用，用来管理用户帐号和网络服务

## 本书内容的组织

第一章描述了Windows NT的内部结构。NT与DOS大不相同，它从最初开始就被设计成模块化、可扩充、与设备无关、多任务和安全的系统。此章还描述了NT的基本构成成份及其相互作用。

第二、三章引导编程人员熟悉新的工具和新的要求。其中，介绍了建立Windows NT程序的步骤，并说清了为16位Windows所编写的代码与为32位Windows所编写的代码两者之间的差异。

第四章到第七章描述了Windows NT编程技术的核心成份，这些对于Windows程序员来说是全新的特色成份。掌握了线程、进程、同步对象、管道、结构化异常处理、内存映射文件和虚拟内存管理技术之后，读者就可以编写出充分利用这些新特色成份的32位应用程序，而正是这些特色成份使Windows NT具有十分的吸引力。

第八章到第十二章转为介绍Windows程序员所熟悉的旧的API中的特色成份，Windows NT则已予以增强和扩充。现在，GUI可以支持Bezier曲线、路径、可缩放的区域、连接方式和化妆笔等。新增加的几个命令使位图更加灵活，还提供了一种新的增强型格式使源文件更加独立于设备。此外，新增加了坐标转换层，使旋转、缩放和错移图像的操作更加容易。动态连接库具有更完善的入口点机制和改进的内存管理功能。文件I/O命令也已发展成为充满特色的API。

第十三章到第十六章的主题对于Windows来说也并非新内容，但它们在Windows NT环境下成为更受重视的特色成份。为保护每个程序内存空间而设的私有地址空间，使程序间不能直接共享数据，所以DDE管理库（DDEML）成了进程间通信的重要渠道。OLE和多媒体机制都向系统中引入了新的数据类型；由于Windows NT将运行在功能更强的机型上，因此，使它具备一些处理密集使用CPU的数据（例如波形声音和动画）的能力是很恰当的。

最后一章转为介绍Windows NT编程技术方面更专门性的主题。并非所有程序都需要处理安全结构、在不同的网络API之间作出选择、使用Unicode码或者在基于字符的控制台窗口中运行；但是，这些特色也都是新系统吸引用户的方面。

## 安装配套软盘

本书备有一张配套软盘，其中含有所有实例程序的源代码。源程序共35个，将占用一兆多字节空间。它们存放于一个名为programs.exe的能自行提取出的LHA文档文件之中。为了把软盘内容提取出来并放到硬盘上，请键入下面的命令：

```
programs c:\masternt
```

这时，该文档文件负责建立对应于各章的目录，它们放在命令行中给出的目录之下。

## 印刷上的约定

我们使用了两种不同的字体，以区分来自C语言的命令和Windows库的命令。“Windows字体”是用粗体字，用来表明是为Win32 API所定义的名字和符号：例如CreateNamePipe、LOWORD。“程序字体”用于程序中的其它成份，包括函数名、C保留字和变量；例如WinMain、atoi、if和bResult等。（本中译本保留了粗体字——译注。）

某些符号以大写字母表示，它们是定义于 Windows头文件中的标识符。这包括数据结构（例如PAINTSTRUCT）、消息名（例如WM_COMMAND）和新的变量类型（例如HPEN和DWORD）。

在介绍和定义重要单词时，将它们用斜体字印出。尖括号“<like these>”用于表示语法上的空白区间，这一区间由程序员负责填写适当的名字或正文。

在命名变量时，我们遵从通用的“匈牙利标记法”，该标记法约定用一串字母作为前缀，以指出数据类型。前缀中包括了下列缩写词：

| 前缀 | 类型 |
|---|---|
| a | Array（数组） |
| b | BOOL（int） |
| by | BYTE（unsigned char） |
| cb | Count of bytes（字节数） |
| ch | char |
| d | double |
| dw | DWORD（unsigned long） |
| fl | float |
| fn | Function（函数） |
| h | HANDLE（void *） |
| i | int |
| l | LONG（long） |
| lp | Long（or far）pointer（长指针或远指针） |
| n | short |
| p | Pointer（指针） |
| sz | String ending with 0（null-terminated）〔以0（NULL）结尾的串〕 |
| u | UNIT（unsigned int） |
| w | WORD（unsigned short） |
| x | （Anx coordinate（usually int）〔x 坐标（通常为int型）〕 |
| y | （Ay coordinate（usually int）〔y 坐标（通常为int型）〕 |

通常，将把指针前缀添加到某些其它类型的前缀之前。例如，pdw是指向DWORD型数据的指针。大写字母表示的单词是普通的C数据类型的同义词。Windows定义这些同义词，是为了使变量定义能表达出更多的信息。例如，HANDLE就比unsigned int 更具有描述性。

Windows NT 不区分近指针和远指针；所有指针的大小均相同。然而，为了兼容，头文件定义中仍然保留了旧式的“lp“前缀（例如LPSTR）。

前缀的使用有助于避免类型不匹配。假设bResult是一个布尔型变量，而dwWritten是一个4字节的双字型变量。它们的定义形式为：

```
BOOL      bResult;        //error   detecetd?
DWORD     dwWritten;      //bytes   written
```

深入到某个子函数中后，程序员可能会因忘却而把某一值赋给一个不相配的变量。

```
dwWriten=bResult;
```

前缀的使用可使这一错误更易于暴露；如果只是简单地使用下述表达形式，就不易发现它：

```
Written=Result;
```

富于信息的前缀表达式使代码更易于理解，所以命名规则对于共同开发同一个应用程序的程序设计小组极有帮助。

在调用函数时，我们总是把括弧与函数名紧接在一起。而在定义函数时，则总是要在函数名和括弧之间插入一个空格：

```
MyFunction   (bMyFlag);                    //function   call
void  MyFunction  (BOOL  bMyFlag)          //function   definition
```

所以，插入这一空格是为了方便在程序中搜索何处定义了一个函数，而跳过所有只是调用函数之处。

# 目 录