C++

黑客编程揭秘与防范（第2版）

冀云 编著

讲解Windows安全和网络编程知识，包括注册表、系统服务、文件读写、进程和线程等

介绍Windows下PE格式、调试技术、挂钩技术等各种软件安全知识

Windows内核编程以及软件逆向知识，Windows病毒分析、反汇编、脱壳、免杀、破解

人民邮电出版社
POSTS & TELECOM PRESS

C++黑客编程揭秘与防范（第2版）

冀云 编著

人民邮电出版社
北京

图书在版编目（CIP）数据

C++黑客编程揭秘与防范 / 冀云编著. -- 2版. -- 北京 : 人民邮电出版社, 2015.2
ISBN 978-7-115-38057-9

Ⅰ. ①C… Ⅱ. ①冀… Ⅲ. ①C语言－程序设计 Ⅳ. ①TP312

中国版本图书馆CIP数据核字(2015)第018573号

内 容 提 要

市面上关于黑客入门的书籍较为繁多，比如黑客图解入门、黑客工具详解、黑客木马攻防等。但是，对于很多读者来说，可能并不是单单掌握简单的工具使用就能满足的。很多读者学习黑客知识是为了真正掌握与安全相关的知识。与安全相关的知识涉及面比较广，包括数据安全、存储安全、系统安全、Web 安全、网络安全等，本书围绕 Windows 系统下应用层的开发来介绍一些关于安全方面的知识。

本书是《C++黑客编程揭秘与防范》的升级版，在前书的基础上新添加了一些内容，同时也删除了一些过时的内容。本书以 Win32 应用层下安全开发为中心，介绍 Windows 系统下的安全开发。

本书介绍了操作系统的相关操作，比如进程、线程、注册表等知识。当读者掌握了关于进程、线程、注册表等相关的开发知识后，就可以把一些常规的操作进程、注册表、文件等用代码进行实现，这样，一些日常的操作可与学习的编程知识相结合。除了操作的知识外，本书还介绍了关于网络应用程序的开发，了解 Winsock 的开发后，读者就会明白在应用层客户端与服务器端通信的原理。当然，本书除了介绍 Win32 开发外，还介绍了 PE 结构、调试接口、逆向等相关的知识。本书的最后部分介绍了关于恶意程序、专杀工具、扫描器等工具的开发。读者只要将前面章节的知识掌握后，后面的实例部分就水到渠成了。

◆ 编　　著　冀　云
责任编辑　张　涛
责任印制　张佳莹　焦志炜

◆ 人民邮电出版社出版发行　　北京市丰台区成寿寺路 11 号
邮编　100164　　电子邮件　315@ptpress.com.cn
网址　http://www.ptpress.com.cn
北京昌平百善印刷厂印刷

◆ 开本：787×1092　1/16
印张：26
字数：622 千字　　2015 年 2 月第 2 版
印数：9 501- 1 3000 册　　2015 年 2 月北京第 1 次印刷

定价：55.00 元

读者服务热线：(010)81055410　印装质量热线：(010)81055316
反盗版热线：(010)81055315

序

我与冀云兄因黑客反病毒论坛结识于一年前，在认识初期就能感觉到冀云兄是一个非常踏实且又富有思想的人，对安全编程的诸多方面也有自己独到的认识，这点令我十分欣赏。认识几个月后，通过一次无意的聊天，我有幸读到冀云兄《C++黑客编程揭秘与防范》第1版，而后承冀云兄高看，才得以诞生此序。

通过阅读《C++黑客编程揭秘与防范》第1版，我有一种相见恨晚的感觉。这本书从最基本的Windows编程到Windows下的各种安全编程技术都有涉及，例如PE文件、DLL注入技术、各种Hook技术、后门编写的技术关键点，乃至像MBR的解析这种很难涉及的点与Rootkit编程这样比较深入的面，都有恰到好处的介绍与详解。

因此，就整书而言，将诸如文件/注册表操作、网络通信、PE文件、Rootkit、逆向工程等数个知识点有效组织在一起，是一个非常巨大的工程。这一点对于有过类似写作经验的我来说，体会尤其深刻。但是不得不说，作为读者，我真的非常幸运，这本书基本上完成了这个艰巨的任务。就我个人而言，这本书至少可以被当成一部“技术字典”来使用。当我在实际的工作中对某种技术生疏后，可以拿起这本书翻一翻，顿时会感觉受益匪浅。

从整书的结构以及知识的组织方式来看，不难发现，这其实是一本相当重视初学者技术的图书。作者在第1章中对于工作环境的搭建以及对应IDE的使用都做了必要的介绍，而后又通过第2章使用一个非常有趣且简单的例子教读者如何打造一个木马的雏形，这些无不体现出了作者对于基础薄弱的读者的细心照顾。

除此之外，当前的政策环境以及社会整体的大环境都对信息安全产业释放了大量的利好信号，无论是十八大时国家将信息安全提到国家战略层面，还是随后发生的著名“棱镜门”事件，抑或是当前势不可挡的移动互联网大潮，都在预示着信息安全领域在未来势必将摆脱“边缘群体”，进而成为“主流群体”中重要的一员，这些改变势必将极大地加剧当前信息安全领域人才的稀缺现状。但是，我相信本书定会为中国的信息安全领域崛起贡献一份力量，进而使得更多的读者从信息安全的“门外汉”成为“圈内人”，以缓解现在信息安全领域人才稀缺的现状。

——任晓珲[A1Pass]，北京蓝森科技有限公司创始人，
15PB计算机高端培训品牌创始人，《黑客免杀攻防》作者

甲午年正月十二于北京朝阳

前言

备受关注的黑客到底是什么

“黑客”已经成为一个热门的话题，“黑客事件”更是已经受到各大媒体的关注，甚至很多媒体对黑客事件进行不负责任的报道与炒作。从传统媒体到互联网媒体，从平面媒体到多元化的媒体平台，都在报导黑客事件，报道新爆发的蠕虫、病毒等一系列相关事件。

在各种媒体的炒作下，越来越多的年轻人都在追逐“当黑客的潮流”，很多热爱计算机的年轻人都追求自己能成为黑客。很多人都期望自己能一蹴而就成为一名顶级的黑客。很多人读着“头号电脑骇客”凯文米特尼克的传奇故事，在波涛汹涌的心情下，幻想着自己也能成为网络上出神入化的黑客风云人物。为此，很多“追黑”的网友深入各大黑客网站下载各种黑客工具来进行入侵、破坏（在用别人工具的同时，说不定也会遭到一些不怀好意的黑客网站的暗算，比如自己用的黑客工具本身就被放置了后门），从而满足自己追求成为黑客的心理。

在追求成为黑客的过程中真地学到了多少知识，自身的技术水平距离真正的黑客到底还有多远，这是很多只会使用工具的黑客应该冷静下来认真思考的问题。他们应该思考自己是否了解 TCP、UDP、ICMP 等 TCP/IP 中常用协议的结构与协议原理，是否知道 ARP 欺骗的原理，是否能独立开发并完成一个后门等。

什么是黑客？百度百科里黑客的含义如下（摘自百度百科，且有改动）：

“热衷研究、撰写程序的专才，精通各种计算机语言和系统，且必须具备乐于追根究底、穷究问题的特质。‘黑客’一词是由英语 Hacker 音译出来的，是指专门研究、发现计算机和网络漏洞的计算机爱好者。早期在美国的电脑界是带有褒义的。”

看到上面百度百科给出的黑客含义后，是不是很多只会使用工具的所谓的黑客能明白一个道理，即黑客是要会编写程序的？

再看一下百度百科里对只会使用工具的黑客的解释（摘自百度百科）：

“脚本小子，英文 script kiddie 或 script boy。脚本小子指的是用别人写的程序的人。脚本小子是一个贬义词，用来描述以黑客自居并沾沾自喜的初学者。”

那些自以为是的工具黑客只不过是一个“脚本小子”，还是一个被大家所“鄙视”的“小子”。是不是心里觉得不是很舒服了？是不是觉得自己应该提高了？如果是的话，那么就请抛开以前当工具黑客的想法，开始编写黑客工具吧！

心态影响成长

新手可能会问：编写自己的黑客工具是不是很难，是不是要懂编程语言，要懂哪种编

程语言呢？笔者的回答是肯定的。抛开用工具当黑客的想法，其实是让大家抛开浮躁的想法，认真地学一些真正的技术，哪怕只是一些入门的基础。想做黑客就要有创新、研发的精神，如果只是做一个只会用软件的应用级的计算机使用者，那么必定永远达不到黑客级的水平，因为工具人人都会用，只是比别人多知道几个工具而已。抛开浮躁，静下心来从头开始学习基础，为将来的成长做好足够的准备。

攻防的广义性

黑客做得最多的就是“入侵”，这里所说的入侵不是一个狭义上的入侵，因为它不单单针对网络、系统的入侵。这里说的是一个广义上的入侵，“入侵”一词是指“在非授权的情况，试图存取信息、处理信息或破坏系统以使系统不可靠、不可用的故意行为。”由此可以看出，入侵并非单指网络或系统。这里说的“入侵”包括两个方面，一个是针对网络（系统）方面的入侵，另一个是针对软件的入侵。网络的入侵是通常意义上的入侵，而软件的入侵通常就是人们说的软件破解（包括漏洞挖掘等内容）。无论是侵入别人系统，还是破解某款软件，都是在非授权的情况下得到相应的权限，比如系统权限或者软件的使用权限。

本书讲点什么

本书针对“网络入侵”和“软件入侵”两方面来介绍黑客编程，从攻防两个角度来学习黑客编程的知识，通过一系列知识体系完成“黑客编程”的养成计划。

本书会介绍大量的基础知识，这些基础知识看起来与普通的应用程序编程没有什么差别。其实，所谓 “黑客编程”（也称为“安全编程”），“是指采用常规的编程技术，编写网络安全、黑客攻防类的程序、工具”。因此，普通的编程技术与黑客编程技术并没有本质的差别，只是开发的层面不同。普通的编程注重的是客户的需求，而黑客编程注重的则是攻与防。

黑客编程有其两面性，按照攻防角度可以分为“攻击类入侵编程”和“防范类安全编程”。结合上面提到的“网络”和“软件”两方面来说，常见的“网络攻击”程序有扫描器、嗅探器、后门等；常见的“软件攻击”程序有查壳器、动态调试器、静态分析器、补丁等（这些工具是一些调试工具和逆向分析工具，因为软件破解、漏洞挖掘等会用到这些调试工具，因此称其为“软件攻击”工具）。常见的“网络（系统）防范”程序有“杀毒软件”、“防火墙”、“主动防御系统”等；常见的“软件防范”程序有“壳”、 “加密狗”、“电子令牌”等。

根据前面提到的攻防两方面的内容，本书会涉及扫描器的开发、后门的开发、应用层抓包器的开发等黑客攻防方面的相关内容。本书还会讲解关于软件方面的知识，主要涉及 PE 结构、加壳、脱壳、逆向分析等一系列相关知识。由于技术的两面性，希望读者有一个良性的学习心态。

读者能从本书中得到什么

通过本书，读者能学到 Windows 下基于消息的软件开发、基于 Winsock 的网络应用程序的开发、软件逆向分析和调试知识等一系列的编程、调试及安全知识。在学习的过程中，

读者应该大量阅读和参考其他相关资料，并且一定要亲自动手进行编程。编程绝对不是靠看书能够学会的！

通过本书的指导，再加上自身实践和练习，读者可以具备基本的 Windows 下的应用程序开发、网络程序开发的能力，基本的系统底层开发能力。除了相关开发能力外，读者还能具备初级的病毒分析能力、软件保护等相关的安全知识。

如何无障碍阅读此书

阅读本书的读者最好具有 C 和 C++编程的基础知识，有其他编程语言基础知识的读者也可以无障碍阅读。无编程知识的读者阅读本书的同时，只要学习了本书中涉及的相关基础知识，同样可以阅读本书。

本书涉及范围较多，知识面比较杂，但是本书属于入门级读物，专门为新手准备，只要读者具备一定的基础知识，即可顺利进行阅读。在阅读本书的基础上，读者可以接着学习更深层次的知识，希望本书能帮助读者提高自身的能力。

建议：请读者深入学习操作系统原理、数据结构、编译原理、计算机体系结构等重要的计算机基础知识。

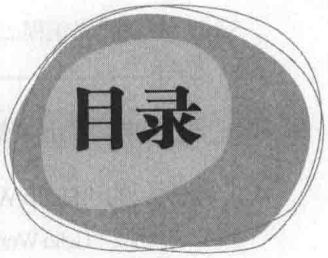
免责

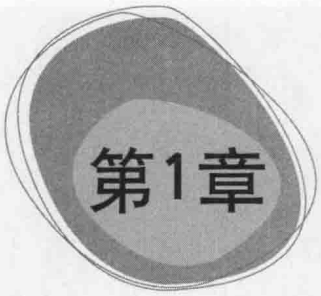
本书属于入门级图书，无法保证读者成为黑客。作者本人也不是黑客，但是至少要有一个成为黑客的想法和成为黑客的动力。因此，如果本书没能达到读者所期待的目标，那么也请恕笔者无奈，笔者只是带领读者入门。

本书中内容主要用于教学，指导新手如何入门、如何学习编程知识，从编程的过程中了解黑客编程的基础知识。请勿使用自己的知识做出有碍公德之事，在准备通过技术手段进行蓄意破坏时，请想想无数“高手”的下场。读者如若作奸犯科，与作者本人和出版社无任何关系，请读者自觉遵守国家法律。

由于作者水平有限，书中难免会有差错，敬请谅解。中肯取代无礼，客观代替谩骂。

编辑联系邮箱：zhangtao@ptpress.com.cn。





第1章 黑客编程入门

读者是否曾经用别人开发的工具尝试“入侵”，是否希望开发出自己的黑器？本章将介绍Win32开发平台的开发基础，带领读者进入Windows编程的大门。

Windows是一个庞大而复杂的操作系统，它提供了丰富而强大的功能，不但操作灵活方便，而且有众多的应用软件对其进行支持。Windows因有众多软件的支持，长期雄霸于PC系统。之所以有众多软件的支持，是因为Windows提供了良好的应用程序开发平台（接口）、完整的开发文档和各种优秀的开发环境。对于一个程序员来说，除了要掌握基本的开发语言以外，还要掌握具体的开发环境和系统平台的相关知识。在掌握编程语言和开发环境等知识后，还要掌握调试技术以及各种调试分析工具。同样，Windows操作系统提供了良好的调试接口，并且有非常多的调试工具。

本章主要介绍Windows的消息机制，Windows下的开发工具、辅助工具，还有调试工具。本章的目的在于对Windows消息机制进行回顾，它是Windows开发的基础，方便后续章节内容的讲解。本章对于Windows编程的一些基本概念不会进行过多的介绍。除了对消息机制进行回顾外，本章还要介绍集成在Visual C++（VC6）中的调试工具和其他一些开发辅助工具。

1.1 初识Windows消息

大部分Windows应用程序都是基于消息机制的（命令行下的程序并不基于消息机制），熟悉Windows操作系统的消息机制是掌握Windows操作系统下编程的基础。本节将带领读者认识和熟悉Windows的消息机制。

1.1.1 对消息的演示测试

在真正学习和认识消息之前，先来完成一个简单的任务，看看消息能完成什么样的工作。首先写一个简单的程序，通过编写的程序发送消息来关闭记事本的进程、获取窗口的标题和设置窗口的标题。

程序的具体代码如下：

```
void CMsgTestDlg::OnClose()
{
  // TODO: Add your control notification handler code here
  HWND hWnd = ::FindWindow("Notepad", NULL);
  if ( hWnd == NULL )
```

```
    {
      AfxMessageBox("没有找到记事本");
      return ;
    }

    ::SendMessage(hWnd, WM_CLOSE, NULL, NULL);
}

void CMsgTestDlg::OnExec()
{
    // TODO: Add your control notification handler code here
    WinExec("notepad.exe", SW_SHOW);
}

void CMsgTestDlg::OnEditWnd()
{
    // TODO: Add your control notification handler code here
    HWND hWnd = ::FindWindow(NULL, "无标题 - 记事本");
    if ( hWnd == NULL )
    {
      AfxMessageBox("没有找到记事本");
      return ;
    }

    char *pCaptionText = "消息测试";
    ::SendMessage(hWnd, WM_SETTEXT, (WPARAM)0, (LPARAM)pCaptionText);
}

void CMsgTestDlg::OnGetWnd()
{
    // TODO: Add your control notification handler code here
    HWND hWnd = ::FindWindow("Notepad", NULL);
    if ( hWnd == NULL )
    {
      AfxMessageBox("没有找到记事本");
      return ;
    }

    char pCaptionText[MAXBYTE] = { 0 };
    ::SendMessage(hWnd, WM_GETTEXT, (WPARAM)MAXBYTE, (LPARAM)pCaptionText);

    AfxMessageBox(pCaptionText);
}
```

编写的代码中有 4 个函数：第 1 个函数 OnClose()是用来关闭记事本程序的；第 2 个函数 OnExec()是用来打开记事本程序的，主要是测试其他 3 个函数时可以方便地打开记事本程序；第 3 个函数 OnEditWnd()是用来修改记事本标题的；第 4 个函数 OnGetWnd()是用来获取当前记事本标题的。程序的界面如图 1-1 所示。

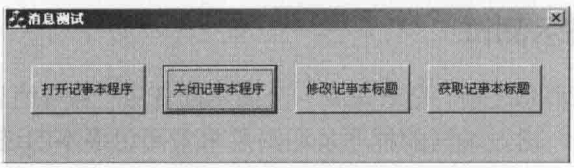


图 1-1　消息测试窗口

简单测试一下这个程序。首先单击“打开记事本程序”按钮，出现记事本的窗口（表示记事本程序被打开了）；接着单击“修改记事本标题”按钮，可以发现记事本程序的窗口标题改变了；再单击“获取记事本标题”按钮，弹出记事本程序窗口标题的一个对话框；最后单

击“关闭记事本程序”按钮，记事本程序被关闭。

1.1.2 对 MsgTest 代码的解释

上面的代码中要学习的 API 函数有两个，分别是 FindWindow()和 SendMessage()。下面看一下它们在 MSDN 中的定义。

FindWindow()函数的定义如下：

```
HWND FindWindow(
  LPCTSTR lpClassName,  // class name
  LPCTSTR lpWindowName  // window name
);
```

FindWindow()函数的功能是，通过指定的窗口类名（lpClassName）或窗口标题（lpWindowName）查找匹配的窗口并返回最上层的窗口句柄。简单理解就是，通过指定的窗口名（窗口名相对于窗口类来说要直观些，因此往往使用的是窗口名）返回窗口句柄。FindWindow()函数有 2 个参数，分别是 lpClassName 和 lpWindowName。通过前面的描述，该函数通常使用的是第 2 个参数 lpWindowName，该参数是指定窗口的名称。在例子代码中，为程序指定的窗口名是“无标题—记事本”。“无标题—记事本”是记事本程序打开后的默认窗口标题，当 FindWindow()找到该窗口时，会返回它的窗口句柄。例子代码中也使用了 lpClassName（窗口类名），在窗口的名称会改变的情况下，只能通过窗口类名来获取窗口的句柄了。FindWindow()函数返回的窗口句柄是为了给 SendMessage()函数来使用的。

SendMessage()函数的定义如下：

```
LRESULT SendMessage(
  HWND hWnd,      // handle to destination window
  UINT Msg,       // message
  WPARAM wParam,  // first message parameter
  LPARAM lParam   // second message parameter
);
```

该函数的作用是根据指定窗口句柄将消息发送给指定的窗口。该函数有 3 个参数，第 1 个参数 hWnd 是要接收消息的窗口的窗口句柄，第 2 个参数 Msg 是要发送消息的消息类型，第 3 个参数 wParam 和第 4 个参数 lParam 是消息的两个附加参数。第 1 个参数 hWnd 在前面已经介绍过了，该参数通过 FindWindow 获取。

程序的代码中，SendMessage()函数的第 2 个参数分别使用的是 WM_CLOSE 消息、WM_SETTEXT 消息和 WM_GETTEXT 消息。下面来看这 3 个消息的具体含义。

WM_CLOSE：将 WM_CLOSE 消息发送后，接收到该消息的窗口或应用程序将要关闭。WM_CLOSE 消息没有需要的附加参数，因此 wParam 和 lParam 两个参数都为 NULL。

WM_SETTEXT：应用程序发送 WM_SETTEXT 消息对窗口的文本进行设置。该消息需要附加参数，wParam 参数未被使用，必须指定为 0 值，lParam 参数是一个指向以 NULL 为结尾的字符串的指针。

WM_GETTEXT：应用程序发送 WM_GETTEXT 消息，将对应窗口的文本复制到调用者的缓冲区中。该消息也需要附加参数，wParam 参数指定要复制的字符数数量，lParam 是接收文本的缓冲区。

例子代码在 VC6 下进行编译连接，生成可执行文件后，可以通过按钮的提示进行测试，以便感性认识消息的作用。

1.1.3　如何获取窗口的类名称

编写程序调用 FindWindow()函数的时候，通常会使用其第 2 个参数，也就是窗口的标题。但是有些软件的窗口标题会根据不同的情况进行改变，那么程序中就不能在 FindWindow()函数中直接通过窗口的标题来获得窗口的句柄了。而窗口的类名通常是不会变的，因此编程时可以指定窗口类名来调用 FindWindow()函数以便获取窗口句柄。那么，如何能获取到窗口的类名称呢？这就是将要介绍的第 1 个开发辅助工具——Spy++。

Spy++是微软 Visual Studio 中提供的一个非常实用的小工具，它可以显示系统的进程、窗口等之间的关系，可以提供窗口的各种信息，可以对系统指定的窗口进行消息的监控等。它的功能非常多，这里演示如何用它来获取窗口的类名称。

打开开始菜单，在 Visual Studio 的菜单路径下找到 Spy++，打开 Spy++窗口，如图 1-2 所示。

选择工具栏中的“Find Window”按钮，如图 1-3 所示。

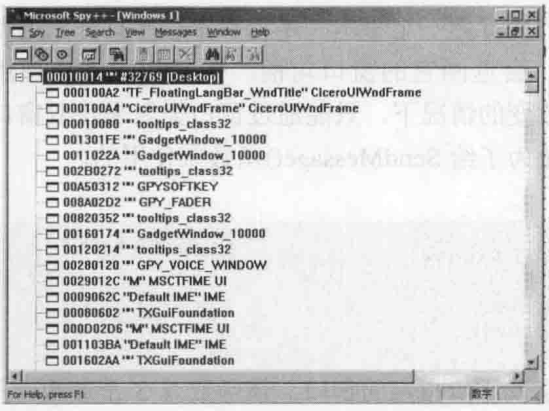


图 1-2　Microsoft Spy++窗口

图 1-3　Find Window 按钮

单击“Find Window”按钮，出现如图 1-4 所示的窗口。

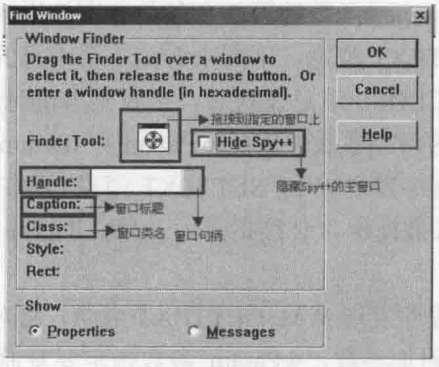


图 1-4　Find Window 窗口

在图 1-4 中，用鼠标左键单击“Finder Tool”后面的图标，然后拖曳到指定的窗口上，会显示出“Handle”（窗口句柄）、“Caption”（窗口标题）和“Class”（窗口类名），其中“Class”

是编程时要使用的类名称。

“Hide Spy++”是一个比较实用的功能，它用来隐藏 Spy++主窗口界面。选中该复选框后，拖曳“Finder Tool”后的图标时，图 1-2 所示为窗口将被隐藏。这个功能的实用之处在于，有些应用软件有反 Spy++的功能，隐藏 Spy++主窗口有助于避免被反 Spy++的软件检测到。为什么隐藏 Spy++的“Find Window”窗口会有反检测的功能，反检测的原理是什么？原理很简单，目标程序也是通过调用 FindWindow()函数来查找 Spy++窗口的，如果有该窗口，就进行一些相应的处理。

注：通过 Spy++找到的窗口句柄是不能在编程中使用的，每次打开窗口时，窗口的句柄都会改变。

将“Finder Tool”后的图标拖曳到记事本的标题处，Spy++的 Find Window 窗口显示的内容如图 1-5 所示。

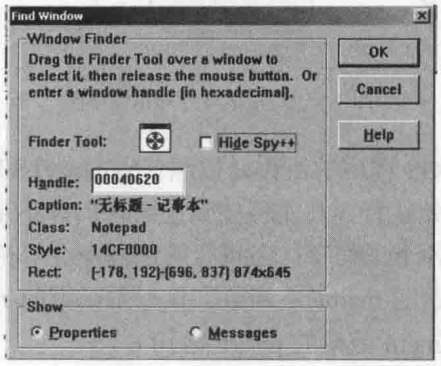


图 1-5 获取到信息的 Find Window 窗口

从图 1-5 中可以得到记事本程序的标题和类名称。当编写程序调用 FindWindow()函数，不能通过程序的标题文本得到窗口的句柄时，可以通过窗口类名称得到窗口的句柄。

1.2 Windows 消息机制的处理

SendMessage()将指定的消息发送给指定的窗口，窗口接收到消息也有相应的行为发生。那么窗口接收到消息后的一系列行为是如何发生的？下面通过熟悉 Windows 的消息机制来理解消息处理背后的秘密。

1.2.1 DOS 程序与 Windows 程序执行流程对比

Windows 下的窗口应用程序都是基于消息机制的，操作系统与应用程序之间、应用程序与应用程序之间，大部分都是通过消息机制进行通信、交互的。要实际掌握 Windows 应用程序内部对消息的处理，必须分析实际的源代码。在编写一个基于消息的 Windows 应用程序前，先来比较 DOS 程序和 Windows 程序在执行时的流程。

1．DOS 程序执行流程

在 DOS 下将编写完的程序进行执行，在执行时有明显的流程。比如用 C 语言编写程序后，程序执行时的大致流程如图 1-6 所示。

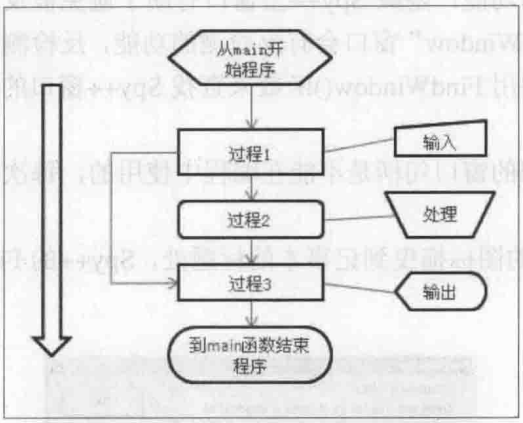


图 1-6　传统 DOS 程序执行流程

在图 1-6 中可以看出，DOS 程序的流程是按照代码的顺序和流程依次执行。大致步骤为：DOS 程序从 main()主函数开始执行（其实程序真正的入口并不是 main 函数）；执行的过程中按照代码编写流程依次调用各个子程序；在执行的过程中会等待用户的输入等操作；当各个子程序执行完成后，最终会返回 main()主函数，执行主函数的 return 语句后，程序退出（其实程序真正的出口也并不是 main 函数的 return 语句）。

2．Windows 程序执行流程

DOS 程序的执行流程比较简单，但是 Windows 应用程序的执行流程就比较复杂了。DOS 是单任务的操作系统。在 DOS 中，通过输入命令，DOS 操作系统会将控制权由 Command.com 转交给 DOS 程序从而执行。而 Windows 是多任务的操作系统，在 Windows 下同时会运行若干个应用程序，那么 Windows 就无法把控制权完全交给一个应用程序。Windows 下的应用程序是如何工作的？首先看一下 Windows 应用程序内部的大致结构图，如图 1-7 所示。

图 1-7 可能看起来比较复杂，其实 Windows 应用程序的内部结构比该示意图更复杂。在实际开发 Windows 应用程序时，需要关注的部分主要是“主程序”和“窗口过程”两部分。但是从图 1-7 来看，主程序和窗口过程没有直接的调用关系，而在主程序和窗口过程之间有一个“系统程序模块”。“主程序”的功能是用来注册窗口类、获取消息和分发消息。而“窗口过程”中定义了需要处理的消息，会根据不同的消息执行不同的动作，而不需要程序处理的消息则会交给默认的系统过程进行处理。

在“主程序”中，RegisterClassEx()函数会注册一个窗口类，窗口类中的字段中包含了“窗口过程”的地址信息，也就是把“窗口类”的信息（包括“窗口过程的地址信息”）告诉操作系统。然后“主程序”不断通过调用 GetMessage()函数获取消息，再交由 DispatchMessge()函数来分发消息。消息分发后并没有直接调用“窗口过程”让其处理消息，而是由系统模块查找该窗口指定的窗口类，通过窗口类再找到窗口过程的地址，最后将消息送给该窗口过程，由窗口过程处理消息。

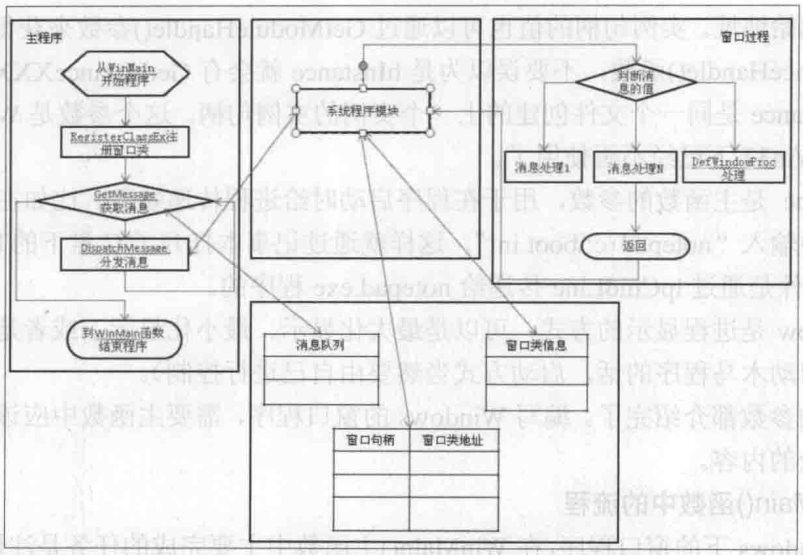


图 1-7 Windows 应用程序执行原理图

1.2.2 一个简单的 Windows 应用程序

相对一个简单的 DOS 程序来说一个简单的 Windows 应用程序，要很长。下面的例子中只实现一个特别简单的程序，这个程序在桌面上显示一个简单的窗口，它没有菜单栏、工具栏、状态栏，只是在窗口中输出一段简单的字符串。虽然程序如此简单，但是也要编写 100 行左右的代码。考虑到初学的读者，这里将一部分一部分地逐步介绍代码中的细节，以减少代码的长度，从而方便初学者的学习。

1. Windows 窗口应用程序的主函数——WinMain()

在DOS时代，在Windows下的命令行的程序，要使用C语言编写代码的时候都是从main()函数开始的。而在Windows下编写有窗口的程序时，要用C语言编写窗口程序就不再从main()函数开始了，取而代之的是WinMain()函数。

既然 Windows 应用程序的主函数是 WinMain()，那么就从了解 WinMain()函数的定义开始学习 Windows 应用程序的开发。WinMain()函数的定义如下：

```
int WINAPI WinMain(
  HINSTANCE hInstance,      // handle to current instance
  HINSTANCE hPrevInstance,  // handle to previous instance
  LPSTR lpCmdLine,          // command line
  int nCmdShow              // show state
);
```

该函数的定义取自 MSDN 中，在看到 WinMain()函数的定义后，很直观地会发现 WinMain 函数的参数比 main()函数的参数变多了。从参数个数上来说，WinMain()函数接收的信息更多了。下面来看每个参数的含义。

hInstance 是应用程序的实例句柄。保存在磁盘上的程序文件是静态的，当被加载到内存中时，被分配了 CPU、内存等进程所需的资源。这样，一个静态的程序被实例化为一个有各种执行资源的进程了。句柄的概念随上下文的不同而不同，句柄是操作某个资源的“把手”。当需要对某个实例化进程操作时，需要借助该实例句柄进行操作。这里的实例句柄是程序装