Broadview
www.broadview.com.cn

天书夜读

驱网核心技术丛书

天书夜读
——从汇编语言到Windows内核编程

从汇编语言到Windows内核编程

谭文 邵坚磊 著　　罗云彬 审校

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
http://www.phei.com.cn

驱网核心技术丛书

从汇编语言到Windows内核编程

谭文 邵坚磊 著 罗云彬 审校

電子工業出版社
Publishing House of Electronics Industry
北京•BEIJING

内 容 简 介

本书从基本的 Windows 程序与汇编指令出发，深入浅出地讲解了 Windows 内核的编程、调试、阅读，以及自行探索的方法。读者在使用 C/C++开发 Windows 程序的基础上，将熟练掌握汇编和 C 语言的应用，深入了解 Windows 底层，并掌握阅读 Windows 内核的基本方法，以及 Windows 内核的基本编程方法。

本书适合使用 C/C++在 Windows 上编程的读者，尤其适合希望加深自己技术功底的 Windows 应用程序员、计算机专业的有志于软件开发的大中院校学生；专业的 Windows 内核程序员，亦可从本书得到超越一般内核程序开发的启发。

图书在版编目（CIP）数据

天书夜读：从汇编语言到 Windows 内核编程 / 谭文，邵坚磊著. —北京：电子工业出版社，2008.10
（驱网核心技术丛书）
ISBN 978-7-121-07339-7

Ⅰ. 天… Ⅱ. ①谭… ②邵… Ⅲ. ①汇编语言—程序设计 ②窗口软件，Windows—程序设计
Ⅳ. TP313 TP316.7

中国版本图书馆 CIP 数据核字（2008）第 136007 号

责任编辑：葛 娜
印 刷：北京智力达印刷有限公司
装 订：北京中新伟业印刷有限公司
出版发行：电子工业出版社
北京市海淀区万寿路 173 信箱 邮编 100036
开 本：787×980 1/16 印张：17.75 字数：393 千字
印 次：2008 年 10 月第 1 次印刷
印 数：5000 册 定价：45.00 元

凡所购买电子工业出版社图书有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系，联系及邮购电话：（010）88254888。

质量投诉请发邮件至 zlts@phei.com.cn，盗版侵权举报请发邮件至 dbqq@phei.com.cn。

服务热线：（010）88258888。

前　言

Windows 是庞大复杂的系统。由于 Windows 并不公开源代码，我们在调试程序的时候，往往就调到自己未知的领域去了。没有 C 代码，只能看到令人眼花缭乱的汇编指令和机器码。我曾对它们望而生畏，敬而远之。尤其在这个黑客、破解、病毒、木马横行的时代，如果作为安全软件的开发者，同样不能期盼病毒的作者提供可以阅读的高级语言代码。

如果那些东西，也和 C 语言一样亲切易懂，那多么好啊！这样的话，即便是 Windows 这样庞大复杂而且封闭的系统，或者是再诡异和隐蔽的破坏技术，至少只要我愿意去探索，对我来说就不再有秘密可言。

其实这个梦想并非不切实际。既然我们能读懂 C 代码，何以就不能读懂汇编呢？很多高手眼中，机器指令和 C 代码一样熟悉。

这本书并没有系统地介绍 Windows 系统底层。但是我尝试寻找正确的方法和手段，为读者打开 Windows 底层知识宝库的大门，使读者可以在其中自由阅读，自己去获取所需知识。

本书强调汇编语言的应用。虽然没有像指令手册那样罗列所有指令的细节，但是让读者在实践中熟悉实用的汇编指令。我一般把汇编语言用于调试和理解程序，在写代码时应用较少。但是汇编是了解没有代码的二进制代码的基础。

这里着重于 Windows 内核编程，包括 WDK、WinDbg 的使用，但是 Windows 内核开发体系庞大复杂，这方面已经有很多的经典书籍可以阅读。在这里我只提到了简单、实用和必要的部分。

对于安全软件的开发者，尤其是有兴趣在 Windows 系统下开发反病毒软件的内核驱动读者，会学到基础的研究方法。

由于这一切都从阅读和理解机器指令出发，所以名为“天书夜读”。

“天书夜读”是通向软件底层技术的一个大门与捷径。这里涉及的每个方面的技术，无论是 CPU 底层架构与机器指令、汇编语言、Windows 内核编程、软件逆向工程还是软件调试，无一不是入门艰难的技术，许多技术者在开始时就被汗牛充栋的庞杂资料吓倒。用一条简洁明确的线，把它们串联起来，使读者可以从最基础入门，循序渐进地接触到高深的技术，并应用于实际。避免一开始就面对浩瀚的文档资料而茫然不知如何入手，也不知如何学以致用，这就是我写这本书的目的。

本书正文的内容是从读者很可能已经遗忘的汇编语言的基础指令开始，介绍 C 语言与汇编指令的关系，为阅读用 C 语言编写的 Windows 内核做准备。然后开始讲解 Windows 内核的 C 语言编程的基础。让读者能熟悉使用 C 语言开发 Windows 内核程序。接着指导读者进行内核的开发和调试环境的配置，并进行了一系列的尝试：阅读 Windows 内核的部分实现代码，并尝试自己去实现它。与此同时，产生了修改已有内核的需求。于是接下来介绍了机器码指令和汇编语言的关系，以及能够解读机器码的反汇编引擎，并以此为基础，介绍了进行 Windows 内核 Hook 开发的方法。之后是 Windows 内核 Hook 的一个利用：我们举出了一个防止病毒木马感染的安全防毒软件的例子。在此之后，简要介绍了更深层次的病毒与安全的对抗：Rootkit 与 HIPS。最后是一些防止代码与技术被阅读者“偷学”的章节。

相信读者读完此书，无论是对汇编语言的掌握，还是 Windows 编程的技术，以及对 Windows 系统的了解和调试程序、查找修正故障的能力，都会得到一个飞跃式的提高。

谭　文

2008 年 4 月 2 日

目　录

入手篇　熟悉汇编

本书的第一部分，将帮助读者消除对汇编的恐惧，熟悉汇编。本部分包括第 1~3 章。稍显枯燥的是，它们和 Windows 内核无关，是纯 C 语言与汇编语言的关系的章节。如果读者已经精通汇编语言，并能顺利阅读汇编代码，请直接跳过本部分。

基础篇 内核编程

本书的第二部分，是编写 Windows 内核程序编程方法的基础。本部分包括第 4~7 章，如果读者对 Windows 内核编程已经有一定的了解，可以跳过本部分；如果读者从未接触过 Windows 内核编程，本部分将指导读者开始 Windows 内核编程，学会使用 WDK，并熟悉内核编程的习惯与方法。

探索篇 研究内核

本书的第三部分，开始探索 Windows 内核程序，并尝试阅读反汇编代码作为指引。本部分包括第 8~10 章。如果读者对 Windows 内核编程已经有一定的了解，这一部分会比较有趣；如果读者从未接触过 Windows 内核编程，读者应该先学习第二部分。能自己编写内核程序并不意味着可以读懂内核，虽然反过来是一定成立的。读懂别人编写的没有代码的程序，比自己编写更困难一些，但的确是值得的。

深入篇　修改内核

这是本书的第四部分。读者已经尝试过探索 Windows 内核程序，并尝试阅读反汇编代码。那么接下来，必须掌握修改内核的方法。每一个 Windows 内核程序，都可以看做 Windows 内核本身的一个“补丁”。有时只需要独立存在，就能起到它的作用；有时却必须对已有的内核二进制代码进行部分修改。本部分包括第 11~13 章，主要介绍的是内核 Hook。

实战篇　实际开发

实战部分是本书最深入和复杂的一部分，包括第 14～17 章。为了让前面练习的成果，在实际应用中产生价值，在这部分我们补充更多的理论知识并尝试用它们去做一点什么。这一部分包括指令分析、硬件基础知识、内核 Hook 的实际开发练习，以及将完成一个用到内核 Hook 的有趣的实例，这个实例有助于计算机阻挡各种病毒和木马的侵袭。

此外，本部分还包括特殊的一章，涉及如何巧妙地编写代码，来防止被其他不受欢迎的读者阅读。这与本书的主旨完全相反，正所谓物极必反。

入手篇

熟悉汇编

本书的第一部分，将帮助读者消除对汇编的恐惧，熟悉汇编。本部分包括第 1~3 章。稍显枯燥的是，它们和 Windows 内核无关，是纯 C 语言与汇编语言的关系的章节。如果读者已经精通汇编语言，并能顺利阅读汇编代码，请直接跳过本部分。

第1章　汇编指令与C语言

本书总是假设读者的“母语”是 C/C++。（如果是 Java 或者 C#，建议读者首先学习 C 语言。）由于 Microsoft 并没有在 Windows 中附带所有的 C 代码，因此读者遇到的很多将是汇编代码。

那么理解汇编代码就成为当前最大的问题。汇编代码之所以看不懂，除了汇编代码可读性本来就比较差之外，更重要的一点原因是，读者的母语是 C 语言。甚至会常常觉得，自己虽然能看懂每一行指令，却完全看不懂一段程序在做什么。

但是实际上，Windows 的内核代码基本上都是由 C 语言代码编译而成的。这些生成的机器码，再被调试器反汇编得到的汇编语言，和 C 语言有着千丝万缕的联系。只要熟悉它们之间的对应关系，像理解 C 语言一样理解汇编代码，就完全是可能的了。

为了强化这种关系，本书有时从汇编代码写出对应的 C 语言，以便于读者理解。

何为软件反工程（逆向工程）

作为编程爱好者、程序员，大多数人从事“软件工程”的工作。这种工作是从人类可以理解的高级语言编译为机器可以理解的机器码的过程。反工程则是相反的，是从机器可以理解的机器码，到人类可以理解的某种形式的过程。其目的在于，通过别人出售的产品得到别人的工程技术。

反工程往往被看作是旁门左道。因此在正式出版的计算机书籍中提及并不多，可谓讳莫如深；而破解者、黑客、病毒制造者则对此乐此不疲。反工程有专门的软件工具、知识体系，许多“秘笈”在网络上流传。系统的反工程技术涉及从还原下层逻辑一直到大型软件的顶级架构。

这些都不是本书的内容，也不是作者有能力讲述的范围。本书仅仅让读者学会阅读调试器中的汇编代码，以加深对 Windows 底层的理解，增加读者解决下层 bug 的能力。对反工程没有涉及。

请注意，对软件的反工程行为，如果未取得合法的授权，则是侵犯知识产权的非法行为。

有兴趣了解反工程的读者，可以阅读《Reversing:逆向工程解密》。此书信息如下：

英文名：Reversing:Secrets of Reverse Engineering

中文名：Reversing：逆向工程解密

作者： Eldad Eilam;Elliot Chikofsky

译者： 韩琪、杨艳等

出版社：电子工业出版社

不过，回归主题，建议读者还是先继续阅读本书，继续做好 Windows 底层程序员这份有前途的职业吧。

1.1 上机建立第一个工程

一条指令可能有很多细节，比如第二个操作数和第一个操作数中，哪个只能是寄存器，哪个不能用寄存器加立即数等。如果要用手写汇编语言来开发一个软件的话，这些知识很重要。但是实际上，现在需要的是用 C 语言或者更高级的语言进行开发，只是需要理解一些没有 C 代码的部分。读者可以认为，那些代码的汇编指令用法一定是正确的，因为它们是编译器生成的。

1.2 节（简要复习常用的汇编指令）将简要地回顾那些指令，并保证这些回顾非常容易理解。

1.1.1 用 Visual Studio 创建工程

建议读者的计算机中操作系统的版本最好是 Windows XP 或者更高的版本，使用 Vista 问题也不大，虽然可能操作细节和界面与本书描述的情况稍有不同。然后，请安装某个版本的 Visual Studio，推荐 Visual Studio 2003 或者更高的版本。下面的描述同时顾及 Visual Studio 2003 和 Visual Studio 2005 的情况，语言版本为英文。

下面的步骤将在 Visual Studio 上建立一个用于实验的工程。有经验的读者可以忽略下面的描述。

1 打开 Microsoft Visual Studio 2005，选择主菜单“File”。

2 选择子菜单“New”下面的“Project”，打开“New Project”对话框。

3 左边选择 Visual C++，Win32；右边选择 Win32 Console Application；下面输入一个工程名，然后单击“OK”按钮，出现向导。一切都选择默认设置，最后单击“Finish”按钮。

此时新建立工程的主文件是一个扩展名为.cpp 的文件，请将它改为扩展名为.c 的文件。

Windows 底层代码基本上都是用 C 语言编写的，研究 C 语言和汇编指令的关系是本节的任务。为此，这个文件必须改变为扩展名为.c 的文件，这样 VC 才会自动以 C 语言的方式进行编译。本书后面专门有一章的内容来研究 C++下的反汇编阅读（第 9 章 用 C++编写的内核程序）。

4 如果读者使用的是 Visual Studio 2005，直接在左边的 Solution Explorer 中用鼠标右键单击文件 YourProjectName.cpp，选择“Rename”，然后把.cpp 改为.c 即可。

如果是更老的 Visual Studio 版本，请单击右键，Remove 这个文件。然后在外面改名，再对工程单击右键，选择“Add”，再选择“Exist Item”，追加进来。

此时程序是这样的：

```
#include "stdafx.h"
int _tmain(int argc, _TCHAR* argv[])
{
        return 0;
}
```

1.1.2　用 Visual Studio 查看汇编代码

C 语言程序对应的汇编代码，可以在 VC 中非常清楚地显示出对应关系。但是并不是所有的读者都知道如何调出汇编指令窗口。这个诀窍在下面描述。

1 VC 必须处于调试状态才能看到汇编指令窗口。因此，请在 return 0 一句上设置一个断点：把光标移到那一行，然后按下 F9 键设置一个断点。

2 按下 F5 键调试程序。当程序停止在这一行的时候，打开菜单“Debug”下的“Windows”子菜单，选择“Disassembly”。这样，出现一个窗口，显示下面的信息：

```
--- f:\root\work\any\t12\t12\t12.c ----------
// t12.cpp : Defines the entry point for the console application.
//
#include "stdafx.h"

int _tmain(int argc, _TCHAR* argv[])
{
00411360  push    ebp
00411361  mov     ebp,esp
00411363  sub     esp,0C0h
```

```
00411369  push    ebx
0041136A  push    esi
0041136B  push    edi
0041136C  lea     edi,[ebp-0C0h]
00411372  mov     ecx,30h
00411377  mov     eax,0CCCCCCCCh
0041137C  rep stos   dword ptr es:[edi]
    return 0;
0041137E  xor     eax,eax
}
00411380  pop     edi
00411381  pop     esi
00411382  pop     ebx
00411383  mov     esp,ebp
00411385  pop     ebp
00411386  ret
```

如果上面的内容完全看不懂，也许读者需要复习一下汇编指令。以上的汇编指令数量非常的少，只需要了解 push、mov、sub、lea、stos、xor、pop、ret，就可以继续本书的学习之旅了。所以请不用担心，接下来就会熟悉这些代码。

1.2 简要复习常用的汇编指令

1.2.1 堆栈相关指令

对指令的详细了解，应该查阅 Intel 发布的指令手册。但是从头开始阅读那些手册是一件令人望而生畏的事情。读者暂时可以只初步了解，忽略一些细节，当实际用到的时候，再具体查阅相关指令。

- push：把一个 32 位的操作数压入堆栈中。这个操作导致 esp 被减 4。esp 被形象地称为栈顶。我们认为顶部是地址小的区域，那么，压入堆栈的数据越多，这个堆栈也就越堆越高，esp 也就越来越小。在 32 位平台上，esp 每次减少 4（字节）。
- pop：相反，esp 被加 4，一个数据出栈。pop 的参数一般是一个寄存器，栈顶的数据被弹出到这个寄存器中。

一般不会把 sub、add 这样的算术指令，以及 call、ret 这样的跳转指令归入堆栈相关指令中。但是实际上在函数参数传递过程中，sub 和 add 最常用来操作堆栈；call 和