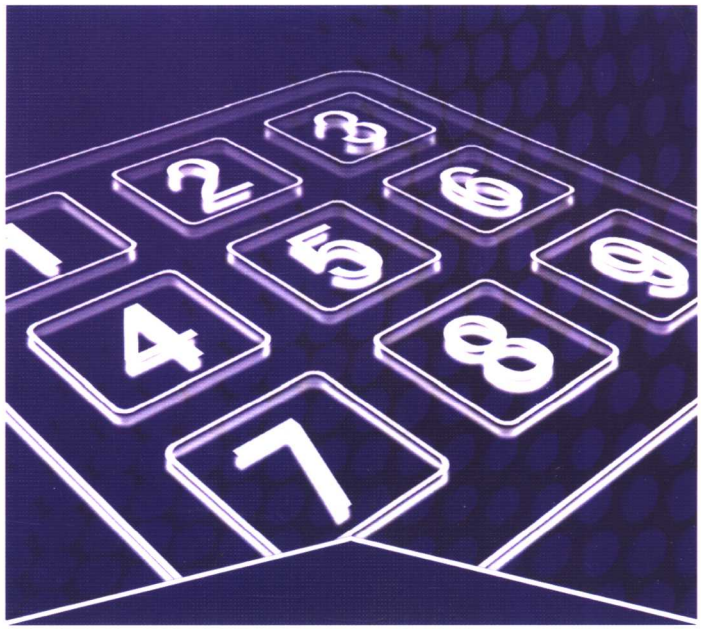
电脑编程实例导航丛书



计算机系统安全与维护编程

典型实例解析

《电脑编程技巧与维护》杂志社 编著

中国水利水电出版社
www.waterpub.com.cn

电脑编程实例导航丛书

计算机系统安全与维护编程典型实例解析

《电脑编程技巧与维护》杂志社　编著

中国水利水电出版社

内 容 提 要

保证计算机系统的安全，保证电脑软硬件的正常运转，已是广大电脑用户迫切需要了解和掌握的技术；维护好电脑正常高效地运转已成为用户的基本要求。多年来，《电脑编程技巧与维护》杂志社收到了许许多多优秀的作者提供的高质量的有关计算机维护、计算机安全应用编程的稿件。这些稿件用编程的方法来解决对电脑系统的安全与维护，其主题鲜明，特点突出。一个个实用技巧，针对性强；一个个活生生的编程应用实例让人感到亲切、实用。但由于《电脑编程技巧与维护》刊物本身页码的限制，在“计算机安全与维护”栏目只能录用极少一部分稿件，而绝大多数这方面内容的好稿件无版面刊登录用。为了充分利用这些宝贵的资源，及时传播电脑安全与维护最新编程的好经验，好技巧，杂志社将近年来有关电脑维护与安全方面的稿件经过精心挑选、调试汇集编写了本书。本书集锦了 70 个编程实例，分三章。第 1 章硬件系统维护编程实例篇，23 个实例；第 2 章 Windows 系统维护编程实例篇，28 个实例；第 3 章网络安全与维护编程实例篇，19 个实例。本书各章所选内容紧扣计算机系统安全与维护编程主题，特点鲜明；汇集了电脑编程高手们的智慧结晶，宝贵经验；选自实际应用中的案例集锦，传送真经。

本书所有实例源代码可以从中国水利水电出版社网站下载，网址：http://www.waterpub.com.cn/softdown/。

图书在版编目（CIP）数据

计算机系统安全与维护编程典型实例解析/《电脑编程技巧与维护》杂志社编著．—北京：中国水利水电出版社，2006

（电脑编程实例导航丛书）

ISBN 7-5084-3856-6

Ⅰ．计… Ⅱ．电… Ⅲ．①电子计算机—安全技术 ②电子计算机—维护 Ⅳ．①TP309 ②TP307

中国版本图书馆 CIP 数据核字（2006）第 083087 号

书　　名	**计算机系统安全与维护编程典型实例解析**
作　　者	《电脑编程技巧与维护》杂志社　编著
出版　发行	中国水利水电出版社（北京市三里河路 6 号　100044）
	网址：www.waterpub.com.cn
	E－mail：mchannel@263.net（万水）
	sales@waterpub.com.cn
	电话：（010）63202266（总机）、68331835（营销中心）、82562819（万水）
经　　售	全国各地新华书店和相关出版物销售网点
排　　版	北京万水电子信息有限公司
印　　刷	北京蓝空印刷厂
规　　格	787mm×1092mm　16 开本　23.25 印张　700 千字
版　　次	2006 年 9 月第 1 版　2006 年 9 月第 1 次印刷
印　　数	0001—5000 册
定　　价	45.00 元

丛 书 序

《电脑编程技巧与维护》杂志是为从事电脑编程、系统应用开发的人员创办的专业性和实用性都很强的技术刊物。自1994年创刊以来，始终以“实用第一，智慧密集”为宗旨，坚持“质量第一”、“读者第一”的原则，为广大电脑编程爱好者、软件开发人员和专业计算机系统维护人员提供第一手技术资料、编程技巧和维护经验；紧紧跟踪计算机软硬件技术发展和应用趋势，不断求变创新，针对软件开发过程中的许多关键技术问题着重提供各类解决方案，在业内获得一致好评，是广大编程和维护人员的首选刊物。在栏目内容上，选题覆盖面广，涉及技术领域宽、信息量大，帮助程序员开阔视野；在技术水平上，始终把握计算机技术发展的大方向，提供先进、详尽、准确的技术指导，并在长期工作中与国际性大公司建立了良好的合作关系，为读者提供全球最新、最全的实用信息；在实用性上，稿源来自于专业开发和维护人员的实践经验，是普通书籍难以获得的编程心得、体会与技巧。

今年是《电脑编程技巧与维护》创刊十二周年，为了最大限度地开发和利用本刊宝贵而丰富的资源，更好地服务和真诚回报多年来一直关爱和支持本刊的广大读者，《电脑编程技巧与维护》杂志社和中国水利水电出版社共同策划出版了这套《电脑编程实例导航丛书》。

这套丛书包括《Visual C/C++系统开发典型实例解析》、《Visual C/C++图形图像与游戏编程典型实例解析》、《Visual Basic编程典型实例解析》、《Delphi编程典型实例解析》、《C#编程典型实例解析》、《Java编程典型实例解析》、《计算机系统安全与维护编程典型实例解析》、《计算机网络与通信编程典型实例解析》、《编程疑难问题解析126例》，一套9册共684个典型实例。每册书的编程实例均依不同的编程应用分成若干章，条目清晰可查，使用极为方便。

这套丛书选编了《电脑编程技巧与维护》杂志近两年发表的和一部分尚未发表而又极为实用、精彩的典型编程实例。该套书的特点是：其各册内容来自编程高手的智慧和经验总结，其中不少文章的作者是业界资深程序员和技术专家，内容有深度、思路有新意、讲解深入浅出，编程技巧新颖实用，构思巧妙；丛书中的实例都是作者从实际项目提炼出的开发范例，实例讲解部分先给出设计目标，然后介绍实现目标的基本思想和方法，最后详细给出其核心程序的源代码，对程序的关键部分进行讲解并给出程序的运行效果；丛书中每一个实例的程序源代码都经过上机调试通过，对编程中的疑难问题进行了深入解答，给程序开发人员

移植源代码和学用编程带来了方便，加快了编程应用的步伐。全套书既讲究内容的深入性、专业性、权威性和实用性，同时兼顾轻松、通俗易懂、时效性强的特点。

这套丛书是《电脑编程技巧与维护》资源的二次深入开发，浓缩了当前主流编程语言Visual C/C++、Visual Basic、Delphi、Java 、C#等程序设计的精华，其目的是力求为读者建造一个真正的知识整合、编程思想、编程技术、技巧交流的平台，让读者从中学习到编程高手的诀窍，丰富读者的编程技巧，拓宽读者的编程思路，迅速提升读者的程序开发能力。对电脑编程人员来说，程序开发能力的提高，除了对语言和算法的不断钻研学习、不断实践、不断总结提高，练好基本功，打好基础外，还要集思广益，善于学习，善于借鉴参考别人的经验，深入透彻地理解其中的精髓，然后溶入到自己的设计方案中去，这无疑是一条有效的学习途径，对于自身编程能力的增强和编程水平的迅速提高十分重要，这也正是我们编写这套丛书想要达到的目的。

该套书可作为高等院校学生进行课程项目开发、毕业项目设计的参考书，也可作为软件从业人员及编程爱好者的珍藏宝典，还可作为高等培训学校的实例教程。

《电脑编程技巧与维护》杂志社

中 国 水 利 水 电 出 版 社

2006 年 5 月

前　言

在《电脑编程技巧与维护》1994年7月创刊号发刊词中就写着“买一台电脑并不难，难的是如何用好这台电脑”。十多年过去了，计算机软硬件技术有了突飞猛进的发展，应用领域不断扩大，应用范围不断向纵深发展，电脑无处不在已经不是一句空话。电脑应用已普及到各个层面，尤其是互联网的出现和应用，上网已成为人们工作、生活中不可缺少的一部分重要内容，真不可想像离开网络世界将变成什么样。但是从另一个侧面来看，保证计算机系统的安全，保证电脑软硬件系统的正常运转，已是广大电脑用户迫切需要了解和掌握的技术；维护好电脑正常高效地运转已成为用户的基本要求。

多年来，杂志社收到了许许多多优秀作者提供的高质量的计算机维护、计算机安全方面的稿件。这些稿件以编程的方法来讲解对电脑系统的安全与维护，主题鲜明，特点突出。一个个实用技巧，针对性强；一个个活生生的编程应用实例让人感到亲切、实用。但由于刊物本身页码的限制，在本刊的“计算机安全与维护”栏目只能录用极少一部分稿件，而绝大多数这方面内容的好稿件无版面刊登录用。为了充分利用这些宝贵的资源，及时传播电脑安全与维护最新编程的好经验、好技巧，杂志社将近年来有关计算机维护与安全方面的稿件经过精心挑选、调试编写了这本《计算机系统安全与维护编程技巧典型案例解析》。

本书集锦了70个典型应用编程实例，分为3章。第1章硬件系统维护编程实例篇，23个实例；第2章Windows系统维护编程实例篇，28个实例；第3章网络安全与维护编程实例篇，19个实例。全书各章所选内容紧扣计算机系统安全与维护编程主题，特点鲜明；汇集了电脑编程高手们的智慧结晶，经验宝贵；选自实际应用中的案例集锦，传送真经。

本书的主要特色如下：

第一，每一章都通过一个个的实例来介绍计算机系统安全与维护应用编程的方法和技巧，避免了枯燥、空洞的理论，并且每一个实例都具有很强的实用性和代表性。在实例的讲解上一般都是先给出设计目标，然后介绍实现该目标的基本思想和方法，最后详细给出其核心程序的源代码，并对程序的关键部分进行讲解给出程序的运行效果。

第二，所选的每一个实例都是从事计算机系统安全与维护应用编程人员的经验总结，具有很强的实用性，其中很多编程技巧可供借鉴。

第三，每一个实例的程序源代码都经过上机调试通过，给程序开发人员移植源代码带来了方便，加快了编程应用的步伐。

第四，对个别版本和开发环境稍微低一些的经典实例进行点评和分析，起到触类旁通的作用。

本书内容深入、概念清晰、层次分明，实例典型而实用，但不足及疏漏之处在所难免，恳请广大读者批评指正。

《电脑编程技巧与维护》杂志社

2006 年 1 月

目　录

第 2 章　Windows 系统维护编程实例篇

第 3 章　网络安全与维护编程实例篇

第 1 章　硬件系统维护编程实例篇

例题 1 用 VC + +编程实现对 CPU 主频的检测

一、引言

目前，家用电脑频繁升级换代，尤其是 CPU 主频越来越高。许多用户常常把 CPU 的主频作为选购电脑的一个重要指标。但是检测 CPU 的主频，通常要在启动计算机时进入 BIOS 设置程序中查看，这种方法比较麻烦，而且该方法不适于初学者使用。于是用 VC + +编写了一个小程序，用这个小程序在 Windows 桌面环境下即可检测 CPU 的主频。

二、RDTSC 指令

Intel 80386 微处理器支持许多功能强大的指令，其中有一条名为 RDTSC（Read Time Stamp Count）的指令，顾名思义，这条指令用于读取 CPU 的时间戳计数器。RDTSC 指令读出的数据共 64 位，高 32 位保存在 EDX 寄存器中，低 32 位保存在 EAX 寄存器中。处理器在每个时钟周期内增加时间戳计数器的数值，并且当处理器复位时把计数器清零。Intel 80386 微处理器设置了 4 个特权级：0 级、1 级、2 级和 3 级，操作系统运行于特权级第 0 级上，而用户程序运行在第 3 级上，操作系统对系统硬件采取了保护的策略，限制运行于第 3 级的应用程序对系统资源（如中断控制器、内存等）的操作，因此不论在保护模式下还是在虚拟 8086 模式下，CR4 寄存器中都有一个名为 TSD（Time Stamp Disable）的标志位，用来标记时间戳（Time Stamp）是否使能，以此来限制用户对 RDTSC 指令的使用。当 TSD 标志被清零时，RDTSC 指令可以在任何特权级下执行；当 TSD 标志被置位时，RDTSC 指令就只能在 0 特权级下执行。幸运地是，TSD 标志默认是被清零的，所以我们可以在 Windows 操作系统的用户级（即第 3 级）下使用 RDTSC 指令。

三、程序实现

1. 算法

Windows 操作系统为程序员提供了丰富的 API 函数，利用 API 函数可以实现对系统函数的调用，功能强大，而且灵活方便。我们可以用 GetTickCount()函数取得一个时间值 T1，用 rdtsc 读出 CPU 计数器的数值 R1，然后写一个定时器函数，每隔 1 秒触发一次，该函数触发时再调用 GetTickCount()取得一个时间值 T2，用 rdtsc 读出当前 CPU 的计数器的数值 R2，用公式(R2 - R1)/(T2 - T1)进行计算，这样就得到在时间间隔（T2 - T1）内 CPU 计数器的数值变化情况。由于 T2 和 T1 的数值是毫秒级的(ms)，所以要乘以 1000，即得到每秒变化多少次，这样就可以得到 CPU 主频的近似值。

2. 选择开发工具

调用 RDTSC 指令需要汇编语言，而用 C 语言编写的程序可以方便地嵌入汇编语言，并且为了向用户提供一个友好的界面，应该用可视化的编程工具，基于以上考虑，本例选用了 VC + +6.0 编程工具。

3. 程序设计

建立一个基于对话框的工程，命名为 Testcpu，在 CTestcpuDlg 类中添加 WM_TIMER 消息并增加响应这一消息的成员函数，在对话框中添加一个 EDIT 控件（IDC_EIDT1）和一个 BUTTON 控件（IDCANCEL），为 EDIT 控件添加一个 CString 类型的成员变量 m_fre，用过 VC 的朋友对这些操作都操作都很熟悉。程序中内嵌大量汇编语言，这是因为汇编语言运行速度快、可靠性高。主要代码如下：

定义全局变量：

```
const WM_TIME = WM_USER + 100; // 定时器标识符
DWORD rs1 = 0, rs2 = 0;
DWORD t1;
```

在初始化函数 BOOL CTestcpuDlg: : OnInitDialog()中添加以下代码：

```
{...
t1 = GetTickCount(); // 读时间值 t1
_asm// 内嵌汇编语言，读出计数器的值
{
rdtsc
mov rs2, edx // 高 32 位保存在 rs2 中
mov rs1, eax// 低 32 位保存在 rs1 中
}
SetTimer(WM_TIME, 1000, 0); // 设置定时器，间隔 1 秒
...
}
```

编写定时器函数 void CTestcpuDlg: : OnTimer(UINT nIDEvent)，代码如下：

```
void CTestcpuDlg: : OnTimer(UINT nIDEvent)
{
    // TODO: Add your message handler code here and/or call default
    DWORD rc1;
    DWORD t2 = GetTickCount(); // 读时间值 t2
    _asm// 嵌入汇编语言
    {
        rdtsc
        push eax
        push edx
        sub eax, rs1
        sub edx, rs2
        pop rs2
        pop rs1
        mov ecx, 1000000
        div ecx
        cmp edx, 500000h
        jle rr
        inc eax
        rr: mov rc1, eax
    }
    t1 = t2 - t1;
    float fre = (float)rc1/t1 * 1000; // fre 即频率
    m_freq. Format("您的 CPU 主频是% 5. 0fMHz", fre); // 显示结果
    UpdateData(FALSE);
    t1 = GetTickCount(); // 读时间值 t1
    CDialog: : OnTimer(nIDEvent);
}
```

最后，添加响应单击 BUTTON 控件的函数：

```
void CTestcpuDlg: : OnCancel()
{
    // TODO: Add extra cleanup here
    KillTimer(WM_TIME); // 取消定时器
```

```
    CDialog: : OnCancel( ); //退出程序
}
```

现在用 VC + +6.0 编译运行。这下你不用进入 BIOS 也可以知道电脑的 CPU 主频了，是不是很方便。本程序在 Windows 98, Windows 2000 下调试通过。

四、总结

Intel 80386 微处理器还支持其他功能强大的指令，用这些指令我们可以实现许多对系统硬件的操作，期望广大编程爱好者共同研究，开发出更多的功能。

（张　研）

例题 2　在 Windows 2000 下取得系统 CPU 占有率

一、应用背景

在许多实际的应用中，需要知道系统当前 CPU 的占有率，以便决定是否该进行某些操作。例如在一个大型系统中，经常要完成数据备份操作，但同时要保证业务不受影响，因此一般是在系统当前的 CPU 占有率较低的情况下进行。

二、实现原理

如何在 Windows NT/2000 下取得系统 CPU 的占有率呢？在微软提供给开发人员的公开 API（Win32 API）中，没有提供取系统信息的 API，所以必须通过本机系统服务（Native API）来完成。Native API 是 Windows 用户模式中为上层 Win32 API 提供接口的本机系统服务。在这里用到一个本机系统服务函数 NtQuerySystemInformation，这个函数为我们提供了丰富的系统信息，同时还包括对某些信息的控制和设置。下面是这个函数的原型：

```
typedef NTSTATUS (_stdcall * NTQUERYSYSTEMINFORMATION)
(IN SYSTEM_INFORMATION_CLASS SystemInformationClass,
IN OUT PVOID SystemInformation,
IN ULONG SystemInformationLength,
OUT PULONG ReturnLength OPTIONAL);
NTQUERYSYSTEMINFORMATION NtQuerySystemInformation;
```

从中可以看到，SystemInformationClass 是一个类型信息，它大概提供了 50 余种信息，也就是我们可以通过这个函数对大约 50 多种的系统信息进行探测或设置。SystemInformation 是一个 LPVOID 型的指针，它为我们提供需要获得的信息或我们需要设置的系统信息。SystemInformationLength 是 SystemInformation 的长度，它根据探测的信息类型来决定。至于 ReturnLength 则是系统返回的数据长度，通常可以设置为空指针（NULL）。

通过该函数获得系统 CPU 占有率的过程如下：

（1）取出系统中 CPU 个数。

（2）取出当前系统时间（Ticks 个数），减去原来保存的系统时间，得出从上次取值到目前的系统时间差。

（3）取出系统当前空闲时间，减去原来保存的空闲时间，得出这个时间段的空闲时间。

（4）用这个时间段的空闲时间除以系统时间，得出这段时间系统的 CPU 空闲率。

（5）用 100 -（系统的空闲率）/（CPU 个数）所得的值就是 CPU 占有率。

三、实现程序

以上功能的实现程序如下：

```
#include <windows.h>
#include <conio.h>
#include <stdio.h>
#define SystemBasicInformation 0
#define SystemPerformanceInformation 2
#define SystemTimeInformation 3
#define Li2Double(x) ((double)((x).HighPart) * 4.294967296E9 + (double)((x).LowPart))
typedef struct
{
  DWORD dwUnknown1;
  ULONG uKeMaximumIncrement;
  ULONG uPageSize;
  ULONG uMmNumberOfPhysicalPages;
  ULONG uMmLowestPhysicalPage;
  ULONG uMmHighestPhysicalPage;
  ULONG uAllocationGranularity;
  PVOID pLowestUserAddress;
  PVOID pMmHighestUserAddress;
  ULONG uKeActiveProcessors;
  BYTE bKeNumberProcessors;
  BYTE bUnknown2;
  WORD wUnknown3;
}SYSTEM_BASIC_INFORMATION;
typedef struct
{
  LARGE_INTEGER liIdleTime;
  DWORD dwSpare[76];
}SYSTEM_PERFORMANCE_INFORMATION;
typedef struct
{
  LARGE_INTEGER liKeBootTime;
  LARGE_INTEGER liKeSystemTime;
  LARGE_INTEGER liExpTimeZoneBias;
  ULONG uCurrentTimeZoneId;
  DWORD dwReserved;
}SYSTEM_TIME_INFORMATION;
typedef LONG (WINAPI * PROCNTQSI)(UINT, PVOID, ULONG, PULONG);
PROCNTQSI NtQuerySystemInformation;
BOOL IsNtOS()
{
    DWORD dwWinVer;
    dwWinVer = GetVersion();
    if(dwWinVer < 0x80000000)
    {
     return TRUE;
    }
    return FALSE;
```

```
}
DWORD GetCpuUsage()
{
 SYSTEM_PERFORMANCE_INFORMATION SysPerfInfo;
 SYSTEM_TIME_INFORMATION SysTimeInfo;
 SYSTEM_BASIC_INFORMATION SysBaseInfo;
 static double  dbIdleTime;
 static double  dbSystemTime;
 long  status;
 static LARGE_INTEGER   liOldIdleTime = {0, 0};
 static LARGE_INTEGER   liOldSystemTime = {0, 0};
 //得到系统中的处理器数
 status = NtQuerySystemInformation(SystemBasicInformation, & SysBaseInfo, sizeof(SysBaseInfo), NULL);
 if (status ! = NO_ERROR)
   return 0;
 //得到新的系统时间
 status = NtQuerySystemInformation(SystemTimeInformation, & SysTimeInfo, sizeof(SysTimeInfo), 0);
 if (status! = NO_ERROR)
    return 0;
 //得到新的 CPU 空闲时间
 status = NtQuerySystemInformation(SystemPerformanceInformation, & SysPerfInfo, sizeof(SysPerfInfo),
 NULL);
 if (status ! = NO_ERROR)
    return 0;
 //如果第一次调用跳过
 if (liOldIdleTime. QuadPart ! = 0)
  {
 //当前值 = 最新值 - 原来的值
 dbIdleTime = Li2Double(SysPerfInfo. liIdleTime) - Li2Double(liOldIdleTime);
 dbSystemTime = Li2Double(SysTimeInfo. liKeSystemTime) - Li2Double(liOldSystemTime);
 //当前系统空闲率 = 空闲时间 / 系统时间
 dbIdleTime = dbIdleTime / dbSystemTime;
 //当前 CPU 占有率% = 100 - (当前系统空闲率 * 100) / CPU 个数
 dbIdleTime = 100. 0 - dbIdleTime * 100. 0 / (double)SysBaseInfo. bKeNumberProcessors + 0. 5;
  }
   //保存新的空闲时间和系统时间
   liOldIdleTime = SysPerfInfo. liIdleTime;
   liOldSystemTime = SysTimeInfo. liKeSystemTime;
   return (DWORD)dbIdleTime;
}
void main(void)
{
 DWORD dwCpuUsage;
 if(! IsNtOS())
 {
    printf("This program can not run on Win9x! \n");
    return;
 }
 NtQuerySystemInformation = (PROCNTQSI)GetProcAddress(
     GetModuleHandle("ntdll"),
```

```
        "NtQuerySystemInformation"
          );
  if (! NtQuerySystemInformation)
  {
     printf("Can not find needed functions in ntdll. dll! \n");
     return;
  }
 printf("\nCurrent CPU Usage(press any key to exit):  ");
  while(! _kbhit())
  {
    dwCpuUsage = GetCpuUsage();
    printf("\b\b\b\b% 3d% %", (UINT)dwCpuUsage);
    Sleep(1000);
  }
  printf("\n");
}
```

该程序在 Windows 2000，VC + +6.0 环境下编译通过。

（文小青）

例题 3　Ring0 的编程实现

处理器有 4 个级的优先权：Ring0，Ring1，Ring2，Ring3。

大多数程序运行于 Ring3 这个级别。Ring3 有很多限制，如：我们不能读 DR（Debug - Registers，调试寄存器），所有关于 DR 的指令都不起作用（如：mov eax, dr7）。对来说当然是优先级越高越好，所以要使我们的程序运行在 Ring0 级别。但是只有少数特殊的程序有 Ring0 的优先权。设备驱动程序运行于 Ring0，但它的代码可不简单！

所幸，Windows 有很多漏洞，特别是 Win9X。有很多方法可以将程序的优先级从 Ring3 变成 Ring0。这些方法被很多病毒所用，当然，我们还有更多、更好的用途！

这些方法只适用于 Win9X，不适用于 WinNT。因为 WinNT 是黑客的主要攻击对象，它的漏洞都填的差不多了。

可以在程序中加入代码，先判断操作系统是否是 Win9X，若是就可以用了。

如果一定要在 WinNT 或 Win2K 下用 Ring0，就要编写设备驱动了。

如果你是 Admin（管理员）级别的用户，那还有别的方法，但前提是你必须有 Admin 的级别。

对一般用户来说这是个好消息，因为 Windows 下的病毒没那么好编，那些病毒作者就要费劲了！

- -

Part I

下面介绍一下改变优先权至 Ring0 的方法使用 LDT（Locale Descriptor Table）。

这个古老的方法极少使用。这不是最好的方法，但至少比用 IDT 好，因为它鲜为人知。下面是程序范例：

- -

```
..386p
..MODEL FLAT, STDCALL
locals
jumps
```