第 7 卷　第 3 期　　光学　精密工程　　Vol. 7, No. 3
1999 年 6 月　　OPTICS AND PRECISION ENGINEERING　　June, 1999

Kernel 模式与虚拟设备

王　沛　　袁晓兵　　王国辉
(中国科学院长春光学精密机械研究所　　长春 130022)

摘　要　由于 Windows 操作系统不允许直接访问硬件, 给图像的实时采集、存储、显示等处理工作带来了很大困难。本文提出在 Kernel 模式下采用编制虚拟设备驱动程序的方法, 对于图像处理工作所要求的实时性及同步性在软件方面提出了一种解决方案。

关键词　Windows 系统　Kernel 模式　虚拟设备

1　引　　言

勿庸置疑, Microsoft 的 Windows(尤其是 Windows95、Windows98 和 Windows NT) 是 PC 机上真正的主流操作系统, 从现在直到将来, Windows 都将是我们的主要工作和开发平台。Windows 编程的特点就在于对编程的支持非常丰富, 不像 DOS 编程, 从计算到界面, 每次都需要从头设计。但是, 正如在 DOS 下仅满足于 INT 21 调用是难以编写出精湛的程序来一样, 在 Windows 下也需要对其内部原理有深刻的认识。在实际应用中, 为了提高应用软件的实时性与同步性, 须对 Windows 内核进行深入剖析。我们就 Windows 的核心之一, 即 Kernel 模式进行一些研究。

2　理论分析

Windows 操作系统是建立在 32 位保护模式基础上的, 保护机制支持四级递增级别, 分别为 Ring3、2、1、0, Ring 0 级是最高特权级。对于可靠性来说, 操作系统中最可靠和最抗毁坏的代码应该运行在最高特权级 Ring 0 级上, 那些可能运行失败或危及系统整体性的应用程序应该运行在最低特权级 Ring 3 级上。由于可以运行在高特权级上的程序数目在接近 0 级处减少

收稿日期: 1998- 12- 18
修稿日期: 1999- 02- 28

少并且 0 级代码很可能仅存在于操作系统内核中, 所以特权级的分类说明是一个同心环, 如图 1 所示。Ring 0 级我们就叫 Kernel。

每个系统对象配属一个特权级并且驻留在特定环内。运行在内环中的过程可以访问外环(拥有低特权级) 中数据对象, 但是外环过程不能访问高特权级的对象。此外, 为了防止不良代码损坏操作系统, 过程不能调用低可靠性(处于外环的过程) 的其他过程。例如: 运行在 Ring1 中某个过程可以访问驻留在 Ring2 或 Ring3 中的一个数据段, 但不能访问特权级是 0 的数据段。可是, 一个 Ring1 代码既不能执行驻留在 Ring2 或 Ring3 中的子程序, 也不能调用 Ring0 中的子程序。图 2 说明了这个概念。

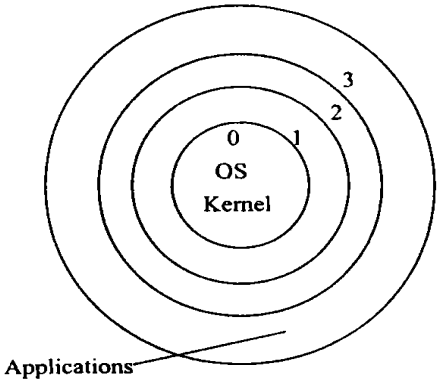

Fig. 1　The priority rings

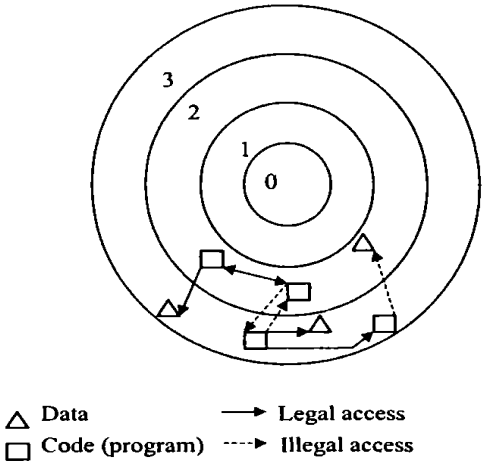

Fig. 2　The access of rings

一个操作系统不必支持全部四个特权级。例如, Windows 及 UNIX 系统通常仅支持两级, Ring0 和 Ring3。而 OS/2 支持三级: 系统级代码运行在 Ring0 中, 应用程序运行在 Ring3 中, 需要访问 I/O 装置的特殊例程运行在 Ring2 中。

作为一个保密措施, 特权级最高的同心环工作最佳, 但存在着运行于 Ring3 中的应用程序需要操作系统服务的可能性, 由于访问限制, 它无法调用操作系统, 这就涉及到级间通信的问题。

我们所说的 Kernel 概念, 就是指一种核心层的结构, 管理系统的大部分基本功能。例如在不同的执行模块之间共享处理器, 处理硬件例程和其它的硬件操作。

Kernel 模式就是指处理器允许完全的、没有保护的对系统进行访问。一个驱动程序或一个线程运行在 Kernel 模式可以对系统内存、硬件进行存取。它本身是一种最接近于计算机操作系统级的软件处理, 可以直接驱动硬件或接口, 工作在 Ring 0 级上。而普通的用户模式 (User Mode) 工作在 Ring 3 级上。

Kernel 模式驱动程序是对逻辑设备、虚拟设备或物理设备的驱动, 并支持 Ring 0 操作。

因为通常的应用程序都工作在用户模式下。对 Windows 和 Windows NT 来说, 用户模式就是一种非特权的处理器模式, 非特权的应用代码执行模式, 在 Windows NT 中还包括被保护的子系统代码。通常所使用的 Win32 多媒体驱动程序和 VDDs 都工作在用户模式下。

一般应用程序的执行过程如图 3 所示, 而 Kernel 程序的执行过程如图 4 所示:

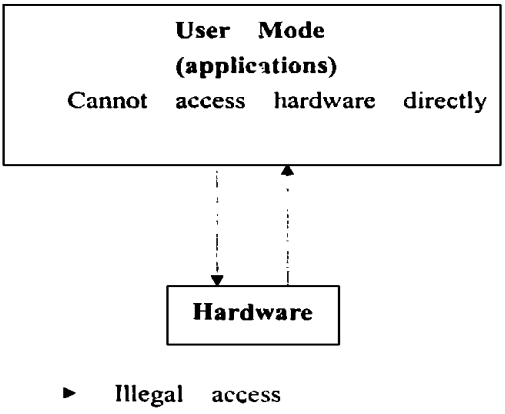

Fig. 3 The process of application program

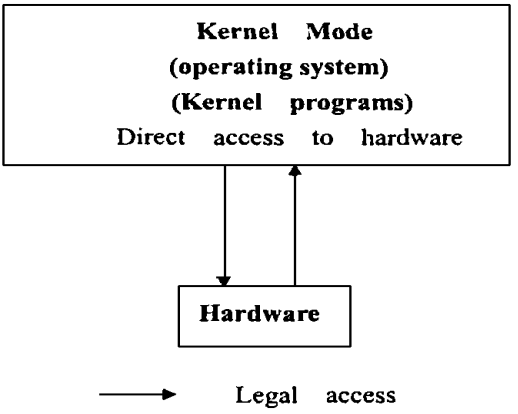

Fig. 4 The process of Kernel program

对 Windows 内核的剖析是一个很复杂的过程, 这里只是简单地分析 Windows95 中的三个核心部件: Ring 0 级上的 VWIN32. VXD, Ring 3 级上的 KERNEL32. DLL 和 KRNL386. EXE。KRNL386. EXE 是 16 位核心部件, KERNEL32. DLL 是它的 32 位等价物, 这些动态链接库负责提供任何 Win16 和 Win32 应用所需的核心函数集。VWIN32. VXD 是高层两个最重要的虚拟设备驱动程序(另一个是 Virtual Machine Manager, 也就是 VMM) 之一, 它提供了操作系统最基本层次上的关键功能 —由 32 位的 KERNEL32. DLL 和 16 位 KRNL386. EXE 都使用的功能, 它包含了影响进程和线程的调度与同步的 Win32 VxD 服务。

Windows 通过三个核心部件的通信和相互作用来实现环间服务和系统服务。因为 KRNL386. EXE 是 Ring 3 级的 16 位调度程序, 相比较来说, 我们最感兴趣的是 VWIN32. VXD 与 KERNEL32. DLL 之间的通信。

VWIN32. VXD 对于 KERNEL32. DLL 的通信过程主要是: 在 VWIN32. VXD 使用的结构中, 存在指向 THREAD-DATABASE 和 PROCESS-DATABASE 的指针。VWIN32. VXD 还有指向由 KERNEL32. DLL 所维护的指向当前进程和线程的全局指针变量。除进程和线程外, 在 KERNEL32. DLL 的启动阶段, VWIN32. VXD 还能得到 KERNEL32. DLL 中各例程的指针列表。

KERNEL32. DLL 对于 VWIN32. VXD 的通信过程主要是: KERNEL32. DLL 调用了 VWIN32. VXD 提供的 VWIN32 VxD 服务这个事实。除了 VWIN32 VxD 服务外, KERNEL32. DLL 和 VWIN32. VXD 之间的其它合作出现在当一个特定的 VWIN32 VxD 服务被调用时, VWIN32. VXD 提交一个 VWIN32. VXD 中的函数地址的列表。

Windows 提出了"虚拟"的概念, 如虚拟机器、虚拟设备、虚拟内存等。在 Windows 中设备已经不是一个具体的概念, 任何一个执行特殊功能的过程都可称作一个设备。对 WDM (Win32 Driver Model) 来说, 经常是指一个目标设备, 也可指硬件的某一单元。可以通过创建某一设备并且为它编制驱动程序来完成所需要的应用, 这就是虚拟设备的概念。

在 Windows95 及 Windows98 中, VxD 是虚拟设备驱动程序如虚拟的显示设备驱动程序 (VDD)、虚拟鼠标驱动程序(VMD)、虚拟键盘设备驱动程序(VKD) 等的统称。VxD 实质上就是工作在 Kernel 模式上的动态链接库, 对硬件设备进行虚拟化, 软件可通过调用 VxD 的服务

来使用相应的设备, 故 VxD 在很大程度上起到了原来 BIOS 的作用。所有的 VxD 和 VMM 一起, 构成了 Windows 操作系统的基础。

3　实际应用

由于 Windows 操作系统不鼓励程序员对硬件直接操作, 而是由操作系统的 API 提供了一个简单的中间应用层来间接对硬件进行编程。但有时为了需要, 不得不打破这一应用层, 直接在 Kernel 模式上编程, 以提高程序的效率, 满足在很多应用中所需要的同步性与实时性。

进行 Kernel 模式的编程可以使用专门的工具, 这样使编程更简单, 更可靠, 抛弃了传统的冗长的汇编程序和烦琐的调试过程。

下面举一个简单例子来说明如何在 Kernel 模式下进行编程。

现编制一个工作在 Kernel 模式下的基于 PCI 总线的数据采集卡驱动程序, 它是一个动态设备, 可以接收数据并送入指定的缓存区, 采集卡采用 DMA 及中断方式传送数据, Kernel 模式主程序主要代码如下, 其它程序略。

```
HCLIENTDEVICE g_ hDevice = NULL ;
HDMABUFFER g_ hDmaBuffer = NULL ; // handle to the DMA buffer
AGENT_PROC(UnloadDriver, VOID) (void) //called when the device is no longer in use...
{
    if ( g_ hDmaBuffer) // Close device objects Interrupts, FIFO's, etc.
        DaCloseDmaBuffer(g_ hDmaBuffer);
}
AGENT_PROC(RunDmaTest, DEVSTATUS) ( )
{
    DEVSTATUS      devStatus = DEVSTATUS_SUCCESS ;
    DMABUFFERINFO      dmaBufferInfo ;

    DebugMessageFmtA("Executing RunPCIDma() \n");

    if(API_SUCCESS (devStatus = DaQueryDmaBuffer
            (g_ hDmaBuffer, &dmaBufferInfo)) )
    {
        int x = 0;
        ULONG segmentSize ;
        PVOID pVirtualAddress ;
        DMA_ADDRESS physicalAddress ;

        for (x=0; API_SUCCESS(devStatus) &&
            (ULONG(x) < dmaBufferInfo.dmaNumberOfSegments) ; x++)
        {
            devStatus = DaDmaBufferGetSegment
                (
                    g_ hDmaBuffer,
```

```
                                    x,
                                    & segmentSize,
                                    & pVirtualAddress,
                                    & physicalAddress
                                );
            }
        }
        return devStatus ;
}

AGENT_PROC(StartDriver, DEVSTATUS) (void)   //called when the Kernel is loaded
{
        DEVSTATUS       devStatus;
        if (API_SUCCESS(devStatus=
                DaOpenDevice(L"PCIDma", &g_hDevice, NULL)))
        {
                DMABUFFERINFO dmaBufferInfo ;

                if (API_SUCCESS(devStatus = DaCreateDmaBuffer
                        (
                            g_hDevice,
                            L"PCIDMA",
                             8192,
                             4096,
                             FALSE,
                             &g_hDmaBuffer,
                             &dmaBufferInfo                    )) )
            {
                devStatus = RunPCIDma() ;           }
                DaCloseDevice(g_hDevice) ;
        }
        return devStatus;
}
```

4 结　论

实验结果表明,在 Kernel 模式下,可以对采集卡直接操作,完成对内存物理地址、IRQ、DMA 通道的直接编程,且由于优先级最高,设备驱动程序不受其它应用程序的干扰,能很好地完成指定的工作,而其它应用程序也能正常运行,说明在 Kernel 模式下的虚拟设备能够最大限度地满足实时性和并发性的要求,同时充分利用了 Windows 系统多线程、多任务的特性,对于后期的图像处理工作有很大现实意义。

参 考 文 献

1　Matt Pietrek. Windows 95 system programming secrets . IDG Books Worldwide, 1995

2　Microsoft corp. Windows 98 DDk documentation, 1998
3　Microsoft corp. Win32 SDk documentation, 1997
4　Vireo corp. Driver agent documentation, 1998

Kernel Mode and Virtual Device

WANG Pei, YUAN Xiao-Bing, WANG Guo-Hui
(Changchun Institute of Optics and Fine Mechanics, Chinese Academy of Sciences, Changchun 130022)

Abstract

Modern operating systems insulate the application programmer from needing to deal directly with hardware. It's difficult to the real time image processing. This possible using a simple, powerful interface is made through studying Kernel mode of Windows.

Key words: Windows system, Kernel mode, Virtual device

王　沛　女, 1970 年 10 月出生, 1992 年 7 月毕业于浙江大学信息与电子工程学系, 1997 年 3 月于中国科学院长春光机所获得硕士学位, 目前在长春光机所攻读博士学位。所从事的工作和研究方向为计算机图像处理。