Windows 驱动程序开发

——概念剖析

郑玉彤　王志娟

中央民族大学出版社

Windows 驱动程序开发

——概念剖析

郑玉彤　王志娟

中央民族大学出版社

图书在版编目(CIP)数据

Windows 驱动程序开发——概念剖析/郑玉彤，王志娟编著．—北京：中央民族大学出版社，2006.7
ISBN 7-81108-234-9

Ⅰ.W　Ⅱ.①郑…②王…　Ⅲ.窗口软件，Windows—驱动程序—程序设计　Ⅳ.TP316.7

中国版本图书馆 CIP 数据核字(2006)第 086532 号

Windows 驱动程序开发——概念剖析

编　　著　郑玉彤　王志娟
责任编辑　路　网
封面设计　赵秀琴
出 版 者　中央民族大学出版社
　　北京市海淀区中关村南大街 27 号　　邮编：100081
　　电话：68472815(发行部)
　　68932218(总编室)
　　传真：68932751(发行部)
　　68932447(办公室)
发 行 者　全国各地新华书店
印 刷 者　北京华正印刷有限公司
开　　本　880×1230(毫米)　1/32　印张：8.875
字　　数　220 千字
版　　次　2006 年 7 月第 1 版　2006 年 7 月第 1 次印刷
书　　号　ISBN 7-81108-234-9/TP·11
定　　价　16.00 元

内容简介

本书介绍 Windows 驱动程序的基本概念、结构、原理和实现技术。尤其从编程的角度，对概念、原理部分进行深入浅出的剖析。

全书由 4 部分组成：第 1 部分介绍 Windows 驱动程序编程的基础知识，由第 1 章组成；第 2 部分讲述驱动程序的基本结构，由第 2 到第 4 章组成；第 3 部分讲述如何处理即插即用硬件、电源管理、定时器、驱动程序分层等问题，包括第 5 章至第 9 章；第 4 部分为附录，介绍 Windows 驱动程序调试和开发的基本内容。

本书可帮助程序员理解 Windows 驱动程序的基本概念、结构和工作原理，对学习、开发 Windows 驱动程序有指导作用，可作为编程人员的技术参考书。

目　录

第 1 章　Windows 2000 内核模式驱动程序编程基础

1.1　内核模式驱动程序与应用程序的对比

1.1.1　内核模式(kernel mode)与用户模式(user mode)

内核模式又称核心模式，是一种处理器访问模式。在内核模式下，处理器处于最高优先级运行状态，可以执行处理器指令集中的任何指令，能够直接访问全部内存空间和控制整个系统资源。在操作系统中，实现诸如硬件资源分配、进程调度等基本功能以及直接与硬件打交道的内核模式驱动程序等都是以核心模式的方式运行的。为了实现保护和资源共享，系统还支持另一种处理器访问模式——用户模式。在用户模式下，处理器处于最小优先级运行状态，不能执行特权指令，并且只能访问经过授权的存储区和系统资源。用户应用程序是以用户模式的方式来执行的。每当应用程序执行系统调用或者被中断挂起时，操作系统将执行模式从用户模式切换到内核模式。切换是通过使用一条特殊的处理器指令实现的。显然，两种运行模式具有不同的优先权等级，并具有自己的内存映射，也即自己的地址空间。因此通常也将这两种运行模式分别称做内核空间和用户空间。

1.1.2　Windows 的体系结构

1. Windows 2000 的操作系统采用分层结构，如图 1－1 所示。

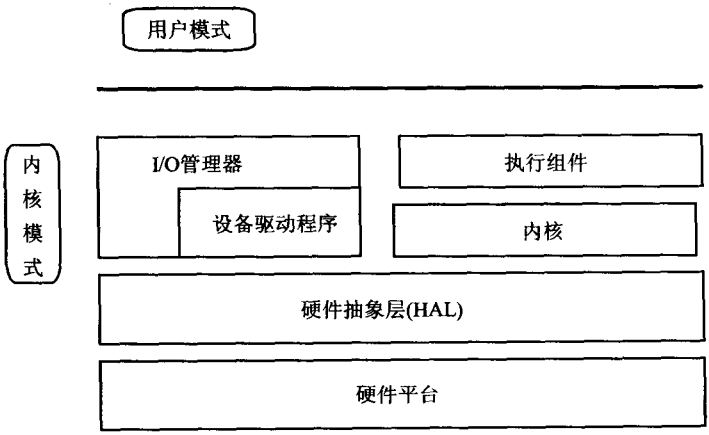

图 1－1　Windows 2000 操作系统的分层结构

硬件抽象层(HAL)：Windows 设计的一个至关重要的方面就是实现在多种硬件平台上的可移植性。HAL 是使这种可移植性成为可能的关键部分，其本身是一个可加载的核心态模块 Hal. dll。HAL 了解各种与硬件有关的细节，如 I/O 接口、中断控制器、总线、寄存器等以及任何体系结构专用的和依赖于计算机平台的函数。它将操作系统、驱动程序和平台隔离。从本质上说，驱动程序是与处理器、设备、平台相关的。为了实现平台无关性，驱动程序可以依赖 HAL 提供的宏来引用硬件寄存器和总线，也可以利用 I/O 管理器提供的抽象代码操作共享资源(如 DMA 通道)，这样驱动程序代码只需经过重新编译就可以移植到新的平台上，实现平台无关和总线无关。

内核(Kernel)：内核执行操作系统中最基本的操作，负责线

程安排和调度，维护线程上下文(包括 CPU 寄存器状态、线程 ID、优先权值，线程局部存储区)以及与线程有关的信息；负责陷阱处理和异常调度、中断处理和调度、多处理机同步以及提供执行部件使用的基本内核对象。内核代码短小紧凑、可移植性很好，一般来说，除了中断服务例程，正在运行的线程是不能抢先内核的。

执行部件：操作系统的主要任务是实现线程调度，其它的操作系统部件提供管理进程、内存、安全和 I/O 的服务，这些部件统称为执行部件，除 I/O 管理器外，执行部件是以模块的方式设计的。具体来讲，执行部件包括

- 系统服务接口：为从用户模式进入核心模式提供简洁而安全的入口。
- 对象管理器：几乎操作系统实现的所有服务都是以对象的形式提供的，包括文件、进程、线程、事件、内存区、信号量，甚至注册键等。用户程序通过句柄的方式加以引用。对象管理器负责对象的生成和撤消，为用户提供一种一致而安全的访问对象的方法。
- 配置管理器：为系统中软、硬件安排建立一个模型，并将其存入称做注册表的数据库中。设备驱动程序依赖于注册表发现执行环境。随着 PnP 的引入，对驱动程序而言，注册表的作用已大大减少了。
- 进程管理器：在 Windows 2000 中，进程是线程的执行环境，每个进程维护一个私有地址空间和安全特性。在 Windows 2000 中，线程是执行单位，而进程是资源分配单位，一个进程拥有多个线程。进程管理器负责管理进程并将其暴露给线程使用。进程管理器严格依赖于对象管理器、虚拟存储管理器等其它执行部件来完成其任务，因此也可以说进程管理器是其它低级系

统服务的一个高级抽象。设备驱动程序很少直接和进程管理器打交道而是利用操作系统提供的其它服务访问进程环境。例如在 I/O 传送过程中，驱动程序必须保证封锁驻留在进程私有地址空间的缓冲区，驱动程序可利用操作系统的内部例程实现加锁操作。

- 虚拟存储管理器：即 VMM(Virtual Memory Manager)。在 Windows 2000 中，每个进程拥有一个线性的 4GB 逻辑地址空间，只有低于 2GB 才能在用户模式下访问，用户代码和数据必须位于该区间，如果应用程序要访问共享库(DLLs)，共享库也必须位于该区间。每个进程拥有的高于 2GB 地址空间存放只能在核心模式下访问的代码和数据。这个高于 2GB 的地址空间由核心模式代码在进程之间共享。驱动程序映射到高 2GB 地址空间。VMM 代表整个系统进行存储管理。对用户模式程序而言，意味着分配和管理低于 2GB 的逻辑地址空间及物理内存，当进程需要的地址空间不在物理内存时，VMM 在磁盘文件和 RAM 之间进行页面交换，实现所谓的虚拟内存的概念。VMM 管理堆区，用于核心模式的内存分配。设备驱动程序可向 VMM 申请从堆区中分配专用的分页或不分页的内存来使用。
- 本地过程调用设备：LPC(Local Procedure Call)是一种支持本地进程间调用的机制。由于进程间调用涉及到不同的地址空间，内核执行部件 LPC 使进程间调用不仅成为可能而且能高效地完成。设备驱动程序不需要使用 LPC。

I/O 管理器：I/O 管理器从功能上讲属于执行部件，但与其它执行部件不同，它是以一组内核模式例程而不是模块的方式实现的。它为用户程序实现 I/O 操作提供了一个统一的抽象，

实现了用户模式下的 I/O 访问的设备无关性。I/O 管理器根据用户进程的请求生成一张工作单称作 IRP（I/O Request Packet）并将其递交给设备驱动程序，然后设备驱动程序根据 IRP 的要求完成 I/O 工作。I/O 管理器负责管理一些 I/O 数据结构，如文件对象、符号链接、IRP、MDL、驱动程序对象、设备对象、错误记录、控制器对象、IoTimer 、DpcForIsr、中断对象和适配器对象，通过这些部件，I/O 管理器把应用程序和系统组件连接到各种虚拟的、逻辑的和物理的设备上，形成了一个支持设备驱动程序的基本构架。因此 I/O 管理器是用户程序和设备驱动程序的接口。

设备驱动程序：是可加载的核心态模块（通常以 .sys 为扩展名），它们是 I/O 系统与相关硬件之间的接口。Windows 2000 下的设备驱动程序不直接控制硬件，而是调用 HAL 功能作为与硬件的接口。

2. Windows 2000 子系统

虽然 Windows 2000 执行部件定义和实现了操作系统的核心服务，但用户应用程序并不能直接使用它们而必须通过应用程序编程接口 APIs，APIs（Application Programming Interfaces）提供了关于操作系统服务的抽象，构成了应用程序运行所依赖的环境子系统。Windows 2000 支持以下环境子系统：

- 虚拟 DOS 机子系统：为旧的 DOS 应用程序提供一个 16 位 MSDOS 环境。
- 窗口子系统：为旧的 16 位 windows 应用程序提供一个 16 位窗口环境。
- POSIX 子系统：为类 Unix 风格的应用程序提供支持，符合 POSIX 1003.1 源码标准。
- OS/2 子系统：为 16 位 OS/2 应用程序提供执行环境。
- Win32 子系统：是 Windows 2000 固有的 API。所有其它

的环境子系统依赖于该子系统来完成工作。所有的 Windows 2000 应用程序依赖于 Win32 子系统构成执行环境。

任何应用程序都和特定的环境子系统相关。环境子系统通常以独立的用户模式进程的方式实现。在需要的时候由应用程序调用。应用程序的请求传给 LPC 执行部件，LPC 将其交给环境子系统，环境子系统可能直接完成请求任务，也可能将其交给适当的执行部件做进一步处理。

除了环境子系统以外，还有一些重要的系统部件是以用户模式的方式实现的。包括：

- 安全子系统：利用一组进程和动态库管理本地和远程访问的安全性。
- 服务控制管理器：也称做 scum 或 SCM，管理服务(后台程序)和设备驱动程序。
- RPC 定位和服务进程：支持分布式网络应用。

3. Win32 环境子系统

Win32 环境子系统在系统中具有特殊地位，它的实现不同于其它环境子系统。其中一部分是以用户模式实现而另一部分则是以核心模式实现。Win32 函数可分为三类：

- USER 函数：管理窗口、菜单、对话框及控件。
- GDI 函数：在物理设备(如屏幕、打印机)上执行画图操作。
- KERNEL 函数：管理非 GUI 资源如进程、线程、文件、同步服务。与执行部件提供的系统服务有密切的映射关系。

出于性能方面的考虑，从 NT 4.0 后，USER 和 GDI 移进内核，模块名为 WIN32K.SYS，如图 1-2 所示。

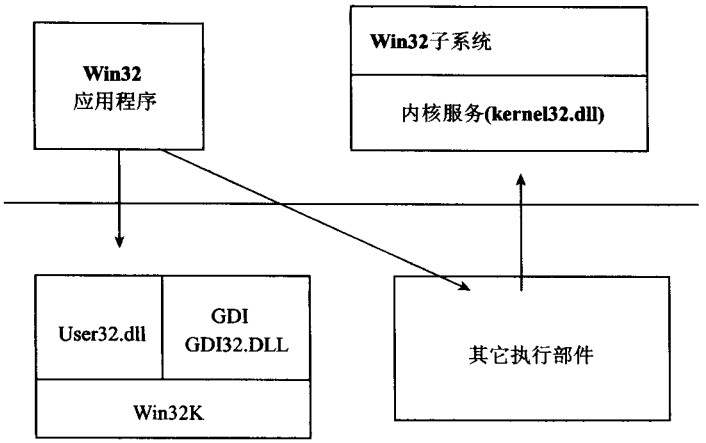

图 1－2　位于内核的 USER 和 GDI 函数

这样用户进程可直接将申请 USER 和 GDI 服务的请求通过系统服务接口传给内核模式的 USER 和 GDI 函数来实现。但申请 KERNEL 服务的请求仍要通过标准服务器进程 CSRSS. exe（用户－服务器运行子系统）访问用户模式下的 KERNEL 函数。

1.1.3　编程方式

内核模式驱动程序需要一个初始化函数，完成注册（以便服务于将来的某个请求）等任务，为以后调用驱动程序做准备。它还需要一个清除函数，以便在驱动程序卸载时，释放资源，撤消初始化函数所做的一切。

内核模式驱动程序的任务有两类：为应用程序提供某种系统服务（程序中的某些函数作为系统调用的一部分而执行），负责中断处理。

大多数小规模及中规模应用程序是从头至尾执行单个任务，而内核模式驱动程序却只是预先注册自己，以便服务于将来的