Broadview
www.broadview.com.cn

Windows
设备驱动程序WDF开发

武安河 编著

CD-ROM

本书配套光盘中含有书中所有实例的驱动程序和应用程序的全部源代码，以及生成的驱动程序和可执行的应用程序。除了USBSample和PCISample实例需要硬件设备的支持才能运行外，其他11个实例均可在Windows XP和Vista下运行。

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

Windows
设备驱动程序WDF开发

武安河 编著

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

内 容 简 介

Windows 最新的 WDF 驱动程序框架，包括 KMDF 和 UMDF，KMDF 是 WDM 的继续。本书主要介绍 KMDF 设备驱动程序的原理及编程方法；详细介绍 KMDF 对象及程序基本框架，KMDF 和应用程序之间的通信、即插即用和电源管理的编程技术，过滤器驱动程序，KMDF 访问硬件设备、处理硬件中断、实现 DMA 操作的编程技术，以及 USB 接口和 PCI 接口设备驱动程序 KMDF 的开发。还对 UMDF 设备驱动程序作了编程入门介绍。本书附有 13 个典型的编程实例，便于读者学习和掌握。

本书既适合具有一定计算机硬件及 C/C++语言基础的计算机应用开发人员阅读，也适合作为计算机应用开发人员和高等院校学生的实用参考书。

图书在版编目（CIP）数据

Windows 设备驱动程序 WDF 开发 / 武安河编著. —北京：电子工业出版社，2009.4
ISBN 978-7-121-08439-3

I. W… II. 武… 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（2009）第 030105 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：22.75 字数：413 千字
印 次：2009 年 4 月第 1 次印刷
印 数：4000 册 定价：49.00 元（含光盘 1 张）

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 设备驱动程序，过去是 WDM（Windows Driver Model）框架，编程复杂，初学者难以掌握其编程要领。为了解决这一问题，微软对 WDM 驱动程序的架构作了改进，形成了全新的 WDF（Windows Driver Foundation）框架结构。它提供了面向对象和事件驱动的驱动程序开发框架，大大降低了开发难度。从现在开始，掌握 Windows 设备驱动程序的开发人员，由过去的“专业”人士，将变为“普通”大众。

WDF 驱动程序包括两个类型，一个是内核级的，称为 KMDF（Kernel-Mode Driver Framework），为 SYS 文件；另一个是用户级的，称为 UMDF（User-Mode Driver Framework），为 DLL 文件。

- 内核模式驱动程序：这类驱动程序作为内核模式操作系统组件的一部分执行，它们管理 I/O、即插即用、内存、进程和线程、安全等。内核模式驱动程序通常为分层结构。
- 用户模式驱动程序：这类驱动程序通常提供 Win32 应用程序与内核模式驱动程序或其他操作系统组件之间的接口。用户模式驱动程序支持基于协议或基于串行总线（如摄像机和便携音乐播放器）的设备。

WDF 的运行平台有：Microsoft Windows Server® 2008、Microsoft Windows Vista™、Microsoft Windows Server® 2003、Microsoft Windows XP、Microsoft Windows 2000 (KMDF only)。

本书内容

本书主要讨论 KMDF，KMDF 是 WDM 的继续，可以把 WDM 驱动程序转换为 KMDF 驱动程序。还对 UMDF 设备驱动程序作了编程入门介绍。

过去，我研究过如何用 DriverStudio 开发工具开发 Windows 下的 WDM 设备驱动

程序，承蒙电子工业出版社的厚爱，出版了《Windows 2000/XP WDM 设备驱动程序开发》一书。

WDF 的简单化编程，导致 DriverStudio 退出了 Windows 设备驱动程序的开发，将 DriverStudio 定格在 3.2 版本。

从形式上看，KMDF 的对象和 DriverStudio 的类，具有相似的地方。本书的内容安排和《Windows 2000/XP WDM 设备驱动程序开发》的基本一致，具体安排如下：

章　名	主 要 内 容
第 1 章　Windows 2000 和 WDM 驱动程序	KMDF 是 WDM 的继续，KMDF 构建在 WDM 之上。对于一些 Windows 驱动程序的基本概念，初学者仍需要了解，如 Windows 2000 中的组件、驱动程序种类，以及 WDM 驱动程序特点等
第 2 章　KMDF 驱动程序框架	KMDF 驱动程序框架由对象和事件回调例程构成。KMDF 框架中所有的事物都由对象来表示，各种事件处理都由事件回调例程来完成。学习 KMDF 编程，主要是学习 KMDF 的各种对象、对象函数和事件回调例程的编程
第 3 章　基本对象	KMDF 提供了许多对象，本章只介绍一些基本的对象，如 WDFREQUEST 对象、WDFQUEUE 对象等，还介绍了数据同步访问、字符串操作和 QueueSample 实例
第 4 章　KMDF 驱动程序编程入门	介绍 Windows Vista 下的 KMDF 驱动程序编程入门，包括 KMDF 驱动程序编程环境的建立，KMDF 的创建、生成及安装过程，以及 Win32 Console 和 MFC 两种应用程序的编程
第 5 章　KMDF 驱动程序和应用程序之间的通信	介绍应用程序与 KMDF 之间的通信，包括应用程序对 KMDF 的通信，以及 KMDF 对应用程序的通信
第 6 章　即插即用例程	主要介绍 PnP 组件、即插即用例程的加载和卸载顺序及 PnpPowerSample 实例等
第 7 章　电源管理	主要介绍系统电源状态与设备电源状态、电源管理控制标志位、设备的唤醒特征和空闲检测、电源管理编程及 IdleSample 实例等
第 8 章　KMDF 过滤器驱动程序	描述如何写一个过滤器驱动程序，该驱动程序可位于功能驱动程序的上面或下面，它通过过滤流经它的 IRP 来修改设备的行为
第 9 章　USB 设备开发	主要介绍 USB 设备的配置/接口/端点、USB 数据的传输方式、USB 描述符、USB 编程对象、USB 编程（如激活配置与中止配置、同步操作、异步操作）及 USBSample 实例等
第 10 章　PCI 设备驱动程序开发	本章就硬件访问、中断处理和 DMA 传输 3 个方面展开讨论，并以 CY7C09449 芯片为例，给出一个经过测试的 PCI 设备驱动程序实例——PCISample 实例。另外，还给出了一个不依靠硬件设备的 DMASample 实例

续表

章　名	主 要 内 容
第11章　UMDF驱动程序编程入门	介绍 UMDF 驱动程序编程入门，包括 UMDF 驱动程序的创建、生成及安装，以及一个简单的实例编程
第 12 章　WinDbg 使用介绍	WinDbg 是微软提供的一个功能非常强大的调试软件，可以在源代码级别调试 Windows 下的 WDF 驱动程序。本章主要介绍用 WinDbg 调试 KMDF 和 UMDF

微软推荐的 WDF 书籍是《Developing Drivers with the Microsoft Windows Driver Foundation》，作者是 Penny Orwick 和 Guy Smith。

微软提供的学习 WDF 的中文网站地址是：www.microsoft.com/china/whdc。

开发工具

微软提供的 WDF 驱动程序开发工具包 WDK 的最新版本是 WDK 1.7，可以从其网站下载。

本书实现 WDF 驱动程序及应用程序实例所用的工具是：Visual C++ 6.0 和 WDK 1.7。

本书配套光盘

本书配套光盘中含有书中所有实例的驱动程序和应用程序的全部源代码，以及生成的驱动程序和可执行的应用程序。除 USBSample 和 PCISample 实例因需要硬件设备的支持，读者无法运行外，其他 11 个实例均可在 Windows XP 和 Vista 下运行。

本书特点

- 编程入门：详细的步骤介绍，初学者的良师益友。
- 应用实例：对程序作了必要的注释，了解驱动程序编程的捷径。
- 简单实用：KMDF 的编程非常简单，Windows 硬件开发工程师必须掌握。
- 理解深刻：笔者精通 Windows 设备驱动程序开发和硬件开发，所有实例均为自己创造，每个实例都有其独特意义，笔者用实例诠释自己对 KMDF 的理解和认识。

微软推荐的 WDF 书籍是“理论派”，以讲解理论为主，讲得非常详细和全面；本书是“实战派”，全书以实例为主，用实例带你快速入门。

善于学习，勤于思考，勇于实践。用这句话与投身硬件开发的青年学子共勉。

希望这本书能对您现在或将来的工作有所帮助。

本书得以顺利出版，要特别感谢周利莉的一贯支持；感谢何海洋和赵庆花的支持；感谢陈建波和何亚垒的支持；感谢朱沭红编辑和电子工业出版社。

由于作者的理论水平有限，书中难免出现差错和遗漏，敬请广大计算机应用开发人员批评指正，联系 E-mail：jsj@phei.com.cn。

十年时间，跟踪 Windows 设备驱动程序的开发，驱动程序 VxD->WDM->KMDF，工具 VtoolsD->DDK、DriverWorks->WDK，但愿 WDF 长久。别了 WDM，别了 DriverStudio。

缺月挂疏桐，漏断人初静。谁见幽人独往来，缥缈孤鸿影。……

武安河

2009 年元旦

目　录

第 1 章

Windows 2000 和 WDM 驱动程序

虽然本书介绍 KMDF 的编程，但是 KMDF 是 WDM 的继续，KMDF 构建在 WDM 之上。对于一些 Windows 驱动程序的基本概念，初学者仍需要了解。所以，保留了参考文献[1]（即《Windows 2000/XP WDM 设备驱动程序开发（第 2 版）》一书）的第 1 章，供初学者阅读。

1.1 Windows 2000 组件概述

图 1-1 显示了 Windows 2000 操作系统环境的主要组件。在 Windows 2000 操作系统环境中，一部分组件运行在用户模式下，其他的则运行在内核模式下。

Windows 2000 操作系统包括许多内核模式组件，它们被精心地定义为功能相互独立的组件。对内核模式驱动程序设计者来说，最感兴趣的就是内核、I/O 管理器、即插即用（PnP）管理器、电源管理器、硬件抽象层、配置管理器、内存管理器、运行支持和进程结构组件。对另一些设计者来说，感兴趣的其他组件可能还包括对象管理器和安全引用监视器。

当用户模式程序需要读取设备数据时，它就调用 Win32 API 函数，如 ReadFile。Win32 子系统模块（如 KERNEL32.DLL）通过调用平台相关的系统服务接口实现该 API，而平台相关的系统服务将调用内核模式支持例程。在 ReadFile 调用中，调用首先到达

系统 DLL（NTDLL.DLL）中的一个入口点 NtReadFile 函数，然后这个用户模式的 NtReadFile 函数调用系统服务接口，最后由系统服务接口调用内核模式中的服务例程，该例程同样命名为 NtReadFile。

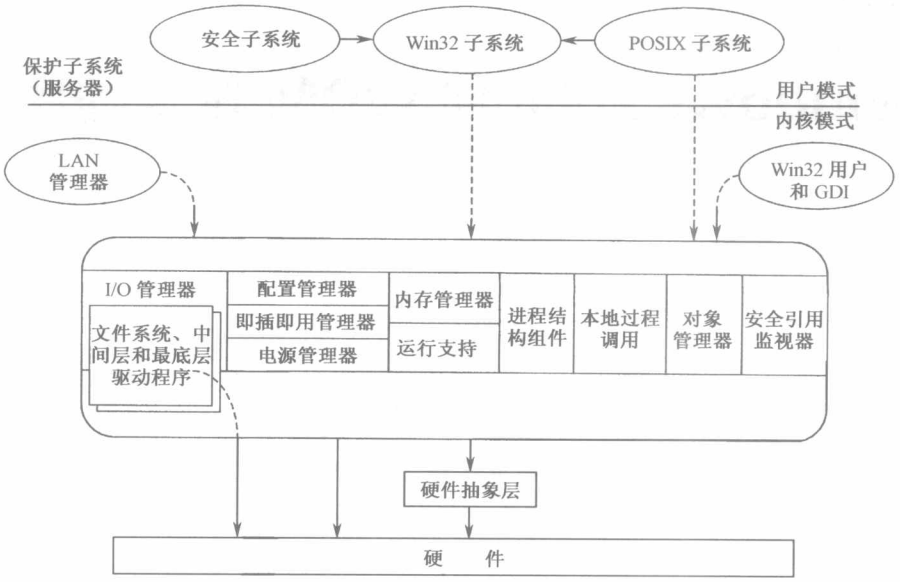

图 1-1　Windows 2000 操作系统环境的组件总览

系统中还有许多与 NtReadFile 相似的服务例程，它们同样运行在内核模式中，为应用程序请求提供服务，并以某种方式与设备交互。它们首先检查传递给它们的参数以保护系统安全或防止用户模式程序非法存取数据，然后创建一个称为“I/O 请求包（IRP）”的数据结构，并把这个数据结构送到某个驱动程序的入口点。在刚才的 ReadFile 调用中，NtReadFile 将创建一个主功能代码为 IRP_MJ_READ（DDK 头文件中的一个常量）的 IRP。实际的处理细节可能会有所不同，但对于 NtReadFile 例程，可能的结果是，用户模式调用者得到一个返回值，表明该 IRP 代表的操作还没有完成。用户模式程序也许会继续其他工作，然后等待操作完成，或者立即进入等待状态。不论哪种方式，设备驱动程序对该 IRP 的处理都与应用程序无关。

执行 IRP 的设备驱动程序最后可能会访问硬件。对于 PIO 方式的设备，一个 IRP_MJ_READ 操作将导致直接读取设备的端口（或者是设备实现的内存寄存器）。尽

管运行在内核模式中的驱动程序可以直接与其硬件会话，但它们通常都使用硬件抽象层（HAL）访问硬件。读操作最后会调用 READ_PORT_UCHAR 从某个 I/O 口读取单字节数据。HAL 例程执行的操作是平台相关的。在 Intel x86 计算机上，HAL 使用 IN 指令访问设备端口；在 Alpha 计算机上，HAL 使用内存读取指令访问设备实现的内存寄存器。

驱动程序完成一个 I/O 操作后，通过调用一个特殊的内核模式服务例程来完成该 IRP。完成操作是处理 IRP 的最后动作，它使等待的应用程序能够继续运行。

1.2　Windows 2000 中的驱动程序种类

如图 1-2 所示，Windows 2000 系统可以使用多种驱动程序。

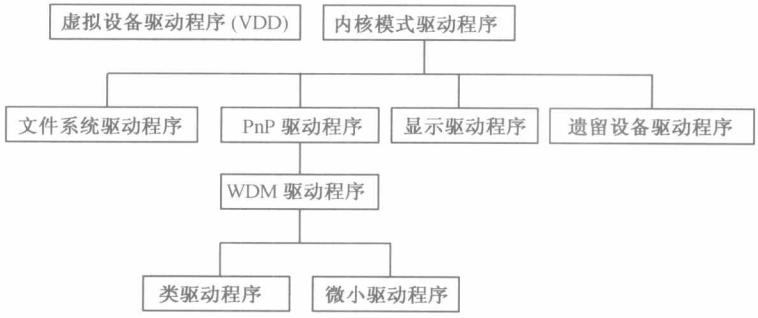

图 1-2　Windows 2000 中的设备驱动程序种类

- 虚拟设备驱动程序（VDD）是一个用户模式部件，可以使 DOS 应用程序访问 x86 平台上的硬件。VDD 通过屏蔽 I/O 权限掩码来捕获端口存取操作，它基本上是模拟硬件操作，这对于那些直接对微机硬件编程的应用程序特别有用。

内核模式驱动程序的分类如下：

- PnP 驱动程序就是一种遵循 Windows 2000 即插即用协议的内核模式驱动程序。
- WDM 驱动程序是一种 PnP 驱动程序，它同时还遵循电源管理协议，并能在 Windows 98、Windows 2000 和 Windows XP 间实现源代码级兼容。WDM 驱动程序还细分为类驱动程序（class driver）和微小驱动程序（minidriver），类驱动程序管理属于已定义类的设备，微小驱动程序给类驱动程序提供厂商专有的支持。

- 文件系统驱动程序在本地硬盘或网络上实现标准 PC 文件系统模型（包括分级目录结构和命名文件概念）。
- 显示驱动程序是用于显示和打印设备的内核模式驱动程序。
- 遗留设备驱动程序也是一种内核模式驱动程序，它直接控制一个硬件设备而不用其他驱动程序帮助。这种驱动程序主要包括 Windows NT 早期版本的驱动程序，它们可以不加修改地运行在 Windows 2000 中。

1.3 WDM 驱动程序特点

1.3.1 内核模式驱动程序的设计目标

内核模式驱动程序的设计目标与 Windows 2000 的许多设计目标相符合，特别是系统 I/O 管理器部分。这些设计目标包括：

- 平台之间的可移植性；
- 硬件和软件的可配置性；
- 永远抢占优先和永远中断；
- 多处理器平台上的多处理器安全；
- 基于对象；
- 带可重用 I/O 请求包（IRP）的包驱动 I/O；
- 支持异步 I/O。

1. 平台之间的可移植性

内核模式驱动程序的源代码应该可以移植到所有 Windows NT 平台上。WDM 驱动程序在其定义中就规定了其源代码可以在 Windows 98、Windows 2000 和 Windows XP 之间相互移植。为了实现这种可移植性，驱动程序应该全部用 C 语言编写，并且只使用 ANSI C 标准规定的语言元素。应避免使用编译器厂商专有的语言特征，并避免使用没有被操作系统内核输出的运行时库函数。如果不能避免驱动程序中的平台依赖，至少

应该用条件编译指令隔离这些代码。如果严格遵循这些设计方针，仅需要重新编译连接源代码，生成的驱动程序就可以运行在任何新的 Windows NT 平台上。

如果仅使用 WDM.H 中声明的内核模式支持函数，那么可以很容易地实现驱动程序的源代码级兼容。但是，操作系统之间的差异会在某些地方影响驱动程序的移植性。

2．硬件和软件的可配置性

内核模式驱动程序应避免对设备特征或某些系统设置作绝对假设，这些系统设置会随着平台的改变而变化。例如，在 x86 平台上，标准串行口使用一个专用的 IRQ 和 8 个 I/O 端口，这些数值持续 20 年从未改变。把这些值直接写到驱动程序中将使驱动程序失去可配置性。

为了实现可配置性，首先应该在代码中避免直接引用硬件，即使是在平台相关的条件编译块中也是这样。应该使用 HAL 工具或调用低级总线驱动程序，或者实现一个标准或定制的控制接口，并通过控制面板程序与用户交互。另外，还应该支持 Windows 管理诊断（WMI）控件，这种控件允许用户和管理员在分布式企业环境中配置硬件特征。最后，应该使用注册表作为配置信息的数据库，这可以使配置信息在系统重新启动后仍然存在。

3．永远抢占优先和永远中断

Windows 2000 是多任务操作系统，可以为任意多个线程分配 CPU 时间。在大部分时间中，驱动程序例程执行在可以被其他线程（在同一个 CPU 上）抢先的环境中。线程抢先取决于线程的优先级，系统使用系统时钟为线程分配 CPU 时间片。

Windows 2000 还使用了一个中断优先级的概念，即 IRQL。本章后面将详细讨论 IRQL，在这里先简单介绍一下。你可以认为 CPU 中有一个 IRQL 寄存器，它记录着 CPU 当前的执行级别。有三个 IRQL 值对设备驱动程序有重要意义：PASSIVE_LEVEL、DISPATCH_LEVEL 和设备中断请求级 DIRQL。在大部分时间里，CPU 执行在 PASSIVE_LEVEL 级上，所有的用户模式代码也运行在 PASSIVE_LEVEL 级上，并且驱动程序的许多活动也都发生在 PASSIVE_LEVEL 级上。当 CPU 运行在 PASSIVE_LEVEL 级时，当前运行的线程可以被任何优先级大于它的线程抢先。然而，一旦 CPU 的 IRQL 大于 PASSIVE_LEVEL 级，线程抢先将不再发生，此时 CPU 执行在

使 CPU 越过 PASSIVE_LEVEL 级的任意线程上下文中。

你可以把高于 PASSIVE_LEVEL 级的 IRQL 看成是针对中断的优先级。这是一个与支配线程抢先机制不同的优先级方案，正如前面所说的，在 PASSIVE_LEVEL 级上没有线程抢先发生。但是，运行在任何 IRQL 级上的活动都可以被更高 IRQL 级上的活动中断。所以，驱动程序必须假定在任何时刻都可能失去控制权，而此时系统可能需要执行更基本的任务。

4. 多处理器平台上的多处理器安全

Windows 2000 可以运行在多处理器计算机上。Windows 2000 使用对称多处理器模型，即所有的处理器都是相同的，系统任务和用户模式程序可以执行在任何一个处理器上，并且所有处理器都平等地访问内存。多处理器的存在给设备驱动程序带来了一个困难的同步问题，因为执行在多个 CPU 上的代码可能同时访问共享数据或共享硬件资源。Windows 2000 提供了一个同步对象——自旋锁（Spin Lock），驱动程序可以使用它来解决多处理器的同步问题。

5. 基于对象

Windows 2000 内核是基于对象的，即驱动程序和内核例程使用的许多数据结构都有共同的特征，这些特征由对象管理器集中管理。这些特征包括名称、参考计数、安全属性等。在内部，内核中包含了许多执行公共对象管理的方法例程，例如打开和关闭对象。

驱动程序使用内核部件输出的服务例程来维护对象或对象中的域。某些内核对象，比如内核中断对象是不透明的，DDK 头文件中没有其数据成员的声明；其他内核对象，比如设备对象或驱动程序对象则是部分不透明的，DDK 头文件中声明了其结构的全部成员，但 DDK 文档中仅描述了可访问的成员并警告驱动程序开发者不要直接访问或修改其他成员。对于驱动程序必须间接访问的不透明域，可以用支持例程访问。部分不透明的对象类似于 C++的类，有任何人都能访问的公共成员，还有必须通过方法才能访问的私有成员和保护成员。

6. 带可重用 I/O 请求包（IRP）的包驱动 I/O

I/O 管理器和设备驱动程序使用 IRP 来管理 I/O 操作的具体细节。首先，某个内核