

# WDF 设备驱动程序的设计与实现

李正平, 徐超, 陈军宁, 谭守标

(安徽大学 电子科学与技术学院, 安徽 合肥 230039)

**摘要:** WDF 是微软提出的全新驱动程序模型, 它提供了面向对象、事件驱动的驱动程序开发框架, 对它的研究是设计高效稳定设备驱动程序的基础。文中介绍了 WDF 模型的特点, 阐述了 WDF 对象模型, 分析了 WDF 驱动程序的基本结构, 并通过一个简单的实例介绍了基本编程技巧。

**关键词:** WDF 模型; 对象模型; 微软 Windows

中图分类号: TP311

文献标识码: A

文章编号: 1673-629X(2007)05-0228-03

## Design and Implementation of WDF Device Driver

LI Zheng-ping, XU Chao, CHEN Jun-ning, TAN Shou-biao

(School of Electronic Science & Technology, Anhui University, Hefei 230039, China)

**Abstract:** WDF is the next generation Microsoft Windows driver model, which supports an object-oriented, event-driven driver development framework. It is very important to study WDF for writing effective and robust driver. Introduces the characteristics of WDF model, addresses the WDF object model, and analyzes the structure of WDF driver. At last, the basic programming technique is introduced based on a simple example.

**Key words:** WDF model; object model; Microsoft Windows

### 0 引言

设备驱动程序是硬件设备连接到计算机系统的软件接口, 任何设备都必须有相应的驱动程序才能在计算机系统上正常工作。设备驱动程序的优劣直接关系到整个系统的性能和稳定性, 因此, 设计和开发稳定高效的驱动程序具有重要意义。

WDF(Windows Driver Foundation)是微软提出的下一代全新的驱动程序模型<sup>[1,2]</sup>, 它是在 WDM(Windows Driver Model)<sup>[3]</sup>的基础上发展而来的, 支持面向对象、事件驱动的驱动程序开发, 提供了比 WDM 更高层次抽象的高度灵活、可扩展、可诊断的驱动程序框架。WDF 框架管理了大多数与操作系统相关的交互, 实现了公共的驱动程序功能(如电源管理、pnp 支持), 隔离了设备驱动程序与操作系统内核, 降低了驱动程序对内核的影响。

WDF 提供了两个框架: KMDF(内核模式驱动程序框架)和 UMDF(用户模式驱动程序框架)。文中只介绍 KMDF 的设计与实现。

### 1 WDF 对象模型

KMDF 框架支持面向对象、事件驱动的驱动程序模型<sup>[1,4]</sup>。它定义了一系列的对象用来表示设备、驱动、中断等, 每个对象有对应的属性、方法和事件。驱动程序利用这些方法创建对象、设置属性和响应事件。

(1) 框架定义的主要对象有:

WDFDRIVER 对象: 对应于 WDM 中的 DRIVER\_OBJECT, 描述驱动在内存中的实例, 包括加载的位置、有关的属性和所管理的设备。

(2) WDFDEVICE 对象: 对应于 WDM 中的 DEVICE\_OBJECT, 描述由驱动程序管理的单个设备实例。设备可以是命名的也可以是未命名的, 用户模式程序可以通过设备接口或设备名称访问设备。WDFDEVICE 对象具有丰富的属性, 如 pnp 和电源管理相关的事件处理回调函数(callbacks)。

(3) WDFREQUEST 对象: 对应于 WDM 中的 IRP, 表示一个 I/O 请求。

(4) WDFQUEUE 对象: 每个 WDFQUEUE 对象和一个 WDFDEVICE 对象关联, 描述一个特殊的 I/O 请求队列。它具有一系列的事件处理回调函数, 当 I/O 请求进入队列时, 框架将自动调用驱动程序中对应的 callback。

收稿日期: 2006-08-15

作者简介: 李正平(1979-), 男, 安徽宣城人, 博士, 副教授, 研究方向为数据采集与处理、操作系统内核、核技术应用以及嵌入式系统。

(5) WDFINTERRUPT 对象:表示设备中断。驱动程序可以通过 WDFINTERRUPT 对象的中断使能和禁止事件处理 callbacks 使能或禁止设备中断;通过 ISR 和 DPCforISR 例程处理设备中断。

WDF 的对象模型是层次化的模型。WDFDRIVER 对象是根对象,其他对象都是它的子对象。对于大多数对象,驱动程序在创建它们的时候可以指定父对象,如果没有指定,则框架默认其父对象为 WDFDRIVER 对象。

WDF 大大简化了 WDM 中的 pnp 和电源管理的开发。WDF 框架为设备停止、设备删除、电源状态切换等 pnp 和电源管理事件提供了适合的缺省行为,驱动程序本身不再纠缠于复杂的 pnp 和电源管理事件处理。此外,WDF 还集成了请求队列的支持,一个设备可以有多个请求队列,每个请求队列可以有一种模式。最简单的是 WdfIoQueueDispatchSerial 模式,在这种模式下,请求队列将请求串行化后再处理;而 WdfIoQueueDispatchParallel 模式则自动在每个请求到来时调用相应的回调函数;最后 WdfIoQueueDispatchManual 模式允许驱动程序手工分发请求,类似于 WDM 的工作方式。

在 WDM 驱动程序中,I/O 请求的取消是一个复杂难以理解的过程,开发人员必须有对内核深刻的理解才能正确处理 I/O 请求的取消。WDF 框架支持内建的 I/O 请求取消处理,使得驱动程序处理取消 I/O 请求的工作大大简化。

## 2 WDF 设备驱动程序的实现

WDF 驱动程序的结构与 WDM 相似。WDF 驱动程序的标准入口函数是 DriverEntry<sup>[2,5]</sup>。与 WDM 不同,WDF 的 DriverEntry 只负责创建和初始化 WDFDRIVER 对象,告诉 WDF 框架处理添加新设备连接的回调函数。

```
NTSTATUS DriverEntry(PDRIVER_OBJECT DriverObj, PUNICODE_STRING RegistryPath)
```

```
{
    NTSTATUS status;
    WDF_DRIVER_CONFIG config;
    WDFDRIVER hDriver;
    // 初始化驱动配置结构,指定设备添加事件 callback
    WDF_DRIVER_CONFIG_INIT_NO_CONSTRAINTS
    (&config, MyEvtDeviceAdd);
    // 创建 WDFDRIVER 对象
    status = WdfDriverCreate(DriverObj,
        RegistryPath,
        WDF_NO_OBJECT_ATTRIBUTES,
```

```
&config, // 指向 config 结构的指针
    NULL);
    return(status);
}
```

每当有新设备连接到系统时,WDF 框架自动调用 EvtDeviceAdd 设备添加 callback。该回调函数初始化 pnp 和电源管理相关结构,设置相应的事件处理 callbacks,然后创建 WDFDEVICE 对象和符号连接,初始化请求队列、中断处理等相关结构,设置相应的回调函数。

```
WDFSTATUS MyEvtDeviceAdd ( WDFDRIVER Driver,
    PWDFDEVICE_INIT DeviceInit)
```

```
{
    WDFSTATUS status = STATUS_SUCCESS;
    WDF_PNPPOWER_EVENT_CALLBACKS pnpPowerCallbacks;
    WDF_OBJECT_ATTRIBUTES objAttributes;
    WDFDEVICE device;
    PMY_DEVICE_CONTEXT devContext;
    WDF_IO_QUEUE_CONFIG ioCallbacks;
    WDF_INTERRUPT_CONFIG interruptConfig;
    // 初始化 PnpPowerCallbacks,设置与 PnP 和电源管理相关
    的事件回调函数
    WDF_PNPPOWER_EVENT_CALLBACKS_INIT
    (&pnpPowerCallbacks);
    pnpPowerCallbacks.EvtDevicePrepareHardware = MyEvtPrepareHardware;
    pnpPowerCallbacks.EvtDeviceReleaseHardware = MyEvtReleaseHardware;
    pnpPowerCallbacks.EvtDeviceD0Entry =
    MyEvtDeviceD0Entry;
    pnpPowerCallbacks.EvtDeviceD0Exit = MyEvtDeviceD0Exit;
    WdfDeviceInitSetPnpPowerEventCallbacks ( DeviceInit,
    pnpPowerCallbacks);
    // 创建设备对象,初始化相关的属性
    WDF_OBJECT_ATTRIBUTES_INIT(&objAttributes);
    WDF_OBJECT_ATTRIBUTES_SET_CONTEXT_TYPE
    (&objAttributes,MY_DEVICE_CONTEXT);
    // 命名设备
    status = WdfDeviceInitUpdateName(DeviceInit, L"\\device\\WDFDEMO");
    if (! NT_SUCCESS(status)) return(status);
    status = WdfDeviceCreate (&DeviceInit, &objAttributes,
    &device);
    if (! NT_SUCCESS(status)) return(status);
    devContext = MyGetContextFromDevice(device);
    devContext->WdfDevice = device;
    // 创建符号连接
```

```

status = WdfDeviceCreateSymbolicLink(device, L"\\Dos-
Devices\\WDFDEMO");
if (! NT_SUCCESS(status)) return(status);
// 创建和初始化请求队列
WDF_IO_QUEUE_CONFIG_INIT(&ioCallbacks, WdfIo-
QueueDispatchSerial, WDF_NO_EVENT_CALLBACK, WDF-
NO_EVENT_CALLBACK);
ioCallbacks.EvtIoDeviceControl = MyEvtDeviceControlIoctl;
status = WdfDeviceCreateDefaultQueue(device, &ioCallbacks,
WDF_NO_OBJECT_ATTRIBUTES, NULL);
if (! NT_SUCCESS(status)) return(status);
// 创建和初始化中断对象, MyIsr 和 MyDpc 分别是中断服
务例程和 DPC 例程
WDF_INTERRUPT_CONFIG_INIT(&interrupt
Config, FALSE, MyIsr, MyDpc);
interruptConfig.EvtInterruptEnable = MyEvtInterrupt En-
able; //中断使能
interruptConfig.EvtInterruptDisable = MyEvtInterrupt Dis-
able; //中断禁止
status = WdfInterruptCreate(device, &interruptConfig,
&objAttributes, &devContext->WdfInterrupt);
//一些其他初始化操作(略)
return(status);
}

```

开发 WDF 驱动程序下一步的工作就是编写各事件处理回调函数, 当相应事件发生时, WDF 框架会自动调用指定的回调函数进行处理。其中 EvtDevicePrepareHardware 回调函数在分配资源的时候被调用, 框架将分配给设备的资源传递给回调函数, 回调函数保存需要的资源, 将共享内存映射到内核虚拟地址空间。与此对应的是 EvtDeviceReleaseHardware 回调函数, 每当设备释放所占用的资源时, 框架都将调用它。

EvtDeviceD0Entry 和 EvtDeviceD0Exit 事件 callbacks 则分别在设备即将进入和离开 D0 电源状态时调用。EvtIoDeviceControl, EvtIoRead, EvtIoWrite 等回调函数分别用来处理 DeviceControl, Read, Write I/O 请求。当框架获得一个 I/O 请求时, 它首先确定该请求应该放入哪个请求队列。如果驱动程序没有提供指定的队列, WDF 框架默认将请求放入缺省请求队列会自

动调用对应的回调函数。然后, 框架寻找处理该请求的回调函数, 如果驱动程序提供了相应的 callback, 则调用它处理请求。对于没有指定回调函数的 I/O 请求, WDF 调用 EvtIoStart 回调函数处理。如果 EvtIoStart callback 也不存在, 框架将返回 STATUS\_NOT\_SUPPORTED。

设备中断的管理由 EvtInterruptEnable callback、EvtInterruptDisable callback、中断服务例程 (ISR) 和 DpcForIsr 例程实现。WDF 框架在调用 EvtDeviceD0Entry callback 和注册 ISR 后, 通过调用 EvtInterruptEnable 回调函数使能设备中断; 而 EvtInterruptDisable 回调函数则在设备离开 D0 状态, EvtDeviceD0Exit callback 调用前获得调用, 完成禁止设备中断的工作。此外, 中断服务例程和 DpcForIsr 例程具体完成中断服务的功能, 与 WDM 驱动程序相似。

### 3 结束语

WDF 驱动程序模型是 Microsoft 下一代驱动程序模型, 它提供的 KMDF 框架为开发内核模式驱动程序提供了一个面向对象、事件驱动的开发框架, 极大地简化了设备驱动程序的开发, 避免了由于驱动程序的错误导致整个操作系统崩溃的现象。文中分析了 WDF 的对象模型和驱动程序结构, 并给出了基本的实例。WDF 驱动程序有很多值得深入研究的地方, 今后还要对其作进一步的探讨。

#### 参考文献:

- [1] Microsoft Corporation. Architecture of the Windows Driver Foundation [EB/OL]. 2005. <http://www.microsoft.com/whdc/driver/wdf/default.mspx>.
- [2] OSR online. The Future Is Now—The WDF Kernel Mode Framework [EB/OL]. 2004. <http://www.osronline.com/article.cfm?article=287>.
- [3] Cant C. Windows WDM 设备驱动程序开发指南 [M]. 北京: 机械工业出版社, 2000.
- [4] OSR online. A New Framework [EB/OL]. 2004. <http://www.osronline.com/article.cfm?id=289>.
- [5] Microsoft Corporation. Microsoft Corporation Windows DDK document [M]. USA: Microsoft Corporation, 2002.

(上接第 227 页)

发展, 2002, 12(4): 53-54.

- [4] Tag Libraries: Pager Tag Library [EB/OL]. 2004-01-24. <http://jsptags.com/tags/navigation/pager/index.jsp>.
- [5] 飞思科技产品研发中心. JSP 应用开发详解 [M]. 第 2 版.

北京: 电子工业出版社, 2005: 310-316.

- [6] Ayers D, Bell J, Calvert Bettis C. Java 数据编程指南 [M]. 戴英, 等译. 北京: 电子工业出版社, 2002.