第9章 Windows设备驱动程序设计

- Windows 2000的设备驱动程序
- WDM的核心概念和数据结构
- WDM驱动程序的结构
- WDM驱动程序的编程实例

WDM (Windows Driver Model)

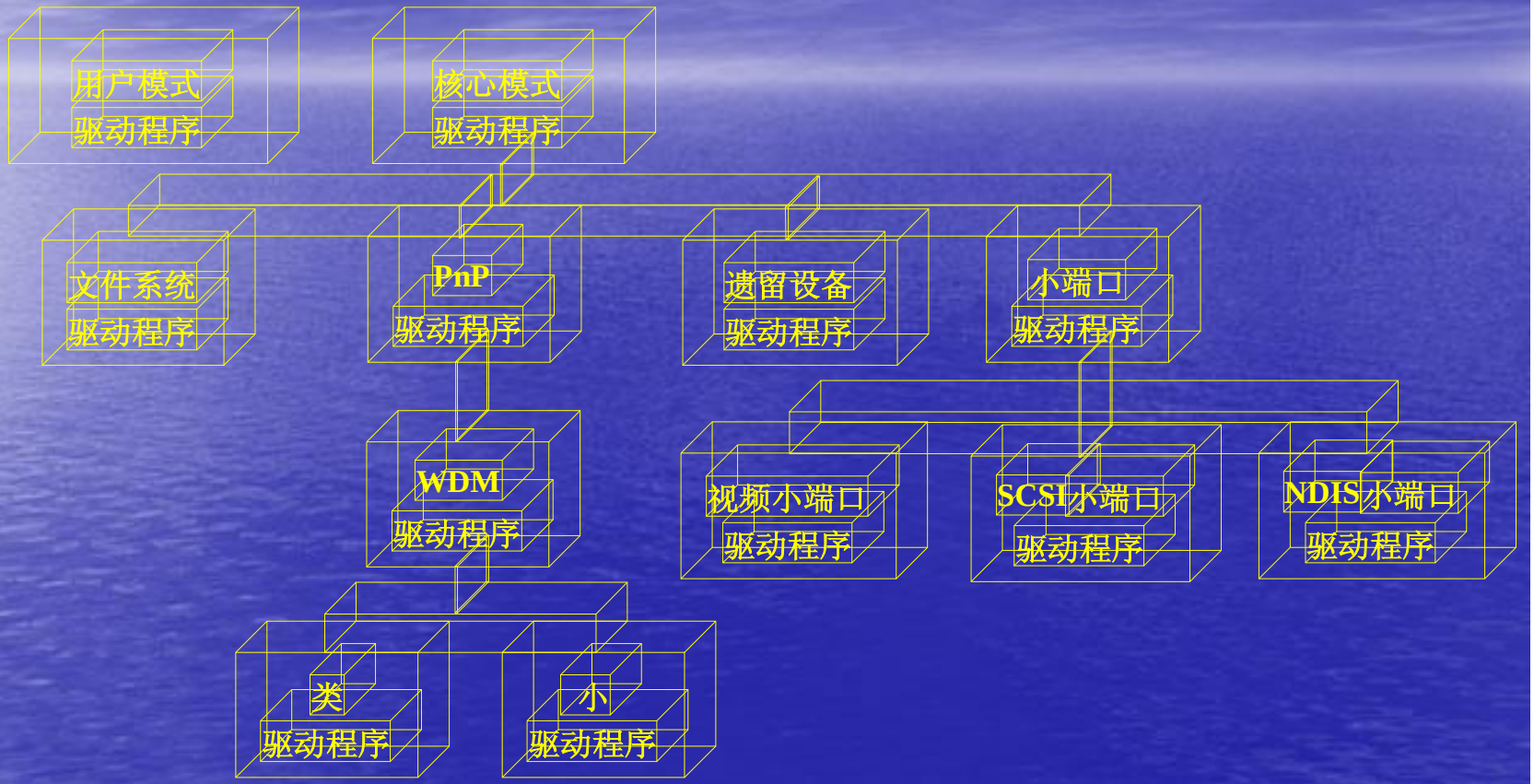
- 微软提出的一种全新的设备驱动程序模型,它是在Windows2000/xp内核驱动程序模型(Kernel2mode Device DriverModel) 的基础上发展而来的.
- 增加了对即插即用(PnP) ,电源管理(Power
- management) ,Windows 管理接口(WMI) 等新的硬件标准的支持.
- WDM模型是一个分层化的驱动程序模型

- 微软公司推出的WINDOWS2000/XP支持WDM，它允许用户通过开发WDM过滤器驱动程序方便地修改已有驱动程序的行为，从而以很低的代价实现象数据加密、防火墙等重要功能，是一种重要而特殊的驱动程序。

WDM驱动程序可分为三类：

- 1 高层驱动程序。例如FAT、NTFS、CDFS 等文件驱动程序，它们需要下层驱动的支持。
- 2 中级驱动程序。例如磁盘镜像、类驱动程序、微型驱动程序和过滤驱动程序等，它们穿插在高层和底层驱动程序之间。用户一般编写中级驱动程序。
- 3 底层驱动程序。例如硬件总线的控制器等。它调用硬件抽象层HAL函数和硬件打交道。

Windows 2000的设备驱动程序



Windows 2000的设备驱动程序

- 用户模式驱动程序

  Win32多媒体驱动
  支持MS-Dos应用程序的虚拟设备驱动程序VDD
  其他保护子系统的驱动程序

- 核心模式驱动程序

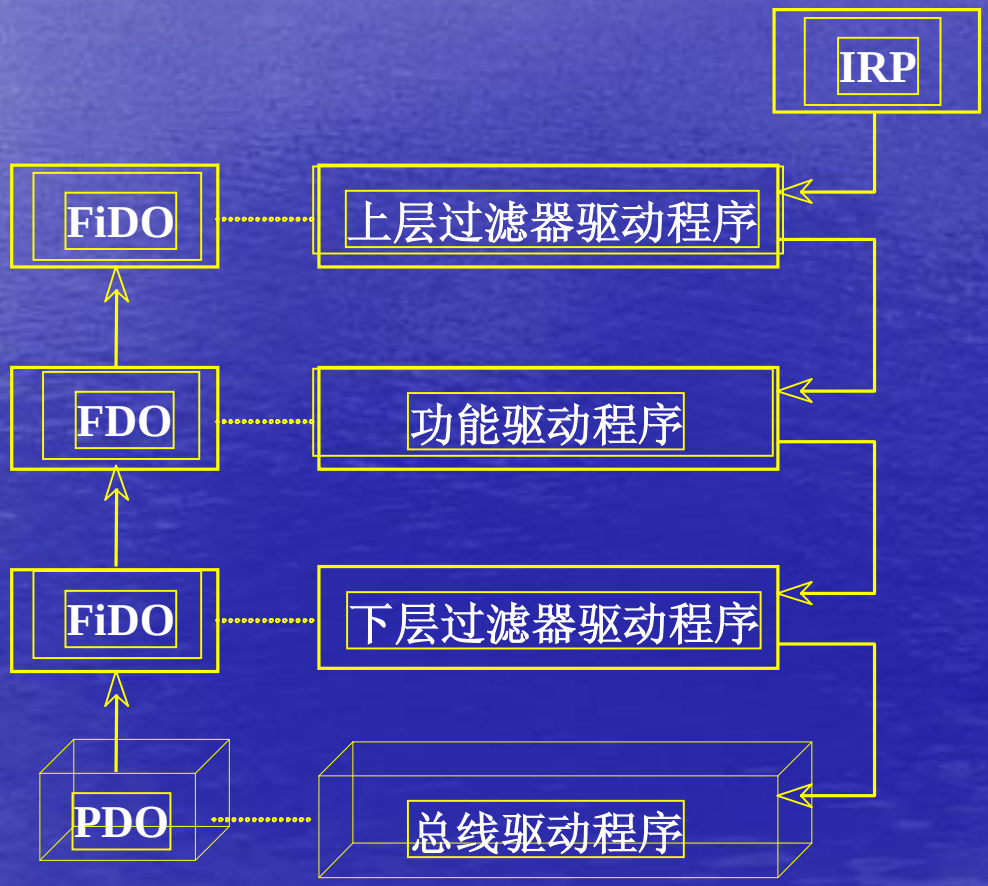
  PnP驱动程序
  WDM驱动程序
  小端口驱动程序
  文件系统驱动程序
  遗留设备驱动程序

核心模式驱动程序

- **虚拟设备驱动程序(VDD)**是一个用户模式部件，它可以使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，这对于那些直接对裸机硬件编程的应用程序特别有用
- **内核模式驱动程序**的分类包含许多子类。
- **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。准确地说，本书涉及的所有内容都是面向PnP驱动程序的。
- **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和Windows 2000间实现源代码级兼容(有限兼容，只限于两者都支持的功能)。WDM驱动程序还细分为**类驱动程序**(class driver)和**小类驱动程序**(minidriver)，类驱动程序管理属于已定义类的设备，小类驱动程序向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(包括多层次目录结构和命名文件概念)
- **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它驱动程序帮助。这种驱动程序主要包括Windows NT早期版本的驱动程序，它们可以不做修改地运行在Windows 2000中。

WDM的核心概念和数据结构

设备和驱动程序的分层



WDM设备驱动程序原理

- 驱动程序对象和设备对象以堆栈的结构处理I/ O 请求.设备对象堆栈最底层的叫作物理设备对象(PDO) ,与总线驱动程序对应;
- 栈中间与完成设备功能的驱动程序对应的是功能设备对象(FDO) ;
- 与其他各层驱动程序对应的设备对象叫作过滤器设备对象(FiDO) .

WDM设备驱动程序原理

- 功能驱动程序处理设备的I/ O 请求(IRP) ,如读写请求包和配置请求包;
- 总线驱动程序管理设备与系统的交互,负责具体的信号传输;而过滤器则监督或修改IRP 流,做一些额外的处理工作.
- 当用户态发一个I/ O请求, I/ O管理器将形成一个I/ O 请求包IRP. 在内核态, IRP 首先被送到最上层的驱动程序,然后逐渐过滤到下面,各层驱动程序对IRP 的处理取决于设备以及IRP 所携带的内容

WDM的核心概念和数据结构

- 功能驱动程序负责完成特定的功能，知道如何控制设备工作。它在驱动程序栈中位于总线驱动程序上面。功能驱动程序负责创建一个功能设备对象。在USB总线情形中，功能驱动程序必须使用总线驱动程序来访问它自己的设备
- 总线驱动程序负责列举设备，也就是说，它负责发现总线上的所有设备并检测设备何时添加到总线上或何时从总线上删除。总线驱动程序每发现一个设备就创建一个对应的物理设备对象。一些总线驱动程序只是简单地控制对总线的访问权。
- 在驱动程序栈中，可以插入各种类型的过滤驱动程序。对于总线上的所有设备，总线过滤驱动程序被添加在总线驱动之上；而对于一个特定类的所有功能的功能驱动程序添加类过滤驱动程序。

WDM的核心概念和数据结构

使用过滤器驱动程序可以实现以下几种主要功能：

- 1 允许修改已有驱动程序在某些方面的行为，而不必重写整个驱动程序。SCSI过滤器就是按这种方式写的。
- 2 可以更加容易地隐藏低级驱动程序的限制。
- 3 允许把诸如压缩、加密等新特征添加到设备上，而不必修改基本的设备驱动程序或者使用设备的程序。
- 4 允许增加或删除驱动程序可能总是不执行的昂贵行为.如性能监视

WDM的核心概念和数据结构

WDM驱动程序也是**分层的**，即不同层上的驱动程序有着不同的优先权，而Windows 9x下的VxD则没有此结构。

WDM还引入了

功能设备对象 FDO

（functional device object）

物理设备对象PDO

（physical device object）

一个PDO代表一个真实硬件

在驱动程序看来则是一个FDO 。

一个硬件只允许有一个PDO，但却可以拥有多个FDO

WDM的核心概念和数据结构

一个硬件只允许有一个PDO
一个硬件可以拥有多个FDO
驱动程序中我们不是直接操作硬件而是操作相应的PDO与FDO。
在Ring-3与Ring-0通讯方面，操作系统为每一个用户请求打包成一个IRP（IO Request Packet）结构，将其发送至驱动程序并通过识别IRP中的PDO来识别是发送给哪一个设备的。
在驱动程序的加载方面WDM既不靠驱动程序名称也不靠一个具有某种特殊意义的ID，而是依靠一个128位的GUID来识别驱动程序（Windows下许多东西都是靠此进行识别的）。

(1) 驱动程序对象 DRIVER_OBJECT

它是驱动程序的镜像，存有与该驱动程序有关的重要信息(如调度例程表)。它在驱动程序装入时被创建，使用 Unload 例程可以将它从系统中删除。

(2) 设备对象 DEVICE_OBJECT

它代表一个逻辑设备或物理设备，在驱动程序的各个例程之间保存和传递设备的特征和状态信息，如设备扩展、设备名称以及标志变量等。

(3) I/O 请求包 IRP

设备管理器及设备驱动程序使用 IRP 来管理具体的 I/O 操作，一个 IRP 代表了一次具体的 I/O 操作，并贯穿在一次 I/O 操作的整个过程，以保存和跟踪本次 I/O 操作的状态和进展。

WDM的核心概念和数据结构

驱动程序对象（driver object）

DRIVER_OBJECT数据结构

I/O管理器使用驱动程序对象来代表每个设备驱动程序，驱动程序对象描述了驱动程序载入到物理内存的什么地方，驱动程序的大小和它的主要入口点。

WDM的核心概念和数据结构

设备对象（DeviceObject）

指向一个设备对象链表，每个设备对象代表一个设备

用户态使用Win32 CreateFile访问驱动程序，dwShareMode为0时来请求独占内核对象在设备对象DEVICE_OBJECT结构中存储设备的信息，对于与设备的每个交互，相关的DEVICE_OBJECT被传递给驱动的回调例程。

设备对象DriverObject

指向与该设备对象相关的驱动程序对象

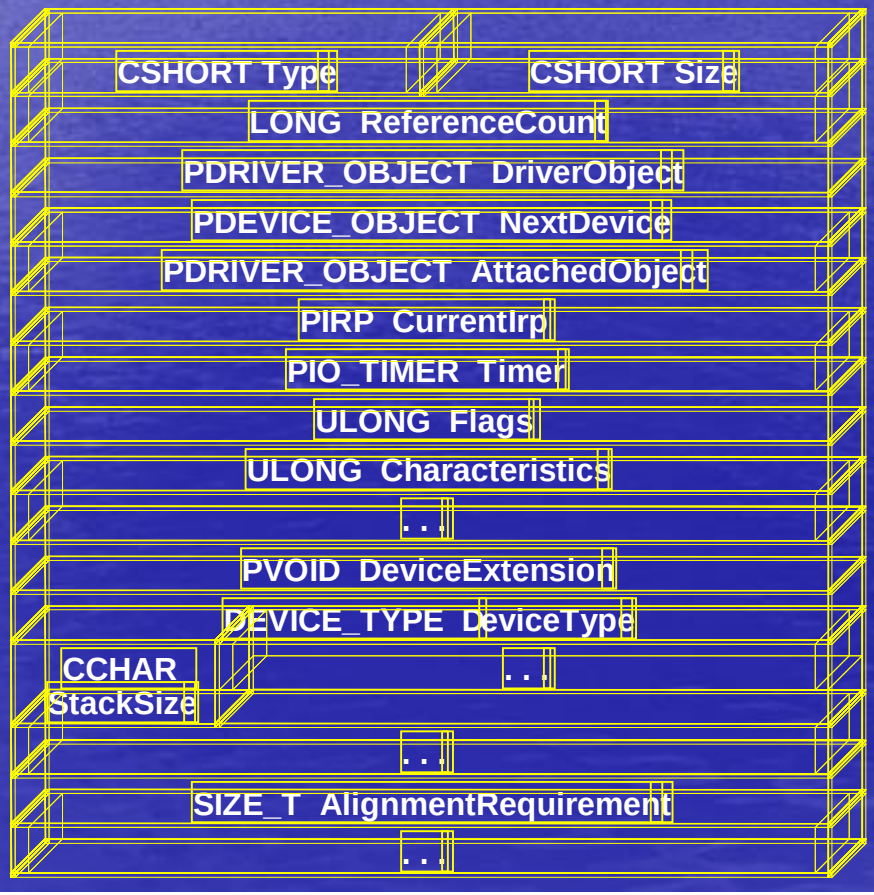
NextObject

指向属于同一个驱动程序的下一个设备对象

WDM的核心概念和数据结构

设备对象（device object）

代表能够成为I/O操作目标的物理设备或逻辑设备，它以DEVICE_OBJECT结构来描述



WDM的核心概念和数据结构

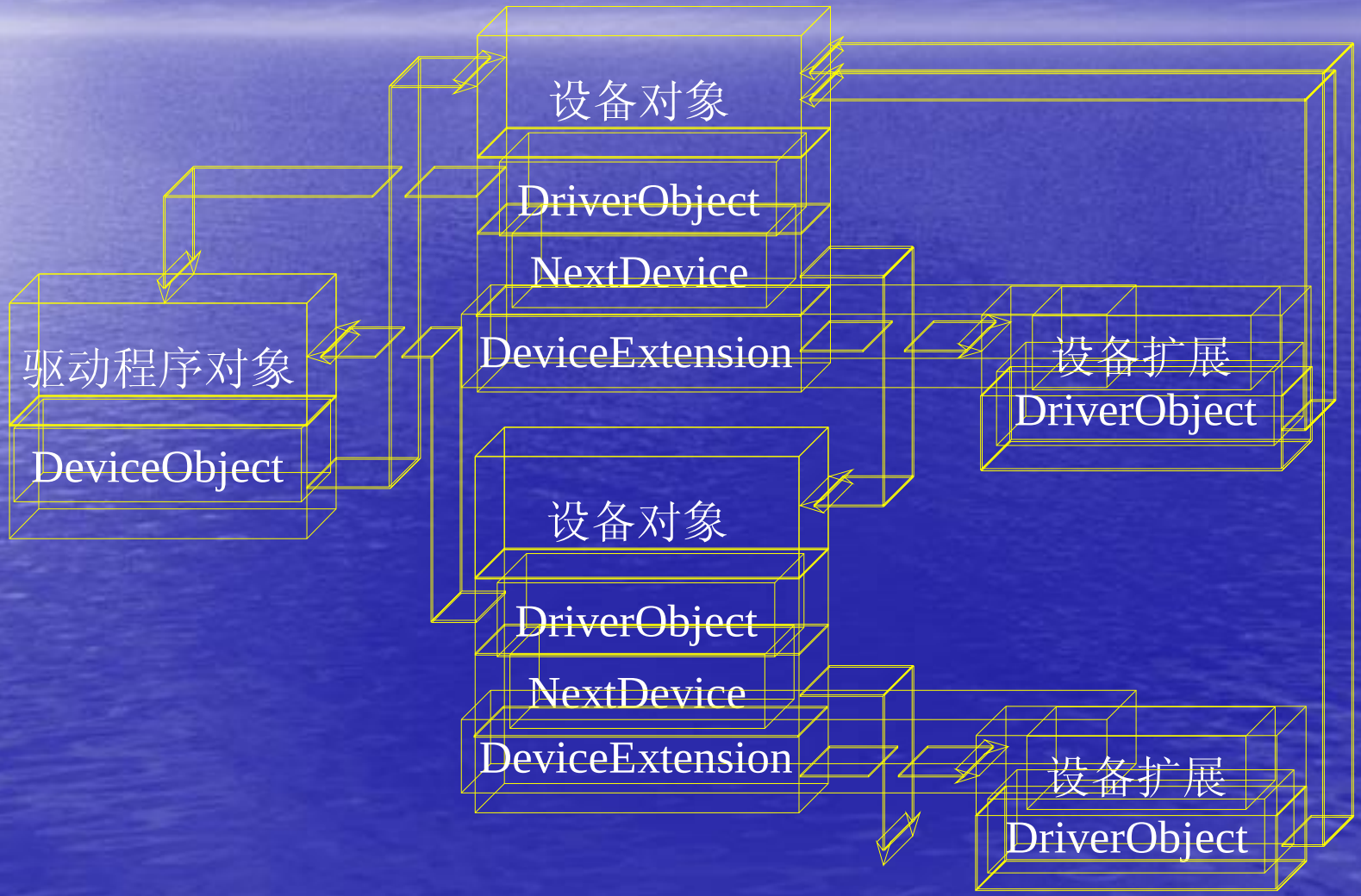
设备对象DeviceExtension

指向一个用户定义的数据结构，驱动程序可以使用该结构保存每个设备实例的信息——设备扩展（device extension），或设备扩展对象

```
{           DebugPrint("AddDevice");
status = IoCreateDevice (DriverObject,创建设备
                sizeof(WDM1_DEVICE_EXTENSION),
                    NULL,     // No Name
                FILE_DEVICE_UNKNOWN,
                    0,
            FALSE,   // Not exclusive，TRUE为独占
                &fdo返回的新设备对象);
        if( !NT_SUCCESS(status))
                    return status;
IoAttachDeviceToDeviceStack(fdo,pdo);与设备栈挂接
```

WDM的核心概念和数据结构

驱动程序对象、设备对象、设备扩展之间的关系

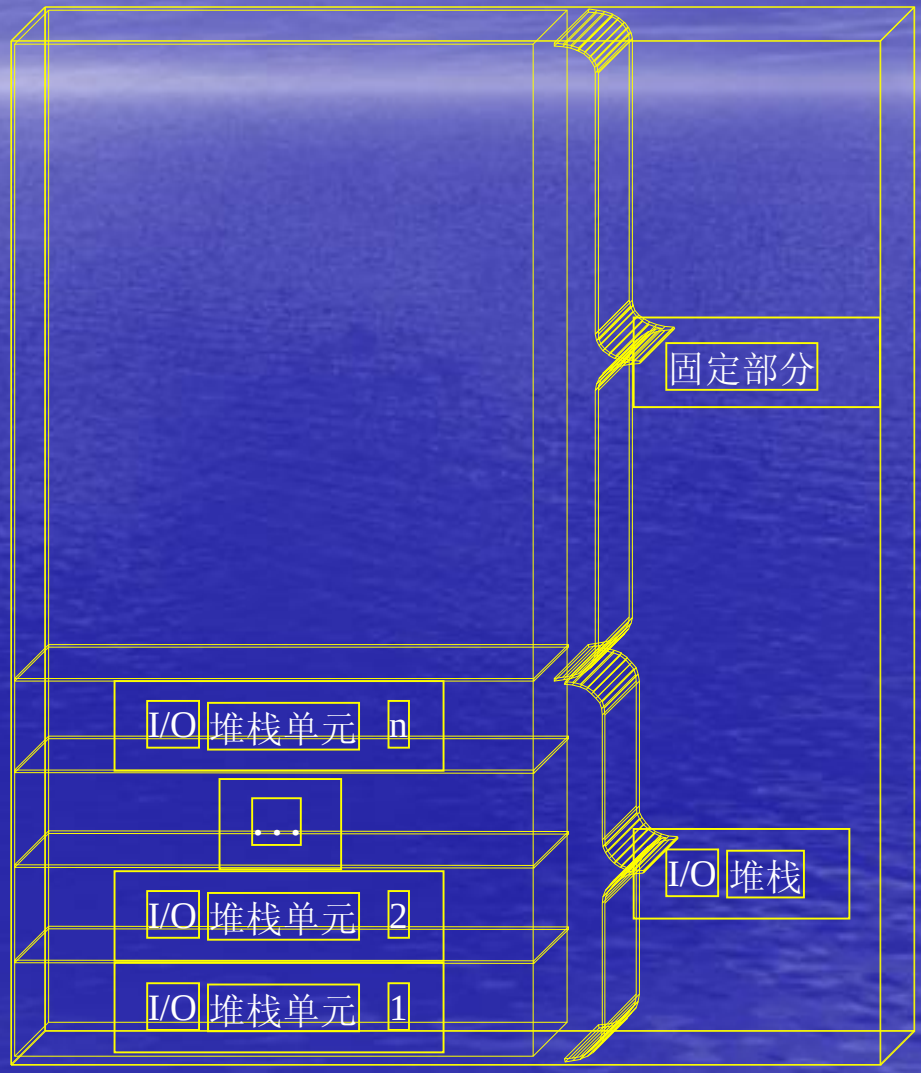


WDM的核心概念和数据结构

I/O请求包

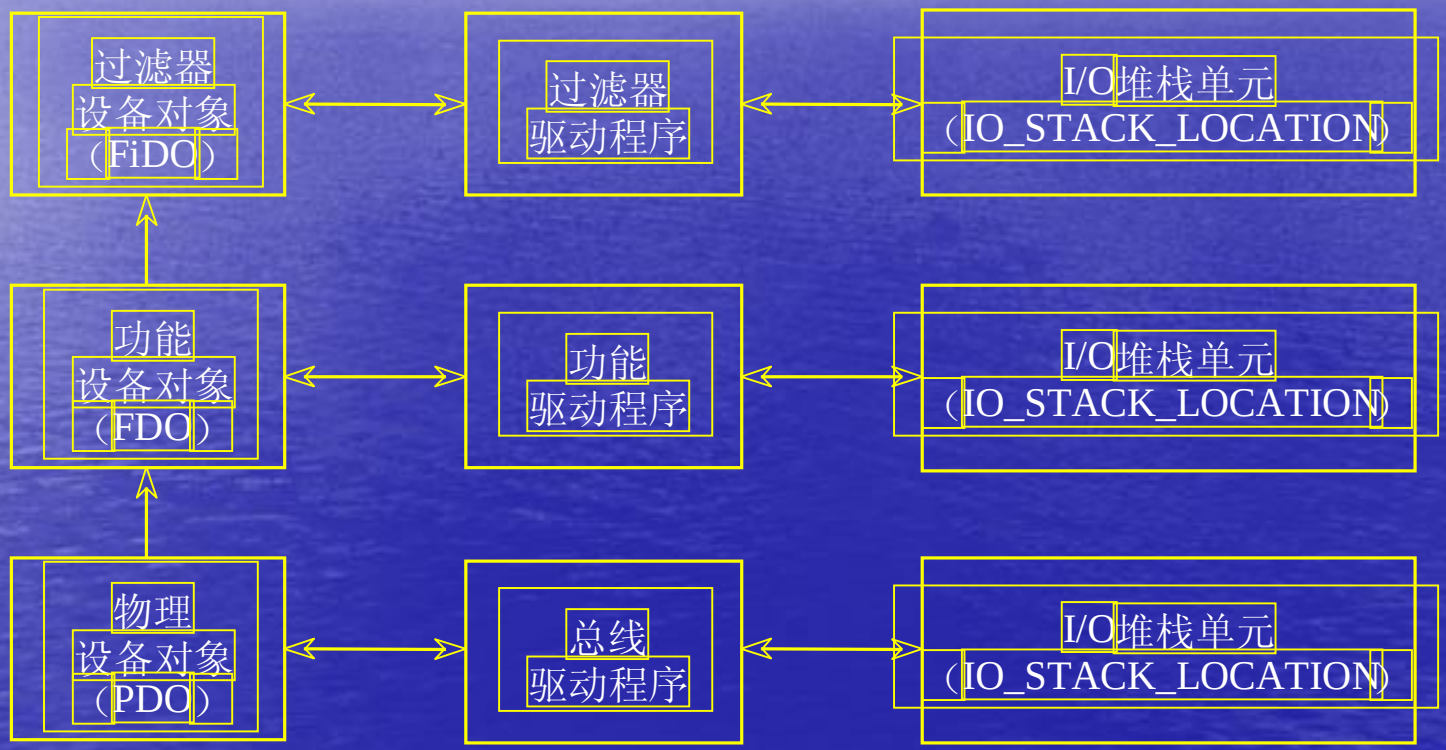
IRP是I/O管理器在响应一个I/O请求时从非分页系统内存中分配的一块可变大小的数据结构内存

I/O请求包IRP是驱动程序操作的中心，IRP是一个内核对象，它是预先定义好的数据结构，带有一组对它进行操作的I/O管理器例程，I/O管理器接受一个I/O请求，然后将它传送到合适的驱动程序栈中的最高驱动程序之前，分配并处始化一个IRP，每个I/O请求有主功能代码



WDM的核心概念和数据结构

I/O请求包



驱动程序和I/O堆栈之间的平行关系

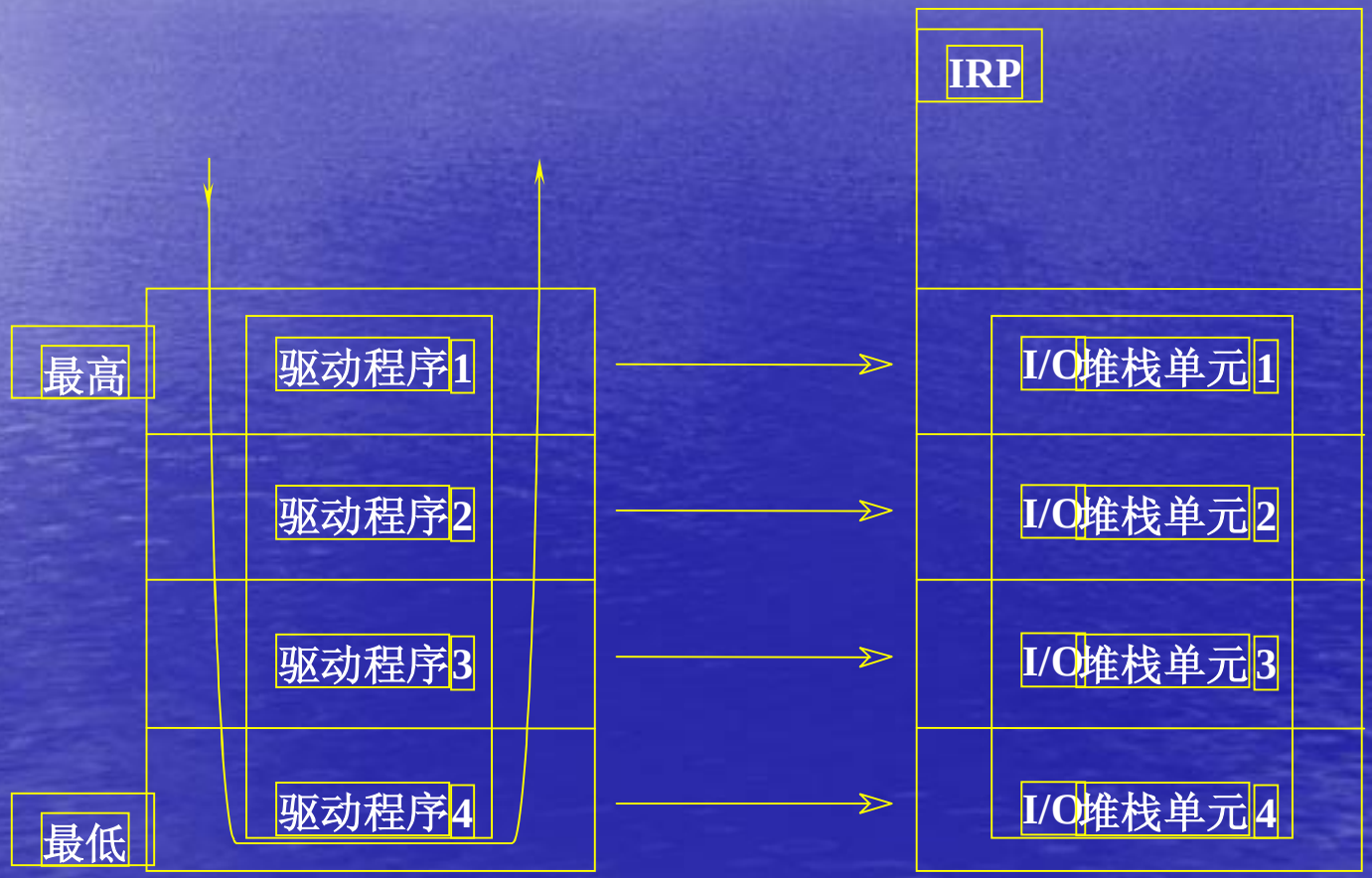
WDM的核心概念和数据结构

I/O功能代码：常用的IRP主功能代码

功能代码	说明	对应的 Win32 API 函数
IRP_MJ_CREATE	打开设备	CreateFile
IRP_MJ_CLEANUP	在关闭设备时，取消挂起的请求	CloseHandle
IRP_MJ_CLOSE	关闭设备	CloseHandle
IRP_MJ_READ	从设备获得数据	ReadFile
IRP_MJ_WRITE	向设备发送数据	WriteFile
IRP_MJ_DEVICE_CONTROL	对用户模式或内核模式客户程序可用的控制操作	DeviceIoControl
IRP_MJ_INTERNAL_DEVICE_CONTROL	只对内核模式客户程序可用的控制操作	没有对应的 Win32 API
IRP_MJ_QUERY_INFORMATION	得到文件的长度	GetFileLength
IRP_MJ_SET_INFORMATION	设置文件的长度	SetFileLength
IRP_MJ_FLUSH_BUFFERS	写输出缓冲区或丢弃输入缓冲区	FlushFileBuffers FlushConsoleInputBuffer PurgeComm
IRP_MJ_SHUTDOWN	系统关闭	InitialSystemShutdown

WDM的核心概念和数据结构

IRP的处理



WDM的核心概念和数据结构

用户对设备的访问过程

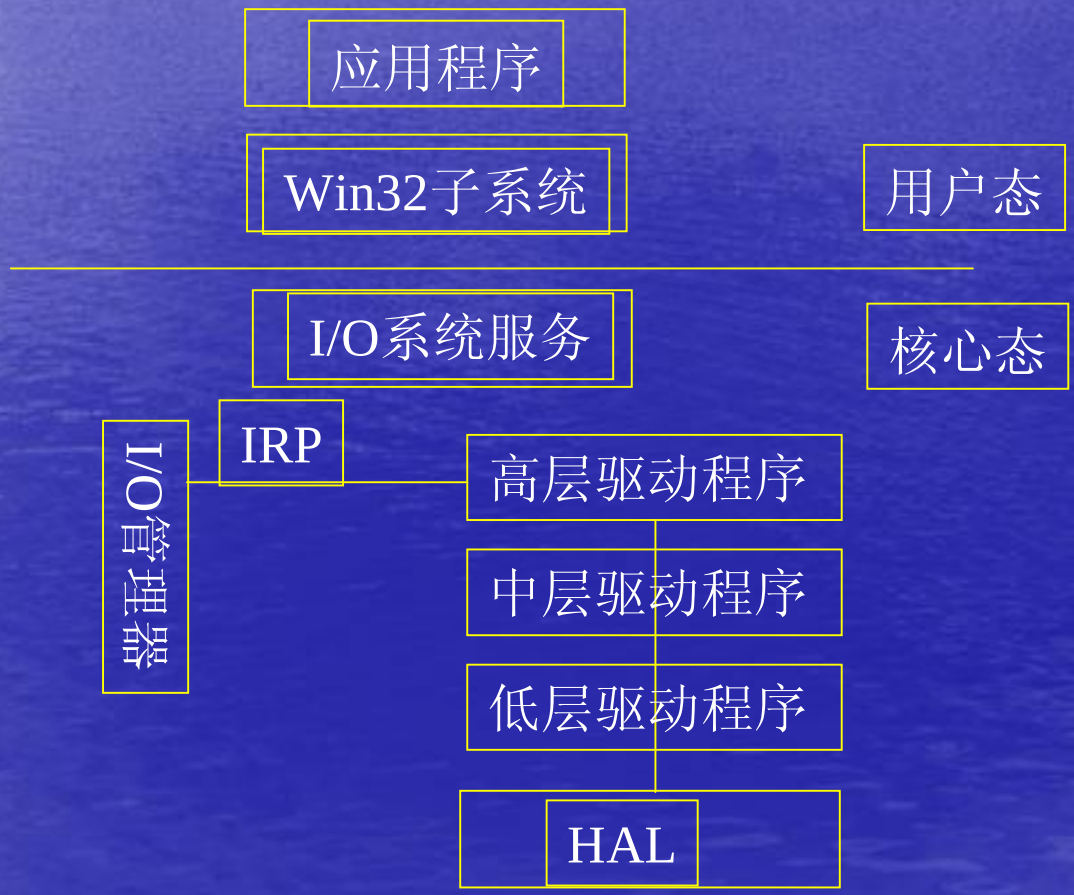
Windows中对设备的访问分为用户态和核心态两种方式：

用户态通过调用Win32 API函数如ReadFile、WriteFile等访问设备，它不能直接控制硬件

核心态通过发送I/O请求包IRP来运行驱动程序实现对设备的控制

WDM的核心概念和数据结构

用户对设备的访问过程



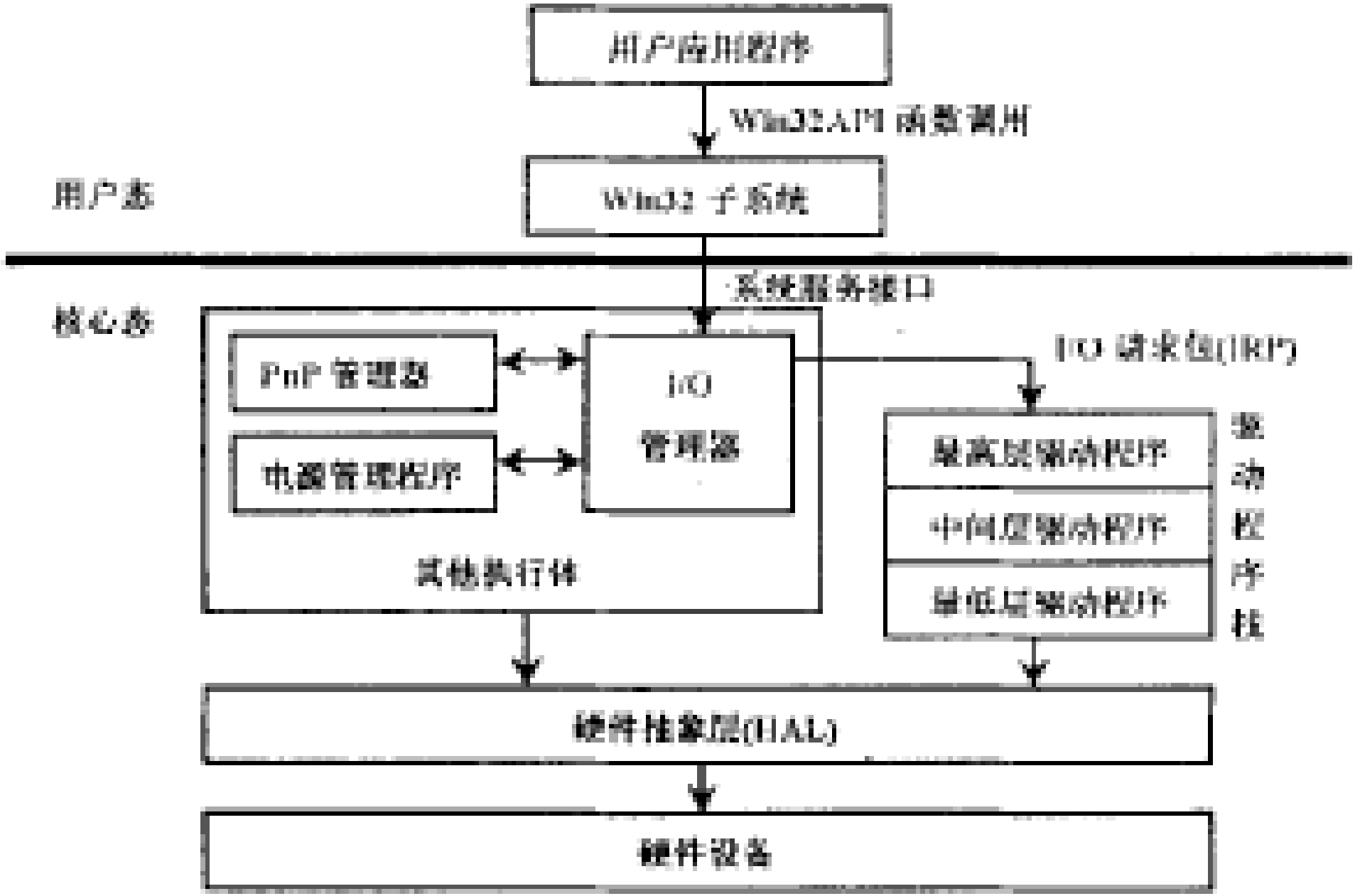


图 3　设备驱动程序的调用

- 用户态应用程序对Windows子系统进行win32 API调用，这个调用由系统服务接口作用到I/O管理器
- I/O管理器进行必要的参数匹配和操作安全性检查，然后由这个请求构造出合适的IRP(IO请求包),并把此IRP传给驱动程序。
- 驱动程序直接执行这个请求包，并与硬件打交道，从而完成I/O请求工作，最后由I/O管理器将执行结果返回给用户态程序。
- 在WDM体系结构中，大部分实行分层处理。即在图中“设备驱动“这部分，分成了若干层，典型地分成高层驱动程序、中间层驱动程序、底层驱动程序。每层驱动再把I/O请求划分成更简单的请求，以传给更下层的驱动执行。

WDM驱动程序的结构

必须的驱动程序例程

处理请求队列需要包含StarIo

如果设备产生中断需要包含中断和DPC例程

DMA操作需要包含AdapterControl例程

可选的IRP分发例程

WDM驱动程序的结构

DriverEntry例程

没有作为入口的main函数或WinMain函数，它向操作系统显露一个名称为DriverEntry的函数，在启动驱动程序的时候，操作系统将调用这个入口。

驱动程序可以被多个类似的硬件使用，但驱动程序的某些全局初始化操作只能在第一次被装入时执行一次，DriverEntry例程就是用于这个目的

DriverEntry例程的主要工作是把各种函数指针填入驱动程序对象，这些指针为操作系统指明了驱动程序容器中各种子例程的位置

- NTSTATUS DriverEntry(IN PDRIVER OBJ ECT DriverObject ,
- IN PUNICODE STRING RegistryPath)
- {
- DriverObject - > DriverUnload = DriverUnload ;
- DriverObject - > DriverExtension - > AddDevice = AddDevice ;
- DriverObject - > DriverStartIo = StartIo ;
- DriverObject - >MajorFunction[IRP MJ PNP] = DispatchP2
- np ;
- DriverObject - > MajorFunction [IRP MJ POWER] = Dis2
- patchPower ;
- DriverObject - > MajorFunction [IRP MJ SYSTEM CON2
- TROL] = DispatchWmi ;
- DriverObject - > MajorFunction [IRP MJ CREATE] = Dis2
- patchCreate ;
- DriverObject - > MajorFunction [IRP MJ CLOSE] = Dis2
- patchClose ;
- DriverObject - > MajorFunction [IRP MJ READ] = Dis2
- patchRead ;
- DriverObject - > MajorFunction [IRP MJ WRITE] = Dis2
- patchWrite ;
- DriverObject - > MajorFunction [IRP MJ DEVICE CON2
- TROL] = DispatchDeviceControl ;
- ...
- return STATUS SUCCESS;

WDM驱动程序的结构

- IRP 处理例程
- 当调用Win32 API CreateFile，CloseHandle 及DeviceIoControl 时，操作系统获得IRP 的主功能码(IRPMJ CREATE ,IRP MJ CLOSE ,IRP MJ DEVICE CONTROL) ,并最终转化为对驱动程序主功能码所对应的子例程MyParDeviceDispatch 的调用.

WDM驱动程序的结构

AddDevice例程

AddDevice函数的基本职责是创建一个设备对象并把它连接到以pdo为栈底的设备堆栈中，主要步骤如下：

(1) 过滤器调用 IoCreatDevice 建立此目标设备的一个无名过滤器设备对象。

(2) 调用 IoAttachDeviceToDeviceStack 把自己堆叠在低级驱动程序之上，并获取目标设备对象的指针。

(3) 把目标设备对象的地址保存在过滤器设备对象的设备扩展中。过滤器驱动程序的其他部分使用此指针调用目标驱动程序。

(4) 把 DeviceType 和 Characteristics 字段从目标设备对象复制到过滤器设备对象。此外，它还复制 Flags 字段的 DO_DIRECT_IO、DO_BUFFERED_IO、DO_POWER_INRUSH 和 DO_POWER_PAGABLE 位。这就保证了过滤器看起来是相同的，并且具有与目标驱动程序相同的策略。

WDM驱动程序的结构

- **DispatchPnp** 例程
- 提供完全的PnP 支持是Windows 2000/ XP 的设计目标之一。WDM 驱动程序支持特定的即插即用IRP(IRP MJ PNP)

  其函数原型为:
- NTSTATUS DispatchPnp(PDEVICE OBJ ECTfdo ,PIRP Irp) ;

WDM驱动程序的结构

- **DispatchPower 例程**

  现有的总线都支持电源管理。为了适应“热插拔”和其他电源管理(如节省移动终端的能耗) 要,WDM驱动程序也必须支持电源管理。微软公司开发的OnNow 规范已经对各种电源管理方法进行了标准化。这个例程的函数原型为:

- NTSTATUS DispatchPower (IN PDEVICE OBJECT fdo , INPIRP Irp) ;

WDM驱动程序的结构

- **DispatchWmi 例程**
- WDM驱动程序模型包括标准化收集、存储和报告设备测试数据的功能———— Windows 管理和设备测试(Windows Management and Instrumentation ,WMI)

WDM驱动程序的结构

其他必须的例程

DispatchPnp例程

DispatchPower例程

DispatchWmi例程

其他可选的例程

Windows应用程序与设备驱动程序打交道主要是通过CreateFile、 ReadFile、WriteFile 和DeviceIoControl等Win32 API来进行的，这些API对应着驱动程序的一些分发例程

驱动程序中除了DriverEntry例程必须以DriverEntry命名以外，其他例程都可以使用程序员自定义的名字，并且都要由DriverEntry例程向系统注册

- 这些IRP 的分发例程都有相同的函数原型,均需
- 传递一个指向设备对象的指针:
- NTSTATUS DispatchCreate (IN PDEVICE OBJ ECT fdo , IN
- PIRP Irp) ;
- NTSTATUS DispatchClose(IN PDEVICE OBJ ECTfdo ,IN PIRP
- Irp) ;
- NTSTATUS DispatchDeviceControl(IN PDEVICE OBJ ECTfdo ,
- IN PIRP Irp) ;
- NTSTATUS DispatchRead(IN PDEVICE OBJ ECTfdo ,IN PIRP
- Irp) ;
- NTSTATUS DispatchWrite(IN PDEVICE OBJ ECTfdo ,IN PIRP
- Irp) ;

WDM驱动程序的编程

- WDMdriver的源代码组成
- 初始化、创建、删除设备
- 即插即用处理与电源管理
- WMI
- 分发例程
- 驱动程序的编译链接

驱动程序的安装

- WDM设备驱动程序的安装由一个以INF 为扩展名的文本文件控制。
- 一个INF 文件是一个被划分为节(Section) 的文本文件,每节由方括号([ ]) 内的标识符表示下面的各项分别控制哪些安装操作,或者表示链接或列举其他节。
- 通过INF 文件,可获得哪一个文件需要复制到用户的硬盘上,应该添加和修改哪一个注册表项等信息,可自动完成驱动程序的安装,或者在对话框支持下通过手工完成对驱动程序的安装