2021/04/26_Windows32位内核_第4课_驱动框架的编写以及0环3环间的通信

笔记本: Windows32位内核
创建时间: 2021/4/26 星期一 15:33
作者: ileemi

驱动框架的介绍

内核框架的设计

操作系统为硬件厂家提供了开发驱动的框架。框架为了支持以后硬件的发展，需要对硬件进行一定的抽象。

NT驱动框架抽象:
硬件操作（I/O） --> 抽象 --> 文件

所有操作系统一般对硬件操作（I/O）的抽象为文件操作。提供打开、关闭、设置文件指针、读取、写入、控制等公共的方法（提供接口函数）。

驱动对象，提供派遣函数（是一个函数指针数组，大小为28），在驱动中注册派遣函数就需要写出对应的回调处理函数。

驱动框架的编写

- 注册派遣函数
- 创建设备
- 创建符号链接

函数回调对应的声明:

```
NTSTATUS DRIVER_DISPATCH (
  _In_ struct _DEVICE_OBJECT *DeviceObject, // 被操作的设备对象
  _Inout_ struct _IRP *Irp // 保存传递操作的数据信息， I/O Request
Packet
);
// 分层的驱动设计（方便不同接口的硬件出现冗余代码）
// 为了增加维护性，硬件驱动才会有分层的需求
// _IRP 结构体中的数据共享
// 通过内核API IoGetCurrentIrpStackLocation 向I/O管理器获取irp堆栈的数据
PIO_STACK_LOCATION pIrpStack = IoGetCurrentIrpStackLocation(Irp);
```

每个驱动都有自己的Irp堆栈，由I/O管理器创建。过滤驱动程序可以修改已有驱动的功能，也可以对数据进行过滤加密。

所有的驱动对象以及设备对象都是在内核空间中进行申请的。

创建的的驱动设备在3环中没有权限打开，只能在内核中才有权限打开。如果需要在3环中访问创建的驱动程序，就需要向3环提供一个符号链接（为设备创建一个符号链接）。符号链接用来指明文件位置。

内核中操作路径的写法：

- 通过磁盘的设备名+文件名："\\??\\hardiskvolume1\\1.txt"
- 通过符号链接："\\??\c\\1.txt"

代码示例

注册派遣函数：

```
// 注册派遣函数 -- 向操作系统注册操作
// IRP_MJ_CLEANUP -- 清理
DriverObject->MajorFunction[IRP_MJ_CREATE] = DispatchCreate;
DriverObject->MajorFunction[IRP_MJ_CLOSE] = DispatchClose;
DriverObject->MajorFunction[IRP_MJ_READ] = DispatchRead;
DriverObject->MajorFunction[IRP_MJ_WRITE] = DispatchWrite;
DriverObject->MajorFunction[IRP_MJ_DEVICE_CONTROL] = DispatchControl;
```

创建设备：

```
// 初始化设备对象名称结构体
RtlInitUnicodeString(&ustrDevName, L"\\Device\\DrvTest123");
// 向 I/O 管理器创建一个设备，和驱动对象进行绑定
Status = IoCreateDevice(DriverObject, // 驱动对象
    0, // 设备的扩展
    &ustrDevName, // 给空，操作系统会动态分配一串数字作为设备对象名字
    FILE_DEVICE_UNKNOWN, // 未知设备
    0, // 设备属性
    FALSE, // 是否允许多个进程操作（驱动是否公开）
    &pDevObj); // 保存创建的设备对象，一个驱动可以创建多个设备

if (!NT_SUCCESS(Status)) {
    KdPrint(("[DrvTest] IoCreateDevice Status:%p\n", Status));
    return Status;
}
```

创建符号链接：

```
RtlInitUnicodeString(&ustrSymbolink, L"\\??\\DrvTest123");
Status = IoCreateSymbolicLink(&ustrSymbolink, &ustrDevName);
```

-1 环 -- VT 模式