# 内核中执行代码后需要将结果动态显示给应用层的用户，DeviceIoControl 是直接发送控制代码到指定的设备驱动程序，使相应的移动设备以执行相应的操作的函数，如下代码是一个经典的驱动开发模板框 架，在开发经典驱动时会用到的一个通用案例。

驱动程序开发通用模板代码如下:

#include <ntifs.h> #include <windef.h>

// 控制器

#define IOCTL\_IO\_LyShark CTL\_CODE(FILE\_DEVICE\_UNKNOWN,0x800,METHOD\_BUFFERED,FILE\_ANY\_ACCESS)

// 卸载驱动执行

VOID UnDriver(PDRIVER\_OBJECT pDriver)

{

PDEVICE\_OBJECT pDev; // 用来取得要删除设

备对象

UNICODE\_STRING SymLinkName; // 局部变量

symLinkName

pDev = pDriver->DeviceObject;

IoDeleteDevice(pDev); // 调用

IoDeleteDevice用于删除设备

RtlInitUnicodeString(&SymLinkName, L"\\??\\LySharkDriver"); // 初始化字符串将symLinkName定义成需要删除的符号链接名称

IoDeleteSymbolicLink(&SymLinkName); // 调用

IoDeleteSymbolicLink删除符号链接

DbgPrint("驱动卸载完毕...");

}

// 创建设备连接

NTSTATUS CreateDriverObject(IN PDRIVER\_OBJECT pDriver)

{

NTSTATUS Status; PDEVICE\_OBJECT pDevObj;

UNICODE\_STRING DriverName; UNICODE\_STRING SymLinkName;

// 创建设备名称字符串

RtlInitUnicodeString(&DriverName, L"\\Device\\LySharkDriver");

Status = IoCreateDevice(pDriver, 0, &DriverName, FILE\_DEVICE\_UNKNOWN, 0, TRUE, &pDevObj);

// 指定通信方式为缓冲区

pDevObj->Flags |= DO\_BUFFERED\_IO;

// 创建符号链接

RtlInitUnicodeString(&SymLinkName, L"\\??\\LySharkDriver"); Status = IoCreateSymbolicLink(&SymLinkName, &DriverName); return STATUS\_SUCCESS;

}

// 创建回调函数

NTSTATUS DispatchCreate(PDEVICE\_OBJECT pDevObj, PIRP pIrp)

{

pIrp->IoStatus.Status = STATUS\_SUCCESS; // 返回成功

DbgPrint(" 派 遣 函 数 IRP\_MJ\_CREATE 执 行 \n"); IoCompleteRequest(pIrp, IO\_NO\_INCREMENT); // 指示完成此IRP return STATUS\_SUCCESS; // 返回成功

}

// 关闭回调函数

NTSTATUS DispatchClose(PDEVICE\_OBJECT pDevObj, PIRP pIrp)

{

pIrp->IoStatus.Status = STATUS\_SUCCESS; // 返回成功

DbgPrint(" 派 遣 函 数 IRP\_MJ\_CLOSE 执 行 \n"); IoCompleteRequest(pIrp, IO\_NO\_INCREMENT); // 指示完成此IRP return STATUS\_SUCCESS; // 返回成功

}

// 主控制器,用于判断R3发送的控制信号

// lyshark.com

NTSTATUS DispatchIoctl(PDEVICE\_OBJECT pDevObj, PIRP pIrp)

{

NTSTATUS status = STATUS\_INVALID\_DEVICE\_REQUEST;

PIO\_STACK\_LOCATION pIrpStack; ULONG uIoControlCode;

PVOID pIoBuffer; ULONG uInSize; ULONG uOutSize;

// 获得IRP里的关键数据

pIrpStack = IoGetCurrentIrpStackLocation(pIrp);

// 获取控制码

uIoControlCode = pIrpStack->Parameters.DeviceIoControl.IoControlCode;

// 输入和输出的缓冲区（DeviceIoControl的InBuffer和OutBuffer都是它）

pIoBuffer = pIrp->AssociatedIrp.SystemBuffer;

// EXE发送传入数据的BUFFER长度（DeviceIoControl的nInBufferSize） uInSize = pIrpStack->Parameters.DeviceIoControl.InputBufferLength;

// EXE接收传出数据的BUFFER长度（DeviceIoControl的nOutBufferSize） uOutSize = pIrpStack->Parameters.DeviceIoControl.OutputBufferLength;

// 对不同控制信号的处理流程

switch (uIoControlCode)

{

// 接收或发送

case IOCTL\_IO\_LyShark:

{

DWORD dw = 0;

// 得到输入参数

memcpy(&dw, pIoBuffer, sizeof(DWORD));

DbgPrint("[+] hello lyshark \n");

// 对输入参数进行处理



dw++;

// 设置输出参数

memcpy(pIoBuffer, &dw, sizeof(DWORD));

// 返回通信状态

status = STATUS\_SUCCESS; break;

}

pIrp->IoStatus.Status = status;

pIrp->IoStatus.Information = uOutSize; IoCompleteRequest(pIrp, IO\_NO\_INCREMENT); return status;

}

// 设定DeviceIoControl的 lpBytesReturned的值（如果通信失败则返回0长度） if (status == STATUS\_SUCCESS)

pIrp->IoStatus.Information = uOutSize; else

pIrp->IoStatus.Information = 0;

// 设定DeviceIoControl的返回值是成功还是失败

pIrp->IoStatus.Status = status; IoCompleteRequest(pIrp, IO\_NO\_INCREMENT); return status;

}

// 入口函数

NTSTATUS DriverEntry(PDRIVER\_OBJECT pDriver, PUNICODE\_STRING RegistryPath)

{

// 调用创建设备

CreateDriverObject(pDriver);

pDriver->DriverUnload = UnDriver; // 卸载函数pDriver->MajorFunction[IRP\_MJ\_CREATE] = DispatchCreate; // 创建派遣函数pDriver->MajorFunction[IRP\_MJ\_CLOSE] = DispatchClose; // 关闭派遣函数pDriver->MajorFunction[IRP\_MJ\_DEVICE\_CONTROL] = DispatchIoctl; // 分发函数

DbgPrint("By:LyShark ...");

return STATUS\_SUCCESS;

}

# 应用层通用测试模板代码如下:

#include <iostream> #include <Windows.h> #include <winioctl.h>

#define IOCTL\_IO\_LyShark CTL\_CODE(FILE\_DEVICE\_UNKNOWN,0x800,METHOD\_BUFFERED,FILE\_ANY\_ACCESS)



int main(int argc, char argv[])

{

HANDLE hDevice = CreateFileA("\\\\.\\LySharkDriver", GENERIC\_READ | GENERIC\_WRITE, 0,

NULL, OPEN\_EXISTING, FILE\_ATTRIBUTE\_NORMAL, NULL);

if (hDevice == INVALID\_HANDLE\_VALUE)

{

CloseHandle(hDevice); return 0;

}

// 发送控制信号

// input = 发送数据 output = 接受数据 ref\_len = 数据长度

DWORD input = 100, output = 0, ref\_len = 0;

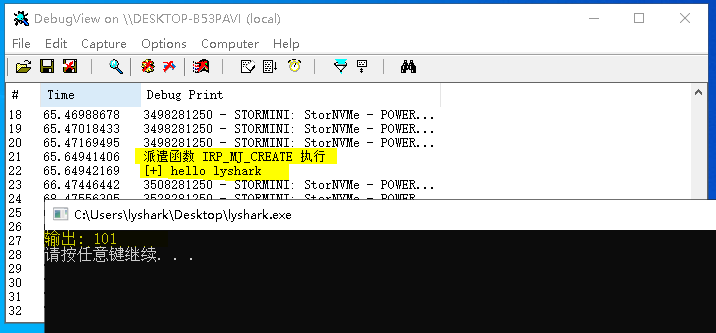
DeviceIoControl(hDevice, IOCTL\_IO\_LyShark, &input, sizeof(input), &output, sizeof(output), &ref\_len, 0);

printf("输出: %d \n", output);

system("pause"); CloseHandle(hDevice); return 0;

}

输出效果如下：



本书作者： 王瑞 (LyShark)

作者邮箱： m [e@lyshark.com](mailto:e@lyshark.com)

作者博客： h ttps://lyshark.cnblogs.com

团队首页： w ww.lyshark.com