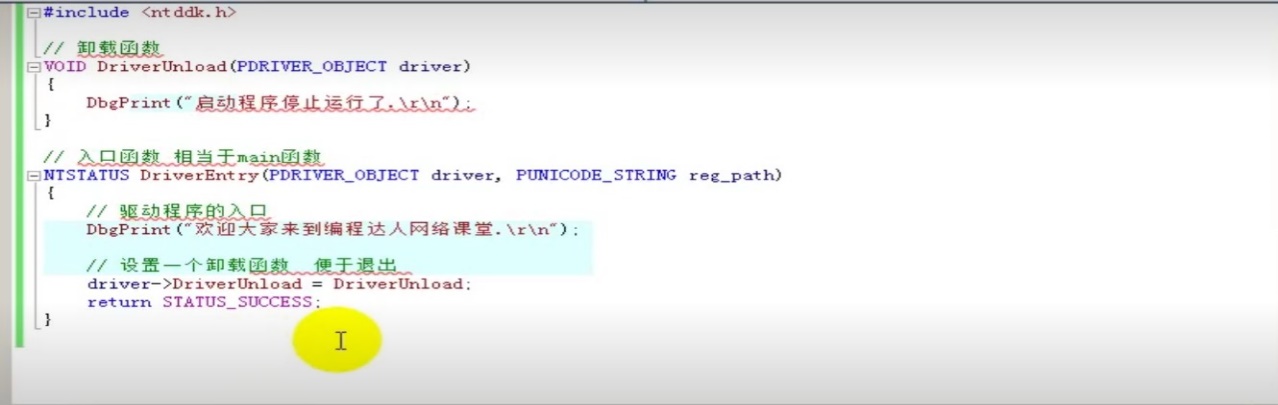
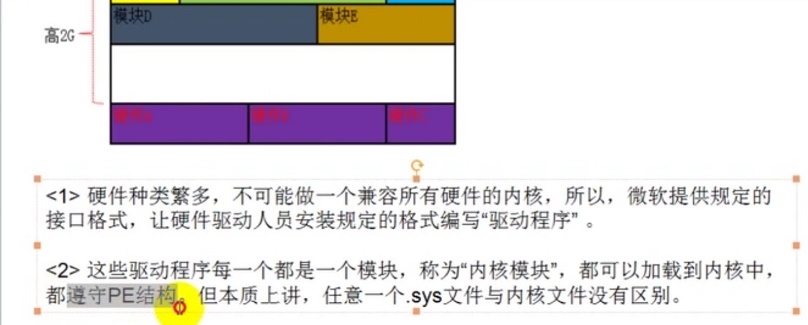
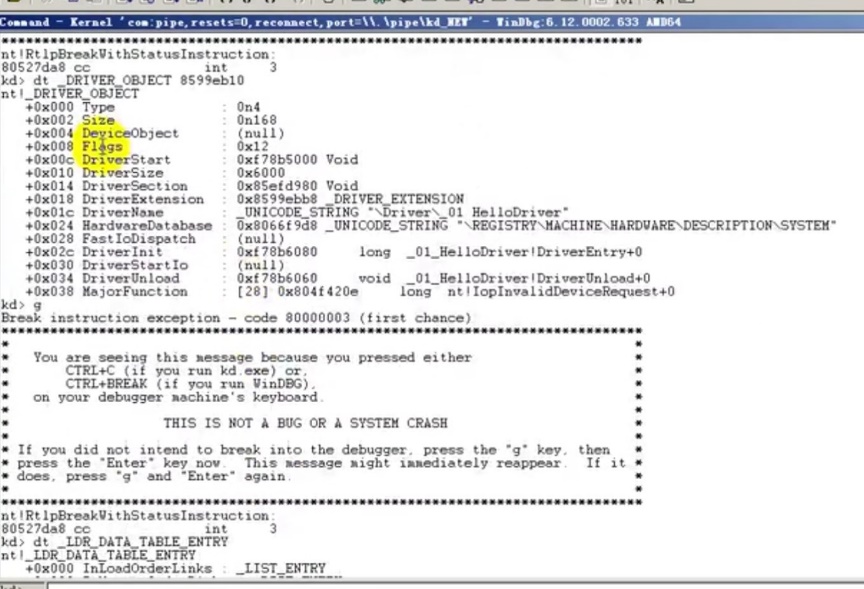


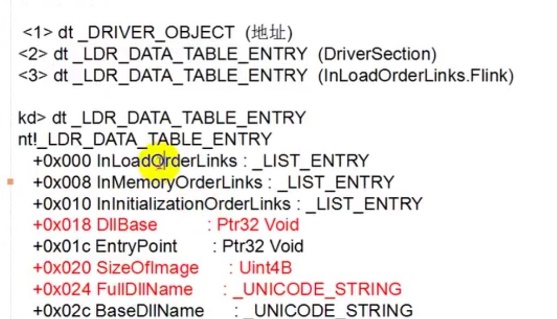
驱动程序的基本结构 ：





Drive\_object结构体的详细信息：





Drive\_object结构体描述了当前模块的重要信息。是一个驱动对象。

Drive\_object结构体中存在一个项DriverSection这个是一个链表，指向内核中现在虽有存在的模块。

Drive\_object结构中还有一个MajorFunction数组，该数组中就是对应的派遣函数。

创建设备通过IoCreateDevice函数来创建设备结构体指针，该函数返回一个结构体device\_object的指针pDeviceObj

任何设备必须要属于一个驱动对象🡺Drive\_object，也就是IoCreateDevice的第一参数。

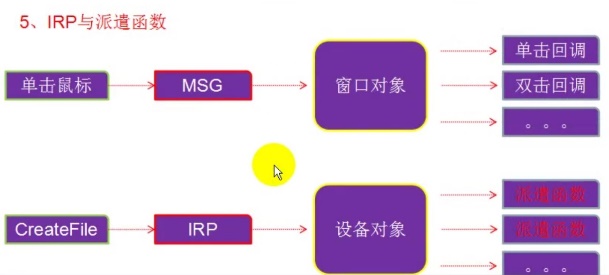


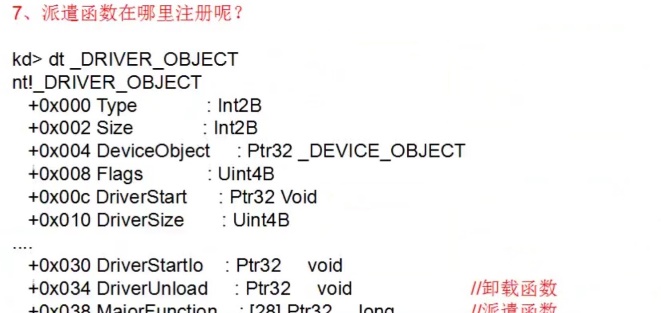
该函数的最后的参数就是返回值，设备对象指针。

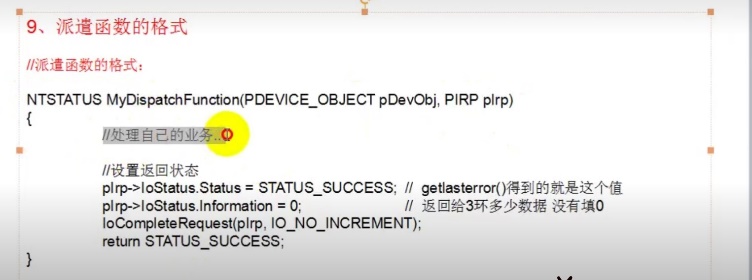
有了设备才可以接收消息，消息是IRP格式的，就是一个结构体。【把IRP理解成一个消息就行】



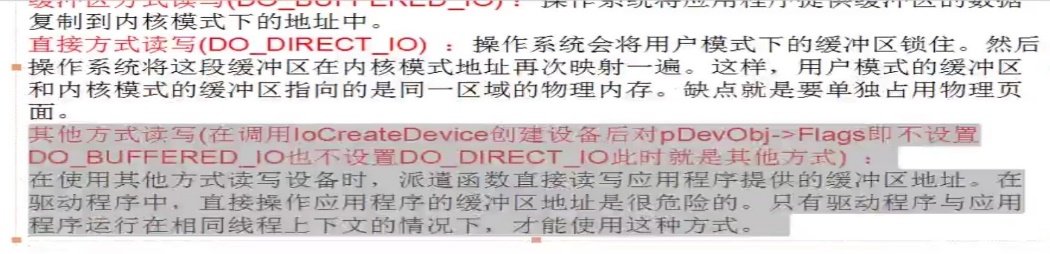


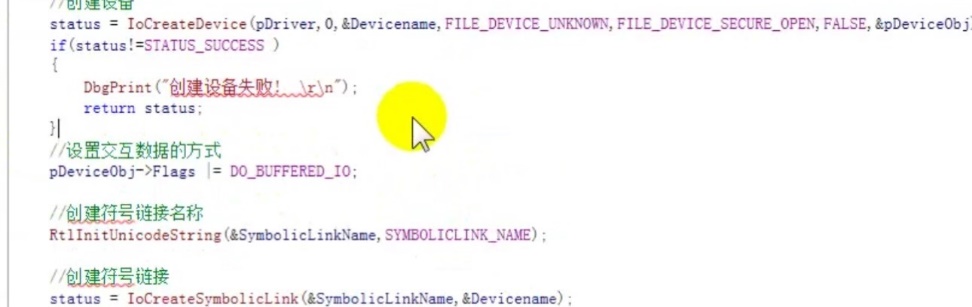


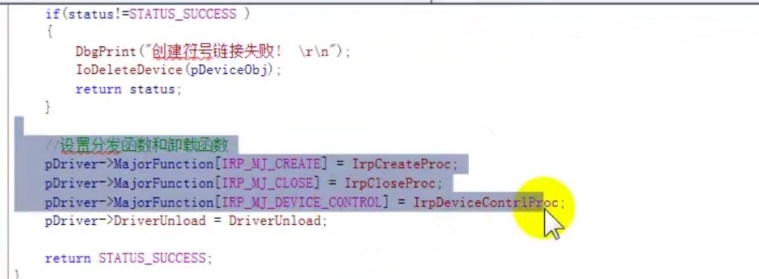




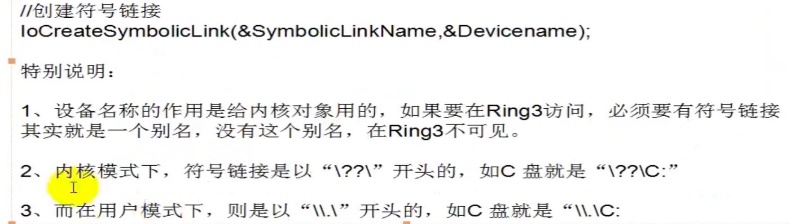
创建完设备之后，还要告诉这个设备，应该通过什么方式与3环进行数据交互：pDeviceObj->Flags|=DO\_BUFFERED\_IO







符号链接一旦创建成功，3环的应用程序才能找到他。

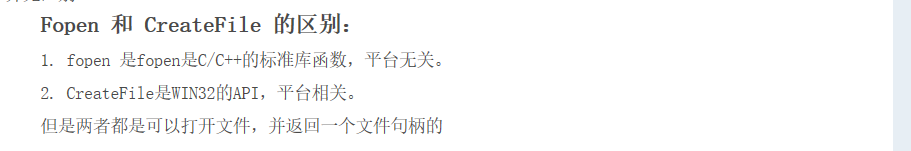


3环在调用CreateFile的时候，由操作系统把参数都封装在IRP结构体中，并且操作系统来根据参数信息将IRP发给具体的设备对象，这个结构体就可以传给设备对象，设备对象根据不同IRP的类型来调用对应的派遣函数。例如如果IRP是create的类型的就调用create类型的派遣函数等等类似的。类似于窗口对象的回调函数的处理过程。

CreateFile函数返回一个设备句柄，之后ReadFile,WriteFile等等函数都要用到这个设备句柄。

在3环要和设备进行通信，首先要创建一个设备，也就是把设备打开，一调用CreateFile函数，在0环就产生了IRP\_MJ\_CREATE的IRP。

Fopen函数最终会调用CreateFile函数，fread函数最终会调用ReadFile函数。



Cli关中断，sti打开中断。

SSDT Hook

过程：通过直接声明引用ntosknrl.exe导出的

并且根据要被hook函数的系统服务号，也就是在系统服务表中函数的索引。

从而确定要修改的地址在该系统服务表中该函数所在的位置处，修改为自己写的函数。

自己写的hook函数中再调用原来初始的函数，否则功能未能实现，这样既可以添加自己的代码，也可以实现原来函数的功能。



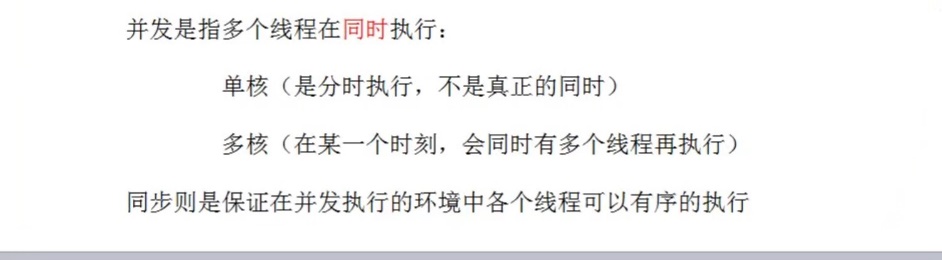
Inline hook：

Jmp/call指令至少占用5个字节，修改时还要绕开全局变量，

Call的汇编指令填写的地址并非真正的地址，例如：call 0x400563

Code =要跳转的地址 – 补丁地址 - 5

多核同步：

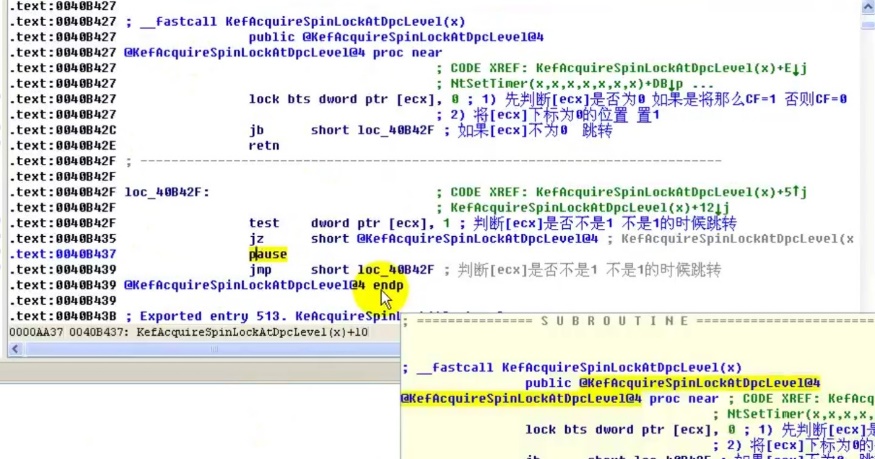




自旋锁是针对多核的来说：







https://www.cnblogs.com/ltyandy/p/11439466.html

重载内核：

1. 申请内存，将ntkrnlpa.exe加载到内存中，按内存对齐展开
2. 根据重定位表修复ntkrnlpa.exe中的全局变量
3. ntkrnlpa.exe也会引用其他dll或者内核的函数，因此也会用到IAT表，修复IAT表，由于是读取的静态的未加载内存的文件，因此IAT表中未填写实际的地址，而是指向函数名称表，因此需要根据dll和函数名来获取地址。
4. 使用当前正在使用的内核的数据来填充新内核的SSDT表.

5.之后Hook KiFastCallEntry来，使调用号走新内核的路线