驱动程序一般调试手段及方法 PDF转换可能丢失图片或格式，建议阅读原文 https://www.100test.com/kao_ti2020/646/2021_2022__E9_A9_B1_E5_8A_A8_E7_A8_8B_E5_c97_646870.htm Windows驱动程序的难于调试是众所周知的，调试步骤繁琐，而且内核环境下固有的多线程环境和代码执行的顺序不确定性，更增加了调试的难度，我自己感觉最好的办法，就是利用DbgPrint（我自己则最常使用KdPrint）打印出足够多的信息，以便于我们分析。下面是一些打印出详细trace的一些手段： 利用__LINE__ __FILE__以及__FUNCTION__定位代码位置 这几个编译器指令分别指示当前代码所在的行号、文件名和函数名；你可以在你的代码中定义如下的宏： #define DBG_TRACER \"%s(%d)-%s\" #define DBG_ARGS __FILE__,__LINE__,__FUNCTION__ 然后，你可以像这样使用这两个宏： KdPrint((DBG_TRACER\"你的字符串消息\\n\",DBG_ARGS)). 2. 利用Osr上的OsrNTStatusToString将你的NTSTATUS状态码值转换成对应的字符串值：当你要打印一个错误状态码时，直接打印出表意的字符可能会为你节省一些时间； 3. 打印UNICODE_STRING：本来这是一个比较easy的事情，但是驱网上不时有人说到微软给的sfilter例子里面获取文件全路径名的函数无法获取中文路径，实际是不是不能获取，而是KdPrint打印的时候，没有把中文字符打印出来，转换成ANSI_STRING，然后再打印出来，代码如下： { ANSI_STRING AnsiStr. NTSTATUS status. status = RtlUnicodeStringToAnsiString(if (NT_SUCCESS(status)) { KdPrint((AnsiStr.Buffer)). RtlFreeAnsiString(} } 4. 利用Filemon中

的代码：Filemon本身是很有用的一个工具，当然，它的源代码也是一座金矿，你可以从中学到很多东西。利用FileMon中的若干代码，可以将对应的IRP转换成对应的字符打印出来（FileSpy或者MiniSpy中的PrintIrpCode可以做同样的事情）；也可以将FileInformationClass对应的字符打印出来；最后，你可能用到的最多的，就是将FileMon中的打印当前进程名的代码捣鼓出来自己使用，稍加改变，你也可以用它来打印任意一个你获取了PEPROCESS进程指针的进程名(注意在一个较低的IRQL下使用它)；5. 文章的最后，给出一些其他的代码，包括：打印IRP的Flags值、打印FILE_OBJECT的Flags值、打印Volume的DeviceType值以及打印FILE_OBJECT结构中的BOOLEAN值设置，代码中也包括了上文提到了从filemon中抽出的部分代码，都是些体力活，如果你自己不想写，直接下载就好!#ffffff>百考试题(www．100test。com) 100Test 下载频道开通，各类考试题目直接下载。详细请访问 www.100test.com