这次不是调试，是总结一下异常处理流程，总结来自看雪《0day2》

(1)CPU执行异常时发生并捕捉异常，内核接过进程的控制权，开始内核态的异常处理

(2)内核处理异常结束，将控制权还给ring3

(3)ring3中的第一个异常处理的函数是ntdll.dll中的KiUserExceptionDispatcher()函数

(4)KiUserExceptionDispatcher()首先检查程序是否处于调试状态，如果程序正在被调试，会将异常交给调试器进行处理

(5)在非调试状态下，KiUserExceptionDispatcher()调用RtlDispatchException()函数对线程的S.E.H表进行遍历，如果找到能够处理异常的回调函数，将再次遍历先前调用过的S.E.H句柄 ，即unwind操作，以保证异常处理机制自身的完整性

(6)如果栈中的S.E.H都失败了，且用户曾经使用过SetUnhandledExceptionFilter()函数设定进程异常处理，则这个异常处理将被调用

(7)如果用户自定义的进程异常处理失败，或者用户根本没有定义进程异常处理，那么系统默认的异常处理UnhandlerExceptionFilter()将被调用，U.E.F会根据注册表里的相关信息决定是默默地关闭程序，还是弹出错误对话框

以上就是windows 2000平台的异常处理流程，windows xp以后的操作系统的异常处理流程大致相同(看雪这本书出版了有点年头了，所以这句话只是相对xp，win7说的，有待修改)，只是KiUserExceptionDispatcher()在遍历栈帧中的S.E.H之前，会先去尝试一种新加入的异常处理类型V.E.H