Win32调试API第一部分 PDF转换可能丢失图片或格式，建议阅读原文
https://www.100test.com/kao_ti2020/259/2021_2022_Win32_E8_B0_83_E8_AF_c98_259397.htm 理论:Win32有一些供程序员使用的API,它们提供相当于调试器的功能. 他们被称作Win32调试API(或原语).利用这些API,我们可以:加载一个程序或捆绑到一个正在运行的程序上以供调试 获得被调试的程序的低层信息,例如进程ID,进入地址,映像基址等. 当发生与调试有关的事件时被通知,例如进程/线程的开始/结束, DLL的加载/释放等. 修改被调试的进程或线程 简而言之,我们可以用这些API写一个简单的调试器.由于这个题目有些过大,我把它分为几部分,而本教程就是它的第一部分.在本教程中,我将讲解一些基本概念及Win32调试API的大致框架.使用Win32调试API的步骤如下:创建一个进程或捆绑到一个运行中的进程上. 这是使用Win32调试API的第一步.由于我们的程序要扮演调试器的角色,我们要找一个供调试的程序.一个被调试的程序被称为debuggee.可以通过以下两种方式获得debuggee: 通过CreateProcess创建debuggee进程.为了创建被调试的进程,必须指定DEBUG_PROCESS标志.这一标志告诉Windows我们要调试该进程. 当debuggee中发生重要的与调试有关的事件(调试事件)时,Windows 会向我们的程序发送通知.debuggee会立即挂起以等待我们的程序准备好.如果debuggee还创建了子进程,Windows还会为每个子进程中的调试事件向我们的程序发送通知.这一特性通常是不必要的.我们可以通过指定DEBUG_ONLY_THIS_PROCESS与 DEBUG_PROCESS的组合标志来禁止它. 我们也可以用 DebugActiveProcess标志捆绑

到一个运行中的进程上. 等待调试事件. 在获得了一个debuggee进程后,debuggee的主线程被挂起,这种状况将持续到我们的程序调用WaitForDebugEvent为止.这个函数和其他的WaitForXXX函数相似,比如说,它阻塞调用线程直到等待的事件发生.对这个函数来说, 它等待由Windows发送的调试事件.下面是它的定义: WaitForDebugEvent proto lpDebugEvent:DWORD, dwMilliseconds:DWORDlpDebugEvent is the address of a DEBUG_EVENT这个结构将被填入关于debuggee中发生的调试事件的信息.dwMilliseconds 该函数等待调试事件的时间,以毫秒为单位.如果这段时间没有调试事件发生, WaitForDebugEvent返回调用者.另一方面,如果将该参数指定为INFINITE 常数,函数将一直等待直到调试事件发生.现在我们看一下DEBUG_EVENT 结构.DEBUG_EVENT STRUCT dwDebugEventCode dd ? dwProcessId dd ? dwThreadId dd ? u DEBUGSTRUCT DEBUG_EVENT ENDS dwDebugEventCode 该值指定了等待发生的调试事件的类型.因为有很多种类型的事件发生,我们的程序要检查该值,知道要发生事件的类型并做出响应. 该值可能的取值如下:取值 含义 CREATE_PROCESS_DEBUG_EVENT 进程被创建.当debuggee进程刚被创建(还未运行) 或我们的程序刚以DebugActiveProcess被捆绑到一个运行中的进程时事件发生.这是我们的程序应该获得的第一个事件. EXIT_PROCESS_DEBUG_EVENT 进程退出. CREATE_THEAD_DEBUG_EVENT 当一个新线程在deuggee进程中创建或我们的程序首次捆绑到运行中的进程时事件发生.要注意的是当debugge的主线程被创建时不会收到该通知.

EXIT_THREAD_DEBUG_EVENT debuggee中的线程退出时事件发生.debugee的主线程退出时不会收到该通知.我们可以认为debuggee的主线程与debugge进程是同义词. 因此, 当我们的程序看到CREATE_PROCESS_DEBUG_EVENT标志时,对主线程来说,就是CREATE_THREAD_DEBUG_EVENT标志. LOAD_DLL_DEBUG_EVENT debuggee装入一个DLL.当PE装载器第一次分解指向DLL的链接时,我们将收到这一事件. (当调用CreateProcess装入 debuggee时)并且当debuggee调用LoadLibrary时也会发生. UNLOAD_DLL_DEBUG_EVENT 一个DLL从debuggee中卸载时事件发生. EXCEPTION_DEBUG_EVENT 在debuggee中发生异常时事件发生. 注意: 该事件仅在debuggee开始它的第一条指令之前发生一次.异常实际上是一个调试中断(int 3h).如果想恢复debuggee事,以 DBG_CONTINUE 标志调用ContinueDebugEvent 函数. 不要使用DBG_EXCEPTION_NOT_HANDLED 标志否则debuggee会在NT下拒绝运行(Win98下运行得很好). OUTPUT_DEBUG_STRING_EVENT 当debuggee调用DebugOutputString函数向我们的程序发送消息字符串时该事件发生. RIP_EVENT 系统调试发生错误 dwProcessId 和dwThreadId发生调试事件的进程和线程Id.我们可以用这些值作为我们感兴趣的进程或线程的标志符.记住如果我们使用CreateProcess来装载debuggee,我们仍可在PROCESS_INFO结构中获得debuggee的进程和线程.我们可以用这些值来区别调试事件是发生在debuggee中还是它的子进程中(当没有指定 DEBUG_ONLY_THIS_PROCESS 标志时).u 是一个联合,包含了调试事件的更多信息.根据上面dwDebugEventCode的不同,它

可以是以下结构:dwDebugEventCode u的解释
CREATE_PROCESS_DEBUG_EVENT 名为CreateProcessInfo
的CREATE_PROCESS_DEBUG_INFO结构
EXIT_PROCESS_DEBUG_EVENT 名为ExitProcess
的EXIT_PROCESS_DEBUG_INFO结构
CREATE_THREAD_DEBUG_EVENT 名为CreateThread
的CREATE_THREAD_DEBUG_INFO结构
EXIT_THREAD_DEBUG_EVENT 名为ExitThread
的EXIT_THREAD_DEBUG_EVENT 结构
LOAD_DLL_DEBUG_EVENT 名为LoadDll
的LOAD_DLL_DEBUG_INFO 结构
UNLOAD_DLL_DEBUG_EVENT 名为UnloadDll
的UNLOAD_DLL_DEBUG_INFO结构
EXCEPTION_DEBUG_EVENT 名为Exception
的EXCEPTION_DEBUG_INFO结构
OUTPUT_DEBUG_STRING_EVENT 名为DebugString
的OUTPUT_DEBUG_STRING_INFO 结构 RIP_EVENT 名
为RipInfo的RIP_INFO 结构 