 一：引言： 你也许一直对金山词霸的屏幕抓词的实现原理感到困惑，你也许希望将你的键盘，鼠标的活动适时的记录下来，甚至你想知道木马在windows操作系统是怎样进行木马dll的加载的…..其实这些都是用到了windows的钩子函数。因此本文将对钩子函数的相关知识进行阐述。当然，本文的目的并不是想通过此程序让读者去窃取别人的密码，只是由于钩子函数在windows系统中是一个非常重要的系统接口函数，所以想和大家共同的探讨，当然本文也对怎样建立动态连结库（DLL）作了一些简单的描述。（本文的程序为vc6.0的开发环境，语言是：C和win32 api）。二：钩子概述： 微软的windowsX操作系统是建立在事件驱动的机制上的，也就是通过消息传递来实现。而钩子在windows操作系统中，是一种能在事件（比如：消息、鼠标激活、键盘响应）到达应用程序前中途接获事件的机制。而且，钩子函数还可以通过修改、丢弃等手段来对事件起作用。 Windows 有两种钩子，一种是特定线程钩子（Thread specific hooks），一种是全局系统钩子(Systemwide hooks)。特定线程钩子只是监视指定的线程，而全局系统钩子则可以监视系统中所有的线程。无论是特定线程钩子，还是全局系统钩子，都是通过SetWindowsHookEx ()来设置钩子的。对于特定线程钩子，钩子的函数既可以是包含在一个.exe也可以是一个.dll。但是对于一个全局系统钩子，钩子函数必须包含在独立的dll中，因此，当我们要捕捉键盘响应时，我

们必须创建一个动态链接库。但是当钩子函数在得到了控制权，并对相关的事件处理完后，如果想要该消息得以继续的传递，那么则必须调用另一个函数：CallNextHookEx。由于系统必须对每个消息处理，钩子程序因此增加了处理的负担，因此也降低了系统的性能。鉴于这一点，在windows ce中对钩子程序并不支持。所以当程序完成并退出时，应当释放钩子，调用函数：UnhookWindowsHookEx。 下面我们将举一个例子（捕捉键盘）来详细的讲解钩子函数的程序设计。三：程序的设计：I:设置钩子设置钩子是通过SetWindowsHookEx ()的API函数.原形: HHOOK SetWindowsHookEx(int idHook,HOOKPROC lpfn,HINSTANCE hMod,DWORD dwThreadId)idhook:装入钩子的类型.lpfn: 钩子进程的入口地址hMod: 应用程序的事件句柄dwThreadId: 装入钩子的线程标示参数：idHook:这个参数可以是以下值：WH_CALLWNDPROC、WH_CALLWNDPROCRET、WH_CBT、WH_DEBUG、WH_FOREGROUNDIDLE、WH_GETMESSAGE、WH_JOURNALPLAYBACK、WH_JOURNALRECORD、WH_KEYBOARD、WH_KEYBOARD_LL、WH_MOUSE、WH_MOUSE_LL、WH_MSGFILTER、WH_SHELL、WH_SYSMSGFILTER。对于这些参数，我不想一一加以解释，因为MSDN中有关于他们的详细注解。我只挑选其中的几个加以中文说明。WH_KEYBOARD：一旦有键盘敲打消息（键盘的按下、键盘的弹起），在这个消息被放在应用程序的消息队列前，WINDOWS将会调用你的钩子函数。钩子函数可以改变和丢弃键盘敲打消息。WH_MOUSE：每个鼠标消息在被放在

应用程序的消息队列前，WINDOWS将会调用你的钩子函数。钩子函数可以改变和丢弃鼠标消息。 WH_GETMESSAGE：每次当你的应用程序调用一个GetMessage()或者一个PeekMessage()为了去从应用程序的消息队列中要求一个消息时，WINDOWS都会调用你的钩子函数。而钩子函数可以改变和丢弃这个消息。II:释放钩子钩子的释放使用的是UnhookWindowsHookEx（）函数原形：BOOL UnhookWindowsHookEx(HHOOK hhk)UnhookWindowsHookEx（）函数将释放的是钩子链中函数SetWindowsHookEx所装入的钩子进程。hhk: 将要释放的钩子进程的句柄。III：钩子进程 钩子进程使用函数HookProc.其实HookProc仅仅只是应用程序定义的符号。比如你可以写成KeyBoardHook.但是参数是不变的。Win32 API提供了诸如：CallWndProc、GetMsgProc、DebugProc、CBTProc、MouseProc、KeyboardProc、MessageProc等函数，对于他们的详细讲解，可以看MSDN我在此只讲解一下KeyBoardHook的含义。原形：LRESULT CALLBACK KeyBoardHook (int nCode, WPARAM wParam, LPARAM lParam)说明：钩子进程是一些依附在一个钩子上的一些函数，因此钩子进程只被WINDOWS调用而不被应用程序调用，他们有时就需要作为一个回调函数（CALLBACK）。参数说明： nCode:钩子代码，钩子进程使用钩子代码去决定是否执行。而钩子代码的值是依靠钩子的种类来定的。每种钩子种类都有他们自己一系列特性的代码。比如对于WH_KEYBOARD，钩子代码的参数有：HC_ACTION，HC_NOREMOVE。HC_ACTION的意义：参数wParam 和lParam 包含了键盘敲打消息的信息

，HC_NOREMOVE的意义：参数wParam 和lParam包含了键盘敲打消息的信息，并且，键盘敲打消息一直没有从消息队列中删除。(应用程序调用PeekMessage函数，并且设置PM_NOREMOVE标志)。也就是说当nCode等于HC_ACTION时，钩子进程必须处理消息。而为HC_NOREMOVE时，钩子进程必须传递消息给CallNextHookEx函数，而不能做进一步的处理，而且必须有CallNextHookEx函数的返回值。 wParam:键盘敲打所产生的键盘消息，键盘按键的虚拟代码。lParam:包含了消息细节。注意:如果钩子进程中nCode小于零，钩子进程必须返回(return) CallNextHookEx(nCode,wParam,lParam).而钩子进程中的nCode大于零，但是钩子进程并不处理消息，作者推荐你调用CallNextHookEx并且返回该函数的返回值。否则，如果另一个应用程序也装入WH_KEYBOARD 钩子，那么该钩子将不接受钩子通知并且返回一个不正确的值。如果钩子进程处理了消息，它可能返回一个非零值去阻止系统传递该信息到其它剩下的钩子或者windows进程。所以最好在钩子进程的最后都返回CallNextHookEx的返回值。 IV:调用下一个钩子函数调用下一个钩子函数时使用CallNexHookEx函数。原形：LRESULT CallNextHookEx(HHOOK hhk, int nCode, WPARAM wParam, LPARAM lParam)CallNexHookEx()函数用于对当前钩子链中的下一个钩子进程传递钩子信息，一个钩子进程既可以在钩子信息处理前，也可以在钩子信息处理后调用该函数。为什么使用该函数已在iii钩子进程中的“ 注意 ”中，加以了详细的说明。hhk: 当前钩子的句柄nCode: 传送到钩子进程的钩子代码。wParam:传送到钩子进程的值

。lParam:传送到钩子进程的值。参数：hhk: 当前钩子的句柄. 应用程序接受这个句柄，作为先前调用SetWindowsHookE函数的结果 nCode: 传送到钩子进程的钩子代码，下一个钩子进程使用这个代码以此决定如何处理钩子信息 wParam:传送给钩子进程的wParam 参数值，参数值的具体含义与当前钩子链的挂接的钩子类型有关lParam : 传送给钩子进程的wParam 参数值，参数值的具体含义与当前钩子链的挂接的钩子类型有关 返回值：返回值是链中下一个钩子进程返回的值，当前钩子进程必须返回这个值，返回值的具体含义与挂接的钩子类型有关，详细信息请参看具体的钩子进程描述。V 建立一个动态连接库（DLL） 当我们熟悉了以上的各个函数后，现在我们开始编写一个动态连接库（DLL）。在这儿我采用的是WIN32 DLL,而不是MFC DLL。而且以下所有的程序也都是采用C语言去编写。这主要是因为使用WIN32 API能够更详细、更全面的控制程序的如何执行，而使用MFC，一些低级的控制是不可能实现的(当然，仅对该程序来说，也是可以使用MFC的)。