钩子技术、HOOK技术在VC编程中的应用计算机等级考试
 本文针对HOOK技术在VC编程中的应用进行讨论，并着重对应用比较广泛的全局HOOK做了阐述。 引言 Windows操作系统是建立在事件驱动机制之上的，系统各部分之间的沟通也都是通过消息的相互传递而实现的。但在通常情况下，应用程序只能处理来自进程内部的消息或是从其他进程发过来的消息，如果需要对在进程外传递的消息进行拦截处理就必须采取一种被称为HOOK（钩子）的技术。钩子是Windows操作系统中非常重要的一种系统接口，用它可以轻松截获并处理在其他应用程序之间传递的消息，并由此可 以完成一些普通应用程序难以实现的特殊功能。基于钩子在消息拦截处理中的强大功能，本文即以VC 6.0为编程背景对钩子的基本概念及其实现过程展开讨论。为方便理解，在文章最后还给出了一个简单的有关鼠标钩子的应用示例。 钩子的基本原理 钩子的本质是一段用以处理系统消息的程序，通过系统调用，将其挂入到系统。钩子的种类有很多，每一种钩子负责截获并处理相应的消息。钩子机制允许应用程序截获并处理发往指定窗口的消息或特定事件，其监视的窗口即可以是本进程内的也可以是由其他进程所创建的。在特定的消息发出，并在到达目的窗口之前，钩子程序先行截获此消息并得到对其的控制权。此时在钩子函数中就可以对截获的消息进行各种修改处理，甚至强行终止该消息的继续传递。 任何一个钩子都由系统来维护一个指针列表（钩子链表），其指针指向钩子的各个处理

函数。最近安装的钩子放在链的开始，最早安装的钩子则放在最后，当钩子监视的消息出现时，操作系统调用链表开始处的第一个钩子处理函数进行处理，也就是说最后加入的钩子优先获得控制权。在这里提到的钩子处理函数必须是一个回调函数（callback function），而且不能定义为类成员函数，必须定义为普通的C函数。在使用钩子时可以根据其监视范围的不同将其分为全局钩子和线程钩子两大类，其中线程钩子只能监视某个线程，而全局钩子则可对在当前系统下运行的所有线程进行监视。显然，线程钩子可以看作是全局钩子的一个子集，全局钩子虽然功能强大但同时实现起来也比较烦琐：其钩子函数的实现必须封装在动态链接库中才可以使用。 钩子的安装与卸载 由于全局钩子具有相当的广泛性而且在功能上完全覆盖了线程钩子，因此下面就主要对应用较多的全局钩子的安装与使用进行讨论。前面已经提过，操作系统是通过调用钩子链表开始处的第一个钩子处理函数而进行消息拦截处理的。因此，为了设置钩子，只需将回调函数放置于链首即可，操作系统会使其首先被调用。在具体实现时由函数SetWindowsHookEx()负责将回调函数放置于钩子链表的开始位置。 SetWindowsHookEx()函数原型声明如下: HHOOK SetWindowsHookEx(int idHook. HOOKPROC lpfn. HINSTANCE hMod. DWORD dwThreadId). 其中：参数idHook指定了钩子的类型，总共有如下13种： WH_CALLWNDPROC 系统将消息发送到指定窗口之前的"钩子" WH_CALLWNDPROCRET 消息已经在窗口中处理的"钩子" WH_CBT 基于计算机培训的"钩子" WH_DEBUG 差错"钩子" WH_FOREGROUNDIDLE 前台空闲窗口"钩子"

WH_GETMESSAGE 接收消息投递的"钩子"
WH_JOURNALPLAYBACK 回放以前通过WH_JOURNALRECORD"钩子"记录的输入消息
WH_JOURNALRECORD 输入消息记录"钩子"
WH_KEYBOARD 键盘消息"钩子" WH_MOUSE 鼠标消息"钩子" WH_MSGFILTER 对话框、消息框、菜单或滚动条输入消息"钩子" WH_SHELL 外壳"钩子" WH_SYSMSGFILTER 系统消息"钩子" 参数lpfn为指向钩子处理函数的指针，即回调函数的首地址；参数hMod则标识了钩子处理函数所处模块的句柄；第四个参数dwThreadId 指定被监视的线程，如果明确指定了某个线程的ID就只监视该线程，此时的钩子即为线程钩子；如果该参数被设置为0，则表示此钩子为监视系统所有线程的全局钩子。此函数在执行完后将返回一个钩子句柄。 