钩子类型

每一种类型的Hook可以使应用程序能够监视不同类型的系统消息处理机制。下面描述所有可以利用的Hook类型。

1、WH\_CALLWNDPROC和WH\_CALLWNDPROCRET Hooks

WH\_CALLWNDPROC和WH\_CALLWNDPROCRETHooks使你可以监视发送到窗口过程的消息。系统在消息发送到接收窗口过程之前调用WH\_CALLWNDPROCHook子程，并且在窗口过程处理完消息之后调用WH\_CALLWNDPROCRET Hook子程。

WH\_CALLWNDPROCRET Hook传递指针到CWPRETSTRUCT结构，再传递到Hook子程。

CWPRETSTRUCT结构包含了来自处理消息的窗口过程的返回值，同样也包括了与这个消息关联的消息参数。

2、WH\_CBT Hook

在以下事件之前，系统都会调用WH\_CBT Hook子程，这些事件包括：

1. 激活，建立，销毁，最小化，最大化，移动，改变尺寸等窗口事件；

2. 完成系统指令；

3. 来自系统消息队列中的移动鼠标，键盘事件；

4. 设置输入焦点事件；

5. 同步系统消息队列事件。

Hook子程的返回值确定系统是否允许或者防止这些操作中的一个。

3、WH\_DEBUG Hook

在系统调用系统中与其他Hook关联的Hook子程之前，系统会调用WH\_DEBUGHook子程。你可以使用这个Hook来决定是否允许系统调用与其他Hook关联的Hook子程。

4、WH\_FOREGROUNDIDLE Hook

当应用程序的前台线程处于空闲状态时，可以使用WH\_FOREGROUNDIDLEHook执行低优先级的任务。当应用程序的前台线程大概要变成空闲状态时，系统就会调用WH\_FOREGROUNDIDLEHook子程。

5、WH\_GETMESSAGE Hook

应用程序使用WH\_GETMESSAGE Hook来监视从GetMessage orPeekMessage函数返回的消息。你可以使用WH\_GETMESSAGEHook去监视鼠标和键盘输入，以及其他发送到消息队列中的消息。

6、WH\_JOURNALPLAYBACK Hook

WH\_JOURNALPLAYBACKHook使应用程序可以插入消息到系统消息队列。可以使用这个Hook回放通过使用WH\_JOURNALRECORDHook记录下来的连续的鼠标和键盘事件。只要WH\_JOURNALPLAYBACKHook已经安装，正常的鼠标和键盘事件就是无效的。

WH\_JOURNALPLAYBACK Hook是全局Hook，它不能象线程特定Hook一样使用。

WH\_JOURNALPLAYBACKHook返回超时值，这个值告诉系统在处理来自回放Hook当前消息之前需要等待多长时间（毫秒）。这就使Hook可以控制实时事件的回放。

WH\_JOURNALPLAYBACK是system-wide localhooks，它??不??射到任何行程位址空駣??

7、WH\_JOURNALRECORD Hook

WH\_JOURNALRECORDHook用来监视和记录输入事件。典型的，可以使用这个Hook记录连续的鼠标和键盘事件，然后通过使用WH\_JOURNALPLAYBACKHook来回放。

WH\_JOURNALRECORD Hook是全局Hook，它不能象线程特定Hook一样使用。

WH\_JOURNALRECORD是system-wide local hooks，它??不??射到任何行程位址空駣??

8、WH\_KEYBOARD Hook

在应用程序中，WH\_KEYBOARD Hook用来监视WM\_KEYDOWN andWM\_KEYUP消息，这些消息通过GetMessage or PeekMessagefunction返回。可以使用这个Hook来监视输入到消息队列中的键盘消息。

9、WH\_KEYBOARD\_LL Hook

WH\_KEYBOARD\_LL Hook监视输入到线程消息队列中的键盘消息。

10、WH\_MOUSE Hook

WH\_MOUSE Hook监视从GetMessage 或者 PeekMessage函数返回的鼠标消息。使用这个Hook监视输入到消息队列中的鼠标消息。

11、WH\_MOUSE\_LL Hook

WH\_MOUSE\_LL Hook监视输入到线程消息队列中的鼠标消息。

12、WH\_MSGFILTER 和 WH\_SYSMSGFILTER Hooks

WH\_MSGFILTER 和 WH\_SYSMSGFILTERHooks使我们可以监视菜单，滚动条，消息框，对话框消息并且发现用户使用ALT+TAB or ALT+ESC组合键切换窗口。WH\_MSGFILTERHook只能监视传递到菜单，滚动条，消息框的消息，以及传递到通过安装了Hook子程的应用程序建立的对话框的消息。WH\_SYSMSGFILTERHook监视所有应用程序消息。

WH\_MSGFILTER 和 WH\_SYSMSGFILTERHooks使我们可以在模式循环期间过滤消息，这等价于在主消息循环中过滤消息。

通过调用CallMsgFilter function可以直接的调用WH\_MSGFILTERHook。通过使用这个函数，应用程序能够在模式循环期间使用相同的代码去过滤消息，如同在主消息循环里一样。

13、WH\_SHELL Hook

外壳应用程序可以使用WH\_SHELLHook去接收重要的通知。当外壳应用程序是激活的并且当顶层窗口建立或者销毁时，系统调用WH\_SHELL Hook子程。

WH\_SHELL 共有５钟情?r：

1. 只要有个top-level、unowned 窗口被产生、起作用、或是被摧毁；

2. 当Taskbar需要重画某个按钮；

3. 当系统需要显示关于Taskbar的一个程序的最小化形式；

4. 当目前的键盘布局状态改变；

5. 当使用者按Ctrl+Esc去执行Task Manager（或相同级别的程序）。

按照惯例，外壳应用程序都不接收WH\_SHELL消息。所以，在应用程序能够接收WH\_SHELL消息之前，应用程序必须调用SystemParametersInfofunction注册它自己。

### WH\_SHELL

函数原形:

**[cpp]** [view plain](http://blog.csdn.net/wangjieest/article/details/7107779" \o "view plain)[copy](http://blog.csdn.net/wangjieest/article/details/7107779" \o "copy)

1. **LRESULT** CALLBACK ShellProc( **int** nCode, **WPARAM** wParam,**LPARAM** lParam );

函数功能:

挂钩处理过程是应用程序或库中定义的回调函数,它与函数 SetWindowsHookEx搭配使用.此函数从系统接受外壳(shell)通知.类型HOOKPROC定义了指向此类回调函数的指针.ShellProc时应用程序或库中定义的相应回调的位置标志符.

参数:  
nCode:

指示挂钩类型,该参数可以是以下值之一:

HSHELL\_ACCESSIBILITYSTATE: 对Windows NT5.0或以上版本有效,指示"可访问性"已被改变.  
  
HSHELL\_ACTIVATESHELLWINDOW: shell应当激活其主窗口.  
  
HSHELL\_GETMINRECT: Windows 95 only: 一个窗口正在被最小化或者最大化,系统需要该窗口被最小化时的矩形坐标.  
  
HSHELL\_LANGUAGE : Windows 95 only: 键盘语言被改变或者一个新的键盘布局被加载.  
  
HSHELL\_REDRAW: Windows 95 only: 一个窗口在任务条上的标题已被重画.  
  
HSHELL\_TASKMAN: Windows 95 only: 用户已选择其任务列表.一个提供了任务的列表的shell应用程序当返回TRUE,以阻止Windows开始(执行)其任务列表.  
  
HSHELL\_WINDOWACTIVATED: Windows 95 only: 激活状态已被转移给一个不同的顶层无属性主窗口.  
  
HSHELL\_WINDOWCREATED: 一个顶层无属性主窗口已被创建.当系统调用一个SheProc函数时,该窗口存在.  
  
HSHELL\_WINDOWDESTROYED:一个顶层无属性主窗口即将销毁.当系统调用SheProc 函数时.该窗口仍然存在.  
  
若nCode的值小于0,则此挂钩处理过程必须将该消息不加处理地传送给函数CallNextHookEx 并返回其返回值.  
  
wParam:

此参数的值依赖于参数nCode,其依赖关系如下所示:

HSHELL\_ACCESSIBILITYSTATE: 指示哪一个可以访问特征已被改变了状态,可以是以下值之一:  
  
ACCESS\_FILTERKYS,ACCESS\_MOUSEKEYS,ACCESS\_STICKKEYS.  
  
HSHELL\_GETMINRECT:被最小化或者最大化的窗口句柄.HSHELL\_LANGUAGE: 窗口的句柄.  
  
HSHELL\_REDRAW:被重画的窗口的句柄.HSHELL\_WINDOWACTIVATED:被激活的窗口的句柄.  
  
HSHELL\_WINDOWCREATED:被创建的窗口的句柄.HSHELL\_WINDOWDESTROYED:被销毁的窗口的句柄.  
  
lParam:

此参数的值依赖于参数nCode,其依赖关系如下所示:

HSHELL\_GETMINRECT:指向该RECT结构的指针.HSHELL\_LANGUAGE: 键盘布局的句柄.  
  
HSHELL\_REDRAW: 若该窗口正在闪现,则其值为TRUE,否则为FALSE.  
  
HSHELL\_WINDOWACTIVATED: 若该窗口是全屏模式,则其值为TRUE,否则为FALSE.返回值:返回值应为0.

备注:

此挂钩处理过程通过调用函数SetWindowsHookEx ,指定WH\_SHLL挂钩类型及其相应挂钩处理过程首地址来安装.

### WH\_SYSMSGFILTER

函数原形:

**[cpp]** [view plain](http://blog.csdn.net/wangjieest/article/details/7107779" \o "view plain)[copy](http://blog.csdn.net/wangjieest/article/details/7107779" \o "copy)

1. **LRESULT** CALLBACK SysMsgProc(**int** nCode,**WPARAM** wParam,**LPARAM** lParam);

函数功能:

挂钩处理过程是应用程序或库中定义的回调函数,它与函数 SetWindowsHookEx搭配使用.当一个对话框、消息框、菜单条、或滚动条中的输入事件发生，且由此引发的消息尚未被处理之前,系统对此挂钩处理过程进行调用.此函数可以对系统中所有的对话框、消息框、菜单条、或滚动条消息进行监视.  
  
类型HOOKPROC定义了指向此类回调函数的指针.SysMsgProc 是库中相应回调函数名的位置标志符.  
  
参数:  
  
nCode:

指示产生此消息的输入事件类型.此参数可以是以下值之一:  
  
MSGF\_DIALOGBOX: 输入事件由一个消息框或者对话框产生.  
  
MSGF\_MENU: 输入事件由一个菜单条产生.  
  
MSGF\_SCROLLBAR: 输入事件由一个滚动条产生.  
  
MSGF\_NEXTWINDOW: 输入事件由于用户摁下组合键<Alt+Tab>以激活另一个窗口而产生.  
  
若nCode的值小于0,则此挂钩处理过程必须将该消息不加处理地传送给函数 CallNextHookEx 并返回其返回值.

wParam:

为NULL(0).  
  
  
lParam:

指向MSG结构的[指针].  
  
返回值:若nCode的值小于0,则此挂钩处理过程必须返回CallNextHookEx 的返回值.若nCode的值大于或等于0,并且此挂钩处理过程未对该消息进行处理,则调用函数CallNextHookEx 并返回其返回值是被推荐的.否则,其他安装了WH\_SYSMSGFILTER挂钩的应用程序将无法收到此挂钩通知,并可能由此导致错误的行为.若此挂钩处理过程处理了此消息,它应返回一个非零值以避免系统再将此消息传送给目标窗口处理过程.  
  
备注:一个应用程序通过调用函数SetWindowsHookEx 指定 WH\_SYSMSGFILTER挂钩类型及相应挂钩处理过程首地址来安装此挂钩.

### 

### WH\_MSGFILTER

函数原形：

**[cpp]** [view plain](http://blog.csdn.net/wangjieest/article/details/7107779" \o "view plain)[copy](http://blog.csdn.net/wangjieest/article/details/7107779" \o "copy)

1. **BOOL** CallMsgFilter(LPMSG lpMsg, **int** nCode);

函数功能:

该函数将特定消息和挂钩代码发送给与挂钩 WH\_SYSMSGFILTER和WH\_MSGFILTER 相联系的挂钩处理过程,一个 WH\_SYSMSGFILTER 或者 WH\_MSGFILTER挂钩处理过程是指应用程序定义的回调函数,这些回调函数负责检查,并可有选择地修改关于对话框、消息框、菜单条、滚动条的消息。  
  
参数：  
  
lpMsg：

指向一个MSG结构,该结构包含发送给挂钩处理过程的消息.  
  
nCode:

指定挂钩处理过程需要使用的应用程序定义的代码,以确定如何对消息进行处理,该代码切忌采用与系统预定义的与WH\_SYSMSGFILTER和WH\_MSGFILTER挂钩相关的挂钩代码(MSGF\_ 和 HC\_开头)相同的值.  
  
返回值:

若该应用程序可对消息作进一步处理,则返回值为0,若该应用程序不能对该消息作进一步处理,则返回值为非0值.  
  
备注:系统对CallMsgFilter进行调用,以使得应用程序能够检查和控制在对对话框、消息框、菜单条、滚动条的内部处理过程中产生的,或是由用户按下<Alt+Tab>组合键以激活不同窗口时产生的消息流.  
  
可通过调用函数SetWindowsHookEx来安装此挂钩处理过程.

### 

### 

### WH\_CALLWNDPROC

函数原形:

**[cpp]** [view plain](http://blog.csdn.net/wangjieest/article/details/7107779" \o "view plain)[copy](http://blog.csdn.net/wangjieest/article/details/7107779" \o "copy)

1. **LRESULT** CALLBACK CallWndProc(**int** nCode,**WPARAM** wParam,**LPARAM** lParam );

函数功能:

该函数挂钩处理过程是由应用程序定义或由库定义的回调函数,它与函数 SetWindowsHookEx配套使用,系统调用此函数无论何时一旦函数 SendMessage被调用,在将消息发送至相应的窗口处理过程之前,系统先将该消息发送至挂钩处理过程,该挂钩处理过程可以对改消息进行检查,但不能修改之.  
  
类型HOOKPROC定义了指向此类回调函数的指针,CallWndProc是应用程序定义或库定义的函数名的位置标志位.  
  
参数:  
  
nCode:

指定该挂钩处理过程是否必须对该消息进行处理,若nCode 是HC\_ACTION,则该挂钩处理过程必须处理该消息.若nCode的值小于0,则此挂钩处理过程必须在未对该消息作进一步处理的情况下,将其发送给函数 CallNextHookEx并返回由 CallNextHookEx返回的值.  
  
wParam:

指示该消息是否由当前线程发出,若该消息由当前线程发出,其值为非0值,否则,其值为0.  
  
lParam:

指向一个CWPSTRUCT结构,该结构包含了此消息的详细消息.  
  
返回值:

若nCode的值小于0,则此挂钩函数必须返回函数CallNextHookEx所返回的值;若nCode的值大于或等于0,我们极力推荐对函数CallNextHookEx进行调用,并返回其返回值.否则,其他安装了WM\_CALLWNDPROC挂钩的应用程序将无法收到挂钩通知,从而导致错误的行为.若此挂钩没有调用函数CallNextHookEx,则返回值为0.  
  
备注:挂钩处理函数CallWndPro可以对消息进行检查,但不能修改之.在该挂钩处理过程将控制返回给系统之后,该消息被发送至相应的窗口过程.  
一个应用程序通过调用SetWindowsHookEx时指定WM\_CALLWNDPROC挂钩类型及其相应挂钩处理过程的首地址来安装挂钩处理过程.  
挂钩WM\_CALLWNDPROC是在调用函数SendMessage的线程中被调用的,它并非在接受消息的线程中被调用.

### 

### 

### WH\_CALLWNDPROCRET

函数原形:

**[cpp]** [view plain](http://blog.csdn.net/wangjieest/article/details/7107779" \o "view plain)[copy](http://blog.csdn.net/wangjieest/article/details/7107779" \o "copy)

1. **LRESULT** CALLBACK CallWndRetProc(**int** nCode,**WPARAM** wParam,**LPARAM** lParam);

函数功能:

该函数挂钩处理过程是在应用程序或库中定义的回调函数,它与函数 SetWindowsHookEx搭配使用,系统将在调用完 SendMessage之后调用之,该挂钩处理过程可以检查该消息,但不能修改之.  
  
类型HOOKPROC定义了一类指针指向此类回调函数,CallWndRetProc 仅仅是在应用程序或库中定义的相应函数名的位置标志符.  
  
参数:  
nCode:

指定该挂钩处理过程是否必须对该消息进行处理,若nCode 是HC\_ACTION,则该挂钩处理过程必须处理该消息.若nCode的值小于0,则此挂钩处理过程必须不加处理的将其发送给函数CallNextHookEx并返回由 CallNextHookEx返回的值.  
  
wParam:

指示该消息是否由当前线程发出,若该消息由当前线程发出,其值为非0值,否则,其值为0.  
  
lParam:

指向一个CWPSTRUCT结构,该结构包含了此消息的详细消息.  
  
返回值:

若nCode的值小于0,则此挂钩函数必须返回函数CallNextHookEx所返回的值;若nCode的值大于或等于0,我们极力推荐对函数CallNextHookEx进行调用,并返回其返回值.否则,其他安装了挂钩WM\_CALLWNDPROCRET的应用程序将无法收到挂钩通知,从而导致错误的行为.若此挂钩处理过程没有调用函数CallNextHookEx,则返回值为0.

备注: 一个应用程序通过调用SetWindowsHookEx时指定WM\_CALLWNDPROCRET挂钩类型及其相应挂钩处理过程的首地址来安装挂钩处理过程.

### 

### 

### WH\_CBT

函数原形:

**[cpp]** [view plain](http://blog.csdn.net/wangjieest/article/details/7107779" \o "view plain)[copy](http://blog.csdn.net/wangjieest/article/details/7107779" \o "copy)

1. **LRESULT** CALLBACK CBTProc(**int** nCode,**WPARAM** wParam,**LPARAM** lParam);

函数功能:

该函数挂钩处理过程是在应用程序或库中定义的回调函数,它与函数 SetWindowsHookEx配套使用,系统在缉获、创建、销毁、最小化、最大化、移动或改变一个窗口尺寸之前调用它;在结束一个系统命令之前调用它;在将一个鼠标或键盘事件从系统事件队列中删除之前调用它;在设置键盘焦点之前调用它,或在与系统消息队列进行同步之前调用它,一个基于计算机培训(CBT)的应用程序使用此挂钩处理过程来从系统接收有用的通知.  
  
类型HOOKPROC定义了指向此类回调函数的指针,CBTProc是应用程序定义或库定义的相应回调函数名的位置标识符.

参数:  
  
nCode:

指定一个代码,该代码被此挂钩处理过程用来决定如何处理此消息,该参数可以是下值中的一个:  
HCBT\_ACTIVATE: 系统将要激活一个窗口.  
HCBT\_CLICKSKIPPED: 系统已经从系统消息队列中删除了一个鼠标消息,依据收到的这一挂钩代码,一个CBT应用程序必须安装一个 WH\_JOURNALPLAYBACK 挂钩处理过程以响应此鼠标消息.  
HCBT\_CREATEWND:一个窗口将要被创建,系统在发送WM\_CREATE 或WM\_NCCREATE消息至相应窗口之前调用该挂钩处理过程.若该挂钩处理过程返回一个非0值,则系统销毁该窗口,函数CreateWindow返回值为NULL(0),但消息WM\_DESTROY不会被发送至相应窗口,若挂钩处理过程返回为零,窗口将被正常创建.  
当HCBT\_CREATEWND通知发出之时,相应窗口已被创建,但其最终尺寸及位置可能尚未确定,其父窗口也可能还尚未被创建,存在这样一种可能,消息被发送给新被创建的窗口,虽然该窗口尚未接收到WM\_NCCREATE 或 WM\_CREATE 消息.也可能通过对CBT\_CREATEWND结构的hwndInsertAfter域的修改,来改变新被创建窗口在Z序列中的位置.  
HCBT\_DESTROYWND:一个窗口将被销毁.  
HCBT\_KEYSKIPPED:系统已从系统消息队列中删除了一个键盘消息,依据收到的这个挂钩代码,一个CBT应用程序必须安装一个 WH\_JOURNALPLAYBACK挂钩处理过程以响应相应的键盘消息.  
HCBT\_MINMAX: 一个窗口将被最小化或最大化.  
HCBT\_MOVESIZE: 一个窗口将被移动或改变尺寸.  
HCBT\_QS: 系统已从系统消息队列中取到一个WM\_QUEUESYNC 消息.  
HCBT\_SETFOCUS: 一个窗口将要获得键盘焦点.   
HCBT\_SYSCOMMAND: 一个系统命令将被执行.这使得一个CBT应用程序可以避免因热键响应而进行任务切换.  
若nCode的值小于0,此挂钩处理过程必须不加处理地将此消息发送给函数CallNextHookEx,并返回函数CallNextHookEx的返回值.  
  
wParam:

取决于参数nCode的值,详细信息参见以下的说明部分.  
  
lParam:

取决于参数nCode的值,详细信息参见以下的说明部分.  
  
返回值:

此挂钩处理过程的返回值决定了系统允许或禁止以下操作之一,对于与以下挂钩代码相应的操作,返回值为0则允许这样的操作,返回值为1则禁止.  
  
HCBT\_ACTIVATE  
HCBT\_CREATEWNDHCBT\_DESTROYWND  
HCBT\_MINMAX  
HCBT\_MOVESIZEHCBT\_SETFOCUS  
HCBT\_SYSCOMMAND  
对于与以下CBT挂钩代码相对应的操作,返回值被忽略.  
HCBT\_CLICKSKIPPED  
HCBT\_KEYSKIPPED  
HCBT\_QS  
  
备注:

除以下挂钩代码之外,不应当安装WH\_JOURNALPLAYBACK挂钩处理过程.  
  
此挂钩处理过程必须放在某个动态链接库中,一个应用程序通过调用函数 SetWindowsHookEx指定WH\_CBT 挂钩类型与相应挂钩处理过程首地址来安装挂钩处理过程.  
  
以下列示了参数wParam和lParam与每一种HCBT挂钩代码的关系.  
  
指定一个长指针指向结构CBTACTIVATESTRUCT 的长指针,该结构包含了指向激活窗口的句柄,并指示被激活者是否由于点击而发生改变.  
  
 值                            参数wParam 　　 　　 参数lParam   
  
HCBT\_ACTIVATE 　　指定将要被激活的窗口句柄.  
  
指定一个指向结构MOUSEHOOKSTRUCT的长指针,该结构包含点击**[测试](http://lib.csdn.net/base/softwaretest" \o "软件测试知识库" \t "http://blog.csdn.net/whatday/article/details/_blank)**代码及 鼠标消息发往的窗口句柄.HCBT\_ CLICKSKIPPED 值仅当一个WH\_MOUSE挂钩被安装了之时才发送一个CBTProc挂钩处理过程,关于点击 测试代码的列表,参见 WM\_NCHITTEST.  
  
HCBT\_CLICKSKIPPED　 标识鼠标消息从系统消息队列中被删除.  
  
指定一个指向 CBT\_CREATEWND结构的长指针,该结构包含窗口的初始化参数,这些参数包括窗口的坐标和尺寸,通过改变这些参数.一个CBTProc挂钩处理过程能够设置窗口的初始尺寸和位置.  
  
HCBT\_CREATEWND 指定新窗口的句柄.   
  
HCBT\_DESTROYWND 　指定将被销毁的窗口句柄. 　　 　未定义,必须设置为0.  
  
指示重复次数,扫描码,键转义码,前驱键状态和上下文代码.仅当WH\_KEYBOARD挂钩被安装了之时,此HCBT\_KEYSKIPPED值被发送 CBTProc 挂钩处理过程.欲获得更多信息,参见 WM\_KEYUP 或WM\_KEYDOWN  
  
HCBT\_KEYSKIPPED 指示虚拟键的代码.   
  
低"字"部分指定一个显示窗口的值(SW\_),以指定相应操作,详情参见ShowWindow,高"字"部分未定义.  
  
HCBT\_MINMAX 指定被最小化或最大化的窗口句柄    
  
指定一个指向RECT结构的长指针,该结构包含窗口的坐标,通过改变结构的值,一个CBTProc的子处理过程能够设置窗口的最终位置.  
  
HCBT\_MOVESIZE 指定将被移动或改变尺寸的窗口句柄.   
  
HCBT\_QS 未定义,必须设置为0 未定义,必须设置为0  
  
HCBT\_SETFOCUS 指定获得键盘焦点的窗口句柄. 指示失去键盘焦点的窗口句柄.  
  
指示系统命令的值(SC\_)以标识相应系统命令,欲知更多关于系统命令值的信息,参见WM\_SYSCOMMAND.  
  
包含与WM\_SYSCOMMAND消息的lParam值相同的数据.若一个系统菜单命令被鼠标选种,则低"字"部分包含光标在屏幕坐标系中的X坐标值,高"字"部分包含Y坐标值,其他情况下,此参数未被使用.  
  
HCBT\_SYSCOMMAND   
  
类型HOOKPROC定义了一类指针指向此类回调函数,CBTProc 仅仅是在应用程序或库中定义的相应函数名的位置标识符.

### 

### 

### WH\_DUBEG

函数原形:

**[cpp]** [view plain](http://blog.csdn.net/wangjieest/article/details/7107779" \o "view plain)[copy](http://blog.csdn.net/wangjieest/article/details/7107779" \o "copy)

1. **LRESULT** CALLBACK DebugProc(**int** nCode,**WPARAM** wParam,**LPARAM** lParam);

函数功能:

该挂钩处理过程是应用程序或库中定义的回调函数,它与函数SetWindowsHookEx搭配使用,系统在调用与任何挂钩类型和相关的挂钩处理过程之前调用它,系统传送关于被调用的挂钩的信息给DebugProc挂钩处理过程,此挂钩处理过程检查上述信息,并决定该挂钩是否被调用.  
  
类型HOOKPROC定义了指向此类回调函数的指针, DebugProc是应用程序定义或库定义的相应回调函数名的位置标识符.  
  
参数:  
nCode:

指示挂钩处理过程是否必须处理此消息.若nCode的值是HC\_ACTION,则挂钩处理过程必须处理此消息,若nCode之值小于0,此挂钩处理过程必须将此消息不加处理地传送给函数CallNextHookEx并返回其返回值.  
  
wParam:

指示将被调用的挂钩类型,此参数可以为以下值:  
WH\_CALLWNDPROC: 安装一个监视发送至目标窗口处理过程之的消息的挂钩处理过程,对该消息进行,详情参见CallWndProc挂钩处理过程.  
WH\_CALLWNDPROCRET :安装一个挂钩处理过程,以监视已刚被目标窗口处理过程处理过了的消息,详情参见 CallWndRetProc 挂钩处理过程.  
WH\_CBT :安装一个挂钩处理过程,接受对CBT应用程序有用的消息 ,详情参见 CBTProc 挂钩处理过程.  
WH\_DEBUG:安装一个挂钩处理过程以便对其他挂钩处理过程进行调试, 详情参见DebugProc挂钩处理过程.  
WH\_FOREGROUNDIDLE:安装一个挂钩处理过程,该挂钩处理过程当应用程序的前台线程即将进入空闲状态时被调用,它有助于在空闲时间内执行低优先级的任务.  
WH\_GETMESSAGE:安装一个挂钩处理过程对寄送至消息队列的消息进行监视,详情参见 GetMsgProc 挂钩处理过程.  
WH\_JOURNALPLAYBACK:安装一个挂钩处理过程,对此前由WH\_JOURNALRECORD 挂钩处理过程纪录的消息进行寄送.详情参见 JournalPlaybackProc挂钩处理过程.  
WH\_JOURNALRECORD:安装一个挂钩处理过程,对寄送至系统消息队列的输入消息进行纪录.详情参见JournalRecordProc挂钩处理过程.  
WH\_KEYBOARD:安装一个挂钩处理过程对击键消息进行监视. 详情参见KeyboardProc挂钩处理过程.  
WH\_MOUSE:安装一个挂钩处理过程,对鼠标消息进行监视. 详情参见 MouseProc挂钩处理过程.  
WH\_MSGFILTER:安装一个挂钩处理过程, 以监视由对话框、消息框、菜单条、或滚动条中的输入事件引发的消息.详情参见MessageProc挂钩处理过程.  
WH\_SHELL:安装一个挂钩处理过程以接受对外壳应用程序有用的通知, 详情参见 ShellProc挂钩处理过程.  
WH\_SYSMSGFILTER:安装一个挂钩处理过程,以监视由对话框、消息框、菜单条、或滚动条中的输入事件引发的消息.这个挂钩处理过程对系统中所有应用程序的这类消息都进行监视.详情参见 SysMsgProc挂钩处理过程.  
  
lParam:

指向一个DEBUGHOOKINFO结构,该结构包含希望被传送给相应挂钩处理过程的参数,为避免系统调用此挂钩,此挂钩处理过程必须返回一非0值,否则,此挂钩处理过程必须调用函数CallNextHookEx.  
  
备注:

一个应用程序通过调用函数 SetWindowsHookEx指定 WH\_DEBUG 挂钩类型与相应挂钩处理过程首地址来安装此挂钩处理过程.

### 

### 

### WH\_FOREGROUNDIDLE

函数原形:

**[cpp]** [view plain](http://blog.csdn.net/wangjieest/article/details/7107779" \o "view plain)[copy](http://blog.csdn.net/wangjieest/article/details/7107779" \o "copy)

1. **DWORD** ForegroundIdleProc(**int** code,**DWORD** wParam,**LONG** lParam);

函数功能:

该函数处理过程是应用程序或库定义的回调函数, 它与函数SetWindowsHookEx搭配使用,系统在前台线程进入空闲状态之前调用此函数.  
  
类型HOOKPROC定义了指向此类回调函数的指针, ForegroundIdleProc是应用程序定义或库定义的相应回调函数名的位置标识符.  
  
参数:  
Code:

指示此挂钩处理过程是否必须处理消息, 若nCode的值是HC\_ACTION,则挂钩处理过程必须处理此消息,若nCode之值小于0,则此挂钩处理过程必须将此消息不加处理地传送给函数CallNextHookEx并返回其返回值.

wParam:

未被使用.

lParam:

未被使用.  
  
返回值:

若nCode的值小于0,则此挂钩函数必须返回函数CallNextHookEx所返回的值;若nCode的值大于或等于0,我们极力推荐对函数CallNextHookEx进行调用,并返回其返回值.否则,其他安装了挂钩WH\_FOREGROUNDIDLE的应用程序将无法收到挂钩通知,从而导致错误的行为.若此挂钩处理过程没有调用函数CallNextHookEx,则返回值为0.

### 

### 

### WH\_GETMESSAGE

函数原形:

**[cpp]** [view plain](http://blog.csdn.net/wangjieest/article/details/7107779" \o "view plain)[copy](http://blog.csdn.net/wangjieest/article/details/7107779" \o "copy)

1. **LRESULT** CALLBACK GetMsgProc(**int** code,**WPARAM** wParam,**LPARAM** lParam );

函数功能:

该函数处理过程是应用程序或库定义的回调函数, 它与函数SetWindowsHookEx搭配使用,每当函数GetMessage从应用程序的消息队列中获得了一个消息时,系统就调用此函数,在将此获得的消息传送给相应窗口处理过程之前,系统将此消息传送给该挂钩处理过程.  
  
类型HOOKPROC定义了指向此类回调函数的指针, GetMsgProc是应用程序定义或库定义的相应回调函数名的位置标识符.  
  
参数:  
Code:

指示挂钩处理过程是否必须处理此消息.若nCode的值是HC\_ACTION,则挂钩处理过程必须处理此消息,若nCode之值小于0,此挂钩处理过程必须将此消息不加处理地传送给函数CallNextHookEx并返回其返回值.

wParam:

指示此消息是否已从消息队列中被删除,此参数可以是以下值之一:  
PM\_NOREMOVE :指出该消息尚未从消息队列中被删除(一个应用程序调用函数PeekMessage ,设置PM\_NOREMOVE 标志).  
PM\_REMOVE: 指出该消息已从消息队列中被删除. (一个应用程序调用函数 GetMessage或PeekMessage, 设置 PM\_REMOVE标志.)

lParam:

指向结构MSG,该结构包含了关于此消息的详细消息  
  
返回值:

若nCode的值小于0,则此挂钩函数必须返回函数CallNextHookEx所返回的值;若nCode的值大于或等于0,我们极力推荐对函数CallNextHookEx进行调用,并返回其返回值.否则,其他安装了挂钩WM\_GETMESSAGE的应用程序将无法收到挂钩通知,从而导致错误的行为.若此挂钩处理过程没有调用函数CallNextHookEx,则返回值为0.  
  
备注:

挂钩处理过程GetMsgProc可以检查和修改此消息,在此挂钩处理过程将控制返回给系统之后,函数GetMessage将此修改后的消息返回给最初调用他的应用程序,应用程序通过调用函数SetWindowsHookEx指定WH\_GETMESSAGE挂钩类型和相应挂钩处理过程的首地址来安装此挂钩处理过程.

### 

### WH\_JOURNALPLAYBACK

函数原形:

**[cpp]** [view plain](http://blog.csdn.net/wangjieest/article/details/7107779" \o "view plain)[copy](http://blog.csdn.net/wangjieest/article/details/7107779" \o "copy)

1. **LRESULT** CALLBACK JournalPlaybackProc(**int** code, **WPARAM** wParam, **LPARAM** lParam);

函数功能:

该函数处理过程是应用程序或库定义的回调函数, 它与函数SetWindowsHookEx搭配使用.典型地,一个应用程序通过调用此函数来对先前有JournalRecordProc挂钩处理过程纪录的一系列鼠标和键盘消息进行回放.一旦挂钩函数JournalPlaybackProc挂钩处理过程被安装,则通常的鼠标和键盘输入被禁止.  
  
类型HOOKPROC定义了指向此类回调函数的指针, JournalPlaybackProc是应用程序定义或库定义的相应回调函数名的位置标识符.

参数:  
  
code:

指示一个代码,被挂钩处理过程用来决定如何处理此消息,这个参数可以是以下值之一:  
  
HC\_GETNEXT 挂钩处理过程必须将当前的鼠标或键盘消息拷贝至由参数lParam 所指结构EVENTMSG中. HC\_NOREMOVE 一个正用wRemoveMsg为参数调用了 PeekMessage的应用程序对PM\_NOREMOVE标志进行设置,指示此消息在经过函数PeekMessage的处理之后,并不从消息队列中被删除.  
  
HC\_SKIP此挂钩处理过程必须准备拷贝下一个鼠标或键盘消息至由参数lParam 所指定的结构EVENTMSG 中去,依据收到的代码HC\_GETNEXT ,此挂钩处理过程必须将消息拷贝至结构EVENTMSG 中去. HC\_SYSMODALOFF 一个的有系统模式对话框已被销毁,此挂钩处理过程必须恢复对此消息的回调.  
  
HC\_SYSMODALON 一个的有系统模式对话框正在被显示,直至此对话矿被销毁之前,此挂钩处理过程必须停止对消息的回调.  
  
若nCode的值小于0,则此挂钩函数必须返回函数CallNextHookEx所返回的值.

wParam:

指示一个NULL(0)值

lParam:

指向一个EVENTMSG结构,该结构反映了正被此挂钩处理函数处理的消息,这个参数仅当参数nCode的值是HC\_GETNEXT时才有效.  
  
返回值:

为使系统在处理此消息之前先等待一段时间,返回值必须为一个以时钟计的时间量,该时间量指出了系统的等待时间长短.(此时间量可通过计算当前输入的消息与上一个输入消息的time成员的差别获得),若希望立即处理此消息,返回值应为0,此返回值仅当挂钩代码为HC\_GETNEXT时才被使用,否则,此返回值被忽略.  
  
备注:

一个JournalPlaybackProc挂钩处理过程应当将输入消息拷贝给lParam参数,此消息必须是在先前使用JournalRecordProc 挂钩处理过程时被纪录过的,在使用JournalRecordProc挂钩处理过程时,不应对此消息进行修改,为了多次获得相同的消息,此挂钩处理过程可以多次以HC\_GETNEXT为参数被调用.  
  
若nCode的值为HC\_GETNEXT而返回值大于0,则系统按返回值指定的毫秒值进入睡眠状态,当系统开始继续执行时,它会以HC\_GETNEXT为参数nCode的值再次调用次挂钩处理过程以获得相同的消息,此次调用JournalPlaybackProc的返回值为0,否则,系统又将回到睡眠状态,并以此次调用JournalPlaybackProc返回的毫秒数为睡眠时间,睡眠时间到后再重复上述操作.这就使得系统表现为被挂起.  
  
与其余大多数全局挂钩处理过程不同,挂钩处理过程JournalRecordProc和JournalPlaybackProc总是在设置了该挂钩的线程上下文中被调用.  
  
在此挂钩处理过程将控制返回给系统后,此消息继续被处理.若nCode的值为HC\_SKIP,则此挂钩处理过程必须准备在它下一次被调用时返回下一个纪录了的事件消息,安装JournalPlaybackProc挂钩处理过程通过调用函数SetWindowsHookEx,指定WH\_JOURNALPLAYBACK挂钩类型及相应挂钩处理过程首地址来实现,如果用在日志回调过程中按下<Ctrl+Esc>或<Ctrl+Alt+Del>组合键,系统将终止回调,取消此日志回调过程,并寄送一个WM\_CANCELJOURNAL消息给日志应用程序.  
  
若此挂钩处理过程返回的消息在WM\_KEYFIRST 或WM\_KEYLAST的范围内,则以下情形发生.  
  
结构EVENTMSG的paramL成员指示按下的虚拟键编码值,结构EVENTMSG的paramH成员指示扫描吗,这种情况下无法指示出重复按键的次数,这个事件仅表示一个按键事件.

### 

### 

### WH\_JOURNALRECORD

函数原形:

**[cpp]** [view plain](http://blog.csdn.net/wangjieest/article/details/7107779" \o "view plain)[copy](http://blog.csdn.net/wangjieest/article/details/7107779" \o "copy)

1. **LRESULT** CALLBACK JournalRecordProc(**int** code,**WPARAM** wParam,**LPARAM** lParam);

函数功能:

该函数处理过程是应用程序或库定义的回调函数, 它与函数SetWindowsHookEx搭配使用,此函数纪录系统从系统消息队列中删除的消息,此后,一个应用程序可以通过使用挂钩处理过程JournalPlaybackProc对此消息进行回调.  
  
类型HOOKPROC定义了指向此类回调函数的指针, JournalRecordProc是应用程序定义或库定义的相应回调函数名的位置标识符.  
  
参数:  
Code:

指示如何处理此消息,此参数可以是以下值之一:  
  
HC\_ACTION: 参数LPARAM指向结构 EVENTMSG ,该结构包含从系统消息队列中删除了的消息的有关信息,此挂钩处理过程必须纪录下此结构的内容,将其拷贝到一个缓冲区或文件中.  
HC\_SYSMODALOFF: 一个系统的有模式对话框正在被销毁,此挂钩处理过程必须进行纪录.  
HC\_SYSMODALON:一个系统的有模式对话框正在被显示,此挂钩处理过程必须停止纪录.  
若nCode之值小于0,此挂钩处理过程必须将此消息不加处理地传送给函数CallNextHookEx并返回其返回值.  
  
wParam:

指定为NULL(0).  
  
lParam:

指向结构EVENTMSG ,该结构包含将被纪录的消息.  
  
返回值:

返回值被忽略.  
  
备注:一个JournalRecordProc挂钩处理过程必须拷贝但不能修改此消息,在次挂钩处理过程控制交还给系统之后,此消息继续被处理.  
  
安装一个JournalRecordProc挂钩处理过程是通过调用函数SetWindowsHookEx指定WH\_JOURNALRECORD挂钩类型和相应挂钩处理过程的首地址来安装此挂钩处理过程.过程并非只能存在于动态链接库中,一个应用程序自身也可拥有自己的JournalRecordProc挂钩处理过程.  
  
与其余大多数全局挂钩处理过程不同,挂钩处理过程JournalRecordProc和JournalPlaybackProc总是在设置了该挂钩的线程上下文中被调用.一个安装了JournalRecordProc挂钩处理过程的应用程序该对虚拟键VK\_CANCEL进行监视(在大多数键盘上,虚拟键VK\_CANCEL被实现为组合键< CTRL+BREAK>),此虚拟键应被应用程序解释为用户希望终止进行日志纪录的信号,而应用程序对此进行响应,结束相应的纪录序列,并删除挂钩处理过程JournalRecordProc,在这里,删除挂钩处理过程是很重要的,此举使得应用程序免于因为在一个挂钩处理过程内部被挂起而将系统锁住.  
  
这种作为终止日志纪录信号的角色使得组合键<CTRL+BREAK>本身无法被纪录,而组合键<CTRL+C>并不充当日志纪录信号的角色,所以可以被纪录,这里还有两种组合键是不能被纪录的:即<CTRL+ESC> 和<CTRL+ALT+DEL>,这两组组合键导致系统终止所有的日志活动(纪录或回调),删除所有的日志挂钩,并将消息WM\_CANCELJOURNAL寄送给日志应用程序.

### 

### 

### WH\_KEYBOARD

函数原形:

**[cpp]** [view plain](http://blog.csdn.net/wangjieest/article/details/7107779" \o "view plain)[copy](http://blog.csdn.net/wangjieest/article/details/7107779" \o "copy)

1. **LRESULT** CALLBACK KeyboardProc(**int** code,**LPARAM** lParam);

函数功能:

该函数是应用程序或库文件定义的回调函数, 它与函数SetWindowsHookEx搭配使用,当应用程序调用函数 GetMessage 或PeekMessage 并且当前恰好有一个键盘消息 (WM\_KEYUP or WM\_KEYDOWN) 将要被处理时,系统就调用此挂钩处理过程.  
  
类型HOOKPROC定义了指向此类回调函数的指针, KeyboardProc是应用程序定义或库定义的相应回调函数名的位置标识符.

参数:

ncode:

指定一个代码,该代码被挂钩处理过程用于决定如何处理此消息,此参数可以为以下值之一:  
  
HC\_ACTION 参数 wParam 和 lParam包含一个击键消息的信息.  
  
HC\_NOREMOVE 参数wParam and lParam包含一个击键消息的信息,并且此击键消息尚未从消息队列中被删除(一个应用程序调用函数PeekMessage 并设置了PM\_NOREMOVE 标志).  
  
若nCode的值小于0,则此挂钩函数必须不加处理地传送给函数CallNextHookEx,并返回其返回值.  
  
  
wParam:

指示产生此击键消息的虚拟键代码.

lParam:

指示重复次数,扫描码,扩展标志,上下文代码,此前的键状态标志和变化状态标志,此参数可以是以下值的组合值:  
  
0-15位:指示重复次数,此值纪录力偶由于用户继续摁键引发的击键重复次数.  
  
16-23位:指示扫描码,此值依赖于键盘生产厂家.  
  
第24位:指示此键是否为扩展键,比如功能键或数字小键盘上的键,当该键为扩展时,其值为1,否则为0.  
  
25-28位:保留未用.  
  
第29位:上下文代码.若Alt键被摁下,则此值为1,否则为0.   
  
第30位:指示此前的键状态,若在此消息被发送之前该键是摁下的,其值为1.若此前该键未被摁下,则其值为0.  
  
第31位:指示变化状态,若此键正在被摁下,则其值位0. 正在被释放其值为1.  
  
欲获得更多关于此lParam参数的信息,请参见 Keystroke Message Flags.

返回值:

若nCode的值小于0,则此挂钩函数必须返回函数CallNextHookEx所返回的值;若nCode的值大于或等于0,并且此挂钩处理过程并未对此消息进行处理,我们极力推荐对函数CallNextHookEx进行调用,并返回其返回值.否则,其他安装了挂钩WM\_KEYBOARD的应用程序将无法收到挂钩通知,从而导致错误的行为.若此挂钩处理过程对此消息进行了处理,它应返回非0值以避免系统将此消息传递给挂钩键上的其他处理过程或目标窗口的处理过程.  
  
备注:

一个应用程序通过调用函数SetWindowsHookEx指定WM\_KEYBOARD挂钩类型和相应挂钩处理过程的首地址来安装此挂钩处理过程.

### 

### 

### WH\_MSGFILTER

函数原形:

**[cpp]** [view plain](http://blog.csdn.net/wangjieest/article/details/7107779" \o "view plain)[copy](http://blog.csdn.net/wangjieest/article/details/7107779" \o "copy)

1. **LRESULT** CALLBACK MessageProc(**int** code,**WPARAM** wParam,**LPARAM** lParam);

函数功能:

挂钩处理过程是应用程序或库中定义的回调函数,它与函数 SetWindowsHookEx搭配使用.当一个对话框、消息框、菜单条、或滚动条中的输入事件发生，且由此引发的消息尚未被处理之前,系统对此挂钩处理过程进行调用.此函数可以对一个特定应用程序或所有应用程序的对话框、消息框、菜单条、或滚动引发的条消息进行监视.  
  
类型HOOKPROC定义了指向此类回调函数的指针. MessageProc 是库中相应回调函数名的位置标志符.  
  
参数:  
nCode:

指示产生此消息的输入事件类型.此参数可以是以下值之一:  
MSGE\_DDEMGR:当动态数据交换管理库(Dynamic Data Exchange Management Library (DDEML)正在等待一个同步处理的结束时产生此事件,关于DDEML的详情,参见Dynamic Data Exchange Management Library .  
MSGF\_DIALOGBOX: 输入事件由一个消息框或者对话框产生.  
MSGF\_MENU: 输入事件由一个菜单条产生.  
MSGF\_SCROLLBAR: 输入事件由一个滚动条产生.  
MSGF\_NEXTWINDOW: 输入事件由于用户摁下组合键<Alt+Tab>以激活另一个窗口而产生.  
若nCode的值小于0,则此挂钩处理过程必须将该消息不加处理地传送给函数 CallNextHookEx 并返回其返回值.  
  
wParam:

为NULL(0).

lParam:

指向MSG结构的[指针].  
  
返回值:

若nCode的值小于0,则此挂钩处理过程必须返回CallNextHookEx 的返回值.若nCode的值大于或等于0,并且此挂钩处理过程未对该消息进行处理,则调用函数CallNextHookEx 并返回其返回值是被推荐的.否则,其他安装了WH\_MSGFILTER挂钩的应用程序将无法收到此挂钩通知,并可能由此导致错误的行为.若此挂钩处理过程处理了此消息,它应返回一个非0值以避免系统再将此消息传送给挂钩链上的其他挂钩处理过程或目标窗口处理过程.  
  
备注:

一个应用程序通过调用函数SetWindowsHookEx指定WH\_MSGFILTER挂钩类型和相应挂钩处理过程的首地址来安装此挂钩处理过程.  
若一个应用程序使用了DDEML并实行同步处理,而且要求必须在消息被分派前处理之,则一定要使用WH\_MSGFILTER挂钩.

### 

### WH\_MOUSE

函数原形:

**[cpp]** [view plain](http://blog.csdn.net/wangjieest/article/details/7107779" \o "view plain)[copy](http://blog.csdn.net/wangjieest/article/details/7107779" \o "copy)

1. **LRESULT** CALLBACK MouseProc(**int** nCode, **WPARAM** wParam, **LPARAM** lParam);

函数功能:

该函数是应用程序或库文件定义的回调函数, 它与函数SetWindowsHookEx搭配使用,当应用程序调用函数 GetMessage 或PeekMessage 并且当前恰好有一个鼠标消息将要被处理时,系统就调用此挂钩处理过程.  
  
类型HOOKPROC定义了指向此类回调函数的指针, MouseProc是应用程序定义或库定义的相应回调函数名的位置标识符.  
  
参数:  
  
nCode:

指示一个代码被挂钩处理过程用于决定如何处理此消息,此参数可以是以下值之一:  
HC\_ACTION 指示参数wParam 和lParam 包含了关于鼠标消息的信息.  
HC\_NOREMOVE 指示参数 wParam和lParam 中包含了关于鼠标的信息,而且此鼠标消息尚未从消息队列中被删除(一个应用程序调用函数PeekMessage并设置了 PM\_NOREMOVE标志).  
若nCode的值小于0,则此挂钩函数必须不加处理地传送给函数CallNextHookEx,并返回其返回值.

wParam:

指示鼠标消息的标识符,

lParam:

指向MOUSEHOOKSTRUCT 结构的[指针].

返回值:

若nCode的值小于0,则此挂钩函数必须返回函数CallNextHookEx所返回的值;若nCode的值大于或等于0,并且此挂钩处理过程尚未对此消息进行处理,我们极力推荐对函数CallNextHookEx进行调用,并返回其返回值.否则,其他安装了挂钩WM\_MOUSE的应用程序将无法收到挂钩通知,从而导致错误的行为.若此挂钩处理过程对此消息进行了处理,它应返回非0值以避免系统将此消息传递给目标窗口的处理过程.

备注:

一个应用程序通过调用函数SetWindowsHookEx指定WM\_MOUSE挂钩类型和相应挂钩处理过程的首地址来安装此挂钩处理过程.此挂钩处理过程一定不能安装JournalPlaybackProc回调函数.