Hook

HHOOK SetWindowsHookEx(        
  
    int *idHook*,

    HOOKPROC *lpfn*,

    HINSTANCE *hMod*,

    DWORD *dwThreadId*

);

int  Idhook : Just like WH\_KEYDOWN,WH\_MOUSE ….. WH represent “Windows Hook”

This parameter is the type of hook procedure being install.

HOOKPROC lpfm :A procedure monitors messages before system sends them to destination windows procedure ,like :

//Keyboard Hook Procedure

LRESULT CALLBACK MyKeyboardProc(int code,

WPARAM wParam,

LPARAM lParam

)

{

if (VK\_SPACE == wParam)

{

//It is going to occur for 2 times ,Just from WM\_KEYDOWN and WM\_CHAR ,WM\_CHAR is produced when DefWndProc is running,TranslateMessage function too.

//MessageBox(NULL, \_T("123"), \_T("456"), MB\_OK);

//return 1 means this message did not being process

return 1;

}

//this can process message to next hook

return CallNextHookEx(g\_hKeybord,WH\_KEYBOARD,wParam,lParam);

}

HINSTANCE *hMod :* If u set a globle hook , u must get DLL’s hInstance ,else inner hook just get NULL

DLL’s hInstance:

methed1: u can get it through DLLmain

method2: use function GetModuleHandle(“Hook”);

DWORD dwThreadId: If u set a globe hook, u must fill 0 ,else inner hook just fill CurrentThreadId

(GetCurrentThreadID())

Return value:

If the function succeeds, the return value is the handle to the hook procedure.

If the function fails, the return value is NULL. To get extended error information, call GetLastError.

UnHook:

BOOL UnhookWindowsHookEx( HHOOK hhk

);

Eg:

Globle HHOOK g\_hKeyboard;

LRESULT CALLBACK MyKeyboardProc(int code,

WPARAM wParam,

LPARAM lParam

)

{

//u can dothing to this Hook

}

g\_hKeyboard = SetWindowsHookEx(WH\_KEYBOARD,

MyKeyBoadProc,

Null or (hInstance),

GetCurrentTreadID() or 0

);

UnhookWindowsHookEx( g\_hKeyboard );