文章编号: 1001-2486 (2002) 01-0077-04

Hook 技术及其在软件研发中的应用*

邱建雄
(长沙大学计算机科学与技术系, 湖南 长沙 410003)

摘要: 介绍了Hook 技术的基本概念和方法, 对常用的几类Hook 技术重点进行了分析, 并举例说明了在进行软件研发时, 这些 Hook 技术可以应用的场合, 最后介绍了在电子字典, 带滚轮的鼠标驱动程序等实际软件开发项目过程中, 如何应用 Hook 技术的详细方法和步骤。

关键词: hook; 视窗编程
中图分类号: TP311.52 文献标识码: A

Hook Technology and Its Applications in Software Development

QIU Jian-xiong
(Department of Computer, ChangSha University, ChangSha 410003, China)

Abstract: This paper first introduces the basic concepts and methods of hook technology. Then analyzes some major hooks and their usage in software development. Finally, how to use hook technology in the processes of developing the Electronic Dictionary and the Wheel Mouse driver is described in detail.

Key words: hook; windows programming

众所周知, Windows 是基于"事件 (消息) 驱动"机制的图形界面操作系统, 每个应用程序都有一个窗口与之对应, 用户与该应用程序的交互都限制在这个窗口范围内, 用户的键盘、鼠标等输入都被系统转化成为所谓的"事件", 发送"消息"给相应的应用程序, 而应用程序的任务就是根据这些消息进行响应, 处理相应的鼠标点击、键盘输入等事件。

Hook 技术是指Windows 系统中一个具有强大威力的特性, 在某个应用程序或系统接受相应的消息, 进行处理之前, 其它应用程序可以拦截正要传入的消息, 先行进行处理。它不仅可以拦截某一个线程的事件, 还可以拦截整个 windows 系统的事件。这有些类似于在 DOS 系统中编程, 采取改变中断向量地址, 设置新的中断向量的技术, 当系统调用这个中断时, 会先进入新的中断服务程序, 然后返回再调用原来的中断服务程序。这种技术可以用于许多实际应用软件的研发之中。例如: 在电子字典的研发中, 实现"屏幕取词"技术; 在带滚轮的网络鼠标的驱动程序的研发中, 用于拦截用户滚轮滚动事件, 从而实现应用程序中的屏幕滚动这个动作。

1 Hook 技术的基本实现方法

1.1 Local Hook 和 Remote Hook

Local Hook 是拦截原本预定给"该进程中设定此 Hook 之线程"的事件。挂上 Local Hook 比较容易, 对系统的影响很小。而Remote hook 则是拦截原本要给"其它进程之线程"的事件。Remote Hook 又有两种形式: 一种是针对特定线程, 所拦截的是"其它进程中的某个特定的线程"的事件; 另一种是针对整个系统的, 所拦截的是"整个系统中所有进程地址空间中的所有线程"的事件。Remote Hook 对系统的影响较大, 特别是针对整个系统的 Remote Hook, 因为它要拦截系统中所有的事件, 所

* 收稿日期: 2001-10-12
基金项目: 国家部委基金项目资助 (19.6.2 (3)); 湖南省教育厅科研项目资助 (00C022)
作者简介: 邱建雄 (1961—), 男, 讲师, 硕士。

以可能会影响 Windows 系统的效率和稳定性。

1.2 挂上和卸除 Hook 的方法

Windows 共有 14 种不同的 Hook，每一种都可以是 Local 或 Remote Hook，它们的挂上和卸除的方法都是一样的。

应用程序调用 SetWindowsHookEx 以挂上一个 Hook：

```
HHOOK SetWindowsHookEx ( int idHook, HOOKPROC hkprc, HINSTANCE hmod, DWORD dwTHreadID);
```

其中：idHook 用来表示要挂上那一种 hook，我们后面还要讨论；

hkprc 指 filter 函数的地址；

hmod 是 DLL 模块的代码，存放 filter 函数的地方，如果要挂上 local Hook，hmod 应该为 NULL；

dwThreadID 是所希望挂上该 Hook 的线程之代码，如果想拦截系统中所有程序的消息，则该参数应该设为 0。

每当有事件发生，并且与所挂上的 hook 有关联时，Windows 会呼叫在 SetWindowsHookEx 函数中所指定的 filter 函数（hook 函数），其原型如下所示：

```
LRESULT WINAPI FilterFunc ( int nCode, WPARAM wParam, LPARAM lParam);
```

其中：nCode 代表动作的形态，类似于消息，即 filter 函数该做什么处理，而 wParam 和 lParam 的含义则随着 nCode 的不同而不同。

当程序不再需要 Hook，可以利用 UnhookWindowsHookEx 函数来卸除之：

```
BOOL UnhookWindowsHookEx ( HHOOK hhook ) ;
```

其中：hhook 是要卸除的 hook 的句柄（handle）。

1.3 Hook 的串链结构

当许多程序都安置了某种 hook 时，就会形成一个 filter 函数链，一旦这个特定的事件发生，Windows 就只会调用其最新挂上的 filter 函数。若要求原来的 filter 函数也能够执行，必须在最新的 filter 函数上加一些处理，调用 CallNextHookEx 函数，以保证在 filter 函数链中的下一个 filter 函数被调用。

例如：假如程序 A 挂上了一个系统级的 WH_KEYBOARD hook，每当有任何线程取得键盘消息时，Windows 都会调用该 filter 函数。如果程序 B 又挂上了一个系统级的 WH_KEYBOARD hook，则当有键盘消息时，Windows 不会调用程序 A 的 filter 函数，而是只调用程序 B 的 filter 函数，若要求 A 的 filter 函数也被调用，则必须在 B 的 filter 函数中进行适当的处理。

2 Hook 的分类及其应用

下面将介绍在应用软件的研发中最常用到的几种 Hook，并说明其特性、用法和应用的场合。

2.1 H_CALLWNDPROC，WH_CALLWNDPRORET，WH_GETMESSAGE

当一个线程调用 SendMessage 函数时，Windows 会先检查看是否有一个 WH_CALLWNDPROC Hook 被挂在这个线程上，如果有，则调用其 filter 函数．由于 Windows 环境中消息的发送非常频繁，所以挂上该 hook 会对 Windows 操作系统的效率产生很大的影响，通常可用于软件调试（Debug）的目的。例如，Visuial C+ + 中的除错工具 Spy 就是利用了该 hook．它允许拦截送往某一个视窗的消息和其返回值，并将结果显示于自己的视窗工作区内。可用于监视送给某一个视窗或系统中所有视窗的消息。

2.2 WH_KEYBOARD，WH_MOUSE，WH_HARDWARE

该三个 Hooks 用于拦截键盘、鼠标或其它硬件的消息，当某个线程调用 GetMessage 或 PeerMessage 并得到一个硬件消息时，系统就会调用相应的 Hook 来处理它。

例如：Windows 应用程序中的热键（Hot Key）的作用原理，就是依靠 WH_KEYBOARD Hook 来随时检验按键的状态信息，并将所有按键动作录制成一个宏，若挂上一个系统 hook，即使其他程序正在前台工作，它也可以监视所有的按键信息，从而启动相应的热键动作。又例如，若想开发教导使用者如何使用 Windows 的软件，可以在软件中挂上一个 WH_MOUSE Hook，即使在程序没有得到鼠标的

捕捉权时，程序也能够检测到鼠标移动和鼠标的按键状态。当使用者将鼠标移动到屏幕上某个元素，如系统选单、标题栏、滚动轴时，由 filter function 判断鼠标目前的位置，并显示一些说明文字到说明视窗中，同时不会与正常的鼠标动作的处理有任何冲突。

2.3 WH_JOURNALRECORD，WH_JOURNALPLAYBACK

这两种最常用的 hook 主要用于拦截低层的硬件输入事件。它们都是 local system-wide hooks，其 filter 函数不需写在 dll 之中。当系统中任何一个线程发出一个硬件输入消息时，系统会唤醒挂上 journal playback hook 的那个线程，并且执行其 filter 函数。同理，当系统中任何一个线程从虚拟输入队列中取到一个硬件输入消息时，系统会唤醒挂上 journal record hook 的那个线程，并且执行其 filter 函数。

2.4 WH_CBT

这个 hook 通常用来提供 Computer-Based Training (CBT)，特别适用于计算机辅助教学或训练软件的研发当中。挂上该 hook 之后，Windows 在产生、摧毁、极大化、极小化、移动、缩放一个视窗时之前，或是在令一个视窗成为活动视窗之前，调用对应的 filter 函数。Windows 同时也会在完成一个系统命令之前，或是在线程的硬件输入队列中移出鼠标或键盘事件之前，或是在设定输入焦点之前，或是在收到 WM_QUEUESYNC 消息之前，调用对应的 filter 函数。这个 hook 可以让程序监视使用者的演变发展，可以自动执行某些工作而辅助使用者。

例如：应用软件可以在使用者刚开始接触它时，给予适当的指导，使其快速地学会使用该软件。程序可以告诉使用者如何操作本软件，而 CBT hook 可以用来监视使用者是否成功地执行了这些操作步骤。再如，假如当使用者想要学习在字处理软件中如何将一个文件格式化，这时需要自动输入一段文字后，才显示文件的格式化，这时 WH_CBT hook 可以挂上 WH_JOURNALPLAYBACK hook，强迫产生按键动作，输入文字，等到所有的按键都播放完毕，再卸除 WH_JOURNALPLAYBACK hook。

3 Hook 技术应用实例

3.1 电子字典

电子字典在线翻译的主要功能是将目前鼠标的位置上的文字中翻译或解释说明出来，并显示在旁边的“说明视窗”中，“屏幕取词”(即找出当前的文字内容) 的实现方法中就是利用了 WM_MOUSE hook 技术。

首先我们可用 SetWindowsHookEx () 安装 WM_MOUSE hook；当用户在屏幕上移动鼠标时，系统就会调用鼠标 filter 函数 FilterMouseProc；进入 FilterMouseProc 后，我们可以获得鼠标当前的坐标 (x，y)，进而设置对 TextOut ()、ExtTextOut () 等的跟踪程序，用 InvalidateRect () 告诉 Windows 该点 (x，y)“失效”；系统会发出 WM_PAINT 消息，指示该点 (x，y) 处的应用程序重绘“失效”的区域。负责绘制该点 (x，y) 的应用程序在接受到 WM_PAINT 消息后，就会有机会调用 TextOut ()、ExtTextOut () 等函数。调用的函数被拦截进入跟踪程序，设置好了的跟踪程序截获了该次调用，从应用程序的堆栈中取出该点 (x，y)“文字”的指针；从应用程序的数据段中将“文字”指针的内容取出，即完成了一次“屏幕抓字”；退出跟踪程序，返回到鼠标 hook FilterMouseProc；在 FilterMouseProc 中解除对 TextOut ()、ExtTextOut () 的跟踪；退出 FilterMouseProc 鼠标 hook 程序，控制权交还给了 Windows 系统。用户在屏幕上移动鼠标时，又开始了下一次“屏幕取词”过程。

3.2 带滚轮鼠标 (Wheel Mouse) 驱动程序

滚轮驱动功能的实现实际由两大部分完成，即低级驱动程序和高级驱动程序。低级驱动程序，如 Win95/Win98 系统中的鼠标 vxd 程序，NT 系统中的 Mouse Class Driver 与 Port Driver，主要功能是对鼠标硬件进行正确的初始化，使之从加电初态的基本模式变为具有滚轮功能的高级模式，并设置鼠标硬件中断处理程序。此后，点击鼠标按键或滚动滚轮，将产生硬件中断，激活鼠标硬件中断处理程序。鼠标硬件中断处理程序负责读取鼠标寄存器内容，清除中断，然后根据读取的寄存器内容，解析出按键点击，X-Y 轴向移动及上下/左右滚轮滚动的信息，最后按照系统的要求，形成系统消息并发送出去。由于篇幅所限，在此不作详细介绍。

高级驱动程序负责实现用户窗体的滚动行为。有过 Windows 编程经验的都知道，要控制自己应用程序中某个带滚动条窗体的滚动行为，只需获取该窗体的句柄，然后通过该句柄向该窗体发送滚动消息，如 WM_HSCROLL 或 WM_VSCROLL，分别控制窗体横向滚动或纵向滚动。高级驱动程序实现原理与此类似，当收到滚动消息后，它调用 WindowFromPoint 函数获取当前光标下活跃窗体的句柄，然后将滚动信息转换成对应的 WM_HSCROLL 或 WM_VSCROLL 消息，直接向目的窗体发去。系统鼠标消息虽包含滚动编码信息，但与 WM_HSCROLL 和 WM_VSCROLL 消息不同，高级驱动程序必须负责这一消息的转换。实际应用中，根据目的窗体的性质，如滚动条是捆绑在某个窗体上，在目的窗体的选取和消息发送上会略有不同，但原理是一样的。在此不作赘述。

问题的关键是，高级驱动程序如何获得其它线程窗体下的鼠标消息。这就要用到 Hook 技术，特别是 Remote Hook 技术。按照 Remote Hook 技术要求，我们实现一个 Dll 模块，称之为 Mouse Hook 模块。高级驱动程序首先通过 Mouse Hook 模块的某个入口程序，如 SetHWnd (HWND hWnd)，将自己的主窗体句柄传递给它，由它保留于全局共享数据区 (Dll 特有特性)，然后调用 Mouse Hook 模块的另一入口 (该入口程序内调用 SetWindowsHookEx)，设置 Hook 程序——HookProc。HookProc 必须位于 Mouse Hook 模块内。HookProc 设置成功后，低级 Mouse Driver 所产生的鼠标消息在到达目的窗体前，必先经过 HookProc，在此可以分离出感兴趣的消息，如滚轮消息，然后通过高级驱动程序主窗体句柄将这一消息传送给高级驱动程序，由它负责实现目的窗体的滚动行为。

4 结 论

Hook 技术是一种 Windows 环境下的系统软件开发技术，具有强大的威力，能够处理许多软件研发中看似难以解决的问题，可应用于许多实际应用软件研发过程之中，值得深入地进行研究与探讨，以提高软件研发技术和水平，并推广到实际应用软件的研发项目中。

参考文献:

[1] Charles Petzold. Programing Windows [M]. Microsoft Press , 1998.

[2] Jeffrey Richter. Advanced Windows [M]. Microsoft Press, 1998.

[3] Jeffrey Richter , Jonathan Locke. Windows 95: A Developer's Guide [M]. Microsoft Press. 1995.

[4] Matt Pietrek. Window95 System Programming SECRETS [M]. Microsoft Press 1995.