

**研 究 生 课 程 论 文**

**(2018-2019学年第二学期)**

**大作业报告**

**研究生：陈家浩**

**提交日期： 2019年7月8日 研究生签名：**

|  |  |  |  |
| --- | --- | --- | --- |
| **学 号** | **201820132886** | **学 院** | **计算机科学与工程学院** |
| **课程编号** | **S0812019** | **课程名称** | **软件测试与质量控制** |
| **学位类别** | **硕士** | **任课教师** | **陆璐** |
| **教师评语：** | | | |
| **成绩评定： 分 任课教师签名： 年 月 日** | | | |

**Windows平台下应用程序录制回放软件的设计与实现**

1. **背景介绍**

Windows 操作系统是基于消息传递、事件驱动的运行机制，消息就是为了描述事件特征而产生的格式化信息，消息产生后被追加到消息队列中，并被分发到对应的应用程序窗口进行处理。此外，Windows提供了一种系统编程接口——Windows hooks(Windows 钩子) ，通过这个接口可以截获并处理将要发送到应用程序的各种消息，系统钩子因此获取了系统控制权。

钩子机制的核心是与钩子相关联的钩子函数, 钩子函数是一个能响应特定消息的回调函数, 它是一个由用户定义其实现, 但由系统在特定的时机来调用的函数。它用以监视系统或某一特定类型的事件, 这些事件可以是与某一特定线程关联的, 也可以是系统中所有线程的事件。Windows 有多种类型的钩子, 每种钩子可以截获并处理相应的消息, 每当特定的消息发出, 在到达目的窗口之前, 系统将控制权转向了钩子函数, 这样钩子函数先行得到对此消息的控制权。这种控制权的转移对目标窗口来说是透明的。获取控制权的钩子函数便可对消息进行加工处理或者不作处理而继续传递该消息, 还可以强制结束传递消息。通过安装不同类型的钩子, 就可以对系统中发生的不同类型消息( 事件) 进行监视、拦截、修改和转向等操作。例如, 可利用钩子来记录输入的中文字符、监控计算机的使用 、截获应用程序口令等。Windows hooks 中共有15 种不同类型的钩子, 其中有两种比较特殊的钩子——录制钩子（WH\_JOURNALRECORD）和回放钩子（WH\_JOURNALPLAYBACK），本系统在实现过程中也充分利用了这两种特殊钩子来达成对windows平台下应用程序的基本操作录制与回放这一功能。

现阶段的软件回放应用一般都通过录屏软件，即开启录制视频的方式重现用户操作，这带来的是巨大的系统开销和冗长的录制文件; 也有少量的软件针对各自程序的数据结构、I/O 交互特性编写回放代码，但由于其专门性，因而带来了大量额外的设计、编码、调试时间。

为了实现Windows平台下的录制与回放操作，采用了一种基于 Windows Hook 的通用的录制/回放机制，充分利用 Windows 系统中事件驱动、消息队列和消息分发的特点，能够在用户操作的同时，动态地监视、截获、记录用户的操作动作; 并在需要回放时读取被持久化保存的消息，通过 Windows Hook 分发到被录制窗口，从而达到高效、透明的录制效果和完整再现用户操作的回放效果。

1. **系统框架**
   1. **系统设计架构**

本系统使用守护进程的方式启动录制、回放，考虑以下几方面的内容：

1. 能够独立于目标进程，可通过配置文件指定具体窗口，不需要对接受控制的程序进行修改。
2. WH-JOURNALRECORD 类型的 Hook 是全局类型，能够拦截所有消息，为了保证录制到目标窗口消息，需要强制最大化、前端化被录制的目标窗口; WH-JOURNALPLAYBACK 类型的 Hook 情况类似，特别的是，往往在回放时记录下的目标窗口句柄已经不存在了，所以必须强制使目标窗口在前端接受控制。

系统架构如图2-1所示。

Windows系统

消息循环

|  |
| --- |
| 消息1 |
| 消息2 |
| 消息3 |
| … |

分发消息

|  |
| --- |
| 消息处理程序① |
| 消息处理程序② |
| 消息处理程序③ |
| … |

拦截消息

Windows消息队列

远程监控

归还消息

Hook回调函数

DefWindowPro

守护进程

持久化存储

窗口过程

图2-1 守护进程系统架构示意图

**2.2. 系统主要模块及功能**

该系统由守护进程完成了记录/回放的整个流程，守护进程按照响应顺序共有三个主要模块。

1) 进程监控系统主体框架，经过一定的时间间隔查询系统的当前进程快照，当发现具有目标进程时，激活自身状态。用户此时可以开始记录目标进程窗口或选择回放动作序列。

2) Hook 的安装 / 卸载 / 响应 开始记录 / 回放窗口时，安装相应的 Hook; 在完成记录/回放时，删除相应的 Hook。并在记录/回放过程中，完成 Hook响应动作。

3) 持久化存取；将 Hook 拦截的消息从结构体EVENTMSG 持久化存储入磁盘文件，以供回放时读取。

按设计目标，该系统能够完成以下功能:

1) 可配置目标进程信息，目标进程实际录制后应立即开始记录;

2) 录制 / 回放时，最大化、前端化目标进程窗口，并在录制过程中保存此状态;

3) 能够记录 / 回放鼠标轨迹、点击、键盘动作、数据输入等;

4) 能够完成数据的持久化保存;

1. **关键技术解析**

**3.1 使用钩子的一般步骤**

Windows hooks 的使用可分为以下三个步骤：

1. 安装钩子。利用Windows API 函数SetWindowsHookEx()安装钩子, 该函数的原型为:

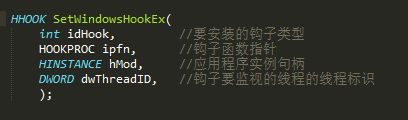


图3-1 函数SetWindowsHookEx()原型

用户户点击开始录制后，需要为系统安装录制 Hook，安装上述录制 Hook 后，每当消息队列分发消息，就会自动触发回调函数 JournalRecordProc()；

（2）编写钩子函数。该过程是一个回调函数(Callback Function)，将这个函数的地址(即函数名) 传递给SetWindowsHookEx()的第二个实参。

（3）卸载钩子。调用Windows API 函数UnhookWindowsHookEx()来卸载前面安装的钩子。用户点击停止录制后，则卸载当前正在运行的Hook:UnhookWindowsHookEx( hHook) ;

**3.2 录制/回放钩子**

类型为WH\_JOURNALRECORD，WH\_JOURJOURNALPLAYBACK的两种钩子要求配对使用, 通常使用WH\_JOURNALRECORD来录制用户的操作, 包括键盘和鼠标输入消息, 而WH\_JOURNALRECORD则用来回放用户的操作, 其效果相当于先录制宏然后运行宏。

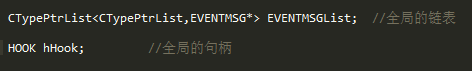
这两种钩子和其他钩子相比有一定的特殊性, 主要体现在以下几个方面:

( 1) 一般来说, 全局钩子要求在动态链接库中实现其回调函数( 也称为钩子函数) , 在一个可执行的.exe 应用程序中安装钩子。但是对于WH\_JOURNALRECORD、WH\_JOURNALPLAYBACK这两种类型的钩子, 不管它们是在动态链接库中实现, 还是在应用程序( .exe) 中实现都是全局钩子( global hook) 。因此, 可以将它们的实现代码直接写在应用程序中, 而不必一定写在动态链接库中。

2) 一般情况下这两种钩子需要配对使用。即先用WH\_JOURNALRECORD 钩子来录制消息，再用WH\_JOURNALPLAYBACK 来回放消息。

( 3) 当WH\_JOURNALPLAYBACK 钩子安装成功并且正在运行时, 系统除了响Ctrl+Alt+Del或Ctrl+Esc 组合键( 这些能强行终止回放钩子的运行) 外, 其他的一切键盘和鼠标消息都不起作用, 直至钩子被释放。所以, 在回放钩子使用结束时一定要及时释放, 否则, 会影响键盘鼠标的正常输入。

为了实现消息的录制与回放, 先定义两个重要的全局变量:



( 1) 定义全局链表。这个链表是全局变量, 以便在录制和回放这两个回调函数之间共享数据。利用MFC 中链表模板类CtypedPtrList, 可以很方便定义一个链表中所有元素类型为EVENTMSGB 的链表。链表中的元素之所以是EVENTMSGB 型是因为在利用WH\_JOURNALRECORD 录制消息时, 与捕获的消息有关的信息都放在一个EVENTMSG 结构体中, 并用指针指向了它。每个消息对应一个EVENTMSG 结构体, 一系列的消息对应了一系列的这样的结构体, 我们应该将这一系列的数据保存起来, 适合于保存这些数据的方法有链表或文件。链表的数据在内存中, 读写时速度快, 但数据在应用程序关闭时丢失; 文件的数据保存在磁盘, 读写较慢, 但数据能持久保存。为了简单起见, 本文采用了链表保存数据。

( 2) 定义全局的钩子句柄。在利用SetWindowsHookEx()安装钩子时, hHook 用来保存该函数返回的钩子句柄; 在利用UnhookWindowsHookEx()卸载钩子时, 将hHook 作为实参传递给它, 表示将该句柄对应的钩子卸载。

**3.3 消息的录制**

消息的录制是用WH\_JOURNALRECORD 钩子来实现的, 该钩子过程的编写比较简单。下面的代码实现将监视到的消息序列保存在链表中的功能。



图3-2 消息录制

代码部分解释如下：

( 1) 指向EVENTMSG 的指针, 其中有刚从系统消息队列中移出的消息, 即刚被处理的那一条消息, 这正是我们要录制的消息。

( 2) 上述录制过程必须要先动态地开辟一块存储空间, 并将lParam 参数指向的结构体复制到这块内存中。不能直接将lParam 指针添加到EventMsgList 链表中去, 因为lParam 指针每次指向的应用程序没有控制权的内存块, 这块内存空间有可能在你回放消息之前就已经被释放掉了,在回放时, 如果仍然EventMsgList 链表读出这个指针, 再去访问它所指向的EVENTMSG 结构,将导致违规访问的严重异常。

( 3) 消息复制的方法有两种: 按结构体的逐个成员复制法和按内存块的逐个字节复制法。前者把内存块看成是有结构的, 而后者把内存块当作是无结构的。本设计采用无结构的方式。

**3.4 消息的回放**

利用WH\_JOURNALPLAYBACK 钩子回放刚才录制时保存在链表中的消息, 回放的顺序是从头至尾顺序地进行。但是, 如果将所有的消息在一个很短的时间间隔内全部回放的话, 结果可能不正确。原因是: 录制的操作中有些操作比较费时间, 比如说某些窗口的打开需要比较长的时间间隔才能完成, 在它没有完成之前, 试图给它发送消息将是无效的。因此, 为了以录制时各个消息的时间间隔来回放, 应该有一种机制能控制回放的速度, 而这种机制在WH\_JOURNALPLAYBACK 钩子中就已经具备。

( 1) 变量i 与bFlag是静态变量。因为在两次函数调用之间 i 和bFlag 的值被保留, 并且初始化操作只做一次。i 变量作为索引, 用来检索链表中的某个元素。bFlag 是一个布尔型变量, 用来防止反复地回放同一条消息而进入死循环。

( 2) 为了实现回放消息时的速度与录制消息时的速度相同, 要利用记录在EVENTMSG 结构体中的time 成员, 它表示在录制时消息发生的时刻。计算前后两个消息的时间之差即为两条消息之间的时间间隔。根据MSDN 关于回放钩子函数的说明, 将此时间间隔作为返回值返回, 将是系统在回放消息前的延时, 延时的时长由返回值决定。也就是说, 本回调函数的返回值表示在回放lParam 指向的消息前应该先延迟的时间。如果返回值等于0, 表示无延迟地立即回放消息。



图3-3 消息回放

**3.5 回放钩子函数的执行过程解析**

系统在回调JournalPlaybackProc 函数时的流程如图1 所示:

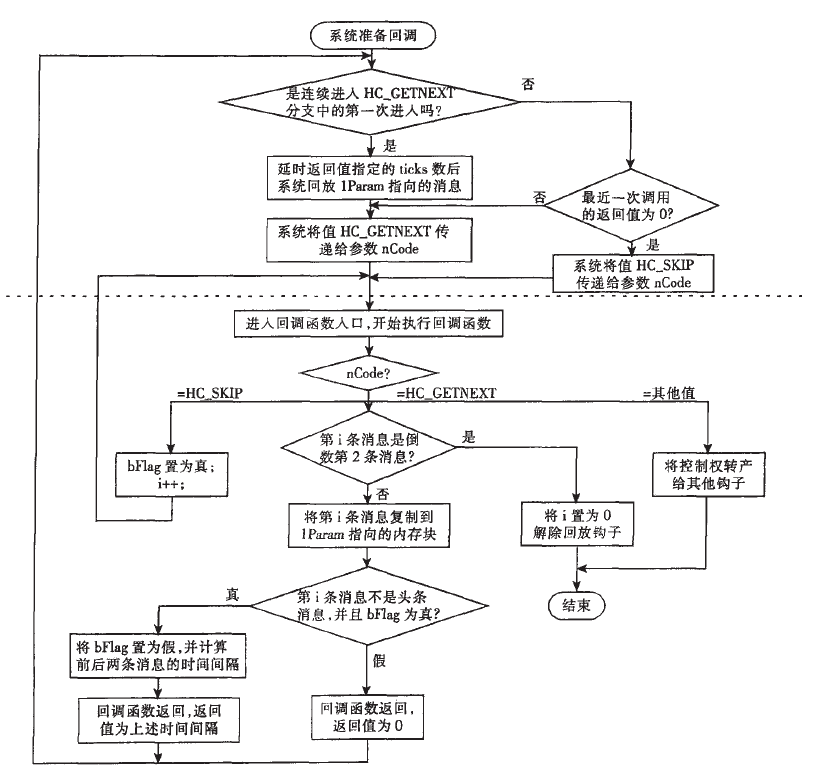


图3-4 回放钩子SET\_JOURNALPLAYBACK的回调函数执行过程流程图

（1）图1 中虚线以上的部分表示回调时操作系统所执行的动作,虚线以下的部分才是JournalPlaybackProc 函数内部定义的行为。从图3-4可以看出, 本回调函数在执行时的行为是系统特定的外部行为与回调函数内部行为相互作用的结果, 对它的完整的理解应该将回调函数的内部行为与系统的回调行为结合起来。

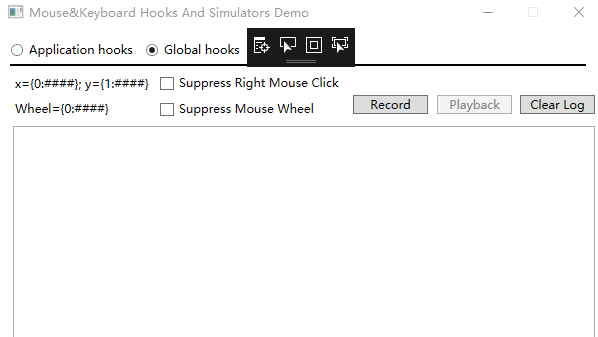
（2）对于每个消息而言, 系统都会在连续两次或两次以上的回调中进入HC\_GETNEXT 分支。对于同一个消息, 不管是第一次还是第二次执行到HC\_GETNEXT 分支, 必须使lParam 每次都指向有效的EVENTMSG 结构。如果只在其中某一次进入到HC\_GETNEXT 分支时使lParam 指向有效的EVENTMSG 结构, 而在另一次中不给lParam 传递有效地址, 则回放时的效果是只有鼠标的动作显示, 而不能产生真正的鼠标操作。

（3）同一个消息在第一次进入HC\_GETNEXT分支时, 由于bFlag 的值为“真”, 所以可以返回与前一个消息之间的时间差。事实上, 这个时间差一定是大于或等于0 的。第二次进入HC\_GETNEXT分支后, 由于bFlag 的值已经变为“假”, 此次的返回值必为0, 从而保证不会无限重复地回放同一条消息。这样既保证了按原来录制的速度来回放,又保证了系统不会陷入死循环。

1. **实现效果**

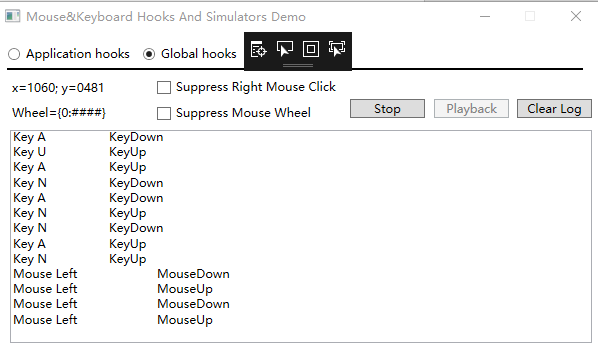
本软件可实现对Windows平台下应用程序基本操作的录制与回放，以word程序为例展现实验效果。运行步骤如下：

1. 打开Record And Playback\src\ Record and Playback.sln
2. 运行Reord and Playback 项目，显示如下运行界面：



上述界面上的按钮Record表示录制，Playback表示回放刚刚录制的动作。

1. 点击Record按钮后，开始录制，随后在Word中进行一系列操作，界面上会同步显示当前的操作（包括鼠标和键盘操作）。



1. 点击Stop，停止录制。
2. 点击Playback，程序开始自动回放刚才录制的操作。
3. **总结**

为了实现Windows平台下的录制与回放操作，本系统采用了一种基于 Windows Hook 的通用的录制/回放机制，充分利用 Windows 系统中事件驱动、消息队列和消息分发的特点，能够在用户操作的同时，动态地监视、截获、记录用户的操作动作; 并在需要回放时读取被持久化保存的消息，通过 Windows Hook 分发到被录制窗口，从而达到高效、透明的录制效果和完整再现用户操作的回放效果。