# 进程间通讯在声卡模式自动切换系统中的应用

王 涛[1,2],李龙澍[1,2],张 婷[1,2],马阳成[3]
(1.安徽大学 计算智能与信号处理教育部重点实验室,安徽 合肥 230039;
2.安徽大学 计算机科学与技术学院,安徽 合肥 230039;
3.安徽大学 物理与材料科学学院,安徽 合肥 230039)

**摘 要**:进程间通讯是大规模软件开发中不能回避的问题。声卡模式自动切换系统用于实现X-Fi声卡在相关应用程序启动和退出时自动地切换到相应的声音模式以发挥最好的声音效果,为此声卡模式自动切换系统中的监控程序需要同被它监视的应用程序之间交换相关数据并实施同步。文中通过声卡模式自动切换系统详细介绍了Windows操作系统中的内存映射文件、互斥、事件等进程间通讯机制和及其在使用中需要注意的问题。

**关键词**:Shell钩子;进程;内存映射文件;同步;事件;互斥
**中图分类号**:TN912.34 **文献标识码**:A **文章编号**:1673-629X(2007)09-0192-04

# Application of IPC in Sound Card Auto Mode Switcher

WANG Tao[1,2],LI Long-shu[1,2],ZHANG Ting[1,2],MA Yang-cheng[3]
(1.Ministry of Education Key Lab. of IC & SP, Anhui University, Hefei 230039,China;
2.College of Computer Science and Technology, Anhui University, Hefei 230039,China;
3. College of Physics and Material Science, Anhui University , Hefei 230039,China)

**Abstract**:The IPC(interprocess communication) is the avoidless issue in the large scale software development. The sound card auto mode switcher is used for X-Fi sound blaster to achieve the best sound effects by automatically switch its mode to the corresponding mode when its associated applications launched and closed. Thus, the monitor program of sound card auto mode switcher need to exchange its data to its associated applications and synchronize to its associated applications. Combines the real case of sound card auto mode switcher to analyse common IPC mechanism in Windows OS and describes problems that should be paid attention to in application.

**Key words**:Shell hook; course; memory mapped file; synchronization; event; mutex

## 0 引 言

自从新加坡创新科技公司于1989年推出了Sound Blaster声霸卡以来,声卡技术又有了极大的提高。创新科技公司于2004年最新推出的X-Fi高档声卡更是增加了独创的模式切换功能,它可以根据用户的需要自动地优化X-Fi声卡的性能和设置。X-Fi声卡支持的三种模式分别为:音乐创作模式、游戏模式以及娱乐模式。每一种模式都可以集中运用声卡的资源,以实现在不同应用中达到最有效的使用。但是在实际使用中却遇到了这样一个问题,用户在启动他们的游戏之前往往忘记切换模式。因此,大多数时间游戏在娱乐模式下启动,这样不能给用户最好的游戏体验。由于游戏通常装载很慢,手动退出游戏切换到游戏模式再重新启动游戏是件很麻烦的事情。文中所述的声卡自动切换系统目标就是要通过一个配置相关应用程序及其模式的列表来实现声卡模式的自动切换。

**收稿日期**:2006-10-21
**基金项目**:安徽省高校学科拔尖人才基金(05025102);安徽省自然科学基金(050420204)
**作者简介**:王 涛(1973-),男,安徽宿州人,硕士研究生,研究方向为智能软件;李龙澍,教授,博士生导师,研究方向为智能软件和知识工程。

## 1 声卡模式自动切换系统简介

声卡模式自动切换系统从功能上可以分为两部分:一是给用户提供一个配置相关应用程序和特定声音模式的列表及其它配置的用户界面;一是监控其他Windows应用程序运行状况以便在其启动和关闭的时候能够实现声卡模式自动切换的监控程序。

图1所示的是声卡模式自动切换系统主窗口界面。列表中的Application项描述的是相关应用程序的全路径名,Mode项描述的是该应用程序在启动时需要

切换的声音模式。窗口右侧的"Add...","Edit...","Remove"按钮用于增加、修改列表项。

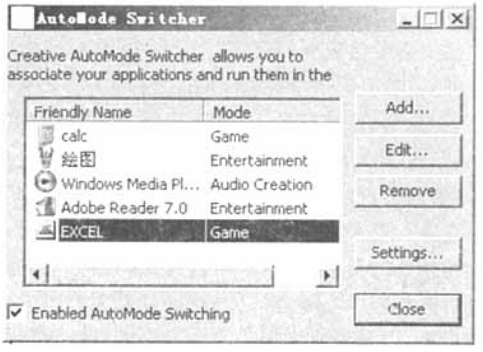


图 1　用户主窗口界面

图 2 所示的是监控程序的处理过程流程图。监控程序通过使用一个 WH_SHELL 类型的系统钩子监控相关应用程序的运行状态。由于这是一个系统范围的钩子,所以在任何应用程序的创建和销毁期间,钩子函数都会被操作系统调用。钩子函数的参数 wParam 中包含了应用程序将要创建和销毁的主窗口的窗口句柄。

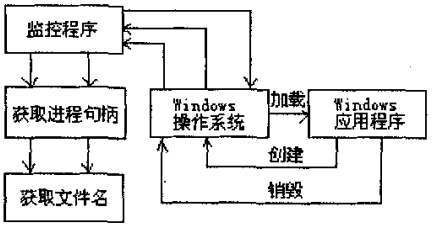


图 2　监控系统流程图

有了有效的窗口句柄,通过调用 Windows API 函数 GetWindowsThreadProcessId 就可以得到进程 ID。然后由这个进程 ID 再通过调用 Windows API 函数 OpenProcess 就可以得到进程句柄。最后,由进程句柄通过调用 Windows API 函数 GetModuleFileNameEx 就可以得到感兴趣的应用程序的全路径名。

如果这个应用程序的路径名与配置文件列表中的一个文件相匹配,监控程序就会再拿与这个应用程序相应的声音模式与当前声卡的声音模式相比较。在两者不同的情况下,监控程序就会把声卡切换到该应用程序相应的声音模式。从而实现声卡声音模式的自动切换。

## 2　Windows 进程间通讯机制简介

进程间通讯包括进程之间的同步和进程之间的数据共享。为了对资源访问实施同步,Windows 提供了许多基本要素来帮助操作,如事件、信标、互斥对象和关键代码段等[1]。为了使应用程序能够迅速而方便地共享数据和信息,Windows 提供了多种机制来帮助操作,这些机制包 RPC、COM、OLE、DDE、窗口消息(尤其是 WM_COPYDATA)、剪贴板、邮箱、管道和套接字等[1]。下面对一些常用的进程间通讯技术做一个简单的描述。

### 2.1　钩　子

钩子(Hook)是 Windows 消息处理机制的一个平台,应用程序可以在上面设置子程以监视指定窗口的某种消息,而且所监视的窗口可以是其他进程所创建的[2]。

钩子实际上是一个处理消息的程序段,通过系统调用,把它挂入系统。每当特定的消息发出,在没有到达目的窗口前,钩子程序就先捕获该消息,亦即钩子函数先得到控制权。这时钩子函数既可以加工处理(改变)该消息,也可以不作处理而继续传递该消息,还可以强制结束消息的传递。

钩子有消息钩子、鼠标钩子、键盘钩子和外壳钩子等多种类型,不同类型的钩子可以使应用程序能够监视不同类型的系统消息。

### 2.2　内存映射文件

在单个计算机上共享数据的最底层机制是内存映射文件。

使用内存映射文件在进程之间共享数据的方法是通过让两个或多个进程映射同一个文件映射对象的视图来实现的[3],这意味着它们将共享物理存储器的同一个页面。因此,当一个进程将数据写入一个共享文件映射对象的视图时,其他进程可以立即看到它们视图中的数据变更情况。

如果多个进程需要共享单个文件映射对象,则要求所有进程必须使用相同的名字来表示该文件映射对象。

### 2.3　互　斥

互斥对象(mutex)能够确保线程拥有对单个资源的互斥访问权。互斥对象通常用于保护有多个线程访问的内存块。互斥对象能够保证访问内存块的任何线程拥有对该内存块的独占访问权,这样就能够保证数据的完整性。

### 2.4　事　件

在所有的内核对象中,事件内核对象是个最基本的对象。事件能够通知一个操作已经完成[4]。通过创建命名事件,事件就可以跨越进程边界使用。

有两种不同类型的事件对象。一种是人工重置的事件,另一种是自动重置的事件。当人工重置的事件得到通知时,等待该事件的所有线程均变为可调度线程。当一个自动重置的事件得到通知时,等待该事件

的线程中只有一个线程变为可调度线程。

事件对象在当一个线程执行初始化操作,然后通知另一个线程执行剩余的操作时使用得最多。事件初始化为未通知状态,然后,当该线程完成它的初始化操作后,它将事件设置为已通知状态。这时,一直在等待该事件的另一个线程发现该事件已经得到通知,因此它就变成可调度线程。这第二个线程知道第一个线程已经完成了它的操作。

下面结合声卡模式自动切换系统的具体实现,对在声卡模式自动切换系统中使用到的进程间通讯技术做进一步的探讨。

## 3 Windows 进程间通讯在声卡模式自动切换系统中的应用举例

### 3.1 用钩子实现监控程序

声卡模式自动切换系统中的监控程序需要在相关的应用程序启动和关闭时及时得到消息以便能够在其启动和关闭时实现声卡模式的自动切换。这就要求监控程序能够在被监控的应用程序中加入一段代码,这段代码可以在被监控的应用程序启动和关闭时首先把应用程序启动和关闭的信息发送给监控程序,然后让应用程序等待,直到监控程序处理完毕后再让应用程序继续执行。

钩子是 Windows 消息处理机制的一个平台,应用程序可以在上面设置子程以监视指定窗口的某种消息,而且所监视的窗口可以是其他进程所创建的。当消息到达后,应用程序可以在目标窗口处理之前处理它。钩子机制允许应用程序截获处理 Window 消息或特定事件。其中的 Shell 钩子可以监控 Windows 应用程序主窗口创建和销毁的消息,使用 Shell 钩子可以实现监控程序对其他 Windows 应用程序创建和销毁阶段的监控。

监控程序通过使用 Windows API 函数 SetWindowsHookEx 安装一个 WH_SHELL 类型的系统钩子,钩子函数(HookProc)放在独立的动态连接库(DLL)中,系统会自动将包含钩子回调函数的 DLL 映射到受钩子函数影响的所有进程的地址空间中,即将这个 DLL 注入了那些进程。

钩子函数的原型如下:

LRESULT CALLBACK HookProc ( int nCode, WPARAM wParam, LPARAM lParam);

对于 nCode,仅仅需要处理 HSHELL_WINDOWCREATED 和 HSHELL_WINDOWDESTROYED 消息。参数 wParam 中包含了应用程序将要创建和销毁的主窗口的窗口句柄。

利用这个窗口句柄,监控程序可以获取该应用程序的全路径名,从而决定该应用程序是否为被监控的应用程序并决定是否要切换声卡的声音模式。

使用钩子给应用程序带来便利的同时,也会增加系统的负担,引起系统性能下降。因此在钩子使用完毕后应及时用 Windows API 函数 UnhookWindowsHookEx 将其卸载并释放其所占资源。

### 3.2 用内存映射文件实现数据共享

在声卡模式自动切换系统的设计过程中,为了减轻系统钩子的负担,将声卡声音模式切换部分的代码放到了监控程序中处理。同时,为了避免监控程序的负担过重,也需要系统钩子能够及时获得配置了应用程序及其相应声音模式的列表信息,这个列表是可由用户动态修改的。这就要求监控程序和被监控的应用程序之间能够实现数据共享。共享的数据是一个配置了相关应用程序及其相应声音模式的列表,以下是其中一个条目的数据结构:

```
typedef struct tagItem
{
    DWORD dwAppPathLength;
    TCHAR szAppPathName[MAX_PATH];
    DWORD dwFriendlyNameLength;
    TCHAR szFriendlyName[MAX_PATH];
    DWORD dwModeStart;
    DWORD dwModeEnd;
    DWORD dwProcessNum;
    MyProcessInfo aProcessInfo[MAX_PROCESS];
} Item;
```

在这个数据结构中除了 MyProcessInfo 之外的其它部分包括应用程序全路径名、应用程序别名和应用程序创建和关闭时要切换到的声音模式等都是条目的静态数据,仅由用户界面程序改变,保存在一个磁盘文件中。数据结构 MyProcessInfo 描述的是单个条目信息的动态数据,本系统不限制一个相关的应用程序只能运行一次,所以需要一个列表保存应用程序各个运行着的进程的信息。一般情况下,一个应用程序进程只用一个主窗口,但是现在有很多应用程序在一个进程中都可以有多个顶层无主窗口,例如 Word,Acrobat 等都是这样。对于同一个应用程序进程,声卡的声音模式只需要切换一次,字段 dwWindowNum 就是用于对同一个进程的顶层无主窗口进行计数,这样就可以保证在应用程序的这个实例退出时,声卡模式可以切换且仅切换一次,字段 dwWindowNum 的值在钩子中改变。对于同一个应用程序进程,如果在应用程序创建的时候用户选择不改变声音模式,这也就表明在应用程序退出的时候用户同样不希望改变声音模式,字

段 dwExitNoChange 就是用于此目的,字段 dwExitNoChange 的值在监控程序中改变。

监控程序使用 Windows API 函数 CreateFileMapping 创建应用程序及其相关声音模式的列表的内存映射文件,其中 hFile 参数的值为 INVALID_HANDLE_VALUE,用来创建页文件方式的内存映射文件[5]。页文件方式的内存映射文件比起磁盘文件方式的内存映射文件具有更高的灵活性和更高的存取效率。监控程序在创建文件映射对象后调用 Windows API 函数 MapViewOfFile 将内存映射文件映射到监控进程的地址空间内。

钩子程序通过调用 Windows API 函数 OpenFileMapping 来获得应用程序列表的内存映射文件句柄。一旦其他进程获得映射对象的句柄,就可以像创建进程那样调用 MapViewOfFile 函数来映射对象视图。用户可以使用该对象视图来进行数据读写操作,以达到数据通讯的目的。

当用户进程结束使用共享内存后,应调用 UnmapViewOfFile 函数以取消其地址空间内的视图。

### 3.3　用互斥对象实现对列表共享数据的独占操作

对于上述的相关应用程序及其声音模式列表的共享数据,在监控程序启动时由监控程序把磁盘文件中的列表信息写入内存映射文件,用户可以通过用户界面程序增删修改列表。钩子函数需要读取该列表信息以判断应用程序是否为被监控的程序,并在确定是被监控的应用程序时要记录顶层无主窗口的数目。监控程序也要由用户对提示信息的选择情况来决定应用程序退出时是否修改声音模式。为了避免这些操作同时进行而带来的列表共享数据的不一致性,需要对这些操作采取同步机制。

互斥对象能够确保线程拥有对单个资源的互斥访问权,可以满足对列表共享数据操作的同步控制的需要。

在本例中,互斥对象的使用方法是:在监控程序的初始化阶段,监控程序首先调用 CreateMutex 函数以创建互斥对象;在钩子函数中通过调用 OpenMutex 函数,其他应用程序进程可以获得同一互斥对象相关的句柄。在每次访问被监控应用程序及其声音模式列表的共享数据前,通过调用等待函数 WaitForSingleObject,线程就能够获得对列表共享数据的访问权。拥有互斥对象的线程在完成自己的操作后,调用 ReleaseMutex 函数释放互斥对象。其他等待互斥对象的线程在互斥对象被释放后只能有一个线程可以获取对资源的独占访问权,其余的线程则要继续等待。

由于在互斥对象释放前,其他需要访问列表共享数据的操作都不能进行,因此需要尽量减小互斥对象的封锁粒度。在钩子函数中,将对列表共享数据的文件名比较部分和对顶层无主窗口的计数部分分别放在两个单独的互斥操作中进行。

### 3.4　用事件对象实现进程间的同步

当相关应用程序的创建或销毁时,监控程序需要在应用程序处理之前,弹出一个如图 3 所示的对话框,询问用户是否要将声卡切换到相应的声音模式。只有在用户作出回答以后应用程序才能继续运行。为此,监控程序和被监控的应用程序之间需要采取一定的同步措施。

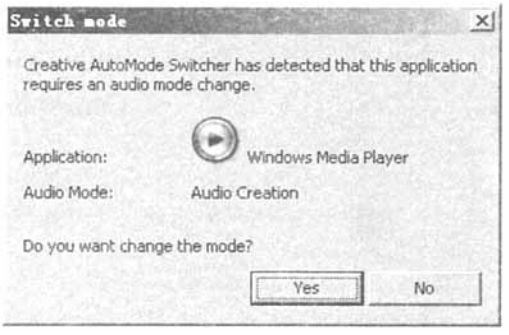


图 3　模式切换对话框

在本例中,使用事件对象作为监控程序和应用程序之间同步的手段。事件对象的使用方法是:对每一次发送给监控程序的同步消息,钩子函数都调用 CreateEvent 函数创建一个包含应用程序路径名和应用程序窗口句柄信息的事件对象,以此保证了事件对象的唯一性,并设置该事件对象的初始状态为未通知状态,然后调用 WaitForSingleObject 函数将自己挂起。事件名的产生代码如下:

```
pDest = _tcsrchr(szPathName, _T('\\'));
_tcscpy(szTemp, pDest+1);
pDest = _tcsrchr(szTemp, _T('.'));
*pDest = _T('\0');
_stprintf(szMsgEvent, _T("CTSModeMsgEvent%d%s"), (DWORD)wParam, szTemp);
```

监控程序在处理完相关工作后,调用 OpenEvent 函数获取该事件对象句柄,再调用 SetEvent 将该事件对象设置为通知状态,被挂起的钩子进程得以继续执行。

## 4　结束语

实践证明,使用 Windows 操作系统的进程间通讯机制可以实现复杂的程序功能。Windows 操作系统的进程间通讯机制还包括信号灯、邮箱、管道等,限于篇

(下转第 202 页)

其中,占用资源有:一个 18×18 位,两个 18×6 位,一个 6×6 位的乘法器,3 个 48 位加法器,7 个 48 位的寄存器。

对于有符号数的相乘结构与此大致相同,只需将符号位考虑在内即可。综合布线后仿真,器件采用 Virtex-II xcv1000 fg456-6,最高频率可达到 195.160MHz。

## 4 总 结

通过比较,可以看到采用相同的器件,该方法已经将运算速度提高了 150MHz 以上,大大提高了运算速度,而且采用流水技术,实现了并行计算。此外,这种方法还具有结构电路简单的明显优势。

在工程实践中,已经将这种方法应用于浮点乘法器的设计当中,收到了良好的效果。

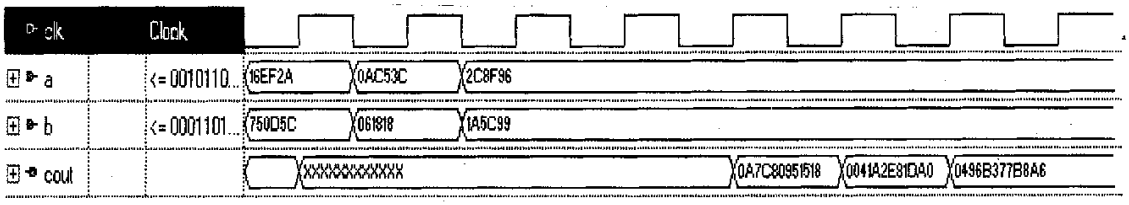


**图 1 2xn 结构乘法器仿真波形**

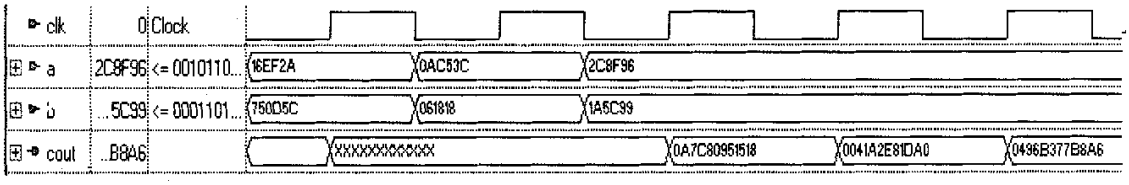


**图 2 18x18 结构乘法器仿真波形**

**参考文献:**

[1] 梁 峰,邵志标,梁 晋. Radix-16 Booth 流水线乘法器的设计[J].西安交通大学学报,2006,40(10):1111-1114.

[2] Booth A D. A signed binary multiplication technique[J]. Quarterly Journal of Mechan & Appl Math,1951,4(2):236-240.

[3] 李小进.定点符号高速乘法器的设计与 FPGA 实现[J].微电子学与计算机,2005,22(4):119-125.

[4] 常静波,郭 立.一种 3 级流水线 Wallace 树压缩器的硬件设计[J].微电子学与计算机,2005,22(1):160-165.

[5] 侯伯亨,顾 新. VHDL 硬件描述语言与数字逻辑电路设计[M].西安:西安电子科技大学出版社,2000.

(上接第 195 页)

幅,不再一一描述。希望文中描述的声卡模式自动切换系统中的进程间通讯机制应用能够给大家带来借鉴作用。

**参考文献:**

[1] Richter J. Programming Applications for Microsoft Windows [M]. 4th Edition. [s.l.]:Microsoft Press,1999.

[2] 吴群飞. Hook 技术在监控 PSpice 中的应用研究[J]. 计算机应用研究,2005(4):185-186.

[3] 杨宁学. 内存影射文件及其在大数据量文件快速存取中的应用[J]. 计算机应用研究, 2004(8):187-188.

[4] 王文磊. 多线程编程技术实现经典进程同步问题[J]. 计算机技术与发展, 2006, 16(3):110-112.

[5] 谈 蓉. Win32 系统下进程间通讯[J]. 电脑开发与应用, 2003(10):19-20.

(上接第 198 页)

**参考文献:**

[1] Wright G R, Stevens W R. TCP/IP 详解 卷 2:实现(英文版)[M].北京:机械工业出版社,2002.

[2] Stevens W R. TCP/IP Illustrated Volume I: The Protocols [M]. Beijing: China Machine Press, 2000.

[3] 李 勇,戴瑜兴.基于 UDP 协议的实时监控系统[J].电子技术,2003(11):37-40.

[4] 费仁元,王 民,徐洪安. 产品生产服务过程中的网络监控技术[J].机械工程学报,2003(9):35-37.

[5] Yuan F, Huang C. Performance Analysis of Resilient Packet Ring with Single Transit Buffer [EB/OL]. 2002. http://www.sce.carleton.ca/faculty/huang/ICT2002 final version.pdf.

[6] Francisco M, Yuan F, Huang C, et al. A Comparison of Two Buffer Insertion Ring Architectures with Fairness algorithms, Anchorage [EB/OL]. 2002. http://www.sce.carleton.ca/faculty/huang/mark-icc03.pdf.