

# Win32 环境下的多线程同步技术的研究

许斌龙,张 晶,王国明

(安徽理工大学 计算机科学与工程学院,安徽 淮南 232001)

**摘 要:**由于多线程同步技术是当今软件开发中的一项重要技术,所以在许多软件中得到广泛的应用。针对目前 Win32 环境下几种常用的线程同步技术进行了详细的研究,分析了它们各自的特点,总结了各种线程同步技术的应用场合。为了能够进一步提高多线程同步技术的执行效率,文中在此基础上,提出了一种利用链表对现有的线程同步技术进行改进的方法。改进后的算法较好地解决了 CPU 时间片的浪费问题,提高了 CPU 时间片的利用率。

**关键词:**Windows 操作系统;多线程;同步技术;时间片;链表

**中图分类号:**TP301

**文献标识码:**A

**文章编号:**1673-629X(2013)12-0026-04

**doi:**10.3969/j.issn.1673-629X.2013.12.006

## Study on Synchronization Technology of Multi-thread in Win32 Environment

XU Bin-long, ZHANG Jing, WANG Guo-ming

(School of Computer Science & Engineering, Anhui University of Science & Technology, Huainan 232001, China)

**Abstract:** Because the multi-thread synchronization technology is an important technology in software development, so it is applied widely in many software. Studied several common thread synchronization technology currently under Win32 environment, analyzed their features, and summarized the applications of all kinds of thread synchronization technology. On this basis, an improvement method on the thread synchronization using the linked list was proposed. The improved algorithm can solve the problem of waste CPU time slice, improve the utilization rate of the CPU time slice.

**Key words:** Windows operating system; multi-thread; synchronization technology; time slice; linked list

## 0 引 言

随着计算机在社会上的快速普及,不管是在生活上,还是在工作上,人们现在已经越来越离不开计算机了,而每台计算机都需要有一个管家式的角色,那就是操作系统。

美国微软公司的 Windows 操作系统在个人计算机操作系统市场处于垄断地位,所以 Windows 操作系统的用户数量之大就不言而喻了。现在虽然 64 位的操作系统早已面市,但大部分的 Windows 操作系统用户还是使用 32 位的操作系统,所以既然有如此大的用户数量,那么对这个系统上运行的软件需求也肯定是一个不小的数目。这也是有如此多的程序员选择在 Win32 环境下工作的原因,当然,Win32 环境下的多线程同步技术与 Win64 下的非常相近。

虽然不是每个人都懂得如何用程序来实现软件的

功能,但随着人们对计算机越来越多的使用,软件的需求变得越来越复杂,那么在程序员实现这种复杂的需求时,更多的技术将运用于其中,但多线程技术是必不可少的。

文中将分析几个 Win32 环境下常用的多线程技术,同时对这几个多线程技术的适用场合进行分析和比较。

## 1 进程和线程

进程是指装入内存,且准备执行的程序<sup>[1]</sup>。每个进程都有自己私有的虚拟地址空间。每个进程都有一个主线程,但可以建立另外的线程<sup>[2-4]</sup>。进程中线程是并行执行的,每个线程占用 CPU 的时间由系统来划分<sup>[5-7]</sup>。可以把线程看成是操作系统分配 CPU 时间的基本实体。系统不停地在各个线程之间切换,它对

收稿日期:2013-03-03

修回日期:2013-06-12

网络出版时间:2013-09-29

基金项目:安徽省淮南市自然科学基金(2009A05012)

作者简介:许斌龙(1989-),男,硕士研究生,研究方向为网络与信息安全;王国明,硕士,副教授,研究方向为网络与信息安全。

网络出版地址:<http://www.cnki.net/kcms/detail/61.1450.TP.20130929.1541.031.html>

线程的中断是汇编语言级的。系统为每一个线程分配一个 CPU 时间片,某一个线程只要在分配的时间片内才有对 CPU 的控制权。实际上,在 PC 机中,同一时间只有一个线程在运行。由于系统为每个线程划分的时间片很小,所以看上去好像是多个线程在同时运行<sup>[8]</sup>。

进程中的所有线程共享进程的虚拟地址空间<sup>[9]</sup>,这意味着所有线程都可以访问进程的全局变量和资源。这一方面为编程带来了方便,但另一方面也容易造成冲突。比如说,线程 A 正在操作进程中的一块内存,如果在操作到一半时,这个线程的时间片到了,那么 CPU 将转去执行另一个线程,但在这时,如果这个线程也去操作了线程 A 操作过的内存,那么当 CPU 再转回执行线程 A 时,线程 A 的操作结果就是一个不确定的结果了。

为了能够更加直观地描述这个问题,可以举一个小例子。在主线程中建立一个子线程,在主线程中完成隔 40 ms 打印一串字符串六个 0 并换行的操作,在子线程中完成隔 40 ms 打印一串字符串六个 1 并换行的操作,那么图 1 就是想象中的程序执行的结果。

```
"F:\小论文\win32\自己的论文\test\Debug\test.exe"
000000
111111
000000
111111
000000
000000
111111
000000
000000
111111
000000
000000
111111
000000
000000
111111
000000
000000
111111
```

图 1 想象中的程序执行结果

但是真正的程序执行结果并非如此,图 2 就是真正的程序执行结果。如图所示,执行结果出现了在同一行中 0 和 1 间隔,或者只有换行,或者不止六个字符的不正常情况,这就是线程不同步带来的严重后果。

当然,这种情况不仅仅发生在共用同一块内存的时候,在共用同一硬件资源,如通讯接口时,也会发生这种情况。这时,就需要运用线程同步技术来避免这

种情况的发生,以保证程序运行的正确性。

```
"F:\小论文\win32\自己的论文\test\Debug\test.exe"
000000
000000
111111
101100010

101100010

1110000
1110000
111111
111111
000000
111111
111111
000000
110000

110000
```

图 2 真正的程序执行结果

## 2 线程同步技术

线程同步技术比较多,此节主要分析其中比较常用的有代表性的几种,包括:原子锁、临界区、事件和可等候定时器。

### 2.1 原子锁

用 C 或 C++写的代码如果转换成汇编,那么一句简单的代码可能就会被翻译成几句,如++操作,转换成汇编就是三句语句,如果在这三句语句还没执行完时,CPU 的时间片就已经到了,那么只要在此时有其他线程操作了刚才还未操作完的内存,就有可能引发异常。

原子锁就是专门用于对单条指令进行操作的同步技术。有这样一个 API 函数专门用于实现对++运算的原子锁:

```
LONG InterlockedIncrement(
    LPLONG lpAddend// 要进行自增运算的变量地址
);
```

只要把要进行自加运算的变量地址作为参数传进去就好了。这样就相当于把这个自增操作所用到的内存给锁定了,当语句还没执行完而 CPU 的时间片到了的时候,如果有其他线程想操作这块内存,那么就会被拒绝,这样就保证了这块内存中的数据安全,也就保证了运行结果的正确。

原子锁的特点是执行效率较高,但是它只能用于锁定单行指令,使用较麻烦,因为要用不同的函数去对

应地锁定不同的操作。

## 2.2 临界区

原子锁只能锁定单条语句,而临界区可以锁定多句代码,防止多个线程同时使用同一段代码,它通过提供一个进程内所有线程必须共享的对象来控制线程<sup>[10]</sup>。

在使用临界区时,首先要在主线程中用 `InitializeCriticalSection` 函数初始化一个新的临界区,然后就可以在子线程中使用这个临界区了,在需要锁定的代码前面通过 `EnterCriticalSection` 函数进入临界区,当代码功能实现完毕之后通过 `LeaveCriticalSection` 函数离开临界区,最后在主程序中要用 `DeleteCriticalSection` 函数删除临界区资源<sup>[11]</sup>。

使用临界区进行多线程同步的好处比较明显,那就是使用方便,因为只要 4 个函数就能实现,而且它可以作用于多句代码。所以对程序员来讲,这种多线程技术在使用的便捷性上较原子锁有明显的优势。

## 2.3 事件

事件主要用于解决程序之间的通知问题。上文介绍的原子锁和临界区都只涉及到程序员对单个线程的控制,如果希望线程之间能够进行交互来达到线程间的同步,那么可以使用事件来实现。

事件具有两种状态,分别是有信号状态和无信号状态。用事件实现线程同步的基本思想是:线程通过设置和检测事件的状态来实现同步。也就是说,一旦某个线程在执行需要锁定的代码之前获得了事件的有信号状态,那么这个线程就要马上把事件置为无信号状态,以达到不被其他线程干扰的目的,当然,当需要锁定的代码已经执行完毕后,该线程也要马上把事件置为有信号状态,以使其他线程能够顺利执行下去。

在使用事件时,首先要在主线程中用 `CreateEvent` 函数初始化一个新的事件,然后就可以在子线程中使用这个事件了,在需要锁定的代码前面通过等待函数 (`WaitForSingleObject` 或 `WaitForMultipleObjects`) 得到事件的有信号状态,一旦得到了事件的有信号状态,就马上用 `ResetEvent` 函数把事件设置为无信号状态,然后当代码功能实现完毕之后通过 `SetEvent` 函数将事件设置为有信号状态,最后在主程序中要用 `CloseHandle` 函数释放事件。

在使用事件实现线程间的同步技术时很容易犯一个错误:死锁。如果两个线程之间是相互等待的关系就会产生死锁,这是在使用事件是常犯的错误。

## 2.4 可等候定时器

原子锁、临界区和事件这三个技术在实现线程同步时,程序员都不能精确地通过时间来切换线程,如果

那就需要使用可等候定时器。

可等候定时器可以按指定的时间间隔通知程序,精度高(纳秒级),准度高(因为不走消息循环),第一次启动的间隔时间以 100 纳秒为单位,而后面的间隔时间以毫秒为单位。

可等候定时器的使用方法与上文介绍的其他技术类似,但是在通过 `CreateWaitableTimer` 函数创建可等候定时器之后,需要使用 `SetWaitableTimer` 函数来设置可等候定时器,这是此技术的关键所在。下面是 `SetWaitableTimer` 函数的原型及各个参数的介绍:

```
BOOL SetWaitableTimer(  
    HANDLE hTimer// 可等候定时器句柄  
    const LARGE_INTEGER * pDueTime// 第一次启动时间(是一个可以表示 64 位数的结构体)  
    LONG lPeriod// 第一次之后的间隔时间  
    PTIMERAPCROUTINE pfnCompletionRoutine// APC 回调函数(现在一般不用)  
    LPVOID lpArgToCompletionRoutine// APC 回调函数的参数(现在一般不用)  
    BOOL fResume// 待机处理标识(TRUE/FALSE)  
);
```

其中,参数 `pDueTime` 有两种情况:

(1) 如果取正数,表示绝对时间,具体系统时间(年月日时分秒);

(2) 如果取负数,表示相对时间,和当前时间的的时间差,如: -10000000 表示 1 秒之后启动。

参数 `lPeriod` 的单位为毫秒,如果为 0,可等候定时器只执行一次。

参数 `fResume` 只有在机器处于待机状态时有用,如果为 `TRUE` 表示时间到了,唤醒机器,继续运行;如果为 `FALSE` 表示时间到了,不唤醒机器,即取消通知。

## 3 改进后的线程同步技术算法

基于上述分析的同步技术,下面详细分析文中改进后的一种线程同步技术。

这种方法是受 CPU 时间片的分配机制的启发,因为 CPU 在分配时间片时是以线程为基本单位的,那么假设有一个程序运用了上文所讲的线程同步技术,那么有一种情况是非常常见的,那就是占用当前 CPU 时间片的线程要操作的内存或者硬件资源正被其他线程所使用,此时该线程将会不能进行任何操作,只能“眼睁睁地看着”CPU 时间片被耗费完。虽然 CPU 时间片是一段极短的时间,但是如果是一个相对较复杂而且需要长时间运行的程序,那么这样的浪费肯定也是会在一定程度上影响运行效率的。

为了解决这一问题,可以在程序中维护若干个链表,每个链表都用于存储这个程序中需要访问同一资

源的子线程的句柄,这一句柄可以通过 CreateThread 函数的返回值得到。

有了这个链表之后,可以在每个子程序的运行之初通过 SuspendThread 函数将各自链表中的除了当前线程的其他子线程都挂起,而在每个线程结束时再通过 ResumeThread 函数将它们再一一唤醒(当然,在执行挂起和唤醒操作时需要使用上文提到的线程同步技术对它们进行保护)。

因为被挂起的线程,CPU 是不会分配时间片给它们的,所以这样就可以避免上述问题中对时间片的浪费。

这种同步技术主要适用于需要对同一硬件资源进行操作的多个线程,因为一般情况下硬件资源如通讯端口地址都是比较稳定的,所以可以对不同的端口建立不同的链表,用于存储特定的线程句柄。对于在同一程序中存在多个线程都需要频繁并且较长时间地操作同一硬件资源时,这一方法的效果就会有所体现。

## 4 结束语

文中主要分析了四种较常用的 Win32 环境下的多线程技术,并分析了各种同步技术的特点。最后,在常用的线程同步技术的基础上提出了一种改进的线程同步技术算法,这种同步技术的使用面可能没用常用的线程同步技术那么广,但是针对特定的情况,这一同步技术可以很好地提高程序的运行效率,提高了 CPU 时间片的利用率。

各个技术之间不存在哪个技术好,哪个技术不好,只有适合与不适合。在不同的程序环境下需要由程序员来选择合适的同步技术。甚至,在一个复杂的程序

中程序员需要使用多种线程同步技术来进行协同的工作。

## 参考文献:

- [1] 马魁涛,蔡颖,郭宝峰. Win32 进程间信息共享的实现方法研究[J]. 计算机应用与软件,2007,24(12):119-120.
- [2] 刘红海,候向华,蔡勇. 基于 Win32 的多线程技术及其应用[J]. 计算机工程与设计,2003,24(10):113-115.
- [3] 郝晓艳,杨波,孙奕奇. Win32 下线程同步及其在 MFC 中的处理[J]. 济南大学学报(自然科学版),2002,16(4):332-335.
- [4] 朱华章,孟凤珍. 基于 Win32 的多线程 PCI 设备驱动程序设计[J]. 计算机工程与应用,2004(4):107-111.
- [5] Feng Jing,Zhong Hu,Ao Guoqiang,et al. Principles and application of the real-time hardware-in-the-loop simulation platform based on multi-thread and CAN [C]//Proc of ISIE. [s. l.]:IEEE,2008:2225-2230.
- [6] Wang Yongwen,Zheng Qianbing,Dou Qiang,et al. Low power design for a multi-core multi-thread microprocessor [C]//Proc of GreenCom. Hangzhou:IEEE,2010:351-356.
- [7] Liu Hong, Zheng Jianxiang, Chen Ying. The application of multi-thread-based embedded system in the fire monitor [C]//Proc of ISECS. Washington DC:IEEE Computer Society,2009:506-508.
- [8] 孙云霞. Win32 多线程编程的控制技术[J]. 电脑编程技巧与维护,2008(17):13-15.
- [9] 张红玲. Win32 环境下的多线程技术[J]. 沈阳师范学院学报(自然科学版),2000,18(4):11-16.
- [10] 王日宏. 基于 VC 的 Win32 多线程同步问题[J]. 计算机系统应用,2004(7):60-62.
- [11] 袁松茂. Win32 线程同步对象浅析[J]. 株洲工学院学报,2003,17(2):41-45.

(上接第 25 页)

- [7] Kudo T. CRF++ 0.57: Yet another crf toolkit [EB/OL]. 2012. <http://crfpp.googlecode.com/svn/trunk/doc/index.html>.
- [8] Tseng H, Chang Pichuan, Andrew G, et al. A conditional random field word segmenter for sishan bakeoff 2005 [C]//Proceedings of the fourth SIGHAN workshop on Chinese language processing. Jeju Island, Korea: [s. n.], 2005:168-171.
- [9] SIGHAN bakeoff 2005. Second international Chinese word segmentation bakeoff result summary [EB/OL]. 2005. <http://www.sishan.org/bakeoff2005/data/results.php.htm>.
- [10] 滕少华. 基于 CRFs 的中文分词和短文本分类技术 [D]. 北京:清华大学,2009.
- [11] Toutanova K, Manning C D. Enriching the knowledge sources used in a maximum entropy part-of-speech tagger [C]//Proceedings of joint SIGDAT conference on empirical methods in natural language processing and very large corpora. [s. l.]: Association for Computational Linguistics, 2000:63-70.
- [12] 李实,叶强,李一军,等. 挖掘中文网络客户评论的产品特征及情感倾向 [J]. 计算机应用研究,2010,27(8):3016-3019.

Win32环境下的多线程同步技术的研究

作者：[许斌龙](#)，[张晶](#)，[王国明](#)，[XU Bin-long](#)，[ZHANG Jing](#)，[WANG Guo-ming](#)

作者单位：[安徽理工大学 计算机科学与工程学院, 安徽 淮南, 232001](#)

刊名：[计算机技术与发展](#)

英文刊名：[Computer Technology and Development](#)



年，卷(期)：2013(12)

本文链接：[http://d.g.wanfangdata.com.cn/Periodical\\_wjfz201312006.aspx](http://d.g.wanfangdata.com.cn/Periodical_wjfz201312006.aspx)