实时信号量机制的研究

钟　剑[1]，蒋泽军[1]，王丽芳[1]，刘志强[2]
（1.西北工业大学 计算机学院，陕西 西安　710129；2.西北工业大学 软件与微电子学院，陕西 西安　710072）

摘　要：随着实时嵌入式技术的发展，越来越多的实时系统采用 Windows 操作系统平台。为了满足实时系统严格的响应时间要求，增加 Windows 操作系统的实时能力非常必要。进程间的通信是任何一个实时操作系统中必不可少的一部分，在此分析 Windows 操作系统原有进程间通信的弊端，提出改进方法，在 Windows 操作系统上减少进程间通信所消耗的时间，提高进程间通信的效率。

关键词：进程间通信；信号量；MDL；实时操作系统
中图分类号：TN919-34；TP316.86　　**文献标识码**：A　　**文章编号**：1004-373X(2012)02-0040-03

Research of real-time semaphore mechanism

ZHONG Jian[1]，JIANG Ze-jun[1]，WANG Li-fang[1]，LIU Zhi-qiang[2]
（1. School of Computer Science，Northwestern Polytechnical University，Xi'an 710129，China；
2. School of Software and Microelectronics，Northwestern Polytechnical University，Xi'an 710072，China）

Abstract：With development of real-time embedded technology，the operating system platform of Windows series is increasingly adopted by real-time systems. To meet the strict request of response time for real-time systems，it is necessary for Windows operating system to add its real-time ability. Since the inter process communication (IPC) is an absolutely essential part of any real-time operating system，the original IPC disadvantages of Windows operating system are analyzed，and an improvement method is proposed. That is，the time consumed by PIC is reduced in Windows operating system，and the efficiency of IPC is enhanced.

Keywords：IPC；semaphore；MDL；real-time operating system

0　引　言

由于国防、航天航空、自动控制等领域越来越需要操作系统具有实时性，所以实时操作系统[1-2]越来越受到关注，目前正成为研究的热门领域。特别是对于Windows操作系统的实时化扩展变得越来越多，这是因为 Windows 平台具有更强的性能和更低的价格，该平台上可以运行多种成熟的应用，并且提供了种类繁多的开发工具，丰富的 Win 32 应用程序接口和大量熟悉该平台的开发支持人员，而且还有数量庞大的用户群体。这些原因促使了越来越多的研究人员从事 Windows 操作系统的实时化研究。

1　问题描述

影响 Windows 操作系统的实时性有多方面，包括中断的延迟、线程优先级太少、分发器对象造成的延迟、定时器与调度周期模糊、虚拟存储换页造成的延迟和优先级反转[3]等。

Windows 操作系统已经提供了实现 IPC 的方法[4]，但是却无法达到实时性的要求。本文主要研究如何提高信号量机制的实时性，进而提高 Windows 操作系统 IPC 的实时性。在进程间通信中协调不同进程运行步骤的事件、信号量、互斥体等都是 Windows 操作系统的分发器对象。这些对象缺乏必要的实时机制，它们既无法防止线程侵占，也无法使线程按优先级顺序等待对象响应，进而会造成非常严重的优先级反转现象。因此，实时扩展需要实现自己的事件、信号量、互斥体等对象。Windows 操作系统内核功能强大，内核管理系统中的所有资源，用户态下的绝大部分程序不能脱离内核独立运行，它们需要内核提供支持，即用户进程会间接调用系统调用[5]，然后进入内核，让内核处理相应的请求，内核处理完毕后，再退出内核。

由用户态向内核态切换，这个过程造成了一定程度上的时间开销，如果能把这段时间省下来，就能提高 Windows 操作系统的实时性。但是不可能完全消除这部分时间的消耗，除非不调用系统。但是在实际应用

收稿日期：2011-08-26
基金项目：航空科学基金资助项目（2010ZD53，2009ZD53044）；陕西省自然科学基金资助项目（2009JM8017，2009JQ021，2010JM8023）；西北工业大学种子基金资助项目（Z2011133）

中,完全不产生系统调用的应用程序是很少的,所以只能尽量减少用户态和内核态之间来回切换的时间消耗。

2　解决方案

2.1　对象的存储位置

由于进程间的通信需要事件、信号量、互斥体等协调相互通信进程的运行步骤,以实现对共享资源访问的同步与互斥,这些对象的存储位置都在内核地址空间中,Windows 操作系统的内核统一对它们进行管理,所以每次用户程序都需要创建一个事件、信号量和互斥体等对象,都需要用户态向内核态切换,待内核创建完这些对象之后,再从内核态切换回用户态。

在 32 位的 Windows 操作系统中,每个进程都有 4 GB的虚拟内存地址空间,虚拟地址从 0x00000000～0x7FFFFFFF 是用户空间,虚拟地址 0x80000000～0xFFFFFFFF 是内核空间。用户地址空间是每个进程独有的,但是内核地址空间是所有进程共享的。一个信号量对象存储在内核地址空间中,由于内核地址空间是所有进程共享的,即使进行了进程切换,当前进程进入内核后,还是可以访问到被切换出去的进程所创建的信号量对象。根据内核空间的特性,方案在内核空间中申请一片内存区域用来存储信号量对象。如果减少用户态和内核态之间的来回切换,就可以减少切换时间。可是用户态进程不能直接操作内核空间中的信号量对象,所以方案把信号量对象同时映射到用户空间中,用户态进程就能直接访问了。

Windows 操作系统对内存管理[6]时,用户地址和内核地址都是虚拟地址,虽然 2 片地址范围不存在交集,但是对它们的访问最终都映射成访问实际的物理内存地址。Windows 操作系统内核定义了一个结构 MDL(内存描述符表),可以通过使用 MDL 把用户空间的一片内存区域和内核空间的一片内存区域跟同一片物理内存相关联,如图 1 所示。

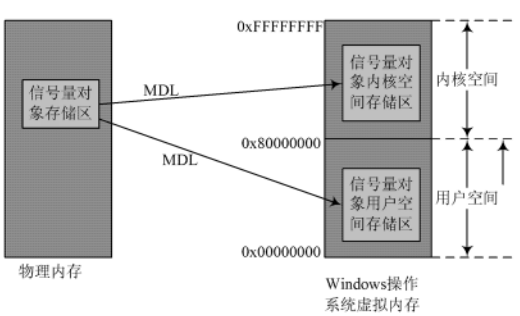

图 1　映射示意图

每当任务申请创建一个信号量对象时,原有的 Windows 操作系统内核,都需要遍历一遍信号量链表,查找有没有同名的信号量对象。如果有,内核就不需要再创建了,但是如果没有,内核就需要申请一片大小为一个信号量对象结构体大小的内存区域,并对其初始化,然后插入到信号量对象链表中。然而申请内存具有不确定性,如果申请不到一片连续的足以存储一个信号量对象结构体大小的内存区域,那么就会造成一些任务从内存换出,然后再把内存分配给对象。这当中的时间开销将会很巨大,因为内存是一个高速设备,而硬盘是一个低速设备。所以方案在还没有线程创建信号量对象之前,先向操作系统申请一片足够存放任务所需信号量对象数量的内存区域。这里所说的"足够存放任务所需信号量对象数量的内存区域"是相对而言,只是针对特定的应用场合,不适合所有情况,所以在申请足够存放任务所需信号量数量对象的内存区域之前,需要分析相应的任务量,不能固定在某个值。内存的换页是造成任务运行不确定的关键因素之一,既然向操作系统申请了一片内存存放信号量对象,就不希望信号量对象从这片内存区域中换出,所以向系统申请内存时,特别指明了要申请一片不分页内存的内存区域,避免了承担换页所造成的时间开销。上面提到,可以使用 MDL,把存储在同一片物理内存的事件、信号量和互斥体等对象分别映射到用户空间和内核空间。由于 MDL 是在内核中使用的,并且信号量对象需要存储在内核空间,所以在内核空间申请一片内存,然后通过 MDL 把这片内存同实际的物理内存关联起来,最后再把物理内存映射到用户空间,所以物理内存总共被映射了 2 次,一次映射到内核空间,另一次映射到用户空间。存放信号量对象的区域在内核空间是固定,但是在用户空间,不同进程使用不同的内存地址存放信号量对象。所以对于不同进程,在每当一个进程需要创建存放信号量对象前,先转到内核,把存放信号量对象的物理内存映射到用户空间,以后对信号量对象的操作,就不用再切换到内核,直接通过对用户空间存放信号量对象的区域进行操作就可以了。为了进一步减少创建信号量对象的时间,把物理内存映射的过程放在每个任务初始化时,在每个任务进程初始化的时候,通过调用 Win 32 API DeviceIoControl[7]来获得当前进程信号量对象内存区域的用户空间地址,这样创建信号量对象也不需要进行用户态和内核态之间的来回切换。通过此方法,节省了进程在用户态和内核态之间来回切换的时间。

2.2　对象结构体的设计

信号量对象是可以跨进程使用的,既然是跨进程使用,那么当一个进程创建了信号量对象,其他进程需要使用这个信号量对象时就要能够准确获得。如果存放信号量对象的内存区域只有一个信号量对象时,要想获得信号量对象就直接获取该信号量对象即可,但是如果

存在多个信号量对象时，每个信号量对象就需要独一无二的标识，可以让不同的进程获得指定的信号量对象。于是方案给信号量对象结构定义了一个存储信号量对象名字的成员变量，创建信号量对象时根据名字创建，搜索时也根据名字搜索。一个信号量能同时被多个线程获得，所以还在信号量对象结构中增加了一个能决定它同时被多少线程获得的成员变量，以及反应在某一时刻还能被多少线程获得的成员变量。同时还增加了一个非常重要的成员变量，因为有删除信号量对象的操作，就需要记录一个信号量被创建了多少次，这是保证一个信号量对象能被安全删除所必需的。

既然一个信号量对象可以同时被多个线程所获得，就有可能出现同时多个线程等待一个信号量对象。对于等待信号量对象的线程有多种处理方法，例如可以让等待信号量对象的线程一直在原地不停地测试信号量对象是否满足，直到某一时刻信号量对象满足，或者让线程休眠一段时间在唤醒，然后再测试信号量对象是否满足，或者直接休眠，直到有线程释放信号量对象，并将等待信号量对象的线程唤醒。对于第一种方法，如果信号量对象很长一段时间得不到满足，那么会造成计算机资源的极大浪费，影响计算机上所有任务的运行效率，所以不是一个好方法。对于第 2 种方法，它与其设定的休眠时间有关，如果休眠时间比较长，虽然不会像第一种方法那样，造成计算机资源的极大浪费，但是有可能造成线程长期得不到信号量对象，比如线程刚休眠结束，再去判断信号量对象是否满足，有可能等待的信号量在其刚唤醒前不久就被其他线程获得，如果休眠时间比较短，就会造成调度器频繁的调度，操作系统切换任务是一个很费时的过程，这样也会造成任务效率不高。第 3 种方法正好具有前 2 种方法所不具有的优点，当信号量对象不满足时，不会浪费计算机的资源，当信号量对象满足时，不用自己醒再去判断，而是由释放信号量对象的线程去唤醒，可以在第一时间获得信号量对象。方案采用第 3 种方法设计信号量机制中获得信号量对象的操作。其中有个唤醒线程的顺序问题，就是多个线程会因等待同一个信号量对象而进入休眠状态。方案采用的方法是唤醒第一个进入休眠状态的线程。由于存在线程因为等待信号量对象而进入休眠状态，所以信号量对象需要记录等待信号量对象的线程 ID，下面会提到信号量对象是如何记录线程 ID 的。

Windows 操作系统有自己的信号量机制[8]，但是 Windows 操作系统的信号量机制缺少必要的实时机制，这也就是一个 Windows 操作系统不是一个实时操作系统的重要原因，即 Windows 操作系统的信号量机制会造成优先级反转。所谓优先级反转就是当一个高优先级任务通过信号量机制访问共享资源时，该信号量已被一低优先级任务占有，而这个低优先级任务在访问共享资源时可能又被其他一些中等优先级任务抢先，因此造成高优先级任务被许多具有较低优先级任务阻塞，实时性难以得到保证。解决优先级反转有多种方法，其中优先级天花板[9]比较常用，优先级天花板就是将申请共享资源任务的优先级提升到可能访问该资源的所有任务中最高优先级任务的优先级。既然要解决优先级反转这个问题，就要知道获得信号量对象的线程与等待信号量对象的线程所具有的优先级。所以在信号量对象结构体中增加了一个结构体成员，该结构体成员中包括分别记录每个获得信号量对象和每个等待信号量对象的线程 ID 和优先级的成员变量。如果有高优先级线程需要信号量对象，但是发现信号量对象不满足，这时候就提高那些已经获得信号量对象的线程的优先级，把它们的优先级提高到正在等待信号量对象的高优先级线程的优先级，一旦低优先级线程释放信号量对象就将其优先级复原。所以记录的优先级不能仅仅是记录获得信号量对象线程的当前优先级，要分别记录提升优先级之前的优先级和提升之后的优先级，没有提升优先级的两个记录一样。

2.3 对象创建流程

方案设计的实时信号量机制的基本使用流程如图 2所示。

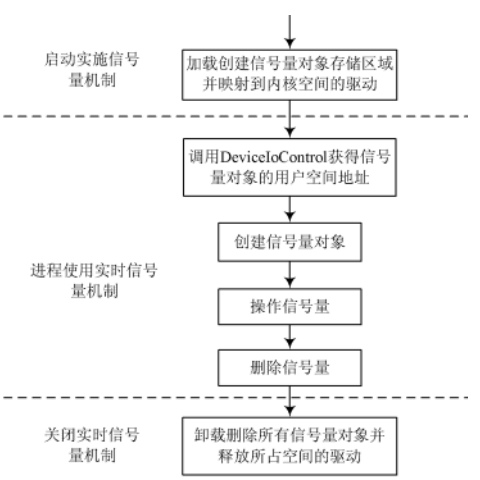

图 2 实时信号量机制的使用流程

3 结 语

Windows 操作系统不是实时操作系统，减少进程间通信的时间和提高进程间通信的稳定性只解决了其中的一个问题，并且各个实时模块需要结合起来，例如实施进程间通信模块需要跟实时调度器[10]结合起来，所以一个完整的实时操作系统是需要多方面考虑的。

（下转第 50 页）

图 4 表明电信 Intranet 企业内网某台带病毒的 PC 机向 Internet 网络发起可执行 UDP 报文攻击，被本文设计的系统成功捕获和封杀。

3 结 语

鉴于 intranet 接入网组网结构复杂，用户接入和应用的多样性，目前很少有网络安全模型可以借鉴，使用集中式内容检测系统不能很好地监控内在的变化因素，使用分布式检测网络将检测中间件置放在网络终端，不但对网络质量产生的影响比较大，而且实施有一定困难。

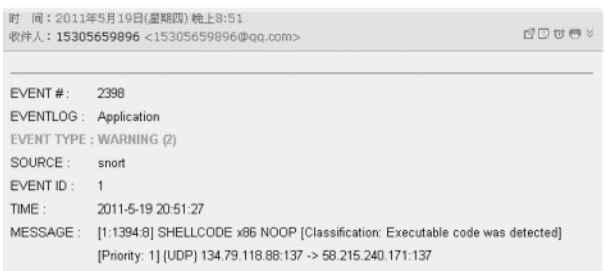

图 4 网络攻击告警

本文给出的 P2TD3R 安全模型，避免上述 2 种安全模型的缺点，既考虑到网络安全系统的完毕性，又考虑了方案实施的可行性，P2TD3R 安全模型对网络和用户而言是完全透明的，几乎不对用户网络产生任何影响。本文提出的 P2TD3R 网络安全模型是对现有动态安全模型的一次重要补充，是解决典型 intranet 接入网良好方法，通过工程案例的实践应用，取得了较为稳定的安全效果。

参 考 文 献

[1] 李颖，刘金刚，李锦涛. 一种面向用户的网络层安全通信模型[J]. 计算机工程，2005(11)：136-138.
[2] 侯小梅，毛宗源. 基于 P2DR 模型的 Internet 安全技术[J]. 计算机工程与应用，2000(23)：1-2.
[3] 陈运明. 动态网络安全模型的系统研究[J]. 网络安全技术与应用，2005(5)：47-49.
[4] 龚俭，董庆，陆晟. 面向入侵检测的网络安全监测实现模型[J]. 小型微型计算机系统，2001(2)：145-148.
[5] SCHNOS M. Internet plagues spread rapidly [J/OL]. [2001-04-02]. http://physics.aps.org/story/v7/st15.
[6] 单蓉胜，王明政，李建华. 基于策略的网络安全模型及形式化描述[J]. 计算机工程与应用，2003(13)：68-71.
[7] 李胜现，田东平，刘建华. 基于改进隐马尔可夫模型的网络动态风险评估[J]. 现代电子技术，2011，34(3)：76-77.
[8] 傅钢善，李婷，马少星. 基于 MANET 的移动学习及其系统构建研究[J]. 现代电子技术，2011，34(3)：42-45.
[9] 赵可新，陈玉芳. 包过滤防火墙系统的设计与研究[J]. 现代电子技术，2011，34(6)：112-113.
[10] 付世凤，梁建武. 基于生物免疫原理的动态入侵检测模型的设计[J]. 电子科技，2008(4)：52-55.
[11] 宦娟，庄丽华. 基于信誉度的移动自组网入侵检测分簇算法[J]. 电子科技，2009(5)：23-27.

作者简介：杨德胜 男，1982 年出生，硕士研究生。研究方向为数据挖掘，人工智能。
张义超 男，1977 年生，硕士。主要研究方向为嵌入式系统，网络安全。
郭 星 男，1983 年生，博士。主要研究方向为智能 EDA，图像处理。
李 炜 女，1969 年生，博士，副教授。主要研究方向为图像处理，计算机视觉，中文信息处理等。

(上接第 42 页)

参 考 文 献

[1] 成杏梅，刘鹏，顾雄礼，等. 支持多线程处理器的实时操作系统实现研究[J]. 浙江大学学报，2009(2)：59-62.
[2] 陈积明. 弱硬实时系统及其调度算法[D]. 杭州：浙江大学，2005.
[3] 段中兴，张德运. 实时多任务操作系统优先级反转与预防[J]. 计算机科学与工程，2005(9)：11-15.
[4] 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(11)：62-65.
[5] 葛仁北. 系统调用与操作系统安全[J]. 计算机工程与应用，2002(7)：32-35.
[6] 潘爱民. Windows 内核原理与实现[M]. 北京：电子工业出版社，2010.
[7] 张帆，史彩成. Windows 驱动开发技术详解[M]. 北京：电子工业出版社，2009.
[8] 毛德操. Windows 内核情景分析[M]. 北京：电子工业出版社，2009.
[9] 胡国珍，范生凯. 嵌入式 RTOS 优先级天花板协议研究[J]. 计算机工程与设计，2009(6)：45-47.
[10] 杜旭，胥海鹏. Linux 操作系统调度器实时性能的研究与改进[J]. 计算机工程，2005(3)：12-14.

作者简介：钟 剑 男，1986 年出生，硕士研究生。主要研究方向为嵌入式计算与应用。