• 体系结构与外围设备 •

实时控制系统中优先级变化问题的研究

陈飞云[1]， 龙 风[2]

(1. 上海应用技术学院 计算机科学与信息工程学院，上海 200235；
2. 中国科学院 等离子体物理研究所，安徽 合肥 230031)

摘 要：分析并研究了在实时控制系统中由于采用优先级抢占式调度策略而可能导致优先级倒置问题和优先级改变现象。根据实时控制系统实际运行情况，给出了相应的解决措施。最后，通过多次实际测试表明，针对实时控制系统采用的调度策略和优先级变化问题的解决措施是有效的，能够保证在程序运行期间，所有进程都能及时调度运行。

关键词：实时控制系统；优先级倒置；优先级改变；优先级抢占；系统分析程序

中图法分类号：TP302.7 **文献标识码**：A **文章编号**：1000-7024 (2010) 22-4839-04

Study of priority change in real-time control system

CHEN Fei-yun[1], LONG Feng[2]

(1. Shanghai Institute of Technology Engineering, Shanghai Institute of Technology, Shanghai 200235, China;
2. Institute of Plasma Physics, Chinese Academy of Science, Hefei 230031, China)

Abstract: Firstly, two reasons are analyzed and studied, resulted in priority change in real-time control system, which adopted preemptive priority scheduling algorithm. And then some kinds of useful methods are offered to handling the problem of priority change, according to the actual operation of real-time control system. Some experiments prove, those methods can work successfully and availably.

Key words: real-time control system; priority inversion; priority change; preemptive priority; system analysis program

0 引 言

EAST(experimental advanced super-conducting tokamak)超导托卡马克聚变实验装置是国家九五重大科学工程。极向场电源控制系统是托卡马克主要子系统之一，它为等离子体的产生、约束、维持、加热，以及等离子体的电流、位置、形状、分布和破裂的控制，提供必要的工程基础和控制手段。整个电源控制系统由三层网络组成：Windows 监测层、QNX 实时控制层、现场总线执行层。QNX 实时控制层选用 QNX 实时操作系统，担负着电源系统多变量反馈控制、系统连锁控制、保护及各种运行方式的调配，决定了整个系统的实时性和同步性，由电源主控制节点、电源子控制节点、数据库节点等节点组成[1]。

QNX 中的进(线)程调度策略具有一定的复杂性，另极向场电源控制系统是一个多进程、多线程并涉及中断调用、内核调用、设备驱动等复杂应用程序。利用系统分析程序[1-3]截取内核有关电源控制系统在运行时数据，多次研究分析，发现有任务没有在特定时限内得到处理，即有进程没有得到及时调度。这对实时性和可靠性要求极高的电源控制系统而言是极其危险的。查阅有关文献[4-7]，发现若存在多个任务，因为竞争共享的临界资源而可能使各个任务之间存在同步关系，进而导致优先级倒置问题。另外，电源控制系统中各个进程间的同步采用消息传递和 pulse，按照客户/服务器模式来解决，而这种模式也会导致进程优先级改变现象。

本文着重对优先级倒置问题和优先级改变现象进行深入分析，并根据实际情况提出解决方法。最后，利用系统分析程序对整个电源控制系统的调度情况进行检测，以验证其调度策略是否合理以及所提出的解决方法是否有效。

1 优先级倒置问题

1.1 问题分析

优先级倒置是指一个任务等待比它优先级低的任务释放临界资源而被阻塞，如果这时有中等优先级的就绪任务，阻塞会进一步恶化，甚至超过了任务的最后期限。具体情况如图 1 所示。图 1 中任务 A 的优先级高于任务 B 的优先级，当 B 执行到 T1 时刻获得临界资源 s1 而进入临界区。在 T2 时刻，由于导致任务 A 执行的事件发生，B 被 A 抢占，当运行到 T3 时，A 申请临界资源 s1，而该临界资源却被 B 占有，故 A 被堵塞等待 B 释放临界资源 sl，直到在 T4，B 释放临界资源 s1，A 才得到调度执行。在等待释放临界资源 sl 的这一时段，高优先级

收稿日期：2010-01-05；修订日期：2010-03-05。

作者简介：陈飞云 (1975－)，女，博士，讲师，研究方向为实时控制、系统分析、计算机应用技术； 龙风 (1975－)，男，工程师，研究方向为系统结构与性能测量。E-mail：cfy@sit.edu.cn

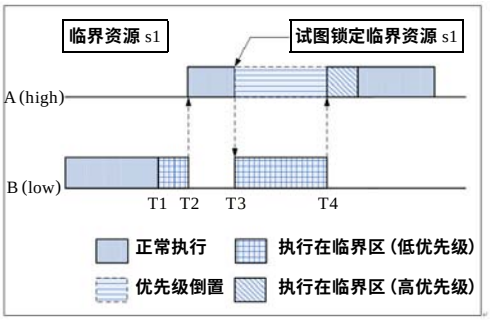

图 1 优先级倒置

任务 A 被低优先级任务 B 堵塞而造成了优先级倒置现象。

另外在某些情况下,高优先级任务 B(high)不仅仅被占有与其竞争临界资源的低优先级 D(low)任务堵塞,而且会被其它优先级介于的 B 和 D 的任务 C(medium)堵塞,从而造成无限优先级倒置,如图 2 所示。

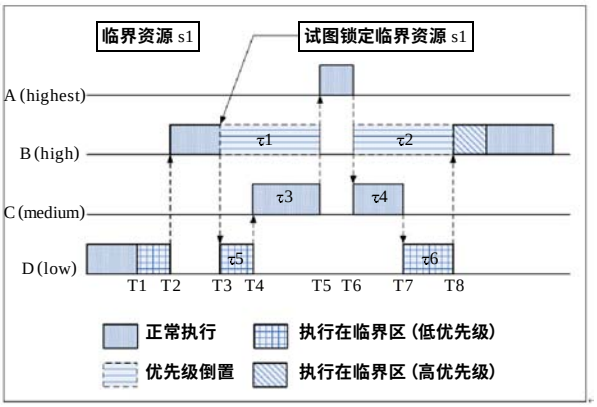

图 2 无限优先级倒置

在图 2 中,任务 A 优先级最高,任务 D 优先级最低。D 在 T1 时刻获得临界资源 s1,在 T2 时刻 D 被 B 抢占,在 T3 时刻,B 由于无法获得已被任务 D 占有的临界资源 s1 而堵塞,使得任务 D 又得到调度执行。在 T4-T7 期间 D 被优先级高于 D 的任务 C 和 A 抢占。在 T8 时刻,D 释放临界资源 s1 后 B 才获得临界资源 S1 得以调度执行。B 被优先级低于 B 的任务堵塞的优先级倒置时间为 $\tau1+\tau2=\tau3+\tau4+\tau5+\tau6$,其中 $\tau3$,$\tau4$ 是被没有与 B 有同步关系的任务 C 堵塞,使得 B 由于优先级倒置被堵塞的时间大大延长,这就造成了无限优先级倒置现象,特别是当存在多个优先级介于 B 和 D 的任务时或者在该中间优先级任务执行时间很大时,B 任务将被无限延后。无限优先级倒置大大影响了高优先级实时任务的执行,在实时系统中必须要避免。

1.2 解决方法

在 QNX 操作系统中,针对这种无限优先级倒置问题可以采用优先权继承办法来解决,如图 3 所示。

当任务 B 于 T3 时刻试图锁定临界资源 s1 时,由于临界资源 s1 已经被任务 D 占有,B 任务被堵塞,这时占有临界资源 s1 的低优先级任务 D 的优先级被提升为任务 B 的优先级 high。当 T4 时刻导致任务 C 的事件发生时,由于优先级高于任务 C 的任务 D 在执行(任务 D 的优先级已经继承任务 B 的优先级被提升为 high),任务 C 的执行被推迟。这样就避免了任务 B 被优先级介于 B 和 D 的任务 C 堵塞的时间,从而避免了无限优先级倒置。

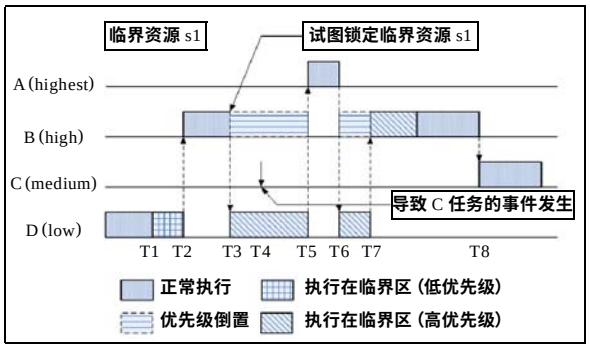

图 3 优先权继承算法解决无限优先级倒置

但当有多个任务竞争临界资源 s1 且它们的优先级相对接近时,利用优先权继承法,可能会导致过多的上下文切换,降低任务运行效率。针对这个问题可以采用优先权极限方法来解决优先级倒置问题。在优先权极限方案中,系统把每一个临界资源与 1 个极限优先权相联系,这个极限优先权等于系统此时最高优先权加 1。当 1 个任务取得临界资源时,系统便把这个极限优先权传递给这个任务,使得这个任务的优先权最高;当这个任务释放这个临界资源时,系统立即将它的优先权恢复原先值,从而避免系统出现优先级倒置问题。

1.3 实际应用设计

由于任务占用临界资源的不可剥夺性,要彻底排除优先级倒置问题是不可能的,所能做的只是把高优先级任务的等待时间局限到只需等待当前占用资源的任务释放该项资源。只要能把等待时间限制在这个程度,那该是可以接受的,因为对于允许任务占用临界资源的时间长度毕竟是有限的。

对于电源控制系统,为防止出现优先级倒置问题,在硬件上要尽量避免出现临界资源情况,同时合理安排本电源控制程序同 QNX 系统例程的优先级关系,尽量提高控制程序各个进程的优先级。但对于不可避免要出现的临界资源,就采用优先权继承法或优先权极限法来尽量避免优先级倒置问题带来的不良后果。假设系统中有 3 个线程需要调用同一段代码,该代码段可以看成是关键代码段(或某一资源)。为了避免优先级倒置,需要使用互斥体来同步各个线程,在进入关键代码段时,需要给互斥体上锁,使用 pthread_mutex_lock(pthread_mutex_t * mutex),该函数将使得每一个进入该代码段的线程都试图为该代码段上锁。当退出该代码段时应该给互斥体解锁,使用 pthread_mutex_unlock(pthread_mutex_t * mutex)。对于优先权继承或优先权极限两种方法的设置是通过设置互斥体的属性来实现的,操作步骤如下:

(1)通过函数 pthread_mutexattr_init(pthread_mutexattr_t * attr)生成互斥体属性的结构;

(2)使用函数 pthread_mutexattr_setprotocol(pthread_mutexattr_t * attr, PROTOCOL)设置使用优先权继承还是优先权极限,如:

pthread_mutexattr_setprotocol (&mutex_attr, PTHREAD_

PRIO_INHERIT)；//设置为优先权继承

pthread_mutexattr_setprotocol (&mutex_attr, PTHREAD_PRIO_PROTECT)；//设置为优先权极限

(3)若设置为优先权极限，通过以下函数来设置最高优先级；

pthread_mutexattr_setprioceiling(&mutex_attr,50)；//最高优先级设为 50

(4)最后，根据互斥体的属性，通过pthread_mutex_init(pthread_mutex_t * mutex, pthread_mutexattr_t * attr)初始化互斥体，这样就可以使用此互斥体来防止优先级倒置了。

2 优先级改变现象

2.1 现象分析

电源控制系统利用 QNX 操作系统的实时性和 Windows 操作系统的友好性，同时引入现场总线技术，形成三层拓扑结构，实现了一个控制周期为 1 毫秒的分布式、实时计算机系统。同时为了提高计算机处理效率、便于升级替换软件模块，每个控制子节点采用多进程和多线程软件设计方法，这就涉及到各个控制子节点内部进程或线程间同步的问题。对于该问题的解决主要是利用消息传递和 pulse，按照客户/服务器模式来进行的。

但为了对客户及时提供服务，服务器进程的运行需要具有与客户进程相同或更高的优先级。其结果，服务器进程所使用的优先级可能会高于其所有客户进程的优先级。如果一个低优先级的客户进程向高优先级的服务器进程发送了一个消息，那么该请求将由服务器进程以较高的优先级进行处理。这种由于客户进程的请求是引起服务器进程运行的原因，这种做法间接地提高了客户进程的优先级[8]。

如果服务器进程完成客户进程请求的服务所用的时间很短，问题不至于很大；但如果服务器进程用了较长时间来响应客户进程的服务请求，那么低优先级的客户进程就有可能因此影响其它优先级比其高但又比服务器进程低的进程的运行。具体分析如图 4 所示。

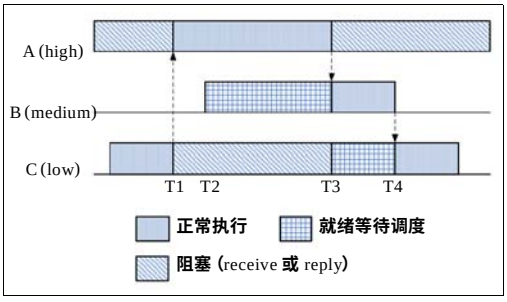

图 4 低优先级进程影响高优先级进程运行

如图 4 所示，低优先级进程 C(客户进程)在 T1 时刻向高优先级进程 A(服务器进程)发送请求服务信号，进程 A 则以高优先级在 T1 时刻开始运行；而在 T2 时刻中优先级进程 B 就绪，却由于高优先级 A 的运行处于等待调度，直到 T3 时刻进程 A 运行完毕处于 Receive 阻塞状态，进程 B 才得以运行；在 T4 时刻进程 B 运行结束，进程 C 继续运行。可见，由于客户进程优先级间接受服务器进程的影响，进而导致优先级处于客户进程和服务器进程之间的进程 B 在 T2 到 T3 时刻处于就绪等待调度状态。

2.2 解决措施

为了解决这个问题，QNX 允许采用由发送消息的客户进程的优先级决定服务器进程的优先级的办法。当服务器进程收到一个消息时，便将自己设置为与客户进程同样的优先级。如果该服务器进程在运行期间又收到另一消息，且新客户进程的优先级比服务器进程的优先级高，则该服务器进程的优先级将被进一步提高。具体分析如图 5 所示。

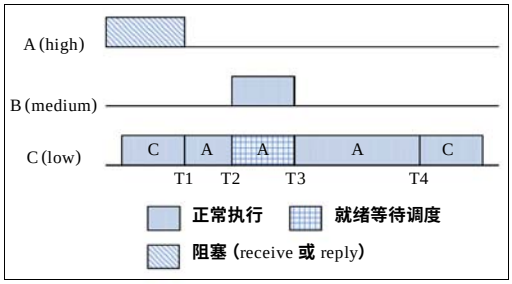

图 5 客户进程决定服务器进程的优先级

如图 5 所示，低优先级进程 C(客户进程)在 T1 时刻向高优先级进程 A(服务器进程)发送请求服务信号，进程 A 的优先级由进程 C 决定，使得进程 A 以低优先级在 T1 时刻开始运行；而在 T2 时刻中优先级进程 B 就绪，抢占进程 A 运行，使得进程 A 处于等待调度状态，直到 T3 时刻进程 B 运行完毕，进程 A 才得以继续运行；在 T4 时刻进程 A 运行结束，进程 C 继续运行。客户进程决定服务器进程的优先级，不影响中优先级进程 B 的运行。

对于极向场电源控制系统，其中各个控制子节点中的每个进程都负责一项实时任务，并根据任务的缓急分配不同的优先级。希望在程序整个运行过程中，进程的优先级是固定不变的。即不希望采用 QNX 默认的客户进程决定服务器进程的优先级办法来解决低优先级进程影响高优先级进程运行这样的问题。对于这个问题的解决采用以下方法：

(1)服务器进程利用函数 ChannelCreate(unsigned flags)在建立通道时，flags 参数指定为_NTO_CHF_FIXED_PRIORITY，这样服务器进程在收到客户进程发来的信号时，优先级不会随之而变化；

(2)利用消息传递和 pulse 只是为了程序内部进程间的同步和周期性循环，因此设立定时器进程为客户进程，优先级最高，而其它进程均为服务器进程，优先级随任务的缓急有所不同。

如主控制节点程序，timer 进程作为消息传递中的客户进程，同时根据其它各个进程(服务器进程)优先级大小按先后顺序周期性发送同步信号。而其它各个进程只有在收到 timer 进程发来的同步信号后才能继续运行。Timer 进程的优先级最高，为 30。建立通道部分具体代码如下：

服务器代码：

```
/******connection with timer process using channel*****/
timer_chid = ChannelCreate(_NTO_CHF_FIXED_PRIORITY);
if(timer_chid < 0){
    pf_perror("ChannelCreate\n", __FILE__, __LINE__);
```

```
        return;
    }
    //sychronization with timer
    timer_rcvid = MsgReceive(timer_chid, &init_recv, 1, NULL);
    if ((timer_rcvid > 0) && (init_recv IS 1)){
        init_send = 1; //this process init successfully
        MsgReply(timer_rcvid, 0, &init_send, 1);
    }
    else return;
```

客户代码：

```
    /***********connect to ps_manage process***********/
    for( ; ; ){
        ps_manage_coid = connectAttach (0, ps_chid.pid, ps_
chid.chid, 0, 0);
        if (ps_manage_coid IS ERROR){
            pf_perror("connect to ps manage channel id");
            delay(10);
            continue;
        }else{
            MsgSend(ps_manage_coid, &timer_init_send, 1, &
timer_init_rcv, 1);
            break;
        }}
```

3 结果验证

为了保证电源控制系统运行的安全可靠与实时，程序运行时的真实调度情况是否与程序设计时希望情况一致是需要准确验证的。此前的验证方法是在程序的每个进程中添加 IO 输出点，当某个进程处于运行状态时，则向 IO 卡输出高电平，否则为低电平。同时利用示波器记录下来，就可得到程序运行时各个进程执行的时序，这样可以间接反映进(线)程的调度情况。但这种方法具有以下局限性：示波器输出数量及检测通道的限制，使得每次只能记录固定数目个进程执行的时序；添加了额外的硬件 IO 输出和软件编程。因此利用这种验证方法是不能准确反映整个系统的真实调度情况。

以下就利用系统分析程序截取内核有关数据来验证极向场电源控制系统进程调度策略。

测试环境：两台型号为 CPCI-3840 (PM-1.7G/512M/40G) Compact PCI 主机，用来运行极向场电源控制系统 ps_con 程序，以及系统分析程序 subsat 部分程序；一台内存为 512M，CPU 为 P4-1.0G 工控机，用来运行数据分析图形界面；一台型号为 3com 4400SE-3C17206 10/100Mbps 交换机。

每次启动程序时产生的进程号是不同的，做测试时 ps_con 程序的进程名、进程号以及优先级如图 6 所示。

ps_con 程序:

进程名	进程号	进程优先级
init	634906	20
log_sig	639009	21
output	639011	22
ps_manage	639003	23
fault	639010	24
ds_fieldbus	639007	25
qp_fieldbus	639006	26
snu_fieldbus	639005	27
cu_fieldbus	639004	28
com	639008	29
timer	639012	30

图 6 进程名、进程号以及进程优先级对应

由于 ps_con 程序所有进程是以 1ms 为工作周期运行的，就以 1ms 为周期来分析数据。根据某 1ms 数据做如图 7 所示的进程运行顺序和状态改变图。

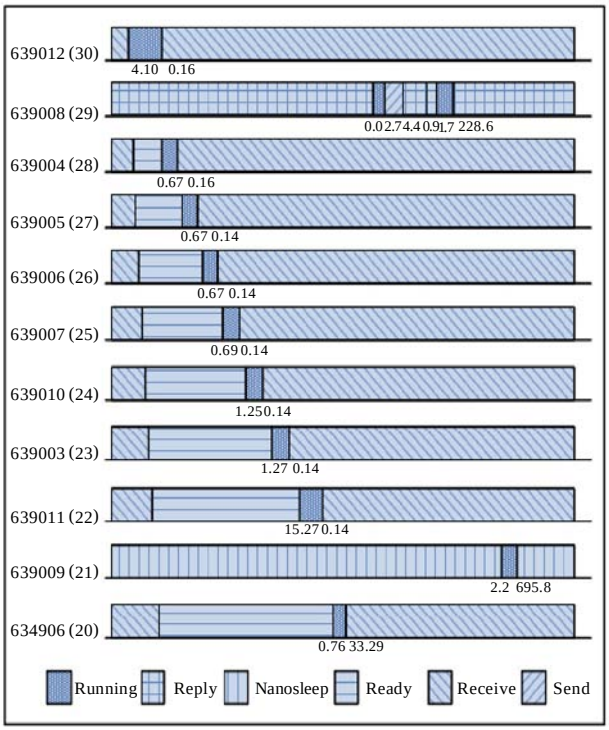

图 7 ps_con 程序进程运行顺序与状态改变

在图 7 中，每个线程状态框底下前面的数字表示该线程在该状态下实际运行时间，后面的数字表示上下文切换时间，单位均为微秒。分析图 7，可以得出如下结论：

(1) 整个截取数据是以 1ms 为周期循环出现，符合 ps_con 程序的以 1ms 为工作周期的循环运行规则；

(2) 所有进程在每 1ms 周期内都得到调度运行，且运行时间都在 1ms 内；

(3) 当进程处于 READY 状态后，运行时序基本按照优先级高低顺序执行，但也出现例外现象，如进程 639008(通信进程)和进程 639009(记录进程)没有按优先级高低顺序执行。

对于结论 3，结合 ps_con 程序原码和原始数据进行分析。通信进程之所以没有按照优先级高低顺序执行，是由于通信进程的运行还要看其是否接收到其它程序发来的组播数据包。只有在通信进程接收到数据包后，才解除 Reply 阻塞状态，并按优先级高低顺序执行。记录进程同其它进程不同，它的循环不受 timer 进程控制而是由函数 delay() 来控制，因此其运行不完全受优先级高低控制。通过以上数据的分析，总的来说，ps_con 控制程序的运行性能基本满足系统要求，其调度策略的设计和各个进程优先级的设置是合理可取的。(下转第 4846 页)

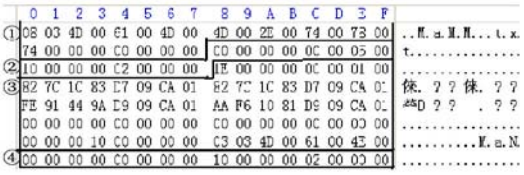

图 9 删除 MaN 索引项后的索引缓冲区

重新添加到目录 dir_3 中后，其索引项作为第一个记录存储在 MaN 的索引项指向的 VCN 为 1 的索引缓冲区中。图 10 中的区域①为索引头结构；区域②为新增加的 MbM.txt 的索引项。

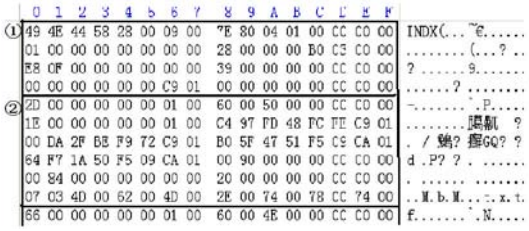

图 10 添加索引项 MbM.txt 后的索引缓冲区

2.3 复制与移动操作分析

文件的复制过程可看作是在新的磁盘空间创建一个相同的文件。其具体过程参照前面所讲的创建操作。文件的移动可以分为两种情况：第 1 种情况为同分区移动；第 2 种为跨分区移动[9]。同分区移动时，目标文件的 MFT 记录和数据存储位置都不改变，只是 MFT 记录中的 0x30 属性中相应区域发生变化。试验中，将目录 dir_3 下的文件 p.xls 移动到同分区下的目录 Pdir 下，将引起如下相应的结构变化：

(1)移动前后文件 p.xls 的 MFT 中 0x30 属性中父目录的文件记录号、最后一次 MFT 访问时间和最后一次的访问时间字节发生改变；

(2)文件 p.xls 的索引项在目录 dir_3 的索引缓冲区中被删除；

(3)文件 p.xls 的索引项被增加到目录 Pdir 的 MFT 中。

跨分区移动时，则如同在新分区创建文件，并在原分区删除相应的文件。此操作可参照文件的创建和删除操作。

3 结束语

本文在现有文献基础上分析具体磁盘上的 NTFS 目录结构，根据大目录的存储结构在磁盘上实现的规律，提出三级大目录概念，深入了解 NTFS 目录中索引 B+树结构的变化和各种目录之间的联系。并在此基础上对相关文件操作下目录结构的变化进行动态分析，对比实验数据，总结了 NTFS 目录结构在磁盘上的变化规律。通过在非 Windows 环境下进行 NTFS 的应用开发，表明在此分析理论基础上可以实现直接对磁盘的 NTFS 文件系统进行操作。

参考文献：

[1] Microsoft TechNet. Optimizing NTFS [EB/OL]. http://technet.microsoft.com/en-us/library/cc767961.aspx,2010.

[2] 戴士剑,涂彦晖.数据恢复技术[M].北京:电子工业出版社,2005:223-303.

[3] Richard Russon, Yuval Fledel. NTFS documentation [EB/OL]. http://www.linux-ntfs.org/content/view/104/43,2009.

[4] Brian,Carrier.File system forensic analysis[M].Boston:Pearson Education Inc Press,2005:273-396.

[5] David A Solomon,Mark E Russinovich.Inside Microsoft Windows 2000[M].US:Microsoft Press,2000.

[6] 居锦武, 王兰英. NTFS 文件系统剖析 [J]. 计算机工程与设计, 2007,28(22):5437-5460.

[7] Alex Ionescu.NTFS on-disk structures:Visual basic NTFS programmer 's guide[EB/OL]. http://www.alex-ionescu.com,2009.

[8] NTFS Research Group. Disk scan for deleted entries [EB/OL]. http://www.ntfs.com/disk-scan.htm,2009.

[9] 黄步根. NTFS 系统存储介质上文件操作痕迹分析[J]. 计算机工程,2007,33(33):281-283.

(上接第 4842 页)

4 结束语

在本文，分析研究了在多任务实时控制系统中可能存在的两种优先级变化问题，由于多个进程因为竞争共享的临界资源而使各个进程之间存在同步关系，进而所导致的优先级倒置问题，和由于多个进程采用客户/服务器模式来解决同步问题所导致的优先级改变现象，并根据实际情况提出相应解决措施。通过对电源控制系统多次实际测试和实际运行验证，所采用的调度策略是合理可取的，优先级变化问题解决措施是有效的。文中所提出的优先级变化问题研究方法和解决措施有利于提高实时系统的实时性和可靠性，具有一定的实际工程意义。

参考文献：

[1] 陈飞云.EAST 极向场电源实时控制系统——系统分析程序的实现与应用[D].合肥:中国科学院合肥物质科学研究院等离子体物理研究所,2007.

[2] 陈飞云,翁佩德,傅鹏.基于 QNX 的 SAT 系统的研究与实现[J].化工自动化及仪表,2007,34(2):39-42.

[3] QNX Momentics Development Suite. System analysis toolkit user's guide[M].Canada:QNX Software system Ltd,2003.

[4] 钟利明.RTLinux 系统实时性能评测及相关问题的研究[D].中科学院计算技术研究所,2005.

[5] 段中兴,张德运.实时多任务操作系统优先级反转与预防[J].计算机工程与科学,2005,27(2):62-64.

[6] 陈明俊,钟昊,王毅.关于实时事务调度中的优先级反转[J].计算机工程与应用,2003,39(29):122-124.

[7] 张辰,王自强,都思丹.硬实时操作系统优先级倒置的解决[J].微处理机,2005(1):23-25.

[8] Rob krten.The QNX Cookbook:Recipes for programmers[M]. Canada:QNX Software Systems Ltd,2004.