****

|  |  |
| --- | --- |
| **论文题目：** | 嵌入式实时操作系统中断处理概况 |
| **专 业：** | 计算机科学与技术 |
| **学生姓名:** | 史奎锐 |
| **学生学号：** | 3200604001 |
| **指导教师：** | 郜东瑞 |
| **成 绩：** |  |

2021 年 3 月 26 日

成都信息工程大学

计算机学院

**嵌入式实时操作系统中断处理概况**

**摘要**：在嵌入式系统的应用开发中，采用嵌入式[实时操作系统](https://baike.baidu.com/item/%E5%AE%9E%E6%97%B6%E6%93%8D%E4%BD%9C%E7%B3%BB%E7%BB%9F)（简称RTOS）能够支持多任务，使得程序开发更加容易，便于维护，同时能够提高系统的稳定性和可靠性。实时系统对中断的支持，不仅满足了多任务的要求，而且能够保证系统能够及时处理紧急任务，达到实时性要求。本文分别介绍了四种主流嵌入式实时系统的中断处理机制。

**0 引言**

随着计算机技术的迅速发展和芯片制造工艺的不断进步，[嵌入式系统](https://baike.baidu.com/item/%E5%B5%8C%E5%85%A5%E5%BC%8F%E7%B3%BB%E7%BB%9F)的应用日益广泛：从民用的电视、手机等电路设备到军用的飞机、坦克等武器系统，到处都有嵌入式系统的身影。而在工业控制、军事设备、航空航天等领域对系统的响应时间有苛刻的要求，这就需要实时嵌入式操作系统。本文将就四种不同的实时系统，包括RT-LINUX、μC/OS-Ⅲ、FreeRTOS、Vx Works等的中断机制分别做出介绍。

**1 四种嵌入式实时系统的中断机制**

**1.1 RT-LINUX中断处理机制**

实时抢占补丁( PＲEEMPT\_ＲT) 的核心思想是“提高内核本身的可抢占性，即让代码的一部分尽可能的可抢占”。 在原来的低延迟补丁和抢占补丁的基础上，实时抢占补丁又引入了对中断进行线程化、高精度的时钟、临界区的可抢占以及优先级继承等关键性技术。为满足用户不同的实时性需要，实时抢占补丁设置了四种不同的内核配置选项。

1. **中断线程化**

在Linux标准内核中，中断被设为最高优先级，中断一旦触发，系统就会立刻对其响应，且在中断发生到中断返回这段期间，由于执行中断处理例程，实时任务无法得到调度。若中断频繁发生，系统将一直忙于对中断进行响应，实时任务很有可能无法执行。针对这个问题，实时抢占补丁对中断进行了线程化处理，让中断处理程序转为可调度的实体，以内核的一个线程的身份来运行，同时为其设置不同的实时优先级。通过这种方式，高优先级的实时任务可以优于中断线程执行，确保了系统的实时性。

1. **高精度时钟**

在 Linux 系统中，当调度短周期任务时，时钟周期设置得太小，会造成时钟中断频繁，而时钟中断不能被线程化，会严重影响系统的实时性。为解决这个问题，实时抢占补丁实现了高精度的定时器，时钟的精度取决于具体硬件所提供的精度。只有硬件的精度够高，就可以获得 ns 级的时间。

1. **临界区可抢占**

为保障数据结构不被破坏，Linux系统中存在大量的临界区，包括自旋锁、大内核锁等，由于它们被禁止抢占，很大程度上削弱了系统的性能。为确保这些锁可被抢占，实时抢占补丁将它们转变为了优先级继承的互斥锁。转变后，互斥锁保护的临界区可以被没有共享资源的其他高优先级任务抢占。

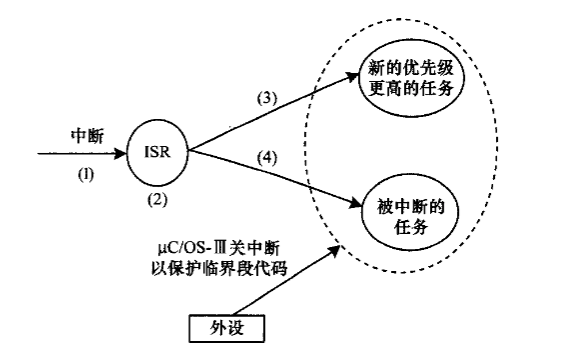
1. **优先级继承**

在上文中，自旋锁转化为互斥锁，会发生优先级的反转现象，即由于低优先级的进程保持了竞争资源而使得高优先级的进程被迫等待，从而使优先级处于中间的进程得以执行。高优先级进程会由于中间优先级的进程运行时间的不稳定性造成抢占延迟增大。为消除这种现象，确保系统的实时性，实时抢占补丁实现了优先级的继承协议。保持自旋锁的进程的优先级继承自高优先级竞争者，进而能早于中间优先级进程执行，并尽可能快地释放锁，从而高优先级进程能尽快获得竞争的自旋锁，使得抢占延迟更确定、更短。

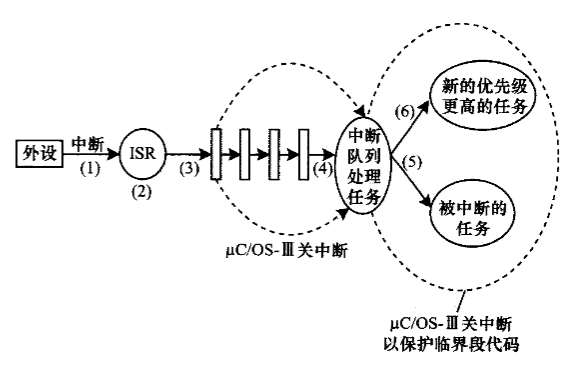
**1.2 μC/OS-Ⅲ中断处理机制**

μC/OS-Ⅲ采用2种中断管理模式，在系统初始化时选择其中的一种。一种是传统中断管理模式， 即中断以ISR方式处理，在当前任务执行的过程中，只要没有关硬件中断，无论何种级别的IRQ发生就立即处理，也称为直接处理模式; 另一种是以 IST方式处理，在IRQ发生后先进入ISR，向中断队列 OS\_ INT\_Q 发送信号，退出ISR直接调度中断队列处理任务OS\_IntQTask，即延时处理模式。

μC /OS-Ⅲ的2种中断管理模式如图1和图2所示。在直接模式中，外设产生 IRQ 并调用 ISR，ISR 处理该IRQ的实际工作，这个过程中ISR可能会激活一个任务，该任务优先级可能比当前运行任务的优先级高，也可能低。退出ISR之后会进行任务调度，ISR激活任务过程中系统关中断。



**图 1 直接处理模式**

****

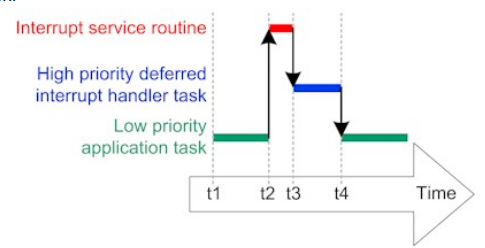
**图 2 延时处理模式**

在延时处理模式中，外设产生IRQ后，依旧调用ISR，但不在ISR中激活新任务，ISR主要是向中断队列OS\_INT\_Q发送信号量，有内核服务参与的工作由中断处理任务OS\_IntQTask 来处理，OS\_IntQTask 是系统中优先级最高的任务，一旦ISR退出，系统就立即调度OS\_IntQTask，该任务主要是向新任务发送内核消息(如信号量)来激活新任务。该模式本质是把ISR分为两部分处理，前半段是处理没有涉及内核服务参与的工作，后半段向OS\_INT\_Q发送内核信息，激活新任务的工作由OS\_IntQTask完成，并且ISR只在向OS \_INT\_Q发送内核信息时关中断，即在图2中时刻(4)关中断，这样即可减少关中断时间。

**1.3 FreeRTOS中断处理机制**

FreeRTOS中同样采取延迟中断处理。延迟中断处理通常包括记录中断的原因并在ISR中清除中断，但随后解除对RTOS任务的阻止，以便中断所需的处理可以由未阻塞的任务执行，而不是在ISR中执行。

如果中断处理被延迟的任务被分配了足够高的优先级，那么ISR将直接返回到未阻塞的任务（中断将中断一个任务，然后返回到另一个任务），从而导致中断所需的所有处理都在时间上连续执行（没有间隔），就像所有的处理都是在ISR中进行的。如图3所示，所有的中断处理都发生在t2和t4之间，即使部分处理是由任务执行的。



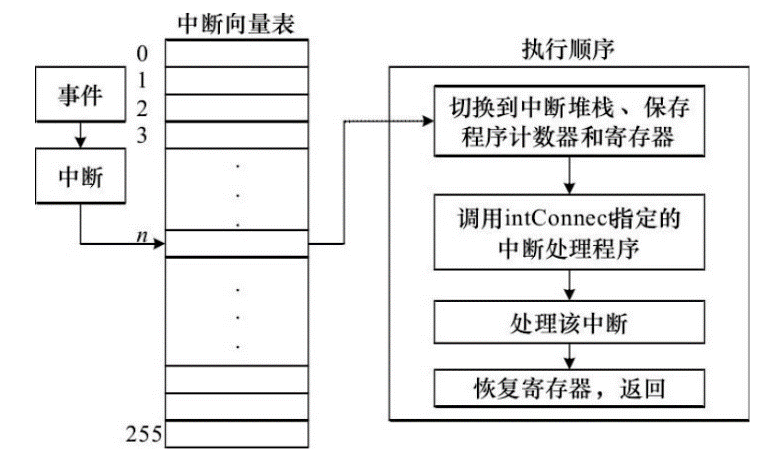
**图3 当延迟处理任务具有高优先级时延迟中断处理执行序列**

可以将中断处理延迟到任务的方法分为两类：

1. 集中式延迟中断处理：使用此方法的每个中断都在同一个RTOS守护进程任务的上下文中执行。RTOS守护进程任务由FreeRTOS创建，也称为计时器服务任务。
2. 应用程序控制的延迟中断处理：使用此方法的每个中断都在应用程序编写器创建的任务上下文中执行。

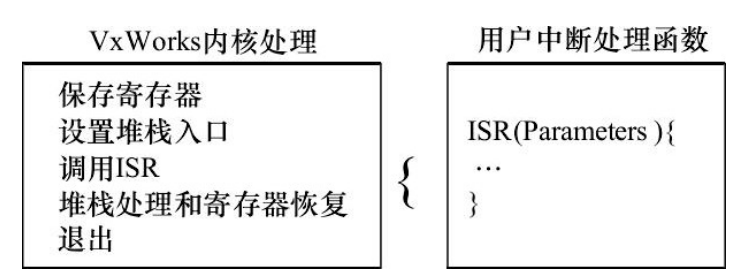
**1.4 Vx Works中断处理机制**

为了获得尽可能快的中断响应时间, Vx Works的中断处理程序运行在特定的上下文中, 它在所有任务上下文之外。因此, 中断处理不会涉及任何任务上下文的交换。Vx Works的库int Lib和int Arch Lib提供了中断处理的相关函数。对于使用了内存管理单元的底板, Vx Works还提供了可选产品Vx VMI以提供中断向量写保护机制。通常情况下中断处理过程如图4所示。

****

**图4 Vx Works中断处理过程**

应用程序可以使用中断向量表中未使用的中断号资源。Vx Works提供函数int Connect() , 它允许将指定的C函数与任意中断相联系。该函数将指定的C函数routine与指定的中断向量vector相联系, 函数的地址将存储在这个中断向量里。所以, 这个中断发生时, 系统将调用该函数, 使用指定的参数parameter作为参数。中断处理程序在中断级以supervisor方式调用, 并建立一个合适的C环境, 保存必要的寄存器, 建立堆栈。中断处理函数可以是任何正常的C代码。但是它必须保证不调用任何可能引起阻塞和执行I/O操作的函数。该函数简单地调用int Handler Create() 和int Vec Set() 。处理程序的地址由int Handler Create() 函数返回, 保存在中断向量中。int Connect() 将创建一小段代码, 用以保存必要的寄存器, 设置堆栈入口, 包含将要传递的参数, 在堆栈中调用这个连接函数。相反, 当从该函数返回时, 这段代码先恢复寄存器和堆栈, 然后退出中断, 如图5所示。



**图5 用int Connect() 构造的用户中断处理函数**

**2 结语**

本文从四种主流的嵌入式实时操作系统入手，分别介绍了其中断机制。当然，这些系统应用到工业生产中时，根据不同的行业标准和需求，都可能会在其原有中断机制的基础之上，做出相应的改进。随着技术的发展成熟，也有相应的研究分别从硬件和软件算法上改进中断系统，以此来提高中断响应和处理能力。在以后的学习研究中，我也会更多的关注在相应的算法和硬件技术上。

参考文献：

1. 董艳雪,韩卫光.基于多核ARM的Linux操作系统的实时性研究[J].小型微型计算机系统,2017,38(06):1262-1266.
2. 袁志祥,甘正良.基于μC/OS-Ⅲ的实时内核中断管理机制[J].计算机工程,2015,41(11):100-105.
3. Richard Barry. [Mastering the FreeRTOS Real Time Kernel – a Hands On Tutorial Guide](https://www.freertos.org/fr-content-src/uploads/2018/07/161204_Mastering_the_FreeRTOS_Real_Time_Kernel-A_Hands-On_Tutorial_Guide.pdf)[M/OL]. (2016)[2016-12-04]. <https://www.freertos.org/Documentation/RTOS_book.html.>
4. 唐晓平,何峰,梁甸农.基于VxWorks的多中断处理设计[J].计算机工程,2009,35(13):249-251.