**改善嵌入式Linux实时性能的方法研究**

　　摘要：分析了linux的实时性，针对其在实时应用中的技术障碍，在参考了与此相关研究基础上，从三方面提出了改善linux实时性能的改进措施。为提高嵌入式应用响应时间精度,提出两种细化linux时钟粒度方法；为增强系统内核对实时任务的响应能力,采用插入抢占点和修改内核法增强linux内核的可抢占性；为保证硬实时任务的时限要求，把原linux的单运行队列改为双运行队列,硬实时任务单独被放在一个队列中,并采用lf调度算法代替原内核的fif调度算法。

　　关键词：linux；实时性；调度策略；抢占

　　1引言

　　目前，无论是在日常生活，还是在工业控制，航空航天，军事等方面，嵌入式系统都有着非常广泛的应用。嵌入式系统目前主要有：indse、vxrks、qnx等，它们都具有较好的实时性、系统可靠性、任务处理随机性等优点,但是它们的价格普遍偏高。而嵌入式linux以其非常低廉的价格，可以大大的降低成本，逐渐成为嵌入式操作系统的首眩但是,作为通用操作系统的linux，由于其在实时应用领域的技术障碍，要应用在嵌入式领域，还必须对linux内核作必要的改进。许多嵌入式设备都要求与外部环境有硬实时的交互能力，将最初按照分时系统目标设计的linux改造成能支持硬实时性的操作系统显得十分重要。幸运的是,linux及其相关项目的开放源码特征为深入研究其内核并加以改造提供了可行性,可以修改linux内核中的各个模块以达到满足嵌入式应用的需求，提高软件方面的开发速度。目前，改善linux内核的设计与实现，使其适用于实时领域吸引了许多研究和开发人员的注意力[1-4]。常用的实时性改造方法是采用双核方法，这种方法的弊端在于实时任务的开发是直接面向提供精确实时服务的小实时核心的，而不是功能强大的常规linux核心。基于此，近年来修改核的方法越来越受到科研人员的重视，这种方法是基于已有linux系统对于软件开发的支持，进行源代码级修改而使linux变成一个真正的实时操作系统。本文分析了标准linux在实时应用中的技术障碍，参考了修改核方法的思想，从内核时钟管理、内核的抢占性、内核调度算法三方面论述了改善标准linux实时性能的方法。

　　2linux在实时应用中的技术障碍2.1linux的实时性分析linux作为一个通用操作系统，主要考虑的是调度的公平性和吞吐量等指标。然而，在实时方面它还不能很好地满足实时系统方面的需要，其本身仅仅提供了一些实时处理的支持，这包括支持大部分psix标准中的实时功能，支持多任务、多线程，具有丰富的通信机制等；同时也提供了符合psix标准的调度策略，包括fif调度策略、时间片轮转调度策略和静态优先级抢占式调度策略。linux区分实时进程和普通进程，并采用不同的调度策略。

　　为了同时支持实时和非实时两种进程，linux的调度策略简单讲就是优先级加上时间片。当系统中有实时进程到来时，系统赋予它最高的优先级。体现在实时性上，linux采用了两种简单的调度策略，即先来先服务调度（shed-fif）和时间片轮转调度（shed-rr）。具体是将所有处于运行状态的任务挂接在一个run-queue队列中,并将任务分成实时和非实时任务，对不同的任务，在其任务控制块task-strut中用一个pliy属性来确定其调度策略。对实时性要求较严的硬实时任务采用shed-fif调度，使之在一次调度后运行完毕。对普通非实时进程，linux采用基于优先级的轮转策略。

　　2.2linux在实时应用中的技术障碍尽管linux本身提供了一些支持实时性的机制，然而，由于linux系统是以高的吞吐量和公平性为追求目标，基本上没有考虑实时应用所要满足的时间约束，它只是提供了一些相对简单的任务调度策略。因此，实时性问题是将linux应用于嵌入式系统开发的一大障碍,无法在硬实时系统中得到应用。linux在实时应用中的技术障碍具体表现在:#p#分页标题#e#

　　（1）linux系统时钟精度太过粗糙，时钟中断周期为10s，使得其时间粒度过大，加大了任务响应延迟。

　　（2）linux的内核是不可抢占的,当一个任务通过系统调用进入内核态运行时,一个具有更高优先级的进程，只有等待处于核心态的系统调用返回后方能执行，这将导致优先级逆转。实时任务执行时间的不确定性，显然不能满足硬实时应用的要求。

　　（3）linux采用对临界区操作时屏蔽中断的方式，在中断处理中是不允许进行任务调度的，从而抑制了系统及时响应外部操作的能力。

　　（4）缺乏有效的实时任务调度机制和调度算法。

　　针对这些问题，利用linux作为底层操作系统，必须增强其内核的实时性能，从而构建出一个具有实时处理能力的嵌入式系统，适应嵌入式领域应用的需要。

　　2.3当前增强linux内核实时性的主流技术近年来，人们对于linux内核实时性改造提出了一些方法和设想，它们采用了不同的思路和技术方案。归纳总结，支持linux的硬实时性一般有两种策略[5]：一种是直接修改linux内核,重新编写一个由优先级驱动的实时调度器（real-tiesheduler），替换原有内核中的进程调度器shed.，kurt是采用这一方案较为成功的实时linux操作系统；另外一种是在linux内核之外,以可加载内核模块（ladablekerneldule）的形式添加实时内核，确保其高响应特性，实时内核接管来自硬件的所有中断，并依据是否是实时任务决定是否直接响应。新墨西哥科技大学的rt-linux，就是基于这种策略而开发的。以上两种策略有其借鉴之处，但如果综合考虑任务响应、内核抢占性、实时调度策略等几个影响操作系统实时性能的重要方面，它们还不能很好的满足实时性问题。为了增强嵌入式linux实时性能，下文将就内核时钟精度、内核的抢占性以及内核调度算法等相关问题重点研究相应的解决方法。

　　3改善嵌入式linux实时性能的方法针对linux在实时应用中的技术障碍，将linux改造成为支持实时任务的嵌入式操作系统,主要从下面三个方面进行着手。

　　3.1细化时钟粒度精确的计时是实时调度器正确操作所必须的，调度器通常要求在一个特定的时刻进行任务切换，计时的错误将导致背离计划的调度，引起任务释放抖动。标准linux系统时钟精度太过粗糙，时钟中断周期为10s，不能满足特定嵌入式应用领域中对于响应时间精度的要求。因此，在实时linux应用中，需要细化其时钟粒度，具体有两种方式可以解决时钟粒度问题：一是通过直接修改内核定时参数hz的初值来细化时钟粒度，如将标准linux中内核定时参数hz改为10000,则时钟粒度可以达到100us，这种方式虽然会增加一些系统开销，但在强周期性环境下，对定时器的设置只需初始化一次，在一定程度上保证了处理效率；二是通过对可编程中断定时器8254或先进的可编程中断控制器进行编程来改进linux时钟机制，以提高其时钟的分辨率，使毫秒级的粗粒度定时器变成微秒级的细粒度定时器。

　　3.2增强linux内核的抢占性标准linux内核是不可抢占的，导致较大的延迟，增强内核的可抢占性能，可提高系统内核对实时任务的响应能力。目前，有两种方法修改linux内核以提高实时任务抢占非实时任务的能力：一是在内核中增加抢占点的方法；二是直接将linux内核改造成可抢占式内核。插入抢占点方法是在linux内核中插入一些抢占点，当一个系统调用执行到抢占点时，如果有更高优先级的实时进程正在等待运行，那么正在执行系统调用的内核进程将会把pu的控制权转交给等待运行的实时进程；如果没有更高优先级的实时进程等待，则当前进程将继续执行，此时系统增加的开销仅仅是检测一下调度标志。将linux内核改造成可抢占式内核方法的基本思想是产生运行调度器的机会，缩短任务发生到调度函数运行的时间间隔。这种方法修改了linux源代码中的自旋锁宏以避免竞争，并在其中引入一个称作抢占锁计数器（pl）的新的计数信号允许内核代码抢占，当它为0时，允许抢占；当其为大于0的任何值时，禁止抢占。目前，针对这两种修改linux内核的方法，已经有两种比较成熟的linux内核补丁被研制出来：抢占式补丁和低时延补叮其中，抢占式补丁是ntavista开发的，它修改了内核代码中的spinlk宏和中断返回代码，使得当前进程可被安全抢占，当自旋锁释放或者中断线程完成时，调度器就有机会执行调度；低时延补丁是由ingalnr提出，该方法只是在执行时间长的代码块上抢占，不采用强制式抢占，因此，如何找到延时长的代码块是解决问题的关键。#p#分页标题#e#

　　3.3改善linux内核实时调度器的调度策略将进入系统的所有任务按实时性分成三类：硬实时、软实时、非实时任务[6]。硬实时要求系统确保任务执行最坏情况下的执行时间，即必须满足实时事件的响应时间的截止期限，否则，将引发致命的错误；软实时是指统计意义上的实时，一般整体吞吐量大或整体响应速度快，但不能保证特定任务在指定时期内完成。针对不同的实时性任务，分别采用不同的调度方法进行处理。

　　为了严格保证硬实时任务的时限要求，改善的linux内核实时调度器采用了优先级调度算法，目前最小松弛时间优先调度算法lf（iniu-laxity-firstshedulingalgrith）是动态优先级调度最常见的实时调度策略。它在系统中为每一个任务设定松弛时间（任务的松弛时间等于任务的截止期减去任务执行时间、当前时间）即:laxity=deadline―urrent\_tie―pu\_tie\_needed，系统优先执行具有最小松弛时间的任务。根据就绪队列的各任务的松弛时间来分配优先级，松弛时间最小的任务具有最高的优先级。为了提高linux的实时性，我们设计了lf调度器，并把它作为可加载模块加入linux内核中，在实现中需对内核进行相应的修改。为了区分任务的类型，对基本linux的task\_strut属性进行改进，在其中增加shed\_lf调度策略，并按task\_strut中的pliy的取值来进行区分,分别用shed\_lf，shed\_rr，shed\_ther来标识硬实时、软实时、非实时任务。将处于运行状态的三类任务放入两个队列，硬实时任务放入hard\_real\_queue队列,采用lf调度算法，软实时和非实时任务放入nn\_real\_queue队列（空闲任务也在其中），沿用原内核的rr调度算法。这两个队列可以用一个run\_queue[2]的指针数组来指向,如图1所示。

　　图1双对列任务运行

　　双队列任务运行过程与原内核的单运行队列执行流程的主要区别在：首先，各硬实时任务采用了lf调度算法代替原内核的fif调度，提高了linux系统的实时性能。其次，在判断是否有软中断需要处理之前需判断硬实时任务队列是否为空，如果不为空，即使存在中断的后半部分需要处理,也要先调度硬实时任务投入运行，在硬实时队列为空的条件下才去处理中断的后半部分(因为中断的后半部分没有硬实时任务紧急)。最后，如果没有硬实时任务存在，则说明只有run\_queue[1]队列中有软实时或非实时任务存在，这时的处理方法和原内核对单运行队列的处理方法相同。这样改进后，可以明显提高硬实时任务的调度效率，而在没有硬实时任务时，系统性能没有变化。

　　4结束语嵌入式技术具有广阔的应用前景，渗透于社会生活的诸多领域，把linux应用于嵌入式系统，把linux自身固有的优越性融入嵌入式技术，是嵌入式技术发展的一个重要方向。然而，由于linux在实时应用领域的技术障碍，要将其应用在嵌入式领域还必须修改linux内核中的各个模块以达到满足嵌入式实时应用的需求。本文在分析linux实时性的同时，探讨了其本身提供的一些支持实时性的机制以及在实时应用中的技术障碍。而后，基于增强linux内核实时性的主流技术，从细化时钟粒度、增强内核抢占性及实时调度策略三方面入手，提出了改善linux实时性能的优化方法。

　　参考文献：[1]赵慧斌,李小群,孙玉芳.改善linux核心可抢占性方法的研究与实现.计算机学报,2004,27(2):244-251

　　[2]蒋溢,李琳皓,陈龙.linux系统实时性探讨.重庆大学学报,2005,28(3):61-64

　　[3]董晓峰,顾新.关于linux内核可抢占性的研究.计算机工程,2005,31(1):82-84

　　[4]陈敏,周兴社.基于嵌入式linux的实时优化方案.计算机应用研究,2005,22(3):235-237

　　[5]齐俊生,崔杜武,黑新洪.嵌入式linux硬实时性的研究与实现.计算机应用,2003,23(6):34-36#p#分页标题#e#

　　[6]李凡,卢杜阶,邱鹏.在嵌入式应用中增强linux实时性的方法研究.华中科技大学学报,2005,33(2):