计算机科学2000Vol.27№.11

基于 LINUX 的硬实时计算

Supporting Hard Real-Time Computing within LINUX

陈 宇 熊光泽 刘锦德 李 允

(电子科技大学计算机科学与工程学院 成都610054)

Abstract In this paper, we present that real-time kernel could be mixed with LINUX to provide real-time function and general-purpose function within a single system. This paper describes the problems of using LINUX to support hard real-time computing in detail, and bring forward the resolution.

Keywords Real-time operating systems, General-purpose operating system, LINUX, CRTOS

一、引言

随着科学技术的发展，计算机的应用已经深入到人类社会的每一个领域，成为人们日常生活中不可缺少的一部分。从军用高技术装备到信息家电，从通信设备到医疗仪器，从工业控制系统到智能仪器、仪表，无一处不依靠计算机。作为计算机的灵魂，操作系统的重要性是不容置疑的。这些应用对所使用的操作系统有着共同的要求——实时性、嵌入性，在满足这一基本要求的同时，这些应用还希望获得文件系统、网络和图形界面等过去仅由通用操作系统才能提供的服务。本文将对目前嵌入式实时操作系统在这方面的发展做一简要叙述，并针对基于 LINUX 的实时计算进行较深入的分析。

二、实时操作系统及其目前发展概述

实时操作系统，由于应用的领域不同，对其概念的理解也有所不同。但通过从各种不同的应用中抽取共性，可将其定义为：具有通用操作系统的基本功能，并保证任务对外界、异步信号及时响应的操作系统。实时操作系统的基本特性是确定性，它的设计要求是在确定的时间界限到来前完成用户任务，另一个重要的特性是任务的吞吐量，即在一定时间段内运行尽可能多的用户任务。

如同计算机的发展，实时操作系统也经过了一个从简单到复杂，从低级到高级的发展过程。早期的实时应用没有实时操作系统作为软件平台，程序员针对确定的目标系统编写应用，代码直接运行在裸机上，系统的资源由应用自己管理，多个任务采用轮转法运行。利用这种方法设计的系统代码的重用率低，设计和调试的工作量大，但效率较高，实时操作系统及其应用开发平台的出现，使得应用程序的开发者可以基于开发平台，利用实时操作系统提供的系统调用，完成具体应用程序的编制和调试，但此时的实时操作系统仅仅提供操作系统所能提供的最基本的功能，如内存管理、任务管理和设备管理等。

随着可以使用的资源日益增多，过去只能在通用操作系统平台上实现的功能，如今嵌入式系统也需要实现。例如，用户要求普通模拟制式电视机加装了机顶盒后，不仅可以接受数字信号节目，还可以访问 INTERNET 以获取更大量的信息。又如，PLC(可编程逻辑控制器)已经在工控领域广泛使用，用户希望它不仅可以控制单台设备，还可以将多台 PLC 通过网络连接起来实现群控。PLC 还应该可以将工作中采集的数据形成文件保存起来。有些 PLC 还需要提供用户通过图形界面在线修改控制程序并调试等的功能。因此，目前的实时操作系统除了保留传统实时操作系统的功能外，还需要根据用户的要求提供文件系统，网络，用户图形界面和数据库系统等扩展功能。

为了在实时操作系统中实现这些扩展功能，业内各机构提出了许多解决方案，大体可以分为三类。第一类以 WindRiver 公司为代表，它通过对实时操作系统 VxWorks 改造，使文件系统、网络等功能在其内核(Wind)内得以实现。其优点在于扩展功能是内核的一部分，内核与用户任务运行在同一空间，系统运行速度快。但是内核较为庞大，即使提供了内核裁剪功能，其

陈 宇 博士生，主要研究方向：实时操作系统。熊光泽 教授、博士生导师，主要研究方向：实时计算机系统、实时软件可靠性评测。刘锦德 教授，博士生导师，主要研究方向：开放式系统、分布式实时系统。李 允 博士生，主要研究方向：实时操作系统。

对空间的要求仍然较高。第二类解决方案以 QNX 最为成功。QNX 采用微内核结构，核内仅有任务调度、中断管理、下层网络功能和进程通信模块。其他功能，如文件系统和设备驱动，都运行于用户空间。QNX 有很好的安全性、裁剪性，但时间性能指标相对 VxWorks 有一定差距。第三类解决方案是对通用操作系统进行改造，使之满足硬实时计算的需要。早在20年前，贝尔实验室的研究人员设计了一种实验操作系统，MERT，其研制目标是使该系统能同时运行实时任务和非实时任务。与以上提出的方案不同，MERT 的设计者使一个实时操作系统内核和一个通用操作系统内核（分时系统）并存，这样既满足了实时任务对时间确定性的要求，又提供了通用操作系统强大的功能。目前，国外很多研究机构以 MERT 的设计思想为基础，正在进行实时 LINUX 的研究。以下将结合笔者所在单位的研究结果，对这一问题做进一步探讨。

三、实时 LINUX

如上所述，目前已经有许多实时操作系统在提供基本的实时功能外，还能够提供用户所需求的扩展功能。但用户还有更多的期望：开放、标准、充分的支持和低廉的价格。LINUX 作为类 UNIX 的通用操作系统，在继承 UNIX 稳定、开放、易于移植和功能强大等优点的同时，源代码对用户是完全公开的。因此，对那些希望针对自己的特殊应用修改系统代码以优化性能的用户，LINUX 最具吸引力。这类用户大多涉及实时计算，特别是硬实时计算。

但是，作为通用操作系统，LINUX 要支持硬实时应用，当前存在着许多需要解决的问题。

1. 任务调度　LINUX 是一个分时系统，它的调度算法的设计目标是提供一种公平的调度机制，平衡系统响应时间和吞吐量，保证系统中运行的所有进程都能获得一段运行时间。因此，LINUX 采用多级反馈轮转调度算法，系统中每个进程都拥有可变的优先级，当进程没有在规定时间片中结束，该进程将放弃 CPU，其优先级降低一级，并被挂到当前所在优先级进程等待队列的尾部。但分时调度通常与实时应用中要求的低延迟和高度的可预测性相矛盾。实时操作系统必须保证目前运行的任务的优先级是可运行任务中最高的，除非出现了更高优先级的任务、该任务运行结束或主动放弃控制权，其他任务无法获得 CPU。因此，传统实时操作系统都采用静态或动态优先级调度，如 RM 或 EDF。

2. 内存管理　为了提高内存的利用率，LINUX 采用了虚拟内存管理技术。进程对内存的访问必须通过地址映射将逻辑地址转换成物理地址。当内存不足以满足新的需求时，操作系统将选择一部分已分配内存，将其中的数据写入交换分区，然后对这段内存进行再分配，这一方法仅对没有时间要求的 LINUX 进程有效。然而，实时任务首先要满足时间的确定性。如果一个实时任务被换出内存，当再次调度它运行时，必须首先经过一个时间不确定的换入过程，这将极大地影响系统的响应时间。

3. 时间粒度　时间粒度是指操作系统所能提供的最小时间间隔。LINUX 的两次系统时钟中断间的时间间隔就是系统的时间粒度。时间粒度越小，系统开销越大。时间粒度越大，进程的响应延迟越大。LINUX 的时间粒度处于百毫秒级，这足以满足 LINUX 进程对于时间粒度的要求，系统开销也较为合理。但对于一个要求在100μs 后被唤醒的实时任务而言，则时间粒度过大。

4. 内核不可抢占　LINUX 的内核是不可抢占的。当进程调用系统调用而陷入核心运行时，即使出现了一个具有更高优先级的进程，该任务也只有等待系统调用返回后方能抢占低优先级的进程。这一算法与硬实时应用中优先级高的就绪任务可以抢占低优先级任务调用的系统调用相矛盾。

5. 可屏蔽中断　为了互斥地访问临界区，LINUX 采用在进行临界区操作时屏蔽相应的可屏蔽中断。这种方法的效率往往高于使用信号量实现互斥。但同时，屏蔽中断也抑制了系统及时响应外部时间的能力。多数实时任务的运行是为了响应外部事件。因此，LINUX 对中断的屏蔽极大地影响系统对于时间确定性的保证。

6. 优化硬件使用　LINUX 会分析进程对硬件资源的请求，以优化访问顺序，以提高使用效率。例如，当多个进程都提出对硬盘的读要求时，系统会对要求排序，使得磁头在运动的最短距离上读出的数据最多。因此，进程的排序有可能出现低优先级进程先于高优先级进程获得数据。这是实时应用领域中所不能允许的。

通过以上分析，通用操作系统实现硬实时计算的主要障碍在于：通用操作系统的设计目标是平均性能最佳，而实时操作系统的目标是最坏性能最佳。

理论上，有针对性的修改 LINUX 的代码是实现基于 LINUX 的硬实时计算最直接的方式。POSIX1003.13（多功能实时系统轮廓）（PSE54）中定义了 mlock 调用以将实时任务锁在内存中，sched_setsched 调用实现实时任务的基于优先级的调度，以及 POSIX 的实时信号用于任务间的通信。这些标准已经在某些 LINUX 版本中实现，但实验证明，这类系统仅适用于软实时应用，因为 POSIX1003.13并没有考虑内核抢占、中断屏蔽和时间粒度等问题。同时，修改代码将使 LINUX 丧失作为通用操作系统在非实时计算所具有的优点。

改造 LINUX 以实现硬实时计算的另一手段就是

采用 MERT 技术，使 LINUX 的一个小的实时内核并存，其中心思想是：整个系统分为三个空间，用户空间，LINUX 内核空间，实时内核空间。运行于实时内核空间的实时内核是系统真正的核心；实时任务也运行在实时内核空间，这有助于提高系统的运行效率；相对于实时任务而言，LINUX 内核是实时内核的最低优先级的任务，运行在 LINUX 内核空间，可以被其它实时任务抢占，避免了实时任务因为 LINUX 内核运行而造成的不必要的等待；非实时任务，也就是 LINUX 的进程运行在用户空间。

为了禁止 LINUX 屏蔽中断，Yodaiken 提出中断抽象层的概念（如图1），当 LINUX 企图屏蔽中断时，实际是对中断抽象层中的相应数据结构进行标记，然后返回 LINUX，使之认为中断屏蔽成功。LINUX 直接接受中断是不允许的。当系统产生中断时，实时内核接受中断请求，并决定采取何种处理方式，若对应于发生的中断，存在一实时中断处理句柄，则该句柄被使用。如果不存在实时句柄，或存在的实时句柄要求 LINUX 句柄的参与，实时内核将在中断抽象层的相应数据结构中做标记。当 LINUX 恢复运行时，LINUX 内核再从数据结构中获取中断信息。这样，实时任务不会因为 LINUX 对中断的屏蔽而导致无法及时响应外部事件。

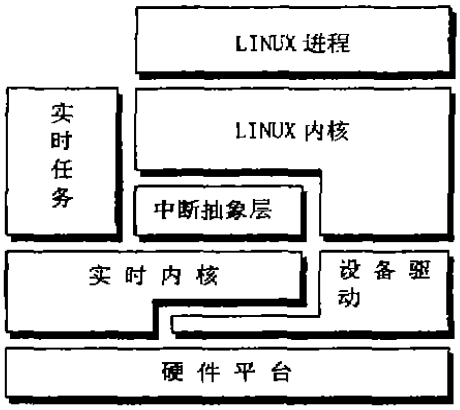

图1　实时 LINUX 的体系结构

实时任务不能直接使用 LINUX 提供的系统调用，而是通过与 LINUX 进程的交互实现对 LINUX 提供的各种服务的访问。实时任务和进程间不能采用 LINUX 提供的传统的 IPC，否则会导致实时任务等待非实时任务的异常情况。为了在保证实时任务运行确定性的前提下实现实时任务和 LINUX 进程间的通信，采用实时管道作为工具，实时管道的空间由实时空间提供。就实时任务而言，对实时管道的读、写是原子操作，因此不会产生阻塞。对于 LINUX 进程而言，实时管道只不过是一个标准的字符设备。

因此，采用 LINUX 内核与实时内核相结合，由实时内核提供时间保证，LINUX 内核提供扩展功能，无须对 LINUX 和实时内核的代码做大的修改，就可以实现基于 LINUX 的硬实时计算。基于实时 LINUX 的硬实时应用程序包括两个部分，实时任务和 LINUX 程序。因此，在进行应用设计时，应遵循以下法则：

精心地划分任务，使尽可能少的代码由实时内核控制运行，尽最大可能发挥 LINUX 的功能。

CRTOS 是电子科技大学微机所嵌入式软件设计中心在"八五"期间研制成功的硬实时操作系统，它由内存管理，任务管理，中断管理，任务间通信等基本模块组成。CRTOS 的优点在于代码量小，仅有20K；运行速度快；稳定性较好。它的缺点是尚未提供有如 LINUX 的丰富的扩展功能，一时难以满足当前市场对实时操作系统的广泛要求。

基于以上分析，采用 MERT 的基本思想，将 CRTOS 作为实时内核，提供基本的实时功能，LINUX 内核作为 CRTOS 的一个任务，提供网络、文件系统和用户图形界面，这时，系统设计的主要工作集中到两个内核的接口上。

由于系统启动时没有实时要求，CRTOS 可以作为可装入模块，在 LINUX 启动之后，再由系统自动加载。

结束语　本文提出的实时 LINUX 实现方案，能够利用 LINUX 强大的功能，丰富的免费资源，结合本国已有的技术积累，较为快速地开发出具有自主版权，功能强大，应用范围广阔的实时操作系统。

应该指出，实时 LINUX 方案还有许多有待完善的地方，比如对实时任务与进程间的通信，仅用实时管道并不能完全满足需要。由于两个内核并存，对于系统资源的要求比较高，应该提供内核裁剪功能。

我们相信，随着研究的深入，这些问题都会逐步得以解决，Real-Time LINUX 技术及其相关产品必将在不远的将来，得到广泛的应用。

参 考 文 献

1　Lycklama H, Bayer D L. The MERT operating system. Bell System Technical Journal, 1978, 57(6): 2049～2086
2　Barabanov M, Yodaiken V. Real-time linux. Linux journal, 1997(Feb.)
3　Radisys Corporation. Intime kernel: [Technical report]. Radisys Corporation. Available at: http://www.radisys.com 1997
4　Ripoll I. Real-Time LINUX(RT-LINUX). Available at: http://www.nt.linuxfocus.org/English/May 1998/article4.html
5　Ripoll I. Earliest deadline first scheduler: [Technical report]. University of Valencia (Spain). Available at: http://bernia.disca.upv.es/, 1998
6　Yodaiken V. The RTLinux Manifesto. Available at: http://www.rtlinux.org
7　Stankovic J A. Real-Time and Embedded System. Available at: http://www.cs.bu.edu/pub/