

嵌入式实时操作系统 wJLinux 的设计与实现

王凌武¹, 蒋文杰²

(1. 浙江财经学院, 浙江 杭州 310012; 2. 浙江大学, 浙江 杭州 310027)

摘要: 为了研发具有自主知识产权的基于 Linux 的嵌入式实时操作系统, 需要分析实时系统的任务管理, 中断处理和存储管理。在改造 Linux 内核基础上, 对任务管理, 中断处理和存储管理进行精心设计, 实现一个全新的实时核心。由此, 使用此新实时内核开发出了一个具有优良实时性能以及自主知识产权的嵌入式实时操作系统——wJLinux。

关键词: 任务管理; 中断处理; 存储管理; 嵌入式实时操作系统; 实时核心; wJLinux

中图分类号: TP316.2 **文献标识码:** A **文章编号:** 1000-7024(2007)01-0135-03

Design and implementation of embedded operation system wJLinux

WANG Ling-wu¹, JIANG Wen-jie²

(1. Zhejiang University of Economics, Hangzhou 310012, China; 2. Zhejiang University, Hangzhou 310027, China)

Abstract: To develop an embedded real time operation system with own property rights, the task management, memory management and interrupt management of real time operation system are analyzed. On the basis of modify the Linux nucleus, three parts of nucleus compose: Task management, interrupt handling and memory management are elaborately designed. Based on this new real time nucleus, a hard real-time embedded operation system with excellence performance and own property rights—wJLinux are developed.

Key words: task management; interrupt management; memory management; embedded real time system; wJLinux; real time nucleus

0 引言

嵌入式实时操作系统的研究与开发是嵌入式领域的一个研究热点。比较有代表性的嵌入式实时操作系统有 RTLinux, vxWorks, QNX, Lynx 等。由于 Linux 的稳定高效, 有着方便的开发环境以及在嵌入式领域的广泛应用^[1], 因此在 Linux 的基础上构造嵌入式实时操作系统是一个不错的选择。wJLinux 就是根据这种理念所开发的嵌入式硬实时操作系统。

1 wJLinux 的总体结构

在 Linux 基础上设计实时操作系统的通常做法是: 在硬件和 Linux 之间插入一个实时子系统, Linux 内核作为一个低优先级的实时任务的和其它实时任务一起运行在实时子系统之上, Linux 用户进程则运行在 Linux 内核之上, 例如 RTLinux 就是采用了这种双内核结构^[2]。wJLinux 也是在 Linux 基础上进行设计, 但是它不要求运行用户进程, 系统中所有的任务都在内核态以线程的形式运行。wJLinux 是一个单一的内核, 在设计和实现中结合 Linux 是为了获取 Linux 众多稳定完善的支持, 如文件系统、网络、底层存储管理等。整个 wJLinux 的总体结构如图 1 所示。从图 1 中可以看出 wJLinux 系统由下往上分为 3 层: 硬件层; 实时核心: 是系统的重要组成部分, 它负责管理实时任务, 操控硬件, 控制中断信号、管理中断服务 (ISR), 负责实时任务、ISR 和非实时任务之间的通信; Linux

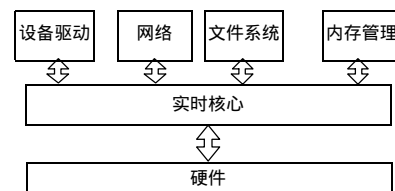


图 1 wJLinux 总体结构

服务层: 可以看作系统运行时的库支持, 主要由 Linux 文件系统、网络、底层存储管理等等组成, 为系统提供各种文件系统、网络支持。wJLinux 的设计和实现重点在于实时核心, 它由任务管理、中断管理、存储管理 3 个主要模块组成。

2 wJLinux 的任务管理设计与实现

2.1 任务管理的设计

wJLinux 不运行用户进程, 系统中所有的任务都以线程的形式运行在单一的内核空间。这样的设计不仅可以使得任务切换时只需切换堆栈和寄存器组, 减小了切换所需的开销, 还可以让系统中所有任务都能直接访问任何有效的地址空间和设备, 避免了访问过程中可能产生的异常, 提高了系统实时性能^[3]。

wJLinux 采用实时系统典型的优先级调度策略, 优先级高的可以抢占优先级低的任务。wJLinux 首先将系统中运行的任务分为 3 个等级, 相同等级的任务又可以按照不同的优先级加以区分: 第一等是 SERVER 级, 在这一等级的主要是一些

收稿日期: 2005-12-15 E-mail: frumer@sina.com

作者简介: 王凌武 (1977 -), 男, 浙江宁海人, 助教, 研究方向为通信与信息系统、工作流系统; 蒋文杰, 男, 硕士研究生。

实现实时中断服务的系统服务线程 wJLinux 将外设中断的实时中断处理以线程的形式运行,第二等是 RTTHREAD 级,在这一等级的是实时任务;最后一等是 DAEMON 级,是非实时任务所处的等级,主要是一些为兼容 Linux 设备、中断管理所需的内核线程(如管理 Linux 软中断的内核线程 keventd 和 ksoftirqd)以及运行在内核态的非实时应用程序(如终端控制程序)。

2.2 任务管理的实现

根据 wJLinux 管理的设计,将系统中的任务分为‘等待(SYSPEND)’和‘可运行(READY)’两种状态,在调度器数据结构中设置 3 个优先级队列:Service_thread_list、RT_thread_list 和 Daemon_thread_list,分别对应优先等级 SERVER、RTTHREAD 和 DAEMON。当一个任务线程 READY 时就根据其等级和优先级挂入某个队列;要 SYSPEND 就从所在的队列中删除。每当系统引发一次调度,内核先搜索 Service_thread_list 队列,取队列中第一个任务即 priority 最高的任务;如果队列为空,则搜索 RT_thread_list 队列;仅当以上两个队列都为空的时候才调度 Daemon_thread_list 队列上的任务。为了保证系统的实时性,系统在时钟中断实时处理程序中调用任务调度函数。

wJLinux 在任务调度的实现上使用优先级队列,摒弃了其它实时操作系统(例如 RTLinux)普遍采用的单链表方式组织任务,每次调度都需要遍历所有任务^[5]的做法,这使得无论系统中有多少任务,内核每次调度最多只需访问 3 个队列的头部,所需的时间是固定的^[6]。因此当系统中任务较多的时候,wJLinux 任务调度所需的时间将比同类的实时操作系统(例如 RTLinux)在相同情况下要少。经过测试,在 MIPS(150 MHz)平台上,当系统中同样有 15 个任务的情况下,一次任务调度 wJLinux 所需时间 $\approx 10 \mu s$,而 RTLinux 所需时间 $\approx 14 \mu s$ 。

3 wJLinux 的中断处理设计与实现

wJLinux 将外设中断的处理分两个部分:先进行的实时中断处理和延后的 Linux 中断处理并在设计和实现中引入了线程和消息机制。在 wJLinux 中,除了时钟中断,每个外设中断都可以最多申请一个优先等级处于 SERVER 级的系统服务线程来执行相应的实时中断处理(为区分不同优先级的中断可以通过设置系统服务线程的 priority 来实现)。系统服务线程可以在系统初始化或者运行时由实时任务动态地加载、撤销(例外的时钟中断服务程序由内核直接定义无法更改,它的主要工作是引发系统任务调度)。此外通过对 Linux 中断流程稍作更改,Linux 中断服务由优先级较低的实时线程 Linux_IRQ_Thread 调用执行。整个 wJLinux 的中断处理结构如图 2 所示。

wJLinux 的中断流程是:当实时中断服务程序被加载后,

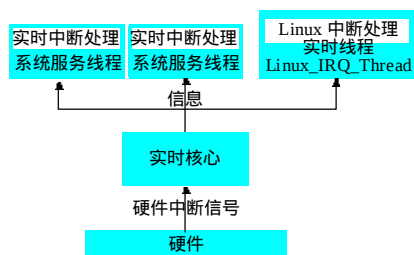


图 2 wJLinux 的中断处理

系统创建一个系统服务线程来执行它,在做完初始化工作后进入消息等待。当某个硬件中断信号到来后,系统进入实时核心中统一的中断入口函数。统一的中断入口函数做 4 步简单的工作,这些工作都在禁止中断的环境下执行:屏蔽当前的中断号;如果当前的中断号有相应的实时中断处理,则向对应的系统服务线程发送消息并同时唤醒线程;唤醒并发送消息给实时线程 Linux_IRQ_Thread,由它来完成 Linux 中断服务的调用;进行系统调度,最后系统中断返回。中断所对应的系统服务线程由于接收到消息而被唤醒,然后执行具体的中断处理操作,完成后取消对当前中断号的屏蔽并再次进入消息等待,直到下次中断被再次唤醒。

实时中断处理的系统服务线程中的大部分操作都可以在允许中断的情况下执行^[7],它的一般格式是:

```

unsigned int rt_irqxx_handler(void)
{ 初始化
  for(;;)
  { 消息等待
    具体的中断处理
    取消当前中断号的屏蔽
  }
}

```

在 wJLinux 中断处理中引入线程和消息机制是 wJLinux 创新点,是区别其它基于 Linux 的实时操作系统最大的不同之处。

4 wJLinux 的存储管理设计与实现

为了加快响应实时任务申请、释放内存的请求,确保系统的实时性,wJLinux 在 Linux 存储管理的基础上添加了存储快缓存机制^[8]。wJLinux 的存储快缓存机制是:系统在初始化时建立一组内存池(mem_region),每一个内存池都是一段连续的内存空间并被划分为一组指定大小的内存块(mem_block),每一个 mem_region 所维护的 mem_block 大小相同,系统中所有 mem_region 中的 mem_block 大小则从 256 B 到 1 MB 不等。当实时任务需要分配内存的时候,首先根据所需内存的大小从相应 mem_block 大小的 mem_region 中获取一个 mem_block,如果 mem_region 为空则先通过 Linux 存储管理申请一段连续空间并新建一个 mem_region 加入存储块缓存,然后再从新的 mem_region 中获取 mem_block。当实时任务释放内存时根据释放存储块的地址找到相应的 mem_region 释放一个 mem_block。

5 结束语

RTLinux 是一个在 Linux 基础上进行设计、实现的嵌入式实时操作系统并且在 x86、ARM、MIPS、PowerPC 体系结构的嵌入式平台上成功移植。wJLinux 是一个实时性能优良的嵌入式硬实时系统,在任务切换时间,中断响应时间,最大中断禁止时间等实时性能参数上不输于同类的实时操作系统。如何进一步增强系统的可靠性及可维护性以及如何投入实际应用将是下一阶段的研究重点。

参考文献:

- [1] 廖根为,王力生,陈望斌.嵌入式 Linux 操作系统的实时性能分析与改进[J].计算机应用研究,2004,21(3):199-201.

- [2] 毛德操, 胡希明. 嵌入式系统 [M]. 杭州: 浙江大学出版社, 2003.54-57.
- [3] 慕春棣. 嵌入式系统的构建 [M]. 北京: 清华大学出版社, 2004.23-54.
- [4] 胡泽明. 嵌入式系统开发要素的选择分析[EB/OL]. 2003.http://www.hqew.com.
- [5] 胡泽明. 嵌入式系统开发要素的选择分析[EB/OL]. 2003.http://www.hqew.com.

//www.hqew.com.

- [6] Michael Barr. Programming embedded systems in C and C++ [M].北京:中国电力出版社, 2001.110-189.
- [7] 邹思铁. 嵌入式 Linux 设计与应用[M].北京:清华大学出版社, 2002.5-305.
- [8] Brian W Kernighan, Dennis M Ritchie. C programming language [M]. USA: Prentice Hall, 2001. 10-120.

(上接第 5 页)

变换的应用领域,比如信号分析,图像处理等,正在得到越来越多的实际应用。因此,实现 Harr 变换的量子计算过程,对于量子小波变换理论和实际应用的发展有着非常重要的意义。根据 Harr 小波函数的定义,以及 Harr 函数矩阵表示的研究^[8],我们得到了在 2 阶、4 阶和 8 阶情况下基本 Harr 变换矩阵 H_2 、 H_4 和 H_8 的表示(不考虑变换系数)。经过分析,进一步推广到 N ($N=2^n$)阶情况下 H_N 的矩阵表示形式。

$$H_2 = \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} = \sqrt{2}W$$

$$H_4 = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & -1 \end{pmatrix} H_8 = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & 1 & -1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & -1 & -1 \\ 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 \end{pmatrix}$$

分析 Harr 矩阵随着阶数以 2 幂级数数量级的提高,矩阵元素所发生的变换,我们得到了 2^n 阶 Harr 矩阵 H_{2^n} 是通过 $H_{2^{n-1}}$ 与 $I_{2^{n-1}}$ 构成一个矩阵元组,与 2 阶 W-H 矩阵 W 进行扩展 Kronecker 乘积后再由 2^n 阶正移置换矩阵作用而得到的。 H_{2^n} 表示为

$$H_{2^n} = \Pi_{2^n}((H_{2^{n-1}}, I_{2^{n-1}}) \otimes W) \quad (\otimes \text{为扩展 Kronecker 积}) \quad (10)$$

由式(9)可知,式(10)可以进一步分解为

$$H_{2^n} = \Pi_{2^n}((H_{2^{n-1}}, I_{2^{n-1}}) \otimes I_2) \times (I_{2^{n-1}} \otimes W)$$

进一步分解可得

$$H_{2^n} = (I_{2^{n-1}} \otimes W) \cdots (I_{2^{n-2}} \otimes W \otimes I_{2^{n-2-1}}) \cdots (W \otimes I_{2^{n-2}}) \times (\Pi_4 \otimes I_{2^{n-4}}) \cdots (\Pi_4 \otimes I_{2^{n-6}}) \cdots (\Pi_4 \otimes I_{2^{n-8}}) \Pi_{2^n} \quad (11)$$

$$\text{令 } H_{2^n}^{(1)} = (I_{2^{n-1}} \otimes W) \cdots (I_{2^{n-2}} \otimes W \otimes I_{2^{n-2-1}}) \cdots (W \otimes I_{2^{n-2}})$$

$$H_{2^n}^{(2)} = (\Pi_4 \otimes I_{2^{n-4}}) \cdots (\Pi_4 \otimes I_{2^{n-6}}) \cdots (\Pi_4 \otimes I_{2^{n-8}}) \Pi_{2^n}$$

$$\text{则 } H_{2^n} = H_{2^n}^{(1)} \times H_{2^n}^{(2)} \quad (12)$$

由式(11)可得出实现 H_{2^n} 的量子逻辑线路如图 4 所示。

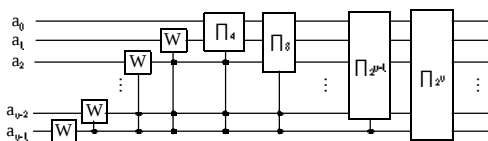


图 4 Π_{2^n} 的量子逻辑线路

从式(12)可以看出 H_{2^n} 的实现复杂度取决于 $H_{2^n}^{(1)}$ 和 $H_{2^n}^{(2)}$ 的实现复杂度。很清楚, $H_{2^n}^{(1)}$ 的实现复杂度为 $O(n)$, 所需的 W-H 变换的数量为 $O(n)$ 。从上文正移置换矩阵 Π_{2^n} 的实现复杂度分析来看, $(\Pi_4 \otimes I_{2^{n-4}})$ 的实现复杂度为 $O(4)$, 即可以通过 $O(4)$ 数量的 Π_4 变换来实现。则可以推得 $H_{2^n}^{(2)}$ 的实现复杂度为 $O(n^2)$, 即所需 Π_4 门

的数量为 $O(n^2)$ 。由此可得 H_{2^n} 的实现复杂度同样为 $O(n^2)$ 。

4 结束语

从以上 Harr 函数量子计算实现的分析可以看出,在实现复杂度上, H_{2^n} 函数的实现复杂度属于多项式关系范围,即是可解的。而它的物理实现完全取决于 W-H 变换 W 和 2 量子位(4 阶)正移置换矩阵的实现。随着量子计算物理实现研究的进展,各种实现量子计算的方法正在不断被开发和完善。目前核磁共振(NMR)是最有希望实现量子计算的物理体系之一^[5],在 NMR 系统中, W-H 变换和 CNOT 门都已经被很成功地实现了。因此, H_{2^n} 小波变换的物理实现应该只是 NMR 脉冲序列的设计问题,理论上已不存在任何无法解决的问题。我们对 Harr 小波函数的研究和量子逻辑线路设计,不仅对于 Harr 函数的物理实现,同时对其它小波函数的研究和整个量子小波变换的实现都应该具有一定的参考价值。

参考文献:

- [1] Ioannis G Karafyllidis. Visualization of the quantum fourier transform using a quantum computer simulator[J]. Quantum Information Processing, 2003, 2(4): 271-288.
- [2] Yang X D, Wei D X, Luo Jun, et al. Preparation of pseudo pure state in nuclear spin ensemble using CNOT gates combination [J]. Chinese Science Bulletin, 2002, 47(22): 1856-1860.
- [3] Arasu K T, de Launey W. Two-dimensional perfect quaternary arrays[J]. IEEE Trans Inform Theory, 2001, 47(4): 1482-1493.
- [4] Wei Daxiu, Luo Jun, Sun Xianping, et al. NMR experimental realization of seven-qubit D-J algorithm and controlled phase-shift gates with improved precision [J]. Chinese Science Bulletin, 2003, 48(3): 239-143.
- [5] Wei Daxiu, Yang Xiaodong, Luo Jun, et al. NMR experimental implementation of three parties quantum super dense coding[J]. Chinese Science Bulletin, 2003, 49(5): 423-426.
- [6] Jozsa R. Quantum factoring, discrete logarithms and the hidden subgroup problem[J]. Computing in Science and Engineering, 2001, 3(2): 34-43.
- [7] Zeyad Al Zhou, Adem Kilicman. Matrix equalities and inequalities involving Khatri-Rao and tracy-singh sums[J]. Journal of Inequalities in Pure and Applied Mathematics, 2006, 7(1): 496-513.
- [8] Fatih Bulut, Polyzou W N. Wavelet methods in the relativistic three-body problem[J]. Phys Rev C, 2006, 73:34-48.