嵌入式操作系统进程调度研究

吴景锋
(华东师范大学 软件学院,上海 200062)

摘要:进程调度是嵌入式操作系统的关键问题,决定了操作系统的优劣。为了深入了解嵌入式操作系统的进程调度技术,针对现有嵌入式操作系统的发展现状,选取较流行的 μC/OS-Ⅱ和 Linux 为研究对象,对它们的进程调度机制和策略进行了研究,指出它们的特点。在相互对比的基础上,提出了相应的改进方案。

关键词:嵌入式操作系统;实时操作系统;μC/OS-Ⅱ;Linux;进程调度
中图分类号:TP316　　文献标识码:A　　文章编号:1009-3044(2006)17-0107-02

Research on Process Scheduling in Embedded Operating System

WU Jing-feng
(Software Engineering Institute,East China Normal University,Shanghai 200062,China)

Abstract:Process scheduling is the key problem in embedded operating system, which determines the operating system is good or not. In order to go deep into the techniques of process scheduling in embedded system, under the current developing situation of embedded system, chooses μC/OS-Ⅱ and Linux as examples, researches their scheduling mechanism and scheduling policy, points out their characteristics. By the contrast of each other, brings forward an improved scheme accordingly.

Key words:embedded operating system;real-time operating system;μC/OS-Ⅱ;Linux;Process scheduling

1 引言

嵌入式操作系统是一种支持嵌入式系统应用的系统软件,它是嵌入式系统极为重要的组成部分。与通用操作系统相比较,嵌入式操作系统在系统实时高效性、硬件的相关依赖性、软件固态化以及应用的专用性等方面具有较为突出的特点[1]。在多任务嵌入式系统中,对重要性各不相同的任务进行合理调度是保证每个任务能够及时执行的关键。

目前常用的调度算法分为两大类:优先级法和时间片法。优先级法是根据进程的优先级来决定选择次序,它又分为静态优先级法和动态优先级法两类。时间片法对各个就绪进程的选择不按优先级次序排列,而是轮流把 CPU 分配给它们。

进程的合理调度是一个复杂的工作,它取决于可执行程序的类型和调度策略的目的。

随着嵌入式技术的发展,对嵌入式系统的要求越来越高,多样化的调度方法已成为一种趋势。在发展较成熟的实时系统中有多种任务调度方法可以借鉴,如截止期最早优先算法、速率单调算法以及可达截止期最早优先算法等。选择调度算法与特定的应用有关。

2 嵌入式操作系统进程调度分析

在多进程的操作系统中,进程调度是一个全局性的、关键性的问题,它对系统的总体设计、系统的实现、功能设置以及各方面的性能都有决定性的影响。在设计操作系统的进程调度模块时需要考虑的主要有进程调度的机制和调度的策略和依据两个方面。进程调度的机制,包括调度的时机和进程调度的方式,需要解决的问题包括系统什么情况下需要调度,以怎样的方式进行调度,按照方式的不同可以分为可抢占式调度和不可抢占式调度。调度的策略和依据,也就是我们常说的调度算法。

2.1 μC/OS-Ⅱ任务调度分析

μC/OS-Ⅱ是一个源代码公开的实时的多任务操作系统,它提供了实时系统所需的基本功能,没有提供输入输出管理、文件系统、网络之类的额外服务。但是由于 μC/OS-Ⅱ的可移植性和开源性,用户可以自己添加所需的各种服务。μC/OS-Ⅱ允许建立多达 63 个用户任务,每个任务都有自己单独的栈。

2.1.1 μC/OS-Ⅱ任务控制块数据结构

一旦任务建立,一个任务控制块 OS_TCB 就被赋值。任务控制块是一个数据结构,当任务的 CPU 使用权被剥夺时,μC/OS-Ⅱ用它来保存该任务的状态。当任务重新得到 CPU 使用权时,任务控制块能确保任务从当时被中断的那一点丝毫不差地继续执行。

2.1.2 μC/OS-Ⅱ任务调度机制

μC/OS-Ⅱ总是通过 OSSched()函数和 OSIntExit()函数进行任务调度,前者用于正常的任务状态变化时发生的调度,而后者用于中断返回时的重新调度。

μC/OS-Ⅱ内核是一个抢先式内核,可以进行任务间切换,也可以让一个任务在得不到某个资源时休眠一定时间后再继续运行。

2.1.3 μC/OS-Ⅱ任务调度策略和依据

μC/OS-Ⅱ系统采用静态优先级分配策略,由用户来为每个任务指定优先级,总是运行就绪条件下的优先级最高的任务。

μC/OS-Ⅱ中每个任务任何时候都处于以下五种状态之一:休眠态、就绪态、运行态、挂起态和中断态。所有的任务控制块都是放在任务控制块列表数组 OSTCBTbl[]中。

μC/OS-Ⅱ分配给系统任务若干个任务控制块,供其内部使用。

每个任务的就绪标志都放在就绪表(ready_list)中,就绪表中有两个变量 OSRdGrp 和 OSRdyTbl[]。在 OSRdyGrp 中,任务按优先级分组,8 个任务为一组。OSRdyGrp 中的每一位表示 8 组任务中每一组中是否有进入就绪状态的任务[2]。任务进入就绪态,就绪表 OSRdyTbl [] 中对应元素的相应位置 1。OSRdyGrp 和 OSRdyTbl[]的关系如图 1 所示。

如果一个任务被删除了,就绪任务表数组 OSRdyTbl[]中对应元素的相应位清零。而对于 OSRdyGrp,只有当被删除任务所在的任务组中全部任务都未进入就绪态时,才将相应位清零。为了查找进入就绪态的优先级最高的任务,需要查优先级判定表 OSMapTbl[]。OSMapTbl[]中每个字节的 8 位代表这一组的 8 个任务中哪些进入了就绪态,低位的优先级高于高位。利用这个字节为下标来查询 OSMapTbl[]表,返回的字节就是该组任务中就绪任

收稿日期:2006-02-18
作者简介:吴景锋(1977-),男,江西省赣州市人,硕士研究生,研究方向为嵌入式系统。

务中优先级最高的那个任务所在的位置。此返回值在 0-7 之间。在 μC/OS-Ⅱ中，确定该哪个任务运行的工作是由调度器(scheduler)完成的。

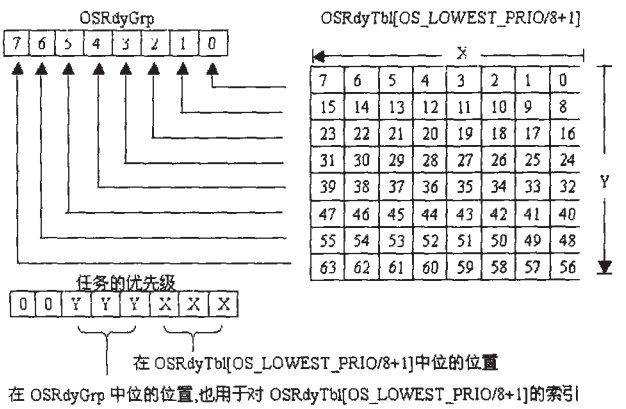

图 1 OSRdyGrp 和 OSRdyTbl[]的关系

2.2 Linux 进程调度分析

Linux 是一个多任务、分时、类 Unix 操作系统。它稳定、源码开放，吸引了很多组织和公司对它进行嵌入式实时改造。

Linux 采用用户空间/内核空间的系统架构，用户的程序以进程的方式运行于用户空间，每个进程拥有相互独立的地址空间。控制和支撑用户程序运行的操作系统各管理模块如内存管理、进程管理、进程通信和文件系统等均位于内核空间，内核空间则是统一的地址空间以提高管理模块间的切换速度。

2.2.1 Linux 进程控制块数据结构

进程控制块(PCB)，既 Linux 内核中的 task_struct 数据结构是 Linux 操作系统调度过程中必须使用的最重要的数据结构，它在进程调度过程中向操作系统调度器提供方方面面的必要信息。尽管 task_struct 数据结构很大很复杂，但它的字段可以被分成几个功能区：

1 状态 随着进程执行，它根据其环境改变其状态。

2 调度 信息调度器需要这些信息以公平地决定系统中哪个进程最值得运行。

2.2.2 Linux 进程的调度机制

自愿的调度随时都可以进行。在内核里面，一个进程可以通过 schedule()启动一次调度，当然也可以在调用 schedule()之前，将本进程的状态设置为 TASK_INTERRUPTIBLE 或 TAS K_UNINTERRUPTIBLE，暂时放弃运行而进入睡眠。在用户空间中，则可以通过系统调用 pause()来达到同样的目的[3]。

除此之外，调度还可以非自愿地，即强制性地发生在每次从系统调用返回的前夕，以及每次从中断或异常处理返回到用户空间的前夕。但这只是发生调度的必要条件，而不是充分条件。是否发生调度，还要看以下情况的发生：当前进程通过系统调用自愿让出运行以及在系统调用中因某种原因受阻，因某种原因而唤醒了另外一个进程，以及时钟中断程序发现当前进程已经连续运行太久。以上情况都可能导致调度的发生。

Linux 操作系统中的进程调度是基于时间片固定优先级有条件的可抢占式调度，表示 Linux 的调度方式是可抢占的，当前进程的运行是可以被可能存在的高优先级的进程所抢断执行，但这种抢占调度的发生是有条件的，这从 Linux 选择的调度器运行时机上可以看到。因此内核空间运行的不可抢占是 Linux 抢占调度方式的限制条件，这对于实时性能的影响是很大的，所以我们称 Linux 操作系统的调度方式是有条件的可抢占式调度。

2.2.3 Linux 进程的调度策略和依据

由于 Linux 作为分时操作系统，系统目标是较好的平均系统响应时间和较高的吞吐量，并不具备实时系统的特征，还不能够直接满足硬实时系统方面的需要。

在内核 1.3 版本之后，POSIX 实时扩展部分被添加近来，Linux 开始逐渐兼容 POSIX 1003.lb 实时规范。引入了实时进程的概念，允许将一个进程的属性确定为实时进程。Linux 从调度策略上区分实时进程和普通进程，实时策略先于普通进程运行，采用不同的调度策略。对实时进程，Linux 采用两种调度策略，即先来先服务调度 SCHED_ FIFO 和时间片轮转调度 SCHED _RR。SCHED_FIFO 是运行直至阻塞的策略。SCHED_FIFO 任务按优先级调度，一旦开始就一直运行到结束或者阻塞在某种资源上，它们不像 SCHED_RR 基于 linux 的嵌入式实时操作系统研究与应用任务那样共享处理器。SCHED _RR 这种轮转的调度策略中为每个任务分配一个时间片，一旦时间片用完，就被移动到优先级队列的队尾，并允许同一优先级的其他任务运行。如果同一优先级没有其他任务，该任务将继续运行下一个时间片。此外，Linux 还提供内存锁定 (Mlock)，POSIX 实时信号等机制以在一定的范围内支持粗粒度的软实时应用，但许多重要的实时性能指标，如任务切换时间、中断响应时间等不能够明确，与支持硬实时性还有明显差距，还有较多的问题需要解决。

3 嵌入式操作系统改进

体现嵌入式操作系统的实时性，任务调度算法的选择就显得非常的重要，μC/OS-Ⅱ系统采用静态优先级分配策略，由用户来为每个任务指定优先级。μC/OS-Ⅱ的任务调度是按抢占式多任务系统设计的。为了简化系统的设计，μC/OS-Ⅱ规定所有任务的优先级必须不同，任务的优先级同时也唯一地标识了该任务。即使两个任务的重要性是相同的，它们也必须有优先级上的差异。

在很多实时系统中，给每个任务分配不同的优先级，执行频率越高的任务分配给它的优先级越大，并且各个进程严格按照优先级的顺序运行。而非实时系统中，例如 Linux 的非实时内核，普通的服务进程对响应时间要求不高，并且它们之间没有优先权的高低之分，采用的是时间片轮转调度算法。对每个进程分配一定的时间片，并以此作为任务调度的依据，各个进程轮流得到调度。如果一个实时的嵌入式操作系统既要处理某些实时的任务，又要提供一定的非实时服务，对于实时任务，需要按照优先级不同顺序完成；对于非实时的服务进程，则希望它们轮流执行。

3.1 μC/OS-Ⅱ改进

为了让 μC/OS-Ⅱ能够同时处理具有实时进程和普通进程的情况，可以取出 μC/OS-Ⅱ中的一组(8 个)优先级供非实时进程使用，也可以根据情况选取二组或更多组供非实时进程使用，设立时间片，在 TCB 控制块中增加一个 counte 作为权值。调度时，首先判断优先级最高的进程是否为非实时进程，若不是，则直接将其运行，若是，则顺序遍历所有就绪非实时进程，计算其时间片 counte 的值，取出时间片最大的进程运行。若遇到时间片大小相同的进程，则取出优先级大的进程运行。如果在遍历中发现所有非实时任务的时间片都已经用完，则仿照 Linux 的方法，再将它们赋予一定的初值，并取出优先级最大的进程投入 CPU 运行。

4 结束语

实时嵌入式操作系统的调度策略有很多种，每一种都有其各自的优缺点，应该根据不同的情况选取不同的调度策略，也可以通过混合调度法，将不同的调度策略结合起来使用，达到更好的调度性能。

参考文献：

[1]王田苗.嵌入式系统设计与实例开发[M].北京:清华大学出版社,2003.

[2]Jean.J.Labrosse,邵贝贝等译.嵌入式实时操作系统 μC/OS-Ⅱ(第 2 版)[M].北京:北京航空航天大学出版社,2003.

[3]李善平,刘文峰.Linux 与嵌入式系统[M].北京:清华大学出版社,2003.