嵌入式实时操作系统任务调度算法优化

谢　敏[1,2]，李桥梁[1]

(1. 南京航空航天大学 机电学院285信箱，江苏 南京 210016；
2. 南京化工职业技术学院 自动控制系，江苏 南京 210048)

摘　要　在嵌入式系统中，任务调度器的好坏很大程度上决定了系统的性能。该文分析了实时系统中有代表性的静态以及动态调度算法，在此基础上，结合RMS 和EDF 算法各自的优点，对嵌入式实时内核μc/os-II的调度算法进行了优化。

关键词　嵌入式实时操作系统；调度；优先级
中图分类号　TP316

嵌入式实时操作系统兼有嵌入式和实时性的特点。作为一种嵌入式操作系统，它具有嵌入式软件共有的可裁剪、低资源、低功耗等特点；作为实时操作系统除了要满足应用的功能需求以外，更重要的是还要满足应用提出的实时性要求。实时操作系统所遵循的最重要的设计原则是：采用各种算法和策略始终保证系统行为的可预测性。实时操作系统的首要任务是调动一切可利用的资源完成实时控制任务。如何使任务集内各任务满足各自的时限，使系统得以正常、高效率工作的任务调度算法一直是实时系统领域内研究的焦点。根据其应用领域及追求精简、高效角度的不同，任务调度算法从简单的合理安排任务循环，发展到基于优先级的速率单调调度(RMS)、截止时间单调调度 (DMS)、最早时限优先(EDF)、最短空闲时间优先(LLF)等算法。任务调度算法的好坏以及执行效率直接关系到嵌入式内核的应用范围及实时性程度。

1　静态调度

静态调度是在系统开始运行前进行调度的，严格的静态调度在系统运行时无法对任务进行重新调度。 静态调度的目标是把任务分配到各个处理机，并对每一处理机给出所要运行任务的静态运行顺序。 静态调度算法实现简单，调度的额外开销小，在系统超载时可预测性好。 但也具有很大的局限性，例如资源利用率低、受系统支持的优先级个数限制以及灵活性和自适应性差等。下面介绍两种常见的静态调度算法。

1.1 速率单调调度

RMS算法于1973年由C.L.Liu和J.Layland提出的，RMS算法的核心思想是根据任务的周期来设置任务的优先级，周期越小，其优先级越高，并以单调的顺序对剩余的任务分配优先级。

RMS做了一系列假设：

（1）所有任务都是周期性的；

（2）任务间不需要同步没有共享资源没有任务间数据交换等问题；

（3）CPU必须总是执行优先级最高且处于就绪态的任务，即须用可剥夺型调度法。

由于采用抢占式的调度方式，高优先级的任务就绪后立即抢占正在运行的任务。Liu 和Layland的理论证明了下列公式，即用RMS 调度的独立的周期性任务总能满足其截止时间(deadline)的要求。

$$\frac{C_1}{T_1}+\frac{C_2}{T_2}+\mathrm{K}+\frac{C_n}{T_n}\pounds\ n(2^{1/n}-1)$$

式中 C_i为任务i的最长执行时间；T_i 为任务i的周期，n是系统中的任务数。已经证明：对于在单处理器上调度独立、可抢占的周期任务，RMS是最佳的静态优先级算法。RMS 算法的优点是开销小，灵活性好，可调度性测试简单。但在某些情况下，最高执行率的任务并非是最重要的任务。

1.2 截止时间单调调度

DMS是在速率单调调度的基础上发展起来的，不同的是任务的优先级按截止时间来分配。 截止

收稿日期：2005-09-16

时间短的任务优先级高，截止时间长的任务优先级低。截止时间调度具有与速率单调算法相同的优点， 但DMS 算法放松了对任务的周期必须等于其截止时间的限制。 在任务的截止时间小于或等于其周期的情况下，DMS 已被证明是静态最优的调度算法。

2 动态调度

在嵌入式实时系统中，动态调度依赖于任务的优先级。 优先级可以静态分配或者依据不同的特征参数，如截止时间、空闲时间或关键性（即任务的重要程度或者价值）等进行动态分配。动态调度可以是抢占式的或非抢占式的。 当检查到一事件时，动态抢占式算法立即决定是运行与此事件相关的任务，或继续执行当前的任务；对于动态非抢占式算法，它仅仅知道有另一个任务可以运行，在当前任务结束后，它才在就绪的任务中选择一个来运行。以下介绍的是两个经典的动态调度算法：最早截止时间优先算法和最小松弛时间算法。

2.1 最早截止时间优先算法

抢占式EDF调度算法是一个动态优先级驱动的调度算法，其中分配给每个任务的优先级根据它们当前对最终期限的要求而定。当前请求的最终期限最近的任务具有最高的优先级，而请求最终期限最远的任务被分配最低优先级。这个算法能够保证在出现某个任务的最终期限不能满足之前，不存在处理器的空闲时间。抢占式EDF调度算法最大的优势在于，对于任何给定的任务集，只要处理器的利用率不超过100%，能够保证它的可调度性。EDF算法在系统的负载相对较低时非常有效，EDF的缺点在于不能解决过载问题。在系统负载较重时，系统性能急剧下降，引起大量任务错过截止期，甚至可能导致CPU 时间大量花费在调度上,在这时系统的性能还不如FIFO(First In First Out)方法。

2.2 最小松弛时间算法

LSF算法优先级的分配基于每个任务的有效松弛时间，一个任务的松弛时间被定义为在其错过截止期之前能够承受的最大时间延迟，这个算法给具有最小松弛时间的任务分配最高优先级，以此来保证紧急任务的优先执行。但是这个算法会导致调度开销没有上界，并且当两个或多个任务具有相似的松弛时间时调度情况会变坏。由于等待任务的松弛时间是严格递减的，其等待执行的缓急程度也随时间越来越紧迫，因此在系统执行过程中，等待任务随时可能会抢占当前执行的任务。LSF 算法造成任务之间的频繁切换现象较为严重，增大了系统开销并限制了LSF 算法的应用。

3 调度算法的选择

嵌入式实时系统中资源是非常有限的，所以开销要尽可能小。开销主要包括运行开销和调度开销。 运行开销与队列分析和从调度队列中增加、删除任务相关。每个任务在一个调度周期内至少被阻塞和唤醒一次，所以任务调度器在一个周期内不得不对一个任务进行两次选择。RMS 算法根据任务的执行频率设置优先级，有较小的运行开销，但执行频率最高的任务不一定是最重要的。EDF 算法中则是对整个任务列表的调度开销进行全面比较，选择最高优先级任务进行调度，有较小的调度开销，但对多个任务具有同一优先级的情况考虑不足。

通过上述分析，如果能够混合RMS和EDF 的优点，采用RMS 较小的运行开销和EDF较小的调度开销，得到一个优先级分配的优化算法，在多个任务要求调度时选择截止时间最短(即优先级最高)的任务运行，若任务的优先级相同时则选择运行频率最高的任务运行。

结合源码公开的嵌入式实时操作系统μc/os-II为例，μc/os-II可以管理64个任务，保留了其中8个系统，所以应用程序最多可以有56 个任务，采用抢占式的优先级调度策略，绝大多数服务的执行时间具有确定性，要求赋予每个任务的优先级必须是不相同的，这样的限制简化了任务管理系统，但也给大型应用带来了限制。如果采用上述改进的优先级分配的优化算法，则突破了原系统对任务数量的限制，并去除对每个任务必须有不同优先级的要求。算法改进的关键之处是对优先级相同的任务建立散列表，并在散列表中按照任务执行频率进行降序排列，表中的第一个任务就是在该优先级上执行频率最高的任务，这要求在一个优先级上可以建立

多个任务。在对某优先级任务执行调度的时候，首先看该优先级上是否有其他任务，如果有，则找到执行频率最高的进行调度。改进后的任务创建和任务调度的伪码如下。

```
建立任务:
{
    建立任务控制块;
    建立任务堆栈OSTaskStkInit () ;
    if (在该优先级上已有任务存在)
    {
     比较任务的执行频率;
     将该TCB按降序插入到散列表中 ;
    }
    else
    {
    新任务是优先级散列表上的第一个任务;
    }
}
任务调度:
if (在该优先级上有任务存在)
{
 if (任务数大于1)
   {
     在该优先级散列表中找到执行频率最高的任务进行调度;
   }
  else
   {执行任务调度;}
 }
else
 {返回错误，任务不存在;}
```

4 结束语

通过对实时系统中的经典调度算法进行分析、比较，在周期性任务组成的实时系统中，以嵌入式实时内核μc/os-II为例，结合RMS 算法具有较小运行开销和EDF 算法具有较小调度开销的特点，对μc/os-II的任务调度策略进行了优化，是对嵌入式系统调度策略的一次有益探索。

参考文献

1 吴明晖. 基于ARM 的嵌入式系统开发与应用. 北京: 人民邮电出版社,2004.

2 Labrosse J J. 嵌入式实时操作系统μc/os-II. 第2版. 北京: 北京航空航天大学出版社, 2003.

作者简介

谢敏（1975—），男，硕士研究生，讲师。研究方向：嵌入式系统。

李桥梁，男，副教授。研究方向：嵌入式系统等研究。

Optimization of RTOS′ Task-Scheduling Algorithms

Xie Min[1.2], Li Qiaoliang[2]

(1.College of Mechanical and Electrical, NanJing University of Aeronautics and Astronautics, Nanjing 210016, China;
2.The Department of Automatic Control, NanJing College of Chemical Technology, Nanjing 210048, China)

Abstract The performance of an embedded system is determined, to a great degree, by the underlying task scheduler of RTOS (Real-time Operating System). This paper first analyzes the static scheduling algorithm and the dynamic scheduling algorithm, both of which are representative in RTOS. Then it presents a new algorithm for optimizing the existing scheduling algorithm of the real-time kernel of μc/os-II by combining the advantages of both RMS and EDF.

Keywords RTOS; priority; task scheduling