嵌入式操作系统调度机制的研究*

何福贵，侯义斌，李 辉
（北京工业大学 嵌入式软件与系统研究所，北京 100022）

摘 要：近年来提出了一些调度机制的实现方案，但这些方案不能实现由用户来选择调度算法和多算法集成，不能给用户提供统一的用户使用界面。为此，提出了一种新的调度机制，它能够给用户提供统一的使用界面，支持所有的调度算法且隐含实现的细节，使用户更方便地使用各种调度算法。在 Linux上进行了实现，给出了主要的数据结构和实现过程，对结果进行了分析。

关键词：调度机制；实时算法；Linux
中图分类号：TP319 文献标志码：A 文章编号：1001-3695(2009)01-0165-03

Research of embedded operating system scheduling mechanism

HE Fu-gui, HOU Yi-bin, LI Hui
(Embedded Software & Systems Institute, Beijing University of Technology, Beijing 100022, China)

Abstract: In recent years, some scheduling schemes were presented, but these schemes could not support for user to choose scheduling algorithm and multi-algorithm integration, and could not offer a uniform operating interface for users. The paper presented a new scheduling mechanism, it could offer a uniform operating interface, and could support available scheduling algorithms, and could hide the implementing details, and let user use all algorithm conveniently. It realized in Linux, presented principal data structure and realizing procedure, and gave experimental result and analysis.

Key words: scheduling mechanism; real-time algorithms; Linux

引言

实时系统的发展已有四十多年的历史，主要应用于嵌入式领域。运行在这些系统中的操作系统被称为实时操作系统。实时调度是实时操作系统的核心，它保障实时系统的实时性。实时调度理论在过去的几十年中进行了广泛的研究，提出了许多种实时调度算法，如 RM[1]（rate-monotonic 单调速率）算法、EDF[1]（earliest deadline first 截止期最早最优先）算法、DM[2]（deadline monotonic）算法、LFS[3]（least slack first 空闲时间最短最优先）算法、HVF[4]（highest value first 价值最高最优先）、HVDF[4,5]（highest value density first 价值密度最大最优先）、PTS(preemption threshold scheduling)[6]、DS[7]（deferrable server）算法、SS[8]（sporadic server）算法、DDS[9]（deadline deferrable server）算法、DSS[9]（deadline sporadic server）算法、DES[9]（deadline exchange server）算法、TBS[10]（total bandwidth server）算法和 CBS[11]（constant bandwidth server）算法等。

这些算法对周期任务系统和非周期任务系统的调度从理论上进行了分析，得出了相应的可调度的充分条件和必要条件，为实时系统的发展奠定了理论基础。

近年来对调度机制提出了一些改造的方案，能够运行一些实时调度算法，以适合实时调度的要求。

调度机制简介

. 两层调度框架

文献[12]提出了两层调度框架，它把调度结构分两部分：调度器（dispatcher）和分配器（allocator）。调度器运行在内核态，分配器可运行在内核态或用户态。定义了四个通用的调度属性：优先级（priority）、开始时间（start time）、完成时间（finish time）和预算（budget），然后靠分配器对四个属性不同的解释来实现各种不同的调度算法，取得了很大的灵活性，并在 Linux上进行了实现，称为 RED-Linux。调度框架如图 1所示。

. 开放的实时系统调度框架

文献[13]提出了一种实时调度框架，如图 2所示。它能够调度实时和非实时任务，使用一些 CPU利用率为常数的服务器来执行硬实时任务，而非实时任务作为背景程序来执行。调度器有一个服务器任务队列，这个队列按照 EDF调度算法来调度；每个服务器再按照某一种算法调度相应的任务队列。服务器是动态的，当有相应的任务需执行时，调度器建立一个服务器。当这个任务执行结束时，调度器销毁这个服务器。所有的非实时任务运行在一个优先级最低的时间共享的服务器上。这种方法实际上将原来的一个调度器分成两个，以实现不同的调度算法。

. 硬实时操作系统 的调度机制

RTLinux是在原 Linux内核的基础上增加一个实时内核，

收稿日期：2008-03-16；修回日期：2008-05-05 基金项目：北京市教委重点资助项目（KZ200510005006）

作者简介：何福贵（1966-），男，博士后，主要研究方向为实时操作系统、嵌入式软硬件协同设计（hefugui@163.com）；侯义斌（1952-），男，教授，博导，主要研究方向为新型计算机交互技术、网页可控的嵌入式系统、Internet理论与技术、中文信息处理；李辉（1973-），博士研究生，主要研究方向为嵌入式软硬件协同设计。

形成一个双内核结构。在实时内核中，RTLinux提供实时调度模块。它把调度模块的编写任务交给系统的使用者，使用者编写的模块可以很容易地加入到系统中，这样，使用者可根据自己的不同需求，采用适宜的调度算法。RTLinux本身提供了两个实时调度模块。RTLinux实现了采用不同的实时调度算法进行调度，但算法的编写由用户来完成，这就增加了程序开发的难度。

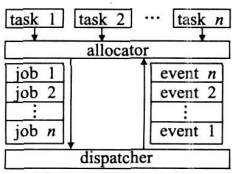

图1 两层调度框架

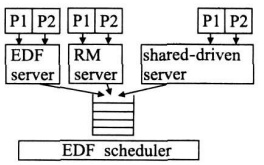

图2 开放的实时系统调度框架

从以上的分析可以看出，这些调度框架有以下不足之处：a)支持的调度算法很少；b)需要用户参与实时调度算法的编写，难度较大；c)对调度参数的解释没有统一的规定，容易造成歧义性。为了克服这些不足，本文提出了一种新的调度框架，可以实现对现在所有实时调度算法的支持，对用户提供了使用的统一接口，使调度规范化。

新调度机制

新调度机制的结构如图 3所示，调度模型支持双层调度。如果采用双层调度机制，第二层调度器所指向的任务其实是一个服务器，这个服务器去调度它指向的一个任务队列。

新调度机制的核心是 SW，它向用户提供使用调度算法的统一界面，应用程序通过提供的系统调用使用 SW。SW以下对用户是透明的，具体实现的细节包含在操作系统中，简化了用户的操作。用户通过系统调用选择单层调度和双层调度，同时在各层可选择各种调度算法(包括实时和非实时调度算法)，系统可以支持所有的调度算法。

新调度机制的实现

由于 Linux源代码开放，本文选择 Linux为实现目标。

. 主要数据结构

核心数据结构为调度节点数据结构，其作用是向上层用户提供统一的使用界面。此数据结构中包含有各种调度算法的数据结构，这些算法的数据结构作为一个共同体(union)，每一个调度节点数据结构只能包含各种算法的数据结构中的一个。

调度节点数据结构描述如下：

```
struct scheduling_inode {
  调度公有的属性;
  指向任务结构的指针;
  union {
    struct rm_scheduling_inode rm_inode; //RM调度节点
    struct edf_scheduling_inode edf_inode; // EDF调度节点
    ……
    } a_inode;
  };
```

每一种调度算法都有其对应的数据结构，以 EDF算法结构为例：

```
struct edf_scheduling_inode
{
  unsigned long start_time; /开始时间
  unsigned long calculate_time; /计算时间
  unsigned long finish_time; /完成时间
  union{
    unsigned long period; /周期
    unsigned long interval; /最小间隔
    } u
  unsigned long deadline; /截止期
  unsigned long weight_value; /权值
}
```

为了能够操作调度节点，在 Linux任务数据结构 struct task_struct结构中增加指向调度节点的指针。

. 实现过程函数

. . 各种算法的优先级计算函数

每一种实时调度算法都有一个计算优先级的函数，在此函数中可加入接收测试，如果通不过测试，则作为非实时任务来执行。

. . 各种算法的初始化函数和释放函数

各种算法初始化函数的执行应在进程建立时执行。初始化的功能是首先在内存申请一个调度节点，然后对调度节点相应的成员进行赋值。释放函数应在进程释放时执行，释放函数是释放调度节点所占用的内存空间，从所在队列进行脱链等相关操作。释放函数的过程与初始化函数的过程正好相反。

. . 设置调度参数和读取调度参数

调度参数可以由用户进行设置。这个函数要解决的关键问题是：给定一个参数名和参数的值，如何使这个参数名与 scheduling_inode结构的成员名进行比较和匹配。此函数的代码较长，简单地描述如下：

首先定义一个结构体 struct param{ char* name; offset}。这个结构包含两个成员：a)Name表示 scheduling_inode结构的成员名；b)Offset表示此成员在 scheduling_inode结构中的相对位移。然后定义 struct param结构的一个数组，每一个数组成员表示 scheduling_inode结构的一个成员名和它的相对位移，数组的大小由用户可访问的参数数量决定。最后在设置参数的函数中让给定的参数名与这个结构数组的每一个 name成员相比较，如果出现相同的一对，就以这个数组的位移为相对位移找到对应的 scheduling_inode结构的成员名进行赋值，从而修改此成员的值。

读取 scheduling_inode结构成员值的方法与此类似。

3.2.4 对 Linux的 schedule函数的修改

Linux在 schedule[14]函数中实现进程调度的功能。进程优先级的计算是在 schedule函数中进行的，在 schedule函数调用 goodness函数。Goodness函数是 Linux计算优先级的函数。新调度机制使用任务所指向的调度节点数据结构来进行计算。为了保持与 Linux的兼容性，在新调度的参数没有定义的情况下，可使用原 Linux优先级的计算方法；在新调度的参数存在的情况下，优先使用在新调度的参数计算优先级。

结果分析

. 调度机制的复杂性分析

Linux进程的优先级计算和从就绪队列选择一个优先级最高的进程投入运行都是在 schedule()函数中进行的，其中 goodness函数计算进程的优先级，其他部分的作用从就绪队列选择一个优先级最高的进程。从这一段程序可以看出，每一次

调度都要把就绪队列的所有任务访问一次, 所以 Linux调度的复杂性与就绪队列中任务的个数成正比。本文提出的新调度机制的复杂性与 Linux相同, 这样在不增加调度复杂性的前提下, 可支持多种调度算法, 实现了调度算法的集成。

. 实验测试

本文采用的 Linux版本为 Linux-2.4.20。在内核修改完成后, 设计一组运行时间逐渐增多的程序来进行测试, 分别测试在 Linux调度机制和新调度机制的程序运行时间, 测试运行的平台为桌面计算机。其配置如下: Celeron 2.00 GHz CPU, 内存 256 MB, 磁盘为 ST31640021A。运行的平台为 Red Hat Linux 7.3。测试结果如图 4所示。从图中可以看出新调度机制可以有效地减少程序的运行时间。

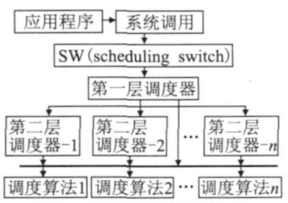

图3　新调度机制的结构

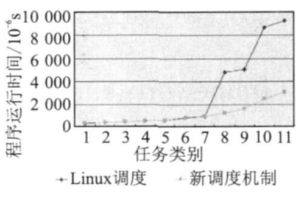

图4　程序执行时间比较

结束语

实时调度是嵌入式操作系统的核心, 现在已发展了许多实时调度算法, 调度算法是通过调度机制实现的。为了使嵌入式操作系统的调度机制更具有适用性, 本文提出了一种新的调度机制, 这种调度机制能够克服现有调度机制存在的缺点, 并且选择 Linux为实现目标。测试结果表明, 这种调度机制具有很强的实时性和通用性, 能够满足现有的和未来的实时调度发展的要求。

参考文献:

[1] LIU C L, LAYLAND J W. Scheduling algorithms for multiprogramming in a hard real-time environment[J]. Journal of the ACM, 1973, 20(1): 46-61.

[2] LEUNG J Y T, WHITEHEAD J. On the complexity of fixed-priority scheduling of periodic, real-time tasks[J]. Performance Evaluation, 1982, 2: 237-250.

[3] DERTOUZOS M L, MOK A K. Multiprocessor on-line scheduling of hard-real-time tasks[J]. IEEE Trans on Software Engineering, 1989, 15(12): 1497-1506.

[4] JENSEN E D, LOCKE C D, TODUDA H. A time-driven scheduling model for real-time operating systems[C]//Proc of the 6th IEEE Real-time Systems Symposium. San Diego: IEEE Computer Society Press, 1985: 112-122.

[5] BUTTAZZO G, SPURI M, SENSINI F. Value vs deadline scheduling in overload conditions[C]//Proc of the 19th IEEE Real-time Systems Symposium. Pisa: IEEE Computer Society Press, 1995: 90-99.

[6] SAKSENA M, WANG Yun. Scalable real-time system design using preemption thresholds[C]//Proc of the 21st IEEE Real-time Systems Symposium. Los Alamitos: IEEE Computer Society Press, 2000: 25-34.

[7] STROSNIDER J K, LEHOCZKY J P, SHA L. The deferrable server algorithm for enhanced aperiodic responsiveness in hard real-time environments[J]. IEEE Trans on Computers, 1995, 44(1): 73-91.

[8] SPRUNT B, SHA Lui, LEHOCZKY J. Aperiodic task scheduling for hard-real-time systems[J]. Real-time Systems, 1989, 1(1): 27-60.

[9] GHAZALIE T M, BAKER T P. Aperiodic servers in a deadline scheduling environment[J]. Real-time Systems, 1999, 9(1): 31-67.

[10] SPURI M, BUTTAZZO G. Scheduling aperiodic tasks in dynamic priority systems[C]//Proc of the 16th IEEE Real-time Systems Symposium. Pisa, Italy: [s.n.], 1995: 210-219.

[11] ABENI L, BUTTAZZO G. Integrating multimedia applications in hard real-time systems[C]//Proc of IEEE Real-time Systems Symposium. Madrid, Spain: [s.n.], 1998.

[12] WANG Yu-chung. Design and implementation of RED-Linux[D]. California: University of California, 2002.

[13] DENG Z, LIU J W S. Scheduling real-time application in open sysem environment[C]//Proc of the 18th IEEE Real-time Symposium. San Francisco: [s.n.], 1997: 308-319.

[14] 毛德操, 胡希明. Linux情景源代码分析(上册)[M]. 杭州: 浙江大学出版社, 2001: 263-398.

(上接第 139页)针对组织和项目 QA活动的策划、跟踪, 支持过程符合性评价。相关活动包括 QA活动策划、QA活动跟踪、不符合问题的状态跟踪、项目过程状态评价。

5)人员技能管理子系统　该系统提供企业人员基本信息查询、人员能力框架定义和人员能力评价等功能。

. . 度量数据使用层

度量数据使用层包括服务于企业各类角色的度量视图、PCB和交流平台。

1)度量视图　它为企业不同角色人员提供与业务目标及度量目标相适应的分析结果展示功能。

2)PCB　它为度量数据的分析提供基准支持。

3)交流平台　它提供与度量活动相关的一个交流平台, 包括专题讨论、专家咨询等功能。

结束语

企业集成化软件过程度量系统支持企业度量数据的收集、分析及使用, 提供项目软件过程动态监控功能及各级业务单元经营状态分析功能, 可以满足企业不同层级各类角色的度量需求。目前, 该系统已在东软集团得到了应用, 有效提高了企业软件过程度量活动的效率及目的性, 大大提高了项目度量活动与企业度量需求的适配性, 对提升项目及组织生产效率有很大帮助。

参考文献:

[1] CMMI Product Team. Capability maturity model integration(CMMI) for development version 1.2[R]. CMU/SEI-2006-008. [S.l.]: SEI, 2006.

[2] MCGARRY J, CARD D, JONES C, et al. Practical software measurement: objective information for decision makers[M]. [S.l.]: Addison-Wesley, 2002.

[3] PARK E R, GOETHERT B W, FLORAC A W. Goal-driven software measurement: a guidebook[R]. CMU/SEI-96-HB-002. [S.l.]: SEI, 1996.

[4] 钱红兵, 朱丽娟, 曹惠民. 基于 CMM的软件过程度量系统的研究与设计[J]. 计算机应用研究, 2004, 21(6): 49-52.

[5] 王青, 李明树, 刘霞. 一种支持软件过程控制和改进的主动度量模型[J]. 软件学报, 2005, 16(3): 407-418.