第33卷 第21期 计 算 机 工 程 2007年11月
Vol.33 No.21 Computer Engineering November 2007
·软件技术与数据库· 文章编号：1000—3428(2007)21—0028—03 文献标识码：A 中图分类号：TP393

基于多参数的μC/OS-Ⅱ任务优先级和调度方法

周本海[1]，王溪波[1,2]，乔建忠[1]，沈国文[1]
（1. 东北大学信息科学与工程学院，沈阳 110004；2. 沈阳工业大学信息科学与工程学院，沈阳 110023）

摘 要：在μC/OS-Ⅱ进行实时任务调度时，可以使用单一的调度算法分配任务优先级。优先级判定标准的片面性、“错过率”较高的截止期，影响了μC/OS-Ⅱ的实时调度性能。该文提出了多参数任务优先级分配策略和μC/OS-Ⅱ任务的调度方法，实验证明，该方法截止期的平均错过率为60.1%，有效地改善了μC/OS-Ⅱ的实时调度性能。

关键词：μC/OS-Ⅱ；实时任务；优先级；多参数；截止期错过率

Task Priority and Scheduling Method Based on Multi-parameter in μC/OS-Ⅱ

ZHOU Ben-hai[1], WANG Xi-bo[1,2], QIAO Jian-zhong[1], SHEN Guo-wen[1]
(1. School of Information Science and Engineering, Northeastern University, Shenyang 110004;
2. School of Information Science and Engineering, Shenyang University of Technology, Shenyang 110023)

【Abstract】When the real-time tasks are scheduled, μC/OS-Ⅱ takes use of single algorithm to assign the priority of tasks. Because of the singularity of determinant standard of priorty, the high deadline missing ration of real-time tasks influences the real-time scheduling performance seriously. This paper advances the assigning policy of priority of multi-parameter tasks and the task scheduling method in μC/OS-Ⅱ. Experimental results show that DMR of multi-parameter scheduling algorithm is 60.1%, and improvement of real-time scheduling performance.

【Key words】μC/OS-Ⅱ;real-time task; priority; multi-parameter; deadline missing ration(DMR)

在实时系统中，计算任务的正确性依赖于计算过程中逻辑上的正确性和输出结果时间的及时性，相应的处理过程必须在规定的时间内完成[1]。目前实时任务的调度方法有：单调速率算法(rate-monotonic, RM)[2]，截止期最早最优先(earliest deadline first, EDF)[2~3]，空闲时间最短优先(least slack first, LSF)和最早放行优先[4]，价值最高优先[5]，价值密度最大优先[6]等策略。

基于优先级抢占调度策略的实时内核，μC/OS-Ⅱ任务的优先级通常由一个特征参数来确定，如截止期、空闲时间、关键性(任务的重要程度)等。优先级仅由某个特征参数来确定是不够的，比如EDF策略将最高优先级指派给具有截止期最早的任务，但是截止期短的任务不一定是关键的。在任务调度算法的基础上，笔者综合任务的多种特征参数来考虑任务优先级别，提出了多参数任务优先级分配策略和在μC/OS-Ⅱ中任务的调度方法，提高了μC/OS-Ⅱ的任务调度成功率，降低了任务的截止期错失率，改善了μC/OS-Ⅱ的实时调度性能。

1 μC/OS-Ⅱ任务调度

1.1 μC/OS-Ⅱ内核调度流程

μC/OS-Ⅱ调度任务的流程(图1)如下：

(1)通过 OSTaskCreate()函数建立任务，OSTaskCreate()需要4个参数：任务代码的指针，任务开始执行时传递给任务的参数的指针，分配给任务的堆栈的栈顶指针，分配给任务的优先级。

(2)初始化任务控制块TCB，及其该任务的堆栈。

(3)将优先级索引表上的优先级索引指针与其对应的任务控制块相连，系统判断优先级最高的就绪任务并进行调度。

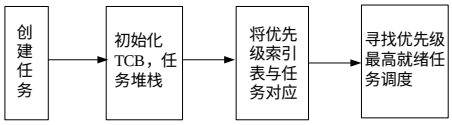

图1 μC/OS-Ⅱ调度流程

1.2 调度特性

μC/OS-II的任务调度具有以下特点：

(1)所有被创建的任务经调度而被执行，当且仅当任务的优先级最高，并且状态是“就绪状态”时，才可运行。

(2)正在处于运行状态的较低优先级任务，会被处于就绪状态的高优先级的任务所抢占，而变为就绪状态。

(3)运行的任务由于某种原因而被挂起，因此下一个就绪的最高优先级任务会被调度并运行。

(4)在给定的任意时间，仅有一个任务可以运行。

2 常用的实时任务调度算法

速率单调(rate-monotonic, RM)算法，任务周期越短，优先级越高。该算法的优点为：根据任务的执行周期设置优先级，有较小的运行开销；缺点为：执行周期最短的任务不一定是最重要的。

最早截止期优先(earliest-deadline-first, EDF)算法，每个

作者简介：周本海(1981－)，男，博士研究生，主研方向：操作系统，并行计算；王溪波，博士、教授；乔建忠，教授、博士生导师；沈国文，硕士研究生

收稿日期：2007-06-19 **E-mail：**zhoubenhai_neu@126.com

任务绝对时限越短，优先级越高。EDF 算法有着较小的调度开销，但对多个任务具有同一优先级的情况考虑不足。另外，EDF 算法在超载系统中，性能会急剧下降。

最小空闲时间优先(least-slack-time-first, LSF)算法，空闲时间越小，优先级最高。由于空闲时间动态变化，任务切换次数增多，因此会产生抖动现象。

由于这种单独的参数的优先级驱动算法，存在某种不足，因此可以将多种调度算法进行综合考虑，改善这些缺陷，更好地进行任务调度。

3 多参数 μC/OS-II 任务优先级分配策略

3.1 μC/OS-II 的任务模型

在描述 μC/OS-Ⅱ的任务模型之前，对于任务 r_i，具有下面的参数：

(1) a 表示任务的到达时间，即任务被启动并准备执行的时间(也就是说，任务由其他状态转变为就绪态的那个时刻)。

(2)C 表示任务的最坏情况执行时间(worst case execution time, WCET)，即任务在最坏情况下，无中断执行所需的处理器时间。WCET 常作为实时调度中任务对处理机的可能占用情况的一个衡量值。

(3)c 表示任务的剩余执行时间，即任务的最坏情况执行时间与任务已经执行时间的差。

(4)d 表示任务的绝对截止期，即任务在这个时间应该完成执行并产生一个有价值的结果。

(5)D 表示任务的相对截止期，有 $D = d - a$ 。

(6)T 表示任务的周期。

(7)P 表示任务的优先级。

(8) S 表示空闲时间，t 为当前时间， $s = d - t - c$ 。

设含有 n 个任务的集合 $S = \{J_1, J_2, \cdots, J_n\}$，它们的周期分别是 $T_1, T_2, \cdots, T_n$，每个周期内的最坏情况执行时间分别为 $C_1, C_2, \cdots, C_n$，相对截止期为 $D_1, D_2, \cdots, D_n$，对于第 $i(1 \le i \le n)$ 个任务 J_i，必须满足 $C_i \le D_i \le T_i$ 。

3.2 μC/OS-II 中任务优先级的确定

在 μC/OS-II 中，任务到达时，将任务表示为一个四元组，$J(a, D, T, c)$。为了克服 μC/OS-II 中任务调度的不足之处，本文综合使用截止期最早最优先(EDF)，空闲时间最短优先(LSF)，单调速率算法(RM) 3 个调度方法的特征参数分配 μC/OS-II 的任务优先级。空间坐标系确定任务优先级见图 2。

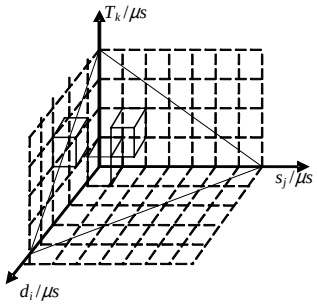

图 2 空间坐标系确定任务优先级

根据 EDF 算法定义，任务的优先级为

$$P(J_{ijk}, d_i, s_j, T_k) = d_i$$

根据 LSF 算法定义，任务的优先级为

$$P(J_{ijk}, d_i, s_j, T_k) = s_j$$

根据 RM 算法定义，任务优先级为

$$P(J_{ijk}, d_i, s_j, T_k) = T_k$$

任务的优先级可由相对截止期 d_i 、空闲时间 s_j 、周期 T_k 3 个维度来确定。

由截止期、空闲时间、周期这 3 个特征参数构成的空间坐标系中可以得出，如果只使用截止期最短优先算法，空闲时间与周期的长短不会对优先级产生影响。而利用这 3 个特征参数综合确定优先级时，截止期越短，空闲时间越短，周期越短的任务优先级越高。在此空间坐标系中，每个任务的权重为

$$w(J_{ijk}, d_i, s_j, T_k) = d_i + s_j + T_k$$

有些任务具有相同的权重，因为在 μC/OS-Ⅱ中，每个任务被分配了唯一的优先级，所以可以将截止期、空闲时间、周期这 3 个特征的重要性由高到低排列，根据任务特征参数的重要性不同，为任务分配不同的优先级。任务的优先级为

$$pr = s_j + (d_i - 1)[2(d_i + s_j + T_k) - d_i - 2]/2 + (d_i + s_j + T_k - 1)(d_i + s_j + T_k - 2)(d_i + s_j + T_k - 3)/6$$

其中，i, j, k 分别表示任务在截止期、空闲时间、周期序列中的位置；任务权重可由 $w=i+j+k$ 表示，则任务的优先级可以表示为

$$pr = j + (2w - i - 2)(i - 1) + (w - 1)(w - 2)(w - 3)/6$$

4 多参数 μC/OS-Ⅱ任务调度算法

笔者扩展了 μC/OS-Ⅱ任务控制块，将特征参数添加其中。创建 3 个任务链表，将任务截止期、空闲时间、周期按时间长短排列。用链表的序号表示任务在该调度算法中的权重。新的任务控制块结构如下：

```
typedef struct   OS_TCB {…
INT8U    a;   //任务到达时间
INT8U   s;   //任务空闲时间
INT8U    T;   //任务周期
INT8U    d;   //任务截止期
 struct      OS_TCB
 *EDF_PRIO, *EDF_NEXT; //EDF 任务链表前驱、后继指针
 struct      OS_TCB
 *LSF_PRIO, *LSF_NEXT; //LSF 任务链表前驱、后继指针
 struct      OS_TCB
 *RM_PRIO,    *RM_NEXT; //RM 任务链表前驱、后继指针
 INT8U    PRIO; //用于记录任务的优先级
 INT8U    EDF_Right, LSF_Right, RM_Right;
 //记录各调度算法任务的权重
} OS_TCB;
OS_TCB   *OS_EDF_LINK; /*EDF, LSF, RM 任务的头结点*/
OS_TCB   *OS_LSF_LINK;
OS_TCB   *OS_RM_LINK;
```

μC/OS-Ⅱ在每次调度时刻，需要对系统中任务的优先级作出判断，寻找优先级最高的任务，使其调度运行。新任务到来时，计算出该任务在各个调度算法链表的权重。系统要根据这 3 个维度的特征参数确定任务的优先级。任务 EDF,LSF,RM 特征参数的提取算法描述如下：

(1)EDF 算法特征参数提取

1)EDF 任务链表中任务按截止期升序排列。任务在链表中的位置为任务的权重。

2)新任务到达时，从头结点开始，即

```
OS_Pedf= OS_EDF_LINK-> EDF_NEXT
```

3)将其截止期与链表中任务的截止期进行比较

while(OS_NEWTASK->d > OS_Pedf->d) OS_Pedf= OS_Pedf-> EDF_NEXT，确定新任务位置。

4)确定任务在 EDF 链表中的权重，即

OS_NEWTASK-> EDF_Right= OS_Pedf-> EDF_Right

从 OS_Pedf 指向的任务结点开始，向后搜索，将这些结点的 EDF_Right 值增 1，直至尾结点。

5)将新任务插入 EDF 任务链表。

(2)LSF 算法特征参数提取

1)LSF 任务链表中任务按空闲时间升序排列.任务在链表中的位置为任务的权重。

2)新任务到达时，从头结点开始，OS_Plsf= OS_LSF_ LINK-> LSF_NEXT。

3)将其空闲时间与链表中任务的空闲时间进行比较 while(OS_NEWTASK->s > OS_Plsf->s) OS_Plsf= OS_Plsf-> LSF_NEXT，确定新任务位置。

4)确定任务在 LSF 链表中的权重，即

OS_NEWTASK-> LSF_Right= OS_Plsf->LSF_Right

从 OS_Plsf 指向的任务结点开始，向后搜索，将这些结点的 LSF_Right 值增 1，直至尾结点。

5)将新任务插入 LSF 任务链表。

(3)RM 算法特征参数提取

1)RM 任务链表中任务按周期升序排列。任务在链表中的位置为任务的权重。

2)新任务到达时，从头结点开始，即

OS_Prm= OS_RM_LINK-> RM_NEXT

3)将其周期与链表中任务的周期进行比较 while(OS_NEWTASK->T > OS_Prm->T) OS_Prm= OS_Prm-> RM_NEXT，确定新任务位置。

4)确定任务在 RM 链表中的权重，即

OS_NEWTASK->RM_Right=OS_Prm->RM_Right

从 OS_Prm 指向的任务结点开始，向后搜索，将这些结点的 RM_Right 值增 1，直至尾结点。

5)将新任务插入 RM 任务链表。

(4)优先级的最终确定

将任务 3 个维度的特征参数提取后，建立一个优先级判定函数，将特征参数传入任务优先级判定函数以确定最终的优先级，之后系统就可以根据任务的优先级进行调度了。优先级判定函数如下：

```
INT16U PRIO(INT8U OS_EDF,INT8U OS_LSF,INT8U OS_RM)
{INT8U OS_P; OS_PRIO;
 OS_P= OS_EDF+OS_LSF+OS_RM;
OS_PRIO=(OS_P-1)(OS_P-2)(OS_P-3)/6+(2*OS_P-OS_EDF-1)
(OS_EDF-1)/2+OS_LSF
Return OS_PRIO;//返回至 OS_TCB->PRIO}
```

5 μC/OS-Ⅱ中调度算法的比较

截止期错过率(deadline missing ration, DMR)的定义为：假设在某一段确定的时间内，错过其截止期的作业数为 n，正常完成的作业数为 m，则

$$DMR = \frac{n}{m+n}$$

DMR 是描述实时任务在实际运行中，实时性能是否满足需求情况的一个重要衡量值。

根据 μC/OS-Ⅱ的任务模型，将任务的到达时间 a 取值范围为 0ms~10ms，执行时间 C 范围为 1ms~9ms，截止期 d 范围为 10ms~15ms，周期范围为 10ms~15ms。随机建立 10 个任务，运用 EDF,LSF,RM 以及多参数调度算法分别进行任务调度。利用 μC/OS-Ⅱ中提供的统计函数 OSTaskStat()来计算实际的完成时间，根据此完成时间与该任务的截止期比较，看是否超过了截止期，并计算出该组任务的 DMR。进行 10 组上述的测试，几种调度算法的 DMR 比较如图 3 所示。

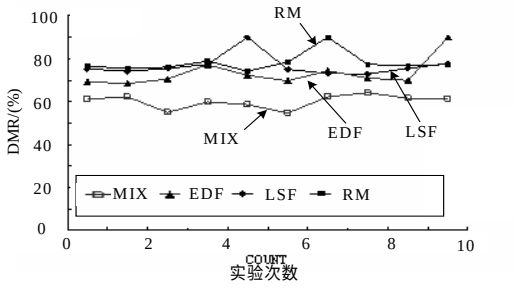

图 3 μC/OS-Ⅱ中各种调度算法的截止期错过率

从这个截止期错过率曲线图可知，在 μC/OS-Ⅱ中，单一使用 EDF,LSF,RM 调度算法的任务截止期错过率高于利用多参数确定优先级的调度算法的任务截止期的错过率。因此，使用多参数确定优先级截止期错过率低，这样对 μC/OS-Ⅱ的实时调度性能有很大的改善。

6 结束语

笔者分析、设计了利用多参数确定优先级和进行任务调度的方法，并通过扩展 μC/OS-Ⅱ的内核结构，将此调度方法应用到 μC/OS-Ⅱ中。由于截止期错过率是衡量系统实时性能的重要指标，因此比较了 μC/OS-Ⅱ内核单一使用 EDF, LSF,RM 算法与使用多参数确定任务优先级算法的截止期错过率。实验结果表明，μC/OS-Ⅱ内核使用多参数确定优先级的调度算法，任务的截止期错过率最低。采用该调度算法可以有效地改善 μC/OS-Ⅱ的实时调度性能。

参考文献

1 Stankovic J A. Misconceptions About Real-time Computing: A Serious Problem for Next-generation Systems[J]. IEEE Computer, 2003, 21(10): 10-19.

2 Liu C, Layland J. Scheduling Algorithms for Multiprogramming in a Hard Real-time Environment[J]. Journal of the ACM, 1973, 20(1): 46-61.

3 Haritsa J R, Livny M, Carey M J. Earliest Deadline Scheduling for Real-time Database Systems[C]//Proceedings of the 12th IEEE Real-time Systems Symposium. Los Alamitos, CA: IEEE Computer Society Press. 1991: 232-243.

4 Liu Y S, He X G, Tang C J, et al. Special Type Database Technology[M]. 北京: 科学出版社, 2000.

5 Jensen E D, Locke C D, Toduda H. A Time-driven Scheduling Model for Real-time Operating Systems[C]//Proceedings of the IEEE Real-time Systems Symposium. Washington, DC: IEEE Computer Society Press. 1985: 112-122.

6 Abbott R, Garcia M H. Scheduling Real-time Transactions[J]. ACM SIGMOD Record, 1988, 17(1): 71-81.