VxWorks系统的任务调度机制

于宏坤，丛 伟，王 勇
（空军工程大学 工程学院 陕西 西安 710038）

摘 要：针对多任务系统而言，调度是指根据一定的算法，将CPU分配给符合条件的任务使用，不同的系统任务调度机制不同。本文介绍VxWorks系统的任务调度策略和算法，分析优先级倒置产生的原因并给出解决方案，重点阐述静态表方式的实现方法并给出C语言程序框架。

关键词：调度；任务；优先级；倒置；静态表
中图分类号：TP316 **文献标识码**：B **文章编号**：1004-373X（2004）02-075-03

Multi-task Schduling Mechanism in VxWorks Operating System

YU Hongkun, CONG Wei, WANG Yong
(Engineering College, Air Force University of Engineering, Xi'an, 710038, China)

Abstract: Scheduling is assigned CPU to task according to stated arithmetic in multi-task system. Task scheduling methods differ in different system. In this paper, task sheduling's strategy and arthmetic in VxWroks OS are introduced. The reasons of priority invertion and solving method are analysed. The method of building static table and program frame in C language are presented.

Keywords: scheduling; task; priority; invertion; static table

嵌入式系统不但要满足应用的功能需求，更重要的是要满足应用提出的实时性要求。因此，嵌入式系统的关键在于，采用各种算法和策略，始终保证系统行为的可预测性，即在系统运行的任何时刻、任何情况下，任务调度程序都能为每个任务合理地分配资源，使每个任务的实时性要求都能得到满足。VxWorks系统提供的调度算法主要针对非周期性任务，并没有提供调度和管理周期性任务的机制，而且在优先级抢占调度中也存在着优先级倒置等问题。这些问题的存在，严重地影响了系统的实时性和可预测性，降低了系统的性能。因此有必要对这些问题进行深入讨论。

1 任务调度依据

VxWorks系统中，每个独立的程序称为一个任务，为了便于Wind内核对任务实施管理和调度，每个任务都有自己的上下文，包括：程序计数器（PC）、延时定时器（Priority）、时间片定时器（Counter）和堆栈（Stack）等。

调度程序的功能是在所有处于就绪状态的任务中选择最值得运行的任务投入运行，选择任务的主要依据是任务上下文中的Priority和Counter选项，其中Priority是任务的静态优先级，Counter是任务剩余的时间片，他是决定一个任务可否处于运行状态的重要标志。

2 任务调度策略

2.1 基于静态优先级的抢占调度

优先级是创建任务时，由系统或用户设置的整数；静态是指任务的优先级不随时间而改变，只能由用户进行修改。静态优先级指明了在被迫和其他任务竞争CPU之前，该任务被允许的最大时间片，即Priority的值。

所谓抢占是指，如果系统发现有一个任务转为就绪态，并且该任务的静态优先级比当前正在运行的任务的优先级高，内核立即停止当前任务的执行，然后切换到这个高优先级任务的上下文执行。

静态优先级是调度程序选择任务投入运行的标志，优先级越高，该任务得到CPU时间的机会也就越大。

2.1.1 优先级倒置

在静态优先级抢占方式中，调度程序保证高优先级的任务先执行。但是由于任务间的资源竞争问题会

收稿日期：2003-10-13

使得一个高优先级的任务被迫等待一个低优先级任务完成，才能执行。这种情况就是优先级倒置。

在图1中，t_1、t_2 和 t_3 的优先级由高到低，t_3 占有了由信号量保护的某种资源，进入临界区执行，当 t_3 在临界区执行时，t_1 就绪，t_1 抢占 t_3 执行（t_3 仍有信号量）。在 t_1 执行过程中，需要进入相同的临界区，这时需要取得相同的信号量，由于资源竞争，t_1 被阻塞，此时 t_3 先执行。然而，t_3 在执行时有可能被 t_2 抢占，t_3 再次阻塞（仍有信号量），这种状态可能持续下去，因而 t_1 将在下一个不确定的时间段内被阻塞。这种情况下，如果不采取措施，可能造成该实时系统的不可预测性。

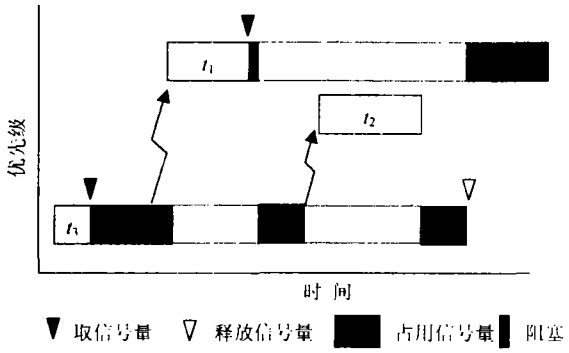

图 1　优先级倒置示意图

2.1.2　解决方案

（1）优先级继承

让拥有资源的任务以阻塞在该资源上的所有任务的最高优先级运行，直到他持有的所有资源全部释放，然后该任务返回正常优先级。

在图2中，当 t_1 阻塞在该信号量时，把 t_3 的优先级提高到 t_1 的优先级，这样不但保护了 t_3，也间接地保护了 t_1 不被 t_2 抢占。

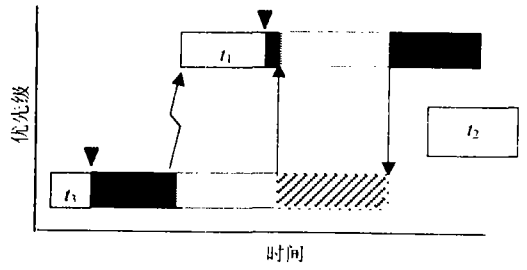

图 2　优先级继承解决优先级倒置

（2）极限优先级

把临界资源赋予极限优先级，他等于此时系统中的最高优先级加1。当任务到达临界区时，系统把极限优先级传递给该任务，使得他的优先级最高；当任务退出临界区后，系统恢复他的正常优先级，从而保证不会出现优先级倒置的情况。

在图2中，t_3 进入临界区时，立即把他的优先级升高到极限优先级，保证 t_3 能尽快退出临界区，进而释放其占有的资源。当高优先级的任务 t_1 执行时就不会出现等待低优先级任务 t_3 释放资源而阻塞的情况。

采用这种方案的好处是只需改变某个临界资源的优先级，不需改变任务的优先级就可以使多个任务共享这个临界资源：

```
void TaskA(void)
{  ……
   SetTaskPriority(RES_X_PRIO) ;
   // 访问临界资源
   SetTaskPriority(TASK_A_PRIO) ;
   ……
}
```

2.2　基于动态优先级的轮转调度

动态优先级是指只要任务拥有CPU，优先级就随着时间不断减小，当他小于0时，标记任务可以重新调度。动态优先级指明了同优先级的任务共享CPU之前，允许占用CPU的剩余时间片，即Counter的值。

Counter的初值与函数TaskSpawn（）中Priority参数的初值相同，在任务运行过程中，Counter不断减少，Priority保持不变，以便在Counter变为0时对Counter重新赋值。这说明，任务在运行过程中，Counter的减小给了其他任务得以运行的机会，直到Counter减为0时才完全放弃使用CPU。

2.3　静态表调度

用静态表构造周期任务的调度和管理程序，以满足VxWorks调度不同类型任务的需求。

（1）周期任务与静态表

周期任务是指每个周期运行一次的任务，其特点有：

①每个周期内任务运行一次。

②周期完成，任务被挂起，等待下一个周期到来，然后再运行。

③在规定周期内，任务若不能完成，将产生超时，要有相应的处理程序。

静态表是指在系统运行前根据各周期任务的实时要求用手工方式或辅助工具的帮助下生成一张任务的运行时间表，指明了各任务的起始运行时间以及运行长度，运行时间表一旦生成就不再变化，在运行时调度器只需根据这张表在指定的时刻启动相应的任务即可。

（2）静态表调度管理的功能

为适应周期任务的调度要求，静态表调度管理提供以下功能：

①建立静态表，登记周期任务的相关信息。

②周期任务一次运行结束，在静态表中记录任务的状态，并将其挂起。

③任务周期数到达时，判断该任务是否结束，在未超时的情况下，任务被挂起，否则报告并处理任务超时。

(3) 静态表调度管理的实现

数据结构

①TASKMAX：记录静态表中周期任务的最大数目。

```
#define TASKMAX 30;
```

②task_table：登记周期任务信息的静态表。

```
typedef struct TASK_TABLE_TYPE
{ int TaskId;
  int Priority;
  int Counter;
  BOOL TaskEnding;
};
struct TASK_TABLE_TYPE task_table[TASKMAX];
```

③current_task_count：记录系统中当前周期任务的个数。

```
int current_task_count = 0;
```

调度接口

①void Static_TickSet (int periodsLengh)

设置周期长度，该长度是所有周期任务时间的最大公约数。

②int Static_taskSpawn (int periods, char * name, int priority, int oprions, int stackSize, FUNCPTR entry, int arg1, ……, int arg10);

把周期任务的相关信息写进静态表，并创建该任务。其中参数periods记录该任务的周期个数，其他参数的含义与系统函数taskSpawn ( ) 一致。任务创建成功，返回任务ID号，否则返回ERROR值。

③void Static_taskEnding (int taskOrder)

结束周期任务的一次运行，参数taskOrder是周期任务在静态表中的序号。注意：序号必须从0开始，依次递增。

周期任务调度

任务调度的程序框架如下：

```
void Period_Scheduling ( void )
{  ……
   for( int t=0 ; t < current_task_count ; t++ )
   { task_table[t]. counter- - ;
     if ( task_table[t]. counter ! =0 ) continue ;
     if ( task_table[t].taskEnding = = FALSE )
     { //任务超时处理 }
     else
        { task_table[t].counter = task_table[t].
priority ;
          ……
        }
   }
   ……
}
```

3 结 语

VxWorks系统的任务调度机制保证了非周期任务之间的协调关系，优先级继承和极限优先级机制有效地解决了优先级倒置问题。静态表调度方式中的静态表在任务运行之前生成，因此具有运行时调度器开销较小、系统具有非常好的可预测性以及实时性验证方便等优点，所以静态表的引入丰富了VxWorks系统的任务调度机制。

参 考 文 献

[1] Micbael Barr. Programming embedded systems in C and C++ [M]. Reilly Press.

[2] 孔祥营，柏桂枝. 嵌入式实时操作系统VxWorks及其开发环境Tornado [M]. 北京：中国电力出版社，2002.

[3] 张丙平，VxWorks操作系统环境下一种周期任务管理方法 [J]. 航空计算技术，2003，(1).

作者简介 于宏坤 男，1973年出生，辽宁省丹东人，硕士。主要从事航空计算机分布式系统的研究。

丛 伟 女，1973年出生，辽宁省铁岭人，硕士，讲师。主要从事机载计算机软件的教学和研究。