实时操作系统中的时间车轮算法①

顾清山 杨 顺 张 亮 (辽宁工程技术大学 电信学院 辽宁 葫芦岛 125105)

摘 要: 操作系统中时间车轮算法被广泛应用，不过其最大调度时间有限，分层时间车轮算法也存在着算法复杂的不足，在二者基础上提出了对时间车轮的改进方法。该算法不但消除了时间车轮计时时限的缺点，而且比分层时间车轮在算法上简单很多，很适用于程序存储空间紧张的嵌入式实时系统。

关键词: 操作系统；最大调度；计时器；时间车轮；分层时间车轮

Time Wheel Algorithm in Real-Time Operating System

GU Qing-Shan ,YANG Shun, ZHANG Liang

(College of Information and Electronic Engineering，Liaoning Technical University，Huludao 125105, China)

Abstract: In the operating system, the time wheel algorithm is widely applied. But its biggest dispatch time is limited. The lamination time wheel algorithm also has the algorithm complex insufficiency. This article proposes the time wheel's corrective method based on that. This algorithm does not only eliminate the time wheel's time limitation, it is also easier than score strata time wheel algorithm. It is suitable for the procedure storage space tense embedded real time system.

Keywords: operating system; biggest dispatch; timer; time wheel; lamination time wheel

1 引言

嵌入式系统是当今最热门的领域之一，在其系统中，系统任务和用户任务经常需要进行调度和执行相应的任务，实时操作系统中经常需要使用时间轮换调度算法调度同优先级的就绪任务，内核中的任务调度器也必须周期地在相等优先级任务中执行实现设定好的上下文切换，以保证执行的公平性。在嵌入式网络设备中，各种通信协议调度数据进行重新传输和恢复协议的活动，实时操作系统中系统任务的延时、同步和通信等，都需要一个快速准确的计时器[1]。实时操作系统一般都提供了时间管理对象完成系统的计时任务。

2 软件计时器功能模型

一个软件计时器的工作模型分为两部分，一部分是系统时钟中断服务处理，一部分是在系统初始化时创建的计时器处理任务，并称其为工作者任务(worker task)。工作者任务于系统计时器中断服务连接起来组成了系统计时服务的功能模型。

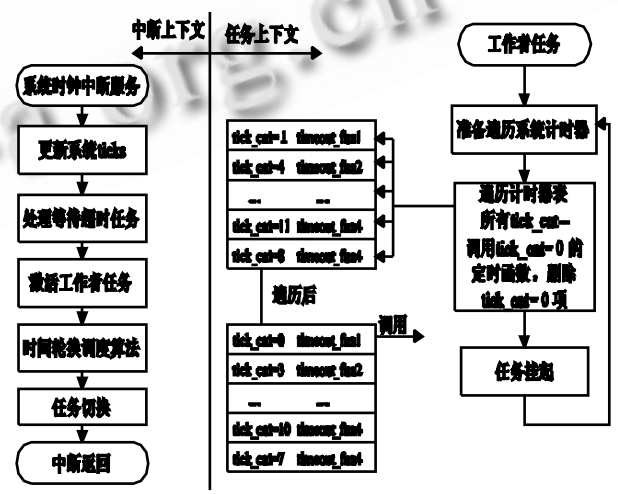

图 1 软件计时器的工作模型

软件计时器的工作模型如图 1 所示。在系统进入

① 收稿时间:2009-06-02

时钟中断服务程序后，系统首先更新系统 ticks，记录系统从开始运行到现在的滴答数。然后查询等待某一时间的任务是否等待超时(如任务调用 task_sleep 是否超时，系统等待某一内核对象是否超时等)。然后激活工作者任务，工作者任务进入就绪状态，工作者任务在处理所有系统计时对象完毕后会马上挂起自身等待下一次时钟中断激活该任务。在中断服务的结尾系统会调用时间轮换调度算法，对与被中断任务同优先级的任务实施分时调度。

软计时器应当允许有效的计时器的插入、删除、取消、以及更改。所以一般都用双向链表来链接系统中所有的计时器。由于计时器没有按特定的次序排列，系统维护计时器的所花费的时间是很昂贵的。每个系统滴答工作者任务都需要遍历整个链表并更改每个计时器的倒计数项。如果计时器计数值达到 0 时还需要调用它的回调函数。典型的解决方案是使用时间车轮方法。

3　时间车轮原理

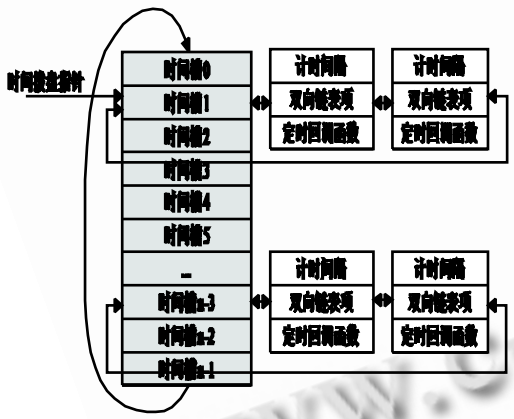

图 2　时间车轮框图

时间车轮是固定长度数组的结构，其中每个数组项代表一个时间单元[2]，我们称之为时间槽[3]。时间单元代表软件计时器的精度。每次系统滴答时钟拨盘下拨一个时间槽，拨到数组末尾以后折回，每个槽对应一个计时器链表，该链表中所有计时器具有相同的定时时刻。当时间拨盘指针到达该时间槽后，系统将会调用链表中每一个计时器的回调函数。时间车轮通过固定长度的数组和定时移动的时间盘指针，实现了软计时器的快速安装、删除、修改等操作。时间车轮实现模型如图 2 所示。图 2 中时间盘指针指向了时间槽 1，这是时间槽 1 对应的计时器双向链表为非空状态。该链表中所有的定时器定时时间到。系统将遍历该链表内所有的计时器对象，调用它们的回调函数，完毕后清空时间槽对应的计时器链表。

由时间车轮原理可知，当安装一个新的计时器事件时，时间拨盘的当前位置作为引用指针，决定了新计时器项存储到时间槽的哪个位置。例如每个时间槽表示跨越 10，时间槽个数为 10，时间指针位于时间槽 1，则开发者希望设置一个 40 的定时器时。新的定时器应该安装到时间槽的位置索引为：1+(40/10)%10=5 的时间槽上。

4　分层时间车轮原理

分层时间车轮类似于数字时钟，不是只有单个的时间车轮，而是按分层的次序组织多个时间车轮[4]。分层中的每个时间车轮具有不同的颗粒度。一个时钟拨盘与每个时间车轮相连。当分层的较底层的拨盘循环一周时，其上一层的时间车盘只下拨一个时间槽。例如需要表示分辨率为 100 最大时限为 5 的计时器模块。系统需要第一层时间车轮容纳 10 个时间槽来表示的时间。第二层时间车轮需要 60 个时间槽来表示的时间。第三层需要表示 5 个时间槽来表示的时间，共需要个时间槽。而当需要分辨率为 100 最大实现为 10 的计时时只需要个时间槽。分层时间车轮的实现模型如图 3 所示：

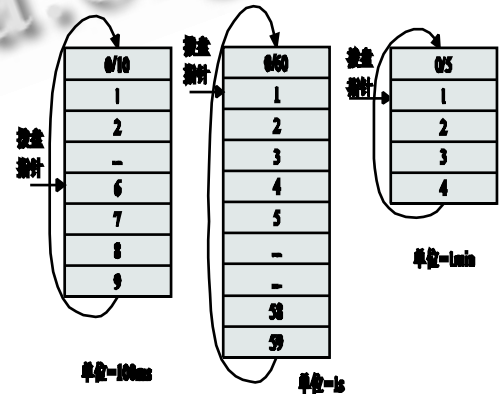

图 3　分层时间车轮框图

如图 3 所示，时间指针指向 2 分 2 秒 600 毫秒上，要想定时 2 分 3 秒 500 毫秒需要把计时器放到 4 分 5 秒 100 毫秒的时间槽上，所以先把计时器对象安

装到 4 分得时间槽上，等指针到达 4 分时，再把计时器移动到 5 秒的时间槽上，等到指针移动到 5 秒的时间槽上，依此类推。当 100 的时间指针移动到 100 时，调用计时器的回调函数。

5 改进型时间车轮算法

在操作系统中使用时间槽计数的方法对时间车轮进行改进。每个时间槽对应着有序的时间计时器链表。链表内每个计时器内含有一个计数值，该值代表本计数器和前一个计数器相差的时间车轮指针周期数。这样每一个车轮周期到来时只需将链表中第一个计时器的计数值减去 1，然后调用链表中所有计数值为 0 的计时器的回调函数，因为链表是有序的所以不用遍历整个链表，遇到非 0 值便可直接退出。这样每个系统滴答的计算量减少到了最低。只是在添加计数器时需要将计数器插入的排序的链表内，但由于使用了时间车轮。计数器被分散到了车轮内所有的时间槽中所对应的链表中，减少了链表的长度，从而减少了按序插入时所用的时间。计数时间车轮的系统模型如图 4 所示：

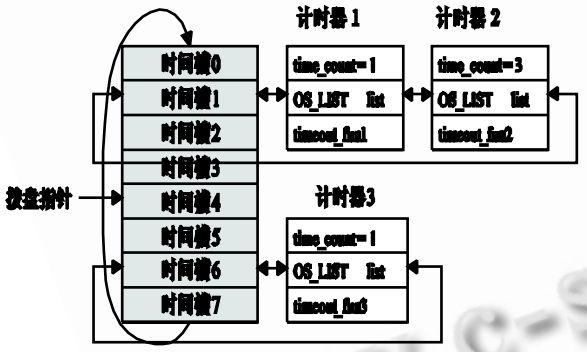

图 4　计数时间车轮框图

从图 4 可以看出如果时间槽的时间间隔为 10，则当流过 20 后计数器 1 的 time_count 减 1，time_count 等于 0，这时计时器 1 计时间隔到，系统调用其回调函数。同理当流过 50 后指针指向时间槽 1，计数器 1 的定时时间到。由于计时器 2 的 time_count 不等于 0 所以定时间隔未到，其值为 3 表示还需要 3 个车轮周期定时间隔才能到，而系统中车轮周期为，所以定时器 3 的定时时间为。如果想添加一个计时时间间隔为 130 的计数器 4，只需要将计时器 4 插入到时间槽 1 对应的链表上，添加后时间槽 1 对应的链表如图 5 所示。

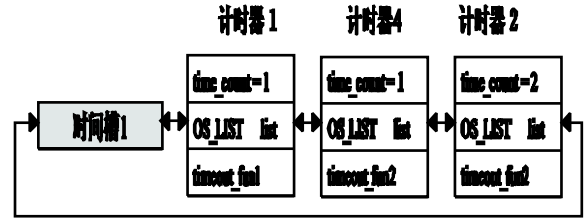

图 5　插入计时器时间槽对应的链表

如果想删除计时器 1，只需要将计时器 1 的 time_count 值加到计时器 2 上，然后从链表中将计时器 1 删除即可。

6 算法性能比较与分析

在时间车轮算法中，无论对于怎样的系统时间槽的个数是有限的，所能表示的最大可调度时间是有限的。本文以图 2 和 4 为例，令图 2 中时间槽个数为 8，对时间车轮和改进型车轮之间进行比较。

表 1　时间车轮和改进型时间车轮比较

	时间车轮	改进时间车轮
时间槽分辨率	10ms	10ms
时间槽个数	8	8
最大调度时间	80ms	不限
遍历程度	整个链表	部分链表
添计时器耗时	较少	较多

表 1 中可以看出，改进型车轮算法在最大调度时间和遍历链表的程度上都具有一定的优越性。表 2 和表 3 中给出了在两种算法中，当时间槽个数确定为 8 时，分辨率和最大调度时间之间的关系表。

表 2　时间车轮对应关系表

分辨率	10ms	20ms	30ms	40ms	50ms
最大调度时间	80ms	160ms	240ms	320ms	400ms

表 3　改进型时间车轮对应关系表

分辨率	10 ms	20 ms	30 ms	40 ms	50 ms
最大调度时间（ms）	不限	不限	不限	不限	不限

当然对于时间车轮算法最大调度实现问题，一个可能的办法是将所有超时的计时器按顺序排列到一个等待链表中，每一次拨动时间拨盘都要检查等待链表中的计时器是否可以加入到时间槽中。其次每次添加

溢出的计时器时，都需要遍历等待链表将计时器按顺序插入。当等待链表中有很多项时这样的操作代价是很高的。

分层时间车轮虽然可以以较少的时间槽就能容纳很长的定时时限，但它也带来的新的问题。本文以图 3 和图 4 为例，对分层时间车轮和改进型时间车轮的进行比较。

表 4　分层时间车轮和改进型时间车轮比较

	分层时间车轮	改进时间车轮
时间槽分辨率	100ms	100ms
时间槽个数	80	8
最大调度时间	10min	不限
遍历程度	较多链表	较少链表
添计时器耗时	较少	较多
算法实现程度	较难	较易

由表 4 中可知，时间车轮算法的最大调度实现时间可以变的更长，可是仍然是受限的。相对于改进的车轮算法，分层车轮算法在系统的代码设计更加复杂，必然会占用更多的代码空间；同时每次从高层的时间槽向底层的时间槽移动计时器时都需要遍历高层时间槽内所有的计时器[5]，并计算其移动到下一层时间槽的位置。虽然由于时间槽的原因，相对于时间车轮算法分散了每次遍历的个数，减少了每个系统滴答的计算量，但其总的计算量并没有减少。

和普通的时间车轮和分层时间车轮比起来，改进时间车轮算法在添加计时器时耗费了更多的时间，但消除了时间车轮计时时限的缺点，并且在调用次数最多的系统滴答内检查到时定时器的处理上和时间车轮基本相同。

7　结语

本文在车轮算法基础上提出了一种新的算法，可以实现用相同的空间可以得到比分层时间车轮更大的定时时限，却比分层时间车轮在算法上简单很多，很适用于程序存储空间紧张的嵌入式实时系统内。

参考文献

1 何福贵,侯义斌,李辉.嵌入式系统调度机制的研究.计算机应用研究, 2009,26(1):165－167.
2 刘晋,孙楠.时间车轮算法在 uC/OS-II 中的应用.计算机系统应用, 2008,17(11):80－81.
3 吴晶,熊璋,王晔.利用动态时间槽分配的多目标防冲突射频识别.北京航空航天大学学报, 2005,31(6):618－622.
4 俞雪山.嵌入式 MINIX 操作系统的设计[硕士学位论文].兰州:兰州大学,2007.
5 Massa AJ.颜若麟译.嵌入式可配置实时操作系统 eCos 软件开发.北京:北京航空航天大学出版, 2006.