VOS内核调度算法----位图加链表法

作者：[156439848@qq.com](mailto:156439848@qq.com); vincent\_cws2008@gmail.com

时间： 2020-09-07



VOS内核调度算法，初始设计考虑到相同优先级任务也必须支持，不能像ucos那样只能支持不同优先级，对用户来说比较不友好，还有相同优先级的任务是按时间片来调度的，默认的设计时间片是10ms, 不同优先级就必须服从高优先级（数值低的）必须完成后才能到低优先级执行，除非高优先级任务自己阻塞（延时，信号量或互斥锁等）。

初始想法任务调度直接使用链表来实现，其实也能正常使用，但是效率可能比较低，经过一些文献，发现很多都是基于位图来实现任务调度，于是就有了这个位图来支持不同优先级，链表来支持同优先级的多个任务的调度算法，算法总体不难，下面概况说明一下。

VOS目前支持最大优先级256个，从0开始到255结束，0表现优先级最高，255表示优先级最低，目前255优先级的任务是留给IDLE任务，因为IDLE任务是不可以阻塞，因为当所有高优先级任务都阻塞，假如IDLE任务也阻塞，还能运行哪个任务？所以IDLE就工作在两种状态，要不就是正在运行，要不就是在就绪任务表中，不可能在阻塞链表中。

这里要说明一下延时，延时有个专门的链表，把所有延时任务都连接在一起，而且是按闹钟时间从小到大来排序，这里要注意，信号量，互斥锁，这样阻塞对象也是有延时的，基本可以说阻塞的肯定有延时，所以可以在延时链表中找到阻塞的所有任务，就绪位图表中可以找到所有就绪任务，加上当前运行的任务就凑成了全部任务，在遍历全部任务时经常需要这样来遍历。

位图表表示优先级的好处是占用空间小，查找快，因为字节（8bit）的最低位置1可以通过查表方法来定位，然后按字节来遍历（VOS按INT型4个字节）来查找最高优先级，也就是位图中最低位置1的偏移位置，方法很简单，从低到高位，先比交是否都是0，如果都是0就下个字节（或者INT型），直到不是全0，就可以确定这个字节里有置1的位，然后查表就可以定位出偏移值，偏移值就是优先级。

链表是每个优先级对应一个链表，这个链表没链表头，开始就是任务控制块，如果链表中有多个任务，这些任务都必须优先级相同，基于时间片的调度，时间片调度就是被动调度，或者说中断调度。

这里要说明一下，如果根据优先级来定位第一个任务（没链表头，直接是任务），这里用了一个映射表，就是256个byte（unsigned char bitmap2task[256]）对应位图的每一个位，假如位图第n个位置1，直接这个bitmap2task[n]就返回链表头的任务id，通过任务id去查找任务指针，这里也是很巧妙，就是任务数组里，偏移号就是任务id,分别从0,1,2…m，m就是用户定义的任务个数。这样就可以查找到第一个任务指针，那么剩下连接到这个任务的相同优先级的兄弟任务就通过这个链表去一个一个查找。

最坏的情况就是全部都是相同任务，就变成了链表把所有任务链接起来，效率跟链表来组织就绪任务同样的效率。

信号量，互斥锁等对象，都有自己的阻塞任务链表(有链表头的)，把阻塞在这些阻塞对象上的任务按优先级从高到低全部链接起来，注意，这些任务也被同时链接到延时链表中，因为这些阻塞任务也有超时功能，所以任务恢复到就绪表时，记得要同时删除延时链表和阻塞对象链表，然后再位图就绪表中置位。