openEuler内核编程

课程讲稿

第6章 第3讲

《内核实时性》

软件所制

第6章 第3讲 内核实时性

**学时：**1学时

**教学目的：**学习实时系统的定义以及相关概念，能区分非抢占式调度和抢占式调度的区别，明确两种模式的优缺点和各自适用的使用环境。

**课程时间线：**

实时系统的定义及相关概念

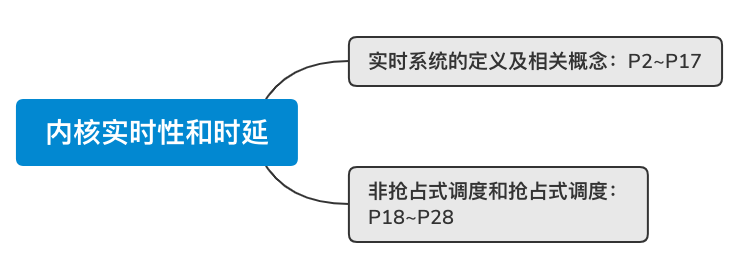
非抢占式调度和抢占式调度

25 min

45 min

**课外参考读物：**

**知识框图：**

****

**PPT讲稿：**



这一节课来讲解内核的实时性问题。



主要通过两方面来讨论，一方面是实时操作系统，另一方面是抢占式调度。



先看一下实时操作系统的定义：实时系统指系统的计算正确性不仅取决于计算的逻辑正确性，还取决于产生结果的时间。如果未满足系统的时间约束，则认为系统失效。什么意思呢？就好比说学生考试，考试时间两个小时，考题必须要在两个小时之内做，还得作对。如果你三个小时才作对，那没有用了，抬头一看监考老师都走了，照样是零分。



所以实时系统有两大特性，一是确定性。要求系统应具备在事先定义的时间范围内识别和处理系统所需要的大量数据的能力。二是实时性，要求对任务尽快的进行响应，不要有较长的延迟。要注意的是，在设计实时系统的时候，要优先保证实时性。



下面考虑一个问题，我们最常使用的Windows操作系统是不是一个实时操作系统？一个操作系统在面对变化的负载时，也就是各种不同强度的应用，都必须确定性的保证时间要求。但是我们用Windows系统的时候，很多时候程序的响应时间根本没办法保证，从最基本的开始时间，到多任务处理时，开启新任务会十分缓慢，有时甚至直接蓝屏。所以不论你的电脑配置多好，Windows的版本多新，都没有办法提供确定性。所以Windows并不是一个实时操作系统。



实时系统可以分为两类，一类是硬实时系统，一类是软实时系统。前一类要求所有任务都要在指定时间内完成。后一类要求绝大多数任务在指定时间内完成即可。我们下面分别来讲一下这两种操作系统。



先来看硬实时系统。在一些高精尖的领域中，比如航空航天、军事、核工业，对实时性要求非常高。比如卫星的变轨，精度要求非常高，在计划的时间点让你点火就必须要点火，要求点火多久就点火多久，差一点都不行，不然问题就大了。可能本来要去火星的一个探测器，由于系统实时性问题，点完火没来得及熄灭，轨道不知道偏到哪里去了，轻则再浪费一些燃料校准轨道，减少卫星寿命。重则任务直接失败，几十亿的投资直接打水漂。这里最核心的技术就是科学合理的任务调度，怎么样安排才能满足我所有的需求。



介绍几个有名的实时操作系统。第一是VxWorks。这个操作系统大名鼎鼎，美国的好奇号火星车上运行的就是这个系统。我们知道火星降落从进入大气到着陆一共只有七分钟的时间，这期间必须靠火星车完全自主操作，对实时性要求非常高，所以对操作系统的要求也是最为严苛的。同样这款来自于美国的软件服务于很多美国的公司，包括波音，以及军方的飞机项目等等。



那么我们中国有没有自己的实时操作系统呢？答案是肯定的，我国从2001年就开始研制航天器专用的操作系统SpaceOS，首次使用是2006年。根据公开的信息，我国的嫦娥三号采用的是SpaceOS2操作系统。我国嫦娥三号和嫦娥四号的月球着陆任务堪称完美，月球车运行也十分稳定，直接印证了我国SpaceOS的能力和稳定性。2013年起SpaceOS3也开始研制，预计不久我们就会看到搭载最新操作系统的航天器了。



说完硬实时系统再来说一下软实时系统。这类系统远没有硬实时系统要求那么严苛，只要绝大多数任务可以在指定时间内完成就可以了。换句话说，哪怕有个别几个任务稍稍晚了点也没有关系，不会造成什么致命的影响。就好比学校里老师给学生布置大作业，绝大多数同学都按时交了作业，个别学的慢的同学晚个一两天也没什么大问题，只要做完了作业，就是好同学。所以软实时系统注定没办法应用在对实时性要求极高的领域。



举一个软实时系统的例子，QNX。这个操作系统应用非常广泛，全世界70%以上的汽车制造上使用的都是基于QNX的车机系统。其核心是微内核架构，保证了其响应迅速，稳定性高，不易出错。微内核仅支持关键内容。



总的来讲，实时系统有一下三个特点。1.时间约束。2.可预测性。3.高可靠性。



时间约束，顾名思义，就是每一个任务都有要求的完成时间。通过对时间约束的严格性，又分为硬实时和软实时。



第二、可预测性。表明的是系统要对任务所需的完成时间进行判断，以确定是否能满足时限要求。对于程序里函数、命令运行的开销应该是有界的，这样才能保证运行时间的有界性。



第三、可靠性。实时系统对于外部环境的也有严格的要求，外部环境和实时系统高度相关。实时系统很可能需要长时间不间断的运转，来保证子系统正常工作。同时对于任何异常行为也要时刻准备反应。



不同的实时操作系统不尽相同，那么怎么样来评判一个操作系统的实时性呢？这里主要由两个指标，一个是中断的响应时间，一个是任务切换的时间。任务切换时间比较好理解，接下来说一下中断的响应时间是怎么一回事。



中断响应时间由四部分组成。分别是：关中断的最长时间，保护CPU内部寄存器的时间，进入中断服务函数的执行时间，以及开始执行中断服务历程的第一条指令时间。这里中断时间的计算不是系统运行不同任务的平均中断时间，而且最差情况下的中断时间。哪怕有一次反应慢了，那么系统整体上就认为反应很慢。



说完了实时系统，再来说一下非抢占式调度和抢占式调度。



说到调度问题，那么有一个前提，就是操作系统必须是支持多任务的操作系统。每一个任务都有四种基本状态。一是就绪态，也就是万事俱备只欠东风，只要CPU给我用，我的程序就能跑起来。二是挂起态，就是程序还没准备好，某些需要的资源还没到位，被堵在这里了，就称为挂起态。三是休眠态，如果系统不需要某一个任务运行，那这个任务就处于休眠状态，有需要的时候再唤醒。四是延迟态，是任务被延迟时所处的状态。



我们再看下调度的定义。给定一组实时任务和系统资源，确定每个任务何时何地执行的整个过程就是调度。任务调度系统仿佛是一个建筑工地的工头，在确定了建筑任务后，根据时间和建筑需求，被不同的工人派不同的活，来达到最优效果。在非实时的系统中，比如Windows系统，调度的工作是缩短响应时间，俗话说就是尽可能让电脑别卡，提高资源利用率。在实时系统中，调度的工作就是满足每个任务的时间约束，并且对外部的响应要及时回复。



总的来说，调度的根本目的就是尽可能最大限度的利用处理的时间，不让CPU出现空闲等待的状态。系统中只要有可以执行的进程，那么就一定会有进程被执行。如果可以运行的进程数目比处理器的数目多，那么就一定会有进程被挂起。事实上从操作系统启动之后一直会有大量的进程在后台运行，即使用户什么都不操作也在运行。这就导致了永远会有进程被挂起等待被执行。任务调度的基本任务就是从这些挂起的进程里选择一个来执行。



对于进程调度而言，也分为两种类型，一种是非抢占方式，另一种是抢占方式。下面我们对这两种进程调度方法进行详细的讲解。



先看非抢占式调度。指的是进程运行之后，不允许被其他进程中断，只能是当前任务运行结束或者是自愿放弃才可以。这种模式缺点很明显，稍微有点紧急的任务都没有办法插入进来，实时系统中作用不大。唯一的优点就是上下文切换少。



另一种是抢占式调度。调度系统可以根据某一个原则，来停止当前进程并将CPU分配给新的进程。这样的优点就是实时性非常好，可以保证优先级高的程序先运行。缺点是是要频繁调度，上下文切换较多。



抢占式调度的原则有。优先权原则。就是系统根据当前正在等待进程的优先级来排序，如果有比正在运行的任务优先级更高的任务，那么就挂起当前任务，执行高优先级任务。这就好比是救护车，甭管前方有多少辆车，有多少个红绿灯，都得给它让步，因为救人要紧，优先级是最高的。另一个原则是时间片原则，每个进程按照时间片来运行，当前时间片运行完毕后就挂起，然后等待下一个可用时间片。这种调度就像是游乐场里面的娱乐项目，比如说过山车，玩一圈就是3分钟，你玩完一轮想再玩就得继续排队去，一轮三分钟，人人平等。这样可以让每一个程序都有机会运行。

结合这两种调度原则，可以有三种调度机制。分别是优先级抢占式调度、时间片轮转调度、两者相结合的混合式调度。



采用基于优先级的抢占式调度的操作系统，每一个任务都有一个优先级。我们举一个例子。假设在t时刻，系统中只有一个任务在运行，没有任何排队的任务，这个时候可以说是岁月安好，一切都是那么完美，任务1安安静静的执行。然后时间走到了t+1时刻，系统中出现了一个任务2，处于挂起状态，并没有运行，但是这个任务2和任务1不太一样，挂了一个小星星，在这里我们表示更高的优先级。操作系统一看，任务2更重要，但CPU只有一个，任务1就只能先停停，先来执行任务2。任务1就转变为了阻塞态，等任务2运行完了再运行。

这里的优先级可以是预先就设置好的，比如例子里的任务2的优先级上来就比任务1要高。另一方面也可以设置动态优先级，比如任务1在运行的过程中，不断的被插入更高优先级的任务，导致任务1一直没有运行完。系统就可以动态的提高等待时间过长的任务，以确保每一个任务都能被运行到。

具体到抢占时机有两种，一种是基于时钟的中断。等中断来了再抢占。就好比一个班里的学生在上自习，下一节课从体育课变成了数学课，那也要从下节课的上课铃开始才能上数学课。而另一种是立即抢占，就像是突发新闻，不管你看哪个频道，全部一瞬间变成同一个新闻，达到高效的快速响应。



另一种是时间片轮转调度算法。操作系统在时间的维度上对CPU进行了划分，即时间片，每一个时间片运行一个任务，时间片到期了之后运行下一个任务，如此反复。当一个进程结束后就退出。比如图中的例子，一开始任务1在第一个时间片运行，时间片结束后，任务1被挂起，任务2开始占用第二个时间片，第二个时间片结束后任务1再次占用当前CPU，运行了一会以后，任务1已经圆满完成，退出了。这个时候提前结束当前时间片，换任务2执行，如果没有别的等待的任务，任务2就会继续执行。以此类推。这个就是时间片轮转调度算法。

这个算法的好处显而易见，各个任务都能快速并均匀的分配到CPU资源，当CPU主频足够高，时间片足够短，那么当系统运行起来，用户的感受就是任务1和任务2是在同时进行的，即实时多任务处理。就如同我们的日光灯，在220v电压下其实是在50Hz的频率下频闪的，但由于闪烁的太快，人的眼睛根本感受不出来，所以就认为灯没有闪，一个道理。

时间片时间长短的分配是一门学问，时间片太长，系统的响应就会减慢，就像如果日光灯闪烁频率是2Hz，一秒钟闪两次，没有人会认为灯不在闪。反应到操作系统上，可能就是卡顿或者延迟。如果把时间片设置的太短，就意味着系统需要频繁的进行任务切换，要知道切换任务也是需要消耗资源的，相应的就降低了CPU的效率。所以优先级抢占和时间片轮转算法都不完美，都有各自的优缺点。



所以发展出了结合两种算法的算法。假设我们有六个任务，前三个是高级别的，后三个是低级别的。那么这时我就把它们分为了两组，低优先级的先给挂起，不去管它，运行高优先级的。那么高优先级队列里面的任务，大家都一样重要，谁先来呢？这里就结合了时间片轮转。大家按顺序一个一个来，三个任务都完成了，再去运行4、5、6任务，也是按照时间片轮训的。低级别任务在运行的时候，也是可以插入高级别任务的，旧的低级别任务组就会整体的停止，指导插入的高优先级任务组完成之后再继续。



**略**