讨论几个要点

设计调度算法时要考虑以下几个问题：

- 进程控制块PCB中

  需要记录哪些与CPU调度有关的信息

- 进程优先级及就绪队列的组织
- 抢占式调度与非抢占式调度
- I/O密集型与CPU密集型进程
- 时间片

之后呢，我们要 讨论以下几个问题。

1.进程优先级(数)

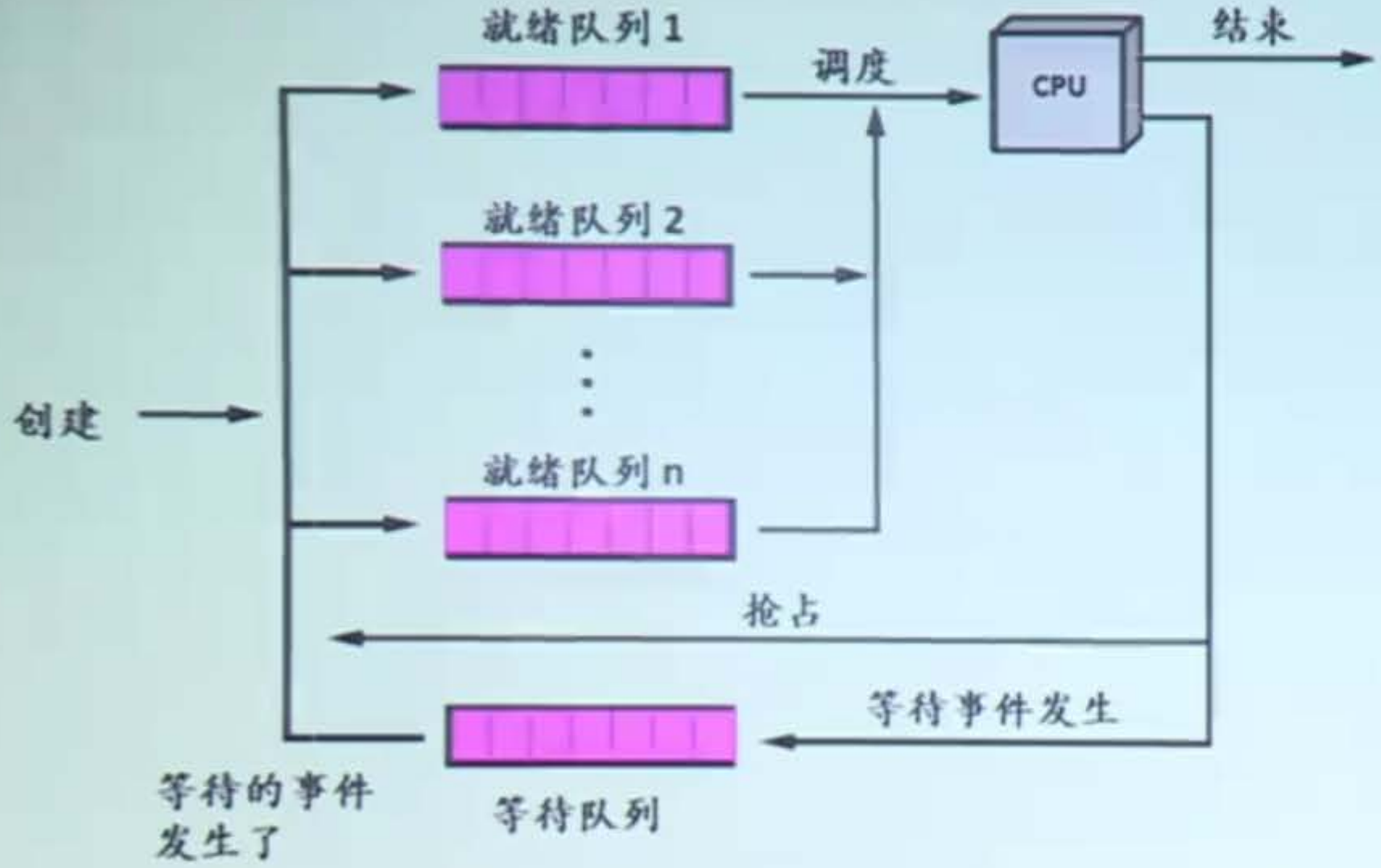
优先级 与 优先数 ？ 静态 与 动态

静态优先级：
进程创建时指定，运行过程中不再改变

动态优先级：
进程创建时指定了一个优先级，运行过程中可以动态变化
如：等待时间较长的进程可提升其优先级

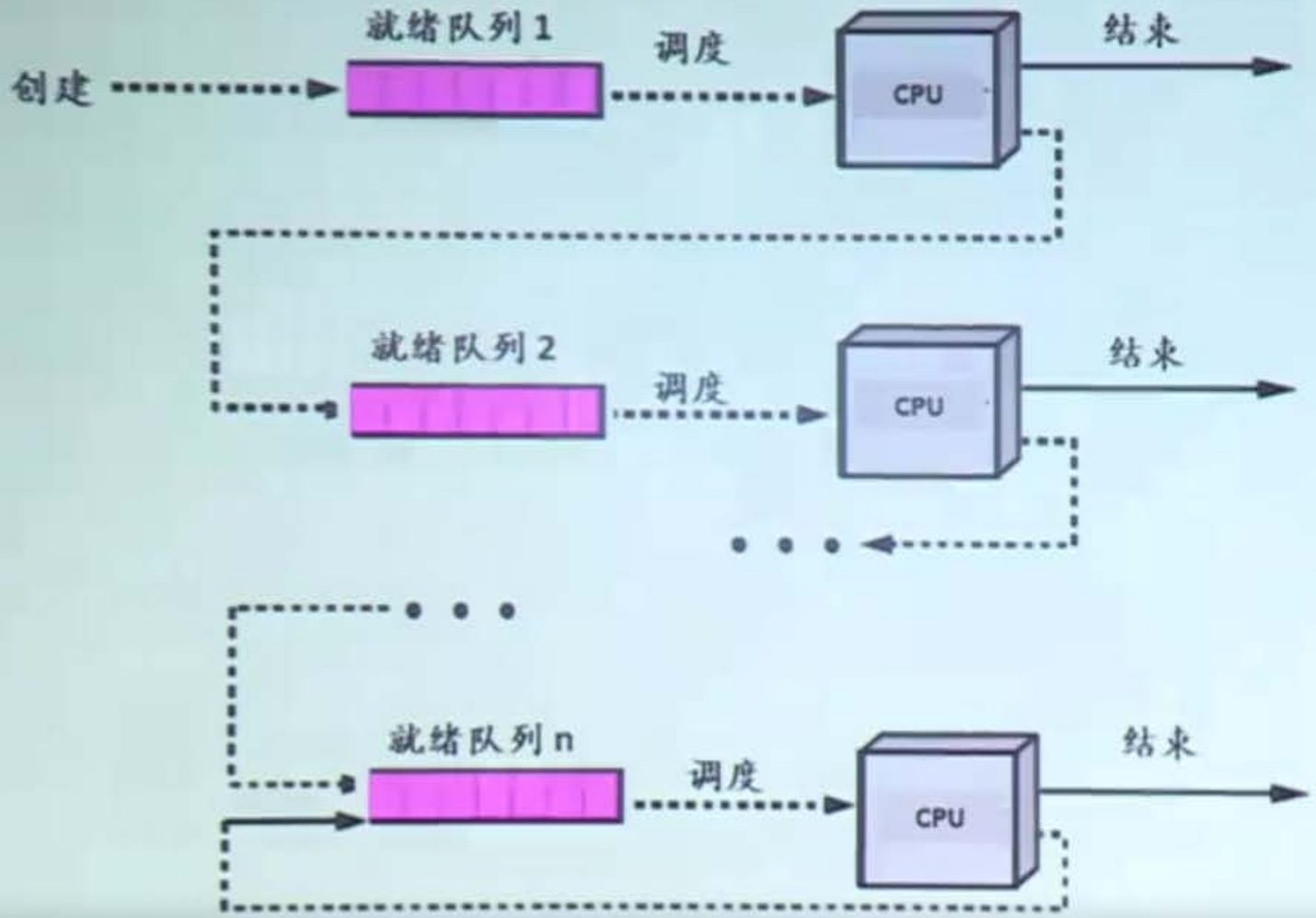
那我们要提升它的优先级，让它尽快有机会得到 CPU

2.进程就绪队列组织1



按优先级排队

2.进程就绪队列组织2



另一种排队方式

3.抢占与非抢占

指占用CPU的方式：

- 可抢占式Preemptive（可剥夺式）

当有比正在运行的进程优先级更高的进程就绪时，系统可强行剥夺正在运行进程的CPU，提供给具有更高优先级的进程使用

- 不可抢占式Non-preemptive（不可剥夺式）

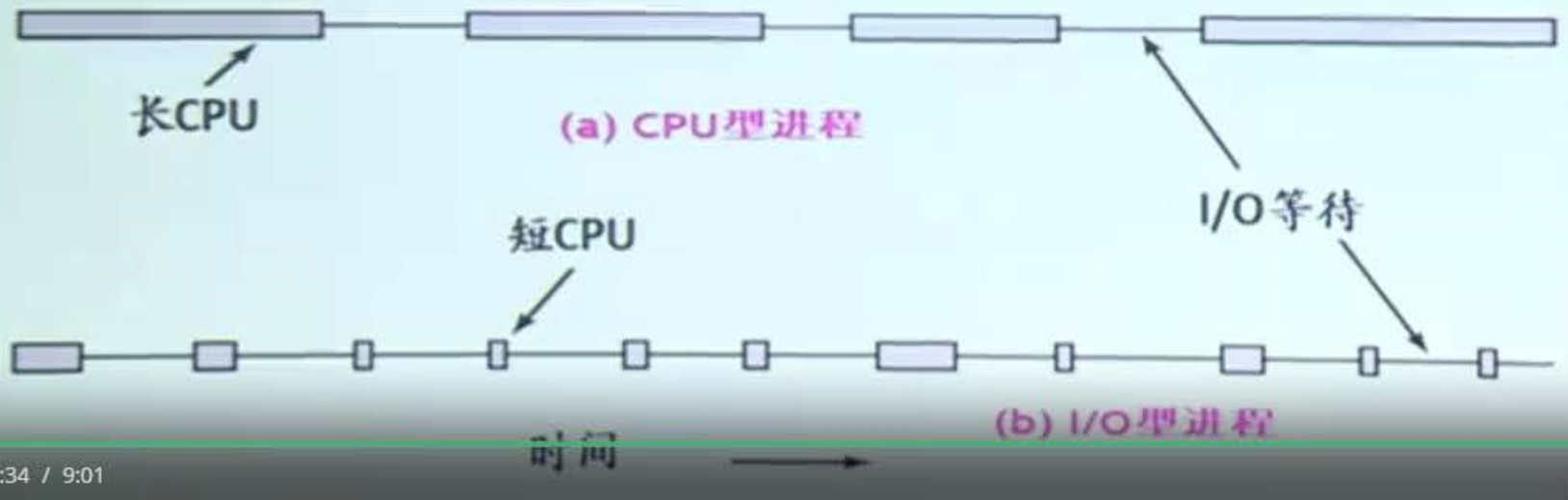
某一进程被调度运行后，除非由于它自身的原因不能运行，否则一直运行下去

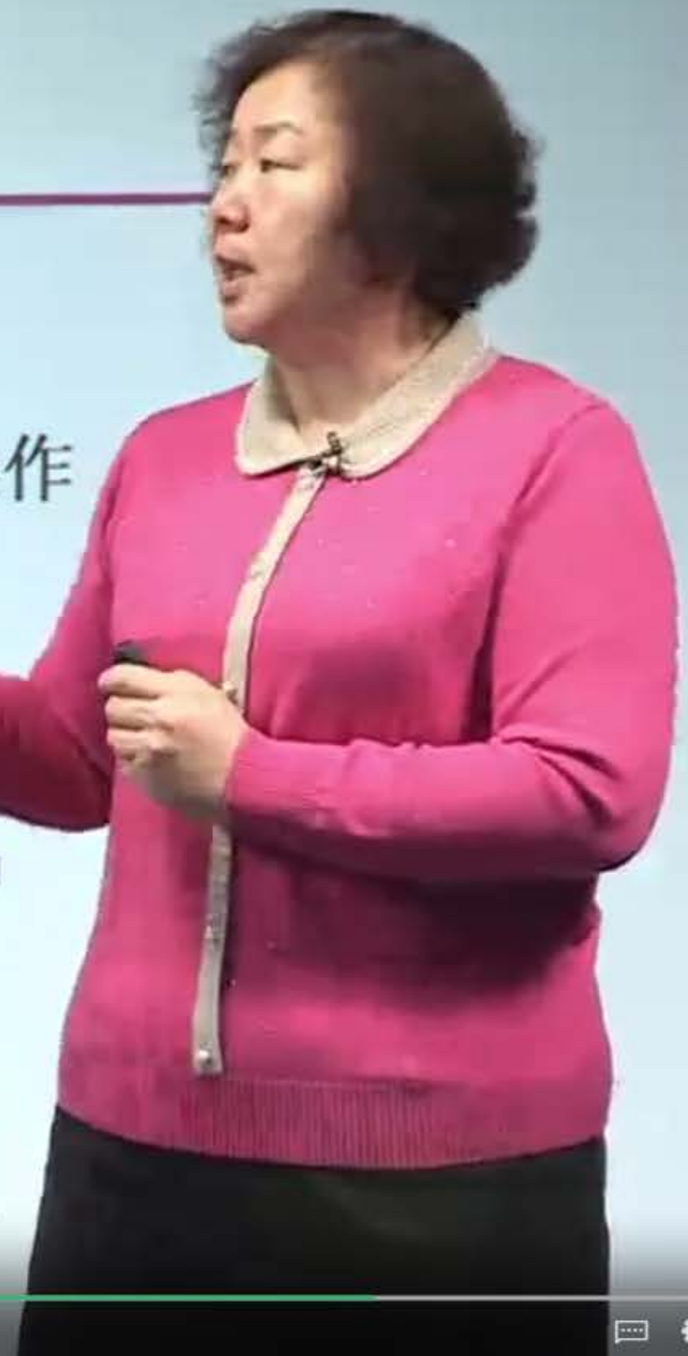
CPU 之后才进入就绪队列的 这就是抢占和不可抢占

4.I/O密集型与CPU密集型进程

按进程执行过程中的行为划分：

- I/O密集型或I/O型(I/O-bound)
  - 频繁的进行I/O，通常会花费很多时间等待I/O操作的完成
- CPU密集型或CPU型或计算密集型(CPU-bou
  - 需要大量的CPU时间进行计算



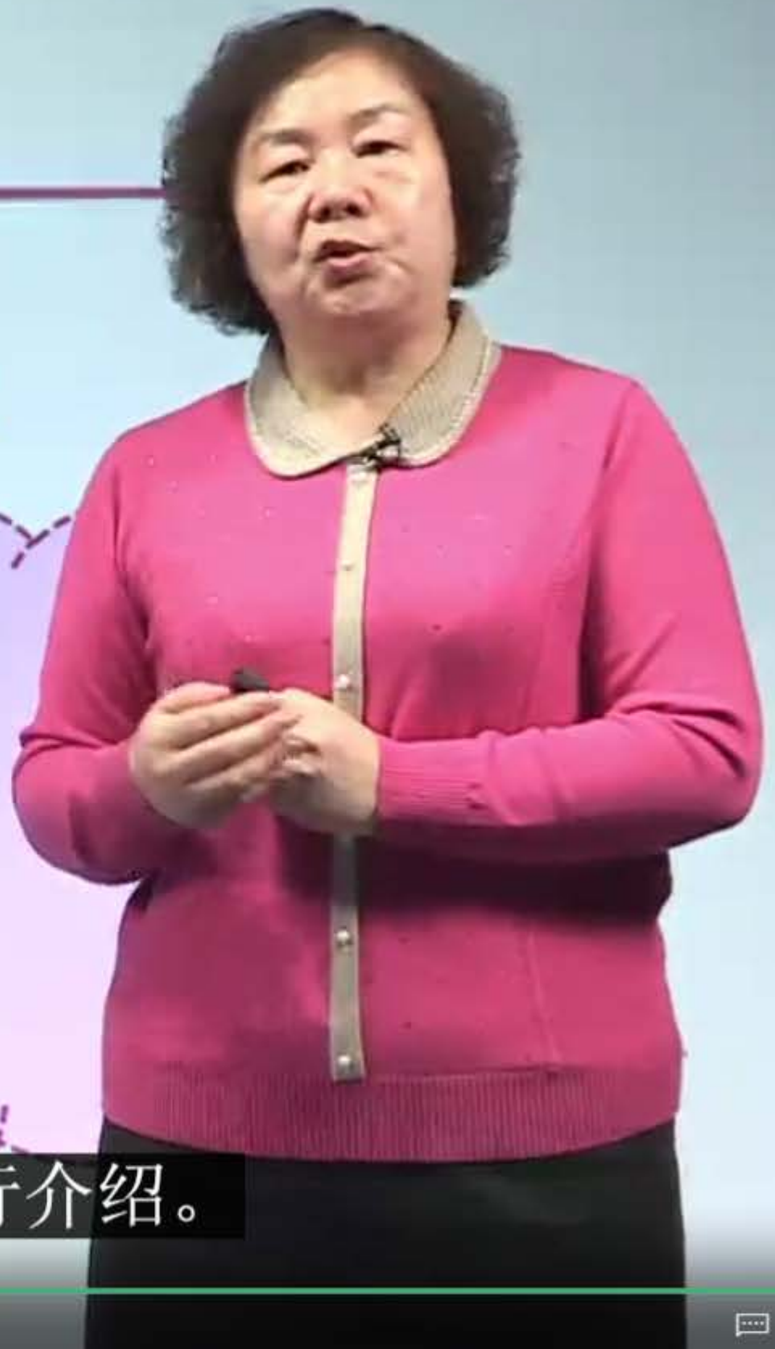


5.时间片

时间片长一点or短一点 ？ 固定与可变

- Time slice 或 quantum
- 一个时间段，分配给调度上CPU的进程，确定了允许该进程运行的时间长度
- 如何选择时间片呢？

考虑因素：
- ✓ 进程切换的开销
- ✓ 对响应时间的要求
- ✓ 就绪进程个数
- ✓ CPU能力
- ✓ 进程的行为



时间片轮转算法的时候呢会进行介绍。

0:00 下面我们来讨论在设计一个调度算法的时候要考虑的几个问题 首先我们应该考虑数据结构 进程控制块当中需要记录哪些 跟调度相关的信息，在设计 PCB 的时候，我们要为调度设计相应的字段 之后呢，我们要 讨论以下几个问题。 我们首先来介绍 进程的优先级和优先数 优先级和优先数呢 是两个不同的概念，要区分 优先级呢表现出了进程的重要性 和紧迫性。 那么优先数呢实际上呢是一个数值 它反映了某一个优先级。 有些系统像 UNIX 优先数小的优先级高，这就像日常生活中一样 围棋，那么 9 段比 1 段水平要高 一般一般的运动员 一级运动员要比三级运动员水平要高 所以不能完全根据数的大小来决定优先级的高低 确定了优先级之后，那么优先级是固定不变的呢 还是可以改变，这就是静态优先级还是动态优先级的概念 所谓静态优先级呢指的是在进程创建的时候，指定的优先级之后 在进程运行的过程中，这个优先级就不再改变了 而动态优先级呢往往是在进程运行过程中，优先级的 级别还会不断地调整。 有的时候呢我们需要动态优先级，比如说 当一个进程在就绪队列当中等待的时间很长的时候 那我们要提升它的优先级，让它尽快有机会得到 CPU 去运行 有了优先级以后 那么就绪队列呢可以重新来组织了 我们可以按照优先级来组织就绪队列 当创建一个新的进程的时候 那么根据它的优先级排不同的 就绪队列，在这里头我们设定了 n 个就绪队列 优先级，不同的优先级进入不同的就绪队列 那么就绪队列 1 的优先级是最高的 当调度程序选择进程的时候呢 首先应该从高优先级的队列来选择进程 如果高优先级队列没有进程了，再从次高就绪队列来选择进程 好，这是按照优先级来排队 也可以另外一种排队的方式 我们来看一下，所有的进程 第一次创建之后，都进到第一级就绪队列 那么随着进程的运行 我们可能会降低某些进程的优先级 比如说当一个进程分配给它的时间片用完了，那么它就会 降一级，降到第二级就绪队列。 如果它 经常地用完时间片，慢慢慢慢这个进程就会降低到最后一级就绪队列 进程调度首先在第一级队列里选 那就说明第一级就绪队列的优先级比较高 如果这个队列是空的，那么调度就会从 其他的就绪队列里头选择进程 那么如果你的进程已经进到了最后一级队列，那么它的优先级就越来越低 它被调度上 CPU 的机会呢就会变得很小 这就是这样一种排队以及一些结果 下面呢我们来讨论一下 占用 CPU 的方式。 通常有两种方式 一种叫做抢占式，一种叫做非抢占式 所谓抢占式呢指的是这样一种情况 当有比正在运行进程优先级更高的进程就绪的时候 系统可以强行剥夺正在运行进程的 CPU 然后呢把它提供给具有更高优先级的进程 这就是抢占。 所以在 CPU 上运行的 永远是优先级最高的进程。 所谓不可抢占呢 指的是当某一个进程被调度

运行之后，除非由于它自身的原因放弃了 CPU，让出了 CPU 否则这个进程一直运行下去，那么这就是不可抢占。 因此在某种情况下，正在运行的进程可能 比某个在就绪队列中的进程的优先级略微低一点 是因为那个进程呢是在这个进程上 CPU 之后才进入就绪队列的 这就是抢占和不可抢占 另外 两个概念呢，一个是 I/O 密集型 一个是 CPU 密集型进程，这个在我们介绍 进程分类的时候呢我们已经介绍过了，它是根据进程的行为来划分的 所谓 I/O 密集型呢我们也简称 I/O 型 它是频繁进行 I/O 的进程，大部分的时间会去等 I/O 的结果 用 CPU 的时间比较少，我们来看这张图，那么 这个些线呢表示的是等待 I/O 的，是在 I/O 的这个情况。 那么 这些就表示占用 CPU，我们可以看到 I/O 型的进程占用 CPU 的时间都比较小 那么所谓 CPU 型，有的时候也称为 CPU 密集型，或者计算密集型，它往往指的是呢 需要大量的 CPU 时间来进行计算的进程，那么我们可以看到 这就是在 CPU 上的时间，需要很长的 CPU 的计算 那么在设计调度算法的时候呢，通常会对 I/O 型的进程呢会有一些友好的 表示，希望更多的 I/O 型进程早一点上 CPU 运行 因为这些进程上CPU 之后 只用了很短的一下 CPU 时间就会让出 CPU，因为它要去做其它的输入输出操作了 因此对于一般的这个 调度程序，都会对 I/O 型进程更偏好一些 下面我们来看一下时间片这个概念 这个概念呢大家不陌生，它呢指的是一个时间段 那么分配给调度上 CPU 的进程，允许这个 进程在 CPU 上执行多长时间，它是一个时间的长度 那么如何选择时间片呢？ 通常，我们应该考虑这么多的因素。 比如说 进程切换它的开销有多大， 进程对响应时间的一个要求。 那么还有呢 系统当中有多少进程处于就绪 CPU 的能力有多大，以及进程的行为 不同进程我可能区别对待。 因此 在设计时间片大小的时候呢，要考虑到这些因素 到底时间片长一点好呢，还是短一点好？ 每个进程分配的时间片都是固定的呢 还是可以变化的？那么这些问题呢我们在讨论 时间片轮转算法的时候呢会进行介绍。