
三状态模型、五状态模型、七状态模型
进程状态及状态转换
下面呢我们来介绍一下，进程状态及状态转换 我们首先，先来看一下最基本的
0:04 / 14:51

进程的三种基本状态

- 进程的三种基本状态：
  运行态、就绪态、等待态

- 运行态（Running）
  占有CPU，并在CPU上运行
- 就绪态（Ready）
  已经具备运行条件，但由于没有空闲CPU，而暂时不能运行
- 等待态（Waiting/Blocked）
  因等待某一事件而暂时不能运行

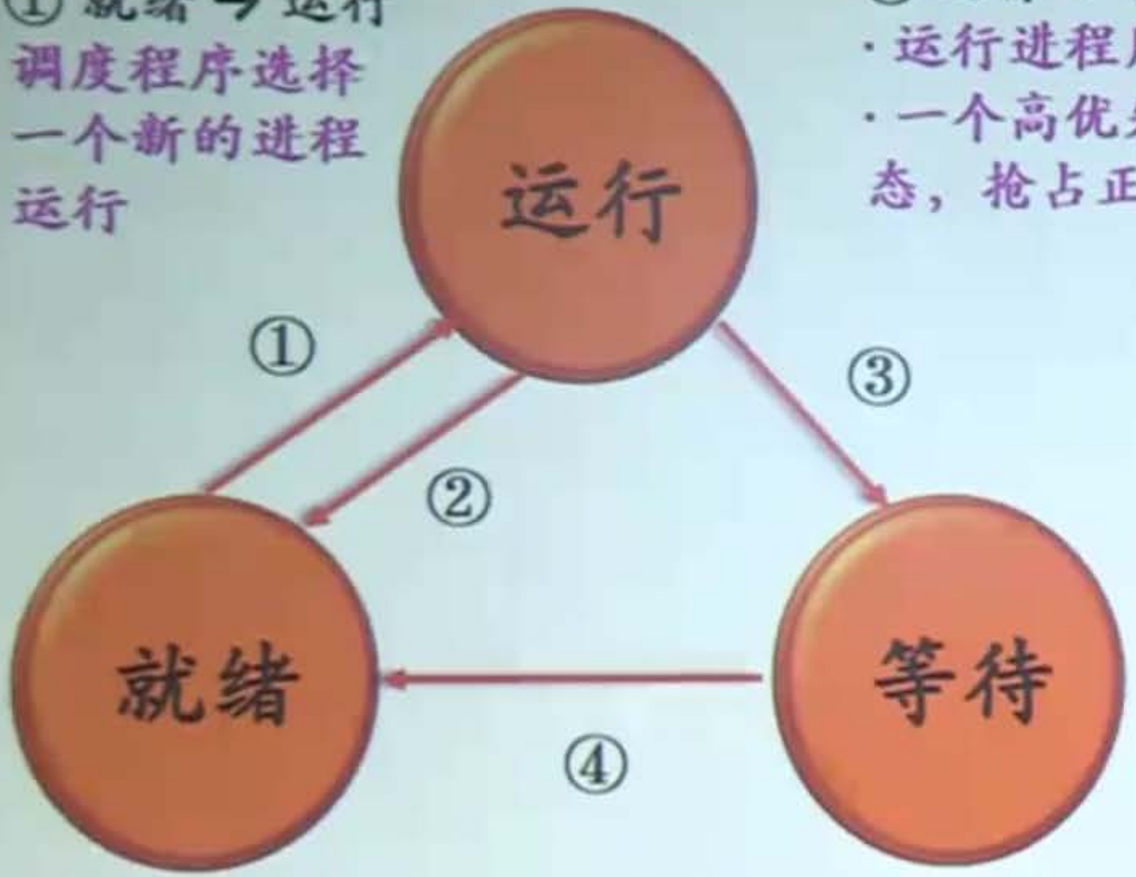
阻塞态、封锁态、睡眠态

如：等待读盘结果

三状态模型及状态转换

① 就绪 → 运行
调度程序选择一个新的进程运行

② 运行 → 就绪
· 运行进程用完了时间片
· 一个高优先级进程进入就绪状态，抢占正在运行的进程



③ 运行 → 等待
当一个进程等待某个事件发生时

· 请求OS服务
· 对资源的访问尚不能进行
· 等待I/O结果
· 等待另一进程提供信息
· ……

④ 等待 → 就绪
所等待的事件发生了

进程的其他状态

创建 new

- 已完成创建一进程所必要的工作
  - PID、PCB
- 但尚未同意执行该进程
  - 因为资源有限

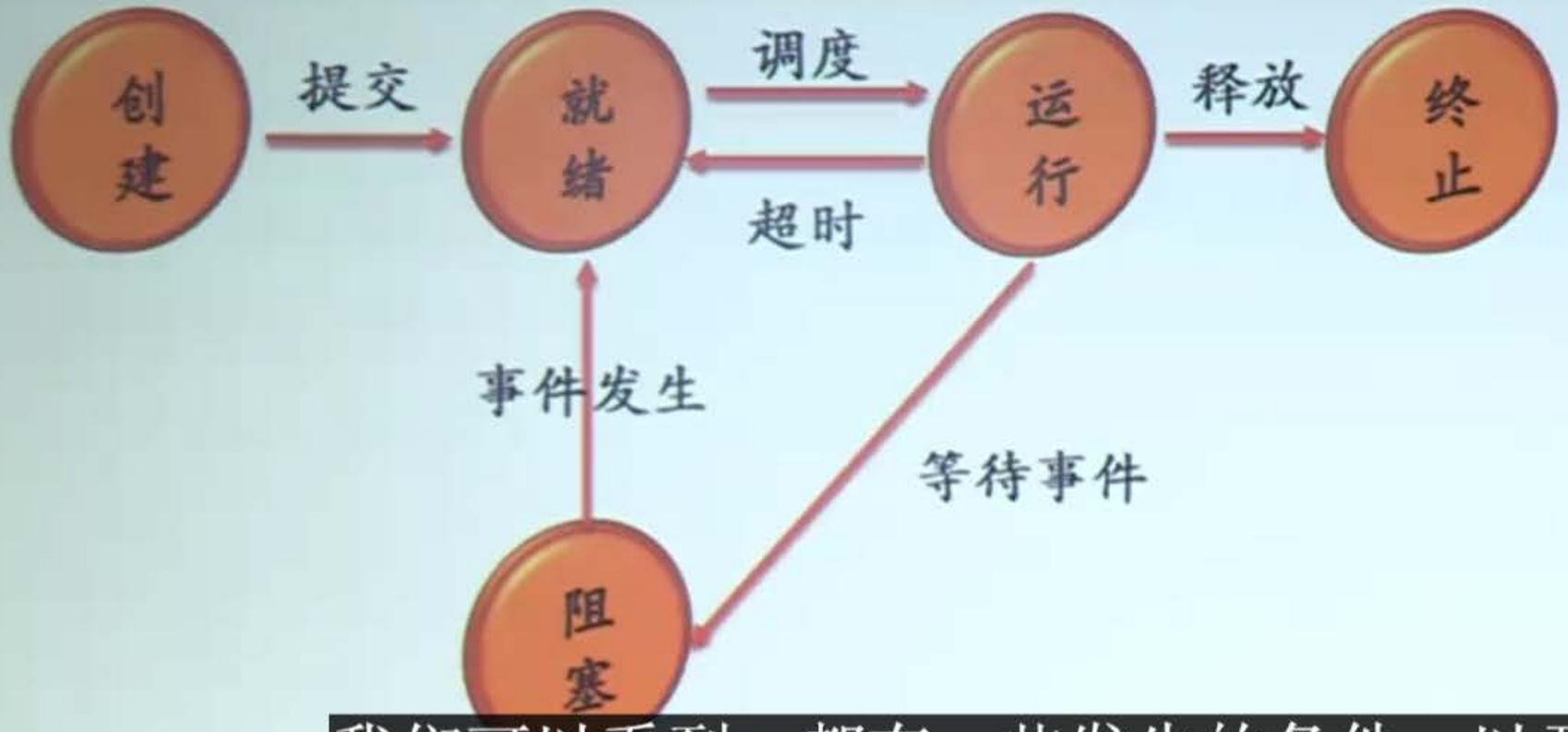
终止 Terminated

- 终止执行后，进程进入该状态
- 可完成一些数据统计工作
- 资源回收

挂起 suspend

- 用于调节负载
- 进程不占用内存空间，其进程映像交换到磁盘上

五状态进程模型

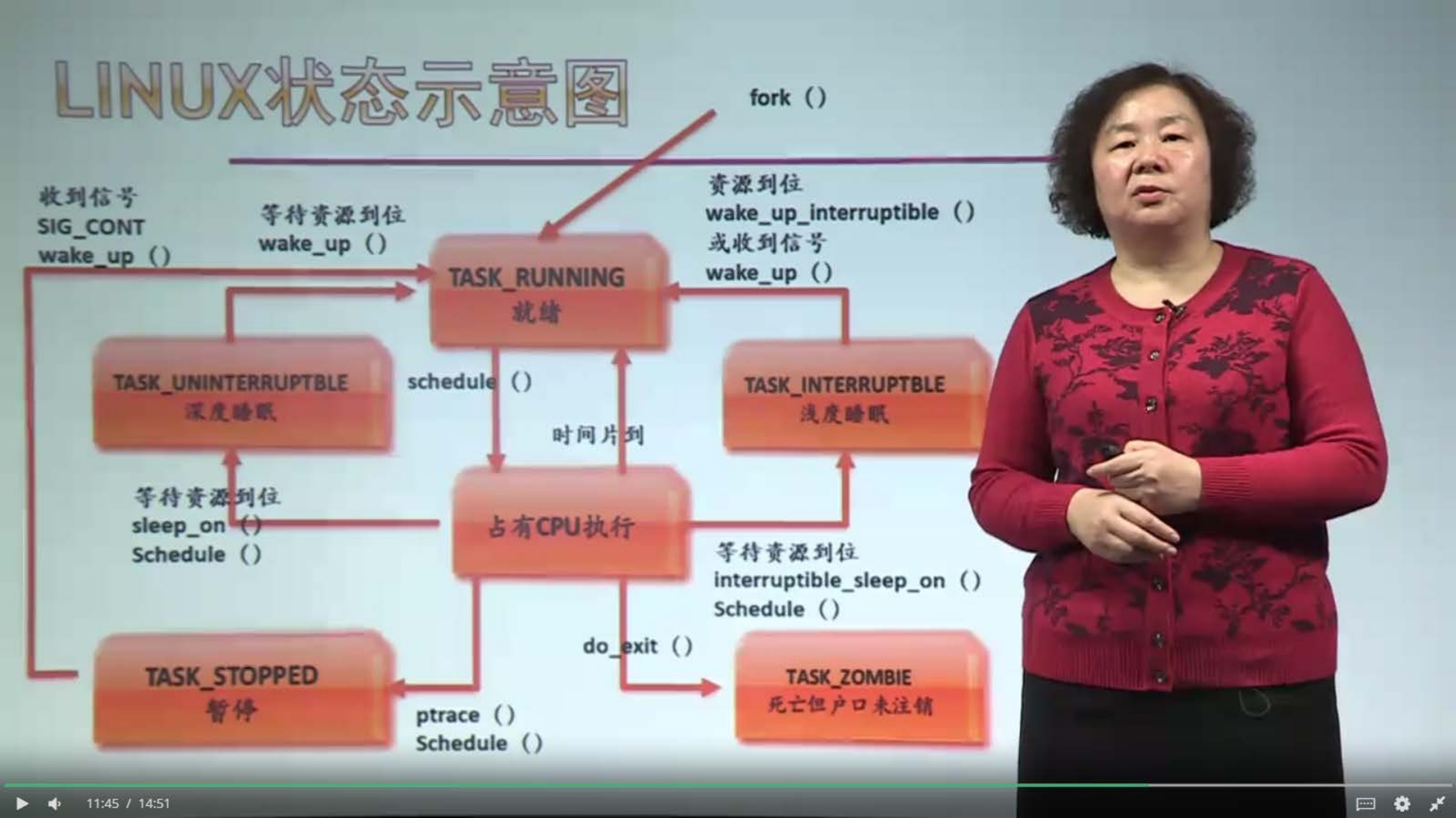


我们可以看到，都有一些发生的条件，以及相应的操作

七状态进程模型

就绪挂起的队列，那么这就是一个七状态的模型

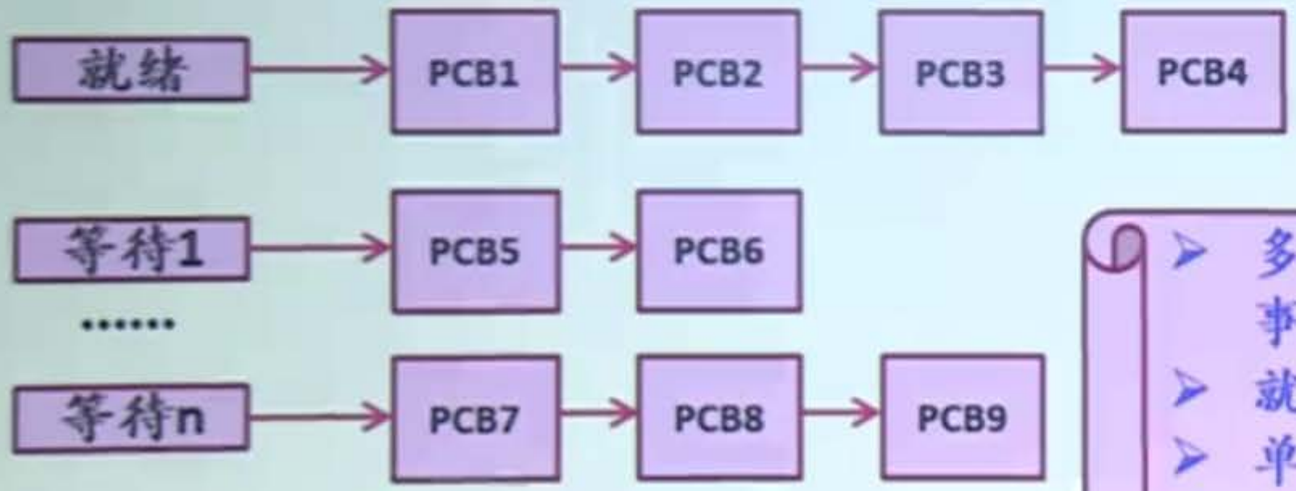
LINUX状态示意图



11:45 / 14:51

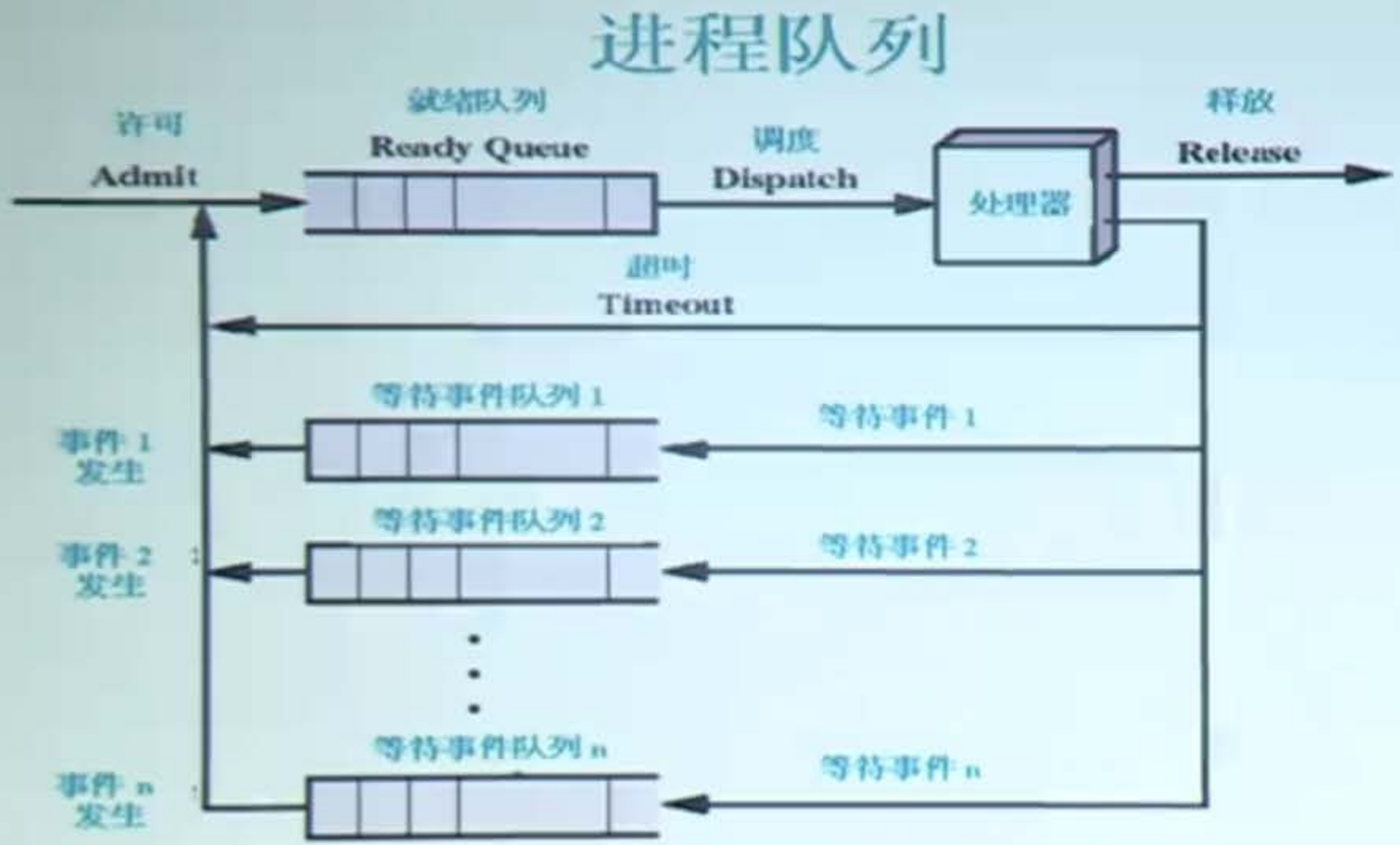
进程队列

- 操作系统为每一类进程建立一个或多个队列
- 队列元素为PCB
- 伴随进程状态的改变，其PCB从一个队列进入另一个队列



- 多个等待队列等待的事件不同
- 就绪队列也可以多个
- 单CPU情况下，运行队列中只有一个进程

五状态进程模型的队列模型



所以这是一个五状态进程模型当中的一个队列的模型

选自W. Stallings教材

0:00 下面呢我们来介绍一下，进程状态及状态转换 我们首先，先来看一下最基本的 状态一共有三个，运行态、 就绪态、 等待态 那么运行态指的是 这个进程已经拥有了CPU 并且在CPU上执行。 就绪态 指的是进程已经具备了运行的条件 但是由于没有空闲的CPU，所以这个进程暂时还不能运行 打个比方就是说，万事俱备 只欠CPU。 第三种状态呢 我们把它称之为等待态，那么指的是这个进程 因为等某一个事件，而暂时不能运行 那么什么样的事件可以使得 进程处于等待态呢？比如说你要去读盘 一个进程要去到磁盘读数据 在数据还没有读完之前，没有读到内存之前，这个进程 暂时是不能运行的，因为它要之后的工作是要处理数据 好，那么等待态还有其它的一些说法 这也是在学习操作系统的过程中，大家要关注的 一个名词会有多种说法，那么等待态呢，又称为阻塞态 或者封锁态，或者睡眠态 这是进程的三种基本状态 在进程运行过程中 由于进程自身的进展，或者是一些外界条件的变化 进程的状态呢，会有改变 也就是说，从某一个状态进入到另外一个状态 我们来看一下有哪些状态的转换 就绪态什么时候进入运行态呢？ 就绪态进入运行态，是因为这个进程被调度程序选中了 那么运行态什么时候 又回到了就绪态呢？一个正在运行的进程 在两种情况下可能会回到就绪态 一个是它运行完了，分配给它的时间片 所以它不能再继续运行了它回到就绪态，等待下一个时间片的分配，或者是 一个高优先级的进程，它进入了就绪态，因此 它会去抢占正在运行的这个进程 那么被抢占的这个进程呢，它也要回到就绪态 什么情况下，正在运行的进程会进入等待态呢？ 等待是有原因的 往往是当一个进程等待某个事件发生的时候 那么它暂时不能运行，而进入等待态，比如说它请求了一个操作系统的服务 啊，现在服务还没有结束，或者是它要用某个资源，这个资源不能使用 还不能被使用，或者是它等待一个I/O的结果 也可能是 等待另外一个进程，给它发来消息，总之它 拿不到这个事件，等不到这个事件，这个进程就不能执行 那么一旦事件发生了，那么 从等待态这个进程，就进入了什么呢？就绪态 继续准备上CPU，所以当事件发生的时候 那么它就从等待态，进入了就绪态 这是三种基本状态，以及 它们之间的状态转换。 这张图其实我们可以看到 没有从就绪态进入等待态的这样一个转换 也没有，从等待态直接进入运行态的这么一个转换 大家可以思考一下，为什么？除了 这三种基本状态之外呢，操作系统可能还有其它的一些 状态，那么这是状态的设计，就是操作系统在设计这个 进程模型的时候，必须要首先考虑的问题 我们现在来看一下，另外两种 典型的状态，一个是创建态，一个是终止态 所谓创建态呢 就是说已经完成了，创建一个进程的必要的工作 分配了PID，

填写了PCB 但是呢由于某种原因，这个进程 还不能够马上执行，操作系统没有许可它执行 那么原因当然可能是，由于现在系统资源有限，现在系统负载太大 你不要暂时，暂时你不要运行，那么终止态 呢，当然就是程序结束了，进程结束了 它进入了终止态，那么 在进入终止态之后呢，操作系统还要为这个进程做一些工作 一项工作呢，是做一些数据统计工作 你用了多少CPU，你用了多少资源，那么你要做一些记录 另外最重要的，就是要把你 所申请的，所拿到的资源要收回来 因为你已经不再运行了，这个进程已经不再运行了，所以它的所有资源要收回 那么这是两个，也是比较常见的状态 还有一个状态呢叫做挂起态 挂起态呢比较特殊，它往往指的是 在操作系统当中，如果想进行一些负载调节的时候 那么可能会把一个，一些进程，把它送入这个状态 那么如果现在系统中进程太多 那么CPU也忙不过来了，所以操作系统会把一部分进程，让它暂时不能运行 但是它又不是等待，因为不是等待某个事件发生，所以就把它弄成一个特殊的状态，叫挂起态 一旦进程进入了挂起态，操作系统会把它的内存空间呢收回来 把这些内容，把它的进程的相关的内容，把它送到磁盘上保存起来 那么一旦 还继续让它运行，我们通常称之为 激活，那么这个进程的内容，再从磁盘上读入内存就可以了 所以呢，我们在三种基本状态之外呢，还可以为 一个进程设计这么三种不同的状态，好，我们来看一下 五状态模型，也就是增加了创建态和终止态 创建完了进程之后，把它通过提交，进入了就绪态 通过调度就绪态进程，变成了 运行态，运行过程中可能因为超时了 等因素，那么从运行回到了就绪 如果运行的进程等待某个事件发生，那么就它变成了等待态 当等待的事件发生了呢，它就又回到了就绪态 如果进程运行过程中结束了，那么就进入了 终止态，这就是五状态模型的进程状态的转换 我们可以看到，都有一些发生的条件，以及相应的操作 那么引入了，挂起态之后呢，那么那就变成了一个七状态模型 我们来看一下七状态模型的，一个状态转换 [空白音频] 那么当系统的负载过重的时候，操作系统会把一部分 就绪的进程，让它进入就绪挂起 之后呢，可能在某个时候再把它激活 但是这部分 内容呢，这部分的这个进程呢，可能还不够多，所以呢操作系统还会把一些 处于阻塞态的进程把它 变成阻塞挂起，进一步减轻系统的负载。 当然了，如果 条件允许再次激活它，可能在激活之前，发生的事，等待的事件发生了，所以 那么阻塞挂起的这个进程，可以直接进入什么呀？就绪挂起 当然有的时候，也可能会出现 让一个正在运行的进程，当它正好下CPU的时候，把它直接送入到 就

绪挂起的队列，那么这就是一个七状态的模型 在一个实际 操作系统当中，可能涉及不同的进程状态 同时不同的进程状态之间的转换，也有不同的条件和不同的操作 我们简单地看一下Linux的状态及状态转换 Fork是创建了一个进程，变成就绪 通过调度，上了CPU。 可能等某个 条件，某个事件进入了睡眠态 当这个事件发生了，就又回到了就绪 如果正在运行的进程时间片到了 就重新进入就绪队列 可能在运行过程中还会等其他的条件，就进入了另一种睡眠 那大家可以看到，有两种睡眠状态 那么，它的区别呢主要是说，浅度睡眠在睡眠过程中会接收信号 这是一种特定的通信机制 而深度睡眠的进程是不接收信号的 这就是它们的区别，然后 正在运行的进程呢，可能因为调试，设定了断点 所以它处于了一个叫作暂停状态，这也是跟调试有关系 那么暂停状态，然后以后接着执行，又进入就绪 正在执行的进程如果结束了，我们说终止了 那么就进入了一个，我们说终止态，当然linux里头，把终止态称之为 僵尸态，僵尸态，其实就是进程终止了 那么，这是linux的状态的示意图。 所以我们可以看到 任何一个操作系统，它在设计 进程模型的时候呢，要确定有什么样的状态 确定状态之间的转换，在什么条件下转换 通过什么样的操作来促成这种转换 那么操作系统当中 有很多的进程，它们都处于不同的状态 所以呢，需要按不同的状态把它们管理起来 因此，操作系统设计了一个若干个进程队列 为每一个类进程呢，建立一个或者多个队列也是可以的 每个队列的元素呢，实际上就是PCB 状态的改变，其实就是 某个进程的PCB，从一个队列出队 然后，对在另一个队列里头入队的过程 也就是伴随着状态的改变，它的PCB 从一个队列进入到另外一个队列。 我们来看一下 这就是一个队列的 描述。 那么这里头进程 如果处于就绪态，那么就排一个队 那么就绪态通常 我们看到是一个队列，其实在实际系统中，就绪态的 这个进程可以排多个不同的队列 那么等待态有不同的原因 所以呢，不同的原因等在不同的这个队列里头 根据等的事件不同，来等在不同的队列里头 所以我们可以看到，就是多个等待队列 实际上呢，就是表示等待的事件不同。 就绪队列呢 也可以是多个，我们这里虽然画了一个，也可以是多个 那么在单CPU的情况下，由于 运行的进程只能有一个，所以我们就没有画出运行的队列 这里给出了一个五状态进程模型的队列模型 我们可以看到，有一个就绪队列 那么创建好的进程通过许可进入就绪队列 被调度程序选中上CPU，运行完了以后呢退出 如果运行的时间片到了，超时了，它就继续回到就绪队列 那么运行过程中如果等待某个事件，就进入到相应的等待事

件的队列 当这个事件发生呢，就又回到了就绪队列 所以这是一个五状态进程模型当中的一个队列的模型