本讲内容

- 死锁的基本概念
- 资源分配图
- 死锁预防
- 死锁避免
- 死锁检测与解除
- 哲学家就餐问题

最后呢我们以哲学家就餐问题为一个实例，来介绍对于死锁这样一个主题

死锁的基本概念

首先我们来介绍一下死锁的基本概念

死锁的基本概念

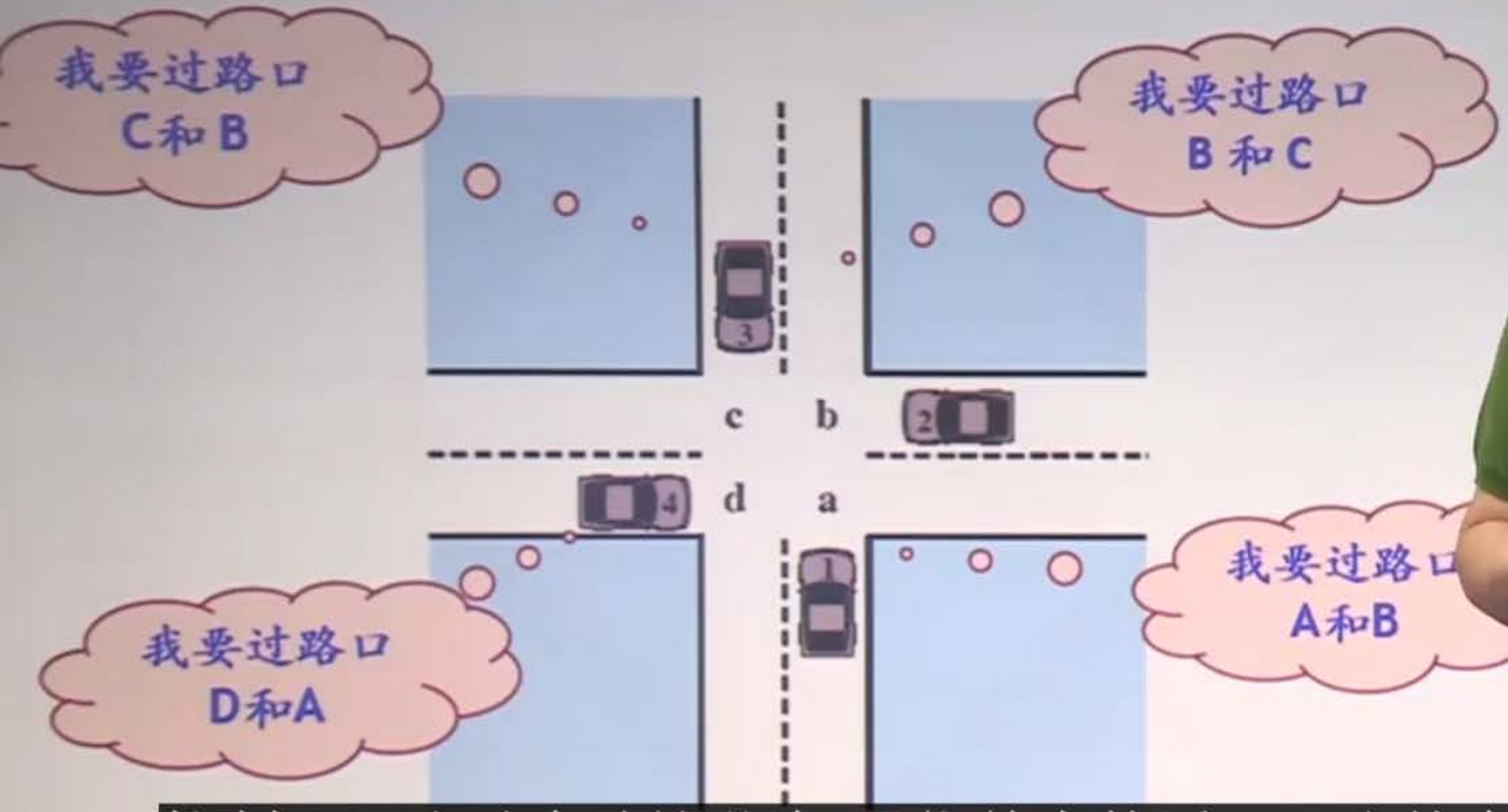
死锁的定义

- 一组进程中，每个进程都无限等待被该组进程中另一进程所占有的资源，因而永远无法得到的资源，这种现象称为进程死锁，这一组进程就称为死锁进程

- 如果死锁发生，会浪费大量系统资源，甚至导致系统崩溃

> ➢ 参与死锁的所有进程都在等待资源
>
> ➢ 参与死锁的进程是当前系统中所有进程的子集

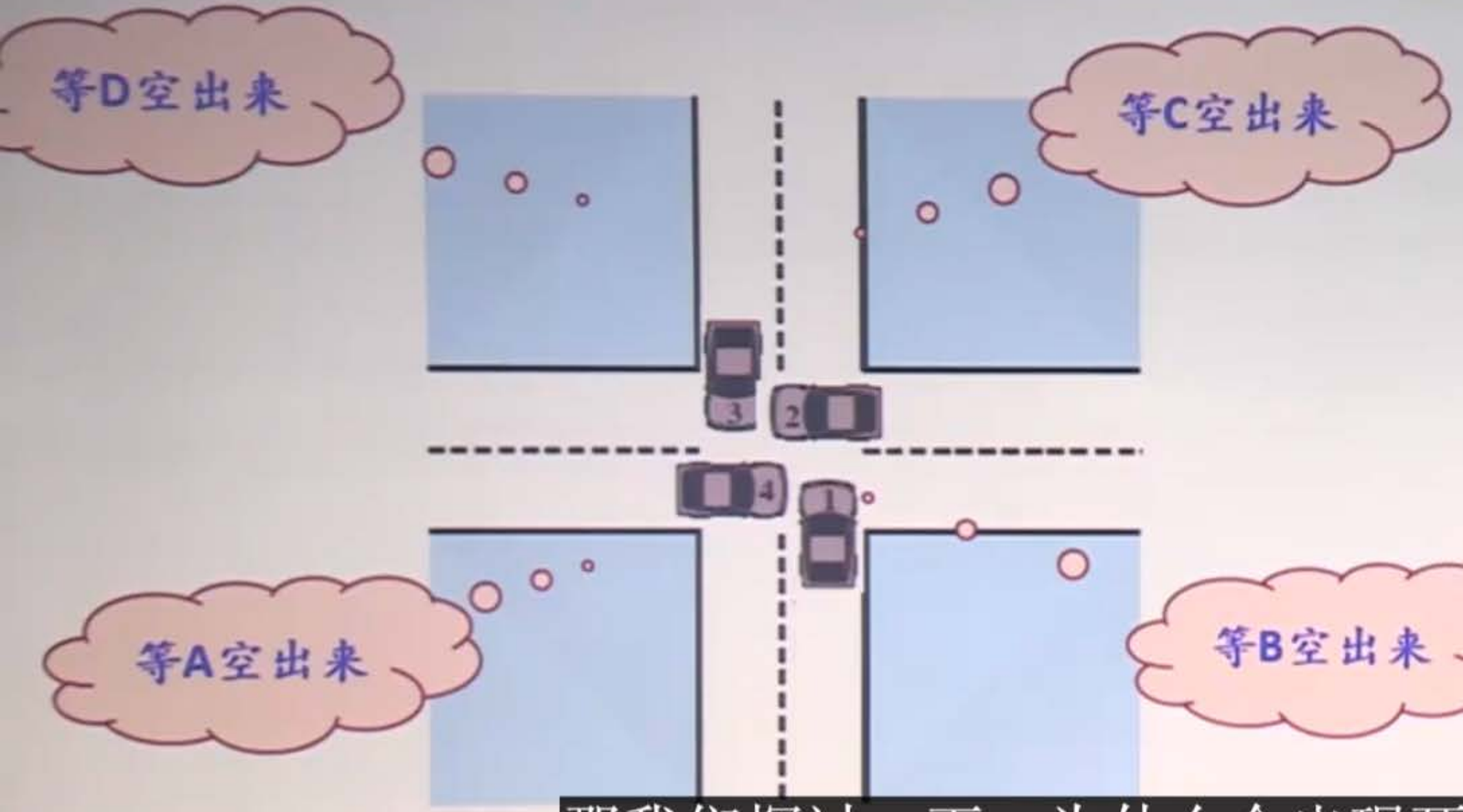
死锁的现象



控制，那么随意让这些车啊 往前走的话，那么就会出现这样一种现象

William Stallings

死锁的现象



那我们探讨一下，为什么会出现死锁？

William Stallings

为什么会出现死锁？

资源数量有限、锁和信号量错误使用

资源的使用方式：

“申请--分配--使用--释放”模式

可重用资源：可被多个进程多次使用

- 可抢占资源与不可抢占资源
- 处理器、I/O部件、内存、文件、数据库、信号量

可消耗资源：只可使用一次、可创建和销毁的资源

- 信号、中断、消息

进程竞争可重用资源

D：磁盘文件
T：磁带设备

Process P

Step	Action
P_0	Request (D)
P_1	Lock (D)
P_2	Request (T)
P_3	Lock (T)
P_4	Perform function
P_5	Unlock (D)
P_6	Unlock (T)

Process Q

Step	Action
q_0	Request (T)
q_1	Lock (T)
q_2	Request (D)
q_3	Lock (D)
q_4	Perform function
q_5	Unlock (T)
q_6	Unlock (D)

可分配内
200KB

P1

. . .
Request 80 Kbytes;
. . .
Request 60 Kbytes;

P2

. . .
Request 70 Kbytes;
. . .
Request 80 Kbytes;

William Stallings

进程竞争可消耗资源

P1	P2
...	...
Receive (P2);	Receive (P1);
...	...
Send (P2, M1);	Send (P1, M2);

P2 也在等待 于是它们俩就互相等待，都等对方给自己发消息，就产生了死锁现象

William Stallings

12:59 / 20:52

活锁和饥饿

应用Peterson算法

活锁

- 先加锁
- 再轮询

→ 既无进展也没有阻塞

```
void process_A(void) {
        enter_region(&resource_1);
        enter_region(&resource_2);
        use_both_resources( );
        leave_region(&resource_2);
        leave_region(&resource_1);
}
```

饥饿

资源分配策略决定

```
void process_B(void) {
        enter_region(&resource_2
        enter_region(&resource_1
        use_both_resources( );
        leave_region(&resource_1);
        leave_region(&resource_2);
}
```

所以死锁呢要和活锁、 饥饿，啊 这样两种其它不同的现象呢，要加以区分

产生死锁的必要条件

- 互斥使用(资源独占)

  一个资源每次只能给一个进程使用

- 占有且等待(请求和保持，部分分配)

  进程在申请新的资源的同时保持对原有资源的占有

- 不可抢占(不可剥夺)

  资源申请者不能强行的从资源占有者手中夺取资源，资源只能由占有者自愿释放

- 循环等待

  存在一个进程等待队列 $\{P_1, P_2, \ldots, Pn\}$，

  其中P_1等待P_2占有的资源，P_2等待P_3占有的资源，...，Pn等待P_1占有的资源，形成一个进程等待环路

0:00 大家好！今天我给大家带来的是操作系统原理课的第十二讲 死锁。 本讲的内容呢包括对死锁的基本概念进行介绍 呃，资源分配图来表示死锁的一个 状态。 死锁预防、 死锁避免、 死锁 检测与解除，这三个是关于对死锁问题的解决方案 最后呢我们以哲学家就餐问题为一个实例，来介绍对于死锁这样一个主题 如何从各个方面来解决这样一些问题 首先我们来介绍一下死锁的基本概念 这里我们给出了死锁的定义。 什么是死锁呢？ 死锁呢是在一组进程当中 其中每一个进程都无限等待 被该组中另一个进程所占有的资源 那么它等待 另外一个进程占有资源而另外一个进程也在等待，因此 这个进程永远无法得到它所需要的资源 当出现了这种现象，我们就称为进程死锁 那么这一组进程呢就称为死锁进程 为什么我们要研究这个问题呢？ 因为如果发生了死锁 那么系统中的资源就会大量的浪费 随着资源的不断地消耗 最后，可能会导致系统的崩溃 所以我们要研究死锁的问题 从死锁的定义当中呢，我们可以得到几个推论 第一个呢是参与死锁的所有进程都在等待资源 那么从这个进程状态上说呢，这个进程是处于等待状态 阻塞状态。 第二个推论呢是指的是说 参与死锁的进程是当前系统中所有进程的一个子集 也就是它是部分进程，首先发生的现象 当然如果这些进程不断地有新的进程 进入死锁状态的话，那么这个子集就扩大扩大，那么会导致系统的崩溃 我们来看一下日常生活中的一个死锁的 例子。 那么这是一个典型的十字路口 我们知道十字路口里头有四个方向 在十字路口中间呢，我们划分了 A、 B、 C、 D 四个区域 现在呢有四个方向上的，同时有车呢到达了 我们现在的所示的位置，那么我们来看一下从 这个方向要通过这个路口的话呢，它可能是需要 B 和 C 这样一个区域，那它说我要经过 B 和 C。 那么从 南，比如向北的方向 那么就要需要 A 和 B 这两个区域 那么当出现了这样一个场景的时候我们来看一下，如果没有交通 控制，那么随意让这些车啊 往前走的话，那么就会出现这样一种现象 也就是换句话说，每一个路口的 位置，被一辆车占住了。 那么比如说从南 向北的车占住了 A 这个位置 但是它又需要等的 B 这个位置空出来 同样的道理，从东向西的车也是要等 C 来空出来，因为它占有了 B 这个位置 那我们看看到了这样一种情况，那么四辆车都不能往前走了 那么大家都在等，等那个资源让出来。 那么我们说这种现象 就是在日常生活当中的一个死锁 那我们探讨一下，为什么会出现死锁？ 死锁这个现象，在系统运行过程中出现了 主要有这样几个内在的原因。 第一个呢是 整体系统当中的资源的数量是有限的 比如说像刚才那样一个路口 那么不管哪辆车它都需要两个资源，就是需要两个路口 那么整体就一共就四个路口，那么四辆车呢 每个需要两个，那么它

们对资源的这个总体的要求呢，实际上呢是由于 资源的数量有限，所以导致了死锁的产生。 当然还可能由于 使用了锁呀，信号量这样的。 由于它们错误的使用，也可能导致死锁的出现 那么现在我们着重以资源 为一个对象来探讨死锁的问题 至于说锁和信号量 那么错误的使用呢，那么应该把它交给这个 开发人员来进行相应的这个处理 好，那么我们来看一下，资源的使用方式 在我们的第一讲当中，我们已经 给出了在操作系统当中的资源的一个使用的一般的模式 这个模式就是进程提出申请，操作系统呢 进行相应的分配。 在分配的过程中 如果资源数量有限，暂时进程所需要的资源 不能满足的话，那么这个进程就进入了阻塞或等待状态 当然如果这个资源可以满足这个进程 那么下一步，操作系统进行分配，然后进程去使用资源 当进程使用完这个资源之后呢 它把这个资源要归还给系统，释放资源。 因为我们讲的是 在我们现在的操作系统当中，所有资源呢都是一个动态分配的这样一个过程 那么由于资源的使用方式是这样一个方式，所以 有些进程得不到资源，它就会处于等待阻塞状态 那么，如果资源数量有限的话呢，那么就可能会出现死锁 那么我们继续来探讨一下，对于资源的分类 从研究死锁的角度，我们把这个资源分成了两类 一类呢，叫做可重用资源。 它指的是呢可被 多个进程，多次使用 对可重用资源呢，我们又把它进一步划分成 可抢占的与不可抢占的 比如说，在我们所列的这样一些资源当中 像 CPU 处理器就是一个典型的可抢占资源 那么我们前面在讲进程调度的时候呢，我们用一种抢占式的调度策略 实际上就是，当一个进程在 CPU 上运行的过程中，允许 比它优先级更高的进程的来抢占这个进程的 CPU 那么让这个进程暂时呢回到 就绪，然后这个，高优先级的进程呢上 CPU 运行 这就是可抢占。 那么还有一些资源呢，是不可抢占的。 比如说 打印机，打印机这种资源，当一个进程使用打印机的时候 不允许另外一个进程从这个进程 手中把打印机夺过来自己用，所以呢我们的资源呢，分成了可 抢占的和不可抢占的两类。 针对这个可 重用资源呢，另外一种类型呢，我们称之为可消耗资源 那么可消耗资源呢，它是只使用一次 而且它是可以创建，可以消 毁的这样的资源。 所以它创建的时候呢使用一次，然后呢可能就销毁掉了 那么这种资源呢，我们称之为可消耗资源 当然可消耗资源呢，在一个系统当中，比如说有信号，比如说中断或者是发来的消息 这些呢都是可消耗资源，比如说一旦 一个中断来了，如果我们没有及时处理，那么这个中断下一次被别的中断给覆盖掉之后，这个- 中断其实就丢失了 所以这是属于可消耗资源。 那么在我们这

个 死锁的主题当中，我们所要针对的呢，主要是在可重用资源上 如果对这类资源产生了死锁现象，我们来怎么处理？啊，至于说可消耗资源呢不在我们这里探讨了 那么下面我们来举两个例子。 一个是 进程之间呢竞争一个可重用资源 比如说我们有两类资源，有一个磁盘文件 有一个磁带，那么有两个进程 这个每个进程呢，都要使用这两个资源。 那么我们来看一下比如说以进程 Q 那么它呢可能先要申请一个磁带，磁带机 然后它要使用的时候呢，它要把这磁带机先上锁。 这样 的话呢，它可以以后就是用起来的时候，不受别的进程的干扰 然后它又去申请一个磁盘文件。 那么它可能要把磁盘文件读到磁带上，或者是把 磁带上的文件读到磁盘上，所以呢有一个申请一个磁盘 那然后呢，也要把它上锁，然后执行相应的，比如说，拷贝数据的操作 做完这些事情之后呢，我们要解锁解锁，是这样一个过程 那么在这里头呢，我们给出这个过程当中，我们可以看到，如果另外一个进程也要使用这- 两种资源 当进程 P 拿到了，比如说磁盘这个资源 而进程 Q 呢拿到了磁带这个资源 而进程 P 又想要磁带，进程 Q 又想要磁盘 那这样的话，这两个进程呢实际上就是互相等待，这样的话就导致了死锁 那然后我们再看另外一个例子 这个例子呢是以内存为例。 假设我们内存呢，现在可以分配的内存呢有 200K 那我们看有两个进程 P1 和 P2 ，那么 P1 呢，它先申请了 80K 然后过一会它又要申请 60 K 对于 P2 来讲呢，它先要申请 70 K ，然后再要申请 80 K ，那么两个进程对资源的一个总和呢是超过了可分配的资源数量的 那么如果 P1 申请到了 80 K P2 申请到了 70 K ，它们要再申请它剩余的要的资源呢就都不能满足了 因为系统的可用资源已经没了。 因此呢 P1 和 P2 呢也要 等待。 那么这样呢，如果系统中没有剩余的资源了，那么这两个进程其实就处于死锁了 那么这是进程呢竞争可重用资源，带来的一个死锁的一种可能 那么我们举一个进程竞争可消耗资源的例子 尽管我们不对它进行相应的处理，我们可以举一个例子来说明 当进程去使用可消耗资源的时候呢，也可能会产生死锁 比如说， P1 它要做的事情是从 P2 得到一个消息，然后呢，再给 P2 发一个信息 而 P2 呢是要从 P1 得到消息 然后呢再给 P1 发个信息。 如果我们的代码是 这样一个编写的顺序，那么 P1 没有得到 P2 所发来的消息的时候呢，P1 就等待 P2 呢也没有得到 P1 发来的消息，P1。 P2 也在等待 于是它们俩就互相等待，都等对方给自己发消息，就产生了死锁现象 那么死锁呢，是系统 运行过程中，进程之间可能发生的一种状态 那么在系统当中呢，其实我们还有另外两个 现象呢，是要和死锁这个

概念加以区分的 其中一个呢叫活锁，一个呢是饥饿 下面呢，我们来看一个例子。 这个例子呢是我们在第四讲 进程同步问题的时候呢，列举的这个 Peterson 算法 用 Peterson 算法来解决进程竞争 进入临界区的问题。 我们来看有两个进程 进程 A 呢需要使用两个资源。 一个是资源 1 ，一个是资源 2 那么它调用了 enter....region 资源 1 呢，来获取资源 1 的使用权 然后再去调用 enter.....region 资源 2 呢 获取资源 2 的使用权，然后它可以做 相应的临界区的资源使用工作 之后呢去退出临界区 如果恰好有一个进程 B 它也要做相同的工作，或者说使用相同的资源 但是它的使用资源的顺序呢是先 使用资源 2 ，啊，先调用 enter.....region 资源 2 ，啊，获取资源 2 的 使用权，再去调用 enter .....region 资源 1 获取资源 1 的使用权，然后做一些资源的临界区的工作 在一个并发系统当中呢，进程 A 和进程 B 呢可以并发执行 如果恰好出现了这样一个情况。 进程 A 呢 已经拿到了资源 1 的使用权 然后它被切换，下面是进程 B 上 CPU 执行。 它获取了资源 2 的使用权 之后 不管是进程 A 执行，还是进程 B 执行，我们都会看到 enter.....region 资源 2 ，或者 enter.....region 资源1 ，都会导致这个进程 在 CPU 上忙等待，因为它，我们讲过了 这个 Peterson 算法呢，它是一个忙等待的一个算法 那么如果进程 A 上 CPU 了，它想获得资源 2 ，那么它就会 不断的去探寻，资源 2 是不是获得，能获得 进程 B 也是一样 那么这两个进程其实都可以上 CPU 执行，但是都没有往前进展 所以我们总结出了这样一个现象 就是首先你要先加锁，啊，实际上我们就说enter.....region 获得这个资源，实际上就给资源加锁 然后给这个资源加完锁之后，它就想获得另外一个资源的锁的时候它就要去轮询 我们知道忙等就是轮询。 因此，每个都是先加了一个锁 然后去轮询下一个资源，然后这个轮询过程中就始终得不到，因此它的状态就是 每一个进程可以去上 CPU，然后时间片到了又下 CPU 没有进展也不阻塞。 它永远能上 CPU，上完 CPU 之后运行一段时间 得不到资源它就下 CPU 了，然后再上 CPU。 这种现象呢我们把它称之为活锁 它得不到资源，但是它总能够有机会去运行 不像死锁，死锁是进程进入了等待状态，那么它不会去竞争 CPU 了 所以活锁是描述了这样一个场景。 另 一个概念呢是饥饿。 这是我们都很熟悉的概念 而饥饿呢，是往往由于资源分配的策略所决定是不是会产生饥饿现象 所以死锁呢要和活锁、 饥饿，啊 这样两种其它不同的现象呢，要加以区分 下面呢我们探讨一下 产生死锁的必要条件。 也就是说，如果死锁发生了 这四个条件就一定存在。 第

一个条件呢 是互斥使用，也称之为资源独占 它指的是呢，资源的这样一种特性 一次只能给一个进程使用。 我们肯定知道像打印机这样的资源 一定是一个独占的资源。 它具有这样一个性质 第二个条件呢，称之为占有且等待 也可以说请求和保持 或者是部分分配。 那么它指的是 进程在申请新的资源的同时 保持对原有资源的一个占有 那么什么叫做部分分配呢，实际上也是说，我可能需要 多种资源，我先拿到了其中一部分资源 然后我再去申请另外的资源，所以是部分分配 前面资源拿着，我申请新的资源 如果新的资源得不到，进入等待状态，而 原来申请到资源的，还在这个进程的手里头。 这就叫做占有且等待 或者是请求，然后呢还保持。 当然了，它的 部分分配是它的一个过程。 第三个条件呢 我们称之为不可抢占，或者是不可剥夺 它指的是资源的申请者 不能够强行从资源的占有者手中夺取资源 也就是这个资源一个进程拿到这个资源之后，只能够由资源的占有者 主动地来，自愿地来释放。 好，那么这就是第三个条件 第四个条件呢，我们称之为循环等待 所谓循环等待呢是指 在系统中，假设有这样一个进程的等待序列 那么不失一般性的，我们把它表示成 P1 P2 到 Pn，有 n 个进程的这样一个进程等待序列 它们满足呢 P1 等待 P2 所占有的资源 也就是 P1 它申请一个资源，这个资源呢 被 P2 这个进程所占有，而 P2 呢又等待 P3 占有的资源。 那么如此 Pn 因为是一个循环等待，所以 Pn 呢 等待的是 P1 占有的资源 这样的话就形成了这样一个循环的等待的对列。 也就是进程存在一个等待的环路 那么这就是产生死锁的四个必要条件 当出现死锁之后，这四个条件都成立 其中，任何一个条件不成立的话，就一定没有死锁发生