操作系统
第十一讲

张涛

Review

进程间通信类型

消息缓冲通信的实现

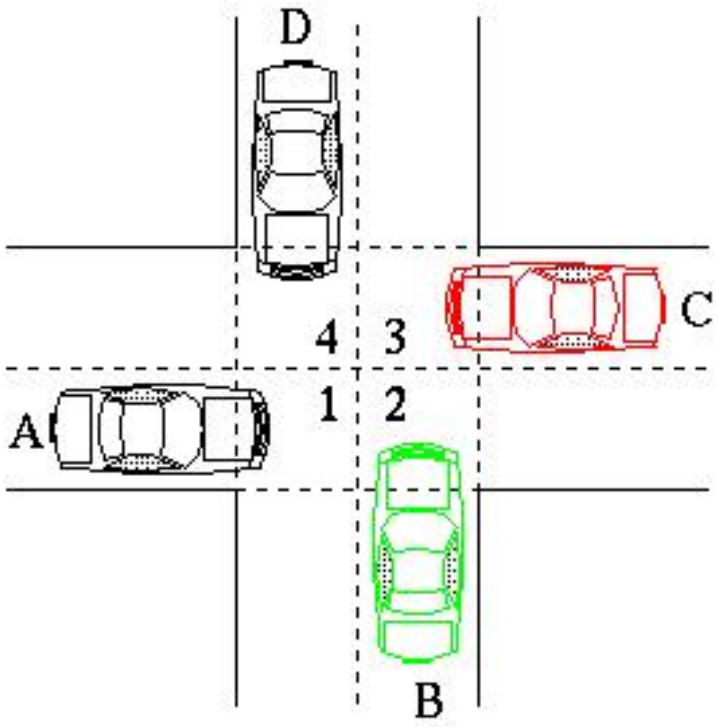
信箱通信的实现

管道通信的实现

3.7 死锁（Deadlock）

- 死锁举例
- 产生死锁的原因和必要条件
- 处理死锁的基本方法
  - 死锁的预防
  - 死锁的避免
  - 死锁的检测
  - 死锁的解除

3.7.1 死锁举例



显然各路车队等待的事件都不会发生。（假设它们都不改变行车方向）这样若不采用特殊方法，它们将永远停留在这“ 井”字形的路上，而处于死锁状态。

系统模型

- 系统拥有一定数量的资源，分布在若干竞争进程之间。
- 资源：
  - 物理资源：内存、CPU、I/O设备（打印机和磁带机等）
  - 逻辑资源：文件、信号量等
- 资源分成多种类型，每种类型有相同数量的实例。如果系统中有两个CPU，那么资源类型CPU就有2个实例。
- 正常操作模式下，进程按如下顺序使用资源
  - 申请（获得资源或者等待）－使用－释放

可重用资源与可消费资源

- 可重用资源
  - 一次只能供一个进程安全地使用，且不会由于使用而耗尽
  - 例子：处理器，I/O通道，主存和辅存，设备，文件、数据库、信号量等数据结构
- 可消费资源
  - 可以创建并且可以销毁的资源
  - 数目没有限制，当一个进程得到一个可消费资源时，这个资源就不再存在了
  - 例子：中断，消息，I/O缓冲区中的信息

两个进程竞争可重用资源死锁的例子

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)

- 两个进程：一个访问磁盘文件D，一个访问磁带设备T
- 如果执行序列为：P0、P1、q0、q1、P2、q2，则发生死锁

竞争外部设备：设系统中有输入、输出设备各一台，进程A、B的代码形式如下，（竞争资源造成死锁）

进程A：

(1) 申请输入设备P(x_1)

(2) 申请输出设备P(x_2)

(3) 释放输入设备V(x_1)

(4) 释放输出设备V(x_2)

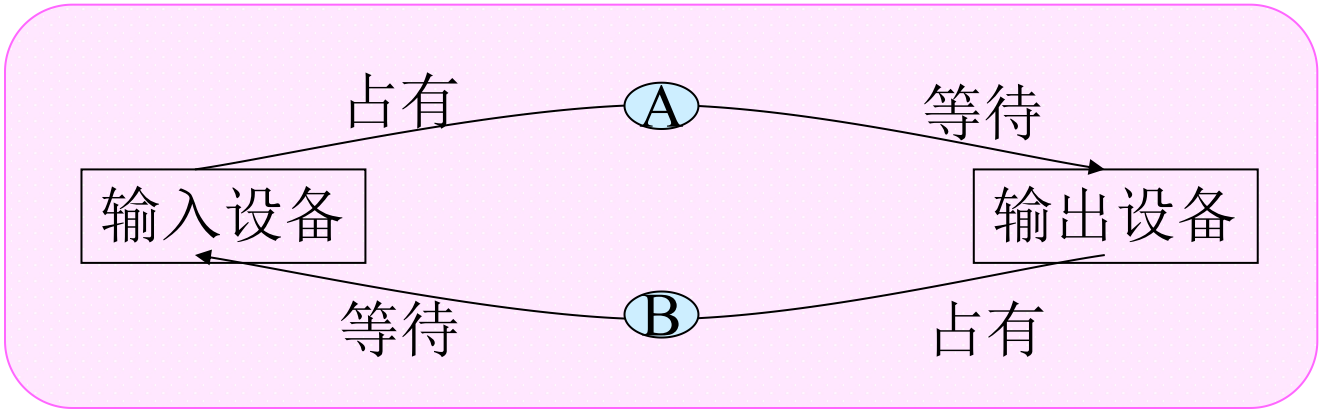
进程B：

(1) 申请输出设备P(x_2)

(2) 申请输入设备P(x_1)

(3) 释放输出设备V(x_2)

(4) 释放输入设备V(x_1)



涉及可消费资源死锁的例子

P1:	*P2*:
…	…
receive(P2);	receive(P1);
…	…
send(P2, M1);	send(P1, M2);
…	…

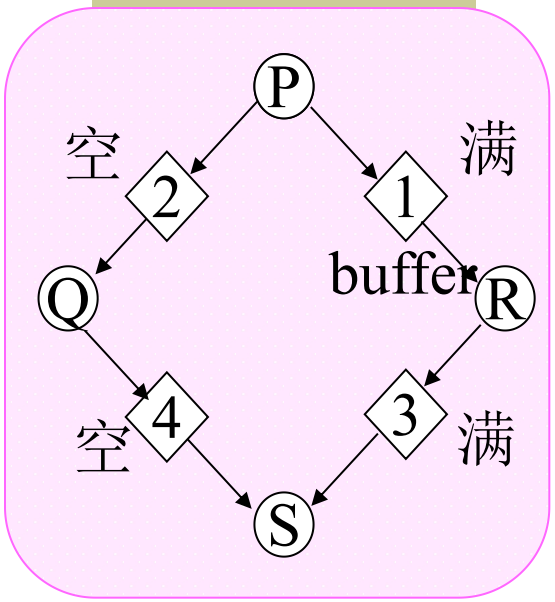
◆每个进程试图从另一个进程接收消息，然后再给它发送一条消息。

◆如果receive阻塞（接收进程被阻塞直到收到消息），则可能发生死锁。

进程通讯 (计算机通讯中)设有四个进程P、S、Q、R，用四个(buffer)缓冲区进行通讯，进程分别有如下代码：

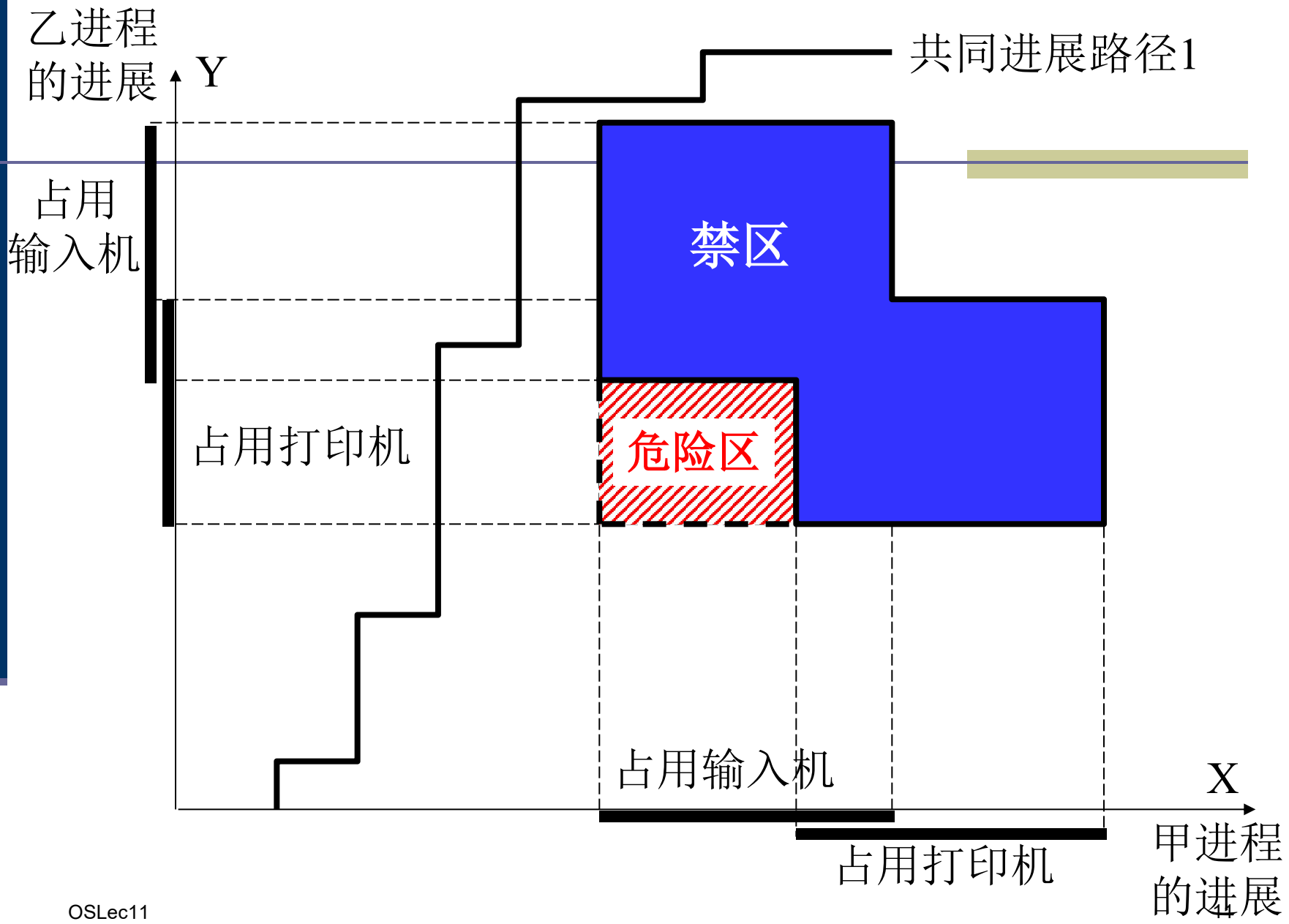
进程P： send (R.1); 通过1号缓冲区向R发信息
waiter (R.answer); 等待R的回答
send (Q.2); 通过2号buffer向Q发信息

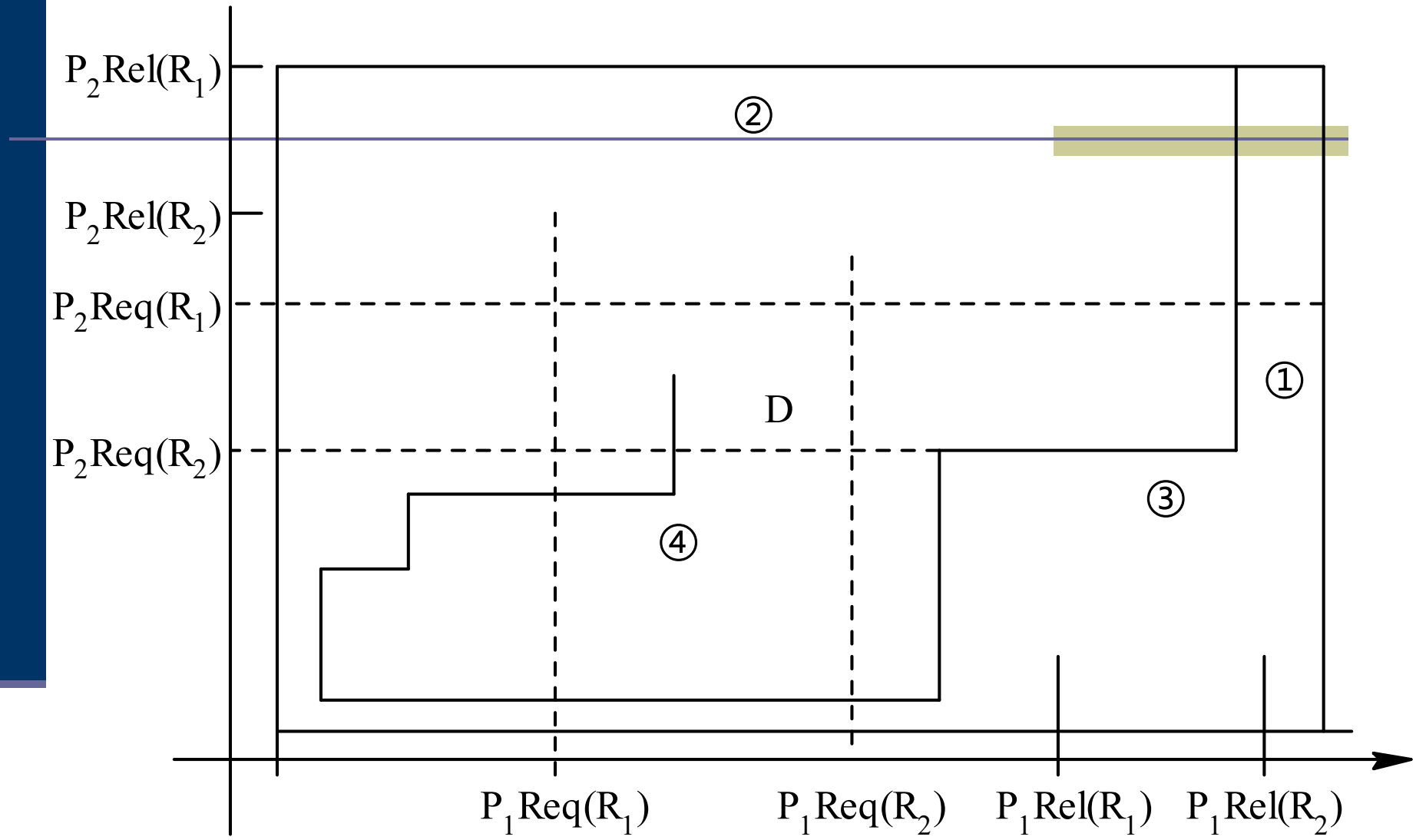
进程R： send (S.3); 通过3号buffer向S发信息
waiter (S.answer); 等待S的回答
receive (P.1); 接收P从1号送来的信息
answer (P); 回答P



进程S： receive (Q.4); 接收Q从4号buffer送来的信息
receive (R.3); 接收R从3号buffer送来的信息
answer (R); 回答R

进程Q： receive (P.2); 接收P从2号buffer送来的信息
send (S.4); 通过4号buffer向S发信息


乙进程
的进展
Y
共同进展路径1
占用
输入机
禁区
占用打印机
危险区
占用输入机
X
占用打印机
甲进程
的进展


P2Rel(R1)
②
P2Rel(R2)
P2Req(R1)
①
D
P2Req(R2)
③
④
P1Req(R1)
P1Req(R2)
P1Rel(R1)
P1Rel(R2)

Deadlock

- 死锁的规范定义

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。

3.7.2 产生死锁的原因和必要条件

- 死锁指进程处于等待状态，且等待事件永远不会发生。
  - 参与死锁的进程最少是两个；
  - 参与死锁的进程至少有两个已经占有资源；
  - 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。
- 产生死锁的原因：
  - 资源不足导致的资源竞争：多个进程所共享的资源不足，引起它们对资源的竞争而产生死锁。
  - 并发执行的顺序不当。进程运行过程中，请求和释放资源的顺序不当，而导致进程死锁. 如P,V操作的顺序不当

产生死锁的四个必要条件

- 互斥(mutual exclusion)
- 占有等待 (hold and wait)
- 不可剥夺 (no preemption)
- 环路等待(circular wait)

G. E. Coffman

Coffman's conditions '71
1. Mutual Exclusion
2. Hold and wait
3. No preemption
4. Circular wait

2 or more Processes

Strategies
1. Ostrich Algorithm
2. Detection + Recovery
3. Dynamic Avoidance
4. Prevention → use Coffman's conditions.

common ☺ when there are usually enough resources & sysadmins to fix or reboot..

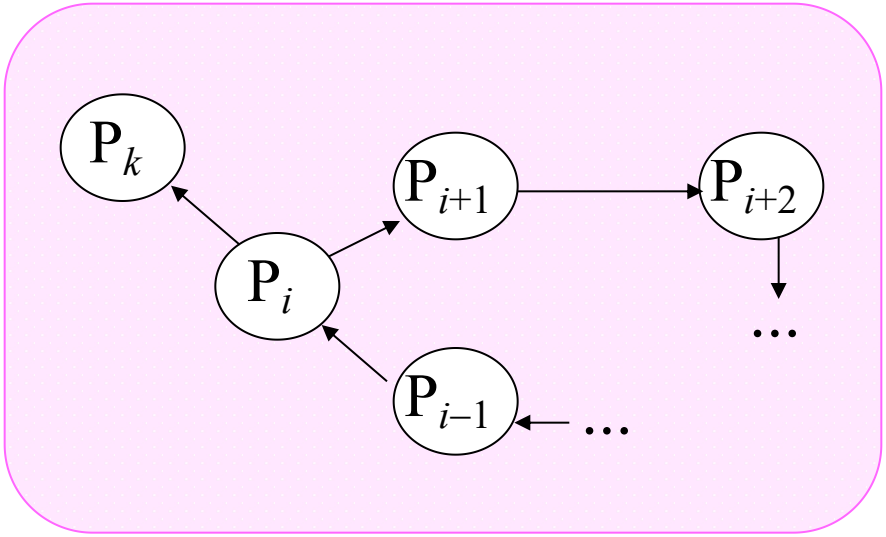
plan ahead

eg: Prevent circular waits by (partially) ordering the resources and program all processes order their holds in non-decreasing order.

四个必要条件

- 互斥条件
  - 指进程对所分配到的资源进行排它性使用，即在一段时间内某资源只能由一个进程占有。如果此时还有其它进程申请该资源，则它只能阻塞，直至占有该资源的进程释放。
- 占有且等待（请求和保持条件）
  - 进程已经保持了至少一个资源，但又提出了新的资源要求，而该资源又已被其它进程占有，此时请求进程阻塞，但又对已经获得的其它资源保持不放。
- 非抢占（非剥夺）条件
  - 进程已获得的资源，在未使用完之前，不能被剥夺，只能在使用完时由自己释放。
- 循环等待条件
  - 在发生死锁时，必然存在一个进程-资源的封闭的环形链．即进程集合{P_0, P_1, P_2, …, P_n}中的P_0正在等待一个P_1占用的资源；P_1正在等待P_2占用的资源，……，P_n正在等待已被P_0占用的资源．

- 系统中有两台输出设备，Pi+1 占有一台，Pk 占有另一台，且 k {0,1,…,n}。虽然系统有一个循环等循圈，但Pk 不在圈内。若Pk释放了输出设备，则可打破循环等待圈。因此循环等待只是死锁的必要条件。



3.7.3 处理死锁的基本方法

- 忽略此问题不做任何实际处理 (鸵鸟策略)
- 不允许出现死锁
  - 死锁预防 (deadlock prevention)
  - 死锁避免 (deadlock avoidance)
- 允许系统出现死锁后排除之
  - 死锁检测 (deadlock detection)
  - 死锁恢复 (deadlock recovery)

死锁处理方法

- 预防死锁：通过限制如何申请资源的方法来确保至少有一个条件不成立。
- 避免死锁：根据有关进程申请资源和使用资源的额外信息，确定对于一个申请，进程是否应该等待。
- 检测死锁和恢复：通过算法来检测并恢复
- 忽视此问题：认为死锁不可能在系统内发生。如Unix采用这种方法。

3.7.4 死锁的预防

- 定义：在系统设计时确定资源分配算法，保证不发生死锁
- 死锁的四个必要条件记做C1，C2，C3，C4，死锁记做D，则有逻辑公式：$D \rightarrow C1 \wedge C2 \wedge C3 \wedge C4$。

  推导得：$\neg C1 \vee \neg C2 \vee \neg C3 \vee \neg C4 \rightarrow \neg D$
- 具体的做法：破坏产生死锁的四个必要条件之一
  - 互斥条件 ×
  - 请求和保持条件（占有和等待）
  - 不可剥夺条件
  - 环路等待条件

破坏请求和保持条件

- 方法一：要求每个进程在运行前必须一次性申请它所要求的所有资源，
- 方法二：进程提出申请资源前必须释放已占有的一切资源。
- 优点：简单、易于实现、安全
- 缺点：
  - 一个进程可能被阻塞很长时间，等待资源，发生饥饿
  - 资源严重浪费，进程延迟运行；

破坏不可剥夺条件

- 方法一： OS可以剥夺一个进 程占有的资源，分配给其他进程（只有当两个进程优先级相同时）。
- 方法二：一个已经保持了某些资源的进程，当它再提出新的资源请求而不能立即得到满足时，必须释放它已经保持的所有资源，待以后需要时再重新申请
- 适用条件
  - 资源的状态可以很容易地保存和恢复，如CPU寄存器、内存空间，不能适用于打印机、磁带机
- 缺点
  - 实现复杂、代价大，反复申请/释放资源，系统开销大，降低系统吞吐量

破坏环路等待条件

- 采用资源有序分配法：
  - 把系统中所有资源编号，进程在申请资源时必须严格按资源编号的递增次序进行，否则操作系统不予分配

$$\begin{array}{cccccc} r_1, & r_2, & \cdots, & r_i \\ \downarrow & \downarrow & & \downarrow \\ 1 & 2 & & i \end{array} \qquad F(r_i) = i$$

- 例：哲学家就餐问题
- 问题：
  - 此方法要求资源类型序号相对稳定，不便于添加新类型的设备.
  - 易造成资源浪费，类型序号的安排只能考虑一般作业的情况，限制用户简单、自主地编程
  - 限制进程对资源的请求；资源的排序占用系统开销；

死锁的预防

- **Prevention**
  - 破坏第一条件（互斥条件）：适用于磁盘
  - 破坏第三条件（非剥夺条件）：适用于CPU、内存
  - 破坏第三条件（占有等待条件）：静态分配策略
  - 破坏第四条件（循环等待条件）：层次分配策略

3.7.5 死锁的避免

- 在系统运行过程中，对进程发出的每一个系统能够满足的资源申请进行动态检查，并根据检查结果决定是否分配资源，若分配后系统可能发生死锁，则不予分配，否则予以分配。
- 安全状态：指系统能按某种进程顺序(P1, P2, …, Pn)来为每个进程Pi分配其所需资源，直至满足每个进程对资源的最大需求，使每个进程都可顺利地完成。如果系统无法找到这样一个安全序列，则称系统处于不安全状态(称〈P1, P2, …, Pn〉序列为安全序列)。

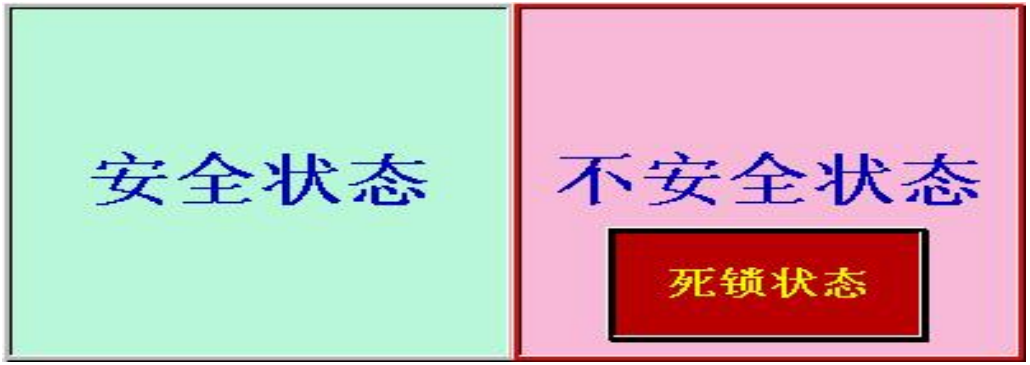
安全状态与不安全状态

安全状态：

如果存在一个由系统中所有进程构成的安全序列P_1，...P_n，则系统处于安全状态

不安全状态：

不存在一个安全序列。不安全状态不一定导致死锁，只是很可能死锁。



安全序列

一个进程序列$\{P_1, \ldots, P_n\}$是安全的，如果对于每一个进程$P_i(1 \leqslant i \leqslant n)$，它以后尚需要的资源量不超过系统当前剩余资源量与所有进程P_j（$j < i$）当前占有资源量之和，系统处于安全状态

- 安全序列可以不唯一！

安全状态一定是没有死锁发生的

不安全状态：不存在一个安全序列。

不安全状态不一定导致死锁

安全状态举例

- 假定系统中有三个进程P1、 P2和P3，共有12台磁带机。进程P1总共要求10台磁带机，P2和P3分别要求4台和9台。假设在T0时刻，进程P1、P2和P3已分别获得5台、2台和2台磁带机，尚有3台空闲未分配，如下表所示：

进 程	最 大 需 求	已 分 配	可 用
P_1	10	5	3
P_2	4	2	
P_3	9	2	

死锁 → 四个必要条件

¬(4个必要条件)→¬(死锁)

4个必要条件 ↛ 死锁（不一定）

死锁预防是严格破坏4个必要条件之一，一定不出现死锁；而死锁的避免是不那么严格地限制死锁必要条件的存在，其目的是提高系统的资源利用率。万一当死锁有可能出现时，就小心避免这种情况的发生。

死锁避免（Deadlock Avoidance）

- 不需象死锁预防那样，事先采取限制措施破坏产生死锁的必要条件；在资源的动态分配过程中，采用某种策略防止系统进入不安全状态，从而避免发生死锁
  - 如果一个新的进程的资源请求会导致死锁，则拒绝启动这个进程
  - 如果满足一个进程新提出的一项资源请求会导致死锁，则拒绝分配资源给这个进程

银行家算法

例：假设有三个进程P、Q、R，系统只有某类资源共10个，而三个进程合计申请资源数为20个。

目前的分配情况如下：

进程	已占资源个数Loan	需申请资源个数Claim	need = loan + claim
P	4	4	8
Q	2	1	3
R	2	7	9
合计	8	12	20

而系统　　8 + 2 = 10

↑　　↑　　↑

已分配　剩　共有

Loan　Cash　Capital

此后P、R再申请资源就不能分配了。因为现在只剩下2个资源，不能满足它们的最大要求 (P:4, R:7)，如果将剩下2个分配给P或R，则会产生死锁。

	已占	还申						
P	5	3	或	5	3	或	4	4
Q	2	1		2	1		2	1
R	3	6		3	6		4	5

然而将2个资源分给Q (只需一个)则

P	4	4		P	4	4
Q	3	0	$\Rightarrow$ Q可运行结束、释放			系统资源回收为4个
R	2	7		R	2	7

此后可将4个资源分组P…

即 $Q \rightarrow P \rightarrow R$

* 单一种类资源引出的银行家算法的基本思想也同样适用于多种类的资源情况：

	r_1	r_2	r_3	r_4
A	3	0	1	1
B	0	1	0	0
C	1	1	1	0
D	1	1	0	1
E	0	0	0	0

资源已分配情况

r_1	r_2	r_3	r_4
4	1	1	1
0	2	1	2
4	2	1	0
1	1	1	1
2	1	1	0

最大需求

还需(剩余需求)矩阵

$$\begin{array}{c} \\ A \\ B \\ C \\ D \\ E \end{array}\begin{pmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 2 \\ 3 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 2 & 1 & 1 & 0 \end{pmatrix}$$

总共资源数　r = (6, 3, 4, 2)

已分资源数　s = (5, 3, 2, 2)

余下资源数　t = (1, 0, 2, 0)

(1) 寻找剩余矩阵的某一未标记的行x，使得它的每一个元素都不大于向量t的对应元素，如果找不到转(4)。

(2) 对于找到的行x，表示可以满足它，标记此进程，并将它占有的资源加到向量t。

(3) 重复上述步骤，转(1)。

(4) 如果所有进程都已标记，则状态是安全的，否则为不安全。

What you need to do?

- 复习课本3. 7节的内容
- 课后作业： 习题17、18

See you next time!