操作系统

Operating system

孔维强

大连理工大学

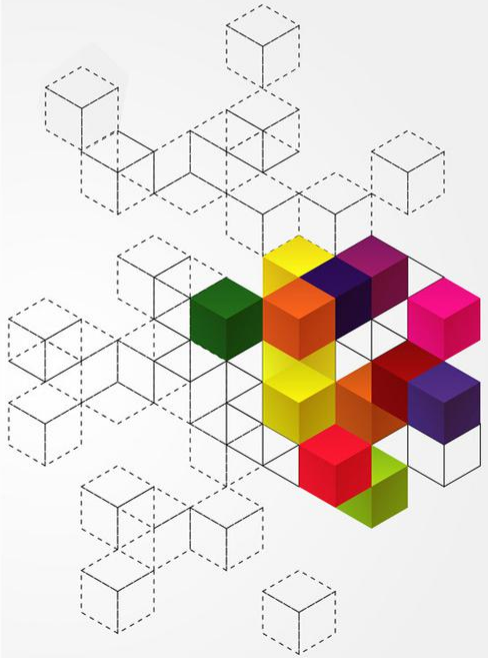
内容纲要

6.2 临界区问题

一、 互斥概念

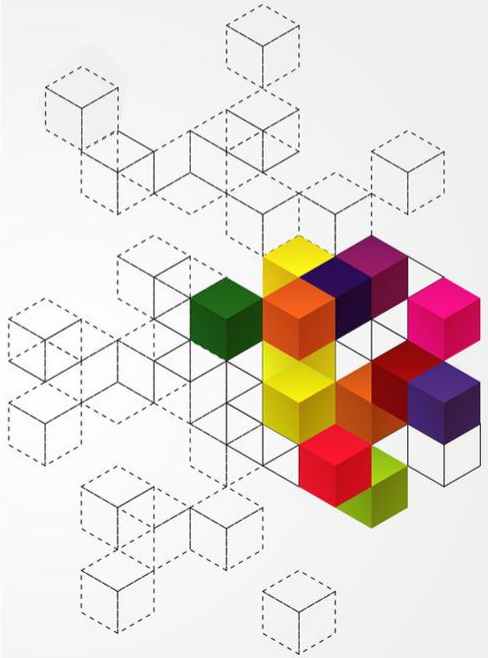
二、 临界区

三、 临界资源



一、互斥概念

- **互斥是进程间接协作的关键性质**
  - 间接进程协作下的进程推进的无序性，导致对其处理的难度相对较大
  - 对共享数据的并发访问可能导致数据不一致
  - 为了保证数据一致性，需保证协作进程按照特定的操作执行顺序进行



一、互斥概念

- **为什么需要分析间接协作性质（示例）**
  - 不同的进程推进顺序，导致不同的结果（不一致）

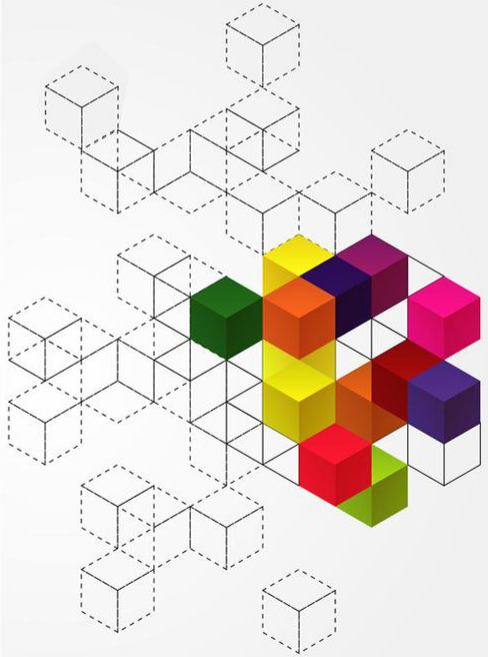
P1:

```
if(x>=100){
    x-=100;
}
```

P2:

```
if(x>=100){
    x-=100;
}
```

进程P1 if(x>=100);	进程P1 if(x>=100);
进程P2 If(x>=100)	进程P1 X-=100;
进程P1 X-=100;	进程P2 If(x>=100)
进程P2 X-=100;	进程P2 X-=100;



竞争条件：输出的结果与特定的执行顺序相关（进程同步解决此问题）

二、临界区

临界区示例:

P_1:

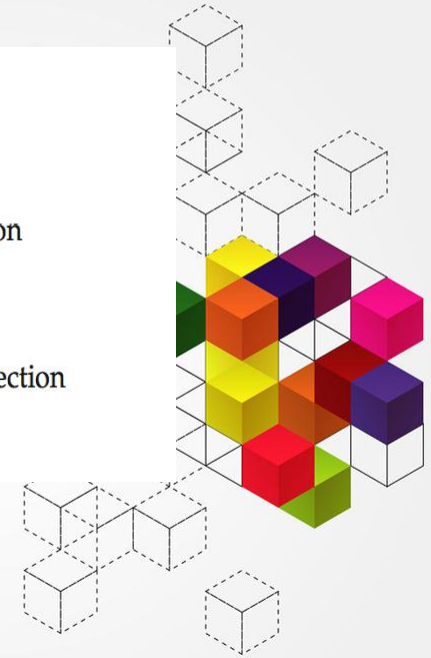
```
while (true)
{
   if(x>=100){
      x-=100;
   }
}
```

P_2:

```
while (true)
{
   if(x>=100){
      x-=100;
   }
}
```

红框区域： **Critical Section**
(Access to shared variable x)

```
do {
      entry section
         critical section
      exit section
         remainder section
} while (true);
```



二、临界区

临界区:

考虑有n个进程{p_0, p_1, …, p_{n-1}}

每个进程有其自身的临界区（Critical Section, CS）代码

- ✓ 进程可能在临界区内改变共享变量、更新表格、写文件等
- ✓ 当一个进程在其CS内时，其他进程不能在其CS内

  也就是说，同时不能有2个及以上进程再其CS内

临界区问题即设计协议（protocol）去解决此问题

每个进程在进入临界区前应获得允许（entry section）

counter++及counter--应放在CS中

二、临界区

临界区条件:

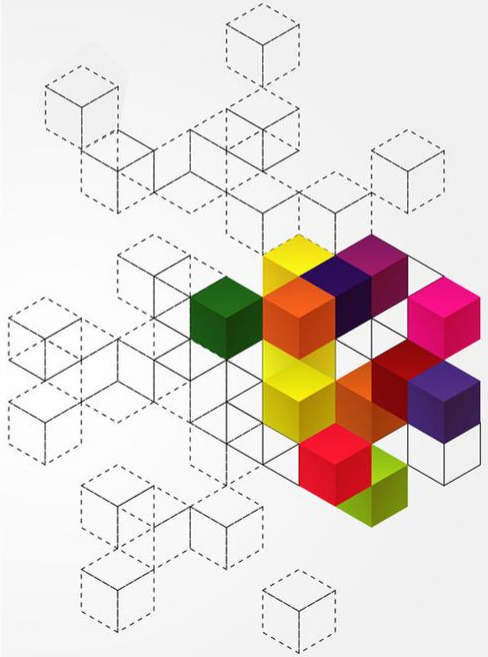
临界区的解决方案应满足3个条件：

1. 互斥（mutual Exclusion）：当进程Pi在其CS时，其他进程不能在其CS内
2. 前进（progress）：当无进程在其CS内且有进程希望进入到其CS时，选择下一进入CS的进程不能无限期推迟
3. 有限等待（bounded waiting）：存在一个限界bound，当一个进程请求进入其CS其在该请求得到允许的期间内，其他进程进入其CS的次数不能超过bound

三、临界资源

- **典型的临界资源**

临界资源类型	临界资源实例
硬件资源	打印机、磁带机
软件资源	消息缓冲队列、共享变量、 全局数组、共享缓冲区



本讲小结

- 互斥概念
- 临界区
- 临界资源