信号量（Semaphore）

信号量 (Semaphore)

- 信号量 S 是个整型变量
- 信号量 S 只允许两个标准操作：wait() 和 signal()
  - 或者，发明人称之为 P 操作、V 操作
- wait() 和 signal() 是原子操作 (atomic operations)

如何实现这种信号量？

- 为信号量设计一个等待队列 queue
- 等待队列的节点 (node) 类型一致，都包含 2 项数据单元:
  - int value;
  - pointer to next node in the queue

如何实现这种信号量？

等待队列带 2 个操作函数：

- Block() – 阻塞自己，主动交出 CPU，引起新一轮 CPU 调度
- Wakeup() – 唤醒等待队列里的一个进程，将它移入就绪队列

如何实现这种信号量？

- wait 操作

```
wait (S){
    value--;
    if (value < 0) {
        add this process to waiting queue
    block();  }
}
```

如何实现这种信号量？

- signal 操作

```
signal (S){
   value++;
   if (value <= 0) {
      remove a process P from the waiting queue
      wakeup(P);  }
}
```

信号量解决无限多进程的临界区问题

- 进程的临界区必须符合如下框架

```
Semaphore S;    //   初始值为  1
do{
    wait (S);
        Critical Section;
    signal (S);
        remainder section;
} while(1);
```

WHY ？

- 此临界区框架符合 Mutual Exclusive
- 此临界区框架符合 Progress
- 此临界区框架符合 Bounded Waiting

信号量解决一般性进程同步问题

- 假设要求：执行完进程 P_i 的 A 之后，才允许执行进程 P_j 的 B
- 定义信号量 *flag* ，初值为 0
- 代码框架：

P_i	P_j
⋮	⋮
A	*wait*(*flag*)
signal(*flag*)	B


End