经典同步问题（Classical Problems）

经典同步问题

- 生产者 - 消费者问题

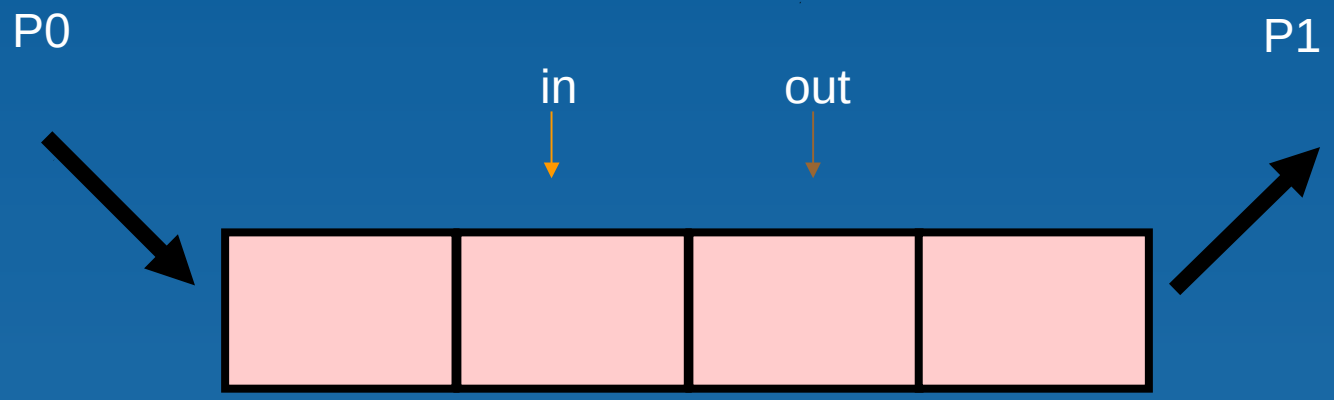
  Producer-Consumer Problem
- 读者 - 写者问题

  Readers and Writers Problem
- 哲学家就餐问题

  Dining-Philosophers Problem

生产者 - 消费者问题

- 生产者进程“生产”出信息，存储在缓冲区，供消费者进程“消费”
  - *unbounded-buffer* 缓冲区的容量无限
  - *bounded-buffer* 缓冲区的容量有限



生产者 - 消费者问题的一种解决方案

- 假设一个拥有 *N* 个 items 的缓冲区

- 定义信号量 mutex ，初始值为 1
- 定义信号量 full ，初始值为 0
- 定义信号量 empty ，初始值为 N

生产者-消费者问题的一种解决方案（续）

- 生产者进程 producer

```
while (true)  {
            //   produce an item
        wait (empty);
        wait (mutex);
            //  add the item to the  buffer
         signal (mutex);
         signal (full);
}
```

生产者 - 消费者问题的一种解决方案（续）

- 消费者进程 consumer

```
while (true) {
        wait (full);
        wait (mutex);
                //  remove an item from  buffer
        signal (mutex);
        signal (empty);
                //  consume the removed item
}
```

死锁 (Deadlock) vs 饥饿 (Starvation)

- 如果 wait 操作排序不当，会怎样？

Producer

```
wait (empty);
wait (mutex);

.

.

signal  (mutex);
signal (full);
```

Consumer

```
wait (mutex);
wait (full);

.

.

signal (mutex);
signal (empty);
```

死锁 vs 饥饿

- 死锁 (Deadlock) – two or more processes are waiting indefinitely for an event that can be caused by only one of the waiting processes
- 饥饿 (Starvation) – indefinite blocking. A process may never be removed from the semaphore queue in which it is suspended
- 关于死锁的概念，会在后续单元详述

读者 - 写者 (Readers-Writers) 问题

- 场景：一组并发进程共享一个数据集，其中
  - 读者 (Readers) 进程 – 只对数据集实施“读”操作，从不试图更新数据集
  - 写者 (Writers) 进程 – 对数据集既实施“读”操作，也实施“写”操作
- 问题

  允许若干读者进程同时读取数据集；在任何时刻，只允许单个写者进程访问数据集。
- 如何保证？

第一类读者 - 写者问题 (First Readers-Writers)

- No readers will be kept waiting unless a writer has already obtained permission to use the shared object.
- 读者进程不会被要求等待，除非已经有一个写者进程获准访问共享数据集
- In other words, no reader should wait for other readers to finish simply because a writer is waiting.

第一类读者 - 写者问题的一个解决方案

- 共享数据集
- 定义
  - 信号量 mutex ，初始值为 1
  - 信号量 wrt ，初始值为 1
  - 整型变量 readcount ，初始值为 0

第一类读者 - 写者问题的一个解决方案

- 写者进程

```
while (true) {
        wait (wrt) ;

            //    writing is performed

        signal (wrt) ;
}
```

第一类读者 - 写者问题的一个解决方案

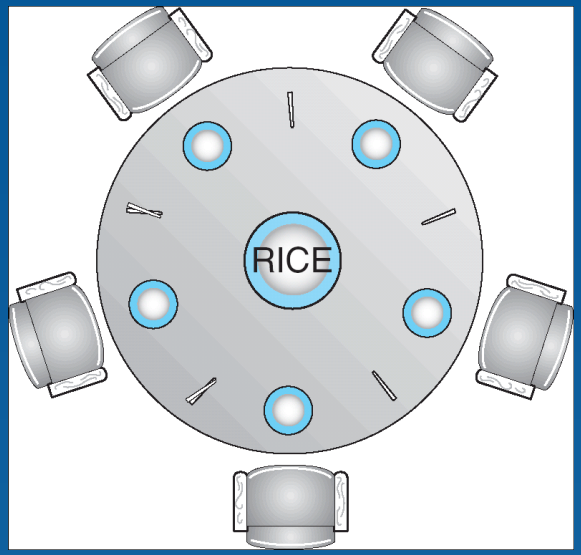
- 读者进程

```
while (true) {
        wait (mutex) ;
        readcount ++ ;
        if (readercount == 1)  wait (wrt) ;
        signal (mutex)

            // reading is performed

        wait (mutex) ;
        readcount  - - ;
        if (redacount  == 0)  signal (wrt) ;
        signal (mutex) ;
}
```

哲学家就餐 (Dining-Philosophers) 问题



- 共享数据集：碗里的食物
- 定义信号量数组 chopstick [5] ， 初始值为 1

哲学家就餐 (Dining-Philosophers) 问题

- 哲学家进程 *i*

```
While (true)  {
        wait ( chopstick[i] );
        wait ( chopStick[ (i + 1) % 5] );
                // eat
        signal ( chopstick[i] );
        signal (chopstick[ (i + 1) % 5] );
                // think
}
```

  - 此方案有缺陷：死锁！


End