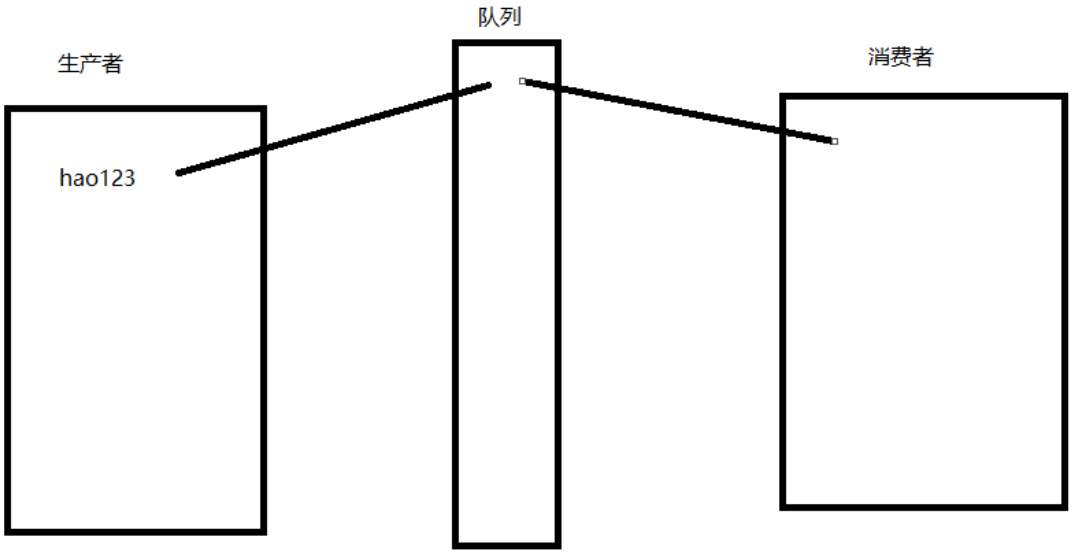
- 进程间的通信
  - 进程队列

```python
import time
from multiprocessing import Process, Queue

# maxsize设置队列中数据的上限，小于或者等于0  则不限制
# 容器中大于这个数则阻塞，直到队列中的数据被消除
q = Queue(maxsize=0)  # 创建队列

# 写入队列数据
q.put(0)
q.put(1)
q.put(2)
q.put(3)

# 删除队列数据，并返回这个数据
print(q.get())
print(q.get())
print(q.get())
print(q.get())
print(q.get()) # 没有了    会阻塞


print(q.qsize())
```

  - 生产者消费者模型
  - 例子：爬虫
    - 爬取5000个网页
    - 把网页的url拿到 url：全局资源定位符 www.baidu.com
    - 获取网页、从网页的内容中提取所有的url



- 实现生产者消费者模型

```python
from multiprocessing import Process, Queue
import time
import random

# 消费者
def consumer(q, name):
    # 处理数据
    while True:
        food = q.get()
        if food is None:
            break
        print("{}吃了一个{}".format(name, food))
# 生产者
def producer(q, name, food):
    for i in range(10):
        time.sleep(random.uniform(0.2, 0.9))
        print("{}生产了{}{}".format(name, food, i))
        q.put(food+str(i))
if __name__ == '__main__':
    q = Queue()
    c1 = Process(target=consumer, args=(q, "故辞"))
    c2 = Process(target=consumer, args=(q, "幕后煮屎"))
    c1.start()
    c2.start()
    p1 = Process(target=producer, args=(q, "尹精赛", "小熊软糖"))
```

```
    p2 = Process(target=producer, args=(q, "胡与同", "老干妈"))
    p1.start()
    p2.start()
    p1.join()
    p2.join()
    q.put(None)
    q.put(None)
```

- 进程池
  - 为什么要有进程池?
    - i. 一个跑道最多一次跑8个人，那我500个人怎么跑?
    - ii. 开启过多的进程并不能提高你的效率，反而降低我的效率。
    - iii. 原因是操作系统不能同时调度这500个进程，而且进程还得占空间，浪费资源
  - a. 我们的程序有两种
    - i. 计算密集型：充分利用cpu，多进程可以充分利用多核

      (适合开启多进程，但是不适合开启很多多进程)
    - ii. IO密集型：大部分的时间在阻塞队列，而不是在运行状态中

      (根本不适合开启多进程)

```
# 500任务（衣服）


# 信号量
# 500件衣服      任务
# 500个人        进程
# 只有4台机器    cpu    （抢这4台机器）

# 多进程
# 500件衣服      任务
# 500个人        进程
# 只有4台机器    cpu    （抢这4台机器）

# 进程池
# 500件衣服      任务
# 4个人          进程
# 只有4台机器    cpu

```

```python

import time
from multiprocessing import Pool, Process


def func(num):
    print("做了第{}件衣服".format(num))


if __name__ == '__main__':
    # 进程池
    start = time.time()
    # 创建一个进程池
    p = Pool(4)
    for i in range(500):
        # 异步提交任务到一个子进程中
        p.apply_async(func, args=(i,))

    p.close()  # 关闭进程池，用户不能在提交任务
    p.join()
    time1 = time.time() - start
    print("==============")
    print("==============")
    print("==============")
    print("==============")

    # 多进程
    start1 = time.time()
    p_li = []
    for i in range(500):
        p = Process(target=func, args=(i, ))
        p.start()
        p_li.append(p)
    for p in p_li:
        p.join()
    time2 = time.time() - start1

    print("进程池做500个任务需要{}".format(time1))
    print("多进程做500个任务需要{}".format(time2))
```