- join控制子进程
  - 先看一个例子

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


def func(index):
    time.sleep(random.random())
    print('第%s个邮件已经发送完毕' % index)


if __name__ == '__main__':
    for i in range(10):
        Process(target=func, args=(i,)).start()
    print('10个邮件发送完毕')
```

  - 这时发现了问题，怎么解决?

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


def func(index):
    time.sleep(random.random())
    print('第%s个邮件已经发送完毕' % index)


if __name__ == '__main__':
    p_li = []
    for i in range(10):
        p = Process(target=func, args=(i,))
        p.start()
        p_li.append(p)
    for p in p_li:
        p.join()  # 阻塞    直到p进程的任务结束才解除阻塞

```

```
    print('10个邮件发送完毕')
```

- 守护进程

```
from multiprocessing import Process
import time

def func():
    print("子进程 start")
    time.sleep(3)
    print("子进程 end")


if __name__ == '__main__':
    p = Process(target=func)
    # 设置p为守护进程(必须在start()方法之前)
    p.daemon = True
    p.start()  # 启动守护进程
    time.sleep(2)
    print("主进程")


# 守护进程会随着主进程代码的执行完毕而结束
```

再来看一个例子

```
from multiprocessing import Process
import time


def func1():
    count = 1
    while True:
        time.sleep(0.5)
        print("*" * count)
        count += 1

def func2():
    print("func2 start")
    time.sleep(10)
```

```
    print("func2 end")


if __name__ == '__main__':
    # 守护进程
    p1 = Process(target=func1)
    p1.daemon = True
    p1.start()

    # 子进程
    p2 = Process(target=func2)
    p2.start()

    time.sleep(3)
    print("主进程")

# 如果主进程的代码已经执行完毕，但是子进程还有执行完，守护进程都不会继续执行。

# 那守护进程能做什么？
# 程序的报活

# 守护进程：每隔一段时间就向一台机器汇报自己的状态

def fun():
    while True:
        time.sleep(5 * 60)
        print("我还活着")

def main():
    # 主程序会7*24小时提供服务
    pass
```

- 同步控制
  - Lock锁
  - 例子：抢票
    - 建一个txt文件

      {"count": 2}
    - 实现抢票，先查票

```python
import time
import json
from multiprocessing import Process


def search(person):
    with open('a') as f:
        dic = json.load(f)
    time.sleep(0.2)  # 查票的时候经历一个网络延迟
    print('{}查询余票 : '.format(person), dic['count'])


if __name__ == '__main__':
    for i in range(10):
        p = Process(target=search, args=('person%s' % i,))
        p.start()
```

- 再抢票

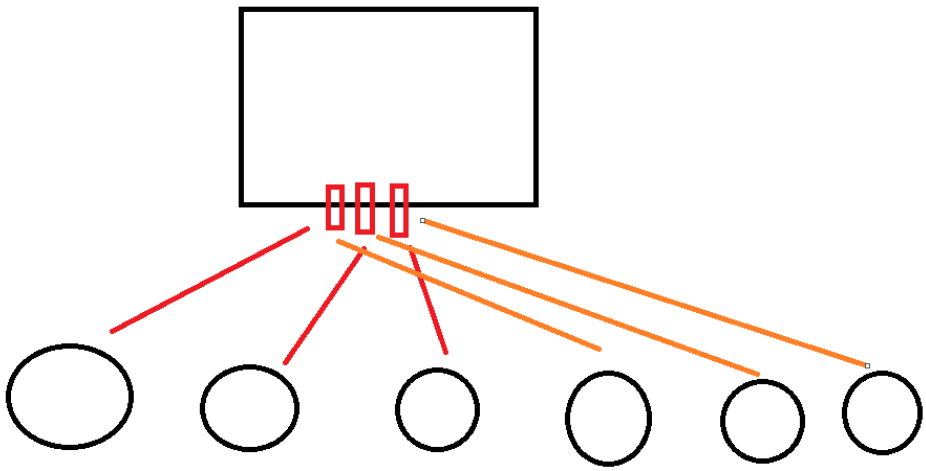
```python
def get_ticket(person):
    with open('a') as f:
        dic = json.load(f)
    time.sleep(0.2)
    if dic['count'] > 0:
        print('%s买到票了' % person)
        dic['count'] -= 1
        time.sleep(0.2)
        with open('a', 'w') as f:
            json.dump(dic, f)
    else:
        print('%s没有买到票' % person)


if __name__ == '__main__':
    for i in range(10):
        p = Process(target=get_ticket, args=('person%s' % i,))
        p.start()
```

- 怎么解决这个问题（加锁）

```python
import time
import json
from multiprocessing import Process, Lock  # 进程锁（互斥锁）


# def search(person):
#     with open('a') as f:
#         dic = json.load(f)
#     time.sleep(0.2)  # 查票的时候经历一个网络延迟
#     print('{}查询余票 : '.format(person), dic['count'])

def get_ticket(person, lock):
    # 加锁
    lock.acquire()

    with open('a') as f:
        dic = json.load(f)
    time.sleep(0.2)
    if dic['count'] > 0:
        print('%s买到票了' % person)
        dic['count'] -= 1
        time.sleep(0.2)
        with open('a', 'w') as f:
            json.dump(dic, f)
    else:
        print('%s没有买到票' % person)

    # 释放锁
    lock.release()


if __name__ == '__main__':
    # 创建锁对象
    lock = Lock()

    for i in range(10):
        p = Process(target=get_ticket, args=('person%s' % i, lock))
        p.start()
```

```
39
40 # 之所以加锁，是为了保证数据的安全
41 # 在异步的情况下，多个进程有可能同时修改同一份资源
```

- 信号量
  - 举个例子



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


def ktv(person, sem):
    # 加锁
    sem.acquire()

    print('%s走进ktv' % person)
    time.sleep(5)
    print('%s走出ktv' % person)

    # 释放锁
    sem.release()


if __name__ == '__main__':
    # 创建一个信号量
    sem = Semaphore(4)

    for i in range(10):
        p = Process(target=ktv, args=('person%s' % i, sem))
```

```
        p.start()
```

- 事件
  - 举个例子（红灯停，绿灯行）

```
import time
from multiprocessing import Process, Event


def traffic_light(e):
    print('\033[31m红灯亮\033[0m')
    # flag = False
    while True:
        # if flag:
        if e.is_set():
            time.sleep(2)
            e.clear()
            print('\033[31m红灯亮\033[0m')
        else:
            time.sleep(2)
            e.set()
            print('\033[32m绿灯亮\033[0m')

def car(e, i):
    if not e.is_set():
        print('car %s 在等待' % i)
        e.wait()
    print('car %s 通过了' % i)


if __name__ == '__main__':

    # 事件
    e = Event()  # flag属性（默认是fales）
    # print(e.is_set())
    # 守护进程（红绿灯循环）
    p1 = Process(target=traffic_light, args=(e, ))
    p1.daemon = True
    p1.start()
```

```
35
36      p_lst = []
37      for i in range(20):
38          time.sleep(1)
39          p2 = Process(target=car, args=(e, i))
40          p2.start()
41          p_lst.append(p2)
42
43      for p in p_lst:
44          p.join()
45
46  """
47  对象.set()
48      作用：设置一个事件的状态为True
49  对象.clear()
50      作用：设置一个事件的状态为False
51  对象.is_set()
52      作用：查看当前事件的状态
53  对象.wait()
54      根据事件的状态，判断是否阻塞
55          若状态为True,则不阻塞
56          若状态为False，则阻塞
57      注意：阻塞的是 对象.wait()后面的内容
58  """
```