

- join控制子进程
 - 先看一个例子

```
1 import time
2 import random
3 from multiprocessing import Process
4
5
6 def func(index):
7     time.sleep(random.random())
8     print('第%s个邮件已经发送完毕' % index)
9
10
11 if __name__ == '__main__':
12     for i in range(10):
13         Process(target=func, args=(i,)).start()
14     print('10个邮件发送完毕')
```

- 这时发现了问题，怎么解决？

```
1 import time
2 import random
3 from multiprocessing import Process
4
5
6 def func(index):
7     time.sleep(random.random())
8     print('第%s个邮件已经发送完毕' % index)
9
10
11 if __name__ == '__main__':
12     p_li = []
13     for i in range(10):
14         p = Process(target=func, args=(i,))
15         p.start()
16         p_li.append(p)
17     for p in p_li:
18         p.join() # 阻塞    直到p进程的任务结束才解除阻塞
19
```

```
20 print('10个邮件发送完毕')
```

- 守护进程

```
1 from multiprocessing import Process
2 import time
3
4 def func():
5     print("子进程 start")
6     time.sleep(3)
7     print("子进程 end")
8
9
10 if __name__ == '__main__':
11     p = Process(target=func)
12     # 设置p为守护进程(必须在start()方法之前)
13     p.daemon = True
14     p.start() # 启动守护进程
15     time.sleep(2)
16     print("主进程")
17
18
19 # 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

```
1 from multiprocessing import Process
2 import time
3
4
5 def func1():
6     count = 1
7     while True:
8         time.sleep(0.5)
9         print("*" * count)
10        count += 1
11
12 def func2():
13     print("func2 start")
14     time.sleep(10)
```

```

15     print("func2 end")
16
17
18 if __name__ == '__main__':
19     # 守护进程
20     p1 = Process(target=func1)
21     p1.daemon = True
22     p1.start()
23
24     # 子进程
25     p2 = Process(target=func2)
26     p2.start()
27
28     time.sleep(3)
29     print("主进程")
30
31 # 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。
32
33 # 那守护进程能做什么？
34 # 程序的报活
35
36 # 守护进程：每隔一段时间就向一台机器汇报自己的状态
37
38 def fun():
39     while True:
40         time.sleep(5 * 60)
41         print("我还活着")
42
43 def main():
44     # 主程序会7*24小时提供服务
45     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

```

1 import time
2 import json
3 from multiprocessing import Process
4
5
6 def search(person):
7     with open('a') as f:
8         dic = json.load(f)
9         time.sleep(0.2) # 查票的时候经历一个网络延迟
10        print('{}查询余票 : '.format(person), dic['count'])
11
12
13 if __name__ == '__main__':
14     for i in range(10):
15         p = Process(target=search, args=('person%s' % i,))
16         p.start()

```

■ 再抢票

```

1 def get_ticket(person):
2     with open('a') as f:
3         dic = json.load(f)
4         time.sleep(0.2)
5         if dic['count'] > 0:
6             print('%s买到票了' % person)
7             dic['count'] -= 1
8             time.sleep(0.2)
9             with open('a', 'w') as f:
10                json.dump(dic, f)
11        else:
12            print('%s没有买到票' % person)
13
14
15 if __name__ == '__main__':
16     for i in range(10):
17         p = Process(target=get_ticket, args=('person%s' % i,))
18         p.start()

```

■ 怎么解决这个问题（加锁）

```
1 import time
2 import json
3 from multiprocessing import Process, Lock # 进程锁（互斥锁）
4
5
6 # def search(person):
7 #     with open('a') as f:
8 #         dic = json.load(f)
9 #         time.sleep(0.2) # 查票的时候经历一个网络延迟
10 #         print('{}查询余票 : '.format(person), dic['count'])
11
12 def get_ticket(person, lock):
13     # 加锁
14     lock.acquire()
15
16     with open('a') as f:
17         dic = json.load(f)
18     time.sleep(0.2)
19     if dic['count'] > 0:
20         print('%s买到票了' % person)
21         dic['count'] -= 1
22         time.sleep(0.2)
23         with open('a', 'w') as f:
24             json.dump(dic, f)
25     else:
26         print('%s没有买到票' % person)
27
28     # 释放锁
29     lock.release()
30
31
32 if __name__ == '__main__':
33     # 创建锁对象
34     lock = Lock()
35
36     for i in range(10):
37         p = Process(target=get_ticket, args=('person%s' % i, lock))
38         p.start()
```

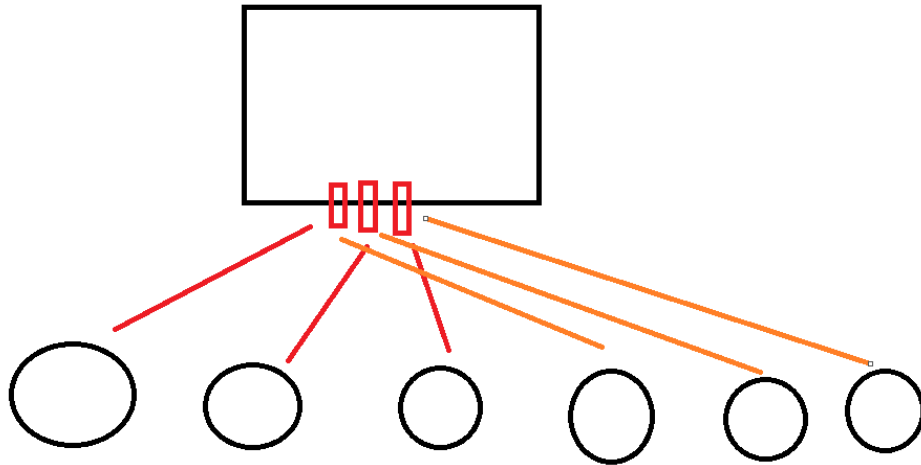
39

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



```
1 import time
2 from multiprocessing import Process, Semaphore
3
4
5 def ktv(person, sem):
6     # 加锁
7     sem.acquire()
8
9     print('%s走进ktv' % person)
10    time.sleep(5)
11    print('%s走出ktv' % person)
12
13    # 释放锁
14    sem.release()
15
16
17 if __name__ == '__main__':
18     # 创建一个信号量
19     sem = Semaphore(4)
20
21     for i in range(10):
22         p = Process(target=ktv, args=('person%s' % i, sem))
```

- 事件

- 举个例子 (红灯停, 绿灯行)

```
1 import time
2 from multiprocessing import Process, Event
3
4
5 def traffic_light(e):
6     print('\033[31m红灯亮\033[0m')
7     # flag = False
8     while True:
9         # if flag:
10            if e.is_set():
11                time.sleep(2)
12                e.clear()
13                print('\033[31m红灯亮\033[0m')
14            else:
15                time.sleep(2)
16                e.set()
17                print('\033[32m绿灯亮\033[0m')
18
19 def car(e, i):
20     if not e.is_set():
21         print('car %s 在等待' % i)
22         e.wait()
23     print('car %s 通过了' % i)
24
25
26 if __name__ == '__main__':
27
28     # 事件
29     e = Event() # flag属性 (默认是fales)
30     # print(e.is_set())
31     # 守护进程 (红绿灯循环)
32     p1 = Process(target=traffic_light, args=(e, ))
33     p1.daemon = True
34     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 """
```