一般情况下，线程都是一个接着一个执行，但某些情况下需要同时执行。

利用屏障同步

java中的CyclicBarrier，cb.wait会是线程阻塞在这个点，直到所有线程到到达这个点，然后执行CyclicBarrier里面的方法。

```java
public class Cyclicbarrier {

    public static void main(String [] args) {
        final CyclicBarrier cb= new CyclicBarrier(3, new Runnable() {
            @Override
            public void run() {
                System.out.println("人员全部到齐，拍照留恋");
                try {
                    Thread.sleep(3000);
                } catch (Exception e) {
                    e.printStackTrace();
                }
            }
        });
        ExecutorService es = Executors.newCachedThreadPool(); //线程池
        for(int i = 0; i < 3; i ++) {
            final int user = i + 1;
            Runnable runnable = new Runnable() {
                @Override
                public void run() {
                    try {
                        Thread.sleep((long)(Math.random() *1000));
                        System.out.println(user + "用户到达聚餐地点，当前已有" + (cb.getNumberWaiting() + 1) + "人到达'
                        cb.await();//所有线程阻塞在这儿
                        if(user == 1) {
                            System.out.println("人员全部到齐，开始吃饭");
                        }
                        Thread.sleep((long)(Math.random() *1000));
                        System.out.println(user + "吃完饭了，准备回家");
                    } catch (InterruptedException e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                    } catch (BrokenBarrierException e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                    }
                }
            };
          es.execute(runnable);
        }
    }
}
```

```
2用户到达聚餐地点，当前已有1人到达
1用户到达聚餐地点，当前已有2人到达
3用户到达聚餐地点，当前已有3人到达
人员全部到齐，拍照留恋
人员全部到齐，开始吃饭
2吃完饭了，准备回家
1吃完饭了，准备回家
3吃完饭了，准备回家
```