线程实现

继承Thread

实现Runnable

实现Callnable，可以有返回值，可以抛出异常，需要用FutureTask转化，用futureTask的get（阻塞）方法获取返回值

线程关闭方法

运行完毕

发生异常

stop()不推荐用，资源没有释放

interrupt()中断一个线程，中断标志位置为true

interrupted()判定当前线程是否处于中断状态

static 方法interrupted()判定当前线程是否处于中断状态，中断标志位置为false

Thread使用方法

主线程调用子线程的interrupt方法

子线程循环判断isInterrupt，若flase，停止运行

Runnable使用方法

Thread.currentThread.方法进行

run和start

只调用run不开启线程

线程优先级 1–10，不可靠，有些系统不能用

守护线程

守护线程中finally不一定起作用

synchrized

volatile

使用场景：只要一个线程写，其他线程读

ThreadLocal线程变量

**Fork-Join框架**

分而治之

规模为N的问题，N<阈值，直接解决，N>阈值，分解为K个小规模问题

工作密取

ForkJoinPool

ForkJoinTack的子类 RecursiveTask有返回值时使用 RecursiveAction无返回值时使用（都是抽象类）

实现后compute方法

同步用法：有返回值

异步用法：无返回值

**ConutDownLatch**

一个线程等待其他现场完成后再执行，加强版join

countdown扣减 ,await等待

**CyclicBarrier**

让一组线程到达某个屏障，被阻塞，一直到组内最后一个线程达到屏障，屏障开放，所有阻塞线程运行

CountDownLatch和CyclicBarrier

CountDownLatch放行由第三者控制，CyclicBarrier放行由自身控制

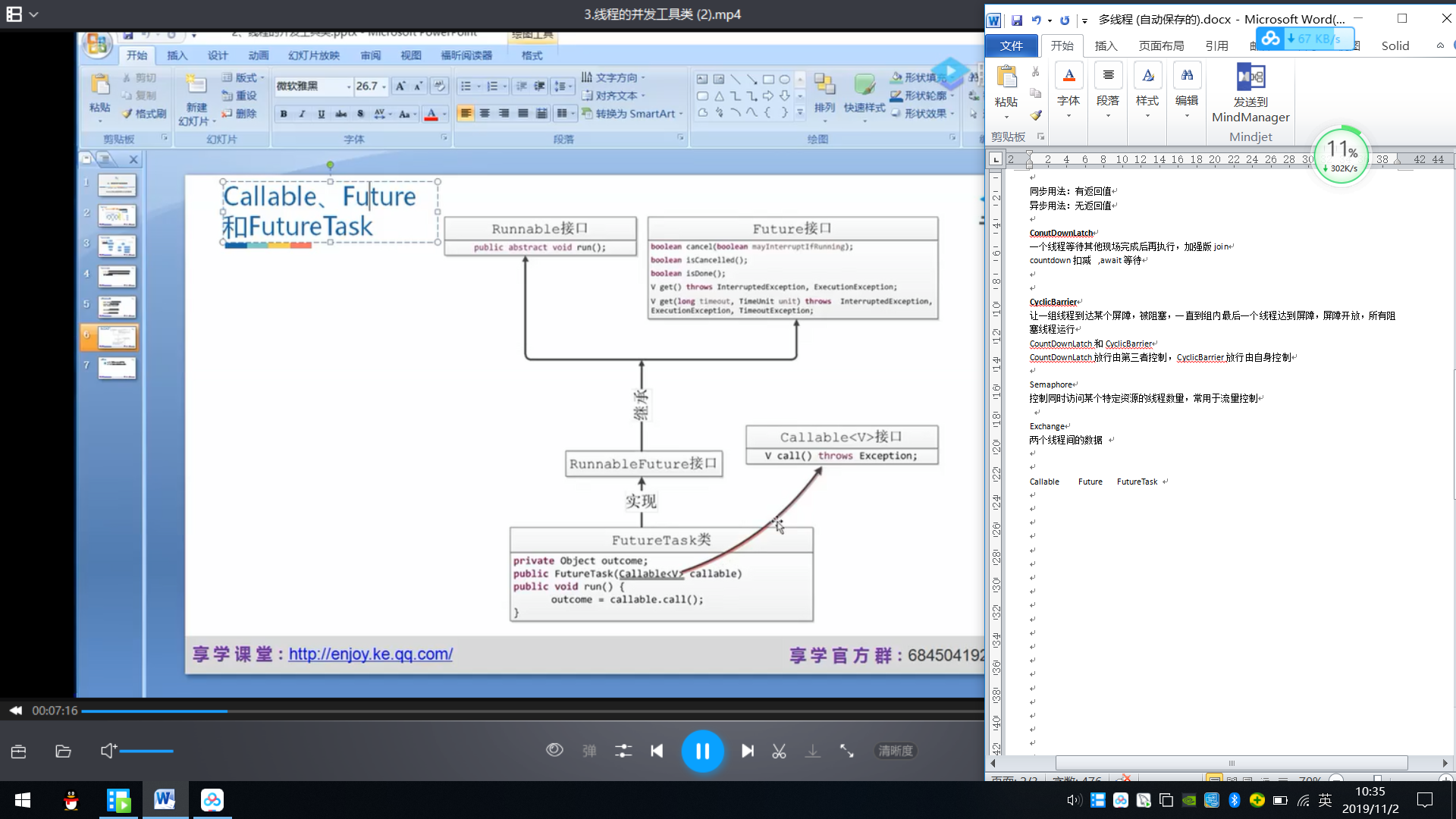
Semaphore

控制同时访问某个特定资源的线程数量，常用于流量控制

Exchange

两个线程间的数据

Callable Future FutureTask



isDone 结束，正常还是异常结束或者取消，返回true

isCancelled 任务提前取消，返回true

cancel（boolean）:终止任务

任务还没开始，返回false

任务已经开始 cancel(true)，尝试中断任务,中断成功，返回true, cancel（false）不会去中断

任务结束 返回false

原子操作CAS

Syn基于阻塞的锁的机制的问题

1. 被阻塞的线程优先级高问题
2. 拿到所得线程一直不释放锁问题
3. 大量的竞争，消耗CPU，同时带来死锁或其他安全问题

**CAS的原理**：利用了现代处理器都支持的CAS的指令，循环这个指令，直至成功为止。

**CAS的问题**

ABA问题

开销问题

只能保证一个共享变量的原子操作