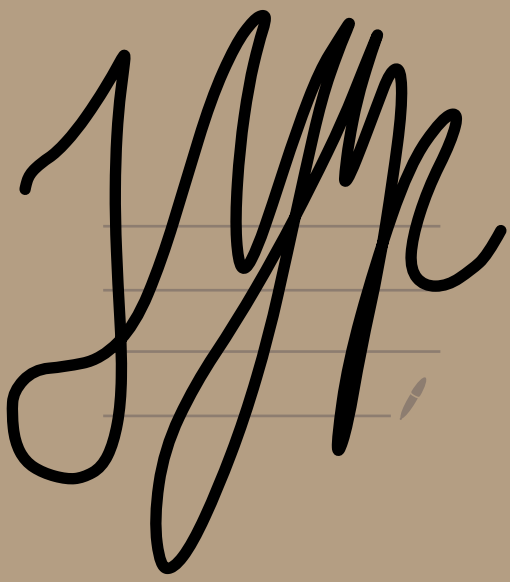
多线程



进程和线程的区别：

进程：操作系统分配资源的基本单位

线程：CPU调度的基本单位，执行的基本单位

同一进程内，线程资源共享。

ALU：计算

Registers：存储

PC：记录着位置

线程切换：越多的线程并不一定好。

协程

JVM级别的线程：

hotspot：JVM线程：内核线程1-1

synchronized(o) JDK1.2之前，由OS帮你管理线程调度。

重量级锁。

无锁 → 偏向锁 → 自旋锁 → 重量级锁。

{ 重量级 — OS帮
  else — 轻量级。

虚拟寄存器 虚拟PC | 自己管 自己

管理属于用户空间的线程

轻量级的线程 — 协程：纤程

多个T → K

用户级别的线程

- 短的计算 ✓
- IO ✗

syn：持有锁

互斥锁：同一个时刻只允许一个线程持有这把锁

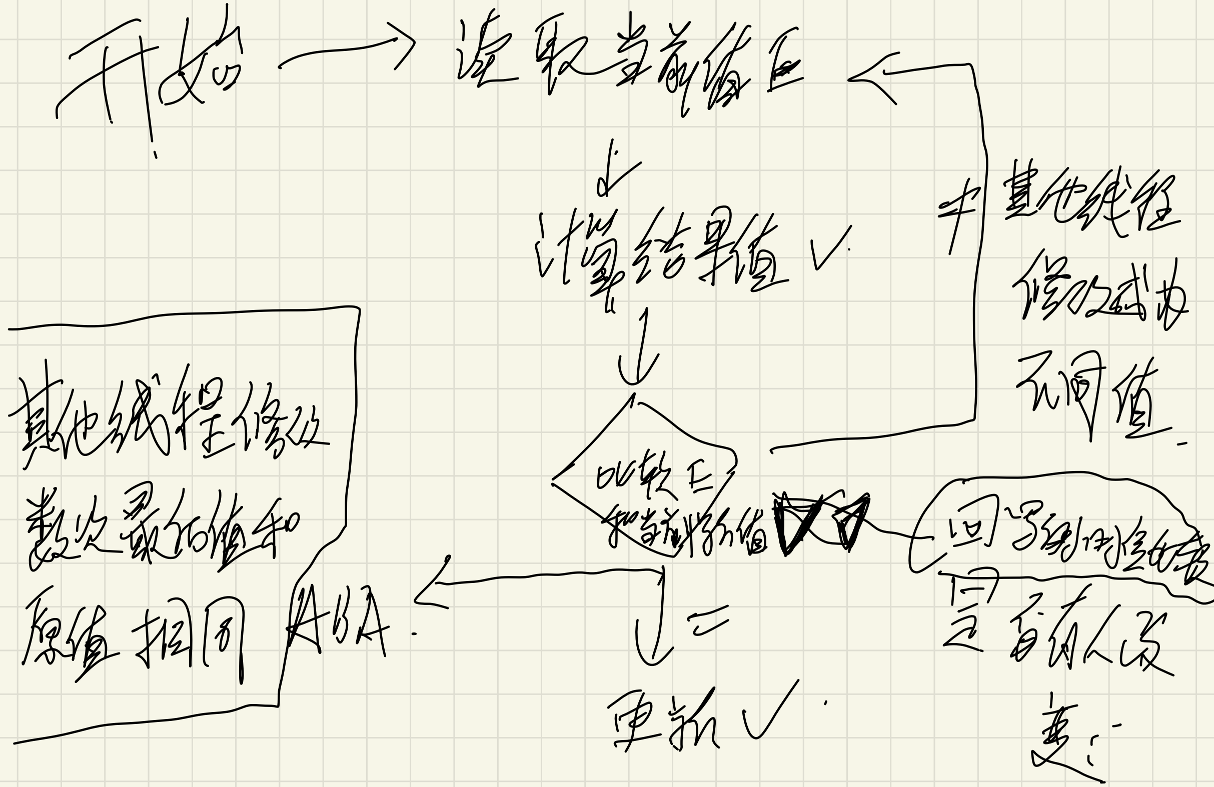
O.wait()等待 O.notify()唤醒

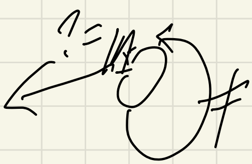
重锁 OS管

AtomicInteger 原子性的 incrementAndGet

CAS：compareAndSwapInt

- set
- exchange





自旋锁：乐观锁

ABA问题：

加版本！

表少比重量级锁效率高

消耗CPU

计算快，执行快。✓

1000个线程，cpu一直切换，不合适，每个任务很重 ✗

Compare and set（不具备原子性）

CAS最底层：Lock cmpxchg 汇编

锁总线　数据
指令
控制

JOL: Java object layout.

new → 对象，几部分构成

64 bit

new XX() {
- markword → 8字节
- 类型指针 Class pointer — 4字节 Object.
- 实例数据 instance data
- 对齐 padding — 8字节对齐 就是能被8整除.

Object header: markword
8+4 : object class

object没有成员变量.

所谓锁就是改变了markword.

markword里面到底是啥? 锁定信息
GC信息
hashcode 值.

偏向锁: 偏向 → 第一个线程

创建线程的四种方式

1. 继承Thread方法
2. 实现Runnable接口
3. Callable + FutureTask
4. 线程池方法创建
   Executors
   ThreadPoolExecutor

Synchronized 是可重入锁 → 同一线程可重入锁
可以进入不死锁

程序出现异常默认锁会释放

偏向锁
↓
自旋锁　10次
↓
OS锁

执行时间长，线程数多：系统锁
执行时间短，线程数少：自旋锁

锁的是对象
this 或 xx.class
锁定方法 非锁定方法同时执行

Synchronized (object)

— 不能用 String常量, Integer. Long

— "object".

饿汉模式: 直接new.

懒汉模式　先不new, 先判断是null 则. new.

!=null

sync

!=null.

Volite { 线程可见性
　　　　 禁止指令重排序

Atomicinteger

CAS: 无锁:

compare and set

cas(V, Expected, NewValue)

0　1　1　✓

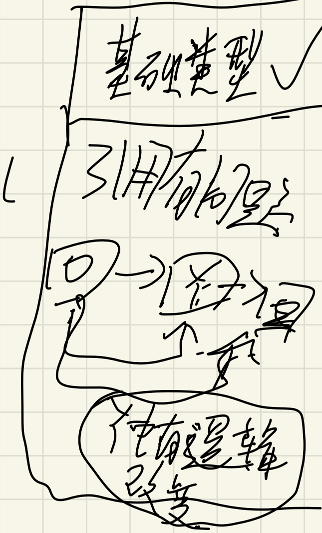
0　2　3　×

if V==E

V=new

otherwise try again or fail

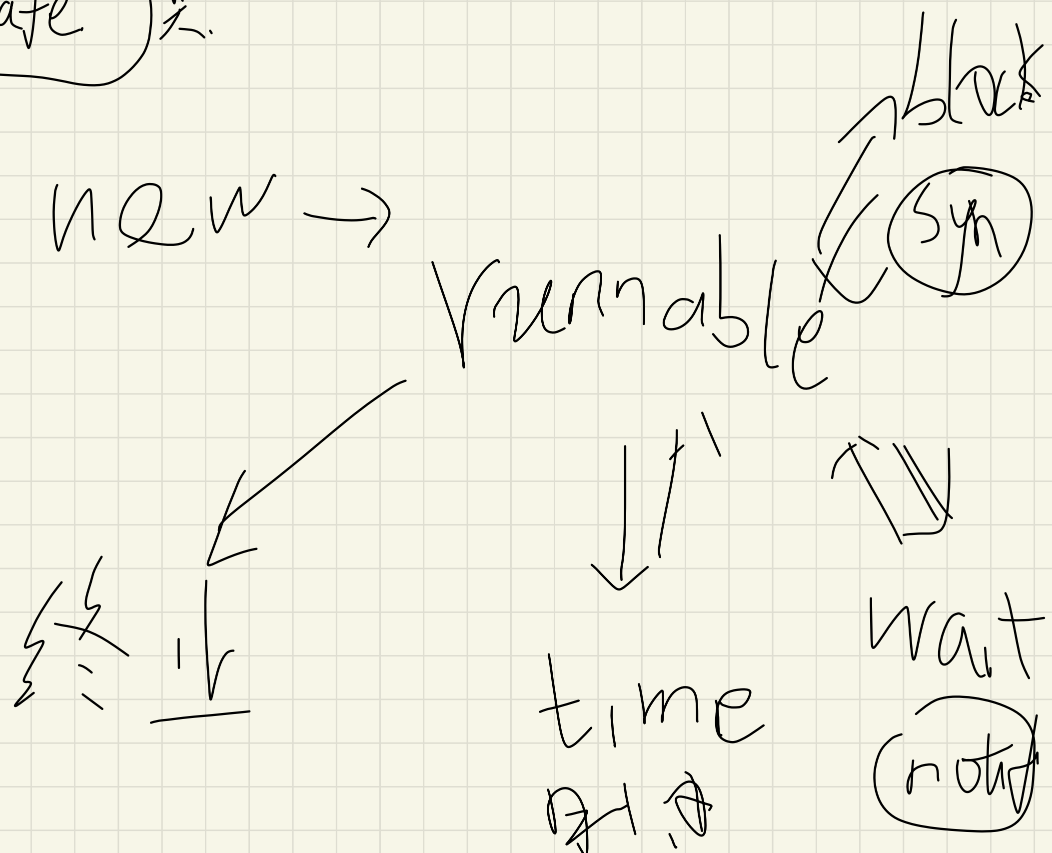
— CPU原语支持



ABA问题 版本号 version

—cas(version) AtomicStampedReference

Unsafe 类



timeout