- 锁
  - 自旋锁：忙等，不涉及线程休眠和调度
    - OSSpinLock
      - 原理：do while忙等，轮询锁的状态
      - 优点：由于避免了上下文调度，在critical section任务时间短时（非IO操作即定义为时间短），效率最高。
      - 问题：优先级反转问题。低优先级线程先拿到锁进入critical section时，高优先级线程请求锁。由于低优先级线程CPU时间片被抢占，导致无法释放锁，同时导致高优先级线程拿不到锁，无法进入critical section。占用大量cpu
  - 互斥锁：等待锁期间线程休眠，涉及线程唤醒。
    - pthread_mutex
      - 跨平台锁，使用C语言
    - NSLock
      - 封装pthread_mutex，属性为 PTHREAD_MUTEX_ERRORCHECK，它会损失一定性能换来错误提示
    - NSRecursiveLock
      - 递归锁定义：允许同一个线程在未释放其拥有的锁时反复对该锁进行加锁操作
      - 递归锁，封装pthread_mutex，属性为PTHREAD_MUTEX_RECURSIVE
    - NSCondition
      - 条件锁，封装了互斥锁和条件变量
    - 信号量
      - 调用SYS_futex进行线程休眠。主动让出时间片。内部与pthread_mutex类似
    - @synchronized
      - 内部函数：objc_sync_enter、objc_sync_exit
      - 内部原理：为传入对象分配递归锁
      - 使用结构体：typedef struct SyncData {
id object;//传入对象
recursive_mutex_t mutex;//互斥锁
struct SyncData* nextData;//链表结构
int threadCount;//使用锁的线程数量
} SyncData;
      - 特点
        - 1、入参object为nil，代码不安全
        - 2、入参object被释放，内存上分配了新对象，锁仍然可以使用。
        - 3、性能最差，但是调用最简单