声明：面试是对自我审视的一种过程，面试题和iOS程序员本身技术水平没任何关联，无论你能否全部答出，都不要对自己产生任何正面或消极的评价！

腾讯一面：http://www.jianshu.com/p/0e9e7486e1a7

腾讯二面：http://www.jianshu.com/p/dd17bdcff9f7

腾讯三面问题预览：

1.OC你了解的锁有哪些？在你回答基础上进行二次提问；
追问一：自旋和互斥对比？
追问二：用C/OC/C++，任选其一，实现自旋或互斥？口述即可！

2.内存泄漏可能会出现的几种原因，聊聊你的看法？
追问一：非OC对象如何处理？
追问二：若常用框架出现内存泄漏如何处理？

3.容错处理你们一般是注意哪些？

4.项目开始容错处理没做？如何防止拦截潜在的崩溃？

1.OC你了解的锁有哪些？在你回答基础上进行二次提问；

什么是锁?

在计算机科学中，锁是一种同步机制，用于在存在多线程的环境中实施对资源的访问限制。你可以理解成它用于排除并发的一种策略！

在iOS中，锁分为递归锁、条件锁、分布式锁、一般锁（根据NSLock类里面的分类进行划分）。

2.常用的有以下几种:

1.@synchronized 关键字加锁

2. NSLock 对象锁

3. NSCondition

4. NSConditionLock 条件锁

5. NSRecursiveLock 递归锁

6. pthread_mutex 互斥锁（C语言）

7. dispatch_semaphore 信号量实现加锁（GCD）

8. OSSpinLock

9.pthread_rwlock

10.POSIX Conditions

11.os_unfair_lock

追问一：自旋和互斥对比?

自旋锁和互斥锁

相同点：都能保证同一时间只有一个线程访问共享资源。都能保证线程安全。

不同点:

互斥锁：如果共享数据已经有其他线程加锁了，线程会进入休眠状态等待锁。一旦被访问的资源被解锁，则等待资源的线程会被唤醒。

自旋锁：如果共享数据已经有其他线程加锁了，线程会以死循环的方式等待锁，一旦被访问的资源被解锁，则等待资源的线程会立即执行。

自旋锁的效率高于互斥锁。

使用自旋锁时要注意：

由于自旋时不释放CPU，因而持有自旋锁的线程应该尽快释放自旋锁，否则等待该自旋锁的线程会一直在哪里自旋，这就会浪费CPU时间。

持有自旋锁的线程在sleep之前应该释放自旋锁以便其他可以获得该自旋锁。内核编程中，如果持有自旋锁的代码sleep了就可能导致整个系统挂起。

使用任何锁都需要消耗系统资源（内存资源和CPU时间），这种资源消耗可以分为两类：

1.建立锁所需要的资源

2.当线程被阻塞时所需要的资源

追问二：用C/OC/C++，任选其一，实现自旋或互斥？口述即可！

cpp实现：

```cpp
#pragma once

#include <atomic>

class spin_lock {
private:
    std::atomic<bool> flag = ATOMIC_VAR_INIT(false);
public:
    spin_lock() = default;
    spin_lock(const spin_lock&) = delete;
    spin_lock& operator=(const spin_lock&) = delete;
    void lock(){   //acquire spin lock
        bool expected = false;
        while(!flag.compare_exchange_strong(expected, true))
            expected = false;
    }
    void unlock(){   //release spin lock
        flag.store(false);
    }
};
```


两种锁的加锁原理：

互斥锁：线程会从sleep（加锁）——>running（解锁），过程中有上下文的切换，cpu的抢占，信号的发送等开销。

自旋锁：线程一直是running(加锁——>解锁)，死循环检测锁的标志位，机制不复杂。

2.内存泄漏可能会出现的几种原因，聊聊你的看法？

第一种可能：第三方框架不当使用；

第二种可能：block循环引用；

第三种可能：delegate循环引用；

第四种可能：NSTimer循环引用

第五种可能：非OC对象内存处理

第六种可能：地图类处理

第七种可能：大次数循环内存暴涨

追问一：非OC对象如何处理?

非OC对象，其需要手动执行释放操作例：CGImageRelease(ref)，否则会造成大量的内存泄漏导致程序崩溃。

其他的对于CoreFoundation框架下的某些对象或变量需要手动释放、C语言代码中的malloc等需要对应free。

追问二：地图类内存若泄漏，如何处理?

地图是比较耗费App内存的，因此在根据文档实现某地图相关功能的同时，需要注意内存的正确释放，大体需要注意的有需在使用完毕时将地图、代理等滞空为nil；

注意地图中标注（大头针）的复用，并且在使用完毕时清空标注数组等。

3.容错处理你们一般是注意哪些?

在团队协作开发当中，由于每个团队成员的水平不一，很难控制代码的质量，保证代码的健壮性，经常会发生由于后台返回异常数据造成app崩溃闪退的情况，为了避免这样的情况项目中做一些容错处理，显得格外重要，极大程度上降低了因为数据容错不到位产生崩溃闪退的概率。

例如：

1.字典

2.数组；

3.野指针；

4.NSNull

等~

4.如果项目开始容错处理没做？如何防止拦截潜在的崩溃?

例:

1、category给类添加方法用来替换掉原本存在潜在崩溃的方法。

2、利用runtime方法交换技术，将系统方法替换成类添加的新方法。

3、利用异常的捕获来防止程序的崩溃，并且进行相应的处理。

总结:

1、不要过分相信服务器返回的数据会永远的正确。

2、在对数据处理上，要进行容错处理，进行相应判断之后再处理数据，这是一个良好的编程习惯。