解决iOS中常见的几种Crash

最近正好有空，总结一下以前项目中遇到的几种常见崩溃，并且在无侵入的情况下解决这些崩溃。

常见的崩溃类型

1.数组越界，nil值初始化导致的崩溃。

2.对字典插入nil值，或者读取NSNULL导致的崩溃。

3.字符串的截取越界导致的崩溃。

4.doesNotRecognizeSelector导致的崩溃。

5.子线程初始化UIView导致的崩溃。

6.KVO的重复添加、删除，或者忘了删除导致的崩溃。

对于以上崩溃，如果是新产品，或者是代码全部是可以修改的情况下，可以通过检查代码的方式来解决这些问题，但是对于一些老的项目，或者集成了第三方库的项目呢，老项目修改动作太大，第三方库只能指望更新来解决了，这都不是我们需要的方式，所以这种情况下，就需要我们的无侵入解决方案了。

说到无侵入，大家首先想到的肯定是Method Swizzling，没错，下面我们就利用Method Swizzling来解决以上崩溃。

先写上大家熟悉的方法交换的代码

```
+ (void)exchangeInstanceMethod:(Class)anClass originMethodSel:(SEL)originSE

    MethodorigIndex =class_getInstanceMethod(anClass, originSEL);

    MethodoverrideIndex =class_getInstanceMethod(anClass, replaceSEL);

    if(!origIndex || !overrideIndex) {

        return;
```

```
    }
    method_exchangeImplementations(origIndex, overrideIndex);
}
+ (void)exchangeClassMethod:(Class)anClass originMethodSel:(SEL)originSEL r
{
    MethodorigIndex =class_getClassMethod(anClass, originSEL);
    MethodoverrideIndex =class_getClassMethod(anClass, replaceSEL);
    if(!origIndex || !overrideIndex) {
        return;
    }
    method_exchangeImplementations(origIndex, overrideIndex);
}
```

然后说一下整体的思路

1.数组越界，nil值初始化导致的崩溃。

这种情况要解决很容易，网上很多这方面的文章，就是通过方法交换原NSArray的objectAtIndex，然后加一层索引判断就够了，这里就不多做介绍了。

需要特别注意的就是NSArray初始化的时候有空值的情况。

2.对字典插入nil值，或者读取NSNULL导致的崩溃。

这种崩溃的解决思路和NSArray一样，单独提出来，只是因为经常出现服务器返回的数据是null的，如果在使用时不对数据类型进行判断的话，就会出现NSNULL类型与所需要的类型不一样，导致崩溃 。所以这是一个需要注意的地方。

3.字符串的截取越界导致的崩溃。

崩溃处理方式与上面一样。

说到这，再提个概念类簇，只有明白什么是类簇，才能知道为什么我们做方法交换的时候不直接使用[self class]，我们上面要修改的几个类NSArray，NSDictionary，NSString都是类簇，它们的Class比较多，需要尽量多的枚举所有可能的Class。

4.doesNotRecognizeSelector导致的崩溃

doesNotRecognizeSelector也是一种比较常见的崩溃，相信大家都了解iOS的消息转发机制的几个步骤了，我们就不再重复说明了，接下来我们再来看看如何选择我们的实现。

1）动态决议

需要动态实现这个未知的方法，而且需要考虑到参数问题，比较麻烦，不采用。

2）备用接收

将这个未知的方法转交给其它对象，结果还是需要实现这个未知方法，同上，不采用。

3）消息转发

完整的消息转发，将未知的方法打包成一个NSInvocation转交给别的对象，但我们在forwardInvocation:完全可以不实现任何真的转发，就可以拦截掉这次的转发，所以采用这种方式最合适。

在实现的时候，我们在MethodSignature方法里，将方法签名指向一个我们自定义的类的方法，并且拿到签名，返回给系统。在forwardInvocation里，不做任何实现就可以了。

5.子线程初始化UIView导致的崩溃。

这种情况也比较简单，就是把UIView的初始化方法及addSubview的方法交换

一下，然后判断一下当前线程是不是主线程，如果不是主线程，那么GCD到主线程里实现就行了。

6.KVO的重复添加、删除，或者忘了删除导致的崩溃。

KVO出现最多的崩溃可能就是忘记删除或者重复删除了，要解决这个问题，最简单的实现就是记录每次添加的observer和keyPath，所以我在addObserver:forKeyPath:options:context:这个方法里新建了一个字典，用来记录observer和keyPath。

下面需要解决的就是什么时候去调用的问题，我首先想到的是在dealloc时去判断是否添加了KVO的监视，但是当我直接交换了dealloc方法后发现，这个方法调用的太多了，并不适合直接交换，不然整个程序都会卡顿起来，需要找一个时机，于是我又修改为在添加监视的时候去交换dealloc，这次成功了，这样可以减少对不必要的类进行方法交换，同时提高效率。

唯一需要注意的是在ARC的情况下，不能直接@selector(dealloc)来做方法交换，需要变形一下NSSelectorFromString(@"dealloc")，这样才能做方法交换。

[完整Demo地址]