iOS笔记

1.静态库和动态库

介绍

- 什么是库?
  库简单理解就是一系列代码的组合，分为静态库和动态库
- 静态库
  程序在编译阶段就会被拷贝进入可执行文件的库，即为静态库
- 动态库
  程序在运行阶段会被动态加载到内存，供程序调用的库，即为动态库

1.1 iOS库的形式

- 静态库
  .a和.framework后缀的形式

- 动态库
  .dylib和.framework后缀的形式
  注意：系统的framework为动态库，我们自己创建的framework为静态库

- .a与.framework区别

  1. .a是一个纯二进制文件，.framework中除了二进制文件以外还有资源文件
  2. .a文件不能直接使用，需要.h文件配合，.framework可以直接使用

1.2 动态库的作用

- 应用插件化

  > 每一个功能点都是一个动态库，在用户想使用某个功能的时候让其从网络下载，然后手动加载动态库，实现功能的的插件化
  > 虽然技术上来说这种动态更新是可行的，但是对于AppStore上上架的app是不可以的。iOS8之后虽然可以上传含有动态库的app，但是苹果不仅需要你动态库和app的签名一致，而且苹果会在你上架的时候再经过一次AppStore的签名。所以你想在线更新动态库，首先你得有苹果APPStore私钥，而这个基本不可能。除非你的应用不需要通过AppStore上架，比如企业内部的应用，通过企业证书发布，那么就可以实现应用插件化在线更新动态库了。

2.OC的类可以多重继承么?可以实现多个接口么?Category是什么?重写一个类的方式用继承好还是分类好?为什么?

OC的类不可以多重继承;可以实现多个接口，通过实现多个接口可以完成C++的多重继承;Category是类别，一般情况用分类好，用Category去重写类的方法，仅对本Category有效，不会影响到其他类与原有类的关系。

3.#import 跟#include 又什么区别，@class呢, #import<> 跟 #import""又什么区别?

#import是Objective-C导入头文件的关键字，#include是C/C++导入头文件的关键字,使用#import头文件会自动只导入一次，不会重复导入，相当于#include和#pragma once;@class告诉编译器某个类的声明，当执行时，才去查看类的实现文件，可以解决头文件的相互包含;#import<>用来包含系统的头文件，#import""用来包含用户头文件。

4.属性readwrite，readonly，assign，retain，copy，nonatomic 各是什么作用，在那种情况下用?

1. readwrite 是可读可写特性;需要生成getter方法和setter方法时

2. readonly 是只读特性 只会生成getter方法 不会生成setter方法 ;不希望属性在类外改变

3. assign 是赋值特性，setter方法将传入参数赋值给实例变量;仅设置变量时;

4. retain 表示持有特性，setter方法将传入参数先保留，再赋值，传入参数的retaincount会+1;

5. copy 表示赋值特性，setter方法将传入对象复制一份;需要完全一份新的变量时。

6. nonatomic 非原子操作，决定编译器生成的setter getter是否是原子操作，atomic表示多线程安全，一般使用nonatomic

5.写一个setter方法用于完成@property (nonatomic,retain)NSString *name,写一个setter方法用于完成@property(nonatomic，copy)NSString *name

```
-(void)setName:(NSString*)str
{
 [str retain];
 [name release];
 name = str;
 }
-(void)setName:(NSString*)str
{
 id t =[str copy];
 [name release];
  name =t;
}
```

6.对于语句NSString*obj = [[NSData alloc] init]; obj在编译时和运行时分别时什么类型的对象?

编译时是NSString的类型;运行时是NSData类型的对象

7.Xcode的Other Linker Flags简介

摘要：在ios开发过程中，有时候会用到第三方的静态库(.a文件)，然后导入后发现编译正常但运行时会出现selector not recognized的错误，从而导致app闪退。接着仔细阅读库文件的说明文档，你可能会在文档中发现诸如在Other Linker Flags中加入-ObjC或者-all_load这样的解决方法。 那么，Other Linker Flags到底是用来干什么的呢？还有-ObjC和-all_load到底发挥了什么作用呢？

7.1链接器

一个程序从简单易读的代码到可执行文件往往要经历以下步骤：

```
源代码 > 预处理器 > 编译器 > 汇编器 > 机器码 > 链接器 > 可执行文件
```

源文件经过一系列处理以后，会生成对应的.obj文件，然后一个项目必然会有许多.obj文件，并且这些文件之间会有各种各样的联系，例如函数调用。链接器做的事就是把这些目标文件和所用的一些库链接在一起形成一个完整的可执行文件。

7.2为什么会闪退

Objective-C的链接器并不会为每个方法建立链接表，而是仅仅为类建立了链接表。这样的话，如果静态库中定义了已存在的一个类的分类，链接器就会以为这个类已经存在，不会把分类和核心类的代码合起来。这样的话，在最后的可执行文件中，就会缺少分类里的代码，这样函数调用就失败了。

7.3解决方法

解决方法在背景那块我就提到了，就是在Other Linker Flags里加上所需的参数，用到的参数一般有以下3个：

-Objc
简单说来，加了这个参数后，链接器就会把静态库中所有的Objective-C类和分类都加载到最后的可执行文件中，虽然这样可能会因为加载了很多不必要的文件而导致可执行文件变大，但是这个参数很好地解决了我们所遇到的问题。但是事实真的是这样的吗?
当静态库中只有分类而没有类的时候，-ObjC参数就会失效了。这时候，就需要使用-all_load或者-force_load了

-all_load
-all_load会让链接器把所有找到的目标文件都加载到可执行文件中，但是千万不要随便使用这个参数！假如你使用了不止一个静态库文件，然后又使用了这个参数，那么你很有可能会遇到ld: duplicate symbol错误，因为不同的库文件里面可能会有相同的目标文件，所以建议在遇到-ObjC失效的情况下使用-force_load参数。
-force_load
-force_load所做的事情跟-all_load其实是一样的，但是-force_load需要指定要进行全部加载的库文件的路径，这样的话，你就只是完全加载了一个库文件，不影响其余库文件的按需加载。

8 TextFiled的文本框属性security

属性为NO时，系统键盘会出现小地球
监控TextFiled获取焦点，当获取焦点时，设置security为YES；监控键盘的状态，当键盘弹出时，设置security为NO。
TextField设置security在NO和YES之前切换时，游标不会随之移动，另外设置为NO时，此时如果该TextField仍然为第一相应者，文字的颜色和字体会发生变化 该问题属于IOS系统bug，各位有好的解决方案可以共享。