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。

9 常见的OC的数据类型有那些， 和C的基本数据类型有什么区别?如：NSInteger和int

OC的数据类型有NSString，NSNumber，NSArray，NSMutableArray，NSData等等，这些都是class，创建后便是对象，而C语言的基本数据类型int，只是一定字节的内存空间，用于存放数值;NSInteger是基本数据类型，并不是NSNumber的子类，当然也不是NSObject的子类。NSInteger是基本数据类型Int或者Long的别名(NSInteger的定义typedef long NSInteger)，它的区别在于，NSInteger会根据系统是32位还是64位来决定是本身是int还是Long。

10 id声明的对象有什么特性?

id声明的对象具有运行时的特性，即可以指向任意类型的objcetive-c的对象;

11 Objective-C如何对内存管理的,说说你的看法和解决方法?

Objective-C的内存管理主要有三种方式ARC(自动内存计数)、手动内存计数、内存池。
1. (Garbage Collection)自动内存计数：这种方式和java类似，在你的程序的执行过程中。始终有一个高人在背后准确地帮你收拾垃圾，你不用考虑它什么时候开始工作，怎样工作。你只需要明白，我申请了一段内存空间，当我不再使用从而这段内存成为垃圾的时候，我就彻底的把它忘记掉，反正那个高人会帮我收拾垃圾。遗憾的是，那个高人需要消耗一定的资源，在携带设备里面，资源是紧俏商品所以iPhone不支持这个功能。所以“Garbage Collection”不是本入门指南的范围，对“Garbage Collection”内部机制感兴趣的同学可以参考一些其他的资料，不过说老实话“Garbage Collection”不大适合适初学者研究。

解决：通过alloc – initial方式创建的，创建后引用计数+1，此后每retain一次引用计数+1，那么在程序中做相应次数的release就好了.
2. (Reference Counted)手动内存计数：就是说，从一段内存被申请之后，就存在一个变量用于保存这段内存被使用的次数，我们暂时把它称为计数器，当计数器变为0的时候，那么就是释放这段内存的时候。比如说，当在程序A里面一段内存被成功申请完成之后，那么这个计数器就从0变成1(我们把这个过程叫做alloc)，然后程序B也需要使用这个内存，那么计数器就从1变成了2(我们把这个过程叫做retain)。紧接着程序A不再需要这段内存了，那么程序A就把这个计数器减1(我们把这个过程叫做release);程序B也不再需要这段内存的时候，那么也把计数器减1(这个过程还是release)。当系统(也就是Foundation)发现这个计数器变 成员了0，那么就会调用内存回收程序把这段内存回收(我们把这个过程叫做dealloc)。顺便提一句，如果没有Foundation，那么维护计数器，释放内存等等工作需要你手工来完成。
解决:一般是由类的静态方法创建的，函数名中不会出现alloc或init字样，如[NSString string]和[NSArray arrayWithObject:]，创建后引用计数+0，在函数出栈后释放，即相当于一个栈上的局部变量. 当然也可以通过retain延长对象的生存期.

3. (NSAutoRealeasePool)内存池：可以通过创建和释放内存池控制内存申请和回收的时机.
解决:是由autorelease加入系统内存池，内存池是可以嵌套的，每个内存池都需要有一个创建释放对，就像main函数中写的一样. 使用也很简单，比如[[[NSString alloc]initialWithFormat:@”Hey you!”] autorelease]，即将一个NSString对象加入到最内层的系统内存池，当我们释放这个内存池时，其中的对象都会被释放.

12 原子(atomic)跟非原子(non-atomic)属性有什么区别?

1. atomic提供多线程安全。是防止在写未完成的时候被另外一个线程读取，造成数据错误
2. non-atomic:在自己管理内存的环境中，解析的访问器保留并自动释放返回的值，如果指定了 nonatomic ，那么访问器只是简单地返回这个值。

13 类别的作用?继承和类别在实现中有何区别

category 可以在不获悉，不改变原来代码的情况下往里面添加新的方法，只能添加，不能删除修改，并且如果类别和原来类中的方法产生名称冲突，则类别将覆盖原来的方法，因为类别具有更高的优先级。

类别主要有3个作用：
1.将类的实现分散到多个不同文件或多个不同框架中。
2.创建对私有方法的前向引用。
3.向对象添加非正式协议。

继承可以增加，修改或者删除方法，并且可以增加属性。

14 类别和类扩展的区别

category和extensions的不同在于 后者可以添加属性。另外后者添加的方法是必须要实现的。

extensions可以认为是一个私有的Category。

15 oc中的协议和java中的接口概念有何不同?

OC中的代理有2层含义，官方定义为 formal和informal protocol。前者和Java接口一样。
informal protocol中的方法属于设计模式考虑范畴，不是必须实现的，但是如果有实现，就会改变类的属性。

16 KVO和KVC

KVC:键 — 值编码是一种间接访问对象的属性使用字符串来标识属性，而不是通过调用存取方法，直接或通过实例变量访问的机制。
KVO:键值观察机制，他提供了观察某一属性变化的方法，极大的简化了代码。

17 代理的作用

代理的目的是改变或传递控制链。允许一个类在某些特定时刻通知到其他类，而不需要获取到那些类的指针。可以减少框架复杂度。
另外一点，代理可以理解为java中的回调监听机制的一种类似。

18 我们说的oc是动态运行时语言是什么意思

主要是将数据类型的确定由编译时，推迟到了运行时。

19 通知和协议的不同

通知在发出以后，发出方不会考虑接收方在收到通知以后的处理；协议跟接收方的当前业务有关联

20 frame和bounds有什么不同

frame指的是：该view在父view坐标系统中的位置和大小。(参照点是父亲的坐标系统)
bounds指的是：该view在本身坐标系统中 的位置和大小。(参照点是本身坐标系统)

21 OC的垃圾回收机制

OC2.0有Garbage collection，但是iOS平台不提供。
一般我们了解的objective-c对于内存管理都是手动操作的，但是也有自动释放池。
但是差了大部分资料，貌似不要和arc机制搞混就好了。

22 block

Objective-C是对C语言的扩展，block的实现是基于指针和函数指针
__block是一种特殊类型，使用该关键字声明的局部变量，可以被block所改变，并且其在原函数中的值会被改变。
使用block和使用delegate完成委托模式有什么优点
首先要了解什么是委托模式，委托模式在iOS中大量应用，其在设计模式中是适配器模式中的对象适配器，Objective-C中使用id类型指向一切对象，使委托模式更为简洁。了解委托模式的细节：

iOS设计模式——委托模式
使用block实现委托模式，其优点是回调的block代码块定义在委托对象函数内部，使代码更为紧凑；
适配对象不再需要实现具体某个protocol，代码更为简洁。

23 iOS私有方法和私有变量

首先，OC是一门动态定型（dynamicaly typed)语言，它是动态传递消息机制，所有的方法都是函数调用（有时甚至连系统调用（syscalls）也是如此），在运行时可以允许根据字符串名字来访问方法和类，还可以动态连接和添加类。
本身的机制特点让它并不存在真正意义上的私有。因为在类中创建了该方法之后，在别的类中只要import这个类，实际上就会导入h文件和m文件，所以你不论是写在h文件还是m文件，都是可以强制调用到的。
只是你把方法写在m文件中不在h文件声明，这样的情况在arc环境中，编译器会爆黄提醒你，但是还是会编译通过。
因为OC在编译阶段可以调用任何函数，甚至是这个函数并未实现，（在真正运行的时候才会通过函数的名去对应的函数来调用它），更何况你这个方法在别的类中已经实现过了。
只是写在m文件中，它并不提醒你有这个方法（想调用它的前提是你知道这个隐藏的方法名和参数，然后通过 performSelector 系列方法或者用 NSInvocation 调用,或者用 objc_msgSend 函数（传递消息机制）或者直接拿到方法对应的 IMP类似函数指针，然后像 C 函数那样直接调用）。

再说变量：
@private 定义私有变量，一般来说，设置这样的变量就是私有变量了，可是也不纯粹，因为私有变量只有声明此变量的类本身才能调用。
但是呢，学过OC语法的都知道KVC（key-value-coding），它提供一种机制来间接访问对象的属性，它的存在就打破了类的封装性，强制性访问类的属性。
（不过前提还是得知道你这个属性名），像上面的，如果你把属性写到m文件的Class Extension也和私有变量一样只能通过kvc的方式去修改。
总结一下：
不论是方法还是属性，就算它是私有的，如果你想强制性的去调用的话，死抠方法还是能给你修改到属性或者调用到方法的。不过，从理论层面上来讲，方法不存在私有，而变量存在私有。
还有一种普遍的说法是:OC里面只有类方法和实例方法两种，所有的实例变量默认都是私有的，所有实例方法默认都是共有的。

24 Object-C有多继承吗？没有的话用什么代替?

多继承在这里是用protocol 委托代理 来实现的；oc本身不支持类的多继承

25 Objective-C堆和栈的区别

管理方式：对于栈来讲，是由编译器自动管理，无需我们手工控制；对于堆来说，释放工作由程序员控制，容易产生memory leak。

申请大小：

栈：在Windows下,栈是向低地址扩展的数据结构，是一块连续的内存的区域。这句话的意思是栈顶的地址和栈的最大容量是系统预先规定好的，在 WINDOWS下，栈的大小是2M（也有的说是1M，总之是一个编译时就确定的常数），如果申请的空间超过栈的剩余空间时，将提示overflow。因 此，能从栈获得的空间较小。

堆：堆是向高地址扩展的数据结构，是不连续的内存区域。这是由于系统是用链表来存储的空闲内存地址的，自然是不连续的，而链表的遍历方向是由低地址向高地址。堆的大小受限于计算机系统中有效的虚拟内存。由此可见，堆获得的空间比较灵活，也比较大。

碎片问题：对于堆来讲，频繁的new/delete势必会造成内存空间的不连续，从而造成大量的碎片，使程序效率降低。对于栈来讲，则不会存在这个问题，因为栈是先进后出的队列，他们是如此的一一对应，以至于永远都不可能有一个内存块从栈中间弹出

分配方式：堆都是动态分配的，没有静态分配的堆。栈有2种分配方式：静态分配和动态分配。静态分配是编译器完成的，比如局部变量的分配。动态分配由alloca函数进行分配，但是栈的动态分配和堆是不同的，他的动态分配是由编译器进行释放，无需我们手工实现。

分配效率：栈是机器系统提供的数据结构，计算机会在底层对栈提供支持：分配专门的寄存器存放栈的地址，压栈出栈都有专门的指令执行，这就决定了栈的效率比较高。堆则是C/C++函数库提供的，它的机制是很复杂的。

26 TCP和UDP

TCP全称是Transmission Control Protocol，中文名为传输控制协议，它可以提供可靠的、面向连接的网络数据传递服务。传输控制协议主要包含下列任务和功能：

* 确保IP数据报的成功传递。

* 对程序发送的大块数据进行分段和重组。

* 确保正确排序及按顺序传递分段的数据。

* 通过计算校验和，进行传输数据的完整性检查。

TCP提供的是面向连接的、可靠的数据流传输，而UDP提供的是非面向连接的、不可靠的数据流传输。

简单的说，TCP注重数据安全，而UDP数据传输快点，但安全性一般

https://www.jianshu.com/p/1aa25be32bfa

27 沙盒机制

http://www.cocoachina.com/ios/20171124/21306.html

28 控制器加载视图的顺序

http://www.cocoachina.com/ios/20171124/21306.html

29 各种生命周期描述

https://www.jianshu.com/p/4a62c10a36f1

30 iOS UITableView性能优化

http://www.mamicode.com/info-detail-1125512.html