**[iOS](http://www.cnblogs.com/ios8/p/ios-oc-test.html)**[求职之](http://www.cnblogs.com/ios8/p/ios-oc-test.html)**[OC](http://www.cnblogs.com/ios8/p/ios-oc-test.html)**[面试题](http://www.cnblogs.com/ios8/p/ios-oc-test.html)

1、Objective-C的类可以多重继承么？可以采用多个协议么？

答：不可以多重继承，可以采用多个协议。

2、#import和#include的区别是什么？＃import<> 跟 #import""有什么区别？

#import能避免头文件被重复包含的问题:

1） 一般来说，导入objective c的头文件时用#import，包含c/c++头文件时用#include。

使用include要注意重复引用的问题：

class A，class B都引用了class C，class D若引用class A与class B,就会报重复引用的错误。

2）#import 确定一个文件只能被导入一次，这使你在递归包含中不会出现问题。

所以，#import比起#include的好处就是它避免了重复引用的问题。所以在OC中我们基本用的都是import。

＃import<> 包含iOS框架类库里的类，#import""包含项目里自定义的类。

3、Category是什么？扩展一个类的方式用继承好还是类目好？为什么？

答：Category是类目。用类目好，因为继承要满足a is a b的关系，而类目只需要满足a has a b的关系，局限性更小，你不用定义子类就能扩展一个类的功能，还能将类的定义分开放在不同的源文件里, 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。

4、延展是什么？作用是什么？

答：延展（extension）:在自己类的实现文件中添加类目来声明私有方法。

5、类实例（成员）变量的@protected,@private,@public声明各有什么含义？

@protected：受保护的，该实例变量只能在该类和其子类内访问，其他类内不能访问。

@private：私有的，该实例变量只能在该类内访问，其他类内不能访问。

@public：共有的，该实例变量谁都可以访问。

6、id声明的对象有什么特性？

Ø  没有 \* 号

Ø  动态数据类型

Ø  可以指向任何类的对象(设置是nil)，而不关心其具体类型

Ø  在运行时检查其具体类型

Ø  可以对其发送任何（存在的）消息

7、委托是什么？委托和委托方双方的property声明用什么属性？为什么？

委托：一个对象保存另外一个对象的引用，被引用的对象实现了事先确定的协议，该协议用于将引用对象中的变化通知给被引用对象。

委托和委托方双方的property声明属性都是assign而不是retain

为了避免循环引用造成的内存泄露。

      循环引用的问题这样理解：

       比如在main函数中创建了两个类的对象A和B，现在引用计数都是1。现在让A和B互相引用(A有一个属性是B对象，属性说明是retain；B有一个属性是A对象，属性说明是retain)，现在两个对象的引用计数都增加了1，都变成了2。

  现在执行[A release]; [B release]; 此时创建对象的main函数已经释放了自己对对象的所有权，但是此时A和B的引用计数都还是1，因为他们互相引用了。

  这时你发现A和B将无法释放，因为要想释放A必须先释放B，在B的dealloc方法中再释放A。同理，要想释放B必须先释放A，在A的dealloc方法中再释放B。所以这两个对象将一直存在在内存中而不释放。这就是所谓的循环引用的问题。要想解决这个问题，一般的方法可以将引用的属性设置为assign,而不是retain来处理。

8、浅拷贝和深拷贝区别是什么？

       浅层复制：只复制指向对象的指针，而不复制引用对象本身。

深层复制：复制引用对象本身。

意思就是说我有个A对象，复制一份后得到A\_copy对象后，对于浅复制来说，A和A\_copy指向的是同一个内存资源，复制的只不过是是一个指针，对象本身资源还是只有一份，那如果我们对A\_copy执行了修改操作,那么发现A引用的对象同样被修改，这其实违背了我们复制拷贝的一个思想。深复制就好理解了,内存中存在了两份独立对象本身。

用网上一哥们通俗的话将就是：

浅复制好比你和你的影子，你完蛋，你的影子也完蛋

深复制好比你和你的克隆人，你完蛋，你的克隆人还活着。

9、内存管理的几条原则是什么？按照默认法则，哪些关键字生成的对象需要手动释放？哪些情况下不需要手动释放，会直接进入自动释放池？

•       当使用new、alloc或copy方法创建一个对象时，该对象引用计数器为1。如果不需要使用该对象，可以向其发送release或autorelease消息，在其使用完毕时被销毁。

•       如果通过其他方法获取一个对象，则可以假设这个对象引用计数为1，并且被设置为autorelease，不需要对该对象进行清理，如果确实需要retain这个对象，则需要使用完毕后release。

•       如果retain了某个对象，需要release或autorelease该对象，保持retain方法和release方法使用次数相等。

使用new、alloc、copy关键字生成的对象和retain了的对象需要手动释放。设置为autorelease的对象不需要手动释放，会直接进入自动释放池。

10、怎样实现一个单例模式的类，给出思路，不写代码。

•       首先必须创建一个全局实例，通常存放在一个全局变量中,此全局变量设置为nil

•       提供工厂方法对该全局实例进行访问，检查该变量是否为nil，如果nil就创建一个新的实例，最后返回全局实例

•       全局变量的初始化在第一次调用工厂方法时会在+allocWithZone:中进行，所以需要重写该方法，防止通过标准的alloc方式创建新的实例

•       为了防止通过copy方法得到新的实例，需要实现-copyWithZone方法

•       只需在此方法中返回本身对象即可，引用计数也不需要进行改变，因为单例模式下的对象是不允许销毁的，所以也就不用保留

•       因为全局实例不允许释放，所以retain,release,autorelease方法均需重写

11、@class的作用是什么？

答：在头文件中， 一般只需要知道被引用的类的名称就可以了。 不需要知道其内部的实体变量和方法，所以在头文件中一般使用@class来声明这个名称是类的名称。 而在实现类里面，因为会用到这个引用类的内部的实体变量和方法，所以需要使用#import来包含这个被引用类的头文件。

•       @class的作用是告诉编译器，有这么一个类，用吧，没有问题

•       @class还可以解决循环依赖的问题，例如A.h导入了B.h，而B.h导入了A.h，每一个头文件的编译都要让对象先编译成功才行

•       使用@class就可以避免这种情况的发生

12、KVC是什么?KVO是什么?有什么特点？

•       KVC是键值编码，特点是通过指定表示要访问的属性名字的字符串标识符，可以进行类的属性读取和设置

•       KVO是键值观察，特点是利用键值观察可以注册成为一个对象的观察者，在该对象的某个属性变化时收到通知

13、MVC是什么？有什么特性？

–      MVC是一种设计模式，由模型、视图、控制器3部分组成。

–      模型：保存应用程序数据的类，处理业务逻辑的类

–      视图：窗口，控件和其他用户能看到的并且能交互的元素

–      控制器：将模型和试图绑定在一起，确定如何处理用户输入的类

14、定义属性时，什么情况使用copy、assign、retain？

使用assign: 对基础数据类型 （NSInteger，CGFloat）和C数据类型（int, float,double, char, 等等）

使用copy： 希望获得源对象的副本而不改变源对象内容时，对NSString

使用retain： 希望获得源对象的所有权时，对其他NSObject和其子类

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

assign用于简单数据类型，如NSInteger,double,bool,

retain和copy用于对象，

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

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

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；

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;

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。

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

16.id 声明的对象有什么特性？

答：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；

17.Objective-C如何对内存管理的,说说你的看法和解决方法?

答：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。

18.内存管理的几条原则时什么？

谁申请，谁释放

遵循Cocoa Touch的使用原则；

内存管理主要要避免“过早释放”和“内存泄漏”，对于“过早释放”需要注意@property设置特性时，一定要用对特性关键字，对于“内存泄漏”，一定要申请了要负责释放，要细心。

19.那些关键字生成的对象 需要手动释放？

答：关键字alloc 或new 生成的对象需要手动释放

20在和property结合的时候怎样有效的避免内存泄露？

答：设置正确的property属性，对于retain需要在合适的地方释放

21.如何对iOS设备进行性能测试?

Profile-> Instruments ->Time Profiler

22.Object－c的类可以多重继承么？可以实现多个接口么？

答：Object-c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；

23.Category是什么？重写一个类的方式用继承好还是分类好？为什么？

答：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。

24.描述一下iOS SDK中如何实现MVC的开发模式

MVC是模型、试图、控制开发模式，对于iOS SDK，所有的View都是视图层的，它应该独立于模型层，由视图控制层来控制。所有的用户数据都是模型层，它应该独立于视图。所有的ViewController都是控制层，由它负责控制视图，访问模型数据

25. Object C中创建线程的方法是什么？如果在主线程中执行代码，方法是什么？如果想延时执行代码、方法又是什么？

线程创建有三种方法：使用NSThread创建、使用 GCD的dispatch、使用子类化的NSOperation,然后将其加入NSOperationQueue;在主线程执行代码，方法是 performSelectorOnMainThread，如果想延时执行代码可以用performSelector:onThread:withObject:waitUntilDone

26、iPhone5 的屏幕分辨率大小为  1136＊ 640  ?

答：屏幕分辨率：用于量度位图图像内数据量多少的一个参数。通常表示成ppi（每英寸像素Pixel per inch）。屏幕物理尺寸不变，分辨率越高，每单位面积内包含的细节（像素点）越多。

27、struct strA {      int a;     float b;   char c;  } expA;

   printf("%ld",sizeof(expA));     输出结果为  12  ？

       该问题涉及编译器的“内存对齐”问题：

现代计算机中内存空间都是按照byte（字节）划分的，从理论上讲似乎对任何类型的变量的访问可以从任何地址开始，但实际情况是在访问特定变量的时候经常在特定的内存地址访问，这就需要各类型数据按照一定的规则在空间上排列，而不是顺序的一个接一个的排放，这就是对齐。

对齐的作用和原因：各个硬件平台对存储空间的处理上有很大的不同。一些平台对某些特定类型的数据只能从某些特定地址开始存取。其他平台可能没有这种情况， 但是最常见的是如果不按照适合其平台的要求对数据存放进行对齐，会在存取效率上带来损失。比如有些平台每次读都是从偶地址开始，如果一个int型（假设为 32位）如果存放在偶地址开始的地方，那么一个读周期就可以读出，而如果存放在奇地址开始的地方，就可能会需要2个读周期，并对两次读出的结果的高低 字节进行拼凑才能得到该int数据。显然在读取效率上下降很多。这也是空间和时间的博弈。

通常，我们写程序的时候，不需要考虑对齐问题。编译器会替我们选择适合目标平台的对齐策略。当然，我们也可以通知给编译器传递预编译指令而改变对指定数据的对齐方法。

但是，正因为我们一般不需要关心这个问题，所以因为编辑器对数据存放做了对齐，而我们不了解的话，常常会对一些问题感到迷惑。最常见的就是struct数据结构的sizeof结果，出乎意料。

       对于结构体来说，按成员中所占字节最大的是float类型，占用4个字节，一共有3个成员，所以总的占用字节为：4＊ 3 ＝ 12.

       可通过编译器命令来设定：

      #progma pack (2)

28、@property语法中readonly/readwrite，atomic/nonatomic的作用，@dynamic的作用？

      @Property：Objective-C语言关键词，与@synthesize配对使用。xcode4.5以及以后的版本，@synthesize可以省略。

功能：让编译器自动编写一对与数据成员同名的方法声明来省去读写方法的声明。

声明property的语法为：

@property (参数1,参数2) 类型 名字;

如：@property(nonatomic,retain) UIWindow \*window;

其中参数主要分为三类：

读写属性： （readwrite/readonly）

setter语意：（assign/retain/copy）

原子性： （atomicity/nonatomic）

各参数意义如下：

readwrite：同时产生setter\getter方法

readonly：只产生简单的getter,没有setter。

assign：默认类型,setter方法直接赋值，而不进行retain操作

retain：setter方法对参数进行release旧值，再retain新值。

copy：setter方法进行Copy操作，与retain一样

atomic：原子性，它没有一个如果你没有对原子性进行一个声明（atomic or nonatomic），那么系统会默认你选择的是atomic。

原子性就是说一个操作不可以被中途cpu暂停然后调度, 即不能被中断, 要不就执行完, 要不就不执行. 如果一个操作是原子性的,那么在多线程环境下, 就不会出现变量被修改等奇怪的问题。原子操作就是不可再分的操作，在多线程程序中原子操作是一个非常重要的概念，它常常用来实现一些同步机制，同时也是一些常见的多线程Bug的源头。当然，原子性的变量在执行效率上要低些。

关于异步与同步：并非同步就是不好，我们通常需要同时进行多个操作，这时使用异步，而对于程序来说，一般就是使用多线程，然而我们很多时候需要在多个线程间访问共享的数据，这个时候又需要同步来保证数据的准确性或访问的先后次序。当有多个线程需要访问到同一个数据时，OC中，我们可以使用@synchronized（变量）来对该变量进行加锁（加锁的目的常常是为了同步或保证原子操作）。

nonatomic：非原子性，是直接从内存中取数值，因为它是从内存中取得数据，它并没有一个加锁的保护来用于cpu中的寄存器计算Value，它只是单纯的从内存地址中，当前的内存存储的数据结果来进行使用。在多线环境下可提高性能，但无法保证数据同步。

29、OSI（Open System Interconnection）开放式系统互联参考模型 把网络协议从逻辑上分为了7层，试列举常见的应用层协议。

       注意问的是应用层协议，有些同学直接答了七层模型。

       在开放系统互连(OSI)模型中的最高层，为应用程序提供服务以保证通信，但不是进行通信的应用程序本身。

Telnet协议是TCP/IP协议族中的一员，是Internet远程登陆服务的标准协议和主要方式。它为用户提供了在本地计算机上完成远程主机工作的能力。

FTP文件传输协议是TCP/IP网络上两台计算机传送文件的协议，FTP是在TCP/IP网络和INTERNET上最早使用的协议之一，它属于网络协议组的应用层。

超文本传输协议 (HTTP-Hypertext transfer protocol) 是分布式，协作式，超媒体系统应用之间的通信协议。是万维网（world wide web）交换信息的基础。

SMTP（Simple MailTransfer Protocol）即简单邮件传输协议,它是一组用于由源地址到目的地址传送邮件的规则，由它来控制信件的中转方式，它帮助每台计算机在发送或中转信件时找到下一个目的地。

时间协议(TIME protocol)是一个在RFC 868内定义的网络协议。它用作提供机器可读的日期时间资讯。

DNS 是域名系统 (Domain NameSystem) 的缩写，是因特网的一项核心服务，它作为可以将域名和IP地址相互映射的一个分布式数据库。

SNMP(Simple Network ManagementProtocol,简单网络管理协议)的前身是简单网关监控协议(SGMP)，用来对通信线路进行管理。

TFTP（Trivial FileTransfer Protocol,简单文件传输协议）是TCP/IP协议族中的一个用来在客户机与服务器之间进行简单文件传输的协议，提供不复杂、开销不大的文件传输服务。端口号为69。

30、网络传输层协议中，基于TCP/IP协议和UDP/IP的连接有什么区别？

TCP：TransmissionControl Protocol 传输控制协议TCP是一种面向连接（连接导向）的、可靠的、基于字节流的运输层（Transport layer）通信协议，由IETF的RFC 793说明（specified）。

UDP 是User DatagramProtocol的简称， 中文名是用户数据包协议，是OSI 参考模型中一种无连接的传输层协议，提供面向事务的简单不可靠信息传送服务，IETF RFC 768是UDP的正式规范。

面向连接：是指通信双方在通信时，要事先建立一条通信线路，其有三个过程：建立连接、使用连接和释放连接。电话系统是一个面向连接的模式，拨号、通话、挂机；TCP协议就是一种面向连接的协议。

面向无连接：是指通信双方不需要事先建立一条通信线路，而是把每个带有目的地址的包（报文分组）送到线路上，由系统自主选定路线进行传输。邮政系统是一个无连接的模式，天罗地网式的选择路线，天女散花式的传播形式；IP、UDP协议就是一种无连接协议。

31、简述MVC模式中M、V、C分别指代什么及发挥的作用？

MVC开始是存在于Desktop（桌面）程序中的，M是指数据模型，V是指用户界面，C则是控制器，使用MVC的目的是将M和V的实现代码分离。C存在的目的则是确保M和V的同步，一旦M改变，V应该同步更新。

视图是用户看到并与之交互的界面，视图没有真正的处理发生，不管这些数据是联机存储的还是一个雇员列表，作为视图来讲，它只是作为一种输出数据并允许用户操纵的方式。

模型表示企业数据和业务规则，模型返回的数据是中立的，就是说模型与数据格式无关，这样一个模型能为多个视图提供数据，由于应用于模型的代码只需写一次就可以被多个视图重用，所以减少了代码的重复性。

控制器接受用户的输入并调用模型和视图去完成用户的需求，控制器本身不输出任何东西和做任何处理。它只是接收请求并决定调用哪个模型构件去处理请求，然后再确定用哪个视图来显示返回的数据。

32、声明@property的语法中，retain、copy、assign的含义及作用？试写出 @property中带retain和assign关键字，通过@synthesize自动生成的的合成存取方法（set、get方法）的实现代码。

**getter**分析：

@property (nonatomic, retain) test\*aTest;

@property (nonatomic, copy) test\*aTest;

等效代码：

-(void)aTest {

    return aTest;

}

＝＝＝＝＝＝＝＝＝＝ 貌似我是分割线 ＝＝＝＝＝＝＝＝＝＝＝

@property (retain) test\* aTest;

@property (copy) test\* aTest;

等效代码：

-(void)aTest

{

    [aTest  retain];

    return [aTest  autorelease];

}

**setter**分析：

@property (nonatomic, retain) test\*aTest;

@property (retain) test\* aTest;

等效于：

-(void)setaTest:(test \*)newaTest {

    if (aTest !=newaTest) {

       [aTest  release];

       aTest = [newaTest  retain];

    }

}

＝＝＝＝＝＝＝＝＝＝ 貌似我是分割线 ＝＝＝＝＝＝＝＝＝＝＝

@property (nonatomic, copy) test\*aTest;

@property (copy) test\* aTest;

等效于：

-(void)setaTest:(test \*)newaTest {

    if (aTest != newaTest){

       [aTest  release];

       aTest = [newaTest  copy];

    }

}

33、iOS中有哪些回调机制，并作简单的比较。

各种回调机制的比较：

1）目标动作对：当两个对象之间有比较紧密的关系时，如视图控制器与其下的某个视图。

2）代理：也叫委托，当某个对象收到多个事件，并要求同一个对象来处理所有事件时。委托机制依赖于某个协议定义的方法来发送消息。

3）通告机制：当需要多个对象或两个无关对象处理同一个事件时。

4）Block：适用于回调只发生一次的简单任务。

34、列出在编码中哪些编码习惯有助于提高代码质量、软件性能和健壮性，减少程序崩溃。

＃使用严格的命名规则（如匈牙利命名法）能够避免不必要的类型转换错误。

＃在编码前先设计好流程图或使用伪代码，清晰化整个设计意图。

＃对自己的代码进行严格的单元测试（unit testing）。

单元测试是指对软件中的最小可测试单元进行检查和验证。如C语言中单元指一个函数，Java里单元指一个类，图形化的软件中可以指一个窗口或一个菜单等。总的来说，单元就是人为规定的最小的被测功能模块。单元测试是在软件开发过程中要进行的最低级别的测试活动，软件的独立单元将在与程序的其他部分相隔离的情况下进行测试。

＃异常的处理

首先不要轻易使用异常的捕获，其次要尽可能捕获具体的异常。对于异常的处理最好能够采用封装的方式，大家统一使用。这样可以保证异常处理的一致性也可以保证当异常出现时性能的稳定。

# 使用内省的方法检查方法的输入

＃采用增量式的编程方式。

采用增量式编程和测试，会倾向于创建更小的方法和更具内聚性的类。你应该经常评估代码质量，并不时的进行许多小调整，而不是一次修改许多东西。在写了几行代码之后，就应该进行一次构建/测试。在没有得到反馈时，你不要走的太远。

＃使用工具（如Instrument）来帮助检查内存泄漏、过早释放内存、CPU使用效率等问题。

＃消除所有的编译警告，警告就是错误。

＃写防御性的代码，使用内省的方法检查传入的参数。

35、JSON中{ }代表\_\_\_\_\_，[ ]代表\_\_\_\_\_，试将下面的JSON串用OC对象表示出来：

{ "people": [

{ "firstName": "Brett","lastName":"McLaughlin", "email":"aaaa" },

{ "firstName": "Jason","lastName":"Hunter", "email": "bbbb"},

{ "firstName": "Elliotte","lastName":"Harold", "email": "cccc" }

]，

 “location”:”中华人民共和国”

}

JSON中{ }代表对象，数据结构为｛key1:value1, key2:value2, key3:…… ｝

[ ]代表数组，与其他语言中的数组类似。

//

@interface People: NSObject

@property(nonatomic, copy) NSString\* strFirstName;

@property(nonatomic, copy) NSString\* strLastName;

@property(nonatomic, copy) NSString\* strEmail;

@end

//

@interfaceJSonData : NSObject

@property(nonatomic, retain) NSMutableArray\* arrPeople;  // 存放People对象

@property(nonatomic, copy) NSString\* strLocation;

@end

36.  Object-C有多继承吗？没有的话用什么代替？

答：没有，cocoa 中所有的类都是NSObject 的子类，多继承在这里是用protocol 委托代理来实现的  
 ，ood的多态特性在obj-c中通过委托来实现。

37．bject-C有私有方法吗？私有变量呢？

objective-c – 类里面的方法只有两种, 静态方法和实例方法.

在类里面声名一个私有方法  
@interface Controller : NSObject

{ NSString \*something;

}  
+ (void)thisIsAStaticMethod;  
–(void)thisIsAnInstanceMethod;  
@end  
@interface Controller

(private)

-(void)thisIsAPrivateMethod;  
@end

@private可以用来修饰私有变量  
在Objective‐C中，所有实例变量默认都是私有的，所有实例方法默认都是公有的

38.  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：栈是向低地址扩展的数据结构，是一块连续的内存的区域

                   堆：是向高地址扩展的数据结构，是不连续的内存区域。

分配方式：堆都是动态分配的 ，动态分配由alloca函数进行分配

                   栈的动态分配由编译器进行释放，无需我们手工实现

39. kvc和kvo的区别？

kvc：键值编码，是一种间接访问对象的属性，使用字符串来标示属性

kvo：键值观察机制，提供了观察某一属性变化的方法

40. 线程和进程的区别？

答：主要不同的是操作系统资源管理方式

线程是一个进程中不同的执行路径，线程有自己的堆、局部变量

进程有独立的地址空间，一个进程死掉，整个线程就会死掉

线程是指进程内的一个执行单元,也是进程内的可调度实体.  
与进程的区别:  
(1)地址空间:进程内的一个执行单元;进程至少有一个线程;它们共享进程的地址空间;而进程有自己独立的地址空间;  
(2)资源拥有:进程是资源分配和拥有的单位,同一个进程内的线程共享进程的资源  
 (3)线程是处理器调度的基本单位,但进程不是.

（4）二者均可并发执行.

进程和线程都是由操作系统所体会的程序运行的基本单元，系统利用该基本单元实现系统对应用的并发性。进程和线程的区别在于：

简而言之,一个程序至少有一个进程,一个进程至少有一个线程.   
线程的划分尺度小于进程，使得多线程程序的并发性高。   
另外，进程在执行过程中拥有独立的内存单元，而多个线程共享内存，从而极大地提高了程序的运行效率。   
线程在执行过程中与进程还是有区别的。每个独立的线程有一个程序运行的入口、顺序执行序列和程序的出口。但是线程不能够独立执行，必须依存在应用程序中，由应用程序提供多个线程执行控制。   
从逻辑角度来看，多线程的意义在于一个应用程序中，有多个执行部分可以同时执行。但操作系统并没有将多个线程看做多个独立的应用，来实现进程的调度和管理以及资源分配。这就是进程和线程的重要区别。

进程是具有一定独立功能的程序关于某个数据集合上的一次运行活动,进程是系统进行资源分配和调度的一个独立单位.   
线程是进程的一个实体,是CPU调度和分派的基本单位,它是比进程更小的能独立运行的基本单位.线程自己基本上不拥有系统资源,只拥有一点在运行中必不可少的资源(如程序计数器,一组寄存器和栈),但是它可与同属一个进程的其他的线程共享进程所拥有的全部资源.   
一个线程可以创建和撤销另一个线程;同一个进程中的多个线程之间可以并发执行.

41.  #import和#include的区别，@class代表什么？

答：@class一般用于头文件中需要声明该类的某个实例变量的时候用到，在m文件中还是需要使用#import  
而#import比起#include的好处就是不会引起重复包含。

42. 类别的作用？

答：有时我们需要在一个已经定义好的类中增加一些方法，而不想去重写该类。可以使用类别对该类扩充新的方法。

注意：类别只能扩充方法，而不能扩充成员变量。

   代理的作用

委托代理（degegate），目的是改变和传递控制链

    顾名思义，把某个对象要做的事情委托给别的对象去做。那么别的对象就是这个对象的代理，代替它来打理要做的事。反映到程序中，首先要明确一个对象的委托方是哪个对象，委托所做的内容是什么。

委托机制是一种设计模式。

多态：子类的指针可以赋值给父类

43．链表和数组的区别在哪里?

二者都属于一种数据结构

从逻辑结构来看

1. 数组必须事先定义固定的长度（元素个数），不能适应数据动态地增减的情况。当数据增加时，可能超出原先定义的元素个数；当数据减少时，造成内存浪费；数组可以根据下标直接存取。

2. 链表动态地进行存储分配，可以适应数据动态地增减的情况，且可以方便地插入、删除数据项。（数组中插入、删除数据项时，需要移动其它数据项，非常繁琐）链表必须根据next指针找到下一个元素

从内存存储来看

1. (静态)数组从栈中分配空间, 对于程序员方便快速,但是自由度小

2. 链表从堆中分配空间, 自由度大但是申请管理比较麻烦

从上面的比较可以看出，如果需要快速访问数据，很少或不插入和删除元素，就应该用数组；相反， 如果需要经常插入和删除元素就需要用链表数据结构了。

44. main()

 { int a[5]={1,2,3,4,5};

   int \*ptr=(int \*)(&a+1);

  printf("%d,%d",\*(a+1),\*(ptr-1));

}

答：2,5   \*(a+1)就是a[1]，\*(ptr-1)就是a[4],执行结果是2，5

&a+1不是首地址+1，系统会认为加一个a数组的偏移，是偏移了一个数组的大小（本例是5个int）

    int \*ptr=(int \*)(&a+1);则ptr实际是&(a[5]),也就是a+5

原因如下：&a是数组指针，其类型为 int (\*)[5];

   而指针加1要根据指针类型加上一定的值，不同类型的指针+1之后增加的大小不同。

   a是长度为5的int数组指针，所以要加 5\*sizeof(int)，所以ptr实际是a[5]，

   但是prt与(&a+1)类型是不一样的(这点很重要)，所以prt-1只会减去sizeof(int\*)

    a,&a的地址是一样的，但意思不一样

    a是数组首地址，也就是a[0]的地址，&a是对象（数组）首地址，         a+1是数组下一元素的地址，即a[1],&a+1是下一个对象的地址，即a[5].

45. 写一个委托的interface

@protocol MyDelegate;

@interface MyClass: NSObject

{    id <MyDelegate> delegate; }

//委托方法@protocol MyDelegate- (void)didJobs:(NSArray \*)args;

@end

46. 写一个NSString类的实现

+(id)initWithCString:(const char \*)nullTerminatedCStringencoding:(NSStringEncoding)encoding;

+ (id)stringWithCString: (const char\*)nullTerminatedCString             encoding: (NSStringEncoding)encoding

 {   NSString \*obj;

     obj = [selfallocWithZone: NSDefaultMallocZone()];

     obj = [objinitWithCString: nullTerminatedCString encoding: encoding];

     returnAUTORELEASE(obj);

}

47. 关键字const有什么含意？修饰类呢?static的作用,用于类呢?还有extern c的作用

const意味着"只读"，下面的声明都是什么意思？

const int a;

int const a;

const int \*a;

int \* const a;

int const \* a const;

前两个的作用是一样，a是一个常整型数。第三个意味着a是一个指向常整型数的指针（也就是，整型数是不可修改的，但指针可以）。第四个意思a是一个指向整型数的常指针（也就是说，指针指向的整型数是可以修改的，但指针是不可修改的）。最后一个意味着a是一个指向常整型数的常指针（也就是说，指针指向的整型数是不可修改的，同时指针也是不可修改的）。

结论：·;关键字const的作用是为给读你代码的人传达非常有用的信息，实际上，声明一个参数为常量是为了告诉了用户这个参数的应用目的。如果 你曾花很多时间清理其它人留下的垃圾，你就会很快学会感谢这点多余的信息。（当然，懂得用const的程序员很少会留下的垃圾让别人来清 理的。）  ·; 通过给优化器一些附加的信息，使用关键字const也许能产生更紧凑的代码。  ·; 合理地使用关键字const可以使编译器很自然地保护那些不希望被改变的参数，防止其被无意的代码修改。简而言之，这样可以减少bug的出现。

（1）欲阻止一个变量被改变，可以使用 const 关键字。在定义该 const 变量时，通常需要对它进行初 始化，因为以后就没有机会再去改变它了；（2）对指针来说，可以指定指针本身为 const，也可以指定指针所指的数据为 const，或二者同时指 定为 const；

（3）在一个函数声明中，const 可以修饰形参，表明它是一个输入参数，在函数内部不能改变其值； （4）对于类的成员函数，若指定其为 const 类型，则表明其是一个常函数，不能修改类的成员变量； （5）对于类的成员函数，有时候必须指定其返回值为 const 类型，以使得其返回值不为“左值”。

关键字volatile有什么含意?并给出三个不同的例子。一个定义为volatile的变量是说这变量可能会被意想不到地改变，这样，编译器就不会去假设这个变量的值了。精确地说就是，优化器在用到这个变量时必须每次都小心地重新读取这个变量的值，而不是使用保存在寄存器里的备份。下面是volatile变量的几个例子：

· ;并行设备的硬件寄存器（如：状态寄存器）

· ; 一个中断服务子程序中会访问到的非自动变量(Non-automatic variables)

· ; 多线程应用中被几个任务共享的变量

· ;一个参数既可以是const还可以是volatile吗？解释为什么。

· ; 一个指针可以是volatile 吗？解释为什么。

下面是答案：

· ; 是的。一个例子是只读的状态寄存器。它是volatile因为它可能被意想不到地改变。它是const因为程序不应该试图去修改它。

·; 是的。尽管这并不很常见。一个例子是当一个中服务子程序修该一个指向一个buffer的指针时。

static关键字的作用：

（1）函数体内 static 变量的作用范围为该函数体，不同于 auto 变量，该变量的内存只被分配一次，因此其值在下次调用时仍维持上次的值； （2）在模块内的 static 全局变量可以被模块内所用函数访问，但不能被模块外其它函数访问；

（3）在模块内的 static 函数只可被这一模块内的其它函数调用，这个函数的使用范围被限制在声明 它的模块内；

（4）在类中的 static 成员变量属于整个类所拥有，对类的所有对象只有一份拷贝；

（5）在类中的 static 成员函数属于整个类所拥有，这个函数不接收 this 指针，因而只能访问类的static 成员变量。

extern "C"的作用：

（1）被 extern "C"限定的函数或变量是 extern 类型的；

 extern是 C/C++语言中表明函数和全局变量作用范围（可见性）的关键字，该关键字告诉编译器，其声明的函数和变量可以在本模块或其它模块中使用。

（2）被 extern "C"修饰的变量和函数是按照 C 语言方式编译和连接的；

extern "C"的惯用法

（1）在 C++中引用 C 语言中的函数和变量，在包含 C 语言头文件（假设为 cExample.h）时，需进 行下列处理：

 extern "C"  { #include "cExample.h"   }

而在 C语言的头文件中，对其外部函数只能指定为 extern 类型，C语言中不支持 extern "C"声明， 在.c 文件中包含了 extern "C"时会出现编译语法错误。

（2）在 C 中引用 C++语言中的函数和变量时，C++的头文件需添加 extern "C"，但是在 C 语言中不 能直接引用声明了 extern "C"的该头文件，应该仅将 C 文件中将 C++中定义的extern "C"函数声明为 extern 类型。

48.为什么标准头文件都有类似以下的结构？

 #ifndef \_\_INCvxWorksh

   #define \_\_INCvxWorksh

    #ifdef \_\_cplusplus

    extern "C" {

     #endif     /\*...\*/

      #ifdef \_\_cplusplus

      }

  #endif

  #endif /\* \_\_INCvxWorksh \*/

显然，头文件中的编译宏“#ifndef \_\_INCvxWorksh、#define \_\_INCvxWorksh、#endif” 的作用是防止该头文件被重复引用。

49. #import跟#include的区别,@class呢?

@class一般用于头文件中需要声明该类的某个实例变量的时候用到，在m文件中还是需要使用#import而#import比起#include的好处就是不会引起交叉编译。

50.线程与进程的区别和联系?

答：进程和线程都是由操作系统所体会的程序运行的基本单元，系统利用该基本单元实现系统对应用的并发性。

程和线程的主要差别在于它们是不同的操作系统资源管理方式。

进程有独立的地址空间，一个进程崩溃后，在保护模式下不会对其它进程产生影响，而线程只是一个进程中的不同执行路径。

线程有自己的堆栈和局部变量，但线程之间没有单独的地址空间，一个线程死掉就等于整个进程死掉，所以多进程的程序要比多线程的程序健壮，但在进程切换时，耗费资源较大，效率要差一些。但对于一些要求同时进行并且又要共享某些变量的并发操作，只能用线程，不能用进程。

51.列举几种进程的同步机制，并比较其优缺点。

答案：原子操作、信号量机制、自旋锁、管程、会合、分布式系统

进程之间通信的途径

答案：共享存储系统消息传递系统管道：以文件系统为基础

进程死锁的原因

答案：资源竞争及进程推进顺序非法

死锁的4个必要条件

答案：互斥、请求保持、不可剥夺、环路

死锁的处理

答案：鸵鸟策略、预防策略、避免策略、检测与解除死锁

52.什么是键-值,键路径是什么

答：模型的性质是通过一个简单的键（通常是个字符串）来指定的。视图和控制器通过键来查找相应的属性值。在一个给定的实体中，同一个属性的所有值具有相同的数据类型。键-值编码技术用于进行这样的查找—它是一种间接访问对象属性的机制。

键路径是一个由用点作分隔符的键组成的字符串，用于指定一个连接在一起的对象性质序列。第一个键的性质是由先前的性质决定的，接下来每个键的值也是相对于其前面的性质。键路径使您可以以独立于模型实现的方式指定相关对象的性质。通过键路径，您可以指定对象图中的一个任意深度的路径，使其指向相关对象的特定属性。

For example, the keypath address.streetwould get the value of the address property from thereceivingobject, and then determine the street property relative to the addressobject.

53.c和obj-c如何混用

1）obj-c的编译器处理后缀为m的文件时，可以识别obj-c和c的代码，处理mm文件可以识别obj-c,c,c++代码，但cpp文件必须只能用c/c++代码，而且cpp文件include的头文件中，也不能出现obj-c的代码，因为cpp只是cpp

2) 在mm文件中混用cpp直接使用即可，所以obj-c混cpp不是问题

3）在cpp中混用obj-c其实就是使用obj-c编写的模块是我们想要的。 如果模块以类实现，那么要按照cpp class的标准写类的定义，头文件中不能出现obj-c的东西，包括#import cocoa的。实现文件中，即类的实现代码中可以使用obj-c的东西，可以import,只是后缀是mm。 如果模块以函数实现，那么头文件要按c的格式声明函数，实现文件中，c++函数内部可以用obj-c，但后缀还是mm或m。

总结：只要cpp文件和cpp include的文件中不包含obj-c的东西就可以用了，cpp混用obj-c的关键是使用接口，而不能直接使用实现代码，实际上cpp混用的是obj-c编译后的o文件，这个东西其实是无差别的，所以可以用。obj-c的编译器支持cpp.

54.目标-动作机制

答：目标是动作消息的接收者。一个控件，或者更为常见的是它的单元，以插座变量（参见"插座变量"部分） 的形式保有其动作消息的目标。

    动作是控件发送给目标的消息，或者从目标的角度看，它是目标为了响应动作而实现的方法。程序需要某些机制来进行事件和指令的翻译。这个机制就是目标-动作机制。

55. cocoa touch框架

这些框架包括：

Core Animation

通过Core Animation，您就可以通过一个基于组合独立图层的简单的编程模型来创建丰富的用户体验。

Core Audio

Core Audio是播放，处理和录制音频的专业技术，能够轻松为您的应用程序添加强大的音频功能。

Core Data提供了一个面向对象的数据管理解决方案，它易于使用和理解，甚至可处理任何应用或大或小的数据模型。

   功能列表：框架分类

   下面是 Cocoa Touch 中一小部分可用的框架：

     音频和视频

       CoreAudio

       OpenAL

       MediaLibrary

       AVFoundation

     数据管理

       Core Data

       SQLite

     图形和动画

       CoreAnimation

       OpenGL ES

       Quartz 2D

      网络/li>

        Bonjour

        WebKit

        BSDSockets

      用户应用

        AddressBook

        CoreLocation

        MapKit

        StoreKit

56.objc的内存管理

答：如果您通过分配和初始化（比如[[MyClass alloc] init]）的方式来创建对象，您就拥有这个对象，需要负责该对象的释放。这个规则在使用NSObject的便利方法new 时也同样适用。

如果您拷贝一个对象，您也拥有拷贝得到的对象，需要负责该对象的释放。如果您保持一个对象，您就部分拥有这个对象，需要在不再使用时释放该对象。反过来，如果您从其它对象那里接收到一个对象，则您不拥有该对象，也不应该释放它（这个规则有少数的例外，在参考文档中有显式的说明）。

57.自动释放池是什么,如何工作？

答：当您向一个对象发送一个autorelease消息时，Cocoa就会将该对象的一个引用放入到最新的自动释放池。它仍然是个正当的对象，因此自动释放池定义的作用域内的其它对象可以向它发送消息。当程序执行到作用域结束的位置时，自动释放池就会被释放，池中的所有对象也就被释放。

1）ojc-c是通过一种"referring counting"(引用计数)的方式来管理内存的, 对象在开始分配内存(alloc)的时候引用计数为一,以后每当碰到有copy,retain的时候引用计数都会加一, 每当碰到release和autorelease的时候引用计数就会减一,如果此对象的计数变为了0, 就会被系统销毁.

2） NSAutoreleasePool就是用来做引用计数的管理工作的,这个东西一般不用你管的.

3）autorelease和release没什么区别,只是引用计数减一的时机不同而已,autorelease会在对象的使用真正结束的时候才做引用计数减一。

58.类工厂方法是什么？

答：类工厂方法的实现是为了向客户提供方便，它们将分配和初始化合在一个步骤中，返回被创建的对象，并进行自动释放处理。这些方法的形式是+ (type)className...（其中 className不包括任何前缀）。工厂方法可能不仅仅为了方便使用。它们不但可以将分配和初始化合在一起，还可以为初始化过程提供对象的分配信息。类工厂方法的另一个目的是使类（比如NSWorkspace）提供单件实例。虽然init...方法可以确认一 个类在每次程序运行过程只存在一个实例，但它需要首先分配一个“生的”实例，然后还必须释放该实例。工厂方法则可以避免为可能没有用的对象盲目分配内存。

59. 单件实例是什么？

Foundation和Application Kit 框架中的一些类只允许创建单件对象，即这些类在当前进程中的唯一实例。举例来说，NSFileManager和NSWorkspace 类在使用时都是基于进程进行单件对象的实例化。当向这些类请求实例的时候，它们会向您传递单一实例的一个引用，如果该实例还不存在，则首先进行实例的分配和初始化。单件对象充当控制中心的角色，负责指引或协调类的各种服务。如果类在概念上只有一个实例（比如NSWorkspace），就应该产生一个单件实例，而不是多个实例；如果将来某一天可能有多个实例，您可以使用单件实例机制，而不是工厂方法或函数。

60.动态绑定

—在运行时确定要调用的方法

动态绑定将调用方法的确定也推迟到运行时。在编译时，方法的调用并不和代码绑定在一起，只有在消实发送出来之后，才确定被调用的代码。通过动态类型和动态绑定技术，您的代码每次执行都可以得到不同的结果。运行时因子负责确定消息的接收者和被调用的方法。运行时的消息分发机制为动态绑定提供支持。当您向一个动态类型确定了的对象发送消息时，运行环境系统会通过接收者的isa指针定位对象的类，并以此为起点确定被调用的方法，方法和消息是动态绑定的。而且，您不必在Objective-C 代码中做任何工作，就可以自动获取动态绑定的好处。您在每次发送消息时，特别是当消息的接收者是动态类型已经确定的对象时，动态绑定就会例行而透明地发生。

61.obj-c的优缺点objc

优点：

1) Cateogies   2) Posing   3) 动态识别   4) 指标计算   5）弹性讯息传递   6) 不是一个过度复杂的 C 衍生语言   7) Objective-C 与 C++ 可混合编程

缺点:

1) 不支援命名空间   2)  不支持运算符重载  3） 不支持多重继承  4） 使用动态运行时类型，所有的方法都是函数调用，所以很多编译时优化方法都用不到。（如内联函数等），性能低劣。

62.sprintf,strcpy,memcpy使用上有什么要注意的地方？

答：strcpy是一个字符串拷贝的函数，它的函数原型为strcpy(char \*dst, const char \*src);将src开始的一段字符串拷贝到dst开始的内存中去，结束的标志符号为'\0'，由于拷贝的长度不是由我们自己控制的，所以这个字符串拷贝很容易出错。

具备字符串拷贝功能的函数有memcpy，这是一个内存拷贝函数，它的函数原型为memcpy(char \*dst,const char\* src, unsigned int len);将长度为len的一段内存，从src拷贝到dst中去，这个函数的长度可控。但是会有内存叠加的问题。

sprintf是格式化函数。将一段数据通过特定的格式，格式化到一个字符串缓冲区中去。sprintf格式化的函数的长度不可控，有可能格式化后的字符串会超出缓冲区的大小，造成溢出。