Object－c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？

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；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。

 2.#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。

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

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

4.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

1. - (void) setName:(NSString\*) str
2. {
3. [str retain];
4. [name release];
5. name = str;
6. }
7. - (void)setName:(NSString \*)str
8. {
9. id t = [str copy];
10. [name release];
11. name = t;
12. }

5.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

6.常见的object-c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

object-c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。

7.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；

8.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。

9.内存管理的几条原则时什么？按照默认法则.那些关键字生成的对象

需要手动释放？在和property结合的时候怎样有效的避免内存泄露？

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。

关键字alloc 或new 生成的对象需要手动释放；

设置正确的property属性，对于retain需要在合适的地方释放，

10.如何对iOS设备进行性能测试?

Profile-> Instruments ->Time Profiler

11.看下面的程序,第一个NSLog会输出什么？这时str的retainCount是多少？第二个和第三个呢？ 为什么

1. =======================================================
2. NSMutableArray\* ary = [[NSMutableArray array] retain];
3. NSString \*str = [NSString stringWithFormat:@"test"];
4. [strretain];
5. [aryaddObject:str];
6. NSLog(@"%@%d",str,[str retainCount]);
7. [strretain];
8. [strrelease];
9. [strrelease];
10. NSLog(@"%@%d",str,[str retainCount]);
11. [aryremoveAllObjects];
12. NSLog(@"%@%d",str,[str retainCount]);
13. =======================================================

str的retainCount创建+1，retain+1，加入数组自动+1 3

retain+1，release-1，release-1 2

数组删除所有对象，所有数组内的对象自动-1 1

12. 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:

13.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。

14. 浅复制和深复制的区别？ 答案：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。

15. 类别的作用？继承和类别在实现中有何区别？

答案：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。

16. 类别和类扩展的区别。

答案：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。

17.KVO是什么？

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。 很多情况下可以简化程序代码。apple文档其实给了一个很好的例子。 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。 具体用看到嗯哼用到过的一个地方是对于按钮点击变化状态的的监控。

18. 代理的作用？

答案：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。

19. 什么是推送消息？

推送则是服务器端主动push。

20. 关于多态性

答案：多态，子类指针可以赋值给父类。 这个题目其实可以出到一切面向对象语言中， 因此关于多态，继承和封装基本最好都有个自我意识的理解，也并非一定要把书上资料上写的能背出来。 最重要的是转化成自我理解。

21. 说说响应链

 答案： 事件响应链。包括点击事件，画面刷新事件等。在视图栈内从上至下，或者从下之上传播。 可以说点事件的分发，传递以及处理。具体可以去看下touch事件这块。因为问的太抽象化了 严重怀疑题目出到越后面就越笼统。

22. 方法和选择器有何不同？

答案：selector是一个方法的名字，method是一个组合体，包含了名字和实现.

23. NSOperation queue?

答案：存放NSOperation的集合类。 操作和操作队列，基本可以看成java中的线程和线程池的概念。用于处理ios多线程开发的问题。 网上部分资料提到一点是，虽然是queue，但是却并不是带有队列的概念，放入的操作并非是按照严格的先进现出。 这边又有个疑点是，对于队列来说，先进先出的概念是Afunc添加进队列，Bfunc紧跟着也进入队列，Afunc先执行这个是必然的， 但是Bfunc是等Afunc完全操作完以后，B才开始启动并且执行，因此队列的概念离乱上有点违背了多线程处理这个概念。

24. 是否在一个视图控制器中嵌入两个tableview控制器？

 答案：一个视图控制只提供了一个View视图，理论上一个tableViewController也不能放吧， 只能说可以嵌入一个tableview视图。当然，题目本身也有歧义，如果不是我们定性思维认为的UIViewController， 而是宏观的表示视图控制者，那我们倒是可以把其看成一个视图控制者，它可以控制多个视图控制器，比如TabbarController 那样的感觉。

1. 一个tableView是否可以关联两个不同的数据源？你会怎么处理？

 答案：首先我们从代码来看，数据源如何关联上的，其实是在数据源关联的代理方法里实现的。 因此我们并不关心如何去关联他，他怎么关联上，方法只是让我返回根据自己的需要去设置如相关的数据源。 因此，我觉得可以设置多个数据源啊，但是有个问题是，你这是想干嘛呢？想让列表如何显示，不同的数据源分区块显示？

27.给出委托方法的实例，并且说出UITableVIew的Data Source方法

答案：CocoaTouch框架中用到了大量委托，其中UITableViewDelegate就是委托机制的典型应用，是一个典型的使用委托来实现适配器模式，其中UITableViewDelegate协议是目标，tableview是适配器，实现UITableViewDelegate协议，并将自身设置为talbeview的delegate的对象，是被适配器，一般情况下该对象是UITableViewController。

UITableVIew的Data Source方法有- (NSInteger)tableView:(UITableView \*)tableView numberOfRowsInSection:(NSInteger)section;

* (UITableViewCell \*)tableView:(UITableView \*)tableView cellForRowAtIndexPath:(NSIndexPath \*)indexPath;

28.在应用中可以创建多少autorelease对象，是否有限制？ 答案：无

29.在iPhone应用中如何保存数据？

答案：有以下几种保存机制：1.通过web服务，保存在服务器上2.通过NSCoder固化机制，将对象保存在文件中3.通过SQlite或CoreData保存在文件数据库中.

30. 谈谈对Block 的理解？并写出一个使用Block执行UIVew动画？

答案：Block是可以获取其他函数局部变量的匿名函数，其不但方便开发，并且可以大幅提高应用的执行效率（多核心CPU可直接处理Block指令）

1. [UIView transitionWithView:self.view
2. duration:0.2
3. options:UIViewAnimationOptionTransitionFlipFromLeft
4. animations:^{ [[blueViewController view] removeFromSuperview]; [[self view] insertSubview:yellowViewController.view atIndex:0]; }

                completion:NULL];

31. 写出上面代码的Block的定义。

答案：

typedef void(^animations) (void);

typedef void(^completion) (BOOL finished);

32. 做过的项目是否涉及网络访问功能，使用什么对象完成网络功能？

答案：ASIHTTPRequest与NSURLConnection

33. 简单介绍下NSURLConnection类及[+ sendSynchronousRequest:returningResponse:error:](file:///Users/xunyanan/Library/Developer/Shared/Documentation/DocSets/com.apple.adc.documentation.AppleiOS6.0.iOSLibrary.docset/Contents/Resources/Documents/documentation/Cocoa/Reference/Foundation/Classes/NSURLConnection_Class/Reference/Reference.html#//apple_ref/occ/clm/NSURLConnection/sendSynchronousRequest:returningResponse:error:)与[– initWithRequest:delegate:](file:///Users/xunyanan/Library/Developer/Shared/Documentation/DocSets/com.apple.adc.documentation.AppleiOS6.0.iOSLibrary.docset/Contents/Resources/Documents/documentation/Cocoa/Reference/Foundation/Classes/NSURLConnection_Class/Reference/Reference.html#//apple_ref/occ/instm/NSURLConnection/initWithRequest:delegate:)两个方法的区别？

答案:  NSURLConnection主要用于网络访问，其中[+ sendSynchronousRequest:returningResponse:error:](file:///Users/xunyanan/Library/Developer/Shared/Documentation/DocSets/com.apple.adc.documentation.AppleiOS6.0.iOSLibrary.docset/Contents/Resources/Documents/documentation/Cocoa/Reference/Foundation/Classes/NSURLConnection_Class/Reference/Reference.html#//apple_ref/occ/clm/NSURLConnection/sendSynchronousRequest:returningResponse:error:)是同步访问数据，即当前线程会阻塞，并等待request的返回的response，而[– initWithRequest:delegate:](file:///Users/xunyanan/Library/Developer/Shared/Documentation/DocSets/com.apple.adc.documentation.AppleiOS6.0.iOSLibrary.docset/Contents/Resources/Documents/documentation/Cocoa/Reference/Foundation/Classes/NSURLConnection_Class/Reference/Reference.html#//apple_ref/occ/instm/NSURLConnection/initWithRequest:delegate:)使用的是异步加载，当其完成网络访问后，会通过delegate回到主线程，并其委托的对象。

34. 多线程是什么

多线程是个复杂的概念，按字面意思是同步完成多项任务，提高了资源的使用效率，从硬件、操作系统、应用软件不同的角度去看，多线程被赋予不同的内涵，对于硬件，现在市面上多数的CPU都是多核的，多核的CPU运算多线程更为出色；从操作系统角度，是多任务，现在用的主流操作系统都是多任务的，可以一边听歌、一边写博客；对于应用来说，多线程可以让应用有更快的回应，可以在网络下载时，同时响应用户的触摸操作。在iOS应用中，对多线程最初的理解，就是并发，它的含义是原来先做烧水，再摘菜，再炒菜的工作，会变成烧水的同时去摘菜，最后去炒菜。

35. iOS 中的多线程

iOS中的多线程，是Cocoa框架下的多线程，通过Cocoa的封装，可以让我们更为方便的使用线程，做过C++的同学可能会对线程有更多的理解，比如线程的创立，信号量、共享变量有认识，Cocoa框架下会方便很多，它对线程做了封装，有些封装，可以让我们创建的对象，本身便拥有线程，也就是线程的对象化抽象，从而减少我们的工程，提供程序的健壮性。

36. 在项目什么时候选择使用GCD，什么时候选择NSOperation？

项目中使用NSOperation的优点是NSOperation是对线程的高度抽象，在项目中使用它，会使项目的程序结构更好，子类化NSOperation的设计思路，是具有面向对象的优点（复用、封装），使得实现是多线程支持，而接口简单，建议在复杂项目中使用。

项目中使用GCD的优点是GCD本身非常简单、易用，对于不复杂的多线程操作，会节省代码量，而Block参数的使用，会是代码更为易读，建议在简单项目中使用。

37. **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。

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： 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。

39. 什么是键**-**值**,**键路径是什么

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。

40. 类别的作用

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。

类别的局限性

有两方面局限性：

(1)无法向类中添加新的实例变量，类别没有位置容纳实例变量。

(2)名称冲突，即当类别中的方法与原始类方法名称冲突时，类别具有更高的优先级。类别方法将完全取代初始方法从而无法再使用初始方法。

无法添加实例变量的局限可以使用字典对象解决

41. 代理的作用？

答案：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。

42. 通知和协议的不同之处？

 答案：协议有控制链(has-a)的关系，通知没有。

43. **HTTP**协议详解

HTTP是一个属于应用层的面向对象的协议，由于其简捷、快速的方式，适用于分布式超媒体信息系统。目前在WWW中使用的是HTTP/1.0的第六版，HTTP/1.1的规范化工作正在进行之中。

 http（超文本传输协议）是一个基于请求与响应模式的、无状态的、应用层的协议，常基于TCP的连接方式，HTTP1.1版本中给出一种持续连接的机制，绝大多数的Web开发，都是构建在HTTP协议之上的Web应用。 HTTP协议的主要特点可概括如下： 1.支持客户/服务器模式。 2.简单快速：客户向服务器请求服务时，只需传送请求方法和路径。请求方法常用的有GET、HEAD、POST。每种方法规定了客户与服务器联系的类型不同。由于HTTP协议简单，使得HTTP服务器的程序规模小，因而通信速度很快。 3.灵活：HTTP允许传输任意类型的数据对象。正在传输的类型由Content-Type加以标记。 4.无连接：无连接的含义是限制每次连接只处理一个请求。服务器处理完客户的请求，并收到客户的应答后，即断开连接。采用这种方式可以节省传输时间。 5.无状态：HTTP协议是无状态协议。无状态是指协议对于事务处理没有记忆能力。缺少状态意味着如果后续处理需要前面的信息，则它必须重传，这样可能导致每次连接传送的数据量增大。另一方面，在服务器不需要先前信息时它的应答就较快。

48. **URL**

HTTP URL (URL是一种特殊类型的URI是他的子类，包含了用于查找某个资源的足够的信息)的格式如下： **[Error! Hyperlink reference not valid.](http://host%5b%22%22port%5d%5babs_path/)**] http表示要通过HTTP协议来定位网络资源；host表示合法的Internet主机域名或者IP地址；port指定一个端口号，为空则使用缺省端口80；abs\_path指定请求资源的URI；如果URL中没有给出abs\_path，那么当它作为请求URI时，必须以“/”的形式给出，通常这个工作浏览器自动帮我们完成。

49. TCP/UDP区别联系

TCP---传输控制协议,提供的是面向连接、可靠的字节流服务。当客户和服务器彼此交换数据前，必须先在双方之间建立一个TCP连接，之后才能传输数据。TCP提供超时重发，丢弃重复数据，检验数据，流量控制等功能，保证数据能从一端传到另一端。

UDP---用户数据报协议，是一个简单的面向数据报的运输层协议。UDP不提供可靠性，它只是把应用程序传给IP层的数据报发送出去，但是并不能保证它们能到达目的地。由于UDP在传输数据报前不用在客户和服务器之间建立一个连接，且没有超时重发等机制，故而传输速度很快

TCP（Transmission Control Protocol，传输控制协议）是基于连接的协议，也就是说，在正式收发数据前，必须和对方建立可靠的连接。一个TCP连接必须要经过三次“对话”才能建立起来，我们来看看这三次对话的简单过程：1.主机A向主机B发出连接请求数据包；2.主机B向主机A发送同意连接和要求同步（同步就是两台主机一个在发送，一个在接收，协调工作）的数据包；3.主机A再发出一个数据包确认主机B的要求同步：“我现在就发，你接着吧！”，这是第三次对话。三次“对话”的目的是使数据包的发送和接收同步，经过三次“对话”之后，主机A才向主机B正式发送数据。

UDP（User Data Protocol，用户数据报协议）是与TCP相对应的协议。它是面向非连接的协议，它不与对方建立连接，而是直接就把数据包发送过去！  UDP适用于一次只传送少量数据、对可靠性要求不高的应用环境。

tcp协议和udp协议的差别

是否连接 面向连接 面向非连接

传输可靠性 可靠 不可靠

应用场合 传输大量数据 少量数据

速度 慢 快

50. **socket连接和http连接的区别**

简单说，你浏览的网页（网址以http://开头)都是http协议传输到你的浏览器的, 而http是基于socket之上的。socket是一套完成tcp，udp协议的接口。

HTTP协议：简单对象访问协议，对应于应用层  ，HTTP协议是基于TCP连接的

tcp协议：    对应于传输层

ip协议：     对应于网络层  TCP/IP是传输层协议，主要解决数据如何在网络中传输；而HTTP是应用层协议，主要解决如何包装数据。

Socket是对TCP/IP协议的封装，Socket本身并不是协议，而是一个调用接口（API），通过Socket，我们才能使用TCP/IP协议。

http连接：http连接就是所谓的短连接，即客户端向服务器端发送一次请求，服务器端响应后连接即会断掉；

socket连接：socket连接就是所谓的长连接，理论上客户端和服务器端一旦建立起连接将不会主动断掉；但是由于各种环境因素可能会是连接断开，比如说：服务器端或客户端主机down了，网络故障，或者两者之间长时间没有数据传输，网络防火墙可能会断开该连接以释放网络资源。所以当一个socket连接中没有数据的传输，那么为了维持连接需要发送心跳消息~~具体心跳消息格式是开发者自己定义的

我们已经知道网络中的进程是通过socket来通信的，那什么是socket呢？socket起源于Unix，而Unix/Linux基本哲学之一就是“一切皆文件”，都可以用“打开open –> 读写write/read –> 关闭close”模式来操作。我的理解就是Socket就是该模式的一个实现，socket即是一种特殊的文件，一些socket函数就是对其进行的操作（读/写IO、打开、关闭），这些函数我们在后面进行介绍。我们在传输数据时，可以只使用（传输层）TCP/IP协议，但是那样的话，如果没有应用层，便无法识别数据内容，如果想要使传输的数据有意义，则必须使用到应用层协议，应用层协议有很多，比如HTTP、FTP、TELNET等，也可以自己定义应用层协议。WEB使用HTTP协议作应用层协议，以封装HTTP文本信息，然后使用TCP/IP做传输层协议将它发到网络上。 1)Socket是一个针对TCP和UDP编程的接口，你可以借助它建立TCP连接等等。而TCP和UDP协议属于传输层 。   而http是个应用层的协议，它实际上也建立在TCP协议之上。

 (HTTP是轿车，提供了封装或者显示数据的具体形式；Socket是发动机，提供了网络通信的能力。)

 2）Socket是对TCP/IP协议的封装，Socket本身并不是协议，而是一个调用接口（API），通过Socket，我们才能使用TCP/IP协议。Socket的出现只是使得程序员更方便地使用TCP/IP协议栈而已，是对TCP/IP协议的抽象，从而形成了我们知道的一些最基本的函数接口。

51. **什么是TCP连接的三次握手**

第一次握手：客户端发送syn包(syn=j)到服务器，并进入SYN\_SEND状态，等待服务器确认； 第二次握手：服务器收到syn包，必须确认客户的SYN（ack=j+1），同时自己也发送一个SYN包（syn=k），即SYN+ACK包，此时服务器进入SYN\_RECV状态； 第三次握手：客户端收到服务器的SYN＋ACK包，向服务器发送确认包ACK(ack=k+1)，此包发送完毕，客户端和服务器进入ESTABLISHED状态，完成三次握手。

握手过程中传送的包里不包含数据，三次握手完毕后，客户端与服务器才正式开始传送数据。理想状态下，TCP连接一旦建立，在通信双方中的任何一方主动关闭连接之前，TCP 连接都将被一直保持下去。断开连接时服务器和客户端均可以主动发起断开TCP连接的请求，断开过程需要经过“四次握手”（过程就不细写了，就是服务器和客户端交互，最终确定断开）

52. **利用Socket建立网络连接的步骤**

建立Socket连接至少需要一对套接字，其中一个运行于客户端，称为ClientSocket ，另一个运行于服务器端，称为ServerSocket 。

套接字之间的连接过程分为三个步骤：服务器监听，客户端请求，连接确认。

1。服务器监听：服务器端套接字并不定位具体的客户端套接字，而是处于等待连接的状态，实时监控网络状态，等待客户端的连接请求。

2。客户端请求：指客户端的套接字提出连接请求，要连接的目标是服务器端的套接字。为此，客户端的套接字必须首先描述它要连接的服务器的套接字，指出服务器端套接字的地址和端口号，然后就向服务器端套接字提出连接请求。

3。连接确认：当服务器端套接字监听到或者说接收到客户端套接字的连接请求时，就响应客户端套接字的请求，建立一个新的线程，把服务器端套接字的描述发给客户端，一旦客户端确认了此描述，双方就正式建立连接。而服务器端套接字继续处于监听状态，继续接收其他客户端套接字的连接请求。