2019/09/20

58同城面试：

1. **进程和线程** <https://www.cnblogs.com/bsjl/p/7693029.html>
2. **进程和线程的区别：**

线程是进程内的一个执行单元，也是进程的可调度实体。

一个程序至少有一个进程，一个进程至少有一个线程。

根本区别：进程是操作系统资源分配的基本单位，而线程是任务调度和执行的基本单位。

1. 地址空间：线程是进程的一个执行单元，进程至少有一个线程，各线程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：进程是操作系统资源分配的基本单位，同一进程内的线程共享进程的资源。
3. 开销方面：每个进程都有独立的代码和数据空间，程序之间的切换会有较大的开销；线程可以看作是轻量级的进程，同类进程共享代码和数据空间，每个线程都有自己独立的运行栈和程序计数器，线程之间的切换开销小。
4. 二者均可并发执行。
5. **多线程的优势**
6. 提高程序的运行性能
7. 充分利用系统的处理能力，提高系统的资源利用率
8. 提高系统响应性，即线程可以在运行现有任务的情况下立即开始处理新的任务
9. **多线程通信**

多线程之间需要进行通信，线程的通信依赖共享的内存和线程方法的调用来实现。Java内存模型分为主内存和工作内存，通过内存之间的数据交换实现线程之间的通信；主动调用线程的wait()、notify()方法（等待唤醒机制）也可以实现线程之间的通信。

1. **多线程并发为什么不安全？如何解决？**

线程安全的定义：多个线程访问同一个对象时，如果不用考虑这些线程在运行时环境下的调度和交替执行，也不需要进行额外的同步，或者在调用方进行任何其他操作，调用这个对象的行为都可以获得正确的结果，那么这个对象就是线程安全的。

竞态条件：多线程同时对数据进行改变，读取到脏数据或写入错数据

重排序、有序性、可见性：线程之间不可见

1. **如何按照指定顺序执行多个线程？**

有三个线程T1，T2，T3，确保它们被顺序执行。在多线程中有多种方法让线程按特定顺序执行，你可以用线程类的join()方法在一个线程中启动另一个线程，另外一个线程完成该线程继续执行。为了确保三个线程的顺序你应该先启动最后一个(T3调用T2，T2调用T1)，这样T1就会先完成而T3最后完成。

1. **如何查看锁状态？**
2. **等待的线程是以什么机制查看锁状态？**

循环机制

1. **为什么你应该在循环中检查等待条件?**

处于等待状态的线程可能会收到错误警报和伪唤醒，如果不在循环中检查等待条件，程序就会在没有满足结束条件的情况下退出。因此，当一个等待线程醒来时，不能认为它原来的等待状态仍然是有效的，在notify()方法调用之后和等待线程醒来之前这段时间它可能会改变。这就是在循环中使用wait()方法效果更好的原因。

1. **线程的生命周期**

新建、就绪、运行、等待/阻塞/睡眠、终止

1. **进程有几种状态？**

就绪状态、执行状态、阻塞状态

**网络：**

1. **tcp三次握手、四次挥手**

<https://www.cnblogs.com/Andya/p/7272462.html>

1. **http协议的方法**

**get**：GET是最常用的方法，通常用于请求服务器发送某个资源

**head**：HEAD方法和GET方法的行为很类似，但是服务器在响应中只返回首部，不会返回实体的主体部分。这就允许客户端在未获取实际资源的情况下，对资源的首部进行检查

**put**：与GET从服务器读取文档想法，PUT方法会向服务器写入文档。PUT方法的语义就是让服务器用请求的主体部分来创建一个由所请求的URL命名的新闻的，或者替换已存在的URL

**post**：POST方法期初是用来向服务器输入数据的，实际上，通常会用它来支持HTML的表单。表单中填好的数据通常会被送给服务器，然后由服务器将其发送到它要去的地方。

Trace：客户端发起一个请求时，这个请求可能要穿过防火墙、代理、网关或者其他一些应用程序。每个中间节点都可能会修改原始的HTTP请求。TRACE方法允许客户端在最终将请求发送给服务器时，看看它变成什么样子了。

        TRACE请求会在目的服务器端发起一个“环回”诊断。行程最后一站的服务器会弹回一条TRACE响应，并在响应主体中携带它收到的原始请求报文。这样客户端就可以查看在所有中间HTTP应用程序组成的请求/响应链上，原始报文是否被毁坏，以及如何被毁坏或修改过。

**Options**：OPTIONS方法请求Web服务器告知其支持的各种功能。可以询问服务器通常支持哪些方法，或者对某些特殊资源支撑哪些方法。（有些服务器可能只支持对一些特殊类型的对象使用特定的操作）

**Delete**：DELETE方法所做的事情就是请服务器删除请求URL所指定的资源。但客户端应用程序无法保证删除操作一定会被执行。因为HTTP规范允许服务器在不通知客户端的情况下撤销请求。

**扩展方法**：

**GET和POST的区别**：

        1、GET请求的数据是放在HTTP包头中的，也就是URL之后，通常是像下面这样定义格式的：

login.action?name=hyddd&password=idontknow&verify=%E4%BD%E5%A5%BD

其中，以?来分隔URL和数据；以&来分隔参数；如果数据是英文或数字，原样发送；如果数据是中文或其它字符，则进行BASE64编码。

而Post是把提交的数据放在HTTP正文中的。

        2、GET提交的数据比较少，最多1024B，因为GET数据是附在URL之后的，而URL则会受到不同环境的限制的，比如说IE对其限制为2K+35，而POST可以传送更多的数据（理论上是没有限制的，但一般也会受不同的环境，如浏览器、操作系统、服务器处理能力等限制，IIS4可支持80KB，IIS5可支持100KB）。

        3、Post的安全性要比Get高，因为Get时，参数数据是明文传输的，参数直接暴露在url中，所以不能用来传递敏感信息。而且使用GET的话，还可能造成Cross-site request forgery攻击。而POST数据则可以加密的，但GET的速度可能会快些。

        4、get请求只能进行url编码，而post支持多种编码方式；get请求会浏览器主动cache，而post支持多种编码方式；get请求参数会被完整保留在浏览历史记录里，而post中的参数不会被保留。

        5、GET和POST本质上就是TCP链接，并无差别。但是由于HTTP的规定和浏览器/服务器的限制，导致他们在应用过程中体现出一些不同。

        6、GET产生一个TCP数据包；POST产生两个TCP数据包。对于GET方式的请求，浏览器会把http header和data一并发送出去，服务器响应200（返回数据）；而对于POST，浏览器先发送header，服务器响应100 continue，浏览器再发送data，服务器响应200 ok（返回数据）。

1. **http协议的模型**

osi七层模型：物理层、数据链路层、网络层、传输层、会话层、表示层、应用层

tcp/ip五层模型：物理层、数据链路层、网络层、传输层、应用层

1. **tcp和udp的区别？**

1、TCP面向连接（如打电话要先拨号建立连接）;UDP是无连接的，即发送数据之前不需要建立连接

2、TCP提供可靠的服务。也就是说，通过TCP连接传送的数据，无差错，不丢失，不重复，且按序到达;UDP尽最大努力交付，即不保证可靠交付

3、TCP面向字节流，实际上是TCP把数据看成一连串无结构的字节流;UDP是面向报文的，应用层交给UDP多长的报文，UDP就照样发送，即一次发送一个报文。UDP没有拥塞控制，因此网络出现拥塞不会使源主机的发送速率降低（对实时应用很有用，如IP电话，实时视频会议等）

4、每一条TCP连接只能是点到点的;UDP支持一对一，一对多，多对一和多对多的交互通信

5、TCP首部开销20字节;UDP的首部开销小，只有8个字节

6、TCP的逻辑通信信道是全双工的可靠信道，UDP则是不可靠信道

**数据结构：**

1. **栈和队列的区别？**
2. **如何利用栈实现队列？**

**数据库：**

1. **索引的作用以及如何存储的？**

<https://www.cnblogs.com/liulin1187740947/p/9165234.html>

首先明白为什么索引会增加速度，DB在执行一条Sql语句的时候，默认的方式是根据搜索条件进行全表扫描，遇到匹配条件的就加入搜索结果集合。如果我们对某一字段增加索引，查询时就会先去索引列表中一次定位到特定值的行数，大大减少遍历匹配的行数，所以能明显增加查询的速度。

那么在任何时候都应该加索引么？这里有几个反例：1、如果每次都需要取到所有表记录，无论如何都必须进行全表扫描了，那么是否加索引也没有意义了。2、对非唯一的字段，例如“性别”这种大量重复值的字段，增加索引也没有什么意义。3、对于记录比较少的表，增加索引不会带来速度的优化反而浪费了存储空间，因为索引是需要存储空间的，而且有个致命缺点是对于update/insert/delete的每次执行，字段的索引都必须重新计算更新。

索引存储了指向表中某一行的指针

 那么索引是如何实现的呢？大多数DB厂商实现索引都是基于一种数据结构——B树。因为B树的特点就是适合在磁盘等直接存储设备上组织动态查找表。

B-Tree 是最常用的用于索引的数据结构。因为它们是时间复杂度低， 查找、删除、插入操作都可以在对数时间内完成。另外一个重要原因存储在B-Tree中的数据是有序的。数据库管理系统（RDBMS）通常决定索引应该用哪些数据结构。但是，在某些情况下，你在创建索引时可以指定索引要使用的数据结构。

哈希表是另外一种你可能看到用作索引的数据结构-这些索引通常被称为哈希索引。使用哈希索引的原因是，在寻找值时哈希表效率极高。所以，如果使用哈希索引，对于比较字符串是否相等的查询能够极快的检索出的值。例如之前我们讨论过的这个查询(SELECT \* FROM Employee WHERE Employee\_Name = ‘Jesus’) 就可以受益于创建在Employee\_Name 列上的哈希索引。哈系索引的工作方式是将列的值作为索引的键值（key），和键值相对应实际的值（value）是指向该表中相应行的指针。因为哈希表基本上可以看作是关联数组，一个典型的数据项就像“Jesus => 0x28939″，而0x28939是对内存中表中包含Jesus这一行的引用。在哈系索引的中查询一个像“Jesus”这样的值，并得到对应行的在内存中的引用，明显要比扫描全表获得值为“Jesus”的行的方式快很多。

哈希索引的缺点：

哈希表是无顺的数据结构，对于很多类型的查询语句哈希索引都无能为力。举例来说，假如你想要找出所有小于40岁的员工。你怎么使用使用哈希索引进行查询？这不可行，因为哈希表只适合查询键值对-也就是说查询相等的查询（例：like “WHERE name = ‘Jesus’）。哈希表的键值映射也暗示其键的存储是无序的。这就是为什么哈希索引通常不是数据库索引的默认数据结构-因为在作为索引的数据结构时，其不像B-Tree那么灵活