2020字节跳动测试开发岗实习生面经总结

一、测试

1.你为什么选择测试开发

首先，我觉得测试的过程像一种解谜游戏，我从小就比较喜欢玩类似于密室逃脱的益智游戏。这一过程让我能够一点一点的观察，发现，归纳，分析，解决问题，能够让自己的学习和思维进一步发散。

其次，当我能够测出一个bug，并解决掉他时，会让我产生很大的成就感，也会让我产生一种玩游戏过关的动力，努力去发现下一个问题，并解决掉它的冲动。

2.你觉得测试开发是做什么的

测试开发是保证软件质量的一个重要环节，而且软件测试在整个开发中的作用越来越重要。我认为，开发的目标是实现功能，而测试的目标是确定功能是否能够正常运作。软件测试可以简单的说是两个关键词：“发现”和“分析”。以前我认为软件测试只需要简单的测试一下功能是否实现就可以了，后来慢慢发现，在测试的时候要考虑到很多不同的状况，从而得出不同的组合，再进行一组一组的验证测试的过程。

3.电梯测试：有一个按钮控制两个电梯，你会怎么来做测试？

首先，应该对每个电梯的：功能，性能、压力、安全、可用性、稳定性、外观等方面进行测试。

其次，要对两台电梯联动功能进行测试；

联动测试：两部均空闲时，应采取就近原则；有一部运行时，以同行方向且顺路电梯优先运行，否则安排空闲电梯；两部均运行时，以同行方向且顺路的优先运行；一台电梯出现故障，是否会影响另一部电梯等等

功能方面：按键是否灵敏、上升下降是否正常、电梯内灯光空调监控等硬件设施是否正常等；

性能测试：电梯的反应时间、上升下降速度是否合理等等；压力测试：电梯最大承重、碰撞电梯壁、晃动影响等；

安全测试：在危急情况例如停电的时候会怎样、报警装置、电梯门是否会夹到人等等；

可用性测试：按键高度、操作是否方便，对于不同人群（老人、小孩、残疾人、孕妇）的适用性。

稳定性测试：在长时间运行时的电梯状况；

4.假设是一个登陆界面，包括：用户名，密码，登陆，勾选框，注册；你会如何进行测试？

（1）功能测试：

输入为空，点击提交，看是否会有提示信息。（非空检查）

输入正确的用户名和密码，点击提交，验证是否正确登录。（正常登录）

输入错误的用户名或错误的密码，验证登录失败，并提示相应的错误信息。（错误校验）

用户名和密码中有特殊的字符（如空格）。

用户名和密码，如果太长或太短，应该如何处理。（安全性，密码太短会不会出现提示）

登录成功后能否正常跳转页面。

用户名和密码前后有空格的处理。

密码是否加密显示。

有无记住用户名的功能。

登录失败后，只记录用户名，不记录密码。

登录页面中的注册，忘记密码，退出后用另一账号登录等跳转页面是否正常。

输入密码时，键盘的大小写是否开启是否有提示。

当有验证码的时候，需要考虑文字是否扭曲过度导致辨认难度大，考虑颜色（色盲、色弱使用者），刷新或换一张图片的按钮是否正常。

（2）性能测试：

打开登录界面，需要多长时间。

输入正确的用户名和密码后，页面跳转需要多久。

（3）界面测试（UI测试）：

界面布局是否合理，两个testbox和一个按钮排版是否合理。

testbox和按钮的长度，高度是否合理。

界面中的文字简洁，无错别字。

界面设计风格是否与UI的设计风格统一。

（4）兼容性测试：

不同的浏览器下能否正常显示功能。

不同的操作系统是否能够正常工作。

移动设备上是否正常工作。

不同的分辨率。

（5）安全性测试：

登录成功后生产的Cookie，是否是httponly (否则容易被脚本盗取)。

用户名和密码是否是加密方式发生给服务器的。

考虑同一用户在多态机器上登录。

是否支持多用户在同一机器上登录。

错误登录的次数限制。（防止玻璃破解）

用户名和密码的输入框，应该禁止输入脚本。（防止XSS攻击）

用户名和密码的输入框，应该屏蔽SQL注入攻击。

用户名和密码的验证，应该用服务器端验证，而不简单的在客户端用JavaScript验证。

（6）可用性测试：

是否可以用回车键代替登录框，按回车直接登录。

是否可以全用键盘操作，是否有快捷键。

输入框是否可以使用Tab间进行切换。

（7）本地化测试：

不同语言环境下，页面的显示是否正常。

三、计网

1.说一说TCP三次握手，四次断开？

置位概念：根据TCP的包头字段，存在三个重要的标识ACK、SYN、FIN

ACK：表示验证字段

SYN：位数置为1，表示建立TCP连接

FIN：位数置为1，表示断开TCP连接

TCP的三次握手：

（1）由客户端先向服务端发送建立TCP连接的请求报文，其中报文包含seq序列号，是由发送端随机生成的，并且还将报文中SYN字段置为1，表示需要建立TCP连接。

（2）服务端会回复客户端发送的TCP连接请求报文，其中包含seq序列号，是由服务端随机生成的，并且将回复报文的SYN字段置为1，而且会产生ACK字段，ACK字段数值是在客户端发过来的seq序列号基础上加1进行回复，以便客户端收到信息时，知晓自己的TCP建立请求已经得到了验证。

（3）客户端收到服务端发送的TCP建立验证请求后，会使自己的序列号加1表示，并且再次回复ACK验证请求，在服务端发送过来的seq基础上加1，进行回复。

TCP建立过程状态机转变：

（1）一开始建立连接之前，服务端和客户端的状态都为CLOSED。

（2）服务端创建socket后开始监听，变为LISTEN状态。

（3）客户端建立连接请求，向服务端发送SYN报文，客户端的状态变为SYN\_SENT。

（4）服务端收到客户端的报文后会向客户端发送ACK和SYN报文，此时服务端的状态变为SYN\_RCVD。

（5）然后，客户端收到ACK和SYN，会向服务端发送ACK，客户端状态变为ESTABLISHED，服务端收到客户端的ACK后也变为ESTABLISHED。

（6）至此，TCP3次握手完成，建立连接！

TCP断开连接四次挥手过程：

（1）客户端发送TCP连接请求的报文，其中报文中包含了seq序列号，是由发送端随机生成的，并且还将报文中的FIN字段置为1，表示需要断开TCP连接。

（2）服务端会回复服务端发送的TCP断开请求报文，其中包含seq序列号，是由回复端随机生成的，而且会产生ACK字段，ACK字段数值，是在服务端发过来的seq序列号基础上加1进行回复的，一边服务端收到信息时，知晓自己的TCP断开请求已经得到了验证。

（3）服务端在回复完服务端的TCP断开请求后，不会马上就进行TCP连接的断开，客户端会先确保断开前，所有传输到服务端的数据是否已经传输完毕，一但确认传输数据完毕就会将回复报文的FIN字段置为1，并产生随机seq序列号。

（4）客户端收到客户端的TCP断开请求后，会回复客户端的断开请求，包含随机生成的seq字段和ACK字段，ACK字段会在客户端的TCP断开请求的seq基础上加1，从而完成客户端请求的验证回复。

（5）至此TCP断开的四次挥手过程完毕。

2.为什么连接的时候是三次握手，关闭的时候确是四次握手？

因为当服务端收到客户端的SYN连接请求报文后，可以直接发送SYN+ACK报文。其中ACK报文是用来应答的，SYN报文是用来同步的。但是关闭连接时，当服务端收到FIN报文时，很可能并不会立即关闭SOCKET，所以只能先回复一个ACK报文，告诉客户端，你发的FIN报文我收到了。只有等到我服务端所有的报文都发送完了，我才能发送FIN报文，因此不能一起发送，故需要四步握手。

3.为什么TIME\_WAIT状态需要经过2MSL（最大报文段生存时间）才能返回到CLOSE状态？

虽然按道理，四个报文都发送完毕，我们可以直接进入CLOSE状态了，但是我们必须假象网络是不可靠的，有可能最后一个ACK丢失。所以TIME\_WAIT状态就是用来重发可能丢失的ACK报文。在客户端发送出最后的ACK回复，但该ACK可能丢失。服务端如果没有收到ACK，将不断重复发送FIN片段。所以客户端不能立即关闭，他必须确认服务端接收到了ACK。客户端会在发送出ACK之后进入到TIME\_WAIT状态。客户端会设置一个计时器，等待2MSL是两倍的最大报文段生存时间。如果直到2MSL，客户端都没有再次收到FIN，那么客户端推断ACK已经被成功接收，则结束TCP连接。

4.为什么不能用两次握手进行连接？

三次握手完成两个重要的功能，既要双方做好发送数据的准备工作（双方都知道彼此已经准备好），也要允许双方就初始序列号进行协商，这个序列号在握手过程中被发送确认。

如果改成两次握手，可能会发生死锁现象。eg.客户端给服务端发送了一个连接请求，服务端收到了这个分组，并发送了确认应答分组。按照两次握手的协议，服务端认为已经成功地建立了，可以开始发送数据分组。可是客户端的应答分组在传输中被丢失的情况下，将不知道服务端是否已经准备好，不知道服务端建立什么样的序列号，客户端甚至怀疑服务端手否收到自己的连接请求分组。在这种情况下，客户端认为连接还未建立成功，将忽略服务端发来的任何数据分组，只等待连接确认应答分组。而服务端在发出的分组超时后，重复发送同样的分组。这样就形成了死锁。

5.如果已经建立了连接，但客户端突然出现故障了怎么办？

TCP还设有一个保活计时器，显然，客户端如果出现故障，服务器不能一直等下去，白白浪费资源。服务器没收到一次客户端的请求后都会重新复位这个计时器，若两小时还没有收到客户端的任何数据，服务端就会发送一个探测报文段，以后每隔75分钟发送一次，若一连发送10个探测报文任然没反应，服务端就认为客户端出了故障，接着就关闭连接。

6. Http与Https的基本概念和他们的区别

HTTP的中文叫超文本传输协议，它负责完成客户端到服务端的一系列操作，是专门用来传输注入HTML的超媒体文档等web内容的协议，它是基于传输层的TCP协议的应用层协议

HTTPS可以理解为HTTP+ssl/tls（数字证书）的组合

http和https的区别

http的URL以http://开头，而https的URL以https://开头

http是不安全的，而https是安全的

http的标准端口是80，而https的标准端口是443

在OSI网络模型中，https的加密是在传输层完成的，因为SSl是位于传输层的，TSL的前身是SSL，所以同理

http无需认证证书，而https需要认证证书

7.https的工作原理

首先服务端给客户传输证书，这个证书就是公钥，包含了很多信息，比如说证书的颁发机构，证书的过期时间

客户端进行证书的解析，若果发现没问题，就生成一个随机值（私钥），然后用证书（也就是服务端发送的公钥）进行加密，并发给服务端

服务端使用私钥将这个信息进行解密，得到客户端的私钥，然后客户端和服务端就可以通过这个私钥进行通信了

服务端将信息进行对称加密，私钥正好直有客户端和服务端知道，所以信息就比较安全了

服务端将进行对称加密后的消息进行传递

客户端使用私钥进行信息的解密

8.常用的HTTP方法有哪些

GET从服务器获得资源

POST客户端向服务器提交资源

PUT修改服务器相关资源

DELETE删除服务器相关资源

9.GET与POST方法的区别，什么时候应该使用GET，什么时候应该使用POST

GET:一般用于信息获取，使用URL传递参数，对所发送信息的数量也有限制，一般在2000个字符

POST：一般用于修改服务器上的资源，对所发送的信息数量没有限制

GET方式需要使用Request.QueryString来取得变量值，而POST方式通过Request.Form来获取变量的值，也就是说GET是通过地址栏来传值，而POST是通过提交表单来传值。

对于信息的获取一般使用get，在以下情况最好使用POST：

1.向服务器发送大量数据（因为POST没有发送数据的数量限制）

2.无法使用缓存文件（会更新服务器上的文件）

3.发送包含未知字符的用户输入时

10.常见的HTTP的状态码

2xx : 代表服务端已经成功接收并处理了该请求

200:服务端成功接收了该请求并进行了处理,请求所需要的数据应该伴随这次请求对应的响应返回给客户端

202:服务端已经成功接收了请求,但是并未进行处理

3xx : 通常代表客户端需要进行进一步请求用,常用来进行重定向的状态码

301: 客户端请求的资源已经永久移动到别的位置,服务端会自动将该请求重定向到新的位置

4xx : 通常表示客户端请求有问题

401 : 需要用户进行登录

403 : 服务端收到请求,但是拒绝进行处理

404 : 访问的资源不存在

5xx : 通常表示服务端内部错误

500 : 服务端代码出错

502 : 作为网关或者代理工作的服务器尝试执行请求时，从上游服务器接收到无效的响应

503: 服务器正在维护或者访问过载,过一段时间可能恢复正常

504 : 作为网关或者代理工作的服务器尝试执行请求时，未能及时从上游服务器（URI标识出的服务器，例如HTTP、FTP、LDAP）或者辅助服务器（例如DNS）收到响应

11.一次完整的HTTP请求与响应都发生了什么

1.在浏览器端输入网站的url地址

2.浏览器查找缓存（如果有的话，直接返回显示，如果没有的话，继续下一步）

3.通过DNS获取url对应的IP地址（先在本机的host文件中查找是否有这个url对应的IP，如果没有的话，就请求DNS进行IP地址的获取）

4.建立TCP连接（http在工作之前，需要客户端和服务端建立连接，这个链接的建立时通过TCP三次握手来完成的）

5.浏览器向web服务器发送请求（一旦连接已经建立，浏览器就可以给web服务器发送请求命令，比如GET/deom/hello.jsp HTTP/1.1）

6.浏览器给web服务器发送请求头信息（浏览器在发送了请求后，还要给web服务器请求头信息）

7.web服务器进行应答（http/1.1 200 OK)

8.web服务器发送应答头消息

9.web服务器发送数据

10.web服务器关闭连接（TCP四次挥手）

12.http优化

资源内联：既然每个资源的首次访问都会存在握手等rtt损耗，那么越少数量的资源请求就越好，例如在一个html中src访问css，不如直接将其这个css集成到html中

图片懒加载：用到的时候再加载

服务器渲染：让服务端先将页面渲染好，再发送给客户端，也可以减少rtt的次数

13.HTTP请求、响应报文格式

http请求报文格式：

http响应报文格式：

PS：URI、URL和URN之间的区别

URI全名为Uniform Resource Indentifier（统一资源标识），用来唯一的标识一个资源，是一个通用的概念，URI由两个主要的子集URL和URN组成

URL全名为Uniform Resource Locator（统一资源定位），通过描述资源的位置来标识资源

URN全名为Uniform Resource Name（统一资源命名），通过资源的名字来标识资源，与其所处的位置无关，这样即使资源的位置发生变动，其URN也不会变化

HTTP规范将更通用的概念URI作为其资源标识符，但是实际上，HTTP应用程序处理的只是URI的URL子集

14.http的缓存机制

15.TCP的拥塞控制

TCP进行拥塞控制的四种算法：慢开始、拥塞避免、快速重传、快速恢复。

16.OSI七层示例图

17.TCP和UDP的区别

TCP位于传输层，提供可靠的字节流服务，即为了方便传输，将大块数据分割成以报文段为单位的数据包进行管理。TCP连接相当于两根管道（一个用于服务器到客户端，一个用于客户端到服务端）管道里面数据传输是通过字节码传输，传输是有序的，每个字节都是一个一个来传输的。

TCP是双全工连接，以客户机发起关闭连接为例：1.服务器读通道关闭；2.客户端写通道关闭；3.客户端读通道关闭；4.服务器写通道关闭。

UDP是无连接协议，也称透明协议，位于传输层。

TCP和UDP二者区别

TCP提供面向连接的传输，通信前需要先建立连接（三次握手机制）；UDP提供无连接的传输，通信前不需要建立连接。

TCP提供可靠的传输（有序、无差错、不丢失、不重复）；UDP提供不可靠的传输。

TCP面向字节流的传输，它能将信息分割成组，并在接收端将其重组；UDP是面向数据报的传输，没有分组开销。

TCP提供拥塞控制和流量控制；UDP不提供拥塞控制和流量控制机制。

四、数据结构

1.怎样判断是否是平衡二叉树？

从下往上遍历，如果子树是平衡二叉树，则返回子树的高度；如果发现子树不是平衡二叉树，则直接停止遍历：

class Solution { public boolean isBalanced(TreeNode root) { return height(root) >= 0; } private int height(TreeNode root){ if(root == null) return 0; int lh = height(root.left),rh = height(root.right); if(lh >= 0 && rh >= 0 && Math.abs(lh - rh) <= 1){ return Math.max(lh, rh) + 1; }else{ return -1; } }}

2.了解哪些排序算法，简单介绍一下，复杂度？

3.线性表和链表有什么区别？

五、算法

1.已知一个大小为n的数组，找出出现次数超过数组长度一半的数字。时间复杂度？是否可以优化？

class Solution {

public boolean search(int[] nums) {

Array.sort(nums); return nums[nums.length/2];

}}

class Solution {

public int majorityElement(int[] nums) {

int count = 1; int maj = nums[0];

for (int i = 1; i < nums.length; i++) {

if (maj == nums[i])

count++;

else {

count--;

if (count == 0) {

maj = nums[i + 1];

}

}

}

return maj;

}

}

2.给出一个正整数n>2，找出小于n的所有质数。可以优化吗？

class Solution {

public int countPrimes(int n) {

boolean[] isPrime = new boolean[n + 1];

int count = 0;

for(int i = 2; i < n;i++){

if(isPrime[i] == false){

count++;

for(int j = i + i; j < n; j += i){

isPrime[j] = true;

}

}

}

return count;

}

}

3.实现一下链表的反转？

class Solution {

public ListNode reverseList(ListNode head) {

ListNode pre = null, cur = head, next = null;

while(cur != null){

next = cur.next;

cur.next = pre;

pre = cur;

cur = next;

}

return pre;

}

}

二、java

1.int和Interger有什么区别？它们两个如何进行赋值操作？

java有两种数据类型：基本类型，引用类型 int是基本数据类型，直接存储值；

interger是一个对象/类，用一个引用指向这个对象；

当需要往ArrayList，HashMap中放东西时，像int，double这种内建类型是放不进去的，因为容器都是装object的，这是就需要这些内建类型的外覆类了。

初始化方式不同

Integer i01 = 59;int i02 = 59;Integer i03 = Integer.valueOf(59); //因为引用类型值放在堆里面，如果堆里有值，i03这种取值方式就直接引用原有值的地址Integer i04 = new Integer(59);//i04是new创建新对象，要在堆里新开空间存放值，也就会有新地址

i01与i02比较，引用类型和值类型用等号比较，比较的是值(引用类型自动拆箱为int)，故返回true; i01与i03比较，引用类型和引用类型用等号比较，比较的是地址，两个地址一样，故返回true;

i03与i04比较，引用类型和引用类型用等号比较，比较的是地址，两个地址不同，故返回false;

i02与i04比较，值类型与引用类型用等号比较，比较的是值(引用类型自动拆箱为int)，故返回true;

2.如何判断两个对象是否相等？是根据什么进行判断的？

equals通常用来比较两个对象的内容是否相等，==用来比较两个对象的地址是否相等。

3.详细说一下Collection；java collection 和collections区别，collections.sort()具体是怎么实现的？

Collection:是集合的顶层接口，它提供了对集合对象进行基本操作的通用接口的方法。有子接口List和Set。Collection的基本操作有：add();addAll();remove();removeAll();clear();retainAll();

Collections:是针对集合操作的工具类，它包含有各种有关集合操作的方法，有对集合进行排序和二分查找的方法，此类不能实例化，服务于Java的collection框架。

Collections中的sort方法的源码

public static <T extends Comparable<? super T>> void sort(List<T> list) { Object[] a = list.toArray(); Arrays.sort(a); ListIterator<T> i = list.listIterator(); for (int j=0; j<a.length; j++) { i.next(); i.set((T)a[j]); } }

可以看出，Collections.sort()方法先将传入的list转成数组对象后，调用了Array的sort方法，将数组排序，再用list迭代器一个一个的赋值回去，就使得传入的list变成了一个有序的list。

sort方法默认的顺序是升序，如果你想要降序排，你可以调用Collections中的另一个sort方法

public static <T> void sort(List<T> list, Comparator<? super T> c) { Object[] a = list.toArray(); Arrays.sort(a, (Comparator)c); ListIterator i = list.listIterator(); for (int j=0; j<a.length; j++) { i.next(); i.set(a[j]); } }

可以看到，这个sort方法是传入了一个比较器，所以可以定义如何来排序。

4.ArrayList与LinkedList有什么区别？Set与List又有什么区别？

ArrayList与LinkedList主要区别来自于他们的数据结构不同。

（1）ArrayList是基于数组实现的，LinkedList是基于双链表实现的；

（2）ArrayList是基于索引（index）的数据结构，在使用索引查找和读取数据是很快的，可以直接返回数组中index位置的元素，因此在随机访问集合元素上有较好的性能。时间复杂度为O(1)，但是要插入、删除数据却是开销很大的，因为这需要移动数组中插入位置之后的元素。

（3）相对于ArrayList，LinkedList的随机访问集合元素时的性能较差，因为需要再双向链表中找到index的位置，再返回，时间复杂度为O(n)；但在插入、删除操作时更快，时间复杂度为O(1).

（4）LinkedList需要更多的内存，以内ArrayList的每个索引的位置是实际的数据，而LinkedList中的每个节点中存储的是实际元素的数据和前后节点的位置。

（5）如果应用程序对数据有较多的随机访问，ArrayList对象要优于LinkedList对象；如果应用程序有更多的插入或者删除操作，较少的数据读取，LinkedList对象要优于ArrayList对象。

set和list的区别：

（1）list可以允许重复对象，而set不允许；

（2）list容器是有序的，而set容器是无序的；

（3）list可以允许插入多个null值，而set方法只允许插入一个null元素。

Java中的集合共包含三大类，它们分别是Set（集）、List（列表）以及Map（映射）。它们都处在java.util中，并且都为接口。他们各自都有各自的实现类，如下图：

5.java override和overload

方法重写（override）和方法重载（overload）时Java多态性的不同表现，重写是父类与子类之间多态性的一种表现，重载可以理解成多态的具体表现形式，是一个类的多态表现：

重写（override）

重写（override）时子类对父类的允许访问的方法的实现过程进行重新编写，返回值和形参都不能改变，即外壳不变，核心重写。

方法重写规则

参数列表必须完全与被重写方法的参数相同

返回类型与被重写方法的返回类型可以不相同

访问权限不能比父类中被重写的方法的访问权限更低

父类成员的方法只能被他的子类重写

声明为final的方法不能重写

声明static的方法不能被重写，但是能够被再次声明

构造方法不能被重写

子类和父类在同一个包中，那么子类可以重写父类所有方法，除了声明为private和final的方法

子类和父类不在同一个包，那么子类只能够重写父类的声明为public和protect的非final方法

子类中调用父类的被重写方法时，要用super关键字

重载（overload）

重载（overload）是在一个类里面，方法名字相同，而参数不同。返回类型可以相同也可以不同。最常用的地方就是构造器的重载。

重载规则

被重载的方法必须改变参数列表

被重载方法可以改变返回类型

被重载方法可以改变访问修饰符

被重载方法可以声明新的或更广的检查异常

方法能够在同一个类中或再一个子类中被重载

无法与返回值类型作为重载函数的区分标准

6.This的作用是什么？用代码实现a().b().c()?

这是java语言的链式编程,当一个类中的函数返回值是其类本身(this)时,就可以用链式编程

7.进程与线程的区别

（1）进程是资源的分配和调度的一个独立单元，而线程是CPU调度的基本单元

（2）同一个进程中可以包括多个线程，并且线程共享整个进程的资源（寄存器、堆栈、上下文），一个进程至少包括一个线程。

（3）进程的创建调用fork或者vfork，而线程的创建调用pthread\_create，进程结束后它拥有的所有线程都将销毁，而线程的结束不会影响同个进程中的其他线程的结束

（4）线程是轻量级的进程，它的创建和销毁所需要的时间比进程小很多，所有操作系统中的执行功能都是创建线程去完成的

（5）线程中执行时一般都要进行同步和互斥，因为他们共享同一进程的所有资源

（6）线程有自己的私有属性TCB，线程id，寄存器、硬件上下文，而进程也有自己的私有属性进程控制块PCB，这些私有属性是不被共享的，用来标示一个进程或一个线程的标志

（7）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

（8）调度：线程作为调度和分配的基本单位，进程作为拥有资源的基本单位

（9）并发性：不仅进程之间可以并发执行，同一个进程的多个线程之间也可并发执行

（10）拥有资源：进程是拥有资源的一个独立单位，线程不拥有系统资源，但可以访问隶属于进程的资源.

8.进程和线程的通信方式

进程间的通信方式：

1.管道(pipe)及有名管道(named pipe)：

管道可用于具有亲缘关系的父子进程间的通信，有名管道除了具有管道所具有的功能外，它还允许无亲缘关系进程间的通信。

2.信号(signal)：

信号是在软件层次上对中断机制的一种模拟，它是比较复杂的通信方式，用于通知进程有某事件发生，一个进程收到一个信号与处理器收到一个中断请求效果上可以说是一致的。

3.消息队列(message queue)：

消息队列是消息的链接表，它克服了上两种通信方式中信号量有限的缺点，具有写权限得进程可以按照一定得规则向消息队列中添加新信息；对消息队列有读权限得进程则可以从消息队列中读取信息。

4.共享内存(shared memory)：

可以说这是最有用的进程间通信方式。它使得多个进程可以访问同一块内存空间，不同进程可以及时看到对方进程中对共享内存中数据得更新。这种方式需要依靠某种同步操作，如互斥锁和信号量等。

5.信号量(semaphore)：

主要作为进程之间及同一种进程的不同线程之间得同步和互斥手段。

6.套接字(socket)；

这是一种更为一般得进程间通信机制，它可用于网络中不同机器之间的进程间通信，应用非常广泛。

线程之间的同步通信：

1.信号量：二进制信号量、互斥信号量、整数型信号量、记录型信号量

2.消息：消息队列、消息邮箱

3.事件event

互斥型信号量：必须是同一个任务申请，同一个任务释放，其他任务释放无效。同一个任务可以递归申请。(互斥信号量是二进制信号量的一个子集)

二进制信号量：一个任务申请成功后，可以由另一个任务释放。(与互斥信号量的区别)

整数型信号量：取值不局限于0和1,可以一个任务申请，另一个任务释放。(包含二进制信号量，二进制信号量是整数型信号量的子集)

二进制信号量实现任务互斥：

打印机资源只有一个，a bc三个任务共享，当a取得使用权后，为了防止其他任务错误地释放了信号量(二进制信号量允许其他任务释放)，必须将打印机房的门关起来(进入临界段)，用完后，释放信号量，再把门打开(出临界段)，其他任务再进去打印。(而互斥型信号量由于必须由取得信号量的那个任务释放，故不会出现其他任务错误地释放了信号量的情况出现，故不需要有临界段。互斥型信号量是二进制信号量的子集。)

二进制信号量实现任务同步：

a任务一直等待信号量，b任务定时释放信号量，完成同步功能

记录型信号量(record semaphore)：

每个信号量s除一个整数值value(计数)外，还有一个等待队列List，其中是阻塞在该信号量的各个线程的标识。当信号量被释放一个，值被加一后，系统自动从等待队列中唤醒一个等待中的线程，让其获得信号量，同时信号量再减一。

9.Java内存模型

java内存模型(Java Memory Model，JMM)是java虚拟机规范定义的，用来屏蔽掉java程序在各种不同的硬件和操作系统对内存的访问的差异，这样就可以实现java程序在各种不同的平台上都能达到内存访问的一致性。

Java内存模型中涉及到的概念有：

主内存：java虚拟机规定所有的变量(不是程序中的变量)都必须在主内存中产生，为了方便理解，可以认为是堆区。可以与前面说的物理机的主内存相比，只不过物理机的主内存是整个机器的内存，而虚拟机的主内存是虚拟机内存中的一部分。

工作内存：java虚拟机中每个线程都有自己的工作内存，该内存是线程私有的为了方便理解，可以认为是虚拟机栈。可以与前面说的高速缓存相比。线程的工作内存保存了线程需要的变量在主内存中的副本。虚拟机规定，线程对主内存变量的修改必须在线程的工作内存中进行，不能直接读写主内存中的变量。不同的线程之间也不能相互访问对方的工作内存。如果线程之间需要传递变量的值，必须通过主内存来作为中介进行传递。

10.java中的垃圾回收机制

jvm怎么确定哪些对象应该进行回收：

对象是否会被回收的两个经典算法：引用计数法，和可达性分析算法。

在确定了哪些对象可以被回收之后，jvm会在什么时候进行回收

1会在cpu空闲的时候自动进行回收

2在堆内存存储满了之后

3主动调用System.gc()后尝试进行回收

如何回收说的也就是垃圾收集的算法

算法又有四个：标记-清除算法,复制算法,标记-整理算法,分代收集算法.

11.接口和抽象类的区别

抽象类？什么是抽象类？

从名字上来讲，我觉得就是对类的一个抽象，把类（事物）抽象出来，当做模板，也就是说在有很多类的时候，我们把一些相似的类的某些方法，某些成员变量抽象出来作为一个模板，让这些类更方便的去继承。所以，在抽象类中，有抽象方法也有普通方法。一个抽象类就是包含抽象方法的类（java编程思想）。

在继承抽象类的时候，子类必须得实现抽象类中的所有抽象方法，抽象类也无法去创建实例，因为里面含有抽象的方法。

接口？什么是接口？

接口通常来说是一个对行为的抽象，它是提供给别人调用的方法。它里面所有的方法都必须为public，而且它的成员变量必须是public static final 的。

接口与抽象类的区别：

1.继承抽象类的子类们的本质都是相似的，它们体现的是一种 “is-a" 的关系，就像动物中的猫和狗；而对于接口的继承更多的是一种行为的相似，是一种 “like-a” 的关系，比如飞机和鸟，它们都具有飞的行为，却并不需要在本质上相似。

2.抽象类可以拥有任意范围的成员数据，既可以是抽象，也可以是非抽象；但是接口，所有的方法必须是抽象的，所有的成员变量必须是 public static final的，某种程度上来说，接口是对抽象类的一种抽象。

3.一个类只能继承一个抽象类，但却可以实现多个接口。