|  |
| --- |
| 技术问题(至少50个) |
| 1.Dubbo 用什么协议  dubbo://（推荐）rmi:/hessian:/http:/Webservice/thrift:/memcached/redis/rest  2. kafka 可以脱离 zookeeper 单独使用吗？为什么？  kafka 不能脱离 zookeeper 单独使用，因为 kafka 使用 zookeeper 管理和协调 kafka 的节点服务器。  3. kafka 有几种数据保留的策略？  kafka 有两种数据保存策略：按照过期时间保留和按照存储的消息大小保留。  4. kafka 同时设置了 7 天和 10G 清除数据，到第五天的时候消息达到了 10G，这个时候 kafka 将如何处理？  这个时候 kafka 会执行数据清除工作，时间和大小不论那个满足条件，都会清空数据。  5. 什么情况会导致 kafka 运行变慢？  cpu 性能瓶颈  磁盘读写瓶颈  网络瓶颈  6. 使用 kafka 集群需要注意什么？  集群的数量不是越多越好，最好不要超过 7 个，因为节点越多，消息复制需要的时间就越长，整个群组的吞吐量就越低。  集群数量最好是单数，因为超过一半故障集群就不能用了，设置为单数容错率更高。  Zookeeper  7. zookeeper 是什么？  zookeeper 是一个分布式的，开放源码的分布式应用程序协调服务，是 google chubby 的开源实现，是 hadoop 和 hbase 的重要组件。它是一个为分布式应用提供一致性服务的软件，提供的功能包括：配置维护、域名服务、分布式同步、组服务等。  8. zookeeper 都有哪些功能？  **集群管理**：监控节点存活状态、运行请求等。  **主节点选举**：主节点挂掉了之后可以从备用的节点开始新一轮选主，主节点选举说的就是这个选举的过程，使用 zookeeper 可以协助完成这个过程。  **分布式锁**：zookeeper 提供两种锁：独占锁、共享锁。独占锁即一次只能有一个线程使用资源，共享锁是读锁共享，读写互斥，即可以有多线线程同时读同一个资源，如果要使用写锁也只能有一个线程使用。zookeeper可以对分布式锁进行控制。  **命名服务**：在分布式系统中，通过使用命名服务，客户端应用能够根据指定名字来获取资源或服务的地址，提供者等信息。  9. zookeeper 有几种部署模式？  zookeeper 有三种部署模式：  单机部署：一台集群上运行；  集群部署：多台集群运行；  伪集群部署：一台集群启动多个 zookeeper 实例运行。  10. zookeeper 怎么保证主从节点的状态同步？  zookeeper 的核心是原子广播，这个机制保证了各个 server 之间的同步。实现这个机制的协议叫做 zab 协议。 zab 协议有两种模式，分别是**恢复模式**（选主）和**广播模式**（同步）。当服务启动或者在领导者崩溃后，zab 就进入了恢复模式，当领导者被选举出来，且大多数 server 完成了和 leader 的状态同步以后，恢复模式就结束了。状态同步保证了 leader 和 server 具有相同的系统状态。  11. 集群中为什么要有主节点？  在分布式环境中，有些业务逻辑只需要集群中的某一台机器进行执行，其他的机器可以共享这个结果，这样可以大大减少重复计算，提高性能，所以就需要主节点。  12. 集群中有 3 台服务器，其中一个节点宕机，这个时候 zookeeper 还可以使用吗？  可以继续使用，单数服务器只要没超过一半的服务器宕机就可以继续使用。  13. 说一下 zookeeper 的通知机制？  客户端端会对某个 znode 建立一个 watcher 事件，当该 znode 发生变化时，这些客户端会收到 zookeeper 的通知，然后客户端可以根据 znode 变化来做出业务上的改变。  MySQL  14. 数据库的三范式是什么？  第一范式：强调的是列的原子性，即数据库表的每一列都是不可分割的原子数据项。  第二范式：要求实体的属性完全依赖于主关键字。所谓完全依赖是指不能存在仅依赖主关键字一部分的属性。  第三范式：任何非主属性不依赖于其它非主属性。  15. 一张自增表里面总共有 7 条数据，删除了最后 2 条数据，重启 MySQL 数据库，又插入了一条数据，此时 id 是几？  表类型如果是 MyISAM ，那 id 就是 8。  表类型如果是 InnoDB，那 id 就是 6。  InnoDB 表只会把自增主键的最大 id 记录在内存中，所以重启之后会导致最大 id 丢失。  因为MyISAM表会把自增主键的最大ID记录到数据文件里，重启MySQL自增主键的最大ID也不会丢失。  16. 如何获取当前数据库版本？  使用 select version() 获取当前 MySQL 数据库版本。  17. 说一下 ACID 是什么？  Atomicity（原子性）：一个事务（transaction）中的所有操作，或者全部完成，或者全部不完成，不会结束在中间某个环节。事务在执行过程中发生错误，会被恢复（Rollback）到事务开始前的状态，就像这个事务从来没有执行过一样。即，事务不可分割、不可约简。  Consistency（一致性）：在事务开始之前和事务结束以后，数据库的完整性没有被破坏。这表示写入的资料必须完全符合所有的预设约束、触发器、级联回滚等。  Isolation（隔离性）：数据库允许多个并发事务同时对其数据进行读写和修改的能力，隔离性可以防止多个事务并发执行时由于交叉执行而导致数据的不一致。事务隔离分为不同级别，包括读未提交（Read uncommitted）、读提交（read committed）、可重复读（repeatable read）和串行化（Serializable）。  Durability（持久性）：事务处理结束后，对数据的修改就是永久的，即便系统故障也不会丢失。  18. char 和 varchar 的区别是什么？  char(n) ：固定长度类型，比如订阅 char(10)，当你输入"abc"三个字符的时候，它们占的空间还是 10 个字节，其他 7 个是空字节。  char 优点：效率高；缺点：占用空间；适用场景：存储密码的 md5 值，固定长度的，使用 char 非常合适。  varchar(n) ：可变长度，存储的值是每个值占用的字节再加上一个用来记录其长度的字节的长度。  所以，从空间上考虑 varcahr 比较合适；从效率上考虑 char 比较合适，二者使用需要权衡。  19. float 和 double 的区别是什么？  float 最多可以存储 8 位的十进制数，并在内存中占 4 字节。  double 最可可以存储 16 位的十进制数，并在内存中占 8 字节。  20. MySQL 的内连接、左连接、右连接有什么区别？  内连接关键字：inner join；左连接：left join；右连接：right join。  内连接是把匹配的关联数据显示出来；左连接是左边的表全部显示出来，右边的表显示出符合条件的数据；右连接正好相反。  21. MySQL 索引是怎么实现的？  索引是满足某种特定查找算法的数据结构，而这些数据结构会以某种方式指向数据，从而实现高效查找数据。  具体来说 MySQL 中的索引，不同的数据引擎实现有所不同，但目前主流的数据库引擎的索引都是 B+ 树实现的，B+ 树的搜索效率，可以到达二分法的性能，找到数据区域之后就找到了完整的数据结构了，所有索引的性能也是更好的。  22. 怎么验证 MySQL 的索引是否满足需求？  使用 explain 查看 SQL 是如何执行查询语句的，从而分析你的索引是否满足需求。  explain 语法：explain select \* from table where type=1。  23. 说一下数据库的事务隔离？  MySQL 的事务隔离是在 MySQL. ini 配置文件里添加的，在文件的最后添加：  transaction-isolation = REPEATABLE-READ  可用的配置值：READ-UNCOMMITTED、READ-COMMITTED、REPEATABLE-READ、SERIALIZABLE。  READ-UNCOMMITTED：未提交读，最低隔离级别、事务未提交前，就可被其他事务读取（会出现幻读、脏读、不可重复读）。  READ-COMMITTED：提交读，一个事务提交后才能被其他事务读取到（会造成幻读、不可重复读）。  REPEATABLE-READ：可重复读，默认级别，保证多次读取同一个数据时，其值都和事务开始时候的内容是一致，禁止读取到别的事务未提交的数据（会造成幻读）。  SERIALIZABLE：序列化，代价最高最可靠的隔离级别，该隔离级别能防止脏读、不可重复读、幻读。  脏读 ：表示一个事务能够读取另一个事务中还未提交的数据。比如，某个事务尝试插入记录 A，此时该事务还未提交，然后另一个事务尝试读取到了记录 A。  不可重复读 ：是指在一个事务内，多次读同一数据。  幻读 ：指同一个事务内多次查询返回的结果集不一样。比如同一个事务 A 第一次查询时候有 n 条记录，但是第二次同等条件下查询却有 n+1 条记录，这就好像产生了幻觉。发生幻读的原因也是另外一个事务新增或者删除或者修改了第一个事务结果集里面的数据，同一个记录的数据内容被修改了，所有数据行的记录就变多或者变少了。  24. 说一下 MySQL 常用的引擎？  InnoDB 引擎：mysql 5.1 后默认的数据库引擎，提供了对数据库 acid 事务的支持，并且还提供了行级锁和外键的约束，它的设计的目标就是处理大数据容量的数据库系统。MySQL 运行的时候，InnoDB 会在内存中建立缓冲池，用于缓冲数据和索引。但是该引擎是不支持全文搜索，同时启动也比较的慢，它是不会保存表的行数的，所以当进行 select count(\*) from table 指令的时候，需要进行扫描全表。由于锁的粒度小，写操作是不会锁定全表的,所以在并发度较高的场景下使用会提升效率的。    MyIASM 引擎：不提供事务的支持，也不支持行级锁和外键。因此当执行插入和更新语句时，即执行写操作的时候需要锁定这个表，所以会导致效率会降低。不过和 InnoDB 不同的是，MyIASM 引擎是保存了表的行数，于是当进行 select count(\*) from table 语句时，可以直接的读取已经保存的值而不需要进行扫描全表。所以，如果表的读操作远远多于写操作时，并且不需要事务的支持的，可以将 MyIASM 作为数据库引擎的首选。  25. 说一下 MySQL 的行锁和表锁？  MyISAM 只支持表锁，InnoDB 支持表锁和行锁，默认为行锁。  表级锁：开销小，加锁快，不会出现死锁。锁定粒度大，发生锁冲突的概率最高，并发量最低。  行级锁：开销大，加锁慢，会出现死锁。锁力度小，发生锁冲突的概率小，并发度最高。  26. 说一下乐观锁和悲观锁？  乐观锁：每次去拿数据的时候都认为别人不会修改，所以不会上锁，但是在提交更新的时候会判断一下在此期间别人有没有去更新这个数据。  悲观锁：每次去拿数据的时候都认为别人会修改，所以每次在拿数据的时候都会上锁，这样别人想拿这个数据就会阻止，直到这个锁被释放。  数据库的乐观锁需要自己实现，在表里面添加一个 version 字段，每次修改成功值加 1，这样每次修改的时候先对比一下，自己拥有的 version 和数据库现在的 version 是否一致，如果不一致就不修改，这样就实现了乐观锁。  27. MySQL 问题排查都有哪些手段？  使用 show processlist 命令查看当前所有连接信息。  使用 explain 命令查询 SQL 语句执行计划。  开启慢查询日志，查看慢查询的 SQL。  28. 如何做 MySQL 的性能优化？  为搜索字段创建索引。  避免使用 select \*，列出需要查询的字段。  垂直分割分表。  选择正确的存储引擎。  Redis  29. Redis 是什么？都有哪些使用场景？  Redis 是一个使用 C 语言开发的高速缓存数据库。  Redis 使用场景：  记录帖子点赞数、点击数、评论数；  缓存近期热帖；  缓存文章详情信息；  记录用户会话信息。  30. Redis 有哪些功能？  数据缓存功能  分布式锁的功能  支持数据持久化  支持事务  支持消息队列  31. Redis 和 memcache 有什么区别？  存储方式不同：memcache 把数据全部存在内存之中，断电后会挂掉，数据不能超过内存大小；Redis 有部份存在硬盘上，这样能保证数据的持久性。  数据支持类型：memcache 对数据类型支持相对简单；Redis 有复杂的数据类型。  使用底层模型不同：它们之间底层实现方式，以及与客户端之间通信的应用协议不一样，Redis 自己构建了 vm 机制，因为一般的系统调用系统函数的话，会浪费一定的时间去移动和请求。  value 值大小不同：Redis 最大可以达到 512mb；memcache 只有 1mb。  32. Redis 为什么是单线程的？  因为 cpu 不是 Redis 的瓶颈，Redis 的瓶颈最有可能是机器内存或者网络带宽。既然单线程容易实现，而且 cpu 又不会成为瓶颈，那就顺理成章地采用单线程的方案了。  关于 Redis 的性能，官方网站也有，普通笔记本轻松处理每秒几十万的请求。  而且单线程并不代表就慢 nginx 和 nodejs 也都是高性能单线程的代表。  33. 什么是缓存穿透？怎么解决？  缓存穿透：指查询一个一定不存在的数据，由于缓存是不命中时需要从数据库查询，查不到数据则不写入缓存，这将导致这个不存在的数据每次请求都要到数据库去查询，造成缓存穿透。  解决方案：最简单粗暴的方法如果一个查询返回的数据为空（不管是数据不存在，还是系统故障），我们就把这个空结果进行缓存，但它的过期时间会很短，最长不超过五分钟。  34. Redis 支持的数据类型有哪些？  Redis 支持的数据类型：string（字符串）、list（列表）、hash（字典）、set（集合）、zset（有序集合）。  35. Redis 支持的 Java 客户端都有哪些？  支持的 Java 客户端有 Redisson、jedis、lettuce 等。  36. jedis 和 Redisson 有哪些区别？  jedis：提供了比较全面的 Redis 命令的支持。  Redisson：实现了分布式和可扩展的 Java 数据结构，与 jedis 相比 Redisson 的功能相对简单，不支持排序、事务、管道、分区等 Redis 特性。  37. 怎么保证缓存和数据库数据的一致性？  合理设置缓存的过期时间。  新增、更改、删除数据库操作时同步更新 Redis，可以使用事物机制来保证数据的一致性。  38. Redis 持久化有几种方式？  Redis 的持久化有两种方式，或者说有两种策略：  RDB（Redis Database）：指定的时间间隔能对你的数据进行快照存储。  AOF（Append Only File）：每一个收到的写命令都通过write函数追加到文件中。  39. Redis 怎么实现分布式锁？  Redis 分布式锁其实就是在系统里面占一个“坑”，其他程序也要占“坑”的时候，占用成功了就可以继续执行，失败了就只能放弃或稍后重试。  占坑一般使用 setnx(set if not exists)指令，只允许被一个程序占有，使用完调用 del 释放锁。  40. Redis 分布式锁有什么缺陷？  Redis 分布式锁不能解决超时的问题，分布式锁有一个超时时间，程序的执行如果超出了锁的超时时间就会出现问题。  41. Redis 如何做内存优化？  尽量使用 Redis 的散列表，把相关的信息放到散列表里面存储，而不是把每个字段单独存储，这样可以有效的减少内存使用。比如将 Web 系统的用户对象，应该放到散列表里面再整体存储到 Redis，而不是把用户的姓名、年龄、密码、邮箱等字段分别设置 key 进行存储。  42. Redis 淘汰策略有哪些？  volatile-lru：从已设置过期时间的数据集（server. db[i]. expires）中挑选最近最少使用的数据淘汰。  volatile-ttl：从已设置过期时间的数据集（server. db[i]. expires）中挑选将要过期的数据淘汰。  volatile-random：从已设置过期时间的数据集（server. db[i]. expires）中任意选择数据淘汰。  allkeys-lru：从数据集（server. db[i]. dict）中挑选最近最少使用的数据淘汰。  allkeys-random：从数据集（server. db[i]. dict）中任意选择数据淘汰。  no-enviction（驱逐）：禁止驱逐数据。  43. Redis 常见的性能问题有哪些？该如何解决？  主服务器写内存快照，会阻塞主线程的工作，当快照比较大时对性能影响是非常大的，会间断性暂停服务，所以主服务器最好不要写内存快照。  Redis 主从复制的性能问题，为了主从复制的速度和连接的稳定性，主从库最好在同一个局域网内。  JVM  44. 说一下 JVM 的主要组成部分？及其作用？  类加载器（ClassLoader）  运行时数据区（Runtime Data Area）  执行引擎（Execution Engine）  本地库接口（Native Interface）  组件的作用： 首先通过类加载器（ClassLoader）会把 Java 代码转换成字节码，运行时数据区（Runtime Data Area）再把字节码加载到内存中，而字节码文件只是 JVM 的一套指令集规范，并不能直接交给底层操作系统去执行，因此需要特定的命令解析器执行引擎（Execution Engine），将字节码翻译成底层系统指令，再交由 CPU 去执行，而这个过程中需要调用其他语言的本地库接口（Native Interface）来实现整个程序的功能。  45. 说一下 JVM 运行时数据区？  不同虚拟机的运行时数据区可能略微有所不同，但都会遵从 Java 虚拟机规范， Java 虚拟机规范规定的区域分为以下 5 个部分：  程序计数器（Program Counter Register）：当前线程所执行的字节码的行号指示器，字节码解析器的工作是通过改变这个计数器的值，来选取下一条需要执行的字节码指令，分支、循环、跳转、异常处理、线程恢复等基础功能，都需要依赖这个计数器来完成；  Java 虚拟机栈（Java Virtual Machine Stacks）：用于存储局部变量表、操作数栈、动态链接、方法出口等信息；  本地方法栈（Native Method Stack）：与虚拟机栈的作用是一样的，只不过虚拟机栈是服务 Java 方法的，而本地方法栈是为虚拟机调用 Native 方法服务的；  Java 堆（Java Heap）：Java 虚拟机中内存最大的一块，是被所有线程共享的，几乎所有的对象实例都在这里分配内存；  方法区（Methed Area）：用于存储已被虚拟机加载的类信息、常量、静态变量、即时编译后的代码等数据。  46. 说一下堆栈的区别？  功能方面：堆是用来存放对象的，栈是用来执行程序的。  共享性：堆是线程共享的，栈是线程私有的。  空间大小：堆大小远远大于栈。  47. 队列和栈是什么？有什么区别？  队列和栈都是被用来预存储数据的。  队列允许先进先出检索元素，但也有例外的情况，Deque 接口允许从两端检索元素。  栈和队列很相似，但它运行对元素进行后进先出进行检索。  48. 什么是双亲委派模型？  在介绍双亲委派模型之前先说下类加载器。对于任意一个类，都需要由加载它的类加载器和这个类本身一同确立在 JVM 中的唯一性，每一个类加载器，都有一个独立的类名称空间。类加载器就是根据指定全限定名称将 class 文件加载到 JVM 内存，然后再转化为 class 对象。  类加载器分类：  启动类加载器（Bootstrap ClassLoader），是虚拟机自身的一部分，用来加载Java\_HOME/lib/目录中的，或者被 -Xbootclasspath 参数所指定的路径中并且被虚拟机识别的类库；  其他类加载器：  扩展类加载器（Extension ClassLoader）：负责加载\lib\ext目录或Java. ext. dirs系统变量指定的路径中的所有类库；  应用程序类加载器（Application ClassLoader）。负责加载用户类路径（classpath）上的指定类库，我们可以直接使用这个类加载器。一般情况，如果我们没有自定义类加载器默认就是用这个加载器。  双亲委派模型：如果一个类加载器收到了类加载的请求，它首先不会自己去加载这个类，而是把这个请求委派给父类加载器去完成，每一层的类加载器都是如此，这样所有的加载请求都会被传送到顶层的启动类加载器中，只有当父加载无法完成加载请求（它的搜索范围中没找到所需的类）时，子加载器才会尝试去加载类。  49. 说一下类装载的执行过程？  类装载分为以下 5 个步骤：  加载：根据查找路径找到相应的 class 文件然后导入；  检查：检查加载的 class 文件的正确性；  准备：给类中的静态变量分配内存空间；  解析：虚拟机将常量池中的符号引用替换成直接引用的过程。符号引用就理解为一个标示，而在直接引用直接指向内存中的地址；  初始化：对静态变量和静态代码块执行初始化工作。  50. 怎么判断对象是否可以被回收？  一般有两种方法来判断：  引用计数器：为每个对象创建一个引用计数，有对象引用时计数器 +1，引用被释放时计数 -1，当计数器为 0 时就可以被回收。它有一个缺点不能解决循环引用的问题；  可达性分析：从 GC Roots 开始向下搜索，搜索所走过的路径称为引用链。当一个对象到 GC Roots 没有任何引用链相连时，则证明此对象是可以被回收的。   1. 线程局部副本ThreadLocal   如果当类中的某个变量希望根据不同的线程提供不同的值，而且任意一个线程修改这个变量不会影响到其他线程，那么这个变量就应该是线程的局部变量。 一般的用法是用 private static 修饰变量，这是因为ThreadLocalMap中的key值是一个弱引用，是以ThreadLocal为key，所以要用static来延长ThreadLocal的生存时间。  在ThreadLocal中维护着一个map，并且这个map定义在在ThreadLocal内部（ThreadLocalMap），Key是ThreadLocal对象、value是线程局部变量，这样维护和定义的map有三个好处，第一，线程间**不需要竞争访问Map；第二，所有局部变量随线程一起被销毁回收；第三key值是弱引用，尽可能地释放过时的键值对Entry，回收内存。**   1. **ThreadLocal内存泄漏**   **ThreadLocal里面使用了一个存在弱引用的map, map的类型是ThreadLocal.ThreadLocalMap. Map中的key为一个threadlocal实例。这个Map的确使用了弱引用，不过弱引用只是针对key。每个key都弱引用指向threadlocal。 当把threadlocal实例置为null以后，没有任何强引用指向threadlocal实例，所以threadlocal将会被gc回收。**  **但是，我们的value却不能回收，而这块value永远不会被访问到了，所以存在着内存泄露。因为存在一条从current thread连接过来的强引用。只有当前thread结束以后，current thread就不会存在栈中，强引用断开，Current Thread、Map value将全部被GC回收。最好的做法是将调用threadlocal的remove方法，** |
| **人资问题(至少30个)** |
| 1.如何安排自己的时间？会不会排斥加班？  回答提示：基本上，如果上班工作有效率，工作量合理的话，应该不太需要加班。可是我也知道有时候很难避免加班，加上现在工作都采用责任制，所以我会调配自己的时间，全力配合。  分析：虽然不会有人心甘情愿的加班，但依旧要表现出高配合度的诚意。  2.你觉得你个性上最大的优点是什么？  沉着冷静、条理清楚、立场坚定、顽强向上、乐于助人、知应能力强，并且有一定的项目实战能力    3.你最大的缺点是什么？  性格有些浮躁（年轻人的通病，无可厚非，说完后一定要说自己正在努力改正）、社会阅历尚浅。      4.你对加班的看法是什么？  如果工作需要会义不容辞，现在没有家庭负担可全身心投入工作，但会提高工作效率减少不必要的加班。    5.你对薪资的要求是多少？  我对薪资不是很看重，我在意的是咱们公司这个平台，因为上家薪资是12k，我希望能有所提升，但是工资也不是我选择工作的因素，更看重个人提升的机会        6.在五年的时间内，你的职业规划是什么？  我希望能在公司稳定发展，并且个人技术水平也要利用空闲时间进行提升，并且能够为公司创造更多的利益；    7.你朋友对你的评价怎样？  我的朋友觉得我是个值得信赖的人，并且我也乐于帮助他们解决一些生活或者工作上的困难，我们关系很融洽；    8.你还有什么问题要问的吗？  办理入职时要带什么资料？  什么时候入职合适？  如果入职了有没有人带一带，以便尽快熟悉公司业务。    9.如果面试录用了你，一段时间后发现你不适合这个职位，你怎么办？  通过自身努力学习、虚心求教在，尽快适应尽量不出现这种情况，如果实在不适合不给公司添加任何麻烦。      10.工作时，你认为领导要求的方式不是最好的，你应该怎么做？  原则上服从领导，私底下找机会请教、婉转的表达自己的想法看看领导是否会改变，如果没改变就按领导的走。    11.如果你的工作出现失误，给本公司造成经济损失，你认为该怎么办？  我本意是为公司努力工作如果造成损失首要问题是想办法去弥补或挽回损失，分清责任各负其责，如果是本人的责任自愿受罚，如果是团队成员的失误帮助其找到原因并解决。    12.如果你做的工作受到上级领导的表扬，但你的主管说是他做的，你怎样？  如果没有领导的正确指导和大力支持自己是不能做好的，是谁做好工作并不重要，只要对公司有利就行。    13.谈谈你对跳槽的看法？  正常的跳槽能促进人才合理流动，应该支持。频繁跳槽对单位和个人双方都不利，应该反对。    14.工作中你难以和同事、上司相处，你该怎么办？  会服从领导指挥配合同事工作，从自身找原因分析是不是自己的问题让领导不满意同事看不惯，是不是为人处事做的不好，如果是就努力去改正，如果不是就去找机会沟通，找出问题关键并解决。      15.假设你工作做的很好，得到领导肯定，但同事孤立你，你怎么看这个问题？  工作做的好是好事也是本分，以后也会更加努力。同事孤立需要检讨一下自己是不是没有正确认识成绩，取得成绩要共享荣誉，遇到问题要检讨自己。  16.你对我们公司了解多少？  关键点：面试前要提前对面试企业全面的了解一下。  介绍点：了解内容包括（企业文化、性质、经营范围、取得的成就等）。    17.请说出你选择这份工作的动机？  我之前了解过公司的文化，我很喜欢\*\*\*\*，我觉得公司文化的某些方面和我的观念一致，我觉得在这样的公司能够实现自己的价值，能够让自己的能力最大化的为公司创造价值；        20.最能概括你自己的三个词是什么？  适应力强、有责任心、做事善始善终等。    21.你的业余爱好是什么？  比较喜欢打篮球和登山。    22.作为被面试者给我打一下分？  亲和力强，责任意识强、为人和善等。      23.你怎么理解你应聘的职位？  应聘的职位和自身很适合，也有经验，可以和领导以及同事更完美的配合。    24.喜欢这份工作的哪一点？  有挑战性的工作，可以在为公司创造效益的时候更大的提升自己。      25.为什么要离职？  由于自身的原因，想找一个适合自己发挥的平台让自己能更好的发展。    26.说说你对行业、技术发展趋势的看法？  游戏开发行业是吸金度最高的行业，也是行业的领头羊技术含量较高，行业发展前景很好，在人们对物质条件满意的时候就会选择游戏，这就让我们游戏行业的用户量只增不减。    27.对工作的期望与目标是什么？  期望就是让自己得到更好的发展，目标是用自己的努力为公司创造更高的效益。    28.说说你的家庭？  家庭的生活经历构建了自身吃苦耐劳、认真负责、有始有终的良好素养。    29.你觉得你还欠缺什么？  社会经验不足、沟通能力需要提高    30.你欣赏哪种性格的人？  欣赏守信和负责的人，守信的人可以放心做朋友，负责的人可以一起共事。 |