代码随想录知识星球精华（最强八股文）第四版（面经篇））

代码随想录知识星球精华（最强八股文）第四版为九份PDF，分别是：

- 代码随想录知识星球八股文概述
- C++篇
- go篇
- Java篇
- 前端篇
- 算法题篇
- 计算机基础篇
- 问答精华篇
- 面经篇

本篇为最强八股文之面经篇。

面经总结

面经01-字节

> @author ironartisa

1. JAVA基础

1. 用户自己写一个String类，会发生什么?
2. sleep()和wait()的区别?
3. Object类里有哪些方法?
4. 讲一下equals()与hashcode()，什么时候重写，为什么重写，怎么重写?
5. Java多态，如何实现？动态绑定
6. Java如何保证多线程安全
7. I/O多路复用讲一下，epoll优势在哪，为什么，epoll水平触发与边缘触发

2. 操作系统

1. 进程与线程区别，进程通信方式，说一下socket，socket在本机和在网络通信区别
2. 多进程与多线程的区别以及使用场景?

3. 计算机网络

1、http和https的区别?

（1） 端口

HTTP的URL由“http://”起始且默认使用端口80
而HTTPS的URL由“https://”起始且默认使用端口443。

（2）安全性和资源消耗

HTTP协议运行在TCP之上，所有传输的内容都是明文，客户端和服务器端都无法验证对方的身份

HTTPS是运行在SSL/TLS之上的HTTP协议，SSL/TLS 运行在TCP之上。所有传输的内容都经过加密，加密采用对称加密，但对称加密的密钥用服务器方的证书进行了非对称
加密。所以说，HTTP 安全性没有 HTTPS高，但是 HTTPS 比HTTP耗费更多服务器资源;

（3）对称加密

密钥只有一个，加密解密为同一个密码，且加解密速度快，典型的对称加密算法有DES、AES等;

（4）非对称加密

密钥成对出现（且根据公钥无法推知私钥，根据私钥也无法推知公钥），加密解密使用不同密钥（公钥加密需要私钥解密，私钥加密需要公钥解密），相对对称加密速度较慢，典型的非对称加密算法有RSA、DSA等。

2、https用到了哪些加密技术?

见上

3、打开一个网页 使用到了哪些其他协议?

4、讲一下dns过程。给一个网址www.google.com，dns服务器如何逐级解析的?

DNS解析的过程就是寻找哪台机器上有你需要资源的过程。

（1）首先在本地域名服务器中查询IP地址;

（2）如果没有找到的情况下，本地域名服务器会向根域名服务器发送一个请求;

（3）如果根域名服务器也不存在该域名时，本地域名会向com顶级域名服务器发送一个请求，依次类推下去;

直到最后本地域名服务器得到google的IP地址并把它缓存到本地，供下次查询使用

从上述过程中，可以看出网址的解析是一个从右向左的过程:

com -> google.com -> www.google.com。

但是你是否发现少了点什么，根域名服务器的解析过程呢？事实上，真正的网址是www.google.com.，并不是我多打了一个 . ，这个 . 对应的就是根域名服务器，默认情况下所有的网址的最后一位都是 . ，既然是默认情况下，为了方便用户，通常都会省略，浏览器在请求DNS的时候会自动加上。

所以网址真正的解析过程为:

. -> .com -> google.com. -> www.google.com.

5、讲一下tcp四次挥手，time-wait干嘛的，close-wait干嘛的，在哪一个阶段?

1. 客户端发送 FIN
2. 服务器返回 ACK 号
3. 服务器发送 FIN
4. 客户端返回 ACK 号

这4步详细的来说，应该是如下：

（1）客户端发送一个FIN

用来关闭客户端到服务器的数据传送，此后客户端进入FIN_WAIT_1状态。

（2）服务器收到FIN后

进入CLOSE_WAIT状态。正常情况下会发送一个ACK给客户端，确认序号为收到序号+1（与SYN相同，一个FIN占用一个序号）。

（3）服务器发送一个FIN

用来关闭服务器到客户端的数据传送，服务器进入LAST_ACK状态。

（4）客户端收到FIN后

客户端进入TIME_WAIT状态，接着发送一个ACK给服务器，确认序号为收到序号+1，服务器进入CLOSED状态，完成四次挥手。

也就是说，**第二步进入了CLOSE_WAIT状态**。

Q: 我们看到CLOSE_WAIT出现在什么时候呢?

A: 在Sever端收到Client的FIN消息之后。

Q: 状态CLOSE_WAIT在什么时候转换成下一个状态呢?

A: 在Server端向Client发送FIN消息之后。

6、Tcp粘包拆包问题?

1. TCP 拆包的作用是将任务拆分处理，降低整体任务出错的概率，以及减小底层网络处理的压力。
2. 拆包过程需要保证数据经过网络的传输，又能恢复到原始的顺序。这中间，需要数学提供保证顺序的理论依据。
3. TCP 利用（发送字节数、接收字节数）的唯一性来确定封包之间的顺序关系。
4. 粘包是为了防止数据量过小，导致大量的传输，而将多个 TCP 段合并成一个发送。

7、TCP拥塞控制与流量控制区别?

（1）拥塞控制就是为了防止过多的数据注入到网络中

这样就可以使网络中的路由器或链路不致过载。拥塞控制所要做的都有一个前提，就是网络能够承受现有的网络负荷。拥塞控制是一个全局性的过程，涉及到所有的主机，所有的路由器，以及与降低网络传输性能有关的所有因素。相反的

（2）流量控制往往是点对点通信量的控制

是个端到端的问题。流量控制所要做到的就是抑制发送端发送数据的速率，以便使接收端来得及接收。

8、http状态码有哪些（1xx-5xx），那讲一下301与302的区别?

9、http，请求头，状态码?

10、现在我们视频面试用到哪些协议（DNS,HTTP,HTTPS,TCP,UDP），UDP用在哪里?

11、讲一下你了解的对称加密算法和非对称加密算法（DES,AES,RSA），非对称加密如何保证它的安全性?

12、xss攻击、cors攻击原理?

4. 数据库

1、Mysql了解吗，讲一下事务，那mysql是如何保证ACID的呢（答了undo-log,redo-log,加锁,mvcc），那讲一下MVCC

事务是逻辑上的一组操作，要么都执行，要么都不执行。

事务的四大特性原子性（Atomicity）、一致性（Consistency）、隔离性（Isolation）、持久性（Durability）。

我们以从A账户转账50元到B账户为例进行说明一下ACID，四大特性。

（1）原子性

定义：原子性是指一个事务是一个不可分割的工作单位，其中的操作要么都做，要么都不做。即要么转账成功，要么转账失败，是不存在中间的状态！

Q：如果无法保证原子性会怎么样?

OK，就会出现数据不一致的情形，A账户减去50元，而B账户增加50元操作失败。系统将无故丢失50元~

（2）隔离性

定义：隔离性是指多个事务并发执行的时候，事务内部的操作与其他事务是隔离的，并发执行的各个事务之间不能互相干扰。

Q：如果无法保证隔离性会怎么样?

OK，假设A账户有200元，B账户0元。A账户往B账户转账两次，金额为50元，分别在两个事务中执行。如果无法保证隔离性，会出现下面的情形：

如图所示，如果不保证隔离性，A扣款两次，而B只加款一次，凭空消失了50元，依然出现了数据不一致的情形！

（3）持久性

定义：持久性是指事务一旦提交，它对数据库的改变就应该是永久性的。接下来的其他操作或故障不应该对其有任何影响。

Q：如果无法保证持久性会怎么样?

在Mysql中，为了解决CPU和磁盘速度不一致问题，Mysql是将磁盘上的数据加载到内存，对内存进行操作，然后再回写磁盘。好，假设此时宕机了，在内存中修改的数据全部丢失了，持久性就无法保证 (redo log 保证)。

设想一下，系统提示你转账成功。但是你发现金额没有发生任何改变，此时数据出现了不合法的数据状态，我们将这种状态认为是数据不一致的情形。

（4）一致性

定义：一致性是指事务执行前后，数据处于一种合法的状态，这种状态是语义上的而不是语法上的。

Q：那什么是合法的数据状态呢?

例一:

A账户有200元，转账300元出去，此时A账户余额为-100元。你自然就发现了此时数据是不一致的，为什么呢？因为你定义了一个状态，余额这列必须大于0。

例二:

A账户200元，转账50元给B账户，A账户的钱扣了，但是B账户因为各种意外，余额并没有增加。你也知道此时数据

是不一致的，为什么呢？因为你定义了一个状态，要求A+B的余额必须不变。

2、实战解答

一、Mysql 怎么保证一致性的?

（原子性、隔离性、持久性和应用层代码判断）

OK，这个问题分为两个层面来说。

（1）从数据库层面

数据库通过原子性、隔离性、持久性来保证一致性。也就是说ACID四大特性之中，C(一致性)是目的，A(原子性)、I(隔离性)、D(持久性)是手段，是为了保证一致性，数据库提供的手段。数据库**必须要实现AID 三大特性，才有可能实现一致性**。

例如，原子性无法保证，显然一致性也无法保证。但是，**如果你在事务里故意写出违反约束的代码，一致性还是无法保证的**。例如，你在转账的例子中，你的代码里故意不给B账户加钱，那一致性还是无法保证。因此，还必须从应用层角度考虑。

（3）从应用层面

通过代码判断数据库数据是否有效，然后决定回滚还是提交数据！

2、Mysql 怎么保证原子性的？（undo log）

OK，是利用Innodb的 undo log 。

undo log 名为回滚日志，是实现原子性的关键，当事务回滚时能够撤销所有已经成功执行的sql语句，他需要记录你要回滚的相应日志信息。

例如：

当你 delete 一条数据的时候，就需要记录这条数据的信息，回滚的时候，insert 这条旧数据;

当你 update 一条数据的时候，就需要记录之前的旧值，回滚的时候，根据旧值执行 update 操作;

当你 insert 一条数据的时候，就需要这条记录的主键，回滚的时候，根据主键执行 delete 操作.

undo log 记录了这些回滚需要的信息，当事务执行失败或调用了 rollback，导致事务需要回滚，便可以利用undo log 中的信息将数据回滚到修改之前的样子。

3、Mysql 怎么保证持久性的?(redo log)

OK，是利用 Innodb 的 redo log。
正如之前说的，Mysql是先把磁盘上的数据加载到内存中，在内存中对数据进行修改，再刷回磁盘上。如果此时突然宕机，内存中的数据就会丢失。

Q：怎么解决这个问题啊?

A：简单啊，事务提交前直接把数据写入磁盘就行啊。

Q：这样做有什么问题吗?

A：只修改一个页面里的一个字节，就要将整个页面刷入磁盘，太浪费资源了。毕竟一个页面16kb大小，你只改其中一点点东西，就要将16kb的内容刷入磁盘，听着也不合理。

毕竟一个事务里的SQL可能牵涉到多个数据页的修改，而这些数据页可能不是相邻的，也就是属于随机IO。显然操作随机IO，速度会比较慢。

于是，决定采用 redo log 解决上面的问题。当做数据修改的时候，不仅在内存中操作，还会在 redo log 中记录这次操作。当事务提交的时候，会将 redo log 日志进行刷盘(redo log一部分在内存中，一部分在磁盘上)。当数据库宕机重启的时候，会将 redo log 中的内容恢复到数据库中，再根据 undo log 和 binlog 内容决定回滚数据还是提交数据。

Q：采用redo log的好处?

A：其实好处就是将redo log进行刷盘比对数据页刷盘效率高，具体表现如下:

redo log体积小，毕竟只记录了哪一页修改了啥，因此体积小，刷盘快。

redo log是一直往末尾进行追加，属于顺序IO。效率显然比随机IO来的快。

4、Mysql 怎么保证隔离性的?（锁 和 MVCC 机制）

OK，利用的是 锁 和 MVCC 机制。还是拿转账例子来说明，有一个账户表如下：

表名：t_balance

id	user_id	balance
1	A	200
2	B	0

其中id是主键，user_id为账户名，balance 为余额。还是以转账两次为例，如下图所示：

隔离级别为Repeatable Read

事务1	事务2
mysql> begin; Query OK, 0 rows affected	mysql> begin; Query OK, 0 rows affected
mysql> update t_balance set balance=balance-50 where user_id = 'A'; Query OK, 1 row affected	
	mysql> update t_balance set balance=balance-50 where user_id = 'A'; //由于获取不到表锁，阻塞住
mysql> update t_balance set balance=balance+50 where user_id = 'B'; Query OK, 1 row affected	
mysql> commit; Query OK, 0 rows affected	
	//由于事务1提交，获取到锁 Query OK, 1 row affected
	mysql> update t_balance set balance=balance+50 where user_id = 'B'; Query OK, 1 row affected
	mysql> commit; Query OK, 0 rows affected

由于user_id列没有索引，将t_balance整表锁上

ironartisa*

至于MVCC,即多版本并发控制(Multi Version Concurrency Control)，一个行记录数据有多个版本对快照数据，这些快照数据在undo log中。

如果一个事务读取的行正在做 DELELE 或者 UPDATE 操作，读取操作不会等行上的锁释放，而是读取该行的快照版本。
由于 MVCC 机制在可重复读(Repeateable Read)和读已提交(Read Commited)的MVCC表现形式不同，就不赘述了。

但是有一点说明一下，在事务隔离级别为读已提交(Read Commited)时，一个事务能够读到另一个事务已经提交的数据，是不满足隔离性的。但是当事务隔离级别为可重复读(Repeateable Read)中，是满足隔离性的。

MVCC

MVCC 旨在控制多版本并发读写，早期版本只有读读之间可以并发，而读写，写读，写写操作都是阻塞的，引入 mvcc 后，只有写写操作是阻塞的，mysql的实现原理是在数据节点有两个字段列，trid，rollbackpointer。

主要解决问题为在修改操作时，会加上行级锁，但上锁的开销较大，MVCC就是为了解决开销问题而存在，很多情况下，可以代替行级锁。

实现原理是因为每个事务进行修改操作时，都会往 undo.log 中写入内容，而历史版本可以通过前面的 trid 和 rollbackpointer 寻找到快照信息，具体读取哪个快照按隔离级别计算。

5、快照读在提交读和可重复读级别下有什么区别?

1. 在RC级别下每次都是读取最新的快照版本
2. 在RR级别下是事务开启时生成一个全局快照，后续的快照读都读取这个快照

（1）快照读（Snapshot Read）

MySQL 数据库，InnoDB 存储引擎，为了提高并发，使用 MVCC 机制，在并发事务时，通过读取数据行的历史数据版本，不加锁，来提高并发的一种不加锁一致性读（Consistent Nonlocking Read）。

（2）读提交（Read Committed）

1. 数据库领域，事务隔离级别的一种，简称 RC
2. 它解决“读脏”问题，保证读取到的数据行都是已提交事务写入的
3. 它可能存在“读幻影行”问题，同一个事务里，连续相同的 read 可能读到不同的结果集

（3）可重复读（Repeated Read）

1. 数据库领域，事务隔离级别的一种，简称RR
2. 它不但解决“读脏”问题，还解决了“读幻影行”问题，同一个事务里，连续相同的read读到相同的结果集

Q：6、在读提交（RC），可重复读（RR）两个不同的事务的隔离级别下，快照读有什么不同呢?

先说结论：

1. 事务总能够读取到，自己写入（update /insert /delete）的行记录
2. RC 下，快照读总是能读到最新的行数据快照，当然，必须是已提交事务写入的
3. RR 下，某个事务首次 read 记录的时间为 T，未来不会读取到 T 时间之后已提交事务写入的记录，以保证连续相同的 read 读到相同的结果集

可以看到：

1. 和并发事务的开始时间没关系，和事务首次read的时间有关；
2. 由于不加锁，和互斥关系也不大；

7、你提到了隐藏列有一个DB_ROW_ID，是干嘛的？那假设有10个update，到第九个回滚了，DB_ROLL_PTR如何做的，那提交了是否更新DB_ROLL_PTR?

TODO

8、讲一下索引及其底层，非叶子节点存储的是什么，只有b+树索引吗？（MEMORY是hash索引）

索引是为了快速查找数据，就跟一本书的目录一样。在目录中检索到具体章节的页码，然后直接跳转，就相当于根据索引快速找到数据在磁盘上的地址，然后去读取数据。

MySQL数据库索引采用的是B+Tree结构，在B-Tree结构上做了优化改造。

（1）B-Tree结构

1. 索引值和data数据分布在整棵树结构中
2. 每个节点可以存放多个索引值及对应的data数据
3. 树节点中的多个索引值从左到右升序排列

（2）B+Tree结构：

1. 非叶子节点不存储data数据，只存储索引值，这样便于存储更多的索引值

2. 叶子节点包含了所有的索引值和data数据
3. 叶子节点用指针连接，提高区间的访问性能

mysql索引：

从存储结构分：分成B+索引，hash索引

InnoDB 只支持显式创建 B+Tree 索引，对于一些热点数据页，

InnoDB 会自动建立自适应 Hash 索引，也就是在 B+Tree 索引基础上建立 Hash 索引， 这个过程对于客户端是不可控制的，隐式的。

9、讲一下Mysql聚集索引与非聚集索引，主键索引使用int与string有啥区别，你刚才说了索引底层b+树的结构，那么使用String会不会影响到这个结构

TODO

10、Mysql存储引擎的作用，事务隔离级别，分别存在什么问题

TODO

5. 算法

1、给定一个字符串数组[“hello”,”max”,”aello”,”world”], search(String s),判断字符串数组中是否存在一个字符串s1使得：s修改1个字符变为s1

2、二维数组，从左往右递增，从上到下递增，有重复数字，找一个数字是否存在于数组中。

剑指 Offer 04. 二维数组中的查找

思路：左下角开始遍历，大于target减小行数，小于target增大列数

1. arr[]数组，n个奇数，n个偶数，重排，奇数位置是奇数，偶数位置是偶数（从0开始），空间复杂度O（1），时间复杂度O（n）
2. 判断数组arr[]中是否存在2个数的异或为k（时间复杂度最优）
3. 之字型打印二叉树
4. 二维数组顺时针旋转90度
5. 找出数字字符串中最长的连续上升子序列（连续上升：前后两数之差为1）（LeetCode 300变种）

面经中提到的算法题，牛客大多数都有收录，多刷多练，面试很容易考原题。

- 按之字形顺序打印二叉树
- 旋转数组
- 寻找连续子序列
- 二维数组中的查找

6. 参考链接

mysql：

https://www.yuque.com/renyong-jmovm/ds/zgpx1l#985980fa
https://www.yuque.com/noahnyy/mysql/yl3xrb#edb07147

面经02-美团

> @author ironartisa

1. Java多线程

1.1 volatile

1、保证内存可见性。

可见性是指线程之间的可见性，一个线程修改的状态对另一个线程是可见的。也就是一个线程修改的结果，另一个线程马上就能看到;

实现原理：

（1）当对非volatile变量进行读写的时候，每个线程先从主内存拷贝变量到CPU缓存中，如果计算机有多个CPU，每个线程可能在不同的CPU上被处理，这意味着每个线程可以拷贝到不同的CPU cache中。

（2）volatile变量不会被缓存在寄存器或者对其他处理器不可见的地方，保证了每次读写变量都从主内存中读，跳过CPU cache这一步。当一个线程修改了这个变量的值，新值对于其他线程是立即得知的。

2、禁止指令重排

（1）指令重排序是JVM为了优化指令、提高程序运行效率，在不影响单线程程序执行结果的前提下，尽可能地提高并行度;

（2）指令重排序包括编译器重排序和运行时重排序;

（3）latile变量禁止指令重排序。针对volatile修饰的变量，在读写操作指令前后会插入内存屏障，指令重排序时不能把后面的指令重排序到内存屏障.

3、与synchronized 的对比

1. volatile 关键字是线程同步的轻量级实现，所以volatile性能肯定比synchronized关键字要好;
2. 但是volatile 关键字只能用于变量而 synchronized 关键字可以修饰方法以及代码块;
3. volatile 关键字能保证数据的可见性，但不能保证数据的原子性。synchronized 关键字两者都能保证;
4. volatile关键字主要用于解决变量在多个线程之间的可见性，而 synchronized 关键字解决的是多个线程之间访问资源的同步性。

1.2 threadLocal介绍，key是什么

通常情况下，我们创建的变量是可以被任何一个线程访问并修改的。如果想实现每一个线程都有自己的专属本地变量该如何解决呢?

JDK 中提供的 ThreadLocal 类正是为了解决这样的问题。

实现原理:

ThreadLocal 类主要解决的就是让每个线程绑定自己的值，可以将 ThreadLocal 类形象的比喻成存放数据的盒子，盒子中可以存储每个线程的私有数据。

如果你创建了一个 ThreadLocal 变量，那么访问这个变量的每个线程都会有这个变量的本地副本，这也是 ThreadLocal 变量名的由来。
他们可以使用 get() 和 set() 方法来获取默认值或将其值更改为当前线程所存的副本的值，从而避免了线程安全问题。

例子:

比如我们在同一个线程中声明了两个 ThreadLocal 对象的话，会使用 Thread 内部都是使用仅有那个 ThreadLocalMap 存放数据的， ThreadLocalMap 的 key 就是 ThreadLocal 对象，value 就是 ThreadLocal 对象调用 set 方法设置的值。

ThreadLocalMap 中使用的 key 为 ThreadLocal 的弱引用,而 value 是强引用。

所以，如果 ThreadLocal 没有被外部强引用的情况下，在垃圾回收的时候，key 会被清理掉，而 value 不会被清理掉。这样一来，ThreadLocalMap 中就会出现 key 为 null 的 Entry。假如我们不做任何措施的话，value 永远无法被 GC 回收，这个时候就可能会产生内存泄露。

ThreadLocalMap 实现中已经考虑了这种情况，在调用 set()、get()、remove() 方法的时候，会清理掉 key 为 null 的记录。使用完 ThreadLocal方法后 最好手动调用remove()方法

1.3 java的4种引用

JDK1.2之前:

Java中引用的定义很传统：如果reference类型的数据存储的数值代表的是另一块内存的起始地址，就称这块内存代表一个引用;

JDK1.2以后:

Java对引用的概念进行了扩充，将引用分为强引用、软引用、弱引用、虚引用四种（引用强度逐渐减弱）。

1、强引用(StrongReference)

以前我们使用的大部分引用实际上都是强引用，这是使用最普遍的引用;

如果一个对象具有强引用，那就类似于**必不可少的生活用品，垃圾回收器绝不会回收它;**

当内存空间不足，Java 虚拟机宁愿抛出 OutOfMemoryError 错误，使程序异常终止，也不会靠随意回收具有强引用的对象来解决内存不足问题。

2、软引用(SoftReference)

如果一个对象只具有软引用，那就类似于**可有可无的生活用品。**

如果内存空间足够，垃圾回收器就不会回收它。

如果内存空间不足了，就会回收这些对象的内存。只要垃圾回收器没有回收它，该对象就可以被程序使用。

软引用可用来实现内存敏感的高速缓存。

软引用可以和一个引用队列（ReferenceQueue）联合使用，如果软引用所引用的对象被垃圾回收，JAVA虚拟机就会把这个软引用加入到与之关联的引用队列中。

3、弱引用(WeakReference)

如果一个对象只具有弱引用，那就类似于**可有可无的生活用品**。

弱引用与软引用的区别在于：

只具有弱引用的对象拥有更短暂的生命周期。在垃圾回收器线程扫描它所管辖的内存区域的过程中，**一旦发现了只具有弱引用的对象，不管当前内存空间足够与否，都会回收它的内存**。不过，**由于垃圾回收器是一个优先级很低的线程**，因此不一定会很快发现那些只具有弱引用的对象。

弱引用可以和一个引用队列（ReferenceQueue）联合使用，如果弱引用所引用的对象被垃圾回收，Java虚拟机就会把这个弱引用加入到与之关联的引用队列中。

4、虚引用（PhantomReference）

"虚引用"顾名思义，就是形同虚设，与其他几种引用都不同，虚引用并不会决定对象的生命周期。如果一个对象仅持有虚引用，那么它就和没有任何引用一样，**在任何时候都可能被垃圾回收**。

虚引用主要用来跟踪对象被垃圾回收的活动。

虚引用与软引用和弱引用的一个区别在于：

虚引用必须和引用队列（ReferenceQueue）联合使用。当垃圾回收器准备回收一个对象时，如果发现它还有虚引用，就会在回收对象的内存之前，把这个虚引用加入到与之关联的引用队列中。

程序可以通过判断引用队列中是否已经加入了虚引用，来了解被引用的对象是否将要被垃圾回收。程序如果发现某个虚引用已经被加入到引用队列，那么就可以在所引用的对象的内存被回收之前采取必要的行动。

特别注意：

在程序设计中**一般很少使用弱引用与虚引用**，使用软引用的情况较多，这是因为软引用可以加速JVM对垃圾内存的回收速度，可以维护系统的运行安全，防止内存溢出（OutOfMemory）等问题的产生。

2. JVM

1、CMS垃圾回收器介绍

CMS（Concurrent Mark Sweep）收集器是一种以获取最短回收停顿时间为目标的收集器。它非常符合在注重用户体验的应用上使用。

CMS（Concurrent Mark Sweep）收集器是 HotSpot 虚拟机第一款真正意义上的并发收集器，它第一次实现了让垃圾收集线程与用户线程（基本上）同时工作。

从名字中的 Mark Sweep 这两个词可以看出，CMS 收集器是一种**标记-清除”算法**实现的，它的运作过程相比于前面几种垃圾收集器来说更加复杂一些。整个过程分为四个步骤：

（1）初始标记

暂停所有的其他线程，并记录下直接与 root 相连的对象，速度很快;

（2）并发标记

同时开启 GC 和用户线程，用一个闭包结构去记录可达对象。但在这个阶段结束，这个闭包结构并不能保证包含当前所有的可达对象。因为用户线程可能会不断的更新引用域，所以 GC 线程无法保证可达性分析的实时性。所以这个算法里会跟踪记录这些发生引用更新的地方;

（3）重新标记

重新标记阶段就是为了修正并发标记期间因为用户程序继续运行而导致标记产生变动的那一部分对象的标记记录，这个阶段的停顿时间一般会比初始标记阶段的时间稍长，远远比并发标记阶段时间短;

（4）并发清除

开启用户线程，同时 GC 线程开始对未标记的区域做清扫。

从它的名字就可以看出它是一款优秀的垃圾收集器，

主要优点： 并发收集、低停顿

但是它有下面三个明显的**缺点**：

1. 对 CPU 资源敏感
2. 无法处理浮动垃圾
3. 它使用的回收算法-“标记-清除”算法会导致收集结束时会有大量空间碎片产生

2、如何判断对象死亡

堆中几乎放着所有的对象实例，对堆垃圾回收前的第一步就是要判断哪些对象已经死亡（即不能再被任何途径使用的对象）。

（1）引用计数法

给对象中添加一个引用计数器

- 每当有一个地方引用它，计数器就加1
- 当引用失效，计数器就减1

任何时候计数器为0的对象就是不可能再被使用的。

（2）可达性分析算法

这个算法的基本思想就是通过一系列的称为 “GC Roots” 的对象作为起点，从这些节点开始向下搜索，节点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的话，则证明此对象是不可用的。

3、垃圾回收中的三色标记法

（1）首先，我们重新定义黑、白、灰三种颜色的含义

- 白色代表需要 GC 的对象;
- 黑色代表确定不需要 GC 的对象;

- 灰色代表可能不需要 GC 的对象，但是还未完成标记的任务，也可以认为是增量任务。

（2）在三色标记-清除算法中

准备工作：

一开始所有对象都染成白色初始化完成后，会启动标记程序。在标记的过程中，是可以暂停标记程序执行 Mutation，算法需要维护 3 个集合，白色集合、黑色集合、灰色集合。3 个集合是互斥的，对象只能在一个集合中。执行之初，所有对象都放入白色集合。

第一次执行：

算法将 Root 集合能直接引用的对象加入灰色集合；

往后执行：

不断从灰色集合中取出元素进行标记；

标记的过程主要分为 3 个步骤：

1. 如果对象在白色集合中，那么先将对象放入灰色集合；
2. 然后遍历节点的所有的引用对象，并递归所有引用对象；
3. 当一个对象的所有引用对象都在灰色集合中，就把这个节点放入为黑色集合。

当标记算法执行完成的时候，所有不需要 GC 的元素都会涂黑；标记算法完成后，白色集合内就是需要回收的对象。

4、cpu到达100%了，如何排查代码

参考链接：论线上如何排查一次CPU100%的情况 - 知乎

3. 操作系统

1、进程与线程的区别

进程是程序的一次执行过程，是系统运行程序的基本单位，因此进程是动态的。系统运行一个程序即是一个进程从创建，运行到消亡的过程。

线程与进程相似，但线程是一个比进程更小的执行单位。一个进程在其执行的过程中可以产生多个线程。与进程不同的是同类的多个线程共享进程的堆和方法区资源，但每个线程有自己的程序计数器、虚拟机栈和本地方法栈。

2、进程之间的通信方式

进程间通信，常见的方式主要有：

（1）基于内存的方式

1. 匿名管道
2. 信号量
3. 消息队列
4. 共享内存

（2）基于磁盘的方式

1. 文件
2. 命名管道

（3）基于网络的方式

1. socket
2. 消息队列
3. RPC
4. HTTP等各种网络协议

3、僵尸进程是什么，为什么会产生

如果一个进程已经终止，但是它的父进程尚未调用 wait() 或 waitpid() 对它进行清理，这时的进程状态称为僵死状态，处于僵死状态的进程称为僵尸进程;

产生原因:

任何一个子进程(init除外)在exit()之后，并非马上就消失掉，而是留下一个称为僵尸进程(Zombie)的数据结构，等待父进程处理。这是每个子进程在结束时都要经过的阶段。如果子进程在exit()之后，父进程没有来得及处理，这时用ps命令就能看到子进程的状态是“Z”;

解决办法:

当系统中出现了僵尸进程时，我们是无法通过 kill 命令把它清除掉的。但是我们**可以杀死它的父进程，让它变成孤儿进程，并进一步被系统中管理孤儿进程的进程收养并清理**。

4、虚拟内存到物理内存的寻址方式

CPU通过虚拟地址来访问主存，访问内存使用的物理地址，MMU通过将虚拟地址进行翻译，转化为物理地址，然后再用这个物理地址去访问内存数据。

5、页表的作用

页表是一种特殊的数据结构，记录着页面和页框的对应关系。（映射表）

页表的作用:

是内存非连续分区分配的基础，实现从逻辑地址转化成物理地址

页表实际上就是进程的虚存空间与系统中的物理存储空间的一个映射关系

在页式管理中，**页表的作用是实现从页号到物理块号的地址映射**，存储页表的作用是记录内存页面的分配情况

5、进程调度算法

为了确定首先执行哪个进程以及最后执行哪个进程以实现最大 CPU 利用率，计算机科学家已经定义了一些算法，它们是:

（1）先到先服务(FCFS)调度算法

从就绪队列中选择一个最先进入该队列的进程为之分配资源，使它立即执行并一直执行到完成或发生某事件而被阻塞放弃占用 CPU 时再重新调度。

（2）短作业优先(SJF)的调度算法

从就绪队列中选出一个估计运行时间最短的进程为之分配资源，使它立即执行并一直执行到完成或发生某事件而被阻塞放弃占用 CPU 时再重新调度。

（3）时间片轮转调度算法

时间片轮转调度是一种最古老，最简单，最公平且使用最广的算法，又称 RR(Round robin)调度。每个进程被分配一个时间段，称作它的时间片，即该进程允许运行的时间。

（4）多级反馈队列调度算法

前面介绍的几种进程调度的算法都有一定的局限性。如短进程优先的调度算法，仅照顾了短进程而忽略了长进程 。多级反馈队列调度算法既能使高优先级的作业得到响应又能使短作业（进程）迅速完成。，因而它是目前被公认的一种较好的进程调度算法，UNIX 操作系统采取的便是这种调度算法。

（5）优先级调度

为每个流程分配优先级，首先执行具有最高优先级的进程，依此类推。具有相同优先级的进程以 FCFS 方式执行。可以根据内存要求，时间要求或任何其他资源要求来确定优先级。

4. 计算机网络

1、为什么是三次握手，两次会怎么样

三次握手的目的是建立可靠的通信信道，说到通讯，简单来说就是数据的发送与接收，而三次握手最主要的目的就是双方确认自己与对方的发送与接收是正常的。

第一次握手：

Client 什么都不能确认；
Server 确认了对方发送正常，自己接收正常;

第二次握手：

Client 确认了：自己发送、接收正常，对方发送、接收正常；Server 确认了：对方发送正常，自己接收正常;

第三次握手：

Client 确认了：自己发送、接收正常，对方发送、接收正常； Server 确认了：自己发送、接收正常，对方发送、接收正常;

所以三次握手就能确认双发收发功能都正常，缺一不可。

2、Q：为什么连接建立需要三次握手，而不是两次握手?

防止失效的连接请求报文段被服务端接收，从而产生错误。

若建立连接只需两次握手，客户端并没有太大的变化，仍然需要获得服务端的应答后才进入 ESTABLISHED 状态，而服务端在收到连接请求后就进入 ESTABLISHED 状态。

此时如果网络拥塞，客户端发送的连接请求迟迟到不了服务端，客户端便超时重发请求，如果服务端正确接收并确认应答，双方便开始通信，通信结束后释放连接。

此时，如果那个失效的连接请求抵达了服务端，由于只有两次握手，服务端收到请求就会进入ESTABLISHED状态，等待发送数据或主动发送数据。但此时的客户端早已进入CLOSED状态，服务端将会一直等待下去，这样浪费服务端连接资源。

3、挥手为什么要有time-wait这个状态

为什么需要等待 2 倍最大报文段生存时间之后再关闭链接，原因有两个：

1. 保证 TCP 协议的全双工连接能够可靠关闭；
2. 保证这次连接的重复数据段从网络中消失，防止端口被重用时可能产生数据混淆。

4、tcp怎么保证数据的可靠性

（1）应用数据被分割成 TCP 认为最适合发送的数据块。

（2）TCP 给发送的每一个包进行编号，接收方对数据包进行排序，把有序数据传送给应用层。

（3）校验和

TCP 将保持它首部和数据的检验和。这是一个端到端的检验和，目的是检测数据在传输过程中的任何变化。如果收到段的检验和有差错，TCP 将丢弃这个报文段和不确认收到此报文段。

（4）TCP 的接收端会丢弃重复的数据

（5）流量控制

TCP 连接的每一方都有固定大小的缓冲空间，TCP的接收端只允许发送端发送接收端缓冲区能接纳的数据。当接收方来不及处理发送方的数据，能提示发送方降低发送的速率，防止包丢失。TCP 使用的流量控制协议是可变大小的滑动窗口协议。（TCP 利用滑动窗口实现流量控制）

（6）拥塞控制

当网络拥塞时，减少数据的发送。

（7）ARQ协议

也是为了实现可靠传输的，它的基本原理就是每发完一个分组就停止发送，等待对方确认。在收到确认后再发下一个分组。

（8）超时重传

当 TCP 发出一个段后，它启动一个定时器，等待目的端确认收到这个报文段。如果不能及时收到一个确认，将重发这个报文段。

5. 数据库

1、B+树的数据结构，B+树是怎么分裂的

B+ 树继承于 B 树，都限定了节点中数据数目和子节点的数目。B 树所有节点都可以映射数据，B+ 树只有叶子节点可以映射数据。

单独看这部分设计，看不出 B+ 树的优势。为了只有叶子节点可以映射数据，**B+ 树创造了很多冗余的索引**（所有非叶子节点都是冗余索引），这些冗余索引让 B+ 树在插入、删除的效率都更高，而且**可以自动平衡**，因此 B+ 树的所有叶子节点总是在一个层级上。

所以 B+ 树可以用一条链表串联所有的叶子节点，也就是索引数据，这让 B+ 树的范围查找和聚合运算更快。

6. 算法题

1、判断链表有没有环

思路：快慢指针

2、合并两个有序链表

思路：虚拟头结点

面经03

> @author习惯过了头

1、自我介绍

2、雪花算法生成id，为什么不用数据库自增

怎么判断用户是否登录、怎么部署项目、如果有多台服务器，怎么判断一个请求是否已经登录

3、volite了解吗（JVM）

4、synchronize原理，自旋锁等一系列锁了解吗

5、spring框架，前端是怎么知道调用后端的方法。

我说后端写有接口，是用response、request然后加注解。接着问加了这些后，为什么前端就可以访问到后端接口了

6、tcp三次握手，没有第3次会怎么样。

说一下滑动窗口作用，如果接收方的接收窗口告诉发送方已经没有缓存位置了，这时发送方就不会再发送数据，然后假设接收方又有100个缓存位置，然后发送报文给发送方，但是报文丢失，这样双方都在等待，就形成了死锁，怎么解决（我的回答是发送方发送探测报文）

7、知道 G1、CMS 吗（不知道，但是知道GC的3种回收算法：标记-清除、标记-复制、标记-整理）

> CMS 上面的面经有。。

8、算法题：股票系列，只能交易一次，求最大利润

算法导航原题，前几天刷动态规划还理解了一遍，用贪心做出来了，动态规划没写出来，太难受了

反问环节问了面试官什么是G1、CMS，原来是垃圾回收器，他说我说的那几种回收算法，他们没有在用了

今晚的面试，面试官无意间的一句话让我领悟了之前同学说的：面试官问他知不知道管程，他说不知道，然后面试官就默认操作系统这一块知识掌握不过关。

面试官看着自己的电脑说，jvm问了、java基础问了....还有什么没问呢。然后我想了一下，什么时候问jvm了，用来是volite关键字啊，恍然大悟

面试官还是很友好的，我应该是之前被大厂面试虐了，心理有点害怕面试。导致我现在有面试就很慌，手无足措。这次面试是我第一次看到面试官的样子，视频面试，之前面试了6，7次，即使是视频面试，面试官也不开摄像头😂

面经04

> @author weikunkun

二面（40分钟）

1. 自我介绍，项目中如何维护状态机的
    1. 然后详细阐述了每个状态的转化
    2. 异常状态的处理
    3. 还有些细节忘了，实习期间没有做好笔记，然后面试官说这个感觉状态机流转的有些问题
2. 然后就看到我简历上写了了解Linux命令（说面了那么多实习生，终于遇到一个写熟悉Linux命令的。。。），然后后面30分钟就开始共享屏幕写各种命令
    1. 查看自己电脑运行了多少idea进程
        - ps -ef | grep idea，然后数了数
    2. 然后问了idea启动的进程是哪个
        - 瞅了了一眼，说路径最短的那个。。。
    3. 进入这个目录里面
        - cd XXXX (空格需要转译)
        - 查看文件大小
            - 返回上一级，ls -h
                - 第一列代表啥意思
                - 当前用户、用户组、其他用户权限?
                - 可执行、可读、可写? 1，2，4?
                - 反正能说的都说了
        - 查看文件类型
            - 直接vim看来里面内容，然后感觉很RDB文件很像，说是二进制文件，同时是可执行文件。。。
            - 原来是想问file命令
    4. 把这个路径配置到当前的环境变量里面
        1. export PATH = $PATH:/xxxx
    5. 如何直接通过命令行执行idea
        1. 输入 idea（和配置MySQL到当前会话一个道理）
3. awk的问题，一个服务端业务逻辑耗时统计日志，统计服务端的耗时情况
    1. 不咋会awk，所以就说明了大概思路。
    2. 然后说先 cat xxxx.log | (xxxxxx) | printf(xxxx) > result.log
    3. 最后如果要排序的话，sort xxx result.log > result.log
    4. xxx表示不会写的东西。。。
4. 反问环节
    1. 有每周的技术分享会吗
    2. 对于codereview的流程，怎么避免需要频繁的rebase操作
    3. 业务场景，一些较为复杂的业务场景使用了哪些技术栈，会技术栈做一些自定义的横纵向拓展吗
    4. shell编程咋个练
        1. 不需要太过了解，但是对于一些提高效率的操作可以多试试，玩玩儿
    5. 本来还想问能不能进入下一轮，最后还是忍住了，随缘吧
5. 评价：
    1. 作为实习生，通过刚才的阐述能感觉到开发能力是属于还不错的类型，八股文这些，一面已经问过了，所以就不问你了
    2. 经验不太够，本来挺期待关于命令操作的，shell编程有待提高，一些基础命令用的还行。

面经05

> @author 习惯过了头

今天开始写分享会的内容，写了一些大致的内容：我的人生观、做事提前规划、大学三年历程、找实习的经历、关于学习方式、对于为知识付费的看法、对于大学阶段理财的看法、心理防御之精神胜利法、体验产品经理的工作。

下午学校就业指导课程安排我们去面试，我面了2家公司，规模不大，应该是小公司来的，都是Java开发岗位，2点45分去的时候，人太多了，就先去操场体测，回来后再面试。

具体问什么我不太记得了

第一家公司：

先让我自我介绍，然后问项目，遇到了什么难点，怎么解决，用了什么框架，了解Mybaits吗，知不知道微服务，还有问我个人网站是写内容，我说是写个人博客；Java基础、JVM、网络、操作系统这些都不问，面试了5分钟左右

反问环节我问了薪资和工作时间，薪资她说不知道，这个要问人事，工作时间是朝九晚六；然后问被录用了主要是做什么工作，她说是培训1,2个星期，熟悉框架后就参与项目开发。

第二家公司：

问的就比较深入一点了，先自我介绍，边介绍边看我简历。因为我介绍时提到软考中级证书，然后他问这是什么证书捂脸，然后问项目，介绍一下项目、数据库是怎么设计的、数据库索引了解吗、索引是什么、为什么是用b+树，而不用哈希或者其他数据结构。然后问框架怎么搭建、介绍一下Hibernate、说一下面向对象特性、说一下项目用到什么集合类、ArrayList 和 LinkedList的区别。

反问环节我问了工作内容个工作地点，他说该公司是有自己的产品，目前也有4个外包项目，主要是做智慧城市的项目。工作地点可以自己和人事沟通选择。然后我问后续还要不要去公司本部面试，他说不用，这次是技术面，过了就有人事联系，到时候薪资可以谈。

面试官还是挺友好的，说他只是比我们早工作2,3年，不用那么拘束，不用说什么有幸被贵公司录用之类的话，哈哈。都是出来混的😂

学校安排的公司规模不大，本来不怎么想去的，但是要签到，算入课程的成绩里。不过也有收获，大概知道中小企业、外包的面试难度，如果面试通过了，后续再更新，没有的话就更新到这里了，哈哈

分享会内容还有Java后端学习路线、书籍推荐、资源分享没写完，今天继续写。

说来惭愧，虽然“刷题导航”做了一遍，但是八大排序、图的深度、广度搜索我都不会。18号刷完题后就有点划水了这几天，得有一个新的规划，坚持下去。

工作室明天可以回去，今天要写一篇反思发给指导老师，不然不能回去😂

面经06

> @author 白夜

今天下午主要去面试去了，下面说说面试经历和感受

首先说下背景，目前是在一家小公司已经工作了三年，最近考虑到未来的职业规划和发展，遂决定离职，然后面试的公司是家中等规模的公司，听HR说开发人员人数在四百多

一开始的时候给了三个笔试题，分别是：

1. 设计一个线程安全的单例类
2. 实现一个内存安全的memcpy函数
3. 实现双向链表的插入函数

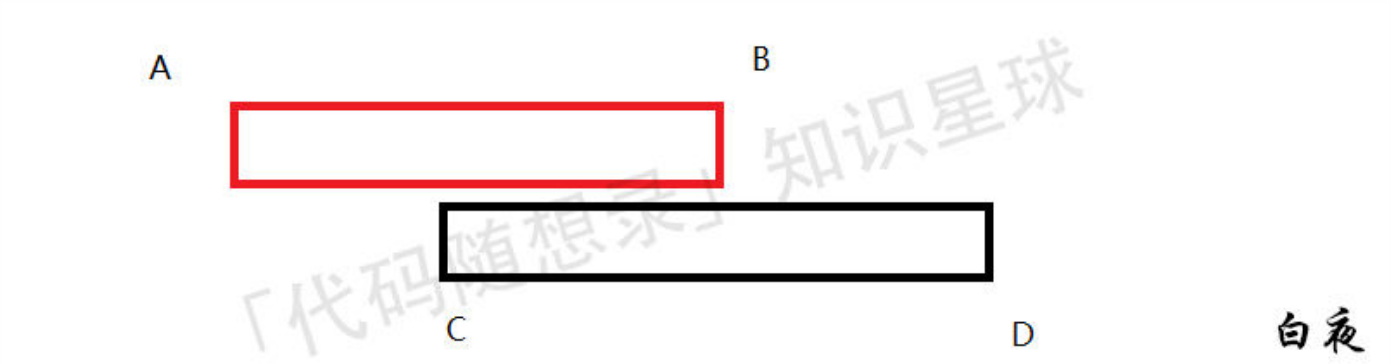
第一个问题：

```
class SafeSingle {
    public:
        SafeSingle* Instance() {
            Wait(event);
            if (_ins == nullptr)
                _ins = new SafeSingle;
            Release(event);
            return _ins;
        }
    private:
        static SafeSingle* _ins;
        static HANDLE event;
        SafeSingle() { event = CreateEvent(); }
}
```


其实单例模式平时没怎么用过，然后就是看了下设计模式，然后面试官问我为什么需要线程安全，我说如果_ins为nullptr的话可能会重复new，面试官问我还有吗，我就说想不出来了，然后问我每次获取实例的时候都wait一下不会造成资源浪费吗，我说那判断下如果为nullptr要new了的时候才wait，然后面试官没作答，直接问下一个问题了（当时挺虚的）

第二个问题：

我当时可能有点紧张，就没想到内存安全到底要咋整，就直接判断了一下目标内存地址和src地址是否为nullptr，然后面试官问我为什么需要内存安全的时候我才突然想到要是出现内存重叠就会出问题了，如下：



如果从A-B复制到C-D的话，就会出现问题，但是但是做这个题的时候没想到，后来面试官过程中想到了立马说了出来（哈哈）

第三个问题：

没啥好说的，没啥问题

然后就是问了一些C++的内容，问我C++11的特性了不了解，然后我说decltype类型推断之类的，然后我说我学的时候就学的C++，不知道哪些是11才新增的特性，他就问我知道智能指针吗，我说知道，问我有哪几种，我说shared_ptr和unique_ptr，然后问我就这两种吗，我说是的，问我没听说过weak_ptr吗，（其实知道有这个指针，但是当时看相关内容的时候觉得好像不会咋用到，就没太在意，果然学习还是要认真啊），然后我就说知道，但是

不太了解，然后面试官问我知不知道weak主要的应用场景，我说应该是用作标识用的吧，然后面试官就跳过这个问题了（还是不够扎实哈哈）

最后问我常用容器有哪些，我就说set，vector，list，map。问我list和map的区别，我说list是链表，map是红黑树，问我红黑树的特性，我说是颗二叉平衡搜索树（肯定不是他想要听到的答案哈哈），然后问我严格平衡吗，这里我就懵了，（我只知道红黑树就是二叉平衡搜索树，但是还没深入了解红黑树）我就直接说不知道

后面还问了些Windows的内存管理之类的，就不细述了哈哈

主要就是以上那些，然后面完之后问我有没有什么想了解的，我直接说刚刚面试中我觉得自己有不足之处，能不能请他客观的说下，然后面试官说我技术面窄了点啥的，我个人也确实是这么感觉的哈哈

最后有惊无险顺利拿到offer，嘿嘿

面经07

> @author weikunkun

一面（50min，终于记得录音了！！！）

自我介绍。

为什么投递Android，看简历更适合后端嘛，然后说了HR联系说没有HC了，问是否愿意转岗

1. HashMap的阐述 ⚠️
    1. 先笼统的说了整体上采用的什么数据结构，然后解决hash冲突的方式
    2. 然后详细阐述了HashMap的具体实现细节，节点的构成、触发扩容的情况、红黑树、CRUD的逻辑、equals、hashcode关系等
    3. 面试官调侃说讲的挺细的，然后自己感觉有些地方说的有些问题🐵
2. ArrayList的阐述 ⚠️
    1. 底层实现、扩容机制、尽量初始化时制定容量大小，避免频繁扩容的开销，
    2. 说了头部插入、中间插入、尾部插入的性能区别
3. 线程安全
    1. 线程安全需要解决的核心问题：
        1. 原子性
        2. 可见性
        3. 有序性
    2. 实现线程安全的方式
        1. 阻塞同步：加锁
        2. 非阻塞同步：cas
        3. 无同步方案：本地存储
4. final关键字
    1. 修饰
        1. 类
        2. 方法
        3. 成员变量
    2. 然后讲了保证了可见性
    3. 然后说了看了一些源码里面的finaly的修饰、譬如AQS的release、acquire方法、String类本身等
        1. 主要是为了保证自身的安全性的机制

5. Integer、int的区别 ⚠️
    1. 集合类的不支持基础数据类型存储
6. 问了泛型⚠️
    1. 泛型说的比较含糊
    2. 避免不必要的类型强转
    3. 设计的类或者说数据结构，更加具有通用性，支持各种数据类型：JDK定义的、自定义的
7. String、StringBuilder、StringBuffer
    1. 对与需要频繁进行拼接、截取的操作，使用StringBuilder、StringBuffer
    2. StringBuffer相比于StringBuilder，保证了单个API操作的线程安全
8. TCP、UDP的区别
9. 问了拥塞控制 ⚠️
    1. 有些模糊了，所以说的极其凌乱。。。
10. 单例模式
    1. 基础饿汉模式
    2. 基础懒汉模式
    3. DCL 注意指令重排
    4. 还想说其他的实现，被打断了，说差不多了
11. 单元测试
    1. 复杂逻辑
    2. 模拟请求，输出格式是否符合要求
    3. 避免线上出现问题，消耗的排查时间吧
12. 算法：求两个有序数组是否有交集
    1. 数据类型会明确吗，如果不明确的话，需要使用泛型。然后补充说int类型。
    2. 然后和面试官说了几个思路，以及对应的时间、空间复杂度
        1. 双指针
        2. set
            1. 判断是否存在
            2. 判断长度是否相同
        3. map
            1. 统计次数
    3. 说的思路里面挑了一个实现，然后验证
13. 逻辑题目：第一次遇到，还挺有趣的哈哈哈
    1. 100层楼，2个球，寻找球不会破的最小期望值
        1. 最开始没太理解，直接脱口说了二分
        2. 然后就合二分杠上了，各种思路都能绕到二分
        3. 面试官说二分不适合，二分适合球的数量不限
        4. 算是稍微理解了题目，然后开始往数组的模拟操作想，后面说用双指针
            1. 类似双指针
                1. 第一个指针确定大致范围
                2. 第二个指针借用第一个指针产生的信息，在一个更加精确的区间寻找
14. 反问：
    1. 问了些公司业务或者技术栈一些比较有挑战性的使用
        1. 说了Android的技术不像15年那段时间，现在都绝大多数都很成熟了，拿来即用
        2. 如果深入的话就需要考虑提高响应速度之类的，然后说了JVM的具体应用以及更底层的C++的内容

3. 说其实对于实习来说，公司内部是不强行规定方向，譬如你现在面的是Android，按照你的自己的方向，后面来了是完全可以转后端开发的
    1. emmmm？？？

面经08

> @author weikunkun

1. 自我介绍，说了自己的Github有个100多⭐的爬虫仓库，然后就开始问了爬虫相关的问题。。。
    1. 目前爬过的最难爬的内容
        1. 微博，讲了除了单纯的爬取之外，自己还添加了Cookies池和IP代理池，提升爬虫的抓取效果
        2. 问了对Scrapy了解多少
    2. 阐述先爬虫的核心内容（直讲了当初自己接触过的内容）
        1. 只讲了web端的爬虫
            1. 爬
                1. 直接网页渲染的，使用正则或则一些基于标签的第三方库来抓内容
                2. Ajax类型的，首先需要分析接口，需要哪些参数，如果复杂的话，还要分析一些参数是怎么生成的，通常会和网页的js有关
            2. 其他
                1. 怎么和反爬斗智斗勇
    3. 反爬措施了解哪些
        1. 说了基于IP的策略，封IP
        2. 根据请求头来封策略，譬如User-Agent、Refer、Host这种
        3. 根据Cookie来封
        4. 图片识别、文字识别、9宫格这些（当初没说）
2. 项目内容介绍
    1. 就根据：业务介绍、实现细节、难点解决这几个步骤讲的
    2. 然后仔细问了线程池、Redis分布式锁、Kafka
3. JMM阐述下，有多少说多少
    1. 定义：一种内存模型，用来屏蔽各种硬件和操作系统的不同，保证Java程序在各个平台下对内存的访问都能达到一致的结果。
    2. 目的：解决由于多线程通过共享内存进行通信时，存在的原子性、可见性（缓存一致性）以及有序性问题
    3. 接着讲了主存和工作内存，以及JMM是怎么规定对主存中变量的读写
    4. 然后详细展开来讲了原子性、可见性、有序性。以及对应的支持
4. MySQL索引说一下，有多少说多少
    1. 基于InnoDB而言，索引的类型：hash索引、B+树索引、全文索引（这个没有深入了解）
    2. 然后作中将了B+树索引
        1. 先阐述了B+树的概念
        2. 然后阐述了磁盘如何利用局部性原理做的磁盘的预读，然后将了MySQL是怎么巧妙的利用磁盘的预读来设计B+树节点的大小，从而降低IO的次数
    3. 然后将了索引的类型
        1. 聚集索引、辅助索引、联合索引、唯一索引
    4. 使用索引的注意事项
        1. 选择区分度大的字段 count(distinct(field) / count(*))
        2. 最左匹配原则
        3. = 可以乱序
        4. 索引列保持干净，不参与运算，注意类型的匹配
        5. 尽量横向拓展，而不是新增索引

kafka还是怵，特别怕问到kafka相关的超出当前认知的内容。

weikunkun

面经09

> @author 壮

MySQL

1、都知道数据库索引采用B+树而不是B树，原因也有很多，主要原因是什么？【11】

主要原因：

B+树只要遍历叶子节点就可以实现整棵树的遍历，而且在数据库中基于范围的查询是非常频繁的，而B树只能中序遍历所有节点，效率太低

2、文件索引和数据库索引为什么使用B+树【12】

（1）方便扫库

B树必须用中序遍历的方法按序扫库，而B+树直接从叶子结点挨个扫一遍就完了，B+树支持range-query非常方便，而B树不支持，这是数据库选用B+树的最主要原因。

（2）B+tree的磁盘读写代价更低

B+tree的内部结点并没有指向关键字具体信息的指针(红色部分)，因此其内部结点相对B 树更小。如果把所有同一内部结点的关键字存放在同一块盘中，那么盘块所能容纳的关键字数量也越多。一次性读入内存中的需要查找的关键字也就越多，相对来说IO读写次数也就降低了；

（3）B+tree的查询效率更加稳定

由于内部结点并不是最终指向文件内容的结点，而只是叶子结点中关键字的索引，所以，任何关键字的查找必须走一条从根结点到叶子结点的路。所有关键字查询的路径长度相同，导致每一个数据的查询效率相当

3、听说过视图吗？那游标呢？【13】

视图（子查询）是一种虚拟的表，通常是有一个表或者多个表的行或列的子集，具有和物理表相同的功能；

两类：单表视图一般用于查询和修改，会改变基本表的数据； 多表视图一般用于查询，不会改变基本表的数据；

作用：

（1）简化了操作

把经常使用的数据定义为视图【设计聚合函数相关操作】

（2）安全性

用户只能查询和修改能看到的数据

（3）逻辑上的独立性

屏蔽了真实表的结构带来的影响【动态的数据的集合，数据是随着基表的更新而更新】

游标是对查询出来的结果集作为一个单元来有效的处理。游标是处理结果集的一种机制，定位到结果集中的某一行，多数据进行读写；

作用如下：

1. 定位到结果集中的某一行
2. 对当前位置的数据进行读写
3. 可以对结果集中的数据单独操作，而不是整行执行相同的操作
4. 是面向集合的数据库管理系统和面向行的程序设计之间的桥梁

4、MySQL中为什么要有事务回滚机制？【14】

恢复机制是通过回滚日志（undo log）实现的，所有事务进行的修改都会先记录到这个回滚日志中，然后在对数据库中的对应行进行写入。当事务已经被提交之后，就无法再次回滚了。

回滚日志作用：

1. 能够在发生错误或者用户执行 ROLLBACK 时提供回滚相关的信息
2. 在整个系统发生崩溃、数据库进程直接被杀死后，当用户再次启动数据库进程时，还能够立刻通过查询回滚日志将之前未完成的事务进行回滚，这也就需要回滚日志必须先于数据持久化到磁盘上，是我们需要先写日志后写数据库的主要原因。

5、数据库引擎InnoDB与MyISAM的区别【15】

事务: InnoDB 是事务型的，可以使用 Commit 和 Rollback 语句。

索引：InnoDB 聚簇索引关键字中存放的是数据，而MyISAM聚簇索引中存放的是数据对应的地址。

并发: MyISAM 只支持表级锁，而 InnoDB 还支持行级锁。

外键: InnoDB 支持外键。

备份: InnoDB 支持在线热备份。

崩溃恢复: MyISAM 崩溃后发生损坏的概率比 InnoDB 高很多，而且恢复的速度也更慢。

其它特性: MyISAM 支持压缩表和空间数据索引。

面经10

> @author 壮

数据库：

索引：

1. 索引与主键的区别
2. 索引的优缺点
3. 使用索引的原因
4. 聚簇索引和非聚簇索引的区别
5. 索引什么时候是否会失效，最左匹配原则是什么？
6. 索引的类型
7. 建立索引的场景

计算机网络：

1. HTTP常见的状态码
2. HTTP如何禁用缓存？如何确认缓存？

数据库

索引

1、索引与主键的区别

1. 主键是为了标识数据库记录**唯一性**,不允许记录重复,且键值不能为空,主键也是一个**特殊索引**
2. 数据表中只允许有一个主键,但是可以有多个索引
3. 使用主键的数据库会自动创建**主索引**,也可以在非主键上创建索引,方便查询效率
4. **索引可以提高查询速度,它就相当于字典的目录**,可以通过它很快查询到想要的结果,而不需要进行全表扫描
5. 主键索引外索引的值可以为空. 主键也可以由多个字段组成,组成复合主键,同时主键肯定也是唯一索引
6. 唯一索引则表示该索引值唯一,可以由一个或几个字段组成,一个表可以有多个唯一索引

2、索引的优点与缺点

优点：

1. 创建唯一性索引，保证数据库表中每一行数据的唯一性
2. 大大加快数据的检索速度，这也是创建索引的最主要的原因
3. 加速表和表之间的连接，特别是在实现数据的参考完整性方面特别有意义。
4. 在使用分组和排序子句进行数据检索时，同样可以显著减少查询中分组和排序的时间。
5. 通过使用索引，可以在查询的过程中使用优化隐藏器，提高系统的性能。

缺点：

1. 创建索引和维护索引要耗费时间，这种时间随着数据量的增加而增加
2. 索引需要占物理空间，除了数据表占数据空间之外，每一个索引还要占一定的物理空间，如果要建立聚簇索引，那么需要的空间就会更大
3. 当对表中的数据进行增加、删除和修改的时候，索引也要动态的维护，降低了数据的维护速度

3、使用索引的原因[优点]

1. 通过创建唯一性索引，可以保证数据库表中每一行数据的唯一性；可以大大加快数据的检索速度，这也是创建索引的最主要的原因
2. 帮助服务器避免排序和临时表
3. 将随机IO变为顺序IO
4. 可以加速表和表之间的连接，特别是在实现数据的参考完整性方面特别有意义

4、聚簇索引和非聚簇索引的区别?

聚簇索引：

1. 以主键创建的索引
2. 在叶子节点存储的是表中的数据
3. 不是一种索引类型,而是一种存储数据的方式.Innodb的聚簇索引是在同一个数据结构中保存了索引和数据
4. 聚簇索引记录的排序顺序和索引的排序顺序一致，所以查询效率快

【因为只要找到第一个索引值记录，其余的连续性的记录在物理表中也会连续存放，一起就可以查询到】

缺点：

新增比较慢

【因为为了保证表中记录的物理顺序和索引顺序一致，在记录插入的时候，会对数据页重新排序】

非聚簇索引：

1. 以非主键创建的索引（二级索引）
2. 在叶子节点存储的是主键和索引列
3. 索引的逻辑顺序与磁盘上行的物理存储顺序不同，非聚集索引在叶子节点存储的是主键和索引列

【当我们使用非聚集索引查询数据时，需要拿到叶子上的主键再去表中查到想要查找的数据。这个过程就是我们所说的回表】

5、索引什么时候是否会失效，最左匹配原则是什么?

最左匹配原则：

最左优先，以最左边的为起点任何连续的索引都能匹配上，如不连续，则匹配不上；

索引失效在以下情况：

1. 不满足最左【前缀】匹配规则
2. 在索引列上做任何操作如计算、函数、（手动或自动）类型转换等操作，会导致索引失效而进行全表扫描
3. 使用不等于（!= 、<>）
4. like 中以通配符开头('%abc')
5. 字符串不加单引号索引失效
6. or 连接索引失效

6、索引的类型

（1）从存储来看

全文索引：

目前只有MyISAM引擎支持。其可以在CREATE TABLE ， ALTER TABLE ， CREATE INDEX 使用，不过目前只有CHAR、VARCHAR ， TEXT 列上可以创建全文索引。

哈希索引 ：

由于HASH的唯一（几乎100%的唯一）及类似键值对的形式，很适合作为索引。 HASH索引可以一次定位，不需要像树形索引那样逐层查找,因此具有极高的效率。但是，这种高效是有条件的，即只在“=”和“in”条件下高效，对于范围查询、排序及组合索引仍然效率不高。

B树索引：

BTREE索引就是一种将索引值按一定的算法，存入一个树形的数据结构中（二叉树），每次查询都是从树的入口root开始，依次遍历node，获取leaf。这是MySQL里默认和最常用的索引类型。

R树索引 ：

RTREE在MySQL很少使用，仅支持geometry数据类型，支持该类型的存储引擎只有MyISAM、BDb、InnoDb、NDb、Archive几种。 相对于BTREE，RTREE的优势在于范围查找。

（2）应用层次来看

普通索引：

即一个索引只包含单个列，一个表可以有多个单列索引。

唯一索引：

索引列的值必须唯一，但允许有空值。

复合索引【联合索引】：

一个索引包含多个列。所有的索引列只可以进行最左前缀匹配【通过查询来反推索引,以使某个固定的查询可以尽可能的命中索引以提高查询速度】

从表记录的排列顺序和索引的排列顺序是否一致来划分：

聚集索引：表记录的排列顺序和索引的排列顺序一致。

非聚集索引：表记录的排列顺序和索引的排列顺序不一致。

（3）其他

前缀索引：

在对一个比较长的字符串进行索引时,可以仅索引开始的一部分字符,这样可以大大的节约索引空间,从而提高索引效率.但是这样也会降低索引的选择性

覆盖索引：

当一个索引包含(或者说是覆盖)需要查询的所有字段的值时
【在InnoDB存储引 擎中，如果不是主键索引，叶子节点存储的是主键+列值。最终还是要“回表”，也就是要通过主键再查找一次,这样就会比较慢。覆盖索引就是把要查询出的列和索引是对应的，不做回表操作！】

7、建立索引的场景

在最频繁使用的、用以缩小查询范围的字段,需要排序的字段上建立索引。

不宜：

1. 对于查询中很少涉及的列或者重复值比较多的列
2. 对于一些特殊的数据类型，不宜建立索引，比如文本字段（text）等

计算机网络

2、HTTP常见的状态码

状态码	类别	含义	具体实例
1XX	Informational（信息性状态码）	接收的请求正在处理	100 Continue ： 表明到目前为止都很正常，客户端可以继续发送请求或者忽略这个响应。
2XX	Success（成功状态码） 请	请求正常处理完毕	200 OK
3XX	Redirection（重定向状态码） 需	需要进行附加操作以完成请求	301 Moved Permanently ： 永久性重定向 302 Found ： 临时性重定向
4XX	Client Error（客户端错误状态码）	服务器无法处理请求	400 Bad Request ： 请求报文中存在语法错误。 403 Forbidden ： 请求被拒绝。 404 Not Found
5XX	Server Error（服务器错误状态码）	服务器处理请求出	500 Internal Server Error ： 服务器正在执行请求时发生错误。 503 Service Unavailable ： 服务器暂时处于超负载或正在进行停机维护，现在无法处理请求。

3、HTTP如何禁用缓存？如何确认缓存？

HTTP/1.1 通过 Cache-Control 首部字段来控制缓存。

（1）禁止进行缓存

no-store 指令规定不能对请求或响应的任何一部分进行缓存。

（2）强制确认缓存

no-cache 指令规定缓存服务器需要先向源服务器验证缓存资源的有效性，只有当缓存资源有效时才能使用该缓存对客户端的请求进行响应。

Cache-Control: no-store [禁止进行缓存]/no-cache [强制确认缓存]

补充：缓存中的私有公有字段

Cache-Control:private [将资源作为私有缓存，只能被单独用户使用，一般存储在用户浏览器中]/public[将资源作为公共缓存，可以被多个用户使用，一般存储在代理服务器中]

面经11

> @author 壮

问题汇总：

0、系统调用与库函数的区别

1、场景题：所在的公司选择MySQL数据库作数据存储，一天五万条以上的增量，预计运维三年，你有哪些优化手段?

2、SQL语言四大类别

3、垂直分表和水平分表

4、淘汰策略【页式系统】

5、任务（进程）调度算法

操作系统

系统调用与库函数的区别

1、系统调用

（1）操作系统提供了一组特殊接口——系统调用

通过这组接口用户程序可以使用操作系统内核提供的各种功能。例如分配内存、创建进程、实现进程之间的通信等。

（2）用户程序向操作系统提出请求的接口就是系统调用

所有的操作系统都会提供系统调用接口，只不过不同的操作系统提供的系统调用接口各不相同。

（3）系统调用按照功能分类

进程控制、进程间通信、文件系统控制、存储管理、网络管理、套接字控制、用户管理等。

2、库函数

对系统调用的一种封装，因为系统调用是面对的是操作系统，系统包括Linux、Windows等，如果直接系统调用，会影响程序的移植性，所以这里使用了库函数。

用户编程接口API:

前面提到利用系统调用接口程序可以访问各种资源，但在实际开发中程序并不直接使用系统调用接口，

而是使用用户编程接口（API）【各种库（最重要的就是C库）中的函数】。

不直接使用系统调用接口原因如下:

1. 系统调用接口功能非常简单，无法满足程序的需求
2. 不同操作系统的系统调用接口不兼容，程序移植时工作量大

页面淘汰算法

1、淘汰策略（页式系统）

（1）置换算法

当主存块已全部用完，添加新的一页进主存，选择淘汰哪一页的规则

（2）颠簸（抖动）

导致系统效率急剧下降的主存和辅存之间的频繁页面置换现象。

（3）缺页（中断）率

f' = f/a（缺页次数/访问总次数）

（4）固定空间页面调度

系统为每一个进入主存的程序分配的主存块数m是固定的；

2、最佳算法（OPT算法）

每次选择的淘汰页面将是以后永不使用，或者在最长时间内都不再访问的页面（保证最低的缺页率）

该算法是无法实现的

原因：操作系统无法提前判断页面访问序列

一般该算法作为衡量各种具体算法优劣的标准
淘汰时参考的是未来将要执行的，但是只有在进程执行过程中才能知道接下来会访问到的是哪个页面

3、先进先出算法（FIFO算法）

每次淘汰选择在主存中居留时间最长（即进入最早）的一页淘汰。

实现的方法：

把调入内存的页面根据调入的先后顺序排成一个队列，需要换出页面时选择队头；页面队列的最大长度取决于系统为进程分配了多少个内存块。

Belady异常：

当为进程分配的物理块数增大时，缺页次数不降反增的异常现象。（只有FIFO算法有该现象）

缺点：

该算法与进程实际运行时的规律不适用，因为先进入的页面可能经常被访问，算法性能较差。

4、最近最久未使用淘汰算法(Least Recently Used)（LRU算法）

每次淘汰的页面时最近最久未使用的页面。

实现方法：

（1）计数器：

赋予每个页面对应的页表项中，用访问字段记录该页面自上次被访问以所经历的时间t（需要硬件支持，实现困难，开销大）

（2）堆栈（软件实现）

利用那个栈来登记主存中可淘汰的页号。每当一个页面被访问过，就立即将它的页号记载页号栈的顶部，而将栈中原有的页号依次下移。(淘汰的页面对应栈底页面)

5、LRU近似算法（软硬相结合）

在一个存储块有一个引用位：

当某块中的页面被访问时，这一位由硬件自动置1，而页面管理软件周期性（T）将所有引用位重新置“0”（在T内，访问过的页面为1，未访问过为0，需要置换一页时，淘汰应用那个位为0）

替换指针：

总是指向最近被替换的页所在的块号。（发生缺页中断时，从替换指针的下一位位置开始查找：引用位为1，将其置为0；直至找到第一个0为止进行替换）

6、最不经常使用淘汰算法（LFU算法）

淘汰最近应用次数最少的页；

实现方法：

为对应的每一页设置一个计数器，对每一页访问一次后，就使它相应的计数器 +1；当需要淘汰一页时，选择计数器最小对应的页面；

过一段时间后，将所有计数器一律清除；（实现不难，但是代价较高）

任务调度算法

任务（进程）调度算法

非抢占式：FCFC 、SJF

抢占式：SRTN

1、先来先服务 first-come first-serverd（FCFS）

非抢占式的调度算法，按照请求的顺序进行调度。

有利于长作业，但不利于短作业，因为短作业必须一直等待前面的长作业执行完毕才能执行，而长作业又需要执行很长时间，造成了短作业等待时间过长。

2、短作业优先 shortest job first（SJF）

非抢占式的调度算法，按估计运行时间最短的顺序进行调度。

长作业有可能会饿死，处于一直等待短作业执行完毕的状态。因为如果一直有短作业到来，那么长作业永远得不到调度。

3、最短剩余时间优先 shortest remaining time next（SRTN）

最短作业优先的抢占式版本，按剩余运行时间的顺序进行调度。当一个新的作业到达时，其整个运行时间与当前进程的剩余时间作比较。

如果新的进程需要的时间更少，则挂起当前进程，运行新的进程。否则新的进程等待。

4、时间片轮转

将所有就绪进程按 FCFS 的原则排成一个队列，每次调度时，把 CPU 时间分配给队首进程，该进程可以执行一个时间片。

当时间片用完时，由计时器发出时钟中断，调度程序便停止该进程的执行，并将它送往就绪队列的末尾，同时继续把 CPU 时间分配给队首的进程。

时间片轮转算法的效率和时间片的大小有很大关系：

因为进程切换都要保存进程的信息并且载入新进程的信息，如果时间片太小，会导致进程切换得太频繁，在进程切换上就会花过多时间。
而如果时间片过长，那么实时性就不能得到保证。

5、优先级调度

为每个进程分配一个优先级，按优先级进行调度。

为了防止低优先级的进程永远等不到调度，可以随着时间的推移增加等待进程的优先级。

6、多级反馈队列

一个进程需要执行 100 个时间片，如果采用时间片轮转调度算法，那么需要交换 100 次。

多级队列是为这种需要连续执行多个时间片的进程考虑，它设置了多个队列，每个队列时间片大小都不同，例如 1,2,4,8,..。进程在第一个队列没执行完，就会被移到下一个队列。

这种方式下，之前的进程只需要交换 7 次。每个队列优先权也不同，最上面的优先权最高。因此只有上一个队列没有进程在排队，才能调度当前队列上的进程。

可以将这种调度算法看成是时间片轮转调度算法和优先级调度算法的结合

数据库

一、场景题：所在的公司选择MySQL数据库作数据存储，一天五万条以上的增量，预计运维三年，你有哪些优化手段?

1. 设计良好的数据库结构，允许部分数据冗余，尽量避免join查询，提高效率。选择合适的表字段数据类型和存储引擎，适当的添加索引。
2. MySQL库主从读写分离。
3. 找规律分表，减少单表中的数据量提高查询速度。
4. 添加缓存机制，比如Memcached，Apc等。
5. 不经常改动的页面，生成静态页面。
6. 书写高效率的SQL。比如 SELECT * FROM TABEL 改为 SELECT field_1, field_2, field_3 FROM TABLE。

二、SQL语言四大类别：

1. 数据定义语言（DDL）
2. 数据操纵语言（DML）
3. 数据查询语言（DQL）
4. 数据控制语言（DCL）:授予或回收访问数据库的某种特权，并控制数据库操纵事务发生的时间及效果。

类别	DDL	DML	DQL	DCL
创建对象	表、视图、索引同义词、聚簇			
包含语句	Create Drop Alter	1) 插入：INSERT 2) 更新：UPDATE 3) 删除：DELETE	Select From Where	GRANT 授权 ROLLBACK 回滚
关于是否能回滚	隐性提交的！不能rollback；操作立即生效	必须**提交**才能生效！执行的操作会放到**回滚段**，可回滚		

三、垂直分表和水平分表

分表的目的：

降低单次查询数据量，从而提高查询速度【问题：单表数据量过大的问题，这就降低了查询速度，影响了客户体验】

1、水平分表：

概念：

一条记录一条记录切断分出来！【按照行拆分，将一张表拆分成多张表】

拆分的方法：

数据取模，按照模相同的放到同一张表中【随机分表】；时间维度分表【连续分表】

优点：

1. 表关联基本能够在数据库端全部完成；
2. 不会存在某些超大型数据量和高负载的表遇到瓶颈的问题；
3. 应用程序端整体架构改动相对较少；
4. 事物处理相对简单；
5. 只要切分规则能定义好，基本上较难遇到扩展性限制；

缺点：

1. 切分规则相对更为复杂，很难抽象出一个能满足整个数据库的切分规则；
2. 后期数据的维护难度有所增加，人为手工定位数据更为困难；
3. 应用系统各模块耦合度较高，可能会对后面数据的迁移拆分造成一定的困难。

2、垂直分表：

概念：

把常用的、不常用的字段很长的拆出来！是指表数据列的拆分，把一张列比较多的表拆分成多张表。

表的记录并不多，但是字段却很长，表占用空间很大，检索表的时候需要执行大量的IO,严重降低了性能。

这时候需要把大的字段拆分到另外一个表，并且该表与原表是一对一的关系。

垂直拆分规则：

1. 把不常用的字段单独放在一个表；
2. 把text,blob等大字段拆分出来放在附表中；
3. 经常组合查询的列放在一张表中；

优点：

1. 数据库的拆分简单明了，拆分规则明确
2. 应用程序模块清晰明确，整合容易
3. 数据维护方便易行，容易定位

缺点：

1. 部分表关联无法在数据库级别完成，需要在程序中完成；
2. 对于访问极其频繁且数据量超大的表仍然存在性能瓶颈，不一定满足需求；
3. 事务处理相对更为复杂；
4. 切分达到一定程度后，扩展性会遇到限制；
5. 过度切分可能会带来系统过度复杂而难以维护；

面经12

> @author 壮

问题汇总：

数据库三范式

锁：乐观锁与悲观锁、典型锁、死锁相关问题

拥塞控制：四种拥塞算法

数据库

数据库三范式

1、简单归纳

第一范式（1NF）：字段不可分
第二范式（2NF）：有主键，非主键字段依赖主键
第三范式（3NF）：非主键字段不能相互依赖

2、解释

1NF：原子性，字段不可再分,否则就不是关系数据库
2NF：唯一性，一个表只说明一个事物
3NF：每列都与主键有直接关系，不存在传递依赖

锁

乐观锁和悲观锁

1、概念

乐观锁和悲观锁是两种思想，用于解决并发场景下的数据竞争问题。

乐观锁：

乐观锁在操作数据时非常乐观，认为别人不会同时修改数据。因此乐观锁不会上锁，只是在执行更新的时候判断一下在此期间别人是否修改了数据：如果别人修改了数据则放弃操作，否则执行操作。

悲观锁：

悲观锁在操作数据时比较悲观，认为别人会同时修改数据。因此操作数据时直接把数据锁住，直到操作完成后才会释放锁；上锁期间其他人不能修改数据。

2、区别

悲观锁的实现方式是加锁，加锁既可以是对代码块加锁（如Java的synchronized关键字），也可以是对数据加锁（如MySQL中的排它锁）。

乐观锁的实现方式主要有两种：CAS机制和版本号机制

3、应用场景

（1）功能限制

与悲观锁相比，乐观锁适用的场景受到了更多的限制，无论是CAS还是版本号机制。

例如，CAS只能保证单个变量操作的原子性，当涉及到多个变量时，CAS是无能为力的，而synchronized则可以通过对整个代码块加锁来处理。再比如版本号机制，如果query的时候是针对表1，而update的时候是针对表2，也很难通过简单的版本号来实现乐观锁。

（2）竞争激烈程度

如果悲观锁和乐观锁都可以使用，那么选择就要考虑竞争的激烈程度：

当竞争不激烈时(出现并发冲突的概率小)：

乐观锁更有优势，因为悲观锁会锁住代码块或数据，其他线程无法同时访问，影响并发，而且加锁和释放锁都需要消耗额外的资源。

当竞争激烈时(出现并发冲突的概率大)：

悲观锁更有优势，因为乐观锁在执行更新时频繁失败，需要不断重试，浪费CPU资源。

（3）实际操作

悲观锁：

先获取锁，再进行业务操作，一般就是利用类似 SELECT … FOR UPDATE 这样的语句，对数据加锁，避免其他事务意外修改数据。

当数据库执行SELECT … FOR UPDATE时会获取被select中的数据行的行锁，select for update获取的行锁会在当前事务结束时自动释放，

因此必须在事务中使用。

乐观锁：

先进行业务操作，只在最后实际更新数据时进行检查数据是否被更新过。

Java 并发包中的AtomicFieldUpdater 类似，也是利用 CAS 机制，并不会对数据加锁，而是通过对比数据的时间戳或者版本号，来实现乐观锁需要的版本判断。

乐观锁两种实现机制

1、CAS机制（Compare And Swap）

（1）CAS操作包括了3个操作数

1. 需要读写的内存位置(V)
2. 进行比较的预期值(A)
3. 拟写入的新值(B)

（2）操作逻辑

如果内存位置V的值等于预期的A值，则将该位置更新为新值B，否则不进行任何操作。
许多CAS的操作是自旋的：如果操作不成功，会一直重试，直到操作成功为止。

（3）问题

问题一、既然CAS包含了Compare和Swap两个操作，它又如何保证原子性呢?

CAS是由CPU支持的原子操作，其原子性是在硬件层面进行保证的。

问题二、CAS有哪些缺点？**----面试会问**

ABA问题：

假设有两个线程——线程1和线程2，两个线程按照顺序进行以下操作：

1. 线程1读取内存中数据为A
2. 线程2将该数据修改为B
3. 线程2将该数据修改为A
4. 线程1对数据进行CAS操作

在第 4 步中，由于内存中数据仍然为A，因此CAS操作成功，但实际上该数据已经被线程2修改过了。这就是**ABA问题**（就是乐观锁并不知道数据已经改变了，仅仅基于值是否变化作为判断依据）

但是在某些场景下，ABA却会带来隐患，例如栈顶问题：一个栈的栈顶经过两次(或多次)变化又恢复了原值，但是栈可能已发生了变化。

解决方法：

引入版本号，内存中的值每发生一次变化，版本号都+1；在进行CAS操作时，不仅比较内存中的值，也会比较版本号，只有当二者都没有变化时，CAS才能执行成功。
Java中的AtomicStampedReference类便是使用版本号来解决ABA问题的。

高竞争下的开销问题：

在并发冲突概率大的高竞争环境下，如果CAS一直失败，会一直重试，CPU开销较大。

解决方法：

思路一：是引入退出机制，如重试次数超过一定阈值后失败退出；

思路二：更重要的是避免在高竞争环境下使用乐观锁。

2、功能限制

CAS的功能是比较受限的，例如CAS只能保证单个变量（或者说单个内存值）操作的原子性，这意味着：

(1) 原子性不一定能保证线程安全，例如在Java中需要与volatile配合来保证线程安全；
(2) 当涉及到多个变量(内存值)时，CAS也无能为力。

除此之外，CAS的实现需要硬件层面处理器的支持，灵活性受到限制。

版本号机制

基本思路：

在数据中增加一个字段version，表示该数据的版本号，每当数据被修改，版本号加1。

当某个线程查询数据时，将该数据的版本号一起查出来；当该线程更新数据时，判断当前版本号与之前读取的版本号是否一致，如果一致才进行操作

也可以使用时间戳作为标记；

典型锁

1、互斥锁(mutex)--线程锁

一次只能一个线程拥有互斥锁，其他线程只有等待。

线程间互斥机制

属于sleep-waiting类型的锁。

互斥锁是在抢锁失败的情况下主动放弃CPU进入睡眠状态直到锁的状态改变时再唤醒，而操作系统负责线程调度，为了实现锁的状态发生改变时唤醒阻塞的线程或者进程，需要把锁交给操作系统管理，所以互斥锁在加锁操作时涉及上下文的切换。

2、条件变量(cond)--线程锁

条件变量分为两部分

1. 条件
2. 变量

条件本身是由互斥量保护的，线程在改变条件状态前先要锁住互斥量，它利用线程间共享的全局变量进行同步的一种机制。

用来等待而不是用来上锁的。条件变量用来自动阻塞一个线程，直到某特殊情况发生为止。

条件变量通过允许线程阻塞和等待另一个线程发送信号的方法弥补了互斥锁的不足，他常和互斥锁一起使用，以免出现竞态条件。

当条件不满足时，线程往往解开相应的互斥锁并阻塞线程然后等待条件发生变化。

一旦其他的某个线程改变了条件变量，他将通知相应的条件变量唤醒一个或多个正被此条件变量阻塞的线程。满足条件则被唤醒

3、自旋锁(spin)--线程锁

属于busy-waiting类型的锁

如果线程A是使用pthread_spin_lock操作去请求锁

如果自旋锁已经被线程B所持有，那么线程A就会一直在core 0上进行忙等待并不停的进行锁请求，检查该自旋锁是否已经被线程B释放，直到得到这个锁为止。

因为自旋锁不会引起调用者睡眠，所以自旋锁的效率远高于互斥锁。

缺点：

（1）自旋锁一直占用CPU

在未获得锁的情况下，一直进行自旋，所以占用着CPU，如果不能在很短的时间内获得锁，无疑会使CPU效率降低。

（2）在用自旋锁时有可能造成死锁

当递归调用时有可能造成死锁。

（3）自旋锁只有在内核可抢占式或SMP的情况下才真正需要

在单CPU且不可抢占式的内核下，自旋锁的操作为空操作。自旋锁适用于锁使用者保持锁时间比较短的情况下。

4、读写锁

1. 多个读者可以同时进行读
2. 写者必须互斥（只允许一个写者写，也不能读者写者同时进行）
3. 写者优先于读者（一旦有写者，则后续读者必须等待，唤醒时优先考虑写者）

死锁相关问题

1、死锁

两个（多个）线程相互等待对方数据的过程，死锁的产生会导致程序卡死，不解锁程序将永远无法进行下去。

在两个或多个并发进程中，如果每个进程持有某种资源而又等待着别的进程释放它或它们现在保持着的资源，在未改变这种状态之前都不能向前推进。

5.2 产生死锁的原因

系统能够提供的资源个数比要求该资源的进程数要少。

5.3 产生死锁的必要条件

5.3.1 互斥条件

涉及的资源是非共享的，即一次只有一个进程使用。【如果有另一个进程申请该资源，那么申请进程必须等待，直到该资源被释放】

5.3.2 不剥夺条件（非抢占）

进程所获得的资源在未使用完毕之前，不能被其他进程强行夺走，即只能由获得该资源的进程自己来释放。

5.3.3 占有并等待（部分分配）

进程每次申请它所需要的一部分资源。在等待一新资源的同时，进程继续占用已分配到的资源。

5.3.4 环路条件（循环等待）

存在一种进程的循环链，链中的每一个进程已获得的资源同时被链中下一个进程所请求。

5.4 死锁的解决方案

保证上锁的顺序一致。

5.5 处理方法

5.5.1 鸵鸟策略

把头埋在沙子里，假装根本没发生问题。

因为解决死锁问题的代价很高，因此鸵鸟策略这种不采取任务措施的方案会获得更高的性能。

当发生死锁时不会对用户造成多大影响，或发生死锁的概率很低，可以采用鸵鸟策略。

大多数操作系统，包括 Unix，Linux 和 Windows，处理死锁问题的办法仅仅是忽略它

5.5.2 死锁检测与死锁恢复

不试图阻止死锁，而是当检测到死锁发生时，采取措施进行恢复。

1. 每种类型一个资源的死锁检测

   每种类型一个资源的死锁检测算法是通过检测有向图是否存在环来实现，从一个节点出发进行深度优先搜索，对访问过的节点进行标记，如果访问了已经标记的节点，就表示有向图存在环，也就是检测到死锁的发生。
2. 每种类型多个资源的死锁检测

算法总结如下：

每个进程最开始时都不被标记，执行过程有可能被标记。

当算法结束时，任何没有被标记的进程都是死锁进程。

1. 寻找一个没有标记的进程 Pi，它所请求的资源小于等于 A（剩余资源）
   - 如果找到了这样一个进程，那么将 C 矩阵（每个进程已分配资源）的第 i 行向量加到 A 中，标记该进程，并转回 1
   - 如果没有这样一个进程，算法终止

5.5.4 死锁预防

在程序运行之前预防发生死锁。

1. 破坏互斥条件

例如假脱机打印机技术允许若干个进程同时输出，唯一真正请求物理打印机的进程是打印机守护进程。

2. 破坏请求和保持条件

一种实现方式是规定所有进程在开始执行前请求所需要的全部资源。

3. 破坏不剥夺条件

允许抢占资源。

4. 破坏循环请求等待

给资源统一编号，进程只能按编号顺序来请求资源

5.5.5 死锁避免

在程序运行时避免发生死锁。

安全状态（必须要求不能发生死锁）：

如果没有死锁发生，并且即使所有进程突然请求对资源的最大需求，也仍然存在某种调度次序能够使得每一个进程运行完毕，则称该状态是安全的。

计算机网络

网络拥塞

在某段时间，若对网络中某一资源（带宽、交换结点中的缓存和处理机）的需求超过了该资源所能提供的可用部分，网络性能就要变坏。

出现拥塞而不进行控制，整个网络的吞吐量将随输入负荷的增大而下降。

1、拥塞窗口（cwnd）

由发送方维护的状态变量，其值取决于网络的拥塞程序，并动态变化。

拥塞窗口的维护原则：

只要网络没有出现拥塞，窗口就再增大一些；
但是网络出现拥塞，拥塞窗口就减少一些。

2、判断出现网络拥塞的依据：没有按时收到应当到达的确认报文（即发生重传）

3、发送方将拥塞窗口作为发送窗口，即 swnd = cwnd

4、维护一个慢开始门限ssthresh状态变量

当cwnd < ssthresh时，使用慢开始算法
当cwnd > ssthresh时，停止使用慢开始算法，改用拥塞避免算法；
当cwnd = ssthresh时，即可用使用慢开始算法，也可使用拥塞避免算法；

拥塞控制的四种方法

1、慢开始（指数增长）

TCP连接刚建立，一点一点地提速，试探一下网络的承受能力，以免直接扰乱了网络通道的秩序

慢开始是指向网络中注入的报文段少，并不是拥塞窗口增长速度慢

慢启动算法步骤：

连接建好的开始先初始化拥塞窗口cwnd大小为1，表明可以传一个MSS大小的数据；
每当收到一个ACK，cwnd大小加一，呈线性上升。
每当过了一个往返延迟时间RTT(Round-Trip Time)，cwnd大小直接翻倍，乘以2，呈指数让升。
还有一个ssthresh（slow start threshold），是一个上限，当cwnd >= ssthresh时，就会进入“拥塞避免算法”

2、拥塞避免（加法增长）

拥塞窗口大小cwnd大于等于慢启动阈值ssthresh后，就进入拥塞避免算法。拥塞避免并不能完全避免拥塞，而是将拥塞窗口增长控制为按照线性规律增长，使网络比较不容易出现拥塞。

快重传和快恢复目的是改进TCP性能的，引入一下两个算法的场景：个别报文会在网络中丢失（丢包现象）

但实际并未发生网络拥塞；由于以上场景导致以下两个操作（这是早期解决丢包的方法）

（1）发送方超时重传

原理：

在发送一个数据以后就开启一个计时器，在一定时间内如果没有得到发送数据报的ACK报文，那么就重新发送数据，直到发送成功为止，并误认网络发生了拥塞。

（2）发送方错误地启动慢开始算法，并把拥塞窗口cwnd又设置为最小值1，因为降低了传输效率。

3、快重传

就是是发送方尽快进行重传，而不是等超时重传计时器超时再重传。（可使得整个网路的吞吐量提高约20%）

发送方一旦收到3个连续的重复确认（目前判断丢包的方法），就将相应的报文段立即重传，而不是等该报文段的超时重传计时器超时再重传。

即使收到了失序的报文段也要立即发出对已收到的报文段的重复确认；

要求接收方不要等待自己发送数据时才进行捎带确认，而是要立即发送确认；

快重传算法步骤：

cwnd大小缩小为当前的一半
ssthresh设置为缩小后的cwnd大小（ssthresh = cwnd）
然后进入快速恢复算法Fast Recovery

4、快恢复

（1）cwnd = cwnd + 3 MSS，加3 MSS的原因是因为收到3个重复的ACK

（2）重传DACKs指定的数据包

- 如果再收到DACKs，那么cwnd大小增加一

- 如果收到新的ACK，表明重传的包成功了，那么退出快速恢复算法。将cwnd设置为ssthresh，然后进入拥塞避免算法

总结：

慢开始和快恢复的快慢指的是拥塞窗口（cwnd）的设定值，而不是cwnd增长速率，慢开始cwnd设置为1，快恢复cwnd设置为ssthresh

面经13

> @autuor 壮

汇总：

1. union和join区别
2. 三次握手的流程
3. 三次握手的原因
4. 三次握手是否可携带数据

数据库

union

操作对象：两个或多个记录集（对应列要相同）

并集是来自多个集合的元素的组合。

合并两个或多个SELECT语句的结果集；

1. SELECT语句中使用的col长度，数据类型和列数在两个表中应相同
2. 通过消除重复项，结果与第一个表的列名一起显示为两个表中数据的组合
3. 要使用UNION，您至少应有两个SELECT语句
4. UNION的结果集列名与UNION运算符中第一个Select语句的结果集的列名相同。另一个Select语句的结果集列名将被忽略

join

操作对象：两个表或者多个表

连接是多个集合的叉积的子集；满足条件相同的列产生的结果集

根据所使用的JOIN的类型(外部，内部，笛卡尔型)，显示查询中提到的所有列的结果

计算机网络

三次握手的流程

假设 A 为客户端，B 为服务器端。

首先 B 处于 LISTEN（监听）状态，等待客户的连接请求。

A 向 B 发送连接请求报文，SYN=1，ACK=0，选择一个初始的序号：seq = x。

B 收到连接请求报文，如果同意建立连接，则向 A 发送连接确认报文，SYN=1，ACK=1，确认号为ack = x+1，同时也选择一个初始的序号 seq = y。

A 收到 B 的连接确认报文后，还要向 B 发出确认，确认号为ack = y+1，序号为 seq = x+1。

说明：

SYN = 1报文 段不携带数据，但要消耗掉一个序号
ACK报文可以携带数据，不携带数据则不消耗序号
在socket编程中，客户端执行connect()时，将触发三次握手。

三次握手的原因

1、从浪费资源角度

三次握手可以防止已经失效的连接请求报文突然又传输到服务器端导致的服务器资源浪费。

例如，客户端先发送了一个SYN，但是由于网络阻塞，该SYN数据包在某个节点长期滞留。

然后，客户端又重传SYN数据包并正确建立TCP连接，然后传输完数据后关闭该连接。

该连接释放后失效的SYN数据包才到达服务器端。

在二次握手的前提下，服务器端会认为这是客户端发起的又一次请求，然后发送SYN ，并且在服务器端创建socket套接字，一直等待客户端发送数据。

但是由于客户端并没有发起新的请求，所以会丢弃服务端的SYN 。此时服务器会一直等待客户端发送数据从而造成资源浪费。

2、可靠性角度

本质是信道不可靠，但通信的双方需要就某个问题达成一致，三次通信是保证双方相互明确对方能收发的最小值；

理论上讲不论握手多少次都不能确认一条信道是“可靠”的，但通过3次握手可以至少确认它是“可用”的，再往上加握手次数不过是提高“它是可用的”这个结论的可信程度。另外Tcp的可靠传输更多的是靠重传机制来保证的。

3、从初始序列号角度：（本质）

三次握手的本质是为了同步双方的初始序列号。

三次握手的过程即是通信双方相互告知序列号起始值，并确认对方已经收到了序列号起始值的必经步骤。如果只是两次握手，至多只有连接发起方的起始序列号能被确认，另一方选择的序列号则得不到确认，TCP的3次握手是优化的结果，其实它应该是4次握手，由于是从零开始的建立连接，因此将SYN的ACK以及被动打开的SYN合并成了一个SYN-ACK。

握手的作用：

旨在确定两个双向的初始序列号，TCP用序列号来编址传输的字节，由于是两个方向的连接，所以需要两个序列号，握手过程不传输任何字节，仅仅确定初始序列号

三次握手是否可携带数据：

其实第三次握手的时候，是可以携带数据的。但是，第一次、第二次握手不可以携带数据。

原因如下

假如第一次握手可以携带数据的话，如果有人要恶意攻击服务器，那它每次都在第一次握手中的 SYN 报文中放入大量的数据。因为攻击者根本就不理服务器的接收、发送能力是否正常，然后疯狂着重复发 SYN 报文的话，这会让服务器花费很多时间、内存空间来接收这些报文。

第一次握手不可以放数据，其中一个简单的原因就是会让服务器更加容易受到攻击了。而对于第三次的话，此时客户端已经处于 ESTABLISHED 状态。对于客户端来说，他已经建立起连接了，并且也已经知道服务器的接收、发送能力是正常的了，所以可以携带数据。

面经14-虾皮软开岗

> @author Mona

计算机网络

上来先是 url 解析过程，然后引出了下面一堆问题

1、涉及到dns的解析过程，如何进行解析的

2、http方法了解哪些

(1) 为什么要有这些不同的方法，post get 不就可以实现了吗
(2) get post 区别是什么，这个没复习到

3、TCP 和 UDP 区别，三次握手

4、套接字的组成

服务器端的端口号是知名端口，客户端的端口号是随机的，
一台主机可以有多少个端口？ 65535 ，这个值是可以修改的
为什么会有端口号的限制呢？与哪些因素有关

5、假设一个服务器支持10万个客户端同时进行tcp连接，但是同一时刻，只有二三百的客户端传输数据，属于活跃连接，如何能快速查询哪些是活跃连接呢?

用什么数据结构，相当于让你设计一个服务器端的小程序，如何实现?

这个问题不知道他是想考察什么，好像也没有固定答案，也不知道我说的他是认可还是不认可，搞不懂

2、数据库

事务隔离了解吗，什么是事务

数据库我说了没怎么看，没问很多，但是最后他说这个还是挺重要的，有时间要好好复习

3、算法

先问了刷了多少题…

太太太墨菲定律了，感觉树都快忘了，偏偏出了一道树的公共祖先，唉，最后应该是写的不大对

第二道，在上一题做变形，其实类似于链表找公共节点，写出来了

总结：

最后给的评价，我没有注意边界条件，他说如果严格的话，看我不先判断边界就写代码就直接给我挂了，看简历应该是还算满意，不过说我跟科班比还是差很远的，自己学了这么多也还挺算不错，听不出来是夸是扁，面试官竟然就一个人，他那边还一直断线，不过这样倒是减轻了我很多紧张感。

感觉应该走不到后面了，重在参与吧，原来面试真的就像是聊天一样，没问Cpp ，网络编程算是问到了套接字，其他好像也没什么，中间因为面试官说他家停电，还给我打电话沟通了一会儿，还加了微信，是不是可以问他面试状况🤫哈哈哈虽然感觉是凉了

面经15-C++

> @author 壮

汇总：

1. TCP协议如何保证可靠性
2. TCP的粘包与拆包
3. 进程与线程的区别
4. 引入协程的原因
5. C++中的关键字【volatile/const/static】
6. 重载与重写的区别

计算机网络

TCP协议如何保证可靠性

CP主要提供了检验和、序列号/确认应答、超时重传、滑动窗口、拥塞控制和 流量控制等方法实现了可靠性传输。

1、检验和

通过检验和的方式，接收端可以检测出来数据是否有差错和异常，假如有差错就会直接丢弃TCP段，重新发送。

2、序列号/确认应答

序列号的作用不仅仅是应答的作用，有了序列号能够将接收到的数据根据序列号排序，并且去掉重复序列号的数据。TCP传输的过程中，每次接收方收到数据后，都会对传输方进行确认应答。也就是发送ACK报文，这个ACK报文当中带有对应的确认序列号，告诉发送方，接收到了哪些数据，下一次的数据从哪里发。

3、滑动窗口

滑动窗口既提高了报文传输的效率，也避免了发送方发送过多的数据而导致接收方无法正常处理的异常。

4、超时重传

超时重传是指发送出去的数据包到接收到确认包之间的时间，如果超过了这个时间会被认为是丢包了，需要重传。最大超时时间是动态计算的。

5、拥塞控制

在数据传输过程中，可能由于网络状态的问题，造成网络拥堵，此时引入拥塞控制机制，在保证TCP可靠性的同时，提高性能。

6、流量控制

如果主机A 一直向主机B发送数据，不考虑主机B的接受能力，则可能导致主机B的接受缓冲区满了而无法再接受数据，从而会导致大量的数据丢包，引发重传机制。

而在重传的过程中，若主机B的接收缓冲区情况仍未好转，则会将大量的时间浪费在重传数据上，降低传送数据的效率。所以引入流量控制机制，主机B通过告诉主机A自己接收缓冲区的大小，来使主机A控制发送的数据量。流量控制与TCP协议报头中的窗口大小有关

TCP的粘包与拆包

1、解释

TCP是面向流，没有界限的一串数据。TCP底层并不了解上层业务数据的具体含义，它会根据TCP缓冲区的实际情况进行包的划分，所以在业务上认为，一个完整的包可能会被TCP拆分成多个包进行发送，也有可能把多个小的包封装成一个大的数据包发送。

2、为什么会产生粘包和拆包呢?【具体情况】

1. 要发送的数据小于 TCP 发送缓冲区的大小，TCP 将多次写入缓冲区的数据一次发送出去
2. 接收数据端的应用层没有及时读取接收缓冲区中的数据
3. 要发送的数据大于 TCP 发送缓冲区剩余空间大小
4. 待发送数据大于 MSS（最大报文长度），TCP 在传输前将进行拆包。即 TCP 报文长度-TCP头部长度 > MSS

3、解决方法

1. 发送端将每个数据包封装为固定长度
2. 在数据尾部增加特殊字符进行分割
3. 将数据分为两部分，一部分是头部，一部分是内容体；其中头部结构大小固定，且有一个字段声明内容体的大小

操作系统

进程与线程的区别

1、调度

线程是调度的基本单位（PC，状态码，通用寄存器，线程栈及栈指针）；进程是资源拥有和分配的基本单位（打开文件，堆，静态区，代码段等）。

2、并发性

一个进程内多个线程可以并发（最好和CPU核数相等）；多个进程可以并发。

3、拥有资源

线程不拥有系统资源，但一个进程的多个线程可以共享隶属进程的资源；进程是拥有资源的独立单位。

4、系统开销

线程创建销毁只需要处理PC值，状态码，通用寄存器值，线程栈及栈指针即可；进程创建和销毁需要重新分配及销毁task_struct结构。

引入协程的原因

1、协程概念

本质用户空间下的线程，拥有自己的寄存器上下文和栈。

切换情况：

先将寄存器上下文和栈保存，等切换回来的时候再进行恢复

2、原因

（1）节省 CPU

避免系统内核级的线程频繁切换，造成的 CPU 资源浪费。而协程是用户态的线程，用户可以自行控制协程的创建于销毁，极大程度避免了系统级线程上下文切换造成的资源浪费。

（2）节约内存

在 64 位的 Linux 中，一个线程需要分配 8MB 栈内存和 64MB 堆内存，系统内存的制约无法开启更多线程实现高并发。而在协程编程模式下，可以轻松有十几万协程，这是线程无法比拟的。

（3）稳定性

前面提到线程之间通过内存来共享数据，这也导致了一个问题，任何一个线程出错时，进程中的所有线程都会跟着一起崩溃。

（4）开发效率

使用协程在开发程序之中，可以很方便的将一些耗时的 IO 操作异步化，例如写文件、耗时 IO 请求等。

面经16-C++-百度提前批一面

> @author 高欧叶尼兹

1. 自我介绍，巴拉巴拉
2. 纯虚基类和虚类对比
3. 深拷贝和浅拷贝区别
4. 知道多态嘛，有几种
5. mutable和volatile是啥呀
6. vector的resize和reverse有什么不同啊
7. 如果要你实现一个map，你底层会用哪些数据结构
8. inline知道嘛，和define有啥区别
9. 多线程和多进程的比较
0. 进程间通信方式有哪些，优缺点呢
1. select和epoll区别知道嘛，它们算同步还是异步io，同步异步区别在哪里
2. malloc是怎么实现滴
3. 项目介绍一下吧，巴拉巴拉
4. 看了用了数据库啊，那里觉得数据库有几种啊，mysql和redis比优势劣势呢
5. redis除了做缓存，还能当啥呢
6. mysql隔离级别有哪些啊，各有什么问题呢
7. mysql怎么处理并发访问呢

4道编程题

8. 两个单项链表，怎么找第一个公公节点呢，如果只有这么一个公共节点，你怎么找
9. 完全二叉树怎么判断呢着一下吧，你这if else用的有点多啊，你还有更简洁的方法嘛
0. 写一个字符串转换数字的题吧，字符串不做任何限制
1. 无锁队列知道嘛，你会怎么实现呢，具体写写插入的部分吧

聊了一个半小时，聊完脑子有点懵，就记得这么多了，哎，感觉发挥不是很好，写编程题的时候，脑子有点懵了，发挥一般流泪

面经17-C++

> @author 壮

汇总：

1. 构体内存对齐的问题
2. 指针与引用的区别
3. 计算机网络的各层协议及作用?
4. Nagle 算法与延迟确认
5. 数据库事务
6. MVCC版本控制

1. 结构体内存对齐的问题

1.1 概念

访问特定类型变量的时候经常在特定的内存地址访问，这需要各种类型数据按照一定的规则在空间上排序，而不是顺序的一个接一个的排放。

1.2 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）

未对齐：

会导致访问变量时发生错误，以及读取效率上下降很多；

1.3 对齐的原则

有效对齐值N是最终用来决定数据存放地址方式的值，该数据的"存放起始地址%N=0。

数据结构中的数据变量都是按照定义的顺序来排放的，第一个数据变量的起始地址就是数据结构的起始地址。

结构体本身也要根据自身的有效对齐值取整（结构体成员变量占用总长度需要是对结构体有效对齐值的整数倍）

小总结：

各变量要对齐 + 结构体整体也要对齐。

1.4 基础知识

数据类型自身的对齐值：

对于char型数据，其自身对齐值为1，对于short型为2，对于int,float,double类型，其自身对齐值为4，单位字节。

结构体或者类的自身对齐值：

其成员中自身对齐值最大的那个值。

指定对齐：

#pragma pack (value)时的指定对齐值value。

数据成员、结构体和类的有效对齐：

自身对齐值和指定对齐值中小的那个值；

在结构体和类中成员函数中的非虚函数不占空间，虚函数占一个指针的字节；

静态成员变量所有的类对象共享一份，在静态区域中，并不占用类对象的空间；

没有任何成员变量的类对象占用一个字节的空间；

2. 指针与引用的区别

1. 指针是一个变量，存储的是一个地址，引用跟原来的变量实质上是同一个东西，是原变量的别名；
2. 指针可以有多级，引用只有一级。
3. 指针可以为空，引用不能为NULL且在定义时必须初始化；
4. 指针在初始化后可以改变指向，而引用在初始化之后不可再改变；
5. sizeof指针得到的是本指针的大小，sizeof引用得到的是引用所指向变量的大小；
6. 当把指针作为参数进行传递时，也是将实参的一个拷贝传递给形参，两者指向的地址相同，但不是同一个变量，在函数中改变这个变量的指向不影响实参，而引用却可以。
7. 引用本质是一个指针，同样会占4字节内存；指针是具体变量，需要占用存储空间（具体情况还要具体分析）。
8. 引用在声明时必须初始化为另一变量，一旦出现必须为typename refname &varname形式；指针声明和定义可以分开，可以先只声明指针变量而不初始化，等用到时再指向具体变量。
9. 引用一旦初始化之后就不可以再改变（变量可以被引用为多次，但引用只能作为一个变量引用）；指针变量可以重新指向别的变量。
0. 不存在指向空值的引用，必须有具体实体；但是存在指向空值的指针。

计算机网络

1. 计算机网络的各层协议及作用

1.1 分类

OSI七层模型：

大而全，但是比较复杂、而且是先有了理论模型，没有实际应用。

TCP/IP四层模型：

是由实际应用发展总结出来的，从实质上讲，TCP/IP只有最上面三层，最下面一层没有什么具体内容，TCP/IP参考模型没有真正描述这一层的实现。（网络接口层、网际层IP、运输层、应用层）

五层模型：

五层模型只出现在计算机网络教学过程中，这是对七层模型和四层模型的一个折中，既简洁又能将概念阐述清楚。（物理层、数据链路层、网络层、运输层、应用层）

1.2 七层协议含义以及作用

1、应用层

为应用程序提供交互服务。在互联网中的应用层协议很多，如域名系统DNS，支持万维网应用的HTTP协议，支持电子邮件的SMTP协议等。

2、表示层

主要负责数据格式的转换，如加密解密、转换翻译、压缩解压缩等。

3、会话层

负责在网络中的两节点之间建立、维持和终止通信，如服务器验证用户登录便是由会话层完成的。

4、运输层

有时也译为传输层，向主机进程提供通用的数据传输服务。该层主要有以下两种协议：

（1）TCP

提供面向连接的、可靠的数据传输服务；

（2）UDP

提供无连接的、尽最大努力的数据传输服务，但不保证数据传输的可靠性。

5、网络层

选择合适的路由和交换结点，确保数据及时传送。主要包括IP协议。

6、数据链路层

数据链路层通常简称为链路层。将网络层传下来的IP数据包组装成帧，并再相邻节点的链路上传送帧。

7、物理层

实现相邻节点间比特流的透明传输，尽可能屏蔽传输介质和通信手段的差异。

2. Nagle 算法与延迟确认

Nagle 算法

2.1 引入该算法的场景

TCP/IP协议中，无论发送多少数据，总是要在数据前面加上协议头，同时，对方接收到数据，也需要发送ACK表示确认。为了尽可能的利用网络带宽，TCP总是希望尽可能的发送足够大的数据。Nagle算法就是为了尽可能发送大块数据，避免网络中充斥着许多小数据块。

2.2 定义

任意时刻，最多只能有一个未被确认的小段。所谓“小段”，指的是小于MSS尺寸的数据块，所谓“未被确认”，是指一个数据块发送出去后，没有收到对方发送的ACK，确认该数据已收到

2.3 算法实现规则

如果包长度达到MSS，则允许发送；

如果该包含有FIN，则允许发送；

设置了TCP_NODELAY选项，则允许发送；

未设置TCP_CORK选项时，若所有发出去的小数据包（包长度小于MSS）均被确认，则允许发送；

上述条件都未满足，但发生了超时（一般为200ms），则立即发送；

2.4 延迟确认

定义：

接收方收到数据包后，如果暂时没有数据要发给对端，它可以等一段时再确认（Linux上默认是40ms）。如果这段时间刚好有数据要传给对端，ACK就随着数据传输，而不需要单独发送一次ACK。如果超过时间还没有数据要发送，也发送ACK，避免对端以为丢包。

不能延迟的场景：

乱序包、接收到了大于一个 窗口大小的报文，且需要调整窗口大小。

总结：

一般情况下，Nagle算法和延迟确认不能一起使用，Nagle算法意味着延迟发，延迟确认意味着延迟接收，就会造成更大的延迟，会产生性能问题。

数据库

1. 数据事务的四大特性（ACID）

原子性(atomicity)：

事务的最小工作单元，要么全成功，要么全失败。

一致性(consistency)：

事务开始和结束后，数据库的完整性不会被破坏。

隔离性(isolation)：

不同事务之间互不影响，四种隔离级别为RU（读未提交）、RC（读已提交）、RR（可重复读）、SERIALIZABLE（串行化）。

持久性(durability)：

事务提交后，对数据的修改是永久性的，即使系统故障也不会丢失。

2. MVCC-多版本并发控制

1、问题：

多个事务同时操作可能会产生的问题，会出现某个事务的操作被覆盖而导致数据丢失。

2、解决方式：

（1）LBCC

基于锁的并发控制（使用锁的机制，在当前事务需要对数据修改时，将当前事务加上锁，同一个时间只允许一条事务修改当前数据，其他事务必须等待锁释放之后才可以操作）。

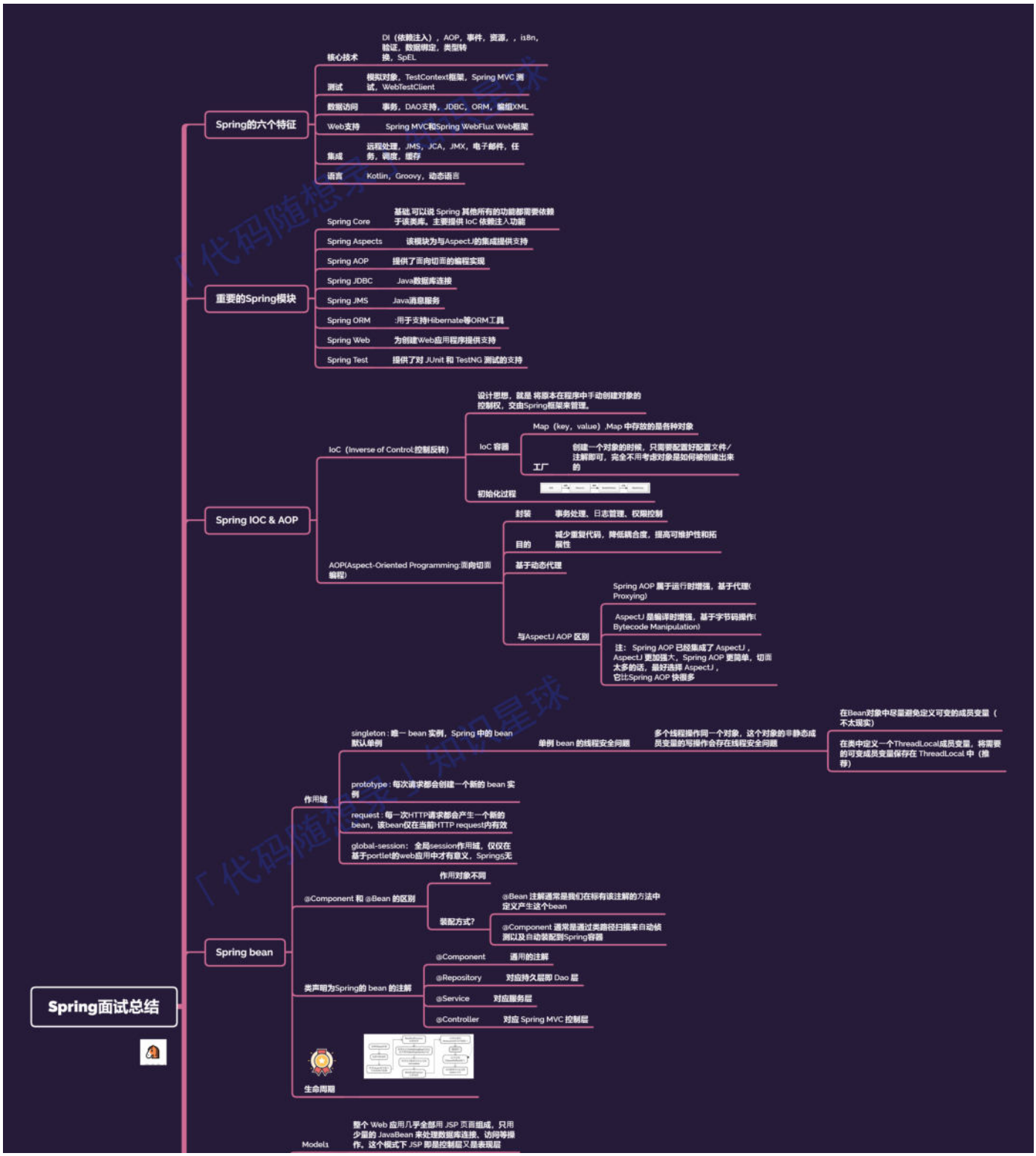
（2）MVCC

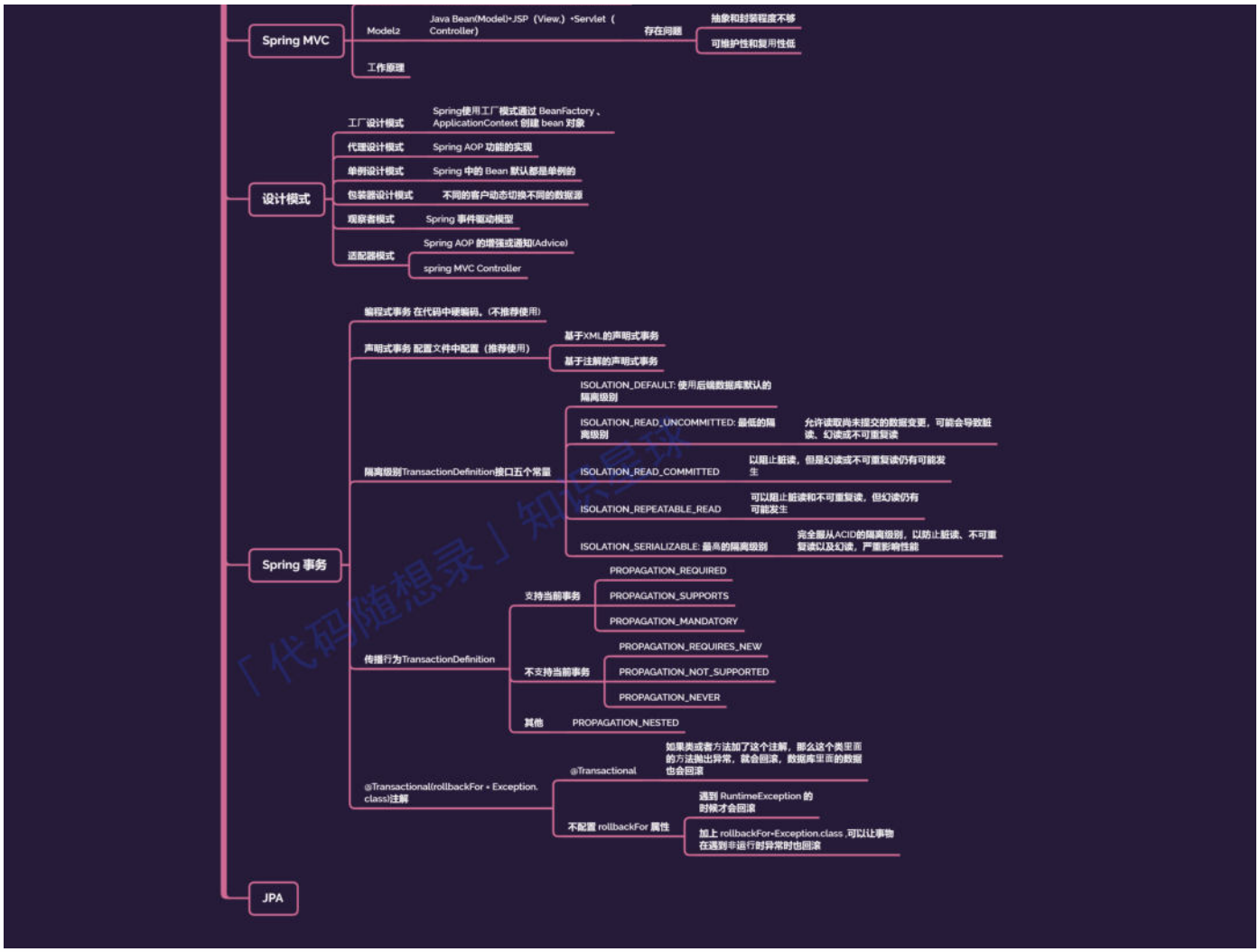
多版本的并发控制：

1. MVCC 使得数据库读不会对数据加锁，普通的 SELECT 请求不会加锁，提高了数据库的并发处理能力
2. MVCC 数据库可以实现提交读，可重复读的隔离级别，用户可以查看当前数据的前一个或者前几个历史版本，保证了ACID中的隔离性

面经18-Spring

> @author 🐴





面经19-C++

> @author 壮

汇总：

1. 深拷贝与浅拷贝
2. inline 与 deifine 的区别
3. 半连接队列和全连接队列
4. SYN（洪范）攻击

C++

1. 深拷贝与浅拷贝

浅拷贝只是拷贝一个指针，并没有新开辟一个地址，拷贝的指针和原来的指针指向同一块地址，如果原来的指针所指向的资源释放了，那么再释放浅拷贝的指针的资源就会出现错误。

深拷贝不仅拷贝值，还开辟出一块新的空间用来存放新的值，即使原先的对象被析构掉，释放内存了也

不会影响到深拷贝得到的值。在自己实现拷贝赋值的时候，如果有指针变量的话是需要自己实现深拷贝的。

2. inline与deifine的区别

主要区别：

1. 宏在预编译时进行，只做简单字符串替换
2. 内联函数在编译时直接将函数代码嵌入到目标代码中，省去函数调用的开销来提高执行效率，并且进行参数类型检查，具有返回值，可以实现重载

内联函数适用场景：

1. 使用宏定义的地方都可以使用inline函数
2. 作为类成员接口函数来读写类的私有成员或者保护成员，会提高效率

为什么不能把所有的函数写成内联函数：

1. 函数体内的代码比较长，将导致内存消耗大
2. 函数体内有循环，函数执行时间要比函数调用开销大

计算机网络

1. 半连接队列和全连接队列

TCP进入三次握手前，服务端会从CLOSED状态变为LISTEN状态,同时在内部创建了两个队列：

1. 半连接队列（SYN队列）
2. 全连接队列（ACCEPT队列）

半连接队列（SYN队列）：

TCP三次握手时，客户端发送SYN到服务端，服务端收到之后，便回复ACK和SYN，状态由LISTEN变为SYN_RCVD，此时这个连接就被推入了SYN队列，即半连接队列。

全连接队列：

当客户端回复ACK, 服务端接收后，三次握手就完成了。这时连接会等待被具体的应用取走，在被取走之前，它被推入ACCEPT队列，即全连接队列。

2. SYN（洪范）攻击

1、概念

典型的DoS (Denial of Service，拒绝服务) 攻击，它在短时间内，伪造不存在的IP地址,向服务器大量发起SYN报文。当服务器回复SYN+ACK报文后，不会收到ACK回应报文，导致服务器上建立大量的半连接，致使半连接队列满了，这就无法处理正常的TCP请求。

它利用 TCP 协议缺陷，通过发送大量的半连接请求，耗费 CPU 和内存资源。

2、检测

- 当在服务器上看到大量的半连接状态时，特别是源 IP 地址是随机的，基本上可以断定这是一次 SYN 攻击。

3、解决方法

（1）通过防火墙

服务器防火墙会对收到的每一个SYN报文进行代理和回应，并保持半连接。等发送方将ACK包返回后，再重新构造SYN包发到服务器，建立真正的TCP连接、路由器等过滤网关防护。

（2）通过加固 TCP/IP 协议栈防范

如增加最大半连接数，缩短超时时间。

（3）SYN cookies技术

SYN Cookies 是对 TCP 服务器端的三次握手做一些修改，专门用来防范 SYN 洪泛攻击的一种手段。

【在收到SYN包后，服务器根据一定的方法，以数据包的源地址、端口等信息为参数计算出一个cookie值作为自己的SYNACK包的序列号，回复SYN+ACK后，服务器并不立即分配资源进行处理，等收到发送方的ACK包后，重新根据数据包的源地址、端口计算该包中的确认序列号是否正确，如果正确则建立连接，否则丢弃该包。】

面经20

> @author 壮

汇总：

1. 三次握手连接阶段，最后一次ACK包丢失，会发生什么?
2. 四次握手流程
3. 为什么需要四次握手?
4. 重做日志与二进制日志的比较
5. 三种日志（重做 / 二进制 / 回滚）

计算机网络

1、三次握手连接阶段，最后一次ACK包丢失，会发生什么?

服务端:

第三次的ACK在网络中丢失，那么服务端该TCP连接的状态为SYN_RECV,并且会根据 TCP的超时重传机制，会等待3秒、6秒、12秒后重新发送SYN+ACK包，以便客户端重新发送ACK包。
如果重发指定次数之后，仍然未收到 客户端的ACK应答，那么一段时间后，服务端自动关闭这个连接。

客户端:

客户端认为这个连接已经建立，如果客户端向服务端发送数据，服务端将以RST包（Reset，标示复位，用于异常的关闭连接）响应。此时，客户端知道第三次握手失败。

2、四次握手流程

设定 A为客户端，B为服务端， ACK 在连接建立之后都为 1。

A 发送连接释放报文，FIN=1。【停止发送数据，主动关闭TCP连接】
B 收到之后发出确认，此时 TCP 属于半关闭状态，B 能向 A 发送数据但是 A 不能向 B 发送数据。
当 B 不再需要连接时，发送连接释放报文，FIN=1。
A 收到后发出确认，进入 TIME-WAIT 状态，等待 2 MSL（最大报文存活时间）后释放连接。
B 收到 A 的确认后释放连接。

3、什么需要四次握手?

服务器在收到客户端的 FIN 报文段后，可能还有一些数据要传输，所以不能马上关闭连接，但是会做出应答，返回 ACK 报文段.

接下来可能会继续发送数据，在数据发送完后，服务器会向客户单发送 FIN 报文，表示数据已经发送完毕，请求关闭连接。服务器的ACK和FIN一般都会分开发送，从而导致多了一次，因此一共需要四次挥手。

任何一方都可以在数据传送结束后发出连接释放的通知，待对方确认后进入半关闭状态。当另一方也没有数据再发送的时候，则发出连接释放通知，对方确认后就完全关闭了TCP连接。

数据库

日志类别	逻辑层架构	文件空间	作用	写入方式	内容	恢复数据的效率
redo log	InnoDB引擎特有	空间大小固定	保证事务的持久性的，是事务层面的	循环写入和擦除	物理日志；是数据页面的修改之后的物理记录。	高
binlog	通过MySQL的Sever层实现，所有引擎都可以使用	空间不固定，写完会切换下一个文件	还原的功能，是数据库层面的	追加写入，不会覆盖已写文件	逻辑日志；记录的就是sql语句。	低
两者日志产生的时间，可以释放的时间，在可释放的情况下清理机制，都是完全不同的。						

1、重做日志（redo log）--崩溃恢复

作用：

确保事务的持久性。防止在发生故障的时间点，尚有脏页未写入磁盘，在重启mysql服务的时候，根据redo log进行重做，从而达到事务的持久性这一特性。

内容：

物理格式的日志，记录的是物理数据页面的修改的信息，其重做日志是顺序写入重做日志文件的物理文件中去的。

产生时机：

事务开始之后就产生重做日志，重做日志的落盘并不是随着事务的提交才写入的，而是在事务的执行过程中，便开始写入重做日志文件中。

释放时间：

当对应事务的脏页/脏数据【是指在内存中未刷到磁盘的数据】写入到磁盘之后，重做日志的使命也就完成了，重做日志占用的空间就可以重用（被覆盖）

更新步骤：

重做日志先将数据写入缓冲区；然后将缓冲区中的数据写入重做日志文件。

脏数据刷盘：

重做日志的大小是固定的，设置了两个标志位置，checkpoint和write_pos，分别表示记录擦除的位置和记录写入的位置，标识之间的为空闲可用空间，写入和擦除都是往后推移，循环往复的。当write_pos = checkpoint时，表示redo log日志已经写满，执行checkpoint规则【checkpoint触发后，将buffer中脏数据页和脏日志页都刷到磁盘】腾出可写空间

脏日志刷盘：

重做日志两部分

1. 存在易失性内存中的缓存日志
2. 保存在磁盘上的重做日志文件

为了确保每次记录都能够写入到磁盘中的日志中，每次将缓存区中的日志写入重做日志文件的过程中都会调用一次操作系统的fsync【同步内存中所有已修改的文件数据到储存设备】操作。

2、二进制日志（binlog）--服务层日志

作用：

主从库的同步；用于数据库的基于时间点的还原；

内容：

逻辑格式的日志，主要记录数据库的变化情况，内容包括数据库所有的更新操作，执行过的事务中的sql语句；

产生时机：

事务提交的时候，一次性将事务中的sql语句（一个事物可能对应多个sql语句）按照一定的格式记录到binlog中。

3、回滚日志

作用：

保存了事务发生之前的数据的一个版本，可以用于回滚，同时可以提供多版本并发控制（（MVCC））下的读，也即非锁定读；

内容：

逻辑格式的日志，在执行undo的时候，仅仅是将数据从逻辑上恢复至事务之前的状态，而不是从物理页面上操作实现的，这一点是不同于重做日志的。

产生时机：

事务开始之前，将当前版本生成回滚日志，undo 也会产生 redo 来保证回滚日志的可靠性;

释放时机：

当事务提交之后，回滚日志并不能立马被删除，而是放入待清理的链表，由purge线程判断是否由其他事务在使用undo段中表的上一个事务之前的版本信息，决定是否可以清理回滚日志的日志空间

面经21

> @author 壮

计算机网络

APR

1、需要ARP解析的原因

（1）MAC地址是数据链路层和物理层使用的地址，而IP地址是网络层和以上各层使用的地址。

（2）MAC帧在传送时使用的源地址和目的地址都是硬件地址。

连接在通信链路上的设备（主机或路由器）在接收MAC帧时，根据是MAC帧首部的硬件地址

（3）数据链路层的以太网协议接到上层IP协议提供的数据中，只包含目的主机的IP地址

需要根据一种方法，根据目的主机的IP地址，获得其MAC地址，完成不同主机的通信。

（4）ARP地址解析协议

完成IP地址到MAC地址的映射【该映射关系存放到主机中的ARP缓存表】

2、主机A与主机B数据链路层通信的过程

（1）主机A中ARP进程在本局域网上广播发送一个ARP请求分组

（2）本局域网上所有的主机上运行的ARP进程都收到此ARP请求分组

（3）主机B在ARP分组中见到自己的IP地址就向A发送ARP响应分组，并写入自己的硬件地址，相应分组是普通的单播

（4）主机A收到主机B的ARP响应分组后，就在其ARP高速缓存中写入主机B的IP地址到硬件地址的映射

另外，当发送主机和目的主机不在同一个局域网中时，即便知道目的主机的MAC地址，两者也不能直接通信，必须经过路由转发才可以。【委托ARP或ARP代理】：发送主机此时通过ARP协议一台可以通往局域网外的路由器的MAC地址，发送主机发往目的主机的所有帧，都将发往该路由器，通过它向外发送

小总结：

一般在局域网中ARP请求分组时才能拿到MAC地址，否则获得是连接外网的路由器地址，通过路由器进行转发。

面经22

> @author 壮

ICMP【网络层协议】详解

背景：

IP协议并不提供可靠传输。如果丢包了，IP协议并不能通知传输层是否丢包以及丢包的原因，所以引入了ICMP协议。

作用：

1. 确认IP包是否成功到达目标地址
2. 通知在发送过程中IP包被丢弃的原因

类别：

1. 一类是通知出错原因
2. 一类是用于诊断查询

应用：

ping命令：

功能：

1. 能验证网络的连通性
2. 会统计响应时间和TTL(IP包中的Time To Live，生存周期)

过程：

1. ping命令会先发送一个 ICMP Echo Request给对端
2. 对端接收到之后, 会返回一个ICMP Echo Reply
3. 若没有返回，就是超时了，会认为指定的网络地址不存在

traceroute：

功能：打印出可执行程序主机，一直到目标主机之前经历多少路由器。

3、面试

晚上一面商汤感知算法；

1、项目

2、线程与进程的区别

3、线程通信与进程通信

4、虚函数

5、实现memcpy函数，直接进行顺序拷贝的；最后面试官想知道的是关于正向拷贝与逆向拷贝，这个没答好；

面经23-Java-七牛云一面

> @author 🐴

1、自我介绍 介绍项目

自我介绍需要重新好好准备一下，项目要重新做一个简单的

Java

JAVA了解什么，容器相关的，HashMap了解吗，如何扩容的，put操作的底层实现

HashMap（JDK1.7为例）

存储结构

内部包含了一个 Entry 类型的数组 table。Entry 存储着键值对。它包含了四个字段，从 next 字段我们可以看出 Entry 是一个链表。即数组中的每个位置被当成一个桶，一个桶存放一个链表。HashMap 使用拉链法来解决冲突，同一个链表中存放哈希值和散列桶取模运算结果相同的 Entry

拉链法的工作原理

```
HashMap<String, String> map = new HashMap<>();
map.put("K1", "V1");
map.put("K2", "V2");
map.put("K3", "V3");
```

新建一个 HashMap，默认大小为 16；
插入 <K1,V1> 键值对，先计算 K1 的 hashCode 为 115，使用除留余数法得到所在的桶下标 115%16=3。
插入 <K2,V2> 键值对，先计算 K2 的 hashCode 为 118，使用除留余数法得到所在的桶下标 118%16=6。
插入 <K3,V3> 键值对，先计算 K3 的 hashCode 为 118，使用除留余数法得到所在的桶下标 118%16=6，插在 <K2,V2> 前面。
应该注意到链表的插入是以头插法方式进行的，例如上面的 <K3,V3> 不是插在 <K2,V2> 后面，而是插入在链表头部。

查找需要分成两步进行：

1. 计算键值对所在的桶
2. 在链表上顺序查找，时间复杂度显然和链表的长度成正比

put操作

```
public V put(K key, V value) {
    if (table == EMPTY_TABLE) {
        inflateTable(threshold);
    }
  // 键为 null 单独处理
    if (key == null)
        return putForNullKey(value);
    int hash = hash(key);
    // 确定桶下标
    int i = indexFor(hash, table.length);
    // 先找出是否已经存在键为 key 的键值对，如果存在的话就更新这个键值对的值为 value
    for (Entry<K,V> e = table[i]; e != null; e = e.next) {
        Object k;
        if (e.hash == hash && ((k = e.key) == key || key.equals(k))) {
            V oldValue = e.value;
            e.value = value;
            e.recordAccess(this);
```

```
                return oldValue;
            }
        }
        modCount++;
        // 插入新键值对
        addEntry(hash, key, value, i);
        return null;
    }
```

HashMap 允许插入键为 null 的键值对。但是因为无法调用 null 的 hashCode() 方法，也就无法确定该键值对的桶下标，只能通过强制指定一个桶下标来存放。HashMap 使用第 0 个桶存放键为 null 的键值对。

使用链表的头插法，也就是新的键值对插在链表的头部，而不是链表的尾部。

确定桶下标

很多操作都需要先确定一个键值对所在的桶下标。

```
int hash = hash(key);
int i = indexFor(hash, table.length);
```

计算哈希值

```
final int hash(Object k) {
    int h = hashSeed;
    if (0 != h && k instanceof String) {
        return sun.misc.Hashing.stringHash32((String) k);
    }
    h ^= k.hashCode();
    // This function ensures that hashCodes that differ only by
    // constant multiples at each bit position have a bounded
    // number of collisions (approximately 8 at default load factor).
    h ^= (h >>> 20) ^ (h >>> 12);
    return h ^ (h >>> 7) ^ (h >>> 4);
}
public final int hashCode() {
    return Objects.hashCode(key) ^ Objects.hashCode(value);
}
```

取模

令 x = 1 << 4，即 x 为 2 的 4 次方，它具有以下性质：

```
x : 00010000
x-1 : 00001111
```

令一个数 y 与 x-1 做与运算，可以去除 y 位级表示的第 4 位以上数：

```
y   : 10110010
x-1  : 00001111
y&(x-1) : 00000010
```

这个性质和 y 对 x 取模效果是一样的：

```
y : 10110010
x : 00010000
y % x : 00000010
```

我们知道，位运算的代价比求模运算小的多，因此在进行这种计算时用位运算的话能带来更高的性能。

确定桶下标的最后一步是将 key 的 hash 值对桶个数取模：hash%capacity，如果能保证 capacity 为 2 的 n 次方，那么就可以将这个操作转换为位运算。

```
static int indexFor(int h, int length) {
    return h & (length-1);
}
```

5. 扩容 - 基本原理

设 HashMap 的 table 长度为 M，需要存储的键值对数量为 N，如果哈希函数满足均匀性的要求，那么每条链表的长度大约为 N/M，因此查找的复杂度为 O(N/M)。

为了让查找的成本降低，应该使 N/M 尽可能小，因此需要保证 M 尽可能大，也就是说 table 要尽可能大。HashMap 采用动态扩容来根据当前的 N 值来调整 M 值，使得空间效率和时间效率都能得到保证。

参数含义 capacitytable 的容量大小，默认为 16。

需要注意的是 capacity 必须保证为 2 的 n 次方。size键值对数量。thresholdsize 的临界值，当 size 大于等于 threshold 就必须进行扩容操作。

loadFactor 装载因子，table 能够使用的比例，threshold = (int)(capacity* loadFactor)。从下面的添加元素代码中可以看出，当需要扩容时，令 capacity 为原来的两倍。

```
void addEntry(int hash, K key, V value, int bucketIndex) {
    Entry<K,V> e = table[bucketIndex];
    table[bucketIndex] = new Entry<>(hash, key, value, e);
    if (size++ >= threshold)
    resize(2 * table.length);
}
```

扩容使用 resize() 实现，需要注意的是，扩容操作同样需要把 oldTable 的所有键值对重新插入 newTable 中，因此这一步是很费时的。

6. 扩容-重新计算桶下标

在进行扩容时，需要把键值对重新计算桶下标，从而放到对应的桶上。在前面提到，HashMap 使用 hash%capacity 来确定桶下标。

HashMap capacity 为 2 的 n 次方这一特点能够极大降低重新计算桶下标操作的复杂度。

假设原数组长度 capacity 为 16，扩容之后 new capacity 为 32：

capacity : 00010000
new capacity : 00100000

对于一个 Key，它的哈希值 hash 在第 5 位：

1. 为 0，那么 hash%00010000 = hash%00100000，桶位置和原来一致；
2. 为 1，hash%00010000 = hash%00100000 + 16，桶位置是原位置 + 16。
3. 计算数组容量

HashMap 构造函数允许用户传入的容量不是 2 的 n 次方，因为它可以自动地将传入的容量转换为 2 的 n 次方。

先考虑如何求一个数的掩码，对于 10010000，它的掩码为 11111111，可以使用以下方法得到：

```
mask |= mask >> 1 11011000

mask |= mask >> 2 11111110

mask |= mask >> 4 11111111
```

mask+1 是大于原始数字的最小的 2 的 n 次方。

num 10010000

mask+1 100000000

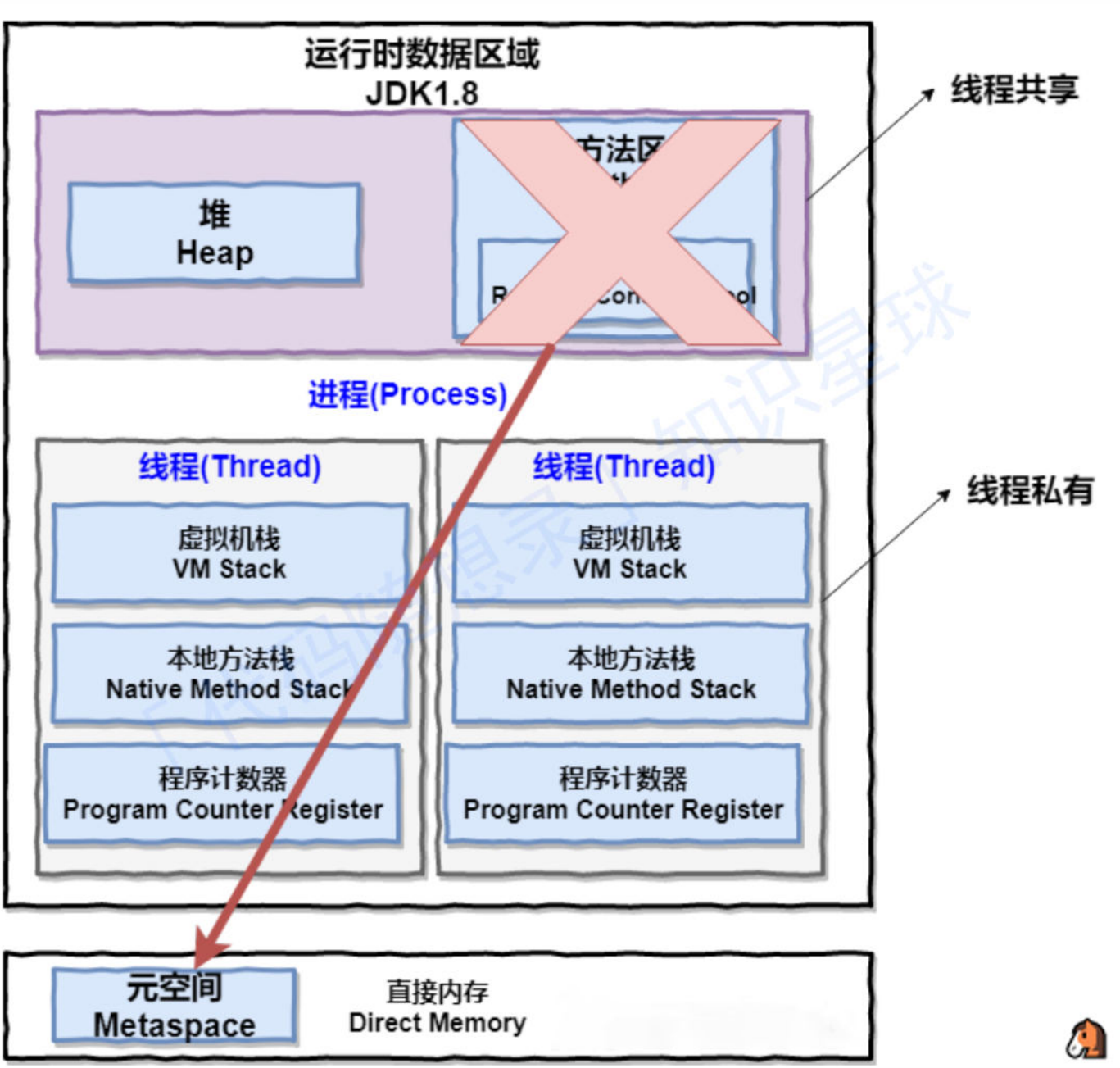
以下是 HashMap 中计算数组容量的代码：

```
tatic final int tableSizeFor(int cap) {
    int n = cap - 1;
    n |= n >>> 1;
    n |= n >>> 2;
    n |= n >>> 4;
    n |= n >>> 8;
    n |= n >>> 16;
    return (n < 0) ? 1 : (n >= MAXIMUM_CAPACITY) ? MAXIMUM_CAPACITY : n + 1;
}
```

8. 链表转红黑树

从 JDK 1.8 开始，一个桶存储的链表长度大于等于 8 时会将链表转换为红黑树。

线程和进程的区别



从上图可以看出：

一个进程中可以有多个线程，多个线程共享进程的堆和方法区 (JDK1.8 之后的元空间)资源，但是每个线程有自己的程序计数器、虚拟机栈和本地方法栈。

总结：

线程是进程划分成的更小的运行单位,一个进程在其执行的过程中可以产生多个线程。线程和进程最大的不同在于基本上各进程是独立的，而各线程则不一定，因为同一进程中的线程极有可能会相互影响。线程执行开销小，但不利于资源的管理和保护；而进程正相反。

子进程和线程的区别

子进程有自己的资源，比线程独立于父进程，可以在父进程退出后执行，如LINUX的守护进程就是这样的，线程使用的是父进程的资源，在父进程退出后跟着退出

协程

协程运行在线程之上，当一个协程执行完成后，可以选择主动让出，让另一个协程运行在当前线程之上。协程并没有增加线程数量，只是在线程的基础之上通过分时复用的方式运行多个协程，而且协程的切换在用户态完成，切换的代价比线程从用户态到内核态的代价小很多

Java并发、锁

Java 提供了两种锁机制来控制多个线程对共享资源的互斥访问，第一个是 JVM 实现的 synchronized，而另一个是 JDK 实现的 ReentrantLock。

ReentrantLock 可中断，而 synchronized 不行

	实现	等待可中断	公平锁	锁绑定多个条件	优先级
synchronized	JVM	不能	非公平	-	首选（JVM原生支持，不用担心锁释放导致的死锁，JVM确保
ReentrantLock	JDK	可以	默认非公平（可公平）	同时绑定多个 Condition 对象	

CAS

CAS操作了解吗，怎么实现的，为什么要用，既然你说是硬件层次的，软件层次的如何实现你简单讲一下，用自旋锁实现CAS简单说一下

1、乐观锁需要操作和冲突检测这两个步骤具备原子性

这里就不能再使用互斥同步来保证了，只能靠硬件来完成。硬件支持的原子性操作最典型的是：比较并交换（Compare-and-Swap，CAS）。CAS 指令需要有 3 个操作数，分别是内存地址 V、旧的预期值 A 和新值 B。当执行操作时，只有当 V 的值等于 A，才将 V 的值更新为 B。

2、CAS实现自旋锁

代码块只能这种黑色的有人会改吗

```
package sjq.mylock;

import java.util.concurrent.atomic.AtomicReference;

public class SpinLock {
    private AtomicReference<Thread> owner = new AtomicReference<>();

    public void lock(){
        Thread currentThread = Thread.currentThread();
        while(!owner.compareAndSet(null, currentThread)){  // owner == null , 则
compareAndSet返回true，否则为false。
```

```
                //拿不到owner的线程，不断的在死循环
            }
        }

        public void unLock(){
            owner.set(null);
            // 也可以这样写，太麻烦，没必要
            /*
            Thread cur = Thread.currentThread();
            owner.compareAndSet(cur, null);
             */
        }

    }
```

MySql的InnoDB引擎

InnoDB 是 MySQL 默认的事务型存储引擎，只有在需要它不支持的特性时，才考虑使用其它存储引擎。

实现了四个标准的隔离级别，默认级别是可重复读（REPEATABLE READ）。在可重复读隔离级别下，通过多版本并发控制（MVCC）+ Next-Key Locking 防止幻影读。

主索引是聚簇索引，在索引中保存了数据，从而避免直接读取磁盘，因此对查询性能有很大的提升。

内部做了很多优化，包括从磁盘读取数据时采用的可预测性读、能够加快读操作并且自动创建的自适应哈希索引、能够加速插入操作的插入缓冲区等。

支持真正的在线热备份。其它存储引擎不支持在线热备份，要获取一致性视图需要停止对所有表的写入，而在读写混合场景中，停止写入可能也意味着停止读取。

Redis

Redis 是速度非常快的非关系型（NoSQL）内存键值数据库，可以存储键和五种不同类型的值之间的映射。

键的类型只能为字符串，值支持五种数据类型：字符串、列表、集合、散列表、有序集合。

Redis 支持很多特性，例如将内存中的数据持久化到硬盘中，使用复制来扩展读性能，使用分片来扩展写性能。

MySql的索引

MySql的索引是什么结构，如何实现的，为什么用B+树，还有用别的吗

1、B+ Tree索引

是大多数 MySQL 存储引擎的默认索引类型。
因为不再需要进行全表扫描，只需要对树进行搜索即可，所以查找速度快很多。
因 B+ Tree 的有序性，所以除了用于查找，还可以用于排序和分组。
可以指定多个列作为索引列，多个索引列共同组成键。
适用于全键值、键值范围和键前缀查找，其中键前缀查找只适用于最左前缀查找。如果不是按照索引列的顺序进行查找，则无法使用索引。

InnoDB 的 B+Tree 索引分为主索引和辅助索引。主索引的叶子节点 data 域记录着完整的数据记录，这种索引方式被称为聚簇索引。因为无法把数据行存放在两个不同的地方，所以一个表只能有一个聚簇索引

2、哈希索引

哈希索引能以 O(1) 时间进行查找，但是失去了有序性：
无法用于排序与分组；
只支持精确查找，无法用于部分查找和范围查找。

InnoDB 存储引擎有一个特殊的功能叫“自适应哈希索引”，当某个索引值被使用的非常频繁时，会在 B+Tree 索引之上再创建一个哈希索引，这样就让 B+Tree 索引具有哈希索引的一些优点，比如快速的哈希查找。

3、全文索引

MyISAM 存储引擎支持全文索引，用于查找文本中的关键词，而不是直接比较是否相等。
查找条件使用 MATCH AGAINST，而不是普通的 WHERE。
全文索引使用倒排索引实现，它记录着关键词到其所在文档的映射。
InnoDB 存储引擎在 MySQL 5.6.4 版本中也开始支持全文索引

4、空间数据索引

MyISAM 存储引擎支持空间数据索引（R-Tree），可以用于地理数据存储。空间数据索引会从所有维度来索引数据，可以有效地使用任意维度来进行组合查询。
必须使用 GIS 相关的函数来维护数据。

哈希索引

你说还有哈希索引，那么什么时候用哈希什么时候用B+树呢?

B+树适用于普遍情况，可以查找，排序，分组，适用于全键值，键值范围，键前缀查找

哈希索引适用于精确查找

B树和B+树的区别

B树和B+树的区别，为什么MySql用B+树而不是B树

为什么MySQL选择B+树做索引

1、B+树的磁盘读写代价更低

B+树的内部节点并没有指向关键字具体信息的指针，因此其内部节点相对B树更小，如果把所有同一内部节点的关键字存放在同一盘块中，那么盘块所能容纳的关键字数量也越多，一次性读入内存的需要查找的关键字也就越多，相对IO读写次数就降低了。

2、B+树的查询效率更加稳定

由于非终结点并不是最终指向文件内容的结点，而只是叶子结点中关键字的索引。所以任何关键字的查找必须走一条从根结点到叶子结点的路。所有关键字查询的路径长度相同，导致每一个数据的查询效率相当。

3、B+树更便于遍历

由于B+树的数据都存储在叶子结点中，分支结点均为索引，方便扫库，只需要扫一遍叶子结点即可，但是B树因为其分支结点同样存储着数据，我们要找到具体的数据，需要进行一次中序遍历按序来扫，所以B+树更加适合在区间查询的情况，所以通常B+树用于数据库索引。

4、B+树更适合基于范围的查询

B树在提高了IO性能的同时并没有解决元素遍历的我效率低下的问题，正是为了解决这个问题，B+树应用而生。B+树只需要去遍历叶子节点就可以实现整棵树的遍历。而且在数据库中基于范围的查询是非常频繁的，而B树不支持这样的操作或者说效率太低。

计算机网络

HTTP报文格式

请求报文结构：

第一行是包含了请求方法、URL、协议版本；
接下来的多行都是请求首部 Header，每个首部都有一个首部名称，以及对应的值。
一个空行用来分隔首部和内容主体 Body
最后是请求的内容主体

```
GET http://www.example.com/ HTTP/1.1
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8,
application/signed-exchange;v=b3;q=0.9
Accept-Encoding: gzip, deflate
Accept-Language: zh-CN,zh;q=0.9,en;q=0.8
Cache-Control: max-age=0
Host: www.example.com
If-Modified-Since: Thu, 17 Oct 2019 07:18:26 GMT
If-None-Match: "3147526947+gzip"
Proxy-Connection: keep-alive
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 xxx

param1=1&param2=2
```

HTTP状态码

状态码	类别	含义
1XX	Informational（信息性状态码）	接收的请求正在处理
2XX	Success（成功状态码）	请求正常处理完毕
3XX	Redirection（重定向状态码）	需要进行附加操作以完成请求
4XX	Client Error（客户端错误状态码）	服务器无法处理请求
5XX	Server Error（服务器错误状态码）	服务器处理请求出错

你说的是POST,那Respond的呢

响应报文结构：

第一行包含协议版本、状态码以及描述，最常见的是 200 OK 表示请求成功了
接下来多行也是首部内容
一个空行分隔首部和内容主体
最后是响应的内容主体

```
HTTP/1.1 200 OK
Age: 529651
Cache-Control: max-age=604800
Connection: keep-alive
Content-Encoding: gzip
Content-Length: 648
Content-Type: text/html; charset=UTF-8
Date: Mon, 02 Nov 2020 17:53:39 GMT
Etag: "3147526947+ident+gzip"
Expires: Mon, 09 Nov 2020 17:53:39 GMT
Keep-Alive: timeout=4
Last-Modified: Thu, 17 Oct 2019 07:18:26 GMT
Proxy-Connection: keep-alive
Server: ECS (sjc/16DF)
Vary: Accept-Encoding
X-Cache: HIT

<!doctype html>
<html>
<head>
    <title>Example Domain</title>
 // 省略...
</body>
</html>
```

TCP和UDP的区别

略

面经24

> @author 壮

分库与分表

一、分库

1、概念：

是为了解决服务器资源受单机限制，顶不住高并发访问的问题，把请求分配到多台服务器上，降低服务器压力。

2、分库的原则：

一般分库都是按照业务划分的

会针对一些特殊的库再作切分，比如一些活动相关的库都做了拆分【做活动的时候并发可能会比较高，怕影响现有的核心业务，所以即使有关联，也会单独做拆分】。

3、分库带来的影响：

（1）事务问题

使用关系型数据库时，需要保证事务的完整性；分库之后单机事务就用不上了，必须使用分布式事务来解决，而分布式事务基本的都是残缺的。

（2）连接问题：

跨库之后无法使用JOIN。

4、解决方案：

在业务代码中进行关联，也就是先把一个表的数据查出来，然后通过得到的结果再去查另一张表，然后利用代码来关联得到最终的结果。
适当的冗余一些字段。比如以前的表就存储一个关联 ID，但是业务时常要求返回对应的 Name 或者其他字段。这时候就可以把这些字段冗余到当前表中，来去除需要关联的操作。

二、分表

1、概念：

是为了解决由于单张表数据量多大，而导致查询慢的问题。大致三、四千万行数据就得拆分，不过具体还是得看每一行的数据量大小，有些字段都很小的可能支持更多行数，有些字段大的可能一千万就顶不住了。

2、分表的方式：

（1）垂直分表

把常用的，不常用的，字段很长的拆出来！是指表数据列的拆分，把一张列比较多的表拆分成多张表。

表的记录并不多，但是字段却很长，表占用空间很大，检索表的时候需要执行大量的IO,严重降低了性能；把一些无用的数据拆分出去，一页就能存放更多行的数据，内存存放更多有用的数据，就减少了磁盘的访问次数，性能就得到提升。

（2）水平分表

一张表内的数据太多，数据越多 B+ 树就越高，访问的性能就差，所以进行水平拆分。

3、分表带来的问题

排序、count、分页问题：用户的数据被拆分到多个表中，那查询结果分页就不像以前单张表那样直接就能查出来【count同样如此】；

解决：只能由业务代码来实现或者用中间件将各表中的数据汇总、排序、分页然后返回。【另外： count 操作的结果其实可以缓存下来，然后每次数据增删都更新计数】

路由问题【分表的路由】：

分类	Hash路由	范围路由	路由表
概念	选择表中的某一列，然后进行 Hash 运算，将 Hash 运算得到的结果再对子表数进行取模，这样就能均匀的将数据分到不同的子表上。	表示一定的范围；可以基于时间或者地址	专门搞个表来记录路由信息
优点	数据分布均匀	容易扩展	灵活‘迁移数据’，直接迁移后修改路由表即可
缺点	增加子表的时候比较麻烦，对HashMap的扩容，需要迁移数据	数据可能分布不均匀【例如：某个月搞了活动，日志特别大；BJ用户特别多】	多一次查询；每次都要查询访问路由表，有对应的缓存设计。
关于拆分键【Sharding-Key】设计问题	采用冗余数据的方式，无论如何对应拆分的列都不可能满足所有需求		

全局主键问题：

分表之前采用自增主键来 保证全局主键的唯一性

还是自增，根据子表数目来改变自增步长。比如现在有三张表，

步长设置为3，三张表 ID 初始值分别是1、2、3。这样第一张表的 ID 增长是 1、4、7。第二张表是2、5、8。第三张表是3、6、9，这样就不会重复了。

通用唯一标识码【UUID】，这种最简单，但是不连续的主键插入会导致严重的页分裂，性能比较差。

分布式 ID，Twitter 开源的 sonwflake 雪花算法，或利用 redis 来递增也行。

面经25-Java-百度二面

> @author 习惯过了头

面试分4个部分：自我介绍、项目、基础知识、算法。总共问了35分钟

基础知识：
spring事务
反射原理
lock底层实现
线程池实际应用及如何设计参数
数据库事务隔离级别
动态代理两种实现方式
hashmap底层数据结构
concurrentHashMap如何保证线程安全
双亲委派原则

算法：两个栈实现队列、链表

项目和算法都答得不错，基础知识答了4题，反射原理不会，还有3题没有答到核心，没有给我反问的机会，然后面试完看官网状态显示挂了

面试刚开始，面试官说我这边有点吵，我解释说家附近有公路，货车经过会比较吵，已经关窗户了。后来面试官那边就开始有小孩子吵闹的声音，不知道是不是故意的还是怎样，刚开始不会。

后面做了广州某公司的笔试题，有道编程题

题目描述：

设计一个生成50万个（由0-9组成的，6-15个字符组成）的不同随机字符串。
要求：
a.不能出现连续3个以上相同数字（比如：111234、2333456）
b.随机字符串不能出现3个连号的情况（比如：123456、123457、123458）
c.生成的字符串是无规律可循的（比如不能是等差数列）
d.要求算法尽可能高效而且少占用内存

面试复盘：

复盘了一下，发现还是有挺多问题的：语速太慢，说话不够有力，项目介绍不够流利，重复语句较多。基础知识答得不够全面。算法逻辑说得不够清晰，演技待提高。

以前面试没有录音继续复盘，现在复盘还是发现的挺多问题的

建议：

面试官指定不是故意的啊，哈哈哈，面试官应该是有家事的，小孩子就在旁边， 不过 你的 语速慢，不够流利，重复语句多，这些毛病都是 应届生刚开始面试的通病，这次面试之后 反思一下，整个面试流程自己练几十遍，要说的足够流利，不能总想着心里知道怎么回答 不就说出来了，一定要说出来，平时这块就要练一练的

面经26-Java-京东一面

> @author 习惯过了头

西瓜今天晚上8点京东面试，先是在牛客找了10篇面经，收集问题
早上把算法题做了一遍
下午复习了面经中的Java基础、集合、多线程、JVM
吃完饭复习操作系统和网络的面试题

然后面试，感觉答得良好，复盘后感觉答得一般，面经如下：

电话面试，大概36分钟
自我介绍+项目+基础+场景题+反问环节

项目

说一下项目难点以及如何解决

Java基础
线程的几种状态，以及状态的转换
阻塞状态和等待状态的区别
怎么实现线程安全
锁怎么实现线程安全
synchronized可重入吗，为什么需要可重入
悲观锁和乐观锁
怎么判断JVM里是否出现死锁
如何预防线程死锁

MySQL
为什么数据库索引用B+树，而不用list、map、二叉树或红黑树

计组
二进制有原码，为什么还要有反码和补码

场景题
一个数组，可以不断地添加元素，而不出现数组下标越界异常。怎么实现?
有A、B两个大文件，每个文件几十G，而内存只有4G，其中A文件存放学号+姓名，而B文件存放学号+分数，要求生成文件C，存放姓名和分数。怎么实现?

其他
是否接受工作地点调剂

反问环节
岗位的业务方向或所在部门

感觉答得还可以，后面面试官也说如果复试面试官筛选通过会通知二面，当时感觉应该可以过，回来查看状态发现挂了

拥抱反思：
场景题第2题没答好，之前看过，没去思考，计组问题也没答好，不知道红黑树特征，Java线程和锁还有待提高。

月亮几个月前准备春招的时候，老师说过，在没面试的时候，跟着自己的规划走，有面试的时候，可以提前准备一下，面试完后心态不会崩，可以从面试中查漏补缺，不会因为一次面试失败导致规划混乱。当时基础太差，做不到这样，现在开始有点感觉了。

面经27-C++-京东一面

> @author Nx

京东一面：

1. 自我介绍
2. 说说网络服务相关的内容
3. 进程和线程
4. 为什么使用线程池
5. 内存池使用过吗
6. 如何实现线程池
7. TCP和UDP的区别
8. HTTP报文都包含什么
9. HTTP如何保持长链接
0. 了解HTTPS吗?

反问:

1. 培养新人的流程
2. 面试官的部门：做音视频偏通信方向的，底层使用C++,业务上使用Java

面试官还问我没有实习经历，项目是怎么来的？研究生阶段做了什么项目？
我说看书，看博客，介绍了下毕设项目

原定十点开始，十点半结束，面试官十点十分左右来的
整体流程很快，20分钟结束，没有算法题，感觉面试官挺累的样子，好像不是很想招人
最后说了一句自学的基础还不错。

一面来的很突然，昨天中午发的邮件，只有今天上午十点的面试时间
现在官网显示的是复试未安排，也不知道有没有二面

面经28

> @author 壮

多进程与多线程的区别

维度	多进程	多线程	总结
概念	同一个时间里，同一个计算机系统中如果允许两个或两个以上的进程处于运行状态	多线程是一种执行模型，它允许多个线程存在于进程的上下文中，以便它们独立执行但共享其进程资源	
数据共享、同步	数据是分开的，共享复杂，需要用IPC；同步简单	多线程共享进程数据，共享简单；同步复杂	各有优势
内存、CPU	占用内存多，切换复杂，CPU利用率低	占用内存少，切换简单，CPU利用率高	**线程**占优
创建销毁、切换	创建销毁、切换复杂，速度慢	创建销毁、切换简单，速度快	**线程**占优
编程调试	编程简单，调试简单	编程复杂，调试复杂	进程占优
可靠性	进程间不会相互影响	一个线程挂掉将导致整个进程挂掉	进程占优
分布式	适应于多核、多机分布；如果一台机器不够，扩展到多台机器比较简单	适应于多核分布	**线程**占优
应用场景	需要频繁创建销毁或者进行大量计算的优先用线程 扩展到多机分布的用进程，多核分布的用线程		

同步、异步和互斥

同步

发出一个功能调用时，在没有得到结果之前，该调用就不返回或继续执行后续操作。

对于多进程的同步而言：表明多进程存在直接制约惯性系，有一定的先后执行顺序；

进程同步方式：

1. 临界区：对临界资源进行访问的那段代码【为了互斥访问临界资源，每个进程在进入临界区之前，需要先进行检查】
2. 信号量：是一个整型变量。可对其进行down和up操作；即为常见的P和V操作，down和up操作被设计成原语，不可分割。【down：大于0执行-1；等于0，进程休眠，等待信号量大于0；up：信号量+1操作，唤醒睡眠的进程让他完成down操作】
3. 信号量取0和1即为互斥量【0：表示临界区已经给加锁，1：表示临界区解锁】

异步

当一个异步过程调用发出后，调用者在没有得到结果之前，就可以继续执行后续操作。当这个调用完成后，一般通过状态、通知和回调来通知调用者。对于异步调用，调用的返回并不受调用者控制。

1. 状态：即监听被调用者的状态（轮询），调用者需要每隔一定时间检查一次，效率会很低。
2. 通知：当被调用者执行完成后，发出通知告知调用者，无需消耗太多性能。
3. 回调：与通知类似，当被调用者执行完成后，会调用调用者提供的回调函数

同步与异步的区别：请求发出后，是否需要等待结果，才能继续执行其他操作。

互斥

多个进程在同一时刻只有一个进程能进入临界区【对临界资源进行访问的一段代码】。

面试29-C++-多家公司

> @author 壮

字节一面【7.27】

1. 继承、封装、多态相关解释？
2. 继承是否破坏了封装？
3. 问STL容器了解哪些【自己应到说了熟悉vector，和list主要问了关于vector如何扩容的机制】
4. vetor添加元素的方式？下标遍历是否会涉及到迭代器？
5. 基于范围的遍历与迭代器遍历区别是什么？
6. STL容器引出，数组和链表的区别？
7. 多线程的通信方式？
8. 并发与并行的区别？
9. 关于git的命令【项目引出】
0. 手撕代码：对应的是leetcode.138.复制带随机指针的链表

百度一面【7.27】

1. 协程有了解？协程【用户级线程：用户自己来管理】具体是如何实现的？【需要看源码】
2. 线程与进程的区别？
3. 假如实现一个线程级别的日志系统，有一个日志模块，需要多线程去调用，需要考虑效率；如何去设计、有哪些关键的数据结构，来完成多线程之间的切换与处理？I/O交互的部分【用到了哪些数据结构、调用哪些接口，内部流程是怎样的？】
4. UDP与TCP的区别？
5. TCP传输中的常见问题：丢包、乱序、重复的解决方法？
6. 主要用过哪些容器（我引出了vector，然后问了底层实现？以及扩容的机制是什么？小场景：有一个一个对象构造和析构均会print，它在扩容的是时候，涉及到拷贝操作，是否会打印？）

7. 智能指针有用过？说一下有哪些智能指针？关于智能指针的安全性【看源码】
8. 如何自己实现智能指针【例如：shared_ptr】？计数器变量，定义成一个静态变量，调用构造函数+1；调用析构函数-1；其实核心要问的：多线程访问的机制？【说了加锁来保证多线程，但是太慢；让去了解STL源码智能指针的线程安全的级别】
9. 虚函数有了解嘛？具体访问时，虚函数表指针的偏移量是如何计算出来的？如何真正找到我想到访问的函数？【如何通过手动的方式强行调用虚表中的函数？】
0. 正常函数与虚函数的区别？ 正常函数是强行计算地址；虚函数时通过偏移+的方式
1. 有用过GDB嘛？
2. 常用的设计模式有哪些？关于简单工厂模式对外提供怎样的访问接口？
3. 编程题：反转链表【需要自定义链表结构体 + 创建链表的函数（头插/尾插）】对应leetcode 206. 反转链表

百度二面【7.30】

1. 存储器的分层以及每一层的大小是多少？以及作用
2. new申请内存，操作系统所做的操作流程是怎样的？【问的底层实现，回答的不是他想到的答案】
3. 异常ball，实层与虚层？
4. 多线程通信的方式？
5. 代码题：随机出一道工程应用题，具体记不清楚，没写的很好；估计挂了

图森未来一面【7.28】

1. 虚函数、多态
2. 构造函数、析构函数可以是虚函数嘛？
3. 右值引用概念、为何要引入右值引用？
4. 智能指针概念以及应用场景？
5. shared_ptr当出现相互引用后，具体分析如何解决相互引用的【有点深】？
6. 多线程与多进程概念、应用场景？
7. 条件变量与锁
8. 多进程间的通信方式
9. 整个项目突然崩溃了，该如何找到错误的位置？平时是如何去调试的？
0. 代码题：问题描述：实现支持取出最接近元素对的存储结构。代码要求:请设计一个类 ClosestStorage，实现下面三个函数：
    1. insert(item)：插入一个元素 item，如果它不存在。否则什么也不干。
    2. delete(item)：删除一个元素 item，如果它存在。否则什么也不干。
    3. getClosest()：返回整个结构中最接近的一对元素（如果存在多对，任意返回一对）；所有的item都是 int64 类型

shopee测开一面【7.29】

1. C++的三大特性?
2. 多态实现原理
3. 为什么索引会加快查询的速度?
4. 索引的分类有哪些?
5. TCP如何保证可靠性传输的?
6. HTTPS和HTTP的区别?
7. 有哪些设计模式?
8. 知道哪些测试工具?
9. 平时写代码是如何进行测试的?
0. 代码题：手撕快速排序

拼多多【7.31】

1. 多进程与多线程区别以及各自的通信方式?
2. 为什么需要引入虚拟内存?
3. 代码题：搜索旋转排序数组；对应leetcode33.搜索旋转排序数组

ps：有些忘记了录音，记得就少一点；一周面了六场，两场笔试，人都傻了；还是要会引导到自己熟悉的，这样不熟悉就不会继续往下问了；核心还是刷题！！！目前最大的问题还是如果题目没有见过类似可能写出来一堆问题！！希望下周二面都好好加油！！！！

面经30

> @author 本人

字节提前批一面

本人

- 自我介绍
- 项目
  - 第一个项目:DH算法的实现及中间人攻击
  - 第二个项目
    - 简单问问
  - DH算法的主要实现
    - 不流畅
  - 加密算法
    - 去年学过但全忘了
    - 对称和非对称的区别
    - 对称加密算法有哪些
    - 非对称加密算法有哪些?
    - AES和DES有什么区别?
    - RSA了解吗?
- 操作系统
  - 程序动态加载的过程?
  - 编译的过程?
    - 链接的具体操作?
  - 进程同步
- 数据库
  - 没有复习,简历上就没写，也问了
  - 常用命令
- 计网
  - TCP和UDP的区别
  - OSI7层协议分层
  - TCP三次握手
  - 应用程序的协议有哪些?
    - 举例不够多
  - HTTPS加密过程
  - 有什么方式能为UDP提供可靠性?上层应用怎么实现?
- 算法题
  - 题目:两数之和改良版
    - 给一个数组，和一个target，寻找数组两个元素和>=target且最接近target的下标
  - 我给出的解决方案
    - 差点给我整不会了
    - 暴力搜索
      - 时间复杂度O(n^2)
      - 空间复杂度O(1)
    - (双指针是类似leetcode中三数之和的解决方法)
    - 数组排序+哈希表+双指针
      - 时间复杂度O(nlogn)
      - 空间复杂度O(n)
- 其他
  - 工具使用
    - 抓包用过哪些工具?
    - WireShark捕获数据包的原理是?
  - Linux
    - 常用命令有哪些
  - 漏洞
    - 你知道有哪些具体漏洞吗?
    - 栈溢出漏洞了解吗?
- 反问环节
  - 您认为这道算法题该怎么解决比较好?
    - 面试官没给答案，说算法没有最优解，每个人写的都不同(我估计是他是临时出来的，他也没想好)
  - 那对于算法主要看重什么呢?
    - 回答:思路和代码的可读性

面经31-京东二面

> @author Nx

京东二面（20分钟结束）

1. 自我介绍
2. 介绍课题（问了许多）
3. 项目来源
4. 使用多进程通信了吗，都用了哪些函数？（我听得迷迷糊糊，回答锁，原子操作，条件变量，不明白他究竟想问啥，就差把代码写给他看了）
5. Linux命令，打开端口（使用过，但记不清了）
6. Linux命令，查找100个文本文件，将含有abc字段行数统计出来（没答出来）
7. Linux命令，管道了解吗？（就是"|"这个，回答说看TCP连接状态时使用过）
8. 工作想做什么具体方向
9. 都投过哪些公司
0. 对语音业务有兴趣吗

二面没通过，很糟糕，感觉面试官看了我的简历不知道问啥，问我在学校的项目，说了半天，反馈给我说虽然我听不懂，但是能感觉到你做了不少东西。

整体感觉就是面试官不知道问啥，我不知道答啥，二十分钟一到，面试官就让开始反问，我问他做什么业务的，他说技术方向，我不知道他做技术吗？

还聊到了语音业务，我说本科的时候了解一点，他反问到那什么711编码你知道吗？哎，本科就接触了奈奎斯特定律和霍夫曼编码，说了一点，他就不说话了，还催我说有下一场面试。

面试官整体态度很好，但是明显感觉到他没做什么准备就来了，这种问法防不住，而且在Linux命令上确实掌握较浅，我问为什么没有问算法方面的，他说后面还会有考试？我又疑惑了，不过无所谓，我已经挂了

面试挂了，情绪down了一天，不知道该怎么做了，面经一遍遍看，但是现在的面试官不问，整个人傻掉了，数据库没问，操作系统没问，计算机网络没问，设计模式没问，编程语言不问，抓着课题和Linux命令问，我也是服气了。

面经32-C++-拼多多+字节

> @author 壮

拼多多二面

项目
两道算法题
合并两个有序数组并去重
找给定的两个节点深度最大的祖先节点

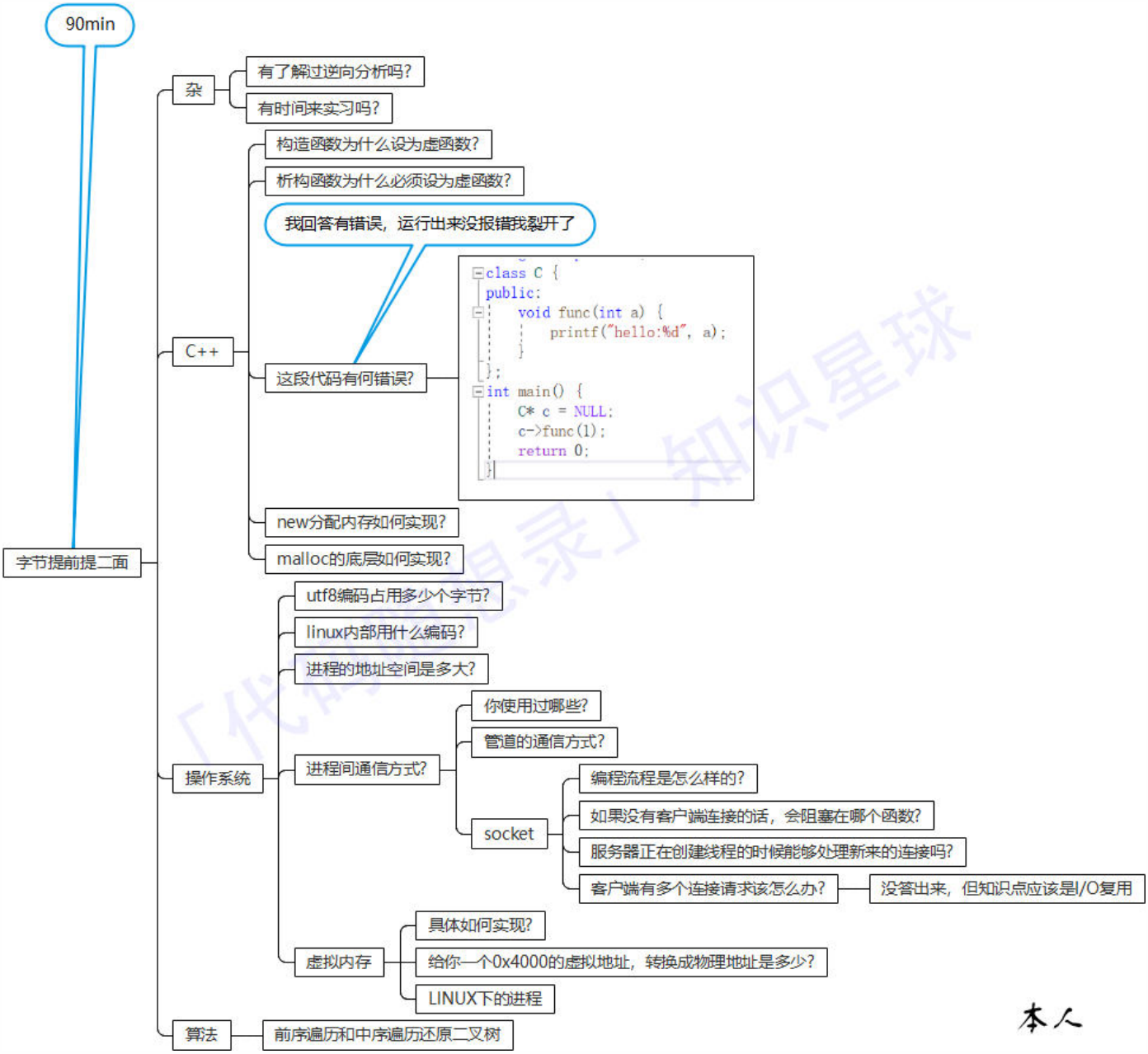
字节二面

项目
智能指针的实现原理、以及有哪些常用的指针?
分析我和面试官视频通信信号如何传递的? 【计算机网络涉及到的五层协议】
路由器转发的原理是什么?
如何统计全世界会钢琴的人数?
给一个字符串中各个字符按照空格分隔开，找到出现频率最高的K个字符?
代码题：汉诺塔【没写出来】；换了一道题：找旋转数组中的最小值

ps：已经三场面试都是和旋转数组相关的了，庆幸第一面试遇到没有做出的时候，找了整个相关的都做了一遍，面试看来还是要多面，复盘很重要！！！另外问一下，面试官问关于今后职业规划该怎么回来更好呢?

面经33

> @author 本人



面经34

> @author 壮

虚函数的开销问题：

一、表面上开销

空间开销：

包含虚函数的类生成一个虚函数表，致使程序的二进制文件大小会相应的增大；
类的实例包含一个虚函数表指针，致使对象实例空间占用增加一个指针大小【32位系统，4个字节】

时间开销：

增加了一次内存寻址，通过虚函数表指针找到虚函数表，虽对程序性能有一些影响，但是影响并不大

二、背后的开销：【更深入的解释】

从汇编层生成的代码来看，普通函数与虚函数的区别是，普通函数是一个直接调用，而虚函数是一个间接调用，直接调用与间接调用的区别就是跳转地址是否确定，直接调用的跳转地址是编译器确定的，而间接调用是运行到该指令时从寄存器中取出地址然后跳转；

直接调用而言，是不存在分支跳转的，因为跳转地址是编译器确定的，CPU直接去跳转地址取后面的指令即可，不存在分支预测，这样可以保证CPU流水线不被打断。

而对于间接寻址，由于跳转地址不确定，所以此处会有多个分支可能，这个时候需要分支预测器进行预测，如果分支预测失败，则会导致流水线冲刷，重新进行取指、译码等操作，对程序性能有很大的影响。

面经35

> @author weikunkun

一面

一面（1h）

1. 面试岗位的了解
    1. 简单的阐述了下部门，然后说了面试的Java开发岗位
2. 自我介绍
3. 项目介绍
    1. 介绍项目的背景以及解决了什么问题
    2. 这个业务需求的痛点在哪儿
    3. 重难点
        1. 业务难点
        2. 技术难点
4. 介绍下策略模式、其他的设计模式了解那些
    1. 阐述了策略模式
    2. 其他的直白了说了没实际应用过，只写过demo，就没有细问
5. 分布式锁的介绍
    1. 为什么出现分布式锁
    2. 实现分布式锁需要注意的事项
6. Redis的底层 String的实现，让自己设计，用C如何实现。(当初没仔细看SDS的设计，就按照ArrayList来设计的)
    1. 如何设计数据结构

2. 扩容策略
3. 缩容策略
    1. 什么时候执行
4. 频繁阈缩容如何解决
    1. 惰性删除 ⚠️
5. 过期策略
    1. 有哪些过期策略
    2. 如何设计过期策略的执行，value设置什么值?
        1. map? ？ 存时间戳? ⚠️
7. 买卖股票
    1. 时间复杂度、空间复杂度
    2. 给了两种思路
8. 36匹马求前3名
9. topK
    1. 根据数据量和k的值来去定方法，整体方法有
        1. 基础排序
        2. 堆排序
        3. 快速选择
        4. 基数排序
        5. 分治
10. 场景题:
    1. 业务场景:
        1. 关于钱的问题，在数据库中的存储，以及实际编码阶段上对钱的操作
            1. 因为精度问题，一般钱的存储不直接使用float、double这种以二进制形式表示的数据类型，而是使用字符串表示的数据类型。譬如使用Decimal、Numeric。`salary DECIMAL(9,2)` (能表示的最大面额，以及保留几位小数)
            2. Java编码，用BigDecimal，根据业务需求选择保留1、2位，是否需要四舍五入，或者其他的舍弃形式
    2. MySQL什么情况下不走索引，给出具体的示例场景
        1. select * from student where score = 100; --- 回表
        2. select xxxx, xxx from student where score + 1 = 100; --- 索引列不干净
11. 反问:
    1. 部门职能划分介绍
    2. 技术栈

二面

二面（1h）

1. 自我介绍
2. 实习为什么不考虑转正?
3. 岗位申请是成都，计划留在成都?
4. 深入项目
    1. 项目背景、解决了什么问题
    2. 分支太多了，如何解决，之前有写过博客，然后就说的挺全的
        1. 提前return，去除不必要的else
        2. 使用三元运算
        3. 使用枚举
        4. 使用optional
        5. 表驱动法，Map<?, Function<?> action> actionMappings = new HashMap<>();
        6. 优化逻辑，让正常流程走主干
        7. 策略模式+工厂方法消除if-else
    3. 不同接口不同，这个是怎么做的
        1. 厂商的参数变了，如何满足
            1. 把所有的参数封装成一个对象，然后在对象里面获取， 最后再根据配置中心的内容，执行不同的逻辑
            2. 重载
            3. 所有的api抽象成一个配置，然后修改某个字段，再结合配置中心来实现
5. 算法：二叉树转链表，要15min中做完。。。
    1. 最开始没听清楚，直接前序遍历，然后再构造ListNode，尾插法做了
        1. 递归、非递归都写了一遍
    2. 后面说不要新建ListNode，然后直接在TreeNode上做
        1. 后序遍历解决了
6. 有没有什么喜欢的框架、组件这种，分享下
    1. spring
        1. 就把最近看Spring揭秘的内容都讲了一遍
7. Object有哪些方法
    1. hashcode、equals、wait、notify、notifyAll
    2. wait如何实现的
        1. ？？？ 不是native方法吗
        2. 然后开始尝试分享了两种思路（还好心态没崩，面试官也在鼓励我，就开始天马行空了，最后感觉就是monitor的实现）
            1. 对象头做标志位
            2. monitor的实现原理
7. 反问
    1. 技术分享
    2. 为了能胜任这个岗位，需要对哪些内容做深入学习
8. 评价
    1. 项目上还可以深挖，能做的更好
    2. 学习建议
        1. 带着问题去学习，主要是思想
        2. 长期来看需要对某个方向有一个深入的了解，作为自己的竞争点

weikunkun

三面（40min）

1. 学校经历讲一些，有成就感的
    1. 本科实验室的内容
    2. 研一数学建模比赛
    3. 实习经历
2. 问了数学建模比赛的内容
    1. 问了建模比赛的内容，使用了什么模型这些
    2. 问了线性回归的大概内容
3. 问了算法相关的课程
    1. dfs、bfs、贪心、动态规划这些
    2. 详细阐述下动态规划
4. 设计模式了解吗
    1. 详细阐述了策略模式
    2. 单例模式、代理模式、工厂模式带过
5. IoC、AOP
    1. 结合Spring来讲了一大坨
6. 问了自己的GitHub的相关的内容，然后讲了获得⭐最多的项目
7. 分享出来之后，大家都拿来干什么
    1. 学习
8. 对我们的部门有了解吗
    1. 阐述了之前在面试官那里了解的信息
9. 如果愿意接受百度的机会的话，有意愿来百度实习吗
    1. 来，肯定来，必须来
10. 什么时候毕业?
11. 之前实习还在做吗
12. 反问
    1. 问了技术分享相关的内容
    2. 询问对个人的评价
13. 问了我对百度的认识
    1. 讲了技术驱动、2019年的春晚红包历程
14. 确认了应聘岗位
15. 问了后序的事情
    1. 有测评
    2. 问了手头的offer，如果给了你offer，该怎么选择呢?
16. hr的速度不会特别的快，需要耐心等待，然后面试官说个人还是很看好我的

weikunkun

> @author 壮

分库

1、概念：

是为了解决服务器资源受单机限制，顶不住高并发访问的问题，把请求分配到多台服务器上，降低服务器压力。

2、分库的原则：

一般分库都是按照业务划分的

会针对一些特殊的库再作切分，比如一些活动相关的库都做了拆分【做活动的时候并发可能会比较高，怕影响现有的核心业务，所以即使有关联，也会单独做拆分】。

3、分库带来的影响：

事务问题

使用关系型数据库时，需要保证事务的完整性；分库之后单机事务就用不上了，必须使用分布式事务来解决，而分布式事务基本的都是残缺的。

连接问题：跨库之后无法使用JOIN。

解决方案：

在业务代码中进行关联，也就是先把一个表的数据查出来，然后通过得到的结果再去查另一张表，然后利用代码来关联得到最终的结果。
适当的冗余一些字段。比如以前的表就存储一个关联 ID，但是业务时常要求返回对应的 Name 或者其他字段。这时候就可以把这些字段冗余到当前表中，来去除需要关联的操作。

二、分表

1、概念：

是为了解决由于单张表数据量多大，而导致查询慢的问题。大致三、四千万行数据就得拆分，不过具体还是得看每一行的数据量大小，有些字段都很小的可能支持更多行数，有些字段大的可能一千万就顶不住了。

2、分表的方式：

（1）垂直分表

把常用的，不常用的，字段很长的拆出来！是指表数据列的拆分，把一张列比较多的表拆分成多张表。

表的记录并不多，但是字段却很长，表占用空间很大，检索表的时候需要执行大量的IO,严重降低了性能；把一些无用的数据拆分出去，一页就能存放更多行的数据，内存存放更多有用的数据，就减少了磁盘的访问次数，性能就得到提升。

（2）水平分表

一张表内的数据太多，数据越多 B+ 树就越高，访问的性能就差，所以进行水平拆分。

3、分表带来的问题

排序、count、分页问题：

用户的数据被拆分到多个表中，那查询结果分页就不像以前单张表那样直接就能查出来（count同样如此）；

解决：

只能由业务代码来实现或者用中间件将各表中的数据汇总、排序、分页然后返回。（另外： count 操作的结果其实可以缓存下来，然后每次数据增删都更新计数）

路由问题（分表的路由）：

分类	Hash路由	范围路由	路由表
概念	选择表中的某一列，然后进行 Hash 运算，将 Hash 运算得到的结果再对子表数进行取模，这样就能均匀的将数据分到不同的子表上。	表示一定的范围；可以基于时间或者地址	专门搞个表来记录路由信息
优点	数据分布均匀	容易扩展	灵活‘迁移数据’，直接迁移后修改路由表即可
缺点	增加子表的时候比较麻烦，对HashMap的扩容，需要迁移数据	数据可能分布不均匀【例如：某个月搞了活动，日志特别大；BJ用户特别多】	多一次查询；每次都要查询访问路由表，有对应的缓存设计。
关于拆分键【Sharding-Key】设计问题	采用冗余数据的方式，无论如何对应拆分的列都不可能满足所有需求		

分表之前采用自增主键来 保证全局主键的唯一性

还是自增，根据子表数目来改变自增步长。比如现在有三张表，步长设置为3，三张表 ID 初始值分别是1、2、3。这样第一张表的 ID 增长是 1、4、7。第二张表是2、5、8。第三张表是3、6、9，这样就不会重复了。

通用唯一标识码（UUID），这种最简单，但是不连续的主键插入会导致严重的页分裂，性能比较差。

分布式 ID，Twitter 开源的 sonwflake 雪花算法，或利用 redis 来递增也行。

面经37-算法

> @author Douglas

技术面试问了差不多八十分钟，然后问技术问题就差不多问了一个小时了。说的我口都干了。
question:
自我介绍

DBNet介绍一下
DBNet的训练和推理有什么差别
DBNet的预处理与处理过程
还用过什么文本检测算法，
说下psenet和east
说下crnn的ctc算法
crnn识别的字典集包含多少字符
还用过哪些文本识别方案
表格识别的方案
有没有了解过其他公司的ocr方案
文本旋转矫正的方案
竖直方向字体的识别
模板匹配怎么实现的

YOLO3和YOLO5的区别
anchor怎么生成的
yolo中anchor的使用
RPN网络
FPN网络
yolo的框是怎么预测，回归出来的
目标检测常用的损失函数
目标检测发展现状，比较新的技术
yolo的分支
讲一下centernet
还用过哪一些one stage目标检测算法
各种IOU

有没有使用过分割算法
语义分割和实例分割区别
有没有用语义分割做过卡证的检测

OpenCV用过哪些函数
使用过OpenCV的模板匹配吗

数据增强中的随机裁剪实现
用过什么数据增强
仿射变换与透射变换怎么实现的，用于数据增强中效果

ncnn的细节:哪家公司开源的，推理时用到哪几个文件，哪个是权重，哪个是网络结构，哪个文件比较大，有没有打开看过，修改过

用什么工具看网络结构
模型转换时有没有遇到什么不支持的算子
用于哪些平台
还记得运行时的FPS吗
部署过哪一些设备
移动端还有哪一些部署方案
推理时有没有遇到要特别处理输入尺寸，为什么

做检索任务时用的tricks，以及有什么比较提升，哪个提升大一些
做这个项目的心得
介绍一下当中的损失函数

基本就记得这些了，后面就是介绍公司团队，业务，然后问我有什么想了解的。

我问:

- 算法团队怎么样
- 公司数据集的数量，计算资源如何
- 想招个什么样的人
- 团队的价值观
- 长短期的发展目标

这次技术基本就是围绕做过的项目来问当中的细节。估计我做过的项目，面试官他们公司都做过类似的，所以也在提问的当中加入了他们遇到过的问题。
问的问题整体不难，但是就比较细，没有亲身做过，估计答不出。面试官感觉也不错，不像有些面试官好像不太懂技术，或者说在套取方案。感觉第一次技术聊了这么多，这么久。

整体下来认为回答的也还可以，公司业务看起来也比较符合我预期，技术栈。可惜就是人事开的薪资有点低，17K，打算节后再看其他机会了。

本人二本本科非科班毕业，两年工作经验。刚入职现在这家公司不到半年，因公司在以后都推迟半个月才发放工资，所以不得已出来找机会。只想说太难了，以目前的学历，工作经验太难找好的了。最近公司也似乎开始疯狂试探底线，拼命压榨员工，这几晚都加班到九点多。今天请假还没得调休。

幸好之前一直保持学习，如今不至于很慌，虽然现在也很焦虑，不知道之后怎么选比较好。国庆加紧学习，把各种结构网络，损失函数，激活函数等再看看，也继续刷力扣的题，但是我面试碰到的都是不用写题的，也继续看C++内容。

未来，道阻且长，共勉。

最后，也提前祝各位国庆快乐。

面经38-Java-美团

JVM

1、反射

忘了相关八股文了，用类加载过程产生的堆中 class 对象来解释的，面试官说不算错也不算对

2、堆是唯一可分配对象的地方？栈中的对象和堆中的对象内存分布有啥区别

不是，如果不发生逃逸分析，那么可以分配到栈中。

区别:

堆中的对象内存分布为：对象头，实例数据，填充部分。栈中的对象会经过变量分离操作， 将引用类型对象分解为多个基本数据类型直接存放在局部变量表中（好久没看 jvm 了，这个我会的 QAQ）

3、为什么 Java 对象占用的内存大小是 8 的整数倍

以某些空间的代价使其能够实现更快的访问内存。

操作系统

1、锁升级过程

2、Java 中的锁，乐观锁悲观锁

回答了 cas 和 syn 和 Lock

3、如何实现乐观锁

并没有搞懂想问什么，是 actom 原子类吗?

4、看过原子类的源码吗

当然没有

5、创建线程的方法

Thread 类 callable 接口和 runnable 接口。callable 接口会返回一个 Future 的对象，表示执行的结果 线程池

6、线程池的常用参数

阿巴阿巴，这个没啥

MySQL

1、B+ 树

按照 MySQL 是怎样运行的里面的解释了 B+ 树，讲了叶子节点是完整的记录，内节点是索引目 录，其值是对应的叶子节点中的索引列最小值和叶子节点的页号（这个面试官表示疑惑，但是我觉 得自己肯定没错），页和页之间是双向链表，记录和记录之间是单向链表，还有页内使用槽 + 二分查找来快速确定记录位置巴拉巴拉，能记得的都说了

2、为什么要用自增的主键

ID 如果是随机的，可能导致分页操作，会造成内节点的更改，也许会导致更多的分页操作，带来大量 的 IO，影响性能。如果是自增的主键 ID，可以保证增加记录的时候是按照页面的顺序往后添加的。

3、B+ 树，B 树的区别

老八股文了

4、联合索引，联合索引列（如 where a=,b=,c=）如何确定是 a b c 的顺序

联合索引就老八股文了，带了自己的理解解释了下 第二个问题，我回答的是 如果某一列频繁的需要查找和排序/分组，就应该放在前面，可以实现索 引的复用，降低内存空间浪费。面试官问，如果查询频率一样呢，我说，那么索引列值小的往前 放，如果索引列值是一个 char 类型的很长的字符串，那么匹配字符串就很浪费时间。

5、覆盖索引

老八股 ++

6、union /union all 的区别

不会哈哈，好久没写过 sql 了 非科班垃圾

数据结构

1、二叉树的遍历方式

前中后层

2、根据前中后能否复原二叉树

前中，中后可以，我举例子解释了前中怎么复原（不过好像回答的不是他想要的）。前后不行

3、为什么前后不行

回答的是前后的话无法确定根节点和左右子树。然后陷入了好一会的冷场

计算机网络

1、五层网络模型

老八股了，但是我忘了，自己硬扯的

2、数据链路层的作用

这个忘了，离谱

3、三次握手和四次挥手

自己从头开始说，说了为什么要三次握手，为什么不能两次，为什么不用四次。四次挥手，为什么要四次。

4、为什么要等 2MSL

八股 ++

大数据题

1、一个很大的数据文件，找其中最高频的 10 个

不会，没见过。

问了解决方法，他说自己可以了解一下，思想很简单的，用过就懂

算法

1、反转链表

面经39-腾讯云计算

> @author mingze

腾讯是boss上投随便投的，投的深圳，然后被捞走去北京的。绿牌实习生。

面试：自我介绍

项目：

算法俩和牛客这位一样，不一样的是俩算法写完不算完，得优化一下。

然后问你咋这么多奖，我就吹了30分钟左右的牛，说说比赛的事情，这些奖咋回事，面试官好像挺愿意听，将近2小时吧，结束。一面时间耽误在编译器上了，vs出现bug，调试好久一直以为自己写错了，然后重启vs就好了。。。天杀的微软，差点以为挂了。

二面，很快

自我介绍

问点基础，然后算法约瑟夫环

写了一个链表实现，然后改进成数组，然后再改进成递归数学实现，有问能不能再优化我就说递归变递推，啊吧啊吧

然后就是同不同意转go语言啥的问题，结束，就是工资少，需要朋友救济。得合租了。房价太贵，本没想去腾讯，一直想去百度。没想到去了腾讯，emmm，想到朋友6面技术面，去字节，我太容易了。

面经40-客户端-字节

> @author 欧尼蒋

我投的是字节的抖音客户端开发，事情是这样的，前两面都挺正常，自我介绍八股文算法，答挺好的，然后面评应该也不错。第三面上来的就是部门负责人了，具体的面试内容在下图中。

字节三面(1h20min)

1. 写一道算法题吧：看了下是力扣两数之和的变体

   排序 + 双指针简单解决，未考虑极端情况，然后面试官没有让我分析代码说明思路，直接问：如果让你设计测试案例，你会怎么做？

   接下来我说了几种比较常见的边界：

   1. 数组为空或者元素个数小于2
   2. 数组元素大数情况（即做加法的时候会发生溢出）
   3. 存在重复项满足条件的情况，如 arr = [3, 3, 5, 5], target = 8, 最终的结果应该是四个
   4. 不存在合适解，返回空

   接下来面试官让我检查算法是否能够处理这些情况，但是由于未考虑到重复项，前面的方法作废了，然后想了想可以用哈希表实现，哈希key记录元素的值，value记录值为key元素的索引（可能有多个，所以用了数组vector去存储），然后就是排序加双指针大框架去做遍历，只是在找到答案的时候需要进行O(N^2)的组合（全组合问题，需要二重循环）

2. 第二道算法：一共n个人（编号 0 ~ n - 1），初始每个人手中都有糖果，交换他们手中的糖果，保证均匀分布，并且每个人不会拿到原来的糖果（重点是如何保证均匀性）

其实我个人感觉还挺好的，算法慢了点(两道40+min包括沟通)，基础部分也还好，正常回答。但是结果就是挂了。

后续是hr帮忙联系了内推的学长最后我被捞起来了，然后应该是转岗了，这样就来到了第四面，这一面也是最无语的一次，什么内容都往死了扣，我说一个常见的情况他偏不听，比如问我智能指针都是干啥用的，我说解决内存泄漏(当然了，给他解释了原理以及一般的应用)，面试官就说要是我不内存泄漏呢，还有什么别的用处吗，就那种不太想听我知道的这种八股，专门挑刺恶心wo。好在最后的算法a得比较快(面试官只给了15min左右，一道中等题)，感觉他想挂我，但没想到写出来了，问我写过没有(这肯定回答没写过)。

五面挺好的，问的都很常规，写了几个算法，生产者消费者模型以及单例吧，不太记得了，其余的都是老八股给他整得妥妥的。

本来以为五面就是技术最终面了(面之前hr跟我说这是最后一面技术面，完事就hr面了)，没想到还有下一轮部门负责人的技术面，当时我人都傻了，心态爆炸啊，想了想还是算了，继续面吧。

就这样来到了第六面，面试内容如下图，整体下来效果超级好，所有的问题都答上了。

字节六面

1. 快排（写了个简单的）
2. 优化一下快排（减少了函数调用）
3. 智能指针？
4. shared_ptr循环引用的小demo
5. 面试官在编辑框写代码让判断是否出错，为什么？（考了智能指针引用普通指针的错误、前向声明）
6. 循环数组下一个更大的元素

以为这就结束了？哈哈哈，六面结束的时候反问了面试官还有几面，他跟我说两面？还有两面？当时人麻了。还好事后问了hr，今天下午两点半安排了hr面。

顺利来到了hr面，我也是准备妥当了，昨天晚上把字节范看了几遍，又捋了捋常见的问题。面得很轻松，问了些项目的问题，比如：

- 项目是怎么来的，学校的大作业？
- 既然是团队开发，那团队是哪些人，你们又是怎么分工的？
- 遇到了问题发生过争吵吗？
- 怎么解决争吵的？
- 遇到了技术难题你一般怎么做？
- 评价一下你在项目中做的怎么样(10分给几分)？
- 那对你团队成员的评价呢？
- 你觉得上学阶段最艰难是什么时候？
- 对工作的方向有什么规划吗？
- 如果能来，工作多久呢？

一大堆，不一一列举了，但是我回答的超级好，都是自己的亲身经历，HR说我很正能量很阳光，嗯，就是这样了，希望hr面能过，这要是被挂了我都不到该怎么面对生活了。

面经41-实习

> @author 在线求offer

今天终于把offer定下来了，来应一下卡哥之前的实习经验分享吧

我有两段实习经历，就按时间顺序来介绍一下吧。

一些经历：

本科的时候没有出去实习过，而我又是一个比较爱规划的人，读研的目的就是为了找工作，所以在我拿到硕士录取通知书的那一刻起，就决定不再搞本科的那些什么三好学生，去争一等奖学金之类的，就算拿不到奖学金我也要出去实习（好在我们学校是奖学金全覆盖，所以我没去加那些分也有奖学金拿，感谢学校😂）。

研一刚上来的时候，其实不知道到底要去找什么岗位的实习，组里的研究方向是nlp，但是刷知乎牛客，清一色说算法要双985的，学历劝退，所以我就开始在开发岗里选，本科的时候有搞过前端，但是当时感觉前端的上限太低了（没有说前端不好的意思），所以就去学了java后端，众所周知java后端的技术栈是很庞大的，然后就开始按一些技术帖子上的路径逐步开始学习，但与此同时为了毕业，nlp的知识也会同步学习。

第一段实习：

因为20年的那波疫情，研一下学期没有开学，每天在家的学习效率也是很低，到了三月末感觉自己不能再这么堕落下去了，就开始去投简历，但也不知道自己想去或者能去哪个岗，所以各个岗位都有投，算法开发测试，看见哪个投哪个那种，研一还有试错的机会。

但当时也不尽如人意，因为当时还不懂招聘的规则，没有刷题，就简单准备了一些面试问题，直接就开始面试，结果也可想而知，挂的很惨，比秋招挂的还惨。

最后到了5月末被一家top的安防公司选中了，去做了nlp实习生，一方面去学习知识，另一方面看看自己到底适不适合这个岗位。

第二段实习：

研二上学期回学校根据实习学到的内容憋了一篇论文出来，因为感觉算法岗还是挺看论文的，有一篇至少是加分的，写完初稿大概到了12月中，准备再出去实习一下，其实这中间岗位选择也有一些波折，因为经历了实验室师兄师姐的秋招，考虑到算法岗学历不占优势，考虑要不要转去测开岗，至少能冲一冲大厂，所以当时决定过年前再试一下算法岗，没找到合适的再去转测开岗，然后就开始投算法，但是这回找工作比第一次有经验了，提前一个月就开始刷题了，本来做好持久战的打算了，结果没想到投的第二家公司就面过了，公司也算个1.5线大厂，然后没有犹豫就去了，这里有一个时间点，下面会具体说。

面经：

越大的厂，面试流程及难度越趋近与秋招，所以刷题八股就按照秋招那么准备就行，但是实习面试没有秋招要求那么高，如果有几个没答上来也没关系，整体上可以应该就可以过。牛客什么的多刷刷面经，应该没啥问题。

日常与暑期：

实习生其实是分日常与暑期实习生的，以上说的都是日常实习，暑期实习生其实要求会更高一点，是针对下一年应届毕业生准备的，一般在春招那阵进行招聘，流程和难度近似于秋招，转正留用概率很大，但也得看组里具体hc，可以先打听好，情况不对赶紧准备秋招。

时间点：

其实实习我觉得有几个时间点是比较好找的，第一个时间点是12月中旬到过年之前，年底了会有好多人从一线城市跑路回家，其实这个时候公司是挺缺人的，招的越急，面试越容易过；第二个时间点是8月中下旬，这个时候暑期实习生陆续开学返校，实习生岗位会有很大空缺。而且实习的话，大家尽量说可以实习的久一点，3个月是至少的，最好是能6个月往上，单从时间长度就能压倒一片人。

请假：

没有经历过本科实习请假，所以这里就不说了，来说一下硕士实习请假的问题吧，这个很大一部分要看导师，如果导师完全放养不管还好，出去就出去了，但我导师是属于那种一定程度push型的，所以你要跟导师达成某种“协议”，比如写完论文专利或者项目结题再走，反正就是要顺着导师来，为了实习忍一忍就过去了，千万别闹掰，毕业要紧。

总结：

实习是很重要的，尤其是我这种双2的学历去混在双9的堆儿里找算法岗的学生，实习经历真的能很大程度帮助你提升简历，至少不至于连面试都进不去，但也不是去实习就行了，是真的要从实习里学到或者做好一些东西，有一定成果，而且在面试里能讲明白，这样才是加分的。我感觉其实作为在校生，简历就看这么几点：学历学校、实习经历、论文、比赛，这几点能占的越多越加分。

面经42-go-字节日常实习

> @author 🅰 🅽 🅺

平时在星球看了不少，大家每天打卡都很努力，最近找到了实习，写一片面经来回馈社会。

blog地址

牛客地址

基本情况：

博主末流985，软件工程2019级本科，绩点中下，主要使用Golang，无竞赛，一个学校Web开发程序设计实践的课程设计的curd项目，leetcode200道左右。

岗位投递：

字节跳动-基础架构-后端开发 base成都 内推 2021-12-12简历投递 12-20简历评估通过 12-21安排面试

> （投递简历后被别的部门锁了一周的简历，可是只投了一个部门，不是很懂为什么会被别的部门锁简历）

字节一面

> 时间：2021-12-22 时长：1h10min

1、自我介绍

2、OSI网络7层模型

OSI七层模型与TCP/IP五层模型 - yinrw - 博客园 (cnblogs.com)

3、tcp和udp的区别

一文搞懂TCP与UDP的区别 - Fundebug - 博客园 (cnblogs.com)

4、一个MTU最大是1500字节，那么最多包含多少的数据

ip头部字节大小为20~60，最多数据即1500-20=1480

5、tcp三次握手是否能够减少为两次?

TCP为什么是三次握手？两次、四次握手不行吗？ (baidu.com)

6、golang中常用的并发模型

Golang并发模型 - Go语言中文网 - Golang中文社区 (studygolang.com)

这个地方复盘的时候发现面试官应该是想问channel和一些同步原语，但是答到csp和gmp上面去了，又因为gmp能讲的比较多，就没有主动说下去了。所以总体答的很浅，两三句就说完了。。。

7、进程、线程、协程、goroutine区别

进程、线程、协程、goroutine区别SlagSea-CSDN博客goroutine是线程还是协程

8、linux中线程的状态

Linux进程状态解析 之 R、S、D、T、Z、X (主要有三个状态)sdkdlwk的博客-CSDN博客进程状态d

9、golang中有几种锁

mutex、rwmutex

这个地方也是只回答了两种锁，也可能本来问的比较干。。。这个地方应该主动提出自己了解这两种锁的一些底层实现，需要我讲一下嘛？

10、go中变量分配在什么地方

变量分配

11、go的gc

图示Golang垃圾回收机制 - 知乎 (zhihu.com)

12、mysql的ACID

什么是事务,事务的四个特性是什么kayla的博客-CSDN博客事务的四个特性

13、分布式的cap理论

CAP 定理的含义 - 阮一峰的网络日志 (ruanyifeng.com)

14、提问：cap中的c和acid中的c有区别吗

回答：个人认为有一定的区别。cap中的c，主要指在分布式事务中，保证数据的强一致性。acid的c，主要指mysql的状态一致性，数据库的约束。

面试官解答：acid的c其实就是指所有数据的一致性。

15、mysql的存储引擎了解的有哪些

mysql 的存储引擎介绍 - 张冲andy - 博客园 (cnblogs.com)

16、innodb和MyISAM的区别

MyISAM与InnoDB 的区别（9个不同点）张花生的博客-CSDN博客innodb和myisam的区别

17、innodb中主键索引和非主键索引都是聚簇索引吗

回答：主键索引是聚簇索引，非主键索引不是聚簇索引，因为非主键索引的叶子节点只会存主键和index，所以会有索引下推，索引覆盖，回表。

18、事务回滚的实现

mysql 事务 回滚 原理mysql 事务的实现原理weixin39660931的博客-CSDN博客

19、mysql主从架构

回答了主从复制机制和主从架构的作用

20、提问mysql双主架构在不分表的情况下保证数据一致性

回答：pc两阶段提交

21、如何避免mysql双主架构出现会循环的数据更新

回答：使用一个标志位，来标记请求的来源，如果是来自用户则执行后广播至其他主节点，如果来自其他主节点则执行。

面试官对于回答没有做任何评价

> eg.用户在节点A插入，节点A插入后通知节点B插入，节点B插入后又反过来通知节点A来插入...

22、项目最大的难点

23、算法

给定一个数n，如23121;给定一组数字A如{2,4,9}，求由A中元素组成的、小于n的最大数。如小于23121的最大数为22999。（时间关系没有让写测试用例）

反问：部门主要从事什么样的工作，主要业务。

面试结束，当即告知一面通过，能否直接二面。回复有急事，能不能过1-2小时，面试官说可以，结果被鸽了。。。第二天中午收到面试通过邮件，选择二面时间。

总结：一面主要以基础为主，也会问到相关编程语言底层实现，进阶知识。算法不难。

字节二面

> 时间：12.27 时长：50min

1、自我介绍

2、讲一下项目

curd项目，很简单

3、自己在项目实现了哪些

全部，前端后端

4、项目体量有多大

不是很懂怎么衡量项目体量，于是说了实体类有多少个，有多少张表

5、项目数据库有哪些表

6、说到了外键

提问：外键的约束的作用

mysql 中的外键作用_MySQL - UCloud云社区

没有准备到，乱答用于连接两张表。

7、你觉得项目的难点在哪里

答curd项目，很简单。面试官没有多问。

8、golang的gmp模型

Go语言的GPM调度器是什么？ - 掘金 (juejin.cn)

9、gmp中m和p的数量关系

10、go的gc

图示Golang垃圾回收机制 - 知乎 (zhihu.com)

11、了解docker镜像

答：image相关

12、了解docker网络吗

答：不了解，但是了解docker资源隔离，于是回答资源隔离

聊一聊Docker所使用到的Linux底层技术(Namespace,Cgroup与存储驱动和容器引擎)小凯的博客-CSDN博客

13、了解k8s吗

答：听说过，不了解

14、tcp状态机的切换

即三次握手和四次分手过程中，客户端和服务端的过程状态。

详解 TCP 连接的“ 三次握手 ”与“ 四次挥手 ” (baidu.com)

重点在于四次分手时，timewait和closewait

15、tcp滑动窗口，拥塞控制

TCP滑动窗口和拥塞控制机制详解genzld的博客-CSDN博客tcp窗口滑动以及拥塞控制

16、Linux内核了解吗

Linux内核_百度百科 (baidu.com)

17、内存分页、分段

分段和分页机制续航fff的博客-CSDN博客分段和分页

18、os内存伙伴算法

没答上，不知道。。。面试官说太过底层了吗?

伙伴算法csdnkou的博客-CSDN博客伙伴算法

19、算法

给定一个二叉树，请计算节点值之和最大的路径的节点值之和是多少。这个路径的开始节点和结束节点可以是二叉树中的任意节点

思路：

使用递归，从结束节点开始向上返回节点值，其余节点返回零值。左右子节点有非零返回值，则说明子节点处于路径中，返回子节点值+自己的节点值。到开始节点时，将值保存在全局变量res中。

> 复盘时，感觉这道题题意有点怪，这个路径有开始节点和结束节点，那不是直接计算这条路上的节点值之和不就好了吗，需要强调节点值之和最大的路径吗？前半句感觉是使用回溯，找出最大的通路。后半句感觉是遍历到开始节点和结束节点节点，用递归标记。感觉就很怪。

没有反问，面试结束，当即告知二面通过，能否直接三面。回答可以，休息10min后，开始三面。

总结：和一面没有什么区别

字节三面

> 时间：12.27 时长：40min

1、自我介绍

2、介绍项目

3、用户态和内核态的区别

怎样去理解Linux用户态和内核态？ - 知乎 (zhihu.com)

4、为什么要区分用户态和内核态

资源隔离和通过系统调用接口提供硬件资源

5、golang中使用goroutine使用系统调用会阻塞线程吗

6、如果golang中所有goroutine调用一个系统，会导致没有线程可用吗

答：gmp中m不会耗尽，但是os的线程会耗尽

7、追问gmp中m是什么

M代表OS内核线程，是操作系统层面调度和执行的实体。M仅负责执行，M不停地被唤醒或创建，然后执行。M启动时进入的是运行时的管理代码，由这段代码获取G和P资源，然后执行调度。另外，Go语言运行时会单独创建一个监控线程，负责对程序的内存、调度等信息进行监控和控制。--《Go语言核心编程》

8、mysql事务隔离级别

一文讲清楚MySQL事务隔离级别和实现原理，开发人员必备知识点 - 风的姿态 - 博客园 (cnblogs.com)

9、不可重复读是什么

10、故障测试了解吗

不知道

11、ut了解吗

不知道

12、了解docker的XXX（忘了）

答：只了解资源隔离，就没问了

13、go闭包的一道题

```
func calc(base int) (func(int) int, func(int) int) {
  add := func(i int) int { base += i
  return base }
  sub := func(i int) int { base -= i
  return base }
  return add, sub
}
func main() {
  f1, f2 := calc(10)
    fmt.println(f1(1),f2(2))
    fmt.println(f1(3),f2(4))
    fmt.println(f1(5),f2(6))
    fmt.println(f1(7),f2(8))
}
```

请说明上述程序的输出是什么?

```
11 9
12 8
13 7
14 6
```

14、两个list求相同元素

题目描述:

（1）有两个有限队列，求两个队列的相同元素

（2）对自己的代码进行测试用例编写

（3）通过自己的用例，优化自己的代码

（4）说出自己代码的时间复杂度

（5）如果是两个无限队列，怎么办

思路：哈希表

反问：部门主要业务?

答：大方向存储相关

总结:

三面leader面，问了一些场景题，比较考验理解，八股文很少。中间一问三不知，后面两道代码题也没做好，面到一半感觉裂开，觉得g了。

> 面试官中途一直说，我不太懂go你能给我讲讲这个吗？ 我不懂go，帮不了你哦。 你确定是这个答案吗？ 我直接裂开

第二天hr打电话说通过了，约了hr面，感谢面试官放海。

HR面

> 时间：12.31 时长：15min

1、你主要的经历有哪些呢

2、根据经历提问

3、会转正吗

答：会，于是介绍了转正流程

4、为什么想来到字节呢

答：1.技术成长。2.字节文化

反问：实习生培养流程

总结：日常聊天，可以提前看看字节范公众号了解一些。

面经43-前端-知乎暑期实习

知乎一面

1.自我介绍➕项目介绍

2.你觉得你项目最难的点是什么

3.你vue3.0用的多还是vue2.x用得多，说一下你认为vue3.0和vue2.x的差异是什么

4.了解react吗？说一下react的生命周期函数

5.了解react hook吗

6.说一下你的第一个项目为什么要用webassembly，它的强项是什么，为什么不用js原生库呢(因为js没有好用的audio库啊😭

7.说一下vue你用过什么指令

8.你刚刚提到了vue-module，说一下vue是怎么实现双向数据绑定的

接下来是css部分

1.怎么在不知道宽高的情况下实现水平垂直居中

2.flex布局说一下

3.flex:1是什么意思

4.BFC布局

然后下面是js部分

1.给一段代码说出输出是什么，为什么是这样，总共两段代码，第一段比较简单，第二段有点复杂

2.说一下promise你用过哪些方法，然后又问我使用promise.all需要注意什么吗

3.说一下闭包

4.说一下js的事件循环机制

5.说一下ES5和ES6的继承实现有什么不同

6.了解TS吗，说一下TS和js有什么不同，TS的类型断言是什么

7.说一下TS的泛型

8.了解跨域吗，说一下怎么解决

9.sessionstorage和localstorage有什么异同

10.说一下https和http的区别

知乎二面

1.自我介绍➕闲聊

2.熟悉react还是vue

3.你踩过什么vue3.0的坑吗

4.说一下BFC

5.说一下js变量提升

6.反转链表

7.手写节流防抖

8.用数组方式实现一个栈

9.提问

面经44-微软SDE

微软一面

1.处理SQL插入语句

微软二面

单调栈相关

微软三面

模拟比赛，需保证一方一定赢

面经45-测开-外企社招

高通

高通一面

智力题：两头烧绳子计时，64匹马找最快4匹马，蚂蚁爬绳子

算法题：链表是否有环，两种方法写斐波那契数

概念题：Tuple List 区别 ，Python2/3区别，一道一句代码完成两个List合并成Set

高通二面

项目经历：Pytest框架相关的问题，考察测场景的设计

高通三面

项目经历：（代码为主）项目中的设计框架，手写一个简单的框架表示。

有AI算法经历，问了AI算法相关的问题，如BP传播，激活函数对比等。

Python的基本概念，Yield使用方法，Nonlocal Global区别，内存管理。

高通四面

项目经历：装饰器，Np.array 和 List 的区别，Linux常用命令。

TCP/IP三次握手四次挥手，滑动窗口流量控制，HTTP和HTTPS的区别。

描述一个你遇到的困难，怎么解决的。

Nvidia

Nvidia一面

有AI算法经历，问了几个AI相关的问题，比如朴素贝叶斯、MLP原理、BP、偏差方差，过拟合怎么调试。

英文面试：描述一个遇到的困难，如何解决的

Nvidia二面

项目经历：Python相关问题，装饰器，生成器等。

AI相关的：BP传播原理。

Nvidia三面

项目经历：AI相关的问题，CN在CV上咋用的，Pytorch写BP，GPU和CPU的区别，Linux常用命令，进程线程的区别，进程间怎么通信，线程间怎么通信，进程锁。

Nvidia四面

Pytest框架，用到哪些模块，Fixture怎么用，Dependency怎么用，怎么传参，Jenkins的用法。

C++熟不熟悉，C++和C的区别，C++和Python的区别。

如何跟踪一个测试问题，测试日报关键要素，如果遇到拉通上的问题怎么解决。

Nvidia五面

沟通能力：如何请求高级资源，如何协调Delay的项目，领导有不同意见你怎么处理，测试问题跟踪，与开发观点不一致如何解决。

你自己的优缺点，期望薪资。

Intel

Intel一面

项目经历：Python中的简单概念，项目里KNN考虑哪种距离及原因，MLP网络、KNN算法公式口述推导过程，MLP层数的计算题，策略类强化学习Reward设计，模型优化方向的问题讨论。

Intel二面

Linux常用命令，例如：Vim快捷键，常用查看Log的命令，路径相关，权限相关问题。

Python和Pytest的相关问题。如何评价测试完成度？测试完成度与产品质量的关系，如果有一个没有交付标准的产品转测，你怎么做用例设计

Intel三面

项目经历：Git常用命令，微信红包相关的测试场景设计。

Python相关：用装饰器实现单例模式（口述），生成器使用方法，Yield、Return的区别。

AI的相关：什么是过拟合，如何解决等。

Intel四面

MLP、GBDT模型大小，适用场景；Attentiom算子，MLP网络结构，DDQN网络结构，Tricks。PG和AC的区别，Reward设计。

模型优化从哪些方面考虑，考虑哪些性能，对测开的看法。

面经46-Java-蚂蚁暑期实习

蚂蚁一面

1. 自我介绍
2. 介绍一下硕士期间的研究方向
3. 项目是自己做的还是照着别人的做的？改进的点？
4. 说说Redis缓存和数据库的数据一致性
5. Redis主从复制
6. Redis持久化机制了解吗？
7. Redis是单线程还是多线程？为什么？
8. 说说类加载机制
9. Java内存区域了解吗？
10. MySQL索引了解吗？为什么不用B树？为什么不用二叉树？为什么不用hash？
11. MySQL存储引擎有哪些？区别是什么？
12. SpringBoot和Spring Cloud区别？
13. Spring框架了解哪些？
14. 了解MongoDB吗？
15. 了解RPC吗？
16. 了解NIO吗？
17. 多线程的创建方式有哪些？
18. 说一说线程池的七大参数
19. Java并发了解吗？JUC包了解哪些？
20. CountDownLatch和CyclicBarrier的区别？
21. 线程和进程的区别
22. 进程间的通信方式有哪些？
23. 看过哪些JDK源码
24. 给一个大文件，电脑内存有限，如何给这个大文件中的整数排序？
25. 职业规划
26. 有什么爱好
27. 什么时候能实习？导师放实习吗？
28. 反问：对我面试的评价？对今后学习的建议？

蚂蚁二面

1. 自我介绍
2. 项目用的什么框架?
3. 项目用的什么数据库?框架和数据库的连接用的什么?
4. 了解MyBatis缓存机制吗?
5. 为什么选Redis缓存?为什么不直接用MyBatis自带的缓存?
6. Redis有什么优缺点?
7. Redis持久化机制的缺点?
8. 进程间的通信方式有哪些?
9. MySQL存储引擎了解哪些?
0. 说一说CMS垃圾回收器
1. synchronized和ReentrantLock性能上的区别?
2. 介绍下研究方向
3. 多方安全计算了解吗?工业界的一些相关框架了解吗?
4. 为什么做开发不做算法?
5. 什么时候能来实习?
6. 反问:部门用的技术栈?

蚂蚁三面

1. 自我介绍
2. 讲讲项目或者论文或者比赛的一两个亮点
3. 聊聊论文
4. 为什么做开发不做算法?
5. 导师放实习吗?
6. 反问:如果这轮通过,后面还有几轮?

蚂蚁HR面

1. 自我介绍
2. 说说项目或者论文或者比赛中的难点
3. 说说你的优势和劣势
4. 为什么投递蚂蚁?
5. 还投递了什么公司?考虑上海的机会吗?如果都拿到offer会选哪个?
6. 反问:什么时候会有结果?

面经47-客户端-字节暑期实习

字节一面

1. 为什么想来客户端开发?
2. 介绍一下自己（要多往客户端开发方面靠一靠）
3. 了解客户端的技术栈吗?
4. 挑一个你觉得最好的项目介绍一下
5. HTTP请求结构?
6. get post 区别
7. HTTP底层基于什么协议?
8. UDP和TCP区别是什么?
9. TCP如何保证可靠传输的?
10. time wait状态了解么？为什么要有这个状态?
11. select和epoll的区别?
12. reactor模式解决了什么问题？（游双）
13. 了解proactor模式吗？说说和reactor模式的区别
14. 进程和线程的区别说一说
15. 进程中的数据段包含的内容有什么？（PCB）
16. 虚拟内存的原理？操作系统在内存映射中做了什么方面的工作?
17. 分页机制（包括页命中、页异常、页面置换）
18. 分页和分段机制懂不懂？缺页异常了解吗?
19. 指针、地址和引用的区别是什么？（指针对类型也有一定的抽象）
20. 具体描述一下C++中多态的作用？（原理+具体描述）
21. 堆和栈的区别?
22. 对智能指针有哪些了解?
23. 使用了多态的设计模式？（工厂模式）
24. STL容器用过哪些?
25. vector和list区别?
26. map和unordered_map区别?
27. 算法题：有序数组统计target数量（也就力扣34. 在排序数组中查找元素的第一个和最后一个位置）

字节二面

1. 职业规划问题
2. http1.0 1.1 2.0 3.0区别
3. io多路复用是什么？说说其优势，从操作系统的角度来讲讲
4. http2.0 的服务器推送具体怎么实现的?
5. HTTP底层除了TCP和UDP还能用什么协议？我回答的TLS协议
6. 算法题：问了一道LeetCode 3 无重复最长字串

字节三面

1. 学校相关问题
2. 学校里学的啥
3. 项目里用的局部敏感哈希算法说一下
4. 说说你的服务器项目吧
5. 你觉得swift和C++区别在哪里
6. 递归会不会？会出现什么问题，怎么优化
7. 进程和线程的区别
8. 碎片知道吗？内部碎片外部碎片区别？怎么解决
9. 算法题：斐波那契数列，要求递归来写，并优化

面经48-go-深信服暑期实习

1. 自我介绍、项目介绍；
2. defer相关知识 的考察（读代码题）；
3. 树的子结构， leetcode 原题目；
4. Go 的 panic 和 recover 原理，panic 的是什么；
5. panic 可以 被其他 recover 捕获吗；
6. map 的数据结构，及一些相关机制；
7. go map 线程安全吗；
8. 说说线程安全的 map ，sync.map 和 cmap 的实现原理；
9. goroutine 的结构；
10. 讲讲 gmp，分别讲讲 g m p（go 有什么特殊的办法防止 协程长期抢占一个 m）；
11. goroutine 的 开销小，为什么小；
12. channel 的结构，同步与异步，同步的实现原理；
13. Go 的 gc，三色标记、插入写屏障、删除写屏障、混合写屏障，提问这几个发展的区别，混合写屏障还有 STW 吗？
14. 了解哪些锁；
15. 流量控制、拥塞控制；滑动窗口有几个，讲讲接收端和发送端等等；
16. https；
17. 四次挥手；四次挥手为什么要 time_wait，为什么是2 MSL；
18. 了解 websocket 吗？
19. tcp 和 http 分别在哪一层，讲讲 why http；
20. 有些 rpc 的实现，使用的是私有协议，为什么不用 http？
21. 二进制和基于文本协议的区别，他们底层不应该都是二进制吗？
22. 操作系统的僵尸进程与孤儿进程；
23. 僵尸进程过多会怎样；
24. Linux 的文件系统；
25. Linux 的内存管理；
26. Linux 的进程调度机制；
27. 死锁、死锁的产生条件；
28. MySQL 的 隔离级别与实现机制；
29. mvvc，怎么实现；
30. MySQL 的锁怎么实现；
31. 几个日志； 索引、结构、区别、为什么；
32. 事务是怎么实现的；

3. 框架和库的区别；
4. gin 框架讲讲；
5. 了解哪些 Nosql（redis、mongodb）；
6. docker 和 k8s；
7. pod deployment service 都讲讲；
8. 为什么用 deployment 不用 pod；
9. 保活机制；
0. orm 了解多少等等；

面经49-算法-抖音推荐暑期实习

> 个人背景：C9本硕，23届非科班，从20年开始准备转算法，去年有一份二线互联网公司推荐算法实习，无论文无科研经历
>
> 3月15号约面试，21号一面，24号二面，29号三面，31号HR面；4月2号收到了offer。

抖音一面

1. 聊实习项目，深入提问
2. 机器学习基础：AUC的物理意义，为什么常用？梯度消失和梯度爆炸为什么出现，怎么缓解？用过word2vec，讲讲它的原理等等
3. 写代码：给定一个词-频率的hash map和一个样本长度l，按照词频采样l个词。分析复杂度

抖音二面

1. 聊实习项目。问到一些图embedding的方法、PCA的原理等
2. 开放问题：样本不平衡的问题
3. 概率题：一个圆上均匀任取三个点，连成一个三角形，是锐角三角形的概率
4. 代码题：翻转字符串里的单词（LeetCode easy）

抖音三面

1. 聊实习项目。推荐相关的开放性问题
2. 代码题：与一面的题一样，本来是要写的，但面试官好像没搞好链接，就让我口述了一下。分析复杂度，还能怎么优化。

面经50-客户端-字节暑期实习

> 个人背景：本科末流211，硕士末流985，三月初开始背八股文，看阿秀的求职笔记以及CS-Notes还有卡哥的八股文，背了一两周就开始面试了。项目可能稍微多一些，有三个关于C++的项目，基本是有关深度学习的，深挖一下还是能挖不少东西出来，在面试时还是有的聊。简历上也没写数据库和设计模式，主要是没怎么接触过，只背过数据库的八股。

字节一面(八股为主，1h)

1. 自我介绍 1分钟
2. 聊项目 5分钟（有一个Android的深度学习部署项目，用C++实现，比较贴近部门）
3. 多态，重载重写
4. inline
5. inline能否用在虚函数
6. 可变参数如何实现，只讲了main函数（后面查了一下，是用va_list解决）
7. 强制类型转换
8. 智能指针
9. const变成可变的（mutable）
10. const*和*const 区别没回答出来，只记得区别（值不变和地址不变）
11. 数据库说不太了解没问
12. new和malloc区别
13. map和unordered_map实现
14. 红黑树特点，与avl树区别
15. hash表，负载因子，扩容，冲突解决
16. 多线程 死锁 预防死锁
17. 自旋锁 互斥锁
18. 线程除了加锁以外的方法 条件变量 信号量
19. 输入URL过程
20. tcp和udp区别，适用场景，在哪一层
21. socket实现
22. socket能不能模拟http（可以）
23. http2.0介绍
24. 是否了解新的协议 quic http3.0
25. python迭代器（简历里写了解python）
26. 算法题:最长回文字符串

字节二面(深入项目，1h)

1. 自我介绍 1分钟
2. 聊项目 15分钟
3. Main函数执行前发生了什么
4. 为什么基类析构函数要求是虚函数
5. 安卓的四大组件
6. 安卓动画
7. 如何使用c++编写安卓（jni、ndk），在项目中具体如何实现
8. i+1会比i小吗（溢出）

9. 多态
0. 模板
1. 如何链接dll，指令
2. cmake
3. 跨平台，在不同平台上（比如32位和64位）int不一样，怎么实现相同位数？（宏定义判断机器字节数）
4. 算法题：LRU

字节三面(场景设计题，1h)

1. 自我介绍 1分钟
2. 聊项目 20分钟 包括介绍项目、询问深度学习、介绍研究生方向等
3. 询问学习的课程
4. 静态链接库和动态链接库区别、优缺点
5. c++编译过程
6. c++ static
7. 进程间通信
8. 多态实现
9. 虚函数表
0. 秒杀系统（扯了一下消息队列、静态页面还有缓存，说实话没想到C++会问这个，之前被美团暴击后回去看了看，还是有用的）
1. 内存池（扯了一下内存分配算法八股，还有STL源码剖析里的内存池实现）
2. 单例模式（没接触过设计模式，所以换了一个二叉树的题）
3. 二叉树的路径总和等于某个target，输出全部（要建树）

HR面（二十多分钟）

1. 为什么选择字节?
2. 对面试岗位的业务有了解吗？（抖音界面的加号）
3. 平常用过抖音吗?
4. 这段时间有更深入的了解吗?
5. 聊项目
6. 最有成就感的事情?
7. 详细介绍了第一个项目
8. 询问人员构成（前端、Java后端、c++客户端、产品经理、老板）
9. 主要负责什么?
0. 询问团队是否出现过问题，如何解决?
1. 如何学习项目中遇到的新知识？（看博客，看官方文档）
2. 你的最大贡献是什么?
3. 最低谷，如何排解压力，如何平衡所有的事情?
4. 技术面试复盘，给自己打分
5. 面试中没有回答出的问题，后续是否有深入了解?
6. 除了上课和项目，平常通过什么进行学习（书籍、博客）
7. 最近看了什么书籍？举例一个印象最深的点
8. 实习时间

9. 实习城市
10. 职业规划
11. 有没有其他在投的实习？进度如何？
12. 希望从字节实习中获得什么？

面经51-Java-字节暑期实习

> 本科双非，211硕，非科班。技术栈Java
>
> 2-17号开学，开始着手准备项目，疯狂看八股文，3月8号开始投了第一家

字节一面

1. 自我介绍
2. 为什么选择计算机？
3. 进程和线程的区别？
4. 进程间的通信方式
5. TCP/UDP的区别
6. JVM内存模型
7. java的集合类
8. ArrayList,LinedList,HashMap
9. ArrayList和LinkedList的区别，使用场景
10. HashMap：数组+链表，为什么链表长度超过8，链表—>红黑树
11. MySQL索引
12. 讲到了索引底层的B+树，聚簇索引
13. 为什么使用B+树，不用其它数据结构？
14. BeanFactory
15. 算法题：判断链表是否是回文的
16. 反问：部门业务和技术栈

字节二面

1. 自我介绍
2. 学习计算机软件开发遇到的问题
3. 为什么要选择计算机
4. 这部分还聊了一些其它的，比如以后会不会继续做科研等等(让我一度觉得这是不是KPI面)
5. 为什么栈内存的空间使用效率高于堆内存？
6. 为什么把热点数据存入Redis里面，就能提高查询性能？
7. 为什么访问内存的效率要大于磁盘？(这里主要是接着上面Redis问的,这里答的不是很好，我回答了虚拟内存，页表，物理内存)
8. HTTP是短连接还是长连接？(回答的HTTP1.0/1.1/2.0/3.0，简单的说明)
9. 详细回答了HTTP1.0/1.1/2.0/3.0(这个问题回答了挺久，应该是个加分项)

0. UDP如何解决丢包的问题?
1. HTTP是短连接还是长连接跟HTTP是无状态的协议有没有关系?
2. 事务的隔离级别是解决什么问题?（这里我首先回答了读未提交带来的脏读问题）
3. 接着上面问了，为什么有四个隔离级别?（回答了还存在不可重复读，幻读问题，需要不同的隔离级别解决）
4. 算法：写了一个SQL(记得不是很清楚了)，给了三个表，大概是学生，课程，分数的一些字段，然后让查询课程的平均成绩大于等于60的学生信息(这里真记不清了，可能不是很准确)
5. 最长无重复的子串(与leetcode原题不同，要求输出这个子串，而不是输出长度)
6. 反问：给一些学习技术的一些建议

字节三面

1. 自我介绍
2. 介绍一下项目有哪些功能。
3. 拦截器是如何实现的?（很尴尬忘了拦截器的英文怎么拼了）
4. 多个拦截器的拦截顺序是怎样的?
5. 为什么使用md5的加密算法?
6. 如何实现使用随机验证码登陆的?
7. 关于发帖的一些功能实现?
8. 一个用户发特别多的帖子怎么办(答没有考虑到)?
9. 点赞帖子如何实现?
0. 如何实现保存是谁给帖子点过赞(用Redis里面的zset集合)?
1. 接着问了为什么是zset(答，用zset的score保存点赞的时间，这样是为了以后可以查询到按时间排布点赞用户)
2. Redis的持久化(RDB,AOF)
3. 做项目遇到的问题
4. 做项目有哪些收获
5. 算法：给一个日志类，然后让查询访问最多的十个IP，回答先用HashMap做统计，然后用大顶堆排序(面试官提示，其实用小顶堆更方便)
6. 手写堆排序(是因为上面提到了堆排序，可以看出来，面试处处都是坑，一个不小心就掉进去)
7. 统计在哪个时间点，精确到秒数，访问的用户最多。

面经52-Java-众安保险暑期实习

1. JVM类加载机制
2. nginx怎么做到负载均衡
3. HashMap的红黑树和扩容机制
4. 怎么用前缀树过滤敏感词
5. JVM垃圾回收机制
6. 线程安全
7. IO多路复用
8. 讲一下抽象类和接口的区别
9. 讲一下static和final的作用
0. 什么时候会不能用static
1. string可以被继承吗
2. synchonized 和 locked的区别
3. synchonized底层实现原理

4. 多线程的用过吗
5. 多线程创建的方式
6. 线程池怎么用，核心参数，整个执行的过程：包括消息队列
7. InnoDB和Mysam搜索引擎的区别，索引的区别
8. Innodb索引介绍下
9. Innodb的锁介绍下
0. 项目里Redis当中数据不一致性怎么解决的
1. redis三大问题介绍一下
2. Kafka如果有三个分区，四个consumer会怎么样
3. 反问

面经53-前端-百度暑期实习

百度一面

1. 自我介绍
2. 介绍项目
3. 项目的难点
4. 图片懒加载的原理与实现
5. JS的数据类型有什么
6. JS的事件循环原理
7. Vue.nextTick的原理
8. 模块输出题

```
//a.js
const b = require('./b.js');
console.log(exports.x);
exports.x = 'x';
require('./c.js');
//b.js
const a = require('./a.js');
console.log(a);
a.x='y';
//c.js
const a = require('./a.js');
console.log(a.x);
//问运行node a.js的结果
```

9. UDP与TCP的区别
0. 说一说HTTP协议
1. 手写防抖和节流函数
2. 反问（一般你的mentor就是你的一面面试官）

百度二面

1. 自我介绍
2. 介绍项目
3. 项目的难点
4. 组件库中按需引用的原理
5. B站弹幕遮罩的原理，弹幕穿过人物不会遮挡人物的原理
6. 反问

百度三面 + HR面

1. 自我介绍
2. 介绍项目
3. 项目难点
4. 闲聊（聊得相当愉快，很加印象分）
5. 常规HR面，问你兴趣爱好，学前端多久怎么学的，坚持最久的事情，最有挑战性的事情
6. 闲聊的时候聊到岗位，出了个思维题。有打车系统，现在有两个订单，后台根据什么去规划路线。
7. 反问

面经54-go-字节暑期实习

字节一面

区块链知识：

1. 讲下自己对区块链的理解
2. 讲下共识算法
3. 上面那个问题讲到了Raft协议，介绍下Raft协议
4. Raft协议中各节点的账本怎么达成一致
5. Raft中是否会有多个节点收到超过一半的投票？
6. Raft协议中序号是什么，投票id？
7. 问了简历上的Gossip算法
8. 区块链账本会不会被篡改？

web项目：

1. 问了简历上写的jwt
2. session和token的区别，为什么用token
3. token的缺点？
4. 当用户主动登出系统时，怎么让token马上失效

go语言：

1. 介绍下go routine
2. go routine怎么切换，讲下具体的细节

3. go routine切换时信息保存在哪里?
4. 介绍下defer

数据库:

1. 说下聚簇索引
2. 存储索引有什么数据结构
3. 用B+树和B树的区别
4. 为什么不用二叉树存储?
5. 想让我讲下红黑树，我说红黑树是c++的map底层数据结构，不太了解，说知道go的map的底层结构。然后非常详细地介绍了map底层（hmap，bmap中的字段，怎么进行hash（位运算替代取余），扩容机制（扩容时机，扩容原因，扩容后的操作，渐进式扩容）
6. 说下delete，drop和truncate的作用
7. 说下MVCC（介绍了mvcc是为了解决什么问题，隐藏字段，redo log，read review，整个流程包括db_trx_id怎么和read review中的字段比较，rr和rc下快照读的区别，rr级别下怎么解决不可重复读的问题）
8. 上面讲到了不可重复读，介绍下脏读，不可重复读，幻读的具体概念
9. 说下binlog，redo log和undo log

计算机网络:

1. http和tcp的关系
2. 为什么是三次握手不是两次握手
3. 什么是半连接

其它:

知道什么是布隆过滤器吗

编程题（25mins）

1. leecode 3.无重复字符最长子串（medium）
2. leetcode 200.岛屿数量（medium）

字节二面

数据库:

1. 数据库的索引存储结构
2. B树和B+树的区别，为什么用B+树不用B树
3. 为什么不用红黑树，可以说说红黑树吗

操作系统:

1. 线程是什么
2. 线程间的通信方式
3. 一个页面多大
4. 多级页表？回答了二级页面
5. 介绍下虚拟内存
6. 虚拟地址是怎么变到物理地址的
7. 内存是怎么分配的，回答了linux的伙伴算法

8. 僵尸进程和孤儿进程，分别是因为什么原因产生的

计算机网络：

1. tcp的三次握手流程
2. https介绍一下
3. golang
4. 介绍下go的channel（介绍了底层hchan里面重要的字段，读阻塞和写阻塞时的具体流程）

数据结构：

说说数组和链表的区别

编程题：

滑动窗口值求和。求数组内所有窗口大小为k的和。

字节三面

1. 自我介绍
2. 说下自己的优势
3. 未来想做什么
4. 介绍下简历上的项目
5. 介绍下paper的内容
6. 介绍下虚拟内存的概念
7. 虚拟内存的优势
8. 虚拟内存面对使用者有什么好处
9. 虚拟地址转换成物理地址对使用者有什么好处
0. 使用者直接操作物理地址会有什么问题
1. 进程的创建和销毁做了哪些工作
2. 杀掉一个进程怎么操作
3. kill 命令背后操作系统做了什么东西
4. 进程什么时候响应kill发送的信号
5. 两个栈实现队列的功能，口述，不用写代码
6. golang map是并发安全的吗
7. 怎么让map并发安全
8. sync.Map的优势
9. sync.Map读的时候怎么操作
0. sync.Map比普通map加锁性能上的优势
1. sync.Map的使用场景

编程题：

全排列

面经55-C++-字节前端暑期实习

字节一面

1. 自我介绍
2. 项目介绍
3. Web服务器里怎么设计的IO复用
4. 是否有做压测
5. 请求是同时请求还是分开的
6. 你觉得现在一般大厂使用的支持百万千万并发的服务器与你这个项目是一个概念吗
7. 有没有了解过Nginx，Nginx里使用的连接模式是怎样的
8. 项目里用了线程池，线程池怎么建立起来的，为什么要用线程池
9. 那你从请求队列里取请求那不还是一个连接对应一个线程吗?
0. 请求队列中请求的插入与取出的过程
1. 如果请求量非常大，那从请求队列里取对于后面的请求是不是会有延迟
   - 答得会有影响，但自己的项目里由于业务的读写内容不高所以不会有（给自己挖坑了，说自己的没有，面试官马上就开始设计场景了，无缝切换）
2. 那假设有这样一个场景，有着百万并发，请求各不相同，按照你的这样做法会不会存在延迟的问题
   - 答了使用更好的机器，提高线程数量（追问：线程数量应该如何设置，老八股答得很顺）
   - 提出了对于各种请求进行分类，对不同的请求执行不同的策略（具体分析了计算密集型和IO密集型请求的区别）
   - 追问：那一个IO密集型的请求，如果有缓存了，是不是就不需要IO读写了，那是不是就变成了计算密集型了，要如何去分类呢？答：不会哈哈哈哈哈哈（这个时候觉得一面比较稳了，因为面试官说了一句：等你进来教你）
3. 聊比赛，问最近一次的比赛，答得数学建模，然后面试官觉得这个项目不是很贴近软件，就又追问了其他比赛，聊了比较久一个人工智能的竞赛，问到了深度学习里用了什么网络模型，自己做了什么贡献、如何更新模型，怎么开展的，识别出错怎么办，如果遇到恶意攻击怎么办（说了一个自己的方案）
4. 业余爱好，怎么拿的奖
5. 强项、缺点（如何解决）
6. 聊一下C++里面的智能指针吧
7. 聊一下哈希表
8. 详细介绍一下unordered_map
9. 既然是用哈希函数，那么肯定有哈希冲突，遇到冲突怎么办，分析一下时间复杂度
0. 讲一下https和http的区别
1. https为什么是安全的
2. https中传输会经过网关，为什么网关不能知道他们的密钥
   - 答得非对称加密与对称加密相结合
   - 后来被面试官换了个问法又问了一遍，被问懵了，其实还是非对称加密，自己阵脚真的不能乱
3. 介绍一下深拷贝与浅拷贝
4. 如果想让你用深拷贝拷贝一个字典，你怎么做
5. 智力题：抛硬币，抛到正面就不抛了，抛到反面继续抛，问一百个人抛，最后正反面的比例
   - 答得1：1，不知道标答（追问有没有可能给出具体公式，答的条件概率，公式没写出来）

6. 以前面试过没有
7. 算法题：手写atoi函数

字节二面

1. 自我介绍
2. 聊竞赛
   - 把我所有的竞赛全部问了一遍，感觉是看竞赛中的参与程度
3. Web服务器为什么要去关闭长时间没有请求的连接
4. 客户端使用keep-alive不也可以控制连接时长吗
   - 答得针对自己业务，主要是静态资源访问，然后时间一般都不长，所以用服务器主动断开比较好
   - 然后如果客户端请求keep-alive的话服务器是支持长连接的
5. 断开连接这个时间是怎么定的
6. 高并发情况下，如果网络出现大面积波动，导致很多客户端下线了，但其实它们很快就会连接回来，断开连接就会很影响用户的体验，怎么解决?
7. 问了下硬件项目，面试官搞过嵌入式的，聊了一下硬件
8. 回归Web服务器项目，多进程、多线程的通信方式
9. 为什么要设计内核态、用户态两种状态
0. 说一下C++里面inline有什么用
1. CPU是怎么执行代码的
2. 对CPU流水线有没有了解（指出我前一个问题没有说流水线）
   - 答不了解
3. 算法题：链表带进位的求和，每个链表的节点只有一位数字，然后对所有位求和返回一个新链表
   - 面试官追问代码哪里可以优化，优化完，面试结束

字节三面

（没有自我介绍是我没想到的）

1. 会经常写代码吗，学校内的课程、竞赛、项目讲一讲
2. 熟悉汇编吗
3. 对于字符编码有哪些了解
4. unicode和utf-8的区别
   - “Unicode是“字符集”,UTF-8 是“编码规则”
5. 线程同步说一下
6. 为什么会有线程同步的出现，从底层汇编的角度说一下
7. 读写锁说一下，应用场景、原理

8. 说一下线程池，详细说说你认为构建一个线程池最重要的部分
9. 线程数量在运行中是一成不变的吗
10. 多线程是怎么处理许多个任务一起到来的
11. EPOLL跟你怎么处理多个任务有关系吗?
    - （这里是因为我上面啰嗦了，提了EPOLL，然后解释了一下，EPOLL不涉及任务处理，它做的是一个任务分发，使得一个线程可以处理分发多个请求）
12. 如果现在有远远多于线程的任务数量在等待处理你要怎么分析这个问题
    - 首先判断这个请求数量远远多于线程是常态吗，如果是常态则先要考虑是不是线程数量设置有问题，然后判断机器是否能力不足以处理这些数据
    - 假如说硬件软件都没有问题，那么考虑这个问题为什么会产生，是不是我们的高计算密集型任务太多了，如果说我们的请求任务比较复杂，那么考虑进行一个分类，将处理特别慢的请求作为一个特定类，将一些线程固定给他们使用，而不是所有线程都用来处理这种任务，因为既然很慢，那么慢1个和慢10个点区别就没有那么大了，将其他线程用来处理能够快速处理完的请求，这样子就保证了一个相对的处理效率
13. 说一下缺页中断
14. 说一下我们现在面试我的摄像头采集到的视频信息是如何发送到你屏幕上的
15. 说一说那我这边是怎么寻址到你的ip的，你现在看到的自己的ip是我发送时填入的ip吗
16. 了解NAT吗
17. udp不需要建立连接，那是不是所有人都可以向我发送数据
    - 答得：是的，虽然网卡能够接受这些数据包，但是会判断这个udp包的ip地址是否接受，然后端口号对应的服务是否开启，开启的话对应的应用层应用也会判断这个包是否是有效的，因此不会出现数据包错乱的情况
18. 对DNS劫持了解吗
19. 怎么规避呢?
    - 没打出来，就说了自动切换
    - 面试官说去了解一下HTTPDNS
20. 了解HTTP3吗，说一下HTTP1到HTTP3所有版本使用了哪些技术，相比上一代技术有哪些改进
21. 数据库的一组操作，如果中间崩溃了数据会丢失吗
22. 问了几个事务的场景：（答得不好，有数据库大佬可以说说自己的理解）
    - 执行的过程中，数据库服务器死机了会丢吗
    - 执行的过程中，数据库服务器断电了会丢吗，跟死机情况是否一样
    - 执行的过程中，数据库服务中间崩溃了，但很快重启了，重启前后查询的结果是否相同
23. 讲一讲C++里面的多态，怎么实现的
24. 讲一讲C++的new、delete和malloc、free的区别
25. 讲一讲C++中extern "C" 的作用
26. 了解C++中编译时的优化吗
    - 不了解……只说了O2级别优化很常用
27. 说一下`push_back()`和`emplace_back()`的区别
28. 算法题：一道DFS深搜加优化的题目
29. 闲聊
    - 印象最深的问题：如果我给你挂了，你接下来会去做什么

HR面

1. 经过前三轮面试，说一下你个人的感受
2. 对于这次面试的岗位，都做了哪些准备
3. 参与过ACM竞赛吗
4. 没参与过ACM竞赛的话，算法题方面是怎么准备的呢
   - 第一个答了本科阶段有数据结构和算法的基础
   - 然后老老实实说了针对题型在力扣上刷了一些题
5. 那在力扣上应该刷了很多题了？具体刷了多少道?
6. 是从什么时间开始准备这次面试的
7. 你觉得你在这几轮面试中你的优势是什么，劣势是什么
8. 面试完之后会复盘吗
9. 为什么选择字节跳动
0. 说一下职业规划，看到简历偏后端一些，是否会并不喜欢客户端的方向呢?
1. 个人的优势与短板
2. 你说擅长团队协作是因为自己带队作为负责人参与了很多比赛吗
3. 说一下你带团队时会考虑哪些问题，一个团队的关键点是什么
4. 大概什么时间能够到岗，实习多久
5. 对于转正你有什么问题吗
6. 对于工作的城市有要求吗
7. 家里对于工作地点上有意见吗
8. 进入到字节跳动团队后，你希望从你的mentor得到哪些帮助
9. 当你遇到问题你会主动找他人问吗
0. 有过压力很大时候吗?
1. 是怎样克服这个压力
2. 还有其他公司在面试吗
3. 对于这边的团队有了解吗
4. 反问:
   - 技术团队的人员组成是什么样子的
   - offer结果什么时候能收到

面经56-算法-百度pnc暑期实习

百度一面

基础问题：

1. dqn，ddpg，ppo算法优缺点，适用场景
2. 重要性采样的作用，实际中的计算方式
3. KL散度的作用，具体的计算方式（PPO）
4. python的生成器
5. 根据map的val从大到小排序（map默认按key从小到大排，不能用sort排），转成数据为pair的vector数组去排序，重构sort的cmp函数，返回l.second > r.second；
6. c++的智能指针
7. vector，list，queue适用哪种情况（基本数据结构和特点）
8. push*back*和*emplace*back的区别
9. 多态和虚函数的理解
0. Linux命令，要获得文档的某一行的某一个关键字--快捷命令

算法题：

63. 不同路径 II：一个机器人位于一个 m x n 网格的左上角 （起始点在下图中标记为“Start” ）。机器人每次只能向下或者向右移动一步。机器人试图达到网格的右下角（在下图中标记为“Finish”）。现在考虑网格中有障碍物。那么从左上角到右下角将会有多少条不同的路径——动态规划。

百度二面

基础问题：

1. 其他的多智能体RL：MADDPG，VDN，QMIX，COMA，MAPPO简要描述，QMIX和MADDPG区别（对抗，合作）
2. 游戏方面的MARL——星际争霸的了解
3. 博弈理论

算法题：

二维网格图中探测环：dfs

百度三面

基本问题：

1. SAC，RL算法的大概体系（基于值，基于策略，AC）
2. Linux和Ubuntu系统，苹果系统的基础介绍
3. python中效率高的代码写法
4. numpy数据处理的基本用法
5. 单元测试
6. 你到目前为止你认为自己说是期间遇到的最难的事情是什么呢？然后你是怎么怎么处理这个事情？
7. 硕士期间最难学的课，用的什么方法，怎么学的

8. 到现在为止你这个最有成就感的事情
9. 科研和项目是怎么学习新技术的
0. 读过的经典书目

面经57-C++-百度

> @草原上沐风

投递的一般都是C++开发相关岗位，两个项目一个是Web服务器，另一个是卡哥的跳表数据库。

百度一面

1、个人介绍

2、两个项目的背景

3、两个项目都是基于个人兴趣做的吗

4、和导师有做开发项目吗

5、单例模式实现的优点

6、懒汉式局部静态变量提前销毁有什么办法（销毁后不会再创建了）

7、了解C++14，17特性吗

8、介绍redis的几种数据类型和底层数据结构

9、redis除了五大数据类型，其他数据类型有了解吗

10、HTTP访问流程

11、HTTPs建立连接的过程

12、int* 和const不同位置代表什么意思

13、C++11左值右值和移动语义

14、手撕算法：使用模板的归并排序（不使用额外空间，用swap函数）

百度二面

1、自我介绍

2、web服务器项目介绍

3、创建线程用什么api

4、如何销毁线程

5、detach和join有什么区别

6、这个项目遇到什么困难

7、timewait状态原因

8、closewait状态原因

9、很多closewait状态是什么原因

10、客户端close了服务端发送数据会发生什么

12、介绍跳表数据库项目

13、如何保证线程安全

14、互斥锁粒度大怎么解决

15、介绍有哪些锁

16、redis的数据结构

17、数据库如何存盘

18、存入其他类型数据如何读取出来（正常读出来都是string）

19、gdb如何调试

20、gdb运行报错如何通过core文件找到错误

21、如何加断点，具体api调用

22、如何看函数调用栈

23、epoll函数参数

24、ET和LT模式是什么

25、select和epoll区别

26、类中static作用

27、虚函数和纯虚函数的区别

28、了解哪些C++11特性

29、weakPtr如何获得sharedPtr

30、weakPtr的作用

31、lambda函数的返回值

32、右值引用的用途

33、C++11创建线程方法

34、C++11互斥锁

35、反问

百度三面

1、自我介绍

2、介绍一下你的项目

3、研究生的课题是哪方面的

4、你为什么选择C++语言

5、有对比过其他语言吗

6、分布式，高并发有了解吗

7、redis做缓存有哪些问题，如何解决？（从缓存穿透，缓存击穿，缓存雪崩，缓存一致性等方面来答）

8、常用的redis命令有哪些

9、设计一个秒杀系统，思路是怎样的

10、sql语法：一个表，有id,name，如何查找某一个名字有几个重名（他口述的说不清楚，大概是这意思）

11、几亿条数据，查找topk（堆，快排）

12、linux常用的命令

13、一个文件，有几千万行，有IP地址、访问时间、url，访问topk出现频率的IP或url的命令是什么样的

14、AWS用过吗

15、其他的中间件，框架用过哪些

16、在学校里除了课程哪些方面投入比较多

17、你有哪些优点和不足

18、团队合作的经历

19、合作上遇到什么问题

20、学校期间比较有成就感的事情

21、你后续的计划大致是哪些

22、你个人有哪些兴趣爱好

23、反问

> @草原上沐风

ZEKU哲库一面

> 30 min

1、自我介绍

2、算法：912. 排序数组。（对，没看错就是没任何套路的排序，我写了个快排）

3、说一下快排的思路

4、说一下其他时间复杂度为nlogn的排序算法

5、链表和哈希表的区别

6、链表和队列的区别

7、进程是什么

8、进程调度算法

9、进程上下文切换的内容

10、说一下C++的多态

11、函数重载底层原理

12、C++11有哪些特性

13、说一下lambda函数

14、跳表是什么

15、跳表增加和删除应该怎么操作

16、说一下硕士期间的论文

17、说一下电赛做了什么

18、反问

ZEKU哲库二面

> 40 min

1、自我介绍

2、介绍一个熟悉的项目（我介绍的web服务器）

3、为什么做这个项目

4、介绍一下动态线程池运行逻辑

5、为什么要采用动态线程池

6、动态和静态线程池的性能差异是多少

7、线程池线程数量如何设置的

8、项目有没有其他难点（说了下小顶堆管理定时器）

9、添加定时器是为了解决什么问题

10、关于项目还有没有其他要介绍的点（又说了epoll和select，poll的区别）

11、线程和进程的区别

12、数组和链表的区别

13、项目的异常处理有哪些（说了下SIGPIPE和ET模式下读数据时的EAGAIN）

14、介绍一下跳表数据库项目

15、说一下数据库的存盘功能（顺便说了下redis的持久化）

16、介绍一下实验室的科研情况（科研做的集成光学方面的仿真）

17、仿真是怎么做的

18、你的专业学过哪些课

19、反问

ZEKU哲库HR面

> 20 min

1、自我介绍

2、为什么本科从南开考到天大

3、研究生期间压力最大的时候是什么时候

4、同级的小导师招了几个人？（不懂为啥问这问题）

5、为什么科研方向不好就业

6、为什么当初选这个科研方向

7、科研用什么编程语言

8、你最擅长的编程语言是什么语言

9、自学编程的动力是什么

10、为什么做这两个C++项目

11、实验室的工作时间要求

12、你找工作最看重哪些因素

13、除了ZEKU你还申请了哪些公司呢

14、从现在到毕业前你的时间如何安排

15、反问

面经59-C++-VIVO

> @草原上沐风

VIVO 一面

1、自我介绍

2、两个项目是自己做的还是公司或实验室项目

3、研究生期间的项目是做什么的

4、算法：给两个数，计算它们二进制位中不同的个数

5、MySql数据库查找的背后算法逻辑（B+树）

6、你做的数据库最多存储数据多少

7、MySql性能优化怎么做（建立索引，覆盖查询，避免索引失效）

8、linux命令了解哪些

9、视频会议用的什么协议（不会）

10、VPN在哪一层网络模型实现的

11、反问

VIVO HR面

1、自我介绍

2、硕士期间主要做哪些方面的研究

3、选择一个项目详细介绍

4、做这个项目的背景

5、为什么想做软件开发

6、你的项目是团队合作的项目吗

7、合作时与其他人想法不一样会怎么做

8、你坚持自己的想法的时候多还是参考其他人意见的时候多

9、科研项目的创新点是什么

10、整个项目中你承担了哪些责任

11、什么样的人是你合作人不太想遇到的

12、和其他人合作是你更关注他的性格还是能力

13、为什么不找本专业的工作

14、工作地点有哪些考虑

15、有哪些在面试的企业

16、反问

面经60-C++-TPLINK

> @泡面盖子

一面 17min

1. 自我介绍 考研还是保研
2. 王者荣耀是Tcp还是Udp链接?
3. 如何判断tcp还是udp呢?
4. C++的struct
5. 代码编译过程（四大步）
6. 池化层作用
7. 介绍项目
8. 了解的测试工具介绍（没答上来）
9. struct是什么意思?

二面 30min

1. 自我介绍
2. 项目介绍
3. yolov5介绍
4. 为什么图像会使用卷积?
5. 数据集选择 考虑多方面因素没有?
6. 对腾讯会议进行测试
7. 智力题 三个盒子
8. 反问环节

三面

1. 自我介绍
2. 自我介绍相关提问
3. 兴趣爱好 爱好相关提问
4. 考研保研 为什么选择跨学院
5. 为什么选择测试?
6. 深度学习理解
7. 测试兴趣在哪里
8. 反问环节

面经61-测试-海尔智家

> @泡面盖子

一面

1. 自我介绍
2. 对测试工程师的理解
3. 铅笔测试
4. 介绍项目

二面 : # (这次面试的小姐姐特别温柔！会引导我的回答，会对回答的内容进行补充和落实)

1. 介绍项目
2. 为什么选择测试
3. 对测试流程的理解
4. 测试用例设计方法
5. 自我职业规划
6. 反问环节

hr面试+交流 25min

1. 年龄
2. 考研保研?
3. 之后打算（深造打算
4. 工作地点选择
5. 为什么不选择北京编程语言
6. 项目介绍
7. 承担职责
8. 导师
9. 爸妈家里情况
0. 期待年薪
1. 群?

面经62-C++-蔚来

> @泡面盖子

蔚来一面 1h10min

1. 自我介绍
2. 面向对象 四大特征
3. 封装的概念
4. 单例模式
5. 多线程 进程线程区别
6. 线程安全
7. 如何保证线程安全
8. 父类只想被子类调用
9. cookie和session的区别
0. 接口请求
1. get和post的区别
2. mysql语句
3. 测试用例
4. 登陆界面测试
5. 接口测试

面经63-测开-字节

> @泡面盖子

一面 37min

1.自我介绍

2.进程线程区别

3.线程池

4.多线程

5.索引

6.复合索引

7.java buffer和builder

8.深拷贝浅拷贝

9.反转字符串

10.测试用例 红包 算法

11.代码

二面 50min（面试官找不到我相关的项目，直接没兴趣..扭头就挂啦！秋招首挂）

1. 项目
2. Java C+ Python理解
3. 软件工程
4. 测开的理解
5. 微服务架构项目 查询 测试用例
6. 压力测试 并发测试

字节二面 1h （其他部门）

1. 自我介绍
2. 项目介绍
3. 数据结构 如何构建队列?
4. 用数组实现队列和用链表实现队列
5. 用通俗的语言解释死锁（我说你有香蕉，我有苹果，我说你给我香蕉我再给你苹果，你说“你给我苹果我就给你香蕉” 这就是死锁，把面试官逗乐了...）
6. 手撕 实现split，一部分回文我没有处理好

面经64-测试-中望

> @泡面盖子

一面 35min

1. 自我介绍
2. 学习测试大概来源
3. 为什么对测试有兴趣
4. 介绍介绍测试
5. 冒烟测试
6. alpha 和 beta测试
7. 单例设计模式
8. 内存泄露
9. 智能指针 内存泄露
0. 反问

二面 39min

1. 自我介绍
2. 为什么选择测开
3. 外向的例子 社团活动
4. 更偏向哪方面
5. 广州
6. 未来计划
7. 定居点
8. 反问

面经65-测开-ZEKU

> @泡面盖子

一面 35min

1. 自我介绍
2. 项目难点 如何解决
3. 组织管理者
4. 死锁

5. 内存管理区域
6. 其他方面了解
7. 为什么选择测开
8. 反问

二面 45min

1. 自我介绍
2. 为什么选择测开
3. 特质
4. 沟通
5. 不想合作的人会怎么办
6. 怎么解决的
7. 抗压能力
8. 反问

面经66-格力

> @泡面盖子

一面 30min

1. 自我介绍
2. 项目介绍
3. 数据分析
4. 项目难点
5. 反问

二面 20min

1. 自我介绍
2. 项目介绍
3. 工作状况和态度你最大的优点
4. 目标薪资
5. 挂科情况
6. 英语成绩

面经67-测试-科大讯飞

> @泡面盖子

一面 40min

1. 自我介绍
2. 最近在学习的东西
3. 网络http报文头
4. get和post区别
5. 测试流程

6. 解决难题
7. 测试应用到项目里

面经68-测试-大疆

> @泡面盖子

8.25 12:00 一面

1. 自我介绍
2. 自己情况的讯问
3. 单元测试工具
4. 单元测试指标
5. 测试用例要素
6. 测试用例指标
7. 项目介绍
8. 红外图像了解程度
9. 反问
0. 工作岗位接受调动吗
1. 智力题：小球碰撞 一个十米长的光滑轨道，两边是墙，在位置第1 3 5 7 9米的地方，分别有5个小球，它们的速度都是朝右且值为13579，问当前在第5米的小球，第11s的时候在哪?

面经69-C++-华为

> @向左

一面（2022.8.19 19.00-19.40）

1. 自我介绍
2. 引用和指针
3. 多态
4. new和malloc区别，malloc如何分配内存
5. 堆和栈的区别
6. 进程和线程区别
7. 进程间通信
8. 为什么进程切换开销更大
9. 网络模型
0. http属于哪一层
1. tcp和udp区别
2. tcp三次握手和四次挥手
3. 为什么是三次?
4. 做题，用dfs解决，具体题目就不说了
5. 介绍本科和研究生获得的竞赛奖励
6. 介绍你在这些项目中做的事情
7. 你从实习中学习到了什么?
8. 你开发服务端有什么收获?
9. poll、epoll和select的区别?

10. 消息队列的作用

五分钟后通过（周五晚上面的，所以二面就到下周一了）

二面（2022.8.22 15.40-16.40）

1. 自我介绍
2. 问参加了什么比赛
3. 数学建模美赛做了什么?
4. 特征如何选择的?
5. 使用了什么算法?
6. 数据来源?
7. 问项目
8. 你负责什么部分?
9. 对程序有什么改进?
10. 除了协议和开源框架以外有什么改进?
11. 重构了什么?
12. 重构之后是否有性能提升?
13. 你是如何接手这些项目的?
14. 如何进行学习的?
15. 程序不同部分的同学之间数据格式如何协商?
16. 项目中使用过什么设计模式?
17. 介绍一下观察者模式?
18. 项目中单例模式使用场景?
19. 线程池是怎么设计的?
20. 多线程冲突如何解决?
21. 线程之间是否有依赖?
22. c++无锁队列？知道概念但项目中没有使用过
23. 是否对程序进行了性能检测和内存泄漏检测？如何进行的?
24. 手撕：快速排序
25. 反问

五分钟后通过

主管面（2022.8.29 15.10-15.30）

（本来是明天的，面试官有事调到今天下午了）

1. 自我介绍
2. 性格评测（因为之前实习这个挂了，前两天重测了一次才过）
3. 父母是干什么的？家庭情况
4. 介绍实习
5. 对华为有什么了解?
6. 为什么想来华为?
7. 为什么不在本校读研?
8. 喜欢编程吗?
9. 为什么参加这些比赛?
10. 平常喜欢打游戏吗?

1. 项目中你主要负责什么?
2. 你有低谷期吗?如何排解压力?
3. 目前面了哪些公司?情况如何?
4. 反问

半小时左右收到通过短信

面经70-C++-中望

> @向左

一面 2022.6.23 14:30-15:00

（有俩面试官，不过只有一个在问问题）

1. 自我介绍
2. 多态
3. 虚函数
4. 虚函数指针的位置（开始答的是虚函数表在全局静态区，后来面试官解释了一下才知道他想问的是虚函数指针在类中的什么地方）
5. 如何定位bug（log日志加debug调试）
6. 平常用Python干什么?
7. Python2和Python3区别
8. Python基本数据类型
9. dict和list访问速度
0. list和tuple区别
1. 用过stl的什么容器?
2. map的底层?
3. 红黑树的性质?
4. 介绍排序算法（介绍了插入排序、快速排序、堆排序）
5. tcp和udp区别
6. 五层网络模型介绍
7. Socket介绍
8. 介绍项目
9. 守护进程实现
0. 介绍mqtt
1. 多线程同步
2. 项目的难点，有什么可以改进的地方?
3. 平常写博客或者一些小玩意儿吗?
4. 反问

二面 2022.7.1 10:30-11:00

（开始一个面试官，后来又来了一个）

1. 自我介绍
2. 介绍c++项目，包括类的实现
3. 如何用c++进行深度学习模型部署

4. 进程间通信如何实现?
5. Static
6. Const
7. 堆和栈区别
8. new/malloc delete/free区别
9. 用过哪些stl容器?
0. 用过哪些stl算法? sort
1. 讲一讲你了解的排序算法 快排、堆排、插入排序
2. 如何判断一个数是否是2的n次方 n&(n-1)
3. 如何判断点p是否在三角形内部? 海伦-秦九韶公式
4. 笔试题有印象深的吗?
5. 你觉得你笔试多少分?
6. 你对cad有多少了解?
7. 你觉得开发cad需要什么知识? c++、qt、opengl、mfc
8. 然后面试官1开始讲他们公司业务...
9. 你的职业规划?
0. 此时另一个面试官进入了房间.....

面试官2:

1. 你为什么想加入我们公司?
2. cad就业面窄，你对此有何看法?
3. 你是哪里人?
4. 反问

HR面（电话面） 2022.7.6 17:25-17:30

1. 通知我技术面和主管面的面试通过
2. 询问意向工作部门
3. 询问期望薪资
4. 询问未来打算
5. 询问为什么选择中望
6. 询问家庭情况
7. 然后结束了，发了个评测

面经71-C++-米哈游

> @乌鸦坐PCJ600

一面

1. define 和 const 的区别
2. 移动语义有什么作用，原理是什么
3. 引用和指针的区别
4. 常量指针和指针常量有什么区别
5. vector push_back 的时间复杂度是什么，什么情况下会发生扩容，扩容如何实现
6. 线程和协程有什么区别，各自的优越性是什么
7. 进程之间如何进行通信

8. 什么是信号，信号是如何实现的
9. 三次握手中每一次的目的是什么
0. 四次挥手为什么是四次
1. 什么是粘包和拆包，为什么会出现，如何解决
2. 数据库：什么是关系型数据库
3. 什么是事务
4. select a, b, c, d from t where a = 1 and b = 2 order by c desc 如何建立索引，为什么
5. 反转链表：要求使用递归，不能用迭代
6. 实现 LRU 类

面经72-C++-旷世

> @乌鸦坐PCJ600

一面

1. C++ 的多态如何实现
2. 简单说说智能指针
3. vector 和自建数组有什么区别，vector 如何扩容
4. 堆和栈有什么区别
5. 进程和线程有什么区别
6. 进程之间如何通信
7. 线程之间如何实现同步
8. TCP 和 UDP 有什么区别
9. TCP 的流量控制是如何实现的
0. 描述向跳表中插入新kv对时的的具体操作过程，时间复杂度是多少
1. 如何在跳表中查找 topK 元素，可以修改跳表中维护的变量
2. 手撕：
3. 二叉树左叶子节点元素和
4. 有序数组中删除重复元素

二面

1. lambda 表达式、std::function、函数指针这三者有什么区别
2. 如果一个 lambda 表达式作为参数传递给一个函数，那这个函数可以使用这个 lambda 表达式捕获的变量吗
3. 手撕：LC.735

面经73-C++-超参数

> @乌鸦坐PCJ600

一面

1. 自我介绍
2. 实习经历，问的比较细，大概 20 分钟
3. 开放题：如果你设计一个后台，会做哪些反爬措施
4. 在 showmebug 上写算法题：给一个列表，每个元素代表平面上点的座标（[(0, 0), (1, 2),.....]），求能组成的矩形

的个数。
5. 写完以后让估算一下时间复杂度，没有让继续优化。
6. 如果服务端出现大量 time_wait，可能的原因是什么，会造成什么后果。

二面

1. 自我介绍
2. 实习经历，大概10分钟
3. 问了一下学过哪些课，数据结构、计算机网络、操作系统，听到没有学过计算机组成说建议学一下
4. 关于多态
5. C++ 的多态是怎么实现的
6. 虚函数是怎么实现的
7. 虚函数表的本质是什么
8. 如何在 C 语言中实现多态（懵了）
9. 进程在内存中的分布是什么样的（应该是在问堆、栈这些，没说全）
0. 最近在看什么书？我说在看 MySQL和Redis，面试官说数据库没有计算机组成重要
1. 一面的算法题做出来了吗？（没做出来我能进二面吗？）
2. 家是哪里的，有没有女朋友

面经74-C++-蔚来

> @乌鸦坐PCJ600

一面

1. 自我介绍
2. 语言？面试官是 Java，这个岗位也只有是 Java，我 C++ 有点尴尬
3. MySQL 中 innoDB 的索引为什么使用 B+ 树
4. 聚簇索引和非聚簇索引的区别是什么？
5. ACID 是什么？
6. MVCC 的实现方式是什么？
7. 悲观锁和乐观锁的区别？
8. 有哪些锁？什么情况下会触发加锁，锁和索引有什么关系？
9. 用过 Redis 吗？Redis 有哪些应用场景，了解 Redis 的定时删除机制吗？
0. 了解函数式编程吗？
1. 浏览器输入一条 url 后的过程
2. 开放一个公网接口，需要考虑哪些问题？
3. 算法题出了道 easy，判断合法的括号。
4. 做完以后面试官说一面过了，但是二面还会问 Java，让我自求多福。

二面

1. 自我介绍
2. 讲之前实习内容
3. 问了下 mongo，可惜不会
4. 数组和链表的区别
5. MySQL 八股
6. 有哪些类型的索引
7. 建立索引有哪些原则

8. 每种隔离级别会遇到哪些冲突
9. SSL 握手的全流程，对称加密?
0. 算法题：二分稍微改了下
1. 找工作会考虑哪些因素

面经75-C++-第四范式

> @乌鸦坐PCJ600

一面

1. 自我介绍
2. 会哪些语言?
3. 写过 Go 的话，说说 channel 的底层数据结构。
4. HTTP2.0 相比 HTTP1.1 有哪些改进，1.1 相比 1.0有哪些改进。
5. 知道跨域吗？跨域发生在前端还是后端?
6. docker 和 k8s 会不会?
7. 口述拓扑排序的思路
8. 口述如何判断一个有向图是否有环（直接用topsort的思路）
9. 口述如何计算一个二叉树的最大深度（DFS或者BFS）
0. 口述层序遍历。
1. - LRU 全称是什么？LFU 呢？（LRU 答出来了，LFU 没看过）
   - 手撕 LRU，自己写主函数测试一下
2. LRU 的线程安全如何保证?
3. 了解机器学习和深度学习的基础知识吗?
4. 一个最常见的卷积层有哪些参数?
5. 卷积的物理意义是什么?

反问

- 技术栈是什么？面试官大概讲了5分钟团队结构和业务方向，后端主 Java，辅 Go 和 Python

- Base 哪里，总部在北京，上海也有

二面

先问了实习经历

一道八股都没有，都是场景设计题，因为我没做过支持高并发的后端系统，所以答得很差

过了两周通知进三面了，还以为二面稀烂会过不了

三面

没有问八股和算法，项目是C++的所以也没问，问了问我 Github 上的两个十几个星的 Python 项目：

- 为什么做这个项目
- 遇到哪些困难
- 最大的收获是什么
- 还有哪些可以优化的地方

最后介绍了业务和技术栈，问了一下对工作地点的要求，总共35分钟，聊天为主。

面经76-C++-TP-Link

> @乌鸦坐PCJ600

一面

1. 自我介绍
2. 保研的还是考研的，本科排名多少。
3. 实验室方向，有没有论文。
4. 之前实习具体是在做什么，你做了哪些工作。
5. C++的多态体现在哪里，如何实现的。
6. 讲讲智能指针的作用和原理，不用每种展开讲。
7. IO多路复用里，三种技术的区别

二面

1. 自我介绍
2. 实习经历，主要做的工作（大概说了10分钟，比第一面问的多）
3. 又问了一遍 C++ 多态的实现原理
4. static 关键字有哪些作用
5. 如何检测内存泄漏
6. 如果让你实现一个检测内存泄漏的工具，你会怎么做?
7. 进程间如何通信?
8. TCP 四次挥手的过程，为什么需要 2MSL?
9. 两数之和，共享屏幕在本地 IDE 去做，秒了

三面

面试官看起来年龄比较大，应该是大部门leader级别了

1. 自我介绍
2. 聊研究生期间的经历，换过两次方向，面试官说你这经历还挺曲折
3. 看你实验室主要在做算法和硬件，为什么想转软件开发，你怎么自学的，学了哪些?
4. 问对云计算有什么了解？只知道大概分成公有云，私有云

面试官讲了接近10分钟，具体业务、和阿里云/腾讯云的区别等等

5. 有什么兴趣爱好

面经77-C++-快手

> @乌鸦坐PCJ600

一面

1. 什么是死锁，死锁的条件，如何避免死锁
2. 智能指针有哪几种，weak_ptr 的作用是什么
3. 如何确定代码中的类循环引用
4. 手撕：
5. 最长回文子串，分析一下时间复杂度
6. 股票最佳售卖时间，最基础的那个版本

二面

1. TCP 和 UDP 有什么区别
2. 操作系统如何进行进程的上下文切换
3. 给了一个很复杂的代码，说出返回值类型，函数指针
4. C++ 从代码到可执行文件的生成顺序
5. C++ 手写一个生产者消费者模型

面经78-C++-猿辅导

> @乌鸦坐PCJ600

一面

1. 介绍之前参加的比赛和使用的方法
2. C++ 里的自增运算符是线程安全的吗
3. HTTPS 握手的全过程
4. HTTPS 如何防止中间人攻击
5. 常见状态码和对应的意义
6. epoll 的两种模式 LT 和 ET 的区别
7. 写一道 SQL，考察子查询和 group by，order by，limit
8. 索引为什么用 B+ 树，不用 B 树
9. MySQL 中如果修改了一个 4kb 的页的内容，存入磁盘时修改的是完整的页还是页中的一段
0. 手撕：
1. 第 K 大
2. 层序遍历

二面

1. 说一下三次握手、四次握手全过程
2. 四次挥手能不能合并第2、3次？为什么
3. 服务端出现大量 close_wait 的原因可能是什么？
4. MySQL 索引是什么结构，一个 B+ 树有多高，估算一下两层的 B+ 树可以存放多少数据，三层呢？
5. MySQL 有哪些锁，什么是 Gap-Lock
6. 给了一个 MySQL 语句分析加了什么锁
7. 手撕：各种旋转数组
8. LC.153，然后口述如果数组元素可以重复怎么修改，其实就是 LC.154
9. LC.33

面经79-C++-元戎启行

> @乌鸦坐PCJ600

一面

1. 自我介绍、简单讲了讲实习经历
2. 说一下三次握手、四次挥手的全过程
3. 为什么是3次握手，2次行不行？
4. 如果出现大量 close_wait，可能的原因是什么？
5. 说一下 HTTPS 和 HTTP 的区别
6. 说一下 SSL 握手的全过程
7. HTTPS 通信时是对称加密还是非对称加密？
8. 说一下死锁的产生条件和避免方法
9. 说一下粘包可能的产生原因和解决方法
10. 说一下缺页中断的解决方法
11. 为什么需要 I/O 多路复用？
12. 说一下数据库事务的四种隔离级别
13. 会 MySQL 调优吗？答曰不会，面试官说那 explain 总用过吧
14. MySQL 联合索引，给了一个 SQL 问走不走索引
15. 有一个人口普查表，要不要对性别这一列建立索引
16. 有没有做过分布式系统？
17. 布隆过滤器用过吗？
18. Redis 用过吗？
19. 如何实现一个下载器中的进度条？
20. 给了一段代码，具体忘了，主要是考察 C++ 拷贝构造函数
21. 为什么栈分配内存比堆快？
22. 什么是多态，如何实现？
23. 虚函数表是什么，虚函数指针是什么？
24. 析构函数为什么要定义成虚函数？
25. Go 的垃圾回收原理和 GMP 模型，选一个讲讲
26. 你的项目里用了 MongoDB，为什么不用 Redis？
27. 面试官：我上周遇到一个 MongoDB 的问题，blabla，你会不会？我：不会，你后来解决了吗？面试官：没有，算了。
28. 算法题：股票，测试用例给我挖了个坑，用 int 会爆，讨论了一下用 long 还是 long long
29. 反问环节：
30. 部门情况？面试官说他是算法的，不太清楚软件开发部门。
31. 一共几面？面试官说他当年是5面。可能这就是特别优秀的大佬吧。

二面

1. 自我介绍
2. 简单介绍项目
3. 网络方面遇到过发送请求没有收到响应的情况吗？怎么解决的？
4. 遇到过死锁吗？怎么解决的，不要背那四条。
5. 线程和进程有什么区别？什么时候用多线程，什么时候用多进程？
6. 线程崩溃进程会崩溃吗？
7. .cpp -> 可执行文件的过程
8. 了解链接的详细过程吗？
9. vector size/capacity 的区别，如何缩容？

0. 知道哪些键值型存储数据结构?
1. 这些数据结构的时间、空间复杂度分别是什么?什么时候选用?
2. 算法题:如何深拷贝一张连通无向图,showmebug 上写代码,不用测试

面经80-C++-文远知行

> @乌鸦坐PCJ600

一面

1. 快排的平均时间复杂度,最坏时间复杂度,什么时候最坏,如何防止塌缩到最坏情况
2. 看过 STL 里 sort 的实现吗?它是怎么避免的
3. 知道哪些数据结构?AVL 树如何保证平衡
4. 数据库存储为什么用 B+ 树,不用 AVL 树。
5. B+ 叶子结点是连续存储的,读取的时间更短;遍历的时候磁盘 I/O 的时间远大于算法的时间
6. 手撕:
   - 求数组第 K 小的数。partition 的平均/最坏时间复杂度是什么,为什么?
   - 口算 (9^999999)%5=?
   - 求(a^b)%n。本质是快速幂,只写出递归。a、b、n都很大,int会爆,必须用long

二面

1. 数据库:
   - 让你设计一张表,你会怎么设计
   - 为什么要建立索引,索引为什么能够加快查询的速度
   - 什么情况下不要建立索引比较好
   - 数据库的三大范式是什么
2. 知道哪些锁,这些锁的区别是什么?
3. 两道手撕:
   - 给定一个n个数的数组,求两个不相交的区间,使得区间里的数和最大,输出该最大的数。
   - 给了一个实际情境,本质是需要自己建一个有向图,然后求其中的最长路径

面经81-java-用友

> @椰子树小岛

一面(50分钟左右)

感觉用友的面试官问的比较浅,每个知识点都是点到为止,不会深入,主要是问一些项目细节和八股文,更多的是在考察基础。

1. 京东实习时长,介绍项目主要负责部分
2. 实习项目的多线程的访问压力有多大?
3. 实习项目的用户数据怎么存储的,存储在哪里,用的什么数据结构?索引的设计思想

4. 四种引用，强引用、软引用、弱引用、虚引用分别介绍
5. String、Stringbuilder、Stringbuffer的区别，String s = "abc"和String s = new String("abd")一样吗
6. 序列化时，某些字段不想被序列化，怎么做？用什么注解
7. 线程生命周期
8. 创建线程的方式
9. 常用的线程池有哪些，实习项目用的哪个？
0. Sleep和Wait的区别
1. Java中加锁的方式有哪些？可重入锁是什么意思？谁是可重入锁？
2. 介绍JVM内存结构，哪些是线程私有的？哪些会发生OOM？
3. Java类加载过程介绍
4. 双亲委派机制是什么?tomcat中的是这样的吗?
5. 如何判断垃圾是否该被回收?介绍常用的垃圾回收算法
6. 什么时候触发完全回收full GC
7. Spring启动过程、分哪些步骤
8. Spring @Autowired和@Resource有什么区别?
9. Spring bean作用域介绍
0. AOP简单介绍，AOP的几个主要核心概念是什么？AOP中一些常用的注解的作用
1. Redis缓存雪崩和缓存穿透，介绍，如何解决这些问题?

二面（40分钟）

二面更多的是深入的聊，还有一些场景设计，考察的更深入也更广泛。有一些很深入的说不上来也不要紧张，但是需要有自己的思考，尽量往自己会的方向扯，尽可能多的表达自己的思路和会的知识点。

项目:

1. 项目是做数据处理还是后端查询接口?
2. 自测用什么工具，自测看什么指标?
3. 多线程是怎么用的？Tomcat本来是有线程池的，可以接受多个请求，那你的多线程是什么作用?
4. 介绍中国卫星通信的项目,属于前端的内容多一些，他说他们某些项目组的后端也需要会前端

Java基础:

1. 怎么写一个懒加载的单例，说一下怎么写的呢？volatile的可见性是具体怎么实现的?
2. Hashmap的线程问题，如何解决的
3. 大并发量的情况下，冲突大的情况下（上万的情况下），使用sycronized还是用CAS呢?
4. 雪花算法用过吗?
5. Mysql怎么实现的可重复读，底层实现
6. MVCC实现原理，Mvcc解决了什么问题，隐藏列包括什么内容
7. 消息队列等幂了解吗?
8. 消息队列交替打印123456如何实现，有多少种实现思路?
9. 分享对JVM的理解

反问:

业务是企业管理，大部门是做人力管理

针对岗位，要求不一样，对spring要求高，中间件最好有一些要求，微服务，分布式技术

三面（25分钟）

三面就是HR面，我当时感觉HR很严肃，问的问题也很尖锐，中间都不停的。每个HR的风格都不太一样，有的是温柔的聊天，有的是快速的问答，但是他们问题背后的动机是一样的，就是看看你是否稳定？能长期留在北京？是否独生子？是否抗压，有过一些大压力下仍然能高效工作的经历，性格是否极端，产生冲突的时候如何解决？是否善于沟通，与其余的技术人员之间的交流是否顺畅，学历是否保真，是否挂科？成绩如何？还有就是通过你手里现有的Offer来判断你这个人的价值和能力

1. 学历、是否挂科、是否奖学金
2. 家庭、家庭印象最深刻的事情
3. 遇到过大的冲突没有？如何解决的
4. 朋友怎么形容你？你怎么形容自己？
5. 最早什么时间可以实习？
6. 有没有其余的Offer

面经82-测开-快手

> @椰子树小岛

一面

1. 求职意向：为什么从后端转为测开？

我觉得整个秋招就是在找准自己定位的过程，我做测开的原因就是因为我了解到测试需要细心一些，既能考虑全局又能深入发现细节问题，我本身就是比较能考虑细节的人，并且我有一些开发的经验，在测试之余能够发挥开发的优势，开发一些测试工具使得测试效率更高。

2. 研究方向
3. 实习经历
4. 项目难点
5. 实习各方面的积累、技巧
6. 对测试的认识、测试的思维、理解需求、擅长沟通
7. 性能测试
8. 实习的阶段写过多少代码？
9. 接口和抽象类的区别

本质的区别：接口里面是不能有任何的实现类的，但是抽象类里面是可以有一些实现的，所以接口是更抽象的，抽象类只是绝对的抽象。

0. static关键字的用法，static在内存模型中的位置、static修饰类的时候放在哪？
1. tcp、udp的区别
2. tcp有哪些机制保证可靠
3. 什么场景用tcp，什么场景用udp
4. 网络层次
5. 单链表反转
6. 字符串前排列

反问：

1. 测开要求的开发能力和技术
2. 简历不要写后端开发
3. 为什么做测开？测开的内容、自己为什么适合

4. 业务：快手客户端的测试

测试开发主要做的是测试，开发只是起到辅助性的作用，测试方面的要求更高一些，比如沟通、理解、逻辑

5. 基础知识、算法、网络这些基础看重，网上的测试太书面了，不用太看，公司也不看重这些，关于测试可以多和之前测试的同事聊，聊聊经验比较好。

二面

1. 信息安全的为什么选择互联网
2. 为什么选择测开?
3. 京东开发的工具、架构
4. 整个项目中的难点
5. 项目中的最安全的什么，数据的敏感性、准确性是怎么做到的，和数据进行联调的时候需要考虑的
6. 研究方向的安全是指具体什么?
7. 做开发的时候如何与测试进行配合的?
8. Java的三大特性，用自己的理解来说
9. Java产生死锁的条件是什么？怎么避免死锁
0. SQL和Redis的区别，使用场景
1. 如果让你测这两个，怎么测?
2. 二叉树展开为链表
3. 缺失的两个数字

反问:

测试的知识怎么补?

互联网的测试都是，敏捷开发的流程，所以可以多了解一下

另外就是方向，客户端还是后端，安卓的，

后端的Java，IDE，linux的日志查看、SQL、压测、抓包工具、postman

压测的原理和方法，性能。

三面

1. 自我介绍
2. 三维视频融合项目、如何做的、如何实现的、这个领域做的比较好的公司、遇到问题只能找底层问题吗，有没有做的更好的公司？为什么要做室内，什么场景才比较适用?
3. 场景题（微博请求错误，抛异常之后，让他自动再次请求，会产生什么问题，四个操作，点赞、评论，看评论列表、看热点话题）
4. 三面就是通过之前的项目中的问题和表达，看看你这个人的做事思路，使用场景题，看看你能不能将所学的知识用到这些从未遇到过的问题上面。另外就是一直不笑，属于一点压力面，给你一种比较紧张的范围。
5. 是否长久留在北京、是否选择互联网
6. 两周给结果，前两面+第三面都比较A的话，会比较早收到，否则的话，前两面都不太好的话，会后期进行综合比较之后再给结果
7. 面试反馈、有没有不满意的地方

面经83-客户端-字节

> @Woods Lin

客户端一面 55min

1. 自我介绍
2. 选一个自己觉得比较有意义的项目或者实习经历介绍
3. IO多路复用介绍一下
4. 那什么时候使用select
5. 使用了多线程是吗，那问一下多线程之前变量访问要注意什么
6. 都有什么锁，你比较熟悉的是什么
7. 那还有其他锁吗
8. 原子性是什么
9. 那i++是原子的吗?
10. 可以介绍下HTTP和HTTPS的差别吗
11. HTTPS怎么保证安全的
12. SSL的过程可以介绍下吗
13. DNS的过程可以介绍下吗
14. DNS传输的内容是什么
15. TCP和UDP的区别介绍下
16. 那对应的应用场景是啥嘞
17. TCP的流量控制可以介绍下吗
18. 如果接受窗口已经是0了，还会发生吗
19. TCP拥塞控制是什么
20. C++里面class和struct的区别是什么
21. 那介绍一下和C的struct的差别
22. class默认是private的，那有没有其他方法访问private变量
23. 友元有传递效果吗
24. 介绍下malloc和new的差别
25. 堆和栈的区别
26. 虚函数怎么起作用的
27. 有虚函数表对class大小有影响吗
28. C++为什么要有内存对齐，怎么实现的
29. 智能指针了解吗，介绍下
30. map和unordered_map的差别知道吗
31. 这两个哪个比较占内存
32. 数据库有了解吗，为什么要用索引
33. 一般用什么
34. 为啥用B+，不用红黑或者B
35. 算法题，再二维矩阵中查找单词，dfs即可
36. 反问

客户端二面 1h10min

1. 想问下你一面觉得有哪些答得好哪些答得不好呢
2. 你一面的面评显示你整体的计算机基础还是ok的，你本身又是非科班的，所以想问下你的一个学习方法是什么呢?
3. web服务器出于什么样的目的做的，背景是什么，从中间实现了什么学到了什么
4. 登陆是如何实现的，浏览器是如何记录用户登入状态的（简单mysql验证，状态利用类似http的cookie，服务端session）
5. 那cookie和session的区别还有什么呢？（只停留在了概念上，具体细节有点不清楚）- GET请求和POST请求的区别（语义差别，长度限制）

6. 还有其他差别吗？（没了解了）
7. 那你了解其他请求嘛？（1.1里面新增的几个DELETE）
8. HTTP返回状态码了解吗（2345介绍）
9. 实际使用计算机的时候发现QQ和微信都可以正常使用，但是访问网站比如百度无法访问，怎么确定问题（Ping判断对端响应）
10. ping存在两种情况，ping得通和ping不通，如果是ping不通的话你后续怎么做呢（不太了解）
11. ping不同有那种原因造成呢（黑名单，防火墙block）
12. 相比于qq微信，网页打开网址的一个过程是怎么样的呢（四层模型介绍）
13. HTTP使用什么协议呢，TCP还是UDP呢（TCP）
14. 那现在根据你介绍的这些步骤，刚才浏览器打不开的场景可能是出现在哪些环节呢（应用层这里把）
15. 有没有可能出现在HTTP呢
16. 浏览器打开网页第一步是什么呢
17. 域名转到IP地址这一步是不是可能存在问题
18. 其实刚才说的这个问题其实就是DNS出现问题，那现在想排查DNS，怎么判断呢？
19. 如何快速判断是否是DNS的问题呢（其实就是ping域名，ping ip看是否存在问题）
20. 操作系统内核态和用户态的区别是什么，为什么要有这么一个设定
21. 哪些事情可以在用户态做，哪些事情可以在内核态做
22. 比如说现在用键盘敲击a，这个a展现到了输入框内，这个过程出现了几次用户态到内核态的切换呢（我认为是2次）
23. 数学题。。。你可以说一下一个时钟一天分针和时针相交几次嘛（太紧张了没算明白。。。）
24. 现在你想不出来的话让你用程序去编写，并且写出相遇的时刻的时间（又是败在做题的一天，，太紧张了，，其实很简单。。）
25. 总结：前面DNS，ping之类的需要巩固，面试刷到没做过的题得冷静点

面经84-嵌入式-芯动科技

> @曲突徙薪

芯动科技我投的提前批所以开奖比较早 开了37w+10w的期权，BASE大连武汉苏州任选

一面

自我介绍然后简历上写了用过gdb和git随口问了我几个gdb和git的用法。然后是提问

Q1:Arm汇编中bl的意思

A:抱歉，arm汇编我没深入研究过，我用的平台指令集是TI自研的，但我研究过X86的汇编我推测一下。B是跳转的意思,bl的意思可能是小于跳转等于X86汇编的jl，可也能是类似x86的long jump。

这个题答案是小于跳转，没答出来但是面试官说还可以。

Q2:static 修饰的C语言变量存放在哪里，有什么作用

A：存放在data段，不会被重复初始化。

Q3:C语言变量有几种储存方式

A:存放在stack data heap bss

Q4：变量未初始化值是多少

A:stack是垃圾值，不确定，全局变量未初始化是0.

Q5: 什么是野指针

A:我认为是存放了一个不应该访问地址的指针，比如free之后的指针再次访问，访问了一个未进行初始化的指针，访问了一个函数返回的指向局部变量的指针。

Q6：外设和处理器交互的方式

A：中断，DMA，普遍嵌入式设备的外设会被映射到地址空间中，所以可以直接通过读写被映射的地址进行交互。

Q7:使用gcc编译一个hello.c的程序使用什么指令

A:gcc hello.c -o hello.out

反问环节

Q:工作中会比较多的使用gdb吗?

A:调试会用到，看你写了所以问你一下

Q:还有有几轮面试

A:一般就两轮，特别优秀的三轮。

二面

自我介绍然后：

Q1:看你简历上写了使用fft进行信号处理，讲讲吧

A:我们控制这边是用来获取对象的频率响应特性的，就是使用频率成分丰富的信号作为被控对象的输入然后获取被控对象的输出，对输入输出信号做fft分析。

Q2:在线还是离线

A:离线

Q3 简历上写了解决了cache一致性维护讲讲这个

A:实验室用的DSP是八核的一款DSP，测试读写的时候发现自读自写没有问题，但是0核写1核读读取不到正确的数据，查阅芯片手册发现可能是cache的原因，写没有写入下一级的内存，读没有无效化cache。去论坛看了一下相关的帖子，然后解决了这个问题。主要通过官方提供的两条指令cache_invalid cache_writeback。读的核需要无效化cache,写的核需要writeback写到下一级。

Q4:你还写了解决了多核同步问题，怎么解决的

A:我用的芯片没有像X86提供原子指令，转而提供了一种原子外设，官方叫做硬件信号量，访问这个外设的时候是原子的，我模拟实现了一把spinlock 访问临界区的时候上锁。

Q5:了解linux吗

A:了解，增加模仿linux写过一个简易版本的操作系统内核。

Q6:我看你简历上没写啊，讲讲你这个内核吧，跑在什么平台上的，都实现了什么功能

A:跑在i386平台，用qemu模拟器模拟的。可以实现内存管理、进程创建、Sytemcall、进程调度还有一个简易版本的内核调度

Q7:从Boot开始吧，讲讲你的内核启动过程和你实现的功能。

A：i386自导bios在启动之后会从0x7c80读取第一条指令执行，所以内核需要使用链接器来修改镜像的地址。启动后会跳转到镜像entry处的地址，这是一个初始化函数，首先会进行内存的初始化，探测剩余的内存大小，申请一块地址作为page directory,然后对剩余内存按照4kb进行分割，使用链表连接，实现内存池。需要时向内核申请。我的进程创建时模仿linux fork实现的，利用的是写时拷贝技术，只创建一个新的栈，拷贝父进程的page directory的映射，但是标记页表为只读不写，如果写会触发缺页中断，然后进行相应的拷贝。调度是就是在可以准备运行的进程中挑选一个可以准备运行的进程，简单的round-robin算法，后来加入了优先级调度，利用hash-map寻找优先级最高的进程。

Q8:在内存初始化中，怎么探测剩余内存的。

A:镜像在bss段的最后会创建的一个标记叫做end,可以探测一个cmos管知道总共的内存大小，可以间接知道剩余内存的大小。

Q9:为什么对内存进行4kb分割，如何对页表做标记

A:这是i386的MMU要求的吧，因为页表是4kb对齐的所以低12位是零每一个比特位都可以携带信息。

Q10:描述一下fork中你提到的缺页中断的执行步骤

A:首先当写一个标记为不能写的地址时，会触发缺页错误，处理器读取idt获取相应中断向量的地址，中断向量是一段汇编指令主要是储存现在寄存器的内容，跳转到isr中。此时错误地址存放在cr2中。跳转到相应的isr，检测缺页错误是否是fork标记的写时拷贝标志。如果是申请一个新的页，拷贝原来的内容然后返回错误处的地址。

Q11:返回错误处的地址具体是哪个地址，是错误的那条指令还是下一条

A:看保存的eip地址，缺页中断保存的是错误的指令地址，其他的一些systemcall是下一条指令地址

然后面试官建议我更新一下简历，把这个项目写上，还是非常感谢芯动的，面试官都很不错，尤其这位面试官给了我非常关键的建议。反问就问了工作内容，具体做什么。

面经85-嵌入式-韶音科技

> @曲突徙薪

韶音科技也是提前批，最后开了总包46w。但是我觉得赛道有点小没去，整个面试过程还是比较简单的。

一面

HR问了问基本信息，爱好啥的，没问技术

二面

Q1:简历上写了SRIO详细说一下这个总线

A:实验室用的板卡是DSP+FPGA的异构方案，通过SRIO总线进行通讯，SRIO是摩托罗拉提出的一种高速总线，主要分三层，传输、物理、链路。竞品是PCIE，PCIE网络拓扑是星型拓扑，SRIO强调点对点通讯。FPGA部分使用xilinx官方提供的IP核，DSP部分驱动是自己写的。

Q2:FPGA你了解吗

A:能看懂Verilog,调试的时候抓过FPGA部分的信号，自己写过传感器的biss协议。

Q3:简历上写了解决cache一致性维护，说一说这个

A:通上

Q4:你还写了解决了多核同步问题，怎么解决的

A:同上

Q5:为么叫硬件信号量

A:官方是这么叫的，我原来也只接触过软件信号量，官方文档大概意思是和软件信号量作用类似，没有获取到就把核心加入到一个硬件队列，资源可用了使用中断通知。

还有几个基础的C语言问题，印象里是关于C语言text data stack段的含义的

三面

有几个问题和二面重复了，围绕我项目问了我关于中断的还有一个关于C语言volatile的问题，挺基础的。对项目抓的很细，让我讲整个控制系统的机构，然后听见关键字就问我。

面经86-嵌入式-大疆

> @曲突徙薪

一面

自我介绍后，面试官自我介绍然后没咋看项目直接开问。

Q1:同学请你谈一下你对操作系统的理解，为什么有操作系统。

A:我认为操作系统在嵌入式中是对最简单的前后台机制的进一步抽象，嵌入式系统中最原始的做法是死循环然后等待中断，这可以被视为操作系统的雏形，一个后台一个前台，随着任务的增多，不能把所有任务都放到ISR中执行，同时为了更好的管理内存甚至拥有文件系统，引入了操作系统。

Q2:你谈到了中断是操作系统的雏形，那我多创建几个进程不久可以执行多个任务了。

A:理论上可以，但是中断触发是异步的，没有办法控制执行顺序。如果你的所有任务可以被精确分时，也可以做到执行多个任务的效果，更本质的原因是中断是执行了一个函数，没有办法像进程那样使用一个数据结构表示，所以谈不上对中断的调度，所以中断不能够睡眠，只能一条路执行到黑。

Q3:进程和线程有什么区别

A:教科书上说进程是管理资源的最小单位，进程调度花费比线程更高啥的。但从我看过的两个操作系统一个TI的sys-bios和linux来说，进程和线程没有区别，linux中线程就被叫做轻量级进程，sys-bios中没有进程的概念，最小单位就是线程。

Q4:什么时候会进行进程调度，如何实现的进程调度

A:从内核态返回用户态会发生进程调度，进程在执行时由于发生systemcall或者因为中断进入内核态时会保存所有的寄存器值，在调度选择执行某个进程时，恢复这些寄存器然后先前进程的地址空间就可以实现一次调度了。

Q5:为什么进程的调度花费比线程大

A:这要看线程时如何实现的，如何在用户态实现线程，就不会发生页表切换。开销小，但是在用户态实现的线程非常少，在内核态实现可以由内核进行调度，用户态线程不放弃就一直执行。

面试官还问了我俩问题，记不清了应该不难。问我为什么想来大疆，能为大疆带来什么。

面经87-测开-美团暑期实习

> @米新举

一面（3.16）

这个测开一面,感觉纯属聊天啦, 主要是看有没有兴趣做,面试官开始先问我对测开的认知,然后给我普及,说测开, 测试和开发一般是八三的关系,一位在百度测开实习的老哥说七三,但基本差不多啦.

全程,好像是面试官在给我上课,教我一些测试理论的东西,这一点也给二面打下了还行的基础

1. 计算机科学与技术目前的培养方案,你们学校的方向,都上了什么课
2. 你对测试开发的理解(我说测试开发也是开发,吧啦吧啦,面试官说,是这样的,,但是我们部分偏测试,有些部门的确是开发自动化工具这些的) 一通讲解以后,问我,那你现在是想做偏测试还是偏开发(我只是想要offer,于是回答说,还是想测试)
3. 一个单词文本,统计前 k 个出现频率最高的单词, 这个面试官主要想问就是, JAVA 中的HashMap 可以自定义一个比较方法吧,但是我是C++ 选手呀,我就按照C++的给他说了.C++ 应该是没有办法直接定义 map 按照v的排序方法的.要给一个 map排序,只能先推进一个容器,然后对容器排序,这个问题我最早在大一遇到过
4. SQL 题, 第一个就是用个 count 和grpup by 就好了.然后又给了一个, 这个我不太会了,面试官一直在引导,但是我不太行,最后说,这个你下去在看看
5. 平时如何学习的,现在给你一个新的东西,你如何去学习(我就举了我当时使用gtest 的例子,先去搜一下这个东西大概是什么,做一个了解,然后如果有官方文档的话,去看官方的文档, 有某一点不太清楚了再去特意搜索一些, 如果有不错的书籍,我会借一本或者买一本来读一读)
6. 一个例如微信扫码和支付宝扫码的功能如何测试(安全性, 复杂网络环境, 光线较弱,我答了这三个方面,但是实际就是覆盖了测试理论的安全性和功能性这两个方面) 然后就是面试官给我上课啦
7. 编写测试用例,从那几个方面考虑,我说,边界值一定要考虑,然后还有就是合法的值, 因为合法的很多,我们只能选去一部分,还有就是不合法的

二面（3.22）

1. 你对测开的了解
2. 说一下你的项目
3. 计算机网络典中典(三握四挥)
4. 测试用例设计
5. redis 中的哈希表的结构及实现
6. 哈希表解决冲突的方法
7. SQL 语句,就问了一下,也没有具体的题目
8. 问项目的时候,我主动说了我遇到的问题,当时是如何debug 的,如何解决的