查看下什么是间隙锁 1

什么是MVCC 1

网页怎样去重 2

二级索引 2

消息队列mq 2

朴素贝叶斯算法 3

倒排索引 3

页分裂 3

完美的单例 3

布隆过滤器 3

ajax跨域问题 3

在内存中定位数组的位置，是一下找到这个位置还是从头遍历 3

索引最重要的是区分度要高，其次最好能有顺序。 3

文件排序 3

索引为什么只能是最左前缀 3

查看下事务的原理 3

什么是句柄 3

18、HTTP与socket的区别 3

oss与cdn的区别 4

什么是CDN 4

查看域名对应的IP 4

消息队列Linux实现原理 4

共享内存实现原理 5

信号量实现原理 5

剑指offer 5

查看shadowsocket源码 5

查看blender源码 5

查看open VPN源码 5

看证书的原理 5

对称加密 5

非对称加密 5

Linux中的inode 5

手机抓包原理 6

## 查看下什么是间隙锁

## 什么是MVCC

全称：Multi-Version Concurrent Control。原理类似于Java中的copyonwrite，在写操作的过程中先复制一份原来的数据，读操作会读取复制的数据，这样来保证写操作不会阻塞和影响读操作，该机制一般用于读操作多于写操作的场景中。

在MySQL中，update操作会分成两步，insert和delete。

## 网页怎样去重

## 二级索引

MySQL中每个表都有一个聚簇索引，除此之外的索引都叫非聚簇索引也叫二级索引。

在InnoDB引擎中，主键就是聚簇索引。如果没有定义主键，那么就会选第一个唯一索引并且只含非空列作为主键。如果没有这样的列，MySQL就会自己创建一个主键而且是隐藏的。

## 消息队列mq

## 朴素贝叶斯算法

## 倒排索引

## 页分裂

## 完美的单例

## 布隆过滤器

## ajax跨域问题

## 在内存中定位数组的位置，是一下找到这个位置还是从头遍历

## 索引最重要的是区分度要高，其次最好能有顺序。

## 文件排序

## 索引为什么只能是最左前缀

## 查看下事务的原理

## 什么是句柄

## 18、HTTP与socket的区别

HTTP为短连接，客户端发送请求给服务器，服务器回送响应，请求结束，主动释放连接。通常的做法是，不需要任何数据，也要保持每隔一段时间向服务器发送“保持连接”请求。这样可以保证客户端在服务端是“上线”状态。

Socket是对TCP/IP协议的封装，Socket只是个接口不是协议，通过Socket我们才能使用TCP/IP协议或者UDP协议。

## oss与cdn的区别

简单来说，OSS是存文件的，相当于网盘；CDN是给你分发内容的，相当于缓存。

## 什么是CDN

## 查看域名对应的IP

使用命令nslookup可以查看域名对应的IP或者IP对应的域名。

## 消息队列Linux实现原理

## 共享内存实现原理

## 信号量实现原理

## 剑指offer

## 查看shadowsocket源码

## 查看blender源码

## 查看open VPN源码

## 看证书的原理

### 对称加密

加密与解密使用相同的密钥。一般这种加密方式用在加密大量的数据的时候，因为其运算非常快。

缺点：密钥的管理很困难。比如，密钥在传输的过程中很容易被黑客截获。那么密钥的传输通常使用非对称加密。

常用的对称加密算法为DES和AES

### 非对称加密

加密和解密使用不用的密钥------公钥和私钥。

缺点：运算比较慢

常用的非对称加密算法RSA

## Linux中的inode

文件存储在硬盘上，硬盘的最小存储单位叫做“扇区”(sector)，每个扇区储存512字节。操作系统在读取硬盘的时候不会一个扇区一个扇区的读取，而是多个连续的扇区读取，即一次性读取一个“块”，“块”是文件存储的最小单位。块的大小通常是4KB，即连续8个扇区。

文件的数据储存在块中，那么文件的元信息，即文件的创建者、文件的创建日期、文件的大小等储存的区域叫做inode，即“索引节点”。

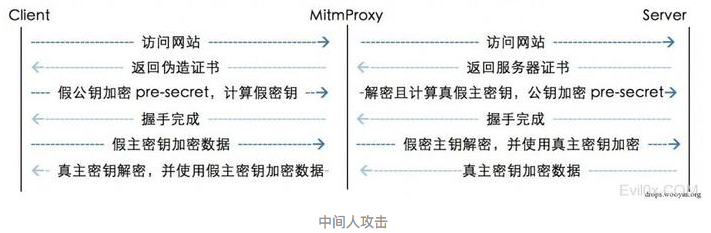
可以通过使用df –i列出inode的信息。

可以使用stat命令查看某个特定文件的inode信息，stat example.txt

inode也会消耗磁盘的大小。索引当inode区域没有可使用空间时，即使还有数据存储空间，但是也无法创建文件了。

每一个inode都有一个号码，操作系统用inode来识别文件。可以通过inode号码来删除文件，使用ls –i来列出文件的号码。

## 手机抓包原理



* 代理服务器作为一个中间人，client先向proxy发出请求，proxy再将请求转发给server
* server返回证书给proxy，proxy伪造一个证书给client。疑问：为什么可以随便伪造一个证书给client？客户端通过以下三种方式来验证证书的合法性：1、查看证书是否过期2、服务器证书上的域名是否和服务器的实际域名相匹配。3、校验证书链。这里可以手动添加信任的证书给客户端。证书当中包含服务器的公钥，来使客户端对后面的对称加密的密钥进行加密，然后传输给服务器。
* client使用假证书的公钥来对pre-secret(主密钥)进行加密，之后proxy解出该pre-secret，使用真的公钥进行加密传输给服务器。服务器使用私钥进行解密。
* 握手完成

后面的两步就是使用对称加密进行通信了。

# OAEP

# Linux进程调度