软件工程的核心概念，

Q：如何从一个需求落实到一个系统设计

A：

Q：如何衡量两个不同设计的好坏

可伸缩性（Scalability）当服务的负载增长时，系统能被扩展来满足需求，且不降低服务质量。简单的说，服务是可扩展的，并且扩展的成本是比较合理的。

高可用性（Availability）尽管部分硬件和软件会发生故障，整个系统的服务必须是每天24小时每星期7天可用的。一般通过软件和硬件的冗余来实现。

可管理性（Manageability）整个系统可能在物理上很大，但应该容易管理。需要开发对应的管理工具。

价格有效性（Cost-effectiveness）整个系统实现是经济的、易支付的。这点很多人可能会忽略，架构设计需要考虑ROI因素，如果一个架构很好，但成本高得惊人不一定是合适的架构。

Q：如何在各种限制下（人员、时间、资源等）选择其中更合适的设计，以及提升该设计的可拓展性等。

A：

**计算机系统**

Q：进程、线程、协程

进程是系统进行资源分配的一个独立单元；线程是CPU调度的基本单元；协程比线程更加轻量，协程不被操作系统所控制，而是由程序所控制。

同一个进程的多个线程可并发执行

一个线程就是一个子程序，协程在子程序内部是可中断的。

协程的特点在于一个线程执行，与多线程相比，其优势是：执行效率高，因为子程序切换不是线程切换，而是由程序控制，无线程切换开销；不需要多线程的锁机制，因为只有一个线程，不存在同时写变量冲突

Q：多进程与多线程比较



Q：进程间的通信IPC(Inter-process communication)

管道（pipe）、命名管道（FIFO）、消息队列（message queue）、信号量（semaphore）

、共享内存（shared memory）、嵌套字（socket）

1)管道

管道分为有名管道和无名管道

无名管道是一种半双工的通信方式,数据只能单向流动,而且只能在具有亲缘关系的进程间使用.进程的亲缘关系一般指的是父子关系。无明管道一般用于两个不同进程之间的通信。当一个进程创建了一个管道,并调用fork创建自己的一个子进程后,父进程关闭读管道端,子进程关闭写管道端,这样提供了两个进程之间数据流动的一种方式。

有名管道也是一种半双工的通信方式,但是它允许无亲缘关系进程间的通信。

无名管道：优点：简单方便；缺点：1）局限于单向通信2）只能创建在它的进程以及其有亲缘关系的进程之间;3）缓冲区有限；

有名管道：优点：可以实现任意关系的进程间的通信；缺点：1）长期存于系统中，使用不当容易出错；2）缓冲区有限

2)信号量

信号量是一个计数器,可以用来控制多个线程对共享资源的访问.,它不是用于交换大批数据,而用于多线程之间的同步.它常作为一种锁机制,防止某进程在访问资源时其它进程也访问该资源.因此,主要作为进程间以及同一个进程内不同线程之间的同步手段.

优点：可以同步进程；缺点：信号量有限

3)信号

信号是一种比较复杂的通信方式,用于通知接收进程某个事件已经发生.

4)消息队列

消息队列是消息的链表,存放在内核中并由消息队列标识符标识.消息队列克服了信号传递信息少,管道只能承载无格式字节流以及缓冲区大小受限等特点.消息队列是UNIX下不同进程之间可实现共享资源的一种机制,UNIX允许不同进程将格式化的数据流以消息队列形式发送给任意进程.对消息队列具有操作权限的进程都可以使用msget完成对消息队列的操作控制.通过使用消息类型,进程可以按任何顺序读信息,或为消息安排优先级顺序.

优点：可以实现任意进程间的通信，并通过系统调用函数来实现消息发送和接收之间的同步，无需考虑同步问题，方便；缺点：信息的复制需要额外消耗CPU的时间，不适宜于信息量大或操作频繁的场合

5)共享内存

共享内存就是映射一段能被其他进程所访问的内存,这段共享内存由一个进程创建,但多个进程都可以访问.共享内存是最快的IPC(进程间通信)方式,它是针对其它进程间通信方式运行效率低而专门设计的.它往往与其他通信机制,如信号量,配合使用,来实现进程间的同步与通信.

优点：无须复制，快捷，信息量大；缺点：1）通信是通过将共无法实现享空间缓冲区直接附加到进程的虚拟地址空间中来实现的，因此进程间的读写操作的同步问题；2)利用内存缓冲区直接交换信息，内存的实体存在于计算机中，只能同一个计算机系统中的诸多进程共享，不方便网络通信

6)套接字：可用于不同及其间的进程通信

优点：1）传输数据为字节级，传输数据可自定义，数据量小效率高；2）传输数据时间短，性能高；3) 适合于客户端和服务器端之间信息实时交互；4) 可以加密,数据安全性强

缺点：1) 需对传输的数据进行解析，转化成应用级的数据。

Q：线程之间的通信方式

# 锁机制：包括互斥锁、条件变量、读写锁

\*互斥锁提供了以排他方式防止数据结构被并发修改的方法。

\*读写锁允许多个线程同时读共享数据，而对写操作是互斥的。

\*条件变量可以以原子的方式阻塞进程，直到某个特定条件为真为止。对条件的测试是在互斥锁的保护下进行的。条件变量始终与互斥锁一起使用。

# 信号量机制(Semaphore)：包括无名线程信号量和命名线程信号量

# 信号机制(Signal)：类似进程间的信号处理

线程间的通信目的主要是用于线程同步，所以线程没有像进程通信中的用于数据交换的通信机制。

Q：什么时候用多进程？什么时候用多线程？

1）需要频繁创建销毁的优先用线程

这种原则最常见的应用就是Web服务器，来一个连接建立一个线程，断了就销毁线程，要是用进程，创建和销毁的代价是很难承受的

2）需要进行大量计算的优先使用线程

所谓大量计算，当然就是要耗费很多CPU，切换频繁了，这种情况下线程是最合适的。这种原则最常见的是图像处理、算法处理。

3）强相关的处理用线程，弱相关的处理用进程

什么叫强相关、弱相关？理论上很难定义，给个简单的例子就明白了。

一般的Server需要完成如下任务：消息收发、消息处理。“消息收发”和“消息处理”就是弱相关的任务，而“消息处理”里面可能又分为“消息解码”、“业务处理”，这两个任务相对来说相关性就要强多了。因此“消息收发”和“消息处理”可以分进程设计，“消息解码”、“业务处理”可以分线程设计。

当然这种划分方式不是一成不变的，也可以根据实际情况进行调整。

4）可能要扩展到多机分布的用进程，多核分布的用线程

Q：进程有哪几种状态？

就绪状态：进程已获得除处理机以外的所需资源，等待分配处理机资源；

运行状态：占用处理机资源运行，处于此状态的进程数小于等于CPU数；

阻塞状态： 进程等待某种条件，在条件满足之前无法执行；

Q：线程同步方式

1、互斥量 Synchronized/Lock：采用互斥对象机制，只有拥有互斥对象的线程才有访问公共资源的权限。因为互斥对象只有一个，所以可以保证公共资源不会被多个线程同时访问

2、信号量 Semphare：它允许同一时刻多个线程访问同一资源，但是需要控制同一时刻访问此资源的最大线程数量

3、事件(信号)，Wait/Notify：通过通知操作的方式来保持多线程同步，还可以方便的实现多线程优先级的比较操作

**C++**

Q：new和malloc的区别

1. 属性

new/delete是C++关键字，需要编译器支持。malloc/free是库函数，需要头文件支持。

1. 参数

使用new操作符申请内存分配时无须指定内存块的大小，编译器会根据类型信息自行计算。而malloc则需要显式地指出所需内存的尺寸。

1. 返回类型

new操作符内存分配成功时，返回的是对象类型的指针，类型严格与对象匹配，无须进行类型转换，故new是符合类型安全性的操作符。而malloc内存分配成功则是返回void \* ，需要通过强制类型转换将void\*指针转换成我们需要的类型。

1. 分配失败

new内存分配失败时，会抛出bac\_alloc异常。malloc分配内存失败时返回NULL。

1. 自定义类型

new会先调用operator new函数，申请足够的内存（通常底层使用malloc实现）。然后调用类型的构造函数，初始化成员变量，最后返回自定义类型指针。delete先调用析构函数，然后调用operator delete函数释放内存（通常底层使用free实现）。

malloc/free是库函数，只能动态的申请和释放内存，无法强制要求其做自定义类型对象构造和析构工作。

1. 重载

C++允许重载new/delete操作符，特别的，布局new的就不需要为对象分配内存，而是指定了一个地址作为内存起始区域，new在这段内存上为对象调用构造函数完成初始化工作，并返回此地址。而malloc不允许重载。

1. 内存区域

new操作符从自由存储区（free store）上为对象动态分配内存空间，而malloc函数从堆上动态分配内存。自由存储区是C++基于new操作符的一个抽象概念，凡是通过new操作符进行内存申请，该内存即为自由存储区。而堆是操作系统中的术语，是操作系统所维护的一块特殊内存，用于程序的内存动态分配，C语言使用malloc从堆上分配内存，使用free释放已分配的对应内存。自由存储区不等于堆，如上所述，布局new就可以不位于堆中。

Q：C++里的智能指针

C++里面的四个智能指针，auto\_ptr，unique\_ptr，shared\_ptr，weak\_ptr，其中后三个是c++11支持，并且第一个已经被c++11弃用。

使用原因：智能指针的作用是管理一个指针，因为存在以下这种情况：申请的空间在函数结束时忘记释放，造成内存泄漏。使用智能指针可以很大程度上的避免这个问题，因为智能指针是一个类，当超出了类的实例对象的作用域时，会自动调用对象的析构函数，析构函数会自动释放资源。所以智能指针的作用原理就是在函数结束时自动释放内存空间，不需要手动释放内存空间。

auto\_ptr：采用所有权模式。p2剥夺了p1的所有权，但是当程序运行时访问p1将会报错。所以auto\_ptr的缺点是：存在潜在的内存崩溃问题。

unique\_ptr：实现独占式拥有或严格拥有概念，保证同一时间内只有一个智能指针可以指向该对象。它对于避免资源泄露(例如“以new创建对象后因为发生异常而忘记调用delete”)特别有用，可以通过标准库的move()函数实现指针转移。

shared\_ptr：实现共享式拥有概念。多个智能指针可以指向相同对象，该对象和其相关资源会在“最后一个引用被销毁”时候释放。从名字share就可以看出了资源可以被多个指针共享，它使用计数机制来表明资源被几个指针共享。

weak\_ptr：是一种不控制对象生命周期的智能指针, weak\_ptr 设计的目的是为配合 shared\_ptr 而引入的一种智能指针来协助 shared\_ptr 工作, 它只可以从一个 shared\_ptr 或另一个 weak\_ptr 对象构造, 它的构造和析构不会引起引用记数的增加或减少。

Q： 拷贝构造函数

定义：拷贝构造函数是一种特殊的构造函数，函数的名称必须和类名称一致，它必须的一个参数是本类型的一个引用变量。

作用：用来复制对象，使用这个对象的实例来初始化这个对象的一个新的实例。

调用时机：

当函数的参数为类的对象时

函数的返回值是类的对象

对象需要通过另外一个对象进行初始化。

拷贝构造函数默认为浅拷贝。浅拷贝是指当出现类的等号赋值时，它能够完成静态成员的值复制。当数据成员中没有指针时，浅拷贝是可行的。但当数据成员中有指针时，如果采用简单的浅拷贝，则两个类中的指针将指向同一个地址，对象即将结束时，两个类会分别调用析构函数，导致指针悬挂现象。所以这时必须采用深拷贝，在堆内存中另外申请空间来储存数据，防止指针悬挂现象。

拷贝构造函数必须是引用传递，不能是值传递，这是为了防止递归引用。

**计网**

Q：Http和Https的区别

Http协议运行在TCP之上，明文传输，客户端与服务器端都无法验证对方的身份；Https是身披SSL(Secure Socket Layer)外壳的Http，运行于SSL上，SSL运行于TCP之上，是添加了加密和认证机制的HTTP。二者之间存在如下不同：

端口不同：Http与Https使用不同的连接方式，用的端口也不一样，前者是80，后者是443；

资源消耗：和HTTP通信相比，Https通信会由于加减密处理消耗更多的CPU和内存资源；

开销：Https通信需要证书，而证书一般需要向认证机构购买；

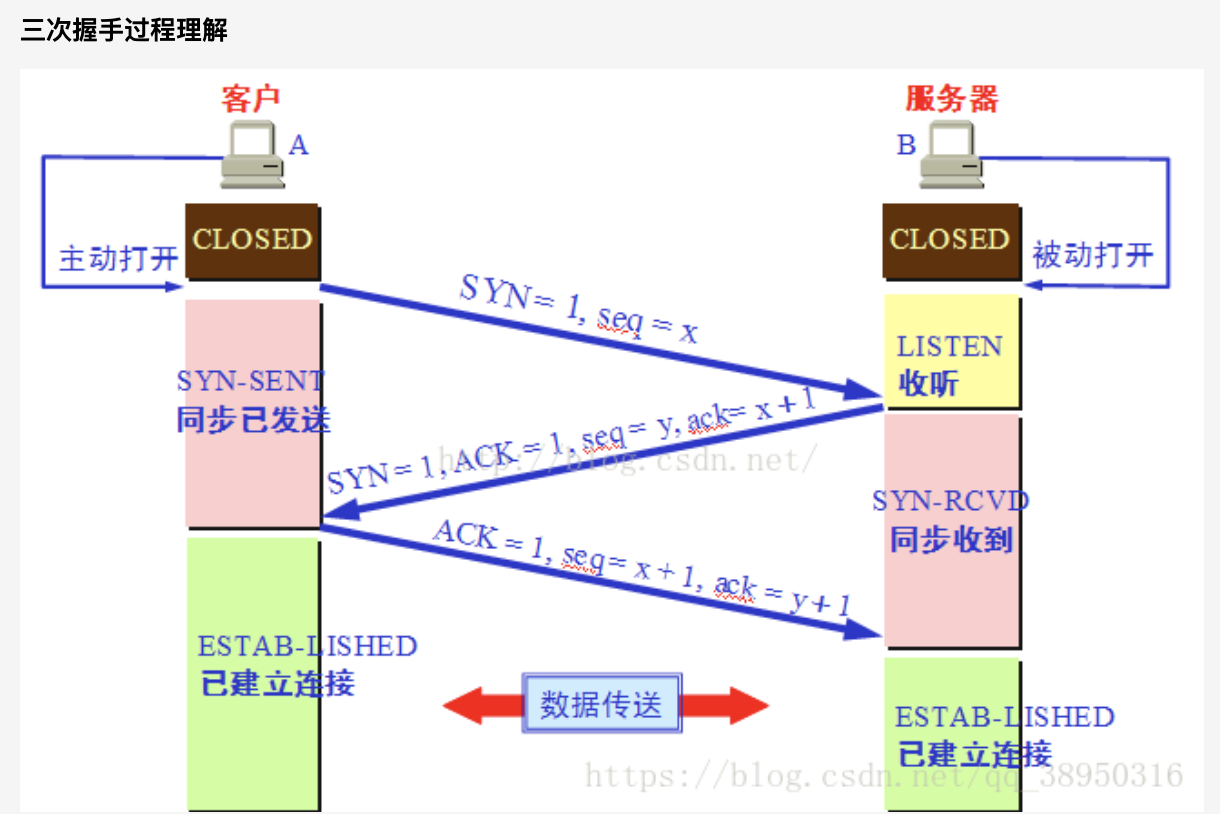
Https的加密机制是一种共享密钥加密和公开密钥加密并用的混合加密机制。

Q：TCP的三次握手与四次挥手

确认ACK：占1位，仅当ACK=1时，确认号字段才有效。ACK=0时，确认号无效

同步SYN：连接建立时用于同步序号。当SYN=1，ACK=0时表示：这是一个连接请求报文段。若同意连接，则在响应报文段中使得SYN=1，ACK=1。因此，SYN=1表示这是一个连接请求，或连接接受报文。SYN这个标志位只有在TCP建产连接时才会被置1，握手完成后SYN标志位被置0。

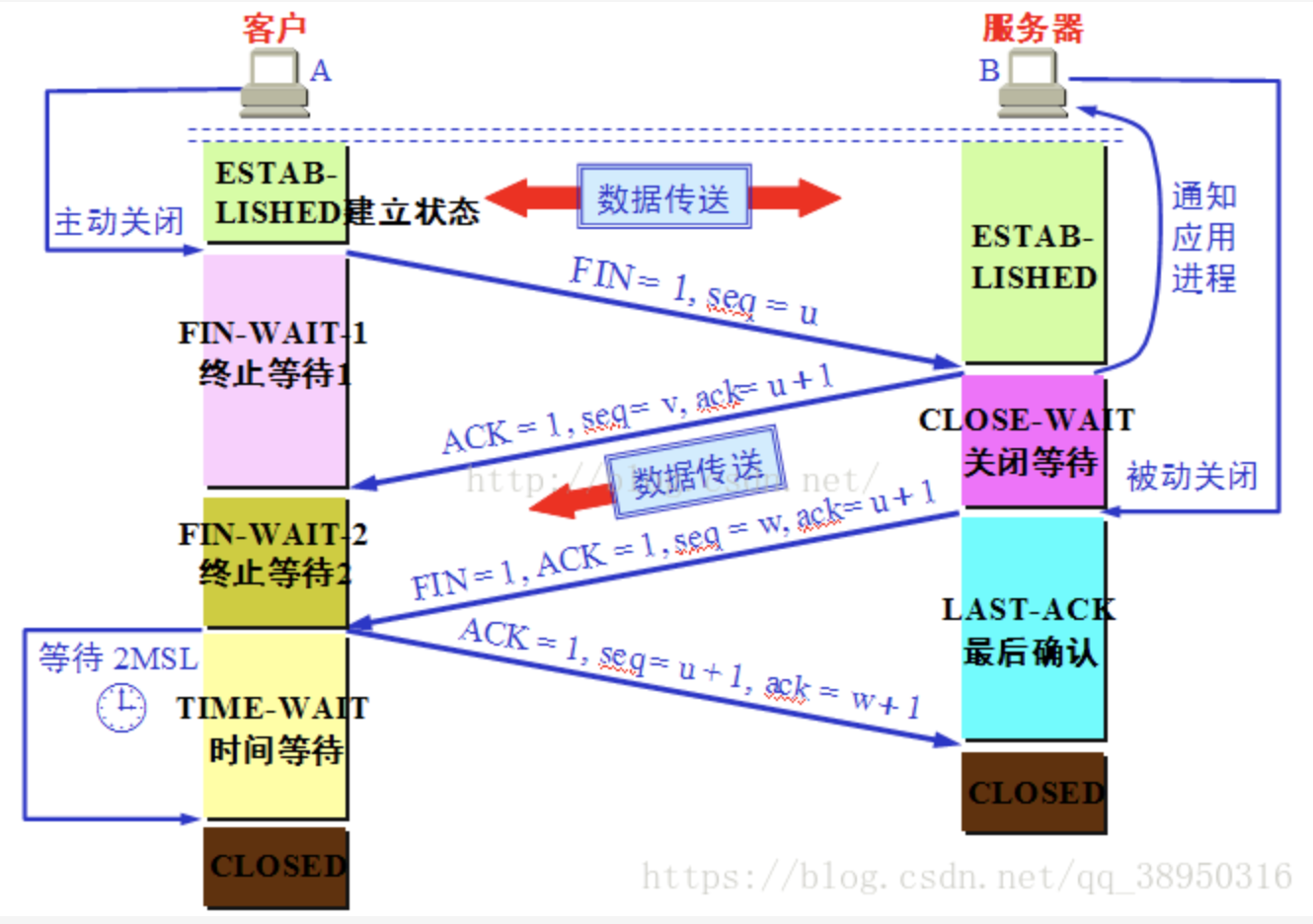
终止FIN：用来释放一个连接。FIN=1表示：此报文段的发送方的数据已经发送完毕，并要求释放运输连接



第一次握手：建立连接时，客户端发送syn包（syn=j）到服务器，并进入SYN\_SENT状态，等待服务器确认；SYN：同步序列编号（Synchronize Sequence Numbers）。

第二次握手：服务器收到syn包，必须确认客户的SYN（ack=j+1），同时自己也发送一个SYN包（syn=k），即SYN+ACK包，此时服务器进入SYN\_RECV状态；

第三次握手：客户端收到服务器的SYN+ACK包，向服务器发送确认包ACK(ack=k+1），此包发送完毕，客户端和服务器进入ESTABLISHED（TCP连接成功）状态，完成三次握手。



1）客户端进程发出连接释放报文，并且停止发送数据。释放数据报文首部，FIN=1，其序列号为seq=u（等于前面已经传送过来的数据的最后一个字节的序号加1），此时，客户端进入FIN-WAIT-1（终止等待1）状态。 TCP规定，FIN报文段即使不携带数据，也要消耗一个序号。

2）服务器收到连接释放报文，发出确认报文，ACK=1，ack=u+1，并且带上自己的序列号seq=v，此时，服务端就进入了CLOSE-WAIT（关闭等待）状态。TCP服务器通知高层的应用进程，客户端向服务器的方向就释放了，这时候处于半关闭状态，即客户端已经没有数据要发送了，但是服务器若发送数据，客户端依然要接受。这个状态还要持续一段时间，也就是整个CLOSE-WAIT状态持续的时间。

3）客户端收到服务器的确认请求后，此时，客户端就进入FIN-WAIT-2（终止等待2）状态，等待服务器发送连接释放报文（在这之前还需要接受服务器发送的最后的数据）。

4）服务器将最后的数据发送完毕后，就向客户端发送连接释放报文，FIN=1，ack=u+1，由于在半关闭状态，服务器很可能又发送了一些数据，假定此时的序列号为seq=w，此时，服务器就进入了LAST-ACK（最后确认）状态，等待客户端的确认。

5）客户端收到服务器的连接释放报文后，必须发出确认，ACK=1，ack=w+1，而自己的序列号是seq=u+1，此时，客户端就进入了TIME-WAIT（时间等待）状态。注意此时TCP连接还没有释放，必须经过2∗∗MSL（最长报文段寿命）的时间后，当客户端撤销相应的TCB后，才进入CLOSED状态。

6）服务器只要收到了客户端发出的确认，立即进入CLOSED状态。同样，撤销TCB后，就结束了这次的TCP连接。可以看到，服务器结束TCP连接的时间要比客户端早一些。

Q：为什么连接的时候是三次握手，关闭的时候却是四次握手？

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

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

虽然按道理，四个报文都发送完毕，我们可以直接进入CLOSE状态了，但是我们必须假象网络是不可靠的，有可以最后一个ACK丢失。所以TIME\_WAIT状态就是用来重发可能丢失的ACK报文。在Client发送出最后的ACK回复，但该ACK可能丢失。Server如果没有收到ACK，将不断重复发送FIN片段。所以Client不能立即关闭，它必须确认Server接收到了该ACK。Client会在发送出ACK之后进入到TIME\_WAIT状态。Client会设置一个计时器，等待2MSL的时间。如果在该时间内再次收到FIN，那么Client会重发ACK并再次等待2MSL。所谓的2MSL是两倍的MSL(Maximum Segment Lifetime)。MSL指一个片段在网络中最大的存活时间，2MSL就是一个发送和一个回复所需的最大时间。如果直到2MSL，Client都没有再次收到FIN，那么Client推断ACK已经被成功接收，则结束TCP连接。

Q：为什么不能用两次握手进行连接？

3次握手完成两个重要的功能，既要双方做好发送数据的准备工作(双方都知道彼此已准备好)，也要允许双方就初始序列号进行协商，这个序列号在握手过程中被发送和确认。

现在把三次握手改成仅需要两次握手，死锁是可能发生的。作为例子，考虑计算机S和C之间的通信，假定C给S发送一个连接请求分组，S收到了这个分组，并发 送了确认应答分组。按照两次握手的协定，S认为连接已经成功地建立了，可以开始发送数据分组。可是，C在S的应答分组在传输中被丢失的情况下，将不知道S 是否已准备好，不知道S建立什么样的序列号，C甚至怀疑S是否收到自己的连接请求分组。在这种情况下，C认为连接还未建立成功，将忽略S发来的任何数据分 组，只等待连接确认应答分组。而S在发出的分组超时后，重复发送同样的分组。这样就形成了死锁。

Q：如果已经建立了连接，但是客户端突然出现故障了怎么办？

TCP还设有一个保活计时器，显然，客户端如果出现故障，服务器不能一直等下去，白白浪费资源。服务器每收到一次客户端的请求后都会重新复位这个计时器，时间通常是设置为2小时，若两小时还没有收到客户端的任何数据，服务器就会发送一个探测报文段，以后每隔75秒钟发送一次。若一连发送10个探测报文仍然没反应，服务器就认为客户端出了故障，接着就关闭连接。

拥塞控制和流量控制

快重传和超时重传

DNS解析全过程

http2.0，http3.0

https的连接过程

CA证书的验证过程

**MySQL**

Q：什么是事务

事务是数据库操作的最小工作单元，是作为单个逻辑工作单元执行的一系列操作；这些操作作为一个整体一起向系统提交，要么都执行、要么都不执行；事务是一组不可再分割的操作集合（工作逻辑单元）；

Q：事务的四大特性：

1 、原子性

事务是数据库的逻辑工作单位，事务中包含的各操作要么都做，要么都不做

2 、一致性

事务执行的结果必须是使数据库从一个一致性状态变到另一个一致性状态。因此当数据库只包含成功事务提交的结果时，就说数据库处于一致性状态。如果数据库系统 运行中发生故障，有些事务尚未完成就被迫中断，这些未完成事务对数据库所做的修改有一部分已写入物理数据库，这时数据库就处于一种不正确的状态，或者说是 不一致的状态。

3 、隔离性

一个事务的执行不能其它事务干扰。即一个事务内部的操作及使用的数据对其它并发事务是隔离的，并发执行的各个事务之间不能互相干扰。

4 、持续性

也称永久性，指一个事务一旦提交，它对数据库中的数据的改变就应该是永久性的。接下来的其它操作或故障不应该对其执行结果有任何影响。

Q：事务的并发问题

1、脏读：事务A读取了事务B更新的数据，然后B回滚操作，那么A读取到的数据是脏数据

2、不可重复读：事务 A 多次读取同一数据，事务 B 在事务A多次读取的过程中，对数据作了更新并提交，导致事务A多次读取同一数据时，结果 不一致。

3、幻读：系统管理员A将数据库中所有学生的成绩从具体分数改为ABCDE等级，但是系统管理员B就在这个时候插入了一条具体分数的记录，当系统管理员A改结束后发现还有一条记录没有改过来，就好像发生了幻觉一样，这就叫幻读。

小结：不可重复读的和幻读很容易混淆，不可重复读侧重于修改，幻读侧重于新增或删除。解决不可重复读的问题只需锁住满足条件的行，解决幻读需要锁表

Q：MySQL事务隔离级别

|  |  |  |  |
| --- | --- | --- | --- |
| 事务隔离级别 | 脏读 | 不可重复度 | 幻读 |
| 读未提交（read-uncommitted） | 是 | 是 | 是 |
| 不可重复度（read-committed） | 否 | 是 | 是 |
| 可重复读（repeatable-read） | 否 | 否 | 是 |
| 串行化（serializable） | 否 | 否 | 否 |

MySQL默认的事务隔离级别为repeatable-read

Innodb，B+树，聚簇索引和非聚簇索引区别

一千万条数据，问 b+ 数大概多高

为什么用 b+ 树而不是 b树，红黑树

在设计联合索引需要注意什么

联合索引最左原则的底层原理

**场景题和智力题**

打字的智能提示和纠错的原理，让你来设计，你怎么做。

如果有一个访问过的所有网站的记录集合，如何快速判断当前网站是已经访问过。（考虑布隆过滤器）

从一个网站下载的时候，进度条如何实现的。

64匹马，八个赛道，问至少跑多少次能找到最快四匹马。

操作系统给你一块空闲内存，你如何设计使用和回收。

构造函数为什么不能是虚函数，而基类的析构函数必须是虚函数

当定义一个对象的时候，首先会分配对象内存空间，然后调用构造函数来初始化对象。vptr变量是在构造函数中进行初始化的。又因为执行虚函数需要通过vptr指针来调用。如果可以定义构造函数为虚函数，那么就会陷入先有鸡还是先有蛋的循环讨论中。析构函数写成virtual虚析构函数，为了防止构造函数中分配的内存泄漏。

如何用固定长度数组实现队列？

使用两个变量i和j，i指向队列的头，j指向队列的尾，如果加入新的元素，将i++（超过数组长度时，从0开始）；如果弹出元素的话，j++（超过数组长度时同上）。

TCP和UDP的区别

TCP提供了一个稳定可靠的传输协议，通过各种手段(如:三挥四握)提供了稳定的传输，消耗也会大很多，而且只能一对一，TCP面向字节流；UDP就是不需要建立连接，传输效率高，但不保证稳定可靠，数据可能会丢失，可以多对多，UDP是面向报文的