unordered\_map/std::map的区别

底层实现：

unordered\_map底层是哈希表，内部数据的排放是无序的，数据的查找可达到O(1)的时间复杂度，特别适合于海量数据，map底层是红黑树，内部所有数据都是有序的。

优缺点：

map内的数据是有序的，这是它最大的优点，红黑树的底层实现使得很多操作只需要log（n），但空间占用率高，每个节点都要额外保存父节点、孩子节点和红黑性质，使得每个节点都要占用大量空间。

unordered\_map内部实现了哈希表，查找速度很快，但哈希表的建立比较耗费时间。

TCP粘包

指发送方发送的若干数据包到达接受方时粘成一包，后一包的数据的头紧接着前一包数据的尾。

原因：

发送方默认使用NAGLE算法（只有上一个分组的确认到达，才会发送下一个分组，并收集多个小分组在一个确认到来时一起发送）

接收方收到数据包以后，并不会马上送到应用层处理，而是保存在接收缓存里，如果缓存的速度大于处理的速度，就可能出现粘包现象。

处理办法：

关闭NAGLE算法，使消息有边界，告知消息长度

SQL四个隔离级别及可能引发问题

未提交读：可能导致脏读、幻读，不可重复读

已提交读：可能导致幻读和不可重复读

可重复读：可能导致幻读

可串行化：无

\*C++多继承虚继承虚析构函数

在浏览器地址栏输入URL后，到显示页面的过程：

1. 浏览器输入域名
2. 浏览器查找域名对应的IP地址（DNS）
3. TCP三次握手建立连接
4. 浏览器通过https协议发送请求

浏览器向主机发起一个HTTP-GET方法报文请求

1. 某些服务会做永久重定向相应

对于大型网站存在多个主机站点，为了负载均衡或导入流量。浏览器在获取重定向响应之后，重新访问。

1. 浏览器跟踪重定向地址

浏览器知道了重定向后最终的访问地址以后，重新发送HTTPS请求

1. 服务器处理请求
2. 服务器发出一个HTML相应
3. 释放TCP连接

针对UDP数据的4种NAT行为

1. Full Cone：所有从相同的内部IP和PORT发出的请求都会被映射成相同的外部IP和PORT，而后任何外部主机只要发送数据包给NAT的IP和PORT都会被转发给内部主机
2. Restricted Cone：所有从相同的内部IP和PORT发出的请求都会被映射成相同的外部IP和PORT，但只有内部主机曾经发送过得外部IP才可将数据包通过NAT的IP：PORT发送给内部主机。
3. PORT Restricted Cone：和Restricted Cone 类似，但是除了IP的限制之外增加了PORT的限制，即只有内部主机层发送过得IP：PORT才可将数据发至内部主机。
4. Symmetric NAT：从内部主机相同的IP和PORT发出的请求，当访问不同外部IP和PORT时，都会在NAT上创建不同的映射。

TCP如何实现流量控制

通过滑动窗口协议实现流量控制，接收方会维护一个接收窗口反应可用的缓冲区资源，并告知发送方，发送方的发送窗口不能超过接收窗口与拥塞窗口的最小值。

TCP如何实现拥塞控制

拥塞控制主要由四个部分组成：慢启动，拥塞避免，快速重传，快速恢复。

慢启动：刚开始发送数据包时，先将拥塞窗口设置为一个最大报文段MSS的数值，每收到一个新的确认报文，就把拥塞窗口增加一个MSS，这样每经过一个传输轮次，拥塞窗口的大小就会加倍。

拥塞避免：当拥塞窗口的大小达到慢启动阙值，开始执行拥塞避免算法，拥塞窗口大小不再指数增加，而是线性增加，没经过一个传输间隔只增加1MSS。

快速重传：快速重传要求接收方收到一个失序报文段后立即发出重复确认，而不等到自己发送数据时捎带确认，快速重传算法规定，发送方只要一连收到三个重复确认就应当立即重传尚未收到的报文段，而不必继续等待设置的重传计时器超时到期。

快速恢复：当发送方连续收到三个重复确认时，就把慢开始阙值减半，将拥塞窗口设为当前阙值，执行拥塞避免算法，也有的快速恢复是将拥塞窗口设为ssthresh+3\*MSS（既然发送方收到了三个重复确认，就表明有三个分组已经离开了网络，网络中可以容纳新的三个分组）。

TCP和UDP的区别

TCP是面向连接的，UDP是无连接的

TCP是可靠的，UDP不可靠

TCP只支持点对点通信，UDP支持一对一、一对多、多对一、多对多

TCP是面向字节流的（指发送数据时以字节为单位，一个数据包可以分成若干组），UDP是面向报文的（UDP一个报文只能一次发完）

TCP有拥塞控制，UDP没有

TCP首部开销（20字节）要比UDP首部开销（8字节）大

UDP的主机不需要维持复杂的连接状态表

TCP如何保证传输的可靠性

数据包检验

对失序数据包重新排序

丢弃重复数据

应答机制：收到数据后发送确认

超时重发

流量控制

网络层的ARP协议工作原理

网络层ARP协议完成了IP地址到MAC物理地址的映射，首先，每台主机在自己的ARP缓冲区中建立一个ARP列表，标识IP和MAC地址的映射，源主机需要将一个数据包发送到目的地时，会首先检查自己ARP列表里是否存在该IP地址对应的MAC地址，如果有就直接将数据包发送到这个MAC地址，如果没有就向本地网段发起一个ARP请求的广播包，查询此目的地址对应的MAC地址，此ARP请求数据包里包括源主机的IP地址、硬件地址、以及目的主机的IP地址，网络中所有主机收到这个ARP请求后，会检查数据包中的IP地址是否和自己一样，如果一样，该主机首先将发送端的MAC地址和IP地址添加到自己的ARP列表里，如果ARP中已经存在该IP地址的信息，就将其覆盖，然后给源主机发送一个ARP相应数据包；源主机收到该响应包后，将得到的目的主机IP地址和MAC地址添加到自己的ARP列表里，并利用此信息开始数据传输。如果一直么有ARP相应，标识ARP查询失败。

RIP距离矢量路由协议（距离指跳数）

每个路由器维护一张表，记录该路由器到其他网络的“跳数”，路由器到与其直接相连的网络跳数是1，每多经过一个路由器跳数就加1，更新该表时和相邻路由器交换路由信息；路由器允许一个路径最多包含15个路由器，如果跳数为16，则不可达，交付数据报时优先选取距离最短的路径。

优缺点：

道听途说型，可能导致环路无法送达

实现简单，开销小

随着网络规模扩张，开销也会增大

最大距离为15，限制了网络的规模

当网络出现故障时，要经过较长时间才能将此信息传递到所有路由器

OSPF开放最短路径优先

向本自治系统中所有路由器发送信息，这种方法是洪泛法

发送的信息就是相邻路由器的链路状态，链路状态包括与哪些路由器相连以及链路的度量，度量用、距离、时延、带宽等来表示

只有当链路状态发生变化时，路由器才会发送信息

所有路由器以自己为根，都具有全局的拓扑结构图，并且是一致的，相比于RIP，OSPF收敛的很快

不再受跳数15的限制

LINUX硬链接与软连接的区别

硬链接（不能跨越文件系统，不能对目录进行链接）

A是B的硬链接，则A和B享有一样的inode和block，两个文件名指向同一个文件，对文件系统来说地位是平等的，每创建一个硬链接，inode链接数就加1，只有当链接数变为0时，这个文件才会被系统回收。

软链接

A是B的软链接，则A与B的inode不一样，且A的block中记录的是B的路径名，A和B是主从关系，如果删除B，A依旧存在，但指向的是一个无效的链接。

介绍一下红黑树

****Linux系统向用户提供申请内存的有brk、sbrk和mmap函数。（malloc和new的底层函数）****

函数调用步骤

参数入栈：将参数从右向左依次压入系统栈中  
返回地址入栈：将当前代码区调用指令的下一条指令地址压入栈中，供函数返回时继续执行  
代码区跳转：处理器从当前代码区跳转到被调用函数的入口处  
栈帧调整，具体包括：  
保存当前栈帧状态值，以备后面恢复本栈帧时使用。（EBP入栈）  
将当前栈帧切换到新栈帧。（将ESP值装入EBP，更新栈帧底部）  
给新栈帧分配空间。（把EBP减去所需空间的大小，抬高栈顶）

定义一个不能被继承的类

思路：通过构造函数的链式调用入手

Class A{

Friend Class B；

private：

A(){}

~A(){}

}

Class B:virtual public A{

}

此时若再有类C继承B，由于B虚继承A，C要直接调用A的私有构造函数，构造失败（如果B不是虚继承，则C先调用B，再由B调用A，构造成功）。

Epoll工作原理  
epoll是通过内核与用户空间mmap同一块内存实现的。mmap将用户空间的一块地址和内核空间的一块地址同时映射到相同的一块物理内存地址（不管是用户空间还是内核空间都是虚拟地址，最终要通过地址映射映射到物理地址），使得这块物理内存对内核和对用户均可见，减少用户态和内核态之间的数据交换。内核可以直接看到epoll监听的句柄，效率高。  
红黑树将存储epoll所监听的套接字。上面mmap出来的内存如何保存epoll所监听的套接字，必然也得有一套数据结构，epoll在实现上采用红黑树去存储所有套接字，当添加或者删除一个套接字时（epoll\_ctl），都在红黑树上去处理，红黑树本身插入和删除性能比较好，时间复杂度O(logN)。  
通过epoll\_ctl函数添加进来的事件都会被放在红黑树的某个节点内，所以，重复添加是没有用的。当把事件添加进来的时候时候会完成关键的一步，那就是该事件都会与相应的设备（网卡）驱动程序建立回调关系，当相应的事件发生后，就会调用这个回调函数，该回调函数在内核中被称为：ep\_poll\_callback,这个回调函数其实就所把这个事件添加到rdllist这个双向链表中。一旦有事件发生，epoll就会将该事件添加到双向链表中。那么当我们调用epoll\_wait时，epoll\_wait只需要检查rdlist双向链表中是否有存在注册的事件，效率非常可观。这里也需要将发生了的事件复制到用户态内存中即可。

Const和define的区别：

1. const定义的是个带类型的变量，而define定义的是个不带类型的常量
2. define在预处理阶段起作用，const在编译运行时起作用
3. define只是定义常量，没有类型检查，const有类型检查
4. Define定义的常量存在于代码段，const定义的变量存在于数据段

C++中static关键字的作用

1. 作用域隐藏。当一个工程有多个文件时，用static修饰的函数或变量只在当前文件可见
2. 全局生命周期。用static修饰的变量或函数生命周期是全局的，存储在静态数据区。
3. Static修饰的变量默认初始化为0
4. Static修饰的变量和函数是属于类的，所有对象只有一份拷贝。

介绍proactor和reactor

它们都是IO复用下的事件驱动模型，reactor是同步IO，proactor是异步IO

如何避免time\_wait状态

服务器可以设置SO\_REUSEAADR选项通知内核，如果端口忙，但TCP连接位于TIME\_WAIT状态时可以重用端口。

PVTE面试复盘：

什么场景下只能用select和poll，不能用epoll

跨平台、文件描述符不多

Weak\_ptr应用场景

解决循环引用

线程池（IO密集型和CPU密集型），如何管理线程

CPU密集型（计算密集型）：最佳线程等于CPU核数或核数+1

主要消耗CPU资源，此时要尽可能避免线程之间的切换

IO密集型：最佳线程数=（线程等待时间+线程CPU时间）/（线程CPU时间\*CPU数目）

CPU消耗少，大部分时间都在等待IO操作，此时任务越多，CPU效率越高

Lamaba表达式捕获种类

值捕获和引用捕获

判断链表有环最多有几种方法

emplace\_back和push\_back有何区别

Empalce\_back是原地构造，不涉及具体的内存移动和拷贝

Class A

{

int \_a=0;

int a1()

{

Return \_a;

}

int a2()

{

Return 0;

}

}

Int main()

{

A\* p=nullptr;

}

Main函数执行过程

子类的虚函数表里有啥，父类和子类是同一张虚函数表吗

https的连接过程

1. 客户端向服务端发送请求，同时发送客户端支持的加密规则（包括对称加密、非对称加密、摘要算法）
2. 服务器从中选出一组加密算法和HASH算法，将自己的身份信息以证书的形式发回给浏览器，证书里包含网站地址、加密公钥（非对称加密），以及证书的颁发机构等信息
3. 客户端验证服务器的合法性，包括：证书是否过期，CA是否可靠，公钥能否解开证书的数字签名
4. 浏览器生成随机密钥（对称加密），并用服务器提供的公钥加密（采用非对称加密进行加密），使用hash算法对握手消息进行摘要计算，并对摘要使用对称算法加密，将加密后的随机密钥和摘要一起发给服务器
5. 服务器使用自己的私钥解密，得到对称加密的秘钥，用这个秘钥解出hash摘要，验证握手信息是否可靠，使用对称加密的秘钥加密握手消息发给浏览器
6. 浏览器解密并验证摘要，若一致握手结束，之后的数据传送都采用对称加密

GET和POST的区别

1. GET是幂等的，即使读取同一个资源，总是得到相同的数据，POST不是幂等的
2. GET一般用于从服务器上获取资源，而POST有可能改变服务器上的资源
3. GET将请求的数据附在URL之后，POST则附于请求体中
4. GET请求可以被缓存，POST不可缓存
5. GET只允许ASCII字符，POST对数据类型没有要求，也允许二进制数据
6. GET的长度有限制，POST数据大小无限制

C++值初始化列表与构造函数的区别

初始化列表：

所有类非静态数据成员都可以在这里初始化

静态成员不可在此初始化

构造函数：

Const成员不可在此初始化

引用型成员不可在此初始化

没有默认构造函数的成员不能在此初始化

类静态数据成员可在此修改，但必须在外部初始化

InnoDB和MyISAM之间区别：

1. InnoDB支持事务，MyISAM不支持事务，Innodb把每一条SQL语言都默认转化为事务，也可以用begin和commit提高效率
2. InnoDB支持外键，MyISAM不支持
3. InnoDB是聚集索引，使用B+树作为索引结构，数据文件和主键索引绑在一起，通过主键索引效率很高，但辅助索引需要两次查询。MYISAM是非聚集索引，主键索引和辅助索引的叶子节点都是文件的地址
4. InnoDB支持表、行级锁(默认)，而MyISAM支持表级锁
5. InnoDB必须有主键(用户未指定的话，会自己选一个)，MYIsam可以没有
6. InnoDB不保持表的具体行数，MYISAM会用变量保存具体行数。