static:

①修饰普通变量，修改变量存储区域和生命周期，使变量存储在静态区，在main运行前就分配了空间，有用初始值或者默认值初始化

②修饰普通函数，表示函数作用范围，仅在文件内有效。

③修饰成员变量，使所有对象只保存一个变量，不需要生成对象就可以访问该成员。

④修饰成员函数，使不需要生成对象就可以访问该函数，static函数内不能访问非静态成员

this指针 → 指代当前对象地址

①隐含在每个非静态成员函数中

②调用成员函数时，先将对象地址赋给this指针，然后调用成员函数，每次成员函数存取数据成员时，隐含使用this.

③成员函数被调用时自动传递一个隐藏参数。

④this被隐式声明为 className *const this，是一个底层const，不可以被赋值；在className类的const成员函数中，this的类型为 const className *const，说明this指针指向的对象是不可修改的。

⑤this是右值，不能 &this.

const

① 顶层 const → 指向对象本身

② 底层 const → 指针指向的对象是一个常量

const对对象类型加以限定，const对象一旦创建，其值不改变。

const的引用（可引用非const对象）

指针常量（指向常量的指针）：int *const p = &n（不可改指向）.

常量指针：const int *p = &n（可以改指向）.

常量指针常量：const int *const p3 = p2.

①修饰变量，说明变量不可以被改变

②修饰指针，分为指向常量的指针和指针常量

③常量引用，避免拷贝，也避免修改

④修饰成员函数，不能改成员变量

inline内联函数

①相当于把内联函数里内容写在调用内联函数处。

②不用执行进入函数步骤，直接执行函数体。

③相当于宏，但比宏多了类型检查，真正有函数特性。

④不能有循环、递归，switch等复杂操作。

⑤在类中定义的函数，虚函数之外的其他函数会自动隐式地当成内联函数。

C++11

1. nullptr

2. auto, decltype (int ci=0, decltype(ci) x=3.

3. 范围for

4. 智能指针

5. 右值引用，移动构造函数.

虚函数：

①只需要在声明函数的类体中使用virtual，定义函数不需要virtual.

②虚函数是基类对象的指针指向不同对象，根据不同对象，来调用虚函数

③虚析构函数是先调用派生类析构，再调用基类析构

主要是通过基类指针访问派生类定义的函数，每个有虚函数的类，有一个虚函数表指针.

红黑树

①要么是红，要么是黑

②根结点为黑

③叶结点为黑

④红节点儿子全黑

⑤牺牲了严格高度平衡的代价，只要达到部分平衡，降低了对旋转的要求.

运输层

1. 运输层提供进程和进程之间的逻辑通信
2. 复用和分用（复用：不同的进程可以使用同一个运输层协
3. 运输层对收到的报文进行差错检测。（纠错传输复杂
4. 传输层的TCP与UDP。

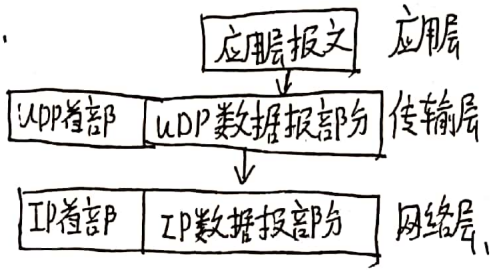
分用是指接收方的运输层在剥去首部之后能把这些数据正确交付应用进程。

TCP：确认，流量控制，计时器及连接管理等。
UDP：比IP数据报，加了复用分用和差错检测。
复用：从传输层 → 网络层
分用：网络层 → 具体的进程
端口：是传输层的SAP，标志主机中的应用进程。

端口号
- 服务端使用的端口号
  - 熟知端口号（0~1023）
  - 登记端口号（1024~49151）
- 客户端使用的端口号：在客户进程运行时动态选择。

Socket =（主机IP地址，端口号）。

$2^{16} = 65535$。



UDP一边发一个完整的报文。

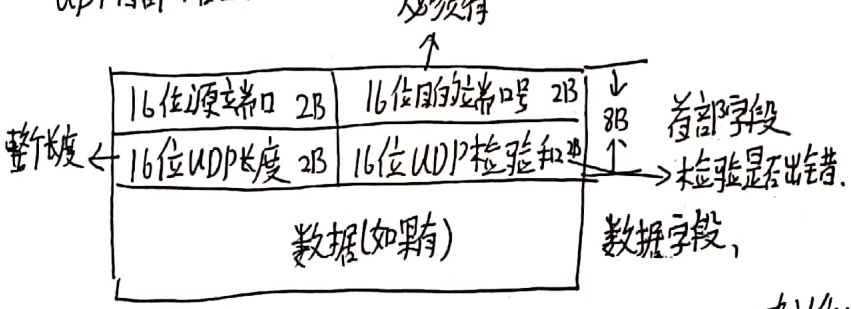
UDP：
- 无连接
- 不保证可靠交付
- 面向报文
- UDP无拥塞控制
- 首部开销小（8字节，TCP有20字节）

UDP：16点。

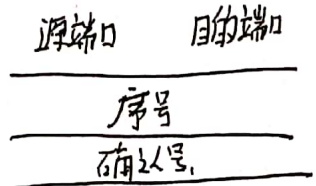
TCP
- 面向连接
- TCP连接只能点对点，无法广播或多播。
- 可靠有序，不丢不重。
- TCP提供全双工通信。→ 发送缓存、接收缓存。
- 面向字节流，无结构字节流。

TCP窗口表示 A ← B（证明自己窗口大小）

UDP首部格式：



UDP用户数据报中检验和的计算方法比较复杂，主要加12字节的伪首部，不是UDP数据报真正的首部，只在计算和时临时加上。



确认位：
- URG：紧急位。
- ACK：确认号，为1时有效。确认号是ack=x+1，
- PSH：尽快交付应用进程。→ 交付上层进程。
- RST：出现差错，重新建立连接。

同步位
- SYN：=1时，是一个连接请求。
- FIN：数据已发完，=1。

epoll_create(int size)
size：epoll大小.

epoll_ctl：
① int epfd：注册epoll例程文件描述符
② int op：用于指定对epoll例程文件描述符添加、删除、更改等操作
③ int fd：需注册的监视对象文件描述符.
④ epoll_event *event：监视对象事件类型.

EPOLL_CTL_ADD：将文件描述符注册到epoll例程.
EPOLL_CTL_DEL：从epoll例程删除文件描述符
EPOLL_CTL_MOD：更改注册的文件描述符发生的情况.

epoll_wait：
① int epfd：epoll例程文件描述符
② events：发生事件的文件描述符集合结构体地址
③ maxevents：第二个参数可以保存最大事件数.
④ timeout：以 1/1000 s为单位的等待时间.

多个触发和边缘触发：
① 条件触发(LT)，只要输入缓冲中有数据，就会一直通知该事件.
② select是条件触发，条件触发(默认)二水平触发.
③ 边缘触发只发送一次，可以为离接收数据和处理数据的时间点.

多线程：共享数据区和堆，栈独立.
多进程：数据区，堆，栈相互独立

pthread_create：
① thread：ID变量地址值
② attr：传递线程参数
③ start_routine：线程main函数的函数指针
④ arg：传参.
编译时要加：-lpthread.

pthread_join(thread, status)
① thread：该ID的线程终止后才返回
② status：保存线程的main返回地址的指.
会阻塞，直到线程结束.

线程安全：多个线程同时调用同一函数.
临界区：多个线程共用的代码.
根据临界区是否引发问题：
① 线程安全函数
② 非线程安全函数.
解决线程安全的办法：线程同步
同步的特点：
① 同时访问同一内存空间的情况
② 需要指定访问同一内存空间的线程执行顺序的情况.

互斥量
创建：pthread_mutex_init
销毁：pthread_mutex_destory
加锁：pthread_mutex_lock
解锁：pthread_mutex_unlock
分别为在临界区代码的开始与结束.
线程退出临界区时，如果没有调用unlock，那么其他为了进入临界区而调用lock函数的线程就无法摆脱阻塞状态，这就是死锁.
多次使用lock和unlock会消耗时间.

信号量：用"二进制信号量"完成"控制线程顺序"为中心的同步方法。
创建：sem_init
销毁：sem_destory
信号量值+1：sem_post
-1：sem_wait.
信号量值不能小于0，在为0时用wait会阻塞
通过这种特性完成临界区同步.

线程销毁：
① pthread_join：等待终止，会引发销毁，但会阻塞
② pthread_detach：不引起终止或进入阻塞.

TCP主要比UDP多了流控制，TCP是面向字节流的。
UDP面向报文，把应用层数据原封不动加首部直接传向网络层。

TCP		UDP	
server	client	server	client
①socket ②bind ③listen ④accept ⑤read/write	①socket ②connect ③write/write	①socket ②bind ③recvform ④sendto	①socket ②sendto ③recvfrom

半关闭：int shutdown(int sock, int howto)
SHUT_RD：断开输入流
SHUT_WR：断开输出流
SHUT_RDWR：都断开。

实现并发服务器三种方法：
①多进程：通过fork，每次来请求就用子进程处理。
②多I/O复用
③多线程。

fork函数：父子进程有完全相同的内存结构。
父进程：返回子进程ID。
子进程：fork返回0。

僵尸进程：进程完成之后，没有被回收，父进程没有回收子进程，如果父进程退出，init会自动回收。
孤儿进程：父进程完成或终止，但子进程仍然运行。会被init进行收养。

防止僵尸进程：
①利用wait函数，阻塞程序直到有子进程终止。
②用waitpid，防止程序阻塞

信号处理：
信号是特定事件发生时，向操作系统发送的消息。
①signal，两个参数，信号类型，函数指针。
发生信号时会唤醒由于sleep进入阻塞的进程。
②sigaction（更通用）
int sigaction(int signo, const struct sigaction *act, struct sigact)
信号处理器。
信号处理技术也可以处理僵尸进程。
在多进程中，可以分割I/O事使程序不阻塞。

select：
①maxfd：监视的文件描述符数量
②readset：是否待读取又注册到fd_set
③write_set：是否可传输无阻塞
④exceptset：是否发生异常注册。
⑤timeout：超时时间
发生变化时，fdset其他位0，发生的为1。
每次提前存一下temps。

进程间通信：
①通过管道(PIPE)。

I/O复用，多进程会浪费资源。
通过select，移植性好。
步骤：①设置文件描述符
②指定监视范围
③设置超时
④调用select
⑤查看调用结果。

优于select的epoll。
select慢的原因：
①调用select后针对所有文件描述符的循环语句。
②每次调用select要传递监视对象信息。
快点：①数有少②程序位置有变化。
epoll无需编写以监视状态变化为目的的针对描述符的语句。
epoll_create：创建保存epoll文件描述符的空间。
epoll_ctl：向空间注册并注销监视文件描述符
epoll_wait：等待文件描述符变化。

UDP：	TCP
①无连接 ②尽最大努力交付 ③面向报文 ④无拥塞控制 ⑤UDP支持1对1,1对多,多对多 ⑥首部小 8字节	①有连接 ②点对点 ③全双工通信 ④面向字节流 ⑤可靠有序,不丢不重

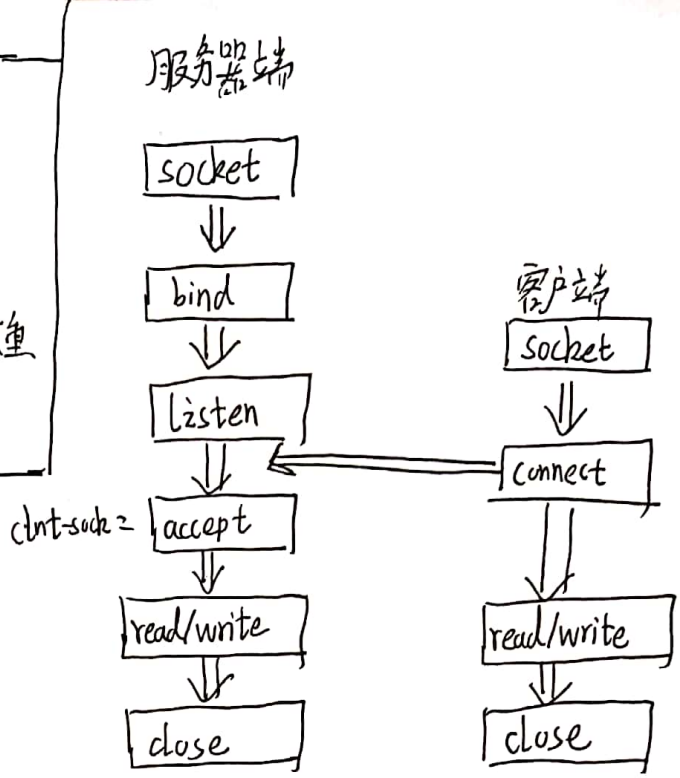
TCP实现可靠传输的机制：

①校验

②序号

③确认

④重传（通过标志位）.



智能指针：

1. auto_ptr → 弃用
2. shared_ptr ：多个共享
3. weak_ptr ：可复制shared_ptr，但构造或释放对资源不影响.
4. unique_ptr ：不允许多个指针共享资源

无法复制构造，无法使2个unique_ptr 指向同一对象.

移动赋值.

top-k：分治+trie

hash+小顶堆.

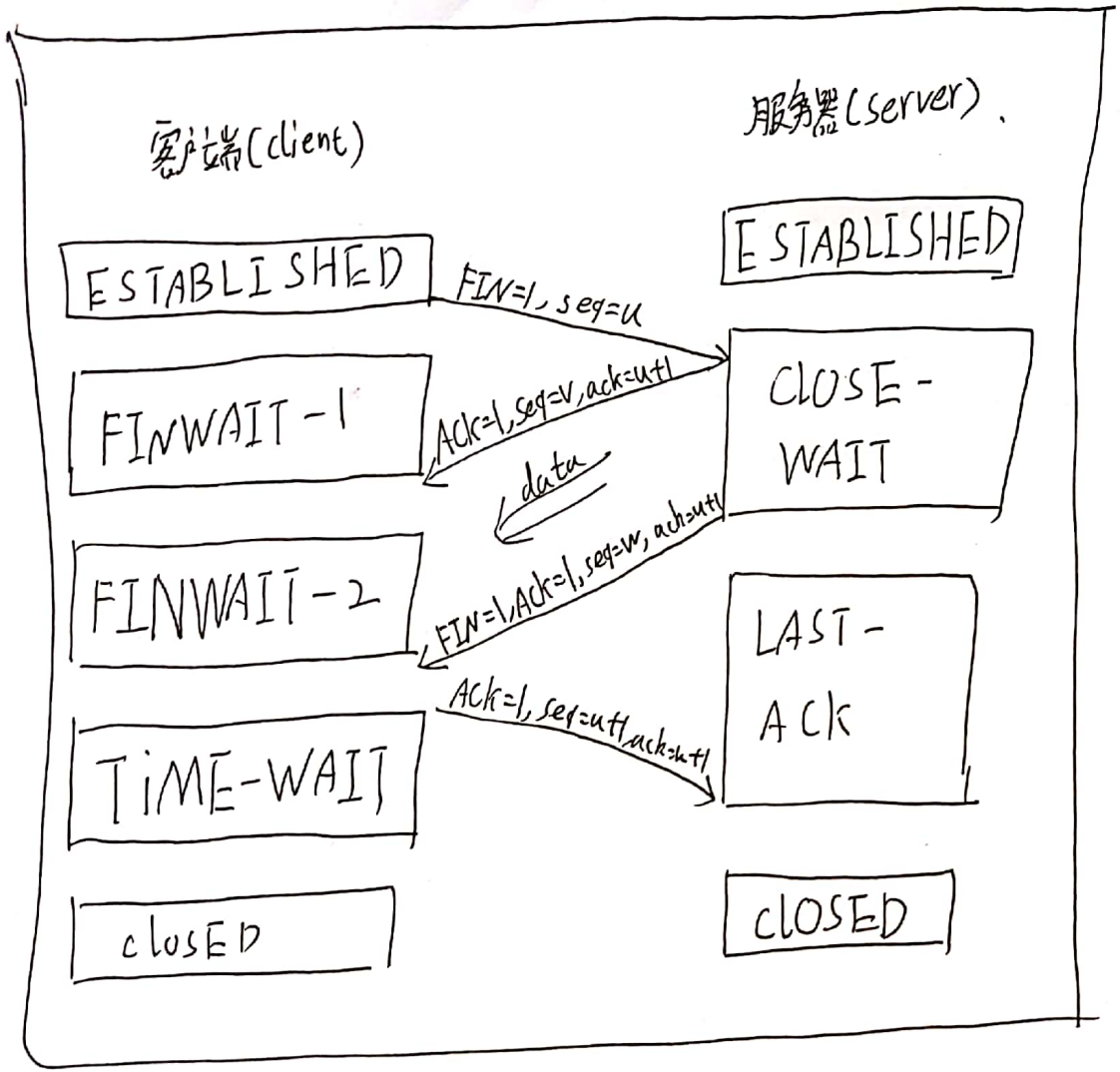
快排：$2N\log N - O(N\log N\log N)$

一，分解成十数据集，统计词频，

二，局部淘汰

三，分治，

四：hash.



① client的u为之前的seq+1.

② B收到连接释放报文段后发出确认，确认号是ack=u+1. seq为v，为之前的值+1. B进入半关闭等待状态.

③ TCP连接此时处于半关闭，A已知没有数据要发送了，但B要发送，A仍然会接收. A收到B的确认后，就进入FIN-WAIT-2状态，等待B发出的连接释放报文段.

④ 若B已经没有要发送的数据，~~其应用进程~~ B发出释放报文段使FIN=1，现在B假定序号为w，B必须重复上次已发送的确认号ack=u+1，B进入LAST-ACK状态，等待A的确认.

⑤ A对于B发出的连接表示确认.