# select/poll/epoll

select

设置一个最大数目，比如有100个文件描述符，扫描一遍，查看是否有等待事件发生

每次调用select都要把fd集合从用户态拷贝到内核态，开销大

支持文件描述符数量小

详细过程：

int select(int nfds, fd\_set \*readfds, fd\_set \*writefds, fd\_set \*exceptfds, struct timeval \*timeout);

第一个参数设置fd\_set的最大文件描述符加一

第二三四个参数表示需要关注的读，写，错误的文件描述符集合，

第四个参数设置超时

Select 对应到内核的sys\_select调用，sys\_select首先把第二三四个参数指向的fd\_set拷贝到内核，然后对每个被set的描述符进行poll操作，poll会返回当前设备fd的状态（比如是否可读可写），如果由事件发送，则记录在fd\_set，并返回，如果没有，select会将睡眠到超时，睡眠结束后再来一次轮询，然后返回

最后只要检查关注的描述符是否被set

Poll

不用设置最大数目，且文件描述符不同

int poll(struct pollfd \*fds, nfds\_t nfds, int timeout);

第一个参数传递要检测的包含文件描述以及事件的结构

第二个参数是fds的数量

打三个参数超时

与select不同，poll把每个文件描述符以及事件都放在了fds，检测后不会被清除，只是更改发生的事件成员revents

变量，因此fds只要初始化一次，也没了最大检测限制，更高效

Select/poll都需要自己不断查找所有fd集合，直到设备就绪，

Epoll（底层红黑树和链表）

只是在注册新事件才把fd拷贝到内核，不用像他们每次调用函数都拷贝

添加事件后，与设备驱动程序建立回掉关系，当事件发生，会调用回掉方法，会将发生的事件fd添加到双链表

只是访问就绪链表

 使用mmap加速内核与用户空间的消息传递

# Slt：

Vector 数组 支持快速访问

List 双向链表 支持快速增删改查

Deque deque 的最大任务，便是在这些分段的定量连续空间上，维护其整体连续的假象，·并提供随机存取的接口。避开了“重新配置，复制，释放”的轮回

1、如果你需要高效的随即存取，而不在乎插入和删除的效率，使用 vector；  
2、如果你需要大量的插入和删除，而不关心随即存取，则应使用 list；

3、如果你需要随即存取，而且关心两端数据的插入和删除，则应使用deque。

Stack list/deque

Queue list/deque

# 指针数组，数组指针，指针函数，函数指针

指针数组：存放指针的数组int \*a[10]

数组指针：指向数组的指针 int (\*a)[10];

指针函数：返回值为指针的函数int\* a();

函数指针：指向函数的指针(int(\*f)(int a))

# 红黑树：

一种平衡检索树，并不是平衡二叉树，他放宽了平衡二叉树的某些要求，由于一定限度的不平衡，才得到了性能的提升

应用于stl的 set,mutiset,map,multimap等

平均高度log(N),最坏不会超过2log(N);

能以o(log2(N))的时间复杂度搜索，插入删除，任何不平衡都可以三次内解决

满足五个性质

1:节点是红色或者黑色

2:根节点是黑色  
 3 : 每个叶子节点（NULL或者空节点）是黑色

4：每个红色节点的两个子节点都是黑色的（不存在两个连续的红色节点）

5：从任意节点到其每个叶子结点的所有路径都包含相同数目的黑色节点

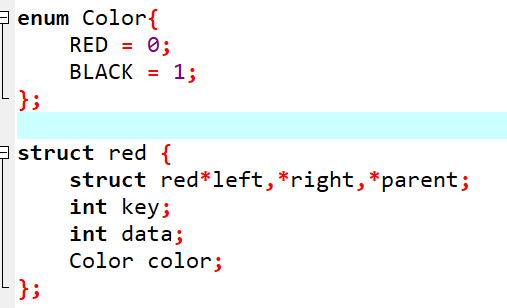
<http://blog.csdn.net/sun_tttt/article/details/65445754>

红黑树面试常考问题

（1）：stl的set底层用的什么数据结构

红黑树

（2）：红黑树的数据结构怎么定义的



（3）：红黑树有哪些性质

满足五个性质

（4）：红黑树的各种操作的时间复杂度

能保证在最坏情况下，基本的动态几何操作时间均为O(lgN)

（5）：红黑树相比BST和AVL树有什么有点

红黑树是牺牲了严格的高度平衡的优越条件为代价，它只是要求部分的达到平衡要求，降低了对旋转的要求，从而提高了性能，能以log2N的时间复杂度进行搜索，插入，删除操作，此外，由于他的设计，任何不平衡都可以在三次旋转内解决

相比于BST，因为红黑树可以确保树的最长路径不会超过两倍最短路径的长度，所以查找效果是有最低保证的，最坏情况也可以保证logN。而BST最坏情况可达N

相比AVL，时间复杂度相同，但是统计性能比AVL树高，插入等操作节约时间

（6）：红黑树相比于哈希表，在选择时有什么依据

权衡三个因素，查找速度，数据量，内存使用，可扩展性

Hash查找速度更快，但是消耗内存，当数据特别多，构造速度也变慢

# C++有了malloc/free，为什么还要new/delete

1:malloc/free是标准库函数，而new/delete是运算符

2.对于非内部数据类型的对象而言，光是malloc/free无法满足动态对象的要求

对象在创建的同时，自动执行构造函数，对象在消亡之前自动执行析构函数

由于malloc/free是库函数，而不是运算符，不在编译器控制权限之内，不能把执行构造函数的任务强加于malloc/free

3:new/delete是运算符，不是库函数

New的过程：获得一块内存空间，调用构造函数，返回一个正确的指针，如果创建的是内部数据类型变量，则没有第二步

而malloc没有调用构造函数这一步。所以，初始化得单独调用

A\* pa = new A(3);

相当于

A\* pa = (A\*)malloc(sizeof(A));

pa->A::A(3);

return pa;

解释：

内部数据对象，如int ,double

非内部数据对象：自定义的对象

库函数为什么不在编译器控制权限之内：库函数是已经编译的代码，编译器不会编译检查，由连接器直接同用户写的代码合成exe文件.

# 线程有几种状态

新生状态，可运行状态，被阻塞状态，死亡状态

# **进程间几种通信方式，及其优缺点**

1. **管道：**
   1. **无名管道局限于单向通信方式，且只能在创建它的进程和其子进程之间实现管道的共享**
   2. **有名管道可以给任意关系的进程使用，但是由于长期存储在系统中，使用不当容易出错**
2. **消息：**
   1. **允许任意进程通过共享消息队列来时间进程间的通信，**

# **C++ public/private/protected继承**



# ****C++如果构造函数定义为private/protected。有什么方法可以调用？****

**如果是protected ，只需在用它派生public一个派生类。派生类也可以实例化，因为，对于protected函数，子类的函数可以调用**

**如果是protected，可以定义一个public static函数，这个函数内部可以实例化该类，这样，这个函数就的调用就不需要提前实例化一个对象，而可以直接被调用，而它又属于类的内部函数，也就可以调用构造函数了**

# **C++友元函数：**

**为该类提供一个外部访问的接口**

**这个友元函数不属于该类的成员函数，他是定义在类外的普通函数，只是这个函数可以访问类的成员变量**

**使用友元函数注意四点：**

# **内联函数和宏的区别**

**内联函数在编译时展开，而宏在预编译时展开**

**在编译时，内联函数可以直接镶嵌在目标代码中，而宏只是简单的文本替换**

**内联函数可以完成类型检测，语句是否能正确编译等功能，宏不具备这种功能**

**内联函数是函数。而宏不是函数**

**内联函数传参不会出现二义性，而宏如果括号没写好，容易出现错误**

**指针和引用的区别**

**初始化要求不同，引用要在创建的同时初始化，而指针可以暂时不用初始化**

**可修改性不同，引用一旦初始化指向一个对象，他就不能被改变成另一个对象的引用，而指针可以**

**不存在NULL引用，引用不能指向空值**

**测试需要的区别，引用不用检测是否为NULL,而指针需要检测----所以传引用比传指针安全**

# **内联函数和宏的区别**

**内联函数在编译时展开，而宏在预编译时展开**

**在编译时，内联函数可以直接镶嵌在目标代码中，而宏只是简单的文本替换**

**内联函数可以完成类型检测，语句是否能正确编译等功能，宏不具备这种功能**

**内联函数是函数。而宏不是函数**

**内联函数传参不会出现二义性，而宏如果括号没写好，容易出现错误**

# **Main函数执行前会执行什么代码**

**全局对象的构造函数**

# 复制构造函数（拷贝构造函数）是什么。什么是深复制和浅复制

复制构造函数是基于同一类的其他对象的构建及初始化

C++下，一般三种情况对象需要复制构造函数

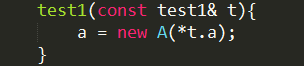
1. 一个对象以传值的方式进入函数体
2. **一个对象以传递的当时从函数体返回**
3. **一个对象需要通过另一个对象进行初始化**

**浅复制：（被拷贝对象A，拷贝对象B）**

**只被复制函数一点不改变被复制，如果A的某个成员变量是指针，指向某个区域，浅拷贝后，如果B被释放，那么指向的这个区域也会被释放，对A来说，就会产生错误**

**深拷贝：**

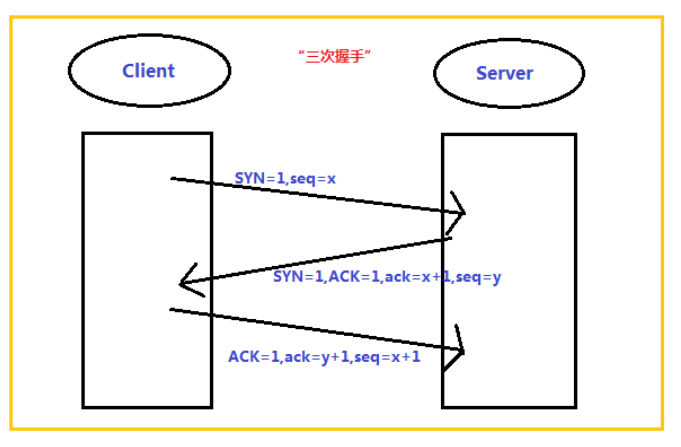
**针对浅拷贝的问题，深拷贝对拷贝构造函数进行重写**



# **复制构造函数和赋值函数的区别**

1. **复制构造函数是一个对象来初始化一块内存区域，这块内存就是新对象的内存区，而赋值函数是对一个已经被初始化的对象进行operator=操作**
2. **。。。**

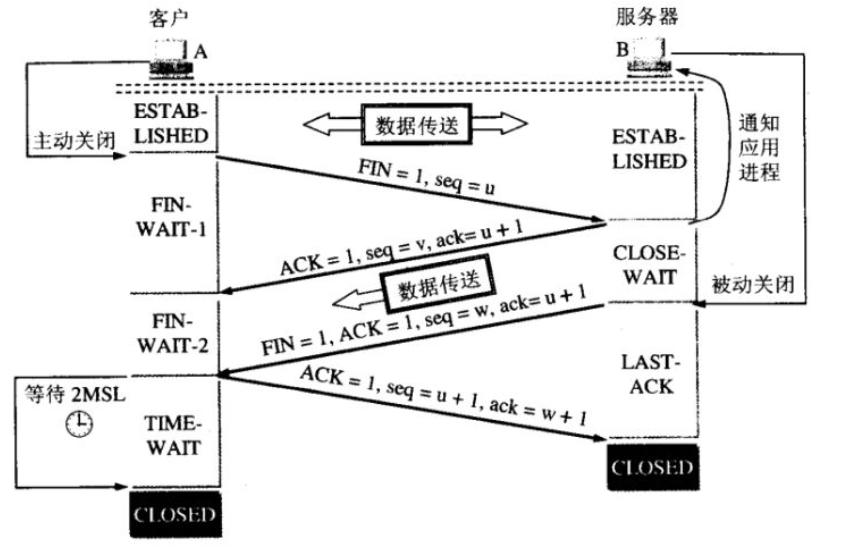
# **Tcp/ip三次握手**



# **TCP/IP为什么要三次握手，而不是两次或者四次.设A为客户端B为服务器**

**为了解决“网络中存在的延迟的重复分组”。A发出的第一个请求报文段并没有丢失，而是在某个网络站点长时间滞留了，A请求超时后又重写发出了请求，而先建立了连接，而此时之前滞留的分组也到达B，原本这个分组已经失效，但是，B却认为又是一次连接，。如果采用二次握手，这次连接又会被建立，资源会被白白浪费。采用三次握手，B又发出一个请求给A，此时A不会给B回复，自然无法建立第二个连接了**

# **Tcp/ip四次挥手**



# 为什么A要先进入TIME-WAIT状态，等待2ms才进入CLOSED状态

为了保证B能收到A的确认应答

如果A发完确认应答后直接进入CLOSED状态，如果这个分组丢失，B等待超时后就会重新发送连接释放亲求，但此时A已经关闭了，不会做出任何响应因此，B永远无法正确关闭

# TCP如何保证可靠传输 C:\Users\wangh\AppData\Roaming\Tencent\Users\1603635122\QQ\WinTemp\RichOle\%E[L1~(V7S9E8PJOZK9Q%NM.png

# 互斥锁，自旋锁和读写锁

锁是一种为了解决某种临界资源互斥而使用的一种机制。

自旋锁和互斥锁功能差不多，每一时刻只能有一个执行单元占有锁，而占有锁的单元才能获得访问临界资源的所有权，从而达到互斥的目的

自旋锁和互斥锁的区别：

自旋锁在请求锁时，如果发现锁被占用，则不停的循环判断锁的状态，直到锁被释放，期间并不阻塞自己，它会一直占有cpu

互斥锁在等待锁被释放时，会把自己阻塞并放入队列种。当锁被释放，会唤醒它，阻塞会耗时，但是，不会消耗CPU

**读写锁实际是一种特殊的自旋锁。把访问者分为读者和写者，允许多个读者，或者一个写着**

**Linux内存管理机制**

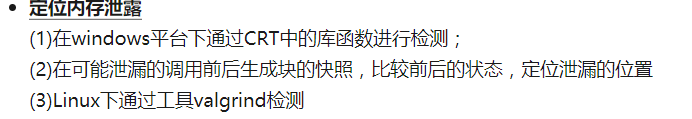
# ****必须使用类的初始化成员列表初始化的三种情况****

1. **如果成员变量也是一个类，且这个类没有无参数构造函数**
2. **成员变量时const修饰或者是引用。这两个都需要在创建的同时初始化，后面无法赋值**
3. **子类初始化父类的静态成员变量**

**使用初始化成员列表效率高**

**不使用，成员变量先初始化，在赋值。效率低**

# ****定位内存泄露****



多进程和多线程的选择**（安全，消耗资源）**

**使用多线程：**

**需要更可靠**

**使用多线程：（共用存储区，通信方便，节约储存，不安全）**

**需要频繁创建销毁**

**需要大量计算**

**C++不支持哪些函数为虚函数**

**构造函数，内联函数，友元函数，静态函数**

1:数组和链表的区别

2:STL中有哪些容器，它们之间的区别

标准STL序列容器：vector、string、deque和list

标准STL关联容器：set、multiset、map和multimap

非标准序列容器slist和rope

3:vector和list的区别

vector类似于数组，能高效的进行随机存取，时间复杂度为o(1);

list类似于链表，只能通过指针访问数据，所以list的随机存取非常没有效率，时间复杂度为o(n);

所以vector的迭代器可以+1，而list不可以.

总之，如果需要高效的随机存取，而不在乎插入和删除的效率，使用vector;  
 如果需要大量的插入和删除，而不关心随机存取，则应使用list。

Vector 的capacity保存的就是 可以存放的元素个数

4:vector的push\_back实现原理

vector是一段连续的内存空间。start，finish，end\_of\_storage三个指针描述了空间状态，这三个是普通的指针。start到finish是已经使用的内存，里面有元素。finish到end\_of\_storage是未使用的内存，里面没有元素。

vector有预存的内存（capacity），如果存入的元素大于了capacity，就重新分配一个比原来capacity大两倍的内存

5:map和hash\_map的区别

STL的map底层是用红黑树实现的，查找时间复杂度是log(n)；

STL的hash\_map底层是用hash表存储的，查询时间复杂度是O(1)；

一般情况下，如果记录非常大，考虑hash\_map，查找效率会高很多，如果要考虑内存消耗，则要谨慎使用hash\_map。

6:分析线性表、二叉平衡树和哈希表存储数据时各自的优劣。

链式：优点：插入和删除不需要移动，空间有效利用缺点：大量访问操作时不如顺序存储结构。

顺序：优点：可随机存取表中任一元素。缺点：插入或删除操作时，需大量移动元素。

二叉平衡数存储，查找O(lgn), 插入O(lgn);

优点在平衡树中按序查找方便， 缺点二叉平衡树的实现代价比较高。

哈希表存储，查找O(1),插入O(1)。    缺点：不能显示序的信息，不能找出最大最小值。

二叉平衡树O(lgn)比哈希表O(1)不会大太多，若储存的数有序的要求，用二叉平衡树比哈希要好

7:B树、存储模型（B树、B+树与数据库引擎这块是热门问题）

B树是为了提高磁盘或外部存储设备查找效率而产生的一种多路平衡查找树。

8:linux  select poll epoll

9:哈希表时间复杂度如何?冲突了如何解?

构造哈希函数的原则是：①函数本身便于计算；②计算出来的地址分布均匀，即对任一关键字 k，f(k) 对应不同地址的概率相等，目的是尽可能减少冲突。

冲突解决方式

1.开放定址法

2.再哈希法

3.链地址法

4.建立公共溢出区

10:Struct 和class 的区别。

C++中两者的唯一区别在于：Class定义中默认情况下的成员都是私有的，而结构Struct定义中默认情况下的成员都是公有的。

在C中，结构中不能有成员函数，而在C++中可以有成员函数.

11:函数调用过程

在函数调用时，第一个进栈的是主函数中后的下一条指令（函数调用语句的下一条可执行语句）的地址，然后是函数的各个参数，在大多数的C编译器中，参数是由右往左入栈的，然后是函数中的局部变量。

12:ICMP是什么协议,处于哪一层?

Internet控制报文协议，处于网络层（IP层）

13:触发器怎么工作的?

触发器主要是通过事件进行触发而被执行的，当对某一表进行诸如UPDATE、 INSERT  
  
、 DELETE 这些操作时，数据库就会自动执行触发器所定义的SQL 语句，从而确保对数  
  
据的处理必须符合由这些SQL 语句所定义的规则。

14:动态连接库的两种方式

第一种是静态链接方式，在这种方式下，[动态链接库](https://www.baidu.com/s?wd=动态链接库&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1dBuWNhnhD4mhD4nH6suWmd0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3En1c4nWTLPHn4rjn3PHTzrjfd)中的所有数据代码都将拷贝到调用程序的[代码空间](https://www.baidu.com/s?wd=代码空间&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1dBuWNhnhD4mhD4nH6suWmd0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3En1c4nWTLPHn4rjn3PHTzrjfd)中去，此时它和调用程序本身的函数没有什么区别；

　　第二种是动态链接方式，在这种方式下，[动态链接库](https://www.baidu.com/s?wd=动态链接库&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1dBuWNhnhD4mhD4nH6suWmd0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3En1c4nWTLPHn4rjn3PHTzrjfd)中的数据代码是在需要的时候才拷贝到内存中去的；

15:IP组播有那些好处?

Internet上产生的许多新的应用，特别是高带宽的多媒体应用，带来了带宽的急剧  
  
消耗和网络拥挤问题。组播是一种允许一个或多个发送者（组播源）发送单一的数据包  
  
到多个接收者（一次的，同时的）的网络技术。组播可以大大的节省网络带宽，因为无  
  
论有多少个目标地址，在整个网络的任何一条链路上只传送单一的数据包。所以说组播  
  
技术的核心就是针对如何节约网络资源的前提下保证服务质量。

16:.列出3个常用网络协议使用的端口。  
 HTTP协议用80端口，FTP协议用21端口，POP3协议用110端口

17:**HTTP返回码中301与302的区别**

**18:**．请问交换机和路由器各自的实现原理是什么**?**分别在哪个层次上面实现的**?**

**交换机属于ＯＳＩ第二层即数据链路层设备。它根据ＭＡＣ地址寻址，通过站表选择路由，站表的建立和维护由交换机自动进行。路由器属于ＯＳＩ第三层即网络层设备，它根据ＩＰ地址进行寻址，通过路由表路由协议产生。交换机最大的好处是快速，路由器最大的好处是控制能力强。**

19:ASSERT()是干什么用的

ASSERT() 是一个调试程序时经常使用的宏，在程序运行时它计算括号内的表达式，如果表达式为FALSE (0), 程序将报告错误，并终止执行。如果表达式不为0，则继续执行后面的语句。这个宏通常原来判断程序中是否出现了明显非法的数据，如果出现了终止程序以免导致 严重后果，同时也便于查找错误。例如，变量n在程序中不应该为0，如果为0可能导致错误，你可以这样写程序：  
……  
ASSERT( n != 0);  
k = 10/ n;  
……  
ASSERT只有在Debug版本中才有效，如果编译为Release版本则被忽略。  
assert()的功能类似，它是ANSI C标准中规定的函数，它与ASSERT的一个重要区别是可以用在Release版本中。

20:**malloc之后再进行free，free的内存空间一定被OS回收了吗？**

**os管理是对页进行分配和回收，如果free的内存只是某个页的某小块内存，而这个页的其他内存依旧被使用，这时os就不会回收内存空间。**

**21:windows内存管理方式及其优缺点**

**页式存储**

**优点：没有外碎片，每个碎片不超过业的大小**

**缺点：程序全部装入内存，要求有相应的硬件支持。**

**段式存储**

**优点：可以分别编写和编译，可以针对不同类型的段采用不同的保护，可以按照段为单位进行 共享。（信息共享，动态链接，方便编写，信息保护）**

**缺点：会产生碎片**

**段页式存储：**

**具有两种优点**