北邮人论坛：[小孩\_harry]

虚函数与多态的使用及原理、STL的内部实现、智能指针的使用及原理、引用和指针的区别、static关键字的用法、设计模式（单例、工厂）及代码实现、内存管理（如new和malloc的区别）、内存分布（堆内存栈内存区别）、多线程的实现方式、判断机器是大端存储还是小端存储

1. 虚函数与多态的使用及原理

类内成员函数名前加virtual；2.类外函数体前不需要写；3.构造函数和静态函数不能是虚函数。

有两种实现形式：基类指针(派生类的指针可以赋值给基类指针；基类指针指向基类和派生类中同名的虚函数时，指向基类或者派生类对象就调用对应的虚函数)

基类引用（派生类的对象可以赋给基类引用；基类引用调用基类和派生类中的同名的虚函数时，引用引用的是基类或者派生类的对象，就调用对应的虚函数）

作用：增强程序的可扩充性，即在程序需要修改或增加功能时需要改动的代码较少。

实现：在种类的对象中添加一个指向虚函数表的指针的地址，根据地址找到虚函数表，然后在找到对应的虚函数。

包含纯虚函数的类叫抽象类，可以作为基类实例化，但不能单独实例化。

1. STL的内部实现

顺序容器：vector(可变大小数组), list（双向链-表）,deque（双端队列）, 关联容器：map（关联数组）, set（关键字即值）, unordered\_map（哈希函数实现的）, unordered\_set 容器适配器：queue, priority\_queue, stack

1. 智能指针的使用及原理（参考c++ primer）

背景：程序中使用三种内存：静态内存，栈内存，以及堆（heap）, 前两者分别用来保存局部stastic 对象和非stastic对象，生存期为[stastic对象使用前和程序运行时->程序运行结束]，而堆内存用来存储动态分配的对象（程序运行时分配的对象），生存期由程序控制，需在使用结束时，显式地销毁他们。为了防止内存泄露，C++ 11中引入了share\_ptr, unique\_ptr, weak\_ptr, 头文件为Memory。

智能指针实质是一种模板类，初始化可以通过shared\_ptr<string> p1;或者使用makeshare的方法，shared\_ptr<string> p2 = makeshare<string> (“hello”) 。

Shared\_ptr中多个指针可以指向相同的对象，拷贝和赋值通过一个引用计数器控制，每次使用计数器加一，每次析构（赋予新值或者被销毁）计数器减一，直至变为零，自动释放管理的对象。

Unique\_ptr “唯一”地拥有其所指对象（通过禁止拷贝，仅有移动语义实现），指针销毁，所指对象也被销毁。

Weak\_ptr用于协助shared\_ptr工作，检测资源使用情况。

1. 引用和指针的区别:
2. static关键字的用法:

静态全局变量，静态局部变量，面向对象中（类中使用）：静态数据成员，静态成员函数。

这些不同的用法中，共同之处是：对应的静态对象存放在区别于堆和栈的全局存储区。

静态数据成员：可以由类的所有对象共享。

静态成员函数：因为它不与任何对象关联，所以没有this指针。

（6）设计模式（单例、工厂）及代码实现：

单例：一些类是工具性质的，不需要存储很多与自身相关的数据，因此为了简化设计，只需一个实例。而声明静态变量或者全局变量的方式，会影响封装性，可能会被别的代码改动。通过将构造函数声明私有，避免被外部new.

（7）多线程的实现方式：

（8）大端小端机器判别

小端同逻辑：内存低地址保存数据低位，大端：内存低地址保存数据高位。

阿谢

1. stl中vector的实现，占用大小，默认初始化后分配的空间。Push\_back的后果，超出内从大小时，如何再实现。

重新配置，移动元素，释放空间。

1. map和unordered\_map的区别（实现、查询复杂度、插入复杂度），红黑树与AVL树的区别，红黑树的性质。

平衡二叉树的插入/删除操作带来的旋转操作可能会达到logn次，而红黑树的插入（2次）/删除（3次）操作带来的旋转操作最多为2/3次。

1. C++如何管理内存，用过智能指针吗，说说shared\_ptr的实现，是否线程安全。

New和delete，以及allocator类。

堆（new，delete）、栈（函数局部变量）、自由存储区(malloc,free)、全局/静态存储区和常量存储区。

1. 线程进程的区别

作者：Oscarwin  
链接：https://www.zhihu.com/question/34574154/answer/253165162  
来源：知乎  
著作权归作者所有。商业转载请联系作者获得授权，非商业转载请注明出处。

今年刚找完工作的应届生来答一发，校招主要考察基础和思维，主要涉及C++语言基础，数据结构与算法，TCP/IP协议，网络编程，Linux。

虽然面试技巧和心态也很重要，但这里只有24K纯技术干货。**点赞后，收藏吧！**

**C和C++语言基础**

参考书籍：《C++ primer》，《effective C++》，《STL源码解析》，《深度搜索C++对象模型》

* [**extern关键字作用**](https://link.zhihu.com/?target=http%3A//www.cnblogs.com/yc_sunniwell/archive/2010/07/14/1777431.html)
* [**static关键字作用**](https://link.zhihu.com/?target=http%3A//blog.csdn.net/shanghairuoxiao/article/details/72904292)
* **volatile是干啥的**

https://www.cnblogs.com/yc\_sunniwell/archive/2010/07/14/1777432.html

* **说说const的作用，越多越好**
* [**new与malloc区别**](https://link.zhihu.com/?target=http%3A//blog.csdn.net/shanghairuoxiao/article/details/70337890)
* [**C++多态性与虚函数表**](https://link.zhihu.com/?target=http%3A//blog.csdn.net/hackbuteer1/article/details/7475622)

1. C++多态的实现？
2. 虚函数的作用？
3. 虚函数用于实现多态，这点大家都能答上来但是虚函数在设计上还具有封装和抽象的作用。比如抽象工厂模式。
4. 动态绑定是如何实现的？
5. [静态多态和动态多态](https://link.zhihu.com/?target=http%3A//blog.csdn.net/u013630349/article/details/48009815)。静态多态是指通过模板技术或者函数重载技术实现的多态，其在编译器确定行为。动态多态是指通过虚函数技术实现在运行期动态绑定的技术。
6. [**虚函数表**](https://link.zhihu.com/?target=http%3A//blog.csdn.net/haoel/article/details/1948051/)
7. 虚函数表是针对类的还是针对对象的？同一个类的两个对象的虚函数表是怎么维护的？
8. 编译器为每一个类维护一个虚函数表，每个对象的首地址保存着该虚函数表的指针，同一个类的不同对象实际上指向同一张虚函数表。

* **纯虚函数如何定义，为什么对于存在虚函数的类中析构函数要定义成虚函数**
* 函数声明的括号后加=0； 使得指针能够调用正确的析构函数
* **析构函数能抛出异常吗**
* **构造函数和析构函数中调用虚函数吗？**
* **指针和引用的区别**
* **指针与数组千丝万缕的联系**
* [**智能指针是怎么实现的？什么时候改变引用计数？**](https://link.zhihu.com/?target=http%3A//www.cnblogs.com/xiehongfeng100/p/4645555.html)

1. 构造函数中计数初始化为1；
2. 拷贝构造函数中计数值加1；
3. 赋值运算符中，左边的对象引用计数减一，右边的对象引用计数加一；
4. 析构函数中引用计数减一；
5. 在赋值运算符和析构函数中，如果减一后为0，则调用delete释放对象。
6. share\_prt与weak\_ptr的区别？

* [**C++四种类型转换**](https://link.zhihu.com/?target=http%3A//www.jellythink.com/archives/205)：**static\_cast, dynamic\_cast, const\_cast, reinterpret\_cast**
* **内存对齐的原则**
* **内联函数有什么优点？内联函数与宏定义的区别？**
* **C++内存管理**
* **STL里的内存池实现**
* https://blog.csdn.net/a987073381/article/details/52245795
* **STL里set和map是基于什么实现的。红黑树的特点？**
* **STL里的其他数据结构和算法实现也要清楚**   
  这个问题，把STL源码剖析好好看看，不仅面试不慌，自己对STL的使用也会上升一个层次。
* **必须在构造函数初始化式里进行初始化的数据成员有哪些**
* [**模板特化**](https://link.zhihu.com/?target=http%3A//blog.csdn.net/thefutureisour/article/details/7964682/)
* [**定位内存泄露**](https://link.zhihu.com/?target=http%3A//www.cnblogs.com/skynet/archive/2011/02/20/1959162.html)   
  (1)在windows平台下通过CRT中的库函数进行检测；   
  (2)在可能泄漏的调用前后生成块的快照，比较前后的状态，定位泄漏的位置   
  (3)Linux下通过工具valgrind检测
* **手写****strcpy，memcpy，strcat，strcmp等函数**

**数据结构与算法**

这一块考察范围太广，主要靠多刷题吧，牛客网，剑指OFFER，LeetCode等。

[**Hash表**](https://link.zhihu.com/?target=http%3A//blog.csdn.net/shanghairuoxiao/article/details/73693458)

* Hash表实现（拉链和分散地址）
* Hash策略常见的有哪些？
* STL中hash\_map扩容发生什么？   
  (1) 创建一个新桶，该桶是原来桶两倍大最接近的质数(判断n是不是质数的方法：用n除2到*sqrt*(*n*)范围内的数) ；   
  (2) 将原来桶里的数通过指针的转换，插入到新桶中(注意STL这里做的很精细，没有直接将数据从旧桶遍历拷贝数据插入到新桶，而是通过指针转换)   
  (3) 通过swap函数将新桶和旧桶交换，销毁新桶。

[**树**](https://link.zhihu.com/?target=http%3A//blog.csdn.net/shanghairuoxiao/article/details/73752446)

* 二叉树结构，二叉查找树实现；
* 二叉树的六种遍历；
* 二叉树的按层遍历；
* 递归是解决二叉树相关问题的神级方法；
* 树的各种常见算法题([http://blog.csdn.net/xiajun07061225/article/details/12760493](https://link.zhihu.com/?target=http%3A//blog.csdn.net/xiajun07061225/article/details/12760493))；
* **什么是红黑树？**
* **红黑树与AVL树的区别**
* [**Trie树(字典树)**](https://link.zhihu.com/?target=http%3A//blog.csdn.net/hackbuteer1/article/details/7964147)

**链表**

* 链表和插入和删除，单向和双向链表都要会
* 链表的问题考虑多个指针和递归   
  (1) 反向打印链表(递归)   
  (2) 打印倒数第K个节点(前后指针)   
  (3) 链表是否有环(快慢指针)等等。

**栈和队列**

* **队列和栈的区别**？(从实现，应用，自身特点多个方面来阐述，不要只说一个先入先出，先入后出，这个你会别人也会，要展现出你比别人掌握的更深)
* 典型的应用场景

**海量数据问题**

* 十亿整数（随机生成，可重复）中前K最大的数
* **十亿整数（随机生成，可重复）中出现频率最高的一千个**

[**排序算法**](https://link.zhihu.com/?target=http%3A//blog.csdn.net/shanghairuoxiao/article/details/74063684)

* 排序算法当然是基础内容了，必须至少能快速写出，快排，建堆，和归并
* 每种算法的时间空间复杂度，最好最差平均情况

[**位运算**](https://link.zhihu.com/?target=http%3A//blog.csdn.net/shanghairuoxiao/article/details/75386508)

**布隆过滤器**

几十亿个数经常要查找某一个数在不在里面，使用布隆过滤器，布隆过滤器的原理。布隆过滤器可能出现误判，怎么保证无误差？

**网络与TCP/IP**

参考书籍：《图解TCP/IP》，《TCP/IP详解 卷一》，《图解HTTP》，《HTTP权威指南》

* [TCP与UDP之间的区别](https://link.zhihu.com/?target=http%3A//blog.csdn.net/shanghairuoxiao/article/details/68927070)   
  (1) IP首部，TCP首部，UDP首部   
  (2) TCP和UDP区别   
  (3) TCP和UDP应用场景   
  (4) 如何实现可靠的UDP
* [TCP三次握手与四次挥手](https://link.zhihu.com/?target=http%3A//blog.csdn.net/shanghairuoxiao/article/details/68927100)
* [详细说明TCP状态迁移过程](https://link.zhihu.com/?target=http%3A//blog.csdn.net/shanghairuoxiao/article/details/68927100)   
  (1) 三次握手和四次挥手状态变化；   
  (2) 2MSL是什么状态？作用是什么？   
  (3)三次握手为什么不是两次或者四次？
* [TCP相关技术](https://link.zhihu.com/?target=http%3A//blog.csdn.net/shanghairuoxiao/article/details/68927100)

1. TCP重发机制，Nagle算法
2. TCP的拥塞控制使用的算法和具体过程
3. TCP的窗口滑动

* [TCP客户与服务器模型，用到哪些函数](https://link.zhihu.com/?target=http%3A//blog.csdn.net/shanghairuoxiao/article/details/69803044)
* [UDP客户与服务器模型，用到哪些函数](https://link.zhihu.com/?target=http%3A//blog.csdn.net/shanghairuoxiao/article/details/69951345)
* [域名解析过程，ARP的机制，RARP的实现](https://link.zhihu.com/?target=http%3A//blog.csdn.net/shanghairuoxiao/article/details/68926923)
* Ping和TraceRoute实现原理

[**HTTP**](https://link.zhihu.com/?target=https%3A//github.com/semlinker/awesome-http%23bs-%25E7%25BB%2593%25E6%259E%2584%25E5%25AE%259A%25E4%25B9%2589)

* **http/https 1.0、1.1、2.0的特点和区别**
* **get/post 区别**
* **HTTP返回状态码**
* **http 协议头相关**

http数据由请求行，首部字段，空行，报文主体四个部分组成   
首部字段分为：通用首部字段，请求首部字段，响应首部字段，实体首部字段

* **https与http的区别？如何实现加密传输？加解密方式？**
* **浏览器中输入一个URL发生什么，用到哪些协议？**

**安全相关**

至少了解攻击的原理和基本的防御方法，常见的攻击方法有一下几种

* SQL注入
* XSS
* CSRF
* SYN洪水攻击
* APR欺骗

[**数据库**](https://link.zhihu.com/?target=http%3A//blog.csdn.net/shanghairuoxiao/article/details/76888423)

主要参考书籍：《数据库系统概念》，《高性能MySQL》

* SQL语言(内外连接，子查询，分组，聚集，嵌套，逻辑)
* MySQL索引方法？索引的优化？
* InnoDB与MyISAM区别？
* 事务的ACID
* 事务的四个隔离级别
* 查询优化(从索引上优化，从SQL语言上优化)
* B-与B+树区别？
* MySQL的联合索引(又称多列索引)是什么？生效的条件？
* 分库分表

**Linux**

主要参考书籍：《现代操作系统》，《APUE》，《UNP》，《LINUX内核设计与实现》，《深入理解LINUX内核》

[**进程与线程**](https://link.zhihu.com/?target=http%3A//blog.csdn.net/shanghairuoxiao/article/details/74012512)

(1) 进程与线程区别？   
(2) 线程比进程具有哪些优势？   
(3) 什么时候用多进程？什么时候用多线程？   
(4) LINUX中进程和线程使用的几个函数？   
(5) 线程同步？   
在Windows下线程同步的方式有：互斥量，信号量，事件，关键代码段   
在Linux下线程同步的方式有：互斥锁，自旋锁，读写锁，屏障(并发完成同一项任务时，屏障的作用特别好使) 知道这些锁之间的区别，使用场景？

* [**进程间通讯方式**](https://link.zhihu.com/?target=http%3A//www.cnblogs.com/CheeseZH/p/5264465.html)

1. **匿名管道与命名管道的区别**：匿名管道只能在具有公共祖先的两个进程间使用。
2. **共享文件映射mmap**   
   mmap建立进程空间到文件的映射，在建立的时候并不直接将文件拷贝到物理内存，同样采用缺页终端。mmap映射一个具体的文件可以实现任意进程间共享内存，映射一个匿名文件，可以实现父子进程间共享内存。
3. **常见的信号有哪些？**：SIGINT，SIGKILL(不能被捕获)，SIGTERM(可以被捕获)，SIGSEGV，SIGCHLD，SIGALRM

* [**内存管理**](https://link.zhihu.com/?target=http%3A//blog.csdn.net/shanghairuoxiao/article/details/70256247)

1. 虚拟内存的作用？
2. 虚拟内存的实现？
3. 操作系统层面对内存的管理？
4. 内存池的作用？STL里[内存池如何实现](https://link.zhihu.com/?target=https%3A//github.com/oscarwin/MemoryPool" \t "_blank)？
5. 进程空间和内核空间对内存的管理不同？
6. Linux的slab层，VAM？
7. 伙伴算法
8. 高端内存

* **进程调度**

1. Linux进程分为两种，实时进程和非实时进程；
2. 优先级分为静态优先级和动态优先级，优先级的范围；
3. 调度策略
4. 交互进程通过平均睡眠时间而被奖励；

* [**死锁**](https://link.zhihu.com/?target=http%3A//blog.csdn.net/shanghairuoxiao/article/details/70444940)

(1) 死锁产生的条件；   
(2) 死锁的避免；

* **命令行**

1. Linux命令 在一个文件中，倒序打印第二行前100个大写字母

cat filename | head -n 2 | tail -n 1 | grep '[[:upper:]]' -o | tr -d '\n'| cut -c 1-100 | rev

1. 与CPU，内存，磁盘相关的命令(top，free, df, fdisk)
2. 网络相关的命令netstat，tcpdump等
3. sed, awk, grep三个超强大的命名，分别用与格式化修改，统计，和正则查找
4. ipcs和ipcrm命令
5. 查找当前目录以及字母下以.c结尾的文件，且文件中包含”hello world”的文件的路径
6. 创建定时任务

* **IO模型**

1. **五种IO模型：**阻塞IO，非阻塞IO，IO复用，信号驱动式IO，异步IO
2. **select，poll，epoll的区别**

* 线程池，内存池 自己动手实现一遍

**Linux的API**

* **fork与vfork区别**   
  fork和vfork都用于创建子进程。但是vfork创建子进程后，父进程阻塞，直到子进程调用exit()或者excle()。   
  对于内核中过程fork通过调用clone函数，然后clone函数调用do\_fork()。do\_fork()中调用copy\_process()函数先复制task\_struct结构体，然后复制其他关于内存，文件，寄存器等信息。fork采用写时拷贝技术，因此子进程和父进程的页表指向相同的页框。但是vfork不需要拷贝页表，因为父进程会一直阻塞，直接使用父进程页表。
* **exit()与\_exit()区别**   
  exit()清理后进入内核，\_exit()直接陷入内核。
* 孤儿进程与僵死进程

1. 孤儿进程是怎么产生的？
2. 僵死进程是怎么产生的？
3. 僵死进程的危害？
4. 如何避免僵死进程的产生？

* **Linux是如何避免内存碎片的**

1. 伙伴算法，用于管理物理内存，避免内存碎片;
2. 高速缓存Slab层用于管理内核分配内存，避免碎片。

* **共享内存的实现原理？**
* 系统调用与库函数(open, close, create, lseek, write, read)
* 同步方法有哪些？

1. 互斥锁，自旋锁，信号量，读写锁，屏障
2. 互斥锁与自旋锁的区别：互斥锁得不到资源的时候阻塞，不占用cpu资源。自旋锁得不到资源的时候，不停的查询，而然占用cpu资源。
3. [死锁](https://link.zhihu.com/?target=http%3A//blog.csdn.net/shanghairuoxiao/article/details/70444940)

**其他**

* ++i是否是原子操作   
  明显不是，++i主要有三个步骤，把数据从内存放在寄存器上，在寄存器上进行自增，把数据从寄存器拷贝会内存，每个步骤都可能被中断。
* 判断大小端

**设计模式**

* [单例模式线程安全的写法](https://link.zhihu.com/?target=http%3A//www.jellythink.com/archives/82)
* STL里的迭代器模式，适配器模式

**分布式系统**

* [map\_reduce原理](https://link.zhihu.com/?target=http%3A//blog.jobbole.com/80619/) （这篇文章讲的很通俗易懂）
* 负载均衡
* CDN

**主要列出考察内容的方向，问题的理解就得靠自己实践和阅读书籍了，相关问题的答案在我**[**这篇博客**](https://link.zhihu.com/?target=http%3A//blog.csdn.net/shanghairuoxiao/article/details/72876248)**中给出了自己理解的程度，欢迎交流！**