

此系列内容以及观点仅个人感受，不妥之处直接私我！目的为了大家能更好的知道面试题难度以及如何准备，希望能让大家少浪费时间寻找资料，多点时间学点干货！因为篇幅原因，大部分题目题解简洁，但有相关书籍推荐进阶阅读，望谅解！

当时投递的时候，岗位只是说了是c/c++开发工程师，到了二面问了面试官才知道我面的是无人驾驶部门。

一 面试情况

此战终结于技术面最后一面，值得深思。由于内推简历没有直接通过筛选，参加了笔试才并有幸参加了面试，但是一面电话面结束后，面试官说如果有二面希望能现场面！

1 一面(电话面25分钟)

- 简述一下项目

一面提项目，一般说明项目背景，自己做了什么就好了，不会深问，但是能准备着更好

- 项目中遇到过什么问题，怎么解决

这个问题，凡是涉及项目基本上都跑不了，前面说过需要准备几个面试官百分之80会问的关于项目的题。

- 都学过什么课程，计算机方向是软件工程吗

计算机网络，数据结构，操纵系统，编译原理，人工智能，大数据等随便你选几个，保证自己能说出个123

- C++中的类的大小计算

C++中类的成员函数，静态成员是不占类的大小的。类的大小等于基类的大小+子类个non-static成员变量的大小再+非虚基类大小，如果有多态性还要考虑vptr（可能不止一个）大小，这里成员变量是会被字节对齐的。

- 介绍一下http与https及区别

HTTPS和HTTP的区别

超文本传输协议**HTTP**协议被用于在Web浏览器和网站服务器之间传递信息。

HTTP协议以明文方式发送内容，不提供任何方式的数据加密，如果攻击者截取了Web浏览器和网站服务器之间的传输报文，就可以直接读懂其中的信息，因此**HTTP**协议不适合传输一些敏感信息，比如信用卡号、密码等。

为了解决**HTTP**协议的这一缺陷，需要使用另一种协议：安全套接字层超文本传输协议**HTTPS**。为了数据传输的安全，**HTTPS**在**HTTP**的基础上加入了SSL协

议，SSL依靠证书来验证服务器的身份，并为浏览器和服务器之间的通信加密。

HTTPS和HTTP的区别主要为以下四点：

- 一、https协议需要到ca申请证书，一般免费证书很少，需要交费。
- 二、http是超文本传输协议，信息是明文传输，https 则是具有安全性的ssl加密传输协议。
- 三、http和https使用的是完全不同的连接方式，用的端口也不一样，前者是80，后者是443。
- 四、http的连接很简单，是无状态的；HTTPS协议是由SSL+HTTP协议构建的可进行加密传输、身份认证的网络协议，比http协议安全。

- 打印int时不小心用了%s会出现什么问题
- 段错误

```
int i = 10;
char *s = "12";
printf("%d\n", s); // 数据不对
printf("%s\n", i); // 段错误
```

- 链表成环

```
class Solution {
public:
    bool hasCycle(ListNode *head) {
        ListNode *slow = head, *fast = head;
        while((fast != NULL) && (fast->next != NULL))
        {
            //慢指针
            slow = slow->next;
            //快指针
            fast = fast->next->next;
            if(slow == fast) return true;
        }
        return false;
    }
};
```

```

class Solution(object):
    def hasCycle(self, head):
        """
        :type head: ListNode
        :rtype: bool
        """
        fast=slow=head
        while slow and fast and fast.next:
            # 慢指针
            slow=slow.next
            # 快指针
            fast=fast.next.next
            if slow is fast:
                return True
        return False

```

```

public class Solution {
    public boolean hasCycle(ListNode head) {
        if(head == null || head.next == null) return false;
        ListNode fast = head;
        ListNode slow = head;
        while(fast != null && fast.next != null){
            fast = fast.next.next;
            slow = slow.next;
            if(slow == fast) return true;
        }
        return false;
    }
}

```

• 逻辑题

1000瓶无色无味的药水，其中有一瓶毒药，10只小白鼠拿过来做实验。喝了无毒的药水第二天没事儿，喝了有毒的药水后第二天会死亡。如何在一天之内(第二天)找出这瓶有毒的药水？

思路就是用二进制， $2^{10}=1024$ ，也就是10只小白鼠最多能验出1024瓶药水，哪个有毒。小白鼠编号，1-10。瓶子也编号，1-1000，然后把瓶子的编号转变为二进制数。如果第几位是1，就把这瓶水给第几个小白鼠喝。最后大概每个小白鼠喝500瓶药水的混合液。如果还不懂，下面列几个数字解释一下。

瓶子编号 二进制数 第几个小白鼠喝

1 0000000001 1

2 0000000010 2

3 0000000011 1, 2

4 0000000100 3

5 0000000101 1, 3

大概就是这意思，再反过来，假如1号和3号小白鼠死了，死的小白鼠用1表示，再写成2进制数：0000000101，转化为十进制数是5，从上面列出来的也可以看出1, 3都喝了5号瓶的水，所以就是第五瓶水有毒。

解决方案 1) 我们将1000瓶液体编号1~1000，然后将编号转化为10位二进制，如1号就是0000000001； 2) 将十只小白鼠编号1~10； 3) 将液体的二进制编号上为1的位数给对应的小白鼠喝，如液体编号为 1111100000，那就是1~5号小白鼠不喝这瓶液体，6~10号小白鼠喝这瓶液体； 4) 一星期后观察小白鼠的死亡情况，如果1~5号小白鼠死亡，6~10号小白鼠存活，那么有毒的那瓶液体对应的二进制编码为0000011111； 5) 将第四步得到的二进制编码转化为十进制，这里是31号，因此我们可以推断出编号为31的液体是被污染的。

- cookie 和session 的区别:

cookie和session的共同之处在于：
cookie和session都是用来跟踪浏览器用户身份的会话方式。

cookie 和session 的区别:

1、cookie数据存放在客户的浏览器上，
session数据放在服务器上。

2、cookie不是很安全，别人可以分析存放在本地的COOKIE并进行COOKIE欺骗，考虑到安全应当使用session。

3、session会在一定时间内保存在服务器上，超过时间会销毁这个SESSION。当访问增多，会比较占用你服务器的性能考虑到减轻服务器性能方面，应当使用COOKIE。

4、单个cookie保存的数据不能超过4K，很多浏览器都限制一个站点最多保存20个cookie。

5、所以个人建议：将登陆信息等重要信息存放为SESSION，其他信息如果需要保留，可以放在COOKIE中

- 还有什么可以问我的吗

这个问题一般来说会有下文，只要不问一些敏感话题就行了。

第二天早上收到二面通知，但是说需要现场面。思考了半天，决定还是去现场面试，虽然需要差不多单程900的车费，万一现场面试更简单呢是吧。当然我也知道，我不去一定会后悔，索性还是去尝试未尝不是一件好事，毕竟正好还有小米和滴滴，bigo的现场面试。从南方到北方，邮箱通知上面是九点到西土城的泰富酒店签到，下火车时间差不多为六点多，所以到达那里的时候差不多七点，我是第一个到那里并签到的，然后就在那里看自己准备的算法题，其实昨天晚上在火车上也复习了很久，我会尽全力的去完成这次任务，以至今也没后悔之言。北京的天空依然那么的纯蓝！



在这里插入图片描述

2 二面(现场面)

我们签到以后，面试官可以看见签到时间，是一哥很温柔的小哥，让我把行李放了坐下，别紧张，先自我介绍，然后他说，你这么远过来其实没必要的，可以申请远程的，不然太折腾了，今天我们就简单问问。

- 自我介绍

自我介绍完了以后

面试官:你觉得你的一面感觉如何

我:我说一面面试官很好(其实我从之前的沟通中已经感觉一面二面是同一个面试官了),不太会的都会引导我,然后回

头查了相关的资料。面试官还是比较满意的。注意：复盘很重要，一般都有面试记录的。

- 我看你写了三个项目，说一个熟悉一些的，背景，你做了啥，有什么难点

我们看几个简单题

- 构造函数为什么不能是虚函数

虚函数的调用需要虚函数表指针，而该指针存放在对象的内容空间中；若构造函数声明为虚函数，那么由于对象还未创建，还没有内存空间，更没有虚函数表地址用来调用虚函数。

- Makefile、GDB应该都用过吧

这一篇中有相关的书籍

- 原子变量和volatile区别（C++11）

Volatile变量可以确保先行关系，即写操作会发生在后续的读操作之前,但它并不能保证原子性。例如用volatile修饰count变量那么 count++ 操作就不是原子性的。而AtomicInteger类提供的atomic方法可以让这种操作具有原子性如getAndIncrement()方法会原子性的进行增量操作把当前值加一，其它数据类型和引用变量也可以进行相似操作。

- 智能指针介绍（C++11）、

1.auto_ptr主要是用来解决资源自动释放的问题;auto_ptr支持赋值和复制，将

指针的所有权转移，但是如果转移后再访问原来得指针，行为不确定，程序可能会在运行时出错。

2.unique_ptr与auto_ptr一样，也是建立所有权机制，但是不支持复制和赋值，所以将一个unique_ptr对象赋值给另一个时，程序编译出错；但如果将临时的unique_ptr赋值或复制给另一个对象时，没有问题。unique_ptr比auto_ptr更安全。

3.shared_ptr和unique_ptr都只能一个智能指针引用对象，而shared_ptr则是可以多个智能指针同时拥有一个对象。

shared_ptr实现方式就是使用引用计数。引用计数的原理是，多个智能指针同时引用一个对象，每当引用一次，引用计数加一，每当智能指针销毁了，引用计数就减一，当引用计数减少到0的时候就释放引用的对象。这种引用计数的增减发生在智能指针的构造函数，复制构造函数，赋值操作符，析构函数中。

这种方式使得多个智能指针同时对所引用的对象有拥有权，同时在引用计数减到0之后也会自动释放内存，也实现了auto_ptr和unique_ptr的资源释放的功能。

注意，智能指针默认使用delete来释放资源，如果资源是FILE*怎么办？释放的时候就需要用fclose了。如何实现呢？

shared_ptr构造函数可以传递一个删除器。

```
FILE* pStm = fopen(...);
```

```
shared_ptr fileRes(pStm, &fclose);
```

4.weak_ptr,shared_ptr是一种强引用的关系，智能指针直接引用对象。那么这个

会代码一个隐含的问题，就是循环引用，从而造成内存泄漏，首先来看一个循环引用的例子。

```
class Parent

{

public:

    shared_ptr<Child> child;

};

class Child

{

public:

    shared_ptr<Parent> parent;

};

void Function()

{

    shared_ptr<Parent> pA(new Parent);

    shared_ptr<Child> pB(new Child);

    pA->child = pB;

    pB->parent = pA;

}

// 第一条语句使得pA引用了Parent一个指针，Parent引

// 第二条语句使得pB引用了Child一个指针，Child引用

// 第三条语句，调用了shared_ptr<Child>类的赋值操

// 第四条语句，调用了shared_ptr<Parent>类的赋值操
```

//函数返回之前调用了shared_ptr<Parent>和shared_ptr<Child>

看！函数执行完之后new出来的Parent和Child并没有释放，所以出现了内存泄漏。

出现泄漏的原因就是pA和pB相互引用了，导致两者所引用对象的引用计数不能减少到0，造成泄漏。

如果把第三条语句或者第四条语句任意删除一个，就不会有泄漏了。这就是强引用所带来的问题。weak_ptr从字面意思上可以看出是一个弱指针，不是说明这个指针的能力比较弱，而是说他对他所引用的对象的所有权比较弱，说得更直接一点儿就是他并不拥有所引用对象的所有权，而且他还不能直接使用他所引用的对象。

在stl中，weak_ptr是和shared_ptr配合使用的，在实现shared_ptr的时候也就考虑了weak_ptr的因素。weak_ptr是shared_ptr的观察者，它不会干扰shared_ptr所共享对象的所有权，当一个weak_ptr所观察的shared_ptr要释放它的资源时，它会把相关的weak_ptr的指针设置为空，防止weak_ptr持有悬空的指针。注意：weak_ptr并不拥有资源的所有权，所以不能直接使用资源。可以从一个weak_ptr构造一个shared_ptr以取得共享资源的所有权。

weak_ptr是为配合shared_ptr而引入的一种智能指针，它更像是shared_ptr的一个助手，而不是智能指针，因为它不具有普通指针的行为，没有重载operator*和operator->，它的最大作用在于协助

`shared_ptr`，像旁观者那样观测资源的使用情况。

`weak_ptr`被设计为与`shared_ptr`共同工作，可以从一个`shared_ptr`或者另一个`weak_ptr`对象构造，获得资源的观测权。但`weak_ptr`没有共享资源，它的构造不会引起指针引用计数的增加。同样，在`weak_ptr`析构时也不会导致引用计数的减少，它只是一个静静地观察者。

使用`weak_ptr`的成员函数`use_count()`可以观测资源的引用计数，另一个成员函数`expired()`的功能等价于`use_count() == 0`，但更快，表示观测的资源（也就是`shared_ptr`管理的资源）已经不复存在了。

`weak_ptr`没有重载`operator*`和`->`，这是特意的，因为它不共享指针，不能操作资源，这是它弱的原因。但它可以使用一个非常重要的成员函数`lock()`从被观测的`shared_ptr`获得一个可用的`shared_ptr`对象，从而操作资源。当`expired() == true`的时候，`lock()`函数将返回一个存储空指针的`shared_ptr`。

- 智能指针内部实现（C++11）

智能指针类将一个计数器与类指向的对象相关联，引用计数跟踪该类有多少个对象共享同一指针。每次创建类的新对象时，初始化指针并将引用计数置为1；当对象作为另一对象的副本而创建时，拷贝构造函数拷贝指针并增加与之相应的引用计数；对一个对象进行赋值时，赋值操作符减少左操作数所指对象的引用计数（如果引用计数为减至0，则删除对象），并增加右操作数所指对象的引

用计数；调用析构函数时，构造函数减少引用计数（如果引用计数减至0，则删除基础对象）。智能指针就是模拟指针动作的类。所有的智能指针都会重载 `->` 和 `*` 操作符。智能指针还有许多其他功能，比较有用的是自动销毁。这主要是利用栈对象的有限作用域以及临时对象（有限作用域实现）析构函数释放内存。

- DPDK内部实现(这个是因为简历上有写，关于一个高性能数据包处理库)

Winpcap:它的一个流程是 npf网络组包过滤器首先负责从网络中采集数据包，完成数据的过滤拷贝到内核缓存区，然后调用相应的动态库文件将数据传递到应用层缓冲区，最后应用程序处理。具体工作原理

(1)网卡接受数据包到达信息，然后产生硬件中断，通知cpu调度处理，中断服务程序判断数据包的有效性，分配一个缓冲。

(2)BPF模块根据用户的规则过滤数据包，并把数据包插入到内核的网卡驱动缓冲队列中。

(3)用户程序通过系统调用用来读取内核缓冲区的数据包 完成数据采集

优化方案

(1)使用双缓冲减少线程锁

(2)多线程

(3)将原始的数据包还原成流保存 减少对数据包的存储

在内核层提供了通用socket环形缓冲，不进入内核协议栈，最后在应用层通过socket链接同时使用mmap技术直接访问socket环状缓冲区

DPDK

(1)强大的 高度优化的用户空间库和驱动程序。帮助用户将控制面和数据面平台进行整合

四个技术点

(1) 通过大页提高内存的使用效率

(2)uio 驱动的很少一部分放在内核空间，大部分功能在用户空间实现，使用UIO可以避免设备的驱动程序需要随着内核的更新而更新

(3)cpu affinity:将控制面线程和各个数据面线程绑定到不同的cpu内核，省去了反复调度的性能消耗。

(4)zero copy 数据包设备到内核空间再到用户程序空间的床底过程中，减少拷贝次数和系统调用，降低cpu在这个方面的负载，使得cpu更多的在数据处理上。

- 13、libevent结构、内部实现（回调+同步）、多线程实现

首先I/O复用经过封装；

统一事件源（I/O事件，信号事件，定时事件）；

事件处理提前注册（回调函数）

Libevent是线程不安全的，但是libevent提供了锁机制，而且在实现框架上尽量避免使用锁，像memcache多线程使用

libevent，他的实现框架是主线程监听到读写事件分发任务（使用CQ队列），每个工作线程对应一个CQ队列

- epoll内部实现

红黑树 就绪事件双向链表；每当就绪事件到来时，通过实现注册好的回调函数将就绪事件加入到就绪事件队列中，epoll_wait返回时只需要遍历就绪事件双向链表

- timewait作用

客户端收到服务的释放连接的请求后，不是立马进入CLOSE状态，而是还要再等待2MSL。理由是：

- 确保最后一个确认报文能够到达。如果不能到达，服务端就会重发FIN请求释放连接。等待一段时间没有收到重发就说明服务的已经CLOSE了。如果有重发，则客户端再发送一次LAST ack信号
- 等待一段时间是为了让本连接持续时间内所产生的所有报文都从网络中消失，使得下一个新的连接不会出现旧的连接请求报文

- 编程题1 手撕快排
- 编程题2 二分查找变种

3 总结

最终虽然败北，但是学会了一下几点

- 公司招你去是干活了，不会因为你怎么怎么的而降低对你的要求标准。
- 工具上面写代码和手撕代码完全不一样。
- 珍惜每一次面试机会并学会复盘
- 对于应届生主要考察的还是计算机基础知识的掌握，项目要求没有那么高，是自己做的就使劲抠细节，做测试，这样你就知道会遇到什么问题，遇到什么难点，如何解决的。当被问到的时候就可以侃侃而谈了。

有收获？希望老铁们来个三连击，
给更多的人看到这篇文章

1、给俺点个赞呗，可以让更多的人看到这篇文章，顺便激励下我，嘻嘻。

2、老铁们，关注我的原创微信公重号「我是程序员小贱」，专注于写各大中小厂面经，保存让你看完有所收获，不信你打我。♡
看完三件事：如果您看完有一点点收获，快速迎娶白富美方式：

作者简介：刚经历完秋招，可知计算机基础知识的重要性和可操作性。因此申请了WX公重号[我是程序员小贱]，希望能够帮助大家，尽量少走弯路，多学点干货！转载说明：未获得授权，禁止转载