> 文章首发于此。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！

> 深信服偏向安全类公司，做VPN起家。技术栈c/c++系列偏多。

一 面试情况

> 注意了，有些同学可能不需要笔试环节，通过在牛客寻找内推，简历合格直接通知面试，效率非常高(这一点我的印象非常深刻)，各自都不耽搁，如果合适两三天一定出结果，不合适就是不匹配！

1 一面(电话面25分钟)

> 一面基础面，注重的c/c++语法基础。参看书籍见Linux后台开发必看。

- 自我介绍

> 姓名，学校，和岗位的匹配度等。

- 阻塞非阻塞区别

> **阻塞，非阻塞**：进程/线程要访问的数据是否就绪，进程/线程是否需要等待；
>
> **同步，异步**：访问数据的方式，同步需要**主动**读写数据，在读写数据的过程中还是会阻塞；异步只需要I/O操作完成的通知，并不主动读写数据，由操作系统内核完成数据的读写。
>
> 对unix来讲：阻塞式I/O(默认)，非阻塞式I/O(nonblock)，I/O复用(select/poll/epoll)都属于**同步I/O**，因为它们在数据由内核空间复制回进程缓冲区时都是阻塞的(不能干别的事)。只有异步I/O模型(AIO)是符合异步I/O操作的含义的，即在 1数据准备完成、2由内核空间拷贝回缓冲区后通知进程，在等待通知的这段时间里可以干别的事。

- c/c++如何相互调用

> （1）C调用C++的函数或变量，在C++的头文件声明为extern "C"，C调用的时候只使用extern 声明。
>
> （2）extern "C" {}，声明用于C++中，告诉编译器对{}中声明的函数或变量使用C的方式生成（或寻找）目标符号。

- 如何处理**僵尸进程**

> 如果父进程在子进程之前终止，则所有的子进程的父进程都会改变为**init**进程，我们称这些进程由**init**进程领养。这时使用**ps**命令查看后可以看到子进程的父进程**ppid**已经变为了1。
>
> 而当子进程在父进程之前终止时，内核为每个终止子进程保存了一定量的信

> 息，所以当终止进程的父进程调用**wait或waitpid**时，可以得到这些信息。这些信息至少包括进程ID、该进程的终止状态、以及该进程使用的CPU时间总量。其他的进程所使用的存储区，打开的文件都会被内核释放。
>
> 一个已经终止、但是其父进程尚未对其进行善后处理（获取终止子进程的有关信息，释放它仍占用的资源）的进程被称为僵尸进程。**ps**命令将僵尸进程的状态打印为Z。
>
> 当子进程终止时，内核就会向它的父进程发送一个**SIGCHLD**信号，父进程可以选择忽略该信号，也可以提供一个接收到信号以后的处理函数。对于这种信号的系统默认动作是忽略它。
>
> 我们不希望有过多的僵尸进程产生，所以当父进程接收到SIGCHLD信号后就应该调用 **wait 或 waitpid** 函数对子进程进行善后处理，释放子进程占用的资源。
>
> 在UNIX 系统中，一个进程结束了，但是他的父进程没有等待(调用wait / waitpid)他， 那么他将变成一个僵尸进程。（查看进程表中可以通过Z标示查看僵尸进程）

- 三次握手中，第二次握手丢失如何处理

> 三次握手协议中，服务器维护一个**未连接队列**，该队列为每个客户端的SYN包（syn=j）开设一个条目，该条目表明服务器已收到SYN包，并向客户发出确认，正在等待客户的确认包。这些条目所标识的连接在服务器处于 Syn_RECV状态，当服务器收到客户的确认包时，删

> 除该条目，服务器进入ESTABLISHED状态。

> 三次握手协议
>
> 服务器发送完SYN－ACK包，如果未收到客户确认包，服务器进行首次重传，等待一段时间仍未收到客户确认包，进行第二次重传，如果重传次数超过系统规定的最大重传次数，系统将该连接信息从半连接队列中删除。
>
> **半连接存活时间**是指半连接队列的条目存活的最长时间，也即服务器从收到SYN包到确认这个报文无效的最长时间，该时间值是所有重传请求包的最长等待时间总和。有时我们也称半连接存活时间为Timeout时间、SYN_RECV存活时间。

- send函数成功表明什么

> send()用于向一个已经连接的socket发送数据，如果无错误，返回值为所发送数据的总数，否则返回SOCKET_ERROR。注意成功地完成send()调用并不意味着数据传送到达。只是把数据放到缓冲区，数据的传送依赖于建立的TCP链接。

- time_wait状态什么时候产生

> **主动关闭方收到被动关闭方的FIN包并发送出ACK时进入TIME_WAIT**
>
> 1）可靠地实现TCP全双工连接的终止

> TCP协议在关闭连接的四次握手过程中，最终的ACK是由主动关闭连接的一端（后面统称A端）发出的，如果这个ACK丢失，对方（后面统称B端）将重发出最终的FIN，因此A端必须维护状态信息（TIME_WAIT）允许它重发最终的ACK。如果A端不维持TIME_WAIT状态，而是处于CLOSED 状态，那么A端将响应RST分节，B端收到后将此分节解释成一个错误（在java中会抛出connection reset的SocketException)。
>
> 因而，要实现TCP全双工连接的正常终止，必须处理终止过程中四个分节任何一个分节的丢失情况，主动关闭连接的A端必须维持TIME_WAIT状态 。
>
> 2）允许老的重复分节在网络中消逝
>
> TCP分节可能由于路由器异常而“迷途”，在迷途期间，TCP发送端可能因确认超时而重发这个分节，迷途的分节在路由器修复后也会被送到最终目的地，这个迟到的迷途分节到达时可能会引起问题。在关闭“前一个连接”之后，马上又重新建立起一个相同的IP和端口之间的“新连接”，“前一个连接”的迷途重复分组在“前一个连接”终止后到达，而被“新连接”收到了。为了避免这个情况，TCP协议不允许处于TIME_WAIT状态的连接启动一个新的可用连接，因为TIME_WAIT状态持续2MSL，就可以保证当成功建立一个新TCP连接的时候，来自旧连接重复分组已经在网络中消逝。

- tcp如何保证可靠性

> TCP提供一种面向连接的、可靠的字节流服务。　面向连接：意味着两个使用TCP的应用（通常是一个客户和一个服

务器）在彼此交换数据之前必须先建立一个TCP连接。在一个TCP连接中，仅有两方进行彼此通信。广播和多播不能用于TCP。 TCP通过下列方式来提供可靠性：

1、应用数据被分割成TCP认为最适合发送的数据块。这和UDP完全不同，应用程序产生的数据报长度将保持不变。(将数据截断为合理的长度)

2、当TCP发出一个段后，它启动一个定时器，等待目的端确认收到这个报文段。如果不能及时收到一个确认，将重发这个报文段。

3、当TCP收到发自TCP连接另一端的数据，它将发送一个确认。这个确认不是立即发送，通常将推迟几分之一秒 。(对于收到的请求，给出确认响应) (之所以推迟，可能是要对包做完整校验)

4、 TCP将保持它首部和数据的检验和。这是一个端到端的检验和，目的是检测数据在传输过程中的任何变化。如果收到段的检验和有差错，TCP将丢弃这个报文段和不确认收到此报文段。 (校验出包有错，丢弃报文段，不给出响应，TCP发送数据端，超时时会重发数据)

5、既然TCP报文段作为IP数据报来传输，而IP数据报的到达可能会失序，因此TCP报文段的到达也可能会失序。如果必要，TCP将对收到的数据进行重新

> 排序，将收到的数据以正确的顺序交给应用层。 (对失序数据进行重新排序，然后才交给应用层)

> 6、既然IP数据报会发生重复，TCP的接收端必须丢弃重复的数据。(对于重复数据，能够丢弃重复数据)

> 7、TCP还能提供流量控制。TCP连接的每一方都有固定大小的缓冲空间。TCP的接收端只允许另一端发送接收端缓冲区所能接纳的数据。这将防止较快主机致使较慢主机的缓冲区溢出。(TCP可以进行流量控制，防止较快主机致使较慢主机的缓冲区溢出)TCP使用的流量控制协议是可变大小的滑动窗口协议。

> 字节流服务:

> 两个应用程序通过TCP连接交换8bit字节构成的字节流。TCP不在字节流中插入记录标识符。我们将这称为字节流服务（bytestreamservice）。 TCP对字节流的内容不作任何解释:: TCP对字节流的内容不作任何解释。TCP不知道传输的数据字节流是二进制数据，还是ASCII字符、EBCDIC字符或者其他类型数据。对字节流的解释由TCP连接双方的应用层解释。

- seq为1000，发送了1000个数据，下一个seq是多少?

> 2001

- 使用过捕包工具吗，说一下抓一个明文http过程

> 讲述一下wireshark使用方法，以及http明文并可扩展到密文https。

- gcc优化参数介绍几个

> -O设置一共有五种：-O0、-O1、-O2、-O3和-Os。 -O0：这个等级（字母“O”后面跟个零）关闭所有优化选项，也是CFLAGS或CXXFLAGS中没有设置-O等级时的默认等级。这样就不会优化代码，这通常不是我们想要的。

> -O1：这是最基本的优化等级。编译器会在不花费太多编译时间的同时试图生成更快更小的代码。这些优化是非常基础的，但一般这些任务肯定能顺利完成。

> -O2：-O1的进阶。这是推荐的优化等级，除非你有特殊的需求。-O2会比-O1启用多一些标记。设置了-O2后，编译器会试图提高代码性能而不会增大体积和大量占用的编译时间。

> -O3：这是最高最危险的优化等级。用这个选项会延长编译代码的时间，并且在使用gcc4.x的系统里不应全局启用。自从3.x版本以来gcc的行为已经有了极大地改变。在3.x，-O3生成的代码也只是比-O2快一点点而已，而gcc4.x中还未必更快。用-O3来编译所有的软件包将产生更大体积更耗内存的二进制文件，大大增加编译失败的机会或不可预知的程序行为

> （包括错误）。这样做将得不偿失，记住过犹不及。在gcc 4.x.中使用-O3是不推荐的。

> -Os：这个等级用来优化代码尺寸。其中启用了-O2中不会增加磁盘空间占用的代码生成选项。这对于磁盘空间极其紧张或者CPU缓存较小的机器非常有用。但也可能产生些许问题，因此软件树中的大部分ebuild都过滤掉这个等级的优化。使用-Os是不推荐的。

二面(50分钟)

> 我的深信服二面全是项目面，所以在简历上一般把自己更加熟悉的项目放在上面，不过问题不大，因为一般面试官会说:"请介绍一下你熟悉的项目"。那么鉴于每个人的项目不同，我就在这提上几点可能是公共的问题。

- 项目背景，项目人员构成，自己所担任角色
- 你在此项目中做了什么
- 在做这个项目中的难点是什么，又遇到困难？如何解决的
- 项目扩展性如何？

三面(30分钟)

> 我也不知道三面是个什么意思，深信服大部分都是一共三面。在我的记忆里，三面面试官很和蔼，跟我说了一下规

> 划，甚至聊了下他以前的故事，so，这里的三面也给大家带来不了什么。

四面(hr面)

> hr都是我的偶像，他们的应变能力让小蓝常常不知所措。比如上一句问你最讨厌的人，下一句就是如果以后你的上司也是这样的怎么处理呢，哭唧唧！下面也整理一下hr常问问题

- 你最擅长的技术方向是什么？
- 现在几个offer？
- 最大优缺点
- 对加班的看法
- 薪资的要求
- 工作中你难以和同事，上司相处，你该怎么办

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。