Follow Up：

工作一年多 在原来的公司涨了两次薪 一次主动 一次被动（领导帮涨的）

这次跳槽面试总时长3周 在互联网大裁员的背景下找到了几个年包40左右的offer

最后选了一个月base 26K 975的工作 项目也是这个公司的核心项目 分布式 高并发相关 这个组招了几个月就进了我一个

字节的某个边缘部门3轮技术面和HR面都过了 但是没给offer 可能是没名额

没有polish自己的经历 别人问我空白期 我就老老实实说考研去了 他们也表示OK

跟大家简单分享一下工作之后的学习和成长路线

> **工作部分**：
> **学习部分**:
> **私货**：

工作部分：

- 先弄清需求再动手 弄清需求之前 可以一行代码都不用写 因为会做无用功 尤其是初中级的Java程序员 大部分做的东西都是跟业务逻辑相关 你需要先弄清楚业务需求是什么 反复跟需求提出方确认
- 多沟通交流 程序员的沟通与表达能力与编码能力同等重要 不要一个人闷头写代码
- 优先处理领导关心的任务和需求 其次再满足其他部门或者客户丢过来的需求

  主动跟领导要活

  主动跟领导同步任务状态

  根据自己从不同地方收获的信息点（测试 售后 客户）主动提需求来优化你的项目（优化的点可以是功能性的也可以是非功能性的）

  君子不器 不要只是做一个执行者 来一个任务你就执行 那样你的可替代性也会非常强

  这样你在领导心中的信誉积分就会越涨越多 下次有挑战性的任务或者涨薪的事情 就会第一个想到你
- **学会摸鱼**：这个很重要 我很多时候 每天的学习时间都要比工作时间长 状态正常一天能学5-6个小时 工作大概是1055这样一个节奏 下午5点之后不工作 专心摸鱼学习 我们公司6点下班 我能在公司学到10点多再回去 周末也会跑去公司上自习 感觉比在外面的咖啡厅 自习室舒服 **划重点：周末不要待在自己租的房子 出去学习或者玩都行！**

学习部分:

方向：

- **面向面试学习**

找一家你一年后想去的公司的岗位 看职位描述 需要掌握什么技能 其次看这个岗位的社招面经 看会问什么方向的问题 有目的的去**系统地**学习这个方向的内容 而不是到面试前背背面经 对你的技术成长没有太大帮助（我这次跳槽就只看了计算机网络的八股 其他的Java还有中间件八股全部没看）

当然这方面我吃了亏 因为我的target是微软 主要考察方向是算法和系统设计 然后别人没给我面试机会（哭）所以面国内大厂的时候 别人问我中间件八股文的时候 就比较尴尬

面其他国内大厂一般就是项目 + 八股文 + 算法 + 系统设计

算法不同公司权重不一样 但是BAT这样的大厂的算法难度也就是leetcode medium 基本上就是medium的原题或者变形题 字节抖音会出动态规划的leetcode hard 但是也是常见的hard原题 所以还好 我这次面试没被算法卡过

- **面向项目学习**

  学习你的项目中用到的技术 比如Java中用到的类库 源码可以去过一遍 用到的中间件原理 开发框架源码 都可以去看 这个优先级要比上面你想去的公司的要面的内容稍微低一点 但是也很重要 因为跟你的项目直接相关 你学了之后 工作效率会更高 产出质量也会更高 这样就能有更多的时间**摸鱼学习** 领导也会给你派更难的活 这是一个正循环

  “生而知之者，上也；学而知之者，次也；困而学之，又其次也；困而不学，民斯为下矣”

  我们很多人都是困而不学的（当然我有时候也是） 工作中遇到问题了 查一下CSDN 就过了 不会去系统地学习 这样你的知识体系永远建立不起来

内容：

会直接上链接 都是我看过 有些是上面的代码我都敲过一遍的 觉得还可以的内容

我看过的觉得一般的不会推荐（笑）

我一般会优先选择需要付费的资源 免费的永远是最贵的

> 首先需要先确定一个基本理念 就是不管你学什么 算法题、源码、计算机基础知识还是中间件原理
>
> 最重要的是学习他们解决问题的思想 看他们是在什么场景下 遇到了什么问题 用什么方式解决了这个问题 这样做的好处是什么
>
> 然后在**你的项目中遇到的类似的问题时 你可不可以用类似的方式去解决掉** 这才是学习的价值所在 我们学习不单单是为了了解知识 而是为了找到解决问题的方案和思路 我们上学的时候 做题也是为了找到做题的思路 而不是背题 然后期盼在考试中遇到原题 同样的 我们学习这些原理 也不是为了在面试的时候 能给面试官背出来 而是要学以致用 用到项目的优化中
>
> 举一些例子：
>
> 1. 比如说Kafka和MySQL里面都用到了顺序写盘
>
>   你知道了这个功能点后 需要去思考下
>
>   它为什么要这样设计 这样做是为了解决什么问题：写append-only的数据 同时避免随机写盘带来的性能消耗（当然主要是寻道耗时）
>
>   之后你可以看下你的项目里面有哪些数据的是具有append-only性质的 同时需要持久化 你也可以试着把这个技术应用到你的项目中
> 2. 再比如说304这个HTTP返回码 它表达的一般是缓存重定向 表明客户端存储的缓存的数据没有变化 还能继续用 所以不会重传数据 只会返回一个304的标识

> 一般情况下你看到这个知识点之后就过了 也不会想它这么设计的原因 顶多遇到了之后去查一下 但是你完全可以再进一步 想想它的设计原因 应用场景
>
> 设计原因：减少服务端向客户端返回的变化较少但体积较大的资源的传输 从而减少服务端和客户端还有网络传输的压力
>
> 然后比如说你的项目里面有类似的需要做一个大屏的功能 同样是客户端向服务端轮询一些字节数较大的资源（音视频这些静态资源或者其他的变化很少的动态数据） 你可以用相同的处理逻辑 返回给客户端静态资源时同时附带上资源最近一次更改的时间戳 然后客户端下一次轮询就可以带上这个时间戳 你一比对发现没有变化 就不用每次都向客户端返回全量数据 而是一个表明数据没有变化的标识就行
>
> 诸如此类的还有很多 你的解决问题的能力会随着你的知识积累和思考一点点地提高
>
> 我就是用学到的东西 把项目的核心处理服务的性能优化了好几倍（涉及到并发编程和亿级数据的高性能处理的相关问题）

- 算法题：

  这个很重要 不只是面试 算法和数据结构能力的提升对你整个人的逻辑思维提升 看源码 编码能力 看其他中间件 其他基础原理都有很大的帮助 过去一年中我花最大精力的部分就是算法 把学习算法的时间学习其他内容的话 我能学的东西应该是现在的2-3倍 但是我觉得还是很值

  普林斯顿那本算法书和算法导论我都看过一部分 但是get不到 不是很推荐 所以不要迷信经典 根据自己的生产力 选择适合自己的发展水平的内容

  理论：https://time.geekbang.org/column/article/39922 我争哥的专栏 上面的所有代码最好自己敲一遍 数据结构都自己实现一遍 大概需要花2-3个月的时间过完 他的算法课我也去报了 还可以 有些内容还是有价值的

  刷题List：https://github.com/youngyangyang04/leetcode-master 这是我见过最好的list 我一开始没按分类和难度刷 感觉算法很难 走了很多弯路 按照这个题解的顺序和内容刷3遍以上（我自己现在在过第3遍 刷错题） 基本上国内大厂的算法题卡不了你

- 系统设计：

  https://www.jiuzhang.com/course/77/ 九章的系统设计课 这个有点贵 但是内容没问题 教的一些设计解决思路 我都在面试和工作中用到了

  https://github.com/donnemartin/system-design-primer 还有一个github的资料 但是我没看过 但是也有推荐的人 有兴趣的可以去看看 关于系统设计这块的学习资源确实还是少

- 计算机基础：操作系统和计算机网络

  操作系统：

  哈工大李志军老师的书 https://book.douban.com/subject/30391722/ 是跟课程配套的 但是课程我没去上 书本身很好 结合linux源码讲解 也很薄 在我心中跟CSAPP是一个级别的好书 前置知识是汇编语言 也不难 可以去看下王爽的那本汇编语言 有兴趣的同学可以自己去跟着课程敲一个linux的demo出来（我之后有时间就会去做）

  计算机网络：

  这个我正在学习 所以没有多少可以推荐的资源 面向面试的话可以去看下小林的图解网络 https://mp.weixin.qq.com/s/_23WhJ9bOV9vjRq5EXsaXA 基本上面试的内容都能包含 小林的图解操作系统的第一章跟CPU缓存相关的内容和第八章网络系统也需要去看下 很多高性能中间件都用到了其中的原理

- 设计模式：

设计模式对你写高质量的代码来说很关键 https://time.geekbang.org/column/article/160463 推荐争哥的这个专栏 大概需要3个月的时间过完 设计模式部分的代码我都手敲过一遍 学习这个专栏的3个月 我感觉是我个人编程能力提升最快的3个月（把学到的东西不断应用到项目需求中 ）

- 中间件：

  MySQL：https://time.geekbang.org/column/article/67888 专栏内容很好 很多都是面试会问的 最关键的是你自己去实操一下 把上面的一些场景的例子 自己找个环境验证

  Redis：项目中没用到 所以没学 但是推荐去学下 面试八股常客

  消息队列中间件：同上 没用到 所以没学 但是推荐去学下你的项目用到的消息队列 面试八股常客

- Java：

  Spring：源码级别地学习Spring很重要 面试经常问 但是之前我没去学 因为当时不确定自己下一份工作是不是还是Java语言 我现在看的https://book.douban.com/subject/3897837/ 这本书我感觉还可以 不推荐极客时间上的小马哥的Spring课程 很烂

  Tomcat：这个面试中基本上不会去问你 单纯的是因为八股文里面没有 面试官也不懂 而不是这个中间件不重要 面向面试学习可以不用看 但是我个人觉得Tomcat比Spring做的事情要有意思的多 有兴趣的同学可以去看下 https://time.geekbang.org/column/article/94958 这个专栏上面的内容有些章节我都需要看2遍以上才能懂 但是也算是开启了我的源码阅读之路 源码阅读推荐在算法和设计模式学习之后进行 JDK的源码比较偏算法和数据结构 框架和中间件的源码比较偏设计模式

  并发：https://time.geekbang.org/column/article/83087 推荐这个并发专栏 上面的代码我都敲过一遍 讲到的有些并发相关的源码 Hikari Distruptor JUC这些我也去看了 那本Java并发编程实战我看过一遍 内容没这个好 看完之后 结合项目中用到的并发相关的类 去follow一下源码 学习他们的处理思路 基本上你会发现并发就那些解决方案 都不是很难 像CAS 线程私有 CPU缓存这些你自己就能用在项目中去解决一些问题

私货：

关于成长和价值观 可以去看看我偶像写的文章 不用加我好友 我就是个小废物 follow我偶像就行

偶像对我个人的影响很大 相信看过文章的人很多 但是愿意去践行的可能是少数

价值观：

https://coolshell.cn/articles/19271.html

https://coolshell.cn/articles/20276.html

https://coolshell.cn/articles/19464.html

成长路线：具体学习内容可以自己灵活选择 但是方向是没问题的 这个内容我从2018年刚开始接触计算机的时候看到现在 常看常新

https://time.geekbang.org/column/article/8136

同时推荐樊登读书这个App 可以去上面听一些其他领域的书 这个App对我影响也很大（笑）

可以优先听下 《刻意练习》 《意志力》 《终身成长》这几本书 （我反复听过很多次）

其他的关于脑科学和人文社科方面或者其他你感兴趣的书也可以听

不要在地铁公交上看手机 在一线城市 每天的通勤时间都不会少

最好是把通勤的时间利用起来 我是自己之前每天通勤时间1小时左右 在地铁上就是闭眼睛做英语泛听或者听樊登读书 实在很累了 听不进东西了 才会去玩手机消磨时间