2017工作小感

Feb 12th, 2018

从去年年初校招入职百度，到现在也有一年了。过去一年学习了很多，踩过一些坑，也成长了很多，非常感谢帮助过我的人，这篇主要分享下自己的一些思考和想法，也和大家共勉。

1.保持持续学习

这一点永远是老生常谈的问题，任何行业任何职业都需要不断学习，像迭代产品一样进行自我迭代，这一点非常重要。而且好消息是，就我个人经验而言，如果把获得作为y轴，学习的付出作为x轴，这条曲线不是线性的，是一条斜率越来越大的曲线，也就是说已经跑起来的人会跑得越来越快。由于知识通常是有关联的，并且具备迁移性（一个思想可以在多个问题里使用），所以在相同努力的情况下，你原本学得越多，那么收获会越大。这听起来像投资，其实学习本身就是投资，可以形成利滚利。如何衡量学习的成果？让自己的速度跟上摩尔定律，即每18个月自己的综合能力需要翻倍。

2.深入思考以及持续深入思考的能力

这个世界上有很多会深入思考的人，但能持续性地深入思考是一个非常稀缺的品质。我自己也在不断地练习，我通常做法是自己和自己辩论，自己怼自己：这个设计/功能为什么要这么做？为什么不那样做？那样做的话有什么潜在问题，需要处理什么tradeoff，这个做法是最好的了吗，有没有更好的？另外一个练习是我会在碎片时间把脑子里queue里面的问题拿出来思考。一个广为流传的“好做法”是用碎片时间来阅读，但这个做法并不适合我，碎片时间根本读不进书，读书需要大片成段的专注时间，还需要做笔记，这还不是主要原因，更重要的是我需要大量的时间来把我之前的输入内化成自己的东西，碎片时间刚好可以做这件事情，很多时候我在地铁/健身/洗澡/散步的时候把某个问题想清楚了，这种体验非常好。另外还有一点，做技术的同学通常都很认真敬业，同时也很容易沉浸于技术的海洋中，但这个世界是紧紧互相联系的，读各种各样的书，然后思考各种类型的问题能让自己的思路开阔很多。对于一个问题自己心里有了一个初步的想法，觉得暂时想不出什么了的时候可以去干别的事了，下一次再想到这个问题的时候，可能就会有一个比较清晰的思路了。

3.学会写工业级代码，即高质量的代码设计和高性能高稳定的代码实现

我从去年年初至今参与了brpc（百度内部的rpc框架，现已开源）的开发和维护工作，代码实现和抽象能力进步了很多。一个self-contained例子是，有一次需要给brpc加QueryRemover的支持：给定一个QueryString，能通过迭代器的方式来遍历key，然后决定是否删除当前的key，遍历完后返回修改后的QueryString。用naive的方式要实现这个功能其实很简单，但这不是我们追求的，这里要考虑的是：各种corner case/错误输入的处理，内存分配做到最优，性能尽可能好，当什么key都不删除的情况下应该任何

内存都不分配的。最后改了几次以后代码变成了这个样子。现在再回过头来看这段代码，脑子里已经不再是当初naive的想法了，而是intuitively这应该就是这么实现的。类似的例子还有很多，当自己已经尽全力写好一段代码但还是被code review打回，这是一件很好的事情，因为这说明要么是自己，要么是对方还有一些不知道自己不知道的东西存在，无论是哪种情况都会对一方产生正面的作用。看代码也是一个很好的学习方式，就我个人经验来说盲目地看代码很容易坚持不下来。一种理想的方式是项目中用到了某种技术，正好需要用到/借鉴某个开源代码，然后趁热打铁有目的地去学习源码会容易很多。怎么看代码也是有技巧的，先在脑子里想一下让自己来实现会怎么做，难点记下来，然后再去看代码作者怎么做。例如我前面提到的QueryRemover，有兴趣的朋友可以想一下让你来做会怎么做，然后看一下我们的实现。目前brpc有许多known issues需要开发和解决，欢迎大家领一个走然后给我们提pr :)

4.区分问题的表象和root cause

发现了一个bug，调试发现是某个模块A报的错，那这个bug的表象就是出在模块A，但有些时候root cause可能不出在模块A，真正的原因是模块B中一个已经隐藏了很久的race condition，甚至可能是一个kernel bug。而如何提高这个区分能力，一方面在于经验，更重要的一方面在于调试能力。调试其实是有一套固定流程/方法论的，首先分析现象（问题的表象），复现bug，如果是一个随机出现的bug，那就加大压力，给系统造成瓶颈，增加bug出现概率（减小工作线程数也可以给系统造成瓶颈，但不推荐这种做法，因为如果这是个多线程bug，那减少线程会降低bug出现的概率），竭尽所能以后如果还是无法复现，那就带log去线上去复现，等复现的时候要抓住现场，然后解决问题，再次上线验证，最后总结bug。每一次bug，特别是线上问题，都是一个很好的总结反思的机会，一定是流程上的某一环节出问题了。

5.对问题/数字敏感，一个优秀的开发者一定是敏感的

我的工作导师jamesge曾经和我说，一个程序在control^C退出的时候只要卡了一下下，就一定是哪里出问题了。还有类似的问题比如请求超时，很容易想到的解决方案是把超时时间调长，但某些情况下这只是掩盖了问题，如果有些本来应该很快返回函数处理地慢了一些，必有猫腻。不要拖，立刻去找到root cause，否则就是为以后埋坑。

6.锻炼抽象代码的能力，代码要尽可能低耦合高内聚，易拓展易维护

如果来了一个超紧急功能，我更倾向于的做法是好好评估一下，加下班做好代码抽象，而不是先粗糙实现然后加个TODO: refine this code，大部分情况下就渐渐就忘记了，变成以后的技术债了，更何况通常也不会有这种“今天开发明天上线”的需求。从软件开发的一开始就要保证每一次代码CheckIn都是高质量的，做好规范，一次写烂了就会发生第二次，然后慢慢地整个项目里都是烂代码，这就是破窗效应。在过去几年我C++写的比较多，以前有一段时间以为，功能本身的实现是需要关注的东西，而现在的观点是功能本身实现是开发中最基础最简单的东西，难的是代码抽象、性能压榨和对象的生命周期管理。另外不能偷懒，该重构的地方就立刻去重构，目的是降低代码的复杂度。让代码易维护的核心在于降低复杂度，这是一个很大的topic，推荐看《代码大全2》，最好的一本讲如何写好代码的书。

7.对自己设置高标准

曾经看到一个说法，叫面向离职编程，不是让你离职，意思就是说，你要把你的每一次代码提交，文档撰写、团队讨论都要以像在交接工作的态度来完成。写代码是写作的一个派生类，精神的传递是写作的目的之一，所以写代码同样也有这个目的，让别人看到你的代码能被惊艳到。能决定你能走多远的下限靠两样东西，基础和hardwork。当对自己设定了较高的标准，hardwork是一件为了达到标准而自然而然发生的事情，随之而来的是慢慢拥有owner意识，之后就会主动去思考目前系统的问题，然后提出问题并去推动。自己给自己找需求，自己就是产品经理，开发者，测试。当拥有一些owner意识以后，就会

去把玩目前的系统，corner case、压力、异常、极限情况下的表现都会去尝试测一下，没准线上就会遇到类似的情况。对于一个系统，别人看来已经完美了，正常运行不会挂了，但真正对问题把控到位的人是不会这么想的，他会感觉整个系统都是潜在问题，担心晚上随时会挂掉，第二天迫不及待起床要优化。

8.随时应对变化

不太喜欢打标签，比如你是前端，他是后端，更喜欢的说法是，我是一名开发，目前正在做XXX相关的事情。话里的意思是，随时准备好公司/环境/时代的改变，做技术转型，不要抱着自己已有的东西死死不放，那会是优势也是枷锁，同时也要意识到有些东西是沉没成本，一个理性人在考虑未来决策的时候不应该过多地考虑沉没成本。另外我认为现在是一个最适合学机器学习的时间，它不会像之前的安卓ios大数据那样成为一时的议论焦点，然后慢慢热潮退去，对于这一点我的行动是从两年前开始正经地系统学习，不算早但也不算太晚，就算以后不做这一行也要了解未来的趋势。现在最缺的不是研究人员，去看看每年顶会的投稿量就知道了，太夸张了，而是能把技术落地的人。

9.8/2法则

花20%的时间去了解一个领域80%，而不是花全部的时间去了解一个领域。你在一个领域花的时间越多，你的边际收益会越小，当边际成本大于边际收益的时候，如果没有特殊的理由，就该换一个领域去深入了，不同领域的思维碰撞没准也会产生更有价值的想法。

10.保持谦虚，保持开放的心态

从大了看，人类只是宇宙中的蝼蚁，而从长远看，人类的历史相对宇宙而言只是昙花一现，更何况我们每个人只是生活在自己有限的圈子里，没有理由去骄傲自大，有太多未知的东西了，自己引以为傲的东西可能就是别人的习以为常。我们唯一能做得，就是找到厉害的人，向他们学习，了解他们的想法，思考过后形成融合，让自己的想法和思路更加开阔，结果就是你会变得更厉害，然后会认识更多更厉害的人，形成良性循环。能看到和他人的差距然后去提升自己最后看到自己身上的成长，这件事本身就让人激动。

最后一点，也是最重要的：

11.自信自律，早早进行各方面的积累，形成马太效应

自信是，一件事还没有做呢，就知道自己可以做成。这和谦虚并不矛盾，谦虚的本质是保持开放和学习的心态。这里有个问题，怎么培养自己长期持续的自信？有一个小方法，建立一本自己的成功笔记本，把自己做成功的事记在里面，多小的事都可以，每当受挫失落的时候都拿出来看一下，就会原地复活的。当自信慢慢培养起来后，就会慢慢出现一种解决问题的惯性思维，可以理解为强者思维。遇到一个问题，拥有强者思维的人不会找借口，会去找要解决问题所缺少的东西，比如他会想，只要搞定了xx、yy和zz，那么这件事就搞定了。

未来的路还很长，认准的事不要过早放弃，2018希望各位朋友都顺利，生活开心。

（同时发布于知乎）

Posted by zyearn Feb 12th, 2018 Life

« 如何生成[0, N)中的随机数

Categories

- Algorithms (1)
- ComputerSystem (7)
- Life (2)
- Program (17)
- Reading (3)
- Thoughts (7)