# 61讲工作之余，专业之外

程序员的主流成长发展路线，是一个明显的“T”形线路。在纵深方向上，工作到一个阶段后，可能我们就会感到深入不下去了，而且越走会越有沉滞的感觉；在横向上，是广度方面，包括技术专业之外的领域，也会感觉了解甚少，短板明显。

有时候，要想产生真正的成长转变与发展突破，就不应自我局限于当下的工作内容和技术专业。

## 一、工作之余

工作，是技术发展纵深线中很重要的一个实践部分，但因为工作的内容和环境的限制，会把你困在一定的阶段，此时工作之余的内容将发挥很关键的作用。

工作之余，你都在做什么？我猜有人会说，工作已经够忙碌了，业余时间就该好好休息和娱乐了。的确，有很多人是这样选择的，但也有不少人不是的。即使再忙，有些人就喜欢在业余时间做点事情，这可能是一种性格特质，拥有这种性格和热情的人，总是能在忙碌的工作之余安排点其他内容，比如：

1. 看看程序设计相关的书、文章和博客；
2. 参加一些技术主题论坛或会议；
3. 写写技术博客；
4. 创建自己的业余项目（Side Project）。

以上前两条是接收和学习知识，第 3 条是总结和提炼知识，最后第 4 条则是实践所学，获得新的技能或加强旧的技能经验。

特别是第 4 条“创建自己的业余项目”，我感觉这是每一个程序员都应该做的事，为什么呢？在现实中切换一次工作环境是有比较高的成本的，开启自己的业余项目能帮助你打破工作内容和环境的限制，让你去做一些你喜欢做，但在工作中还没机会做的事。另一方面，业余项目也是你练习新技术和新技能的最佳试验场，相比你直接用真实的项目去实验，承担的风险和压力都要小很多，这样你也就有了机会去接触你想要学会的新技术。

记得几年前，我还参与过一个关于程序员业余项目的活动，那个活动的口号是下面这样的：

> 世界在被代码改变着，而我们在创造着代码。
>
> 仅仅是因为好玩，他开发了一款操作系统，连想都没想过，这会让自己有一天成为开源世界的领袖级人物。
>
> 只是想创造一个很酷的东西，所以他动手，坚持，因而有了让这个世界上的每一个人都可以免费地获取人类所有知识的百科全书。
>
> 成功者和其他人最大的区别就是，他们真正动手去做了，并且做了下去。

这也说明了业余项目的积极价值，而且这个世界上也有不少著名的产品来自业余项目的转正，比如：Gmail、Instagram、Slack，甚至包括 Facebook 本身。确实这些闪耀的例子激励着我们去尝试着各种各样的业余项目，但真正能做到像上述例子中那样光彩夺目，只怕这概率也和中头彩差不多了。

即使没有辉煌的成功，那么你做业余项目对自身还有什么积极的意义和价值吗？我想应该有的，你之所以要用自己的业余时间来开启一个业余项目，想必它是让你感兴趣的。全职工作的内容是你的职责，它支付你的账单；业余项目的内容则是你的兴趣，它满足你的好奇和探索之心。

在我学习写程序的前七八年里，业余时间也做了一些练习性质的项目。在 Github 之前的时代，Google 还能访问，我就在 Google Code 上维护了应该不止十万行的业余代码之作。后来 Github 兴起后迁移过来，不断练习重构优化和维护自己的专属工具库，删减了很多冗余代码，又新增了不少，剩下几万行代码。这个过程大约持续了七年，基本每年重构优化一次。每一次重构，都是对以前自己的否定，而每一次否定又都是一次成长。

**在做业余项目中最大的收获是：完整地经历一次创造**。这样的经历，对于程序员来说可能在很多年的工作中都不会有太多机会。写程序，实现系统，发布交付，仅仅是创造的一个中间部分。而完整创造的第一步应是确定你要创造什么，明确它，规划它，找出创造它的方向和路径，做出决策，然后才是下定决心去实现它。

一方面，业余项目只能在业余时间做，而业余时间又是那么有限，这样的时间制约决定了你只能走极简路线，要么保持足够简单，要么就可能陷入膨胀的泥潭，从而失控导致失败。另一方面，正因为业余项目不会给你带来直接的金钱收益，所以你选择增加的每一个特性，要么让你感觉有意思，要么能磨练提升你的手艺，打磨你的深度。

然而，大部分的业余项目最终都失败了，但这没什么关系，你已经从中收获了趣味与成长。

## 二、专业之外

专业是你的核心领域，而专业之外则是你的辅助领域；核心属于硬技能领域，辅助属于软技能领域，这也是“T”线中的横向延伸部分。

那么该怎样选择辅助的软技能领域呢？如果你的工作之余是在做一件业余项目，那么我想下面一些领域就是你在做业余项目之时更感缺乏的技能。

### 1. 创造与洞察

工程师，是一个创造者，创造模型来解决问题，但又不应该止步于此。

你的业余项目是你的作品，作品是创造出来的，按作品原始的需求是满足了作者创造的愿望，但业余项目要能取得成功就需要得到真正的用户，而获取用户就需要洞察，洞察用户的需要。

我记得以前读过一篇博文，来自著名 JavaScript 程序员尼古拉斯·泽卡斯（Nicholas C.Zakas，《JavaScript高级程序设计》一书作者），他写了几条职业建议，其中第一条就是：

> 不要成为做快餐的 “厨师”。

也就是说，不要像外卖接单一样，别人点什么，你就做什么。应该搞清楚你正在做的事情的价值和出发点，你不仅仅是实现代码，还要想想为什么要实现它。当你多想了后一步，在实现的过程中就会有更多的洞察。

开启过自己业余项目的程序员，已经走出了 “创造” 这一步，但多数还是失败在 “洞察” 这一点上。

**2. 表达与展现**

安安静静地写代码固然是不错的，但代码很多时候没法很直接方便地展现出你的真实能力和水平。

你可能会用 Linus 的“Talk is cheap, show me the code.”来反驳我。是的，也许在开源的世界，每个个人贡献者都隐藏在网络的另一端，他们只能通过代码来直接交流。但其他更多的现实是，当别人要来判断你的能力和水平时，通常都不是通过你写的代码，而是其他的表达与展现方式。

如果你的代码能给你作证，只有一个可能场景，那就是找到了大量直接使用你代码的用户，这就是成功开源作品的方式。否则，大部分时候你只能说你用代码完成了什么事情，做出了什么作品。

如果，你有好作品，就值得好好地展现，甚至还要不遗余力地推销它。

**3. 沟通与决策**

一个人的能力再强，也是有限的。当你想做更多、更大的事情时，就不可避免地要借助他人的力量，这时所面临的就将是大量的沟通了。

沟通一般有两个目的：一是获取或同步信息；二是达成共识，得到承诺。前者需要的是清晰的表达和传递，后者就需要更深的技巧了。这些技巧说起来也很简单，**核心就是换位思考、同理心，外加对自身情绪的控制**，但知易行难在沟通这件事上体现得尤其明显。

关于决策，如果都是在好或更好之间的话，那就真没什么纠结的问题了。而决策，是在优劣相当的情况下做出选择，更多的决策难点发生在取舍之间。程序员能碰到的大部分决策场景都是关于技术的，技术相对来说还有一些相对客观的标准来掂量，比如通过测试数据来验证技术决策的结果。

而其他方面的更多决策会让人陷入困境和纠结。如果要问我在这点上获得过怎样的教训，那就是：即使是一个坏的决策也比始终不做决策要好，因为在行动的过程中比“陷”在原地有可能产生好的改变。

决策和沟通有时是紧密联系的，大量的沟通之后可能产生决策，而决策之后也需要大量沟通来落地实施。

最后总结下：工作之余你可以有多种选择，但若被工作环境所困，导致专业力进境阻碍，可以开启业余项目来突破这种限制；而业余项目带来的诸多益处，从此也为你走向专业之外打开了一个新的视角与空间。

工作之余，专业之外，就是一条“T”线纵横交错发展的路线，当两条线都画得足够长了，在面临成长路上的断层时，才有机会与可能实现跨越。

那么，你的工作之余和专业之外都在忙些什么呢?

极客时间

# 程序员进阶攻略

## 每个程序员都应该知道的成长法则

胡峰 京东成都研究院 技术专家



精选留言