Joe Armstrong 面对面

原创 陈天 程序人生 2018-03-20 08:54

Joe 老爷子是 Erlang 世界里的一个图腾。

他书中的句子，他的公开演讲或是私下的谈话，都被不同的人到处引用 —— 今年的 code beam，好些 speaker 用 Joe 的原话来佐证自己的观点。Joe 因为 Erlang 的诞生被视作天才，视作英雄，视作传奇。但 Joe 自己却将其归功于 Ericsson 的「大度」：你知道 Ericsson 为什么把 Erlang 开源了么？因为有一天高层突然宣布：所有项目只能使用 C++ 或者 Java，不能用除此之外的任何语言。于是我们跟头头们商量，既然 Erlang 我们自己都不待见了，那干脆开源吧，头头说：随你便，我不关心。于是 Erlang 才得以摆脱 Ericsson 的控制，获得新生。

感谢 Ericsson 的愚蠢决定，否则这个世界便少了一门如此奇特而优雅的语言，而我也无法对半个地球外的一位老人顶礼膜拜。

如果你对 Erlang 没有了解，可以读一下我之前的文章：上帝说：要有一门面向未来的语言，于是有了 erlang。

在那篇文章里，我介绍了 Joe 的 worldview：

- everything is a process.
- process are strongly isolated.
- process creation and destruction is a lightweight operation.
- message passing is the only way for processes to interact.
- processes have unique names.
- if you know the name of a process you can send it a message.
- processes share no resources.
- error handling is non-local.
- processes do what they are supposed to do or fail.

他关于 Erlang 的所有想法，都或多或少和这个 worldview 相关。

跟老爷子坐在一起没聊多久，我就问了个「中二」的问题：「Erlang 在未来有没有打算引入 type system？」我知道 type system 会让 hot code reload 变得难以处理；也知道 type system 会让这门语言变得复杂，但就是莫名其妙问了这么一句。Joe 没有直接回复我，而是这样娓娓道来 ：

> John Hughes（haskell 的发明人之一）在谈起 haskell 时总会说：think type first。而我会说：think concurrency first。这是这两门语言的本质不同。我并不关心 type system —— **programming is not about code, it's about understanding**。良好的类型设计反映了设计者对所设计系统的理解，然而 type system 并不能帮助你更好地理解你所设计的系统。在一个对所处理的问题理解错误的情况下，一个人可以写出正常编译通过的代码，并且这代码是可以测试通过的 —— 因为作者对 problem 理解是错误的，因而他写出的 test case 也是错误的。一切都如此漂亮，可是软件却并未解决问题。

> 所以纠结 type system，寄希望于 type system 让你不犯错误（或者让工程团队不犯错误），这是在寻找银弹。

> think concurrency first。我们的世界本来就是并发/并行的。OOP 试图用一种错误的方式来描绘世界，因而得到的是复杂且不好理解的系统。比如一个电梯调度系统 —— 三部电梯停在各自的楼层，等待指令来决定如何运行。你如果试图用 OOP 描述电梯，为其设置属性，提供方法，其实是把问题复杂化了。三部电梯就是三个独立的 process，有他们自己的 state（所停楼层），它们把 state 通过 message 传递给调度系统，调度系统根据用户输入的 message，给最合适的 process 发消息，然后 process 决定自己如何行动。如果你 think concurrency first，那么，你对问题首先更容易理解，也更容易建模。

随后，老爷子一锤定音：

> if you want type system, use a different language.

整个谈话中，老爷子语速很快，往往我还在思考他抛出的上一个结论时，他就把下一个问题抛出来了 —— 这让你很难想象他已经是 67 岁的「高龄」。偶尔，他的思维会跳跃到物理学，比如 Minkowski space —— 跟一个差一点成为物理学家的计算机先驱谈话就是这么不着痕迹地在时空中跳跃。我问他还在写代码么，他给了我非常肯定的回答 —— 他说他在试图解决一些非常基础的问题。当我问是什么问题时，他狡黠一笑：你听了我明天的 keynote 就知道了。

我的好友孙博谈到白日梦旅行时，常说的两个词就是想象力和好奇心。想象力和好奇心是人类作为一个物种而言，最珍贵的品质。在聊天的过程中，Joe 老爷子的声音始终是柔和的，探寻的，他不断地抛出问题，像儿童一样处处充满好奇心，又像哲人一样通过问题启发你探寻事物的本源。他看问题的角度经常出乎我的意料，天马行空却扣着关键之处。讲一会，他会笑一笑，亲和地像一个邻家老爷爷，毫无架子，毫无世故，仿佛对面坐着的不是一个跪着的仰慕者，而是许久未见的忘年交。

聊着聊着，我渐渐轻松起来，问题也随意起来。我问他对这次 Code Beam 怎么看？他坦率地说听了几场，这届 speaker 不行。我会心一笑。他继续说 —— 人们在演讲

的时候往往忽略了问题，而直接给出答案。**people didn't really distinguish problem & solution. what's your problem? why are you doing this?** 在你的问题没有阐述清楚之前，我们之间无法达成共鸣，那么我为什么要关心你的答案？不解决问题，或者不解决我关心的问题（你要想办法让我关心），再光鲜亮丽的 solution 都是没有意义和价值的。我战战兢兢，汗出如浆 —— 因为，我自己的讲稿也是上来就是谈 tubi 的 elixir service 是如何构建，部署，升级和监控的，并没有谈我们遇到了什么问题，为什么要这么做，尤其是讲为什么要单做一个软件来做 release 时，非常突兀。好在，有了这次谈话，我正式讲的时候花了六七分钟来把 problem 和 why 讲清楚。

顺着 what's your problem 这个话题，他说做研究，找好的 problem 要 look at cracks on a marble floor：

> 我们现在有太多太多的编程语言了 —— python，ruby 这样如此相似的语言，为什么我们需要发明两个？有了 Java 为什么还要有 C#？大家把太多太多精力都放在构建一门更好的语言，可是，真正称得上有自己思想的语言很少。大部分语言可以被归到几个类目下面。如果把一门门语言看成一个 blackbox 的话，那么，大家都只顾自己的一亩三分地；如果把一台台机器看成一个个 blackbox 的话，那么，大家挤在一方小小的土地上厮杀，此消彼长。我们应该更多地 **build things outside the boxes**。在 blackbox 之间，广袤的领地，无人问津。blackbox 如何被连接在一起，如何 communicate & collaborate，还有大量的工作要做。

说到兴起，他继续提点我：look at cracks 也分场合，优先选择那些重要的 cracks。他给我讲了个 Altair 的故事 —— 补充一下：Altair 是 PC 时代的先驱 —— apple I 就是受其启发而发明的，而盖老师在 DOS 之前的主营业务 Basic，生意也是始于 Altair Basic。他说七十年代末，Altair 在 geek 群体间已经很火爆，有个混 synth 圈子的哥们寻思着为 Altair 做一款 synth 的工具，必定火爆。结果软件吭哧吭哧做出来，也就几个人感兴趣。所以你可以在一个小众的，还未被认识的领域有所作为，但不要试图在两个小众的领域的交集间有所作为。这样的话市场太小了，你熬不过黎明前的黑夜。

接下来我问老爷子怎么看 blockchain?

老爷子一下子声音提高了八度。what are you talking about? are you referring cryptocurrency? or open ledger? or smart contract? or something else? people usually mixed these things up with "blockchain"

我赶忙说，我们先聊聊 cryptocurrency，或者 bitcoin。你觉得...

老爷子打断我的话：bitcoin is morally wrong. Isn't it?

接下去差不多十分钟时间，老爷子就像拷问战犯一样，拷问 bitcoin，或者更严格地说，拷问毫无意义浪费资源，让地球变暖的 PoW。我好几次想插嘴，想把话题往技术上撩，都被老爷子弹开了。老爷子情绪激昂，我怕我再执拗下去，被放在火上烤的就不是 PoW 而是我 —— 毕竟，morally wrong 这顶大帽子是实锤，辨无可辨，因而我知趣地放弃这个话题，静静做个好听众。

> Bitcoin created a world of mess… As a software engineer, our responsibility is to **reduce complexity**, or to reduce entropy….

这句话我很认同，复杂是软件的天敌，我们搞这么多 principles，methodologies，paradigms，patterns，目的都是减少复杂度。我曾经写的一篇文章：是时候想想该怎么删代码了 提过一个思路：以构建可删除的代码为设计目标。这也是一种降低 complexity，减少软件中的 entropy 的方式。

在抨击完 bitcoin 道德上立不住后，老爷子显得有些疲态，我赶紧顺势把话题扯回到 Erlang —— 我还有一堆来自于朋友圈的问题有待发问。我挑了个简单的：你最喜欢 erlang 哪一点？

老爷子立刻精神起来，目光炯炯：

> 简单。你看我们处理 concurrency 的方式 —— 这是唯一一个把 security，isolation，fault tolerance 完整地，且又简单清晰地，涵盖在内的方案。我们八十年代解决掉的问题，大家现在还在穷尽心力去解决。

> 我一个快 70 的老头儿，如果从凳子上摔下来，我能 recover 我自己么？显然不行。我只能发个 signal（message）给你，然后你把我扶起来 —— 甚至，我都不用主动发，我们之间有 link（link 是 erlang process 之间的一种连接），使得你密切关注着我，以至于我一旦跌倒，你立刻得到视觉上的通知（就像 `{'EXIT', Pid, Reason}` 一样），于是你很快把我扶起来。这就是 fault tolerance，简单，明了，不用深奥的计算机知识佐证（说你呢，exception handling），小朋友都能懂。这个方案之所以浅显易懂，是因为这是很自然的 solution，我们每天就是如此生活。你再看，process 只能通过 message 通讯，所以 process 和 process 完全独立（isolation），就像一个个人。由此一个 process 的问题也很难扩展到整个系统，因而 security 也得到了保证。

「完全赞同！」我附和道。「而且 message passing 作为 concurrency 的一种方式，不仅适用于单机，也无缝适用于分布式系统。其它方案，单机内部一种模式，机

器间还是要 message passing，因而不得不在两种模式间切换，考虑起来很复杂。」

老爷子点点头。「一件事情如果过于复杂，那么一定是哪里出问题了 —— 大部分情况下是对问题的理解出现偏差」。老爷子话锋一转：

> 人们总是为没有想明白的问题创造解决方案，这是个严重的问题。比如说 JSON。what's the reasoning? 我不理解为什么我们需要发明 JSON，也许是 JSON 之前的交流方式太复杂，或者太不可用，所以它就出现了？像 JSON 这样的 serialization 方案，human readable 绝对是个伪命题。想想网络上传输的内容 —— 它们不过是 signal 而已，传输的是一个个 bit —— 那为什么我们要用 text，而不是 binary 来节省带宽，降低消耗？human readable 在屏幕上是有意义的，在网络上，在内存中，完全是无稽之谈。

聊了一个多钟头，快到午餐时间了。老爷子冲我狡黠一笑：你是不是准备了一堆问题？还剩什么问题想问的？我想出去走走，活动活动筋骨了。

我看了看我剩下的问题：

- What's the future direction of erlang/OTP?
- How do you see current status of distributed erlang? Any plan to improve it? for example, will we have gen_raft, or gen_pbft? or replace the full mesh network to p2p network?
- How do you compare erlang and elixir?
- How do you do trade off between performance and debugability, giving that OTP put so many efforts to make things traceable?

还有不少，我便挑了几个问他。

对 erlang/OTP 里加入更多的 distributed consensus behavior，如 gen_raft，老爷子是无所谓的态度。他说这不重要。他虽然有能力影响 OTP team，但这是他们的选择。对于 elixir，老爷子赞赏有加，说他有三个孩子，erlang 是第三个，elixir 是他所喜爱的小孙子。另外一个 erlang VM 上的语言，他的老友 Robert Verding 的 LFE（Lisp Flavored Erlang），就没那么好运气了 —— 老爷子说这是远房的侄子。对于 performance，老爷子看法一直未变：correctness is 1st, performance is the last thing I considered。这让我想起他曾经说过的：

> Make it work, then make it beautiful, then if you really, really have to, make it fast. 90% of the time, if you make it beautiful, it will already be fast. So really, just make it beautiful!

临了，我问老爷子对中国的印象。他说他很喜欢中国，很喜欢北京，还拿了他和夫人在天安门前的合影给我看。我说，那你打算什么时候再去中国看看啊？他笑笑，那看你们什么时候邀请我来中国参加有关 erlang 的大会啊。

iOS 用户赞赏通道：