“开源的同行评审是得到高可靠和高质量软件的唯一的可扩展方法。”

——《大教堂与集市》5分钟解释开源背后的运行机制

对很多人来说，开源社区的成功看起来像魔法一样不可思议，免费的高质量软件如果能不断生产出来固然很好，但在充满竞争和资源短缺的真实世界里，这能持续下去吗。

开源是怎么做到的？

对于开发者来说，他们为什么会投入大量个人精力在一件看起来没有直接回报的工作中？

对于企业主来说，将代码开源比闭源能带来更多的经济效益吗？

对于开源项目来说，如何才能保证其充满氧气和营养的run起来？

《大教堂与集市》的作者Eric.S.Raymond（以下简称ESR）是开源软件运动的旗手。他是INTERCAL编程语言的主要创作者之一，曾经为EMACS编辑器作出贡献。 ESR还是著名的Fetchmail程序的作者。他还编写了一个最初用于Linux内核设置的设置程序。

《大教堂与集市》一书包括了著名的“Eric.S.Raymond五部曲”，这“五部曲”第一次总结了自由软件和Linux的成功经验，第一次将黑客文化、自由软件运动及其发展规律总结成文，更是第一次将开源社区背后的运行机理用社会学、心理学、经济学和人类学加以解释，并以浅显易懂的方式呈现出来。可以说ESR的这五部曲解答了参与开源中面对的方方面面的问题，更预测了可能出现的问题，并给出可行的解决的方案。

附件中我上传了《大教堂与集市》一书的电子版本，大家可自行下载阅读，保证受益匪浅。

如果没有时间，或者想快速获得我在文端提出的几个问题的答案的同学，可以继续阅读接下来的内容，我将我认为最有启发性的段落简单整合，刨除了一些集体无意识的基础理论，让大家可以迅速的get到开源的神奇力量。

1. 黑客和骇客有什么区别？

在书中，作者将优秀的程序开发者统一称为黑客，要解释一下黑客(hacker) 不是骇客（cracker）。黑客是褒义的，黑客构建了互联网，黑客造就了现如今的UNIX操作系统，黑客运转起了Usenet，黑客让WWW发挥作用。如果你是这个文化的一部分，如果你对这个社区有贡献，社区中其他人知道你并称你为黑客，那你就是一名黑客。

有一群人高调声称自己是黑客，但他们并不是。这些人（大多是些毛头小子）的主要目的是攻入他人计算机或者破解电话系统。真正的黑客称他们为“骇客”（cracker），而且完全不想搭理他们。多数真正的黑客认为，骇客懒惰，缺乏责任感，而且不是很聪明，如果你的目标是能够攻破安全系统，那不会让你成为黑客，就好比学会热线（hotwire）发动汽车并不会让你成为汽车工程师一样。遗憾的是，很多记者和作者错误地使用“黑客”一词来描述骇客，这使得真正的黑客极为不满。

两者最根本的区别是：黑客搞建设，骇客搞破坏。

1. 对于开发者来说，他们为什么会投入大量个人精力在一件看起来没有直接回报的开源项目中？

对于程序员来说，在同侪（其它程序员）中拥有好名声、获得声望是非常有益的，甚至可以说，同行对于你的认可程度决定了你的价值。

首先，也是最明显的一点，在同侪中拥有好名声是一种最基本的激励，出于人类进化上的原因，我们都渴望体验它。

第二，声誉是很好的吸引他人注意和合作的途径。如果一个人慷慨、智慧、公平交易、有领导能力，或者有其他优秀品质，就更容易让人相信和他合作能获得好处。而这些品质在一个开源项目中可以体现。

第三，如果你所处的礼物经济和交换经济或命令体系互相关联交织，你的名声就可能传播到后两种环境中，使你在那里获得更高的地位。简单解释下，如果你在一家公司中做程序员，同时你也从事开源项目，那么开源项目的成功也能帮助你未来获得更高的薪资待遇和权力地位。

比起“真实世界”，黑客文化中的一些特殊情况使声誉有着更高的价值。这是因为写程序这件事情，不像实物或者金钱，其价值体现很不明显，也很难客观区分的好坏，所以，黑客的地位竞争能否成功，很微妙地取决于同侪们挑剔的评价。

1. 在一个开源项目中，需要怎么做、做什么会让你更加牛逼？

总体的牛逼级别如下排序：做一个功能被纳入发行版本中>做一个牛逼的功能还被发行版本收录>重大或难以寻找的bug修补>简单的调试和写文档

评判具体标准如下：

1. 如果它不能像我所预期的那样工作，那就不是好的——不管它多么聪明和有原创性。

这条规则使得开源软件倾向于长期停留在beta版，开发者只有在确信软件不会有很多问题时，才会发布1.0版。开源世界的1.0版意味“开发者愿意拿自己的名誉赌它好使”，而闭源世界的1.0版则意味着“如果你很谨慎，不要用这版”。

b. 拓展性工作要比在某功能内（对现有作品）的重复性工作好。

对这条规则的简单理解是：原创性工作比功能复制要好。但实际上不完全是，如果你复制的是闭源软件的功能，并借此破解了封闭的协议或格式，使得这个领域在开源世界里也可使用，那你的工作就和原创作品一样值得赞许。

c. 能进入主要发行版的作品比不能进入的好。在所有主要发行版中都包含的作品最令人尊敬。

主要发行版不仅包括大的Linux发行版如Red Hat、Debian、Caldera和SuSE，也包括其他一些发行版——如BSD发行版或自由软件基金源码集，由于发行者会维护自己的声誉，其质量也自然能得到保证。

d. “使用”是最真实的赞美，类别杀手比同类竞争者好。

信任别人的判断是同侪评价的基础。这很正常，因为没人有时间去仔细评判所有可能的选择，所以人们会认为：多数人使用的作品优于那些少数人使用的。

如果作品好到没人再想使用其他备选，作者将会获得巨大的威望。那些被最广泛使用的原创型类别杀手，会被纳入所有的主要发行版中，并获得最大可能的同侪尊重。成功做到这点超过一次的人，将会被人们半开玩笑、半认真地称为“大神”（demigods）。

e. 相比那些只挑有趣和简单工作的人，长期致力于艰苦和乏味工作（如调试、写文档）的人更令人钦佩。

这条规则描述社区如何回报那些黑客天生不愿意做但实属必要的工作，某种程度上它和下面这条矛盾：

f. 重要的功能扩展比低层次的修补好。

这条规则似乎是针对一次性工作的评价。相对于修补bug而言，给软件增加功能特性有可能得到更多回报——除非这个bug异常令人厌恶或者难以寻找，因为将这种bug找出来本身就证明了非凡的技术和才能。但当这些工作是长期行为的话，一个长期关注和排除bug（甚至是普通bug）的人，其地位要高于那些花费相近工作量在增加简单功能上的人。”

1. 为什么说开源的同行评审是得到高可靠和高质量软件的唯一的可扩展方法？

根据上面所论述的，参与开源项目可以给程序员带来最有价值的同侪声誉，那么更多牛逼的程序员就会聚集在开源项目上，通过同行评审提高它的质量。

相比于传统、闭源、严格管理模式，开源是靠“玩”来创造价值。

这种情况下，开源黑客通过“自我组织”来最大化生产力就显得加倍重要，自愿者自主选择项目，社会环境则无情地选择能力。开源之所以成功，部分原因是开源文化只接受编程人员中那最有才华的5%。（因为只有最好的内容才能被纳入一个开源项目中）。最有才华的程序员和那些刚刚及格的程序员之间，生产率能相差100倍。

人们常常会因为这种差异提出一个棘手的问题：对单个项目或者整个行业来说，撤离出那50%能力较差的程序员会不会更好一些？思考已久的管理者们早就明白，如果传统软件管理的目的仅仅是把能力最差那部分人的净损耗转变成微弱盈余的话，那问题就简单多了。

开源社区的成功使得这个问题更加尖锐，强有力的证据表明，从互联网上招募自主选择的志愿者，通常更便宜、更有效。而传统方式管理的那些写字楼里的人，其中有很多可能宁愿去干点别的。”

如果传统、闭源、严格管理模式的软件开发真的想靠这种由“无聊”部分组成的马其诺防线来防御，那么它之所以在某个应用领域能继续生存下去，只是因为还没人发现这些问题是真正有趣的，并且还没人发现迂回包抄的路径。一旦有开源力量介入这些领域，用户就会发现终于有人是因为问题自身的魅力而去解决它的，就像其他所有需要创造力的工作，若论激励效果，问题自身的魅力比单纯的金钱要有效得多。

单纯为解决动机问题而设立一个传统的管理架构，可能战术不错，但其战略是错误的，这一套在短期看会有效，但长远来说一定会失败。

“现在看来，开源成功的一个最重要成果，就是告诉我们，“玩”是创造性活动中最具经济效能的工作模式。”

1. 目前的软件企业将产品闭源的危害是什么？闭源软件真的是靠卖软件赚钱的吗？

被编写用来出售的代码仅仅是编程冰山的一角，大部分程序员的工作是在维护、排错和扩展上。常见的价格策略——给软件定以较高的购买价和相对较低甚至为零的维护费——就必然导致很差的服务。

其实，软件很大程度上是一个服务行业。

传统闭源买软件的模式是一种工厂模式，它鼓励软件供应商生产出“存架软件”——市场销量很好但没有实际价值的软件。因为如果以固定的销售价格支付永无期限的服务费用，那么企业卖出去的越多，维护成本就会呈现更高的指数型增长。

如果不用工厂模型，用什么？要想有效调整软件生命周期的真实成本结构，我们需要一个建立在服务合约、按期订购以及在供应商与消费者之间价值持续交换基础之上的价格结构。ERP（企业资源规划）系统这种最大商业软件产品的价格结构正是如此，由于其开发成本过于庞大，固定采购价格不可能完全覆盖，像Baan和Peoplesoft这样的公司实际上是通过售后咨询赚钱。在追求效能为前提的自由市场中，我们可以预测，在软件行业成熟之后，这将成为大多数厂商最终遵循的价格结构。

以上分析让我们明白，开源软件不仅在技术上，而且也在经济上挑战着现有秩序。将软件“免费”所带来的效果，看来会强制我们进入以服务费为主导的世界，并让我们清楚了解，闭源软件一直以来依靠软件销售价值的做法是多么脆弱。

这里证明了，销售软件的核心代码并不能让企业主一劳永逸的赚钱，更可持续的收费模式是收取服务费。

1. 企业主选择闭源可能是想要保护核心技术，防止竞争，形成垄断，如果开源了，这些优势不都全没了吗？

假如你雇人给你的企业写了一个专业的财会软件，选择闭源不会比开源好到哪里，如果你希望闭源，唯一合理的原因是你想把这个软件卖给别人，或者防止竞争者使用它。

答案很明显，你在保护销售价值，但95%的软件写出来是供内部使用的，在没有销售价值可言的情形下，从闭源中还能得到什么好处？

需要稍微分析一下的是第二种情况（保护竞争优势），假设你把这个财会软件开源了，它变得很受欢迎，在社区的帮助下它也变得更好了。你的竞争对手也开始使用它，竞争对手没有支付开发成本却获得好处，而且还影响到你的业务，这是不是一个反对开源的理由？”

也许是，也许不是。真正应该考虑的问题是：你从分散开发负担中获取的益处是否超过了因（“搭便车”行为导致的）竞争加剧而带来的损失，一些人往往在这个权衡中失算：

(a)忽略了社区开发带来的功能改进。(b)不把已经支出的开发成本当做沉没成本。根据假设，不管怎样，你都是要付出开发成本的，所以把它归入开放源码（如果你这样认为）的成本是不对的。”

我们前面已经论证过“开源的同行评审是得到高可靠和高质量软件的唯一的可扩展方法”，所以即便有竞争，企业主的开源软件在更多有智慧的人贡献下，获得的收益也是有可能超过所谓的核心代码的开发成本的。

我们再来看所谓的核心代码和垄断。

如果你是硬件供应商，你可能害怕开源会泄露产品如何工作的重要细节，会使竞争者趁机复制并获得不公平的竞争利益。在产品周期长达3年到5年的那些年代里，这个观点还算说得过去，但现如今，竞争者花在复制和理解上的时间，将会占据产品周期的很大一部分，他们本该把时间花在创新和考虑如何让产品差异化的。”

这个观点并不新鲜，前克格勃首脑Oleg Kalugin说得很好：“比如说，我们计划窃取IBM这类公司或者其他电子领域的先进技术，由于西方在这方面远超我们，我们需要花费数年让这些情报成果得到实现。而那时，大概五年或七年，西方又往前走了，我们只能跟着一偷再偷，而且会落得越来越远”

这说明什么？在短期内看，将驱动程序保密的做法似乎还行；但从长远看，这可能是个糟糕的策略（尤其当竞争对手已经开放源码时）。如果你不得不保密源码，可以把代码烧到板子上的ROM中，然后发布ROM接口。你要尽可能地开放，要建立市场并向潜在客户展示你的自信——你在思想上和创意上超越竞争者的自信。

如果你坚持闭源，你可能要面对最坏的结果——你的机密还是会被曝露，你不会得到免费的开发帮助，你也别指望你的对手会傻到花时间克隆你的东西。最重要的是，你错过了及早获取认可的阳光大道。一个巨大而又有影响力的市场（那些有效运转整个互联网和多数商业数据中心的服务器管理员们）会因为你的无可救药和顽固抵抗而把你的公司打入冷宫。你不懂为什么要开放，没关系，他们会从懂的人那里买板子

1. 对于企业主来说，将代码开源比闭源能带来更多的经济效益吗？

相比于闭源的销售模式，软件开源对于企业主来说，是建立一种非直接的销售价值模型。

这种模型有很多优势：

1. 占领市场。

在这种模型中，你可以利用开源软件建立或维护专有软件（能产生直接收入）的市场地位。最常见的形式是，将客户端软件开源，用来帮助服务器端软件的销售，或帮助门户站点获得订阅及广告。”

b. 硬件糖霜

这种模式适用于硬件制造商（这里的硬件包括网卡、外围设备乃至整个计算机系统）。市场压力迫使硬件公司编写和维护软件（设备驱动程序、配置工具乃至整个操作系统），但软件本身不是盈利中心，它是一项开销——并且常常是一项很实在的开销。

这种情况下，开源是最好的选择，因为并没有收入流可以损失，也没有什么负面效果。供应商可以获得快速膨胀的开发者队伍，对客户需求会有更快和更灵活的响应，并因同行评审获得更好的可靠性。它会被免费移植到其他环境下，可能还会获得客户忠诚度的提高，因为客户的技术团队会投入更多的时间根据其需要改进代码。

c. 送配方，开餐馆

开源软件不是为闭源软件而是为服务占领市场。

Cygnus Solutions最先使用这个模型，大概这是开源的第一次商业行为（1989）。当时，GNU工具集提供了一个可用于多种机器的通用的开发环境，但在不同的平台下，每个工具的配置过程都不一样，并且要打上不同的补丁包。Cygnus改良了GNU工具集并写了一个“配置”脚本来统一构建过程（也即配方），然后，他们出售这个版本的GNU工具集以及与之绑定的二进制软件及支持服务（也即餐馆）。遵从GPL要求，他们允许客户自由使用、发布和修改这些软件，但若使用支持服务的用户超出合同所承诺的数量，服务合同将会中止或需要支付更高的费用。

这同样是Red Hat和其他Linux发行商所采用的做法。他们真正销售的东西不是软件，不是那些比特，而是他们所提供的附加价值：他们通过集成和测试，向客户提供一个不断发展的操作系统，保证它是商业级的（要是再低调点就好了），并可以和该品牌其他操作系统向后兼容。他们还提供免费的安装支持和可选的持续支持合约。

d. 附属物策略

人们出售开源软件的附属物，如低端市场上的杯子或T恤，高端市场上的专业书籍或文章。

附属物策略的一个很好例子是O'Reilly&Associates公司，它是很多优秀开源软件参考书的出版商。为提高在市场上的声誉，O'Reilly实际上雇佣和支持了一些著名的开源黑客（如Larry Wall和Brian Behlendorf）。

e. 当下收费、未来免费

“这种模型以封闭许可证的形式发布软件的二进制及源码，但其中的封闭条款有过期时间。比如，你可以在许可证中允许自由发布、禁止免费的商业使用，并保证在发布一年后或者供应商关张后，软件将遵循GPL条款。

这种模型的好处在于，它使顾客确信产品是可以客户化的——因为他们有源码，而且产品是“始终可用”的，因为当原创公司消亡后，开源社区会接管起来。

f. 软件免费，品牌收费

“这是一个探索性的商业模型。你可以开源一项软件技术，保留其测试套件或兼容性标准，然后卖品牌认证，如果某公司的产品通过认证，则表明他们对该技术的实现和其他拥有该品牌认证的产品是兼容的。

这是Sun公司应该对待Java和Jini的方式。

g. 软件免费、内容收费

这是另一种探索中的商业模式。想象一个类似股票报价订阅的服务，其价值既不在客户端也不在服务器端，而在于提供客观可信的信息。所以，你可以开源软件而销售内容订阅。当黑客们把客户端移植到新平台或以各种方式做了改进时，你的市场也自然而然地扩展了。

（所以AOL应该开源其客户端软件。）

6. 什么样的源码适合开放？

1.可靠性/稳定性/可扩展性非常重要。

2.除了独立的同行评审，没有其他便捷易行的方法验证设计和实现的正确性。

3.该软件对客户的业务非常关键。当软件对消费者越来越重要时，消费者会在理性上希望避开垄断供应者，这导致他们对开源的兴趣变大，开源供应商的市场竞争力会因此增强。

4.该软件创建或运转一个公共计算或通信基础架构。

5.关键方法（或能实现同等功能的方法）属于公共知识。”

7. 对于开源项目来说，如何才能保证其充满氧气和营养的run起来？

开源项目虽然是在一定程度上很自由，但如果想要让它持续但run起来，管理是不可缺少的。而管理的核心问题是所有权和激励，不过这两点其实是相辅相成的，因为让项目私有者决定谁该被激励、谁该被责备是最通用简单的方式。

这种所有权，一方面是控制意义上的所有权，另一方面是声誉回报意义上的所有权。项目的所有权人需要维护该项目，在有冲突时有权做出强制性决策，同时，他也通过交易一部分声誉回报来换取他人的工作。这让人不禁联想到与之非常类似的佃农耕作，当然这点除外：即便贡献者不再为项目工作，其名字仍然保留在名誉表上，并能继续“赚取”一定程度的声誉。

那么所有权一般分为三种方式，现在最有效的方式是“善意独裁者”模型（如Linux内核），独裁者不一定是一个人，也有可能是一个拥有共同偏好的几个人项目团队。另外一种就是投票委员（如Apache），这种方式常用在缺乏优秀领导当情况下，但其产物常常过于复杂、内部不一致、缺乏一直的愿景。还有一种是Perl的开发者模式，即在资深合作者内部轮流掌权，就是由子系统的责任人轮流坐庄。

我们已经看到设计权的分布和所有权的分散会导致项目角色复杂性增加，虽然它是一种有效的激励分配机制，但同时也稀释了项目领导者的权力，稀释了领导者平息潜在冲突的权力。

另一个解决冲突的方式是“资深者胜”——如果两个或更多个贡献者有分歧，该分歧在客观上很难解决，并且谁也不拥有该分歧的领土权，那么在整个项目中投入工作最多的一方（也即在整个项目中拥有最多领土权的一方）胜出。

当然，若想让这些冲突解决机制生效，需要整个黑客社区愿意执行它们，唯一可用的执行机制是“批判”和“放逐”——公开谴责那些破坏习俗的人，并拒绝和他们再次合作。

把以上问题弄懂，你就会发现，开源不仅仅是一件自由主义、开放共享的概念意义，它具有实际的经济效益和发展潜力。对于程序开发者来说，它更是积累影响力、提升自我价值的有效手段。

当然，也不能说闭源就是错误的，开源和闭源并不是水火不容，短期之内谁也不能取代谁，最重要的是要看到在未来发展过程中哪种方式能够更好地解决问题。

最后，开源社区是一个规则社区，想要做开源，习惯于遵守规则是一切的基础，想要成为大师，最有效的方法是模仿大师的思维习惯——无论是在理智上，还是情感上。