

# BoHolder的网站：博客，小玩意及其他



博客



分类




关于




## 译文：从源码中学习（阅读源码，初学者的有效成长方式）

为什么我们需要阅读源代码；如何阅读代码，并尽可能从源代码中学到更多

 2021.5.25

 2023.8.8

 programming

 4197

 9 分钟

## 目录

1. 前言
2. 为什么我们需要读源码
  - 2.1 站在巨人的肩膀上
  - 2.2 解决困难问题
  - 2.3 扩展你的边界
3. 应该读什么样的源码
4. 如何读源码
  - 4.1 预先准备
  - 4.2 流程与技巧
    - 4.2.1 结合上下文阅读代码

- 4.2.2 把实例跑起来并与之交互
- 4.2.3 了解数据结构间的关系
- 4.2.4 了解模块间的依赖关系与边界
- 4.2.5 使用测试用例
- 4.2.6 点评

5. 一些好书

6. 译者的话

这是一篇译制博客文章，原博客为 [Learn from Source Code \(an Effective Way to Grow for Beginners\)](#)，原作者为[Nick Mose](#)。我向 Nick 申请获得了此文的中文翻译权与中文翻译文章发布权。

## 前言

上周我在和一位年轻程序员聊天时，他问到我：“如何阅读源码？”，我们讨论了一段时间，我还列举了几种有效阅读源码的方式。然后他说：“你应该就这个话题写篇文章，这对初学者很有帮助，而且这种经验无法从书籍和教程中获得。”那么开始吧，下面是我关于阅读源码的小技巧。

## 为什么我们需要读源码

我们程序员每天都要和源码打交道。经过数年的学习，大多数程序员可以“写”代码，或者至少是拷贝并修改代码。而且，我们教授编程的方式强调

编写代码的艺术，而不是如何阅读代码。当我说“阅读代码”，我是指**有意地专门阅读代码**。

众所周知，编程和写作有诸多相同之处。唐纳德·克努特甚至引入了文学编程 (literate programming) 编程范式。编程与写作有相同的理念：表达我们的想法。还记得你在学校是怎么学习写作的吗？我们的写作能力来源于从小学开始直到现在的大量的文本阅读。多年以来，我们阅读了不同难度的伟大作家的作品，并练习了多种写作技巧。

“如果你没时间读，你就没时间（或工具）写，就这么简单。” —— 斯蒂芬·金，《写作这回事：创作生涯回忆录》

正如斯蒂芬·金所观察到的那样，一个作家必须广泛而频繁地阅读，才能形成自己的声音，并学会写出促使读者拿起书并痴读的句式和故事结构。**和读书一样，有意地阅读代码可以帮助程序员加速成长，尤其是对中级 (intermediate) 程序员而言。**这样做有三个好处。

## 站在巨人的肩膀上

我们从他人身上学习。优秀的源代码就像文学杰作，它不仅仅只提供了知识和信息，还提供了启迪。

通过浏览 Linux 内核、Redis、Nginx、Rails 或其他著名项目，你可以从全球范围的成千上万的顶级程序员那里汲取智慧。在这些项目中可以找到无数的良好编程示例、编程范式选择、设计和架构。向他人学习的另一个好处是能够避免常见的坑，大多数坑早已被他人踩过。

## 解决困难问题

在你的职业生涯中，你终将会碰到谷歌都无法解决的问题。如果你还没碰到过这种问题，这只是因为你的编程的时间还不够长 :)。阅读源码是调查这类问题的好方法，也是学习新东西的好机会。

## 扩展你的边界

大多数程序员只在少数特别领域编过程。一般而言，如果你不时常推自己一把，你的编程技能会维持在你同事间的平均水平。不要满足于修补 bug 或在现有系统中添加琐碎特性的工作。相反，你可以试着扩展到一个新的领域，持续尝试找到一个你在日常工作中接触不到、但你感兴趣的领域。这将从整体上拓宽你对编程的理解。

## 应该读什么样的源码

综上，阅读源码是有益的。那么下一个问题，有这么多优秀作品可供选择，我们该选择并阅读什么样的源码呢？你必须从选择目标开始。如果不在这个步骤上下点功夫，你从源码中学习的效果就会打折扣。这里有一些典型场景：

- 当你想学习一门新语言。学新语言可不只是学会语法。不管怎样，阅读源码是一个非常有效的学习新语言的方式。我从rust-rosetta项目中学到了很多 Rust 语言知识。Rosetta Code 是一个收集同一批通用任务在不同语言上的解决方案的项目，这是一个可用来学习新语言的有用资源。
- 当你想了解一个特殊的算法或实现。例如，我们都会使用标准库中的 `sort` 函数，你有没有好奇过它是怎么实现的？或者当你要使用 Redis 中

的 Set 结构，它是用什么数据结构实现的？为了解决这些疑惑，你只需要读源码中与之相关的实现部分，通常只有很少的文件或函数。

- 当你在特殊的框架中编程。这意味着你对该框架已经有了一定的经验，这是个阅读一些框架本身的源码的好机会。很显然，了解框架的源码有助于提高你对框架的理解。
- 当你想拓展进入新的领域，你可以阅读这个领域的经典著名的项目的源码。比如说，如果你在做 Web 开发的工作，你对分布式系统感兴趣吗？如果你的答案是“是”而且你懂 Golang，也许 etcd 是你的选择。你想钻研操作系统的内部构成吗？那么也许 xv6 是一个好的开始。我们处在一个许多优秀开源项目都托管在了 Github 的好时代，请试着寻找一些这种项目。

记住，**选择与你当前的编程技能与知识水平相当的项目**。如果你选择了远超你当前技能水平的项目，最终你会感到沮丧。读一些相对较小的项目，接着读更大的项目。如果目前你不能理解某些特定的代码片段，这意味着你有个知识缺口(knowledge gap)。把代码放到一边去，试着读一些相关的书、论文或其他文档，当你更有信心时再回来接着读代码。我们总能在一个模式中取得进展：读（代码、书、论文），写，更多的读，更多的写。

## 如何读源码

《How to read a book》 是一本指导人进行明智地阅读的书。作为初学者，我们值得投入时间和精力去思考我们应该如何阅读代码。**阅读代码不是件容易的事**。光是阅读源码是不够的，你要试着去理解他人的设计和想法。

## 预先准备

为了更有效率地阅读代码，你需要提前在手边准备这些东西：

- 一个你可以熟练使用的编辑器。你需要拥有快速搜索关键字或变量名的能力。有时你需要查找函数的引用和定义。和你的编辑器相处融洽些。为了更加有效率，试着学习仅使用键盘操作编辑器。这会使你专注于代码而不受打扰（译：指额外思考编辑器操作）。
- 掌握基本的 Git 或其他版本控制工具的技能，这样你就能比较代码在版本间的差异。
- 与源码有关的文档。文档可以为你的阅读提供参考，尤其是设计文档、编码规范等文档。
- 具有一定的编程语言与设计模式的知识 and 经验。这对（阅读）大项目是强制性的。如果你很了解一门编程语言，你也会了解关于源码组织与编程范式的最佳实践。当然，这需要时间来积累。要有耐心。

## 流程与技巧

阅读过程不是线性的。你不会就那么一个接一个地读源文件。相反，大多数时候我们会从顶到底地阅读代码。下面是一些更有效率阅读代码的小技巧：

## 结合上下文阅读代码

当你阅读代码时，请持续提出问题。例如，如果一个应用有缓存策略，一个好问题就是：如果键无效了会怎样？缓存中的值如何更新？带着这些问题阅读代码，就是结合上下文。或者说因为你有了一个目标，你会变得享受阅读的过程。你甚至可以自己做一些假设，然后在代码中寻找验证。

你有点像侦探：**你想发现代码的真相，代码的逻辑，代码是如何像故事一般上下流动的。**

## 把实例跑起来并与之交互

源码就像乐高积木，只是已经组装好了。如果你想了解它们是怎么组装在一起的，你需要和它交互，有时甚至要把它拆开。阅读同一模块的老版本同样有帮助。从 Git 中阅读版本差异，试着弄清楚特定的特性是如何实现的（修改日志在这个场景很有用）。举个例子，我发现 Lua 的第一个版本相当简单，这可以帮助我了解作者最初的设计理念。

Debug 是另一种与代码交互的方式。试着在代码中加一些断点（或打印一些变量值），然后弄明白打印到控制台中的所有输出。

如果你对代码了解比较透彻了，试着对代码做一些修改，重新 build 并把它跑起来。最简单的方式是试着调整配置项，去看不同配置的运行结果。之后你可以试着添加一些细微的特性。如果这些特性对其他人也有用，你应该把代码贡献到上游。

## 了解数据结构间的关系

“糟糕的程序员担心代码，优秀的程序员担心数据结构和它们的关系。” -  
Linus Torvalds

数据结构是一个程序中最重要元素。用笔或者你喜欢的其他工具画出数据结构间的关系。这个图就是源码的映射。你会在阅读过程中时常参考这个图。一些工具比如 scitools 可以用来生成 UML 类图。（译：这个方法用在写代码中能节约翻 Model 声明文件的时间，推荐用纸笔，不占屏幕）



## 了解模块间的依赖关系与边界

大项目中会包含许多模块，一个模块经常只拥有单一职责。这有助于我们减少代码复杂度，在适当的层级上做抽象。模块的接口是抽象的边界，我们可以一个接一个地阅读模块。如果你在阅读一个使用 Make 构建的C/C++项目，Makefile 是了解模块间如何组织的好切入点。

边界本身也很有用。优秀的代码组织得很好，变量名与函数名的命名风格体现着可读性。你不需要阅读全部源文件，你可以**忽略不重要的或你熟悉的部分**。如果你确定一个模块是仅仅是为了被解析而设计的(just designed for parsing)，那么你已经大致了解了它的功能；那么你就可以跳过不读这个模块。当然，这将大大节约时间。

## 使用测试用例

测试用例也是帮助代码理解的一个很好的补充。**测试用例就是文档**。当你在阅读一个类时，试着把对应的测试代码一起读了。测试用例能帮你弄清一个类的接口，和该类的典型用法。集成测试用例可以让你顺着走过程序的整体流程，适合输入一些特殊值并 debug 运行。

## 点评

为什么不在花了不少时间阅读一个项目后，写一篇代码点评呢？就像写一篇书评一样。你可以写下代码中好的和不好的部分，还可以记下你从中学到了什么。撰写这类文章可以帮助你阐明自己的理解，也有助于其他人阅读源码。



# 一些好书

我发现阅读代码是一个远超我想象的广泛话题。没有系统性训练该技能的方法。总而言之，不断练习，找到你自己的方式。下面是一些帮助你提升代码阅读能力的好书：

《Design Patterns: Elements of Reusable Object-Oriented Software》

《Clean Architecture: A Craftsman's Guide to Software Structure and Design》

《How to Read a Book: The Classic Guide to Intelligent Reading》

啊哈，这本书对程序员也很有用。

## 译者的话

这篇文章是为苦于不知从何开始阅读陌生项目代码的人（包括我）准备的。截止撰文时间为止，我认为这篇文章是在同话题中较为务实的一篇，不仅包含理念，还包含不少具体可实践的建议，因此我想把它分享给更多人。而且我想如果它有中文版本，愿意读下去的人会多一些，我也方便向周围的人分享，于是我向 Nic 申请了翻译权。我把标题改了改，这样搜索“源码”“学习”、“阅读源码”都能搜到。

这篇文章对我而言，最大作用是帮我突破了“不需要有意地专门阅读代码”的心理障碍。我一直以为既然“写代码的时间中十之六七都是花在读既有代码上”（语出《Clean Code》），就没必要再花时间专门读代码，我大错特错。

个人经验：驱动我“专门读代码”的最大动机是好奇心，和小孩拆小物件为了看内部构造差不多。学习 OO 设计模式，我建议阅读《Head First 设计模式》，这本书超有趣，比四人帮那本删减了一些不常用的模式，但是你能轻松读下去。

阅读了解项目代码是参与（开源）项目的第一步，希望这篇文章能帮助你参与到心仪的项目中去。这也可以帮助你在工作中了解同组同事的工作，而“了解同组同事的工作对工作有诸多潜在益处”。来吧，花点时间挑一个看上眼的项目（或者就读你手头的项目别人写的部分），找到你最感兴趣的功能，读一读它是怎么实现的。

作者： : Nick Mose

链接： : <https://coderscat.com/learn-from-source-code/>

版权： : all rights reserved

updated 2023-08-08

? Feedback



相关文章：

SRP 提醒你不要再盲目遵守 DRY 设计原则

白话解释单元测试中的 Mock 概念

编程所需要的东西

programming

blog

#programming #blog

programming

< 编程所需要的东西

单元测试的不同方式 >

2 Comments - powered by utteranc.es

boholder commented on 2021年5月28日

那个“了解同组同事的工作对工作有诸多潜在益处” 的出处博客找到了：  
<https://laike9m.com/blog/jin-ji-nian-wo-zai-zhi-chang-cai-guo-de-keng,143/#-4>  
  
ps:原来我错误配置了utteranc.es，所以实际上没能开启评论系统.....抱歉啦。

boholder commented on 2023年5月19日

<https://www.joelonsoftware.com/2000/05/26/reading-code-is-like-reading-the-talmud/>

WritePreview

Sign in to comment

 Styling with Markdown is supported

© 2019–2023 ∴ BoHolder  
Powered by Hugo | Theme is MemE  
CC BY-NC-SA 4.0