


手把手教你阅读开源代码（长文）



编程实战营
萌萌的会做菜的程序员

关注他

49 人赞同了该文章

一、开源项目初体验

1、走读一遍官网介绍文档（走读）

有多少人愿意花时间好好看官网文档的？估计很少，大家更倾向于了解个一知半解后就开始撸代码，强撸！！

撸代码过程中会遇到很多稀奇古怪的问题，再回过头用搜索引擎来解决这些问题，发现是自己的使用方式有问题，其实某个api的使用方法在demo中或者文档中，已明明白白的写的很清楚。你说，是不是大冤种？

重点看文档中什么内容：

1、开源项目背景

- 采用了什么技术？
- 提供了什么功能？
- 解决了什么问题（或者核心痛点）？
- 和同类产品相比有哪些优势？

2、适用场景

- 优点是什么？
- 缺点是什么？
- 适用于什么场景？
- 不适用什么场景？
- 明确了适用场景，才能在工作中恰当的使用出来。



3、哪些公司在生产中使用

是否有中型以上公司在生产环境中去实践了此开源项目，可以从侧面证明这个开源项目很有潜力和竞争力，很值得你花费大量的时间和精力去研究。

比如使用brpc的公司如下：

Used by













相信很多小伙伴看工作内容相关的开源项目能阅读下去，但是其他领域的开源项目看不懂，并不是因为你能力不够，而是缺乏领域知识。

在已有的知识体系之下去学习新的开源项目，需要对新概念新技术有个大致的了解，才能更好的理解项目的整体实现思路，阅读起来源代码才事半功倍。

比如阅读Ivs源代码，起码要明白以下概念

RS: Real Server【后端真实服务器】

Client IP: CIP【远程访问的客户端ip】

Director Virtual IP:VIP【用于对外公布的虚拟ip】

NAT、FULLNAT、DR、TUN模式。

千万不要一上来就看代码，核心概念不清楚，核心原理不懂，核心算法没吃透，看代码很难看懂。

5、尽量看官网英文的资料（时效和准确性更好）

尽量看官网英文的资料，从时效性和内容准确性上都要比中文技术博客的内容要好的多。

网上的博客，尤其是csdn的博客，大家都是抄来抄去，难免有错误和纰漏。

重点关注下Get Started和Example之类的文档，能够帮助更快的入门。

2、了解代码目录结构

在下载源代码之后，先看下代码目录组织结构，可先从示例代码或测试代码入手。

如base代表基础库，net代表网络库，

demo/example：示例代码，样本程序

tests：测试代码

docs：文档目录（类图，流程图，活动图，业务知识相关资料等）

入手之初，建议从例子代码开始看（demo，example，test），从而熟悉开源项目的流程或者使用方法。

对于不理解的技术点，要认真的查询文档来验证其准确性，技术是一门容不得马虎的手艺。

3、相关视频或ppt

文档和代码毕竟是静态资源，能传达的信息不够生动形象，如果能有相关的技术视频搭配来一起看，效果就好的多了。

4、看cmake或makefile构建脚本

github上流行的开源项目，比如C/C++/Go语言相关的项目，多数还是以cmake或makefile体系来构建的。

代码在构建时，有以下讲究：

1、查看依赖库有哪些？

2、开启或关闭指定功能的编译宏，传递一些宏到代码中

4、熟悉开源代码的构建脚本，才能方便改代码，编译，调试的后续流程

二、安装部署，运行（很重要！）

好的开源项目，其文档都比较完备，安装部署文档一般会在README中或者doc目录中，一般会提供几种部署方式

2、1 yum或apt源安装

2、2 rpm或deb包安装

2、3 源代码编译安装部署

2、4 docker镜像部署

将程序运行起来！！

将程序运行起来！！

将程序运行起来！！

重要的事说三遍，程序运行起来有，我们就有了直观感受。

就我的经验而言，一个项目代码，是否能顺利的搭建调试环境，效率大不一样。

跑起来之后，还要尽量的精简自己的环境，减少调试过程中的干扰信息。比如，Nginx使用多进程的方式处理请求，为了调试跟踪Nginx的行为，我经常把worker数量设置为1个，这样调试的时候就知道待跟踪的是哪个进程了。

也就是，为了方便调试，我们会修改开源项目的代码。

玩起来，嗨起来，搭建成功后必须装逼起来，哈哈

首先是便于从视觉感官上体会项目的功能，

其次方便下断点调试，修改代码，再调试，查看是否符合修改的预期；

最后，方便截图，截牛逼的图去吹牛逼，2333~

三、运行简单Example例子

example目录：例子目录，先通过例子上手学会如何使用此开源项目

3、1 直观感受程序功能

安装部署好项目，最好能有web界面等交互式界面或者有数据结果输出，从宏观上熟悉了开源项目的大体功能。

然后从微观上入手，转到局部的example例子上，扣细节。

阅读简单的入门例子，能够抛开复杂的处理细节和高级特性，便于尽快熟悉项目的主干和核心模块。

3、2 由浅入深剖析例子代码

简单例子都遇到问题，可用邮件列表、社区、Issue来解决遇到的问题。

3、3 记录心中疑惑，带着问题阅读

熟悉简单例子的过程中，对于有疑问的地方先记录下问题，后面带着问题去阅读源代码解决心中疑惑。

熟悉简单例子后，也可阅读些高级点的例子，或手动修改代码看看例子是否能正常运行？

如果公司使用的开源项目实在没有example例子，可考虑自己根据公司需求编写一个demo例子

3、4 编写符合公司场景的demo代码

看懂demo或example代码，仅能说明熟悉了api的使用流程和功能边界，但是想要真正吸收开源代码的知识变成自己的知识，需要根据公司场景动手敲代码写个demo，并对此demo做功能性验证，期间遇到各种问题并解决问题，会增长对于开源项目的技术理解。

四、认真细致的看文档，看单元测试用例

4、1 看代码之前认真看文档

一个好的开源项目是有完善的文档的，比如百度开源的RPC框架brpc

docs目录：

库的开发入门指南，代码的风格文档，程序的命令操作手册，程序中采用的第三方库

类图，流程图，活动图，架构图，设计文档

业务相关基础知识介绍文档

项目技术细节详细介绍

阅读这些文档可以了解该项目的大体设计和结构，读源码的时候不会无从下手。

务必要耐心好好的看，多看多实践多思考，少BB

4、2 认真看测试用例

如果测试用例写的很仔细，那么很值得好好去研究一下。

原因在于：测试用例往往是针对某个单一的场景，独自构造出一些数据来对程序的流程进行验证。

test目录：测试目录，学习下单元测试是如何写的

通过看单元测试用例，可了解到某个类或某个组件是如何使用的

五、尽情的玩耍

各种玩：

执行example中的例子查看效果，也可以修改example中的例子来验证自己的想法。

修改test目录中的代码，看看测试代码是否能通过

比如nDPI项目，开始玩时，可以只加载一个插件，玩熟练后可以加载N个插件；

先学会如何使用，然后再去从源代码角度 理解实现原理。

六、阅读源代码

6、1 明确阅读代码的目的（推荐带着问题读）

在开始阅读代码之前，首先要明确自己的阅读目的：

是需要了解其中一个模块的实现？比如基础模块还是业务模块？基础模块中有内存池、线程池、网络数据收发epoll模型等

还是需要了解这个框架的大体结构，

还是需要具体熟悉其中的一个算法的实现，等等。

心里很清楚自己想要什么，才能更有动力去朝着目标努力奋斗。

6、2 区分主线和支线剧情

阅读源代码时，分清主线和支线

先看主线：

想了解一个业务逻辑的实现流程，中间调用了第三方库函数、utils函数、定制的数据结构和算法等，其实不需要了解其内部实现；

只需要了解其对外接口，了解这些接口的入口、出口参数以及作用，把这部分当成一个“黑盒”即可。

再看支线：

在对主线有充分了解前提下，可以考虑打开“黑盒”研究下其内部实现，比如看看内存池的实现代码，调度器的代码，dpdk中对于海量数据包是如何处理的等。

具体做法：

从main函数进入，使用gdb单步跟踪理清一次完整流程（如程序初始化）的代码调用路径，这可以通过debug来观察运行时的变量和行为。

6、3 挑选感兴趣的“分支”来阅读

这里的分支指的是功能或版本：

从功能上说：

比如你对网络通讯感兴趣，就阅读网络层的代码，深入到实现细节：

如它用了什么库，

采用了什么设计模式，

为什么这样做等。

如果可以，debug细节代码。

从源代码版本上说：

不一定选择最新版本的开源代码进行阅读，如果感觉有点吃力，可选用同一个开源项目的老版本（如1.0版本），

6、4 理清核心数据结构

程序设计 = 数据结构 + 算法

因为结构定义了一个程序的架构，结构定下来了才有具体的实现。好比盖房子，数据结构就是房子的框架结构，如果一间房子很大，而你并不清楚这个房子的结构，会在这里面迷路。

而对于算法，如果属于暂时不需要深究的细节部分，可以参考前面“区分主线和支线剧情”部分，先了解其入口、出口参数以及作用即可。

Linus说：“烂程序员关心的是代码。好程序员关心的是数据结构和它们之间的关系。”

因此，在阅读一份代码时，厘清核心的数据结构之间的关系尤其重要。

6、5 添加日志 and 单步调试（情景分析）

看文档或看代码的过程中，对一些技术点，是需要验证的。

所谓的“情景分析”，就是自己构造一些情景，然后通过加断点、调试语句等分析在这些场景下的行为。

我惯用的做法，是在某个重要的入口函数上面加上断点，然后构造触发场景的调试代码，当代码在断点处停下，通过查看堆栈、变量值等等来观察代码的行为。

情景分析的好处在于：不会在一个项目中大海捞针似的查找，而是能够把问题缩小到一个范围内展开来理解。

从而把个人的时间和精力聚焦到一个小的范围内不断的研究，一直研究到自己满意为止。

修改源码加入日志和打印可以帮助你更好的理解源码。

gdb单步调试，一步步跟踪，是了解源代码最好的途径，没有之一。

6、6 画图梳理源代码逻辑

适当画图来帮助你理解源码，在理清主干后，可以将整个流程画成一张流程图或者标准的UML图，帮助记忆和下一步的阅读。

画图虽然很浪费时间，但是对帮助理解架构和流程很有帮助。

6、7 解决遇到的问题

相信经过前面三个阶段，你已经是各种玩，各种瞎折腾，心里还是有很多疑问吧？

这个功能真牛逼，是如何实现的呢？

我的配置文件为什么没有生效呢？

程序为什么被我搞挂了呢？

我添加的打印为什么没有被执行？

打印的值为什么不符合我的预期？

好！很好！带着这些你心里的问题，去源代码中找答案吧

1.先总体后局部，把握好主体逻辑代码，然后再逐层深入下去

一上来就陷入太多的实现细节是大忌！！

是个什么样的处理流程，定位到具体代码，一行行的分析它，这个过程会持续一段时间。

2.带着问题去看代码

一个系统实现了什么功能，为什么要实现这些功能，是基于什么业务场景？

3.思考自己如何实现类似系统（功能）

如果我要来实现一套类似的系统，我会如何来考虑问题，如何来实现这套系统？

然后再看看别人是如何实现的，找到两者之间的差距，并不断缩小之间的差距。

七、写笔记记录想法，促进思考和总结

1.记录解开问题谜团的过程

2.记录下开源代码中好的设计思想，好的代码技巧，以及任何你觉得好的东西

3.画整个程序的流程图，有利于理解程序的整个流程，而不被代码的细节所干扰。

4.坚持记录源代码学习笔记，记笔记能够有助于更深入的思考，以前很多问题只是浅层思考，不够深入

八、抄写项目源代码

以groupcache来举例，它包括lru，singleflight，protobuf，一致性hash这四个技术

步骤一：抄写前，可以大致了解下这些技术，比如lru是什么？lru解决什么问题？简单了解下lru的实现原理（后续深入具体实现）

步骤二：抄写过程中，会发现和之前了解的实现原理有不同的地方，比如有优化，或者有疑惑点，记录下来，继续抄写。

步骤三：抄写后，针对有疑惑的点，找资料，找人交流，必须搞清楚前面记录的不动的地方。

九、仿写项目

如果感觉自己对于项目理解的还不到位，我推荐一个笨方法，抄项目代码，抄着抄着你就懂了。

如果想进一步深刻的学习到源代码的精髓，可以仿写一个相近的程序进行操练。

理解了这个程序并不表明掌握了这个程序，只有在操练一个相近的程序时，才知道你到底理解了多少，掌握了多少

十、教给他人（输出知识）

检验是否掌握一门技术，就是将知识输出，看他人能否听懂

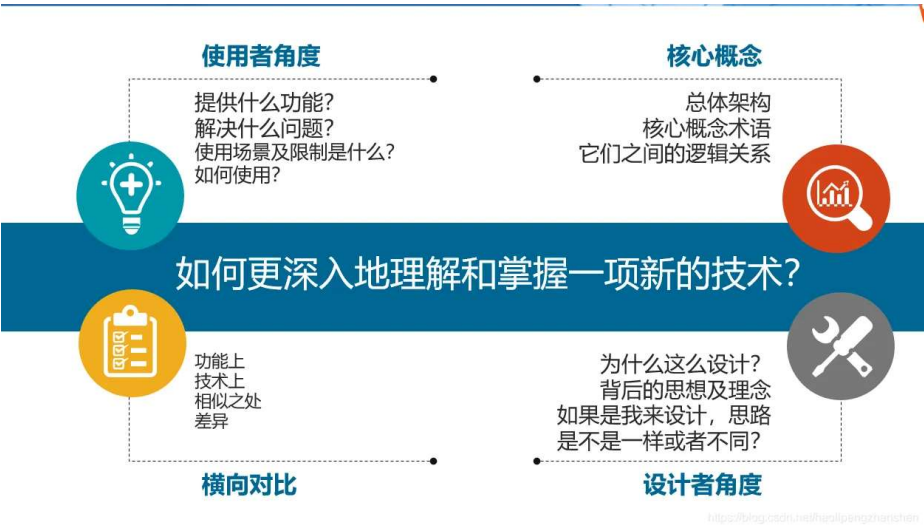
当自己对项目的方方面面都很熟悉后，可以在B站上开直播，将自己所学的知识传授给他人。

输出形式：

- 1、csdn博客、知乎、头条、微信公众号等平台发文章
- 2、组内技术讨论或者微信群讨论
- 3、公司内技术分享会（正式）

/////////////////////////////////2021.02.09更新////////////////////////////////

阅读开源代码，一上来就陷入技术细节是大忌！！



上图是总结的学习方法论，

核心概念：

总体架构

核心概念和术语

逻辑关系如何

从使用者的角度来看：

- 1、提供什么功能？
- 2、解决什么问题？
- 3、使用场景和限制是什么
- 4、如何来使用

横向对比：

功能上相似之处和差异

技术实现原理上相似之处和差异

从设计者角度：

- 1、为什么这么设计？
- 2、背后的思想和理论
- 3、如果我来设计，会从哪些方面来考虑？

如何阅读一份源代码？（2020年版） - codedump的网络日志

编辑于 2022-11-14 22:10 · IP 属地四川

源码阅读 开源项目 高效学习

赞同 49 5 条评论 分享 喜欢 收藏 申请转载 ...

发布一条带图评论吧

- 5 条评论 默认 最新
- 

漫步者

技术专家用心的文章，👍👍👍，我也正在读开源代码，这些方法很有帮助。

04-28

回复 喜欢
- 

编程实战营 作者

能帮忙到大家我很开心

04-28

回复 喜欢
- 

正心寻真

写的真好!

02-03

回复 喜欢
- 

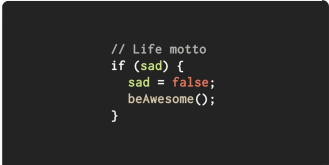
编程实战营 作者

谢谢，一点小心得，能帮忙大家就好

02-03

回复 喜欢

推荐阅读



开源代码 “All in One”：6 份最新「Paper + Code」等你...

Paper... 发表于Paper...



【实用】面对枯燥的源码，如何才能看得下去？

慕课网 发表于猿论



【实用】面对枯燥的源码，如何才能看得下去？

慕课网



程序员如何阅读源码？（端必看）

慕课网