黑客与画家读书笔记

内容简介

本书是硅谷创业之父，YCombinator 的创始人 Paul Graham 的文集。之所以叫这个名字，是因为作者认为黑客（非负面意义）与画家有着极大的相似之处，他们都是在创造作品，而不是为了完成某个任务。书中主要描述了黑客（即优秀程序员）的成长过程，黑客的喜好和心理，黑客对世界的贡献以及编程语言和黑客的工作方法等内容。本书对于在计算机时代下，理解计算机以及计算机背后的人有着重要的启发。

精编书摘

1.书呆子为什么不受欢迎

书呆子毫无疑问想让自己受欢迎，但是他们更愿意让自己聪明。而时间资源是有限的，他们的注意力都在读书和观察世界上面。一个人只有全身心投入，才可能在这个领域出类拔萃。所以书呆子们没有时间和精力，更没有那种意识去努力使自己受欢迎。
孩子们欺负书呆子的另一个原因是为了让自己感到好受一些。当你踩水的时候，你把水踩下去，你的身体就会被托起来。

2.黑客与画家

黑客与画家的共同之处在于有创意的创作，而且它们大都需要时间的考核和评判。在现行的世界规则里，黑客们很难在大企业自由的做自己的事情。
与其说优秀的软件设计师是工程师，还不如说是建筑师。建筑师决定做什么，工程师想出怎么做。
你把整个程序想清楚的时间点，应该是编写代码的同时，而不是编写代码之前，这与作家、画家和建筑师的做法完全一样。
软件的部分功能就是解释自身。为了写出优秀软件，你必须假定用户对你的软件基本上一无所知。你要明白，用户第一次使用你的软件的时候，不会预先做好功课，他们没有任何准备就开始用了，所以软件的使用方式最好能符合用户的直觉，别指望用户去读使用手册。
程序写出来是给人看的，附带能在机器上运行。

3.不能说的话

有时候书呆子说出了自己觉得正确的观点，而这些不合适的真话会致使惹上麻烦。
触怒他人的言论是那些可能会有人相信的言论。我猜想，最令人暴跳如雷的言论，就是被认为说出了真相的言论。
自由思考比畅所欲言更重要，在心里无所不想，但是不一定要说出来。
如果自己就是潮水的一部分，怎么能看见潮流的方向呢？你只能永远保持质疑。
问自己，什么话是我不能说的？为什么？

4.良好的坏习惯

在大众眼里，“黑客”就是入侵计算机的人。可是，在程序员的眼里，“黑客”指的是优秀程序员。
法律和当前政局一定程度上限制了黑客的自由，这恰恰不利于黑客的发展。
常见的犯罪动机不外乎毒品、金钱、性、仇恨等。满足智力上的好奇心并不在 FBI 的犯罪动机清单之上。说实话，这个概念对他们来说完全陌生。
黑客是不服从管教的，这是他们的天性。

5.另一条路

在自己刚刚写好的代码中，找出 bug 往往会比较快。有时，你只要看到出错提示，就知道问题出在哪里，甚至都不用看源码，因为潜意识中你已经在担心那个地方可能会出错。如果你要解决的 bug 出自于 6 个月前写好的代码（假定你一年发布一个新版本，那么 6 个月就是发现 bug 的平均时间），那么就麻烦了，就要大费周章了。那时，你对代码也已经不熟悉了，就更可能采用危险的方式解决问题，甚至引入更多的 bug。
勇于尝试，但要关注用户的使用体验和反馈，不断调整自己的产品和运行模式。
复合式 bug 有一个子类型：两个 bug 是互相弥补的，好比“负负得正”，软件反而能正常运行。这种 bug 可能才是最难发现的 bug。

6.如何创造财富

从经济学观点看，你可以把创业想象成一个压缩过程，你的所有工作年份被压缩成了短短几年。你不再是低强度地工作四十年，而是以极限强度工作四年。在高技术领域，这种压缩的回报尤其丰厚，工作效率越高，额外报酬就越高。
通过创造有价值的东西而致富，这种方法的优势不仅仅在于它是合法的（许多其他方法如今都是不合法的），还在于它更简单。你只需要做出别人需要的东西就可以了。
如果一家公司能够按照贡献付薪，它将取得巨大成功，小公司更适合。
要更好的创造财富，你做的事情需要两点保证：可测量性，可放大性。硅谷的诀窍，可测量性来自小团队，可放大性来自开发新技术。

政府禁止个人积累财富，本质上就是在命令人民减慢工作速度。强大起来的社会，都允许创造财富（注意不是通过转移获得财富）的人保住自己的财富。
要鼓励大家去创业。制药懂得藏富于民，国家就会变得强大。

7.关注贫富分化

每个人的技能不同，导致收入不同，这才是贫富分化的主要原因，正如逻辑学的“奥卡姆剃刀”原则所说，简单的解释就是最好的解释。
你想一想，一个篮球队会同意用一个运动员交换 100 个普通人吗？如果苹果公司不是由乔布斯掌管，而是由一个 100 人组成的委员会掌管，那么这家公司的下一代产品会是什么样^？人与人之间的差别并不是那么稳定的线性关系。也许 CEO 和运动员的技能和决心只比普通人高出 10 倍（倍数不重要），但是人与人之间就是存在着重大差别。

8.防止垃圾邮件的一种方法

在所有对抗垃圾邮件的方法之中(从软件方法到法律方法)，我认为单独来看，“贝叶斯过滤”是最有效的工具。但是，我也认为，我们使用的不同方法越多，综合效果就越好。

9.设计者的品味

- 好设计是简单的设计。
- 好设计是永不过时的设计。
- 如果解决方法是丑陋的，那就肯定还有更好的解决方法，只是还没有发现而已。
- 好设计是解决主要问题的设计。
- 好设计是启发性的设计。
- 好设计通常是有点趣味性的设计。
- 好设计是艰苦的设计。如果观察那些做出伟大作品的人，你会发现他们的共同点就是工作得非常艰苦。如果你工作得不艰苦，你可能正在浪费时间。
- 好设计是看似容易的设计。
- 好设计是对称的设计。
- 好设计是模仿大自然的设计。
- 好设计是一种再设计。
- 好设计是能够复制的设计。
- 好设计常常是奇特的设计。
- 好设计是成批出现的。推动人才成批涌现的最大因素就是，让有天赋的人聚在一起，共同解决某个难题。互相激励比天赋更重要，达·芬奇之所以成为达·芬奇，主要原因不仅仅是他的天赋，更重要的是他生活在当时的佛罗伦萨，而不是米兰。
- 好设计常常是大胆的设计。今天的实验性错误就是明天的新理论。如果你想

做出伟大的新成果，那就不能对常识与真理不相吻合之处视而不见，反而应该特别注意才对。

优秀作品的秘块就是：非常严格的品味，再加上实现这种品味的能力。

10.编程语言解析

“你用什么语言并不重要，重要的是你对问题是否有正确的理解。代码以外的东西才是关键。”

冗余的代码会导致更多冗余的代码，不仅软件如此，一件垃圾会产生更多垃圾。

11.一百年后的编程语言

编程语言存在一个进化的脉络，我们需要思考，在整个进化过程中某一种语言的位置到底在哪里，从而启发我们去尽可能选择那些靠近主干的语言，这样对当前的编程最有利。

那些内核最小，最干净的编程语言才会存在于进化的主干上。

12.拒绝平庸

别忘了你的对手与你一样，能用任何想用的语言编写互联网软件。

创业公司对竞争对手应该越保密越好。

13.书呆子的复仇

各种编程语言的编程能力是不同的。

在高科技行业，只有失败者采用“业界最佳实践”。

14.梦寐以求的编程语言

编程语言是一种为了便于使用而被设计出来的工具。

一种语言必须是某一个流行的计算机系统的脚本语言，才会变得流行。

编程语言必须要有介绍它的书，必须要有在线文档，同时它需要是一种免费的实现。

黑客欣赏的一个特点就是简洁，简洁性最重要的方面就是必须使语言更抽象。

流行的语言通常很适合开发一次性程序，就是那些为了完成某些很简单的临时性任务而在很短时间内写出来的程序，比如自动生成测试数据，转化数据的程序等。

15.设计与研究

研究必须是“新”的，而设计必须是“好”的。