# 黑客与画家读书笔记

首先，标题处所指的黑客并非恶意入侵计算机系统的人，而是指最优秀的程序员，黑客行为应包含：好玩、高智商、探索精神。应该把黑客和画家当作同一种人来对待，即开发软件与画家作画、雕刻家雕刻、建筑师设计房屋并没有本质不同。

第一章到第四章，作者解释了黑客如何成长以及他们看待世界的一些观点。第五到第九章解释了黑客如何做出自己的成果以及这些成果的影响。第十到第十五章解释了黑客的工具和工作方法。

书呆子不受欢迎的真正原因是，他们脑子里想着别的事情，他们的注意力都放在了读书或者观察世界上面，而不是穿衣打扮、开晚会上面。同时格雷厄姆批判了校园教育，校园生活真正的问题是空虚，指出校园管者只是关心学生应该呆在自己的位置上，对学生内部形成的各种各样的小集团则放任不理。

黑客与画家的共同之处，在于他们都是创作者，他们的本质都不是在做研究，虽然在创作的过程可能会发现一些新的技术。格雷厄姆并不喜欢将计算机科学这个术语，因为他将不同类型的工作绑定到了一起，如果一个黑客的工作被称为科学，这会让他门感到自己应该做的像搞科学研究一样，所以大学和实验室里面的黑客，就不会去做那些真正想做的事情如设计优美的软件，而是觉得自己应该写一些研究性的论文。真正竞争软件设计的战场是新型领域的市场，因为这里还没有人建立防御工事，只要能够做出大胆的设计，由一个或者一批人同时负责设计和实现产品，在这里就能战胜大公司，所以开发优秀软件的方法之一就是自己创业。格雷厄姆还认为大多数黑客不是通过大学课程学会编程的，而是从实践中学习，这就像画家的学习过程是一样的，但和科学家学习却大相径庭，后者通过实验和解题来学习。

格雷厄姆又给出了团队开发合作的建议，不能过于“合”，不能多人一起开发小块的代码。黑客应该时刻考虑到用户的人性需要，尽可能的“换位思考”，这样才能做出伟大的产品。“程序写出来是给人看的，附带能在机器上运行。”

要有自己的观点和思想，并且主动给但是不要随意的发表观点，避免给自己带来麻烦，要找信得过的人讨论，让想法变得更有意义。优秀作品往往来自于其他人忽视的想法，而最被忽视的想法就是那些被禁止的思想观点。不管是哪一个原因，看来存在一个很清晰的关联关系：智力越高的人，越愿意去思考那些惊世骇俗的思想观点。这不仅仅因为聪明人本身很积极地寻找传统观念的漏洞，还因为传统观念对他们的束缚力很小，很容易摆脱。从他们的衣着上你就可以看出这一点：不受传统观念束缚的人，往往也不会穿流行的衣服。互联网软件的目标客户应该是个人和小企业。这些客户决策比较灵活，又需要低成本的新技术，所以他们更愿意尝试新事物。大公司付出的高价之中，很大一部分是商家为了让大公司买下这个商品而付出的费用。这就是为什么公司内部的局域网软件明明不可取、但是还会继续存在并且不断发展的一个原因。这样的软件更昂贵，但是你对这个难题就是无能为力。所以，最好的安排就是把个人和小企业客户放在第一位。其他的客户该来的时候就会来。开发互联网软件的创业公司会把与创业有关的每一件事做到极致。只用更少的人、更少的钱，就可以把软件写出来，并且开始运作。你必须打破常规、快速行动，循规蹈矩不可能成功。你完全能够在只有三个人的情况下让产品开始运营，你们唯一的办公场所就是一间公寓，里面放着一台连着ISP的服务器。创业公司要成功，你必须有他人难以复制的技术或者模式。创业公司不像能经受打击的黑熊，也不像有盔甲保护的螃蟹，而是像蚊子一样，不带有任何防御，就是为了达到一个目的而活着。蚊子唯一的防御就是，作为一个物种，它们的数量极多，但是作为个体，却极难生存。创业公司如同蚊子，往往只有两种结局，要么赢得一切，要么彻底消失。你通常不知道自己会是哪一个结局，只有等到最后一刻才会明了。保险的做法就是在早期卖掉自己的创业公司，放弃未来发展壮大（但风险也随之增大）的机会，只求数量较少但是更有把握的回报。思考：把公司“当孩子养还是当猪养”。买卖的艺术上，大多数时候，促成买方掏钱的最好办法不是让买家看到有获利的可能，而是让他们感到失去机会的恐惧。对于买家来说，最强的收购动机就是看到竞争对手可能收购你。

高级语言使得程序更具有可移植性。编译器不是高级语言唯一的实现方法，另一种方法是使用解释器，它的作用是实时地将代码解释为相应的机器语言，然后一行行运行。相比之下，编译器则是先将整个程序全部翻译成机器语言，然后再运行。我们一直被教育编程应该面向对象，然而作者却有着自己的看法。有好有坏。　　任何一种编程语言都可以分成两大组成部分：基本运算符的集合（扮演公理的角色）以及除运算符以外的其他部分（原则上，这个部分可以用基本运算符表达出来）。基本运算符是一种语言能否长期存在的最重要因素。慎重选择公理还不够，还必须控制它的规模。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。一种语言的内核设计得越小、越干净，它的生命力就越顽强。浪费可以分成好的浪费和坏的浪费好的浪费，即用更多的钱得到更简单的设计。正确提高效率的做法应该是将语言的语义与语言的实现予以分离。在语义上不需要同时存在列表和字符串，单单列表就够了。而在实现上做好编译器优化，使它在必要时把字符串作为连续字节的形式处理。

设计与研究的区别看来就在于，前者追求“好”（good），后者追求“新”（new）。优秀的设计不一定很“新”，但必须是“好”的；优秀的研究不一定很“好”，但必须是“新”的。只有应用“新”的创意和理论，才会诞生超越前人的最佳设计；只有解决那些值得解决的难题（也就是“好”的难题），才会诞生最佳研究。做“用户需要的设计”，而不是“用户要求的设计”。需求和要求不同。