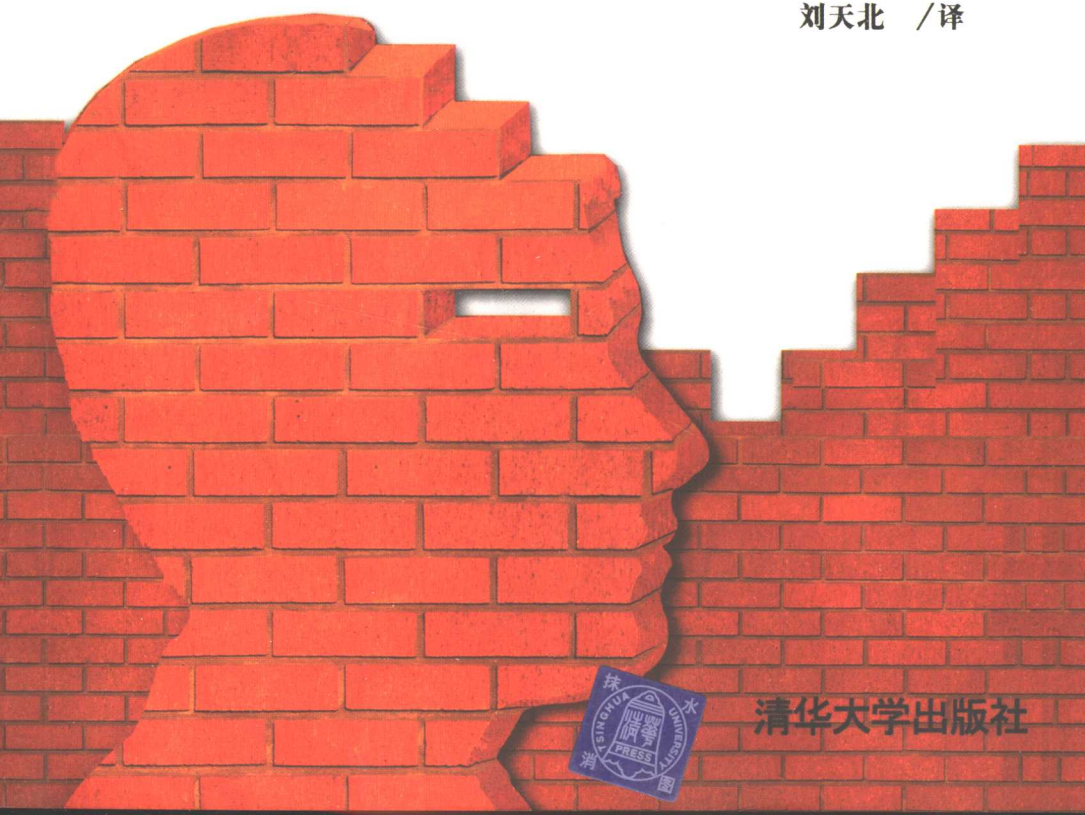
Understanding the Professional Programmer

理解专业程序员

(美)杰拉尔德·温伯格 /著
刘天北 /译



清华大学出版社

Understanding the Professional Programmer

理解专业程序员

(美)杰拉尔德·温伯格 /著
刘天北 /译

清华大学出版社
北京

Understanding the Professional Programmer
By Gerald M. Weinberg
EISBN:0-932633-09-9

图书在版编目(CIP)数据

理解专业程序员 / (美)温伯格(Weinberg, G. M.)著;刘天北译. —北京:清华大学出版社,2006.7
书名原文:Understanding the Professional Programmer
ISBN 7-302-12994-0

Ⅰ.理… Ⅱ.①温… ②刘… Ⅲ.程序设计—文集 Ⅳ.TP311.1-53

中国版本图书馆 CIP 数据核字(2006)第 051059 号

出 版 者:清华大学出版社 **地 址**:北京清华大学学研大厦
http://www.tup.com.cn **邮 编**:100084
社 总 机:010-62770175 **客户服务**:010-62776969
责任编辑:贺 岩
封面设计:李尘工作室
印 刷 者:清华大学印刷厂
装 订 者:三河市李旗庄少明装订厂
发 行 者:新华书店总店北京发行所
开 本:148×210 **印张**:6.125 **插页**:1 **字数**:177 千字
版 次:2006 年 7 月第 1 版 2006 年 7 月第1 次印刷
书 号:ISBN 7-302-12994-0/TP·8247
印 数:1~4000
定 价:25.00 元

代译者序

认识你自己

可以用一句略带美国味的话来总结你捧在手上的书："它让你笑，它让你叫，它让你跳。"

和所有的温伯格作品一样，这本书首先是一本易读的作品——毋宁说，是一本可读性很强的作品。我们这些从十多岁就开始接受理工科教育，多年来一直善于和机器打交道的程序员，常常给外人（乃至我们自己）留下一些神秘，甚至古怪的印象——当你每天穿着牛仔裤和旅游鞋上班，每句话带上一两个三字母缩写词，趴在电脑前一整天看着那些稀奇古怪的符码，外人很难不把你看做一个与众不同的"geek"。在geek的圈子里，温伯格就算得上是咱们大家的老前辈、老祖师了，听他讲起那些几十年前的geek们的笑话，文化认同极度匮乏的中国同行们该有一种加倍的亲切感。

但是，我得提醒你注意，这本书的名字叫《理解专业程序员》。要照温伯格老先生的说法，15年时间恐怕还不够造就一个专业程序员。于是你也就不难想到，在他老先生的眼里，你、我，以及咱们身边几乎所有的程序员或多或少都带着些"不专业"的味道。所以你就得做好准备了，这位老先生讲的笑话一多半是在讽刺"不专业"的程序员，这些带着刺的笑话极有可能句句刺着你的心窝子。比方说吧，要是你刚离开学校没多久（甚或还没离开学校），现在就翻开"想打板球的蟋蟀"和"想打棒球的蟋蟀"这两个小故事看看，我敢打赌你脸上的表情不会太优雅——那种泛着红晕的苦笑，我可是太熟悉了。

很多时候，被人讽刺不一定就是坏事，尤其是这

些来自专业程序员的讽刺。对专业程序员的误解来自两方面。于其外，那些西装革履、衣着光鲜的外人不仅不理解程序员的“专业”，他们甚至常常就是故意制造着误解——用温伯格的话来说，“程序员们成了各种行业媒体寻常而方便的靶子”；于其内，众多的程序员就像那只可敬的蟋蟀一样找不到“专业”的门径，再加上形形色色的FUD(Fear, Uncertainty, Doubt)不停地在耳边萦绕，很多人自己就开始相信……唔，点到为止吧，在网络上你不难找到他们相信的东西。这本《理解专业程序员》首先就是要厘清误解，把一个“专业”的目标、一种职业的认同感还给程序员。

和温伯格所有的作品一样，这本书一直在强调着、渲染着程序员的文化。谈起“文化”，温伯格举得最多的例子就是医生——既然医生可以用一种内生的文化为自己筑起藩篱，以便提升自己的地位和价值，程序员为什么不可以？(有心的读者不妨留意书中和医生有关的故事，看看他们是如何借助来自希腊文的生僻词汇营造文化壁垒的，这算得上是一个典型的传播学案例了。)而营造一种文化，第一件要务无疑是区分“专业人士”和“业余爱好者”(或许还应该加上“不那么专业的从业人员”)。既然包括Fred Brooks在内的众多专家都赞同程序员水平的差异会对生产率造成几个数量级的影响，那么这种专业/非专业的划分就该是对所有人——而不仅仅是程序员自己——有利的。

说实话，对于你在阅读本书时即将体验到的复杂心情，我非常理解——因为我也刚刚体验过。上一分钟，你也许还在为一个与己无关的笑话而捧腹；下一分钟，被嘲笑的对象好像又变成了你自己。我衷心希望本书的读者能珍视这种难得的阅读体验。特尔斐神庙的先知告诉人们：“认识你自己。”作为一个有志于在专业程序员之路上探索跋涉的旅行者，能够看到这条路的方向，能够有人(尽管有些尖锐地)指出自己的不足之处，这该是弥足珍贵的。

透　明

2005年3月于杭州

著者序

1944 年，我 11 岁，我读到了《时代》(*Time*)杂志上一篇谈计算机的文章！在那个年纪，我的脑子是一块干净的石板，等着它的粉笔，所以那篇文章给我留下了很深的印象。我记得自己坐在起居室那把高背椅上，《时代》放在膝头，而整个“时代”都在我手中，当时我就决定，我要成为一个“计算机人”。1944 年以来，计算机发生了巨大的变革。由于计算机的应用，我们的生活也发生了巨大的变革——我自己的生活就更其如此。但至少一件事没有变化。显然，从那以后几乎过去了一生的时光，我也从计算机文章的读者变成了作者。但是即使是今天，仍然没人理解“计算机人”是种什么人，他们究竟做什么，他们应该做什么。

不能说“没人知道”。这里的问题是：**人人**都知道——但每人知道的都不一样。我在计算机界度过近乎一生之后，尘埃还未落定。计算机人士仍有**选择**他们做什么的自由——而依据这个自由，他们也就能选择究竟让计算机做什么。也许，再过 50 或 100 年，计算机人士将被塞入界线清晰的鸽子窝中，度过一生。可是今天，我们的命运似乎还在自己手上。

迄今为止，计算机产业一直建筑在思想的自由交流上。从前，你每月都能参加一次会议，在会上能和——比如说洛杉矶市内的——每

一家大公司的某个代表谈话。现在呢，怀旧派还会去 NCC[①] 大会，想在来自世界各地的 75 000 名同行之中找回过去的那种氛围。可那其实已经大不相同了。

确实，永远也找不回旧日子了。我们从前凭借直觉（有点儿类似于动物本能）取得的经验，现在只能通过更加明确的结构体系来获得。比如说，在那些大型会议上，已经没有留给“小”主意的空间了。而在那些小型会议上，更没有留给小人物的空间——大家都是来听大人物咳珠唾玉的。

从前呢，我们的工作进程特别依赖于会议，因为很少有书面材料。当 1956 年我开始为 IBM 工作的时候，我用了不到一周的时间，就读完了旧金山分部的每一篇技术文献（包括公共库中的所有文件）！这也大概就是 IBM 在 1956 年拥有的所有技术文献了。

今天，我们简直淹没在技术文献当中，很少再有时间进行面对面的交流，至于“非正式读物”，那似乎就完全没时间看了。就这样，各处的计算机人士失去了彼此的联络，失去了思想交流的机会——尤其是那些算不上特别“技术性”的思想，因为它们的模糊特质，也很少能被收入各种使用手册或教科书。而一旦不再交流这些思想，我们也就失去了对于自身现实中很重要的一部分的接触，我们的工作也因此受到了损害。

我定期为几种刊物撰写专栏文章（专栏标题分别是“Stateside”，“Phase 2”以及“From Eagle, Nebraska”[②]），会收到大量读者来信，当阅读这些来信时，我就能特别强烈地感受到上面说的那种“缺乏对现实的接触”。这些读者都是计算机人士，分别在杜布克、都柏林和丹尼丁[③]工作。虽然他们彼此有那么多相近之处，但却不能有机会面对面交谈。如果他们真能遇上，他们肯定能马上认出对方就是“计算机人

---

① NCC 大会，全称是“全国计算机业大会”（National Computer Conference），美国计算机业曾经举办的系列年会，现在大概已经停办了——作者为中译本做的注解。

② 这些专栏刊载在全球多种著名专业杂志上，包括英国、澳大利亚、新西兰的 Computerworld 杂志，以及日本的 BIT 杂志——作者为中译本做的注解。

③ 杜布克、都柏林和丹尼丁（Dubuque, Dublin, Dunedin）是作者按照字母顺序随机举出的 3 个城市名，足见读者遍及天下。

士”——就像他们从我的文章中认出自身那样。但是我说的“计算机人士”是什么意思呢？是那种人，他读着这些文章，会不时说“是呀，我就是这样做的”，或者是“我也这么想”，或者是“但愿我的经理能懂这个”，或者是“我可以用上这个技巧”，甚或是“这个叫温伯格的家伙已经弄不明白我工作的这种环境了”。

所以，这一组文章，是写给那些想要更好地了解计算机人士的读者的——所谓计算机人士，也就包括程序员、分析师、经理、设计师、培训师、测试员、维护人员、操作员、系统管理员、架构师、企业领导，或者其他各种头衔下的那些人。大部分计算机人士的时间都很宝贵，为了不浪费这宝贵的阅读时间，我也力图把文章写得有趣。只要你有 5 分钟空闲时间（比如你在等着 TSO[①] 响应的那一会儿工夫），你就可以打开本书的任何一页，读完其中的一点想法。这可能不是世界上最伟大的想法，但还是会有一定益处。即使没有益处，大概也能为你带来一点儿消遣——这本身也算是一种益处吧。

① TSO：即 Time-Sharing Option，时分复用。

前言

“专业的”(professional)这个词,既包括几层明晰清楚的“显义”,也带有好些含混模糊的“隐义”。我想先考察一下本书标题中的“专业的”是取自哪些意思,这样读者也就能更好地理解,本书究竟涉及了哪些范围。

我的《美国传统词典》说,“专业的”的意思是:“从属于、关联于、介身于或适合于一项职业”,这又把该词的意思抛给了“职业”一词的定义。这样,也会引发下列争论:编程是一种“职业”吗?它应该是一种职业吗?它又怎样才能成为一种职业?可在本书中,我并不打算让读者投身于以上争论,虽然可以肯定,书中的多篇随笔将向上述“职业”主题抛去少许光亮,或是些微暗影。

词典里的第二和第三个定义强调了工作的报酬。但是我所说的“专业的程序员”,可能为编程工作收取报酬,但并不一定收取。而且,还有很多人,他们确实为编程工作收取报酬,但我却**不**会把他们当成“专业的”,我的理由呢,读者看了以下随笔后自然就能明了。

跟我的想法最接近的定义是这个:**在特定的活动领域里,具备了不起的技艺或经验**。

这本书,讲的就是在计算机编程领域具备了不起的技艺或经验的那些人。

归属于“计算机编程”这个题目下的内容非常丰富,与我早前写的《程序开发心理学》(*The Psychology of Computer Programming*)那本书不同,本书并未试图囊括所有那些内容。这里的重点,是那些技艺高超、经验丰富的工作者——在程序开发这样一个波谲云诡、变动无常的环境中,我们怎样才能成为这样一个人呢?我们又怎样才能保持这样的水准呢?

我的老朋友、老对头 Phil Kraft 曾经为了《程序开发心理学》的书名狠批过我一回。按我的理解，他想说的意思是：从字面上讲，不存在一项"活动"的"心理学"，只存在个人的心理学，或是人群的心理学。《理解专业程序员》(Understanding the Professional Programmer)这个书名，想必会让 Phil 好过一些(不过我怀疑，本书的不少内容又会让他火冒三丈)。在书名里，我用的是"程序员"的单数形式，这是因为本书中我重点考察的是作为个体的程序员。

《理解专业程序员》的首要目标，是为专业程序员提供一种自我考察、自我测试的方法。一位非常机智的咨询顾问 Eugene Kennedy[①] 说过：

> 有些人严厉、粗疏的自我考察，往往包含某种混杂的动机，半是出于担惊受怕，半是由于自己具有某些欲望，又不愿被人发现。正如大多数传教士和募捐者知道的那样，让人们感到自己有罪，这实在并非难事……可是，由于自我考察往往不必要地让人自认有罪，所以在很多人眼里，自我考察也担上了恶名；他们或逃避，或拖延自我考察，因此也就从未意识到它的价值。

我的意图，不是在专业程序员中间引发罪责感。恰恰相反，我认为，程序员们成了各种行业媒体寻常而方便的靶子，这些行业媒体主要关心的是兜售硬件；在那些缺乏认真思维习惯的记者们看来，程序员，对于提高硬件销量和广告收入来说，仅仅是一块绊脚石。

Kennedy 接下来的这段话正好可以描述我的观点：

> 有另一种看待自我的方式；专业人士——无论是医生还是运动员——更多地就是这么做的：为了解放自我，持久地提升自己的表现，个人需要接受某种特定的规训[②]。换句话说，健康的专业人士自我考察，不是为了惩罚自身，而是为了获得自我提升。

如果你想成为上述意义上的"专业程序员"，这本《理解专业程序

---

① Eugene Kennedy，美国著名的非虚构类畅销书作家。所著《怎样成为咨询顾问：非专业咨询顾问指南》是该领域的经典读物。

② 规训，原文 discipline，一般译为学科、训练、纪律、惩戒等。中文中很少有现成语汇能涵盖上述多个义项。该词也是法国后结构主义哲学的一个核心概念，大陆学术译著中通译为"规训"，现从此译法。

员》就恰恰是为你而作。

致谢

我想向以下同行致谢，我在书中引用了他们和我的通信：David Coan, Jo Edkins, Bob Finkenaur, David Flint, D. A. Martin 和 Barbara Walker。我也想感谢其他的很多通信者，他们激发了我的灵感，不过很难在这里一一列出姓名了。

在准备本书时，我搜集了想归入这里的多篇随笔，有几位同事通读了它们，并帮我划分了内容的级别，在此我对他们表达特别的感激。他们的评语和建议又一次告诉我，评论对于创造性工作来说有多么重要。我的这个"专家组"包括：Jim Fleming, Mason C. Gibson, Tim Gill, Bill Hetzel, Roger House, Bob Marcus 和 Paul Mellick。没有他们，本书就没法完成。

当然，要是没有我家里的那个"内部专家组"（包括 Dani[①] 和 Judy），本书也不可能完成。她们从不惮于把想入非非的我带回现实。

① Dani 是本书作者 Gerald M. Weinberg 的太太，一位人类学家。

目录

第1章 对专业人士来说，有哪些重要问题

成为一个程序员要花多长时间

对于有些事情，似乎每个人都是专家。教学就是一个好例子。任何人，只要智商超过80，又懂得一点儿什么东西，似乎都可以当老师。至少美国的教育体系就是建立在上面这个理论上的。在美国，但凡你敢对一个教授说，他的课堂教学还有可改进的地方，那他就会感到羞辱、恼怒，还很可能采取法律行动。

还是在美国，每个人都是当招待的专家。在欧洲，一个侍者可能要经过10年，甚至20年的训练，才能获准在一个一流饭馆服务。在美国，只要按照广告应征，在小臂上搭一条毛巾，那就是侍者了。

编程是另一个不缺乏专家的领域。按照标准看法，6个星期的“培训”就足以把一个人提升到“专家”层次，该人不必再学习任何新的知识，即具有设计在线生命救援系统的资格。如果你看到一条广告招收“有经验的”程序员，那意思往往就是一年或者两年经验。实际上，如果谁有15年的编程经验，人们倒会觉得这人简直是个智障。如果他真有一点点智力的话，那总应该在14年前就学会了全部编程知识。在此之后，他就早该做腻了这一行，去换个管理呀，销售呀之类的职位了。

先别忙着嘲笑持这种观点的人，首先我们还是应该承认，15年的经验，就其自身而言，在编程方面不一定就能教会你任何东西。我认识一些有“15年经验”的美国侍者，甚至不知道餐前如何在餐桌上放盘

子。我也知道一些有“15 年经验”的美国大学教授，甚至教不会小狗摇尾巴。同样，我也认识一些有“15 年经验”的美国程序员，他们仍然会在一个多程序访问的系统中，在更新直接存取主文件(master file)之前，就给事务文件(transaction file)排序。

如果说这个例子还太难懂，那我就来列举几个前两天读“有经验的”程序员写的代码时发现的问题：

1. 在做整数除法时，有些人不懂“余数”是什么东西！

2. 为了把一个取值在 0～5 的变量转化成取值在 1～6 的变量(用于 FORTRAN 语言的下标)，有人用了 5 个 IF 语句，再加上 5 个赋值语句！

3. 在写 COBOL 程序的时候，有些人不用“ELSE”子句，原因是“这不一定管用”。

4. 在写 PL/I 程序的时候，有些人从来不用变长字符串，原因是“这个不够高效”。

5. 有些人根本不写子程序，原因是“这太复杂了”。

这个单子能够无限地写下去。这里的要点不是在于，居然有这么多看似专业程序员的人在四处丢人现眼，而在于，没有几个管理者知道，正在和自己打交道的到底是“他们”中的一员，还是“我们”中的一员。

这和美国侍者的处境特别相似。在美国，很少有人曾经享受过专业侍者的服务，所以即使人们真正遇到了一个专业侍者，他们也无从辨别。或者这样说更好，他们根本无法意识到，他们心目中的“标准”侍者其实还处于“亚专业”层次。

同样，除非你自己就是一个胜任的程序员，否则也就很难衡量一个程序员的工作质量。世上有很多可怜的企业，这些企业中从来没能长期留住一个真正胜任的程序员，因此他们也就没有一套标准来衡量程序员的专业性。这些企业的标准就是把庸人当成奇才。而这样的标准也千奇百怪，各地均不相同，甚至同一公司中的不同部门也不相同。

每次我到一家新公司去做咨询顾问的时候，我都提前让经理给我看一些典型代码。经理们往往都不敢相信我真是要看代码，我总得坚

持索要好几次才能得手。只要看一小段代码，我通常就能对该公司的工作环境具有相当准确的了解。有时候我说得特别准，管理层听了都大吃一惊，以为此前我跟员工们私下谈过话。

经理们自己永远也不看代码。代码之于经理，如同脏盘子之于领班侍者。一旦你从那个垃圾堆里提升出来，你就再也不碰那些垃圾了——开玩笑碰一下都不成。

有一回，在大学里的时候，我们学生提议，教授们也应该和学生一起参加硕士生考试，好给学生们做个榜样、立个标准。2/3 以上的教授对此满是惊恐，敬谢不敏。他们自己也经过 20 多年的考试折磨，再也不愿意回到考生的位置上去——这会让他们想起从前卑微的地位。

同样，在我们的行业里经理不愿意编码，这说明写代码这个职业在人类等级体系中的地位略高于盗墓者，低于管理层。对于这样的思考方式来说，编写代码不可能构成一种独立的技艺，不可能是一种天分，也不可能是一种有着自身地位的体面职业——所谓体面，就是说不必和盗墓呀，管理呀之类的在同一个尺度下衡量。只要这种态度在数据处理行业还处于主导地位，那就仍然会有 6 个星期培养出来的专家，也还会有那些经理——他们甚至不愿倾听公司高薪聘请的、有 15 年经验的程序员说话。

当老师、当侍者、当程序员，这 3 件事有什么共同之处吗？为什么人人都觉得自己能够像专业人士一样做这 3 件事？首先，这些工作似乎是容易理解的，因为很多挺普通的人都有过相关的经验。每个人都或多或少曾经教过别人。每个人都做过把盘子放在桌上，或者收拾脏盘子的事。但是不是每个人都曾经在一个活人大脑上做过手术，也不是每个人都曾经在陪审团前为一个案件辩护。

但是编程序又是什么情形呢？当然了，并不是每个人都写过程序，对不对？也许不是每个人都写过，但是似乎每个经理、会计、工程师，或者其他大学毕业的专业人士都写过程序。编程课程在大学里相当风行，在很多职业教育中，这也是必修的课程。比如说，IBM 在 20 年来，在行政人员培训班中就设置了一定的“编程经验”。

我不太清楚现在 IBM 的行政人员培训班的具体课程内容，但是有好多年这门课程中包括了那个著名的“曼哈顿问题”，作为唯一的编

程练习。在美国，数据处理课程的主流入门教科书大多会讲到这个“曼哈顿问题”，如果读者中有人不巧没学过这个，我就按照教科书上的写法，在这里重复一遍：

问题是这样的：据说在1627年，白人们用24块钱买了曼哈顿岛。如果这笔钱被存入一个银行户头，按年利率4.5%计算，今天会有多少钱？

（如果4.5%的年利率偏低的话，那是因为这道题是1956年出的，从那时起就被一代代的作者在不同的教科书中抄来抄去。）

这道题的“解法”，如果抛开一些无关紧要的细节，按照FORTRAN语言编写，那就是这样一个循环：

```
      I = 1627
      PRINC = 24.00
2     PRINC=PRINC * 1.045
      I = I + 1
      IF(I-IYEAR)2,1,1
1     WRITE (3,601) PRINC
```