



MacTalk

人生元编程

◎作者：池建强



池建强

70后程序员，Blogger，微信平台 MacTalk 作者。先后任职洪恩软件和用友集团，从事互联网和企业应用软件研发，目前担任瑞友科技 IT 应用研究院技术负责人。

热爱技术和编码工作，Apple 和 Google 产品重度用户，分享技术，坚持梦想。

博客：<http://macshuo.com>

微博：@池建强

微信平台：MacTalk By 池建强 (sagacity-mac)

图书在版编目 (C I P) 数据

MacTalk • 人生元编程 / 池建强著. -- 北京 : 人民邮电出版社, 2014.1
ISBN 978-7-115-34223-2

I. ①M… II. ①池… III. ①操作系统—程序设计—文集 IV. ①TP316.84-62

中国版本图书馆CIP数据核字(2013)第301996号

◆ 著 池建强
责任编辑 杨海玲
责任印制 程彦红
◆ 人民邮电出版社出版发行 北京市丰台区成寿寺路 11 号
邮编 100164 电子邮件 315@ptpress.com.cn
网址 <http://www.ptpress.com.cn>
北京 印刷
◆ 开本: 720×960 1/16
印张: 19.75
字数: 298 千字 2014 年 2 月第 1 版
印数: 1 - 000 册 2014 年 2 月北京第 1 次印刷

定价: 元

读者服务热线: (010)81055410 印装质量热线: (010)81055316
反盗版热线: (010)81055315
广告经营许可证: 京崇工商广字第 0021 号

内容提要

《MacTalk · 人生元编程》是一本随笔文集，主要内容来自作者的微信公众平台“MacTalk By 池建强”。本书撰写于2013年，书中时间线却不止于此。作者以一个70后程序员的笔触，立于Mac之上，讲述技术与人文的故事，有历史，有明天，有技术，有人生。70多篇文章划分为六大主题：Mac、程序员与编程、科技与人文、人物、工具、职场。篇篇独立成文，可拆可合，随时阅读。

此外，作者还对原来散落在各篇文章中的Mac技巧进行了统一的整理和规划，形成130个Mac Tips，通过阅读这些技巧，读者既可以了解Mac，增长知识，又能够提高工作效率。

本书行文采用了一种技术和人生感悟相结合的风格，起于Mac却不止Mac，文风幽默又能笔底见风雷。王小波说，“每一本书都应该有趣，对于一些书来说，有趣是它存在的理由；对于另一些书，有趣是它应该达到的标准”。本书就是一本达到了有趣标准的技术书，它不仅适合Mac用户阅读，更值得所有技术人员随时翻阅。

写在前面的话

序

在这本书付印的前一天，本书的责任编辑杨海玲老师找到了我说：书要出版了，你要不要写个前言和读者说两句？我说：来得及吗？杨老师郑重地看着我点了点头。好吧，那就说两句，于是就有了这一篇前言。

我小学四年级之前一直在农村居住，那时候没见过汽车，只见过马车和小推车。搬到城市之后才发现，这世上居然有汽车这种交通工具，那时我的理想是当一名 212 吉普车的司机。后来见的车多了，觉得司机不够神气，我希望拥有一辆自己的汽车。这个理想在长大后也实现了。但是我从没有想过自己会出版一本书！今天这本书也要出版了，我很欣慰。

现在回想起来，这本书的诞生经历是传奇和值得称道的。我从来没有出版过一本书，曾经有几次心血来潮想写本技术书，结果规划完目录后自己已经疲惫不堪了，看着层峦叠嶂的目录结构，我仿佛走在一个没有出口的迷宫，想到以后的每个日子都要用业余时间去写这些枯燥的技术文字，我仿佛置身凄风苦雨中而茫然若失，所以，写书计划还没开始，就已经结束了。但是大家都知道，胸中有丘壑的人总能找到吐槽的土壤，是金子总会发光。2012 年 8 月微信推出了公众平台，我终于找到了一个写作的契机。

2012 年 12 月，我开始运营自己的微信公众平台“MacTalk By 池建强”，几乎每天一篇，半年内写了 100 多篇文章。之所以能够坚持下来，我想得益于公众平台的交互性和正向激励，如果每天都有人告诉你，晚上不读完你推送的文章就不睡觉，你也会坚持下去的。慢慢地读者越来越多，很多数字出版界的朋友也加入了这个队伍。突然有一天，他们不再潜伏，而是爬起来告诉我，你的这些文字该收拾收拾出本书了！我原本是不信的，不过，先是多看的朋友这么说，然后是豆瓣，然后是亚马逊，然后我就信了。这种感觉就像是我一个人混迹于敌占区多年，突然冒出了很多自己的同志，拍着我的肩膀说：小同志，你干得不错！我感到无比荣耀，紧紧握住他们的双手，激动地说：可算找到组织了，其实，俺早就想出本书！

于是电子版的《MacTalk · 人生元编程》问世了，书中的内容截止在 2013 年 8 月底。

1

写在前面的话

电子书上线后获得的赞誉超乎我的想象，在多看平台上线当天就跃居畅销榜第一名，从此一路榜首，整整霸占了一个月之久。随后上架的很多新书都对我这本书发起了猛烈的攻击，最终都如潮水般褪去，它们失败了，包括韩寒推出的《一个》。因为这个原因，程序员老赵时不时调侃我为“超越韩寒的男子”，虽然这句话看起来错误百出、逻辑混乱，完全不像出自一个优秀程序员之口，但我听起来还是若有所思的。

截止到 2013 年底，多看上的《MacTalk · 人生元编程》获得了 700 多个评分，平均得分 9.1，另有 230 多条读者评论。有些读者说这是他们的第一本电子书，有些读者说一口气读完，酣畅淋漓。我的感受有两点：

1. 我为电子书事业做出了些许贡献；
2. 又不是武侠小说，读那么快有必要吗？

事情终于到了峰回路转的时候。电子书虽然风光，但仅限于电子阅读的领域，而且，单纯的电子书是没有 ISBN 的，想做个图书认证都不行，还不能签售，你总不能卖本书搭个镌刻版 Kindle 吧。这时候另一群关键人物出现了，他们是传统出版界的好朋友。每个人都告诉我，电子阅读确实是未来的方向，你已经考虑了未来的事儿，那现在的事要不要管？只有电子版，你让广大钟情于纸书的群众怎么看？让想要签名版的读者怎么看？……于是我觉得，是时候出一本升级版的《MacTalk · 人生元编程》了。最终这个重担落到了程序员的好朋友、人民邮电出版社的杨海玲老师肩上。

和人民邮电出版社的合作是轻松愉快的。我继续写作，除了完善之前的文字之外，还要为这本纸质版再增加十万字。出版社的编辑和美编则开始对于原有的文字进行校对和美术设计。一切都很顺利，除了我偶尔出现写不下去或改不下去的情形。杨老师对我是非常宽容的，遇到这种情况，她就会语重心长地告诉我：对你我心里有底，选题咱已经报了，选题号也已经批了，情况就是这么个情况，所以您写也得写，不写也得写，还是写吧。于是我的文章顺利地完成了……

剩下的就是杨老师和编辑们的工作了，他们像打磨璞玉一样一点一点地打磨这本书，从文字、内容、印张和色系选择，到最终的版式设计和封面设计，一个细节都不放过。修改顺利的时候他们不会来找我，偶尔找我的时候也是咨询一下我的个人意见，我只要一言不发或若有所思就可以了，因为杨老师他们总能找到最佳的做法。我唯一的任务就是等待，或想点儿别的。

最终的成书就是您手中的这本书，它的封面设计和正文版式都是独一无二的。书封简约大方，干净得一塌糊涂，体现了 MacTalk 所倡导的纯粹之美。没有腰封，没有邀请业界大牛和土豪在封底写推荐语（年底了，他们都在做新年计划，我就甭给人家添麻烦了）。正文采用图文并茂的方式，页码和标题都精心设计过，如果您获得了良好的阅读体验，那么我们要感谢这本书的美术设计。

书的内容比原来的电子版增加了 Mac 入门和演化史系列、Linux 系列、Vim 插件系列、职场系列、编程与写作等内容，全书共计 70 多篇文章，划分为六大主题：Mac、程序员与编程、科技与人文、人物、工具、职场。篇篇独立成文，可拆可合，您可以随时翻阅。Mac 主题的最后部分是 130 个 Mac 使用技巧，供 Mac 用户参考使用。另外，我根据书中的主题和时间线，重新调整了文章的顺序，更符合读者阅读习惯。

这既不是传统意义上的 Mac 用户手册或技术图书，也不是一本思想文集，而是一个 70 后程序员，以柔软的笔触，立于 Mac 之上，讲述技术与人文的故事，起于 Mac 而不止 Mac。书中写了 Mac，说了苹果，聊了技术，侃了人生元编程，汇聚了我个人的经历和悲喜。您读完了，有收获就是好事，不用想的太多。如果书里在讲 Mac，那就是苹果公司的 Mac；如果在说 Shell，那就是操作系统的 Shell，如果在聊程序员，那就是你我身边的程序员。我尽量做到文字简单、内容有趣，因为简单和有趣也是一种力量，希望您读完这本书以后，也能拥有这种力量。

王小波说，“每一本书都应该有趣，对于一些书来说，有趣是它存在的理由；对于另外一些书，有趣是它应该达到的标准”。我希望这本书达到了有趣的标准，同时还讲了一些有用的东西。如果您说没达到，那我下次继续努力。

在写作的过程中，我得到了很多朋友的帮助，包括技术圈和出版界的朋友。其中冯大辉先生在我运营微信平台和写作的过程中给予了诸多支持和推荐，并为此书作序；微信平台“二爷鉴书”的作者邱岳为此书作序并多次推荐；网友“密码有误”为此书作序并进行了部分文字的校对。杨海玲老师在成书的过程中起到了中流砥柱的作用，没有她的督促和辛劳的工作，此书无法出版，在此一并谢过。

特别感谢我的妻子和女儿，写作占用了我大量的业余时间，虽然我很努力地抽出时间去陪伴家人，但在这方面显然是不达标的。没有她们的包容、照顾和期盼，此书无法出版。

序

写在前面的话

3

最后，我把这本书献给我的父母。我的父母就像千千万万的普通父母一样，永远在远方默默注视着子女的成长。我在他们身边得到了最好的教育和关怀，他们是这个世界上我最敬爱的人！

序

写在前面的话

池建强

写于北京，2014年1月8日，晴

4

技术写作与减熵

序

池建强兄告知我他的书即将出版，作为他的微信公众账号“MacTalk”的忠实读者，必须要支持一下。“MacTalk”和我的“小道消息”差不多同时期开始进行写作，我们也经常交流写作和运营的经验，从内容流行性上来说，我的“小道消息”赢得了更多读者和一点名声，但从内容上看，“MacTalk”则对读者更有价值，所以，一有机会，我总要推荐一下他的账号。

这本书的几个主题中，我最为喜欢科技与人文的这一系列文章，阅读的过程中我也用自己的经历来印证，回顾自己从业过程中那些错误的抉择，给了我不小的启发，建议对自己职业生涯有困惑的朋友能读一下这个主题。如果是5年前看到这本书，我可能会更喜欢程序员与编程这个主题。而作为Mac用户，一系列的Mac技巧则直接提升了我的工作效率，这篇文章就是在MacTalk推荐过的工具上写出来的。

或许有人觉得，写书会带来经济收入和名气，但实际上，我个人认为这可能是一种误解。在我们这个领域，写书的确会带来一点“好处”，但肯定比很多人预期的要小得多，对于作者来说，投入和产出简直不成比例。如若不信，也可以先算一笔经济账。假定一本讲技术人文的书能卖到5000~10000册，按照8%的版税来计算，也就是几万块钱的收入，这还没缴税呢。当然如果是出版电子版图书的话，则需要另作计算，但经济上的收入也不会有多高。而一个作者在这上面投入的时间和精力如果用来做点别的事情的话，比如出去做几次培训，或者是做个技术顾问收点咨询费什么的，轻轻松松也能拿到这个收入。至于“赚”名气，写再多的书怕也不如多参加几次行业会议、多做几次公开演讲赚取名气来得快。

王小波称自己的写作是一个“减熵”的过程，通俗一点说就是“吃力不讨好”。考虑到王小波曾经也是个不错的程序员，套个近乎，算是咱们这个行当的，所以这一点上IT人倒是容易对他的文章有共鸣，至少我就受到了一些影响。我曾在微博上说过，尽管曾被怂恿或是鼓励去写本书，但是总还下不了这个决心。一方面是考虑到经济账。另一方面，心里也有点担心总干这个“减熵”的事情是不是有点自己跟自己过不去？还有一个不好意思说的原因是，我一想到像建强这样把文章重新整理一遍就头大，总是有畏难情绪。但是世界上

技术写作与
减熵

5

总是有人在不停地做这种“减熵”的事情，我想这么做的人可能就是因为这是一种乐趣，一种真正的乐趣，尤其是你的文字能够帮到人、能够启发别人的时候，那种快乐难以言说。

序

技术写作与
减熵

6

作为读者，我能做到的事情并不多，想来想去，唯有尽可能地支持作者的创作，并且向更多人推荐这本书。顺便说一下，如果他这本书销量好的话，不排除我也会“减熵”一次。

冯大辉

丁香园技术负责人，网络 ID: Fenng
微信公众账号“小道消息”作者

但行好事，莫问前程

序

MacTalk 微信推送的文章我都读过，结册成电子书后我又买下读了一遍。看过整册后，我竟然产生了一种敬畏感。

这种敬畏感源自我心中对“写了十几年代码的老程序员”这样头衔的赞叹和崇拜——每一个写了十几年代码的程序员都是一本书。

7

我也差点儿能成为一个“老程序员”，我父亲该算是国内第一代程序员，我出生时他在一所大学教编程，而我母亲当时在另一所大学教统计。诞生在这样的一个家庭，我几乎在育婴室就被贴上了“很可能会是个码农，请注意安全”的标签，接生的护士说：“这小宝贝好可爱，来给姐姐笑一个，笑，笑，你倒是笑啊。”我冷漠而淡淡地回应：“Bad command or file name.”

我曾有过在短时间内带着极大的心理压力，以极大的工作量写代码的经历，我甚至不太敢回忆那时的感受。我知道那种长时间面对一台机器的冰冷和绝望。机器没有感情，它执行你要求的每一句逻辑，你对它没什么脾气，因为起承转合都来自于你的编排，机器只是精确执行。

有人评价程序员说他们“成天跟机器打交道”，别闹了，电脑才不会跟人打交道，在这台坚硬的机器前，一切喜怒哀惧都是程序员心中的自言自语。

于是很多人觉得程序员很可怜，木讷、神经质、不善交际、不修边幅，嘲笑程序员的段子足够装一火车。我听到不少姑娘偷偷地跟自己的闺蜜说“别嫁程序员，没情趣”，或者“嫁个程序员，老实、有钱、死得早”。

在这种诡异的环境中，很多程序员都在“转型”和“突破”，做产品，做运营，做职业吹货，他们想办法甩开“做技术的”或“写代码的”标签。我不止一次听到“程序员是吃青春饭的，你还是得赶紧转型”这样的所谓前辈忠告。

如此背景之下，“一个写了十几年代码的老程序员”头衔的背后，似乎一定有一曲二胡拉就的挽歌，你忍不住掏出纸巾想递给对面已经年逾不惑的他，伸手叫服务员“再上一箱啤酒，冰的”，你悲怆，你泣涕涟涟，你起了范儿，准备听他开讲。

但行好事，
莫问前程

但行好事，
莫问前程

8

结果他一开口，竟然是……

活泼幽默轻松直率荤素搭配清爽可口……

“可是船长，这明明是一场悲剧，你笑什么呢？” “谁说这是一场悲剧了！”

这本书就是这样一个老程序员哼的这样一段小曲儿。端起来有很多技术性文章，放下去有嬉笑怒骂和语重心长。

当然了，书名叫做《MacTalk · 人生元编程》，自然有不少 Mac 技巧。看得出他对苹果情有独钟，书中开篇就是 Mac 的历史和故事。他还有一篇被广泛转载的“趣谈个人建站”，细致到代码级别。他还写人，写书评，说是书评，实则是自省、梳理和复盘，他读《黑客与画家》，势必跟我这个只写过几行代码的产品狗读起来深度不同，我喜欢他那篇书评。

他有代码情结，感觉就是随便一撩动，他就马上坐那儿不起来非要给你写上一段儿，还拉着你让他写，你瞅瞅书中字里行间那些代码，像着了魔似的。

书里我最喜欢的一篇文章是“人生元编程”，他做了一个跨度非常大的类比，用元编程的思想类比人生，那段时间我正在看《自私的基因》，我在这三个概念中间也找到了些似是而非的暗合。

我觉得这篇东西是程序员用程序机制思考人生的一个缩影（当然了书里还有些其他的缩影比如并发什么的），如果你只是一个把堆代码当差事的程序员，我特别推荐你看看这一系列文章，它们或许会帮你打开一扇门，看看这种奇异的关联。

比如你可以在爱情中拒绝 GC，自己照顾（take care）你的每一段回忆（memory），多浪漫啊。

他认同对他文笔“相对轻松温和”的评价，他说自己在网际多年看惯了刀锋和鲜血，所以他不愿意参与或挑起争端。但我不同意他的文笔“温和”，我觉得不温和，而且挺有情绪的。有情绪的文章读起来像作者在跟你聊天儿，也很容易有代入感，是好事儿。

这本书有 70 多篇文章（其中还有 130 个 MacTips），写了这么多内容，书的价格只有 40 多块，要说作者好歹也是一个正儿八经的技术大牛，到底图了个啥？他在开篇里是这样回答这个问题的——“但行好事，莫问前程”，

我太喜欢这句话了！

我想把它送给每一个在深夜烧烤摊前黯然神伤的人们，精于算计的人们，小心翼翼的人们，举轻若重的人们，当然也包括我自己——但行好事，莫问前程。

泰山崩于前，我依然沐浴更衣焚香沏茶，诚心正意，手起键落：

Hello World!

序

但行好事，
莫问前程

邱岳

微信公众账号“二爷鉴书”作者

9

生命中遇见的每一本书，都不是偶然

序

生命中遇见的每一本书，都不是偶然

11

霍夫斯塔特（《哥德尔、埃舍尔、巴赫——集异璧之大成》的作者）在给内格尔的《哥德尔证明》一书作序时，写过这样一段令人印象深刻的话：“……当时是 1959 年秋天，在门罗公园的开普勒书店里，我完全偶然地见到了《哥德尔证明》。可能此刻在中国，有个人——可能就是你——正在书店里面随意浏览这本书，正在翻动它的书页并正在读着眼前这些词句。或许如果你买了这本书，会使你的生活发生革命性的变化，就像这本书对我那样！当然，也可能什么也没有发生。或许正在读这些话的不是你，而是站在书店别处的另外的人。也可能你根本就不在书店里。或许你还正在睡觉呢！但是不管是哪种情况，不管是你或是别的什么人，我的确希望在中国有人能发现这本书，能感受到它是如此之美妙，如此之激动人心……”

按理说，作为一个没有 Mac，不怎么懂编程的人，我和 MacTalk 应该是没有任何交集的。不过现在仍然记得第一次看到 MacTalk 的那一天：一个初夏的中午，阳光很好，看到 Feng 在“小道消息”里推荐了 Mac 君的公众账号，正好吃完饭闲来无事，于是顺手加上了，然后就看到了沉没成本那篇文章，正好之前在曼昆的《经济学原理》里看到过这个名词，印象很深刻，于是查看历史消息，发现作者在精通技术之余，还有着深深的人文情怀，这在码农界是不多见的，于是慢慢变成了 Mac 君的忠实粉丝，每一篇文章都能给人启发，更难能可贵的是，作者对每一条回复到公众账号的消息都会认真阅读并且尽力解答读者的问题，这在很多傲娇的公众账号中更是不多见的。

然后 Mac 君就出了书，在多看买了，又看到豆瓣也上架了，说实话，对于我这种早已告别了文艺青年这个称号的人来说，实在不好意思来这么小清新的豆瓣发评论，不过，这本书确实给了我很多启发，所以，写下这些，也是为了让更多人能够有缘和它相遇。

乔帮主在著名的“遗失的访谈”里曾经说过：每个人都应该学习一门编程语言。我个人认为，这句话是非常有道理的，不仅仅是为了会写代码，能够实现自己的想法，更重要的，是通过这种训练，能够培养程序员的思维方式。正在我为学习哪门语言难以选择时，Mac 君适时推出“如何学习一门编程语言”这篇文章，以一位过来人的身份提点后学。在他的推荐下，开始学习 Python

生命中遇见的每一本书，都不是偶然

12

(和那位码模没有关系，完全是因为今年是蛇年)。有人说，养成一个好习惯需要21天，坚持到现在，每天写几段代码已经成为我的习惯，也许在高级码农眼里，我只是一个非常非常初级的入门者，但是对我自己而言，能够从coding中获得乐趣，将来某天有能力实现自己想完成的事情，就已经很令人满足了。

我一直认为，书和人是有特殊的缘分的，缘分这个词也许有些矫情，更确切的说，遇到的每一本书，冥冥中与你都有特殊的connection。在之前的印象中，码农都是一群带着高度近视镜，脖子僵硬，弯腰驼背的nerd(s)，对人生有着令人不敢苟同的奇怪看法。但是Mac君这本书的出现，很大程度上颠覆了这种印象，作者不仅具有high level的专业技能(在百余篇MacTips里有充分体现)，而且行文生动，文字幽默，在保证知识性之余，也兼顾了全书的可读性，再加上这么便宜的定价，不买本支持一下，怎能对得起这难得的偶遇呢？

正如题目《MacTalk·人生元编程》中“人生元编程”所暗示的，这本书也是作者思考与自省的结晶。所谓“元”(英文“meta”)，就是指能够对自身状态进行描述。例如希尔伯特当年所说的“元数学”，就是指关于整个数学系统的语言，那么元编程，就是能够操作代码的代码，人生元编程，就是具有自省能力，随时检查和控制自身的情绪和行为、思考自己的想法、改变大脑的动机……或者，用一句简单的话来说，就是对当下的状态保持清醒的“觉知”。

继续用霍夫斯塔特的话来结尾吧：“……如果你问我是否取得了最后的成功，答案是‘当然没有！’如果是的话，生活将会变得令人厌烦。如果人的心灵会被化简为几条僵化的规则，或者是相当大的一个僵化规则的集合，那会是一件令人极度悲哀的憾事……我们是幸运的，因为我们的心灵是如此不可预知；正因为如此，生活才充满了情趣。尽管如此，我们仍在进行努力来科学地了解我们自身……”

希望每位有缘读到这本《MacTalk·人生元编程》的读者，都能像作者期待的那样，具有人生元编程的能力，洞察自身的微妙与精深。

密码有误

开篇：为何而写

序

在盛夏的某个周末，我在杭州度过了一段美好的时光。见到了很多老朋友，认识了很多新朋友，参加了阿里嘉年华的“夜聊”节目，领略了夜行西湖的美妙，此行非虚。

MacTalk

开篇：
为何而写

13

杭州的天气和北京截然不同，北京要么是热烈的阳光直射大地，干燥温暖，要么是大风裹挟着暴雨席卷京城，要么以漫天的雾霾夹杂着浓浓的PM2.5气息为你带来“这才是北京的直观感受”。杭州就温和多了，阳光似乎总被一层薄雾笼罩，温度很高，但空气中总夹杂着水汽，出去转一圈浑身湿漉漉回来的一定是北方土鳖，偶尔碰到一个PM2.5，它会告诉你，“我要旅行去北方，那里才是我的家”，想想让人颇为绝望。说是有台风，但台风也只是轻轻掠过杭州上空，带来了些许清凉，带走了一些红尘。

现在我即将离开这座美好的城市，坐在开向北方的列车上，颇为惆怅地写下今天的 MacTalk 主题：为何而写？

在阿里嘉年华的“夜聊”节目中，有一个问题是“你为什么要写东西？”换个文艺一点的说法就是，你就是个写代码的，不去好好做程序员这份有前途的职业，为什么要来写文章呢？谁让你写了？能挣钱吗？有人看吗？你爸爸妈妈知道吗？你的老板允许吗？……

每个嘉宾都谈了自己的看法，我整理一下自己的思路，分了以下这么几点，看能不能把这事说得清楚一点。

程序员崛起。我觉得程序员这个群体是非常幸运的，我们生在一个技术改变世界的时代，而我们可能正在做着能够改变世界的技术，这是何等的荣耀和机遇。想想 Apple 的终端设备、Google 的眼镜和汽车、Amazon 的电子阅读、阿里的电商和金融梦，难以想象 10 年以后这个世界会发生什么样的改变，更不用提那些还在车库里酝酿的各种奇思妙想，而这些改变，程序员都将参与其中。虽然我们现在依然会遭遇到这样的烦恼：“大哥，听说你是搞电脑的？”“我做了 10 年技术了。”“好的，能帮我装个 XP 吗？要番茄版的。”

这时候就可以把兜里准备好的西红柿扔出去了，如果不会扭到自己的老腰，再加个 360° 回旋踢就更加完美了。

不要给自己设限。谁说程序员只能写代码了，谁说程序员都是宅男了？程序员里才华横溢的多了去了，无论是《黑客与画家》还是《乔布斯传》都描述了很多具备文艺气息的技术大师，他们要么作画、要么弹琴、要么写作、要么运动，同时还写得一手好代码。其实万事万物都是相通的，要么熊样要么鸟样，如果你能够把代码写得很好，那么为什么不去把自己的思想和设计通过文字表达出来呢？如果你能够把技术文章写好，慢慢就能写出人文类的文字，慢慢你就会发现自己已经站在科技与人文的十字路口了。最高境界就是，你站在哪，哪里就有一个刻着科技和人文路标的十字路口。在 MacTalk 之前，我写博客，也为一些媒体写文章，最好的时候一个月会写 2 ~ 3 篇，每次约稿都要拖到最后才能完成，但是开始写 MacTalk 之后，到今天为止，近 8 个月的时间我写了 180 多篇有效文章，平均每周 5 篇左右，文字总量在 15 万到 20 万，这个惊人的数量是我从来没想到的，而且花费的时间是我剪切了刷微博、浏览网页、扯淡、看电视的时间，粘到 MacTalk 上的，每个人的时间就那么多，我觉得我并没有损失什么。所以，不要为自己设限！

写作即思考。写作其实就代表思考，你需要言之有物，需要架构、需要梳理，要有开端、有结尾、有结论、有主题，特别神奇的是你构思了一篇文章。写完后发现文章像具备了生命一样生长出了很多奇异的果实，它们就在那些文字中间微微颤动，闪烁着独特的光泽，仿佛被岁月冲刷过的鹅卵石一样，而这一切你可能完全没有想到过，而且不可复制。所以有时候我们去看之前写的文章，会产生两种感觉：要么是觉得写得太烂了，怎么会写得如此臭不可闻；要么是觉得写得太好了，妈妈我再也写不出这么牛的文字了，我觉得这两种感觉，都挺好。优秀的写作者不仅能让事情变得容易理解，而且能够换位思考，沟通顺畅思维敏捷。与这样的程序员交流是赏心悦目的。遇到问题时他会抽丝剥茧，告诉你问题的前因后果，由表及里，并且能够反映问题的各种信息都提供给你，包括他自己尝试解决问题的措施和结果。所以，为自己写作！

附加值。写 MacTalk 的初衷是给大家介绍一些 Mac 相关的技术和技巧，从来没想过商业化和媒体属性，但是做到现在，积累了几万名读者，每天都有用户的反馈和赞助，他们告诉我这些文字为他们带来了欢乐、思考和启发，解决了问题，我告诉他们开心就好。MacTalk 还为我带来了

很多好朋友，有知名大公司的，也有不知名的小公司的，还有在创业的。天地悠悠过客匆匆，能认识这些朋友，真好！所以，为读者写作，附加值会随之而来。

MacTalk，为自己而写，为读者而来。但行好事，莫问前程；河狭水急，人急计生。

序

开篇：
为何而写

目录

| |
|----------------------|
| 写在前面的话 /1 |
| 技术写作与减熵 /5 |
| 但行好事，莫问前程 /7 |
| 生命中遇见的每一本书，都不是偶然 /11 |
| 开篇：为何而写 /13 |

1

Mac /1

| |
|-----------------------------|
| Macintosh 的命名 /3 |
| 1984， Mac 诞生 /5 |
| Macintosh 演化史 /7 |
| 说说我和 Mac /24 |
| 品评 OS X Mavericks——唯快不破 /28 |
| 免费的代价——从 OS X 免费谈起 /40 |
| 选择 Mac /43 |
| 开始使用 Mac /45 |
| 非同凡想 /56 |
| 怀念 2007 /58 |
| 年轻时的梦想还在吗 /60 |
| 苹果的语言 /62 |
| Mac Tips /64 |

程序员与编程 /97

| |
|--------------------|
| 并发的错觉 /99 |
| 程序员的性格 /103 |
| 程序员如何提高英语阅读水平 /106 |
| 普通人之殇 /108 |
| 趣谈个人建站 /109 |
| 人生元编程 /123 |
| 如何提问 /126 |
| 如何学习一门编程语言 /128 |
| 神奇的程序员——王小波 /135 |

目录

2

Linux 的文件系统王国 /137

科技与人文 /147

- 不要做一个 Hater /149
- 沉默的坚持和沉没的成本 /151
- 缅怀那些沉没的项目 /153
 - 锤子和钉子 /157
 - 读书日谈书 /159
 - 付费阅读 /162
 - 技术成长 /164
 - 克隆高手 /169
- 老兵不死，只能自我提升 /171
 - 没文化有人文 /173
 - 明天的科技 /175
 - 你有多少时间 /177
 - 自由软件 /179
 - 允许吹牛 /181
- 用好自己的份额 /183
 - 也无风雨也无晴 /186
 - 写作与编程 /189
 - 奇特的一生 /192
 - 是旅行还是长跑 /194
 - 西塘古色 /196
- 《晓说》不小 /197
 - 遗失的访谈——岁月无声 /198
 - 怎能忘了西游 /203
 - 重读黑客与画家 /207

人物 /213

- 传统的黑客——史蒂夫·沃兹 /215

目录

从汇编到太空——保罗·艾伦 /218
敬畏之心 /222
设计巨匠——乔纳森·艾维 /226

3 工具 /229

Vim /231
神兵利器——Alfred /253
终极 Shell /256

职场 /263

留不住的人才 /265
薪水几何 /266
顾得上就问，是为“顾问”也 /271
跟着老大去跳槽 /273
一生要面试多少回 /275
去创业还是继续编程 /285
去公司上班还是独自在家 /287
独自在家续篇兼答读者问 /290

Mac

Macintosh 的命名

Mac

1976 年，美国有两个叫 Steve 的孩子开了一家以水果命名的电脑公司，叫做苹果电脑公司，为什么叫苹果，据说是那阵儿乔布斯天天吃素，顿顿水果餐，瘦得跟干煸豆角一样，他闪烁着迷离的眼神（估计是饿的）和沃兹说，“我刚从苹果农场回来，咱公司就叫苹果电脑吧，阳光健康，还不吓人，有科技有人文，在电话黄页上还能排在前列，多好。”于是苹果电脑公司就诞生了。

Macintosh
的命名

3

现在我们知道，苹果公司所有电脑产品的命名都和 Mac 相关，比如 MacBook Pro、MacBook Air、iMac、Mac Pro 等，但是第一代苹果电脑却和 Mac 没任何关系，人家叫 Apple。开天辟地的是 Apple I，居功至伟的是 Apple II，后面写沃兹小传的时候会提到，这两个神作都是沃兹单枪匹马做出来的，乔布斯也就是焊焊板子买点披萨什么的。

有了 Apple I 和 Apple II 垫底，很快就衍生出一些新产品项目，我们把时针拨回到 1979 年，那时苹果公司有 4 款产品在研发，Apple II、Apple III、Lisa 和本文的主角——Macintosh。其中 Apple II 一直充满活力，在退出历史舞台之前都是苹果公司的支柱产品，Apple III 和 Lisa 就比较惨了，命运多舛，一会万般宠爱，一会无人问津，这种境遇就不可能做出好产品，结果 Apple III 只生产了 9 万台，Lisa 更可怜，1983 年推出，1986 年彻底终止，余货被埋在犹他州的垃圾堆里。这时候 Mac 的原型正在孕育，这是个微不足道的小项目，项目名称叫做“安妮”，项目的负责人是杰夫·拉斯金。



大家一看到“安妮”二字是不是立刻就懵了，洋气的苹果公司居然有这么激情四射的项目名称，但是具体为什么叫安妮，我也无从考证，大家还是来认识一下项目经理吧，这兄弟叫做杰夫·拉斯金，同样是一位技术牛人，是苹果的第 31 位员工。苹果公司的开创者们似乎都离不开人文与技术的情怀，拉斯金的专业是计算机科学，但是教过音乐和视觉艺术，在厌倦教书之后，就租了一只热气球，跑到校长家上空大声喊道，我不干了。谁这么辞过职？

Mac

Macintosh
的命名

4

拉斯金在自己的博士论文里写着，计算机应该是给人用的，不仅仅是给极客黑客各种家用的，除了神秘的命令行，还得有图形界面。所以他向世界宣布：“I have a dream，那就是为大众制造价廉物美的电脑”！

想到做到，1979年，拉斯金说服了当时苹果公司的管理者迈克·马库拉，成立了一个小规模的项目组用来研发廉价的、同时具备图形界面和命令行的电脑，当时这个项目代号就非常神奇地被命名为“安妮”，咱也不知道拉兄是怎么想的，终于有一天，拉斯金觉得这名字太女性化了，于是主动更换了项目代号，用自己喜欢的一种苹果（又是苹果！）来命名，叫做麦金托什（McIntosh）。但是为了避免与另一家音响设备制造商麦金托什实验室（McIntosh laboratory）重名，拉斯金对这个单词做了些微调，改为大家现在熟知的 Macintosh。

感谢拉斯金，如果不是他把 Mac 的名字改了，现在大家用的都是 iAnnie 和 Anniebook，那感觉有多么怪异。

1984, Mac 诞生

Mac

“所有伟大的事业都源于梦想，始于微不足道。”

拉斯金的梦想是“数以百万计的电脑”，是“如果个人电脑能够真正面向个人，那么任何一个家庭都该拥有一台”。但是 Macintosh 项目初期，整个项目组只有 4 名研发工程师，4 名啊，各位兄弟，这是在做操作系统和硬件工业设计，不是在写移动 App！就像秦始皇告诉蒙恬，“兄弟，哥有件事托你办一下，你挑选 20 名精锐勇士，去修一下万里长城，Okey? You can do it!” 但事实就是这么残酷，搁到现在您就算打破脑袋也找不到这样的团队！

这还没完，人少不算什么，您别没事就来捣乱啊。每隔一段时间 Mac 项目就会被拿出来讨论一下，讨论的主题不是加人加钱，而是啥时候解散？为什么还不解散？好在拉斯金也不是吃素的，每次都能凭借三寸不烂之舌让项目化险为夷，又惊又喜。终于有一天，Macintosh 被乔布斯盯上了，这下拉斯金老实了，他妈妈再也不担心他的项目被解散了，而是什么时候这个项目会被小乔据为己有。

根据乔帮主的各种言行访谈，我相信他对编程有着深刻的认知，但是根据各种史料传记，帮主应该是没有真正写过代码的。没编过程但是对编程有深刻的认识，这种行为是无法解释的，所以史称“天才行为”！

虽然不会编程，但是乔布斯非常欣赏拉斯金的想象力，在 Lisa 项目折戟沉沙之后，乔布斯盯上了 Macintosh，不过在帮主的词典中是没有“价廉物美”这个词的。他告诉拉斯金，“你太不高端大气了，搞什么价廉物美呢？咱有钱，不用考虑电脑的成本，要做就做完美的产品，可操作性、图形界面、性能，一样都不能少。”洗脑完毕，乔布斯开始让工程师为 Mac 换上性能强劲的摩托罗拉 68000，用来支持鼠标和华丽的图形效果。工程师这一次站到了乔布斯这一边，哪个程序员会不希望自己的程序跑在更好的硬件上呢？

终于，乔布斯代替拉斯金接管了 Macintosh 项目组。当时的乔布斯迫切希望用 Macintosh 再次证明自己，他开始动用自己的资源和现实扭曲力场来改造和影响 Macintosh 项目组成员，同时也吸引了更多优秀的技术天才加入 Macintosh 的研发。乔布斯甚至不切实际地希望 Macintosh 能在 1982 年左

1984, Mac
诞生

5

Mac

1984, Mac
诞生

6

右发布，他的激情和工作方式虽然让项目组成员间歇性崩溃，但同样在激励着他们奋力前行，实现了很多似乎不可能完成的天才技术创意。

Macintosh研发期间那个著名的缩短 Mac 启动时间的故事尤其让人印象深刻。有一天，乔布斯走进负责 Macintosh 操作系统的工程师的办公隔间查看电脑演示，随即开始抱怨系统启动时间太长了。像所有程序员一样，那个工程师开始解释……乔布斯打断了他，说“如果能够救人一命的话，你愿意想办法让启动时间缩短 10 秒钟吗？”工程师说可以考虑。乔布斯拉过白板开始计算，如果 500 万人使用 Macintosh，每天开机多用 10 秒钟，那加起来每年要浪费的时间大约是 3 亿小时，如果我们把这些时间节省下来，那每年就相当于挽救了数以百计的生命。

工程师一听吓坏了，几个星期后乔布斯回来一看，系统的启动时间减少了 28 秒！年轻时的乔布斯虽然工作方式方法上有很多值得探讨的地方，但是他能够看到宏观层面的东西，从而激励别人奋勇向前。

终于，时间来到 1984 年，在经历了数不清的延期之后，Macintosh 要发布了，乔布斯为此策划了著名的“1984”广告，这是一个注定名垂千古的广告片，但第一次给苹果董事会成员播放时，大多数人居然认为这是看过最差的广告片（你们就知道大部分光管理不干活的人有多么没品了吧）。不过乔布斯和沃兹都非常喜欢这个广告的创意，甚至沃兹提议广告费由他和乔布斯一人一半承担（厚道的沃兹）。经过努力，1984 广告按照原计划播放。

1984 年 1 月 22 日，在第 18 届美国职业橄榄球“超级碗”大赛中，苹果公司播放了这段构思独特的广告片“1984”。广告借用乔治·奥威尔的小说《1984》映射 IBM 对市场的垄断和不思进取，把 Macintosh 作为挑战 IBM 的新生力量，在广告的结尾，屏幕上出现广告词：“1 月 24 日，苹果电脑公司将推出 Macintosh（麦金塔）电脑。你将明白为什么今天的 1984 不会变成《1984》。”

“30 年过去了，IBM 似乎又开始不思进取了，但这次 IBM 的角色已经从当年的老大哥变为时代的 follower，能否跟上还要看其自身的变革力度。”

这则广告取得了令人震撼的效果，最终被《电视指南》和《广告时代》评为有史以来最伟大的商业广告。

从此，Macintosh 拉开了另一个伟大时代的序幕。

说说我和 Mac

Mac

说说我和 Mac

24

30岁之后，时间仿佛开闸的河水一样滚滚而去，感觉自己浪费的时间太多。我们不得不承认，先知先觉的人会比我们领先10年甚至更多的身位。进入21世纪，世界迎来了信息时代，人才竞争尤其激烈，与其他人感受不同的是，我觉得中国涌现出了很多非常厉害的年轻人，很多人年纪轻轻就成为某个领域的行家里手，端的是少年英雄。不过放眼未来，无论现在的你是年少成名还是大器晚成还是默默无闻，都需要不停地奔跑和追赶。

感慨完了，说说今天的内容，很多读者让我讲讲自己的经历，非常惭愧的是，

工作十余年并无可圈可点之事，实属籍籍无名之辈，谈之无物。倒是可以讲讲我对Mac的些许认知，中间也可以穿插讲点有意思的事情。今天是第一篇。

说起Mac，还得从2001年说起，那是我第一次接触Mac电脑，当时我在洪恩软件开发了一套叫做数字校园的软件系统，由于与一家厂商合作，需要把我们的软件移植到Mac Server上。



软件是BS架构的，基于JDK 1.3构建，由于一直在Linux上编程（当时Java几乎没有像样的IDE，Eclipse、NetBeans、IDEA等后来如日中天的工具，有的刚刚起步，有的还在孵化），所以Mac基本上是被我们当做Unix用的，印象中移植并没有太大的工作量，细节也不记得了，反正当时Mac对我来说就是一个Unix Server，以至于我现在完全不记得当年那台Mac服务器是什么样子了。

后来有另外一个组的兄弟要做音乐，公司专门给他配了一台Mac Pro，价格相当昂贵，那个兄弟估计也是没用过好东西，护得紧，基本不让我们这些土鳖程序员靠近，那时候Mac OS X已经告别了9，进入了10。如果记忆没有失误的话，那个系统用的是Mac OS X 10.2 Jaguar，其华丽的界面让一直用土土的Windows的程序员口水留了一堆，但当时我们已经被Linux下各种华丽但不实用的GUI伤透了心，像GNOME、KDE基本都是浮云。于是苹果在我眼中就是个酷酷的操作系统，界面优美，适用于图形图像视频制作，价格贵得离谱，用来工作娱乐什么的，基本上是天方夜谭了。

现在留下的印象就是头发杂乱的流浪歌手，谈着吉他，安详地坐在 Mac Pro 前调音和谱曲的画面。那个兄弟叫老郭，专门为软件做音乐，经常对我们这些不懂艺术的程序员说，“嗯，你们都是土鳖，就知道编程，多无聊。”当时我想，这话反过来说，也成！

我在洪恩软件一共工作了 3 年半的时间，这段经历让我的能力和见识得到了长足的成长。期间做过互联网（没错，就是传说中的第一波互联网神奇泡沫，一触即溃），基于 Perl 构建洪恩在线网站；做过企业级软件数字校园，基于 Java 和 JSP 技术；做过英语培训软件，基于 .Net 和 C；还管理过儿童产品事业部。不过让人伤感的是，我参与的大部分软件项目都以失败告终，洪恩在线遭遇互联网泡沫，数字校园过于超前，销量一般，英语培训碰到 2003 年非典，完全无人问津（这部分内容在“缅怀那些沉没的项目”一文中有述）。当时洪恩的主要业务还是电脑教育、高教、英语、儿童软件等，现金流良好。但安则思变，公司开始尝试其他的业务类型，比如互联网、游戏、企业软件、培训等，这些项目我基本都参与了，但成者寥寥，这对当时年少轻狂的我来说打击非常大，经常参与失败或无疾而终的项目会让人产生对己对人的怀疑和不自信。

要么留下来继续坚持，要么离开寻找新的挑战，这是很多人当时的选择。有些人选择坚持，有些人选择离开，留下的很多技术人员组成了后来的游戏公司完美时空的技术和策划班底。完美时空 2007 年在纳斯达克上市，目前发展得非常好。离开的人，则就各自有各自的生活和精彩。

所以说选择这个动作真的很神奇，人生漫漫征途，到处十字路口，每次选择就把你带向一个完全不同的路和沿途的风景，我们只能慨叹，嗟夫，人无常势，水无常形。

当时我选择了离开，那段时间是 1999 ~ 2003 年。在这段时期里，地球的那一边，乔布斯正在重新整肃苹果，进行了精兵健身并相继推出了 iMac 和 iPod。苹果正在一步步走向巅峰，只不过还没有看到这家没落帝国的潜力。在国内，Mac 仍然是稀有物品，我也仅仅见过两款。

后来工作上兜兜转转，使用的技术越来越多，工作范围也越来越广，但一直没忘关注 Apple 和 Mac，先是 iPod 火得一塌糊涂，之后 OS X 相继发布了

10.3 Panther、10.4 Tiger 和 10.5 Leopard。GUI 领域 Apple 强者恒强，界面炫酷，新功能层出不穷。类似窗口管理器的 Exposé，全文搜索 Spotlight，智能文件夹，Dashboard，强大的备份工具 Time Machine 等，再加上 Mac 纤薄的腰身、靓丽的面容，整个产品线变得越来越吸引人。

时光来到 2006 年，Apple 在 Mac 的硬件方面进行了重大的架构调整，开始全面采用 Intel 系列 CPU，Power 架构渐行渐远。硬件架构的调整和 Bootcamp 的推出，使得在 Mac 上安装双系统变得触手可及，这一点在当时还是非常吸引我的。在这期间，微软这个巨人似乎步履蹒跚，Vista 恶评如潮，被 Windows 折腾了很多年的我失去了尝鲜的勇气，依然用着古老的 XP，只是不断地更新微软的补丁，不断地重装系统。

2007 年，Apple 推出了 iPhone，如同 iPod 一样，再一次重新定义了产业格局，我想是时候尝试一下 Mac 了。而且手里的 IBM ThinkPad 已经用了 4 年，Mac 的价格也不那么离谱，除了 GUI 的吸引力，作为多年的 Linux/Unix 用户，Mac 下的原生 Shell 也是吸引我的一个重要原因之一。

于是我入手购置了一台 MBP985，操作系统是 Leopard（豹），也就是 10.5，Snow Leopard（雪豹）还没有出。当时对 Mac 能否完成日常的工作还是有一些担心的，同时对 Bootcamp 和 Windows 下的 C 盘、D 盘划分充满迷惑。这时程序员的劣根性和不作死就不会死的革命乐观主义精神，让我做了一个后续带来了无数麻烦的决定：分区方案采用旧式的 MBR（主引导记录）而不是 GUID（全局唯一标识）。320G 的硬盘，给 XP 分了 200G，而且做了 C 盘和 D 盘，最苦逼的是，为了让 Mac 的虚拟机能识别 D 盘，还修改了虚拟硬盘分区表。

当时我像一个勤劳的小蚂蚁一样对硬盘和系统进行了各种修修补补，以至于现在我基本记不起来到底修改了什么东西。现在看来，这些工作除了让我提高解决问题的能力和更好地使用 Google 之外，基本没起什么好作用。

很快我就从双系统（Leopard+ 虚拟 XP）切换到了单系统（仅使用 Leopard），由于自己经常在 Linux 下工作，所以 Mac 的 Shell 对我来说简直是天赐良缘。早年在 Linux 服务器端编程时，最爱的工具就是 SecureCRT。每天的工作就是从 Windows 下打开 SecureCRT，登录服务器上自己的账号开始编程，以至于后来我一直在 Windows 下构建自己的 Linux 环境，每台使用过的电脑，都必装 Vim/grep 等工具。所以我从 XP 切换到 Mac 下就像大自然一样自然，当时我的感觉是：怎么能这么晚才用 Mac 呢？

同时发出了“这才是电脑”的吼声。

我为自己在 Mac 下打造了全新的工作、娱乐、阅读和学习环境，后来基本上只有需要网银或多浏览器测试时，才会切换到 Windows 下。

2009 年 8 月 28 日，Snow Leopard (10.6) 全球发售，我撑到 11 月，终于忍不住升级了，痛苦的安装过程再次开始。为了把雪豹无缝安装到我的 MBR 分区上，我痛苦地思考了不作死就不会死的道理，然后花了一个周末，在不影响全部数据和程序的基础上把雪豹安装成功了，一直用到下一个操作系统发布。

最初分区的不合理性造成的影响是显而易见的：首先，无法升级固件；其次，没法直接升级大版本的 OS X（时间如白驹过隙，有折腾这个的功夫我看看书不行吗？）；第三，当初磁盘空间的不合理造成很大的资源浪费——已经很少用 Windows 了，却划了 200GB 空间，Mac 这边呢，磁盘经常满。我再次像个小蚂蚁一样，满了就搬一部分数据到移动硬盘上，艰辛的劳作让我非常痛苦，人生如梦，一樽还酹江月！

终于，下一个版本 Lion 要发布了。2011 年 6 月，苹果的 WWDC 大会上，Phil Schiller 对 Lion 的介绍给我的影响非常深刻。看完 WWDC 的主题演讲，我对自己的 MBP 分区说，该对你动手了，颤抖吧，小样儿。

Lion 如期发售，我用 Time Machine 备份了雪豹系统，用 Winclone 备份了 Bootcamp，分区修改为 GUID，然后重新恢复雪豹系统，购买、下载、安装、升级 Lion，空闲空间 200GB。哗……整个世界清净了。我指天对地地发誓，我再也不装双系统了，要用只用虚拟机！

从此，我算是踏上了不折腾的康庄大道，两耳塞耳，不闻雷霆，专心 Mac 上的读和写，一切变得无比和谐。

2012 年，Apple 发布了视网膜屏的 MacBook Pro。这个锐利的屏幕对程序员的吸引力是致命的，SSD 硬盘也变成了标配的，我的 Mac 终于有了更新换代的理由，整个系统顺利地迁移到了 RMBP 上，后续的事情你们都知道了，因为我开始写 MacTalk 了……

Mac

说说我和 Mac

27

免费的代价——从 OS X 免费谈起

Mac

免费的代价
——从 OS X
谈起

40

免费的代价——这个主题我封存在 MacTalk 的 to do list 里已经很久了，一直没有动笔写，直到今天看到知乎上的一个问题：“操作系统（OS）免费是进步还是退步，对软件行业有何影响？”我终于知道我原来在等这样一个问题。

背景：苹果公司在 2013 年 10 月的发布会宣布，OS X 10.9 和 iWork、iLife 系列软件免费。



首先需要明确的是，免费不等于开源，免费也不是盗版。国内很多用户听着盗版音乐，用着盗版软件，看着自己硬盘里几千本一辈子也看不完的盗版电子书，满心欢喜地赞叹：“啊，这真是个 Free 的美丽新世界！”您倒是美了，不要忘记，这些内容的创造者可能因为您的盗版行为过着凄惨的生活，他们原本早上可以喝双份豆浆、吃两根油条的。一半是海水，一半是火焰，有人欢喜有人忧伤，大致如此。

关于免费，我想到了这么几点，写出来探讨一下。

没有免费的午餐

如果说有的话，那是运气而不是惯例。

插播：法国作家福楼拜一生不为生计发愁，过着最为慵懒堕落的生活，声色犬马无所不为，那是因为人家是富二代，老爹给他留下了 50 万法郎。除了写作，他的生活就是起床、吃饭、抽烟、游泳、吃饭、抽烟、晒太阳、吃饭、抽烟、睡觉，不是富二代的感受一下……这是命。

这时候就会有杠头表示不服气了：“俺不是富二代，也能天天上网看免费网站，打免费游戏，用免费 App，怎么能说没有免费午餐呢”。杠头可能没注意，当他浏览网站的时候，会突然蹦出来个漂漂美眉告诉你：“黑头别乱动，教你一招全扫净”。因为他心思都在美眉身上，所以没意识到这是个广告！

当你免费享受网站服务的时候，付出的代价是广告的干扰，为你免费上网买单的是投放这些广告的企业主。那么这些企业的现金流从哪来呢？通过出售自己的产品或为客户提供服务获得，可能杠头的公司就是这些企业的客户之一……这样整个产业链就形成了一个闭环，就像大自然的生态圈一样。有产出，有输入，没有人或公司能够像永动机一样源源不断地提供免费服务和产品，如果有过的话，现在已经死掉了。

免费的模式

虽然没有免费的午餐，但是免费又是一种很好的商业模式。

你可以通过免费的模式积累用户和流量，然后通过广告获取利润，比如门户和搜索。你可以搭建线上的免费平台，获取线下的商家利润，比如电商、点评、订餐。你可以先提供免费服务，然后通过提供更多的附加服务把免费客户转换为付费用户，比如游戏、Evernote。多说一句，我之所以成为 Evernote 的付费用户，就是因为高级用户版本提供了文档演示功能，这个特征直接击中痛点，所以转化也属自然。如果你的企业在采用这种模式，那就要看有没有找到这样的痛点。

对于包含知识产权的作品，比如书、音乐、影视、软件等，则很少采用免费模式，但承载这些作品的平台往往是免费的，比如 iBooks Store、Kindle Store、iTunes App Store 等，对于普通用户，可以免费享受这些平台的服务。如果你想获取内容，那就需要付费了，你付的费用一部分给了创作者，一部分给了构建平台的企业。用户消费了内容，创作者和企业获得了利润，可以创造出更好的作品，提供更好的平台服务，这就是所谓的多赢。苹果采用的就是这种模式，当然，人家搞的是升级版。

无论采用什么模式，无论是个人还是企业，能盈利的模式才是好模式。现金流是王道，你必须挣到钱，然后才能生存。

苹果的大平台战略

谈到苹果，我们总是去说这家公司对软件和硬件的全盘把控，所谓软硬兼施，这当然是苹果的特征之一，而且不可效仿。但是无论是软件还是硬件，都是为了苹果的大平台战略服务的。

这个平台整合了 iTunes App Store、Mac App Store、iBook Store、OS X、

iOS、iPod 系列、iPhone 系列和 Mac 系列，当平台的规模大到足够的程度，平台本身和硬件设备已经为苹果赚取了支撑自己持续高速发展的利润。这时候，OS X 和 iOS 及其相关软件带来的边际收入已经不会影响整个平台的规模和发展了。从长远来看，开放这些产品给更多的用户使用，第一可以增加平台的粘性和用户的忠诚度，第二可以吸引更多的用户加入这个平台，实乃长治久安的良策。

2013 年 9 月，苹果发布 iOS 7 时，库克宣布 iOS 平台的 iWork 和 iLife 系列软件全线免费。那时我和几个伙伴就探讨过 OS X 免费的可能性，没想到这次苹果一步到位，除了操作系统免费，面向个人用户的 iWork 和 iLife 也一并免费，步子虽然大了点，但是并不扯淡。

这种基于平台的免费模式是免费的高级模式，当然是进步，不是退步，无论是苹果、谷歌还是亚马逊，大都在这个趋势上。目前看来，苹果做得最好。从长远来看，这种免费对巨型 IT 厂商应该有深远的影响，对个人开发者和用户也利好多多。

比如我，已经用上了免费的新版操作系统、办公软件和生活软件，赞！

至于微软，我想暂时对这个软件巨无霸应该没有太大影响，毕竟目前 PC 的规模依然惊人，但是按照平台和移动设备的增长速度，长此以往，就难以评说了。

选择 Mac

Mac

今天去苹果官网上扫了一眼，发现 MacBook 的价格比我之前购买的优惠了不少，加上有不少朋友让我说说如何选择 Mac，那就简单说两句。

选择 Mac

苹果目前 Mac 系列的产品有 MacBook Air、MacBook Pro、Mac mini、iMac 和 Mac Pro。

43

MacBook Air

首先来说说 MBA。如果追求轻薄便携的话，当然首选 Air 系列。如果选 Air 系列，首推 11 英寸系列，因为 11 英寸的键盘宽度和 13 英寸差不多，13 英寸只是增加了屏幕高度，所以在输入性上二者并无太大区别，但在移动方面 11 寸就能落 13 寸的几条街了。Air 系列目前是续航能力最强的笔记本，11 英寸款续航时间长达 9 小时。全系列采用了 SSD 硬盘，性能无需担心，打开程序文档基本秒开，所以不仅是商务人士，程序员一样可以选择 Air，你可以随时随地拿出 11 寸的 Air 进行编程、写作、浏览，无需担心热量、重量和续航。同时，如果需要使用大屏编程，接上蓝牙键盘、鼠标和外接显示器，即可打造一台强劲的台式电脑。



目前 MBA 全系列不支持 Retina 显示屏，应该是技术上还没有解决如此轻薄的笔记本如何适配视网膜屏。

MacBook Pro

MacBook Pro 是 Mac 系列里的中坚力量，深得广大人民群众的喜爱。如果你想追求更强的性能、更好的体验、更清晰的画面，那么我推荐你购买配备了 Retina 显示屏的 MBP15，MBP 系列里的顶配，也就是传说中的 RMBP15，笔记本中的跑车，跑车里的战斗机！如果你选择了足够大的内存和硬盘，可谓一本在手走遍天下，画面之清晰性能之强悍让你恨不得贫贱不能移威武不能屈。你可以编程、可以作画、可以视频、可以阅读，还可以建三个虚拟机并且同时启动，然后搭一个夹杂着私有云和公有云的混合云，什么 LVS、

Cloud Foundry、Hadoop、NoSql，能用的都给它用上。当客户问你们的私有云有 demo 吗，你沉默着指着自己的包包，都在这里，无比的高端大气！

当然 RMBP15 的缺点是价位较高，虽然已经变得轻薄了，但依然不适合手提斜跨奔波街头，电池的续航能力也是个问题，只能支撑 4 ~ 6 小时。退而求其次的话，可以选 RMBP13，屏幕、性能、CPU、重量、价位同步降级，也算个上选。旧版的 MBP 就不推荐了，基本上都是过渡产品。

Mac mini

Mac mini 分为普通版和 Server 版，操作系统也分为 OS X Mountain Lion 和 OS X Server。和普通 PC 不同，mini 就是一个铝合金的漂亮盒子，除了一堆接口啥都没有，但是价位低廉，性能强劲，适合学生或在经济性上考虑较多的用户，配上外置键盘、鼠标和显示器，应该是个很好的 Mac 解决方案。

iMac

iMac 是 Mac 的一体机，外观看上去就是一台帅到逆天的显示器（本文的配图），其中低调隐藏了主机和各种外设，屏幕分为 21.5 英寸和 27 英寸。购买前者还是后者，我觉得主要看你自己自己的书房和书桌大小，在苹果店看起来不算什么的 iMac27，放到自己书房，就一个字，大！有了 iMac 27，什么双显示器，什么 9 个 Space 都弱爆了，你只要把自己的窗口平铺开放到屏幕上就行了，还是那个字，大！如果是穷人家的孩子，第一次用 iMac 会眩晕的，你会说妈妈我再也不用窗口最大化了……

iMac 主要在工作室或家用，配合一个 11 寸 MBA 似乎是不错的选择。

Mac Pro

最后介绍一下 Mac Pro，这是 Mac 系列中动力最为强劲的机型，号称战斗机里的宇宙飞船，而且是桶装的……它那忧郁的光泽，浑圆的腰身，神乎其技的性能和存储，还有那基于 Xeon E5 芯片组的 12 核 CPU，都掩饰不住 Mac Pro 的出众，它代表了专业台式电脑的未来……

如果你们不信，看看这个：<http://www.apple.com.cn/mac-pro/>，如果不是专业人士，我不告诉他！

开始使用 Mac

Mac

写这篇文章的目的有以下两个。

第一，身边使用 Mac 的朋友越来越多，经常会有人跑过来问一些诸如此类的问题：“为什么把界面右上角的红色按钮叉掉，程序还没有关闭？”“为什么选中文件后回车打不开文件？”“妈呀，开始菜单不见了！”……另外，作为一个具备人文情怀的程序员，看到别人仅仅用到 OS X 的一些最基本的功能，常常扼腕叹息夜不能寐，所以干脆写个入门篇，如果有人胆敢再问类似问题的话，我就可以直接把这本书甩到他的脸上，说：读！

第二，作为一个没见过世面的 70 后，作为一个常年为 Apple 和 Mac 写软文没捞着一个儿子的程序员，我经常被若干不明真相的群众指认为 Apple 的枪手，这就搞得我很荣幸，既然给了这么高的荣誉，那咱就得把工作做得再扎实一些。

对普通用户来说，用好 Mac 主要有以下三点。

1. 理解 OS X 的基本结构和特点。
2. 掌握多手势和快捷键，少量即可，多多益善。
3. 用好工具，否则就算是把“屠龙刀”，搁庸人手里也是废铁一块，还得嫌沉。

理解 OS X 的基本结构和特点

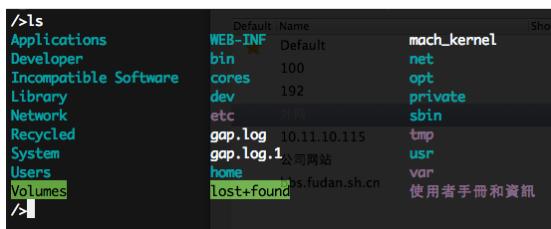
OS X 本身是基于类 Unix 内核的 Darwin 构建的，整个系统的设计思想、内核、文件系统、命令行都和 Unix 一脉相承，所以具备 Linux/Unix 经验的同学上手会非常快，也会感觉很顺畅。没有相关经验的也不用伤心，看完这篇文章，你们操作起来一样会快得让自己不好意思。

1. 用户目录

OS X 采用了 Unix 的多用户系统，所有用户的目录都在 /Users 目录下，这里面的 “/” 表示根目录。用户登录系统后，自己的用户目录下一般有公共、图片、文稿、下载、音乐、影片、站点、桌面、资源库（资源库在 10.7

开始使用 Mac

45



所以在 OS X 中，无需纠结于 C 盘、D 盘以及数据与程序的分离，因为它们本身就是分离的。没有特殊需求，不需要再进行分区，所有的数据都可以放在你的用户目录下，如果你觉得系统提供的目录不够，可以自行增加目录。关于数据文件，根据自己的喜好分门别类进行组织即可，也可以一股脑扔到 Documents (文稿) 目录下，需要什么文件，通过 **ctrl+space** 快捷键呼出 Spotlight，一问便知(在工具部分会做介绍)。其他图片、音像文件，按照系统提供的目录放置即可，简单明了不费脑。

2. Finder 和 Dock——资源管理

Finder 是 OS X 的资源管理器，原型来自早期的 Mac OS 系统，最初功能十分简陋。

我最早开始用 Mac 的时候，发现 Finder 既不支持页签，也不支持剪切，最令人发指的是，如果你想调整 Finder 的窗口大小，只能把鼠标移到窗口的右下角进行拖曳操作。当然在我知道了事实真相后，立刻觉得这并不是最令人发指的，因为所有的程序窗口都是这么设计的！那时我就想，苹果的设计人员也有脑子进水的时候。（有时候，我会觉得苹果标榜的“think different”的意思就是像精神病一样地去考虑问题。）

当然，系统发展到现在，除了 Finder 依然不支持剪切外，功能已经非常丰富了。Finder 提供了多种资源浏览方式，比如图标、列表、分栏、CoverFlow 等，左边栏是个人收藏和设备信息。你可以把常用的文件夹拖放到左边栏个人收藏区域，便于快速访问。近期新增的功能包括“我的所有文件、AirDrop 无线共享、Tag、页签”等。Finder 对所有文件都提供了 QuickLook (快速预览) 功能，想知道是什么效果的，选中一个 PDF 或 MP4，单击空格键即可。

在 Finder 中选中文件后按回车是对文件重命名，想打开文件可以用鼠标双击，也可以用快捷键 **command+o**。

那为什么不提供文件剪切的功能呢？为何不使用 Finder 插件实现剪切操作呢？

在 Finder 里选中文件或文件夹，单击右键，你可以看到“复制”和“拷贝”这样的菜单，但没有“剪切”。之所以这么设计，是因为 OS X 继承自 Unix，在 Unix 中没有 cut（剪切），只有 mv（移动），这似乎也更符合软件的隐喻，剪切更多表示剪掉一部分内容，而独立的文件是不能剪切的，只能移动。

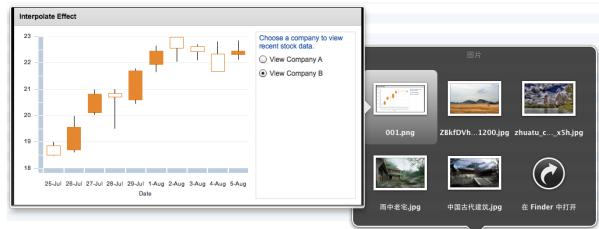
Mac

开始使用 Mac

47

如何通过快捷键实现文件移动的功能呢？用 command+c 复制后，用 option+command+v 可以实现 Move to here 功能，如果还需要复制到其他地方，可以继续用 command+v。

最后说说 Dock。Dock 是一个工具栏，一般在窗口的底部，相当于资源访问的快捷方式，除了对程序的快速访问，还提供了 Stack（栈）的功能。把需要快速访问的资源文件夹拖放到 Dock 中，在 Dock 右键单击相关文件夹，把显示内容改为网格，然后单击 Dock 中的文件夹，效果如下，同样可以 QuickLook，也可以回车直接打开文件。



3. 应用程序的安装和卸载

一直以来很喜欢 OSGi 技术，用了 Mac 以后，觉得 OS X 对程序的封装也很好。最后发现二者有一个共同点，那就是 Bundle，OSGi 以 Bundle 的形式封装 Java 程序和资源文件，而 OS X 中的所有应用程序都是 Bundle。

什么是 Bundle？大家可以把它理解为“包”，比如你出门旅行，随身携带的肯定是大包小包，你不可能把包里的东西拆散了插自个儿身上，就算你身上口袋多，安检也不能答应是不是？OS X 和 OSGi 对程序的封装，就采用了这种“包”的形式。

对于普通用户来说，你在 Launchpad（F4 键呼出）中看到的所有程序都像一个图标，但这个图标不是 Windows 中的快捷方式，而是封装好的 Bundle，从程序角度而言，这是一个文件夹，对普通用户来说，知道点这个图标可以打开程序就够了。Bundle 的设计是为了动态化，正是这种设计方式使得 OS X 中 95% 以上的软件的安装变得十分简单。如果你是从 Windows 平台转过来的，你会认为 Mac 上的软件安装和卸载简单得令人发指。安装程序就是把

XXX.app 拖进 /Applications (应用程序文件夹) , 卸载就是把程序从该目录删掉, That's enough! 对于有洁癖的同学, 可以使用清除软件 CleanAPP, CleanAPP 可以找到所有 App 相关的文件, 由用户决定是否一并删除。

大体上你可以这么理解, OS X 中 95% 以上的软件都是 Windows 中的“绿色软件”。

4. 磁盘映像

顾名思义, 磁盘映像文件可以直接挂接到 OS X 中, 其表现形式就像是一个磁盘分区。磁盘映像文件类似于 Windows 下常用的 iso 文件, 不过在 OS X 中后缀名是 dmg。双击 dmg 文件可以直接打开, 打开后在 Finder 左边栏的设备中可以找到挂接好的磁盘映像。dmg 文件是 Mac 下最常用的文件组织方式, 几乎所有的安装程序都是以 dmg 方式发布的。一般情况下安装程序就是打开发布文件 dmg, 里面有一个 app 文件和应用程序文件夹, 把 app 拖入应用程序即可。另外, 我们也可以使用磁盘工具对现有硬盘进行分区, 把 dmg 文件恢复为真正的硬盘文件。

5. 状态保持

OS X 下的大部分软件具备状态保持的功能。打开文件、浏览、关闭。下次打开该文件时会自动恢复到上次文件的进度。预览 (阅读 PDF) 、音视频软件 (如 iTune 和 MPlayerX 等) 、 Pages 、 Keynote 等都有这样的功能。 10.7 Lion 以上的版本在这方面做得就更过分了, 文件的全屏状态也能记录。比如, 你打开一个 PDF 文件, 全屏阅读, 然后使用 command+q 关闭, 下次打开这个 PDF 时会自动全屏。Lion 中着力宣传的新功能 “重返” , 就是记录所有窗口程序的状态, 关机重启后重新打开程序并恢复到程序关闭前的状态。

6. OS X 的维护

- ◆ 不需要安装杀毒软件。
- ◆ 定期通过磁盘工具进行验证和修复磁盘权限即可。
- ◆ 手动执行脚本 sudo periodic daily weekly monthly。

使用过程中遇到问题, 可以先来实施维护三板斧: 修复磁盘权限, 执行维护脚本, 重启, 基本上大部分问题就都解决了。遇到硬件相关的问题, 比如电源、灯光、风扇、触摸板等出现问题, 可以尝试重置 SMC, 具体可以参考 http://support.apple.com/kb/HT3964?viewlocale=zh_CN&locale=zh_CN。

另外, OS X 系列的升级相对平滑, 我从 Leopard 、 Snow Leopard 、 Lion 、

Mountain Lion 到现在的 Mavericks，一路升级上来，从未全新安装过。或者说，我自从用了 OS X，还没有重装过系统。

掌握多手势和快捷键

掌握好多手势操作和快捷键（少量即可，多多益善）可以有效地提高工作效率，触控板和快捷键让你基本可以脱离鼠标。多手势我就不说了，苹果的官网和操作系统自带的说明已经非常详尽。我简单介绍几个常用的快捷键。

Mac

开始使用 Mac

49

| | |
|-----------------------------|---|
| command+tab | 任意情况下切换应用程序 - 向前循环 |
| shift+command+tab | 切换应用程序 - 向后循环 |
| command+delete | 把选中的资源移到废纸篓 |
| shift+command+delete | 清倒相关程序的废纸篓（提示） |
| shift+option+command+delete | 清倒相关程序的废纸篓（不提示） |
| command+~ | 同一应用程序多窗口间切换 |
| command+f | 呼出大部分应用程序的查询功能 |
| command+c/v | 复制 / 粘贴 |
| command+option+v | 移动文件 |
| command+n | 新建应用程序窗口 |
| command+q | 退出当前应用程序。说明一下，所有应用程序界面左上角都有红、黄、绿三个小图标，单击绿色扩展到最适合的窗口大小，黄色最小化，红色关掉当前窗口，但并没有退出程序。用 command+q 配合 command+tab 关闭应用程序最为迅速 |
| command+l | 当前程序是浏览器时，可以直接定位到地址栏 |
| command+"+/-" | 放大或缩小字体 |
| control+space | 呼出 spotlight |
| command+space | 切换输入法 |

OS X 下快捷键非常多，对于普通用户来说，根据二八原则，用好上述快捷键，已经可以解决好大部分效率问题了。

用好工具

1. 搜索

Spotlight 是 OS X 自带的强力搜索工具，可以进行全方位搜索。借助 Spotlight，可以在输入文字信息时即时查找电脑上的任何内容。可以找文稿、电子邮件、应用程序、歌曲、联系人等，还可以通过设置获得快速定义或执行简单数学计算。Spotlight 菜单位于屏幕的右上角，带有一个放大镜图标。点按它即可开始搜索。默认热键是 **ctrl+space**。

Spotlight 的检索速度非常快，除了第一次初始化索引占用资源较多，平时都

开始使用 Mac

50



是增量索引，用户基本不会感觉到什么。

所以我在介绍用户目录和文件时说，在 OS X 下可以不进行文件整理，只要你记得文件的任何信息，通过 Spotlight 都可以快速定位到该文件，回车可以打开文件，按住 command 键单击可以打开文件所在位置的文件夹。

Spotlight 的高级用法，可以参考 MacTips 一章中第 130 个技巧。

2. Launcher

Launcher（启动器）的主要作用之一就是快速定位并启动应用程序，还可以当做计算器，定位文件，打开网页，Google 搜索等，是使用 Mac 的必备软件，有了启动器，你就不会再想念“开始菜单”了。相关软件推荐两款，都是免费应用，即 Alfred 和 Quicksilver。

目前 Alfred 发展更为迅猛，所以推荐使用 Alfred，在 App Store 可以直接下载，轻量级软件。默认热键是 Option+Space，如果你想打开 Twitter for Mac，只需按 4 个快捷键，option+space 呼出窗口，输入 t，回车。

在输入框输入计算表达式，如 200/999，会自动计算结果 0.2002002。输入 find 和文本信息，会找到最近使用的文件等。

Alfred 的功能远不止于此，其强大之处足以用一整篇文章描述。详细的使用说明，请参考本书中的另一篇文章“神兵利器——Alfred”。

3. 办公软件

推荐三套，即 iWork（Pages, Keynote, Numbers）、OpenOffice 和 Microsoft Office，前二套是免费软件，第三套是收费软件。

iWork 的所有软件都可以在 App Store 上直接下载，最新版本全部免费。微软的 Office 可以从官网购买，应该还是光盘介质。



目前我大部分文档工作都在 iWork 上进行，推荐大家在 Mac 上使用 iWork，上手很快，习惯后效率和效果都不错。具体请大家根据自己的使用习惯选择。

4. 浏览器

常用的浏览器有 Safari、Firefox、Chrome、Opera 等。Safari 是我使用的最多的浏览器。阅读列表和阅读器配合使用，效果非常不错。多手势的轻扫导航（在浏览过的网页间切换）和轻点缩放也是用户体验设计的精华之处。毫无疑问它们都是免费软件。

5. 文本编辑器

文本编辑器是大家常用的工具软件之一，除了 OS X 自带的原生文本编辑器之外，我再为大家介绍几款大家喜闻乐见的编辑器。

- ◆ Vim：号称“编辑器之神”，全键盘操作，充满速度感，良好的插件体系，几乎满足一切程序语言的编写需求。具体介绍请参考书中工具主题的 Vim 系列文章。
- ◆ Emacs：号称“神的编辑器”，捆绑了文本编辑器的操作系统。没了，大家感受一下。
- ◆ TextMate：Mac 专有编辑器，号称“Ruby 程序员的最爱”，当年 1.0 版 39 欧元一份，总共卖了十几万份，现在 2.0 免费开源，原来的开发者已经消失无踪，据说挣足银子去太平洋的小岛晒太阳了。
- ◆ Sublime Text：文本编辑器的后起之秀，发展迅猛，媲美 TextMate，跨平台，比 Vim 和 Emacs 容易上手，号称“性感编辑器”。

以上 4 款自成体系，都有完善的插件生态环境，大家可任意选择。

6. Markdown

我一般用以上文本编辑器进行编程和文字修改，说到大段写作，还得用 Markdown 软件，我常用的软件有以下三个。

- ◆ Day One：笔记类写作软件，支持大部分 Markdown 语法，按日期分类，同时支持标签、检索和导出功能，我的大部分文章都是用这个软件完成的。收费软件，可以直接从 App Store 下载。
- ◆ Byword：专注写作的 Markdown 文本编辑器，支持所有 Markdown 语法，提供简洁干净的写作界面，支持段聚焦、行聚焦和打印机模式。收费软件，可以直接从 App Store 下载。
- ◆ Mou：中国开发者编写的 Markdown 编辑器，支持一边写作一边预览，我常常用它来修改文章，实时预览效果，多种主题模式是 Mou 的特色。免费软件，接受赞助。

7. 文件比较

对于程序员来说，文件比较也算必备工具，OS X 中提供了原生的比较工具 FileMerge，不过这个工具对非 ASCII 内容的文件支持非常不好，我给大家推荐的是 VisualDiffer。

VisualDiffer 支持文件和文件夹比较，支持文件过滤、多重比较模式、颜色标注等，操作简单，响应迅速，实乃程序员居家旅行之必备工具。收费软件，可以直接从 AppStore 下载。

另外，习惯命令行操作的朋友，直接使用 diff 和 vimdiff，也是不错的选择。

8. 资源管理增强

以下几个软件可以让你的管理资源和窗口更加得心应手。

- ◆ TotalFinder：Finder 插件，为 Finder 增加多标签（类似于 Chrome 的多页签）、双面板、UI 设置等功能。收费软件。
- ◆ XtraFinder：Finder 插件，功能和 TotalFinder 类似，支持 tab、文件夹置顶、多窗口、剪切、全局热键等功能，重要的是这是一个完全免费的自由软件。
- ◆ Breeze：窗口管理软件，Option+1/2/3 分别对应最大化窗口、左半屏幕窗口、右半屏幕窗口，适合键盘控制。收费软件。
- ◆ Window Tidy：窗口管理软件，支持自定义窗口大小，以拖曳的方式进行程序窗口的管理，非常方便，适合鼠标一族。收费软件。

9. 风扇控制

推荐 smcFanControl——Macbook 的专用风扇控制软件，免费。OS X 对风扇控制并不敏感，CPU 使用率比较高的时候才会增加风扇转速，那时铝合金机身已经比较热了，尤其是在炎热的夏天。用这个软件可以自由控制风扇转速。夏天空调屋里一般 3000 ~ 4000 转就够了，冬天一般不需要开启。抱怨 Macbook 表面热的同学，可以试试，我保证你用了以后，“妈妈再也不用担心你的 Mac 发烫啦”。

10. 邮件

如果你已经升级到 10.7 Lion 以上的版本，系统提供的邮件将是你的不二选择。系统原生的邮件系统一直在进行改进，邮件会话的管理、全屏体验、个人收藏栏、智能邮箱等都是亮点。目前原生邮件的版本是 7。

重度 Gmail 用户可以使用 Sparrow 和 Unibox，有 Foxmail 情节的也可以选用

11. 输入法

OS X 的原生输入法已经有了很大改进，关于原生输入法的介绍可以参考 MacTips 的技巧 16。其他可选的输入法包括百度输入法、搜狗输入法、FIT 智能拼音等。

12. IM

OS X 系统自带的 Messages 是个选择，支持 iMessage、Gtalk、Yahoo 等，但是不支持 MSN，虽然可以通过配置方式支持，但比较麻烦。其他可选的有 QQ for Mac, MSN for Mac, Skype for Mac 等，还有整合工具 Adium 和 Trillian。

以前我使用 Trillian 比较多，整合了 MSN、GTalk、Twitter 等，表现稳定，用户体验也不错。免费软件。随着 MSN 渐行渐远，现在更多的使用独立应用，如 Messages、QQ、Skype 等。

13. 影音娱乐

MPlayerX 是 Mac 下的通用播放器，几乎适配所有影音格式。免费软件。由于新版本不适配 App Store 的沙箱，App Store 上的版本已经不再更新，可以直接去其官网下载。

射手影音最大的特色是智能匹配字幕，几乎适配所有影音格式，收费软件，可以直接在 App Store 下载。

VLC 是一款免费、自由、开源的跨平台多媒体播放器及框架，开发者可以基于该框架进行扩展开发，几乎适配所有影音格式，可以直接去其官网下载。

14. 下载软件

现有的网络条件让下载软件不像以前那么风光无限，断点续传这样的功能浏览器都能支持了，所以直接使用浏览器的下载即可。

命令行下的下载推荐使用 wget，打开终端输入 `wget`，后面跟要下载的链接即可。P2P 软件可以使用 aMule，功能和 Windows 下的电驴类似。

这里要特别推荐一下迅雷的离线下载功能。迅雷利用自己强大的硬件资源（带宽、服务器群和存储能力）可以实现快速离线下载，大部分资源直接秒杀，之后用户只需要用很短的时间把资源从离线迅雷服务器下载到本地即可（网

络情况好的话可以达到 M 级别的下载速度）。在下载方面，离线迅雷不提供任何资源，但是提供非常吸引人的服务能力，费用也很低，大概每月几块钱的样子。目前迅雷已经提供了 Mac 版客户端。

15. 虚拟机

主要有 VMware Fusion、Parallels Desktop(收费软件)、VirtualBox(免费软件)，具体可以参考 MacTips 的技巧 89。

16. 系统清理

Mac OS X 基本上不需要系统清理软件，有洁癖的同学可以考虑使用 CleanAPP 和 CleanMyMac 对系统进行优化和清理，它们都是收费软件。

17. 阅读笔记类软件

常用阅读笔记类软件有以下几个。

- ◆ iBook：原生阅读类软件，在新版操作系统 Mavericks 中首次亮相，支持在 Mac 上阅读 epub 格式的软件，支持阅读进度、标签、笔记等云同步，支持 iBooks Store，阅读体验优秀。免费软件。
- ◆ EverNote：优秀的笔记软件，支持多终端云同步，5.0 之后 UI 有了很大的改进。免费，可以直接从 App Store 下载。收费用户可以使用更为高级的功能，比如大容量文档、PDF 搜索、实时聊天、全屏播放等。
- ◆ Kindle for Mac：支持本地阅读和 Amazon 商店，支持中英文字典，电子阅读体验一流。遗憾的是不能整合中国和美国 Amazon 的账户，导致电子书商品也没法使用同一个账户阅读。免费软件。
- ◆ ReadKit：优秀的 Read it Later 和 RSS 阅读器，支持的源非常丰富（Feedly、Pocket、Instapaper、Readability 等），如果你依然喜欢博客和传统阅读，那么推荐使用。收费软件。
- ◆ Pocket：最好的稍后读 App，支持标签分类、编辑等功能，支持 Safari、Chrome 等插件，非常适合知识积累，免费软件。

18. 压缩软件

iUnarchive 和 The Unarchiver，都是 App Store 上的免费软件，支持大部分软件的压缩和解压缩。

iPack 的功能更强大一些，类似 Windows 下的 Winrar，收费软件，可以直接从 App Store 下载。

19. 词典

原生词典，支持屏幕取词，可以自定义添加词典，具体参考 MacTips 的技巧 76。

欧路词典，支持屏幕取词，免费软件，可以直接从 App Store 下载。

Mac

20. 备份软件

OS X 提供了非常方便的备份工具 TimeMachine（时间机器），看到这个名字，大家不要以为这是为了让你在厌倦了现代生活后穿越到清朝去玩格格和王爷的游戏，它是一款软件，专为备份操作系统和重要文件而生。

开始使用 Mac

55

我的第一台 Mac 用的操作系统是 Leopard，一路升级到 Snow Leopard → Lion → Mountain Lion → Mavericks，包括更换 Mac，从未重装过系统，我想这对于 Windows 系统来说是不可想象的，所有的功劳和荣耀都归于时间机器！

我个人每周会备份一次，如果你觉得自己资料非常重要，可以每隔几小时备份一次。具体的用法我就不介绍了，可以参考苹果的官方文档：http://support.apple.com/kb/HT1427?viewlocale=zh_CN。

本想写个简单的新手入门，结果罗哩罗嗦写了这么多，看来年龄大了爱唠叨这事儿真是伤不起。

对于使用 Mac 工作和娱乐的人来说，这篇文章的内容差不多就够用了。但是，如果你是个程序员，那么万里长征才刚刚迈了第一步，OS X 简直就是为程序员而生的操作系统，我们在后续的文章中会更多地涉及这一话题……

OS X 平台上有大量的免费软件可用，部分软件是收费的，有了 Mac App Store 后，可选的软件就更多了。希望大家支持正版，这样程序员就可以更为专心地写出更好的软件来回馈用户！

年轻时的梦想还在吗

Mac

年轻时的梦想
还在吗

60

每个孩子从小都怀揣梦想，积极上进，在充满无穷可能的选择里为自己编织最灿烂的明天。我想大家小时候都这样吧，没人从小立志要做个坏人或者流氓，很多时候别人问你的理想是什么啊？回答多半是科学家、作家、医生、老师，从来没见过一个孩子从小信誓旦旦地要做一个黑社会或贪污犯。即使人们长大了，由于各种原因变成了坏蛋和流氓，他们也希望自己的子女将来做个好人而不是坏蛋。

但是，你年轻时候的梦想只是你生命中的一个 Logo，而且是个会变的 Logo，比如苹果最初的 Logo 是这样，后来变成了这样（见附图）。我小时候因为作文被老师读，想当作家，因为参加数学竞赛，想当数学家，因为练毛笔字，想当书法家……这些都不重要，重要的是你在长大的过程中没有把



这个 Logo 丢掉，你还在想要做点什么或改变点什么，你没有变成小时候痛恨的那些人，这就够了。

人长大总是伴随着痛苦，现实变得越来越现实，那些无限可能都变成了没有可能或很小可能，这东西是很残酷的。当年我进入洪恩软件的时候，大概 24 岁，洪恩的总裁也姓池，比我大 4 岁左右，清华的本科生，当时已经拥有一家几百人的全国知名的科技教育公司。那时我暗暗地想，自己在 28 岁的时候，也要发展成什么样……这就是我当时的梦想。于是拼命工作、彻夜编程、努力学习，但是毕竟资质和眼界在那摆着，28 岁了，有提升，但是远远达不到我设定的梦想。这时候你就会慢慢认识到，想取得什么样的成就，或者想成为什么样的人，除了通过长期艰苦的练习和有意识的提升，资质、环境、勇气、运气等同样重要。

然后很快你就三十了，三十而立，梦想似乎已经远走，每个人都在感慨，这么多年付出了真心却收获了一堆下水，不知道那些真心和梦想都去哪了，不是说能量守恒吗？都去哪了？其实都在，它们就藏在某个角落默默地等着，一旦你准备重新上路，它们就会跳出来，指引你前行。所以最重要的是你还想做事，而不是混世。有人说不知道自己想做什么，那么最好的办法就是把目前正在做的事情做好，如果你把不感兴趣的事情都做好了，一旦你找到自

己的方向，那你得做得多好啊。

我自己离 30 已经很久远了，正在加速冲向四十不惑的那条金线。子曾经曰过，四十而不惑，好像是告诉你到了 40 岁就什么都明白了，我曾经期望着大彻大悟的那一天的到来。后来我发现子就是个骗子，他的意思是到了 40 岁，该明白的都明白了，不明白的估计你也不想去明白了，所以就不惑了。但现实的情况是，如果你一直在思考、读书、实践，做自己认为对的事情，到了 40 岁，你会发现不是不惑，而是有了更多的惑。这种未知的恐惧可能会伴随我们的一生……

附图是苹果 Logo 的变迁，大部分人熟悉被吃了一口的苹果图案，但不知道第一个图是什么意思。那是苹果公司最早的合伙人韦恩设计的第一代苹果 Logo，图案是一个缠绕了缎带的徽章，徽章正中是牛顿在苹果树下读书的场景。所以说乔布斯的苹果和牛顿的苹果，还是有点关系的。

乔布斯、沃兹和韦恩创建了苹果公司，但韦恩在公司成立没几天就卖掉了公司股份退出了，因为他觉得乔布斯和沃兹太孩子气了。多年以后苹果成了伟大的公司，有人找到韦恩，问他是否后悔退出苹果，他的回答是，一点也不，我喜欢现在的生活。所以说命中有时终须有，命中无时莫强求，后悔也没用，还不如不后悔！

Mac

年轻时的梦想
还在吗

61

程序员与编程

97

并发的错觉

程序员与编程

今天聊一聊电脑和人脑的并发问题。

在计算机发展初期，CPU 的计算能力非常有限，计算资源稀缺而昂贵。最早的时候一个 CPU 只能同时运行一个任务，这简直让人无法忍受。什么叫做只能运行一个程序呢？这就像大学上自习占座一样，一旦一本书、一张纸、一个包或一个活人占有了那个桌子，其他人就再也没法用了，无论是这个人出去上厕所，还是踢球，你都不能去用那个座位，如果你贼胆包天敢偷着去坐，这时候就会有个神秘人突然拍拍你的肩膀告诉你“同学，这里有人。”这就是常说的“见鬼的故事”。故事里的座位就是 CPU，无论当前任务在使用 CPU 进行计算，还是在读写磁盘 IO 或进行网络交互，它都得占着 CPU，黑客、极客和各种无证程序员们觉得，这，不，科，学！

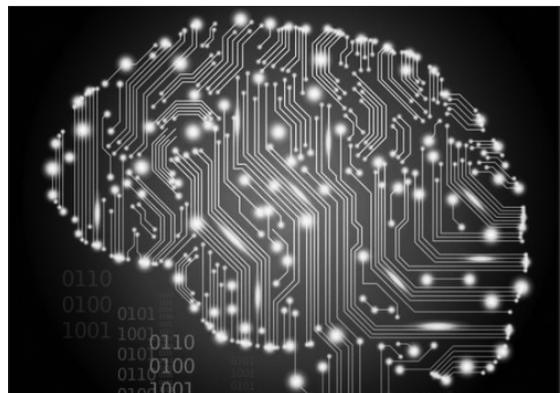
于是大家试图通过各种方式来改变这一现象。首先出场的是多通道程序。程序员们很快写了一个监控程序，发现当前任务不用 CPU 计算时，就唤醒其他等待 CPU 资源的程序，让 CPU 资源能够得到充分利用。但多通道的问题是调度乏力，不分青红皂白和轻重缓急，不管是急诊还是普通门诊，该等都得等。

第二个出场的是分时系统。分时系统是一种协作模式，每个程序运行一小段时间都得主动把 CPU 让出来给其他程序，这样每个程序都有机会用到 CPU 一小段时间。这时操作系统的监控程序也完善了一些，能够处理相对复杂的请求。早期的 Windows 和 Mac OS（注意没有 X）都是采用这种方式来调度程序的。分时系统的问题是，一旦某个程序死循环，系统就没招了，只能干等着，就像死机了一模一样，程序员们说，这是不可接受的！

第三个隆重登场的是多任务系统。程序员们让操作系统接管了所有的硬件资源，变得更加高级智能，系统进程开始分级，有的是特权级别，有的是平民

并发的错觉

99



并发的错觉

100

级别（你就知道，在计算机世界都是这个样子！），所有的应用程序以进程和线程方式运行，CPU 的分配方式采用了抢占式，就是说操作系统可以强制把 CPU 的资源分配给目前最需要的程序。程序员们成功了，几乎完美地控制了一切，并造成了很多任务都在同时运行的假象，如果用两个字来形容的话，那就是“和谐”！目前 OS X、Unix、Linux 和 Windows 都是采用这种方式进行任务管理的。

以上都是单核单 CPU 的情况，但无论线程间的切换多么快，这些都是并发，而不是并行。

好吧，中间插播一段并发和并行的区别。并发的英文单词是 Concurrency，平行是 Parallelism。如果一个系统支持两个或多个动作（Action）同时存在，那就是一个并发系统。如果一个系统支持两个或多个动作同时执行，那就是一个并行系统。也就是说，单个 CPU 永远无法同时执行两个或以上的任务，但是允许任务同时存在。所以，只有多核或多个 CPU 才可能发生并行，如果单核单 CPU 只能发生并发行为。

如果有人以为单核单 CPU 的并发就是同时执行很多任务，那么这是个错觉。

不知道解释清楚了没有。插播完毕！

终于，多核 CPU 和分布式系统被造出来了，一台计算机可以拥有多个 CPU，每个 CPU 可以有多核。同时，成千上万台的机器被连接在一起进行计算，大家一看都晕了，史称“云计算”。随着硬件的变化，软件技术同时开始革新，各种语言开始支持并行计算，比如 Erlang 的 Actor 模型、Scala 的 Message 模型、Go 语言的 goroutine 机制、Java 的 ForkJoinPool、Objective-C 的 Grand Central Dispatch 技术，当然还有 Hadoop 等分布式框架。

总之，到了目前这个阶段，无论是并发，还是并行，计算机和 CPU 都算是解放了。它们不仅在单台机器上可以执行并行计算，在横向扩展上也变得随心所欲，各种云平台应运而生，公有云、私有云、混合云……反正是比较晕……

人脑就比较惨了，在电脑突飞猛进的几十年里，几乎没有进展，脑袋仍然只有一个，也没有裂变出多核……

上文书谈了电脑的并发和并行的事情，有读者反馈，像一个有趣的教科书！

我最烦教科书，就因为教科书，我写了几年程序才把这些事捋清楚，为了让你们不再重蹈覆辙，我容易吗我！

好，下面我们谈一下人脑的并发，先看一个读者反馈，你们感受一下：

看过一个关于人脑的理论，说不清是并发还是异步。有时候我们很努力地想一个问题，但却怎么也想不起来，于是我们放弃了。但是大脑并没有放弃，此时它会自动发起一个 gorouting，继续在大脑的各个角落里寻找这个记忆碎片，当找到时执行回调告诉你。而此时你可能正边洗澡边哼着小曲儿！

关于人脑的机制，其复杂程度超过 CPU 何止万倍，比如上古奇人周伯通、郭靖、小龙女的双手互搏到底是并发还是并行呢，Mac 君万不敢断言，未来还是让研究人脑图谱的人去探索其真正的奥秘吧。我们在这篇文章中把“一心二用”或“一心多用”统称为人脑的并发。

俗话说“一心不能二用”，这句话常常送给那些做事三心二意的人，但是我们真的不能一心多用吗？或者说并发带给我们的到底是效率的提升还是状态的下降？关于这件事 Mac 君的看法是，不可一概而论。“好吧，那位同学请把砖头继续放入怀中，我们还没有讲完”。

关于人脑的多任务处理，应该从个人特点、所处环境和任务特性来考虑。

其实人脑天生就是用来处理多任务的，比如你可以一边洗澡一边唱歌，一边看电影一边磕瓜子，还要注意不要被飞来的砖头砸到，等等，不过这样的多任务都是在放松环境下的简单任务，对我们提升效率没什么意义。

但是，当我们在健身房跑步时听英语，写文章或编码的时候听歌（所有不让听音乐编程的公司都将死于心碎），坐地铁的时候阅读，步行的时候思考，这就变得非常有意义。因为我们在一个相对宽松的环境下把复杂的逻辑任务和简单的机械任务结合在一起，既不影响 A，也不会干扰 B，这种情况是我们优先要采取的并发策略。

类似的事情，比如开车时听英语，就要因人而异了。我有近 10 年的驾驶经验，喜欢开车，驾驶基本上已经形成下意识的动作，从出发到目的地往往不会记得自己做了哪些操作，所以我经常开车时听英语并有所收获。但有些人开车仅仅是驾驶已经够紧张忙乱了，倒一次车能车头入绝不车尾进，开次长途出的汗够洗澡的，那么就专心开车好了，车内最好保持安静或听一些舒缓的音

102

乐。

我曾经看过一本叫做《错觉》的书，书中有一段描述了一位机长在飞机飞行的过程中发现机上设备出了点小故障，于是他和副机长一起排查，接着又找来机械师，哥仨忙得不亦乐乎。过了一段时间，有人问，谁在开飞机呢？这时飞机无人驾驶已经很久了，等反应过来之后，飞机已经开始俯冲坠地，机上人员全部罹难！这种空难并不是意外，一架状况良好的飞机直接撞向地面不是偶尔发生，这种现象在航空领域被称作“可控飞行撞地”，其根本原因就是，人们太相信自己的多任务处理能力！

驾车虽然比驾驶飞机简单多了，但同样是一项非常危险的工作，所以我建议大家，听听音乐就好，另外千万别玩手机。

还有一种情况就是，在同一时间做两项或多项复杂任务，比如你让程序员在编码的同时帮助别人解决问题，能不能做好？也许有人可以，但我的感觉是，这种安排效率反而会打折扣。人们在很多时候会低估自己的能力，但在更多时候会高估自己。在复杂任务并发处理的时候，人脑往往会高估自己的处理能力，以为可以，其实任务的并行、上下文的切换、注意力的分散，都会让你的效率大打折扣，所以设计模式中的职责单一原则不是盖的，一个类尽可能只做一件事情，无论是效率还是后期维护都会好很多，人脑其实也是一样。

总结一下。

1. 简单任务的并发是大脑天生的 nature，每个人都在不自觉地应用。
2. 在宽松的环境中让简单机械的任务和复杂有机的任务并行完成是非常不错的做法，提高效率节省时间。
3. 在高危环境中（驾驶、高空作业等等）我们应该专心致志地只做当前的工作。
4. 对于复杂任务，我们最好一件一件完成，即使有些人能够同时处理多重任务，那也需要长期的艰苦训练，比如郭靖君，你能否做到，就得看有没有周伯通那样的大哥！

程序员的性格

程序员与编程

今天的话题来自一封读者来信，他在邮件里写了大概 2 000 多字，其中一大部分描述了自己的经历，我觉得非常精彩，写过汇编，写过 C，玩过 Flex，做过 Web (HTML5+CSS+JS)，还用过 Scala 和 OC，经历过创业人生，整个技术历程也足够丰富，可能比很多技术人员丰富。这样一个典型的程序员遇到的问题是什么呢？如下：

我是个静下来喜欢反思自己的人。我虽然对自己过去的一年挺自豪的，但我也发现了在我身上发生了一些变化。我发现我的幽默感渐渐消失，而且我很喜欢从“另一面去看问题”，一开始我觉得这是我具备了独立思考的能力，不会轻易被别人的言论左右。但这也给我身边的人带来了困扰。因为当他们说出一些东西的时候，我首先的反应就是：可能并不是这样，那样也许更好，其实是这样的……而且几乎是反射性的，不假思索就能找到问题的另一面，并且表达出来。然而说者无意，听者有心。朋友之间，说说笑笑还好，他们最多觉得是我很喜欢抬杠、反驳别人，送我外号“杠王”。但我老婆和我生活在一起，就受不了了，觉得我太自负，自以为是，愤青……

我自己知道，我不是故意这样的，但这已经成为了下意识的行为。虽然这现象以前就有，但是这一年变化得很明显，明显到我自己都察觉到了。我不想变成一个让身边人觉得不舒服的人，我也怕以后会发展成“阴谋论”者。我不是不懂得谦逊，只是对自己的观点比较执着，而且想强烈地表达出来，并且希望别人也赞同。到后来，我自己都分不清楚，我到底是真的发现了这些事物的“猫腻”，还是我为了反驳而反驳的。我觉得这是一个很不好的习惯，因为我身边的人和我探讨某些问题的时候，在我这里得到的几乎都是否定的结论。

就我关注的一些技术前辈，包括我同学，也都是这样。我发现我们这些技术同类，好像“戾气”都比较重。而您在我关注的人里，是属于比较温和的，没有那么张扬，没有显示出“自负”和“自以为是”，我想请教下，您是否

程序员的性格

103



遇到我这样的问题？而您又是怎么控制自己的？为了表述清楚，我可能把我的问题夸大了一点点，但我觉得如果不改善的话，迟早会变成那样，甚至更坏。

其是我想说的是，大部分牛人都很张扬，“戾气”也重，我之所以比较温和，一个是因为岁数大了，另一个就是确实不够牛。另外的一点就是，当牛人牛到一定的境界，可能就会重归和平，比较文艺的描述就是：“大牛领会了返璞归真和万物生长的道理，知行合一，遇事抖抖衣袖，不溅起一片涟漪。”你现在浑身都是杠头和愤怒，其实也只是不够牛而已。

当然，我说的这种张扬也好，愤怒也罢，主要是反映在你的工作和技术领域，如果你把这些情绪完全带入到生活中，那就比较危险了，严重时会导致人见人踹，花见花败，除了至交好友，少有人愿意搭理你，如果再不悔改，可能会像你预想的那样，再也没有人愿意和你交流。这种结果很容易理解，换位思考一下就好了，如果说的每一句话和每一个观点都遭到聆听者的反驳和质疑，那这种交流就没法进行下去，如果是工作中的讨论还可以商榷，真理越辩越明嘛，但是在生活中，很多时候大家就是聊聊天，舒缓一下心情。比如对方见面问，“你好！”你说，“你怎么知道我好？”这时候板砖就飞过来了。

程序员有性格是好事，但是谁也不可能永远都对，多聆听少说话不是坏事。我有一个朋友，我知道他学识非常渊博，但是每次大家聊天或讨论问题，他都处于一种聆听的状态，频频点头若有所思，但是如果你征求他的意见，就会发现他总能一针见血、一剑封喉，一下就能找到问题的关键。我就知道了，他一直在从别人的谈话中获取自己需要的知识。当然，他不是程序员。

程序员一般都比较自负，我年轻也是一个德行，谁要是对我的代码说三道四，恨不得掏刀子和他拼了，但是慢慢你就会意识到，不停地反驳别人不会证明自己的聪明和独立思考，正确的讨论技巧和解决问题才是王道。比如有人提出了一个创意，你觉得有问题，可以这样说：“我说 Mac 君啊，你今天的文章总体来说还是不错的，照顾了大多数的读者，观点也比较新颖，但是呢，似乎没有重点还多了点狗血内容，如果能够……一下，就更完美了。”这种说法基本上会让你避开板砖与西红柿齐飞的场面，并造成大家在同一战壕的“假象”，有利于迅速有效地解决问题。

当然有人会说，乔布斯骂人都是直接说人家狗屎的，哪那么多弯弯绕呢？其

实很简单，因为他是乔布斯，你不是。

总之，程序员要保持自己的性格、激情、愤怒，这样你才能写出传世的代码，同时也要温和、有理有据有节地与别人探讨问题，还要有健康的生活和好的娱乐活动。除了技术书籍，多看一些人文类的著作，有助于完善自己，善待他人！

今天似乎写了一堆碎碎念，也不知道是否回答了这位读者的问题，就这样吧，你们可以认为我说的都是错的！

程序员的性格

程序员如何提高英语阅读水平

程序员与编程

程序员如何提高
英语阅读水平

106

问题：作为一名程序员，虽说每天都在和英语打交道，但是当看到一篇英语文档或者英语技术文章的时候还是比较头疼，理解他们的意思也只能是20%。尤其是使用Google搜索的时候，很多问题解决办法都是英文的，还有一些国外比较有名的网站比如Stack Overflow，上面也有很多学习的资源。怎样才能让自己顺利阅读这些技术文章呢？

回答：其实学英语和其他技能没什么太大区别，无论是你想在英语阅读方面提升自己，还是想在口语或写作方面提升自己，都需要进行长期的不间断的练习，坚持一段时间后（时间长短根据你自己的效率、每天用时、频率都有关系），你会发现自己的水平自然就提升了。举个例子，以前写博客似乎是最难坚持的，但是如果你每个月都能写一篇略有价值文章的话，5年就会有60篇高质量的博客，你几乎都能集结出书了。到了微信时代，似乎写微信公众平台是最难坚持的，但是如果你能坚持一年，那就有300多篇文章，于人于己，都是一份宝贵的财富。

作为程序员，英语阅读能力是最基本的要求，相对口语和写作来说也是最容易达到的，因为计算机类图书的那些常用单词就那么多，多读几本英语类技术图书，想不认识都难。

一个相对容易坚持的办法就是，找一本和当前工作相关的、急需的技术图书，每天拿出一小时阅读，不认识的单词，如果不影响阅读可以不查词典。如果某个单词多次出现，那么就该查下词典并计入生词本。如果你能坚持查阅英英词典，那么提高就会更快了。每天坚持一小时，这本书读完，你就会发现自己的阅读能力提高了一大截。

这样看来提高阅读能力似乎很简单，但是，问题的关键是，你能否坚持下来，尤其是阅读初期，你只能看懂20%，需要不停地查字典，单词记了又忘，几个星期过去了，进展依然缓慢。这时候最容易放弃，你会痛苦地怀疑自己，我究竟是不是这块料。

毕竟人类的大脑都是倾向于舒适和懒惰的，谁都知道看美剧刷微博，是容易的、愉悦的。学英语和学编程是痛苦的，有时候你甚至会有意识地去避免开

始这件事情，先干点这个，再干点那个，熬到最后，发现没时间了，再拿起书来读一会，困了，今天先睡吧。就这样，一天天很快过去了，你发现自己似乎每天都在坚持，但依然没效果。但事实是，你既没坚持，也没效率，这根本不是刻意练习。

提高英语阅读能力这事，比提高英语口语和写作能力容易多了。如果程序员缺乏英语阅读能力，这将是你非常大的一块短板，如果意识到了，就尽早补上。

另外说一点，英语翻译对提升英语阅读有一定的帮助，但这个要求比较高，必须要查字典，遣词造句，力求原汁原味地表达作者的意图。翻译并不是提升阅读能力的捷径。我曾经翻译过很多文章和半本书，没觉得自己英语水平有很大提升。

即便是最简单的坚持，最后的结果也都是惊人的。这世界上一直存在一条路，让我们的能力从平庸到杰出，这条路漫长而且艰辛，只有少数人愿意走下去，所以，优秀的人永远是少数。

程序员与编程

程序员如何提高
英语阅读水平

人生元编程

程序员与编程

如果你是个诗人，那你的人生就是一场风花雪月的事；如果你是个演员，除了学好《演员的自我修养》之外，你的人生就是一场接一场的剧本；如果你是个侠客，对不起，现代没有侠客；如果你是个程序员，好吧，你的人生将由一行行飘逸的代码和捉摸不定的 bug 组成，所谓编程人生，就是你的一生已经与编程密不可分，为代码欢笑，为 bug 忧伤。

那一年你初入江湖，你不懂什么是汇编，什么是语言，你搞不懂 Lisp 和 Smalltalk 的区别，为什么 C++ 比 C 多了两个加号就成了对象，2000 年以后那么多人都在用 Java，现在却说 Objective-C 是最贵的语言，这又是为什么？

你对着大海说，我要学尽天下武功！大海对你说，
你算术不太好……

孩子，世界上的编程语言成百上千，常用的也有数十种，光学会这些语言你的时间就得用微积分计算，学完以后估计手抖得都敲不动键盘了，哪还有时间创造奇迹？更别说了语言你还得掌握前端后端 UI 体验，这个数据库那个操作系统……

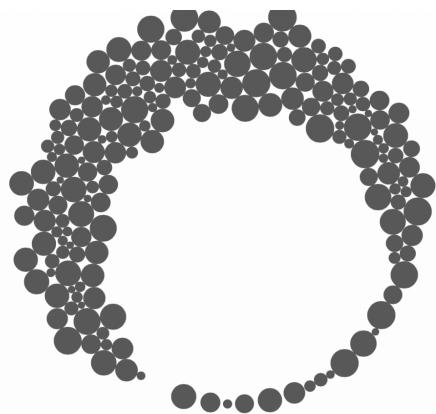
你在知道了这些真相以后，依然痴心不改，抹干眼泪冲到编程兵器排行榜“TIOBE Index”面前，挑选了前十名开始勤学苦练。你在满天星斗的夜色中编写 C 程序，在清晨的微光中调试算法，上午你敲打键盘输出日志，中午吃完五又四分之一口米饭之后就匆匆离开，因为你要去看看系统为什么崩溃……你学会了 5 种语言、3 种操作系统和 4 种数据库，你写了一个 MIS、两个 OA、三个 App，你觉得你开始了编程人生，其实是你的人生被编程了，你被代码驱动和驱赶，你变得疲惫不堪。

这次你在清晨的寒风里对着高山说：“这是为什么？”高山对你说：“因为你不懂元编程！”

好吧，扯了这么多其实是想和大家谈谈元编程的事儿。元编程？估计小白一

人生元编程

123



听又懵了，啥是元编程呢？与云计算、大数据不同，元编程并不是一个抽象的概念和名词，这里面代表了很多务实的技术，相伴而行的概念还有元数据。

元在英文里就是 meta，元编程就是 meta programming，元数据就是 meta data。元编程就是能够操作代码的代码，元数据就是能够描述数据的数据。

听完这样一个介绍，大家是否更加晕菜了呢？如果回答是肯定的，那么效果达到了。

在接着介绍元编程之前，我们先看一下代码的世界。如果把代码比作一座小镇，那么其中的类、函数、方法、变量、代码块和宏，就是小镇上安居乐业的居民，他们相互协作，相互依赖，一起建设着美好家园。

在能够支持元编程的语言世界里，你可以和这些居民打招呼，还可以进行内省（introspection），获取其自身的一些信息和行为，甚至你能够为这些居民动态增加一些能力和行为，或者在这些居民奔跑的时候改变他们的行为，或者创建一些新的居民。这样的语言有 Ruby、Python 等。

在不支持元编程的语言世界里，大家分为两个状态：编译时和运行时。一旦编译器完成了自己的工作，这些方法和函数就看不到了，它们成为内存中的幽灵，你只能通过固定的方式使用它们，而无法获取它们自身的信息。当然，即使是这样的语言，为了增加编程的灵活性，也要通过各种方式来提升元编程的能力，比如 Java 和 C# 笨手笨脚地使用反射方式，C++ 则通过模板方式，但古老的 C 就无能为力了，因为它没有元编程能力。

现在我们就知道了，编程语言虽然各有侧重，但是语言和语言之间的能力和特点区别还是很大的，不管你现在使用的是什么语言，我都建议你们去学一门具备原生的元编程能力的语言，比如 Ruby、Python、Lisp、Objective-C 等。

我第一次接触元编程和元数据还是在一家外企，那家外企的名字和火箭有关，他们有很多年纪一大把的老程序员，据说是制定 CORBA 标准的牛人，他们在这个火箭公司开发了一套分布式的软件平台，名字不能提，因为老外的版权意识太强了。我一位前同事移民国外，只是在自己的开源项目中引用了一点平台文档，结果一纸法院传票追杀到异国他乡，而且直接导致这个同事的上司被辞退。“好吧，上司不是我，不过我当时确实想过，如果我引用了他们的代码，也许会见到真的杀手吧。”

这套平台的持久化、权限和业务逻辑引擎都采用了元编程和元数据的方式实

现，实现语言是 Python。当时看到那些优雅的代码，我再次感受到编程的魅力，原来代码还可以这样写！我在那个外企的两点收获：第一是平台和元编程；第二是版权意识。后来当我有机会主导从头构建一个软件开发平台的时候，我吸取了这些思想和经验，基于元编程的思路构建了平台组件数据字典，你可以编写少量代码或不编写代码就生成各种业务应用，这就是操作代码的代码，描述数据的数据。

这时候就有同学问了，你啰哩啰嗦扯了这么多元编程，干嘛标题叫做“人生元编程”？

因为无论是编程还是人生，都是相通的，想清楚了这一点，你就会觉得百无聊赖，因为万事万物要么是熊样要么是鸟样，都脱不出那个框框。具备元编程的语言就具备更强大的操控自己的能力，可以自省，可以反射，可以动态改变和控制自己；具备人生元编程能力的人，同样有自省能力，随时检查和控制自身的情绪和行为，思考自己的想法，改变大脑的动机。

举个简单的例子，当你的理智告诉自己 9 点就必须开始看书学习的时候，你的大脑会对你说：“亲，可以再看会儿电视呦，你看沙发都这么舒服……”

缺乏元编程能力的回答是：“那……就再看会儿”，具备元编程能力的回答是：“不行！”

如何学习一门编程语言

程序员与编程

如何学习一门
编程语言

128

关于学习编程这个主题，有各种读者多次要求写一写，而且要求文艺地写、抒情地写、充满社会主义特色地写，要做到：问题看起来巨复杂，读起来巨简单，学起来巨容易！看把你们惯成什么样子了，Mac 君你继续去面壁吧。

好吧，不管他，我们接着聊。

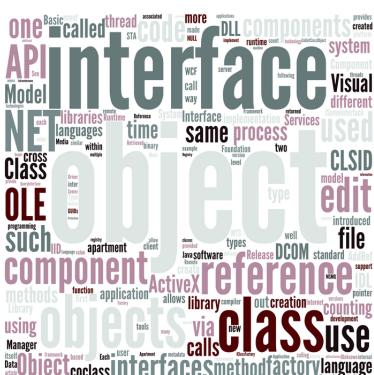
如果你准备未来投身到 IT 江湖从事编码这份有前途的职业，学习一门语言显然是远远不够的，就像你初入江湖，告诉别人：“兄弟只会太祖长拳！”问：“Level 可及乔峰？”答曰：“不及万一。”人家一看你就是 P2 的命，PK 时

一个大招直接秒掉。写到这我想起了一个叫做冰河的兄弟，也是奇葩一朵，在程序语言方面一生只爱 Lisp (Lisp 号称编程语言的祖宗)，在人类语言方面则除了中英文，还在同时学习意、法、西、德等 4 门语言，而且不是随便学学，神志还不错乱，这一点让我简直佩服到逆天。一门英语已经从初中折磨我到现在了，在人类语言层面，我常常是被秒杀的。

所以，如果编程有可能成为你的职业，那么 5 ~ 10 年的学习和实践时间是需要的，因为你可能要学习编程语言、操作系统、算法、数据库 (SQL 的 NoSQL 的)、Web 开发等，

还有各种数不清的引擎和架构，特别令人发指的是，当你熟练地掌握了一门技术之后，就会有位赤脚大仙走过来告诉你：“孩子，你学的技术已经不是方向了。”然后在你绝望的眼神里飘然离去……写 Java 的兄弟感受一下……

如果你的职业发展与编程无关，只是想学习一门语言磨练人生意志，那么这事就比较容易了，根据自己常用的操作系统选一个就好，可选的语言有 Python、Shell、AppleScript、Ruby 等。如果你用 Mac，这些语言可以任选，而且环境都是现成的，如果你用其他操作系统……，对不起！MacTalk 只说 Mac。这些语除了能够帮助你锻炼意志，提高逻辑思维能力，同时还可以在某些关键时刻帮助你处理各种繁琐复杂的工作，比如大量文本、定时任务、自动化任务、编写常用小工具等，还可以引发跨界编程的轰动效应，不信的话去百度搜索“Python 女神”便知。



好的，写到这如果还没有打消你学习编程的热情，那就可以继续往下读，下面才是正文。

要有光

无论学习什么，一定要有明确的目的和目标，如果是抱着玩票的心态，那么最多就是能够“知道……”，而达不到“学以致用”。所以，搞清楚自己为什么要学习编程，准备学习哪门语言，要达到什么程度，想用多长时间等问题，在你的头脑里有个大概的思路和计划，就基本解决了 Why 和 What 的问题。下面我们来找 How。

多说一句，其实学什么都是有用的，大部分时间你只是不知道会在什么时候用在什么地方。

程序员与编程

如何学习一门
编程语言

129

经典教程

选定了语言不要着急去网上搜索各种秘籍、评价和下载各类盗版电子书，每个技术领域都会有一些经典的圣级别的图书，找到它们，购买一本纸质书或电子书，最好是带练习题的，可以边学边做。

如何找到这些图书，豆瓣读书网应该是个不错的选择，虽然豆瓣的电影评价口味过于小清新，但图书评价还是值得信赖的，另外找乐于分享的老鸟推荐一下也是个不错的选择。

掌握基础，持续练习

每一门编程语言的学习内容都会涉及：基础运行环境、数据类型（数字、字符串、数组、集合、字典等）、表达式、函数、流程控制、类、方法等等。不同的语言还有一些不同的特性，这些内容并不复杂，尽快通过大量的练习击倒它们，然后再去深入了解面向对象、并发、异常、文件与目录、网络、标准库等内容，并辅以持续的练习，这些内容才能够让你真正进入编程领域并做出实际的软件。

初学者每天花 1 ~ 2 个小时是需要的，尽量保证阅读和练习的持续性和时间长度。其实 1 ~ 2 个小时根本不算什么，想想你们花费在看电视和刷微博上的时间吧，如果说没时间，那就是不抽不舒服斯基了。

外事不决问 Google

现代人的生活和学习是如此的方便，因为我们有 Google！俗话说内事不决问百度，外事不决问 Google，技术绝对属于外事，你要是去问度娘技术问题，被人家的回旋踢踢飞可别怪我没告诉过你。

以前学习技术只能通过技术图书和口口相传，现在遇到问题从 Google 那里就可以找到答案，所以用好 Google 你就能如猛虎加之羽翼而翱翔四海。如果你还在认为 Google 就是个搜索框，那就图样图森破了，Mac 君今天为你推荐以下两篇文章。

1. “Google, Google, 再 Google” (<http://wordpress.lixiaolai.com/archives/7572.html>)。
2. “如何用好 Google 搜索引擎” (<http://www.zhihu.com/question/20161362>)。

让你的搜索与众不同。

用好工具

俗话说得好，“欲练神功挥剑自宫”，sorry，不是这句，“工欲善其事，必先利其器”。想要学习编程一定要写代码，我们不提倡咬破手指写 bloody code，所以一定要找到趁手的武器。我个人把工具分为三种：第一种是部分程序语言自带的 Shell，第二种是文本编辑器，第三种是集成开发环境（IDE）。

1. Shell

如果你在学习 Python，那么 Python Shell、bpython 和 IPython 都是不错的选择。如果你在学习 Ruby，那么 IRB 就是 Ruby 的 Shell。如果你在学习 Shell，打开终端就是 Shell。如果你在学习 Java 或 Objective-C，对不起，这些语言没有 Shell。

Shell 能够单步执行你的编程语句并给出即时反馈，这种交互式编程方式非常适合初学者，所见即所得，所以凡是提供 Shell 工具的语言，推荐大家优先使用 Shell 学习。

2. 文本编辑器

这个领域向来是“员家必争之地”，溢美之词和吐槽之声交相辉映，从古至今绵延不绝，说起来都是眼泪。比如 Emacs 和 Vim 程序员，大家沿着不同的道路和目标前进，但总是会在某个点交叉相遇，见面就扔石头和臭鸡蛋，砸得对方鼻青脸肿，然后擦擦眼泪和口水继续前行。还有 IDEer 说 Vimer 装，Emacser 说 IDEer 垃圾……种种血淋淋的事实足以拍一部惊悚科幻动作言情片。

我自己比较喜欢文本编辑器，但是也不排斥 IDE，这种人俗称两边不待见，但我还是那句话，不为自己设限，不同的环境应该选择最好的工具。下面给大家推荐几款文本编辑器。

- ◆ Vim：号称编辑器之神，全键盘操作，充满速度感，良好的插件体系，几乎满足一切程序语言的编写需求。
- ◆ Emacs：神的编辑器，捆绑了文本编辑器的操作系统。没了，大家感受一下。
- ◆ TextMate：Mac 专有编辑器，号称 Ruby 程序员最爱。当年 1.0 版一份 39 欧元，总共卖了十几万份拷贝，现在 2.0 免费开源。原来的开发者已经消失无踪，据说挣足银子去太平洋的小岛晒太阳去了。
- ◆ Sublime Text：文本编辑器的后起之秀，发展迅猛，媲美 TextMate，跨平台，比 Vim 和 Emacs 容易上手，号称性感编辑器。

以上 4 款自成体系，都有完善的插件生态环境，诸君可任意选择。

对于 TextMate 开发者赚了钱就跑的恶劣行径，大家完全可以批判，有时我们不得不痛苦地承认，国外程序员的鸡贼是我泱泱大国之 IT 民工永远无法理解的“泪”。

3. 集成开发环境 (IDE)

IDE 是图形化的集成开发工具，具备精准的词法分析、编程提示、调试等功能，功能之繁复用户自知，如果做工业级编程和团队协作的话，还是推荐使用 IDE。

在这里推荐几个系列。

- ◆ Eclipse 系列，通过插件方式几乎支持所有的常用编程语言。免费。
- ◆ JetBrains 系列，产品线丰富，几乎都是精品，Java、Python、Ruby、PHP、Objective-C、Web 等一应俱全。收费。

- ◆ Xcode，Mac 上优秀的集成开发工具，所有的 Mac App 和 iOS App 都出自它之手。免费。

微软的技术不懂，就不推荐了，嘿嘿……

程序员与编程

如何学习一门
编程语言

132

找到你的 Master

小时候看《西游记》发现，师傅原来是用来人肉的。后来看《天龙八部》发现，牛人都不需要师傅，即使有也是要被别人一掌震飞的。再后来看《射雕英雄传》发现，愚钝的人首先得有师傅，其次得有很多师傅，再次每增加一个师傅功力都以指数级别增长，2、4、8、16……

所以，如果有人告诉你三人行，一个老师都没有，你至少要质疑这一观点，同时考虑自己会不会筋斗云，是否天赋异禀以一当百等等。如果不成，那还是去找师傅好了。

有老师的好处有这么几个。

- ◆ 老师能够看到你自己看不到的地方。人这一辈子，很少人能给自己一个清晰的评价和认知，要么高估自己，要么低估自己，而旁观者，尤其是老师，往往能够看到你的弱点、长处、威胁、变化，并给你适时的提醒和指导，少走弯路。
- ◆ 所有领域的知识都是成体系的，如果有这个领域的行家里手在你早期的学习阶段进行指导甚至设计练习技巧，与自己琢磨的效果是不可同日而语的。估计每个人都会有这样的经历，一个问题自己想到心碎想到梦醒也没有结果，别人过来抽丝剥茧条理清晰地一讲，不仅你懂了，连你的小伙伴都懂了。这就是听君一席话胜读十年书的道理。
- ◆ 好处多多，余不一一。

但是走出校门之后再想找传统意义的师傅就很难了，像绝地武士那样和 Master 出双入对同生共死更无可能，这时你就需要把身边的朋友、同事当做老师和资源，不耻下问，而且要问得有智慧，让人有回答欲望，那么如何提问呢，请参考前面的“如何提问”一文。

参与社区和技术会议

自己学习和同事交流之余，可以参与一些网络社区的交流，推荐以下两个。

- ◆ 技术问答社区：<http://stackoverflow.com>，在技术领域几乎包括万象，无所不知。
- ◆ GitHub：<https://github.com>，几乎全世界优秀的开源软件作品都在上面。

另外还可以参与一些群组，订阅一些优秀的个人博客，这个时代依然有人愿意贡献优质内容。

选择性参与一些技术会议，比如 QCon，不指望在会场能学到什么，但可以了解技术趋势，并看看别人在做什么。

刻意练习

之前写过两篇“刻意练习”（<http://macshuo.com/?tag=刻意练习>）的文章，自感对学习编程有一定帮助，大家可以去读一下。

逃离舒适区

这一部分适合已经有一定编程基础的同学。

什么是舒适区？如果你是个新手，你就没什么舒适区，什么都不懂就没什么舒适可言，在磕磕绊绊的学习中懵懂前行，期间可能还伴随着老鸟的嘲笑和进度的压力。终于有一天你武功精进，乾坤大挪移练到了第五重，工作中开始得心应手，游刃有余，不断有新人或老人来找你解决问题，你微笑着迎接挑战，淡淡地送走难题。你挥一挥手，不带走一片云彩，这是什么境界？这就是你的舒适区，这和靠在沙发上看电视的舒适不是一回事，通常进入舒适区需要花费你很多的时间和精力，需要你不断地练习，一旦进入，你会 enjoy it!

这时候，如果有人胆敢让你脱离舒适区，可算要了亲命了，你会勃然大怒，轻则争吵，重则离职。这种事遇到太多了，一个写前端的你让他学习一些后端技术，一个写 Java 的你让他学习一下 C，得到的答复可能会是：“Sorry, I feel very uncomfortable!”

没有人学新东西的时候非常舒服，一旦经历过从新人到老鸟的过程，再让你

进入陌生的领域，那种痛苦会让你自发的去抗拒。但是一个人不可能永远躲在舒适区里，逃离舒适区会有助于你从不同的角度看问题，视野会更加开阔。人总要往前走的。

程序员与编程

如何学习一门
编程语言

134

很多人在某个地方待久了就会非常懈怠，没退休就像在养老，这时候你就知道，他们在舒适区太久了，与在哪个地方无关。

最后一招 “见龙在田”

实战总是很重要，为大家推荐一个在线学习编程网站：<http://www.codecademy.com>。

假以时日，各位必定武功大成，那时横刀立马、拔剑四顾，说英雄谁是英雄！

科技与人文

147

不要做一个 Hater

科技与人文

MacTalk 开通以后，我收到过很多寻求建议的问题，也尝试回复过一些，还有一些是我没有能力回复的。人生一路走来我们会寻求很多建议，也有很多人给你忠告。需要警惕的是，这里面有相当一部分人的“忠告”总是负面的。比如你想去学编程，他说你的逻辑能力不适合编程；你说要做销售，他说性格决定命运，你的性格做不了销售；你说我要去创业，他说这个项目类型没人会投资的，早做早死，晚做晚死；你说我要站着把钱挣了，他说这是在中国……当你稍微遭遇了一点失败的时候，这些人就会祭出万试万灵的杀手锏：你看，我早就说过……

我们把这样的人统称为 Hater，这种人对自己不了解或没有勇气尝试的事物永远持否定态度。如果你发现一个人大部分时间在否定着什么，那么他们的意见不听也罢，甚至于那些鼓励的建议也仅仅是建议而已，仅供参考，因为最终不是那些提建议的人去做事和承担后果。做任何事情都是我们自己的选择，想清楚了也好，没想清楚也罢，想去做的话，尽可能鼓励自己去做。做事的人总是让人敬佩的，而且由于我们在做事，所以总会遇到失败，这时候那些口诵大悲咒“我早就说过”的 Hater 是完全可以忽视的，因为所有人都是在试错中成长的，那些不犯错的人充满了各种幻觉，其实是因为他不再成长了。

李笑来在《把时间当作朋友》一书中写道：

他们一定要给你泼冷水的。泼冷水的愿望之强烈，你无法想象。那种强烈借助了太多的力量：怀疑、嫉妒、恐惧、愤怒。而在表现的过程中却又包装上另外一层表皮：关怀、爱护、友爱、帮助。

当然李笑来没有把“他们”定义为 Hater，但我想应该是一个意思。所以我们首先不要自己成为 Hater，另外也没必要去听取 Hater 的忠告。地球也不会因为 Hater 的存在而停止转动！

当老罗的锤子手机发布的时候，我仿佛是看到了一群 Hater 冲上去一顿狂贬，

不要做一个
Hater

149



当然里面也包含了一些看似有道理的贬，但是这些东西有什么力量呢？老罗不也说了么，你们的感受我根本不在乎。我不认为锤子有一天能够砸烂苹果，但锤子会有自己要砸的东西……

科技与人文

当时就有写这一篇的冲动，不过一直放到今天才写，是为记：Don't be a Hater！

不要做一个
Hater

150

写作与编程

科技与人文

最近断断续续读了很多技术之外的书籍，包括钟阿城、王小波、王朔、刘震云的作品，还有那些老翻译家们的译作：王道乾译的《情人》、查良铮译的《青铜骑士》、黄锦炎译的《百年孤独》等。看着这些优秀的文学作品，顶尖的文字，和《Unix 高级环境编程》、《数学之美》、《HTTP 权威指南》这样的“技术干货”静静地躺在那里，我时常感觉到困惑：编程诚可贵，码字价不高，若为梦想故，两者都不能抛！

Paul Graham 在《黑客与画家》中描述了黑客与画家的相同之处，认为计算机和绘画看似截然不同：计算机冰冷、精确，而且井然有序；绘画则狂热，充满畅想和欲望。但黑客与画家却有很多共同之处：他们都是创作者，本质上不做研究，而是试图做出优秀的作品，当然过程中可能会发明出一些新的艺术形式或编程语言。

在阅读这么多文字和代码之后，我逐渐领会到 Graham 想表达的一些东西，当然不是万物滋长、天人合一的奥妙，而是文字和代码的相通之处：它们都在表达这个世界上需要表达的东西。文字仅仅能看是不够的，要能听，读来朗朗上口，有韵律有节奏，才是好文字。

代码只能运行也是不行的，要能读能改，上可处理异常，下能控制事务，设计精妙，逻辑严谨，才是好代码。

程序员常常感慨，技术领域广若高山大川，常走常新。问世间“是否此山最高？”答曰：“在世间自有山比此山更高（《射雕英雄传》的主题曲，70 后感受下）。”听起来让人无比寂寥。但和人类的文字比起来，编程就是蹒跚学步的小朋友。如果说编程之路上的那些坑让我们叫苦不迭，鼻青脸肿，那文字路途上的陷阱就是黑洞，你趴在洞口呼喊都没有回声，扔个石子也听不见响动，多少文学中青年因为缺乏才情在这条路上坠入黑洞郁郁而终，即使那些极具天份的文字工作者，也可能由于环境、政局、时代而错过写出最好作品的机会，想起来就让人扼腕叹息。

写作与编程

189



王小波在《我的师承》一文中，高度推崇老翻译家王道乾、查良铮和他们的作品。这二人为少年天才，在壮年时逢那个疯狂的年代，满腹才情无处抒写。

王道乾早年留学法国，在巴黎索邦大学文学院攻读法国文学，回国后翻译了很多法国文学作品，晚年因为翻译法国女作家杜拉斯的《情人》蜚声海内外，在汉语世界里创造了“另一个玛格丽特·杜拉斯”，王小波评其“文字功夫炉火纯青。他一生坎坷，晚年的译笔沉痛之极，无限沧桑尽在其中”。

查良铮是金庸的叔伯兄弟，同属良字辈，笔名穆旦，著名诗人，翻译家。早年创作了《探险者》、《穆旦诗集（1939—1945）》、《旗》等作品，20世纪50年代末期被调往图书馆和洗澡堂工作，先后十多年受到管制劳改，停止诗歌创作，开始翻译。比较有影响的译著有俄国普希金的作品《波尔塔瓦》、《青铜骑士》，英国雪莱的《云雀》、《雪莱抒情诗选》，英国拜伦的《唐璜》等。20世纪70年代后期恢复诗歌创作，一举创作了《智慧之歌》、《停电之后》、《冬》等近30首作品。临终前在《冥想》的诗中写出了自己的内心独白：

而如今突然面对坟墓，我冷眼向过去稍稍四顾，只见它曲折灌溉的悲喜，都消失在一片亘古的荒漠。这才知道我全部的努力不过完成了普通生活。

可谓黄钟大吕，大音希声。

王小波在文中对这两位先生的评价极高，谓之影响了其一生写作风格的导师：

我觉得我们国家的文学次序是彻底颠倒了的：末流的作品有一流的名声，一流的作品却默默无闻。最让人痛心的是，最好的作品并没有写出来。这些作品理应由查良铮先生、王道乾先生在壮年时写出来的，现在成了巴比伦的空中花园了……以他们二位年轻时的抱负，晚年的余晖，在中年时如有现在的环境，写不出好作品是不可能的……

回顾这些先贤的作品，让我们知道，生活中并不是只有油盐酱醋和升官发财，人类在历史的长河中经历了各种苦难和悲情，但是总有那么一些人，不屈不挠地去创造那些美好的事物，可能大部分人一生都不会注意到这些闪烁着光芒的珍宝，但是它们就那样存在着，总会有人看到的……

本来是想捡起“Linux文件系统的权限王国”系列写写的，不过今天颁了诺贝尔文学奖，那就写点文字的事儿，算个念想。今年的获奖者是82岁的加

拿大女作家爱丽丝·门罗，82岁了！同学们，所有想步入文学殿堂的，大家感受一下。感受完了就可以去编码了……

今日北京大风，一扫多日雾霾，下班时走在园区的石子路上，抬头远远望开，夜色清亮，拍一张作为今日题图，是为记。

科技与人文

写作与编程

191

重读黑客与画家

科技与人文

《黑客与画家》这本书的中译本出版于2011年4月，它的作者是美国互联网界“创业教父”，哈佛大学计算机博士Paul Graham，它的译者是著名blogger 阮一峰先生。

这本书在2011年一上市就受到了广大人民群众的爱戴。我在第一时间拿到纸质书后，通读了两遍，当时感觉很震撼，可以说本书是我近年来读过的最优秀的人文类技术图书，个人非常喜欢。所以在去年在图灵推出多看电子书后，又购买了电子版《黑客与画家》，放到手机和Pad上随翻随看，最近又开始重读。

好书的特点是常读常新，一本好书往往沉淀了作者几十年的岁月，绝不是你草草翻上一遍就可以理解和掌握的。虽然是同样的文字，在不同的时间和环境阅读，往往给你带来不同的启示和感悟。下面是我重读的读书笔记。

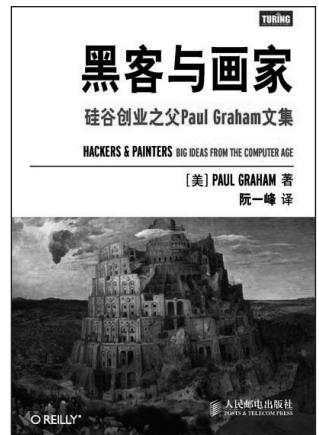
Redesign——设计永无止歇

Paul Graham在《黑客与画家》的第14章“梦寐以求的编程语言里”，写了一节关于“再设计”（Redesign）的随笔。什么是好的文字？好的文字来自于不停的修改，好的编程语言和软件产品同样如此，在个人的工作生涯里，我的体会是，再多的修改也不过分。可以说没什么软件产品是完美的，完美主义者都是不断打磨产品以趋近完美。如果不信，那么各位看官可以打开你们一年前写的代码或文章，如果脸红的话就吱一声吧。

同样，如果你想不断地调整自己的设计和实现，那你就需要保证你的工作在某个特定阶段是可持续的。我的建议是无论选择公司，还是在公司内部选择工作，尽可能选择能够长期投入和完善的事情做，如果你半年做一个项目，之后又换成另一个，然后再换，除了积累了一大堆项目经验之外，你个人能力没有得到提升，你永远无法完成一个优秀的产品。在产品公司，你可以为优化某个算法或Ajax效果花费一个月的时间，这在以单纯做项目或外包的公司是不可想象的。找一家程序员被当做天才和宝贝的公司，去做可持续的产品……

207

重读黑客与画家



推荐两段核心文字，大家体会下阮一峰老师的翻译功底：

为了写出优秀软件，你必须同时具备两种互相冲突的信念。一方面，你要像初生牛犊一样，对自己的能力信心万丈；另一方面，你又要像历经沧桑的老人一样，对自己的能力抱着怀疑态度。在你的大脑中，有一个声音说“千难万险只等闲”，还有一个声音却说“早岁哪知世事艰”。

原文：

To write good software you must simultaneously keep two opposing ideas in your head. You need the young hacker's naive faith in his abilities, and at the same time the veteran's skepticism. You have to be able to think *how hard can it be?* with one half of your brain while thinking *it will never work* with the other.)

如果你能平衡好希望和担忧，它们就会推动项目前进，就像自行车在保持平衡中前进一样。在创新活动的第一阶段，你不知疲倦地猛攻某个难题，自信一定能够解决它。到了第二阶段，你在清晨的寒风中看到自己已经完成的部分，清楚地意识到存在各种各样的缺陷。此时，只要你对自己的怀疑没有超过你对自己的信心，就能够坦然接受这个半成品，心想不管多难我还是可以把剩下的部分做完。

原文：

If you can keep hope and worry balanced, they will drive a project forward the sameway your two legs drive a bicycle forward. In the first phase of the two-cycle innovation engine, you work furiously on some problem, inspired by your confidence that you'll be able to solve it. In the second phase, you look at what you've done in the cold light of morning, and see all its flaws very clearly. But as long as your critical spirit doesn't outweigh your hope, you'll be able to look at your admittedly incomplete system and think, how hard can it be to get the rest of the way?

时间与流行

一种编程语言要想变得流行，最后一关就是要经受住时间的考验。

——《黑客与画家》

时间有时候是很无情的，很多流行的东西，随着时间的流逝将变得面目全非。美貌与时尚如此，技术同样要经历岁月的洗礼。进入 21 世纪以来，技术热点不断变更，每次的技术更迭，就像流水冲刷河床一样，虚无地流走，沉淀下来的是有价值的东西。我们要做的就是那些能沉淀下来的东西，这需要去判断和选择。

进入 21 世纪，从静态语言到动态语言、面向对象编程到函数式编程、从模型驱动设计到领域驱动设计、从 SOA 到云计算、从 BI 到大数据、从 BS 到移动互联，无论技术热点如何变化，站着挣钱的永远是那些踏踏实实做产品和技术的。2009 年 SOA 火热的时候，每家软件公司和互联网公司都号称自己面向服务了，甚至 IBM、BEA 等公司为 SOA 确定了 SCA 和 SDO 规范，但 3 年以后，无人再提 SOA，无非就是用开放的技术实现原来的 EAI 么……

有位在校大学生问：“我们下学期要学习数据库，以后想从事大数据方向，不知道从什么方面学习大数据”。我的建议是，不要被现在大数据的噱头忽悠了，等你两年后毕业出来，可能已经没人提大数据这回事了。如果真的对数据感兴趣，踏踏实实把关系数据库学好，有时间的话再学一门编程语言，掌握数据结构，再有余力学习一些数据挖掘和推荐算法等知识，这就行了，相信我，你没那么多时间！

所以，正在流行的东西并不一定值得投入，流行感冒倒是流行，也没人追啊……

最大化你的价值

有人问，你为什么要从事 IT 技术研发工作？如果是乔布斯，可能的答案是改变世界；如果是人生导师，可能的答案是跟随你心。如果是我回答呢，答案就是：如果不从事这个行业的话呢，我还真不知道该如何养家糊口。哈！

每个人进入一个行业，有必然性，也有偶然性。现在想来，我进入软件行业可能是兴趣使然。我小时候就比较喜欢与电子相关的东西，这个习惯保持到现在就是喜欢电子设备。大学的时候电子和计算机相关的课程都学得很好，其他专业课则一塌糊涂。毕业后进入一个工厂做电子设备测试，晚上则用车间的电脑学习编程，很快自己觉得可以找到工作了，就去面试，一面之下进入洪恩，稀里糊涂算是进入了 IT 行业。

工作了十几年，发现自己确实对技术比较感兴趣，曾经有很多机会转岗成为纯粹的管理和业务人员，但最终都没舍得放弃技术，当然技术也没什么大成，爱好而已。

为什么做技术，技术是什么？我的观点是：

与其他工种一样，技术可以谋生。很多导师说，看准一件事情就全情投入，不要考虑收入，奋斗不息，财富会随之而来。且不说只有你成事了，财富才能来，就说没成事的时候，我们总不能饿得头昏眼花去奋斗吧。准确一点说，是工作初期不要过分考虑金钱。总要解决温饱问题吧，那么做技术研发可以很容易达到这个目标。

技术是一种手段和做事方式。尤其是在现在这样一个互联网和数据的时代，可以说技术面前人人平等。你付出了多少，差不多就会得到多少。很多人羡慕创业公司的人获得的财富，他们只不过是把你20年平稳的打工生涯压缩成4年艰苦卓绝的创业，当他们冒着成为炮灰的风险在清晨的寒风里编程时，你正在温暖的被窝里做着美梦。所以就别羡慕了，那是他们应得的。

做技术需要终身学习，如果你是个学习狂，那么恭喜你找到了一份完美的工作。有一次，一个工程师告诉我，每次感到恐慌的时候，他就开始学习，掌握了一门又一门语言和技术。最后他成为了一个通才之上的专才。书到用时方恨少，事非经过不知难，有时间就学点东西，没坏处，还能预防老年痴呆。

做技术可以最大化你的价值。如果你是卖煎饼的，卖一个是一个。如果你开发了一个千万人使用的软件，那你做这个软件的价值就放大了千万倍。如果你做的互联网产品服务了千万个用户，你做的这个产品的价值也就放大了千万倍。如果你在做这样的工作，那么你就最大化了自己的价值，财富也会随之而来。如果没有，就去找这样的工作。

难易相成

古人云：有无相生，难易相成，长短相形，高下相倾。

很早就想说说困难和容易这点事。人生一世，我们到底是要选择做容易的事，还是做难事？这个想法源于春节前的业务讨论会，有两位经理各执一词，一个说，为了让部门能够生存下去，我尽可能去做那些简单的相对稳定的任务，因为这样可以保证良好的现金流。另一个说，为了能让部门生存下去，我尽可能去做那些困难的利润更高的任务，因为这样可以保证良好的现金流。

难易相成，困难和容易会在不同的环境下相互转化，我觉得这两位经理在特定场景下说得都没错。

就个人而言，我觉得我们应该尽可能做那些困难的事情，让别人变得不那么困难。无论是做软件还是做互联网服务，其实终极意义就是：你做出来的东西能否解决用户的问题。如果是一个容易解决的问题，那么很多人早已经解决了；如果是一个困难的问题，那就意味着很多坑等着要埋你。咋选呢，似乎怎么选都是炮灰，权衡一下，选择前者基本是无用功，那我们只好选择后者，我们本身也是爱挖坑的人，况且困难面前不是人人平等么。

JetBrains 是一家捷克的软件公司，该公司做的事情就是为 Java、Objective-C、Python、Ruby、JS 等语言写开发 IDE。给程序员写工具可不是闹着玩的，他们对 IDE 的挑剔基本比肩女性对化妆品，但 JetBrains 开发出来的工具深受程序员喜爱，各种智能，各种效率，JetBrains 获得了极大的成功。为啥？因为做着别人很难做成的事。

让自己困难点，让别人容易点，说了半天，好像就这么点事。Paul 在《黑客与画家》里提到选择哪种技术和语言去实现软件的时候，同样选择了那些困难的有竞争力的事情去做。

创造财富

《黑客与画家》里有两章是描述财富的，如果你是个财迷，那么就该去读读，如果你不是，读读我的读后感就行了。

财富和钱从来就不是一码事，所以大部分人只能去创造财富，因为你不是印钞机。但钱毕竟是流动的财富，所以大部分人还得通过创造财富去挣钱，那么一辈子挣多少钱合适呢？

在很多年以前，100 万还能在北京买一套房子，大家还没有听说过 PM2.5 这个术语，蓝天还不是那么稀有。有人写过一篇文章，大意就是一个家庭要多少钱才能正常生活（什么是正常？好吧，你就当及格线 60 分理解），作者从购房、购车、赡养父母、教育子女、家庭生活、休闲娱乐、养老、货币贬值、物价飞涨等各个方面算了一笔账，这个数字是 600 万。

这位同学你不用站起来了，我知道现在 100 万在北京只能买个洗手间，我说的重点不是这个。而且要算也很简单么，如果你觉得现在北京买套房需要 400 万，那就加上 300，如果在老家 50 万就能买，减去 50 即可，不离谱。

如何去挣这个抽象的 600 万呢？如果按照 60 岁退休计算，大部分人要工

作 40 年左右，大家可以计算下，如果妥妥地工作了 40 年，你的平均工资要达到多少才能过正常生活。当然你也可以选择创业，高风险高回报，一旦创业成功，你就摆脱了这个“正常生活”的羁绊，可以有更多的时间和空间做自己更喜欢的事。但是这些成功的创业者不会就此不再工作了，他们可能会比普通打工者工作更长的时间。

为什么呢？很多有钱人完全不必再工作，但是他们工作起来比普通人还欢实。不是因为他们天生就是受虐狂，也不是因为社会压力，而是因为无所事事使人感到孤独和消沉。我有个朋友移民国外，由于有房产完全不必工作，天天烧烤钓鱼，半年以后，他在 MSN 上告诉我“看到烧烤钓鱼就想吐，我必须要找个工作了”。

另外，如果你不是银行劫匪或公务员，你应该知道财富是创造出来的，而不是抢来的。乔布斯和沃兹创建了苹果公司，为自己、员工、开发者和社会带来了巨大财富，但别人并未因此而变得贫穷。所以建议大家尽可能去做创造财富的工作，而不是掠夺财富。

另外，如果没人给你报酬，你是否愿意去工作呢？唯一可能的就是这件事比较有趣，比如 Linus Benedict Torvalds 免费写了著名的操作系统 Linux，他当时可没想着用这个操作系统赚钱。那我为什么写博客呢？嗯，思考中……

关于这个话题，Paul 表现出了很多技术之外的 Political 智慧，比如允许赚到大钱的人保留自己的财富，比如藏富于民：

一旦自己的财产有了保证，那些想致富的人就会愿意去创造财富，而不是去偷窃。由此导致的新技术不仅被转化成财富，还被转化成军事力量。

要鼓励大家去创业。只要懂得藏富于民，国家就会变得强大。让书呆子保住他们的血汗钱，你就会无敌于天下。

完结

《黑客与画家》可以为你带来技术、生活、自由、财富等各方面的思考，确实是难得一见的技术图书。事实上，这次重读还有很多感受没有形成文字，希望以后有机会还能继续补充这篇文章。

人物

213

敬畏之心

人物

敬畏之心

222

最近在读两个人的著作，一个是冯唐，一个是吴军。

喜爱文学的人应该对冯唐不陌生，当年韩方大战时，冯唐及时抛出文学“金线论”，后被戏称为“冯金线”。古诗有云：“冯唐易老，李广难封”，冯唐当然是他的笔名，此人真名“张海鹏”，似乎比 Mac 君的名字还要俗气一些。

冯唐是协和医院妇科博士，前麦肯锡合伙人，作家。我最初知道冯唐的时候，被他的三重身份惊呆了，这得多分裂才能把这三个职业捏合到一个人身上？比无间道牛，比韩寒的赛车手和作家身法高端大气。



冯唐的文字有幼功，初中的时候熟读大量的古文和外语著作，所以他的文字非常体现功力，能少用一词，绝不多填一字，文字充满速度而不凝滞，厚重而有质感。文学的幼功咱不懂，围棋的幼功我是知道的，我哥哥半路出家学习围棋，打败同时代学棋小伙伴无敌手，但是经常被那些三四岁开始摸棋的小屁孩打得落花流水而无还手之力，我问何解？我哥说这是幼功！可见一斑。

冯唐写过很多书，我最近在读的是《三十六大》，是他今年的杂文集。以冯唐的才气，写杂文当属大理石压咸菜缸——大材小用，但是这些文字依然能够体现他的风格和特点，我非常喜欢，摘录几则供大家欣赏。

个人和全体古人的关系，应该是昆仑山上一棵草和昆仑山的关系。在长出草之前，需要先爬昆仑山。如果不明白什么叫高山仰止，先别说“俱往矣”，先背三百首唐诗。知道昆仑山有高度之后，开始爬吧，学杜甫学到风雨掀翻你家屋顶，学李白学到梦里仙人摸你头顶，学李商隐学到你听到锦瑟的一刹那裤裆里铁硬。学到神似之后，是血战古人，当你感觉到不是自己像杜甫、李白、李商隐，而是杜甫、李白、李商隐像自己，就是到了昆仑山顶。是时候长自己的草了，不是杜甫的草，不是李白的草，是自己的草。这个时候，

长一寸，也是把昆仑山增高一寸，也比自己在平地蹦跶一米，高万丈，强百倍。

关于现场，你说：“笔与墨会，是为氤氲，氤氲不分，是为混沌……不可雕凿，不可板腐，不可沉泥，不可牵连，不可脱节，不可无理。在于墨海中立定精神，笔锋下决出生活，尺幅上换去毛骨，混沌里放出光明。纵使笔不笔，墨不墨，画不画，自有我在……人写树叶苔色，有深墨浓墨，成分子、个字、一字、品字、么字，以至攒三聚五，梧叶、松叶、柏叶、柳叶等垂头、斜头诸叶，而形容树木山色、风神态度。吾则不然。点有风雪雨晴四时得宜点，有反正阴阳衬贴点，有夹水夹墨一气混杂点，有含苞藻丝缨络连牵点，有空空阔干燥没味点，有有墨无墨飞白如烟点，有如胶似漆邋遢透明点。更有两点，未肯向学人道破，有没天没地当头劈面点，有千岩万壑明净无一点。噫！法无定相，气概成章耳。”

现场有神。

无论我是钢笔在纸面上书写还是手指在键盘上敲打，我知道，字句的黑白疏密凸凹之间，有小鱼和小雀在。

两千多年前，人平均寿命不到五十，孔丘说，一个人到了四十，知道了自己能力的边界，能做什么和不能做什么，于是不惑。两千多年后，人平均寿命超过七十，孔丘说的依旧适用，这个老怪物。这几天冬去春来，换季节，睡得不安稳，昨夜醒来，看到你就倚在窗台边抽烟，生命就像一头驴一样蹲在你旁边，因为彼此熟悉、天人相知，驴血已经不滋滋作响。一时，我想，我想骑就骑，要下就下，打打小鸟、看看小星、码码小说，向死骑去而不知死之将至，一切挺好。

说了冯唐，有些读者表示不喜欢冯唐，有的说看不懂，有的说 Mac 君你怎么能喜欢冯唐呢？声明一下，Mac 君不喜欢冯唐，俺是异性恋，只是喜欢冯唐的一些文章和文字罢了。另外，写冯唐并不是推荐大家去买或去读冯唐的书，只是主观表达，文学这东西，确是仁者见仁，志者见志，萝卜白菜各有所爱，大家只管去读自己喜欢的书就好了，多读书，是好事。

下面说说吴军博士和他的书。吴军应该算是科学家范畴的，毕业于清华大学和约翰·霍普金斯大学，获博士学位，主要研究领域是语音识别和统计语言模型，和李开复老师的早年研究方向类似，当然李老师……咳咳，已经不做技术和研究很多年了，但吴军博士依然是冲在第一线的，他先后加入 Google

和腾讯，都是负责搜索领域的研究和研发工作。

吴军目前出了两本书，分别是《浪潮之巅》和《数学之美》，原稿都来自他在 Google 黑板报的系列文章。很多人知道吴军可能是因为《浪潮之巅》，那本书我当然也读过，目前在读的是《数学之美》。

我以前说过，很多人技术做得很广很深，但是文字能力欠缺，写出东西来让读者痛苦不堪；有的文字功底深厚，但技术不行，著作虽然读来容易，但无法深入浅出，读者一旦醒悟过来，后果同样很严重。二者兼而有之的人则少之又少，所以优秀的技术图书真是凤毛麟角。吴军在技术和文字层面都属上上之选。

《浪潮之巅》出版之后赞誉多多，我读完以后感觉比那本《硅谷之光》好看，有些人觉得有不符史实之处，但我觉得吴军凭一己之力纵论 IT 科技和公司发展史，揭示规律，品评趋势，书写科技公司的人、事、势，同时辅以自己的思考和论证，确属难能可贵，行文流畅，读来收获多多。

今天主要说说我正在读的《数学之美》。小时候念书时，数学是我的强科，在几门学科里处领先之列，但工作以后除了记得几个算法和编程所需的一些数学知识之外，其他大部分都还给老师了，想想老师也不易，那么多同学那么多知识还回来，得有多沉多重。知识虽然还了，但兴趣还在，常读一些数学相关的著作，这本《数学之美》很早就买了，但近期才拿起来看，说来也是惭愧得紧。

数学是众多自然科学、社会科学、管理科学的基础，誉为是科技之根毫不为过。《从一到无穷大》一书在第一章“大数”里讲了个故事，说两个贵族决定做数数游戏，谁说出的数大谁赢。

一个贵族绞尽脑汁想到了一个大数：“3”（请以 shan 平音读出并感受下……）。

另一个贵族傻眼了，想了想说：“你赢啦！”

这可能只是一个故事，但如果这个故事发生在原始部族中，大概就完全可信了，说明原始社会物质匮乏导致无法产生大数。吴军在《数学之美》里也引述了这个故事，并由此开始讲述数学之美。整个书的基调是简洁、扎实、深入浅出又栩栩如生，于平凡之中见深邃，于无声处闻惊雷，读来有趣而又长知识。

对于从事 IT 技术的人来说，《数学之美》里的以下内容值得细细研读：

- ◆ 自然语言处理
 - ◆ 统计语言模型
 - ◆ 中文分词
 - ◆ 搜索引擎的索引
 - ◆ 图论
 - ◆ 网络爬虫
 - ◆ PageRank 网页排名
 - ◆ 地图和本地搜索
 - ◆ 拼音输入法的数学原理
 - ◆ 逻辑回归和搜索广告
-

人物

敬畏之心

225

这本书读起来是比较费力的，不跟看网络小说那样一个晚上翻 250 页，但费力之事自有其好处，人生总要做一些艰难费力的事情，读书也是一样。

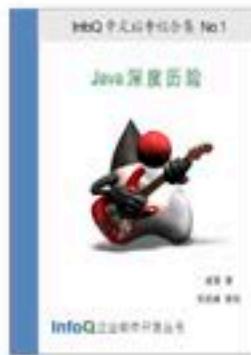
写到这儿很多读者要问了，这和敬畏之心有啥关系？这是近期 Mac 君在读这些书的时候产生的一些感受。很多读者看到我写了一些文和事，经常发信息说好，说赞，但我自己经常处于一种惶恐的状态，比如读书的时候，你就会知道自己文不如冯唐、术不及吴军，其实用不如是不妥的，实在是难以望其项背。一个人学了十年再工作十年，以为学了一点文和术，但你了解越多，越会发现自己仅仅是大海之一滴水、沙漠之一粒石，永远有无数优秀和卓越的人在你前方奔跑，你只能看到他们奔跑中的一缕尘埃！

但在现实中，却是敬畏之心不常有，常起轻视之意。所谓文人相轻，自古如此，大家执笔对骂，恨不得在对方身上戳几个窟窿。程序员也是一个德行，看见别人的代码要么是 Garbage，要么就 Refactoring，缺乏起码的尊重和敬畏。总之文章和代码都是自己的好，LD 都是别人的好（LD 可做多重含义理解）。

以前说过，人的一生要么就低估自己，要么就高估自己，想要做到既不妄自尊大，也不妄自菲薄，实在是太难了！我们只能保持敬畏之心，保持孤独之意，一路向前！

InfoQ软件开发丛书

欢迎免费下载



商务合作: sales@cn.infoq.com

读者反馈/内容提供: editors@cn.infoq.com



软件
正在改变世界!

InfoQ 促进软件开发领域知识与创新的传播

软件正在改变世界！InfoQ是一个在线新闻/社区网站，旨在通过促进软件开发领域知识与创新的传播，为软件开发者提供帮助。为达到这个目的，InfoQ基于实践者驱动的社区模式建立平台，提供新闻、文章、视频演讲和采访等资讯服务，所有的这一切也都是为了研发团队中那些有创新精神的人群：团队领导者、架构师、项目经理、工程总监和高级软件开发人员等。

InfoQ目前在全球有四种语言版本，分别是英文、中文、日文、葡文。

另外，InfoQ在伦敦、北京、东京、纽约、圣保罗、上海、杭州等城市举办过QCon全球软件开发大会。

InfoQ中文站将和InfoQ全球网站一样，秉承“扎根社区、服务社区、引领社区”的经营理念，与中国技术社区的专家一起，为中国软件企业和个人提供及时、高质量的技术资讯，成为连接中国企业软件技术高端社区与国际主流技术社区的桥梁。