TURING

8

2013/10/18

码农

codeMaker()...的面试

微软、亚马逊、谷歌、苹果、
Facebook、雅虎的面试迷局

金领简历的六大特点

数独问题

云风：一个编程的自由人

离岸找工作

《程序员你伤不起》编辑的话

目　录

鲜阅

八++

出版的未来

书榜

成书手记

编者的话

成为更好的自己

编者 / 李盼

码农们普遍情商偏低，和人类交互总是不如和计算机交互来得驾轻就熟。可是工作和合作仍然是每个码农必须要面对的课题。如果你想为更优秀的团队工作，或成为更靠谱项目的一部分，就必须冲破自己的怯懦。《码农》第2期人物陈皓曾经分享过他的个人求职经历，他曾经在面试中“整个过程都低着头，不敢看别人一眼”。为了克服自己的弱点，耗子哥变身面试狂人，“每周都要去面试”。这么做的意义，不只是为了找到更好的工作，拿到更高的薪水，更是要成为更好的自己。

本期《码农》为你戳破互联网巨头公司的面试迷局，帮你从团团迷雾的表象中突出重围。让Stack Overflow的创始人告诉你，他眼中优秀的程序员是什么样子。也请纽约大学的数学教授和你一起做一次逻辑思维训练，解读数独难题。《人人都有好工作》的作者还会教你如何应对面试中遇到的棘手问题，避免那些“过于”坦诚的答案。当然，我们并不希望你变得口若悬河，只是不想面试官低估你的能力。

当有人为了找到好工作而忙碌时，也有人为了追寻自己的梦想和兴趣而努力。云风从小就喜欢游戏和编程，三十年后，他仍然为了游戏而编程，为了乐趣而游戏。他在网易十年，临别时丁磊为他留下了网易永久的一席之地。他为乐趣而编程，不为

健身而攀岩。少些目的，多些过程，也许人生本该如此。

成为更好的自己，享受更美的过程。别让恐惧挡住你前进的方向，让挑战变成生活的游戏吧！

专题：码农的面试

揭开微软、亚马逊、谷歌、苹果、Facebook和雅虎的“面试迷局”

好了，不用再挖空心思，再三思索了，我来告诉你。

在本文中，我们邀请了来自顶尖科技公司（微软、亚马逊、谷歌、苹果、Facebook及雅虎）的面试专家来为大家答疑解惑，揭秘面试中的那些事儿。

这些专家会让我们了解各家公司的面试流程，帮助还原那些发生在面试会议室之外的事情，以及面试结束后的事项。

这些专家还会告诉我们各家公司面试流程的不同之处。比如，亚马逊的“调杆员”[1]是怎么回事，谷歌的招聘委员会是如何运作的。是的，每家公司各具特色。了解这些“怪癖”会让你更加胸有成竹，不会被突如其来的亚马逊“调杆员”给吓住，也不会对苹果居然同时派出两位面试官来考察你而感到意外。

此外，这些专家也强调了各家公司的面试重点。尽管这些顶尖公司都喜欢考察求职者的编码能力和算法基础，他们其实也各有侧重。不管这是源自各家公司的技术背景或是历史，至少你知道该如何做好准备。

接下来，让我们一起揭开微软、亚马逊、谷歌、苹果、Facebook和雅虎的“面试迷局”吧。

[1] “bar raiser”（调杆员）的概念来自亚马逊美国总部。这个词原指在跳高比赛中，一次次将杆调高的工作人员。而亚马逊的调杆员则是一群在招聘过程中负责从企业文化以及行为准则的角度考察应聘者，从而维护招聘质量的人。在招聘中，调杆员会用很苛刻的眼光考察应聘者是否在至少一点上高过亚马逊的平均水准，如果是，那么雇用这样的人实际上就等于在提升公司的能力，这就起到了“抬杆”的作用。

作者/ Gayle Laakmann McDowell
曾在微软、苹果与谷歌担任软件工程师。麦克道尔作为软件工程师在谷歌工作过三年，她还是该公司资深面试官及招聘委员会成员。她在美国本土与海外面试过150位求职者，为招聘委员会评估过1000多份求职者的应聘材料，还审查了数以百计的简历。除了作为面试官的丰富经验外，她先后面试并收到了来自12家高科技公司的录用邀约，其中包括微软、谷歌、亚马逊、IBM和苹果。2005年，麦克道尔创建了CareerCup.com网站，从面试官和求职者的角度，与人们分享她丰富的面

微软面试

微软喜欢招聪明人，尤其青睐计算机极客。求职者必须对技术满怀热情。微软的面试官不大会问你一些C++ API的个中细节，而是直接让你在白板上写代码。

参加面试时，求职者最好在早上约定时间之前赶到微软，先填好一些表格。接着你会和招聘助理碰面，他会给你一个面试样题。招聘助理主要是帮你热热身，不大会问技术问题；就算真的问了几个简单的技术问题，也是想让你放松心情，等到面试真正开始时，你就不会那么紧张了。

对招聘助理一定要以礼相待。说不定他们会帮上大忙，在你首轮面试表现欠佳时，他们有可能帮你争取重新面试的机会。夸张地说，他们甚至还能左右你的应聘结果。

面试当天你会接受4到5轮面试，面试官一般来自两个团队。许多公司会把面试安排在会议室，而微软的面试一般在面试官的办公室进行。你正好可以借机四处看看，感受一下他们的团队文化。

一轮面试过后，不同的团队，做法不一样，面试官可能会根据个人习惯决定是否将你的表现反馈给后续的面试。

试经验。CareerCup.com提供了一个庞大的面试资料库，其中包括出自数千家大公司的面试题。此外，该网站还建有论坛，供人们交流面试经验。

完成所有面试后，你有可能会见到招聘经理。假如真是这样的话，那可是好兆头，这意味着你通过了某个团队的基本考察。接下来，就要看招聘经理要不要录用你了。

快的话，面试当天你就会知道结果，慢的话，则可能要等上一周。要是等了一周还没收到人事部的通知，不妨发封邮件，客气地问一下进展。

如果你没有马上收到回应，有可能是因为招聘助理太忙了，这并不代表你就没戏。

必要准备事项：

你为什么想要加入微软？

提这个问题，微软是想了解你是否对技术满怀热情。一个比较好的答案是："自打接触计算机以来，我就一直在用微软的软件，贵公司开发的软件产品令人赞不绝口。比如，我最近一直在Visual Studio开发环境中学习游戏编程，它的API实在是太好用了。"注意这个答案是如何展示你对技术满怀热情的。

独特之处：

如果到了招聘经理这一关，说明你面试表现得不错。这可是个好兆头！

亚马逊面试

亚马逊的招聘流程一般从两轮电话面试开始，期间求职者会接受某个团队的面试。偶尔也会出现面试3轮甚至更多轮的情况，可能是有位面试官对你的评价不高，或是别的团队对你有兴趣。此外，还有其他特殊情况，比如求职者就在亚马逊总部所在地西雅图，或他以前面试过其他职位，也许一次电话面试就够了。

在电话面试中，面试你的工程师通常会要求你通过共享文档工具（如CollabEdit）写些简单的代码。他们问的技术问题可谓五花八门，意在探测你究竟熟悉哪些领域。

接下来，如有一两个团队根据你的简历和在电话面试中的表现相中你，你就要飞到西雅图接受4到5轮面试。在白板上写代码是少不了的，有些面试官会着重考察你的其他技能。每一轮面试官都会侧重不同的领域，所以他们的提问会大相径庭。在提交自己的评价报告之前，他们看不到其他面试官对你的评价，而且公司也不鼓励面试官在面试过程中互相交流，一切讨论都得等到几轮面试全部结束后。

顾名思义，“调杆员”主要负责把控面试质量。他们受过专门训练，并且是从其他团队抽调来的，以减少面试中的主观倾向。

在面试中，如果有位面试官风格迥异且要求格外严格，那他可能就是传说中的“调杆员”。这种人不仅面试经验丰富，而且跟招聘经理一样，拥有生杀大权。不过，切记：这一轮面试表现磕磕绊绊，并不等于你的整体表现就很差。面试官会比照其他求职者来评价你的水平，而不是只看你答对多少问题。

等到所有面试官提交评价报告后，他们会在一起讨论你的表现，并决定是否录用你。

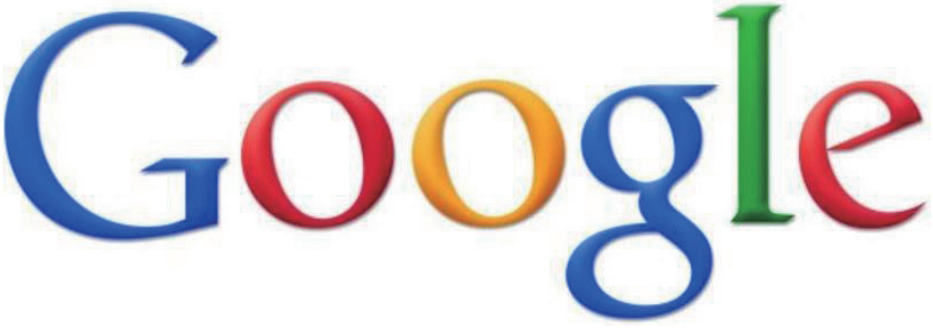
一般来说，亚马逊的招聘团队都会很快给出录用结果，很少有耽搁。要是一周内都没等到结果，建议你发封措辞得当的邮件询问进展。

必要准备事项：

亚马逊是一家互联网公司，这也意味着他们非常关注“扩展性”问题。请做好相应的准备。当然，回答这些问题，并不要求你具备分布式系统方面的知识。具体建议可参看“扩展性与存储限制”一节。
此外，亚马逊还会问很多面向对象设计的问题。请参看“面向对象设计”一节，里面有一些样题和建议。

独特之处：

“调杆员”来自其他团队，旨在提高面试标准。他和招聘经理一样重要，请尽量表现得出色一些。



谷歌面试

业界流传很多有关谷歌面试的可怕谣传，但多数也只是谣传。谷歌的面试与微软或亚马逊的并无太大区别。

谷歌的面试也从电话面试开始，来面试你的人是技术工程师，因此免不了会问些技术难题，求职者切不可掉以轻心。这些问题也可能涉及编程，有时你还要通过共享文档工具写些代码。电话面试的问题和现场面试的类似，要求也一样。

现场面试一般有4到6轮，其中一轮为午餐面试。面试官之间不能透露自己的评价报告，因此每一轮面试你都可以从零开始。午餐面试不会有评价报告，你可以借机问些其他环节不方便问的问题。

谷歌不会要求面试官侧重不同的领域，也没有所谓的标准流程或结构。每个面试官可以自行决定问哪些问题。

面试过后，评价报告会以书面形式提交给由工程师和经理组成的“招聘委员会”，由他们作出录用结论。面试评价报告由分析能力、编程水平、工作经验和沟通能力等四部分组成，最后你会得到总的评分，在1.0到4.0之间。“招聘委员会”里一般

不会有你的面试官。就算有，那也纯属巧合。

通常，在决定录用与否时，招聘委员会更看重那种有面试官给你打高分的情况，打个比方，如果你的得分是3.6、3.1、3.1和2.6，效果要好过拿4个3.1。

也就是说，每轮面试不一定都要有上佳表现。此外，你在电话面试中的表现一般起不了决定性作用。

如果招聘委员会给出的意见是“聘用”，你的材料就会转给“薪酬委员会”及“执行管理委员会”。最终结果可能要等上几周，因为还有不少流程要走，等待多个委员会审批。

必要准备事项：

作为一家互联网公司，谷歌非常看重如何设计可扩展的系统。因此，务必掌握“扩展性与存储限制”一节的问题。此外，谷歌的面试官很喜欢问些涉及“位操作”的问题，也请重点复习这些方面的知识。

独特之处：

面试官不是决策者。他们只提交评价意见供招聘委员会参考。招聘委员会给出录用与否的决定，当然，该决定偶尔也会被谷歌高管否决。

苹果面试

苹果的面试流程与公司本身的风格非常相符，是最没官僚味儿的。苹果的面试官很看重技术功底，但求职者对应聘职位和公司的热情也非常重要。虽然成为Mac用户并不是应聘苹果的先决条件，但你至少要对该系统有一定了解。

在苹果的面试流程中，招聘助理会先给你打电话了解一些基本情况，接下来团队成员会对你进行一连串的技术电话面试。

当你受邀去参加现场面试时，招聘助理会出面接待你，并介绍面试的大致流程。然后，你要接受招聘团队成员6到8轮的面试，其中这个团队的重要人物也会来面试你。

苹果的面试形式是一对一或二对一。请做好在白板上写代码的准备，交流的时候一定要把自己的思路表达清楚。你可能会跟未来的上司共进午餐，这看似随意，但其实也是一次面试。每个面试官都会侧重不同的领域，面试官之间一般不会过问彼此的面试情况，除非他们想让后续面试官就求职者某一方面多挖掘点内容。

当天所有面试结束后，面试官会在一起商议你的表现。如果大家都认为你表现不错，接下来会由你应聘部门的主管或副

总来面试你。能见到主管也不见得你一定会被录用，不过总归是个好兆头。让不让你见主管的决定对你是不公开的，如果你落选了，他们只是默默送你离开公司，也不会透露你为什么落选了。

如果你得以进入主管或副总面试环节，面过你的面试官会聚到会议室正式表决录用意见。副总通常不会列席，但如果你没能打动他们，他们照样可以直接否决。招聘人员通常会在几天后联系你，要是等不及的话，你也可以主动联系。

必要准备事项：

如果你知道哪个团队会来面试你，务必先熟悉他们的产品。你喜欢该产品的哪些方面？你觉得有哪些可以改进的地方？给出独到见解可以有力展示你对这份工作的激情。

独特之处：

在苹果的面试中，二对一的形式司空见惯，不过也不用太紧张——这跟一对一面试并无分别。

此外，苹果的员工都是超级果粉，在面试中，你最好也能展现出同样的热情。

Facebook 面试

Facebook的在线工程难题[1]曾引发热议，其实这无非又是吸引眼球的手段之一。除了解答这些难题，你还可以通过传统渠道申请该公司的职位，比如提交在线职位申请，或者参加校园招聘会。

一旦被Facebook挑中，求职者一般至少要接受两轮电话面试。不过，公司所在地[2]的求职者可以少一轮。电话面试主要涉及技术问题，求职者通常要用Etherpad或其他共享文档工具写些代码。

如果你还在上学，在学校接受面试，那你还要写代码。面试官会要求你在白板或白纸上写代码。

现场面试时，主要由其他软件工程师来面试你，不过，招聘经理有空的话也会参与。所有面试官都受过专业面试培训，他们只提供意见，对你的应聘结果不作决断。

现场面试的每个面试官都各有侧重，以确保大家不会重复提问，并全面考察求职者的能力水平。面试问题主要分为算法、编程水平、软件架构/设计能力等几大块，同时，面试官也会

1 感兴趣的读者可以访问页面Facebook Engineering Puzzles: www.facebook.com/careers/puzzles.php。

2 Facebook总部位于美国加利福尼亚州的门罗帕克市，地址为黑客路1号(1 Hacker Way)。

考察你能否适应Facebook快节奏的开发环境。

面试过后，在交流你的表现之前，面过你的面试官会先提交书面评价报告。这么做是为了确保各位面试官能对你的表现作出相对独立的评价。

一旦收到所有的评价报告，面试小组和招聘经理便会商讨你的面试结果。他们会先达成统一意见，然后提交给招聘委员会。

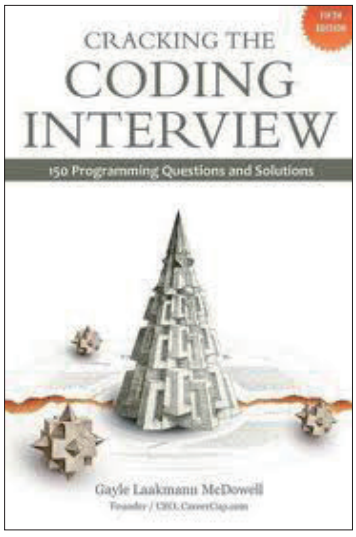
Facebook很看重“忍术”（灵活应变）——也就是使用任何语言快速构建优雅、可扩展解决方案的能力。懂PHP并不会显得特别突出，因为Facebook也有很多后台工作要用到C++、Python、Erlang和其他语言。

必要准备事项：

作为网络科技的新贵及“当红炸子鸡”，Facebook也更青睐那些富有创业精神的开发人员。在面试过程中，你要展现出自己热衷创造新事物的激情。

独特之处：

Facebook由公司统一招聘员工，而不是专门针对某个团队。面试成功并入职后，你会先参加为期六周的“新兵训练营”，帮你快速适应大规模的代码库。资深工程师会担任你的导师，辅导你掌握最佳实践和必备技能，最终让你可以游刃有余地加入自己喜欢的项目组。



《程序员面试第一书(第5版)》介绍了如何成功通过程序员面试，内容涉及面试过程、面试官的幕后决策及可能提出的问题、特定场景的面试过程(大型公司)、面试前准备工作及行为准备、技术问题的准备和复杂算法的解决方法，给出了150个程序员面试问答等。本书作者Gayle Laakmann McDowell是CareerCup.com网站创建人，该网站为提供了一个庞大的面试资料库，其中包括出自数千家大公司的面试题。此外，该网站还建有论坛，供人们交流面试经验。本文摘自《程序员面试第一书(第5版)》。

YAHOO!®

雅虎面试

雅虎往往只招美国排名前20的高校毕业生，不过其他求职者仍可通过雅虎公开招聘渠道(或者，可以内部推荐的话就更好了)得到面试机会。取得面试资格后，你会先接受一轮电话面试。对你进行电话面试的一般是资深员工，比如技术主管或经理。

在现场面试中，一般由来自同一团队的六七个人来面试你，每轮面试时长45分钟。每个面试官都会侧重不同的领域。比如，有的面试官可能侧重于数据库知识，而有的面试官则会关注你对计算机体系结构的理解。每轮面试的时间安排大致如下。

开头5分钟：一般对话。比如，自我介绍，聊聊项目经历等。

中间20分钟：编程问题。比如，实现归并排序。

最后20分钟：系统设计问题。比如，设计一个大型分布式缓存系统。这些问题往往与你以往的项目经历或面试官当前在做的工作有关。

当天面试结束后，你可能还会跟项目经理或其他人面谈一次。内容包括产品展示、你对雅虎的疑虑以及你手上有无其他公司的录用通知，等等。这次面谈旨在增进双方了解，通常不会影响你的面试结果。

与此同时，之前的面试官会讨论你的表现并尝试作出结论。最终录用与否由招聘经理决定，他会综合考虑面试官对你的正面及负面评价。

如果你的表现不错，有可能当天就会收到口头录用通知，但也不一定。也许他们要过几天才通知你，个中原因不一，比如，你应聘的团队可能还想再面试几个人看看。

必要准备事项：

雅虎面试少不了系统设计问题，几乎成了惯例，所以，还请做好相应的准备。他们想要确认你不仅会写代码，而且还能设计软件。要是没有这方面的知识，也不要紧，你仍然可以给出自己的设计思路。

独特之处：

雅虎的电话面试一般由拥有决定权的人负责，比如招聘经理。此外，雅虎往往会在当天给出面试结果（如果你能入他们法眼），这一点很特别。在你进行最后一轮面试的同时，其他面试官也正在讨论你的表现。

专题：码农的面试

Jeff Atwood感兴趣的程序员

作者/ Jeff Atwood
Jeff Atwood是一位美国开发者，同时他还是一位作者和创业者。他的博客Coding Horror很有名气。同时，他还是Stack Overflow以及Stack Exchange Network的创始人之一。

招聘程序员没有什么妙招，但是我可以分享一些我认为有效的且多年来一直在使用的技巧方面的建议。

1. 首先，通过一些简单的“Hello World”在线测试

我知道这听起来很疯狂，但是一些自称是程序员的人勉强能编写出来。时至今日，我仍能经常听到有人告诉我他们的候选人失败在最基本的编程测试上。

这就是为什么非常简单的编程测试成了任何明智的招聘过程中的第一步。这些测试应该是在线的，目的并不是为了证明候选人是一些编程天才，而是让他们应该知道什么是所谓的编程。是的，这有点可悲甚至有时是令人沮丧的，但是如果你不做这个明智的测试，相信我——你会后悔的。

部分提供在线程序测试服务的网站（我确信还有好多，但是我就知道这一俩个）有InterviewZen和codility。

2. 询问一下他们的履历

任何称职的程序员都应该有一个他们的工作履历，它不一定有多吸引人，但通过它我能发现一些蛛丝马迹，了解到你放在网上的那些对别人有用的大手笔。一份 Stack Overflow 的个人简历，我就能了解到你是以何种方式进行沟通的，你是如何解决问题的。给我一个你的开源代码库的链接，一个专业的 blog ？一个 Tumblr ？一个 Twitter ？或者我从来没听过的，最好能给我们展示下。分享下你设计的应用程序，或者你参与过的网站，并且描述下哪些部分是你做的。

仅仅看下他们做过什么样的工作，他们创建过什么样的网络产品，对于获得他们什么在行，什么不在行非常有益。

3. 聘用文化背景相适的

正如 GitHub，我发现相比整天埋头编程，文化背景相适应更是成功的前兆。

在招聘的过程中我们的谈论 [哲学]，我们很重视这件事。我们想让每个我们潜在的 GitHub 职工知道他们即将进入的是个什么样的公司，确定是否真的适应公司的环境。其中一步就是一起晚餐，一起谈论比如文化、哲学、曾经做过的错事、规划等乱七八糟的东西。

在早期，我们根据技能招聘，很少考虑到他们如何融入公司的文化或者他们是否懂得哲学。自然，这些招聘没有起到有效的效果。因此，当我们关注一个有潜力的职工的同时，他们是否能理解我们的做法也是一个重要组成部分。

我知道并不是每个企业都有业务相关的社区，但是如果你确实有，那么当你需要招聘的时候务必从你们的社区中挑选，这些伙计自然而然就融入到你们的工作环境中，这些候选人适应文化的可能性将是异常的高，而这就是你想要的。

有没有一些用户对你的游戏进行了惊人的改造？他们是否曾发现一个不起眼的安全漏洞并尝试去告诉你？立刻聘用他们！

4. 做一个详尽、结构化的电话约谈（phone screen）

一旦上面所说的都已经通过，现在是该给候选人打个电话了，记住，电话约谈不是聊天而是面试，通话应该是结构清晰的技术性问题，因此如果确实不合适，双方可以立即终止面试。电话约谈涵盖一些基础要点，概要的说：

1. 一点即时编程（on-the-fly coding）。“查找一个整型数组中最大的整数”。
2. 一些基本的设计。“设计一个HTML模型的表现层。”
3. 脚本和正则表达式。“从这个目录中提取一个包含特定格式的电话号码文本列表文件给我。”
4. 数据结构。“你什么时候会采用哈希表而不是一个数组？”
5. 位和字节。“为什么程序员认为10月31日和12月25日是同一天这个问题是有趣的？”（译者注：Oct 31和Dec 25，Oct为英文单词十月的缩写，同时Oct为八进制的意思，八进制31转换为十进制是25；Dec为英文单词十二月的缩写，同时为十进制的意思，因此八进制31和十进制25是一样的）

你在寻找的不是神奇的完美答案，而是探知这个人如何解决问题的、他们是否了解他们的工作职责（加或减10%），我们的目标是确保这个候选人在下一步中测试中不是浪彼此的时间，所以如果有太多不合适的预兆，立场要坚定，尽早结束这个通话。

5. 给候选人一个面试项目（audition project）

候选人轻轻松松的通过了Hello World编程测试，有一个惊人的履历，非常适合企业文化背景，同时出彩的通过了电话约谈，是时候进入面试了，是吗？伙计，并没有这么快！

我曾看到过候选人通过了所有上述步骤，进入公司后，根本无法把事情做好。雇佣程序员确实不容易。

如果你想打消这个招聘是否正确的疑虑，那么给他们一个**面试项目**。我不是说一个一般的、抽象的编程任务，而是一个**你们现在正需要立即解决的真实产品的模块**，你原本会将这个模块交给正式的职工去做，如果正式职工都不忙，那么让他做点其他的事情。

这应该是一个正规的操练方式，按工时明确的定义了项目的进度。选择一个可以在1-2周就能理想解决的小项目，无论这个候选人是在办公室办公还是远程办公。我知道并不是每个公司都有这些小模块可以外包出去——但可以尽力让他们在公司内完成。我想说，如果你想不出任何方式让一个不错的候选人参与面试小型项目，也许你现有员工的工作结构构存在不合理性。

如果面试项目成功了，那么太棒了——你现在拥有一个高素质的可以真正完成任务的候选人，你已经完成了要做的事情。时

至今日，我从来没有见过有谁通过面试项目而无法胜任工作的。我把面试项目的表现看的很重，因此在雇用之前让他们尽可能的做接近实际的工作。如果这个面试项目没有达到理想效果，那么，就把这个操练的付出看成是退出费吧，这与在公司里和4、5个人进行撒网式的面试过程相比成本少多了，最坏的情况下，你可以把这个面试项目交给下一个不错的候选人。

（招聘的试用期也可以工作，理论上和正式职工没什么差别，在大家一致同意的前提下，你可以在6-8周做“留还是不留”的决定）

6. 进入面试、推销环节

最后，你应该选个时间和候选人见上一面。这是不可避免的，但是之前还有一步就是**你应该在他们还没进入这个面试会议室之前，95%的确定这个候选人将会是个不错的职工**。

我还远远不是一个面试专家，但是我不喜欢面试难题，让问题温和善意化。

然而，对于如何面试程序员我有自己的理论：给候选人15分钟的时间让他们展现他们的专业领域。我认为这比传统面试好多了，因为你很快就会确定……

- 这个人对于他将要从事的工作充满激情吗？
- 他们能否的在小组中进行有效沟通？
- 他们能很好的应付他们所在的专业领域吗？
- 你的队伍会喜欢和这个人一起工作吗？

译者 / 刘永新
太原理工大学2005届计算机科学与技术专业本科毕业生，无鸿鹄之志，毕业后奋斗在县级市，习惯了不是很拥挤繁杂的城市，服务于制造业的信息化，常想有位老师能带带自己，为社会做点微薄贡献。图灵社区ID: 迷茫

每个程序员都应该懂得如何推销你自己、你的代码和你的项目，我完全同意。现在，推销自己吧！

7. 没有任何担保

以上的技巧不要生搬硬套，我知道这些技巧很好用，但是偶尔也有不起作用的时候，因此要要因“况”制宜，调整这些建议适用于你的特殊情况，剔除那些不适合于你的（尽管我强烈建议你不要跳过第一步骤）。即使在最好的环境下，**招聘职工也是很困难的**。一个工作机遇可能会远远超出任何人的控制，正如他们所说的，人本来就很复杂。

如果你认为工作像维持一种关系，那么余生中你将花掉每周40小时（或更多）时间，每个人理应合理支配时间。公司和候选人都应该真诚的尽最大努力决定对方是否合适。你的目标不应该仅仅是得到一份工作，或者雇用一个人为我工作，而应该寻找乐趣，热爱你的选择。不要急于任何事情，除非双方都感觉不错。

（顺便说一句，如果你正在寻找吸引程序员的方式，那么你不容错过来自SamuelMullen的出色建议）

英文原文：How to Hire a Programmer

金领简历的六大特点

一份优秀的简历能够直接传达出这样的信息：“我！我才是你需要雇用的那个人！” 每一行每一句话都应该让雇主迫不及待地想要录用你。那为什么还会有人在简历中列上含糊不清且完全

作者/ Gayle Laakmann McDowell

曾在微软、苹果与谷歌担任软件工程师。麦克道尔作为软件工程师在谷歌工作过三年，她还是该公司资深面试官及招聘委员会成员。她在美国本土与海外面试过150位求职者，为招聘委员会评估过1000多份求职者的应聘材料，还审查了数以百计的简历。除了作为面试官的丰富经验外，她先后面试并收到了来自12家高科技公司的录用邀约，其中包括微软、谷歌、亚马逊、IBM和苹果。2005年，麦克道尔创建了CareerCup.com网站，从面试官和求职者的角度，与人们分享她丰富的面试经验。CareerCup.com

无法证实的事情，比如，喜爱跑步？简历就只有一点点宝贵篇幅，所以除非你是在申请到健身俱乐部工作，否则不相关的事项就应该全部删除。

在提交简历之前，你应该逐行审阅每一句话，并且问一问自己，这些话语凭什么能够打动雇主，从而给你一个面试机会。如果你自己都给不出理由，那就表示你的简历完全没有说服力。

以下我列举了关于优秀简历的六大特点，你可以对照着检查，看你的简历是否符合这些标准。

以成就作为导向

如果你的简历读起来干巴巴的像职位描述，那你的思路就完全错了。简历应该突出你做过的事情，而不是你应该做什么。举例说明如下。

- **以职责作为导向是这样的**:“分析新兴市场并探索进入中国市场的可行性策略。”
- **以成就作为导向是这样的**:“主导了Foobar产品进入中国市场的策略，而且成功地说服CEO重新关注企业级市场，从而使得公司的利润提升了7个百分点。”

这种以成就作为导向的简历更加具有说服力。每个人都想招到这样“能够做些事情”的员工。

你要谨慎使用诸如“**做出了贡献**”，“**参与到**”或是“**帮助做了**”这类字眼。这些都恰恰表明你更注重职责而不是成就。毕竟，微软的某些员工也可以说他们“为实施微软Office产品做出了贡献”，可这有实际意义吗？

提供了一个庞大的面试资料库，其中包括出自数千家大公司的面试题。此外，该网站还建有论坛，供人们交流面试经验。

可量化的结果

你是否也曾经见过这样的广告营销语——“我们将一部分利润捐献给了慈善事业”？其中的玄妙之处就在于，哪怕你只捐出了0.0001%，从技术层面讲，这个声明也是成立的。

每次当我看到简历上“缓解了服务器的延迟情况”或“提高了客户的满意度”这样的话，我的感受就和看到上述广告营销语差不多。如果你真的做了这些事情（并且确实产生了一定影响），那为什么没办法告诉我具体的数值呢？

量化产出可以使雇主了解这些说法的实际意义，并能表明你产生的影响力。如果你真的取得了某些成就，降低了公司的开支或提高了利润，雇主们都会愿意聘请你。

对于商业性角色而言，用金钱来量化你的成就是最有意义的。然而，如果很难得到具体数值，你也可以通过员工流失率的降低、客服电话的减少或是任何最为贴切的衡量标准来量化你的成就。在描述事实性的变化之外，你也许还应该提供百分比的变化，或者用后者来代替前者。

对于技术岗位来说，采用更专业的技术术语来量化结果效果更好：延迟的秒数、bug的数量，又或者以Big-O度量出的算法改善程度。不过，你还是需要找到一个平衡点。尽管在同为工程师的人眼中，你的成就是了不起的，但过目你简历的人可能是一位不那么懂技术的人事部门人员，你要确保写出的话语能打动所有人。举例说明如下。

- **之前的版本**：“实施了系统崩溃报告功能，并应用其修复了三个最严重的导致系统崩溃的错误。”

- **改进后的量化版本**:“实施了系统崩溃报告功能，并应用其修复了三个最严重的导致系统崩溃的错误，从而使得客户技术支持电话减少了45%。”

查看之前的版本时，虽然我了解到你完成了一项非常重要的工作，但我并不太明白它的重要性。可改进后的版本却让我不由自主地发出“哇”的一声赞叹。

目标明确

回到打字机的年代，一份泛泛而谈的简历会得到人们的体谅。因为编辑简历着实是一个繁琐的过程，而且求职者经常是把简历打印个200份，再将同一份简历四处投给各个公司。人们当然喜欢看到很有针对性的简历，但也不会对此有过高要求。

现在，修改简历变得轻而易举，而且我们甚至极少将简历打印出来，所以针对不同职位来定制简历就成为必须要做的事情。随着竞争日趋白热化，为了让简历不输给他人，你就必须多付出一点心血，更何况还要让它脱颖而出呢。

你应该针对职位来定制简历，有可能的话还应该针对具体的公司。对于跨行业的求职者来说，这一点尤为重要。比如说，如果你在做过多年软件工程师之后想去申请一个技术领导的职位，你就得提到自己曾经领导了一项新功能的设计工作。或者，如果你想申请加入一家创业公司，而且你也了解到该公司要处理客户支持事务，那你就应该强调过去你在处理棘手客户方面都有哪些经验。

幸运的是，想让简历具有针对性其实并不难。通常，比较直接的办法就是查找关于公司和职位的信息，你只需要去检查他们

的网站或是研究职位描述。问问你自己，该公司当前面临的主要问题是什么？我的角色能够带来怎样的影响？即使你从未解决过该公司面临的具体问题，你也可能具备解决这些问题所需要的技能。

有通用意义

有些简历充斥着过多杂乱的技术术语，让人们很难领会其中的意思。所谓的技术术语并不单单指与计算机相关的词，它还可能是一些不常见的销售术语、市场用语甚至于内部“黑话”。那些出自大公司的求职者更要注意！他们在自己的公司里呆得太久了，以至于都忘记了像S+这样的术语其实在外面一点都不流行。（是的，微软那帮家伙，我指的就是你们！）

你应该让招聘人员以及未来的经理和同事都能读懂你的简历。避免缩写，并且要将高度技术化的术语转变成平实的文字。尽量量化你的目标和影响力，从而使得非专业人士也能了解你的贡献。哪怕没办法面面俱到，这也没有关系，你只是要确保大家都能从简历中领会到你的“过人之处”。

也就是说，有一些术语对某一些人来说更易于理解。比如，如果一位微软员工的简历中提到她曾经从60级升职到63级[1]，在西雅图的谷歌招聘人员一定会明白这是什么意思。

清晰、专业、简洁

很多招聘人员会因为一个书写小错误就丢开你的简历。他们觉得要查看的简历那么多，为什么要把时间浪费在一个不具备良好沟通技能的人身上呢？

[1] 微软内部使用59、60、61这样的数字来代表级别。比如，59/60的头衔是SDE，61/62是SDE II，63/64是Senior SDE，65/66是Principal SDE。西雅图是微软总部所在地，谷歌后来也在西雅图成立了研发机构，并快速扩张增员。

科技公司对此更宽宏大量一些，这是因为本身他们的公司氛围更随意，而且还因为他们有遍及全世界的员工。然而，我们不能因此就马虎大意，尤其是申请那些非常需要沟通技能的职位的时候。

所以，你要确保自己反复检查过以下这些方面。

- **简洁**。请避免在简历中堆砌大段大段的文字。人们讨厌阅读长篇大论，而且通常都喜欢跳过长段落。你的简历里面应该只罗列一些要点，每一个要点的篇幅大概为一到两行字。
- **拼写检查**。我要感谢我五年级的老师奥康纳女士，她教会了我一个拼写检查的小窍门。根据心理学原理，如果我们猜得到接下来的单词是什么，我们的大脑会倾向于忽略那些拼写错误。你可以尝试一下**从后往前**读你的简历来检查拼写错误。
- **语法**。你可以使用微软Word的语法检查器，不过不要完全依赖这个工具。如果你不是英语母语者，请确保找一位母语是英语的人—— 他应该在语法和拼写方面都很强——来帮你审阅简历。
- **页面空白**。0.5英寸的页面空白是不够的。理想状态下，你的页面空白应该有1英寸，当然至少不要少于0.75英寸。
- **正常的字体**。使用一种标准字体，比如Times New Roman或是Arial，字体大小不要小于10 pt。不要使用Comic Sans这样的花哨字体。
- **一致性**。你可以用逗号或分号来分隔列表中的项目，但是要保持前后一致。要么每一个要点都以句号作为结束，要么什么都不加。请确保你在粗体字、下划线、斜体等格式上保持一致。选择任何格式都不如保持格式的一致来得重要。
- **空格**。使用足够的空格会让简历更易于阅读。招聘人员每天要处理的事情已经很多了，不要让密密麻麻的简历增加他们的负担。

《金领简历：敲开苹果、微软、谷歌的大门》全面介绍了如何获得顶级科技公司的职位。为求职者详细介绍了招聘程序，如何应用、设计和定制简历，如何准备面试并在面试中胜出，如何处理被拒的情况，如何就录用条件进行谈判，以及如何卓有成效地处理工作中的事务，使自己的职业生涯更上一层楼。每部分都给出了回答范例并讲述如何应对。本书特别适合在校学生及希望进入顶级科技公司的求职者（特别是程序员、设计师和游戏开发员）阅读。本文摘自《金领简历：敲开苹果、微软、谷歌的大门》。

- **不使用第一人称**。虽然做到这一点很难，你还是要避免使用**第一人称**。在你的简历里面，除了求职意向这一部分需要第一人称外，其他部分应该尽量使用第三人称。

组织得良好清晰

当一位招聘人员拿起你的简历时，她的眼睛会不由自主地跳转到某些东西上。她想了解你的教育背景（学校、学历、专业和毕业的年份）及职业经历（公司、头衔、就职的期限）。对于软件工程师职位而言，她也许还想知道你是否具备相应的技术技能。

请记住，对招聘人员来说，想省力的最好办法就是干脆丢开这份简历。如果她在里面找不到想要的信息，那她很有可能会将你的简历扔到一边，然后再接着看下一份简历。

除了采用直观方式组织好你的简历之外，你还可以略微变通一下格式来凸显亮点。来看一看以下同一个求职者的两种不同格式的简历（极其精简版）：

鲍勃·琼斯（简历1）	鲍勃·琼斯（简历2）
软件设计工程师（2008年至今） 微软公司 • 设计了Visual Studio的一些模块。	微软公司（2008年至今） 软件设计工程师 • 设计了Visual Studio的一些模块。
软件工程师（2006~2008） 英特尔公司（加州圣克拉拉市） • 改善了芯片中的嵌入式代码。	英特尔公司（2006~2008） 软件工程师 • 改善了芯片中的嵌入式代码。
软件工程师（2002~2006） 思科公司 • 在6年工作生涯中发布了8个产品。	思科公司（2002~2006） 软件工程师 • 在6年工作生涯中发布了8个产品。

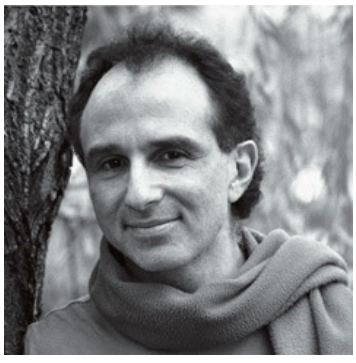
虽然两份简历传递的信息完全相同，但第一份简历强调的是鲍勃一直从事软件工程方面的工作。当然，这很切题，可是简历里面并没有什么亮点。第二份简历却突出了这些公司的显赫大名：微软、英特尔和思科。你认为哪一份简历效果更佳呢？

在撰写简历的时候，问问自己：哪些东西最能将自己与其他求职者区分开来？哪些因素会使得招聘人员将我的简历放在“很好”的那一堆里？最理想的情况下，这些信息应该显眼到哪怕只是瞥一眼，人们也不会就此错过。

专题：码农的面试

数独问题

——程序员面试逻辑题解析



作者/ Dennis E. Shasha
纽约大学柯朗数学研究所计算机科学教授，先后获得耶鲁大学理学学士、雪城大学理学硕士和哈佛大学哲学博士学位。《科学美国人》网站 和Dr. Dobb’ s Journal的谜题专栏作家。除本书外，还著有《奇思妙想:15位计算机天才及其重大发现》，以及自然计算方面的一系列著作。

我第一次接触数独，是在火车上。当时，我12岁的孩子给我一本书，让我做上面最难的一道题。在行进的火车里解题让我有点不舒服，因此我决定把这道题交给程序来做。我花了3个小时编写了一个100行的程序，即便是网络上最难的数独，它也能在2秒钟内解出来。我不是在自吹自擂。迄今为止，我所知的最短的数独求解程序是由亚瑟·惠特尼(Arthur Whitney)编写的，他用的是自己开发的语言——q语言，程序的长度是103个字符。虽然我不提倡程序越短越好，因为这往往以牺牲代码的可读性为代价，但是103个字符的数独求解程序着实让我惊叹不已。

数独是可应用排除法的谜题。在9×9的九宫格中，用1到9这9个数字填满整个格子。每个数字在每行、每列及每个小九宫格(从左上角开始的不重叠的3×3的9个方格)里正好只出现一次。

热身问题

考虑以下数独谜题：

								7
7		4				8	9	3
		6	8		2			
		7	5	2	8	6		
	8				6	7		1
9		3	4				8	
			7		4	9		
6				9				
4	5	9				1		8

用数字 0 代表此格还未被填写。

0	0	0	0	0	0	0	0	7
7	0	4	0	0	0	8	9	3
0	0	6	8	0	2	0	0	0
0	0	7	5	2	8	6	0	0
0	8	0	0	0	6	7	0	1
9	0	3	4	0	0	0	8	0
0	0	0	7	0	4	9	0	0
6	0	0	0	9	0	0	0	0
4	5	9	0	0	0	1	0	8

让我们先考虑左下角的小九宫格：

0	0	0						
7	0	4						
0	0	6						
0	0	7						
0	8	0						
9	0	3						
0	0	0	7	0	4	9	0	0
6	0	0	0	9	0	0	0	0
4	5	9	0	0	0	1	0	8

可以推断出，在左下角小九宫格中由5个0代表的空格中，肯定有两格分别为7和8。但是倒数第3行已经有一个7了，因此小九宫格中的7肯定不在最上面那行。同时注意到，在第3列中也已经有一个7了，因此在小九宫格中，数字7唯一可能的位置是6右边的那一格，得：

0	0	0						
7	0	4						
0	0	6						
0	0	7						
0	8	0						
9	0	3						
0	0	0	7	0	4	9	0	0
6	**7**	0	0	9	0	0	0	0
4	5	9	0	0	0	1	0	8

同理可以推断出数字7在右下角的小九宫格中的位置只能是最后一行，在空格中填入7，得:

0	0	0						
7	0	4						
0	0	6						
0	0	7						
0	8	0						
9	0	3						
0	0	0	7	0	4	9	0	0
6	7	0	0	9	0	0	0	0
4	5	9	0	0	0	1	**7**	8

我们可以先拿一个例子，手工演算一下，往往大致算法也就出来了。此题亦可如此。首先，检查每个填入0的空格的约束条件，得出该空格可以填入的数字。一旦找到一个空格存在唯一可选的数字，将该数字填入空格中，重新计算所有其他空格的约束条件。

让我们这样试试看，九宫格现在的状态是:

0	0	0	0	0	0	0	0	7
7	0	4	0	0	0	8	9	3
0	0	6	8	0	2	0	0	0
0	0	7	5	2	8	6	0	0
0	8	0	0	0	6	7	0	1
9	0	3	4	0	0	0	8	0
0	0	0	7	0	4	9	0	0
6	7	0	0	9	0	0	0	0
4	5	9	0	0	0	1	7	8

我们先考虑左上角的那个空格，跟它相关的所有格子如下图所示：

0	0	0	0	0	0	0	0	7
7	0	4						
0	0	6						
0								
0								
9								
0								
6								
4								

左上角空格内的数字可以是除4、6、7和9之外的所有1到9的数字，也就是说，可以是1、2、3、5或8。这并没有缩小多大范围。我们继续考虑左上角小九宫格内的中心格：

0	0	0						
7	0	4	0	0	0	8	9	3
0	0	6						
	0							
	8							
	0							
	0							
	7							
	5							

这格的数字可以排除3、4、5、6、7、8和9，因此只可能是1或2。相比之前，这已经大大缩小了范围，但还是不够完美。

继续检查各个空格的可选值，看看是否能找到一个空格，可以填入的数字是唯一的。例如，考虑位于左中小九宫格内的左上角空格：

0								
7								
0								
0	0	7	5	2	8	6	0	0
0	8	0						
9	0	3						
0								
6								
4								

可以排除掉2、3、4、5、6、7、8和9。因此，唯一可能的数字是1，九宫格的状态为：

0	0	0	0	0	0	0	0	7
7	0	4	0	0	0	8	9	3
0	0	6	8	0	2	0	0	0
1	0	7	5	2	8	6	0	0
0	8	0	0	0	6	7	0	1
9	0	3	4	0	0	0	8	0
0	0	0	7	0	4	9	0	0
6	7	0	0	9	0	0	0	0
4	5	9	0	0	0	1	7	8

现在，让我们考虑1右边的那个空格，跟它相关的所有格子如下图所示：

	0							
	0							
	0							
1	0	7	5	2	8	6	0	0
0	8	0						
9	0	3						
	0							
	7							
	5							

 这一格可以排除掉1、2、3、5、6、7、8和9，这么一来，4便是唯一的选择。（有的时候可以利用更多的相关格子。例如，在第一列中有一个4，这说明1和9中间的那个空格便不可能是4。但是千万不要在前行的火车上尝试这样的推理。）下面留给你了，试试看能不能解开这道数独，你会发现其实一点都不难。

热身问题解答

8	1	5	3	4	9	2	6	7
7	2	4	6	5	1	8	9	3
3	9	6	8	7	2	4	1	5
1	4	7	5	2	8	6	3	9
5	8	2	9	3	6	7	4	1
9	6	3	4	1	7	5	8	2
2	3	1	7	8	4	9	5	6
6	7	8	1	9	5	3	2	4
4	5	9	2	6	3	1	7	8

在热身问题的解答过程中，总是至少存在一个空格，可以根据约束条件排除到只剩下一个可能值。也就是说，我们并不需要作出“试探猜测”——给空格内赋一个（并非唯一）满足约束条件的值，然后一一验证。让我们试着来设计一个算法。

以下伪代码是针对不需要依靠试探猜测的数独的（basicsud）：

过程basicsud的执行流程如下。

第1行设置标志变量stillchanging初始值为真，此变量为真代表数独还可以继续填写。第2行至第11行为一个while循环，当stillchanging为假时循环结束。在此循环体内执行以下步骤：

1. 首先将stillchanging设为假；
2. 随后为数独中所有依然为0的单元格（为0表示还未被填写）找出它们的约束条件；
3. 如果存在一单元格e，根据约束条件存在唯一解v，则将v填入e中，并设置stillchanging为真；
4. 如果存在一单元格e，根据约束条件没有可以填入的数字，则返回“不一致状态”。

所有具有唯一可能值的单元格都填满后，循环结束，第12行返回数独状态，过程结束。

这个算法不但可以用来应对简单的数独，对于解决更难的数独问题同样至关重要。较难的数独问题需要依靠试探猜测，并且需要检验给出的可能值是否符合各类约束。例如，如果某个空格所在的行和列已经包含了所有1到9的数字，就会出现“不一致的状态”。

需要试探猜测的求解过程是怎么样的呢？让我们跟着直觉走。

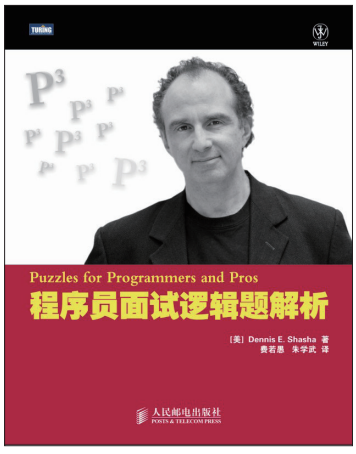
假设我们从“一致的状态”开始。首先调用basicsud，如果能够使数独达到“完成的状态”（每一个空格/0都填入了相应数字），那么问题便解决了。如果每个空格的可能值都在两个或两个以上，那么对于每个空格的每个可能的值，我们都系统地进行检验。也就是说，对于当前的每个空格，我们先保存现有的状态，尝试某一个可能的值。如果这个值将导致数独进入“不一致的状态”，我们便回退到之前的状态，然后尝试下一个可能值。

以下这段伪代码（specsud）使用了一个堆栈来保存各个状态。当需要作出猜测时，便将当前状态入栈保存。如果此次猜测不可行，便将栈顶状态出栈。

过程specsud的执行流程如下。

首先调用basicsud。如果basicsud返回“不一致状态”，则说明该数独无解，过程结束。如果basicsud返回的数独s'为已完成数独，则返回s'，过程结束。第6行至第18行对应s'为未完成数独的情况，主要逻辑为两层嵌套循环，其中第9行至18行为外层循环，用R代表数独s'中未填写的有两个以上可能值的单元格集合，对于R中的每个单元格e，执行内层循环。第11至17行为内层循环，遍历单元格e的所有可能值，对于每个可能值v，执行以下步骤：

1. 首先将数独现在的状态s'放入堆栈中保存；
2. 再将v填入数独，记数独状态为s''；
3. 递归调用specsud，传入参数s''，记返回数独状态为s'''；
4. 如果返回的s'''是一个完成的数独，将此状态s'''返回，过程结束；
5. 状态s'出栈。



《程序员面试逻辑题解析》作者在纽约大学柯朗数学研究所开设了多年的谜题分析课程，积累了不少题型，总结了多种解题思路。书中从不同角度阐释了各种类型谜题的解题技巧，从广为人知的数独、幸运轮盘赌、赛程编排、旅行推销员问题到独具一格的猫鼠游戏、同盟最大化及选择性贪心等。通过学习本书，读者可以开拓视野，启发思路，不仅能从容面对面试中遇到的各种谜题，更能培养在实践中确定最佳方案的技巧。如果你想挑战一下自我，不妨拿起本

这个算法几乎用任何语言都很容易实现，得出的代码的运行时间一般不会超过1秒。你可以试试看，用一台新款的个人电脑，能否在3秒之内解决下面这道数独。我之前提到过的103个字符的程序耗时不到100毫秒，但是这并不是一场比赛。真的，确实并非比赛。

求解下面这个数独。

0	3	0	0	0	0	0	4	0
0	1	0	0	9	7	0	5	0
0	0	2	5	0	8	6	0	0
0	0	3	0	0	0	8	0	0
9	0	0	0	0	4	3	0	0
0	0	7	6	0	0	0	0	4
0	0	9	8	0	5	4	0	0
0	7	0	0	0	0	0	2	0
0	5	0	0	7	1	0	8	0

答案

求解下面这个数独。

这道数独有点难度，答案参见下图。根据之前给出的伪代码，我简单地编写了一段伪代码求解此题。回溯的次数非常少——低于50次。

书，来一场头脑风暴。本文摘自《程序员面试逻辑题解析》。

8	3	5	1	2	6	7	4	9
4	1	6	3	9	7	2	5	8
7	9	2	5	4	8	6	3	1
6	4	3	9	1	2	8	7	5
9	8	1	7	5	4	3	6	2
5	2	7	6	8	3	1	9	4
2	6	9	8	3	5	4	1	7
1	7	8	4	6	9	5	2	3
3	5	4	2	7	1	9	8	6

顺便说一下，可能会有下面这样的数独的变体游戏，但是我还没找到类似的。我称之为数独对杀。这是一个两人对弈的游戏。假设我跟你一起玩，我们俩轮流在数独板上填入数字。如果你没有数字可填，那么我就赢了。所谓没有数字可填，一种可能是因为数独已经被填满了，还有一种可能是无论你填入什么数字，都会违反数独的约束。类似地，如果我无路可走，则你赢得比赛。

专题：码农的面试

面试官：

“谈谈你进展不顺利的项目”

作者/ Andy Lester
《人人都有好工作:IT行业求职面试必读》一书作者。他通过制作编程工具，演讲和给人们切实的建议，来帮助更多的人找到他们真正喜欢的工作。他的博客petdance.com.

下面列出的棘手问题并不全面。事实上，列出这些的目的仅仅是希望你能够学会如何正确地讨论应聘工作和个人资质，而并非让你将它们作为背诵模板。

再次提醒，面试是你来到公司的第一天。你需要把重点放在自己能给招聘经理、给公司带来什么利益。

谈谈你自己

这是最重要的，也是最可能被问到的最棘手的问题。如果你只打算准备一个问题，那么就准备这个吧。其他任何问题都只是有可能被问到，但这个问题势必会出现在每一场面试中，最多是措辞不同。这是一个经典的开放式问题，给你足够的空间阐述自己的优势。

它是一个论文式的问题，并不是多选题。对方会突然间询问你是怎样的人，能为他带来什么。所以这个答案必须事先考虑好。

下面是一个最糟糕的答案，这个答案说明了你完全没有任何准备。

面试官　谈谈你自己吧。

糟糕的回答　您想知道些什么？

这个回答表现出你一无所知，或是根本没有能力推断面试官想了解的信息。这表明你未曾考虑过为何这份工作适合你。面试官很有可能就会在这儿终止面试。

那么你该说多少呢？就用30秒时间把自己最闪光的地方陈述一遍。你的目标就是给出一系列的优势，而不让听众觉得厌烦。总的来说你应该把简历的自述部分用口语化的方式叙述出来。

糟糕的回答　那么就从我小时候开始说吧……

糟糕的回答　我是个程序员。哦，当然目前还不完全算是，因为从1月份开始我就失业了。在家里可没什么机会做编程。但我还是很希望能得到这份工作，因为老实说您也知道在没有收入的情况下身背贷款是件多艰难的事，对吧？哦，我爱好学习早期编程语言。

恰当的回答　现在算起来我做系统管理员已经有7个年头了。最开始我在一家只有10个人的公司中维护一台Windows NT服务器。后来我们在一台Windows NT的域上建了一个Samba服务器，管理150名用户，这些用户有的使用Windows，有的使用Linux。同时我也做一些编程工作，以及写shell脚本。我还曾经为Nagios远程监控服务写过插件，并且为Bugzilla提交了几个补丁。过去我一直从事的是市场营销行业，所以我觉得是时候丰富自己的阅历了。能在Yoyodyne工作对我来说是个不错的转变。

现在你自己试一试，像上面这个例子一样写一段自己的回答。使用完整的句子，而不仅仅摘录关键的要点。而且这只是一个通用的答案，并非为某份工作或是某家公司量身定做。但是先这样练习一遍也无妨。

说真的，赶紧照做吧。如果你愿意可以就写在这本书的某个空白页上。

现在大声地念出来。给自己的朗读计时。你能控制在大约30秒吗？如果需要1分钟，那也许你加进了过多的细节。如果不到30秒，有可能你遗漏了一些值得讲述的重点。

大声练习朗读，直到能对答如流。
Practice your answers out loud to get comfortable with them.

朗读答案时你的语气是自然的吗？必须学会流畅地回答，而且还要对推销自己和阐述自己的优点感到自豪。把你的答案说给朋友听，问问他们你的语气是正常还是怪异。我并不要求你像背台词一样去强记，但你还是需要勤加练习。向脑中植入这些句子好让自己到时候能够顺畅地说出口。

麻烦的是，“谈谈你自己”这个问题的答案每一次都会不同。你的回答取决于谈话对象以及所处的面试阶段。思考在下面场合你该如何回答这个问题。

- 面对人力资源筛选人员：用保险的方式回答。谈谈你这些年的工作经历以及都在哪些公司供职。聊一聊你的高水平技能（数据库、Linux方面），但不要过于细化（具体到Oracle和Postgres，Red Hat和Ubuntu）。当然在提到那些招聘广告中列举的具体技能要求时可以适当细化。
- 面对你的未来同事：尽可能细化你掌握的技术，还可以时不时地蹦几个时髦的技术词汇让你们之间的对话产生火花。
- 面对非技术型管理层领导：强调你的成就、完成的项目以及贡献出的商业价值。尽量避免使用那些时髦技术词汇。
- 面对你的未来上司：什么内容都提一点，强调一下团队合作精神以及软技能。

你的回答会根据应聘职位以及公司的不同而调整。如果招聘广告中强调过某项专业技能要求，那就在你的回答中突出它。如果你有相同行业的从业经验，那么这也是你需要强调的地方。

面试官提问的方式也可能各有不同，也许会采取更直接的方式，诸如："你觉得自己比起别人有哪些方面的优势？"。

你对我们公司有什么了解

这是我个人偏爱的开场问题，因为它能让我了解到应聘者为面试做了哪些准备。应聘者对每个面试都是无差别对待的吗，还是他只对我们公司特别感兴趣？他已经做足了功课把公司调查清楚了吗？对于应聘者来说，这个话题也可以成为自我介绍的引子，并且你还能从中了解公司的需求。

糟糕的回答　那个，说实话不太了解。我觉得你们的网站做得不错。

恰当的回答　HoseCo是一个专门生产工业用软管及配件的公司。公司1954年成立，在70年代时搬到目前的所在地。我想知道你们会为航空业提供材料吗？因为我曾经在波音公司的冷暖装置部门做过顾问，那对我来说是一段难忘的经历。另外，我发现你们网站使用的是JSP技术，但是在线产品目录似乎使用的又是CGI。请问那是用Perl语言写的吗？而且我知道你们也在辛辛那提设立了第二分部，这也说明了公司运营情况不错。那么请问你们和俄亥俄分部之间是如何建立通信的呢？

通过回答问题，你显示出自己是有备而来的，而且还证明了自己能用老板的思维来看问题。因为你提到的个人背景正是能够为公司带来利益的。

你对公司的什么地方感兴趣

在其他条件都对等的情况下，没有哪个招聘经理希望录用对工作或公司毫无兴趣的员工。而且关键是，你也不应该找一份自己提不起兴趣的工作。如果你对这个问题的答案根本毫无头绪，那么也许根本不应该申请这份工作。

糟糕的回答 这个，是因为离我家很近。

即使公司离家的距离很重要，那也不应该让对方觉得这是你的首要关注点。永远都要记得把公司利益放在第一位。另外，这个问题也是个机会，让你提起自己和公司之间的联系，或者说起公司中熟人。

恰当的回答 我一直都很喜欢汽车，所以能为一家生产汽车零部件的公司工作对我来说是十分理想的选择。招聘广告上说公司使用的是Ruby on Rails，这也正是我很愿意做的工作。因为过去我一直使用Java Struts。另外，我在贵公司财务部门的好朋友苏西·德金斯也向我推荐这儿。她总是和我说起这个大家庭里大家相处得有多融洽，而且贵公司离我家也只有10分钟的路程。总而言之，这份工作对我来说十分理想。

你最大的优势是什么

面试刚开始的时候，话题重点是“谈谈你自己”部分。而在面试后期，很可能话题将会转到如何评价自己这儿。无论哪部分，都应该挑选自己擅长的内容说，同时给出支持自己说法的例子。最好能提供软技巧和硬技能各一个。

糟糕的回答 我工作很努力。（每个人都是，兄弟。）

糟糕的回答 我是一名优秀的程序员。（语义不明确，没有提供细节支持。）

糟糕的回答 我就像是活的ASCII码表！我可以说出任意字符的十六进制值！（好吧，那又怎样？）

恰当的回答 遇到危机时我都能保持冷静并很快集中精力，很少有事情会让我感到绝望。旁人都对我处理棘手问题的能力感到惊讶。您也知道，对于我们系统管理员来说，眨眼之间就可能面临一大堆麻烦。

恰当的回答 我对数据抽象化很有心得。建立数据库模式、编写接口这一类的事情对我来说就像是与生俱来的能力。这是工作中我最爱做的部分。

不用对分辨哪一项才是自己最大的优势发愁。实际上你只要选择那些对公司最有益的优势展开就可以。

你最大的缺点是什么

这似乎是大家最头疼的问题。面试官想让你亲口坦白自己究竟有哪些不适合这份工作的缺陷。当然事实上你根本不需要说自己不适合的地方。请确保自己不会被突如其来的问题吓到，从而坐在位子上苦思冥想哑口无言。

大多数求职书籍都会建议使用一个巧妙的回答，诸如“我工作过于拼命”，或是“当我发现大家不如我工作努力时会很失望”，亦或是“我是个完美主义者”。这些回答的思路没有错，因为他们知道要把对自己不利的局面转向有利的方向。但是这样的回答存在两方面的问题。首先，这很可能并非你的真实情况，也就是说你在撒谎。其次，更糟糕的是，它们完全就是一通废话，而且面试官很清楚这一点。这样的回答可以明显看出求职者在刻意撒谎。

正面地处理这个问题。把它当做是“你希望自己在哪方面得到提高？你现在提高的情况如何？”这样的问题来对待。确保回

答的语气体现了这些的确是你个人需要提高的部分，而不要表现得一切似乎都是别人的错。选择技术方面的弱势，而非个人的失败之处或是人格上的缺陷。

糟糕的回答 当我发现大家不如我工作努力时会很失望。（不仅是个无用的回答，而且提出的还是别人的错误。）

糟糕的回答 我是个完美主义者。（不仅是个无用的回答，而且你能说出自己打算如何改进吗？）

糟糕的回答 我真的很讨厌测试代码。（这对于一名程序员来说可是个巨大的失败。）

恰当的回答 我对JavaScript和Ajax了解得还不够多。我目前从事过的编程工作都是偏向服务器方面的，但是显然，Ajax技术会继续成为主流。前阵子我买了一本Pragmatic Ajax，现在正在努力学习中。

我们凭什么雇你

如果这个问题在面试最后或是与更高层的第二轮面试一开始提出，那么意味着对方正在邀请你进行自我总结。概括在之前的面试中讨论过的内容，把新的强调重点放在自己对于公司需求以及问题本身的了解上。说说一旦入职之后自己将会为公司采取哪些具体行动："您刚才提到了公司面临的数据库规范化问题，这正是我可以提供帮助的地方。去年我曾经率领一个数据库管理员团队……"

有些人会觉得这个问题是一种威胁。因为它给人的感觉就像是在问："你究竟有什么好的，兄弟？" 这感觉就好比将一个重担压在应聘者身上。千万别这么想。就把它当成是用略带攻击性语气的 "谈谈你自己"，这样你的表现会自如很多。如果这个问题出现在面试开场，那么在回答完之后再加一句："当然，这

些并不是我能够为公司带来利益的全部，我希望了解更多公司和部门面临的挑战，这样我才能知道如何能发挥自己全部的力量。”

谈谈你经历过的进展不顺利的项目

对于这样的问题不存在所谓的正确或错误答案。面试官希望从中了解你是如何处理问题以及如何面对逆境的。案例在这儿非常重要。如果你被问道“你曾经遇到过某某事件吗？”那么就把这个问题转为“给我讲一个关于某某事件的故事。”

从这个问题中能得出两方面的结论。面试官想看看你究竟是如何处理日常工作中遇到的不顺，以及了解你是否是一个抱怨者。你是会坦诚自己的错误还是将责任推卸到别人身上。最好能从自身和他人两方面的角度来说明这个问题，但是千万不能埋怨他人。看看下面这个典型的抱怨例子。

糟糕的回答 哦，好的，该讲哪一个呢？我们网络团队总要处理一些非常紧急的状况，因为市场部净安排些计划时间表非常不合理的项目。这样一来我们能腾出的时间往往不合乎他们的期望，于是他们就不停地责怪我们，搞得我们像群傻瓜一样。他们自己没能力提前安排好计划又不是我们的错。

接着这个问题的往往会是：“那么你从中学到了什么”或是：“那么你会从哪些方面着手来防止这样的情况发生？”

糟糕的回答 没学到太多。我想我们只能和市场部的人讲，没法事事都按照他们的意愿进行。我真想给市场部主管上堂课，告诉他软件工程究竟是怎么进行的！

每一句回答都在指责别人，而且使用的是侮辱性语言。面试官

从中可以看出一旦这位应聘者入职之后她很可能也会不停地抱怨中伤团队其他人。依应聘者之见谁才是需要改进的人？不是她自己！而是这位罪孽深重的市场部主管！

面对相同情况，请看下面这个回答。这一次的叙述并没有将错误指向任何人，也不带任何仇视心理。

恰当的回答 近来我们和市场部之间出现了一些摩擦。新来的市场部主管对网站有一些宏伟的方案，而这些方案的完成期限不容商议，是为了即将到来的贸易展的。第一次合作时，我们就没办法按照他的预想办好事，从而引发了很多不愉快。他对我们这个网络团队很失望，而我们又觉得他的要求非常不合理。很显然，这是因为我们之间的沟通还不够。

现在，在面试官还没开口接下去提问时就主动回答。

恰当的回答 第一次的不愉快之后，我们就主动碰头总结了这件事情。我们很高兴能看到市场部有着这样的进取心，但同时我们也商量好保证之后的项目要求合情合理。老实说一开始还很难谈拢，但是当市场部主管意识到我们其实是站在他那边后，进展便开始顺利了。

谈谈你犯过的最大错误

这个问题直指两方面的信息。首先，它能体现出应聘者如何处理“项目进展不顺”的指责。其次，这也展示了应聘者的经验水平。

只有实实在在地工作过才会犯错。如果你从来没犯过错，那只能说明你并没有付出足够的努力。除非你所在的行业绝对不允许错误发生，譬如医疗以及航空业，犯错意味着有人丢掉性命。同时，错误也是学习与成长的过程。

挑选一个可以证明你有能力为错误承担责任的例子，并且在回答中增加说明“你从中学到了什么”。

糟糕的回答 我不知道，我想我没犯过什么错。

糟糕的回答 我曾经有一次错误地将C盘格式化了。那简直糟透了。

恰当的回答 当时我们试图更换一个新的电子邮件系统，我使用Perl语言从Notes中读取数据，并写进Exchange里。周一早上，用户气愤地发现他们存档文件被错误地重新分类了。所有存档文件中的邮件都被移至单一的文件夹中，而那时候已经来不及重新运行指令了。打从那以后，我就发誓今后每一个数据转移项目都必须添加转移前的确认步骤，这样可以让我在指令实施之前重复确认以保证万无一失。

如果……你将怎么处理

这样的问题通常建立在某个特定场景下，这个场景并没有显而易见的正确处理方式。这时候你采取的行动能体现出处理事务的灵活性以及解决问题的能力。即使问题场景似乎与技术有关，但通常情况面试官让你做的却是非技术层面的判断。下面是可能出现的问题：

- 在进行全公司范围的软件升级时，你发现了某员工的电脑里有一个色情内容的文件夹。当然，在电脑里存放色情内容是与公司政策相违背的。这台电脑的主人刚来公司不久，而且他看起来是个不错的小伙子。这时候你该怎么处理？
- 假设你处在代码冻结阶段，距离某个软件的发行日子只剩下最后两天，这时候你发现由同事负责编写的代码中有一个小bug。你知道这位同事之前也出现过代码质量问题，而且他也十分担心自己在公司的业绩评估。你可以轻松地修改这个bug而不用告诉任何人。但是现阶段，照理说任何改

动都需要经过项目经理的批准。那么这时候你该如何处理？

- 你所在的是一个小公司，某天公司总裁找到你，表示他对公司网站极度不满。他坚持用户注册表格中应该使用单选按钮而不是现在的下拉菜单。他要求你今天就改过来。从技术上来说，你可以轻松地按照总裁的要求修改，但这样就和部门现有的流程规定不符。哦对了，你的头儿正好在度假，不在公司。这时候你该如何处理？

上面三个例子都没有所谓的正确答案。面试官也许只是对你的思考过程感兴趣，那么请确保回答中包含了你的思路。

恰当的回答 当然，我肯定是不想让这人惹上麻烦的，而且对我来说删掉这个文件夹并且警告他这有悖公司规定是很容易的事。但从另一方面说，规定就是规定，可是对像我这样的技术支持人员来说也无权强制执行某个规定。那么最后，我想我自己也许会……（并解释你这么做的原因）

看看你究竟如何深入挖掘问题也是面试官希望了解的一部分。也许面试官想听到的就是你继续问“那个bug有多严重？”。那么就接着挖掘更多信息吧。但是这也并不意味着你需要连续问20个问题才得出最终结论。面试官期待的仍然是你的果断回答。

这些“如果……？”句式的问题对你来说正好是一个建立优势的机会。因为它们通常会建立在招聘经理真实经历的问题基础上，这也就给了你一个了解对方的机会，同时也提供了继续追问的空间。在回答完自己的想法之后，继续对新产生的疑问刨根问底。

恰当的回答 我会照着总裁的意思做网站调整。但同时我也会对原有的版本做好代码备份，以防头儿回来之后有不同的意见。这的确是一个两难的境地。请

问是不是部门过去遇到过这样的情况呢？

你不需要询问经理正确答案是什么，因为很可能自己就能挖掘出来。他也许会说："是的，我们负责销售的副总就曾经如此利用了自己的权威。现在每个下属都会找我做决定，不管我处在什么情况下。"

请注意上面这个例子中，你的回答和经理希望发生的情况是不同的，但是这么回答也无妨。他想要检验的是你的决定过程，而不是最终选择的正确性。

这样的问题也是检验你和公司文化是否合拍的一种方式。你也许会发现公司总是严格按照规章制度办事，或永远都唯老总马首是瞻。如果你和公司文化有无法融合的地方，最好趁现在赶紧发掘。请干脆地回答问题而不要过于絮叨。

你更喜欢团队行动还是自己单干

独立完成工作与团队合作代表了一枚硬币的正反两面。经理总是希望团队中的每个成员都能够独立完成工作而非事事都需要他的指引。但从另一方面说，不愿意或不能够和他人共事的员工，或是无法听从经理指示的员工都是团队生产力极大的阻碍。对于经理来说，比必须手把手地指示员工干活还要浪费时间的，就是处理团队成员之间的矛盾问题了。

请注意回答这个问题的态度。千万别给人你只能在自己偏好的状态下才能工作的印象。而且对于大多数情况来说，答案都不应该只是简单的A或B，应当给出合适的例子。

面试官　你更喜欢团队行动还是自己单干？

糟糕的回答　哦，饶了我吧，拜托还是让我自己单干吧。

近乎糟糕的回答　我发现最佳的工作状态就是自己单干而不加入任何人的情况。

恰当的回答　一切都依项目情况而定。如果我编写的是一次性工具代码，那么一般来说还是由个人独立完成比较好，最多也就是再安排一个同事帮我把关，确保没有任何地方出错。当然，对于大型项目来说，都是需要一整个团队人员的支持，而且项目的每一个分支该以个人完成还是两人一组完成也都是依据任务的性质而定。我想我个人是更倾向于可以发挥出最佳效率的独立工作状态，但是大多数情况下，都是会要求整个团队之间合作的。

你希望自己 5 年之后在做什么

就是它了——史上最让人头疼的面试问题。说它最让人头疼是因为网上有非常多的人在抱怨这问题，或是私下向我诉苦。

即使痛恨这个问题，应聘者的答案的确会提供很多关于未来打算以及商业头脑的信息。请参考下面这位应聘者目光短浅的回答。

猜猜我最爱的语言！

英国多西特软件开发人员　艾德里安·霍华德

刚开始面试程序员时，我很惊讶原来有那么多人根本没准备或是没思考过关于面试的任何问题，有那么多人压根儿没对公司进行过调查，有那么多人没想过自己的技能如何能够应用到面试中去，有那么多人无法跳出狭隘的技术领域去拓展更广泛的思路。

有一个人给我印象深刻。当我们刚开始面试的时候，我就顺口问了他一句最近在看什么书。他说刚开始看一本讲 Visual C++ 的书。那时候我还觉得这样挺好的。

后来在面试时我问了他觉得自己适合做什么工作。他说做“编写Visual C++程序的工作”，而这时候我们根本还没谈到项目具体需要使用什么编程语言。他也根本没提自己的技能如何与公司产品相结合的问题。这时候我开始对他有一点点担心。

在面试快结束时我询问他如何打算自己在5年之后做的工作。他的回答就如同你猜测的一样：“希望那时候我在开发Visual C++的程序。”这下我害怕了。这人居然把所有重心以及长期计划都建立在单一的语言与单一的平台上。

最后他收到了一封礼貌的拒绝信。

面试官　你希望自己5年之后在做什么？

糟糕的回答　我没想法。

糟糕的回答　谁能说得准呢？这可是计算机行业，我现在说的任何回答都不会是正确的，因为技术发展日新月异。

恰当的回答　就我目前的职业发展来看，我希望自己到时候能处在团队领导的位置，但是我还并不确定管理层的位置是否适合自己。从技术方面说，当然我很难给一个具体的回答，但是未来我主要感兴趣的两个大方向是社交网络以及大型数据库。一直以来我都对大型数据库兴趣盎然，另外像Facebook这样的社交网站的兴盛也让我觉得从事这方面的业务是不错的选择。请问Yoyodyne公司目前是否在做社交网络的内容？

回答你所知的内容，而非猜测他们想听的。

Answer the question with what you know, not what you think they want to hear.

没有所谓的正确或错误的回答。如果你对领导位置没有兴趣，那么就别想着“我希望担任程序员团队的领导”或者其他可能会让你被贴上懒鬼标签的回答。而且，万一面试官根本不想找一个做领导位置的人呢？

恰当的回答 我希望自己能做一些和今天所应征的职位不同的工作，当然还是在这家公司。我发现自己与其说是一个专才不如说是一个通才，这个特点让我一直不断地追求新技术，也希望每两三年都能迎接全新的挑战。

即使只有很少甚至根本没有工作经历，你也依然得拥有放眼未来的视角。

恰当的回答 因为这是我的第一份全职工作，所以很难讲未来的具体计划是什么。我希望到时候自己在这家公司已经完成一些出色的项目，有过一两级的晋升，并且在工作过程中对系统管理员方面的知识有更深入的了解。

最重要的是，你必须在参加面试之前主动考虑清楚这个5年后的计划，因为如果申请的这份工作根本不符合你的5年计划，那么从事这份工作只会将你引向一条死路。

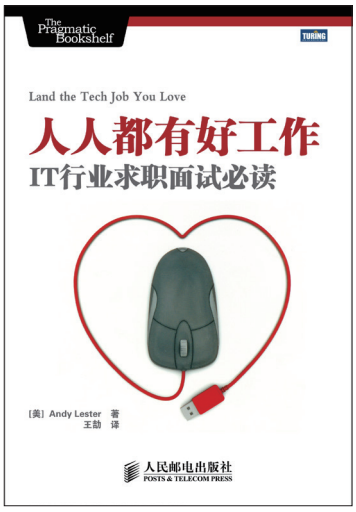
为什么你想加入我们

这根本不应该是个问题，如果你是有选择地参加面试的话。参加面试之前你就应该清楚这个问题的答案。请确保首要理由是和公司有关。陈述完这个理由之后，才可以聊聊关于你个人的考虑。和之前说的一样，公司利益永远应该是你的第一考虑。

糟糕的回答 不知道，只是觉得这份工作很酷。

较弱的回答 因为我一直都很喜欢编程。（这并没有体现这家公司与其他公司之间的区别。）

恰当的回答 能够管理像贵公司那么大的网络系统是促进我个人技能提高的好机会，我相信自己能从中学到很多东西。另外，打从小时候我就对飞机非常喜爱，所以航空业也是我的理想行业。



身处方兴未艾的技术行业，却要为找到心仪的工作而绞尽脑汁，你是否也遇到这样的困惑？作者长期从事招聘技术人员的工作，将其经验与心得记录在此书中，告诉大家：只要方法得当，准备充分，其实人人都能有好工作。《人人都有好工作:IT行业求职面试必读》大量真实的案例，无数正反两面的经验教训，告诉你怎样充分展现自我，避开陷阱，顺利地走过布满荆棘的求职之路。书中介绍的大量求职技巧，不仅仅限于IT行业，对任何正在找工作的人士都极具启发性。本文摘自《人人都有好工作:IT行业求职面试必读》。

为什么你要离开原来的公司

这可是个大大的雷区。请小心脚下每一步。你将要说明的是自己对某个处境不愉快的原因，而不是抱怨。无论如何，千万不能将不愉快都归咎于他人身上。

最直接的恐怖回答　我的老板就是个混蛋，而且那些团队伙伴根本就是一帮无能的白痴。

糟糕的回答　那个，只能说老板和我有一些分歧。倒不是说他做人方面有问题，只是我们之间存在很多冲突。虽然我很不愿意这么说，但是他真的没招到什么好程序员。我很喜欢我的同事们，但是跟一帮成天无所事事的人一起工作的确也是件令人沮丧的事。

温和的用词也无法掩盖你的抱怨行为。
Soft language can't hide a griper or blamer.

第二个例子中使用的都是一些比较温和的用词，但是这么回答同样糟糕。因为你依然在表达自己的不愉快都是由他人造成的。也就是说你是个不折不扣的抱怨者。

恰当的回答　我觉得在Yoyodyne没什么发展前景。我们所做的所有工作项目，都是对于现有系统的维护，没什么新的项目计划。我喜欢每一天的工作都充满挑战。我曾经就这个问题和我的头儿讨论过，但是他也说自己对此无能为力。我在家自学了Ruby，但是公司也没有适合我进步提高的空间。

恰当的回答　我需要找一份离家近的工作。当三年前我刚进那家公司时，一个半小时的上班路程对我来说也不算什么问题。但现在我发现路上花费大量的时间将和家庭生活起冲突，而且路上的花费也越来越高。当然，尽管距离并不是唯一的原因，但发现Yoyodyne公司有如此适合我的工作而且车程也只有15分钟，我真的很兴奋。

恰当的回答　我和之前公司的合同6月底就到期了。我也试图在公司中寻找过

别的工作机会，但是现在似乎没有适合我的技能与背景的职位空缺。

恰当的回答　上一家公司的收入完全不符合我的技能水平和个人背景。我曾经看过许多关于收入的调查，也和一些公布的生活消费指标进行过比对。我喜欢之前的工作，但是上司跟我说公司的薪酬体系很难再做改变。（但请准备好应对面试官的下一个问题："那你一开始怎么会接受这份工作呢？"）

最后的例子倒是一个可以放心地讨论收入问题的样板，因为这只是在正面回答面试官的提问。但请注意千万别自行踏入下一个问题，询问面试官："那么告诉我，这份工作的收入是多少？"

在所有情况下，你的回答都应该只是陈述事实，而不加以任何憎恨或责备的情绪。对于合同到期的情况，你也无需进行太多的解释：只是没有更合适的工作而已。

你有什么问题要问我的吗

除非你们已经聊了好几个小时，不然最好还是准备一些问题。

专题：码农的面试

Facebook面试实录

作者/ Bozhidar Bozhanov
Ontotext架构师。业余时间创 造 了computoser.com，一家独一无二的用电脑创作音乐的网站，上面的每一首音乐都是用算法生成的。同时还创立了一家社交分享工具 网 站welshare.com。 他在StackOverflow上荣誉排名第28位。

在经历了3轮电话面试且被Facebook拒绝之后，我发表了本文，但这不是出于报复心理。事实上，我几个月前就已经打算写这篇文章了。现在言归正传：技术公司（至少包括Google,Facebook,VMWare，毫无疑问还有很多）正在试图寻找最棒的技术人才。（所以他们联系我，询问是否有兴趣与他们一起“探寻工作机遇”）。但是他们如何做到这点呢？

典型的面试（比如电话面试，或者现场面试）充斥着解题过程。有些人将其称为“智力测验”。它们通常不是真实世界的问题，目的是为了核实你的算法技巧和你的计算机科学知识。简单的包括递归，二分查找，基础数据结构（链表，散列表，树）。更复杂的要求红-黑树，迪杰斯特拉（Dijkstra）算法，NP完全性（NP-completeness）知识等。如果在电话里，你在一个共享文件中写下代码。如果在现场--你在白板上写下来。因此，这些难题应该检验你的计算机科学和算法技能。不过让我们退一步从另一个角度来看这幅场景。

- 你在这些面试里做的事情都是你从未在实际生活中做过的：你在不用编译器或调试器的条件下写代码。你在有限的时间内做这些，并且是在别人的监视下或者在线等待中进行。但是暂时把这些放一边，我们假设写无法运行的代码有助于面试。
- 这些智力测验测试的技能都是大多数开发者从来不需要的。大多数人撰写商业软件，它并不需要红-黑树。你最近一次在商业软件中使用递归是什么时候？你最后一次完成这类事情是在大学。并且如果你是一名大一新生，其中的很多问题实际上都很简单，因为你前几天刚作为一项家庭作业把它们做完。但另一方面是，撰写即使像二分查找这样简单的事情也变得有点乏味，因为你只是昨天没有做这件事而

已。当然你将会完成它，但是要花一点时间，以便你能够回忆起来，并且为了确保无误而使用编译器。（顺便说一下，facebook的面试题真的很简单。然而我没有完成得很理想的确是我的失误，也许应该归根于面试紧张，或者只是因为我过去3年来都没有做过此类事情而已。）

- 不管怎么说，测试的技能都是在你日常工作中很少用到的。即使是在像Google和Facebook这样酷的公司里，也仍然存在相当平常的项目，譬如需要编写API代码，维护现有代码等。我不认为在你进公司的第一周会被允许微调搜索引擎，不论你在面试中表现有多棒。
- 在这些面试之前的面试准备是建议的，甚至实际上是必需的。准确的说，面试就像一场大学考试。但这样就令人无语了--你并不希望人们针对你设定的面试标准去做准备。你希望他们都是。。。好的程序员。
- 聚焦于这些计算机科学技能意味着这些公司可能会错失优秀的工程师，即那些仅仅不对底层细节如此感兴趣的人。

顺便说一句，这是我在第一次Facebook电话面试后的一条回复摘录：

在另一方面，我认为在面试高级开发者时涉及计算机科学一年级家庭作业题并不是一个好主意。事实上--大多数人（包括我）在大学之后就没有做过这些了，而且这看起来更像无意义的问题而不是实际编程。

以上列出的问题就是为什么我不喜欢这类面试的原因。很显然是因为我不喜欢解决这类难题。我只是不喜欢它们，它们对我毫无吸引力。你可能会争辩说，除了日常工作以外，为了持续训练算法技能你可以参加（像TopCoder之类的）编程比赛。我会介绍一段自己高中时期的经历。那时有两种学生比赛--一

种就极像这种类型的编程难题--在规定时间内布置了大量这种难题，你必须提交一个覆盖了尽可能多预定义(但是你尚不知道)的测试案例的解决方案。另一个比赛是在家创建一个软件，然后当着评委会的面展示它。我在后者中是第一名，但是在前者中惨极了。为什么？因为我讨厌仅仅为了解题而去解决无用且不切实际的问题。相反我喜欢开发软件。如果我爱好它们的话，我也许会擅长解题。我只是不喜欢而已。这并没有技能的等级之分--一种是能解决复杂算法难题的人(高级)，另一种无法解决的人，因此只能去开发软件(低级)。而是存在两种不同种类的技能，并且在创建优良软件的过程中它们都是有用的。一个撰写底层部分，另一个设计API，架构，部署方案，处理代码抽象。那么，回到“除了日常工作之外我现在做什么”的问题上来--我负责构建过程。我已经在几个我很喜欢的私人项目中工作过。这远远超出了我或许本应该很享受的TopCoder比赛的期望。

不幸的是这些很酷的公司主要雇用TopCoder类型的人。大概说得通的原因是，因为他们有大量的求职申请者，并且他们能够负担得起许多“错误-否定”。但是许多更小的公司采纳了这些面试惯例，也因此他们未能得到最棒的技术人才。我读过的最棒的关于软件工程师面试的文章在几周前刚刚出现。Jeff Atwood建议如何雇用一名程序员，我完全支持他的方法。

而我在面试时的问题就是他们实际上并不验证你是否能做现实的编程工作。并且很明显我的问题是我并不喜欢底层与算法之类的东西，因此我无法为像Google和Facebook这样酷的公司工作。

重要提示：我并没有说你不应该知道计算复杂度是什么，一个

译者 / 旁观者
IT 人，对未来，$$ 和智能感兴趣，喜欢冷眼旁观整个世界的过去、现在和未来。

散列表如何工作，或者如何撰写递归。你应该知道这些，因为那是为了能够撰写优良代码所必需的基础。但是我认为对这些事情过于关注是与日常编程毫不相干的。（话说回来，如果我是一个只能用 PHP 写网站并且认为散列表是一种家具的彻底的蠢蛋，是不可能通过前两轮电话面试的。）

英文原文：My Problem With Your Interviews

读《码农》吐吐槽 还能赚银子！

《码农》电子刊如今慢慢悠悠已经出版了8期，从一开始的好评如潮，到现在的“怎么这么抽象啊”“赶脚内容没有以前好了？”“一下就翻完了，没什么内容啊”……小编表示压力很大，现在的码农读者太难伺候了，明明是本免费杂志，￥%#%&*%@# ￥*@ ￥#@！

但是，一看到好评，盼盼姐还是会热泪盈眶，热血沸腾，各种正能量啊“这么好质量的杂志还免费，天上掉馅饼呀”“希望一直出！每一版都读了，很值得推荐！！”……

所以《码农》还是会一直做下去，但是，是好是坏，您得给个话儿啊！

活动规则：在吐槽贴中留言，选出任何一期中你最喜欢的文章和最不喜欢的文章，即可获得图灵社区银子2两！凡吐槽吐得掷地有声者，加赠银子3两（共5两）！（怎样使用银子兑换图书）

活动时间：本活动长期有效。

云风：一个编程的自由人

云风
程序员，游戏玩家，攀岩爱好者。常写blog，十五年来一直将所思所想通过个人网站http://www.codingnow.com分享。为网易工作十年后，目前创业中。为简悦科技的联合创始人。

云风从三十年前开始编程，对于一个孩子来说，编程和游戏的界限很模糊。在三十年后的今天，云风仍然在编程，他的程序世界从未和游戏分开过。他在网易写《大话西游2》、做3D游戏引擎，离开网易后，他又和朋友联合创办了游戏公司简悦科技。云风相信快乐可以很简单，就像他攀岩不是为了健身一样。我们中有多少人因为游戏而爱上了编程，又有多少人可以一直坚守这种简单的快乐？

属于小伙伴们的年代

> 他不知道我是个小孩，他到我家里找老吴，我妈以为找我爸的，然后就把我爸从单位里面叫回来，说有一个老头找你。其实他是来给我送十块钱的软件注册费的。

你从什么时候开始编程的?

我从小学一年级开始接触电脑。80年代初，电视都没有普及，我爸喜欢编程序。家里就买了一台9寸的黑白电视，专门为了接电脑，因为以前电脑是没有显示器的。所以电视就没看电视节目看，反正也没什么节目可看。那台电脑Z80的CPU，只有16K的内存，比Apple II 都差远了。这台电脑基本上只能写一些很简单的程序。我爸喜欢，这是他的玩具。

我刚上学，我爸就会给我写一些很简单的游戏。他最早给我练习100以内的加减法的程序也是用游戏形式呈现的。我爸给我写一些小游戏玩，我估计写程序是他的爱好，他不会觉得写出来的游戏好玩，但是我觉得游戏也是极好玩的。那时候写高性能的游戏程序就需要使用汇编。而电脑上没有什么开发工具，我爸当时都是手写的汇编，然后拿手册去翻译成机器码，再导到机器里边，当时用的磁带机，程序需要保存在磁带上面。在磁带封壳上用笔标注每段程序的位置，玩的时候进带到那个位置，再连接上电脑读出来。

我小时候喜欢看书，家里的书我都拿来看，无论看不看得懂，反正我觉得有字的就看。有几本写程序的书，我就拿来读，看起来写程序可以帮助我完成学校留下的作业。我妈当时在上成人大学，学校开了个编程的课。他们是晚上上课，白天上班。她看我有兴趣，于是就把我拎到课堂上去听课。当时课堂上就我一个小孩，其他都是大人。很多人可能只是想要个文凭，所以那个课程都不大用心听。但我有兴趣，很认真的听课。老师很高兴，认真批改我交的作业。

真正系统学编程是什么时候?

真正系统学习是读初一的时候。我们中学新开了一个机房，购了一批中华学习机，也就是苹果2在中国的山寨版。学校没有什么老师教，唯一的一个计算机老师是我爸的大学同学，他想让我爸去代课，于是我爸就成了我们初中的计算机老师。编程课程整个学校只开了一个班大概有20、30个人，一个星期2节课，就在学校机房里上课。

当时气氛很好，玩得好的那些小伙伴，他们大部分现在也在写程序。我们有了个小圈子，在一起大家会有共同话题。不知道哪天开始就想自己做游戏玩，因为各种各样的游戏都玩过了，

觉得也没什么，自己也可以做。所谓环境就是有很多同龄的孩子一起都在学这个，大家会比较，会交换自己写的程序，会讨论一下哪里没有写好。没什么资料，有些复杂的问题大家就会去想怎么把它做好。

当时的工具比较缺，不像现在电脑上什么编程工具都没有。所以你可能得顺带连工具一起做，比如说在Apple II上没有好的汇编器，你就得用BASIC自己写一个。那时候也没有互联网，书店里面的书虽然没有现在这么丰富，和编程有关的也有很多。我们没钱把书全买下来，就挑着买，大家买不同的互相借阅。

小时候有没有自己做出一些成型的软件?

我上中学的时候，就开始做一个看电子书的软件。我先在拨号BBS上发。后来《电脑报》每年出合集光盘的时候把它收进去了。当时见过很多共享软件需要注册，我就学样做了个启动几分钟后会弹出来的注册框。说你需要注册才能继续使用，注册费10块钱人民币，并留了地址。结果还真有人给我寄钱。那是我第一次靠写程序赚钱。我用一个本把谁买过这个软件记下来，用平信把注册码寄回去。互联网流行之前，比较流行干这种事情。软件几年间大概卖了100份，也有1000块钱。

这里还有一个有趣的故事。竟然有一个退休的老头，他真的按我在软件里留的地址找到我家里去了，他去到我家里说找老吴，我妈以为找我爸的。打电话把我爸从单位里面叫回家，说有一个老头找你。后来问来问去说是找我啊，我当时在长沙读书。他是来送十块钱的。

上大二的时候，我有几个师兄接了个单：广东佛山有一个图书馆要做一个软件。我跟我发小，中学的朋友就想在假期打份零工。听说师兄们准备8月份去，我们就打算7月份跑过去先去

做了。等那些人来，就没他们什么事了。就这样，事先没联系，一放假我们就直接坐火车跑到佛山去了。

他们的需求很简单，图书馆购置了一台当时很少见的光盘刻录机和扫描仪，想靠这个赚些外快。利用这些把附近居民拍的照片扫描成电子版，做成电子相册。图书馆的馆长说，你们有两个人，如果觉得一个人做这个东西就够了的话，我再给你们点事情。所以我们又帮他们做了一个图书馆的管理系统，两个人各做各的，拿当时比较新潮的 Delphi 做了两个星期。管吃管住之外额外收了几百块钱。我觉得过程挺有意思的，虽然他们有点抠门，但至少比学校的老师好多了。

大学毕业之后干什么去了?

临毕业最后一年我已经在北京一家游戏公司做兼职了。认识了一些朋友，刚毕业的时候，打算做手机平台的游戏。诺基亚刚发布了市面上唯一的一款智能机，投资人觉得手机应用会是未来的一个趋势。我刚毕业的那会，一直在折腾这个事情，我们的切入点就是做一个手机游戏的引擎。我们想做一个开发平台，大家在上面可以做游戏。当时选的是塞班平台。

上大学的时候，没想过去拉投资什么的，就是被人带着玩，开始比较有意思，搞了几个月，兴趣就没有那么大了。我觉得那时候手机上做游戏，只能做贪吃蛇、连连看，这些游戏不符合我的游戏观，我喜欢玩复杂一点的游戏。

在要签投资合同的时候，我作为创始团队成员，需要保证在这个团队若干年。我发现我要是未来几年全做手机游戏，就做不了我想做的东西了。于是在签字之前，说我不干了。大家也挺理解的。

接下来被北京游戏圈的朋友叫去做 web 游戏，那家叫可乐吧的公司。干了半年后没什么兴趣，就离开北京回武汉了。

网易那些年

丁磊最后说，你走也没关系，在网易留一个宿舍，你可以住这里，没问题。我觉得我在网易干得很开心，在这里不受限制。我不觉得公司欠我什么，我也不欠公司什么。

怎么去网易的?

大学快毕业的时候，我曾经帮一家游戏公司做了点东西。他们有一款游戏用了我的游戏引擎，需要我在此基础上再写一个模块。他们问我要多少报酬，我没多想，就按照同学在外面做家教的行情算了一下，我说就按家教1个小时10块钱算就行了。最后给了几百块钱，关系不错。

后来他们公司被网易收购，网易启动了《大话西游》项目，听说我没在工作，于是就继续让我帮他们做东西。我在武汉家中写一些模块，当然再不能按10块钱时薪做了，当时一个月有4000块兼职工资吧。做了段时间，他们觉得沟通太麻烦。丁磊打电话给我说，你不如就来广州网易干，机票帮你买好了，公司也帮你在公司旁边租了房子，过了马路就到办公室。你安心每天来上班，不用打卡。我爸说，你这么整天窝在家里也不像话，我就飞去了广州。

在网易做的第一个项目是什么？

2001年的网易人还不多，就几十个人吧，我感觉那里的人都挺好的，特亲切。跟游戏部门一开始没在一起上班，他们上他们的，我上我的，我只需要完整需求。《大话西游》做了一年多就完成了。当时网易对他们的进度要求挺紧，似乎在按进度走，但我自己觉得质量很有问题，只是抢时间把它做出来。理所当然的，完成以后问题特别多，服务器容易宕机，客户端容易崩溃。

当时也没想过网络游戏后来会怎么发展，网易应该是因为石器时代的火爆而收购这个团队的。但网易当时的思路都是传统游戏开发运营的那一套，比如要找代理商来卖游戏客户端，就像一个有网络功能的传统游戏一样。

《大话西游》第一版不太成功，你接着做《大话西游2》了吗？

刚到网易时，我没把自己看成是游戏团队成员，只是他们让我

做什么，我觉得事情还有点意思，就用心把图形引擎一小块的事情做好。项目结束后，整个项目没做好，觉得很不爽。丁磊那时候也没有信心，好在网易靠短信的业务开始赚钱了。

我记得那段时间，公司附近特别多的酒吧，丁磊喜欢泡酒吧。我就住在旁边，他在酒吧就打电话把我叫出来，也没什么事情，就是喝酒聊天。我记得那时候他觉得游戏没做好，而网易的聊天室服务还挺热闹，他就想把大话西游第一版改成一个图形聊天室。我说这个不好，客户端太大，那时候带宽又小，不行。

我始终觉得很多东西我来做的话，做好也挺容易的。原来的客户端不稳定，我在家里闷头搞了两三个月，重新写了客户端的雏形。弄好之后给大家看，说其实把客户端做稳定也不难。丁磊安排了原来技术部门的叮当来负责游戏项目，叮当从网易技术部门抽调了当时在网易技术最好的几个程序，决定把项目重做一遍。然后我就认认真真开始做这个项目了，这就是《大话西游》第二版。

在2002年，我们其实花了很短的时间，就把它重写完了。游戏的大框架都在，只是前一版程序写得不好。新的版本完成以后不放心，内部测试了很久很久，直到技术上没什么问题，才推了出来。我当时的想法就是在程序这个方面我尽力了，比以前稳定，至少不会崩溃，也没有内存泄露。

你在《大话西游2》上完全实现自己技术上的想法了吗?

我的一些想法也实施了，因为以前客户端很容易崩，我没太想去读老的代码。只是感觉如果按照我的想法做，会稳定一些。也的确好了很多。以前没把项目当成自己的东西做，所以有想法，最多说说，不会用力去推。后来到这个项目，就觉得是自己的项目，尽量把它做好，很多想法之前就考虑过，当时也比较成熟了。

我大学头两年以及中学时代主要是用C语言做开发，到大四开始写C++的程序。后来几年都对C++很有热情，有新的项目开始也就有了新的实践想法的机会。C++里可以玩的东西很多，每接触到一些就会想着如何用到项目里去。当时对语言特别热情，还做过C++的T恤，把C++印到名片上，利用公司的资源开办了一个C++讨论组邮件列表，邀请了国内很多玩C++的程序员参加。

接下来网易开始《梦幻西游》的项目，我参于的一个很大的动力是觉得C++可以换个玩法，能够把新学到的东西用进去，可以换个方法写程序。因为前面一个项目还是比较传统的C++的用法，后来就在《梦幻西游》里面用了当时比较新潮的C++技术。等到2004年《梦幻西游》项目开发结束后，我就逐渐对C++没兴趣了。然后到2005年的时候，在上海开了一次C++的会议，我被邀请去做一个演讲。那时候，我记得听了一些关于C++语言编程的高级玩法，虽然之前自己也花了许多时间去研究，但突然就觉得这些只是一个智力游戏了。

那个时候你写了一本书吧?

04年是我最闲的时候，我写了一本书，到05年的时候，那个书出版，我也到杭州工作，开始忙了。从2005年我正式写博客，最初的想法挺简单：我的书出版了，里面有很多错误的地方，以及想法的变迁，我想利用网络平台，来更正我写过的东西，所以这个博客最早的作用是维护勘误表，接着把新的想法写在上面。

我从1997年开始维护了很多年个人网站，之前也经常写些东西，网站本来就有一些固定的读者。在书出版后，有很多人就因为那本书找到我的博客，读者就更多了。我发现博客有更好的交互性，坚持写下来，一直没有停，写到现在。

你怎么去到杭州的?

2005年的网易，我已经算是干的比较久的员工。一些老员工离开了。我当时的想法是既然项目做完，《大话西游》都有几十万在线，在国内算是最好的一个网络游戏，那么我对自己的工作有了个很好的交代。网易的待遇不错，第一笔的期权收益拿到，似乎日后的生活没有问题。就想离开重新做点别的。

没想好去哪，当时也有其它公司让我过去，第一个找我的是盛大。盛大的陈大年在我读大学时，我们就是网友。他说想搞个研究院，问我有没有兴趣。

我直接跟领导叮当(前网易COO詹钟晖，于2011年5月15日卸任网易首席运营官，离职前是网易管理团队重要成员。他于1999年10月加入网易公司，2001年4月加入在线游戏事业部。2006年5月，詹钟晖被任命为网易联合首席运营官，2009年

3月，詹钟晖担任网易首席运营官，主要负责游戏业务。)说我想换个地方，换个环境生活。盛大的朋友邀请我过去。他就问，去盛大能做啥呢？在网易不能做吗？

当晚丁磊找我喝酒，说你觉得哪里环境比较好，我说杭州感觉不错，然后他就说不如我们在杭州开个分公司吧，我说好啊。丁磊是浙江人，在杭州开分公司的事情估计他也是想了很久，恰巧借我的口提了出来。这个事情网易董事会批的很快，没几天我就一个人到杭州折腾去了。

刚到杭州，我在西湖边租了一个小办公室。一开始的几个人是我在游戏圈子里的朋友，还有过去的同学。接下来就是自己招人，初期人手不够时我还兼职过前台，行政，出纳。

当时在杭州想做什么?

一开始就想做游戏引擎，除了游戏引擎，还想做一款跟市面上的网络游戏不一样的游戏。04、05年是整个网易感觉最好的时期。没有竞争对手，收入不错，自我感觉非常好。刚到杭州时，老板不急，我们也不急，耐着心做就行了，我们想做一个不一样的游戏。我不太喜欢MMO的枯燥模式，觉得很枯燥。2004年《魔兽世界》在美国在Beta测试，我却玩得挺开心的。发现网络游戏也不是那么无趣。

我想做点好玩的游戏，但觉得需要一步一步的来。技术跟不上，就先把技术做好，然后再做好的游戏。我想法很单纯，只想做事。我们只要7、8个人就可以了，人多了也管不过来。后来网易开始把重心移到杭州，前期因为在杭州已经有了我们这个办公室，并注册了分公司，成立杭州研究院也就方便了很多。早

期我们在杭州研究院的筹备工作上做了不少事情，但后来的杭研的实际业务和我那里也没有什么关系。

看起来工作有些琐碎，你一个人能兼顾这些工作和开发吗？

我的确有一些非技术的工作，比如说我需要做公司内部的技术评审。除了杭州的事情，也经常回广州帮广州的项目解决一些问题，频率大约半年一次吧。因为丁磊的用人风格就是信任他熟悉的人。如果有项目出现问题，他觉得谁能搞定，他就把这个人调过去帮帮忙，他对我技术方面解决难题的能力很信任。经常把我叫回去做点这个，做点那个。我也乐意去做。

开发进行了几年后，我们有几个主要的程序员觉得累了，换到别的部门去。每次有人离开，我都很苦恼，因为需要接手那些原本别人负责的模块。后来发现我自己接的事情越来越多，我光接那些别人做的东西，就没有什么精力做新的东西了。每走一个人，我把他们做的东西都看一遍，再接着做，很累。最终整个工作室可能只有我知道所有的数十万行代码都是怎么回事。

在开发上我们摇摆过很多次，又想把游戏做好，又想把引擎做好，结果两件事情都没有做好。技术上的东西也是做了又改，改了又做。一开始我们的想法就是什么都自己做，觉得如果用公开的东西，外挂就容易钻漏进来。比如，如果我们连runtime都自己写的话，可以做出很强的加密壳。我们还可以自己开发脚本语言，不仅可以贴合游戏的需求，还不容易被人分析。这些想法现在看来是很可笑的，但在头一年，我们的确

是在做这些。估计是受公司大气氛的影响，觉得不用着急，什么都可以慢慢来。做着做着就发现，软件大厦不是几个人就可以搭起来的，你不能说自己干就从头自己干，我们当时不仅做了runtime，还做了执行代码的加载器，我们做得特别特别细，干了许多本该是操作系统做的琐碎工作。一年多以后才醒悟精力不应该放在这些上面。

最后你们有成型的产品吗?

我们最后的3D引擎半成品没有对外公开。到2010年的时候，网易的情况已经不一样了，网易的竞争者在08、09年之后开始多起来。但是我们还是比较封闭，没有很快的思路转换，觉得什么事情慢慢踏踏实实做就可以了，也不用管市场需要什么，因为网易有很多赚钱的产品，也不缺我这一个。我当时觉得有些东西做出来，公司内部可以用，我在里面学到的东西，我可以跟公司内部去分享。那段时间做的挺多的就是到广州做内部的分享，给大家讲我们做的东西，网易的技术氛围一直很好。

在杭州开发游戏引擎期间，我们工作室出过一款完整的卡牌游戏，我自己很喜欢，但用户不多，喜欢的人都很喜欢，不喜欢的人似乎也很难吸引他们来玩。公司也不愿意花资源去推这种特别小众市场的游戏。

我在杭州做的东西没有特别成功的，但是有很多技术上的想法，以及对游戏的想法总在和整个公司的同事分享。3D引擎这个东西也很难做，它是需要很多年积累的，现在自己做3D引擎的人也比较少。我们尝试过和广州的小组合作开发游戏，但没有合作成功。最后我们觉得我们小组要想在公司活下去有

发言权，还得自己做游戏。所以到2010年，就开始重新招人自己做游戏。但这个时候，公司的环境对我们的支持已经不如2005年。

因为只是想先让工作室生存下去，所以也没什么高追求，加上之前做的卡牌游戏不被市场接受，我们这次觉得做一个市场认可的游戏就可以了，不打算做什么惊世骇俗的东西。最终开发时间太紧。2010 年底这个游戏就叫停了。其实对于公司来说，确实应该停掉，公司已经有太多项目了，没有特别的理由保留我们这个并不特殊的项目。

有没有想过把这个3D引擎项目开源?

到2010年工作室解散的时候，我想过这事，和叮当说的时候，他说他做不了主，所以我们就退一步在公司内部开源。但毕竟在网易顶多也就1000个程序员，形成不了开源软件良好发展的社区，我们在这个引擎上继续做的工作很多。我跟丁磊说，我们做的东西如果公司不用，不妨把它开源，让大家一起完善。当时我们工作室刚解散，丁磊觉得我心里有想法。他有话直说，说你花公司的钱，做了4年的东西。你没做成功，就觉得你要找回一定的价值。你觉得开源就可以把自己这4年做的东西的价值体现出来。我说我不是这样想的，我只是很单纯觉得技术发展依赖交流分享。这个也是我最后走的原因之一，我觉得和老板有些理念很难达成一致。丁磊是很独断的一个人，公司什么都是他说了算，他认可的东西就很认可，他不认可的东西别人跟他解释也没用。丁磊为人不错，网易整体也很好，尤其是技术氛围，但并不是事事让人满意。

那时候是不是人也走的比较多，他也比较茫然，看问题负面的东西比较多？

网易游戏在前两年离开的核心人员比较多。大话西游的核心团队突然就走掉了一大拨，当时丁磊心里不舒服，突然飞到杭州。我带他去我一个朋友开的小酒吧，喝了一晚上红酒听了一晚上抱怨。最后我那朋友一直在我背后催我说太晚了要打烊了。

那个时候经常被他叫过去在办公室闲聊。我记得有一次他跟我说，中国搞航天什么的，花这么多钱，太浪费了。我说我跟你看法不一样，然后他就一个劲的说这个没用没用，浪费钱。他就觉得搞航天不如养猪[1]，他有他自己的一整套的道理，所以你是很难说服他的。

到2011年的夏天的时候我离开。丁磊最后说，你走了也没关系，在网易大楼里给你留一个宿舍，你想住就住在这里，没问题。我觉得我在网易干得很开心，在这里不受限制。我不觉得公司欠我什么，我也不欠公司什么。

现在网易还有你这种类似于“自由人”的角色吗？

网易的牛人很多，我这种闲人现在估计没有了。我在广州的时候，以及杭州工作室解散后，经常在各个部门，工作室跑来跑去，对跟我没关系的业务发表些意见。比如Twitter正火的时候，我就说我们要不要做一个，我有很多想法。有事没事，我就会找人问说你们部门是不是闲了，我有个点子不错，我们要不要搞一下。因为我自己是闲不下来，就算没事，我也会到一些部门去问，你们最近有没有遇到什么问题，我们一起看一

[1] 网易养猪场坐落于浙江省湖州市安吉县，总面积约1200亩，主要负责人为网易CEO丁磊，目标是建一座网易养猪示范基地，它可以在其他地方复制。网易养猪场要为中国探索“第三代养猪模式”，为中国的养猪业寻找一条全新的路子。

看。我在网易工作 10 年多，大部分时间就是一个人，不用管人，也没人管我。我和谈的来的同事说，最近有些好玩的东西，我们研究一下。如果碰上可以做的事情，可能就搞上一两个星期。

最近听老同事说现在网易跟以前有点不一样了，以前老员工在网易还是比较悠闲的，爱什么时候上班，就什么时候上班。比如我当年就经常下午去上班，半夜离开。现在好象不行了，不管你有没有资历。

自由的程序员

> 每个人都会有自己的挑战，对别人来说很简单，但是对你来说很难，其实你都可以一点一点来完成。

后来怎么想到自己出来创业的?

2011年，我离开网易的时候想的蛮简单，觉得不在网易可能更自由一点。比如我想把我写的东西开源就开源，而在网易是不行的。我就想让自己独立出来玩一玩，想做点事情也好，更随便一点。

叮当比我早几个月离开，一开始我就知道他们几个人在想创业的事情。他们最后才找到我。起初一起合伙的人有做美术的，有做市场的，而做技术的人没有确定。后来这一伙人好些人还是留在网易了，并没有走。

他们一直缺个做技术的合伙人，我单身没有负担，是创业很好

的合伙人选。叮当跟我联系。我本来是想先休息，不过又想想，其实创业的话，能自己说了算也挺好的。我在网易07、08年以后研究的东西，都是做网游的平台，如果我是一个人做着玩，缺乏个平台，绝对是没法继续搞这些东西的。一个产品需要有很多的用户，游戏这东西不可能一个人弄。若是组建起新团队，做这些东西，知识和技能便可以延续下来，这个事情也蛮好的。

你们拉了很多网易的同事出来吗?

我们不太想找网易的人，之前大话西游的团队离开我也看到丁磊挺受伤。我知道这个对他确实打击挺大，以至于后来公司管理风格都变了很多。所以我们出来不太想找网易的人，我觉得就我们几个核心足够了，我们市场有市场的人，做产品有产品的人，做技术我也很有信心。

那你现在公司也成立了两年了，那你觉得达到你的目的了吗?

不到两年，还算达到了。其实我跟叮当干也蛮多波折，最主要是我觉得他们一开始人太多。都是很牛的人，我觉得做事情不能这么多人开圆桌会议来决定事情，所以我当时只有一个要求，我说叮当我跟你干，我听你的，没问题。反正我一直是你下属早习惯了。但是以后做核心决策加上我不应超过3个人。他们也挺犹豫的，毕竟我是后来的。

最后的结果是我们三个人出资成立了现在这家公司，然后再以我们三个人的名义去找了风险投资。最后公司董事会有8个人。

新产品什么时候上线?

我们的MMO产品叫《斗罗大陆》，这原本是起点一部非常有名的网络小说。原计划是今年底上线，但是现在可能进度赶不上。最大的原因是我们代理了一家由成都一家公司开发的产品《狂刃》。代理的产品有很多技术问题，我们抽调了很多人员帮他们去解决问题。不光是技术，还有产品的设计，都有问题。

《狂刃》已经上线。当初开董事会我赞成做这个代理项目的主要原因是可以借机锻炼一下运营团队，况且我们原本也有计划做用户平台，付费支付系统等。把这些用户平台提前做好，我们自己产品上线也会顺利一点。因为游戏最怕的就是刚上线时

各种不稳定因素。大话西游那个团队出去，第一个产品失败遇到的重大问题之一就是他们的用户平台没有经过检验，第一次砸的宣传费几百万，因为第一天用户的接入平台出问题，搞了8个小时玩家不能登陆。这等于一个 BUG 就浪费了几百万的市场费用。我们提前验证一下自己开发的用户平台，代理一个游戏是个很好的机会。《狂刃》的第一天也出了用户登陆故障的事故，但是我们10分钟就搞定了，应急速度很快。估计等我们自己的产品上线的时候，出现这种问题的机会就更少了，也有了应急的经验。

你好象在skynet上面花了好多心思吧?

skynet 是我们服务器的一套框架程序。最初想法是在2010年，我们正在筹备做新的游戏的过程产生了新的想法。由于游戏项目后来迅速停掉，想法也没有实践。到2011年出来了以后，反正要做新项目了，就想想该怎么做。写了半年后尝试做开源。因为一直都想把自己做的东西迅速的放到程序员社区里面，大家都可以用，然后得到反馈，我们再改进。我一直觉得这是可以提高我们在技术圈中的影响力的好方法，同时也可以监督我们把自己的东西质量提高。

最近有很长一段时间，我在手游上面做开发研究，所以也不是全部精力在做这个东西。iOS游戏开发方面，也是我们全部自己做的。不过手游项目没有开源是因为有很多东西做得太快，针对性的代码太多，还没整理好。

我们写的很多东西跟业务没有关系的都尽量开源了。我对现在的同事也是这么说，如果你觉得你的东西值得拿出来，我们都鼓励你去开源。开源有一个好处，你如果能把项目推上正轨，

有很多人会为它出力。就像我们skynet出来以后，一开始都是我一个人在维护，慢慢的发现有一些别的公司的项目也在用。会给你提意见，会推一些代码，发现BUG也会告诉我，慢慢这个社区就起来了。

手游方面你们是不是可以借用一些现有的游戏框架，比如Cocos2d?

也有人问我们为什么不用Cocos2d，我曾考察过一些iOS上面的图形引擎。我从大学开始就做2D游戏，包括后来做《大话西游》、《梦幻西游》。我对这个领域很熟悉，知道2D图形核心部分没有太多工作。所以我觉得没有必要用别人的东西。如果要开发3D游戏，我们会选一个3D引擎的开源东西做，因为自己做过3D引擎，知道这里面代码量特别大，从头做很难。如果想尽早出产品，没有必要把精力花在引擎上面。但是2D的东西，我自己花了一个星期的时间，就把基础的东西做的差不多了，工作量对我来说就是两周的工作时间。如果去用现成了的2D 引擎，可能我也需要两周时间去熟悉。而自己写的引擎，可以更清楚如何优化。手机游戏性能优化很重要，有点像20年前的感觉。机器性能都很差，你必须把优化做好一点。这方面我觉得自己还是比较擅长的。有很多东西新鲜有趣，比如在手机上考虑怎么省电等等。如果用现成的东西，还会放弃这些有趣的过程，我选择了自己来做。

你刚刚一直在提你觉得好玩的游戏，还有你的游戏观，你觉得好玩的游戏应该是什么样的?

我们现在做的3D网游是按照魔兽世界的模式在做，我们不敢

说做的比它好，因为我自己也是魔兽世界的玩家，我觉得做得比它好很难，但是我们是走这种模式，庞大的世界观，丰富的情节，有很多很多的细节，然后让你觉得是一个世界。

单纯个人选择来说，我觉得游戏系统越复杂，越有挑战性，我就觉得越好。比如说我原来不怎么玩手机游戏，我觉得手机游戏太简单。过年吸引我玩手游的第一个游戏是COC（《部落战争》），我突然觉得挺好玩，有人玩这个游戏就觉得不花钱就玩不下去了。而我反而觉得不花钱才是乐趣，有挑战。我没花钱但玩得挺嗨的。

我玩游戏对图象声效没什么要求，只需要它系统好玩。我也玩一些用字符拼出来的抽象游戏，比如矮人要塞这样的，画面上只有一些符号，但它有一个很复杂的游戏机制。还有一些策略游戏，通常会选最高难度，要求一步一步的精心策划，不准犯一点错误。可能我会不停的失败，但是我觉得失败也是一种乐趣，只要最终能成功就挺开心。

我玩游戏的口味跟大多数普通的玩家可能真不一样，所以我们自己的游戏，我不提太多的意见，交给策划去做。而我喜欢的东西，市场可能不接受。

你在业余时间经常攀岩？玩了多少年了？

我04年开始玩攀岩，05年就停了，在2011年工作闲下来才捡回来。我的身体素质不太适合玩这个，因为我体力不好，力量不怎么样，个子太高，韧带又不太好。攀岩比较适合1米7几的人玩，亚洲人超过1米8，力量就不太跟的上。虽然攀岩运动在20出头是黄金年龄，但玩到60、70岁绝对没问题。这不

是项跟人较劲的运动，只是不断的挑战自己而已。这个东西，自己觉得开心就好。不为什么，甚至不是为了锻炼身体，纯粹是喜欢。玩攀岩也不占用什么时间，每次1个小时，一周三次不过3个小时。当然偶尔会开车出去玩。花上一个周末在野外。

攀岩是一项孤独的运动，就是让你觉得定期有事情做，让你觉得有很多挑战要去完成，每个人都会有自己的挑战，对别人来说很简单，对你来说很难，你可以一点一点来完成。你可以去各个地方，一直爬下去。玩攀岩的圈子，除了大学生刚毕业的以外，我认识圈子里面坚持下来的人，就是玩了好多年的人，很多岩友都是三四十岁。我现在是34，至少10年内我不会觉得玩得吃力。去野外爬，你经常会见到五六十岁的人挂在岩壁上。它不是一个纯年轻人做的事情。

今天完成不了的线路，明天可能也完成不了，但是你坚持，过一段时间，突然开窍了，就会发现很简单。攀岩是用脑子玩的，有的东西说不清，要自己体会。突然之间，原来很难的东西变简单了，然后你就会找下一个目标，可以爬更难的线路。也不一定是需要越爬越难，还可以尝试各种风格的，不同的岩壁。

你作为程序员出来创业，有什么切身感受？要是其他码农有这样的机会的话，鼓励他们出来创业吗？

我很满意现在的情况。虽然我们公司还没有正收益，我们还在花钱。现在公司最好的一点是我们可以很好的分工合作，大家做各自擅长的事情。就像我现在写程序，不管人。做开发的阶段，我只要把我的事情做好就行了。每个人把擅长的东西做好就行了。如果想自己什么都做，那就会很累。我自己的负面经

验就是在杭州干了几年，什么都管，觉得自己什么都能做。当时想，只要其他人能做的事情，肯学肯适应，你都能做。结果是你没有那么多精力。还是把自己喜欢做的事情做好。你不喜欢、不擅长的事情，有人却喜欢擅长，交给他们就好了。

组建创业公司，做事的规则很重要。我觉得与人合作，矛盾迟早会有的。一开始就应该把决策机制和遇到矛盾大家怎么来协调定好。把坏的事情事先想好。不能依靠朋友间的友情来处理问题。

践　行

一位老码农：

离岸找工作你需要知道的事

作者 / 肖鹏
9年丰富编程经验，前ThoughtWorks高级软件工程师，之前曾在摩托罗拉任高级工程师。经常在QCon, AgileChina等大会上发表精彩演讲。个人博客xiaopeng.me.

这篇文章主要是面向Senior Developer，对其他角色也许有些帮助。另外，这篇文章又主要是针对从大陆来澳洲的同学的，对于在大陆本地找工作，或者本来就在澳洲的同学帮助或许也不会那么大。

面试这件事

首先，面试不是水平考试，对于大量招人的公司，在招毕业生的时候（比如华为校招），可能真有点水平考试的味道。但是对Senior Developer的面试更像是相亲。你看对了眼什么都好说，看不对眼怎么都很难办。

这里面倒不是存在什么黑幕或者不公平之类的东西，而是你想表达的、你表达出来的和面试官接收到的，三者之间差别很大。面试官也是主观判断，可能一念之差就被pass了。所以，对每场面试都要认真对待，对面试的结果要看的开。

说起来简单，做起来却不是那么简单的。人难免会在面试之后后悔。比如，他怎么就问到Java Bean的Scope这么细节的问题了呢？不知道这个对于成为一个优秀的Developer有很大影响吗？很难说。

认识到了这一点，投简历的时候就不要看到一个perfect match的job description就以为自己稳拿了。在可控的范围内多投几个简历没有什么坏处。反正也不费电:)

好了接下来开始正题。

离岸找工作

如果说找工作像是相亲，那么离岸找工作就像是不见面就想订亲一样。不是不可能，更加依赖缘分。这里我需要提醒一点，离岸找工作不是投的越多越好。原因是，如果你投了A公司的职位，A公司HR一看你不在境内，遂决定pass掉。不好意思，很有可能你就进了黑名单了，登陆以后再投可能都没有机会了（未经证实）。你说冤不冤？

离岸找工作就算是最后找到了也不是一定比登陆后再找好。这边Senior Developer的年薪一般在10w左右，一个月大概到手六七千，一周就是一两千。那么假设离岸找工作花了2个月，登陆找工作花了一个月。从经济上也不如登陆找工作更合适。

为啥要离岸找工作

我们想离岸找工作的理由其实是非常充分的，过来之后很多事情都依赖于工作。工作定了才知道在哪儿租房合适，然后才能往这边寄东西，买家具等等。如果这边有朋友，这些就会简单很多。比如，我来到之后先住在朋友这边，没有房租，开销很小，所以心态要好很多。试想如果是自己短租的房子，一周可能要几百块澳币，出行什么的还不方便，然后办理各种事务都没头绪。确实很折磨人。所以，前面说的，也要一分为二的看，别一棵树上吊死。我最终采取的策略是——离岸找悉尼的工作，找不到登陆墨尔本。

在哪儿找

Seek.com.au

这是澳洲第一大的招聘网站。每天墨尔本都会有几十个Developer的职位出现在Seek上，虽然存在一些重复，但是总的来说，seek上仍然是最大的职位来源。

LinkedIn

LinkedIn的作用似乎越来越大，给我的感觉LinkedIn上的职位水分更少，中介也要少一些。下面会讲LinkedIn上的自我营销。

中介

中介通常都是在Seek上发职位的，其实Seek上绝大部分职位都是中介发的，这也是找工作的一个主流途径，对于举目无亲的新移民来说，中介更有可能成为我们的最重要的帮手。这里要注意一些潜规则。一个中介通常不会让你同时持有两个以上的面试机会。原因是，中介通常在给他的客户（雇主）推荐人的时候，队列里面不能超过两个人。而且对于雇主来说，面试也是有成本的，如果一个中介推荐的人自己面过了，但是总是最终没来，这个中介也没法混了。

所以（所以后面的总是比较重要的结论），你可以多找几个中介。跟中介也有点像是相亲，但是好的中介和差的中介差别很大。我跟好几个中介打过交道，Edward无疑是其中最好的——

负责任(帮助我修改简历、设置期望), 反应迅速(每个面试之后必然快速反馈)。不论是离岸还是境内，大家都可以跟他联系。

朋友推荐

找朋友推荐要注意最好是有已知的空缺职位，否则只是转发一下简历帮助就不会很大。这个我的经验不多。总的来说，虽然朋友推荐本身很高效，但是不能过度依赖这个途径。

写简历

在国内很多Senior的同学早就不写简历了，因为这些人光等着猎头挖都应付不过来。简历主要还是敲门砖嘛。但是我们到澳洲来就不一样了，因为基本上没有人认识你。

一般来说工作经验超过6年的简历写到四五页甚至更多才是正常的，像我一开始的两页式简历被Edward详细点评。一般来说简历里面要包含:

1. 项目名称，起止时间
2. Technologies
3. Responsiblity
4. Achievements

你想想，每个项目写一个，写个四五页实在是很正常。我的简历就不贴了，大家可以去搜一下澳洲的简历模板。

Tips

- 不要等着别人去发掘。你的第一关往往是中介或者HR。这些人主要是看关键字。Edward跟我讲，你的简历上只写了JavaScript，在HR眼里面可能就是这个人没有HTML和CSS经验。虽然在我们看来，写了JavaScript怎么可能不会HTML和CSS呢。
- Scala我只是看了一本书，我还要不要写？写！你反正也不会投Scala的职位(假设)，写上Scala, Ruby, RoR, AngularJs这些东西说明咱是多么认真的跟踪新的技术。
- Github的帐号要不要写。如果有尽量写。如果你几个月之后才开始找工作，从现在开始在Github上搭个博客，参与参与开源项目对于找工作是很有帮助的。请放弃作弊的想法，但是如果能成为你一个新的开始，也真的不错呢！
- LinkedIn。LinkedIn上的信息要及时更新，认识我的人可以上去看看我的LinkedIn。我的Skill & Expertise比大部分同事都要多，其实是我有一次给很多同事发了邮件，让大家帮我endorse我的技能，反正这也不是作弊。另外，我后面的几场重要面试之前，面试官都到LinkedIn上去看我的简历了。

面试

我面的基本上都是Senior Java Developer的职位。技术方面这个职位必然会被问道一些高级的话题总结来说包括:

技术面试

- Java基础知识中的细节。比如一个ASCII、UTF-8字符占

几个byte（题目记得不是很清楚了）之类的。

- Java稍微高级的设计问题。比如重载equal的时候还要重载哪个方法，为啥。
- Design Patterns, SOLID。
- 多线程。
- 垃圾回收。
- SQL。
- Spring & Hibernate。
- 对于一些Buzz Word的理解，比如AOP, SOA, Cloud, SOAP, REST等等。

其中6，对于DB不是很灵通的同学不必要过于担心，掌握一下稍微复杂一点的关键字比如join，明白left join和right join是怎么回事就好了。不会有公司让你写一个完全正确的SQL的。而且，我觉得即使你连join也搞不明白似乎问题也不大，直接说不会，人家知道你DB不灵通就好了。当然，你最好同时说一下，自己还是能看懂SQL的，平时也使用google / stackoverflow来解决问题。

其中7只有一个公司问到了，但是我觉得还是稍微准备一下比较好。因为，你说自己用过Spring，人家随口问一句Spring里面常用的Annotation有哪些，实在是不过分吧。

感　想

这一两个月下来，其实我自己都补了很多姿势。我说的这些点，你加上interview questions关键字一搜，能找到不少很好的资料。不要像我之前那样小看这些知识。

第8个，你可以先准备我说的这几个。这个块儿就是考个知识面，如果你说的很漂亮那么就会有很大加分。一个捷径是上StackOverflow上搜一下，别人的解释。

行为面试

这个容易犯的错误是临时想不到合适的例子。问的问题其实很集中，看一下下面的列表，用英语自己把答案准备一下就可以了。

Initiative

1. Tell me about any ideas or processes that you have implemented in your current job. ? Have you ever suggested a new way to improve your team/project’ s performance? Problem Solving
2. Tell me about a complex problem you have solved. Walk me through the process you took. ? Tell me about a potential problem you have prevented from occurring.

Leadership Skills

1. How do you go about allocating work for your staff? Can you give me an example?
2. Tell me about a time when you have provided coaching to one of your staff.
3. Tell me about a time when you have had staff members resist your leadership. What did you do to overcome this?

Decision Making

1. Tell me about a recent decision you have made in your role. Walk me through your thought processes.
2. What is the most difficult decision you have made in your current role?

Team Skills

1. Tell me about a time when you had to work with a team of people you did not know.
2. Tell me about a specific situation where you were able to help out a team member or colleague.

Project Management

1. Tell me about a project you have managed recently. Walk me through your planning and tracking process.
2. Tell me about a project you managed that didn’ t go to plan.

Analysis Skills

1. Tell me about a project where you were asked to gather and evaluate complex information.
2. Tell me about a time when you were asked to make a recommendation based on statistical information.

Time Management

1. Tell me about a specific situation when you managed conflicting priorities. What did you do?
2. Tell me about a time when the project you were working on seemed in danger of missing a deadline. What did you do?

Building Rapport

Tell me about a time when you have had to deal with a difficult customer/colleague. What happened? What was difficult about them?

Quick Learner

1. Tell me about a time in your current role when you had to learn new skills quickly. Negotiation Skills
2. Tell me about a difficult negotiation that you had to handle.

你该特别注意的几件事

Cover Letter

上次说简历的时候忘了Cover Letter了。据说，在澳洲必须要有Cover Letter的，但是说实话Cover Letter到底能起到多大的作用，我不知道，也没有得到证实。我用了两个不同的Cover Letter的模板：一个是通用式，这里面就是简单的自我介绍，针对要求稍微加重一下相关技术或者项目管理方面的经验介绍，最后是从同事和客户那里收到的反馈；另一个是针对性的，如果我发现一个Position非常合适，而且其中具体的要求match的也非常好，我就会画一个表格，一一对应地列出哥的那些经验和成绩是针对你这个工作的。总的来说，效果未经证实。

市场

说一下市场吧。从我的面试经验来看市场还是不错的，据中介说现在行情不太好。这都是相对的。我说几个例子，大家可见一斑。我是在飞机上收到面试邀请的。A公司从LinkedIn上找到的我，然后到了就安排面试第二周拿到口头Offer。虽然这个Offer后来因为财年盘点headcount被hold（情况复杂不

解释），但是至少说明他们也是在正经招人。由于A公司的口头Offer在手，耽误了将近两周的时间。差不多从7月1号重新开始，陆续收到了R公司、AR公司、AC公司、OD公司的面试邀请，还有几家其他中介中间也有联系。特别是AC和OD，我告知他们需要在周五之前完成面试的时候也都积极安排加速推进。

面试气氛

再说面试的气氛。面试气氛总体上是非常棒的。举例来说，我已经一年没有用C#了，面试的时候有一个题目是关于命名空间访问权限的，面试官主动跳过，说这是个很tricky的题目。C#中的多线程更是忘的差不多了，我说能不能说一下JAVA中的实现，面试官也欣然答应。写SQL的时候，面试官主动说，就是看看你的思路和想法，不用试图写出完整的SQL语句。也有一些稍微tough一点，总得来说跟国内的情况差不多吧。话说，其实我也没怎么在国内面过试:)

英语

英语。英语估计是大家很担心的吧。英语不好肯定是个劣势，但是也不至于糟糕到哪儿去。因为这个地方毕竟是一个移民国家，面试你的人大多都不是本地人。而且，我拿到这个Lead的角色的面试中，我的Manager问我你觉得最大的挑战是什么，我说当前肯定是英语。他说，你的英语虽然不太流利，但是我基本上能听懂，咱们沟通应该不会有很大问题。我的意思是，别太因为自己的英语降低了期望。跟老外合作过的童鞋们，如果在工作中跟老外打交道不会太怵头，在面试中基本上英语不会造成太大的障碍。注意，在面试中，我的经验是积极猜测，

对于面试官自言自语，或者说给你听但是并不需要你回应的听个大概就行了。如果是直接问你，宁愿让他重复或者放慢速度也别瞎蒙。话说回来，你有的点因为英语不过关，没有表现出来是另一个话题啦。

项目经验

项目经验。关于项目经验的部分，其实是最容易准备的。要是不准备现场说肯定丢分。不信你可以尝试现在回答一下：请介绍一下你做过的最复杂的项目以及它的架构。不准备的话，用中文恐怕都不会说的很好。涉及到项目经验的部分还经常问到你有没有影响客户决策的经历。这个我以前倒是见到的不多。

谈Offer

接下来就是谈Offer了。刚过来的时候，心态容易着急。恨不得有个Offer赶紧要了，给钱就行。我不想做那种站着说话不腰疼的事情。如果我的两个Offer不是同时拿到，我也没有本钱去讨价还价。当我还有后面的面试的时候，中介跟我说能不能今天确定，如果今天确定的话我可以去跟雇主谈看能不能多给一点，明天再谈可能就没有什么理由了。我当时差点一狠心拒掉后面的面试。幸亏我头脑迅速清醒过来。注意，永远都是你手上机会越多，主动权越大。特别是当你对后面的面试比较有信心的时候，要顶得住压力。一个公司口头Offer出来是不会轻易放弃的，这里面有很多成本，特别是Offer一般都是几个部门的老板签字Approve的，不会因为晚一天就没了。Offer的钱能往上谈的。一般来说通过中介拿到的Offer，雇主是不会问你要多少钱的。中介告诉你一个价格，然后你可以跟

他讨价还价。主要的本钱，就是别的Offer。特别是当你的其他Offer不是通过这个中介拿到的话，他就会非常积极的帮你去谈。当然，谈到多少主要还是看你的面试表现啦。工资又两种谈法，一种是package，一种是base+super+(optional bonous)。Base大概是Junior年薪9万，Senior年薪10万，Tech Lead年薪11万。

Contractor

你想做Contractor吗？刚过来的时候，我们可能都会觉得permanent的职位更踏实。就我现在的知识和认识来看，Contractor可能是个更好的选择。首先就是钱多。Contractor，如果是通过中介的可能是在$600~$750每天。这个价格相当于$160,000的package（还帮你扣掉了4周的无薪假期），比一般的Senior Dev $110,000多了不是一点半点。Contractor也不是说你做了3个月就必须走人了，一般来说大家还是希望接着雇佣这个人的。就算是中间空出1个月找工作也无所谓吧。而且，你也不用担心交税和Super什么的，中介公司可以帮你做这些。好像（非常不确定）还有一种是自己找的，价格还要更高。信息不确定我也不乱说了。其次，没有了。

Location

位置。我知道你离岸的时候很少考虑这么细节的事情。但是当你真正开始找了，就得考虑了。好处是你仍然不必太过担心。无非是两个地方，一个是City，一个是非City。City呢，如果是全家过来就做好40分钟到1个小时的单程准备；如果是非City，就幸福了。一般就在附近找个房子，不要下班太早哈。

盲人程序员的编程生涯

本文来自于Stackoverflow上的一个问题，下面确实有很多盲人程序员作出回答，笔者感触颇深，故整理成文。

具体技术细节不必深究，主要了解一下他们的工作状态，以期让更多的朋友关注残障人士的生活与工作。

原问题大致如下：

视力应是大多数程序员理所当然应有的感官之一，大多数程序员都会花大量的时间盯着显示器（尤其是当他们处于巅峰状态时），不过我知道还存在很多的盲人程序员（比如目前供职于Google的T.V. Raman）。

作者/姬光
@姬小光，前端开发一枚，目前就职于腾讯电商控股用户体验设计部，《精彩绝伦的 CSS》译者。博客: http://44ux.com。图灵社区 ID:姬光

如果你是个盲人(或者视力衰退严重)，那么你会怎样设置你的开发环境来协助你编程呢？

下面整理一些精彩回答:

From Jared ：

我是一个完全失明的大学生，已经参加过几个编程实习，所以我的回答将基于这些经验。我使用Windows xp作为操作系统，使用Jaws的合成语音来阅读屏幕上的内容。对于Java编程，我使用eclipse，因为它是一个全功能可访问的IDE。

以我的经验来看，java编程中使用SWT作为GUI工具的可访问性要比使用Swing好一些，这就是为什么我放弃netbeans的原因。对于.net编程，我使用visual studio 2005，因为这是我实习时使用的标准版本，而且通过使用Jaws和一系列脚本(比如简化表单设计的脚本)，可以使操作变得非常简单。

对于C和C++编程，我使用cygwin加上gcc作为编译器，使用emacs或者vim作为编辑器，视具体工作而定。我的大部分实习涉及到Z/OS的编程，我使用一个登录会话通过Cygwin访问USS子系统，并且使用C3270作为我的3270模拟器来访问主框架的ISPF部分。

我一般依赖于带点字(Braille ：点字/盲文)显示(注：下图为一种点字设备)的合成语音。我发现在标点很重要且比较复杂的情况下使用点字显示的话，工作效率会更高。一个例子就是当有很多if声明很许多嵌套的括号和JCL时，标点就显得非常重要了。

更新

我在玩cygwin下的Emacspeak http://emacspeak.sourceforge.net，我不确定这个作为编程使用的编辑器是否有用，因为它有时好像反应迟钝，不过我还没有查看任何配置选项。

From Saqib

我是个盲人，我已经使用Windows、mac、Linux和DOS编程大约13年了，使用的语言从C/C++、Python、Java、C#到各种小众语言。尽管原问题是关于环境配置的，不过我认为最好应该从盲人如何使用电脑开始回答。

有些人使用有声环境，例如T. V. Raman楼上提到的Emacspeak环境。迄今为止更通用的解决办法是使屏幕阅读器运行在后台，并监视操作系统的活动，然后通过合成语音或者物理点字显示(一般一次显示20至80个字符)提醒用户。这就意味着盲人可以使用任何可访问的应用程序了。

那么，我自己这些日子在用Visual Studio 2008，只做了很少的更改。我关掉了一些诸如在输入时显示错误等功能，因为我发现这个功能很容易使你分心。在加入微软行列之前，我的所有开发工作都在类似记事本(Notepad)这样的标准的文本编辑器中完成，所以仍然没有任何自定设置。

让屏幕阅读器读出缩进也是可能的，我自己不用这个功能，因为Visual Studio会处理这些，并且C#中是用大括号的。但是在像Python这样空格很关键的语言中就很重要了。最后，Emacspeak可以使用不同的声音/音高来指出语法的不同部分(关键字、注释、标识符等)。

From Manish

我是个盲人，已经编程大约12年了。目前我是Sapient Corporation（一个剑桥的顾问公司，专注基于Web的和胖客户端的企业解决方案）的高级架构师。我使用几个屏幕阅读器，不过大部分是在Windows上使用的Jaws和NVDA。

我大部分在微软平台上工作，使用visual studio作为开发环境。我也使用像MS Sql企业版以及其他的数据库操作工具、网络监控工具等。我曾尝试花些时间使用emacspeak，不过由于我的工作大部分基于MS平台，所以不会在那上面花太多时间。我也花过几年在Linux上使用C++，大部分时候是在windows上使用记事本或者visual studio完成所有的编码工作，然后用samba与linux环境共享文件。我也使用borland C来做些小实验。最近在玩python，它普遍被认为是对盲人用户最不友好的程序语言，因为它使用缩进作为嵌套机制。说到这里，最流行的开源屏幕阅读器NVDA就完全是用python写的，而且这个项目的一些贡献者本身就是盲人。一个很有意思的问题就是，我经常被问到，作为一个架构师我是如何处理各种图表的（UML、viso以及rational rose等等）。Visio可能是最易操作的图表工具了，我还可以写jaws脚本来为我读出rational rose图表。我曾用过一个叫T-dub（technical diagram understanding for the blind）的工具来处理UML 2.0图表，它是一些德国大学开发出来的。我还用过一个基于java的非常丑陋的工具叫做magic draw来处理模型驱动（model-driven）的开发工作，并且作为androMDA项目的一个提交者（commiter）协助开发了从UML模型生成.Net代码的生成器。

总体来说，我发现我的自力更生激励了整个团队。例如，当一个图表对沟通/文档化一个设计非常重要时，实际的设计过程牵扯到大量的思考和头脑风暴，并且当设计定稿时，你的一个队友可以帮你快速将其整理为一个干净整齐的图片。人们通常将这种情况视为缺乏独立性或能力，而我却认为这是纯正的相互依存，因为我很确定那个队友靠他/她自己或着几人轮流都不可能想出那样的设计，如果我依赖他将设计文档化也是如此。我遇到的大部分障碍都是基于工具的不可访问的问题。例如所有的oracle的产品都鼓吹其访问性好多年了（鄙视他们），但在团队环境中却只搞了个在屏幕阅读器和自定义脚本之上的额外的防御层。

From Edward Kmett

我为大底特律盲人辅导会（Greater Detroit Society for the Blind）工作三年了，运营一个为盲人访问量身定制的BBS，并且和很多盲人用户一起工作探寻如何能够更好地满足他们的需求，并帮助新的盲人用户训练他们使用软件和硬件。如果不出意外的话，我至少学会了阅读点字（盲文）来防止万一我遇到跟他们一样的处境。

大部分盲人计算机使用者及程序员使用类似屏幕阅读器的东西。Jaws在某种程度上是最受欢迎的。幸运地，现今的大多数应用程序都提供了某种形式的残障人士访问方式。你可能需要将你的环境稍微调整一下，让它少说一些，比如，可以考虑禁止Visual Studio中的智能感知（Intellisense）。

点字显示设备就不那么常用了，相比之下也贵很多，它可以显示40或80列文本，而且可以用在当精确定位/标点很重要的场合。而屏幕阅读器可以配置成快速读出标点，很多人发现它

容易令人分心，其实通过它可以很容易找到适合自己的方式。Jaws可以配置成显示驱动的，因此你无法兼顾可访问性应用程序。

同时，很多法律上的盲人用户仍然有一点遗留的视力。使用高对比度的背景和放大功能可以帮助很多这样的用户。

在Windows中使用ToggleKeys可以在你不小心敲击了"caps lock"、"num lock"、"scroll lock"等健时让你能够听到。

我知道至少有一个Haskell（注：一种纯函数式编程语言）程序员使用屏幕阅读器，并且不使用Haskell的布局规则直接编程，并且不使用非惯用（non-idiomatic）的选项，而是用支持{;}的。因为它不会使阅读器读出大量的标点，而且还得计算出Haskell布局规则中精确的缩进，这样他就不会太过分心。同样的，我还听说一些盲人程序员在写Python的时候发些牢骚。

最终，你还是要学会发挥自己的长处。

From Kyle Burton

可以从Blinux项目开始：http://leb.net/blinux/

这个项目描述了如何获得Emacspeak（带文本阅读的编辑器）并且还有许多其他资源。

我曾经跟这样的一个人工作，他的视力导致他不能使用显示器，但他使用屏幕阅读器软件并花费大量时间使用基于文本的应用程序和shell也工作得很好。

维基百科上有个屏幕阅读器得列表，也可以从这里开始：

http://en.wikipedia.org/wiki/Listofscreen_readers

From ifwzh

我是来自中国北京的一个研究生，我是计算机科学专业的并且大部分工作是编程。我天生弱视，需要使用放大工具才能看清屏幕上的文字。我在windows上使用微软的放大镜工具，在linux上使用compiz的放大插件。我一般将工具设置成放大原始字体的三倍。对于我来说，放大工具就够了，主要问题是速度，我需要移动鼠标来确保指针跟随我所看到的文本，微软的放大镜提供了一个选项”自动跟随文本编辑光标“，这可以让我在编辑文本或编码时摆脱频繁移动鼠标的困扰。但是这招并不总是管用，因为编辑软件或者IDE可能不支持。linux上的放大工具比较难用。KDE中自带的KMag拥有令人恐怖的刷新率，让我的眼睛很不舒服，我现在使用的compiz的放大插件还可以，不过没有自动聚焦功能。对我来说，iOS提供了十分完美的全屏放大解决方案，尤其是ipad的9.7英寸显示屏。它们的自动聚焦就没必要了，因为我很少用它们编辑或编码。安卓（Android）系统只提供了非常少的可访问性功能，只有像摇动反馈这样的功能，对我根本没用。在安卓上没有什么放大工具，更别提像iOS上这种全屏放大的功能了。我以前研究Qt，希望做一个linux上好用的放大工具，甚至是安卓上的，不过很难有什么进展。

通过以上一些回答，我们可以大致了解一些盲人程序员每天的工作状况，我很惊讶居然有这么多盲人程序员，对于我们来说，可能蒙上眼睛就什么也干不了了，真的很敬佩这些同仁们！

我也希望能够通过本文让更多Web开发者更加关注网站的可用性及可访问性问题，更多地关注残障人士。

八++

写代码如坐禅:你是哪一类程序员

作者/ Jiri Novotny
自由开发者，Swift To-Do List创建人。

当编译占用你时间时你会怎么做。不只是编译，即使是在等待任何短暂的计算机操作结束，这段时间你会干些什么？

和你的工作日休息比起来这点时间是微不足道的，但是总的来说它还是能对你的生产率和幸福感感带来巨大帮助的。

顺便说一下，这篇文章内容不是只针对开发者和程序的。它对任何使用电脑的聪明人都有效。后面还会附上图片。现在让我们开始吧！

为什么要写这篇文章

我最近开始在我的工作习惯里使用某种时间管理技巧来提升我的生产率，减小压力，并帮助我的身体和大脑得到休息。我基本上想要在不被打扰的100%注意力集中的一两个小时里工作，然后就是20到30分钟的休息。

然而，我几乎立刻就进入了一个巨大的问题里：当我正在编译或部署什么东西时，我会自动地打开邮件客户端，facebook，新闻阅读器，新闻站点中的某一个或几个。这是一个坏习惯。它很难改掉。它扰乱了我本来专注的注意力。

所以我决定搜索相关资料。在 StackExchange 有一个关于“编译时应该做什么”的论坛。投票最多的答案是“减少编译时间”一类的。然而，这些答案并不能解决更多的普遍问题：在电脑上工作时总是有些事项是需要时间来等待的。

与之不同的，论坛上的人多半建议那些时间可以做些什么（类似收发邮件或是看新闻），这是一个差劲的想法。一个好的建议是这个——“同一时间干多件事是不好的”。我赞同这个观点。但其他人呢？不认为？好的。这个答案令人失望。所以在这篇文章里，我将深入探索这个问题并且向你展示最理想的方法。

两种类型的程序员

有两种极端类型的程序员——“禅宗的程序员”，我们称为心如止水的程序员，还有“不能克制分心的程序员”，我们称为心绪不定的程序员。通常认为两者都人数不多，并处于钟形分布曲线的两端。果真如此？

错。实际上，心绪不定的程序员是难以置信地普遍存在着。

图片的效果要比文字有用，所以我现在用图片来说明两种程序员的精神世界的状态。

心如止水的程序员

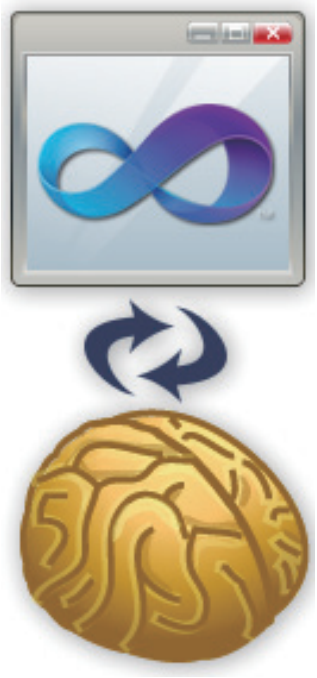
心如止水的程序员会怎么做？

他写代码。这是他做的唯一的事情，这也说明了一切。也许更重要的问题是什么事情他没有做：他心无旁骛，专注于代码的

修改和编写。一到两个小时的纯代码时间，然后才是20 - 30分钟的休息，通常是离开电脑。之后他会继续开始写代码。（以上所说的时间只是一个例子。他可以用其它方式安排时间 - 不过他总是能平衡并搞定长时间的生产率。）

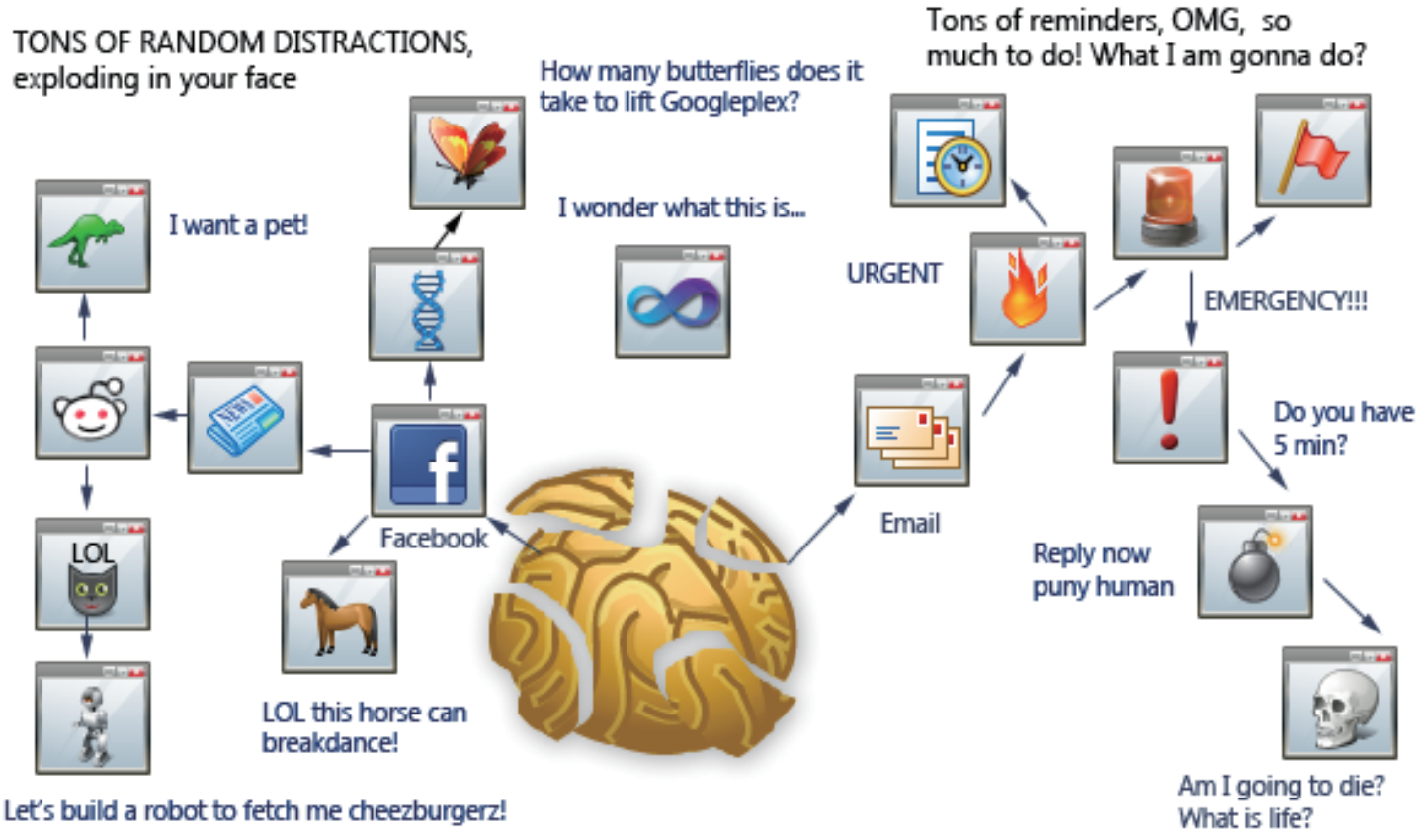
他的思维像平静的水缓慢地流动。当然它不像那种恒久不变的凝固的冰。

这就是心如止水的程序员精神世界看起来的样子：



心绪不定的程序员

即使你的精神世界现在看起来像一个心如止水的程序员，只要你在编译的时候打开了Facebook或者检查了你的邮件还是做了其它什么事情。几分钟之后它就会像这样：



女士们先生们，这就是你检查你的邮件和打开Facebook后的状况。如果你还拥有一堆实时通知，即时通讯，还要去检查你的RSS源和推特，结果就是大脑像上面图中变成碎片的灾难一样。

你可能还没有完全从这张图恢复过来。不在特定的某天，任何时候都有可能发生。即使你把精神世界所有无用的东西最小化到任务栏，它仍会留在那里，不停地引诱你，消耗你的精神资源。朋友，时间就这样走了，你会困惑它是怎么不见的。

顺便说一下，如果你是一个心绪不定的人，你还是可以看看我们网站的网页漫画！不必立刻回来把这篇文章读完。别忘了订

阅那个漫画源！看完所有的漫画后可以分享给你的朋友并聊聊！做这些事不会过于分心的，真的！

心如止水的程序员与心绪不定的程序员之间的区别

心如止水的程序员更倾向于长期的幸福感和生产率。他有能力高度集中注意力，关注目标。

心绪不定的程序员更倾向于短暂的满足感，长期这样不能到达全部的潜力并且会对他自己的身体和精神带来双方面的伤害。他不能定下心来集中注意力。

为了彻底领悟这点，我们首先需要理解我们的大脑是怎样工作的。

我们的大脑是怎么工作的

事实上我们不能完全知道，不过，基于我们现在已有的理解，心理学家已经提出了一些有用的类比方法，可以帮助我们领悟复杂的东西。

电脑的类比

你的大脑像一台电脑。当然，它要比一台个人电脑复杂多了，不过电脑的比喻说法可以让我们形容许多在大脑里运行的进程。有些东西像硬盘，有些东西像内存和中央处理器，还有进程和线程，它们的确需要一些时间来访问信息并计算数据。

很明显我们的大脑拥有一定的容量在任一时刻集中注意力。你

可以很好地集中在某一东西上，或是不太好地集中到几样东西上，但你不能两者都做到。把注意力从一个任务转到另一个任务甚至会影响认知能力，特别是两种毫不相干的任务。

只要你把意识关注在某样东西上，就会花些时间处理它和它所分配的资源并会从内存和后台进程里初始化它。它可以存在几个小时甚至几天。最近的关注到的东西在任何时刻都会不断出现的。即使你没有注意到它们在你的大脑里流动，它们一直都在那儿——而且它们占用资源，使你不能集中注意力。

抽屉的类比

另一种有用的比喻说法是柜子里的抽屉。你的大脑像存放东西的抽屉。每个抽屉都是拥有数据且互相连接的，基于某种上下文关系和概念。在特定的时间里，一些抽屉是打开的，一些是关闭的。打开的抽屉代表着你当前的精神空间，它们很容易访问到。

> 主要的问题是关闭抽屉是需要很长时间的，而打开它们是非常快的。

所以，当你在工作时，注意力转到新的地方，它会立即打开一堆抽屉。当你回到工作状态时，那些抽屉仍然是开着的。这样唯一的好处是可以提高你的创造力和头脑风暴——但是不需要的抽屉开着会有影响的。它们抑制左脑的思维并降低注意力。

编译时应该做什么

在等待电脑执行操作的时间里，你的行为可以判断出你是一个心如止水的程序员还是心绪不定的程序员。

主要的意见是：

1. 不要分心
2. 短暂的休息

保持注意力并不是那么难——你必须要切断你大脑的电源，或者保持注意力集中在你的代码上。不过，关闭大脑效果更好。这样会进入一个微冥想状态，在短暂休息之后它仍会有效。

为什么要进入冥想？好的，你的大脑每天都在咀嚼代码，为什么不给它一个休息呢？此外冥想科学上被证实能不断增加幸福等级。想象一下每天进行两次30-60秒表面上看起来不重要的冥想所带来的长期优势吧。

现在，短暂的休息时应该做些什么？从以下内容中选择任何事，按你的想法组合在一起。你可以把它们按照这个顺序全部做一遍，这取决于（也可以不在意）这些行为所需要的时间。

1. 站起来
2. 眺望远处
3. 把双手放在脑后，斜靠在椅子上
4. 伸长双腿，把双手尽量举高
5. 闭上双眼
6. 轻揉眼睛
7. 慢慢地把头向各个方向转，舒展颈椎
8. 闭上双眼然后深呼吸
9. 倒一杯水（要小心路上别被同事分心了）

你也可以做任何主要身体参与而不影响思维的事——静力锻炼，变戏法，决斗，哈哈

顺便说一下，如果你的老板不给你足够的时间做以上的任何事，那么推荐他来看这篇文章。让自己更快地恢复精神并保持注意力，比看起来在工作其实已经分散注意力并很快就感到累的情况要好很多。

伸展放松是很有益处的。我上一次见我的理疗师时她发现我的肩膀很多地方很紧绷。她告诉我这可能是因为我坐在电脑前双手总是弯曲造成的，所以肌肉缩短而且不能用力。那我应该怎么做最好呢？举高我的双手，试着举到最高。这是我在编译或是等待电脑执行操作时常做的一件事。

你可能已经听说过当你在用电脑时进行有规律伸展放松和短暂休息是一个不错的想法。问题是怎么做到——即使你设置了一个计时器，但假如到时你正处在一个复杂的事情，正做到一半的时候怎么办？这样看来就无法实现了。然而，如果你在编译时短暂休息一下，就能很好地做到了，你甚至可以把它养成一种习惯，让它变得更“自然”！现在编译就开始能提醒你做伸展放松了。这太令人不可思议了。

编译时你不应该做的事情

我其实想用力点击我的主页，这样做能让事情变得简单，这里有一张你在编译时不应该做的事情的列表。以这种方式思考：这些事情不仅会使你不能集中注意力，还会让你的思维和身体不能很好地短暂休息。

1. 阅读你的RSS源
2. 阅读新闻（任何新闻）
3. 收发邮件

4. 浏览各类社交媒体(Facebook,twitter,google+,linkedin, reddit)
5. 观看视频

做以下两件事要比上面的好一点，但依然不推荐:

1. 和同事聊天
2. 读一本物理书

专注的注意力和专注地修改

我马上就要结束这篇我所想的每日工作流程的理想方法——禅宗程序员的文章了。

保证生产率和效率的关键是注意力100%集中在你现在正在做的事情上，做完后再把注意力完全转移到其他地方。注意力从一件事情到下一件事情的过渡不能有任何的拖泥带水。

把你的工作时间分成1-2个小时一段。注意力100%集中在这些时间段里。然后是20-30分钟的休息，完全可以做任何事情。在休息时间你可以浏览你的邮件和社交媒体，当然，散个步打个盹吃些健康的点心会更好。休息结束后，检查任务管理软件(比如，我用的是我的Swift To-Do List)里下一个工作是什么，再开始另一段需要百分百注意力集中的工作。在较大程度上，这与番茄工作法有点相似。

休息不是可以选择要不要的。别想去跳过它。你的身体需要休息。即使你在做你喜欢的工作，你也需要休息一下——在这种情况，你会在接下来的时间段里有动力做更多事。

你的工作流程看起来应该像这样:

(任务1 - 任务2) - 休息 - (任务2 - 任务3 - 任务4) - 休息 - (另一个1-2小时的时间段) - 休息 ……

不要让它完成起来是像这样混乱随机，让人看不下去的的:

任务1 - 邮件 - 任务1 - Facebook - 任务1 - 任务2 - 短暂休息 - Facebook - 任务2 - 邮件 - reddit - 任务3 - 邮件 - 休息 - 任务2 - 邮件 - 任务3 - 推特 - 黑客新闻 - 推特 - 任务1 - 任务3 - 休息 - 任务4

如果你的工作流程是像上面这样，那你既不能完全放松也不能做完你能完成的事情。这是最低级的方法。这不仅浪费了你的潜力和时间，也让你长期身体会不适。

我不想撒谎。集中注意力真的不容易。它很困难，因为当你没有集中注意力时，你基本上会朝着相反的方向过去。习惯和根深蒂固的惯例像潜意识一样难改变。

好消息就是，你还可以练习。你可以学习怎样去集中精神。这是每个人都可以学的技能，而且它非常有用，值得为之努力。

——转载于Jiri Novotny+，作者由于Windows平台任务管理软件的困扰。他把这篇用户文章投递在 ComponentOwl.com，因为他的Swift To-Do List要使用 Component Owl 上基于.Net框架的Better ListView作为它的核心组件。

另附：你有没有身边的人会每天要检查50次邮箱？把这篇文章发给他。他之后会感谢你的。

英文原文：Are You a Zen Coder or Distraction-Junkie?

译者/周庆成
江西南昌人，毕业于上海海洋大学数学系。拥有多年互联网与移动应用开发经验，对iOS、Android等移动系统拥有极大兴趣，熟练使用Cocos2d-x与Unity3D等引擎，开发过iPad版三人斗地主等游戏，爱好广泛，在各种系统平台与编程语言上都有研究。此外还翻译了《Mac功夫》以及最新版的《Obiective-C基础教程》。目前居住于上海。从事游戏与网络应用开发。

谈谈纸书和电子书

——以读者、作者、出版者的三重身份感受它

周围很黑，没有光。
黎明前的黑暗让人很难受。

作为读者

作为读者，我是坚定的电子书拥护者。只要可能，我就会购买或下载电子书。除非某本我想看的书实在找不到电子版，我才会购买纸书。

作者 / 陈冰
图灵教育本版策划副主编。曾著有畅销书《Flash第一步》(销量3.5万册)和2011年出版的让人捧腹的电脑幽默畅销书《电脑使用说明书》(当当计算机入门类排行榜第一名)。他策划出版的《大话设计模式》(当当计算机类三本终身五星畅销书之一),销量已超8万册。陈冰策划的其他图书还包括《Java程序员,上班那点事儿》、《程序员羊皮卷》、《广告公司的秘密》、《Flex第一步》、《我是设计师》、《Android软件安全与逆向分析》等。图灵社区ID:陈冰。

对我来说,电子书好处多多:全天候、随时随地阅读,无论白天黑夜走路坐车都能阅读;单手把持和操作,腾出来的另一只手可以干别的事,或者只是简简单单地休息着;永远不必担心找不到它,都就在手机里存着呢;阅读速度更快,数据显示亮度更高的屏幕比不发光的纸面能明显提高阅读速度;想确认一下某句话是不是这本书里的,搜索一下就可以知道;搬家也不用发愁,几万本书带在身上,依然轻松步行。而纸书,以上种种好处都不具备且不说,时不时困扰我的一个问题就是,书多了后,经常记不得某本书搁那儿了。永远不会找不到的东西就是始终放在你手边的东西,托手机的福,电子书就是如此。

对于价格问题,就我来说,我很愿意付费,甚至比免费送给我更喜欢。因为你这里给我免费了,必然会在另外什么地方赚回去,我倒宁可你明明白白地赚我的钱。就我而言,我认为合理并且愿意接受的价格是纸书的1/3。

对于我喜欢的书,有时我也会下载非官方出品的免费电子书,但那只是因为这本书没有正版的电子书可买,如果有正版且价格合理,我必然买正版。

有些出版人(包括某些社长级人物)声称,电子书不适合深阅读,只能浅阅读。这就好比一个人便秘了,坐马桶上苦恼半个钟头,没有动静,站起来说这个马桶不适合深拉。这显然不是马桶的问题,这是便秘的问题,自己没把自己搞通透,反怨马桶不给力,这是不对的。生活越来越艰难,社会越来越浮躁,能自由呼吸的空气越来越少,这个情况下,还想让人们心平气和地把大把时间慷慨地用来读书,委实有些难了,人们更愿意把大把时间用来干更“实际”更“有用”的事情。

但知识的重要性人们其实很清楚,所以还是在尽量抽出边角时间来阅读。上下班地铁上、午饭后、马桶上,人们都在投入的

阅读，甚至投入到本末倒置的程度。举个栗子，很多时候，我为了增加阅读时间，不得不延长在马桶上的时间，因为一旦离开马桶，我立马就意识到还有比看书更重要的事情等待我去做，只有在马桶上时，我才能忘记这一点，心安理得的看书。而上述这些情况下，电子书比纸书做得好得多。

此外，就我个人而言，我在阅读电子书时的阅读完成度较高。也即，一本厚书如果是纸版，则我很可能看不完，半途而废。但如果是看电子版，则几乎总是能看完它。我揣摩了导致这一现象的原因，有了以下结论：

- 看纸版厚书时，它的厚度和重量时时刻刻在向你示威，在强调它是难以对付的。你可以拿《追忆似水年华》这种书感受一下。而电子书则隐匿了这种压迫感。
- 看纸版书时，你感到你就是在看一本书，它的质感、它的气味、它的颜色，不管从哪个角度看，它就是一本老老实实的书，不可能是任何别的东西。但看电子书时，由于是在手机里或平板电脑上看，因此它可能是一本书，但也可能是别的什么东西，比如一个游戏，因为你玩手机游戏的时候正是面对着这么一个东西。这就好比不管你吃什么鱼，如果你把鱼肉在醋和姜末中蘸一下，这块鱼肉吃起来就有了一些蟹肉的味道。这是形式对内容形成的反作用。电子书的阅读形式给书增加了一些趣味性和可能性，不显得那么呆板。你可以尝试一下，把一本你读纸版读不下去的书，找一个排版舒服的电子版，在手机上或平板电脑上阅读，你会发现这本书确实变得有趣了一点。

因此，从做为读者的角度，我希望所有书都推出电子版，甚至只出电子版即可。只有一种情况下，我会希望买到纸书，那就是这本纸书不仅内容绝佳，而且其整体装帧和气质让我想要把它作为一件艺术品收藏的时候。我不仅要阅读它，还要抚摸它，这时我才会买纸书。

作为作者

作为作者，我的心理很矛盾。一方面我很清楚我做为读者时的心理，所以很希望能提供我作品的电子版给像我一样的越来越广大的电子书读者来享受阅读，但另一方面，我又有两大担心，使我只能在万不得已的情况下才提供电子版：

一是目前国内图书盗版、非法上传扫描版现象极为严重。销得好的书自不用说，即使是那些只比一般书销得稍微好一点点的书，在网上也能找到高清扫描版。随着高速扫描仪、单反相机这些设备普遍白菜价，以及专为制作图书扫描版而开发的软件的推出，扫描一本书的成本低到可以忽略不计，时间就是个把钟头。作者与出版社齐心协力折腾一两年才搞出来的书，上市没几个月就出现高清扫描版这种事现在已成为常态。

以我本人策划（这里暂时切换到我作为出版者的身份）的一本计算机书为例，拢共预计也就1万册左右的销量，结果上市后1个半月就出现了高清扫描版，在网上搜了一下，已经有几十个网站在提供下载，其中人气最旺的两家在不长时间内下载量就都已超过了1000次。能达到几十万甚至上百万册的超级畅销书毕竟是少数，大多数书的销量也就在5千到3万之间，尤其对于计算机书这种工具性较强且达到1万册就算畅销书的种类，这种高清扫描版对正版图书的危害还是相当可观的。发现被侵权后我们联系了其中几家下载量大的网站，要求它们删除文件，但收效甚微，只有两家删除了，其他几家依然故我，这结果也在我意料之中，因为已经遇到多次了。这种事你打官司也没用，投入很多时间最后赔不了几个钱，维权成本高收益低，一旦碰上，作者和出版社也只能眼睁睁地看着它继续发生。

可想而知，在不提供电子版的情况下，非法的高清扫描版（还

带有制作好的目录标签，相当人性化）都这么快就冒出来了，如果再主动提供电子版，一旦被破解，那岂不等于是给敌人提供弹药。毕竟不同于正式的电子版，高清扫描版是图片化的，不具备搜索等需文字化后才能提供的功能。

担心自己的作品推出电子版后会更容易地被盗版，是作者普遍担心的一个问题，虽然出版社在与作者签订图书出版合同时通常是一次性签下包括电子版在内的所有版权，但就作者自身而言，其对推出电子版并无多大意愿。

二是担心电子版的推出影响纸书的销售。对于作者来说，稿费的意义十分重大，稿费一旦出问题，就好比洗澡时，在脑袋上打满了洗发液，好，开始挠，开开心心地把泡沫都搞出来了，准备冲水时，居然停水了！这种打击是致命的。如果不巧再赶上当天还有重大的社交活动没有完成，那更是会留下难以磨灭的心灵创伤。所以任何一个作者在稿费问题上的态度都是坚决的、毫不含糊的。虽然作者通常很难亲自监控一本书到底印了多少册，但目前大多数出版社在印数上还是诚信的，多数也在书上印有累积印数或当次印数（对于不印印数的出版社，我个人是非常鄙视的，这是一种可耻的行为），作者在这点上还是较为信任出版社，通过印数和版税率作者可以知道自己大约会获得多少稿费。

但一旦推出电子版，由于电子版的售价通常只有纸版的1/3到1/5，所以作者会很自然的认为电子版将夺去一大批原本会买纸书的读者，而同时由于作者无法直观地得到电子版的销售数，所以难免会担心这里面可做的文章太大，保不齐这部分收入会被出版社独吞。纸书销售受损，又无法从电子版销售中找补回来。也罢，老夫宁可只出纸书。

站在作者的角度上，电子版卖得少，对作者的收入毫无意义，只徒增了被彻底破解充分盗版的风险；电子版卖得多，直接影响纸书销售，作者的直观收入降低，而出版社是否能诚信的回馈电子版的销售收入，也很难讲。因此，无论电子版卖得好还是不好，对作者似乎都是弊大于利。

此外，绵延近两千年的对纸书的习惯，使得人们已将出版与纸书划上了等号，作者也是这么想的。抛开金钱问题，在作者看来，如果出版的不是纸书那就等于没出版。只出版电子版？你都不好意思跟人打招呼。

> “嗨，我出了本新书。”
> “哇，厉害啊，叫什么名，上市了吗，当当上能买了吗？”
> “是电子版，你去XX直接看就行了，我给你网址。”
> 然后，你就得到以下回答。
> 完全不懂行的：“电子版？什么意思？”
> 懂行的：“人穷不能志短！能有点志气么你？！”
> 心肠软的：“行了，别说了兄弟，是不是日子有些紧？”

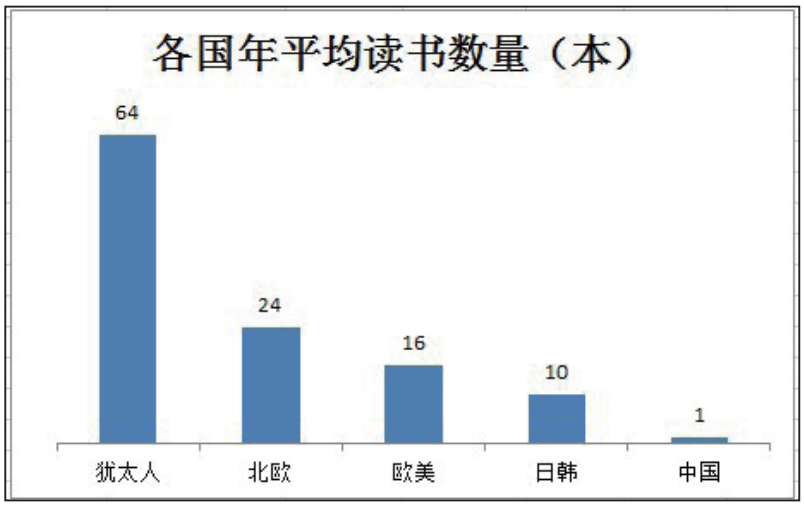
就眼下而言，作者只在一种情况下，才对出版电子版感兴趣，那就是在纸书无法出版，或一时之间难以出版的情况下，才愿意尝试电子版，主要是出于一种聊胜于无的心理。

作为出版者

我整理了和分析了一些数据，我们先一起感受一下。

过去两年中，也就是2012年和2011年，18至70岁的国人的图书阅读率为54.4%，人均年读书4.37本。请注意，十多年前，1999年的时阅读率就曾超过60%。那么，其他国家是个

什么情况呢，全世界每年图书阅读量排名第一的是犹太人，平均每人一年读书64本，欧美国家人均年阅读量约为16本，北欧国家达到24本，日韩人均年阅读10本左右，而中国13亿人口，(扣除教科书)平均每人一年读书连1本都不到。



然而，与上述数据形成鲜明对比、极具讽刺意味的是，我们这样一个人人不读书的国家每年的出书品种却屡创新高，2011年出书品种达到令人叹服的37万种，2012年更是强弓劲弩，在景阳岗上英勇地射杀了40万种这只大虫，把位于落基山脉的第二名的美国小伙伴当场吓傻(美国2012年出版的传统图书只有35万种)。

那么，一年40万种新书这到底是个什么概念呢？从在龟壳上凿字，直到辛亥革命，伟大的中华文明一共出版了20万种图书。也就是说，现在一年的出书量相当于过去3000年出书量总和的两倍。

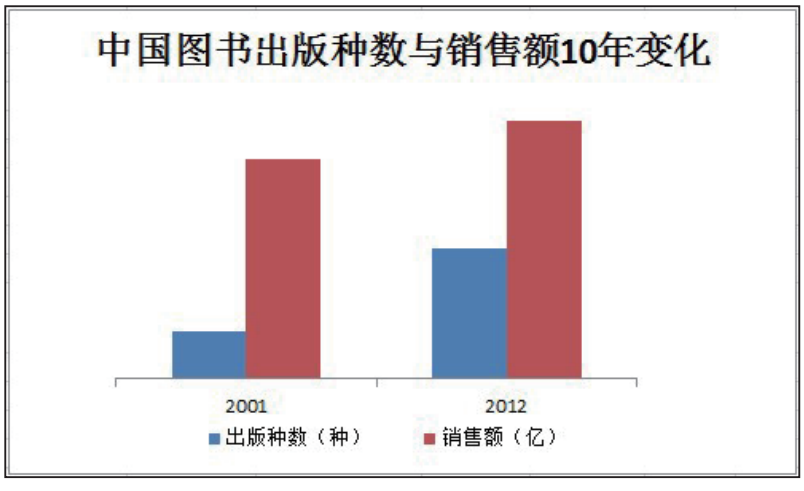
感谢蔡伦的纸和毕升的小泥活字，在1450年时，我国累计出版的图书已达4万种，比同期整个欧洲的出书总量还多1/3，

稳占“全球出书品种数”第一的位置。不过德国人谷腾堡在1450年发明了现代印刷术(他也凭此重大发明在“影响人类历史进程的100名人排行榜”上位列第8，紧跟在蔡伦和释迦牟尼的后面)，几十年后，欧洲的出书总量就远远超过了我国。

时光荏苒、岁月如梭，谁成想,500年后，我们又重新夺回“全球出书品种数”第一的位置。

但质量如何呢？让我们来看一看。太远的就不说了，看看近十年的情况，2001年，全国共出版图书约15万种，销售额约700亿元;2012年，全国共出版图书约40万种(确切说是41.4万种)，销售额约824亿元，其中全国新华书店系统和出版社自办发行销售额约为694亿元，网店销售约130亿元。有人可能会感觉这里指定有问题，不是说当当一家就占了图书销量的1/3吗，这数据明显对不起来啊，事实上，人家当当说的是占了图书“零售”市场的1/3的。

十一年的时间，品种增长率是166.7%，而码洋增长率仅有17.7%，这期间因通货膨胀纸价上涨等原因导致2012年的书均售价(40元)约为2001年的两倍。



在2006年之前，动销品种一直稳定维持在当年新书品种的5到6倍，按此计算，2001年书均销售册数约为7780册；但自2006年起，由于年人均购书数逐年下降，加之网店的低价冲击，导致实体书店大批倒闭了事(2006至2012年间倒闭超过1万家)，使得可用于展示和存放图书的空间减少，当然更重要的原因是我们的新书出版能力确实太强，双重夹击下致使动销品种与新书品种的比率逐渐下降，2012年的动销品种数为125万种(只有新书的3倍了)，书均销售册数约为1650册。也就是说，2012年的图书书均销售册数只有2001年的1/5。如果考虑到这十年间图书销售生命期的缩短，这个比率实际上会更小。

来看看同期美国图书市场的情况，2001年美国出版图书约21.5万种(其中，传统图书19.5万种，自出版2万种。因自出版总共销售额仅20万美元故忽略)，销售额250亿美元，书均售价为9.75美元，书均销售册数约为2390册;2012年美国共出版图书75万种，其中传统出版35万种，自出版40万种，销售额271亿美元，其中电子书销售额为61亿美元，占22.55%，传统书平均售价为15.75美元，电子书平均售价为5美元，传统书书均销售册数约为760册(约为2001年书均销售册数的1/3)，电子书书均销售册数约为610册。美国的这一数据可能会出乎不少人的意外，敢情我们跌跌不休十一年后书均销量居然还是美国2倍多，哈哈哈哈，我骄傲！

其实没啥可骄傲的，因为我们前面计算出的书均销售册数实际上是动销品种的年书均销售册数，没有考虑图书销售生命期的问题，当仅对国内图书进行比较时，可以不做考虑。但要同美国图书进行比较，则必须纳入计算，因为美国图书的书均寿命是4年，而中国则是2年(十年前其实也曾有过4年的时候)，简化一下的话，中国图书的典型销售曲线是第二年的销售是第一年的1/3。美国图书的典型销售曲线是基本上匀速下降的。

美国图书的书均寿命之所以长出一倍，是两个原因造成的：一是美国的出版社对于一本书都是先推出精装版，一年后再推出平装版，平装推出时的宣传会对精装版形成第二次购买刺激，两个版本互相照应，延长了彼此的销售寿命，二是美国对版权高度重视，你不可能买到盗版，也几乎没有作者剪刀浆糊来做抄袭书和跟风书，而原创的内容总是有些价值的。考虑图书的销售生命期后，重新计算，目前国内图书在整个销售生命期内的书均销售数量约为1650+1650*33%=2195册，而美国图书的该数值约为760+570+380+190=1900册。

说到版权和抄袭问题，目前确实很严重，不过大家也不必太过担心，由于人人都不读书（而少数读书的老百姓的眼睛自然是雪亮雪亮的），所以这些生产出来的书都如愿以偿地没有到达读者（万幸），大部分垃圾书生产出来后连上架的机会都没有（确实没地儿搁），不曾与亲爱的读者们见面，在书店库房转了一圈就直接拉回出版社库房了，但库房还不是这些书永久的家，在暗无天日的库房中抱着一丝侥幸枯坐两年，把确实没有人想要买它们这件事情坐实之后，才重新被拉回造纸厂打纸浆，至此，整个出版流程才算画上一个句号。

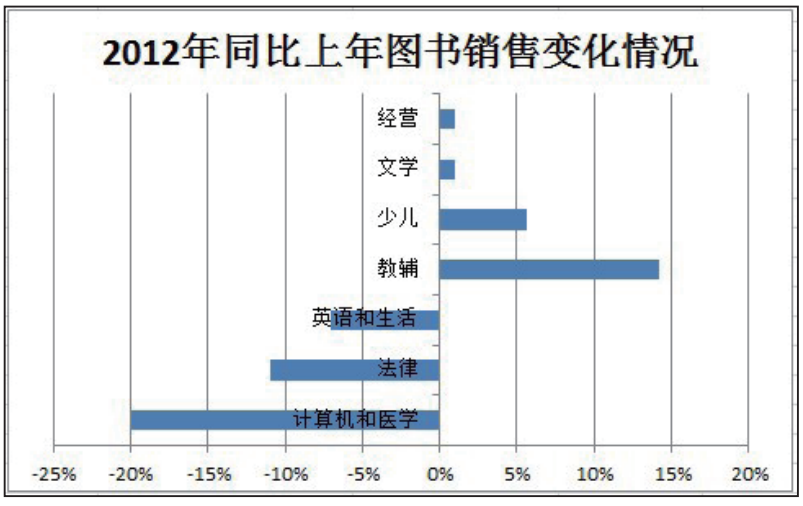
说到库房，就捎带说下库存这个突出问题。虽然最近一两年首印数都已经降到4000册左右了，但因为之前几年首印5、6、7千册的积压书有些还在占据库房，导致目前库存问题还在攀升，不过相信不久就会降下来。从目前计算出的书均销售册数看，对于拿不定主意的书，首印4000册显然也是高了，3000-3500册才是未来几年首印数的归宿啊。美国图书目前的平均首印量为3000册。

虽然书均销量我们比美国稍多，但由于美国过半图书都已自出版（含电子书），加之纸书的平均首印数低，使得美国的库存问

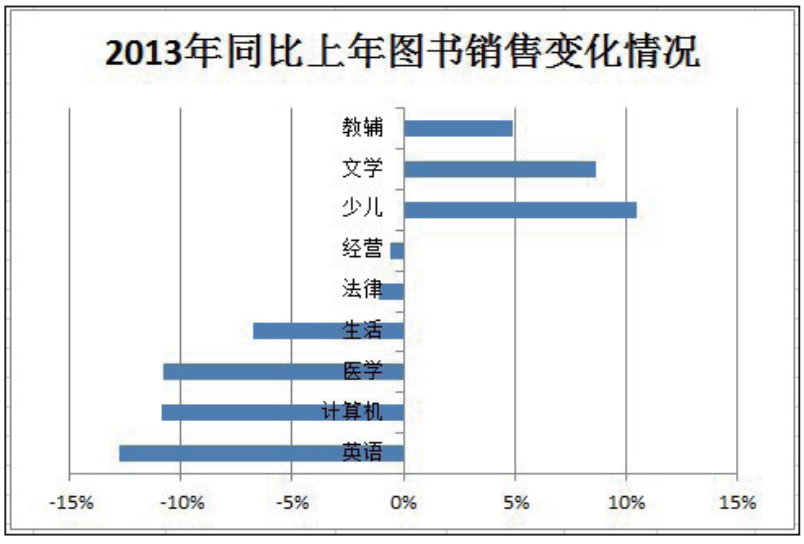
题比我国好得多，整体情况健康得多。再多说两句，美国是开放出版的（目前不开放出版的国家，放眼全球，只有我国和朝鲜了），出版业已是充分竞争，全美目前约有图书出版机构8.7万家，但每年出版1种以上图书的只有1.2万家（有没有觉得另外7.6万家很萌），其中规模在150人以上的只有100多家。美国的图书出版市场已经完全细分，7.6万家萌萌的出版公司的存在本身就标示着美国出版业的繁荣与强悍。

再看看更近的数据（采纳开卷的数据）：

2012年6月较之2011年6月，全国9个受监控主要细分市场中，5个出现下降。其中，计算机和医学类同比下降20%以上，降幅最大；法律类同比下降10.98%，英语和生活类同比降幅居于2%至5%之间。其余4个细分类表现为同比增长，其中教辅类同比增长14.18%，增幅最大；其次是少儿类，同比增长5.66%；文学和经管类的增幅较小，均为1%左右。但需注意的是，由于我国自2005年起进入第四个生育高峰（2005年至2020年），因此，少儿类和教辅类的增长只是一种硬性需求的反映，而非真实的大众购书意愿。抛开这两类后，大众购书需求实为大幅下降。



接下来，今年6月较2012年6月，9个主要细分市场中除教辅(4.90%)、文学(8.69%)和少儿(10.50%)同比上升外，其余6个细分市场均为同比下降，其中英语(-12.75%)、计算机(-10.83%)、医学(-10.76%)、生活(-6.78%)、法律(-1.08%)、经管(-0.59%)。抛开少儿和教辅类后，除文学增长明显外，其他6个市场均为下降，其中三个呈现大幅下降。



为什么近十年图书销量大幅下降了，为什么大众不读书了？原因很简单，老百姓的生活忒难了，生活的压力太大了，坑爹和腐败的事情忒多了，每天都在发生各种震惊世界和挑战下限的事情，老百姓疲于和生活做殊死搏斗，没有心思和时间来看书了。

现在政府觉得老百姓都不读书挺丢脸的，出于对老百姓身心健康的关怀，推动精神文明建设，决定搞全民阅读，几个月前推出了《关于开展2013年全民阅读活动的通知》，出于树立典型的考虑，还要进一步评选出十大读书人物。大多数老百姓不

知道此事，注意力都被其他更加凄凄惨惨的事情吸引了。督促老百姓看书是好事，但看书，不是随便喊喊口号就行的，这需要三个条件：一，能买得起书，二，有看书的时间，三，看书有用。

对于一，2012年美国人的平均年薪约为5万美元，书均售价为15.75美元，书价占年薪比约为0.03%；2012年日本人的平均年薪约为377万日元，书均售价为1117日元，书价占年薪比约为0.03%。2012年我国居民的平均年薪约为37760元，书均售价为40元，书价占年薪比约为0.10%，为美国和日本的3.3倍。

对于二，鲁迅先生说过“时间就像海绵里的水，只要愿挤，总还是有的。”所以有没有时间不是根本问题。只要读书真的有用，那怎么着也会抽出时间来读。

对于三，说到底，这个是最关键的，看书有没有用这个问题实际上是知识有没有用的问题，那么在眼下的中国，知识有用么？几乎在所有的领域，知识似乎都在贬值。唯有两个领域却闪闪发光，这两类书也成了个中翘楚，去看看公务员考试的书有多火，去看看饭局上的学问这类书有多火吧。对于这两类书，我除了联想到腐败和行贿，实在想不出其他。

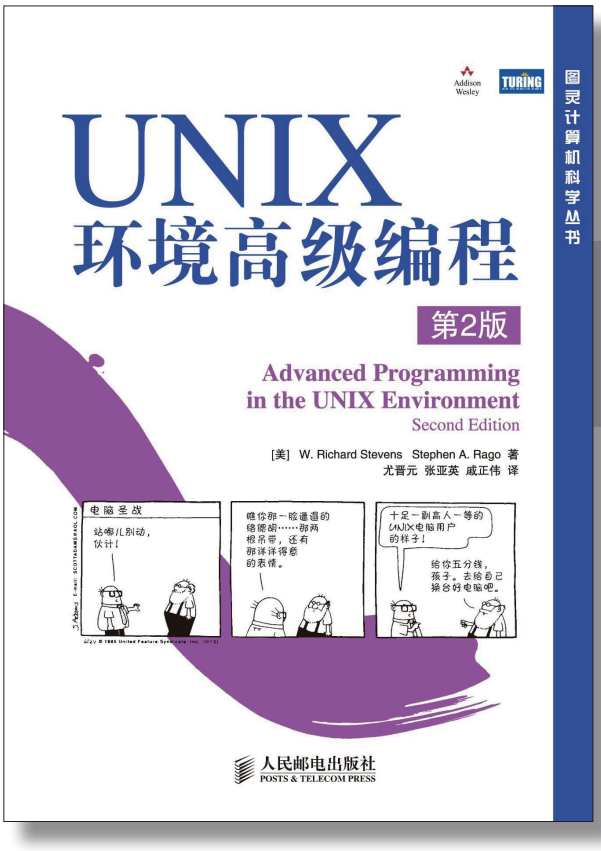
那这个世界上，什么国家的人民看书多呢，毫无疑问是那些买得起书，而且因为看书确实管用所以也愿意拿出时间来读书的民族。这样的民族，都是率真自由，自视甚高的民族。在这里，我想说一下以色列这个国家，以色列建国时间只比我们早一年，很小的国土很少的人，但在自由民主的制度下，迄今为止已出现10位诺贝尔奖获得者，人均GDP 32060美元，人均年读书64本。看书，说到底，依然有用。

那么，在目前这个情况下，摆在我国出版者面前的是什么情况呢，美国这出版业的先行者已经用事实的成功给我们指出了一条道路。

随着2007年底亚马逊推出Kindle，以及2008年之后的几年，大屏手机和平板电脑的潮水淹没全球，潜水十年的电子书终于浮上海面，开始登陆、攻城掠地，以自出版的形式、以按需出版的形式，无法阻挡。

感谢社区会员yinchuan制作了美观大方的图表

书　榜



UNIX环境高级编程（第2版）

作者：W.Richard Stevens，
Stephen A.Rago

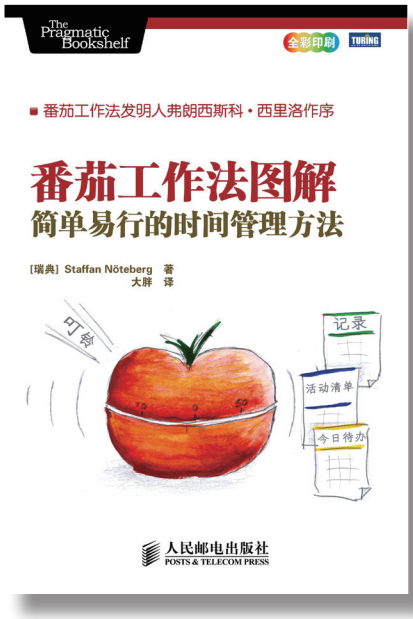
译者：尤晋元　张亚英　戚正伟

书号：978-7-115-14731-0

图灵社区推荐：

本书是被誉为UNIX编程“圣经”的Advanced Programming in the UNIX Environment一书的更新版。在本书第1版出版后的十几年中，UNIX行业已经有了巨大的变化，特别是影响UNIX编程接口的有关标准变化很大。本书在保持了前一版风格的基础上，根据最新的标准对内容进行了修订和增补，反映了最新的技术发展。

本书由Dennis Ritchie作序推荐。



番茄工作法图解：简单易行的时间管理方法

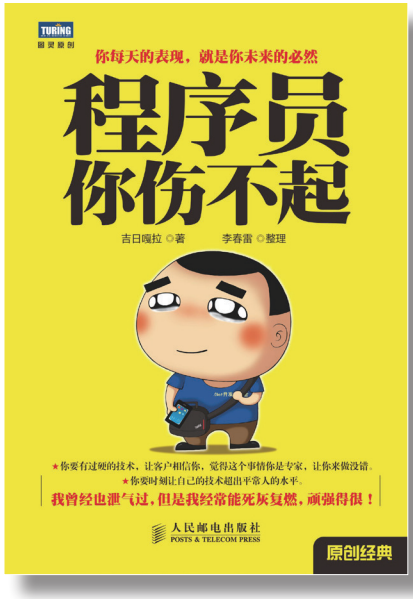
作者：Staffan Noteberg

译者：大胖

书号：978-7-115-24669-1

图灵社区推荐：12

谁不想活得轻松？谁不想妙计百出？谁不想与时俱进？谁不想享受假期？但要怎么实现呢？频繁的中断、重复的活动、迫近的期限，常常使我们力不从心。本书介绍了时下最流行的时间管理方法之一——番茄工作法。番茄工作法简约而不简单，本书亦然。



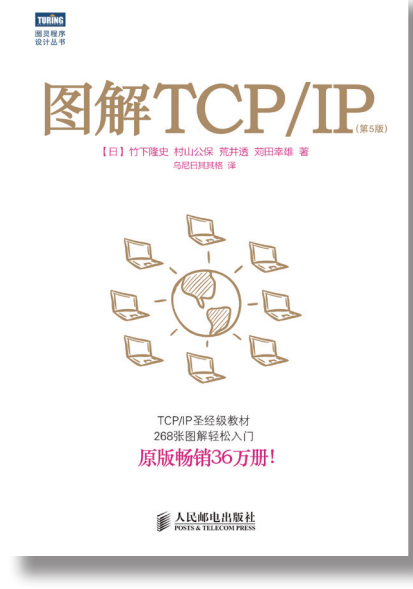
程序员，你伤不起

作者：吉日嘎拉

书号：978-7-115-31834-3

图灵社区推荐：6

四年时间记录十年程序历程。本书是作者博客文章的精选集。是作者作为老牌程序员、现在的IT创业者15年软件开发生涯的心路历程和经验总结。涉及程序人生、开发经验、职业规划、创业心得。字里行间充满了不屈不挠的码农正能量。



图解TCP/IP(第5版)

作者：竹下隆史，村山公保，荒井透，苅田幸雄

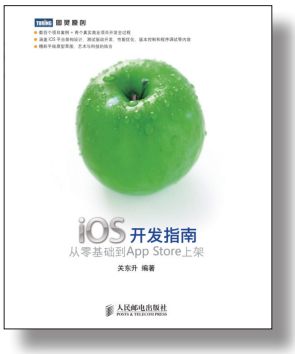
译者：乌尼日其其格

书号：978-7-115-31897-8

图灵社区推荐：9

这是一本图文并茂的网络管理技术书籍，旨在让广大读者理解TCP/IP的基本知识、掌握TCP/IP的基本技能。书中讲解了网络基础知识、TCP/IP基础知识、数据链路、IP协议、IP协议相关技术、TCP与UDP、路由协议、应用协议、网络安全等内容，引导读者了解和掌握TCP/IP。

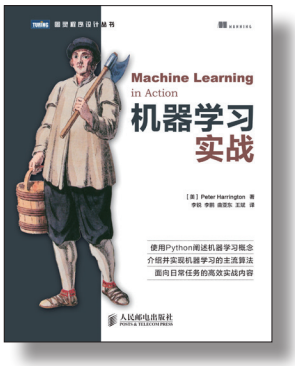
5



iOS开发指南 从零基础到App Store上架

作者：关东升
书号：978-7-115-32444-3
图灵社区推荐：8

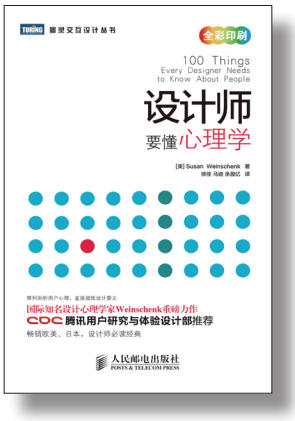
6



机器学习实战

作者：Peter Harrington
译者：李锐　李鹏　曲亚东　王斌
书号：978-7-115-31795-7
图灵社区推荐：29

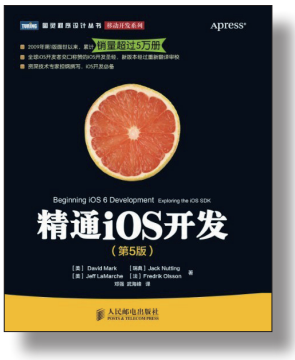
7



设计师要懂心理学

作者：Susan Weinschenk
译者：徐佳　马迪　余盈亿
书号：978-7-115-31308-9
图灵社区推荐：11

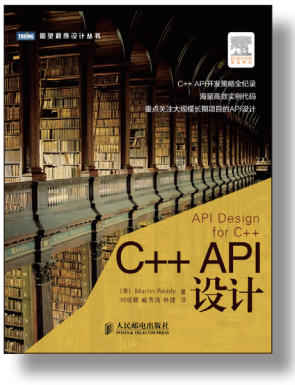
8



精通iOS开发（第5版）

作者：Fredrik Olsson , Jeff LaMarche等
译者：邓强　武海峰
书号：978-7-115-32761-1
图灵社区推荐：3

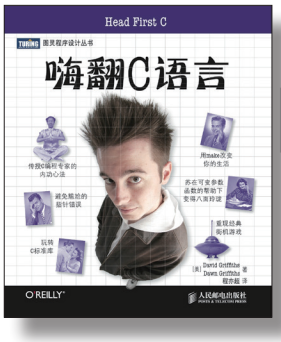
9



C++ API设计

作者：Martin Reddy
译者：刘晓娜　臧秀涛　林健
书号：978-7-115-32299-9
图灵社区推荐：19

10



嗨翻C语言

作者：David Griffiths , Dawn Griffiths
译者：程亦超
书号：978-7-115-31884-8
图灵社区推荐：18

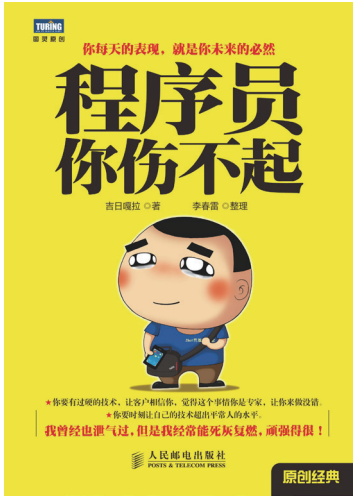
《程序员，你伤不起》编辑的话

作者 / 陈冰
图灵教育本版策划副主编。曾著有畅销书《Flash第一步》（销量3.5万册）和2011年出版的让人捧腹的电脑幽默畅销书《电脑使用说明书》（当当计算机入门类排行榜第一名）。他策划出版的《大话设计模式》（当当计算机类三本终身五星畅销书之一），销量已超8万册。陈冰策划的其他图书还包括《Java程序员，上班那点事儿》、《程序员羊皮卷》、《广告公司的秘密》、《Flex第一步》、《我是设计师》、《Android软件安全与逆向分析》等。图灵社区ID ：陈冰。

去年春天的一天，我在博客园看到一篇文章《你妹啊，程序员你伤不起的一些谬论》，这好玩的标题吸引了我。“程序员，你伤不起”，这作为一本书的名字会很不错。我边琢磨，边开始阅读该文作者吉日嘎拉的其他博文。这些博文约有五六百篇，是作者在几年里陆续写成的，内容是对自己编程生涯的体会和心得，有技术方面的，也有职场方面的。文章风格则是风趣幽默与浑然不自知的冒傻气的混合体。

我写过博客，懂得写博客的困难在哪儿，困难就在——一直坚持写。写出一篇精彩的博客不难，写出几篇精彩的博客公平地讲就有点难度，而要坚持写出上百篇有价值的博客那是需要极大毅力的。

我佩服那些能坚持写日记的人，一篇日记单独看多半没什么意义，但如果你把写日记这件事情坚持下来，即使记录的都是生活琐事，多年下来后，当你重新翻看这些日记时，你会发现它们的价值。你会看到你走过了哪些路，遇到了哪些人，做过了哪些事，那些你现在已完全遗忘的日子你也的确是亲自走过的，你会看到你思想和意识的变化。你得到的不再是一个武断的结果，而是整个过程，因为你将它记录了下来。这会给你，以及所有看过这份日记的人以启示。



四年时间记录十年程序历程。《程序员，你伤不起》是作者博客文章的精选集。是作者作为老牌程序员、现在的IT创业者15年软件开发生涯的心路历程和经验总结。涉及程序人生、开发经验、职业规划、创业心得。字里行间充满了不屈不挠的码农正能量。

吉日的这些博客是在2009年到2012年四年中写下来的，但记录的并不只是这四年中发生的事情，因为写作本身会勾起回忆，这里也写下了作者在1996年至2009年间发生的一些IT往事。

事实上，这些博文记录了一个IT人15年的成长和经历。你可以从这些文章中读到一个人在长达15年中的学习历程、职场闯荡、创业经历。你会看到放弃与坚持，看到变化与成长。

我相信，这些触动了我的内容，也会触动读者。这些内容对程序员和IT从业者是有帮助的。所以，我决定把这些博文精选后成书。吉日的中文不好，博文中好多话说得怪怪的，另一些话则完全前言不搭后语，所以我请好友李春雷来进行文字的整理以及博文的筛选。春雷是IBM的项目经理，也是文笔幽默的作家。他的网名是主席，而且会抽烟，不著一字，已尽得风流。闻听此事，他胸有成竹一口答应下来，还回复了一个“我很棒”的微笑。在他历经数月终于整理完吉日的稿子后，他的第一句获奖感言是“我～～～靠，简直比我重新写一本书都累！”我相信这是他的肺腑之言。感谢春雷。

这本书里写到了技术，分享了技术，但如果你想找的是一本专门讲解开发技术的书，这本书完全满足不了你的期待。

这本书里写到了职场，分享了创业，但如果你只想看IT业界叱咤风云的领袖级人物写的圣经级心得，那这本书也会令你失望。

这本书的长处不在以上，这本书的长处在于，它是一位草根级牛X程序员写的，写得很实在，一点不虚伪。字里行间充满了

直视自身的坦诚。阅读此书，很多讲述你会感同身受。作者在智商上并非天才，因为文中有一句话“我看书很慢，一个字一个字地看”，天才不会这么干；在情商也普普通通，因为文中还有一句话“就像我这么满脑子都是经商意识的人”，一般能这么说话的人，经商意识都不会太强，情商与你我应在伯仲之间。就是这样一个普通的草根程序员，经过十年的坚持，逐渐拥有了一个令他自己感到自豪的产品，并拿到了一笔风险投资。

这在有些人看来可能算不得什么，但我相信，对于很多程序员来说，开发出一个令自己感到自豪的产品，并能获得一笔风投资金去做自己想做的事情去完善自己的产品去实现自己的理想，这就是幸福的人生。

通过这本书，你会看到，即使你并不天赋异禀，成不了乔布斯和马云，但你依然可以做到牛X，有所成就，成为幸福的人。

你可以经常失败，但唯有在放弃的时候，你才成为失败者。吉日没有成为失败者，因为“我曾经也泄气过，但是我经常能死灰复燃，顽强得很。”

你，也可以做到。

本书策划编辑　陈冰

2013年5月3日

欢迎加入
图灵社区

最前沿的IT类电子书发售平台

电子出版的时代已经来临。在许多出版界同行还在犹豫彷徨的时候，图灵社区已经采取实际行动拥抱这个出版业巨变。作为国内第一家发售电子图书的IT类出版商，图灵社区目前为读者提供两种DRM-free的阅读体验：在线阅读和PDF。

相比纸质书，电子书具有许多明显的优势。它不仅发布快，更新容易，而且尽可能采用了彩色图片（即使有的书纸质版是黑白印刷的）。读者还可以方便地进行搜索、剪贴、复制和打印。

图灵社区进一步把传统出版流程与电子书出版业务紧密结合，目前已实现作译者网上交稿、编辑网上审稿、按章发布的电子出版模式。这种新的出版模式，我们称之为“敏捷出版”，它可以让读者以较快的速度了解到国外最新技术图书的内容，弥补以往翻译版技术书“出版即过时”的缺憾。同时，敏捷出版使得作、译、编、读的交流更为方便，可以提前消灭书稿中的错误，最大程度地保证图书出版的质量。

现在购买电子书，读者将获赠书款20%的社区银子，可用于兑换纸质样书。

最方便的开放出版平台

图灵社区向读者开放在线写作功能，协助你实现自出版和开源出版的梦想。利用“合集”功能，你就能联合二三好友共同创作一部技术参考书，以免费或收费的形式提供给读者。（收费形式须经过图灵社区立项评审。）这极大地降低了出版的门槛。只要你有写作的意愿，图灵社区就能帮助你实现这个梦想。成熟的书稿，有机会入选出版计划，同时出版纸质书。

图灵社区引进出版的外文图书，都将在立项后马上在社区公布。如果你有意翻译哪本图书，欢迎你来社区申请。只要你通过试译的考验，即可签约成为图灵的译者。当然，要想成功地完成一本书的翻译工作，是需要有坚强的毅力的。

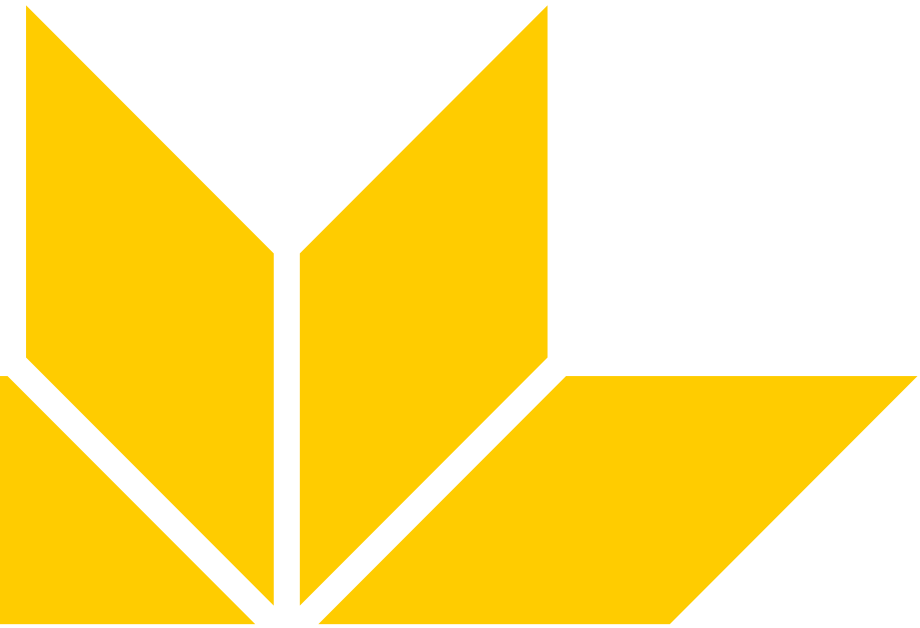
最直接的读者交流平台

在图灵社区，你可以十分方便地写作文章、提交勘误、发表评论，以各种方式与作译者、编辑人员和其他读者进行交流互动。提交勘误还能够获赠社区银子。

你可以积极参与社区经常开展的访谈、乐译、评选等多种活动，赢取积分和银子，积累个人声望。

图灵社区 出品

出版人：武卫东
编辑：李盼
顾问：杨帆、谢工、李松峰
设计：大胖



地址：北京市朝阳区北苑路13号院领地OFFICE C座603室
电话：010-51095181
微博：weibo.com/ituring
Email: ebook@turingbook.com