

你是如何成为 Lisp 程序员的(转)

我成为 Lisp 程序员的道路曲折而漫长。我曾于 2007 年 10 月 3 日在自己的日记中总结了自己的学习经历，现抄录于此。

最早在 2000 年 5 月，斯托曼院士访华时告诉我，Lisp（或者它的现代变种 Scheme）是功能最强大的编程语言，他本人就是一位高级的 Lisp 程序员，他还精通 C，GNU Emacs 就是采用 C 和 Lisp 两者开发的。我当时已经掌握了 C，但不会用 Lisp，但是我完全相信他说的都是真的。于是，一心想成为编程高手的我，决定学习和掌握这门编程语言。我从 2000 年下半年开始学习 Lisp。下面总结的学习经历大致按照时间先后顺序排出：

有必要先说明一点，在我遇到斯托曼院士之前，我在上大学时曾经阅读过一本关于人工智能的著作，《Artificial Intelligence --- Making machines "think"》，作者是 Neill Graham，这本书的最后一章（第14章）是关于 Lisp 语言的简介，所以我对人工智能和 Lisp 的概念并不完全陌生。这本书前面的章节都很好理解，但是在这最后一章，我遇到了很大困难。我花了许多时间试图明白其中的道理，不过最后成效不彰，现在回想起来，究其原因，最主要的就是我没有一个适当的上机练习环境，在读了许多书中的东西后，当时我感觉似乎明白了其道理，但是实际上并没有真正理解清楚。不过，有两点在我看来无疑是确定的：一、Lisp 早已经成为人工智能研究项目的首选（或者说是默认的）编程工具，在人工智能领域没有其他语言能撼动其领导地位。二、对于具有表结构的数据操作，对于列表（list）头元素（pair 的 car 的部分）的处理采用递归方式比较好，而对于众多的体元素（pair 的 cdr 的部分）则采用迭代的方式处理效率更好。

斯托曼院士回国后，我首先在计算机上尝试用 Emacs Lisp 编程，它是嵌入在 GNU Emacs 文本编辑器中的解释器。在庞大的 Lisp 家谱中，Emacs Lisp 不是 Common Lisp，而是早期的 MacLisp 的一个直系后代，同时在一些方面作了简化和强化。同时我开始阅读 Robert Chassell 所著《Introduction to Emacs Lisp Programming》，Robert Chassell 是斯托曼院士早年结识的战友，也是自由软件基金会的合创人之一，他很早就使用 GNU Emacs，而且使用 Emacs Lisp 程序定制 GNU Emacs，斯托曼友善地把 Robert Chassell 介绍给我认识。这本书既是自由文档（可以从 GNU 的网站免费下载），又是自由软件基金会出版社（GNU Press）的出版物。等我读完了这本书之后，我觉得这本书实在太美妙了，作者的文笔十分了不起（即使对于想学习英语写作的人，帮助也应该很大），把这本书介绍给其他人是完全值得的。我于是找了两位翻译人员（毛文涛博士和吕芳女士），把它译成了中文，我则担任了全书的编辑和审校工作。中文版质量很高，我很满意，它作为一本很伟大的编程入门书籍十分适合广大读者自学（我认为读者应该搞到一本阅读）。我至今还想自己动手翻译这本书的第三版，可惜如今我很难再找到当年那么多的时间做编辑和审校之类的工作了。

阅读完这本书之后，我意识到如果想使用 Emacs Lisp 开发非玩具级别的实际应用程序，那么根据作者的推荐，自由软件基金会出版的《GNU Emacs Lisp Reference Manual》是必不可少的工具书，我打印了这份文档的第 2.4 版本，厚厚的共四本。后来这份文档正式出版，从 GNU 网站上订购的图书升级到了 2.6 版本，针对的是 GNU Emacs version 21。我不太认同 Eric Raymond 在他的名著《The Art of Unix Programming》中对 Emacs Lisp 的评论，他以为 Emacs Lisp 只能为 Emacs 编辑器本身编写控制程序，而赶不上其他脚本语言全面。实际上，我认为只要熟悉了 Emacs Lisp 的细节，其他任何脚本语言能完成的工作，都可以使用 Emacs Lisp 程序完成。我亲眼看见斯托曼院士在 GNU Emacs 内完成电子邮件的编辑、收发等工作，不用 Eric Raymond 开发的 fetchmail 程序一样干得很好。我自己也利用 Emacs Lisp 编写过 CGI 应用程序，效果也不错。

Bob Glickstein 曾经写过一本《Writing GNU Emacs Extensions》，可以配合 Robert Chassell 的书与《GNU Emacs Lisp Reference Manual》，作为补充读物。

读了 Robert Chassell 的书之后，我开始花时间阅读 David Touretzky 博士所著的《Common Lisp: A Gentle Introduction to Symbolic Computation》，这本书可以从互联网上免费下载，读者可以自行在万维网上 google 得到它。这也是一本伟大的 Lisp 著作，内容已经是基于 Common Lisp 的，但是作者并没有特意强调这一点。我把下载的 PDF 文件打印出来，自己动手把打印出的文档纸张装订成了两卷手册。我从这本书中得到的最大收获是我充分认识到 Lisp 中的一切都是对象：数字原子（numeric atoms）和符号原子（symbolic atoms）都是对象。数字原子求值返回它自身的值，而符号原子则有名称（name）、类型（type）、值（value）、秉性表（plist）和绑定表（bindlist）。这五个字段可以放入一个数据结构中，并在实现中以 C 语言的 struct 表达。

在阅读这些材料的同时，我又从网上找到了 Gary Knott 教授编写的一份文档，《Interpreting Lisp》，这份文档篇幅不长，从来没有正式出版成书。在这份文档中，作者利用 C 语言编写了一个微小的 Lisp 实现，非常接近于最初的 Lisp 实现。最可贵的是他将实现的源代码和盘托出。从这本书中，我清晰地看到了如何构造 Lisp 对象的结构，我开始认识到内存垃圾收集算法的重要性。在理解了 David Touretzky 博士所著的《Common Lisp: A Gentle

公告

昵称： 果丁
园龄： 7年11个月
粉丝： 0
关注： 1
[+加关注](#)

< 2020年7月				
日	一	二	三	
28	29	30	1	
5	6	7	8	
12	13	14	15	
19	20	21	22	
26	27	28	29	
2	3	4	5	

搜索

常用链接

[我的随笔](#)
[我的评论](#)
[我的参与](#)
[最新评论](#)
[我的标签](#)

我的标签

[c++\(1\)](#)
[others\(1\)](#)

随笔分类

[C++\(11\)](#)
[others\(6\)](#)
[随笔\(2\)](#)

随笔档案

[2015年11月\(1\)](#)
[2015年10月\(2\)](#)
[2015年9月\(2\)](#)
[2015年8月\(2\)](#)
[2015年7月\(1\)](#)
[2015年6月\(1\)](#)
[2015年5月\(1\)](#)
[2015年4月\(2\)](#)
[2015年3月\(2\)](#)
[2015年2月\(1\)](#)
[2015年1月\(9\)](#)
[2013年9月\(3\)](#)
[2013年8月\(5\)](#)

文章分类

[c++\(1\)](#)

Introduction to Symbolic Computation》介绍的 Lisp 对象的结构基础上，我明白了书中图示的 Lisp 对象中若仅在结构设计时安排五个字段是不够的，还需要有供垃圾回收（GC，Garbage CCollector）模块操作的字段才行。

在 2001-2002 期间，我开始接触 Scheme。在此之前的 2000 年 8 月，Rorbert Chassell 曾来中国访问，我们在西安时，他向我介绍了 Scheme 是我应该关注的语言。Scheme 于 1985 年诞生于 MIT，发明人有两位：一位是 Gerald Sussman 教授，他是自由软件基金会的董事会成员之一，另一位是 Guy J.Steele 博士，下面即将更多地提到。我首先使用了 Dorai Sitaram 所著的教材《Teaching Yourself Scheme in Fixnum Days》，这份文档可以从网络自由下载，不过坦率地说，这本书教材不太适合初学者，阅读它的人至少应该具有很多基础知识或者经验才行。我并没有从这本书中获得太多的帮助，对 Scheme 的基本概念也没有搞清楚，特别是连续（Continuation）之类的概念没有理解。不过，收获还是有很多，我意识到 Scheme 是一个非常优美的 Lisp 变种，它继承了源自 Algol60 和早期 Lisp 两者的特点。这份教材的最后还列举出了一个具体的编程实例，讲授如何利用 Scheme 编写 CGI 程序。现在在我看来，在 CGI 编程等领域，Perl 和 PHP 等脚本语言获得广泛应用简直就是钻了 Lisp 社团不善营销的空子。

到了 2001 夏天，我从美国获得了《Structure and Interpretation of Computer Programs》的教师手册，是我的朋友 James Gray 帮我买的，他是我一个很好的朋友，不过我这里要善意地埋怨一下，他做事有些马大哈，我原来希望他给我搞到这本书的学生用书，没有想到他却寄来了教师手册，我想他在买书时没有仔细区分一下，而此著作的教师手册和学生用书的封面都采用了同一图案，作者和 MIT 出版社的编辑的确很优秀，他们在营销本书时非常成功，后来专门还开设了一个网站推广本书和相关的教学材料，网上公布了教材的全部内容，加上两位作者课堂教学的视频录像。教师手册价格比较便宜，James 在购买时要么没有仔细辨认清楚，要么就是帮我省钱，才寄来了这本教师手册。我当时没有学生手册，也就没有继续学习 Scheme。不过话说回来，James 寄给我的教师手册我一直都留在我手头，后来对于我在黑客道教学中讲授 Scheme 帮助极大，在这里还是应该深深地感谢 James Gray。

不久（2003年），我买了一本 Patrick Winston 教授所著的《Common Lisp》第三版，作者是麻省理工学院人工智能实验室的主任（斯托曼早年就是他手下的兵），在美国的人工智能研究领域名气很大，我就是冲着他的名气才买此书的。阅读完之后，我觉得这位教授名副其实，而不像我在国内见到的一些人徒有虚名，拿不出真东西。这是一本介绍 Common Lisp 的极好教材。后来 Hans Hagen（ConTeXt 排版软件包的主要作者之一）告诉我这本书的合著者 Berthold Klaus Paul Horn 在 TeX 社团名气也很大，的确如此，从本书的排版质量即可看出许多名堂来，排版样式一看就是典型的 TeX 风格。作为中文 TeX 用户俱乐部（CTUG）的主席，我已经知道，国内学术界（我指的是数学界和计算机科学技术界）很少有人精通 TeX 排版系统，鲜有人能使用它排版自己的讲义或著作。

读了这本书之后，我感觉自己必须阅读 Common Lisp 的语言参考手册，许多问题必须在看到语言规范（这是基本的尺度）之后才能搞清楚。Guy L. Steele 博士写过这样的手册，而且他写了两次，第一个版本是在 Common Lisp 标准化之前完成的，第二个版本是在标准化完成之后写成的，但是网上有人评论说有些该写的东西没有写进去，此书是否会有第三版不得而知。Guy L. Steele 博士是非常著名的语言手册的作者，他已经为 C、Common Lisp、Java 等编程语言都写过的语言参考手册，都非常成功，这些参考手册都是一版再版，销路极好，比位于瑞士的国家标准化组织（ISO）发布的非常昂贵的标准文档销路好许多。他写的这些语言参考手册已经成为编写这些语言编译器作者们的大救星。

大约在同时，我下载了《On Lisp》，这是 Paul Graham 博士编写的一本优秀著作，从中我得到了许多 Lisp 概念的细节，特别是 Lisp 的 macro 机制，以及黑客们如何利用 Lisp 思考问题。作者介绍的自底向上（bottom up）的方法论对我触动很大，而作者的讲解是非常富于启发性的（作者曾专程赴意大利的美术学院学习过油画创作，所以具有很高的艺术修养）。从那时开始，混合编程（Hybrid Programming）的思想在我头脑中开始成型，我坚信 Lisp 将会成为一种非常长寿的编程语言，这使我联想起斯托曼院士当年在四川九寨沟就 GNU Emacs 开发对我讲过的话。在 GNU Emacs 和 Lisp 背后隐含的方法论是永远不会过时的。

2004 年，我真正找到了 Lisp 编程的感觉，觉得自己开始进入状态，并开始使用 Scheme 开发真正的应用程序，我编写的程序是一个网络应用程序，即一个网络留言板（Web-based bulletin System），在万维网上可以运行，CGI 的模块是采用 Scheme 写的，Apache 在服务器上通过 Scheme 的 CGI 程序接上了 PostgreSQL 数据库。我使用的是 PLT Scheme 的 103 版本，我非常喜欢这个版本，既简单又很干净，我用 C 语言和 PostgreSQL 提供的 libpg 编写了一个 DA（database adaptor），让 Scheme 程序可以访问 PostgreSQL 数据库。

完成了这个项目之后，好事成双，我得到了渴望已久的《Structure and Interpretation of Computer Programs》（简称 SICP，或者“紫皮书”），作者就是 Harold Abelson 教授和 Gerald Sussman 教授。正是这一年，我开始利用自己头脑中形成的数学观点，特别是在我的泛系尺度论中表达的思想，来认真学习 Scheme，并且主动地从中国古代的阴阳太极图模型来理解当今电子计算机系统上的计算模型。这一过程延续了很长时间，直到 2005 年的冬天才最终获得成功！这期间的许多思想写入了我的著作《自由软件：新的游戏规则》第三卷内篇的第二章“论尺度”。今后我还准备花更多的时间把它扩展开来，形成一部单行本的著作《泛系尺度论》，在这个单行本中，我将利用更长的篇幅把中国古代的哲学思想、现代数学思想和计算机编程融为一体，对整个计算理论提出自己完整的一家之言。在黑客道九个段位中，初段就是讲“计算的本质”，里面就纳入了我的思想方法和编程经验。

2004-2005 年期间，我仔细地研究了 R5RS 文档中除了第七章之外的所有内容，收获巨大。对于第七章的内容，当时仍然有些疑惑，因为这些材料需要理解大量的关于 lambda calculi 的细节和大量的预备知识，我当时还没有找到充分的材料钻研。另外，在研究 PLT Scheme 的源代码时，内存垃圾回收算法对我而言，仍然是一大疑难问题，显然，

java(1)

文章档案

2013年8月(3)

阅读排行榜

1. 40款免费开源游戏(4606)
2. emacs配置(985)
3. QuickLisp常用命令(778)
4. putchar问题(582)
5. OMG 接口定义语言(IDL)(5

推荐排行榜

1. 40款免费开源游戏(1)

对于内存垃圾回收技术，我还需要学习更多的背景材料。

到了 2005 年的年底，我把 R5RS 翻译成了中文。在完成翻译的过程中，我知道了如何利用形式语言和扩展的巴科斯 - 劳尔范式 (EBNF) 来定义一门编程语言的形式句法和 语法规则，以及如何正确理解和读懂它。

2005-2006期间，我学习了其他许多关于 Lisp 编程的书籍，包括 Paul Graham 博士的《ANSI Common Lisp》、Matthew Flatt 等人合著的《How to Design Programs》, Brian Harvey 和 Matthew Wright 合著的《Simply Scheme --- Introducing Computer Science》(此书的封面设计别出心裁，非常值得回味)、Daniel Friedmann 和 Matthias Felleisen 合著的《The Little Schemer》和《The Seasoned Schemer》。另外我花了相当多的时间仔细阅读《The Scheme Programming Language, 3e》，这是 R. Dybvig 教授的代表作，他是 Chez Scheme 实现的设计大师。年底我得到了《Hackers and Painters》(“黑客和画家”)，这是 Paul Graham 博士所著的散文集，与 Robert Chassell 一样，他也是一位伟大的作家，他的行文非常容易阅读，而且这本书中的内容如同其书名副标题一样，的确收入了许多伟大的想法，这些想法 对于创新公司利用 Lisp 开发创新项目是非常富有启发性的。

2006 年 7 月 15 日，我的学生干俊哲从南韩的汉城大学带来了他学习的两本著作的复印件：George Springer 和 Daniel P. Friedman 合著的《Scheme and the Art of Programming》，以及 Mark Watson 的《Programming in Scheme: Learn Scheme through Artificial Intelligence Programs》。前一本的难度在紫皮书之下，比较好读，其中许多程序如同棋谱一样，展现了许多高级编程技巧，值得反复思考，我立即在 PLT Scheme 实现上验证了书中的大部分代码；后一本则介绍如何使用 MIT Scheme 来设计人工智能程序，非常精彩。

2006 年 7 月送走了干俊哲之后的夏天，我一直在瑞士苏黎世度假（八月下旬我还去了西班牙马德里参加了国际数学家大会），苏黎世中央图书馆（ZB, Zentralbibliothek Zuerich）是一所了不起的图书馆，现有藏书一百二十万种。据说列宁当年在欧洲流亡时曾来到苏黎世，就睡在这个图书馆里读书学习。八月份时，大多数瑞士人也在休假，图书馆的人不多，非常安静。我在这段时间从图书馆中找到了非常多的背景材料，包括 H.P. Barendregt 所著的数学经典教材《The Lambda Calculus --- Its Syntax and Semantics》，这本书于 1981 年由 North Holland 出版社出版，对这一数学分支作了详尽的介绍，我认为对于这一主题，今后再也无人可以写得比这本著作更好了。另外，我找到了第一本关于 lambda calculus 的著作，是由这个理论的创始人 Alonzo Church 教授创作的，《The Calculi of Lambda-conversion》简直是无价之宝，在这本小册子中，作为数学家，作者清晰而精炼地阐明了 lambda calculi 的全部内容。任何一位想掌握 lambda calculus 的人都应该仔细阅读本书。在图书馆中还找到了《An Introduction to Lambda Calculi for Computer Scientists》，作者是 Chris Hankin。Matthias Felleisen 和 Matthew Flatt 合写的《Programming Language and Lambda Calculi》也打印出来了，并仔细阅读了两遍，这两人是 PLT Scheme 研发小组的核心成员。在苏黎世中央图书馆的书架上，我还看到了 SCIP 紫皮书的第一版的德文本，书中的内容与英文版第二版大同小异，但是我敏锐地发现，第一版的第四章中没有收入 eval 和 apply 两个高阶算子构成的太极推手图，第二版中则出现了。

我花时间研究了《Lisp 1.5 Programmer's Manual》，这是世界上第一份真正意义上正式发布过的 Lisp 稳定实现版本的手册，作者就是 John McCarthy 等人，极具学术权威性，我认为任何一位 Lisp 程序员都应该阅读这本手册。时隔多年后，我又开始阅读关于人工智能方面的著作，《Lisp, Lore and Logic》是 W. Richard Stark 写的，《Artificial Intelligence, theory and practice》是 Thomas Dean 等人写的，他们都已经使用 Common Lisp 来说明问题。阅读时，我参考了前面已经提到的 Patrick Winston 教授编写的经典教材《Artificial Intelligence, 3e》。

2007 年初，我开始关注 Scheme 社团中尚处于起草状态中的 R6RS，这个文件将成为新的 Scheme 语言规范。我现在仍然认为 Common Lisp 太复杂、太庞大，回厂大修似乎也不太可能，因为工业界已经很好地接受了 Common Lisp，而 Scheme 将是未来的主流。我开始按照这一规范来开发自己的 Scheme 实现版本，这一实现版本称为 MNM Scheme。

2007年6月至7月间，我在瑞士苏黎世的 Campus Zollikerberg 打印了 R5.97RS，我花了许多时间理解这一新的规范，特别是它与前一个版本（R5RS）的差异。同时，我重新思考了 PLT Scheme 实现的源代码和涉及模块（modules）、名称空间（namespaces）、盒子（box）类型、define-values 和其他附加在 R5RS 规范之上的特性与实现风格。

从苏黎世中央图书馆借到的另外一本具有重大价值的著作就是 Richard Jones 和 Rafael Lins 合著的《Garbage Collection》，这本书极大地帮助我理解了内存垃圾回收算法设计的细节。我从此开始真正明白了 Scheme 实现工作中的最后一个阴暗角落。对于一切 Lisp 对象，内存垃圾收集的算法设计时其实不存在理论上的最优算法，算法的效率受到多种因素的影响，而 Lisp 的设计者可以根据自己的设计思想来决定应该怎样回收内存垃圾。

这时的我已经成长为熟练的 C++ 程序员，站在 C++ 程序员的立场看，一切 Lisp 对象都有类型，我可以用 C++ 语言内置的类（class）来刻画它们（即声明各种用户自定义的类），一切 Lisp 对象从存储空间分配和回收的角度来看具有共性，所以，这可以利用 C++ 的模板来表达 Lisp 对象的存储管理结构，而各个 Lisp 对象占有的存储空间大小，则可以利用类的构造函数（constructor）和析构函数（destructor）对内存分配和内存垃圾回收在模板的支持下进行操作。

源自 Lisp 发展起来的 GC 是一项“古老”的技术，实际上它已经广泛地被采用了，Java 这门当今新的商业编程语言中

就有，而且 Java 的 GC 算法设计得非常好，我决定在我的 Scheme 实现中参考它。2007 年 9 月，我在从香港飞往苏黎世的飞机上，我阅读了美国著名的程序员 Bruce Eckel 所著的《Thinking in Java》的第四版原著，从作者的介绍中，我结合从《Garbage Collection》中获得的知識，我理解了 Java 的内存垃圾回收算法的总体思路，并构思了如何利用他们的算法来改进我的设计。而在我离开苏黎世回国的同一天（2007年9月26日），R6RS 技术委员会的全体编辑成员决定冻结对草案的讨论，正式发布了这一规范，从那一天起，我实现 MNM Scheme 的步伐也大大加快了。（不过，请读者留意在委员会举办的投票时，也有许多人投了反对票，并给出了反对意见，这些意见与支持的意見一起，都是极有研究价值的。）

Scheme 的实现版本非常多，而且自己能否动手实现一个是考验一个计算机专业人士学术修养深浅的好指标（这也是我在黑客道中把第五段的教学内容定为“解释器的原理与构造”的原因之一）。迄今优秀的 Scheme 实现有：PLT Scheme、MIT Scheme、Chez Scheme 等等。我认为 PLT Scheme 是非常优秀的实现版本，它是按照 GPL 发布的自由软件，值得在这里推荐给广大读者。毫不客气地讲，我的 MNM Scheme 也应该算一个。


Common Lisp 的实现版本很多，抛开 Franz Lisp 等商业版本不谈，自由软件社团中最有名的两个实现分别是：Bruno Haible 等人从 1992 年以来一直维护和开发的 CLISP，以及卡耐基梅隆大学的小组开发 CMU Common Lisp。这两个 Common Lisp 实现都很好，我个人比较喜欢使用 CLISP。目前最好的 Common Lisp 编程著作可推荐 Peter Seibel 写的《Practical Common Lisp》，这位作者的天分显然比我高，他原来是 Java 和 Perl 程序员，2004 年才开始学习 Common Lisp，他花了一年的时间学习它，就完全学会了，而且在学习的同时，边练习、边写书，结果很丰硕，写出的这本书就是读者能看到的結果，这本书得到了广泛的认可，它出版后获得了美国出版界计算机图书创作的震撼大奖。他在一次采访中提出了一个新的说法：时代的发展需要“第二代 Lisp 程序员”，而且每个程序员都应该学习 Lisp。（让我再次回忆起斯托曼院士在 2000 年时对我说过的话。）

正如读者在上面所读到的，成为一位了解 Lisp 一切内幕的程序员可真不容易，但是我很高兴，因为我已经成功地逾越过了这个初看起来曾经非常高的门槛，我现在已经是这样一位程序员了。学习 Lisp 给我带来了巨大的乐趣，如果没有这种在编程中产生的乐趣相伴，我绝对不会花这么长的时间来学习它。今天，我由衷地自豪，因为如果按照 Peter Seibel 的说法和衡量标准，我已经是第二代优秀的 Lisp 程序员群体中的一分子。

上面罗列的学习经历对于一般人而言显然太长了，黑客道学员们则可以站在我的肩膀上，借鉴我的经验和教训，少走许多弯路、避免走死胡同。值得庆幸的是，我已经在黑客道的课程设计中自觉地做了许多工作，凡是参加了黑客道的初段课程学习的学员（S1：“计算的本质”），即可在较短的时间内学会掌握 Lisp。

分类: [others](#)




 果工
关注 - 1
粉丝 - 0
[+加关注](#)

0 0
[推荐](#) [反对](#)

« 上一篇: [MSVC vs. MinGW 之dll玩转攻略手记\[转\]](#)
» 下一篇: [40款免费开源游戏](#)

posted @ 2015-06-17 12:41 果丁 阅读(332) 评论(0) 编辑 收藏

[刷新评论](#) [刷新页面](#) [返回顶部](#)

 注册用户登录后才能发表评论，请 [登录](#) 或 [注册](#)， [访问](#) 网站首页。

- 【推荐】了不起的开发者，势不可挡的华为，园子里的品牌专区
- 【推荐】有道智云周年庆，API服务大放送，注册即送100元体验金！
- 【推荐】超50万行VC++源码：大型组态工控、电力仿真CAD与GIS源码库
- 【推荐】《Flutter in action》开放下载！闲鱼Flutter企业级实践精选



相关博文：

- [Lisp的本质\[转\]](#)
- [大学，你是怎么教我们的，我们是怎么成为程序员的？？](#)
- [我是如何从煤矿工成为程序员的](#)
- [Matt DeBoard：我是如何成为程序员的](#)
- [继电器是如何成为CPU的（1）](#)
- » [更多推荐...](#)

最新 IT 新闻:

- 特斯拉财报电话会议实录：完全自动驾驶是未来几年最重要的研发工作
 - 蚂蚁金服三届操盘手：从破壁者到海贼王到霸主
 - 微软财报电话会议实录：企业业务将在未来微软的体系中发挥重要作用
 - 你永远也叫不醒一个被PUA催眠的人
 - 抵制退潮 中国手机在印度“大促销”
- » 更多新闻...