TURING

Apress®

Douglas Crockford

Simon Peyton Jones Brendan Eich

Joshua Bloch

Guy Steele Brad Fitzpatrick

Ken Thompson

Peter Norvig Donald Knuth

编程人生

15位软件先驱访谈录

CODERS AT WORK

[美] Peter Seibel 著

图灵

人民邮电出版社

POSTS & TELECOM PRESS

TURING

Douglas Crockford Joe Armstrong
Simon Peyton Jones Brendan Eich
Dan Ingalls Joshua Bloch
Guy Steele Brad Fitzpatrick
Bernie Cosell Fran Allen
Ken Thompson
Jamie Zawinski Peter Deutsch
Peter Norvig Donald Knuth

编程人生

15位软件先驱访谈录

CODERS AT WORK

[美] Peter Seibel 著
图灵俱乐部 译

人民邮电出版社
北京

图书在版编目（CIP）数据

编程人生 : 15位软件先驱访谈录 / (美) 塞贝尔(Seibel,P.) 著 ; 图灵俱乐部译. -- 北京 : 人民邮电出版社, 2011.1(2011.12 重印)
书名原文: Coders At Work
ISBN 978-7-115-23907-5

Ⅰ. ①编… Ⅱ. ①塞… ②图… Ⅲ. ①计算机科学－科学家－访谈录－世界 Ⅳ. ①K816.16

中国版本图书馆CIP数据核字(2010)第189039号

内 容 提 要

这是一本访谈笔录，记录了当今最具个人魅力的15位软件先驱的编程生涯。包括Donald Knuth、Jamie Zawinski、Joshua Bloch、Ken Thompson等在内的业界传奇人物，为我们讲述了他们是怎么学习编程的，在编程过程中发现了什么以及他们对未来的看法，并对诸如应该如何设计软件等长久以来一直困扰很多程序员的问题谈了自己的观点。

本书适合所有程序员，也适合所有对计算机行业、对软件开发感兴趣的人。

编程人生
15位软件先驱访谈录

◆ 著 [美] Peter Seibel
译 图灵俱乐部
责任编辑 朱 巍

◆ 人民邮电出版社出版发行 北京市崇文区夕照寺街14号
邮编 100061 电子邮件 315@ptpress.com.cn
网址 http://www.ptpress.com.cn
北京铭成印刷有限公司印刷

◆ 开本：700×1000 1/16
印张：30.5
字数：546千字 2011年1月第1版
印数：5 001 –6 000 册 2011年12月北京第2次印刷

著作权合同登记号 图字：01-2009-6909号
ISBN 978-7-115-23907-5

定价：79.00元

读者服务热线：(010)51095186 印装质量热线：(010)67129223
反盗版热线：(010)67171154

版 权 声 明

译者简介

丁雪丰　网名DigitalSonic，InfoQ中文站编辑，满江红开放技术研究组织核心成员，Spring Framework 2.0 & 2.5文档翻译项目负责人。平时积极投身开源项目，是著名SFTP/SCP软件WinSCP的简体中文汉化者。参与翻译及编著的书籍有《Spring攻略》、《JRuby实战》等。

叶淮光　嵌入式软件工程师一枚，hgye@twitter。

米全喜　现从事金融软件的项目管理工作，关注质量管理和软件过程改进。译作有《SQL解惑》、《团队之美》等。

张　龙　同济大学软件工程硕士，InfoQ中文站Java社区编辑，满江红开放技术研究组成员。热衷于编程，对新技术有强烈的探索欲，对Java轻量级框架有一定研究。目前对Mac、iPhone/iPad、Android开发、动态语言及算法产生了浓厚兴趣。翻译出版了《Dojo构建Ajax应用程序》、《Spring高级程序设计》等书籍。有5年的Java EE培训讲师经历。联系方式：zhanglong217@yahoo.com.cn。博客地址：http://blog.csdn.net/ricohzhanglong。新浪微博是http:/t.sina.com/fengzhongye，欢迎follow。

吴　珂　中国传媒大学在读研究生。刚刚结束在英国GTI公司的程序员实习生活，回到北京继续修炼。喜读书、电影、音乐、运动，果粉和Google控，热爱互联网和各类电子产品；恶不整洁的厨房和书桌，轻度强迫症。梦想成为一个伟大的程序员。要找到我，请在twitter上找@diamrem。

李琳骁　译有《Linux 命令详解手册》等多本计算机相关图书，目前从事Linux内核驱动及嵌入式系统开发。爱好Python、Vim、技术翻译、给女儿讲故事。个人网站：http://linxiao.net。

李　清　曾攻读英语专业，后辅修计算机，现在做测试开发，对自然语言和计算机语言都比较感兴趣，平时业余喜欢写点小程序玩儿。

郝培强　网名tinyfool，身高180cm，体重240斤，标准的中年老胖子。有妻有女，无房无车，现居上海，程序员，二手经济学家，现任职于盛大创新院。Blog：http://tiny4.org/blog/。twitter：http://twitter.com/tinyfool。

董金乾　软件工程师。关注分布和并发网络应用，对软件设计和开源社区有着浓厚兴趣。

戴　玮　80后SOHO宅男。硕士期间的研究方向是人工智能与模式识别，从此一直对机器学习、数据挖掘、计算机视觉怀有浓厚兴趣，曾经承担为中国医学科学院等单位开发智能系统的课题项目。编程方面喜爱C/C++、C#、Python，对Java有不明原因的厌恶感。闲暇时喜欢翻译（译言网ID是loveisp，欢迎加好友）、琢磨哲学（苏菲的世界莫非就是我们的世界？）、打篮球（不过一年多没碰过了）、玩Flash游戏（Kongregate ID是loveispdvd，欢迎加好友）、用emule下电子书（硬盘就像那什么，挤挤总会有的）。

（本书正文中脚注除说明的以外，均为译者注。）

前　言

李清　译

抛开Ada Lovelace（19世纪的一位伯爵夫人，她为Charles Babbage未完成的分析机设计了算法）的工作不说，人类在计算机编程领域奋斗的时间还不及一个人的寿命长，从1941年Konrad Zuse完成Z3电子机械计算机（首个可运行的通用计算机）算起，只有短短68年。曾有6位女性（Kay Antonelli、Jean Bartik、Betty Holberton、Marlyn Meltzer、Frances Spence和Ruth Teitelbaum）在美军的计算机部队工作，她们手工计算弹道数据表，后来被调去做ENIAC（首台通用电子计算机）最早的程序员。若从这时候算起，人类的编程历史则只有64年。在婴儿潮早期出生的人们以及他们的父母，很多至今仍然健在，他们出生时世界上还不存在计算机程序员。

当然，这些已经是历史了。现在世界上有很多程序员。劳动统计局在2008年对美国125万人进行了统计，大约每106个工作者当中就有1个是计算机程序员或软件工程师。这还没算美国之外的职业程序员、数不清的学生和业余编程爱好者，还有很多人从事其他正式工作，但却花费了一部分或很多时间来试图驯服计算机。虽然有数以百万计的人写过代码，虽然在编程出现后人们写过的代码没有数万亿行也有数十亿行，我们仍然不断地在这一领域进行创造。人们仍然在争论编程到底是数学还是工程，是工艺、艺术还是科学。我们仍然在（经常是带有强烈情绪地）争论编程的最佳方式，因特网上有无数的博客文章和论坛帖子来讨论这些问题。书店也摆满了各种论述新编程语言、新编程方法、新编程思想的书。

本书按照文学期刊《巴黎评论》（*The Paris Review*）的传统，采取了一种不同的方法来讲述什么是编程。这家期刊曾派了两位教授去采访小说家E. M. Forster，这次采访和随后的一系列问答式的采访后来辑录为*Writers at Work*一书。

我采访了15位成就斐然、经验丰富的程序员，其中有些人是系统黑客，如Ken Thompson（Unix的发明者）和Bernie Cosell（ARPANET早期实现者

之一)；有些人既有强大学术实力本身，又是著名黑客，如Donald Knuth、Guy Steele和Simon Peyton Jones；有些人是业界的研究员，如IBM的Fran Allen，爱立信的Joe Armstrong，Google的Peter Norvig，以及曾在施乐帕克研究中心工作过的Dan Ingalls和L Peter Deutsch；有些人是Netscape的早期实现者，如Jamie Zawinski和Brendan Eich；有些人参与设计和实现了现在的万维网，如刚才提到的Eich，以及Douglas Crockford和Joshua Bloch；还有Live Journal的发明人Brad Fitzpatrick（在伴随Web成长起来的一代程序员当中，他是一个当之无愧的典型)。

在采访中，我问他们有关编程的问题，问他们是怎么学习编程的，在编程过程中发现了什么，以及他们对未来的看法。而且我很用心地请他们多谈谈长久以来程序员一直在苦苦思索的那些问题：我们应该如何设计软件？编程语言在帮助我们提高生产力和避免错误方面扮演了什么角色？有什么办法可以更容易地查出难以发现的bug?

这些问题远远还没有解决，所以我的采访对象持有非常不同的观点也就不那么奇怪了。Jamie Zawinski和Dan Ingalls强调尽早让代码跑起来的重要性，而Joshua Bloch则描述了在实现之前，他如何设计API并测试它们能否支持要写的代码。Donald Knuth讲述了他在编写排版软件TeX的时候，怎样在敲代码之前先用铅笔在纸上完整地实现整个系统。Fran Allen大力批判近几十年来人们躺在C语言的脚下对计算机科学的兴趣越来越低，Bernie Cosell称之为“现代计算机最严重的安全问题”，Ken Thompson却认为安全问题是程序员而不是编程语言造成的，Donald Knuth也说C的指针是他所看到过的“最令人赞叹的记法改进之一”。一些受访者对“形式化证明可能有助于改善软件质量”这一观点嗤之以鼻，而Guy Steele则漂亮地展示了这种做法的优点和限制。

然而，仍然有一些主题是大家都认同的。几乎所有人都强调保持代码可读性是很重要的。大部分受访者都认为最难查找的bug出现在并发代码中。没有人认为编程问题已经完全解决了，他们大多数人仍然在寻找编写软件的更好办法，比如怎样自动分析代码，如何让程序员更好地协作，或者寻找（或设计）更好的编程语言。同时几乎所有人都认为多核CPU的大量采用将会给软件开发带来重大改变。

这些谈话发生在计算机发展史的一个特定时刻。因此，本书中讨论的一些话题在当前是紧迫问题，今后将不再是问题而变成了历史。但即使是像编

程这种新兴领域，历史也能为我们提供很多教训。除此之外，我觉得我的受访者们可能有一些共识，包括什么是编程，如何更好地编程，等等，不仅现在的程序员会从中受益，将来几代程序员也将从中受益。

最后提一下本书的书名：*Coders at Work*。这个书名与前面提到的《巴黎评论》出的*Writers at Work*系列以及Apress的*Founders at Work*[①]（该书讲述如何创办技术公司，而本书讨论计算机编程）相呼应。我意识到编程涵盖的范围太广了，而“编码”（coding）则可以特指其中一个很窄的部分。我个人从不认为一个糟糕的程序员会是一个优秀的编码者，也不相信好的程序员会不是出色的设计者、沟通者和思考者。毋庸置疑，这些受访者都是优秀的编码者、程序员、设计者和思考者，而且还不仅仅如此。我相信接下来你在阅读他们的谈话内容时一定能够体会到这一点。希望你能喜欢这本书！

（编辑：谢灵芝）

① 中文版《创业者》由机械工业出版社出版。——编者注

致　谢

李清　译

首先我要感谢我的受访者们，他们为采访慷慨奉献了许多时间，没有他们，本书就只能是一本充满未答问题的小册子。另外要特别感谢Joe Armstrong和Bernie Cosell两家人，他们分别为我在斯德哥尔摩和弗吉尼亚提供了住所。还要特别感谢的是Peter Norvig和Jamie Zawinski，他们不仅接受我的录音采访，还帮我介绍和联系其他受访者。

我在周游世界做采访时，还收到了其他几个家庭的邀请，在此我要感谢他们的热情好客，包括波士顿的Dan Weinreb和Cheryl Moreau，英国剑桥的Gareth McCaughan和Emma McCaughan夫妇，以及我自己的父母，他们为我在纽约市的活动提供了极大便利。在采访的闲暇时间，Christophe Rhodes邀我游览了剑桥大学。他和Dave Fox还陪我吃晚饭，泡剑桥的酒吧。

Dan Weinreb不仅在波士顿招待我，并且还是本书最为勤奋的审稿人，在我准备采访名单的时候，他就开始帮我把关了。Zach Beane、Luke Corrie、Dave Walden和我的母亲也阅读了一些章节，及时给我鼓励。Zach长久以来一直为我的书提供帮助，这次则帮我想出了本书的副标题。Alan Kay建议我去采访Dan Ingalls和L Peter Deutsch，这个提议非常好。Scott Fahlman为我提供了一些有关Jamie Zawinski早期职业生涯的宝贵背景材料。Dave Walden为我寄来了一些有关Bolt Beranek和Newman的历史材料，有助于我准备对Bernie Cosell的采访。对于那些无法在此一一提及的人，我也要表示谢意和歉意。

感谢Apress的朋友，尤其是Gary Cornell，是他首先提议我来写这本书。感谢John Vacca和Michael Banks，他们提了许多好的建议。还要感谢我的文字编辑Candace English，她帮我修改了数不清的错误。

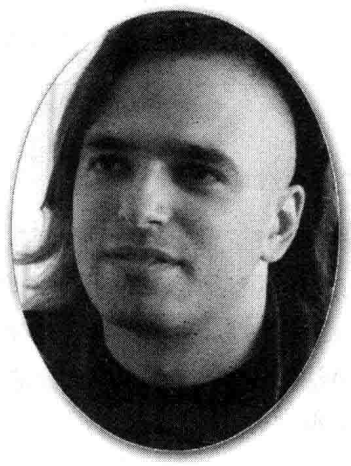
最后要深深地感谢我所有的家人。我母亲和岳母经常来帮我照顾小孩，让我腾出手做更多工作。我父母接我的妻子和孩子住了一星期，使我能够再做一次冲刺。最最感谢的是我的妻子Lily和女儿Amelia，她们是我工作的动力，我爱她们。

（编辑：谢灵芝）

目　录

1

Jamie Zawinski



李琳骁 译

名字的三字母简写与全名同样知名的黑客并不多，Lisp黑客、Netscape早期开发者和夜总会老板Jamie Zawinski，又称jwz，便是其中之一。

Zawinski十几岁就开始编程，当时受雇于卡内基·梅隆大学（CMU）人工智能实验室，从事Lisp开发。他在大学没待多久就选择了退学，因为他发现自己厌恶大学。随后近十年他一直投身Lisp和人工智能（AI）领域，阴差阳错地浸染于一种日渐式微的黑客亚文化中，而同年龄段的其他程序员则是伴着微型计算机一起成长。

Zawinski曾在加州大学伯克利分校（UC Berkeley）为Peter Norvig工作过，后者形容他是“自己雇过的最优秀的程序员”。后来Zawinski去了Lisp公司Lucid，最终领导开发了Lucid Emacs。Lucid Emacs后来更名为XEmacs，终成一大Emacs流派，堪称最著名的开源分支之一。

1994年，Zawinski最终离开了Lucid公司和Lisp领域。随后他加入当时羽翼未丰的初创公司Netscape。他是Netscape浏览器Unix版本及其后Netscape

邮件阅读器最初的开发人员之一。

1998年，作为主要推动者之一，Zawinski与Brendan Eich一道，通过mozilla.org促成了Netscape浏览器的开源。一年后，因对发布遥遥无期备感失望，他退出了该项目，在旧金山买了一家夜总会，这就是他现在运营的DNA Lounge。目前，他正集中精力与加州酒类管制局打官司，力争让这家夜总会成为各年龄层都能进入的现场音乐表演场所。

在这次访谈中，我们谈到C++为什么令人厌恶，几百万人使用其软件给他带来的快乐，以及新手程序员多动手实践的重要性。

Seibel：你是怎么开始学习编程的？

Zawinski：哇，多久以前的事了，都快没什么印象了。没记错的话，我第一次真正使用计算机编程大概是在八年级。当时学校里有几台TRS-80[①]，我们边玩边学了点BASIC。我记不清是不是专门开了门课，印象里好像只是课后摆弄。我记得那些机器没法保存程序，只能照着杂志或手册什么的，将程序逐行敲进去。当时我看了很多书。书中讲到的一些计算机语言，我没办法实际运行，只好在纸上编写那些语言的程序。

Seibel：你都学了哪些语言？

Zawinski：我记得其中一门是APL。我读了一篇讲APL的文章，觉得它非常精妙。

Seibel：嗯，只在纸上写程序，倒是省得配专用的键盘了。你念高中时上过计算机方面的课吗？

Zawinski：高中时我学过Fortran，仅此而已。

Seibel：后来你是怎么开始接触Lisp的？

Zawinski：我看了许多科幻小说，觉得人工智能实在太迷人了，计算机将统治世界。为此我学了点那方面的东西。我高中时有个朋友叫Dan Zigmond，当时我们俩互相换书看，于是一起学习Lisp。有一次，他去参加Apple用户组在卡内基·梅隆大学举办的活动。所谓活动其实就是大家聚在一起交换

① 这是Tandy公司于20世纪七八十年代推出的桌面微型机产品线，拥有QWERTY键盘，体积小，支持浮点BASIC编程语言。

软件，而我朋友就是想去搞点免费的东西。在那里，他还找了个大学生模样的人搭话，那个大学生说："喂，大伙来看，这里有个15岁的孩子会Lisp，真是少见。你该去找Scott Fahlman要份活干。"Dan就照做了。而Fahlman还真给了他一份活。随后Dan又说："对了，我有个朋友你也一起要了吧。"他指的就是我。Fahlman就那么雇了我们。我猜他大概是这么想的，哇哦，有两个高中生居然对这东西感兴趣，让他们在实验室里晃荡也不会有什么大碍。于是我们开始做些简单的活，比如用新版编译器重新编译整套代码。那段经历真是棒极了，就我们两个小毛孩，置身于一群研究语言和人工智能的研究生当中。

Seibel：你是在卡内基·梅隆大学才第一次有机会真正跑Lisp?

Zawinski：我想是的。我记得我们还玩过跑在苹果机上的XLISP。不过那好像是后来的事。我在CMU学会了怎样真刀真枪地编程，那时我们用的机器是PERQ工作站，它是Spice项目的一部分，使用的语言是Spice Lisp，后来演变成CMU Common Lisp。当时的环境非常奇特。那时候每周开一次例会，我们就在一边旁听，来学习软件开发是怎么一回事。不过当时那个组里有几个很有意思的"怪人"。比如Rob MacLachlan，算是我们的主管。他身材高大，满头金发，貌似野人，样子有些吓人。他平时话不多。我大部分时间都会坐在开放式隔间里干活，做点杂事，写些Lisp程序。他时常会慢慢踱进来，手拿装满啤酒的陶瓷马克杯，光着脚，然后就站在我身后。我会打声招呼，而他要么咕哝几声要么一言不发，只是站在那里看我敲键盘。有时，我正干着活，他会突然来上一句："噗哧，错了!"说完就走开了。我的感觉就像是被抛入深渊。这倒与禅法颇为相近，大师点到为止，接下来必须自己参悟。

Seibel：我给Fahlman发过邮件，他说你很有天赋，学得非常快。不过他还提到你有些不守纪律。他的原话是："我们曾委婉地教他如何与组内其他成员共事，怎么编写清晰的代码，好让自己或其他人过一个月后仍能看懂。"你还记得他们当时是怎么教你的吗?

Zawinski：过程不记得了。当然，编写自己回头还能理解的代码，这点至关重要。不过，我都快39了，而当时只有15岁，实在记不太清了。

Seibel：去CMU干活是从哪一年开始的?

Zawinski：不是1984年就是1985年。大概是从十到十一年级之间的那个夏天开始的。下午4点左右学校放学后，我会直接去那里，一直待到晚上八九点。

好像不是每天都去，反正前前后后在那里待了很长时间。

Seibel：高中毕业后，你自然是去了CMU。

Zawinski：是的。事情是这样的，我讨厌高中，那是我生命中最糟糕的日子。所以临近毕业时，我去找Fahlman要一份全职工作，他回答说："不大好办，不过我有几个朋友刚开了家公司，可以找他们谈谈。"那家公司叫Expert Technologies，也就是ETI。我猜他是董事之一。他们正在打造一个专家系统，可以自动给电话簿标页码。他们使用Lisp开发，我认识其中几个人，之前都在Fahlman的小组里待过。他们雇了我，一切顺风顺水，约莫过了一年，我开始惶恐不安：哦，天哪，得到这两份工作完全是撞大运，绝不会有下次了。一旦丢了这份工作，没有大学文凭的话，我就只能去打打零工了，看来我应该去拿张文凭。

我原本计划半工半读，一边到ETI上班，一边求学。结果却变成全工全读，前后持续了大概6个或9个礼拜。反正那段时间不短，以致我错过了退课截止期，最后学费一个子儿也没要回来。不过我上大学的时间又不够长，没拿到学分，因此要说我没真正上过大学，我也只能认了。

那段时间真的很糟。上高中时，所有人都自我安慰说："净是些没完没了的老掉牙的标准化测试，上了大学，一切都会好起来的。"结果上大学第一年，跟高中毫无区别。"哦，放心，等你念了研究生，一切都会好起来的。"所以在我看来，大学和高中一样糟糕，换了时间而已，我可受不了。每天早上8点钟起床，就开始往脑子里塞东西。比如，有门叫做外设介绍的课还非上不可，这门课教你怎么用鼠标。我找到他们说："我都在这所大学里工作了一年半，我知道鼠标怎么用。"但所有人都得上，概莫能外，"这是规定"。其他也都差不多，我实在无法忍受，索性退学了事。我觉得自己做得很对。

我在ETI干了大概4年，后来公司开始走下坡路。当时ETI用的Lisp机器是TI Explorer，那时我除了做专家系统的开发工作外，还把大块时间用在捣鼓用户界面上，还有那些Lisp机器的工作机制，我也从里到外学了个遍。我喜欢那些机器，我喜欢折腾操作系统，琢磨各个部分如何融为一体。

那时我已经写了不少代码，便找了个新闻组，发帖子找工作，还顺带提到自己写过不少代码。Peter Norvig[①]看到帖子后安排了面试。我当时的女友已经搬到加州大学伯克利分校求学，我正好可以随她而去。

① Google研发总监，《十年编程无师自通》一文作者，本书第8篇主人公。

Seibel：Norvig当时在伯克利？

Zawinski：是啊。那份工作很奇特。他们有一大群研究生在做自然语言理解方面的研究，大家基本上都是语言学家，偶尔写些程序。因此他们打算找个人接手他们编写的那些零零碎碎的代码，并整合成真正能用的东西。

这活儿对我来说相当困难，因为我没有相关背景，无法理解他们到底在做些什么。因此常常碰到这样的情形：我盯着某样东西，但完全不知所措。我不理解那是什么意思，不知道下一步该做什么，也不了解要读些什么才能真正理解它。于是我跑去问Peter。他很礼貌地回应我："你现在理解不了，这很正常，周二我有时间，到时给你讲解一下。"结果我就无所事事。于是我把大块时间都用在折腾窗口系统、摆弄屏幕保护程序以及之前出于好玩而捣鼓的那些用户界面之类的程序上。

就这么过了6个月或8个月，然后我意识到自己完全是在虚掷光阴。我什么都没为他们做，觉得自己就像在度假。后来有几次我真的忙得一塌糊涂，那种时候回想起在伯克利的那段日子，我就问问自己："你怎么会放弃那份度假般的工作？不会是脑袋短路了吧？他们可是付钱让你写屏保的！"

最后我去了Lucid[①]，当时仅存的两家Lisp环境开发商之一。我决定离开伯克利的主要原因是我觉得自己一事无成，那种感觉很糟。我周围的人都不是程序员。当然他们都不赖，我仍和其中几个人保持着友谊。只不过他们都是语言学家，比起解决实际问题来，他们对抽象的事物感兴趣得多。而我想做出点实际的东西，有一天就能指着某样东西说："瞧，这活儿漂亮吧，是我干的。"

Seibel：你在Lucid的工作成果是XEmacs，不过，你去那里一开始就是做Lisp方面的开发吗？

Zawinski：是的，我在那里做的第一个项目就是用Lisp，哦，我都记不得是什么机器了，不过应该是台有着16个处理器的并行计算机，我们使用的Lucid Common Lisp变体提供了几个控制结构，可以将创建的进程分别部署到不同的处理器上。

我做了一些后端优化工作，主要是减少创建线程的开销，从而让那台机

① 由Richard P. Gabriel于1984年创办，1994年破产，Lucid Common Lisp的所有权被Harlequin收购，后者于1999年被Global Graphics收购，随后Global Graphics将Lucid Common Lisp相关权利卖给Xanalys公司，由此催生了LispWorks公司，现在仍以Liquid Common Lisp为名在出售Lucid Common Lisp。

器可以完成有用的计算，比如实现并行Fibonacci算法，而不用再把时间耗费在为每个线程新建栈组（stack group）上。我非常享受整个过程。那是我第一次有机会使用那么奇特的机器。

在这之前我还负责把Lisp迁移到新机器上。大致过程就是，有人已经针对新架构写好了编译器后端，并且已编译好自举代码。我会拿到这个二进制文件，据称是针对这台机器的可执行代码，接着我必须剖析它们的装载器格式，以便写个简单的C程序，装载拿到的文件，将页面置为可执行，并跳转到那个页面。幸运的话，你就会看到Lisp提示符，之后即可开始手工装载其他东西。

对任何架构来说，这都是件难事，因为装载器几乎没什么对路的文档。你只能找个C程序编译一把，然后用Emacs逐字节分析，编辑其中的字节，尝试把某个字节改成零，看看有什么结果，程序会不会停止运行。

Seibel：你刚才提到它没什么对路的文档，指的是文档不准确，还是根本就没有文档？

Zawinski：通常都有文档，不过往往都是错的。或者文档太过陈旧，讲的是三个版本之前的东西，天知道。而遇到问题，往往就是要追究精确细节的时候。有时只是略微改了一下某个文件，装载器就不再认为那是个可执行文件了，你必须弄清楚是怎么回事。

Seibel：这种事无处不在，从最底层的系统编程到上层API，总有些事跟你预想的不一样，或者不像文档描述的那样。碰到这种情况，你是怎么处理的？

Zawinski：是啊，这种事情你得料到迟早会来的。你越早意识到自己手里的地图是错的，就能越快发现哪儿走错了。就拿我前面碰到的情况来说，如果要生成一个可执行文件，那好，我知道C编译器会生成一个。挑个正常的可执行文件，不断破坏，直到不可用为止。这是逆向工程的基本方法。

回想起来，我修正过的最难对付的bug，大概就是那段时间遇到的。经过一番努力，可执行文件已经能够运行，它试着引导装载Lisp，并载入了500条指令，之后就崩溃了。于是我不得不靠着S键，单步调试，试图找出崩溃的位置。不过，每次崩溃的位置似乎都不一样，摸不着规律。我开始研读这个自己并不熟悉的体系结构的汇编输出。最后我回过神来："天哪！当我单步调试时，可执行文件的执行顺序是不同的。那个bug有可能是时序相关的。"最终，我找到了真正的原因，那是台支持投机执行（speculative execution）的早期机器，它会执行两条分支路径。而在单步调试分支指令时，GDB总