EFFECTIVE 系列丛书

HZ BOOKS 华章IT

Pearson

Google高级软件工程师Diomidis Spinellis融合自己30多年系统开发和软件调试的实战经验，深入探讨调试的策略、方法、工具和技巧

涵盖调试工作的方方面面，汇聚66条专业调试技巧，分步骤详细讲解，包含大量实用范例代码

Effective Debugging

66 Specific Ways to Debug Software and Systems

Effective Debugging

软件和系统调试的66个有效方法

[希] 迪欧米迪斯·斯宾奈里斯（Diomidis Spinellis）著

爱飞翔 译

机械工业出版社 China Machine Press

EFFECTIVE
系列丛书

Effective Debugging
66 Specific Ways to Debug Software and Systems

Effective Debugging

软件和系统调试的66个有效方法

[希] 迪欧米迪斯·斯宾奈里斯（Diomidis Spinellis）著
爱飞翔 译

机械工业出版社
China Machine Press

图书在版编目（CIP）数据

Effective Debugging：软件和系统调试的 66 个有效方法 /（希）迪欧米迪斯·斯宾奈里斯（Diomidis Spinellis）著；爱飞翔译 . —北京：机械工业出版社，2017.6
（Effective 系列丛书）
书名原文：Effective Debugging: 66 Specific Ways to Debug Software and Systems

ISBN 978-7-111-56889-6

I. E… II. ①迪… ②爱… III. 调试软件 IV. TP311.562

中国版本图书馆 CIP 数据核字（2017）第 090186 号

本书版权登记号：图字：01-2016-8666

Effective Debugging
软件和系统调试的 66 个有效方法

出版发行：机械工业出版社（北京市西城区百万庄大街 22 号　邮政编码：100037）
责任编辑：关　敏　　责任校对：殷　虹
印　　刷：三河市宏图印务有限公司　　版　　次：2017 年 6 月第 1 版第 1 次印刷
开　　本：186mm×240mm　1/16　　印　　张：14
书　　号：ISBN 978-7-111-56889-6　　定　　价：59.00 元

凡购本书，如有缺页、倒页、脱页，由本社发行部调换
客服热线：（010）88379426　88361066　　投稿热线：（010）88379604
购书热线：（010）68326294　88379649　68995259　　读者信箱：hzit@hzbook.com

The Translator's Words 译 者 序

十几年前，我看过 Spinellis 先生所写的《代码阅读》(Code Reading) 和《代码质量》(Code Quality) 两本书，特别喜欢这种全面讲解编程工作中某个领域的教程，这次又读到同一位作者所写的《Effective Debugging》，感觉依然很精彩。

这是一本在思路和技巧上都较为丰富的调试手册。

从思路方面来说，本书介绍了许多宏观与微观的调试办法。例如，在面对软件故障时，既可以从整体情况入手，进行自上而下的调试，也可以从具体故障入手，进行自下而上的调试，还可以考虑用高级的抽象机制、便捷的程序库、直白的算法、简洁的逻辑，乃至另外一门更为合适的编程语言，对 bug 繁多的代码进行改写——这些思路，都能在调试工作中给人以启发。

从技巧方面来说，本书介绍了众多的调试工具与手法：有些可以在程序运行之前，设置断点并对表达式与程序的状态做出断言；有些可以在程序运行之中，将各种调试机制与程序进行连接，并对其执行情况进行记录；有些则可以在程序崩溃之后，通过核心转储等信息来还原当时的情境。此外，作者还讲解了如何把这些前置、中置和后置技巧与版本控制系统、静态分析工具、动态分析工具及性能测评工具结合起来使用，以提升调试的效率。

全书的 8 个大类中含有 66 条技巧，这些技巧都是围绕着“重现 bug——探查 bug——解决 bug”这一主线而展开的。针对这三个阶段，作者进行了详细的分步讲解，给出了很多实用的范例代码与建议，而且特别强调了如何才能稳定地捕获并重现 bug，以便给后面两个阶段打下良好的基础。

本书或许还能促使大家思考另外一个问题，那就是：在修复完 bug 之后，怎样防止有人向程序中引入类似的 bug？这可以从代码质量与测试两方面入手。提高代码质量，

能够减少程序员对代码的误解，进而降低引入 bug 的概率；而对测试进行完善，则能够提前捕获很多问题，从而不会使这些问题逐渐积累成复杂的 bug。这些理念，作者在书中也时常会提到。

总之，这是一本可以打开思路并拓宽眼界的书籍，大家不妨在自己惯用的调试环境之外，多尝试一下作者所介绍的其他技法、工具和语言，以求达到旧学与新知的融合。

翻译本书的过程中，我得到了机械工业出版社华章公司诸位编辑和工作人员的帮助，在此深表谢意。

由于译者水平有限，不足与疏漏之处请大家发邮件至 eastarstormlee@gmail.com，或访问 github.com/jeffreybaoshenlee/debugging-errata/issues 留言，给我以批评和指教。

爱飞翔

Preface 前　言

我们在开发软件或对运行软件的系统进行管理的时候，经常会遇到故障。有些故障是因代码问题而引发的编译错误，这种故障可以在短时间内修复；还有一些故障则会使大型系统停机，这将给公司带来每小时数百万的损失（具体货币单位依情况而定）。要想成为一名优秀的专业人士，你就必须在发生故障时迅速找出背后的原因并加以修复。这正是调试的意义所在，也是本书所要谈论的主题。

本书是写给有一定经验的开发者看的，而不是一本介绍性质的读物。它假设读者能够理解用各种编程语言所写成的代码片段，并且会使用高级的 GUI 编程工具以及基于命令行的编程工具。另一方面，我会在书中详细描述调试技巧，因为我发现：即便是对某些开发方法很有经验的编程专家，也依然需要一些手把手的指导，才能够掌握其他的开发方法。此外，如果你已经花了至少几个月时间来调试一些颇具规模的软件，那么应该会更容易理解书中某些高级技巧所适用的场合。

本书所涵盖的范围

本书所要讲解的调试知识，包括与调试有关的策略、工具及方法。我们当前在开发并运作一款复杂的计算系统时，可能会遇到各种问题，而这些调试知识，则使大家能够应对这些问题。过去我们所说的调试，主要是指检测并修复程序错误，而当前却很少有哪个程序会孤立地运作，即便是一个很小的程序，也会与外部的程序库相链接（通常是动态链接）。更为复杂的程序会运行在应用程序服务器中，会调用 Web 服务，会使用关系型数据库及 NoSQL 数据库，会从目录服务器上获取数据，会运行外部的程序，会利用其他的中间件，也会纳入很多第三方的软件包。于是，要想令整个系统及服务正常地运作，就必须确保其中的组件不会发生故障，这些组件可能是由公司内部人员所开发

的，也可能是由第三方所提供的，它们所在的主机或许分布在全球各地。为了应对这种局面，软件开发行业开始重视 DevOps 规程，这套规程旨在同时强调开发者和其他 IT 专业人员所应担负的职责。与之类似，本书想使读者在面对故障时也能够具备这样一种全面的观念，因为在面对一些极为困难的问题时，我们通常无法立刻判断出该问题到底是由哪一个软件组件所引发的。

本书的内容按照从一般到特殊的顺序来进行安排。首先讲解调试策略（第 1 章）、调试方法（第 2 章）以及调试时所用的工具与技术（第 3 章），这些知识使我们能够应对各种软件故障及系统故障。接下来，讨论在调试工作的各个阶段所用到的具体技巧，也就是在使用调试器（第 4 章）、编写程序（第 5 章）、编译软件（第 6 章）以及运行系统（第 7 章）时所用到的调试技巧。与多线程和并发有关的 bug 是很难寻找的，所以最后我们专门用一章（第 8 章）来讲解特定的调试工具及调试技术，使大家能够找出这些 bug。

怎样运用书中的内容

你可以从第一页读起，一页一页往后翻，直到看完。但是别急，其实还有更好的读法。书里给出的建议可以分成以下三种。

- **策略与方法**。这些内容包括我们在面对故障时所应具备的知识以及所应采取的做法。本书第 1 章和第 2 章里面的内容就属于这一类，此外，第 5 章中的很多技巧也可以归入此类。阅读并理解了这些内容之后，你需要在工作中对其加以运用，以便逐渐养成习惯。调试程序的时候，我们需要系统地反思自己所用的办法，如果某个办法行不通，那就应该把自己所经历的路线回顾一遍，这样可以帮助我们发现解决该问题的其他办法。
- **技巧与工具**。这些内容值得大家投入时间去学习，它们主要出现在第 3 章里面，其他章节中的某些内容（如第 36 条）也同样可以归为这一类，我们可以在日常工作中运用这些内容来解决问题。大家应该花时间去学习这些内容，并且要逐步实践它们。这或许意味着我们要放弃自己所熟悉的调试工具，而去使用一些学习曲线较为陡峭但是功能上更加先进的调试工具，那些工具虽然一开始学起来比较困难，可是从长远来看，却能够帮助你成为调试方面的专家。
- **调试的思路**。当我们遇到困难时，可以根据这些思路来找寻合适的技巧。这些内容不一定每天都会用到，但是当你遇到一个琢磨不透的问题时，它可以帮助

你节省一整天（或者说至少几小时）的时间。比如，如果你不清楚自己所写的C和C++代码为什么无法编译，那么第50条或许能给你一些启发。大家应该快速浏览这些内容，使自己意识到它们可以在某些场合派上用场，等到真正需要使用它们的时候，再去详细研究。

本书对软件开发的其他方面所起的作用

本书里的所有条目都是针对故障的诊断与调试而写的，不过其中有很多建议同样可以用来缩减代码中的bug数量，并且可以使你在遇到这样的bug时能够更为迅速地将其修复。严谨的调试技术与优秀的软件开发方式之间能够形成良性的循环，因此，书中的建议对于你当前或者将来要面对的软件设计、软件构建以及软件管理工作，是可以起到帮助作用的。

设计软件的时候，应该遵循下列建议：

- 使用与其角色相称的高级机制（参见第47条和第66条）。
- 提供调试模式（参见第6条和第40条）。
- 提供对系统操作进行监控与记录的机制（参见第27条、第41条和第56条）。
- 提供一个选项，使得开发者可以用Unix命令行工具来编写与组件有关的脚本（参见第22条）。
- 把内部的错误暴露出来，使其表现为软件故障，而不要将其隐藏起来，使其成为软件中的不稳定因素（参见第55条）。
- 提供一种方式，使得开发者能够在软件发生故障之后获得内存转储（memory dump）信息（参见第35条和第60条）。
- 从数量和范围方面，尽量缩减软件在执行时的不确定因素（参见第63条）。

构建软件的时候，应该遵循下列建议：

- 征求同事的意见（参见第39条）。
- 为你所编写的每个例程创建单元测试（参见第42条）。
- 用断言来验证自己所做的假设是否成立，以及代码的功能是否正确（参见第43条）。
- 尽量把代码写得易于维护，也就是要写出易读、稳定且便于分析和修改的代码（参见第46条和第48条）。
- 在构建程序时避免不确定的因素（参见第52条）。

在对软件的开发及运作进行管理时，应该遵循下列建议（无论是要管理一个团队，还是只管理自己的流程）：

- 用适当的事务追踪系统，把遇到的问题记录下来（参见第 1 条）。
- 对各种有待处理的事务进行分类，并排定其优先次序（参见第 8 条）。
- 把对软件所做的修改适当地记录在修订管理系统中，并且对该系统进行较好的维护（参见第 26 条）。
- 渐进地部署软件，使得我们可以在新旧版本之间进行对比（参见第 5 条）。
- 尽量采用各种不同的工具来开发，并试着把程序部署在各种环境中（参见第 7 条）。
- 经常对工具与程序库进行更新（参见第 14 条）。
- 如果使用了第三方的程序库，那么可以考虑购买该程序库的源代码（参见第 15 条）；考虑购买一些较为完善的工具来锁定那些不太容易找到的错误（参见第 51 条、第 59 条、第 62 条、第 64 条及第 65 条）。
- 寻找专门的工具来调试硬件接口及嵌入式系统（参见第 16 条）。
- 使得开发者能够远程调试软件（参见第 18 条）。
- 对于消耗资源较多的故障诊断任务来说，要留出足够的 CPU 及磁盘资源（参见第 19 条）。
- 鼓励开发者之间通过代码评审及编程指导等手段进行协作（参见第 39 条）。
- 鼓励大家进行测试驱动开发（参见第 42 条）。
- 在构建软件的时候，要做性能分析、静态分析以及动态分析（参见第 57 条、第 51 条及第 59 条），并且要打造一套迅速而高效的构建流程与测试流程（参见第 53 条及第 11 条）。

对书中术语的说明

本书所说的 fault（错误）一词，遵循 ISO-24765—2010（Systems and software engineering——Vocabulary）标准，它的意思是：“计算机程序里面某个不正确的步骤、流程或数据定义。”这也称为 defect（缺陷）。在日常工作中，我们把这叫做 bug。与之类似，本书所用的 failure（故障）一词，也遵循 ISO-24765—2010 标准，它指的是：“因系统或系统组件未能在规定限制之下执行所需功能而引发的事件。”故障可以表现为程序崩溃、程序冻结或是程序的结果有误。故障一词强调的是程序运行的后果，而错误一词强调的则是我们所遇到或使用的某个程序本身有问题。不过有的时候，大家也会用错误及缺陷

这两个词来指代故障，ISO 标准也承认了这一点。笔者在本书中会按照上述定义来区分这几个词，然而在语境比较明确的情况下，我通常也会用问题（problem）一词来指代错误（如“代码有问题”）或故障（如“可以重现的问题”），这样写能够使本书读起来更加流畅，而不至于变成一份法律文件。

Unix 操作系统的 shell、程序库以及工具，目前都可以运行在各种各样的平台上。笔者采用 Unix 一词来指代遵从 Unix 原则及 API 的任何一种系统，包括 Apple 的 Mac OS X、各种 GNU/Linux 发行版（如 Arch Linux、CentOS、Debian、Fedora、openSUSE、Red Hat Enterprise Linux、Slackware 及 Ubuntu）、直接从 Unix 继承而来的系统（如 AIX、HP-UX 及 Solaris）、各种 BSD 衍生系统（如 FreeBSD、OpenBSD 及 NetBSD），以及运行在 Windows 系统上的 Cygwin。

本书中所列出的 C++、Java 或 Python 代码，针对的也是较新的编程语言版本，不过笔者会避开那些奇怪的或是刚刚推出的特性。

书里面会出现“你的代码”和“你的软件”这两种说法，它们指的是你正在调试的代码以及正在开发的软件。这两种说法听起来比较简洁，而且也暗含了一种对代码所有权的宣示，这对于软件开发人员来说是十分重要的。

笔者所说的例程（routine）一词，是指可供调用的代码单元，如成员函数、方法、函数、过程以及子例程等。

Visual Studio 及 Windows 指的是 Microsoft 公司的相关产品。

修订控制系统（revision control system）及版本控制系统（version control system），是指像 Git 这样能够对软件配置进行管理的工具。

排版约定

Unix 命令行选项使用 --this 这样的形式，与该选项等价的单字母选项使用 -t 这样的形式。Windows 工具中的对应选项，使用 /this 这样的形式。

- 按键采用 Shift-F11 这样的形式。
- 文件路径采用 /etc/motd 这样的形式。
- 菜单操作采用 Debug-New Breakpoint-Break at Function 这样的形式。
- 为了缩短篇幅，笔者会省略 C++ 代码中的 std:: 限定符及 std 名称空间。
- 描述 GUI（图形用户界面）工具时，笔者所说的功能针对的是撰写本书时所能找到的最新版本。如果你所使用的是另外一个版本，那么请参照对应的菜单或窗

口来进行操作，也可以在那个版本的文档中查找对应功能的操作办法。值得注意的是，命令行工具的界面几十年来一直都比较稳定，而不像 GUI 工具那样，每个版本都会有一些新的东西。大家可以从这个现象中推出很多结论。

代码及勘误

范例代码及英文原版书的勘误表请参见 www.spinellis.gr/debugging。

Acknowledgements 致　谢

首先感谢本书的编辑——Addison-Wesley 出版社的 Trina Fletcher MacDonald，以及 Effective 系列的编辑 Scott Meyers，他们在本书的推进过程中提供了良好的指导与管理。也要感谢本书的技术评审 Dimitris Andreadis、Kevlin Henney、John Pagonis 及 George Thiruvathukal，他们提供了几百条绝妙的思路、评论及建议，使得本书的品质得以改善。特别要感谢文字编辑 Stephanie Geels，她准确地指出书稿中的问题，并适当地加以修改，这个过程本来是令我心生畏惧的，然而她高超的工作水准使该过程变得很轻松。我也要感谢出色完成生产管理工作的 Melissa Panagos、督导整个成书过程的 Julie Nahil，以及为书籍进行排版的 LaTeX 高手 Lori Hughes。Sheri Replin 提出了编辑意见，Olivia Basegio 管理了本书的技术评审组，Chuti Prasertsith 设计了优秀的封面，Stephane Nakib 指导了营销工作，感谢他们，并向 Alfredo Benso、Georgios Gousios 与 Panagiotis Louridas 致意，他们帮助我确立了本书的早期构想。

书中有 4 条是根据我发表在《 IEEE Software 》期刊“ Tools of the Trade”专栏中的材料扩展而成的：

- 第 5 条来自“Differential Debugging” vol.30, no.5, 2013, pp.19-21.
- 第 22 条来自“Working with Unix Tools” vol.22, no.6, 2005, pp.9-11.
- 第 58 条来自“I Spy” vol.24, no.2, 2007, pp.16-17.
- 第 66 条来自“Faking It” vol.28, no.5, 2011, pp.96, 95.

此外还有 4 条需要说明：

- 第 63 条是根据 Martin Fowler 在《 Eradicating Non-Determinism in Tests 》（2014 年 4 月 14 日）及《 TestDouble 》（2006 年 1 月 17 日）这两篇文章中所提出的想法而写成的。

- 第 48 条所提出的大部分重构，都源自 Martin Fowler 所写的《 Refactoring 》（重构）一书（Addison-Wesley, 1999 ）。
- 第 60 条的写作动力，来自《 Real-World Concurrency 》这篇文章（ Bryan Cantrill and Jeff Bonwick, ACM Queue, October 2008 ）。
- 第 66 条中的 Java 代码是根据 Tagir Valeev 的意见写成的。

雅典经贸大学（ Athens University of Economics and Business）的诸位同事，在我的学术生涯中提供了多方面的慷慨帮助，并且（有时是在无形之中）促成了本书的写作，他们包括：Damianos Chatziantoniou、Georgios Doukidis、Konstantine Gatsios、George Giaglis、Emmanouil Giakoumakis、Dimitris Gritzalis、George Lekakos、Panagiotis Louridas、Katerina Paramatari、Nancy Pouloudi、Angeliki Poulymenakou、Georgios Siomkos、Spyros Spyrou 及 Christos Tarantilis。

调试是一种技艺，要在实践中才能学成，因此，我要感谢过去 40 年来所遇到的诸位同事，感谢他们在我所写的代码中发现 bug、给我提供有效的问题报告、对我的代码做出评审及测试，并且教我怎样避免问题、追查问题及修复问题。下面大致按照从近到远的顺序列出我在职场与学术界所遇到的同事：

- 在 Google 的 Ads SRE FE 团队工作时的同事：Mark Bean、Carl Crous、Alexandru-Nicolae Dimitriu、Fede Heinz、Lex Holt、Thomas Hunger、Thomas Koeppe、Jonathan Lange、David Leadbeater、Anthony Lenton、Sven Marnach、Lino Mastrodomenico、Trevor Mattson-Hamilton、Philip Mulcahy、Wolfram Pfeiffer、Martin Stjernholm、Stuart Taylor、Stephen Thorne、Steven Thurgood 及 Nicola Worthington。
- CQS 的同事：Theodoros Evgeniou、Vaggelis Kapartzianis 及 Nick Nassuphis。
- 当前或者曾经在雅典经贸大学管理科学与技术系进行研究和实验的诸位同事：Achilleas Anagnostopoulos、Stefanos Androutsellis-Theotokis、Konstantinos Chorianopoulos、Marios Fragkoulis、Vaggelis Giannikas、Georgios Gousios、Stavros Grigorakakis、Vassilios Karakoidas、Maria Kechagia、Christos Lazaris、Dimitris Mitropoulos、Christos Oikonomou、Tushar Sharma、Sofoklis Stouraitis、Konstantinos Stroggylos、Vaso Tangalaki、Stavros Trihias、Vasileios Vlachos 及 Giorgos Zouganelis。
- 担任希腊财政部的信息系统秘书长时的诸位同事：Costas Balatos、Leonidas Bogiatzis、Paraskevi Chatzimitakou、Christos Coborozos、Yannis Dimas、

Dimitris Dimitriadis、Areti Drakaki、Nikolaos Drosos、Krystallia Drystella、Maria Eleftheriadou、Stamatis Ezovalis、Katerina Frantzeskaki、Voula Hamilou、Anna Hondroudaki、Yannis Ioannidis、Christos K.K.Loverdos、Ifigeneia Kalampokidou、Nikos Kalatzis、Lazaros Kaplanoglou、Aggelos Karvounis、Sofia Katri、Xristos Kazis、Dionysis Kefallinos、Isaac Kokkinidis、Georgios Kotsakis、Giorgos Koundourakis、Panagiotis Kranidiotis、Yannis Kyriakopoulos、Odyseas Kyriakopoylos、Georgios Laskaridis、Panagiotis Lazaridis、Nana Leisou、Ioanna Livadioti、Aggeliki Lykoudi、Asimina Manta、Maria Maravelaki、Chara Mavridou、Sofia Mavropoulou、Michail Michalopoulos、Pantelis Nasikas、Thodoros Pagtzis、Angeliki Panayiotaki、Christos Papadoulis、Vasilis Papafotinos、Ioannis Perakis、Kanto Petri、Andreas Pipis、Nicos Psarrakis、Marianthi Psoma、Odyseas Pyrovolakis、Tasos Sagris、Apostolos Schizas、Sophie Sehperides、Marinos Sigalas、George Stamoulis、Antonis Strikis、Andreas Svolos、Charis Theocharis、Adrianos Trigas、Dimitris Tsakiris、Niki Tsouma、Maria Tzafalia、Vasiliki Tzovla、Dimitris Vafiadis、Achilleas Vemos、Ioannis Vlachos、Giannis Zervas 及 Thanasis Zervopoulos。

- 在 FreeBSD 项目工作时的同事：John Baldwin、Wilko Bulte、Martin Cracauser、Pawel Jakub Dawidek、Ceri Davies、Brooks Davis、Ruslan Ermilov、Bruce Evans、Brian Fundakowski Feldman、Pedro Giffuni、John-Mark Gurney、Carl Johan Gustavsson、Konrad Jankowski、Poul-Henning Kamp、Kris Kennaway、Giorgos Keramidas、Boris Kovalenko、Max Laier、Nate Lawson、Sam Leffler、Alexander Leidinger、Xin Li、Scott Long、M.Warner Losh、Bruce A.Mah、David Malone、Mark Murray、Simon L.Nielsen、David O'Brien、Johann'Myrkraverk'Oskarsson、Colin Percival、Alfred Perlstein、Wes Peters、Tom Rhodes、Luigi Rizzo、Larry Rosenman、Jens Schweikhardt、Ken Smith、Dag-Erling Smørgrav、Murray Stokely、Marius Strobl、Ivan Voras、Robert Watson、Peter Wemm 及 Garrett Wollman。
- LH Software 及 SENA 的同事：Katerina Aravantinou、Michalis Belivanakis、Polina Biraki、Dimitris Charamidopoulos、Lili Charamidopoulou、Angelos Charitsis、Giorgos Chatzimichalis、Nikos Christopoulos、Christina Dara、Dejan Dimitrijevic、

Fania Dorkofyki、Nikos Doukas、Lefteris Georgalas、Sotiris Gerodianos、Vasilis Giannakos、Christos Gkologiannis、Anthi Kalyvioti、Ersi Karanasou、Antonis Konomos、Isidoros Kouvelas、George Kyriazis、Marina Liapati、Spyros Livieratos、Sofia Livieratou、Panagiotis Louridas、Mairi Mandali、Andreas Massouras、Michalis Mastorantonakis、Natalia Miliou、Spyros Molfetas、Katerina Moutogianni、Dimitris Nellas、Giannis Ntontos、Christos Oikonomou、Nikos Panousis、Vasilis Paparizos、Tasos Papas、Alexandros Pappas、Kantia Printezi、Marios Salteris、Argyro Stamati、Takis Theofanopoulos、Dimitris Tolis、Froso Topali、Takis Tragakis、Savvas Triantafyllou、Periklis Tsahageas、Nikos Tsagkaris、Apostolis Tsigkros、Giorgos Tzamalis 及 Giannis Vlachogiannis。

- 欧洲计算机产业研究中心（European Computer Industry Research Center，ECRC）的同事：Mireille Ducassé、Anna-Maria Emde、Alexander Herold、Paul Martin 及 Dave Morton。
- 伦敦帝国学院（Imperial College London）计算机科学系的同事：Vasilis Capoyleas、Mark Dawson、Sophia Drossopoulou、Kostis Dryllerakis、Dave Edmondson、Susan Eisenbach、Filippos Frangulis Anastasios Hadjicocolis、Paul Kelly、Stephen J.Lacey、Phil Male、Lee M.J.McLoughlin、Stuart McRobert、Mixalis Melachrinidis、Jan-Simon Pendry、Mark Taylor、Periklis Tsahageas 及 Duncan White。
- 加州大学伯克利分校计算机科学研究组（Computer Science Research Group，CSRG）的同事：Keith Bostic。
- Pouliadis&Associates 公司的同事：Alexis Anastasiou、Constantine Dokolas、Noel Koutlis、Dimitrios Krassopoulos、George Kyriazis、Giannis Marakis 及 Athanasios Pouliadis。
- 在各种会议及各种场合认识的诸位朋友：Yiorgos Adamopoulos、Dimitris Andreadis、Yannis Corovesis、Alexander Couloumbis、John Ioannidis、Dimitrios Kalogeras、Panagiotis Kanavos、Theodoros Karounos、Faidon Liampotis、Elias Papavassilopoulos、Vassilis Prevelakis、Stelios Sartzetakis、Achilles Voliotis 及 Alexios Zavras。

最后感谢家人多年来对我的支持，以及在我撰写本书时所给予的鼓励，有时我要在家里调试系统，甚至放弃度假和周末休息时间而去写稿。尤其感谢 Dionysis 帮我绘制图 5.2，并感谢 Eliza 和 Eleana 帮我选定本书的封面。

Contents 目　录