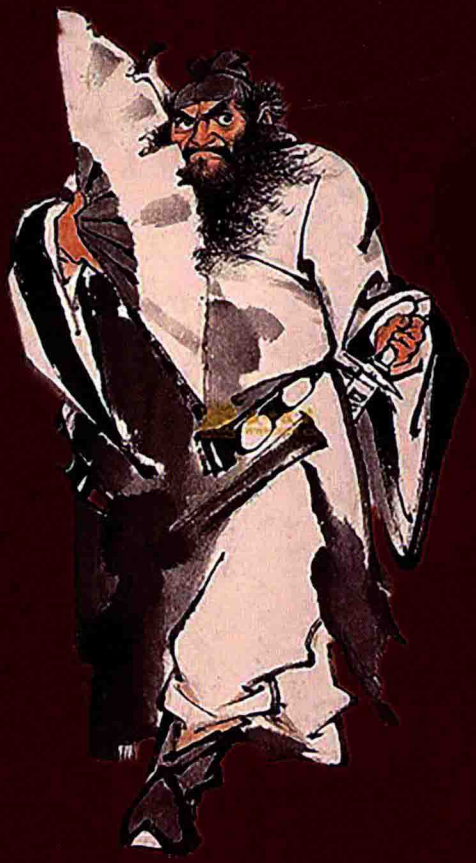
软件调试领域的经典著作，十年沉淀，新版再现

软件工程师的贴身宝典，向超一流技术高手进阶的武功秘籍



软件调试

（第2版）

卷1：硬件基础

张银奎 著

中国工信出版集团
人民邮电出版社
POSTS & TELECOM PRESS

软件调试

（第2版）

卷1：硬件基础

张银奎 著

人 民 邮 电 出 版 社

北 京

图书在版编目（CIP）数据

软件调试 : 第2版. 卷1, 硬件基础 / 张银奎著. --
北京 : 人民邮电出版社, 2018.12
ISBN 978-7-115-49250-0

Ⅰ. ①软… Ⅱ. ①张… Ⅲ. ①调试软件 Ⅳ.
①TP311.562

中国版本图书馆CIP数据核字(2018)第203145号

内 容 提 要

本书堪称是软件调试的“百科全书”。作者围绕软件调试的“生态”系统（ecosystem）、异常（exception）和调试器3条主线，介绍软件调试的相关原理和机制，探讨可调试性（debuggability）的内涵、意义以及实现软件可调试性的原则和方法，总结软件调试的方法和技巧。

第1卷主要围绕硬件技术展开介绍。全书分为4篇，共16章。第一篇“绪论”（第1章），介绍了软件调试的概念、基本过程、分类和简要历史，并综述了本书后面将详细介绍的主要调试技术。第二篇“CPU及其调试设施”（第2～7章），以英特尔和ARM架构的CPU为例系统描述了CPU的调试支持。第三篇“GPU及其调试设施”（第8～14章），深入探讨了Nvidia、AMD、英特尔、ARM和Imagination这五大厂商的GPU。第四篇“可调试性”（第15～16章），介绍了提高软件可调试性的意义、基本原则、实例和需要注意的问题，并讨论了如何在软件开发实践中实现可调试性。

本书理论与实践紧密结合，既涵盖了相关的技术背景知识，又针对大量具有代表性和普遍意义的技术细节进行了讨论，是学习软件调试技术的宝贵资料。本书适合所有从事软件开发工作的读者阅读，特别适合从事软件开发、测试、支持的技术人员，从事反病毒、网络安全、版权保护等工作的技术人员，以及高等院校相关专业的教师和学生学习参考。

◆ 著　　张银奎
责任编辑　陈冀康
责任印制　焦志炜

◆ 人民邮电出版社出版发行　　北京市丰台区成寿寺路11号
邮编　100164　　电子邮件　315@ptpress.com.cn
网址　http://www.ptpress.com.cn
涿州市京南印刷厂印刷

◆ 开本：787×1092　1/16
印张：32
字数：781千字　　2018年12月第1版
印数：1－4 000册　　2018年12月河北第1次印刷

定价：118.00元

读者服务热线：(010)81055410　印装质量热线：(010)81055316
反盗版热线：(010)81055315
广告经营许可证：京东工商广登字20170147号

为读者出好书
与作者共成长

历史回眸

我是 1949 年进入麻省理工学院（MIT）的。就在那一年，第一台存储程序计算机在英国的剑桥和曼彻斯特开始运行。我的一个本科同学 Kenneth Ralston 是学数学的，他偶尔会和我如痴如醉地谈起一台神秘的机器，说这台机器当时正在 MIT 附近的 Smart 街上的 Barta 楼内组装。我的好奇心后来在 1954 年的秋天得到了满足，那时我开始学习我的第一门计算机课程“数字计算机编码与逻辑”。那门课程是 Charles Adams 教的，他是自动编程（现在称为编译）领域的先锋。当时使用的机器叫作“旋风”，被放置在一间充满了真空管电路的房间内。它由美国海军投资建立，用来研究飞机模拟。

因为我的知识背景及我所完成的电子工程专业的硕士课程，一个助研基金约请我在旋风计算机上用“最速下降法”解决一个最优化问题。这让我彻底熟悉了那一套烦琐的程序准备工作。我们以旋风机器的汇编语言编写程序，然后使用 Friden 电传打字机将以字符和数字表示的代码以打孔的方式输出到纸带上。纸带是用一个 Ferrante 光电读出器读入计算机的，然后交给“综合系统 2”的“系统软件”进行处理。处理结果是一个二进制纸带，以大约每秒钟 10 行的速度打孔出来，每行代表一个 6 位字符。而后，用户可以调用一个简单的装载程序（装载程序是保存在几个可以来回交换的内存单元中的）将二进制的纸带装入 2048 字的内存中，之后就期待着程序的正常运行。用户也可以在控制台的电传打字机上调用“综合系统”的输出例程来把结果打印出来，或者把它们写到一个原始的磁带单元中，留待以后离线打印。

那时最漂亮的输出设备是 CRT 显示屏，用户可以在上面一个点一个点地画出图表和图片。上面配备了一部照相机，可以把显示的图片录制在胶片上。系统程序员已经开发好了“崩溃照相”功能，可以把程序出错时内存中的内容显示在 CRT 显示屏上。用户可以在第二天早上取到显影后的胶片，然后用一个缩微胶卷阅读器来研究上面的八进制数字。在那时，这是调试旋风程序的主要方法，除此之外，就是把中间结果打印出来。

大多数我们这样的普通用户不知道的是，在 Barta 楼里有一个后屋，在那里第一个基于计算机的飞机跟踪和威胁检测系统上的分类工作正在进行。那里放置了一些更先进的设备，有很多台 PPI（计划和位置标识器）显示器，并且已经开发出了第一个定点设备——光笔，用来跟计算机实时交互。

旋风计算机最初的主内存是威廉斯管型的，这还不足以满足实时操作的可靠性标准。这一需求带动了相关研究工作并促进了磁心内存的产生。旋风工程师建造了一个非常简单的计算机，称作内存测试机（MTC），用来测试新的内存。因为新内存表现良好，所以立刻把它安装在旋风计算机上，而后 MTC 也就功成身退了。

旋风计算机上的工作促使了 MIT 林肯实验室的成立，实验室的主要责任是基于旋风计算机上的实时系统技术开发一个美国国家空中防御系统。同时，林肯实验室也进行了计算机技术的研究，并建立了两台使用新的晶体管技术的机器 TX-0 和 TX-2。之所以编号都是偶数，是因为奇数（odd）在英文中同时有古怪的意思，主管设计者之一 Wesley Clark 曾经说："林肯不做奇数的（古怪的）计算机"。TX-0 和 TX-2 的关系类似于 MTC 和旋风的关系：TX-0 用于测试非常大的（按当时标准）内存，然后这些内存再用于功能更强大的 TX-2。这些新机器继承了旋风系统中使用 CRT 显示屏和发光笔这些与用户实时交互的能力，同时也保留了使用纸带作为程序的主要介质。

在开发 TX-0 的同时，在 MIT 安装了一台 IBM 704 机器。它用来补充并最终接替了旋风作为 MIT 一般用户的主计算机。当林肯实验室不再需要 TX-0 后，MIT 电子工程系长期租用了它。MIT 的师生（特别是电子研究实验室的师生），都为拥有了一台计算机而大喜过望，因为从此研究人员便可以自由使用并亲手操作这台计算机，这要比 IBM 704 计算机采用的批处理方式方便得多。

我于 1958 年 8 月完成了我的博士论文，成为一个四处寻找机遇的学校教员。我的新办公室在康普顿实验室楼（26 号楼）的二楼。有一天那里发生的事情引起了我的注意，人们正在一块宽广的区域安装一台 TX-0，它的位置就在 IBM 704 计算机的正上方。

与 TX-0 一起到来的软件工具只有两个，一个是简单的汇编器程序，另一个是"UT-3"（3 号工具纸带）。两个程序都是二进制打孔纸带的形式，没有源代码。因为它们是以八进制代码手工输入的。UT-3 通过一个控制台打字机与用户交互（这里仍然是一个电传打字机，它包含了普通打字机的功能，可以被用户或被 TX-0 所驱动，将输入的字符传递到计算机或打印在纸上；这台打字机还带有一个机械纸带打孔器和阅读器，可以将字符打在纸带上或从纸带把字符读入计算机中）。用户可以以八进制形式把数据输入到指定的内存位置，也可以要求打印指定内存位置或区域的内容。在 MIT，我们马上着手给这两个程序增加功能。汇编器最后演化为一个叫作 MACRO 的程序，除了有其他熟悉的汇编语言功能外，它还支持宏指令（宏功能是从 Doug McIlroy 在贝尔实验室的研究工作中得到启发的）。

有了汇编器后，就使得大范围重写和扩展 UT-3 成为可能。Tom Stockham 和我使新的程序支持符号，新的程序可以使用汇编器生成的符号表。我们把这个程序称作 FLIT（电传打字机询问纸带），这个名字仿用了当时一个很常用的杀虫喷雾剂的名字（当 Grace Hopper 在哈佛的继电器计算机上工作时，跟踪到一次故障是由于继电器触点上的一只飞蛾造成的，从此人们开始把计算机的问题称作 bug，即"臭虫"）。FLIT 最重要的功能是为调试程序（"除虫"）提供了断点设施。用户可以要求 FLIT 在被测试程序中向指定的指令位置插入最多 4 个断点。当被测试的程序遇到一个断点时，FLIT 会通知用户，并且允许用户分析或修改内存的内容。分析结束后，用户可以要求 FLIT 继续执行程序，就像没有中断过一样。FLIT 程序是后来的 DDT（另一种杀虫剂）调试程序的典范，DDT 是 MIT 的学生为 DEC 公司生产的 PDP-1 计算机开发的。

FLIT（以及 TX-0）的缺点之一是，没有办法防止被测试程序向调试程序占用的内存里存储数据，这会使调试程序停止工作。在给 DEC PDP-1 建立分时系统时，我们做了特别的设计，使得 DDT 与待测试的程序在各自的地址空间中执行，但 DDT 仍可以观察和改变被测试程序中的信息。我们把它称为“隐身调试器”。为了提供这种保护，需要对 PDP-1 增加一些逻辑，它们是随着为支持分时系统而做的更改和补充一起安装的。这个系统在 1963 年前后开始运行。

PDP-1 上的分时系统为伯克利加州大学在 SDS 940 上建立的分时系统提供了典范（L. Perter Deutsch 兜里装着的那个小操作系统从 MIT 转移到了伯克利加州大学）。我隐约地相信，隐身调试器的机制对于 DEC PDP-11/45 的设计产生了重要影响，贝尔实验室就为这个系统开发了 UNIX。

Jack B. Dennis

2008 年 4 月于马萨诸塞州贝尔蒙特

第 2 版前言

在 900 多年前的一个秋夜，一轮明月高高地挂在黄州的天空。夜深了，很多人都已经入睡。但在承天寺的庭院里，还有两个人在散步。他们一边交谈，一边欣赏美丽的夜景。洁白的月光泼洒在庭院里，像是往庭院里注入了一汪汪清水，把地面变成了水面，清澈透明。翠竹和松柏的影子映在其中，随风摇摆，仿佛水草在晃动。这两个人中，一位是大文豪苏轼，另一位是他的好朋友张怀民。这一年是公元 1083 年，苏轼 46 岁。

可能是在当晚，也可能是在第二日，苏轼写了一篇短文来记录这次夜游。这篇短文便是著名的《记承天寺夜游》。第一次看到这篇散文，我便爱不释手。每次读苏轼文集，都喜欢把这一篇再读一遍。文章很短，不足百字，但意境隽永，令人回味无穷。

“元丰六年十月十二日夜，解衣欲睡，月色入户，欣然起行。念无与为乐者，遂至承天寺寻张怀民。怀民亦未寝，相与步于中庭。庭下如积水空明，水中藻荇交横，盖竹柏影也。”

文末的议论尤其脍炙人口：“何夜无月？何处无竹柏？但少闲人如吾两人者耳。”

诚然，月夜常有，竹子和松树也很平常，但是这样的夜游不常有。

2013 年深秋，与十几位喜欢调试技术的朋友在庐山五老峰下的白鹿洞书院聚会，吃过晚饭大家坐在古老的书院里交流调试技术，直到夜里 10 点左右。然后，大家又聚集在延宾馆的庭院里，一边海阔天空地聊天，一边欣赏美丽的夜景。说话的间隙可以听见院子外面贯道溪的哗哗水声；抬起头，便看到满天的星斗。

2008 年 6 月 3 日，作者收到了出版社快递给我的《软件调试》第 1 版，喜不自禁，写了一篇博客，名为“手捧汗水的感觉”。

弹指一挥间，十年过去了。十年中，因为《软件调试》作者认识了很多朋友。他们有不同的年龄，不同的背景，工作在不同的地方，但都有一个共同点——读过《软件调试》。

2011 年 9 月，《软件调试》第 1 版出版 3 年后，作者便开始计划和写作第 2 版。但只坚持了一年便停顿了。之后写写停停，进展很缓慢。直到 2016 年年底，从工作了十几年的英特尔公司辞职后，作者才又“重操旧业”。

过去的十年中，计算机领域发生了很多重大的变革。顺应这些变革，新的版本需要增加很

多内容。简单来说，第 2 版卷 1 新增了以下内容。

- 关于 CPU 增加了 ARM 处理器的相关内容。
- 关于操作系统增加了 Linux 系统的相关内容。
- 关于编译器增加了 GCC 的相关内容。
- 关于调试器增加了 GDB 的相关内容。
- 增加了全新的 GPU 内容。

新增这些内容后，如果再装订成一本书，那么肯定比砖头还厚。经过反复思考和调整，最后终于确定了分卷出版的方案。卷 1 覆盖处理器等基础内容，卷 2、卷 3 分别介绍 Windows 系统和 Linux 系统的调试。

确定了新的分卷结构后，作者强迫自己投入更多的时间写作，快步向前推进。终于在 2018 年 6 月把卷 1 的书稿发给了出版社。

卷 1 共 16 章，分为 4 篇。

第一篇：绪论（第 1 章）

作为全书的开篇，这一篇介绍了软件调试的概念、基本过程、分类和简要历史，并综述了本书后面将详细介绍的主要调试技术。

第二篇：CPU 及其调试设施（第 2～7 章）

CPU 是计算机系统的硬件核心。这一篇以英特尔和 ARM 架构的 CPU 为例，系统描述了 CPU 的调试支持，包括如何支持软件断点、硬件断点和单步调试（参见第 4 章），如何支持硬件调试器（参见第 7 章），记录分支、中断、异常和支持性能分析的方法（参见第 5 章），以及支持硬件可调试性的错误检查和报告机制——MCA（机器检查架构）（参见第 6 章）。为了帮助读者理解这些内容，以及本书后面的内容，第 2 章介绍了关于 CPU 的一些基础知识，包括指令集、寄存器和保护模式，第 3 章深入介绍了与软件调试关系密切的中断和异常机制。与第 1 版相比，第 2 版不仅扩展了原来关于 x86 处理器的内容，还新增了 ARM 处理器的内容。

第三篇：GPU 及其调试设施（第 8～14 章）

这是第 1 版没有的全新内容，分 7 章深入探讨了 Nvidia、AMD、英特尔、ARM 和 Imagination 这五大厂商的 GPU。从某种程度上说，CPU 的时代已经过去，GPU 的时代正在开启。经历了半个多世纪的发展，CPU 已经很成熟，CPU 领域的创新机会越来越少。CPU 仍会存在，但不会再热门。而 GPU 领域则像是一块新大陆，有很多地方还是荒野，等待开垦，仿佛 19 世纪的美国西部，或者 20 世纪末的上海浦东。

第四篇：可调试性（第 15～16 章）

提高软件调试效率是一项系统的工程，除了 CPU、操作系统和编译器所提供的调试支持外，被调试软件本身的可调试性也是至关重要的。这一篇首先介绍了提高软件可调试性的意义、基

本原则、实例和需要注意的问题（参见第 15 章），然后讨论了如何在软件开发实践中实现可调试性（参见第 16 章）。第 16 章的内容包括软件团队中各个角色应该承担的职责，实现可追溯性、可观察性和自动报告的方法。

在内容格式上，第 2 版也有所变化。首先，新增了名为“格物致知”的实践模块。读者可以下载试验材料，然后按照书中的指导步骤进行操作。在理论和实践方面，朱熹曾说：“言理则无可捉摸，物有时而离。言物则理自在，自是离不得。”这句话的意思是，空讲理论可能让人摸不着头脑，把理论和实践分离开来；相反，讲具体的事物，自然就包含了道理，二者是分不开的。好一个“言物则理自在”，真是至理名言。其实，“言物”除了有朱熹说的“言物则理自在”好处外，还有生动有趣的优点。为此，第 2 版不仅新增了专门言物的“格物致知”模块，很多章节的正文内容也是本着这个思想来写作的。

另外，第 2 版还增加了评点模块——“老雷评点”和“格友评点”。“老雷评点”是“格蠹老雷”所评，“格友评点”为“格友”评点。“格蠹老雷”是作者的绰号。“格友”者，“格蠹老雷”之友也。“格”字源于上文所说之格物。在古老的《易经》中，8 个基本符号中有一个为震，象征雷，代表着锐意创新和开拓进取。

感谢苏轼，他用优美的文字清晰记录了 900 多年前的那个夜晚表现了作者心向往之的那种意境，让我们可以穿越时空，领略一代文豪的生活和心灵世界。感谢更多曾经著书立说的前辈，他们用文字向我们传递了他们的思想和智慧。

感谢缔造软件的前辈们，他们创造了一种新的形式来传递智慧。感谢父母，把我生在这个美好的软件时代。乐哉，三生有幸做软件。

因为书，自古便有读书之乐，穿越时空，悟前人心境，获前人智慧。因为软件，今天有调试之乐，电波传语，与硅片对谈，赏匠心之美，品设计之妙。希望本书可以让读者同时体验读书之乐和调试之乐。当然，如果读者能以此结缘，结交一两个可以在月朗星稀之夜“相与步于中庭”的朋友就更好了。

张银奎（Raymond Zhang）

2018 年 7 月 25 日于上海格蠹轩

第1版前言

现代计算机是从20世纪40年代开始出现的。当时的计算机比今天的要庞大很多，很多部件也不一样，但是有一点是完全相同的，那就是靠执行指令而工作。

一台计算机认识的所有指令被称为它的指令集（instruction set）。按照一定格式编写的指令序列被称为程序（program）。在同一台计算机中，执行不同的程序，便可以完成不同的任务，因此，现代计算机在诞生之初常被冠以通用字样，以突出其通用性。在带来好处的同时，通用性也意味着当人们需要让计算机完成某一件事情时，首先要编写一个能够完成这件事的程序，然后才执行这个程序来真正做这件事。使用这种方法的过程中，人们很快就意识到了两个严峻的问题：一是编写程序需要很多时间；二是当把编写好的程序输入计算机中执行时，有时它会表现出某些出乎意料的怪异行为。因此，首先不得不寻找怪异行为的根源，然后改写程序，如此循环，直到目标基本实现为止，或者因没有时间和资源继续做这件事而不得不放弃。

程序对计算机的重要性和编写程序的复杂性让一些人看到了商机。大约在20世纪50年代中期，专门编写程序的公司出现了。几年后，模仿硬件（hardware）一词，人们开始使用软件（software）这个词来称呼计算机程序和它的文档，并把将用户需求转化为软件产品的整个过程称为软件开发（software development），将大规模生产软件产品的社会活动称为软件工程（software engineering）。

如今，几十年过去了，我们看到的是一个繁荣而庞大的软件产业。但是前面描述的两个问题依然存在：一是编写程序仍然需要很多时间；二是编写出的程序在运行时仍然会出现意料外的行为。同时，后一个问题的表现形式越来越多，在运行过程中，程序可能会突然报告一个错误，可能会给出一个看似正确却并非需要的结果，可能会自作聪明地自动执行一大堆无法取消的操作，可能会忽略用户的命令，可能会长时间没有反应，可能会直接崩溃或者永远僵死在那里……而且总是可能有无法预料的其他情况出现。这些“可能”大多是因为隐藏在软件中的设计失误而导致的，即所谓的软件臭虫（bug），或者软件缺欠（defect）。

计算机是在软件指令的指挥下工作的，让存在缺欠的软件指挥强大的计算机硬件工作是件危险的事，可能导致惊人的损失和灾难性的事件发生。2003年8月14日，北美大停电（Northeast Blackout of 2003）使50万人受到影响，直接经济损失60亿美元，其主要原因是软件缺欠导致报警系统没有报警。1999年9月23日，美国的火星气象探测船因为没有进入预定轨道从而导致受到大气压力和摩擦而被摧毁，其原因是不同模块使用的计算单位不同，使计算出的轨道数据出现

严重错误。1990 年 1 月 15 日，AT&T 公司的 100 多台交换机崩溃并反复重新启动，导致 6 万用户在 9h 中无法使用长途电话，其原因是新使用的软件在接收到某一种消息后会导致系统崩溃，并把这种症状传染给与它相邻的系统。1962 年 7 月 22 日，“水手一号”太空船发射 293s 后因为偏离轨道而被销毁，其原因也与软件错误有直接关系。类似的故事还有很多，尽管我们不希望它们发生。

一方面，软件缺欠难以避免；另一方面，软件缺欠的危害很大。这使得消除软件缺欠成为软件工程中的一项重要任务。消除软件缺欠的前提是要找到导致缺欠的根本原因。我们把探索软件缺欠的根源并寻求其解决方案的过程称为软件调试（software debugging）。

软件调试是在复杂的计算机系统中寻找软件缺欠的根源。这是让软件从业者头疼的一项任务。要在软件调试中游刃有余，需要对软件和计算机系统有深刻的理解，选用科学的方法，并使用强有力的工具。

本书的写作目的

在复杂的计算机系统中寻找软件缺欠的根源不是一个简单的任务，需要对软件和计算机系统有深刻的理解，选用科学的方法，并使用强有力的工具。这些正是作者写作本书的初衷。具体来说，写作本书的 3 个主要目的如下。

- **论述软件调试的一般原理，包括 CPU、操作系统和编译器是如何支持软件调试的，内核态调试和用户态调试的工作模型，以及调试器的工作原理。**软件调试是计算机系统中多个部件之间的一个复杂交互过程。要理解这个过程，必须要了解每个部件在其中的角色和职责，以及它们的协作方式。学习这些原理不仅对提高软件工程师的调试技能至关重要，还有利于加深他们对计算机系统的理解，将计算机原理、编译原理、操作系统等多个学科的知识融会贯通。

- **探讨可调试性（debuggability）的内涵、意义和实现软件可调试性的原则与方法。**所谓软件的可调试性就是在软件内部加入支持调试的代码，使其具有自动记录、报告和诊断的能力，从而更容易调试。软件自身的可调试性对于提高调试效率、增强软件的可维护性，以及保证软件的如期交付都有着重要意义。软件的可调试性是软件工程中一个很新的领域，本书对其进行了深入系统的探讨。

- **交流软件调试的方法和技巧。**尽管论述一般原理是本书的重点，但是本书同时穿插了许多实践性很强的内容。其中包括调试用户态程序和系统内核模块的基本方法，如何诊断系统崩溃（BSOD）和应用程序崩溃，如何调试缓冲区溢出等与栈有关的问题，如何调试内存泄漏等与堆有关的问题。特别是，本书非常全面地介绍了 WinDBG 调试器的使用方法，给出了大量使用这个调试器的实例。

第 2 版说明	上一段所描述内容将在后续分卷中单独介绍。

总之，作者希望通过本书让读者懂得软件调试的原理，意识到软件可调试性的重要性，学

会基本的软件调试方法和调试工具的使用，并能应用这些方法和工具解决问题和学习其他软硬件知识。历史证明，所有软件技术高手都是软件调试高手，或者说不精通软件调试技术不可能成为（也不能算是）软件技术高手。本书希望带领读者走上这条高手之路。

本书的读者对象

第一，本书是写给所有程序员的。程序员是软件开发的核心力量。他们花大量的时间来调试他们所编写的代码，有时为此工作到深夜。作者希望程序员读完本书后能自觉地在代码中加入支持调试的代码，使调试能力和调试效率大大提高，不再因为调试程序而加班。本书中关于CPU、中断、异常和操作系统的介绍，是很多程序员需要补充的知识，因为对硬件和系统底层的深刻理解不但有利于写出好的应用程序，而且对于程序员的职业发展也是有利的。之所以说写给“所有”程序员是因为本书主要讨论的是一般原理和方法，没有限定某种编程语言和某个编程环境，也没有局限于某个特定的编程领域。

第二，本书是写给从事测试、验证、系统集成、客户支持、产品销售等工作的软件工程师或 IT 工程师的。他们的职责不是编写代码，因此软件缺欠与他们不直接相关，但是他们也经常因为软件缺欠而万分焦急。他们不需要负责修改代码并纠正问题，但是他们需要知道找谁来解决这个问题。因此，他们需要把错误定位到某个模块，或者至少定位到某个软件。本书介绍的工具和方法对于实现这个目标是非常有益的。另外，他们也可以从关于软件可调试性的内容中得到启发。本书关于 CPU、操作系统和编译器的内容对于提高他们的综合能力并巩固软硬件知识也是有益的。

第三，本书是写给从事反病毒、网络安全、版权保护等工作的技术人员的。他们经常面对各种怪异的代码，需要在没有代码和文档的情况下做跟踪和分析。这是计算机领域中非常具有挑战性的工作。关于调试方法和 WinDBG 的内容有利于提高他们的效率。很多恶意软件故意加入了阻止调试和跟踪的机制，本书的原理性内容有助于理解这些机制。

第四，本书是写给计算机、软件、自动控制、电子学等专业的研究生或高年级本科生的。他们已经学习了程序设计、操作系统、计算机原理等课程，阅读本书可以帮助他们把这些知识联系起来，并深入到一个新的层次。学会使用调试器来跟踪和分析软件，可以让他们在指令一级领悟计算机软硬件的工作方式，深入核心，掌握本质，把学到的书本知识与计算机系统的实际情况结合起来。同时，可以提高他们的自学能力，使他们养成乐于钻研和探索的良好习惯。软件调试是从事计算机软硬件开发等工作的一项基本功，在学校里就掌握了这门技术，对于以后快速适应工作岗位是大有好处的。

第五，本书是写给勇于挑战软件问题的硬件工程师和计算机用户的。他们是软件缺欠的受害者。除了要忍受软件缺欠带来的不便之外，有时软件生产方还将责任推卸给他们，找借口说是硬件问题或使用不当造成的 bug。使用本书的工具和方法，他们可以找到很充足的证据来为自己说话。另外，本书的大多数内容不需要很深厚的软件背景，有基本的计算机知识就可以读懂。

或许不属于上面 5 种类型的读者也可以阅读本书。比如，软件公司或软件团队的管理者、软件方面的咨询师和培训师、大学和研究机构的研究人员、非计算机专业的学生、自由职业者、编程爱好者、黑客，等等。

要读懂和领会本书的内容，读者应具备以下基础。

- 曾经亲自参与编写程序，包括输入代码、编译，然后执行。
- 使用过某一种类型的调试器，用过断点、跟踪、观察变量等基本调试功能。如果对这些功能充满了好奇并且希望了解它们是如何工作的，则更好。
- 参加过某个软件开发项目，对软件工程有基本的了解，承认软件的复杂性，认为开发一个软件产品与写一个 HelloWorld 程序根本不是一回事。

尽管本书给出了一些汇编代码和 C/C++代码，但是其目的只是在代码层次直截了当地阐述问题。本书的目标不是讨论编程语言和编程技巧，也不要求读者已经具备丰富的编程经验。

本书的主要内容

第 2 版说明	根据读者意见，第 2 版将分多卷组织。第 1 版中的第一篇、第二篇和第五篇包含在卷 1 中，第 1 版中的第三篇、第四篇和第六篇将包含在后续分卷中。新的结构以卷 1 为公共基础，其他分卷结合各自的平台环境深入介绍。因为结构变化较大，所以此处关于第 1 版内容结构的介绍删除。

本书的三条线索

第 2 版说明	针对第 2 版的变化，关于本书线索和本书阅读方法的内容略有调整。

本书的内容是按照以下三条线索来组织的。

第一条线索是软件调试的“生态”系统（ecosystem）。

第二条线索是异常（exception）。异常是计算机系统中的一个重要概念，出现在 CPU、操作系统、编程语言、编译器、调试器等多个领域，本书逐一对其做了解析。

第三条线索是调试器。调试器是解决软件问题非常有力的工具，它是逐步发展到今天这个样子的。第 1 章介绍了单纯依赖硬件的调试方法。第 4 章分析了 DOS 下调试器的实现方法。第 7 章介绍硬件仿真和基于 JTAG 标准的硬件调试器。后续分卷将基于不同的操作系统平台详细介绍调试器的工作原理和用法。另外，全书很多地方都使用了调试器输出的结果，穿插了使用调试器解决软件问题的方法。

本书的阅读方法

本书的厚度决定了不适合一口气将它看完。以下是作者给出的阅读建议。

下载并安装 WinDBG 调试器。如果你还不了解它的基本用法，那么请先参考 WinDBG 的帮助文件，学会它的基本用法，能读懂栈回溯结果。有了这个工具后，你就可以尝试本书所描述的相关调试方法，自己在系统中探索书中提到的内容。

建议选择前面提到的三条线索中的一条来阅读。如果你有充裕的时间，那么可以按第一条线索来阅读。如果你想深入了解异常，那么可以按第二条线索来阅读。如果你有难题等待解决，希望快速了解基本的调试方法，那么你可以选择第三条线索，选择阅读与调试器有关的内容。

先阅读每一篇开始处的简介，了解各篇的概况，浏览主要章节，建立一个初步的印象。当需要时，再仔细查阅感兴趣的细节。

以上建议中，第一条是希望读者遵循的，其他建议谨供参考。

本书的写作方式

这是一本关于软件调试的书，同时它的大多数内容也是依靠软件调试技术来探索得到的。在作者使用的计算机系统中，一个名为 Toolbox 的文件夹下保存了 100 多个不同功能的工具软件。当然，使用最多的还是调试器。书中给出的大多数栈回溯结果是使用 WinDBG 调试器产生的。

写作本书的一个基本原则是首先从有代表性的实例出发，然后从这个实例推广到其他情况和一般规律。例如，在 CPU 方面作者选择的是 IA-32 CPU；在操作系统方面选择 NT 系列的 Windows 操作系统；在编译器方面选择的是 Visual Studio 系列；在调试器方面选择的是 Visual Studio 内置的调试器和 WinDBG。

本书的示例、工具和代码可以从高端调试网站（advdbg.org）免费下载，单击该网站左下方的“特别链接”中的《软件调试》即可。

第 2 版说明	第 2 版的电子资源网站为 advdbg.org 网站。

尽管作者和编辑已经尽了最大努力，但是本书中仍然可能存在这样那样的疏漏，欢迎读者通过上面的网站反馈给我们。

关于封面

人们遇到百思不得其解或者难以解释清楚的问题时可能不由自主地说：“见鬼了。”在软件开发和调试中也时常有这样的情况。钟馗是传说中的“捉鬼”能手，因此我们选取他作为本书

的封面人物，希望这本书能够帮助读者轻松化解“见鬼了”这样的复杂问题。

第 2 版说明	第 1 版书出版后某月，在上海长风公园偶然购得一幅皮影材质的钟馗画像，拿回家后把它装在玻璃镜框中，尺寸大约为 50cm × 38cm。在英特尔公司工作的几年里，这幅画一直挂在我的办公桌前。

免责声明

本书的内容完全是作者本人的观点，不代表任何公司和单位。你可以自由地使用本书的示例代码和附属工具，但是作者不对因使用本书内容和附带资料而导致的任何直接和间接后果承担任何责任。

致谢

第 2 版说明	倏忽之间，十年过去了，重读这个致谢名单，一个个熟悉的面孔浮现在眼前。时光流转，真情不变，对各位朋友的感激之情永存我心。

首先感谢 Jack B. Dennis 教授，他向我讲述了大型机时代的编程环境和调试方法以及他和 Thomas G. Stockman 为 TX-0 计算机编写 FLIT 调试器的经过，并专门为本书撰写了短文“历史回眸”。FLIT 调试器是作者追溯到的最早的调试器程序。

感谢 Windows 领域的著名专家 David Solomon 先生，他回答了我的很多提问，并为本书第 1 版写了推荐序。

第 2 版说明	因为 David Solomon 先生所写推荐序主要与 Windows 系统有关，所以把该推荐序放到了本书后续分卷中。

感谢《Showstopper》一书的作者 Greg Pascal Zachary 先生，他允许我引用他书中的内容和该书的照片。

感谢 CPU 和计算机硬件方面的权威 Tom Shanley 先生，他在计算机领域的著作有十几本，他关于 IA-32 架构方面的培训享誉全球。感谢他允许我在本书中使用他绘制的关于 CPU 寄存器的大幅插图（因篇幅所限最终没有使用）。

探索 Windows 调试子系统让我感受到了软件之美，创造这种美感的一个主要技术专家便是 Mark Lucovsky 先生，感谢他在邮件中给予我的鼓励。

感谢 DOS 之父 Tim Paterson 先生，他向我介绍了他编写 8086 Monitor 的经过，并允许我使用这个调试器程序的源代码。

感谢 Syser 调试器的作者吴岩峰先生，我们多次讨论了如何在单机上实现内核调试的技术

细节，他始终关心着本书的进度。

感谢我的老板和同事，他们是：Kenny、Michael、Feng、Adam、Jim、Neal、Harold、Cui Yi、Keping、Eric、Yu、Wei、Min、Fred、Rick、Shirley、Vivian、Luke、Caleb、Christina 和 Starry（请原谅，我无法列出所有名字）。

感谢我的好朋友刘伟力，我们一起加班解决了一个大 bug 后，他感慨地说“断点真神奇”，这句话让我产生了写作本书的念头。

感谢曾华军（我们共同翻译了《机器学习》）与李妍帮助我翻译了 Jack B. Dennis 为本书写的“历史回眸”和 David Solomon 为本书第 1 版写的推荐序。

感谢以下朋友阅读了本书的草稿，提出了很多宝贵的意见：王毅鹏、王宇、施佳、夏桅、周祥、李晓宁、侯伟和吴巍。

第 2 版说明	以下朋友帮助检查了第 2 版卷 1 的初稿，在此表示感谢！ 感谢谭添升、彭广杰、郁丛祥、邹冠群、卜道成、金睿、Yajun Yang、张耀欣和黎小红。

感谢本书第 1 版的编辑周筠和陈元玉，感谢两位编辑对我的一贯支持，以及编辑本书所花费的大量时间。感谢本书第 1 版美术编辑胡文佳，她的精心设计让本书的封面如此美丽。

第 2 版说明	特别感谢人民邮电出版社为本书安排强大的编辑团队：陈冀康、谢晓芳、吴晋瑜，他们出色的工作让我非常感动。也要感谢我的好朋友张文杰，他为本书第 2 版精心绘制了很多幅插图。

感谢我的家人，在写作本书的漫长而且看似没有尽头的日子里，她们承担了繁重的家务，让我有时间完成本书。

最后，感谢你阅读本书，并希望你能从中受益！

张银奎（Raymond Zhang）
2008 年 4 月于上海
2018 年 7 月 26 日更新于上海 863 软件园