

程序调试 思想与实践

The Science of Debugging

[美] Matt Telles, Yuan Hsieh 著

邓劲生 等译



中国水利水电出版社
www.waterpub.com.cn



程序调试思想与实践

[美] Matt Telles, Yuan Hsieh 著

邓劲生 等译

中国水利水电出版社

内 容 提 要

本书将调试作为一门专业的学科进行研究和分析,提供大量的代码实例和问题描述,对调试的各个方面进行细致而深入的阐述和讨论。全书以 bug 为中心,围绕调试这一主题进行组织。第2章到第5章包括 bug 的诊断、分类以及它们的症状。第6章到第10章讨论那些处理 bug 的策略,包括可以使用的工具以及在不同情况下如何最有效地使用它们。第11章到第16章包括对专业调试的解释以及如何成为一个出色的调试员。

本书主要面向的读者群是软件项目的开发人员、调试人员、测试人员以及管理人员。

Original English language edition published by The Coriolis Group LLC, 14455 N. Hayden Drive, Suite 220, Scottsdale, Arizona 85260 USA, telephone (480) 483-0192, fax (480) 483-0193. Copyright © 2001 by The Coriolis Group. Simplified Chinese Language edition copyright © 2002 by China WaterPower Press. All rights reserved.

北京市版权局著作权合同登记号:图字 01-2001-2599

图书在版编目(CIP)数据

程序调试思想与实践/(美)特列斯(Telles,M.), (美)元赫(Yuan Hsieh)著;邓劲生等译. —北京:中国水利水电出版社, 2002

书名原文: The Science of Debugging

ISBN 7-5084-1008-4

I. 程… II. ①特… ②元… ③邓… III. 程序设计—研究 IV. TP311

中国版本图书馆 CIP 数据核字(2002)第 014414 号

书 名	程序调试思想与实践
作 者	[美]Matt Telles, Yuan Hsieh 著
译 者	邓劲生 等译
出版、发行	中国水利水电出版社(北京市三里河路6号 100044) 网址: www.waterpub.com.cn E-mail: mchannel@public3.bta.net.cn (万水) sale@waterpub.com.cn
经 售	电话: (010) 68359286 (万水)、63202266 (总机)、68331835 (发行部) 全国各地新华书店
排 版	北京万水电子信息有限公司
印 刷	北京市天竺颖华印刷厂
规 格	787×1092 毫米 16 开本 22.75 印张 491 千字
版 次	2002 年 3 月第一版 2002 年 3 月北京第一次印刷
印 数	0001—5000 册
定 价	40.00 元

凡购买我社图书,如有缺页、倒页、脱页的,本社发行部负责调换
版权所有·侵权必究

译者序

自从有了计算机以来，甚至更早的时候，bug 和 debug 的斗争就一直纠缠在每个编程人员的心头。但是过去 30 年以来程序员们并没有真正做出什么改变。那些在软件工业发展的初期就存在的问题，现在仍然存在。软件的 bug 和在早期一样，仍然在当今的软件中出现，但是数量已经大不一样了。过去 bug 只出现在独立的单个用户或者一组用户之中，现在它们却出现在整个世界。那些过去仅需要手工重新制作报告并重新输入数据的问题，而现在可能意味着一个组织的生存。

尽管调试在软件的各个领域都极其重要，但它仍然被看作软件工程师的例行工作，这种状况越来越明显地需要改变。调试必须和设计 and 需求分析一样进行考虑：就是说，作为它本身的工作。用特殊的态度来考虑如何编写系统的设计，并不起多大的作用；但是如果用特殊的态度来考虑如何维护和修正系统的代码，则不然。

在目前的各种软件开发书籍中，大量的篇幅被用于讨论关于设计软件、编写代码和配置系统的更好方式等内容，通常到最后才考虑调试。但是它是软件开发生命过程中的一个自然的部分，是不容忽视地存在于我们每个软件项目中的，并且关系到整个项目的成功与否。调试和测试介入项目的时间越早，最后成品软件的质量就越高。

本书的作者都在软件领域工作多年，有着丰富的工程实践经验。本书将调试作为一门专业的学科进行研究和分析，提供大量的代码实例和问题描述，对调试的各个方面进行细致而深入的阐述和讨论，系统而有条理地解决了许多我们长期思考但仍然迷惑不解的问题。阅读本书，就像一名长者正在娓娓道来，真有一种如沐春风的感觉。

全书以 bug 为中心，内容围绕调试这一主题进行组织，大致可以划分为三个部分。第 2 章到第 5 章包括 bug 诊断、分类以及它们的症状。第 6 章到第 10 章讨论那些处理 bug 的策略，包括可以使用的工具以及在不同情况下如何最有效地使用它们。第 11 章到第 16 章包括对专业调试的解释以及如何成为一个出色的调试员。每章的最后都有一个 bug 问题留待读者思考，并将解答放在附录中。

本书主要面向的读者群是软件项目的开发人员、调试人员、测试人员以及管理人员。本书由邓劲生、刘锋、陈永然、陶阳等完成主要章节的翻译工作，参加录入、校对和排版的其他人员还有王江书、刘培英、何易非、任芳、金姜等，全文由邓劲生统校。作为译者，我们逐字逐句地细细研读本书，感觉从中受益匪浅。但是由于我们水平有限，书中难免出现一些不当甚至错误之处，恳请读者批评指正。

译者

2001 年 10 月

作者简介

Matt Telles 是软件开发界一名有 15 年经验的老手。他经历过 FORTRAN、COBOL 和其他奇怪的语言。现在是 Microsoft 公司的高级工程师。他的时间主要花在查找和修复别人产生的 bug 上。除此之外，他还喜欢将他掌握的技术教给其他开发人员。拥有了对应用程序进行编程、设计、做文档和调试的专门技术之后，他已经到达了当今程序员的巅峰：能够为专著写自己的简历。Matt 还出版过其他 5 本书，他居住在 Colorado 州的 Lakewood，正在忙着玩他心爱的 DEC 10。

Yuan Hsieh 还记得那些能够在互联网上阅读每个新闻组的日子。他花了 12 年来制造 bug。同样也花了 12 年修复这些 bug。他的修复工作非常好。目前他是 InfoNow 公司的软件开发主管。他不再制造和修复 bug 了，而是看别人制造和修复 bug。现在他把大部分时间花在如何将开发人员集中而作为一个统一而高效的编程团体，以能够及时在预算内产生无 bug 的软件。他现在住在 Colorado 州的 Lone Tree，将大部分闲暇时间用来在山上游玩。

致 谢

我要感谢我的雇主——Microsoft 公司给我时间和空间来完成本书。而且我要感谢我的妻子 Dawnna 和三个孩子能容忍来自我的办公室的激昂的讨论。最后，任何书的完成都离不开幕后人员的工作。Tom Lamoureux、Kevin Weeks、Peggy Cantrell、Jodi Winkler、Quentin Correll、Bob Arnson 和 Anne Marie Walker 不知疲倦地将我的空泛的文字转换成像样的英文。

——Matt

有人说一个艺术家通常在思维被打乱的情况下才发挥得最好。虽然这是一本技术类的书，也需要一定的思维和灵感。像我这样一个学者，没有比经历约定的典礼更能打乱思维了。这样，我就要感谢所有的过去和现在的出现在我生活中的异性，谢谢那些焦虑、灵感和不眠的夜晚（这时是写作的最好时候、独自在黑暗中发泄、涌现有哲理的词句）。

严肃地讲，我必须要感谢一些人。感谢 Kim Wright 帮我校对一些早期的章节。感谢 Stephen Shiller 爽快地为本书提供了令人惊讶的例子。感谢在 Coriolis 的人们：Kevin Weeks、Tom Lamoureux、Peggy Cantrell、Jodi Winkler 和技术参考人员 Quentin Correll，以及副本编辑者 Anne Marie Walker，谢谢他们的帮助。而且有许多与我一同工作的人员以及我所工作过的公司和地方。这些经历是无价的。

最后，感谢 Matt 为我提供了与他一起编写此书的机会。

——Y'

目 录

译者序

作者简介

致谢

第 1 章 Debug 简介.....	1
1.1 本书的内容.....	2
1.2 为什么要关注 bug.....	3
1.3 什么是 bug.....	4
1.4 本书的对象.....	4
1.5 本书的组织.....	5
1.6 调试简史.....	6
1.7 小结.....	8
1.8 bug 问题.....	8
第 2 章 研究著名的（以及不太著名的）bug 例子.....	10
2.1 大致情况.....	11
2.1.1 现实生活中的分布式系统.....	11
2.1.2 Therac-25.....	15
2.2 bug 实例 #1.....	19
2.3 bug 实例 #2.....	22
2.4 bug 实例 #3.....	24
2.5 bug 实例 #4.....	28
2.5.1 AT&T 电话中断.....	29
2.5.2 缓冲区溢出.....	32
2.6 小结.....	35
2.7 bug 问题.....	38
第 3 章 什么是 bug.....	39
3.1 什么是 bug.....	39
3.1.1 什么是软件缺陷.....	41
3.1.2 调试不是什么.....	42
3.1.3 什么是调试.....	43
3.2 为什么要考虑 bug.....	44

3.2.1	bug 的信心代价	44
3.2.2	名声和形象的代价	45
3.2.3	bug 的财产损失	46
3.3	bug 的本质	47
3.3.1	bug 的发生都有原因	47
3.3.2	bug 是可以重现的	47
3.3.3	bug 通常在有变化时出现	48
3.3.4	bug 也会产生 bug	49
3.3.5	bug 吸引 bug	49
3.3.6	证明缺乏理解	49
3.3.7	难的代码对大家都一样难	51
3.3.8	在软件生命周期中不同阶段的 bug 有不同的性质	51
3.3.9	稳定的系统中的 bug 的原因可能比错误的系统中的 bug 的原因 更难以识别	53
3.4	小结	54
3.5	bug 问题	54
第 4 章	bug 的生命周期	55
4.1	为什么会出现 bug	55
4.1.1	复杂性	55
4.1.2	现实	61
4.1.3	人类的弱点	63
4.2	bug 是如何产生的	64
4.2.1	对软件做修改	64
4.2.2	拙劣的描述	67
4.2.3	方法的复杂性	68
4.2.4	缺少一致的观点	70
4.2.5	程序员错误	71
4.3	bug 是如何躲过测试的	74
4.3.1	遵循形式过程代价太大	75
4.3.2	政策/市场决策	75
4.3.3	时间不充分	75
4.3.4	缺少重现能力	75
4.3.5	自负	76
4.3.6	差劲的描述/不知道要测试什么	76
4.3.7	缺乏测试环境	77

4.4	小结	77
4.5	bug 问题	77
第 5 章	bug 的分类	79
5.1	bug 的种类	80
5.1.1	需求阶段的 bug	80
5.1.2	设计阶段的 bug	80
5.1.3	实现阶段的 bug	81
5.1.4	处理阶段的 bug	83
5.1.5	编译的 bug	85
5.1.6	配置的 bug	86
5.1.7	未来计划 bug	86
5.1.8	文档 bug	87
5.2	严重性	88
5.3	bug 分类法	88
5.3.1	名字	89
5.3.2	描述	89
5.3.3	最一般的环境	89
5.3.4	症状	89
5.3.5	例子	89
5.4	bug 的分类	89
5.4.1	内存或资源泄漏	89
5.4.2	逻辑错误	91
5.4.3	编码错误	92
5.4.4	内存侵占	94
5.4.5	循环错误	96
5.4.6	条件错误	97
5.4.7	指针错误	99
5.4.8	分配/释放错误	100
5.4.9	多线程错误	102
5.4.10	定时错误	103
5.4.11	分布式应用程序错误	105
5.4.12	存储错误	107
5.4.13	集成错误	108
5.4.14	转换错误	109
5.4.15	硬编码长度/尺寸	110

5.4.16	版本 bug	112
5.4.17	不恰当地重用 bug	113
5.4.18	布尔 bug	114
5.5	为什么分类重要	116
5.6	小结	116
5.7	bug 问题	117
第 6 章	检测工作	118
6.1	整体调试	118
6.1.1	复制和粘贴错误	119
6.1.2	全局变量	120
6.1.3	副作用	121
6.1.4	观察意外的消息或结果	123
6.1.5	跟踪诊断显示	124
6.2	调试方法	124
6.2.1	科学方法	124
6.2.2	直觉	125
6.2.3	思维跳跃	125
6.2.4	诊断	126
6.3	商业技巧	126
6.3.1	内嵌式调试器	127
6.3.2	日志对象	127
6.3.3	跟踪对象	128
6.3.4	隐藏的诊断屏	129
6.3.5	为以后的程序运行保存 bug 数据	129
6.4	可重现实例	130
6.4.1	测试实例	130
6.4.2	数据依赖	130
6.4.3	从根本原因中分离症状	131
6.4.4	收集观察结果	131
6.4.5	统计/公理	132
6.5	小结	133
6.6	bug 问题	133
第 7 章	调试工具及其使用时机	135
7.1	测试和调试环境	135
7.1.1	测试组	135

7.1.2	测试套	136
7.1.3	过时的 bug 组	136
7.1.4	日志	137
7.1.5	跟踪	138
7.2	中级调试技术	138
7.2.1	内存漏洞检测工具	138
7.2.2	交叉索引及工具用法	140
7.2.3	调试器	141
7.2.4	Heisenberg 的不确定原理	141
7.2.5	嵌入式诊断	142
7.2.6	断言的弊端	143
7.2.7	同用户一起工作	143
7.2.8	bug 跟踪	144
7.2.9	代码覆盖范围分析	144
7.2.10	编译器	145
7.3	小结	147
7.4	bug 问题	147
第 8 章	调试的一般过程	148
8.1	识别问题	148
8.1.1	这是 bug 吗	148
8.1.2	为什么它是一个 bug	149
8.1.3	程序应该在做什么	150
8.1.4	程序到底在做什么	151
8.2	收集信息	152
8.2.1	用户对问题的描述	152
8.2.2	日志文件	154
8.2.3	新自观察	156
8.2.4	症状	156
8.2.5	失败的测试实例	157
8.2.6	相似的问题	157
8.2.7	近期变化	158
8.2.8	运行的环境信息	159
8.3	形成假设	161
8.4	测试假设	166
8.5	崩溃的 Web 服务器示例	166

8.6	重复直到某个假设被证实	168
8.7	提出解决方法	169
8.8	测试解决方法	170
8.9	重复直到某个解决方法被证实	170
8.10	回归测试	171
8.11	小结	171
8.12	bug 问题	172
第 9 章	调试技术	173
9.1	插入式调试和非插入式调试	173
9.2	短期调试和长期调试技术	174
9.3	成品软件调试的折衷办法	175
9.4	调试技术介绍	176
9.4.1	面对实际用户	176
9.4.2	查看观察日志	176
9.4.3	记录调试代码和过程	177
9.4.4	模拟代码和问题	177
9.4.5	简化重现性	178
9.4.6	把问题简化成最简单的元素	178
9.4.7	代码消去法	179
9.4.8	简化法	180
9.4.9	使用调试器	181
9.4.10	跳跃	182
9.4.11	分解法	183
9.4.12	种植错误法	184
9.4.13	编译检测法	185
9.4.14	整体考虑法	185
9.4.15	使用不同操作系统上的另一个编译器	186
9.4.16	每次改变一个变量	187
9.4.17	数字命理学和边界条件	188
9.4.18	检查最近的修改	189
9.4.19	清除系统中的“死代码”	190
9.4.20	问题假设法	192
9.4.21	检查未测试的代码	194
9.4.22	不变式法	195
9.4.23	存储器使用情况	196

9.4.24	互斥	197
9.4.25	显示系统的运行情况	198
9.4.26	和工作的系统进行代码比较	199
9.4.27	理解算法	199
9.4.28	检查连通性	200
9.4.29	核心文件	201
9.4.30	增加跟踪	202
9.4.31	数据关系检查	203
9.4.32	重放能力（记录动作）	204
9.4.33	生成系统的副本	205
9.5	小结	205
9.6	bug 问题	206
第 10 章	不同应用系统的调试	207
10.1	小规模单机系统	207
10.1.1	成为系统的用户	208
10.1.2	复制环境	208
10.1.3	提防 DLL Hell	209
10.1.4	输入/输出错误	209
10.2	中规模单机系统	210
10.3	中规模客户/服务器系统	211
10.3.1	生成后端数据库的一个简单描述	212
10.3.2	保留数据的使用情况来发现哪些数据经常被使用	212
10.4	大规模系统	213
10.4.1	为测试安装一个“后门”	213
10.4.2	观察外部条件的改变	214
10.5	实时系统	214
10.5.1	注意添加调试语句	214
10.5.2	监视硬件软件的冲突	215
10.5.3	时间问题	215
10.6	嵌入式系统	216
10.6.1	模拟器问题	216
10.6.2	抑制中断	216
10.6.3	协议错误	216
10.6.4	看门狗时钟	217
10.6.5	调试嵌入式系统	217

10.7 分布式系统	217
10.7.1 中间设备错误	218
10.7.2 预测错误	218
10.7.3 连接错误	218
10.7.4 安全错误	219
10.7.5 信息数据库	219
10.7.6 记录的事后调查分析	220
10.8 模拟系统	220
10.8.1 封装硬件接口	220
10.8.2 把模拟调用封装成预测错误返回	220
10.8.3 把实际系统模拟器的问题简化成最简单的形式	221
10.9 小结	221
10.10 bug 问题	221
第 11 章 调试之后	223
11.1 是不是在其他地方犯同样的错误	224
11.2 bug 背后隐含了什么	225
11.3 怎样预防同类 bug	226
11.3.1 理解原因	226
11.3.2 保留测试用例	226
11.3.3 利用 bug 为将来的设计	226
11.4 怎样更容易发现类似的 bug	227
11.4.1 创造工具	227
11.4.2 记录 bug	228
11.4.3 预留调试框架	229
11.5 我是否有所进步	231
11.5.1 bug 量度	232
11.5.2 bug 跟踪	232
11.5.3 怎样处理数据	233
11.6 小结	234
11.7 bug 问题	234
第 12 章 bug 预防	236
12.1 什么是 bug 预防	236
12.2 一般技术	238
12.2.1 预防 bug 的人	238
12.2.2 问题原因分析	241

12.2.3	检测错误	243
12.2.4	重用	247
12.2.5	减小复杂度和管理复杂度	253
12.2.6	现实世界的文档	257
12.2.7	内部基础结构支持	260
12.3	需求中的 bug 预防	262
12.3.1	理解问题	263
12.3.2	正确获得需求	265
12.4	设计中的 bug 预防	266
12.4.1	不同的设计产生不同的 bug 类型	266
12.4.2	设计接口	269
12.4.3	设计包含文档的代码	274
12.5	实现中的 bug 预防	275
12.5.1	理解开发工具	275
12.5.2	防御编码	276
12.5.3	实现选择	280
12.5.4	逐步测试	282
12.6	小结	283
12.7	bug 问题	284
第 13 章	测试	285
13.1	单元测试	285
13.1.1	测试什么	285
13.1.2	单元测试为什么重要	285
13.1.3	怎样应用到调试中	286
13.1.4	单元测试有什么重要问题	286
13.2	验证测试	286
13.3	质量保证测试	287
13.4	测试方法	288
13.4.1	路径测试	288
13.4.2	事务处理测试	290
13.4.3	输入验证	290
13.4.4	算法测试	293
13.4.5	决策表	294
13.4.6	状态机分析	294
13.4.7	综合测试	296

13.4.8	自上向下测试和自下向上测试	297
13.4.9	配置调试	298
13.4.10	恢复崩溃和掉电测试	299
13.4.11	安全性测试	300
13.4.12	第三方测试	301
13.4.13	多用户测试	302
13.4.14	负载和性能测试	303
13.4.15	测量和统计	304
13.5	小结	304
13.6	bug 问题	305
第 14 章	维护	307
14.1	什么是软件维护	307
14.1.1	维护工作	308
14.1.2	维护的挑战	308
14.1.3	软件的退化需要软件维护	309
14.2	创建一个可维护的软件系统	309
14.2.1	创建可理解的软件	309
14.2.2	可维护设计	313
14.2.3	提供一个回归测试环境	316
14.2.4	创建一个可维护的环境	316
14.3	维护现有的软件	317
14.3.1	修改现有软件系统	317
14.3.2	围绕现有的需求和设计 bug 展开工作	318
14.3.3	彻底的回归测试	319
14.3.4	保持修改跟踪	319
14.4	什么时候要放弃	320
14.5	小结	321
14.6	bug 问题	321
第 15 章	专业调试	323
15.1	学习成为一名调试员	323
15.1.1	在软件维护和改进组中工作	323
15.1.2	学习编写源代码的一个好办法	325
15.2	什么地方需要专业调试员	325
15.2.1	在设计阶段	326
15.2.2	在需求阶段	326

15.2.3	在代码审查阶段	327
15.2.4	在代码移交阶段	328
15.2.5	项目阶段总结	329
15.3	优秀专业调试者的特点	329
15.3.1	老练	329
15.3.2	耐性	330
15.3.3	探测技巧	330
15.3.4	处理压力的能力	331
15.3.5	工程的/科学的方法	332
15.3.6	忘我	332
15.3.7	坚定不移	333
15.4	专业调试者的一天	334
15.5	小结	336
15.6	bug 问题	336
附录 A	bug 问题的答案	337
附录 B	附加的阅读资料	344