第一图书网, tushu007.com

<<软件调试实战>>

图书基本信息

书名：<<软件调试实战>>

13位ISBN编号：9787115218858

10位ISBN编号：7115218854

出版时间：2010-2

出版时间：人民邮电出版社

作者：Thorsten Grotker,Ulrich Holtmann,Holger Keding,Markus Wloka

页数：190

译者：赵俐

版权说明：本站所提供下载的PDF图书仅提供预览和简介，请支持正版图书。

更多资源请访问：http://www.tushu007.com

第一图书网, tushu007.com

前言

在所有软件开发工作中，调试或许是最令人苦恼的。
调试工作极易受到指责，因为技术失败即意味着做人失败，而且矛头直指调试人员，昭示出他们所犯下的错误。
由于必须反复思考每个假设，反复斟酌从需求到实现的每个步骤，因此调试将耗费大量时间。
最糟糕的是，调试还无法预测，我们永远无法知道修复一个bug需要多长时间，甚至根本不知道是否能够修复它。
问一下开发人员他们生活中最沮丧的时刻，大多数回答将与调试有关。
也许现在正是深夜11点，你仍在忙着调试，正当对程序进行走查时，你的家人打电话给你，问你到底什么时候才能回家，而你只希望尽快放下电话，因为好不容易得到的观察结果和推断正要从脑中溜走。
此时，你可能最后有两种选择：一是重新调试，二是过后再试图重修旧好。
据我个人估计，调试是导致程序员离婚的第一大原因。
然而，调试也蕴含着乐趣，就像解出难题、猜出谜语或破获谋杀案一样令人激动，但前提条件是必须采用系统性的方式并配备正确的工具。
这正是本书的用武之地。
本书四位作者直接与那些固持己见的开发者对话，直截了当地提出解决调试问题的建议，并给出了真正快速的解决方案。
无论是解决内存问题，调试并行程序，还是处理工具链引入的问题，本书都能够提供“急救措施”，书中的建议都是经过反复尝试和验证的。
如果我最初开始调试程序时就能有这样一本书，该多好啊！
我想我会屏息注视，看看这些调试工具将带给我什么惊喜，而且采纳书中的建议，必然会节省大量手工调试的时间，并可将这些时间投入到其他工作中。
譬如说，可以使代码更可靠，这样最后可能根本不必做任何调试了。
当然，这是专业编程的长期目标，即从一开始就编写正确的代码，通过某种确认或验证方法来杜绝所有错误（或至少检测到错误）。
目前，断言和单元测试已经在提高程序可信度方面提供了很多帮助。
未来可能会有一些用于行业级系统的成熟验证方法。
我们现在尚未实现这样的方法，这可能需要很多年，而且当达到这个目标时，所实现的方法肯定不会适用于现在的编程语言。
在处理当今的程序，特别是那些C和C++程序时，我们仍将在调试上花费一定时间——这正是本书的宝贵价值所在。

第一图书网, tushu007.com

<<软件调试实战>>

内容概要

《软件调试实战》主要讲述C/C++程序的调试和分析，书中的调试技术也可以应用于其他语言编写的程序。

《软件调试实战》在讲述简单的源代码分析和测试的基础上，讲述了现实的程序中经常遇到的一些问题（如程序链接、内存访问、并行处理和性能分析）并给出了解决方案。

《软件调试实战》适合软件开发人员、调试人员阅读和使用。

第一图书网, tushu007.com

作者简介

作者：(德国)Thorsten Grotker (德国)Ulrich Holtmann (德国)Holger Keding 等 译者：赵俐Thorsten Grotker，Ulrich Holtmann，Holger Keding，Markus Wloka4位作者均拥有德国或美国著名高等学府的博士学位，目前都任职于EDA（电子设计自动化）软件领导厂商Synopsys（新思）公司，分别担任研发主管、资深软件工程师等职位，负责开发编译器和调试工具，具有解决各种调试问题的丰富经验。

第一图书网, tushu007.com

<<软件调试实战>>

书籍目录

第1章 谁编写软件，谁制造bug（为什么需要本书）1第2章 系统性调试方法32.1 为什么要遵循结构化的过程32.2 充分利用机会32.3 13条黄金规则52.3.1 理解需求52.3.2 制造失败62.3.3 简化测试用例62.3.4 读取恰当的错误消息62.3.5 检查显而易见的问题62.3.6 从解释中分离出事实72.3.7 分而治之72.3.8 工具要与bug匹配82.3.9 一次只做一项更改92.3.10 保持审计跟踪92.3.11 获得全新观点92.3.12 bug不会自己修复92.3.13 用回归测试来检查bug修复102.4 构建一个好的工具包102.4.1 工具箱112.4.2 每天运行测试，防止出现bug112.5 认清敌人——遇到bug家族132.5.1 常见bug132.5.2 偶发性bug132.5.3 Heisenbug132.5.4 隐藏在bug背后的bug142.5.5 秘密bug——调试与机密性142.5.6 更多读物15第3章 查找根源——源代码调试器173.1 可视化程序行为173.2 准备简单的可预测的示例183.3 使调试器与程序一起运行183.4 学习在程序崩溃时执行栈跟踪213.5 学习使用断点213.6 学习在程序中导航223.7 学习检查数据：变量和表达式223.8 一个简单示例的调试会话23第4章 修复内存问题274.1 C/C++中的内存管理——功能强大但很危险274.1.1 内存泄漏274.1.2 内存管理的错误使用284.1.3 缓冲区溢出284.1.4 未初始化的内存bug284.2 有效的内存调试器284.3 示例1：检测内存访问错误294.3.1 检测无效的写访问304.3.2 检测对未初始化的内存的读取操作304.3.3 检测内存泄漏314.4 示例2：对内存分配/释放的不完整调用314.5 结合使用内存调试器和源代码测试器334.6 减少干扰，排查错误334.7 何时使用内存调试器344.8 约束344.8.1 测试用例应该有很好的代码覆盖率344.8.2 提供更多计算机资源354.8.3 可能不支持多线程354.8.4 对非标准内存处理程序的支持35第5章 剖析内存的使用375.1 基本策略——主要步骤375.2 示例：分配数组385.3 第1步：查找泄漏385.4 第2步：设置期望值385.5 第3步：测量内存使用395.5.1 使用多个输入395.5.2 在固定时间间隔停止程序395.5.3 用简单工具测量内存使用405.5.4 使用top405.5.5 使用WindowsTaskManager415.5.6 为testmalloc选择相关输入值425.5.7 确定机器上的内存是如何被释放的425.5.8 使用内存剖析工具435.6 第4步：查明大部分内存被哪些数据结构占用了445.7 综合练习——genindex示例455.7.1 核实没有大的内存泄漏465.7.2 估计内存使用465.7.3 测量内存使用465.7.4 查找使用内存的数据结构47第6章 解决性能问题516.1 分步查找性能bug516.1.1 执行前期分析516.1.2 使用简单的时间测量方法526.1.3 创建测试用例526.1.4 使测试用例具有可再现性536.1.5 检查程序的正确性536.1.6 创建可扩展的测试用例536.1.7 排除对测试用例的干扰546.1.8 用time命令测量时可能会发生错误和偏差546.1.9 选择一个能够揭示运行时间瓶颈的测试用例556.1.10 算法与实现之间的差异566.2 使用剖析工具586.2.1 不要编写自己的剖析工具586.2.2 剖析工具的工作原理586.2.3 了解gprof596.2.4 了解Quantify636.2.5 了解Callgrind646.2.6 了解VTune666.3 分析I/O性能68第7章 调试并行程序717.1 用C/C++编写并行程序717.2 调试竞争条件727.2.1 使用基本调试器功能来查找竞争条件737.2.2 使用日志文件来查找竞争条件747.3 调试死锁767.3.1 如何确定正在运行的是哪个线程777.3.2 分析程序的线程787.4 了解线程分析工具787.5 异步事件和中断处理程序80第8章 查找环境和编译器问题838.1 环境变更——问题的根源838.1.1 环境变量838.1.2 本地安装依赖848.1.3 当前工作目录依赖848.1.4 进程ID依赖848.2 如何查看程序正在做什么848.2.1 用top来查看进程848.2.2 用ps来查找应用程序的多个进程858.2.3 使用/proc/来访问进程858.2.4 使用strace跟踪对操作系统的调用858.3 编译器和调试器也有bug878.3.1 编译器bug878.3.2 调试器和编译器兼容性问题88第9章 处理链接问题899.1 链接器的工作原理899.2 构建并链接对象899.3 解析未定义的符号919.3.1 丢失链接器参数919.3.2 搜索丢失的符号919.3.3 链接顺序问题929.3.4 C++符号和名称改编939.3.5 符号的反改编949.3.6 链接C和C++代码949.4 具有多个定义的符号959.5 信号冲突969.6 识别编译器和链接器版本不匹配969.6.1 系统库不匹配979.6.2 对象文件不匹配979.6.3 运行时崩溃989.6.4 确定编译器版本989.7 解决动态链接问题1009.7.1 链接或载入DLL1009.7.2 无法找到DLL文件1019.7.3 分析载入器问题1029.7.4 在DLL中设置断点1039.7.5 提供DLL问题的错误消息104第10章 高级调试10710.1 在C++函数、方法和操作符中设置断点10710.2 在模板化的函数和C++类中设置断点10910.3 进入C++方法11010.3.1 用step-into命令进入到隐式函数中11210.3.2 用step-out命令跳过隐式函数11210.3.3 利用临时断点跳过隐式函数11310.3.4 从隐式函数调用返回11310.4 条件断点和断点命令11410.5 调试静态构造/析构函数11610.5.1 由静态初始化程序的顺序依赖性引起的bug11710.5.2 识别静态初始化程序的栈跟踪11810.5.3 在静态初始化之前连接调试器11810.6 使用观察点11910.7 捕捉信号12010.8 捕获异常12210.9 读取栈跟踪12410.9.1 带调试信息编译的源代码的栈跟踪12410.9.2 不带调试信息编译的源代码的栈跟踪12410.9.3 不带任何调试信息的

第一图书网, tushu007.com

帧12510.9.4 实际工作中的栈跟踪12510.9.5 改编后的函数名称12610.9.6 被破坏的栈跟踪12610.9.7 核心转储12710.10 操纵正在运行的程序12810.10.1 修改变量13010.10.2 调用函数13110.10.3 修改函数的返回值13210.10.4 中止函数调用13210.10.5 跳过或重复执行个别语句13310.10.6 输出和修改内存内容13310.11 在没有调试信息时进行调试13510.11.1 从栈读取函数参数13710.11.2 读取局部/全局变量和用户定义的数据类型13810.11.3 在源代码中查找语句的大概位置13910.11.4 走查汇编代码140第11章 编写可调试的代码14311.1 注释的重要性14311.1.1 函数签名的注释14411.1.2 对折中办法的注释14411.1.3 为不确定的代码加注释14411.2 采用一致的编码风格14411.2.1 仔细选择名称14511.2.2 不要使用“ 聪明过头 ”的结构14511.2.3 不要压缩代码14511.2.4 为复杂表达式使用临时变量14511.3 避免使用预处理器宏14611.3.1 使用常量或枚举来替代宏14611.3.2 使用函数来替代预处理器宏14811.3.3 调试预处理器输出14911.3.4 使用功能更强的预处理器15011.4 提供更多调试函数15111.4.1 显示用户定义的数据类型15111.4.2 自检查代码15211.4.3 为操作符创建一个函数，以便帮助调试15311.5 为事后调试做准备153第12章 静态检查的作用15512.1 使用编译器作为调试工具15512.1.1 不要认为警告是无害的15612.1.2 使用多个编译器来检查代码15812.2 使用lint15812.3 使用静态分析工具15812.3.1 了解静态检查器15812.3.2 将静态检查器检测到的错误减至（接近）零16012.3.3 完成代码清理后重新运行所有测试用例16012.4 静态分析的高级应用161第13章 结束语163附录A 调试命令165附录B 工具资源167附录C 源代码179参考文献189

第一图书网, tushu007.com

章节摘录

插图：有了源代码调试器（以下简称调试器）以后，就可以逐行地走查源代码，查看程序的条件语句和循环语句都经过了哪些路径，显示哪些函数被调用了，以及显示目前正处于函数调用栈的什么位置。
我们可以检查变量值，并在个别的代码行中设置断点，然后让程序运行，直至到达此行。
这是在复杂程序中进行导航的便利方法。
调试器将显示程序都执行了哪些操作，这是修复任何bug的先决条件。
如果代码是自己编写的，调试器可以帮助实现预期行为，即代码的功能。
如果是别人编写的，调试将呈现出代码执行的动态视图，以补充静态代码检查。
本章将介绍基本的源代码调试器功能，并讲述如何用它们来查找C和C++程序中的bug。
同时介绍独立于特定的计算机平台或工具。
示例中将使用两个非常常见的调试器：GDB和VisualStudio。
同时列出GDB和VisualStudio访问每个被讨论的特性的命令。
为了节省篇幅，这里只展示节选的GDB输出并简要介绍Visual Studio如何输出结果，不会给出屏幕截图。
GNU调试器GDB代表从具有命令行接口的命令行解释器（command shell）运行的调试器。
GDB与GCC编译器一起使用，已经被植入很多操作系统中，如Windows、Solaris、15NIX、Linux和用于嵌入式系统的操作系统。
有关GDB的下载信息和文档，参见附录B.2.3。

第一图书网, tushu007.com

媒体关注与评论

“如果我最初开始调试程序时就能有这样一本书，该多好啊！
我想我会屏息注视，看看这些调试工具将带给我什么惊喜，而且采纳书中的建议必然会节省大量手工调试的时间，可以将这些时间投入到其他工作中。
譬如说，我可以使代码更可靠，这样最后可能根本不必做任何调试了。
”　　——Andreas Zeller，GNU DDD创始人“逐页阅读完本书之后，我必须承认这是我读过的最好的一本软件调试图书，在很多方面都是其他书不能匹敌的……强烈推荐软件调试人员阅读。
”　　——www.dumpanalysis.org（崩溃转储分析和调试门户网站）

第一图书网, tushu007.com

<<软件调试实战>>

编辑推荐

《软件调试实战》是4位深谙软件调试之道的资深开发人员的实战经验总结，不仅讲述了简单的源代码调试，还涵盖了各个领域的最常见的实际问题，包括程序链接、内存存取、并行处理和性能分析。最后几章讨论了静态检查器，介绍了一些较好地运用了调试技巧的代码编写方法。
书中讲述的调试技术不仅可以用于C／C++程序，还可以用于其他语言编写的程序。
软件调试是软件开发中最令人苦恼的环节。
反复思考每个假设，反复斟酌从需求到实现的每个步骤，将耗费大量时间。
最糟糕的是，调试根本无法预测，我们永远无法知道修复一个bug需要多长时间，甚至根本不知道是否还能修复它。
然而，如果采用系统的方式并配备合适的工具，调试也会充满乐趣，成功的调试就像解出难题、猜出谜语或破获奇案一样令人激动。
这本书就能帮你实现这一惊天逆转。
Amazon五星图书Synopsys公司专家的调试经验总结软件调试权威指南

第一图书网, tushu007.com

版权说明

本站所提供下载的PDF图书仅提供预览和简介，请支持正版图书。

更多资源请访问:http://www.tushu007.com