

<<软件调试修炼之道>>

图书基本信息

书名：<<软件调试修炼之道>>

13位ISBN编号：9787115252647

10位ISBN编号：7115252645

出版时间：2011-6

出版单位：人民邮电出版社

作者：Paul Butcher

页数：158

译者：曹玉琳

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>

## <<软件调试修炼之道>>

### 内容概要

调试对软件开发至关重要。  
然而，即使对于有经验的程序员，调试也决非易事。

本书是一部优秀的软件调试实战指南，作者总结了自己和身边同事多年的经验教训，详细阐述了调试的方方面面。

书中内容共分为三大部分。

第一部分借助软件特有的功能展示缺陷是怎么产生的，介绍了建立在实证方法之上的核心调试方法；  
第二部分阐述怎样发现代码中存在需要修复的问题，以及如何将调试融入到整个软件开发过程中去；  
第三部分讨论如何避免一些常见的缺陷。

本书秉承了Pragmatic图书简洁实用的风格，总结了大量方法与经验，适合软件开发人员、调试人员阅读并迅速付诸实践。

## <<软件调试修炼之道>>

### 作者简介

Paul Butcher

资深程序员，涉猎广泛，从单片机编码到高级声明式编程无所不精。

Paul是一位少年天才，8岁时就已经开始在8位机上编写游戏。

最近几年他开始痴迷于赛车，认为自己是可以和汉密尔顿比肩的赛车手。

## <<软件调试修炼之道>>

### 书籍目录

#### 第一部分 问题的核心

##### 第1章 山重水复疑无路

- 1.1 调试不仅是排除缺陷
- 1.2 实证方法
- 1.3 核心调试过程
- 1.4 先澄清几个问题
  - 1.4.1 你知道要找的是啥吗
  - 1.4.2 一次一个问题
  - 1.4.3 先检查简单的事情
- 1.5 付诸行动

##### 第2章 重现问题

- 2.1 重现第一，提问第二
  - 2.1.1 明确开始要做的事
  - 2.1.2 抓住重点
- 2.2 控制软件
- 2.3 控制环境
- 2.4 控制输入
  - 2.4.1 推测可能的输入
  - 2.4.2 记录输入值
  - 2.4.3 负载和压力
- 2.5 改进问题重现
  - 2.5.1 最小化反馈周期
  - 2.5.2 将不确定的缺陷变为确定的
  - 2.5.3 自动化
  - 2.5.4 迭代
- 2.6 如果真的不能重现问题该怎么办
  - 2.6.1 缺陷真的存在吗
  - 2.6.2 在相同的区域解决不同的问题
  - 2.6.3 让其他人参与其中
  - 2.6.4 充分利用用户群体
  - 2.6.5 推测法
- 2.7 付诸行动

##### 第3章 诊断

- 3.1 不要急于动手——试试科学的方法
- 3.2 相关策略
  - 3.2.1 插桩
  - 3.2.2 分而治之
  - 3.2.3 利用源代码控制工具
  - 3.2.4 聚焦差异
  - 3.2.5 向他人学习
  - 3.2.6 奥卡姆的剃刀
- 3.3 调试器
- 3.4 陷阱
  - 3.4.1 你做的修改是正确的吗
  - 3.4.2 验证假设

## <<软件调试修炼之道>>

3.4.3 多重原因

3.4.4 流沙

3.5 思维游戏

3.5.1 旁观调试法

3.5.2 角色扮演

3.5.3 换换脑筋

3.5.4 做些改变，什么改变都行

3.5.5 福尔摩斯原则

3.5.6 坚持

3.6 验证诊断

3.7 付诸行动

### 第4章 修复缺陷

4.1 清除障碍

4.2 测试

4.3 修复问题产生的原因，而非修复现象

4.4 重构

4.5 签入

4.6 审查代码

4.7 付诸行动

### 第5章 反思

5.1 这到底是怎么搞的

5.2 哪里出了问题

5.2.1 我们已经做到了吗

5.2.2 根本原因分析

5.3 它不会再发生了

5.3.1 自动验证

5.3.2 重构

5.3.3 过程

5.4 关闭循环

5.5 付诸行动

## 第二部分 从大局看调试

### 第6章 发现代码存在问题

6.1 追踪缺陷

6.1.1 缺陷追踪系统

6.1.2 怎样才能写出一份出色的缺陷报告

6.1.3 环境和配置报告

6.2 与用户合作

6.2.1 简化流程

6.2.2 有效的沟通

6.3 与支持人员协同工作

6.4 付诸行动

### 第7章 务实的零容忍策略

7.1 缺陷优先

7.1.1 早期缺陷修复可以大大降低软件运行的不确定性

7.1.2 没有破窗户

7.2 调试的思维模式

7.3 自己来解决质量问题

## <<软件调试修炼之道>>

- 7.3.1 这里没有“灵丹妙药”
- 7.3.2 停止开发那些有缺陷的程序
- 7.3.3 从“不干净”的代码中将“干净”的代码分离出来
- 7.3.4 错误分类
- 7.3.5 缺陷闪电战
- 7.3.6 专项小组
- 7.4 付诸行动

### 第三部分 深入调试技术

#### 第8章 特殊案例

- 8.1 修补已经发布的软件
- 8.2 向后兼容
  - 8.2.1 确定你的代码有问题
  - 8.2.2 解决兼容性问题
- 8.3 并发
  - 8.3.1 简单与控制
  - 8.3.2 修复并发缺陷
- 8.4 海森堡缺陷
- 8.5 性能缺陷
  - 8.5.1 寻找瓶颈
  - 8.5.2 准确的性能分析
- 8.6 嵌入式软件
  - 8.6.1 嵌入式调试工具
  - 8.6.2 提取信息的痛苦路程
- 8.7 第三方软件的缺陷
  - 8.7.1 不要太快去指责
  - 8.7.2 处理第三方代码的缺陷
  - 8.7.3 开源代码
- 8.8 付诸行动

#### 第9章 理想的调试环境

- 9.1 自动化测试
  - 9.1.1 有效的自动化测试
  - 9.1.2 自动化测试可以作为调试的辅助
  - 9.1.3 模拟测试、桩测试以及其他的代替测试技术
- 9.2 源程序控制
  - 9.2.1 稳定性
  - 9.2.2 可维护性
  - 9.2.3 与分支相关的问题
  - 9.2.4 控制分支
- 9.3 自动构建
  - 9.3.1 一键构建
  - 9.3.2 构建机器
  - 9.3.3 持续集成
  - 9.3.4 创建版本
  - 9.3.5 静态分析
  - 9.3.6 使用静态分析
- 9.4 付诸行动

#### 第10章 让软件学会自己寻找缺陷

## <<软件调试修炼之道>>

### 10.1 假设和断言

#### 10.1.1 一个例子

#### 10.1.2 等一下——刚才发生了什么

#### 10.1.3 例子，第二幕

#### 10.1.4 契约，先决条件，后置条件和不变量

#### 10.1.5 开启或关闭断言

#### 10.1.6 防错性程序设计

#### 10.1.7 断言滥用

### 10.2 调试版本

#### 10.2.1 编译器选项

#### 10.2.2 调试子系统

#### 10.2.3 内置控制

### 10.3 资源泄漏和异常处理

#### 10.3.1 在测试中自动抛出异常

#### 10.3.2 一个例子

#### 10.3.3 测试框架

### 10.4 付诸行动

## 第11章 反模式

### 11.1 夸大优先级

### 11.2 超级巨星

### 11.3 维护团队

### 11.4 救火模式

### 11.5 重写

### 11.6 没有代码所有权

### 11.7 魔法

### 11.8 付诸行动

## 附录A 资源

## 附录B 参考书目

## &lt;&lt;软件调试修炼之道&gt;&gt;

## 章节摘录

第1章 山重水复疑无路 如果你的软件不能正常工作，该怎么做呢？

一些开发人员似乎有窍门能够准确无误地找出发生缺陷的根本原因，而另一些开发人员似乎总在漫无目的地寻找原因却得不到确切的结果。

是什么使他们之间存在着这样的差异呢？

在这一章中，我们将仔细探究一种调试方法，这种方法在专业的软件开发中已经被反复证实。它绝非灵丹妙药，它仍然依赖于调试者的智慧、直觉、探查缺陷的技巧，甚至一点儿运气。但是，它会使你的努力更加有效，避免做无用功，并且能够尽快地找到问题的核心。

具体来说，我们将介绍以下内容： 调试与排除缺陷的区别； 实证方法——借助软件本身来告诉你现在的运行状态； 核心调试过程（问题重现，问题诊断，缺陷修复，反思）； 做最先应该做的事——在深入调试之前应该先考虑的事情。

1.1 调试不仅是排除缺陷 试问一个没有经验的程序员什么是调试，他可能回答调试就是“找到一种修复缺陷的方法”。

事实上，这仅仅是调试诸多目标中的一个，甚至不是最重要的目标。

有效的调试需要采取以下步骤。

- （1）弄清楚软件为什么会运行失常。
- （2）修复这一问题。
- （3）避免破坏其他部分。
- （4）保持或者提高代码的总体质量（可读性、架构、测试覆盖率、性能等）。
- （5）确保同样的问题不会在其他地方发生，也不会再次发生。

其中，目前最重要的是首先查明问题的根本原因，这是一切事情的基础。

理解万岁 一些没有经验的开发人员（遗憾的是，有时就是我们中本应知道得更多的那些人）经常完全忽略问题诊断这一过程，而是往往立即采取他们认为能够修复程序的措施。

如果他们走运，只是修改了的程序运行不了，他们所做的一切是在浪费时间而已。

真正的危险是修改了的程序可以运行，或者看来可以运行，因为这时开发人员在一知半解的情况下改变了源程序。

他或许修复了缺陷，但是实际上很可能掩盖了潜在的根本原因。

更糟糕的是，这种改变可能会导致新问题——破坏一些曾经可以正确运行的程序。

⋮



<<软件调试修炼之道>>

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>