由鸿铭
E-mail：hithongming@163.com

第七章第四节 调试

防不胜防的bug引入了代码，如何发现并消除之？

Outline

- 什么是bug
- 调试的基本过程
- 调试的技术与工具

Notes

什么是bug

- bug即程序中的错误，导致程序以非预期或未预料到的方式执行。
- 一个包含大量bug和/或严重干扰其功能的bug的程序被称为buggy。
- 报告程序中的bug通常被称为bug报告、故障报告、问题报告、故障报告、缺陷报告等

【bug产生的原因】

- 代码错误
- 未完成的要求或不够详细
- 误解用户需求
- 设计文档中的逻辑错误
- 缺乏文件
- 没有足够的测试

【bug的常见类型】

- 数学bug：例如 零除法，算术溢出
- 逻辑bug：例如 无线循环和无限递归
- 源头bug：例如 使用了为被定义的变量、资源泄漏，其中有限的系统资源如内存或文件句柄通过重复分配耗尽而不释放。缓冲区溢出，其中程序试图将数据存储在分配存储的末尾。
- 团队工程bug：例如 评论过时或者评论错误、文件与实际产品的区别

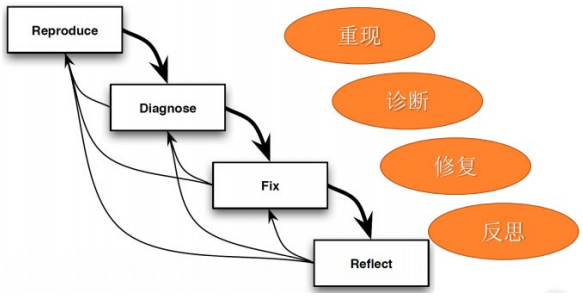
调试的基本过程

- Debug是测试的后续步骤：测试发现问题，debug消除问题；当防御式编程和测试都无法挡住bug时，我们就必须进行debug了；
- Debug的目的：寻求错误的根源并消除它；（Debug占用了大量的时间）

【调试的过程】

常用方法：假设-检验

- 重现（Reproduce）-->诊断（Diagnose/Locating）-->修复（Fix）-->反思（Reflect）
- 重现（Reproduce）：寻找一个可靠、方便得在线需求问题的方法。
  - 从最小的测试用例开始复现错误（保持复现bug的前提下降低输入规模）
  - 消除因版本、环境、配置等不同引起的差异（通过构建软件实现），确定bug出现的环境（通过程序模拟硬件平台的细节，实现不同的操作系统环境）
  - 利用逆向设计推断导致错误的输入
  - 若无法重现，则无法观察以证明分析和修补的正确性
- 诊断（Diagnose/Locating）：构建假设，并通过执行实验来测试它们，直到您确信已识别错误的根本原因。
  - 从假设开始，构造实验，证明它是对的或者错的
  - 从不符合理论的观察结果开始，修正理论
  - 查看导致错误的测试输入，以及错误的结果，失败的断言以及由此导致的堆栈跟踪
  - 提出一个与所有数据一致的假设，说明错误发生的位置或错误发生的位置，设计实验测试假设
  - 收集实验数据，减少错误可能出现的范围，做出新的假设
  - 设计不同的实验：检查内部状态、修改运行方式、改变本身逻辑
  - 每次只做一个修改、做好记录、不忽略细节、运行不同的测试用例、设置断点、用可实现相同功能并且被证实无问题的组件替代当前组件
- 修复（Fix）：设计和实施解决问题的变化，避免引入回归，并保持或提高软件的整体质量。
  - 确保从干净的源代码树开始
  - 运行现有的测试，并证明它们通过
  - 添加一个或多个新测试，或修复现有测试，以演示错误
  - 修复错误、发现可改进之处
  - 证明你的修复工作正常且没有引入回归（以前通过的测试现在失败）
  - 如果引入回归，通过回顾以前的版本来找出确切的变化
- 反思（Reflect）：思考需求、设计、测试、结构（库、编译器等）



调试的技术和工具

【调试技术】

- 暴力调试（Brute Force Attack）
  - 蛮力方法可以分为至少三类：
    - 看内存导出文件
    - 根据“在整个程序中分散打印语句”的常见建议进行调试。
    - 自动化调试工具
- 递归调试（Induction）
- 演绎调试（Decution）
- 回溯调试（Backtracking）

测试调试（Testing）

【调试工具】

语法和逻辑检查（本课程未涵盖）
源代码比较器(Source-code comparator)
（参见第2章中的SCM）
内存堆转储（Memory heap dump）
打印调试/日志记录(Print debugging / logging)
堆栈跟踪(Stack trace)
编译器警告消息(Compiler Warning Messages)
调试器(Debugger)
执行分析器(Execution Profiler)（见第8章）
测试框架(Test Framework)（见第7-5节）