Atitit 《软件驱魔：调试和优化遗留代码的艺术 读后感attilax总结

Line 18: 第1章　预防药 1

Line 48: 第2章　调试技巧 55

Line 70: 第3章　理解问题 86

Line 84: 第4章　调试器内部机制 119

Line 101: 第5章　优化：内存占用 165

Line 125: 第6章　优化：CPU周期 202

Line 168: 第7章　最后的赠言 247

《软件驱魔：调试和优化遗留代码的艺术》   
译者序   
前言   
第1章　预防药 1   
1.1?核心问题 2   
1.1.1　上市时间的压力 2   
1.1.2　不确定的规格说明 4   
1.1.3　以功能特性换时间 5   
1.1.4　写下来 5   
1.1.5　复杂性 8   
1.2　防御性编程 9   
1.2.1　内聚和耦合 9   
1.2.2　错误输入检查 12   
1.2.3　数据范围 18   
1.2.4　日志 20   
1.2.5　文档 28   
1.2.6　为改变而设计 31   
1.2.7　增量精炼 33   
1.3　单元测试 34   
1.3.1　自动化测试的动机 35

1.3.2　实现框架的步骤 36   
1.3.3　框架扩展 43   
1.4　工具的配置 46   
1.4.1　使用编译器警告 46   
1.4.2　发行版本的设置 47   
1.5　机器相关性 48   
1.5.1　字节序 49   
1.5.2　内存对齐 50   
1.5.3　数据类型大小 51   
1.5.4　虚拟机的好处 52   
1.6　小结 53   
1.6.1　底线：为什么出现bug 54   
1.6.2　改进清单：bug主动预防 54   
第2章　调试技巧 55   
2.1　初始步骤 56   
2.1.1　复现问题 56   
2.1.2　无法复现的问题 56   
2.1.3　验证bug是真实存在的 59   
2.2　消除缺陷：快速修复 60   
2.2.1　检查近期改动 60   
2.2.2　使用跟踪信息 61   
2.2.3　似曾相识 61   
2.2.4　明确何时放弃 61   
2.3　消除缺陷：科学方法论 62   
2.3.1　一般步骤 62   
2.3.2　定位问题：增量集成法 63   
2.3.3　定位问题：二分法 64   
2.3.4　理解问题 64   
2.3.5　防范失误 65   
2.3.6　诊断工具 67   
2.3.7　基础调试操作 75   
2.4　保留记录 80   
2.4.1　个人记录 80   
2.4.2　协同开发下的记录 81   
2.5　小结 84   
第3章　理解问题 86   
3.1　知识是如何丢失的 87   
3.1.1　竞争 87   
3.1.2　人员流失 89   
3.1.3　升职 90   
3.2　难懂的代码 91   
3.2.1　设计问题 91   
3.2.2　混淆 95   
3.2.3　误导性的代码 104   
3.3　反向工程 105   
3.3.1　通用策略 105   
3.3.2　对策 111   
3.3.3　建立知识库 116   
3.4　小结 118   
第4章　调试器内部机制 119   
4.1　调试器的种类 119   
4.1.1　机器调试器与符号调试器 119   
4.1.2　调试基础：自定义构建 125   
4.1.3　调试基础：系统调用 136   
4.1.4　调试基础：解释器 151   
4.1.5　内核调试器 155   
4.1.6　界面：命令行与图形用户界面 157   
4.2　符号调试器扩展 157   
4.2.1　动态断点 157   
4.2.2　单步执行 158   
4.3　应对策略 159   
4.3.1　系统调用 159   
4.3.2　移除调试信息 160   
4.3.3　代码盐 161   
4.3.4　混合内存模型 162   
4.4　小结 163   
第5章　优化：内存占用 165   
5.1　被遗忘的历史 167   
5.2　程序的内存布局 168   
5.2.1　场景：单段程序 169   
5.2.2　场景：仅代码段和数据段 170   
5.2.3　场景：所有4种段类型 171   
5.3　代码段 172   
5.3.1　剪切粘贴式编程 172   
5.3.2　宏 175   
5.3.3　僵尸代码 177   
5.4　数据段 177   
5.4.1　双重用途的数据结构 178   
5.4.2　位域 180   
5.4.3　压缩算法 181   
5.5　栈 183   
5.5.1　活动记录 184   
5.5.2　函数参数 188   
5.5.3　局部变量 190   
5.6　堆 191   
5.6.1　内存池 192   
5.6.2　回收 196   
5.6.3　延迟实例化 197   
5.6.4　跟踪内存使用情况 199   
5.7　小结 201   
第6章　优化：CPU周期 202   
6.1　程序控制跳转 203   
6.1.1　标签与GOTO 203   
6.1.2　函数参数 205   
6.1.3　带可变参数的函数 206   
6.1.4　系统调用 207   
6.1.5　递归 210   
6.2　程序控制分支 211   
6.2.1　查找表 211   
6.2.2　switch与if-else 213   
6.2.3　常见情况放在前，罕见情况放在后 215   
6.3　程序控制循环 215   
6.3.1　循环不变量 216   
6.3.2　函数调用 217   
6.3.3　数组引用 219   
6.3.4　分解复合布尔表达式 220   
6.3.5　循环展开 221   
6.3.6　循环干涉 221   
6.3.7　提取程序分支语句 221   
6.4　内存管理 222   
6.4.1　处理开销 223   
6.4.2　引用局部性 226   
6.5　输入/输出 227   
6.5.1　缓存 228   
6.5.2　缓冲 229   
6.5.3　先进的技术 230   
6.6　异常 231   
6.6.1　动态注册模型 234   
6.6.2　静态表模型 235   
6.6.3　处理开销 235   
6.6.4　滥用异常 237   
6.7　昂贵的操作 237   
6.7.1　消除常见的子表达式 237   
6.7.2　浮点运算神话 237   
6.7.3　强度折减 239   
6.7.4　同步 240   
6.7.5　简写操作符的神话 243   
6.8　快速修复 243   
6.8.1　更好的硬件 243   
6.8.2　约束问题 244   
6.8.3　编译器设置 244   
6.9　小结 245   
6.10　信息汇总 246   
第7章　最后的赠言 247   
7.1　对于源代码完整性的其他威胁 248   
7.1.1　时髦技术：一个案例研究 248   
7.1.2　洗脑101 249   
7.1.3　真正的问题 249   
7.2　保持书面记录 250   
7.2.1　悄悄记录 250   
7.2.2　隐私的神话 250   
7.3　历史总是重演 251