第一图书网, tushu007.com

<<测试驱动的嵌入式C语言开发>>

图书基本信息

书名：<<测试驱动的嵌入式C语言开发>>

13位ISBN编号：9787111366232

10位ISBN编号：7111366239

出版时间：2012-1

出版时间：机械工业出版社

作者：James W. Grenning

页数：256

译者：尹哲

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

内容概要

本书深入介绍如何把测试驱动的开发方法应用于嵌入式C语言开发，第一部分介绍了两个开源的测试框架，通过测试驱动开发方法开发第一个模块：第二部分深入介绍了与系统中其他模块进行交互的代码的测试技术，如测试替身、仿制对象等；第三部分介绍了设计与持续改进代码，如写出更好代码的一些重要原则，建立可测并灵活设计的高级技术，改进已有代码的实践方法一一重构技术，改进遗留代码，以及编写和维护测试的指导原则。
本书的代码几乎全部用C写成，并且可以用于嵌入式的、受约束的开发和执行环境。

本书是作者多年实践经验的总结，实用性强，适合嵌入式C／C＋＋语言程序员、工程师阅读。

作者简介

作者：(美国)格伦宁 (James W.Grenning) 译者：尹哲 等格伦宁(James W.Grenning)在全球范围内从事培训以及咨询工作、他在软件开发的技术和管理方面拥有丰富的经验。
、他是把敏捷开发实践引入到嵌入式领域的带头人他发明了计划扑克(Planning Poker)他还是2001年2月《软件开发敏捷宣言》的作者之一。
尹哲是Odd-e团队中的一名敏捷教练。
他在世界各地从事培训和软件项目咨询，如测试驱动开发、系统工程实践以及分析实践等。
他在敏捷与精益教练方面拥有丰富经验，在研发管理和软件设计这两方面也造诣颇深。
他同时还是敏捷社区和开源软件的积极参与者，他与本书作者James相识多年，得其言传身教，对本书中所涉及的知识及其应用领域都相当了解。

书籍目录

译者序
推荐序一
推荐序二
前言
致谢
第1章　测试驱动开发
　1.1　为什么我们需要TDD
　1.2　什么是测试驱动开发
　1.3　TDD的机理
　1.4　TDD的微循环
　1.5　TDD的好处
　1.6　对于嵌入式开发的益处
第一部分　开　　始
第2章　测试驱动开发的工具和约定
　2.1　什么是自动化单元测试框架
　2.2　Unity：一个全部用C实现的自动化测试框架
　2.3　CppUTest：一个用C++实现的自动化单元测试框架
　2.4　单元测试也会崩溃
　2.5　“四阶段”模式
　2.6　我们到哪里了
第3章　开始一个C语言模块
　3.1　具有可测性的C模块的那些元素
　3.2　LED驱动都做些什么
　3.3　写一个测试列表
　3.4　写第一个测试
　3.5　先测试驱动接口再测试驱动内部实现
　3.6　增量式前进
　3.7　测试驱动开发者的状态机
　3.8　测试要做到FIRST
　3.9　我们到哪里了
第4章　一路测试直到完成
　4.1　从简单入手“生长”出解决方案
　4.2　保持代码整洁——边做边重构
　4.3　重复直到完成
　4.4　声明完成之前先向回走一步
　4.5　我们到哪里了
第5章　嵌入式系统TDD策略
　5.1　目标硬件的瓶颈
　5.2　双目标开发的好处
　5.3　双目标测试的风险
　5.4　嵌入式的TDD循环
　5.5　双目标的不兼容性
　5.6　和硬件一起测试
　5.7　欲速则不达
　5.8　我们到哪里了

第6章　是的，但是……
6.1　我们没那个时间
6.2　为什么不在写了代码之后再写测试
6.3　测试也需要维护
6.4　单元测试不能发现所有的bug
6.5　我们的构建时间太长
6.6　我们有现存的代码
6.7　我们的内存有约束
6.8　我们不得不和硬件交互
6.9　为什么要用C++的测试框架来测试C
6.10　我们到哪里了
第二部分　测试有合作者的模块
第7章　测试替身
7.1　合作者
7.2　脱离依赖关系
7.3　何时使用测试替身
7.4　用C来仿冒，下一步
7.5　我们到哪里了
第8章　监视产品代码
8.1　灯光调度测试列表
8.2　对于硬件和操作系统的依赖
8.3　链接时代换
8.4　监视被测试代码
8.5　控制时钟
8.6　先0后
8.7　处理多个的情况
8.8　我们到哪里了
第9章　运行时绑定的测试替身
9.1　测试随机性
9.2　冒仿函数指针
9.3　外科手术般地插入间谍
9.4　用间谍来校验输出
9.5　我们到哪里了
第10章　仿制对象
10.1　闪存驱动程序
10.2　MockIO
10.3　测试驱动开发驱动程序
10.4　模拟设备超时
10.5　这值得吗
10.6　用CppUMock来仿制
10.7　生成仿制对象
10.8　我们到哪里了
第三部分　设计与持续改进
第11章　SOLID、灵活并可测试的设计
11.1　SOLID设计原则
11.2　C语言中的SOLID模型
11.3　演进的需求和有问题的设计

11.4 用动态接口来改进设计
11.5 更灵活的基于类型的动态接口
11.6 做多少设计才是足够的
11.7 我们到哪里了
第12章 重构
12.1 软件的两个价值
12.2 三项关键技能
12.3 代码中的坏味道以及如何改进它们
12.4 转化代码
12.5 那性能和大小怎么办
12.6 我们到哪里了
第13章 为遗留代码加测试
13.1 遗留代码改动准则
13.2 童子军原则
13.3 遗留代码改动步骤
13.4 测试点
13.5 两步结构体初始化
13.6 崩溃直到通过
13.7 鉴别测试
13.8 为第三方代码做学习测试
13.9 测试驱动缺陷修正
13.10 增加策略测试
13.11 我们到哪里了
第14章 测试的模式与反模式
14.1 “喋喋不休”测试反模式
14.2 “拷贝－粘贴－调整－重复”反模式
14.3 “格格不入的测试用例”反模式
14.4 “测试组之间的重复”反模式
14.5 “不尊重测试”反模式
14.6 “行为驱动开发”测试模式
14.7 我们到哪里了
第15章 结束语
第四部分 附录
附录A 开发系统的测试环境
附录B Unity快速索引
附录C CppUTest快速索引
附录D 开始之后的LedDriver
附录E 操作系统隔离层的例子
附录F 参考书目

章节摘录

版权页：插图：最终，当一些保护测试就绪后，就应该可以比较安全地应用一些重构变化了，因为我们已经让代码准备好来迎接我们想做的改变。
有了测试来稳固遗留的行为，就可以安全地测试驱动新的行为了。
在13.4 节中，我们会看到在一些遗留代码中选择测试点。
13.4 测试点我们需要测试点来确认对代码所做事情的理解。
有些测试点很容易找到，有些则比较困难。
接缝函数调用在代码中的不同部分之间形成了“ 接缝 ”。
这些接缝是最好的测试点。
接缝让我们可以看到和干预被测试代码所做的事情。
Michael这样定义接缝：“ 接缝就是这样一个位置，在那里你可以改变程序的行为，却不用修改那个位置。
”接缝就是你可以使用测试替身来监视传给合作者数据的地方，让测试用例可以确保被测试代码给合作者传去了正确的指令。
函数调用接缝还是测试替身可以为被测代码提供间接输入的地方，这通过测试替身的返回值来完成。
你的代码中可能已经有了很多函数接缝。
这里的例外是那种铁板一块的函数。
为这种函数创建接缝可能会很危险。
加入一个“ 感知变量 ”可能会更安全一点。

媒体关注与评论

我想这本书将成为嵌入式软件工程师的“圣经”。
——Robert C.Martin

编辑推荐

《测试驱动的嵌入式C语言开发》由Robert C. Martin鼎力推荐，TDD领域不可多得的佳作。
测试驱动开发（TDD）是C语言开发人员需要掌握的一种现代编程实践。
它是一种不同于以往的编程方法——在一种紧致的反馈循环中写出单元测试与产品代码。
你时刻都会得到有价值的反馈。
你会在失误变成bug前就找到它们，你会有更多的时间花在为产品增加有价值的特性上。
James是为数不多的把测试驱动开发应用于嵌入式C的专家之一。
他在培训以及实践用C、C++、Java和C#做测试驱动开发方面拥有15年的经验，可以引领你从TDD的入门走向精通。

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