TURING 图灵程序设计丛书

C现代编程

集成开发环境、设计模式、极限编程、测试驱动开发、重构、持续集成

[日] 花井志生/著 杨文轩/译

提高开发质量
减轻工作负担

适合嵌入式开发者阅读

中国工信出版集团
人民邮电出版社
POSTS & TELECOM PRESS

TURING 图灵程序设计丛书

C现代编程

集成开发环境、设计模式、极限编程、测试驱动开发、重构、持续集成

[日] 花井志生/著 杨文轩/译

人民邮电出版社
北 京

图书在版编目（CIP）数据

C现代编程：集成开发环境、设计模式、极限编程、测试驱动开发、重构、持续集成/（日）花井志生著；杨文轩译. --北京：人民邮电出版社，2016.3

（图灵程序设计丛书）

ISBN 978-7-115-41775-6

Ⅰ. ①C… Ⅱ. ①花… ②杨… Ⅲ. ①C语言－程序设计 Ⅳ. ①TP312

中国版本图书馆CIP数据核字（2016）第029097号

内 容 提 要

本书从使用C语言进行嵌入式开发的特点入手，主要讲解了如何将集成开发环境、设计模式、极限编程、测试驱动开发、重构、持续集成这些现代编程方法应用到C语言的嵌入式开发中去，即将服务器端的通用设计方法、工具的使用方法、开发方式等逐一“翻译”为可以在C语言嵌入式开发过程中使用的方法。本书适合所有对C程序开发感兴趣的读者阅读。

◆ 著　[日] 花井志生
译　杨文轩
责任编辑　乐　馨
执行编辑　高宇涵
责任印制　杨林杰
◆ 人民邮电出版社出版发行　北京市丰台区成寿寺路11号
邮编　100164　电子邮件　315@ptpress.com.cn
网址　http://www.ptpress.com.cn
北京鑫正大印刷有限公司印刷
◆ 开本：800×1000　1/16
印张：16.5
字数：379千字　2016年3月第1版
印数：1－3 000册　2016年3月北京第1次印刷
著作权合同登记号　图字：01-2015-5652号

定价：59.00元

读者服务热线：(010)51095186转600　印装质量热线：(010)81055316
反盗版热线：(010)81055315
广告经营许可证：京东工商广字第 8052 号

站在巨人的肩上

Standing on Shoulders of Giants

iTuring.cn

译者序

选择翻译本书是在看到本书作者的经历与我的经历竟十分相似后做出的决定。我从学校毕业后，同样是先参与过几年的嵌入式开发，然后才转向服务器端应用程序开发的。在当时的嵌入式开发工作中，我们是在装有 Windows 系统的开发机中使用文本编辑器编写代码，然后再将其传输至 Linux 服务器上进行编译。如果出现了编译错误则修改后再编译，如此反复。另外，我们还要编写测试代码来测试自己编写的功能代码。由于我们的代码需要与日方的代码进行集成，然后才能在实际设备上测试，因此在国内几乎只能完成单元测试工作。等到最后进行集成测试时，问题就都暴露出来了。集成测试最初的那几天对于我们而言是最黑暗的几天，各种编译链接问题导致程序根本无法运行。

之后，我开始转向服务器端应用程序开发的相关工作。在这期间，接触并喜欢上了设计模式、重构、测试驱动开发与持续集成。特别是经过近几年的发展，开源的测试驱动开发技术与持续集成技术已经越来越成熟。现在，它们几乎已经被运用到我参与的所有项目中，可以说已经成为我们项目中的“标准配置”了。

遗憾的是，考虑到 C 语言并非面向对象的编程语言，而且嵌入式开发具有开发周期短、成本控制严格和测试依赖于硬件等特点，我还从来没有想过可以将这些开发技术和开发方法运用于嵌入式开发过程中。但本书作者做到了。当然，这并非简单地生搬硬套就可以实现的。在本书中，作者首先讲解了如何使用 C 语言进行面向对象方式编程。这样，适用于面向对象编程语言的设计模式和重构方法就都可以应用于 C 语言开发中。接着，作者讲解了如何实现测试驱动开发和持续集成。这部分内容对于使用 C++ 语言进行嵌入式开发的读者朋友同样具有参考价值。它们不仅可以帮助我们提高开发质量，还可以减轻我们的工作负担。

当然，由于篇幅限制，作者在本书中只精选了部分适用于嵌入式开发的设计模式来举例讲解。其实翻阅 Gof 的《设计模式》一书我们可以看到，诸如外观（Façade）模式、工厂方法（Factory Method）模式、单例（Singleton）模式等都在嵌入式开发中有用武之地。

另外，顺着作者提供给我们的思路继续思考可以发现，在改善开发过程方面，我们可以考虑以结对编程方式来编写代码；可以考虑引入 SonarQube 进行代码质量管理；还可以考虑引入 Phabricator 或是 ReviewBoard 进行代码评审等。希望大家能积极地在工作中使用它们，相信这一定能够给大家带来帮助。

非常遗憾的是，我已经远离 C 语言嵌入式设计开发工作多年，为了能够高质量地完成本书

的翻译工作，虽进行了复习，但自身技术水平有限，对于译文中出现的错误和不妥之处还望读者朋友不吝赐教。

为了翻译本书，我牺牲了许多陪伴家人，特别是与女儿玩耍的时间，想对他们说声抱歉。

最后，祝福所有读者朋友工作顺利，生活如意。

杨文轩

于 2015 年 12 月 27 日

前言

计算机可以以远超人类的速度，完全准确地执行简单的计算处理。在稍早以前，“应用程序开发自动化”这个概念是指自动生成应用程序代码，即只要编写了软件需求说明书，那么计算机就可以帮助我们编写全部的代码。部分从业人员对“编写代码只是简单的作业”这件事深信不疑，他们相信生成代码的“工厂”是存在的，这个工厂可以阅读和理解人类编写的设计书，并据此制作出应用程序。诚然，如果这些梦想能够成为现实，那么应用程序的开发将会发生巨大的变化。可以根据 UML 时序图（表示处理流程的示意图）生成代码的工具就曾经受到过大家的热捧。

但是，没过多久，人们就注意到制作时序图远比编写代码要麻烦得多。要想让设计内容能够自动转换为程序，设计书就必须编写得十分详尽且缜密，这样做所花费的人力和时间远比开发人员自己进行编程要多得多。实际上，在软件开发过程中将模糊的需求实现为代码，并非那么简单。

那么，应用程序开发自动化真的完全是白日梦吗？也并非如此。开发过程中也还是有一些“简单作业”的。在应用程序的开发现场你会意外地发现，居然有这么多的东西还没有实现自动化啊。不过最近，这种状况正在飞速地改变。现在的技术已经可以实现应用程序的编译、链接、测试以及在目标机器上的安装等工作的自动化，而且这些技术正在迅速普及。这样已经可以算得上是真正的应用程序开发工厂了吧。开发人员编写或是修改代码后，立即进行自动构建、测试，最后安装到目标机器上。在服务器端应用程序的开发现场，这样的做法已经是常态了。有些站点在 1 天内重新安装新应用程序的次数甚至会达到 100 次以上（GitHub 在 2012 年 8 月 23 日当天进行了 175 次部署。https://github.com/blog/1241-deploying-at-github）。

在这种背景下，产品或者应用程序如果跟不上变化的速度将无法生存下去。如今，许多服务器应用程序都在激烈地竞争着，更新只要稍微停滞，就会瞬间在竞争中落后。但另一方面，就算在这样惨烈的竞争下，也不能牺牲产品的质量来应对应用程序的更新。请大家试着在脑海中回忆一下 GMail 和 Amazon 这些服务器端应用程序的质量，可以说几乎从来没有发生过质量下降的情况。这是因为，只要质量稍微有所下降，或者是响应速度稍微变慢、操作上稍有不便，就会立即在与对手的竞争中落败，进而被市场淘汰。

另一方面，嵌入式设备的开发情况如何呢？虽然开发人员肯定不会被要求在同一天中多次升级新的应用程序（固件），但是每年都会被要求提高开发速度。当然，嵌入式设备的产品质量

要求也肯定比服务器端应用程序的要求更加严格（因为在某些领域，嵌入式系统中的缺陷会关系到人身安全）。也就是说，嵌入式开发的情况与服务器端应用程序开发的情况并无太大区别。那么，大家可能会想，在服务器端应用程序中研究出的开发方法，一定也能在嵌入式开发中发挥作用吧。话虽如此，服务器端应用程序开发与嵌入式程序开发相比，在使用的工具、编译器以及各种各样的环境方面均不相同。因此，直接将服务器端应用程序中的做法生搬硬套过来，多半是行不通的。

为了解决这个问题，本书将服务器端中通用的设计方法、工具的使用方法、开发方式等逐一“翻译”为可以在C语言嵌入式开发过程中使用的方法。除了对这些内容感兴趣的读者外，也希望那些认为自己一直以来的做法已经足够好了，不需要新的改变的从业者能够阅读本书。单纯地凭感觉不愿意使用本书中介绍的方法，与尝试过本书的方法后不愿意使用这些方法是完全不同的。本书虽然是在讨论嵌入式开发，但是所讨论的问题在嵌入式开发以外的C程序开发中也会或多或少存在着，因此希望从事嵌入式开发工作以外的编程人员也能够阅读本书。

最后，借此机会向对本书内容进行了细致审校的川崎编辑，以及我的妻子三姬子表达我最真挚的感谢。

花井志生

于2013年8月

编写本书的初衷

编写本书的契机是几年前与客户一起工作时的经历。在服务器端开发领域，本书所介绍的面向对象、设计模式、TDD、重构以及持续集成等理论、开发方法已经非常普及了，但是在嵌入式的世界中却使用得非常少。另一方面，在书店翻阅介绍C语言的书籍时，我发现这些书籍大部分都是讲解语法与算法的，鲜有介绍开发方法的著作。

编写本书的目的是在传统的C语言编程开发方式与服务器端领域的最新开发方式之间搭建一座桥梁。在编写过程中，我谨记要让那些具有C语言语法知识与嵌入式开发经验，但缺少其他相关知识的读者也能够理解本书的内容。当然，我并不认为本书介绍的开发方法绝对正确，我自身在开发过程中运用本书所介绍的开发方法时也偶有碰壁，但是如果有读者朋友能够在自身工作中稍微运用下本书中介绍的内容，不论成功也好，失败也罢，只要能从中吸取教训、积累经验，那么我都将深感欣慰。

目录

第 1 章
概要

C 语言（后文简称 C）诞生于 1972 年，至今已有 40 多年的历史。笔者初次接触 C 是在 30 多年前。那时候，我们使用的计算机被称为“单片机”，编写程序一般使用 BASIC 解释器。由于当时 C 编译器是需要付费的，因此个人（特别是学生）很难接触到 C。但在那之后，IT 业界发生了很大变化。如今，包括 C 在内的许多编程语言的编译器都已开源了，大家均可使用。

因为 C 的特点是提供了许多低级处理功能，因此常被称作“高级汇编”。结构体、指针、数组、局部变量等与内存地址是一一对应的关系，因此可以说 C 非常匹配机器语言。实际上，许多 C 编译器都允许在代码中直接编写汇编代码，而如果是用于嵌入式开发的 C 编译器，一般还会允许直接实现硬件中断。

1.1 现在 C 依然很热门

现在出现了以 C++ 语言（后文简称 C++）为首的许多编程语言，它们都以取代 C 为目标。笔者首次使用 C++ 大约是在 20 年前。当时笔者认为将来 C 会被 C++ 吸收和替代。但前几天，笔者在 TIOBE（`http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html`）的统计中发现 C 排在了第一位。TIOBE 是定期统计工程师们正在使用的编程语言的网站。通过在该网站上查看 C 的市场份额变化（`http://www.tiobe.com/content/paperinfo/tpci/c.html`），可以得出一个非常有趣的结论：C 的市场份额在 2002 年曾经到达 20% 以上，然后逐年减小，在 2008 年降至 13%，之后开始触底反弹，在 2013 年再次突破 20%，目前也仍维持在 15%～20%；另一方面，C++ 的市场份额（`http://www.tiobe.com/index.php/paperinfo/tpci/C__.html`）自 2002 年以来逐年减少，虽然曾经高达 17% 左右，但现在已跌破了 10%（图 1-1）。

那么到底是什么原因导致上述现象呢？如果仅从 PC 领域来看的话，这是一个很奇怪的现象。除去极小部分问题以外，C 与 C++ 之间具有良好的兼容性。C 可以实现的功能，使用 C++ 也都可以实现。因此，工程师们没有必要特意回避使用 C++。当然，C++ 的某些功能相对更加复杂，但是只要不使用这部分功能不就可以了吗？

让我们退一步将视角再扩大一些。最近，您应该可以经常听到“PC 的销售额增长缓慢，但智能手机和平板电脑的销售额却快速增长”这样的新闻，或是“混合动力汽车的出现使汽车的

构造更加复杂，一台车辆中嵌入了多台小型计算机设备”这样的消息吧。这说明计算机发挥威力的世界不仅仅局限于 PC 领域。

图 1-1 TIOBE 统计的 C/C++ 的市场份额的变化

（出处：http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html）

如果浏览嵌入式开发方面的网站就会发现，2009 年至 2013 年间，嵌入式 CPU 的销售额以每年 10% 左右的速度增长（EMBEDDEDNEWS：http://www.embeddednews.co.uk/ArticleItem.aspx?Cont_Title=Smartphones+boost+microcontroller+shipments+with+Arm+seeing+major+growth；PCM007：http://pcb007.com/pages/zone.cgi?a=58069）。有报道称，著名嵌入式产品供应商 Microchip 公司截至 2011 年 9 月 CPU 的出货量已经达到 100 亿个。报道中还提到完成从 90 亿至 100 亿的跨度只花费了 10 个月的时间（http://www.techrockies.com/microchip-billion-pic-microcontrollers-

shipped/s-0038192.html）。

笔者认为，嵌入式市场的繁荣发展与 C 市场份额的增加有着密切的联系。虽然统称为嵌入式设备，但是其范围非常广泛。以美国微芯科技公司的 PIC18F14K50（http://www.microchip.com/wwwproducts/Devices.aspx?dDocName=en533924）嵌入式芯片为例，该芯片是内置了 8 位 CPU 的嵌入式设备。由于其内置了 USB 控制器，因此可以方便地使用它制作出 USB 设备（图 1-2）。

器件	程序存储器		数据存储器		I/O[1]	10位 A/D（通道）[2]	ECCP (PWM)	MSSP		EUSART	比较器	8/16位定时器	USB
	闪存（字节）	单字（指令数）	SRAM（字节）	EEPROM（字节）				SPI	主 I^2C^{TM}				
PIC18F13K50/ PIC18LF13K50	8K	4096	512[3]	256	15	11	1	有	有	1	2	1/3	有
PIC18F14K50/ PIC18LF14K50	16K	8192	768[3]	256	15	11	1	有	有	1	2	1/3	有

图 1-2 具有 USB 功能的嵌入式芯片的规格

（出处：http://ww1.microchip.com/downloads/en/DeviceDoc/41350E.pdf）

现在市面上很多 USB 设备都是使用这样的微型单片机制作的。

这样的单片机只支持 C 或是汇编，不支持 C++。程序可以使用的内存只有 16 KB，RAM 更是只有 768 B。当想要链接一些高级功能函数（例如 sprintf）时，就会因为内存不足而发生链接错误。这样，即使单片机支持 C++ 也无法正常使用输入输出流和模板功能，C++ 的优点也就得不到发挥。以前，这种嵌入式芯片性能很差，通常都是编写汇编程序。但是最近，芯片性能得到了很大的提升，即使使用 C 编写程序，在运行速度上也不会有问题。（明明都是 8 位单片机和相同的内存容量，但是现在的处理速度比以前提升了 2 个数量级，笔者那个年代的人一定会对此非常惊讶吧。）于是，嵌入式设备提供的功能也越来越强大。笔者认为，这就是 C 的市场份额逐渐扩大的原因吧。

1.2 使用 C 进行嵌入式开发的特点

使用 C 进行嵌入式开发有如下特有的难点。

可以说，在很多情况下，测试那些控制硬件的程序都很困难。通常，我们可以通过 ICE（In-Circuit Emulator，在线仿真器）来追踪代码的运行，但是要测试代码在临界值下是否可以正确工作，必须事先让硬件返回这个临界值。此外，为了测试硬件在发生错误和故障时会如何工作，必须将真实的硬件作为测试对象，而这在实际开发中是非常困难的。

几乎所有控制嵌入式设备的程序可以使用的资源都是非常有限的。为了能达到硬件所要求的响应速度，程序一般都会采用中断的方式即时响应硬件的状态变化。某些时候，还会使用专门的双端口存储器（Dual Port Memory，DPM）和直接内存访问（Direct Memory Access，DMA）进行高速数据传输。同时，嵌入式设备中搭载的内存容量也非常有限，因此程序必须尽

可能地减小运行过程中使用的内存量，并且在使用后必须尽快释放内存空间。此外，栈的空间也非常有限，必须非常注意局部变量的数量和程序中函数的调用深度。也正因为如此，嵌入式设备中的程序有时不得不以牺牲代码的可维护性为代价，换取运行时的高性能和低内存使用量。

与服务器程序不同，大部分嵌入式设备通常都大量散布在各个地方，这样会导致设备维护特别困难。如果设备数量很多，那么就会很容易出现那些在测试环境中很难重现的Bug。假设某天散布在各地的1000台设备在某个地方发生了1次故障，那么想要在测试环境中的1台设备上重现这个Bug，从概率上计算就需要1000天。即使运气很好，很快找到了Bug所在，那也必须要将修改后的程序发送到各个地方进行安装。而嵌入式设备的使用者通常并非IT专家，因此还必须确保程序安装步骤足够简单才行。

PC和服务器中一般都使用了OS（操作系统）。嵌入式设备如果规模足够大，也会安装诸如ITRON和嵌入式Linux等OS；但是如果规模不大，就没有OS这个后盾了。这时，通常情况下OS所提供的基本功能（例如文件系统）等都必须自己去实现。

C与其他大多数编程语言不同，很多事情都依赖于程序员的操作。例如，如果程序员将一个局部变量数组的下标弄错了，甚至会导致保存了函数返回值地址的内存数据被破坏。要确保堆空间中有足够的内存，必须时刻注意手动释放内存。而且，即使释放了指向该内存空间的指针也不能省心，还必须注意不能再使用该指针。既然您购买了本书，那想必您也是C程序员吧。以上这些痛苦，您一定也经历过吧。如今，PC中安装的OS都带有内存保护措施，即使程序中存在上面提到的Bug，至少也可以在一定程度上检测出来。但是嵌入式系统中大多不具有内存保护功能，稍微不注意产生了问题，就可能需要花费很大气力去找出原因。而且与数据中心里被谨慎保护着的服务器不同，嵌入式系统还受到许多其他外界因素的影响，如静电、振动、温度等；甚至连防盗装置发出的强力电子信号，以及电池快耗尽时不稳定的电压都会对嵌入式系统产生影响。当故障发生时，服务器应用程序还会留下日志和内存转储信息，通过分析这些信息可以帮助我们迅速找到故障原因，但是在嵌入式系统中仅有一部分能够使用这些功能。

通常情况下，面向企业的应用程序都会使用数年甚至是10年以上，而另一方面，嵌入式设备虽然千差万别，但是其中有些设备生命周期却非常短。虽然生命周期长的系统在开发中有其特有的难点，但是生命周期短的系统的开发也有其自身的困难，其中一个难点就是开发成本。厂商必须在产品的生命周期内回收产品的开发成本，如果产品的生命周期很短，则开发成本也会被严格限制（当然这取决于销售数量与利润，但与可以通过网络下载的软件相比，嵌入式设备的销售还包括了流通环节。如果产品生命周期很短，那么可以卖出的设备数量自然也就十分有限）。以有限的成本开发高质量的软件是一项非常困难的工作。而且由于成本问题，现在许多公司的软件开发工作正在部分或者全部外包给劳动力更加廉价的国家和地区。如何才能确保外包出去的代码质量呢？外包开发的代码出现问题了，自己能够维护吗？当然，这些已经超越了技术问题的范畴，无法用技术方法解决。

不仅是软件，硬件也可能存在Bug。在用于PC的CPU中，Bug被称为“勘误表”（Errata）。一旦在量产开始后才发现Bug，不论是改正还是回收已经出售的产品，都会产生非常

高额的费用，这对企业来说是不现实的。因此，除了非常严重的 Bug 外，其他 Bug 都必须通过修改软件来应对。如果产品的寿命很长，其中使用的部分部件已经停产了，还需要寻找其替代品。有时后面批次的部件规格发生了变化，甚至也有可能找不到替代品。为了避免发生这些情况，虽然可以事先准备一些部件的存货，但是由于很难估计到底需要多少备货，而且仓库储存也需要成本，因此很难确保充足的备货量。最后，不得已只能在软件中进行应对。

如上所述，嵌入式开发中存在着各种各样的困难。其实在生活中，汽车、家电、AV 设备等都是工程师在这样困难的环境中凭借他们不断的努力创造出来的，只是我们从来没有意识到而已（认为设备正常工作是理所当然的事情）。

1.3 本书的目标

每位嵌入式开发人员都会在心中有个疑问：自己编写的代码到底能不能在任何情况下都正常运行呢？上一节所描述的嵌入式开发中特有的难点，有些虽然已经不能用纯粹的技术问题去解决，但是还是有一些问题可以通过使用工具和开发方式来应对。本书就将介绍这些工具和开发方式。

1.3.1 C与集成开发环境

现在许多编程语言都可以使用集成开发环境（IDE）进行开发。实际上，美国微芯科技公司已经为嵌入式开发人员提供了基于 Java 开发工具 NetBeans 改造的、适用于 PIC 开发的 IDE（图 1-3）。

但是在现实中的开发现场，情况却并非如此。笔者前几日有幸参观某厂商的嵌入式设备开发现场，多半的技术人员还是使用文本编辑器和命令行进行开发的。

图 1-4展示了传统的 C 嵌入式开发环境。代码被保存在服务器中，开发人员从服务器中获取代码（签出），然后在自己的 PC（主机）上使用文本编辑器修改代码。修改后，在自己的 PC 上进行编译、链接，并将目标代码烧录到目标机中。如果开发环境允许调试，会为每位开发人员分配 ICE；如果不允许调试，则需要依靠输出日志或是 `printf`进行调试，亦或使用芯片厂商提供的目标机模拟器进行调试。不管怎样，嵌入式设备的调试与 PC 上的可视化调试相比，更像是摸着石头过河，遇到难以解决的 Bug 时，花上几天的时间去攻克它也不足为奇。

到底是文本编辑器好还是 IDE 好，现在仍然时常会有争议。笔者认为应当在理解双方的特点后根据情况进行选择，不应当执着于其中任何一方。例如，调试、代码导航、宏展开的解析、重构等在 IDE 上进行作业会更好。本书将简单介绍 Eclipse 的 C/C++ 版的安装方法，在后面的章节中也会使用 Eclipse 讲解 C 开发。