Broadview
www.broadview.com.cn

站在C++98的最高峰沐浴C++11的阳光

Boost
程序库完全开发指南
——深入C++“准”标准库

（第2版）

罗剑锋 著

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
http://www.phei.com.cn

Boost

程序库完全开发指南

——深入C++“准”标准库

（第2版）

罗剑锋 著

電子工業出版社
Publishing House of Electronics Industry
北京•BEIJING

内 容 简 介

Boost 是一个功能强大、构造精巧、跨平台、开源并且完全免费的 C++程序库，有着“C++‘准’标准库”的美誉。

Boost 由 C++标准委员会部分成员所设立的 Boost 社区开发并维护，使用了许多现代 C++编程技术，内容涵盖字符串处理、正则表达式、容器与数据结构、并发编程、函数式编程、泛型编程、设计模式实现等许多领域，极大地丰富了 C++的功能和表现力，能够使 C++软件开发更加简捷、优雅、灵活和高效。

本书基于 2012 年 8 月发布的 Boost1.51 版，介绍了其中的所有 117 个库，并且结合 C++11 标准详细、深入地讲解了其中数十个库，同时实现了若干颇具实用价值的工具类和函数，可帮助读者迅速地理解、掌握 Boost 的用法及其在实际开发工作中的应用。

本书内容丰富、结构严谨、详略得当、讲解透彻，带领读者领略了 C++的最新前沿技术，相信会是每位 C++程序员的必备工具书。

图书在版编目（CIP）数据

Boost 程序库完全开发指南：深入 C++“准”标准库 / 罗剑锋著. —2 版. —北京：电子工业出版社，2013.1
ISBN 978-7-121-19089-6

Ⅰ. ①B… Ⅱ. ①罗… Ⅲ. ①C 语言－程序设计－指南 Ⅳ. ①TP312-62

中国版本图书馆 CIP 数据核字（2012）第 285458 号

责任编辑：孙学瑛
印　　刷：北京中新伟业印刷有限公司
装　　订：三河市皇庄路通装订厂
出版发行：电子工业出版社
　　北京市海淀区万寿路 173 信箱　邮编 100036
开　　本：787×980　1/16　印张：38.75　字数：801 千字
印　　次：2013 年 1 月第 1 次印刷
印　　数：3000 册　　定价：99.00 元

凡所购买电子工业出版社图书有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系，联系及邮购电话：（010）88254888。

质量投诉请发邮件至 zlts@phei.com.cn，盗版侵权举报请发邮件至 dbqq@phei.com.cn。

服务热线：（010）88258888。

推荐序

最近一年我电话面试了数十位 C++应聘者，惯用的暖场问题是“工作中使用过 STL 的哪些组件？用过 Boost 的哪些组件？”得到的答案大多集中在 vector、map 和 shared_ptr。如果对方是在校学生，我一般会问问 vector 或 map 的内部实现、各种操作的复杂度，以及迭代器失效的可能场景。如果是有经验的程序员，我还会追问 shared_ptr 的线程安全性、循环引用的后果及如何避免、weak_ptr 的作用等。如果这些都回答得不错，进一步还可以问问如何实现线程安全的引用计数，如何定制删除动作等。这些问题让我能迅速地判别对方的 C++水平。

我之所以在面试时问到 Boost，是因为其中的许多组件确实可以用于编写可维护的产品代码。Boost 包含近百个程序库，其中不乏具有工程实用价值的佳品。每个人口味与技术背景不一样，对 Boost 的取舍也不一样。就我的个人经验而言，首先可以使用绝对无害的库，例如 noncopyable、scoped_ptr、static_assert 等，这些库的学习和使用都比较简单，容易入手。其次，有些功能自己实现起来并不困难，正好 Boost 里提供了现成的代码，那就不妨一用，比如 date_time 和 circular_buffer 等。然后，在新项目中，对于消息传递和资源管理可以考虑采用更加现代的方式，例如用 function/bind 在某些情况下代替虚函数作为库的回调接口、借助 shared_ptr 实现线程安全的对象回调等。这二者会影响整个程序的设计思路与风格，需要通盘考虑，如果正确使用智能指针，在现代 C++程序里一般不需要出现 delete 语句。最后，对某些性能不佳的库保持警惕，比如 lexical_cast。总之，在项目组成员人人都能理解并运用的基础上，适当引入现成的 Boost 组件，以减少重复劳动，提高生产力。

Boost 是一个宝库，其中既有可以直接拿来用的代码，也有值得借鉴的设计思路。试举一例：正则表达式库 regex 对线程安全的处理。

早期的 RegEx 类不是线程安全的，它把“正则表达式”和“匹配动作”放到了一个类里边。由于有可变数据，RegEx 的对象不能跨线程使用。如今的 RegEx 明确地区分了不可变（immutable）与可变（mutable）的数据，前者可以安全地跨线程共享，后者则不行。比如正则表达式本身（basic_regex）与一次匹配的结果（match_results）是不可变的；而匹配动作本身（match_regex）涉及状态更新，是可变的，于是用可重入的函数将其封装起来，不让这

些数据泄露给别的线程。正是由于做了这样合理的区分，RegEx 在正常使用时就不必加锁。

Donald Knuth 在“*Coders at Work*”一书里表达了这样一个观点：如果程序员的工作就是摆弄参数去调用现成的库，而不知道这些库是如何实现的，那么这份职业就没啥乐趣可言。换句话说，固然我们强调工作中不要重新发明轮子，但是作为一个合格的程序员，应该具备自制轮子的能力。非不能也，是不为也。

C/C++语言的一大特点是其标准库可以用语言自身实现。C 标准库的 strlen、strcpy、strcmp 系列函数是教学与练习的好题材，C++标准库的 complex、string、vector 则是类、资源管理、模板编程的绝佳示范。在深入了解 STL 的实现之后，运用 STL 自然手到擒来，并能自动避免一些错误和低效的用法。

对于 Boost 也是如此，为了消除使用时的疑虑，为了用得更顺手，有时我们需要适当了解其内部实现，甚至编写简化版用做对比验证。但是由于 Boost 代码用到了日常应用程序开发中不常见的高级语法和技巧，并且为了跨多个平台和编译器而大量使用了预处理宏，阅读 Boost 源码并不轻松惬意，需要下一番功夫。另一方面，如果沉迷于这些有趣的底层细节而忘了原本要解决什么问题，恐怕就舍本逐末了。

Boost 中的很多库是按泛型编程的范式来设计的，对于熟悉面向对象编程的人而言，或许面临一个思路的转变。比如，你得熟悉泛型编程的那套术语，如 *concept*、*model*、*refinement*，才容易读懂 Boost.Threads 文档中关于各种锁的描述。我想，对于熟悉 STL 设计理念的人而言，这不是什么大问题。

在某些领域，Boost 不是唯一的选择，也不一定是最好的选择。比如，要生成公式化的源代码，我会首选用脚本语言写一小段代码生成程序，而不用 Boost.Preprocessor；要在 C++程序中嵌入领域特定语言，我会首选用 Lua 或其他语言解释器，而不用 Boost.Proto；要用 C++程序解析上下文无关文法，我会首选用 ANTLR 来定义词法与语法规则并生成解析器（parser），而不用 Boost.Spirit。总之，使用 Boost 时心态要平和，别较劲去改造 C++语言。把它有助于提高生产力的那部分功能充分发挥出来，让项目从中受益才是关键。

要学习 Boost，除了阅读其官方网站的文档、示例与源码之外，最好能有一本比较全面的中文书在手边随时翻阅。对于不谙英文的开发者，这更是可幸之至。您手上这本《Boost 程序库完全开发指南》是很好的使用指南与参考手册。作者由浅入深地介绍了 Boost 的大部分常用内容，能让读者迅速了解 Boost，并从中找到自己需要的部分。拿到这本书稿之后，我有粗有细地阅读了一遍，总体来看，作者水平很高，也相当务实，对 C++和 Boost 的理解与运用很到位，我从这本书学到了不少新知识。为此，我乐于向希望学习 Boost 程序库的开发者推荐这本靠谱的书。

须知“功不唐捐”，作为一名现代 C++程序员，在 Boost 上投入的精力定能获得回报。

陈硕

《代码大全》译者之一

中国·香港

第2版前言

本书第 1 版面世至今忽忽然不觉已经两年有余，其间多蒙读者厚爱，褒奖有加，不胜感激，在此聊写些文字，以志心性。

闲谈碎语 C++

这两年里对于 C++社区来说最重大的事件莫过于是 C++11 标准的发布了：历经十余年的磨砺，崭新的 C++终于扬刀出鞘，诸多新语言特性和库的加入令 C++旧貌换新颜，从此程序员手里的这把宝刀更是增添了无穷的威力，上天入地屠龙伏虎，不在话下。

最近开源界的一桩新闻也不得不提一下：著名的 C/C++编译器 GCC 于 2012 年 8 月完成了从 C 实现到 C++实现的转换。虽然这件事比不上 C++11 发布，但足可以从一个侧面证明 C++的实力已经得到了开源界的高度认可，今后没有什么是 C++做不出来的了。

作为 C++标准的后备，随着 C++11 的正式发布，Boost 程序库现在进入了一个新的历史时期：一方面依据新标准不断完善自身，另一方面则秉承传统继续开拓 C++11 未涉及的领域。这两方面可以从 Boost 历次巨细靡遗的更新记录中看出来——不断修正既有库中不符合标准的地方，同时再谨慎地引入新的组件，“小步快跑”地奔向“康庄大道”。

由于 Boost 程序库正逐渐向 C++11 标准靠拢，曾经的“准标准库”美誉已经不太合适了，它更像是一个比 C++标准库更加“兼容”、更加“标准”的“超级标准库”——使用 Boost 可以完全消除 C++11 和 C++98 之间的差异，稍微有点夸张地说：学习 Boost 就相当于学习现代 C++，使用现代 C++必然避不开 Boost。

国内外的许多公司都已经把 Boost 作为自己源码资产的一部分，在这些高质量的软件组

件之上开发产品，更激进的如 Facebook 则是走在了潮流的前面，同时使用最新的 C++11 和 Boost 来开发软件（如开源的 folly）。面对新技术，我们应该克服心中的怯懦和懒惰，积极学习，力争“站在 C++98 的最高峰沐浴 C++11 的阳光”。

对第 1 版的改动

本书第 1 版撰写于 2010 年年初，彼时 C++新标准尚未确定，国内了解、使用 Boost 的人也不多，故书的内容偏重于入门和介绍。而现在的情况则已经大有不同：C++11 标准相当于是一个全新的语言，获得了诸多编译器生产商的积极响应，移动互联网的兴起也使得国内的程序员群体开始越来越多地关心起 C++和 Boost，整体 C++水平有所提升。

鉴于这些变化，第 2 版在保持原书风格的同时做了适当调整，删去了一些浅显的部分，并依据最新的 C++11 和 Boost 程序库全面更新，较第 1 版略增加深度，但仍然还是以入门为主，不过多介绍库的实现细节。由于 C++11 标准有很多新的语言特性和库，包含了部分 Boost 库原有的功能，故作者弱化了一些与 C++11 语言特性重复的库，对于库组件则着重讲解符合 C++11 标准的功能。[①]

本书第 2 版几乎每页都较前一版有修改，各章的重大变化列举如下。

- 第 1 章 ： 重新组织了结构，分别介绍 UNIX 和 Windows 开发环境；
- 第 3 章 ： 增加对 unique_ptr 的介绍，补充完善对 weak_ptr 的论述；
- 第 4 章 ： typeof 库推荐改用 C++11 的 auto，删除 4.11.3 节；
- 第 6 章 ： 增加对 C++11 static_assert 关键字的介绍；
- 第 7 章 ： array、unordered、tuple 库更新为 C++11 标准；
- 第 8 章 ： foreach 推荐改用 C++11 的 for，minmax 库更新为 C++11 标准；
- 第 9 章 ： random 库更新为 C++11 标准；
- 第 10 章 ： 新增 cpu_timer 库，system 更新为 C++11 标准，filesystem 更新为 V3；
- 第 11 章 ： result_of、ref、bind、function 等库更新为 C++11 标准；
- 第 12 章 ： thread 库更新为符合 C++11 标准的 V3（变动非常大）；
- 第 14 章 ： 更新至 Boost1.51 版；
- 附录 ： 补充了对 C++11 标准的简介，同时删除了用处不大的网络资源和 ref_array 源代码。

① C++11 标准虽然已经公布一年有余，但可能是因为内容太过庞杂，作者尚未见到有全面完整论述的专著，书中对 C++11 的理解难免有错漏不足，还请读者见谅。

第 1 版的版式个人感觉比较满意，但代码行多的时候还是不容易阅读，所以本次对部分重要的代码片段加粗醒目，用于提醒读者注意要点。

作者最早的开发环境是 Windows，近年来工作重心逐渐转移，Mac OS X 和 Linux 成了主力工作环境（个人非常欣赏 Apple 公司），因此第 2 版中淡化了 Windows 操作系统的色彩，不再细述 VC 相关的配置，示范代码也基本改为 UNIX 风格，请读者明鉴。

致谢

同第 1 版一样，我首先要感谢 Bjarne Stroustrup 博士、C++标准委员会和 C++/Boost 社区，感谢他们长久以来的坚持和努力，为我们带来了 C++11 标准和越来越接近完美的 Boost 程序库。

然后是我的家人：感谢我的父母、弟弟和妻子，你们永远是我生命中最重要的人；感谢即将满 4 岁的女儿，这本书是和你一同成长起来的，希望你将来能够读懂它。

最后感谢读者选择本书，希望和您再次一起分享学习 C++和 Boost 的快乐。

罗剑锋

2012 年 10 月 12 日 于 北京 WFC

第1版前言

屈指算来，接触 C++语言至今已经有十余个年头了。回首往事，不禁感慨良多。

缘起

1996 年我上大学最开始学的是 Pascal，不得不说，Pascal 严谨的程序风格确实很适合作为一门教学语言，然而用于实际开发就不那么合适了（直到出现 Delphi）。由于当时学校并未开设 C 语言课程，因此在 Pascal 课程结束后我就买书自学 C/C++语言，并在次年报名计算机软件专业技术资格和水平考试，靠着一点点编程和考试的“天分”获得了高级程序员资质（当年很热衷考级考证，后来就“淡定”多了）。虽然有了资格证，但我仍然算是个 C++的初学者，对于 C++的认识还处于 C 的面向过程和简单的基于对象层次上。

新千年伊始我考入了北京理工大学就读研究生，因为跟导师做项目开始接触 STL 与 C++标准库，大概是 2005 年从 1.33 版结识了 Boost，这才真正领略了 C++的精髓。那段时期 Java 和 C#正在国内大行其道，C++则势单力薄，有关 STL 和 C++标准的技术书籍寥寥无几，而讲解 Boost 的书更是为零，故对 Boost 的学习基本只能靠自己的摸索与实践。好在 Boost 自带的文档相当丰富（尽管看全英文的资料十分辛苦），而且源码也写得比较清晰规范，在熟悉了 STL 的基础上学习 Boost 倒也并不算太难。

但 Boost 的一个最大的特点就是“庞大”，功能组件众多，要想把它全部装进脑子里融会贯通基本上是不可能的，使用时需要经常查阅英文文档，相当麻烦。因此，在学习的过程中，我逐渐产生了编写学习笔记的想法。一开始只是一个简单的纯文本文件，记录了一些使用经验的片断，随着积累不断增加，纯文本形式已经不能够满足知识整理的需求了，于是我又把这些文字迁移到了 Word 文档里，把使用经验分类编目，进行较系统的归纳梳理。慢慢地，这份学

习笔记居然有了上万字的规模，成为了一份很好的 Boost 备忘参考，在日常的开发工作中给了我很大的帮助——就像《设计模式》一书中所说的那样，捕获了很多使用 Boost 解决问题的实践经验，避免了重复发现。不过，这份资料一直仅限于我个人使用，属于“自娱自乐”的作品，从未示人。

时间一晃到了 2010 年 1 月份的某天夜里，不知道是什么原因我忽然失眠了，躺在床上翻来覆去怎么也睡不着。突然，一个念头闯入了脑海：把 Boost 开发经验整理出版吧，让更多人能够分享这些知识，正所谓“独乐乐，与众乐乐，孰乐？”这个大胆的想法的出现让我那天的失眠又延长了几个小时——关于书的各种构想在头脑中“肆虐横行”。

随后的几天里我就把这个想法付诸行动了，虽然以前也写过并发表很多文章，也在网上印刷了几本个人文集，但出版正式的书还是第一次。在把学习笔记进一步整理完善，编写出较完整的结构和一个样章后，我就开始联系出版社了。当初并没有多大的信心，毕竟我这个作者名不见经传，也没有什么资历、背景和名气（而且还是个“网盲”，从未跟随潮流开个人博客）。很幸运，发出的第一个 E-mail 就是电子工业出版社，而且编辑也在第一时间回复了我，这才给了我以持续写作完成全书的动力。

写作过程中我也进一步加深了对 Boost 的认识，澄清了许多原来未曾注意到的细节。原本只打算写 20 万字左右、三百多页，但写到中途才发现 Boost 库的博大精深远非当初的理解，也意识到了自己当初学习的肤浅。历经了近半年近乎不眠不休的努力，最终呈现给读者的是这本厚达 500 多页的图书，文字量是最初学习笔记的数十倍，内容也翔实丰满了很多——达成这个结果，我个人可以说是问心无愧了。

C++与 Boost

C++较 Java 和 C#等语言的一个最大不同在于它并非是由某个公司或个人把持的，它的真正发展动力来自于广大程序员。Boost 就是这样的一个典范，它成功地填补了从 C++98 到 C++0X 这“失落的十年”间的空白，在竞争对手 Java 和 C#不断更新版本新增特性的时候以库的形式极大地增强了 C++的能力，使 C++不至于因为标准规范的滞后而落后于时代，而且 Boost 还深层次地挖掘了 C++的潜力，开创了泛型编程、模板元编程、函数式编程等崭新的境界。

就个人来说，我比较喜欢的 Boost 版本有两个，分别是 1.35 和 1.39。1.35 版增加了 asio、bimap、circular_buffer 等许多重要组件，而 1.39 版则增加了 signals2 库，这两个版本都在我的工作用机上停留了相当长的时间。落笔之时，Boost 已经更新到了 1.43 版，成长为一个相当完善、全面、强大的 C++程序库。可以毫不夸张地说，现在的 C++程序员，

如果不熟悉 Boost，那么至少丧失了一半使用 C++的好处，同时会多耗费数倍的开发精力和时间。

随着 C++0X 标准的即将来临，Boost 程序库的发展也出现了加速的趋势，由原来间隔数月不定期更新版本，改为定期（每 3 个月左右）发布新版本，而且每个新版本都会包含大量极有价值的更新内容。因此，希望读者在阅读本书时及时访问 Boost 的官网（http://www.boost.org），以便获取最新的版本。

感谢读者选择本书，再说一句真心的“套话”（笑）：限于作者水平有限，书中错漏在所难免，敬请读者原谅、指正。

致谢

首先我要感谢整个 C++群体，特别是：C++语言的发明者 Bjarne Stroustrup 博士——他给我们带来了美妙的 C++；然后是 Alexander Stepanov 和 C++标准委员会——他们把 STL 引入了 C++，开创了 C++的现代编程风格；以及 Beman G. Dawes、Boost 程序库的所有作者和 Boost 社区——他们为我们奉献了如此高水准的程序库。

其次我要感谢电子工业出版社博文视点公司，他们给了我这个把自己的开发经验出版成书的机会，在把潦草的个人学习笔记变成正式图书的过程中他们付出了艰辛的努力。还要感谢陈硕先生，他审阅了本书的部分手稿，提出了很多有价值的参考意见，并慨然为本书撰写序言。

接下来我要感谢我的家人：感谢我的父母和弟弟，他们永远是我生命中最重要的人；感谢我的妻子，她自始至终都支持我的写作，并担负了大部分照顾孩子的家务（虽然偶有怨言）；还要对已满一岁半的女儿说声抱歉，为了写作本书，我已经牺牲了很多陪她玩耍的时间。

我还要感谢黄美华、冯薇、戚天龙、罗玉震、颜静、陈刚、张秋香、缪泽波等同事，长期的共事令我们建立了深厚的友谊。对后两位同事致以特别的感谢，他们对完成本书提供了大力的支持和帮助。

最后，感谢多年以来的好友岳大海、时吉斌、王峰，感谢我的中学老师邓英、杜爱芹、练鑫云、陈静，感谢我的研究生导师贾云得，以及所有在我成长过程中曾经给予我关心和帮助的朋友们！

罗剑锋

2010 年 6 月 7 日 于 北京 王府井

目录