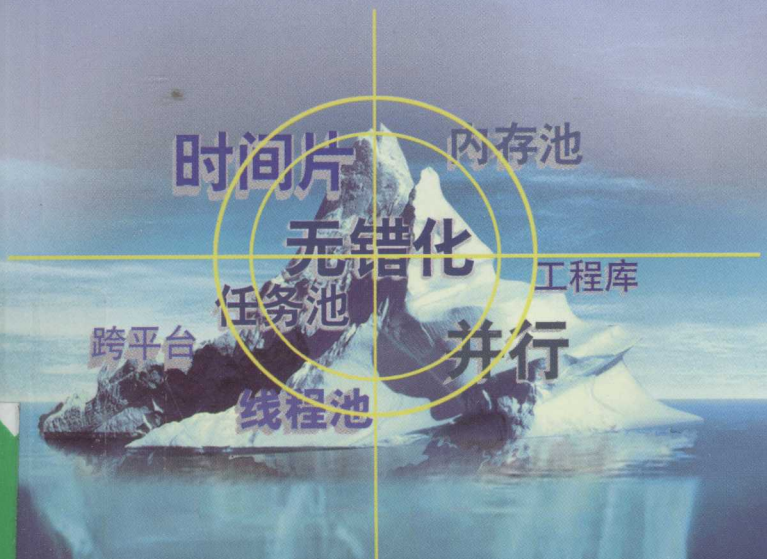
0 bug

C/C++商用工程之道

肖舸 著



- 多年的一线经验，无错化的开发之道
- 产品与工程视角，系统化的开发之道
- 商用与市场思维，务实化的开发之道

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

0 bug

——C/C++商用工程之道

肖 舸 著

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

内 容 简 介

本书主要针对C/C++语言在商用工程开发中的程序实战进行论述，从需求出发，从商用解决方案的角度来理解C和C++语言的程序设计技巧。首先讨论商用开发的原则，然后是基础知识、基本技巧，接着是无错化方法，最后提升到世界观层面论述并行开发的正确理念。商用程序员在实际工作中最为关注的无错化、并行、时间片、内存池、线程池、任务池、工程库和跨平台等相关问题，在本书中都有宝贵的经验总结和理念梳理。本书不是教科书，更多的是在开发技巧、测试调试、工程代码库等方面给出实例与总结。本书也可以说是教科书，作者试图通过实战技巧的训练，帮助读者升华出一种全新的程序设计理念，本书可以帮助你摆脱“Training”式编程开发思维与方法，培养“商用”和“产品”标准的工程开发技能。

本书适合作为C和C++的程序员进行“商用化开发”和“工程化开发”的参考。

图书在版编目(CIP)数据

0 bug：C/C++商用工程之道 / 肖舸著. —北京：电子工业出版社，2010.1
ISBN 978-7-121-09848-2

Ⅰ. 0… Ⅱ. 肖… Ⅲ. C语言—程序设计 Ⅳ. TP312

中国版本图书馆CIP数据核字(2009)第203358号

责任编辑：窦昊(douhao@phei.com.cn)
印　　刷：北京市天竺颖华印刷厂
装　　订：三河市鑫金马印装有限公司
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：36.5　字数：736千字
印　　次：2010年1月第1次印刷
印　　数：4000册　　定价：68.00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。

谨以此书，献给我的妻子孙清，你的温柔和善良，使我重新燃起了生活的信心！

此书也献给刚出生的笑笑，你是上天对我的补偿，我期待你纯真的笑容！

肖　舸

前 言

0.1 为什么要写本书

在本书定名的时候，笔者做了很多思考。本书究竟想说什么？关注的重点在哪里？看起来，本书所讲述的知识、经验和技巧，在很多书上都有讲，那么，我们的差别在哪里呢？

笔者看到过无数的年轻学子，兴高采烈地从学校出来，走向职场，但是，通常立即会遇到两个问题：

（1）他们虽然在学校中学到了很好的知识，但是到了企业中，却没有办法投入实用，甚至找不到工作。是他们学习的书不对？还是他们的老师没有教对？很多人陷入迷茫之中。

（2）另外，即使一个学子，顺利进入了企业，成为一名程序员，但无穷无尽的加班和做不完的项目任务，使生活充满了压力，也充满了苦闷。即使能赚取高薪，但生活毫无乐趣可言。这又是为什么呢？

笔者在IT研发领域工作了十几年，在这里有一点心得。

首先，笔者认为，学生即使学会了基本的程序开发技能，但还不能算作一个标准的商用程序员，其工作习惯、做事的思路和方法，特别是在具体程序工作中所秉持的设计思想，系统性思维，与企业需求相差甚远，这导致了就业上的困难，因此需要学习和调整。

其次，笔者认为程序员生活压力大，很多时候并不是任务量的压力，做过一些程序设计工作的朋友大概都有印象，“**程序好写，bug难追**”。真正导致我们大量加班的，往往不是程序的设计和书写过程，更多的，是debug。而企业中开发商用程序，由于对bug有着几乎为0的容忍度，这导致了商用程序员压力很大。

笔者一直在企业中做事，自己也深有体会，笔者曾经在2000年左右做过一个统计，发现工作中对bug的查找，占据了自己60%～80%的工作量。当时笔者就思考，如果能有一种方法，使程序一写出来就没有bug，那该是多么美妙的一件事情。

由于笔者一直从事C和C++语言方面的开发工作，于是笔者就着这个熟悉的领域，开始

了一点探索和研究工作，其间也参考了很多大师写的书籍。慢慢地，自己形成了一套程序书写原则和方式，笔者将其定名为“**C/C++无错化程序设计方法**”。经过实际工程试用，发现效果不错，很多程序出来之后 bug 很少，成熟度很高，得到了一些好评。

但是，随之发现了另外一个问题：商用工程，是为客户需求服务的，一段程序，如果不能满足客户需求，即使写得再正确也毫无意义；同时，一个系统的设计，如果脱离了需求分析，即使采用再精妙的算法，再优秀的设计，也是毫无意义的。

这使笔者不得不思考一个更深层次的问题，商用软件工程，其实已经不仅仅涵盖程序设计的领域，仅仅就程序谈程序，其实并无意义。一名商用程序员，不仅仅要是程序设计的专家，也必须是商务沟通的专家、客户需求理解的专家。这大大扩展了“程序员”这个职业的内涵和外延。

笔者发现目前市面上有很多关于程序设计的书籍，也有很多职业训练方面的书籍，但是，却从来没有人（也许是笔者孤陋寡闻），以程序开发的角度，讲述程序员进入企业后应该学习的商业开发思维和设计思想。

因此，笔者希望能根据自己的经验，写一本书，帮助大家快速掌握一些工程化的开发技巧，并和自己过去所学的相结合，快速成长为企业合用的人才。

0.2 本书包括哪些内容

本书主要针对 C/C++语言在商用工程开发中的程序实战进行论述。

本书无意重复无数书籍已经写过的一些 C 和 C++语言的基础知识，而是试图从另外一个角度，从需求出发，从商用解决方案的角度，去理解 C 和 C++语言的程序设计技巧。

因此，本书更多的以一种实用主义的态度，从需求出发去挑选需要的技术，并针对需求做出相应的优化，最终形成合用的商用开发方案。

本书可以说是一本教科书，因为里面有大量的经验和技巧，更有笔者多年积累的解决方案思路。但本书也可以说不是教科书，如果出于一种系统全面学习知识的角度来看本书，可能会有一点点失望，因为本书的知识已经打乱，完全在为需求服务。

本书是一本 C 和 C++语言的开发类书籍，里面讲述了大量开发的技巧，比如如何实现无错化的程序设计，使程序在写出来的时候就已经具有较高的鲁棒性，接近 0 bug 的地步。

本书还是一本兼顾调试、debug、测试的书籍。本书讲述了大型工程开发中一些基本的白盒测试技巧，也讲述了工程实战中性能测试的重要性和方法。本书可以作为一些大型商用工程的白盒测试参考。

本书也是一本实践类书籍，内附大量的工程代码，其中就包括笔者历经差不多 10 年时间总结出来的一套工程库，这套库代码已经在多个商用工程中获得检验，稳定可靠，并取得了可

观的经济效益。本书附有的所有源代码，均执行 BSD License，即大家可以免费使用，开源或者闭源发布产品，唯一的要求是，请大家保留笔者的原作者信息。

笔者近年来主要从事数据传输、分布式数据库以及服务器集群方面的研发工作，因此本书中举出的很多具体实例与这类应用开发有关，但笔者认为，这并不重要。贯穿于本书始终的商用化开发思维、以需求为导向的开发思路、实用主义的开发态度，才是最重要的。

笔者希望通过本书给大家传递一个信息：商用开发和学校内的程序设计，根本就是两个概念，大家可能使用同一种工具，同一个平台，但是，好比一个修理自行车的个体户老板，和汽车 4S 店的维护工人比较，二者有着本质的“度”的差别。提供的服务品质是完全不一样的。

从这个意义上说，不仅仅是 C 和 C++的程序员，其他领域的程序员，也可以看看本书，从自己的专业领域出发，理解一下商用工程开发的魅力所在。

0.3 商用工程开发和软件编程的区别

首先，我们要讨论一下何谓商用。

在学校中，大家是学生，第一任务是学习更多的先进知识，因此，求知是第一要务，对于一个问题，要求不厌其烦、精益求精，知其然，还要知其所以然。

而在企业中，大家是公司的一分子，要不断为企业赚取利润，同时赚取自己的薪水。因此，对于工作，一方面要能做到，另一方面还要以尽可能低的成本实现，才能赚取更多的利润。这其实已经说明了二者最大的差异性：**“严格的成本意识和质量意识”**，在商用程序员眼中，没有程序；只有产品。

既然是产品，就要严格地控制产品的质量，将 bug 率降到接近 0 的地步，还要严格地控制成本，商用程序员深刻理解自己工作的价值，能用一天完成的工作，绝对不用两天，万事从需求出发，满足需求即可，既不会不求甚解，也不会过犹不及，商用程序员善于把握平衡。

商用程序员，工作的目的是为产品质量负责，更是为产品的市场表现负责，但最终目的，还是为公司赚取的利润负责。因此，贯穿于商用程序员思维的核心，应该是最大限度地满足客户需求，创造企业价值，至于具体使用什么技术，技术是否很高深，是否很能显示自己的水平，其实商用程序员并不关心。

就笔者个人理解，这也是目前很多企业倡导的：**“研发工作，市场为主导，不要搞科研”**，的真实含义。

体现在实际的工作中，传统的程序员，往往以自我为中心，评判产品正确与否的标准，是自己的程序是否有 bug，会不会挂死，只要自己的程序 OK，就视为已经完成任务了。

而拥有商用程序开发思维的程序员，会更加关心工程项目的中心思想，核心的客户需求，随时随地判断自己所做的工作，对整个项目的实现是否起到正面的作用，一切的优化方向，是否贴合需求所需要的业务数据类型以及使用方向。

商用程序员不会仅仅关心自己的程序是否 OK，而更加关心整体系统的品质、性能能否满足客户需求，同时，自己的开发工作消耗了公司多少人力和时间成本，在同等品质的条件下有没有更快、更廉价的解决方案，等等。

最终，商用数据传输程序员的这些思考，将直接导致工程项目能否为公司赚取利润。说句俗一点的话：一般程序员，关心技术；商用程序员，关心成本，关心利润，关心“钱”。

0.4 商用程序员的核心思想

软件编程是一项技术性很强的工作。同时，软件开发发展这么多年，产业结构也逐渐细分，一个人的精力，不太可能面面俱到，精通所有的技术。

传统思维的程序员，更多的是站在技术的角度上思考问题，遇到问题，首先想到的是利用自己熟悉的技术来实现方案。

而商用程序员，更多的是站在客观的立场，理智地分析客户需求实现方案，需要用到哪些技术，或者说哪些技术更加合适，成本更低。

同时，商用程序员也不迷信最新的技术、最前沿的技术，事实上，很多时候，商用程序员偏保守，因为他知道尊重商用工程最为核心的需求：**稳定**！

举个例子，一个普通的 Windows 程序员，在接到一个 C/S 类型的工程项目后，往往会首先思考如何利用 Windows 搭建服务器和客户端，迅速实现。一个普通的 Linux 程序员，可能也是如此，仅仅是操作系统换成 Linux 而已。

而实际中，大家都知道，客户端开发大多数是 Windows 平台，因为这个操作系统市场占有率最高，而服务器平台最好使用 Linux，因为不花钱，可以有效降低公司的运营成本。

一个商用程序员，不管自己精通的是 Windows 开发还是 Linux 开发，首先就会针对上述市场上的具体情况做出分析，提出合适的解决方案。

另外，即使这个商用程序员是 C 或者 C++的高手，但在 Server 端的方案设计时，除了特殊的一些保证效率的需求，大多数需求会建议放到 Apache 上，利用 PHP 或者 JavaScript 之类的脚本语言完成。因为众所周知，C 和 C++语言是底层语言，其开发和测试成本远高于 PHP 之类的脚本语言。

0.5 本书适合哪些读者看

关于本书适合给哪些人看，其实前面已经有过一些描述，但这里还是具体细述一下：

- 很多软件专业的学生，初次进入职场，需要迅速掌握企业商业化开发的思路和技巧，建议看一看本书。
- C 和 C++的学习者和爱好者，建议看一看本书。可以掌握很多实际的技巧，并获得一个现成可用的工程库。
- 商业公司的程序员，建议看一看本书。可能您已经掌握了很多商用开发的思维和技巧，但也许本书能给你一点新的提示。
- 网络游戏公司的开发人员，建议看一看本书。本书的多任务工程库可能会对您很有帮助。
- 各种商用服务器的开发人员，建议看一看本书，本书中很多技巧，实际上是如何利用 C 实现 7x24 小时稳定性的服务器的技巧，很有帮助。
- 嵌入式的开发人员，建议看一看本书。本书中严格的代码规范和数据边界意识，对嵌入式之类资源较少，且有长期运行要求的设备开发，很有帮助。并且，如果使用开发服务器的技巧开发嵌入式产品，产品的稳定性会非常高。
- 其他语言的程序员，有可能的话，建议也看一看本书。本书中很多基本模块的实现，如内存池、线程池等，对 Java 等程序员理解自己平台的相同模块，很有帮助，并且，本书提出的商用开发思想是跨语言跨平台的，也很有参考价值。
- 各个公司的产品经理、项目经理、架构师，有可能的话，建议也看一看本书。本书中提出的很多商用系统工程的设计理念，是笔者多年开发的经验结晶，对于系统的设计、商用项目的风险管控，有很好的参考作用。

0.6 本书中一些名词的解释

API：Application Programming Interface，应用程序编程接口。作为现代商用工程最重要的协同开发和模块划分手段，API 几乎无处不在。商用程序员或多或少都接触过各种 API，如 Win32 API，socket API 等，甚至已经有程序员自己开始建立 API 意识，主动设置 API 来与其他模块接口。API 一般为 C 的函数定义形式，以及一些关键数据结构的定义。能被 C、C++、Java、PHP 等大多数语言所识别并使用。

NPI：Network Programming Interface，网络编程接口。这是笔者在工作中自己总结出来的一个概念，API 作为接口标准，受到各个模块工程师的尊重，但 API 毕竟是本机访问，大多数时候也限于 C 语言一种，在复杂的网络协同环境中不能满足需求，笔者提出 NPI 的概念，就

是在网络界面层，提出一种接口概念，不同平台，不同语言开发的模块，可以借由这个接口层互相联系、发生交互，进而完成业务。NPI 除了包括 API 所有信息外，也包括网络信令，IP 地址及端口描述等信息。

Loading：这是一个舶来词了，笔者也是向台湾的同事学习到的。原意指网络服务器的带宽占用，由于带宽通常需要向运营商花钱购买，因此，这个词就有了成本的含义。一个系统使用的 Loading 越高，运营成本就越高。不过推而广之，大家后来渐渐习惯用这个词代指一切需要花钱购买的资源，如 CPU Loading 过高，就是需要占用很大的 CPU 计算资源。内存 loading 过高，就是内存占用太多，等等。本书会常常用到这个词汇。

Log：商用工程，由于一般都有 7×24 小时长期运行的需求，因此一般都有自己的日志系统，用以记录运行期间的重大事件、关键报文的流转、一些可能导致崩溃的重大错误等，用以事后分析。这类日志系统，一般也称为 Log 系统。

Server/Client：网络通信中的服务器和客户端角色。通常说来，客户端是网络动作的发起者，服务器是被动的接收和执行者。但请注意，网络情况千变万化，角色经常会发生变化，一个 Server，可能是另外一个 Server 的 Client。

UI：User Interface，用户界面。相关的还有 GUI，图形用户界面；CUI，文本控制台用户界面（就是类似 DOS 和 Linux 的文本界面，依靠用户输入文字执行命令。Windows 下也有文本控制台窗口）。

目 录