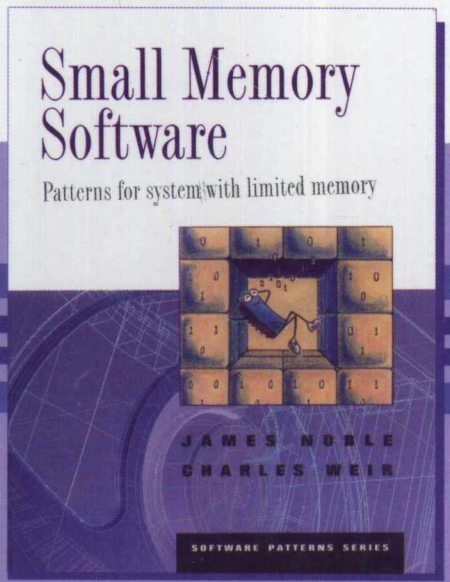
内存受限系统之软件开发

— 针对内存受限系统而整理的模式 —

Small Memory Software Patterns For System With Limited Memory



James Noble & Charles Weir 著
Duane Bibby 插图
侯捷 王飞 罗伟 译

华中科技大学出版社

内存受限系统之软件开发

— 针对内存受限系统而整理的模式 —

Small Memory Software
Patterns for system with limited memory

James Noble and Charles Weir 著

Duane Bibby 插图

侯捷　王飞　罗伟 译

图书在版编目(CIP)数据

内存受限系统之软件开发/(美)James Noble & Charles Weir 著；侯 捷 王 飞 罗 伟 译
武汉：华中科技大学出版社，2003 年 1 月
ISBN 7-5609-2894-3

Ⅰ.S…
Ⅱ.①J… ②C… ③侯… ④王… ⑤罗…
Ⅲ.计算机软件-开发
Ⅳ.TP311.1

责任编辑：周 筠(http://yeka.xilubbs.com)
技术编辑：孟 岩
出版发行：华中科技大学出版社 （武昌喻家山 邮编：430074）
录 排：华中科技大学惠友科技文印中心
印 刷：湖北新华印务有限公司
开 本：787×1092 1/16 印 张：22.5 字 数：400 000
版 次：2003 年 1 月第 1 版 印 次：2003 年 1 月第 1 次印刷
印 数：1—6 000 定 价：58.00 元
ISBN 7-5609-2894-3/TP・495

商标说明

侯捷译序

内存曾经是兵家必争之地，曾经被喻为CPU之外最宝贵的电脑硬件资源。在那“640K天堑”的远古年代里，程序员对内存锱铢必较的程度，必然令生活于“虚内存”环境下的当今世代瞠目结舌。当时的人们（我也属其中之一）即便在config.sys中挥汗调校节省区区数十个bytes，都觉得欢欣鼓舞。

而后，PC操作系统的技术有了突破性的发展，逐渐地运用int67h进入EMS内存，运用int21h进入XMS内存，最后是全面地、隐藏式地、通透性地提供了虚内存（virtual memory）。当虚内存操作系统（如Windows、OS/2、Linux）走进群众，苦乐俱往矣。非人道的痛苦折磨被迅速遗忘，缁铢必较的轶趣成了白头宫女话天宝当年的回忆。我们不再被代码大小所限，也不再被数据量所限。所有内存不足的问题只要“加一条256M内存”就获得解决。程序员“从此过着幸福美满的日子”。

从极度严苛到极度自由，PC程序员在内存用量上得到了完全的解放，开始胡天胡地了起来，不再把内存用量看在眼里，想在心里。当时的我也总这么说：“能以小钱解决的问题，都不是问题”。这原本是好的发展，快乐的走向，但是当嵌入式系统逐渐大行其道，内存问题再度浮上台面，而且较之DOS的“640K天堑”远远更为严苛。

如此严峻的环境下，还是有许多成功的嵌入式系统问世。它们是怎么办到的呢？本书收纳多个嵌入式系统成功案例，从中萃取整理出内存受限环境下的软件生存之道。本书以patterns的严谨描述方式，给予每一个方案以标准名称、别名、问题出处、解决办法、结果与影响、实作摘要、示例、旁征参考。

这是一份良好的整理与总结，提供了一个完整涵括，适合给预备或即将在嵌入式系统上冲锋陷阵的程序员阅读。读者必须具备相当的技术基础，因为本书告诉你的是“如果你要解决某某问题，请使用某某技术，它会带来某某成果，并造成某某（正面或负面的）影响”。虽然本书也带有数量颇丰的示例程序，但它们都是具体而微——能够从这些小程序的蕴涵看出大世界的应用，你得具备我所谓的“相当程度的技术基础和实务经验”。

Small Memory Software

本书极端侧重 **patterns**。这个词在台湾、大陆两地的译法，我见过三种：范式、样式、模式。我个人最喜欢“范式”，足以说明 patterns 的“典范”意味。但顾及大陆术语习惯，本书以“模式”称 patterns。本书所有 patterns 都保留英文名称，并以特殊字型标示，例如 Small Architecture, Memory Limit, Small Interfaces....。本书也提到所谓 **forces**（作用力，意义详见附录），起始数章以中英并陈方式显现，例如 *Memory requirements*（内存需求），*Real-time response*（实时响应），*Start-up time*（启动时间），*Power consumption*（电力消耗），*Programmer effort*（编程心力），*Programmer discipline*（编程素养）... 后继篇幅才以中文独现。所有 patterns 和 forces，在封面内页和封底内页，有完整的列表和简要说明。

本书具有较为浓厚的学术味道，读者定位也比较高阶。考虑读者的层次和需求，我时而将一些可能引起疑虑的术语以中英并陈的方式呈现，这样或可弥补中文术语不统一造成的缺憾，以及作为中文难以表达英文原意时的小小补偿。书中有少部分技术超越我的领略能力，对此我的处理方式是：保留原文（1~2 个小句子）由读者自行品味。

此外，一般谈及内存“释放”（'free' 或 'deallocate' 或 'release'），有时是指客端将内存还给 pool，有时是指 pool 管理器将内存还给系统（system heap），需要读者根据上下文体会。一般而言我把 'free' 译为“释放”，'deallocate' 译为“归还”，'release' 译为“释出、释还、释回”。但这只是通则，实际译法视上下文调整。

本书由我和王飞、罗伟两位先生共同完成。没有他们的协助，这本书不可能在这个时间点上和各位见面。我要在此表达我的谢意。在合译态度上，我个人一向不喜欢一刀切的二分法或三分法，本书由王飞负责第三章（含）前初稿，罗伟负责第四章（含）之后初稿，初稿以外的一切由我总揽。本书有繁简两个版本，繁体版采用台湾术语，简体版采用大陆术语。两版勘误皆可于以下网站获得。

侯捷 2002.11.30 台湾新竹
jjhou@jjhou.com
http://www.jjhou.com（繁体） http://jjhou.csdn.net（简体）

Small Memory Software

王飞译序

却纳须弥于芥子

计算机技术发展一日千里，如今的家用电脑堪与从前的商用小型机媲美。内存容量更是呈跳跃式扩大，于是今日编写程序似乎再也不必考虑内存了，当年先辈为了节约内存而发展出的种种巧思、妙技，似乎已成明日黄花。

但是，嵌入式系统的蓬勃发展却使人惊叹“小容量内存技术”复活了！

嵌入式设备受体积和电源供电能力等因素影响，内存容量很小。如果沿用 PC 上“挥霍无度”的编程方式，嵌入式系统一定会因为内存耗尽而寿终正寝。科技的发展似乎总难以满足人类的需求和欲望。为了将须弥山之大的各种功能纳入芥子般纤小的内存中，尘封已久的小容量内存技术重获人们的青睐。我想，这正是本书的缘起。

本书两位作者向大家展示的是节约内存的 patterns（模式），是从“道”的高度来分析与解决问题。我们不应指望从书中拷贝几段代码，就能解决现实生活中的问题。诚如“独孤九剑”讲的是剑法而非剑招，需要大家不断地在实践中思考与揣摩，应势而发，见招破招。

本书的 patterns（模式）提炼自许多成功的系统，两位作者对这些纷繁复杂的 patterns 进行了梳理和分类，辅以图解和示例程序，文笔轻松诙谐，因此本书既实用又具亲和力。对于从事嵌入式系统的朋友以及对小容量内存技术感兴趣的朋友来说，本书精彩不容错过。欲纳须弥于芥子，请读这本书！

Small Memory Software

这是我参与翻译的第一本书，分量很重。我的工作范围是第三章之前的所有内容。我要对侯捷老师和周筠老师给予我的宽容和关爱表示感谢，谢谢您们物质和精神上的帮助，请原谅我的无知和孟浪。我要对协译者罗伟先生表示诚挚的谢意，很高兴认识您这位远方的朋友。我同样要感谢本书的作者 James Noble，感谢您对我所提的问题所做的详尽解答。我还要感谢我所在部门的赵治本、张志东、姜巍、刘波、张国华等各位领导和同事，感谢大家在生活和工作中给予我的关怀和帮助。我更要感谢我的母亲和哥哥，感谢你们给予我的始终不渝的爱。

王飞
2002.12.01 于辽宁沈阳

罗伟译序

作为建筑学上第一个具体而直接的体系，古希腊时代产生的“柱式”，历经两千多年延伸至今。以致很长一段时期里，欧洲的建筑艺术教育就以研究“柱式”为主要内容。它之所以能有如此强大的生命力，首先因为它在实用基础上已经上升为一套非常严谨、十分成熟的规范。在大工业化时代中，你也许无法得到像米开朗基罗式的教皇级待遇，身披创造者的荣耀，但与你共事的同僚，或你自己，也（因为用了标准柱式而）不至于尝试不入流的技巧，导致业主的围攻。这倒也算得上另一种幸福。

计算机界常提到的 patterns（模式），与上面所说建筑学上的柱式有异曲同工的意味。但只要稍加了解实际使用的 patterns，你或许会发现，这些 patterns 之于编程，与柱式之于建筑，层面上着实有好大不同。初学 patterns 的人，也许在 Abstract Factory pattern 的行进过程中，就四肢无力摇旗投降了，这大抵是因为你还缺少实际程序的历练。毕竟 patterns 不是你拿着 Gang of Four（GOF）那本圣经（《*Design Patterns*》）就可以横扫江湖的。GOF 只是一本指南！唯有实务锻炼和经验累积，才能使你对一个现实系统作出明智的解析。相比之下，柱式的各种比例测算、花式运用，散见于各类建筑设计手册，依葫芦画瓢对受过常年艰苦训练的你来说，并不算一件天大的难事。

最早在网络上看到这本《*Small Memory Software*》，对它有很高的期待，不仅因为这本书是与嵌入式系统相关的第一本 patterns 总结数据的书，更因为当时我正在思考面向对象思想于嵌入式器件里的使用价值，很希望得到一些实务启发。整本书研读下来的结果并没有让我失望（当然，也没有解决我的所有问题），它给我的思考指出了一些道路：不仅某种具体技术有其使用局限，面向对象思想也有其实施上的局限性。这对正处于上升阶段之嵌入式系统与传统辉煌之面向对象思想的结合，有一定指导意义。毕竟，在总体设计阶段把握住整个项目，决定最适合当前要求的解决方案，是绝对优先于 classes 设计与编码的。

对嵌入式系统来说，系统分析人员有一定的特殊性，因为需求设计基本完成后，要进行软件与硬件的划分，而不是径自奔向软件总体分析。这是一个需要很高经验和技巧的步骤。常用的硬件处理器不下 200 种，软件系统平台也有 Vxwork, PSOS, VRTX, Nucleus, WinCE, QNX, CMX 等各种选择。长期于 PC 和 workstation 上修炼的程序员一旦接触嵌入式项目，一定会意识到，原本的世界真是个 big big world ☺。

本书有三大亮点：

一是对各种 patterns 的论述非常清晰，尤其在“环境讨论”和“结果”这两项，让程序员得以高屋建瓴地审视应用环境；

其次，本书实作和范例使用了多种语言，避免过分依赖语言特性。此外，作者网页上的 C++ 和 Java 范例代码，对理解书中讨论内容有莫大作用，强烈建议你亲自操练一遍；

最后，每一种 pattern 论述完毕后，通常会提供数种参考文献。本书参考文献达 210 种之多。这对计算机科班毕业或常年从事 PC/workstation 编程的程序员有特别意义，因为这些文献充分显示了作者的专业成长历程。当然，对电子背景的工程师而言，软硬件结合部分的参考文献稍嫌不足，不过要找到这种类型的书籍应当不难。

得到这本书的翻译机会纯属偶然。也正是这次偶然之后的互动过程，让我对侯捷先生更为尊敬。感谢华中科技大学出版社的周筠女士一力促成我和侯先生的合作，祝你健康。我还要表达对本书的另一位译者王飞的敬意，很高兴能和千里之外的你成为封面上的邻居。特别地，我的朋友孟岩对我的 OO 之路起到重要作用，谢谢你十多年来的友谊。以我父母为首的家人是我永远的力量来源，你们的支持使我能振作起因牙痛而昏沉的头脑，按时完成了译稿。

恭喜你的书架上多了一本出色的技术书籍，愿它带给你快乐...Mmm...与财富！

罗伟

2002.11.30 于湖北武汉

Small Memory Software

序言

by John Vlissides

宣称“个人电脑（Personal Computers, PCs）是唯一拥有庞大市场的计算平台”显然很愚蠢。Forrester 调查显示，1999 年 PCs 的销售量已达巅峰，而 non-PC 计算设备的消费量却如烟火般地直向上窜，而且成长趋势看来还会持续。宣称“内存不再是编写程序时的主要限制”同样愚蠢。目前流行的观点是：如果内存是问题，多买一点不就得了！但是你要知道，所有桌面应用程序的背后，甚至是最节省内存的应用程序背后，隐藏着一头“软件基础设施”（software infrastructure）巨兽，它对内存简直贪得无厌。数百个 megabytes — 不久前用来描述硬盘和磁盘容量的数字 — 现在不过是价格上的入门选择。称得上大的内存一律按 gigabytes 计算。天哪，这些 RAM 里面究竟装了些什么？

某些浪费有好的理由。通过代码重复运用，虽然浪费一点空间，却可能缩短程序员的开发周期；编译器实作技术上的折衷妥协，有可能造成二进制码（binary code）的累赘；为了让程序反应速度更快，不得不以空间换取时间。内存恣意运用也可能反映出真正的内存密集型（memory-intensive）问题。也许，呵，也许以上兼而有之。

然而更常见的是，大量内存需求肇因于马虎的编程、马虎的设计、马虎的开发方法和马虎的组合。这类因素完全可以避免，悲剧在于它们往往不被认为值得避免，至少短期内如此。市场的现实和“较差即较好”症候群鼓吹大家先把产品推出门，“日后”再修改。让人沮丧的是，内存问题都太容易解决了（多买点不就好了），“日后”遂永不来临。内存消费是典型的“最后处理”问题——如果它确实被处理了的话。

但令人惊奇的事情近来不断发生：嵌入式系统和手持设备蓬勃发展，在 IBM 我们称其为“散布式计算”（pervasive computing）。受到当前粗糙的电池技术影响，这些系统的计算能力有限；受到尺寸或成本的影响，这些系统的 RAM 容量有限，迫使程序员将内存消耗量当作发展条件重新检视。终于，人们发现，内存效能绝不是系统蹒跚之后才需着手解决的问题。它必须一开始就加以设计。最困难之处在于指出怎么干。

Small Memory Software

经过这么久之后，程序员们不得不重拾内存节约技巧，尽管这些技巧自从 von Neumann（冯诺曼）架构就有了。确实，“节约内存”是计算界最初 25 年玩的游戏，直到虚内存普遍被运用，这一局面才发生改变。尽管如此，唯有配备特别少量内存的计算设备，才能适应残酷的小型化运动（miniaturization）。今昔差别在于，如今这类设备不论在数量或多样性方面，都迅速超越了其他种类的计算机。

这正是本书耀眼之处。James 和 Charles 针对在这类设备上讨生活的程序员所撰写的这本书，如果算不上救生圈（lifesaver），起码也是节约时间的法宝（timesaver）。作者以简捷有力的方式把握了计算机领域中大多数古老而不可或缺，但非显而易见的内存节约技术。这些技术从前也有人描述过，但从未像本书这样将一系列模式（patterns）梳理成有机体，完全实用，技术新颖，而且立竿见影。不论我如何称赞本书对软件发展产生多么巨大的潜在影响，都不为过——不论你的软件运行于 PDA, cell phone, pager, 或闻所未闻的嵌入式系统。每一位小容量内存程序撰写者都可以从本书得到收获。即使你认为你有足够的内存可供挥霍，这些模式（patterns）也会帮你树立起一个健康的内存运用观念，让你的程序不那么饕餮。

你可以逐页遍读本书，但这样终究只是“读书”。真正的利益在于遭遇实际问题时，找出相关模式（patterns）并用它来解决问题。因此，我建议你在设计和实作程序时研究并实际运用书中的模式。一旦你这么做，你会发现你必须修改模式使之适应具体问题。优秀的模式将使一切成为可能，但并不是直接告诉你答案——永远不可能那么做，因为没有两个问题完全相同——而是授你以知识。这就是书中“forces”段落弥足珍贵的原因：它们揭示问题解决过程中可能遇到的限制条件，把解决方案域空间切割为小得多的可能性集合（但这集合可能还是很大）。其中究竟何者最适合你，部分依赖于你愿意并能够做出哪些必要取舍，以及你无意中可能遭遇的限制，其余就有赖于你的创造力了。当然啦，这样一本书能够帮助你更具创造力，但创造力无法来自书中；创造力取决于你。

可以这么说，如果你从事内存受限系统上的编程工作，而且你开始运用本书，很快你就会发现它实在是一份天赐的礼物。

John Vlissides
IBM, *T.J. Watson Research*

前言

by James & Charles

从前，计算机内存是地球上最昂贵的物品之一。当时大量的人类智慧耗费在超新星爆炸模拟上，全部的投入不过换得一个后来的诺贝尔奖得主和一大堆真空管。可现在许多人的移动电话、数字记事本或微波炉里的内存，就足以模拟多数星系的毁灭。

但至少有两件事在计算机历史中恒长不变。软件设计依旧艰难（Gamma et al. 1995），软件功能不断拓展以至填满所有可用内存（Potter 1948）。本书同时涉及这两个问题：模式（patterns）已被证实是记录软件设计知识的有效形式，而本书所论的模式专门用来解决内存需求方面的问题。

身为本书作者，我们的写作还有其他几个目的。以模式研究人员和作家的身份，我们希望多学一些模式和模式编写技巧；以软件设计者（designers）和体系结构师（architects）的立场，我们希望研究目前存在的系统并向它们学习。更明确地说：

- 我们希望获得并分享关于“可移植之小容量内存技术”的全面而深入的知识；这些技术在许多不同环境中都能发挥作用。
- 我们希望写出一组完整的模式集，目标单一而集中，就是解决内存需求问题。
- 我们希望研究这些模式之间的关系，并根据这些关系将它们群组化、次序化。
- 我们想要一本亲和力高的书，它让你感觉轻松有趣，而不是让你活受罪。

本书就是我们的成果。无论你在你的工作中是否遭遇内存的束缚，它都是像我们这样的软件开发者和体系结构师的好帮手。

为使本书亲和力更强（也为写作带来乐趣），我们以轻松的态度完成大部分模式示例，并搭配 Duane Bibby 的漫画。如果你对这些“轻浮的”☺东西不感兴趣，请跳过漫画和示例描述——是的，其他内容我们尽可能保持严肃。

本书仍在发展之中。我们已经把许多人的建议和评论纳入，欢迎更多指教。您可以在我们的网站上与我们联系，网址是 http://www.smallmemory.com/

Small Memory Software

致谢

没有任何作者可以独享一本书的功劳。首先要感谢 John Vlissides，永不倦怠的编辑：我们手头上还保留他对这份疯狂作品的许多提议。从最初的 EuroPLoP 论文到最终的草稿，他对我们所整理的模式（patterns）作了大量评论，我们深为感激。其次，如果没有 Duane Bibby 的插图，本书风貌将大不相同，希望大家和我们一样喜爱这些插图。

我们要为本书的缺点负责，荣耀则应归功于花时间阅读和评论模式草案及本书草稿的各位 Memory Preservation Society 成员和特约研究员。这些人包括（以下可能有所疏漏）：John Vlissides（再次感谢），Paul Dyson, Linda Rising, Klaus Marquardt, Liping Zhao （他们是 EuroPLoP 和 KoalaPLoP 研讨会上的指导员），Tim Bell, Jim Coplien, Frank Buschmann, Alan Dearle, Martine Devos, Martin Fowler, Nick Grattan, Neil Harrison, Benedict Heal, David Holmes, Ian Horrocks, Nick Healy, Dave Mery, Matt Millar, Alistair Moffat, Eliot Moss, Alan O'allaghan, Will Ramsey, Michael Richmond, Hans Rohnert, Andreas Rüping, Peter Sommerlad, Laurence Vanhelsuwe, Malcolm Weir, 以及设于 University of Illinois 和 Urbana-Champaign 的 Software Architecture Group，成员包括：Federico Balaguer, John Brant, Alan Carrol, Ian Chai, Diego Fernandez, Brian Foote, Alejandra Garrido, John Han, Peter Hatch, Ralph Johnson, Apu Kapadia, Aaron Klish, An Le, Dragos-Anton Manolescu, Brian Marick, Reza Razavi, Don Roberts, Paul Rubel, Les Tyrrell, Roger Whitney, Weerasak Witthawaskul, Joseph W. Yoder, Bosko Zivaljevic。

英国的 Addison-Wesley（或 Pearson Education，我们忘了究竟是哪个）工作团队在地球的另一端帮了两位作者大忙。我们感谢 Sally Mortimore （他启动本案），Allison Birtwell（他结束本案），Katherin Ekstrom（他打破距离）。本书荣耀也要归功于两位艺术家：George Platts 构思书中插图，Trevor Coard 亲手绘制了多幅插图。

最后，我们一定要感谢模式社群（patterns community）的全体成员，尤其要感谢曾经参加 Kloster Irsee 召开的 EuroPLoP 会议的诸位成员。我们两人都是“模式文学”的仰慕者（就像人们可能是科幻小说迷一样），我们希望这部模式选粹能为模式文学增辉。

Small Memory Software

目录

0

导读

Introduction

“小巧即是美” E. F. SCHUMACHER
“你永远不可能既富有又苗条” BARBARA HUTTON

设计“在有限内存空间内高效执行”的小型软件，是一门行将就木的艺术，直到最近这种情况才有所改变。过去由于个人电脑、工作站、大型主机的内存容量和处理器速度呈指数规模递增，程序员几乎可以完全不必顾虑内存限制了。

身处新世纪之交，我们发现了一个伸手可及的庞大市场，数以亿计的移动设备需要大量高规格软件；但受实际尺寸和能量（电力）供应限制，使它们的内存容量相对有限。与此同时，web 服务器和数据库服务器的开发人员发现，他们的应用程序必须高效运用内存，同时支持数十万个他们赖以生财的客户。甚至 PC 和工作站上的程序员也发现，视频需求和多媒体需求向系统内存容量提出了超乎合理的挑战。小容量内存软件（small memory software）复活了！

何谓“小容量内存软件”？内存的大小就像财富或美貌一样，并不是绝对的。某个数量的内存到底是大是小，取决于软件被满足的程度，也取决于底层软硬件体系结构和其他许多东西。大型计算机上的气象软件也许和移动电话中的文字处理器或智能卡（smart card）里头的嵌入式应用软件一样受到内存限制的困扰。所以：

“小容量内存软件”就是“内存欲求不满”的软件！

本书专为小容量内存软件的程序员、设计者和系统体系结构者而写。也许你正着手设计并实现新系统，或维护现行系统，或只是为了拓展软件设计知识，那么都可以成为本书读者。

本书描述了我们在成功的小容量内存系统中见到的最重要的编程技术。我们分析这些技术，把它们整理为模式（patterns）——一种“可有效运作之事物”的特别描述形式（Alexander