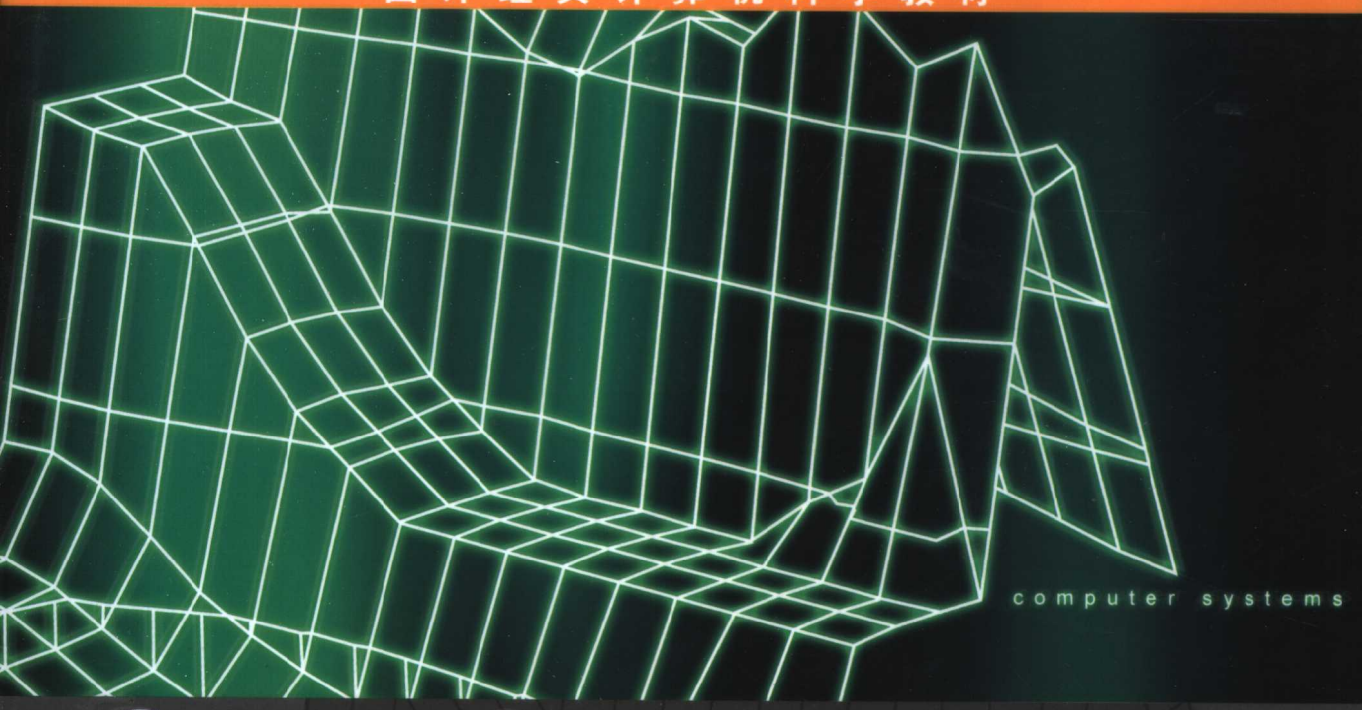
国外经典计算机科学教材



Computer Systems
A Programmer's Perspective

深入理解计算机系统

[美] Randal E. Bryant
David O'Hallaron 著
龚奕利 雷迎春 译

Randal E. Bryant and David O'Hallaron

PEARSON Prentice Hall
Pearson Education
中国电力出版社
www.infopower.com.cn

国外经典计算机科学教材

Computer Systems
A Programmer's Perspective

深入理解计算机系统

[美] Randal E. Bryant
David O'Hallaron 著
龚奕利 雷迎春 译

中国电力出版社
www.infopower.com.cn

Computer Systems: A Programmer's Perspective (ISBN 0-13-034074-X)
Randal E. Bryant, David R. O'Hallaron

北京市版权局著作权合同登记号 图字：01-2003-1020 号

图书在版编目（CIP）数据
深入理解计算机系统 /（美）布赖恩特（Bryant,R.E.）等著；龚奕利，雷迎春译. —北京：中国电力出版社，2004
（国外经典计算机科学教材系列）
ISBN 7-5083-2175-8
Ⅰ.深... Ⅱ.①布...②龚...③雷... Ⅲ.计算机系统－高等学校－教材 Ⅳ.TP30
中国版本图书馆 CIP 数据核字（2004）第 014328 号

丛 书 名：国外经典计算机科学教材系列
书　　名：深入理解计算机系统
编　　著：（美）Randal E. Bryant, David R. O'Hallaron
翻　　译：龚奕利　雷迎春
责任编辑：姚贵胜
出版发行：中国电力出版社
　　地址：北京市三里河路6号　　邮政编码：100044
　　电话：（010）88515918　　传　　真：（010）88518169
印　　刷：汇鑫印务有限公司
开　　本：787×1092　1/16　　印　　张：53.25　　字　　数：1209千字
书　　号：ISBN 7-5083-2175-8
版　　次：2004 年 5 月北京第 1 版　　2004 年 5 月第 1 次印刷
定　　价：85.00 元

出 版 说 明

新世纪的朝阳刚刚露出丝抹微红，如火如荼的全球信息化浪潮便汹涌而至，让人无时无刻不感受到新一轮产业革命的气息。如何在这场变革中占尽先机，既是对民族信息业的挑战，也是机遇。从而，作为民族信息产业发展基石的高等教育事业就被赋予了比以往更重的责任，对培养和造就我国 21 世纪的一代新人提出了更高的要求。但在计算机科学突飞猛进的同时，专业教材的发展却严重滞后，越来越成为人才培养的瓶颈。同时，以美国为代表的西方国家计算机科学教育经历了充分的发展，产生了一批有着巨大影响力的经典教材，因此，以批判、借鉴的态度有选择地引进这些国外经典计算机教材，将促进国内教学体系和国外接轨，大大推动我国计算机教育事业的发展。

中国电力出版社进入计算机图书市场已有近 6 个年头，通过坚持“高端、精品、经典”战略，致力于与国外著名出版机构合作，出版了大批博得计算机业界和教育界赞誉的作品。通过与信息技术教育界人士的广泛沟通，同时依托丰富的出版资源，中国电力出版社适时推出了“国外经典计算机科学教材”的出版计划。本次教材出版计划是和美国最大的计算机教育出版机构——Pearson 教育集团（Addison-Wesley、Prentice-Hall 等皆为其下属子公司）合作，依托其数十年积累的大批经典教材资源，确保了教材选题的权威经典。

为保证这套教材的含金量，并做到有的放矢，我们在国内组织了由中国科学院、北京大学等一流院校教师组成的专家指导委员会，对高校课程教学体系做了系统、详细的调查，听取了众多教育专家、行业专家的意见，对教育部的教育规划进行了认真研究，并深入了解国外大学实际教学选用的教材状况，对国外教材做了理性的分析，确立了依托国家教育计划、传播先进教学理念、为培养符合社会需要的高素质创新型人才服务，来作为本次“国外经典计算机科学教材”出版计划的宗旨。

我们从 2002 年的下半年开始着手这套教材的策划工作，并多次组织了专家研讨会、座谈会等，分析现有教材的优点与不足，采其精华，并力争体现本套教材的质量和特色。

1. 深入理解国内的教学体系结构，并比照国外相同专业的课程设置，既具有现实的适用性，又立足发展眼光，具备一定的前瞻性。

2. 以计算机专业的核心课程为基础，同时配合专业教学计划，争取覆盖专业选修课程和专业任选课程。

3. 选取国外的最新教材版本，同时对照国内同专业课程的学时要求，对不适用的版本进行剔除，充分满足国内教学要求。

4. 根据专业对口和必须具备同课程教学经验的要求，严格挑选译者，并严把质量关，确保教材翻译的高质量。

5. 通过从原出版社网站下载勘误表及与原书作者进行沟通的方式，对原书中的错误一一做了修改。

6. 对教材出版的后期工作，如审校、编辑、排版、印刷进行了严格的质量把关。

经过专家指导委员会的集体讨论，并广泛听取广大高等院校师生的意见，反复比较，从数百种国外教材中遴选出数十种，列入第一阶段的出版计划。这些教材的作者无一不是学富五车的大师，如 Stallings，Date，Ullman，Aho，Bryant，Sedgewick 等，他们的作品均是一版再版，并被众多国外一流大学如 Stanford University，MIT，UC Bekerley，Carnegie Mellon Univeristy，University of Michigan 等采用为教材。拟订的第一阶段出版计划包括 30 种图书，内容覆盖程序设计、数据结构、操作系统、计算机体系结构、数据库、编译原理、软件工程、图形学、通信与网络、离散数学等计算机专业核心基础课程，基本满足国内计算机专业的教学要求。

此外，为了帮助广大任课教师加深对本系列教材的理解，减轻他们的备课难度，我们从国外出版机构引进了大批的课程教学辅助资料，并积极延请国内优秀教师，根据其使用该系列教材中的教学经验，着手编写更加适合国内应用状况的教辅材料。

由于我们对国内高校计算机教育存在认识深度上的不足，在选题、翻译、编辑加工出版等方面的工作中还有许多有待提高之处，恳请广大师生和读者提出批评和建议，并期待有更多的人加入到我们的工作中来。我们的联系方式是：

电子邮件：csbook@cepp.com.cn

联系电话：010-88515918-300

联系地址：北京市西城区三里河路 6 号中国电力出版社

邮政编码：100044

译 序

作为一个程序员，我们经常被一些奇怪的程序问题所困扰。例如最近，我的一位朋友从经典的数据结构书上抄了一段关于计算有向图的函数实现代码，这个函数实现在 gcc 环境下的编译一点问题都没有，但只要一实际运行，就会报段错误。我看了这段代码后，立刻就意识到问题出现在哪里了——他在函数实现里分配了一个 16MB 的局部变量导致栈溢出。类似这样的问题，还会有许多。不过在这些问题中，让程序员有麻烦的已经不是编程语言本身的问题，而是需要程序员更好地理解计算机系统，知道程序如何在计算机上被执行。理解计算机系统，不是简单地从书市上购买一些介绍计算机系统的书，读一读而已。迄今为止，我对市面上这类书的了解是：对于大多数程序员而言它们都过于专业化，且从书的内容和语言组织上都偏重于原理的介绍，一般程序员很难有时间和精力去消化和吸收书中的内容，更无从用这些计算机系统的知识来帮助自己解决程序问题。

事实上，高级语言编程和计算机系统被编程环境如 gcc 划分成两张皮，尽管程序员能用高级语言驱动计算机完成指定的计算任务，可是却不一定能很清楚地知道计算机是如何解释和执行程序代码的。

我本人是一个计算机专业科班出身的人，学生期间学习到许多关于计算机系统的知识。可在实际研究工作中，过去所学的计算机系统知识变得遥远和模糊。1999 年初，出于研究兴趣的目的，我设计了一个高性能网络服务器结构，并编写了它的实现。在此期间，使我明白一个高性能服务器程序与计算机系统之间的唇齿相依的关系。过去，促使我建立高级语言和计算机系统的联系来主要自于研究的压力，其采用的方法是遍寻国外关于系统编程的邮件列表，并结合以往所学的计算机系统知识。这种方法固然能帮助我解决实际所碰到的问题，但却需要花费大量的时间且没有条理。2003 年元月，编辑部让我帮助他们从一批刚出版的外文书中挑一些可以在国内推广的书，我一眼就看中了这本由 Bryant 和 O'Hallaron 所著的《Computer Systems: A Programmer's Perspective》，它就是我过去想要的书，我相信也是每一个想了解计算机系统的程序员想要的书。我迫不及待地从编辑手中抢下此书的翻译工作，这个临时添加的任务改变了我和另一位译者 2003 年的生活。2003 年 8 月底，终于完成此书的翻译工作，并起中文名为《深入理解计算机系统》。

《深入理解计算机系统》的最大优点是为程序员描述计算机系统的实现细节，帮助其在大脑中构造一个层次型的计算机系统，从最底层的数据在内存中的表示（如大多数程序员一直陌生或疑惑的浮点数表示），到流水线指令的构成，到虚拟存储器，到编译系统，到动态加载库，到最后的用户态应用。贯串本书的一条主线是使程序员在设计程序时，能充分意识到计算机系统的重要性，建立起被所写程序可能被执行的数据或指令流图，明白当程序被执行时，到底发生了什么事。从而能设计出一个高效、可移植、健壮的程序，并能够更快地对程序排错、调整程序性能等。

本书是通过程序员的视角来介绍计算机系统，即首先把高级语言转换成计算机所能理解的一种中间格式（如汇编语言），然后描述计算机如何解释和执行这些中间格式的程序，是系统的哪一部分影响程序的执行效率。所以，在讲述计算机系统知识的同时，也顺便给出了关于 C 语言和汇编语言

（有可能是编译系统产生的）的编程和阅读技巧，以及基本的系统编程技巧和工具，同时，还给出一些方法帮助程序员基于对计算机系统的理解来度量和改善程序的性能、及其它棘手问题。

本书的主要内容是关于计算机体系结构（高级硬件设计）与编译器和操作系统的交互，包括：数据表示；汇编语言和汇编级计算机体系结构；处理器设计；程序的性能度量和优化；程序的加载器、链接器和编译器；包括 I/O 和设备的存储器层次结构；虚拟存储器；外部存储管理；中断、信号和进程控制。对这些不同领域知识的介绍使我们能在编写系统程序时，基于系统性能的考虑，采取一个更好的折中方案。

本书强调对计算机系统的概念的理解，但并不意味着不动手。如果按照本书的安排做每一章后面的习题，将有助于理解和加深正文所述的概念和知识，并且有时候，可以从实际动手中学习到新的知识。如果不动手，空洞地去看文字，是很难理解文字背后的意义的。我个人的经验是，有许多系统设计和概念，看似简单或不理解，可一旦自己动手做同样的试验，才更明白当初的设计者为什么要如此设计。计算机系统就像自然界的生态环境，对每一个部件的设计都要求它能融洽地和系统内其他部件和平相处，我们不能站在一个微观的视角去看待系统部件的设计是否最优，而应该从宏观来观察和思考。

为方便理解本书的内容，本书的读者假定具备 C 语言编程的能力。由于原书是卡内基梅隆大学（CMU）的教材，且被其他一些著名的大学也选用为教材，因此，本书的读者不仅仅是那些因为工作和兴趣而关注本书的人，还包括一些在校的大学生，作为他们的教材或辅助性资料。个人认为，在校学生越早接触本书的内容，将越有利于他们学习计算机的相关课程，培养对计算机系统的研究兴趣。

总而言之，《深入理解计算机系统》一书是一个桥梁，它帮助程序员衔接了计算机系统的各个领域的知识，为程序员构造了一个概念性框架。对于各个领域（如计算机系统结构、处理器、操作系统、编译器、网络、并发编程）的知识进一步获取，还需要参考相关书籍。

参加翻译的还有龚奕利、易金华和陈永兴等，在此也特别表示感谢。

由于此书的内容量大，加上翻译时间并不很宽裕，尽管我们十分努力，但还是难以避免出现错误，以及存在许多不尽人意的地方，欢迎广大读者批评指正，以便改进。

雷迎春

2004.2.15

于北京中关村（中科院）青年公寓

关于术语的翻译

本书跨越计算机的多个领域，涉及了许多专业的术语。在翻译的过程中，我们尽可能地忠实反映原文的意思，但并不是每个术语的翻译都那么恰当，符合每个读者的阅读习惯。不可避免地，对某些术语的翻译带了我们个人的习惯和偏好，希望读者谅解。下面，我们解释在本书中频繁出现的一些术语的翻译。

directive

这个单词多用来描述 C 语言中类似#include 的语句，或汇编语言中类似.pos 的语句。按照我的认识，这个单词应该译做“指令”比较恰当，起着指导或导引的作用。但是，在 directive 单词出现的地方，还同时出现了 instruction 单词（这种现象以第 3 章为主），其中文的含义也是“指令”。相比于 directive、instruction 显然是一个更强势的单词。为了从中文字面上区分这两个单词，方便读者阅读，我们在不影响基本意思的前提下，翻译 directive 为命令。

operation

这是一个遍布全书的单词。它众多的意思中有两个是：“操作”和“运算”。对这个单词的翻译，我们没有采用一刀切的方法，而是尽可能采用国内读者的习惯来翻译。根据我自己的切身体会，以及网友对 operation 译法的讨论，我们更倾向于在数学的领域内使用“运算”，而在计算机领域使用“操作”。基于这个认识，以及章节内容的安排，我们把第 2 章（该章涉及大量的数学描述）中出现的 operation 主要译做“运算”，而把其他章节中的 operation 主要翻译为“操作”。需要注意的是，这种划分并不是绝对的。

memory 与 storage

memory 是一个我们非常熟悉的术语，我们一般把它习惯地称为“内存”。但是，通过本书第 6 章对 memory 的解释来看，仅有这样的理解是不足够的。本书认为 memory 可以是不同容量、成本和访问时间的存储设备，我们过去所认识的 memory 只是 DRAM。所以，不能把 memory 简单地翻译为“内存”。

从 memory 和 storage 这两个单词的中文意思来看，memory 是“存储器”，而 storage 是“存储，存储器”。另外，我们还观察到，memory 更多地以名词出现，描述一个静态的物理设备，而 storage 除了可以作为名词出现外，还有动词的形式（store、storing 和 stored）。所以，我们取 memory 的中文意思为“存储器”，而取 storage（以及 store、storing 和 stored）的中文意思为“存储”。除此之外，如果在一句话中，有 memory 和 storage 同时出现时，我们除了给出 storage 的中文释义外，还尽可能地附英文单词，以示与 memory 的区别。

hazard

这是一个很扰人的体系结构领域的术语。在本书中，我们选用它的中文释义为“冒险”。实际上，我们在做学生时，大都直呼它的英文，很少说它的中文，选择它的中文译法真的是一件很麻烦的事情。曾经有一个网友告诉我，他看到一个“险象”的译法比较贴切。呵呵，为了这个术语的翻译，我和他在网上争论了两天。仔细想想“险象”这种译法，确实不为错，但还不能完全说服我选用它。因为，这个释义太过陌生，许多读者可能无法联想到其对应的英文单词。而选用“冒险”，尽管不是那么完美，但是大多数研究体系结构的读者会很熟悉。所以，对“冒险”的选用只是一种习惯和默认。

timer

timer 的中文释义有：“定时器，计时器”。尽管这两个中文释义都可以描述一个现象：间隔一段时间后产生一个事件，但是我们认为这两者之间是有区别的。从中文字面来说，定时器的间隔更多是固定的，倾向于静态性；而计时器的间隔更多是不固定的，有计算的意思，倾向于动态性。所以，在本书的第 9 章中（该章主要描述系统评价），我们主要把 timer 翻译为计时器，而在其他章节翻译为定时器。

local

local 的中文释义是：“本地，局部”。我们没有严格地区分它，完全是根据上下文描述的方便，来选用不同的释义。

原书还出现了一些错误，这些错误只是我们个人认为的，很有可能是我们理解错了。所以，我们在“篡改”原文的意思时，还尽可能地给出了原文的意思，以帮助读者甄别。

我很喜欢这本书，且认为它的内容在 5~10 年内都有它存在的价值。但是，鉴于我们的能力和时间有限，不能保证完全忠实、准确地重述原文的意思，还需要广大读者的支持。希望广大读者在阅读本书的时候能积极地给我们指出其中错误，改善此书的质量，方便后来的读者从中更顺畅地获取知识。

前　言

《深入理解计算机系统》（Computer Systems: A Programmer's Perspective，CS: APP）这本书的主要读者是那些想通过学习计算机系统的内在运作而提高自身技能的程序员。

我们的目的是解释所有计算机系统的本质概念，并向你展示这些概念是如何实际地影响应用程序的正确性、性能和实用性的。与其他主要针对系统构造人员的系统类书籍不同，这本书是写给程序员的，是从程序员的角度来描述的。

如果你学习和研究这本书里的概念，你将步入稀缺的“权威程序员”的行列，将知道事情是如何运作的，也知道在出现故障时如何进行修复。同时，你也将做好学习其他具体系统主题的准备，比如编译器、计算机体系结构、操作系统、嵌入式系统和网络互联。

读者所应具备的背景知识

本书中的范例是基于英特尔兼容的处理器（英特尔称之为“IA32”，即俗称的“x86”）、在 Unix 或类 Unix（比如 Linux）操作系统上运行的 C 语言程序。（为了简化我们的描述，我们将用 Unix 统称 Solaris 和 Linux 这样的系统。）文中包括了大量已在 Linux 系统上编译和运行过的程序范例。我们假设你能访问一台这样的机器，并且能够登录，然后做一些诸如修改目录之类的简单操作。

如果你的计算机运行的是 Microsoft Windows 系统，你有两种选择。第一，获取一个 Linux 的拷贝（参见 www.linux.org 或者 www.redhat.com），然后以“双重启动”模式安装它，这样你的机器就能运行任一个操作系统了。另一种选择就是，通过安装 Cygwin 工具（www.Cygwin.com），你就能在 Windows 下得到一个类似 Unix 的 shell 以及一个非常类似于 Linux 提供的环境。不过，Cygwin 并不能提供所有的 Linux 功能。

我们还假设你对 C 和 C++有一定的了解。如果你以前只有 Java 经验，那么这种转换将需要你自己付出更多的努力，不过我们也将帮助你。Java 和 C 有相似的语法和控制语句。

但是，有一些 C 语言的内容，特别是指针、显式的动态存储器分配和格式化 I/O，Java 中都是没有的。所幸的是，C 是一个较小的语言，并且在 Brain Kernighan 和 Dennis Ritchie 经典的“K&R”文字中得到了清晰优美的描述[40]。无论你的编程背景如何，K&R 都应是你个人图书收藏的一部分。

这本书的前几章揭示了 C 语言程序和它们相应的机器语言程序之间的交互作用。机器语言示例都是用运行在 Intel IA32 处理器上的 GNU GCC 编译器生成的，我们不需要你以前有任何硬件、机器语言或是汇编语言编程的经验。

给 C 语言初学者：关于 C 编程语言的建议

为帮助 C 语言编程背景薄弱的（或全无背景的）读者，也为了强调 C 中一些重要特性，我们做了专门的注释。我们假设你熟悉 C++或 Java。

怎样阅读此书

从程序员的角度来学习计算机系统如何工作将非常有趣，主要是因为这个过程可以非常主动。无论何时你学到一些新的东西，都可以马上试验并且直接看到运行结果。事实上，我们相信学习系统的惟一方法就是做（do）系统，即在真正的系统上解决具体的问题，或是编写和运行程序。

这种主题观念贯穿全书。当引入一个新概念时，紧随其后的将是一个或多个练习题，你应该马上做一做来检验你的理解。练习题的解答在每章的末尾。当你阅读时，尝试自己来解答每个问题，然后再查阅答案，看看自己是否正确。每一章后面都有一组不同难度的家庭作业题。你的指导老师在教师手册中有这些问题的答案。对每个家庭作业题，我们标注了我们认为的难度级别：

◆只需要几分钟。几乎或完全不需要编程。

◆◆可能需要将近 20 分钟。通常包括编写和测试一些代码，许多都取自我们在考试中的题目。

◆◆◆需要很大的努力，也许是 1～2 个小时。一般包括编写和测试大量的代码。

◆◆◆◆一个实验作业，需要将近 10 个小时。

文中每段代码示例都是 C 程序，经过版本为 2.95.3 的 GCC 编译并在内核版本为 2.2.16 的 Linux 系统上测试后直接生成的，没有任何人为的改动。所有源程序代码均可从本书的主页（csapp.cs.cmu.edu）上获取。在文中，源程序的文件名列在两条水平线的右边，水平线之间是格式化代码。比如，图 P.1 中的程序能在 code/intro 目录下的 hello.c 文件中找到。我们鼓励你，当遇到这些示例程序时，在你的系统上试试运行它们。

code/intro/hello.c

```
1  #include <stdio.h>
2
3  int main()
4  {
5      printf("hello, world\n");
6  }
```

code/intro/hello.c

图 P.1　一个典型的代码示例

最后，有些部分（用“*”标注的）包含了一些你可能会觉得有趣但可以略过而不影响阅读连贯性的东西。

旁注：什么是旁注？

整本书中，你将会遇到很多以这种形式出现的旁注。旁注是附加说明，能使你对当前讨论的主题多一些了解。旁注有很多目的。一些是小的历史故事，例如，C、Linux 和 Internet 是从何而来的？有时，旁注是用来阐明学生们经常感到疑惑的问题，例如，高速缓存的行、组和块有什么区别？还有的时候，旁注给出了一些现实世界的例子，例如，一个浮点错误怎么毁掉了法国的一枚火箭，或是一个真正的 IBM 磁盘驱动器看上去是什么样子。最后，还有一些旁注仅仅就是笑料，例如，什么是“hoinky”？

把一个常量乘法转化为一系列的移位和加法。我们用 C 的位级操作来说明布尔代数的原理和应用。我们从如何表示浮点值和浮点操作的数学属性方面讲述 IEEE 标准的浮点格式。

对计算机算术的深刻理解是写出可靠程序的关键。比如，不能用（x-y<0）来取代（x<y），因为可能会产生溢出。甚至也不能用表达式（-y<-x）来取代，因为在二进制补码表示中负数和正数的范围是不对称的。算术溢出是程序错误的一个常见根源，然而很少有书从一个程序员的角度去讲述计算机算术的特性。

- 第 3 章：程序的机器级表示。我们教学生如何读由 C 编译器生成的 IA32 汇编语言。我们说明为不同控制结构，比如条件、循环和开关语句，生成的基本指令模式。我们还讲述过程的执行，包括栈分配、寄存器使用惯例和参数传递。我们讨论不同数据结构如结构、联合（union）和数组的分配和访问方式。学习本章的概念能够帮助学生成为更好的程序员，因为他们懂得他们的程序在机器上是如何表示的。另外一个妙处在于学生们对指针有了具体的了解。
- 第 4 章：处理器体系结构。这一章讲述基本的组合和时序逻辑元素，并展示这些元素在数据路径（datapath）中如何组合到一起，来执行 IA32 指令集的一个称为“Y86”的简化子集。我们从设计单时钟周期、非流水线化的数据路径开始，然后扩展成一个五阶段、流水线化的设计。本章中处理器设计的控制逻辑是用一种称为 HCL 的简单硬件描述语言来描述的。用 HCL 写的硬件设计能够编译和链接成本书中提供的图形处理器的模拟器。
- 第 5 章：优化程序性能。在这一章里，我们介绍许多提高代码性能的技术。我们从与机器无关的程序转换开始，这些标准是在任何机器上写任何程序时都应该遵循的。然后是那些功效有赖于目标机器和编译器特性的转换。为了促进这些转换，我们介绍了一个简单的操作模型，它描述了现代乱序（out-of-order）处理器是如何工作的，然后向学生们展示怎样利用这个模型来改进他们的 C 程序的性能。
- 第 6 章：存储器层次结构。对应用程序员来说，存储器系统是计算机系统中最直接可见的部分之一。到目前为止，学生们一直认同这样一个存储器系统概念模型，认为它是一个有一致访问时间的线性数组。实际上，存储器系统是一个由不同容量、造价和访问时间的存储设备组成的层次结构。我们讲述不同类型的随机存取存储器（RAM）和只读存储器（ROM）以及现代磁盘驱动器的几何形状和组织构造。我们描述这些存储设备是如何放置在层次结构中的，讲述访问局部性是如何使这种层次结构成为可能的。我们通过一个独特的观点使这些理论具体化、形象化，那就是将存储器系统视为“存储器山”，山脊是时间局部性，而斜坡是空间局部性。最后，我们向学生们阐述如何通过改善时间和空间局部性来提高应用程序的性能。
- 第 7 章：链接。本章讲述静态和动态链接，包括的概念有可重定位的（relocatable）和可执行的目标文件、符号解析、重定位（relocation）、静态库、共享目标库，以及与位置无关的代码。大多数系统书中都不涉及链接，而我们出于下面几个原因要讲述它。第一，学生们遇到的最迷惑的问题中，有一些是和链接时的小故障有关，尤其是对那些大型软件包来说。第二，链接器生成的目标文件是与一些像加载、虚拟存储器和存储器映射这样的概念相关的。
- 第 8 章：异常控制流。在课程的这个部分，我们通过介绍异常控制流（比如，正常分支和过程调用以外的控制流变化）的一般概念打破单一程序的模型。我们给出存在于系统所有层次的异常控制流的例子，从底层的硬件异常和冲突，到并发进程的上下文切换，到 Unix 信号

本书的起源

本书起源于 1998 年秋季我们在卡内基梅隆（CMU）大学开设的一门编号为 15-213 的介绍性课程：计算机系统导论（Introduction to Computer System，ICS）[7]。从那以后，每学期都开设了 ICS 这门课程，每期有 150 名左右的学生，大多数是计算机科学和计算机工程专业二年级的学生。后来，这门课程还成为了卡内基梅隆大学计算机科学系以及电子和计算机工程系中大多数高级系统课程的先行必修课。

ICS 课程的宗旨是用一种不同的方式向学生介绍计算机。因为，我们的学生中几乎没有人有机会构造计算机系统。另一方面，大多数学生，甚至是计算机工程师，也要求日常能使用计算机和编写计算机程序。所以我们决定从程序员的角度来讲解系统，并采用这样的过滤方法：我们只讨论那些影响用户级 C 程序的性能、正确性或实用性的主题。

比如，我们排除了诸如硬件加法器和总线设计这样的主题。虽然我们谈及了机器语言，但是不关注如何编写汇编语言，而是关心 C 程序是如何被构造的，例如编译器是如何翻译指针、循环、过程调用和返回以及开关（switch）语句的。更进一步，我们将更广泛和现实地看待系统，包括硬件和系统软件，涵盖了链接、加载、进程、信号、性能优化、评估、I/O 以及网络与并发编程。

这种做法使得我们讲授 ICS 课程的方式对学生来讲既实用、具体，还能动手，同时也非常能调动学生的积极性。很快地，我们收到来自学生和教职工非常热烈和积极的反响，我们意识到卡内基梅隆大学以外的其他人也可以从我们的方法中获益。因此，历时两年，这本书从 ICS 课程笔记中应运而生了。

旁注：与 ICS 有关的数字

跟 ICS 课程有关的数字很特别。在第一学期过半的时候，我们发现课程的编号（15-213）正好就是卡内基梅隆大学的邮政编码，因此，还有了这样的话："15-213：给予卡内基梅隆大学精神的课程！"[1] 无独有偶，手稿的第一版是在 2001 年 2 月 13 日（2/13/01）印刷的。当我们在 SIGCSE[2] 教育会议上介绍这门课程时，被安排在了 213 房间，并且此书的最后一版有 13 个章节。好在我们并不迷信！

本书概述

本书由 13 章组成，旨在阐述计算机系统的核心概念。

- 第 1 章：计算机系统漫游。这一章通过研究"hello, world"这个简单程序的生命周期，介绍计算机系统的主要概念和主题。
- 第 2 章：信息的表示和处理。我们讨论计算机算术，重点描述对程序员有影响的无符号和二进制补码（two's complement）的数字表示法的特性。我们考虑数字是如何表示的，以及由此确定对于一个给定的字长，其可能编码值的范围。我们探讨有符号和无符号数字之间类型转换的效果，还阐述算术操作的数学特性。学生们很惊奇地了解到（二进制补码表示的）两个正数的和或者积可以为负。另一方面，二进制补码算法满足环的特性，因此，编译器可以

1 zip 既有"邮政编码"也有"精神"之意。——译者

2 SIGCSE 代表 Special Interest Group on Computer Science Education，计算机科学教育特殊兴趣组。——译者

传送引起的控制流突变，到 C 中破坏栈原则的非本地跳转（nonlocal jump）。

在这一章，我们还向学生们介绍进程的基本概念。学生们了解进程是如何工作的，以及如何在应用程序中创建和操纵进程。我们向他们展示应用程序员如何通过 Unix 系统调用使用多进程。学完本章，他们就能够编写带作业控制的 Unix 脚本了。

- 第 9 章：测量程序运行时间。这一章教给学生计算机是如何理解时间的［时间间隔计时器、CPU 周期计时器（cycle timer）和系统时钟］，当我们试图用这些时间来测量程序运行时时间的错误根源，以及怎样运用这些知识来得到准确的度量值。据我们所知，这是惟一的在以前还未以任何常规的方式讨论过的内容。我们在此讨论这个主题是因为它需要对汇编语言、进程和高速缓存有所了解。
- 第 10 章：虚拟存储器。我们讲述虚拟存储器系统是希望学生们对它的工作和特性有所了解。我们想让学生了解为什么不同的并发进程各自都有一个相同的地址范围，能共享某些页，但另外一些页又是独占的。我们还覆盖一些管理和操作虚拟存储器的问题。特别地，我们讨论了存储分配操作，比如 Unix 的 malloc 和 free 操作。阐述这些内容是出于下面几点目的。它加强了虚拟存储器空间只是字节数组，程序可以把它划分成不同存储单元的概念。它帮助学生理解包含有像存储泄漏和非法指针引用这样存储器引用错误的程序的后果。最后，许多应用程序员编写自己的优化了的存储分配操作来满足应用程序的需要和特性。
- 第 11 章：系统级 I/O。我们讲述 Unix I/O 的基本概念，例如文件和描述符。我们描述如何共享文件，I/O 重定向是如何工作的，还有如何访问文件的元数据。我们还开发了一个健壮的带缓冲区的 I/O 包，可以正确处理 short counts。我们阐述 C 的标准 I/O 库，以及它与 Unix I/O 的关系，重点谈到标准 I/O 的局限性，这些局限性使之不适合网络编程。总地说来，本章的论题是后面两章网络和并发编程的基础。
- 第 12 章：网络编程。对编程而言，网络是非常有趣的 I/O 设备，将许多我们前面文中学习的概念，比如进程、信号、字节顺序（byte ordering）、存储器映射和动态存储器分配，联系在一起。网络程序还为并发提供了强制性上下文，这是下一章的论题。本章是网络编程的细小片段，使学生们能够编写 Web 服务器。我们还讲述位于所有网络程序底层的客户端-服务器模型。我们展现了一个程序员对 Internet 的观点，并且教给学生们如何用套接字（socket）接口来编写 Internet 客户端和服务器。最后，我们介绍超文本传输协议 HTTP，并开发了一个简单的迭代式（iterative）Web 服务器。
- 第 13 章：并发编程。这一章以 Internet 服务器设计为例向学生们介绍了并发编程。我们比较对照了三种编写并发程序的基本机制（进程、I/O 多路复用技术以及线程），并且展示如何用它们来建造并发 Internet 服务器。我们探讨了用 P、V 信号操作、线程安全和可重入（reentrancy）、竞争条件以及死锁等来实现同步的基本原则。

可以基于本书的课程

指导教师可以使用本书来教授五种不同的系统课程（图 P.2）。特殊的课程则有赖于课程需要、个人品位、学生的背景和能力。图中的课程从左往右，逐渐强调以程序员的角度看待系统，以下是简单的描述：

- **ORG**：一门以非传统风格介绍传统问题的计算机组成原理课程。传统的主题包括逻辑设计、处理器体系结构、汇编语言和存储器系统。然而，需要更多地强调对程序员的影响。例如，要反过来考虑数据表示对C程序的影响。学生们将学习到如何用机器语言来表示C结构。
- **ORG+**：ORG课程特别强调硬件对应用程序性能的影响。和ORG课程相比，学生要更多地学习代码优化和改进他们C程序的存储器性能。
- **ICS**：基本的ICS课程，旨在培养开明的程序员，他们理解硬件、操作系统和编译系统对应用程序的性能和正确性的影响。和ORG+课程的一个显著不同是，本课程不论及低级处理器体系结构。相反地，程序员与现代乱序处理器的高级模型打交道。ICS课程非常适合安排成一个10周的学期，如果步调更从容一些，也可以延长为一个15周的学期。
- **ICS+**：基本的ICS课程，额外论述一些系统编程问题，比如系统级I/O、网络编程和并发编程。这是一门一学期长度的卡内基梅隆大学课程，会讲述本书中除了低级处理器体系结构以外的每一章。
- **SP**：一门系统编程课程。和ICS+课程相似，但是抛弃了浮点和性能优化，更加强调系统编程，包括进程控制、动态链接、系统级I/O、网络编程和并发编程。指导教师可能会想从其他渠道对某些高级论题做些补充，比如守护进程（daemon）、终端控制和Unix IPC（进程间通信）。

		课程				
章节	论题	ORG	ORG+	ICS	ICS+	SP
1	系统漫游	•	•	•	•	•
2	数据表示	•	•	•	•	⊙ (d)
3	机器语言	•	•	•	•	•
4	处理器体系结构	•	•			
5	代码优化		•	•	•	
6	存储器层次结构	⊙ (a)	•	•	•	⊙ (a)
7	链接			⊙ (c)	⊙ (c)	•
8	异常控制流			•	•	•
9	性能测量				•	•
10	虚拟存储器	⊙ (b)	•	•	•	•
11	系统级I/O				•	•
12	网络编程				•	•
13	并发编程				•	•

图P.2 五门基于本书的课程

注意：（a）只有硬件；（b）无动态存储分配；（c）无动态链接；（d）无浮点。ICS+是卡内基梅隆的15-213课程。

图P.2要表达的主要信息是本书给了你多种选择。如果你希望你的学生更多地了解低级处理器体系结构，那么通过ORG和ORG+课程可以达到目的。另一方面，如果你想将当前的计算机组成课程转换成ICS或者ICS+课程，但是又担心突然做这样猛烈的变化，那么你可以逐步递增转向ICS课程。你可以从OGR课程开始，它以一种非传统的方式教授传统的问题。一旦你对这些内容感到驾轻就熟

了，就可以转到ORG+，最终转到ICS。如果学生没有C的经验（比如他们只用Java编过程序），你可以花几周的时间在C上，然后再讲述ORG或者ICS课程的内容。

最后，我们认为ORG+和SP课程适合安排为两期（两个季度或者两个学期）。或者你可以考虑按照一期ICS和一期SP的方式来教授ICS+课程。

课堂测试的实验练习

卡内基梅隆大学的ICS+课程得到了学生们很高的评价。这门课的中值分数一般为5.0/5.0，平均分数一般为4.6/5.0。学生们表扬说这门课非常有趣，令人兴奋，主要就是因为相关的实验练习。下面是本书提供的一些实验的示例。

- *数据实验*。这个实验要求学生们实现简单的逻辑和算术函数，但是只能使用一个高度受限的C的子集。比如，他们必须只能用位级操作来计算一个数字的绝对值。这个实验帮助学生们了解C数据类型的位级表示，和数据操作的位级行为。
- *二进制炸弹实验*。二进制炸弹是一个作为目标代码文件提供给学生们的程序。运行时，它提示用户输入6个不同的字符串。如果其中的任何一个不正确，炸弹就会“爆炸”，打印出一条错误信息，并且在分级（grading）服务器上记录事件日志。学生们必须通过对程序反汇编和逆向工程来测定应该是哪6个串，从而解除他们各自炸弹的雷管。该实验教会学生理解汇编语言，并且强制他们学习怎样使用调试器。
- *缓冲区溢出实验*。它要求学生们通过研究一个缓冲区溢出的错误，来修改二进制可执行文件的运行时行为。这个实验教会学生们栈的原理，并让他们了解到写那种易于遭受缓冲区溢出攻击的代码的危险性。
- *体系结构实验*。第4章的几个家庭作业问题能够组合成一个实验作业，在实验中，学生们修改处理器的HCL描述以增加新的指令、修改分支预测策略，或者增加或删除旁路路径和寄存器端口。设计出来的处理器能够被模拟，并通过运行自动化测试检测出大多数可能的错误。这个实验使学生们能体验到处理器设计中令人激动的部分，而不需要他们学习和建造用Verilog或者VHDL语言写的复杂而低级的模块。
- *性能实验*。学生们必须优化应用的核心函数（比如卷积积分或矩阵转置）的性能。这个实验非常清晰地表明了高速缓存的特性，并给学生们低级程序优化的经验。
- *shell实验*。学生们实现他们自己的带有作业控制的Unix shell程序，包括ctrl-c和ctrl-z按键、fg、bg和jobs命令。这是学生们第一次接触并发，并且让他们对Unix的进程控制、信号和信号处理有清晰的了解。
- *malloc实验*。学生们实现他们自己的malloc、free和realloc（可选地）版本。这个实验让学生们清晰地理解数据的布局和组织，并且要求他们评估时间和空间效率的各种权衡和折中。
- *代理实验*。学生们实现一个位于浏览器和万维网其他部分之间的并行Web代理。这个实验向学生们揭示了Web客户端和服务器这样的问题，并且联系起了课程中许多概念，比如字节排序、文件I/O、进程控制、信号、信号处理、存储器映射、套接字和并发。

本书的教师手册有对实验的详细讨论，还有关于下载支持软件的说明。

致 谢

我们衷心地感谢那些给了我们中肯批评和鼓励的众多朋友及同事。特别感谢我们 15-213 课程的学生们，他们充满感染力的精力和热情鞭策我们前行。Nick Carter 和 Vinny Furia 无私地提供了他们的 malloc 程序包。

Guy Blelloch、Greg Kesden、Bruce Maggs 和 Todd Mowry 在多个学期中教授此课，给我们鼓励并帮助改进课程内容。Herb Derby 提供了早期的精神指导和鼓励。Allan Fisher、Garth Gibson、Thomas Gross、Satya、Peter Steenkiste 和 Hui Zhang 从一开始就鼓励我们开设这门课程。Garth 早期给的建议促使本书的工作得以开展，并且在 Allan Fisher 领导的小组的帮助下又细化和修订了本书的工作。Mark Stehlik 和 Peter Lee 提供了极大的支持，使得这些内容成为本科生课程的一部分。Greg Kesden 针对 ICS 在操作系统课程上的影响提供了有益的反馈意见。Greg Ganger 和 Jiri Schindler 提供了一些磁盘驱动的描述说明，并回答了我们关于现代磁盘的疑问。Tom Striker 向我们展示了存储器山的比喻。James Hoe 在处理器体系结构方面提出了很多有用的建议和反馈。

有一群特殊的学生极大地帮助我们发展了这门课程的内容，他们是 Khalil Amiri、Angela Demke Brown、Chris Colohan、Jason Crawford、Peter Dinda、Julio Lopez、Bruce Lowekamp、Jeff Pierce、Sanjay Rao、Balaji Sarpeshkar、Blake Scholl、Sanjit Seshia、Greg Steffan、Tiankai Tu、Kip Walker 和 Yinglian Xie。尤其是 Chris Colohan 建立了愉悦的氛围并持续到今天，还发明了传奇般的“二进制炸弹”，这是一个对教授机器语言代码和调试概念非常有用的工具。

Chris Bauer、Alan Cox、Peter Dinda、Sandhya Dwarkadis、John Greiner、Bruce Jacob、Barry Johnson、Don Heller、Bruce Lowekamp、Greg Morrisett、Brian Noble、Bobbie Othmer、Bill Pugh、Michael Scott、Mark Smotherman、Greg Steffan 和 Bob Wier 花费了大量时间阅读此书的早期草稿，并给予我们建议。特别感谢 Peter Dinda（西北大学）、John Greiner（莱茨大学）、Wei Hsu（明尼苏达大学）、Bruce Lowekamp（威廉&玛丽大学）、Bobbie Othmer（明尼苏达大学）、Michael Scott（罗彻斯特大学）和 Bob Wier（落基山学院）在教学中测试此书的试用版。同样特别感谢他们的学生们！

我们还要感谢 Prentice Hall 出版公司的同事。感谢 Marcia Horton、Eric Frank 和 Harold Stone 不懈地支持和远见。Harold 还帮我们提供了对 RISC 和 CISC 处理器体系结构准确的历史观点。Jerry Ralya 有惊人的见识，并教会了我们很多如何写作的知识。

最后，我们衷心感谢伟大的技术作家 Brian Kernighan 以及后来的 W. Richard Stevens，他们向我们证明了技术书籍也能写得如此优美。

谢谢你们所有的人。

Randy Bryant
Dave O'Hallaron