HZ BOOKS
华章教育

计 算 机 科 学 丛 书

CRC Press
Taylor & Francis Group

原书第2版

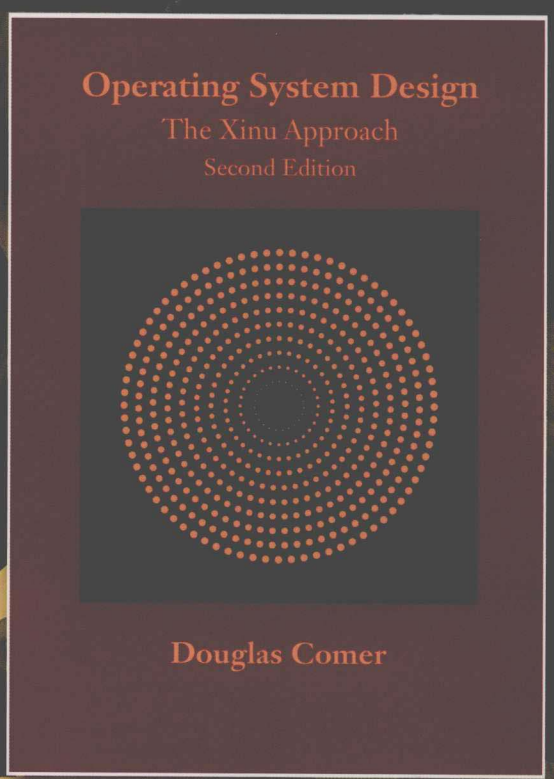
操作系统设计
Xinu方法

[美] 道格拉斯·科默（Douglas Comer） 著
普度大学

陈向群
北京大学

郭立峰
尔雅慧联公司
等译

Operating System Design
The Xinu Approach Second Edition



机械工业出版社
China Machine Press

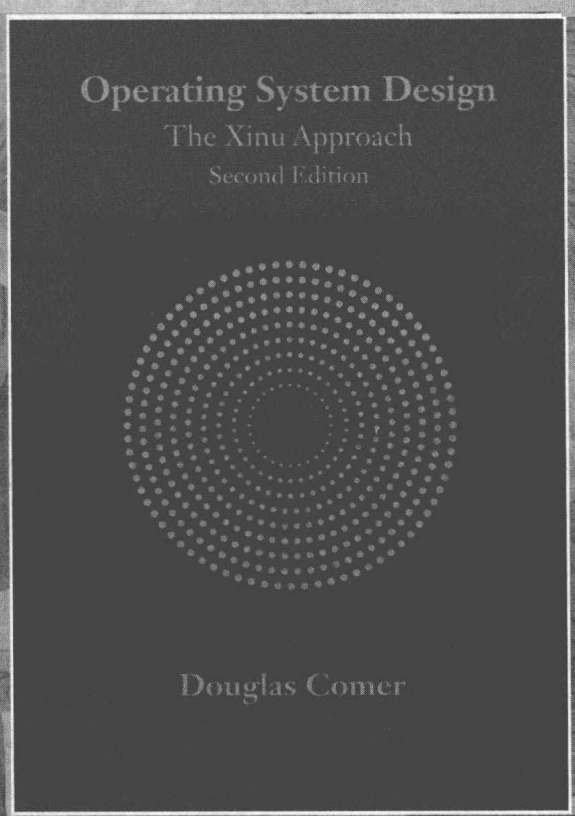
计 算 机 科 学 丛 书

原书第2版

操作系统设计
Xinu方法

[美] 道格拉斯·科默（Douglas Comer） 著 普度大学
陈向群 北京大学
郭立峰 尔雅慧联公司 等译

Operating System Design
The Xinu Approach Second Edition



机械工业出版社
China Machine Press

图书在版编目（CIP）数据

操作系统设计：Xinu 方法（原书第 2 版）/（美）道格拉斯·科默（Douglas Comer）著；陈向群，郭立峰等译．—北京：机械工业出版社，2019.3
（计算机科学丛书）
书名原文：Operating System Design: The Xinu Approach, Second Edition

ISBN 978-7-111-62191-1

I. 操… II. ①道… ②陈… ③郭… III. 操作系统－程序设计 IV. TP316

中国版本图书馆 CIP 数据核字（2019）第 043625 号

本书版权登记号：图字 01-2016-2919

本书以 Xinu（一个小型简洁的操作系统）为例，全面介绍操作系统设计方面的知识。本书着重讨论用于嵌入式设备的微内核操作系统，采用的方法是在现有的操作系统课程中纳入更多的嵌入式处理内容，而非引入一门教读者如何在嵌入式系统上编程的新课程。

本书从底层机器开始，一步步地设计和实现一个小型但优雅的操作系统 Xinu，指导读者通过实用、简单的原语来构造传统的基于进程的操作系统。本书回顾了主要的系统组件，并利用分层设计范式，以一种有序、易于理解的方式组织内容。

作者的网站 www.xinu.cs.purdue.edu 提供了便于学生搭建实验环境的软件和资料。

本书适用于计算机专业高年级本科生或低年级研究生，也适用于需要了解操作系统知识的 IT 相关从业者。

出版发行：机械工业出版社（北京市西城区百万庄大街 22 号　邮政编码：100037）
责任编辑：佘　洁　　责任校对：李秋荣
印　　刷：三河市宏图印务有限公司　　版　　次：2019 年 4 月第 1 版第 1 次印刷
开　　本：185mm×260mm　1/16　　印　　张：30
书　　号：ISBN 978-7-111-62191-1　　定　　价：99.00 元

凡购本书，如有缺页、倒页、脱页，由本社发行部调换
客服热线：（010）88378991　88379833　　投稿热线：（010）88379604
购书热线：（010）68326294　　读者信箱：hzjsj@hzbook.com

文艺复兴以来，源远流长的科学精神和逐步形成的学术规范，使西方国家在自然科学的各个领域取得了垄断性的优势；也正是这样的优势，使美国在信息技术发展的六十多年间名家辈出、独领风骚。在商业化的进程中，美国的产业界与教育界越来越紧密地结合，计算机学科中的许多泰山北斗同时身处科研和教学的最前线，由此而产生的经典科学著作，不仅擘画了研究的范畴，还揭示了学术的源变，既遵循学术规范，又自有学者个性，其价值并不会因年月的流逝而减退。

近年，在全球信息化大潮的推动下，我国的计算机产业发展迅猛，对专业人才的需求日益迫切。这对计算机教育界和出版界都既是机遇，也是挑战；而专业教材的建设在教育战略上显得举足轻重。在我国信息技术发展时间较短的现状下，美国等发达国家在其计算机科学发展的几十年间积淀和发展的经典教材仍有许多值得借鉴之处。因此，引进一批国外优秀计算机教材将对我国计算机教育事业的发展起到积极的推动作用，也是与世界接轨、建设真正的世界一流大学的必由之路。

机械工业出版社华章公司较早意识到“出版要为教育服务”。自 1998 年开始，我们就将工作重点放在了遴选、移译国外优秀教材上。经过多年的不懈努力，我们与 Pearson、McGraw-Hill、Elsevier、MIT、John Wiley & Sons、Cengage 等世界著名出版公司建立了良好的合作关系，从它们现有的数百种教材中甄选出 Andrew S. Tanenbaum、Bjarne Stroustrup、Brian W. Kernighan、Dennis Ritchie、Jim Gray、Afred V. Aho、John E. Hopcroft、Jeffrey D. Ullman、Abraham Silberschatz、William Stallings、Donald E. Knuth、John L. Hennessy、Larry L. Peterson 等大师名家的一批经典作品，以“计算机科学丛书”为总称出版，供读者学习、研究及珍藏。大理石纹理的封面，也正体现了这套丛书的品位和格调。

“计算机科学丛书”的出版工作得到了国内外学者的鼎力相助，国内的专家不仅提供了中肯的选题指导，还不辞劳苦地担任了翻译和审校的工作；而原书的作者也相当关注其作品在中国的传播，有的还专门为其书的中译本作序。迄今，“计算机科学丛书”已经出版了近 500 个品种，这些书籍在读者中树立了良好的口碑，并被许多高校采用为正式教材和参考书籍。其影印版“经典原版书库”作为姊妹篇也被越来越多实施双语教学的学校所采用。

权威的作者、经典的教材、一流的译者、严格的审校、精细的编辑，这些因素使我们的图书有了质量的保证。随着计算机科学与技术专业学科建设的不断完善和教材改革的逐渐深化，教育界对国外计算机教材的需求和应用都将步入一个新的阶段，我们的目标是尽善尽美，而反馈的意见正是我们达到这一终极目标的重要帮助。华章公司欢迎老师和读者对我们的工作提出建议或给予指正，我们的联系方法如下：

华章网站：www.hzbook.com
电子邮件：hzjsj@hzbook.com
联系电话：（010）88379604
联系地址：北京市西城区百万庄南街 1 号
邮政编码：100037

华章科技图书出版中心

译者序

Operating System Design: The Xinu Approach, Second Edition

《操作系统设计：Xinu 方法》一书是道格拉斯·科默先生的倾力巨作。道格拉斯·科默博士是美国普度大学计算机科学系资深教授、美国计算机学会（ACM）会员、因特网体系结构委员会（IAB）成员，是计算机诸多领域的领军人物。

Xinu 经过近 40 年的发展和完善，已经成为一个小而优美的操作系统。它最早可以追溯到 1979 年——在 LSI-11 机器上将网络协议整合进操作系统中。有兴趣的读者可以通过官方网站（https://xinu.cs.purdue.edu/）了解 Xinu 的整个发展历程。

得益于 Xinu 小而美的特性，本书采用代码实践的方式帮助读者快速、深入地学习操作系统的理论知识。不同于介绍其他操作系统的大部头书籍，当前操作系统的基础理论知识全部浓缩在小小的 Xinu 里面，通过对本书进行系统化的研习并且调试 Xinu 提供的源代码，能大大降低读者学习操作系统的难度和缩短学习时间。

自 1996 年开始，Xinu 逐渐在世界上很多知名大学的操作系统课程教学中引入。北京大学于 2018 年春季开始尝试将 Xinu 引入本科生教学中。作为操作系统的实践课程，Xinu 得到了学生们的积极响应，取得了非常好的教学效果。传统的操作系统课程学习一般是先从理论知识开始，抽丝剥茧，然后再触及代码实践。Xinu 非常特别地从动手写代码开始，一点一滴从零开始构建一个操作系统，这种方法能够帮助初学者快速理解操作系统深奥的理论知识。Xinu 课程的开设极大地促进了学生的动手实践能力，在课堂演示、拓展 Xinu 功能的时候，学生们也会将操作系统原理课上学到的理论知识融入 Xinu 的代码中。

简言之，本书有三大特点：

1）实践性强。从实践中来，到理论中去，读者只要循序渐进地按照书中的代码进行调试和分析，就可以快速掌握操作系统的核心知识。

2）简明扼要。在操作系统领域，本书是最薄的一本，却涵盖了整个操作系统知识的方方面面。

3）便于使用。根据本书和 Xinu 网站的指导，读者可以快速地搭建 Xinu 开发环境并进行学习。

Xinu 的学习是非常简单和快乐的，在随处可见的 MIPS、ARM、x86 平台上都可以进行 Xinu 学习的代码实践，这些平台一般都比较廉价和易于获得。通过运行、调试和改善 Xinu 代码，读者可以快速领悟操作系统知识。

打开本书，挑选一个合适的硬件平台（MIPS、ARM 或 x86），并尝试运行和修改 Xinu 吧！操作系统理论知识晦涩、难于理解的困难很快便会在你的学习中迎刃而解。

本书的出版得到了机械工业出版社华章公司副总经理温莉芳女士的大力支持，华章公司计算机出版中心多位编辑也付出了辛勤劳动，在此表示由衷感谢。

参加本书审阅和校对的还有北京大学 2018 年春季和秋季选修操作系统实习课程 Xinu 模块的 56 名学生，在此对他们的贡献表示诚挚的感谢。

由于译者水平有限，译文中必定会存在一些不足或错误之处，欢迎各位专家和广大读者批评指正。

建立计算机操作系统就像编织锦缎，这两种工作的最终成品都是一个和谐一致、大型、复杂的人造系统。且在以上两种工作中，最后的人造成品都是经由细微、精巧的步骤所构造的。在编织锦缎时，细节至关重要，因为一点点不协调的瑕疵都很容易被观察到。就像锦缎里的缎面一样，加入操作系统里的每个新组件都需要与整体设计相协调。因此，将不同组件组装起来的机械加工只是整个建造过程中的一小部分，一个大师级产品必须以某个模式为蓝本，所有参与系统设计的工作人员都必须遵循这个模式。

具有讽刺意味的是，现有的操作系统教材或课程很少对底层的模式和原理进行解释，而这些模式和原理正是操作系统构造的基础。在学生看来，操作系统似乎是一个暗箱，而现有的教材则加深了这种误解，因为这些教材所解释的不过是操作系统的特性，其关注的也只是操作系统各种功能的使用。更为重要的是，学生在学习操作系统时采取的是从操作系统外面来查看的方式，从而常常导致这样一种感觉：操作系统由一组接口函数组成，这些接口下的功能由一大堆晦涩神秘的代码连接在一起，而这些神秘的代码本身还包含着许多与机器硬件直接相关的、无规律可循的奇技巧术。

令人惊奇的是，学生一旦从大学毕业，就马上觉得与操作系统有关的研究工作已经结束，自己不再需要理解或学习操作系统了，因为由商业公司和开源社区所构造的现有操作系统足以应付各种需要——没有什么比这种想法离真理更远了。与之相反，尽管为个人计算机设计传统操作系统的公司数量比以前更少了，但社会和行业对操作系统专门技术的需求却在增长，许多公司雇佣学生来从事操作系统方面的工作。这些需求增长主要源于更便宜的微处理器，而这些便宜的微处理器被广泛嵌入在智能手机、视频游戏产品、无线传感器、线缆和机顶盒以及打印机等设备中。

在与嵌入式系统打交道时，有关原理和结构的知识非常关键，因为程序员可能需要在现有的操作系统内部构造某种新的机制，或者对现有操作系统进行修改以便可以在新的硬件平台上运行。此外，为嵌入式设备编写应用程序需要理解底层操作系统。如果不理解操作系统设计的各种细微之处，则不可能充分开发小型嵌入式处理器的功能。

本书的目的是揭开操作系统设计的神秘面纱，将方方面面的材料整合为一个系统化的整体。本书对操作系统的主要系统组件进行了详细阐述，并以一种层次架构的设计范例来组织这些组件，从而以一种有序、可理解的方式来展开内容。与其他尽可能多地提供不同方案的评述性书籍不同的是，本书引导读者使用实用的、直截了当的原语来构造一个传统的基于进程的操作系统，从裸机开始，一步一步地设计和实现一个小型但优雅的操作系统。这个名为 Xinu 的操作系统将成为系统设计的一个样板和模式。

虽然 Xinu 操作系统的规模较小，可以完全容纳在本书中，但是该系统包含了构成一个普通操作系统的全部组件：内存管理、进程管理、进程协调和同步、进程间通信、实时时钟管理、设备无关的 I/O、设备驱动、网络协议和文件系统。本书将这些组件组织成一个多层次架构，这使得它们之间的相互连接清晰可见、设计过程浅显易懂。尽管规模小，Xinu 却拥有大型系统的大部分功能。此外，Xinu 并不是一个“玩具”系统，它在很多商业产品中

得到了应用。使用该系统的厂商包括 Mitsubishi、Lexmark、HP、IBM、Woodward（woodward.com）、Barnard Software 和 Mantissa 公司。读者通过本书可以学到的重要一课是：不管是小型嵌入式系统还是大型系统，好的系统设计都一样重要，而好的设计是通过选择好的抽象方法来实现的。

本书所覆盖的议题都以一种特定的次序排列，这种次序就是设计人员在构建操作系统时所遵守的工作次序。本书的每一章描述了设计架构里的一个组件，并提供示例软件来演示该层架构所提供的功能。使用这种方式具有如下优点：第一，每一章所解释的操作系统的功能子集均比上一章所讨论的功能子集更大，这种安排使得我们在考虑一层特定架构的设计和实现时不用关心后续层的实现。第二，一个特定章节的细节描述在第一次阅读时可以跳过去，读者只需要理解该层所提供的服务即可，而不是这些服务是如何实现的。第三，如果按次序阅读本书，读者可以先理解某个功能，然后再应用该功能。第四，有智力挑战的议题（如对并发的支持）出现在书的较前面，高层次的操作系统服务则在后面展示。在本书中，读者将看到大多数核心功能仅仅用几行代码就可以完成，这样我们就可以将大体量的代码（网络和文件系统）放到书的较后面，在读者已经做好充分的思想准备后再进行讲解。

如前所述，与许多其他关于操作系统的书籍不同，本书并不试图对每个系统组件的每种实现方案进行评估，也不对现有的商业系统进行综述，而是对一组使用广泛的操作系统原语的实现细节进行阐述。例如，在讨论进程协调的一章，我们解释的是信号量（使用最广泛的进程协调原语），而将其他原语（如监视器）的讨论放至练习中。我们的目的是展示如何在传统硬件上实现原语，消除其神秘感。学生一旦理解了一组特定原语的魔力，其他原语的实现也就容易掌握了。

本书展示的 Xinu 代码可以运行在多种硬件平台上。我们将关注两种使用不同的流行处理器架构的低成本实验板，分别是基于 Intel（x86）处理器的 Galileo 实验板和基于 ARM 处理器的 BeagleBone Black 实验板。这个范例展示了程序员如何使用常规工具（编辑器、编译器和链接器）来创建 Xinu 镜像，然后把这个镜像加载到一个目标板上，并启动 Xinu 操作系统。

本书适用于高年级本科生或者研究生，也适用于那些想了解操作系统的计算机从业人员。虽然本书所提供的议题的难度都在可理解的范围内，但要在一个学期内学完本书依然需要较快的速度，而一般本科生难以达到。极少本科生擅长读代码，理解运行时环境或机器架构的学生则更少。因此，在进程管理和进程同步相关章节中，必须仔细地对学生进行引导。选择要忽略的内容很大程度上取决于选修课程的学生的背景。如果时间有限，笔者建议学习第 1～7 章（进程管理）、第 9 章（基本内存管理）、第 12 章（中断处理）、第 13 章（时钟管理）、第 14 章（设备无关的 I/O）和第 19 章（文件系统）。如果学生已经学习了包含内存管理和链表操作的数据结构课程，那么可以跳过第 4 章和第 9 章。对于学生来说，了解大多数操作系统都涉及网络通信是很重要的。然而，如果他们将要学习一门独立的网络相关课程，那么可以跳过第 17 章的网络协议内容。本书包含一个远程磁盘系统（第 18 章）和一个远程文件系统（第 20 章），其中一个可以跳过。远程磁盘系统章节可能更有针对性，因为它介绍了磁盘块缓存的内容，而磁盘块缓存是许多操作系统的核心。

对于研究生课程，课堂时间可以用来讨论动机、原理、折中、不同原语集和不同的实现方案比较。在本课程学习结束后，学生应当对进程模型、中断和进程之间的关系有一个深刻的理解，同时也将具备理解、创建和修改系统组件的能力。学生应当在大脑中建立起整个系

统的完整概念模型，并且知道所有组件之间是如何交互协作的。不管是本科生课程还是研究生课程，都应该包括的两个议题是：1）在启动过程中，当一个串行程序转化为一个进程时所发生的重要改变；2）当输入行里的字符序列作为一个字符串变量传递给命令进程时，在操作系统外壳（shell）所发生的转化。

在所有情况下，如果学生能够对系统进行动手实验，则学习的效果将大幅提高。我们选择了低成本的实验板（低于 50 美元的价格就可以获取到），这意味着每个学生都可以买得起实验板和将其连接到笔记本电脑或其他计算机所需要的电缆。在理想状态下，学生可以在课程的最初几天或几个星期内开始使用这个系统，然后再试图理解系统的内部结构。本书第 1 章提供了几个例子和一些能够引起学生兴趣的实验（令人吃惊的是，很多学生学习过操作系统课程，却没有写过一个并发程序或使用过操作系统功能）。许多练习都对系统代码提出了改进、测试和替代实现方案。更大的项目同样也是可能的。结合不同硬件来使用的示例包括：分页系统、跨计算机同步执行的机制以及虚拟网络的设计。其他学生将 Xinu 移植到各种处理器或为各种 I/O 设备构建设备驱动程序。当然，编程语言的背景也是需要的——从事代码工作需要具有 C 语言编程能力和对数据结构有基本了解，包括链表、堆栈和队列。

在普度大学，我们有一个实验室，该实验室有一个自动化系统可提供对实验板的访问。一个学生使用传统的 Linux 系统上的跨平台工具创建了一个 Xinu 镜像。然后，该学生使用实验室的网络分配一个实验板、把镜像加载到实验板、将控制台线从板上连接到学生屏幕上的一个窗口并启动镜像来运行一个应用程序。有关详情请联系作者或浏览网页：

www.xinu.cs.purdue.edu

本书的成书要归功于笔者过去在商业操作系统上所获得的各种经验，这些经验有好也有坏。虽然 Xinu 操作系统与现有的操作系统在内部机制上并不相同，但其基本思想并不新颖。另外，虽然 Xinu 系统里的许多思想和名称都来自 UNIX 系统，但读者应当注意，这两个系统的许多函数所使用的参数和内部结构有巨大的差异。因此，为一个系统所写的应用程序在未经修改的情况下不能在另一个平台上运行，因为 Xinu 不是 UNIX。

衷心感谢为 Xinu 项目贡献了思想、辛劳和激情的所有人。在过去的岁月里，普度大学的许多研究生都从事过 Xinu 系统的工作，他们为 Xinu 进行过移植，写过设备驱动。本书的 Xinu 版本是原始版本的一个完全重写，并且普度大学的许多学生都对本书做出了贡献。当我们更新代码时，我们努力保持原始设计的优雅。Rajas Karandikar 和 Jim Lembke 开发了驱动程序和在 Galileo 平台上使用的多步骤下载系统。在笔者所教授的操作系统班级里，包括 Andres Bravo、Gregory Essertel、Michael Phay、Sang Rhee 和 Checed Rodgers 在内的学生，发现了书中的问题并对代码做出了贡献。另外，特别感谢笔者的妻子兼合作伙伴 Christine，她的仔细编辑和建议让本书改善良多。

Douglas Comer

关于作者

Operating System Design: The Xinu Approach, Second Edition

Douglas Comer 是美国普度大学（Purdue University）计算机科学系的杰出教授，国际公认的计算机网络、TCP/IP 协议、Internet 和操作系统设计的专家。Comer 发表论文无数，出版专著多部，是一位为研究和教育而开发课程体系和实验项目的先驱。

作为一个多产作家，Comer 博士的书被翻译成 16 种语言，广泛应用于全世界的计算机科学、工程和工商行政管理等学校院系和相关行业。Comer 博士划时代的三卷巨著《Internetworking with TCP/IP》对网络和网络教育产生了革命性影响。他所编写的教科书和富有创意的实验手册已经塑造和继续塑造着研究生和本科生的教学课程体系。

Comer 博士撰写书籍时的精确和洞见反映出他在计算机系统领域的深厚背景。他的研究横跨硬件和软件。他创建了 Xinu 操作系统，编写了设备驱动程序，并为传统计算机和网络处理器实现了网络协议软件。Comer 博士的研究成果已经应用到工业界的各种产品中。

Comer 博士创建和主讲的课程包括“网络协议”“操作系统”“计算机体系结构”，其听众既有大学生和学术界同仁，也有工业界的工程师。他的创新教育实验让他和他的学生能够设计和实现大型复杂系统的原型，并对结果原型的性能进行度量。Comer 博士长期在企业、大学和会议上讲课和演说，还为工业界提供咨询服务，以帮助他们设计计算机网络和系统。

20 多年来，Comer 教授担任研究期刊《Software—Practice and Experience》的主编。在普度大学停薪留职期间，他在思科（Cisco）公司担任研究副总裁。Comer 博士是 ACM 院士、普度教育学院院士和无数奖项的获得者，其中包括 Usenix 终身成就奖。

关于 Comer 博士的更多信息可在如下网站找到：

www.cs.purdue.edu/people/comer

关于 Comer 博士所著书籍的更多信息可在如下网站找到：

www.comerbooks.com

出版者的话
译者序
前言
关于作者

第 1 章 引言和概述 ……1

1.1 操作系统……1
1.2 本书的研究方法……2
1.3 分层设计……2
1.4 Xinu 操作系统……4
1.5 操作系统的界定……4
1.6 从外部看操作系统……5
1.7 其他章节概要……6
1.8 观点……6
1.9 总结……7
练习……7

第 2 章 并发执行与操作系统服务 ……8

2.1 引言……8
2.2 多活动的编程模型……8
2.3 操作系统服务……9
2.4 并发处理的概念和术语……9
2.5 串行程序和并发程序的区别……11
2.6 多个进程共享同一段代码……12
2.7 进程退出与进程终止……14
2.8 共享内存、竞争条件和同步……14
2.9 信号量与互斥……18
2.10 Xinu 中的类型命名方法……19
2.11 使用 kputc 和 kprintf 进行操作系统的调试……20
2.12 观点……20
2.13 总结……21
练习……21

第 3 章 硬件与运行时环境概述 ……22

3.1 引言……22
3.2 开发平台的物理和逻辑架构……22
3.3 指令集……23
3.4 通用寄存器……23
3.5 I/O 总线和存 – 取范例……24
3.6 DMA 机制……25
3.7 总线地址空间……25
3.8 总线启动和配置……26
3.9 函数调用约定和运行时栈……26
3.10 中断和中断处理……28
3.11 中断向量……29
3.12 异常向量和异常处理……29
3.13 时钟硬件……29
3.14 串行通信……30
3.15 轮询与中断驱动 I/O……30
3.16 存储布局……30
3.17 内存保护……31
3.18 硬件细节和片上系统体系结构……31
3.19 观点……31
3.20 硬件参考资料……32
练习……32

第 4 章 链表与队列操作 ……33

4.1 引言……33
4.2 进程链表的统一数据结构……33
4.3 简洁的链表数据结构……34
4.4 队列数据结构的实现……35
4.5 内联队列操作函数……36
4.6 获取链表中进程的基础函数……37
4.7 FIFO 队列操作……38
4.8 优先级队列的操作……40
4.9 链表初始化……42

4.10 观点 ····· 43
4.11 总结 ····· 43
练习 ····· 43

第 5 章 调度和上下文切换 ····· 45

5.1 引言 ····· 45
5.2 进程表 ····· 45
5.3 进程状态 ····· 47
5.4 就绪和当前状态 ····· 48
5.5 调度策略 ····· 48
5.6 调度的实现 ····· 49
5.7 推迟重新调度 ····· 52
5.8 上下文切换的实现 ····· 52
5.9 内存中保存的状态 ····· 52
5.10 上下文切换操作 ····· 53
5.11 重新启动进程执行的地址 ····· 56
5.12 并发执行和空进程 ····· 57
5.13 使进程就绪和调度常量 ····· 57
5.14 其他进程调度算法 ····· 58
5.15 观点 ····· 58
5.16 总结 ····· 59
练习 ····· 59

第 6 章 更多进程管理 ····· 60

6.1 引言 ····· 60
6.2 进程挂起和恢复 ····· 60
6.3 自我挂起和信息隐藏 ····· 60
6.4 系统调用 ····· 61
6.5 禁止和恢复中断 ····· 62
6.6 系统调用模板 ····· 63
6.7 系统调用返回值 SYSERR 和 OK ····· 63
6.8 挂起的实现 ····· 64
6.9 挂起当前进程 ····· 65
6.10 suspend 函数的返回值 ····· 65
6.11 进程终止和进程退出 ····· 66
6.12 进程创建 ····· 68
6.13 其他进程管理函数 ····· 72
6.14 总结 ····· 74
练习 ····· 74

第 7 章 协调并发进程 ····· 76

7.1 引言 ····· 76
7.2 进程同步的必要性 ····· 76
7.3 计数信号量的概念 ····· 77
7.4 避免忙等待 ····· 77
7.5 信号量策略和进程选择 ····· 77
7.6 等待状态 ····· 78
7.7 信号量数据结构 ····· 79
7.8 系统调用 wait ····· 79
7.9 系统调用 signal ····· 80
7.10 静态和动态信号量分配 ····· 81
7.11 动态信号量的实现示例 ····· 82
7.12 信号量删除 ····· 83
7.13 信号量重置 ····· 84
7.14 并行处理器（多核）之间的协调 ····· 85
7.15 观点 ····· 86
7.16 总结 ····· 86
练习 ····· 87

第 8 章 消息传递 ····· 88

8.1 引言 ····· 88
8.2 两种类型的消息传递服务 ····· 88
8.3 消息使用资源的限制 ····· 89
8.4 消息传递函数和状态转换 ····· 89
8.5 send 的实现 ····· 90
8.6 receive 的实现 ····· 91
8.7 非阻塞消息接收的实现 ····· 92
8.8 观点 ····· 92
8.9 总结 ····· 92
练习 ····· 93

第 9 章 基本内存管理 ····· 94

9.1 引言 ····· 94
9.2 内存的类型 ····· 94
9.3 重量级进程的定义 ····· 95
9.4 示例系统的内存管理 ····· 95

9.5 程序段和内存区域 …… 95
9.6 动态内存分配 …… 96
9.7 底层内存管理器的设计 …… 97
9.8 分配策略和内存持久性 …… 97
9.9 追踪空闲内存 …… 98
9.10 底层内存管理的实现 …… 98
9.11 使用空闲内存的数据结构定义 …… 99
9.12 分配堆存储 …… 100
9.13 分配栈存储 …… 102
9.14 堆和栈存储的释放 …… 103
9.15 观点 …… 105
9.16 总结 …… 106
练习 …… 106

第 10 章 高级内存管理和虚拟内存 … 107

10.1 引言 …… 107
10.2 分区空间分配 …… 107
10.3 缓冲池 …… 108
10.4 分配缓冲区 …… 108
10.5 将缓冲区返还给缓冲池 …… 110
10.6 创建缓冲池 …… 111
10.7 初始化缓冲池表 …… 112
10.8 虚拟内存和内存复用 …… 113
10.9 实地址空间和虚地址空间 …… 113
10.10 支持按需分页的硬件 …… 114
10.11 使用页表的地址转换 …… 114
10.12 页表项中的元数据 …… 115
10.13 按需分页以及设计上的问题 …… 116
10.14 页面替换和全局时钟算法 …… 116
10.15 观点 …… 117
10.16 总结 …… 117
练习 …… 118

第 11 章 高层消息传递 …… 119

11.1 引言 …… 119
11.2 进程间通信端口 …… 119
11.3 端口实现 …… 119
11.4 端口表初始化 …… 120
11.5 端口创建 …… 121
11.6 向端口发送消息 …… 122
11.7 从端口接收消息 …… 124
11.8 端口的删除和重置 …… 125
11.9 观点 …… 128
11.10 总结 …… 128
练习 …… 128

第 12 章 中断处理 …… 130

12.1 引言 …… 130
12.2 中断的优点 …… 130
12.3 中断处理 …… 130
12.4 中断向量 …… 131
12.5 中断和异常集成 …… 131
12.6 使用代码的 ARM 异常向量 …… 132
12.7 设备中断向量号的分配 …… 135
12.8 中断分派 …… 136
12.9 中断的软件结构 …… 137
12.10 禁止中断 …… 139
12.11 中断代码调用函数的限制 …… 140
12.12 中断过程中重新调度的必要性 …… 140
12.13 中断过程中的重新调度 …… 140
12.14 观点 …… 141
12.15 总结 …… 142
练习 …… 142

第 13 章 实时时钟管理 …… 143

13.1 引言 …… 143
13.2 定时事件 …… 143
13.3 实时时钟和计时器硬件 …… 143
13.4 实时时钟中断处理 …… 144
13.5 延时与抢占 …… 145
13.6 抢占的实现 …… 145
13.7 使用增量链表对延迟进行有效管理 …… 146
13.8 增量链表的实现 …… 147
13.9 将进程转入睡眠 …… 148

13.10 定时消息接收 …… 150
13.11 唤醒睡眠进程 …… 154
13.12 时钟中断处理 …… 154
13.13 时钟初始化 …… 156
13.14 观点 …… 159
13.15 总结 …… 159
练习 …… 159

第 14 章 设备无关的 I/O …… 161

14.1 引言 …… 161
14.2 I/O 和设备驱动的概念结构 …… 161
14.3 接口抽象和驱动抽象 …… 162
14.4 I/O 接口示例 …… 163
14.5 打开 – 读 – 写 – 关闭范例 …… 163
14.6 绑定 I/O 操作和设备名 …… 164
14.7 Xinu 中的设备名 …… 164
14.8 设备转换表概念 …… 165
14.9 设备的多个副本和共享驱动 …… 166
14.10 高层 I/O 操作的实现 …… 168
14.11 其他高层 I/O 函数 …… 169
14.12 打开、关闭和引用计数 …… 172
14.13 devtab 中的空条目和错误条目 …… 174
14.14 I/O 系统的初始化 …… 174
14.15 观点 …… 178
14.16 总结 …… 179
练习 …… 179

第 15 章 设备驱动示例 …… 180

15.1 引言 …… 180
15.2 使用 UART 硬件进行串行通信 …… 180
15.3 tty 抽象 …… 180
15.4 tty 设备驱动的组织结构 …… 181
15.5 请求队列和缓冲区 …… 182
15.6 上半部和下半部的同步 …… 183
15.7 UART 硬件 FIFO 与驱动设计 …… 184
15.8 控制块的概念 …… 184
15.9 tty 控制块和数据声明 …… 184
15.10 次设备号 …… 186
15.11 上半部 tty 字符输入 (ttygetc) …… 187
15.12 上半部 tty 读取函数 (ttyread) …… 188
15.13 上半部 tty 字符输出 (ttyputc) …… 189
15.14 开始输出 (ttykickout) …… 190
15.15 上半部 tty 多字符输出 (ttywrite) …… 191
15.16 下半部 tty 驱动函数 (ttyhandler) …… 192
15.17 输出中断处理 (ttyhandle_out) …… 194
15.18 tty 输入处理 (ttyhandle_in) …… 196
15.19 tty 控制块初始化 (ttyinit) …… 202
15.20 设备驱动控制 (ttycontrol) …… 204
15.21 观点 …… 205
15.22 总结 …… 205
练习 …… 206

第 16 章 DMA 设备和驱动 (以太网) …… 207

16.1 引言 …… 207
16.2 直接内存访问和缓冲区 …… 207
16.3 多个缓冲区和缓冲区环 …… 207
16.4 使用 DMA 的以太网驱动示例 …… 208
16.5 设备的硬件定义和常量 …… 209
16.6 环和内存缓冲区 …… 211
16.7 以太网控制块的定义 …… 213
16.8 设备和驱动初始化 …… 215
16.9 从以太网设备读取数据包 …… 221
16.10 向以太网设备写入数据包 …… 223
16.11 以太网设备的中断处理 …… 225
16.12 以太网控制函数 …… 228
16.13 观点 …… 229
16.14 总结 …… 229
练习 …… 229

第 17 章　最小互联网协议栈 …… 230

17.1　引言 …… 230
17.2　所需的功能 …… 230
17.3　同步会话、超时和网络处理进程 …… 231
17.4　设计的影响 …… 232
17.5　ARP 函数 …… 232
17.6　网络数据包的定义 …… 241
17.7　网络输入进程 …… 242
17.8　IP 的相关定义 …… 245
17.9　IP 函数 …… 246
17.10　UDP 表的定义 …… 255
17.11　UDP 函数 …… 256
17.12　互联网控制报文协议 …… 267
17.13　动态主机配置协议 …… 268
17.14　观点 …… 275
17.15　总结 …… 275
练习 …… 275

第 18 章　远程磁盘驱动 …… 277

18.1　引言 …… 277
18.2　磁盘抽象 …… 277
18.3　磁盘驱动支持的操作 …… 277
18.4　块传输和高层 I/O 函数 …… 277
18.5　远程磁盘范例 …… 278
18.6　高速缓存的重要概念 …… 278
18.7　磁盘操作的语义 …… 279
18.8　驱动数据结构的定义 …… 280
18.9　驱动初始化（rdsinit） …… 284
18.10　上半部打开函数（rdsopen） …… 287
18.11　远程通信函数（rdscomm） …… 289
18.12　上半部写函数（rdswrite） …… 291
18.13　上半部读函数（rdsread） …… 293
18.14　刷新挂起的请求 …… 296
18.15　上半部控制函数（rdscontrol） …… 297
18.16　分配磁盘缓冲区（rdsbufalloc） …… 299
18.17　上半部关闭函数（rdsclose） …… 300
18.18　下半部通信进程（rdsprocess） …… 302
18.19　观点 …… 306
18.20　总结 …… 306
练习 …… 306

第 19 章　文件系统 …… 308

19.1　什么是文件系统 …… 308
19.2　文件操作的示例集 …… 308
19.3　本地文件系统的设计 …… 309
19.4　Xinu 文件系统的数据结构 …… 309
19.5　索引管理器的实现 …… 310
19.6　清空索引块（lfibclear） …… 314
19.7　获取索引块（lfibget） …… 315
19.8　存储索引块（lfibput） …… 315
19.9　从空闲链表中分配索引块（lfiballoc） …… 316
19.10　从空闲链表中分配数据块（lfdballoc） …… 317
19.11　使用设备无关的 I/O 函数进行文件操作 …… 319
19.12　文件系统的设备配置和函数名称 …… 320
19.13　本地文件系统打开函数（lfsopen） …… 320
19.14　关闭文件伪设备（lflclose） …… 326
19.15　刷新磁盘中的数据（lfflush） …… 328
19.16　文件的批量传输函数（lflwrite、lflread） …… 328
19.17　在文件中查找新位置（lflseek） …… 330
19.18　从文件中提取一字节（lflgetc） …… 331
19.19　改变文件中的一字节（lflputc） …… 332
19.20　载入索引块和数据块（lfsetup） …… 334
19.21　主文件系统设备的初始化（lfsinit） …… 337
19.22　伪设备的初始化（lflinit） …… 338

19.23 文件截断（lftruncate）…… 339
19.24 初始文件系统的创建（lfscreate）…… 341
19.25 观点 …… 343
19.26 总结 …… 343
练习 …… 343

第 20 章 远程文件机制 …… 345

20.1 引言 …… 345
20.2 远程文件访问 …… 345
20.3 远程文件语义 …… 345
20.4 远程文件设计和消息 …… 346
20.5 远程文件服务器通信（rfscomm）…… 352
20.6 发送基本消息（rfsndmsg）…… 354
20.7 网络字节序 …… 355
20.8 使用设备范例的远程文件系统 …… 355
20.9 打开远程文件（rfsopen）…… 356
20.10 检查文件模式（rfsgetmode）…… 359
20.11 关闭远程文件（rflclose）…… 360
20.12 读远程文件（rflread）…… 361
20.13 写远程文件（rflwrite）…… 363
20.14 远程文件的定位（rflseek）…… 365
20.15 远程文件单字符 I/O（rflgetc、rflputc）…… 366
20.16 远程文件系统控制函数（rfscontrol）…… 367
20.17 初始化远程文件系统（rfsinit、rflinit）…… 370
20.18 观点 …… 372
20.19 总结 …… 372
练习 …… 372

第 21 章 句法名字空间 …… 374

21.1 引言 …… 374
21.2 透明与名字空间抽象 …… 374
21.3 多种命名方案 …… 375
21.4 命名系统设计的其他方案 …… 376
21.5 基于句法的名字空间 …… 376
21.6 模式和替换 …… 376
21.7 前缀模式 …… 377
21.8 名字空间的实现 …… 377
21.9 名字空间的数据结构和常量 …… 377
21.10 增加名字空间前缀表的映射 …… 378
21.11 使用前缀表进行名字映射 …… 379
21.12 打开命名文件 …… 383
21.13 名字空间初始化 …… 383
21.14 对前缀表中的项进行排序 …… 386
21.15 选择逻辑名字空间 …… 386
21.16 默认层次和空前缀 …… 387
21.17 额外的对象操作函数 …… 387
21.18 名字空间方法的优点和限制 …… 388
21.19 广义模式 …… 388
21.20 观点 …… 389
21.21 总结 …… 389
练习 …… 390

第 22 章 系统初始化 …… 391

22.1 引言 …… 391
22.2 引导程序：从零开始 …… 391
22.3 一个通过网络启动的例子 …… 392
22.4 操作系统初始化 …… 392
22.5 Xinu 初始化 …… 393
22.6 Xinu 系统启动 …… 395
22.7 从程序转化为进程 …… 399
22.8 观点 …… 399
22.9 总结 …… 399
练习 …… 400

第 23 章 子系统初始化和内存标记 … 401

23.1 引言 …… 401
23.2 自初始化模块 …… 401
23.3 并发系统中的自初始化模块 …… 402
23.4 重新启动后的自初始化 …… 403
23.5 使用登录号初始化 …… 404
23.6 广义内存标记方案 …… 405

23.7 内存标记系统的数据声明 406
23.8 标记的实现 407
23.9 观点 408
23.10 总结 408
练习 408

第 24 章 异常处理 409

24.1 引言 409
24.2 术语：故障、检测、陷阱和异常 409
24.3 向量异常和可屏蔽中断 409
24.4 异常的类型 410
24.5 处理异常 410
24.6 异常向量初始化 411
24.7 面对灾难时的 panic 411
24.8 panic 函数的实现 411
24.9 观点 412
24.10 总结 412
练习 412

第 25 章 系统配置 413

25.1 引言 413
25.2 多重配置的需求 413
25.3 Xinu 系统配置 414
25.4 Xinu 配置文件的内容 414
25.5 计算次设备号 416
25.6 配置 Xinu 系统的步骤 417
25.7 观点 417
25.8 总结 417
练习 417

第 26 章 一个用户接口例子：Xinu shell 419

26.1 引言 419
26.2 什么是用户接口 419
26.3 命令和设计原则 419
26.4 一个简化 shell 的设计决策 420
26.5 shell 的组织和操作 420
26.6 词法符号的定义 421
26.7 命令行语法的定义 421
26.8 Xinu shell 的实现 422
26.9 符号的存储 424
26.10 词法分析器代码 424
26.11 命令解释器的核心 428
26.12 命令名查询和内部处理 434
26.13 传递给命令的参数 435
26.14 向外部命令传递参数 435
26.15 I/O 重定向 438
26.16 命令函数（sleep）的例子 439
26.17 观点 440
26.18 总结 441
练习 441

附录 1 操作系统移植 443

附录 2 Xinu 设计注解 450

索引 454

引言和概述

我们小小的系统也有风光的时刻。

——Alfred，Lord Tennyson

1.1 操作系统

每一个智能设备和计算机系统中都隐藏着这么一类软件，它们控制处理过程、管理资源以及与显示屏、计算机网络、磁盘和打印机等外部设备进行通信。总的来说，这些进行控制和协调工作的代码通常叫作*执行器*、*监视器*、*任务管理器*或者*内核*；而我们将用一个更宽泛的术语来概括，即*操作系统*。

计算机操作系统是人类创造的最复杂的东西之一：计算机操作系统允许多个计算进程和用户同时共享一个处理器，保护数据免受未经授权的访问，并保持独立输入 / 输出（I/O）设备的正确运行。操作系统提供的高层服务都是通过执行复杂的底层硬件指令实现的。有趣的是，操作系统并不是从外部控制计算机的独立机制——它由软件组成，且这些软件由执行应用程序的同一处理器执行。事实上，处理器运行应用程序的时候，是不能执行操作系统的，反之亦然。

操作系统总在应用程序运行结束后重新夺回控制权这一协调机制使得操作系统的设计变得非常复杂。操作系统最令人印象深刻的特征就是所提供的服务和底层硬件两者功能的不同：操作系统在底层硬件上提供高层服务。随着本书内容的推进，读者就会理解底层硬件是非常粗陋的，以至于即使是一个简单的设备，诸如用于键盘或鼠标的串行 I/O 设备，系统软件也要做很多处理。而其中的哲学原理很简单，即操作系统提供的抽象应该让编程更加容易，而不是反映底层硬件设备的抽象。因此，我们得出结论：

> 设计操作系统时，应该隐藏底层硬件的细节，并创建一个为应用程序提供高层服务的抽象机器。

操作系统的设计并不是人们所熟知的工艺。起初，由于计算机的稀少和价格的高昂，只有很少程序员有机会从事操作系统相关工作。而现在，先进的微电子技术降低了制造成本，推动了个人计算机的普及，操作系统成为一种商品，只有极少数程序员有必要从事操作系统方面的工作。有趣的是，由于微处理器变得非常便宜，大多数电子设备现在都是由可编程处理器构建，而不是从分离的逻辑元件构建得到。因此，设计与实现微处理器和微控制器的软件系统不再是专家的专利，它已成为一个称职的系统程序员必须胜任的技术。

幸运的是，随着新计算机生产技术的发展，我们对操作系统的理解也在不断提高。研究人员已经探索出基本原理，形成了设计原则，定义了基本组件，并设计了这些组件一起工作的方式。更重要的是，研究人员还定义了一系列的抽象，如文件和当前进程（这些抽象概念对于所有操作系统都是相同的），并且已经找到实现这些抽象的有效方式。最后，我们掌握