

PEARSON

深入理解 UNIX系统内核

[美] Uresh Vahalia 著 李雨 薛磊 黄庆新 译

UNIX Internals
The New Frontiers

UNIX[®]
INTERNALS
THE NEW FRONTIERS

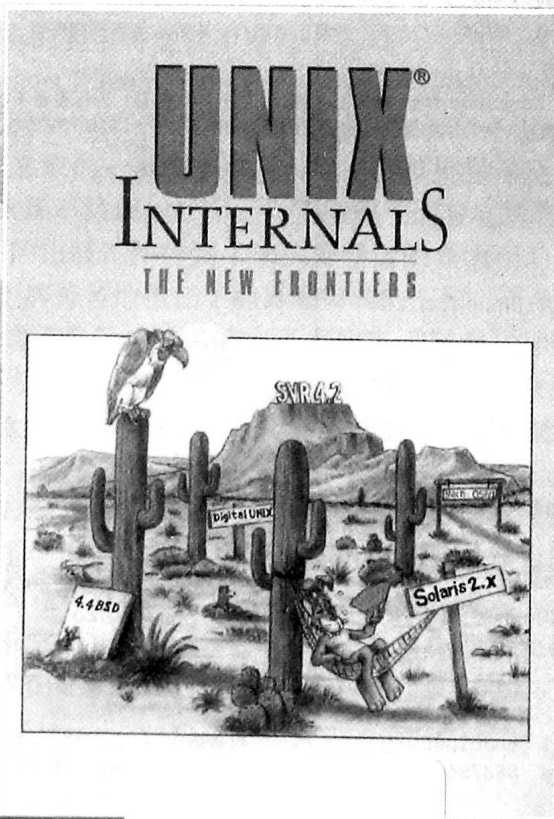


URESH VAHALIA

计 算 机 科 学 丛

深入理解 UNIX系统内核

[美] Uresh Vahalia 著 李雨 薛磊 黄庆新 译



机械工业出版社
China Machine Press

图书在版编目 (CIP) 数据

深入理解 UNIX 系统内核 / (美) 瓦哈利亚 (Uresh Vahalia) 著; 李雨, 薛磊, 黄庆新译.
—北京: 机械工业出版社, 2015.2

(计算机科学丛书)

书名原文: UNIX Internals: The New Frontiers

ISBN 978-7-111-49145-3

I. 深… II. ①瓦… ②李… ③薛… ④黄… III. UNIX 操作系统 IV. TP316.81

中国版本图书馆 CIP 数据核字 (2015) 第 012268 号

本书版权登记号: 图字: 01-2011-3367

UNIX Internals: The New Frontiers, 978-0-13-101908-9, by Uresh Vahalia, published by Pearson Education, Inc., Copyright © 1996.

All rights reserved. No part of this book may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording or by any information storage retrieval system, without permission from Pearson Education, Inc.

Chinese simplified language edition published by Pearson Education Asia Ltd., and China Machine Press Copyright © 2015.

本书中文简体字版由 Pearson Education (培生教育出版集团) 授权机械工业出版社在中华人民共和国境内 (不包括中国台湾地区和香港、澳门特别行政区) 独家出版发行。未经出版者书面许可, 不得以任何方式抄袭、复制或节录本书中的任何部分。

本书封底贴有 Pearson Education (培生教育出版集团) 激光防伪标签, 无标签者不得销售。

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: 吴 怡

责任校对: 董纪丽

印 刷: 北京瑞德印刷有限公司

版 次: 2015 年 5 月第 1 版第 1 次印刷

开 本: 185mm × 260mm 1/16

印 张: 30.25

书 号: ISBN 978-7-111-49145-3

定 价: 119.00 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 88361066

投稿热线: (010) 88379604

购书热线: (010) 68326294 88379649 68995259

读者信箱: hzjsj@hzbook.com

版权所有·侵权必究

封底无防伪标均为盗版

本书法律顾问: 北京大成律师事务所 韩光 / 邹晓东

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，不仅擘划了研究的范畴，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，从他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, Alfred 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”已经出版了近两百个品种，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，我们的联系方式如下：

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



我与这本书结缘还是 2000 年年初在学校前书店的一次偶遇。当时刚刚从《电脑报》对电影《泰坦尼克号》的介绍中得知有一个开源操作系统叫 Linux，便产生了一股莫名的兴奋，开始千方百计地搜寻相关材料。可惜当时几乎没有机会访问互联网，国内又没有像样的 Linux 专业书籍。遇到这本书并买回家的原因只是因为书名包括“UNIX”，而且是讲“内幕”的，但只记得当时是看得N头雾水，小白完全理解不了什么是 LWP。

时光如白驹过隙，一晃过了 10 年，非科班出身的小白已经做了码农 6、7 年，做 Linux 内核工作也 3 年了，整理书架时再次偶然发现了这本略带灰尘的奇书，勾起了无数回忆。再度翻起后，往往每看几十页竟有打通任督二脉的感觉。去年，偶然得知机械工业出版社要再次翻译这本书，不禁心头为之一热，虽有珠玉在前，仍然挡不住跃跃欲试的冲动。

原著的序言中提到，本书刚刚出版时在操作系统领域中就独树一帜，时至今日依然如此，鲜有其他著作能出其右。我想一定有读者会觉得对于这样一本出版于上个世纪的“UNIX 史记”，今天再拎出来，除了考古之外还有意义吗？对于 Linux/Unix 的铁杆粉丝们，本书的书名就是意义！对于其他读者，我也试试举出如下几个理由。

首先，Linux 目前的流行是毋庸置疑的，但是如果你非常质疑 Linux 为什么是现在这个模样，那么本书应该能解答你的部分疑惑。Linux 与 UNIX 的亲密关系决定了两者在设计和实现上有多方面的相似性。假如你看完了第 14 章，而没有联想到 Linux 内核中的 VMA、page cache 等概念，那一定是有什么地方不对 :) 本书以追根溯源的方式探讨了现代 Linux 系统设计的 WHY，市面上太多的书更加关注在 WHAT 和 HOW 上。在阅读本书时，你会有许多这样的感悟：“原来那样做是有问题的！”

其次，二三十年前设计实现本书中这些软件大师们所面对的问题，今天的无数工程师仍然在其他场合不断地再次遇到，尤其是在那些性能攸关的领域，比如在 C100K 甚至 C10M 的场景下。例如，内存池是一种通过改进空间局部性提高性能的常见手段。在第 12 章中，读者可以看到 UNIX 内核在不同的时空下，是如何从简单的资源映射分配器演化出完善 Slab 分配器的，其间的设计权衡经验和对多处理器环境的思考，在许多现在流行软件的设计和实现上仍然闪闪发光。

其实这本书的亮点远不只以上两个。再举个例子，读者一定了解什么是面向对象编程，并且听说过设计模式云云。在本书里你可以看到不用 C++ 的 OOP，而公元前的设计模式遍布在 UNIX 内核的很多核心子系统中，这两点在现代 Linux 内核中依然如故。类似本书这样的史籍还有《链接器和加载器》、《现代体系结构上的 UNIX 系统：内核程序员的 SMP 和 Caching 技术》等。

在以微博、Twitter 为代表的互联网快餐阅读大潮下，许多人可以每天看几十分钟的肥皂文章，却难以坚持读完一份 110 页的干货“*What every programmer should know about the memory*”，即使后者的营养足以秒杀 80% 以上的互联网内容。这样，选择一本如此厚的历史书确实需要挑战自己的耐力，那么，你，敢不敢呢？

本书由三名内核开发人员共同翻译，黄庆新负责翻译第 1 ~ 7 章，薛磊负责翻译第

8、9、11、16、17 章，李雨负责翻译第 10、12、13、14、15 章。非常感谢机械工业出版社的吴怡编辑对我们的信任，使得我们三个有幸翻译这本慕名已久的奇书。在本书成书期间，吴怡编辑一次又一次地容忍了译稿的延期，还不断地在解答和解决我们三个在初次翻译中所遇到的问题和困难。我在阿里云飞天团队的同事黄江伟认真检查了内存管理章节，在正确性、严谨性等方面提出了大量有份量的改进意见，对译文质量的提升有很大贡献。在翻译期间，我们三个人要么步入了婚姻的殿堂，要么即将升级为父亲，也都切身体会到翻译是一件非常消耗时间和精力的工作，这期间如果没有家人的支持是难以想象的，感谢你们的理解和耐心！

最后，经过十年 IT 领域的耕耘，译者深知自己水平有限，虽然原书中的一些错误已经在译文中纠正，但本书中的错误和疏漏仍然在所难免，敬请读者指正，我们的微博是：

李 雨：<http://weibo.com/cloctick>

薛 磊：<http://weibo.com/douzhr>

黄庆新：<http://weibo.com/huangqingxin>

李 雨

2014 年 11 月于北京

Peter H. Salus

《Computer Systems》杂志主编

这个世界上，UNIX 的种类比大多数品牌冰淇淋的口味还要多。尽管整个行业推动了 X/Open 及其成员的发展，但是单一 UNIX 规范与当初宣称的目标渐行渐远。事实上，这个目标可能并不是那么重要，因为自从 Interactive System 提供了第一个商业版 UNIX 系统、Whitesmiths 提供了第一个 UNIX 克隆之后，用户社区就一直在面对各种平台上的不同实现。

UNIX 在 1969 年问世，还未满 10 岁时，各种版本就开始泛滥了。在 UNIX 20 岁之前，形成了两个阵营：自由软件基金会（Open Software Foundation）和 UNIX 国际联盟（UNIX International），以及大量的其他版本。技术上的两个主要流派分别为 AT&T（现为 Novell）和加州大学伯克利分校。Maurice Bach [Bach 86]、Sam Leffler、Kirk McKusick、Mike Karels 和 John Quarterman [Leff 89] 对这些 UNIX 进行了很好的介绍。

然而，没有一本书可以提供关于各种 UNIX 操作系统实现的总览式介绍，而本书作者 Uresh Vahalia 填补了这个空白。他对 SVR4、4.4BSD 和 Mach 技术内幕的阐述是前所未有的，书中甚至还呈现了对 Solaris、SunOS、Digital UNIX 和 HP-UX 的详尽探讨。

他的讲解非常清晰，而且不像其他某些作者那样对某种 UNIX 有所偏爱。最近，像 Linux 这种相对较新的 UNIX 克隆还在不断衍生新的 UNIX 变种，甚至伯克利 UNIX 的衍生版本也开始了新的分裂，像这样一本揭示 UNIX 内部技术并以推动 UNIX 发展为动机的著作，正是生逢其时。

1972 年 6 月，Ken Thompson 和 Denis Ritchie 出版了《UNIX Programmer's Manual》第 2 版。两位作者在这本书的引言中提到：“UNIX 的安装数量增长到了 10 个，已经超出预期。”恐怕他们根本没有预料到现在所发生的情况。

我曾经在别的地方看到过对 UNIX 系统历史的介绍 [Salu94]，但是 Vahalia 在本书中对各种 UNIX 进行了真正独特和全面的对比剖析。

参考文献

- [Bach 86] Bach, M.J., *The Design of the UNIX Operating System*, Prentice-Hall, Englewood Cliffs, NJ, 1986.
- [Leff 89] Leffler, S.J., McKusick, M.K., Karels, M.J., and Quarterman, J.S., *The Design and Implementation of the 4.3 BSD UNIX Operating System*, Addison-Wesley, Reading, MA, 1989.
- [Salu 94] Salus, P.H., *A Quarter Century of UNIX*, Addison-Wesley, Reading, MA, 1994.
- [Thom 72] Thompson, K., and Ritchie, D.M., *UNIX Programmer's Manual*, Second Edition, Bell Telephone Laboratories, Murray Hill, NJ, 1972.

从 20 世纪 70 年代早期开始，UNIX 系统经历了相当大的变化。它从一个小的、实验室性质的由贝尔实验室几乎免费分发的操作系统开始，逐渐成长，直到当前拥有着一群不断增长的忠实的拥护者。多年来，它从众多的学术界和行业成员中吸收了很多贡献，忍受着在其所有权以及标准化上的争执，并且进化为稳定的、成熟的操作系统。现在，有许多商业的和研究用的 UNIX 变种系统，每种变种系统虽然都不同，但是又足够相似，可以作为 UNIX 系统家族中的一种。UNIX 程序员对一种 UNIX 变种的使用经验可以用在很多硬件平台以及其他 UNIX 变种上。

很多图书已经介绍过 UNIX 系统的不同特性，但是大多数都只是介绍诸如命令行或编程接口这些用户可见的方面，很少能介绍 UNIX 系统的内部实现。UNIX 系统的内部实现主要是指其核心组成的部分，也就是 UNIX 内核研究。迄今为止，每本 UNIX 系统内部实现书籍只针对某一种 UNIX 的发行版本。比如，Bach 的《*The Design of the UNIX Operating System*》[Bach 86] 是介绍 System V Release 2 (SVR2) 版本内核的标志性书籍。Leffler 等著的《*The Design and Implementation of 4.3BSD UNIX Operating System*》[Leff 88] 由一些主要的操作系统设计者来详尽介绍 4.3BSD 版本内核。Goodheart 和 Cox 的《*The Magic Garden Explained*》著作详细介绍了 System V Release 4.0 (SVR4) 版本的内核。

设计视角

本书从系统设计角度来审视 UNIX 内核。介绍了一些主流的商业性和研究性的 UNIX 变种。针对内核中的每个模块，本书探索其结构和设计，解释主流 UNIX 系统如何选择具体模块的实现方法，以及每种方法的优缺点。通过这样的对比使本书有着独特的视角，并允许读者从批判性的角度来检查系统的设计。在研究一个操作系统时，关注其优缺点是很重要的一部分。本书将通过分析一些可选的方法来介绍每种系统的优缺点。

UNIX 变种

虽然本书最关注 SVR4.2 的内核实现，但是也详细探索了 4.4BSD、Solaris 2.x、Mach 以及 Digital Unix 系统。而且本书介绍了一些其他变种的有趣特性，这主要包括一些没有进入商业发行版的研究性变种。本书分析了从 20 世纪 80 年代中期到 90 年代中期 UNIX 系统的主要发展。为了完整性，本书同时包含了对传统 UNIX 功能和实现的简要介绍。在必要时，本书还提供了一个发展记录，以传统方案为起始，分析其缺点和局限并介绍现代的解决方案。

阅读对象

本书适合作为大学教材或作为专业参考书。作为大学教材，本书适用于高年级本科

生或研究生操作系统课程。本书并非一本介绍性书籍，而且不会说明诸如内核、线程以及虚拟内存这样的概念性知识。每章都包含一系列精心设计的练习题，用于进一步地思考和研究系统设计。许多练习题都是开放性的，并且需要学生进行一些额外的阅读研究。每章也有一个详尽的参考文献清单，通过该清单可以进一步阅读有关内容。

本书也可用作操作系统开发人员、应用程序开发人员以及系统管理员的专业参考书籍。操作系统设计者和架构师可以研究相近系统的内核结构、评估不同设计的优缺点以及使用本书中的一些想法来开发下一代操作系统。应用程序开发人员可以使用操作系统内核的知识，更好地利用其特征来开发更加高效的程序。系统管理员可以在理解不同参数和使用模式对系统行为的影响之后，更加有效地配置和调整系统。

组织结构

第 1 章追溯了 UNIX 系统的演变并分析了影响系统主要变化的因素。第 2 章到第 7 章介绍了进程子系统。其中第 2 章介绍在传统 UNIX 系统中（SVR3、4.3BSD 和早期变种）的进程和内核架构。第 3 章到第 7 章介绍现代系统（SVR4、4.4BSD、Solaris 2.x 以及 Digital Unix）的特性。第 3 章讨论线程及其在内核和用户库中如何实现。第 4 章介绍信号、作业控制以及登录会话管理。第 5 章介绍 UNIX 调度器和对实时应用程序的支持。第 6 章研究进程间通信的技术（IPC），包括 System V IPC 的特性集。其中还介绍了 Mach 系统架构，该系统使用 IPC 作为最基本的原语来构造内核。第 7 章讨论在现代单处理器和多处理器系统中应用的同步框架。

接下来的四章介绍了文件系统。第 8 章介绍用户可见的文件系统接口，以及定义了内核和文件系统交互的 vnode/vfs 接口。第 9 章提供一些具体的文件系统实现细节，包含原始的 System V 文件系统（s5fs）、伯克利快速文件系统（FFS）以及一些小的、利用 vnode/vfs 接口提供服务的特殊用途的文件系统。第 10 章介绍一些分布式文件系统，比如太阳微系统公司的网络文件系统（NFS）、AT&T 的远程文件共享（RFS）、卡内基梅隆大学的 Andrew 文件系统（AFS）以及 Transarc 公司的分布式文件系统（DFS）。第 11 章介绍一些使用日志提供更高可靠性和性能的高级文件系统，同时介绍一种基于可堆叠 vnode 层的新文件系统框架。

第 12 章~第 15 章介绍内存管理。第 12 章讨论内核内存分配并介绍一些有趣的分配算法。第 13 章介绍虚拟内存的概念并使用 4.3BSD 的实现来说明一些问题。第 14 章介绍 SVR4 和 Solaris 的虚拟内存架构。第 15 章介绍 Mach 和 4.4BSD 内存模型，同时分析诸如旁路转换缓冲区和虚拟地址缓存等硬件特性的影响。

最后两章主要介绍 I/O 子系统。第 16 章介绍设备驱动程序框架、内核与 I/O 子系统的交互、SVR4 设备驱动程序接口以及内核与驱动程序交互接口规范。第 17 章讨论 STREAMS 框架，用于开发网络协议、网络驱动程序和终端驱动程序。

排版约定

我在本书中遵循了一些印刷规范。首次出现的术语或概念用楷体。内部内核函数和变量名字以及代码示例都是等宽字体，比如 `ufs_lookup()`。当指定调用语法时，系统调用名是斜体，而参数是等宽字体。在图中，实线箭头代表直接指针，虚线箭头表明箭

头的源和目标对象是间接推论的。

尽管我做了最大的努力，但错误和疏忽仍然在所难免。如果你发现有什么错误，或者有任何建议，请通过电子邮件（vahalia@acm.org）发送给我。

致谢

许多人为本书的出版做出了贡献。首先，我想感谢我的儿子 Rohan 以及我的妻子 Archana，他们的耐心、爱以及无私的奉献使我得以完成本书。我利用了本属于他们的周末和夜晚时间来进行写作，而他们给予我的一向都是微笑并一直鼓励我。我也要感谢父母对我的爱和支持。

接下来，我要感谢我的朋友 Subodh Bapat，他鼓励我承担这本书的写作工作，并花费大量时间给出了专业性建议和持续性激励，非常感谢他允许我使用他的书（《Object-Oriented Networks》[Bapa 94]）中提到的工具、模块和宏，还要感谢他对我草稿一丝不苟的审核以及对我在写作风格上清晰的指导。

许多审稿人贡献了非常多的时间和专业知识并多次审稿，提供了非常有价值的意见和建议，使得本书的质量有了提升。非常感谢 Peter Salus 的鼓励和支持以及 Benson Marguiles、Terry Lambert、Mark Ellis 和 William Bully 有深度的反馈意见。最后也要感谢 Keith Bostic、Evi Nemeth、Pat Parseghian、Steven Rago、Margo Seltzer、Richard Stevens 以及 Lev Vaitzblit 审核了本书的部分章节。

我还要感谢我的经理 Percy Tzelnic 对我在写作本书过程中的支持和理解。最后，我要感谢出版人 Alan Apt，是他促成了本书，并在每个阶段都提供了帮助。还要感谢 Prentice-Hall 和 Spectrum Publisher Services 出版社的团队人员，特别感谢 Shirlery McGuire、Sondra Chavez 和 Kelly Ricci 的支持和帮助。

参考文献

- [Bach 86] Bach, M.J., *The Design of the UNIX Operating System*, Prentice-Hall, 1986.
- [Bapa 94] Bapat, S.G., *Object-Oriented Networks*, Prentice-Hall, 1994.
- [Good 94] Goodheart, B., and Cox, J., *The Magic Garden Explained—The Internals of UNIX System V Release 4, An Open Systems Design*, Prentice-Hall, 1994.
- [Leff 89] Leffler, S.J., McKusick, M.K., Karels, M.J., and Quarterman, J.S., *The Design and Implementation of the 4.3 BSD UNIX Operating System*, Addison-Wesley, 1989.

目 录

UNIX Internals: The New Frontiers

出版者的话	
译者序	
序言	
前言	
第 1 章 从头说起	1
1.1 简介	1
1.1.1 UNIX 简史	1
1.1.2 起源	1
1.1.3 扩散	2
1.1.4 BSD	3
1.1.5 System V	4
1.1.6 商业化	5
1.1.7 Mach	5
1.1.8 标准	5
1.1.9 OSF 和 UI	6
1.1.10 SVR4 及其之后	7
1.2 变革使命	8
1.2.1 功能	8
1.2.2 网络	8
1.2.3 性能	9
1.2.4 硬件变化	9
1.2.5 质量提升	10
1.2.6 变革	10
1.2.7 其他应用程序领域	11
1.2.8 小即是美	11
1.2.9 灵活性	12
1.3 回顾过去, 展望未来	13
1.3.1 UNIX 系统的优点是什么	13
1.3.2 UNIX 系统的缺点是什么	14
1.4 本书内容说明	15
参考文献	15
第 2 章 进程与内核	17
2.1 简介	17
2.2 模式、空间和上下文	19
2.3 进程抽象	21
2.3.1 进程状态	21
2.3.2 进程上下文	23
2.3.3 用户凭据	23
2.3.4 u 区和 proc 结构	24
2.4 执行在内核态中	26
2.4.1 系统调用接口	26
2.4.2 中断处理	27
2.5 同步	29
2.5.1 阻塞操作	29
2.5.2 中断	30
2.5.3 多处理器	31
2.6 进程调度	31
2.7 信号	32
2.8 新的进程和程序	33
2.8.1 fork 和 exec	33
2.8.2 进程的创建	34
2.8.3 fork 的优化	35
2.8.4 调用新的程序	35
2.8.5 进程终止	37
2.8.6 等待进程终止	37
2.8.7 僵死进程	38
2.9 小结	39
2.10 练习题	39
参考文献	39
第 3 章 线程和轻量级进程	41
3.1 简介	41
3.1.1 动机	41
3.1.2 多线程和多处理器	42
3.1.3 并发和并行	43
3.2 基本抽象	44
3.2.1 内核线程	45
3.2.2 轻量级进程	45
3.2.3 用户线程	46
3.3 轻量级线程设计时要考虑的问题	49
3.3.1 fork 的语义	49
3.3.2 其他系统调用	50
3.3.3 信号传递和处理	50

3.3.4 可见性	51	4.5 SVR4 上的信号	76
3.3.5 栈增长	51	4.6 信号的实现	77
3.4 用户级别的线程库	51	4.6.1 信号生成	78
3.4.1 编程接口	52	4.6.2 交付和处理	78
3.4.2 线程库的实现	52	4.7 异常	79
3.5 调度器激活	53	4.8 Mach 的异常处理	79
3.6 Solaris 和 SVR4 上的多线程	54	4.8.1 异常端口	80
3.6.1 内核线程	54	4.8.2 错误处理	80
3.6.2 轻量级进程的实现	55	4.8.3 调试器交互	81
3.6.3 用户线程	56	4.8.4 分析	81
3.6.4 用户线程的实现	56	4.9 进程组和终端管理	82
3.6.5 中断处理	57	4.9.1 基本概念	82
3.6.6 系统调用处理	58	4.9.2 SVR3 模型	83
3.7 Mach 的线程	58	4.9.3 限制	84
3.7.1 Mach 抽象: 任务和线程	58	4.9.4 4.3BSD 的进程组和终端	84
3.7.2 Mach 的 C-threads	59	4.9.5 缺点	86
3.8 Digital UNIX	60	4.10 SVR4 的会话体系结构	86
3.8.1 UNIX 接口	60	4.10.1 动机	87
3.8.2 系统调用和信号	62	4.10.2 会话和进程组	87
3.8.3 pthreads 库	62	4.10.3 数据结构	88
3.9 Mach 3.0 的 continuation	63	4.10.4 控制终端	89
3.9.1 编程模型	63	4.10.5 4.4BSD 的会话实现机制	89
3.9.2 使用 continuation	63	4.11 小结	90
3.9.3 优化	65	4.12 练习题	90
3.9.4 分析	65	参考文献	91
3.10 小结	65	第 5 章 进程调度	92
3.11 练习题	66	5.1 简介	92
参考文献	66	5.2 时钟中断处理	93
第 4 章 信号和会话管理	68	5.2.1 callout	93
4.1 简介	68	5.2.2 告警	95
4.2 信号生成和处理	68	5.3 调度器目标	95
4.2.1 信号处理	69	5.4 传统的 UNIX 调度	96
4.2.2 信号的生成	71	5.4.1 进程优先级	97
4.2.3 典型场景	72	5.4.2 调度器的实现	98
4.2.4 睡眠与信号	72	5.4.3 运行队列的操作	99
4.3 不可靠的信号	73	5.4.4 分析	99
4.4 可靠的信号	74	5.5 SVR4 调度器	100
4.4.1 主要特性	74	5.5.1 类无关层	100
4.4.2 SVR3 实现	75	5.5.2 调度类的接口	101
4.4.3 BSD 信号管理	75	5.5.3 分时类	103

5.5.4	实时类	104	6.5.2	消息传递接口	136
5.5.5	pricntl 系统调用	105	6.6	端口	137
5.5.6	分析	106	6.6.1	端口命名空间	137
5.6	Solaris 2.x 调度的改善	107	6.6.2	端口数据结构	138
5.6.1	可抢占的内核	107	6.6.3	端口转换	138
5.6.2	多处理器的支持	107	6.7	消息传递	139
5.6.3	隐式调度	108	6.7.1	转换端口权利	140
5.6.4	优先级反转	109	6.7.2	out-of-line 内存	141
5.6.5	优先级继承的实现	110	6.7.3	控制流	142
5.6.6	优先级继承的局限性	112	6.7.4	通知	143
5.6.7	turnstile	113	6.8	端口操作	143
5.6.8	分析	113	6.8.1	销毁端口	143
5.7	Mach 上的调度	113	6.8.2	备份端口	144
	多处理器支持	114	6.8.3	端口集合	144
5.8	Digital UNIX 的实时调度	116	6.8.4	端口插补	145
	多处理器支持	116	6.9	扩展性	145
5.9	其他调度实现	117	6.10	Mach 3.0 的增强	146
5.9.1	公平调度方法	117	6.10.1	一次性的发送权利	147
5.9.2	最终期限驱动调度方法	117	6.10.2	Mach 3.0 的通知	147
5.9.3	三级调度器	118	6.10.3	发送权利的用户引用计数	148
5.10	小结	119	6.11	讨论	148
5.11	练习题	119	6.12	小结	149
	参考文献	120	6.13	练习题	149
				参考文献	149
第 6 章	进程间通信	121	第 7 章	同步和多处理器	151
6.1	简介	121	7.1	简介	151
6.2	通用的 IPC 方法	121	7.2	传统 UNIX 内核里的同步机制	152
6.2.1	信号	122	7.2.1	中断屏蔽	152
6.2.2	管道	122	7.2.2	睡眠和唤醒	152
6.2.3	SVR4 管道	124	7.2.3	传统方法的局限性	153
6.2.4	进程跟踪	124	7.3	多处理器系统	154
6.3	System V IPC	126	7.3.1	内存模型	154
6.3.1	公共元素	126	7.3.2	同步支持	155
6.3.2	信号量	127	7.3.3	软件体系架构	156
6.3.3	消息队列	130	7.4	多处理器的同步问题	157
6.3.4	共享内存	131	7.4.1	唤醒丢失问题	157
6.3.5	讨论	133	7.4.2	惊群问题	158
6.4	Mach IPC	133	7.5	信号量	158
	基本概念	134	7.5.1	信号量提供互斥操作	159
6.5	消息	135	7.5.2	使用信号量提供事件等待	159
6.5.1	消息数据结构	135			

7.5.3 使用信号量来控制可计数的资源	160	8.6 Vnode/Vfs 架构	187
7.5.4 信号量的缺点	160	8.6.1 目标	187
7.5.5 Convoy	161	8.6.2 从设备 I/O 得到的注解	187
7.6 自旋锁	162	8.6.3 vnode/vfs 接口概览	190
7.7 条件变量	163	8.7 实现概览	191
7.7.1 实现问题	164	8.7.1 目标	191
7.7.2 事件	165	8.7.2 Vnodes 以及打开文件	191
7.7.3 阻塞锁	165	8.7.3 Vnode	192
7.8 读写锁	165	8.7.4 Vnode 引用计数	193
7.8.1 设计考虑	165	8.7.5 Vfs 对象	194
7.8.2 实现	166	8.8 文件系统相关对象	194
7.9 引用计数	167	8.8.1 每个文件的私有数据	194
7.10 其他考虑	168	8.8.2 vnodeops 结构	195
7.10.1 死锁避免	168	8.8.3 vfs 层中文件系统相关部分	196
7.10.2 递归锁	169	8.9 挂载文件系统	196
7.10.3 阻塞还是自旋	170	8.9.1 虚拟文件系统转换表	196
7.10.4 锁什么	170	8.9.2 mount 函数实现	197
7.10.5 粒度和持续时间	170	8.9.3 VFS_MOUNT 过程	197
7.11 案例研究	171	8.10 文件操作	197
7.11.1 SVR4.2/MP	171	8.10.1 路径遍历	198
7.11.2 Digital UNIX	172	8.10.2 目录名查找缓存	199
7.11.3 其他实现	173	8.10.3 VOP_LOOKUP 操作	199
7.12 小结	174	8.10.4 打开文件	200
7.13 练习题	174	8.10.5 文件 I/O	201
参考文献	175	8.10.6 文件属性	201
8.10.7 用户凭据	201	8.11 分析	202
第 8 章 文件系统接口和框架	176	8.11.1 SVR4 系统实现的缺点	202
8.1 简介	176	8.11.2 4.4BSD 模型	203
8.2 文件的用户接口	176	8.11.3 OSF/1 方法	204
8.2.1 文件和目录	177	8.12 小结	205
8.2.2 文件属性	178	8.13 练习题	205
8.2.3 文件描述符	179	参考文献	206
8.2.4 文件 I/O	181	第 9 章 文件系统的实现	207
8.2.5 分散 - 聚集 I/O	182	9.1 简介	207
8.2.6 文件锁机制	183	9.2 System V 文件系统 (s5fs)	208
8.3 文件系统	183	9.2.1 目录	208
8.4 特殊文件	184	9.2.2 inode	209
8.4.1 符号链接	184	9.2.3 超级块	210
8.4.2 管道和 FIFO	185	9.3 s5fs 内核组织	211
8.5 文件系统框架	186		

9.3.1	内存 inode	211	10.5	NFS 实现	240
9.3.2	inode 查找	212	10.5.1	控制流	240
9.3.3	文件 I/O	213	10.5.2	文件句柄	241
9.3.4	inode 的分配和回收	214	10.5.3	挂载操作	241
9.4	s5fs 的分析	215	10.5.4	路径名的查找	242
9.5	伯克利快速文件系统 (FFS)	216	10.6	UNIX 语义	243
9.6	硬盘结构	216	10.6.1	打开文件许可	243
9.7	磁盘组织	216	10.6.2	已打开文件的删除	243
9.7.1	块和片段	217	10.6.3	读写操作	244
9.7.2	分配策略	218	10.7	NFS 性能	244
9.8	FFS 的增强功能	219	10.7.1	性能瓶颈	244
9.9	分析	220	10.7.2	客户端缓存	244
9.10	临时文件系统	221	10.7.3	延迟写	245
9.10.1	内存文件系统	221	10.7.4	重传缓存	246
9.10.2	tmpfs 文件系统	222	10.8	专用 NFS 服务器	247
9.11	特殊用途文件系统	223	10.8.1	Auspex 的 Functional Multiprocessor 架构	247
9.11.1	specfs 文件系统	223	10.8.2	IBM 的 HA-NFS 服务器	248
9.11.2	/proc 文件系统	223	10.9	NFS 安全	249
9.11.3	处理器文件系统	225	10.9.1	NFS 访问控制	249
9.11.4	Trans lucent 文件系统	225	10.9.2	UID 重映射	250
9.12	旧的缓冲区缓存	226	10.9.3	根用户重映射	250
9.12.1	基本操作	227	10.10	NFS 版本 3	251
9.12.2	缓冲区头结构	228	10.11	远程文件共享	252
9.12.3	优点	228	10.12	RFS 架构	252
9.12.4	缺点	228	10.12.1	远程消息协议	253
9.12.5	保证文件系统的一致性	229	10.12.2	有状态操作	253
9.13	小结	230	10.13	RFS 实现	254
9.14	练习题	230	10.13.1	远程挂载	254
	参考文献	231	10.13.2	RFS 客户端和服务端	255
			10.13.3	崩溃恢复	255
			10.13.4	其他问题	256
第 10 章	分布式文件系统	232	10.14	客户端缓存	256
10.1	简介	232	10.15	Andrew 文件系统	258
10.2	分布式文件系统的一般特征	232	10.15.1	可伸缩架构	258
10.3	网络文件系统	233	10.15.2	存储和命名空间的组织	259
10.3.1	用户视角	234	10.15.3	会话级语义	260
10.3.2	设计目标	235	10.16	AFS 实现	260
10.3.3	NFS 的组件	235	10.16.1	缓存与一致性	261
10.3.4	无状态设计	237	10.16.2	路径名查找	261
10.4	NFS 协议集	238			
10.4.1	外部数据表示	238			
10.4.2	远程过程调用	239			

10.16.3 安全性	262	11.9.3 “看门狗”进程的应用	290
10.17 AFS 的不足	262	11.10 4.4BSD 的 portal 文件系统	290
10.18 DCE 的分布式文件系统	263	11.11 可堆叠文件系统层次	291
10.18.1 DFS 架构	263	11.11.1 框架和接口	292
10.18.2 缓存一致性	264	11.11.2 SunSoft 原型	293
10.18.3 令牌管理器	265	11.12 4.4BSD 文件系统接口	294
10.18.4 DFS 的其他服务	266	11.13 小结	295
10.18.5 分析	266	11.14 练习题	295
10.19 小结	267	参考文献	296
10.20 练习题	267	第 12 章 内核内存分配	298
参考文献	268	12.1 简介	298
第 11 章 高级文件系统	271	12.2 功能需求	299
11.1 简介	271	12.3 资源映射分配器	301
11.2 传统文件系统的局限	271	12.4 简单的空闲链表分配器	303
11.2.1 FFS 磁盘布局	272	12.5 McKusick-Karels 分配器	305
11.2.2 磁盘上写操作的主导	273	12.6 伙伴系统	307
11.2.3 元数据更新	274	12.7 SVR4 的惰性伙伴算法	308
11.2.4 故障修复	274	12.7.1 惰性合并	309
11.3 文件系统簇 (SUN-FFS)	275	12.7.2 SVR4 的实现细节	310
11.4 日志方法	276	12.8 Mach 和 OSF/1 的区块分配器	310
11.5 日志结构文件系统	277	12.8.1 垃圾回收	311
11.6 4.4BSD 日志结构文件系统	277	12.8.2 分析	312
11.6.1 日志写入	278	12.9 一种针对多处理器系统的分层式 分配器	312
11.6.2 数据检索	279	12.10 Solaris 2.4 的 Slab 分配器	314
11.6.3 崩溃恢复	279	12.10.1 复用对象	314
11.6.4 cleaner 进程	280	12.10.2 利用硬件缓存	315
11.6.5 分析	280	12.10.3 分配器足迹	315
11.7 元数据日志	281	12.10.4 设计与接口	316
11.7.1 正常操作	282	12.10.5 实现细节	317
11.7.2 日志的一致性	283	12.10.6 分析	318
11.7.3 崩溃恢复	284	12.11 小结	318
11.7.4 分析	284	12.12 练习题	319
11.8 Episode 文件系统	285	参考文献	320
11.8.1 基本抽象	285	第 13 章 虚拟内存	321
11.8.2 结构	286	13.1 简介	321
11.8.3 日志记录	287	13.2 按需分页	324
11.8.4 其他特性	287	13.2.1 功能需求	324
11.9 “看门狗”监视器	288	13.2.2 虚拟地址空间	325
11.9.1 目录的“看门狗”进程	289	13.2.3 页面的首次访问	326
11.9.2 消息通道	289		

13.2.4	交换区	326	14.7.2	匿名页处理	361
13.2.5	转换映射	327	14.7.3	创建进程	363
13.2.6	页面替换策略	328	14.7.4	共享匿名页	364
13.3	对硬件的需求	328	14.7.5	处理缺页异常	364
13.3.1	MMU 缓存	330	14.7.6	共享内存	365
13.3.2	Intel 80x86	331	14.7.7	其他组件	366
13.3.3	IBM RS/6000	333	14.8	与 vnode 子系统的交互	367
13.3.4	MIPS R3000	335	14.8.1	对 vnode 接口的修改	367
13.4	4.3BSD——案例研究	336	14.8.2	统一文件访问机制	368
13.4.1	物理内存	337	14.8.3	其他细节	370
13.4.2	地址空间	338	14.9	Solaris 的虚拟交换空间	370
13.4.3	页在哪里	339	14.9.1	交换空间扩展	370
13.4.4	交换空间	340	14.9.2	虚拟交换管理	371
13.5	4.3 BSD 内存管理操作	341	14.9.3	讨论	372
13.5.1	创建进程	341	14.10	分析	372
13.5.2	缺页异常处理	342	14.11	性能改进	374
13.5.3	空闲页面链表	344	14.11.1	缺页异常率偏高的原因	374
13.5.4	交换	345	14.11.2	SVR4 对 SunOS VM 实现的改进	375
13.6	分析	346	14.11.3	结果与讨论	375
13.7	练习题	347	14.12	小结	376
参考文献		348	14.13	练习题	376
第 14 章	SVR4 VM 架构	349	参考文献		377
14.1	简介	349	第 15 章	其他内存管理技术	378
14.2	内存映射文件	349	15.1	简介	378
14.3	VM 的设计理念	351	15.2	Mach 的内存管理设计	378
14.4	基础抽象	352	15.2.1	设计目标	378
14.4.1	物理内存	353	15.2.2	对外接口	379
14.4.2	地址空间	353	15.2.3	基础抽象	380
14.4.3	地址映射	354	15.3	内存共享机制	381
14.4.4	匿名页	355	15.3.1	写时复制共享	382
14.4.5	硬件地址转换	356	15.3.2	读写共享	383
14.5	段驱动程序	357	15.4	内存对象与 Pager	384
14.5.1	seg_vn	357	15.4.1	初始化内存对象	384
14.5.2	seg_map	358	15.4.2	内核与 pager 的接口	384
14.5.3	seg_dev	359	15.4.3	内核与 Pager 的交互	385
14.5.4	seg_kmem	359	15.5	外部 pager 和内部 pager	386
14.5.5	seg_kp	359	15.6	页面替换	388
14.6	交换层	359	15.7	分析	389
14.7	VM 操作	361	15.8	4.4BSD 的内存管理	390
14.7.1	创建新映射	361			