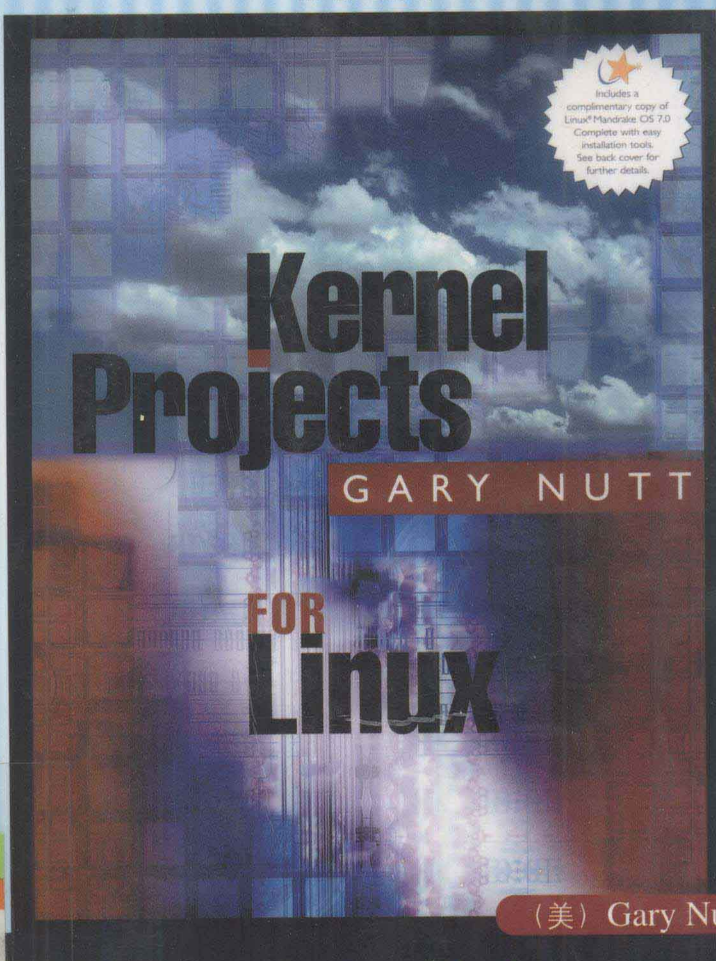
经典原版书库

Linux
操作系统内核实习

(英文版)



(美) Gary Nutt 著

附赠

械工业出版社
na Machine Press
Pearson Education
培生教育出版集团

经 典 原 版 书 库

Linux
操作系统内核实习

(英文版)

Kernel Projects for Linux

(美) Gary Nutt 著

机械工业出版社
China Machine Press
Pearson Education
培生教育出版集团

本书版权登记号：图字：01-2001-2201

图书在版编目（CIP）数据

Linux操作系统内核实习：英文版/（美）加里纳特（Nutt, G）著. –北京：机械工业出版社，2002.6

（经典原版书库）

书名原文：Kernel Projects for Linux

ISBN 7-111-10099-9

Ⅰ. L… Ⅱ. 加… Ⅲ. Linux操作系统–英文 Ⅳ. TP316.81

中国版本图书馆CIP数据核字（2002）第017191号

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：华 章

北京昌平奔腾印刷厂印刷·新华书店北京发行所发行

2002年6月第1版第1次印刷

850mm×1168mm1/32·8.25印张

印数：0 001-4 000册

定价：32.00元（附光盘2张）

凡购本书，如有倒页、脱页、缺页，由本社发行部调换

出版者的话

文艺复兴以降，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；也正是这样的传统，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，还揭櫫了学术的源变，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、从业人员较少的现状下，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图文信息有限公司较早意识到“出版要为教育服务”。自1998年始，华章公司就将工作重点放在了遴选、移译国外优秀教材上。经过几年的不懈努力，我们与Prentice Hall，Addison-Wesley，McGraw-Hill，Morgan Kaufmann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Tanenbaum，Stroustrup，Kernighan，Jim Gray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、研究及庋藏。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襄助，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，有的还专诚为其书的中译本作序。迄今，“计算机科学丛书”已经出版了近百个品种，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，为进一步推广与发展打下了坚实的基础。

随着学科建设的初步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步入一个新的阶段。为此，华章公司将加大引进教材的力度，在“华章教育”的总规划之下出版三个系列的计算机教材：针对本科生的核心课程，剔抉外版菁华而成“国外经典教材”系列；对影印版的教材，则单独开辟出“经典原版书库”；定位在高级教程和专业参考的“计算机科学丛书”还将保持原来的风格，继续出版新的品种。为了保证这三套丛书的权威性，同时也为了更好地为学校和老师们服务，华章公司聘请了中国科学院、北京大学、清华大学、国防科技大学、复旦大学、上海交通大学、南京大学、浙江大学、中国科技大学、哈尔滨工业大学、西安交通大学、中国人民大学、北京航空航天大学、北京邮电大学、中山大学、解放军理工大学、郑州大学、湖北工学院、中国国家信息安全测评认证中心等国内重点大学和科研机构在计算机的各个领域的著名学者组成“专家指导委员会”，为我们提供选题意见和出版监督。

“经典原版书库”是响应教育部提出的使用原版国外教材的号召，为国内高校的计算机教学度身订造的。在广泛地征求并听取丛书的“专家指导委员会”的意见后，我们最终选定了这30多种篇幅内容适度、讲解鞭辟入里的教材，其中的大部分已经被M.I.T.、Stanford、U.C. Berkley、C.M.U.等世界名牌大学采用。丛书不仅涵盖了程序设计、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、离散数学等国内大学计算机专业普遍开设的核心课程，而且各具特色——有的出自语言设计者之手、有的历三十年而不衰、有的已被全世界的几百所高校采用。在这些圆熟通博的名师大作的指引之下，读者必将在计算机科学的宫殿中由登堂而入室。

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，但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。教材的出版只是我们的后续服务的起点。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：

电子邮件：hzedu@hzbook.com

联系电话：（010）68995265

联系地址：北京市西城区百万庄南街1号

邮政编码：100037

专家指导委员会

（按姓氏笔画顺序）

Preface

TO THE STUDENT

Experience has shown that the best way to learn the details of how an operating system (OS) works is to experiment with it—to read, modify, and enhance its code. However, OS software, by its nature, must be carefully constructed. This is because it directly controls the hardware used by all of the processes and threads that execute on it. As a result, experimenting with OS code can be difficult, since an experimental version of the OS might disable the test machine. This laboratory manual provides you with a learning path for studying the Linux kernel with as little risk as possible. By learning about the Linux kernel in this way, you will develop a technique by which you can learn and experiment with other kernels as well. Consider this learning path to be a graduated set of exercises. First, you will learn to inspect various aspects of the OS internal state without changing any code. Second, you will extend the OS by writing new code to read (but not write) kernel data structures. Third, you will reimplement existing data structures. Finally, you will design and add your own functions and data structures to the Linux kernel.

The Linux kernel is written in the C programming language. Therefore you need to be relatively proficient in using C before you can study the kernel. If you know C++, then you will not have difficulty understanding the source code, though when you add or change parts of the kernel code you will not be able to use objects.

This manual is designed as a companion to a general OS textbook. It consists of two parts. Part 1 offers an overview of the Linux design. If you are using this manual at the beginning of your first OS course, you might discover that Part 1 discusses several topics that are new to you. However, work your way through it to get a feeling for how Linux is built. It gives the "big picture" but not many details. Then go back to Part 1 as needed as you work the exercises.

Part 2 consists of a dozen laboratory exercises that help you to learn to use Linux. Each exercise is a self-contained unit consisting of the following sections:

- Introduction
- Problem Statement
- Attacking the Problem

The exercises link the general concepts and the Linux details. Each begins with an introduction that explains Linux concepts and details relevant to the exercise. The introduction explains how the generic concepts that you will have learned in lecture and textbooks are realized in Linux. The next part of the exercise presents the problem on which you will be working. It includes detailed Linux-specific information that you need to solve the problem. Sometimes, a quick review of the pertinent parts of Part 1 will help you to frame the work of the exercises before you dive into the details.

Your school's laboratory probably has already been set up as a Linux lab. For you to solve most of the exercises in this manual, the laboratory administrator will provide you with a copy of the Linux source code and superuser permission to create new versions of the OS. Do not abuse your privilege as a superuser! You need this privilege in order to modify the kernel, but you must not use it for other purposes. This manual includes a CD-ROM, containing the Linux source code, that you can use to install Linux on your own computer.

Good luck with your study of operating systems. I hope that this exercise manual is a valuable learning tool for exploring OS concepts in Linux.

TO THE INSTRUCTOR

Today, abstraction is the basis of most software that is written—in the classroom or in practice. Students are taught to think of software solutions in terms of objects, components, threads, messages, and so on. This perspective teaches them to leverage the power of the hardware to solve increasingly complex tasks. In this way, they reduce programming time while reusing lower-level abstractions. At the bottom of all of these abstractions is the operating system—processes and resources (and sometimes threads). Application software and middleware use these OS abstractions to create their own higher-level abstractions. These range from accounting packages, spreadsheets, and missile trackers to windows, databases, objects, components, messages, and continuous media streams.

This trend toward the heavy use of abstraction prompts some to argue that OSs are no longer worthy of serious study, since they are largely transparent to the application programmers working at higher layers of abstraction. However, the OS is still fundamental because its design and implementation are the basis of the design and implementation of all of the other abstractions. Programmers will always be able to write better middleware and application programs if they understand how OSs work. Moreover, the need for people who understand basic OS technology remains, whether they are to write drivers for new devices, to create new microkernel servers, or to provide new systems capable of efficiently handling evolving requirements such as continuous media.

Typically an OS instructor has to decide whether an OS course should focus on issues and theory or provide an environment in which students can experiment with OS code. The 1991 (and draft 2001) IEEE/ACM undergraduate course recommendation describes a course that consists of a substantial amount of time spent on issues, but also includes a significant laboratory component. Even though the trend is toward courses based on conceptual materials, students and instructors seem to agree that hands-on experience is invaluable in learning about OSs. Many courses attempt to follow the IEEE/ACM lead by dividing the course into lecture and laboratory components, with the lecture component focusing on issues and theory and the laboratory component providing some form of hands-on exercises.

The IEEE/ACM recommendation supports the idea that the laboratory component should allow students to learn how to use the OS mechanisms, specifically by focusing on the OS application programming interface (API) as the primary mechanism for experimentation. The philosophy behind this approach is that students must learn how to *use* an OS before they can really understand how to *design* one. This philosophy drives a companion book on programming Windows NT via the Win32 API [Nutt, 1999] and one on laboratory exercises [Nutt, 2000].

However, in a late 1998 survey of 78 universities conducted by Addison-Wesley, 43 indicated that they teach OS *internals* in the introductory OS course. Of these 43, 26 use a variant of UNIX as the target OS, 13 use Linux, 10 use an unspecified version of UNIX, and 3 use MINIX. Eight said that they use some other OS as the subject system (such as Nachos), and the remaining 9 did not specify the OS that they use. The survey clearly showed that a significant fraction of the community teaches OS internals as a component of an introductory OS class, despite the IEEE/ACM recommendation and despite the heavy use of conceptual OS textbooks. It also showed that most of these courses use two books: a traditional OS theory book (such as [Silberschatz and Galvin, 1998] or [Nutt, 2000]) and a reference book (such as [Stevens, 1993], [McKusick, et al., 1996], or [Beck, et al., 1998]). Of course, no single-term undergraduate course can possibly cover all of the material in both a theory book and a book that describes an entire OS. The lack of a good laboratory manual forces the instructor to have students buy a supplementary book that contains *much* more information than they will have time to learn in a single academic term. Further, the instructor will have to learn all of the material in both books, as well as learn the subject OS, derive a suitable set of exercises, and provide some form of guidance through the OS reference materials so that the students can solve the exercises.

This textbook is a laboratory manual of Linux internal exercises. It complements an OS theory book by providing a dozen specific exercises on Linux internals that illustrate how theoretical concepts are implemented in Linux. The instructor does not need to become a "complete" Linux kernel expert or derive a set of exercises (either with full documentation for the exercise or with pointers to appropriate sections in a supplementary reference book). Instead, the instructor, lab assistant, and students can use this manual as a self-contained source of background data and exercises to study how concepts are implemented. Thus the less expensive laboratory manual replaces a general reference book, while providing focused information for a set of kernel internals exercises. For the student who wants to understand related information that is not required in order to solve the exercise, the background material for exercises provides pointers to reference books (and the literature).

A single-semester OS course consists of 15 weeks of course material. In my experience, many undergraduate students have difficulty doing a substantial programming exercise in less than one and a half to two weeks. This means that my students have been able to complete perhaps six to eight programming assignments in a semester. This manual provides enough exercises to allow you to choose a subset that best suits your students' backgrounds and your preferences. Most of the exercises include options that allow you to vary the assignments from term to term (thereby reducing the probability of public solutions from previous terms). As mentioned previously, my intention is to release frequent editions of this manual. I expect that the new editions will have new exercises that require new solutions.

Also provided is a solution to each exercise. Thus more difficult exercises can be chosen, and as necessary you can distribute parts of the solution that are not published in the manual.

None of these exercises are as difficult as building a new kernel from scratch. Rather, they emphasize having students study the existing Linux kernel by modifying or extending its components. The first ones are easy, and the background material is comprehensive. Later exercises increase in difficulty, with decreasing amounts of "handholding" information. Exercises 1 and 2 will ordinarily require a week or less to complete, but the last third of the exercises are likely to require a couple of weeks each. If your students need extra practice with C programming, you might carefully consider using Exercises 1 and 2 as tutorials. This might require that you provide a little extra assistance, especially with the concurrency elements of Exercise 2.

THE CD-ROM VERSION OF LINUX

Any hands-on study of an OS must commit to a particular version of the OS. With the rapid evolution of Linux, Version 2.2.*x* will be out of date by the time that the book is published. In an attempt to avoid the problem of the book and the OS code being out of sync, I have included the source code for Version 2.2.14. This first edition of the manual was originally written for Version 2.0.36. Then it was updated for Linux Version 2.2.12 just before it went into the publication cycle. As the book was going to the printer, I discovered that only 2.2.14 (and not 2.2.12) was available for distribution with the manual. Small differences exist between 2.2.12 and 2.2.14—generally in coding style rather than content. However, some of these differences show up in a couple of exercises. Specifically, watch for them in the virtual memory and scheduler parts of the kernel code. I will correct them in the next edition. I decided that including a complete 2.2.14 installation was better than the manual's having no CD-ROM. Though newer versions of the source code will be available when you use this book, I encourage you to install this version on your laboratory machines so that your students will have a software environment that is reasonably consistent with the manual. My best wish is that I will be able to release new editions of the manual that roughly track the Linux releases; the next edition might use, for example, Version 2.6*x*.

ACKNOWLEDGEMENTS

This manual represents years of effort—mine as well as other people's—learning about Linux. I benefited considerably from the assistance, insight, and contributions of the teaching assistants for Computer Science 3753: Operating Systems at the University of Colorado, namely: Don Lindsay, Sam Siewert, Ann Root, and Jason Casmira. Phil Levis provided interesting and lively discussions of Linux and the exercises. And when I first installed Linux on a machine, it worked, though not as well as it did after Adam Griff polished the installation.

Many of the exercises derive from projects and exercises in the undergraduate and graduate OS classes at the University of Colorado. In particular, Exercise 3 was created by Sam Siewert in spring 1996 for Computer Science 3753. Exercise 4 takes some material from another exercise created by Sam Siewert. Exercise 9 comes from a course project that Jason Casmira did in Computer Science 5573 (the graduate OS class) in fall 1998. Exercise 10 was first designed by Don Lindsay in fall 1995 and refined by Sam Siewert in spring 1996. Exercise 1 also appears in my companion OS textbook [Nutt, 2000], and Exercise 2 is an extension of another one that also appears in that book. Exercises 11 and 12 resemble exercises for Windows NT that appear in another of my manuals [Nutt, 1999]; Norman Ramsey created the original Windows NT exercises.

Many reviewers helped make the manual much better than its original draft. Richard Guy used the first public draft of the manuscript in a course at UCLA. Paul Stelling (UCLA) did a careful reading of the draft, correcting errors and providing insight into good and bad aspects of it. Simon Gray (Ashland University) provided particularly lucid and insightful comments on the exercises. The following also provided helpful comments that greatly improved the manual: John Barr (Ithaca College), David Binger (Centre College), David E. Boddy (Oakland University), Richard Chapman (Auburn University), Sorin Draghici (Wayne State University), Sandeep Gupta (Colorado State University), Mark Holliday (Western Carolina University), Kevin Jeffay (University of North Carolina at Chapel Hill), Joseph J. Pfeiffer (New Mexico State University), Kenneth A. Reek (Rochester Institute of Technology), and Henning Schulzrinne (Columbia University).

The Addison Wesley staff was very helpful in preparing this work. Molly Taylor and Jason Miranda provided wide ranges of assistance in handling reviews and

otherwise supporting the early stages of development. Lisa Hogue finished the process, particularly by saving the day by finding a version of the Linux source code that could be distributed with the book. Laura Michaels did her usual diligent job in copy editing the work, Gina Hagen helped with production and Helen Reebenacker was the production editor. Last, but not least, Maité Suarez-Rivas, the acquisition editor, recognized the need for the book and relentlessly pushed forward to make it happen.

All of these people helped to produce the content, but of course, any errors are my responsibility.

Gary Nutt

Boulder, Colorado

计算机科学丛书

《编译原理及实践》	Louden 著/冯博琴 等译/39.00元
《UNIX环境高级编程》	Stevens 著/尤晋元 等译/55.00元
《分布式操作系统：概念与实践》	Galli 著/徐良贤 等译/
《分布式系统设计》	Jie Wu 著/高传善 等译/30.00元
《现代操作系统》	Tanenbaum 著/陈向群 等译/40.00元
《UNIX操作系统设计》	Bach 著/陈葆珏 等译/33.00元
《UNIX编程环境》	Kernighan 等著/陈向群 等译/24.00元
《C语言解析》（原书第4版）	Pohl 著/麻志毅 等译/48.00元
《C程序接口与实现》	Hanson 著/
《C程序设计教程》（原书第2版）	Deitel 等著/薛万鹏 等译/33.00元
《C++程序设计教程》（原书第2版）	Deitel 等著/薛万鹏 等译/22.00元
《编码的奥秘》	Petzold 著/陆丽娜 等译/24.00元
《C++核心：软件工程方法》	Shtern 著/李师贤 等译/
《Java 程序设计导引》	Liang 著/王镁 等译/
《Java 语言解析》	Pohl 著/
《Java 程序设计教程》（原书第3版）	Deitel 著/袁兆山 等译/
《大型C++软件设计》	Lakos 著/李师贤 等译/
《C++语言的设计和演化》	Stroustrup 著/裘宗燕 译/48.00元
《C++编程思想》	Eckel 著/刘宗田 等译/39.00元
《C++编程思想》（第2版）	Eckel著/刘宗田 等译/
《Java编程思想》	Eckel 著/京京工作室 译/39.00元
《Java编程思想》（第2版）	Eckel 著/
《电子商务》	Schneider 等著/成栋 译/28.00元
《多媒体应用程序的面向对象设计》	Guzdial 著/
《计算机文化》（原书第3版）	Parsons 著/朱海滨 等译/50.00元
《图论导引》（原书第2版）	West 著/
《专家系统原理与编程》（原书第3版）	Giarratano 著/印鉴 等译/49.00元
《神经网络》	Haykin 著/史忠植 等译/
《设计模式：可复用面向对象软件的基础》	Gamma 等著/吕建 等译/35.00元
《软件工程：JAVA语言实现》（原书第4版）	Schach 著/袁兆山 等译/38.00元
《软件工程：实践者的研究方法》（原书第4版）	Pressman 著/梅宏 等译/48.00元

《软件需求》	Wiegers 著/陆丽娜 等译/19.00元
《数据库系统导论》(原书第7版)	Date 著/孟小峰 等译/66.00元
《数据仓库》(原书第2版)	Inmon 著/王志海 等译/25.00元
《数据挖掘：概念与技术》	Jiawei Han等著/范明 等译/39.00元
《数据库系统概念》(原书第3版)	Silberschatz 等著/杨冬青 等译/49.00元
《数据库系统实现》	Molina 等著/杨冬青 等译/45.00元
《数据库设计》	Stephens 等著/何玉洁 等译/35.00元
《数据库管理系统基础》	Pratt 等著/陆洪毅 等译/20.00元
《事务处理：概念与技术》	Gary 等著/孟小峰 等译/
《数字逻辑：应用与设计》	Yarbrough著/朱海滨 等译/49.00元
《嵌入式计算机系统设计原理》	Wolf 著/孙玉芳 等译/65.00元
《并行程序设计》	Wilkinson 著/陆鑫达 等译/43.00元
《最新网络技术基础》	Palmer 著/严伟 译/20.00元
《计算机网络与因特网》(原书第2版)	Comer 著/徐贤良 等译/40.00元
《计算机网络》(原书第2版)	Peterson 等著/叶新铭 等译/49.00元
《光纤通信技术》	Mynbaev 等著/吴时霖 等译/
《高性能通信网络》(原书第2版)	Walrand 等著/史美林 等译/55.00元
《数据通信与网络》	Forouzan 等著/潘仡等译，吴时霖校/48.00元
《ISDN、B-ISDN与帧中继和ATM》(原书第4版)	Stallings 著/程时端 等译/48.00元
《计算机网络实用教程》	Dean 著/陶华敏 等译/65.00元
《计算机网络实用教程实验手册》	Dean 著/陶华敏 等译/15.00元
《网络》(原书第2版)	Ramteke 著/侯春萍 等译/
《TCP/IP详解 卷1：协议》	Stevens 著/范建华 等译，谢希仁 校/ 45.00元
《TCP/IP详解 卷2：实现》	Wright/Stevens著/陆雪莹 等译，谢希仁 校/78.00元
《TCP/IP详解 卷3：TCP事务协议、HTTP、NNTP和UNIX域协议》	Stevens 著/胡谷雨 等译，谢希仁 校/ 35.00元
《Internet技术基础》(原书第2版)	Comer 著/袁兆山 等译/18.00元
《数据通信与网络教程》(原书第2版)	Shay 著/高传善 等译/40.00元
《密码学导引》	Garrete 著/吴世忠译
《分布计算的信息安全》	Bruce 著/
《计算机信息处理》(原书第7版)	Mandell 等著/尤晓东 等译/38.00元
《信息系统原理》(原书第3版)	Stair 等著/张靖 等译/42.00元

国外经典教材

《编译原理》	Aho 著/李建中 等译/
《Linux操作系统内核实习》	Nutt 著/陆丽娜 等译/29.00元
《操作系统原理与实践》（原书第2版）	Nutt著/尤晋元 等译/
《现代操作系统》（原书第2版）	Tanenbaum 著/陈向群 等译/
《C++程序设计语言》（特别版）	Stroustrup 著/裘宗燕 译/
《C程序设计语言》（原书第2版）	Kernighan/Ritchie 著/徐宝文 等译/28.00元
《程序设计实践》	Kernighan/Pike 著/裘宗燕 译/20.00元
《程序设计语言：概念与结构》（原书第2版）	Sethi 著/裘宗燕 等译/45.00元
《计算理论导引》	Sipser 著/张立昂 等译/30.00元
《离散数学及其应用》（原书第4版）	Rosen 著/袁崇义 等译/75.00元
《组合数学》（原书第3版）	Brualdi 著/冯舜玺 等译/38.00元
《人工智能》	Nilsson 著/郑扣根 等译/30.00元
《神经网络设计》	Hagan 等著/戴葵 等译/
《软件工程：实践者的研究方法》（原书第5版）	Pressman 著/梅宏 等译/
《数据结构、算法与应用：C++语言描述》	Sahni著/戴葵 等译/49.00元
《数据结构与算法分析：C语言描述》（原书第2版）	Weiss 著/冯舜玺 等译/
《数据库系统原理》（原书第4版）	Siblerschatz 等著/杨冬青等译/
《数据库原理、编程与性能》（原书第2版）	O'Neil 等著/周傲英 等译/55.00元
《并行计算机体系结构》	Culler/等著/李晓明 等译/
《结构化计算机组成》	Tanenbaum 著/刘卫东 等译/46.00元
《可扩展并行计算：技术、结构与编程》	Hwang/等著/陆鑫达 等译/49.00元
《计算机图形学的算法基础》（第2版）	Rogers 著/石教英 等译/55.00元
《数据通信与网络》（第2版）	Forouzan 等著/吴时霖 等译/68.00元