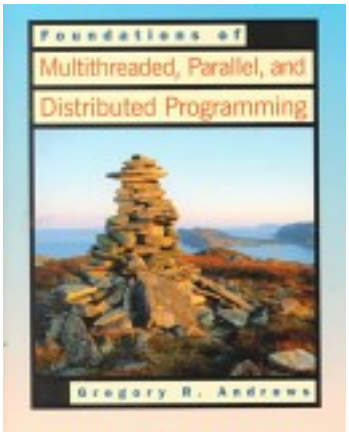

Multithreaded, Parallel, and Distributed Programming, Gregory R. Andrews, Addison-Wesley, 2000, 0201357526, 9780201357523, 664 pages. Foundations of Multithreaded, Parallel, and Distributed Programming covers, and then applies, the core concepts and techniques needed for an introductory course in this subject. Its emphasis is on the practice and application of parallel systems, using real-world examples throughout. Greg Andrews teaches the fundamental concepts of multithreaded, parallel and distributed computing and relates them to the implementation and performance processes. He presents the appropriate breadth of topics and supports these discussions with an emphasis on performance. Features Emphasizes how to solve problems, with correctness the primary concern and performance an important, but secondary, concern Includes a number of case studies which cover such topics as pthreads, MPI, and OpenMP libraries, as well as programming languages like Java, Ada, high performance Fortran, Linda, Occam, and SR Provides examples using Java syntax and discusses how Java deals with monitors, sockets, and remote method invocation Covers current programming techniques such as semaphores, locks, barriers, monitors, message passing, and remote invocation Concrete examples are executed with complete programs, both shared and distributed Sample applications include scientific computing and distributed systems 0201357526B04062001.

Foundations of Parallel Programming , David B. Skillicorn, 1994, Computers, 197 pages. This is the first comprehensive account of this new approach to the fundamentals of parallel programming..

Models for Parallel and Distributed Computation Theory, Algorithmic Techniques and Applications, R. Correa, Jun 30, 2002, Computers, 318 pages. This book focuses on advanced techniques used in the design of efficient parallel programs. It presents a wide variety of different models of parallel and distributed ....

Synchronization Algorithms And Concurrent Programming , Gadi Taubenfeld, 2006, Computers, 423 pages. Synchronization is a fundamental challenge in computer science. It is fast becoming a major performance and design issue for concurrent programming on modern architectures, and ....

Multithreaded Programming with Java Technology , Bil Lewis, Daniel J. Berg, 2000, Computers, 461 pages. Java offers powerful multithreading capabilities--even on operating systems that offer no inherent multithreading support. Now there's a complete guide to multithreaded ....

An Introduction to Parallel Computing: Design and Analysis of Algorithms, 2/e , Grama, , , . .

Parallel and distributed computing a survey of models, paradigms, and approaches, Claudia Leopold, 2001, Computers, 260 pages. An all-inclusive survey of the fundamentals of parallel and distributed computing. The use of parallel and distributed computing has increased dramatically over the past few ....

Distributed systems principles and paradigms, Andrew S. Tanenbaum, Maarten van Steen, 2007, ,

686 pages. Virtually every computing system today is part of a distributed system. Programmers, developers, and engineers need to understand the underlying principles and paradigms as ....

Principles of Concurrent And Distributed Programming , M. Ben-Ari, 2006, Computers, 361 pages. Award for Outstanding Contribution to Computer Science Education. Software today is inherently concurrent or distributed - from event-based GUI designs to operating and real ....

Concurrent programming , Alan Burns, Geoff Davies, 1993, Computers, 377 pages. This book provides a hands-on introduction to concurrent programming principles and techniques. Pascal FC (Functionally Concurrent), a teaching version of the Pascal language ....

Principles of parallel programming , Yun Calvin Lin, Lawrence Snyder, Feb 29, 2008, , 338 pages. With the rise of multi-core architecture, parallel programming is an increasingly important topic for software engineers and computer system designers. Written by well-known ....

The SR programming language concurrency in practice, Gregory R. Andrews, Ronald A. Olsson, 1993, Computers, 344 pages. .

Parallel and distributed simulation systems , Richard M. Fujimoto, Jan 3, 2000, , 300 pages. A state-of-the-art guide for the implementation of distributed simulation technology. The rapid expansion of the Internet and commodity parallel computers has made parallel and ....

Operating Systems: Internals And Design Principles, 6/E , Stallings, Sep 1, 2009, Computer science, 840 pages. .

Concurrent Programming in Java Design Principles and Patterns, Douglas Lea, 2000, Computers, 411 pages. Software -- Programming Languages..

PThreads Programming A POSIX Standard for Better Multiprocessing, Dick Buttlar, Jacqueline Farrell, Sep 1, 1996, Computers, 267 pages. In this book, realistic examples show both the situations where threading is valuable and the ways to use threads to improve the modularity and efficiency of a program. The ....

Concurrent programming , Narain Gehani, Andrew D. McGettrick, 1988, Computers, 621 pages. .

Foundations of Multithreaded, Parallel, and Distributed Programming covers, and then applies, the core concepts and techniques needed for an introductory course in this subject. Its emphasis is on the practice and application of parallel systems, using real-world examples throughout. Greg Andrews teaches the fundamental concepts of multithreaded, parallel and distributed computing and relates them to the implementation and performance processes. He presents the appropriate breadth of topics and supports these discussions with an emphasis on performance.

Gregory Andrews received a B.S. degree in Mathematics from Stanford University in 1969 and a Ph.D. degree in Computer Science from the University of Washington in 1974. From 1974-79 he was an Assistant Professor at Cornell University. Since 1979 he has been at The University of Arizona, where he is currently Professor of Computer Science. From 1986-93 he chaired the department; in 1986 he received a distinguished teaching award.

Greg has been on the editorial board of Information Processing Letters since 1979. He was the general chair of the Twelfth ACM Symposium on Operating Systems Principles in 1989 and has been on the program committees of numerous conferences. From 1988-92 he was on advisory committees for the computing directorate of the National Science Foundation. Since 1991 he has been on the Board of Directors of the Computing Research Association (CRA).

Greg's research interests include all aspects of concurrent programming. A long-term project has been the design and implementation of the SR programming language. Current work focuses on the

development of Filaments, a software package that provides efficient fine-grain parallelism on a variety of parallel machines.

The book provides all material needed for a beginner to easily acquire knowledge required for development and beginner's research in the field of parallel computation. It's written though not for a beginner in programming, solid basics and initial knowledge of OS internals are prerequisites. I found it's easy to read and understand with a mass of useful examples and with coverage of MPI and Java. This was especially important to since it bridges the theory in the earlier sections with practical implementations using production environment tools. In overall I strongly recommend it for those who are new to the field. For a more deep discussion on parallel algorithms one may want to look at F.T. Leighton's "Intro to Parallel Algorithms and Architectures: Arrays, Trees, Hypercubes" - that one is much more technical though.

Unlike many textbooks of its ilk, Dr. Andrews does use coded examples, but they are not complex code excerpts that span several pages. He does an excellent job of covering the topic in both C with Posix, Java, as well as the language he worked on MPD. Since this topic has been his primary focus he really knows the subject matter yet can explain it in a way such that anyone with moderate programming skills can grasp.

Just like his lectures, the fundamentals and theory presented in each chapter is always structured, explained, and numerous examples are given to reinforce the topics that are being taught. I would recommend this book to anyone who requires an introductory to medium exposure to the critical topic of multithreaded, parallel and distributed programming.

Chapter 5 is about monitors, and this is where condition variables are introduced (they're not treated separately as in POSIX, but the author does mention POSIX mutexes+cond.vars approach). Examples include bounded buffer, readers/writer, interval timer, sleeping barber, and a disck scheduling system. There is a section on Java and another one on pthreads;

The book is extremely useful in that it provides actual working example codes for nearly all the topics covered, in C or Java or both. The codes are also very small, in most cases less than a page. This is very important because a lot cases such as multiple readers single writers are not easy to code from scratch and could easily have synchronization problems unless one has strong overall grasp of the concept. This book does very good analysis of potential pitfalls of them. Even if you already knew the concepts, this book provides a valuable reference and code templates.

Some part of the book may seem to be pseudo code to some people since Dr. Andrews uses the MPD language (which is on top of C) for a far easier time dealing with many of the computing issues. Although I have never used to MPD language, I find the syntax useful in understanding many concepts Dr. Andrews is trying to explain, such as how to partition work into small tasks.

I've used this as a course text several times for a senior undergraduate / graduate course. What I like the most about it is that it uses a form of programming logic to give a logical justification for the correctness of the algorithms. This is a direction, however, in which it could have gone further. Having introduced some programming logic, the book could have made more use of it -- for example in the discussion of semaphore-based algorithms. The book is strongest on shared memory parallelism and not so strong on distributed computation; I wouldn't call this a fault, but rather a matter of emphasis. It is not a perfect text, but it is the best I know of. I would recommend it not only as a textbook, but also for the serious practitioner.

Nobody is smarter than you when it comes to reaching your students. You know how to convey knowledge in a way that is relevant and relatable to your class. It's the reason you always get the best out of them. And when it comes to planning your curriculum, you know which course materials express the information in the way thatâ€™s most consistent with your teaching. Thatâ€™s why we give you the option to personalize your course material using just the Pearson content you select. Take only the most applicable parts of your favorite materials and combine them in any order you want. You can even integrate your own writing if you wish. It's fast, it's easy and fewer course

materials help minimize costs for your students.

Our library is vast, and it's all at your fingertips. Create a custom book by selecting content from any of our course-specific collections. Here, you'll find chapters from Pearson titles, carefully-selected third-party content with copyright clearance, and pedagogy. Once you're satisfied with your customized book, you will have a print-on-demand book that can be purchased by students in the same way they purchase other course material.

Pearson Learning Solutions will partner with you to select or create eBooks, custom eBooks, online learning courses, resource materials, teaching content, media resources and media supplements. Simply share your course goals with our world-class experts, and they will offer you a selection of outstanding, up-to-the-minute solutions.

Pearson Learning Solutions offers a broad range of courses and custom solutions for web-enhanced, blended and online learning. Our course content is developed by a team of respected subject matter experts and experienced eLearning instructional designers. All course content is designed around specific learning objectives.

Greg received a distinguished teaching award in 1986 and a career distinguished teaching award in 2002, both from the College of Science at The University of Arizona. In 1998 he was named a Fellow of the Association for Computing Machinery (ACM). In 2010 he was given the inaugural Alumni Achievement Award by the Department of Computer Science and Engineering at the University of Washington.

Greg's research interests include all aspects of parallel and distributed computing: languages, applications, "systems" issues, and performance. A long-term project has been the design and implementation of the SR programming language and its new variant MPD. Work in the 1990s developed Filaments, a software package that supports shared memory and efficient fine-grain parallelism on a variety of parallel machines. For the past several years, he has worked with Saumya Debray and dozens of students on the Solar project (software optimization at link- and run-time), which uses binary rewriting to improve software properties---e.g., performance, size, or security---for many kinds of applications, including parallel programs, embedded systems, and OS kernels.

Greg is currently working with students on a simple, lightweight approach to writing parallel programs for multicore machines, and on a system to enable reproducibility of software experiments. He is also on the faculty advisory team for the iPlant project, a $50M grant from the NSF to build cyberinfrastructure that enables the solution of grand-challenge problems in the plant sciences.

Andrews, Foundations of Multithreaded, Parallel, and Distributed Programming, Addison Wesley, 2000. The book's Web site contains an "instructor's manual" and a variety of other materials. Also available is the MPD programming language, a variant of SR that has a syntax very close to the one used in the book. The book is now in its third printing; it has also been translated into Russian.

Andrews and Olsson, The SR Programming Language: Concurrency in Practice, Benjamin/Cummings, 1993. The book is now in its third printing. The major change from the first printing is the addition of Appendix G , which describes a few new language features. See the SR Web site for online documentation.

The MPD programming language enables students to write programs using a syntax that is very similar to the one used in the book. MPD is essentially an alternative syntax for SR. See the MPD Web site for details on the language---including a tutorial with lots of examples---and information on how to download and install the implementation.

Emphasizes how to solve problems, with correctness the primary concern and performance an important, but secondary, concern Includes a number of case studies which cover such topics as

pthreads, MPI, and OpenMP libraries, as well as programming languages like Java, Ada, high performance Fortran, Linda, Occam, and SR Provides examples using Java syntax and discusses how Java deals with monitors, sockets, and remote method invocation Covers current programming techniques such as semaphores, locks, barriers, monitors, message passing, and remote invocation Concrete examples are executed with complete programs, both shared and distributed Sample applications include scientific computing and distributed systems (less)

This textbook's material is broad enough to provide the reader with a solid foundation in this computer science specialty without leaving big holes in understanding. This book is a big improvement compared to many other textbooks and industry books in comprehensiveness of coverage of this subject. The reader will develop a wide repertoire of strategies and tactics for attacking concurrent and parallel programming problems.

Breadth of knowledge is particularly important to concurrent programming effectiveness. There are many concurrent software architectures, and languages, and hardware systems, and problem domains, and it is critically important to apply the combinations which are suitable, because some combinations are weakly compatible or completely incompatible. For example the mere knowledge of the use of threads and multi-threading is not nearly good enough. Threaded software which is intended for running on MIMD shared memory hardware is pretty much useless on distributed memory systems such as many-processor machines, clusters of machines, or SIMD systems. This book well dispels the myth sometimes encountered among professional software developers who may assume that a little hard-won experience in multi-threading a couple of real-life applications using two or three threads, or a thread-pool, makes one an expert concurrent programmer. The reality is that this is actually just scratching the surface.

The book offers an excellent understanding of the trade-offs of the suitability of many different programming languages, libraries, concurrent computer hardware architectures, and software architectures. As well as critically evaluating the suitabilities of real world languages and libraries (and compiler augmentations), the author also created for this textbook an expressive and concise notation, as well as a computer language, for efficiently describing distributed, concurrent, and parallel concepts and programs. There are of course excellent reasons to learn about sequential programming, but there are plenty of other courses and textbooks which address those things. The tools in this book enable learners to focus their time thinking about the concurrency issues without having to slog through the verbiage of irrelevant sequential-logic clutter.