



UNIX Network Programming, Volume 2: Interprocess Communications, Second Edition

By W. Richard Stevens

 Get Print Book



Download



Read Online

UNIX Network Programming, Volume 2: Interprocess Communications, Second Edition By W. Richard Stevens

Well-implemented interprocess communications (IPC) are key to the performance of virtually every non-trivial UNIX program. In UNIX Network Programming, Volume 2, Second Edition, legendary UNIX expert W. Richard Stevens presents a comprehensive guide to every form of IPC, including message passing, synchronization, shared memory, and Remote Procedure Calls (RPC). Stevens begins with a basic introduction to IPC and the problems it is intended to solve. Step-by-step you'll learn how to maximize both System V IPC and the new Posix standards, which offer dramatic improvements in convenience and performance.



[Download UNIX Network Programming, Volume 2: Interprocess C...pdf](#)



[Read Online UNIX Network Programming, Volume 2: Interprocess...pdf](#)

UNIX Network Programming, Volume 2: Interprocess Communications, Second Edition

By W. Richard Stevens

UNIX Network Programming, Volume 2: Interprocess Communications, Second Edition By W. Richard Stevens

Well-implemented interprocess communications (IPC) are key to the performance of virtually every non-trivial UNIX program. In UNIX Network Programming, Volume 2, Second Edition, legendary UNIX expert W. Richard Stevens presents a comprehensive guide to every form of IPC, including message passing, synchronization, shared memory, and Remote Procedure Calls (RPC). Stevens begins with a basic introduction to IPC and the problems it is intended to solve. Step-by-step you'll learn how to maximize both System V IPC and the new Posix standards, which offer dramatic improvements in convenience and performance.

UNIX Network Programming, Volume 2: Interprocess Communications, Second Edition By W. Richard Stevens **Bibliography**

- Sales Rank: #315658 in Books
- Published on: 1998-09-04
- Ingredients: Example Ingredients
- Original language: German
- Number of items: 1
- Dimensions: 9.50" h x 1.32" w x 7.38" l, 3.05 pounds
- Binding: Hardcover
- 592 pages

 [Download UNIX Network Programming, Volume 2: Interprocess C ...pdf](#)

 [Read Online UNIX Network Programming, Volume 2: Interprocess ...pdf](#)

Download and Read Free Online UNIX Network Programming, Volume 2: Interprocess Communications, Second Edition By W. Richard Stevens

Editorial Review

Amazon.com Review

The first volume of *Unix Network Programming, Networking APIs: Sockets and XTI* covers just about everything you need to know to get your applications to talk to other computers on a network. In this second volume, W. Richard Stevens discusses what you need to know to get your applications to talk to other applications running on your computer. There's a big difference, and Stevens covers it well.

Stevens introduces the reader to the internal structures of Posix interprocess communication (IPC) and System V (SysV) IPC; pipes and first in, first outs (FIFOs); message queues; how to lock and unlock files and records; semaphores; shared memory; and remote procedure calls (RPCs). He explains the difference between the Posix and SysV implementations of semaphores, message queues, and shared memory. There are also plenty of notes and examples for the reader.

This book is invaluable for programmers because it explains all of those little "gotchas" that always seem to pop up. In addition, the explanations of the differences between Posix IPC and SysV IPC really help readers decide which version they'd like to use for their applications. --*Doug Beaver*

From the Inside Flap

Preface

Introduction

Most nontrivial programs involve some form of IPC or Interprocess Communication. This is a natural effect of the design principle that the better approach is to design an application as a group of small pieces that communicate with each other, instead of designing one huge monolithic program. Historically, applications have been built in the following ways:

1. One huge monolithic program that does everything. The various pieces of the program can be implemented as functions that exchange information as function parameters, function return values, and global variables.
2. Multiple programs that communicate with each other using some form of IPC. Many of the standard Unix tools were designed in this fashion, using shell pipelines (a form of IPC) to pass information from one program to the next.
3. One program comprised of multiple threads that communicate with each other using some type of IPC. The term IPC describes this communication even though it is between threads and not between processes. Combinations of the second two forms of design are also possible: multiple processes, each consisting of one or more threads, involving communication between the threads within a given process and between the different processes.

What I have described is distributing the work involved in performing a given application between multiple processes and perhaps among the threads within a process. On a system containing multiple processors (CPUs), multiple processes might be able to run at the same time (on different CPUs), or the multiple threads of a given process might be able to run at the same time. Therefore, distributing an application among multiple processes or threads might reduce the amount of time required for an application to perform a given task.

This book describes four different forms of IPC in detail:

1. message passing (pipes, FIFOs, and message queues),
2. synchronization (mutexes, condition variables, read-write locks, file and record locks, and semaphores),
3. shared memory (anonymous and named), and
4. remote procedure calls (Solaris doors and Sun RPC).

This book does not cover the writing of programs that communicate across a computer network. This form of communication normally involves what is called the sockets API (application program interface) using the TCP/IP protocol suite; these topics are covered in detail in Volume 1 of this series Stevens 1998.

One could argue that single-host or nonnetworked IPC (the subject of this volume) should not be used and instead all applications should be written as distributed applications that run on various hosts across a network. Practically, however, single-host IPC is often much faster and sometimes simpler than communicating across a network. Techniques such as shared memory and synchronization are normally available only on a single host, and may not be used across a network. Experience and history have shown a need for both nonnetworked IPC (this volume) and IPC across a network (Volume 1 of this series).

This current volume builds on the foundation of Volume 1 and my other four books, which are abbreviated throughout this text as follows:

UNPv1: UNIX Network Programming, Volume 1 Stevens 1998,

APUE: Advanced Programming in the UNIX Environment Stevens 1992,

TCPv1: TCP/IP Illustrated, Volume 1 Stevens 1994,

TCPv2: TCP/IP Illustrated, Volume 2 Wright and Stevens 1995, and

TCPv3: TCP/IP Illustrated, Volume 3 Stevens 1996.

Although covering IPC in a text with "network programming" in the title might seem odd, IPC is often used in networked applications. As stated in the Preface of the 1990 edition of UNIX Network Programming, "A requisite for understanding how to develop software for a network is an understanding of interprocess communication (IPC)."

Changes from the First Edition

This volume is a complete rewrite and expansion of Chapters 3 and 18 from the 1990 edition of UNIX Network Programming. Based on a word count, the material has expanded by a factor of five. The following are the major changes with this new edition:

In addition to the three forms of "System V IPC" (message queues, semaphores, and shared memory), the newer Posix functions that implement these three types of IPC are also covered. (I say more about the Posix family of standards in Section 1.7.) In the coming years, I expect a movement to the Posix IPC functions, which have several advantages over their System V counterparts.

The Posix functions for synchronization are covered: mutex locks, condition variables, and read-write locks. These can be used to synchronize either threads or processes and are often used when accessing shared memory.

This volume assumes a Posix threads environment (called "Pthreads"), and many of the examples are built using multiple threads instead of multiple processes.

The coverage of pipes, FIFOs, and record locking focuses on their Posix definitions.

In addition to describing the IPC facilities and showing how to use them, I also develop implementations of Posix message queues, read-write locks, and Posix semaphores (all of which can be implemented as user libraries). These implementations can tie together many different features (e.g., one implementation of Posix semaphores uses mutexes, condition variables, and memory-mapped I/O) and highlight conditions that must often be handled in our applications (such as race conditions, error handling, memory leaks, and variable-length argument lists). Understanding an implementation of a certain feature often leads to a greater knowledge of how to use that feature.

The RPC coverage focuses on the Sun RPC package. I precede this with a description of the new Solaris

doors API, which is similar to RPC but on a single host. This provides an introduction to many of the features that we need to worry about when calling procedures in another process, without having to worry about any networking details.

Readers

This text can be used either as a tutorial on IPC, or as a reference for experienced programmers. The book is divided into four main parts:

message passing,
synchronization,
shared memory, and
remote procedure calls

but many readers will probably be interested in specific subsets. Most chapters can be read independently of others, although Chapter 2 summarizes many features common to all the Posix IPC functions, Chapter 3 summarizes many features common to all the System V IPC functions, and Chapter 12 is an introduction to both Posix and System V shared memory. All readers should read Chapter 1, especially Section 1.6, which describes some wrapper functions used throughout the text. The Posix IPC chapters are independent of the System V IPC chapters, and the chapters on pipes, FIFOs, and record locking belong to neither camp. The two chapters on RPC are also independent of the other IPC techniques.

To aid in the use as a reference, a thorough index is provided, along with summaries on the end papers of where to find detailed descriptions of all the functions and structures. To help those reading topics in a random order, numerous references to related topics are provided throughout the text.

Source Code and Errata Availability

The source code for all the examples that appear in this book is available from the author's home page (listed at the end of this Preface). The best way to learn the IPC techniques described in this book is to take these programs, modify them, and enhance them. Actually writing code of this form is the only way to reinforce the concepts and techniques. Numerous exercises are also provided at the end of each chapter, and most answers are provided in Appendix D.

A current errata for this book is also available from the author's home page.

Acknowledgments

Although the author's name is the only one to appear on the cover, the combined effort of many people is required to produce a quality text book. First and foremost is the author's family, who put up with the long and weird hours that go into writing a book. Thank you once again, Sally, Bill, Ellen, and David.

My thanks to the technical reviewers who provided invaluable feedback (135 printed pages) catching lots of errors, pointing out areas that needed more explanation, and suggesting alternative presentations, wording, and coding: Gavin Bowe, Allen Briggs, Dave Butenhof, Wan-Teh Chang, Chris Cleeland, Bob Friesenhahn, Andrew Gierth, Scott Johnson, Marty Leisner, Larry McVoy, Craig Metz, Bob Nelson, Steve Rago, Jim Reid, Swamy K. Sitarama, Jon C. Snader, Ian Lance Taylor, Rich Teer, and Andy Tucker.

The following people answered email questions of mine, in some cases many questions, all of which improved the accuracy and presentation of the text: David Bausum, Dave Butenhof, Bill Gallmeister, Mukesh Kacker, Brian Kernighan, Larry McVoy, Steve Rago, Keith Skowran, Bart Smaalders, Andy Tucker, and John Wait.

A special thanks to Larry Rafsky at GSquared, for lots of things. My thanks as usual to the National Optical Astronomy Observatories (NOAO), Sidney Wolff, Richard Wolff, and Steve Grandi, for providing access to their networks and hosts. Jim Bound, Matt Thomas, Mary Clouter, and Barb Glover of Digital Equipment Corp. provided the Alpha system used for most of the examples in this text. A subset of the code in this book was tested on other Unix systems: my thanks to Michael Johnson of Red Hat Software for providing the

latest releases of Red Hat Linux, and to Dave Marquardt and Jessie Haug of IBM Austin for an RS/6000 system and access to the latest releases of AIX.

My thanks to the wonderful staff at Prentice Hall-my editor Mary Franz, along with Noreen Regina, Sophie Papanikolaou, and Patti Guerrieri-for all their help, especially in bringing everything together on a tight schedule.

Colophon

I produced camera-ready copy of the book (PostScript), which was then typeset for the final book. The formatting system used was James Clark's wonderful groff package, on a SparcStation running Solaris 2.6. (Reports of troff's death are greatly exaggerated.) I typed in all 138,897 words using the vi editor, created the 72 illustrations using the gpic program (using many of Gary Wright's macros), produced the 35 tables using the gtbl program, performed all the indexing (using a set of awk scripts written by Jon Bentley and Brian Kernighan), and did the final page layout. Dave Hanson's loom program, the GNU indent program, and some scripts by Gary Wright were used to include the 8,046 lines of C source code in the book.

I welcome email from any readers with comments, suggestions, or bug fixes.

Tucson, Arizona

W. Richard Stevens

July 1998

rstevens@kohala

kohala/~rstevens

From the Back Cover

8108A-2

Don't miss the rest of the series!

- Vol. 1, Networking APIs: Sockets and XTI
- Vol. 3, Applications (forthcoming)

The only guide to UNIX(r) interprocess communications you'll ever need!

Well-implemented interprocess communications (IPC) are key to the performance of virtually every non-trivial UNIX program. In *UNIX Network Programming, Volume 2, Second Edition*, legendary UNIX expert W. Richard Stevens presents a comprehensive guide to every form of IPC, including message passing, synchronization, shared memory, and Remote Procedure Calls (RPC).

Stevens begins with a basic introduction to IPC and the problems it is intended to solve. Step-by-step you'll learn how to maximize both System V IPC and the new Posix standards, which offer dramatic improvements in convenience and performance. You'll find extensive coverage of Pthreads, with many examples reflecting multiple threads instead of multiple processes. Along the way, you'll master every current IPC technique and technology, including:

- Pipes and FIFOs.
- Posix and System V Message Queues
- Mutexes and Condition Variables
- Read-Write Locks
- Record Locking
- Posix and System V Semaphores

- Posix and System V Shared Memory
- Solaris Doors and Sun RPC
- Performance Measurements of IPC Techniques

If you've read Stevens' best-selling first edition of UNIX Network Programming, this book expands its IPC coverage by a factor of five! You won't just learn about IPC "from the outside." You'll actually create implementations of Posix message queues, read-write locks, and semaphores, gaining an in-depth understanding of these capabilities you simply can't get anywhere else.

The book contains extensive new source code-all carefully optimized and available on the Web. You'll even find a complete guide to measuring IPC performance with message passing bandwidth and latency programs, and thread and process synchronization programs.

The better you understand IPC, the better your UNIX software will run. One book contains all you need to know: UNIX Network Programming, Volume 2, Second Edition.

Users Review

From reader reviews:

Russell Belcher:

Why don't make it to be your habit? Right now, try to ready your time to do the important work, like looking for your favorite publication and reading a book. Beside you can solve your problem; you can add your knowledge by the guide entitled UNIX Network Programming, Volume 2: Interprocess Communications, Second Edition. Try to make book UNIX Network Programming, Volume 2: Interprocess Communications, Second Edition as your pal. It means that it can for being your friend when you sense alone and beside that of course make you smarter than ever. Yeah, it is very fortunated for yourself. The book makes you far more confidence because you can know everything by the book. So , let's make new experience and knowledge with this book.

Emma Patterson:

The book UNIX Network Programming, Volume 2: Interprocess Communications, Second Edition has a lot details on it. So when you check out this book you can get a lot of help. The book was compiled by the very famous author. The author makes some research prior to write this book. That book very easy to read you may get the point easily after looking over this book.

Jack Williams:

Playing with family within a park, coming to see the sea world or hanging out with good friends is thing that usually you may have done when you have spare time, and then why you don't try point that really opposite from that. 1 activity that make you not sensation tired but still relaxing, trilling like on roller coaster you already been ride on and with addition associated with. Even you love UNIX Network Programming, Volume 2: Interprocess Communications, Second Edition, you can enjoy both. It is very good combination right, you still need to miss it? What kind of hangout type is it? Oh can happen its mind hangout people. What? Still don't obtain it, oh come on its known as reading friends.

Catherine Stoltenberg:

Are you kind of active person, only have 10 as well as 15 minute in your day to upgrading your mind talent or thinking skill even analytical thinking? Then you are having problem with the book compared to can satisfy your short time to read it because this all time you only find book that need more time to be study. UNIX Network Programming, Volume 2: Interprocess Communications, Second Edition can be your answer because it can be read by you actually who have those short time problems.

Download and Read Online UNIX Network Programming, Volume 2: Interprocess Communications, Second Edition By W. Richard Stevens #NML1X8V4H9J

Read UNIX Network Programming, Volume 2: Interprocess Communications, Second Edition By W. Richard Stevens for online ebook

UNIX Network Programming, Volume 2: Interprocess Communications, Second Edition By W. Richard Stevens Free PDF d0wnl0ad, audio books, books to read, good books to read, cheap books, good books, online books, books online, book reviews epub, read books online, books to read online, online library, greatbooks to read, PDF best books to read, top books to read UNIX Network Programming, Volume 2: Interprocess Communications, Second Edition By W. Richard Stevens books to read online.

Online UNIX Network Programming, Volume 2: Interprocess Communications, Second Edition By W. Richard Stevens ebook PDF download

UNIX Network Programming, Volume 2: Interprocess Communications, Second Edition By W. Richard Stevens Doc

UNIX Network Programming, Volume 2: Interprocess Communications, Second Edition By W. Richard Stevens Mobipocket

UNIX Network Programming, Volume 2: Interprocess Communications, Second Edition By W. Richard Stevens EPub