

大学计算机教育国外著名教材、教参系列 (影印版)

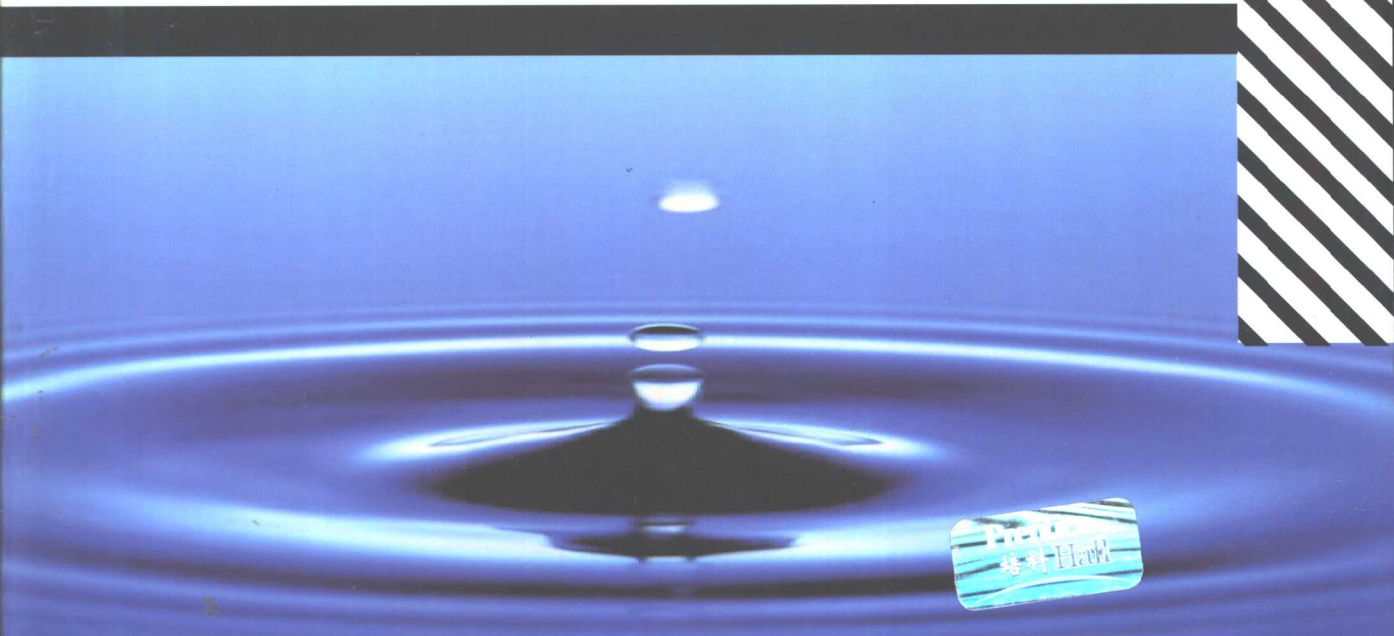
UNIX NETWORK PROGRAMMING
Volume 2: Interprocess Communications

W. RICHARD STEVENS

UNIX

网络编程卷2：进程间通信

(第2版)



清华大学出版社

<http://www.tup.tsinghua.edu.cn>



PRENTICE-HALL, INC.

<http://www.prenhall.com>

UNIX Network Programming

Volume 2

Second Edition

Interprocess Communications

UNIX 网络编程

卷 2

进程间通信

第 2 版

W. Richard Stevens

清华大学出版社

Prentice Hall, Inc.

(京) 新登字 158 号

UNIX Network Programming Volume 2: Interprocess Communications 2nd ed.
W. Richard Stevens

Copyright © 1999 by Prentice Hall PTR.
Original English Language Edition Published by Prentice Hall PTR
All Rights Reserved.
For sale in Mainland China only.

本书影印版由 Prentice Hall 出版公司授权清华大学出版社在中国境内（不包括香港特别行政区、澳门特别行政区和台湾地区）独家出版、发行。
未经出版者书面许可，不得以任何方式复制或抄袭本书的任何部分。

本书封面贴有 **Prentice Hall** 激光防伪标签，无标签者不得销售。
北京市版权局著作权合同登记号：图字：01-2002-0656

书 名：UNIX 网络编程 卷 2：进程间通信（第 2 版）
作 者：W. Richard Stevens
出版者：清华大学出版社（北京清华大学学研大厦，邮编 100084）
[http:// www.tup.tsinghua.edu.cn](http://www.tup.tsinghua.edu.cn)
印刷者：清华大学印刷厂
发行者：新华书店总店北京发行所
开 本：787×960 1/16 印张：36.25
版 次：2002 年 4 月第 1 版 2002 年 4 月第 1 次印刷
书 号：ISBN 7-302-05355-3/TP·3147
印 数：0001~4000
定 价：47.00 元

出版说明

进入 21 世纪, 世界各国的经济、科技以及综合国力的竞争将更加激烈。竞争的中心无疑是对人才的争夺。谁拥有大量高素质的人才, 谁就能在竞争中取得优势。高等教育, 作为培养高素质人才的事业, 必然受到高度重视。目前我国高等教育的教材更新较慢, 为了加快教材的更新频率, 教育部正在大力促进我国高校采用国外原版教材。

清华大学出版社从 1996 年开始, 与国外著名出版公司合作, 影印出版了“大学计算机教育丛书(影印版)”等一系列引进图书, 受到了国内读者的欢迎和支持。跨入 21 世纪, 我们本着为我国高等教育教材建设服务的初衷, 在已有的基础上, 进一步扩大选题内容, 改变图书开本尺寸, 一如既往地请有关专家挑选适用于我国高校本科及研究生计算机教育的国外经典教材或著名教材以及教学参考书, 组成本套“大学计算机教育国外著名教材、教参系列(影印版)”, 以飨读者。深切期盼读者及时将使用本系列教材、教参的效果和意见反馈给我们。更希望国内专家、教授积极向我们推荐国外计算机教育的优秀教材, 以利我们把“大学计算机教育国外著名教材、教参系列(影印版)”做得更好, 更适合高校师生的需要。

计算机引进版图书编辑室
2002.3

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
July 1998

W. Richard Stevens
rstevens@kohala.com
<http://www.kohala.com/~rstevens>

Abbreviated Table of Contents

Part 1. Introduction	1
Chapter 1. Introduction	3
Chapter 2. Posix IPC	19
Chapter 3. System V IPC	27
Part 2. Message Passing	41
Chapter 4. Pipes and FIFOs	43
Chapter 5. Posix Message Queues	75
Chapter 6. System V Message Queues	129
Part 3. Synchronization	157
Chapter 7. Mutexes and Condition Variables	159
Chapter 8. Read-Write Locks	177
Chapter 9. Record Locking	193
Chapter 10. Posix Semaphores	219
Chapter 11. System V Semaphores	281
Part 4. Shared Memory	301
Chapter 12. Shared Memory Introduction	303
Chapter 13. Posix Shared Memory	325
Chapter 14. System V Shared Memory	343
Part 5. Remote Procedure Calls	353
Chapter 15. Doors	355
Chapter 16. Sun RPC	399
Appendix A. Performance Measurements	457
Appendix B. A Threads Primer	501
Appendix C. Miscellaneous Source Code	505
Appendix D. Solutions to Selected Exercises	515

Table of Contents

Preface		xiii
Part 1. Introduction		1
Chapter 1. Introduction		3
1.1	Introduction	3
1.2	Processes, Threads, and the Sharing of Information	5
1.3	Persistence of IPC Objects	6
1.4	Name Spaces	7
1.5	Effect of fork, exec, and exit on IPC Objects	9
1.6	Error Handling: Wrapper Functions	11
1.7	Unix Standards	13
1.8	Road Map to IPC Examples in the Text	15
1.9	Summary	16
Chapter 2. Posix IPC		19
2.1	Introduction	19
2.2	IPC Names	19
2.3	Creating and Opening IPC Channels	22
2.4	IPC Permissions	25
2.5	Summary	26

Chapter 3.	System V IPC	27
3.1	Introduction	27
3.2	key_t Keys and ftok Function	28
3.3	ipc_perm Structure	30
3.4	Creating and Opening IPC Channels	30
3.5	IPC Permissions	32
3.6	Identifier Reuse	34
3.7	ipcs and ipcrm Programs	36
3.8	Kernel Limits	36
3.9	Summary	38
 Part 2. Message Passing		 41
Chapter 4.	Pipes and FIFOs	43
4.1	Introduction	43
4.2	A Simple Client–Server Example	43
4.3	Pipes	44
4.4	Full-Duplex Pipes	50
4.5	popen and pclose Functions	52
4.6	FIFOs	54
4.7	Additional Properties of Pipes and FIFOs	58
4.8	One Server, Multiple Clients	60
4.9	Iterative versus Concurrent Servers	66
4.10	Streams and Messages	67
4.11	Pipe and FIFO Limits	72
4.12	Summary	73
Chapter 5.	Posix Message Queues	75
5.1	Introduction	75
5.2	mq_open, mq_close, and mq_unlink Functions	76
5.3	mq_getattr and mq_setattr Functions	79
5.4	mq_send and mq_receive Functions	82
5.5	Message Queue Limits	86
5.6	mq_notify Function	87
5.7	Posix Realtime Signals	98
5.8	Implementation Using Memory-Mapped I/O	106
5.9	Summary	126
Chapter 6.	System V Message Queues	129
6.1	Introduction	129
6.2	msgget Function	130
6.3	msgsnd Function	131
6.4	msgrcv Function	132
6.5	msgctl Function	134
6.6	Simple Programs	135
6.7	Client–Server Example	140
6.8	Multiplexing Messages	142

6.9	Message Queues with <code>select</code> and <code>poll</code>	151
6.10	Message Queue Limits	152
6.11	Summary	155

Part 3. Synchronization **157**

Chapter 7. Mutexes and Condition Variables **159**

7.1	Introduction	159
7.2	Mutexes: Locking and Unlocking	159
7.3	Producer–Consumer Problem	161
7.4	Locking versus Waiting	165
7.5	Condition Variables: Waiting and Signaling	167
7.6	Condition Variables: Timed Waits and Broadcasts	171
7.7	Mutexes and Condition Variable Attributes	172
7.8	Summary	174

Chapter 8. Read–Write Locks **177**

8.1	Introduction	177
8.2	Obtaining and Releasing Read–Write Locks	178
8.3	Read–Write Lock Attributes	179
8.4	Implementation Using Mutexes and Condition Variables	179
8.5	Thread Cancellation	187
8.6	Summary	192

Chapter 9. Record Locking **193**

9.1	Introduction	193
9.2	Record Locking versus File Locking	197
9.3	Posix <code>fcntl</code> Record Locking	199
9.4	Advisory Locking	203
9.5	Mandatory Locking	204
9.6	Priorities of Readers and Writers	207
9.7	Starting Only One Copy of a Daemon	213
9.8	Lock Files	214
9.9	NFS Locking	216
9.10	Summary	216

Chapter 10. Posix Semaphores **219**

10.1	Introduction	219
10.2	<code>sem_open</code> , <code>sem_close</code> , and <code>sem_unlink</code> Functions	225
10.3	<code>sem_wait</code> and <code>sem_trywait</code> Functions	226
10.4	<code>sem_post</code> and <code>sem_getvalue</code> Functions	227
10.5	Simple Programs	228
10.6	Producer–Consumer Problem	233
10.7	File Locking	238
10.8	<code>sem_init</code> and <code>sem_destroy</code> Functions	238
10.9	Multiple Producers, One Consumer	242
10.10	Multiple Producers, Multiple Consumers	245

10.11	Multiple Buffers	249	
10.12	Sharing Semaphores between Processes	256	
10.13	Semaphore Limits	257	
10.14	Implementation Using FIFOs	257	
10.15	Implementation Using Memory-Mapped I/O	262	
10.16	Implementation Using System V Semaphores	271	
10.17	Summary	278	
Chapter 11.	System V Semaphores		281
11.1	Introduction	281	
11.2	semget Function	282	
11.3	semop Function	285	
11.4	semctl Function	287	
11.5	Simple Programs	289	
11.6	File Locking	294	
11.7	Semaphore Limits	296	
11.8	Summary	300	
Part 4.	Shared Memory		301
Chapter 12.	Shared Memory Introduction		303
12.1	Introduction	303	
12.2	mmap, munmap, and msync Functions	307	
12.3	Increment Counter in a Memory-Mapped File	311	
12.4	4.4BSD Anonymous Memory Mapping	315	
12.5	SVR4 /dev/zero Memory Mapping	316	
12.6	Referencing Memory-Mapped Objects	317	
12.7	Summary	322	
Chapter 13.	Posix Shared Memory		325
13.1	Introduction	325	
13.2	shm_open and shm_unlink Functions	326	
13.3	ftruncate and fstat Functions	327	
13.4	Simple Programs	328	
13.5	Incrementing a Shared Counter	333	
13.6	Sending Messages to a Server	336	
13.7	Summary	342	
Chapter 14.	System V Shared Memory		343
14.1	Introduction	343	
14.2	shmget Function	343	
14.3	shmat Function	344	
14.4	shmdt Function	345	
14.5	shmctl Function	345	
14.6	Simple Programs	346	
14.7	Shared Memory Limits	349	
14.8	Summary	351	

Part 5. Remote Procedure Calls	353
Chapter 15. Doors	355
15.1 Introduction	355
15.2 door_call Function	361
15.3 door_create Function	363
15.4 door_return Function	364
15.5 door_cred Function	365
15.6 door_info Function	365
15.7 Examples	366
15.8 Descriptor Passing	379
15.9 door_server_create Function	384
15.10 door_bind, door_unbind, and door_revoke Functions	390
15.11 Premature Termination of Client or Server	390
15.12 Summary	397
Chapter 16. Sun RPC	399
16.1 Introduction	399
16.2 Multithreading	407
16.3 Server Binding	411
16.4 Authentication	414
16.5 Timeout and Retransmission	417
16.6 Call Semantics	422
16.7 Premature Termination of Client or Server	424
16.8 XDR: External Data Representation	426
16.9 RPC Packet Formats	444
16.10 Summary	449
Epilogue	453
Appendix A. Performance Measurements	457
A.1 Introduction	457
A.2 Results	458
A.3 Message Passing Bandwidth Programs	467
A.4 Message Passing Latency Programs	480
A.5 Thread Synchronization Programs	486
A.6 Process Synchronization Programs	497
Appendix B. A Threads Primer	501
B.1 Introduction	501
B.2 Basic Thread Functions: Creation and Termination	502
Appendix C. Miscellaneous Source Code	505
C.1 unpipc.h Header	505
C.2 config.h Header	509
C.3 Standard Error Functions	510

Appendix D. Solutions to Selected Exercises	515
Bibliography	535
Index	539

Part 1

Introduction