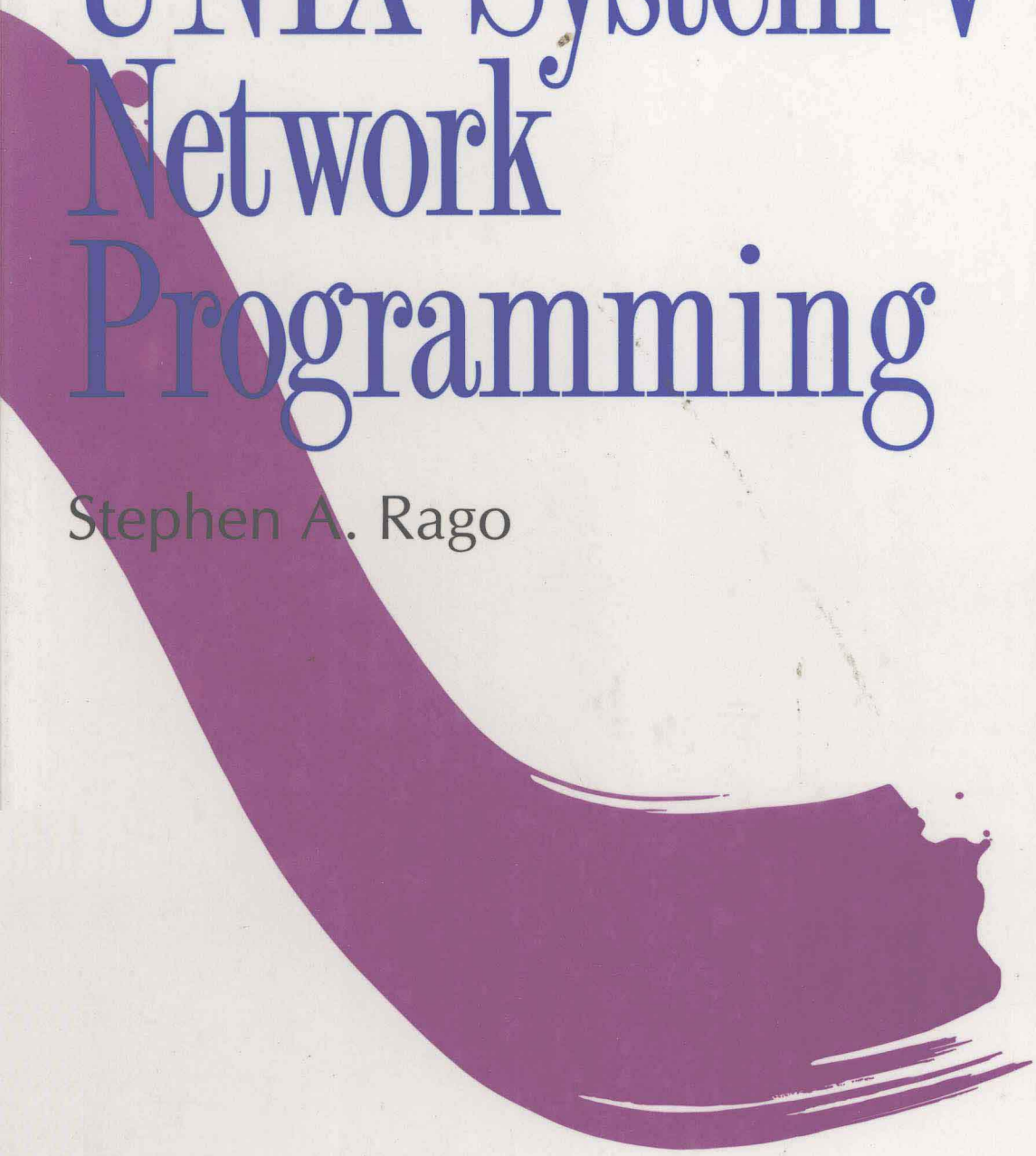
UNIX® System V Network Programming

Stephen A. Rago



ADDISON-WESLEY PROFESSIONAL COMPUTING SERIES

UNIX® System V Network Programming

Stephen A. Rago

ADDISON-WESLEY
An imprint of Addison Wesley Longman, Inc.
Reading, Massachusetts Harlow, England Menlo Park, California
Berkeley, California Don Mills, Ontario Sydney
Bonn Amsterdam Tokyo Mexico City

The publisher offers discounts on this book when ordered in quantity for special sales. For more information please contact:

Corporate & Professional Publishing Group
Addison-Wesley Publishing Company
One Jacob Way
Reading, Massachusetts 01867

Library of Congress Cataloging-in-Publication Data

Rago, Stephen A.
UNIX System V network programming/Stephen A. Rago.
p. cm. (Addison-Wesley professional computing series)
Includes index.
ISBN 0-201-56318-5 (hard)
1. Operating systems (Computers). 2. Unix System V (Computer file). 3. Computer networks. I. Title. II. Series.
QA76.76.063R34 1993
005.7' 11—dc20 92-45276
CIP

Text printed on recycled and acid-free paper

8 9 10111213 MA 00 99 98 97

8th Printing November, 1997

UNIX® System V Network Programming

Addison-Wesley Professional Computing Series

Brian W. Kernighan, Consulting Editor

Ken Arnold/John Peyton, *A C User's Guide to ANSI C*
David R. Butenhof, *Programming with POSIX® Threads*
Tom Cargill, *C++ Programming Style*
William R. Cheswick/Steven M. Bellovin, *Firewalls and Internet Security: Repelling the Wily Hacker*
David A. Curry, *UNIX® System Security: A Guide for Users and System Administrators*
Erich Gamma/Richard Helm/Ralph Johnson/John Vlissides, *Design Patterns: Elements of Reusable Object-Oriented Software*
Erich Gamma/Richard Helm/Ralph Johnson/John Vlissides, *Design Patterns CD: Elements of Reusable Object-Oriented Software*
David R. Hanson, *C Interfaces and Implementations: Techniques for Creating Reusable Software*
Mark Harrison/Michael McLennan, *Effective Tcl/Tk Programming: Writing Better Programs with Tcl and Tk*
S. Keshav, *An Engineering Approach to Computer Networking: ATM Networks, The Internet, and the Telephone Network*
John Lakos, *Large-Scale C++ Software Design*
Scott Meyers, *Effective C++, Second Edition: 50 Specific Ways to Improve Your Programs and Designs*
Scott Meyers, *More Effective C++: 35 New Ways to Improve Your Programs and Designs*
Robert B. Murray, *C++ Strategies and Tactics*
David R. Musser/Atul Saini, *STL Tutorial and Reference Guide: C++ Programming with the Standard Template Library*
John K. Ousterhout, *Tcl and the Tk Toolkit*
Craig Partridge, *Gigabit Networking*
J. Stephen Pendergrast Jr., *Desktop KornShell Graphical Programming*
Radia Perlman, *Interconnections: Bridges and Routers*
David M. Piscitello/A. Lyman Chapin, *Open Systems Networking: TCP/IP and OSI*
Stephen A. Rago, *UNIX® System V Network Programming*
Curt Schimmel, *UNIX® Systems for Modern Architectures: Symmetric Multiprocessing and Caching for Kernel Programmers*
W. Richard Stevens, *Advanced Programming in the UNIX® Environment*
W. Richard Stevens, *TCP/IP Illustrated, Volume 1: The Protocols*
W. Richard Stevens, *TCP/IP Illustrated, Volume 3: TCP for Transactions, HTTP, NNTP, and the UNIX® Domain Protocols*
Gary R. Wright/W. Richard Stevens, *TCP/IP Illustrated, Volume 2: The Implementation*

Preface

This book is for programmers who are interested in learning how to use the networking interfaces in UNIX System V Release 4 (SVR4). We use real-life examples to demonstrate how interfaces are used and techniques are applied. All too often in the workplace we find ourselves faced with new assignments for which we have little background. In these situations, we must educate ourselves as quickly as possible so that we can competently undertake the task at hand. Although technical manuals usually provide the information necessary to complete a task, they often lack the background, motivation, and explanation that help us to understand more clearly what we're doing and why we're doing it.

Intended as a practical reference, this book contains very little coverage of theory, and details better dealt with through manual pages are omitted, although references are used liberally. It could, however, be used to complement a graduate or advanced undergraduate course in networking.

As a prerequisite to reading this book, you should be familiar with the UNIX environment and the C programming language so that the examples can be understood. Some background in data structures and algorithms would be helpful, but is not required.

References to SVR4 manual pages are in the running text, appearing as the command name or function name, followed by the section of the manual in which the page is found, as in `open(2)`. Here, we are referring to the `open` manual page in Section 2 of the system manuals.

Originally, there was only one manual for the system. With the introduction of each new release of the system, the manual grew in size until it had to be split up into separate manuals. In UNIX System V Release 3, there was one manual for users, one manual for programmers, and one manual for system administrators.

In SVR4, however, the manual pages were redistributed by functional area. The user commands are no longer in a single manual, nor can you find all the programming interfaces in one place. This new organization has proven difficult to navigate by novices and experts alike. The following summary should aid in the process of locating the desired manual pages.

Programmer's Reference Manual
- (1) Commands relating to source code management, compilation, and loading
- (2) System calls
- (3, 3C, 3S, 3E, 3G, 3M, 3X) Most library routines
- (4) File formats
- (5) Miscellany (commonly used constants, data structures, and macros)

Programmer's Guide: Networking Interfaces
- (1, 1M) Networking commands
- (3, 3C, 3N) Network-related library routines
- (4) Network-related file formats
- (5) Miscellany, including network-related environment variables
- (7) Networking drivers and modules

Programmer's Guide: STREAMS
- (1, 1M) STREAMS-related commands
- (2) STREAMS-specific system calls
- (3C) STREAMS-specific library routines
- (7) STREAMS modules and drivers

User's Reference Manual
- (1) Commands any user might want to run

System Administrator's Reference Manual
- (1M) Administrative commands
- (4) Administrative file formats
- (5) Miscellaneous facilities
- (7) Special files (devices)
- (8) Administrative procedures

You might find it helpful if these manuals are close by when you read this book.

Background

The first standard network interface incorporated in the UNIX system was the socket mechanism. This mechanism was provided in the 4.2 release of the Berkeley Software Distribution (BSD) version of the UNIX operating system from the University of California at Berkeley. With it was an implementation of the Internet protocol suite (TCP, UDP, IP, et al.). These became available in 1983.

AT&T did not address standard networking interfaces in System V until 1985, when it ported Dennis Ritchie's Streams mechanism from the Version 8 Research UNIX System to UNIX System V Release 2.0p, the unreleased predecessor to System V Release 3.0 (SVR3). With the release of SVR3 in 1986, STREAMS, the framework for networking in System V, became generally available, along with the Transport Layer Interface (TLI) library. Ironically, SVR3 was released without including any networking protocols.

In 1988, X/OPEN, a consortium dedicated to enhancing application portability through standards endorsements, specified its own transport layer interface library, based on AT&T's TLI library. The X/OPEN specification, called the X/OPEN Transport Interface (XTI), is effectively a superset of TLI. In 1990 the Portable

Operating System Interface (POSIX) committee of the Institute of Electrical and Electronics Engineers (IEEE) created the 1003.12 working group to standardize portable networking interfaces for application programs. As of this writing, the 1003.12 working group's efforts are still underway, but it looks as though both sockets and XTI will be included in the standard.

SVR4 is unique in that it includes support for many standards in one operating system. Unlike other versions of UNIX that support dual-universe environments, SVR4 provides applications with one environment consisting of features from previous versions of the System V, SunOS, BSD, Xenix, SCO, and Research UNIX systems, as well as some new features of its own. Support for POSIX 1003.1 (the system application programming interface) is also provided. The major networking interfaces provided include STREAMS, TLI, sockets, and remote procedure calls.

Organization

The material covered in this book pertains mainly to SVR4, although some features were present in earlier releases of UNIX System V. This book is divided into four sections: background material, user-level network programming, kernel-level network programming, and a design example.

Both user-level and kernel-level networking components are described to present a complete picture of network programming in UNIX System V. Although not everyone will be interested in both environments, knowledge of one environment makes programming in the other easier. Instead of just blindly following the instructions in the manuals, it enables the programmer to understand the effects of his or her actions and make better design decisions.

The first two chapters provide some background that will make the rest of the book more useful to readers with less experience. More experienced readers can skip these introductory chapters without much loss of context. Chapter 1 provides a brief introduction to networking concepts, and Chapter 2 provides an overview of application programming in the UNIX System V environment. In particular, Chapter 2 contains example functions that are used throughout the rest of this text. If you skip Chapter 2, you might want to refer back to individual examples as you come across these functions in later chapters.

Chapter 3 is the first chapter concerned with network programming per se. It covers the STREAMS programming environment. Since the STREAMS mechanism is the basis for most of the communication facilities in System V, understanding its services and system call interface is a prerequisite to discussing any System V networking facility.

Chapter 4 covers the Transport Layer Interface library. This is the interface applications use to access the services provided by the transport layer of a computer network. Emphasis is placed on application design to support network independence.

Chapter 5 describes the network selection and name-to-address translation facilities, which further extend the ability of a programmer to design network-independent applications. Chapter 6 covers the network listener process. Using the

listener simplifies the design of server processes. The Service Access Facility (SAF), the administrative framework in which the listener operates, is also discussed.

Chapter 7 gives a brief description of the BSD socket interface and its corresponding implementation in SVR4. The socket and TLI mechanisms are contrasted and compared. Chapter 8 discusses remote procedure calls and the external data representation used to develop distributed applications. This ends the user-level section of the text.

The next four chapters are dedicated to kernel-level network programming. Chapter 9 describes the kernel environment, its utility routines, and the interfaces to the STREAMS environment. Chapter 10 describes how to write STREAMS drivers, centering around the design of a simple Ethernet driver. Chapter 11 describes how to write STREAMS modules, centering around the design of a module that can be used to emulate a terminal over a network connection. Chapter 12 describes how to write STREAMS multiplexing drivers. It uses a simple connection-oriented transport provider as a detailed example.

Finally, the last section of the book, Chapter 13, covers the design of a SLIP package for SVR4, including both the user-level and kernel-level components. It illustrates the application of much from the preceding 12 chapters and, in essence, ties the book together.

Much of the interesting material lies in the examples. You are encouraged to work through each until it is understood. Source code for the examples is available via anonymous FTP from the host `ftp.uu.net` in the file `published/books/rago.netprog.tar.Z`. If you don't have direct access to the Internet, you can use `uucp` to copy the source to your machine as follows:

```
uucp uunet!~/published/books/rago.netprog.tar.Z /tmp
```

(This will place a copy of `rago.netprog.tar.Z` in `/tmp` on your system.) If you have any comments, questions, or bug reports, please send electronic mail to `sar@plc.com`.

Acknowledgements

This book was produced on an Intel i386-based system running UNIX System V Release 4.0, Version 3. The text editor `sam` was used to create and update the text. The pictures were created with `xcip`, a newer version of `cip`, on an AT&T 630MTG terminal. The output for the book was produced with `eqn`, `tbl`, `pic`, `troff`, and `dpost` from the Documenter's WorkBench, Version 3.2.

I would like to thank the following reviewers for their invaluable input: Steve Albert (Unix System Laboratories), Maury Bach (IBM Scientific and Technical Center), George Bittner (Programmed Logic Corporation), Steve Buroff (AT&T Bell Labs), Jeff Gitlin (AT&T), Ron Gomes (Morgan Stanley & Company), Peter Honeyman (University of Michigan), Brian Kernighan (AT&T Bell Labs), Dave Olander (Unix System Laboratories), Dennis Ritchie (AT&T Bell Labs), Michael Scheer (Plexus Systems), Douglas Schmidt (University of California, Irvine), Rich Stevens (independent consultant), and Graham Wheeler (Aztec Information Management). In particular, both Brian Kernighan and Rich Stevens read every chapter and freely

shared their knowledge, experience, and formatting macros and shell scripts. They have greatly increased the quality of the book.

Many people helped by answering questions where written history was vague or incomplete. In addition to the reviewers, this group includes Guy Harris (Auspex Systems), Bob Israel (Epoch Systems), Hari Pulijal (Unix System Laboratories), Usha Pulijal (Unix System Laboratories), Glenn Skinner (SunSoft), Ken Thompson (AT&T Bell Labs), and Larry Wehr (AT&T Bell Labs).

Rich Drechsler (AT&T Bell Labs) provided the PostScript program that increased the width of the constant-width font used throughout this book. Both he and Len Rago (AT&T Bell Labs) helped in debugging problems with the laser printer used during the typesetting of this book. Thanks to them both. Thanks to Dick Hamilton (Unix System Laboratories) for making an early copy of SVR4.2 documentation available. Also, thanks to Gus Amegadzie (Programmed Logic Corporation), who helped test the SLIP software presented in Chapter 13. Special thanks to John Wait (Addison-Wesley) for his advice and encouragement during the last two years.

Finally, I want to thank my family, without whom this book wouldn't have been possible. They have supported me and helped to pull up the slack created by the amount of time I devoted to writing this book. My parents instilled in me the work ethic necessary to get it done (as well as provided their baby-sitting services), and my wife worked harder to give me the time to write it.

Contents

Part 1
Background Material

1
Introduction to Networks

This chapter discusses the motivation behind networking and some of the characteristics of various networks that we will encounter throughout the rest of this book.

1.1 BACKGROUND

A network can be loosely defined as *the hardware and software that enable two entities to communicate*. Humans can communicate over a telephone network. Central processing units in a multiprocessor can communicate over internal system buses. While these systems can be considered networks, this text is concerned only with communicating entities that are independent computer systems (often called *hosts*).

Computer networks are popular for many reasons. They provide a cost-effective alternative to large computing facilities. Rather than place all users and their files on a large mainframe, the users can be distributed among smaller, less expensive computers. By connecting these smaller machines with a network, the same level of sharing can be attained as if everyone were on one big machine.

Networks allow the separate computers to share files, devices such as printers and plotters, and other computing facilities, such as processing ability. Users can exchange electronic mail over a network. Computer systems can be administered remotely, over a network.

Because of networks, computers can be placed in the locations that best fit the organizations that use them, instead of the other way around. Geographical freedom allows organizations to structure themselves in ways that allow them to accomplish their goals more effectively.

Reliability is another benefit of using computer networks. While one computer is unavailable for processing, because it is either overloaded or out of service, processing can be redirected to another computer on the network. Services previously provided by a single computer can be distributed over several.