


Advanced Programming in the UNIX[®] Environment

Second Edition

W. Richard Stevens
Stephen A. Rago



S I T Library
Valachil, Mangalore

Accn No: 013877

Many of the designations used by manufacturers and sellers to distinguish their products are claimed as trademarks. Where those designations appear in this book and the publisher was aware of a trademark claim, the designations have been printed with initial capital letters or in all capitals.

The author and publisher have taken care in the preparation of this book, but make no expressed or implied warranty of any kind and assume no responsibility for errors or omissions. No liability is assumed for incidental or consequential damages in connection with or arising out of the use of the information or programs contained herein.

Copyright © 2005 by Pearson Education, Inc.

This edition is published by arrangement with Pearson Education, Inc. and Dorling Kindersley Publishing Inc.

This book is sold subject to the condition that it shall not, by way of trade or otherwise, be lent, resold, hired out, or otherwise circulated without the publisher's prior written consent in any form of binding or cover other than that in which it is published and without a similar condition including this condition being imposed on the subsequent purchaser and without limiting the rights under copyright reserved above, no part of this publication may be reproduced, stored in or introduced into a retrieval system, or transmitted in any form or by any means (electronic, mechanical, photocopying, recording or otherwise), without the prior written permission of both the copyright owner and the above-mentioned publisher of this book.

ISBN 978-81-317-0005-1

First Impression, 2006
Second Impression, 2007
Third Impression, 2008
Fourth Impression, 2008
Fifth Impression, 2009

Srinivas Institute of Technology

Acc. No.: 13877

Call No.:

This edition is manufactured in India and is authorized for sale only in India, Bangladesh, Bhutan, Pakistan, Nepal, Sri Lanka and the Maldives. Circulation of this edition outside of these territories is UNAUTHORIZED.

Published by Dorling Kindersley (India) Pvt. Ltd., licensees of Pearson Education in South Asia.

Head Office: 482, F.I.E., Patparganj, Delhi 110 092, India.

Registered Office: 14 Local Shopping Centre, Panchsheel Park, New Delhi 110 017, India.

Printed in India by Baba Barkha Nath Printers.

Contents

Foreword		xix
Preface		xxi
Preface to the First Edition		xxv
Chapter 1. UNIX System Overview		1
1.1	Introduction	1
1.2	UNIX Architecture	1
1.3	Logging In	2
1.4	Files and Directories	4
1.5	Input and Output	8
1.6	Programs and Processes	10
1.7	Error Handling	14
1.8	User Identification	16
1.9	Signals	18
1.10	Time Values	20
1.11	System Calls and Library Functions	21
1.12	Summary	23
Chapter 2. UNIX Standardization and Implementations		25
2.1	Introduction	25
2.2	UNIX Standardization	25
2.2.1	ISO C	25

2.2.2	IEEE POSIX	26	
2.2.3	The Single UNIX Specification	29	
2.2.4	FIPS	33	
2.3	UNIX System Implementations	33	
2.3.1	UNIX System V Release 4	33	
2.3.2	4.4BSD	34	
2.3.3	FreeBSD	35	
2.3.4	Linux	35	
2.3.5	Mac OS X	35	
2.3.6	Solaris	35	
2.3.7	Other UNIX Systems	36	
2.4	Relationship of Standards and Implementations	36	
2.5	Limits	36	
2.5.1	ISO C Limits	38	
2.5.2	POSIX Limits	38	
2.5.3	XSI Limits	40	
2.5.4	sysconf, pathconf, and fpathconf Functions	41	
2.5.5	Indeterminate Runtime Limits	48	
2.6	Options	52	
2.7	Feature Test Macros	55	
2.8	Primitive System Data Types	56	
2.9	Conflicts Between Standards	56	
2.10	Summary	58	
Chapter 3.	File I/O		59
3.1	Introduction	59	
3.2	File Descriptors	59	
3.3	open Function	60	
3.4	creat Function	62	
3.5	close Function	63	
3.6	lseek Function	63	
3.7	read Function	67	
3.8	write Function	68	
3.9	I/O Efficiency	68	
3.10	File Sharing	70	
3.11	Atomic Operations	74	
3.12	dup and dup2 Functions	76	
3.13	sync, fsync, and fdatasync Functions	77	
3.14	fcntl Function	78	
3.15	ioctl Function	83	
3.16	/dev/fd	84	
3.17	Summary	85	

Chapter 4.	Files and Directories	87
4.1	Introduction	87
4.2	stat, fstat, and lstat Functions	87
4.3	File Types	88
4.4	Set-User-ID and Set-Group-ID	91
4.5	File Access Permissions	92
4.6	Ownership of New Files and Directories	95
4.7	access Function	95
4.8	umask Function	97
4.9	chmod and fchmod Functions	99
4.10	Sticky Bit	101
4.11	chown, fchown, and lchown Functions	102
4.12	File Size	103
4.13	File Truncation	105
4.14	File Systems	105
4.15	link, unlink, remove, and rename Functions	108
4.16	Symbolic Links	112
4.17	symlink and readlink Functions	115
4.18	File Times	115
4.19	utime Function	116
4.20	mkdir and rmdir Functions	119
4.21	Reading Directories	120
4.22	chdir, fchdir, and getcwd Functions	125
4.23	Device Special Files	127
4.24	Summary of File Access Permission Bits	130
4.25	Summary	130
Chapter 5.	Standard I/O Library	133
5.1	Introduction	133
5.2	Streams and FILE Objects	133
5.3	Standard Input, Standard Output, and Standard Error	135
5.4	Buffering	135
5.5	Opening a Stream	138
5.6	Reading and Writing a Stream	140
5.7	Line-at-a-Time I/O	142
5.8	Standard I/O Efficiency	143
5.9	Binary I/O	145
5.10	Positioning a Stream	147
5.11	Formatted I/O	149
5.12	Implementation Details	153
5.13	Temporary Files	155

5.14	Alternatives to Standard I/O	159
5.15	Summary	159
Chapter 6.	System Data Files and Information	161
6.1	Introduction	161
6.2	Password File	161
6.3	Shadow Passwords	165
6.4	Group File	166
6.5	Supplementary Group IDs	167
6.6	Implementation Differences	169
6.7	Other Data Files	169
6.8	Login Accounting	170
6.9	System Identification	171
6.10	Time and Date Routines	173
6.11	Summary	177
Chapter 7.	Process Environment	179
7.1	Introduction	179
7.2	main Function	179
7.3	Process Termination	180
7.4	Command-Line Arguments	185
7.5	Environment List	185
7.6	Memory Layout of a C Program	186
7.7	Shared Libraries	188
7.8	Memory Allocation	189
7.9	Environment Variables	192
7.10	setjmp and longjmp Functions	195
7.11	getrlimit and setrlimit Functions	202
7.12	Summary	206
Chapter 8.	Process Control	209
8.1	Introduction	209
8.2	Process Identifiers	209
8.3	fork Function	211
8.4	vfork Function	216
8.5	exit Functions	218
8.6	wait and waitpid Functions	220
8.7	waitid Function	226
8.8	wait3 and wait4 Functions	227
8.9	Race Conditions	227
8.10	exec Functions	231

8.11	Changing User IDs and Group IDs	237
8.12	Interpreter Files	242
8.13	system Function	246
8.14	Process Accounting	250
8.15	User Identification	256
8.16	Process Times	257
8.17	Summary	259
Chapter 9.	Process Relationships	261
9.1	Introduction	261
9.2	Terminal Logins	261
9.3	Network Logins	266
9.4	Process Groups	269
9.5	Sessions	270
9.6	Controlling Terminal	272
9.7	tcgetpgrp, tcsetpgrp, and tcgetsid Functions	273
9.8	Job Control	274
9.9	Shell Execution of Programs	278
9.10	Orphaned Process Groups	282
9.11	FreeBSD Implementation	285
9.12	Summary	287
Chapter 10.	Signals	289
10.1	Introduction	289
10.2	Signal Concepts	289
10.3	signal Function	298
10.4	Unreliable Signals	301
10.5	Interrupted System Calls	303
10.6	Reentrant Functions	305
10.7	SIGCLD Semantics	308
10.8	Reliable-Signal Terminology and Semantics	310
10.9	kill and raise Functions	311
10.10	alarm and pause Functions	313
10.11	Signal Sets	318
10.12	sigprocmask Function	320
10.13	sigpending Function	322
10.14	sigaction Function	324
10.15	sigsetjmp and siglongjmp Functions	329
10.16	sigsuspend Function	333
10.17	abort Function	340
10.18	system Function	342

10.19	<code>sleep</code> Function	347	
10.20	Job-Control Signals	349	
10.21	Additional Features	352	
10.22	Summary	353	
Chapter 11.	Threads		355
11.1	Introduction	355	
11.2	Thread Concepts	355	
11.3	Thread Identification	356	
11.4	Thread Creation	357	
11.5	Thread Termination	360	
11.6	Thread Synchronization	368	
11.7	Summary	385	
Chapter 12.	Thread Control		387
12.1	Introduction	387	
12.2	Thread Limits	387	
12.3	Thread Attributes	388	
12.4	Synchronization Attributes	393	
12.5	Reentrancy	401	
12.6	Thread-Specific Data	406	
12.7	Cancel Options	410	
12.8	Threads and Signals	413	
12.9	Threads and <code>fork</code>	416	
12.10	Threads and I/O	420	
12.11	Summary	420	
Chapter 13.	Daemon Processes		423
13.1	Introduction	423	
13.2	Daemon Characteristics	423	
13.3	Coding Rules	425	
13.4	Error Logging	428	
13.5	Single-Instance Daemons	432	
13.6	Daemon Conventions	434	
13.7	Client-Server Model	439	
13.8	Summary	439	
Chapter 14.	Advanced I/O		441
14.1	Introduction	441	
14.2	Nonblocking I/O	441	
14.3	Record Locking	444	

14.4	STREAMS	460	
14.5	I/O Multiplexing	472	
14.5.1	select and pselect Functions	474	
14.5.2	poll Function	479	
14.6	Asynchronous I/O	481	
14.6.1	System V Asynchronous I/O	481	
14.6.2	BSD Asynchronous I/O	482	
14.7	readv and writev Functions	483	
14.8	readn and writen Functions	485	
14.9	Memory-Mapped I/O	487	
14.10	Summary	492	
Chapter 15.	Interprocess Communication		495
15.1	Introduction	495	
15.2	Pipes	496	
15.3	popen and pclose Functions	503	
15.4	Coprocesses	510	
15.5	FIFOs	514	
15.6	XSI IPC	518	
15.6.1	Identifiers and Keys	518	
15.6.2	Permission Structure	520	
15.6.3	Configuration Limits	521	
15.6.4	Advantages and Disadvantages	521	
15.7	Message Queues	522	
15.8	Semaphores	527	
15.9	Shared Memory	533	
15.10	Client-Server Properties	541	
15.11	Summary	543	
Chapter 16.	Network IPC: Sockets		545
16.1	Introduction	545	
16.2	Socket Descriptors	546	
16.3	Addressing	549	
16.3.1	Byte Ordering	549	
16.3.2	Address Formats	551	
16.3.3	Address Lookup	553	
16.3.4	Associating Addresses with Sockets	560	
16.4	Connection Establishment	561	
16.5	Data Transfer	565	
16.6	Socket Options	579	
16.7	Out-of-Band Data	581	

16.8	Nonblocking and Asynchronous I/O	582
16.9	Summary	583
Chapter 17.	Advanced IPC	585
17.1	Introduction	585
17.2	STREAMS-Based Pipes	585
17.2.1	Naming STREAMS Pipes	589
17.2.2	Unique Connections	590
17.3	UNIX Domain Sockets	594
17.3.1	Naming UNIX Domain Sockets	595
17.3.2	Unique Connections	597
17.4	Passing File Descriptors	601
17.4.1	Passing File Descriptors over STREAMS-Based Pipes	604
17.4.2	Passing File Descriptors over UNIX Domain Sockets	606
17.5	An Open Server, Version 1	615
17.6	An Open Server, Version 2	620
17.7	Summary	629
Chapter 18.	Terminal I/O	631
18.1	Introduction	631
18.2	Overview	631
18.3	Special Input Characters	638
18.4	Getting and Setting Terminal Attributes	643
18.5	Terminal Option Flags	643
18.6	stty Command	651
18.7	Baud Rate Functions	652
18.8	Line Control Functions	653
18.9	Terminal Identification	654
18.10	Canonical Mode	660
18.11	Noncanonical Mode	663
18.12	Terminal Window Size	670
18.13	termcap, terminfo, and curses	672
18.14	Summary	673
Chapter 19.	Pseudo Terminals	675
19.1	Introduction	675
19.2	Overview	675
19.3	Opening Pseudo-Terminal Devices	681
19.3.1	STREAMS-Based Pseudo Terminals	683
19.3.2	BSD-Based Pseudo Terminals	686
19.3.3	Linux-Based Pseudo Terminals	689

19.4	pty_fork Function	691	
19.5	pty Program	694	
19.6	Using the pty Program	698	
19.7	Advanced Features	705	
19.8	Summary	706	
Chapter 20.	A Database Library		709
20.1	Introduction	709	
20.2	History	709	
20.3	The Library	710	
20.4	Implementation Overview	712	
20.5	Centralized or Decentralized?	716	
20.6	Concurrency	718	
20.7	Building the Library	719	
20.8	Source Code	719	
20.9	Performance	747	
20.10	Summary	752	
Chapter 21.	Communicating with a Network Printer		753
21.1	Introduction	753	
21.2	The Internet Printing Protocol	753	
21.3	The Hypertext Transfer Protocol	756	
21.4	Printer Spooling	757	
21.5	Source Code	758	
21.6	Summary	805	
Appendix A.	Function Prototypes		807
Appendix B.	Miscellaneous Source Code		843
B.1	Our Header File	843	
B.2	Standard Error Routines	846	
Appendix C.	Solutions to Selected Exercises		853
Index			885

Foreword

At some point during nearly every interview I give, as well as in question periods after talks, I get asked some variant of the same question: “Did you expect Unix to last for so long?” And of course the answer is always the same: No, we didn’t quite anticipate what has happened. Even the observation that the system, in some form, has been around for well more than half the lifetime of the commercial computing industry is now dated.

The course of developments has been turbulent and complicated. Computer technology has changed greatly since the early 1970s, most notably in universal networking, ubiquitous graphics, and readily available personal computing, but the system has somehow managed to accommodate all of these phenomena. The commercial environment, although today dominated on the desktop by Microsoft and Intel, has in some ways moved from single-supplier to multiple sources and, in recent years, to increasing reliance on public standards and on freely available source.

Fortunately, Unix, considered as a phenomenon and not just a brand, has been able to move with and even lead this wave. AT&T in the 1970s and 1980s was protective of the actual Unix source code, but encouraged standardization efforts based on the system’s interfaces and languages. For example, the SVID—the System V Interface Definition—was published by AT&T, and it became the basis for the POSIX work and its follow-ons. As it happened, Unix was able to adapt rather gracefully to a networked environment and, perhaps less elegantly, but still adequately, to a graphical one. And as it also happened, the basic Unix kernel interface and many of its characteristic user-level tools were incorporated into the technological foundations of the open-source movement.

It is important that papers and writings about the Unix system were always encouraged, even while the software of the system itself was proprietary, for example Maurice Bach’s book, *The Design of the Unix Operating System*. In fact, I would claim that

a central reason for the system's longevity has been that it has attracted remarkably talented writers to explain its beauties and mysteries. Brian Kernighan is one of these; Rich Stevens is certainly another. The first edition of this book, along with his series of books about networking, are rightfully regarded as remarkably well-crafted works of exposition, and became hugely popular.

However, the first edition of this book was published before Linux and the several open-source renditions of the Unix interface that stemmed from the Berkeley CSRG became widespread, and also at a time when many people's networking consisted of a serial modem. Steve Rago has carefully updated this book to account for the technology changes, as well as developments in various ISO and IEEE standards since its first publication. Thus his examples are fresh, and freshly tested.

It's a most worthy second edition of a classic.

*Murray Hill, New Jersey
March 2005*

Dennis Ritchie

Preface

Introduction

Rich Stevens and I first met through an e-mail exchange when I reported a typographical error in his first book, *UNIX Network Programming*. He used to kid me about being the person to send him his first errata notice for the book. Until his death in 1999, we exchanged e-mail irregularly, usually when one of us had a question we thought the other might be able to answer. We met for dinner at USENIX conferences and when Rich was teaching in the area.

Rich Stevens was a friend who always conducted himself as a gentleman. When I wrote *UNIX System V Network Programming* in 1993, I intended it to be a System V version of Rich's *UNIX Network Programming*. As was his nature, Rich gladly reviewed chapters for me, and treated me not as a competitor, but as a colleague. We often talked about collaborating on a STREAMS version of his *TCP/IP Illustrated* book. Had events been different, we might have actually done it, but since Rich is no longer with us, revising *Advanced Programming in the UNIX Environment* is the closest I'll ever get to writing a book with him.

When the editors at Addison-Wesley told me that they wanted to update Rich's book, I thought that there wouldn't be too much to change. Even after 13 years, Rich's work still holds up well. But the UNIX industry is vastly different today from what it was when the book was first published.

- The System V variants are slowly being replaced by Linux. The major system vendors that ship their hardware with their own versions of the UNIX System have either made Linux ports available or announced support for Linux. Solaris is perhaps the last descendant of UNIX System V Release 4 with any appreciable market share.

- After 4.4BSD was released, the Computing Science Research Group (CSRG) from the University of California at Berkeley decided to put an end to its development of the UNIX operating system, but several different groups of volunteers still maintain publicly available versions.
- The introduction of Linux, supported by thousands of volunteers, has made it possible for anyone with a computer to run an operating system similar to the UNIX System, with freely available source code for the newest hardware devices. The success of Linux is something of a curiosity, given that several free BSD alternatives are readily available.
- Continuing its trend as an innovative company, Apple Computer abandoned its old Mac operating system and replaced it with one based on Mach and FreeBSD.

Thus, I've tried to update the information presented in this book to reflect these four platforms.

After Rich wrote *Advanced Programming in the UNIX Environment* in 1992, I got rid of most of my UNIX programmer's manuals. To this day, the two books I keep closest to my desk are a dictionary and a copy of *Advanced Programming in the UNIX Environment*. I hope you find this revision equally useful.

Changes from the First Edition

Rich's work holds up well. I've tried not to change his original vision for this book, but a lot has happened in 13 years. This is especially true with the standards that affect the UNIX programming interface.

Throughout the book, I've updated interfaces that have changed from the ongoing efforts in standards organizations. This is most noticeable in Chapter 2, since its primary topic is standards. The 2001 version of the POSIX.1 standard, which we use in this revision, is much more comprehensive than the 1990 version on which the first edition of this book was based. The 1990 ISO C standard was updated in 1999, and some changes affect the interfaces in the POSIX.1 standard.

A lot more interfaces are now covered by the POSIX.1 specification. The base specifications of the Single UNIX Specification (published by The Open Group, formerly X/Open) have been merged with POSIX.1. POSIX.1 now includes several 1003.1 standards and draft standards that were formerly published separately.

Accordingly, I've added chapters to cover some new topics. Threads and multithreaded programming are important concepts because they present a cleaner way for programmers to deal with concurrency and asynchrony.

The socket interface is now part of POSIX.1. It provides a single interface to interprocess communication (IPC), regardless of the location of the process, and is a natural extension of the IPC chapters.

I've omitted most of the real-time interfaces that appear in POSIX.1. These are best treated in a text devoted to real-time programming. One such book appears in the bibliography.

I've updated the case studies in the last chapters to cover more relevant real-world examples. For example, few systems these days are connected to a PostScript printer.

via a serial or parallel port. Most PostScript printers today are accessed via a network, so I've changed the case study that deals with PostScript printer communication to take this into account.

The chapter on modem communication is less relevant these days. So that the original material is not lost, however, it is available on the book's Web site in two formats: PostScript (<http://www.apuebook.com/lostchapter/modem.ps>) and PDF (<http://www.apuebook.com/lostchapter/modem.pdf>).

The source code for the examples shown in this book is also available at www.apuebook.com. Most of the examples have been run on four platforms:

1. FreeBSD 5.2.1, a derivative of the 4.4BSD release from the Computer Systems Research Group at the University of California at Berkeley, running on an Intel Pentium processor
2. Linux 2.4.22 (the Mandrake 9.2 distribution), a free UNIX-like operating system, running on Intel Pentium processors
3. Solaris 9, a derivative of System V Release 4 from Sun Microsystems, running on a 64-bit UltraSPARC III processor
4. Darwin 7.4.0, an operating environment based on FreeBSD and Mach, supported by Apple Mac OS X, version 10.3, on a PowerPC processor

Acknowledgments

Rich Stevens wrote the first edition of this book on his own, and it became an instant classic.

I couldn't have updated this book without the support of my family. They put up with piles of papers scattered about the house (well, more so than usual), my monopolizing most of the computers in the house, and lots of hours with my face buried behind a computer terminal. My wife, Jeanne, even helped out by installing Linux for me on one of the test machines.

The technical reviewers suggested many improvements and helped make sure that the content was accurate. Many thanks to David Bausum, David Boreham, Keith Bostic, Mark Ellis, Phil Howard, Andrew Josey, Mukesh Kacker, Brian Kernighan, Bengt Kleberg, Ben Kuperman, Eric Raymond, and Andy Rudoff.

I'd also like to thank Andy Rudoff for answering questions about Solaris and Dennis Ritchie for digging up old papers and answering history questions. Once again, the staff at Addison-Wesley was great to work with. Thanks to Tyrrell Albaugh, Mary Franz, John Fuller, Karen Gettman, Jessica Goldstein, Noreen Regina, and John Wait. My thanks to Evelyn Pyle for the fine job of copyediting.

As Rich did, I also welcome electronic mail from any readers with comments, suggestions, or bug fixes.

Warren, New Jersey
April 2005

Stephen A. Rago
sar@apuebook.com

Preface to the First Edition

Introduction

This book describes the programming interface to the Unix system—the system call interface and many of the functions provided in the standard C library. It is intended for anyone writing programs that run under Unix.

Like most operating systems, Unix provides numerous services to the programs that are running—open a file, read a file, start a new program, allocate a region of memory, get the current time-of-day, and so on. This has been termed the *system call interface*. Additionally, the standard C library provides numerous functions that are used by almost every C program (format a variable's value for output, compare two strings, etc.).

The system call interface and the library routines have traditionally been described in Sections 2 and 3 of the *Unix Programmer's Manual*. This book is not a duplication of these sections. Examples and rationale are missing from the *Unix Programmer's Manual*, and that's what this book provides.

Unix Standards

The proliferation of different versions of Unix during the 1980s has been tempered by the various international standards that were started during the late 1980s. These include the ANSI standard for the C programming language, the IEEE POSIX family (still being developed), and the X/Open portability guide.

This book also describes these standards. But instead of just describing the standards by themselves, we describe them in relation to popular implementations of the standards—System V Release 4 and the forthcoming 4.4BSD. This provides a real-world description, which is often lacking from the standard itself and from books that describe only the standard.

Organization of the Book

This book is divided into six parts:

1. An overview and introduction to basic Unix programming concepts and terminology (Chapter 1), with a discussion of the various Unix standardization efforts and different Unix implementations (Chapter 2).
2. I/O—unbuffered I/O (Chapter 3), properties of files and directories (Chapter 4), the standard I/O library (Chapter 5), and the standard system data files (Chapter 6).
3. Processes—the environment of a Unix process (Chapter 7), process control (Chapter 8), the relationships between different processes (Chapter 9), and signals (Chapter 10).
4. More I/O—terminal I/O (Chapter 11), advanced I/O (Chapter 12), and daemon processes (Chapter 13).
5. IPC—Interprocess communication (Chapters 14 and 15).
6. Examples—a database library (Chapter 16), communicating with a PostScript printer (Chapter 17), a modem dialing program (Chapter 18), and using pseudo terminals (Chapter 19).

A reading familiarity with C would be beneficial as would some experience using Unix. No prior programming experience with Unix is assumed. This text is intended for programmers familiar with Unix and programmers familiar with some other operating system who wish to learn the details of the services provided by most Unix systems.

Examples in the Text

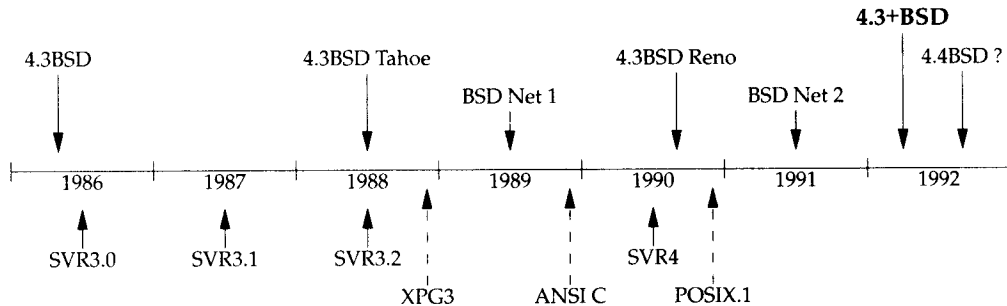
This book contains many examples—approximately 10,000 lines of source code. All the examples are in the C programming language. Furthermore, these examples are in ANSI C. You should have a copy of the *Unix Programmer's Manual* for your system handy while reading this book, since reference is made to it for some of the more esoteric and implementation-dependent features.

Almost every function and system call is demonstrated with a small, complete program. This lets us see the arguments and return values and is often easier to comprehend than the use of the function in a much larger program. But since some of the small programs are contrived examples, a few bigger examples are also included (Chapters 16, 17, 18, and 19). These larger examples demonstrate the programming techniques in larger, real-world examples.

All the examples have been included in the text directly from their source files. A machine-readable copy of all the examples is available via anonymous FTP from the Internet host `ftp.uu.net` in the file `published/books/stevens.advprog.tar.Z`. Obtaining the source code allows you to modify the programs from this text and experiment with them on your system.

Systems Used to Test the Examples

Unfortunately all operating systems are moving targets. Unix is no exception. The following diagram shows the recent evolution of the various versions of System V and 4.xBSD.



4.xBSD are the various systems from the Computer Systems Research Group at the University of California at Berkeley. This group also distributes the BSD Net 1 and BSD Net 2 releases—publicly available source code from the 4.xBSD systems. SVR x refers to System V Release x from AT&T. XPG3 is the X/Open Portability Guide, Issue 3, and ANSI C is the ANSI standard for the C programming language. POSIX.1 is the IEEE and ISO standard for the interface to a Unix-like system. We'll have more to say about these different standards and the various versions of Unix in Sections 2.2 and 2.3.

In this text we use the term 4.3+BSD to refer to the Unix system from Berkeley that is somewhere between the BSD Net 2 release and 4.4BSD.

At the time of this writing, 4.4BSD was not released, so the system could not be called 4.4BSD. Nevertheless a simple name was needed to refer to this system and 4.3+BSD is used throughout the text.

Most of the examples in this text have been run on four different versions of Unix:

1. Unix System V/386 Release 4.0 Version 2.0 ("vanilla SVR4") from U.H. Corp. (UHC), on an Intel 80386 processor.
2. 4.3+BSD at the Computer Systems Research Group, Computer Science Division, University of California at Berkeley, on a Hewlett Packard workstation.
3. BSD/386 (a derivative of the BSD Net 2 release) from Berkeley Software Design, Inc., on an Intel 80386 processor. This system is almost identical to what we call 4.3+BSD.
4. SunOS 4.1.1 and 4.1.2 (systems with a strong Berkeley heritage but many System V features) from Sun Microsystems, on a SPARCstation SLC.

Numerous timing tests are provided in the text and the systems used for the test are identified.

Acknowledgments

Once again I am indebted to my family for their love, support, and many lost weekends over the past year and a half. Writing a book is, in many ways, a family affair. Thank you Sally, Bill, Ellen, and David.

I am especially grateful to Brian Kernighan for his help in the book. His numerous thorough reviews of the entire manuscript and his gentle prodding for better prose hopefully show in the final result. Steve Rago was also a great resource, both in reviewing the entire manuscript and answering many questions about the details and history of System V. My thanks to the other technical reviewers used by Addison-Wesley, who provided valuable comments on various portions of the manuscript: Maury Bach, Mark Ellis, Jeff Gitlin, Peter Honeyman, John Linderman, Doug McIlroy, Evi Nemeth, Craig Partridge, Dave Presotto, Gary Wilson, and Gary Wright.

Keith Bostic and Kirk McKusick at the U.C. Berkeley CSRG provided an account that was used to test the examples on the latest BSD system. (Many thanks to Peter Salus too.) Sam Nataros and Joachim Sacksen at UHC provided the copy of SVR4 used to test the examples. Trent Hein helped obtain the alpha and beta copies of BSD/386.

Other friends have helped in many small, but significant ways over the past few years: Paul Lucchina, Joe Godsil, Jim Hogue, Ed Tankus, and Gary Wright. My editor at Addison-Wesley, John Wait, has been a great friend through it all. He never complained when the due date slipped and the page count kept increasing. A special thanks to the National Optical Astronomy Observatories (NOAO), especially Sidney Wolff, Richard Wolff, and Steve Grandi, for providing computer time.

Real Unix books are written using troff and this book follows that time-honored tradition. Camera-ready copy of the book was produced by the author using the groff package written by James Clark. Many thanks to James Clark for providing this excellent system and for his rapid response to bug fixes. Perhaps someday I will really understand troff footer traps.

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

Tucson, Arizona
April 1992

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