

Table of Contents

List of Programs	xix
List of Figures	xxvii
Preface	xxx
Part 1 Introduction	1
Chapter 1 Introduction	3
1.1 Introduction	3
1.2 Logging In	5
1.3 Shells	6
1.4 Files, Directories, and File Systems.....	7
1.5 Input and Output	12
1.6 Programs, Processes, and Threads.....	16
1.7 Error Handling.....	21
1.8 User Identification	22
1.9 Signals	24
1.10 UNIX Time Values.....	27
1.11 System Calls and Library Functions	28
1.12 Introduction to 64-Bit Programming	31
1.12.1 Writing 64-Bit Clean Programs.....	32
1.12.2 Compiling and Installing 64-Bit Programs.....	38
1.12.3 The Large File Compilation Environment	39
1.12.4 The Transitional Large File Compilation Environment	41
1.13 Summary.....	42

Chapter 2	A Brief History of Solaris	45
2.1	Introduction	45
2.2	The Early Days: SunOS	45
2.3	Beyond SunOS: Solaris	46
2.4	Standards.....	47
2.4.1	ANSI/ISO/IEC C	47
2.4.2	System V Interface Definition	48
2.4.3	IEEE POSIX	48
2.4.4	The Open Group's XPG4.....	50
2.4.5	The Single UNIX Specification	50
2.5	Solaris 2.5.....	52
2.5.1	Solaris 2.5.1.....	52
2.6	Solaris 2.6.....	53
2.7	Solaris 7.....	53
2.8	Solaris 8.....	53
2.9	Solaris 9.....	54
2.10	Solaris Standards Compliance.....	54
2.11	Compiling Standards Conforming Applications.....	55
2.12	Summary.....	55
Part 2	Fundamental Topics	57
Chapter 3	Utility Functions.....	59
3.1	Introduction	59
3.2	Manipulating Character Classes	60
3.2.1	Testing Character Class Membership.....	60
3.2.2	Changing Character Class Membership.....	63
3.2.3	Summary of Character Classes.....	64
3.3	Manipulating Character Strings.....	65
3.3.1	Finding the Length of a String	66
3.3.2	Comparing Strings.....	67
3.3.3	String Concatenation.....	71
3.3.4	Copying Strings	73
3.3.5	String Searching Functions	74
3.3.6	Duplicating Strings.....	80
3.3.7	Splitting a String into Tokens	81
3.3.8	Functions for Transforming Strings	84
3.3.9	Converting Strings to Numbers	87
3.3.10	Converting Numbers to Strings	91
3.4	Manipulating Byte Arrays.....	91
3.4.1	Comparing Byte Arrays.....	92
3.4.2	Copying Byte Arrays.....	92
3.4.3	Searching Byte Arrays.....	93
3.4.4	Initializing Byte Arrays	94
3.5	Dynamic Memory.....	94
3.5.1	Memory Alignment.....	94
3.5.2	Allocating Dynamic Memory.....	96
3.5.3	Freeing Dynamic Memory	99
3.6	Other Memory Management Packages	100

3.6.1	The malloc Library	100
3.6.2	The bsdmalloc Library.....	102
3.6.3	The mapmalloc Library.....	103
3.6.4	The watchmalloc Shared Object.....	103
3.6.5	Comparing the malloc Libraries	106
3.7	Temporary Files	108
3.7.1	Generating Temporary Filenames	108
3.7.2	Creating Temporary Files	109
3.8	Parsing Command Line Arguments	110
3.9	Error Reporting	117
3.10	Suspending a Process	119
3.11	Summary.....	120
Chapter 4	Basic File I/O	123
4.1	Introduction	123
4.2	File Descriptors.....	123
4.3	The open Function	124
4.4	The creat Function	126
4.5	The close and closefrom Functions.....	127
4.6	The lseek and llseek Functions	128
4.7	The tell Function	131
4.8	The read and pread Functions.....	131
4.9	The write and pwrite Functions	132
4.10	The readn and writen Functions	134
4.11	I/O Efficiency.....	135
4.12	File Sharing.....	136
4.13	Atomic Operations	139
4.14	The dup and dup2 Functions	140
4.15	The fcntl Function	142
4.16	The ioctl Function	153
4.17	The fdwalk Function.....	153
4.18	Direct I/O.....	155
4.19	The /dev/fd File System.....	156
4.20	Summary.....	158
Chapter 5	The Standard I/O Library	159
5.1	Introduction	159
5.2	File Streams, Data Types, and Constants.....	159
5.3	Standard Input, Standard Output, and Standard Error	160
5.4	Opening a File Stream	161
5.5	Closing a File Stream.....	163
5.6	Reading and Writing	163
5.6.1	Character Input Functions	164
5.6.2	Character Output Functions	165
5.6.3	Line Input Functions	166
5.6.4	Line Output Functions	167
5.6.5	Binary I/O.....	167
5.7	Stream Status	168
5.8	Formatted I/O.....	170
5.8.1	Formatted Output	170

	5.8.2	Formatted Input	172
	5.8.3	Format Conversion Specifications	173
	5.8.4	C Language Escape Sequences	183
	5.9	Positioning a Stream	184
	5.10	File Stream Locking	185
	5.10.1	Unlocked File Stream I/O	189
	5.11	Buffering	191
	5.12	Standard I/O Efficiency	194
	5.13	Summary	199
Chapter 6		Date and Time Operations	201
	6.1	Introduction	201
	6.2	The Complexities of Converting Time	201
	6.3	Getting the Current Time	203
	6.3.1	The <code>difftime</code> Function	205
	6.4	Setting the Current Time	205
	6.5	Getting the Current Time Zone	206
	6.6	Converting between UNIX Time and Calendar Time	207
	6.6.1	The <code>localtime</code> and <code>localtime_r</code> Functions	209
	6.6.2	The <code>gmtime</code> and <code>gmtime_r</code> Functions	210
	6.6.3	The <code>mktime</code> Function	210
	6.7	Formatted Date I/O	212
	6.7.1	Converting a Date to a Formatted String	213
	6.7.2	Converting a Formatted String to a Date	217
	6.8	Summary	222
Chapter 7		Users and Groups	223
	7.1	Introduction	223
	7.2	User Names	223
	7.3	User IDs	227
	7.4	Group IDs	231
	7.5	Group Membership	234
	7.6	The Password File	237
	7.7	The Shadow Password File	243
	7.8	Reading and Encrypting Passwords	250
	7.9	The Group File	256
	7.10	The <code>utmpx</code> and <code>wtmpx</code> Files	261
	7.11	The <code>utmp</code> and <code>wtmp</code> Files	268
	7.12	The <code>lastlog</code> File	269
	7.13	The <code>shells</code> File	270
	7.14	Summary	273
Chapter 8		System Information and Resource Limits	275
	8.1	Introduction	275
	8.2	System Information and Identification	276
	8.3	System Resource Limits	282
	8.4	Per-Process Resource Limits	297
	8.5	The Resource Control Facility	302
	8.6	Resource Control Examples	311

8.7	Resource Usage Information	319
8.8	Determining Resource Usage Using the <code>/proc</code> File System.....	325
8.9	Determining the System's Load Average.....	334
8.10	Summary.....	335
Chapter 9	Secure C Programming	337
9.1	Introduction	337
9.2	Buffer Overflows.....	338
9.3	The Program's Environment.....	339
9.4	Defensive Programming.....	340
9.5	The Principle of Least Privilege	340
9.6	Using <code>chroot</code> Jails.....	344
9.7	Tips For Writing Secure Programs.....	346
9.8	Summary.....	349
Part 3	Input/Output	351
Chapter 10	Files and Directories.....	353
10.1	Introduction	353
10.2	Pathname Components	353
10.3	The <code>stat</code> , <code>fstat</code> , and <code>lstat</code> Functions.....	355
10.4	File Types.....	357
10.5	Set-User-ID and Set-Group-ID.....	360
10.6	The Sticky Bit.....	361
10.7	File Access Permissions	362
10.8	The <code>access</code> Function.....	365
10.9	The <code>umask</code> Function.....	367
10.10	The <code>chmod</code> and <code>fchmod</code> Functions	368
10.11	The <code>chown</code> , <code>fchown</code> , and <code>lchown</code> Functions.....	371
10.12	File Size	371
10.13	File Truncation.....	373
10.14	File Systems	374
10.15	The <code>link</code> and <code>unlink</code> Functions.....	377
10.16	The <code>remove</code> and <code>rename</code> Functions	380
10.17	Symbolic Links.....	381
10.18	Resolving Paths that Might Contain Symbolic Links.....	385
10.19	The <code>symlink</code> and <code>readlink</code> Functions.....	387
10.20	File Times	387
10.21	Changing a File's Access and Modification Times.....	390
10.22	Creating and Removing Directories.....	392
10.23	Reading Directories	393
10.24	The <code>chdir</code> , <code>fchdir</code> , and <code>getcwd</code> Functions.....	403
10.25	The <code>chroot</code> and <code>fchroot</code> Functions	406
10.26	Special Files.....	406
10.27	The <code>sync</code> and <code>fsync</code> Functions.....	409
10.28	Putting It All Together.....	410
10.29	Summary.....	416

Chapter 11	Working with File Systems.....	417
11.1	Introduction.....	417
11.2	Disk Terminology.....	417
11.3	The Mounted File System Table.....	418
11.4	The <code>mntfs</code> File System <code>ioctl</code> Commands.....	425
11.5	File System Defaults.....	427
11.6	Mounting and Unmounting File Systems.....	432
11.7	Obtaining the Status of a File System.....	438
11.8	Reading File System Data Structures.....	443
11.9	Summary.....	459
Chapter 12	Terminal I/O	461
12.1	Introduction.....	461
12.2	Overview of Terminal I/O.....	461
12.3	Special Input Characters.....	469
12.4	Getting and Setting Terminal Attributes.....	474
12.5	Terminal Option Flags.....	475
12.6	Baud Rate Functions.....	484
12.7	Line Control Functions.....	486
12.8	Terminal Identification.....	488
12.9	Canonical Mode.....	493
12.10	Non-Canonical Mode.....	496
12.11	Terminal Window Size.....	502
12.12	Device-Independent Terminal Control.....	503
12.13	Summary.....	505
Chapter 13	Advanced I/O	507
13.1	Introduction.....	507
13.2	Nonblocking I/O.....	507
13.3	Record Locking.....	511
13.4	Record Locking Using <code>fcntl</code>	513
13.5	Record Locking Using <code>lockf</code>	517
13.6	Deadlock and Livelock.....	517
13.7	Lock Inheritance and Release.....	519
13.8	Mandatory Versus Advisory Locking.....	521
13.9	The STREAMS I/O Subsystem.....	524
13.10	STREAMS Messages.....	526
13.11	The <code>putmsg</code> and <code>putpmsg</code> Functions.....	528
13.12	The <code>getmsg</code> and <code>getpmsg</code> Functions.....	529
13.13	STREAMS <code>ioctl</code> Operations.....	533
13.14	STREAMS I/O Using <code>read</code> and <code>write</code>	534
13.15	I/O Multiplexing.....	537
13.16	The <code>select</code> Function.....	539
13.17	The <code>poll</code> Function.....	543
13.18	The <code>/dev/poll</code> Device Driver.....	545
13.19	Asynchronous I/O.....	549
13.20	Asynchronous I/O With STREAMS Device Files.....	550
13.21	Asynchronous I/O With Other Files.....	551
13.22	The <code>readv</code> and <code>writv</code> Functions.....	553
13.23	The <code>sendfile</code> and <code>sendfilev</code> Functions.....	556

13.24	Memory Mapped I/O	562
13.25	The <code>mmap</code> and <code>munmap</code> Functions.....	562
13.26	The <code>mprotect</code> Function.....	570
13.27	The <code>madvise</code> Function.....	570
13.28	The <code>msync</code> Function.....	572
13.29	Locking Pages in Memory.....	573
13.30	The <code>mmapcntl</code> Function.....	577
13.31	Summary of Memory Mapped I/O.....	579
13.32	Access Control Lists	579
13.33	The <code>acl</code> and <code>facl</code> Functions.....	581
13.34	The <code>aclfromtext</code> and <code>acltotext</code> Functions.....	582
13.35	The <code>aclcheck</code> Function.....	584
13.36	The <code>aclfrommode</code> and <code>acltomode</code> Functions.....	588
13.37	The <code>aclsort</code> Function.....	589
13.38	Extended File Attributes.....	590
13.39	The <code>openat</code> and <code>attropen</code> Functions.....	591
13.40	The <code>fstatat</code> Function.....	594
13.41	The <code>unlinkat</code> Function.....	595
13.42	The <code>renameat</code> Function.....	595
13.43	The <code>fchownat</code> Function.....	596
13.44	The <code>futimesat</code> Function.....	596
13.45	Changing Extended Attribute File Permissions.....	597
13.46	Summary.....	599

Part 4 Processes and Process Control 601

Chapter 14	The Environment of a UNIX Process.....	603
14.1	Introduction	603
14.2	Process Start-Up	603
14.3	Process Termination	604
14.4	Command Line Arguments	608
14.5	Environment Variables.....	611
14.6	The Memory Layout of a C Program.....	613
14.7	Shared Objects.....	616
14.8	Memory Allocation.....	618
14.9	The <code>setjmp</code> and <code>longjmp</code> Functions.....	619
14.10	Resource Limits.....	626
14.11	Summary.....	626
Chapter 15	Process Control	629
15.1	Introduction.....	629
15.2	Process Identifiers	629
15.3	The <code>fork</code> and <code>fork1</code> Functions.....	631
15.4	The <code>vfork</code> Function.....	637
15.5	The <code>exit</code> and <code>_exit</code> Functions.....	639
15.6	The <code>wait</code> Function.....	641
15.7	The <code>waitpid</code> Function.....	643
15.8	The <code>wait3</code> and <code>wait4</code> Functions.....	646
15.9	The <code>waitid</code> Function.....	647

15.10	Race Conditions.....	648
15.11	The <code>exec</code> Functions.....	652
15.12	Interpreter Files.....	662
15.13	The <code>system</code> Function.....	666
15.14	Process Accounting.....	670
15.15	Summary.....	676
Chapter 16	Process Relationships.....	677
16.1	Introduction.....	677
16.2	Terminal Logins.....	677
16.3	Network Logins.....	679
16.4	Process Groups.....	682
16.5	Sessions.....	684
16.6	Controlling Terminal.....	686
16.7	The <code>tcgetpgrp</code> and <code>tcsetpgrp</code> Functions.....	688
16.8	The <code>tcgetsid</code> Function.....	688
16.9	Job Control.....	689
16.10	Shell Execution of Programs.....	693
16.11	Orphaned Process Groups.....	697
16.12	Summary.....	700
Chapter 17	Signals.....	703
17.1	Introduction.....	703
17.2	Signal Concepts.....	703
17.3	The <code>signal</code> Function.....	713
17.4	Unreliable Signals.....	716
17.5	Reliable Signals.....	718
17.6	The <code>sigset</code> Function.....	719
17.7	The <code>pause</code> Function.....	720
17.8	The <code>sighold</code> , <code>sigrelse</code> , <code>sigignore</code> , and <code>sigpause</code> Functions.....	721
17.9	Interrupted System Calls.....	723
17.10	Reentrant Functions.....	725
17.11	Comparing the <code>SIGCHLD</code> and <code>SIGCLD</code> Signals.....	728
17.12	The <code>kill</code> , <code>killpg</code> , <code>raise</code> , <code>sigsend</code> , and <code>sigsendset</code> Functions.....	729
17.13	The <code>alarm</code> Function.....	734
17.14	Interval Timers.....	742
17.15	POSIX Signals.....	744
17.16	Signal Sets.....	746
17.17	The <code>sigprocmask</code> Function.....	747
17.18	The <code>sigpending</code> Function.....	748
17.19	The <code>sigaction</code> Function.....	752
17.20	The <code>sigfpe</code> Function.....	761
17.21	The <code>sigsetjmp</code> and <code>siglongjmp</code> Functions.....	762
17.22	The <code>sigsuspend</code> Function.....	768
17.23	The <code>sigwait</code> Function.....	773
17.24	The <code>abort</code> Function.....	775
17.25	The <code>system</code> Function Revisited.....	776
17.26	The <code>sleep</code> Function Revisited.....	784
17.27	Job Control Signals.....	787

17.28	Software Signals.....	790
17.29	Alternate Signal Stacks.....	792
17.30	System Signal Messages.....	797
17.31	The <code>sig2str</code> and <code>str2sig</code> Functions.....	798
17.32	Summary.....	802
Chapter 18	Daemon Processes.....	805
18.1	Introduction.....	805
18.2	Characteristics of Daemons.....	806
18.3	Error Logging.....	807
18.4	The STREAMS log Driver.....	807
18.5	The <code>syslog</code> Facility.....	812
18.6	Becoming a Daemon.....	817
18.7	Starting Only One Copy of a Daemon.....	821
18.8	Summary.....	825
Part 5	Interprocess Communication	827
Chapter 19	Interprocess Communication Using Pipes and FIFOs.....	829
19.1	Introduction.....	829
19.2	Pipes.....	829
19.3	The <code>popen</code> and <code>pclose</code> Functions.....	839
19.4	Coprocesses.....	848
19.5	FIFOs.....	854
19.6	Iterative Versus Concurrent Servers.....	863
19.7	Summary.....	865
Chapter 20	The System V Interprocess Communication Facility.....	867
20.1	Introduction.....	867
20.2	System V IPC Concepts.....	867
20.3	System V Message Queues.....	873
20.4	System V Semaphores Sets.....	890
20.5	System V Shared Memory.....	908
20.6	Performance Comparisons.....	919
20.7	Summary.....	924
Chapter 21	Advanced Interprocess Communication.....	927
21.1	Introduction.....	927
21.2	Passing File Descriptors.....	927
21.3	An Open Server, Version 1.....	931
21.4	Client-Server Connection Functions.....	938
21.5	An Open Server, Version 2.....	943
21.6	Summary.....	948
Chapter 22	Doors.....	951
22.1	Introduction.....	951
22.2	Basic Door Functions.....	953
22.3	Door Information Functions.....	968

22.4	Advanced Door Facilities	973
22.5	Premature Termination of a Door Client or Server.....	985
22.6	Summary.....	992

Part 6 Pseudo Terminals 995

Chapter 23	Pseudo Terminals	997
23.1	Introduction	997
23.2	Overview	997
23.3	Opening a Pseudo Terminal Device.....	1004
23.4	The <code>pty_fork</code> Function.....	1009
23.5	The <code>pty</code> Program.....	1011
23.6	Using the <code>pty</code> Program.....	1016
23.7	Advanced Features.....	1025
23.8	Summary.....	1033
Appendix A	An Internationalization and Localization Primer.....	1035
A.1	Introduction	1035
A.2	Locales.....	1036
A.3	The <code>setlocale</code> Function.....	1037
A.4	Message Catalogues	1038
A.5	Creating a Message Catalogue	1039
A.6	The <code>bindtextdomain</code> Function.....	1041
A.7	The <code>gettext</code> , <code>dgettext</code> , and <code>dcgettext</code> Functions	1042
A.8	The <code>textdomain</code> Function	1042
A.9	The <code>strcoll</code> and <code>strxfrm</code> Functions.....	1044
A.10	Checklist for Writing Internationalized Programs	1045
A.11	Summary.....	1046
Appendix B	The BSD Source Compatibility Package.....	1047
B.1	Introduction	1047
B.2	Functions That Have Been Removed from the SCP.....	1049
B.2.1	Obsolescent SCP-Originated Functions.....	1049
B.2.2	SCP-Originated Functions That Are Not Obsolescent	1053
B.3	Summary.....	1055
Appendix C	Function Summary	1057
C.1	Introduction	1057
C.2	Function Prototypes	1057
C.3	Function Availability	1104
Appendix D	Miscellaneous Source Code	1117
D.1	Our Header File, <code>ssp.h</code>	1117
D.2	Standard Error Functions.....	1118
D.3	File Status Flags Functions.....	1121
D.4	Section Locking Function.....	1122
D.5	Our <code>readn</code> and <code>writen</code> Functions.....	1122
D.6	Termination Status Function	1124
D.7	Our Version of <code>snprintf</code>	1124

Appendix E Solutions to Selected Exercises.....	1127
Bibliography	1153
Index.....	1159

List of Programs

Program 1.1	List all the files in a directory.....	9
Program 1.2	Copy standard input to standard output using unbuffered I/O functions	14
Program 1.3	Copy standard input to standard output using standard I/O functions	15
Program 1.4	Print our process ID and parent process ID.....	18
Program 1.5	A very simple shell program.....	19
Program 1.6	Print our user and group IDs.....	24
Program 1.7	Our shell, modified to catch SIGINT	26
Program 1.8	Calculate the average system call latency	29
Program 1.9	Sign extension problems in 64-bit code	38
Program 1.10	Large file demonstration	41
Program 3.1	Showing the character class of a character	65
Program 3.2	Using <code>strlen</code> , <code>strspn</code> , and <code>strcspn</code>	67
Program 3.3	Bubble sort using <code>strcmp</code>	69
Program 3.4	Safe string concatenation	72
Program 3.5	Safe string copying.....	75
Program 3.6	Our <code>highlight</code> function	76
Program 3.7	Searching for the first occurrence of a character	76
Program 3.8	Find the first of several characters	79
Program 3.9	Searching for a substring	80
Program 3.10	Splitting lines into tokens	82
Program 3.11	Another method of splitting a line into tokens	84
Program 3.12	String encryption using <code>rot13</code>	86
Program 3.13	Converting numbers using <code>strtol</code>	89
Program 3.14	Comparing the different versions of <code>malloc</code>	107
Program 3.15	Using <code>getopt</code> and <code>getsubopt</code>	114
Program 3.16	Using an assertion to check for a NULL pointer	119
Program 4.1	Creating a sparse file	130
Program 4.2	Copy a file using <code>read</code> and <code>write</code>	133

Program 4.3	Print file status flags.....	148
Program 4.4	Print out the file access mode and status flags.....	149
Program 4.5	Print the extended and regular file descriptor flags.....	150
Program 4.6	A function to set file flags.....	151
Program 4.7	Copy a file using synchronous writes.....	152
Program 4.8	Implementing the <code>closefrom</code> function.....	154
Program 4.9	Investigating how direct I/O affects sequential file I/O.....	157
Program 5.1	Simple conversion specifications.....	178
Program 5.2	Using the alternative conversion introducer.....	178
Program 5.3	Adding precision modifiers.....	179
Program 5.4	Adding conversion specification flags.....	181
Program 5.5	Adding size modifiers.....	183
Program 5.6	Two threads writing a record without file locking.....	188
Program 5.7	Two threads writing a record with file locking.....	190
Program 5.8	Copy standard input to standard output using <code>fgetc</code> and <code>fputc</code>	195
Program 5.9	Copy standard input to standard output using <code>fgets</code> and <code>fputs</code>	196
Program 6.1	Determining a leap year.....	202
Program 6.2	Displaying information about the local time zone.....	208
Program 6.3	Printing login session length using <code>gmtime</code>	211
Program 6.4	Using <code>mktime</code> to determine what day of the week a given date is.....	212
Program 6.5	Using <code>strftime</code> to convert the time to various formats.....	218
Program 7.1	Using <code>getlogin</code>	225
Program 7.2	Using <code>cuserid</code>	227
Program 7.3	Getting and setting real and effective user IDs.....	229
Program 7.4	Getting and setting the current group set.....	236
Program 7.5	Searching by user name.....	239
Program 7.6	List all the users.....	241
Program 7.7	Searching the shadow password file by user name.....	246
Program 7.8	Listing all the passwords.....	248
Program 7.9	Traditional style password encryption.....	252
Program 7.10	New style password encryption.....	254
Program 7.11	Comparing passwords.....	255
Program 7.12	Searching by group name.....	258
Program 7.13	List all the groups.....	260
Program 7.14	Listing the logged-in users.....	265
Program 7.15	Printing users' last login times.....	271
Program 7.16	Listing the legal shells.....	272
Program 8.1	System information provided <code>uname</code> and <code>sysinfo</code>	281
Program 8.2	Displaying CPU and memory information.....	292
Program 8.3	Displaying some run time file limits.....	295
Program 8.4	Listing the supported user page sizes.....	297
Program 8.5	Print the current soft and hard resource limits.....	301
Program 8.6	Our implementation of the <code>print_rctl</code> function.....	312
Program 8.7	Our implementation of the <code>print_rctl</code> function.....	313
Program 8.8	Printing all the resource controls.....	315
Program 8.9	Modifying a resource control.....	317
Program 8.10	Time how long each command line argument takes to run.....	321
Program 8.11	Our implementation of <code>getprusage</code>	329
Program 8.12	Implementing <code>getprusage</code> using <code>ioctl</code>	332
Program 8.13	Displaying the system's load averages.....	335

Program 9.1	Using privilege bracketing	342
Program 9.2	Breaking out of a <code>chroot</code> jail	345
Program 10.1	Splitting a path into its component parts	355
Program 10.2	Print out the file type for each command line argument	359
Program 10.3	Demonstrating the <code>access</code> function	366
Program 10.4	Using the <code>umask</code> function	368
Program 10.5	Using the <code>chmod</code> function	370
Program 10.6	Unlinking a file	379
Program 10.7	Recursively list all the files in a directory	384
Program 10.8	Resolving symbolic links using <code>realpath</code> and <code>resolvepath</code>	386
Program 10.9	Display a file's access, modification, and inode modification times	390
Program 10.10	Changing a file's access and modification times	392
Program 10.11	Count the number of each type of file	397
Program 10.12	Recursively count the number of each type of file	402
Program 10.13	Change the current working directory	404
Program 10.14	Print the current working directory	405
Program 10.15	Print the <code>st_dev</code> and <code>st_rdev</code> fields of a file	409
Program 10.16	Print the entire contents of the <code>stat</code> structure	412
Program 11.1	Displaying information about mounted file systems	422
Program 11.2	Listing file systems mounted with a given option	424
Program 11.3	Setting a tag on a mounted file system	427
Program 11.4	Displaying information from <code>/etc/vfstab</code>	430
Program 11.5	Mounting a UFS file system	436
Program 11.6	Unmounting a file system	438
Program 11.7	Printing file system statistics	441
Program 11.8	Obtaining file system information using <code>ustat</code>	443
Program 11.9	Print the disk and inode use for each user	455
Program 12.1	Changing the INTR and KILL special input characters	474
Program 12.2	Using <code>tcgetattr</code> and <code>tcsetattr</code> to change the character size	476
Program 12.3	Changing the input baud rate	486
Program 12.4	Our implementation of <code>isatty</code>	489
Program 12.5	Our implementation of <code>ttynam_r</code>	491
Program 12.6	A function to read a user's password	495
Program 12.7	Set terminal to <code>cbreak</code> or <code>raw</code> mode	499
Program 12.8	Testing our <code>raw</code> and <code>cbreak</code> modes	501
Program 12.9	Print window size changes	504
Program 13.1	Using nonblocking I/O	510
Program 13.2	Our function to acquire or release a lock	515
Program 13.3	Our function to test for a lock	516
Program 13.4	Deadlock detection when file locking	520
Program 13.5	Copying a file that has a read lock on it	523
Program 13.6	Using <code>isastream</code>	526
Program 13.7	Copy a file using <code>getmsg</code> and <code>write</code>	532
Program 13.8	Listing the modules on a STREAMS device using <code>ioctl</code>	535
Program 13.9	Using <code>fd_sets</code>	541
Program 13.10	Using <code>poll</code> and nonblocking I/O to copy a file	546
Program 13.11	Using <code>/dev/poll</code> and nonblocking I/O to copy a file	548
Program 13.12	Using <code>writtev</code> to write multiple buffers	555
Program 13.13	Copying a file using <code>sendfile</code>	557
Program 13.14	Writing a buffer using <code>sendfile</code>	559

Program 13.15	Copying a file and two buffers using <code>sendfilev</code>	561
Program 13.16	Copy a file using <code>mmap</code> and <code>memcpy</code>	566
Program 13.17	Sharing memory between related processes using <code>/dev/zero</code>	569
Program 13.18	Locking a file in memory	576
Program 13.19	Printing a file's ACL	585
Program 13.20	Setting a file's ACL	587
Program 13.21	Listing a file's extended attributes	593
Program 13.22	Changing permissions on an extended attribute file	598
Program 14.1	Example of how to use exit handlers	608
Program 14.2	Echo command line arguments to standard output	609
Program 14.3	Determining our executable's pathname	610
Program 14.4	Getting and setting environment variables	613
Program 14.5	The ubiquitous "hello world" program	617
Program 14.6	A program using nested functions	620
Program 14.7	Our example changed to use <code>setjmp</code> and <code>longjmp</code>	622
Program 14.8	The effect of <code>longjmp</code> on automatic, register, and volatile variables	624
Program 14.9	Incorrect usage of an automatic variable	626
Program 15.1	An example of the <code>fork</code> function	634
Program 15.2	An example of the <code>vfork</code> function	638
Program 15.3	Describe the termination status of a process	642
Program 15.4	Demonstrating different termination statuses	643
Program 15.5	Avoiding zombies by calling <code>fork</code> twice	645
Program 15.6	Program with a race condition	650
Program 15.7	Reimplementation of Program 15.6 avoiding race condition	651
Program 15.8	Demonstrating the <code>exec</code> functions	658
Program 15.9	Echo all the command line arguments and environment variables	659
Program 15.10	A program that <code>execs</code> an interpreter file	663
Program 15.11	An <code>awk</code> script as an interpreter file	664
Program 15.12	Our implementation of the <code>system</code> function (without signal handling)	668
Program 15.13	Testing our implementation of the <code>system</code> function	669
Program 15.14	Program to generate some process accounting entries	673
Program 15.15	Print selected fields from process accounting file	675
Program 16.1	Creating an orphaned process group	698
Program 17.1	Catching the signals <code>SIGUSR1</code> and <code>SIGUSR2</code> by using <code>signal</code>	715
Program 17.2	Catching the signals <code>SIGUSR1</code> and <code>SIGUSR2</code> by using <code>sigset</code>	721
Program 17.3	Calling non-reentrant functions from a signal handler	727
Program 17.4	A <code>SIGCLD</code> signal handler that can be problematic	730
Program 17.5	Our first attempt at implementing <code>sleep</code>	736
Program 17.6	Avoiding the race condition in our first version of <code>sleep</code>	737
Program 17.7	Calling <code>ssp_sleep</code> from a program that catches other signals	738
Program 17.8	Collecting user input with a timeout	739
Program 17.9	Collecting user input with a timeout, helped by <code>setjmp</code> and <code>longjmp</code>	741
Program 17.10	Collecting user input with an interval timer timeout	745
Program 17.11	Printing the current signal mask	749
Program 17.12	Example of signal sets, <code>sigprocmask</code> , and <code>sigpending</code>	750
Program 17.13	Our implementation of <code>signal</code> using <code>sigaction</code>	759
Program 17.14	Printing signal information in a signal handler	760
Program 17.15	Trapping invalid operation floating-point exceptions	763
Program 17.16	How <code>sigsetjmp</code> and <code>siglongjmp</code> interact with signal masks	765
Program 17.17	Protecting a critical region of code from a signal	770

Program 17.18	Waiting for a global variable to be set	772
Program 17.19	Functions that enable a child and parent to synchronize using signals	775
Program 17.20	Our implementation of the <code>abort</code> function	777
Program 17.21	Using <code>ssp_system</code> to invoke <code>ex</code>	778
Program 17.22	An implementation of <code>system</code> that handles signals correctly	780
Program 17.23	Execute the command line argument using <code>system</code>	783
Program 17.24	Our final implementation of <code>sleep</code>	786
Program 17.25	Handling <code>SIGTSTP</code>	789
Program 17.26	Using software signals	791
Program 17.27	Using an alternate stack for a signal handler	794
Program 17.28	Printing signal information using <code>psignal</code> and <code>psiginfo</code>	799
Program 17.29	Our implementation of the <code>kill</code> command	801
Program 18.1	Generating a log message	810
Program 18.2	Using the <code>syslog</code> facility	816
Program 18.3	Our <code>daemon_init</code> function.....	818
Program 18.4	Testing our <code>daemon_init</code> function	821
Program 18.5	Our <code>one_copy</code> function	823
Program 18.6	Testing our <code>one_copy</code> function.....	824
Program 19.1	Sending data from parent to child over a pipe	833
Program 19.2	Sending email through a pipe	835
Program 19.3	Functions that enable a child and parent to synchronize using pipes.....	838
Program 19.4	Sending email through a pipe using <code>popen</code>	841
Program 19.5	Our implementation of <code>popen</code>	842
Program 19.6	Our implementation of <code>pclose</code>	844
Program 19.7	Support functions for <code>popen</code> and <code>pclose</code>	846
Program 19.8	Our modified version of <code>rot13</code>	848
Program 19.9	Invoking <code>rot13</code> using our version of <code>popen</code> and <code>pclose</code>	849
Program 19.10	A simple coprocess to calculate the square root of a number	850
Program 19.11	Program to drive the <code>sqrt</code> coprocess	851
Program 19.12	Another version of <code>sqrt</code> , using standard I/O	853
Program 19.13	FIFO server that can handle multiple clients.....	859
Program 19.14	FIFO client that works with the server in Program 19.13	862
Program 20.1	Obtain a System V IPC key based on a pathname	870
Program 20.2	Create a System V message queue	879
Program 20.3	Add a message to a System V message queue	880
Program 20.4	Remove a message from a System V message queue	882
Program 20.5	Remove a System V message queue.....	883
Program 20.6	List all messages currently in System V message queues	887
Program 20.7	Create a System V semaphore set.....	897
Program 20.8	Remove a System V semaphore set.....	898
Program 20.9	Set the values of semaphores in a semaphore set	899
Program 20.10	Get the values of semaphores in a semaphore set.....	900
Program 20.11	Perform operations on a System V semaphore set	902
Program 20.12	List all the System V semaphore sets.....	905
Program 20.13	Create a System V shared memory segment	913
Program 20.14	Write to a System V shared memory segment	914
Program 20.15	Read from a System V shared memory segment.....	915
Program 20.16	Remove a System V shared memory segment	916
Program 20.17	List all the System V shared memory segments.....	918
Program 20.18	Copy 100 MB using a System V message queue.....	921

Program 20.19	Copy 100 MB using a pipe.....	922
Program 20.20	Access a shared resource using a lock file.....	923
Program 20.21	Access a shared resource using a System V semaphore.....	925
Program 21.1	The <code>send_fd</code> and <code>send_err</code> functions.....	930
Program 21.2	The <code>recv_fd</code> function.....	931
Program 21.3	The client's main function.....	933
Program 21.4	The <code>cs_open</code> function.....	934
Program 21.5	The first server's main function.....	935
Program 21.6	The <code>do_request</code> function.....	936
Program 21.7	The <code>buf2args</code> and <code>parse_args</code> functions.....	937
Program 21.8	The <code>srv_listen</code> function.....	941
Program 21.9	The <code>srv_accept</code> function.....	943
Program 21.10	The <code>cli_connect</code> function.....	943
Program 21.11	The <code>cs_open</code> function.....	944
Program 21.12	Preamble for version 2 of our open server.....	945
Program 21.13	The second server's main function.....	946
Program 21.14	The <code>loop</code> function.....	947
Program 21.15	The <code>add_entry</code> and <code>del_entry</code> functions.....	949
Program 22.1	A simple door client.....	958
Program 22.2	A simple door server.....	959
Program 22.3	Door client modified to print address and size of result.....	961
Program 22.4	Door server that handles an unreferenced invocation.....	963
Program 22.5	The <code>main</code> function of our multiple unreferenced invocations server.....	966
Program 22.6	The server procedure of our multiple unreferenced invocations server.....	967
Program 22.7	Server procedure that prints the client's credentials.....	969
Program 22.8	Print information about a door.....	972
Program 22.9	Door client for descriptor passing example.....	975
Program 22.10	Door server procedure for descriptor passing example.....	976
Program 22.11	The <code>main</code> function of our server that manages its own threads.....	980
Program 22.12	The server creation function of our server that manages its own threads.....	981
Program 22.13	The thread start function of our server that manages its own threads.....	983
Program 22.14	The server procedure of our server that manages its own threads.....	984
Program 22.15	A server procedure that terminates itself.....	986
Program 22.16	Server procedure for idempotent server example.....	987
Program 22.17	Client that calls <code>door_call</code> again when <code>SIGCHLD</code> is received.....	988
Program 22.18	A client that terminates during a <code>door_call</code>	990
Program 22.19	A server procedure that detects premature client termination.....	991
Program 23.1	Function to open a pseudo terminal master device.....	1007
Program 23.2	Function to open a pseudo terminal slave device.....	1008
Program 23.3	Our <code>pty_fork</code> function.....	1010
Program 23.4	Our <code>pty</code> program's main function.....	1012
Program 23.5	The <code>pty</code> program's <code>loop</code> function.....	1015
Program 23.6	Implementing the <code>script</code> program using <code>pty</code>	1019
Program 23.7	The <code>run_driver</code> function for the <code>pty</code> program.....	1024
Program 23.8	Script to drive the <code>passwd</code> program.....	1025
Program 23.9	The <code>pckt</code> program's <code>loop</code> function.....	1027
Program 23.10	Program to test packet mode pseudo terminals.....	1031
Program A.1	An internationalized version of our greeting program.....	1043
Program D.1	Our header file, <code>ssp.h</code>	1117
Program D.2	Our standard error functions.....	1119

Program D.3	Our file status flag functions.....	1121
Program D.4	Our function to acquire or release a lock on a file section	1122
Program D.5	Our <code>readn</code> and <code>writen</code> functions.....	1123
Program D.6	Our function to print the termination status of a process	1124
Program D.7	Our version of <code>snprintf</code>	1124
Program E.1	Determine a file's size by using <code>lseek</code>	1128
Program E.2	Our function to convert an arbitrarily based number to a string	1129
Program E.3	Get a <code>lastlog</code> entry for a given user ID.....	1132
Program E.4	Writing to a file that is underneath a mount point	1137
Program E.5	Accessing a file after its file system has been unmounted	1138
Program E.6	Measuring the capacity of a pipe by using <code>poll</code>	1140
Program E.7	Incorrect use of <code>vfork</code>	1142
Program E.8	Create a zombie and look at it using <code>ps</code>	1143
Program E.9	Print a terminal's foreground process group ID.....	1144
Program E.10	Create a new session	1145
Program E.11	Print the number of the highest open file descriptor	1147
Program E.12	Portably opening a FIFO for reading and writing	1149
Program E.13	Passing a door descriptor in a cookie to avoid making it global.....	1151
Program E.14	Adding start and stop timestamps to our version of <code>script</code>	1152

List of Figures

Figure 1.1	Summary of the different file system types	8
Figure 1.2	System call latency on four architectures.....	30
Figure 1.3	Relationship of applications, library functions, and system calls	30
Figure 1.4	C data type sizes in bits for the ILP32 and LP64 data type models.....	33
Figure 1.5	The effect of the various compilation environments.....	42
Figure 2.1	Standards compliance of Solaris	54
Figure 2.2	Commands and flags required to build standards conforming applications.....	55
Figure 3.1	Alignment of various word sizes.....	95
Figure 3.2	Memory layout of our structure	95
Figure 3.3	Memory layout of our reordered structure.....	96
Figure 3.4	Red zones	105
Figure 3.5	Results from running Program 3.14	106
Figure 4.1	Availability of <code>open</code> flags	127
Figure 4.2	The effect of buffer size on file I/O	135
Figure 4.3	Kernel structures used for open files.....	137
Figure 4.4	Kernel structures for two processes opening the same file.....	138
Figure 4.5	Kernel structures after <code>dup</code>	142
Figure 4.6	Categories of <code>fcntl cmd</code> arguments.....	143
Figure 4.7	File access flags	144
Figure 4.8	Availability of <code>fcntl cmds</code>	147
Figure 4.9	The effect of <code>O_DSYNC</code> and <code>O_SYNC</code> on file copy times	152
Figure 4.10	Summary of how direct I/O affects sequential file I/O	156
Figure 5.1	The effect of the values of <code>mode</code> on <code>fopen</code> , <code>freopen</code> , and <code>fdopen</code>	161
Figure 5.2	Conversion characters	177
Figure 5.3	Precision modifiers.....	179
Figure 5.4	Conversion specification flag characters.....	181
Figure 5.5	Size modifiers	182
Figure 5.6	The effect of buffer size on standard (character) I/O.....	197

Figure 5.7	The effect of buffer size on standard (line and unlocked) I/O.....	197
Figure 5.8	Summary results from Programs 5.8 and 5.9	198
Figure 5.9	I/O buffer sizes for different values of <code>buf_size</code>	198
Figure 6.1	The relationship of the various time functions.....	209
Figure 7.1	Relationship between the functions that modify the different user IDs	233
Figure 7.2	The different ways to change the three user IDs	234
Figure 8.1	Availability of <code>sysinfo</code> commands.....	280
Figure 8.2	Variables that return <code>-1</code> without setting <code>errno</code>	291
Figure 8.3	The effect of limits on implementation-defined constants	300
Figure 8.4	Resource controls available in Solaris 9	303
Figure 10.1	The effect of <code>dirname</code> and <code>basename</code> on different paths	355
Figure 10.2	File type macros.....	358
Figure 10.3	Masks to determine file access permissions	364
Figure 10.4	Other file access permission masks	364
Figure 10.5	Masks to determine file access permissions	369
Figure 10.6	Logical layout of file systems, cylinder groups, and inode lists	374
Figure 10.7	File system in more detail	376
Figure 10.8	File system detail after creating a test directory.....	378
Figure 10.9	How symbolic links are handled by various functions	382
Figure 10.10	The effect of functions on a file's access, update, and inode update times	389
Figure 11.1	Standard file system types.....	433
Figure 11.2	The standard <code>mount</code> options	434
Figure 11.3	Values of <code>fs_clean</code>	449
Figure 11.4	Constants that help with superbblock I/O.....	449
Figure 11.5	UFS inode direct and indirect blocks.....	453
Figure 11.6	Macros for handling inode numbers.....	453
Figure 12.1	Terminal device input and output queues	463
Figure 12.2	Terminal line discipline	464
Figure 12.3	Terminal flag summary.....	466
Figure 12.4	Summary of terminal I/O functions	468
Figure 12.5	Relationship between the terminal device functions	468
Figure 12.6	Summary of special input characters.....	469
Figure 12.7	Availability of special input characters	470
Figure 12.8	The four cases for non-canonical input.....	497
Figure 13.1	The effect of <code>O_NONBLOCK</code> and <code>O_NDELAY</code> on a blocking <code>read</code>	508
Figure 13.2	The effect of <code>O_NONBLOCK</code> and <code>O_NDELAY</code> on a blocking <code>write</code>	509
Figure 13.3	The interoperability of different lock types.....	512
Figure 13.4	The effect of mandatory locking on I/O by other processes.....	521
Figure 13.5	Effect of mandatory locking on file I/O performance	522
Figure 13.6	Anatomy of a simple stream	525
Figure 13.7	STREAMS message type created by <code>write</code> , <code>putmsg</code> , and <code>putpmsg</code>	529
Figure 13.8	STREAMS message type retrieved by <code>read</code> , <code>getmsg</code> , and <code>getpmsg</code>	530
Figure 13.9	Overview of the <code>talk</code> program	537
Figure 13.10	The <code>talk</code> program using two processes	538
Figure 13.11	Visualization of the <code>fd_set</code> data type	540
Figure 13.12	The effect of <code>poll</code> and <code>/dev/poll</code> on nonblocking I/O efficiency	549
Figure 13.13	Details of the array of <code>iovec</code> structures used by <code>readv</code> and <code>writev</code>	554
Figure 13.14	Timing results for different methods of writing two buffers.....	555
Figure 13.15	Memory mapped file	563
Figure 13.16	Comparing the time to copy a file using Programs 4.2 and 13.16	567

Figure 14.1	Relationship of a C program's start-up and termination functions	607
Figure 14.2	An example environment variable list.....	611
Figure 14.3	Address space for 32-bit SPARC sun4u processes	615
Figure 14.4	Stack frames after <code>do_foo</code> has been called.....	621
Figure 15.1	Sharing of open files between parent and child	636
Figure 15.2	Summary of the six <code>exec</code> functions	661
Figure 15.3	Relationship of the six <code>exec</code> functions.....	661
Figure 15.4	Values for <code>ac_flag</code> in a process accounting record	671
Figure 15.5	Process hierarchy for Program 15.14.....	674
Figure 16.1	Arrangement of processes to allow terminal logins	678
Figure 16.2	Arrangement of processes once we've logged in from a terminal	679
Figure 16.3	Sequence of processes when running the TELNET server.....	681
Figure 16.4	Arrangement of processes once we've logged in from the network.....	681
Figure 16.5	Arrangement of a process and its offspring	682
Figure 16.6	Arrangement of processes in a process group	683
Figure 16.7	A session consisting of three process groups.....	685
Figure 16.8	A session with three process groups showing the controlling terminal.....	687
Figure 16.9	Summary of job control features.....	692
Figure 16.10	Processes in a pipeline when invoked by <code>/bin/sh</code>	695
Figure 16.11	Processes in a pipeline when invoked by <code>/bin/ksh</code>	697
Figure 16.12	Arrangement of processes from Program 16.1	700
Figure 17.1	Solaris signal summary.....	706
Figure 17.2	Features provided by the different signal handling functions	724
Figure 17.3	Reentrant functions that can be called from a signal handler	726
Figure 17.4	Similarities between the SVR4 and POSIX signal mechanisms	746
Figure 17.5	Values of <code>si_code</code> for system-generated signals	758
Figure 17.6	Time line for Program 17.16.....	767
Figure 17.7	Process groups for Program 17.21	779
Figure 18.1	Details of the STREAMS <code>log</code> facility	808
Figure 18.2	Details of the <code>syslog</code> facility	812
Figure 19.1	Conceptual representation of a pipe	830
Figure 19.2	Arrangement of SVR4 STREAMS pipes.....	831
Figure 19.3	Arrangement of a pipe after <code>fork</code>	831
Figure 19.4	Arrangement of a pipe between parent and child	832
Figure 19.5	Arrangement of pipes for synchronizing parent and child processes.....	837
Figure 19.6	Result of <code>fp = popen (command, "r")</code>	839
Figure 19.7	Result of <code>fp = popen (command, "w")</code>	840
Figure 19.8	Driving a coprocess by using two one-way pipes	848
Figure 19.9	Driving a coprocess by using a full-duplex pipe	849
Figure 19.10	Procedure that processes filtered data twice	855
Figure 19.11	Process arrangement when using a FIFO to connect two pipelines	856
Figure 19.12	Clients sending requests to a server via a well-known FIFO	857
Figure 19.13	Client-server communications using FIFOs	857
Figure 20.1	The effect of flags when creating or opening an IPC object.....	871
Figure 20.2	Summary of IPC permissions	872
Figure 20.3	System V message queue tunables	874
Figure 20.4	Message type retrieved by <code>msgrcv</code> for different values of <code>msgtyp</code>	877
Figure 20.5	System V semaphore tunables	892
Figure 20.6	System V shared memory tunables	909
Figure 20.7	Performance comparison of message queues and pipes	920

Figure 20.8	Performance comparison of semaphores and record locking	924
Figure 21.1	Passing a file descriptor from one process to another	928
Figure 21.2	Pipe after pushing the <code>connld</code> module onto one end	942
Figure 21.3	Client-server connections on a named pipe.....	942
Figure 22.1	The three types of function call scenarios.....	952
Figure 23.1	Arrangement of pseudo terminals.....	998
Figure 23.2	Arrangement of processes when using <code>dtterm</code>	999
Figure 23.3	Arrangement of processes when using <code>sshd</code> to provide network logins	1000
Figure 23.4	Arrangement of processes when running the <code>script</code> program	1001
Figure 23.5	Driving a coprocess via a pseudo terminal	1002
Figure 23.6	Monitoring the output of a long-running program using a pseudo terminal.....	1003
Figure 23.7	Arrangement of processes when running <code>cat</code> from our <code>pty</code> program.....	1017
Figure 23.8	Arrangement of processes when using our <code>ssp_script</code> shell script.....	1020
Figure 23.9	Running a coprocess with a pseudo terminal as its input and output	1021
Figure 23.10	Arrangement of processes when using a driver program with <code>pty</code>	1023
Figure A.1	Precedence of environment variables for setting locales	1038
Figure B.1	Obsolescent SCP-originated functions	1053
Figure B.2	SCP-originated functions that are not obsolescent.....	1054
Figure C.1	Function availability (from <code>_exit</code> to <code>asctime</code>).....	1104
Figure C.2	Function availability (from <code>asctime</code> to <code>door_create</code>)	1105
Figure C.3	Function availability (from <code>door_cred</code> to <code>fgets</code>).....	1106
Figure C.4	Function availability (from <code>fgetspent</code> to <code>getgrgid</code>)	1107
Figure C.5	Function availability (from <code>getgrgid_r</code> to <code>getutxent</code>)	1108
Figure C.6	Function availability (from <code>getutxid</code> to <code>malloc</code>).....	1109
Figure C.7	Function availability (from <code>mallopt</code> to <code>psignal</code>).....	1110
Figure C.8	Function availability (from <code>ptsname</code> to <code>rmdir</code>)	1111
Figure C.9	Function availability (from <code>sbrk</code> to <code>sigdelset</code>)	1112
Figure C.10	Function availability (from <code>sigemptyset</code> to <code>strncat</code>)	1113
Figure C.11	Function availability (from <code>strncmp</code> to <code>ttyname_r</code>).....	1114
Figure C.12	Function availability (from <code>tzset</code> to <code>writev</code>)	1115
Figure D.1	Summary of our standard error functions	1119
Figure E.1	Summary of running different versions of Program 1.10.....	1128
Figure E.2	Minimum set of files required in a <code>chroot jail</code>	1134
Figure E.3	Stack frame for Program E.7.....	1141
Figure E.4	Stack frames while Program 17.7 is executing	1145