

CONTENTS IN DETAIL

ACKNOWLEDGMENTS	xxv
------------------------	------------

PREFACE	xxvii
----------------	--------------

INTRODUCTION	xxix
---------------------	-------------

What Will You Learn from This Book?	xxix
How Will This Book Teach You?	xxx
Using Open Source Software	xxxi
Presenting Different Perspectives	xxxi
Using Example Programs	xxxii
What Should You Know to Understand This Book?	xxxii
The Role of C in This Book	xxxii
Utility Programs	xxxiii
System Requirements	xxxiii
About UNIX, Unix, Linux, and More	xxxiv
Scope, Content, and Organization	xxxiv
Chapter Organization	xxxv
Online Materials	xxxvii
Conventions and Format	xxxviii
Typographical Conventions	xxxviii
Notation	xxxviii
Example Program Naming Conventions	xl
Dates and Identities	xl
Suggestions and Corrections	xli

1	
CORE CONCEPTS	1

What Is System Programming?	2
The Magic of Input and Output	2
The Role of the C Library in I/O	4
System Resources	5
System Programs Explained	6
Fundamental Concepts of Unix	7
The Unix Kernel	8
Shells and Commands	11
Users and Groups	13
Privileged and Nonprivileged Instructions	15

Environments	15
Files, Directories, and the Single Directory Hierarchy	17
Processes	25
Threads	27
Online Documentation	29
Using the Manual Pages	34
The Pager	34
The Structure of Man Pages	35
Searching Through the Man Pages	38
Unix History and Standards	41
The Birth of UNIX	41
Early Branches	42
The Free Software Foundation and GNU	42
The Rise of Linux	43
Many Unixes	43
Unix and Related Standards	43
C Standards	45
Summary	47
Exercises	47

2 **FUNDAMENTALS OF SYSTEM PROGRAMMING** **49**

Object Libraries	50
System Libraries	51
Static and Shared Libraries	52
The Advantages of Shared Libraries	53
Commands to Query a Library's Contents	54
Commands to Show the Libraries Linked to a Program	56
The C Standard Library	57
System Calls	58
Wrapper Functions	59
System Call Execution	59
Multiple Paths to Kernel Services	61
Handling Errors from System Calls and Library Functions	62
System Call Errors	62
Errors from Library Functions	66
Portability	67
Feature Test Macros	67
Other Portability Issues	70
System Limits	71
Internationalization	71
Processing the Command Line and Environment	72
Extracting Command Line Arguments	73
Accessing the Environment	75
Reporting Usage Errors	78
Extracting the Program Name	79

Extracting Command Line Options	81
Extracting Numbers from Strings	86
Summary	89
Exercises	90

3

TIME, DATES, AND LOCALES

93

Learning System Programming	93
Organization of Common Code	95
Functions for Extracting Numbers	97
Common Error-Handling Functions	102
File Organization	103
Planning Our First System Program	104
Designing the First Version of <code>spl_date</code>	107
About Calendar Time in Unix	108
Broken-Down Time	108
Calendar Time System Calls	109
Time Conversion Functions	112
Designing a Second Version of <code>spl_date</code>	114
Designing a Third Version of <code>spl_date</code>	116
The User Interface	116
Program Logic	119
Working with Locales	128
Locale Categories	129
About Time Zones	131
The Command-Level Interface to Locales	132
The Structure of Locales	135
The Programming Interface to Locales	138
An Internationalized Version of the <code>spl_date</code> Program	139
Other Ways to Internationalize Programs	140
Locale Objects	144
Summary	146
Exercises	147

4

BASIC CONCEPTS OF FILE I/O

149

High-Level vs. Low-Level File I/O	150
Universal I/O	150
File Permissions Revisited	150
Applying the Umask	151
Setting and Getting Umasks	152
Propagating Umasks	153
A Process's User IDs	153
The <code>setuid</code> Bit	154
Input/Output Mechanics	155

Standard File Descriptors	157
The Kernel I/O Interface	157
Opening Files	158
Closing Files	163
Reading from Files	165
Writing to Files	169
Writing a copy Command	171
Design of the copy Program	173
Implementation of the copy Program	174
Testing of the copy Program	176
The Universality of the copy Program	177
Timing Programs	179
Buffering and Running Time	182
Summary	185
Exercises	186

5

FILE I/O AND LOGIN ACCOUNTING 187

Controlling the Position of I/O Operations	188
The lseek() System Call	188
File Holes	190
Displaying Last Login Information	193
The lastlog Command	194
The lastlog File	196
The lastlog Structure	196
Usernames, User IDs, and the passwd File	199
The Password Database	202
Accessing All User Entries	203
Developing a lastlog Program	205
Design Considerations	206
Program Logic	208
Writing the Program	209
Developing a last Command	213
Login Records	215
The utmp Structure	217
The utmpx API	220
Logins, Logouts, and the utmp and wtmp Files	220
A Program to Show the utmp and wtmp Files	222
Analysis of the wtmp File	227
Designing the spl_last Program	230
User Space Buffering of Input	242
Summary	244
Exercises	245

6

OVERVIEW OF FILESYSTEMS AND FILES

247

Disks and Disk Partitions	248
Disk Geometry	248
Disk Device Drivers	250
Disk Partitioning	250
Many, Many Filesystems	252
Filesystems Supported by Linux	252
The Ext Filesystems	253
Filesystem Structure	253
Partition Layout	254
Block Group Layout	255
Performance Considerations	257
The Kernel's Filesystem Interface	259
Creating a New File	259
Writing Data to a File	260
The Virtual Filesystem	261
Exploring the Filesystem API	263
The stat Command	264
The stat() System Call	266
The stat Structure	267
The File Mode	270
The setgid Bit	272
The sticky Bit	273
An Example lstat Program	274
The statx() System Call	278
The statx Data Structure	278
Calling statx()	281
Writing an spl_stat Command	285
Designing the main() Function	285
Designing the print_statx() Function	287
Writing the Auxiliary Functions	292
Designing an Enhanced spl_stat Command	297
Writing an spl_statfs Command	299
The statfs() System Call	299
The statvfs() Library Function	300
A Hybrid Solution	301
Testing spl_statfs	307
Summary	308
Exercises	309

7	THE DIRECTORY HIERARCHY	311
Directory Structure	312	
Processing Directories	313	
The readdir() Library Function	315	
The dirent Structure	316	
Directory Streams	318	
The opendir() Library Function	319	
The closedir() Library Function	320	
A Simple ls Program	321	
Other Functions in the Directory API	324	
The telldir() and seekdir() Library Functions	325	
The scandir() Library Function	329	
Processing the Directory Hierarchy	335	
Mounting File Systems	337	
An Example of Filesystem Mounting	337	
Commands for Finding Mount Points	339	
Duplicate Inode Numbers	340	
Tree Walks	341	
A Recursive Tree Walk Using readdir()	341	
A Recursive Tree Walk Using scandir()	345	
The nftw() Tree Walk Function	347	
Writing a du Command	354	
The fts Tree Traversal Functions	365	
The pwd Command	371	
An Exercise in Constructing a Directory Tree	371	
A Strategy for Implementing the pwd Command	374	
Summary	381	
Exercises	382	

8	INTRODUCTION TO SIGNALS	383
The Role of Signals	384	
A Signal Delivery Example	385	
Sources of Signals	386	
Signal Concepts	387	
The Lifetime of a Signal	387	
Signal Types	389	
Basic Signal Handling	393	
The signal() System Call	395	
The System V signal() Semantics	397	
Sending Signals	400	
Blocking Signals	405	
Signal Sets	407	
The sigprocmask() Function	408	

The sigaction() System Call	416
The sigaction Structure	417
Signal Information Passed to the Handler	419
Effect of sa_flags on Signal Handler Execution	424
Guidance on Designing Signal Handlers	431
Summary	433
Exercises	434

9 TIMERS AND SLEEP FUNCTIONS 435

Keeping Track of Time	436
Alarm Clocks and Timers	436
Sleep Functions and Timers	437
Time, Clocks, and Timing	438
Overview	438
Hardware Clocks and Hardware Timers	439
The System Clock	440
High-Resolution Sleep Functions	440
The nanosleep() System Call	441
The clock_nanosleep() System Call	447
Software Timers	450
The alarm() System Call	450
A Progress Bar Based on Alarms	454
Interval Timers	461
Overview	461
POSIX Timers	462
A POSIX Timer-Based Progress Bar	466
Resource Monitors	471
Real-Time Signals and Multiple Timers	484
Summary	487
Exercises	488

10 PROCESS FUNDAMENTALS 491

Processes Revisited	492
The Process Tree	493
Process Groups	493
Sessions	496
Foreground and Background Processes and Process Groups	497
Program Files	498
The Contents of an Executable File	498
The Executable and Linking Format	499
The Virtual Memory Layout of a Process	511
The Text Segment	513
The Initialized Data Segment	513

The Uninitialized Data Segment	513
The Heap	513
The Stack Segment	514
A Program That Displays Virtual Memory Locations	515
The Kernel's Process Representation	517
Process Metadata	517
Overview of the Process Descriptor	518
The proc Pseudofilesystem	521
Numbered Directories	521
The Magic of /proc	522
Useful Per-Process Files	523
An ancestors Command	524
A Simple ps Command	526
Summary	534
Exercises	536

11

PROCESS CREATION AND TERMINATION 539

The Lifetime of a Process	540
Creating Processes	540
The Basics of fork()	541
The Child's Memory Image	543
The Child's Process Descriptor	545
Sharing of Open Files	546
Potential Race Conditions	551
Process Synchronization with Signals	553
Other Functions That Create Processes	556
Terminating Processes	557
Executing Programs	560
The execve() System Call	561
The exec() Library Functions	565
Waiting for Children	569
The wait() and waitpid() System Calls	570
The waitid() System Call	582
The SIGCHLD Signal and Asynchronous Waiting	582
Putting It All Together: A Simple Shell	590
The system() Library Function	593
Summary	594
Exercises	595

12

INTRODUCTION TO INTERPROCESS COMMUNICATION 597

Why Do We Need IPC?	598
An Overview of Interprocess Communication	598
Shared Memory Methods	598
Data Transfer Methods	599

Two Different APIs	600
Summary of the Common IPC Facilities	601
POSIX Shared Memory	602
Overview	603
The Shared Memory API	605
A Shared Memory Example Program	607
Pointer Pitfalls in Shared Memory	610
Race Conditions	613
Semaphores	614
Overview	614
System V Semaphores	616
POSIX Semaphores	616
A Shared Memory Producer Consumer Program	623
POSIX Message Queues	628
A Simple Message Queue Example	631
Message Queues and Asynchronous Notification	634
A Program Receiving Asynchronous Notifications	636
Summary	642
Exercises	643

13

PIPES AND FIFOs

645

An Overview of Pipes	645
Pipe Basics	646
Unnamed Pipes	648
The Behavior of Read Operations on Pipes	651
The Behavior of Write Operations on Pipes	652
A Producer-Consumer Example	653
A Shell Pipe Simulation	656
Best Practices Regarding Pipes	662
The popen() and pclose() Library Functions	663
FIFOs	667
Creating Named Pipes in the Shell	667
Creating FIFOs	669
Opening FIFOs	670
Putting It All Together: A Simple FIFO-Based Server-Like Program	671
Summary	676
Exercises	677

14

CLIENT-SERVER APPLICATIONS AND DAEMONS

679

Introduction to Client-Server Applications	680
System Logging Facilities	681
Daemons	683
Overview	683
Converting Processes into Daemons	684

An Iterative Server	686
Overview of the Application	686
The spl_calc Common Header File	689
The spl_calc Client Program	690
The spl_calc Server Program	692
A Concurrent Server	695
The Concurrent Server Client	697
The Concurrent Server	702
Summary	706
Exercises	707

15

INTRODUCTION TO THREADS

709

Background	710
Threads and Processes	711
Support in the Kernel	711
Pros and Cons of Multithreading	711
Shared Resources and Attributes	712
Program Design Considerations with Threads	715
Overview of the Pthreads Library	716
Thread Management	717
Creating a Thread	717
Exiting a Thread	718
Joining a Thread	718
Passing Data to Threads	720
Identifying Threads	722
Detaching Threads	722
Canceling a Thread	724
Setting Thread Stack Size	725
Signals and Threads	727
Thread-Directed Signals	727
Process-Directed Signals	728
Signal Masks and Dispositions	728
A Multithreaded Concurrent Server	731
Summary	735
Exercises	736

16

THREAD SYNCHRONIZATION

739

Correctness and Performance Considerations	740
Mutexes	740
Declaring and Initializing a Mutex	741
Locking and Unlocking a Mutex	742
Destroying a Mutex	743
A Program Using a Normal Mutex	743
Other Types of Mutexes	749

Condition Variables	751
Why Do We Need Condition Variables?	752
The Typical Steps for Using Condition Variables	753
Declaring and Initializing a Condition Variable	754
Waiting on a Condition Variable	754
Signaling a Condition Variable	755
Destroying a Condition Variable	756
Condition Attributes	757
A Multithreaded Multiple Producer, Multiple Consumer Program	757
Barrier Synchronization	762
Pthreads Barriers	764
A Program Using Barrier Synchronization	765
Read-Write Locks	774
Read-Write Lock API Overview	775
Use and Semantics of Read-Write Locks	776
Further Details About Pthreads Read-Write Locks	778
Read-Write Lock Example	779
Summary	787
Exercises	788

17

ALTERNATIVE METHODS OF I/O	791
Nonblocking I/O	792
Enabling Nonblocking I/O	792
A Program to Demonstrate Nonblocking Input	793
Signal-Driven I/O	796
Overview	797
Procedure for Enabling Signal-Driven I/O	798
Events Causing Signal Generation	799
Real-Time Signals and Signal-Driven I/O	802
A Program Using Signal-Driven I/O	802
POSIX Asynchronous I/O	807
Overview	807
The AIO API	808
Performance Benefits of Asynchronous I/O with Disk Files	813
An AIO-Based Implementation of spl_cp1.c	816
Multiplexed I/O	819
Overview	820
The select() System Call	820
An Example Program	824
Summary	830
Exercises	831

18

TERMINALS AND TERMINAL I/O

833

About Interactive Programs	834
An Overview of Terminals	834
An Experiment	835
An Explanation	836
Terminal Drivers	838
Terminal Driver Structure	838
The Terminal Driver as Seen from the Shell	841
Categories of Terminal Attributes	844
The Terminal Driver API	847
Terminal Switches	849
Terminal Special Characters	851
Terminal Attribute Modification	852
Writing an <code>spl_stty</code> Command	856
Program Data Structures	857
The <code>main()</code> Function of <code>spl_stty</code>	860
Functions That Display Attributes	861
Functions That Set Attributes	866
The <code>ioctl()</code> System Call	870
The Form of an <code>ioctl()</code> Call	871
Summary	875
Exercises	876

19

INTERACTIVE PROGRAMMING AND THE NCURSES LIBRARY

877

Canonical and Noncanonical Modes	878
Canonical Mode	878
Overview of Noncanonical Modes	879
The <code>MIN</code> and <code>TIME</code> Parameters	880
An Interactive Program in Noncanonical Mode	884
Program Features and Issues	885
Terminal Control Functions	887
Global Constants, Types, and Variables	888
Support Functions	889
The <code>sprite.c</code> <code>main()</code> Function	893
Curses and the <code>ncurses</code> Library	894
History, Standards, and Names	895
Terminology	896
Compiling, Building, and Running Curses Programs	897
Curses Basics	897
The Curses API	901
A Program with Tiled Windows	909
A Curses Version of <code>sprite.c</code>	911

The top Program	915
Requirements of a Simplified top Command.....	916
Design Considerations.....	917
Input Mode and the Cursor	918
Screen Management	919
Data Structures	924
Sorting Functions.....	928
Acquiring and Storing the Data.....	929
Acquiring Summary Data	930
Acquiring Per-Process Data.....	934
The main() Function	938
Concluding Thoughts	940
Summary	940
Exercises	941

A

CREATING LIBRARIES 943

About Libraries	944
Static vs. Shared Libraries in Unix.....	944
Identifying Libraries.....	946
Creating a Static Library	946
Using (Linking to) a Static Library.....	947
Creating a Shared Library.....	949
Shared Library Names.....	949
Steps to Create the Library	950
Using a Shared Library	951

B

UNICODE AND UTF-8 955

Background.....	956
Terminology	956
Unicode	957
UTF-8	957
Conversion Example 1	959
Conversion Example 2	960

C

DATE AND TIME FORMAT SPECIFIERS 961

BIBLIOGRAPHY 965

INDEX 969