

---

# Table of Contents

<b>Foreword.....</b>	<b>xv</b>
<b>Preface.....</b>	<b>xvii</b>
<b>1. Introduction and Essential Concepts.....</b>	<b>1</b>
System Programming	1
Why Learn System Programming	2
Cornerstones of System Programming	3
System Calls	3
The C Library	4
The C Compiler	4
APIs and ABIs	5
APIs	5
ABIs	6
Standards	7
POSIX and SUS History	7
C Language Standards	8
Linux and the Standards	8
This Book and the Standards	9
Concepts of Linux Programming	10
Files and the Filesystem	10
Processes	16
Users and Groups	18
Permissions	19
Signals	20
Interprocess Communication	20
Headers	21
Error Handling	21

Getting Started with System Programming	24
<b>2. File I/O.....</b>	<b>25</b>
Opening Files	26
The open() System Call	26
Owners of New Files	29
Permissions of New Files	29
The creat() Function	31
Return Values and Error Codes	32
Reading via read()	32
Return Values	33
Reading All the Bytes	34
Nonblocking Reads	35
Other Error Values	35
Size Limits on read()	36
Writing with write()	36
Partial Writes	37
Append Mode	38
Nonblocking Writes	38
Other Error Codes	38
Size Limits on write()	39
Behavior of write()	39
Synchronized I/O	40
fsync() and fdatasync()	41
sync()	43
The O_SYNC Flag	43
O_DSYNC and O_RSYNC	44
Direct I/O	45
Closing Files	45
Error Values	46
Seeking with lseek()	46
Seeking Past the End of a File	47
Error Values	48
Limitations	48
Positional Reads and Writes	49
Error Values	50
Truncating Files	50
Multiplexed I/O	51
select()	52
poll()	58
poll() Versus select()	61
Kernel Internals	62

The Virtual Filesystem	62
The Page Cache	63
Page Writeback	65
Conclusion	66
<b>3. Buffered I/O.....</b>	<b>67</b>
User-Buffered I/O	67
Block Size	69
Standard I/O	70
File Pointers	70
Opening Files	71
Modes	71
Opening a Stream via File Descriptor	72
Closing Streams	73
Closing All Streams	73
Reading from a Stream	73
Reading a Character at a Time	74
Reading an Entire Line	75
Reading Binary Data	76
Writing to a Stream	77
Writing a Single Character	78
Writing a String of Characters	78
Writing Binary Data	79
Sample Program Using Buffered I/O	79
Seeking a Stream	80
Obtaining the Current Stream Position	82
Flushing a Stream	82
Errors and End-of-File	83
Obtaining the Associated File Descriptor	84
Controlling the Buffering	84
Thread Safety	86
Manual File Locking	87
Unlocked Stream Operations	88
Critiques of Standard I/O	89
Conclusion	90
<b>4. Advanced File I/O.....</b>	<b>91</b>
Scatter/Gather I/O	92
readv() and writev()	92
Event Poll	97
Creating a New Epoll Instance	97
Controlling Epoll	98

Waiting for Events with Epoll	101
Edge- Versus Level-Triggered Events	103
Mapping Files into Memory	104
mmap()	104
munmap()	109
Mapping Example	109
Advantages of mmap()	111
Disadvantages of mmap()	111
Resizing a Mapping	112
Changing the Protection of a Mapping	113
Synchronizing a File with a Mapping	114
Giving Advice on a Mapping	115
Advice for Normal File I/O	118
The posix_fadvise() System Call	118
The readahead() System Call	120
Advice Is Cheap	121
Synchronized, Synchronous, and Asynchronous Operations	121
Asynchronous I/O	123
I/O Schedulers and I/O Performance	123
Disk Addressing	124
The Life of an I/O Scheduler	124
Helping Out Reads	125
Selecting and Configuring Your I/O Scheduler	129
Optimizing I/O Performance	129
Conclusion	135
<b>5. Process Management.....</b>	<b>137</b>
Programs, Processes, and Threads	137
The Process ID	138
Process ID Allocation	138
The Process Hierarchy	139
pid_t	139
Obtaining the Process ID and Parent Process ID	140
Running a New Process	140
The Exec Family of Calls	140
The fork() System Call	145
Terminating a Process	148
Other Ways to Terminate	149
atexit()	149
on_exit()	151
SIGCHLD	151
Waiting for Terminated Child Processes	151

Waiting for a Specific Process	154
Even More Waiting Versatility	156
BSD Wants to Play: wait3() and wait4()	158
Launching and Waiting for a New Process	160
Zombies	162
Users and Groups	163
Real, Effective, and Saved User and Group IDs	163
Changing the Real or Saved User or Group ID	164
Changing the Effective User or Group ID	165
Changing the User and Group IDs, BSD Style	165
Changing the User and Group IDs, HP-UX Style	166
Preferred User/Group ID Manipulations	166
Support for Saved User IDs	167
Obtaining the User and Group IDs	167
Sessions and Process Groups	167
Session System Calls	169
Process Group System Calls	170
Obsolete Process Group Functions	172
Daemons	172
Conclusion	175
<b>6. Advanced Process Management.....</b>	<b>177</b>
Process Scheduling	177
Timeslices	178
I/O- Versus Processor-Bound Processes	179
Preemptive Scheduling	179
The Completely Fair Scheduler	180
Yielding the Processor	181
Legitimate Uses	182
Process Priorities	183
nice()	183
getpriority() and setpriority()	184
I/O Priorities	186
Processor Affinity	186
sched_getaffinity() and sched_setaffinity()	187
Real-Time Systems	190
Hard Versus Soft Real-Time Systems	190
Latency, Jitter, and Deadlines	191
Linux's Real-Time Support	192
Linux Scheduling Policies and Priorities	192
Setting Scheduling Parameters	196
sched_rr_get_interval()	199

Precautions with Real-Time Processes	201
Determinism	201
Resource Limits	204
The Limits	205
Setting and Retrieving Limits	209
<b>7. Threading.....</b>	<b>211</b>
Binaries, Processes, and Threads	211
Multithreading	212
Costs of Multithreading	214
Alternatives to Multithreading	214
Threading Models	215
User-Level Threading	215
Hybrid Threading	216
Coroutines and Fibers	216
Threading Patterns	217
Thread-per-Connection	217
Event-Driven Threading	218
Concurrency, Parallelism, and Races	218
Race Conditions	219
Synchronization	222
Mutexes	222
Deadlocks	224
Pthreads	226
Linux Threading Implementations	226
The Pthread API	227
Linking Pthreads	227
Creating Threads	228
Thread IDs	229
Terminating Threads	230
Joining and Detaching Threads	233
A Threading Example	234
Pthread Mutexes	235
Further Study	239
<b>8. File and Directory Management.....</b>	<b>241</b>
Files and Their Metadata	241
The Stat Family	241
Permissions	246
Ownership	248
Extended Attributes	250
Extended Attribute Operations	253

Directories	259
The Current Working Directory	260
Creating Directories	265
Removing Directories	267
Reading a Directory's Contents	268
Links	271
Hard Links	272
Symbolic Links	273
Unlinking	275
Copying and Moving Files	277
Copying	277
Moving	278
Device Nodes	280
Special Device Nodes	280
The Random Number Generator	281
Out-of-Band Communication	281
Monitoring File Events	283
Initializing inotify	284
Watches	285
inotify Events	287
Advanced Watch Options	290
Removing an inotify Watch	291
Obtaining the Size of the Event Queue	292
Destroying an inotify Instance	292
<b>9. Memory Management.....</b>	<b>293</b>
The Process Address Space	293
Pages and Paging	293
Memory Regions	295
Allocating Dynamic Memory	296
Allocating Arrays	298
Resizing Allocations	299
Freeing Dynamic Memory	301
Alignment	303
Managing the Data Segment	307
Anonymous Memory Mappings	308
Creating Anonymous Memory Mappings	309
Mapping /dev/zero	311
Advanced Memory Allocation	312
Fine-Tuning with malloc_usable_size() and malloc_trim()	314
Debugging Memory Allocations	315
Obtaining Statistics	315

Stack-Based Allocations	316
Duplicating Strings on the Stack	318
Variable-Length Arrays	319
Choosing a Memory Allocation Mechanism	320
Manipulating Memory	321
Setting Bytes	321
Comparing Bytes	322
Moving Bytes	323
Searching Bytes	324
Frobnicating Bytes	325
Locking Memory	325
Locking Part of an Address Space	326
Locking All of an Address Space	327
Unlocking Memory	328
Locking Limits	328
Is a Page in Physical Memory?	328
Opportunistic Allocation	329
Overcommitting and OOM	330
<b>10. Signals.....</b>	<b>333</b>
Signal Concepts	334
Signal Identifiers	334
Signals Supported by Linux	335
Basic Signal Management	340
Waiting for a Signal, Any Signal	341
Examples	342
Execution and Inheritance	344
Mapping Signal Numbers to Strings	345
Sending a Signal	346
Permissions	346
Examples	347
Sending a Signal to Yourself	347
Sending a Signal to an Entire Process Group	347
Reentrancy	348
Guaranteed-Reentrant Functions	349
Signal Sets	350
More Signal Set Functions	351
Blocking Signals	351
Retrieving Pending Signals	352
Waiting for a Set of Signals	353
Advanced Signal Management	353
The siginfo_t Structure	355

The Wonderful World of <code>si_code</code>	357
Sending a Signal with a Payload	361
Signal Payload Example	362
A Flaw in Unix?	362
<b>11. Time.....</b>	<b>363</b>
Time's Data Structures	365
The Original Representation	366
And Now, Microsecond Precision	366
Even Better: Nanosecond Precision	366
Breaking Down Time	367
A Type for Process Time	368
POSIX Clocks	368
Time Source Resolution	369
Getting the Current Time of Day	370
A Better Interface	371
An Advanced Interface	372
Getting the Process Time	372
Setting the Current Time of Day	373
Setting Time with Precision	374
An Advanced Interface for Setting the Time	374
Playing with Time	375
Tuning the System Clock	377
Sleeping and Waiting	380
Sleeping with Microsecond Precision	381
Sleeping with Nanosecond Resolution	382
An Advanced Approach to Sleep	383
A Portable Way to Sleep	385
Overruns	385
Alternatives to Sleeping	386
Timers	386
Simple Alarms	386
Interval Timers	387
Advanced Timers	389
<b>A. GCC Extensions to the C Language.....</b>	<b>395</b>
<b>B. Bibliography.....</b>	<b>407</b>
<b>Index.....</b>	<b>411</b>