Contents

Preface **xvii**

1 Preliminaries **1**

1.1 Procedures 1
1.2 Linux Distributions 3
1.3 Kernel Versions 4
1.4 Platforms 6
1.5 Hardware 7
1.6 Linux Driver Project 7
1.7 Documentation and Links 8

2 Device Drivers **11**

2.1 Types of Devices 11
2.2 Mechanism vs. Policy 14
2.3 Avoiding Binary Blobs 14
2.4 How Applications Use Device Drivers 16
2.5 Walking Through a System Call 16
2.6 Error Numbers 18
2.7 **printk()** 19
2.8 Labs 21

3 Modules I: Basics **23**

3.1 What is a Module? 23
3.2 A Trivial Example - Hello World 24
3.3 Module Utilities 25
3.4 Passing Parameters 27
3.5 Compiling a Module 28
3.6 Modules and Hot Plug 33
3.7 Labs 34

4 Character Devices **35**

4.1 Device Nodes . . . 36
4.2 Major and Minor Numbers . . . 36
4.3 Reserving Major/Minor Numbers . . . 38
4.4 Accessing the Device Node . . . 40
4.5 Registering the Device . . . 41
4.6 **udev** and **HAL** . . . 42
4.7 **file_operations** Structure . . . 44
4.8 Driver Entry Points . . . 46
4.9 The **file** and **inode** Structures . . . 49
4.10 Module Usage Count . . . 51
4.11 Labs . . . 52

5 Kernel Configuration and Compilation 53
5.1 Installation and Layout of the Kernel Source . . . 53
5.2 Kernel Browsers . . . 55
5.3 Kernel Configuration Files . . . 56
5.4 Rolling Your Own Kernel . . . 57
5.5 initrd and initramfs . . . 60
5.6 Labs . . . 62

6 Kernel Features 67
6.1 Components of the Kernel . . . 67
6.2 User-Space vs. Kernel-Space . . . 69
6.3 Scheduling Algorithms and Task Structures . . . 70
6.4 Process Context . . . 71
6.5 Labs . . . 72

7 Kernel Style and General Considerations 75
7.1 Coding Style . . . 76
7.2 kernel-doc . . . 77
7.3 Using Generic Kernel Routines and Methods . . . 77
7.4 Making a Kernel Patch . . . 78
7.5 sparse . . . 79
7.6 Using **likely()** and **unlikely()** . . . 80
7.7 Linked Lists . . . 81
7.8 Writing Portable Code - 32/64-bit, Endianness . . . 85
7.9 Writing for SMP . . . 85
7.10 Writing for High Memory Systems . . . 86

7.11 Keeping Security in Mind . . . 86
7.12 Mixing User- and Kernel-Space Headers . . . 86
7.13 Labs . . . 87

8 Interrupts and Exceptions 89
8.1 What are Interrupts and Exceptions? . . . 90
8.2 Exceptions . . . 90
8.3 Interrupts . . . 92
8.4 MSI . . . 94
8.5 Enabling/Disabling Interrupts . . . 95
8.6 What You Cannot Do at Interrupt Time . . . 96
8.7 IRQ Data Structures . . . 96
8.8 Installing an Interrupt Handler . . . 99
8.9 Labs . . . 101

9 Modules II: Exporting, Licensing and Dynamic Loading 103
9.1 Exporting Symbols . . . 104
9.2 Module Licensing . . . 104
9.3 Automatic Loading/Unloading of Modules . . . 106
9.4 Built-in Drivers . . . 107
9.5 Kernel Building and Makefiles . . . 109
9.6 Labs . . . 110

10 Debugging Techniques 113
10.1 **oops** Messages . . . 113
10.2 Kernel Debuggers . . . 116
10.3 **debugfs** . . . 118
10.4 **kprobes** and **jprobes** . . . 119
10.5 Labs . . . 122

11 Timing and Timers 125
11.1 Jiffies . . . 125
11.2 Time Stamp Counter . . . 127
11.3 Inserting Delays . . . 128
11.4 What are Dynamic Timers? . . . 129
11.5 Timer Functions . . . 129
11.6 Timer Implementation . . . 130
11.7 High Resolution Timers . . . 131

11.8 Using High Resolution Timers . . . 132
11.9 Labs . . . 135

12 Race Conditions and Synchronization Methods **137**
12.1 Concurrency and Synchronization Methods . . . 138
12.2 Atomic Operations . . . 139
12.3 Bit Operations . . . 140
12.4 Spinlocks . . . 141
12.5 Big Kernel Lock . . . 143
12.6 Mutexes . . . 144
12.7 Semaphores . . . 145
12.8 Completion Functions . . . 148
12.9 Reference Counts . . . 149
12.10Labs . . . 150

13 ioctls **153**
13.1 What are **ioctl**s? . . . 153
13.2 Driver Entry point for **ioctl**s . . . 154
13.3 Lockless **ioctl**s . . . 155
13.4 Defining **ioctl**s . . . 156
13.5 Labs . . . 158

14 The proc Filesystem **161**
14.1 What is the **proc** Filesystem? . . . 161
14.2 Creating Entries . . . 162
14.3 Reading Entries . . . 163
14.4 Writing Entries . . . 164
14.5 The **seq_file** Interface . . . 165
14.6 Labs . . . 167

15 Unified Device Model and sysfs **171**
15.1 Unified Device Model . . . 171
15.2 Basic Structures . . . 172
15.3 Real Devices . . . 174
15.4 sysfs . . . 175
15.5 Labs . . . 177

16 Firmware **179**
16.1 What is Firmware? . . . 179

16.2 Loading Firmware . . . 180
16.3 Labs . . . 180

17 Memory Management and Allocation **183**
17.1 Virtual and Physical Memory . . . 184
17.2 Memory Zones . . . 185
17.3 Page Tables . . . 186
17.4 **kmalloc()** . . . 186
17.5 **__get_free_pages()** . . . 188
17.6 **vmalloc()** . . . 188
17.7 Early Allocations and **bootmem()** . . . 189
17.8 Slabs and Cache Allocations . . . 190
17.9 Labs . . . 193

18 Transferring Between User and Kernel Space **195**
18.1 Transferring Between Spaces . . . 196
18.2 **put(get)_user()** and **copy_to(from)_user()** . . . 196
18.3 Direct transfer - Kernel I/O and Memory Mapping . . . 198
18.4 Kernel I/O . . . 199
18.5 Mapping User Pages . . . 200
18.6 Memory Mapping . . . 201
18.7 User-Space Functions for **mmap()** . . . 202
18.8 Driver Entry Point for **mmap()** . . . 204
18.9 Relay Channels . . . 207
18.10 Relay API . . . 208
18.11 Accessing Files from the Kernel . . . 209
18.12 Labs . . . 212

19 Sleeping and Wait Queues **213**
19.1 What are Wait Queues? . . . 213
19.2 Going to Sleep and Waking Up . . . 214
19.3 Going to Sleep Details . . . 216
19.4 Exclusive Sleeping . . . 218
19.5 Waking Up Details . . . 218
19.6 Polling . . . 220
19.7 Interrupt Handling in User-Space . . . 221
19.8 Labs . . . 222

20 Interrupt Handling and Deferrable Functions 225

20.1 Top and Bottom Halves . . . 225
20.2 Deferrable Functions and softirqs . . . 227
20.3 Tasklets . . . 228
20.4 Work Queues . . . 231
20.5 Creating Kernel Threads . . . 234
20.6 Threaded Interrupt Handlers . . . 235
20.7 Labs . . . 235

21 Hardware I/O 239

21.1 Buses and Ports . . . 240
21.2 Memory Barriers . . . 240
21.3 Registering I/O Ports . . . 241
21.4 Resource Management . . . 242
21.5 Reading and Writing Data from I/O Registers . . . 244
21.6 Slowing I/O Calls to the Hardware . . . 245
21.7 Allocating and Mapping I/O Memory . . . 246
21.8 Accessing I/O Memory . . . 247
21.9 Access by User - **ioperm(), iopl(), /dev/port** . . . 249
21.10Labs . . . 249

22 PCI 253

22.1 What is PCI? . . . 253
22.2 PCI Device Drivers . . . 256
22.3 PCI Structures and Functions . . . 258
22.4 Accessing Configuration Space . . . 259
22.5 Accessing I/O and Memory Spaces . . . 260
22.6 PCI Express . . . 261
22.7 Labs . . . 261

23 Direct Memory Access (DMA) 263

23.1 What is DMA? . . . 264
23.2 DMA and Interrupts . . . 264
23.3 DMA Memory Constraints . . . 265
23.4 DMA Directly to User . . . 266
23.5 DMA under PCI . . . 266
23.6 DMA Pools . . . 269
23.7 Scatter/Gather Mappings . . . 269

23.8 DMA under ISA . . . 271
23.9 Labs . . . 272

24 Network Drivers I: Basics **273**
24.1 Network Layers and Data Encapsulation . . . 273
24.2 Datalink Layer . . . 276
24.3 Network Device Drivers . . . 276
24.4 Loading/Unloading . . . 277
24.5 Opening and Closing . . . 278
24.6 Labs . . . 279

25 Network Drivers II: Data Structures **281**
25.1 **net_device** Structure . . . 281
25.2 **net_device_ops** Structure . . . 287
25.3 **sk_buff** Structure . . . 289
25.4 Socket Buffer Functions . . . 290
25.5 Labs . . . 293

26 Network Drivers III: Transmission and Reception **295**
26.1 Transmitting Data and Timeouts . . . 295
26.2 Receiving Data . . . 297
26.3 Statistics . . . 297
26.4 Labs . . . 298

27 Network Drivers IV: Selected Topics **301**
27.1 Multicasting . . . 302
27.2 Changes in Link State . . . 303
27.3 **ioctl**s . . . 303
27.4 NAPI and Interrupt Mitigation . . . 304
27.5 NAPI Details . . . 304
27.6 TSO and TOE . . . 305
27.7 MII and ethtool . . . 306

28 USB Drivers **309**
28.1 What is USB? . . . 310
28.2 USB Topology . . . 310
28.3 Descriptors . . . 311
28.4 USB Device Classes . . . 312
28.5 Data Transfer . . . 313
28.6 USB under Linux . . . 314

28.7 Registering USB Devices . . . 314
28.8 Example of a USB Driver . . . 317
28.9 Labs . . . 319

29 Memory Technology Devices **321**
29.1 What are MTD Devices? . . . 321
29.2 NAND vs. NOR . . . 322
29.3 Driver and User Modules . . . 324
29.4 Flash Filesystems . . . 324
29.5 Labs . . . 325

30 Power Management **329**
30.1 Power Management . . . 329
30.2 APM and ACPI . . . 330
30.3 System Power States . . . 331
30.4 Callback Functions . . . 332
30.5 Labs . . . 334

31 Notifiers **335**
31.1 What are Notifiers? . . . 335
31.2 Data Structures . . . 336
31.3 Callbacks and Notifications . . . 337
31.4 Creating Notifier Chains . . . 337
31.5 Labs . . . 338

32 CPU Frequency Scaling **339**
32.1 What is Frequency and Voltage Scaling? . . . 339
32.2 Notifiers . . . 340
32.3 Drivers . . . 342
32.4 Governors . . . 343
32.5 Labs . . . 344

33 Asynchronous I/O **345**
33.1 What is Asynchronous I/O? . . . 345
33.2 The Posix Asynchronous I/O API . . . 346
33.3 Linux Implementation . . . 347
33.4 Labs . . . 350

34 I/O Scheduling **351**
34.1 I/O Scheduling . . . 351

34.2 Tunables 353
34.3 noop I/O Scheduler 353
34.4 Deadline I/O Scheduler 354
34.5 Completely Fair Queue Scheduler 355
34.6 Anticipatory I/O Scheduler 355
34.7 Labs 356

35 Block Drivers **357**
35.1 What are Block Drivers? 357
35.2 Buffering 358
35.3 Registering a Block Driver 358
35.4 **gendisk** Structure 360
35.5 Request Handling 362
35.6 Labs 365