John Madieu 著

Linux设备驱动程序开发

（影印版）

Linux Device Drivers Development

东南大学出版社
SOUTHEAST UNIVERSITY PRESS

Packt

Linux 设备驱动程序开发(影印版)
Linux Device Drivers Development

John Madieu 著

南京　东南大学出版社

图书在版编目(CIP)数据

Linux 设备驱动程序开发:英文/(法)约翰·马杜(John Madieu)著. —影印本. —南京:东南大学出版社,2018.8

书名原文:Linux Device Drivers Development

ISBN 978-7-5641-7753-9

Ⅰ. ①L… Ⅱ. ①约… Ⅲ. ①Linux 操作系统-驱动程序-程序设计-英文 Ⅳ. ①TP316.89

中国版本图书馆 CIP 数据核字(2018)第 099725 号

图字:10-2018-097 号

© 2017 by PACKT Publishing Ltd.

Reprint of the English Edition, jointly published by PACKT Publishing Ltd and Southeast University Press, 2018. Authorized reprint of the original English edition, 2018 PACKT Publishing Ltd, the owner of all rights to publish and sell the same.

All rights reserved including the rights of reproduction in whole or in part in any form.

英文原版由 PACKT Publishing Ltd 出版 2017。

英文影印版由东南大学出版社出版 2018。此影印版的出版和销售得到出版权和销售权的所有者—— PACKT Publishing Ltd 的许可。

版权所有,未得书面许可,本书的任何部分和全部不得以任何形式重制。

Linux 设备驱动程序开发(影印版)

出版发行:	东南大学出版社
地　　址:	南京四牌楼 2 号　　邮编:210096
出 版 人:	江建中
网　　址:	http://www.seupress.com
电子邮件:	press@seupress.com
印　　刷:	常州市武进第三印刷有限公司
开　　本:	787 毫米×980 毫米　　16 开本
印　　张:	36.5
字　　数:	715 千字
版　　次:	2018 年 8 月第 1 版
印　　次:	2018 年 8 月第 1 次印刷
书　　号:	ISBN 978-7-5641-7753-9
定　　价:	108.00 元

本社图书若有印装质量问题,请直接与营销部联系。电话(传真):025-83791830

Credits

Author
John Madieu

Reviewer
Jérôme Pouiller

Commissioning Editor
Gebin George

Acquisition Editor
Gebin George

Content Development Editor
Devika Battike

Technical Editor
Swathy Mohan

Copy Editors
Juliana Nair
Safis Editing

Project Coordinator
Judie Jose

Proofreader
Safis Editing

Indexer
Rekha Nair

Graphics
Kirk D'Penha

Production Coordinator
Arvindkumar Gupta

About the Author

John Madieu is an embedded Linux and kernel engineer living in France, in Paris. His main activities consist of developing drivers and Board Support Packages (BSP) for companies in domains such as automation, transport, healthcare, energy, and the military. John works at EXPEMB, a French company that is a pioneer in electronical board design based on computer-on-module, and in embedded Linux solutions. He is an open source and embedded systems enthusiast, convinced that it is only by sharing knowledge that one learns more.

He is passionate about boxing, which he practised for 6 years professionally, and continues to transmit this passion through sessions of training that he provides voluntarily.

I would like to thank Devika Battike, Gebin George, and all the Packt team for their efforts to release this book on time. They are the people without whom this book would probably never have seen the light of day. It was a pleasure to work with them.

Finally, I would like to thank all the mentors I have had over the years, and who still continue to accompany me. Mentors such as Cyprien Pacôme Nguefack for his programming skills that I have learned over the years, Jérôme Pouillier and Christophe Nowicki for introducing me buildroot and leading me to kernel programming, Jean-Christian Rerat and Jean-Philippe DU-Teil of EXPEMB for their coaching and accompaniment in my professional career; to all those I could not mention, I wish to thank them for having transmitted these connoises to me, which I have tried to disseminate through this book.

试读结束：需要全本请在线购买： www.ertongbook.com

About the Reviewer

Jérôme Pouiller is a true geek and fascinated by understanding how things do work.

He was an early adopter of Linux. He found in Linux a system with no limits, where everything could be changed. Linux has provided an excellent platform to hack anything.

He graduated in machine learning at Ecole Pour l'Informatique et les Technologies Avancées (EPITA). Beside his studies, he learned electronics by himself. He quickly turned his attention to the piece of software at crossroad of all advanced systems: the operating system. It is now one of his favorite subjects.

For 15 years now, Jérôme Pouiller has designed (and often debugged) Linux firmware for a variety of industries (multimedia, healthcare, nuclear, military).

In addition to his consulting activities, Jérôme Pouiler is professor of operating systems at Institut National des Sciences Appliquées (INSA). He has written many course materials about system programming, operating system design, realtime systems, and more.

www.PacktPub.com

For support files and downloads related to your book, please visit www.PacktPub.com.

Did you know that Packt offers eBook versions of every book published, with PDF and ePub files available? You can upgrade to the eBook version at www.PacktPub.com and as a print book customer, you are entitled to a discount on the eBook copy. Get in touch with us at service@packtpub.com for more details.

At www.PacktPub.com, you can also read a collection of free technical articles, sign up for a range of free newsletters and receive exclusive discounts and offers on Packt books and eBooks.

Mapt

https://www.packtpub.com/mapt

Get the most in-demand software skills with Mapt. Mapt gives you full access to all Packt books and video courses, as well as industry-leading tools to help you plan your personal development and advance your career.

Why subscribe?

- Fully searchable across every book published by Packt
- Copy and paste, print, and bookmark content
- On demand and accessible via a web browser

Customer Feedback

Thanks for purchasing this Packt book. At Packt, quality is at the heart of our editorial process. To help us improve, please leave us an honest review on this book's Amazon page at `https://www.amazon.com/dp/1785280007`.

If you'd like to join our team of regular reviewers, you can e-mail us at `customerreviews@packtpub.com`. We award our regular reviewers with free eBooks and videos in exchange for their valuable feedback. Help us be relentless in improving our products!

I would like to thank my girlfriend for her support and all the sleepless nights accompanying the writing of this book, as well as Brigitte and François, my dear parents, for whom I have a thought and to whom I dedicate this book entirely.

- John Madieu

I would like to dedicate this book in the memory of my father, who left too.

- Jérôme Pouiller

Table of Contents

Preface 1

Chapter 1: Introduction to Kernel Development 9

- **Environment setup** 10
  - Getting the sources 10
    - Source organization 11
  - Kernel configuration 12
  - Build your kernel 13
- **Kernel habits** 14
  - Coding style 14
  - Kernel structures allocation/initialization 15
  - Classes, objects, and OOP 16
- **Summary** 17

Chapter 2: Device Driver Basis 19

- **User space and kernel space** 20
  - The concept of modules 21
  - Module dependencies 21
    - depmod utility 21
  - Module loading and unloading 22
    - Manual loading 22
      - modprobe and insmod 22
      - /etc/modules-load.d/<filename>.conf 22
    - Auto-loading 23
    - Module unload 23
- **Driver skeletons** 24
  - Module entry and exit point 25
    - __init and __exit attributes 25
  - Module information 27
    - Licensing 29
    - Module author(s) 30
    - Module description 31
- **Errors and message printing** 31
  - Error handling 31
  - Handling null pointer errors 34
  - Message printing – printk() 35
- **Module parameters** 37
- **Building your first module** 39

试读结束：需要全本请在线购买： www.ertongbook.com

The module's makefile 39
In the kernel tree 41
Out of the tree 44
Building the module 44
Summary 45
Chapter 3: Kernel Facilities and Helper Functions 47
Understanding container_of macro 47
Linked lists 50
Creating and initializing the list 51
Dynamic method 51
Static method 52
Creating a list node 52
Adding a list node 53
Deleting a node from the list 54
Linked list traversal 54
Kernel sleeping mechanism 55
Wait queue 55
Delay and timer management 58
Standard timers 59
Jiffies and HZ 59
Timers API 59
Timer setup initialization 60
Standard timer example 61
High resolution timers (HRTs) 62
HRT API 62
HRT setup initialization 62
Dynamic tick/tickless kernel 64
Delays and sleep in the kernel 64
Atomic context 64
Nonatomic context 65
Kernel locking mechanism 65
Mutex 66
Mutex API 66
Declare 66
Acquire and release 67
Spinlock 68
Spinlock versus mutexes 70
Work deferring mechanism 70
Softirqs and ksoftirqd 70
ksoftirqd 71
Tasklets 72
Declaring a tasklet 72

Enabling and disabling a tasklet 73
Tasklet scheduling 73
Work queues 75
Kernel-global workqueue – the shared queue 75
Dedicated work queue 78
Programming syntax 78
Predefined (shared) workqueue and standard workqueue functions 81
Kernel threads 82
Kernel interruption mechanism 82
Registering an interrupt handler 82
Interrupt handler and lock 85
Concept of bottom halves 86
The problem – interrupt handler design limitations 86
The solution – bottom halves 86
Tasklets as bottom halves 87
Workqueue as bottom halves 88
Softirqs as bottom half 89
Threaded IRQs 89
Threaded bottom half 91
Invoking user-space applications from the kernel 92
Summary 93
Chapter 4: Character Device Drivers 95
The concept behind major and minor 96
Device number allocation and freeing 97
Introduction to device file operations 98
File representation in the kernel 99
Allocating and registering a character device 101
Writing file operations 102
Exchanging data between kernel space and user space 102
A single value copy 103
The open method 104
Per-device data 104
The release method 105
The write method 106
Steps to write 106
The read method 108
Steps to read 109
The llseek method 110
Steps to llseek 111
The poll method 112
Steps to poll 113
The ioctl method 116

Generating ioctl numbers (command) 117
Steps for ioctl 118
Filling the file_operations structure 120
Summary 120

Chapter 5: Platform Device Drivers 121

Platform drivers 122
Platform devices 126
Resources and platform data 126
Device provisioning - the old and depreciated way 126
Resources 127
Platform data 129
Where to declare platform devices? 131
Device provisioning - the new and recommended way 131
Devices, drivers, and bus matching 132
How can platform devices and platform drivers match? 134
Kernel devices and drivers-matching function 135
OF style and ACPI match 136
ID table matching 136
Name matching - platform device name matching 140
Summary 140

Chapter 6: The Concept of Device Tree 141

Device tree mechanism 141
Naming convention 142
Aliases, labels, and phandle 143
DT compiler 144
Representing and addressing devices 145
SPI and I2C addressing 145
Platform device addressing 147
Handling resources 148
Concept of named resources 149
Accessing registers 150
Handling interrupts 151
The interrupt handler 151
Interrupt controller code 152
Extract application-specific data 153
Text string 153
Cells and unsigned 32-bit integers 154
Boolean 155
Extract and parse sub-nodes 155
Platform drivers and DT 156
OF match style 156
Dealing with non-device tree platforms 159

Support multiple hardware with per device-specific data 160
Match style mixing 162
Platform resources and DT 164
Platform data versus DT 166
Summary 167
Chapter 7: I2C Client Drivers 169
The driver architecture 170
The i2c_driver structure 170
The probe() function 171
Per-device data 172
The remove() function 173
Driver initialization and registration 174
Driver and device provisioning 174
Accessing the client 175
Plain I2C communication 175
System Management Bus (SMBus) compatible functions 177
Instantiating I2C devices in the board configuration file (old and depreciated way) 178
I2C and the device tree 179
Defining and registering the I2C driver 180
Remark 181
Instantiating I2C devices in the device tree - the new way 182
Putting it all together 182
Summary 183
Chapter 8: SPI Device Drivers 185
The driver architecture 186
The device structure 186
spi_driver structure 189
The probe() function 189
Per-device data 190
The remove() function 191
Driver initialization and registration 191
Driver and devices provisioning 192
Instantiate SPI devices in board configuration file – old and depreciated way 193
SPI and device tree 194
Instantiate SPI devices in device tree - the new way 196
Define and register SPI driver 196
Accessing and talking to the client 197
Putting it all together 202
SPI user mode driver 202
With IOCTL 204

Summary 207
Chapter 9: Regmap API – A Register Map Abstraction 209
Programming with the regmap API 210
regmap_config structure 211
regmap initialization 214
SPI initialization 214
I2C initialization 215
Device access functions 216
regmap_update_bits function 217
Special regmap_multi_reg_write function 218
Other device access functions 219
regmap and cache 219
Putting it all together 221
A regmap example 221
Summary 224
Chapter 10: IIO Framework 225
IIO data structures 227
iio_dev structure 227
iio_info structure 231
IIO channels 232
Channel attribute naming conventions 234
Distinguishing channels 236
Putting it all together 238
Triggered buffer support 241
IIO trigger and sysfs (user space) 245
Sysfs trigger interface 245
add_trigger file 245
remove_trigger file 246
Tying a device with a trigger 246
The interrupt trigger interface 246
The hrtimer trigger interface 247
IIO buffers 248
IIO buffer sysfs interface 248
IIO buffer setup 249
Putting it all together 251
IIO data access 257
One-shot capture 258
Buffer data access 258
Capturing using the sysfs trigger 258
Capturing using the hrtimer trigger 260
IIO tools 261

Summary 261

Chapter 11: Kernel Memory Management 263

System memory layout - kernel space and user space 265
Kernel addresses – concept of low and high memory 267
Low memory 268
High memory 268
User space addresses 269
Virtual Memory Area (VMA) 272
Address translation and MMU 274
Page look up and TLB 280
How does TLB work 280
Memory allocation mechanism 282
Page allocator 283
Page allocation API 283
Conversion functions 285
Slab allocator 286
The buddy algorithm 286
A journey into the slab allocator 289
kmalloc family allocation 291
vmalloc allocator 294
Process memory allocation under the hood 296
The copy-on-write (CoW) case 297
Work with I/O memory to talk with hardware 298
PIO devices access 298
MMIO devices access 299
__iomem cookie 300
Memory (re)mapping 302
kmap 302
Mapping kernel memory to user space 303
Using remap_pfn_range 303
Using io_remap_pfn_range 305
The mmap file operation 305
Implementing mmap in the kernel 307
Linux caching system 308
What is a cache? 308
CPU cache – memory caching 309
The Linux page cache – disk caching 310
Specialized caches (user space caching) 310
Why delay writing data to disk? 310
Write caching strategies 311
The flusher threads 312
Device-managed resources – Devres 312

试读结束：需要全本请在线购买： www.ertongbook.com