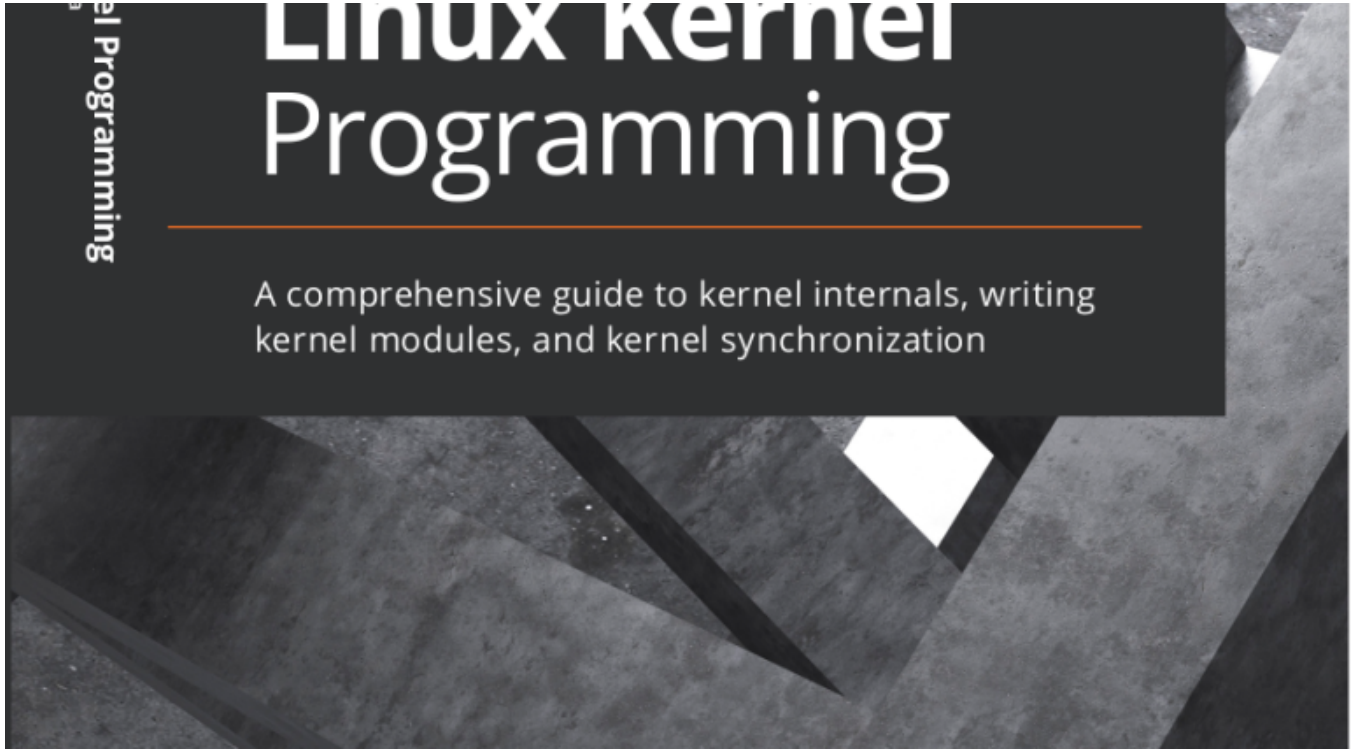
kaiwanTECH: Kaiwan's Tech Blog



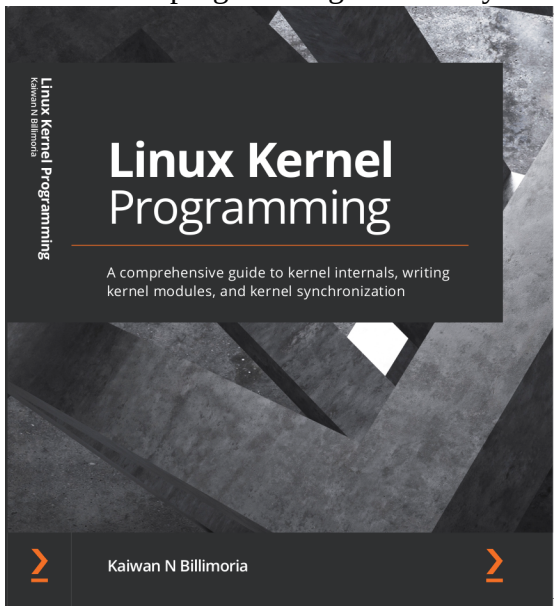
C, DEBUGGING, DESIGN, DEVICE DRIVERS, EMBEDDED LINUX, LINUX KERNEL, LINUX PROGRAMMING, PROGRAMMING

Linux Kernel Programming – my second book

MARCH 21, 2021 | KAIWAN | LEAVE A COMMENT |

I've recently completed a project – the writing of the *Linux Kernel Programming* book, published by Packt (it was announced by the publisher on 01 March 2021). This project took just over two years…

All those long days and nights, poring over the writing and the code, I now feel has definitely been very worth-while and that the book will be a useful contribution to the Linux programming community.



(https://packt.live/399V2uS)

A key point: I've ensured that all the material and code examples are based on the 5.4 LTS Linux kernel; it's slated to be maintained right through Dec 2025, thus keeping the book's content very relevant for a long while!

Due to its sheer size and depth, the publisher suggested we split the original tome into two books. That's what has happened:

- the first part, *Linux Kernel Programming,* covers the essentials and, in my opinion, should be read first (of course, if you're already very familiar with the topics it covers, feel free to start either way)
- the second part, *Linux Kernel Programming Part 2*, covering a small section of device driver topics, focusses on the basics and the character 'misc' class device driver framework.

Many cross-references, especially from the second book to topics in the first, do turn up; hence the suggestion to read them in order.

Here's a quick run down on what's covered in each book.

Lets begin with the ***Linux Kernel Programming*** book; firstly, it's targeted at people who are quite new to the world of Linux kernel development and makes no assumptions regarding knowledge of the kernel. The prerequisite is a working knowledge of programming on Linux with 'C'; it's the medium we use throughout (along with a few bash scripts). The book is divided into three major sections, each containing appropriate chapters:

- *Section 1* covers the basics: firstly, the appropriate setup of the kernel development workspace on your system; next, two chapters cover the building of the Linux kernel from scratch, from code. (It includes the cross compile as well, using the popular Raspberry Pi board as a 'live' example).
  - The following two chapters delve in-depth into the kernel's powerful Loadable Kernel Module (LKM) framework, how to program it along with more advanced features. I also try and take a lot of trouble to point out how one should code with security in mind!

- In *Section 2* we deal with having you, the reader, gain a deeper understanding (to the practical extent required) of key kernel internals topics. A big reason why many struggle with kernel development is a lack of understanding of its internals.
  - Here, Chapter 6 covers the kernel architecture, focusing on how the kernel maintains attribute information on processes/threads and their associated stacks.
  - The next chapter – a really key one, again – delves into a difficult topic for many – memory management internals. I try to keep the coverage focused on what matters to a kernel and/or driver developer.
  - The following two chapters dive into the many and varied ways to allocate and deallocate memory when working within the kernel – an area where you can make a big difference performance-wise by knowing which kernel APIs and methods to use when.
  - The remaining two chapters here round off kernel internals with discussion on the kernel-level CPU scheduler; several concepts and practical code examples have the reader learn what's required.

- *Section 3* is where the books dives into what folks new to it consider to be difficult and arcane matters – how and why synchronization matters, how data races occur and how you can protect critical sections in your kernel / driver code!
  - The amount of material here requires two chapters to do justice to: the first of them focuses on critical sections, concurrency concerns, the understanding and the practical usage of the mutex and the spinlock.
  - The book's last chapter continues this discussion on kernel synchronization covering more areas relevant to the modern kernel and/or driver developer – atomic (and refcount) operators, cache effects, a primer on 'lock-free' programming techniques, with one of them – the percpu one – covered in some detail. Lock debugging within the kernel – using the powerful *lockdep* validator – as well as other techniques is covered as well!

The second book – ***Linux Kernel Programming Part 2 – Char Device Drivers and Kernel Synchronization*** – deliberately covers just a small section of 'how to write a device driver on Linux'. It does *not* purport to cover the many types and aspects of device driver development, instead focusing on the basics of teaching the reader how to write a simple yet complete character device driver belonging to the 'misc' class.

Great news! This book – Linux Kernel Programming Part 2 – Char Device Drivers and Kernel Synchronization – is downloadable for FREE. Enjoy!

Access it now! (https://github.com/PacktPublishing/Linux-Kernel-Programming/blob/master/Linux-Kernel-Programming-(Part-2)/Linux%20Kernel%20Programming%20Part%202%20-%20Char%20Device%20Drivers%20and%20Kernel%20Synchronization_eBook.pdf)

Having said that, the materials covering user-kernel communication pathways, working with peripheral I/O memory, and especially, the topic on dealing with hardware interrupts, is very detailed and will prove to be very useful in pretty much all kinds of Linux device driver projects.

A quick chapter-wise run down of the second book:

- In *Chapter 1*, we cover the basics – the reader understands the basics of the Linux Device Model (LDM) and ends up writing a small, simple, yet complete 'misc' class character driver. Security-awareness is built too: we demonstrate a simple "privesc" – privilege escalation – attack
- *Chapter 2* shows the reader something every driver author will at one time or the other have to do: efficiently communicate between user and kernel address spaces. You'll learn to use various technologies to do so – via procfs, sysfs, debugfs (especially useful to insert debug hooks as well), netlink sockets and the ioctl system call
- The next chapter has the reader understand the nuances of reading and writing peripheral (hardware) I/O memory, via both the memory-mapped I/O (MMIO) as well as the Port I/O (PIO) technique
- *Chapter 4* covers dealing with hardware interrupts in-depth; the reader will learn how the kernel works with hardware interrupts, then move onto how one is expected to allocate an IRQ line (covering modern resource-managed APIs), and how to correctly implement the interrupt handler routine. The modern approach of using threaded handlers (and the why of it) is then covered. The reasons for and using both "top half" and "bottom half" interrupt mechanisms (hardirq, tasklet, and softirqs) in code, as well as key information regarding the dos and don'ts of hardware interrupt handling are covered. Measuring interrupt latencies with the modern [e]BPF toolset, as well as with Ftrace, concludes this key chapter
- Common kernel mechanisms – setting up delays, kernel timers, working with kernel threads and kernel workqueues – is the subject matter of *Chapter 5.* Several example kernel modules, including three versions of a 'simple encrypt decrypt' ('sed') example driver, serve to illustrate the concepts learned in code
- The final two chapters of this book deal with the really important topic of *kernel synchronization* (the same material in fact as the last two chapters of the first book).

I think you'll find that both books have a fairly large number of high quality, relevant code examples, all of which are based on the 5.4 LTS kernel.

[LKP : code on GitHub (https://github.com/PacktPublishing/Linux-Kernel-Programming)] [LKP Part 2 : code on GitHub (https://github.com/PacktPublishing/Linux-Kernel-Programming-Part-2)]

Thanks for taking the time to read this post; more, *I really hope you will read and enjoy these books!*

Get ***Linux Kernel Programming, Kaiwan N Billimoria, Packt, Mar 2021 :***

[On Amazon (US) (https://packt.live/399V2uS)] [On Amazon (India) (https://www.amazon.in/Linux-Kernel-Development-Cookbook-programming-ebook/dp/B07RW915K4/ref=sr_1_1?dchild=1&keywords=linux+kernel+kaiwan&qid=1614745572&sr=8-1)] [On Packt (https://www.packtpub.com/product/linux-kernel-programming/9781789953435)]

BOOK **DEVICE-DRIVER** **KERNEL** **LINUX INTERNALS** **LINUX-KERNEL** **LKM** **PROGRAMMING**