Welcome to the SYS2

After the userland comes the kernel land

Gélules

$ker(g \circ f) = f^{-1}(ker(g))$

Userland was fun

Get ready to dive into the kernel

Real Intelligence is still better

Within SYS2, you are free to use AI the way you want. No restrictions.

Even upload and share the SYS2 PDF files in an online AI/LLM is allowed... ;)

OSDEV books

https://wiki.osdev.org/Books

Ask me for the books, I got them all :)

Book

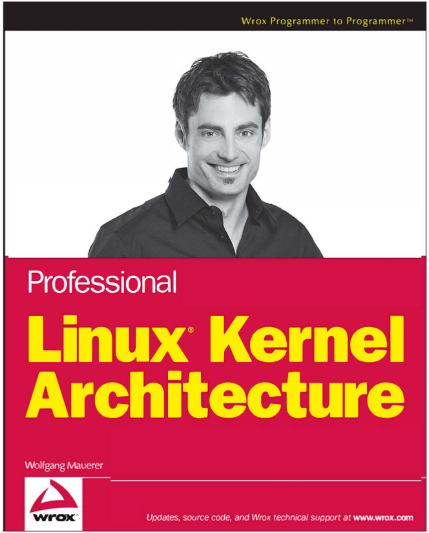

Wrox Programmer to Programmer™
Professional
Linux® Kernel Architecture
Wolfgang Mauerer
wrox
Updates, source code, and Wrox technical support at www.wrox.com

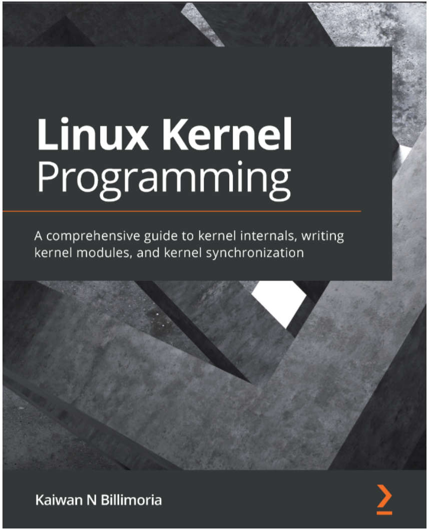
Figure 1: Linux Kernel Architecture

Book

Linux
Kernel Debugging
Leverage proven tools and advanced techniques to effectively debug Linux kernels and kernel modules
Kaiwan N Billimoria

Figure 2: Linux Kernel Debugging

Book



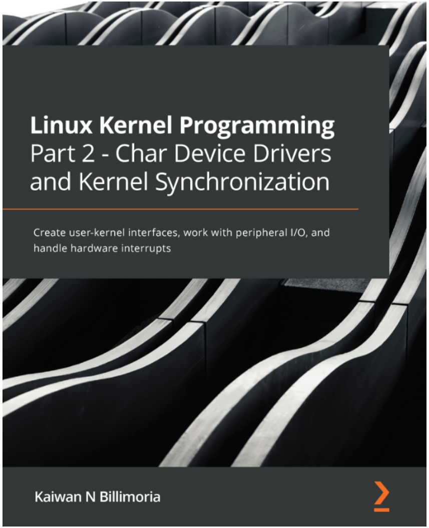
Linux Kernel Programming

A comprehensive guide to kernel internals, writing kernel modules, and kernel synchronization

Kaiwan N Billimoria

Figure 3: Linux Kernel Programming part 1

Book



Linux Kernel Programming
Part 2 - Char Device Drivers and Kernel Synchronization

Create user-kernel interfaces, work with peripheral I/O, and handle hardware interrupts

Kaiwan N Billimoria

Figure 4: Linux Kernel Programming part 2

Userland & kernel land

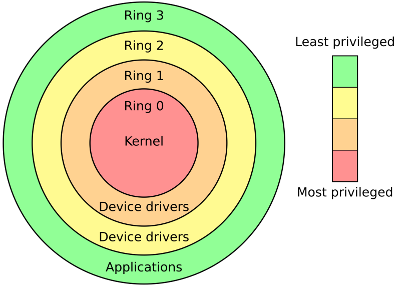


Figure 5: Rings (Wikipedia)

Userland is a memory zone where we execute programs as user

Kernel land is a privileged memory zone where the kernel is loaded and execute its code

Monolithic

If a program in userland crashes, at worst you get a segfault. You patch it and reexecute it. In kernel land, Linux being monolithic, if a module crashes, a **kernel panic** occurs et we need to reboot the computer

It is a critical environment

To call or not to call

```c
/* code */

printf("Hello, world!\n");

/* code */
```

Is there a syscall during the execution of this code?

Modules

Kernel modules are made in C, using the “*liblinux*”

They are loaded in kernel land

TODO

Mainline:

- Dev kernel objects

Bonus:

- Compile the Linux kernel
- Debug the kernel
- Write an init
- Write a syscall
- Make a minimal distro
- Add a syscall to the kernel

I may have not enough time because I want you to work on the project during the class in case you have any questions. Anyway, I will send you files to gain experience on SYS2 even on topics we had no time to work on

Project

Implement and **document** a rootkit

Your new friends

- https://www.kernel.org
- https://docs.kernel.org
- QEMU (also KVM)
- https://elixir.bootlin.com - Not "*official*" but pretty darn good

Do not forget your friends from SYS1 ;)

?

Questions?