第十九讲：I/O 子系统

第 5 节：Linux I/O 子系统

向勇、陈渝

清华大学计算机系

xyong,yuchen@tsinghua.edu.cn

2020 年 5 月 5 日

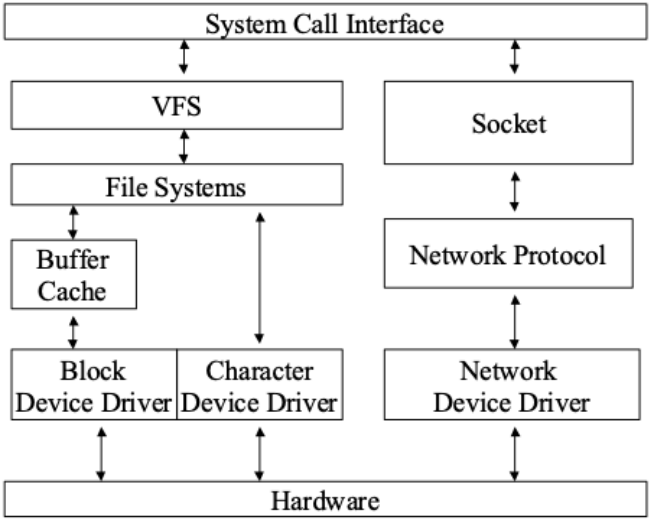
提纲

1. 第 5 节：Linux I/O 子系统
   - Linux I/O Architecture
   - Device Driver
   - Device Driver Programming

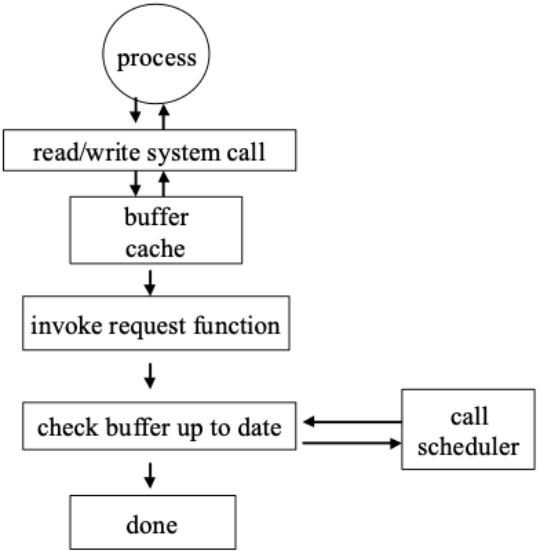
Ref:

- Linux Device Drivers Overview
- Understanding the Linux Kernel

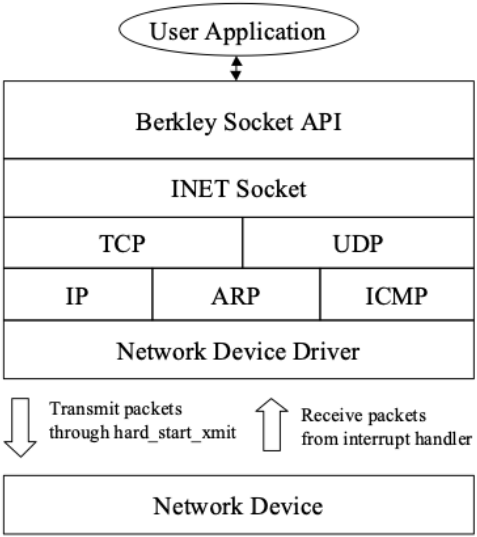
Linux I/O Architecture



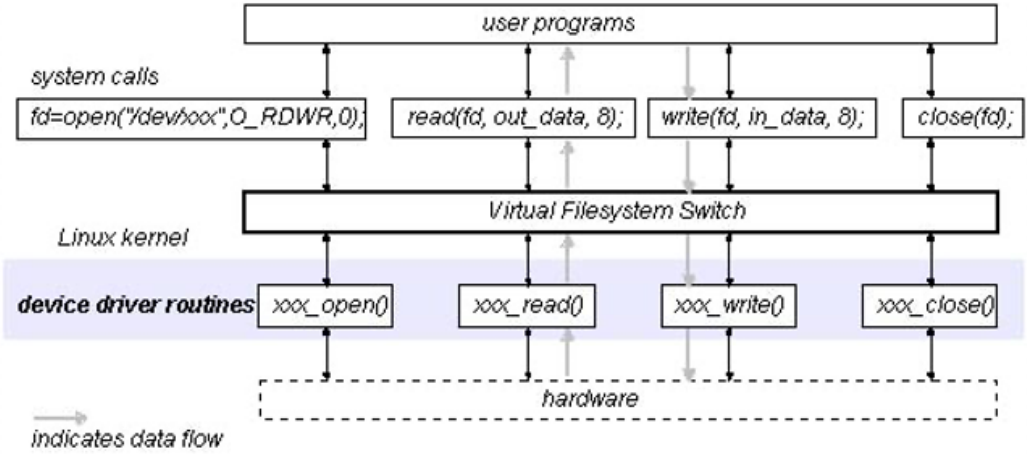
Block Driver



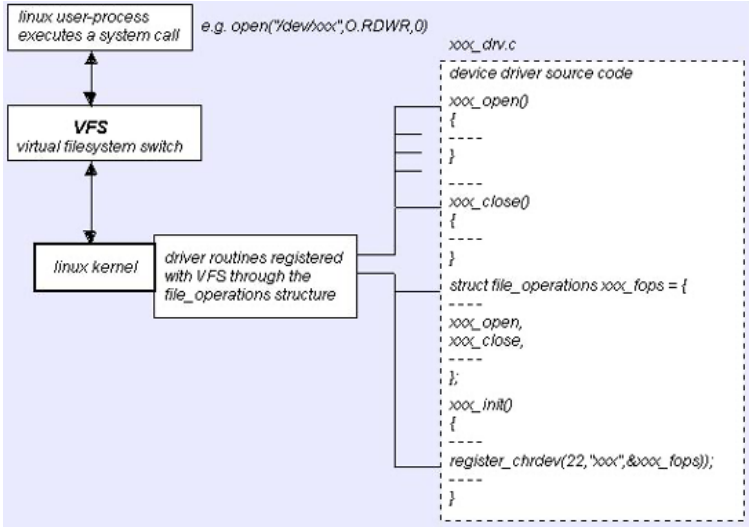
Network Driver



Device Driver interface



Interface between a Device Driver and Linux Kernel



Device Driver Implementation

Assuming that your device name is xxxx

- xxxx_init() initialize the device when OS is booted
- init_module()
- cleanup_module()
- xxxx_open() open a device
- xxxx_read() read from kernel memory
- xxxx_write() write
- xxxx_release() clean-up (close)

Kernel Support Functions

- I/O ports reservations
  - request_region()
- Interrupt Handler Registration
  - request_irq()
- Memory Allocations
  - kmalloc(), vmalloc(), get_free_page()
- Data Transfer between User/Kernel
  - memcpy_fromfs()

API requirements

- Driver functions return positive ints on success, negative ints on failure
- Kernel won't call non-blocking I/O functions if previous request still pending
- Every kernel function that calls kmalloc() should be reentrant
- Common Mistakes
  - Locking, Interrupt time, resource allocation

Locking

- Four kinds of locks in kernel
  - Spin locks and read-write locks
  - Interrupt enable/disable
    - Sometimes combined, e.g., spin_lock_irq
  - Whole kernel lock
- Locks are used all over the place
  - Are they used correctly? consistently?

Interrupt Time

When in_interrupt is true, code cannot

- Access current
- Call the scheduler (may sleep)
- Call kmalloc() (may sleep)
- Copy to/from user-space (may sleep)
- more?

Resource Allocation

- Drivers get and release system resources
  - Memory
  - IRQs
  - module numbers (maybe)
  - space for their code (mod usage count)
- Are the resources handled correctly?
  - Leaks lead to instability – reboot to reclaim