Kernel Communication Channels

Down the rabbit hole...
ECE 373

Prelims

- Questions on homework or reading assignments?
- Questions on class?

System Calls, revisited

- Main communication channel into kernel
- How drivers get work done
- Constrained to passing specific data
  - `read(fd, buf, count), write(fd, buf, len)`
  - `settimeofday(timeval, timezone)`
- Considered "in-band" data
- No channel for "out-of-band" or OOB data

OOB Kernel Channels

- OOB = Out Of Band
  - Not the "normal" way
- Many ways to chat with the kernel
- Allows ways to give data to drivers and the kernel
- Allows ways to get data from drivers and the kernel

ioctl()

`ioctl()`

- System call from userspace to kernel
  - `fd = open(filename, ...)`
  - `ioctl(fd, command, &data)`
  - `close(fd)`
- Handy, quick ways to exchange data
- Full flexibility for what can be exchanged
- Relies on agreement between userspace and kernel
  - Command codes, data structures

loctl's and why we don't (typically) use them

- Very difficult to maintain, no common API
- Userspace can break very easily
- Some still in existence (ethtool, nvme control plane, etc.)
- Windows uses them heavily

- … the Kernel maintainers (usually) "Just Say No ®"

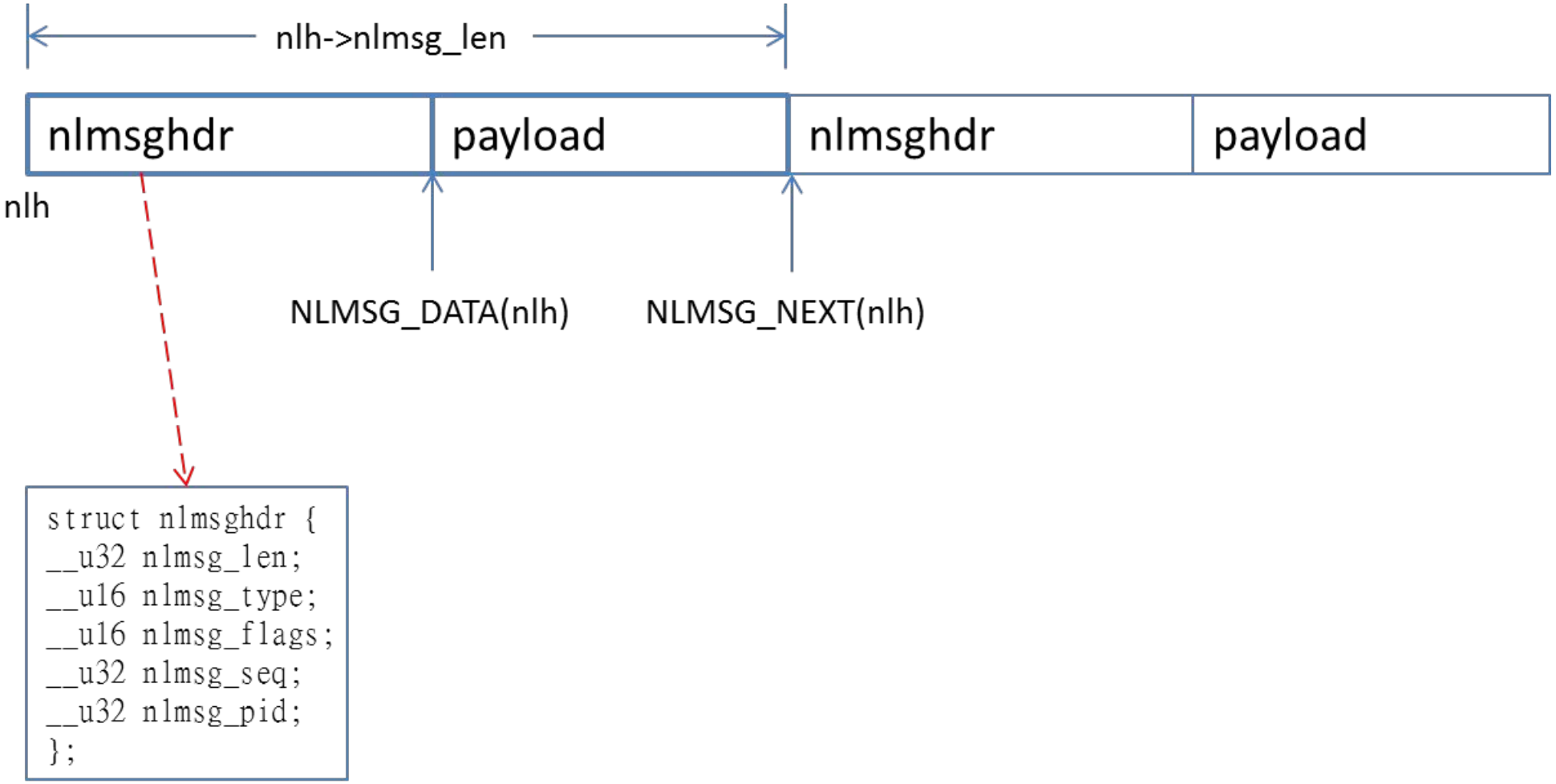
netlink

- Message-based protocol
- Very handy for exchanging information back and forth
- Governed by well-defined structures and APIs
- Popular commands using netlink:
  - iproute2 tools
  - vconfig
  - netstat

Netlink, continued

- Netlink requires packing/unpacking of data
- Self describing data structures
- Not always easy to use
- Message interfaces can be very cryptic
- Which netlink to use?
  - Generic netlink, rtnetlink?

Netlink, visualized



Pseudo-filesystems

- WTH is a pseudo-filesystem???
- Filesystem presented by the kernel
- Look and feel of a real disk-based filesystem
- "Special" files presented
- Managed by `/etc/fstab`
  - On systemd systems, managed by systemd
- Standard UNIX way to handle
- OS interfaces

/dev

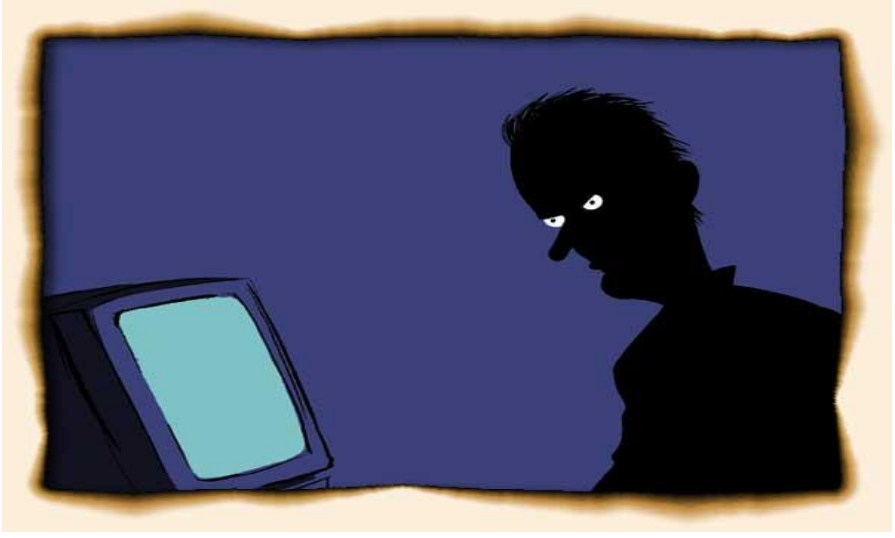
- Puts HW devices into a filesystem
- Legacy filesystem found in many UNIX distributions
- Comprised of major and minor device nodes
- Unique channel for each device
- History of /dev in Linux is twisted and ugly
- The devfs model

/proc

- Puts kernel data into a filesystem
- Exists in many UNIX distributions
- Can be compiled out of Linux, not recommended (not entirely true in 4.x kernels)
- Many programs read contents for information
- Grew way beyond intent in Linux, out of control

/proc lurkers

- Interrupt layout
- Modules currently loaded in-kernel
- All running PID's
- Memory layout and information
- All kernel symbols built-in (important for debugger!)



Sysfs – another one??

- Yet another pseudo-filesystem
- Better hierarchy for device models
- Fits into /dev model for class drivers (more on this later)
- Somewhat object-orientation flavored

More /sysfs

- Complete hierarchy of attached devices
- Large set of symlinks to organize relationships
- Sysfs subsystem can create `/dev` entries correctly
- Contains major and minor dev nodes that links into `/dev`
- Controlled within drivers with a nice API
- Distinct set of rules for use

debugfs

- Enables kernel debugging
- Debugfs framework inside of sysfs
- Usually mounts under `/sys/kernel/debug`
- Useful for exporting debug hooks, buttons and whistles into kernel infrastructure
- No rules for use other than don't build user tools based on it
- Not something maintainers particularly like to use in production drivers...

configfs

- Yet another filesystem…
- Meant to serve as userspace-based object manipulation in the kernel
- Intended to be complimentary to sysfs
- Depends on who you talk to on how to use it…
- Mounted at `/sys/kernel/config`

tmpfs

- Some UNIX systems don‘t have `/tmp` on physical disk
- Tmpfs in systemd-based systems mounted as a RAM disk
- Auto-purged on reboot
- Typically limited to half of physical RAM (can be adjusted)

Udev – AHHHH!!!

- How to manage all pseudo-filesystem attach points?
- Device management lives in userspace
- Consists of daemon and config files
- Abstracts meaningful physical device into device symlinks
- Solves "ordering" problem
- Pain in the butt for developers…
- Now, part of systemd!™

Take a walk in the pseudo-filesystems

- Peek at `/dev`
- Peek at `/proc`
- Peek at udev configuration loveliness
- Peek at sysfs
- Peek at configfs
- Peek at tmpfs
- Peek at fstab
- `hello_kernel` in sysfs

Further Browsing

- Essential Linux Device Drivers, page 103-117
  - more that you'll want to know
- LDD3 chapter 14 – Linux Device Model
  - even more that you'll want to know
- Careful, you can quickly drown, just browse this stuff

Reading Ahead – Char Drivers

- ELDD: Chapter 5
- LDD3: Chapter 3

- Don't try to read all of these pages, you'll hurt yourself!

Upcoming topic...

- Getting inside the drivers
- Character device drivers!
- Yes, time to get dirty in the code
- You won't be the same
- You'll lose sleep
- You'll think about drivers 24/7