Lunatic Driver Development

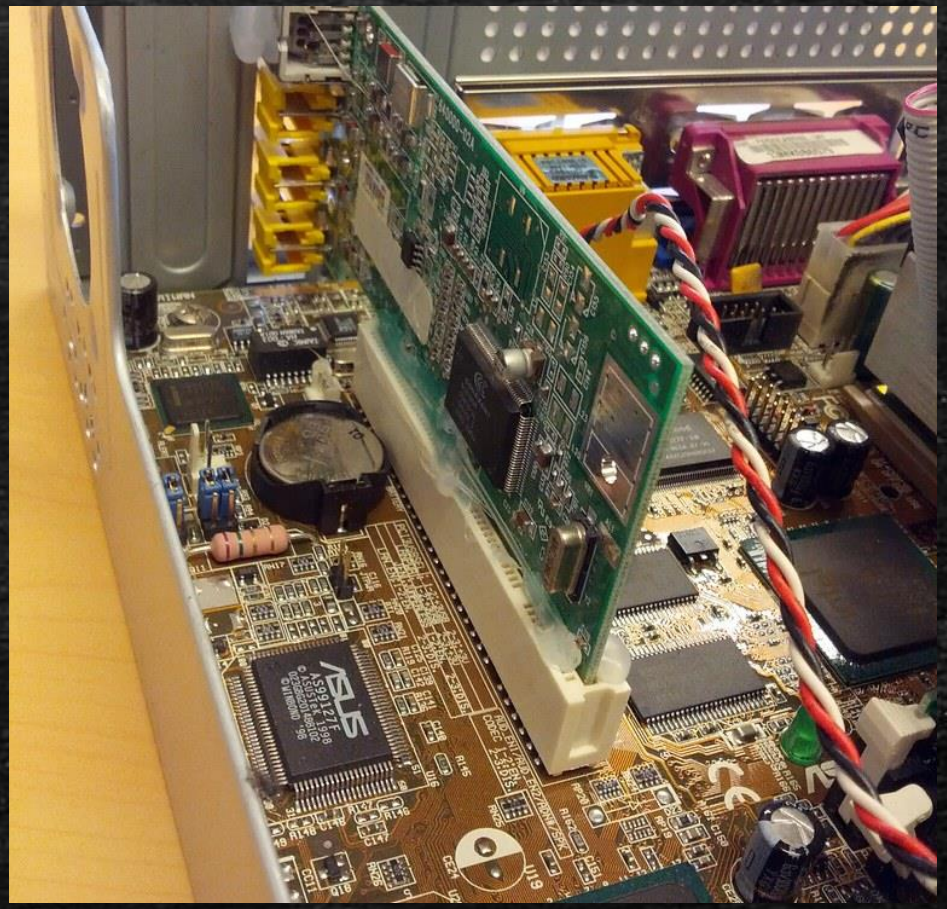
Wer sind diese Fahrer überhaupt, und was haben sie (nicht) mit der Formel 1 zu tun?

Bene (17ostendo) & Finn (17sell)

https://l.mafiasi.de/driver-kbs

Inhalt

- HowTos & Resources
- Geräte, was sind das?
- Babys First Driver
- PCI erklären? (Wie viel?)
- VirtIO (Warum? Was kann das?)
- A more complex driver



HowTos & Resources

- OS Development Wiki: https://wiki.osdev.org/
- VirtIO Spec: https://l.mafiasi.de/virtio
- VGA Text-Mode Driver in Rust: https://os.phil-opp.com/vga-text-mode
- UART Driver in Rust: https://l.mafiasi.de/uart-driver

Babys First Driver

UART Port consisting of Ground, Receive and Transmit pins

Step 1: Spec Reading Time

https://l.mafiasi.de/uart-device-spec

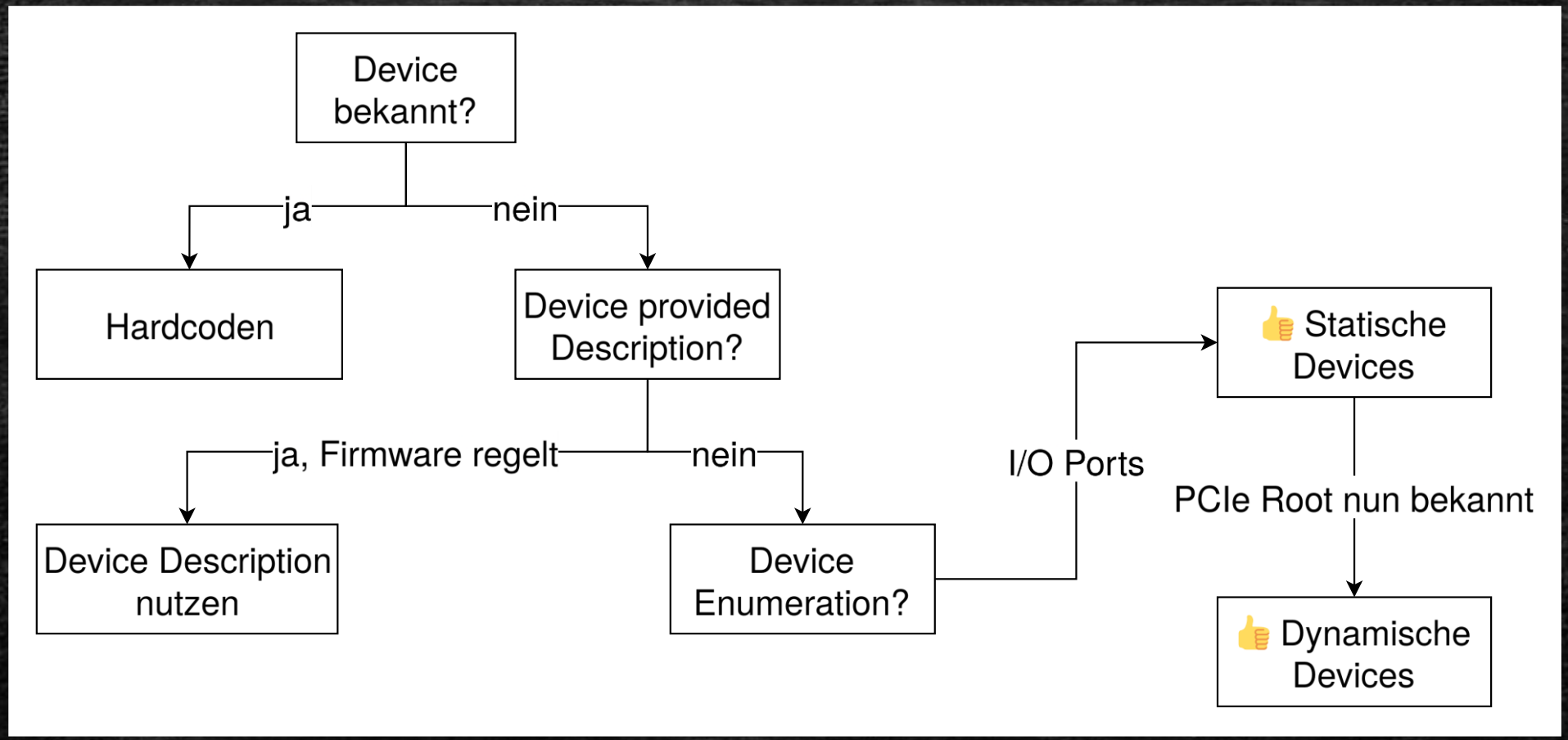
Address & Access type		Register		Bit number 7	6	5	4	3	2	1	0	reset value
General Register Set												
000	R	Receiver Holding Register	RHR	Character Received								00
000	W	Transmitter Holding Register	THR	Character to be Transmitted								00
001	R/W	Interrupt Enable Register	IER	DMA Tx End	DMA Rx End	0	0	Modem Status	Receiver Line Status	THR Empty	Data Ready	00
010	R	Interrupt Status Register	ISR	FIFOs enabled	FIFOs enabled	DMA Tx End	DMA Rx End	Interrupt Identification Code			Interrupt Status	01
010	W	FIFO Control Register	FCR	Receiver's FIFO Trigger Level		0	Enable DMA End	DMA mode	Tx FIFO Reset	Rx FIFO Reset	FIFO enable	00
011	R/W	Line Control Register	LCR	DLAB	Set Break	Force Parity	Even Parity	Parity Enable	Stop Bits	Word Length		00 [1]
100	R/W	Modem Control Register	MCR	0	0	0	Loop back	Out 2 / Int. Enable[2]	Out 1	RTS	DTR	00
101	R	Line Status Register	LSR	FIFO data Error	Transmitter Empty	THR Empty	Break Interrupt	Framing Error	Parity Error	Overrun Error	Data Ready	60
110	R	Modem Status Register	MSR	CD	RI	DSR	CTS	delta CD	trailing edge RI	delta DSR	delta CTS	00 [3]
111	R/W	Scratch Pad Register	SPR	User Data								00
Registers accesible only when DLAB = 1												
000	R/W	Divisor Latch, Least signif. byte	DLL	Baudrate Divisor's Constant Least Significant Byte								01 [4]
001	R/W	Divisor Latch, Most signif. byte	DLM	Baudrate Divisor's Constant Most Significant Byte								01 [4]
101	W	Prescaler Division	PSD	0	0	0	0	Prescaler's Division Factor				00

Babys First Driver, Implementation

```
pub fn print(s: &str) {
    const uart: *mut u8 = 0x1000_0000 as *mut u8;
    for b in s.as_bytes() {
        unsafe { core::ptr::write_volatile(uart, *b); }
    }
}
```

PCIe

Welcher Gerät ist im Büro?



PCIe

- Datenbus für Peripherie
- Prozessor- und Motherboard-unabhängig
- Planar-Devices & Extension Cards

```
$ lspci -t
-[0000:00]-+-00.0
           +-01.0-[01-06]----00.0-[02-06]--+-08.0-[03]--
           |                               +-09.0-[04]--+-00.0
           |                               |            \-00.1
           |                               +-10.0-[05]--+-00.0
           |                               |            \-00.1
           |                               \-11.0-[06]--+-00.0
           |                                            \-00.1
           +-14.0
           +-16.0
           +-16.1
           +-1a.0
           +-1c.0-[07-08]----00.0-[08]----00.0
           +-1d.0
           +-1f.0
           +-1f.2
           +-1f.3
           \-1f.6
```

Basic PCI Interaction

A Device has:

- Address
- Configuration Space
- I/O Space
- Memory Space

PCI Commands are:

- Read/Write Config
- Read/Write IO
- Read/Write Memory
- Special


Actual (Legacy) Hardware
Para-Virtualized Hardware
imgflip.com

VirtIO

VirtIO, Devices

Network Card	Block Device	Console
Entropy Source	Memory Ballooning	IO Memory
SCSI Host	9P Transport	MAC802.11 WLAN

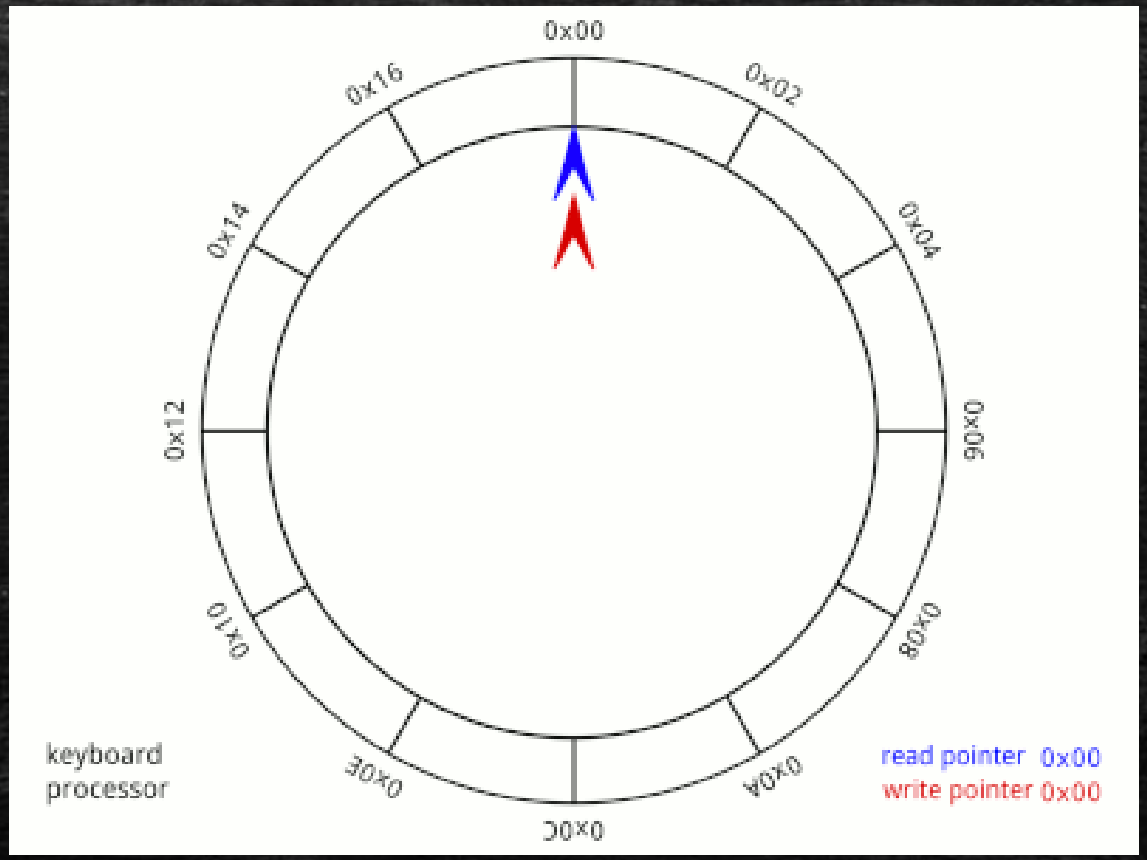
VirtIO

- Method to deliver Messages between Device and Driver
- Uses Shared Memory Buffers
- Buffers are coordinated via 'Virtqueues'


BUFFERS
LIGHTYEAR
BUFFERS EVERYWHERE
imgflip.com

Ring Buffer

- Shared Memory Region
- Read Pointer
- Write Pointer



Ring Buffer, cont.

Requests

Response

Virtqueues

- 3 Areas
  - (Buffer) Descriptor Area
  - Available (Request) Queue
  - Used (Response) Queue
- Messages can be split over multiple buffers
- Buffer is either Readable or Writable by device

VirtIO Protocols

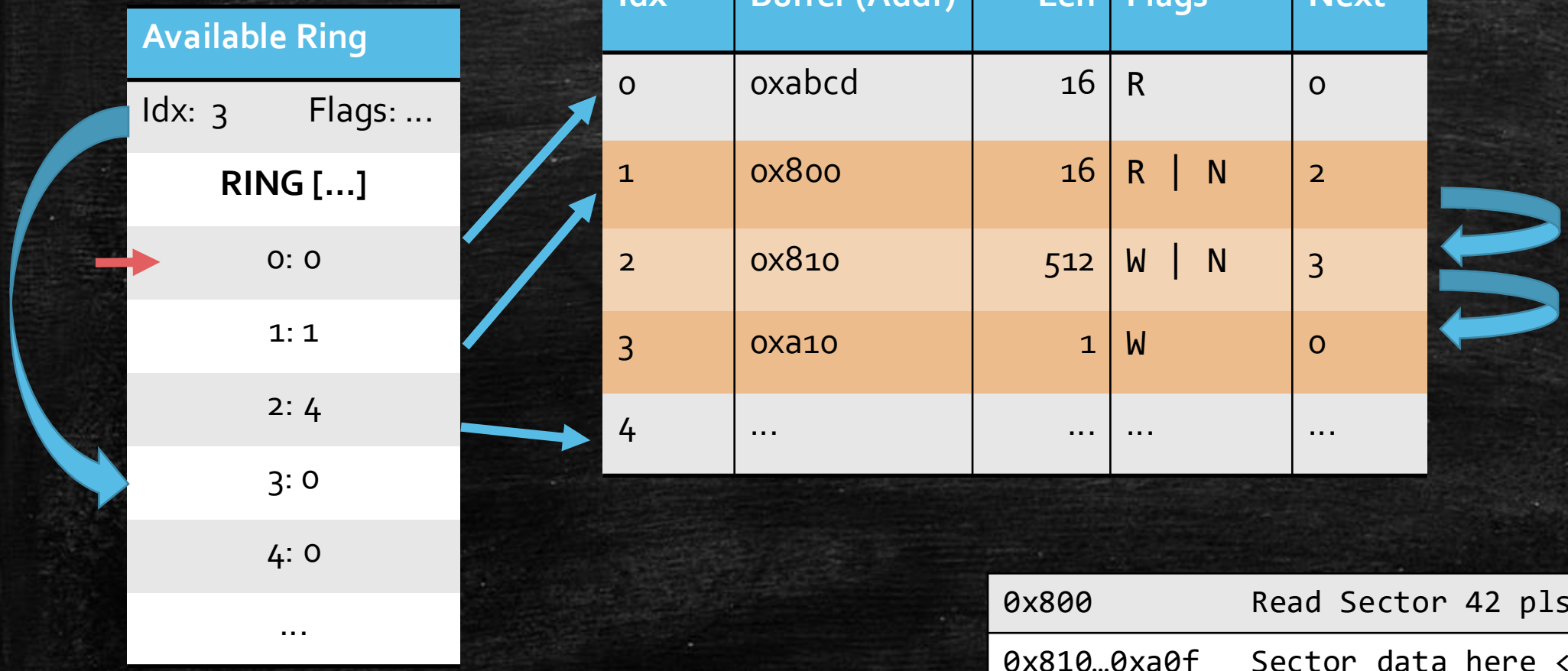
VirtIO Blockdevice

```
struct blk_req {
 le32 type;
 le32 reserved;
 le64 sector;
 u8 data[][512];
 u8 status;
}
```

- 0: BLK_IN
- 1: BLK_OUT
- 4: BLK_FLUSH

- 0: BLK_OK
- 1: BLK_IOERR
- 2: BLK_UNSUPP

VirtIO Descriptor Chaining



Idx	Buffer (Addr)	Len	Flags	Next
0	0xabcd	16	R	0
1	0x800	16	R | N	2
2	0x810	512	W | N	3
3	0xa10	1	W	0
4	…	…	…	…

0x800	Read Sector 42 pls 👉👈
0x810…0xa0f	Sector data here <empty>
0xa10	Status here <empty>


5

Response

Used Ring	
Idx: 4	Flags: …
RING […]	
0: 5	
1: 1	
2: 4	
3: 0	
4: 0	
…	

Idx	Buffer (Addr)	Len	Flags	Next
0	oxabcd	16	R	0
1	0x800	16	R | N	2
2	0x810	512	W | N	3
3	0xa10	1	W	0
4	…	…	…	…
5	…	…	…	…

0x800	Read Sector 42 pls 👉👈
0x810…0xa0f	b"Hello World"
0xa10	0 [Was ok!]

VirtIO Framebuffer

- CMD_GET_DISPLAY_INFO
- CMD_RESOURCE_CREATE_2D
- CMD_RESOURCE_ATTACH_BACKING
- CMD_TRANSFER_TO_HOST_2D
- CMD_RESOURCE_FLUSH

- Create Host Buffer
- Create Client Buffer

- Render to Buffer
- Transfer Client->Host
- Repeat

Waiting for Events

- Incoming Network Packets
- Mouse / Keyboard Events

- Fill Aavail-Queue & Descriptors with 'Give Events' Requests
- Device waits with Respons until Events come in


WE WANT YOU!
TO WRITE DRIVERS!
imgflip.com