

zCore 硬件移植与驱动开发

zCore Summer of Code Day3

王润基

2020.08.05 @ 鹏城实验室

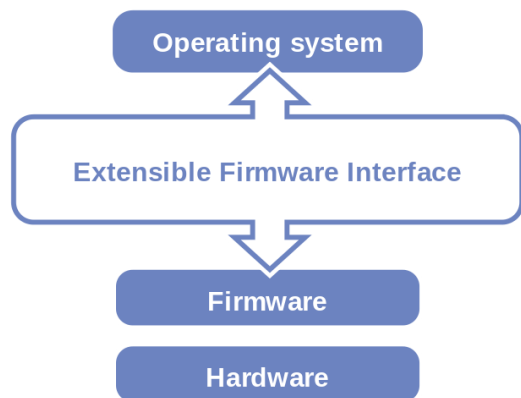
提纲

- x86 底层设施
- 驱动开发方式
- zCore 硬件移植

x86 底层设施

- UEFI 和 Bootloader
- 串口和键盘
- 图形输出：VGA
- 段描述符表：GDT, IDT, TSS
- 中断控制：Local APIC, IO APIC

UEFI -- Unified Extensible Firmware Interface



- UEFI 是继 BIOS 之后的下一代行业标准
- 起源于 x86，正在推广到 ARM 和 RISCV
- 提供了从硬件启动到 OS 之前的执行环境

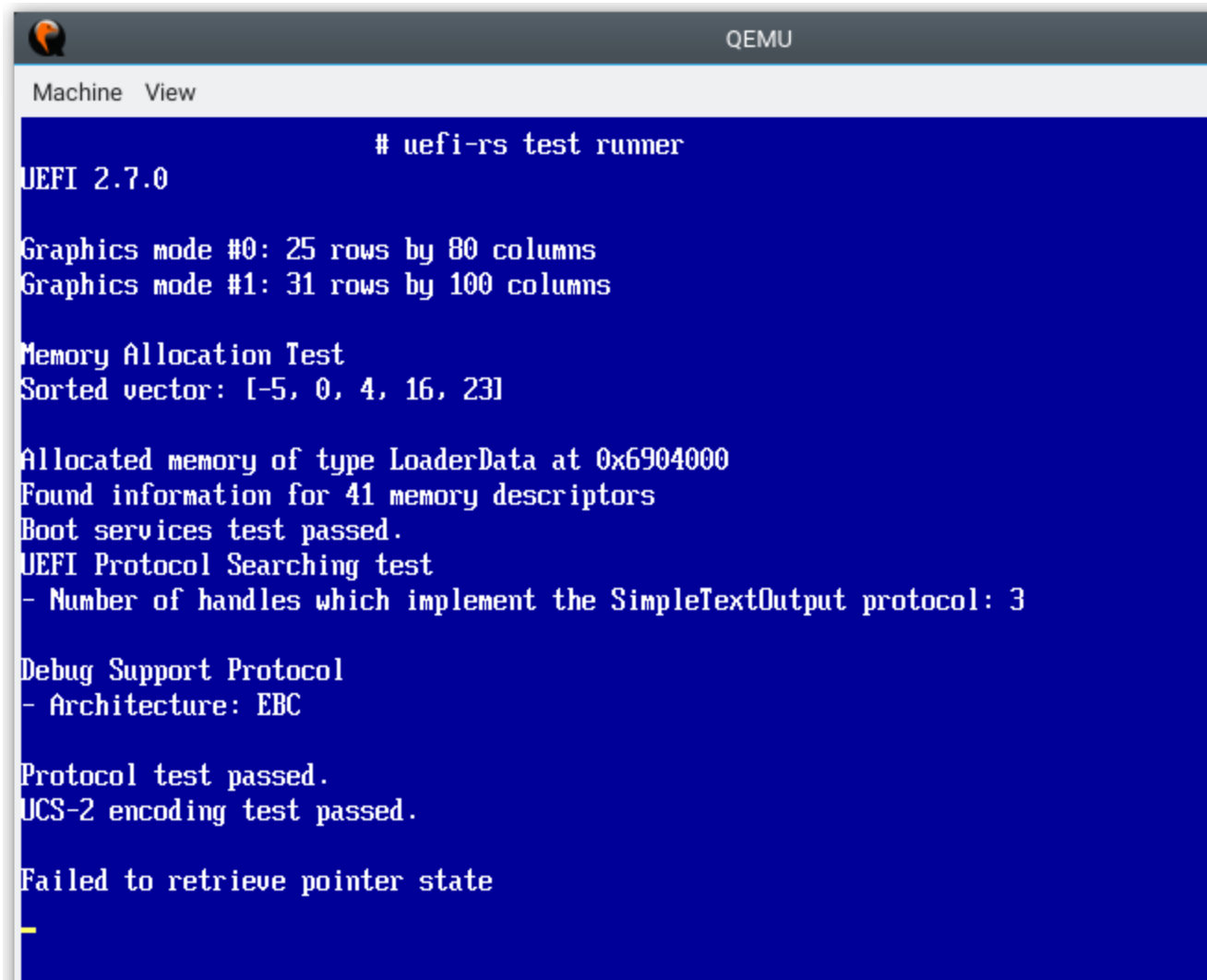
UEFI vs BIOS

	UEFI	BIOS
启动位置	GPT EFI 分区 (FAT)	MBR 扇区
执行环境	64bit Long Mode	16bit Real Mode
编程接口	EFI 函数接口	int 指令
图形设置	支持显卡任意分辨率	VGA 标准

好处：

- 省去了从古老的实模式进入现代长模式的汇编代码
- 丰富的外设驱动支持，快速输入输出
- Rust UEFI 开发环境方便易用

uefi-rs

A screenshot of a QEMU virtual machine window. The window has a title bar with the QEMU logo and the text "QEMU". Below the title bar is a menu bar with "Machine" and "View". The main area is a blue terminal window with white text. The text is the output of the "uefi-rs test runner" program. It shows the version "UEFI 2.7.0", graphics mode information, memory allocation test results, and various protocol tests. The output ends with a yellow cursor line.

```
Machine View

# uefi-rs test runner

UEFI 2.7.0

Graphics mode #0: 25 rows by 80 columns
Graphics mode #1: 31 rows by 100 columns

Memory Allocation Test
Sorted vector: [-5, 0, 4, 16, 23]

Allocated memory of type LoaderData at 0x6904000
Found information for 41 memory descriptors
Boot services test passed.
UEFI Protocol Searching test
- Number of handles which implement the SimpleTextOutput protocol: 3

Debug Support Protocol
- Architecture: EBC

Protocol test passed.
UCS-2 encoding test passed.

Failed to retrieve pointer state
_
```

rBoot: rCore UEFI Bootloader

参考 [rust-osdev/bootloader](https://github.com/rust-osdev/bootloader) 实现的 UEFI 版本

- 从存储设备中加载内核镜像 ELF 文件到内存
- 配置页表，为内核建立虚拟内存映射
- 设置图形输出
- 启动多核
- 收集启动信息传给内核

注：目前正尝试移植到 ARM64

基本输入输出：串口和键盘

- 通过 Port IO 控制（`in/out` 指令）
- 现代 BIOS 将 USB 键盘鼠标 模拟成 PS/2 设备
- rust-osdev: `uart_16550`, `ps2-keyboard`, `ps2-mouse`
- 开发 OS 第一件事：配置串口输入输出

真机环境：图形输出

- rboot 在 UEFI 环境设置好图形输出分辨率
- 向 Framebuffer 写入像素值即可
- 开发库： `rcore-console` + `embedded_graphics`
- Linux：封装为 `/dev/fb0` 设备文件

段描述符表

历史糟粕，x64 已抛弃分段机制。
但出于兼容性考虑，需要进行必要的设置。

GDT：全局描述符表

- 需要建立：内核 CS，内核 SS，用户 CS，用户 SS，TSS

TSS：任务段描述符

- 维护中断处理切换特权级时的栈地址
- IO Port Bitmap：维护开放给用户态的 IO 端口

IDT：中断描述符表

- 维护各中断号的入口地址和权限
- 16 Bytes * 256 项 = 4 KB
- 参考： [Writing an OS in Rust: CPU Exceptions](#)

以上内容已封装到 `trapframe-rs` 库中。

x64 其它事项

`syscall` 指令：快速系统调用

- 相比 `int` 开销更小，需要在 `EFER` 中开启功能
- `int 80` -> `syscall`, `iret` -> `sysret`
- `syscall` 指令破坏 `rcx`, `r11` 寄存器，且不自动换栈

`gsbase` 内核中的 TLS

- x86 线程寄存器：FS, GS, KernelGS
- `swapgs` 指令：用于在中断处理时切换到内核上下文

以上内容已封装到 `trapframe-rs` 库中。

中断控制

Local APIC

- 每个 CPU 核的局部中断处理
- 控制时钟、跨核中断（多核启动）
- 相当于 RISC-V 的 CLINT

IO APIC

- 全局外设中断处理
- 相当于 RISC-V 的 PLIC

开发库: `rcore-os/apic` -> `x86`

驱动开发方式

- 设备发现：PCI + 设备树
- 设备驱动：以 virtio 为例
- 用户态驱动：Zircon DDK 内核对象

设备发现：PCI (x86)

- 配置：PCI Configuration Space (MMIO)
- 编号：Bus : Device : Function
- BAR (Base Address Registers) : MMIO / PIO 地址空间
- MSI (Message Signaled Interrupts) : 外部中断号

开发库：pci-rs

更多资料：[知乎：深入PCI与PCIe之二：软件篇](#)

设备发现：设备树 (ARM + RISC-V ...)

- 文本 `.dts` -> 二进制 `.dtb`
- 写死在板子固件中，boot 时传递地址

```
uart@10000000 {  
    interrupts = <0x0000000a>;  
    interrupt-parent = <0x00000003>;  
    clock-frequency = <0x00384000>;  
    reg = <0x00000000 0x10000000 0x00000000 0x00000100>;  
    compatible = "ns16550a";  
};
```

开发库： `device_tree-rs`

设备驱动

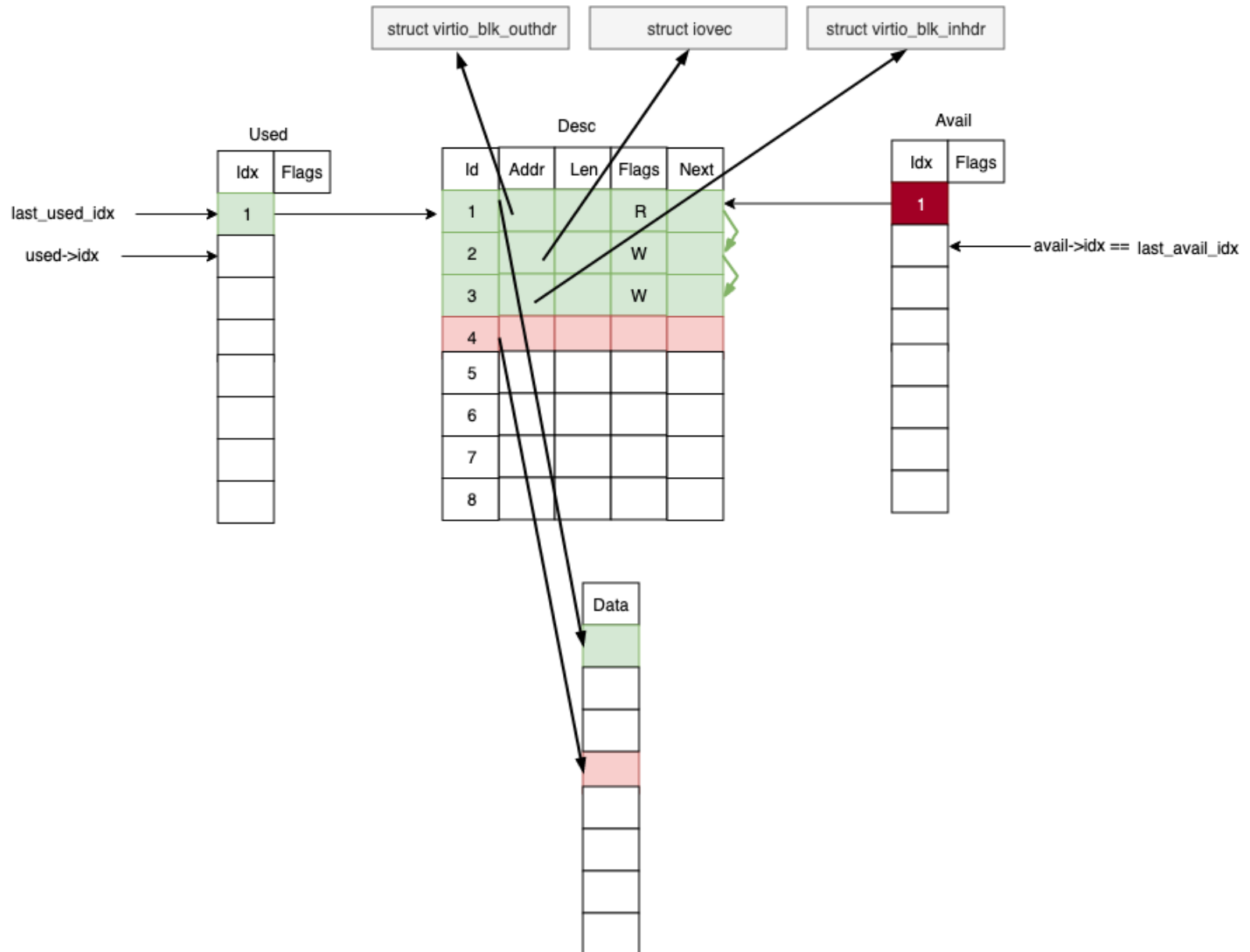
- 配置：MMIO
 - `volatile` 访存
 - `XxxDriver::new(mmio_area: &mut [u8])`
- DMA (Direct Memory Access)
 - 分配连续物理内存
 - 配置虚拟内存映射，禁用缓存
- 中断
 - 将驱动中断处理函数注册到 OS 中
 - 记得 ACK，结束中断信号

设备驱动： virtio

- virtio-queue: 环形队列，核心组件
- virtio-blk: 块设备（磁盘）
- virtio-net: 网络设备（网卡）
- virtio-gpu: 图形设备（GPU）
- virtio-input: 输入设备（鼠标键盘）

开发库： `virtio-drivers` （未来尝试 async? ）

VirtIO Queue



Zircon 驱动相关内核对象

- Resource：表示驱动允许使用的物理/IO地址空间
- Interrupt：外部中断，可以绑定到 Port
- Bus Transaction Initiator：管理 DMA
- Pinned Memory Token：固定物理页面
- PCI：控制 PCI 总线
- IOMMU, IOPorts, MSI：

目前 PCI 的实现照搬 Zircon 的 C++ 源码，
后续计划分离到 pci-rs 库中。

小结

- 「计算机软硬件接口」
- RTFM：从海量文档中快速检索信息的能力
- 善用 QEMU 了解硬件的具体行为
- 软件生态是慢慢积累出来的

zCore 硬件移植

实现 HAL 接口

谢谢大家