

跨操作系统的音频驱动模块设计与实现

答辩展示

穆新宇

北京理工大学计算机学院

指导教师：陆慧梅



- ① 课题概况
- ② 实现
- ③ 测试
- ④ 总结与展望

① 课题概况

背景与动机
创新点

② 实现

③ 测试

④ 总结与展望

① 课题概况

背景与动机

创新点

② 实现

③ 测试

④ 总结与展望

背景与动机

课题背景

在云计算时代，操作系统作为 Guest OS 运行在虚拟化框架上提供服务的情景非常常见，要使 Guest OS 能正确使用虚拟化框架提供的虚拟设备，就需要为这些虚拟设备编写驱动程序，例如，Linux 源码中就有很多虚拟设备驱动程序。

背景与动机

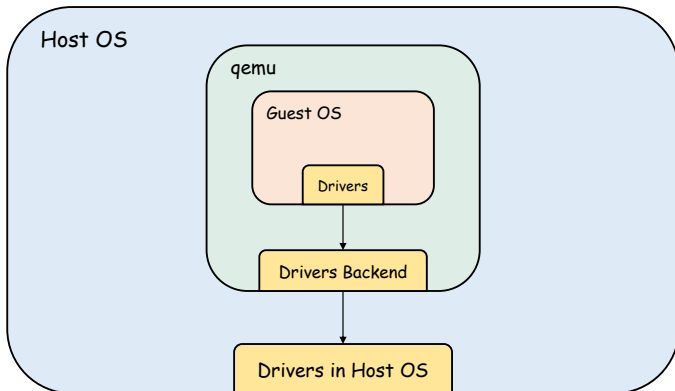


图 1: 在 qemu 上运行客户机时的驱动程序层级

背景与动机

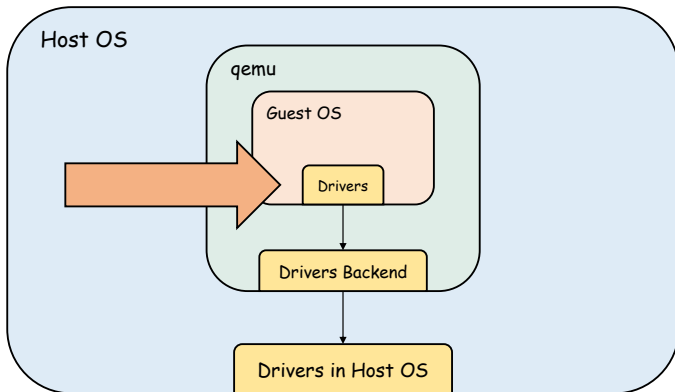


图 2: 客户机上的虚拟驱动程序

背景与动机

动机

每一个作为 Guest OS 的操作系统都要实现一遍 virtio 标准规定的驱动程序，对于一个新出现的操作系统，

- 很难直接利用已有的虚拟驱动程序实现
- 难以移植 Linux 源码中的实现

为此，一个跨操作系统的 virtio 驱动程序框架是必要的，这将极大减轻开发者的“重复造轮子”工作。

virtio-drivers

virtio-drivers 是使用 Rust 编写的跨操作系统的虚拟驱动程序框架，它参考了 Linux 源码中虚拟队列的实现，目前支持 block、net、input、gpu 等虚拟设备。

背景与动机

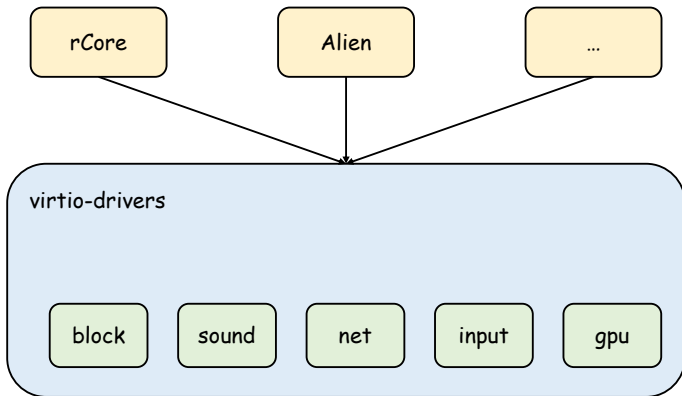


图 3: 使用 virtio-drivers 虚拟驱动程序框架

① 课题概况

背景与动机
创新点

② 实现

③ 测试

④ 总结与展望

创新点

本项目在原有 virtio-drivers 框架基础上，

- 使用 Rust 语言实现了跨操作系统的音频驱动程序
- 在裸机和 Alien 操作系统上进行了正确性验证

① 课题概况

② 实现

整体架构

PCM 帧的传输

③ 测试

④ 总结与展望

① 课题概况

② 实现

整体架构

PCM 帧的传输

③ 测试

④ 总结与展望

整体架构

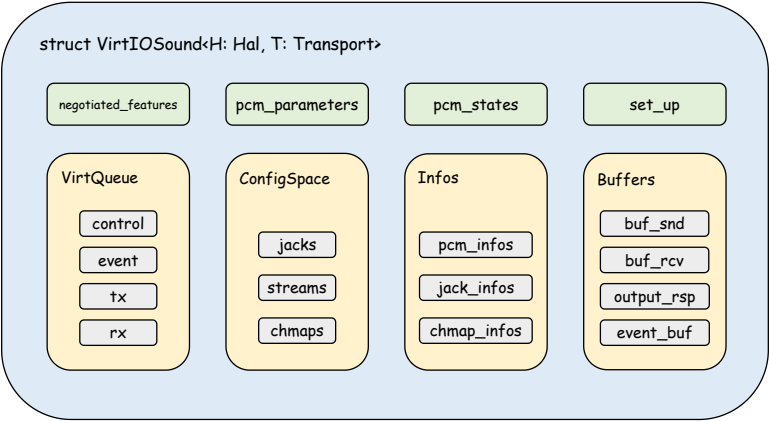


图 4: 驱动程序整体架构

整体架构

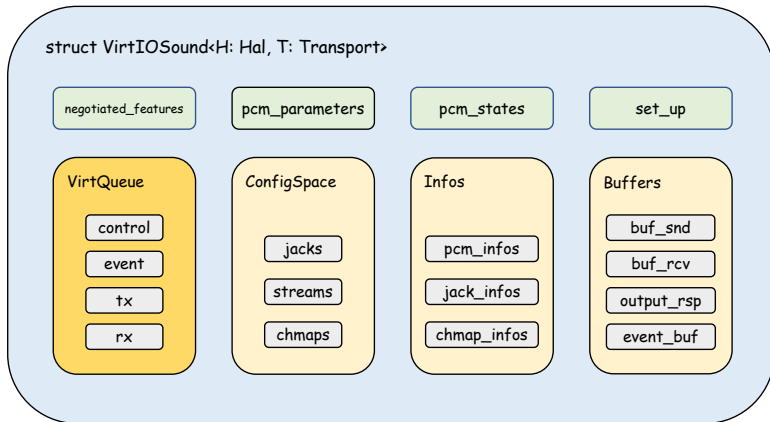
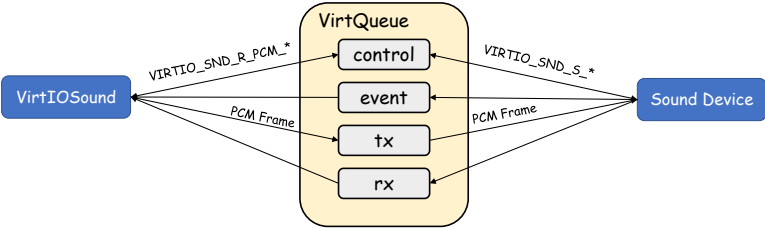


图 5: 虚拟队列

整体架构

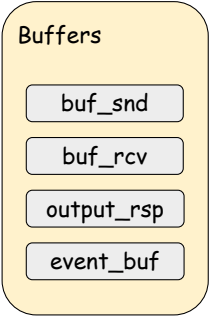
虚拟队列



- control_queue 用于从驱动程序向设备发送控制消息
- event_queue 用于驱动程序接受来自设备的通知
- tx_queue 用于向输出流发送 PCM 帧
- rx_queue 用于从输入流接收 PCM 帧

整体架构

Buffers



- buf_snd 存储驱动程序通过 control_queue 向设备发送的请求
- buf_rcv 存储设备通过 control_queue 向驱动程序发送的响应
- output_rsp 存储驱动程序每传输一个 PCM 帧，设备的响应
- event_buf 存储来自设备的通知

① 课题概况

② 实现

整体架构

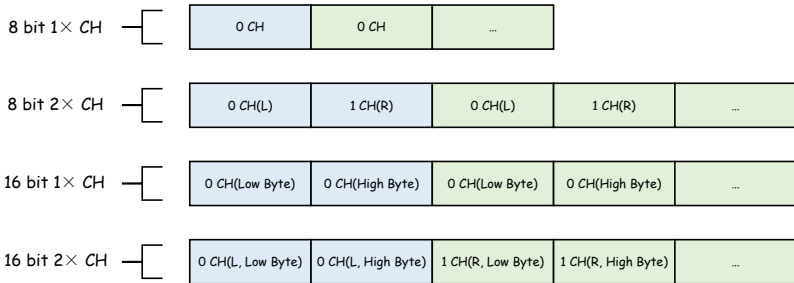
PCM 帧的传输

③ 测试

④ 总结与展望

PCM 帧的传输

PCM 数据结构



PCM 帧的传输

PCM 生命周期

PCM 生命周期有 7 个阶段，这 7 个阶段如下：

- ① **SET PARAMETERS**：设置某 PCM 流的参数
- ② **PREPARE**：PCM 流为传输做准备（资源分配等）
- ③ 仅输出：驱动程序在预缓冲时传输数据
- ④ **START**：设备开始播放/接收一个流
- ⑤ 驱动程序向流传输数据，或从流中接收数据
- ⑥ **STOP**：停止某 PCM 流
- ⑦ **RELEASE**：释放某 PCM 流

设备每使用一个缓冲区，就会将该缓冲区对应的 token 从 tx_queue 中弹出，然后从 tx_queue 中选择一下缓冲区进行播放。因此，为了使音频播放不间断，tx_queue 中至少有两个缓冲区，如图6所示。

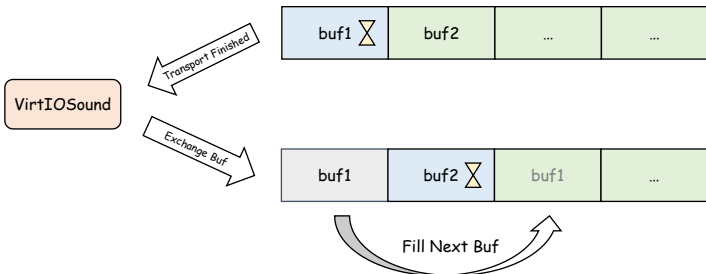


图 6: 双缓冲区交替

- ① 课题概况
- ② 实现
- ③ 测试
- ④ 总结与展望

裸机

编写裸机测试程序，按照 PCM 生命周期的 7 个步骤配置 PCM 流，能够正确播放音乐：

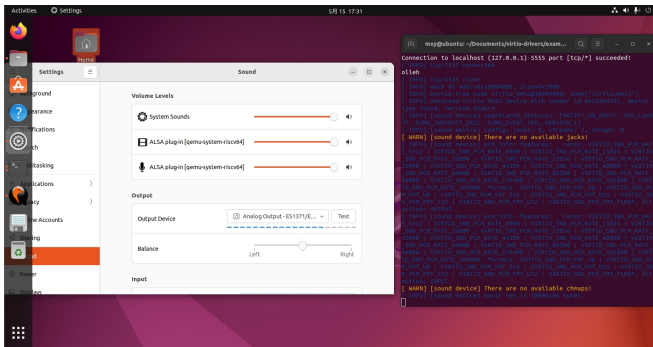


图 7: 裸机测试下正确播放了音乐

- ① 课题概况
- ② 实现
- ③ 测试
- ④ 总结与展望

总结与展望

总结

总的来说，本项目利用 virtio-drivers 框架，按照 virtio v1.2 标准，实现了跨操作系统的音频驱动程序，并分别在裸机和 Alien 操作系统中验证了其正确性，为开源操作系统的发展贡献了自己的力量。

展望

本项目的最终实现虽然保证了正确性，也具有一定的鲁棒性，但是仍然存在不足之处，主要体现在：

- ① 采用程序控制的方式，降低了 CPU 的效率
 - ② 实现了 virtio v1.2 标准，最新的标准是 virtio v1.3
 - ③ 目前只有正确性验证，还没有找到性能测试的基准和指标
- 这些缺点都是未来的工作中需要改进的地方。

谢谢！