R-FreeRTOS 总结报告

李东阳 周宜晖 陈云逸

项目目标

FreeRTOS 作为一种轻量化的实时操作系统，具有源码公开、可移植、可裁减、调度策略灵活的特点，在实际项目中应用广泛。实时操作系统对于安全性有着较高的要求，而 C 语言无法保证较好的内存安全性。为此，我们希望使用具有内存安全性的 Rust 语言重写 FreeRTOS，同时尽可能保证接口和功能与标准实现保持一致。同时，为了有效利用 FreeRTOS 丰富的生态和应用，我们希望 R-FreeRTOS 尽可能地适配标准 FreeRTOS 实现所使用的用户程序和测例。

全志 D1-H 哪吒开发板是全球首款支持 RISC-V64 指令集并支持 Linux 系统的可量产开发板，集成阿里平头哥 C906 CPU，1GHz 主频，支持标准 Linux 内核，支持 2G DDR3、258MB spi-nand、WiFi/蓝牙连接，具有丰富的音视频接口和强大的音视频编解码能力，可以满足日常科研教学、产品项目预研、开发爱好者 DIY 的需求。因此，我们还希望将 R-FreeRTOS 进一步移植到 Qemu、D1-H 等平台上以提高其易用性。

项目设计

项目 Rust 代码包含四个模块：kernel,portable,ffi,test。

kernel 模块为 FreeRTOS 内核，包含了 FreeRTOS 主要功能的实现。模块内部的组织与 FreeRTOS 标准实现类似，实现的功能包括：双向链表的维护；任务调度，包括任务创建、删除、延迟、挂起与恢复、调度器挂起与恢复等；消息队列与信号量；基础事件组操作。除此之外，还增加了 rust 所需的内存分配器等必要组件。

portable 模块对应标准实现中的硬件相关代码。模块内包含与硬件直接相关的时钟中断设置、调度器启动流程、硬件相关宏定义、串口交互等组件。对于移植到 D1-H 开发板的版本，还提供了系统在裸机环境下运行于开发板所必需的部分硬件初始化的相关功能。

ffi 模块用于与 C 代码进行交互，提供了 C 可用的部分接口。在模块中为任务调度、队列、信号量三个功能部件的主要功能函数实现了 C 接口，在 C 中调用相应接口可以实现等价的功能。在模块内同时也使用 Rust 重新实现了 C 语言内可能使用到的断言、输出、内存分配等基本操作。模块内包含 FreeRTOS 原版实现提供的 Demo 程序，使用已实现的 C 接口可以正常运行。

test 模块包含 FreeRTOS 原版实现提供的测例文件。所有测例均以 C 语言实现，测例中与测试框架耦合的代码部分使用 Rust 和 C 实现替换。完成迁移后的测例在 C 代码中逐个运行检验。

D1 硬件部分加上 bare 模块作为硬件初始化代码部分。

bare 模块由 C 代码实现，参考借鉴了论坛上能在 D1 上裸跑的 C 程序，实现了硬件初始化和

参数设置等功能，并引导跳转到 RFreeRTOS 的内核代码部分。

项目成果

R-FreeROTS 实现了原有的任务调度、双向链表、队列、信号量、事件组等核心功能。

R-FreeRTOS 的队列、信号量等核心功能经过了 test 模块中测例文件的测试。由于 ffi 中提供的接口有限，模块中仅包含了全部 42 个测试文件中与队列相关的 14 个。在这 14 个测例文件中，已实现的 140 个测例中有 130 个能够正确运行，仅有 10 个测试失败，另有 78 个测例由于相关功能或测试框架支持未实现而弃用。

R-FreeRTOS 的不同版本可以分别运行在 Qemu-riscv32、Qemu-riscv64、D1-H 哪吒开发板上。其中，在 Qemu 上运行的版本可以仅通过构建时更改编译选项完成切换；在开发板上运行的版本仅有 35K 的文件大小，并可以稳定使用 1000Hz 及更低频的时钟。

此外，我们为 R-FreeRTOS 的模块与功能函数编写了功能文档，并在工程中加入了一些重要功能的使用示例。对于一些与标准实现有区别的函数，我们也在代码中进行了标注。

下一步规划

- 继续完善次要功能和细节的实现，补充主要功能的 C 接口与测试。
- 完善系统的报错信息，使之与 FreeRTOS 标准实现的行为保持更好的一致。
- 优化代码以更好运用 Rust 特性，并补充安全性测试。
- 增加对更多硬件的支持，同时简化不同硬件间的切换方式。
- 通过 ffi 对 FreeRTOS 实际应用程序进行支持，验证 R-FreeRTOS 的可用性。