ISCAS NIST

循序渐进，学习开发一个 RISC-V 上的操作系统

第 6 章 RVOS 介绍

汪辰

2021/4

本章内容

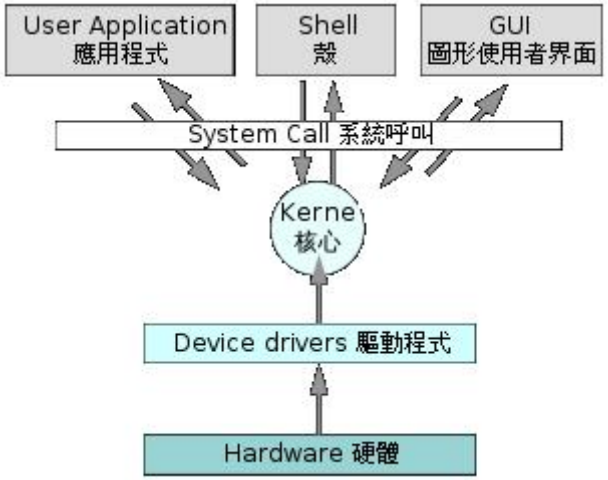
- 操作系统的定义
- 操作系统的分类
- 典型的 RTOS 介绍
- 课程项目简介

参考

- 【参考 1】: The RISC-V Instruction Set Manual, Volume II: Privileged Architecture，Document Version 20190608-Priv-MSU-Ratified

操作系统的定义

- **操作系统（英语：Operating System，缩写：OS）是一组系统软件程序：**
  - 主管并控制计算机操作、运用和运行硬件、软件资源
  - 提供公共服务来组织用户交互。



图片来自：https://zh.wikipedia.org/wiki/操作系统

- **操作系统有广义和狭义之分。**
  - 狭义：内核
  - 广义：发行包 = 内核 + 一组软件。

操作系统的分类

分类	特点	应用场景	RISC-V ISA 对其支持
裸机系统（Bare Metal）	非常小，没有明显的分层设计，没有通用性。通常为单任务+中断处理	微型控制器，简单外设，简单实时任务。	简单的 Machine 模式支持。
实时操作系统（Real-Time Operating Systems）	中等规模，支持多任务，具备一定的通用性，和通用性相比更强调实时性。	比较复杂的多任务和实时场景，丰富的外设。	Machine + User； 或许需要支持物理内存保护（Physical Memory Protection，PMP）。
高级操作系统（Rich Operating Systems）	大型规模，强调用户体验或者复杂通用性。	智能手持设备，PC工作站，云计算服务器	Machine + Supervisor + User，需要支持虚拟内存机制。

Number of levels	Supported Modes	Intended Usage
1	M	Simple embedded systems
2	M, U	Secure embedded systems
3	M, S, U	Systems running Unix-like operating systems

【参考 1】Table 1.2: Supported combinations of privilege modes.

典型的 RTOS 介绍

FreeRTOS（https://www.freertos.org/）是一个很流行的应用在嵌入式设备上的实时操作系统内核。诞生于 2003 年。采用 MIT 许可证发布。

- 设计小巧，整个核心代码只有 3 到 4 个 C 文件
- 可读性强，易维护，大部分的代码都是 C 语言编写，很少的部分采用汇编语言。
- 支持优先级多线程（threads）、互斥锁（mutex）、信号量（semaphore）和软件计时器（software timer），支持低功耗处理以及一定程度的内存保护。
- 支持多种平台架构，包括 ARM，x86，RISC-V 等
- 已经被移植到多款微处理器上。

典型的 RTOS 介绍

RT-Thread（https://www.rt-thread.org/）“是一个集实时操作系统（RTOS）内核、中间件组件和开发者社区于一体的技术平台，..... 也是一个组件完整丰富、高度可伸缩、简易开发、超低功耗、高安全性的物联网操作系统”。诞生于 2006 年。采用 Apache 2.0 许可证发布。

- 面向对象的实时内核；
- 8，32 或 256 个优先级的多线程调度。对于同优先级线程使用时间片轮转调度法；
- 提供信号量，也提供互斥信号量以防止优先级反转；
- 支持其他高效通信方式，比如邮箱、消息队列和事件标志；
- 支持静态内存分配方法，也支持线程安全的动态内存管理；
- 对高层应用提供设备框架。
- 支持多种平台架构，包括 ARM, MIPS, X86, Xtensa, C-Sky, RISC-V 等
- 几乎支持市场上所有主流的 MCU 和 Wi-Fi 芯片。

课程项目简介

ISCAS NIST

RVOS

RVOS（https://github.com/plctlab/riscv-operating-system-mooc）是一个用于教学演示的操作系统内核。诞生于 2021 年。采用 BSD 2-Clause 许可证发布。

- 设计小巧，整个核心有效代码 ~ 1000 行；
- 可读性强，易维护，绝大部分代码为 C 语言，很少部分采用汇编；
- 演示了简单的内存分配管理实现；
- 演示了可抢占多线程调度实现，线程调度采用轮转调度法；
- 演示了简单的任务互斥实现；
- 演示了软件定时器实现；
- 演示了系统调用实现（M + U 模式）；
- 支持 RV32；
- 支持 QEMU-virt 平台。

谢 谢

欢迎交流合作