Collaborators: 12110817张展玮 12110813刘圣鼎 12110714谢嘉楠

Topic selection

proj89-FreeRTOS-rust

Basic idea

This project aims to redesign and implement the widely used real-time operating system kernel, FreeRTOS, using the Rust programming language. The original FreeRTOS is written in C, but this project will use Rust, a modern system programming language with stronger expressive capabilities and richer semantics. The use of Rust will require the team to explicitly write certain rules and designs into the code, which will encourage thoughtful consideration of the project's implementation. The project seeks to maintain the key features of FreeRTOS while leveraging Rust's strengths to improve its overall design and implementation.

Plan

Week 1:

- Understand the project background and goals, study the structure and design of Rust-FreeRTOS.
- Study the structures, variables, and functions in FreeRTOS.
- Familiarize oneself with Cargo feature's kernel trimming functionality and understand the handling of global variables.

Week 2:

- Design the overall architecture of the project, including modular design and task control function design.
- Begin to tackle implementation challenges, such as how to handle basic data structures like linked lists and how to add tasks to the ready list.

Week 3:

- Begin implementing task API functions, including task creation, deletion, and suspension.
- Design and implement task delay functions, adding tasks to the delayed queue.
- Begin preliminary testing of implemented functions.

Week 4:

- Begin designing and implementing Queue and Semaphore, including data structures and implementation methods.
- Begin performance testing.

Week 5:

- Summarize and evaluate the entire project, including functionality correctness, performance testing results, and issues that remain.
- Summarize project experiences and lessons learned, propose suggestions for improvement and optimization.
- Prepare final report, providing detailed explanations and introductions to the project.