SCHOOL OF PHYSICS,
ENGINEERING AND TECHNOLOGY

ELE00138M
Systems Programming for ARM Assessment

Y3884541

January 8, 2024

1 Executive Summary

This report outlines several significant enhancements to the operation of the DocetOS embedded operating system, aimed at expanding the toolset available to users while maintaining robustness and compatibility with low-performance systems. The focus of these modifications is the implementation of fixed-priority scheduling, mutual exclusivity through the integration of a re-entrant mutex, and task management systems such as sleeping and inter-task communications.

Key Modifications:

1. Fixed-Priority Scheduling
   - Introduction of a fixed-priority scheduling algorithm to prioritise tasks based on predetermined priorities
   - Enhances control over task execution
2. Task Sleeping
   - Integration of a sleep orientated wait list to pause the execution of tasks for set periods
   - Expands functionality of tasks
3. Re-entrant Mutex
   - Introduction of a re-entrant mutex to facilitate exclusive access to a shared resource
   - Ensures consistency in data access and prevents race conditions
4. Mutex Priority Inheritance
   - Integration of priority inheritance of tasks owning a mutex with a higher priority task in the mutex wait list
   - Prevents priority inversion issues and streamlines the execution of critical tasks
5. Wait and Notify System
   - Overhaul of the system managing wait and notification of tasks by splitting the centralised wait list into individual sorted wait-lists per blocking item.
   - Enhances speed of task verification and notification
6. Memory Pool
   - Introduction of a memory pool allowing the reservation of memory for use during system run time
   - Consolidates memory management and gives the ability to define maximum system OS memory allocation at runtime
7. Task Communication
   - Integration of a queue-based task communication system, utilising a memory pool to allow the transfer of data across tasks
   - Introduces cooperation between tasks towards a common goal

Benefits:

1. Improved Determinism
2. Enhanced Reliability
3. Optimised Resource Utilisation
4. Compatibility with Low-Performance Systems
5. Memory footprint clarity
6. System Synchronisation

These modifications to DocetOS are a significant step towards meeting the current demands required of real-time embedded systems and provide an opportunity to investigate known techniques used in the professional industry. This report provides a detailed account of the modifications, their rationale, and the anticipated benefits, resulting in a product that can be modified for use in future applications.

Contents

2 Introduction

2.1 DocetOS

DocetOS is a simple embedded system operating system created with the intention of providing a basic skeleton framework to teach the basics of operating system operation. It consists of the routines to initialise task control blocks, and a context switcher to switch between tasks each system tick. Thus, the system at this point is vulnerable to bugs and limitations that necessitate changes to OS functionality to equip it for real-world implementation.
The current state of DocetOS, allows multiple tasks to run concurrently to achieve their goals which works completely fine for a small number of tasks. But with each new task added to the system, the runtime of all other tasks slow down proportionally, and access to shared resources becomes more complex and vulnerable to race conditions. By implementing functionality that provides more in-depth control of tasks, we can modify their behaviour within the scheduler and provide support for the concurrent execution of many additional tasks with minimal impact on performance and risk of unexpected race conditions, removing current limitations.

2.2 Objectives of Modifications

Throughout this report we will increase the functionality of DocetOS by completing the follow objectives:

- Efficient task manipulation capabilities through the implementation of a fixed-priority scheduler with extensive task waiting routines to move tasks to and from custom wait lists.
- Mutual exclusion through the implementation of a re-entrant mutex with task priority inheritance functionality
- Memory management and inter-task communication using a memory pool, utilising memory reserved for OS usage within the embedded system.

2.3 Structure of Report

The subsequent sections of this report provide information on each modification, including the rationale of the modification, an exploration and justification of the design considerations taken during the design process, an overview of the modifications final implemented design and functionality, and if required, information related to the safe usage of the implemented design related to mutual exclusivity that needs to be kept in mind if further modification was to take place in the future.

3 Conclusion