**NANYANG TECHNOLOGICAL UNIVERSITY**

SCE13-0026

# Micrium $\mu$C/OS-III++

Submitted in Partial Fulfillment of the Requirements for the Degree of Bachelor of Computer Engineering of the Nanyang Technological University

by

Do Nhat Minh

School of Computer Engineering

March 11, 2014

**Abstract**

Real-time operating systems play an important role in time and safty-critical software systems used in many fields, such as avionics, automotives and defense applications. $\mu$C/OS-III is an open-source real-time operating system which aims to be used on embedded devices with restricted resources. $\mu$C/OS-III has many useful features; however, it does not have better implementations proven in more recent advances in real-time systems research.

In this thesis, Earliest Deadline First scheduling algorithm and Stack Resource Policy for time-guaranteed resource sharing were implemented.

# Acknowledgements

I am particularly grateful to Dr. Arvind Easwaran from the School of Computer Engineering for being extremely supportive of my project choice and for giving invaluable advice during the course of this final year project.

I would like to thank Tran Ngoc Khanh Thy for the continued emotional support during the course of this final year project.

# Contents

# Chapter 1

# Introduction

An operating system is a collection of software, or software components, which can be characterized as serving the following purposes, (1) interfacing with the underlying hardware to provide convenient abstractions for application programmers, and (2) managing the programs running on the system so that misbehaving programs do not impede others (Witchel E., 2009).

A real-time operating system is an operating system which must adhere to a real-time constraint. In such a system, the timeliness of the results from programs are as important as the correctness of such solutions. The system risks catastrophic failures if deadlines are missed. Some important applications for real-time operating systems are in avionics and automotives, where missing task deadlines leads to lives lost.

$\mu$C/OS-III is an open-source priority-based pre-emptive real-time multitasking operating system for embedded systems. It has many useful features which aims to cut development time (Labrosse J. J., 2010, pp. 27-31).

However, $\mu$C/OS-III does not implement deadline management but defer this task to the programmer. Moreover, uC/OS-III has no facility to specify nor keep track of the deadlines for running tasks of the system.

The goal of this thesis is to implement better algorithms for task scheduling and resource sharing for $\mu$C/OS-III. For task scheduling, Earliest Deadline First scheduling is implemented, and for resource sharing, Stack Resource Policy is implemented.

# Chapter 2

# Overview of Micrium $\mu$C/OS-III

## 2.1 Task Model

Tasks in $\mu$C/OS-III are implemented as normal C functions with their own accompanying Task Control Blocks and stacks. However, unlike normal C functions, tasks are not allowed to return (Labrosse J. J., 2010, p. 83).

Task Control Block (TCB) is a C `struct` which holds necessary task-related information on which the whole of $\mu$C/OS-III depends in order to function properly. The information contained in a TCB includes a pointer to the top of stack, a pointer to the C function underlying this task, the current state of this task, the priority of this task and many more.

There are two type of tasks, namely run-to-completion and infinite loop. Run-to-completion tasks must call `OSTaskDel()` at the end of the function (Labrosse J. J., 2010, p. 84).

## 2.2 Scheduling Algorithm

$\mu$C/OS-III has a priority-based, preemptive scheduler. (Labrosse J. J., 2010, p. 141).

**Priority-based**

Each task are assigned a static priority. The kernel schedule them based on their priorities.

**Preemptive**

Higher priority tasks can preempt lower priority tasks, which means that during execution of a low priority task, if a high priority task is ready, the low priority task may be suspended so as to give CPU time to the high priority task.

## 2.3 Resource Sharing

# Chapter 3

# Modifications

## 3.1 Earliest Deadline First Scheduling

Earliest Deadline First Scheduling (EDF) is a dynamic scheduling algorithm.

### 3.1.1 Notes

EDF does not care about the relative deadline of the task but cares about its absolute deadline and its TCB.

The task spawner which spawns jobs for recurrent tasks cares about their relative deadlines and periodicities.

Because of task periodicities, the task spawner must allocate memory dynamically (for TCBs, stacks) to create job instances for tasks.

The OSRdyHeap is used to keep track of ready tasks with deadlines. The heap is a min heap w.r.t absolute deadlines.

The SpawnerHeap is used to keep track of which task to spawn next; this heap is also a min heap w.r.t. absolute spawn time.

$\Rightarrow$ Need for coarser time tick management: running spawner on every tick is expensive $\rightarrow$ cannot make guarantees about timeliness of task spawning.

Dynamic memory allocation problem:

Let the user allocate a block of memory for TCBs (this leads to problems with how tasks can communicate)

# Bibliography


Witchel, E. (2009) *CS372 Operating Systems.* Retrieved from http://www.cs.utexas.edu/users/witchel/372/lectures/01.OSHistory.pdf on 2014/03/10.

Labrosse, J. J. (2010) *$\mu$C/OS-III: The Real-Time Kernel.* Retrieved from http://micrium.com/books/ucosiii/ti-lm3s9b92/