LAB TUTORIALS-2A

Operating System and System Programming-ECE254

Kamal Lamichhane

Electrical and Computer Engineering, University of Waterloo, University of Waterloo

Sept 15, 2016

Who am I?

- MASc Student in the ECE Department
- Working in the area of Real-time Embedded Software Systems

When and Where to find me ?

- E5-5122
- kamal.lamichhane@uwaterloo.ca
- www.lamichhanekamal.com.np

When and Where to find me ?

- E5-5122
- kamal.lamichhane@uwaterloo.ca
- www.lamichhanekamal.com.np
- Office time - Learn, Calander in the website: www.lamichhanekamal.com.np/Cal.pdf

When and Where to find me ?

- E5-5122
- kamal.lamichhane@uwaterloo.ca
- www.lamichhanekamal.com.np
- Office time - Learn, Calander in the website: www.lamichhanekamal.com.np/Cal.pdf
- Drop an email if you need assistance any other time

Introduction

Task Management in ARM RL-RTX

Introduction

Task Management in ARM RL-RTX

- Read kernel task control block related data structure (PART-A)

Introduction

Task Management in ARM RL-RTX

- Read kernel task control block related data structure (PART-A)
- Block and unblock a task by using context switching related kernel functions (Part-B)

Introduction

Task Management in ARM RL-RTX

- Read kernel task control block related data structure (PART-A)
- Block and unblock a task by using context switching related kernel functions (Part-B)
- Memory Management

Introduction

Task Management in ARM RL-RTX

- Read kernel task control block related data structure (PART-A)
- Block and unblock a task by using context switching related kernel functions (Part-B)
- Memory Management
- NOTE :: Lab2/starter: Two files are changed from original source code of Keil so that the return value in TCB is changed to U32. Remember to include modified HAL_CM3.c and rt_Typredef.h.

Introduction

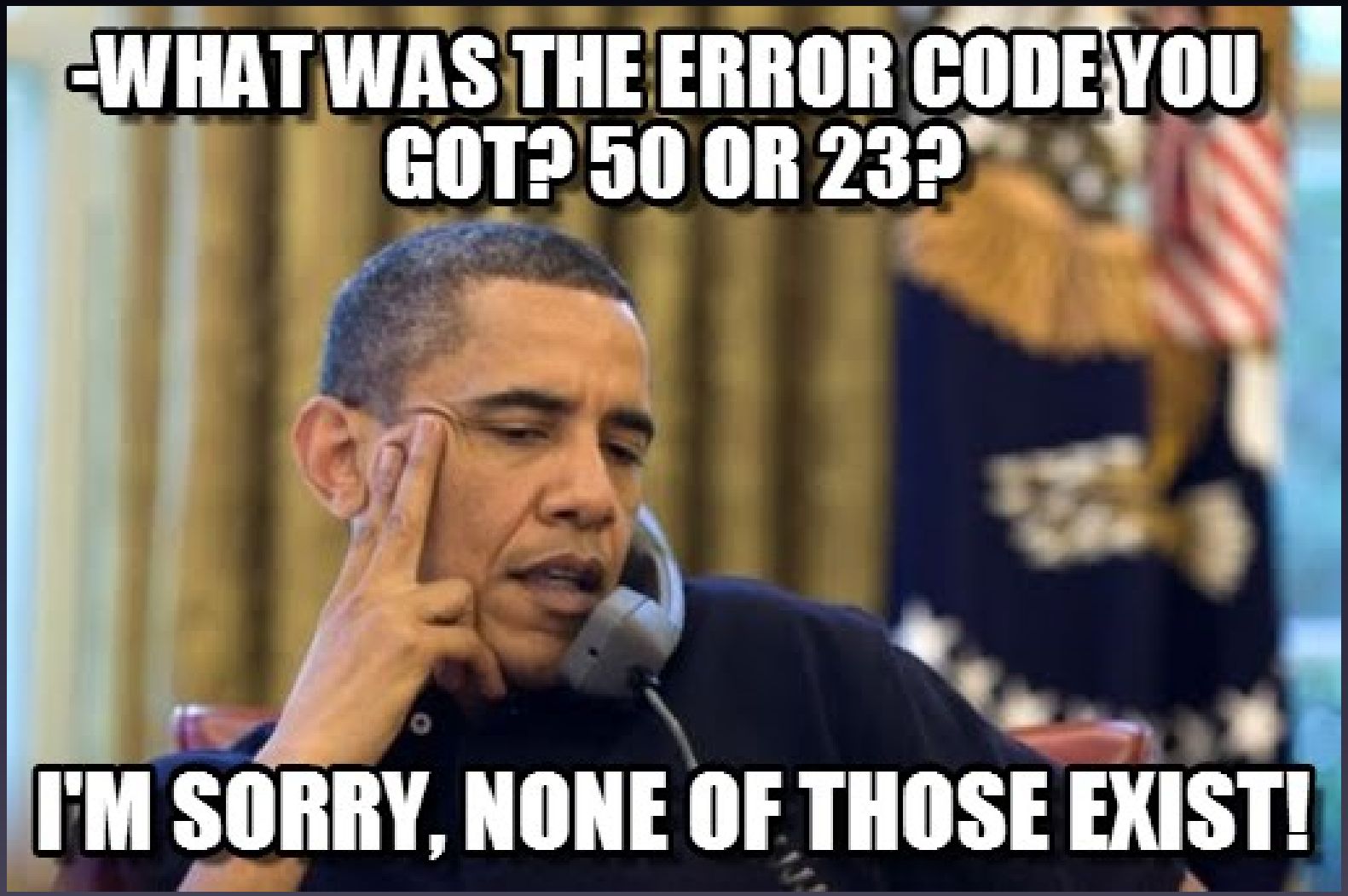
Task Management in ARM RL-RTX

- Read kernel task control block related data structure (PART-A)
- Block and unblock a task by using context switching related kernel functions (Part-B)
- Memory Management
- NOTE :: Lab2/starter: Two files are changed from original source code of Keil so that the return value in TCB is changed to U32. Remember to include modified HAL_CM3.c and rt_Typredef.h.
- main_task_exp.c- Subroutine to map a function pointer address to the function name.

- Stay Hungry. Stay Foolish. — Get your thinking clean to make it simple...

Answer the questions from lab manual 2.4.1.

Programming Project Part-A



What you have to do?

You are to implement a primitive to obtain the task status information from the RTX at runtime given the task id.

- OS_RESULT os_tsk_get (OS_TID task_id, RL_TASK_INFO *buffer)

Programming Project Part-A

The primitive returns information about a task. The system call returns a rl_task_info structure , which contains the following fields:

Programming Project Part-A

The primitive returns information about a task. The system call returns a rl_task_info structure , which contains the following fields:

```
typedef struct rl_task_info {
  U8   state;       /* Task state                                  */
  U8   prio;        /* Execution priority                          */
  U8   task_id;     /* Task ID value for optimized TCB access */
  U8   stack_usage; /* Stack usage percent value. eg.=58 if 58% */
  void (*ptask)(); /* Task entry address                          */
} RL_TASK_INFO;
```

Programming Project Part-A

STATES:

Programming Project Part-A

STATES:

- Inactive -Tasks which have not been started or deleted
- Ready - Tasks which are ready to run
- Runnning - Task that is currently running
- Wait_dly - Tasks which are waiting for a delay to expire.
- Wait_sem - Tasks which are waiting for a semaphore
- Wait_mut -Tasks which are waiting for a free mutex
- Wait_mbx - Tasks which are waiting for a mailbox message
- Wait_mem - Tasks which are waiting for memory

Programming Project Part-A

```
typedef struct rl_task_info {
  U8   state;       /* Task state                                  */
  U8   prio;        /* Execution priority                          */
  U8   task_id;     /* Task ID value for optimized TCB access */
  U8   stack_usage; /* Stack usage percent value. eg.=58 if 58% */
  void (*ptask)(); /* Task entry address                          */
} RL_TASK_INFO;
```

Programming Project Part-A

- The prio field describes the priority of this task.
- The task_id field describes the id of task assigned by the OS.
- The stack_usage describes how much stack space is used by this task. The value is the percent value. For example, if 58% of this task stack is used, stack_usage is set to 58.
- The ptask field describes the entry address of this task function.
- The function returns OS_R_OK on success and OS_R_NOK otherwise.

Deliverables

Submit a compressed archive file that contains the following:

1. A lab2_QA.txt file which contains answer to questions in Section 2.4.1.
2. Your entire multi-project workspace to solve the programming project.
3. A test description file to describe what each testing task does. Name the file Test_spec.txt.
4. A README file to describe what you have submitted and how to build and run your project(s).

Note

- Please Follow Source Code File Organization Convention.
- Check Third party testing in lab manual 2.4.4

Questions

Review Lecture 1, 2, and 3.

1. Introduction
2. Review of Computer Architecture
3. Operating system Structure and Traps