Real-Time Systems (2IMN20)
- Practical training 2015/2016 -

Leo Hatvani, Erik J. Luit and Reinder J. Bril
Technische Universiteit Eindhoven (TU/e),
Department of Mathematics and Computer Science,
Group System Architecture and Networking (SAN),
Den Dolech 2, 5612 AZ Eindhoven, The Netherlands
L.Hatvani@TUe.NL, E.J.Luit@TUe.NL, R.J.Bril@TUe.NL

Abstract

This document briefly describes the practical training of the real-time systems course.

1 Introduction

1.1 Background and motivation

The evaluation of the Real-Time Systems course (2IN26[1]) in 2012/2013 revealed that many MSc-ES students considered the course to be too theoretical and missed the link with real systems. Because a similar course for MSc-ES students at the TUD included a practical training, a practical training has been provided at the TU/e as well[2].

1.2 Goal

1.2.1 General

The *general goal* of the practical training is to bridge the gap between theory and practice of real-time systems, in particular for MSc-ES students. It is *not* the intention that students experiment with general operating systems related topics. Although those topics will re-appear in the course, like atomicity, they are not the focus of this course[3].

The *specific goal* of the practical training is to let students take cognizance of practical aspects, such as *event-latency*, *scheduling and context switching overhead*, *interrupt handling*, the risk of *drift*, and the influence of the platform (microprocessor and kernel) and the development environment (compiler) on the behavior of the system.

The *approach* taken for the practical training is to let students experiment with a set of simple real-time kernels that have been developed for a resource constrained embedded system, i.e. a wireless sensor platform based on a TI MSP430 microprocessor, using a simple application. Note that a wireless sensor platform is time/space/energy constrained. Students have to

- study, use, and adapt those kernels and the application,
- visualize behavior of the system and measure performance characteristics, and
- report upon the "experiments" performed.

With respect to *performing experiments*, students are expected to

[1]The identification of the Real-Time Systems course has been changed from 2IN26 to 2IMN20 in 2015/2016.
[2]As of 2014/2015, this practical training is also used at the TUD.
[3]These topics belong to the prerequisites of the course.

1. *formulate a "hypothesis"* about expected behavior of the system or result, e.g. using drawings or calculations;
2. *validate the hypothesis*, i.e. to
    - describe the steps to determine whether or not the hypothesis holds, e.g. visualize the actual behavior and measure performance characteristics;
    - execute those steps and report upon the actual behavior and results.
3. *reflect on the hypothesis*, e.g. to compare the expected and actual behavior, and comment on the differences encountered.

By the end of the training, students

1. can *explain* how real-time applications consisting of tasks are mapped onto a microprocessor;
2. can *explain* the influence of the platform (microprocessor and kernel) and the development environment (compiler) on the behavior of a system;
3. *know* how to perform experiments and report upon those experiments;
4. can *perform* experiments.

1.3 Specific

The training provides 4 simple real-time kernels, supporting *time-triggered* tasks, and a simple application toggling leds. These kernels differ in the way tasks are implemented and executed, i.e.

1. `SchedulerNPBasic.c`: *non-interruptable* task execution, i.e. tasks are executed as part of the timer-interrupt handler and can therefore neither be interrupted (by the timer) nor preempted;
2. `SchedulerNP.c`: *non-preemptive* task execution, i.e. tasks can be interrupted but can not be preempted;
3. `SchedulerPRE.c`: *preemptive* task execution, i.e. the kernel supports both tasks that can be preempted and tasks that can neither be preempted nor interrupted;
4. `SchedulerMultiTask.c`: *preemptive* task execution, supporting multi-threading, i.e. unlike the other versions, each task has its own stack.

Within the training, the last version will not be used, i.e. it is merely provided for studying purposes of interested students.

The students will be requested

- to study the kernels and answer specific questions about the kernels;
- to formulate, validate, and reflect on hypotheses addressing
    - the behavior of the system;
    - performance characteristics of the system, such as overhead of timer handling and event-latency ;
- to improve the efficiency of `SchedulerNP.c` ;
- to extend `SchedulerPre.c` with support for
    - *event-triggered* tasks, i.e. tasks that can be activated by external events or by other tasks ;
    - to extend `SchedulerPre.c` with support for limited-preemptive scheduling .

1.4 Prerequisites

Students are expected to be proficient with C and knowledgeable with operating systems. Experience with Linux is strongly recommended.

1.5 Organization

- *team-size*: Students are expected to work in pairs.
- *development environment*: The training involves working with Linux in a virtual machine, using Linux compile tools and using a hardware simulator. Although hardware is available (including an oscilloscope), the training is exclusively targeted for a hardware simulator.
- *study-material*: Exercises, sources, and development environment will become available via a web-site.
- *assistance*: Dr. Erik J. Luit will be available for consultation:
  - *when*: Tuesdays and Wednesday, 14:00 - 18:00;
  - *location*: MF-6.118.

  There will be a form available during the regular lecture hours and at Dr. Luit his door allowing you to reserve a time-slot.
- *deliverables*: The practical lasts 8 weeks. The first week is an introduction, so no deliverables need to be submitted. In the next weeks, students are expected to submit their results *before* Friday (i.e. latest Thursday's at 23:59h)!
- *grading*: For each week, you will be graded on a scale from 0 - 10. The grade for the practical training will be the sum of the grades for the weeks divided by the number of weeks (i.e. 7), and counts for 30% of the final grade for the course.
- *homework submissions*:
  - Mail your document to `L.Hatvani@TUe.NL`;
  - Add to the subject line: ”2IN26 practical assignment Week X”;
  - Add names + student IDs of each co-author to your document!
  - **Include code that you produce as text in your answers, not as a picture!** With text, it is easy to copy and run the code.

No deliverable (or too late) means 0 points!

2 General observations and recommendations

2.1 Educational purposes

Be aware that the real-time kernels and the practical training have primarily been developed for *educational purposes*. As a result, you may encounter examples in the exercises:

- of how *not* to proceed;
- of ideas that seem promising but lead to nothing.

2.2 Real-time systems

The utilization of the schedulers is unrealistic, i.e. extremely high and therefore taking a large fraction of the available cycles. This utilization is only chosen so that the effects of the choices made in the implementation are more easily observed.

As described above, the exercises illustrate issues in real-time systems, such as the effects of disabling interrupts, computation times that are larger than designed (or expected), ill-chosen parameters, etc.

2.3 Recommendations

Finally, we have the following recommendations for you when performing the practical training:

1. Keep in mind that this practical training is about real-time systems, so ask yourself if certain behavior that your solutions imply are desirable for this kind of systems.
2. Be critical and study the code carefully.
3. Be attentive to clues in the code provided.
4. Whenever a first question mentions something, this is in many cases a hint towards an implementation in a subsequent question.