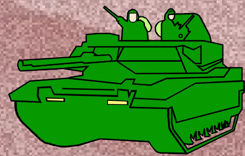
Real-Time Systems

Faculty

Paulo Pedreiras
pbrp@ua.pt
http://ppedreiras.av.it.pt/

Adapted from the slides developped by Prof. Luís Almeida for the course “Sistemas de Tempo-Real”



Preliminaries

What are real-time systems?

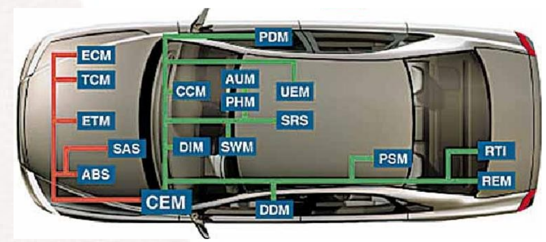
- **Computational** systems
- Subject to the evolution of **real-time**

  real-time progresses continuously, and the world evolve at its own pace
- Are those for which you **can't say**

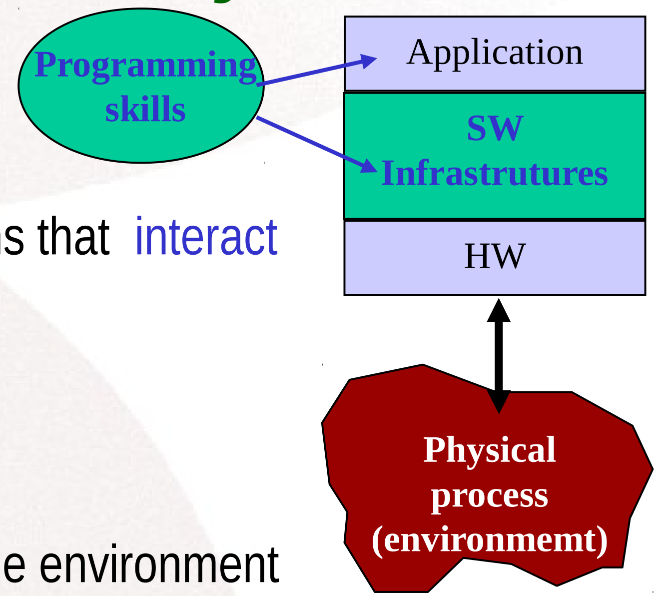
  Oh please, lets rewind ...
- Or, in different words, those in which

  What is done is done! And there are consequences ...
- Thus, the only way these systems perform correctly is when they

  Do the right thing at the right time!



Course objectives



Main topic:

- Software infrastructures and programming techniques for systems that interact with (or simulate a physical process (environment)

so that they can do the right thing at the right time

We will address:

- The source and characterization of the restrictions imposed by the environment to the temporal behavior of the computational system;
- Approaches to allow the computational system to be aware of the state of the environment that surround it;
- The scheduling theory of concurrent tasks associated with real-time processes;
- The structure and internal functionality of real-time operating systems/executives

FAQ

Isn't a quick processor sufficient?

- For a **simple program**, with a single taks, **that may work**. However, when CPUs have to execute **several tasks concurrently**, just a quick processor may **not be enough**! Some tasks may block others and cause big and/or unpredictable delays.

Then, what do we need?

- **Scheduling**! which means, select the right taks to execute in every instant. There are some task sorting techniques (scheduling algorithms) that allow predicting and minimizing the maximum delays that tasks may suffer.

FAQ

<u>So all this stuff applies only when we need *multi-tasking*... ?</u>

- As stated above, we are considering situations where a computer has to perform several tasks simultaneously. It is normal that in such situations we use one multi-tasking operating system/kernel. But often, even when the main body of the program is a simple endless loop, there are various pseudo-tasks, part of asynchronous interrupt routines, which lead to the same situation. The triggering of the interrupt routines may also be delayed, or even discarded. We must use proper techniques to bound and compute these delays.

FAQ

Are those delays so important?

- Well, if we're talking about control systems, and if these delays are such that lead to loss of samples, it is likely that control is lost! If this happens on a plane ... or a car with electronic actuation (X-by-wire) ... or a robot that moves around other people and equipment ... or a rocket ... there will be serious damage! On the other hand, if we are talking about multimedia systems, from games to DVDs, or routers in networks of computers, delays in tasks shall not cause death to anyone but there will be a loss of Quality-of-Service.

Bibliography

Base

- Giorgio Buttazzo (2011). **Hard Real-Time Computing Systems: Predictable Scheduling Algorithms and Applications,** Third Edition, Springer, 2011.
- Doug Abbott (2006). **Linux for Embedded and Real-time Applications,** Second Edition. Newnes, 2006;
- H. Kopetz (1997). ***Design Principles for Distributed Embedded Applications***, Kluwer Academic Publishers.

Complementary

- P. Veríssimo and L. Rodrigues (2001). ***Distributed System for Systems Architects***. Kluwer Academic Publishers.
- Jane W.S. Liu (2000). ***Real-Time Systems***. Prentice Hall.
- Briand, L. and Roy, D.M. (1999). ***Meeting Deadlines in Hard Real-Time Systems: the Rate-Monotonic Approach***. IEEE Computer Society Press, Los Alamitos (CA), USA.

(cont)

Bibliography

Complementary (cont.)

- Stankovic, J. *et al.* (1998). ***Deadline Scheduling for Real-Time Systems: EDF and Related Algorithms***. Kluwer Academic Publishers.
- Krishna, C.M. and K. Shin (1997). ***Real-Time Systems***. McGraw-Hill.
- N. Nissanke (1997), ***Real-Time Systems***, Prentice-Hall.
- Laplante, P.A., ***Real-Time Systems Design and Analysis - An Engineer's Handbook*** *(2nd ed.).* IEEE Press, 1997.
- Welling, A. and A. Burns (1996). ***Real-Time Systems and Their Programming Languages*** *(2nd ed.).* Int. Computer Science Series, Addison-Wesley.
- Klein, M. *et al.* (1993), ***A Practitioner's Handbook for Real-Time Analysis: Guide to Rate-Monotonic Analysis for Real-Time Systems***. Kluwer Academic Publishers.
- Richard Barry (2011). **Using the FreeRTOS Real-Time Kernel - A practical guide,** Real-Time Engineers, Ltd., 2011.

Course organization

- ***Theoretical component* – presentation and discussion of concepts and techniques**
  - Students should read selected parts of the base bibliography
  - Slides are available at the course website
  - Presentation and discussion of reserach works

- ***Lab component* - application of the studied techniques to practical scenarios**
  - Groups of two students
  - Tutorial classes to establish a basic set of practical competences (Linux (GPOS), Xenomai, FreeRTOS.)
  - One medium duration project per group
    - Proposals will be released soon. Suggestions are welcome!

Grading

Normal period

- Theoretical component - 50%:
  - 40% exam, 10% research work (5% by faculty, 5% by peers)
- Lab component – 50%:
  - 25% project, 5% log book, 10% oral presentation + 10% for additional work to the tutorial sessions

Recurrence period:

- Theoretical component – 50%
  - Written exam
- Lab component – 50%
  - Grade from the normal period or lab exam.

Planning

Real-Time Systems

2014/2015

Sept 15

Lecture 0+1: course presentation; basic concepts about real-time systems

Sept 22

Lecture 2: Computational models

Sept 29

Lecture 3: Kernels + tutorial GPOS

Oct 6

Lecture 4: basic concepts on scheduling + tutorial Xenomai

Oct 13

Lecture 5: periodic FP scheduling + tutorial Xenomai

Oct 20

Lecture 6: periodic FP scheduling + tutorial Xenomai (conc.)

Planning

Oct 27

Lecture 7: shared resources + tutorial FreeRTOS

Nov 3

Lecture 8: aperiodic task scheduling + tutorial FreeRTOS (conc.)

Nov 10

Lecture 9: other issues related with RT scheduling + projects

Nov 17

Lecture 10: optimizatons + projects

Nov 24

Lecture 11: projects

Dec 1

Lecture 12: project presentation + final considerations

Dec/8 is holiday and Dec/15 is the UA day (no classes during the afternoon)

And now ....

It is time to start working!