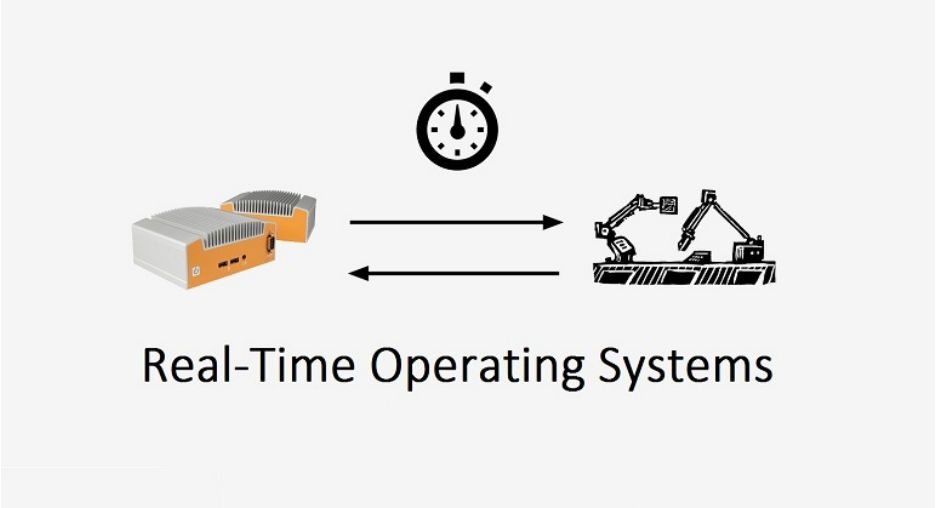

Real-time Operating Systems

Lab Module I - RT Services on Linux
2023/2024

Paulo Pedreiras
DETI/UA/IT
pbrp@ua.pt

September 19, 2023

1 Objectives

- Learn how to use the basic Linux real-time services, including creating periodic threads, assigning real-time priorities to threads, handle shared resources in real-time, etc.
- Use programmatic methods and OS tools to evaluate the real-time performance of threads
- Develop a simple real-time application in Linux.

2 Procedure

The Linux RT services laboratory activity comprises two parts.

The first one has a tutorial nature, aiming to allow students to get familiar with the basic methodologies and tools used for developing real-time applications in Linux. Its realization is optional, but highly recommended, not being subject to formal assessment.

The second part consists in the development of a real-time application in Linux, where students are expected to use the real-time model and the tools available in Linux to obtain the best possible real-time performance. The real-time application shall be demonstrated at the last class dedicated to this module. Students also have to submit a report that, together with the demonstration and code, is subject to formal evaluation.

3 Part I: Basic methodologies and tools

Below follows a set of tasks that students are advised to carry out in order to get familiar with the basic methods and tools available in Linux to develop real-time applications.

- [**PI-01**]: Download the code sample provided at the eLearning course page. Execute the code with and without load and analyze the output
- [**PI-02**]: Modify the code in [**PI-01**] to allow passing two command line arguments:
  i) priority (integer, [1,99]) and,
  ii) periodicity (integer, ms resolution, range [50,500] ms).

Execute the code with and without load, with different priority levels, and analyze output.

- [**PI-03**]: Execute several instances of the code developed in [**PI-02**], in different terminals, with different priorities and periodicity. Periods should not be harmonic and should generate a different phase conditions at runtime (e.g. use {90 ms, 95 ms, 107 ms, ...}. Analyze the results.

- [**PI-04**]: Modify the code developed in in [**PI-03**] so that processes are executed in the same CPU. Repeat the procedure, i.e.execute several instances of the code developed , in different terminals, with different priorities and periodicity. Periods should not be harmonic, generating different phase conditions at runtime (e.g. use {90 ms, 95 ms, 107 ms, ...}). Analyze the results and compare them with the previous ones.

- [**PI-05**]: Create an application composed of three real-time threads. The threads share one FIFO queue (supported on an array - static memory) of 100 integer numbers (uint16_t type). One of the threads (generator) is periodic (10 ms period) and in each job generates one random number that is added to the FIFO. The second thread (max) is sporadic and computes the maximum number that is/has been in the FIFO. The third thread (avg) is also sporadic and computes the average of the values in the FIFO. The max and avg sporadic threads are activated by the generator task. The max thread should be activated every 2 activations of the generator thread, while the avg thread should be activated every 5 activations of the generator thread. The generator thread should have higher priority than the other threads. Synchronization and mutual exclusion should be imposed by means of mutexes and condition variables.

4 Part II: Development of a real-time application

To be announced on Class 2