# ENSEEIHT - 3SN-E
# First Labs on Real-Time Scheduling

## Introduction

The objective of this lab is to experiment several classical algorithms and analysis methods of the real-time scheduling theory. For such a purpose, we will use Cheddar. This software provides a framework for the simulation and the analysis of task configurations with different real-time scheduling algorithms. In this lab, we will consider preemptive scheduling algorithms, except when non preemption is mentioned.

## A brief beginner's guide to Cheddar

In order to use Cheddar you will install it in your *nosafe* directory.
You can download binaries from *beru.univ-brest.fr/svn/CHEDDAR/releases/*. Choose the Cheddar-3.2-Linux64 version. You have to untar the files, set CHEDDAR_INSTALL_PATH in cheddar.bash and type "source cheddar.bash". Then you type ./cheddar.
The process to simulate a configuration is the following.
The hardware and software features of the configuration under study have to be created. This creation follows several steps.

- Creation of computation cores: the menu *edit/hardware/core* opens a window where core features are specified:
  - the name of the core, e.g. *core1*,
  - its scheduling policy, e.g. *Deadline Monotonic*, *POSIX 1003 Highest Priority First Protocol*,
  - the preemptive type, i.e. whether the scheduler is preemptive or not.

  The core is added to the configuration by clicking on button *add*. The window is closed by clicking on button *close*.
- Creation of processors: a processor is composed of a set of cores. The menu *edit/hardware/processor* opens a window where processor features are specified:
  - the name of the processor, e.g. *proc1*
  - its core(s) previously created, e.g. *core1*.

  The processor is added to the configuration by clicking on button *add*. The window is closed by clicking on button *close*.
- Creation of one address space per processor: the menu *edit/software/address space* opens a window where address space features are specified. In the context of this lab, no features but the name of the processor have to be specified. An address space is added to the configuration by clicking on button *add*. The window is closed by clicking on button *close*.
- Creation of tasks: the menu *edit/software/task* opens a window where task features are specified:
  - the name of the task, e.g. *T1*,
  - the type of the task, i.e. periodic, aperiodic, ...,
  - the WCET (*Capacity*) of the task,
  - the start time of the task, i.e. its first release,

  - the deadline of the task,
  - the period of the task.

  A task is added to the configuration by clicking on button *add*. The window is closed by clicking on button *close*.

- Creation of shared resources (if there are some in the configuration): the menu *edit/software/resource* opens a window where resource features are specified:
  - the name of the resource,
  - the state of the resource, i.e. its number of available tokens (at least one at the beginning),
  - the protocol to access the resource, e.g. *Priority Inheritance Protocol*,
  - the tasks sharing the resources, with the starting time and ending time of utilization during their execution.

  A resource is added to the configuration by clicking on button *add*. The window is closed by clicking on button *close*.

In the monoprocessor case, we will create a single processor composed of a single core.
Then, the configuration can be simulated by using menu *tools/scheduling/customize scheduling simulation*.
Cheddar main window is split in two parts. The upper part displays the time lines computed by the simulation process, while the lower part displays various numerical results.

# Exercise 1

1. Simulate the following task configuration on one core using *deadline monotonic*.

| | First release | WCET | D | P |
|---|---|---|---|---|
| $T_1$ | 0 | 2 | 5 | 5 |
| $T_2$ | 0 | 2 | 4 | 10 |

   (a) Is it schedulable?
   (b) What are the worst-case response times of the tasks?

2. Is this configuration schedulable using any fixed priority order?

# Exercise 2

1. Let's assume the following independant task configuration.

| | First release | WCET | D | P |
|---|---|---|---|---|
| $T_1$ | 0 | 2 | 6 | 6 |
| $T_2$ | 0 | 4 | 10 | 10 |
| $T_3$ | 0 | 2 | 20 | 20 |

   Is it schedulable by a fixed priority algorithm? If not, is it schedulable with Earliest Deadline First?

2. Same question with the following independant task configuration.

| | First release | WCET | D | P |
|---|---|---|---|---|
| $T_1$ | 0 | 2 | 6 | 6 |
| $T_2$ | 0 | 4 | 10 | 10 |
| $T_3$ | 0 | 2 | 9 | 20 |

3. Same question with the following independant task configuration.

| | First release | WCET | D | P |
|---|---|---|---|---|
| $T_1$ | 0 | 2 | 6 | 6 |
| $T_2$ | 0 | 4 | 8 | 10 |
| $T_3$ | 0 | 2 | 9 | 20 |

## Exercise 3

In this exercise we investigate another dynamic scheduling policy: LLF (Least Laxity First). This policy selects the task to run among the ready tasks according to a dynamic priority called 'laxity': the smaller the laxity, the higher the priority. Li(t), the laxity of a task i at time t can be computed by

$$Li(t) = Deadline - remaining(t)$$

where remaining(t) is the remaining capacity of the task at time t.
Let's assume the following independant task configuration.

| | First release | WCET | D | P |
|---|---|---|---|---|
| $T_1$ | 0 | 3 | 8 | 8 |
| $T_2$ | 0 | 4 | 9 | 9 |

1. Compare and comment the scheduling sequences obtained by Ealiest Deadline First and Least Laxity First.

## Exercise 4

In this exercise we investigate the effect of non preemption on scheduling.

1. Let's assume the following independant task configuration.

| | First release | WCET | D | P |
|---|---|---|---|---|
| $T_1$ | 0 | 1 | 3 | 3 |
| $T_2$ | 0 | 3 | 9 | 9 |

   Compare and comment the scheduling sequences obtained by Preemptive Rate Monotonic and Non Premptive Rate Monotonic

2. Same question with the following independant task configuration.

| | First release | WCET | D | P |
|---|---|---|---|---|
| $T_1$ | 0 | 2 | 3 | 3 |
| $T_2$ | 0 | 3 | 9 | 9 |

## Exercise 5

Let's assume the following dependant task configuration:

| | First release | WCET | D | P |
|---|---|---|---|---|
| $T_1$ | 0 | 1 | 10 | 10 |
| $T_2$ | 2 | 1 | 8 | 10 |
| $T_3$ | 0 | 2 | 10 | 10 |
| $T_4$ | 0 | 1 | 10 | 10 |
| $T_5$ | 0 | 1 | 10 | 10 |
| $T_6$ | 0 | 8 | 15 | 20 |

We have the following precedence constraints:

- $T_1$ and $T_2$ have to complete execution before $T_3$ starts,
- $T_3$ has to complete execution before $T_4$ and $T_5$.

1. Is it schedulable using the rate monotonic approach presented in the lectures?
2. Same question with the Earliest Deadline First approach presented in the lectures.