Politecnico di Milano

Dipartimento Elettronica, Informazione e Bioingegneria

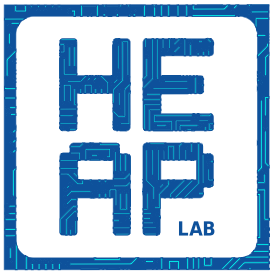
HEAPLab Project Report

SchedSim

Author:
Francesco Ratti

Supervisor:
Dr. Giusppe Massari

21/03/2021



1 Introduction

This application is a real time schedulers simulator developed using Python and PyQt for the GUI.

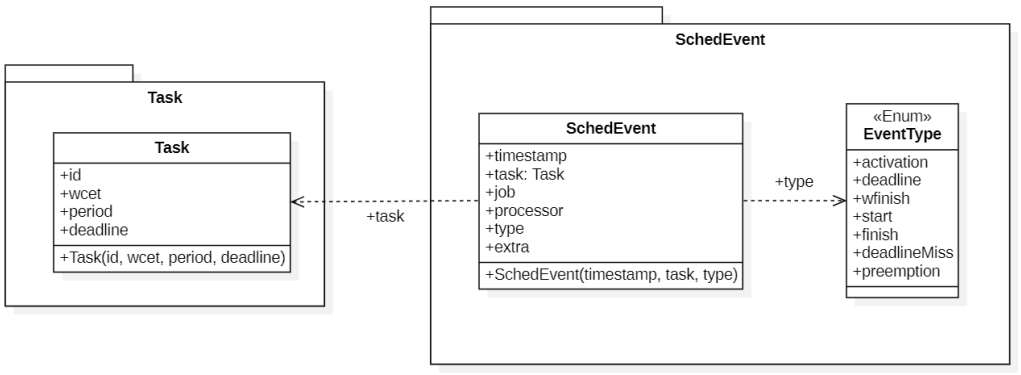
Given a set of tasks, which can be defined in the program and can be imported/exported as XML, SchedSim provides as output a list of temporally ordered events, in CSV format, ready to be displayed in a temporal chart. This tool can be useful for simulating new types of schedulers (for research purposes for example) and, subsequently, visualize the results starting from the "event list" output file by means of an external application.
Output event code is described in the dedicated section.

SchedSim provides a simple interface (abstract class) which can be implemented to allow user to include his/her own scheduling algorythm. Deadline Monotonic algorythm is built-in and provided as a working example.

2 Design and Implementation

All the GUI classes and method are in the main.py file. Other functions are splitted in the corresponding files, as later described.

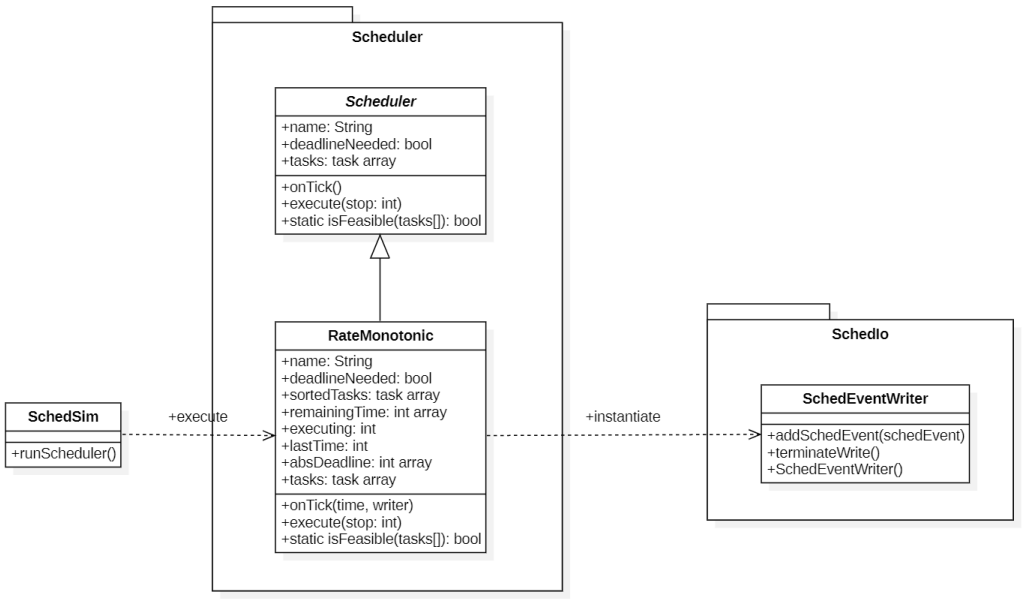
SchedIo.py contains all inputs and outputs functions needed to import and export to XML file the tasks set and SchedEventWriter, which is used to write the output file. More details later.



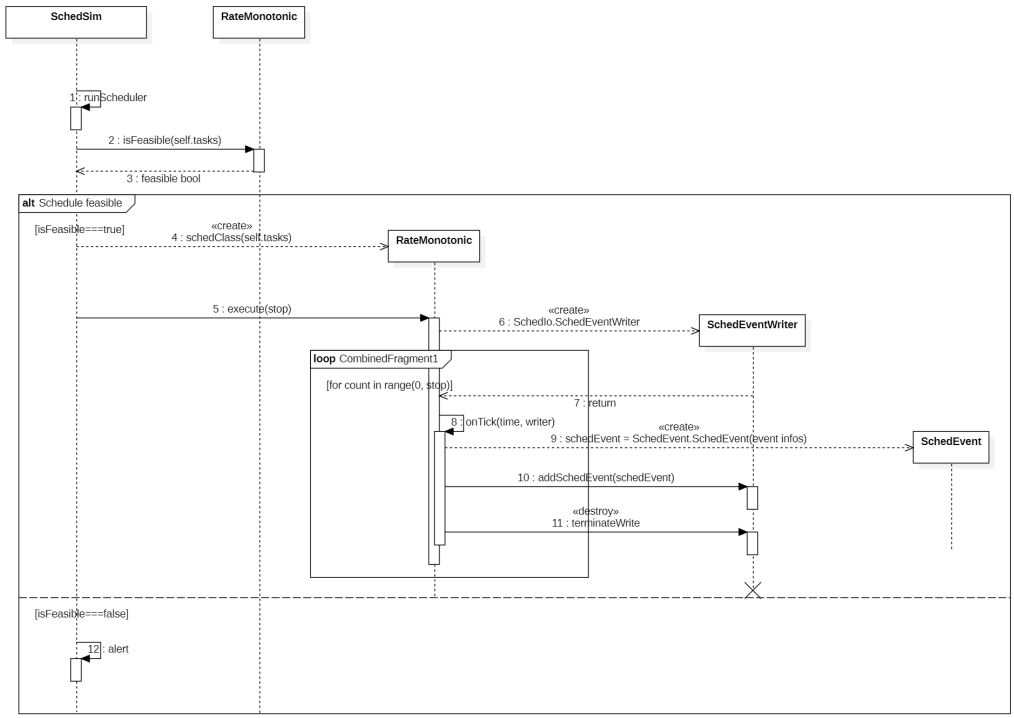
SchedEvent.py contains SchedEvent class, the one dedicated to time events, which will be exported in the output file, and an enum to set the code which corresponds to the given event. SchedEvent represent an event, associated

to a timestamp, which will be part of the output list of temporally ordered events. In the current version, job, processor and extra are set to 0 but they can be easily implemented in future versions by adding these parameters in the SchedEvent class constructor.

Tasks.py contains Task class, which represent a single task with its attributes.



Scheduler.py contains the core of the system: all the scheduling algorythms are located there. In particular, one can add his/her own schedulers by extending Scheduler class, overriding execute and onTick methods and adding the corresponding scheduler class name in the "schedulers" variable (an array) in the file. Moreover, when extending Scheduler class one must override "name" and "deadlineNeeded", which must be set to true when it is mandatory that the user specifies a deadline for every task when using the given scheduler. For instance, it is set to false for RateMonotonic scheduler. It is needed to override "isFeasible" method too, which should return a boolean value. This method will be called by SchedSim to check whether the scheduling is feasible before actually executing it by means of calling "execute"; if it is not, an error message is shown.



runScheduler method, which is part of SchedSim class (main GUI class), is run when user clicks "Run" button. "isFeasible" inside the selected scheduler class is the called to check schedule feasibility and iff schedule is feasible, scheduler is istantated and execute is called. SchedEventWriter class is instantiated inside "execute" and "onTick" method, it will produce the scheduled events output file. When an event occours a SchedEvent is instantiated and SchedEventWriterinstance.addSchedEvent(schedEvent) is called. At the end, SchedEventWriterinstance.terminateWrite() is called to write the file.

3 Input and output

3.1 Exported task set file structure

Example of exported task set as XML by means of "Export" function:

```
<simulation>
<software>
<tasks>
```

```
<periodictask id="1" period="10" deadline="-1" wcet="2" />
<periodictask id="2" period="60" deadline="-1" wcet="10" />
<periodictask id="3" period="30" deadline="-1" wcet="5" />
</tasks>
</software>
</simulation>
```

3.2 Output file format

Each line of the CSV file represents an event and it is composed of 6 fields:

`timestamp,task,job,processor,type_of_event,extra_data`

where:

- timestamp: the elapsed time since the epoch of measurements (usually 0). This can be measured in different time units (e.g. seconds, milliseconds, clock cycles, etc.). This may not be unique in the file, multiple events may happen at the same time.
- task: the task id. If the event refers to a processor-only event, this value is 0.
- job: the job id. If the event refers to a processor-only event, this value is 0.
- processor: the processor id where the event happened. If the event is not processor-related, this value is 0.
- type_of_event: the identificator for the event (see later)
- extra_data: additional data depending on the event type, this value is 0 if not used.

Possible events for tasks/jobs:

- 'A': activation of a job (processor-independent)
- 'D': deadline of a job (processor-independent)
- 'W': theoretical worst-case finish time for the job ('A' + WCET) (it may differ depending on the processor)

- ‘S‘: actual start of a job
- ‘F‘: actual finish of a job
- ‘M‘: deadline miss

Possible processor-only events:

- ‘+‘: processor goes online
- ‘-‘: processor goes offline
- ‘F‘: frequency change (‘extra_data‘ contains the new frequency in MHz)

Example

```
0,1,1,0,A,0   t=0 Task 1 Job 1 activates
0,2,1,0,A,0   t=0 Task 2 Job 1 activates
0,1,1,1,S,0   t=0 Task 1 Job 1 starts on processor 1
0,2,1,2,S,0   t=0 Task 2 Job 1 starts on processor 2
3,2,1,2,F,0   t=3 Task 2 Job 1 finishes
5,2,1,0,D,0   t=5 Task 2 Job 1 deadline
5,2,2,0,A,0   t=5 Task 2 Job 2 activates
5,2,2,2,S,0   t=5 Task 2 Job 2 starts on processor 2
8,1,1,1,F,0   t=8 Task 1 Job 1 finishes
9,2,2,2,F,0   t=9 Task 2 Job 2 finishes
10,2,2,0,D,0   t=10 Task 2 Job 2 deadline
15,1,1,0,D,0   t=15 Task 1 Job 1 deadline
```

4 Screenshots

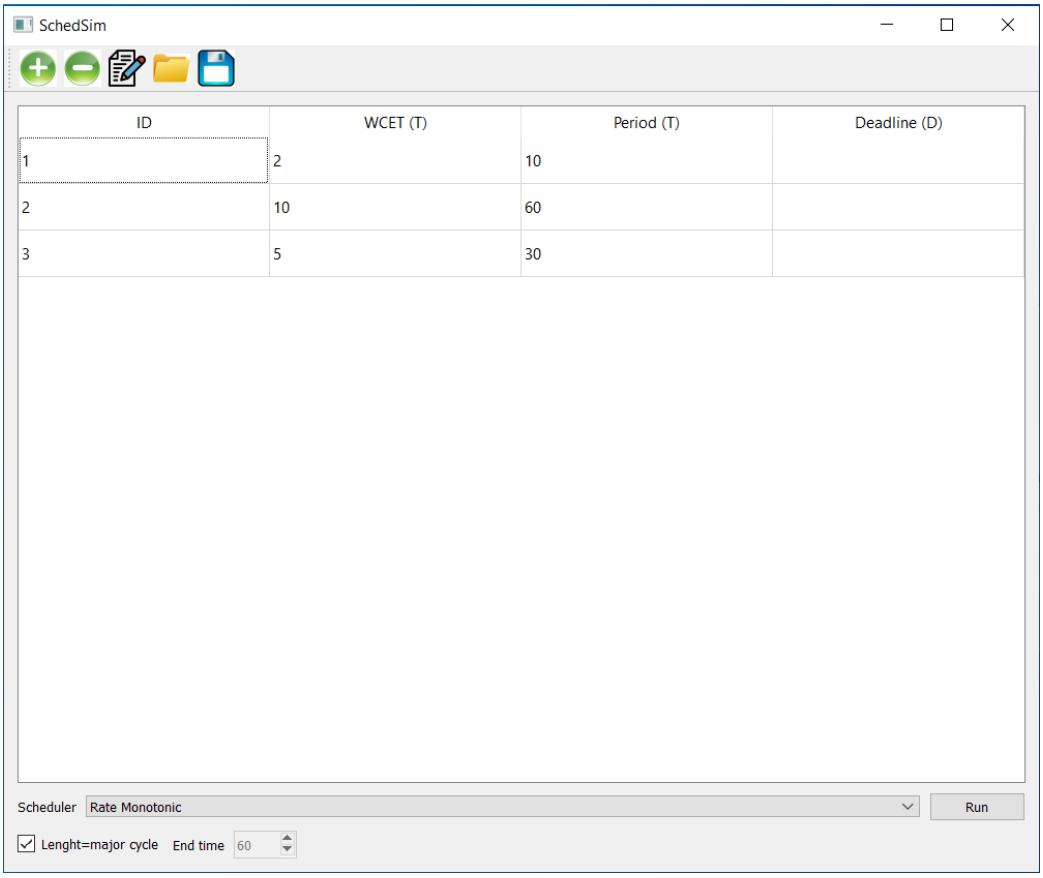


Figure 1: Main page

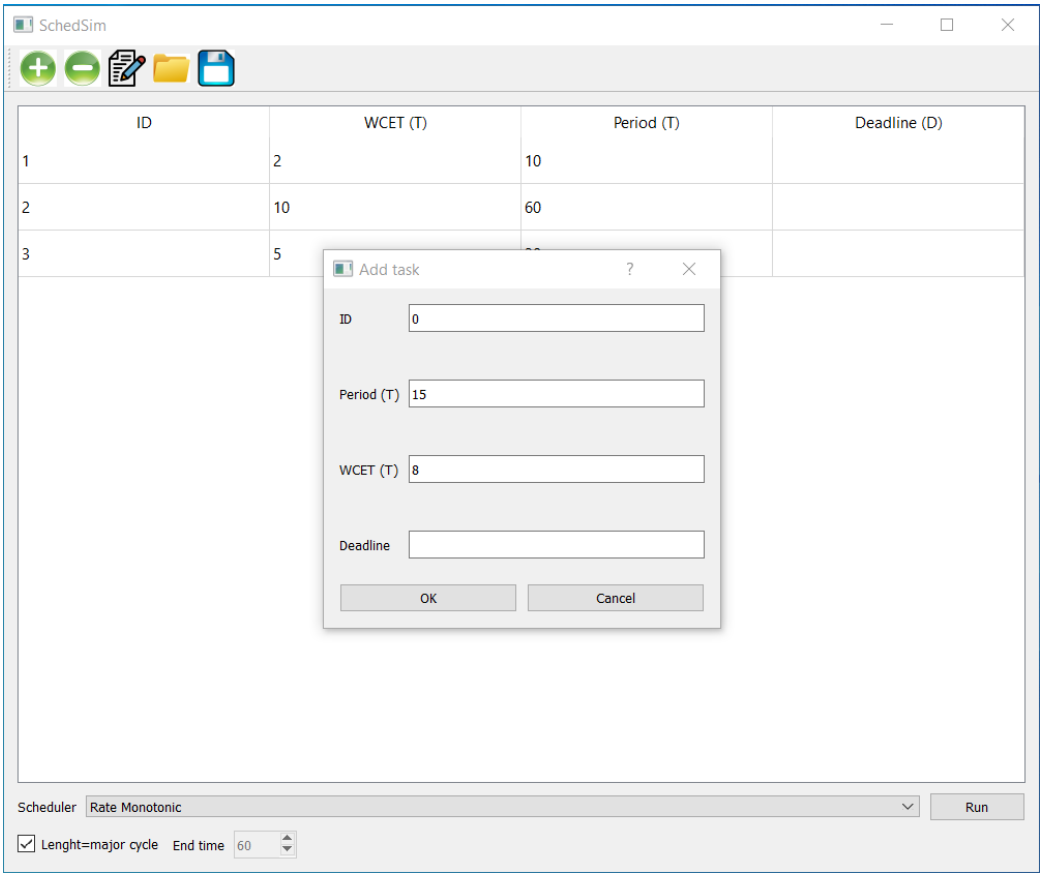


Figure 2: Adding a task

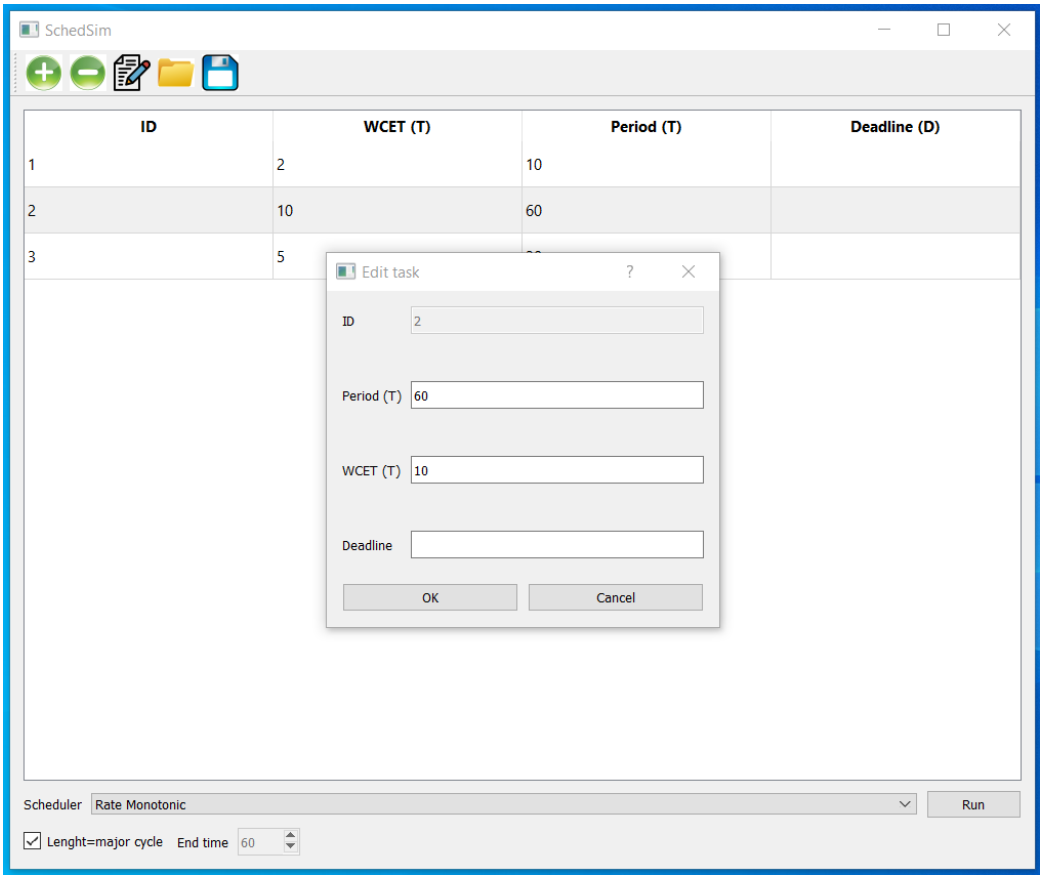


Figure 3: Editing a task

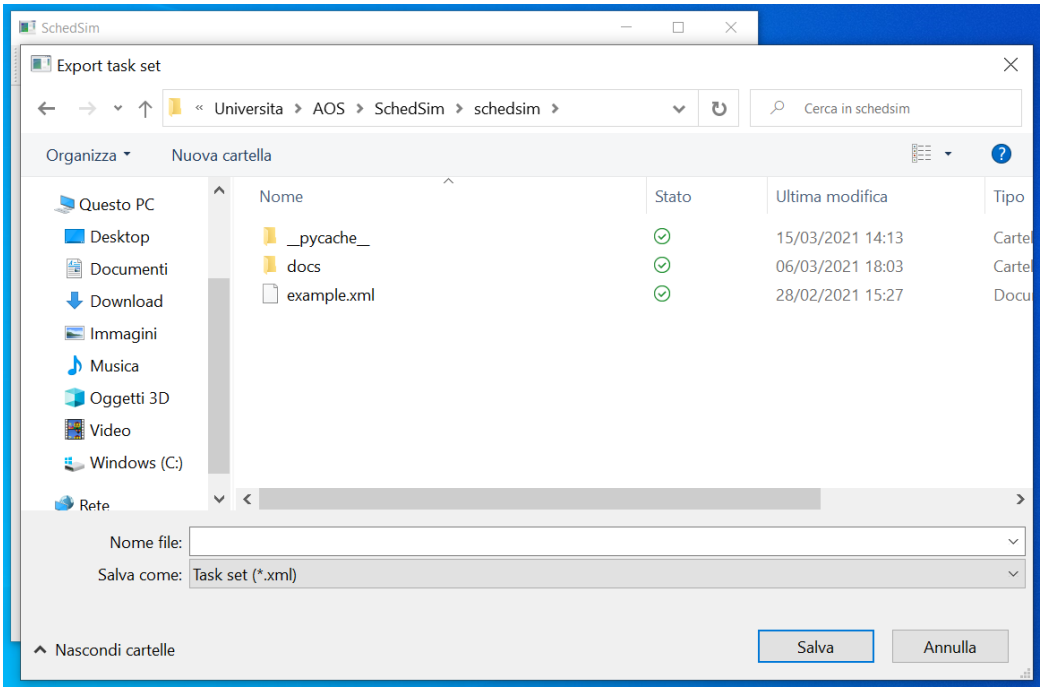


Figure 4: Exporting a task set

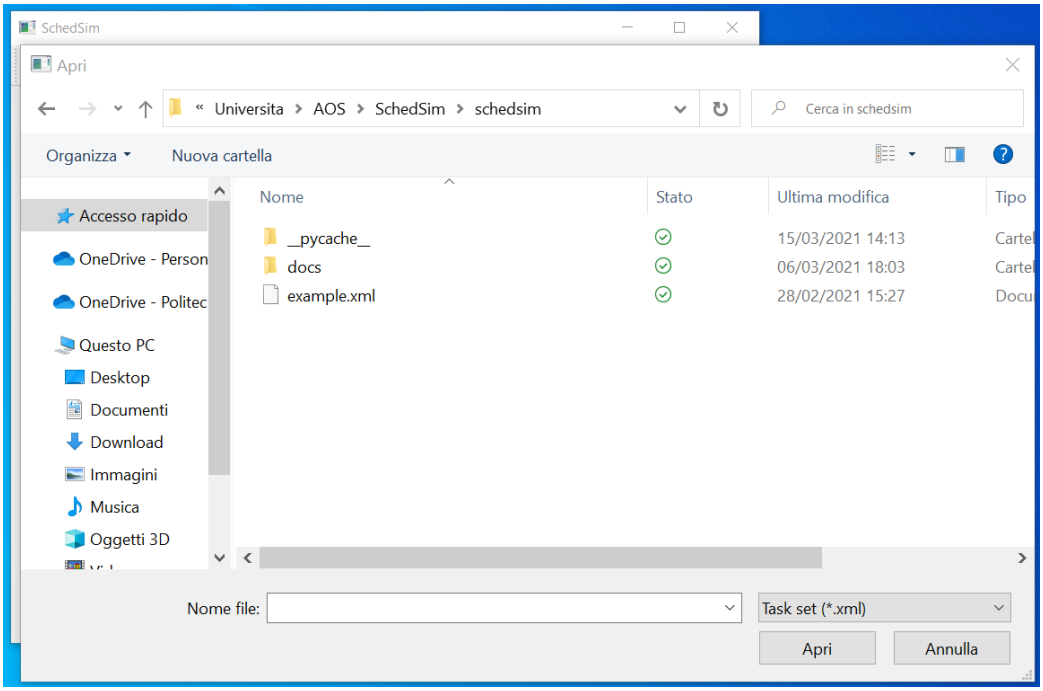


Figure 5: Importing a task set

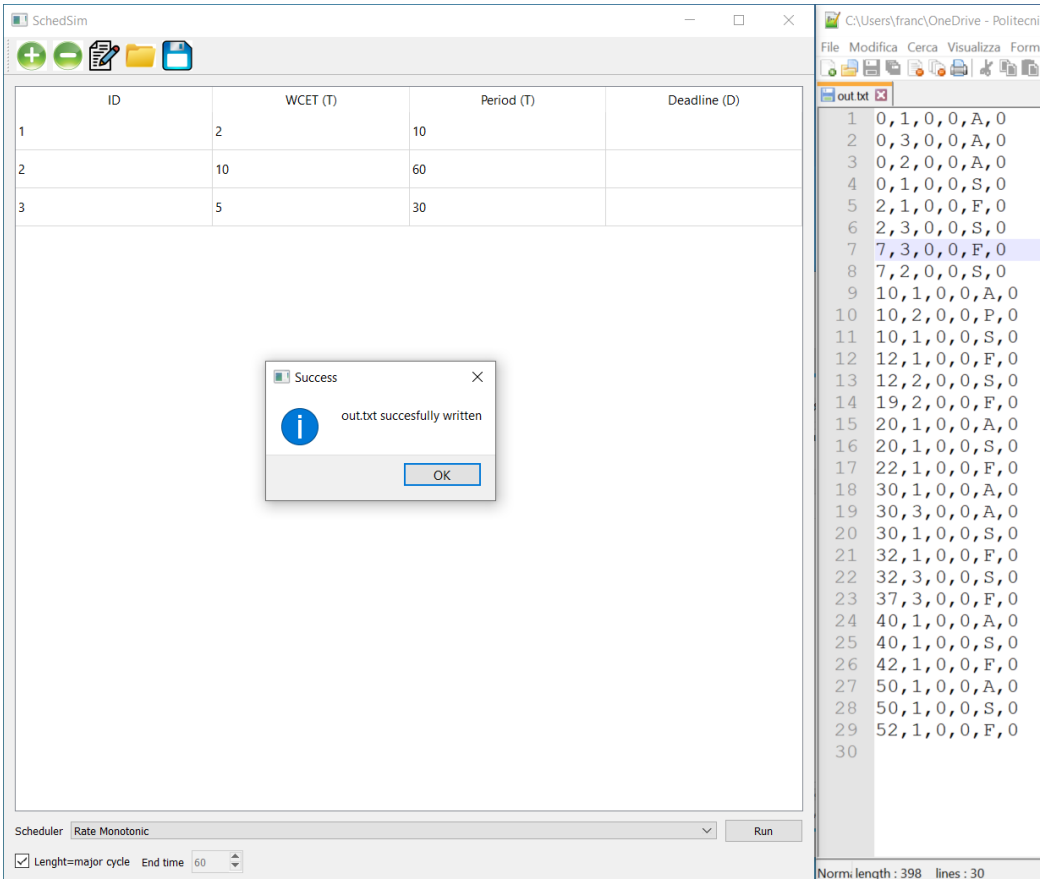


Figure 6: Example of output given a task set, RateMonotonic scheduler