Sistemas Embarcados - Trabalho Prático I

Italo Qualisoni
PUCRS
italo.qualisoni@acad.pucrs.br

1. Algoritmos

Neste trabalho será comparado dois algoritmos de escalonamento sendo eles o Rate-Monotonic(secção 1.1). e Earliest Deadline First (secção 1.2).

1.1. Rate-Monotonic

Algoritmo que leva em cosideração a prioridade da *Task* e a duração do período da mesma, sendo a escalonada aquela com maior prioridade e/ou menor duração de período. Este algoritmo possuí a vantagem de ter a certeza de qua uma tarefa com alta prioridade será executada, mas não da melhor forma possível de questão do escalonamento das tarefas como um todo podendo deixar de escalonar uma tarefa de baixa prioridade.

1.2. Earliest Deadline First

Este algoritmo leva em consideração de modo dinâmico o tempo mais curto do próximo deadline da *Task*, ignorando sua prioridade. É um método muito interessante, mas acaba tirando a garantia de que uma tarefa com prioridade alta vai ser escalonada, fato que o algoritmo RM fornecia como premissa.

2. Parte 1

Nesta seção será mostrada as adaptações realizadas no kernel para a adaptação do algoritmo EDF.

2.1. Primeira Mudança

Primeramente foi adicionado um atributo para a *Task* chamado **deadline_counter** e seu objetivo é de armazenar quantas frações de tempo está o próximo deadline da tarefa, este dado será necessário para usar como critério de priorização no algoritmo EDF. *Arquivo: tcb.h* Linha: 86 *Tipo: uint16_t* Nome do atributo: deadline_counter

```
typedef struct{
  ...
  uint16_t next_deadline;            // Linha 86
  ...
}tcb;
```

2.2. Segunda Mudança

Após adicionar o atributo, foi populada a informação na hora de adicionar a *Task* com o valor incial igual ao **deadline** da *Task*

- Arquivo: **ukernel.c**
- Linha: 537
- Nome do método: HF_AddPeriodicTask

```
int32_t HF_AddPeriodicTask(...){
  ...
  HF_task_entry->next_deadline = deadline; //Linha 537
  ...
};
```

2.3. Terceira Mudança

Após essas mudanças, foi necessário modificar a lógica implementada na hora de realizar o escalonamento das *Tasks* para levar em consideração o atributo adicionado na seção 2.1 e

- Arquivo: **ukernel.c**
- Linha: 537
- Nome do método: HF_TaskReschedule

```
uint8_t HF_TaskReschedule(void){
  ...
  if ((HF_task_entry->status == TASK_READY) || (HF_task_entry->status == TASK_NOT_RUN)){
      if ((HF_task_entry->next_deadline < j) && (HF_task_entry->capacity_counter > 0)){
          //if ((HF_task_entry->period < j) && (HF_task_entry->capacity_counter > 0)){
            j = HF_task_entry->next_deadline;
          //j = HF_task_entry->period;
            schedule = i;
      }
      if (--HF_task_entry->priority == 0){
        HF_task_entry->next_tick_count += HF_task_entry->period;
        HF_task_entry->next_deadline += HF_task_entry->deadline;
        HF_task_entry->priority = HF_task_entry->period;
        if (HF_task_entry->capacity_counter > 0)
          HF_task_entry->deadline_misses++;
          HF_task_entry->capacity_counter = HF_task_entry->capacity;
        }
  }
  ...
};
```

3. Parte 2

Abaixo estão descritos três exemplos de escalonamentos comparando os algoritmos.

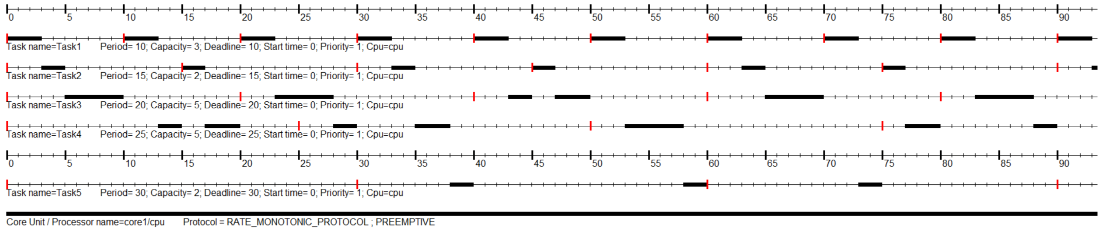
4. Exemplo 1

Neste primeiro exemplo foi abordado um cenário onde o algoritmo EDF se destaca em relação ao RM por conseguir escalonar todas as tarefas por utilizar periodos harmonicos entre as tarefas.

Tarefa	Capacidade	Período	Deadline
task1	3	10	10
task2	2	15	15
task3	5	20	20
task4	5	25	25
task5	2	30	30

Table 1. Dados do exemplo 1

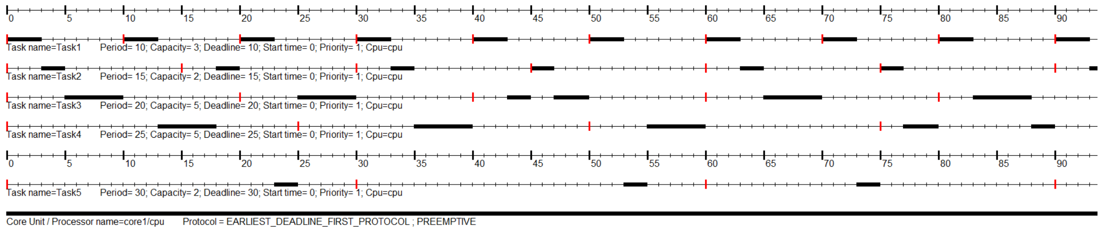
4.1. RM




```
Scheduling simulation, Processor cpu :
- Number of context switches :  97
- Number of preemptions :  11

- Task response time computed from simulation :
    Task1 => 3/worst
    Task2 => 5/worst
    Task3 => 10/worst
    Task4 => 20/worst
    Task5 => 40/worst ,  missed its deadline (absolute deadline =  30 ; completion time =  40)
- Some task deadlines will be missed : the task set is not schedulable.
```

4.2. EDF




```
Scheduling simulation, Processor cpu :
- Number of context switches :  95
- Number of preemptions :  9

- Task response time computed from simulation :
    Task1 => 3/worst
    Task2 => 5/worst
    Task3 => 10/worst
    Task4 => 18/worst
    Task5 => 25/worst
- No deadline missed in the computed scheduling : the task set is schedulable if you computed the scheduling on the feasibility interval.
```

4.3. Simulador

```
#include <prototypes.h>

int32_t lastTasksTicks[6];

void task(void){
    int32_t tid;

    tid = HF_CurrentTaskId();

    for (;;){
```

```
        if(lastTasksTicks[tid] != HF_TaskTicks(tid)){
            lastTasksTicks[tid] = HF_TaskTicks(tid);
            printf("%d,%d\n", tid,HF_TaskTicks(tid));
        }
    }
}

void ApplicationMain(void){
    //period,capacity,deadline,description...
    HF_AddPeriodicTask(task, 10, 3, 10, "task1", 2048, 0, 0);
    HF_AddPeriodicTask(task, 15, 2, 15, "task2", 2048, 0, 0);
    HF_AddPeriodicTask(task, 20, 5, 20, "task3", 2048, 0, 0);
    HF_AddPeriodicTask(task, 25, 5, 25, "task4", 2048, 0, 0);
    HF_AddPeriodicTask(task, 30, 2, 30, "task5", 2048, 0, 0);
    HF_Start();
}
```

Taks	Momento 1	Momento 2	Momento 3	Momento 4	Momento 5	Momento 6	Momento 7	Momento 8	Momento 9	Momento 10
1	X			X			X			X
2		X				X				
3			X						X	
4					X					
5								X		

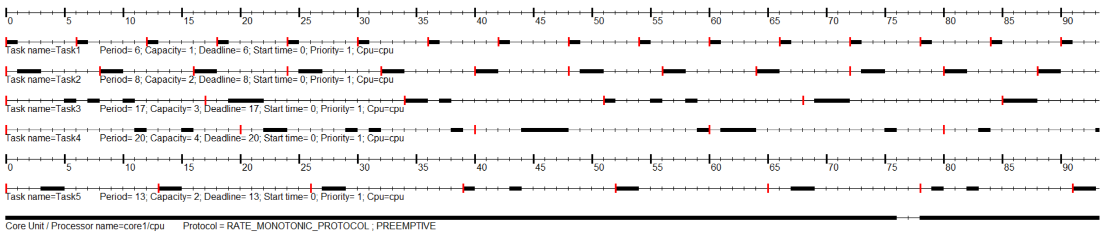
5. Exemplo 2

Neste segundo exemplo foi abordado um cenário onde o algoritmo EDF consiga escalonar todas as tarefas com periodos não harmonicos entre elas e onde o algoritmo RM não consiga realizar o mesmo.

Tarefa	Capacidade	Período	Deadline
task1	1	6	6
task2	2	8	8
task3	3	17	17
task4	4	20	20
task5	2	13	13

Table 2. Dados do exemplo 2

5.1. RM

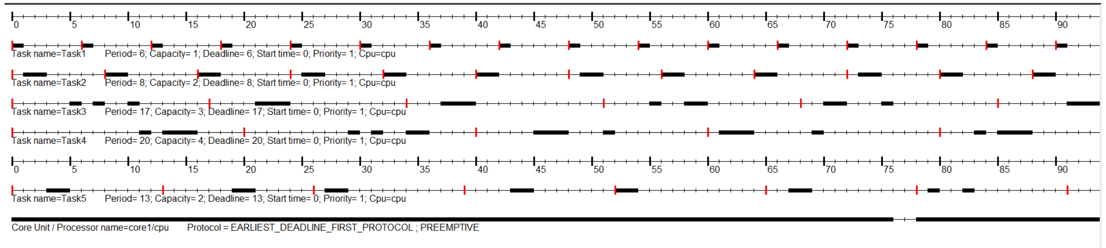



```
Scheduling simulation, Processor cpu :
- Number of context switches :  16170
- Number of preemptions :  3589

- Task response time computed from simulation :
    Task1 => 1/worst
    Task2 => 3/worst
    Task3 => 11/worst
    Task4 => 25/worst ,  missed its deadline (absolute deadline =  20 ; completion time =  24),  missed its deadline (absolute deadline =  40 ; completi
```

```
    Task5 => 5/worst
- Some task deadlines will be missed : the task set is not schedulable.
```

5.2. EDF




```
Scheduling simulation, Processor cpu :
- Number of context switches :  15937
- Number of preemptions :  3277

- Task response time computed from simulation :
    Task1 => 1/worst
    Task2 => 3/worst
    Task3 => 12/worst
    Task4 => 16/worst
    Task5 => 9/worst
- No deadline missed in the computed scheduling : the task set is schedulable if you computed the scheduling on the feasibility interval.```
```

5.3. Simulador

~~~~
#include <prototypes.h>

int32_t lastTasksTicks[6];

void task(void){
    int32_t tid;

    tid = HF_CurrentTaskId();

    for (;;){
        if(lastTasksTicks[tid] != HF_TaskTicks(tid)){
            lastTasksTicks[tid] = HF_TaskTicks(tid);
            printf("%d,%d\n", tid,HF_TaskTicks(tid));
        }
    }
}

Nesta seção será mostrada as adaptações realizadas no kernel para a adaptação do algoritmo EDF.

## Primeira Mudança { #change1 }

Primeramente foi adicionado um atributo para a _Task_ chamado **deadline_counter**
e seu objetivo é de armazenar quantas frações de tempo está o próximo deadline da tarefa,
 este dado será necessário para usar como critério de priorização no algoritmo EDF.

* Arquivo: **tcb.h**
* Linha: 86
* Tipo: uint16_t
* Nome do atributo: deadline_counter

``` cpp
typedef struct{
  ...
~~~~

```
  uint16_t next_deadline;// Linha 86
  ...
}tcb;
```

5.4. Segunda Mudança

Após adicionar o atributo, foi populada a informação na hora de adicionar a *Task* com o valor incial igual ao **deadline** da *Task*

- Arquivo: **ukernel.c**
- Linha: 537
- Nome do método: HF_AddPeriodicTask

```
int32_t HF_AddPeriodicTask(void (*task)(), uint16_t period, uint16_t capacity, uint16_t deadline, int8_t description[], uint32_t stack_size, uint8_t nic
  ...
  HF_task_entry->next_deadline = deadline; //Linha 537
  ...
};
```

5.5. Terceira Mudança

Após essas mudanças, foi necessário modificar a lógica implementada na hora de realizar o escalonamento das *Tasks* para levar em consideração o atributo adicionado na seção 2.1 e

- Arquivo: **ukernel.c**
- Linha: 537
- Nome do método: HF_TaskReschedule

```
uint8_t HF_TaskReschedule(void){
  ...
if ((HF_task_entry->status == TASK_READY) || (HF_task_entry->status == TASK_NOT_RUN)){
    if ((HF_task_entry->next_deadline < j) && (HF_task_entry->capacity_counter > 0)){
        //if ((HF_task_entry->period < j) && (HF_task_entry->capacity_counter > 0)){
          j = HF_task_entry->next_deadline;
       // j = HF_task_entry->period;
                schedule = i;
    }
    if (--HF_task_entry->priority == 0){
            HF_task_entry->next_tick_count += HF_task_entry->period;
            //Atualiza o próximo deadline
      HF_task_entry->next_deadline += HF_task_entry->deadline;
            HF_task_entry->priority = HF_task_entry->period;
            if (HF_task_entry->capacity_counter > 0)
                HF_task_entry->deadline_misses++;
                HF_task_entry->capacity_counter = HF_task_entry->capacity;
       }
    }
  ...
};
```

6. Parte 2

Abaixo estão descritos três exemplos de escalonamentos comparando os algoritmos.

```
![exemp2-simulador]
[exemp2-simulador]: images/exemp2-simulador.png "exemp2-simulador" { width:auto; max-width:150% }


# Exemplo 3
Neste primeiro exemplo foi abordado um cenário onde ambos algoritmos não consigam escalonar as tarefas por possuír utilização superior a 100%. Objetivo

~ TableFigure { #tab-example1 caption="Dados do exemplo 2" }
|--------|------------|---------|----------|
| Tarefa | Capacidade | Período | Deadline |
```

```
+--------|------------|---------|----------+
| task1  | 1          | 5       | 5        |
+--------|------------|---------|----------+
| task2  | 3          | 10       | 10       |
+--------|------------|---------|----------+
| task3  | 3          | 15     |  15      |
+--------|------------|---------|----------+
| task4  | 5          | 20      | 20       |
|--------|------------|---------|----------|
|task5   | 2          | 25      | 25       |
|--------|------------|---------|----------|
~
```

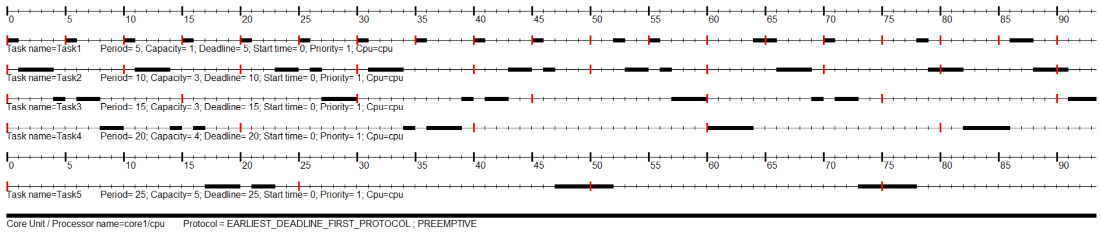
```
##RM
![exmp3-rm]

[exmp3-rm]: images/exmp3-rm.png "exmp3-rm" { width:auto; max-width:150% }
```

Scheduling simulation, Processor cpu : - Number of context switches : 179 - Number of preemptions : 52

- Task response time computed from simulation : Task1 => 1/worst Task2 => 4/worst Task3 => 8/worst Task4 => 20/worst Task5 => 175/worst , missed its deadline (absolute deadline = 25 ; completion time = 59), missed its deadline (absolute deadline = 50 ; completion time = 118), missed its deadline (absolute deadline = 75 ; completion time = 160), missed its deadline (absolute deadline = 100 ; completion time = 219), missed its deadline (absolute deadline = 125 ; completion time = 270), missed its deadline (absolute deadline = 150 ; completion time = 300)
- Some task deadlines will be missed : the task set is not schedulable. `

6.1. EDF




```
Scheduling simulation, Processor cpu :
- Number of context switches :  123
- Number of preemptions :  10

- Task response time computed from simulation :
    Task1 => 24/worst ,  missed its deadline (absolute deadline =  85 ; completion time =  87),  missed its deadline (absolute deadline =  105 ; complet
    Task2 => 29/worst ,  missed its deadline (absolute deadline =  80 ; completion time =  82),  missed its deadline (absolute deadline =  90 ; completi
    Task3 => 36/worst ,  missed its deadline (absolute deadline =  90 ; completion time =  94),  missed its deadline (absolute deadline =  105 ; complet
    Task4 => 43/worst ,  missed its deadline (absolute deadline =  60 ; completion time =  64),  missed its deadline (absolute deadline =  80 ; completi
    Task5 => 47/worst ,  missed its deadline (absolute deadline =  50 ; completion time =  52),  missed its deadline (absolute deadline =  75 ; completi
- Some task deadlines will be missed : the task set is not schedulable.
```

6.2. Simulador

```
#include <prototypes.h>

int32_t lastTasksTicks[6];
```

```
void task(void){
    int32_t tid;

    tid = HF_CurrentTaskId();

    for (;;){
        if(lastTasksTicks[tid] != HF_TaskTicks(tid)){
            lastTasksTicks[tid] = HF_TaskTicks(tid);
            printf("%d,%d\n", tid,HF_TaskTicks(tid));

        }
    }
}
~ End Remote
# Exemplo 1
Neste primeiro exemplo foi abordado um cenário onde o algoritmo EDF se destaca em relação ao RM por conseguir escalonar todas as tarefas por utilizar pe

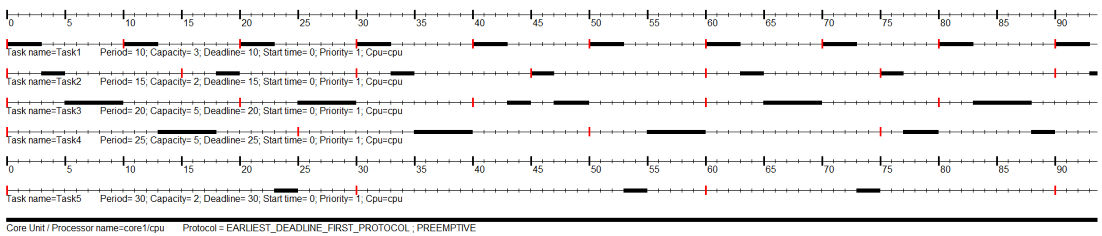
~ TableFigure { #tab-example1 caption="Dados do exemplo 1" }
|--------|------------|---------|----------|
| Tarefa | Capacidade | Período | Deadline |
+--------|------------|---------|----------+
| task1  | 3          | 10      | 10       |
+--------|------------|---------|----------+
| task2  | 2          | 15      | 15       |
+--------|------------|---------|----------+
| task3  | 5          | 20      | 20       |
+--------|------------|---------|----------+
| task4  | 5          | 25      | 25       |
|--------|------------|---------|----------|
|task5   | 2          | 30      | 30       |
|--------|------------|---------|----------|
~


##RM
![exmp1-rm]
[exmp1-rm]: images/exmp1-rm.png "exmp1-rm" { max-width:150% }
```

Scheduling simulation, Processor cpu : - Number of context switches : 97 - Number of preemptions : 11

- Task response time computed from simulation : Task1 => 3/worst Task2 => 5/worst Task3 => 10/worst Task4 => 20/worst Task5 => 40/worst , missed its deadline (absolute deadline = 30 ; completion time = 40)
- Some task deadlines will be missed : the task set is not schedulable. `

6.3. EDF




```
Scheduling simulation, Processor cpu :
- Number of context switches :  95
```

```
- Number of preemptions :  9

- Task response time computed from simulation :
    Task1 => 3/worst
    Task2 => 5/worst
    Task3 => 10/worst
    Task4 => 18/worst
    Task5 => 25/worst
- No deadline missed in the computed scheduling : the task set is schedulable if you computed the scheduling on the feasibility interval.
```

6.4. Simulador

```c
#include <prototypes.h>

int32_t lastTasksTicks[6];

void task(void){
    int32_t tid;

    tid = HF_CurrentTaskId();

    for (;;){
        if(lastTasksTicks[tid] != HF_TaskTicks(tid)){
            lastTasksTicks[tid] = HF_TaskTicks(tid);
            printf("%d,%d\n", tid,HF_TaskTicks(tid));
        }
    }
}

void ApplicationMain(void){
    //period,capacity,deadline,description...
    HF_AddPeriodicTask(task, 10, 3, 10, "task1", 2048, 0, 0);
    HF_AddPeriodicTask(task, 15, 2, 15, "task2", 2048, 0, 0);
    HF_AddPeriodicTask(task, 20, 5, 20, "task3", 2048, 0, 0);
    HF_AddPeriodicTask(task, 25, 5, 25, "task4", 2048, 0, 0);
    HF_AddPeriodicTask(task, 30, 2, 30, "task5", 2048, 0, 0);
    HF_Start();
}
```

Taks	Momento 1	Momento 2	Momento 3	Momento 4	Momento 5	Momento 6	Momento 7	Momento 8	Momento 9	Momento 10
1	X			X			X			X
2		X				X				
3			X						X	
4					X					
5								X		

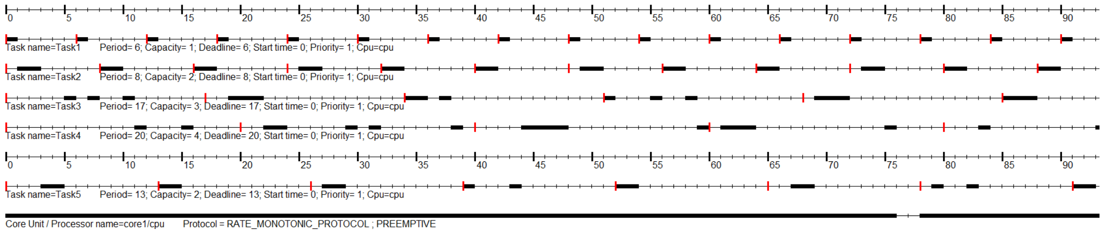
7. Exemplo 2

Neste segundo exemplo foi abordado um cenário onde o algoritmo EDF consiga escalonar todas as tarefas com periodos não harmonicos entre elas e onde o algoritmo RM não consiga realizar o mesmo.

Tarefa	Capacidade	Período	Deadline
task1	1	6	6
task2	2	8	8
task3	3	17	17
task4	4	20	20
task5	2	13	13

Table 3. Dados do exemplo 2

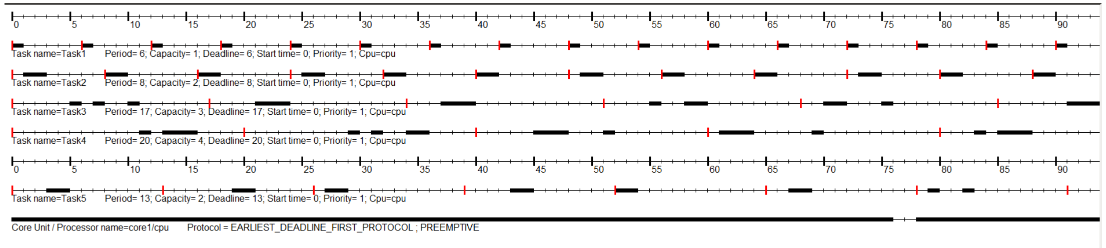
7.1. RM




```
Scheduling simulation, Processor cpu :
- Number of context switches :  16170
- Number of preemptions :  3589

- Task response time computed from simulation :
    Task1 => 1/worst
    Task2 => 3/worst
    Task3 => 11/worst
    Task4 => 25/worst ,  missed its deadline (absolute deadline =  20 ; completion time =  24),  missed its deadline (absolute deadline =  40 ; completi
    Task5 => 5/worst
- Some task deadlines will be missed : the task set is not schedulable.
```

7.2. EDF




```
Scheduling simulation, Processor cpu :
- Number of context switches :  15937
- Number of preemptions :  3277

- Task response time computed from simulation :
    Task1 => 1/worst
    Task2 => 3/worst
    Task3 => 12/worst
    Task4 => 16/worst
    Task5 => 9/worst
- No deadline missed in the computed scheduling : the task set is schedulable if you computed the scheduling on the feasibility interval.```
```

7.3. Simulador

```c
#include <prototypes.h>

int32_t lastTasksTicks[6];

void task(void){
    int32_t tid;

    tid = HF_CurrentTaskId();
```

```
    for (;;){
        if(lastTasksTicks[tid] != HF_TaskTicks(tid)){
            lastTasksTicks[tid] = HF_TaskTicks(tid);
            printf("%d,%d\n", tid,HF_TaskTicks(tid));
        }
    }
}

void ApplicationMain(void){
    //period,capacity,deadline,description...
    HF_AddPeriodicTask(task, 6, 1, 6, "task1", 2048, 0, 0);
    HF_AddPeriodicTask(task, 8, 2, 8, "task2", 2048, 0, 0);
    HF_AddPeriodicTask(task, 17, 3, 17, "task3", 2048, 0, 0);
    HF_AddPeriodicTask(task, 20, 4, 20, "task4", 2048, 0, 0);
    HF_AddPeriodicTask(task, 13, 2, 13, "task5", 2048, 0, 0);
    HF_Start();
}
```

Taks	Momento 1	Momento 2	Momento 3	Momento 4	Momento 5	Momento 6	Momento 7	Momento 8	Momento 9	Momento 10
1	X				X				X	
2		X					X			
3				X		X				
4								X		X
5			X							

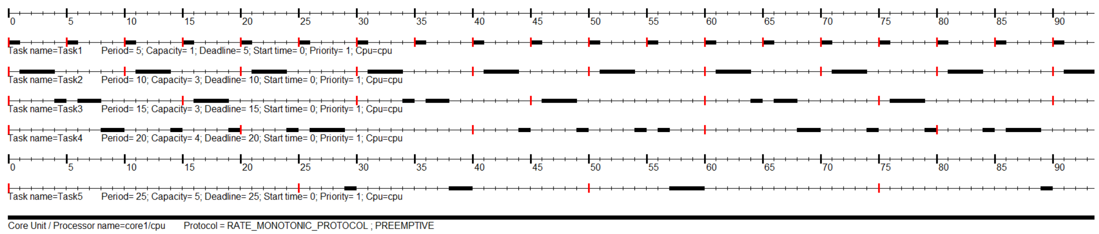
8. Exemplo 3

Neste primeiro exemplo foi abordado um cenário onde ambos algoritmos não consigam escalonar as tarefas por possuír utilização superior a 100%. Objetivo deste caso também é mostrar a diferença da escalonação entre os algoritmos.

Tarefa	Capacidade	Período	Deadline
task1	1	5	5
task2	3	10	10
task3	3	15	15
task4	5	20	20
task5	2	25	25

Table 4. Dados do exemplo 2

8.1. RM

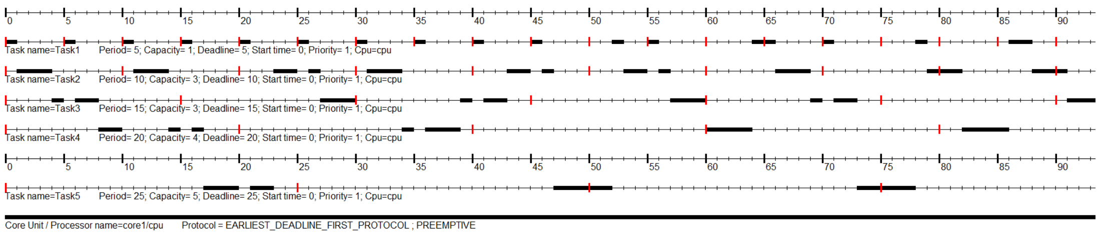



```
Scheduling simulation, Processor cpu :
- Number of context switches :  179
- Number of preemptions :  52

- Task response time computed from simulation :
    Task1 => 1/worst
```

```
    Task2 => 4/worst
    Task3 => 8/worst
    Task4 => 20/worst
    Task5 => 175/worst ,  missed its deadline (absolute deadline =  25 ; completion time =  59),  missed its deadline (absolute deadline =  50 ; complet
- Some task deadlines will be missed : the task set is not schedulable.
```

8.2. EDF




```
Scheduling simulation, Processor cpu :
- Number of context switches :  123
- Number of preemptions :  10

- Task response time computed from simulation :
    Task1 => 24/worst ,  missed its deadline (absolute deadline =  85 ; completion time =  87),  missed its deadline (absolute deadline =  105 ; complet
    Task2 => 29/worst ,  missed its deadline (absolute deadline =  80 ; completion time =  82),  missed its deadline (absolute deadline =  90 ; completi
    Task3 => 36/worst ,  missed its deadline (absolute deadline =  90 ; completion time =  94),  missed its deadline (absolute deadline =  105 ; complet
    Task4 => 43/worst ,  missed its deadline (absolute deadline =  60 ; completion time =  64),  missed its deadline (absolute deadline =  80 ; completi
    Task5 => 47/worst ,  missed its deadline (absolute deadline =  50 ; completion time =  52),  missed its deadline (absolute deadline =  75 ; completi
- Some task deadlines will be missed : the task set is not schedulable.
```

8.3. Simulador

```c
#include <prototypes.h>

int32_t lastTasksTicks[6];

void task(void){
    int32_t tid;

    tid = HF_CurrentTaskId();

    for (;;){
        if(lastTasksTicks[tid] != HF_TaskTicks(tid)){
            lastTasksTicks[tid] = HF_TaskTicks(tid);
            printf("%d,%d\n", tid,HF_TaskTicks(tid));
<!-- end merge -->

        }
    }
}

void ApplicationMain(void){
    //period,capacity,deadline,description...
    HF_AddPeriodicTask(task, 5, 1, 5, "task1", 2048, 0, 0);
    HF_AddPeriodicTask(task, 10, 3, 10, "task2", 2048, 0, 0);
    HF_AddPeriodicTask(task, 15, 3, 15, "task3", 2048, 0, 0);
    HF_AddPeriodicTask(task, 20, 5, 20, "task4", 2048, 0, 0);
    HF_AddPeriodicTask(task, 25, 2, 25, "task5", 2048, 0, 0);
    HF_Start();
}
```

Taks	Momento 1	Momento 2	Momento 3	Momento 4	Momento 5	Momento 6	Momento 7	Momento 8	Momento 9	Momento 10
1	X			X			X			X
2		X						X		
3			X		X					
4						X			X	
5										

void ApplicationMain(void){ //period,capacity,deadline,description… HF_AddPeriodicTask(task, 5, 1, 5, “task1”, 2048, 0, 0); HF_AddPeriodicTask(task, 10, 3, 10, “task2”, 2048, 0, 0); HF_AddPeriodicTask(task, 15, 3, 15, “task3”, 2048, 0, 0); HF_AddPeriodicTask(task, 20, 5, 20, “task4”, 2048, 0, 0); HF_AddPeriodicTask(task, 25, 2, 25, “task5”, 2048, 0, 0); HF_Start(); }

`

Taks	Momento 1	Momento 2	Momento 3	Momento 4	Momento 5	Momento 6	Momento 7	Momento 8	Momento 9	Momento 10
1	X			X			X			X
2		X						X		
3			X		X					
4						X			X	
5										