Systèmes Embarqués et Enfouis
(SMB204)

Tâches périodiques : cas limite

En vous inspirant du code type correspondant à une tâche périodique en Ada (vu en cours) et du code donné en Annexe permettant de simuler une tâche ayant un temps d'exécution donné (temps CPU de la tâche), écrire un programme de simulation d'un ensemble de tâches périodiques spécifiées par leurs temps d'exécution et leur période.

1. Utiliser Cheddar pour simuler l'execution de l'exemple « limite » vu en cours pour les valeurs suivantes (en faisant varier C_2) :
   - $\tau_1 = \{C_1 = 41,\ T_1 = 100\}$ et $\tau_2 = \{C_2 = \mathbf{58},\ T_2 = 141\}$
   - $\tau_1 = \{C_1 = 41,\ T_1 = 100\}$ et $\tau_2 = \{C_2 = \mathbf{59},\ T_2 = 141\}$
   - $\tau_1 = \{C_1 = 41,\ T_1 = 100\}$ et $\tau_2 = \{C_2 = \mathbf{60},\ T_2 = 141\}$

2. On simulera ensuite le même exemple en Ada. Pour ces deux tâches, avec les paramètres $\tau_1 = \{C_1 = 41,\ T_1 = 100\}$ et $\tau_2 = \{C_2 = \mathbf{59},\ T_2 = 141\}$, l'affichage obtenu devra avoir une forme similaire à celui-ci, qui met en évidence l'echec de l'ordonnancement :

```
Task  1 running:  0s 0.000148105 wall clock :  0.000102000
Task  1 ending :  0s 0.410148388 wall clock :  0.411136000
Task  2 running:  0s 0.000018304 wall clock :  0.411144000
Task  1 running:  0s 0.410157198 wall clock :  0.999991000
Task  1 ending :  0s 0.820177468 wall clock :  1.411039000
Task  2 ending :  0s 0.590018516 wall clock :  1.413679000
Task  2 failed to meet deadline.
```

3. Pour une charge légèrement inférieure de $\tau_2 = \{C_2 = \mathbf{58},\ T_2 = 141\}$, l'affichage normal attendu devra ressembler à celui-ci :

```
Task  1 running:  0s 0.000114513 wall clock :  0.000039000
Task  1 ending :  0s 0.410114779 wall clock :  0.411081000
Task  2 running:  0s 0.000026344 wall clock :  0.411093000
Task  2 ending :  0s 0.580026622 wall clock :  0.992547000
Task  1 running:  0s 0.410137115 wall clock :  1.000035000
Task  1 ending :  0s 0.820137479 wall clock :  1.411063000
Task  2 running:  0s 0.580033021 wall clock :  1.411073000
Task  2 ending :  1s 0.160033392 wall clock :  1.992527000
Task  1 running:  0s 0.820156710 wall clock :  2.000038000
Task  1 ending :  1s 0.230157003 wall clock :  2.411065000
...
```

4. Enfin, pour une charge légérement supérieure de $\tau_2 = \{C_2 = \mathbf{60},\ T_2 = 141\}$, donner l'affichage obtenu. Un message d'erreur devra être affiché lorsqu'une tâche ne respecte pas son échéance.

Remarque. Le code donné en annexe contient aussi quelques fonctions auxiliaires pemettant d'afficher les differents types `Time`, `CPU_time` et `Time_Span`.

Annexe

```
pragma Task_Dispatching_Policy(FIFO_Within_Priorities);

with Ada.Text_IO, Ada.Integer_Text_IO, Ada.Real_Time, Ada.Execution_Time;
use Ada.Text_IO, Ada.Integer_Text_IO, Ada.Real_Time, Ada.Execution_Time;


procedure Main is

   function To_String (T : Time_Span) return String is
   begin
      return (Duration'Image(To_Duration(T)));
   end;

   function To_String (T : Time) return String is
      SC : Seconds_Count;
      TS : Time_Span;
   begin
      Split(T, SC, TS);
      return (Seconds_Count'Image(SC) & "s" & Duration'Image(To_Duration(TS)));
   end;

   function To_String (T : CPU_Time) return String is
      SC : Seconds_Count;
      TS : Time_Span;
   begin
      Split(T, SC, TS);
      return (Seconds_Count'Image(SC) & "s" & Duration'Image(To_Duration(TS)));
   end;


   task type Simple_Task (Num : Integer; P : Integer)
   with CPU => 2, Priority => P is
      entry Start;
   end;


   task body Simple_Task is
      J : Integer := 0;
      Start_Time : CPU_Time;
      Now : CPU_Time;
   begin
      accept Start;
      Start_Time := Clock;
      Put_Line ("Task " & Integer'Image(Num) & " waiting: " & To_String(Start_Time));
      while Clock - Start_Time < Milliseconds (1500) loop
         null;
      end loop;
      Now := Clock;
      Put_Line ("Task " & Integer'Image(Num) & " ending : " & To_String(Now));
   end Simple_Task;

   T1 : Simple_Task(1,1);
   T2 : Simple_Task(2,2);
   begin
      T1.Start;
      T2.Start;
   end;
```