UE Systèmes temps réel/STR
Examen session 1, 26 mars 2018
Durée : 1h20
Tous documents autorisés

Singhoff Frank
singhoff@univ-brest.fr
C-203

NB : ce document est recto/verso.

Les deux exercices ne sont pas indépendants : il est conseillé de lire intégralement le sujet avant de traiter l'un ou l'autre des exercices.

Exercice n^o 1 : Ordonnancement temps réel (6 points)

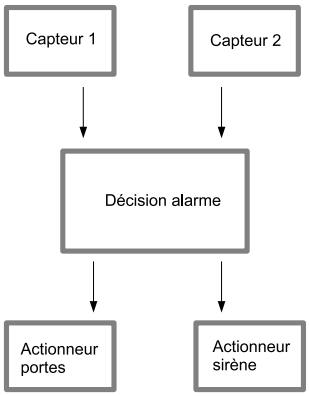

Figure 1 : Boitier alarme incendie

Dans cet examen, nous étudions un système permettant de gérer l'alarme incendie d'un bâtiment. Le système est composé de 5 tâches :

- Les deux tâches CAPTEUR_1 et CAPTEUR_2 contrôlent chacune un capteur de température.
- La tâche DECISION analyse les données des capteurs afin de déterminer la présence d'un incendie.
- Sur demande de la tâche DECISION, la tâche SIRENE pilote un dispositif audio permettant d'indiquer aux personnes présentes d'évacuer le bâtiment.
- Enfin, la tâche PORTE maintient les portes coupe-feu fermées lorsqu'un incendie est détecté.

Les paramètres des tâches sont donnés ci-dessous :

Nom	Période/échéance	Capacité
SIRENE	10 ms	2 à 3 ms
PORTE	10 ms	moins de 2 ms
DECISION	5 ms	normalement 0,5 ms et 1 ms dans de rares occasions
CAPTEUR_1	25 ms	1 ms
CAPTEUR_2	25 ms	3 ms

Nous supposons que toutes les tâches sont périodiques et indépendantes. Elles démarrent à l'instant zéro. Les échéances sont égales aux périodes.

Enfin, chaque tâche a une priorité fixe. Nous appliquons un ordonnancement préemptif à priorité fixe avec une affection des priorités selon Rate Monotonic.

Dans la suite de cet exercice, nous étudions l'ordonnancement de ce système, et en particulier sa capacité à respecter les échéances des tâches.

Question 1

Sans dessiner l'ordonnancement de ce jeu de tâches, vérifier si les tâches respecteront leurs échéances.

Question 2

— Dessiner sur la période d'étude l'ordonnancement de ce jeu de tâches. Attention, pour dessiner votre ordonnancement, n'utilisez pas de couleur afin de faciliter la correction de cette question.
— L'ordonnancement calculé confirme-t-il le résultat de la question 1 ? Expliquer.

Question 3

A partir du chronogramme calculé dans la question 2, déterminer les pires temps de réponse de chaque tâche.

Question 4

En pratique, les tâches ne sont pas indépendantes. En effet, les tâches CAPTEUR_1 et CAPTEUR_2 mémorisent des informations dans une mémoire partagée protégée par un sémaphore et donc accèdé en section critique. Cette mémoire partagée est également accédée en section critique par le tâche DECISION. Expliquer pourquoi les pires temps de réponse calculés dans la question 3 pourraient ne pas être respectés.

Exercice n^o 2 : Programmation concurrente avec Ada (7 points)

On se propose maintenant d'étudier une mise en oeuvre en Ada de l'alarme incendie. Cette mise en oeuvre est constituée du paquetage Ada *capteur* ainsi que de la procédure principale *alarme* donnés en annexe. Ce programme Ada est composé de cinq tâches : les tâches CAPTEUR_1, CAPTEUR_2, PORTE, DECISION et SIRENE.

Question 1

Le fichier *capteur.adb* ne compile pas car il existe une erreur dans cette unité de programme. Expliquer cette erreur et proposer une correction. On suppose que les procédures/fonctions $faire\_sonner\_la\_sirene$, $lire\_la\_temperature$ et $maintenir\_les\_portes$ sont connues et ne provoquent pas d'erreur.

Question 2

Les fichiers *capteur.ads/adb* et *alarme.adb* comportent trois erreurs concernant les tâches et leur synchronisation. Décrire deux de ces erreurs et proposer une correction.

Question 3

Le paquetage *capteur* permet de stocker des informations de type *integer*. On souhaite pouvoir appliquer ce paquetage à des types différents. Proposer une version générique du fichier *capteur.ads*. Justifier chaque élément ajouté dans *capteur.ads*. Comment doit-on modifier la procédure principale pour utiliser la version générique du paquetage *capteur*.

Annexe

```
package capteur is

  temperature_max : integer;

  procedure initialise(val : in integer);
  procedure affiche;
```

```
  task mutex is
    entry P;
    entry v;
  end mutex;

  task type un_capteur;

end capteur;
-----------------------------------------------------
with text_io; use text_io;

package body capteur is

  task body mutex is
  begin
    loop
      accept P;
      accept V;
    end loop;
  end mutex;

  task body un_capteur is
  lue : integer;
  begin
    loop
      lue:=lire_la_temperature;
      mutex.P;
      if lue>temperature_max
         then temperature_max:=lue;
      end if;
      mutex.V;
      delay 25.0;
    end loop;
  end un_capteur;

  procedure initialise(val : in integer) is
  begin
    temperature_max:=val;
  end initialise;

  procedure affiche is
  begin
    put("La temperature maximale est : ");
    put(temperature_max); new_line;
  end affiche;

end capteur;
-----------------------------------------------------
with capteur;
use capteur;

procedure alarme is
```

```
task type SIRENE;
task type PORTE;
task type DECISION;
CAPTEUR_1, CAPTEUR_2 : un_capteur;

alerte : boolean := false;

task body SIRENE is
begin
  loop
    if alerte
       then faire_sonner_la sirene;
    end if;
    delay 10.0;
  end loop;
end SIRENE;

task body PORTE is
begin
  loop
    if alerte
       then maintenir_les_portes;
    end if;
    delay 10.0;
  end loop;
end PORTE;

task body DECISION is
begin
  initialise(50);
  loop
    mutex.P;
    if temperature_max>300
      then alerte:=true;
    end if;
    mutex.V;
    delay 5.0;
  end loop;
end DECISION;

begin
  null;
end alarme;
```