Module STR
Examen session 1, 15 décembre 2016
Systèmes temps réel
Durée : 1h20
Tous documents autorisés

Singhoff Frank
singhoff@univ-brest.fr
C-202

NB : ce document est recto/verso.

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

Exercice n^o 1 : Ordonnancement temps réel (7 points)

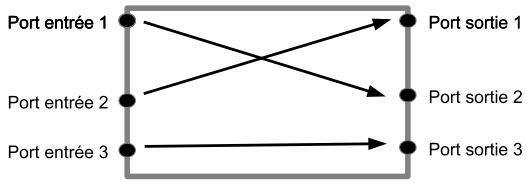

Figure 1 : commutateur

Dans cet examen, nous étudions un commutateur réseau. Le commutateur est décrit par la figure ci-dessus et est constitué :

— D'un ensemble de ports d'entrées et de sortie.
— Chaque port d'entrée et chaque port de sortie mémorise un message.
— Chaque port d'entrée et associé à un port de sortie où ses messages sont envoyés.
— Pour chaque port d'entrée, une tâche copie le message du port d'entrée vers le port de sortie associé toutes les 42 ms.
— Une tâche, dont le nom est STAT, collecte toutes les 21 ms des statistiques sur l'utilisation des ports d'entrée et de sortie.
— Enfin, une tâche de supervision, dont le nom est SUPER, permet aux utilisateurs de configurer l'équipement.

Les paramètres des tâches sont donnés ci-dessous :

Nom	Priorité	Période/échéance	Capacité
SUPER	1	7 ms	2 à 3 ms
STAT	2	21 ms	7 ms
PORT_1	3	42 ms	2 ms
PORT_2	4	42 ms	2 ms
PORT_3	5	42 ms	2 ms

Nous supposons dans un premier temps que le commutateur ne comporte que 3 ports d'entrée et 3 ports de sortie.

Les tâches STAT, PORT_1, PORT_2 et PORT_3 sont périodiques et indépendantes. Elles démarrent à l'instant zéro.

La tâche SUPER est activée à la demande de l'utilisateur et son traitement dure entre 2 et 3 ms selon la quantité de données à traiter. Toutefois, dans cet exercice, nous considérons cette tâche comme périodique.

Pour toutes les tâches, les échéances sont égales aux périodes.

Enfin, chaque tâche a une priorité fixe. Nous appliquons un ordonnancement préemptif à priorité fixe. Les niveaux de priorités vont de 0 à 255 sachant que le niveau de priorité le plus fort est 0. On applique ici 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

Bien que le traitement de la tâche SUPER soit événementiel, nous modélisons ce traitement comme un traitement périodique pour l'analyse d'ordonnancement. Pourquoi ?

Question 2

Sans dessiner l'ordonnancement de ce jeu de tâches, montrez que toutes les tâches respecteront leurs échéances.

Question 3

Dessiner sur la période d'étude l'ordonnancement de ce jeu de tâches. Cet ordonnancement confirme t il le résultat de la question 2 ?

Question 4

On souhaite définir autant de port d'entrée et de sortie que possible. Le facteur limitant est le nombre de tâches, car à chaque port i, il est nécessaire de lancer une tâche PORT_i dont la période/échéance est 42 ms et la capacité 2 ms.

Combien de port au maximum ce système peut-il supporter tout en respectant toutes les échéances des tâches ? Justifier votre réponse.

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

On se propose maintenant d'étudier une mise en oeuvre en Ada du commutateur. Cette mise en oeuvre est constituée du paquetage Ada *commutateur* donné en annexe. Ce paquetage comporte les cinq tâches SUPER, STAT, PORT_1, PORT_2 et PORT_3. On suppose donc un commutateur avec 3 ports d'entrée et 3 ports de sortie. Par ailleurs :

- Les ports d'entrée et de sortie sont implantés par des tableaux d'entiers : les messages sont donc des entiers dans cette mise en oeuvre.
- Lorsqu'un port ne comporte pas de donnée, son entré dans le tableau est initialisée à -1. C'est le cas initialement.
- La tâche STAT affiche périodiquement le nombre de port sans message en vérifiant que la valeur du port est différent de -1.
- On souhaite connecter les ports d'entrée vers les ports de sortie comme indiqué dans la figure 1. Par exemple, le port d'entrée 1 est connecté au port de sortie 2. Pour ce faire, chaque tâche PORT_i, avant de commencer à copier les messages en entrée vers sa sortie, attend via un rendez-vous Ada le numéro de port d'entrée et le numéro de port de sortie qui lui sont associés.

Question 1

Le paquetage *commutateur* ne compile pas car il existe une erreur dans cette unité de programme. Quelle est cette erreur ? Proposer une correction.

Question 2

Soit le programme *main.adb* suivant :

```
with commutateur; use commutateur;

procedure main is
begin
```

```
        -- Initialiser les ports
        -- avant le demarrage de l'application
        for i in 1..nb_port loop
                port_entree(i):=-1;
                port_sortie(i):=-1;
        end loop;
end main;
```

Avec le rendez-vous *numero_port*, compléter *main.adb* afin de connecter les ports d'entrée et de sortie conformément à la figure 1.

Question 3

Le paquetage *commutateur* comporte une erreur de synchronisation/concurrence. Quelle est cette erreur ? Proposer une correction.

Question 4

Le paquetage *commutateur* permet de stocker dans les ports des messages de type *integer*. On souhaite pouvoir appliquer ce paquetage à des types différents. Proposer une version générique du paquetage *commutateur*. Vous ne donnerez que le fichier *commutateur.ads*. Justifier les éléments ajoutés dans *commutateur.ads*.

Annexe

```
package commutateur is

  -- Port d'entree et de sortie
  nb_port : integer :=3;
  type message_port is array (1..nb_port) of integer;
  port_entree, port_sortie : message_port;

  -- Taches du commutateur
  task type PORT is
    entry numero_port(entree : in integer; sortie : in integer);
  end PORT;

  task SUPER;
  task STAT;
  PORT_i : array (1..nb_port) of port;

end commutateur;
-----------------------------------------------------
with text_io; use text_io;

package body commutateur is

  task body STAT is
  nb : integer;
  begin
        loop
           nb:=0;
           for i in 1..nb_port loop
                if port_entree(i)/=-1
```

```
                        then nb:=nb+1;
                end if;
                if port_sortie(i)/=-1
                        then nb:=nb+1;
                end if;
           end loop;
           put("Nombre de message actuellement = ");
           put(nb); new_line;
           delay 0.021;
        end loop;
  end STAT;

  task body PORT is
  s, e: integer:=0;
  begin
        accept numero_port(entree : in integer; sortie : in integer) do
                e:=entree; s:=sortie;
        end numero_port;
        loop
                delay 0.042;
                port_sortie(s):=port_entree(e);
        end loop;
  end PORT;

  -- Le code de la tache SUPER n'est pas donne dans cet exercice
  task body SUPER is ...
end commutateur;
```