Module STR
Examen session 1, 17 décembre 2015
Systèmes temps réel
Durée : 1h20
Tous documents autorisés

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

NB : ce document contient 2 feuilles recto/verso.

Exercice n^o 1 : Ordonnancement temps réel (8 points)

Cet exercice consiste à étudier une version simplifiée du système d'exploration de la mission Mars Pathfinder. En 1997, la mission Mars PathFinder a déployé le robot mobile Sojourner sur Mars. La mission de ce robot et de sa sonde était controlée par un logiciel multitâches s'exécutant sur un système VxWorks. Ce logiciel était constitué des tâches suivantes :

Nom	Priorité	Période/échéance	Capacité
ORDO_BUS	1	5 ms	1 ms
MESURE	3	20 ms	2 ms
PILOTAGE	4	40 ms	5 ms
METEO	5	80 ms	2 ms ou 4 ms

Toutes les tâches sont périodiques et démarrent à l'instant zéro. Les échéances sont égales aux périodes.

La tâche METEO est activée, parfois pour un traitement de 2 ms, parfois pour un traitement de 4 ms selon la quantité de données à traiter.

Sur le système d'exploitation VxWorks, l'ordonnancement est préemptif à priorité fixe. Les priorités indiquées ci-dessus sont des priorités VxWorks. Avec VxWorks, les niveaux de priorités vont de 0 à 255 sachant que le niveau de priorité le plus fort est 0.

Question 1

Dessiner sur la période d'étude l'ordonnancement de ce jeu de tâches. Les échéances des tâches sont elles respectées ?

Question 2

En fait, les tâches ORDO_BUS et METEO utilisent en exclusion mutuelle une ressource partagée durant toute leur capacité. Afin d'éviter les inversions de priorité, l'accès en exclusion mutuelle à cette ressource est implanté grâce à un sémaphore utilisant le protocole PIP (Priority Inheritance Protocol).

- Expliquer ce qu'est une inversion de priorité.
- Décrire le fonctionnement du procotole PIP.

Question 3

- Calculer l'ordonnancement de ce jeu de tâches en tenant compte cette fois-ci des accès à la ressource partagée avec PIP.
- Indiquer sur votre chronogramme quand la ressource est allouée et libérée.

Exercice n^o 2 : Programmation concurrente avec Ada (6 points)

On se propose d'étudier le paquetage Ada *redondance* donné en annexe. Ce paquetage est constitué de trois tâches permettant de lire deux capteurs :

- Les tâches *capteur*1 et *capteur*2 lisent une donnée entière depuis un altimétre.
- La tâche *voteur* reçoit les données de *capteur*1 et *capteur*2 et vérifie que les données sont identiques. Si les données sont différentes, cela indique qu'un capteur ne fonctionne pas correctement et le voteur affiche alors un message d'erreur. Dans le cas contraire, la donnée est affichée par le voteur.
- Le traitement ci-dessus est réalisé toutes les secondes.
- Enfin, le programme *main.adb* est un exemple d'utilisation de ce paquetage.

Question 1

Le paquetage *redondance* ne compile pas car il existe deux erreurs dans cette unité de programme. Quelles sont ces erreurs ? Proposer une correction pour chaque erreur.

Question 2

Le programme a été réalisé par un programmeur qui n'est pas un expert Ada. Ce programmeur souhaite que l'affichage réalisé dans la procédure principale soit toujours exécuté avant que les tâches *capteur*1 et *capteur*2 commencent leurs mesures. Or, le programmeur constate que parfois ce n'est pas le cas. Pourquoi ? Modifier ce programme afin de régler ce problème.

Question 3

Le paquetage *redondance* manipule les mesures grâce au type *data*. On souhaite pouvoir appliquer ce paquetage à des types différents. Proposer une version générique du paquetage *redondance* : vous ne devez donner que le fichier *redondance.ads*.

Annexe

```
with text_io; use text_io;
with redondance; use redondance;

procedure main is
begin
    put_line("Demarrage du capteur redondant");
    delay 10.0;
end main;
-----------------------------------------------------------
package redondance is

   type data is new integer range 0..integer'last;
   procedure lire_equipement(v : out data);

   task type capteur is
      entry emettre_valeur (val : out data);
   end capteur;

   capteur1, capteur2 : capteur;

   task voteur;
end redondance;
```

```
with text_io; use text_io;

package body redondance is

procedure lire_equipement(v : out data) is ...

task body capteur is
mon_val : integer;
begin
   loop
      lire_equipement(mon_val);
      accept emettre_valeur(val : out data) do
         val:=mon_val;
      end emettre_valeur;
      delay 1.0;
    end loop;
end capteur;

task body voteur is
val1, val2 : data;
begin
   loop
      capteur1.emettre_valeur(val1);
      capteur2.emettre_valeur(val2);
      if val1/=val2
         then put_line("Erreur capture donnees");
         else put(val1);
      end if;
   end loop;
end voteur;

end redondance;
```