UE Systèmes temps réel/STR
Examen session 2, 25 juin 2018
Durée : 1h20
Tous documents autorisés

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

NB : ce document est recto/verso.

Il est conseillé de lire intégralement le sujet avant de traiter les exercices.

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

Dans cette partie, on regarde l'ordonnancement d'un système d'air conditionné pour un hôtel.

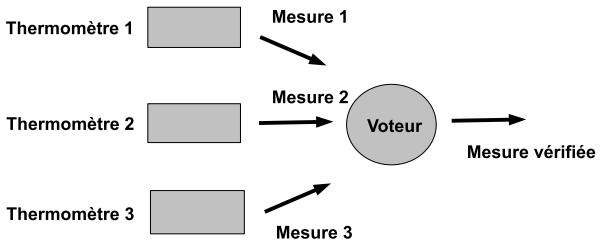
Ce dispositif est un boitier connecté à plusieurs ventilateurs. Un ventilateur est installé dans chaque chambre de l'hôtel et le boitier permet de régler de facon centralisée la température de toutes les pièces. Le boitier central comporte un processeur sur lequel plusieurs tâches s'ex'ecutent. On suppose que les tâches sont périodiques et que l'on utilise un ordonnancement à priorité fixe preemptif. Dans l'environnement utilisé, les niveaux de priorités varient de 0 à 255. 255 est le niveau le plus élevé.

Nom	Capacité (secondes)	Période (secondes)
Ventilateur1	1	30
Ventilateur2	1	30
Clavier	5	10
LCD	1	3

1. Le tableau ci-dessus présente les différents paramètres des tâches du système. On étudie le système pour 2 chambres/ventilateurs. Affectez une priorité fixe à chaque tâche. Expliquez comment vous affectez ces priorités.
2. Dessinez l'ordonnancement des tâches sur la période d'étude.
3. Montrez comment, **sans dessiner l'ordonnancement des tâches**, nous pouvions vérifier le respects des contraintes temporelles des tâches de cette application.
4. Combien de ventilateurs peut-on ajouter à ce jeu de tâches tout en respectant les échéances de toutes les tâches.

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

Dans cette partie, on étudie un système de redondance active. Afin de détecter les pannes d'équipements dans un système embarqué critique (ex : un avion), on utilise souvent la technique de la redondance active. Cette technique consiste à embarquer dans un système plusieurs équipements dont la fonction est identique afin de maintenir fonctionnel le système en cas de panne d'un de ces équipements.



Pour illustrer cette technique, la figure ci-dessous montre un système de mesure de température. Ce système comprend trois capteurs (thermomètres) et un "voteur" dont le rôle est de détecter la panne d'un capteur. Dans cet exemple, on considère comme panne, le cas où un thermomètre effectue une mesure erronée. Le principe de la redondance active est le suivant : chaque

capteur/thermomètre effectue une mesure, puis, les trois mesures sont comparées par le voteur. Si le voteur constate que les trois mesures sont identiques, alors il transmet l'information au reste du système. Lorsque les mesures sont différentes, on considère qu'un ou plusieurs thermomètres ne fonctionnent plus correctement. Dans ce cas, le voteur affiche à l'écran l'existence d'une mesure erronée, c-à-d d'une panne.

On souhaite simuler le fonctionnement d'un voteur en Ada avec le paquetage suivant :

```
generic
  type element is private;
  with function verifier_mesure(v1, v2, v3 : in element)
       return element;

package redondance is
  task type vote is
    entry donner_mesure1 (v : in element);
    entry donner_mesure2 (v : in element);
    entry donner_mesure3 (v : in element);
    entry recevoir_mesure_verifiee(v : out element);
  end vote;
end redondance;
```

La tâche `vote` de ce paquetage fonctionne de la façon suivante : elle attend successivement les températures sur les entrées `donner_mesure1`, `donner_mesure2` et finalement, `donner_mesure3`. Puis, la fonction `verifier_mesure` compare les trois informations et calcule la mesure à restituer au reste du système. Puis, elle attend sur son entrée `recevoir_mesure_verifiee` qu'une tâche demande la mesure préalablement vérifiée par la fonction `verifier_mesure`.

On teste ce paquetage avec la procédure principale suivante :

```
10  with text_io; use text_io;
20  with redondance;
30
40  procedure test_voteur is
50
60  function analyser_mesures(v1, v2, v3 : integer)
70    return integer is
80  begin
90    if (v1=v2) and (v1=v3)
100     then return v1;
110     else put_line("Mesure fausse : panne d'un thermometre !");
120   end if;
130 end analyser_mesures;
140
150
160
160 un_voteur : vote;
170 resultat : integer :=0;
180
190 begin
200   un_voteur.donner_mesure1(10);
210   un_voteur.donner_mesure2(10);
220   un_voteur.donner_mesure3(10);
230
240   un_voteur.recevoir_mesure_verifiee(resultat);
250   put("resultat = " ); put(resultat); new_line;
260 end test_voteur;
```

Questions :

1. La ligne numéro 250 ne compile pas. Pourquoi ? Corrigez cette erreur.
2. Dans la procédure principale, il manque l'instanciation du paquetage `redondance`. Sachant que la fonction de comparaison des mesures est `analyser_mesures`, ajoutez l'instanciation manquante.
3. Donnez le corps du paquetage `redondance`.