Année 2011 - 2012

INF342 - COU
Contrôle de Connaissances
28 juin 2012
3h00 - Sans document – Barème indicatif

Veuillez répondre à cette partie sur une feuille séparée

I. Ordonnancement (6 points)

On rappelle que $3*(2^{1/3} - 1) = 0,779$.

task Task1 **is**
  entry Start (T0 : Time);
end Task1;
	task Task2 **is**
	  entry Start (T0 : Time);
	end Task2;
		task Task3 **is**
		  entry Start (T0 : Time);
		end Task3;

task body Task1 **is**
  T : Time;
begin
  accept (T0 : Time)
  do T := T0; **end;**
  loop
  T := T + 5.0;
  Compute (2.0);
  delay T - Clock;
  end loop;
end Task1;
	task body Task2 **is**
	  T : Time;
	begin
	  accept (T0 : Time)
	  do T := T0; **end;**
	  loop
	  T := T + 15.0;
	  Compute (4.0);
	  delay T - Clock;
	  end loop;
	end Task2;
		task body Task3 **is**
		  T : Time;
		begin
		  accept (T0 : Time)
		  do T := T0; **end;**
		  loop
		  T := T + 3.0;
		  Compute (1.0);
		  delay T - Clock;
		  end loop;
		end Task3;

On considère une configuration de trois tâches périodiques et indépendantes comme décrite ci-dessus. La procédure Compute effectue un calcul dont la durée en seconde est indiquée comme paramètre d'entrée (on reprend ici la fonction utilisée en travaux pratiques).

Ordonnancement temps réel (2.5pts)

On souhaite appliquer l'ordonnancement Rate Monotonic Scheduling à ce groupe de tâches.

- Donner les périodes et temps de calcul des tâches. Donner le facteur d'utilisation U.
- Peut-on conclure que ce groupe de tâches est ordonnançable par Rate Monotonic Scheduling ? On appliquera le test d'ordonnançabilité et/ou le théorème de la zone critique. On détaillera le raisonnement dans les deux cas.

Ada pour le temps réel (1.5pts)

On souhaite que les tâches s'exécutent au niveau de priorité statique prévu par RMS. Les priorités s'étaleraient de Default_Priority + 1 .. Default_Priority + 3.

- Modifier le code ci-dessus pour que les tâches s'exécutent au niveau de priorité prévu.

Cependant le code ci-dessus pose problème en cas d'exécution dans un contexte temps réel.

- Identifier la construction inappropriée en justifiant et proposer une construction alternative.

Serveur de tâches apériodiques (2pts)

Rappeler le fonctionnement d'un serveur à scrutation et celui d'un serveur différé. Expliciter la différence entre ces deux serveurs.

Veuillez répondre à cette partie sur une feuille séparée

IV. Bus et Réseaux Temps Réel (1 point)

Question IV-1

Pourquoi le protocole d'accès au bus CAN est-il qualifié de *bitwise arbitration* (CSMA/BA) ? Justifier sur un exemple mettant en jeu trois sites.

V. Java Temps Réel (2 points)

Question V-1

Donner un schéma d'inversion de priorité impliquant le *garbage collector* et des threads temps réel.

Question V-2

Décrivez succinctement comment implémenter une tâche ordonnancée selon une politique RMS en utilisant RTSJ. Que doit faire la méthode `run` de cette tâche?

VI. Noyaux Temps Réel (1 point)

Question VI-1

Pendant l'exécution d'une application, la ressource mémoire peut être un facteur d'indéterminisme. Donner au moins trois précautions à prendre pour éviter cet indéterminisme.

Veuillez répondre à cette partie sur une feuille séparée

VII. Tolérance aux Fautes (3 pts)

Nous considérons un contrôleur de vitesse qui s'exécute périodiquement. A chaque activation, ce dernier lit un capteur de position et utilise la valeur mesurée lors de la période précédante pour évaluer sa vitesse courante. Cette vitesse est mémorisée dans une variable *vitesse*, et la valeur de la dernière postion mesurée est mise à jour dans la variable *position*. Le contrôleur exécute ensuite une fonction f dont l'entrée est la vitesse estimée et la sortie un nombre représentant l'accélération à appliquer pour maintenir la vitesse. Cette fonction accède de plus à la variable de la dernière postion sauvegardée.

Malheureusement f n'est pas correctement implémentée et peut : « boucler indéfiniment » ou « écraser à la valeur de la dernière position en la mettant à zéro ».
Nous faisons l'hypothèse qu'aucun mécanisme de tolérance aux fautes n'est intégré.

Question VII-1 (0,5 pts)

Définir brièvement ce qu'est une propriété de fiabilité et illustrer ce concept en donnant un exemple de contrainte de fiabilité sur f.

Question VII-2 (0,5 pt)

Rappeler la définition (en donnant la différence) des mécanismes de recouvrement arrière et avant.

Complément d'énoncé : on suppose que l'on détecte l'exécution en boucle de f en définissant une échéance relative à chaque activation de la tâche de contrôle (égale au tiers de la période). Et que

l'on détecte un écrasement de la position en contrôlant que la vitesse estimée ne s'écarte pas trop de sa valeur passée (accélération faible).

Question VII-3 (1 pt)
Expliquer pourquoi le recouvrement arrière est une solution viable si les défaillances de f ne sont pas déterministes.

Question VII-4 (1 pt)
En quoi la détection proposée couplée à un recouvrement arrière permettent-ils de mieux contrôler les défaillances du système complet (notamment dans le domaine temporel)

VIII. Méthodes formelles (2 pts)

On souhaite exprimer des exigences fonctionnelles contraignant le comportement d'un système réactif. Le système en question possède 3 état fonctionnels : process, comm, bascule. Ces états sont exclusifs (on ne peut occuper deux états à la fois). Ils représentent les trois états d'un satellite disposant de très peu de ressources de calcul.

- Process ; identifie l'état où le satellite est autonome et ne communique pas avec le sol.
- Comm : identifie l'état où le satellite communique avec une station de base
- Bascule : identifie l'état où le satellite réalise un changement de mode entre les deux états précédents.

Question VIII-1
Proposez en **justifiant** les expressions LTL correspondant aux phrases suivantes

- En tout instant, le satellite ne peut être que dans un seul état parmi (**process**, **comm**, **bascule**)
- Au cours de l'exécution, si le satellite est dans l'état **bascule** alors il ne peut y rester indéfiniment.
- Lorsque le satellite fonctionne en mode communication, il ne peut atteindre l'état **process** que si il entre dans l'état **bascule** auparavant.

Veuillez répondre à cette partie sur une feuille séparée

IX. Langage AADL (5 points) :

Question IX-1 (0,5 point)
Quelle est la différence principale entre un ***data port*** et un ***event data port*** en AADL ?

Question IX-2 (3 points)
Dans le modèle AADL suivant, nous avons intentionnellement inséré 3 erreurs sémantiques : le modèle est correct d'un point de vue syntaxique, mais certaines incohérences le rendent faux sémantiquement.

```
thread t1
features
  int: in event port;
properties
  period => 5 ms;
  priority => 2;
end t1;

thread t2
features
  int: in event port;
properties
  dispatch_protocol => periodic;
  period => 10 ms;
  priority => 5;
end t2;

thread t3
features
  int: out event port;
properties
  dispatch_protocol => periodic;
  period => 15 ms;
  priority => 4;
end t3;
```

```
process proc
end proc;

process implementation proc.impl
subcomponents
  th1: thread t1;
  th2: thread t2;
connections
  cnx: port th1.int -> th2.int;
end proc.impl;

processor cpu
properties
  scheduling_protocol =>
Rate_Monotonic_Scheduling;
end cpu;

system s
end s;

system implementation s.impl
subcomponents
  c: processor cpu;
  p: process proc;
properties
  actual_processor_binding =>
reference(c) applies to p;
end s.impl;
```

Question IX-3 (1,5 points)
Représenter l'arbre d'instances obtenu après instanciation du système ***s.impl***. Chaque nœud de l'arbre représente une instance, chaque arc représente le lien de contenant/contenu entre instances. Pour chaque instance, donner son type de référence (*type* ou *implémentation* en AADL). L'instance de ***s.impl*** sera nommée **root**.