Examen 2016-2017 Temps Réel

Le 22/06/2017, Tout document autorisé. Tout dispositif électronique interdit.
Durée : 1 heure et demie

Pour avoir des points, il faut détailler sa réponse, même si elle est correcte. Une mauvaise réponse bien étayée peut rapporter des points. Ecrire lisiblement et en bon français.

1. Pour quelles actions a-t-on besoin d'un fonctionnement temps réel pour une machine de bureau ? Indiquez les plus importantes en premier.
2. Dans quels cas ne peut-on pas être temps réel ?
3. Quels sont les composants matériels de l'ordinateur qui influencent le calcul du temps du pire cas ?
4. Si l'on a 8 tâches et 8 coeurs, est-on temps réel ?
5. Quelle est la difficulté pour faire du temps réel avec un langage interprété comme Lisp, Java, Python...
6. Pourquoi est-il plus facile de faire un système temps réel quand il n'y a pas d'interruptions ?
7. Peut-on faire un système temps réel s'appuyant uniquement sur les interruptions et non un ordonnanceur ?
8. Dans quels cas est-on "obligé" de faire des *threads* ?
9. Dans quels cas n'a-t-on vraiment pas besoin de verrous pour faire fonctionner le RCU ?
10. Que ne garantit pas le compilateur C quand vous écrivez :

    ```
    extern volatile int a, b, c ;
    c = a + b ;
    d = b ;
    ```

11. Ecrire la fonction Lustre qui commande le moteur d'une porte automatique comportant les boutons 'ouvrir', 'fermer' et un senseur de proximité qui déclenche l'ouverture de la porte. Pour des raisons de sécurité l'ouverture est toujours prioritaire.
    `node door(open_button, close_button, sensor:bool) returns(open:bool);`
12. Ecrire la fonction Lustre permettant de prouver avec Lesar que les consignes de sécurité de la porte sont vérifiées dans tous les cas.
13. On met une forte priorité à une tâche qui a un période longue. Comment prouver que l'ensemble des tâches est ordonnançable ?
14. Dans quels cas l'approche de programmation asynchrone pure (sans lancer d'autres processus) n'est pas utilisable ?
15. Dans quels cas l'approche de programmation asynchrone pure permet de gagner en temps d'exécution ?
16. Si l'on ajoute la notion de priorité à un programme asynchrone, est-il possible de le faire fonctionner en temps réel ?
17. Dans quels cas l'approche de programmation asynchrone pure nécessite d'utiliser des verrous ?

1. Pour quelles actions a-t-on besoin d'un fonctionnement temps réel pour une machine de bureau ? Indiquez les plus importantes en premier.

   Lire la souris et afficher le curseur sur l'écran. Lire le clavier et afficher l'action sur l'écran. Tous les autres feedback utilisateur (donc le son l'image pour les jeu interactifs)

2. Dans quels cas ne peut-on pas être temps réel ?

   Quand on dépend de quelque chose d'extérieur : disque dur, réseau. Ou bien si les composants matériels ne sont pas fiables.

3. Quels sont les composants matériels de l'ordinateur qui influencent le calcul du temps du pire cas ?

   Vitesse des calculs. Caches mémoire. Mémoire. Pipeline du processeur.

4. Si l'on a 8 tâches et 8 coeurs, est-on temps réel ?

   Oui s'il n'y a aucune zone critique et que les tâches prennent chacune moins de 100% du temps CPU

5. Quelle est la difficulté pour faire du temps réel avec un langage interprété comme Lisp, Java, Python...

   Le garbage collector et éventuellement des fonctions dont le temps de calcul est imprévisible.

6. Pourquoi est-il plus facile de faire un système temps réel quand il n'y a pas d'interruptions ?

   Parce que le calcul du pire cas est simplifié (le cache n'est pas vidé). Parce qu'il n'y a pas besoin de protéger les zones critiques puisque l'on est pas interrompu.

7. Peut-on faire un système temps réel s'appuyant uniquement sur les interruptions et non un ordonnanceur ?

   Si le traitement d'une interruption ne peut être interrompu par une autre : non car c'est aléatoire. Et si cela peut être interrompu, alors il ne faut pas de zones critiques car personne ne pourrait relacher le verrou, sauf si les tâches tournent chacune sur leur propre coeur.

8. Dans quels cas est-on "obligé" de faire des *threads* ?

   Jamais. Eventuellement pour ne pas être bloqué par une fonction synchrone ou pour utiliser plusieurs coeurs.

9. Dans quels cas n'a-t-on vraiment pas besoin de verrous pour faire fonctionner le RCU ?

   Un seul écrivain.

10. Que ne garantit pas le compilateur C quand vous écrivez :

    ```
    extern volatile int a, b, c ;
    c = a + b ;
    d = b ;
    ```

    Que la valeur de `d` *est modifiée après celle de* `c` *et que si l'entier déborde vers les positifs on obtienne une valeur négative.*

11. Ecrire la fonction Lustre qui commande le moteur d'une porte automatique comportant les boutons 'ouvrir', 'fermer' et un senseur de proximité qui déclenche l'ouverture de la porte. Pour des raisons de sécurité l'ouverture est toujours prioritaire.
    `node door(open_button, close_button, sensor:bool) returns(open:bool);`

    Une manière possible d'écrire :
    `open = true -> open_button or sensor or pre(open) and not close_button`

12. Ecrire la fonction Lustre permettant de prouver avec Lesar que les consignes de sécurité de la porte sont vérifiées dans tous les cas. *Une manière possible d'écrire :*
    ```
    ok = door(open_button, close_button, sensor) or not (sensor or open_button);
    assert(ok) ;
    ```

13. On met une forte priorité à une tâche qui a un période longue. Comment prouver que l'ensemble des tâches est ordonnançable ?

    On ne peut pas appliquer le rate monotonic, mais par contre la méthode de la zone critique fonctionne. Il faut évidemment que la tâche avec une période longue soit avec une capacité minuscule pour que la tâche à petite période ne rate pas son échéance.

14. Dans quels cas l'approche de programmation asynchrone pure (sans lancer d'autres processus) n'est pas utilisable ?

    Seulement quand on a besoin de faire du parralélisme. Sinon, une tâche qui prend du temps peut contenir plein de points de préemptions afin que des tâches interactives puissent tourner. (c'est comme cela que la majorité des applications web sont faites.)

15. Dans quels cas l'approche de programmation asynchrone pure permet de gagner en temps d'exécution ?

    Dans le cas non interactif car chaque tâche fera son travail en une seule fois et donc cela maximise l'utilisation des caches et de la mémoire.

16. Si l'on ajoute la notion de priorité à un programme asynchrone, est-il possible de le faire fonctionner en temps réel ?

    Oui, mais il faut ajouter à la capacité de chaque tâche le plus grand temps d'exécution des tâches qui sont moins prioritaires qu'elle. Elle ne pourra pas les interrompre pour démarrer.

17. Dans quels cas l'approche de programmation asynchrone pure nécessite d'utiliser des verrous ?

    Quand on a introduit un point de préemption dans une zone critique.