

Examen - Systèmes temps réel (1h)

M1 IN 2013-2014

Les documents de cours ne sont pas autorisés, sans calculatrice

1 Question de cours

Question 1. Donner la définition d'un système temps réel.

Question 2. Donner la définition des caractéristiques suivantes dans le cadre d'un système temps réel : déterminisme, prédictibilité et fiabilité. Vous préciserez également la traduction des termes en anglais.

2 Ordonnancement

Rappels – $W = \sum \frac{C}{T}$;
– $U(1)=1$, $U(2)=0.83$, $U(3)=0.78$ ($=46.8/60$).

Exercice 2.1 : Ordonnancement de travaux périodiques

Soit un ensemble de travaux périodiques A , B et C :

- A doit s'exécuter pendant 1 secondes toutes les 3 secondes ;
- B doit s'exécuter pendant 1 secondes toutes les 4 secondes ;
- C doit s'exécuter pendant 2 secondes toutes les 6 secondes (échéance relative = 3).

Toutes les travaux sont prêts à l'instant $t = 0$.

Question 3. Donner le facteur d'utilisation du processeur (W) et la valeur de la période d'étude pour cette configuration de tâches.

Question 4. Donner le plan d'ordonnancement du système avec EDF et LLF. Préciser le non respect d'une échéance s'il y a lieu et les priorités des travaux sur les intervals concernés.

Exercice 2.2 : Ordonnancement de travaux périodiques

Soit un ensemble de travaux périodiques X , Y et Z :

- X doit s'exécuter pendant 2 secondes toutes les 5 secondes ;
- Y doit s'exécuter pendant 1.5 secondes toutes les 4 secondes ;
- Z doit s'exécuter pendant 0.5 secondes toutes les 3 secondes.

Tous les travaux sont prêts à l'instant $t = 0$.

Question 5. L'algorithme RMS est-il optimal ? Peut-il exister un ordonnancement possible suivant l'algorithme RMS ? Justifiez vos réponses. Les réponses à ces questions doivent se faire sans aucun plan d'ordonnancement.

Question 6. Donner le plan d'ordonnancement suivant l'algorithme RMS (jusqu'à la première faute temporelle si vous en trouvez une).

Question 7. Déterminer l'augmentation de vitesse minimale du processeur pour que RMS devienne optimal. Aucun plan d'ordonnancement n'est demandé.

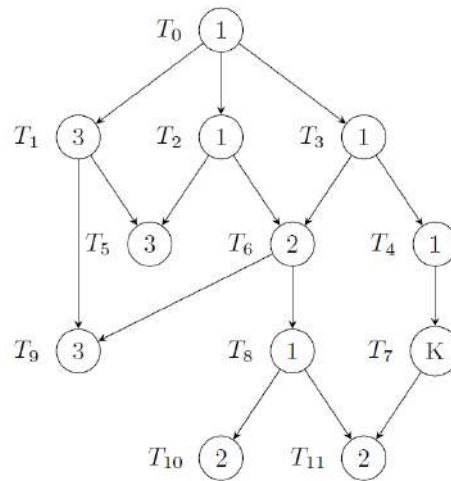
Question 8. Les algorithmes EDF et LLF sont-ils optimaux ? Aucun plan d'ordonnancement n'est demandé.

Exercice 2.3 : Ordonnancement multiprocesseur avec graphe de dépendances

Il s'agit d'ordonner et d'allouer les tâches d'un système temps-réel multiprocesseur. Les tâches sont dépendantes et de durées d'exécution constantes exprimées dans la même unité de temps (cf figure ci-dessous). On dispose d'un ordinateur à N processeurs identiques à mémoire partagée. Les temps de commutation de tâches et les temps de communications pour échanger les données sont nuls. La préemption des traitements est autorisée. À l'instant 0, la tâche T_0 est prête à s'exécuter.

Question 9. Sachant que $K > 0$ et que l'on souhaite que les tâches puissent se terminer en 7 unités de temps que pouvez-vous dire sur le nombre de processeurs N ? Justifiez votre réponse.

Question 10. Le nombre de processeur est maintenant fixé à 4. Pour quelles valeurs de k toutes les tâches peuvent être terminées en 7 unités de temps ? Justifiez votre réponse (méthode du cours). Pour la plus grande valeur possible de k , montrez un plan d'allocation et d'ordonnancement statique possible.



(Les notations sont les mêmes que celles employées dans le cours).

3 Synchronisation de processus

Question de cours

Squelette de programme des processus A et B :

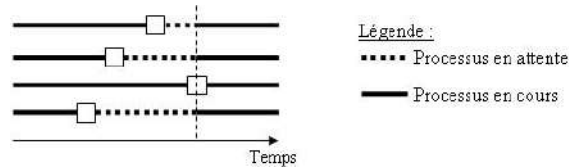
A	B
P(sem1)	P(sem2)
P(sem2)	P(sem1)
...	...
V(sem2)	V(sem1)
V(sem1)	V(sem2)

Question 11. Quel problème peut survenir lors de l'exécution des processus A et B ?

Question 12. Proposez deux solutions différentes pour résoudre ce problème. Quels sont les nouveaux squelettes de programme de chaque processus pour chacune des solutions ?

Exercice 3.2 : Barrière de synchronisation pour N processus

Une barrière de synchronisation permet de garantir qu'un certain nombre de processus ait passé un point spécifique. Ainsi, chaque processus arrivant sur cette barrière devra se mettre en attente jusqu'à ce que tous les processus soient arrivés au niveau de cette barrière. La figure ci-dessous illustre le fonctionnement de la barrière de synchronisation pour 4 processus.



Question 13. Implantez un mécanisme de barrière de synchronisation entre N processus à l'aide de sémaphores et/ou mutex. Un squelette de programme vous est fourni en figure 13.

Notes : – Notez que certaines cases du squelette de programme fourni peuvent être vides et certaines peuvent contenir plusieurs lignes de code si le besoin s'en fait sentir ;
– La variable globale « `nb_process` » représente le nombre restant de processus à synchroniser. Elle est initialisée à N .

```
void barriere(void)
{
    int i = 0;
    
    nb_process--;
    
    if(nb_process == 0)
    {
        for(i=0 ; i<N ; i++)
        {
            
        }
        nb_process = N;
    }
    
}
```