FONDEMENTS DES SYSTEMES EMBARQUES TEMPS-REEL

SPECIALITE SAR, UNIVERSITE PARIS VI

Barème indicatif, **sans document**
Les téléphones portables doivent être éteints et rangés dans vos sacs.

Les parties de l'examen sont indépendantes.

Il sera tenu compte de la présentation et de la clarté dans la rédaction.
Seules les réponses précises et justifiées seront considérées

1 Ordonnancement temps réel (7 pts)

1.1 Ordonnancement de tâches périodiques.

On souhaite tester une configuration où les tâches sont réparties sur deux processeurs A et B. Les tâches sont définies par les paramètres suivants :

Nom	Processeur	Capacité	Période
T1	B	4	6
T2	B	2	20
T3	A	2	3
T4	A	2	5
T5	A	2	5

Les tâches sont ordonnancées selon un algorithme à priorité fixe et on souhaite que les priorités soient affectées selon l'algorithme Rate Monotonic. On utilise un système d'exploitation temps réel compatible POSIX 1003 et offrant des niveaux de priorités allant de 0 à 255, 0 étant le niveau le plus élevé.

1.1.1 Configuration des priorités.

Proposez une priorité pour chacune des tâches ci-dessus.

1.1.2 Ordonnancement de la configuration.

Démontrez qu'il existe au moins une des tâches qui ne respecte pas ses contraintes temporelles.

1.1.3 Nouvelle configuration.

L'affectation des tâches aux processeurs A et B a été réalisée au hasard. En fait, toutes les tâches de notre système peuvent s'exécuter indifféremment sur le processeur A **ou** sur le processeur. Cependant, les processeurs diffèrent par leur rapidité d'exécution. Le processeur B est 2 fois plus rapide que le processeur A. Affectez les tâches aux processeurs A et B de sorte que les contraintes temporelles de toutes les tâches soient satisfaites.

1.2 Ordonnancement de tâches apériodiques.

On considère une configuration de tâches périodiques et indépendantes ordonnancée par Rate Monotonic

	Calcul	Période
Tâche 1	3	10
Tâche 2	2	5

On ajoute à cette configuration deux tâches apériodiques.

	Calcul	Activation
Tâche 3	3	7
Tâche 4	4	11

On va traiter ces tâches apériodiques à l'aide de quatre techniques :

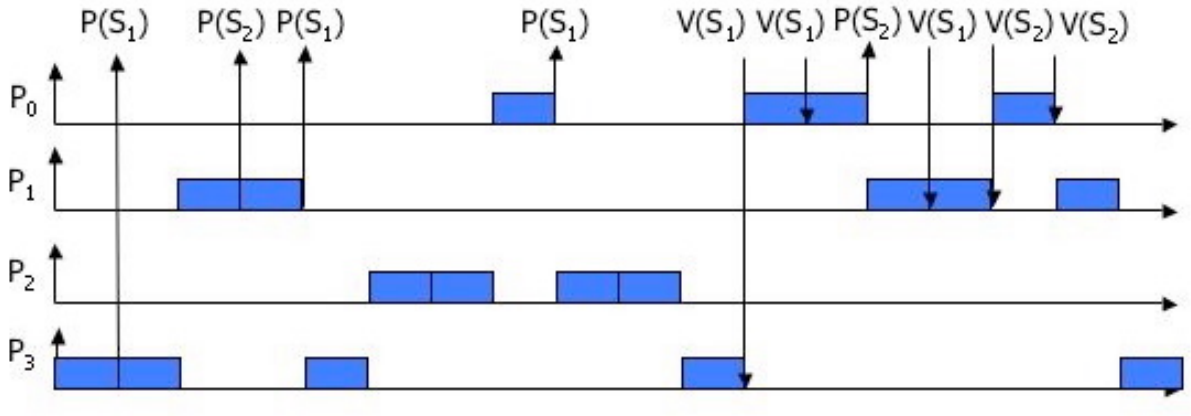
1. Serveur de fond
2. Serveur de scrutation
3. Serveur différé
4. Serveur sporadique

Lorsque nécessaire, on considérera une capacité du serveur de 1 et une période de 4.

- Décrire graphiquement l'enchaînement des tâches dans ces quatre cas. Il sera particulièrement tenu compte des indispensables explications décrivant le fonctionnement du serveur.

1.3 Mécanismes de synchronisation.

On considère l'ordonnancement ci-dessous qui illustre le comportement de quatre processus périodiques. Ces processus ont des priorités déterminées par RMS avec priorité (P0) > priorité (P1) > priorité (P2) > priorité (P3). Chaque période de processus commence à sa date S et termine à sa date T. Les quatre processus partagent deux ressources protégées par des sémaphores S1 et S2. Redessiner le diagramme d'exécution pour illustrer le comportement des processus dans le cas où de l'héritage de priorité est employé. Justifier votre diagramme.



2 Langage Java (2 pts)

- Donnez un scénario d'inversion de priorité impliquant des threads de type temps réel et le ramasse-miettes (garbage collector). Justifiez.
- Comment l'éviter ?

3 Bus et réseaux (3 pts)

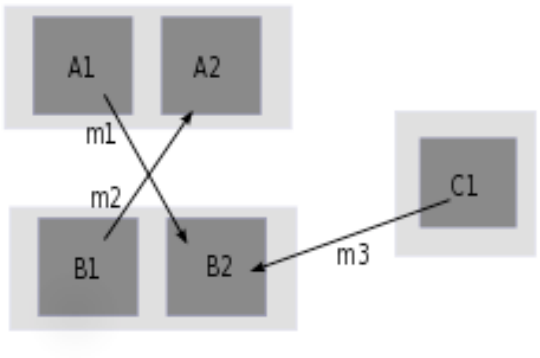
Soit un jeu de tâches s'exécutant sur trois sites :

- Les tâches A1, A2 sur le site A,

- Les tâches B1, B2 sur le site B,
- La tâche C 1 sur le site C.

A1, B1 et C1 sont prêtes au temps t=0. A2 et B2 sont déclenchées sur réception de messages :

- A2 attend le message m2 émis par B1,
- B2 attend les messages m1 émis par A1 et m3 émis par C1



Les messages sont envoyés en fin de tâche.

(*notation : E(T) veut dire échéance de la tâche T, Tk est le délai de transmission du message k)*

Données d'ordonnancement :

- A1, B1 et C1 ont la même échéance E1,
- E(B2) telle que : E(B2) – E1 >= Tm1 + Tm3 + WCET(B2)
- E(A2) telle que :
- E(A2) – E1 >= Tm1 + Tm2 + Tm3 + WCET(A2)

3.1.1 Ordonnancement sans contrôle

On suppose ici qu'il n'y a pas d'ordonnancement pour accéder au réseau : une tâche prête à émettre prend l'accès au medium, et s'il est libre, envoie son message.

Quelles doivent être les contraintes sur les WCET des tâches A1, B1 et C1 (*on notera WCET(T) le WCET de la tâche T*) pour que toutes les échéances soient respectées ? Justifier en donnant un schéma illustrant un scenario dans lequel les échéances ne sont pas respectées.

3.1.2 Ordonnancement avec bus CAN

On **suppose maintenant que le bus est du type CAN**, et que l'on donne :

- WCET(B1) = WCET(C1) = E1 et WCET(A1) < E1,

Indiquer précisément les conditions que doivent satisfaire les tâches et les messages pour que toutes les échéances soient respectées.

4 Noyaux temps réel – POSIX (3 pts)

On veut écrire une fonction qui fournit un mécanisme de rendez-vous à N tâches. On utilisera une variable conditionnelle et un compteur entier N.

- Donner un squelette du programme qui implémente cette fonction. Justifiez en détails.
- Quel est l'intérêt d'utiliser cette fonction dans le cadre d'un ordonnancement temps réel ?

  Indication (extrait du ' man') : ***pthread_cond_broadcast*** *relance tous les threads attendant sur la variable condition cond. Rien ne se passe s'il n'y a aucun thread attendant sur cond.*
- Comment limiter le temps d'attente ? Modifier très simplement le pseudo-code.

5 Tolérance aux fautes (5pts)

5.1 Traitement des erreurs.

Les deux affirmations suivantes concernent une application logicielle s'exécutant sur un réseau de machines interconnectées par un réseau sûr de fonctionnement. Dites si l'affirmation est vraie ou fausse en justifiant votre choix.

1) Les fautes du logiciel peuvent être tolérées grâces à l'utilisation de la diversification des implémentations (1 service – N implémentations) tant que les erreurs qu'elles engendrent ne se propagent pas au matériel.
2) La réplication passive augmente la disponibilité d'un système, et la réplication active sa fiabilité.

5.2 Analyse d'un exemple.

Nous nous intéressons au logiciel permettant de réguler la vitesse d'une voiture autour d'une valeur d'équilibre choisie par le conducteur (description simplifiée non réaliste).

Lorsqu'il est en fonction, le régulateur doit maintenir la vitesse courante malgré les variations de la pente de la route (cause principale de variation de vitesse). La voiture est conçue de telle sorte que l'accélération du moteur est commandée numériquement à travers la lecture périodique d'une variable partagée entre le contrôle moteur et le régulateur de vitesse.

La vitesse de la voiture est transmise au régulateur qui met à jour la valeur de l'accélération pour maintenir la vitesse autour de la valeur souhaitée. Toutefois, la valeur proposée par le régulateur est soumise à une vérification avant écriture dans la mémoire partagée : l'écart entre deux valeurs consécutives doit rester inférieur à une valeur prédéfinie A.

Soit a_1 la dernière valeur proposée et a_2 la valeur recommandée par le régulateur, alors la valeur inscrite dans la mémoire partagée est :

$$a_3 = a_1 + signe(a_2-a_1) * max\ (A, |a_2-a_1|)$$

Avec *signe* la fonction qui retourne le signe d'un nombre (positif si nul), *max* la fonction maximum, et |.| la fonction qui retourne la valeur absolue d'un nombre.

1) Définissez dans un cadre général les notions de *sécurité-innocuité* et *composant auto-testable*.

2) Sur l'exemple :
   a. Proposez un scénario d'exécution du système permettant d'illustrer la mise en défaut d'une propriété de sécurité-innocuité par rapport au conducteur ?
   b. En quoi peut-on considérer que le régulateur de vitesse est un composant auto-testable ?

3) Définissez en général les notions de réplications active et passive. Dites pourquoi la réplication active est applicable ici pour assurer la tolérance aux fautes du régulateur. (indice : nature de l'exécution)