Master 1 - FST - Dpt. Info - 2018-2019
Systèmes Temps-Réel / MIF18
L. Gonnord & T. Excoffier
http://laure.gonnord.org/pro/teaching/systemeTR.html

Mini TD Ordonnancement Temps réel ou pas

Juste deux exercices d'application directe du cours, d'après un TD de F. Singhoff, univ Brest, avec l'aimable autorisation de l'auteur.

1 Ordonnancement Temps Réel

1.1 Rappels

Ordonnancement à priorité fixe avec affection de priorité RM Le calcul de priorité consiste à associer à chaque tâche une priorité fixe inversement proportionnelle à sa périodicité (Rate Monotonic).

La phase d'élection consiste à élire la tâche de plus forte priorité (ordonnancement à priorité fixe).

skip Tests de l'ordonnançabilité (**Hypothèses : tâches indépendantes et périodiques. Algorithmes préemptifs.**) :

1. Test sur le taux d'utilisation avec $\forall i : D_i = P_i : \sum_{i=1}^{n} \frac{C_i}{P_i} \leq n(2^{(1/n)} - 1)$ condition suffisante mais non nécessaire.
2. Utilisation de la période d'étude : ordonnancement à comportement cyclique (propriété du modèle périodique). Période d'étude = $[0, PPCM(P_i)]$ (si $\forall i : S_i = 0$).

L'algorithme EDF Le calcul de priorité consiste à déterminer une échéance. L'échéance $D_i(t)$ d'une tâche i à l'instant t est égale à la somme de la date de début de l'activation courante à l'instant t et du délai critique D_i.

La phase d'élection consiste à élire la tâche de plus *proche* échéance.

Tests d'ordonnançabilité (**Hypothèses : tâches indépendantes et périodiques. Algorithmes préemptifs.**) :

1. Test sur le taux d'utilisation :
   - Cas $\forall i : D_i = P_i : \sum_{i=1}^{n} \frac{C_i}{P_i} \leq 1$ condition nécessaire et suffisante.
   - Cas $\exists i : D_i < P_i : \sum_{i=1}^{n} \frac{C_i}{D_i} \leq 1$ condition suffisante seulement, $\sum_{i=1}^{n} \frac{C_i}{P_i} \leq 1$ condition nécessaire uniquement.
2. Utilisation de la période d'étude (propriété du modèle périodique).

Exercice 1 : ordonnancement à priorité fixe + Rate Monotonic

Soient trois tâches périodiques T1, T2 et T3 définies par les paramètres suivants : $S_1 = S_2 = S_3 = 0, P_1 = 29, C_1 = 7, P_2 = 5, C_2 = 1, P_3 = 10, C_3 = 2$. On suppose un ordonnancement à priorité fixe avec une affection Rate Monotonic des priorités. Les délais critiques sont égaux aux périodes (soient $\forall\ i\ :\ D_i = P_i$).

1. Calculez le taux d'utilisation pour RM. Le jeu de tâches est il ordonnançable ?
2. Dessinez, sur les 30 premières unités de temps, l'ordonnancement généré par RM, d'abord avec la version préemptive, puis, avec la version non préemptive (vous commencerez à la date zéro). Que constatez vous ?
3. Nous modifions la tâche T1 par $P_1 = 30$ et $C_1 = 6$ et la tâche T2 par $C_2 = 3$. Le jeu de tâches est dit "harmonique". En effet, chaque période du jeu de tâches est multiple des autres périodes. Refaites le test d'ordonnançabilité par le taux d'utilisation. Puis, dessinez de nouveau l'ordonnancement généré par RM sur sa période d'étude (mode préemptif). Que constatez vous ?

4. Confirmez ce dernier résultat en calculant le temps de réponse de chaque tâche.

Exercice 2 : EDF

Soient trois tâches périodiques T1, T2 et T3 définies par les paramètres suivants : $S_1 = S_2 = S_3 = 0, P_1 = 12, C_1 = 5, P_2 = 6, C_2 = 2, P_3 = 24, C_3 = 5$. Les délais critiques sont égaux aux périodes (soient $\forall\ i\ :\ D_i = P_i$).

1. Calculez le taux d'utilisation du processeur. Concluez sur l'ordonnançabilité du jeu de tâches.
2. Déterminer le nombre d'unités de temps libre sur la période d'étude (le ppcm des périodes, ie 24) (sans dessiner d'ordonnancement). On trouvera une formule liant ce nb à la période d'étude et le taux d'utilisation.
3. Confirmez les points précédents en dessinant, sur la période d'étude, l'ordonnancement généré par EDF, d'abord avec la version préemptive, puis, avec la version non préemptive.
4. On considère maintenant le même jeu de tâches mais cette fois ci, deux tâches apériodiques TA_1 et $T1_2$ arrivent respectivement aux instants 7 et 12. Leurs capacités sont de 1 et 3 unités de temps. Leurs échéances $D_i(t)$ interviennent aux instants 9 et 21. Le jeu de tâches est il ordonnançable par EDF (mode préemptif) ? Dessinez l'ordonnancement sur les 30 premières unités de temps.

Exercice 3 : RM avec tâches apériodiques

Soient deux tâches périodiques définies par les paramètres suivants : $S_1 = S_2 = 0, P_1 = 15, C_1 = 4, P_2 = 7, C_2 = 1$. Les délais critiques sont égaux aux périodes (soient $\forall\ i\ :\ D_i = P_i$). L'ordonnancement est effectué avec un algorithme à priorités fixes et RM (mode préemptif).

On souhaite faire cohabiter des tâches apériodiques avec les tâches définies ci-dessus. Pour ce faire, on utilise la méthode du serveur par scrutation. Cette méthode consiste à dédier à une tâche périodique (tâche dite "serveur par scrutation") l'exécution des tâches apériodiques. A chaque activation du serveur par scrutation, celui-ci regarde si des tâches apériodiques sont arrivées **avant le réveil du serveur** (les tâches apériodiques doivent évidemment être prêtes). Le cas échéant, le serveur attribue du temps processor à concurrence de la capacité du serveur : l'exécution d'une tâche apériodique peut être donc répartie sur plusieurs activations du serveur. Le temps de traitement nécessaire au serveur pour vérifier si des tâches apériodiques sont présentes est supposé comme nul : si le serveur est réveillé alors qu'aucune tâche apériodique n'est présente, alors, le serveur ne consomme pas de ressource processeur. La capacité du serveur est d'une unité de temps. Cette unité de temps est donc uniquement consacrée à l'exécution des tâches apériodiques.

1. On suppose que le serveur par scrutation possède une période de 5 unités de temps. Le serveur arrive dans le système à l'instant zéro. Le jeu de tâches est il ordonnançable ?
2. On suppose maintenant que deux tâches apériodiques TA1 et TA2 arrivent respectivement aux instants 7 et 12. Leurs capacités sont de 1 et 3 unités de temps. Leurs échéances interviennent aux instants 9 et 21. Dessinez l'ordonnancement généré par le serveur par scrutation sur les 26 premières unités de temps.

Exo 4 : comparaison EDF/autre algo

Soient deux tâches T1 et T2 définies par les paramètres suivants : $S_1 = S_2 = 0, P_1 = 9, P_2 = 8, C_1 = 4, C_2 = 3$. Les délais critiques sont égaux aux périodes (soient $\forall\ i\ :\ D_i = P_i$).

On se propose d'étudier un nouvel algorithme d'ordonnancement dynamique : LLF (Least Laxity First). Ce dernier effectue l'élection des tâches prêtes grâce à leur laxité. La laxité, comme l'échéance, est une information dynamique qui évolue dans le temps. La laxité $L_i(t)$ d'une tâche i à l'instant t s'évalue par $L_i(t) = D_i(t) - reste(t)$ où $reste(t)$ est le reliquat de capacité à exécuter pour l'activation courante.

1. Dessinez l'ordonnancement généré par EDF sur les 20 premières unités de temps (en mode préemptif).
2. Dessinez l'ordonnancement généré par LLF sur les 20 premières unités de temps (en mode préemptif).
3. Que constatez vous ? Conclure sur le choix entre ces deux algorithmes.

2 Ordonnancement avec Posix 1003.1b

Exercice 5 : Gestion des files d'attente

Dans cet exercice, nous utilisons l'implantation des spécifications POSIX 1003.1b sur Linux. Sur Linux, il existe 100 niveaux de priorité : le niveau de priorité 0 est réservé à SCHED_OTHER et les niveaux de priorité 1 à 99 aux politiques SCHED_FIFO et SCHED_RR. Les tâches de priorité 99 sont les tâches de plus forte priorité. SCHED_OTHER est dédiée à l'ordonnanceur temps partagé. Le quantum utilisé par la politique SCHED_RR est d'une unité de temps. Pour simplifier, nous supposons que la politique SCHED_OTHER alloue le processeur de façon inversement proportionnel au temps processeur consommé par les tâches. L'ordonnancement est préemptif.

Nom	Capacité	Date d'arrivée	Priorité	Politique
other1	8	0	0	SCHED_OTHER
rr1	5	2	3	SCHED_RR
rr2	5	2	3	SCHED_RR
fifo1	3	4	5	SCHED_FIFO
fifo2	3	3	2	SCHED_FIFO
fifo3	2	4	5	SCHED_FIFO

Soit le jeu de tâches apériodiques ci-dessus. On suppose qu'une fois arrivée, les tâches sont toujours prêtes. Dessinez de l'instant 0 à l'instant 25, l'ordonnancement généré par l'ordonnanceur POSIX 1003.1b.