

INFO 019 :
TECHNIQUES AVANCÉES DES SYSTÈMES D'EXPLOITATION
TUYAUX VERSION 0.2

Année Académique 2005-2006

Voilà ceci est le deuxième jet pour les tuyaux d'OS1. Ce document n'a donc pas vraiment fait l'expérience du temps et surtout, les questions ici récoltées ne sont issues que de deux sessions.

Il y a sûrement d'autres questions qui ont été posées et qui ne se trouvent donc pas dans ce document.

Une des choses que l'on peut affirmer sans trop de craintes est qu'il faut connaître son cours sur le bout des doigts et qu'il est aussi très important de savoir donner des exemples.

Je rappellerai aussi que, comme toujours, ce document n'est en rien officiel et qu'il peut donc y avoir des erreurs. Si c'est le cas, veuillez envoyer vos remarques à : licence@cerkinfo.be.

Dernièrement, n'hésitez pas à faire part de vos questions/réponses et donc de mettre à jour ce document.

Auteurs :

2004-2005 : Gigo

2005-2006 : Stewball

1 Borne inférieure d'utilisation

1.1 Questions

1. Bien expliquer l'étape 1 (insister sur les Δ)
2. Bien expliquer l'étape 2 (insister sur les Δ)
3. Donner sans détail l'étape 3, on ne veut que le principe

1.2 Réponses

1. Cf cours
2. Cf cours
3. Minimisation grâce à la dérivée

2 Priorité fixe, départ simultané et échéance sur requête

2.1 Questions

1. Soit $\tau_i = (C_i, D_i, T_i)$ avec $\tau_1 = (1, 3, 3)$ et $\tau_2 = (4, 6, 6)$. Ordonnancer ce système avec RM
2. Jusqu'à quel instant faut-il ordonnancer ?
3. Donner les conditions d'ordonnancement ?
4. Lesquelles sont nécessaires/suffisantes ?
5. Donner une autre condition d'ordonnancement
6. Comment calculer ce temps de réponse ?
7. Quand est-ce que l'itération s'arrête ?
8. Si l'itération ne s'arrête jamais, cela veut dire quoi ? Dans quels cas cela se produit ?

2.2 Réponses

- 1.
2. Il ne faut regarder que les premiers travaux des tâches
3. Cf cours
4. Cf cours
5. Temps de réponse de la tâche $i \leq D_i \quad \forall i$
6. De façon itérative : Cf cours
7. Cf cours
8. $U(\tau_i) > 1$

3 Priorité fixe, départ simultané et échéance arbitraire

3.1 Questions

1. Donner un exemple de la propriété spéciale dans ce cas (Temps de réponse pas spécialement le plus long lors du premier travail)
2. Expliquer pourquoi ce n'est pas toujours lors du premier travail que le temps de réponse est le plus long.
3. Définir : intervalle d'étude, période élémentaire et période d'activité de niveau- i
4. Démontrer que le temps de réponse le plus long a lieu dans l'intervalle $[0, \lambda_i)$
5. Donner le théorème qui nous donne l'intervalle d'étude.
6. Donner la formule.
7. Cette formule converge-t-elle toujours ? Si non dans quelle cas.
8. Quelle est la borne maximale si l'utilisation est égale ou inférieure à 1 et pourquoi.

3.2 Réponses

1. Exemple cf cours ou inventé.
2. la justification est que comme c'est départ simultané, on n'a pas de jobs plus anciens qui peuvent être actifs (pour la même tâche) et interférer alors.
3. Cf cours
4. Cf cours
5. Cf cours (découle de la démo)
6. Cf cours
7. Oui sauf si l'utilisation est supérieure à 1 : pas d'instantanés oisifs \rightarrow intervalle énorme
8. C'est P car quand on arrive en P, on remet toutes les tâches au même niveau (instant initial) et s'il reste des tâches non terminées, ça veut dire qu'il reste du boulot à fournir, ça veut donc dire que l'utilisation est supérieure à 1, dans les autres cas on est sûr d'avoir au moins dépassé λ_n .

4 Priorité fixe, départ différé et échéances contraintes

4.1 Questions

1. Définir "Viable au niveau de priorité le plus faible"
2. Définir "Optimalité"
3. Définir "Intervalle d'étude"
4. Expliquer et démontrer l'optimalité d'Audsley
5. Le fait de mettre τ_i au niveau de priorité le plus faible libère combien d'unités de calcul pour intercaler les tâches qui étaient moins prioritaires que τ_i ? (elles sont promues d'un niveau de priorité vu qu'on assigne à τ_i la priorité la plus faible)
6. Donner l'algorithme
7. Si l'algorithme s'arrête sans passer par *return echec* ça veut dire quoi ?
8. Comment vérifier en pratique la condition du if de l'algorithme ? (c'est-à-dire comment vérifier si une tâche est viable au niveau de priorité le plus faible ?)
9. Quel est l'intervalle d'étude pour vérifier qu'une tâche est viable au niveau de priorité le plus faible ?
10. Cet intervalle est-il ouvert ou fermé ?
11. Comment calculer le S_i pour cet intervalle ?
12. Donner la complexité de l'algorithme
13. Démontrer et expliquer que les ordonnancements sont périodiques
14. Extrapoler ces conclusions dans les autres cas

4.2 Réponses

1. Cf cours
2. Cf cours
3. Cf cours
4. Cf cours
5. Intuitivement c'est C_i . Mais il se peut très bien que plusieurs travaux de τ_i apparaissent dans l'intervalle que l'on étudie et donc la réponse est : nombre de fois que τ_i apparait dans l'intervalle * C_i
6. Cf cours
7. Cf cours
- 8.
9. $[0, S_n + P)$
10. ?
11. Cf cours
12. $O(n^2)$ car $n + (n - 1) + \dots + 1 = \frac{n(n + 1)}{2}$
13. Cf cours
14. Cf cours

5 EDF

5.1 Questions

1. Démontrer et expliquer l'optimalité
2. L'ordonnancement par EDF est-il fini
3. Jusqu'à quel point est-ce à priorité dynamique
4. Quelles sont les conditions pour qu'un système périodique à départ différé et échéance arbitraire soit ordonnançable avec EDF ?
5. En quoi est ce fondamentalement différent d'avoir deux processeurs de vitesse 1 plutôt qu'un seul processeur de vitesse 2.

5.2 Réponses

1. Cf cours
2. ??
3. Cf cours
4. Cf cours et savoir bien justifier l'intervalle $[0, O_{max} + 2P]$, pourquoi le 2 et pourquoi cette condition n'est pas suffisante.
5. Pour partitionnement : exemple trois tâches : deux dont U vaut 0,9 et une dont U vaut 0,2. Sur les processeurs de vitesse 1, ce n'est pas ordonnançable(trivial). le U est bien entendu divisé par 2 pour le processeur deux fois plus rapide.
→ Pas ordonnançable sur multiprocesseur avec partitionnement, mais ordonnançable sur le processeur unique 2 fois plus rapide.
Pour les stratégies globales : le problème vient simplement du fait qu'une tâche ne peut pas être exécutée sur deux processeurs en même temps. (en fait ce problème vaut aussi pour les techniques de partitionnement)
Si on permet à une tâche d'être exécutée sur deux processeurs en même temps (impossible dans la pratique hein) alors il n'y a aucune différence

6 LLF

6.1 Questions

1. Expliquer LLF

2. Définir les grandeurs utilisées dans LLF
3. Représenter $L_j(t)$ sur un schéma
4. $L(t) = 0$?
5. Comment évolue la latence pendant l'exécution du travail?
6. Optimalité forte / rapport avec DM (si optimal avec DM \geq optimal avec LLF? si optimal avec LLF \geq optimal avec DM?)
7. Expliquer le phénomène de trashing par un raisonnement général
8. Donner 3 types d'algorithmes au niveau des priorités et expliquer
9. Comparer LLF et EDF
10. Dans quel cas LLF pourrait être plus intéressant que EDF
11. Soit $\tau_i = (C_i, D_i, T_i)$ avec $\tau_1 = (1, 2, 3)$ et $\tau_2 = (4, 5, 6)$. Ordonner ce système avec LLF ou justifier pourquoi il n'est pas ordonnable
12. Soit $\tau_i = (C_i, D_i, T_i)$ avec $\tau_1 = (1, 3, 3)$ et $\tau_2 = (4, 5, 6)$. Ordonner ce système avec LLF

6.2 Réponses

1. Cf cours
2. Cf cours
3. Cf cours
4. ?
5. De manière constante (Cf cours)
6. ?
7. Cf cours
8. Cf cours
9. Cf cours
10. Dans le cas des systèmes à multiprocesseurs
- 11.
- 12.

7 Inversion de priorité

7.1 Questions

1. Poser le problème d'inversion de priorité et montrer que le fait d'interdire la préemption des sections critiques le résout
2. Donner les conditions d'ordonnabilité dans ce cas là et expliquer ce que représente q
3. La formule est-elle suffisante pour conclure de l'ordonnabilité du système?
4. Est ce que le fait d'interdire la préemption tout court est une bonne solution?
5. Expliquer le protocole par héritage de priorité et donner les problèmes qui demeurent
6. Expliquer le protocole à priorité plafond
7. Donner les conditions d'ordonnabilité dans ce cas là et expliquer ce que représente S_i
8. Dans le protocole à héritage de priorité que se passe-t'il (en terme d'inversion de priorité) si une tâche τ_0 plus prioritaire que τ_1 arrive dans le système et veut la même ressource? (voir schéma slide 88 je crois)
9. Comparer les valeurs q et B_i des deux formules

7.2 Réponses

1. Cf cours et savoir donner un exemple
2. Cf cours
3. Non car cette formule envisage le cas le plus pessimiste. Il se pourrait très bien qu'une tâche ne rentre pas dans une section critique ou bien qu'elle soit bloquée pendant une durée plus petite que q . Donc un système qui est ordonnançable pourrait ne pas l'être selon la formule
4. Cf cours
5. Cf cours
6. Cf cours
7. Cf cours
8. τ_3 hérite de la priorité de τ_0 et τ_1 devra patienter plus longtemps
9. La valeur de B_i peut être plus grande car si, comme dans le cas précédent, une tâche τ_0 apparaît dans le système, alors l'attente pour τ_1 est augmentée

8 Tâches apériodiques

8.1 Questions

1. Définir une tâche apériodique
2. Expliquer le polling server
3. Donner un exemple d'ordonnancement avec le polling server
4. Donner une condition d'ordonnançabilité avec le polling server
5. Comment améliorer le Polling Server
6. Expliquer le deferrable server
7. Expliquer le priority exchange server
8. Est-ce que la condition d'ordonnançabilité du deferrable server est la même et que celle du polling server ? Même question pour le priority exchange server
9. Mettre en parallèle le polling server et le background server

8.2 Réponses

1. Cf cours
2. Cf cours
3. Cf cours
4. $U_{totale} \leq 0,69$ dans le cas de RM (cf Cours)
5. Rajout d'un Back Ground Server à la technique du Polling Server et parler des autres types de servers
6. Cf cours
7. Cf cours
8. Le test d'ordonnançabilité ne marche plus pour le deferrable server car la tâche ajoutée n'est plus périodique. Par contre ca marche pour le priority exchange server
9. Le background server est un cas particulier du polling server, c'est-à-dire un polling server avec la tâche périodique ajoutée ayant la priorité la plus faible

9 Multiprocessing

9.1 Questions

1. Expliquer ce qu'est le multiprocessing
2. Donner les hypothèses
3. Expliquer les deux familles d'algorithmes

4. Expliquer le Bin Packing
5. Montrer qu'il n'y a pas d'algo optimal en ligne pour les ordonnancements globaux
6. Si on est dans un système strictement périodique, le raisonnement précédent est-il toujours valable ?
7. Expliquer les anomalies
8. Donner un exemple d'anomalies
9. Savoir ordonnancer avec DM 3 tâches dont on reçoit les valeurs sur 2 processeurs
10. Pourquoi est ce que dans le cours on ne s'est intéressé qu'aux tâches périodiques et pas aux tâches sporadiques ?
11. Ce dernier raisonnement est vrai pour les systèmes monoprocesseur mais est-ce que c'est toujours le cas pour les systèmes multiprocesseur ?

9.2 Réponses

1. Cf cours
2. Cf cours
3. Cf cours
4. Cf cours
5. Cf cours
6. Non car dans le raisonnement précédent on crée 2 futurs différents or si on a un système strictement périodique on a qu'un seul futur possible.
7. Cf cours
8. Cf cours
- 9.
10. Considérer des tâches périodiques correspond à considérer le pire cas et donc si c'est ordonnançable pour des tâches périodiques c'est ordonnançable avec des tâches sporadiques
11. Non car il peut y avoir des anomalies