Examen 2002/2003 Temps Réel

Le 27/06/2003, Tout document autorisé

Durée : une heure

Recopier tel quel des extraits de documents n'apporte jamais de points. Vous devez répondre aux questions qui vous sont posées.

Vous devez argumenter (étayer) vos réponses.

Quand on parle d'*ordonnanceurs* il faut répondre à la question pour tout ceux que vous connaissez.

Chaque question vaut 1 point, sauf celle-ci qui ne compte pas et pour laquelle vous indiquez votre choix sur la copie : Une fois que j'aurais terminé de noter les copies, voulez vous que (A) les notes ne soient pas modifiées, (B) que j'applique une formule à l'ensemble de notes pour remonter la moyenne, formule qui dépendra de l'histogramme des notes.

1. À quoi sert un ordonnanceur ?
2. Dans quels cas un ordonnanceur est inutile?
3. En TP, avec RTLinux, quel ordonnanceur avez-vous utilisé?
4. Dans un processus temps réel, quel algorithme de tri utiliseriez-vous ?
5. Donnez des exemples de tâche pour lesquels il est impossible de calculer la capacité.
6. Que peut-on conclure quand il y a un dépassement de capacité alors qu'elle a été correctement calculée ?
7. Si l'on se trouve dans le cas précédent, que faut-il faire ?
8. Les quantums de temps sont-ils nécessaires aux ordonnanceurs ?
9. À quels moments les ordonnanceurs se "réveillent-t-ils" pour travailler ?
10. Doit-on calculer la capacité de l'ordonnanceur ?
11. Quel doit-être la priorité d'un serveur de tâche sporadique ?
12. Sur une machine bi-processeurs, faut-il un ou deux ordonnanceurs ?
13. Dans quels cas une échéance a-t-elle un sens pour la lecture d'un bloc d'information venant d'un disque ou du réseau ?
14. Quel sont les conditions nécessaires à l'apparition d'une inversion de priorité ?
15. Soit une fonction `void f(int i) { ... }` qui exécute une action fonction de la valeur de `i`.

Valeur de `i`	Probabilité que `i` prenne cette valeur	Action à exécuter
0	99%	A0
1	0.9%	A1
2	0.09%	A2
3	0.01%	A3

Écrire le pseudo-code de la fonction `f` dans le cadre d'une utilisation temps-réel, mais aussi dans le cadre d'une utilisation normale. **Vous n'utiliserez pas de `switch` ou de tableaux, seulement des `if`.**

16. Deux processus périodiques utilisant au total 1% du temps CPU sont-ils toujours ordonnançables ?
17. Les capacités sont-elles changées par une augmentation de la taille du cache du processeur ?
18. Un système généraliste peut-il faire du multimédia ?
19. Donner des exemples de contrôle de processus physiques pour lesquels une échéance doit-être respectée.
20. Même question que la précédente mais l'échéance est calculée par le processus de contrôle.

Un corrigé

La réponse non notée :

A	1/38	La meilleur copie
B	20/38	
?	1/38	Réponse ambiguë
	16/38	Pas de réponse

J'ai ajouté 3 points à toutes les copies.

1. Un ordonnanceur sert à partager la ressource processeur. *Il ne sert pas à synchroniser les tâches, mais peut faciliter la synchronisation*
2. Quand il n'y a qu'une seule tâche ou bien que l'on a un ordonnancement statique.
3. HPF (avec quantum de temps et tourniquet pour quelques-uns qui ont expérimenté).
4. Celui avec le meilleur pire cas, de préférence une implémentation non récursive et sans allocation mémoire. Trie fusion, *heap-sort*, ... *Mais pas le* quick sort*, même en mélangeant aléatoirement les données, car il est possible (mais improbable) que le mélange trie les données.*
5. Les tâches faisant des entrée-sortie, des interactions avec l'utilisateur ou bien effectuant des calculs non déterministes, utilisant des heuristiques, ...
6. La capacité est intrinsèque à la tâche, elle n'est pas modifiée par l'exécution des autres tâches. Le fait que le système soit surchargé ne modifie pas la capacité des tâches. La capacité est le temps processeur nécessaire pour faire le calcul dans le pire cas en tenant aussi compte des verrous. Donc si la capacité est dépassée, cela peut vouloir dire que : le programme est buggé, le compilateur est buggé ou qu'il y a un problème matériel.
7. Détruire la tâche incriminée avant qu'elle ne perturbe les autres tâches temps réel. On peut imaginer lancer une tâche de remplacement ou une recherche de perturbations matérielles.
8. Le quantum de temps est la durée affectée à une tâche pour s'exécuter. Dans les systèmes classiques le quantum à une durée fixe. Il est petit afin que l'utilisateur est l'impression que toutes les tâches s'exécutent simultanément. En temps réel, ce quantum n'est pas nécessaire car l'illusion de l'avancement simultané des tâches ne nous intéresse pas. Le seul cas ou il peut être utile est dans le cas de tâches qui ont la même priorité. Mais même dans ce cas, introduire un quantum de temps ralentie l'éxécution de la tâche à cause des changements de contexte.
9. Quand une tâche est suspendue (elle demande à s'endormir pour attendre quelque chose ou elle a terminée), quand une tâche est lancée ou bien que la priorité d'une tâche change, quand une interruption matérielle prioritaire arrive. Evidemment quand l'ordonnanceur laisse la main à une tâche, il programme l'horloge (RTC) pour reprendre la main lors du prochain changement de contexte qui est donc calculé à l'avance.
10. Il faut la calculer. Mais l'ordonnanceur n'est pas une tâche comme les autres, car il se réveille très souvent pour décider de ce qui doit être fait. On peut par exemple décider d'ajouter la capacité de l'ordonnanceur à la capacité de chacune des tâches en fonctionnement.
11. Un serveur sporadique pour contenir des tâches très prioritaires. On doit donc lui donner une priorité forte, mais il ne doit pas pertuber les autres tâches temps réel. On résoud ce problème en lui allouant un quantum de temps par période, ce quantum est en fait une capacité. Quand le quantum est écoulé, le serveur continue avec la priorité la plus faible.
12. Si l'on prend 2 ordonnanceurs, c'est comme si l'on avait deux machines et donc on ne "gagne" rien. Il faut donc gérer le passage des processus d'un ordonnanceur à l'autre, c'est compliqué et loin d'être optimal. Si l'on a un seul ordonnanceur, il faut le modifier pour faire tourner 2 tâches à la fois, ce n'est pas compliqué. Par contre, il ne pourra pas lancer 2 tâches simultanément. Il faudrai le modifier pour qu'il soit réentrant. Comme c'est compliqué on ne le fait pas (sauf pour des systèmes avec beaucoup de processeurs).
13. Je ne vous demande pas la capacité de la lecture mais son échéance. Cette échéance ne sert pas à calculer la capacité d'un autre processeur. L'échéance est défini par la tâche qui a besoin de la donnée. L'échéance a donc un sens si la tâche utilisant la donnée lui en donne un, par exemple l'affichage d'une video, si la donnée n'est pas là à temps, on rate une image.

14. Avec un ordonnanceur préemptif, quand une tâche faiblement prioritaire possède (verrou) une donnée utile à une tâche très prioritaire et qu'elle est interrompu par une tâche de priorité intermédiaire.
15. Seul 2 bonnes réponses sur 38 à cette question. Aucune réponse exacte (j'ai trouvé la réponse exacte en écrivant ces lignes). La solution présentée est détaillée, je ne vous en demandais pas tant. Je vous détaille les réponses que vous m'avez le plus souvent données. Je simplifie en disant que tester par rapport à 0 est aussi rapide que tester par rapport à une autre valeur.

Fonction	Temps min	Temps moyen moins : 0.99*A0 + 0.009*A1 + 0.0009*A2 + 0.0001*A3	Temps max

if (i == 0) A0
if (i == 1) A1
if (i == 2) A2
if (i == 3) A3
	4
	+ min(A0
	, A1
	, A2
	, A3)
		4	4
			+ max(A0
			, A1
			, A2
			, A3)

if (i == 0) A0 else if (i == 1) A1 else if (i == 2) A2 else A3	min(1+A0 , 2+A1 , 3+A2 , 3+A3)	0.99 + 0.009*2 + 0.0009*3 + 0.0001*3	max(1+A0 , 2+A1 , 3+A2 , 3+A3)
if (i < 2) if (i == 0) A0 else A1 else if (i == 2) A2 else A3	2 + min(A0 , A1 , A2 , A3)	2	2 + max(A0 , A1 , A2 , A3)
if (i == 3) A3 else if (i == 2) A2 else if (i == 1) A1 else A0	min(3+A0 , 3+A1 , 2+A2 , 1+A3)	0.99*3 + 0.009*3 + 0.0009*2 + 0.0001	max(3+A0 , 3+A1 , 2+A2 , 1+A3)

Dans le cadre d'une utilisation normale, il faut minimiser le temps moyen, c'est trivialement le deuxième programme qui fait cela et ceci quelque soit la durée des actions. Dans le cadre d'une utilisation temps réel. il faut minimiser le temps max, si les actions ont la même durée, c'est le troisième programme qui est le bon, sinon si $A0$ prend beaucoup plus de temps que les autres, c'est le deuxième qui est bon. Le premier programme est simplement mauvais. Le quatrième programme souvent proposé pour le temps réel est simplement plus lent en moyenne (il est juste néanmoins si la tache A3 est la plus longue).

16. Si leurs échances est égal à leurs périodes, oui grace à la *rate monotonic*. Sinon, on peut toujours trouver une échéance rendant impossible l'ordonnancement.
17. Non, quand on calcule la capacité on prend toujours le pire cas, c'est-à-dire celui ou rien n'est dans le cache.
18. Oui, car ce n'est pas important si il y a quelques imperfection dans le rendu de l'image ou du son.
19. Si dans votre TP de contrôle vous preniez trop de temps pour calculer l'état de l'aimant, la bille aurait eu un problème.
20. Cette question est difficile. Il faut trouver un processus physique ou la vitesse intervient. Voici une réponse possible : Un système anti-missile surveille le ciel, quand il repère un missile il calcule sa vitesse. Une fois la vitesse connue, il peut calculer le temps maximum de réaction (échéance) avant que cela soit trop tard. Ensuite, il calcule la trajectoire des anti-missiles. On peut imaginer avoir une tâche par missile à détruire.