istic Informatique Électronique

Année scolaire 2017/2018

Informatique temps-réel (ITR)

Travaux dirigés

istic Informatique Électronique

Année scolaire 2017/2018

Politiques d'ordonnancement temps-réel

Exercice 1. Ordonnancement de tâches périodiques

On considère une application composée des trois tâches périodiques synchrones dont les caractéristiques (pire temps d'exécution, échéance relative, période) sont données dans le tableau ci-dessous. On se pose le problème de trouver un ordonnancement qui respecte les échéances de terminaison des tâches.

Tâche	**Pire temps d'exécution (Ci)**	**Echéance relative (Di)**	**Période (Pi)**
T1	2	5	6
T2	2	4	8
T3	4	8	12

Généralités

1. L'exécution de tout ensemble de tâches périodiques est cyclique, le cycle étant nommé *hyperpériode*. Quel est la durée de l'hyperpériode ?
2. Calculer la charge processeur demandée par chaque tâche, ainsi que la charge globale. Qu'en déduisez vous ?

On dispose pour l'exécution de l'application d'un système multi-tâche préemptif utilisant un ordonnancement à priorités. On considère dans un premier temps l'utilisation d'un ordonnancement à priorité fixe, puis d'un ordonnancement à priorité dynamique.

Ordonnancement à priorités fixes

3. Quel est l'ordonnancement à priorité fixe optimal dans ce cadre ?
4. Donner un schéma temporel d'exécution sur l'hyperpériode : (a) en utilisant un ordonnancement RM, (b) en utilisant un ordonnancement DM
5. Est-ce que le système respecte ses contraintes de temps avec un algorithme à priorité fixe ?

Ordonnancement à priorités dynamiques

6. Quel est l'ordonnancement à priorité dynamique optimal dans ce cadre ?
7. Donner un schéma temporel d'exécution sur l'hyperpériode : (a) en utilisant un ordonnancement EDF, (b) en utilisant un ordonnancement LLF
8. Est-ce que le système respecte ses contraintes de temps avec un algorithme à priorité dynamique ? Quel algorithme à priorités dynamiques vous semble le plus adapté pour l'application exemple ?

Ordonnancement hors-ligne non préemptif

Connaissant précisément les dates d'arrivée des tâches, on se propose de générer avant exécution (hors-ligne) un calendrier de lancement des tâches.

9. Est-ce possible ici de respecter les contraintes de temps avec un tel ordonnancement ? Justifier

Mise en œuvre sur un système d'exploitation temps-réel

L'application est destinée à être exécutée sur un microcontrôleur embarqué dans une automobile. Des contraintes de portabilité des applications vous imposent l'utilisation du système d'exploitation OSEK/VDX, utilisant un ordonnancement préemptif à priorités non modifiables.

10. Avec ces contraintes, que proposez vous pour que votre application respecte ses contraintes de temps ?

Exercice 2. Ordonnancement à priorités et surcharge

On considère une application composée des trois tâches périodiques synchrones dont les caractéristiques (pire temps d'exécution, échéance relative, période) sont données dans le tableau ci-dessous.

Tâche	**Temps d'exécution (Ci)**	**Echéance relative (Di)**	**Période (Pi)**
T1	1	2	2
T2	1	4	4
T3	2	8	8

1. Quel est l'algorithme d'ordonnancement optimal dans ce cadre ?
2. Quelle est la charge par tâche et globale ?
3. Dessiner un schéma temporel d'ordonnancement sur l'hyperpériode pour les ordonnancements DM et EDF

Faute d'utilisation de méthode adaptée, l'estimation du pire-temps d'exécution de la tâche T3 est fausse, l'exécution de la tâche T3 prend 4 unités de temps au lieu de 2.

4. Quel est l'impact sur les schémas temporels du c) ?
5. Quel est le taux d'échéances ratées dans les deux cas ? (sur un intervalle de temps de 3 hyperpériodes)

Systèmes d'exploitation temps-réel

Exercice 1. Utilisation des événements et des alarmes OSEK/VDX

Soient T1 et T2 deux tâches périodiques de période identique (500ms).

1. Donner le code permettant à une tâche extérieure de lancer l'exécution de T1 et de T2 périodiquement et indéfiniment, (a) en utilisant l'appel système SetAbsAlarm et une valeur de paramètre *cycle* non nulle, (b) avec une valeur de *cycle* nulle. Reprendre (a) et (b) en utilisant maintenant SetRelAlarm. Comparer les solutions proposées.

Soient p1 (resp p2) deux points dans le code de ces tâches.

2. Donner le code permettant à T2 d'attendre que T1 ait franchi son point p1 avant de poursuivre son exécution en p2 :

- En utilisant les événements OSEK/VDX
- En utilisant des sémaphores à compteur standard
- NB : Le code doit fonctionner quelles que soient les priorités et les dates de démarrage respectives de T1 et T2 (on supposera toutefois qu'elles terminent de s'exécuter avant l'exemplaire suivant de la tâche). Le code doit bien sûr fonctionner quel que soit le nombre d'exécutions des tâches (et pas seulement à leur première exécution)

3. Reprendre la question précédente en codant un rendez-vous plutôt qu'une attente

Exercice 2. Mise en œuvre des sémaphores d'exclusion mutuelle

En utilisant les notations prises dans le cours (*active* pour la tâche active, *insérer* et *retirer* pour les gestions de file), donner le code de sémaphores d'exclusion mutuelle tels qu'ils ont été définis en cours :

- On vérifiera que le P et le V sont faits par le même processus
- On permettra les sections critiques emboîtées

Exercice 3. Inversion de priorités, héritage de priorités

L'objectif de cet exercice est de visualiser le problème d'inversion de priorités sur un exemple un peu plus conséquent que les exemples examinés en cours, et de montrer sur cet exemple l'intérêt des protocoles d'héritage de priorités. Les tâches sont ordonnancées grâce à un algorithme d'ordonnancement préemptif à priorités fixes, la tâche T5 étant la plus prioritaire.

Soit le jeu de tâches suivant, dans lequel [R ; D] indique que la ressource R est utilisée pendant la durée D. [R ; D [R',D']] indique que l'on utilise R' pendant D' unité de temps, sans relâcher la ressource R :

Tâche	**Arrivée Ai**	**WCET Ci**	**Ressources critiques**

T5	7	3	1 ; [gris;1] ; 1
T4	5	3	1 ; [noir;1] ; 1
T3	4	5	5
T2	2	7	1;[gris;2 [noir;1]2];1
T1	0	6	1;[noir;4];1

Sans héritage de priorité

1. Dessiner un chronogramme sans protocole d'héritage de priorité
2. Schématiser les durées de blocage de toutes les tâches (zones d'inversion de priorité)

Protocole PIP

3. Reprendre le chronogramme précédent en utilisant PIP
4. Est-ce que l'on observe toujours le phénomène d'inversion de priorité ?
5. Schématiser les durées de blocage de toutes les tâches
6. Que se passe t'il si T1 demande la ressource « gris » après 4 unités de temps ?

Protocole PCP

7. Reprendre le chronogramme précédent en utilisant PCP à la place de PIP. Indiquer sur le schéma les priorités plafond des ressources, ainsi que la priorité plafond courante en fonction du temps.
8. Est-ce que l'on observe toujours le phénomène d'inversion de priorité ?
9. Schématiser les durées de blocage de toutes les tâches
10. Que se passe t'il si T5 demande la ressource « gris » après 5 unités de temps ?

Exercice 4. Calcul des temps de blocage maximum dans PIP et PCP

Soit le système suivant, dont les tâches sont présentées dans le tableau ci-dessous par priorités décroissantes. Pour chaque tâche, on indique la durée d'utilisation d'une ressource critiques R1, R2 et R3.

Tâche	**R1**	**R2**	**R3**
T1	3	4	-
T2	-	9	3
T3	5	-	8
T4	-	10	7

En appliquant l'algorithme de calcul de temps de blocage maximal introduit en cours, calculer la durée de blocage maximale subie par chaque tâche (1) quand PIP est utilisé, (2) quand PCP est utilisé.

Vérification d'ordonnançabilité

Exercice 1. Vérification d'ordonnançabilité pour ordonnancements à priorités fixes

On considère le jeu de tâches périodiques synchrones suivant, que l'on souhaite exécuter en utilisant un algorithme d'ordonnancement préemptif à priorités fixes :

Tâche	Ci	Pi	Di
T4	1	4	3
T3	1	5	4
T2	2	6	5
T1	1	11	10

1. Quel algorithme d'ordonnancement utilisez vous ? Pourquoi ?
2. Quelles sont les conditions d'ordonnançabilité vues en cours pouvant être utilisées ici ? Préciser pour chaque condition s'il s'agit d'une condition nécessaire, suffisante, nécessaire et suffisante
3. Appliquez les conditions énumérées à la question précédente, de la moins complexe à la plus complexe, en explicitant pour chaque condition ce que son application permet de conclure

Exercice 2. Vérification d'ordonnançabilité pour ordonnancements à priorités dynamiques

1. Reprendre l'exercice précédent en considérant maintenant un algorithme d'ordonnancement préemptif à priorités dynamiques.

Exercice 3. Vérification d'ordonnançabilité pour ordonnancements hors-ligne non-préemptif de tâches apériodiques

On considère le jeu de tâches apériodiques suivant, que l'on souhaite exécuter en utilisant un algorithme d'ordonnancement non préemptif :

Tâche	Ai	Ci	Di
T1	0	6	16
T2	4	2	8

T3	2	4	7
T4	6	2	10

Dessiner l'arbre de recherche complet et les branches explorées et élaguées par l'algorithme de Bratley

Exercice 4. Ordonnancement de tâches et précédences

Cet exercice vise à mettre en application l'algorithme de vérification d'ordonnançabilité de Silly-Chetto vu en cours (ordonnancement de tâches en EDF préemptif avec contraintes de précédence entre tâches).

On dit qu'une tâche Ta précède une tâche Tb (noté Ta $\rightarrow$ Tb) si la date de début de l'exécution de Tb doit être supérieure ou égal à la date de fin de l'exécution de Ta.

Soient six tâches T1, T2, T3, T4, T5 et T6 telles que : T1 $\rightarrow$ T3, T2 $\rightarrow$ T3, T3 $\rightarrow$ T5, T4 $\rightarrow$ T6, T2 $\rightarrow$ T4, T3 $\rightarrow$ T6. Les tâches arrivent à la date 0, ont une échéance commune de 15 et des temps pire cas d'exécution respectifs de 2, 3, 3, 5, 1, 2.

1. Représenter les relations de précédence entre tâches comme un graphe orienté
2. Donner les valeurs de Ai* et Di* pour chacune des tâches
3. Est-ce que les tâches respectent leurs échéances ?

Estimation de pires temps-d'exécution (WCET)

Exercice 1. Méthodes de calcul de WCET

On considère les extraits de code suivants, chargés respectivement de diminuer le nombre de niveaux de gris d'une image et de détecter des contours :

```
#define T 100 // Taille de l'image
void seuil(char in[T][T],char out[T][T]) {
  int i,j;
  // passage de l'image en noir et blanc
  for (i=0;i<T-1;i++) {
    for (j=0;j<T-1;j++) {
      out[i][j] = 0             // si pixel clair
       + 255*(in[i][j]<128);  // si pixel sombre
    }
  }
}
```

```
void contour (char in[T][T],char out[T-1][T-1]) {
int i,j; char a1,a2;
// Détection de contour avec l'opérateur de Roberts
for (i=0;i<T-1;i++) {
  for (j=0;j<T-1;j++) {
    a1=in[i][j]-in[i+1][j+1]; // détection d'une diagonale
    if (a1<0) a1 =-a1;
    a2=in[i][j+1]-in[i+1][j];// détection autre diagonale
    if (a2<0) a2 =-a2;
    out[i][j] = a1+a2; // calcul de distance
  }
}
}
```

1. Pour chacune de ces deux portions de code, est-il possible d'exécuter le code pour obtenir son pire temps d'exécution ?
2. En supposant que chaque bloc de base s'exécute en 100ns et en utilisant la méthode de calcul à base d'arbres vue en cours, donner le pire temps d'exécution de la fonction *contour*.
3. Donner un exemple de graphe de flot de contrôle et de jeu de contraintes pour calculer le WCET de la portion de code de droite en utilisant une méthode IPET

Estimation de WCET et architectures avec cache

Soit la fonction très simple suivante, qui une fois compilée avec un compilateur croisé MIPS, donne le code assembleur donné ci-dessous. Dans le code assembleur, toutes les instructions suivant un branchement (beqz, j, jr) sont exécutées avant que le branchement ne soit effectif (« delay slots » des architectures MIPS)

Required parameters are missing or incorrect.R incorrect.

Code source C

```
int max(int x, int y){

  int res;

  if (x > y) res = x;
  else res = y;
  return res;
}
```

Code assembleur

```
Adresse (hexa)      Instruction
  400000: addiu   sp,sp,-8
  400004: sw      a0,8(sp)
  400008: sw      a1,12(sp)
  40000c: lw      v0,8(sp)
  400010: lw      v1,12(sp)
  400014: nop
  400018: slt     v0,v1,v0
  40001c: beqz    v0,400038
  400020: nop
  400024: lw      v0,12(sp)
  400028: nop
  40002c: sw      v0,0(sp)
  400030: j       400044
  400034: nop
  400038: lw      v0,12(sp)
  40003c: nop
  400040: sw      v0,0(sp)
  400044: lw      v0,0(sp)
  400048: addiu   sp,sp,8
  40004c: ir      ra
```