Examen d'IN412

Rémy Kocik, Nicolas Pernet

7 Novembre 2008

SANS DOCUMENT

A LIRE AVANT DE COMMENCER :

- les tâches notées T_i sont des tâches périodiques et les tâches notées J_i sont des tâches apériodiques ;
- lorsque les dates d'activation ou phases ne sont pas mentionnées, c'est qu'il y a activation synchrone ;
- afin de dessiner les ordonnancements, des trames vous ont été remises. Pensez à indiquer ce que chaque axe représente (tâche T_n, exercice 11 ...), ainsi que votre nom sur chacune des feuilles ;
- ne construisez un ordonnancement que lorsque cela vous est demandé ! Tout recours à un ordonnancement (graphique) pour prouver quoi que ce soit (ordonnançabilité par exemple) sera noté comme faux ;

1 Exercice 1

Soit le jeu de tâche Ψ suivant à activation synchrone :

	T_1	T_2	T_3	T_4	T_5
C_i	1	1	1	1	2
D_i	3	5	4	7	9
P_i	4	5	7	9	15

Question 1 - Calculez l'hyperpériode de ce jeu de tâche.
Question 2 - Quel est l'algorithme optimal pour ce type de jeu de tâche si on se restreint aux algorithmes à priorités fixes ?
Question 3 - Statuez sur l'ordonnançabilité de ce jeu de tâche avec l'algorithme de la question 2 en calculant le pire temps de rêponse de chaque tâche.
Question 4 - Quel est l'algorithme optimal pour ce type de jeu de tâche ?
Question 5 - Représentez sur l'intervalle [0, 25] l'ordonnancement obtenu en appliquant Earliset Deadline First à ce jeu de tâche. Cela vous suffit-il pour statuer sur l'ordonnançabilité du jeu de tâche avec Earliset Deadline First ?

2 Exercice 2

Soit le jeu de tâches suivant :

	T_1	T_2	T_3
C_i	1	3	2
$P_i = D_i$	4	10	6

Question 1 - Quelle est l'hyperpériode de ce jeu de tâche ?
Question 2 - En utilisant seulement les bornes d'ordonnançabilité sur le facteur d'utilisation, peut-on statuer sur l'ordonnançabilité de ce jeu de tâche avec Rate Monotonic ? Justifiez.
Question 3 - Ce jeu de tâches est-il ordonnançable avec Earliest Deadline First (EDF) ? Justifiez.
Question 4 - On veut rajouter une tâche T_4 d'une durée d'exécution $C_4 = 5$. Quelle condition doit remplir la période P_4 de cette nouvelle tâche pour que le jeu de tâche reste ordonnançable avec EDF ? Quelle est alors la valeur entière minimum que peut prendre P_4 ?

3 Exercice 3

Soit le jeu de tâches périodique suivant :

	T_1	T_2
C_i	1	2
$P_i = D_i$	5	8

Ce jeu de tâche est ordonnancé avec Rate Monotonic. On y ajoute un serveur afin de pouvoir traiter des tâches apériodiques. Le serveur est de type Polling Server (serveur à scrutation) a une capacité $C_s = 2$ et une période $P_s = 7$, les tâches apériodiques y sont traités en FIFO. Surviennent alors les tâches apériodiques suivantes (rappel : r_i correspond à la date d'activation) :

	J_1	J_2	J_3
r_i	2	9	17
C_i	1	2	1

Question 1 - Représentez sur l'intervalle [0, 25] l'ordonnancement obtenu.
Question 2 - Donnez les temps de réponse constatés pour chacune des tâches apériodiques.

4 Exercice 4

Soit le jeu de tâches périodique suivant :

	T_1	T_2
C_i	2	3
$P_i = D_i$	8	10

Ce jeu de tâche est ordonnancé avec Rate Monotonic. On y ajoute un serveur afin de pouvoir traiter des tâches apériodiques. Le serveur est de type Deferrable Server (serveur ajournable) a une capacité $C_s = 2$ et une période $P_s = 6$, les tâches apériodiques y sont traités en FIFO. Surviennent alors les tâches apériodiques suivantes (rappel : r_i correspond à la date d'activation) :

	J_1	J_2	J_3	J_4
r_i	5	9	11	16
C_i	2	1	2	1

Question 1 - Représentez sur l'intervalle [0, 24] l'ordonnancement obtenu.
Question 2 - Donnez les temps de réponse constatés pour chacune des tâches apériodiques.
Question 3 - Citez deux autres techniques d'exécution de tâches apériodiques avec respect des contraintes de tâches périodiques.

Background scheduling
Priority exchange

PARTIE RTAI - à rendre sur copie séparée

On considère le code source du module RTAI fourni en annexe "annexe1.c". Ce module implante l'ordonnancement de tâches temps réel.

Question 1 - Analyser le code de ce module, et à partir de cette analyse tracer l'ordonnancement obtenu à l'exécution dans l'intervalle de temps [now,now+48ms].

Question 2 - On ajoute maintenant dans le code de la tâche 1 et de la tâche 2 des accès à un sémaphore binaire. En prenant en compte le nouveau code des tâches 1 et 2 proposé ici, tracer l'ordonnancement obtenu à l'exécution dans l'intervalle de temps [now,now+48ms].

```
void mon_code1( int arg) {
  int ierr;
 while (1)
  {
    charge(1000000); //1ms
    ierr=rt_sem_wait(&ma_ressource);
    charge(2000000); //2ms
    ierr=rt_sem_signal(&ma_ressource);
    rt_task_set_resume_end_times(-nano2count(P1),-nano2count(P1));

  }
}
```

```
void mon_code1( int arg) {
  int ierr;
 while (1)
  {
    ierr=rt_sem_wait(&ma_ressource);
    charge(4000000); //4ms
     ierr=rt_sem_signal(&ma_ressource);
    rt_task_set_resume_end_times(-nano2count(P2),-nano2count(P2));

  }
}
```

Question 3 - Même question que précédement, mais ici le sémaphore utilisé est un sémaphore de ressource (sémaphore binaire qui inclus un protocole de priorité plafond). Le sémaphore est donc initialisé par le code suivant : `rt_typed_sem_init(&ma_ressource, 1, RES_SEM);`

```c
#include <linux/module.h>
MODULE_LICENSE("GPL");
#include <asm/io.h>

#include <asm/rtai.h>
#include <rtai_sched.h>
#include <rtai_sem.h>



#define NUMERO1       1
#define NUMERO2       2
#define NUMERO3       3
#define PRIORITE1     2
#define PRIORITE2     4
#define PRIORITE3     1
#define STACK_SIZE  2000
#define P1      14000000
#define P2      14000000
#define P3       7000000
#define S1       1000000
#define S2       1000000
#define S3       8000000
#define TICK_PERIOD 1000000


static RT_TASK ma_tache1,ma_tache2,ma_tache3;
SEM ma_ressource;

void charge(RTIME val_charge)
{
  //on suppose que cette fonction permet d'attendre dans la tâche
  // un temps constant quelque soit la préemption et quelque soit la machine uti
lisée
  // (génération d'une charge constante)
  // val_charge doit être donné en ns
}

void mon_code1( int arg) {
  int ierr;
 while (1)
  {
    charge(200000); //2ms
    rt_task_set_resume_end_times(-nano2count(P1),-nano2count(P1));

  }
}

void mon_code2( int arg) {
  int ierr;
 while (1)
  {
    charge(4000000); //4ms
    rt_task_set_resume_end_times(-nano2count(P2),-nano2count(P2));

  }
}



void mon_code3( int arg) {
```



```c
 while (1)
  {
    charge(2000000); //2ms
    rt_task_set_resume_end_times(-nano2count(P3),-nano2count(P3));

  }
}



int init_module(void) {

  //Initialisation paramètres

  int ierr;
  RTIME now;



  rt_set_oneshot_mode();

  ierr = rt_task_init(&ma_tache1, mon_code1, NUMERO1, STACK_SIZE, PRIORITE1, 0,
0);
  ierr = rt_task_init(&ma_tache2, mon_code2, NUMERO2, STACK_SIZE, PRIORITE2, 0,
0);
  ierr = rt_task_init(&ma_tache3, mon_code3, NUMERO3, STACK_SIZE, PRIORITE3, 0,
0);

  rt_typed_sem_init(&ma_ressource, 1, BIN_SEM);

  if (!ierr) {

    start_rt_timer(nano2count(TICK_PERIOD));
    rt_preempt_always(1);

    now = rt_get_time();
    rt_task_make_periodic(&ma_tache1, now+nano2count(S1),nano2count(P1));
    rt_task_make_periodic(&ma_tache2, now+nano2count(S2),nano2count(P2));
    rt_task_make_periodic(&ma_tache3, now+nano2count(S3),nano2count(P3));
  }
  return ierr;
}

void cleanup_module(void) {

  stop_rt_timer();

  rt_task_delete(&ma_tache1);
  rt_task_delete(&ma_tache2);
  rt_task_delete(&ma_tache3);
 rt_sem_delete(&ma_ressource);


}
```