Examen Temps Réel IN412

Rémy Kocik, Laurent George

27 octobre 2006

DOCUMENTS AUTORISES

Les 2 parties sont à rendre sur des copies séparées

1 PARTIE ORDONNANCEMENT - L.George

On considère un jeu de 3 tâches τ={τ_1,τ_2, τ_3 } avec τ_i ={C_i, P_i, D_i} respectivement la durée, la période et l'échéance de la tâche τ_i pour i=1 à 3.

Soit le jeu de tâches suivant τ_1={2,6,6}, τ_2={3,12,7}, τ_3={5,18,18}

Question 1 - Proposez un algorithme d'ordonnancement en priorité fixe optimal pour ordonnancer ce jeu de tâches en contexte préemptif.

Question 2 - Déterminez pour cet algorithme en contexte préemptif priorité fixe, le temps de réponse maximum des trois tâches. En déduire si le jeu de tâches est ordonnançable.

Question 3 - Représentez l'ordonnancement obtenu avec cet algorithme sur [0; 24]

Question 4 - Déterminez le temps de réponse des trois tâches avec le même algorithme en contexte non-préemptif priorité fixe. En déduire si le jeu de tâches est ordonnançable avec cet algorithme.

Question 5 - Représentez l'ordonnancement obtenu sur [0; 24].

Question 6 - On considère maintenant l'algorithme d'ordonnancement EDF préemptif. Déterminez si le jeu de tâches est ordonnançable avec EDF préemptif.

Question 7 - Représentez l'ordonnancement obtenu sur [0; 24]

Question 8 - On considère maintenant l'algorithme d'ordonnancement EDF non-préemptif. Déterminez si le jeu de tâches est ordonnançable avec EDF non-préemptif.

Question 9 - Représentez l'ordonnancement obtenu sur $[0; 24]$

Question 10 - On considère maintenant l'ordonnancement FIFO. Quelle est l'échéance minimale des trois tâches pour que ce jeu de tâches soit ordonnançable ?

2 PARTIE RTAI- R.Kocik

On considère le code source du module RTAI fourni en annexe "annexe1.c". Ce module implante l'ordonnancement de 3 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+40ms].

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+40ms].

```
void mon_code1( int arg) {
  int ierr;
 while (1)
  {
    charge(1000000); //1ms
    ierr=rt_sem_wait(&ma_ressource);
    charge(1000000); //1ms
    ierr=rt_sem_signal(&ma_ressource);
    rt_task_wait_period();

  }
}

void mon_code1( int arg) {
  int ierr;
 while (1)
  {
    ierr=rt_sem_wait(&ma_ressource);
    charge(3000000); //3ms
     ierr=rt_sem_signal(&ma_ressource);
    rt_task_wait_period();

  }
}
```

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