Informatique sûre et temps réel
Examen - 2h00

Master SdI – M2 II & M2 I4

2018–2019

Exercice 1 : Questions de cours (/3)

Indiquez la (ou les) bonne(s) réponse(s) sur votre copie.

Question 1 : L'algorithme d'ordonnancement RMS est-il

1. non préemptif à priorités fixes ?
2. préemptif à priorités dynamiques ?
3. préemptif à priorités fixes ?
4. non préemptif à priorités dynamiques ?

Question 2 : On considère un jeu de n tâches périodiques indépendantes à échéance sur requête (l'échéance est la même que la période). Une condition suffisante pour que ce jeu de tâches soit ordonnançable par l'algorithme RMS est que la charge processeur U vérifie

1. $U > 1$.
2. $U \leq 1$.
3. $U \leq n\left(2^{1/n} - 1\right)$.
4. $U > n\left(2^{1/n} - 1\right)$.

Question 3 : L'algorithme PCE (*Priority Ceiling Emulation*) apporte une solution à quel(s) problème(s) parmi les suivants :

1. Les interblocages.
2. Les inversions de priorité.
3. Les chaînes de blocage.

Exercice 2 : Partage de ressources (/4)

On considère un jeu de trois tâches à priorités statiques. Elles se partagent deux ressources protégées par des sémaphores d'exclusion mutuelle (sémaphores bleu et rouge). La séquence d'exécution pour chaque tâche est donnée dans le tableau ci-dessous.

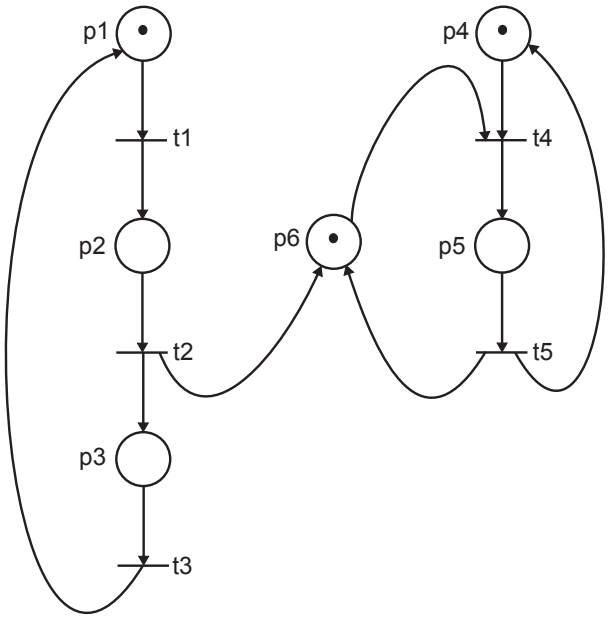
Question 1 : Tracer le chronogramme d'exécution de $t_0 = 0$ à $t_f = 36$ des 3 tâches en utilisant l'algorithme RMS à priorités fixes. Quel(s) problème(s) identifiez-vous sur la séquence d'exécution ? (Dans cette question, il n'est pas demandé d'utiliser l'un des 3 algorithmes vus en cours : PIP, PCE ou PCP)

Question 2 : On choisit d'utiliser l'algorithme PIP pour y remédier. Tracer le nouveau chronogramme d'exécution des 3 tâches. Les problèmes sont-ils tous résolus ? Si non, proposez sans justifier un algorithme qui pourrait résoudre complètement les problèmes identifiés.

tâche	date d'activation	séquence de la charge	échéance	période	priorité
τ_1	6	1+3(bleu)+5(rouge)+1	30	30	6
τ_2	3	1+3(bleu)+8	30	30	4
τ_3	0	1+6(rouge)+1	30	30	2

Exercice 3 : Analyse d'un réseau de Petri (/4)

On se propose d'analyser le réseau de Petri ci-dessous.
- Le réseau est-il vivant ?
- Calculer les invariants (P-invariants et T-invariants).
- Le réseau est-il borné ?



Problème : AmaDrone (/9)

La société de vente par internet AmaDrone souhaite livrer ses petits colis à l'aide de drones aériens de façon à réduire les temps de livraison. On se propose d'analyser plusieurs aspects du système à développer pour réaliser cet objectif.

Ordonnancement de l'algorithme de Navigation-Guidage-Pilotage (NGP) du drone (/3)

Le programme exécuté sur le calculateur du drone est composé de 4 tâches réalisant le NGP à bord du drone :

- L'algorithme de pilotage est cadencé à 100Hz et utilise 1ms maximum de charge de calcul sur le calculateur.
- L'algorithme de navigation est cadencé à 10Hz et utilise 12.5ms maximum de charge de calcul sur le calculateur.
- L'algorithme de guidage est cadencé à 10Hz et utilise 12.5ms maximum de charge de calcul sur le calculateur.
- L'algorithme de planification de trajectoire est cadencé à 0.5Hz et utilise 1s maximum de charge de calcul sur le calculateur.

Question 1 : On souhaite évaluer la possibilité d'ordonnancer ce jeu de tâches avec l'algorithme RMS. Cela est-il possible ? On supposera que ces tâches sont indépendantes, à échéance sur requête (l'échéance est la même que la période) et que leurs dates de première activation sont identiques.
NB : Même si vous n'arrivez pas à obtenir la réponse complète, la démarche utilisée sera prise en compte.

Emballage des colis (/3)

On souhaite modéliser par un réseau de Petri l'emballage de 3 produits dans un colis. Chaque colis a les contraintes suivantes :

1. Un produit 1 (ressource P1) et un produit 2 (ressource P2) sont emballés dans un premier carton d'emballage (ressource C1).
2. Un produit 3 (ressource P3) est emballé avec un carton C1 plein (contenant P1 et P2) dans un deuxième carton d'emballage (ressource C2).
3. Pour fermer un carton d'emballage, on utilise une scotcheuse S1. Cette ressource est unique et utilisée pour fermer les deux types de cartons (C1 et C2)

Question 2 : Faire le réseau de Petri de ce système. Marquer votre réseau de façon à ce qu'il y ait initialement 5 éléments de chaque ressource (P1, P2, P3, C1, C2) et une scotcheuse S1.

Question 3 : Pour le transport par le drone, on souhaite emballer 5 colis (5 cartons C2 pleins) dans un grand carton (ressource C3). La même scotcheuse est utilisée pour fermer C3. Compléter votre réseau de Petri avec le formalisme des réseaux de Petri généralisés de façon à inclure cette phase d'emballage. On marquera la ressource C3 avec un seul élément.

Livraison des colis par le drone (/3)

Disposant de drones opérationnels et d'un système d'emballage de colis, il s'agit désormais pour la société AmaDrone de gérer les différentes phases de la livraison. Le code RTAI fourni en annexe permet de simuler le suivi de la livraison d'un colis à partir de la commande jusqu'à la livraison effective par le drone. Pour cela, trois tâches sont créées :

- Une application smartphone permet au client de passer la commande en fournissant les données de la commande (produit commandé et éléments de paiement) ainsi que son adresse. Lorsque le drone quitte l'entrepôt, le temps restant avant la livraison est affiché sur le smartphone jusqu'à la livraison.
- Une tâche de suivi de la commande par AmaDrone permet de récupérer les données de la commande, de calculer les coordonnées GPS du client (fonction `calcul_gps` non implémentée dans le code qui vous est fourni), d'envoyer un drone avec le colis à ces coordonnées et d'attendre le retour du drone à l'entrepôt.
- Une tâche permettant de gérer la livraison par le drone. Cette tâche récupère les données GPS du client, lance le décollage, envoie toutes les secondes le temps restant avant la livraison à l'application smartphone jusqu'à la livraison et enfin gère le retour à l'entrepôt.

Le code fourni en annexe permet de réaliser ce système de livraison. Le but de cet exercice est de compléter le code permettant de réaliser la tâche gérée par l'application smartphone.

Pour répondre aux questions Q4 et Q5 ci-dessous, on utilisera l'une des fonctions suivantes vues en cours (en justifiant le choix) :
`rt_mbx_send, rt_mbx_send_if, rt_mbx_send_until, rt_mbx_send_timed, rt_mbx_receive, rt_mbx_receive_if, rt_mbx_receive_until, rt_mbx_receive_timed`.
Ces fonctions retournent 0 en cas de succès (réception ou envoi d'un message)

Question 4 : Donner sur votre copie le code qui doit être complété à l'endroit indiqué ("Réponse Q1") de façon à envoyer les données de commande et l'adresse. Pour cela on utilisera les boîtes aux lettres `produit_mbx` et `adresse_mbx`.

Question 5 : Donner sur votre copie le code qui doit être complété à l'endroit indiqué ("Réponse Q2") de façon à rafraîchir le temps restant toutes les 100 millisecondes. Pour cela, on utilisera la boîte aux lettres `temps_mbx`.

```c
/* Squelette code RTAI en mode user : Examen - Amadrone */
#include <stdlib.h>
#include <stdio.h>
#include <signal.h>
#include <sys/io.h>

#include <rtai_lxrt.h>
#include <rtai_mbx.h>
#include <rtai_sem.h>

#define TAILLE_TEXTE    200

// Déclaration boites aux lettres et sémaphores
static MBX *produit_mbx, *adresse_mbx, *temps_mbx, *gps_drone_mbx, *statut_mbx;
static SEM *retour_sem;

void code_suivi_amadrone(int arg) {
  static char donnees_produit[TAILLE_TEXTE]="";
  static char adresse[TAILLE_TEXTE]="";
  static char *GPS = "";

  // PASSAGE EN MODE TEMPS REEL de la tâche
  RT_TASK *ma_tache;
  if (!(ma_tache = rt_task_init_schmod(nam2num("TACHE1"), 1, 0, 0, SCHED_FIFO, 1))) {
        printf("Impossible de créer <TACHE1>\n");
        exit(1);
  }
  // Passage en mode temps réel dur
  rt_make_hard_real_time();

  // Récupération données client
  rt_mbx_receive(produit_mbx,&donnees_produit,TAILLE_TEXTE*sizeof(char));
  rt_printk("Amadrone - données : %s \n",donnees_produit);
  rt_mbx_receive(adresse_mbx,&adresse,TAILLE_TEXTE*sizeof(char));
  rt_printk("Amadrone - adresse : %s \n",adresse);

  // Calcul position GPS du client
  GPS = calcul_gps(adresse);

  // Préparation colis
  rt_printk("Amadrone - colis pret \n");

  // Initialisation drone
  rt_printk("Amadrone - drone initialisé - GPS : %s \n", GPS);
  rt_mbx_send(gps_drone_mbx,GPS,TAILLE_TEXTE*sizeof(char));

  // Fermeture du suivi
  rt_sem_wait(retour_sem);
  rt_printk("Amadrone - fin du suivi \n");

  // Passage en temps reel normal
  rt_make_soft_real_time();
  // detruit la tache temps reel
  rt_task_delete(ma_tache);

}
```



```c
void code_drone(int arg) {
  static char GPS[TAILLE_TEXTE] = "";
  static RTIME temps_restant = 0;

  //PASSAGE EN MODE TEMPS REEL de la tâche
  RT_TASK *ma_tache;
  if (!(ma_tache = rt_task_init_schmod(nam2num("TACHE3"), 1, 0, 0, SCHED_FIFO, 1))) {
        printf("Impossible de créer <TACHE3>\n");
        exit(1);
  }
  // Passage en mode temps réel dur
  rt_make_hard_real_time();

  // Initialisation drone
  rt_mbx_receive(gps_drone_mbx,GPS,TAILLE_TEXTE*sizeof(char));
  rt_printk("Drone - drone initialisé - GPS : %s \n", GPS);

  rt_printk("Drone - récupération colis \n");

  rt_printk("Drone - décollage \n");

  temps_restant = calcul_temps(GPS);
  RTIME temps = temps_restant;
  // Envoi temps restant au client
  rt_printk("Drone - durée restante : %llu secondes \n",temps_restant);
  rt_mbx_send(temps_mbx,&temps,sizeof(RTIME));
  while(temps>0)
  {
        rt_sleep(nano2count(1000000000));
        temps--;

        // Envoi temps restant au client
        rt_printk("Drone - durée restante : %llu secondes \n",temps);
        rt_mbx_send(temps_mbx,&temps,sizeof(RTIME));
  }

  // Colis livré
  rt_printk("Drone - statut : colis livré \n");
  char *statut = "colis livré";
  rt_mbx_send(statut_mbx,&statut,TAILLE_TEXTE*sizeof(char));

  // Retour à l'entrepot
  rt_printk("Drone - retour à l'entrepot dans %llu secondes \n", temps_restant);
  RTIME temps_retour = temps_restant*1000000000;
  rt_sleep(nano2count(temps_retour));

  // Drone de retour à l'entrepot
  rt_printk("Drone - de retour à l'entrepot \n");
  rt_sem_signal(retour_sem);


  // Passage en temps reel normal
  rt_make_soft_real_time();
  // Detruit la tache temps reel
  rt_task_delete(ma_tache);

}
```

```c
void code_suivi_smartphone(int arg) {
  static char donnees_produit[TAILLE_TEXTE]="produit : Nintendo Switch -
  donnees_carte_credit : 1234567898765432";
  static char adresse[TAILLE_TEXTE]="106 avenue Toto, 92120 Montrouge";
  static char *statut = "";
  static RTIME temps_restant = 0;

  // PASSAGE EN MODE TEMPS REEL de la tâche
  RT_TASK *ma_tache;
  if (!(ma_tache = rt_task_init_schmod(nam2num("TACHE2"), 1, 0, 0, SCHED_FIFO, 1))) {
    printf("Impossible de créer <TACHE2>\n");
    exit(1);
  }
  // Passage en mode temps réel dur
  rt_make_hard_real_time();

  // Envoi données client
  /* Réponse Q1 */

  // Temps avant reception colis
  rt_mbx_receive(temps_mbx,&temps_restant,sizeof(RTIME));
  rt_printk("Smartphone - temps restant : %llu secondes \n",temps_restant);
  while(temps_restant>0)
  {
        // Temps avant reception colis
        /* Réponse Q2 */

        rt_printk("Smartphone - temps restant : %llu secondes \n",temps_restant);
        // Raffraichissement toutes les 100ms
        rt_sleep(nano2count(100000000));
  }

  // Colis livré
  rt_mbx_receive(statut_mbx,&statut,TAILLE_TEXTE*sizeof(char));
  rt_printk("Smartphone - statut : %s \n",statut);

  // Passage en temps reel normal
  rt_make_soft_real_time();
  // Detruit la tache temps reel
  rt_task_delete(ma_tache);
}

// Point d'entrée du programme et fonction d'initialisation
int  main (int ac, char **av)
{
  RT_TASK *maint;
  int mon_thread1, mon_thread2, mon_thread3;

  // Autorise a executer des commandes rt aux user
  rt_allow_nonroot_hrt();

  rt_set_oneshot_mode();

  // Passage de la boucle principale en mode temps reel
  if (!(maint = rt_task_init_schmod(nam2num("INIT"), 5, 0, 0, SCHED_FIFO, 1))) {
    rt_printk("Impossible de creer <INIT>\n");
    exit(1);
  }
```



```c
  // Creer une boite aux lettre
  if (!(produit_mbx = rt_mbx_init(nam2num("PRODUIT_MBX"),TAILLE_TEXTE*sizeof(char)))){
    printf("Impossible de creer la BAL <PRODUIT_MBX>\n");
    exit(1);
  }
  if (!(adresse_mbx = rt_mbx_init(nam2num("ADRESSE_MBX"),TAILLE_TEXTE*sizeof(char)))){
    printf("Impossible de creer la BAL <ADRESSE_MBX>\n");
    exit(1);
  }
  if (!(temps_mbx = rt_mbx_init(nam2num("TEMPS_MBX"),sizeof(RTIME)))){
    printf("Impossible de creer la BAL <TEMPS_MBX>\n");
    exit(1);
  }
  if (!(gps_drone_mbx = rt_mbx_init(nam2num("GPS_DRONE_MBX"),TAILLE_TEXTE*sizeof(char)))){
    printf("Impossible de creer la BAL <GPS_DRONE_MBX>\n");
    exit(1);
  }
  if (!(statut_mbx = rt_mbx_init(nam2num("STATUT_MBX"),TAILLE_TEXTE*sizeof(char)))){
    printf("Impossible de creer la BAL <STATUT_MBX>\n");
    exit(1);
  }
  // Creer un semaphore
  if (!(retour_sem = rt_typed_named_sem_init("RETOUR_SEM",0,BIN_SEM))){
    printf("Impossible de creer le SEM <RETOUR_SEM>\n");
    exit(1);
  }

  // Lancer le timer temps reel
  start_rt_timer(0);

  // CREATION DES TACHES pour code_panneau_commande et pour code_smartphone
  // Créer la tâche de mon_thread1
  mon_thread1 = rt_thread_create((void*)code_suivi_amadrone, NULL, 10000);
  // Créer la tâche de mon_thread2
  mon_thread2 = rt_thread_create((void*)code_suivi_smartphone, NULL, 10000);
  // Créer la tâche de mon_thread3
  mon_thread3 = rt_thread_create((void*)code_drone, NULL, 10000);

  // Attend la fin des Tâches
  rt_thread_join(mon_thread1);
  rt_thread_join(mon_thread2);
  rt_thread_join(mon_thread3);

  // Arrete le timer temps reel
  stop_rt_timer();

  // Détruit la boite aux lettres
  rt_mbx_delete(produit_mbx);
  rt_mbx_delete(adresse_mbx);
  rt_mbx_delete(temps_mbx);
  rt_mbx_delete(gps_drone_mbx);
  rt_mbx_delete(statut_mbx);
  rt_sem_delete(retour_sem);
  rt_task_delete(maint);

  return 0;
}
```