Master STL - M1
Module PC2R (MI019)

Programmation Concurrente, Réactive et Répartie

Université Pierre et Marie Curie
Année 2004–2005

Partiel du 29 novembre 2004

Documents autorisés - Durée 2 heures

Exercice 1: Parallèle Excel (C ou O'Caml)

Note: On pourra répondre à cet exercice au choix en langage C ou en O'Caml. On définit une cellule de tableur dans les deux langages de la façon suivante:

En C

```
typedef struct _cellule {
  int valeur;
  phtread_mutex_t lock;
} cellule;

cellule * creer_cellule(int valeur) {
  cellule * cell =
     (cellule*) malloc(sizeof(cellule));
  if(cell==NULL) error("Plus de mémoire");
  cell->valeur = valeur;
  pthread_mutex_init(cell->lock);
}
```

En O'Caml

```
type cellule = { mutable valeur: int;
                 mutable lock: Lock.t };;

let creer_cellule v = { valeur = v ;
                        lock = Mutex.create () };;
```

Question 1:

Définir des fonctions de signatures: `int lire_cellule(Cellule *c)` et `void modifier_cellule(Cellule *c, int valeur)` (en C) ou `lire_cellule : cellule -> int` et `modifier_cellule: cellule -> int -> unit` (en O'Caml) permettant respectivement de lire et d'écrire des contenus de cellules. Ces fonctions peut être appelées depuis des threads différents, on prendra donc bien soin bien synchroniser l'accès au contenu des cellules.

Question 2:

Définir une fonction, correctement synchronisée, de signature: `void EchangerCellules(Cellule * c1, Cellule * c2)` (en C) ou `echanger_cellule : cellule -> cellule -> unit` (en O'Caml) qui permet d'échanger le contenu de deux cellules de tableur.

Question 3:

Soient les déclarations C et O'Caml suivantes :

En C

```
Cellule *c1 = CreerCellule(42);
Cellule *c2 = CreerCellule(36);
void affiche(int id) {
  pthread_mutex_lock(c1->lock);
  pthread_mutex_lock(c2->lock);
  printf("Thread %d : c1 = %d et c2 = %d\n",
          id,c1->valeur,c2->valeur);
  pthread_mutex_unlock(c2->lock);
  pthread_mutex_unlock(c1->lock);
}
void * comportement(void * arg) {
  int id = (int) arg; affiche(id);
  echanger_cellules(c1,c2);
  affiche(id);
}
```

En O'Caml

```
let c1 = creer_cellule 42 ;;

let c2 = creer_cellule 36 ;;

let affiche id = Mutex.lock c1.lock;
  Mutex.lock c2.lock;
  Printf.printf
      "Thread %d : c1 = %d et c2 = %d\n"
      id  c1.valeur  c2.valeur;
  Mutex.unlock c2.unlock;
  Mutex.unlock c1.unlock ;;

let comportement id = affiche id;
  echanger_cellule c1 c2;
  affiche id ;;
```

et les programmes C et O'Caml :

En C	En O'Caml

`int main(void) {`
`  pthread_t t1, t2;`
`  pthread_thread_create(&t1, NULL,`
`  comportement, (void *) 1);`
`  pthread_thread_create(&t2, NULL,`
`  comportement, (viod *) 2);`
`  pthread_join(t1,NULL); pthread_join(t2,NULL);`
`  return 0;`
`}`
	`let main () =`
	`  let t1 = Thread.create comportement 1`
	`  and t2 = Thread.create comportement 2`
	`  in Thread.join t1;`
	`  Thread.join t2 main () ;;`
	
	`main () ;;`

Quels sont les affichages possibles pour le programme si l'on utilise votre définition pour la fonction `EchangerCellules`?

Exercice 2 : RoboVolley (Java)

En Java, on souhaite construire un simulateur de RoboVolley. On dispose d'une classe `Balle` définie de la façon suivante:

En Java

```
public class Balle {
  protected int terrain; // dans le terrain de quelle equipe ? (1 ou 2)
  protected int nb_passes; // nombre de passes public
  Balle(int iterrain) { terrain = iterrain; nb_passes = 0; }
  public void envoyer() { nb_passes=0;
    if(terrain==1) terrain=2; else terrain=1;
  }
  public void passer() { nb_passes++; }
  public int getTerrain() { return terrain; }
  public int getNbPasses() { return nb_passes; }
}
```

On dispose de plus d'une classe `robovolley` permettant de lancer le simulateur, de la façon suivante:

En Java

```
public class robovolley1 {
  public static void main(String args[]) {
    Balle balle = new Balle(1); // l'equipe 1 engage
    Joueur equipe_1[] = new Joueur[7];
    Joueur equipe_2[] = new Joueur[7];
    for(int i=1;i<=7;i++) {
      equipe_1[i-1] = new Joueur(i,1,balle);
      equipe_2[i-1] = new Joueur(i,2,balle);
      equipe_1[i-1].start();
      equipe_2[i-1].start();
    }
    synchronized(balle) { balle.notify(); } // engagement } }
```

Question 1 :

Définir la classe `Joueur` qui hérite de `Thread` et implémente dans sa méthode `run` l'algorithme suivant (dans une boucle infinie):

- Un joueur attend la balle dans une *attente passive*, c'est-à-dire sans gaspiller inutilement le temps processeur
- Lorsque le joueur récupère la balle:
  - si le nombre de passes est inférieur à 2, il passe la balle puis *active* un autre joueur quelconque
  - sinon, il "essaye" d'envoyer la balle dans le camp adverse et *active* un autre joueur quelconque
- **Note**: On fera attention à bien synchroniser les threads `Joueur` et la `Balle` pour éviter les problèmes d'accès concurrents et s'assurer que le programme ne bloque pas.

L'architecture de base de la classe `Joueur` est la suivante:

En Java

```
public class Joueur extends Thread {
  protected int num, equipe;
  protected Balle balle;
  public Joueur(int inum, int iequipe,Balle baballe) {
    num=inum; equipe=iequipe; balle=baballe;
  }
  public void run() {
    while(true) {
      // ?????????
    }
  }
}
```

Question 2:

L'algorithme défini ci-dessus peut conduire à des exécutions du type:

```
Joueur 1 Equipe 1 -> reçoit la balle
                  -> fait une passe à un co-équipier
Joueur 5 Equipe 2 -> reçoit la balle
                  -> fait une passe à un co-équipier
Joueur 3 Equipe 2 -> reçoit la balle
                  -> envoie à l'équipe adverse
Joueur 6 Equipe 2 -> reçoit la balle
```

Le problème est que l'algorithme proposé permet d'effectuer une "gentille" passe à un joueur adverse ou alors d'envoyer dans le terrain en face à son propre coéquipier !

Question 2a : Expliquer pourquoi en termes de concurrence et de mécanismes de synchronisation.

Question 2b : Modifier une ou plusieurs classes de RoboVolley pour s'assurer que les passes se font toujours dans le bon camp et qu'une balle envoyée arrive de façon sûre dans le terrain adverse.

Exercice 3 : Radars de nuit (Esterel)

Un radar routier reçoit un signal d'entrée valué V qui indique qu'une voiture passe à proximité de lui à une vitesse donnée en km/h. Le radar incorpore un mécanisme de flash qui à la réception du signal F envoie immédiatement le signal N valué portant le numéro de la voiture (un entier pour simplifier). Une voiture au plus passe à chaque instant.

Question 1

On demande une programmation du module radar1 en trois branches parallèles.

a) Lorsque la vitesse dépasse une variable vmax (130 par exemple) le détecteur émet le signal D avec la valeur de la vitesse.

b) Quand le signal D est présent, une autre branche déclenche le flash, récupère le numéro de la voiture et envoie le signal de sortie PV associé au numéro de voiture.

c) Une troisième branche émet le signal POL avec le numéro de la voiture lorsque la vitesse dépasse de 50 la valeur vmax.

Question 1a Ecrire en Esterel le fonctionnement de ce radar, y compris les déclarations.

Question 1b Indiquer le comportement de votre programme sur une séquence d'instants en fonction de signaux d'entrée choisis en précisant les signaux de sortie engendrés.

Question 2

On cherche à étendre les fonctionnalités du radar.

Question 2a Le signal valué CONFIG permet de modifier la valeur maximale autorisée. Ajouter la prise en compte de ce signal

Question 2b On désire que le radar ne soit actif que la nuit. On introduit pour cela des signaux JOUR et NUIT indiquant respectivement le lever et le coucher du soleil. Modifier le programme pour en tenir compte.

Question 2c On veut obtenir le record de vitesse sur une période d'un mois. Proposer une solution en indiquant les siganux et variables utilisés? Ajouter cette branche à radar1.

Question 3

On cherche à modulariser les radars.

Question 3a Transformer la première branche en un module detect. On suppose que les autres branches correspondent aux modules flash et police déjà écrits.

Question 3b Ecrire le module radar2 en utilisant les modules précédents.

Question 3c Maintenant, le système contrôle deux radars avec les signaux d'entrée V1 N1 pour le premier, V2 N2 pour le second et les signaux de sortie D1 F1 PV1 POL1 pour le premier D2 F2 PV2 POL2. Ecrire le module radars à partir du module radar2