Contrôle INF 223

23 Novembre 2010

1h30 – Sans document

1 Concurrence

1.1 Threads Posix (2,5 points)

On souhaite faire bloquer un thread tant qu'une variable ne vaut pas une valeur donnée. La solution suivante est-elle correcte ? Pourquoi ? Donner une solution correcte.

```
int x;
pthread_mutex_t lock;
void set_x(int y) {
  pthread_mutex_lock(&lock);
  x = y;
  pthread_mutex_unlock(&lock);
}
```

```
void wait_for_x_equal(int y){
  while (true) {
    pthread_mutex_lock(&lock);
    int r = x;
    pthread_mutex_unlock(&lock);
    if (r == y) break;
    sleep (t);
    }
}
```

Correction

Cette solution provoque de l'attente active et peut manquer des mises à jour si plus de deux d'entre elles surviennent pendant la période t. La solution ci-dessous règle les deux problèmes

L'utilisation conjointe d'une var. cond. et de son verrou permet de bloquer sans « sortir » de l'exclusion mutuelle (problème classique).

```
int x;
mutex_t m;
pthread_cond_t v;
void set_x (int y){
  pthread_mutex_lock(&m);
  x = y;
  pthread_cond_broadcast(&v);
  pthread_mutex_unlock(&m);
}
```

```
void wait_for_x_equal(int y) {
  pthread_mutex_lock &m);
  while (x != y)
    pthread_cond_wait(&v, &m);
  pthread_mutex_unlock(&m);
}
```

1.2 Threads Java (2,5 points)

Si S1 et S2 sont des sémaphores initialisés ainsi :

```
Semaphore S1  = new Semaphore (n);
Semaphore S2  = new Semaphore (0);
```


La gestion suivante d'un schéma producteur/consommateur où put et get sont des méthodes associées à un même objet, est-elle correcte ? Pourquoi ? Corriger en justifiant.

```
synchronized put(Object o){
  S1.acquire();
  last=(last+1)%size;
  buffer[last] = o;
  S2.release();
  }
```

```
synchronized Object get(){
   S2.acquire();
   Object o = buffer[first];
   first=(first+1)%size;
   S1.release();
   return o;
}
```

Correction

```
Object get(){
  fullSlots.acquire();
  synchronized (this) {
     Objet o = b[first];
     first=(first+1)%size;
  }
  emptySlots.release();
  return o;
}
```

```
void put (Object o) {
  emptySlots.acquire();
  synchronized (this) {
    last=(last+1)%size;
    b[last] = o;
  }
fullSlots.release();
}
```

2 Communications

2.1 Sockets (1,5 point)

Comment un serveur en attente sur `accept`, va-t-il être débloqué par un appel distant ? Comment doit se comporter le serveur une fois la connexion établie ?

2.2 Sockets (1,5 point)

Comment un serveur en attente sur accept, va-t-il être débloqué par un appel distant, que se passe-t-il alors ?

3 Patrons de conception

3.1 Questions (6 points)

3.2 Identifier un patron classique utilisé dans la conception d'une des Questions (6 points)

1. Identifier un patron classique utilisé dans la conception d'une des technologies d'intergiciel connues (exemple : RMI, CORBA).

   Note : à défaut (dernier recours seulement !), choisissez un patron de conception présenté en cours pour traiter 2,3 et 4.

2. Exposer brièvement les avantages de l'utilisation de ce patron de conception dans le contexte de l'intergiciel ciblé.

3. Décrire la structure statique de ce patron de conception (ex. via un diagramme de classes UML)

4. Détailler le comportement en exécution de ce patron de conception (ex. via un diagramme de séquence UML)

4 Cas d'étude

4.1 Version threads (2 points)

Dans le cas d'étude en version Thread, expliquer quelle construction est utilisée pour établir les communications entre threads et indiquer quelles propriétés cette construction apporte en termes de synchronisation.
Correction : boite aux lettres ou tampon circulaire. Blocage lorsque vide pour get et lorsque plein pour put.

4.2 Version sockets (2 points)

Dans le cas d'étude en version Socket, expliquer comment est constitué le graphe de communication complètement maillé afin de minimiser le nombre de sockets ouvertes.
Correction : on brise la symétrie en fonction du numéro de noeud pour n'ouvrir qu'une seule socket.

4.3 Version CORBA (2 points)

Dans le cas d'étude en version CORBA, expliquer comment les nœuds arrivent à prendre contact alors qu'ils n'ont aucune information au départ les uns entre les autres. Indiquer une façon alternative pour obtenir le même résultat, en se basant sur une solution présentée dans les notes du cours.
Correction : on stocke les références dans des fichiers. On aurait pu utiliser un serveur de noms.