TD3 : Programmation concurrente et synchronisation

RS : Réseaux et Systèmes

Deuxième année

1 Exercices avec les interblocages

★ **Exercice 1:** Considérez le programme suivant, qui utilise deux sémaphores pour obtenir une exclusion mutuelle.

Conditions initiales

```
1 a = 1
2 b = 0
```

Processus 1

```
1 P(a);
2 V(a);
3 P(b);
4 V(b);
```

Processus 2

```
1 P(a);
2 V(a);
3 P(b);
4 V(b);
```

▷ **Question 1:** Montrez que les processus entrent systématiquement en (inter)blocage.

▷ **Question 2:** Proposez une modification des conditions initiales pour éviter ce problème.

▷ **Question 3:** Montrez graphiquement que votre solution fonctionne.

★ **Exercice 2:** Considérez le programme suivant, qui se termine parfois par un interblocage.

Conditions initiales

```
1 a = 1
2 b = 1
3 c = 1
```

Processus 1

```
1 P(a);
2 P(b);
3 V(b);
4 P(c);
5 V(c);
6 V(a);
```

Processus 2

```
1 P(c);
2 P(b);
3 V(b);
4 V(c);
5 P(a);
6 V(a);
```

Processus 3

```
1 P(c);
2 V(c);
3 P(b);
4 P(a);
5 V(a);
6 V(b);
```

▷ **Question 1:** Listez les paires de sémaphores que chaque processus cherche à obtenir.

▷ **Question 2:** On cherche à éviter les interblocages en ordonnant les réservations selon l'ordre $a < b < c$. Discutez les réservations de chaque processus selon ce critère.

▷ **Question 3:** Proposez une modification pour éviter l'interblocage.

★ **Exercice 3:** Considérez le programme suivant.

Conditions initiales

```
1 a = 1
2 b = 1
3 c = 1
```

Processus 1

```
1 P(a);
2 P(b);
3 V(b);
4 P(c);
5 V(c);
6 V(a);
```

Processus 2

```
1 P(c);
2 P(b);
3 V(b);
4 V(c);
```

▷ **Question 1:** Est-ce que les processus peuvent entrer en interblocage ? Pourquoi ?

2 Exercices avec les conditions de compétition

★ **Exercice 4:** Dans un lavomatique, on cherche une solution pour permettre de répartir les machines à laver équitablement entre les clients. Considérez le programme suivant. Pour obtenir une machine, chaque client doit utiliser la fonction alloue(). Après usage de la machine, il doit utiliser libère().

Conditions initiales

```
1 #define NMACHINES 5
2
3 Semaphore nlibre = sem(5)
4 int dispo[NMACHINES] = (1,1...1)
```

client

```
1  alloue() {
2    int i;
3    P(nlibre)
4    for (i=0; i < NMACHINES; i++)
5      if (dispo[i] != 0) {
6        dispo[i] = 0;
7        return i;
8      }
9  }
10
11 libère(int machine) {
12   dispo[machine] = 1
13   V(nlibre)
14 }
```

▷ **Question 1:** Ce programme présente une condition de compétition. Laquelle ? Pourquoi ?

▷ **Question 2:** Comment corriger le problème ?

3 Problèmes de synchronisation

★ **Exercice 5: Problème du barbier.** La boutique du barbier est composée d'une salle d'attente contenant n chaises et du salon où se trouve la chaise du barbier. Lorsque le barbier a fini de raser un client, il fait entrer le client suivant dans le salon. Si la salle d'attente est vide, le barbier s'y installe pour dormir. Si un client trouve le barbier endormi, il le réveille. Si non, il s'installe dans la salle d'attente s'il reste de la place (et rentre chez lui sinon).

▷ **Question 1:** Écrivez le code du client et du barbier sans vous préoccuper de synchronisation, mais simplement des opérations que veulent réaliser les processus. En particulier, ignorez pour l'instant que le barbier dort parfois.

▷ **Question 2:** Il s'agit maintenant d'ajouter les synchronisations nécessaires au programme écrit à la question précédente. Le premier problème à résoudre est une condition de compétition entre les clients lorsqu'ils rentrent dans la salle d'attente. Corrigez ce problème.

▷ **Question 3:** Assurez-vous ensuite que le barbier ne commence pas à couper les cheveux tant que le client n'est pas prêt.

▷ **Question 4:** Enfin, assurez-vous ensuite que le client ne s'assoit pas sur le siège tant que le barbier n'est pas prêt.

★ **Exercice 6: Problème des philosophes.** Cinq philosophes, réunis pour philosopher, ont au moment du repas un problème pratique à résoudre. En effet, le repas est composé de spaghetti qui, selon la coutume de ces philosophes, se mangent avec deux fourchettes. Or, la table n'est dressée qu'avec une seule fourchette par couvert. Les philosophes décident d'adopter le rituel suivant :

- Chaque philosophe prend une place à table.
- Chaque philosophe qui mange utilise la fourchette à sa droite et celle à sa gauche (pas celle d'en face).
- À tout instant, chaque philosophe est dans l'un des états suivants :
  - il mange avec deux fourchettes ;
  - il a faim, et attend la fourchette de droite, celle de gauche ou les deux ;
  - il pense, et n'utilise pas de fourchette.
- Initialement, tous les philosophes pensent.
- Un philosophe qui mange s'arrête en un temps borné.

▷ **Question 1:** Proposez une solution à ce problème. Vous pourrez utiliser l'une des méthodes vues plus haut pour éviter l'interblocage.

▷ **Question 2:** On ajoute maintenant l'hypothèse suivante au problème :

- Les philosophes sont droitiers : ils ne prennent pas de fourchette à gauche sans avoir d'abord celle de droite.

Proposez une nouvelle solution tenant compte de cette nouvelle hypothèse. Vous serez amené à utiliser l'autre méthode d'évitement des interblocages vue en cours.