

# Série d'exercices #4

IFT-2245

6 février 2017

## 4.1 Conditions de course

Dans beaucoup de systèmes, il peut y avoir des conditions de course. Prenons l'exemple d'un compte en banque partagé par une femme et son mari, et imaginons que le mari fait un appel à `withdraw` au même moment que la femme fait un appel à `deposit`. Décrire tout ce qui peut se passer dans ce genre de circonstance, et comment éviter les différents problèmes.

## 4.2 Mutex avec swap

Utiliser l'instruction spéciale atomique `swap` pour implanter des primitives d'exclusion mutuelle `acquire` et `release` qui assurent l'*attente limitée*.

## 4.3 Atomicité

Montrer que si les opérations `wait` et `signal` des sémaphores ne sont pas exécutées atomiquement, alors l'exclusion mutuelle peut être violée.

## 4.4 Section critique à 2

Voici un candidat-solution au problème de la section critique :

```
bool flag[2] = { false, false };
int turn;

void enter (void) {
    flag[myself] = true;
    while (flag[other] && turn == other) /*wait*/;
}

void leave (void) {
    turn = other;
    flag[myself] = false;
}
```

Pour chacune des trois propriétés désirées (exclusion mutuelle, progrès, et attente limitée), prouver qu'elle est vérifiée ou montrer un contre exemple.

## 4.5 Inhiber les interruptions

Expliquer pourquoi inhiber les interruptions n'est pas une technique appropriée pour implémenter les primitives de synchronisation dans un système multiprocesseur.

## 4.6 Spinlocks

Expliquer pourquoi les *spinlock* sont inappropriés dans un système mono-processeur, mais sont souvent utilisés dans les systèmes multiprocesseurs?

## 4.7 Section critique à 3

Prendre la solution de Peterson au problème de la section critique et le généraliser à 3 processus.