TD TSE1

L3–UPJV

Octobre 2025

Exercices 1 *Emulation des Sémaphores*

En cours, nous avons vu qu'il existe d'autres primitives de synchronisation qui sont les sémaphores. Sur MacOs, les sémaphores ne sont pas présents. Le but de cet exercice est d'écrire des fonctions qui les émule.

En utilisant les variables de conditions, concevez la structure de données et les trois fonctions qui implementent cette fonctionnalité. La structure de donnée se nommera `sema_t`*, la fonction d'initialisation s'appellera* `sema_init(sema_t *s,int v)` *où* `v` *est la valeur initiale du sémaphore, la fonction de décrémentation se nommera* `sema_wait(sema_t *s)` *et la fonction d'incrémentation sera* `sema_post(sema_t *s)`*.*

Exercices 2 *Ordonnancement imposé*

Deux threads T1 et T2 doivent exécuter respectivement les fonctions A() et B(). On veut imposer l'ordre suivant :

1. *T1 exécute A() ;*
2. *puis T2 exécute B() ;*
3. *puis T1 exécute A() ;*
4. *etc...*

Proposer une solution utilisant deux sémaphores pour forcer l'ordre d'exécution.

Exercices 3 *Provisions pour l'hiver*

Une colonie de fourmis a repéré un tas de grains de sucre sur le côté Ouest du ruisseau, malheureusement la fourmilière est sur le côté Est. Les troupes du génie ont réussi à tirer un filin composé de brindilles entre les deux côtés du ruisseau. Cependant, pas plus de 5 fourmis peuvent l'emprunter à la fois, sinon, il casse. D'autre part, les fourmis doivent être dans le même sens sinon elles se bloquent au milieu du filin.

Les fourmis doivent donc récupérer tous les grains pour les ramener à la fourmilière. Elles ne peuvent charger qu'un grain à la fois. Les fourmis appliquent donc l'algorithme suivant (une fourmi est un thread) :

```
while (tas non vide) {
    gotoWest();
    chargeUnGrainDeSucre();
    gobackEast();
    dechargeGrainDeSucre();
}
```

On considère que l'on a comme variables globales : `toWest`, le nombre de fourmis qui veulent aller chercher un grain de sucre ; `toEast`, le nombre de fourmis qui veulent revenir à la fourmilière ; `goingWest`, le nombre de fourmis qui sont sur le filin en direction de l'Ouest ; `goingEast`, l'inverse.

Écrire les fonctions `gotoWest()` et `gobackEast()` qui permettent d'assurer la synchronisation des fourmis, ceci avec des variables de condition*.*

Exercices 4 *Provision pour l'hiver bis*

Refaire le même exercice avec des sémaphores cette fois ci.

Exercices 5 *Les mangeurs de beignets*

Dans cet exercice on va simuler le comportements d'un système composé d'un patissier et de plusieurs enfants amateurs de beignets.

- *Il y a n enfants (threads), un patissier (un autre thread).*
- *Le patissier produit les beignets M par M qu'il dispose sur un plat.*
- *Les enfants mangent les beignets qui sont sur le plat un par un.*
- *Si le plat est vide, les enfants attendent que le patissier refournisse un plat complet.*
- *Le patissier attend que le plat soit vide avant d'en fournir un autre.*

Le nombre de beignets sur le plat présents à un instant t, sera représenté par une variable globale `b`.

Vous devez écrire le code pour les threads enfants ainsi que celui du thread patissier qui assure la synchronisation telle que décrite auparavant.

> *Une idée est d'utiliser trois sémaphores, un qui assure l'exclusion mutuelle pour l'accès à la variable globale, un autre qui représente le fait que le plat est vide, un autre pour le fait que le plat est de nouveau plein.*