# TD Programmation Système

# Série 8 : THREADS

## Exercice 1 : Thread identificateur

Ecrire le programme C sous Unix qui permet à un processus de créer un nombre de THREAD passés en paramètres. Pour chaque THREAD créé, on affiche son rang et son identificateur.

**Indication :**
-Déclarer un tableau de THREAD avec une taille maxi prédéfinie permettant de stocker les threads créés.

-Utiliser la fonction

pthread_t pthread_self(void);

Elle retourne l'identifiant du thread appelant (C'est la même valeur de type pthread_t initialisée par l'appel à la fonction pthread_create()).

Appel Système : pthread_create(), pthread_join(),pthread_self(void), perror();

## Exercice 2 : Synchronisation avec les Threads

On désire gérer la gestion du stock d'un magasin en utilisant les Threads. On distingue deux types de thread :
-Thread Client qui se sert du stock en prenant un nombre d'article
-Thread Stock qui charge le stock du magasin dès qu'il devient trop bas pour satisfaire les clients.

1. Ecrire le programme C qui permet de créer cinq Threads Client et un Thread Stock. On suppose que le nombre d'articles pris du stock est un nombre aléatoire ainsi que l'ordre de passage des Threads clients. Le Thread Stock alimente le stock du magasin dès que celui-ci devient inférieur où égale à zéro.
Appels Système : *pthread_create(), pthread_join(),perror()*

2. On désire gérer l'exclusion mutuelle permettant aux Threads Clients et Stock de ne pas accéder et modifier en même temps la valeur du stock. Modifier le programme de la question 1, pour prendre en considération l'exclusion mutuelle
Appels Système : *pthread_mutex_lock (),pthread_mutex_unlock ()*

3. On désire interdire aux Threads clients de se servir du stock quand le stock <=0 ou quand le nombre d'articles du stock est inférieur à la valeur commandée par un Thread client. Modifier le programme de la question 2, pour prendre en considération la synchronisation entre les Threads Clients et Stock.
AppelsSystème : *pthread_mutex_lock(),pthread_mutex_unlock(),pthread_cond_wait(), pthread_cond_signal ()*