

## Processus de poids léger &amp; Sémaphore

## ■ ■ ■ Création et terminaison de threads

Pour utiliser la bibliothèque Pthreads dans un programme C, il faut :

- inclure son fichier de déclarations standard :  
`#include <pthread.h>`
- « linker » lors de la compilation avec la bibliothèque `libpthread.a` :  
`gcc -o Mon_programme mon_source.c -lpthread`
- déclarer des variables pour un ou plusieurs descripteurs de threads :  
`pthread_t pid, cid; /* descripteurs de thread */`
- créer les threads : `pthread_create(&pid, NULL, start_func, arg);`  
Le premier argument est l'adresse d'un descripteur de thread, suivi de `NULL`.  
Le code de la thread est la fonction `start_func` qui doit avoir un seul argument `arg`.  
*Si la création est effectuée correctement, la fonction renvoie 0, sinon un code d'erreur.*

Une thread **se termine** :

- ★ lorsque la fonction associée s'arrête (retour de la fonction);
- ★ en appelant : `pthread_exit(value)` ; où l'argument peut être `NULL`.

Une thread « parent » peut **attendre la terminaison** d'une thread « enfant » en effectuant :

- ◇ `pthread_join(pid, value_ptr)` ; où le deuxième argument est une adresse pour le stockage de la valeur de retour du thread renvoyée par le `pthread_exit()`.

Pour l'utilisation des sémaphores en programmation C :

```
1 #include <semaphore.h>
2
3 int main()
4 {
5     sem_t semaphore;
6     int compteur = 0;
7     sem_init( &semaphore, 0, 1 ); /* On crée la sémaphore avec la valeur 1
8     *//
9     sem_wait( &semaphore ); /* On « prend » la sémaphore */
10    compteur++;
11    sem_post( &semaphore ); /* On « libère » la sémaphore */
12    sem_destroy( &semaphore );
13 }
```

- 1 – Écrire le programme basé sur l'utilisation des threads, permettant pour deux threads distinctes de communiquer l'une vers l'autre.

*Vous illustrerez le fonctionnement du programme avec un échange de valeur de compteur à échanger (la première thread utilise la valeur 1 qu'elle transmet à la seconde thread qui l'incrémente et la repasse à la première etc.).*

- 2 – Écrire un programme qui convertit des nombres en base 10 vers la base 2, en utilisant un « pipe-line » de threads.

Votre programme doit être constitué de 8 threads :

- \* la première thread lit depuis le clavier le nombre à convertir, puis, transfère cette information vers la thread suivant ;
- \* les 7 autres threads effectuent les opérations suivantes :
  - ◊ affichage de 0 ou 1 selon que le nombre reçu par la thread soit divisible ou non par deux ;
  - ◊ transfert du nombre divisé par 2 au thread suivant (sauf si le nombre divisé est égal à zéro).
- \* la dernière thread du pipe-line doit afficher une erreur si le nombre à convertir dépasse la capacité du pipe-line.

*Remarque : tel que le pipe-line est décrit ici, le résultat de la conversion doit normalement afficher le nombre en base 2 à l'envers.*

- 3 – Le problème des pizzaiolo et des pizzas *Mozarella* :

- \* pour réaliser une pizza *Mozarella*, il est nécessaire de disposer des 3 ingrédients suivants :
  - ◊ pâte ;
  - ◊ mozarella ;
  - ◊ tomate ;
- \* dans la pizzeria, il y a :
  - ◊ 3 pizzaiolo, chacun :
    - \* possède un des ingrédients en quantité illimitée ;
    - \* fabrique une pizza *Mozarella* en se procurant les deux ingrédients qui lui manquent, se repose un court instant avant de recommencer ;
  - ◊ un préparateur : qui dispose sur la table de préparation deux ingrédients au hasard ;
- \* lorsque le préparateur fournit deux ingrédients, le pizzaiolo qui dispose du 3<sup>ème</sup> ingrédient doit prendre ces ingrédients pour fabriquer sa pizza. Le préparateur attend que le pizzaiolo finisse avant de recommencer.

**Questions :**

- a. Modélisez le problème à l'aide de threads et de sémaphores ;
- b. Écrire le programme correspondant.

**Remarque :** vous utiliserez la fonction `long random(void)` ; pour obtenir une valeur aléatoire.