UNIX - TD n^o 7 : Les processus légers (Threads)

Un de vos TD sera ramassé à la fin de la séance pour évaluation (note de CC). Il est conseillé de rédiger au fur et à mesure les réponses aux questions et de bien commenter le code que vous rédigez. Les documents seront nommés selon la convention : td<N>-<Q>[_<I>][.<ext>] (<N> : numéro du TD ; <Q> : numéro de la question ; <I> : info facultative selon le sujet ; <ext> extension si nécessaire ; exemples : td1-2.c, td3-1_client.h).

---

1 Toujours du calcul !

Écrivez un programme qui effectue le calcul $(a + b) * (c + d)/(e + f)$ en utilisant trois threads (voir le manuel des pthreads et la fonction pthread_create()).

Les threads utiliseront la même fonction pour réaliser la somme. Les paramètres et le résultat seront contenus dans une structure qui sera passée en paramètre aux threads :

```
typedef struct _Calcul { float op1, op2, result; } Calcul;
```

Le reste du calcul (multiplication et division) sera effectué après la fin des trois threads (voir la fonction pthread_join()).

Attention : la compilation d'un programme utilisant les pthreads nécessite l'utilisation de l'option "-lpthread" afin que l'éditeur de liens connaisse l'emplacement de cette librairie.

2 Verrou

Écrivez un programme qui crée deux threads :
- l'un écrit des informations dans un fichier puis il s'endore pendant 10 secondes et recommence,
- l'autre va lire dans ce fichier puis s'endore pendant 5 secondes avant de recommencer.

Faites en sorte que l'accès au fichier soit protégé par un mutex (voir les fonctions pthread_init(), pthread_lock(), pthread_unlock(), pthread_destroy()).

3 Synchronisation

Écrivez un programme ayant le comportement suivant :
- création de 5 threads,
- chaque thread effectue un travail durant un temps aléatoire,
- tous les threads se terminement en même temps à l'aide d'une synchronisation.

Vous utiliserez le mécanisme de la barrière pour effectuer la synchronisation (voir les fonctions thread_barrier_init(), thread_barrier_wait(), thread_barrier_destroy()).