Sujet du TP n° 5

Préambule

Les ressources du cours (supports, exemples, et autres documents) et des TP (les sujets, les fichiers de travail ainsi que les corrigés) se trouvent à l'URL : http://www.emse.fr/~lalevee/ismin/pse, dénommé **[site]** dans les documents fournis.

Ce TP suppose que vous avez installé l'archive **PSE.tar**. Si ce n'est pas le cas, consultez le sujet du premier TP.

Placez-vous dans le répertoire **PSE/TP5**.

les THREADS (96 à 106)

Exercice 1

- copiez le fichier **exemples/slide100.c** dans le répertoire **TP5** et renommez-le **exercice1.c**
- compilez-le (tapez **make**) puis testez-le.
- que se passe-t-il ?
- corrigez.

Exercice 2

- modifiez le programme précédent afin que le thread **main** attende la fin du thread créé et affiche son code de fin (celui-ci retournera d'abord la valeur **NULL**)

Note : pour afficher un pointeur, utilisez **%p**

Exercice 3

- modifiez le programme de l'exercice précédent, afin que le thread créé retourne la taille de la chaine transmise.

Exercice 4

- écrire un programme qui crée 10 threads numérotés de 1 à 10 et attend leur fin en affichant leur code de retour
- celui-ci est la somme des entiers de 1 jusqu'à leur numéro

Attention : ne pas retourner d'adresse de variable locale !!

Application multithread

Le but de cette partie, qui constituera le point de départ de votre projet, consiste à rendre multiutilisateur (plusieurs clients) l'application client/serveur du **TP3**. Comme nous l'avons constaté, les demandes de connexion des utilisateurs sont mises en file d'attente, si un client est déjà connecté et en cours de traitement par le serveur.

Pour que les requêtes des clients soient traitées dès leur réception, le serveur devra créer un nouveau thread avec **pthread_create()** à chaque connexion acceptée (avec **accept()**) ; ce thread sera chargé de traiter la requête et de répondre au client, pendant que le thread main se mettra en attente d'une nouvelle connexion.

Vous devrez porter une attention particulière aux données « privées » des threads (mise en place d'une liste chaînée de ces identités). Pour cela, vous disposez du module **datathread** présent dans le répertoire **[pse]/modules** et le répertoire **[pse]/include**. Dans ce dernier :

- le fichier **datathread.h** contient les déclarations des structures et fonctions permettant la gestion de la liste chaînée
- le fichier **dataspec.h** contient la déclaration d'une structure contenant les données spécifiques liées aux threads (identifiant du thread, numéro de canal pour les communications et un identifiant logique, interne à l'application).

Projet

- copiez vos fichiers **serveur.c** et **client.c** du répertoire **TP3** dans le répertoire **TP5**
- vous pouvez aussi recopier ces fichiers du répertoire **TP3-C**
- modifiez le fichier **serveur.c** pour réaliser l'algorithme décrit ci-dessous.

L'algorithme général est le suivant :

```
canal = accept(..);
pthread_create(...);
si thread main alors
    verification des threads fils eventuellement termines
sinon
    traiter_requete(canal)
    close(canal)
    pthread_exit()
finsi
```