TP Threads

Compilation : gcc thread.c -o thread -lpthread
Aide : man pthread _create

1_ Créez un programme C permettant de lancer un thread qui affichera son numéro et son ID (pthread_self()). Cet id sera passé en paramètre à la fonction associée.

2_ Modifez ce programme pour qu'il permette le lancement de 5 threads affichant chacun son numéro et son ID.

3_ Passage d'arguments à un thread. Créez une structure permettant de mémoriser un entier demande, un entier réservation, un message de 50 caractères et une constante NBP. Montrez sur un exemple que l'on peut passer plusieurs arguments à un thread grâce à une structure.

4_ Reprenez le programme de l'exercice 2_ et ajoutez la routine pthread_join() permettant la synchronisation entre threads. Le but étant de mettre le thread principal en attente de la fin de chacun des 5 threads et d'afficher son état à la terminaison.
Attention pour cela il faut créer des threads dans l'état joignable : `pthread_attr_init()`
`pthread_attr_setdetachstate()`

5_ Mutex : Lors d'accès concurrents à une zone de données partagée par plusieurs threads la cohérence des données est incertaine. Aussi l'utilisation de mutex devient nécessaire. Sur l'exemple de l'exercice 3, la structure devra être déclarée comme variable globale accessible par tous les threads, et l'accès régulé par un mutex).

6_ Compilez puis exécutez le programme fourni en exemple et expliquez son fonctionnement global puis commentez toutes les instructions.

TP openMP

Compilation : gcc source.c -o destination -fopenmp

1_ En utilisant les 3 techniques exposées en cours :
- les directives,
- les fonctions,
- et les variables d'environnement

créez 3 programmes permettant de lancer un nombre N de threads et affichez leurs id respectifs.

2_ Soit le programme suivant :

```
   #include <omp.h>
2. #define CHUNKSIZE 100
3. #define N     1000
4.
5. main ()
6. {
7.
8. int i, chunk;
9. float a[N], b[N], c[N];
```

```
10.
11.  /* Some initializations */
12.  for (i=0; i < N; i++)
13.    a[i] = b[i] = i * 1.0;
14.  chunk = CHUNKSIZE;
15.
16.  #pragma omp parallel
shared(a,b,c,chunk)private(i)
17.    {
18.
19.   #pragma omp for schedule(dynamic,chunk) nowait
20.    for (i=0; i < N; i++)
21.      c[i] = a[i] + b[i];
22.
23.  }  /* Fin de la section parallèle  */
24.
25.  }
```

Compilez-le et exécutez le, ajoutez les instructions nécessaires pour observer le nombre de threads créés et expliquez ce qui se passe.