INTRODUCTION AUX SYSTEMES D'EXPLOITATION

TP5
L'ordonnancement

S O M M A I R E

1.Présentation rapide

Ce TP complète le TP4 sur le calcul de produit scalaire par plusieurs threads.

Les exercices de ce TP sont principalement inspirés du livre de C. Blaess "Solutions Temps réel Sous Linux", Ed. Eyrolles, que je conseille vivement.

Les fonctions utilisées dans ce TP sont des extensions GNU non portables sur d'autres systèmes UNIX, elles nécessitent donc de définir (#define) la constant symbolique _GNU_SOURCE avant d'inclure <sched.h> (#define _GNU_SOURCE).

2.Exercice 1:

Avant de faire cet exercice, il faut s'assurer que la machine virtuelle lancée (si vous utilisez la virtualisation) utilise plusieurs cœurs (voir la configuration de votre VM).

Ecrivez un programme qui fait une boucle infinie et qui affiche le CPU sur lequel il s'exécute à chaque fois que le système lui change de CPU.

Votre processus change-t-il de CPU ? vous pouvez lancer votre programme sur plusieurs terminaux (en même temps et observer le comportement).

Pour interroger le système sur le CPU exécutant un processus, vous pouvez utiliser l'appel **`sched_getcpu().`** Cette fonction est une extension GNU non portable sur d'autres systèmes UNIX, elle nécessite donc de définir (#define) la constant symbolique _GNU_SOURCE avant d'inclure <sched.h>.

3.Exercice 2

L'affinité d'une tâche est la liste des CPUs sur lesquels elle peut s'exécuter. On peut la consulter ou la fixer à l'aide des fonctions suivantes:

```
int sched_setaffinity(pid_t pid, size_t size, const cpu_set_t* cpuset)
int sched_getaffinity(pid_t pid, size_t size, cpu_set_t* cpuset)
```

Le second argument correspond à la taille du type de donnée `cpu_set_t`. Les listes de CPU sont représentées par les variables de type `cpu_set_t`. Cet ensemble est manipulé via les macros suivantes:

```
void CPU_ZERO(cpu_set_t *ensemble); // vide l'ensemble
void CPU_SET(int cpu, cpu_set_t * ensemble); // insérer un CPU de l'ensemble
void CPU_CLR(int cpu, cpu_set_t * ensemble);//supprimer un CPU de l'ensemble
int CPU_ISSET(int cpu, cpu_set_t * ensemble);// vérifier qu'un CPU fait
partie de l'ensemble
```

Q) Ecrivez un programme permettant de déterminer le CPU sur lequel ce programme s'exécutera (le programme prend en argument le numéro de CPU).

4.Exercice 3 (suite du TP précédent sur les threads)

Dans le TP précédent, vous avez développé un programme vous permettant de faire le produit scalaire en faisant faire les multiplications par des threads différents.
Dans cet exercice, on vous demande d'abord de trouver le nombre de processeurs disponibles via un appel à la fonction **`long sysconf(int nom)`** qui retourne la valeur d'une configuration/ressource donnée. Pour avoir le nombre de processeurs, il faut utiliser le "nom" _SC_NPROCESSORS_ONLN.
Une fois le nombre de processeurs connu, modifiez l'affinité de chaque thread pour équilibrer la charge sur l'ensemble des processeurs existants. Vous pouvez vous aider de l'offset envoyé à chaque thread.