Parallélisation à l'aide de processus et de threads

On compilera les programmes basés sur des pthreads en utilisant la ligne de commande : `gcc -std=c99 -lpthread -D_REENTRANT`.

Exercice 5.1 Il s'agit de compléter le code de la commande "`hello-goodbye` *`entier`*" qui lance en parallèle *entier* threads effectuant la fonction `HelloGoodbye()`. Dans un premier temps, on se contentera de lancer un seul thread.

Exercice 5.2 En utilisant le nombre de threads passé en argument, paralléliser la boucle du programme `calcul.c` :

```
for(int k=0; k < MAX; k++)
  resultat[k] = tan(atan((double)k));
```

En particulier, il s'agit de partionner le travail entre les différents threads pour éviter que deux threads aient une partie de travail commune. On distribuera les indices aux threads selon deux techniques

- distribution par bloc : chaque thread traite une séquence d'entiers successifs ;
- distibution cyclique : le thread i traite les indices congrus à i modulo le nombre de threads.

Relever les temps d'exécution du programme en fonction de la distribution et du nombre de threads exécutés et du nombre de cœurs de la machine.
Reprendre l'exercice en simplifiant le calcul ;

```
resultat[k] = atan((double)k);
```

Exercice 5.3 La commande "`for-en-parallele x`" lance en parallèle x threads réalisant la boucle suivante :

```
for(unsigned long i=0; i < MAX; i++)
  k++;
```

1. Sachant que `k` est une variable globale initialement nulle, quelle(s) valeur(s) peut-elle avoir à la suite de l'exécution de `x` threads ?
2. Conforter votre intuition en exécutant plusieurs fois le programme `for-en-parallele` tout en faisant varier le nombre de threads.
3. Quelle valeur minimale peut avoir `k` ?
4. Comment faire en sorte que `k` vaille toujours `x.MAX` ?

Exercice 5.4 Il s'agit de paralléliser la résolution du problème des n-reines dont l'énnoncé est le suivant : *«soit un échiquier de dimensions $N \times N$, combien y'a-t-il configurations où N reines sont disposées sur cet échiquier sans qu'aucune d'elles ne se menace.»* Une version séquentielle est disponible dans le programme `nreine.c`, cette version est basée sur un algorithme récursif qui passe en revue toutes les solutions possibles en progressant ligne par ligne.

On peut utiliser le programme `forkN.c` comme point de départ pour paralléliser le problème des N reines en utilisant uniquement des processus. L'idée est que chaque processus fils exécute une version légèrement modifiée du programme `nreines.c`. Appelons `nreines_partiel.c` ce programme qui, en utilisant un paramètre supplémentaire indiquant le rang i du processus, calcule toutes les solutions générées en fixant une reine sur la $i$ème colonne de la ligne 0.

1. Dans une première version, on laisse simplement chaque processus `nreines_partiel` afficher le nombre de solutions qu'il a trouvées sur la sortie standard.
2. Modifiez le programme précédent pour que chaque processus fils inscrive son nombre de solutions à la $i$ème position (en utilisant `lseek`) dans un fichier d'entiers au format binaire. Le programme principal peut alors lire ce fichier après la terminaison des fils pour calculer la somme des N contributions.

Exercice 5.5 Il s'agit de paralléliser le programme `nreines.c` en faisant en sorte que chaque thread travaille sur son propre échiquier initialisé à partir d'une configuration ad hoc.