Exercice 3

Difficulté : 50 points-virgules

L'objet de l'exercice est de réaliser le produit scalaire de deux vecteurs X et Y. On rappelle que le produit scalaire est défini par $X.Y = \sum_{i=1}^{n} x_i y_i$. Plutôt que réaliser les calculs directement, on utilisera la commande `expr` du système pour calculer les différents produits $x_i y_i$ en parallèle ainsi que la somme finale. On vous demande donc de rédiger le programme selon la syntaxe suivante :

`prodscal` *délai* $x_1 ... x_n$ $y_1 ... y_n$

Ainsi, « `prodscal 1000 1 2 3 4 5 6` » lance en parallèle les commandes « `expr 1 * 4` », « `expr 2 * 5` » et « `expr 3 * 6` » en redirigeant leur sortie standard vers des fichiers distincts qu'on nommera par exemple `res0` à `res2`. Avant chaque commande `expr`, on demande d'introduire un délai aléatoire, non identique entre tous les calculs, borné par l'argument `délai` en millisecondes (vous pourrez utiliser pour ce faire les fonctions `rand` et `usleep`).

Ensuite, une fois les commandes terminées sans erreur, le programme lit les fichiers et les efface, puis exécute la nouvelle commande « `expr 4 + 10 + 18` » (sans délai) qui affiche son résultat sur la sortie standard.

On notera que la commande `expr` suppose que chaque opérande ou opérateur est un argument distinct, ce qui impose de les séparer par des espaces si on souhaite l'exécuter depuis le Shell. De même, le caractère « `*` » étant spécial en Shell, il convient de le neutraliser en l'entourant de guillemets ou en le préfixant par « `\` ».

Pour rédiger votre programme, il est impératif de respecter les contraintes suivantes :

- vous ne devez utiliser que les primitives système (ou assimilées comme telles) ; vous pouvez toutefois utiliser les fonctions de bibliothèque pour les affichages ou les manipulations de chaînes de caractères, de mémoire ou la génération de nombres pseudo-aléatoires ;
- pour des raisons d'efficacité, vous ne ferez pas d'appels redondants à des fonctions lentes (primitives système ou autres) ;
- vous ne ferez aucune vérification syntaxique (autre que la longueur) des arguments fournis : les erreurs, s'il y en a, doivent être détectées par la commande `expr` ;
- pour des raisons de simplicité, on pourra considérer que les nombres manipulés sont strictement inférieurs à 1 000 000 000, à l'exception du résultat final ;
- vous vérifierez soigneusement les débordements de tableau (vous pouvez notamment utiliser la fonction de bibliothèque `snprintf` pour contrôler la taille de chaînes complexes) ;
- votre programme doit retourner un code de retour nul (`exit(0)`) si tout s'est déroulé sans erreur ou un code de retour non nul (`exit(1)`) si une erreur a été rencontrée ;
- si votre programme est appliqué avec un nombre d'arguments incorrect, il doit afficher un message de la forme : `"usage : prodscal delai x1 ... xn y1 ... yn"`.
- vous apporterez un soin particulier à la mise en forme de façon à rendre un code lisible et commenté à bon escient. Référez-vous au document « Conseils pour réussir vos TP et projets » mis à votre disposition sur Moodle et, si besoin, utilisez l'utilitaire `clang-format` avec la configuration donnée dans ce document ;
- votre programme doit compiler avec les options `-Wall -Wextra -Werror -pedantic` sur `gcc` version 9.4 minimum (la version disponible sur la machine `turing.u-strasbg.fr`. Alternativement, vous pouvez utiliser l'image Docker `pdagog/refc` (version de `gcc` 12.2) Les programmes qui ne compilent pas au moins sur `turing` avec ces spécifications **ne seront pas examinés**.

Un script de test est mis à votre disposition sur Moodle. Celui-ci exécute votre programme sur des jeux de tests qui serviront de base à l'évaluation de votre rendu. La commande suivante permet de lancer les tests : `sh test3.sh`.

Vous devrez rendre sur Moodle un *unique* fichier nommé `prodscal.c`.

Cet exercice est **individuel**. On rappelle que la copie ou le plagiat sont sévèrement sanctionnés.