Exercices sur les mécanismes de synchronisation

Exercice 1

On désire réaliser une petite librairie implémentant une pile pour des entiers. La taille de la pile sera **statique** et déterminée au moment de sa création. Afin de garantir un comportement cohérent et déterministe dans le cas d'une exécution multi-threadée, la pile devra être thread-safe. Vous utiliserez les primitives d'exclusion mutuelle appropriées pour cela. L'interface des fonctions à implémenter se présente comme suit :

- Création d'une pile de taille déterminée et retour d'un booléen indiquant si la création à réussi:
  `bool stack_create(stack_t **s, int max_size);`
- Détruire une pile et retour de booléen si la destruction a réussi:
  `bool stack_destroy(stack_t *s);`
- Empiler une valeur et retour de booléen si la destruction a réussi:
  `bool stack_push(stack_t *s, int val);`
- Dépiler une valeur:
  `int stack_pop(stack_t *s);`

Décider ce que contiendra la structure `stack_t` définissant un objet de type pile.

Remarque: Penser à insérer des assertions dans le code aux endroits nécessaires.

Exercice 2

A l'aide de votre structure de pile de l'exercice un, écrivez une structure de table de hachage très simple qui possédera les fonctions de création de la table, d'insertion d'un élément et de recherche d'un élément. Il n'y a pas besoin de faire une réallocation de table pleine.

Mesurer la performance de votre table de hachage concurrente.

Exercice 3

On fournit le code source (`dict.tar.gz`) d'une librairie implémentant un dictionnaire de type `< clé,valeur >` où clé et valeur sont des strings. Le code

fourni a été développé dans le cadre d'une exécution séquentielle. Modifier le code fourni en utilisant les mécanismes vu en cours afin que les fonctions de la librairie soient thread-safe. Essayer de minimiser les contentions lors des accès au dictionnaire et éviter toute attente active.

Développer un programme illustrant un scénario de test multi-threadé afin de vérifier la validité de votre implémentation.

Questions 1

Est-ce que la fonction `malloc()` est thread-safe? Comment le déterminer?
