HE-IG VD | REDS Institut Reconfigurable and Embedded Digital Systems

Systèmes d'exploitation (SYE)

Profs Daniel Rossier, Alexandre Corbaz, Fiorenzo Gamba
Assistants : Anthony Jaccard, Basile Cuneo, Jean-Pierre Miceli

Threads

lab06 (Semaine du 30.10.2023)

Objectifs de laboratoire

Ce laboratoire permet d'exercer la notion de *thread* afin de paralléliser un traitement

Récupération et rendu du laboratoire

Pour récupérer la branche de ce laboratoire, utilisez les commandes:

1. "*git fetch upstream*"
2. "*git checkout labXX"*

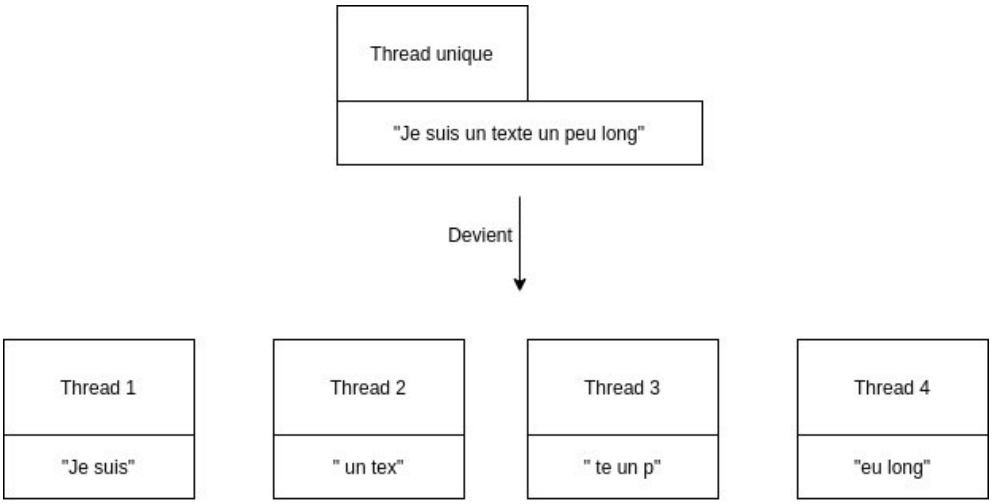
Ce laboratoire sera à rendre selon la méthode vue lors du précédent laboratoire, à savoir

1. Ajouter les fichiers modifiés avec "*git add <vos_fichiers>"*
2. Commiter les changements avec "*git commit –m "<message de commit>"*
3. Envoyer les changements sur votre repo gitlab avec "*git push –u origin"* ou "*git push —set-upstream origin"*

Etape 1 - Création et exécution de *threads*

Un processus peut contenir un ou plusieurs *threads*. Cet exercice permet d'exercer le démarrage et de gérer l'exécution de quelques threads à l'intérieur d'un processus. L'application qui servira de test est compilée à partir du fichier « ***usr/host/threads.c*** ». Les applications compilées à partir du script contenu dans ce dossier seront exécutables nativement dans la VM. Ainsi, pas besoin d'utiliser l'émulateur et SO3

Le but sera de compter le nombre d'occurrence de chaque lettre de l'alphabet dans un fichier (sans prendre en compte la case). Il est possible de le faire dans un seul thread, mais afin d'accélérer l'exécution, plusieurs threads seront lancés et chacun s'occupera d'une partie du texte seulement. L'exemple ci-dessous montre la distribution du texte entre les threads.



a) Compléter la fonction *main* afin que celle-ci permette la création de N *threads* (N passé en paramètre). Chaque thread prendra en argument une référence vers une structure *count_param_t,* qui permettra d'indiquer quelle partie du texte le thread doit *parser* (point c). Chaque *thread* exécutera la fonction *count_letters*. Limiter le nombre de threads à 15 au maximum.

b) Compléter la fonction *main* pour récupérer en second argument le nom du fichier que vous voulez utiliser (deux fichier *lorem.txt* et *lorem_small.txt* sont à disposition pour les tests).

c) Créer et remplir les structures count*_param_t* en fonction de la taille de votre texte. Chaque thread doit s'occuper du même nombre de charactères (*chunk*), et le dernier thread s'occupera de sa partie plus les charactères restants (*leftovers*) s'il y en a.

d) Compléter la fonction *count_letters* pour qu'elle parse sa partie de texte, et qu'elle remplisse le champs *counters* de la structure *count_param_t* qui lui est passée. Chaque case de ce tableau *counter* correspond au nombre d'occurrences de la lettre de l'alphabet (en commençant par le 'a') .

e) Compléter la fonction *main* pour y ajouter l'attente de la terminaison des threads et la récupération des résultats contenu dans les structures *count_param_t*. Les résultats finaux doivent être placé dans le tableau *result_counters.* La partie de code affichant les résultats **ne doit pas être modifiée** !

f) Compiler le programme à l'aide du script "build.sh" se trouvant dans le même dossier que les sources

g) Une fois le programme fonctionnel, utiliser la commande "time <cmd> <cmd_args>" pour comparer le temps d'exécution avec 1, 2, 3 et 4 threads et reportez les résultats dans le fichier rapport.md à la racine