SINF1252 - Factorisation de nombres

O. Bonaventure, D. Lebrun, F. Duchêne

25 mars 2015

Un chercheur en mathématiques doit analyser plusieurs fichiers qui contiennent chacun de nombreux entiers non-signés sur 64 bits. Il sait que chacun des nombres de ces fichiers est le résultat du produit de plusieurs nombres entiers (32 bits) premiers. Il doit retrouver parmi tous ces fichiers le seul nombre premier qui n'est utilisé qu'une seule fois comme facteur premier pour l'ensemble des fichiers.

Votre objectif est de l'aider en concevant un programme qui utilise des threads pour accélérer la recherche sur ordinateur multiprocesseur. Plus précisément, votre programme devra pouvoir :

- charger plusieurs fichiers contenant des nombres depuis l'entrée standard (`stdin`), le système de fichier local ou des serveurs Internet en utilisant la librairie `libcurl` [1]
- factoriser chacun de ces nombres en facteurs premiers
- déterminer le facteur premier qui est présent une seule fois dans l'ensemble des fichiers considérés

Les fichiers qui vous seront fournis contiennent les nombres entiers non-signés sur 64 bits en représentation *BigEndian*. Vous devez utiliser les fonctions de la librairie *endian(3)* [2] pour convertir ces nombres en représentation locale.

Votre programme devra supporter les arguments suivants :

- **–maxthreads *n*** le nombre maximum de threads que le programme pourra utiliser
- **-stdin** indique que des nombres seront aussi lus via l'entrée standard
- **file** indique un nom des fichiers à lire (le programme ne doit pas se planter si un des fichiers passés en argument n'est pas accessible ou provoque une erreur de lecture)
- **http ://www.example.org/fichier** indique un URL de fichier à charger depuis le réseau

Votre programme supportera un nombre quelconque de noms de fichiers/URLs en argument.

A la fin de son exécution, il retournera l'information suivante en ASCII sur la sortie standard :

1. Voir `http://curl.haxx.se/libcurl/`. `curl` est une librairie qui implémente de nombreux protocoles réseaux et offre de très nombreuses fonctionnalités qui sortent du cadre de ce cours. Pour ce projet, il vous suffit d'utiliser le code d'exemple qui est disponible via `http://curl.haxx.se/libcurl/c/fopen.html` pour avoir une interface similaire aux fonctions standard *fopen(3)*, *fread(3)*, *fgets(3)*, *feof(3)*, *fclose(3)*, et *rewind()* tout en manipulant des fichiers stockés sur des serveurs Internet.
2. Voir `http://man7.org/linux/man-pages/man3/endian.3.html`

— **Première ligne :** nombre premier unique [3] trouvé en analysant les fichiers passés en argument
— **Deuxième ligne :** nom du fichier dans lequel le facteur premier a été trouvé
— **Troisième ligne :** durée d'exécution en secondes et millisecondes

A titre d'exemple, considérons le programme qui reçoit les fichiers suivants :
— **Fichier1** : 7, 28, 20, 9
— **Fichier2** : 36, 17, 100, 16

Utilisé sur le **Fichier1** uniquement, le programme trouvera 5 comme facteur premier. Si il est utilisé avec comme arguments **Fichier1** et **Fichier2**, alors il retournera 17.

Délivrables

Ce projet se déroule en trois phases. La première phase est une phase d'architecture, vous devrez présenter aux assistants qui encadrent le cours un document d'une page qui décrit l'architecture que vous proposez pour votre programme. Ce document de deux pages maximum devra décrire :
— l'architecture globale de votre programme
— les différents threads utilisés et leurs rôles
— les mécanismes de synchronisation entre ces threads
— les principales structures de données et la façon dont les facteurs premiers calculés sont échangés
— l'algorithme utilisé pour factoriser un nombre entiers. Cet algorithme peut être simple et issue de la littérature pour autant que vous citiez vos sources. L'écriture d'un algorithme de factorisation efficace n'est pas un objectif de ce projet. Par contre, le temps d'exécution de votre programme doit diminuer lorsque le nombre de threads utilisés sur un ordinateur multiprocesseur augmente. Idéalement, vous devriez pouvoir supporter très grand nombre de processeurs.

Le feedback sur le document d'architecture se déroulera durant la semaine du 20 avril, à raison de 10 minutes par groupe d'étudiants.

La deuxième phase est la phase de codage proprement dite. Vous écrirez votre programme et testerez ses performances en faisant varier le nombre de threads sur les serveurs du département et les machines des salles informatiques. Cette analyse de performances sera reprise dans votre rapport final. La deadline pour la remise du travail sur le site `http://crp.info.ucl.ac.be/SINF1252` est fixée au mercredi 6 mai 2015 à 20h00.

La troisième phase est la phase de review. Vous évaluerez le code et le rapport soumis par deux autres groupes d'étudiants. La deadline pour les reviews est fixée au mercredi 13 mai 2015 à 20h00.

3. Si tous les facteurs premiers que vous avez calculé sont utilisés au moins deux fois, alors vous devez retourner `EXIT_FAILURE` via *exit(3)* et ne rien afficher sur la sortie standard.