[LSINF1252] Système informatique : Fractales

Julian Roussieau, Damien Vaneberck

Mai 2016

1 Architecture

L'implémentation de notre programme se découpe en plusieurs parties, la première est la lecture de la ligne de commande qui est passée au programme, la création des fractals, ensuite vient l'étape de traitement des fractals.

Le programme va calculer les fichiers en entrée, si l'option -d est activée, il est possible d'entrer directement les fractales.

Si se sont des fichiers qui sont lu, un thread est lancé pour chaque fichiers à lire, ce thread lit chaque ligne une à une et retourne la fractal correspondante. Pour se faire, nous avons utiliser la fonction fopen et non la fonction open du cours. fopen nous semblait bien plus facile à utiliser car celle-ci permet de récupérer une ligne directement depuis le fichier, alors que open, récupère un caractère à la fois ou un groupe de caractère dont nous devons définir la taille à l'avance.

Lors de chaque lecture d'une ligne, nous apellons la méthode lineToFractal sur la fractal lue et ensuite, nous utilisons la fonction push dessus afin de la mettre dans la pile.

D'un autre coté, nous avons des threads de calculs qui vont calculer les fractales et retenir ne que celle avec la plus haute moyenne.

Pour arrêter les threads de calcul, nous avons caché des noeuds killeurs au fond de la stack, lorsque les threads de lecture ont fini, on augmente la sémaphore pour permettre à ceux de calculs de charger ces noeuds et donc d'appeler "pthread_exit(NULL)";

2 Performance

3 Test Unitaire

Dans nos tests unitaires, nous avons effectués plusieurs tests sur les fonctions manipulants nos fractales pour nous assurer du bon fonctionnement de notre programme. Afin de parvenir à cela, nous avons essentiellement testé nos fractales avec les méthodes suivantes :

getName(): Ce test permet de vérifié si la méthode fractal_get_name renvois bien la même chaîne de caractère représentant le nom contenu dans la fractale sous la variable 'name'.

setAndGetValue(): Cette méthode nous permet de vérifier à la fois la méthode fractal_set_value et fractal_get_value, pour se faire, nous définissions une 'value', nous la passons en argument dans set avec la fractal et ensuite nous effectuons le test sur get en comparant avec la 'value' de départ. Cela nous permet de voir si la valeur a bien été insérée et si elle est bien retournée.

Ensuite viennent encore d'autres tests, vérifiant à chaque fois si les méthodes get de chaque valeur de la structure fractal est bien retourné. Ces méthodes fonctionnant de la même façons que les deux méthodes ci-dessus, il nous parait pertinent de ne pas les ré-expliquer.

calculFractal(): Cette vérification permet d'être sur que quand nous calculons une fractal, nous obtenons bien la moyenne de celle-ci. Pour se faire, nous avons calculé à la main la moyenne d'une fractal à la main et nous comparons se résultats avec la valeur retournée par 'calculDeFractal' avec les mêmes arguments.

lineToFractal(): Ce test permet de vérifier si la méthode lineToFractal transforme bien une ligne définissant une fractal en une structure fractal. Pour se faire, nous vérifions si chaque argument de la ligne est égal à la sa valeur correspondante dans la structure.

4 Difficultés rencontrées

L'élaboration de ce projet fut plus compliqué que celle du premier projet, MyMalloc. Nous avons particulièrement eu du mal au début pour avoir une idée précise du déroulement de l'algorithme. La gestion des thraeds nous paraissait délicate à manier.

Un autre soucis fut de régler un problème de fuite de mémoire avec notre implémentation, vu que nous n'utilisons pas de mutex pour gérer nos thraeds, nous avons eu des soucis sur des ordinateurs plus puissant. Sur des ordinateurs plus lent, nous n'avons pas de segmentation fault mais sur des ordis bien plus puissant, il nous arrivait souvent des erreurs de segmentations faults même si l'algorithme fonctionnait bien à certains moments.

5 Conclusion

En conclusion, l'élaboration de ce projet fût particulièrement intéressante. D'un niveau plus élever que MyMalloc, nous avons du apprendre à maitriser des concepts plus poussé en C tels que les threads en faisant attention à ce que des données ne se marchent pas dessus ou encore gérer la redirection de la sortie standard. En définitive ce projet fût un défi très intéressant à relever pour améliorer notre comprehenssion du language C.