Projet programmation concurrence et parrallelle

Axel Viala <axel.viala@darnuria.eu>

lundi 13 janvier 2020

Objectifs : Ce travail pratique à pour but de vous initier à la programmation concurrente et de mettre en pratique les concepts de théories des systèmes d'exploitation que nous avons vu ensemble.
Vue d'ensemble : Vue d'ensemble : On souhaite écrire un programme qui compte avec plusieurs threads les nombre de pixel noirs et en bonus fait des manipulations d'images dans une image au format PixMap.
Rendus : J'attends dans votre rendu votre code, mais aussi un court document de maximum une page recto-verso a4, expliquant vos choix de réalisation et relatant les difficultées que vous avez rencontré et ce que vous aurriez aimer faire mais n'avez pas réussi.

Consignes : L'image devra être lue en format binaire et gérer un encodage des pixels sur 24 bits, 8 bits pour chaque canal bleu, jaune, rouge. Je vous reocmmande de bien lire la page Wikipédia sur le format PortablePixMap (PPM).
Vous pourrez compiler votre projet à l'aide de la commande suivante : `gcc -std=c11 -pthread -Werror -Wall -Wextra main.c -o superpixmanip`
Exigences : Attention j'exige que votre code compile sans erreur mémoire, ni warning de compilation avec `-Wall -Wextra`.

J'exige aussi que le code soit documenté, vous pouvez suivre le standard de documentation JSdoc ce format est reconnu par de nombreux éditeurs de texte.

Norme C : Coté norme de programmation vous pouvez déclarer et directement assigner ou vous voulez, cependant une bonne organisation du code est attendue. Si vous avez besoin de tableau aggrandissable codez le ;).

Indices : Gros conseil : Tentez d'utiliser des structures `struct` et des fonctions pour les manipuler. C'est la clef pour être écrire des abstractions en C. ;)

Faites des fonctions simples bien documenté avec un nom explicite, pensez à écrire des structures quand ça deviens complexe. Écrivez des test de vos fonctions, et ou utilisez la fonction `assert` pour avoir des vérifications à l'execution.

Évitez les pointeurs de pointeurs bien souvent on se plante.

1 Questions

1. Écrire une structure `pixel_t` qui represente un pixel dans votre image, un pixel est représenté par 3 canaux un pour le rouge, un pour le vert un pour le bleu, l'intensité du canal va de 0 à 255. Indice : Un canal d'un pixel va de 0 à 255 quelle nombre de bit avez vous besoin ?
2. Écrire des fonctions pour obtenir chacun des canaux de votre `pixel_t`.
3. Écrire une fonction pour verifier que deux `pixel_t` sont égaux entre eux.
4. Écrire une struct `ppm_image_t` qui represente votre image ; vous aurez besoin de quoi sauvegarder la hauteur `height`, largeur `width` et un tableau de pixels avec sa taille `length` Indice : Le type standard `size_t` peut être utile.
5. Écrire des fonctions pour acceder à la hauteur, largeur et taille du tableau de pixel depuis la structure `ppm_image_t`,
   Indice : vous pourrez marquer const le pointeur vers votre structure car nous ne modifions pas la struct.
6. Écrire une fonction pour acceder à un pixel depuis la structure `ppm_image_t`.

7. Réussir à lire en *binaire* un fichier au format PPM avec des pixels sur 24 bits et les charger dans la structure `ppm_image_t`.
8. Afficher dans le terminal la valeur d'un pixel lu et chargé en mémoire depuis un fichier ppm.
9. Écrire une fonction pour compter les pixel noirs à un threads puis à 2 puis à n threads. Attention reflechisez bien à comment vous découper le travail. ;)

2 Bonus

À venir dans la semaine.