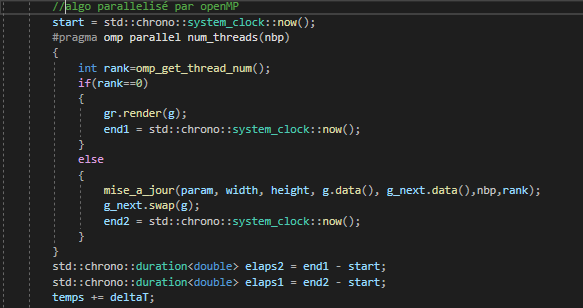
**Compte rendu TP IN203 Alexandre Cor**

**Algorithme de base**

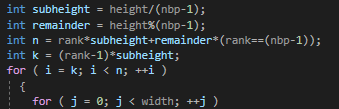
L’algorithme met donc en tout plus de 23ms pour réaliser une itération.

## **Parallélisation de boucle en mémoire partagée**

On change l’algorithme pour qu’un thread s’occupe de l’affichage et que tous les autres s’occupent du calcul de la prochaine itération de galaxie.



Division du travail, 1 thread sur l’affichage, les autres sur le calcul. On note rank en argument de mise\_a\_jour.



Parallélisation de la boucle for dans la fonction mise\_a\_jour.

Avec cette méthode on obtient les résultats suivants :

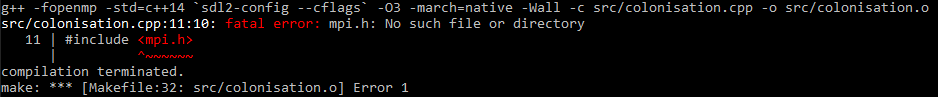
On observe une nette diminution du temps de calcul et une légère augmentation du temps d’affichage. On pouvait s’y attendre car un seul thread s’occupe de l’affichage. Le temps total d’une itération est donc d’approximativement 6.3 ms (on prend le plus grand des deux car il est limitant) et ceci pour 4 threads en tout. On s’aperçoit d’ailleurs qu’augmenter le nombre de thread ne change rien et 4 threads semblent être plutôt optimisé.

On note donc un speed-up de 3,9.

On pourrait penser qu’il y a un problème de data race pour le calcul de end2, mais en fait comme on ne veut garder que la dernière valeur car on veut garder le temps du thread le plus lent, c’est un faux problème.

## **Parallélisation en mémoire distribuée**

Pour la mémoire distribuée, le but était le même, un processus s’occupe de l’affichage et les autres du calcul. En essayant de faire marcher le programme que j’ai écris, je n’arrivais pas à compiler.



J’ai donc d’abord essayé de former mon programme de sorte à ce que le premier processus affiche et calcul puis affiche les temps de calcul pendant que l’autre processus fait la mise à jour mais je ne peux savoir si cela fonctionne effectivement.

