Informatique embarqué

TP n°4 - MION Giovanni

Temps estimé: 2 heures

Temps effectif: 2,5 heures

I)Compilation du code

Compilation avec liaison des dynamique librairies:
make

avant exécution faire:
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:.

Compilation avec liaison statique des librairies:
make static

Suppression des fichiers temporaires:
make clean

Suppression de l'ensemble des fichiers générés par make:
make mrproper

NB: le makefile ne fonctionne qu'avec le compilateur GCC.

II)Choix de l'implémentation

L'algorithme utilisé dans la librairie colimacon est de remplir les cases du tableau par ordre croissant de valeur. Cet algorithme est utilisé car il n'accède qu'une seule fois à chaque case du tableau et qu'il est simple à mettre en place.

Cependant il n'est pas optimum car il fait intervenir 4 boucles for incluses dans une boucle while.

Cela veut dire que l'on pourrait minimiser le nombre de tests et de variables en utilisant une formule utilisant les indices de chaque case pour calculer la valeur à y placer.

III)Problèmes rencontrés

Voici une liste des problèmes rencontrés:

- Organisation du makefile
- Restructuration du code suite aux modifications

IV)Question du TP

Accès Mémoire:

Cet algorithme n'est pas optimal pour les accès mémoires. En effet même s'il n'accède aux cases du tableau qu'une fois, il le fait dans un ordre non linéaire. Lors du chargement en cache de ces cases, elle sont charger par bloc de manière linéaire. Cela a pour effet de provoquer de nombreux défaut de cache. Un algorithme avec un parcours linéaire serait donc plus éfficace.

Parallélisation:

L'algorithme utilisé n'est pas parallélisable. En revanche pour obtenir un algorithme parallélisable on peut si on a une équation pour trouver la valeur en fonction des indices la même fonction qui prend en paramètre un offset les lignes et un autre pour les colonnes.

On appelle alors cette fonction, après initialisation du tableau, depuis plusieurs threads en spécifient dans chaque thread l'adresse du sous tableau que l'on veut traiter, le nombre de lignes et de colonnes du sous tableau et l'offset des ligne et des colonnes (position du premier élément du sous tableau par rapport au premier élément du tableau). Alors chacun des threads peut traiter une partie du tableau séparément.

Temps requis pour un tableau:

J'utilisons la commande time pour mesurer le temps d'exécution du programme.

Le temps repporté ici est le temps CPU systéme de l'execution du programme main (sans printf) sur Nivose.

10x10:	0m0.002s
100x100:	0m0.002s
1000x1000	0m0.007s
10 000x 10 000	0m0.230s

Mémoire alloué:

Pour connaître la taille en mémoire du programme, on va regarder les valeur des adresses du processus qui lui est dédié dans /proc/PID/maps sur Lucien. Les valeur suivante sont celle du tas (heap) du processus.

10x10	135168 bits
100x100	172032 bits
1000x1000	4001792 bits
10 000x10 000	400003072 bits

V)Commentaire d'un autre étudiant

Sachant que ce TP est une relecture personnelle du TP1, il n'a pas été relue par un autre étudiant.