TP de programmation système : les processus légers

Le Jeu de la Vie

Le "Jeu de la Vie" a été inventé par le mathématicien John Horton Conway au début des années 70, et consiste à observer l'évolution d'une grille à deux dimensions dont chaque case est soit occupée par une cellule, soit vide. Chaque case possède (au plus) huit voisines et, d'une génération à l'autre, des naissances et des décès se déroulent mécaniquement dans ces cases selon le nombre de cellules dans les cases voisines. Il existe plusieurs règles définissant ces conditions de vie et de mort, nous utiliserons ici les plus simples :

- si une case est vide et que trois de ses voisines sont occupées par une cellule, alors une cellule apparaîtra dans cette case à la génération suivante ;
- si une case contient une cellule, cette cellule ne restera vivante à la génération suivante que si deux ou trois cases voisines sont occupées ;
- dans tous les autres cas, la case se retrouvera vide à la génération suivante.

Le but du TP est de simuler le Jeu de la Vie en utilisant des threads. Pour cela, on créera un thread pour chaque case de la grille. Chacun de ces threads répètera à l'infini les actions suivantes :

1. Calculer le nouvel état de sa case en fonction de ses cases voisines[1].
2. Attendre que tous les autres threads aient fini ce calcul.
3. Mettre à jour l'état de sa case.
4. Attendre que tous les autres threads aient fini de mettre à jour leur case ; le dernier thread à mettre à jour une case affichera alors la grille complète[2].

De façon plus précise, votre programme comportera les constantes suivantes

```
#define NB_COLONNES 20 // Nombre de colonnes de la grille
#define NB_LIGNES   10 // Nombre de lignes de la grille
```

L'état courant de la grille sera contenu dans un tableau à deux dimensions, déclaré en variable globale :

```
char tab[NB_LIGNES][NB_COLONNES];
```

Chaque case de ce tableau contiendra soit le caractère `'x'`, si la case contient une cellule, soit le caractère `'-'` si la case est vide. Ce tableau sera initialisé en lisant `NB_LIGNES*NB_COLONNES` caractères sur le flux d'entrée standard[3].

Pour synchroniser les différents threads (à la fin des étapes 1 et 3), vous pouvez utiliser un compteur et une variable de condition de la façon suivante :

- le compteur est initialisé à 0 ;
- chaque thread incrémente le compteur lorsqu'il a fini une étape ;
- à ce moment, si le compteur a une valeur inférieure au nombre total de cases (c'est qu'il reste encore des threads n'ayant pas terminé leur étape) alors il faut mettre le thread en attente en faisant un "wait" sur la variable de condition,
- sinon (c'est que tous les threads ont fini leur étape) il faut réveiller tous les threads en attente en faisant un "broadcast" sur la variable de condition... et re-initialiser le compteur à 0 pour la prochaine synchronisation.

Attention : l'accès au compteur et à la variable de condition devra être protégé par un verrou d'exclusion mutuelle "`mutex`".

[1] Vous pouvez pour cela utiliser la fonction `etat_suivant(i,j)` définie dans le fichier `utilitaires.c`.
[2] Vous pouvez pour cela utiliser la procédure `affiche()` définie dans le fichier `utilitaires.c`.
[3] Vous pouvez pour cela utiliser la procédure `init()` définie dans le fichier `utilitaires.c`.