Fils et semaphores

3 décembre 2020

Exercice 1 : Spinlock v. Mutex.

Quelle est la différence entre un *spinlock* et un *mutex*? Pour mettre en évidence cette différence, téléchargez `http ://www.lsv.fr/~hondet/resources/archos/spinvmut.c` et complétez le de manière a ce que le fil `stupid` s'accapare le spinlock et que `wants` soit forcé d'attendre la libération de la ressource (exclusion mutuelle).

La commande `time cmd` mesure le temps d'occupation CPU en mode kernel (*sys*), le temps d'occupation CPU en mode utilisateur (*user*) et le temps réel (*real*), écoulé, présenté sous la forme

```
real    0m0.000s
user    0m0.000s
sys     0m0.000s
```

si vous utilisez la command *bash*. Il existe aussi un binaire `time`, qui présente les résultats de cette manière,

```
0.00user 0.00system 0 :00.00elapsed ?%CPU (0avgtext+0avgdata 1808maxresident)k
```

où le temps réel est noté *elapsed*.

Que donnera un appel à `time spinvmut` concernant le temps réel et le temps CPU ? Modifiez le code pour utiliser des mutex à la place. Comment devrait évoluer le temps réel et le temps CPU, en lançant `time spinvmut` ? Vérifiez.

Exercice 2 : La fonction d'Hénon, partie II.

On va (re) calculer l'orbite d'un système dynamique de dimension 2. La fonction d'Hénon est définie par le système

$$H_{a,b} = \begin{cases} x_{n+1} = a - by_n - x_n^2 \\ y_{n+1} = x_n. \end{cases} \tag{1}$$

On utilisera un thread pour calculer la suite $(x_n)_n$ et un autre thread pour la suite $(y_n)_n$.

Proposez un moyen de synchronisation permettant d'assurer l'entrelacement des calculs de x_n et de y_n. Mettez le en œuvre.

On peut ensuite créer un modèle dit producteur consommateur :

- un thread calcule l'orbite et stocke les données dans un buffer tournant,
- l'autre thread lit les données du buffer et les écrit dans un fichier.

Le fichier `henon.dat` sera sous la forme `0.3415 1.2451` où le premier nombre est x_n et le deuxième y_n.

On pourra tracer la fonction avec la commande `gnuplot henon.p` après avoir téléchargé le script `http ://www.lsv.fr/~hondet/resources/archos/henon.p`. Cela produira un fichier `henon.png`.

Pour $a = 1\ .4$ et $b = -0\ .3$, la fonction est chaotique (au sens de Devaney), c'est-à-dire,

- la fonction est sensiblement dépendante des conditions initiales (effet papillon, imprédictabilité) ;

- la fonction est topologiquement transitive (indécomposabilité) ;
- les points périodiques sont denses dans le domaine de définition de $H_{a,b}$ (régularité).

et possède un attracteur étrange.

Pour plus d'informations sur les systèmes dynamiques, voir *An introduction to Chaotic Dynamical Systems*, Robert L. Devaney, Westview.

Exercice 3 : Mandelbrot.

Soitcun nombre complexe. On considère la série $z_0 = 0$ et $z_{n+1} = z_n^2 + c$ pour $n \geq 0$. L'ensemble de Mandelbrot est défini comme l'ensemble des valeurs c telles que la série des z_n est bornée. On sait que cela est le cas si z_n ne sort jamais d'un cercle de rayon 2 autour de 0. Si jamais la série sort de ce cercle, soit $m(c)$ le plus petit indice n tel que c'est le cas. Une application populaire pour $m(c)$ est de créer de jolies images ; on associe l'écran avec un rectangle et chaque pixel avec la valeur c qui y correspond ; le pixel est ensuite peint avec une couleur correspondant à $m(c)$. La page web du cours contient un tel programme. Votre tâche consiste à l'accélérer en lançant plusieurs threads en parallèle. On peut utiliser `cat /proc/cpuinfo` pour voir combien de cœurs une machine a. Chaque thread va donc travailler sur une partie différente de l'image.

Exercice 4 : Sémaphores inter processus (facultatif).

Les semaphores, bien qu'introduits avec les threads, permettent également la synchronisation inter processus. Pour ce faire, l'API System V fournit des procédures *IPC* pour *Inter Process Communication*. Par exemple, on peut obtenir des sémaphores via `semget`. Cette commande utilise des clefs IPC permettant d'identifier de manière unique un ensemble de sémaphores.

Ces clefs peuvent être générées par la fonction `ftok`.

Pour expérimenter un peu, écrivez un programme créant une clef et deux sémaphores avec cette clef. Lancez le programme et exécutez la commande `ipcs -s`. Que constate-t-on ? On en déduira les précautions à prendre lorsque l'on utilise des clefs. La commande `ipcrm` existe.

On peut maintenant implémenter la fonction d'Hénon multi processus avec des sémaphores. L'utilisation de sémaphores partagés reste assez fastidieuse.