TD 3 - ASR7 Programmation Concurrente

—

Ordonnancement, Travail à la chaîne

Matthieu Moy + l'équipe ASR7 (cf. page du cours)

Printemps 2020

I Ordonancement avec des mutexes

Nous utilisons un ordonnancement préemptif avec priorité (Plus la valeur de priorité est importante plus la tâche est prioritaire). Nous allons utiliser un jeu de tâches qui mélange des tâches périodiques et des tâches ponctuelles. De plus, deux tâches partagent un mutex.

Tâche	Date(s) d'arrivée(s)	Priorité	Durée	Remarque
A	0, 6, 12, 18, 24, 30	10	1	Périodique
B	0, 10, 20, 30	8	4	Périodique, à chaque itération, la tâche doit acquérir le mutex à la fin du temps 1 et la libérer à la fin du temps 3.
C	18	5	6	Ponctuelle
D	0	1	8	Ponctuelle, à la fin du temps 6, la tâche acquiert le mutex et le conserve jusqu'à la fin du temps 8.

Q.I.1) - Faire l'ordonnancement de ces tâches sur 32 unités de temps.

Q.I.2) - Quel est le temps de réponse (ou latence) de chaque tâche ?

Q.I.3) - Que remarque-t-on aux temps 21 à 27 ?

II Travail à la chaîne

Vous devez donner un programme qui simule un travail à la chaine. Il s'agit d'organiser le travail à l'usine de Doubitchous Preskovitch S.A. à Sofia.

Le travail est décomposé en 6 étapes :

1. mélanger la farine et le cacao ;
2. pétrir la pâte avec la margarine et l'huile ;
3. ajouter le sucre et la cannelle ;
4. ajouter un morceau de chocolat ;
5. rouler le doubitchou ;
6. cuire le doubitchou.

Chaque thread est en charge de l'une d'entre elles. Le travail se fait à la chaîne c'est-à-dire que le thread en charge de la cuisson ne peut le faire tant que le doubitchou n'est pas correctement roulé par le thread qui se trouve à la position précédente dans la chaîne.

Votre programme doit lancer le nombre de threads nécessaires (`NB_ETAPE`) puis les faire agir à la chaîne pendant la création de `nb_threads`. À la fin, le programme doit afficher le temps passé à la confection (c'est-à-dire la somme des temps passés par chaque thread).

Pour faire ce travail, vous disposez déjà des fonctions :

— `double travail(int pos, int num_tours)` qui effectue le travail pour le thread à la position `pos` dans la chaîne ; de plus cette fonction affiche un message permettant de savoir ce qu'on fait. Par exemple, pour le thread 5 durant la confection du 3e doubitchou (le numéro 2), cette fonction affichera : `2 : le thread 5 cuit le doubitchou`. Cette fonction mesure aussi le temps nécessaire à sa tâche et le retourne.

— Un tableau d'entiers `nb_doubs` représente le nombre de doubitchous avant chaque étape. `nb_doubs[0]` est le nombre de doubitchous pas encore démarrés, et `nb_doubs[nb_etape]` est le nombre de doubitchous terminés.

— Vous disposez aussi de la fonction `travail_chaine()` qui effectue la tâche. L'algorithme utilisé par ces threads est simple, tant que le nombre voulu de doubitchou n'est pas préparé, le thread :
  — attend que le thread précédent ait fini sa préparation (fonction `attend()`) ;
  — effectue sa tâche (fonction `travail()`) ;
  — s'il n'est pas le dernier, il transmet son travail au suivant (fonction `transmet()`).
  — S'il est le dernier, il pose le doubitchou terminé sur le plat (qui correspond à la dernière case du tableau `nb_doubs`), en appelant la fonction `termine()`.

```
1   double travail_chaine(chaine *c, int pos, int nb_tours, int nb_etape) {
2     int i;
3     double temps = 0;
4     for (i=0; i<nb_tours; i++) {
5       c->attend(pos);
6       temps += travail(pos, i);
7       if (pos != nb_etape-1) {
8         c->transmet(pos+1);
9       } else {
10        c->termine();
11        fprintf(stdout, "doub %d : le doubitchou est terminé\n", i);
12      }
13    }
14    return temps;
15  }
```

Dans cette fonction, `this` est la structure de donnée nécessaire à la synchronisation (le moniteur), `pos` la position du thread dans la chaîne, `nb_tours` le nombre de tours demandés et `nb_etape` le nombre d'étapes nécessaires à la préparation.

Q.II.1) - Expliquez la manière dont vous allez synchroniser les différents threads : les primitives utilisées, l'algorithme des fonctions `attend()` et `transmet()`.

Q.II.2) - Donnez le code des fonctions d'initialisation et de destruction.

Q.II.3) - Donnez le code de la fonction exécutée par chacun des threads.

Q.II.4) - Donnez le code de la fonction principale qui lance les threads et récupère leur résultat.

Q.II.5) - Donnez le code des fonctions `attend()` et `transmet()`.

Q.II.6) - Donnez le code de la fonction `termine()`.

Pour élaborer 3 doubitchous le programme que vous fournissez doit par exemple afficher :

```
0 : le thread 0 mélange la farine et le cacao
1 : le thread 0 mélange la farine et le cacao
0 : le thread 1 pétrie la pâte avec la margarine et l'huile
2 : le thread 0 mélange la farine et le cacao
[...]
0 : le thread 5 cuit le doubitchou
1 : le thread 4 roule le doubitchou sous les aisselles
le doubitchou 0 est terminé
2 : le thread 2 ajoute le sucre et la cannelle
[...]
le doubitchou 2 est terminé
Le thread 0 s'est terminé et a travaillé pendant 0.038278
Le thread 1 s'est terminé et a travaillé pendant 0.023248
[...]
La somme est 0.154813 secondes
```