Université Paul Sabatier
31062 Toulouse Cedex 09

Master 1 Informatique - MCPR

Modèles et Concepts du Parallélisme et de la Répartition
Travaux pratiques n° 6

Threads Java et synchronisation du type « moniteur »

On s'intéresse ici au modèle des lecteurs-rédacteurs étudié en cours et en TD.

Deux familles de processus (les lecteurs et les rédacteurs) accèdent en parallèle à des informations contenues dans un fichier partagé (ou une base de données).

Les lecteurs désirent seulement consulter l'information. Les rédacteurs désirent modifier cette information. Les lecteurs peuvent donc accéder à l'information en parallèle alors que les rédacteurs doivent y accéder en exclusion mutuelle.

Exercice

En reprenant les principes de synchronisation déjà vus en TD, programmer une application pour chacune des variantes présentées ci-dessous (les mêmes qu'en TD). Dans cette application, des threads Java, représentant `NL` lecteurs et `NR` rédacteurs, synchronisent leurs accès à un fichier partagé sachant que `NL` et `NR` peuvent varier d'une exécution à l'autre.

Les classes `Lecteur` et `Rédacteur` permettront de créer les threads lisant ou écrivant des informations dans le fichier partagé.

L'accès au fichier partagé sera géré par une instance de la classe `Fichier` assurant le rôle d'un « moniteur » et proposant les opérations permettant de lire et écrire des informations dans ce fichier de manière synchronisée et donc cohérente.

Pour simplifier, le fichier partagé considéré pourra être l'écran.

Version 1

Lecteurs et rédacteurs ont même priorité, les lectures peuvent se faire en parallèle, une écriture est exclusive. Lorsque un rédacteur a terminé d'écrire, il essaye d'activer prioritairement un rédacteur.

Version 2

Un rédacteur est prioritaire par rapport à un lecteur en attente pour l'accès au fichier.

Version 3

Dans cette variante, un rédacteur qui termine d'écrire doit laisser l'accès en priorité à tous les lecteurs en attente à cet instant, et non au rédacteur suivant comme il est spécifié dans la variante 1. En revanche, les lecteurs qui arriveront ultérieurement (après cette demande d'écriture) devront respecter la règle de priorité des rédacteurs sur les lecteurs telle qu'elle est exprimée dans la variante 2.

Version 4

On suppose maintenant que l'accès aux données est effectué suivant une politique FIFO ; les requêtes de lecture et d'écriture sont traitées dans l'ordre de leur arrivée.

Remarque : Contrairement à l'opération length utilisée dans les moniteurs de Hoare, la classe Condition de Java ne prévoit aucune méthode pour connaître le nombre de threads bloqués à un instant donné dans la file d'attente associée à une variable condition. Il faut donc trouver un moyen de compenser cette absence...