Université Toulouse III – Paul Sabatier
118 route de Narbonne
31062 Toulouse cedex 9

Travaux dirigés – n°3 – Moniteurs de Hoare

Modèle des lecteurs-rédacteurs

Le modèle des lecteurs-rédacteurs schématise une situation rencontrée dans la gestion de fichiers partageables ou dans l'accès à des bases de données.

Deux familles de processus accèdent à ces informations. 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.

Les comportements de ces processus sont donc les suivants :

Un lecteur	**Un rédacteur**

Boucler sur {
...
Demander à lire;
Faire la lecture;
Signaler la fin de lecture;
...
}
	Boucler sur {
	...
	Demander à écrire;
	Faire la modification;
	Signaler la fin d'écriture;
	...
	}

On se propose d'écrire un moniteur de Hoare gérant l'accès à la ressource commune selon la politique définie précédemment et selon les différentes variantes suivantes.

La spécification du moniteur se présente de la manière suivante:

```
Moniteur Lecteur_Redacteur {
  void debutLire();
  void finLire();
  void debutEcrire();
  void finEcrire();
end Lecteur_Redacteur ;
```

Variante 1

Quand aucun lecteur ne lit, les lecteurs et les rédacteurs ont la même priorité. En revanche, dès qu'un lecteur est en train de lire, tous les autres lecteurs qui le demandent peuvent également lire puisque les lectures peuvent être faites en parallèle, quel que soit le nombre de rédacteurs en attente.

Lorsqu'un rédacteur écrit, aucun autre client ne peut accéder à la ressource (ni lecteur, ni rédacteur).

Lorsque le rédacteur a terminé d'écrire, il essaye d'activer un rédacteur en priorité sur les lecteurs.

Variante 2 - Priorité des rédacteurs sur les lecteurs

On souhaite donner la priorité aux rédacteurs afin que l'information disponible ne soit pas obsolète pour le lecteur. Lorsqu'un rédacteur demande à accéder à la ressource, il doit donc l'obtenir le plus

tôt possible. Bien sûr, il ne lui est pas possible d'interrompre des lectures ou une écriture en cours. De même, il n'a pas de passe-droit vis-à-vis d'autres rédacteurs en attente (arrivés avant lui et non encore acceptés). En revanche, il est prioritaire par rapport aux lecteurs en attente.

Variante 3 - Gestion plus équitable des accès

À quelles situations erronées peuvent conduire les variantes 1 et 2 ?

Quelle solution proposer pour corriger de telles situations ?

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.

Variante 4 - Gestion FIFO des accès

On suppose maintenant que l'accès aux données est effectué suivant une politique FIFO ; les requêtes sont traitées dans l'ordre de leurs arrivées.

Pour ordonner globalement les lecteurs et les rédacteurs en attente, tous les processus seront bloqués sur une même condition. En effet, il n'est pas possible de savoir si le 3^{e} rédacteur est arrivé avant ou après le 4^{e} lecteur lorsque lecteurs et rédacteurs sont rangés dans des files distinctes. On notera que, lors du réveil, il n'est pas possible (pour le « signaleur ») de distinguer un lecteur d'un rédacteur. Aussi, on peut être amené à réveiller un processus (lecteur ou rédacteur), puis le rebloquer par la suite si le déblocage s'avère impossible dans la situation actuelle.

Questions

Pour chacune des variantes :

- ☞ Préciser la spécification du moniteur.
- ☞ Préciser les conditions de blocage et de réveil d'un processus lecteur et d'un processus rédacteur.
- ☞ En déduire les variables d'état et les variables « condition » gérées dans le moniteur.
- ☞ Donner le code du moniteur.