Systèmes D'eXploitation - projet

Yohan Cottin *e1701053* & Rosalie Perée *e2004724*

Dans le cadre du cours de *Système d'exploitation* de la première année de Cyberdéfense à l'ENSIBS, il nous a été demandé d'effectuer un projet sur les sémaphores en Java. Ce document est le rapport portant sur ce projet.

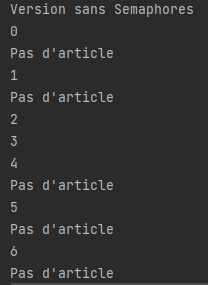
# Producteur-Consommateur

*Deux processus se partagent une mémoire tampon de taille fixe (ex un tableau). Un des processus, le producteur, dépose des éléments dans le tampon. Le deuxième processus extrait l'information (le consommateur). Les problèmes surviennent lorsque :*

* *Le tampon est plein le producteur ne peut plus déposer d'information.*
* *Le tampon est vide le consommateur ne peut plus extraire de l'information*

## Sans sémaphore

Sans sémaphores, le programme fonctionne correctement sauf dans deux cas : si le tampon est plein et qu'on tente d'y déposer de l'information, ou si le tampon est vide et qu'on tente de retirer de l'information.

S'il y a plusieurs producteurs ou consommateurs, ils peuvent essaye d'écrire simultanément dans le tampon, ce qui peut donner à des comportements imprévisibles.

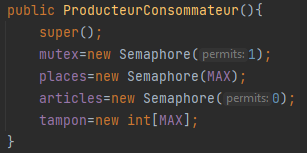
Le sémaphore permet de régler ces problèmes.

Les chiffres sur l'image ci-contre représentent le nombre d'articles total déposé, la phrase "pas d'article" apparaissant lorsque le consommateur tente d'accéder à un tampon vide.

## Avec sémaphores

Pour pallier les problèmes soulevés dans le point précédent, on peut utiliser des sémaphores. On veut vérifier deux conditions :

* Si le tampon est vide, le processus consommateur doit être bloqué jusqu'à ce qu’un producteur dépose un élément.
* L ’accès aux cases du tampon doit être protégé. Le consommateur ne peut extraire l’élément que le producteur est en train de déposer.

Dans le constructeur de la classe nous trouvons les éléments suivants :

* *places* permet de bloquer le producteur quand il n'y a plus de place dans le tampon
* *articles* permet de bloquer le consommateur quand y a pas d'article dans le tampon
* *mutex* permet de s'assurer qu'il n'y a pas d'accès simultané au tampon.

# Messages publicitaires

*Le modèle lecteur(s)-rédacteur : Les lecteurs peuvent consulter l'information qu’un rédacteur construit. Les contraintes sont les suivantes :*

* *Il n'y a pas de limite sur le nombre de lecteurs en parallèle.*
* *Les lectures s'effectuent en exclusion mutuelle avec le rédacteur.*

## Version 1 : priorité des lecteurs sur le rédacteur

Nous disposons d'un *MessageBoard* qui est simplement l'équivalent d'un panneau publicitaire.

Nos lecteurs vont bloquer en écriture le panneau le temps de lire les messages, et de les imprimer dans la console. Pendant ce temps, aucune écriture n'est possible. On peut avoir plusieurs lecteurs simultanément ou à la suite tant que le sémaphore n'a pas étélibéré. Le panneau est bloqué à partir du premier lecteur et débloqué quand plus aucun lecteur ne le lit.

Le rédacteur va écrire sur le panneau en boucle. Il bloque, il écrite cinq messages et il débloque. Les lecteurs peuvent ensuite prendre la suite si besoin. Sinon, le rédacteur continue à bloquer, écrire et puis débloquer.

Le délai entre les affichages des logs dans la console vient du fait que des lecteurs accèdent au panneau et que aucune écriture n'est donc possible.

## Version 2 : priorité du rédacteur sur les lecteurs