TME 6 / TME 7 : SYNCHRONISATIONS PAR SEMAPHORES

REALISATION D'UN MECANISME DE DIFFUSION

On veut réaliser un mécanisme de diffusion. Il y a deux types de processus : les émetteurs (au nombre de NE) et les récepteurs (au nombre de NR). Les émetteurs déposent des messages dans un tampon initialement vide. Chaque message doit être lu par tous les récepteurs avant de pouvoir être effacé.

Les processus émetteurs et récepteurs sont cycliques. Autrement dit, chaque processus est une boucle infinie. Les émetteurs doivent se bloquer lorsqu'il n'y a pas de case libre, les récepteurs lorsqu'il n'y a pas de message à lire. Par contre, on autorise plusieurs récepteurs à lire simultanément un message.

Les synchronisations entre émetteurs et récepteurs sont réalisées au moyen de sémaphores et variables partagées.

LES SOLUTIONS QUE VOUS PROPOSEZ DOIVENT ETRE PROGRAMMEES A L'AIDE DE LA BIBLIOTHEQUE DE MANIPULATION DE SEMAPHORES `libIPC`, dont le descriptif est donné en annexe. Vous pouvez récupérer 2 squelettes de programme dans le répertoire

`/Infos/lmd/2005/licence/ue/li324-2006fev/TME6-7`

1. MISE EN OEUVRE AVEC UN TAMPON A UNE SEULE CASE

On considère d'abord le cas où le tampon ne contient qu'une case. Pour assurer la synchronisation, on définit les sémaphores et variables suivants :

- Le sémaphore `EMET` bloque les émetteurs tant qu'il n'y a pas de case dans laquelle ils puissent écrire.
- Le tableau de sémaphores `RECEP[1..NR]` bloque les récepteurs.
- Le compteur partagé `nb_recepteurs` indique le nombre de consommateurs ayant déjà lu le message produit par le producteur.

1.1.

Quel mécanisme de protection supplémentaire doit être introduit pour assurer la cohérence de `nb_recepteurs` ? Donnez les initialisations des sémaphores et variables partagées.

1.2.

Pourquoi utilise-t-on un tableau de sémaphores `RECEP[1..NR]`, dont chaque case est augmentée de 1 lors de l'émission d'un message, plutôt qu'un unique sémaphore `RECEP`, augmenté de `NR` lors de l'émission d'un message ? Qui prévient les émetteurs de la disponibilité de la case ?

1.3.

Programmez cette synchronisation en utilisant les primitives de la bibliothèque `libIPC`.

2. MISE EN OEUVRE AVEC UN TAMPON A NMAX CASES

On considère maintenant le cas d'un tampon à `nmax` cases, et on souhaite assurer un maximum de parallélisme au niveau de l'accès au tampon. En particulier :

- deux émetteurs doivent pouvoir déposer simultanément des messages s'ils écrivent dans des cases différentes ;
- un émetteur et un récepteur peuvent accéder simultanément au tampon s'ils n'accèdent pas à la même case.

A nouveau, chaque message déposé doit être lu par *tous* les récepteurs. On suppose que le tampon est utilisé de manière circulaire (avec une attribution ordonnée des cases et non avec une liste de cases vides et une liste de cases pleines).

2.1.

Définissez les sémaphores et variables nécessaires pour programmer cette synchronisation. Pour chaque sémaphore et variable vous préciserez son rôle et sa valeur initiale.

2.2.

Donnez les algorithmes des émetteurs et récepteurs. Décrivez brièvement un scénario de fonctionnement du système ainsi synchronisé, en insistant sur le parallélisme des émetteurs entre eux, des récepteurs entre eux, et des émetteurs vis à vis des récepteurs.

2.3.

Programmez cette synchronisation en utilisant les primitives de la bibliothèque libIPC.