Master 1 Informatique - Réseaux et Communications - TD/TP 2
Communication entre processus : Mémoires partagées et sémaphores

1 Gestion d'un service avec une mémoire partagée

On veut mettre en place un système de communication entre un chef de service et les employés de ce service. Ce système consiste à permettre au chef de service d'envoyer des messages (instructions, actualité, blagues, etc.) aux employés pendant les heures de travail. Les messages sont écrits dans une mémoire partagée et chaque poste employé est relié à un processus (qu'on appellera : processus employé ou, simplement, employé) lui permettant de lire un message.

Un segment de mémoire partagée contient un message posté par le chef de service et peut être lu (une seule fois) par tous les employés. Plusieurs messages peuvent être envoyés dans la journée en utilisant le même segment et chaque message doit être lu par tous avant de pouvoir écrire un nouveau message. Le rôle de chaque processus employé (respectivement, processus chef de service) est de lire (resp. écrire) un message. Enfin, chaque lecture/écriture d'un message nécessite une nouvelle exécution d'un processus.

Dans un premier temps, nous supposons un seul employé.

1. Montrer qu'il est nécessaire de mettre en place une exclusion mutuelle entre le processus du chef de service et l'employé.
2. Proposer une solution à l'aide de sémaphore et écrire le schéma algorithmique des processus employé et chef de service. Le programme correspondant est a écrire en TP.

   Nous supposons maintenant qu'il y a n employés, tous doivent lire chaque message envoyé par le chef de service.
3. Modifier les algorithmes précédents pour tenir compte des n employés.
4. Que ce passe-t-il si un $n + 1^{eme}$ processus employé est lancé avant la fin du processus chef de service ? Comment résoudre ce problème ?

2 Rendez-Vous

Il est souvent nécessaire de réaliser un rendez-vous entre processus. Pour lancer un jeu à plusieurs joueurs par exemple, pour synchroniser des calculs, etc.

Proposer une solution utilisant un sémaphore, permettant à n processus d'attendre jusqu'à ce que tous soient présents et qu'ils soient arrivés à un point déterminé dit *point de rendez-vous* de leur code.

3 Traitement synchronisé

On envisage ici le traitement d'une image par plusieurs processus, chacun ayant un rôle déterminé. Par exemple, un processus pourrait faire du lissage, un autre des transformations de couleurs, ou de l'anti-crénelage, etc. D'une façon générale, chaque processus travaille sur un ensemble de points (pixels) de l'image. Appelons ces ensembles *zones*. On peut donc considérer l'image comme une suite de *zones* ordonnées.

Le travail doit se faire :

- avec garantie d'exclusivité : aucun autre processus ne doit accéder à l'ensemble de points en cours de traitement par un processus donné,
- et dans un ordre déterminé entre les processus : sur toute zone, le processus P_1 doit passer en premier, puis P_2 etc.

On se limite d'abord à deux processus. Proposer une solution permettant un fonctionnement correct, sachant que l'image est stockée dans un segment de mémoire partagée.

Pour passer à trois processus (et plus) que peut-on proposer ?

4 En TP

Réaliser l'exercice de la section 2, en affichant la valeur du sémaphore à chaque arrivée d'un processus au point de rendez-vous (utiliser *semctl()*).

Réaliser ensuite l'exercice proposé en section 3, en simulant un temps de travail aléatoire pour chaque processus accédant à la ressource commune. Générer un temps de travail suffisamment long, de sorte à pouvoir montrer que la protection mise en place fonctionne.

Pour les plus avancés, réaliser l'application de la (section 1).