TD/TP objets IPC et synchronisation
Durée moyenne : une séance de 3h

1 Files de message : Utilisation pour mettre en place une exclusion mutuelle

L'objectif de cet exercice est d'utiliser une file de messages (une seule et sans introduire un autre objet IPC) pour mettre en place une exclusion mutuelle. Dans le cas présent, il est question d'éviter que plusieurs processus concurrents utilisent, en même temps, une ressource partagée.

Pour résoudre ce problème, nous mettons en oeuvre une application composée d'un processus P_{ctrl} qui a pour rôle de contrôler les accès à une ressource et d'un ensemble de processus concurrents P_1, P_2, ..., P_N utilisant cette ressource.

Le principe est le suivant : lorsqu'un processus P_i souhaite accéder à cette ressource, il doit adresser une demande à P_{ctrl}. A chaque instant, un et un seul processus P_i peut disposer de la ressource. En parallèle, P_{ctrl} a pour rôle :

1. de recevoir des demandes d'accès,
2. d'annoncer au premier demandeur que la ressource est disponible,
3. de prendre connaissance de la libération de la ressource pour pouvoir l'annoncer à un nouveau processus demandant la ressource (retour à l'étape 1).

1. Décrire le protocole d'échange entre un processus P_i et le processus P_{ctrl} (messages échangés, ordre des échanges et étiquettes des messages, etc.). La solution peut être sous forme algorithmique décrivant la structure des messages et les opérations d'envoi et de réception.
2. A qui revient la création de la file de messages ?
3. Votre solution est elle applicable dans le cas de plusieurs processus P_1,..., P_N ?
4. Votre solution peut-elle aboutir à une situation d'interblocage ? de famine ?
5. à faire chez soi pour le prochain TP : implémenter votre solution.

2 Tableau de sémaphores : Le problème du 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 et permettant à n processus d'attendre jusqu'à ce que tous les processus soient présents et qu'ils soient arrivés à un point déterminé dit *point de rendez-vous* de leur code, ceci avant de poursuivre leur exécution.

Implémenter cette solution, en affichant la valeur du sémaphore à chaque arrivée d'un processus au point de rendez-vous (utiliser *semctl()*).

3 Tableau de sémaphores et segment de mémoire partagé : traitement synchronisé

On envisage le traitement parallèle d'une image par plusieurs processus. Chaque processus a un rôle déterminé. Par exemple, un premier processus pourrait faire du lissage, un second, des transformations de couleurs ou de l'anti-crénelage, etc. Enfin, ces traitements doivent se faire dans un ordre bien déterminé.

Pour pouvoir réaliser les différents traitements en parallèle, l'idée est de diviser l'image en sous-ensembles ordonnés de points (pixels) et de permettre à chaque processus de travailler sur un sous-ensemble (appelé zone) différent. Le travail doit alors se faire de la manière suivante :

— chaque processus doit traiter, dans l'ordre, toutes les zones de l'image,

— avec garantie d'exclusivité : pendant qu'un processus travaille sur une zone, aucun autre processus ne doit pouvoir accéder à cette zone en même temps,
— sur toute zone, les différents traitements doivent se faire dans un ordre déterminé : le processus P_1 doit passer en premier pour traiter cette zone, puis P_2, puis P_3, etc.

1. On se limite d'abord à deux traitements (donc deux processus). Proposer une solution permettant un fonctionnement correct. L'image sera stockée dans un segment de mémoire partagée et pourrait (pour simplifier l'exercice à l'implémentation) être représentée par un tableau d'entiers, ou chaque élément représente une zone et est modifiée par un traitement.

2. Pour passer à trois traitements (et plus), quelle solution proposez vous ?

3. Implémenter progressivement vos algorithmes, en simulant un temps de travail aléatoire pour chaque traitement (une fonction calcul et fournie sur Moodle). Générer un temps de travail suffisamment long, de sorte à pouvoir corriger les éventuelles erreurs et à montrer que la protection mise en place fonctionne.