Master Informatique - Réseaux et Communications - TD/TP 1
Communication inter-processus : files de messages

1 Files de messages : un début

On veut mettre en place une application composée de deux processus : le premier, appelé *client*, génère et envoie des requêtes au second processus, appelé *calculatrice*. La calculatrice résout les requêtes et renvoie les réponses au client. Les requêtes sont des opérations mathématiques simples à effectuer (par exemple : $7 * 5 =?$). Les deux processus (client et calculatrice) communiquent à l'aide d'une **file de message**.

1. Décrire pour chaque processus, comment se passe sa mise en route pour pouvoir utiliser la file.
2. Ecrire le schéma algorithmique des deux processus client et calculatrice. Dans ce schéma, vous devez définir la structure des requêtes et celle des réponses. Les requêtes possibles sont les opérations $+, -, *$ et / avec deux opérandes.
3. Modifier le schéma algorithmique précédent (toujours en utilisant une seule file de messages) pour que la calculatrice puisse répondre aux questions de différents clients.
4. Modifier le schéma algorithmique proposé pour qu'il y ait autant de calculatrice que de types de requêtes, c'est-à-dire une calculatrice chargée de répondre aux requêtes +, une aux requêtes −, une aux requêtes $*$ et une aux requêtes /. Remarque : utilisez toujours une seule file de messages.

2 File de messages utilisée en synchronisation

Un processus P_{file} doit gérer une ressource commune à plusieurs processus, par exemple un espace mémoire commun. L'accès à la ressource passe obligatoirement par une demande adressée à P_{file}, par chaque processus P_i voulant accéder à la ressource.

À chaque instant, un et un seul processus P_i peut disposer de la ressource. En effet, il peut modifier le contenu de l'espace mémoire et le passage par P_{file} permet de garantir l'exclusivité d'accès. Le rôle de P_{file} est :

- de recevoir des requêtes demandant l'accès,
- d'annoncer au premier demandeur que la ressource est disponible,
- de prendre connaissance de la libération de la ressource pour pouvoir l'annoncer à nouveau comme disponible.

On suppose dans un premier temps un fonctionnement sans panne, c'est-à-dire que tout processus P_i accédant à la ressource annonce bien sa fin et relâche la ressource.

On veut mettre en place l'ensemble à partir d'une file de messages. On suppose qu'on dispose d'un accès à la mémoire partagée. Mais on ne dispose pas de primitives relatives aux sémaphores, c'est pourquoi on utilise la file de messages pour réaliser l'exclusivité d'accès. Dans les questions qui suivent, ne pas écrire de programme complet, mais préciser les appels système utilisés lorsque nécessaire.

1. Que faut-il mettre en place (initialisation) pour qu'un tel système fonctionne ?
2. Décrire le fonctionnement général du système.
3. Comment est matérialisée la file d'attente ?
4. Que fait P_{file} pour la prise en compte des requêtes ? En particulier, a-t-il besoin d'entrées-sorties non bloquantes ? Comment est-ce que P_{file} prend connaissance de la fin d'utilisation de la ressource ?

Allons un peu plus loin :

1. On admet maintenant qu'il peut y avoir des pannes. Que se passe-t-il si un P_i disparait ? En particulier, peut-on revenir à une situation correcte ? Pour ce faire, étudier les situations possibles d'un P_i et dire face à chaque situation ce qui peut être fait.
2. Si on ne dispose pas du concept de mémoire partagée, quels choix ferez-vous pour mettre en place un tel partage et satisfaire les requêtes ?

3 En TP

3.1 Un début

Réalisez l'application de la section 1 en suivant les mêmes étapes.
A l'exécution, mettre en œuvre les situations suivantes :

- Lancer l'exécution de plusieurs processus et constater l'existence de la file hors de la vie des processus, avec la commande `ipcs`.
- Rajouter des messages dans la file existante jusqu'à constater le blocage lorsque la file est pleine.
- Exécuter l'application de manière à constater un blocage lorsque la file est vide.
- Dans une situation où la file existe, supprimer et recréer le fichier de calcul de la clé. Relancer des processus clients. Que se passe-t-il ? Proposer une solution pour pouvoir utiliser cette même file par un nouveau processus.

3.2 Décomposition

Si nous avions mis dans le titre *décomposition en facteurs premiers*, une série de grognements l'aurait accueilli. Alors oui, il s'agit bien de la décomposition des nombres en facteurs premiers, en utilisant une file de messages et un nombre quelconque de processus partageant la file et participant à cette décomposition.

Le déroulement demandé est le suivant :

1. Un processus *parent* va déposer dans une file de messages un entier n quelconque. Des processus enfants qu'il engendre appelés *décomposeurs* vont prendre en charge la décomposition et la restitution du résultat. Le nombre d'*enfants* devra être paramétré.
2. Chaque décomposeur dépose dans la file soit les deux facteurs qu'il a trouvés, soit un seul si la décomposition s'arrête là (un facteur premier a été trouvé).
3. Le *parent* récupère les résultats (les facteurs premiers seulement) au fur et à mesure de leurs arrivées. Il multiplie ces facteurs entre eux pour savoir si la décomposition est terminée (le produit est égal au nombre de départ).
4. Le *parent* peut reinjecter des nouveaux nombres à décomposer ou signaler par un message spécial la fin aux *enfants*. Un message de fin par *enfant* est conseillé, pourquoi ?