

6 - SEMAPHORES : PROBLEMES CLASSIQUES

1. LE MODELE PRODUCTEUR-CONSOMMATEUR

Soit un tampon T constitué de n cases dont chacune est destinée à recevoir un message. On nomme Producteur le processus qui dépose l'information dans le tampon et Consommateur le processus qui retire l'information. Un tel type de communication est indirect et asynchrone.

- **Indirect** : car les producteurs et les consommateurs ne s'adressent pas l'un à l'autre.
- **Asynchrone** : car il n'est pas nécessaire que les producteurs et les consommateurs travaillent au même rythme.

1.1.

Quels sont les contraintes à respecter pour l'accès aux cases du tampon ?

On considère les trois tâches utilisateur suivantes :

```

main() {
    sprod = CS(v1);
    scon = CS(v2);
    CT(Production);
    CT(Consommation);
    ...
}

Production() {
    while (1) {
        Produire(message);
        P(sprod);
        Déposer(message);
        V(scon);
    }
}

Consommation() {
    while (1) {
        P(scon);
        Retirer(message);
        V(sprod);
        Consommer(message);
    }
}

```

La primitive CT() permet de créer une tâche. main() est le programme d'initialisation, Production() le code exécuté par un producteur et Consommation() le code exécuté par un consommateur. La primitive Produire() crée un message et la primitive Consommer() traite un message reçu.

1.2.

Expliquer le rôle des sémaphores sprod et scon. Quelles doivent être les valeurs initiales de v1 et v2 ?

On s'intéresse maintenant au contenu des procédures Déposer() et Retirer() des tâches Production() et Consommation(). Ces procédures manipulent le tampon avec les indices respectifs id et ir, initialisés à 0. Ces indices permettent une gestion circulaire du tampon.

```

Déposer(message)
MESS *message;
{
    (a) id = (id + 1) % n;
    (b) T[id] = message;
}

Retirer(message)
MESS *message;
{
    (c) ir = (ir + 1) % n;
    (d) message = T[ir];
}

```

1.3.

Montrer que deux producteurs (resp. consommateurs) peuvent produire (resp. consommer) dans la même case. Comment doit-on modifier les procédures Déposer() et Retirer() pour avoir une gestion correcte des cases du tampon ?

On désire assurer une plus grande indépendance de fonctionnement des producteurs et des consommateurs. Un producteur en train de produire un message ne doit pas empêcher d'autres producteurs d'acquiescer des cases vides pour produire. Pour cela il faut dissocier :

- l'acquisition d'une case vide, de son remplissage.
- l'acquisition d'une case pleine, de son vidage.

1.4.

Réécrivez simplement les procédures `Déposer()` et `Retirer()`. Montrez que dans ce cas, il faut rajouter des sémaphores pour garantir qu'un producteur ne produira pas dans une case en train d'être consommée.

2. LE PROBLÈME DES LECTEURS / ECRIVAINS

Soit un fichier pouvant être accédé en lecture ou en écriture. Pour garantir la cohérence des données de ce fichier, les processus qui souhaitent y accéder sont rangés dans deux classes, en fonction du type d'accès qu'ils demandent :

- les processus lecteurs ;
- les processus écrivains.

Le nombre de processus dans chaque classe n'est pas limité. On établit le protocole d'accès de la manière suivante :

- il ne peut y avoir simultanément un accès en lecture et un accès en écriture ;
- les écritures sont exclusives entre elles (au plus un accès en écriture à un instant donné).

Par contre, plusieurs lectures peuvent avoir lieu simultanément.

On considère dans un premier temps la politique d'accès suivante : si la ressource est disponible ou s'il y a déjà des requêtes de lecture en cours, une nouvelle requête de lecture est traitée immédiatement même s'il y a des demandes d'écriture en attente.

2.1.

Quelles sont les conditions de blocage pour un écrivain ? Pour un lecteur ? Donnez les sémaphores et les variables partagées nécessaires pour tester ces conditions, ainsi que leur initialisation.

2.2.

Programmez cette synchronisation de processus. Quel est l'inconvénient de cette stratégie d'accès ?

Pour assurer l'équité, on gère maintenant les accès dans l'ordre FIFO : lors de leur arrivée, tous les processus sont rangés dans une même file. On extrait de cette file soit un unique écrivain, soit le premier lecteur d'un groupe. Dans le deuxième cas, on autorise les lecteurs suivants dans la file à passer, jusqu'à ce que le processus en tête de file soit un écrivain.

2.3.

Par rapport à la question 3.1, quel sémaphore supplémentaire est nécessaire pour réaliser cette stratégie ? Programmez cette synchronisation de processus.

2.4.

Montrez que, dans la solution de la question précédente, une mauvaise utilisation des sémaphores (inversion de deux opérations P) peut conduire à un interblocage.