I- Synchronisation entre processus par sémaphore

1- Définition de la Synchronisation : Elle consiste à définir et réaliser des mécanismes assurant le respect des contraintes liées à la progression des processus, c'est dire capable de faire attendre un processus et/ou faire passer un processus bloqué.

On dit qu'un processus P est synchronisé avec un processus Q lorsque l'activité de P (resp. Q) dépend d'un événement modifié par Q (resp. P).

La synchronisation exige la mise en place d'un mécanisme permettant à un processus actif de bloquer un autre processus ou de se bloquer lui-même ou d'activer un autre processus. L'un des mécanismes utilisé pour la synchronisation est les *sémaphores*.

2- Expression de contraintes de synchronisation : elles sont reformulées sous deux formes : Point de Rendez-vous et Point de Synchronisation.

Point de Rendez-vous : on impose un ordre de précédence dans le temps logique sur l'exécution des actions (instructions) de processus.

Point de Synchronisation : on impose une condition pour le franchissement de certains points de leurs traces temporelles.

3- Spécification de la synchronisation : elle consiste à :

- définir pour chaque processus ses points de synchronisation/ *Rendez-vous*.
- associer à chaque points de synchronisation/ *Rendez-vous* une condition de franchissement exprimé au moyen des variables d'état du système (nombre d'unité de la ressource, état de la ressource…).

Exemple : considérons un système informatique ayant 3 Imprimantes. Chaque processus veut lancer une impression doit synchroniser avec l'autres processus demandant l'impression en même temps :

ALLOUER (Imprimante)
<Imprimer>
Librere (Imprimante)

Variables d'état du système = le nombre d'imprimante soit Nb_imp = 3; ainsi on a **ALLOUER** est un point de synchronisation bloquante (avec condition de franchissement). **Librere** est un point de synchronisation non bloquante (sans condition).

ALLOUER (Imprimante) :
si Nb_imp = 0 alors attendre
sinon Nb_imp = Nb_imp – 1;

Librere (Imprimante) : Nb_imp = Nb_imp + 1;

4- Relations d'ordre entre deux processus

Hypothèse: On considère un processus P0 dont l'évolution de son exécution est subordonnée à L'émission d'un signal par un processus P1.

```
Process P0 ;
debut
  A1; A2;…..;An;
   Attendre le signal de P1;
        ……;
fin
```

```
Process P1;
debut
 B1;B2;…..;Bm;
 Envoyer le signal d'activation à P0;
        ……;
Fin
```

Problème : Comment faire attendre le processus P1 par processus P0 ou comment le processus P1 envoi un signal d'activation à P0 ?

Solution : synchronisation par sémaphore

Nous définissons un sémaphore que nous appelons signal, initialisé à 0.

signal : sémaphore ; Init(signal, 0);

```
Process P1;
debut
 B1;B2;…..;Bm;
 V(signal);
        ……;
fin
```

```
Process P0 ;
debut
 A1; A2;…..;An;
 P(signal);
        ……;
fin
```

Dans cet exemple deux cas peuvent se présenter :

Cas 1: Le processus **P0** est déjà bloqué sur la primitive **P(signal)** lorsque le signal arrive. Quand le processus **P1** exécute la primitive **V(signal)**, il réveille le processus P0.

Cas 2: Le processus P0 est actif lorsque le signal est émis (il exécute l'instruction Ai). Tout se passe comme si le signal était mémorisé; en effet, la valeur du sémaphore signal est passé à 1 et lorsque le processus P0 exécutera la primitive P(signal), il ne se bloquera pas.

5- Les problèmes type : la plupart de problèmes de synchronisation rencontrés dans la pratique peuvent se remmener à un petit nombre de problème type dont la solution est connue : modèle producteurs/ consommateurs, modèle de lectures/rédacteurs, modèle de ressources banalisées, modèle de client/serveur, modèle des Philosophes….etc.

5.1- Problème des Producteurs /Consommateurs

Hypothèse: On considère deux catégories de processus : des Producteurs et des Consommateurs

- Il s'agit de producteurs qui produisent des objets (un objet est une valeur quelconque) et les déposent dans une mémoire commune appelée. Tampon.
- Les processus consommateurs utilisent les objets déposés dans le tampon.
- Le tampon étant de taille limitée N.

Contraintes de synchronisation:(Schéma de synchronisation) : Le fonctionnement de ces deux catégories de processus doit satisfaire les contraintes suivantes:

- Les producteurs ne déposent pas les objets lorsque le tampon est **plein**.
- Les consommateurs ne consomment pas du tampon lorsqu'il est **vide**.
- Le consommateur ne peut prélever un objet que le producteur est en train de le ranger.
- Si le producteur (resp. consommateur) est en attente parce que le tampon est plein (re sp. vide), il doit être averti dès que cette condition cesse d'être vraie.

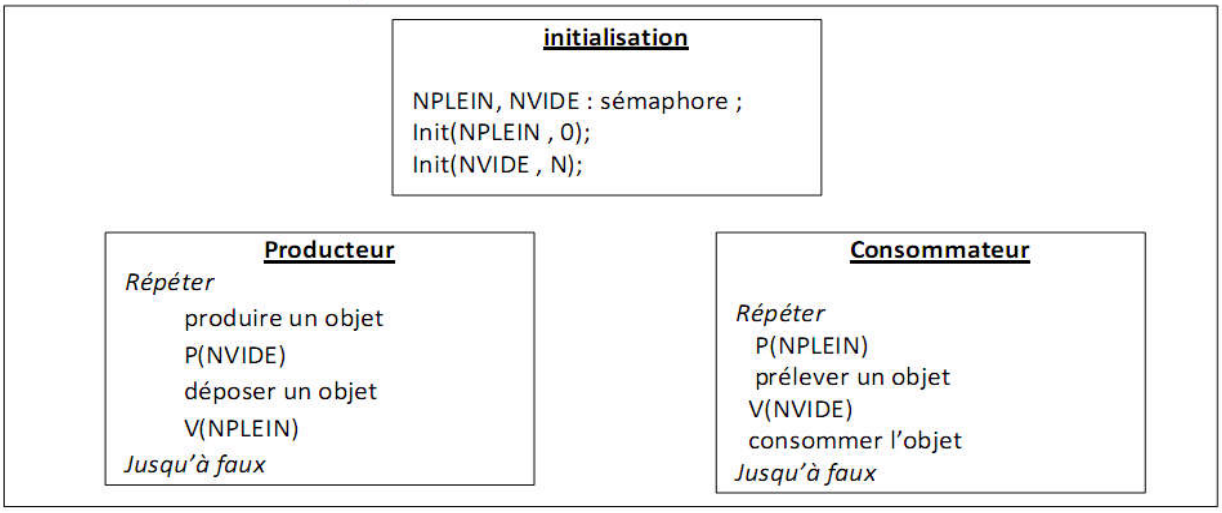
Solution:

On introduit donc deux variables caractérisant l'état du tampon :

NPLEIN : nombre d'objets dans le tampon (début : 0)

NVIDE : nombre d'emplacements disponibles dans le tampon (N au début).

Utilisation de trois sémaphores.



La consommation et la production se font à l'extérieur de la section critique afin de minimiser le temps passé dans la section critique.

5.2- Problème des Lecteurs /Rédacteurs

Hypothèse: On considère deux catégories de processus qui accèdent à une seule ressource commune (Fichier, Base de données).

- La première catégorie représente les Lecteurs: qui n'ont droit qu'à la lecture.
- La seconde dite Rédacteurs qui peuvent lire et mettre à jour (écrire) la ressource.

Contraintes de synchronisation:(Schéma de synchronisation)

- Eviter les accès simultanés des processus rédacteurs à la ressource i.e Les rédacteurs sont en exclusion mutuelle
- Les processus lecteurs peuvent accéder simultanément i.e Les Lecteurs ne sont pas en exclusion mutuelle
- Eviter l'accès simultanés d'un processus rédacteurs avec un ou plusieurs processus lecteurs.

Solution:

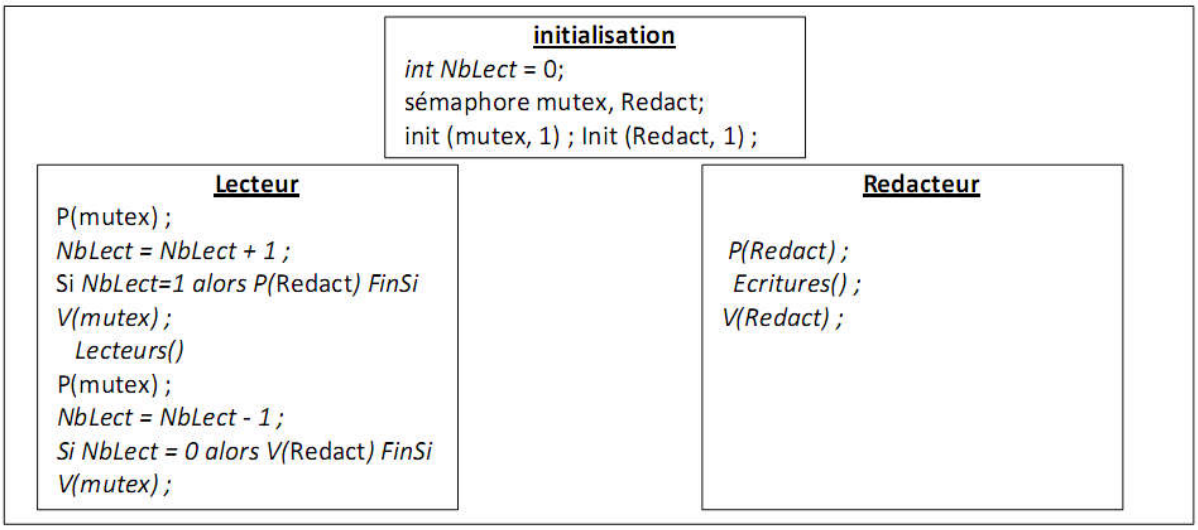
- Pour assurer l'accès exclusif (**écriture**) à **la base** par les rédacteurs, on utilise un sémaphore, appelons le : **Redact**
- Pour contrôler les accès à la base entre un redacteur et les lectures, on a besoin de connaître le nombre de lecteurs qui sont entrain de lire de la base (NbLect). En effet, ce compteur **NbLect** est un objet partagé par tous les lecteurs. L'accès à ce compteur doit être exclusif en utilisant un sémaphore mutex.

Note : Nblect ne peut pas être remplacé par la valeur d'un sémaphore car le nombre de processus lecteurs n'est pas limité.

Regles :

Un lecteur peut accéder à la base, s'il y a déjà un lecteur qui accède à la base (NbLect>0) ou aucun rédacteur n'est en train d'utiliser la base.

- Un rédacteur peut accéder à la base, si elle n'est pas utilisée par les autres.



Rmarque

- Redact qui garantit l'exclusion mutuelle entre les rédacteurs.
- Mutex sémaphore d'exclusion mutuelle entre les lecteurs qui protège la variable NB_lect.