Ecole Supérieur en Informatique de Sidi Bel Abbès

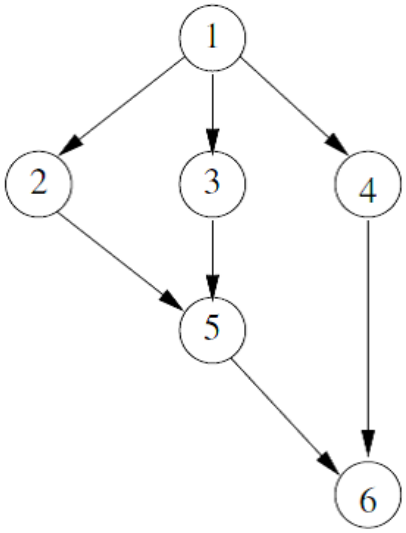
Module: **Système d'exploitation 2** — **1ère année Cycle Secondaire**
Semestre **: S2** — Année universitaire: **2020-2021**

Fiche de TD
Synchronisation des processus

Exercice 1:

1. Comment peut-on utiliser un sémaphore de sorte à ce qu'un processus se bloque si un événement n'a pas encore eu lieu, cet événement étant provoqué par un autre processus ?

2. Soit l'ensemble de processus suivant, où les contraintes de précédence sont données par le graphe.



a. Donner une solution utilisant des sémaphores et permettant de synchroniser ces processus de manière à respecter les contraintes de précédence.
b. Donner une solution de même type n'utilisant pas plus de 3 sémaphores

Exercice 2:

Deux villes A et B sont reliées par une seule voie de chemin de fer. Les règles de circulation sont les suivantes :

- La voie ne doit jamais être empruntée simultanément par deux trains allant en sens inverses
- La voie peut être empruntée par un ou plusieurs trains allant tous dans le même sens
- La priorité de parcours est la même pour les deux sens.

On considère deux classes de processus : les trains allant de A vers B : « T-AB » et les trains allant de B vers A : « T-BA ».

Processus T-AB	Processus T-BA
Début Entree_A(); <Circulation sur la voie de A vers B> Sortie_B(); Fin.	Début Entree_B(); <Circulation sur la voie de B vers A> Sortie_A(); Fin.

1. Quelle est la différence entre ce problème et le modèle des lecteurs/rédacteurs ?
2. En utilisant les sémaphores, écrire les codes des quatre procédures entree_A(), entree_B(), sortie_A() et sortie_B() de façon à ce que les processus respectent les règles de circulation sur la voie. Précisez clairement vos déclarations et initialisations.

Exercice 3:

On se basant sur le pseudo code ci-dessous, proposez une solution basée sur les sémaphores pour synchroniser l'affichage alterné à l'écran de deux processus *Afficheur* (P1, P2, P1, P2,…….). Il faut noter qu'un seul accès à l'écran est possible en même temps.

Processus P1

Tant que (vrai) faire:
//Demander l'accès à l'écran
…..
//Afficher
……
//Libérer l'accès pour l'autre

Processus P2

Tant que (vrai) faire:
//Demander l'accès à l'écran
…..
//Afficher
……
//Libérer l'accès pour l'autre

Exercice 04: Problème du Coiffeur dormeur

Soit un salon de coiffure dans lequel il y a un fauteuil pour la coiffure et *N* chaises pour les clients qui attendent. Quand il n'y a pas de clients, le coiffeur se met dans le fauteuil et pique un roupillon. Lorsqu'un client arrive et trouve le coiffeur endormi, il le réveille. Un client qui arrive et trouve le coiffeur occupé à coiffer s'assoit sur une des N chaises et attend son tour ; s'il n'y a plus de chaises libres il n'attend pas et s'en va.

En assimilant le coiffeur et les clients à des processus, donnez une solution à ce problème, qui utilise les sémaphores.

Corrigé

Exercice 1

1. On utilisant un sémaphore « sem » initialisé à 0. Le processus qui attend l'événement exécute P(sem) pour se bloquer. Et le processus qui provoque l'événement exécute V(sem), après l'événement pour débloquer le premier processus.
2. a.

Déclaration :
1vers2, 1vers3, 1vers4, 2vers5, 3vers5, 4vers6, 5vers6 : sémaphore ;
init (1vers2, 0); init (1vers3, 0); init (1vers4, 0); init (2vers5, 0); init (3vers5, 0); init (4vers6, 0); init (5vers6, 0);

```
Process P1{
Travail 1 ;
V(1vers2) ;
V(1vers3) ;
V(1vers4) ;
}
```

```
Process P2{
P(1vers2) ;
Travail 2
V(2vers5) ;
}
```

```
Process P3{
P(1vers3) ;
Travail 3 ;
V(3vers5) ;
}
```

```
Process P4{
P(1vers4) ;
Travail 4
V(4vers6) ;
}
```

```
Process P5{
P(2vers5) ;
P(3vers5) ;
Travail 5 ;
V(5vers6) ;
}
```

```
Process P6{
P(4vers6) ;
P(5vers6) ;
Travail 6;
}
```

b. Avec 3 sémaphores

s1, s2, s3 : sémaphore ;
init (s1,0), init (s2,0), init(s3,0);

```
Process P1{
Travail 1 ;
V(s1) ;
V(s1) ;
V(s1) ;
}
```

```
Process P2{
P(s1) ;
Travail 2
V(s2) ;
}
```

```
Process P3{
P(s1) ;
Travail 3 ;
V(s2) ;
}
```

```
Process P4{
P(s1) ;
Travail 4
V(s3) ;
}
```

```
Process P5{
P(s2) ;
P(s2) ;
Travail 5 ;
V(s3) ;
}
```

```
Process P6{
P(s3) ;
P(s3) ;
Travail 6;
}
```

Exercice 2:

1.

La différence : Dans le modèle lecteur/rédacteur, on ne peut trouver qu'un seul rédacteur à la fois entrain d'occuper la ressource partagée (fichier), alors que dans ce problème la ressource (la voie) peut être occupée par plusieurs processus en même temps et ceci pour les deux classes T-AB et T-BA.

2.

Déclarations et initialisations :

nbA : entier (init à 0), pour compter le nombre de trains allant de A vers B
nbB : entier (init à 0), pour compte le nombre de trains allant de B vers A.
Acces : Sémaphore (init à 1) pour que des trains de sens contraire ne s'engagent pas dans la voie en même temps.
Mutex1 : Sémaphore (init à 1) pour protéger la mise à jour en exclusion mutuelle de la variable critique nbA
Mutex2 : Sémaphore (init à 1) pour protéger la mise à jour en exclusion mutuelle de la variable critique nbB

Processus T-AB	**Processus T-BA**
Début P(Mutex1) ; nbA++ ; si (nbA==1) alors P(Acces) ; V(Mutex1) ; <Circulation sur la voie de A vers B> P(Mutex1) ; nbA - - ; si (nbA==0) alors V(Acces) ; V(Mutex1) ; Fin.	Début P(Mutex2) ; nbB++ ; si (nbB==1) alors P(Acces) ; V(Mutex2) ; <Circulation sur la voie de B vers A> P(Mutex2) ; nbB - - ; si (nbB==0) alors V(Acces) ; V(Mutex2) ; Fin.

Exercice 3:

On définit 2 sémaphores S1, S2 ;

Init(S1, 1); /* On choisit arbitrairement le 1er processus qui affiche */

Init(S2, 0);

Processus P1

Tant que (vrai) faire:
//Demander l'accès à l'écran
P(S1) ;
//Afficher
V(S2) ;
//Libérer l'accès pour l'autre

Processus P2

Tant que (vrai) faire:
//Demander l'accès à l'écran
P(S2) ;
//Afficher
V(S1) ;
//Libérer l'accès pour l'autre

Exercice 4

```
init(clients,0) ;
init(coiffeur,0) ;
init(mutex,1) ;
int NbClientsAttente=0 ;
Processus Coiffeur
Début
Tant que vrai faire
        P(clients) ;          /* dort si pas de clients */
        /* réveillé par un client, ou bien ne s'est pas rendormi après avoir coiffé un client car il y a
        au moins un client qui attend */
        P(mutex) ;
        NbClientsAttente-- ;        /* dit au client de s'installer sur le fauteuil */
        V(coiffeur) ;
        V(mutex) ;
        Travaille ;
        Fintantque
Fin.
```

```
init(clients,0) ;
init(coiffeur,0) ;
init(mutex,1) ;
int NbClientsAttente=0 ;
Processus Client i quelconque
Début
        P(mutex) ;
        Si (NbClientsAttente < N) alors
                NbClientsAttente++ ;
                V(clients) ;          /* avertit le coiffeur qu'il est là et s'assoit sur une chaise */
                V(mutex) ;            /* libère mutex avant de */
                P(coiffeur) ;         /* se mettre en attente que le coiffeur lui dise de s'installer sur
                        le fauteuil */
                Se faire couper les cheveux ;
        Sinon                         /* la salle est pleine
                V(mutex) ;            /* libère mutex avant d'aller voir un autre coiffeur */
                 Sortir();
Fin.
```