TD DAC : Les sémaphores

Exercice 1 :

Complétez, en ajoutant les sémaphores et les opérations P et V nécessaires, les codes du producteur et du consommateur suivants. Le producteur produit plusieurs ressources à la fois alors que le consommateur consomme une seule ressource à la fois.

```
char T[N]; // tableau de N caractères

Semaphore Plein =0, Vide=N, Mutex=1
```

```
Producteur {
int ip=0, M;
char ch[N];
Repeter
{
M=Lire(ch,N);
Pour i=1 à M Pas 1 faire

……

……

Deposer(ch, M, ip);

 ……

Pour i=0 à M-1 Pas 1 faire

……

ip = (ip + M) % N;
}
}
```

```
Consommateur {
int ic=0;
char c;
Repeter
{ ……

   ……

 c = Retirer( ic);

  ……

  ……

ic = (ic+1) %N
}
Traiter(c);
}
```

La fonction « int Lire(char ch[], int N); » construit, dans ch, une chaîne de caractères de longueur comprise entre 1 et N inclusivement. Elle retourne la longueur de la chaîne.

La fonction « void Deposer(char ch[], M, int ip); » insère, dans le tampon T, la chaîne de caractères ch. M est la longueur de la chaîne.

La fonction « char Retirer(int ic); » retire un caractère du tampon T. Elle retourne le caractère retiré.

La fonction « void Traiter(char c); » traite le caractère.

Exercice 2

On considère un ensemble de six tâches séquentielles {A, B, C, D, E ,F}.

- La tâche A doit précéder les tâches B, C, D. Les tâches B et C doivent précéder la tâche E.
- Les tâches D et E doivent précéder la tâche F.

-Représenter le graphe de précédence des tâches {A,B,C,D,E,F.}
-Réaliser la synchronisation de ces tâches en utilisant les sémaphores.

Exercice 3

On considère trois processus P1, P2 et P3. Le processus P1 produit des messages qu'il dépose dans un tampon T1. P2 prélève les messages contenus dans T1, les traites puis dépose les résultats dans un tampon T2. P3 prélève les messages contenus dans T2 et les consomme.

1) Ecrire les algorithmes de P1, P2 et P3 de façon à garantir le non-interblocage.

2) On considère maintenant que les Pi (i=1..3) travaillent sur le même tampon T (au lieu de T1 et T2).

Réétudier la question 1).

Exercice 4 : Le coiffeur

Dans un salon de coiffure, il y a un coiffeur C, un fauteuil F dans lequel se met le client pour être coiffé et N sièges pour attendre.

Les événements possibles sont les suivants :

- S'il n'a pas de clients, le coiffeur C somnole dans le fauteuil F,

- Quand un client arrive et que le coiffeur C dort, le coiffeur se réveille et se lève. Le client s'assied alors dans le fauteuil F et se fait coiffer,

- Si un client arrive pendant que le coiffeur travaille : si un des N sièges est libre, il s'assied et attend, sinon il ressort.

1. Expliquer les scénarios du coiffeur et celui du client.

2. Proposez une solution qui assure l'exclusion mutuelle en utilisant les sémaphores suivants :

- **SP** : gère la file des clients en attente. Indique le nombre de ressources pour le coiffeur, c'est-à-dire les clients.

- **SCF** : gère l'accès au fauteuil du coiffeur. Indique si le coiffeur est prêt à travailler ou non (il est la ressource unique des clients).

- **SX** : assure l'accès en exclusion mutuelle à la variable partagée Attend. Attend permet de contrôler la taille maximale de la file d'attente.

Exercice 5

N processus ,non cycliques, travaillent par point de rendez-vous :

```
Proc P1   . . .   Proc Pi   . . .   Proc PN
Début             Début             Début
  .                 .                 .
  .                 pt rdv            .
  .                 .                 .
  .                 .                 pt rdv
  .                 .                 .
 pt rdv             .                 .
  .                 .                 .
Fin               Fin               Fin
```

- Programmer le rendez-vous de ces processus en utilisant les sémaphores.

Exercice 6

Le carrefour ou le problème de la gestion des feux de circulation.

La circulation au carrefour de deux voies est réglée par des signaux lumineux (feu vert/rouge).

On suppose que les voitures traversent le carrefour en ligne droite et que le carrefour peut contenir au plus une voiture à la fois.

On impose les conditions suivantes :

1. toute voiture se présentant au carrefour le franchit en un temps fini,

2. les feux de chaque voie passent alternativement du vert au rouge, chaque couleur étant maintenue pendant un temps fini.

Les arrivées sur les deux voies sont réparties de façon quelconque. Le fonctionnement de ce système peut être modélisé par un ensemble de processus parallèles :

1. un processus P qui exécute la procédure **Changement** de commande des feux;

2. un processus est associé à chaque voiture;

3. la traversée du carrefour par une voiture qui circule sur la voie i (i = 1, 2) correspond à l'exécution d'une procédure **Traverseei** par le processus associé à cette voiture.

On demande d'´ecrire le programme **Changement** ainsi que les procédures **Traversee1** et **Traversee2**.

Remarque : Le processus P doit attendre que la voiture engagée dans le carrefour en soit sortie avant d'ouvrir le passage sur l'autre voie.

Changement
{
int Feu = 1
while (1)
{ sleep (Duree_du_feu) ;
if (Feu == 1)
{ ……..
……..
……..
}
else
……..
……..
……..
} }}
	Traversee1
	{
	……..
	……..
	Circuler() ;
	……..
	……..
	}
		Traversee2
		{
		……..
		……..
		Circuler() ;
		……..
		……..

Exercice 7 :

Problème des philosophes

Soient 5 philosophes sont assis à une table circulaire, ils aimeraient manger leur plat de spaghettis avec 2 fourchettes chacun mais il n'y a que N fourchettes. Chaque philosophe, pense, puis prend ses fourchettes de gauche et droite puis mange, une fois qu'il termine de manger il passe à l'état « penser », etc. Evidemment ils ne pourront pas tous manger en même temps. Il s'agit d'écrire un algorithme qui permet de les synchroniser en évitant l**'interblocage.**