esprit

Module	SEA
Enseignant	ANIS ZOUAOUI
Classes	4INFO

TD SEMAPHORE

Exercice 1 : Rendez-vous à N processus

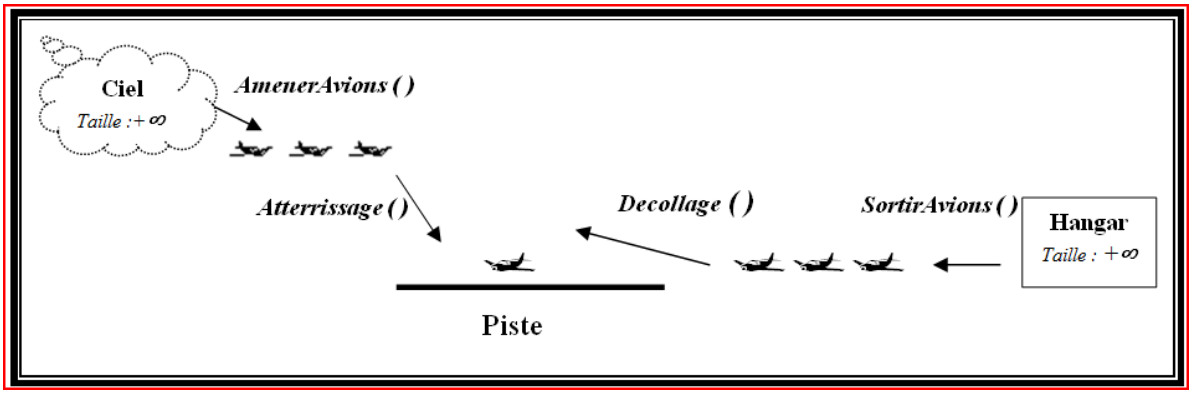
Soient N processus parallèles ayant un point de rendez-vous. Un processus arrivant au point de rendez-vous se met en attente s'il existe au moins un autre processus qui n'y est pas arrivé. Le dernier arrivé réveillera les processus bloqués.
Proposer une solution qui permet de résoudre ce problème en utilisant des sémaphores.

Exercice 2 : Gestion du trafic aérien

Le but de cet exercice est la gestion du trafic aérien. On ne dispose que d'une seule piste à la fois d'atterrissage et de décollage. En plus, cette piste ne peut accepter qu'un seul avion quelque soit la manœuvre (atterrissage ou décollage). Pour cela, on dispose de deux files d'attente :
- en l'**air** de taille **N** pour les avions souhaitant atterrir
- et au **sol** de taille **M** pour les avions souhaitant décoller.

La gestion de ce trafic d'avions nécessite, alors, quatre fonctions :

- une fonction ***SortirAvions ( )*** qui fait sortir les avions du hangar (dépôt) et les place dans la file d'attente de décollage,
- une fonction ***Decollage ( )*** qui prend un avion cloué en sol dans la file d'attente de décollage et le fait décoller en utilisant la piste,
- une fonction ***AmenerAvions ( )*** qui fait entrer, dans la file d'attente d'atterrissage, des avions en vol, et
- une fonction ***Atterrissage ( )*** qui prend un avion de la file d'attente d'atterrissage et le fait atterrir en utilisant la piste.



- Préciser le rôle de chacun des sémaphores utilisés (**voir ci-après**).
- Détailler le code de chaque fonction donnée ci-dessous.

```
#define N 5
#define M 5
semaphore  Decollage_vide, Decollage_plein, Atterrissage_vide, Atterrissage_plein, Piste;
Sem_Init (Decollage_vide, M); //initialisation du sémaphore Decollage_vide par la valeur M
Sem_Init (Decollage_plein, 0);
Sem_Init (Atterrissage_vide, N);
Sem_Init (Atterrissage_plein, 0);
Sem_Init (Piste, 1);
```

void ***SortirAvions ( )***
{
……
……
Ajouter_un_avion_dans_la_zone_attente_decollage( );
……
……
}

void ***Decollage ( )***
{
……
……
Faire_decoller_un_avion ( );
……
……
}
void ***AmenerAvions ( )***
{
……
……
Ajouter_un_avion_dans_la_zone_attente_atterrissage( );
……
……
}
void ***Atterrissage ( )***
{
……
……
Faire_Atterrir_un_avion ( );

……..
……..
}

Remarques :

- Les fonctions : Faire_Atterrir_un_avion() et Ajouter_un_avion_dans_la_zone_attente_atterrissage() sont mutuellement exclusives.
- Les fonctions : Faire_decoller_un_avion() et Ajouter_un_avion_dans_la_zone_attente_decollage() sont mutuellement exclusives

Exercice 3 : Unité de fabrication des stylos

Synchronisez au moyen de sémaphores l'enchaînement des opérations de fabrication de stylos à bille. Chaque stylo est formé d'un corps, d'une cartouche, d'un bouchon arrière et d'un capuchon. Les opérations à effectuer sont les suivantes :

- remplissage de la cartouche avec l'encre (opération RC),
- assemblage du bouchon arrière et du corps (opération BO),
- assemblage de la cartouche avec le corps et le capuchon (opération AS),
- emballage (opération EM).

Chaque opération est effectuée par une machine spécialisée (**mRC**, **mBO**, **mAS**, **mEM**).

Les stocks de pièces détachées et d'encre sont supposés disponibles quand la machine est disponible. Les opérations **RC** et **BO** se font en parallèle. L'opération **AS** doit être effectuée, après ces deux opérations, en prélevant directement les éléments sur les machines **mRC** et **mBO**. Le produit assemblé est déposé dans un stock en attente de l'opération **EM**. L'opération **EM** se fait donc après **AS**, à partir du stock. Le stock est supposé de taille N et de discipline FIFO.

mRC( )
{ while (1)
{

RC( ) ;
}
}

mAS( )
{ while (1)
{
AS( ) ;
}
}

mEM( )

```
{ while (1)
{
EM() ;
}
}
```

Exercice 4 :

Deux villes A et B sont reliées par une seule voie de chemin de fer. Les trains peuvent circuler dans le même sens de A vers B ou de B vers A. Mais, ils ne peuvent pas circuler simultanément dans les sens opposés. On considère deux classes de processus: le nombre N de trains allant de A vers B (Train AversB) et le nombre M de trains allant de B vers A (Train BversA).

Proposez une synchronisation des deux classes de processus Train_BversA() et Train_AversB().