

TP1 – Système temps réel

Question 1

Deux sémaphores sont nécessaires.
Sémaphores case_vides et cases_pleines
Les valeurs initiales sont :
- case_vides $\leftarrow$ N (taille du Buffer)
- case_pleines $\leftarrow$ 0

Question 2

P() -> sem_wait() → demande autorisation
V() -> sem_post() → fin d'utilisation

Thread WriteBuffer()	Thread WriteBuffer()
<pre>{ P(case_vides) Buffer[] = text ; V(case_pleines) }</pre>	<pre>{ P(case_vides) Buffer[] = text ; V(case_pleines) }</pre>

Question 3

(Voir code)
On remarque que l'écriture se fait plus vite que la lecture, des que le sémaphore laisse le thread passer il part immédiatement, il faut gérer cela avec des sleep

Question 4

Buffer Air[]
Entrée : amenerAvion()
Sortie : atterrissage

Buffer Sol[]
Entrée : sortirAvion()
Sortie : décollage()

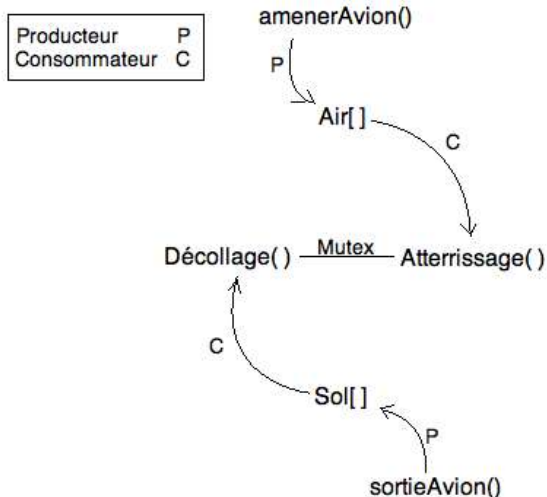
Problèmes "producteurs/consommateurs" :

On ne peut pas amener un avion et en faire atterrir un en même temps car les deux fonctions demandent l'accès à la même ressource.
De même, on ne peut pas faire sortir un avion et en faire décoller un.

Problèmes "exclusion mutuelle" :

Etant donné qu'il n'y a qu'une piste, on ne peut pas faire atterrir et décoller un avion au même moment.
-> Nombre de sémaphore : 4
-> Nombre de mutex : 1

Question 5



Atterrissage :

```

lock(mutex);
    P(case_pleine_air);
    {
        Lecture;
    }
    V(case_vide_air);
unlock(mutex);

```

Décollage :

```

lock(mutex);
    P(case_pleine_sol);
    {
        Lecture;
    }
    V(case_vide_sol);
unlock(mutex);

```

AmenerAvion :

```

P(&case_vide_air);
{
    Ecriture;
}
V(&case_pleine_air);

```

SortirAvion :

```

P(&case_vide_sol);
{
    Ecriture;
}
V(&case_pleine_sol);

```

Question 6

L'ordre des sémaphores est important dans le sens où les instructions P() et V() (dans cet ordre) doivent encadrer le code critique pour éviter les conflits lors de la demande de ressources. Si ces conditions ne sont pas respectées, les threads peuvent essayer d'avoir accès au même buffer. Il en résulterait un deadlock.

Question 7

Si le nombre d'avion géré par le système est infini, il faut s'assurer que le temps mis pour amener un avion est plus long que celui pour le faire atterrir. De même, le temps pour sortir un avion doit être plus long que celui pour le faire décoller. Ces réglages se font par le biais de 'sleep()' que l'on implémentera dans chacune des fonctions. Si ces conditions ne sont pas respectées, le système pourrait bloquer les buffers en laissant la main trop longtemps à l'un des threads producteurs.

Question 8 (Voir code)