TD 1

Gestion d'un parking de stationnement de véhicules

Pour illustrer concrètement les systèmes répartis, prenons l'exemple d'un parking de ***N*** places (ressources) destinées à accueillir des automobiles (processus) en compétition pour se garer. Chaque processus peut exécuter les deux actions (événements) suivantes :

E : entrer dans le parking (garer).
S : sortir du parking (quitter).

Il est clair qu'une synchronisation est nécessaire : une voiture ne peut se garer que s'il existe une place, si ce n'est pas le cas elle doit attendre qu'une autre voiture sorte.

Les séquences d'événements peuvent être légales ou non légales. Par exemple si ***N* = 3** :

EESEES est une séquence légale : il y a toujours une place pour une voiture.
EEEES est une séquence illégale : on ne peut garer 4 voitures de suite sans sortie.

Il est donc nécessaire de contrôler les entrées/sorties du parking pour que le système puisse fonctionner.

Une expression abstraite de synchronisation du système est : Le nombre de places libres ***NBPLib*** dans le parking doit être toujours tel que :

$$0 \leq NBPLib \leq N \quad (1)$$

Par exemple on peut imaginer un compteur ***E*** qui enregistre les entrées et un compteur ***S*** qui enregistre les sorties, alors :

$$NBPLib = N - E + S \quad (2)$$

De (1) et (2), les points de synchronisation (les l'entrées e et les sorties) du système doivent donc vérifier à chaque événement (d'entrer et/ou de sortie) l'expression suivante :

$$E - S < N \quad (3)$$ // Expression abstraite du problème

Les points de synchronisation (les l'entrées e et les sorties) du système doivent aussi effectuer respectivement les actions suivantes :

A l'entrée :

```
if (E - S) < N then
    E := E + 1
```

A la sortie :

```
S := S + 1
```

On remarquera que les opérations « A l'entrée » et « A la sortie » sont indépendantes d'où la possibilité de les exécuter de manière simultanée. Autrement dit, les accès en entrées et les accès en sorties (du parking) des véhicules peuvent se faire en parallèle.

D'un autre côté, si on s'intéresse au nombre de places libres dans le parking représenté par l'expression (2) les actions du contrôle deviendraient :

A l'entrée :

if NBPLib > 0 then

***NBPLib* := *NBPLib* - 1**

A la sortie :

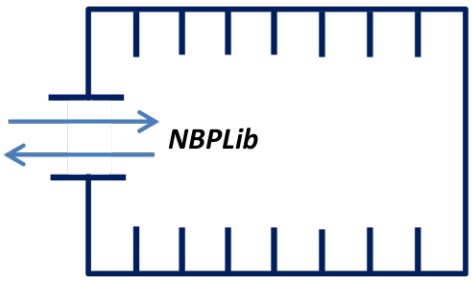
***NBPLib* := *NBPLib* + 1**

Les accès en entrée et en sortie doivent donc être sérialisées et ne peuvent plus se réaliser simultanément, c'est à dire que l'exclusion mutuelle doit porter sur l'ensemble des opérations effectuées aux entrées et sorties.

1. Mise en œuvre dans un environnement centralisé :

```
Parking.monitor
  var  NBPLib : integer;
       attendre : condition;
  NBPLib := N;
  procedure entrer
    Begin
       if  (NBPLib = 0) then
           wait (attendre);
       end if
       NBPLib := NBPLib - 1;
    end procedure

  procedure sortir
    Begin
       NBPLib := NBPLib + 1;
       signal (attendre);
    end procedure
end monitor
```



L'exclusion mutuelle entre les procédures entrer et sortir impliquent que les véhicules ne peuvent entrer et sortir simultanément.

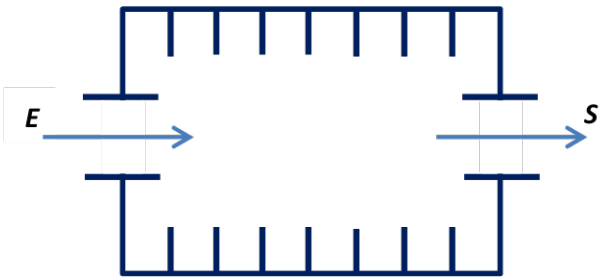
2. Mise en œuvre dans un environnement distribué :

```
Parking.monitor
  var  E, S : integer;
       attendre : condition;
  E := S := 0;
  procedure entrer
    Begin
       if  (E - S ≥ N) then
           wait (attendre);
       end if
       E := E + 1;
    end procedure
  procedure sortir
    Begin
       S := S + 1;
       signal (attendre);
    end procedure
end monitor
```

Dans ce cas, il faut répartir le contrôle sur *M* sites, ce qui revient à considérer *M* processus contrôleurs, un sur chaque site. Deux types de solutions sont alors possible :

a) Répartition des variables :

Considérons un parking à deux accès : un accès en entrée (site 1) et l'autre en sortie (site 2). Le processus 1 contrôleur de l'accès du site 1 comptabilise les entrées *E*, et le processus 2 contrôleur de l'accès du site 2 comptabilise les sorties.



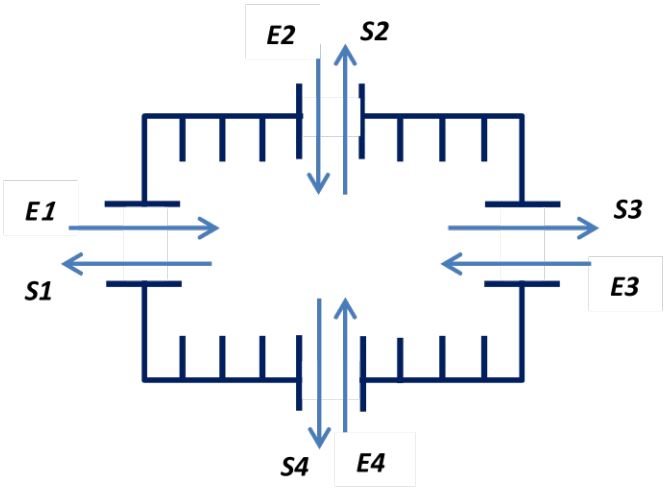
Pour autoriser un véhicule à entrer dans le parking, le processus 1 doit d'abord vérifier l'expression : ***E - S < N*** (3). Il connaît exactement la valeur de *E*, mais la valeur de *S* qui se trouve sur le site de sortie n'est connue que par le processus 2. Il peut alors la lui transmettre pour permettre l'évaluation de l'expression. Compte tenu des délais (non nuls) de transmission, le processus 1 recevra une valeur de *S* en retard par rapport à la valeur courante. Cette valeur *S'* est une image retardée de *S*. Le gardien 1 ne peut donc évaluer que l'expression :

$$E - S' < N \quad (4)$$

Or, on a : $S' < S$, on peut donc affirmer que si (4) est vérifiée, (3) l'est aussi.

b) Décomposition des variables :

On suppose maintenant que le parking comporte ***M*** accès fonctionnant tous en entrée/sortie.



Les ***Ei*** et ***Si*** représentent le nombre d'entrée et de sortie de véhicules enregistrées sur le site ***i*** par le processus N° ***i***. On a alors :

$$E = E1 + E2 + \ldots\ldots + EM$$
$$S = S1 + S2 + \ldots\ldots + SM$$

Chaque processus ***i*** ne connaissant que deux informations ***Ei*** et ***Si***, il a besoin, pour vérifier l'expression de synchronisation, de se faire envoyer par les autres processus les informations restantes. L'expression de synchronisation pour un processus ***i*** devienne donc :

$$(E1'+E2'+\ldots\ldots+Ei+\ldots\ldots+EM') - (S1'+S2'+\ldots\ldots+Si+\ldots\ldots+SM') < N \quad (5)$$

Or, on a : $\forall j \neq i / (Ej' < Ej) \wedge (Sj' < Sj)$, mais on ne peut pas affirmer que si (5) est vérifiée, (3) l'est aussi !. À cause d'absence d'état global cohérent de ce système répartie.