TD N°05.02 : Corrigé
Synchronisation des processus par sémaphore

Sujet : Lecture, traitement et impression en parallèle

- Un programme de calcul scientifique est composé de trois parties:
  - Lecture d'un enregistrement de données,
  - Traitement de ces données
  - et Impression des résultats.

Ceci est inclus dans une boucle infinie.

```
loop
        Lecture
        Traitement
        Impression
end loop
```

Sujet : Lecture, traitement et impression en parallèle

On peut accélérer l'exécution en séparant ce programme en trois processus parallèles. On peut ainsi récupérer les temps perdus en entrée et sortie (lors des phases Lecture et Impression).

```
Process P1
  loop
    Lecture
  end loop
end P1
```

```
process P2
  loop
    Traitement
  end loop
end P2
```

```
process P3
  loop
      Impression
  end loop
end P3
```

- On utilise une ***zone mémoire partagée*** pour transmettre les données de P1 à P2, et de P2 à P3.
  - C'est-à-dire que P1 range à la suite dans une zone mémoire ces données et que P2 les retire au fur et à mesure de ces besoins.
  - Le rangement et le prélèvement des informations dans ces zones ***font partie de Lecture***,
  - Traitement et Impression.
  - On ne s'occupera donc pas de leur écriture. On s'occupera seulement de synchroniser les processus.

- Naturellement, ces processus doivent être synchronisés, car on ne doit exécuter **Traitement** que si les **données sont prêtes (à la fin de Lecture),** et **Impression** ne doit être exécuter que si les **résultats sont prêts (à la fin de Traitement).**

Sujet : Lecture, traitement et impression en parallèle

On vous demande de synchroniser ces processus pour les deux cas suivants en utilisant des sémaphores. Pour les sémaphores, on possède un type et les deux primitives **wait( )** et **signal( ).**

1er cas:

On dispose de zones de mémoire infinie. Ainsi la Lecture peut être exécutée de façon répétitive sans être arrêtée, et le Traitement n'a pas besoin d'attendre après l'Impression.

1er Cas

```
Process P1
      loop

            Lecture

      end loop
end P1
```

```
process P2
      loop

            Traitement

      end loop
end P2
```

```
process P3
      loop

                  Impression

      end loop
end P3
```

1er cas : réponse

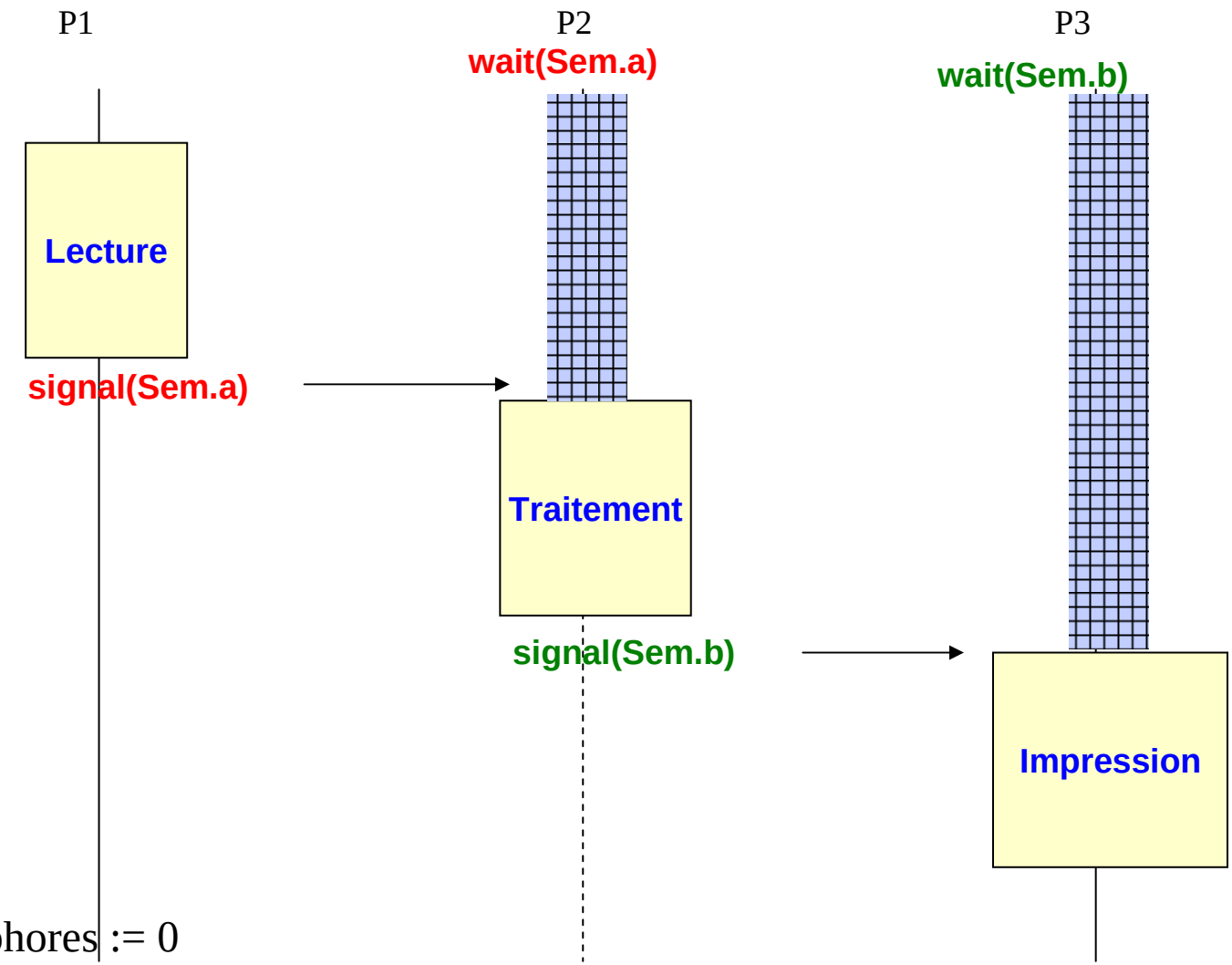
```
Process P1
   loop

      Lecture
       signal(Sem.a)
   end loop
end P1
```

```
process P2
   loop
       wait(Sem.a)
      Traitement
       signal(Sem.b)
   end loop
end P2
```

```
process P3
   loop
      wait(Sem.b)
      Impression

   end loop
end P3
```

var Sem.a : semaphores := 0
var Sem.b : semaphores := 0

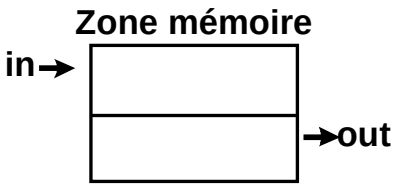
1er cas : réponse

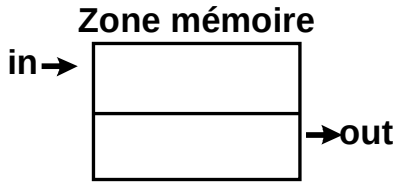


var Sem.a : semaphores := 0
var Sem.b : semaphores := 0

2er cas :

- Les zones mémoires pour le transfert de P1 vers P2 et de P2 vers P3 ne peuvent contenir que **deux informations** à la fois.
  - Par exemple, la lecture de information **#3** ne peut commencer que si le Traitement de l'information **#1** est terminée.
  - De même, le Traitement de l'information **#3** (produisant le résultat **#3**) ne peut commencer que si l'Impression du résultat **#1** est terminé.






```
Process P1
      loop
                Lecture
      end loop
end P1
```

```
process P2
      loop
                Traitement
      end loop
end P2
```

```
process P3
      loop
                Impression
      end loop
end P3
```

2er cas :

```
Process P1
   loop
      <<sync>>

      Lecture
      <<sync>>

   end loop
end P1
```

```
process P2
   loop
         <<sync>>
         <<sync>>
         Traitement
         <<sync>>
         <<sync>>
   end loop
end P2
```

```
process P3
   loop

      <<sync>>
      Impression

      <<sync>>
   end loop
end P3
```

2er cas : trame

```
Process P1
   loop

      Lecture


   end loop
end P1
```

```
process P2
   loop

         Traitement


   end loop
end P2
```

```
process P3
   loop

      Impression


   end loop
end P3
```

2er cas : trame

var a : semaphores :=
var b : semaphores :=
var c : semaphores :=
var d : semaphores :=

Process **P1** loop **Lecture** end loop end **P1**	process **P2** loop **Traitement** end loop end **P2**	process **P3** loop **Impression** end loop end **P3**

2er cas : solution

var Sem.a : semaphores := **0**
var Sem.b : semaphores := **0**
var Sem.c : semaphores := **2**
var Sem.d : semaphores := **2**

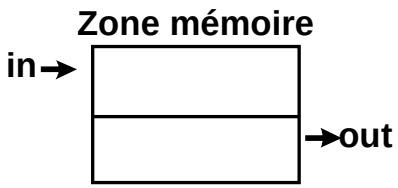
```
Process P1
   loop
      wait(Sem.c)

      Lecture
      signal(Sem.a)

   end loop
end P1
```

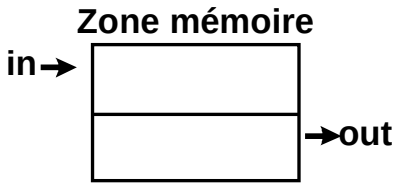
```
process P2
   loop
         wait(Sem.d)
         wait(Sem.a)
         Traitement
          signal(Sem.b)
          signal(Sem.c)
   end loop
end P2
```

```
process P3
   loop

      wait(Sem.b)
      Impression

      signal(Sem.d)
   end loop
end P3
```

Sem.c = **2 : nbr de cases libres**

Sem.d = **2 nbr de cases libres**





2er cas : solution