

Les Sémaphores

- [Définition](#)
 - [Sémaphores d'Exclusion Mutuelle](#)
 - [Sémaphores de Synchronisation](#)
 - [Autres utilisations des sémaphores - le Rendez-vous](#)
 - [Interblocages](#)
-

Définition

Introduits par **Dijkstra** en 1965.

Les sémaphores sont un outil **élémentaire** de synchronisation qui évitent l'attente active.

Un sémaphore s =

- un entier $e(s)$;
- une file d'attente $f(s)$;
- deux primitives $P(s)$ et $V(s)$.

Soit p le processus qui effectue $P(s)$ ou $V(s)$.

| P(s) | V(s) |
|--|--|
| $e(s) = e(s) - 1;$ si $e(s) < 0$ alors . état(p) = bloqué ; . entrer($p, f(s)$); | $e(s) = e(s) + 1;$ si $e(s) <= 0$ alors . sortir($q, f(s)$); . état(q) = éligible ; . entrer($q, f(\text{éligibles})$); |

Sémaphores d'Exclusion Mutuelle

But : protéger l'accès à une ressource **unique** (e.g. variable, imprimante, ...).

$e(s)$ est initialisée à **1**.

Utilisation :

$P(s)$
< Section Critique >
 $V(s)$

Tous les processus doivent suivre la même règle.

Sémaphores de Synchronisation

But : un processus doit en attendre un autre pour continuer (ou commencer) son exécution.

$e(s)$ est initialisée à **0**.

Utilisation :

| Processus 1 | Processus 2 |
|---|---|
| 1er travail $V(s)$ // réveil processus 2 | $P(s)$ // attente processus 1 2ème travail |

Autres utilisations des sémaphores

Le **rendez-vous** (généralisation du problème précédent).

Un processus doit attendre que n autres processus soient parvenus à un endroit précis pour poursuivre son exécution.

On utilise :

- un sémaphore S_{sync} initialisé à **0** ;
- un sémaphore $mutex$ initialisé à **1** ;
- un sémaphore S_{attend} initialisé à **0** ;
- un entier nb initialisé à **0**.

| Processus i | Processus RdV |
|--|---------------------|
| ... | ... |
| $P(mutex);$ | ... |
| $nb = nb + 1;$ | $P(S_{sync});$ |
| si $nb = n$ alors $V(S_{sync}); nb = 0;$ | $V(S_{attend}, n);$ |
| $V(mutex);$ | ... |
| $P(S_{attend});$ | ... |
| ... | ... |

Interblocage

SemA et SemB sont deux sémaphores d'exclusion mutuelle.

| Processus i | Processus j |
|---------------|---------------|
| ... | ... |
| $P(semA);$ | $P(semB);$ |
| $P(semB);$ | $P(semA);$ |
| < SC > | < SC > |
| $V(semB);$ | $V(semA);$ |
| $V(semA);$ | $V(semB);$ |
| ... | ... |

Il existe deux remèdes différents pour s'affranchir des interblocages.

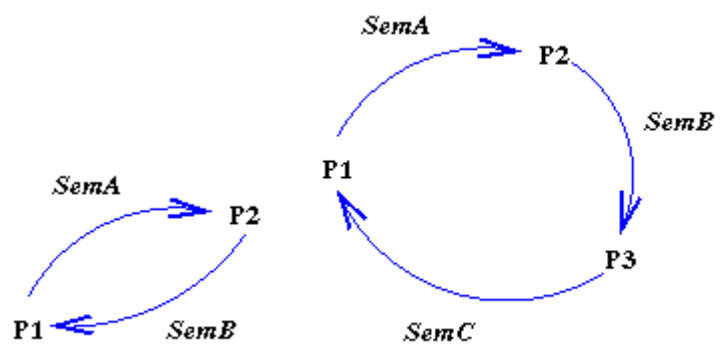
Prévention :

- exécuter les $P()$ toujours dans le *même ordre* ;
- utiliser un algorithme tel que l'algorithme du **banquier** et déclarer quelles sont les ressources que l'on va utiliser.

Détection--Guérison :

1. construire le graphe des conflits (périodiquement) ;

- 2. si un circuit \Rightarrow interblocage ;
- 3. tuer un processus \Rightarrow effectuer les V() manquants.



[Retour au sommaire.](#)