

# Petit résumé du sémaphore

**Important :** les explications peuvent être inexactes et mêmes fausses, je ne suis responsable de rien, si vous trouvez des fautes, ou si vous avez d'autres choses à ajouter veuillez me contacter s'il vous plaît.

## -Le sémaphore :

### Définition :

-Un sémaphore est une structure de données qui permet de gérer l'accès concurrent à une ressource partagée (imprimante par exemple) par plusieurs processus (ou threads), c'est un moyen de coordination entre ces derniers.

-Il est composé de :

1-une variable, appelé habituellement S ou sem , qui contient le nombre de threads qui peuvent encore accéder à la ressource.

->Cette variable est décrémenté à chaque fois qu'un processus accède à la ressource, et est incrémentée si une ressource est de nouveau disponible (le processus a terminé de l'utiliser).

-> Si cette variable est positive donc il y a encore de place pour accéder à la ressource, s'il est nul, n'y a plus de place pour y accéder, sinon, sa valeur absolue correspond au nombre de processus en attente.

->Cette variable est initialisé au début avec le nombre maximal des threads qui peuvent accéder simultanément à une ressource, cela se fait une seule fois à travers une fonction appelé Init , plus d'information [ICI](#)

2-une liste, qui contient les threads qui sont en attente.

### Manipulation :

-Cette variable est manipulé à travers ces trois fonctions :

-P : du néerlandais *Proberen* qui signifie tester, appelé aussi DOWN (peut-être car elle décrémente la valeur de S), cette fonction est appelée par le processus qui veut utiliser la ressource partagée, si cette dernière est disponible (i.e.  $S \geq 0$ , après l'avoir décrémenter) le processus aura le feu vert pour accéder à la ressource, sinon il sera bloqué, ajouté à la liste d'attente et endormit.

---

```
Down
{
  S--;
  Si  $s < 0$ 
    ajouter le processus à la liste d'attente
    endormir le processus
  fin si
}
```

---

-**V**: du néerlandais *Verhogen* qui signifie incrémenter , appelé aussi UP (elle incrémente la valeur de S), cette fonction est appelée par le processus après qu'il finit d'utiliser la ressource partagée, elle incrémente la valeur de S , si cette valeur est négatif ou nulle après l'incrémementation (donc la liste d'attente n'est pas vide), le premier processus dans la liste d'attente (qui est une liste FIFO) est réveillé.

---

```
Up
{
  S++;
  si  $s \leq 0$ 
    sortir le premier élément de la liste d'attente (fifo)
    le reveiller
  fin si
}
```

---

-**Init** : comme on l'a déjà dit, elle permet d'initialiser la variable S, elle n'est pas discutée dans ce document.

### **Un exemple vaut mieux qu'un long discours :**

S prendra la valeur 4 si la ressource partagée est un ensemble d'imprimante, et on en a 4.

Si 4 processus sont en train d'utiliser les imprimantes ( $S=0$ ), et un 5ème processus tente d'utiliser l'une d'eux en appelant la fonction P, cette dernière va décrémenter S à -1 et donc le processus sera endormi et ajouté à la liste d'attente.

Une fois qu'un des processus termine d'utiliser la ressource, il appellera la fonction V qui va incrémenter S, réveiller le processus endormi et lui donner l'accès à la ressource.

### **EXO :**

Lecteurs/rédacteurs :

Lecteurs :

-plusieurs lecteur peuvent accéder à la mémoire simultanément.

-En exclusion mutuelle avec les rédacteurs.

Rédacteurs :

En exclusion mutuelle avec les lecteurs et les autres rédacteurs.

### **Solution :**

```
lectred_mut<-1 // mutex lecteurs - rédacteurs  
lect_mut <-1 // mutex lecteurs pour accéder à la variable lect  
lect <- 0 // le nombre de lecteur qui accède à la ressource
```

#### **Rédacteur :**

```
P(lectred_mut)  
**écriture**  
V(lectred_mut)
```

#### **Fin Rédacteur**

#### **Lecteur :**

```
static n =0  
P(lect_mut)  
n=n+1  
if (n==1)  
    P(lectred_mut)  
V(lect_mut)  
**lecture**  
P(lect_mut)  
n=n-1  
if (n==0)  
    V(lectred_mut)  
V(lect_mut)
```

#### **Fin Lecteur**

---

### **Sémaphore en C :**

il faut 'include' la bibliothèque <semaphore.h>

on déclare la sémaphore : sem\_t nom ;

on l'initialise : sem\_init(&nom,0,valeur\_sem)

la fonction P = sem\_wait(&sem)

la fonction `V = sem_post(&sem) ;`