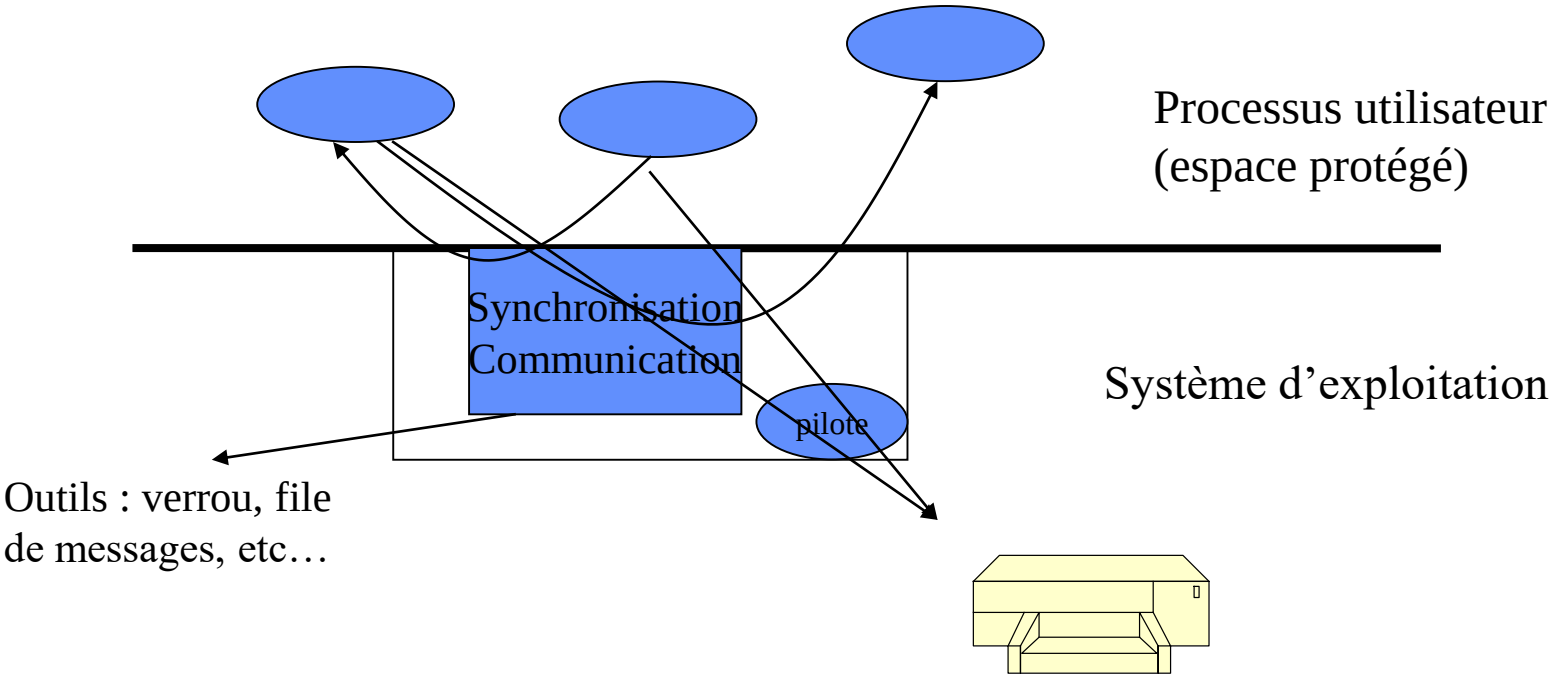
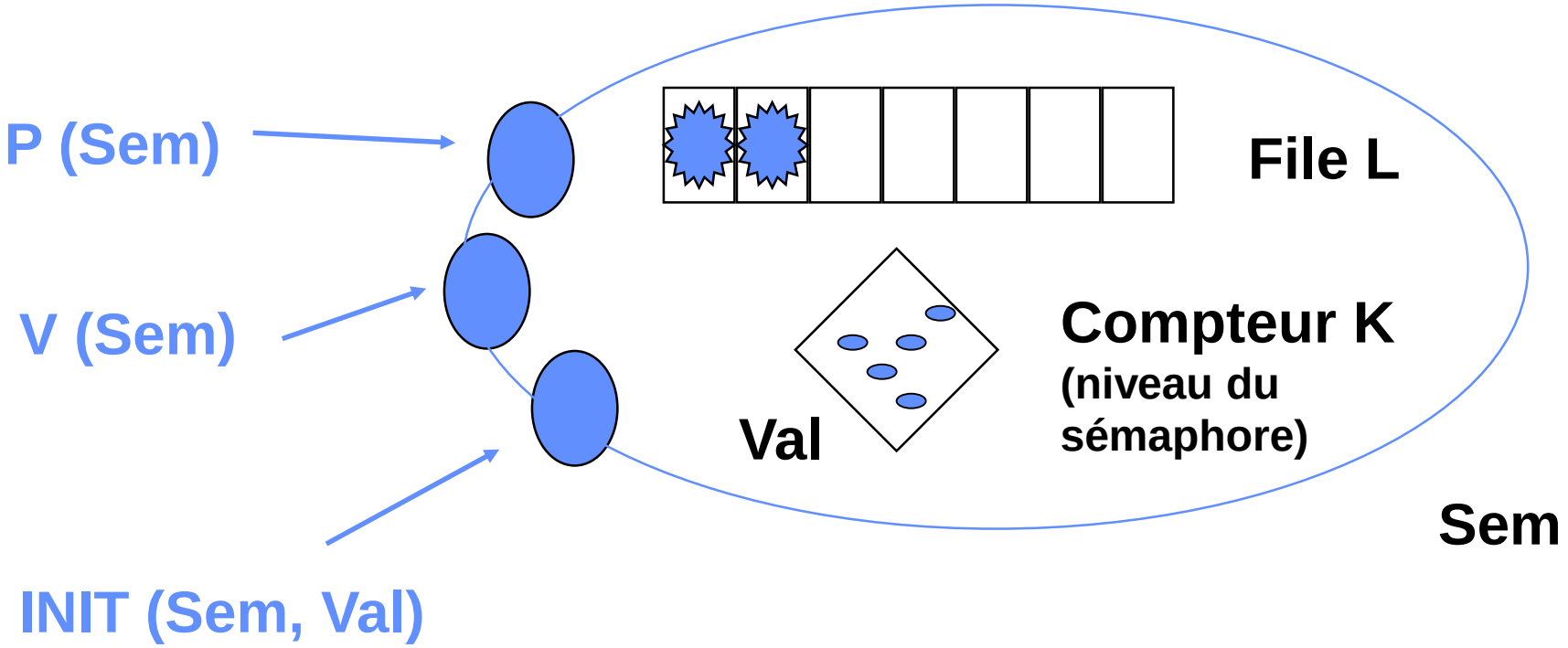
Synchronisation entre processus



Les sémaphores

- **Le système d'exploitation offre un outil appelé sémaphore, constitué d'un compteur K et d'une file L**
- **Trois opérations atomiques sont appliqués au semaphore Sem:**



Les sémaphores

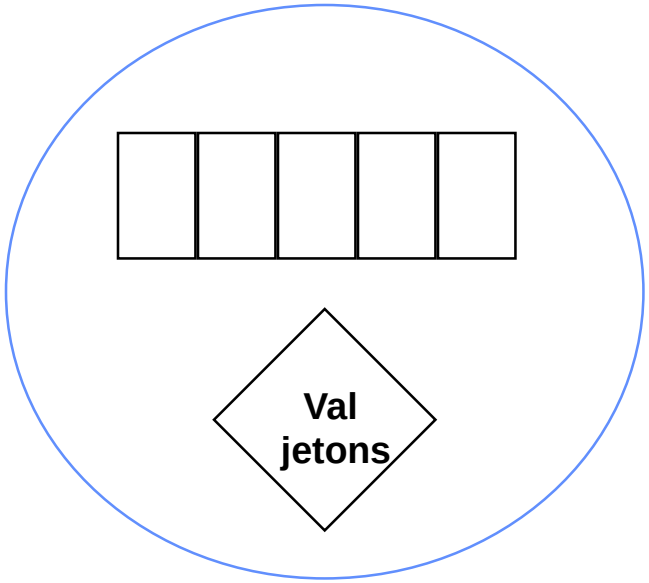
- **Un sémaphore Sem peut être vu comme un distributeur de jetons; le jeton étant un droit à poursuivre son exécution**
  - **l'opération INIT (Sem, Val) fixe le nombre de jetons initial**
  - **l'opération P(Sem) attribue un jeton au processus appelant si il en reste sinon bloque le processus dans Sem.L**
  - **l'opération V(Sem) restitue un jeton et débloque un processus de Sem.L si il en existe un.**

Les sémaphores

- **Opération Init (Sem, Val)**

Init (Sem, Val)

début
masquer_it
Sem. K := Val ***(jetons);***
Sem. L := ∅
demasquer_it
fin

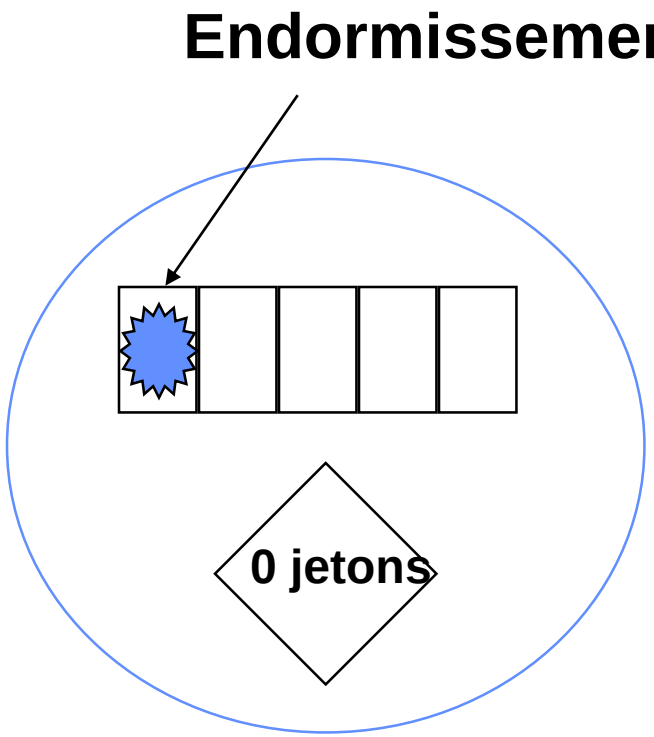


Les sémaphores

- **Opération P (Sem)**

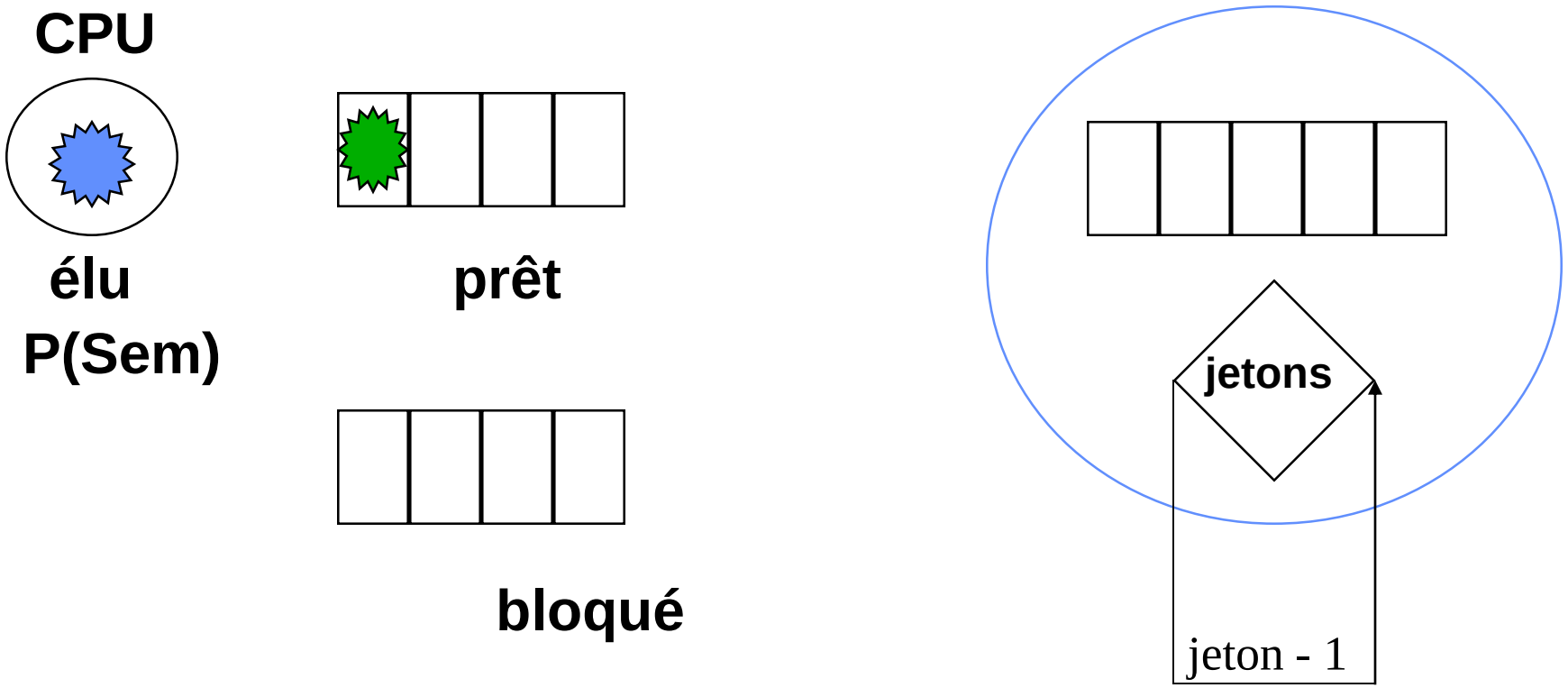
P (Sem)

début
masquer_it
Si (Sem.K > 0) alors
Sem.K = Sem.K – 1; **(donner jeton)**
sinon
ajouter ce processus à Sem.L
bloquer ce processus
fsi
demasquer_it
fin



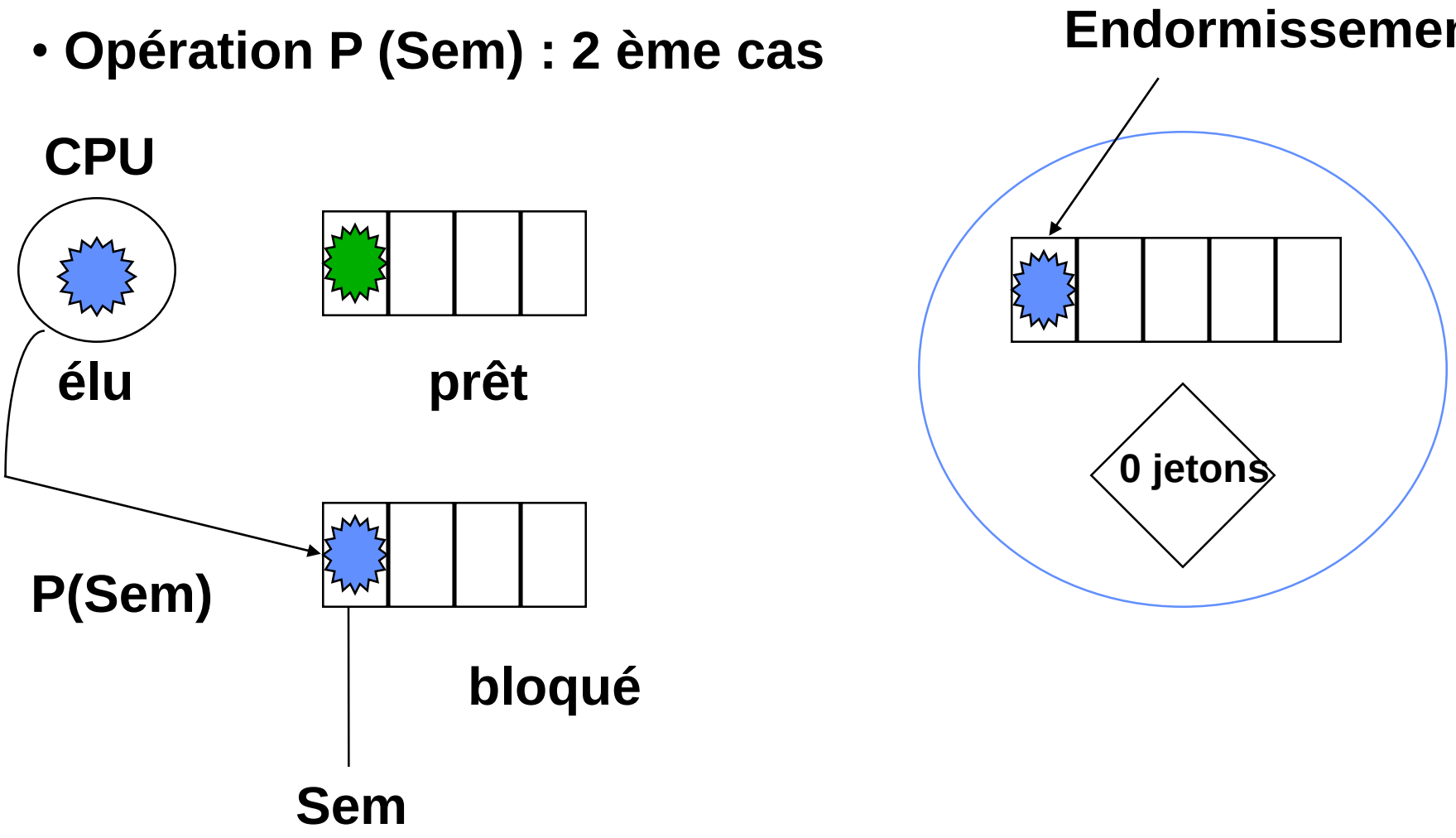
Les sémaphores

- **Opération P (Sem) : 1er cas**



Les sémaphores

- **Opération P (Sem) : 2 ème cas**

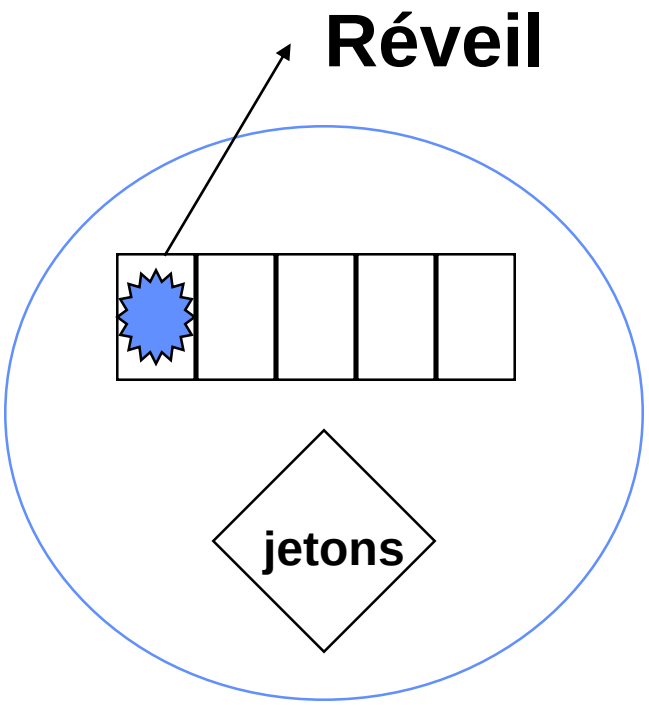


Les sémaphores

- **Opération V (Sem)**

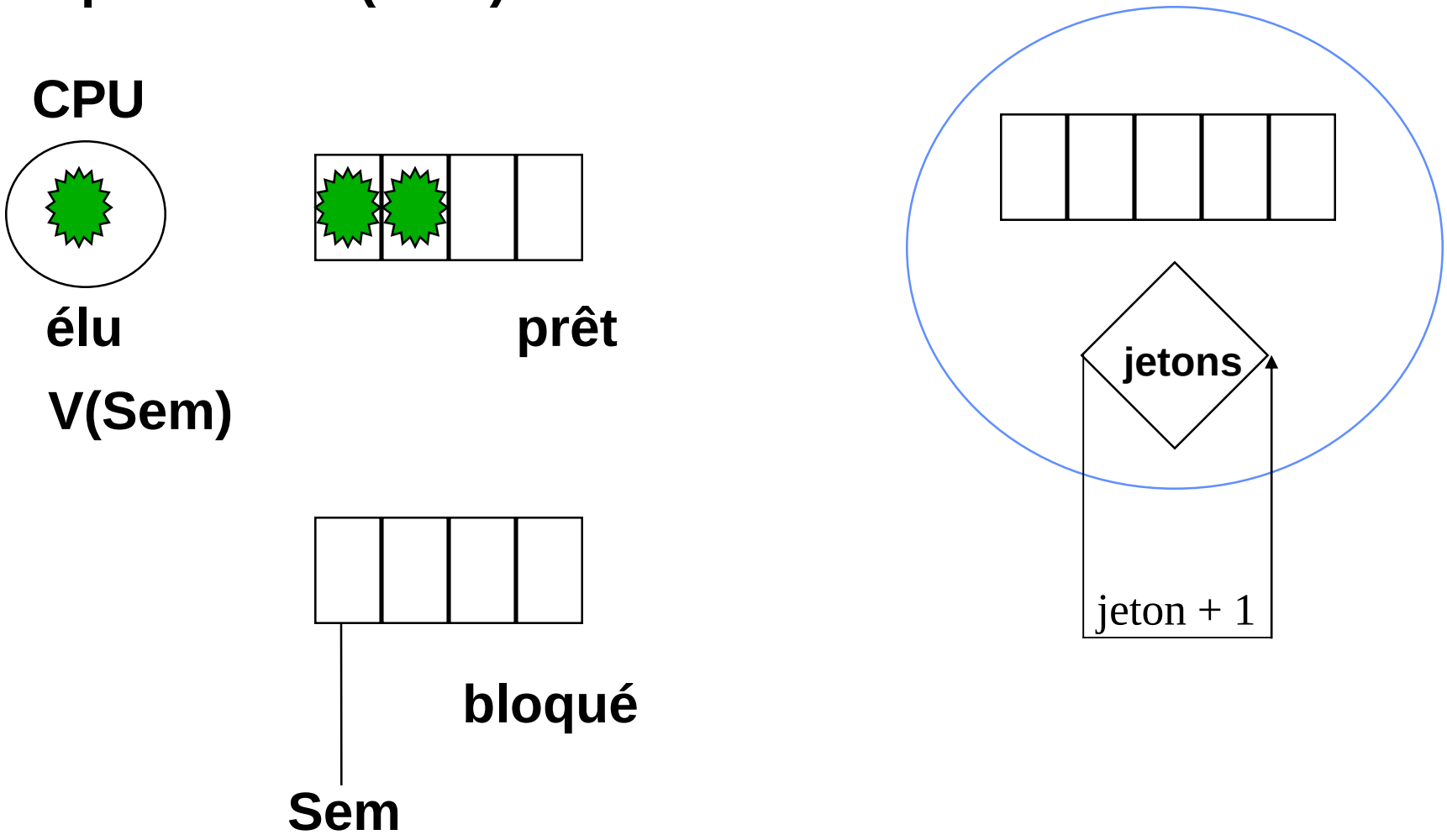
V (Sem)

début
masquer_it
Sem.K = Sem.K + 1; (rendre jeton)
Si il y a un processus en attente de jeton dans Sem.L
alors
sortir un processus de Sem.L
Sem.K = Sem.K – 1; *(donner jeton)*
réveiller ce processus
fsi
demasquer_it
fin



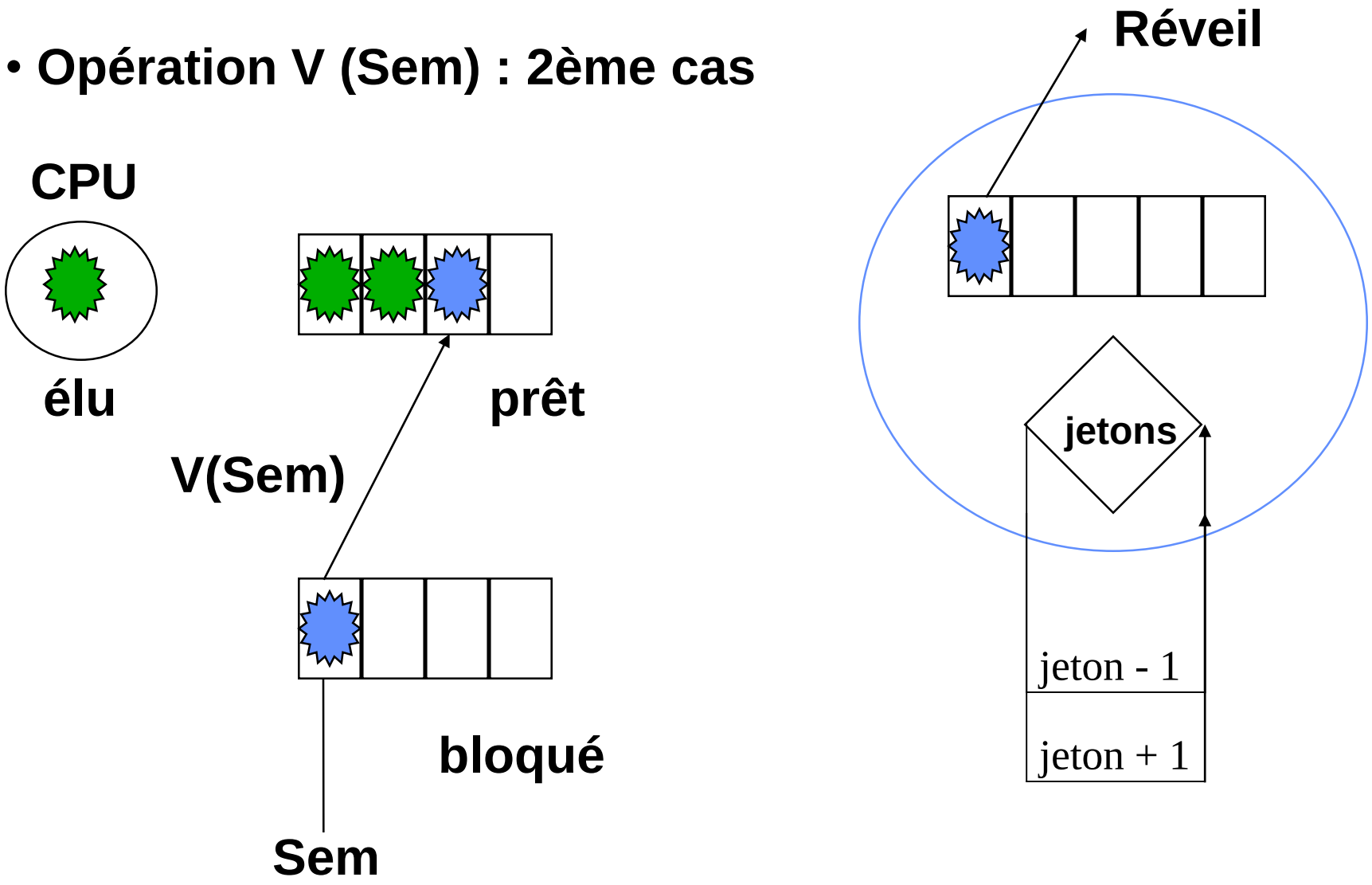
Les sémaphores

- **Opération V (Sem) : 1er cas**



Les sémaphores

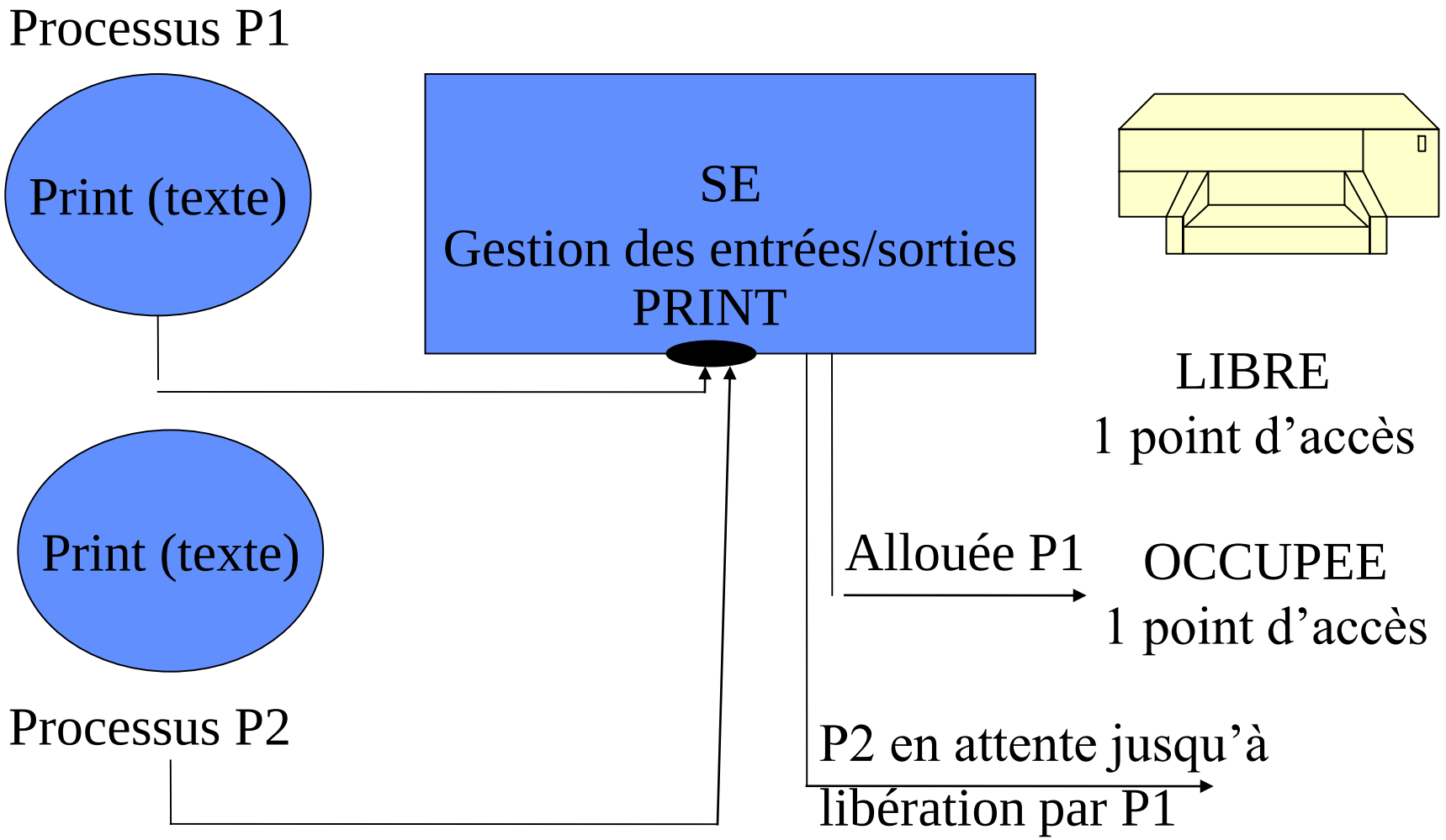
- **Opération V (Sem) : 2ème cas**



Notion de ressources
Exemple

ressource critique à un seul point d'accès

- Ressource matérielle : imprimante



Section critique et exclusion mutuelle

Processus
Début

Entrée Section Critique

Ressource Critique
IMPRIMANTE

Sortie Section Critique
Fin

- **Ressource utilisable par un seul processus à la fois**

SECTION CRITIQUE
(code d'utilisation de la ressource critique)

☞ L'entrée et la sortie de SC doivent assurer qu'à tout moment, un seul processus s'exécute en SC (exclusion mutuelle)

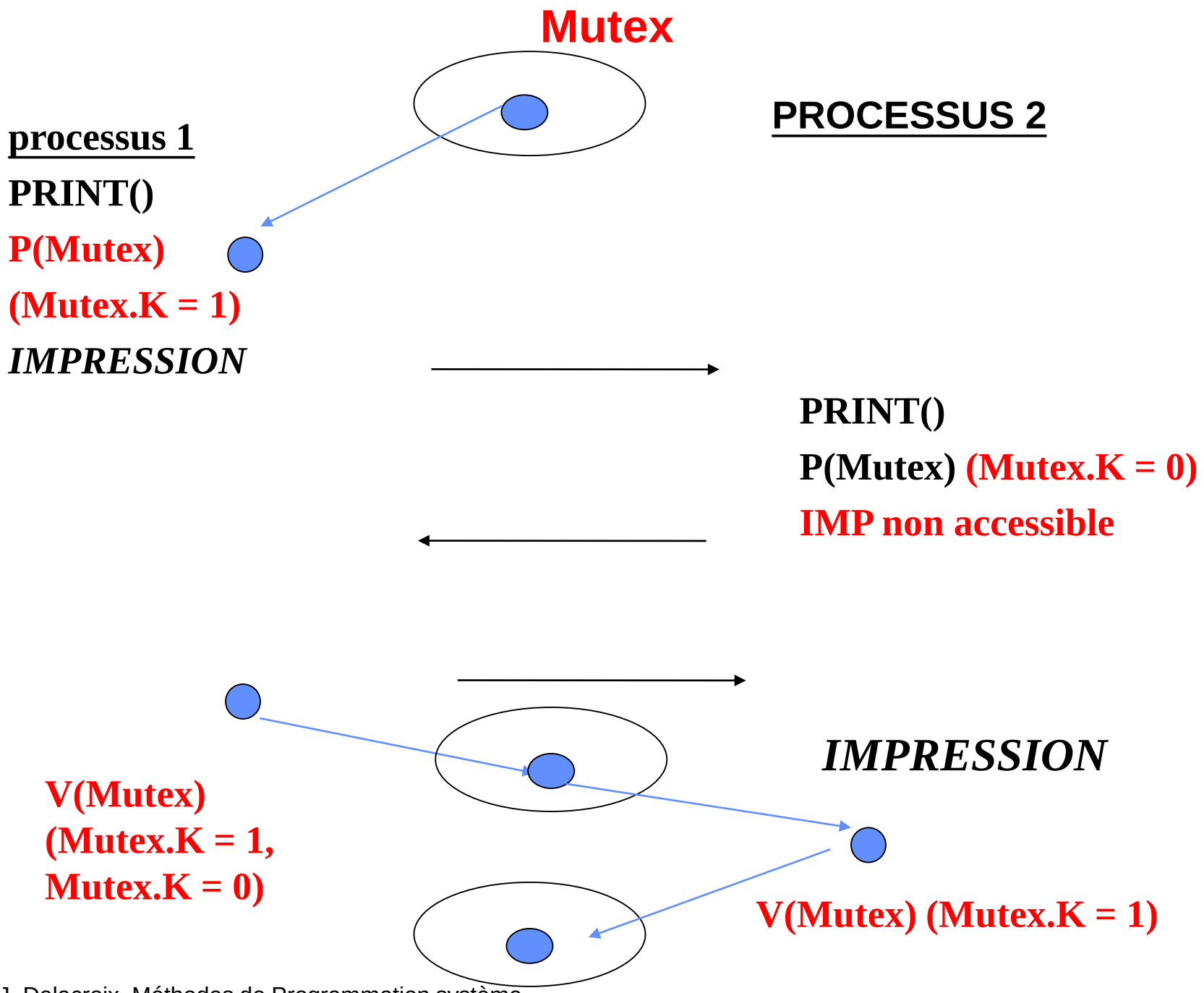
Section critique avec sémaphore

1 seul processus en section critique
=> 1 seul jeton
Sémaphore Mutex initialisé à 1

P (Mutex) — **Entrée section_critique**

Section Critique

V (Mutex) — **Sortie section_critique**


Mutex
processus 1
PRINT()
P(Mutex)
(Mutex.K = 1)
IMPRESSION
PROCESSUS 2
PRINT()
P(Mutex) (Mutex.K = 0)
IMP non accessible
IMPRESSION
V(Mutex)
(Mutex.K = 1,
Mutex.K = 0)
V(Mutex) (Mutex.K = 1)

Synchronisation et communication entre processus

Sémaphores Linux

Sémaphores Linux

- Famille des IPCs, un ensemble de sémaphores est identifié par une clef.

- Les opérations P, V et ATT (attente qu'une valeur de sémaphore soit nulle) s'effectuent sur un tableau de sémaphores, atomiquement

  ↳ L'ensemble des opérations est réalisée avant que le processus puisse poursuivre son exécution..

Sémaphores Linux

- Famille des IPCs, un ensemble de sémaphores est identifié par une clef.

- Les opérations P, V et ATT (attente qu'une valeur de sémaphore soit nulle) s'effectuent sur un tableau de sémaphores, atomiquement

  ↳ L'ensemble des opérations est réalisée avant que le processus puisse poursuivre son exécution..

Sémaphores Linux

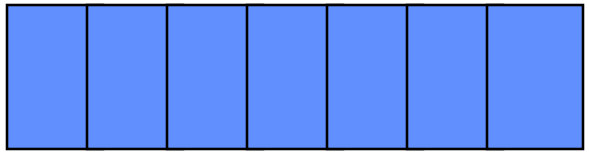
- Création et recherche d'un ensemble de sémaphore

int semget (key_t cle, int nsems, int semflg);

Création ou recherche d'un ensemble de n sémaphores identifiés par clé.

Retourne un identifiant interne de type entier

Semflg = constantes IPC_EXCL, IPC_CREAT, 0



nsems

```
Struct semaphore {
atomic_t count ; /* compteur */
int sleepers; /* nombre de processus endormis */
Wait_queue_head_t wait; /* file d'attente */ }
```

Sémaphores Linux

- Opérations sur les sémaphores

int semop (int semid, struct sembuf *ops, unsigned nsops);

Réalisation d'un ensemble d'opérations (nsops) décrite chacune dans une structure sembuf sur l'ensemble de sémaphore semid.

```
struct sembuf {
    unsigned short sem_num;/* numéro du sémaphore dans le tableau */
    short sem_op; /* opération à réaliser sur le sémaphore */
    short sem_flg; /* options */
     };
```

– si sem_op est négatif, l'opération à réaliser est une opération P;
– si sem_op est positif, l'opération à réaliser est une opération V;
– si sem_op est nul, l'opération à réaliser est une opération ATT.

Sémaphores Linux

- Initialiser un sémaphore

int semctl (int semid, int semnum, int cmd, union semun arg);

semctl (semid, 0, SETVAL, 3) initialisation à la valeur 3 du sémaphore 0 dans l'ensemble désigné par l'identifiant semid.

- Détruire un ensemble de sémaphores

int semctl (int semid, 0, IPC_RMID, 0);

```c
#include <stdio.h>
#include <pthread.h>
#include <sys/ipc.h>
#include <sys/sem.h>

int i, nb_place;
int semid;
struct sembuf operation;

void reservation()
{
/* opération P */
operation.sem_num = 0;
operation.sem_op = -1;
operation.sem_flg = 0;
semop (semid, &operation, 1);
nb_place = nb_place - 1;
/* opération V */
operation.sem_num = 0;
operation.sem_op =  1;
operation.sem_flg = 0;
semop (semid, &operation, 1);
}
```

```c
main()
{ pthread_t num_thread[3];


/* création d'un sémaphore initialisé à
la valeur 1 */
semid = semget (12, 1,
                IPC_CREAT|IPC_EXCL|0600);
semctl (semid, 0, SETVAL, 1);


for(i=0; i<3; i++) {
pthread_create(&num_thread[i], NULL,
(void *(*)())reservation, NULL);
pthread_join(num_thread, NULL);
semctl (semid, 0, IPC_RMID, 0)
}
```