

Sintaxis de Semáforos

```
#include<semaphore.h>
```

Declarar una variable semáforo

```
sem_t sem;
```

Sintaxis de la función sem_init(): Inicia el valor del semáforo

```
int sem_init(sem_t *sem, int pshared, unsigned int value)
```

retorna: éxito/error: true(1)/false(0)

sem: semáforo a iniciar

pshared: indica si la sección crítica es compartida entre procesos,

0: compartida / 1 no compartida

value: Número máximo de procesos dentro de la región crítica.

Llamar a la función: sem_init(&sem, 0, 1) // 0: compartida, sem = 1 ≡ valor inicial

Sintaxis de la función sem_wait(): inicia un proceso de sección crítica

```
int sem_wait(sem_t *sem);
```

retorna: éxito/error: true(1)/false(0)

Ejecuta:

Sem--;

if(sem<0) bloquear este proceso; else este proceso entra a la sección crítica

Llamar a la función: sem_wait(&sem);

Sintaxis de la función sem_post(), la cual finaliza un proceso de sección crítica

```
int sem_post(sem_t &sem):
```

retorna: éxito/error: true(1)/false(0)

Ejecuta:

// equivalente de **signal()**

sem++; el proceso desbloquea su sección crítica

if(sem<=0) este proceso sale de sección crítica y desbloquea otro proceso;

Llamar a la función: sem_post(&sem);

Ejemplo de uso:

```
sem_t sem; // declara semáfo sem
```

```
sem_init(&sem, 0, 1); // inicia el semáforo (&s, compartidoEntreProcesos=Sí, inicio=1)
```

```
sem_wait(&sem); // inicio de sección crítica
```

```
sem_post(&sem); // fin de sección crítica
```

```
/*
```

Programación concurrente C y C++: librería omp.h

Programación con procesadores: librerías C y C++: openmpi, mpich

Libro Pag 114 aprox.

OpenMP (Programación paralela en memoria compartida): <https://b-ok.lat/book/739105/d4f7ab>

```
*/
```