

20/11/2025 - Sistemas Operativos (Ude@)

1. Repaso clase anterior

Primitivas

① Hilos - Concurrencia

$\text{§} \rightarrow f()$

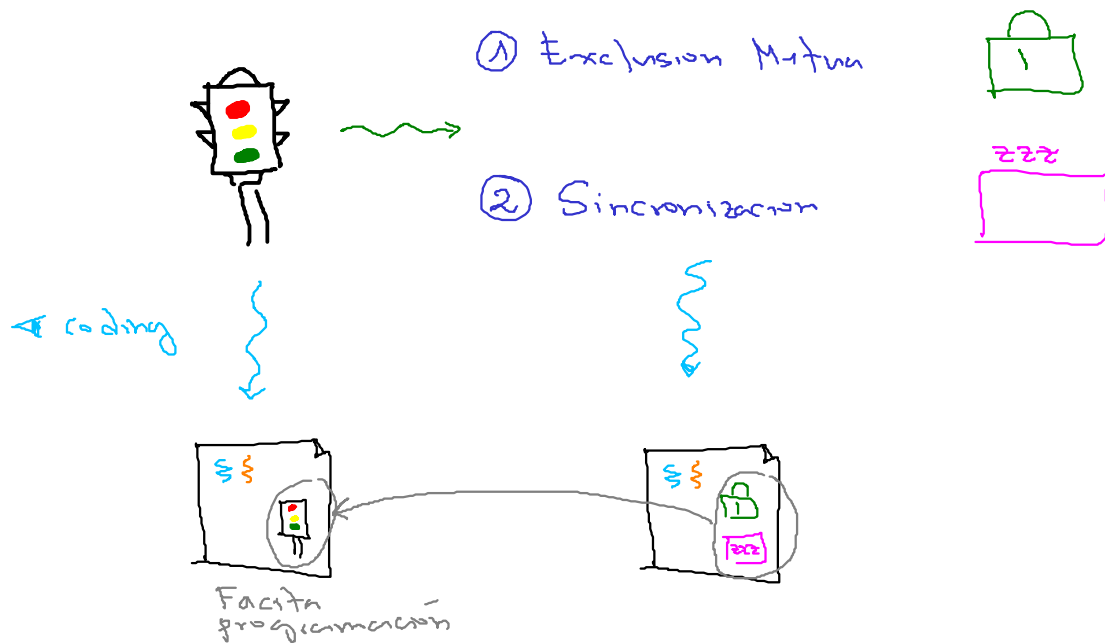
② Locks - Exclusion mutua



③ Variables de condición (CV) - Sincronización



2. Semaforo



Usando semaforos puedo implementar

1. Lock

$$\text{Lock} = f(\text{semaF.}) \quad \checkmark$$

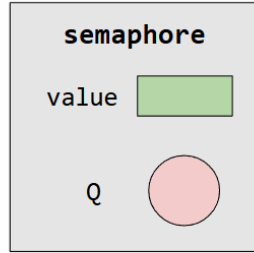
2. CV

$$\text{CV} = f(\text{semaFu.}) \quad \checkmark$$

Clase
atributos
metodos



1. Init.
2. wait
3. post



```
typedef struct {
    int value;
    struct process *Q;
} sem_t;
```

1. init

```
sem_init(sem_t *s, int value) {
    s->value = initial;
}
```

2. wait

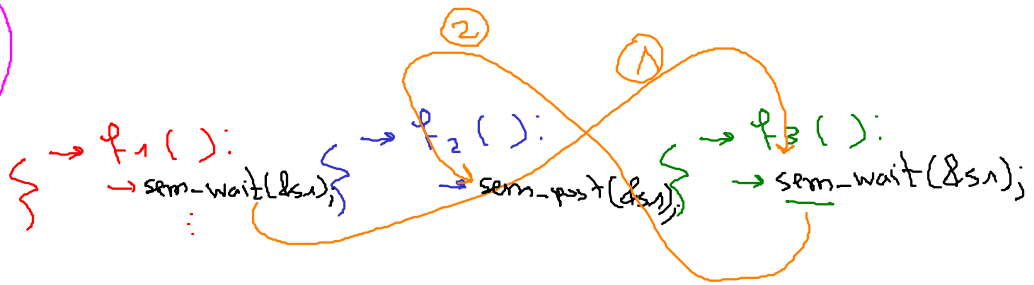
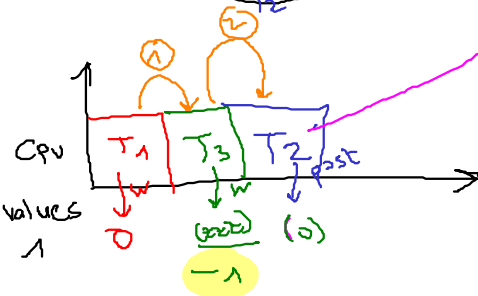
```
void sem_wait(sem_t *s) { // Must be executed atomically
    s->value--;
    if (s->value < 0) {
        add this process to s->Q;
        block();
    }
}
```

3. post

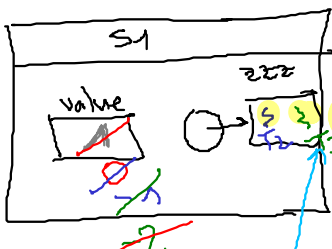
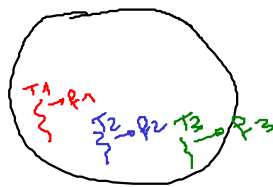
```
void sem_post(sem_t *s) { // Must be executed atomically
    s->value++;
    if (s->value <= 0) {
        remove a process P from s->Q;
        wakeup(P);
    }
}
```



```
Sem S1;
sem_init(&S1, 1);
...
```



Otra situación (BLDQWU=0)



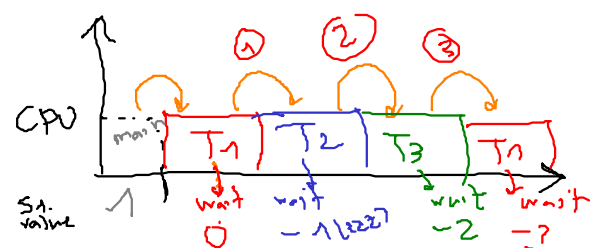
```
main()
{
    Sem S1;
    sem_init(&S1, 1);
    ...
}
```

T1 → P1
sem_wait(&S1)

T2 → P2
sem_wait(&S2)

T3 → P3
sem_wait(&S3)

sem_wait(&S1);



cuantos
hijos
hay bloquados

4. Uso de semaforos (Binario)

a. Semaforo como candado (lock)



- Puedo usar un semaforo inicializado en 1 para asegurar exclusion mutua

Código de implementación del lock empleando semaforos



```
1 sem_init(sem_t *s, int initval) {  
    s->value = initval  
}  
2 sem_wait(sem_t *s) {  
    while (s->value <= 0)  
        put_self_to_sleep();  
    s->value--;  
}  
3 sem_post(sem_t *s) {  
    s->value++;  
    wake_one_waiting_thread();  
}
```

```
typedef struct __lock_t {  
    // whatever data structs you need goes here  
    sem_t value;  
} lock_t;  
1 void init(lock_t *lock) {  
    // init code goes here  
    sem_init(&lock->value, 0, 1);  
}  
2 void acquire(lock_t *lock) {  
    // lock acquire code goes here  
    sem_wait(&lock->value);  
}  
3 void release(lock_t *lock) {  
    // lock release code goes here  
    sem_post(&lock->value);  
}
```



1. init(...)
2. acquire()
3. release(...)

2. Mediante el uso de semaforos puedo implementar variables de condici3n



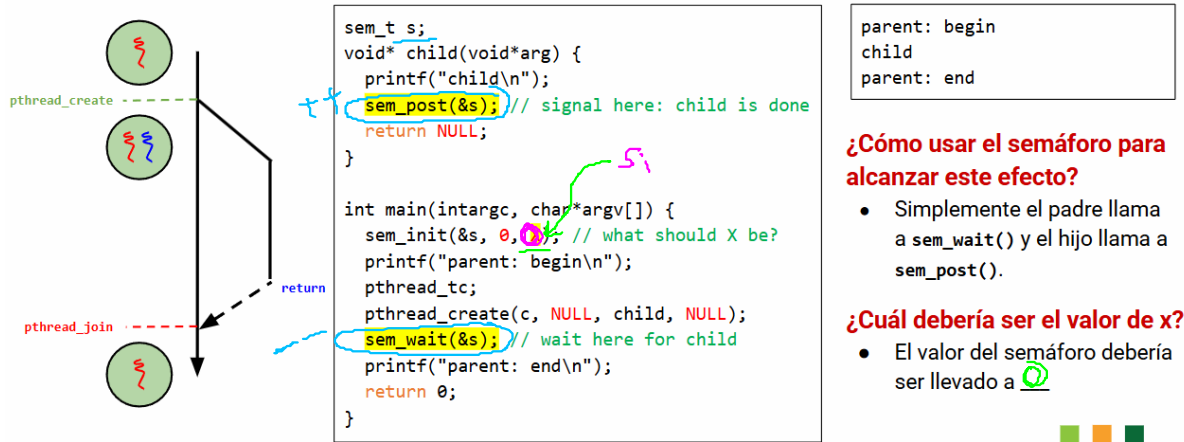
Puedo lograr sincronizaci3n entre hilos?

Rta: Si



Como?

- Los **semáforos** también son útiles para **ordenar eventos** en un programa concurrente lo que los hace aptos para ser usados como **primitiva para controlar el orden de ejecución**.



- Puedo usar un semáforo inicializado en 0 para permitir sincronización

↓
Para explicar hagamos Prueba de escritorío