Lectores-Escritores con Semáforos

- Cualquier numero de lectores puede acceder a la sección critica.
- Solo puede acceder un escritor.
- Si un escritor esta accediendo, ningún lector puede acceder

Lectores con prioridad

- Semaforo x = 1, mutex = 1;

Lectores con prioridad

```
Semaforo x = 1, mutex = 1;
```

Lector	*Escritor*

`...`
`while (TRUE) {`
`  wait(x);`
`  lectores ++;`
`  if(lectores == 1)`
`  wait(mutex);`
`  signal(x);`
`  LEER();`
`  wait(x);`
`  lectores--;`
`  if(lectores == 0)`
`  signal(mutex);`
`  signal(x);`
`}`
	`...`
	`while (TRUE) {`
	`  wait(mutex);`
	`  ESCRIBIR();`
	`  signal(mutex);`
	`}`

Lectores con *wait_for_zero*

- Se introduce una nueva primitiva, parte del estandar POSIX de semáforos.
- La primitiva es *wait_for_zero*, y lo que hace es esperar y desbloquear un proceso cuando el semáforo tiene valor 0.

```
entrada_lectores()
/* Cuando un lector quiera entrar tendrá que esperar que no haya escritores */
{
    wait_for_zero(escritores),
    signal(lectores);
}

salida_lectores()
{
    wait(lectores);
}
```

```
entrada_escritores()
/* Cuando entra un escritor espera que no haya lectores y establece exclusión mutua entre escritores */
{
    signal(escritores);
    wait_for_zero(lectores);
    wait(mutex);
}
salida_escritores()
{
    signal(mutex);
    wait(escritores);
}
```

Lectores con *wait_for_zero*

- Hay problemas en la entrada de lectores, el *wait_for_zero* y el *signal* de la entrada de lectores deben ser atómicas.
  - Si se interrumpe tras el wait_for_zero y luego entra un escritor, tendremos lectores a 1 y escritores a 1
- Se hace con un mutex adicional o con operaciones sobre *arrays* de semáforos.

```
sctruct sembuf sbuf[2];
// escritores:
sbuf[0].sem_num = 0;
sbuf[0].sem_op = 0;
//lectores:
sbuf[1].sem_num = 1;
sbuf[1].sem_op = 1;
sem_op (le, sbuf, 2);
```

RCU (Read Copy Update)

- Hay varios lectores y escritores
- Se hace una versión para el escritor y que así los lectores puedan seguir leyendo la versión anterior.
- Los lectores no se bloquean nunca a diferencia del lectores_escritores clásico, pero tiene una carga adicional de liberar memoria

```
rcu_read_lock();          // entrada lectores
rcu_read_unlock();        //salida lectores

synchronice_rcu;          //entrada/salida escritores
call_rcu;                 //liberar las copias que ya no se necesitan

rcu_assign_pointer();
rec_dereference();
```