Solución al Problema de los Lectores y Escritores

```
1  int data = 5;        /* recurso compartido */
2  int nr_readers = 0; /* número de lectores en la SC o que quieren entrar a ella */
3  sem_t sem_nreaders; /* Para controlar acceso a nr_readers */
4  sem_t sem_read_write;  /* Exclusión mutua lector-escr. y escr.-escr. */
5
6  void main(void){
7     pthread_t th1, th2, th3, th4;
8
9     sem_init(&sem_read_write, 0, 1);
10    sem_init(&sem_nreaders, 0, 1);
11
12    pthread_create(&th1, NULL, Reader, NULL);
13    pthread_create(&th2, NULL, Writer, NULL);
14    pthread_create(&th3, NULL, Reader, NULL);
15    pthread_create(&th4, NULL, Writer, NULL);
16
17    pthread_join(th1, NULL);   pthread_join(th2, NULL);
18    pthread_join(th3, NULL);   pthread_join(th4, NULL);
19
20    /* destrucción semáforos */
21    sem_destroy(&sem_read_write);    sem_destroy(&sem_nreaders);
22    exit(0);
23 }
```

```
24 void Reader(void) {
25   while(1){
26     sem_wait(&sem_nreaders);
27     nr_readers = nr_readers + 1;
28     if (nr_readers == 1)
29       sem_wait(&sem_read_write);
30     sem_post(&sem_nreaders);
31
32     /* Lectura recurso compartido */
33     printf("%d\n", data);
34
35     sem_wait(&sem_nreaders);
36     nr_readers = nr_readers - 1;
37    if (nr_readers == 0)
38       sem_post(&sem_read_write);
39     sem_post(&sem_nreaders);
40   }
41 }
```

```
42 void Writer(void) {
43   while(1){
44     sem_wait(&sem_read_write);
45
46     /* Modificación rec. compartido */
47     data = data + 2;
48
49     sem_post(&sem_read_write);
50
51   }
52 }
```

En esta versión del Problema de los Lectores y Escritores, el recurso compartido es una variable entera llamada `data`. La sección crítica (SC) de los lectores es la sentencia de la línea 33, mientras que la SC de los escritores está constituida por la sentencia de la línea 47.

En la solución proporcionada, se mantiene un contador `nr_readers` para llevar la cuenta del número de lectores que intentan entrar en la sección crítica o que ya están dentro de esta sección. Por lo tanto, cada lector incrementará la variable antes de acceder a la sección crítica. Análogamente, `nr_readers` se decrementará justo después de que cada lector abandone la sección crítica. Para evitar que esa variable se modifique simultáneamente por varios lectores —lo cual puede dar lugar a problemas de concurrencia—, el semáforo `sem_nreaders` se emplea, a modo de cerrojo, para garantizar exclusión mutua. Por lo tanto, el contador de ese semáforo se inicializa a 1 (línea 9) y las declaraciones que

incrementan y disminuyen el contador (en las líneas 27 y 36, respectivamente) se encierran entre wait *()* y *signal()*[1] en ese semáforo.

Por otra parte, para garantizar la exclusión mutua entre múltiples escritores, o un escritor y un lector, se utiliza otro semáforo llamado `sem_read_write`, cuyo contador también se inicializa a 1 (línea 8).

Para ilustrar el correcto funcionamiento de esta solución, consideraremos por separado las tres siguientes restricciones del Problema de los Lectores y Escritores:

1. **Exclusión mutua escritor-escritor:** La solución no debe permitir que múltiples escritores accedan a su sección crítica (línea 47) simultáneamente.
2. **Exclusión mutua escritor-lector:** No se debe permitir a ningún lector acceder a su sección crítica (línea 33), cuando un escritor está dentro de su sección crítica. (línea 47).
3. **Exclusión mutua lector-escritor:** No se debe permitir que un escritor entre en su sección crítica (línea 47), mientras uno o más lectores estén dentro de su sección crítica (línea 33).

Exclusión mutua escritor-escritor

Como puede observarse, la sección crítica de los escritores está encerrada entre wait *()* y *signal()* sobre un semáforo cuyo contador se inicializó a 1. Ésta es exactamente la "receta" para la solución del problema de la sección crítica, analizada en clase. Por lo tanto, sólo se permite que un escritor (como mucho) entre en su sección crítica. Cuando un escritor se encuentre dentro de la sección crítica, el contador asociado al semáforo `sem_read_write` siempre será menor o igual que cero, lo cual impide que otros escritores entren en la sección crítica.

Exclusión mutua escritor-lector

Veamos ahora por qué cuando un escritor está dentro de su sección crítica, la solución no permite entrar a ningún lector a su sección crítica. Por simplicidad, consideraremos el caso en el que un escritor está dentro de su sección crítica (línea 47) y no hay lectores en el sistema. Por lo tanto, el estado de las variables del programa será el siguiente[2]:

```
nr_readers = 0
sem_read_write.c = 0
sem_nreaders.c = 1
```

Ahora, supongamos que un lector quiere entrar en la sección crítica. Al hacerlo, invocará wait *()* sobre el semáforo `sem_nreaders` (línea 26). Como el contador de ese semáforo vale 1, la operación wait *()* retornará inmediatamente tras decrementar de forma atómica el contador del semáforo. A continuación, el lector procederá a incrementar la variable `nr_readers` (sentencia de la línea 27). Ya que éste es el primer lector que intenta acceder a la sección crítica, la condición de la sentencia de la línea 28 será cierta, por lo que este lector invocará *wait()* sobre el semáforo `sem_read_write` (línea 29). Como el contador del semáforo `sem_read_write` es 0 actualmente (el escritor decrementó el contador al entrar en su SC) el lector se bloqueará en la sentencia de la línea 29. De este modo, el primer lector no puede entrar en la sección crítica. En esta situación tendremos lo siguiente:

[1] Nótese q

[2] El atributo "c" de un semáforo representa el contador interno del semáforo.

```
nr_readers = 1
sem_read_write.c = -1
sem_nreaders.c = 0
```

Consideremos ahora que un segundo lector intenta entrar en la sección crítica. Como el contador de `sem_nreaders` es ahora 0, el segundo lector permanecerá bloqueado en la sentencia de la línea 26 (`sem_wait(&sem_nreaders)`). Adicionalmente, cualquier lector adicional que quiera entrar en su sección crítica se quedará bloqueado en esa misma sentencia.

Para finalizar el análisis sobre esta segunda restricción del problema, merece la pena ilustrar cómo dos lectores bloqueados podrán acceder a la sección crítica justo después de que el escritor salga de su sección crítica. La secuencia de acciones que tendrán lugar es la siguiente:

1. El escritor sale de su sección crítica, e invoca `sem_post(&sem_read_write)` en la línea 49.
2. En consecuencia, el primer lector (bloqueado en la sentencia de la línea 29) se despertará, retornando de la operación *wait()*. Acto seguido invocará *signal()* sobre `sem_nreaders` (línea 30).
3. Como resultado de lo anterior, el segundo lector, que se encontraba bloqueado en la sentencia de la línea 26, se despiertará, y accederá a la sección crítica tras incrementar el contador `nr_readers`. Nótese que el segundo lector (o cualquier lector que entre a continuación) no ejecuta la sentencia de la línea 29, ya que la condición del *if* es falsa.

Exclusión mutua lector-escritor

Para concluir analizaremos por qué la solución garantiza que cuando uno o más lectores están dentro de su sección crítica, ningún escritor pueda entrar en su sección crítica. Para ello, asumiremos que tres lectores están actualmente dentro de su sección crítica (línea 33) y que no hay escritores en el sistema. Por lo tanto, tenemos lo siguiente:

```
nr_readers = 3
sem_read_write.c = 0
sem_nreaders.c = 1
```

En estas circunstancias el valor del contador del semáforo `sem_read_write` es 0 porque el primer lector que entró en la sección crítica ejecutó la sentencia de la línea 29 (`sem_wait(&sem_read_write)`). Por lo tanto, cualquier escritor que intente acceder a su sección crítica permanecerá bloqueado en la sentencia de la línea 44 (`sem_wait(&sem_read_write)`) hasta que los lectores salgan de la sección crítica.

En este contexto *¿De qué forma acaba entrando en la SC un escritor que está actualmente esperando acceder a ella?* A pesar del hecho de que todos los lectores actualizarán la variable `nr_readers` (ejecutando la sentencia de la línea 36), sólo el último lector que salga de la sección crítica ejecutará la sentencia de la línea 38 (`sem_post(&sem_read_write)`). Esa sentencia garantiza que un escritor bloqueado en la línea 44 tendrá, tarde o temprano, la oportunidad de entrar en la sección crítica, al salir el último lector de su SC.