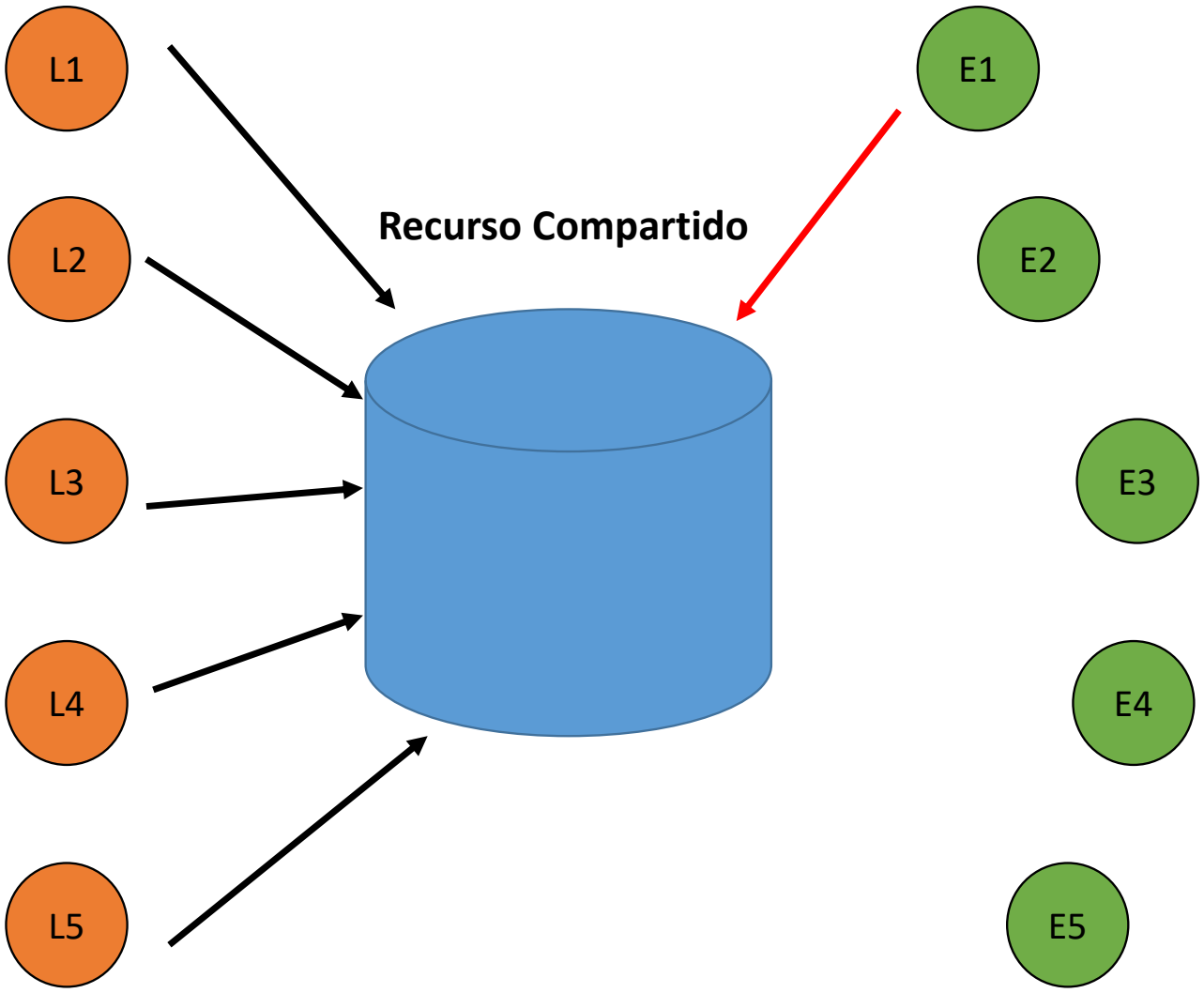
Algoritmos y Programación Paralela

Dra. Ing. Ana Cori Morón

EL PROBLEMA DE LOS LECTORES - ESCRITORES

- Es otro problema recurrente en sistemas operativos, pero difiere del problema anterior que n lectores si pueden acceder al recurso compartido simultáneamente, sin embargo, si un escritor y otro proceso acceden al recurso simultáneamente, se producirán efectos adversos.
- Debe satisfacer las siguientes condiciones:
  - Cualquier número de lectores puede leer el recurso simultáneamente.
  - Solo puede escribir en el recurso un escritor en cada instante.
  - Si un escritor está accediendo al recurso, ningún lector puede acceder al recurso.


L1
L2
L3
L4
L5
Recurso Compartido
E1
E2
E3
E4
E5

EL PROBLEMA DE LOS LECTORES - ESCRITORES

- Existen diferentes versiones para dar solución a este problema:
  - Prioridad en la lectura: Ningún lector debe esperar salvo un escritor haya obtenido permiso para usar el recurso.
  - Prioridad en la escritura: Un escritor debe realizar su escritura lo más pronto que sea posible.
  - Prioridad en la escritura sin espera ocupada

PROCESOS LECTORES-ESCRITORES

```
process type lector
  begin
    repeat
      protocolo de entrada;
      leer del recurso;
      protocolo de salida;
    forever
  end;
```

Sección crítica

```
process type escritor
  begin
    repeat
      protocolo de entrada;
      Escribir en el recurso;
      protocolo de salida;
    forever
  end;
```

Sección crítica

Programa Principal

```
Var LE: array[0..N] of lector
      ES: array[0..N] of escritor
Cobegin
      for i=0 to N do
            LE[i]; ES[i];
coend
```

Solución con Prioridad en la lectura:

- Se usará las siguientes variables:
- lec: cantidad de lectores que hay en el recurso.
- mutex: semáforo para asegurar la ejecución en exclusión mutua.
- writer: semáforo para asegurar la ejecución en exclusión mutua entre escritores.
- **Inicialización:**
  - Initial(mutex,1)
  - Initial(writer,1)
  - lec=0

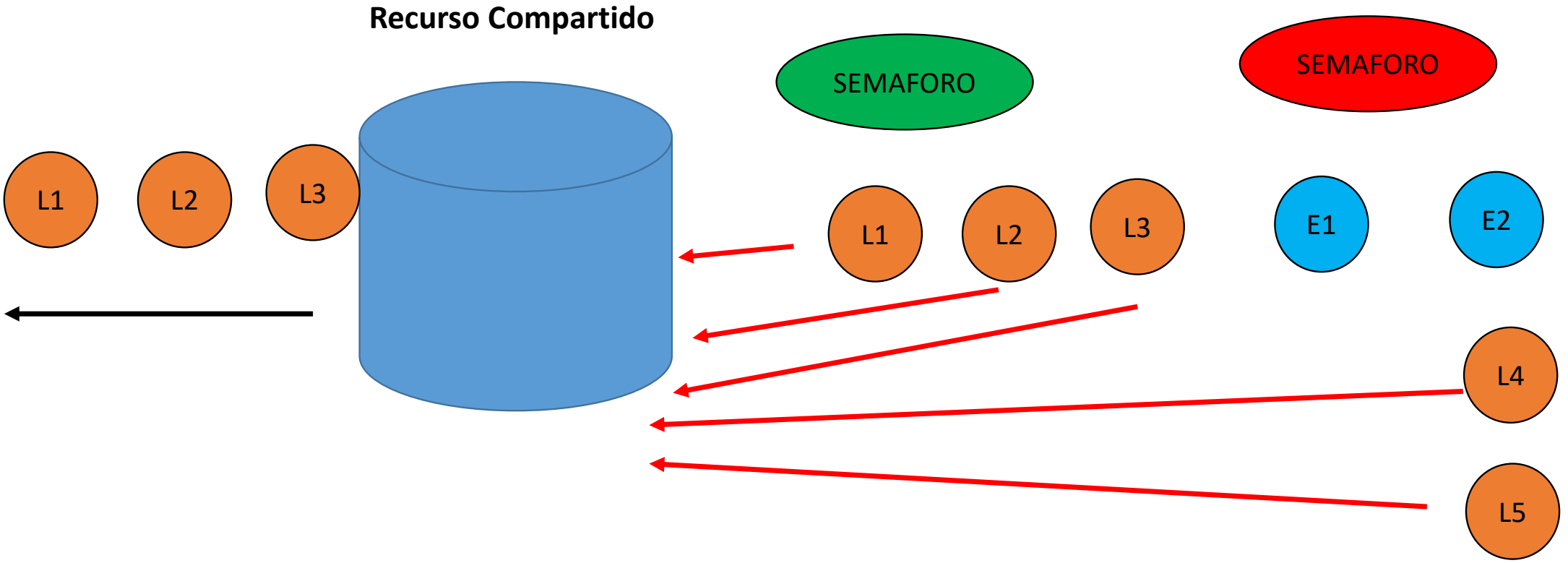

Recurso Compartido
SEMAFORO
SEMAFORO
L1
L2
L3
L1
L2
L3
E1
E2
L4
L5

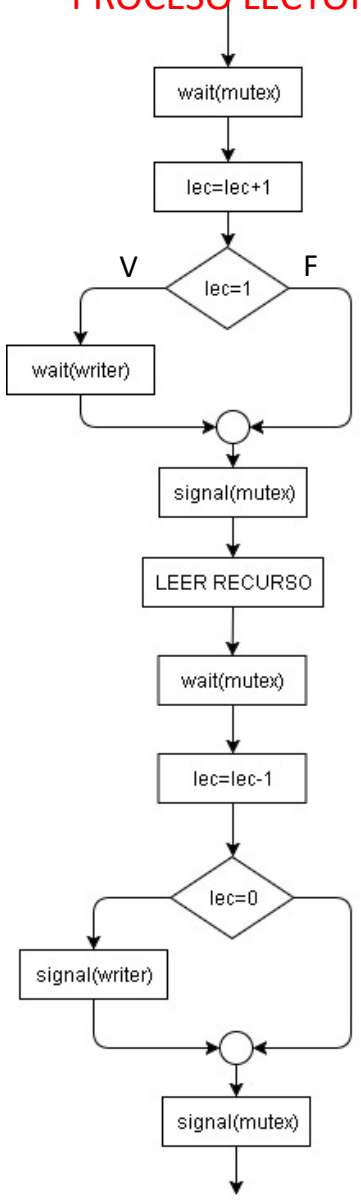
Diagrama de flujo

L1, L2, L3, E1, E2

Initial(mutex,1)
Initial(writer,1)
lec=0

PROCESO ESCRITOR

PROCESO LECTOR



EJECUCIÓN

T	0	1	2	3	4	5	6
lec	0						
mutex	1						
writer	1						
Plector							
Pescrit							

ELABORAR DOS EJECUCIONES DIFERENTES
Suponer 3 procesos lectores y 2 procesos escritores

Solución con Prioridad en la escritura: (con espera ocupa)

- Se usará las siguientes variables:
  - **lec: cantidad de lectores que están accediendo al recurso.**
  - **nee: cantidad de escritores esperando para acceder al recurso**
  - **mutex: semáforo para asegurar la ejecución en exclusión mutua.**
  - **writing: variable booleana que indica si hay un escritor accediendo al recurso.(true=escribiendo y false=no escribiendo)**
- **Inicializacion:**
  - Initial(mutex,1)
  - writing=false
  - lec=0
  - nee=0

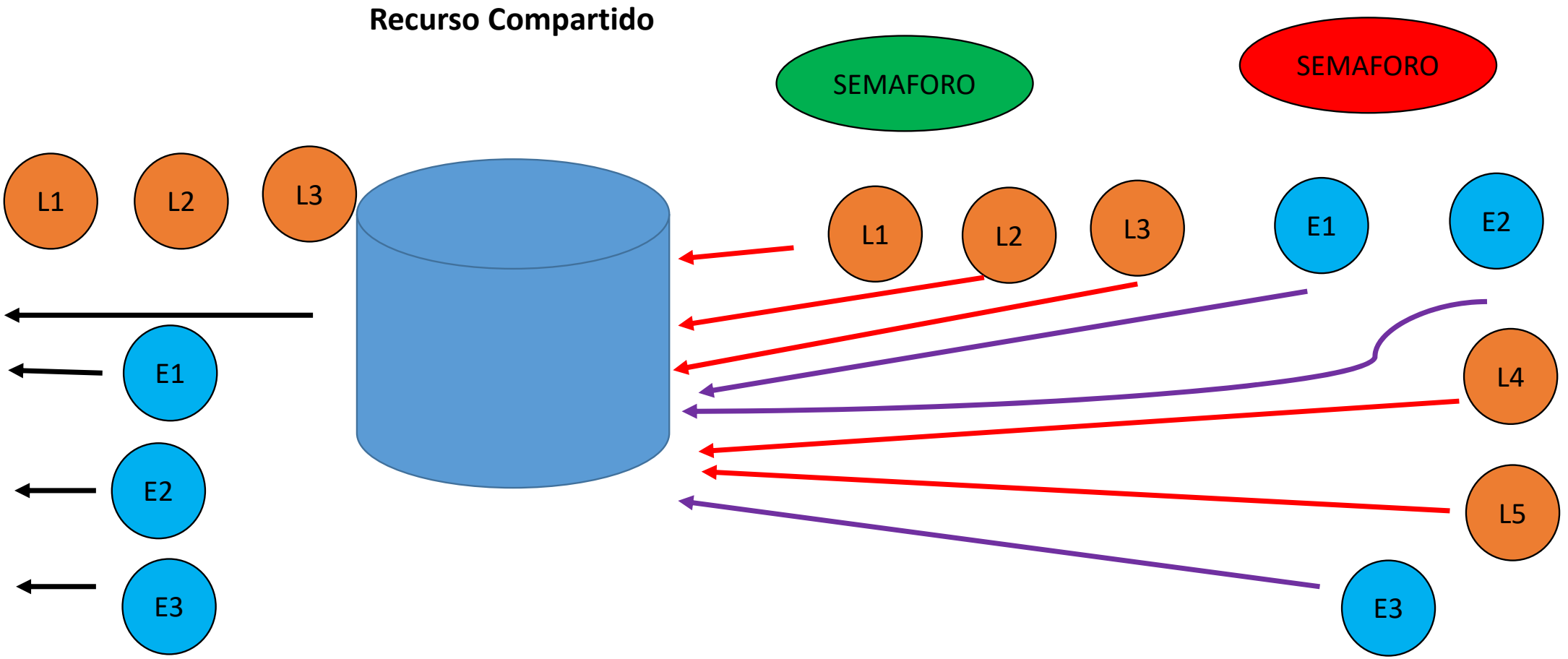

Recurso Compartido
SEMAFORO
SEMAFORO
L1
L2
L3
L1
L2
L3
E1
E2
E1
L4
E2
L5
E3
E3

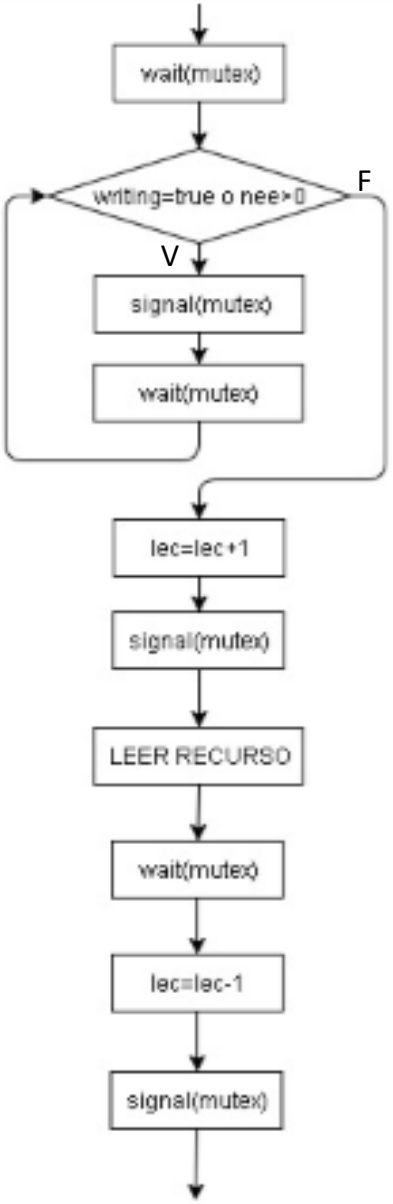
Diagrama de flujo

Mientras haya algún proceso escritor en el recurso o procesos escritores esperando por acceder al recurso

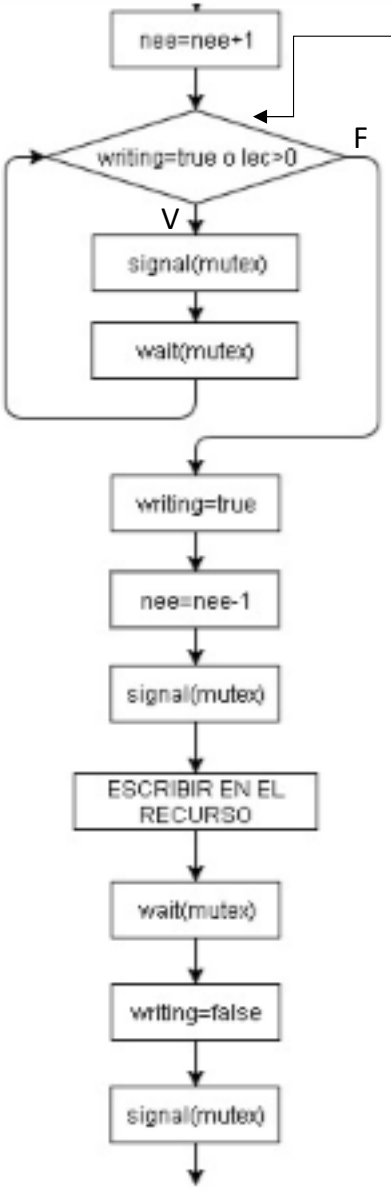
Inicializacion:

Initial(mutex,1)
writing=false
lec=0
nee=0

PROCESO LECTOR



PROCESO ESCRITOR



Mientras haya algún proceso escritor en el recurso o procesos lectores en el recurso

EJECUCIÓN

T	0	1	2	3	4	5	6
mutex	1						
writing	false						
lec	0						
nee	0						
Plector							
Pescrit							

ELABORAR DOS EJECUCIONES SUPONER Suponer 2 procesos lectores y 3 procesos escritores

Solución con Prioridad en la escritura: (sin espera ocupa)

- Se usará las siguientes variables:
  - **lec: cantidad de lectores que están accediendo al recurso.**
  - **nee: cantidad de escritores esperando para acceder al recurso.**
  - **nle: cantidad de lectores esperando para acceder al recurso**
  - **writing: variable booleana que indica si hay un escritor accediendo al recurso, (1=TRUE accediendo al recurso, 0=FALSE no está accediendo al recurso)**
  - **mutex: semáforo para asegurar la ejecución en exclusión mutua.**
  - **writer: bloquea al escritor cuando este no deba usar el recuso.**
  - **reader: bloquea al lector cuando este no deba usar el recuso**
- **Inicializacion:**
  - Initial(mutex,1)
  - Initial(writer,0)
  - Initial(reader,0)
  - writing=0
  - lec=0
  - nee=0
  - nle=0

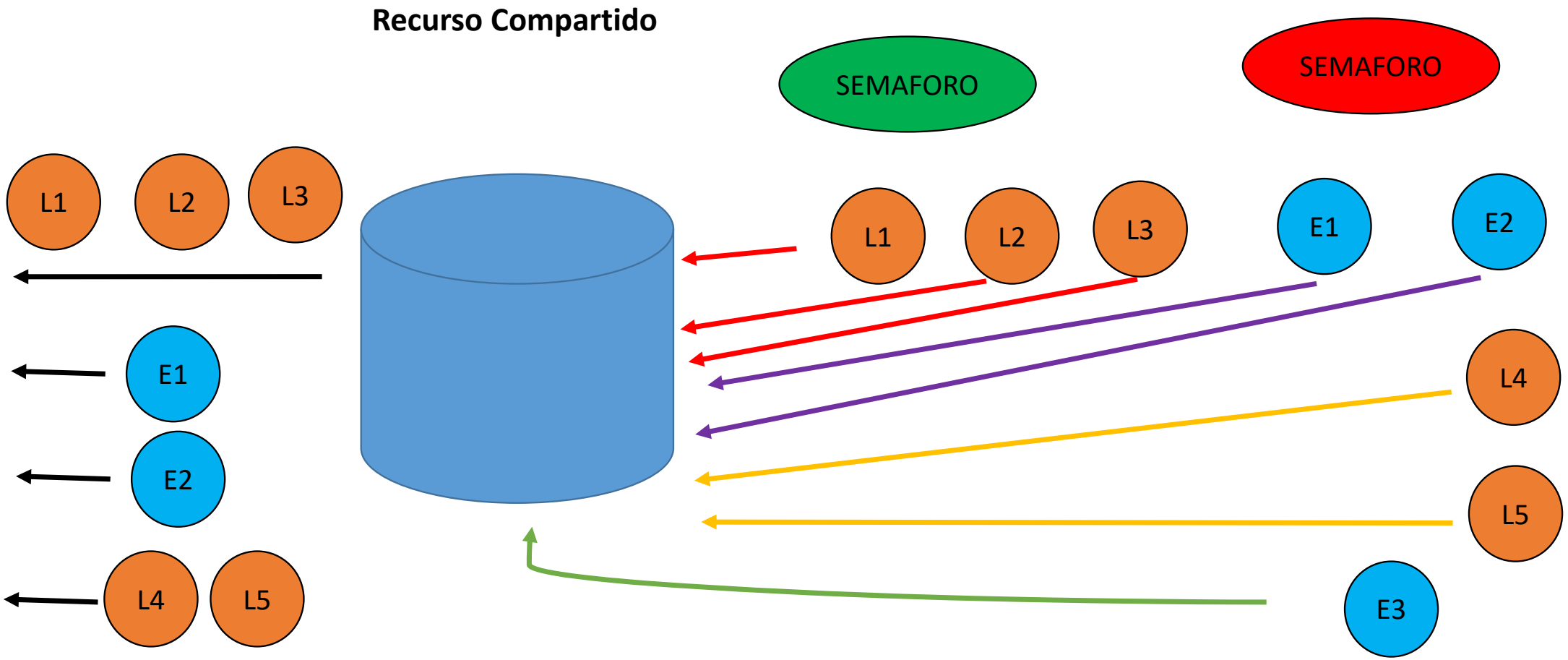
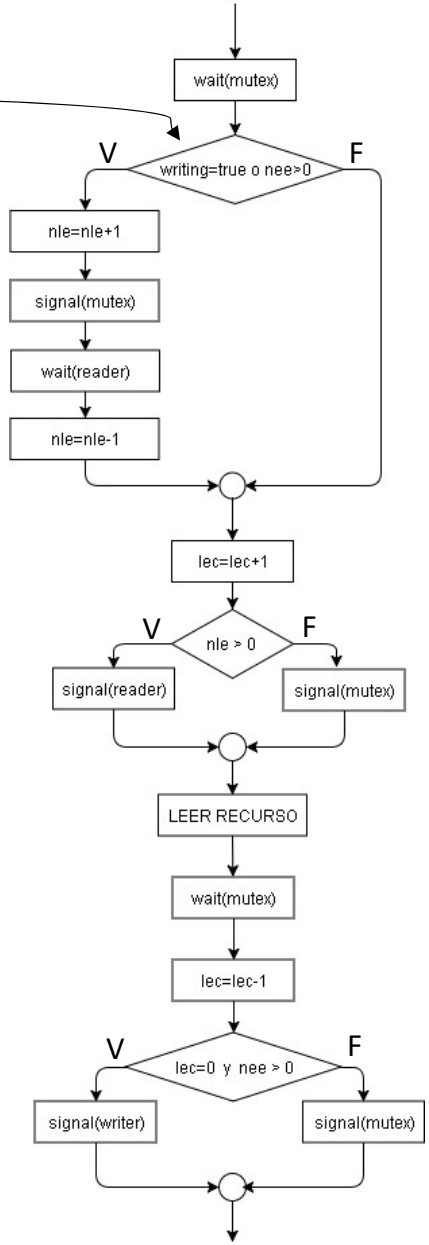

Recurso Compartido
SEMAFORO
SEMAFORO
L1
L2
L3
L1
L2
L3
E1
E2
E1
L4
E2
L5
L4
L5
E3

Diagrama de flujo

Si algún proceso escritor esta en el recurso o hay procesos escritores esperando por acceder al recurso

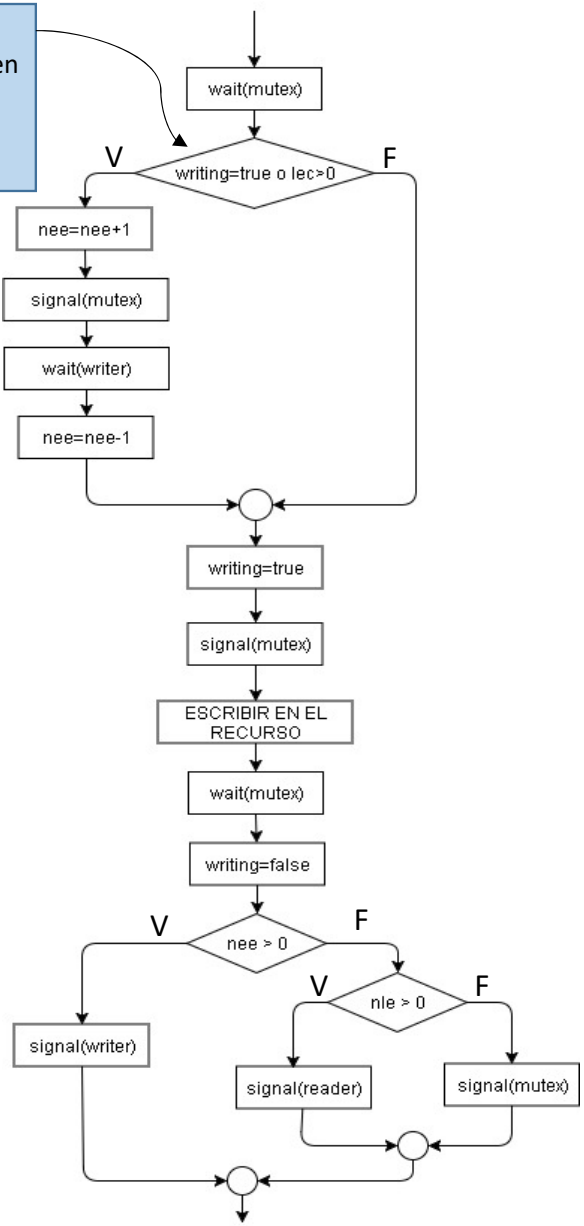
PROCESO LECTOR



Initial(mutex,1)
Initial(writer,0)
Initial(reader,0)
writing=false
lec=0
nle=0
nee=0

Si algún proceso escritor esta en el recurso o hay procesos lectores en el recurso

PROCESO ESCRITOR



EJECUCIÓN

T	1	2	3	4	5	6	7
Mutex							
writing							
nee							
nle							
lec							
reader							
writer							
Plector							
Pescritor							

ELABORAR TRES EJECUCIONES DIFERENTES

CASOS PARA LA EJECUCIÓN MANUAL CON LOS 3 ALGORITMOS

- E1,E2,L1,L2 GRUPO1
- L1,L2,L3,E1 GRUPO2
- L1,L2,E1,E2 GRUPO3
- E1,L1,L2,L3 GRUPO4
- 1L, 1E, 1L,1E GRUPO5
- 1E, 1L, 1L, 1E GRUPO6
- L1,L2,L3,L4,L5, E1 GRUPO7
- E1,E2,E3, L1,L2 GRUPO 8
- L1,E1,L2,L3,L4 GRUPO 9