LABORATORIO 09:
SEMAFOROS: PROBLEMA DE LECTORES Y ESCRITORES

Supongamos que tenemos un recurso representado por un objeto de datos que tiene que compartirse entre varios procesos concurrentes (por ejemplo, un fichero o una base de datos). Algunos de estos procesos puede que únicamente deseen leer el contenido del objeto compartido, mientras que otros pueden que deseen actualizarlo (esto es leer y escribir). Describimos entre estos dos tipos de procesos refiriéndose como lectores a aquellos que están interesados solamente en la lectura, y al resto como escritores. Obviamente, si dos lectores acceden al objeto de datos compartido simultáneamente no se producirán efectos adversos. Sim embargo, si un escritor y algún otro proceso (sea un lector o escritor) acceden al objeto compartido simultáneamente, surgirán problemas.

Para asegurarnos de que estas dificultados no aparecen, requerimos que los escritores tengan acceso exclusivo al objeto compartido. Este problema de sincronización se conoce como el problema de lectores y escritores.

Código Base Lector/Escritor:

```
program lectorescritor;
process type lector;
begin
      repeat
            (*protocolo de entrada;*)
            writeln('leer del recurso');
            (*protocolo de salida;*)
      forever
end;
process type escritor;
begin
      repeat
            (*protocolo de entrada;*)
            writeln('Escribir recurso');
            (*protocolo de salida;*)
      forever
end;
var i : integer;
      LE: array[0..3] of lector;
      ES: array[0..3] of escritor;
begin
      cobegin
            for i:=0 to 3 do
            LE[i]; ES[i];
      coend
end.
```

El problema de los lectores y escritores tiene distintas versiones, dependiendo de a qué tipo de proceso demos prioridad para acceder al recurso:

- Prioridad en la lectura: ningún lector debe esperar salvo que un escritor haya obtenido ya permiso para usar objeto compartido. Es decir, ningún lector debe esperar a que otros lectores acaben por el simple hecho de que un escritor esté esperando

- Prioridad en la escritura: una que un escritor está esperando, debe realizar su escritura tan pronto sea posible. Es decir, si un escritor está esperando, ningún lector nuevo debe iniciar su lectura.

PRIMER CASO: PRIORIDAD EN LA LECTURA

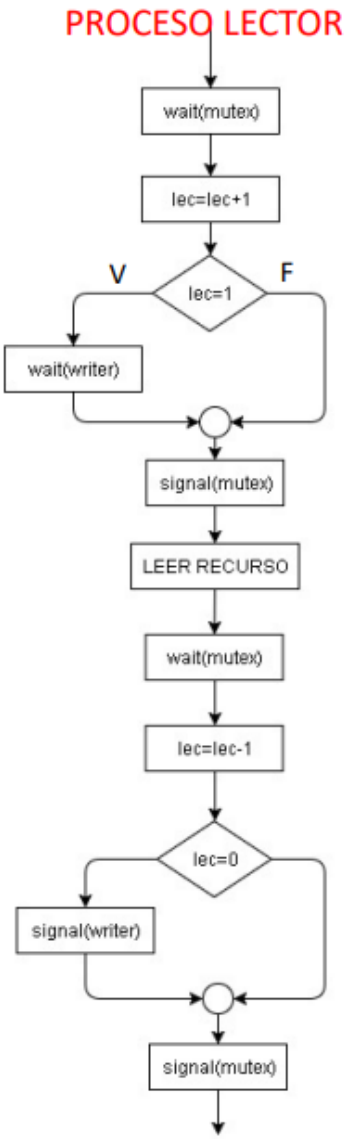
- Una variable entera n1 inicializada a 0 que indica el número de lectores que hay en el recurso compartido en un momento dado.
- Un semáforo **mutex** inicializando a 1 para asegurar la exclusión mutua cuando se actualiza en n1.
- Un semáforo **writer** inicializado a 1 y que es común a los lectores y escritores. Este semáforo funciona como semáforo de exclusión mutua para los escritores. También lo utiliza el primero /ultimo lector para entrar/salir de la sección critica. Sin embargo, no es utilizado por los lectores que entran o salen mientras otros lectores ya se encuentran en la sección critica.

Entonces:
Se usarán 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 exclusión mutua de escritores (true = escribiendo y false = no escribiendo)
Inicialización:
• Initial(mutex,1)
• Initial(writer,1)
• lec=0

DIAGRAMAS DE FLUJO

PROCESO ESCRITOR

PROCESO LECTOR



CÓDIGO EN PASCAL FC

```
var mutex , wrt : semaphore;
var lec : integer;
process type lector;
program lcpl;

begin
      repeat
            (*protocolo de entrada;*)
            wait(mutex);
            lec:=lec+1;
            if(lec=1) then
                  wait(wrt);
            signal(mutex);
            writeln('leer del recurso');
            wait(mutex);
            lec:=lec-1;
            (*protocolo de salida;*)
            if (lec=0) then
```

```
                    signal(wrt);
                signal(mutex);
        forever
end;
process type escritor;
begin
        repeat
                (*protocolo de entrada;*)
                wait(wrt);
                writeln('Escribir recurso');
                (*protocolo de salida;*)
                signal(wrt);
        forever
end;
var i : integer;
        LE: array[0..3] of lector;
        ES: array[0..3] of escritor;
begin
        initial(wrt,1);
        initial(mutex,1);
        lec:=0;
        cobegin

                for i:=0 to 3 do
                LE[i]; ES[i];
        coend
   end.
```

SEGUNDO CASO: PRIORIDAD EN LA ESCRITURA (CON ESPERA OCUPADA)

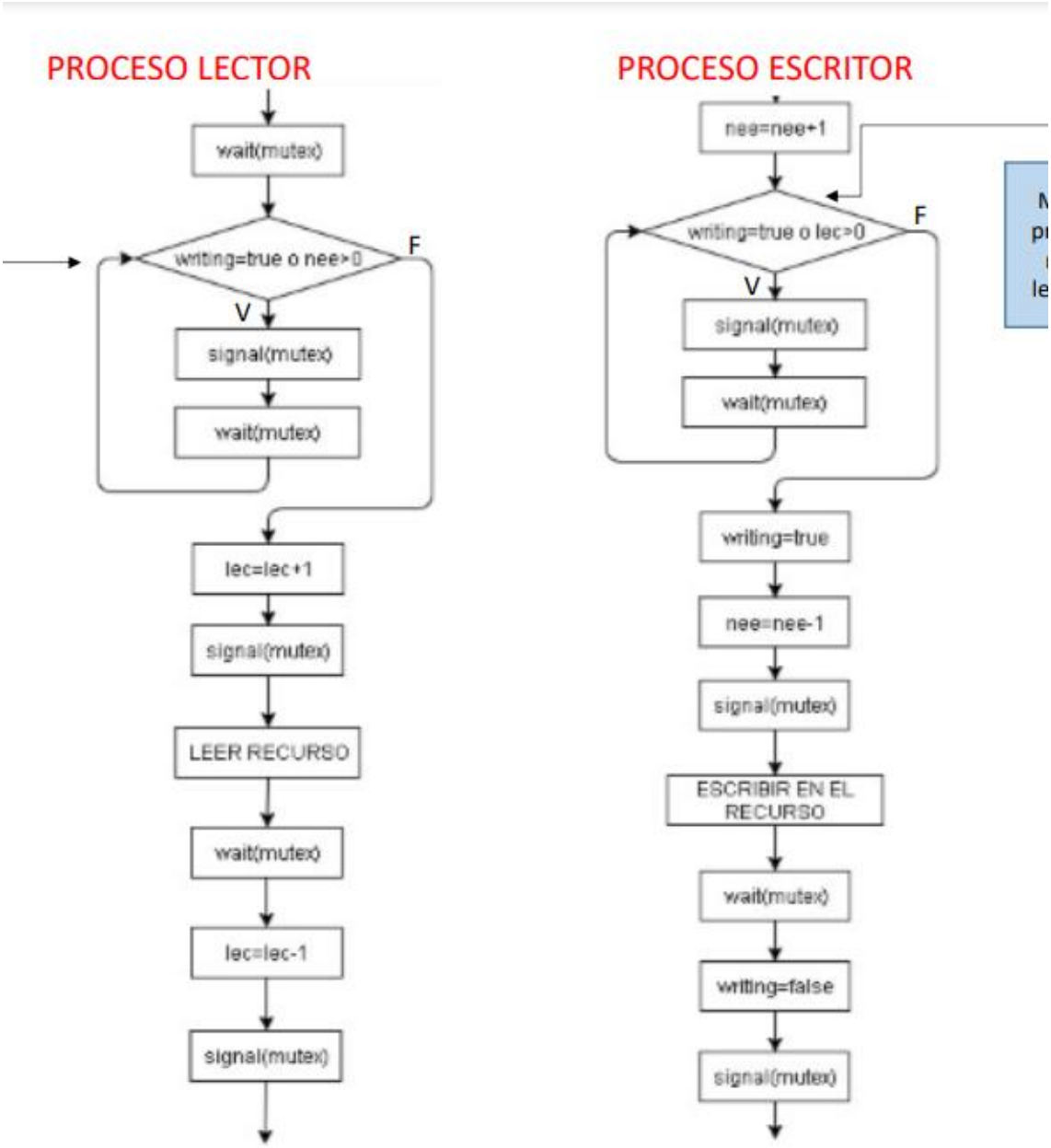
Se usará las siguientes variables:

- **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)

Inicialización:

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

DIAGRAMAS DE FLUJO



CÓDIGO EN PASCAL FC

```
program lcpese;
var mutex: semaphore;
var lec,nee : integer;
var writing : boolean;
process type lector;
begin
      repeat
            (*protocolo de entrada;*)
            wait(mutex);
```

```
            (*mientras existen escritores en espera o algún
escritor*)
            (*este escribiendo esperar*)
            while (writing = true or (nee > 0)) do
                  begin
                        signal(mutex);
                        wait(mutex);
                  end;
            lec:=lec+1;
            signal(mutex);
            writeln('leer del recurso');
            (*protocolo de salida;*)
            wait(mutex);
            lec:=lec-1;
            signal(mutex);
      forever
end;
process type escritor;
begin
      repeat
            wait(mutex);
            (*mientras algún escritor esté accediendo al
recurso*)
            (* o existan lectores leyendo hay que esperar*)
            nee:=nee+1;
            while(writing=true or (lec>0))do
                  begin
                        signal(mutex);
                        wait(mutex);
                  end;
            writing:=true;
            nee:=nee-1;
            signal(mutex);
            writeln('Escribir recurso');
            wait(mutex);
            writing:=false;
            signal(mutex);

      forever
end;
var i : integer;
      LE: array[0..3] of lector;
      ES: array[0..3] of escritor;
begin

      initial(mutex,1);
      writing:=false;
      lec:=0;
      nee:=0;
      cobegin

            for i:=0 to 3 do
            LE[i]; ES[i];
      coend
end.
```

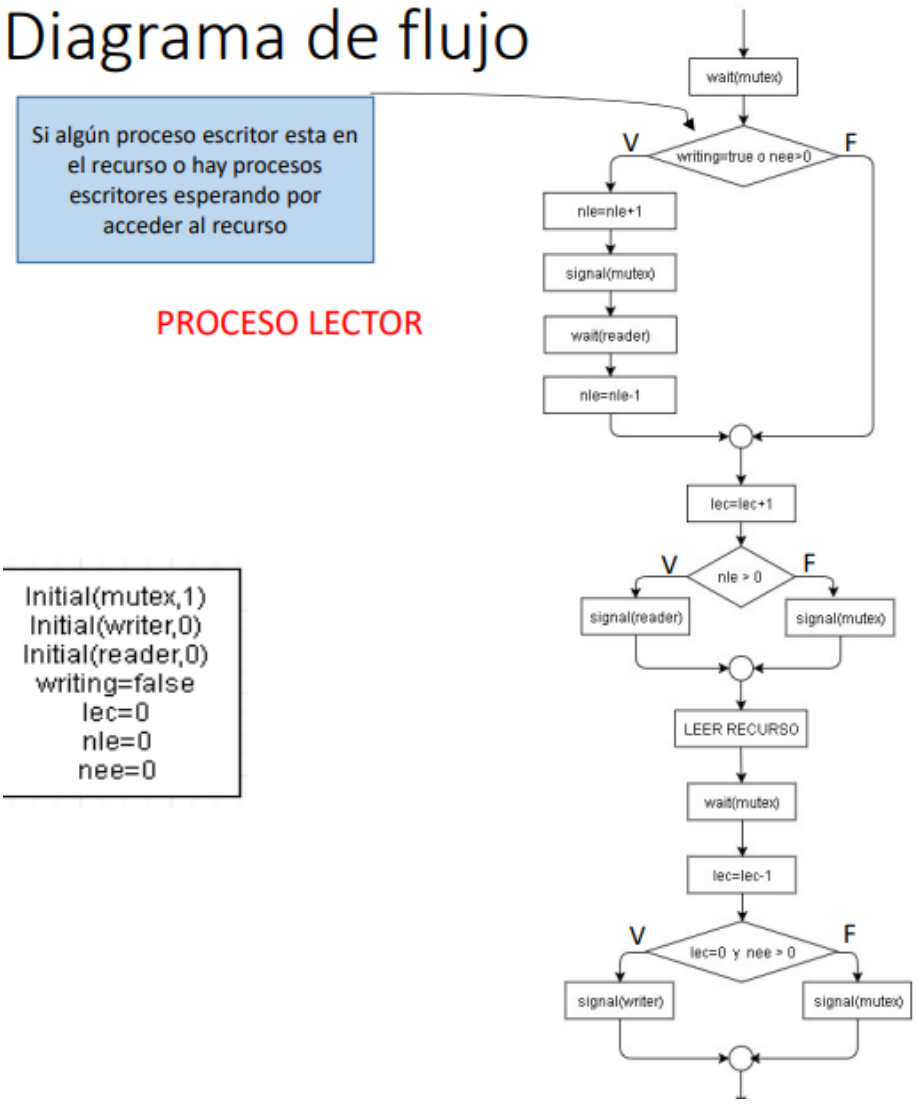
TERCER CASO: PRIORIDAD EN LA ESCRITURA (SIN ESPERA OCUPADA)

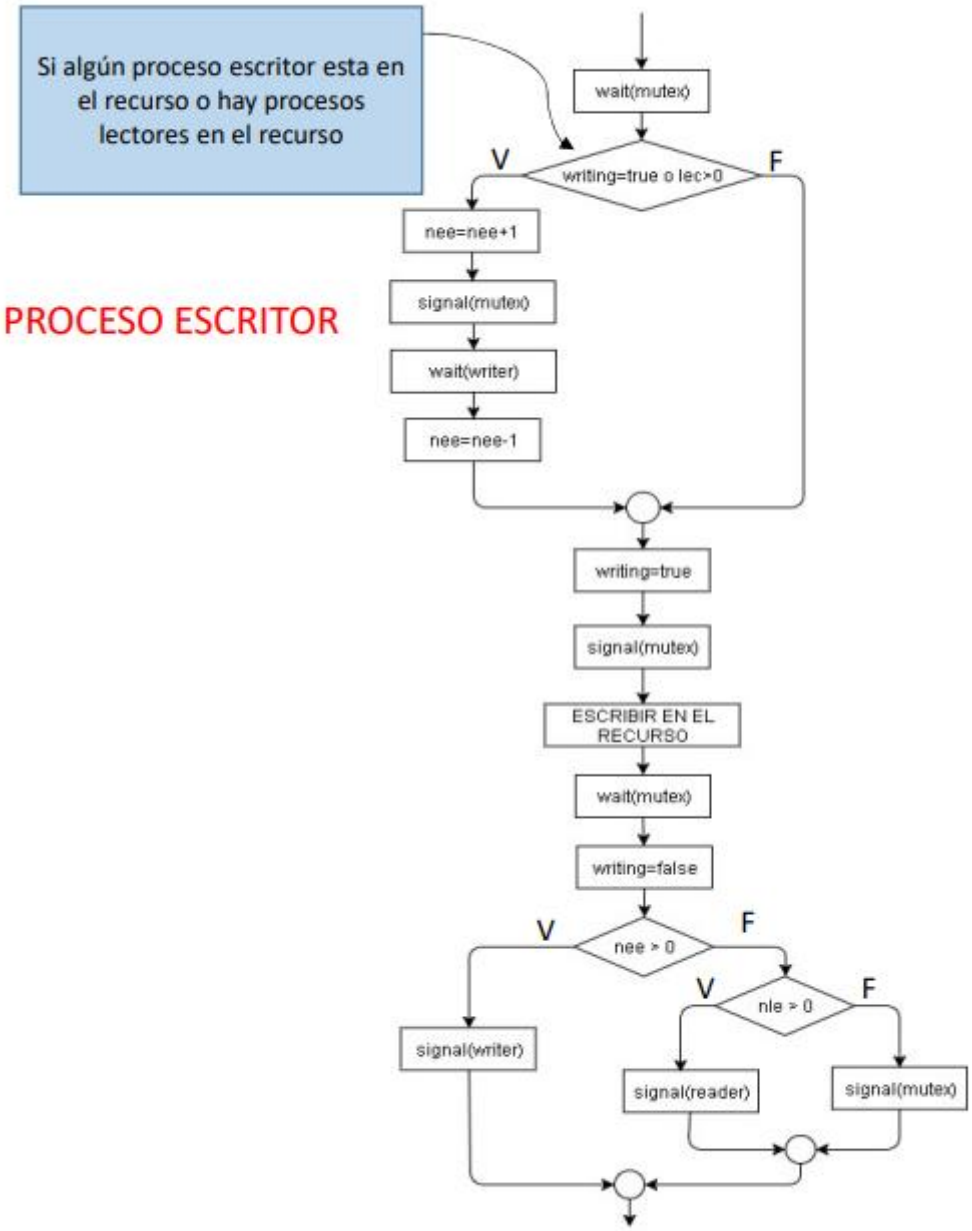
Se usará las siguientes variables:

- **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 recurso

Inicialización:

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




Si algún proceso escritor esta en el recurso o hay procesos lectores en el recurso
PROCESO ESCRITOR
wait(mutex)
writing=true o lec>0
V
F
nee=nee+1
signal(mutex)
wait(writer)
nee=nee-1
writing=true
signal(mutex)
ESCRIBIR EN EL RECURSO
wait(mutex)
writing=false
nee > 0
V
F
nle > 0
V
F
signal(writer)
signal(reader)
signal(mutex)

CÓDIGO EN PASCAL FC

```
program lcpce;
var mutex,writer,reader: semaphore;
var lec,nle,nee : integer;
var writing : boolean;
process type lector;
begin
      repeat
      (*protocolo de entrada;*)
      (* if(nle>0) then
      begin
            signal(reader);
      end
      else
      begin
      signal(mutex);
      end;*)
            wait(mutex);
            (*mientras existen escritores en espera o algún
escritor*)
            (*este escribiendo esperar*)
            if (writing = true or (nee > 0)) then
                  begin
                        nle:=nle+1;
                        signal(mutex);
                        wait(reader);
                        nle:=nle-1;
                  end;
            lec:=lec+1;
            if (nle>0) then
                  begin
                        signal(reader);
                  end
            else
                  begin
                        signal(mutex);
                  end;
            writeln('leer del recurso');
            (*protocolo de salida;*)
            wait(mutex);
            lec:=lec-1;
            if((lec=0) and (nee>0))then
                  begin
                        signal(writer);
                  end
            else
                  begin
                        signal(mutex);
                  end;

      forever
end;
process type escritor;
begin
      repeat
            wait(mutex);
            (*Si se está escribiendo o existen lectores*)
            (* el escritor debe ser bloqueado*)
            if (writing=true or (lec>0))then
```

```
                begin
                        nee:=nee+1;
                        signal(mutex);
                        wait(writer);
                        nee:=nee-1;
                end;
            writing:=true;
            signal(mutex);
            writeln('Escribir recurso');
            wait(mutex);
            (*nee:=nee-1;*)
            writing:=false;
            (*desbloquear un escritor que esté en espera*)
            (* sino desbloquear a un lector en espera*)
            if(nee>0)then
                begin
                        signal(writer)
                end
                else
                begin
                        if(nle>0) then
                                begin
                                        signal(reader);
                                end
                        else
                                begin
                                        signal(mutex);
                                end;
                end;

        forever

end;
var i : integer;
        LE: array[0..2] of lector;
        ES: array[0..2] of escritor;
begin

        initial(mutex,1);
        initial(writer,0);
        initial(reader,0);
        writing:=false;
        lec:=0;
        nle:=0;
        nee:=0;
        cobegin

                for i:=0 to 1 do
                LE[i]; ES[i];
        coend
end.
```

ACTIVIDAD PROPUESTA

- Realizar los algoritmos los tres casos del problema Lectores/Escritores de Pascal FC a C++ con el IDE Codeblocks u otro de preferencia