Semáforos (cont.)

Exclusión mútua

```
init_sem(sem_recurso, 1);
.
.
wait(sem_recurso);
    uso de recurso
signal(sem_recurso);
```

-

-

Sincronización

- El proceso B puede pasar por L2 sólo si A ya pasó por L1

Programa para A

.

.

L1: signal(continuar)

.

Programa para B

.

.

L2: wait(continuar)

.

- El semáforo se inicializa en cero.
    - waits(continuar) <= signals(continuar)
- Típico productor/consumidor

Sincronización

- El proceso B puede pasar por L2 sólo si A ya pasó por L1

Programa para A

.

.

L1: signal(continuar)

.

Programa para B

.

.

L2: wait(continuar)

.

- Ejemplo “asimétrico”: B depende de A pero no viceversa

Productor/Consumidor

- Los productores producen elementos indefinidamente
- Los consumidores los utilizan
  - Se usa un buffer con capacidad finita (N) para depositar/extraer los elementos
  - => Regulación bidireccional
- Ejemplos:
  - Servidor de impresión
  - Transferencias por red (ssh/you tube)
  - Servidor de ventanas/aplicaciones gráficas (eventos GUI)
  - Tuberías de Unix
  - Drivers (Entrada: productores /Salida: consumidores)

Productor/Consumidor

- Los productores no pueden depositar elementos si el buffer está lleno
- Los consumidores no pueden extraerlos si el buffer está vació
- Capacidad de buffer: N
- Invariante:
  - 0 <= depositados - extraídos <= N

Productor/Consumidor

- Los productores no pueden depositar elementos si el buffer está lleno
- Los consumidores no pueden extraerlos si el buffer está vació
- Capacidad de buffer: N
- Invariante:
  - 0 <= depositados - extraídos <= N

Productores/consumidores (Solución parcial)

Productores

```
repite indefinidamente
begin
   produce elemento;

   deposita item en el buffer;

end
```

Consumidores

```
repite indefinidamente
begin

   extrae item del buffer;

   consume elemento;
end
```

Productores/consumidores (Solución parcial)

Productores

```
repite indefinidamente
begin
    produce elemento;
    wait(espacio_disponible);
    deposita item en el buffer;
    signal(elemento_disponible);
end
```

Consumidores

```
repite indefinidamente
begin
    wait(elemento_disponible);
    extrae item del buffer;
    signal(espacio disponible) ;
    consume elemento;
end
```

Productores/consumidores (Solución parcial)

- Valores iniciales de los semáforos?

Productores

```
repite indefinidamente
begin
    produce elemento;
    wait(espacio_disponible);
    deposita item en el buffer;
    signal(elemento_disponible);
end
```

Consumidores

```
repite indefinidamente
begin
    wait(elemento_disponible);
    extrae item del buffer;
    signal(espacio disponible) ;
    consume elemento;
end
```

Productores/consumidores (Solución parcial)

- Valores iniciales de los semáforos?
  - Espacio_disponible := N; Elemento_disponible := 0;

Productores

```
repite indefinidamente
begin
   produce elemento;
   wait(espacio_disponible);
   deposita item en el buffer;
   signal(elemento_disponible);
end
```

Consumidores

```
repite indefinidamente
begin
   wait(elemento_disponible);
   extrae item del buffer;
   signal(espacio_disponible) ;
   consume elemento;
end
```

Productores/consumidores (Solución parcial)

- Valores iniciales de los semáforos?
  - Espacio_disponible := N; Elemento_disponible := 0;

Productores

```
repite indefinidamente
begin
   produce elemento;
   wait(espacio_disponible);
   deposita item en el buffer;
   signal(elemento_disponible);
end
```

Consumidores

```
repite indefinidamente
begin
   wait(elemento_disponible);
   extrae item del buffer;
   signal(espacio_disponible) ;
   consume elemento;
end
```

  - falta algo?

Productores/consumidores (Solución parcial)

- Valores iniciales de los semáforos?
  - Espacio_disponible := N; Elemento_disponible := 0;

Productores

```
repite indefinidamente
begin
   produce elemento;
   wait(espacio_disponible);
   deposita item en el buffer;
   signal(elemento_disponible);
end
```

Consumidores

```
repite indefinidamente
begin
   wait(elemento_disponible);
   extrae item del buffer;
   signal(espacio_disponible) ;
   consume elemento;
end
```

- falta algo?
  - Sección crítica!

Productores/consumidores (Solución genérica)

Productores

```
repite indefinidamente
begin
    produce elemento;
    wait(espacio_disponible);
    wait(buffer_manipulable);
    deposita item en el buffer;
    signal(buffer_manipulable);
    signal(elemento_disponible);
end
```

Consumidores

```
repite indefinidamente
begin
    wait(elemento_disponible);
    wait(buffer_manipulable);
    extrae item del buffer;
    signal(buffer_manipulable);
    signal(espacio_disponible) ;
    consume elemento;
end
```

Productores/consumidores (Solución genérica)

Productores

```
repite indefinidamente
begin
    produce elemento;
    wait(espacio_disponible);
    wait(buffer_manipulable);
    deposita item en el buffer;
    signal(buffer_manipulable);
    signal(elemento_disponible);
end
```

Consumidores

```
repite indefinidamente
begin
    wait(elemento_disponible);
    wait(buffer_manipulable);
    extrae item del buffer;
    signal(buffer_manipulable);
    signal(espacio_disponible) ;
    consume elemento;
end
```

Prueba de corrección de la solución

- Queremos que: 0 <= depositados - extraídos <= N

Productores

```
repite indefinidamente
begin
    produce elemento;
    wait(espacio_disponible);
    deposita item en el buffer;
    signal(elemento_disponible);
end
```

Consumidores

```
repite indefinidamente
begin
    wait(elemento_disponible);
    extrae item del buffer;
    signal(espacio_disponible) ;
    consume elemento;
end
```

- waits(espacio_disponible) <= signals(espacio_disponible)+N (1)
- waits(elemento_disponible) <= signals(elemento_disponible) (2)

Prueba de corrección de la solución

- Queremos que: 0 <= depositados - extraídos <= N

Productores

```
repite indefinidamente
begin
    produce elemento;
    wait(espacio_disponible);
    deposita item en el buffer;
    signal(elemento_disponible);
end
```

Consumidores

```
repite indefinidamente
begin
    wait(elemento_disponible);
    extrae item del buffer;
    signal(espacio_disponible) ;
    consume elemento;
end
```

- waits(espacio_disponible) <= signals(espacio_disponible)+N (1)
- waits(elemento_disponible) <= signals(elemento_disponible) (2)
- signals(elemento_disponible) <= depositados <= waits(espacio_disponible) (3)

Prueba de corrección de la solución

- Queremos que: 0 <= depositados - extraídos <= N

Productores

```
repite indefinidamente
begin
    produce elemento;
    wait(espacio_disponible);
    deposita item en el buffer;
    signal(elemento_disponible);
end
```

Consumidores

```
repite indefinidamente
begin
    wait(elemento_disponible);
    extrae item del buffer;
    signal(espacio_disponible) ;
    consume elemento;
end
```

- waits(espacio_disponible) <= signals(espacio_disponible)+N (1)
- waits(elemento_disponible) <= signals(elemento_disponible) (2)
- signals(elemento_disponible) <= depositados <= waits(espacio_disponible) (3)
- signals(espacio_disponible) <= extraídos <= waits(elemento_disponible) (4)

Prueba de corrección de la solución

Queremos que: 0 <= depositados - extraídos <= N

waits(espacio) <= signals(espacio)+N (1)
waits(elemento) <= signals(elemento) (2)
signals(elemento) <= depositados <= waits(espacio) (3)
signals(espacio) <= extraídos <= waits(elemento) (4)

depositados <= waits(espacio) [por (3)]
<= signals(espacio) + N [por (1)]
<= extraídos + N [por (4)] (5)

extraídos <= waits(elemento) [por (4)]
<= signals(elemento) [por (2)]
<= depositados [por (3)] (6)

Combinando (5) y (6):
extraídos <= depositados <= extraídos + N

=> 0 <= depositados - extraídos <= N

Interbloqueo potencial

Productores

```
repite indefinidamente
begin
    produce elemento;
    wait(espacio_disponible);
    wait(buffer_manipulable);
    deposita item en el buffer;
    signal(buffer_manipulable);
    signal(elemento_disponible);
end
```

Consumidores

```
repite indefinidamente
begin
    wait(elemento_disponible);
    wait(buffer_manipulable);
    extrae item del buffer;
    signal(buffer_manipulable);
    signal(espacio_disponible) ;
    consume elemento;
end
```

Interbloqueo potencial

Productores

```
repite indefinidamente
begin
    produce elemento;
    wait(espacio_disponible);
    wait(buffer_manipulable);
    deposita item en el buffer;
    signal(buffer_manipulable);
    signal(elemento_disponible);
  end
```

Consumidores

```
repite indefinidamente
begin
    wait(elemento_disponible);
    wait(buffer_manipulable);
    extrae item del buffer;
    signal(buffer_manipulable);
    signal(espacio_disponible) ;
    consume elemento;
  end
```

Productores2

```
repite indefinidamente
begin
    produce elemento;
    wait(buffer_manipulable);
    wait(espacio_disponible);
    deposita item en el buffer;
    signal(elemento_disponible);
    signal(buffer_manipulable);
end
```

Consumidores2

```
repite indefinidamente
begin
    wait(buffer_manipulable);
    wait(elemento_disponible);
    extrae item del buffer;
    signal(espacio_disponible) ;
    signal(buffer_manipulable);
    consume elemento;
end
```

Interbloqueo potencial

Productores2

```
repite indefinidamente
begin
    produce elemento;
    wait(buffer_manipulable);
    wait(espacio_disponible);
    deposita item en el buffer;
    signal(elemento_disponible);
    signal(buffer_manipulable);
end
```

Consumidores2

```
repite indefinidamente
begin
    wait(buffer_manipulable);
    wait(elemento_disponible);
    extrae item del buffer;
    signal(espacio_disponible) ;
    signal(buffer_manipulable);
    consume elemento;
end
```

Interbloqueo potencial

Productores2

```
repite indefinidamente
begin
    produce elemento;
    wait(buffer_manipulable);
    wait(espacio_disponible);
    deposita item en el buffer;
    signal(elemento_disponible);
    signal(buffer_manipulable);
end
```

Consumidores2

```
repite indefinidamente
begin
    wait(buffer_manipulable);
    wait(elemento_disponible);
    extrae item del buffer;
    signal(espacio_disponible) ;
    signal(buffer_manipulable);
    consume elemento;
end
```

Prueba de no interbloqueo

Productores

```
repite indefinidamente
begin
    wait(espacio);
    produce item;
    signal(item);
end
```

Consumidores

```
repite indefinidamente
begin
    wait(item);
    extrae item;
    signal(espacio) ;
end
```

- Para que ocurra interbloqueo:
  - (1) Productores bloqueados por wait(espacio) y consumidores no pueden hacer signal(espacio)
  - (2) Consumidores bloqueados por wait(item) y productores no pueden hacer signal(item)

Prueba de no interbloqueo

- Para que ocurra interbloqueo:
  - (1) Productores bloqueados por wait(espacio) y consumidores no pueden hacer signal(espacio)
  - (2) Consumidores bloqueados por wait(item) y productores no pueden hacer signal(item)

de (1): waits(espacio) = signals(espacio) + N ,y
waits(item) = signals(espacio)

de (2): waits(item) = signals(item) , y
waits(espacio) = signals(item)

=> waits(espacio) = signals(espacio)+ N, y
waits(espacio) = signals(espacio)
=> N = 0 (!)

Productores

```
repite indefinidamente
begin
      wait(espacio);
      deposita item;
      signal(item);
end
```

Consumidores

```
repite indefinidamente
begin
      wait(item);
      extrae item;
      signal(espacio) ;
end
```