

Lección 5: Sincronización de procesos mediante semáforos

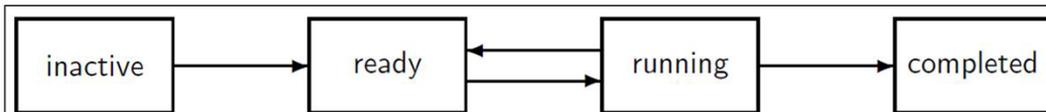
- Introducción
- Estados de un proceso
- Semáforos: sintaxis y semántica
- Técnicas de programación con semáforos:
 - el problema de la sección crítica
 - el problema de la barrera
 - el caso de productores/consumidores
- Algunos ejercicios sencillos

Introducción

- La sincronización por espera activa tiene inconvenientes:
 - los algoritmos son difíciles de diseñar y verificar
 - se mezclan variables del problema a resolver y del método de resolución
 - en los casos de multi-programación, se ocupa innecesariamente el procesador "haciendo nada"
- Los semáforos son una de las primeras soluciones
 - son una solución conceptual, que puede implementarse de diferentes formas
 - propuestos por Dijkstra en 1968

Estados de un proceso

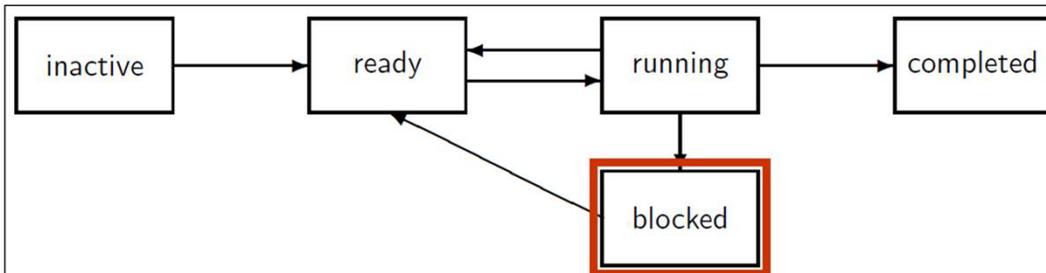
- Para multitarea, y un proceso P, **P.state** es su estado



M. Ben-Ari. Principles of Concurrent and Distributed Programming, Second edition © M. Ben-Ari 2006

Estados de un proceso

- Para multitarea, y un proceso P, **P.state** es su estado



M. Ben-Ari. Principles of Concurrent and Distributed Programming, Second edition © M. Ben-Ari 2006

Semáforos: sintaxis y semántica

- **Semáforo**: instancia de un tipo abstracto de dato
 - **S = (k, L)**
- tal que ...
 - k es un natural
 - L es un conjunto de procesos
- con dos instrucciones atómicas...
 - **wait**
 - **signal**

Semáforos: sintaxis y semántica

- Declaración `semaphore S := (S0, {})`

- Semántica de las operaciones (atómicas) cuando la invoca P:

```
wait(S)
if S.V > 0
  S.V := S.V - 1
else
  S.L := S.L + {P}
  P.state := blocked
```

Semáforos: sintaxis y semántica

- Semántica de las operaciones (atómicas) cuando la invoca P:

```
signal(S)
if S.L = {}
    S.V := S.V + 1
else
    S.L := S.L - {Q}
    Q.state := ready
```

- Propiedad: un semáforo S cumple el siguiente invariante

$$S.V = S_0 + \#signal(S) - \#wait(S)$$

El problema de la sección crítica

- Ejemplo: la sección crítica

semaphore mutex := (1, {})	
<i>Process P</i>	<i>Process Q</i>
loop forever	loop forever
SNC	SNC
p1 <i>wait(mutex)</i>	<i>wait(mutex)</i> q1
SC	SC
p2 <i>signal(mutex)</i>	<i>signal(mutex)</i> q2

Semáforos: sintaxis y semántica

- Semántica alternativa para semáforos: versión “*busy-wait*”

```
semaphore S := (S0, {})
```

```
await S.V > 0 wait(S)  
S.V := S.V - 1
```

```
S.V := S.V + 1 signal(S)
```

Semáforos: sintaxis y semántica

- En realidad, S.L no hace falta, y podemos identificar S con S.V

```
semaphore S := S0
```

```
await S > 0  
  S := S - 1
```

wait(S)

```
S := S + 1
```

signal(S)

Semáforos: sintaxis y semántica

- ¿Cuál es el matiz entre ambas versiones?

```
if S.L = {}
  S.V := S.V + 1
else
  S.L := S.L - {Q}
  Q.state := ready
```

signal(S)

```
S := S + 1
```

signal(S)

Semáforos: sintaxis y semántica

- Existe un caso especial: los semáforos binarios

- Declaración

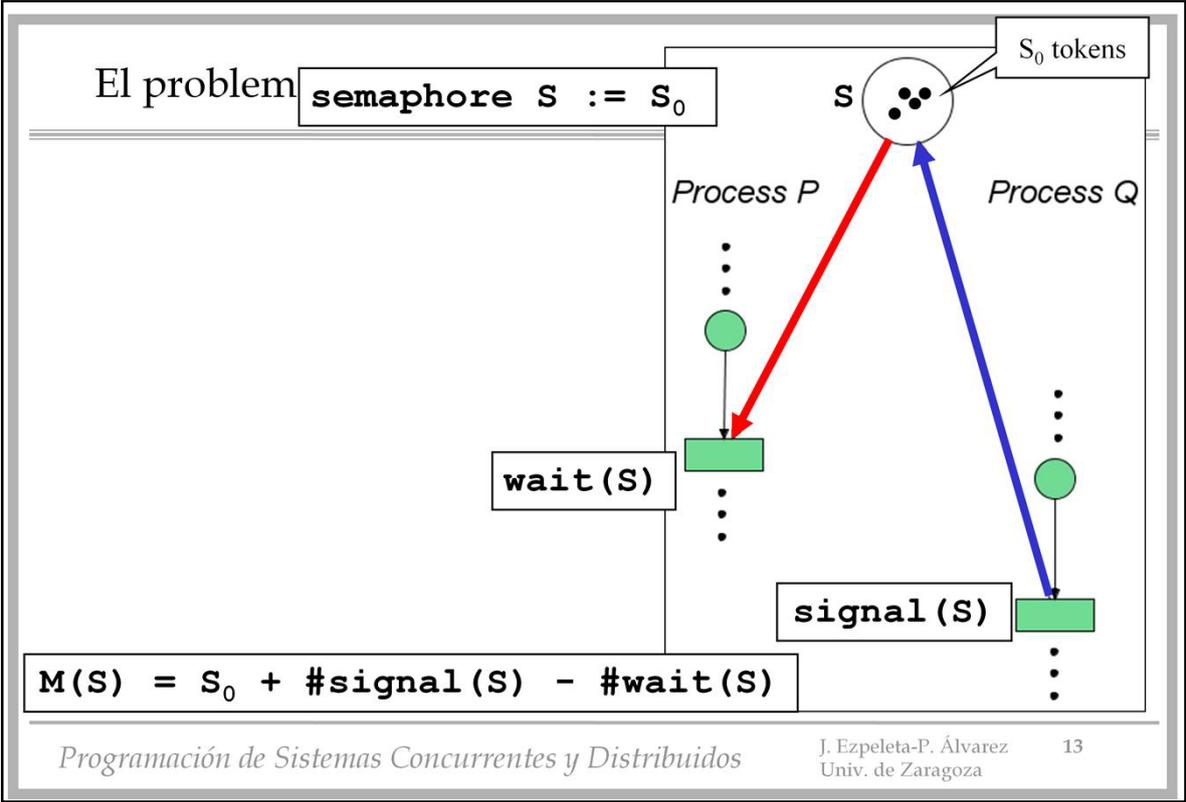
```
binary semaphore S := S0
```

- con $S_0: 0..1$

- Semántica de las operaciones (atómicas) cuando la invoca P:

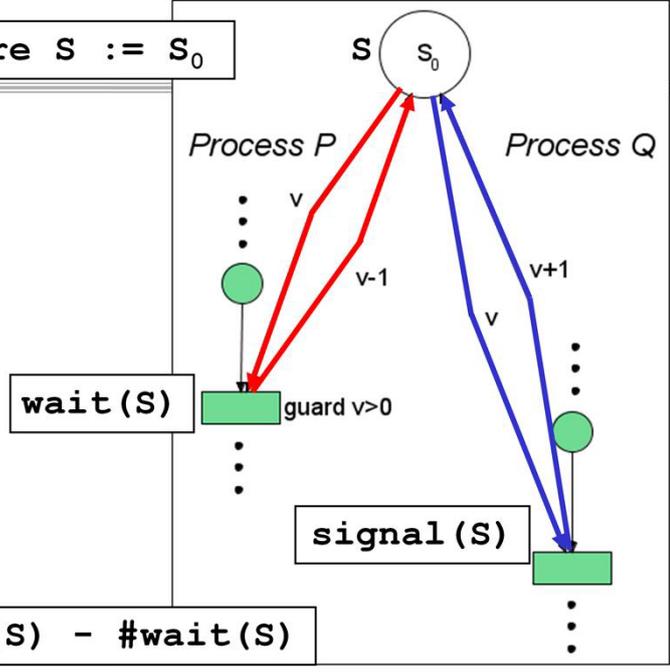
```
wait(S)  
await S = 1  
S := 0
```

```
signal(S)  
await S = 0  
S := 1
```



El problema

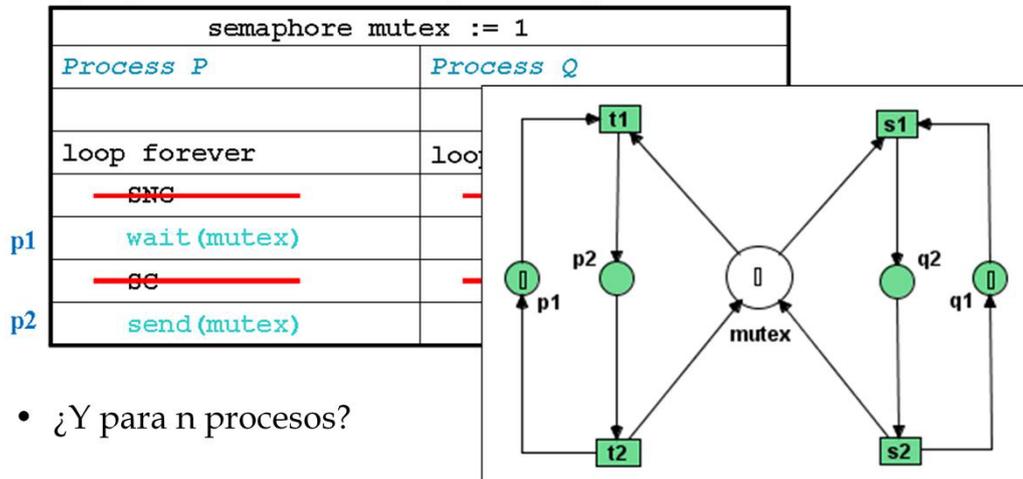
`semaphore S := S0`



$$M(S) = S_0 + \#signal(S) - \#wait(S)$$

El problema de la sección crítica

- Ejemplo: la sección crítica para dos procesos



- ¿Y para n procesos?

El problema de la barrera

- **Ejemplo:** algoritmo para dos procesos iterativos que **sincronizan** cada iteración mediante una **barrera**

????	
<i>Process P</i>	<i>Process Q</i>
loop forever	loop forever
cálculos locales	cálculos locales
espera a Q	espera a P

Productores/consumidores

- **Ejemplo:**

- tenemos un sistema con un proceso productor y un consumidor
- y un buffer intermedio de **capacidad infinita**

queue el_tipo buffer	
...	
<i>Process productor</i>	<i>Process consumidor</i>
el_tipo d	el_tipo d
loop forever	loop forever
produce(d)	consume(d,buffer)
append(d,buffer)	usa(d)

Productores/consumidores

- **Ejercicio 1:** mismo enunciado, pero con un buffer de capacidad 1
- **Ejercicio 2:** mismo enunciado, pero con un buffer de capacidad N
- **Ejercicio 3:** dar un esquema de solución al problema de la barrera para n procesos

Algunos ejercicios sencillos

- **Ejercicio 4:** completar los procesos siguientes de manera que "d" no se puede ejecutar hasta que "a" haya acabado
- **Ejercicio 5:** completar los procesos siguientes de manera que P2 puede ejecutar "d" si se ha ejecutado "e" ó "a"

<i>Process P1</i>	<i>Process P2</i>
a	c
b	d

<i>Process P1</i>	<i>Process P2</i>	<i>Process P3</i>
loop forever	loop forever	loop forever
a	c	e
b	d	f

Considérese el siguiente programa concurrente:

```
-----  
                semaphore r1 := 1,r2 := 1  
  
    process UNO::                                process DOS::  
        loop forever                            loop forever  
p0:      wait(r1)                               wait(r2)      :q0  
p1:      wait(r2)                               wait(r1)      :q1  
p2:      signal(r1)                            signal(r2)     :q2  
p3:      signal(r2)                            signal(r1)     :q3  
        end loop                                end loop  
    end process                                end process  
-----
```

El ejercicio pide:

1. Dar un modelo red de Petri correspondiente al programa
2. A partir del modelo anterior, construir su espacio de estados

Considérese el siguiente programa concurrente:

```
-----  
                semaphore r1 := 1, r2 := 1  
  
    process UNO::                                process DOS::  
        loop forever                            loop forever  
p0:      wait(r1)                               wait(r2)      :q0  
p1:      wait(r2)                               wait(r1)      :q1  
p2:      signal(r1)                            signal(r2)    :q2  
p3:      signal(r2)                            signal(r1)    :q3  
        end loop                                end loop  
    end process                                end process  
-----
```

- a) ¿Puede el programa llegar a bloquearse? En caso afirmativo, decir cuántas historias acaban en bloqueo, y dar una de ellas.
- b) ¿Existe alguna historia no equitativa (entendiendo que es una historia infinita en que a partir de un momento determinado algún proceso no interviene)?