Concurrencia
Conceptos y notación

Salvador Tamarit (UPM)

Grado en Ingeniería Informática
Grado en Matemáticas e Informática
Universidad Politécnica de Madrid

2014-2015

Concurrencia

Concurrencia =

Concurrencia

Concurrencia = Ejecución simultánea

Concurrencia

Concurrencia = Ejecución simultánea & Indeterminismo

Concurrencia

Concurrencia = Ejecución simultánea & Indeterminismo & Interacción

Interacción =

Concurrencia

Concurrencia = Ejecución simultánea & Indeterminismo & Interacción

Interacción = Comunicación

Concurrencia

Concurrencia = Ejecución simultánea & Indeterminismo & Interacción

Interacción = Comunicación & Sincronización

Sincronización =

Concurrencia

Concurrencia = Ejecución simultánea & Indeterminismo & Interacción

Interacción = Comunicación & Sincronización

Sincronización = Exclusión mutua

Concurrencia

Concurrencia = Ejecución simultánea & Indeterminismo & Interacción

Interacción = Comunicación & Sincronización

Sincronización = Exclusión mutua | Sincronización por condición

Proceso y Programa concurrente

- Programa secuencial: Especificación de la ejecución secuencial de una lista de instrucciones.

Proceso y Programa concurrente

- Programa secuencial: Especificación de la ejecución secuencial de una lista de instrucciones.
- Proceso: Ejecución de un programa secuencial.

Proceso y Programa concurrente

- Programa secuencial: Especificación de la ejecución secuencial de una lista de instrucciones.
- Proceso: Ejecución de un programa secuencial.
- Programa concurrente: Especificación de dos o más programas que se pueden ejecutar concurrentemente como procesos paralelos.

Interacción entre procesos

Interacción = Comunicación & Sincronización

- Comunicación: Permite compartir/enviar datos entre procesos.
  - Variables compartidas: Hilos
  - Paso de mensajes: Procesos e hilos
- Sincronización
  - Enfoque operacional:
    - Conjunto de restricciones en el orden de los eventos.
    - El programador emplea un mecanismo de sincronización para retrasar la ejecución de un proceso con el fin de satisfacer dichas restricciones.
  - Enfoque axiomático: La semántica de las instrucciones esta definida por axiomas y reglas de inferencia. (Lógica de la programación)

Sincronización (Enfoque operacional)

a = 0; $\quad$ P $\rightarrow$ b = a; c = b + 1; a = c; $\quad$ Q $\rightarrow$ d = a + 2; a = d;

Sincronización (Enfoque operacional)

a = 0; P $\rightarrow$ b = a; c = b + 1; a = c; Q $\rightarrow$ d = a + 2; a = d;

P $\rightarrow p_1; p_2; p_3;$ Q $\rightarrow q_1; q_2;$

Sincronización (Enfoque operacional)

a = 0; P $\rightarrow$ b = a; c = b + 1; a = c; Q $\rightarrow$ d = a + 2; a = d;

P $\rightarrow p_1; p_2; p_3;$ Q $\rightarrow q_1; q_2;$

entrelazados

$p_1; p_2; p_3; q_1; q_2;$
$p_1; p_2; q_1; p_3; q_2;$
$p_1; p_2; q_1; q_2; p_3;$
$p_1; q_1; p_2; p_3; q_2;$
$p_1; q_1; p_2; q_2; p_3;$
$p_1; q_1; q_2; p_2; p_3;$
$q_1; p_1; p_2; p_3; q_2;$
$q_1; p_1; p_2; q_2; p_3;$
$q_1; p_1; q_2; p_2; p_3;$
$q_1; q_2; p_1; p_2; p_3;$

Sincronización (Enfoque operacional)

a = 0; P $\rightarrow$ b = a; c = b + 1; a = c; Q $\rightarrow$ d = a + 2; a = d;

P $\rightarrow p_1; p_2; p_3;$ Q $\rightarrow q_1; q_2;$

entrelazados

$p_1; p_2; p_3; q_1; q_2;$ a=3
$p_1; p_2; q_1; p_3; q_2;$
$p_1; p_2; q_1; q_2; p_3;$
$p_1; q_1; p_2; p_3; q_2;$
$p_1; q_1; p_2; q_2; p_3;$
$p_1; q_1; q_2; p_2; p_3;$
$q_1; p_1; p_2; p_3; q_2;$
$q_1; p_1; p_2; q_2; p_3;$
$q_1; p_1; q_2; p_2; p_3;$
$q_1; q_2; p_1; p_2; p_3;$

Sincronización (Enfoque operacional)

a = 0; P $\rightarrow$ b = a; c = b + 1; a = c; Q $\rightarrow$ d = a + 2; a = d;

P $\rightarrow p_1; p_2; p_3;$ Q $\rightarrow q_1; q_2;$

entrelazados

$p_1; p_2; p_3; q_1; q_2;$ a=3
$p_1; p_2; q_1; p_3; q_2;$ a=2
$p_1; p_2; q_1; q_2; p_3;$
$p_1; q_1; p_2; p_3; q_2;$
$p_1; q_1; p_2; q_2; p_3;$
$p_1; q_1; q_2; p_2; p_3;$
$q_1; p_1; p_2; p_3; q_2;$
$q_1; p_1; p_2; q_2; p_3;$
$q_1; p_1; q_2; p_2; p_3;$
$q_1; q_2; p_1; p_2; p_3;$

Sincronización (Enfoque operacional)

a = 0; P $\rightarrow$ b = a; c = b + 1; a = c; Q $\rightarrow$ d = a + 2; a = d;

P $\rightarrow p_1; p_2; p_3;$ Q $\rightarrow q_1; q_2;$

entrelazados

$p_1; p_2; p_3; q_1; q_2;$	a=3
$p_1; p_2; q_1; p_3; q_2;$	a=2
$p_1; p_2; q_1; q_2; p_3;$	a=1
$p_1; q_1; p_2; p_3; q_2;$	
$p_1; q_1; p_2; q_2; p_3;$	
$p_1; q_1; q_2; p_2; p_3;$	
$q_1; p_1; p_2; p_3; q_2;$	
$q_1; p_1; p_2; q_2; p_3;$	
$q_1; p_1; q_2; p_2; p_3;$	
$q_1; q_2; p_1; p_2; p_3;$	

Sincronización (Enfoque operacional)

a = 0; P $\rightarrow$ b = a; c = b + 1; a = c; Q $\rightarrow$ d = a + 2; a = d;

P $\rightarrow p_1; p_2; p_3;$ Q $\rightarrow q_1; q_2;$

entrelazados

$p_1; p_2; p_3; q_1; q_2;$	a=3
$p_1; p_2; q_1; p_3; q_2;$	a=2
$p_1; p_2; q_1; q_2; p_3;$	a=1
$p_1; q_1; p_2; p_3; q_2;$	a=2
$p_1; q_1; p_2; q_2; p_3;$	
$p_1; q_1; q_2; p_2; p_3;$	
$q_1; p_1; p_2; p_3; q_2;$	
$q_1; p_1; p_2; q_2; p_3;$	
$q_1; p_1; q_2; p_2; p_3;$	
$q_1; q_2; p_1; p_2; p_3;$	

Sincronización (Enfoque operacional)

a = 0; P $\rightarrow$ b = a; c = b + 1; a = c; Q $\rightarrow$ d = a + 2; a = d;

P $\rightarrow p_1; p_2; p_3;$ Q $\rightarrow q_1; q_2;$

entrelazados

$p_1; p_2; p_3; q_1; q_2;$	a=3
$p_1; p_2; q_1; p_3; q_2;$	a=2
$p_1; p_2; q_1; q_2; p_3;$	a=1
$p_1; q_1; p_2; p_3; q_2;$	a=2
$p_1; q_1; p_2; q_2; p_3;$	a=1
$p_1; q_1; q_2; p_2; p_3;$	
$q_1; p_1; p_2; p_3; q_2;$	
$q_1; p_1; p_2; q_2; p_3;$	
$q_1; p_1; q_2; p_2; p_3;$	
$q_1; q_2; p_1; p_2; p_3;$	

Sincronización (Enfoque operacional)

a = 0; P $\rightarrow$ b = a; c = b + 1; a = c; Q $\rightarrow$ d = a + 2; a = d;

P $\rightarrow p_1; p_2; p_3;$ Q $\rightarrow q_1; q_2;$

entrelazados

$p_1; p_2; p_3; q_1; q_2;$	a=3
$p_1; p_2; q_1; p_3; q_2;$	a=2
$p_1; p_2; q_1; q_2; p_3;$	a=1
$p_1; q_1; p_2; p_3; q_2;$	a=2
$p_1; q_1; p_2; q_2; p_3;$	a=1
$p_1; q_1; q_2; p_2; p_3;$	a=1
$q_1; p_1; p_2; p_3; q_2;$	
$q_1; p_1; p_2; q_2; p_3;$	
$q_1; p_1; q_2; p_2; p_3;$	
$q_1; q_2; p_1; p_2; p_3;$	

Sincronización (Enfoque operacional)

a = 0; P $\rightarrow$ b = a; c = b + 1; a = c; Q $\rightarrow$ d = a + 2; a = d;

P $\rightarrow p_1; p_2; p_3;$ Q $\rightarrow q_1; q_2;$

entrelazados

$p_1; p_2; p_3; q_1; q_2;$	a=3
$p_1; p_2; q_1; p_3; q_2;$	a=2
$p_1; p_2; q_1; q_2; p_3;$	a=1
$p_1; q_1; p_2; p_3; q_2;$	a=2
$p_1; q_1; p_2; q_2; p_3;$	a=1
$p_1; q_1; q_2; p_2; p_3;$	a=1
$q_1; p_1; p_2; p_3; q_2;$	a=2
$q_1; p_1; p_2; q_2; p_3;$	
$q_1; p_1; q_2; p_2; p_3;$	
$q_1; q_2; p_1; p_2; p_3;$	

Sincronización (Enfoque operacional)

a = 0; P $\rightarrow$ b = a; c = b + 1; a = c; Q $\rightarrow$ d = a + 2; a = d;

P $\rightarrow p_1; p_2; p_3;$ Q $\rightarrow q_1; q_2;$

entrelazados

$p_1; p_2; p_3; q_1; q_2;$	a=3
$p_1; p_2; q_1; p_3; q_2;$	a=2
$p_1; p_2; q_1; q_2; p_3;$	a=1
$p_1; q_1; p_2; p_3; q_2;$	a=2
$p_1; q_1; p_2; q_2; p_3;$	a=1
$p_1; q_1; q_2; p_2; p_3;$	a=1
$q_1; p_1; p_2; p_3; q_2;$	a=2
$q_1; p_1; p_2; q_2; p_3;$	a=1
$q_1; p_1; q_2; p_2; p_3;$	
$q_1; q_2; p_1; p_2; p_3;$	

Sincronización (Enfoque operacional)

a = 0; P $\rightarrow$ b = a; c = b + 1; a = c; Q $\rightarrow$ d = a + 2; a = d;

P $\rightarrow p_1; p_2; p_3;$ Q $\rightarrow q_1; q_2;$

entrelazados

$p_1; p_2; p_3; q_1; q_2;$	a=3
$p_1; p_2; q_1; p_3; q_2;$	a=2
$p_1; p_2; q_1; q_2; p_3;$	a=1
$p_1; q_1; p_2; p_3; q_2;$	a=2
$p_1; q_1; p_2; q_2; p_3;$	a=1
$p_1; q_1; q_2; p_2; p_3;$	a=1
$q_1; p_1; p_2; p_3; q_2;$	a=2
$q_1; p_1; p_2; q_2; p_3;$	a=1
$q_1; p_1; q_2; p_2; p_3;$	a=1
$q_1; q_2; p_1; p_2; p_3;$	

Sincronización (Enfoque operacional)

a = 0; P $\rightarrow$ b = a; c = b + 1; a = c; Q $\rightarrow$ d = a + 2; a = d;

P $\rightarrow p_1; p_2; p_3;$ Q $\rightarrow q_1; q_2;$

entrelazados

$p_1; p_2; p_3; q_1; q_2;$	a=3
$p_1; p_2; q_1; p_3; q_2;$	a=2
$p_1; p_2; q_1; q_2; p_3;$	a=1
$p_1; q_1; p_2; p_3; q_2;$	a=2
$p_1; q_1; p_2; q_2; p_3;$	a=1
$p_1; q_1; q_2; p_2; p_3;$	a=1
$q_1; p_1; p_2; p_3; q_2;$	a=2
$q_1; p_1; p_2; q_2; p_3;$	a=1
$q_1; p_1; q_2; p_2; p_3;$	a=1
$q_1; q_2; p_1; p_2; p_3;$	a=3

Sincronización (Enfoque operacional)

a = 0; P $\rightarrow$ b = a; c = b + 1; a = c; Q $\rightarrow$ d = a + 2; a = d;

P $\rightarrow p_1; p_2; p_3;$ Q $\rightarrow q_1; q_2;$

entrelazados

$p_1; p_2; p_3; q_1; q_2;$	a=3 ✓
$p_1; p_2; q_1; p_3; q_2;$	a=2 ×
$p_1; p_2; q_1; q_2; p_3;$	a=1 ×
$p_1; q_1; p_2; p_3; q_2;$	a=2 ×
$p_1; q_1; p_2; q_2; p_3;$	a=1 ×
$p_1; q_1; q_2; p_2; p_3;$	a=1 ×
$q_1; p_1; p_2; p_3; q_2;$	a=2 ×
$q_1; p_1; p_2; q_2; p_3;$	a=1 ×
$q_1; p_1; q_2; p_2; p_3;$	a=1 ×
$q_1; q_2; p_1; p_2; p_3;$	a=3 ✓

Sincronización (Enfoque operacional)

a = 0; P $\rightarrow$ b = a; c = b + 1; a = c; Q $\rightarrow$ d = a + 2; a = d;

P $\rightarrow p_1; p_2; p_3;$ Q $\rightarrow q_1; q_2;$

entrelazados

$p_1; p_2; p_3; q_1; q_2;$	a=3 ✓
$p_1; p_2; q_1; p_3; q_2;$	a=2 ×
$p_1; p_2; q_1; q_2; p_3;$	a=1 ×
$p_1; q_1; p_2; p_3; q_2;$	a=2 ×
$p_1; q_1; p_2; q_2; p_3;$	a=1 ×
$p_1; q_1; q_2; p_2; p_3;$	a=1 ×
$q_1; p_1; p_2; p_3; q_2;$	a=2 ×
$q_1; p_1; p_2; q_2; p_3;$	a=1 ×
$q_1; p_1; q_2; p_2; p_3;$	a=1 ×
$q_1; q_2; p_1; p_2; p_3;$	a=3 ✓

P; Q; or Q; P;

Sincronización (Enfoque operacional)

a = 0; $\quad$ P $\rightarrow$ b = a; c = b + 1; a = c; $\quad$ Q $\rightarrow$ a = a + 2; d = 3;

Sincronización (Enfoque operacional)

a = 0; P $\rightarrow$ b = a; c = b + 1; a = c; Q $\rightarrow$ a = a + 2; d = 3;

P $\rightarrow p_1; p_2; p_3;$ Q $\rightarrow q_1; q_2;$

Sincronización (Enfoque operacional)

a = 0; P $\rightarrow$ b = a; c = b + 1; a = c; Q $\rightarrow$ a = a + 2; d = 3;

P $\rightarrow p_1; p_2; p_3;$ Q $\rightarrow q_1; q_2;$

entrelazados

$p_1; p_2; p_3; q_1; q_2;$
$p_1; p_2; q_1; p_3; q_2;$
$p_1; p_2; q_1; q_2; p_3;$
$p_1; q_1; p_2; p_3; q_2;$
$p_1; q_1; p_2; q_2; p_3;$
$p_1; q_1; q_2; p_2; p_3;$
$q_1; p_1; p_2; p_3; q_2;$
$q_1; p_1; p_2; q_2; p_3;$
$q_1; p_1; q_2; p_2; p_3;$
$q_1; q_2; p_1; p_2; p_3;$

Sincronización (Enfoque operacional)

a = 0; P $\rightarrow$ b = a; c = b + 1; a = c; Q $\rightarrow$ a = a + 2; d = 3;

P $\rightarrow p_1; p_2; p_3;$ Q $\rightarrow q_1; q_2;$

entrelazados

$p_1; p_2; p_3; q_1; q_2;$ a=3
$p_1; p_2; q_1; p_3; q_2;$
$p_1; p_2; q_1; q_2; p_3;$
$p_1; q_1; p_2; p_3; q_2;$
$p_1; q_1; p_2; q_2; p_3;$
$p_1; q_1; q_2; p_2; p_3;$
$q_1; p_1; p_2; p_3; q_2;$
$q_1; p_1; p_2; q_2; p_3;$
$q_1; p_1; q_2; p_2; p_3;$
$q_1; q_2; p_1; p_2; p_3;$

Sincronización (Enfoque operacional)

a = 0; P $\rightarrow$ b = a; c = b + 1; a = c; Q $\rightarrow$ a = a + 2; d = 3;

P $\rightarrow p_1; p_2; p_3;$ Q $\rightarrow q_1; q_2;$

entrelazados

$p_1; p_2; p_3; q_1; q_2;$ a=3
$p_1; p_2; q_1; p_3; q_2;$ a=1
$p_1; p_2; q_1; q_2; p_3;$
$p_1; q_1; p_2; p_3; q_2;$
$p_1; q_1; p_2; q_2; p_3;$
$p_1; q_1; q_2; p_2; p_3;$
$q_1; p_1; p_2; p_3; q_2;$
$q_1; p_1; p_2; q_2; p_3;$
$q_1; p_1; q_2; p_2; p_3;$
$q_1; q_2; p_1; p_2; p_3;$

Sincronización (Enfoque operacional)

a = 0; P $\rightarrow$ b = a; c = b + 1; a = c; Q $\rightarrow$ a = a + 2; d = 3;

P $\rightarrow p_1; p_2; p_3;$ Q $\rightarrow q_1; q_2;$

entrelazados

$p_1; p_2; p_3; q_1; q_2;$	a=3
$p_1; p_2; q_1; p_3; q_2;$	a=1
$p_1; p_2; q_1; q_2; p_3;$	a=1
$p_1; q_1; p_2; p_3; q_2;$	
$p_1; q_1; p_2; q_2; p_3;$	
$p_1; q_1; q_2; p_2; p_3;$	
$q_1; p_1; p_2; p_3; q_2;$	
$q_1; p_1; p_2; q_2; p_3;$	
$q_1; p_1; q_2; p_2; p_3;$	
$q_1; q_2; p_1; p_2; p_3;$	

Sincronización (Enfoque operacional)

a = 0; P $\rightarrow$ b = a; c = b + 1; a = c; Q $\rightarrow$ a = a + 2; d = 3;

P $\rightarrow p_1; p_2; p_3;$ Q $\rightarrow q_1; q_2;$

entrelazados

$p_1; p_2; p_3; q_1; q_2;$	a=3
$p_1; p_2; q_1; p_3; q_2;$	a=1
$p_1; p_2; q_1; q_2; p_3;$	a=1
$p_1; q_1; p_2; p_3; q_2;$	a=1
$p_1; q_1; p_2; q_2; p_3;$	
$p_1; q_1; q_2; p_2; p_3;$	
$q_1; p_1; p_2; p_3; q_2;$	
$q_1; p_1; p_2; q_2; p_3;$	
$q_1; p_1; q_2; p_2; p_3;$	
$q_1; q_2; p_1; p_2; p_3;$	

Sincronización (Enfoque operacional)

a = 0; P $\rightarrow$ b = a; c = b + 1; a = c; Q $\rightarrow$ a = a + 2; d = 3;

P $\rightarrow p_1; p_2; p_3;$ Q $\rightarrow q_1; q_2;$

entrelazados

$p_1; p_2; p_3; q_1; q_2;$	a=3
$p_1; p_2; q_1; p_3; q_2;$	a=1
$p_1; p_2; q_1; q_2; p_3;$	a=1
$p_1; q_1; p_2; p_3; q_2;$	a=1
$p_1; q_1; p_2; q_2; p_3;$	a=1
$p_1; q_1; q_2; p_2; p_3;$	
$q_1; p_1; p_2; p_3; q_2;$	
$q_1; p_1; p_2; q_2; p_3;$	
$q_1; p_1; q_2; p_2; p_3;$	
$q_1; q_2; p_1; p_2; p_3;$	

Sincronización (Enfoque operacional)

a = 0; P $\rightarrow$ b = a; c = b + 1; a = c; Q $\rightarrow$ a = a + 2; d = 3;

P $\rightarrow p_1; p_2; p_3;$ Q $\rightarrow q_1; q_2;$

entrelazados

$p_1; p_2; p_3; q_1; q_2;$	a=3
$p_1; p_2; q_1; p_3; q_2;$	a=1
$p_1; p_2; q_1; q_2; p_3;$	a=1
$p_1; q_1; p_2; p_3; q_2;$	a=1
$p_1; q_1; p_2; q_2; p_3;$	a=1
$p_1; q_1; q_2; p_2; p_3;$	a=1
$q_1; p_1; p_2; p_3; q_2;$	
$q_1; p_1; p_2; q_2; p_3;$	
$q_1; p_1; q_2; p_2; p_3;$	
$q_1; q_2; p_1; p_2; p_3;$	

Sincronización (Enfoque operacional)

a = 0; P $\rightarrow$ b = a; c = b + 1; a = c; Q $\rightarrow$ a = a + 2; d = 3;

P $\rightarrow p_1; p_2; p_3;$ Q $\rightarrow q_1; q_2;$

entrelazados

$p_1; p_2; p_3; q_1; q_2;$	a=3
$p_1; p_2; q_1; p_3; q_2;$	a=1
$p_1; p_2; q_1; q_2; p_3;$	a=1
$p_1; q_1; p_2; p_3; q_2;$	a=1
$p_1; q_1; p_2; q_2; p_3;$	a=1
$p_1; q_1; q_2; p_2; p_3;$	a=1
$q_1; p_1; p_2; p_3; q_2;$	a=3
$q_1; p_1; p_2; q_2; p_3;$	
$q_1; p_1; q_2; p_2; p_3;$	
$q_1; q_2; p_1; p_2; p_3;$	

Sincronización (Enfoque operacional)

a = 0; P $\rightarrow$ b = a; c = b + 1; a = c; Q $\rightarrow$ a = a + 2; d = 3;

P $\rightarrow p_1; p_2; p_3;$ Q $\rightarrow q_1; q_2;$

entrelazados

$p_1; p_2; p_3; q_1; q_2;$	a=3
$p_1; p_2; q_1; p_3; q_2;$	a=1
$p_1; p_2; q_1; q_2; p_3;$	a=1
$p_1; q_1; p_2; p_3; q_2;$	a=1
$p_1; q_1; p_2; q_2; p_3;$	a=1
$p_1; q_1; q_2; p_2; p_3;$	a=1
$q_1; p_1; p_2; p_3; q_2;$	a=3
$q_1; p_1; p_2; q_2; p_3;$	a=3
$q_1; p_1; q_2; p_2; p_3;$	
$q_1; q_2; p_1; p_2; p_3;$	

Sincronización (Enfoque operacional)

a = 0; P $\rightarrow$ b = a; c = b + 1; a = c; Q $\rightarrow$ a = a + 2; d = 3;

P $\rightarrow p_1; p_2; p_3;$ Q $\rightarrow q_1; q_2;$

entrelazados

$p_1; p_2; p_3; q_1; q_2;$	a=3
$p_1; p_2; q_1; p_3; q_2;$	a=1
$p_1; p_2; q_1; q_2; p_3;$	a=1
$p_1; q_1; p_2; p_3; q_2;$	a=1
$p_1; q_1; p_2; q_2; p_3;$	a=1
$p_1; q_1; q_2; p_2; p_3;$	a=1
$q_1; p_1; p_2; p_3; q_2;$	a=3
$q_1; p_1; p_2; q_2; p_3;$	a=3
$q_1; p_1; q_2; p_2; p_3;$	a=3
$q_1; q_2; p_1; p_2; p_3;$	

Sincronización (Enfoque operacional)

a = 0; P $\rightarrow$ b = a; c = b + 1; a = c; Q $\rightarrow$ a = a + 2; d = 3;

P $\rightarrow p_1; p_2; p_3;$ Q $\rightarrow q_1; q_2;$

entrelazados

$p_1; p_2; p_3; q_1; q_2;$	a=3
$p_1; p_2; q_1; p_3; q_2;$	a=1
$p_1; p_2; q_1; q_2; p_3;$	a=1
$p_1; q_1; p_2; p_3; q_2;$	a=1
$p_1; q_1; p_2; q_2; p_3;$	a=1
$p_1; q_1; q_2; p_2; p_3;$	a=1
$q_1; p_1; p_2; p_3; q_2;$	a=3
$q_1; p_1; p_2; q_2; p_3;$	a=3
$q_1; p_1; q_2; p_2; p_3;$	a=3
$q_1; q_2; p_1; p_2; p_3;$	a=3

Sincronización (Enfoque operacional)

a = 0; P $\rightarrow$ b = a; c = b + 1; a = c; Q $\rightarrow$ a = a + 2; d = 3;

P $\rightarrow p_1; p_2; p_3;$ Q $\rightarrow q_1; q_2;$

entrelazados

$p_1; p_2; p_3; q_1; q_2;$	a=3 ✓
$p_1; p_2; q_1; p_3; q_2;$	a=1 ×
$p_1; p_2; q_1; q_2; p_3;$	a=1 ×
$p_1; q_1; p_2; p_3; q_2;$	a=1 ×
$p_1; q_1; p_2; q_2; p_3;$	a=1 ×
$p_1; q_1; q_2; p_2; p_3;$	a=1 ×
$q_1; p_1; p_2; p_3; q_2;$	a=3 ✓
$q_1; p_1; p_2; q_2; p_3;$	a=3 ✓
$q_1; p_1; q_2; p_2; p_3;$	a=3 ✓
$q_1; q_2; p_1; p_2; p_3;$	a=3 ✓

Sincronización (Enfoque axiomático)

$$\{P\}\ S\ \{Q\}$$

Sincronización (Enfoque axiomático)

$$\{P\}\ S\ \{Q\}$$

Ejemplo:

P1:

```
    ...
    { x > 0}
S1: x := 16;
    { x = 16}
    ...
```

P2:

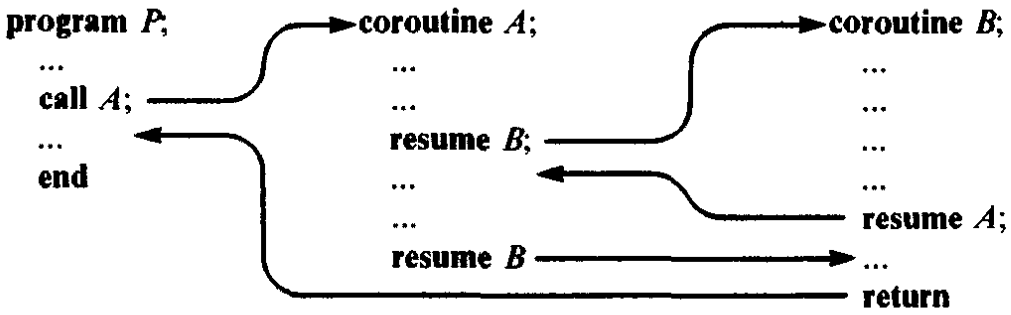
```
    ...
    { x < 0}
S2: x := -2;
    { x = -2}
    ...
```

Corutinas

- Son como subrutinas, pero permiten transferir el control simétricamente en vez de manera estrictamente jerárquica.
- El control se transfiere entre corutinas mediante la instrucción `resume`.

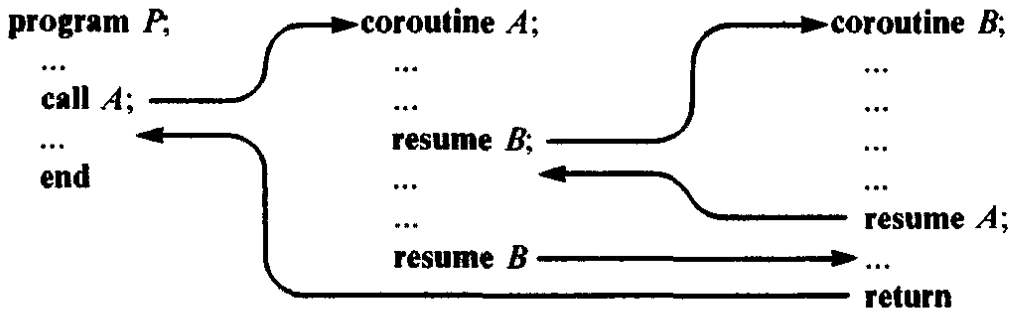
Corutinas

- Son como subrutinas, pero permiten transferir el control simétricamente en vez de manera estrictamente jerárquica.
- El control se transfiere entre corutinas mediante la instrucción `resume`.



Corutinas

- Son como subrutinas, pero permiten transferir el control simétricamente en vez de manera estrictamente jerárquica.
- El control se transfiere entre corutinas mediante la instrucción `resume`.



- Cada corutina puede verse como un proceso concurrente (en los cual el cambio entre procesos ha sido completamente especificado). La ejecución de `resume` produce la sincronización entre procesos.
- Aceptable para un único procesador, inadecuado para procesamiento en paralelo real.

Fork/Join

- La instrucción `fork`:
  - Especifica que una rutina dada debe empezar a ejecutarse
  - La rutina invocada y la rutina que ha hecho el `fork` siguen en paralelo.
- La instrucción `join`:
  - Permite sincronizar con la finalización de la rutina invocada.
  - Se retrasa la rutina que ha hecho el `join` hasta que la rútina invocada termina.

Fork/Join

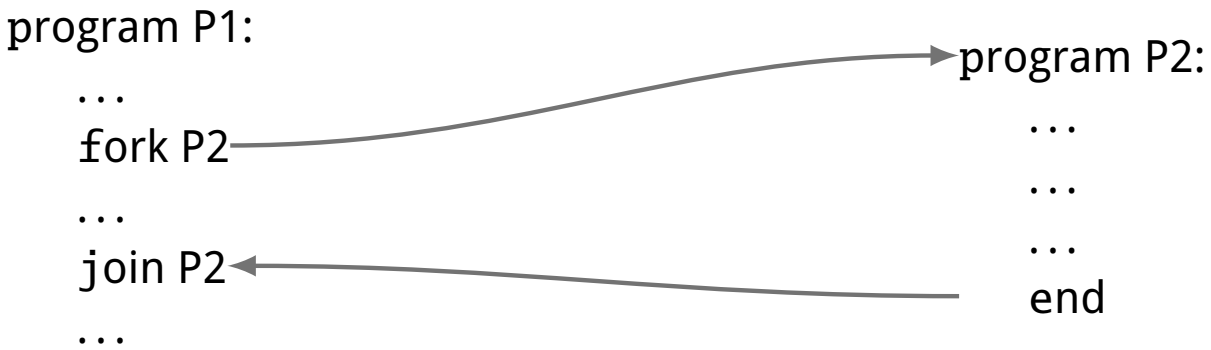
- La instrucción `fork`:
  - Especifica que una rutina dada debe empezar a ejecutarse
  - La rutina invocada y la rutina que ha hecho el `fork` siguen en paralelo.
- La instrucción `join`:
  - Permite sincronizar con la finalización de la rutina invocada.
  - Se retrasa la rutina que ha hecho el `join` hasta que la rútina invocada termina.

```
program P1:
    ...
    fork P2
    ...
    join P2
    ...
```

```
program P2:
    ...
    ...
    ...
    end
```

Fork/Join

- La instrucción `fork`:
  - Especifica que una rutina dada debe empezar a ejecutarse
  - La rutina invocada y la rutina que ha hecho el `fork` siguen en paralelo.
- La instrucción `join`:
  - Permite sincronizar con la finalización de la rutina invocada.
  - Se retrasa la rutina que ha hecho el `join` hasta que la rútina invocada termina.



Cobegin

- Manera estructurada de denotar ejecución concurrente de un conjunto de instrucciones.
- cobegin $S_1 \parallel S_2 \parallel \ \ldots \ \parallel S_n$ coend
  - Ejecución concurrente de $S_1, S_2, \ldots, S_n$.
- No es tan potente como `fork/join`, pero es suficiente para especificar la mayoría de las computaciones concurrentes.
- Hace explícita que rutinas son ejecutadas concurrentemente, y proporciona una estructura de control de una única entrada y salida.

Cobegin

- Manera estructurada de denotar ejecución concurrente de un conjunto de instrucciones.
- cobegin $S_1 \parallel S_2 \parallel \ldots \parallel S_n$ coend
  - Ejecución concurrente de $S_1, S_2, \ldots, S_n$.
- No es tan potente como `fork/join`, pero es suficiente para especificar la mayoría de las computaciones concurrentes.
- Hace explícita que rutinas son ejecutadas concurrentemente, y proporciona una estructura de control de una única entrada y salida.

```
begin /* sequential */
  s1;
  cobegin /* concurrent */
     begin /* sequential */
        s2;
        s4;
        cobegin /* concurrent */
           s5;
           s6
        coend
     end;
     s3
  coend;
  s7

end;
```

Declaración de procesos

- Proporciona una manera mas clara de declarar que una rutina será ejecutada concurrentemente.
- Distintas aproximaciones a la hora de ejecutar:
  - Todas al principio con un cobegin
  - Elegir cuando y que se quiere ejecutar (mediante un fork o similar)

Primitivas de sincronización basadas en variables compartidas

Sincronización = Exclusión mutua | Sincronización por condición

Primitivas de sincronización basadas en variables compartidas

Sincronización = Exclusión mutua | Sincronización por condición

Tipos de sincronización

- Exclusión mutua:
  - Asegura que una secuencia de instrucciones sea tratada como una operación indivisible.
  - Sección critica: Secuencia de instrucciones que debe parecer ejecutarse como una operación indivisible.
  - Ejecución mutuamente exclusiva de secciones críticas.
  - Ejemplo: Instrucción `x+=1`
- Sincronización por condición:
  - Consiste en retrasar un proceso hasta que una de las variables compartidas cumpla las condiciones necesarias para ejecutar la siguiente operación.
  - Ejemplo: Sacar elementos de un buffer vacío, o meterlos en uno lleno.

Problemas que pueden darse

- Bloqueo mutuo (Deadlock):
  - Estado en el cual dos o más procesos están esperando a evento que nunca ocurrirán.
- Innanición (Starvation):
  - Cuando un proceso se retrasa de manera infinita esperando a una condición.
  - Si un mecanismo de sincronización lo evite se le llama justo (fair). Si existe un limite superior de cuanto tiempo el proceso será retrasado, entonces se le llama justo acotado (bounded fair)

Espera activa (Busy-Waiting)

- Los procesos se dedican a dar valor y a examinar (set & test) variables compartidas.
  - Avisar condición $\rightarrow$ Dar valor a la variable
  - Esperar condición $\rightarrow$ Examinar repetidamente la variable hasta que tenga el valor deseado.
- Se llama de esta manera porque un proceso que está esperando a una condición debe examinar repetidas veces la variable compartida.
  - El proceso se dice que está haciendo spinning.
  - Las variables compartidas se llaman spinlocks.
- Buen enfoque para implementar sincronización por condición, pero no para exclusión mutua.

Espera activa (Busy-Waiting)

```
program Mutex_Example;
  var enter1, enter2 : Boolean initial (false,false);
      turn : integer initial ("P1");    { or "P2" }
  process P1;
    loop
      Entry_Protocol:
        enter1 := true;    { announce intent to enter }
        turn := "P2";    { set priority to other process }
        while enter2 and turn = "P2"
          do skip; { wait if other process is in and it is his turn }
      Critical Section;
      Exit_Protocol:
        enter1 := false;    { renounce intent to enter }
      Noncritical Section
      end
    end;
  process P2;
    loop
      Entry_Protocol:
        enter2 := true;    { announce intent to enter }
        turn := "P1";    { set priority to other process }
        while enter1 and turn = "P1"
          do skip; { wait if other process is in and it is his turn }
      Critical Section;
      Exit_Protocol:
        enter1 := false;    { renounce intent to enter }
      Noncritical Section
      end
    end
end.
```

- Sin bloqueo mutuo: Podría pasar si cada proceso pudiese hacer spin infinitamente en su protocolo de entrada. La variable `turn` lo impide.
- Justo acotado: Un proceso esperando a entrar en su sección crítica es retrasado como mucho una ejecución de la sección critica del otro proceso. La variable `turn` lo asegura.

Espera activa (Busy-Waiting)

- Los protocolos de sincronización que utilizan únicamente la espera activa son difíciles de diseñar, comprender y de probar su corrección.
- Se malgastan ciclos de procesador.
- Se carga al programador con la con la responsabilidad de decidir que sincronización se requiere y como proporcionarla.
- Puede no estar claro para el lector que variables se usan para la sincronización y que variables se usan para la comunicación entre procesos.

Semáforos

- Un semáforo es una variable de tipo entero no negativo en el cual se definen dos operaciones, P y V. Dado un semáforo s,
  - $P(s)$ espera hasta que $s > 0$ y entonces ejecuta $s = s - 1$
  - $V(s)$ ejecuta $s = s + 1$
- Implementación de cada uno de los tipos de sincronización:
  - Exclusión mutua:
    - Cada sección crítica es precedida por una operación P y seguida de una operación V en el mismo semáforo.
    - Todas las secciones criticas exclusivas usan el mismo semáforo, que se incializa a 1 (semáforo binario).
  - Sincronización por condición:
    - Las variables compartidas se usan para representar la condición, y un semáforo asociado con la condición se usa para conseguir la sincronización.
    - Cuando un proceso hace la condición verdad, avisa ejecutando una operación V, mientras que retrasa su ejecución ejecutando una operación P.
    - Suelen utilizarse semáforos generales o también llamados de conteo. Llevan el registro de recursos compartidos con un determinado número de unidades.

Semáforos (Exclusión mutua)

```
program Mutex_Example;
  var mutex : semaphore initial (1);
  process P1;
    loop
      P(mutex);        { Entry Protocol }
      Critical Section;
      V(mutex);        { Exit Protocol }
      Noncritical Section
      end
    end;
  process P2;
    loop
      P(mutex);        { Entry Protocol }
      Critical Section;
      V(mutex);        { Exit Protocol }
      Noncritical Section
      end
    end
  end.
```

- Solución simple y simétrica
- Se asegura la exclusión mutua, está libre de bloqueos mutuos y además, si la implementación del semáforo es justa, esta solución también será justa.

Semáforos (Ex. mutua y Sinc. por condición)

```
program OPSYS;
  var in_mutex, out_mutex : semaphore initial (1,1);
      num_cards, num_lines : semaphore initial (0,0);
      free_cards, free_lines : semaphore initial (N,N);
      input_buffer : array [0..N-1] of cardimage;
      output_buffer : array [0..N-1] of lineimage;

  process reader;
    var card : cardimage;
    loop
      read card from cardreader;
      P(free_cards); P(in_mutex);
        deposit card in input_buffer;
      V(in_mutex); V(num_cards)
      end
    end;

  process executer;
    var card : cardimage;
        line : lineimage;
    loop
      P(num_cards); P(in_mutex);
        fetch card from input_buffer;
      V(in_mutex); V(free_cards);
      process card and generate line;
      P(free_lines); P(out_mutex);
        deposit line in output_buffer;
      V(out_mutex); V(num_lines)
      end
    end;

  process printer;
    var line : lineimage;
    loop
      P(num_lines); P(out_mutex);
        fetch line from output_buffer;
      V(out_mutex); V(free_lines);
      print line on lineprinter
      end
    end

end.
```

- Implementa ambos tipos de sincronización
  - Exclusión mutua: Semáforos `in_mutex` and out_mutex
  - Sincronización por condición: Semáforos `num_cards`, `num_lines`, `free_cards` and `free_lines`.

Semáforos (Ex. mutua y Sinc. por condición)

```
program OPSYS;
  var in_mutex, out_mutex : semaphore initial (1,1);
      num_cards, num_lines : semaphore initial (0,0);
      free_cards, free_lines : semaphore initial (N,N);
      input_buffer : array [0..N-1] of cardimage;
      output_buffer : array [0..N-1] of lineimage;
  process reader;
    var card : cardimage;
    loop
      read card from cardreader;
      P(free_cards); P(in_mutex);
        deposit card in input_buffer;
      V(in_mutex); V(num_cards)
      end
    end;
  process executer;
    var card : cardimage;
        line : lineimage;
    loop
      P(num_cards); P(in_mutex);
        fetch card from input_buffer;
      V(in_mutex); V(free_cards);
      process card and generate line;
      P(free_lines); P(out_mutex);
        deposit line in output_buffer;
      V(out_mutex); V(num_lines)
      end
    end;
  process printer;
    var line : lineimage;
    loop
      P(num_lines); P(out_mutex);
        fetch line from output_buffer;
      V(out_mutex); V(free_lines);
      print line on lineprinter
      end
    end
end.
```

- Implementa ambos tipos de sincronización
  - Exclusión mutua: Semáforos `in_mutex` and out_mutex
  - Sincronización por condición: Semáforos num_cards, num_lines, `free_cards` and `free_lines`.
- El orden que se hacen la operaciones P es importante. Un cambio podría llevar a situaciones de bloqueo mutuo. El de las operaciones V es irrelevante.

Concurrencia (bis)

Concurrencia = Ejecución simultánea & Indeterminismo & Interacción

Interacción = Comunicación & Sincronización

Sincronización = Exclusión mutua | Sincronización por condición

Concurrencia
Conceptos y notación

Salvador Tamarit (UPM)

Grado en Ingeniería Informática
Grado en Matemáticas e Informática
Universidad Politécnica de Madrid

2014-2015