

# Sistemas Operativos

## Introducción a la concurrencia

---

Curso 2024

Facultad de Ingeniería, UDELAR

# Agenda

1. Introducción
2. Grafos de precedencia
3. Cobegin - Coend
4. Fork - Join

# Introducción

---

## Procesos cooperativos

- Se llaman **procesos cooperativos** a aquellos que pueden afectar el estado de otros procesos o cuyo estado es afectado por otros procesos.
- Esto puede ocurrir al compartir un espacio de memoria (ej. hilos) o mediante primitivas de comunicación entre procesos.
- El acceso concurrente a los mismos datos puede generar inconsistencias si no se tiene cuidado.
  - Recordar que por ejemplo un planificador expropiativo puede quitarle la CPU a un proceso en cualquier momento.
- Las técnicas de programación concurrente permiten resolver estos problemas en forma segura y eficiente.

## Ejemplo

Variable A es compartida

**Begin**

A := 1;

**Print** (A);

**End**

**Begin**

A := 2;

**Print** (A);

**End**

Resultados posibles:

- 1, 2
- 1, 1
- 2, 1
- 2, 2

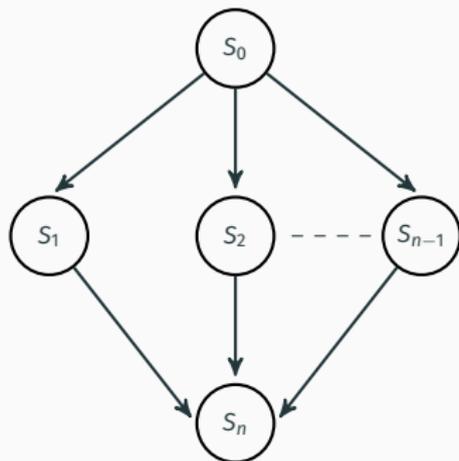
# Grafos de precedencia

---

## Grafos de precedencia

- Una posible solución a este problema es definir que tareas deben esperar por otras y cuales pueden ejecutar en paralelo.
- Esto puede aplicar para procesos, hilos de un mismo proceso o distintas secciones de código de cada proceso.
- Un **grafo de precedencia** es un grafo acíclico y dirigido cuyos nodos son tareas y cuyas aristas indican la precedencia.
- Un grafo de precedencia permite especificar el orden que que se deben ejecutar los procesos

## Grafo de precedencia



- $S_0$  no depende de nadie
- $S_1, S_2, \dots, S_{n-1}$  dependen de  $S_0$ , no hay restricciones entre ellos
- $S_n$  depende de  $S_1, S_2, \dots, S_{n-1}$

**Cobegin - Coend**

---

## Cobegin - Coend

- Una herramienta para declarar procesos concurrentes es el uso de **Cobegin - Coend**.
- Permite definir (algunos) grafos de precedencia declarativamente.
- Todas las sentencias dentro del bloque **Cobegin - Coend** se ejecutan concurrentemente
- Por ejemplo, para representar el grafo anterior:

```
Begin  
  S0;  
  Cobegin  
    S1;  
    S2;  
    ...  
    Sn-1;  
  Coend  
  Sn;  
End
```

# Ejemplos

**Begin**

**Cobegin**

A := SemiFactorial(n, floor(n/2));

B := SemiFactorial(floor(n/2)-1, 1);

**Coend**

**Return** A \* B;

**End**

**Begin**

A := 10

**Cobegin**

**Print** (A);

▷ A = ?

A := 100;

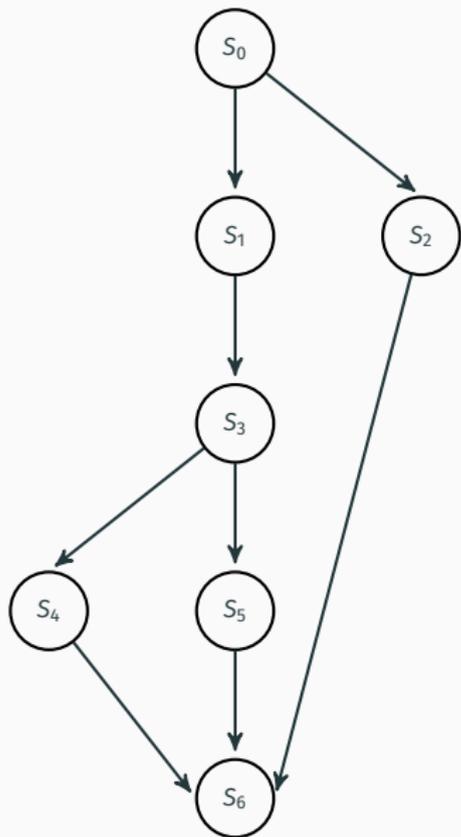
**Coend**

**Print** (A);

▷ A = 100

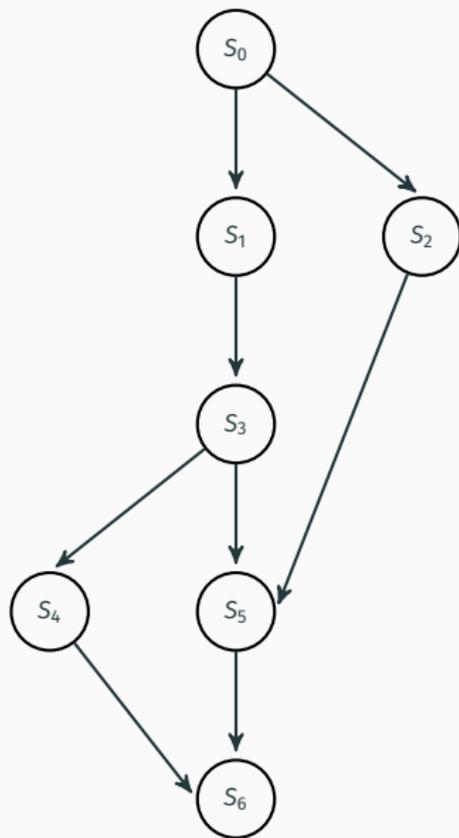
**End**

## Ejemplo de grafo



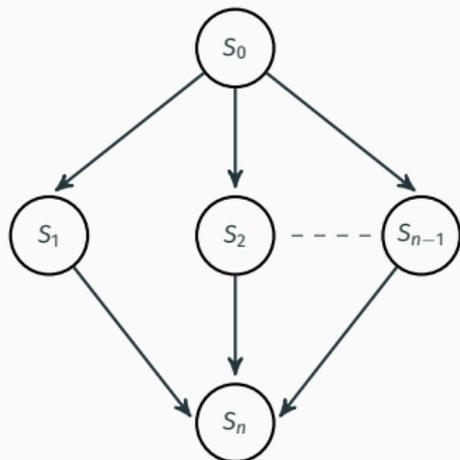
```
Begin  
  S0;  
  Cobegin  
    Begin  
      S1;  
      S3;  
    Cobegin  
      S4;  
      S5;  
    Coend  
  End  
  S2;  
  Coend  
  S6;  
End
```

## Ejemplo no representable



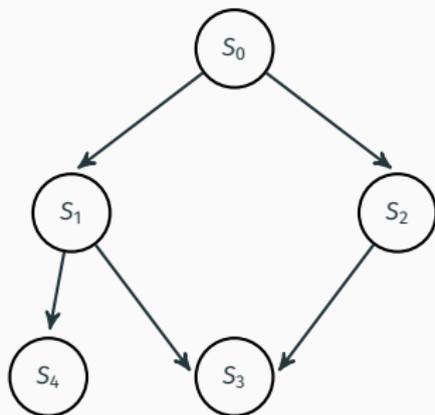
# Grafos representables

- Los grafos representables con cobegin-coend tienen que tener el siguiente formato.



# Grafos representables

- No se pueden representar grafos con la siguiente estructura general.



## Fork - Join

---

## Fork - Join

- Se necesitan más herramientas para poder programar cualquier grafo de dependencias.
- No es lo mismo que el fork-wait de unix aunque es la misma idea. Es una abstracción.
- **Fork** <etiqueta> Divide en dos el proceso. Uno de ellos continúa ejecutando luego del **Fork** y el otro salta a la etiqueta.
- **Join** <contador> <etiqueta> Cada vez que un proceso ejecuta **Join**, el contador (que es una variable global) se decrementa uno. Si el contador queda en 0 el proceso salta a la etiqueta, en otro caso termina.
- **Goto** <etiqueta> El proceso salta a la etiqueta.
- **Quit** El proceso termina.

# Ejemplo

- Para el grafo de la diapositiva 6:

**Begin**

$S_0$ ;

total :=  $N - 1$ ;

**Fork** L2;

**Fork** L3;

...

**Fork**  $L_{N-1}$ ;

$S_1$ ;

**Join** total FIN;

L2:

$S_2$ ;

**Join** total FIN;

...

$L_{N-1}$ :

$S_{N-1}$ ;

**Join** total FIN;

FIN:

$S_N$ ;

**End**

# Ejemplo

- Para el grafo de la diapositiva 10:

```
Begin  
  total1 := 2;  
  total2 := 2;  
  S0;  
  Fork L2;  
  S1;  
  S3;  
  Fork L4;  
  Join total1 NEXT;  
L2:  
  S2;  
  Join total1 NEXT;  
NEXT:  
  S5;  
  Goto L5;  
L4:  
  S4;  
L5:  
  Join total2 FIN;  
FIN:  
  S6;  
End
```