Departamento de Arquitectura

Sistemas Operativos
Práctico 3

Curso 2025

Objetivos

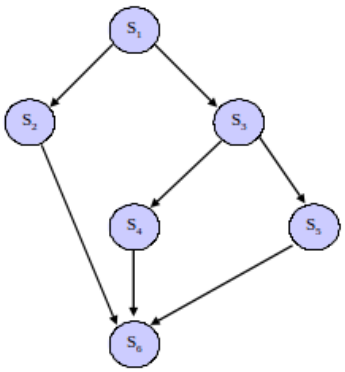
- Comprender el problema de la mutua exclusión y las dificultades de probar la correctitud de programas concurrentes.
- Ver soluciones por software y por hardware al problema de la mutua exclusión.
- Familiarizarse con el uso de semáforos.

Duración

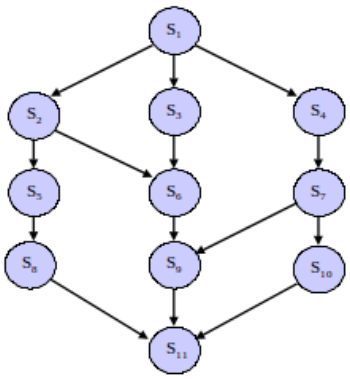
- 1 semana.

Ejercicio 1 (básico) Dados los siguientes grafos de precedencia, ¿Pueden representarse con `cobegin-coend`? Si no es posible, mostrar por qué.

a)



b)



Ejercicio 2 (básico) Represente los grafos del ejercicio anterior usando `fork-join`.

Ejercicio 3 (básico) Escribir el grafo de ejecución y un programa que evalúe la expresión (1), usando `cobegin-coend` de modo de alcanzar el mayor grado posible de concurrencia. La evaluación de una expresión sólo debe esperar por la evaluación de sus subexpresiones.

$$\frac{3 \times a \times b + 4}{(c+d)^{e-f}} \tag{1}$$

Ejercicio 4 (básico) Construir un programa que usando cobegin-coend calcule el producto de una matriz de 3×3 por otra de 3×2 con 6 procesos concurrentes.

Ejercicio 5 (en OpenFing) Se desea contar el número de veces que es ejecutado un determinado proceso y para esto se definen dos contadores globales:

```
var unidades, decenas : integer := 0
```

Que son usados de la siguiente forma:

```
procedure actividad is
begin
  for i in 1..12 loop
    actividad_propia_del_proceso
    unidades := unidades + 1;
    if (unidades = 10) then
      unidades := 0;
      decenas := decenas + 1;
    end if;
  end loop;
end procedure;
```

Se ejecutan **concurrentemente** dos copias de actividad:

- Analizar la conducta del algoritmo e indicar algunos de los resultados posibles de los contadores.
- Indicar cómo podría obtenerse siempre el resultado correcto.

Ejercicio 6 (medio) Implementar una solución al problema 5 usando el algoritmo de Dekker. ¿Qué desventajas tiene presenta este algoritmo?

Ejercicio 7 (básico) Implementar una solución al problema 5 utilizando semáforos.

Ejercicio 8 (medio) Dado el siguiente programa:

```
program prueba_incremento;
  var n : integer;

  procedure incremento;
    var i : integer;
  begin
    for i in 1..20 loop
      n := n + 1;
  end procedure;

begin { program }
  n := 0;
  cobegin
    incremento;
    incremento;
  coend;
  writeln('la␣suma␣es␣:␣', n)
end program.
```

Discutir los posibles resultados en las siguientes situaciones:

1. La máquina tiene un solo procesador y la construcción `n:=n+1` se ejecuta en **una sola** instrucción de máquina.
2. La máquina tiene un solo procesador y la construcción `n:=n+1` se ejecuta en **más de una** instrucción de máquina.

Ejercicio 9 (medio) Explicar por qué las construcciones P(s) y V(s) deben ejecutarse en forma indivisible en un semáforo no binario. Comentar el caso de semáforo binario.

Ejercicio 10 (OpenFing) Se desea controlar el proceso de llenado y tapado de envases de la Compañía Algarrobo Cola. Existe una sola cinta transportadora y una máquina para cada acción. Se dispone de los siguientes procedimientos:

- **avanzo_cinta()** : Avanza la cinta un paso (muy pequeño).
- **llenar_botella()** : Llena la botella de líquido (finaliza cuando está llena).
- **pongo_tapa()** : Comienza la acción de tapado (finaliza cuando la tapa queda colocada).

Y de las siguientes funciones:

- **puedo_llenar()** : Devuelve TRUE si hay un envase bajo la estación de llenado.
- **puedo_tapar()** : Ídem para la estación de tapado.

Se pide: Resolver utilizando **semáforos**.

Nota: Las botellas vienen a intervalos irregulares y no siempre se está en situación de accionar las dos máquinas a la vez. Se desea conseguir los algoritmos más sencillos posibles que resuelvan correctamente el problema.

Ejercicio 11 (avanzado) Se desea modelar con **semáforos** las tareas VENDEDOR, CLIENTE y SUPERVISOR de una cafetería. Esta cafetería dispone de una caja, 8 vendedores y 2 supervisores. Los vendedores toman el pedido, cobran y elaboran el pedido de cada cliente. Tienen orden de no dejar la caja sola, por lo que para poder ir a elaborar el pedido debe haber otro vendedor en el área de la caja. Por razones de espacio no puede haber más de 2 vendedores en el área de caja a la vez, de los cuales solo uno de ellos puede estar atendiendo. Los vendedores solo toman pedidos cuando están en la caja. Los vendedores deben atender a los clientes lo antes posible, no se debe hacer esperar a un cliente si la caja está libre.

Los vendedores le avisarán al grupo de supervisores cada 10 pedidos cobrados por ese vendedor. Los supervisores estarán esperando ser avisados para llenar la planilla. El primer supervisor libre recibirá el número de vendedor y con ese dato llenará la planilla.

Se dispone de los siguientes procedimientos:

- **recibir_pedido()**, **cobrar_pedido()**, **elaborar_pedido()**. Ejecutados por el vendedor.
- **enviar_pedido_pagar_y_recibir_pedido()**. Ejecutado por el cliente, retorna cuando el vendedor termina de elaborar el pedido.
- **llenar_planilla(int nro_vendedor)**. Ejecutado por el supervisor actualiza la planilla de ventas.

Aclaraciones:

- No se permite la implementación de tareas auxiliares.
- Todos los procedimientos disponibles pueden demorar tiempos variables.

Ejercicio 12 (medio) Se desea modelar una piscina pública. Dadas las restricciones por COVID solo se permite hasta 25 personas bañándose simultáneamente. Luego de que está llena, las personas que llegan a continuación deben esperar afuera hasta que se libere un lugar. Cada cierto tiempo un empleado debe aspirar la basura del fondo de la piscina y para ello la misma debe estar vacía. Las personas que se bañan tienen prioridad para entrar sobre el empleado que debe esperar a que nadie quiera usar la piscina.

Se pide: Implementar las personas y el empleado usando **semáforos**. Se dispone de las siguientes funciones auxiliares:

- **bañarse()**: Ejecutada por las personas para usar la piscina
- **limpiar()**: Ejecutada por el empleado para aspirar el fondo de la piscina

Ejercicio 13 (avanzado) Se desea modelar una construcción de un edificio. En la misma participan 4 obreros los cuales levantan paredes obteniendo ladrillos desde un pallet que contiene 300 ladrillos. Los obreros toman de a 3 ladrillos, los colocan en la pared y vuelven a tomar ladrillos nuevos.

Cuando un obrero detecta que no hay suficientes ladrillos le avisa al operador de la grúa para que se coloque un nuevo pallet y espera a que haya uno nuevo para tomar los ladrillos. El pallet solo se podrá cambiar mientras no haya obreros sacando ladrillos.

Cada cierto tiempo el capataz de la obra revisará los ladrillos consumidos. Para eso deberá tener acceso exclusivo al pallet y tendrá prioridad sobre los empleados y la grúa.

Se pide: Implementar usando **semáforos** a los obreros, el operador de la grúa y el capataz. No se pueden usar tareas auxiliares.

Se dispone de las siguientes funciones auxiliares:

- **tomar_ladrillos()**: Ejecutada por los obreros para sacar 3 ladrillos del pallet
- **colocar_ladrillos()**: Ejecutada por los obreros para agregar los ladrillos a la pared
- **cambiar_pallet()**: Ejecutada por el operador de la grúa para cambiar el pallet
- **contar_ladrillos()**: Ejecutada por el capataz para revisar los ladrillos consumidos
- **otras_tareas()**: Ejecutada por el capataz cuando no revisa los ladrillos