Departamento de Arquitectura

Sistemas Operativos
Práctico 3

Curso 2024

Objetivos

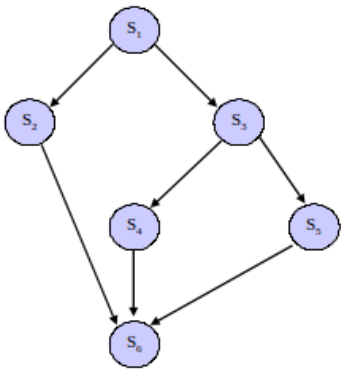
- Comprender el problema de la mutua exclusión y las dificultades de probar la correctitud de programas concurrentes.
- Ver soluciones por software y por hardware al problema de la mutua exclusión.
- Familiarizarse con el uso de semáforos

Duración

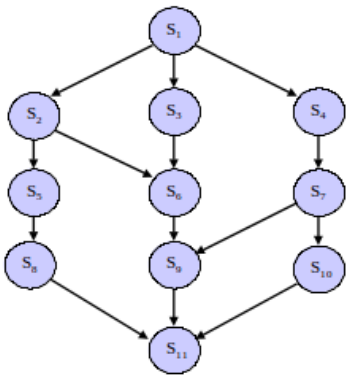
- 1 semana.

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 (medio)

(a) ¿Cuál es el resultado de intercambiar las instrucciones P del procedimiento $consumer\_process$ en la solución al problema $productor - consumidor$ que sigue?

(b) ¿Cuál es el resultado de intercambiar las instrucciones V del procedimiento $producer\_process$ en la solución al problema $productor - consumidor$ que sigue?

```
program producer_consumer_relationship;
  var  exclusive_access : semaphore;
       number_deposited : semaphore;
       buff : buffer;

  procedure producer_process;
    var next_result : integer;
  begin
    while true do
    begin
      calculate_next_result;
      P(exclusive_access);
      agrego_buffer(buffer, next_result);
      V(exclusive_access);
      V(number_deposited);
    end while;
  end procedure;

  procedure consumer_process;
    var next_result : integer;
  begin
    while true do
    begin
      P(number_deposited);
      P(exclusive_access);
      next_result := saco_buffer(buffer);
      V(exclusive_access);
      write(next_result);
    end while;
  end procedure;

begin {programa principal}
  init(exclusive_access, 1);
  init(number_deposited, 0);
  init_buffer(buffer);
  cobegin
    producer_process;
    consumer_process;
  coend
end program.
```