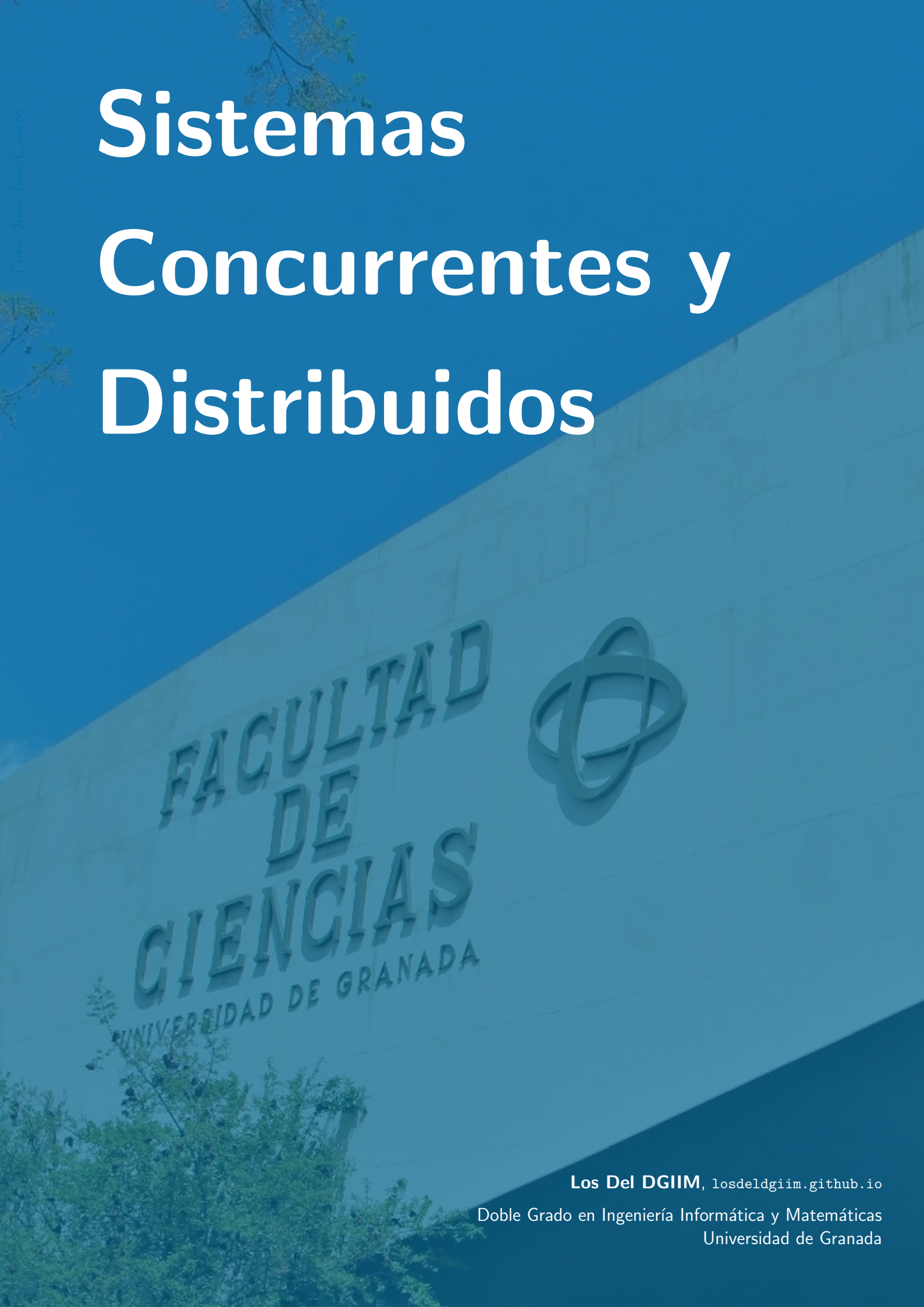
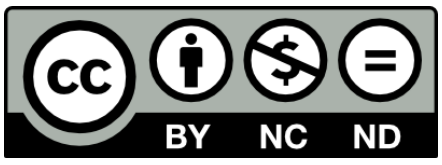
Sistemas Concurrentes y Distribuidos



Los Del DGIIM, losdeldgiim.github.io

Doble Grado en Ingeniería Informática y Matemáticas
Universidad de Granada



Sistemas Concurrentes y Distribuidos

Los Del DGIIM, `losdeldgiim.github.io`

Arturo Olivares Martos

Granada, 2024-2025

Índice general

1. Relaciones de problemas

1.1. Introducción

Ejercicio 1.1.1. Considerar el siguiente fragmento de programa para 2 procesos `P1` y `P2`: Los dos procesos pueden ejecutarse a cualquier velocidad. ¿Cuáles son los posibles valores resultantes para la variable `x`? Suponer que `x` debe ser cargada en un registro para incrementarse y que cada proceso usa un registro diferente para realizar el incremento.

```
{ variables compartidas }
var x : integer := 0 ;
Process P1;
var i: integer;
begin
  begin
    for i:= 1 to 2 do begin
      x:= x + 1;
    end
  end
end
```

```


Process P2;
var j: integer;
begin
  begin
    for j:= 1 to 2 do begin
      x:= x + 1;
    end
  end
end
```

Ejercicio 1.1.2. ¿Cómo se podría hacer la copia del fichero `f` en otro `g`, de forma concurrente, utilizando la instrucción concurrente `cobegin-coend`? Para ello, suponer que:

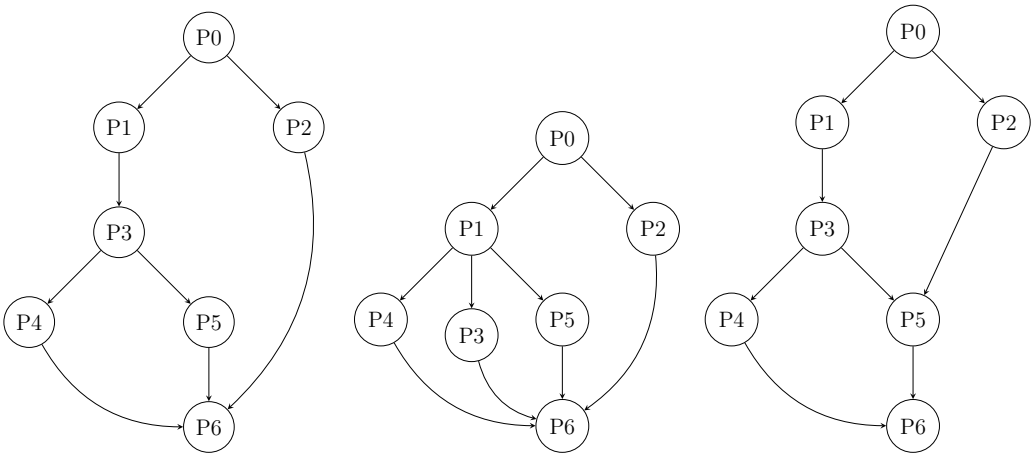
1. Los archivos son una secuencia de items de un tipo arbitrario `T`, y se encuentran ya abiertos para lectura (`f`) y escritura (`g`). Para leer un ítem de `f` se usa la llamada a función `leer(f)` y para saber si se han leído todos los ítems de `f`, se puede usar la llamada `fin(f)` que devuelve verdadero si ha habido al menos un intento de leer cuando ya no quedan datos. Para escribir un dato `x` en `g` se puede usar la llamada a procedimiento `escribir(g,x)`.

2. El orden de los items escritos en `g` debe coincidir con el de `f`.

3. Dos accesos a dos archivos distintos pueden solaparse en el tiempo.

Ejercicio 1.1.3. Construir, utilizando las instrucciones concurrentes `cobegin-coend` y `fork-join`, programas concurrentes que se correspondan con los grafos de precedencia que se muestran en la figura **??**.

1. Grafo de precedencia de la figura 1.1a:



(a) DAG del apartado 1. (b) DAG del apartado 2. (c) DAG del apartado 3.

Figura 1.1: Grafos de precedencia del ejercicio 1.1.3.

2. Grafo de precedencia de la figura 1.1b:

3. Grafo de precedencia de la figura 1.1c:

Ejercicio 1.1.4. Dados los siguientes fragmentos de programas concurrentes, obtener sus grafos de precedencia asociados:

1. Programa de la figura 1.2a.

2. Programa de la figura 1.2b.

```
begin
    P0 ;
    cobegin
        P1 ;
        P2 ;
        cobegin
            P3 ; P4 ; P5 ; P6 ;
        coend ;
        P7 ;
    coend ;
    P8 ;
end ;
```

(a) Programa 1.

```
begin
    P0 ;
    cobegin
        begin
            cobegin
                P1 ; P2 ;
            coend
            P5 ;
        end
        begin
            cobegin
                P3 ; P4 ;
            coend
            P6 ;
        end
    coend
    P7 ;
end
```

(b) Programa 2.

Figura 1.2: Programas concurrentes del ejercicio 1.1.4.