Lección 1: Introducción a la Programación de Sistemas Concurrentes y Distribuidos

- De la programación secuencial a la concurrente y distribuida
- Programas secuenciales, concurrentes y distribuidos
- La asignatura de PSCD
  - organización
  - bibliografía recomendada
  - contenidos del curso
  - notación algorítmica

De la programación secuencial a la concurrente y distribuida

- **Caso 1:**

```
   integer x := 0
--x = 0
x :=  1
--x = 1
```

```
   integer x := 0
--x = 0
x :=  2
--x = 2
```

```
      integer x := 0
       --x = 0
x := 1   ||   x := 2
       --x = ?????
```

De la programación secuencial a la concurrente y distribuida

- **Caso 2:**

```
   integer x := 0
--x = 0
x := x + 1
--x = 1
```

```
   integer x := 0
--x = 0
x := x + 1
--x = 1
```

```
          integer x := 0
            --x = 0
x := x + 1   ||    x := x + 1
            --x = ?????
```

Programas secuenciales, concurrentes y distribuidos

- Un programa concurrente se compone de ***procesos*** **y** ***objetos compartidos***
- Un proceso es un ***programa secuencial*** ejecutado en algún procesador
- Los objetos compartidos se construyen o bien mediante ***memoria compartida*** o bien mediante una ***red de comunicación***

Programas secuenciales, concurrentes y distribuidos

- Para cooperar, los procesos deben comunicar
  - la comunicación permite a un proceso influir en la ejecución de otro
- La comunicación puede tener retrasos
  - lo que implica que la información obtenida por un proceso respecto a otro puede estar desfasada
- En consecuencia: **desarrollar programas concurrentes/distribuidos correctos es más difícil que diseñar un conjunto de procesos correctos**

La asignatura de PSCD

- **Dónde encaja en nuestro plan de estudios**

PLAN DE ESTUDIOS - GRADO EN INGEN		
Cuatrimestre 1º	**Tipo**	**Créds.**
INTRODUCCIÓN A LOS COMPUTADORES	FB	6
FUNDAMENTOS DE ADMINISTRACIÓN DE EMPRESAS	FB	6
MATEMÁTICAS 1	FB	6
MATEMÁTICAS 2	FB	6
PROGRAMACIÓN 1	FB	6
Cuatrimestre 3º	**Tipo**	**Créds.**
TEORÍA DE LA COMPUTACIÓN	FB	6
SISTEMAS OPERATIVOS	OB	6
REDES DE COMPUTADORES	OB	6
PROGRAMACIÓN DE SISTEMAS CONCURRENTES Y DISTRIBUIDOS	OB	6
ESTRUCTURA DE DATOS Y ALGORITMOS	OB	6

La asignatura de PSCD. Resultados del aprendizaje

- El estudiante terminará con un **conocimiento** profundo de cuáles son las características específicas de los **sistemas concurrentes y distribuidos**
- Conocerá los **problemas** generados por el acceso concurrente a datos y recursos, así como las **soluciones conceptuales y tecnológicas** que se han dado a los mismos
- Tendrá nociones de qué son los sistemas **tiempo real**, y sistemas **basados en eventos**
- Conocerá **herramientas** para el diseño y programación de programas con características concurrentes y/o distribuidas

La asignatura de PSCD. Trabajo del estudiante

- 60 horas de actividades presenciales
  - sesiones de teoría, problemas y prácticas de laboratorio
- 85 horas de trabajo y estudio individual efectivo
  - estudio de textos, resolución de problemas, preparación de clases y prácticas, desarrollo de programas, etc.
- 5 horas dedicadas a distintas pruebas de evaluación

La asignatura de PSCD. Sesiones de problemas

- No hay horario específico
  - 15 sesiones con ejercicios programados
  - Planificación en Moodle (ambos grupos)
- Metodología de trabajo:
  - Trabajo previo en casa
  - Un rato de trabajo en grupo en el aula
  - Un rato de desarrollo de la solución

La asignatura de PSCD. Evaluación

- **Actividades de evaluación**
  - Prueba escrita (100%): Se plantearán cuestiones y/o problemas relacionados con el programa impartido en la asignatura, tanto relativos a los contenidos de clases de teoría y problemas como a las prácticas de laboratorio.

- **Calificación final**
  - En ambas convocatorias, para superar la asignatura, habrá que obtener una puntuación mayor o igual que 5.0.

La asignatura de PSCD. Cuestiones y fechas de interés

- Todos los materiales del curso estarán disponibles en **Moodle**
  - Transparencias de clase (**NO son apuntes, solo apoyo para la clase**)
  - Materiales de apoyo, enunciados de prácticas, etc.

- **Si no estás matriculado** en la asignatura, hay que auto-matricularse en el curso **antes del 4 de septiembre a las 17:00**
  - **Importante:** siempre con la cuenta **@unizar.es**
  - Clave = **Pscd_25_26!**

La asignatura de PSCD. Cuestiones y fechas de interés

- Creación de **grupos de prácticas**
  - Vía Moodle a partir del **4 de septiembre a las 17:00h hasta el 8 de septiembre a las 12:00h**
  - **Importante**: los estudiantes del turno de mañana deberán apuntarse en grupos de mañana y los del turno de tarde en grupos de tarde
    - Los **casos (muy) excepcionales** contactar con el profesor **Pedro Álvarez** vía correo electrónico (alvaper@unizar.es) **antes del 8 de septiembre a las 20:00h,** indicando claramente quién eres y justificando el motivo de la solicitud

La asignatura de PSCD. Prácticas de laboratorio

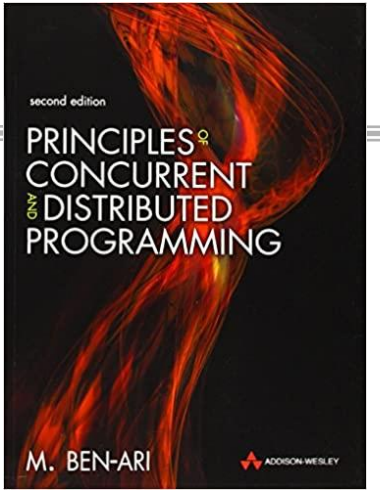
- Fechas de **inicio de las prácticas**:
  - Grupos de martes A: 16 de septiembre
  - Grupos de martes B: 23 de septiembre
  - Grupos de miércoles A: 10 de septiembre
  - Grupos de miércoles B: 17 de septiembre
  - Grupo de jueves A: 11 de septiembre
  - Grupo de jueves B: 18 de septiembre
- **Laboratorios** (en el Ada Byron):
  - L2.11: Grupos de martes A y B de 18:00 a 20:00
  - L0.03: Grupos de miércoles A de 8:00 a 10:00, de 17:00 a 19:00
  - L1.02: Grupos de miércoles B de 17:00 a 19:00
  - L0.01: todos los demás grupos

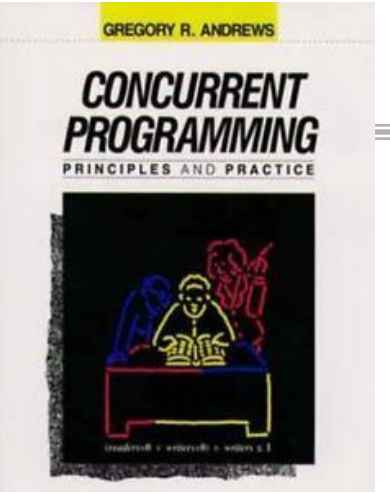
La asignatura de PSCD. Profesorado

- **Pedro Álvarez**
  - **alvaper@unizar.es**, despacho 2.16, ext. 5541
- **Joaquín Ezpeleta**
  - **ezpeleta@unizar.es**, despacho 1.17, ext. 1955
- **Simona Bernardi**
  - **simonab@unizar.es**, despacho 2.12, ext. 5531
- **Pendiente de contratar**

- Las tutorías se gestionarán vía Google Calendar del profesor (disponible en Moodle)

Bibliografía recomendada





- M. Ben-Ari
  Principles of Concurrent and Distributed Programming
  Addison-Wesley, 2006

- G.R. Andrews
  Concurrent Programming. Principles and practice
  Addison-Wesley, 2006

- A. Williams
  C++. Concurrency in Action
  Manning, 2012

- M.L. Liu
  Computación distribuida. Fundamentos y aplicaciones
  Ed. Pearson- Addison Wesley, 2004.

Contenidos del curso

Lección 1: Introducción a la programación concurrente

Lección 2: La programación concurrente

Lección 3: Sincronización de procesos. El problema de la sección crítica

Lección 4: Breve introducción a la lógica temporal y al "model checking"

Lección 5: Diseño de programas concurrentes

Lección 6: Sincronización de procesos mediante semáforos

Lección 7: Sincronización de procesos mediante monitores

Contenidos del curso

Lección 8: Introducción a la programación distribuida

Lección 9: Programación mediante paso síncrono de mensajes

Lección 10: Coordinación mediante espacios de tuplas

Lección 11: Algoritmos distribuidos

Lección 12: Algoritmos de consenso

Lección 13: Introducción a la programación de sistemas de tiempo real

Lección 14: Programación dirigida por eventos

Contenidos del curso. Trabajo de laboratorio

1. La programación concurrente. Threads y datos compartidos. Problemas de interferencias
2. Sincronización mediante esperas activas
3. Programación con semáforos
4. Programación con monitores
5. Programación de sistemas distribuidos-I
6. Programación de sistemas distribuidos-II

Notación algorítmica

```
constant integer i := 27
integer v := 27
real r
constant real PI := 3.1415926535 --el de siempre
boolean ha_ido_bien


integer array[1..100] mis_datos := (1..100, 0)

type integer array[1..n,1..n] mat_cuad
type integer array[1..n] vect
mat_cuad A
vect b,x
```

```
-----------------------------------
process multiplica
    mat_cuad A
    vect b,x
    ------------------------
    operation obtener_mat(REF mat_cuad m)
     --Pre: TRUE
     --Post: m se ha leído de la entrada estándar
    ------------------------
    operation obtener_vect(REF vect v)
     --Pre: TRUE
     --Post: v se ha leído de la entrada estándar
    ------------------------
     ....
end
```

```
----------------
a
.n
x[i]+A[i,j]*b[j]
    end

end
```

```
process multiplica
    mat_cuad A
    vect b,x
    ------------------------
    operation obtener_mat(REF mat_cuad m)
     --Pre: TRUE
     --Post: m se ha leído d
    ----------------------
    operation obtener_vec
     --Pre: TRUE
     --Post: v se ha leído d
    ----------------------
     ....
end
```

```
process multiplica
  ....
    for i:= 1..n
        x[i] := 0
        for j := 1..n
            x[i] := x[i]+A[i,j]*b[j]
        end
    end

    for i:= 1..n
        write(x[i])
    end
end
```

```
operation obtener_mat(REF mat_cuad m)
--Pre: TRUE
--Post: m se ha leído de la entrada estándar

   for i:= 1..n
      for j := 1..n
         read(m[i,j])
      end
   end
end
```

```
operation obtener_vect(REF vect v)
--Pre: TRUE
--Post: v se ha leído de la entrada estándar
   integer i := 1
   while i<=n
      read(v[i])
      i := i+1
   end
end
```