UNIVERSIDAD
DE GRANADA

# Sistemas Concurrentes y Distribuidos: Tema 4. Introducción a los sistemas de tiempo real.

Carlos Ureña / Jose M. Mantas / Pedro Villar

2019-20

**Grado en Ingeniería Informática / Grado en Ingeniería Informática y Matemáticas.**
Dpt. Lenguajes y Sistemas Informáticos
ETSI Informática y de Telecomunicación
**Universidad de Granada**

# Tema 4. Introducción a los sistemas de tiempo real.
## Índice.

1. Concepto de sistema de tiempo real. Medidas de tiempo y modelo de tareas.
2. Esquemas de planificación

# Concepto de sistema de tiempo real. Medidas de tiempo y modelo de tareas..

1.1. Definición, tipos y ejemplos
1.2. Propiedades de los Sistemas de Tiempo Real
1.3. Modelo de Tareas

# Subsección 1.1.
# Definición, tipos y ejemplos.

# Sistemas de Tiempo Real

- Constituyen un tipo de sistema en el que la ejecución del sistema se debe producir dentro de unos plazos de tiempo predefinidos para que funcione con la suficiente garantía.
- En un sistema además concurrente será necesario que todos los procesos sobre un procesador o sobre varios se ejecuten en los plazos de tiempo predefinidos.

# Definición de un Sistema de Tiempo Real

Stankovic (1997) da la siguiente definición:

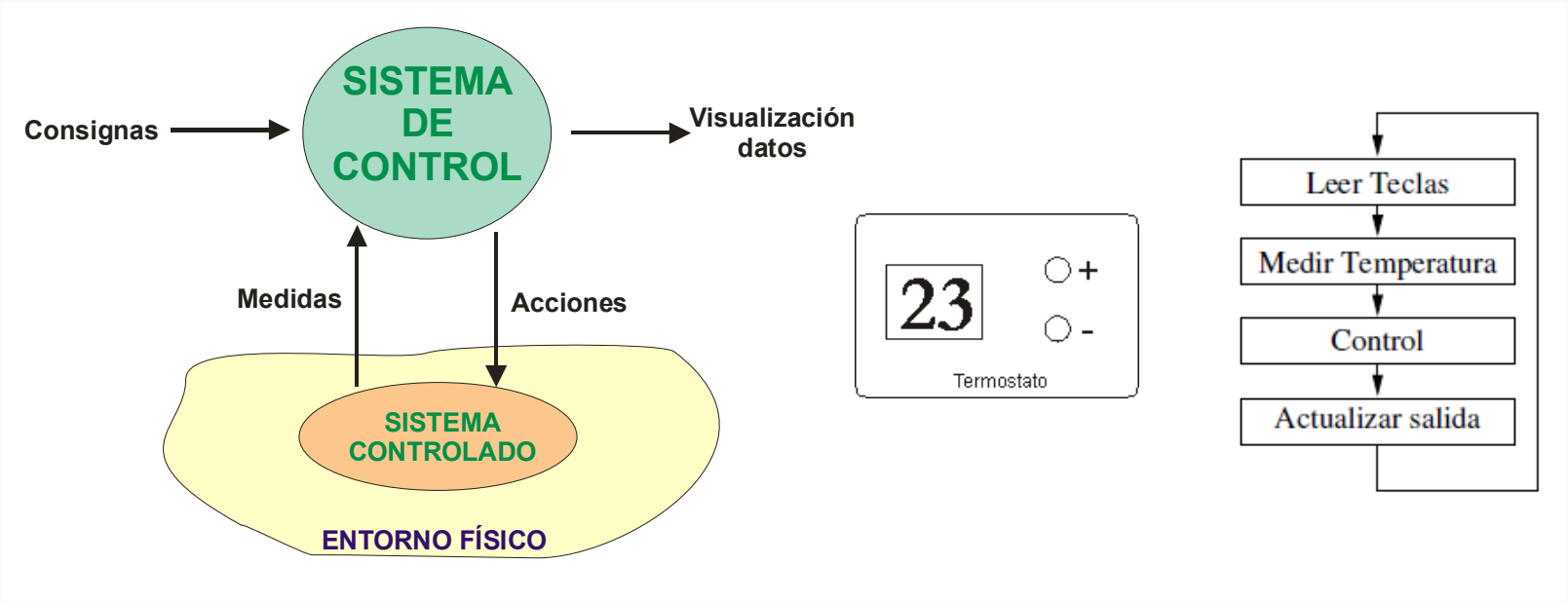
> *Un sistema de tiempo real es aquel sistema cuyo funcionamiento correcto depende no sólo de resultados lógicos producidos por el mismo, sino también del instante de tiempo en el que se producen esos resultados*

## Corrección Funcional + Corrección Temporal

El no cumplimiento de una restricción temporal lleva a un **fallo** del sistema

# Ejemplo: Sistema de Control

- Objetivo: Ejecutar el lazo de control del sistema en instantes de tiempo prefijados.
- Condición de tiempo real: no puede haber retrasos en la ejecución del lazo de control, ya que afecta al rendimiento y provoca perdida de estabilidad.



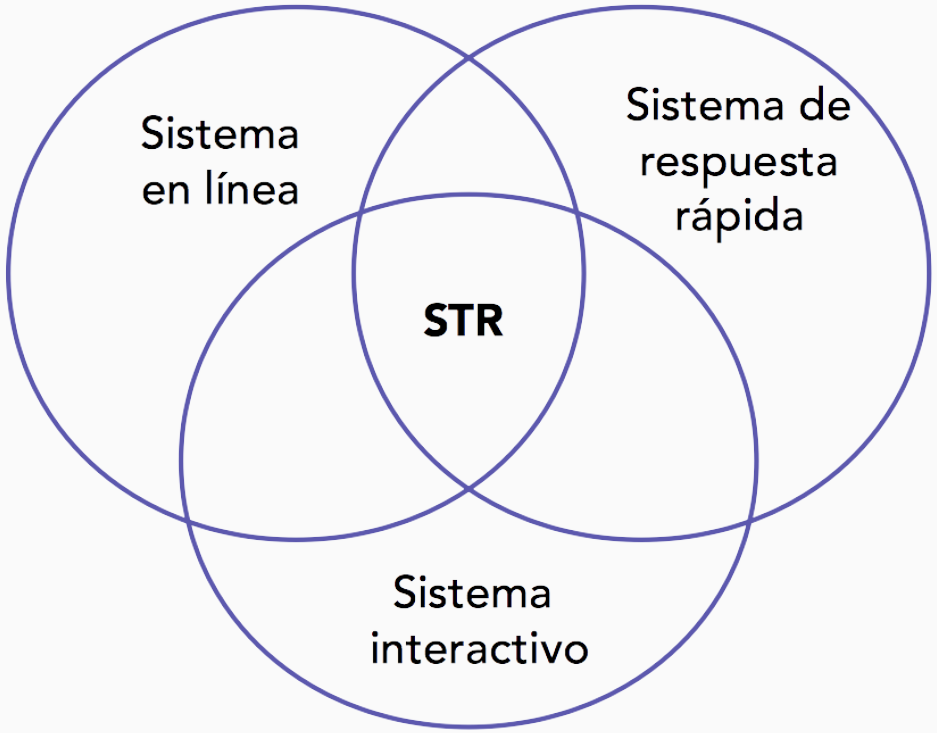
# Tipos de Sistemas de Tiempo Real

Habitualmente suele asociarse la denominación de sistemas de tiempo real a los siguientes tipos de sistemas:

- **Sistema en línea:** Siempre está disponible, pero no se garantiza una respuesta en un intervalo de tiempo acotado.
  - Ejemplos: Cajeros automáticos, sistemas de reservas de vuelo.
- **Sistema interactivo:** Suele ofrecer una respuesta en un tiempo acotado, aunque no importa si ocasionalmente tarda más:
  - Ejemplos: Reproductor DVD, sistema aire acondicionado, juegos, ..
- **Sistema de respuesta rápida:** El sistema ofrece una respuesta en un tiempo acotado y lo más corto posible:
  - Ejemplos: Sistema anti incendios, alarmas, etc.

# Tipos de Sistemas de Tiempo Real

Los sistemas de tiempo real presentan todas estas características:



# Ejemplos de Sistemas de Tiempo Real

En la actualidad hay muchos ejemplos de uso de Sistemas de Tiempo Real, podemos citar algunos de ellos:

- **Automoción:** sistema de ignición, transmisión, dirección asistida, frenos antibloqueo (ABS), control de la tracción, ...
- **Electrodomésticos:** televisores, lavadoras, hornos de microondas, reproductores de videos o DVDs, sistemas de seguridad y vigilancia, ...
- **Aplicaciones aeroespaciales:** control de vuelos, controladores de motores, pilotos automáticos, ...
- **Equipamiento médico:** como sistemas de monitorización de anestesia, monitores ECG,
- **Sistemas de defensa:** como sistemas radar de aviones, sistemas de radio, control de misiles, ...

# Subsección 1.2.
# Propiedades de los Sistemas de Tiempo Real.

# Propiedades de un STR

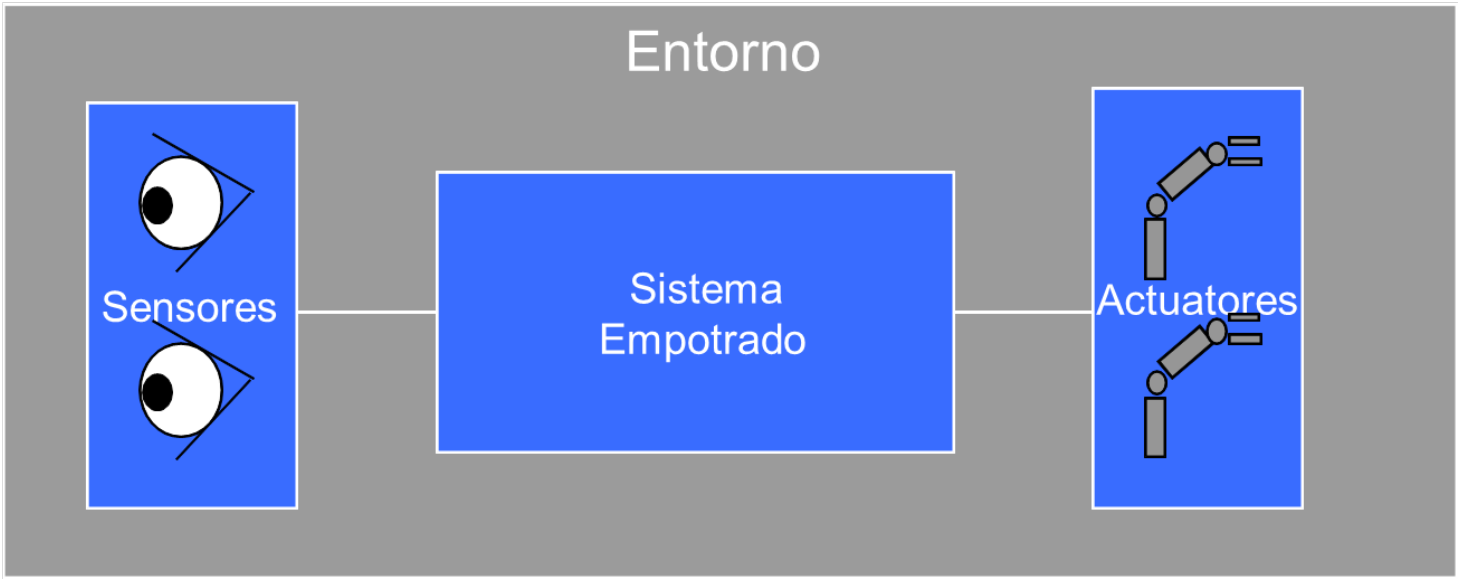
Al depender la corrección del sistema de las restricciones temporales, los sistemas de tiempo real tienen que cumplir una serie de propiedades:

- **Reactividad.**
- **Predecibilidad.**
- **Confiabilidad.**

a continuación veremos cada una de estas propiedades con más detalle.

# Propiedad: Reactividad

*El sistema tiene que interaccionar con el entorno, y responder de la manera esperada a los estímulos externos dentro de un intervalo de tiempo previamente definido.*



# Propiedad: Predecibilidad

Tener un comportamiento predecible implica que la ejecución del sistema tiene que ser determinista, y por lo tanto, se debe garantizar su ejecución dentro del plazo de tiempo definido.

- Las respuestas han de producirse dentro de las restricciones temporales impuestas por el entorno (sistema controlado), y que suele ser diferente para cada proceso del sistema.
- Es necesario conocer el comportamiento temporal de los componentes software (SO, middleware, librería, etc) y hardware utilizados, así como del lenguaje de programación.
- Si no se puede tener un conocimiento temporal exacto, hay que definir marcos de tiempo acotados; p.e. conocer el tiempo de peor ejecución de un algoritmo, el tiempo máximo de acceso a un dato de E/S)

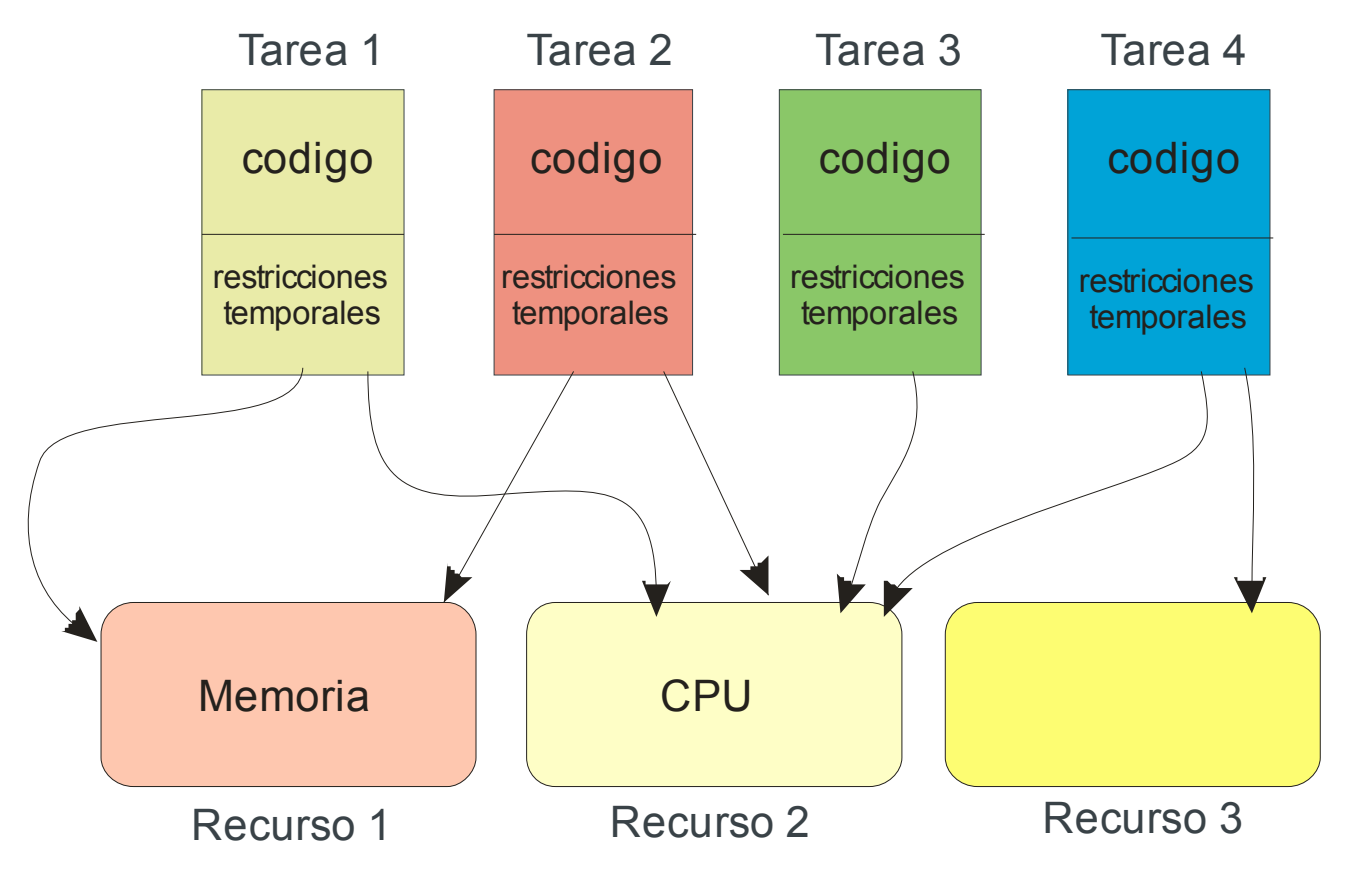
# Propiedad: Confiabilidad

- La **Confiabilidad** (***Dependability***) mide el grado de confianza que se tiene del sistema. Depende de:
  - Disponibilidad (*avalaibility*). Capacidad de proporcionar servicios siempre que se solicita.
  - Robustez o tolerancia a fallos (*fault tolerant*). Capacidad de operar en situaciones excepcionales sin poseer un comportamiento catastrófico.
  - Fiabilidad (*reliability*). Capacidad de ofrecer siempre los mismos resultados bajo las mismas condiciones.
  - Seguridad: Capacidad de protegerse ante ataques o fallos accidentales o deliberados (*safety*), y a la no vulnerabilidad de los datos (*security*).
- Cuando esta propiedad es crítica (su no cumplimiento lleva a pérdida humana o económica), el sistema se denomina sistema de tiempo real **crítico** (***safety-critical system***)
  - Ejemplos: Sistemas de Aviónica, Sistemas de automoción, Ce

# Subsección 1.3.
# Modelo de Tareas.

# Modelo de Tareas

Un sistema de tiempo real se estructura en **tareas** que acceden a los recursos del sistema:



# Tareas en un STR

Una **tarea** es un conjunto de acciones que describen el comportamiento del sistema o parte de él en base a la ejecución secuencial de un trozo de código. Equivalente a *proceso* o *hebra*.

- La tarea satisface una necesidad funcional concreta.
- La tarea tiene definida unas restricciones temporales a partir de los **Atributos Temporales**.
- Una **tarea activada** es una tarea en ejecución o pendiente de ejecutar.
- Decimos que una tarea *se activa* cuando se hace necesario ejecutarla una vez.
- El **instante de activación** de una tarea es el instante de tiempo a partir del cual debe ejecutarse (cuando pasa de desactivada a activada).

# Recursos en un STR. Atributos.

Los **recursos** son elementos disponibles para la ejecución de las tareas. Se distinguen dos tipos de recursos:

- Recursos Activos: Procesador, Red, ...
- Recursos Pasivos: Datos, Memoria, Dispositivos de E/S, ...

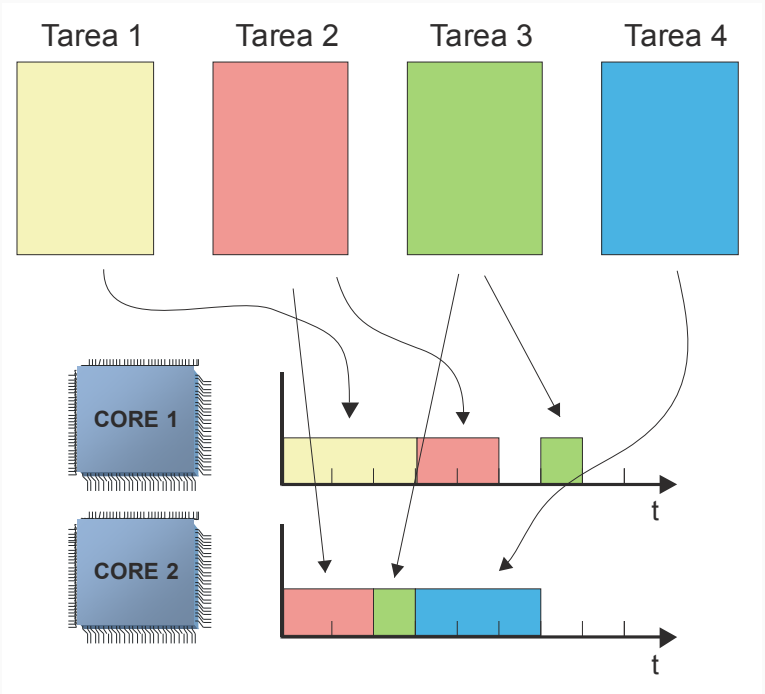
Asumimos que cada CPU disponible se dedica a ejecutar una o varias tareas, de acuerdo al esquema de planificación en uso.

- Si una CPU ejecuta más de una tarea, el tiempo de la CPU debe repartirse entre varias tareas.

Los requerimientos de un sistema de tiempo real obligan a asociar un conjunto de **atributos temporales** a cada tarea de un sistema. Estos atributos son restricciones acerca de cuando se ejecuta activa cada tarea y cuanto puede tardar en completarse desde que se activa.

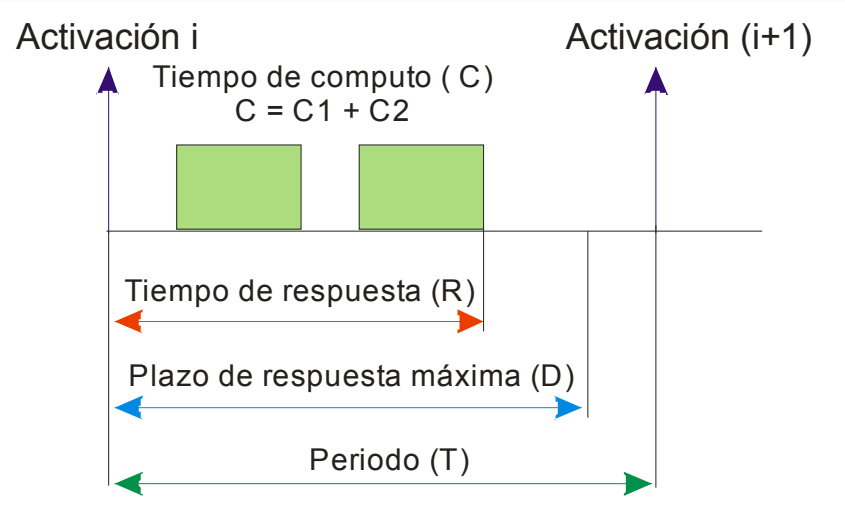
# Planificación de Tareas

La **planificación de las tareas** es una labor de diseño que determina como se le asignan (a lo largo del tiempo) a cada tarea los recursos activos de un sistema (principalmente el procesador o procesadores), de forma que se garantize el cumplimiento de las restricciones dadas por los atributos temporales de la tarea.



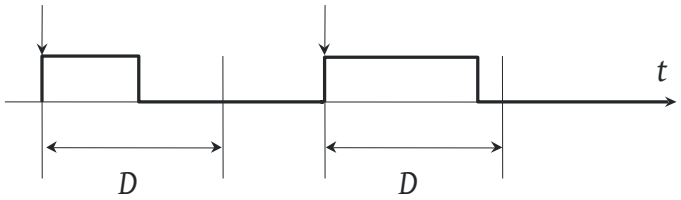
# Atributos temporales de una tarea

- **Tiempo de computo o de ejecución** ($C$): Tiempo necesario para la ejecución de la tarea.
- **Tiempo de respuesta** ($R$): Tiempo que ha necesitado el proceso para completarse totalmente a partir del instante de activación.
- **Plazo de respuesta máxima** (*deadline*) ($D$): Define el máximo tiempo de respuesta posible.
- **Periodo** ($T$): Intervalo de tiempo entre dos activaciones sucesivas en el caso de una tarea periódica.

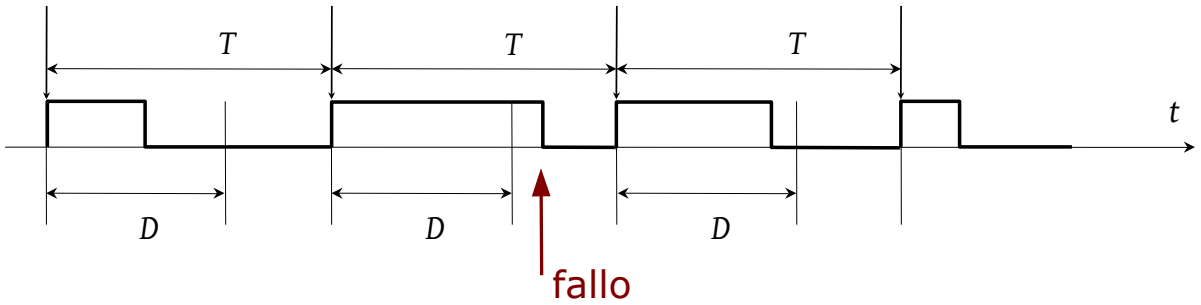


# Tipos de tareas según la recurrencia de sus activaciones

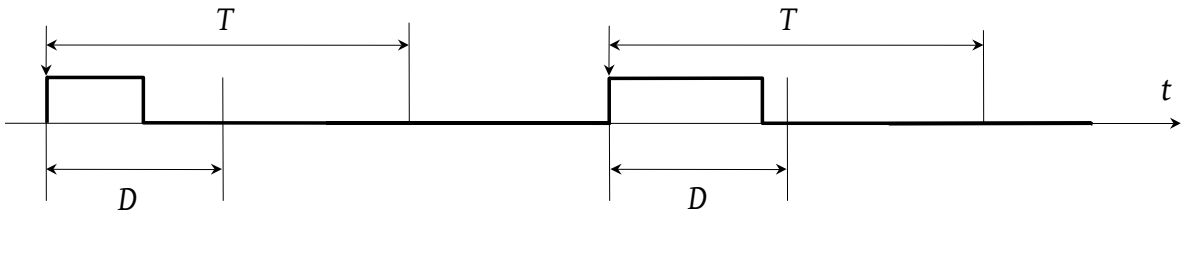
- **Aperiódica:** se activa en instantes arbitrarios (no tiene $T$)




- **Periódica:** repetitiva, $T$ es el tiempo exacto entre activaciones.




- **Esporádica:** repetitiva, $T$ es intervalo mínimo entre activaciones



# Diseño de la planificación de tareas.

El problema de la planificación supone:

- Determinar los procesadores disponibles a los que se puede asociar las tareas.
- Determinar las relaciones de dependencias de las tareas:
  - Relaciones de precedencia que hay entre las distintas tareas.
  - Determinar los recursos comunes a los que acceden las distintas tareas.
- Determinar el orden de ejecución de la tareas para garantizar las restricciones especificadas.

# Esquema de planificación de tareas

Para determinar la planificabilidad de un conjunto de tareas se requiere un esquema de planificación que cubra los dos siguientes aspectos:

- Un **algoritmo de planificación**, que define un criterio (política de planificación) que determina el orden de acceso de las tareas a los distintos procesadores.
- Un **método de análisis** (test de planificabilidad) que permite predecir el comportamiento temporal del sistema, y determina si la planificabilidad es factible bajo las condiciones o restricciones especificadas
  - Se pueden comprobar si los requisitos temporales están garantizados en **todos los casos posibles**.
  - En general se estudia el **peor comportamiento posible**, es decir, con el WCET (*Worst Case Execution Time*).

# Cálculo del WCET.

Suponemos que siempre se conoce el valor WCET ($C_w$), que es el máximo valor de $C$ para cualquier ejecución posible de dicha tarea. Hay dos formas de obtener $C_w$:

- **Medida directa** del tiempo de ejecución (en el peor caso) en la plataforma de ejecución.
  - Se realizan múltiples experimentos, y se hace una estadística.
- **Análisis del código**, se basa en calcular el peor tiempo:
  - Se descompone el código en un grafo de bloques secuenciales.
  - Se calcula el tiempo de ejecución de cada bloque.
  - Se busca el camino más largo.

En los ejemplos y ejercicios, y mientras no se diga lo contrario, asumimos que siempre se tarda lo mismo ($C$) en ejecutar una tarea determinada. Por tanto, se cumple $C = C_w$.

# Restricciones temporales de una tarea

Para determinar la planificación del sistema necesitamos conocer las **restricciones temporales** de cada tarea del sistema.

Aquí vemos dos ejemplos:

- $C = 2\,\text{ms}$, $T = D = 10\,\text{ms}$.

  Es una tarea periódica que se activa cada $10\,\text{ms}$, y se ejecuta en un tiempo de $2\,\text{ms}$ en el peor de los casos.

- $C = 10\,\text{ms}$, $T = 100\,\text{ms}$; $D = 90\,\text{ms}$.

  Es una tarea periódica que se activa cada $100\,\text{ms}$, se ejecuta en cada activación un máximo de $10\,\text{ms}$, y desde que se inicia la activación hasta que concluye no puede pasar más de $90\,\text{ms}$.

# Restricciones temporales. Factor de utilización.

Las restricciones temporales para un conjunto de $n$ tareas periódicas se especifican dando una tabla con los valores de $T_i$,$C_i$ y $D_i$ para cada una de ellas (con $i$ desde 1 hasta $n$).

- Lógicamente, la $i$-ésima tarea ocupa un fracción $C_i / T_i$ del tiempo total de una CPU.
- El **factor de utilización** $U$ es la suma de esas fracciones para todas las tareas, es decir

$$U = \sum_{i=1}^{n} \frac{C_i}{T_i}$$

- En un hardware con $p$ procesadores disponibles para ejecutar las tareas, si $U > p$ entonces el sistema no es planificable: incluso dedicando a ejecutar tareas el 100 % del tiempo de cada uno de los $p$ procesadores, alguna tarea no podrá acabar en su período.

# Sección 2.
# Esquemas de planificación.

2.1. Planificación Cíclica.
2.2. Planificación con prioridades

# Tipos de esquemas de planificación

Para un sistema **monoprocesador** son las siguientes:

- **Planificación estática *off-line* sin prioridades** (ejecutivo cíclico).
- **Planificación basada en prioridades de tareas**
  - **Estáticas:** prioridades preasignadas, no cambian:
    - **RMS** (*Rate Monotonic Scheduling*): prioridad a la tarea con menor período $T$.
    - **DMS** (*Deadline Monotonic Scheduling*): prioridad a la tarea con menor deadline $D$.
  - **Dinámicas:** prioridades cambiantes durante la ejecución:
    - **EDF** (*Earliest Deadline First*): prioridad a la tarea con el *deadline* más próximo.
    - **LLF** (*Least Laxity First*): prioridad a tarea de menor holgura (tiempo hasta deadline menos tiempo de ejecución restante)

# Subsección 2.1.
# Planificación Cíclica..

# Planificación Cíclica: ciclos principal y secundario

La planificación se basa en diseñar un programa (**ejecutivo cíclico**) que implementa un plan de ejecución (**plan principal**) que garantize los requerimientos temporales.

- El programa es un bucle infinito, tal que cada iteración tiene una duración prefijada, siempre igual. El bucle se denomina **ciclo principal**.
- En cada iteración del bucle principal se ejecuta otro bucle acotado con $k$ iteraciones ($k$ es constante). Cada iteración dura siempre lo mismo y en ella se ejecutan completamente una o varias tareas. A este bucle interno (acotado) se le denomina **ciclo secundario**.
- El entrelazado de las tareas en cada iteración del ciclo principal es siempre el mismo. Una iteración número $i$ del ciclo secundario puede tener un entrelazado distinto a otra iteración número $j$ (con $i$ y $j$ entre 1 y $k$, ambos incluidos).

# Duraciones del ciclo principal y el secundario

Las duraciones de ambos ciclos son valores enteros (se supone que el tiempo se mide en múltiplos enteros de alguna unidad de tiempo)

- La duración del ciclo principal se denomina **hiperperiodo** y se escribe como $T_M$. El hiperperiodo es el mínimo comun múltiplo de los periodos de todas las tareas:

  $$T_M = \text{mcm}(T_1, T_2, \ldots, T_n)$$

  por tanto, los instantes de inicio de cada iteración del ciclo principal coinciden con los instantes en los cuales todas las tareas se activan de nuevo a la vez.
- La duración del ciclo secundario se denomina $T_S$. Lógicamente, se debe cumplir $T_M = k\,T_S$.

# Ejemplo de planificación cíclica.

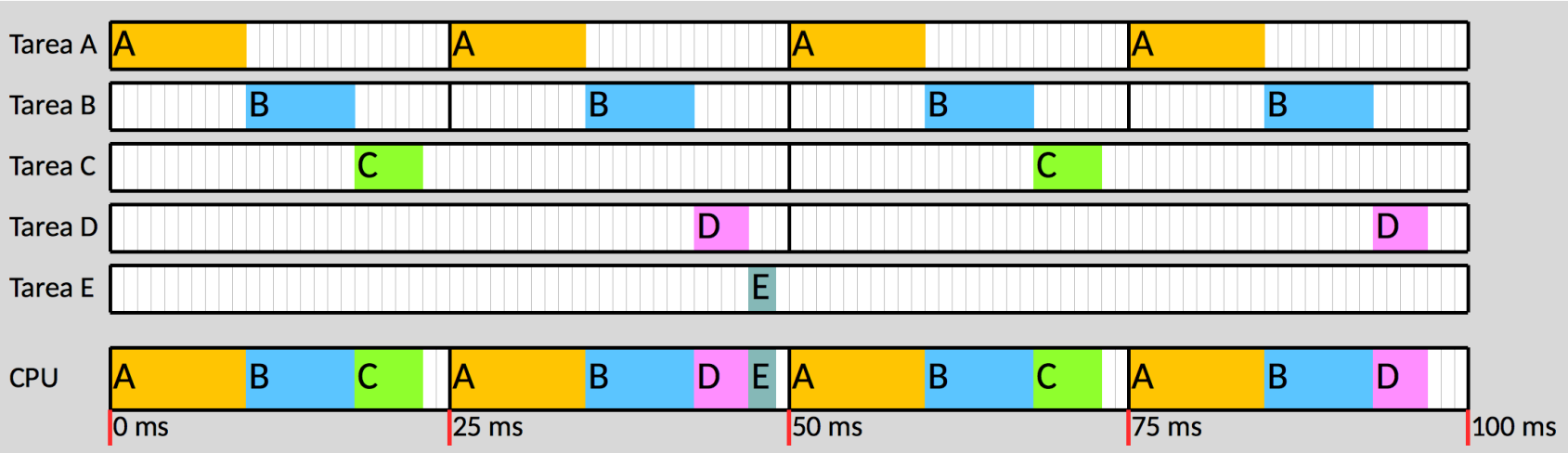
| Tarea | $T$ | $C$ |
|---|---|---|
| $A$ | 25 | 10 |
| $B$ | 25 | 8 |
| $C$ | 50 | 5 |
| $D$ | 50 | 4 |
| $E$ | 100 | 2 |

- El ciclo principal dura 100 ms, ya que

$$T_M = \text{mcm}(25, 25, 50, 50, 100) = 100$$

- Se compone de $k = 4$ ciclos secundarios, con $T_S = 25$ ms.

Una posible planificación con $T_S = 25$ para este sistema es la que se indica a continuación (se asume que $D_i = T_i$)



# Implementación del ejemplo

Cada tarea se ejecuta llamando a un procedimiento homónimo:

```
process EjecutivoCiclico ;
  var inicio : time_point := now() ; { instante inicio ciclo principal }
begin
  while true do begin { ciclo principal }
    { ejecutar cada una de las 4 iteraciones del ciclo secundario }
    A(); B(); C();      sleep_until( inicio+25 );
    A(); B(); D(); E(); sleep_until( inicio+50 );
    A(); B(); C();      sleep_until( inicio+75 );
    A(); B(); D();      sleep_until( inicio+100 );
    inicio = inicio + 100 ; { actualizar instante de inicio de c.p. }
  end
end
```

El tipo **`time_point`** sirve para variables que guardan instantes de tiempo. La función **`now`** devuelve el instante actual. La función **`sleep_until`** deja el proceso bloqueado hasta el instante especificado. Se puede sumar una duración positiva a un instante y obtener otro instante posterior.

# Selección de la duración del ciclo secundario

Para seleccionar un valor apropiado de $T_S$ se deben tener en cuenta estas restricciones y esta sugerencia:

- **Restricciones:** necesariamente se cumplen estas dos:
  - El valor $T_S$ **tiene que ser un divisor** del valor $T_M$
  - El valor de $T_S$ tiene que ser mayor o igual que el tiempo de cómputo ($C_i$) de cualquier tarea, es decir, se cumple:

$$\text{máximo}(C_1, C_2, \ldots, C_n) \quad \leq \quad T_s$$

    (de otra forma una tarea de máxima duración no podría ejecutarse completamente en un ciclo secundario)

- **Sugerencia:** es aconsejable intentar en principio que el ciclo secundario sea menor o igual que el minimo deadline:

$$T_s \quad \leq \quad \text{mínimo}(D_1, D_2, \ldots, D_n)$$

# Propiedades de la Planificación Cíclica

- No hay concurrencia en la ejecución.
  - Cada ciclo secundario es una secuencia de llamadas a **procedimientos**
  - No se necesita un núcleo de ejecución multitarea
- Los procedimientos pueden **compartir datos**.
  - No se necesitan mecanismos de exclusión mutua como los semáforos o monitores
- No hace falta analizar el comportamiento temporal.
  - El sistema es correcto por construcción.

# Problemas de la Planificación Cíclica

- Dificultad para incorporar tareas con periodos largos.
- Las tareas esporádicas son difíciles de tratar.
  - Se puede utilizar un servidor de consulta.
- El plan ciclo del proyecto es difícil de construir.
  - Si los periodos son de diferentes órdenes de magnitud el número de ciclos secundarios se hace muy grande.
  - Puede ser necesario partir una tarea en varios procedimientos.
  - Es el caso más general de sistemas en tiempo real críticos.
- Es poco flexible y difícil de mantener.
  - Cada vez que se cambia una tarea hay que rehacer toda la planificación.

# Subsección 2.2.
# Planificación con prioridades.

# Planificación con prioridades

La **planificación con prioridades** permite solventar los problemas descritos. Cada tarea tiene asociado un valor **entero positivo**, llamado **prioridad** de la tarea:

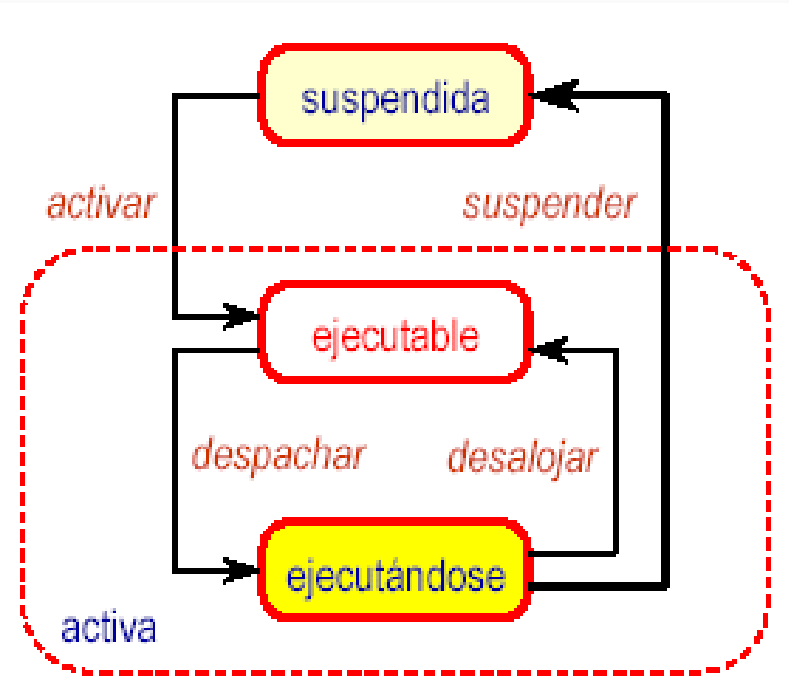
- Es un atributo entero (no negativo) de las tareas, que depende de sus atributos temporales y/o el entrelazamiento entre ellas.
- Por convención se asigna números enteros mayores a tareas más urgentes (más prioritarias).
- La prioridades de cada una de las tareas determinan la selección de la tarea que puede ejecutarse en el (único) procesador cuando este queda libre.
- Las prioridades pueden ser constantes fijadas de antemano (**estáticas**), o bien pueden cambiar con el tiempo (**dinámicas**), en este caso se deben calcular cada vez que hay consultarlas.

# Planificación de las tareas (*scheduling*)

En este tipo de planificaciones debe existir una componente software (llamada **planificador** (***scheduler***)) capaz de:

- Asignar el procesador a una tarea activa o ejecutable (a esto se le llama **despachar** (***dispatch***) la tarea)
- Desasignar la CPU a la tarea en ejecución (**desalojar** la tarea)

- Una tarea puede estar en varios estados (suspendida, ejecutable, ejecutándose)
- Las tareas ejecutables se despachan para su ejecución en orden de prioridad.



# Funcionamiento del planificador

El planificador actua en cada instante de tiempo en el que ocurren alguno de estos dos eventos:

- Una o más de una tarea se activan (pasan al estado ejecutable).
- La tarea en ejecución termina (pasa al estado suspendida).

A continuación, selecciona cualquier tarea $A$ con una prioridad actual $P_A$ máxima entre todas las tareas ejecutables, después

- Si la CPU está libre, $A$ pasa al estado ejecutándose
- Si la CPU está ejecutando una tarea $B$ con prioridad actual $P_B$:
  - Si $P_A > P_B$, la tarea $A$ pasa al estado **ejecutándose** y $B$ pasa al estado **ejecutable** (se dice que $A$ *desplaza a* $B$)
  - Si $P_A \leq P_B$, no hay cambios: $A$ permanece ejecutable y $B$ continua ejecutándose.

Al inicio, **todas las tareas se activan a la vez**.

# Planificación RMS

## RMS: *Rate Monotonic Scheduling*

Es un método de planificación estático on-line con asignación mayor prioridad a las tareas más frecuentes (es decir: con menor periodo $T_i$)

- A cada tarea $i$ se le asigna una prioridad ($P_i$) basada en su período ($T_i$): cuanto menor sea el periodo (mayor frecuencia) mayor será su prioridad.

$$\forall i, j \ : \quad T_i < T_j \implies P_i > P_j$$

- Esta asignación de prioridades es optima en el caso de que todas las tareas sean periódicas, y el plazo de respuesta máxima $D$ coincida con el periodo.

# Ejemplo de planificación RMS (1)

Supongamos que tenemos un conjunto de tres tareas (1,2 y 3) para las cuales se han fijado las siguientes restricciones temporales:

| Tareas | $C$ | $D$ | $T$ |
|---|---|---|---|
| 1 | 10 | 30 | 30 |
| 2 | 5 | 40 | 40 |
| 3 | 9 | 50 | 50 |

La aplicación del método de planificación RMS indica que:

- la tarea 1 tiene la mayor prioridad,
- la tarea 2 tiene prioridad intermedia
- la tarea 3 tiene la menor prioridad.

# Tests de Planificabilidad

Los tests de planificabilidad permiten determinar si el conjunto de tareas del sistema es planificable según un algoritmo de planificación antes de su ejecución. Existen diversos tipos de tests aplicables según el algoritmo de planificación:

- **Tests de planificabilidad suficientes**:
  - En caso de éxito en la aplicación del test el sistema es planificable.
  - En caso contrario no tenemos información: podría ser planificable o no serlo.
- **Tests de planificabilidad exactos**: (suficiente y necesario)
  - En caso de éxito en la aplicación del test el sistema es planificable.
  - En caso contrario el sistema no es planificable: habrá fallos inevitablemente (no se cumplirá algún deadline).

# Test de Liu & Layland

Es un test suficiente, determina la planificabilidad para un sistema de $n$ tareas periódicas independientes con prioridades RMS.

- Se usan dos valores reales, el **factor de utilización** ($U$), y el **factor de utilización máximo** ($U_0(n)$) (para $n$ tareas). Se definen así:

$$U \equiv \sum_{i=1}^{n} \frac{C_i}{T_i} \qquad U_0(n) \equiv n\left(\sqrt[n]{2}-1\right)$$

- El valor de $U_0(1)$ es 1. A partir de ahí decrece al crecer $n$, hasta el límite (para $n$ infinito), que es $\ln 2 \approx 0.693147 \leq U_0(n), \forall n$.
- Un sistema **pasa el test** si el factor de utilización es menor o igual que el máximo posible:

$$U \leq U_0(n)$$

en ese caso el sistema es planificable. En caso contrario, no se puede afirmar nada.

# Ejemplo RMS cumpliendo el Test Liu & Layland

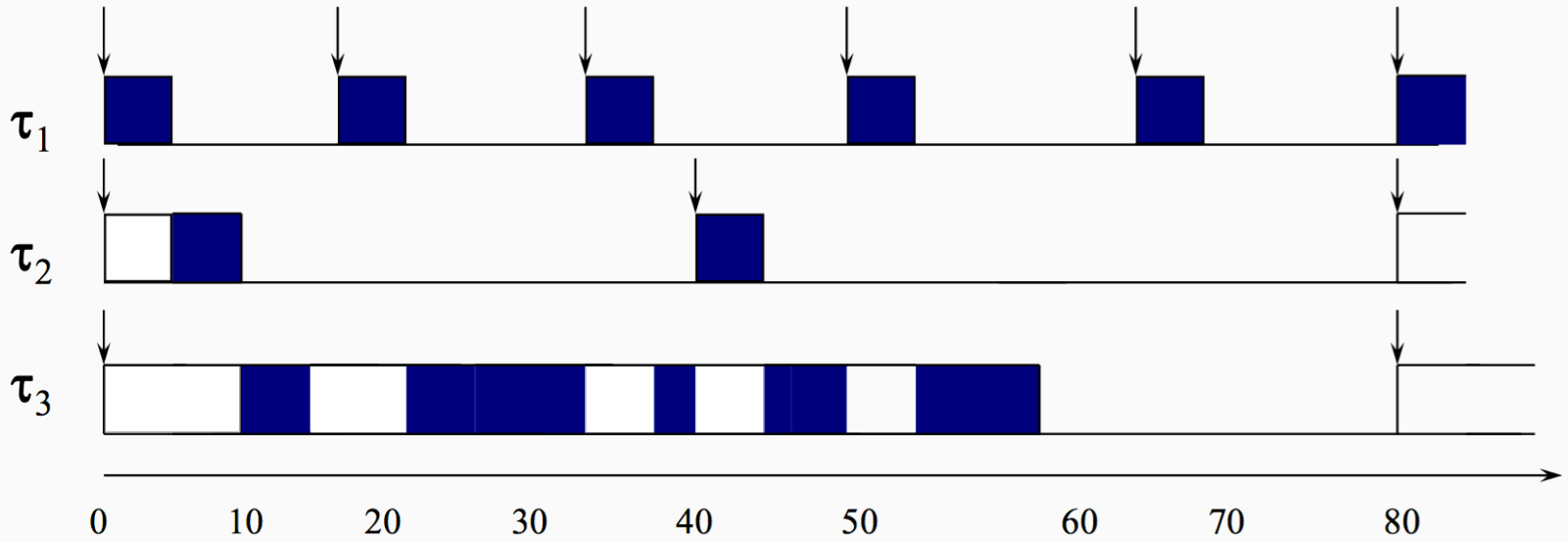
Supongamos ahora que tenemos estos datos de entrada:

| Tareas | $C$ | $D$ | $T$ | $C/T$ |
|---|---|---|---|---|
| $\tau_1$ | 4 | 16 | 16 | 0,250 |
| $\tau_2$ | 5 | 40 | 40 | 0,125 |
| $\tau_3$ | 32 | 80 | 80 | 0,400 |

El sistema **pasa** el Test de Liu & Layland, ya que:

$$U = 0,775 \leq 0,779 = U_0(3)$$

El cronograma de este sistema es este:



# Ejemplo RMS no cumpliendo el Test Liu & Layland

Supongamos ahora que tenemos estos datos de entrada para un sistema:

| Tareas | $C$ | $D$ | $T$ |
|---|---|---|---|
| Tarea 1 | 10 | 30 | 30 |
| Tarea 2 | 10 | 40 | 40 |
| Tarea 3 | 10 | 50 | 50 |

Podemos calcular $U$ y compararlo con $U_0(3)$:

$$U = \sum_{i=1}^{n} \frac{C_i}{T_i} = \frac{10}{30} + \frac{10}{40} + \frac{10}{50} = 0,783333 \not\leq 0,779 = U_0(3)$$

es decir:

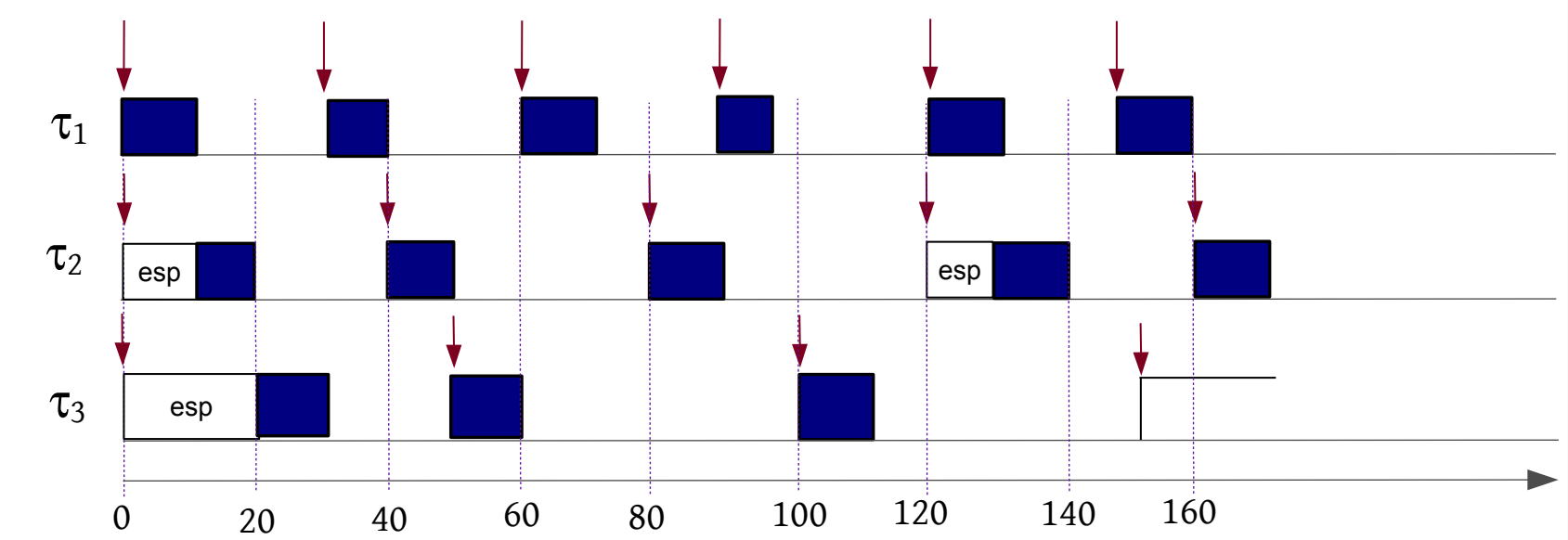
- El sistema **no pasa el test**.
- **No podemos decir si es planificable o no lo es.**
- Para saberlo, podemos analizar el cronograma.

# Ejemplo RMS sin cumplir test: cronograma

Para analizar la planificabilidad del sistema con dichas restricciones, hay que hacer el cronograma y verificar que:

- para cada tarea $i$, se cumple el plazo de respuesta: $R < D_i$
- esto se debe verificar para un hiperperiodo (después se repite).

En el ejemplo, se debería hacer el cronograma completo hasta $t = 600$ ($= \text{mcm}\{30, 40, 50\}$), aquí vemos una primera parte (hasta $t = 160$):



# Planificación EDF (*Earliest Deadline First*)

Es un esquema de planificación con **prioridades dinámicas**. Se denomina **EDF** , o bien **primero el más urgente**:

> *La asignación de prioridad se establece una prioridad más alta a la que se encuentre más próxima a su plazo de respuesta máxima (deadline).*

Características:

- En caso de igualdad se hace una elección no determinista de la siguiente tarea a ejecutar.
- Es un algoritmo de planificación dinámica, dado que la prioridad de cada tarea cambia durante la evolución del sistema.
- Es más óptimo porque no es necesario que las tareas sean periódicas.
- En menos pesimista que RMS.

# Test de planificabilidad para EDF

El test de planificación de Liu & Layland se puede aplicar también para el caso de planificación EDF.

- En este caso, un sistema pasa el test si el factor de utilización es menor o igual a la unidad:

$$U = \sum_{i=1}^{n} \frac{C_i}{T_i} \leq 1$$

- Esto implica que la planificación EDF puede aplicarse a más sistemas que la planificación RMS (la cota máxima de $U$ es mayor en EDF que en RMS)
- El test ahora **es exacto** (necesario y suficiente):
  - Si un sistema pasa el test, es planificable con EDF.
  - Si un sistema no pasa el test, no es planificable con EDF.

# Ejemplo de planificación EDF

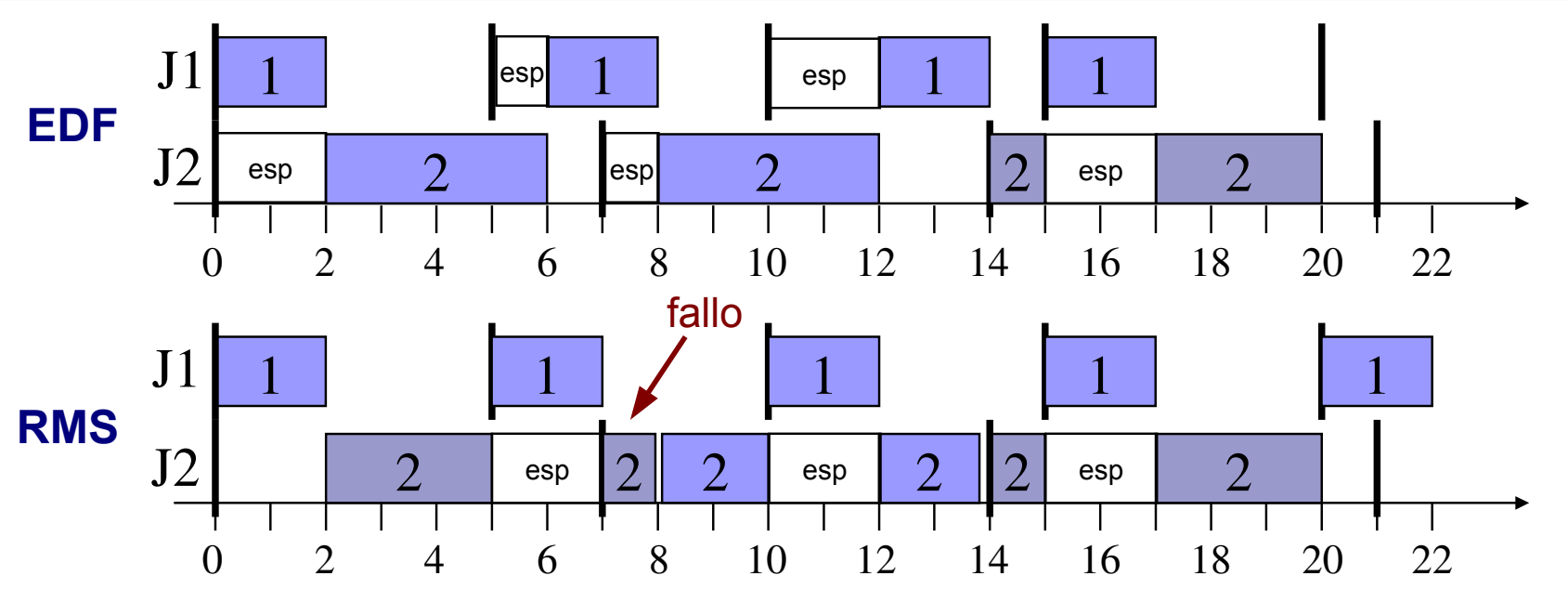
Vemos un sistema de ejemplo, y aplicamos el test

| Tareas | $C$ | $D$ | $T$ |
|---|---|---|---|
| J1 | 2 | 5 | 5 |
| J2 | 4 | 7 | 7 |

**Pasa** el test de Liu & Layland (EDF):

$$U = \frac{2}{5} + \frac{4}{7} = 0,97 \leq 1$$

Se verifica que el sistema **es planificable con EDF**, pero no con RMS:



# Bibliografía del tema 4.

Para más información, ejercicios, bibliografía adicional, se puede consultar:

4.1. **Concepto de sistemas de tiempo real. Medidas de tiempo y modelo de tareas.**
Burns (2005), capítulos 1, 9 y 10 (en Inglés).

4.2. **Esquemas de planificación.**
Burns (2002), capítulo 13 (en Español).
Burns (2005), capítulos 11 y 12 (en Inglés).

Fin de la presentación.