

Tema 6

Planificación de Tareas



Objetivos

1. Conocer el concepto de **scheduling** en un STR
2. Conocer cómo caracterizar las tareas para analizar sus requerimientos temporales
3. Conocer el modelo simple de procesos para facilitar la predicción del comportamiento de un sistema concurrente en el **peor caso**
4. Conocer y saber utilizar prioridades de tareas en Ada y sus políticas de *dispatching*



Índice

1. **Requerimientos en tareas de tiempo real**
2. *Scheduling*
3. Modelo simple de procesos
4. Caracterización de las tareas
5. Tipos de planificadores
6. Prioridades de tareas en Ada
7. *Dispatching* en Ada



Tipos de requerimientos

■ **de tiempo**

- Plazos de ejecución
- Instantes de activación

■ **de relaciones de precedencia**

- Grafo acíclico dirigido

■ **de exclusión mutua en recursos compartidos**

- Sincronización de tareas



Requerimientos de tiempo

- Completar el trabajo antes de un plazo (**deadline**)
- Procesar **actividades periódicas**:
 - Requeridas a intervalos regulares
- Responder a **eventos aperiódicos**:
 - Requeridos a intervalos irregulares
 - **eventos esporádicos**: intervalo mínimo entre eventos consecutivos



Cumplir requerimientos temporales

- **Todas** las acciones deben ocurrir dentro del plazo especificado
 - STR **crítico** (*hard*)
 - Una sola respuesta tardía puede tener consecuencias fatales
- Se pueden perder plazos **ocasionalmente**
 - STR **flexible** (*soft*)
 - El valor o utilidad de la respuesta decrece con el tiempo
 - STR **firme** (*firm*)
 - Una respuesta tardía no tiene valor

Garantizar requerimientos en el **peor caso posible**



Índice

1. Requerimientos en tareas de tiempo real
2. **Scheduling**
3. Modelo simple de procesos
4. Caracterización de las tareas
5. Tipos de planificadores
6. Prioridades de tareas en Ada
7. *Dispatching* en Ada



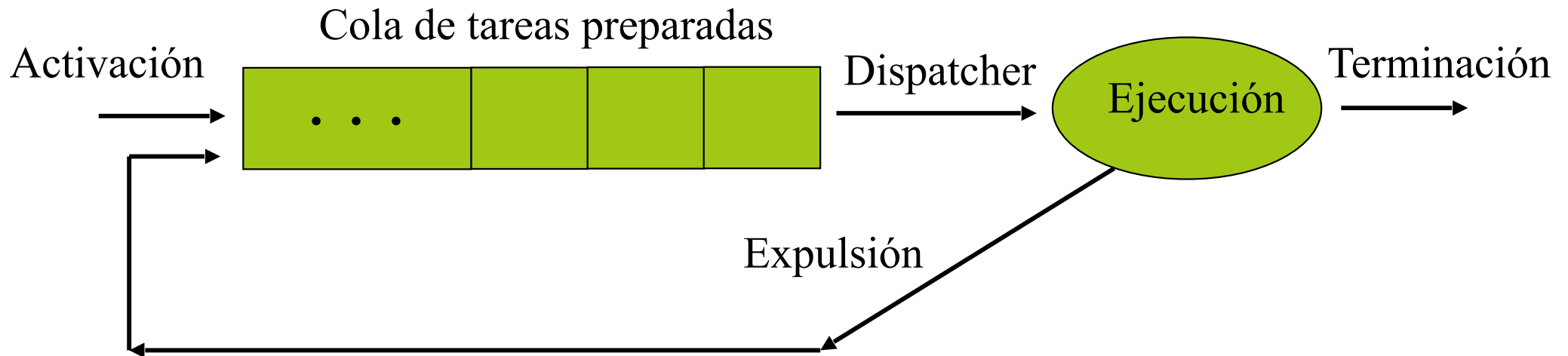
Concepto de scheduling

- El **scheduling** proporciona:
 - un **PLANIFICADOR**
 - algoritmo que determina el orden del uso de recursos del sistema (en particular, la CPU)
 - restringe el indeterminismo de un sistema concurrente (sistema predecible)
 - un **TEST DE PLANIFICABILIDAD**
 - método para determinar si el sistema cumplirá con las restricciones temporales establecidas al usar un determinado planificador



Planificador (*dispatcher*)

- Cuando se trata de asignar la CPU, el planificador determina el orden de ejecución de los procesos



Tests de planificabilidad

- Test basado en **factores de utilización**
- Test basado en **factores de carga**
- Test basado en **tiempos de respuesta**
- Test basado en el **cronograma** (diagrama de tiempo)



Formulación general del problema de *scheduling*

- Un conjunto de n tareas
- Un conjunto de m procesadores
- Un conjunto de r tipos de recursos compartidos
- Un grafo acíclico dirigido con las relaciones de precedencia
- Restricciones de tiempo asociadas a las tareas

Problema **NP-Completo**



Simplificaciones del problema

- ❑ Sistema con un único procesador
- ❑ Modelo expulsivo (*preemptive*)
- ❑ Uso de prioridades fijas
- ❑ Eliminar relaciones de precedencia
- ❑ Eliminar recursos compartidos
- ❑ Asumir activación simultánea de las tareas
- ❑ Conjuntos homogéneos de tipos de tareas (sólo periódicas o sólo aperiódicas)



Índice

1. Requerimientos en tareas de tiempo real
2. *Scheduling*
3. **Modelo simple de procesos**
4. Caracterización de las tareas
5. Tipos de planificadores
6. Prioridades de tareas en Ada
7. *Dispatching* en Ada



Caracterización

- Conjunto con un número fijo de procesos
- Procesos periódicos con periodos T_i conocidos
- Procesos independientes
- Se ignoran las sobrecargas (*overhead*) del núcleo de ejecución (se asume que tienen coste cero)
- Procesos con un *deadline* igual a su periodo $T_i=D_i$
- Tiempo de ejecución C conocido y fijo
- Los procesos se ejecutan en un único procesador



Tiempos de cómputo

- Hay dos formas de obtener el valor C de una tarea:
 - **Medida del tiempo de ejecución**
 - Dificultad para determinar cuándo se produce el peor caso
 - **Análisis del código ejecutable**
 1. Se descompone el código en un grafo de bloques secuenciales
 2. Se calcula el tiempo de ejecución de cada bloque
 3. Se busca el camino más largo
 - La estimación de C puede ser muy pesimista
 - Es difícil tener en cuenta los efectos de los dispositivos hardware (cachés, pipelines, etc.)
- Para poder calcular el tiempo de cómputo hay que evitar utilizar estructuras con un tiempo de cómputo no acotado (bucles no acotados, recursión no acotada, tareas dinámicas)



Índice

1. Requerimientos en tareas de tiempo real
2. *Scheduling*
3. Modelo simple de procesos
4. **Caracterización de las tareas**
5. Tipos de planificadores
6. Prioridades de tareas en Ada
7. *Dispatching* en Ada



Notación estándar

C: Tiempo de ejecución en el peor de los casos

D: Plazo de respuesta (*deadline*) relativo

I: Tiempo de interferencia

N: Número de procesos en el sistema

P: Prioridad

R: Tiempo de respuesta en el peor de los casos

T: Periodo de activación

U: Factor de utilización del procesador de una determinada tarea

$$u = \frac{C}{T}$$



Otros parámetros

r_k : Instante de la activación (*release time*) (k+1)-ésima

- ▣ **r_0** : Instante de la primera activación del proceso
- ▣ En una tarea periódica:

$$r_k = r_0 + k * T$$

d_k : *deadline* absoluto de la (k+1)-ésima activación

$$d_k = r_k + D$$

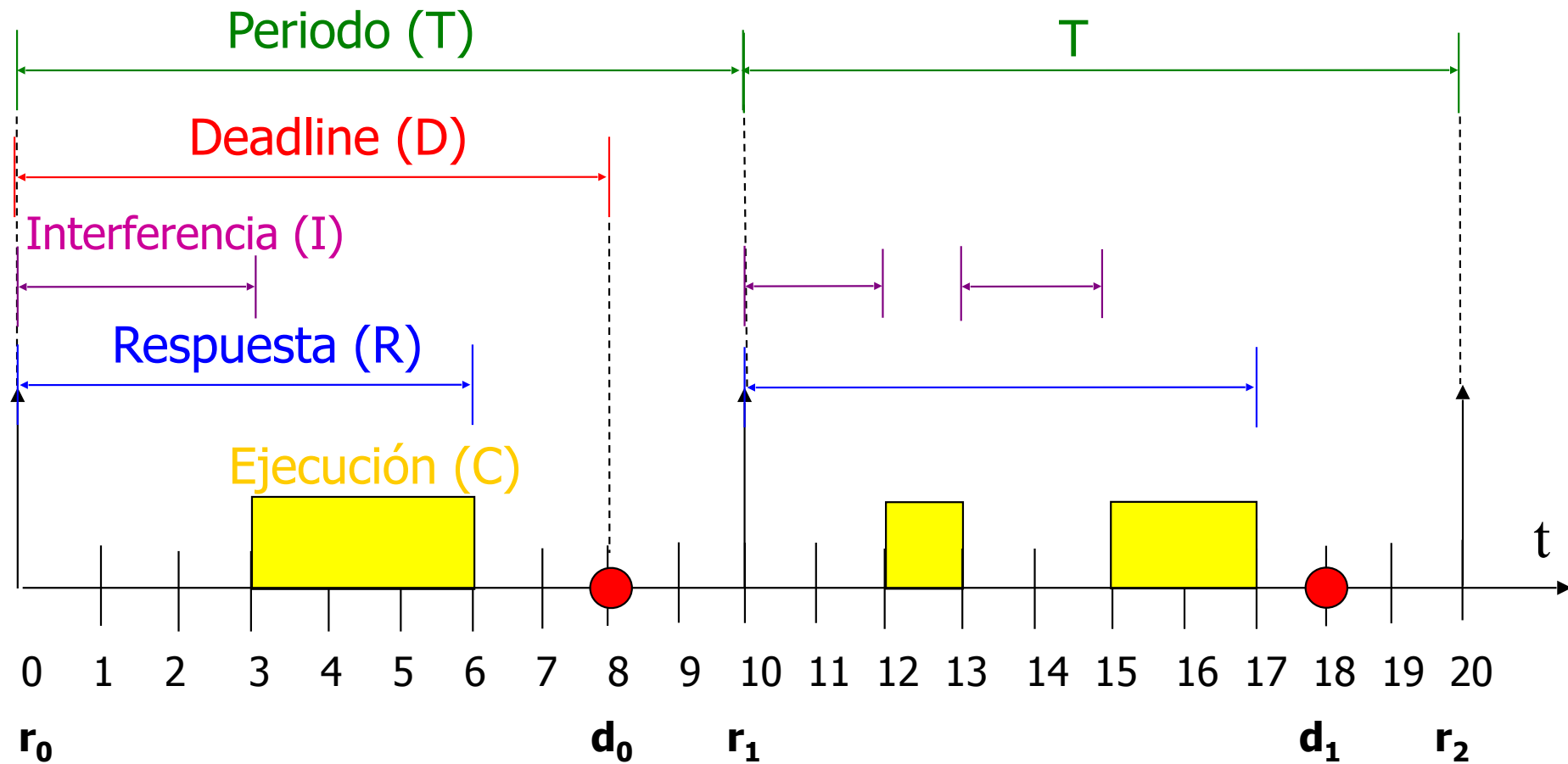
ch : Factor de carga del procesador de una determinada tarea

$$ch = \frac{C}{D}$$

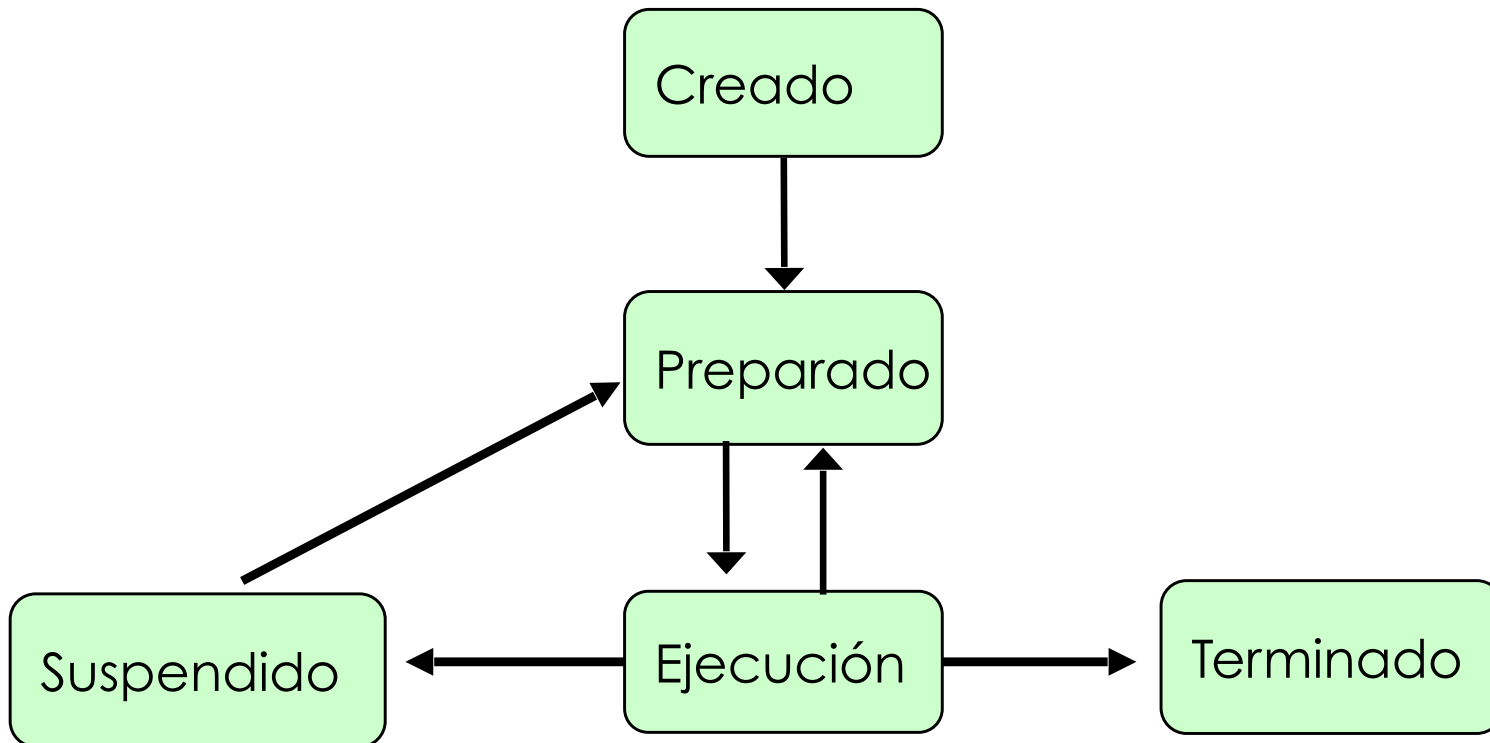


4. Caracterización de las tareas

Diagrama de tiempo



Estados de un proceso

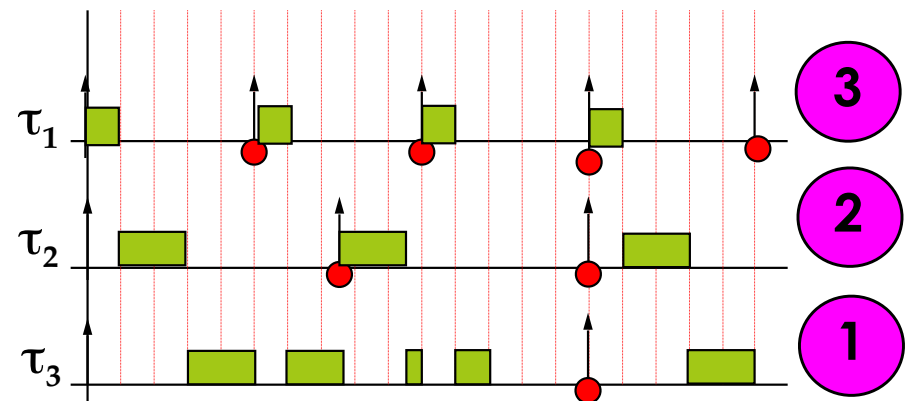
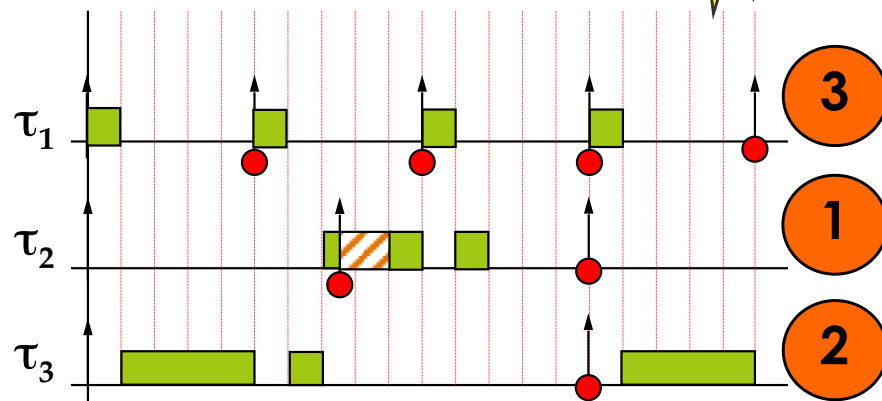
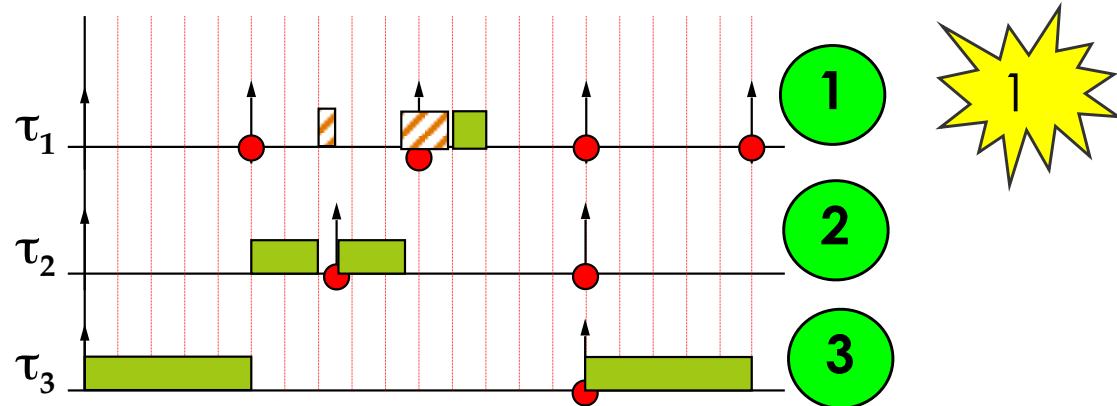


4. Caracterización de las tareas

Ejemplos de cronograma

Prioridad $\rightarrow P_i$

Tarea _i	C_i	T_i	D_i
τ_1	20	100	100
τ_2	40	150	150
τ_3	100	300	300



Índice

1. Requerimientos en tareas de tiempo real
2. *Scheduling*
3. Modelo simple de procesos
4. Caracterización de las tareas
5. **Tipos de planificadores**
6. Prioridades de tareas en Ada
7. *Dispatching* en Ada



Clasificaciones de planificadores

- Expulsivo (*preemptive*) / No expulsivo
- Estático / Dinámico
- Basado en garantía / Del mejor esfuerzo
- ...

En cualquiera de las categorías de planificadores, es importante conocer, si existe, su **planificador óptimo**



Expulsivo vs No expulsivo

■ **Expulsivo (preemptive)**

- La tarea que está ejecutándose puede ser interrumpida en cualquier instante para asignar el procesador a otra tarea
- Pueden evitar incumplimientos de plazo
- Permiten a los procesos de mayor prioridad ser más reactivos

■ **No expulsivo**

- Una tarea una vez que comienza su ejecución, no abandona el procesador hasta que termina
- En arquitecturas con un solo procesador no se requiere mecanismos de exclusión mutua para acceder a recursos compartidos



Estático vs Dinámico

■ Estático

- Las decisiones de *scheduling* se basan en parámetros fijos, asignados a las tareas antes de su activación (*off-line*)
- La planificación generada es almacenada en una tabla y después ejecutada por el *dispatcher*
- No sobrecarga el sistema en tiempo de ejecución

■ Dinámico

- Las decisiones de *scheduling* se basan en parámetros dinámicos que pueden cambiar durante la evolución del sistema (*on-line*)
- Se obtienen planificaciones más flexibles adecuadas para tareas aperiódicas y situaciones excepcionales de sobrecarga



Basado en garantía vs del mejor esfuerzo

■ Basado en garantía

- En STR críticos
- No se tolera ningún incumplimiento de plazo
- Se utiliza test de aceptación para aceptar una nueva tarea
- Se pueden rechazar tareas innecesariamente al asumir el peor caso

■ Del mejor esfuerzo

- En STR no críticos
- Se tolera algún incumplimiento de plazo
- Intenta “hacer lo mejor posible” para cumplir los plazos
- Mejora los tiempos de respuesta promedios
- Se rechazan tareas en condiciones reales de sobrecarga



Planificador óptimo

En cualquiera de las categorías de planificadores, se dice que un planificador es óptimo dentro de esa categoría cuando ...

es capaz de encontrar una planificación factible de cualquier conjunto de tareas planificable



Consecuencias de utilizar un planificador óptimo

- Si un conjunto de tareas no es planificable con el planificador óptimo, podemos afirmar que no es planificable, ya que no lo será con ningún otro planificador (de esa misma categoría)
- Si un conjunto de tareas no es planificable con un planificador no óptimo, no podemos garantizar que no sea planificable



Índice

1. Requerimientos en tareas de tiempo real
2. *Scheduling*
3. Modelo simple de procesos
4. Caracterización de las tareas
5. Tipos de planificadores
6. **Prioridades de tareas en Ada**
7. *Dispatching* en Ada



Paquete System

```
package System is
    . . .
    subtype Any_Priority is Integer range intervalo;
    subtype Priority is Any_Priority
        range Any_Priority'First..valor;
    subtype Interrupt_Priority is Any_Priority
        range Priority'Last+1..Any_Priority'Last;
    Default_Priority : constant Priority :=
        (Priority'First + Priority'Last) / 2;

    private
        . . .
end System;
```



Prioridad base

- Es la prioridad con la que se ejecuta inicialmente la tarea
- Se establece con el **pragma Priority**

```
task Controlador is  
    pragma Priority(48);  
end Controlador;
```

```
task type T_Controlador is  
    pragma Priority(48);  
end T_Controlador;
```

```
task type T_Controlador (Task_Priority : System.Priority) is  
    pragma Priority(Task_Priority);  
end T_Controlador;
```

- Si una tarea no utiliza este pragma, su prioridad base es igual a la de su tarea padre



Prioridad activa

- Una tarea puede heredar una prioridad más alta que su prioridad base

Duración herencia	Prioridad heredada
Activación de la tarea	La del padre
Ejecución de una operación protegida	Techo de prioridad del objeto protegido
Ejecución de una cita	la del cliente

$\text{prioridad Activa} = \text{Mayor}(\text{prioridad Base}, \text{prioridad Heredada})$



Políticas de gestión de colas

- Las colas de *entries* se pueden ordenar

- Por orden de llegada

```
pragma Queuing_Policy(FIFO_Queueing);
```

- Por orden de prioridades

```
pragma Queuing_Policy(Priority_Queueing);
```



Prioridad dinámica

- La prioridad base de una tarea se puede cambiar dinámicamente

```
Ada.Dynamic_priorities.Set_Priority(...);
```

```
package Ada.Dynamic_Priorities is  
  procedure Set_Priority(Priority: in System.Any:Priority;  
                        T: in Ada.Task_Identification.Task_id :=  
                          Ada.Task_Identification.Current_Task);  
  
  function Get_Priority(T: Ada.Task_Identification.Task_id :=  
                       Ada.Task_Identification.Current_Task);  
  
end Ada.Dynamic_Priorities;
```

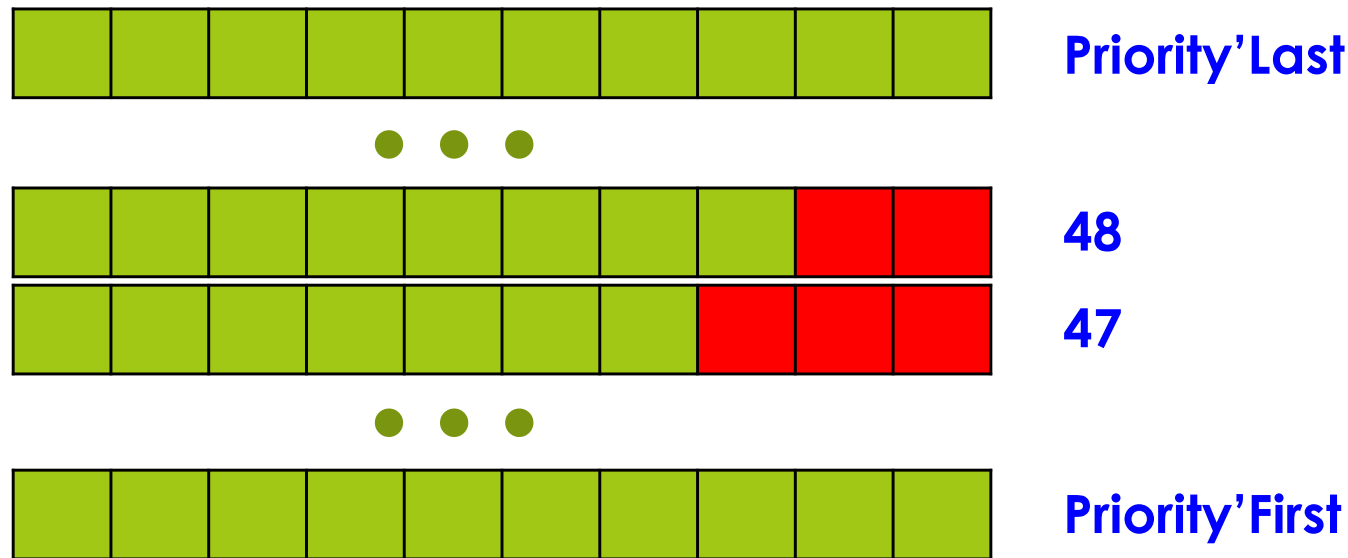


Índice

1. Requerimientos en tareas de tiempo real
2. *Scheduling*
3. Modelo simple de procesos
4. Caracterización de las tareas
5. Tipos de planificadores
6. Prioridades de tareas en Ada
7. ***Dispatching en Ada***



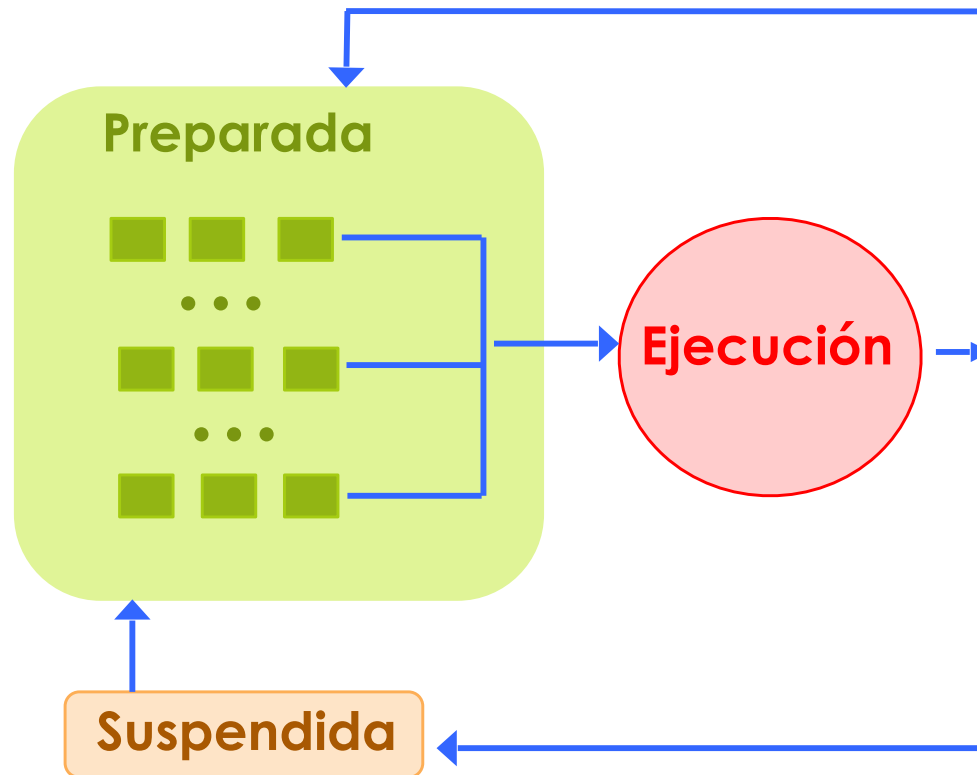
Colas de prioridades



- Una cola para cada nivel de **prioridad activa**
- Cada cola puede tener su política de *dispatching*
- Dos niveles de dispatching:
 1. Elegir la cola no vacía de mayor prioridad
 2. Elegir la tarea de esa cola de acuerdo a su política particular de *dispatching*



Modelo de *dispatching*



- Cuando se llega a un **punto de *dispatching***, la tarea en ejecución:
 - Se coloca al final de su cola de prioridad activa, si ha sido **suspendida**
 - Se coloca al principio de su cola de prioridad activa, si ha sido **desalojada**



Puntos de *dispatching*

- Determinan **cuándo** el *dispatcher* debe elegir una tarea en estado preparada para que pase a ejecución
 - Cuando se bloquea
 - Cuando se suspende
 - Cuando se reanuda (deja de estar suspendida)
 - Cuando termina
 - Otros puntos específicos de determinadas políticas de dispatching, tales como agotar un *quantum*



Pragmas de Políticas de *dispatching*

- Aplicar la política de dispatching a todas las tareas del sistema

```
pragma Task_Dispatching_Policy(policy_identifier)
```

- Aplicar la política de dispatching a un determinado rango de prioridades

```
pragma Priority_Specific_Dispatching (policy, firts_priority,last_priority)
```



Políticas de *dispatching*

■ FIFO_Within_Priorities

- cola FIFO
- una tarea puede expropiar (preempt) la CPU a otra de menor prioridad

■ Non_Preemptive_FIFO_Within_Priorities

- no hay expropiación por tareas de mayor prioridad

■ Round_Robin_Within_Priorities

- Turno circular
- Pueden expropiarse cuando se agota un determinado *quantum*

■ EDF_Across_Priorities

- Se elige la tarea con menor *deadline* absoluto



Conclusiones

- Un STR debe ser predecible, es decir, se debe poder determinar con antelación las respuestas del mismo
- El indeterminismo de un sistema concurrente se reduce con requerimientos temporales de las tareas
- En el *scheduling* de un STR se debe proporcionar un planificador y un test de planificabilidad
- Para hacer tratable computacionalmente el problema del scheduling se han de asumir determinadas simplificaciones a su formulación general
- El lenguaje Ada proporciona librerías para gestionar prioridades de tareas y gestionar el *dispatcher*



Bibliografía Recomendada

Sistemas de tiempo real y lenguajes de programación (**3ª edición**)

Alan Burns and Andy Wellings

Addison Wesley (2002)

 Capítulo 13 (Apartados: 13.1; 13.3.2; 13.14.1)

Hard real-time computing systems (**Second edition**)

Giorgio C. Buttazzo


Kluwer Academic Publishers (2004)

 Capítulo 2

Scheduling in real-time systems

Cottet Francis and others.

Wiley (2002)

 Capítulo 1 (Apartado 1.2)



Otras fuentes de información

Manual de Referencia de Ada2005

 Anexos: D.1 ; D.2.1; D.2.2; D.2.3; D.2.4; D.2.5

