Repaso de
Sistemas de Tiempo Real

Introducción al Tiempo Real

Tomado de:

Introducción a los sistemas de tiempo real. Juan Antonio de la Puente. DIT/UPM

http://web.dit.upm.es/~jpuente/strl/transparencias/01_Introduccion.pdf

Sistemas de Tiempo Real

Un sistema de Tiempo Real

- Interacciona repetidamente con su entorno físico
- Responde a los estímulos que recibe del mismo **dentro de un plazo de tiempo determinado**

- Para que el funcionamiento del sistema sea aceptable no basta con que las acciones sean correctas, sino que tienen que ejecutarse **dentro del intervalo de tiempo especificado**
  - El tiempo en que se ejecutan las acciones del sistema es significativo.
  - Tiempo Real no es sinónimo de sistema de rápida respuesta

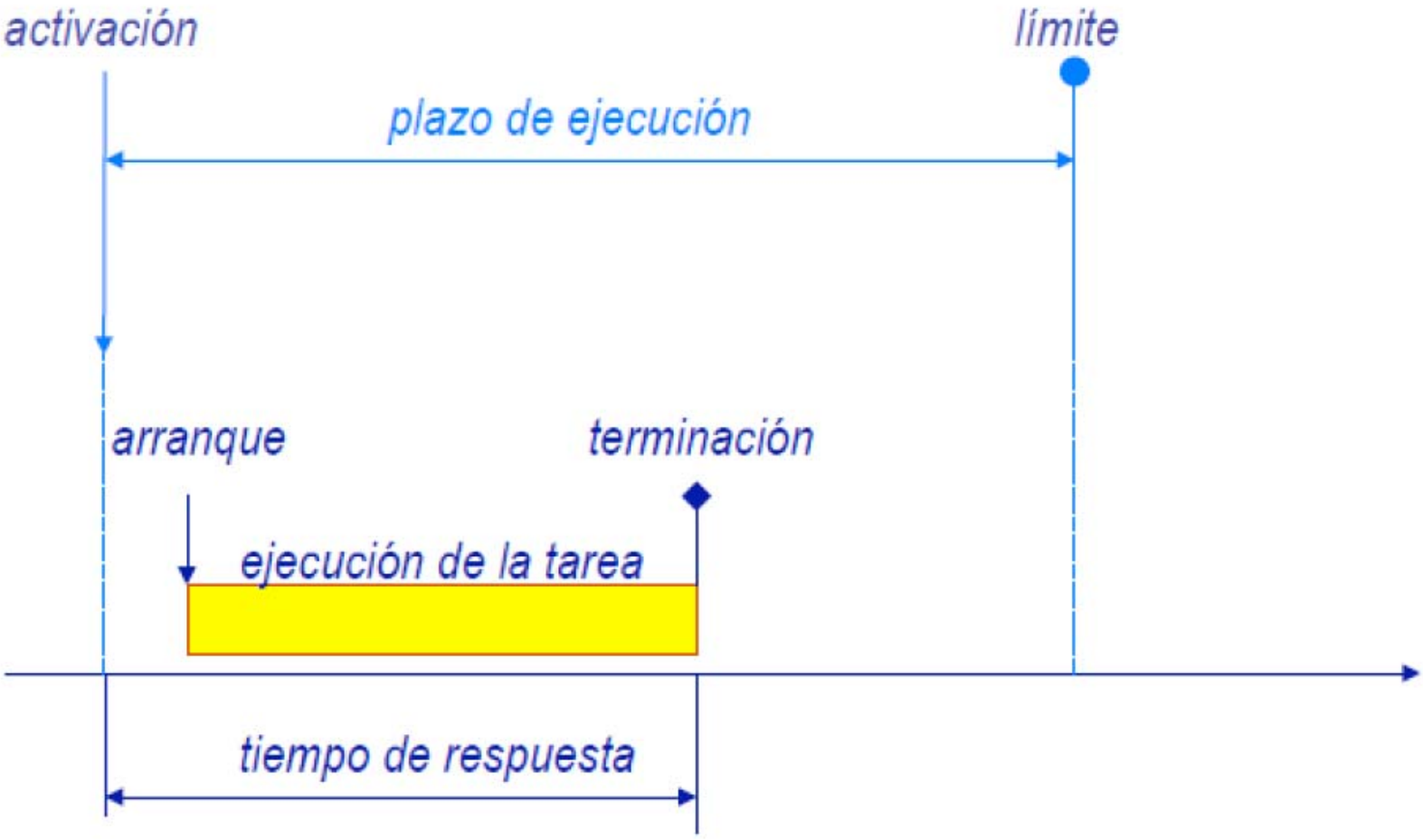
Tareas de Tiempo Real

- Las actividades de un sistema de tiempo real se llaman **tareas**

- Tienen varios tipos de propiedades:
  - **Funcionales**: qué hacen
  - **Temporales**: cuándo lo hacen
  - **Fiabilidad, Seguridad**, etc.

- El comportamiento temporal de las tareas se especifica mediante sus atributos temporales
  - Cuándo se ejecutan: esquema de activación
  - Qué plazo tienen para ejecutar cada acción

Ejecución de Tarea de Tiempo Real



Atributos Temporales de las tareas

- **Activación**
  - *Periódica*: a intervalos regulares, con período T
  - *Aperiódica*: cada vez que ocurre un evento determinado
    - *Esporádica*: separación mínima entre activaciones T
    - *Estocástica*: periodos irregulares, a rachas
- **Plazo de respuesta**
  - *Absoluto*: tiempo límite para terminar
  - *Relativo*: intervalo desde la activación

Objetivo común:

- Garantizar que la ejecución de cada tarea termina dentro del plazo esperado

Tipos de Requisitos Temporales

- **Tiempo real estricto** (*hard real-time*)
  - Todas las acciones deben terminar dentro del plazo especificado.
    - Ejemplo: control de frenado

- **Tiempo real flexible** (*soft real-t*ime)
  - Se pueden perder plazos de vez en cuando.
  - El valor de la respuesta decrece con el tiempo.
    - Ejemplo: sistemas de adquisición de datos

- **Tiempo real firme** (*firm real-time*)
  - Se pueden perder plazos ocasionalmente.
  - Una respuesta tardía no tiene valor.
    - Ejemplo: sistemas multimedia

Heterogeneidad:

- En un mismo sistema puede haber tareas con distintos tipos de requisitos temporales

Características de los sistemas de Tiempo Real

- Suelen ser de gran tamaño y complejidad
  - Algunos STR tienen millones de líneas de código
  - La variedad de funciones aumenta la complejidad incluso en sistemas relativamente pequeños
- Concurrencia: Simultaneidad de acciones
  - Los dispositivos físicos controlados funcionan al mismo tiempo
  - Las tareas que los controlan actúan concurrentemente
- Dispositivos de entrada y salida especiales
  - Los manejadores de dispositivos forman parte del software de aplicación

Características de los sistemas de Tiempo Real

- **Seguridad y fiabilidad**
  - Sistemas críticos: fallos con consecuencias graves
    - Pérdida de vidas humanas
    - Pérdidas económicas
    - Daños medioambientales

- **Determinismo temporal**
  - Acciones en intervalos de tiempo determinados
  - Es fundamental que el comportamiento temporal de los STR sea determinista o, al menos, previsible
    - No hay que confundirlo con la necesidad de que sea eficiente
    - El sistema debe responder correctamente en todas las situaciones
    - En los sistemas de tiempo real estricto hay que prever el comportamiento en el peor caso posible

Determinismo Temporal

- Las tareas de tiempo real se ejecutan concurrentemente
  - Hebras (*threads*) o mecanismos similares
- La planificación del uso del procesador debe permitir acotar el tiempo de respuesta
  - Prioridades y otros métodos de planificación
- Es conveniente analizar el comportamiento temporal del sistema antes de probarlo
  - Las pruebas (tests) no permiten asegurar el comportamiento en el peor caso
  - Se requiere el uso de métodos formales (ej. Model Checking).

Sistemas distribuidos

- Las tareas de tiempo real se ejecutan en varios computadores comunicados mediante una red
  - Más potencia de cálculo
  - Mayor fiabilidad (redundancia)

- En los sistemas críticos el tiempo de comunicación debe estar acotado
  - Redes y protocolos específicos
    - Paso de testigo, prioridades (CAN), TDMA

- El análisis temporal es más complicado que en los sistemas monoprocesadores

Tipos de Sistemas de Tiempo Real

- Según las propiedades del sistema controlado
  - **Sistemas críticos** (*hard RTS*) y **sistemas no-crític**os (*soft RTS*)
  - **Sistemas con parada segur**a (*fail safe*) y **sistemas con degradación aceptable** (*fail soft*)

- Según las propiedades del sistema de tiempo real
  - **Sistemas con tiempo de respuesta garantiza**do (guaranted response) y **sistemas que hacen lo que pueden** (best effort)
  - **Sistemas con recursos suficientes** (*resource-adequate*) y **sistemas con recursos insuficientes** (*resource-inadequate*)
  - **Sistemas dirigidos por eventos** y **sistemas dirigidos por tiempo**

Sistemas críticos y acríticos

Se distinguen por sus requisitos temporales y de fiabilidad

Sistemas críticos
(*hard real-time systems*)

- Plazo de respuesta **estricto**
- Comportamiento temporal determinado por el entorno
- Comportamiento en sobrecargas **predecible**
- Requisitos de seguridad **críticos**
- **Redundancia activa**
- Volumen de datos reducido

Sistemas no-críticos
(*soft real-time systems*)

- Plazo de respuesta **flexible**
- Comportamiento temporal determinado por el **computador**
- Comportamiento en sobrecargas **degradado**
- Requisitos de seguridad **no-críticos**
- **Recuperación de fallos**
- Gran volumen de datos

Sistemas con parada segura y con degradación aceptable

Se distinguen por su comportamiento en caso de avería

Sistemas con parada segura

(*fail-safe*)

- Detención en estado seguro
- Probabilidad de detección de fallos elevada

Sistemas con degradación aceptable

(*fail-soft*)

- Funcionamiento con pérdida parcial de funcionalidad o prestaciones

Sistemas con respuesta garantizada y que hacen lo que pueden

Se distinguen por su grado de determinismo temporal

Sistemas con respuesta garantizada

(*guaranteed response systems*)

- Comportamiento temporal garantizado analíticamente
- Hace falta caracterizar con precisión la carga máxima y los posibles fallos

Sistemas que hacen lo que pueden

(*best-effort systems*)

- Comportamiento temporal de tipo “lo mejor que se pueda”
- No se hace una caracterización precisa de carga y fallos
- Sólo sirve para sistemas no-críticos

Sistemas con recursos adecuados e inadecuados

Se distinguen por la cantidad de recursos disponibles

Sistemas con recursos adecuados

(*resource-adequate systems*)

- Diseño con suficientes recursos para garantizar el comportamiento temporal con máxima carga y en caso de fallos

Sistemas con recursos inadecuados

(*resource-inadequate systems*)

- Diseño con recursos “razonables” desde un punto de vista económico
- Sólo sirve para sistemas no-críticos

Sistemas dirigidos por tiempo y por eventos

Se distinguen por la forma de lanzar las tareas

Sistemas dirigidos por eventos

(*event-triggered systems*)

- Arranque cuando se produce un evento de cambio de estado
- Mecanismo básico: interrupciones

Sistemas dirigidos por tiempo

(*time-triggered systems*)

- Arranque en instantes de tiempo predeterminados
- Mecanismo básico: reloj

En resumen

- **Los sistemas de tiempo real interaccionan con su entorno y ejecutan sus acciones dentro de intervalos de tiempo determinados**

- Suelen tener requisitos muy exigentes
  - Tamaño y complejidad
  - Concurrencia
  - Interfaces de hardware específicas
  - Fiabilidad y seguridad
  - Determinismo temporal

- Hay diferentes clases de sistemas de tiempo real, con distintos requisitos temporales y de seguridad

Sistemas Operativos de Tiempo Real

Tomado de:

Juan Manuel Cruz . Sistemas Embebidos. Sistemas d e Tiempo Real. Simposio Argentino de Sistemas Embebidos, 2013.

Implementación de sistemas de tiempo real

- Los métodos, herramientas y tecnología que se usan para construir otros tipos de sistemas no sirven para los sistemas de tiempo real:
  - No son suficientemente fiables
  - Sólo contemplan el tiempo de respuesta medio, no el peor
  - No garantizan los requisitos temporales

- Las plataformas de desarrollo y ejecución suelen ser diferentes
  - Es difícil hacer pruebas en la plataforma de ejecución
  - Es difícil medir los tiempos con precisión

Sistemas Operativos

- Los sistemas operativos convencionales no son adecuados para implementar sistemas de tiempo real
  - No tienen un comportamiento determinista
  - No permiten garantizar los tiempos de respuesta
  - Son poco fiables

- Un sistema operativo de tiempo real (RTOS) debe soportar
  - Concurrencia: procesos ligeros (hebras o threads)
  - Temporización: medida de tiempos y ejecución periódica
  - Planificación determinista: gestión del procesador y otros recursos
  - Dispositivos de E/S: acceso a recursos de hardware e interrupciones

Componentes básicas de un OS

- **Reloj de Tiempo Real** (*Real Time Clock, RTC*)
  - Proporciona la temporización del sistema
- **Programador de Procesos** (*Scheduler*)
  - Establece el orden de ejecución de las tareas de acuerdo a una estrategia establecida
- **Ejecutor** (*Dispatcher*)
  - Gestiona el inicio y la finalización de la ejecución de una tarea (cambios de contexto, stack, etc.)
- **Administrador de Interrupciones** (*Interrupt Handler*)
  - Gestiona las peticiones de interrupciones de hardware o software

Componentes de un OS

- **Gestor de Recursos** (*Resource Manager*)
  - Administra el acceso a los recursos de hardware del sistema (procesador, memoria, dispositivos de I/O, comunicaciones, etc.)

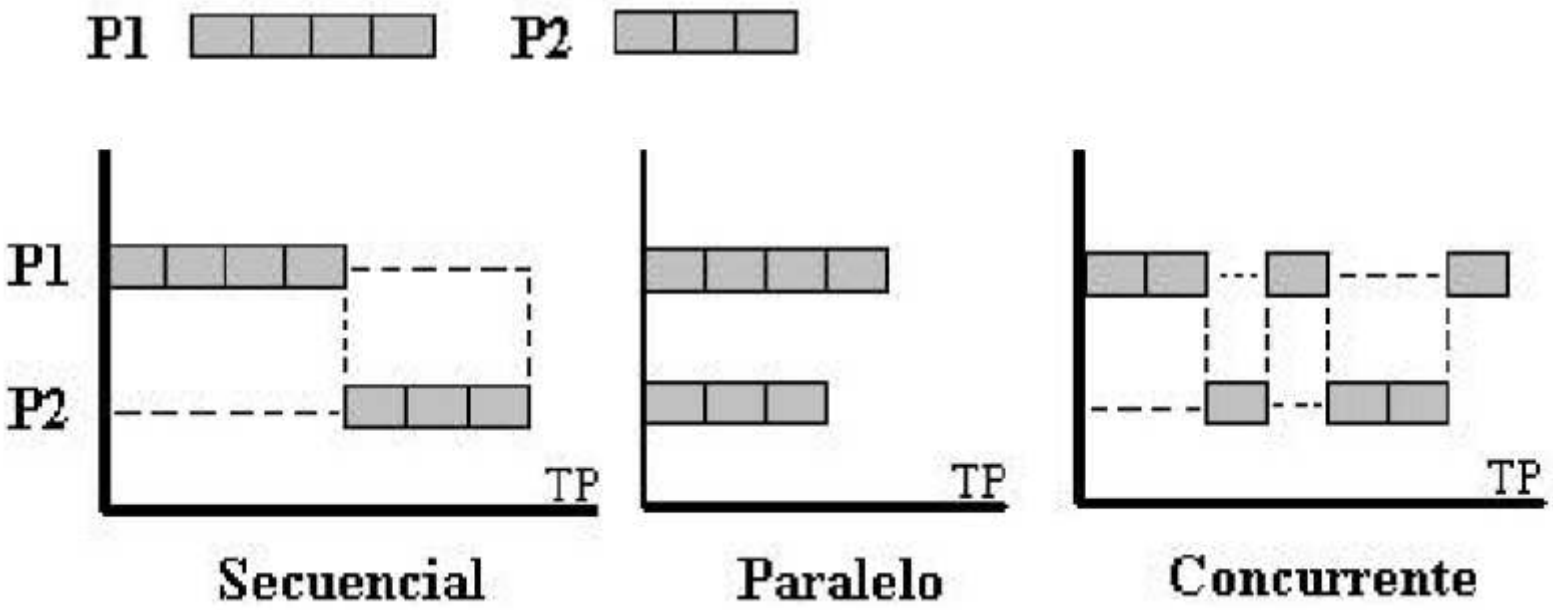
- **Gestor de Configuraciones** (*Configuration Manager*)
  - Administra la configuración del sistema (cambio de componentes de hardware o actualización de software) sin interrumpir el funcionamiento del mismo

- **Gestor de Fallos** (*Fault Manager*)
  - Detecta fallos de hardware y software y toma acciones para asegurar la disponibilidad del sistema

- Estos dos últimos imprescindibles en OS de misión crítica

Multitasking

- Se reparte el tiempo de procesamiento entre las distintas tareas para su ejecución paralela o concurrente



Prioridad de las Tareas

- El concepto de prioridad es básico en sistemas con interrupciones
- El procesamiento de determinados eventos suele ser más importante que el de otros
- Las tareas de alta prioridad están asociadas a eventos que exigen pequeña latencia para producir su respuesta
- Las tareas de baja prioridad se asocian a eventos en los que es posible mayor latencia, a veces por la naturaleza del proceso a controlar y otras veces porque el hardware provee alguna funcionalidad de buffering de datos

Ejecución de Tareas

- El **programador de procesos** (*scheduler*) determina cuál será la siguiente tarea a ser ejecutada, elaborando la lista de tareas a ejecutar
  - Tal determinación obedece a una estrategia de programación (scheduling strategy) preestablecida
  - Usualmente Su accionar se repite a intervalos regulares establecidos por el reloj de tiempo real (RTC)

- El **ejecutor** (*dispatcher*) toma el primer proceso de la lista generada por el scheduler, le asigna memoria y tiempo de procesador (se encarga del cambio de contexto) e inicia su ejecución (le transfiere el control)

Estrategias de Scheduling

- **Cooperativa** o **No Expropiativa** (*Non preemptive*)
  - Cada tarea se ejecuta hasta su finalización, se autobloquea (p/ej.: esperando una respuesta de hardware) o
  - suspende voluntariamente su ejecución para permitir la ejecución de otra tarea

- **Expropiativa** (*Preemptive*)
  - La ejecución de la tarea puede ser suspendida por el scheduler
    - al surgir una tarea con mayor prioridad (expropia la CPU a la tarea que se esté ejecutando y se la cede cambio de contexto mediante) o
    - al finalizar el tiempo asignado a esa tarea

Algoritmos de Scheduling

- **Turno circular** *(Round-Robin)*: Las tareas se ejecutan en secuencia de turnos:
  - Durante un tiempo preestablecido (time slice) e igual para todos o
  - Con un tiempo asociado a cada tarea o
  - Una tarea comienza cuando la anterior finaliza

- **Crecimiento monótono** (*Rate Monotonic*):
  - Cada tarea tiene asociado un nivel de prioridad único y se ejecuta hasta su finalización o bloqueo

- **Menor tiempo de finalización primero:**
  - Se ejecuta primero la tarea que requiere menos tiempo estimado para su finalización

- Y otros muchos…

Status de una Tarea

- Una tarea esencialmente puede estar en dos estados:
  - *Not Running*: No está en ejecución (es un estado compuesto por subestados)
    - *Ready*: Lista para su ejecución
    - *Bloqued*: A espera de un evento para volver a Ready
    - Suspended: Inactiva
  - *Running*: En ejecución

- El scheduler elabora la listas de tareas ready, para que oportunamente el dispatcher le asigna memoria y tiempo de procesador e inicia su ejecución
  - Se basa en el estado de la tarea, la prioridades asignadas a ellas y a la estrategia de scheduling establecida

Comunicación y sincronización de tareas

- **Colas** *(Queues*)
  - Es una colección de estructuras de datos ordenada
  - Las operaciones que se pueden realizar son
    - agregar al final,
    - sacar del inicio (Fist In – First Out),
    - sacar del final (Firt In – Last Out),
    - información del estado de la cola y
    - copias datos (sin modificar la cola)
  - Pueden usarse como buffers de datos si una tarea genera más información que la que otra es capaz de procesar
  - Las implementaciones más comunes son mediante buffers circulares y listas enlazadas
  - Su uso correcto es responsabilidad del desarrollador

Comunicación y sincronización de tareas

Exclusión mutua *(mutex)*

- Se utiliza para bloquear el acceso a recursos de hardware o software que deben ser compartidos por distintas tareas
- El mutex actúa como una ticket (token) que debe ser adquirido por la tarea que desea acceder al recurso compartido
- Una vez que una tarea adquiere el mutex asociado a un recurso, ningún otra tarea puede adquirirlo hasta que sea liberado por la tarea que lo adquirió primero
- Su uso correcto es responsabilidad del desarrollador

Comunicación y sincronización de tareas

- **Semáforos** (*Semaphores*)
  - Se utilizan para sincronizar tareas
  - Pueden ser binarios (tienen dos estados) o contadores (llevan un conteo)
  - Una tarea es propietaria del semáforo y fija su estado o valor de conteo, otras tareas pueden leer el estado o el conteo y actuar en consecuencia
  - Su uso correcto es responsabilidad del desarrollador

Comunicación y sincronización de tareas

Sockets

- En general se utilizan para sincronizar tareas que se ejecutan en distintos dispositivos conectados en red
- Suelen utilizarse una arquitectura cliente-servidor en la que una tarea (servidor) provee información o recursos a otras tareas (clientes)
- Cada tarea dispone de un socket y éstos se conectan entre sí Cada socket puede aceptar conexiones de un número distintos de otros sockets
- Son bidireccionales, secada tarea puede leer y escribir en ellos

Comunicación y sincronización de tareas

Memoria compartida (*Shared Memory*)

- Es un área de memoria física compartida entre dos o más tareas. Todas las tareas las ven como parte de su propia área de memoria
- Normalmente las tareas acceden a ella mediante punteros. Es un método de comunicación de bajo nivel
- No hay controles sobre el acceso a la memoria, los mismos deben implementarse mediante semáforos u otras técnicas
- Su uso correcto es responsabilidad del desarrollador

Comunicación y sincronización de tareas

- En todos los casos presentados es de destacar que su uso correcto es responsabilidad del desarrollador. Hay que evitar posibles problemas de sincronización:
  - **Interbloqueos** (*Deadlocks*): cuando dos o más tareas concurrentes se encuentran c/u esperando a la otra para proseguir (lo que nunca ocurrirá)
  - **Inanición** (*Starvation*): cuando a una tarea se le niega el acceso a un recurso compartido
  - **Inversión de prioridades**: cuando dos tareas de distinta prioridad comparten un recurso y la de menor prioridad bloquea el recurso antes que la de prioridad mayor, bloqueándose esta última al momento que precise el uso del recurso compartido

Apéndice II
Análisis temporal de sistemas de Tiempo Real

Tomado de:

Arquitectura de Sistemas de Tiempo Real (ASTR 5002/3).
TPBN@icaro.eii.us.es. 5º Ingeniería en Informática. Tema 5: Análisis temporal.

Objetivo

El análisis temporal de sistemas de TR sirve para

- estudiar métodos de planificación usados en sistemas de TR
- determinar si un sistema cumplirá sus restricciones temporales
- identificar los problemas asociados a bloqueos entre tareas

Viabilidad de sistemas de TR

- La concurrencia de tareas puede producir indeterminismo en el tiempo de respuesta de cada tarea.
- Para evitarlo, un SOTR debe tener dos características:
  - Un *algoritmo de planificación* (task scheduling) determinista, que regule qué tarea utiliza el procesador en cada momento.
  - Un método para predecir el "peor comportamiento posible" del sistema: *análisis de planificabilidad (schedulability analysis).*
- El análisis de planificabilidad es un análisis del comportamiento temporal de las tareas.
- Su objetivo es asegurar la viabilidad del sistema:
  - Todas las tareas con restricciones temporales terminan antes de su límite temporal.
  - Bajo cualquier circunstancia.

Viabilidad de sistemas de TR

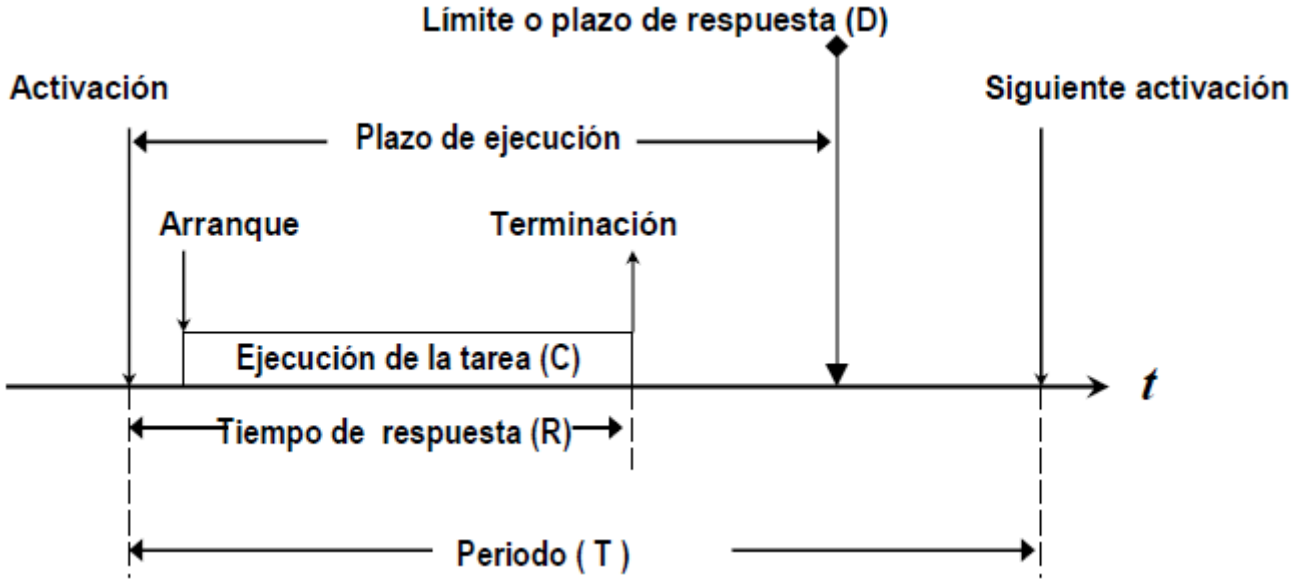
Tipos de algoritmos de planificación

- Manuales: *Ejecutivo cíclico.* Se ordenan manualmente las tareas y el procesador las ejecuta cíclicamente en el mismo orden.

- Automáticos: *Expulsivos guiado por prioridades*: Las tareas tienen asignadas una prioridad, y el SO ejecuta siempre la que tiene mayor prioridad.
  - Con prioridades fijas: *Rate Monotonic (RM) y Deadline Monotonic* (DM).
  - Con prioridades dinámicas: *Earliest Deadline First (EDF).*

Caracterización temporal. Modelado de tareas

- Características temporales de una tarea periódica:



- En las tareas esporádicas, el periodo se sustituye por la separación o tiempo mínimo entre activaciones consecutivas.

Caracterización temporal. Modelado de tareas

Notación asociada a una tarea τ_i:

- N Número total de tareas en el sistema.
- T_i Periodo entre dos lanzamientos consecutivos.
- C_i Tiempo de ejecución máximo.
- D_i Tiempo o plazo de respuesta máximo.
- P_i Prioridad

Se imponen algunas restricciones para permitir/facilitar el análisis de la planificación:

- El conjunto de tareas es estático.
- Todas las tareas son periódicas.
- Las tareas son independientes unas de otras (al menos en su comportamiento temporal).
- Los plazos de respuesta de una tarea es igual al su periodo. Para todas las tareas ti se cumple que:

$$C_i \leq D_i = T_i$$

- Las operaciones del núcleo de multiprogramación son instantáneas (cambios de contexto, interrupciones, ...)
- Hay que asegurar que

$$R_i \leq D_i, \text{ para todo } i$$

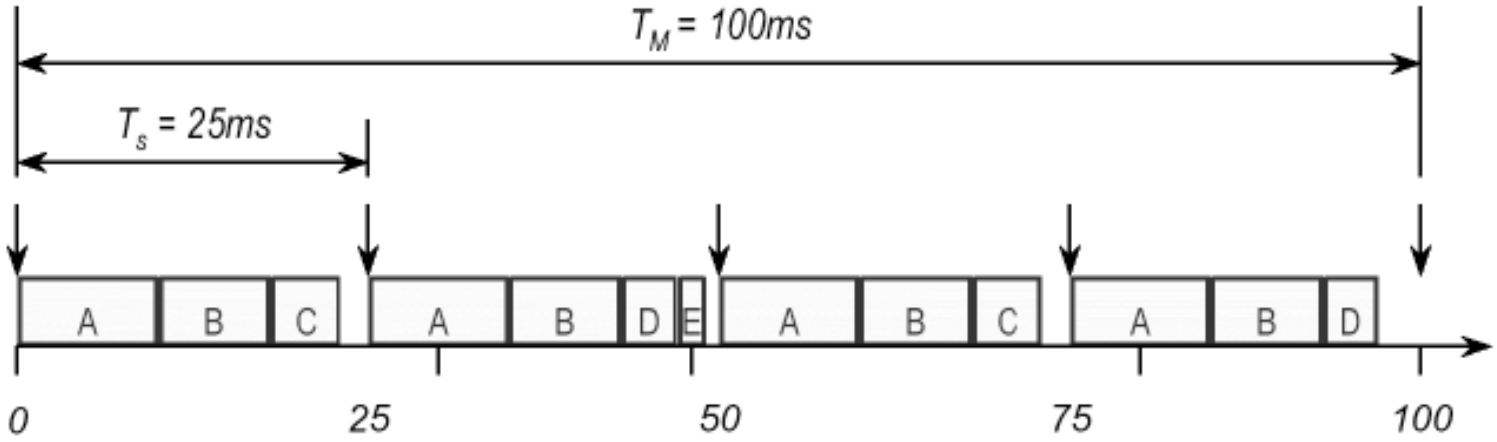
Políticas de planificación: Planificación cíclica

- La planificación es un bucle en el que las acciones se realizan en secuencia, cíclicamente, en un orden preestablecido.
- Si todas las tareas son periódicas se planifican usando un esquema que se repite con un periodo igual al mayor del conjunto de tareas T_M= *Ciclo principal o hiperperiodo* del sistema.
- El ciclo principal se divide en ciclos secundarios, TS. El periodo de los ciclos secundarios debe cumplir las siguientes condiciones:
  - $T_M = k * T_S$, siendo k un número entero.
  - T_S es menor o igual que el menor periodo de las tareas del sistema.
  - Los periodos deben ser armónicos.
- Se programa un timer para que se dispare con periodo T_S. En cada disparo del timer, se ejecuta parte de las tareas hasta completar el hiperciclo.

Planificación cíclica. Un ejemplo

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

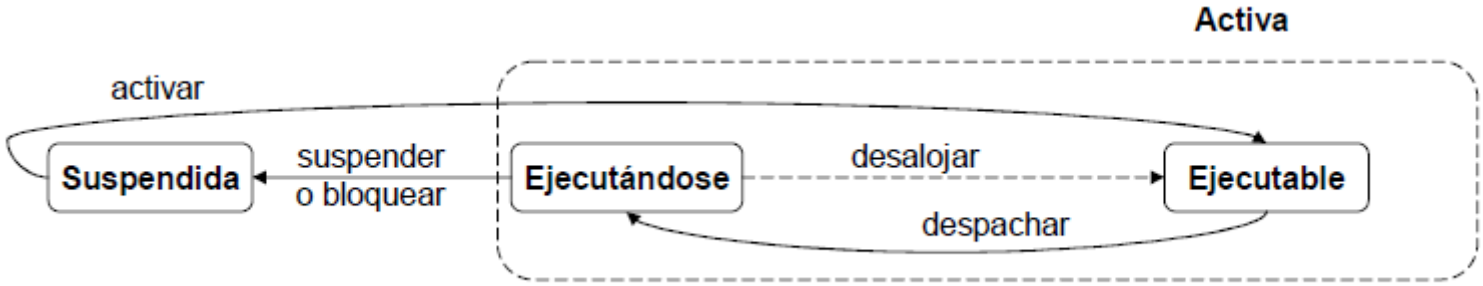
El ciclo principal es de 100 ms, compuesto por 4 ciclos secundarios.



Planificación cíclica. Pros y contras

- Simple si la estructura de tareas es simple.
  - No hay concurrencia en la ejecución de las tareas.
  - No hacen falta mecanismos de exclusión mutua.
  - Los periodos deben ser armónicos.
  - No es necesario analizar comportamiento temporal $\rightarrow$ el sistema es correcto por construcción.
  - No se necesita kernel para la planificación $\rightarrow$ adecuado para sistemas simples y cerrados.
- Se complica exponencialmente con la complejidad de las tareas.
  - Las tareas esporádicas son difíciles de tratar.
  - El plan cíclico es difícil de construir si:
    - Los periodos son de diferente orden de magnitud $\rightarrow$ muchos ciclos secundarios.
    - Es necesario repartir una tarea en varios procedimientos.
  - Es poco flexible y difícil de mantener.

Políticas de planificación: Planificación con prioridades fijas



- Las tareas se ejecutan de forma concurrente.
- Una tarea puede estar al menos en tres estados (depende del SO).
- Las tareas ejecutables se despachan en orden de prioridad.
  - Con desalojo.
  - Sin desalojo.
- Supondremos un sistema con prioridades fijas y desalojo.
- Los métodos de planificación que estudiaremos se diferencia en cómo asignar automáticamente las prioridades.

Planificación Rate Monotonic (RM)

- Se asigna las prioridades en función del periodo de lanzamiento:
  - A menor periodo, mayor prioridad.
- Es una asignación óptima:
  - Si se pueden garantizar los plazos de un sistema de tareas con otra asignación se pueden garantizar con la asignación RM.
  - Esto es válido siempre que tengamos tareas periódicas y $D_i = T_i$.
- Sistema planificable por RM si

$$U = \sum_{i=1}^{N} \frac{C_i}{T_i} \leq N\left(2^{1/N} - 1\right)$$

- A $U_0(N) = N(2^{1/N} - 1)$ se le llama *utilización mínima* garantizada.
- Es una condición suficiente, pero no necesaria.

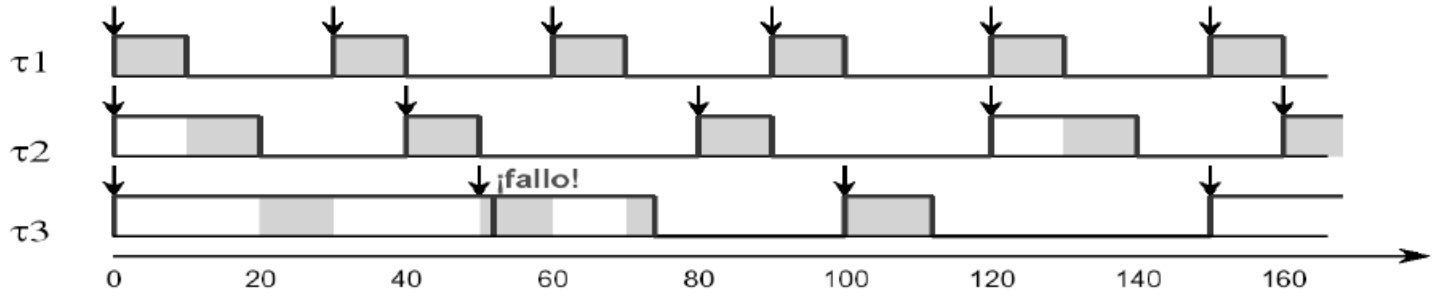
Análisis de la planificación RM. Un ejemplo

N	U_0
1	1.000
2	0.828
3	0.779
4	0.756
5	0.743

$$\lim_{N \to \infty} U_0(N) = \log 2 \approx 0.693$$

Un ejemplo:

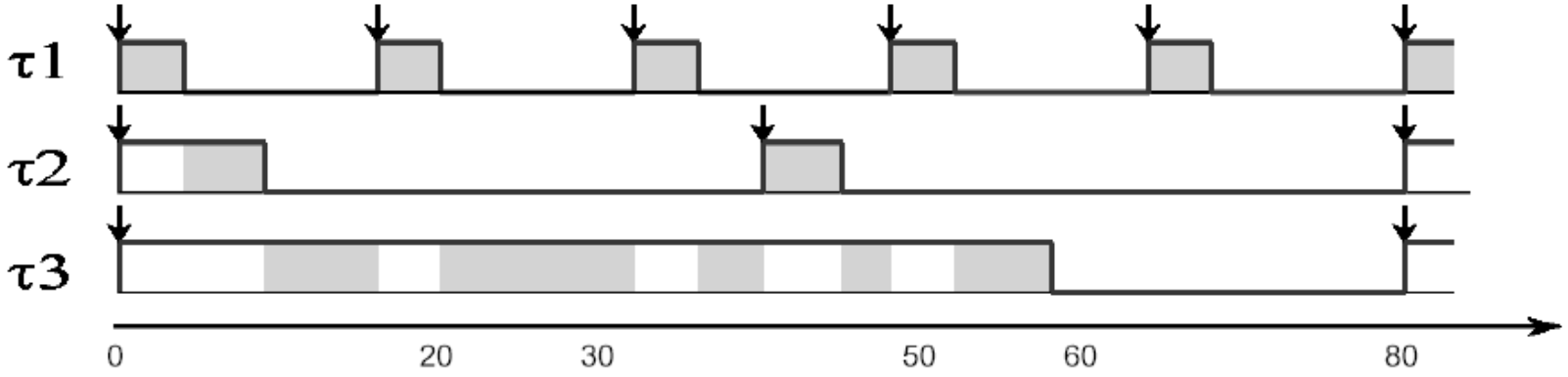
Tarea	*T*	*C*	*P*	*U*
τ1	*30*	*10*	*3*	0,333
τ2	*40*	*10*	*2*	0,250
τ3	*50*	*12*	*1*	0,240
				0,823



Análisis de la planificación RM. Instante crítico (I)

- Válido para cualquier sistema con prioridades fijas, siempre que Di≤ Ti.
- *Instante critico* es aquel en el que todas las tareas se lanzan al mismo tiempo y se ejecutan durante Ci.
- El análisis de la planificación por instante critico se basa en hacer un cronograma del instante crítico para todas las tareas del sistema

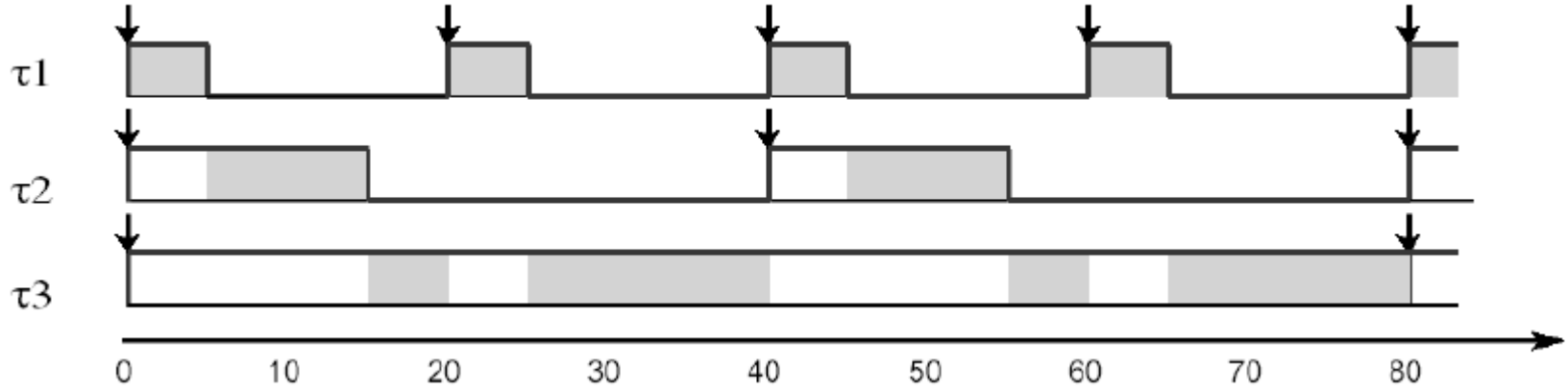
Tarea	T	C	P	U
τ1	16	4	3	0,250
τ2	40	5	2	0,125
τ3	80	32		0,400



Análisis de la planificación RM. Instante crítico (II)

- La prueba del factor de utilización no es exacta:

Tarea	T	C	P	U
τ1	20	5	3	0,250
τ2	40	10	2	0,250
τ3	80	40	1	0,500
				1,000

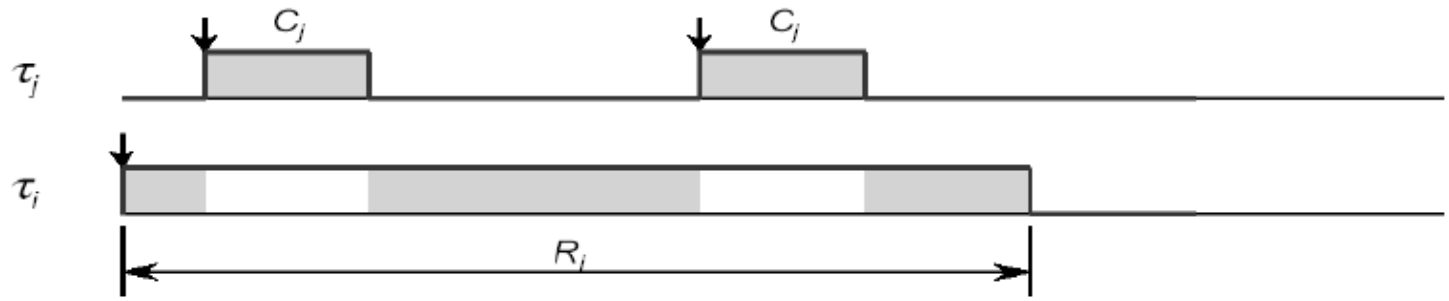


- El sistema es planificable a pesar de que U > 0.779
  - Y no se puede generalizar a modelos de tareas complejos.
  - Construir el cronograma es complejo.

Análisis de la planificación RM. Calculo del tiempo de respuesta (I)

- Se calcula el tiempo de respuesta de cada tarea y se compara con el su plazo de ejecución.

$$R_i = C_i + \sum_{j=hp(i)} \left\lceil \frac{R_i}{T_j} \right\rceil C_j$$



- No es una ecuación continua ni lineal no se puede resolver analíticamente.

Análisis de la planificación RM. Calculo del tiempo de respuesta (II).Iteración lineal

- La ecuación del tiempo de respuesta se puede resolver usando un método iterativo.

$$R_i^0 = C_i + \sum_{j=hp(i)} C_j$$

$$R_i^{n+1} = C_i + \sum_{j=hp(i)} \left\lceil \frac{R_i^n}{T_j} \right\rceil C_j$$

- La iteración termina cuando $R^{n+1}{}_i = R^n{}_i$ o $R^{n+1}{}_i > D_i$

Más información sobre Sistemas de Tiempo Real

Para profundizar en Sistemas de Tiempo Real

Master Oficial en Ingeniería de Sistemas Empotrados (MOISE 2012-13)

Introducción al tiempo real en sistemas empotrados

Contenido del curso (Parte 1)

- Tema 1. Introducción
- Tema 2. Soporte de interrupciones
- Tema 3. Conceptos de sistemas operativos
- Tema 4. Planificación en sistemas de tiempo real
- Tema 5. Mecanismos de sincronización y comunicación
- Tema 6. Planificación de tiempo real con recursos compartidos

Bibliografía

- SOBRE SISTEMAS OPERATIVOS Y SINCRONIZACIÓN
  - A. Lafuente: Sistemas Operativos II. Apuntes de la asignatura. Edición 2009-10.
  - C. Rodríguez, I. Alegría, J. G. Abascal, A. Lafuente: Descripción funcional de los sistemas operativos. Síntesis, 1994.
    S. Sánchez Prieto: Sistemas Operativos. Universidad de Alcalá de Henares, Servicio Editorial, 2005.
  - A. Silberschatz, P. Galvin, G. Gagne: Conceptos de Sistemas Operativos (7a edición). Willey, 2006.
  - A.S. Tanenbaum: Modern Operating Systems (3rd edition). Prentice-Hall, 2008.

- SOBRE TIEMPO REAL
  - G.C. Buttazzo: Hard Real-Time Computing Systems: Predictable Scheduling Algorithms and Applications (3rd edition), Springer 2011.
    Q. Li: Real-Time concepts for embedded systems. CMP Books, 2003.
  - J. Liu: Real-Time Systems, Prentice-Hall, 2000.
  - H. Kopetz: Real-Time Systems: design principles for distributed embedded applications. Kluwer, 1997.

Referencias

- Real-Time Systems and Programming Languages (Fourth Edition) Ada 95, Real-Time Java and Real-Time POSIX by Alan Burns and
- Andy Wellings. http://www.cs.york.ac.uk/rts/books/RTSBookFourthEdition.html
- Sistemas de Tiempo Real, Juan Antonio de la Puente, Departamento de Ingeniería de Sistemas Telemáticos, Universidad Politécnica de Madrid http://web.dit.upm.es/~jpuente/strl/
- Introducción al tiempo real en sistemas empotrados (Master Oficial en Ingeniería de Sistemas Empotrados), Alberto Lafuente, Universidad del País Vasco, http://www.sc.ehu.es/acwlaroa/ITRSE.htm
- Sistemas en Tiempo Real, Escuela Universitaria de Ingeniería de Gijón http://isa.uniovi.es/docencia/TiempoReal/
- Sistemas Empotrados en Tiempo Real - José Daniel Muñoz Frías www.lulu.com/product/ebook/sistemas-empotrados-en-tiempo-real/
- Ing. Juan Manuel Cruz. Sistemas Embebidos. Sistemas de Tiempo Real. Seminario de Electrónica: Sistemas Embebidos. Ingeniería en Electrónica – FIUBA. 2013. http://laboratorios.fi.uba.ar/lse/seminario/
- J. García Martín. Introducción a los Sistemas de Tiempo Real. UPMadrid. http://ocw.upm.es/arquitectura-y-tecnologia-de-computadores/introduccion-a-los-sistemas-de-tiempo-real/contenidos/material-de-clase/transp_cap1_introduccion.ppt/view