Características de los Sistemas Real Time

Aspectos de un Sistema de Tiempo Real

- Aspectos que suelen estar asociados con los sistemas de tiempo real, son:
  a. Naturaleza empotrada: Los programas de tiempo real se presentan en sistemas electrónicos empotrados en equipos físicos (Embedded system), tales como controladores de sistemas robotizados, equipos de control físicos, instrumentación inteligente, etc.
  b. Fuerte interacción con el entorno: La característica más relevante de un sistema de tiempo real es su necesidad de interacción con un entorno físico externo, que es de donde surge los requerimientos de tiempo real. Esto supone que deben ser soportados por computadores especiales, con una componente de entrada/salida muy relevante.

Características de un Sistema de Tiempo Real

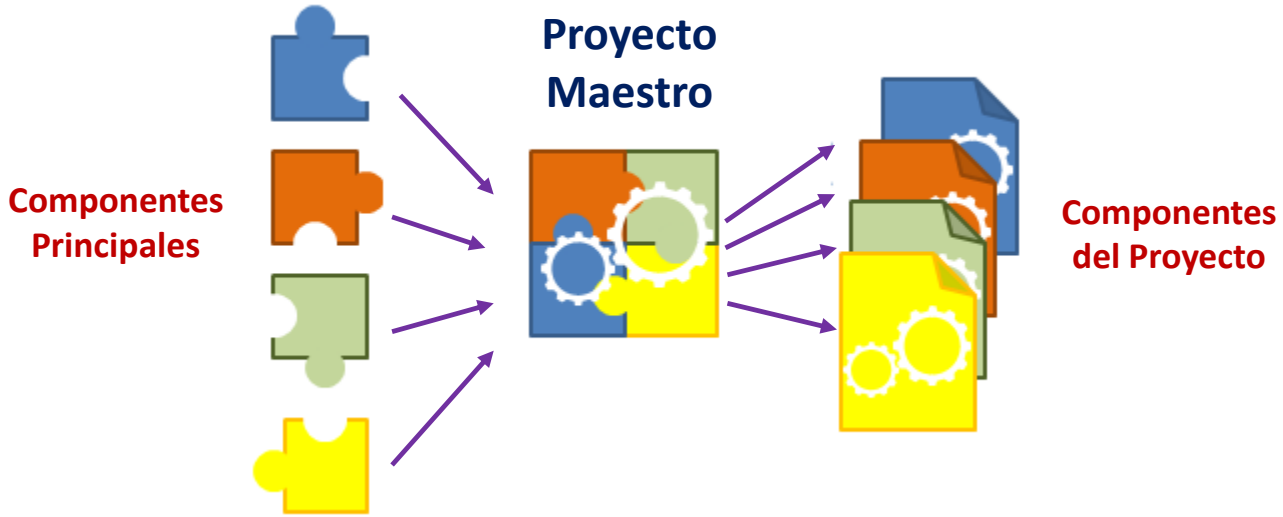
- Restricciones temporales: Ya sea de forma estricta o laxa, los sistemas deben responder a los eventos externos dentro de unos plazos especificados de tiempo físico.
- Control de sistemas físicos: Los sistemas de tiempo real corresponden a menudo con sistemas de control automáticos, en los que de acuerdo con los datos que se adquieren, el sistema toma decisiones de control de forma automática y sin intervención humana.
- Naturaleza reactiva: La mayoría de los sistemas de tiempo real son de naturaleza reactiva, esto es, son sistemas gobernados por eventos, en los que los diferentes componentes del software se activan de acuerdo con estímulos recibidos desde el entorno.

Características de un Sistema de Tiempo Real

- Los Sistemas de Tiempo Real (STR) llevan asociadas una serie de características básicas como las siguientes:
  1. Tamaño y complejidad
  2. Fiabilidad y seguridad
  3. Cálculos con números reales
  4. Interacción con dispositivos físicos
  5. Eficiencia
  6. Dependencia del tiempo
  7. Concurrencia
  8. Tolerancia ante fallos

1. Tamaño y complejidad

- Son dos aspectos ligados al software
- Algunos STR tienen millones de líneas de código.
- La variedad de funciones aumenta la complejidad incluso en sistemas relativamente pequeños.
- Se descompone el sistema en subsistema pequeños



2. Fiabilidad y Seguridad

- Los requisitos de fiabilidad y seguridad en los STR son mayores que en el resto
- En sistemas críticos: fallos con consecuencias graves
  - ✓ Pérdida de vidas humanas
  - ✓ Pérdidas económicas
  - ✓ Daños medioambientales
- Fiabilidad
  - ✓ Es la probabilidad de proporcionar el servicio especificado
  - ✓ Para una tasa de fallos de λ averías/hora la media de tiempo entre averías **MTTF=1/ λ**
- Seguridad
  - ✓ Los sistemas críticos deben ser ultrafiables
  - ✓ Para ciertos proyectos es necesaria una certificación oficial

2. Fiabilidad y Seguridad

- Se estima en los sistemas software complejos, que por cada millón de líneas de código contienen una media de 20.000 errores
  - ✓ El 90% de esos errores pueden ser detectados con sistemas de comprobación
  - ✓ 200 errores de los restantes se detectan durante el primer año. Los 1800 restantes permanecen sin detectar

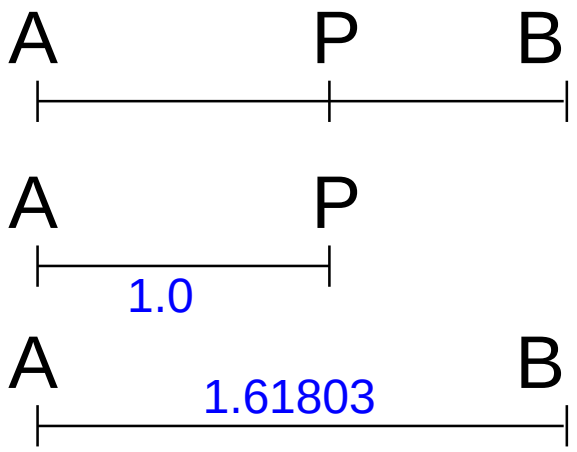
2. Fiabilidad y Seguridad

De dónde provienen los errores?

a. Una especificación inadecuada
b. Errores de diseño de los componentes SW
c. Averías en los componentes de hardware
d. Interferencias transitorias o permanentes en los sistemas de comunicación
   - ✓ Los errores c y d tienen un comportamiento previsible con métodos estadísticos.
   - ✓ El lenguaje de programación debe facilitar el desarrollo de sistemas seguros
   - ✓ Se hace necesario el empleo de técnicas de prevención y tolerancia ante fallos

3. Calculo de números reales

- El sistema controlado proporciona valores (números) asociados a una magnitud física
- Estos valores (datos crudos de ADC) se filtran y se convierten a datos de ingeniería
- El ordenador los representa de una forma aproximada (coma fija, coma flotante)
- Y también realiza los cálculos de forma aproximada (técnicas numéricas de resolución de ecuaciones)
- Cuando algún dato toma valores incorrectos, se generan alarmas



3. Calculo de números reales

Explosión ARIANE 5

El cuatro de junio de 1996 el cohete Ariane 5 lanzado por la Agencia Europea del Espacio (ESA) explotó a los cuarenta segundos de su despegue en Kourou, en la Guayana Francesa.

- Era su primer viaje después de una década de desarrollo y un coste de siete mil millones de dólares de entonces.
- El cohete mismo y su cargamento tenían un valor de 500 millones.
- Era un cohete gigante, capaz de transportar y poner en órbita un par de satélites de tres toneladas por cada lanzamiento,
- Ello otorgaría a Europa una supremacía abrumadora en el negocio espacial. Un error aritmético provocó su destrucción.

3. Calculo de números reales

- A los 39 segundos del lanzamiento, cuando el cohete alcanzó una altitud de dos millas y media, el mecanismo de autodestrucción acabó con el Ariane 5 y su carga de satélites científicos no asegurada.
- La destrucción fue disparada porque las fuerzas aerodinámicas estaban rajando los propulsores, pero había empezado un instante antes, cuando la nave viró bruscamente en una corrección de rumbo innecesaria.
- La dirección es controlada por un sistema empotrado. Erróneamente pensó que el cohete iba a la velocidad que no iba realmente.
- Estos datos de velocidad los proporciona el dispositivo inercial de guía, también controlado por software. Este dispositivo usa giróscopos y acelerómetros que monitorizan el movimiento.

3. Calculo de números reales

- El caso es que el sistema inercial debido al error de truncamiento enviaba datos absurdos, pero datos al fin y al cabo
- La comisión de expertos que investigó el desastre concluyó que la falla, un número fuera de rango, se produjo en una pedazo de código heredado del sistema de Ariane 4.

3. Calculo de números reales

- En 1994, un profesor de matemáticas de la universidad de Lynchburg descubre un error error en la unidad de coma flotante del Intel Pentium.

Error: algunas operaciones de división devolvían siempre un valor erróneo por exceso. Este fallo de diseño se hizo notorio muy rápidamente y se le dio el nombre de **error FDIV del Pentium** (FDIV es la instrucción de división en coma flotante de los microprocesadores x86)

Intel Pentium 66MHz (sSpec=SX837) FDIV.

3. Calculo de números reales

- Aunque Intel corrigió rápidamente el error y se comprometió a reemplazar los procesadores, el daño ya estaba hecho. Cerca de dos millones de procesadores defectuosos habían sido distribuidos alrededor del mundo.
- La falla: un error de diseño en el algoritmo de división de punto flotante.

4. Interacción con dispositivos Especiales

- Mecanismos de entrada/salida dependientes del dispositivo
  - ✓ Monitorización de sensores y actuadores
  - ✓ Registros de entrada y salida para capturar los datos
- Se pueden generar interrupciones o excepciones
  - ✓ Indicar la realización de ciertas operaciones
  - ✓ Alertar de la existencia de condiciones de error
- Los manejadores de dispositivos forman parte del software de la aplicación
- No están bajo el control del sistema operativo
  - ✓ Antes la interfaz con los dispositivos se dejaba en manos del SO.
  - ✓ Ahora, debido a la variedad de dispositivos y a la naturaleza de tiempo de respuesta crítico, el control debe ser directo
- No es necesario programarlos con lenguajes de bajo nivel
  - ✓ Antes se hacía mediante parches en ensamblador
  - ✓ Ahora, las exigencias de fiabilidad lo desaconsejan

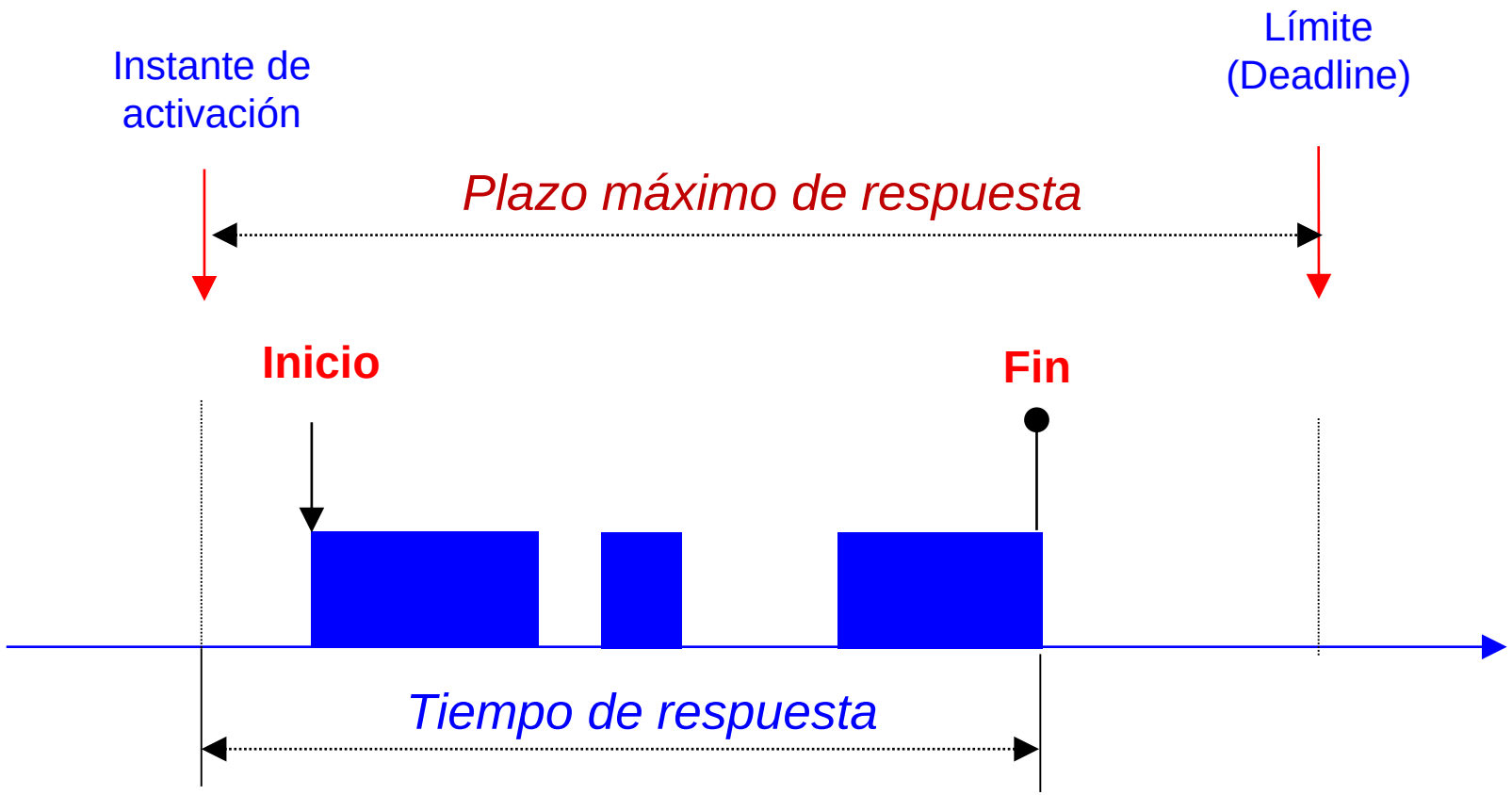
5. Eficiencia

- Eficiencia en la implementación, debido a los requisitos temporales de un STR.
  - ✓ Se requiere una evaluación constante del coste de utilización de las características del lenguaje
- Eficiencia en capacidad, ya que el espacio físico disponible suele ser muy limitado
- Eficiencia en costes económicos, puesto que incide directamente en la viabilidad comercial del producto final

6. Dependencia del Tiempo

- Tiempos en los cuales se deben llevar a cabo determinadas acciones
- Tiempos de terminación de cada acción
- Responder a situaciones transitorias en las que no se pueden garantizar todos los plazos
- Realizar cambios dinámicos en el comportamiento temporal del sistema
- El comportamiento temporal del sistema debe ser DETERMINISTA. Tareas deben ser asignadas y terminadas antes de su plazo
- La ejecución correcta no solo considera la lógica sino también el tiempo en que se producen los resultados

6. Dependencia del Tiempo



El tiempo de respuesta puede variar (*jitter*)

6. Dependencia del Tiempo: Determinismo

- Las aplicaciones que se ejecutan determinísticamente muchas veces realizan una tarea crítica en iteraciones y estas iteraciones consumen siempre la misma cantidad de tiempo del procesador.
- De esta manera las aplicaciones determinísticas son apreciadas no tanto por su velocidad, sino por su confiabilidad a responder con consistencia a entradas y generar salidas con muy poco "parpadeo" o jitter. (Jitter es el término que se usa para identificar el promedio de las diferencias en tiempo que le toma a un ciclo ejecutarse).

6. Dependencia del Tiempo: Determinismo

- Es muy difícil diseñar e implementar sistemas que garanticen todos los plazos en todas las circunstancias posibles.
- Una aproximación al problema es:
  - ✓ Dotar al sistema de una potencia de cómputo bien sobrada para asegurar que la situación más desfavorable no provoque fallo.
  - ✓ Se exige la disponibilidad al programador de interfaces que hacen que el sistema sea predecible.

- Estas interfaces se denominan facilidades de tiempo real:
  - ✓ Especificar los tiempos en que las operaciones han de realizarse.
  - ✓ Especificar los tiempos en que las operaciones han de completarse
  - ✓ Responder a las situaciones donde no pueden ser atendidos todos los plazos

6. Dependencia del Tiempo: Predecible

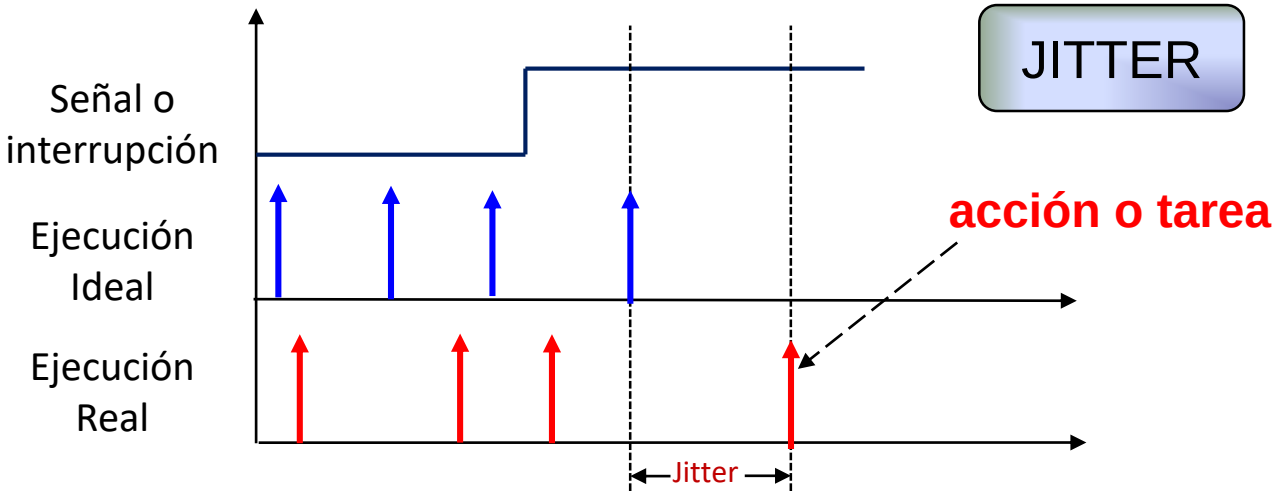
- Una característica distintiva de un sistema en tiempo real es la predecibilidad. La cual implica que debe ser posible demostrar o comprobar a priori que los requerimientos de tiempos se cumplen en cualquier circunstancia.
- Como consecuencia, la predecibilidad implica:
  - ✓ Una cuidadosa planificación de tareas y recursos.
  - ✓ Cumplimiento predecible de requisitos temporales: determinismo.
  - ✓ Anticipación a fallos, y sus requerimientos temporales.
  - ✓ Consideraciones de sobrecargas: degradación controlada.
  - ✓ Consideraciones de elementos de impredecibilidad.

6. Dependencia del Tiempo

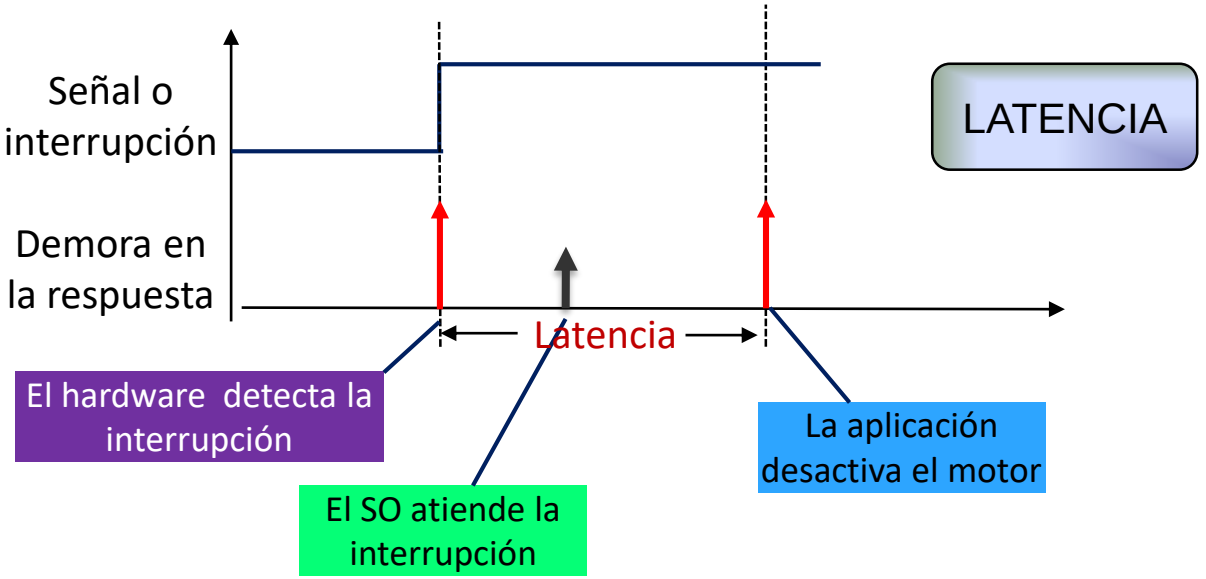
- Se llama «latencia de interrupciones » al máximo tiempo que transcurre desde que una señal de interrupción llega al procesador hasta que se ejecuta su ISR (Interrupt Service Routine) asociada.
- Las interrupciones son elementos clave de los RTOS. Cuando una interrupción tiene lugar, el procesador debe realizar varias tareas antes de ejecutar su ISR asociada:
  - ✓ En primer lugar debe completar la instrucción actual. Esto va a llevar como mínimo un ciclo de reloj, aunque también existen instrucciones complejas que requieren varios o incluso cientos de ciclos de reloj.
  - ✓ Luego, la instrucción debe ser reconocida por el procesador, que comprueba que dicha interrupción está habilitada y su prioridad supera a las de otras que puedan estar produciéndose
  - ✓ Por último la ISR asociada con la interrupción es ejecutada.

6. Dependencia del Tiempo: Predecible

Como producto de una interrupción (tarea) la acción de otra tarea se afecta en su ejecución



Cuando se genera una interrupción la respuesta en la acción demora



6. Dependencia del Tiempo

- Los diseñadores de sistemas de tiempo real invierten gran parte del tiempo en estudiar y definir con detalle el rendimiento que se precisa en el peor de los casos posibles.
- Constantemente se hacen preguntas del tipo:

¿Cuál es el máximo tiempo que puede pasar desde que el conductor pisa el freno hasta que la señal de interrupción llega al procesador?

En el caso de máxima latencia de interrupciones,

¿Cuál es el tiempo entre la llegada de la señal de interrupción y el comienzo de la rutina de atención a la interrupción asociada?

¿Cuál es el máximo tiempo que puede tardar el software en activar el mecanismo de antibloqueo de frenos?

En sistemas de este tipo, el tiempo medio o esperado son valores simplemente insuficiente

7. Concurrencia

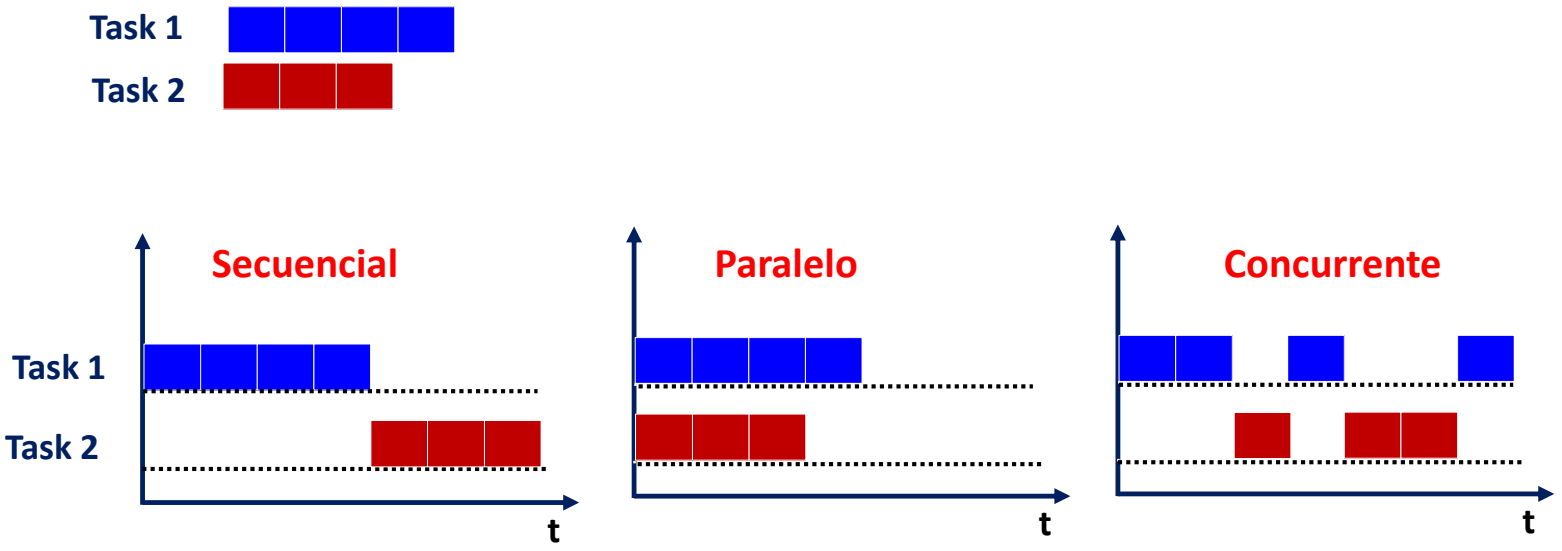
- Los dispositivos físicos controlados (sensores, actuadores, etc.) funcionan al mismo tiempo
- Las tareas software que los controlan actúan concurrentemente
- Se deben atender diversos tipos de eventos en paralelo
- Puede ser suficiente con un sólo procesador si hay suficiente velocidad de proceso para simular paralelismo con un solo procesador
- En ocasiones es necesario recurrir a sistemas multiprocesador. Muchos sistemas empotrados exigen verdadero paralelismo
- ¿Cómo expresar la concurrencia desde el programa?
  - ✓ Con ejecutivos cíclicos
  - ✓ Con tareas independientes

7. Concurrencia

- Tareas independientes, pueden ser soportadas por:
  - ✓ El lenguaje: caso de Ada o Modula
  - ✓ El sistema operativo
- Un SOTR debe proporcionar soporte básico para tareas de tiempo real, tolerancia a fallos, determinismo, etc.
- Aspectos cruciales que deben considerarse:
  - ✓ El factor tiempo
  - ✓ Los protocolos de comunicación
  - ✓ La asignación de recursos

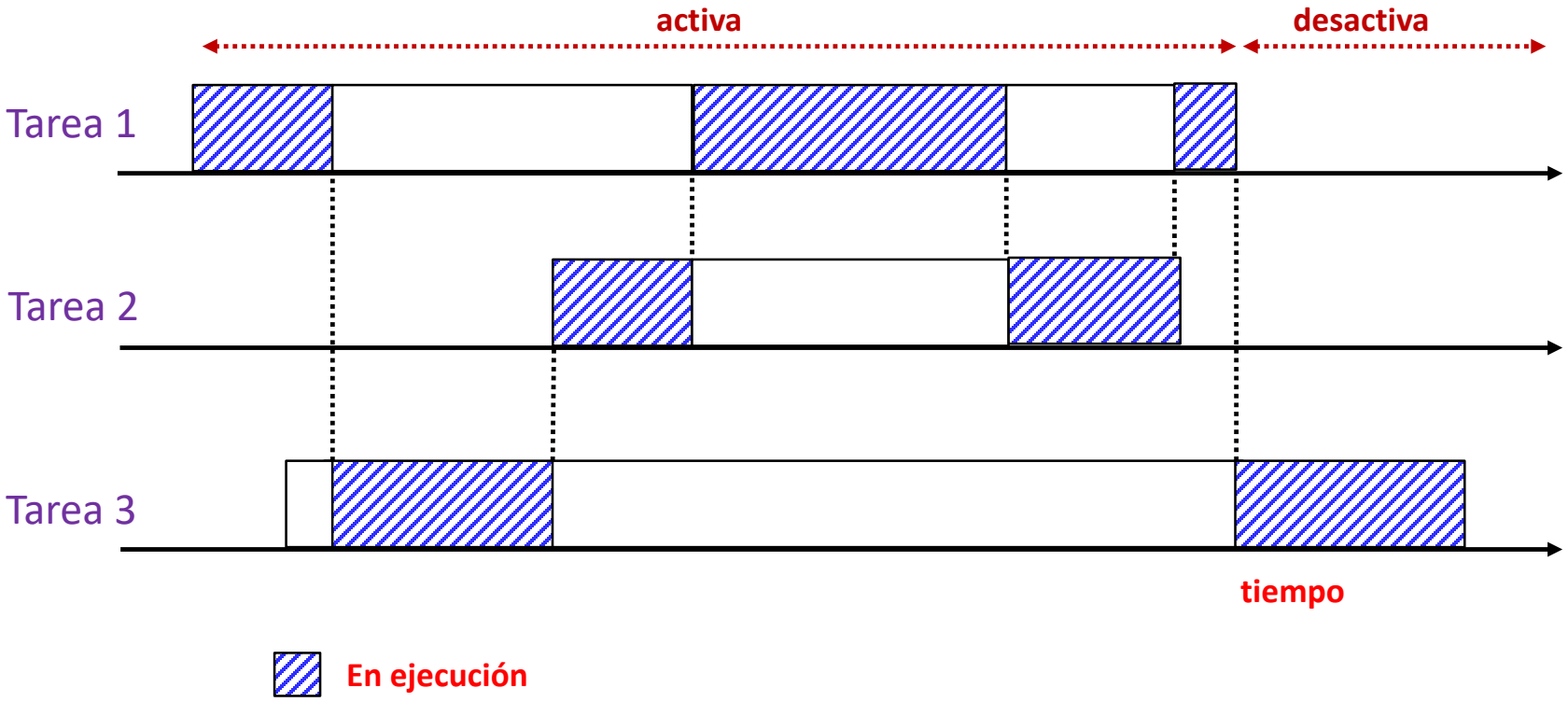
7. Concurrencia

- Se pretende repartir el tiempo de procesamiento entre las distintas tareas creando la ilusión de procesamiento concurrente



7. Concurrencia

- Los conceptos de concurrencia y planificación adquieren en los STR incluso mayor protagonismo que en los Sistemas de Tiempo Compartido
- Cada estímulo del entorno activa una o más tareas. Una tarea es una secuencia de instrucciones que ejecuta en concurrencia con otras tareas



7. Concurrencia

- En un STR la planificación de las tareas concurrentes debe asegurar propiedades que no se exigen en los Sistemas de Tiempo Compartido

1. Garantía de plazos. Un STR funciona correctamente cuando los plazos de todas las tareas están garantizados. En contraste, en un STC, lo importante es asegurar un flujo lo más elevado posible.
2. Estabilidad. En caso de una sobrecarga del STR, se debe garantizar que al menos un subconjunto de tareas *críticas* cumplen sus plazos. En un STC, el criterio es asegurar la equidad del tiempo de ejecución.
3. Tiempo de respuesta máximo. En un STR se trata de acotar el tiempo de respuesta en el peor caso de todas las tareas. En un STC se trata de minimizar el tiempo de respuesta medio.

7. Concurrencia

Los usos más comunes son en tecnologías SMPP y SMS para la telecomunicaciones aquí hay muchísimos procesos corriendo a la vez y todos requiriendo de un servicio.

1. **Trabajo interactivo y en segundo plano:**

Por ejemplo, en un programa de hoja de cálculo un hilo puede estar visualizando los menús y leer la entrada del usuario mientras que otro hilo ejecuta las órdenes y actualiza la hoja de cálculo. Esta medida suele aumentar la velocidad que se percibe en la aplicación, permitiendo que el programa pida la orden siguiente antes de terminar la anterior.

2. **Procesamiento asíncrono:**

Los elementos asíncronos de un programa se pueden implementar como hilos. Un ejemplo es como los softwares de procesamiento de texto guardan archivos temporales cuando se está trabajando en dicho programa. Se crea un hilo que tiene como función guardar una copia de respaldo mientras se continúa con la operación de escritura por el usuario sin interferir en la misma.

3. **Aceleración de la ejecución:**

Se pueden ejecutar, por ejemplo, un lote mientras otro hilo lee el lote siguiente de un dispositivo.

Aplicaciones Multitarea frente a Monotarea

- Protección de memoria: si en un proceso se accede fuera de su memoria asignada, no modifica datos en otro proceso.
- Si el algoritmo de un proceso no es correcto, no afecta a la ejecución de los demás procesos salvo sincronizaciones.
- Se pueden ejecutar varios procesos en computadores diferentes conectados mediante algún tipo de canal de comunicación.
- Los procesos pueden lanzarse y detenerse individualmente, facilitando escalabilidad de aplicaciones.
- Implican una división de tareas adecuada para coordinación de trabajos entre varios programadores.
- Simplifican el código de las aplicaciones cuando éstas tienen que atender a varios dispositivos y tareas diferentes.

Tipos de Tareas (TASK)

Tipo de Tareas: **Hard** y **Soft**

- Tareas Tiempo Crítico (Tiempo Real Duro / Hard Real-Time)

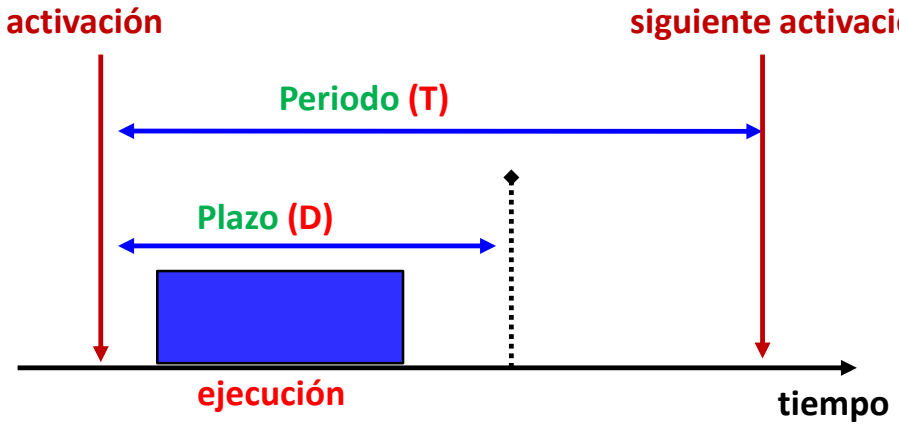
  Las tareas deben completarse antes de su plazo de respuesta. El incumplimiento de un plazo puede producir perdidas humanas o graves perdidas materiales. Ejemplo: *Sistema de frenado de un automóvil, la inyección de combustible en el motor de un avión*

- Tareas Acríticas (Tiempo Real Blando / Soft Real-Time)

  Las tareas deben completarse tan pronto como sea posible. Se puede tolerar que ocasionalmente se incumpla un plazo de respuesta. Ejemplo: *la descompresión y visualización de un fichero mpeg*. Aquí sencillamente se tiene una degradación de la calidad del sistema (la imagen se queda congelada o se pierde algún fotograma).

Tipo de Tareas: Periódicas

- Atienden eventos que ocurren constantemente y a una frecuencia determinada. P. ej, destellar un led.
- Reiniciación periódica de tareas, cada instancia debe completar antes de su plazo.
- Por ejemplo, un controlador debe estar controlando 10 veces por segundo la presión de gas en un tanque.

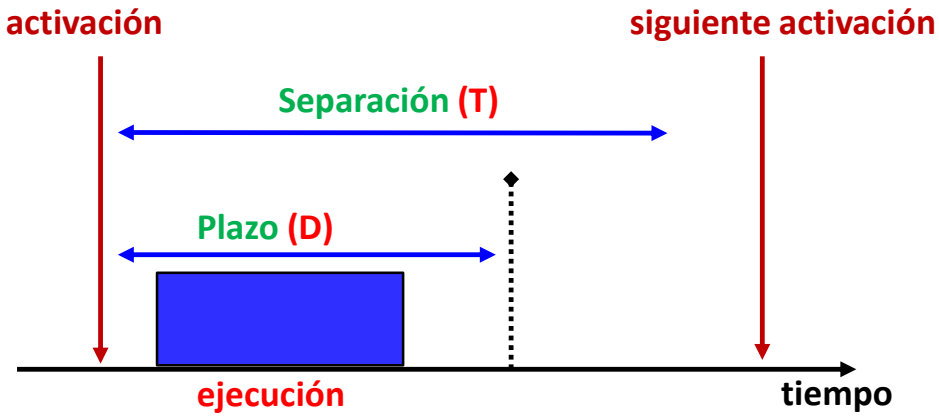


Presión de Gas

T : es el período de la tarea
D : es el plazo de respuesta, relativo al comienzo del período

Tipo de Tareas: Aperiódicas

- Se activan una sola vez, ejemplos: Disparar un misil y verificar el blanco. Una falla de energía en el sistema debe provocar una interrupción la cual debe ser procesada por el sistema
- Atienden eventos que no se sabe cuándo van a darse. Estas tareas están inactivas (bloqueadas) hasta que no ocurre el evento de interés. P. ej, una parada de emergencia.



Disparo de un misil

T es el tiempo mínimo entre activaciones o separación de la tarea
D es el plazo de respuesta, relativo al comienzo del período

Tipo de Tareas: procesamiento Continuo

- Tareas de Son tareas que trabajan en régimen permanente. Por ejemplo, muestrear un buffer de recepción en espera de datos para procesar.
- Estas tareas deben tener prioridad menor que las otras, ya que en caso contrario podrían impedir su ejecución.

consideraciones de hardware y software

A quien afecta la introducción de Tiempo Real ?.

- A los lenguajes, compiladores.
- Al sistema operativo.
- A la arquitectura de hardware.
- A la metodología de diseño.

Consideraciones: de Hardware

- Interfaces eficientes de E/S
- Sistema de interrupciones
- Calculo de números reales
- Relojes en tiempo real
- Protección de memoria

Consideraciones: de SO

- Concurrencia
- Mecanismos de medición del tiempo
- Planificación de tareas para tiempo real
- Acceso a interfaces
- Gestión eficiente de interrupciones
- Mecanismo para la portabilidad

Sistemas Operativos Real Time

Sistemas Operativos Real Time

Consideraciones: de Lenguajes de Programación

- Concurrencia
- Manejo de excepciones
- Mecanismos de medición del tiempo (Control de tiempo)
- Comunicación con dispositivos hardware: Acceso a interfaces
- Gestión de interrupciones
- Portabilidad
- Planificación de la ejecución de tareas
- Control de errores y situaciones excepcionales

Ejemplos:

- ADA, Modula-2, Java-RT, Ocamm.
- C, C++