Real Time Systems

Bibliografía:

- Sistemas Empotrados en Tiempo Real 1ra. Edición - José Daniel Muñoz Frías

- Breve introducción a OSEK-VDX. Un sistema operativo de tiempo real estandarizado. SASE 2015. Mariano Cerdeiro.

- Real-Time Systems and Programming Languages (Fourth Edition) Ada 2005, Real-Time Java and C/Real-Time POSIX - Alan Burns and Andy Wellings

RTS – Introducción

Distintos tipos de Sistemas:

- Sistemas de uso general (ofimática, CAD)
- Sistemas Multimedia (audio/video/juegos)
- Sistemas de control (horno, motor, etc. Ts)

RTS – Introducción

Implementación:

- Sistemas de uso general -> Linux/Windows
- Sistemas Multimedia -> extensiones
- Sistemas de control -> Real Time Systems

RTS – Introducción

Sistemas de Tiempo Real:
Un sistema informático en tiempo real es aquel en el que la **corrección del resultado** depende tanto de su **validez lógica** como **del instante** en que se produce (determinismo).

Determinismo: El tiempo de ejecución de los programas de tiempo real debe estar **acotado** en el caso más desfavorable para cumplir las restricciones temporales. No debe ser necesariamente rápido.

RTS – Introducción

Velocidad o rapidez =/= Determinismo
Acotado en tiempo para el caso mas desfavorable

RTS – Introducción

En los sistemas de control existen cuatro niveles jerárquicos de software:

- Adquisición de datos / Actuadores
  - Ejecuta el Software cuando llega una interrupción
- Algoritmos de control (PID)
  - algoritmo de control digital, toma datos y envia a los actuadores. Se ejecuta periodicamente (periodo de muestreo)
- Algoritmos de supervisión (trayectorias)
  - Capa opcional de control a un nivel superior. Se ejecutan también periódicamente, aunque normalmente con un periodo de muestreo mayor que los algoritmos de control de bajo nivel.
- Interfaz de usuario, registro de datos, etc.
  - no existen restricciones temporales estrictas, por lo que se ejecuta cuando los niveles inferiores no tienen nada que hacer.

RTS – Introducción

Un programa secuencial se divide en funciones que se ejecutan según un orden preestablecido, mientras que un sistema en tiempo real se divide en tareas que se ejecutan en “paralelo”.

La ejecución en paralelo se consigue como la sucesión rápida de actividades secuenciales (multitasking)

RTS – Introducción

Contamos con varias técnicas para ejecutar tareas en paralelo:

- Procesamiento secuencial (Lazo de barrido o scan loop)
- Interrupciones (Background/Foreground)
- Multitarea cooperativo
- Multitarea expropiativo(preemptive)

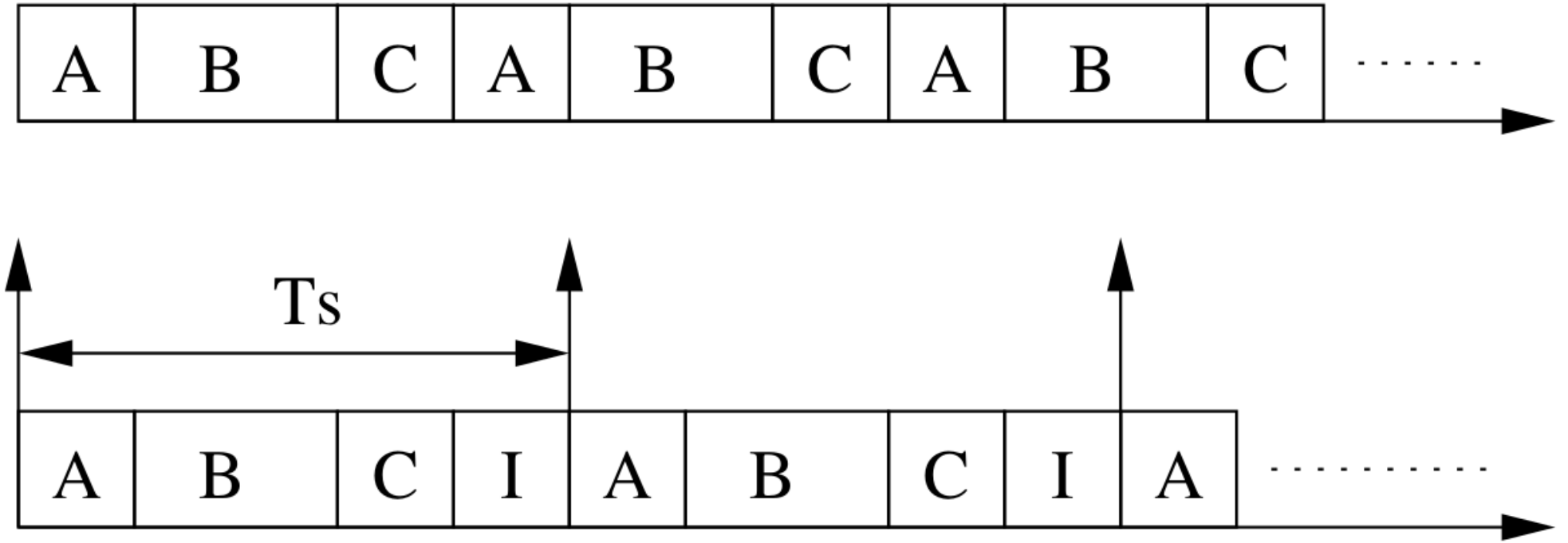
RTS – Proc. Secuencial

Las tareas se ejecutan una tras otra en orden preestablecido.

Condición:

- No debe bloquearse esperando evento externo asíncrono. (y una lectura de un conv A/D?)
- Suma de todos los tiempos max. Ejec < Ts (tiempo de muestreo)
- Si necesitamos un Ts exacto y determinado -> temporizador

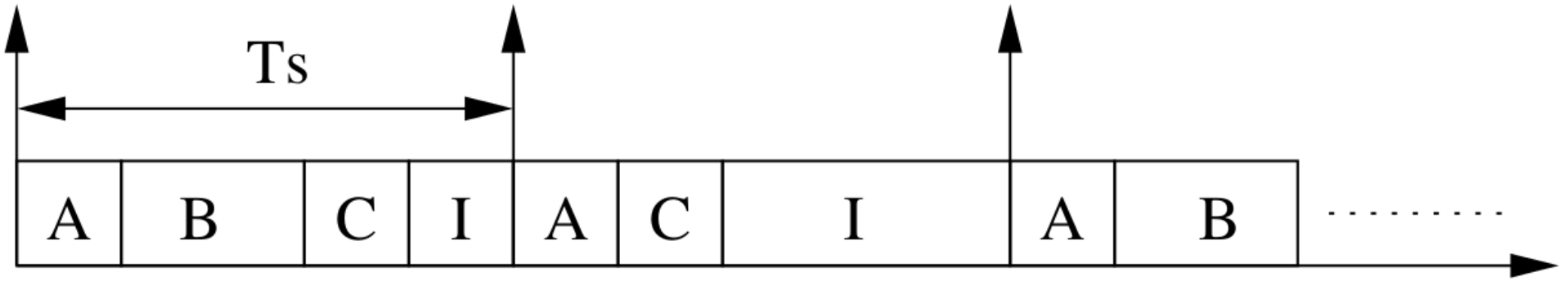
RTS – Proc. Secuencial



RTS – Proc. Secuencial

Problemas:

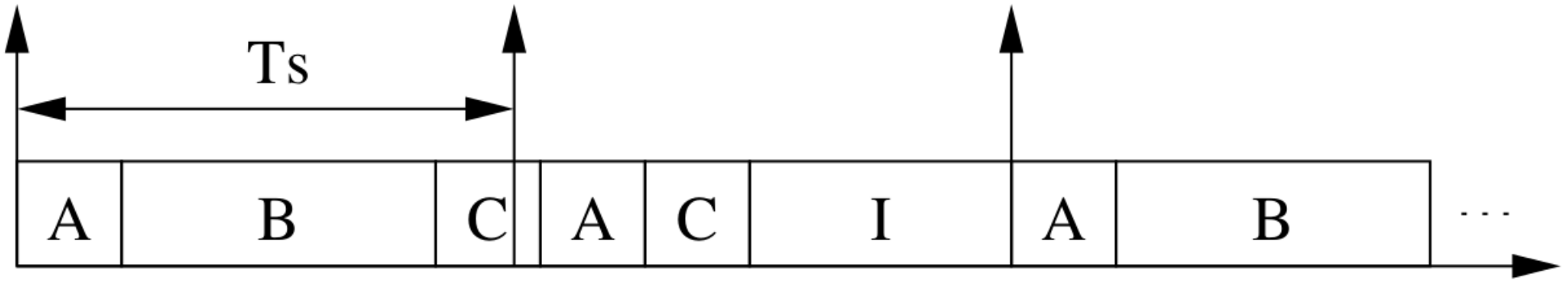
- Si los Ts de las tareas son distintos:
- Ejecutar tarea cada $n_x Ts$



RTS – Proc. Secuencial

Mas Problemas:

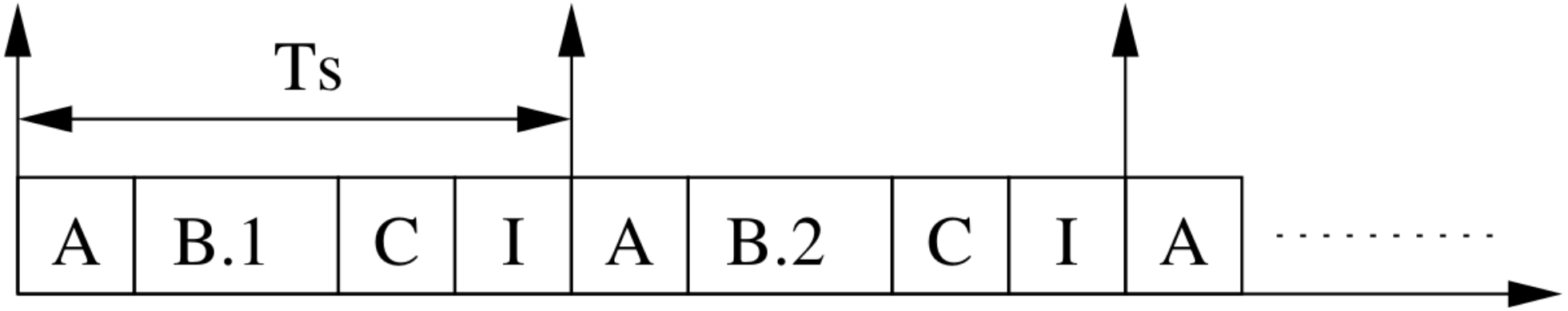
- Si Tiempo ejec. de una tarea es largo:



RTS – Proc. Secuencial

Posibles soluciones:

- Cambio de procesador :-)
- División de tarea, la implementación es complicada a veces



RTS – Proc. Secuencial

Otros Problemas:

- Varias tareas con Ts NO múltiplos de Ts básico y varios tiempos de ejec.
- Latencia (Tiempo entre evento externo y ejecución de la tarea que lo implementa) -> Ts en el peor caso

Soluciones:

- Interrupciones (Background/Foreground)
- Multitarea cooperativo
- Multitarea expropiativo(preemptive)

RTS – Proc. Secuencial

Pros:

- Fácil implementación y depuración
- Fácil compartir datos
- Mayor eficiencia (no hay context switch)

Conts:

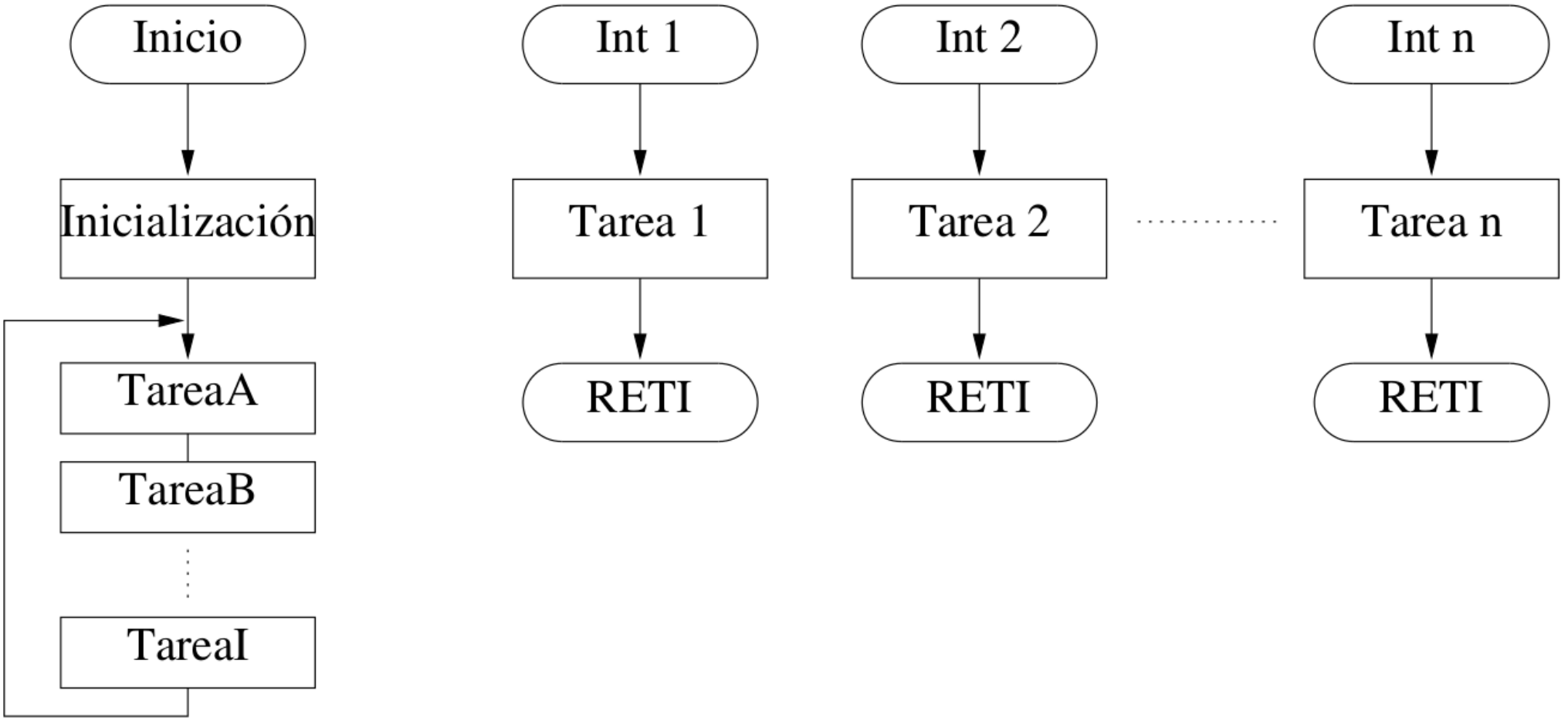
- Latencia asociada al bucle scan
- Difícil para tareas con distinto Ts
- Solo para sistemas sencillos

RTS – Interrupciones

Uso de Interrupciones para esperar eventos externos asíncronos y disminuir Latencia

- Dos tipos de tareas:
  - Foreground
  - Background (atención interrupción)

RTS – Interrupciones



RTS – Interrupciones

Ventajas:

- Latencia => tiempo máximo tarea background (deshabilitación interrupciones)
- No tiene problemas con distintos Ts

RTS – Interrupciones

Conts:

- Necesita Soporte Interrupciones de Hardware
- Cada tarea de background debe asociarse a 1 interrupción.
- Rutinas background deben ser cortas → se lleva parte de la tarea de background a foreground.
- Problemas de concurrencia
- Mas compleja depuración

RTS – Interrupciones

Aspectos a tener en cuenta:

- Datos compartidos (interrupción en zona crítica) -> incoherencia de datos
  - Solución1: ejecución atómica -> deshabilitar interrupciones
  - Solución 2 -> RTOS !!

RTS – RTOS

Que es un RTOS?

- Sistema Operativo para Sistemas de Tiempo Real
- Herramientas para que los programas cumplan compromisos temporales definidos por el programador.
- Comportamiento determinista.
- Un RTOS se emplea cuando hay que administrar varias tareas simultáneas con plazos de tiempo estrictos.

RTS – RTOS

Por que usar RTOS?

- Multitarea: simplifica la escritura del código.
- Escalabilidad: Al tener ejecución concurrente de tareas se pueden agregar las que hagan falta (insertarlas correctamente en el esquema de ejecución del sistema)
- Reusabilidad Código: Si las tareas tienen pocas o ninguna dependencia, es fácil incorporarlas a otras aplicaciones.

RTS – RTOS

Por que usar RTOS? Cont .

- No manejar el tiempo a mano. Los Rtos se encargan del manejo de timers y esperas, ayudando al programador
- Tarea Idle. Si no hay tareas que necesiten procesador, se ejecuta una llamanda idle
- Cumplir requisitos temporales estrictos. Los Rtos aseguran que ocurrido el evento, la respuesta suceda en un tiempo acotado.Ojo esto no lo hace por si mismo, solo da herramientas al programador

RTS – RTOS

Por que NO usar RTOS?

- Uso de tiempo de CPU para cambio de tareas
- Uso de memoria de código para implementar Rtos
- Uso de memoria de datos (TCB)
- El programador NO decide cuando ejecutar cada tarea, queda en manos del planificador.
- Con errores en las reglas de ejecución, puede hacer que los eventos se procesen fuera de tiempo o ni se ejecuten

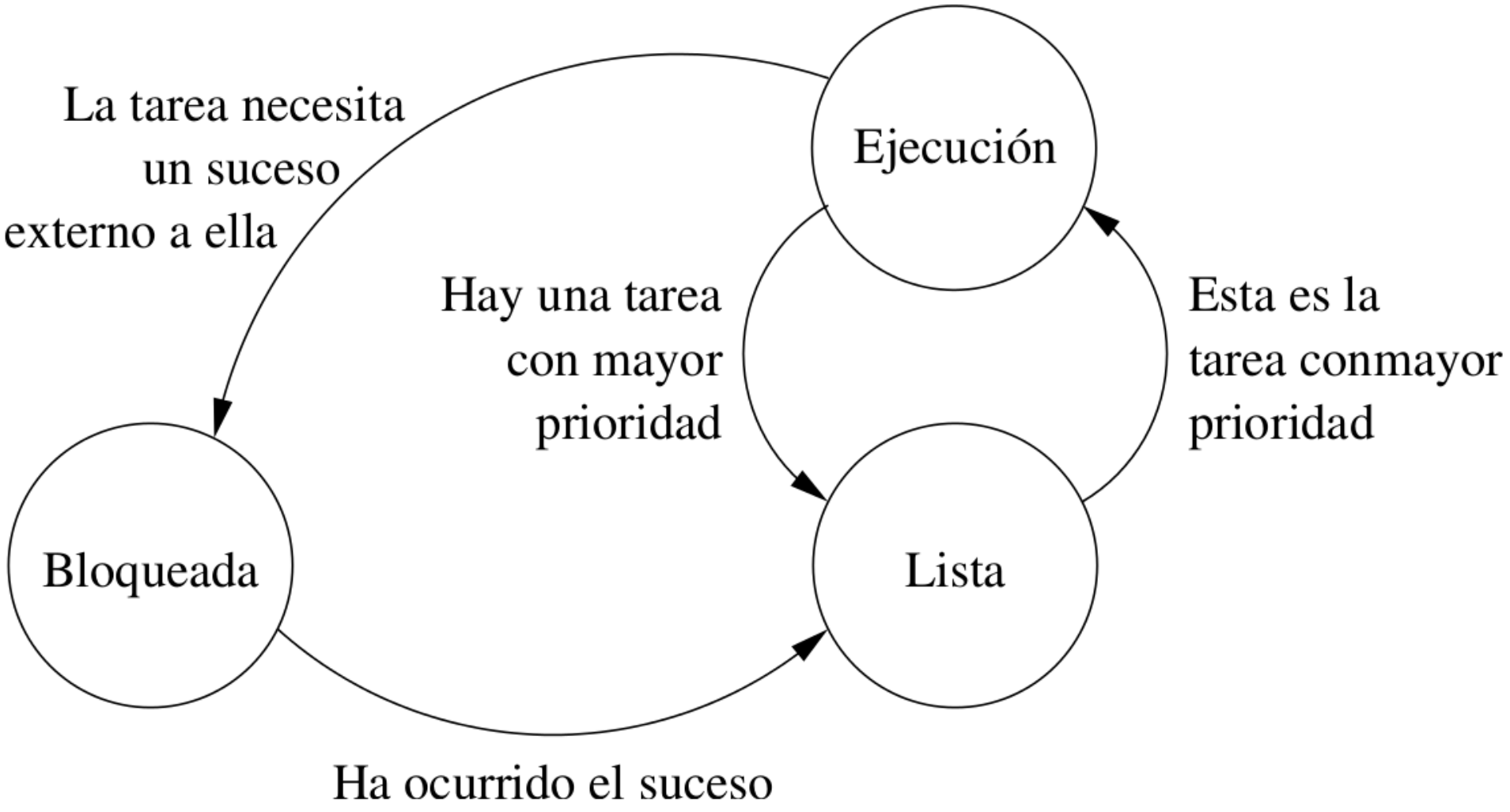
RTS – RTOS

Tareas ?

- Unidad básica de planificación.
- Funciones escritas como si cada una fuera la única que utiliza la CPU.
- Las tareas NO deben terminar nunca.
- Si deja de ser necesaria, se elimina explícitamente
- Se inicializan indicando su prioridad, memoria, función, etc.
- Estados de las tareas

RTS – RTOS

Estados de las Tareas



RTS – RTOS - Componentes

Planificador:

- Planificador cooperativo: las tareas de 1er plano son las responsables de llamar al planificador:
  - Cuando terminan
  - Cuando llevan demasiado tiempo ejecutando(yeld)
  - Cuando se bloquean
- Planificador expropiativo: un timer genera una interrupción y la rutina de anteción de interrupción ejecuta el planificador periodicamente
- El planificador elije la tarea lista de mayor prioridad

RTS – RTOS - Componentes

- Planificador cooperativo:
  - No hay problemas de concurrencia (coherencia)
  - Mas sencilla la codificación
  - No sabe la latencia de las tareas 1er plano

- Planificador expropiativo:
  - Baja latencia
  - La programación de tareas debe ser cuidadosa
  - Gestionar el acceso a recursos compartidos
  - Mas complicada la implementación

RTS – RTOS

Tipo de tareas:

- Tareas Periódicas: con frecuencia determinada.
  Gralmente se usa un timer y un contador. Ej. Destello de led.
- Tareas Aperiódicas: con frecuencia intederminada.
  Tareas inactivas (bloqueadas) hasta que ocurre el evento de interés. P. ej, una parada de emergencia.
- Tareas Contínuas: régimen permanente.
  Ej. muestrear un buffer de recepción en espera de datos para procesar.
  Estas tareas deben tener prioridad menor que el resto, ya que en caso contrario podrían impedir su ejecución

RTS – RTOS

Tareas Periódicas:

- La mayor parte de su tiempo en el estado Bloqueado hasta que expira su tiempo de bloqueo, en este momento pasa al estado Listo.
- Alta prioridad, para tener baja latencia.

RTS – RTOS

Tareas Aperiódicas:

- Nos interesa que permanezca bloqueada hasta que ocurra un evento determinado (interno o externo)
- Queremos que la tarea ceda el CPU a las de menor prioridad, ya que no va a estar haciendo nada

RTS – RTOS

Recursos de Sincronización:

- Semáforos:
  - Su función es restringir el acceso a una sección particular del programa.
  - Dos primitivas , incrementar y decrementar.
  - Semáforos binarios.
  - En algunos Rtos se define máximo tiempo de bloqueo

RTS – RTOS

Recursos para Intercambio de datos:

- Tareas son funciones de C. Reglas de visibilidad.
- Variables globales como mecanismo de intercambio.

  - Problemas de acceso al recurso
  - Tareas aperiódicas que esperan un dato para procesarlo
  - Necesidad de sincronización

RTS – RTOS

Recursos para Intercambio de datos: Cont.

- Colas de mensaje:
  - Son visibles por todas las tareas (deben ser creadas en forma global)
  - Incorporan el mecanismo de sincronización.
  - De la misma manera que un semáforo, se puede bloquear al leer / escribir datos de una cola
  - En algunos Rtos se define máximo tiempo de bloqueo
  -

RTS – RTOS

Administración de recursos:

- Se debe hacer exclusión mutua al código del recurso:
  - Deshabilitar Interrupciones
  - Suspender el planificador
  - Aumentar la prioridad de la tarea que usa el recurso
  - Uso de semáforo o mutex