**APUNTES TEMA 1**

Sistemas en Tiempo Real (RTA):

* Es aquel en el que la exactitud del cómputo realizado no sólo depende de la exactitud lógica del cómputo, sino también del instante de tiempo en el que se produjo.
* Características Principales:
  + Las RTAs no son aplicaciones que se ejecuten rápidamente.
  + Son aplicaciones determinadas en el tiempo -tiempos de respuesta constantes-
  + Pueden existir pequeñas desviaciones ocasionales en el tiempo de respuesta, en términos de milisegundos o segundos.
  + Por otro lado, los sistemas “Hard RTAs” deben cumplir todas sus funciones en un tiempo muy estricto.

Sistemas Operativos en Tiempo Real (RTOS):

* Un RTOS está especialmente diseñado para procesar aplicaciones que requieran tiempos precisos con un alto grado de fiabilidad y tiempos predecibles y determinísticos.
* Para ser considerado como “tiempo-real”, el sistema operativo debe saber el tiempo máximo ejecución de cada operación crítica. En esta categoría tenemos:
  + Procesamiento de las interrupciones y excepciones internas del sistema (System Exceptions)
  + Secciones críticas (Critical Sections): proteger zonas críticas ante un posible context switch o interrupción.
  + Mecanismo de planificación del Kernel (Scheduling): Preemptive (expropiativo), cooperativo, modalidad Round-Robin.
  + Objetos de kernel: Queue, Semáforos binarios, Mutexes, etc.
* Ventajas:
  + Permite dividir la aplicación en tareas.
  + Añadir tareas de baja prioridad no afecta a la capacidad de respuesta de las tareas de alta prioridad.
  + El código middleware puede añadirse al sistema con facilidad.
  + Mayor control de los plazos de desarrollo de una aplicación.
  + Permiten portabilidad a diferentes arquitecturas o CPUs.
  + Resulta muy sencillo gestionar los recursos energía.
* Desventajas:
  + Al ser código extra que debe ser añadido al sistema, se requiere mayor memoria FLASH. Típicamente entre 6 a 20 Kbytes.
  + Precisamos mayor memoria RAM, ya que cada Task requiere de su propia stack de RAM individual. Aproximadamente 256 bytes de memoria RAM para variables propias del RTOS.
  + Los servicios de un RTOS consumen ciclos de CPU y añaden carga de trabajo al sistema, típicamente del 2% al 10%.
  + Para cada RTOS seleccionado, existe una curva de aprendizaje diferente.

RTOS vs GPOS: Planificador de tareas.

* Los GPOS están programados para planificar las tareas de modo que se consiga una alta cantidad de procesos en una unidad de tiempo. Este parámetro se conoce como Throughput. En muchas ocasiones, la ejecución de los procesos de alta prioridad se puede ver retrasada con el fin de procesar un mayor número de tareas de menor prioridad.
* Por otro lado, los RTOS ejecutan sus procesos en función de su prioridad. Si una tarea está lista para ser ejecutada, tomará el control de la CPU de cualquier otra tarea de menor prioridad que esté ejecutándose.
* Todas las tareas de baja prioridad se mantienen en pausa si existe una tarea de alta prioridad lista para ejecutarse.
* En cuanto a la tasa de procesamiento de tareas por unidad de tiempo, los RTOS muestran una menor tasa que los GPOS, debido a la “restricción” de ejecutar siempre la tarea de mayor prioridad.

RTOS vs GPOS: Latencia.

* En el ámbito de la computación, la latencia hace referencia al tiempo que transcurre entre un estímulo y la ejecución de la función asociada.
* En RTOS, cuando hablamos de conmutación entre tareas, la latencia hace referencia a los ciclos de reloj transcurridos entre un evento y el tiempo que toma la tarea en tomar el control de la CPU para poder procesar dicho evento.
* En los GPOS, el tiempo de latencia puede variar significativamente.
* En los RTOS, el tiempo de latencia permanece prácticamente constante.

RTOS vs GPOS: Inversión de Prioridades.

* La inversión de prioridad se produce cuando una tarea de menor prioridad accede previamente a un recurso compartido. El efecto producido es el siguiente:
  + La tarea de baja prioridad se comporta como una tarea de alta prioridad.
  + La tarea de alta prioridad se comporta como una de baja prioridad.
* Una posible solución a este problema es la denominada “Herencia de Prioridades”. La tarea de baja prioridad hereda temporalmente la prioridad más alta. De este modo se consigue liberar el recurso compartido.

Arquitectura Super-Loop:

* No hace uso de interrupciones.
* Solo hay un único thread -por lo que no hay variables o datos compartidos con otros hilos-.
* Se aplica a sistemas simples.
* Se usan funciones de tipo non-reentrant -es aquella función que no puede ser compartida por más de un hilo o tarea a menos que exista algún método de exclusión, no deben usarse en sistemas concurrantes-.
* Latencias y tiempos de respuesta.
* El sistema es muy dependiente de la estructura general del código.

Arquitectura Foregroind-Background:

* Se trata de una arquitectura Super-Loop.
* Las interrupciones forman la parte Foreground.
* En el background disponemos de un único thread de baja prioridad.
* Los threads pueden comunicarse entre sí mediante variables globales.

Función de tipo Reentrant y Non-Reentrant:

* En sistemas en los que se dispone de un único procesador, una función se denomina reentrante cuando puede ser invocada con seguridad en un sistema concurrente.
* Las restricciones básicas para considerar una función como reentrant son:
  + La función no debe contener datos estáticos durante llamadas sucesivas.
  + No debe retornar un puntero que apunte a un dato estático.
  + Debe usar variables locales.
  + No debe llamar a funciones no-reentrantes.

Conceptos Erróneos sobre la Computación en Tiempo Real:

* La computación en tiempo real es equivalente a procesar líneas de código a alta velocidad.
* En las RTA se obtiene mayor rendimiento.
* Disponer de más procesadores, memoria RAM, e interfaces permite transformar nuestro sistema a una RTA.

La Realidad sobre la Computación en Tiempo Real:

* La computación en tiempo real es equivalente a computación PREDECIBLE.
* En las RTA los tiempos de criticidad son más importantes que el rendimiento.
* Un sistema en tiempo real trabaja con GARANTIAS, no promesas.

Código de Creación de Tasks:

Texto

Descripción generada automáticamente

Texto, Carta

Descripción generada automáticamente