**APUNTES TEMA 2**

Simple Cyclic:

* Conceptos e Implementación:
  + Se trata de una de las soluciones más simples, donde el scheduler asigna el control total de la CPU a la primera tarea de la cola, con la diferencia de que no se libera el uso de la CPU hasta que la tarea complete todo el proceso que se le ha asignado. Esto se denomina Run to completion.
  + Una vez que la primera tarea de la cola completa sus procesos, se libera el uso de la CPU y se asigna a la siguiente tarea de la cola. Este cambio de tarea es controlado por el scheduler/dispatcher.
  + La primera tarea completa sus actividades, libera la CPU y el scheduler asigna a una nueva tarea el tiempo de CPU. Este proceso se repite hasta completar todas las tareas de la cola. El scheduler trabaja como una FIFO.
* Se utiliza en sistemas en los que:
  + Las tareas no son críticas y no tiene tanta importancia como afectan al sistema.
  + No son críticas en el tiempo.
  + No importa que las tareas se ejecuten hasta completar sus actividades.
  + No se precisa de una tarea IDLE.

Timed Cyclic:

* Conceptos e implementación:
  + Este tipo de scheduling es una modificación del anterior, para adaptarse a las necesidades de lo que suelen requerir la ejecución de procesos en un sistema embebido (a nivel básico).
  + El problema general, es que el conjunto de tareas debe poder ejecutarse a intervalos de tiempo que deben estar predeterminados.
* Limitaciones:
  + Si una tarea falla, será imposible liberar el uso de la CPU.
  + Añadir una nueva tarea al sistema implicará que será colocada al final de la cola y los tiempos de reacción son lentos ante peticiones de ejecución de nuevas tareas.
  + El sistema es gravemente afectado por tareas de larga duración -puede solucionarse permitiendo que las tareas de larga duración liberen deliberadamente el uso de la CPU. Este proceso se denomina Interleaving-.

Cooperative Scheduling:

* Este tipo de scheduling se basa en la solución al anterior punto de las limitaciones del Time Cyclic Scheduling.
* Se trata de ofrecer al usuario la posibilidad de aplicar interleaving. Cuando se dispone de esta funcionalidad y se aplica al sistema, estaremos trabajando de manera cooperativa.
* Limitaciones:
  + Para mejorar el comportamiento del sistema y no depender de una función de yielding o liberación de CPU, a cada tarea se la asigna un tiempo máximo de ejecución (Time Quanta). Esto se conoce como Time Slicing (Round-Robin Scheduling).

Round-Robin Scheduling:

* Cuando dos o más Tasks tienen la misma prioridad, el propio planificador asigna un tiempo de ejecución a cada una de ellas.
* Asigna unos tiempos de trabajo (time quanta) a cada una de las Tasks.
* La unidad de tiempos del time quanta es medida en Ticks del RTOS.

Scheduling: Ready List.

* Una vez hemos creado las Tasks de nuestra aplicación (OSTaskCreate()), el kernel gestiona las tareas que están listas para ejecutarse (Ready to Run State) mediante la Ready List.
* La Ready List consta de dos partes diferenciadas:
  + Bitmap (Prio List) con los niveles de prioridad, correspondientes a las Tasks listas para ser ejecutadas.
  + Tabla (Ready List) que contiene los punteros o direccionamientos a cada una de las Tasks anteriores.
* El número máximo de niveles de prioridad es establecido mediante OS\_CFG\_PRIO\_MAX, dentro del header de configuración del RTOS os\_cfg.h.
* En uC/OS-III, valores numéricos crecientes indican niveles de prioridad más bajos. Por tanto, un nivel numérico 0 implica la prioridad más elevada del sistema.
* Dentro de cada TCB, disponemos de dos punteros, que se utilizan para enlazar y direccionar Tasks del mismo nivel de prioridad de la lista ready to run:
  + .PrevPtr .
  + .NextPtr.