**APUNTES TEMA 3**

Kernel Objects: Semáforo:

* Los semáforos son mecanismos de bloqueo, sincronización y control de recursos en software que se utilizan en la ejecución de aplicaciones paralelas. En μC-OS-III, hay dos tipos de semáforos disponibles: semáforos binarios y semáforos contadores.
* Los semáforos son objetos del kernel y se definen mediante la estructura OS\_SEM en el archivo header os.h. Es necesario agregar los semáforos a la estructura de datos predeterminados y se pueden crear tantos semáforos como la memoria RAM disponible lo permita.
* La liberación del recurso compartido se realiza mediante la función OSSemPost(), mientras que la espera de la liberación del semáforo se realiza mediante la función OSSemPend().

Kernel Objects: Mutex:

* La inversión de prioridades ocurre cuando una tarea de alta prioridad se comporta como una de baja prioridad debido al acceso a un recurso compartido que también es deseado por una tarea de baja prioridad.
* La duración de esta inversión de prioridades puede variar según la duración de otros procesos de mediana prioridad y se conoce como "unbounded priority inversion" en la literatura de sistemas operativos en tiempo real.
* Para controlar los recursos compartidos, se utilizan mutexes, que son objetos del kernel en RTOS. Los mutexes funcionan de manera similar a los semáforos y se utilizan para evitar la inversión de prioridades.
* A diferencia de los semáforos, los mutexes permiten una herencia temporal de prioridades, lo que significa que una tarea de baja prioridad que tiene acceso al recurso compartido puede tomar temporalmente la prioridad de una tarea de alta prioridad que necesita acceder al mismo recurso. Esto permite que la tarea de alta prioridad obtenga rápidamente el recurso compartido y evita que las tareas de prioridad media interfieran en el proceso.
* El Mutex se aplica del mismo modo que los semáforos. En el código se deben seguir los siguientes pasos:
  + Esperar a la liberación del recurso compartido. Esta operación se realiza mediante OSMutexPend().
  + Una vez liberado, se procede a acceder al recurso compartido.
  + Después de su uso, se debe liberar el recurso compartido. Esto se realiza mediante OSMutexPost().
* Es importante destacar que la severidad de la inversión de prioridades causada por semáforos depende de las tareas de prioridad media que se encuentran entre la tarea de alta prioridad y la tarea de baja prioridad.
* Funciones del Mutex:

Tabla

Descripción generada automáticamente

Kernel Objects: Message Queue:

* El objeto de kernel llamado "Message Queue" se utiliza para realizar comunicaciones entre tareas o entre interrupciones y tareas. También se puede utilizar como método de sincronización, además de transmitir y recibir mensajes.
* En μC-OS-III, existen diferentes tipos de buffers de mensajes. Por un lado, están los buffers individuales que cada tarea tiene y se utilizan para comunicarse con una tarea específica. Por otro lado, se pueden crear buffers externos para comunicaciones que involucren a más de dos tareas, conocido como "broadcasting".
* Definiciones:
  + Broadcasting: Se usa cuando una tarea o ISR transmite un mensaje a una cola de mensajes para ser recibido por varias tareas de la aplicación. Se usan funciones correspondientes a buffer de mensajes externo.
  + Task Message Queue: μC-OS-III incorpora su propio buffer de mensajes que todas las tareas incluyen a modo de cola de mensajes individual. Esto se usa para evitar que varias tareas estén esperando por el mismo mensaje en una cola de mensajes externa común.
  + Task Message Queue: Para habilitar el servicio de buffer interno de mensajes de μC-OS-III, que por defecto puede estar deshabilitado, se deben definir algunos parámetros en el archivo os\_cfg.h.
  + Bilateral Rendez-vous (Sincronización bilateral): Permite sincronizar las actividades de dos tareas. Cada cola de mensajes almacena como máximo un único mensaje.
* Funciones:

Imagen que contiene Texto

Descripción generada automáticamente

Tabla

Descripción generada automáticamente