FreeRTOS

Sistemas operativos de tiempo real

FreeRTOS

- Es un sistema operativo de tiempo real fundado por **Richard Barry**, su kernel ha sido portado a mas de 35 plataformas de microcontrolador para sistemas embebidos. Está distribuido bajo la licencia MIT.
- Está diseñado para ser pequeño y simple. El núcleo en sí consta de solo tres archivos implementados en C. Para hacer que el código sea legible, fácil de portar y mantener, está escrito principalmente en C.

Estructura

- BaseType_t **xTaskCreate**(...);

  Crea e inicializa los espacios de memoria para la ejecución de una tarea.

  La tarea inicia en estado READY. Se la puede llamar en cualquier momento.

Parametros:

- **pvTaskCode**: referencia a la función de C que describe el comportamiento.
- **pcName**: Nombre descriptivo.
- **usStackDepth**: Tamaño del stack en words (valor mínimo configMINIMAL_STACK_SIZ).
- **pvParameters**: Parámetro opcional a la tarea (permite varias instancias d

la misma tarea, con comportamientos distintos).

- **uxPriority** : prioridad de la tarea. Debe ser mayor a tskIDLE_PRIORITY.
- **pxCreatedTask**: Handle de la tarea, si es requerido.

Funciones

- void **vTaskDelete**(TaskHandle_t xTask); elimina una tarea, liberando espacio de memoria asociado.

  Pasándole como parámetro NULL elimina a la tarea que llamó a la función.

- void **vTaskSuspend**(TaskHandle_t xTask).
- void **vTaskResume**(TaskHandle_t xTask).

Temporizaciones

- void **vTaskDelay**(const TickType_t xTicksToDelay);

Temporización de FreeRTOS que bloquea la tarea que la llama durante los xTicksToDelay de tiempo. Produce una demora en la tarea que la llama. Cede el control del CPU mientras este tiempo no expira.

- Void **vTaskDelayUntil**(TickType_t *pxPreviousWakeTime,const TickType_t xTimeIncrement);

  Asegura un delay entre cada uno de los llamados a esta función. Cede el control del CPU mientras este tiempo no expira.

SEMÁFOROS BINARIOS

- SemaphoreHandle_t mi_semaforo =**xSemaphoreCreateBinary**();

  Crea semáforo binario.

- **xSemaphoreTake**(mi_semaforo,TickType_t xTicksToWait);

  Toma el semáforo, retorna **pdTRUE** si se pudo tomar correctamente y **pdFALSE** si durante **xTicksToWait ticks** nadie lo da.

- **xSemaphoreGive**(mi_semaforo);

  Dá el semáforo.

MUTEX

Aplica cuando un recurso se utiliza durante un tiempo considerable desde varias tareas, de distintas prioridades.

Usa la misma API de semáforos, salvo la función para crearlo:

- SemaphoreHandle_t MUT = **xSemaphoreCreateMutex**(void)
- **xSmaphoreTake**(MUT, portMAX_DDELAY)
- **xSemaphoreGive**(MUT);

COLAS

Se utilizan en un modelo de funcionamiento productor/consumidor de datos, por ejemplo cuando los datos se producen más rápido que lo que se los consume.

- QueueHandle_t mi_cola = **xQueueCreate**(UBaseType_t uxQueueLength,UBaseType_t uxItemSize);
- uxItemSize = Tamaño en bytes de cada elemento
- uxQueueLength = Cantidad de elementos en la cola

COLAS

BaseType_t **xQueueReceive**(QueueHandle_t xQueue,void* pvBuffer, TickType_t xTicksToWait);

- xQueue = Nombre de la cola con el valor devuelto por **xQueueCreate**
- **pvBuffer** = Dirección de memoria del lugar en donde se almacenará el elemento removido
- **xTicksToWait** = Tiempo en ticks que como máximo deberá bloquearse la tarea en caso de que la cola esté vacía.

La funcion retorna:

- pdTRUE: Si el elemento se recibió correctamente.
- pdFALSE: Si el elemento no se recibió , y el llamado dio timeout.

COLAS: otras funciones

- **xQueuePeek**: Consulta un elemento de la cola sin removerlo. Opera igual que xQueueReceive.
- **uxQueueMessagesWaiting**: Devuelve la cantidad de mensajes esperando ser removidos de la cola.
- **uxQueueSpacesAvailable**: Devuelve la cantidad de espacios para mensajes, disponible en la cola.
- **vQueueDelete**: Destruye la cola, liberando toda memoria dinámica que haya necesitado cuando se llamó a xQueueCreate.
- **xQueueReset**: Restablece la cola vaciándola, volviendo a su estado inicial (cuando se llamó a xQueueCreate).

Bibliografía

- Developer document: https://www.freertos.org/features.html