



Práctica 2

SISTEMAS TOLERANTES A FALLOS
EN TIEMPO REAL

Sara Marcos Cornejo
José Francisco Romero Rodríguez



UNIVERSIDAD
NEBRIJA

Índice:

- **Introducción.....3**
- **Procedimiento, código.....3**
- **Resultados.....7**
- **Conclusiones.....6**

Introducción:

Round-Robin se trata de un algoritmo de planificación de CPU utilizado en sistemas operativos para programar procesos o tareas en un sistema informático. En un sistema en tiempo real, la fiabilidad es crítica y se deben tomar medidas especiales para garantizar que las tareas se ejecuten dentro de un límite de tiempo específico y de manera confiable.

El objetivo principal del algoritmo de round-robin es permitir que múltiples procesos o tareas tengan acceso a la CPU de manera justa y equitativa, compartiendo el tiempo de procesamiento de manera rotativa. Además, el round-robin es capaz de manejar procesos con diferentes prioridades, lo que permite darle prioridad a los procesos más importantes o críticos en caso de ser necesario.

El algoritmo funciona en base a un tiempo de quantum, que es un intervalo de tiempo fijo en el que cada proceso puede ejecutarse. Trabaja de la siguiente manera:

Se asigna a cada tarea un intervalo de tiempo determinado (cuanto tiempo cada tarea puede ejecutarse en la CPU) y se ejecutan las tareas de acuerdo con este intervalo. Cuando una tarea alcanza su intervalo de tiempo asignado, se pausa y se coloca al final de la cola de tareas. La siguiente tarea en la cola comienza a ejecutarse y así sucesivamente. Este proceso continúa hasta que todas las tareas hayan sido ejecutadas en su intervalo de tiempo asignado o se produzca un evento que interrumpa la ejecución normal del sistema.

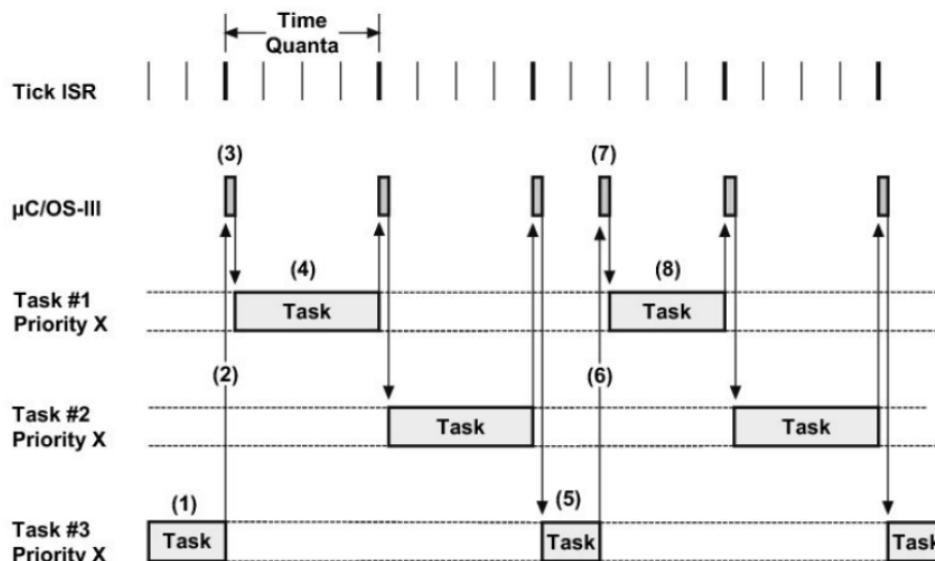


Figure - Round Robin Scheduling

Procedimiento, código:

Para esta práctica, se ha implementado RTOS con planificación Round-Robin. Para ello, se ha utilizado la práctica 1 como base, así como un HyperTerminal para realizar comprobaciones de acuerdo con una placa.

En primer lugar, se ha tenido que activar este flag, que permite habilitar o deshabilitar el algoritmo Round Robin para la planificación de tareas.

Es necesario indicarlo como configuración del Kernel, en tiempo de precompilación:

```
#define OS_CFG_SCHED_ROUND_ROBIN_EN    1u    /* Include code for Round-Robin scheduling
```

Este flag se debe modificar en el fichero os_cfg.h

Al igual que en la anterior práctica, se trabajará con 6 tareas. Se utiliza la función OSTaskCreate(), pasándole como parámetros:

- la dirección de la TCB
- el nombre de la tarea
- la dirección de la función de la tarea
- el parámetro de la función de la tarea
- la prioridad de la tarea
- la dirección de la stack de la tarea
- el tamaño de la stack de la tarea
- el tamaño de los mensajes que se van a enviar a la tarea
- el tiempo de espera de la tarea
- el parámetro de la función de la tarea
- las opciones de la tarea
- el puntero al error

Es importante tener en cuenta que la planificación Round-Robin solo se activará cuando haya al menos dos o más tareas con la misma prioridad. Por lo tanto, se han creado varias tareas con la misma prioridad para poder utilizar esta técnica de planificación. Cada tarea tiene una función específica que debe ejecutarse durante el tiempo de procesamiento asignado.

Tendrían la siguiente forma:

<pre> OSTaskCreate((OS_TCB *)&task_led1_TCB, (CPU_CHAR *)"Tarea 1. Control led y mensaje uart", (OS_TASK_PTR)TASK_LED1, (void *)0, (OS_PRIO)3, (CPU_STK *)&task_led1_STK[0], (CPU_STK_SIZE)0u, (CPU_STK_SIZE)1024u, (OS_MSG_QTY)0u, (OS_TICK)0u, //10*System Tick period (void *)0, (OS_OPT) (OS_OPT_TASK_STK_CHK OS_OPT_TASK_STK_CLR), (OS_ERR *)&os_err); </pre>	<pre> OSTaskCreate((OS_TCB *)&task_led2_TCB, (CPU_CHAR *)"Tarea 2. Control led y mensaje uart", (OS_TASK_PTR)TASK_LED2, (void *)0, (OS_PRIO)3, (CPU_STK *)&task_led2_STK[0], (CPU_STK_SIZE)0u, (CPU_STK_SIZE)1024u, (OS_MSG_QTY)0u, (OS_TICK)0u, //10*System Tick period (void *)0, (OS_OPT) (OS_OPT_TASK_STK_CHK OS_OPT_TASK_STK_CLR), (OS_ERR *)&os_err); </pre>
<pre> OSTaskCreate((OS_TCB *)&task_led3_TCB, (CPU_CHAR *)"Tarea 3. Control led y mensaje uart", (OS_TASK_PTR)TASK_LED3, (void *)0, (OS_PRIO)3, (CPU_STK *)&task_led3_STK[0], (CPU_STK_SIZE)0u, (CPU_STK_SIZE)1024u, (OS_MSG_QTY)0u, (OS_TICK)0u, //10*System Tick period (void *)0, (OS_OPT) (OS_OPT_TASK_STK_CHK OS_OPT_TASK_STK_CLR), (OS_ERR *)&os_err); </pre>	<pre> OSTaskCreate((OS_TCB *)&task_led4_TCB, (CPU_CHAR *)"Tarea 4. Control led y mensaje uart", (OS_TASK_PTR)TASK_LED4, (void *)0, (OS_PRIO)3, (CPU_STK *)&task_led4_STK[0], (CPU_STK_SIZE)0u, (CPU_STK_SIZE)1024u, (OS_MSG_QTY)0u, (OS_TICK)0u, //10*System Tick period (void *)0, (OS_OPT) (OS_OPT_TASK_STK_CHK OS_OPT_TASK_STK_CLR), (OS_ERR *)&os_err); </pre>
<pre> OSTaskCreate((OS_TCB *)&task_led5_TCB, (CPU_CHAR *)"Tarea 5. Control led y mensaje uart", (OS_TASK_PTR)TASK_LED5, (void *)0, (OS_PRIO)3, (CPU_STK *)&task_led5_STK[0], (CPU_STK_SIZE)0u, (CPU_STK_SIZE)1024u, (OS_MSG_QTY)0u, (OS_TICK)0u, //10*System Tick period (void *)0, (OS_OPT) (OS_OPT_TASK_STK_CHK OS_OPT_TASK_STK_CLR), (OS_ERR *)&os_err); </pre>	<pre> OSTaskCreate((OS_TCB *)&task_led6_TCB, (CPU_CHAR *)"Tarea 6. Control led y mensaje uart", (OS_TASK_PTR)TASK_LED6, (void *)0, (OS_PRIO)3, (CPU_STK *)&task_led6_STK[0], (CPU_STK_SIZE)0u, (CPU_STK_SIZE)1024u, (OS_MSG_QTY)0u, (OS_TICK)0u, //10*System Tick period (void *)0, (OS_OPT) (OS_OPT_TASK_STK_CHK OS_OPT_TASK_STK_CLR), (OS_ERR *)&os_err); </pre>

A continuación, para cada tarea se le añade la función de Round-robin mediante la función OSSchedRoundRobinCfg. Al establecer su parámetro en DEF_ENABLED, se ha habilitado esta función y se ha configurado un tick de 100u. Si hay algún problema, se utilizará el puntero &os_err para indicar el tipo de error que se ha producido

```

void TASK_LED1 (void)
{
    OS_ERR os_err;
    OSSchedRoundRobinCfg(DEF_ENABLED, (OS_TICK)100u, &os_err);
    while(1)
    {
        sprintf(txdata, "Practica 2, tarea 1 de Jose y Sara \r\n");
        EnviarString(txdata, Uart3);
        LED_RED = !LED_RED_Read;
        OSTimeDly(500, OS_OPT_TIME_DLY, &os_err); //500 ticks del RTOS -> 500ms
    }
}

```

Permite que el sistema opere con un algoritmo específico para gestionar el orden de ejecución de las tareas, con la opción de saber si hay algún error.

Se ha utilizado un programa llamado HyperTerminal que permite establecer una comunicación serial con la placa de experimento. La placa de experimento es un dispositivo electrónico diseñado para llevar a cabo diferentes tipos de experimentos y pruebas.

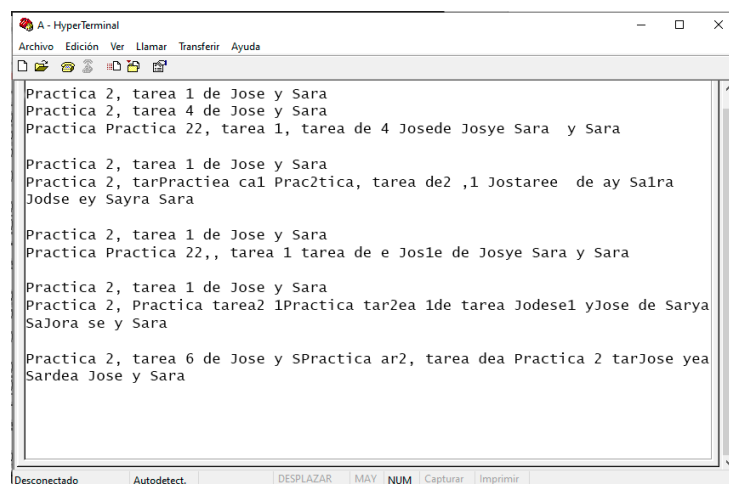
Al establecer una comunicación serial entre el programa HyperTerminal y la placa de experimento, se pueden observar en tiempo real los datos que se están transmitiendo entre ambas. De esta forma, es posible verificar el correcto funcionamiento de la práctica y detectar posibles errores o problemas en la transmisión de datos.

Resultados:

Los resultados obtenidos son los siguientes:

Resultado 1:

En este resultado, se muestra que los datos obtenidos están entremezclados. Esto se debe a que la velocidad de transmisión (baudrate) estaba configurada en 9600 baudios, lo que significa que los datos se estaban enviando a una velocidad más lenta. Además, los ticks se establecieron en 10u al crear las tareas al principio. Los ticks se utilizan para medir el tiempo y programar la ejecución de las tareas. Por lo tanto, una vez que se ejecutan varias tareas juntas, llega un momento en que el sistema no puede manejar más tareas y se produce una especie de "saturación", lo que resulta en datos entremezclados y caóticos.



The screenshot shows a HyperTerminal window titled 'A - HyperTerminal'. The menu bar includes 'Archivo', 'Edición', 'Ver', 'Llamar', 'Transferir', and 'Ayuda'. The toolbar contains icons for file operations and communication. The main text area displays several lines of garbled text, which are repetitions of 'Practica 2, tarea 1 de Jose y Sara' and 'Practica 2, tarea 4 de Jose y Sara' that have become mixed and corrupted due to a baud rate of 9600. The status bar at the bottom shows 'Desconectado', 'Autodetect.', and buttons for 'DESPLAZAR', 'MAY', 'NUM', 'Capturar', and 'Imprimir'.

```
Practica 2, tarea 1 de Jose y Sara
Practica 2, tarea 4 de Jose y Sara
Practica Practica 22, tarea 1, tarea de 4 Josede Josye Sara y Sara

Practica 2, tarea 1 de Jose y Sara
Practica 2, tarPractica ca1 Prac2tica, tarea de2 ,1 Jostaree de ay Salra
Jodse ey Sayra Sara

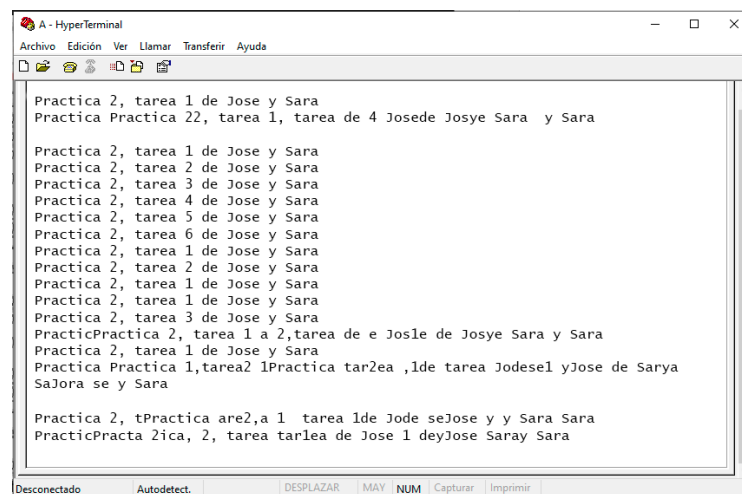
Practica 2, tarea 1 de Jose y Sara
Practica Practica 22,, tarea 1 tarea de e Josle de Josye Sara y Sara

Practica 2, tarea 1 de Jose y Sara
Practica 2, Practica tarea2 1Practica tar2ea 1de tarea Jodesel yJose de Sarya
SaJora se y Sara

Practica 2, tarea 6 de Jose y SPractica ar2, tarea dea Practica 2 tarJose yea
Sardea Jose y Sara
```

Resultado 2:

Se ha modificado el valor de los "ticks" al crear las tareas a 0u, lo que significa que se ha configurado la función de OSSchedRoundRobinCfg de forma predeterminada. Esto ha permitido que más tareas puedan ser realizadas consecutivamente al principio, pero a medida que se satura la cantidad de envíos de datos, el comportamiento se vuelve más caótico y los datos se vuelven menos precisos.



```
A - HyperTerminal
Archivo  Edición  Ver  Llamar  Transferir  Ayuda

Practica 2, tarea 1 de Jose y Sara
Practica Practica 22, tarea 1, tarea de 4 Josede Josye Sara  y Sara

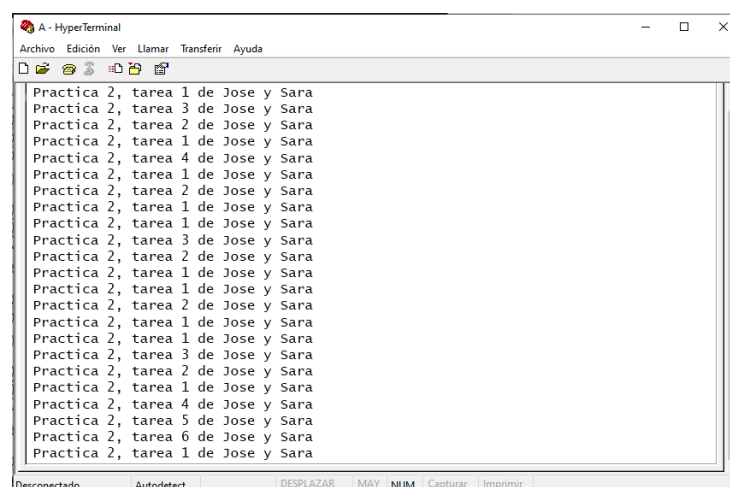
Practica 2, tarea 1 de Jose y Sara
Practica 2, tarea 2 de Jose y Sara
Practica 2, tarea 3 de Jose y Sara
Practica 2, tarea 4 de Jose y Sara
Practica 2, tarea 5 de Jose y Sara
Practica 2, tarea 6 de Jose y Sara
Practica 2, tarea 1 de Jose y Sara
Practica 2, tarea 2 de Jose y Sara
Practica 2, tarea 1 de Jose y Sara
Practica 2, tarea 1 de Jose y Sara
Practica 2, tarea 3 de Jose y Sara
PracticaPractica 2, tarea 1 a 2,tarea de e Josle de Josye Sara y Sara
Practica 2, tarea 1 de Jose y Sara
Practica Practica 1,tarea2 1Practica tar2ea ,1de tarea Jodesel yJose de Sarya
SaJora se y Sara

Practica 2, tPractica are2,a 1  tarea 1de Jode seJose y y Sara Sara
PracticPracta 2ica, 2,  tarea tarlea de Jose 1 deyJose Saray Sara

Desconectado  Autodetect.  DESPLAZAR  MAY  NUM  Capturar  Imprimir
```

Resultado 3:

Se ha modificado la velocidad de transmisión de datos, medida en baudios, a un valor mayor de 921600 baudios. Al aumentar la velocidad de transmisión, se ha notado una mejora en el rendimiento de la tarea, es decir, los datos se están enviando de forma más rápida y eficiente. Es probable que esta mejora en la velocidad de transmisión haya permitido que la tarea se ejecute más rápido y se pueda completar más rápidamente.



```
A - HyperTerminal
Archivo  Edición  Ver  Llamar  Transferir  Ayuda

Practica 2, tarea 1 de Jose y Sara
Practica 2, tarea 3 de Jose y Sara
Practica 2, tarea 2 de Jose y Sara
Practica 2, tarea 1 de Jose y Sara
Practica 2, tarea 4 de Jose y Sara
Practica 2, tarea 1 de Jose y Sara
Practica 2, tarea 2 de Jose y Sara
Practica 2, tarea 1 de Jose y Sara
Practica 2, tarea 1 de Jose y Sara
Practica 2, tarea 3 de Jose y Sara
Practica 2, tarea 2 de Jose y Sara
Practica 2, tarea 1 de Jose y Sara
Practica 2, tarea 1 de Jose y Sara
Practica 2, tarea 2 de Jose y Sara
Practica 2, tarea 1 de Jose y Sara
Practica 2, tarea 1 de Jose y Sara
Practica 2, tarea 3 de Jose y Sara
Practica 2, tarea 2 de Jose y Sara
Practica 2, tarea 1 de Jose y Sara
Practica 2, tarea 4 de Jose y Sara
Practica 2, tarea 5 de Jose y Sara
Practica 2, tarea 6 de Jose y Sara
Practica 2, tarea 1 de Jose y Sara

Desconectado  Autodetect.  DESPLAZAR  MAY  NUM  Capturar  Imprimir
```

Conclusiones:

En esta práctica se ha podido profundizar acerca del uso de Round Robin en el contexto de un RTOS. Round Robin se utiliza comúnmente como un algoritmo de planificación de tareas para garantizar que las tareas se ejecuten de manera predecible y en tiempo real. Como se ha podido observar, el algoritmo de Round Robin es muy útil en este contexto, ya que permite asignar una cantidad fija de tiempo de procesador para cada tarea, lo que asegura que cada tarea tenga un tiempo de CPU determinado y no afecte a la ejecución de otras tareas.