Sistemas en Tiempo Real – P7

Configuración del equipo

Los PCs actuales son multicore, un aspecto que no contemplamos en esta práctica, de modo que necesitamos que Linux explote un único core. Para ello configuraremos el número de cores útiles en tiempo de ejecución en la línea de comandos. Concretamente el segundo core (van del 0 a x-1, siendo x el número de cores) se deshabilita mediante el comando:

$ echo 0 | sudo tee /sys/devices/system/cpu/cpu1/online

Realmente lo que hace el comando es escribir un cero en el fichero online correspondiente al core 1 para deshabilitarlo. Escribir un uno vuelve a habilitarlo.

Ejercicio 1. Cálculo del tiempo de ejecución

Como vamos a realizar ejecuciones con tiempos de ejecución fijados, vamos a utilizar la función *spin_m(void *p)* (está ya implementada, no hay que hacerla) para ejecutar un bloque de código con la duración indicada en el parámetro *p*. El problema es que esta función realiza un bucle usando una variable para indicar la duración, pero el número de vueltas necesarias para esperar 1ms es diferente dependiendo de la máquina en que la ejecutemos. Por tanto, deberemos establecer el valor de dicha variable. Para ello, usando la función *execTime* (también implementada), iremos probando con diferentes valores usando como parámetro de *spin_*m 1000 milisegundos (1 segundo) hasta alcanzar un valor muy próximo a 1 segundo.

El esqueleto de la práctica tiene ya implementado este mecanismo, por lo que únicamente habrá que ejecutar la práctica usando el modo 0 (parámetro de ejecución 1), un valor que iremos variando hasta alcanzar el tiempo de ejecución esperado (parámetro de ejecución 2) y un tiempo de ejecución de 1000 (parámetro de ejecución 3). Para el parámetro de ejecución 4 se puede poner cualquier cosa, ya que no se usa en el modo 0 (no hace falta realizar toda la práctica para esta parte, con definir la estructura *marco_temp_t* debería ser suficiente).

Ejercicio 2. Planificación

En esta segunda parte, se deberá lanzar una hebra periódica con política SCHED_FIFO, la prioridad máxima disponible para dicha política (Ver el anexo para la descripción de las funciones sobre prioridades en POSIX) y los siguientes parámetros de planificación:

- T: Periodo de 3000 ms (3 segundos).
- C: Tiempo de ejecución más desfavorable de 500 ms.

Para ello, definiremos un tipo de datos que almacenará dichos valores. Dicha hebra realizará las siguientes operaciones:

1. Calcula el siguiente instante de ejecución de la tarea periódica en función de la hora actual y T.
2. Obtenemos la prioridad actual de la tarea.
3. Bucle
    3.1. Muestra el mensaje "Ejecutando tarea periódica con prioridad x".
    3.2. Ejecuta la tarea periódica: función *spin_m* usando como parámetro C.
    3.3. Espera hasta el siguiente instante de ejecución.
    3.4. Calcula el siguiente instante de ejecución de la tarea periódica en función del último instante de ejecución y T.

IMPORTANTE: Para poder realizar las modificaciones de prioridades, la ejecución de la práctica debe hacerse en modo superusuario.

La práctica se comprimirá en un único archivo que será entregado a través del campus virtual usando la tarea creada para tal efecto.

Anexo - Funciones POSIX para prioridad de hebras

- **pthread_attr_init(pthread_attr_t *attr)** → Inicializa los atributos de una hebra a sus valores por defecto.
- **pthread_attr_setinheritsched(pthread_attr_t *attr, int inherit)** → Establece, en los atributos de creación de una hebra, si ésta hereda o no la política/prioridad del padre. Por defecto tiene un valor PTHREAD_INHERIT_SCHED (hereda), para establecer la prioridad de un hilo, hay que cambiar este campo a PTHREAD_EXPLICIT_SCHED. Este valor se indica en el parámetro inherit.
- **pthread_attr_setschedpolicy pthread_attr_t *attr, int policy)** → Establece policy como política de planificación de la hebra que se cree usando los atributos attr. Los valores posibles son SCHED_FIFO, SCHED_RR (Round Robin), SCHED_SPORADIC (servidor esporádico) o SCHED_OTHER (depende de la implementación).
- **pthread_attr_setschedparam(pthread_attr_t *attr, const struct sched_param *param)** → Asigna los parámetros de planificación param a los atributos attr. De param sólo nos interesa lo siguiente:

  ```
  struct sched_param {
    ...
    int sched_priority;
    ...
  }
  ```

  El campo sched_priority será el valor de prioridad de los atributos de creación de la hebra. El resto de la estructura no nos interesa.
- **pthread_getschedparam(pthread_t thread, int *policy, struct sched_param *param)** → Devuelve en param los parámetros de la hebra thread y en policy su política. Usaremos esta función para obtener la prioridad de una hebra.
- **int sched_get_priority_max (int policy)** → Devuelve la prioridad máxima de la política policy.
- **int sched_get_priority_min (int policy)** → Devuelve la prioridad mínima de la política policy.

Por tanto, para la asignación de prioridades a una hebra, el programa principal deberá hacer lo siguiente:

1. Crear los atributos de creación de la hebra:
   1.1. Iniciar los atributos de creación de hebra.
   1.2. Indicar en los atributos de creación que la hebra no hereda del padre.
   1.3. Indicar en los atributos de creación de la hebra la política deseada.
   1.4. Asignar al campo sched_priority de una variable de tipo sched_param el valor de la prioridad que se desea dar.
   1.5. Indicar en los atributos de creación de la hebra la prioridad asignada en el paso anterior.
2. Crear la hebra usando los atributos del paso 1.