Sistemas en Tiempo Real – P8

Continuación de la práctica 7

Esta práctica añade/modifica funcionalidades a la práctica 7.

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. Deshabilitaremos todos los cores excepto el 0.

Marco temporal

El tipo *marco_temp_t* se encargará de almacenar el marco temporal de cada hebra. Es una ampliación de la estructura definida en la práctica 7 y deberá contener los siguientes campos (los dos primeros campos son los mismos que los de la práctica 7):

- C --> Tiempo de ejecución en milisegundos (tipo int)
- T --> Periodo de repetición en milisegundos (tipo int)
- P --> prioridad de la hebra (tipo int)
- id --> identificador de hebra (tipo int)
- TCritico --> instante crítico (tipo timespec). Se utilizará para que todas las tareas comiencen a la vez, evitando así los retardos surgidos de la creación de tareas.

Tarea periódica genérica

Se modificará la tarea periódica de la práctica 7 de la siguiente forma:

- La prioridad y la política (SCHED_FIFO para todas las hebras) se establece aquí en lugar de en el programa principal (no se usarán, por tanto, atributos de creación de hilos al lanzar hilos de este tipo).
- Antes de iniciar la acción periódica, se hará una espera hasta el valor indicado por el campo *TCritico* del parámetro de tipo *marco_temp_t* pasado a la hebra.
- El valor del primer instante de espera, que en la práctica 7 era *hora_actual+P*, en este caso será *TCritico+P*.
- La acción periódica que realiza la hebra es la siguiente:

- Muestra el mensaje "Hilo *id* con prioridad *P* inicia la acción periódica\n".
- Ejecuta la función *spin_m* usando como parámetro el campo *C* del parámetro de tipo *marco_temp_t*.
- Muestra el mensaje "Hilo *id* termina la acción periódica\n".
- Espera hasta el siguiente instante de ejecución.
- Calcula el siguiente instante de ejecución de la tarea periódica en función del último instante de ejecución y T.

Tarea Mostrar tiempos

Además de la tarea periódica genérica indicada anteriormente, deberá crearse otra tarea periódica, de periodo 10 ms que, tras esperar al instante crítico (lo recibe por parámetro), su acción periódica será:

- Mostrar el mensaje "********** *x* ms **********\n" (donde x variará de 10 en 10).
- Esperar al siguiente instante de ejecución
- Calcula el siguiente instante de ejecución de la tarea periódica en función del último instante de ejecución y 10 ms.

Esto se repetirá hasta que se hayan ejecutado 2 segundos (200 iteraciones de 10 ms).

El programa principal

Las funciones del programa principal serán:

- Establecer la prioridad de la tarea encargada de mostrar los tiempos (max FIFO) de forma análoga a como se hacía en la práctica 7.
- Crear e inicializar un vector *marcos* de tipo *marco_temp_t* con un tamaño de X, siendo X el número de tareas. Todos los valores del vector se rellenarán a partir de un fichero de entrada, que cargará el número de tareas y rellenará el vector con todos los valores del marco temporal (excepto el instante crítico). Esta parte está ya implementada en el esqueleto de la práctica. El formato del fichero es el siguiente:

  Número de tareas
  C_1 T_1 P_1
  …
  C_n T_n P_n
- Calcular el instante crítico (hora actual más dos segundos) y asignarlo a todos los elementos del vector *marcos*.
- Lanzar todas las hebras.
- Esperar únicamente por la hebra que muestra los tiempos.

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

Ver el anexo para la descripción de las funciones sobre prioridades en POSIX.

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_INHERIT_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. sched_param está definida de la siguiente forma:

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

  El campo sched_priority será el valor de prioridad de los atributos de creación de la hebra.
- **pthread_setschedparam(pthread_t thread, int policy, struct sched_param *param)** --> Establece los parámetros (param) y la política (policy) de la hebra thread.
- **pthread_self()** --> Devuelve el identificador de la hebra actual.

Las funciones *pthread_attr_xxxxx* se invocan antes de la función *pthread_create*, ya que ésta última utilizará los atributos modificados por las primeras. Las otras funciones se usarán dentro de la hebra que las necesite.

Para establecer la política y la prioridad dentro de una hebra en particular seguiremos la siguiente secuencia:

1. Crear una variable de tipo *sched_param*.
2. Asignar al campo *priority* de la variable definida en 1 la prioridad deseada.
3. Establecemos tanto la prioridad como la política de la hebra usando la función *pthread_setschedparam* (para el identificador de la hebra a modificar, usar *pthread_self()*)