Sistemas en Tiempo Real – Práctica Posix 5

Esta práctica añade/modifica funcionalidades a la Práctica Posix 4. Concretamente, mientras que en la Práctica Posix 4 se lanzaban tareas periódicas con recursos no compartidos y con prioridades estáticas, en esta práctica vamos a indicar, para cada hilo, una lista de acciones, que incluirá la utilización de recursos compartidos. Además, utilizaremos el protocolo techo de prioridad inmediato para la gestión de prioridades dinámicas de los recursos compartidos.

Configuración del equipo

Los PCs actuales son multicore, un aspecto que no contemplamos en esta práctica, de modo que necesitamos utilizar un único core. Para ello configuraremos el número de cores útiles en tiempo de ejecución en la línea de comandos (lo que haremos será desactivar los cores que no utilicemos). Para desactivar un core escribiremos un 0 en el fichero *online* de dicho core mediante el siguiente comando:

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

El comando anterior desactiva solamente el core 1 (*cpu1* en la ruta), por lo que para desactivar otros cores hay que cambiar *cpu1* por *cpuX*, siendo *X* el core que queremos desactivar. Escribir un uno vuelve a activarlo y reiniciar el equipo activa todos los cores. Para usar un único core debemos **desactivar todos los cores excepto el 0.**

Modificación de la clase *hilo_t*

Para esta práctica vamos a modificar la clase *hilo_t* implementada en la Práctica Posix 0 y ampliada en la Práctica Posix 4, ya que debe manejar los datos de las diferentes acciones que va a ejecutar el hilo, de los recursos que se usarán en esas acciones y de los mutex asociados a los recursos. Para ello, añadiremos los siguientes elementos:

- Nuevos atributos privados:
  - Lista de acciones del hilo (vector de enteros, usar la clase *vector*). Este vector contendrá los tiempos de ejecución en milisegundos de las acciones que va a ejecutar el hilo.
  - Lista de recursos del hilo (vector de enteros, usar la clase *vector*). Este vector contendrá los recursos que se utilizarán para ejecutar las acciones correspondientes, pudiendo tomar valores en el rango [-1, K), siendo K el número de recursos compartidos. El valor -1 indica que se va a ejecutar un recurso no compartido y un valor en el rango [0, K) indica que se va a ejecutar un recurso compartido concreto. Todos los recursos compartidos tendrán asociado un mutex específico (ver la lista de mutex a continuación).

    Los dos vectores anteriores serán del mismo tamaño ya que cada acción lleva emparejada un recurso.
  - Lista de mutex (vector de tipo puntero a *mutex_t*, usar la clase *vector*). Este vector va a contener los mutex asociados a los diferentes recursos compartidos de la aplicación. Como deben ser compartidos por todos los hilos, deben ser punteros. La posición i-ésima del vector hace referencia al mutex asociado con el recurso i.

    Mediante estos tres atributos tendremos una definición clara de lo que va a ejecutar el hilo. Por ejemplo, si queremos saber la configuración de la cuarta acción que va a ejecutar el hilo, el valor de la posición 3 de la lista de acciones (recordemos que los vectores empiezan en la posición 0, por lo que la cuarta acción estará en la posición 3 del vector) nos dará su tiempo de ejecución, la posición 3 de la lista de recursos nos dirá el recurso que se va a utilizar para realizar dicha acción y tendremos

el mutex asociado al recurso utilizado accediendo a la posición de la lista de mutex indicada por dicho recurso.

- Nuevos métodos públicos:
  - *Int EstablecerAtributos* → Se debe añadir un nuevo método, no modificar el de la Práctica Posix 4. Tendrá los siguientes parámetros (deben estar en el orden indicado):
    - Mismos parámetros que el método *EstablecerAtributos* definidos en la Práctica Posix 4 (mantener el orden definido en dicha práctica).
    - Lista de acciones del hilo (vector de enteros, usar la clase *vector*).
    - Lista de recursos del hilo (vector de enteros, usar la clase *vector*).
    - Lista de mutex (vector de tipo puntero a *mutex_t*, usar la clase *vector*).

    Este método se va a encargar de asignar los valores de los diferentes parámetros a los atributos de clase a los que corresponden (la clase *vector* tiene sobrecargado el operador de asignación, con lo que se pueden hacer asignaciones de vectores sin problemas), así como de establecer los valores de herencia, política y prioridad en los atributos de creación del hilo de la misma forma que se hacía en la Práctica Posix 4 (lo más sencillo es llamar al método *EstablecerAtributos* definido previamente con los parámetros correspondientes).
  - *int ObtenerNumAcciones* sin parámetros → Este método devolverá el número de acciones contenidas en la lista de acciones del hilo.
  - *in*t ObtenerAccion con un parámetro de tipo entero → Este método devolverá el valor de la lista de acciones indicado por el parámetro.
  - *int ObtenerRecurso* con un parámetro de tipo entero → Este método devolverá el valor de la lista de recursos indicado por el parámetro.
  - *mutex_t *ObtenerMutex* con un parámetro de tipo entero → Este método devolverá el valor de la lista de mutex indicado por el parámetro.

Modificación de la clase *mutex_t*

Para esta práctica vamos a tener que modificar la clase *mutex_t* implementada en la Práctica Posix 0 ya que vamos a tener que indicar el protocolo y el techo de prioridad (cuando sea necesario) de los mutex. Además, como esta clase va a ser utilizada por la clase *hilo_t*, habrá que moverla al comienzo del fichero *ClasesPosix.h* (esta acción no es necesaria en *ClasesPosix.cpp*). Lo elementos que debemos añadir son los siguientes:

- Nuevos métodos públicos:
  - *Int AsignarProtocolo* con un parámetro de tipo entero →Este método se encargará de establecer, en los atributos de creación del mutex, el protocolo del mutex indicado en el parámetro (ver el Anexo 1).
  - *Int AsignarTecho* con un parámetro de tipo entero→Este método se encargará de establecer, en los atributos de creación del mutex, el techo de prioridad del mutex indicado en el parámetro (ver el Anexo 1).
  - *Int AsignarProtocoloYTecho* con dos parámetros de tipo entero → Este método se encargará de establecer, en los atributos de creación del mutex, tanto el protocolo del mutex indicado en el primer parámetro como el techo de prioridad del mutex indicado en el segundo parámetro (ver el Anexo 1). Lo más sencillo es llamar respectivamente a los dos métodos que acabamos de definir.

Función *EjecutarRecurso*

Vamos a implementar la función *EjecutarRecurso* que se va a encargar de ejecutar un recurso durante un tiempo de ejecución concreto. Para ello, recibe por parámetros el mutex asociado al recurso, el identificador

del recurso que se va a ejecutar (-1 para recursos no compartidos y de 0 a K-1 para recursos compartidos), el tiempo de ejecución del recurso y el identificador del hilo que realiza la ejecución. El pseudocódigo de la función es el siguiente:

```
Inicio EjecutarRecurso
   Si el recurso es un recurso compartido
      Bloquear el mutex asociado al recurso
   Mensaje de comienzo de ejecución de un recurso
   Ejecutar la función Ejecutar con el tiempo de ejecución del
    recurso
   Mensaje de finalización de ejecución de un recurso
   Si el recurso es un recurso compartido
      Desbloquear el mutex asociado al recurso
Fin EjecutarRecurso
```

Tarea Periódica Genérica

Se deberá implementar una tarea periódica genérica que tomará como parámetro un puntero a *hilo_t* que permitirá acceder a todos los datos del hilo necesarios para la ejecución de la tarea (debe tener el formato de las funciones usadas para crear hilos, ver el Anexo 1). A continuación, se muestra el pseudocódigo de esta tarea:

```
Inicio Tarea Periódica Genérica
   Transformar el parámetro de entrada de void* a hilo_t*
   Obtener el periodo y el instante de comienzo del hilo
   Calcular el instante siguiente de repetición sumando los dos
    valores anteriores
   Dormir el hilo hasta el instante de comienzo
   Bucle infinito
      Mensaje de comienzo de la acción periódica
      Para cada acción del hilo
         Llamar a EjecutarRecurso con los parámetros mutex asociado
          al recurso i-ésimo del hilo, recurso i-ésimo del hilo,
          acción i-ésima del hilo e identificador del hilo
      Fin para
      Mensaje de finalización de la acción periódica
      Dormir el hilo hasta el instante siguiente
      Calcular el nuevo instante siguiente añadiéndole el periodo
   Fin bucle infinito
Fin Tarea Periódica Genérica
```

Tarea Mostrar Tiempos

Esta es la misma tarea que en la Práctica Posix 4, por lo que tiene la misma definición y puede copiarse de dicha práctica o implementarse de nuevo siguiendo el esqueleto de la misma.

El programa principal

El programa principal se encargará de leer los datos de las tareas de un fichero de entrada (está ya implementado en el esqueleto de la práctica), configurar los mutex asociados a los diferentes recursos (todos ellos usarán el protocolo techo de prioridad inmediato), así como de configurar y lanzar los diferentes hilos (un hilo de tipo Tarea Periódica Genérica por cada tarea del fichero de tareas más un hilo de tipo Tarea Mostrar

Tiempos). Además, esperará a que termine la Tarea Mostrar Tiempos (el resto ejecutan bucles infinitos, por lo que cuando esta tarea termine, terminarán todas al ser la única por la que esperamos). A diferencia de las prácticas anteriores, el programa principal de esta práctica intercala zonas de código ya implementadas con otras zonas que deben ser implementadas y que están marcadas entre "Zona de implementación de la Práctica Posix 5" y "Fin de zona de implementación de la Práctica Posix 5", por lo que se debe tener especial cuidado con las zonas de código que se modifican dentro del programa principal.

La configuración de los diferentes hilos será la siguiente:

- Tiempo de ejecución, periodo, prioridad, lista de acciones y lista de recursos de las tareas periódicas genéricas → Vienen dados en el fichero de tareas (ver más abajo el formato del mismo).
- Tiempo de ejecución, periodo, y prioridad de la tarea mostrar tiempos → 0 (no ejecuta ninguna tarea periódica, con lo que su tiempo de ejecución es irrelevante), 10 ms y prioridad máxima para la política FIFO (establecer a este hilo la prioridad máxima permitirá que saque a los demás hilos del procesador para mostrar las franjas temporales). Esta tarea no tiene lista de acciones, lista de recursos ni lista de mutex, por lo que se inicializará con el método definido en la Práctica Posix 4.
- Identificador de hilo → N+1 para la tarea mostrar tiempos y de 1 a N (siendo N el número de tareas) para las tareas periódicas genéricas. Como el identificador del hilo se asigna automáticamente, debemos llamar primero a las tareas periódicas y, por último, a la tarea mostrar tiempos.
- Instante de comienzo → Todas las tareas partirán del mismo instante inicial, que se calculará sumando dos segundos a la hora actual. Sin embargo, las tareas periódicas genéricas deberán sumar a este tiempo el instante de activación indicado en el fichero de tareas (ver más abajo el formato del mismo).
- Política → FIFO para todos los hilos.
- Herencia → Es igual para todos los hilos y debe indicar que no se va a heredar del padre.

El formato del fichero de tareas es el siguiente:

$$
\begin{aligned}
&N\\
&K\ techo\_prio_1\ \ldots\ techo\_prio_K\\
&C_1\ T_1\ Ta_1\ P_1\ NumAcc_1\ R_{1,1}\_TE_{1,1}\ \ldots\ R_{1,1}\_TE_{1,NumAcc1}\\
&\ldots\\
&C_N\ T_N\ Ta_N\ P_N\ NumAcc_N\ R_{N,1}\_TE_{N,1}\ \ldots\ R_{N,1}\_TE_{N,NumAccN}
\end{aligned}
$$

siendo N el número de tareas en el fichero, K el número de recursos compartidos por todas las tareas, $techo\_prio_i$ el techo de prioridad del recurso i, C_i el tiempo de ejecución de la tarea i, T_i el periodo de la tarea i, Ta_i el tiempo de acceso de la tarea i, P_i la prioridad de la tarea i, $NumAcc_i$ el número de acciones que tiene la tarea i, $R_{i,j}$ el recurso usado en la acción j de la tarea i y $TE_{i,j}$ el tiempo de ejecución de la acción j de la tarea i. Para recursos no compartidos se usarán los caracteres 'n' o 'N' y para recursos compartidos valores entre 0 y K-1.

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

Anexo 1 – Posix

- Definición de la función que se usará para lanzar un hilo: *void *NombreFuncion(void *parámetro)*.
- Paso de parámetros a un hilo:
  - Parámetro de tipo puntero:

```
void *funcion(void *aux) { //Por ejemplo, el parámetro es un vector de enteros
        int *n=(int *)aux;
 //Resto del código usando n de forma normal
}
```

  - Parámetro de tipo no puntero:

```
void *funcion(void *aux) { //El parámetro es un dato de tipo TDato
        Tdato d=*((TDato *)aux);
 //Resto del código usando d de forma normal
}
```

- **pthread_mutexattr_setprotocol (pthread_mutex_attr_t *attr, int protocol)** → Establece el protocolo de prioridades de *attr*. Dicho protocolo puede ser *PTHREAD_PRIO_INHERIT* (Herencia de prioridad), *PTHREAD_PRIO_PROTECT* (Techo de prioridad inmediato), *PTHREAD_PRIO_NONE* (Prioridades estáticas).
- **pthread_mutexattr_setprioceiling (pthread_mutexattr_t *attr, int prioceiling)** → Establece *prioceiling* como techo de prioridad de *attr*.