Prácticas Concurrencia y Distribución (17/18)

Arno Formella, Anália García Lourenço, Lorena Otero Cerdeira, David Olivieri

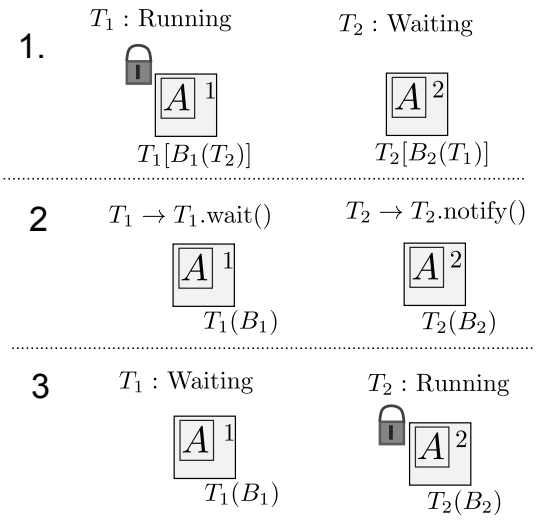
semana 19 – 25 marzo

6. Semana 6 (19/03–25/03): Sincronización

La semana pasada, se probó como usar el mecanismo de `wait()/notifyAll()` para controlar la planificación de hilos dada alguna condición. Ahora, en este laboratorio, queremos llevar esta idea más allá y ordenar los hilos. Esto se hará de dos maneras: 1) utilizando `notifyAll()`, donde cada hilo se despierta y necesita verificar la condición de acceso a la sección crítica, o 2) directamente usando una referencia al siguiente hilo y despertando solo este hilo mediante `notify()` para que sea el siguiente en acceder.
En el primer problema, se aprenderá cómo hacer referencia a un hilo directamente desde otro. En el segundo problema, se volverá al código `EnterAndWait()` de la semana pasada para controlar el orden de los hilos que entran en la sección crítica.

1. **(P4: para entregar en grupo de práctica)**:

   Objetivo: este ejercicio pretende reiterar el concepto de compartir objetos entre hilos y utilizar la referencia del objeto compartido para controlar el acceso de los hilos a las secciones críticas. Consulta la Figura 1 a continuación:



   a) **Configuración del problema:** Comienza usando el código de la semana pasada para la sincronización de `EnterAndWait()`. Ahora crea una clase principal (por ejemplo, `MiProblema`) donde, dentro de su método `main`, se creen dos threads t_1 y t_2. El hilo debe tener un método `set()` para que se pueda establecer la referencia de uno al otro.

   b) **Sincronizar:** Tal y como se muestra en la secuencia de estados de la Figura 1, escribe el código apropiado, de tal forma que se *alternará* el estado de ejecución y espera entre los hilos t_1 y t_2. Des-

pués de que un hilo ejecuta la sección crítica (`EnterAndWait()`), debe notificar explícitamente al siguiente hilo.

c) **Ejecución e interrupción del programa:** Tu código debería ejecutarse en un bucle infinito. Añade el código necesario para que se interrumpan los hilos y así detener el programa.

d) **Lanzando un tercer hilo:** Modifica el código para crear e iniciar un tercer hilo t_3. La operación de alternar entre hilos solo debería tener lugar entre los dos primeros como antes. ¿Puedes observar que t_3 nunca ingresa en la sección crítica?

2. **(P3: para entregar dentro de una semana)**:

Objetivo. En el código `EnterAndWait()` de la semana pasada, cualquier hilo podía entrar en la sección crítica (con un poco más de restricción asociada a los hilos rojos/negros). Ahora, queremos que los hilos entren dado un orden específico. Haremos esto de dos formas diferentes: 1) utilizando `notifyAll()`, que es un método menos eficiente ya que despierta a todos que están esperando, y 2) notificando solo al hilo al que toca su turno. Consulta la figura a continuación para la notificación de llamada explícita en el orden de hilos:

Sigue estos pasos:

a) **Configuración del problema:** Usa tu código del problema `EnterAndWait()` de la semana pasada.

b) **Orden secuencial (método de fuerza bruta):** utilizando directamente el código de la semana pasada, añade el código necesario en los hilos para ejecutar la sección crítica en orden con llamadas usando `notifyAll()`.

c) **Orden secuencial (método explícito/eficiente):** utilizando la idea de la Figura 2. Escribe el código apropiado para que el orden de los hilos que ejecutan la sección crítica se realice explícitamente mediante llamadas al siguiente hilo en la lista.

d) **Análisis:** Compara el tiempo de ejecución del método de fuerza bruta (parte b) con el método explícito (parte c). Haz un plot y describe tus observaciones en palabras.