Prácticas Concurrencia y Distribución (22/23)

Arno Formella, Alba Nogueira Rodríguez, David Ruano Ordás

semana 17 abril – 21 abril

Práctica 8: Sincronización total en orden predeterminado

Objetivos: Dormir y despertar los hilos en un orden cíclico predeterminado. Resolver los problemas de comienzo y de terminación.

La semana pasada, usamos el mecanismo de `wait()` y `notify()`/`notifyAll()` para controlar el acceso a un recurso de hilos. Ahora, en esta práctica, queremos llevar esta idea más allá y garantizar que los hilos actuen en un orden predeterminado. Esto haremos de la siguiente manera. Usamos unas referencias a un objeto de sincronización que comparten dos hilos, uno dormirá (con `wait()` sobre él, el otro despertará el hilo correspondiente cuando toca (mediante `notify()`).

1. Re-usamos en gran parte el código de la semana pasada.
2. Recupera el código de las clases `ClassA` y `ClassB` de la práctica de la semana pasada (5ª práctica), en que se sincronizaba el acceso de varios hilos a una sección crítica (con el método `EnterAndWait()`).
3. Usa ahora tantos objetos de la clase `ClassA` como hay hilos participantes en el juego, de tal manera que cada hilo tenga su propio objeto de tal clase.
4. Añade a la clase `ClassA` una referencia a otro objeto del mismo tipo con el objetivo que finalmente se forme un anillo de referencias entre ellos. Eso será el orden en el cual los hilos partipan en las jugadas.
5. Cambia en contador de jugadas en la clase `ClassA` para que sea un miembro estático, así todos las instancias trabajan con el único contador.
6. Con estas ayudas completa el programa para que la lógica se convierte a un juego (estilo PasaPelota), o sea:
   - Una vez inicializados los hilos (jugadores) cada uno con su objeto de sincronización que a su vez apunta al siguiente, cuando arrancan el juego (entrada en el método `run()` se paran sobre su objeto de sincronización (con un `wait()`.

- El hilo principal (podemos llamarle árbitro), notifique a un jugador para que empieze a jugar (con una llamada `notify()` adecuada).
- Los jugadores imprimen su jugada a la pantalla, cuentan la jugada como realizada, y notifican el siguiente jugador, finalmente vuelven a dormir hasta que les toque otra vez.
- Cuando el contador haya alcanzada cero, ya no se imprime jugada, sino se termina el juego.

  El bucle principal de los hilos se simplifica otra vez a algo como:

```
while(syncObj.EnterAndPlay(id)) {
  ++counter; // cuenta jugadas de este jugador
}
```

7. Manten las estadísticas que cuenta las jugadas de cada jugador, para poder observar cuantas juagadas cada jugador finalmente haya realizadao.
8. Verifica que los hilos juegan de forma equilibrada y que no haya acceso en vano.
9. Seguramente observas que hay que garantizar que el hilo que se quiere despartar (con `notify()`) de verdad está durmiendo (en su `wait()` correspondiente), sino el juego no avanza y el programa se queda bloqueado. !Añade la lógica para garantizar eso!
10. ¿Funciona tu programa con un único jugador (hilo)?