Prácticas Concurrencia y Distribución (20/21)

Arno Formella, Anália García Lourenço, Noelia García Hervella

semana 12 abril – 16 abril

Práctica 5b: Sincronización con esperar y despertar

Objetivos: Objetos compartidos entre procesos con acceso sincronizado y espera entre hilos. Este ejercicio pretende indroducir el concepto de ponerse en espera hasta que se reciba un aviso de otro hilo para despertarse. Se sigue trabajando con el código de la semana pasada.

1. Re-edita el código de la semana pasada con las dos clases `ClassA` (objeto compartido) y `ClassB` (hilos trabajadores) y el método `EnterAndDecrement` con `synchronized` e incluye en el sitio adecuado una notificación (usando `notify()` o `notifyAll()` y a continuación una espera (usando `wait()`.
   - El propósito de está modificación es aliviar el hecho que observamos la semana pasda que en tal versión sin espera unos pocos hilos realizan todos los decrementos mientras otros nuncan llegan a hacer nada. (¡Cuidado que tu programa sigue funcionando quizá necesitas una condición antes de ir a `wait()` sino vas a durmir sin que nadie te despierta!)
   - ¿Observas una diferencia respecto a la distribución del *trabajo*? Analiza los valores mínimo, máximo, y varianza de los accesos.
   - ¿Observas una diferencia significativa entre el uso de `notify()` y `notifyAll()`?
2. Ahora aumentamos el programa para que se convierte en un simple juego en equipos entre hilos:
   - Dividimos los hilos en dos equipos, por ejemplo, equipo rojo y equipo negro (una posibilidad es que los con índices pares van al equipo rojo y los con índices impares al equipo negro).
   - Al objeto compartido (puedes imaginarte que es una bandera) asignamos un color que nos indica cuál de los dos equipos puede actuar (jugar).
   - Actuar o jugar significa que si el color de la bandera es diferente al color del equipo del hilo, el hilo puede cambiar el color de la bandera (es decir, la siguiente jugada toca al otro equipo).

Para realizar esta modificación considera las siguientes ayudas

- Añade a la clase `ClassA` un atributo de color, por ejemplo, mediante un `String canPlay` (que va a indicar el color actual de la bandera).
- Añade a la clase `ClassB` un atributo de color, por ejemplo, mediante un `String team` que indica a que equipo pertenece el hilo.
- La jugada puede ser, por ejemplo, un método como este:

```
private void SwitchTeam() {
  System.out.println("Team "+canPlay+" is playing");
  if(canPlay.equals("red")) canPlay="black";
  else canPlay="red";
}
```

- El método de `EnterAndDecrement()` (que yo renombro ahora en `EnterAndPlay()`) recibe ahora el color del jugador que quiere jugar, es decir, `EnterAndPlay(final String team)`.
- Y con eso implementamos una actuación de un hilo con algo como:

```
public synchronized boolean EnterAndPlay(
  final String team
) {
  try {
    // System.out.println("EnterAndPlay entered for "+caller_id);
    if(team.equals(canPlay)) {
      SwitchTeam();
      // AQUI SIGUE EL CÓDIGO de los partados ANTERIORes
    }
  }
  catch(Exception E) {
    System.out.println("unexpected exception...");
  }
  finally {
    // System.out.println("EnterAndPlay ended for "+caller_id);
  }
  return true;
}
```

3. Una vez conseguido estos cambios:

- Asegurate que tu programa siga funcionando con cualquier número de hilos (observa con un número impar), si es necesario busca un remedio.
- Observa cuantas veces los hilos entran en el método `EnterAndPlay()` sin que finalmente puedan jugar.
- Aumenta tu programa con un contador adicional que cuente solamente los accesos con éxito y cambia la lógica del bucle principal del `run()` en algo como

```
while(!sharedObj.HaveFinished()) {
  if(sharedObj.EnterAndPlay(id,team)) ++success;
  ++counter;
}
```

donde `HaveFinished()` devuelve el estado del contador de jugadas (`true` si está en cero, y `false` si todavía no está en cero). Es decir, `EnterAndPlay()` devuelve ahora `true` si se ha jugado con éxito, y `false` en caso que no.

- Visualiza las estadísticas tanto de las acciones en total como de las acciones con éxito.

4. Intenta buscar una forma de reducir el número de jugadas en vano con colocar el `notify()/wait()` en otro lugar (manteniendo la lógica del juego).
5. Reflexiona si es posible realizar las tareas de esta semana con los cerrojos (`lock`).
6. ¡La semana que viene intentamos bajar los intento en vano a cero!