Sistemas Operativos II

Práctica 2 – Programación Multihilo

Buscador SSOOIIGLE

Javier Alonso Albusac Jiménez
Escuela Superior de Informática :: Universidad de Castilla-La Mancha

En la segunda práctica de la asignatura SSOO II se solicita el diseño y codificación del buscador SSOOIIGLE, capaz de buscar información en un fichero de forma paralela con múltiples hilos. En líneas generales, el usuario debe especificar al buscador el fichero donde buscar, la palabra que desea buscar y el número de hilos que se deben emplear en la búsqueda.

```
SSOOIIGLE <nombre_fichero> <palabra> <numero_hilos>
```

De esta forma, el fichero se «dividirá[1]» en tantas partes como hilos se indiquen. Por ejemplo, imagine el caso en el que un fichero consta de 100 líneas. Si el usuario invocara al buscador de la siguiente forma:

```
SSOOIIGLE historia.txt castillo 2
```

El proceso principal creará dos hilos, el primero de ellos buscará la palabra castillo entre las líneas 1 y 50 del fichero *historia.txt*, mientras que el segundo hilo realizará la misma búsqueda de forma paralela entre las líneas 51 y 100.

Por otro lado, el buscador debe mostrar por pantalla la aparición de la palabra en el archivo con la siguiente información:

- **Fragmento** en el que se ha encontrado la palabra. A su vez un fragmento se define mediante un *número de hilo*, *línea inicial* y *línea final*.
- **Línea del fichero** en la que ha sido encontrada.
- Referencia a las dos palabras entre las que se encuentra la buscada: palabra anterior y palabra posterior.

Un requisito importante es que el resultado de búsqueda se debe mostrar en orden. Es decir, no se pueden mostrar los resultados de un fragmento posterior, aunque el hilo que haya realizado la búsqueda en ese fragmento haya finalizado antes. Un ejemplo de salida podría ser el siguiente:

```
[Hilo 1 inicio:0 – final: 25] :: línea 12 :: … el castillo antiguo …
[Hilo 1 inicio:0 – final: 25] :: línea 20 :: … un castillo abandonado …
[Hilo 3 inicio:51 – final: 75] :: línea 63 :: … el castillo y …
[Hilo 3 inicio:51 – final: 75] :: línea 72 :: … sin castillo y …
[Hilo 4 inicio:76 – final: 100] :: línea 92 :: … el castillo derrumbado …
```

El estudiante debe diseñar una estructura de datos donde almacenar las ocurrencias de una palabra de acuerdo al formato anterior. Esta estructura debe ser compartida por todos los hilos y se debe garantizar exclusión mutua en el acceso. Es decir, mientras un hilo acceda a la estructura para escribir una nueva ocurrencia, el resto debe esperar.

1 El fichero no se divide realmente, sino que cada hilo manipula una zona concreta del fichero.

Evaluación de la práctica – Puntos de control

A continuación se listan una serie de puntos de control que se tendrán en cuenta en el proceso de evaluación para determinar qué objetivos se han alcanzado y en qué grado.

Nº	Puntos de control	Satisfecho/No Satisfecho
	Específicos de la práctica	
1	Procesamiento adecuado de los valores incluidos por línea de órdenes	
2	División en partes iguales (excepto la última) del fichero sobre el que se realiza la búsqueda en tamaños.	
3	Creación adecuada de los hilos y asignación de tareas a hilos.	
4	Se respeta el formato de salida que se indica en el enunciado.	
5	Los resultados se muestran en orden.	
6	Se encuentran todas las palabras en el fichero indicado. Incluido cuando existen varias ocurrencias en la misma línea.	
7	Valoración de la estructura de datos diseñada para almacenar los resultados de búsqueda.	
8	Se garantiza exclusión mutua en el acceso a la estructura de datos anterior.	
9	Forma en la que se espera a la finalización de los hilos ¿se trata de una espera activa? o ¿desbloqueo con notificaciones?.	notificaciones
	OTROS	
10	Número de herramientas de sincronización empleadas (qué herramientas de sincronización de las vistas en teoría han sido empleadas para resolver la práctica).	3
11	Estructuración adecuada del código.	
12	Código limpio, claro, con alto grado de interpretabilidad.	
13	¿Solución genérica?, ¿funciona para cualquier fichero .txt?.	
14	¿Solución robusta y tolerante a fallos?	
15	Uso de patrones de sincronización.	
16	Rendimiento global del sistema en términos de eficiencia temporal.	
17	Se hace una gestión adecuada de los errores.	
18	Se liberan los recursos correctamente al finalizar	
19	Creación de Makefile para facilitar la compilación de código.	
20	Incluye instrucciones README.txt para la compilación y ejecución del código.	
21	Organización adecuada en directorios de los archivos del proyecto: exec, obj, src e include.	