Concurrencia y Paralelismo

Grado en Informática 2023

Práctica 2 – Md5 Checker

El algoritmo md5 es un algoritmo de hash, que permite asociar a entradas de tamaño arbitrario valores dentro de un rango limitado. Un uso frecuente de estos algoritmos es la comprobación de la corrección de secuencias de datos como ficheros, bloques transmitidos por red, ...

Vamos a implementar un comprobador mediante md5, que dado un directorio permita realizar dos operaciones:

1. `md5 -s dir file`, que calcula la suma de todos los ficheros contenidos en `dir` y la guarda en `file`. Dentro de `file` habrá una linea por cada fichero de `dir`, con el nombre del fichero y su hash md5.
2. `md5 -c dir file`, que dado un fichero generado mediante la orden anterior, comprueba la corrección de los ficheros almacenados en `dir`.

El código proporcionado implementa la versión secuencial de este problema. Para la operación 1, el programa:

1. Lista los ficheros contenidos en el `dir`, y guarda sus nombres en una cola.
2. Se itera sobre la cola, y para cada elemento se calcula su hash md5 y se guarda en una cola de salida.
3. Para cada elemento de la cola de salida, se escribe una línea en el fichero con la lista de hashes.

Para la operación 2:

1. Genera una lista con los ficheros y hashes guardados en `file` y se guarda en una cola.
2. Para cada fichero de la cola, se calcula el hash y se compara con el almacenado en el fichero. En caso de haber diferencias, se imprime un mensaje.

El programa acepta las siguientes opciones:

- `-t n`, para especificar el número de threads que vamos a crear.
- `-q n`, para controlar el tamaño de las colas de entrada y salida.

Partiendo de este código se pide modificar el comportamiento del programa para la operación 1 de la siguiente forma:

Ejercicio 1 (**Proteger la cola frente a accesos concurrentes**) Modifique la implementación de la cola para que permite inserciones y borrados simultáneos desde múltiples threads.

Ejercicio 2 (**Separar la generación del listado de fichero del directorio en un thread independiente**) Separe el proceso de leer la lista de ficheros a su propio thread. Una vez hechos los dos primeros ejercicios debería ser posible modificar el tamaño de la cola de entrada a 1, y el programa debería seguir funcionando.

Ejercicio 3 (**Separar el cálculo de hashes a varios threads**) Separe en cálculo de hashes md5 de tal forma que se haga en los threads especificados en las opciones.

Ejercicio 4 (**Separar la escritura**) Separar la escritura de los hashes calculados a su propio thread. Una vez implementado este punto el programa debería funcionar aunque la cola de salida tenga longitud 1.

Para la operación 2:

Ejercicio 5 (**Separar la lectura del fichero con la lista de hashes a un thread independiente**)

Ejercicio 6 (**Separar el cálculo de hashes a varios threads**) Separe el cálculo de los hashes de cada fichero a comprobar en varios threads. Una vez hecho este paso, el programa debería funcionar con una cola de entrada de longitud 1.

OpenSSL

Para el cálculo de los hashes se utiliza la librería openssl. Asegúrese de tener disponibles los ficheros de cabecera de la librería. En distribuciones Linux basadas en debian (como ubuntu o mint) forman parte del paquete `libssl-dev`, que puede instalarse con:

```
sudo apt install libssl-dev
```

Entrega

La fecha límite de entrega es el 8 de marzo (hasta el final del día). El código inicial está disponible a través de github classroom en https://classroom.github.com/a/taAjP-bV.

La corrección se hará sobre el contenido del repositorio al final del día 8 de marzo.