INSTITUTO POLITECNICO NACIONAL

ESCUELA SUPERIOR DE COMPUTO

Sistemas Distribuidos

Práctica 1. Procesos e hilos

Reyes López Maximiliano

7CM1

Antecedentes

Los procesos e hilos son conceptos fundamentales en la programación concurrente. Un proceso es una instancia en ejecución de un programa, mientras que un hilo es una unidad de ejecución dentro de un proceso. Java proporciona soporte para el manejo de hilos a través de la clase Thread y la interfaz Runnable, permitiendo la ejecución concurrente de tareas.

Los hilos se utilizan para mejorar la eficiencia de las aplicaciones al permitir la ejecución paralela de tareas. Java ofrece herramientas como Thread, ExecutorService y ProcessBuilder para manejar procesos e hilos de manera efectiva.

Planteamiento del problema

Se requiere implementar un programa en Java que realice procesamiento concurrente utilizando procesos e hilos. Específicamente, el programa debe dividir un archivo de texto en dos partes y contar la cantidad de palabras en cada una utilizando hilos que ejecuten procesos del sistema.

Propuesta de solución

Para resolver el problema, se propone desarrollar una aplicación en Java con las siguientes características:

- División de un archivo de texto en dos partes.
- Creación de dos hilos para contar palabras en cada parte.
- Uso de **ProcessBuilder** para ejecutar comandos del sistema que realicen el conteo de palabras.
- Sincronización de hilos utilizando **join()** para asegurar que ambos finalicen antes de mostrar los resultados.

Materiales y métodos

Lenguaje de programación: Java
Compilador: JDK 17
Librerías utilizadas:

- **java.io** para manejo de archivos
- **java.lang.Thread** para creación y gestión de hilos
- **java.lang.ProcessBuilder** para ejecutar procesos del sistema

Desarrollo de la solución

El programa lee el archivo de entrada (input.txt), dividiéndolo en dos archivos (part1.txt y part2.txt). La distribución se realiza alternando líneas entre ambos archivos.

```java
private static void splitFile(String inputFilePath, String part1FilePath, String
part2FilePath) throws IOException {
    File inputFile = new File(inputFilePath);
    if (!inputFile.exists() || inputFile.length() == 0) {
        throw new IOException("El archivo de entrada no existe o está vacío.");
    }
    try (BufferedReader reader = new BufferedReader(new FileReader(inputFilePath));
         FileWriter writer1 = new FileWriter(part1FilePath);
         FileWriter writer2 = new FileWriter(part2FilePath)) {
        String line;
        int lineCount = 0;
        while ((line = reader.readLine()) != null) {
            if (!line.trim().isEmpty()) {
                if (lineCount % 2 == 0) {
                    writer1.write(line + "\n");
                } else {
                    writer2.write(line + "\n");
                }
                lineCount++;
            }
        }
    }
}
```

Se crean dos instancias de WordCounterProcessThread, cada una procesando uno de los archivos generados. Estos hilos ejecutan un comando del sistema operativo para contar las palabras.

```java
WordCounterProcessThread thread1 = new WordCounterProcessThread(part1FilePath);
WordCounterProcessThread thread2 = new WordCounterProcessThread(part2FilePath);
thread1.start();
thread2.start();
thread1.join();
thread2.join();
```

Para contar palabras, se utiliza **ProcessBuilder**. Dependiendo del sistema operativo:

- En Windows, se usa PowerShell con el comando Measure-Object -Word.
- En Linux/Mac, se emplea wc -w.

```java
private int countWordsInFileUsingProcess(String filePath) throws IOException,
InterruptedException {
    ProcessBuilder processBuilder = new ProcessBuilder();
    if (System.getProperty("os.name").toLowerCase().contains("win")) {
        processBuilder.command("powershell", "-command",
                "(Get-Content '" + filePath + "' | Out-String | Measure-Object
-Word).Words");
    } else {
        processBuilder.command("sh", "-c", "wc -w < '" + filePath + "'");
    }
    Process process = processBuilder.start();
    process.waitFor();
    try (BufferedReader reader = new BufferedReader(new
InputStreamReader(process.getInputStream()))) {
        String output = reader.readLine();
        if (output == null || output.trim().isEmpty()) {
            throw new IOException("No se pudo leer el conteo de palabras para "
+ filePath);
        }
        return Integer.parseInt(output.trim());
    }
}
```

Los hilos se sincronizan con **join(),** asegurando que el programa principal espere su finalización antes de mostrar el total de palabras.

Resultados

El programa ejecutado con un archivo de prueba produce la salida esperada, mostrando el conteo de palabras de cada parte y el total. Se verificó que los procesos se ejecutan correctamente en ambos sistemas operativos.

```
Palabras en part1.txt: 6260
Palabras en part2.txt: 6540
Procesamiento completado. Total de palabras: 12800
```

Conclusiones

Se logró implementar un programa en Java que combina hilos y procesos para el procesamiento de archivos, aunque también se identificaron las limitantes de usar hilos y procesos centralizados, como la posible sobrecarga de recursos del sistema y la complejidad en la sincronización. Esta práctica refuerza la importancia de comprender las ventajas y desventajas de la programación concurrente para optimizar el rendimiento de las aplicaciones.

Link repositorio: https://github.com/mmaaxi/p1-Distribuidos