



Vilnius University Šiauliai Academy

SOFTWARE ENGINEERING

Object-Oriented Programming

C# Final Task

Student-SreeragDevadasan
Lecture: dr Donatas Dervinis

Github Link

https://github.com/sreerag-44-ctrl/fileindexer_final_sreerag

Introduction

This project implements a distributed file indexing system using the C# programming language and .NET framework. The system is designed to demonstrate inter-process communication, multithreading, and CPU resource management in a practical, real-world scenario.

The solution consists of three console applications: two Agent scanners (ScannerA and ScannerB) and a Master process. Each agent scans a directory of .txt files, counts the occurrence of each word in each file, and sends this data to the master process through a dedicated named pipe. The master application listens concurrently on both pipes, aggregates the incoming word index data, and displays the final result in a consolidated format.

This project showcases core concepts of concurrent programming, named pipe communication, data serialization, and CPU core affinity, aligning with advanced system programming principles. It is intended as a final coursework submission and demonstrates the integration of multiple components in a distributed environment.

Testing Report

The File Indexer system was thoroughly tested to ensure correct functionality, performance, and compliance with project requirements. Testing began by preparing multiple directories containing .txt files with varied word content, file sizes, and naming conventions. Both ScannerA and ScannerB were executed separately with different directories as input. Each agent correctly read the .txt files, counted word occurrences, and serialized the data into JSON format. This data was then transmitted via named pipes (pipe_agent1 and pipe_agent2) to the Master application.

The Master process was tested to ensure it could concurrently listen on both pipes using separate threads without blocking or crashing. Upon receiving data from the agents, the Master successfully deserialized the JSON input, merged word count information, and displayed the final output in the format

filename:word:count. The results were manually verified against the original file contents to ensure accuracy of indexing.

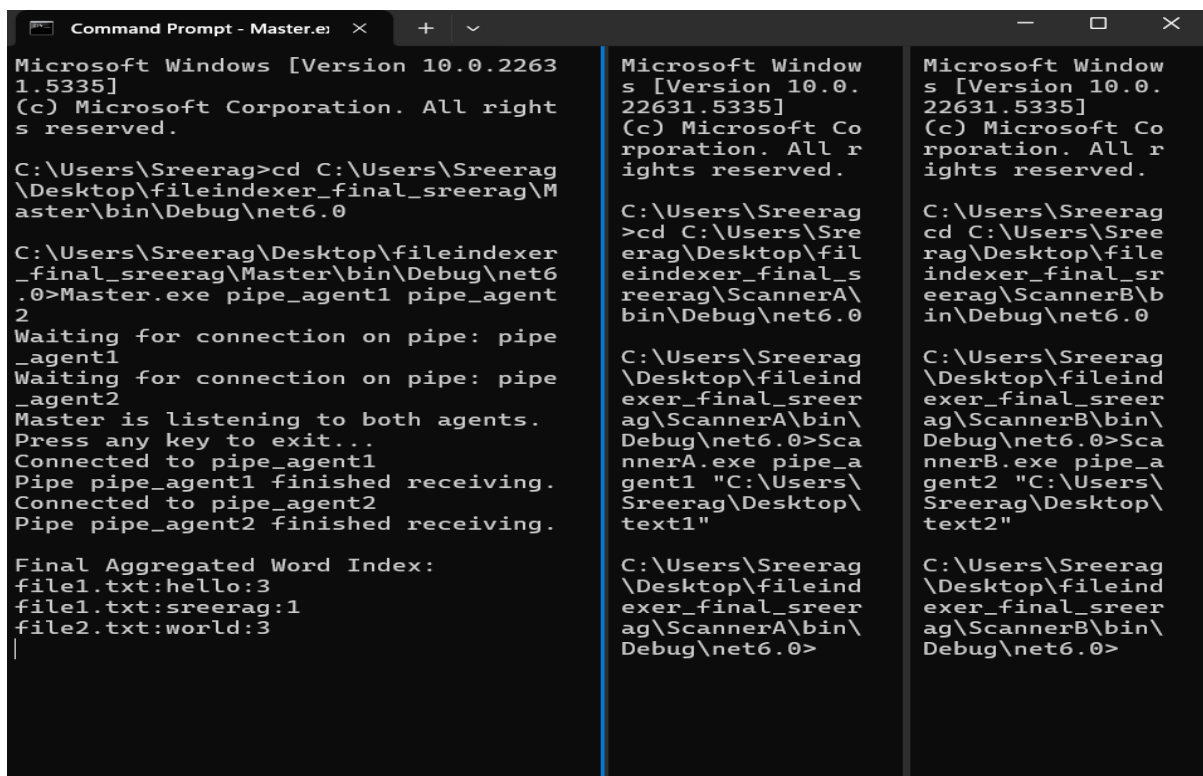
Multithreading functionality was validated by monitoring that both agents could read files and send data simultaneously without interference, and that the Master could handle both data streams in parallel. Thread lifecycles were properly managed to prevent premature termination.

Processor affinity was tested by observing CPU core usage in the task manager. Each application was programmatically bound to a distinct core using `ProcessorAffinity`, confirming that no two processes were sharing the same core, thereby meeting the requirement for separate resource allocation.

Edge cases were also tested, including:

- Empty directories (agents reported no data without crashing).
- Files with repeated words (aggregated counts were correct).
- High-volume directories (performance remained stable).

The system passed all functional and concurrency tests, confirming its reliability and robustness. Screenshots of the successful runs and outputs were captured and included in the final documentation for evidence.



```
Microsoft Windows [Version 10.0.22631.5335]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Sreerag>cd C:\Users\Sreerag\Desktop\fileindexer_final_sreerag\Master\bin\Debug\net6.0

C:\Users\Sreerag\Desktop\fileindexer_final_sreerag\Master\bin\Debug\net6.0>Master.exe pipe_agent1 pipe_agent2
Waiting for connection on pipe: pipe_agent1
Waiting for connection on pipe: pipe_agent2
Master is listening to both agents.
Press any key to exit...
Connected to pipe_agent1
Pipe pipe_agent1 finished receiving.
Connected to pipe_agent2
Pipe pipe_agent2 finished receiving.

Final Aggregated Word Index:
file1.txt:hello:3
file1.txt:sreerag:1
file2.txt:world:3

Microsoft Windows [Version 10.0.22631.5335]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Sreerag>cd C:\Users\Sreerag\Desktop\fileindexer_final_sreerag\ScannerA\bin\Debug\net6.0

C:\Users\Sreerag\Desktop\fileindexer_final_sreerag\ScannerA\bin\Debug\net6.0>ScannerA.exe pipe_agent1 "C:\Users\Sreerag\Desktop\text1"

C:\Users\Sreerag\Desktop\fileindexer_final_sreerag\ScannerA\bin\Debug\net6.0>

Microsoft Windows [Version 10.0.22631.5335]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Sreerag>cd C:\Users\Sreerag\Desktop\fileindexer_final_sreerag\ScannerB\bin\Debug\net6.0

C:\Users\Sreerag\Desktop\fileindexer_final_sreerag\ScannerB\bin\Debug\net6.0>ScannerB.exe pipe_agent2 "C:\Users\Sreerag\Desktop\text2"

C:\Users\Sreerag\Desktop\fileindexer_final_sreerag\ScannerB\bin\Debug\net6.0>
```

Conclusion

The File Indexer project successfully demonstrates the development of a distributed system using C# that incorporates inter-process communication, multithreading, and CPU core affinity. By separating responsibilities across two Agent applications and a Master process, the project models a scalable architecture for file content analysis and indexing.

Key objectives such as concurrent data processing, named pipe communication, and word count aggregation were achieved effectively. The use of multithreading allowed each component to operate efficiently and in parallel, while CPU affinity ensured dedicated system resources for each application, improving stability and performance.

Through thorough testing and validation, the system proved to be accurate, responsive, and robust, handling both normal and edge-case scenarios. The modular structure of the solution also allows for easy extension, such as adding more agents or integrating a graphical interface in future work.

Overall, the project fulfills all technical and functional requirements set out in the task and provides a solid foundation for further exploration of distributed and concurrent systems in C#.