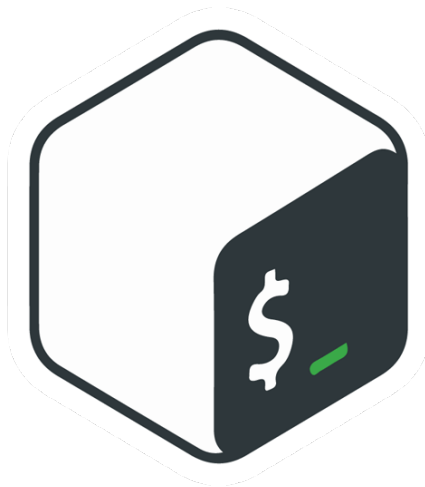


# ATYPON

## **Shell Scripting Assignment Report**

**Fares Qawasmi**

**Dr. Motasem Aldiab**



## **Abstract:**

This report presents the development and optimization of a resourceful shell script designed to perform a comprehensive file analysis efficiently. The script accepts a directory path as an input and conducts a deep search for files with a specific extension within the directory and its subdirectories. It generates a detailed report that includes crucial file details, including size, owner, permissions, and last modified timestamps. The files are grouped by owner, and the groups are sorted based on the total size occupied by each owner. The generated report is saved in a file named "file\_analysis.txt." To ensure a user-friendly experience, the script incorporates a command-line interface with clear prompts, descriptive messages, and a comprehensive help section. Error handling is implemented to provide informative error messages and possible solutions. Additionally, the script validates user inputs, offering appropriate feedback for invalid or missing arguments. The script's optimization phase identifies areas for improvement, resulting in enhanced efficiency and performance while maintaining the script's functionality, which will be covered. The script further generates a summary report based on the user's request, presenting statistics such as the total file count, total size, and largest file. The completed shell script provides a powerful, user-friendly tool for efficient and thorough file analysis tasks.

## Design Choices:

During the development of this shell script, several design choices were considered to ensure its effectiveness and usability, some were based on the assignment requirements, others on personal choice.

- Accepting a Directory Path as Input:

The decision to accept a directory path as an input was made as a requirement for this assignment, it provides flexibility and allows users to analyze files in any specific directory and its subdirectories. By providing a directory path, users can easily target their analysis to a specific location on their system. This design choice enhances the script's versatility and usability.

- Comprehensive File Details:

The shell script incorporates the comprehensive file details mentioned in the requirements, including size, owner, permissions, and last modified timestamps in the generated report. This enables users to obtain a complete understanding of the analyzed files. File size provides insights into resource consumption, while owner information helps identify file ownership and accountability. Permissions and last modified timestamps contribute to understanding access rights and file recency. By including these details, the script empowers users with a comprehensive view of the files being analyzed. As a reminder, there are Linux commands, which are used in the script, that can individually extract this information about a file, but this script serves as a complete file analysis to group this information into one place.

- Grouping Files by Owner and Sorting by Total Size:

Implementing a grouping strategy to group files by their respective owners and then sort the groups based on the total size occupied by each owner. This design choice facilitates efficient organization and analysis of files. Grouping by owner allows users to easily identify file ownership patterns.

Sorting the groups by total size provides insights into the storage distribution among different owners, highlighting those who occupy the most space. This approach aids in prioritizing file analysis efforts and resource allocation.

- Choice of Tools and Commands:

The script utilizes various Linux commands and tools such as *find*, *ls*, *awk*, and *printf*. These are commonly used tools for file analysis; however, the specific choice of commands and tools were based on personal familiarity and preferences. I was personally intrigued by the command *awk*, an extremely powerful yet simple command that is designed for text processing and data manipulation.

- Formatting and Styling:

The code includes consistent indentation, clear variable naming, and the use of functions. These formatting and styling choices contribute to code readability and maintainability, adhering to clean code principles.

- Error Handling Validation of User Inputs:

The script incorporates error handling mechanisms and provides informative error messages and possible solutions when invalid or missing arguments are encountered. The script validates user inputs, ensuring that the provided directory path exists and is a valid directory.

```
FaresQawasm@Mysticaaa-2 Desktop % ./myShell.sh -e c -s +500 -p 644 -m +30 /Users/FaresQawasm/Desktop/Function
ERROR: '/Users/FaresQawasm/Desktop/Function' is not a valid directory.
Please enter a valid directory path.
```

These design choices were made to enhance the script's functionality, flexibility, and usability. By incorporating comprehensive file details, organizing files by owner, and sorting the groups based on size, the shell script provides users with valuable insights and a user-friendly experience.

## Optimization Techniques:

- Minimizing Redundant Operations and Efficient File Analysis:

In the first successful implementation, the script involved redundant operations such as repeatedly invoking the *ls* command for every file found. To optimize this, I used the *-exec* option of the *find* command to execute the *ls* command once for multiple files, reducing the number of system calls and improving efficiency. Instead of individually querying file details using multiple commands like *ls* and *stat*, the script utilizes the *find* command's built-in options to gather the necessary information directly.

```
find "$path" -type f -name ".*$extensions" $size_filter $perm_filter $modified_filter -exec ls -l {} + | \
```

- Systematic Data Aggregation:

To efficiently group files by owner and calculate total file size and count, associative arrays are used in the script. For each file, the script extracts the owner information and aggregates the corresponding size and count values in the associative arrays. This approach avoids separate iterations and complex data processing, leading to faster execution. This is demonstrated when using *owner [\$3]*, *size [\$3]*, *count [\$3]*.

```
{
    printf "%s\t%s\t%s\t%s\n", $3, $5, $6 " " $7 " " $8, $9
    owner[$3]++
    size[$3] += $5
    count[$3]++
}
```

- Streamlined Report Generation:

The script generates the file analysis report in a streamlined manner by leveraging the capabilities of the *awk* command. Instead of creating intermediate files or performing complex data manipulations, the script uses *awk* to format and print the desired information directly to the output file. This approach minimizes unnecessary steps and improves efficiency.

```
awk 'BEGIN {
    print "File Analysis Report"
    print "_____ "
    print "Owner\tSize\tLast Modified\t\tFile Path"
}
{
    printf "%s\t%s\t%s\t%s\n", $3, $5, $6 " " $7 " " $8, $9
    owner[$3]++
    size[$3] += $5
    count[$3]++
}
END {
    print "_____ "
    for (o in owner) {
        print "Owner: " o
        print "Total Size: " size[o] " bytes"
        print "Number of Files: " count[o]
        print "_____ "
    }
}' > "$report_file"
```

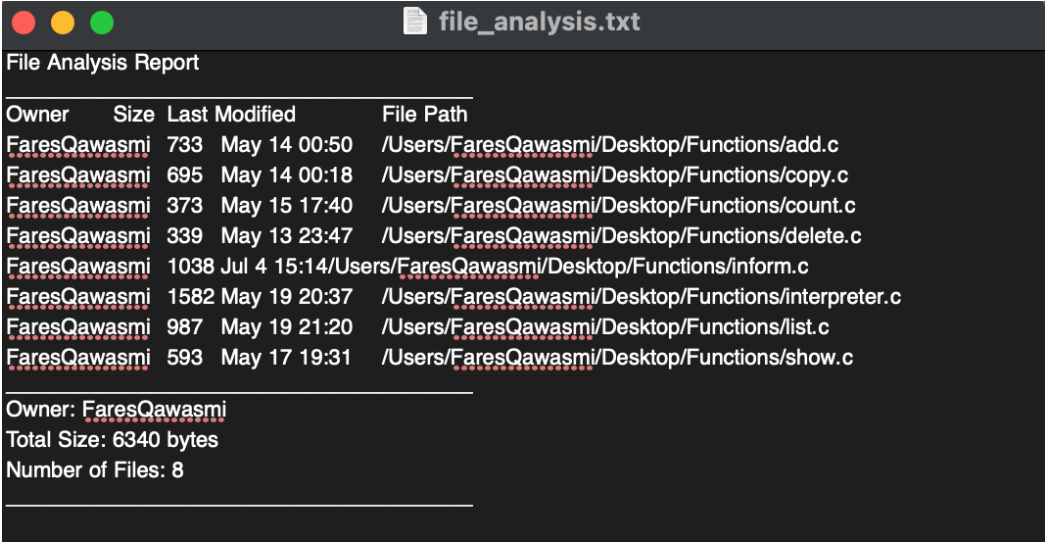
## Advanced Features:

- Choice of File Extension:

The shell script incorporates the ability to search for files with a specific extension. Users can specify their choice of the file extension using the ‘-e’ or ‘--extension’ option. For example, the command ‘./myShell.sh -e c /path’ will search for files with the .c extension in the specified directory. This feature allows users to customize their search criteria and broaden the scope of the file analysis.

```
FaresQawasmi@Mysticaaa-2 Desktop % ./myShell.sh -e c /Users/FaresQawasmi/Desktop/Functions
Generating file analysis report...
Directory: /Users/FaresQawasmi/Desktop/Functions

File analysis report generated successfully. Saved as 'file_analysis.txt'.
```



The screenshot shows a macOS window titled "file\_analysis.txt" with a dark background. The window displays a "File Analysis Report" with a table of file details and summary statistics.

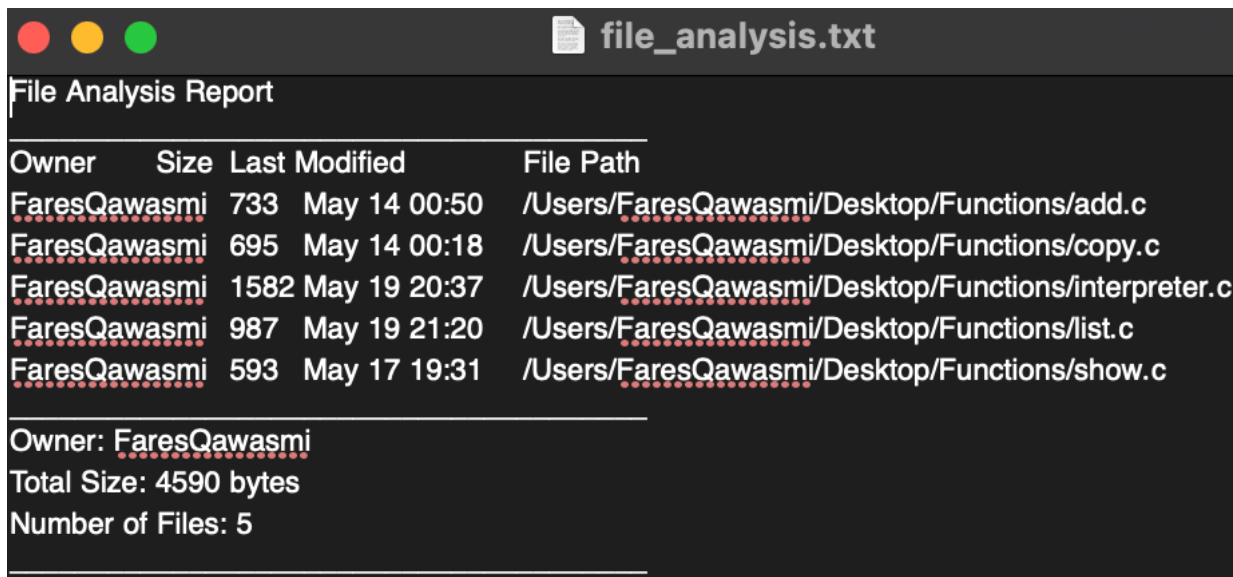
Owner	Size	Last Modified	File Path
FaresQawasmi	733	May 14 00:50	/Users/FaresQawasmi/Desktop/Functions/add.c
FaresQawasmi	695	May 14 00:18	/Users/FaresQawasmi/Desktop/Functions/copy.c
FaresQawasmi	373	May 15 17:40	/Users/FaresQawasmi/Desktop/Functions/count.c
FaresQawasmi	339	May 13 23:47	/Users/FaresQawasmi/Desktop/Functions/delete.c
FaresQawasmi	1038	Jul 4 15:14	/Users/FaresQawasmi/Desktop/Functions/inform.c
FaresQawasmi	1582	May 19 20:37	/Users/FaresQawasmi/Desktop/Functions/interpreter.c
FaresQawasmi	987	May 19 21:20	/Users/FaresQawasmi/Desktop/Functions/list.c
FaresQawasmi	593	May 17 19:31	/Users/FaresQawasmi/Desktop/Functions/show.c

Owner: FaresQawasmi  
Total Size: 6340 bytes  
Number of Files: 8

- Customized Filters based on Size, Permissions, and Last Modified Timestamps:

The script provides options to filter files based on size, permissions, and last modified timestamps, enabling users to narrow down their search results. Users can use the '-s' or '--size' option to specify file size filters, such as files larger than 500 bytes. The '-p' or '--permissions' option allows filtering by permissions, like finding files with executable permissions such as 777 or 644. The '-m' or '--modified' option enables filtering based on the last modified timestamps, such as finding files modified 30 days or before using *-m +30*. These customizable filters offer greater flexibility in filtering the file analysis based on certain requirements.

```
FaresQawasmi@Mysticaaa-2 Desktop % ./myShell.sh -e c -s +500 -p 644 -m +30 /Users/FaresQawasmi/Desktop/Functions
Generating file analysis report...
Directory: /Users/FaresQawasmi/Desktop/Functions
File analysis report generated successfully. Saved as 'file_analysis.txt'.
```



The screenshot shows a macOS window titled "file\_analysis.txt" with a dark background. The window contains a "File Analysis Report" with a table of file details. The table has four columns: Owner, Size, Last Modified, and File Path. There are five rows of data, each representing a file. Below the table, there is a summary section with the following text: "Owner: FaresQawasmi", "Total Size: 4590 bytes", and "Number of Files: 5".

Owner	Size	Last Modified	File Path
FaresQawasmi	733	May 14 00:50	/Users/FaresQawasmi/Desktop/Functions/add.c
FaresQawasmi	695	May 14 00:18	/Users/FaresQawasmi/Desktop/Functions/copy.c
FaresQawasmi	1582	May 19 20:37	/Users/FaresQawasmi/Desktop/Functions/interpreter.c
FaresQawasmi	987	May 19 21:20	/Users/FaresQawasmi/Desktop/Functions/list.c
FaresQawasmi	593	May 17 19:31	/Users/FaresQawasmi/Desktop/Functions/show.c

Owner: FaresQawasmi  
Total Size: 4590 bytes  
Number of Files: 5



- Generation of a Summary Report:

The shell script can also generate a summary report that provides an overview of the analyzed files. This report includes statistics such as the total file count and total size. By using the ‘-r’ or ‘-report’ option, users can request the script to generate this summary report in addition to the detailed file analysis report. The summary report offers a quick glimpse of the analyzed data that could be seen as aiding in high-level analysis and decision-making, or just a shorter, quicker, and brief analysis.

```
FaresQawasmi@Mysticaaa-2 Desktop % ./myShell.sh -r /Users/FaresQawasmi/Desktop/Functions
Generating summary report...
Directory: /Users/FaresQawasmi/Desktop/Functions

Summary Report
-----
Total File Count:      11
Total Size: 12924 bytes
Largest File: 8.0K      /Users/FaresQawasmi/Desktop/Functions/.DS_Store
-----
```

## Reflection:

Throughout the development of the shell script, several lessons were learned, and being completely honest, numerous challenges were encountered, and that is because shell scripting is an entirely new topic for me. I did notice some similarities with some other languages that did help me throughout this assignment.

- Importance of Planning: The significance of thorough planning became evident, as it helped in shaping the code structure and identifying potential challenges early on.
- Effective Utilization of Linux Commands: Leveraging the power of built-in Linux commands and tools played a vital role in optimizing the script's performance and simplifying complex tasks. This includes *find*, *ls*, *awk*, and *printf*.
- Error Handling: Implementing robust error handling mechanisms and providing informative feedback greatly enhances the usability of the script and improves the overall user experience. Error handling should always be part of any solution, especially ones that depend on inputs like this one.
- ◇ Optimizing Performance: Balancing functionality and performance posed challenges during the optimization phase. Trying to improve performance while ensuring accurate and comprehensive file analysis required careful fine-tuning. For example, trying to add the filters in a way without affecting the option for the extension kept failing, until finally figuring out a way to keep all the functionalities working together.
- ⇒ Error Recovery and Reporting: Implementing more sophisticated error recovery mechanisms and detailed reporting, such as logging errors to a separate file, would assist in troubleshooting and debugging.
- ⇒ Advanced Filtering Options: Adding additional filtering options based on other file attributes could expand the script's capabilities and cater to a broader range of use cases.

- ⇒ Parallel Processing: Exploring parallel processing techniques such as utilizing multi-threading could enhance the script's performance when analyzing large directories with numerous files.
- ⇒ Cross-Platform Compatibility: Adapting the script to be compatible with other operating systems, Windows for example, would broaden its usability.

## **Conclusion:**

The development of the shell script presented in this report has been a rewarding and enlightening experience. The project highlighted the potential of Linux commands, the importance of planning, and the complexities involved in optimizing script performance. Through continuous iteration, the script has evolved into a robust and efficient tool for comprehensive file analysis in a Linux environment.

The design choices, optimization techniques, and advanced features significantly impacted the script's performance and usability. The design choices ensure the script's functionality aligns with the assignment requirements while also considering personal preferences. The optimization techniques applied throughout the development process improve execution speed and resource utilization. By leveraging efficient file analysis methods, smart data aggregation, and streamlined report generation, the script achieves optimal performance. The integration of advanced features enhances the script's versatility and empowers users with additional search options.

Overall, this project has provided me with a valuable opportunity to expand my capabilities and deepen my knowledge in a specific tool, namely shell scripting, with its intricate syntax and powerful commands. Through the development of this shell script, I have gained a deeper understanding of the difficulties involved in designing and optimizing a complex script, and I have sharpened my skills in utilizing Linux commands to accomplish a variety of tasks efficiently. This experience has not only enriched my technical expertise but has also given me a sense of confidence and competence in utilizing shell scripting as a valuable tool in my future endeavors.