Shell Scripting (report)

Akram Jaghoub

1.0 Introduction

This document provides an overview of a shell script designed to analyze files in a directory and generate a detailed report. The script offers features such as grouping files by owner, sorting by size, and including information like permissions and last modified timestamp. It incorporates user-friendly prompts, advanced filtering options, and optimizations for improved performance, and finally save the analysis done in the file named file_analysis.txt. The script's comprehensive report and efficient file analysis make it a valuable tool for file management tasks.

2.0 design choices

1. Modular Function Structure: The script is structured using multiple functions, each responsible for a specific task. This modular approach improves code organization and allows for better code maintenance and readability.

2. Comprehensive Output Format: The output format of the file analysis report is designed to provide a clear and comprehensive overview of each file. The report includes the owner, total size per owner, and detailed information for each file, including file name, permissions, owner, size, and last modified timestamp. This format enhances readability and facilitates understanding of the file analysis results.

3.0 Optimization techniques

1. A design choice was made to optimize the code by replacing the use of the "ls" command with the more efficient "stat" command. This change was implemented in the section responsible for retrieving file information (owner, size, permissions, last modified). By utilizing the "stat" command, the code avoids the overhead associated with multiple "ls" commands and improves the performance and speed of the script. The "stat" command is used to extract the required file attributes, which are then processed and stored for further analysis. This design choice demonstrates a conscious effort to enhance efficiency and streamline the execution of the script.

2. Another optimization was achieved by utilizing the "find" command to search for files with the specified extensions in the given directory and its subdirectories. This optimization eliminated the need for a loop to iterate through individual files, which can be computationally expensive. By using the "find" command with appropriate options such as file size, permissions, and last modified time, the script efficiently retrieves the desired files in a single statement. This approach simplifies the code and improves its performance by leveraging the built-in capabilities of the "find" command.

4.0 Advanced Features

- Support for Multiple File Extensions: To enable users to search for files with various extensions simultaneously, the script accepts a list of file extensions as input. In the code, the extensions are stored in an array called "EXTENSIONS" to allow for multiple extensions. The script utilizes regular expressions and the -regex command in the find operation to match files with any of the specified extensions. The script dynamically constructs a regex pattern by iterating over the extension list and joining them with the "|" operator. This allows the script to efficiently search for files matching any of the specified extensions.

- File Filtering Options: The script provides options for users to filter files based on size, permissions, or last modified timestamp. Users can provide size filters using the -s or --size option, permissions filters using the -p or --permissions option, and last modified filters using the -m or --modified option. The script includes separate functions for validating each filter option: validate_size_filter(), validate_permissions_filter(), and validate_last_modified_filter(). These functions ensure that the provided filter arguments are valid and then incorporate them into the find operation accordingly. The script handles different filter formats and allows users to customize their file search based on specific criteria.

- Summary Report Generation: The script includes a generate_report() function that generates a summary report of the file analysis. Which includes the total file count and total size of all files analyzed. The script creates an output file named "file_analysis.txt" and writes the report content into it. The summary report provides users with a concise overview of the analyzed files, allowing them to quickly assess the overall file distribution and characteristics.

5.0 User friendliness

1. Clear Prompts and Descriptive Messages: The script provides clear prompts and descriptive messages throughout, guiding users and providing them with relevant information at each step. This helps users understand the script's actions and respond appropriately.

2. Help Section: The script includes a comprehensive help section that explains the script's purpose, usage instructions, and available options. This allows users to quickly understand how to interact with the script and make informed choices.

3. Input Validation and Error Handling: The script validates user input and handles errors effectively. It ensures that provided directory paths, file extensions, size filters, permission filters, and last modified filters are valid. If any invalid or missing arguments are detected, the script displays informative error messages, guiding users to correct their input.

6.0 Reflection

6.1 lessons learned

- Researching and Leveraging Efficient Commands: Conducting research on command alternatives, such as using "stat" instead of "ls" and leveraging the power of the "find" command, proved crucial in optimizing performance and reducing resource overhead.

- Through the process of validating user input, I gained valuable insights into the importance of considering the user's perspective and ensuring a smooth experience. This experience enhanced my problem-solving skills as I learned to identify and address potential issues independently, resulting in improved script functionality and user satisfaction.

- The importance of providing an easy and intuitive interface for users. By implementing clear prompts, descriptive messages, and a comprehensive help section, I aimed to ensure that users can interact with the script effortlessly. This user-friendly approach enhances the overall experience and reduces the likelihood of errors or confusion during script usage.

6.2 difficulties encountered

- One of the difficulties I encountered was figuring out how to support multiple file extensions. And I solved this problem by using regular expressions and the -regex command in the find.

- Another difficulty that really took some extensive time to fully implement was the filters based on size, permissions, and last modified timestamp. I eventually solved this problem by using the find command which provided an easy interface to deal with these filters.

- Trying to think about all the possible errors the users make was a challenging part. To overcome the challenges, I pinpointed the problems and then actively worked on solving them in an efficient manner. For instance, I performed the necessary validations on the directory, extensions, and filtering techniques.

6.3 potential future improvements

- Enhancing error handling by providing more detailed error messages and suggestions for troubleshooting would further improve the user experience. This could include providing specific guidance on how to resolve common issues or offering alternative options, also implementing features like autocomplete or suggestions when entering directory paths or file extensions.

- Expand the script to include additional file management tasks such as file deletion, renaming, and compression. This would provide users with a more comprehensive tool for managing files within the specified directory.

- Allow users to have the ability to group files based on different criteria, such as file type, owner, or permissions. They can also customize the sorting order based on file size, last modified timestamp, or alphabetical order. These options provide users with greater flexibility in organizing and arranging the file analysis results according to their specific needs and preferences.