

Shell Scripting Assignment

By: Ibrahim Jaradat.

In this assignment we were required to design a shell script that accomplishes some tasks maintaining a complete, functional, clear and optimized implementation.

-Implementation

First of all I implemented this function called - displayHelp() - that displays the help section, I made it as a function to call it whenever I prompt an error to the user so he can see the options and correct usage of the script.

```
displayHelp() {  
    echo "  
    echo "Usage: script.sh [extension1 extension2 ... ] [directory]"  
    echo "  
    echo "This script searches for files with the specified extensions (e.g., .txt) in the given directory and its subdirectories  
    echo "then it generates a report with file details grouped by owner and sorted (ascending) by total size."  
    echo "  
    echo "Usage with file filters: script.sh [options] [extension1 extension2 ... ] [directory]"  
    echo "  
    echo " Options:"  
    echo " -h: Display this help section"  
    echo " -s <size>: Filter files by size (in bytes)"  
    echo " -p <permissions>: Filter files by permissions (in octal format)"  
    echo " -t <timestamp>: Filter files by last modified timestamp after the specified date in (YYYY-MM-DD) format"  
    echo " -r: Generate a summary report in a separate file"  
}
```

Then I have this function called -addFilters()- so I can save the provided filters (size, permissions, timestamp) in an array and returns the contents of it as a space-separated string so I can find the files according to these filters later on in the code.

```
addFilters() {  
    local filters=()  
  
    if [ -n "$sizeFilter" ]; then  
        filters+=("-size $sizeFilter")  
    fi  
  
    if [ -n "$permissionsFilter" ]; then  
        filters+=("-perm $permissionsFilter")  
    fi  
  
    if [ -n "$timestampFilter" ]; then  
        filters+=("-newermt $timestampFilter")  
    fi  
  
    echo "${filters[*]}"  
}
```

To get the options I used a while loop using the `-getopts`- this Shell built-in command retrieves options and its corresponding arguments. Inside the loop I used a case statement to check the value of the option to determine what to do for each option. I assign the value of the argument to the filter variable when the options (`-s`, `-p`, `-t`, `-r`) or call the `displayHelp()` function when `case(-h)`.

Also `:` prints an error message when an option requires an argument but none is provided and `*` prints an error message when an invalid option is encountered.

```
while getopts ":s:p:t:hr" opt; do
    case $opt in
        s ) sizeFilter=$OPTARG ;;
        p ) permissionsFilter=$OPTARG ;;
        t ) timestampFilter=$OPTARG ;;
        r ) summaryReport=true ;;
        h ) displayHelp
            exit 0 ;;
        : ) echo "Option requires an argument."
            displayHelp
            exit 1 ;;
        * ) echo "Invalid option"
            displayHelp
            exit 1 ;;
    esac
done
```

```
shift $((OPTIND-1))
```

After finishing the loop I discard the options and their arguments by shifting the parameters to the left .

```
if [ "$#" -lt 2 ]; then
    echo "Error: Invalid number of arguments."
    displayHelp
    exit 1
fi

extensions=("${@:1:$#-1}")
directory="${@:$#}"
```

This if statement makes sure that the remaining arguments are more than 2 (the extension and the directory) , and if any error occurs to display an error with the help section.

```

if [ ! -d "$directory" ]; then
    echo "Error: make sure to enter a valid directory."
    displayHelp
    exit 1
fi

```

Here I make sure that the user entered an existing directory, if not or not specified display an error with the help section.

```

reportFile="file_analysis.txt"
rm -f "$reportFile"

```

```

filters=$(addFilters)

```

In this step I create a variable to store the name of the report file "file_analysis.txt", and then remove it to ensure that any existing report file is deleted before generating a new report file, And a variable that stores the return statement of the addFilters() function.

```

for extension in "${extensions[@]"; do
    find "$directory" -type f -name ".*$extension" $filters -exec stat -c "%s %U %a %y %n" {} + >> "$reportFile"
done

```

This for each statement is responsible to search for files with the specified extensions in the given directory and its subdirectories then appending the founded file details to the report file.

The (-type f) command is to make sure to search for regular files only, not anything else. And the (-name ".*\$ext" \$filters) command searches for files with the current extension as part of their names while applying the filters specified before to the find command.

The (-exec stat -c "%s %U %a %y %n" {} +) command formats the output to retrieve the file size, owner, permissions, last modified timestamp, and filename.

```
sort -k2,2 -k1,1n -o "$reportFile" "$reportFile"
```

This sort command is used with (-k) to sort the file based on the contents of the columns in the file, so first it will sort the file by the second column which is the user and then sort by the first column which is the size.

```
if [ "$summaryReport" = true ]; then
    summaryReportFile="summary_report.txt"
    rm -f "$summaryReportFile"
    totalFiles=$(wc -l < "$reportFile")
    totalSize=$(awk '{sum+=$1} END {print sum}' "$reportFile")
    averageSize=$(awk '{sum+=$1} END {print sum/NR}' "$reportFile")
    largestFile=$(awk '{if ($1 > max) max=$1} END {print max}' "$reportFile")
    smallestFile=$(awk 'NR==1 { min = $1 } $1 < min { min = $1 } END { print min }' "$reportFile")
    mostRecentEditedFile=$(awk '{print $4, $5}' "$reportFile" | sort -r | head -n 1)
    userFileSize=$(awk '{sum[$2]+=$1} END {for (i in sum) print i, sum[i]}' "$reportFile")

    echo "" >> "$reportFile"
    echo "Total number of files: $totalFiles" >> "$summaryReportFile"
    echo "Total file size: $totalSize bytes" >> "$summaryReportFile"
    echo "Average file size: $averageSize bytes" >> "$summaryReportFile"
    echo "Largest file size: $largestFile bytes" >> "$summaryReportFile"
    echo "Smallest file size: $smallestFile bytes" >> "$summaryReportFile"
    echo "Most recent edit on file: $mostRecentEditedFile" >> "$summaryReportFile"
    echo "users has files of total size: " >> "$summaryReportFile"
    echo "$userFileSize" >> "$summaryReportFile"
    echo "summary_report has been saved to: $summaryReportFile"
fi
```

Finally, if the create summary report flag is true create a new file containing some relevant statistics such as (Total number of files, total size of files for each user, largest file, etc).

-Using the script

The script is designed to search for files of certain extension, so first of all the user can get help by using the command : -h

```
ibrahemjr@ubuntu:~$ ./script.sh -h
Usage: script.sh [extension1 extension2 ... ] [directory]

This script searches for files with the specified extensions (e.g., .txt) in the given directory and its subdirectories.
Then it generates a report with file details grouped by owner and sorted (ascending) by total size.

Usage with file filters: script.sh [options] [extension1 extension2 ... ] [directory]

Options:
-h: Display this help section
-s <size>: Filter files by size (in bytes)
-p <permissions>: Filter files by permissions (in octal format)
-t <timestamp>: Filter files by last modified timestamp after the specified date in (YYYY-MM-DD) format
-r: Generate a summary report in a separate file
ibrahemjr@ubuntu:~$
```

This command displays the help section that explains how the script works and how to use the script, also it lists available options for the user to choose from.

Then the user can use the command :

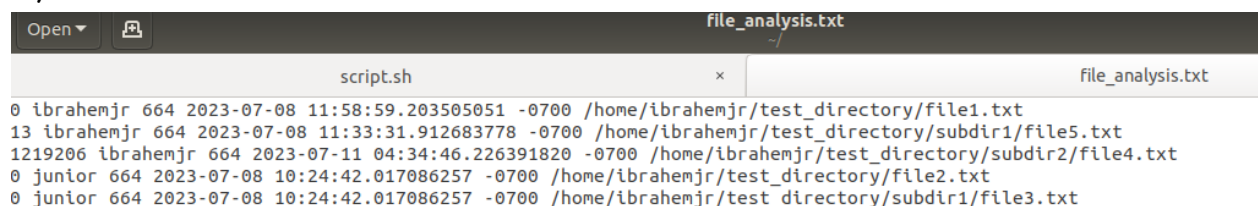
```
txt /home/ibrahemjr/test_directory
```

Where he specifies the extension of the file and the directory he wants to search in.

```
ibrahemjr@ubuntu:~$ ./script.sh txt /home/ibrahemjr/test_directory
Searching for files with extensions 'txt' in '/home/ibrahemjr/test_directory'...
The report has been saved to: file_analysis.txt
```

What that would do is to generate a file called "file_analysis.txt" that contains the search result including all the file's details such as size, owner, permissions and last modified timestamp.

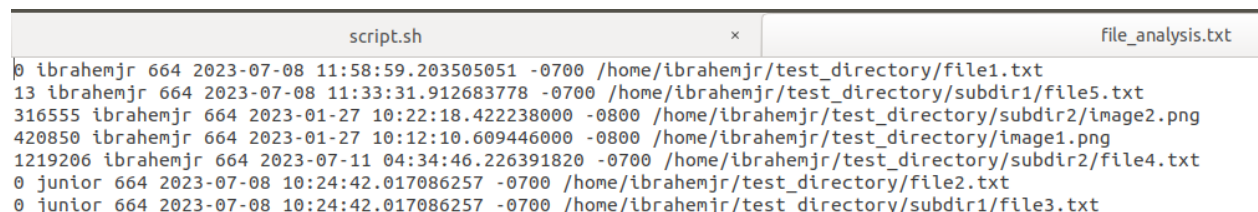
The "file_analysis.txt" has the result grouped by user and sorted by file size.



	script.sh	file_analysis.txt
0	ibrahemjr	664 2023-07-08 11:58:59.203505051 -0700 /home/ibrahemjr/test_directory/file1.txt
13	ibrahemjr	664 2023-07-08 11:33:31.912683778 -0700 /home/ibrahemjr/test_directory/subdir1/file5.txt
1219206	ibrahemjr	664 2023-07-11 04:34:46.226391820 -0700 /home/ibrahemjr/test_directory/subdir2/file4.txt
0	junior	664 2023-07-08 10:24:42.017086257 -0700 /home/ibrahemjr/test_directory/file2.txt
0	junior	664 2023-07-08 10:24:42.017086257 -0700 /home/ibrahemjr/test_directory/subdir1/file3.txt

Also, the user can use the script to search for multiple file extensions simultaneously.

```
ibrahemjr@ubuntu:~$ ./script.sh txt png /home/ibrahemjr/test_directory
Searching for files with extensions 'txt png' in '/home/ibrahemjr/test_directory'...
The report has been saved to: file_analysis.txt
ibrahemjr@ubuntu:~$
```



	script.sh	file_analysis.txt
0	ibrahemjr	664 2023-07-08 11:58:59.203505051 -0700 /home/ibrahemjr/test_directory/file1.txt
13	ibrahemjr	664 2023-07-08 11:33:31.912683778 -0700 /home/ibrahemjr/test_directory/subdir1/file5.txt
316555	ibrahemjr	664 2023-01-27 10:22:18.422238000 -0800 /home/ibrahemjr/test_directory/subdir2/image2.png
420850	ibrahemjr	664 2023-01-27 10:12:10.609446000 -0800 /home/ibrahemjr/test_directory/image1.png
1219206	ibrahemjr	664 2023-07-11 04:34:46.226391820 -0700 /home/ibrahemjr/test_directory/subdir2/file4.txt
0	junior	664 2023-07-08 10:24:42.017086257 -0700 /home/ibrahemjr/test_directory/file2.txt
0	junior	664 2023-07-08 10:24:42.017086257 -0700 /home/ibrahemjr/test_directory/subdir1/file3.txt

In addition, the user has the ability to filter the files by more options such as (size, permissions, timestamp) by choosing one or more of these filtering options.

Here for example we used the (-s) option and gave it (+0) as a threshold to get the files that have the size larger than zero.

```
ibrahemjr@ubuntu:~$ ./script.sh -s +0 txt png /home/ibrahemjr/test_directory
Searching for files with extensions 'txt png' in '/home/ibrahemjr/test_directory'...
The report has been saved to: file_analysis.txt
```

```
13 ibrahemjr 664 2023-07-08 11:33:31.912683778 -0700 /home/ibrahemjr/test_directory/subdir1/file5.txt
316555 ibrahemjr 664 2023-01-27 10:22:18.422238000 -0800 /home/ibrahemjr/test_directory/subdir2/image2.png
420850 ibrahemjr 664 2023-01-27 10:12:10.609446000 -0800 /home/ibrahemjr/test_directory/image1.png
1219206 ibrahemjr 664 2023-07-11 04:34:46.226391820 -0700 /home/ibrahemjr/test_directory/subdir2/file4.txt
```

Or the user can use multiple filters at the same time, here the user is searching for files where size is bigger than zero , permission (664) , and the last modified time stamp is after 2023-07-01.

```
ibrahemjr@ubuntu:~$ ./script.sh -s +0 -p 664 -t 2023-07-01 txt png /home/ibrahemjr/test_directory
Searching for files with extensions 'txt png' in '/home/ibrahemjr/test_directory'...
The report has been saved to: file_analysis.txt
```

```
13 ibrahemjr 664 2023-07-08 11:33:31.912683778 -0700 /home/ibrahemjr/test_directory/subdir1/file5.txt
1219206 ibrahemjr 664 2023-07-11 04:34:46.226391820 -0700 /home/ibrahemjr/test_directory/subdir2/file4.txt
```

Finally we have an option to generate a summary report that displays some relevant statistics using (-r) option.

```
ibrahemjr@ubuntu:~$ ./script.sh -r txt png /home/ibrahemjr/test_directory
Searching for files with extensions 'txt png' in '/home/ibrahemjr/test_directory'...
The report has been saved to: file_analysis.txt
summary_report has been saved to: summary_report.txt
```

```
0 ibrahemjr 664 2023-07-08 11:58:59.203505051 -0700 /home/ibrahemjr/test_directory/file1.txt
13 ibrahemjr 664 2023-07-08 11:33:31.912683778 -0700 /home/ibrahemjr/test_directory/subdir1/file5.txt
316555 ibrahemjr 664 2023-01-27 10:22:18.422238000 -0800 /home/ibrahemjr/test_directory/subdir2/image2.png
420850 ibrahemjr 664 2023-01-27 10:12:10.609446000 -0800 /home/ibrahemjr/test_directory/image1.png
1219206 ibrahemjr 664 2023-07-11 04:34:46.226391820 -0700 /home/ibrahemjr/test_directory/subdir2/file4.txt
0 junior 664 2023-07-08 10:24:42.017086257 -0700 /home/ibrahemjr/test_directory/file2.txt
0 junior 664 2023-07-08 10:24:42.017086257 -0700 /home/ibrahemjr/test_directory/subdir1/file3.txt
```

```
Total number of files: 7
Total file size: 1956624 bytes
Average file size: 279518 bytes
Largest file size: 1219206 bytes
Smallest file size: 0 bytes
Most recent edit on file: 2023-07-11 04:34:46.226391820
users has files of total size:
ibrahemjr 1956624
junior 0
```

-Handling errors and feedback

It is important to help the user use the script correctly without issues, so I tried my best to acknowledge the user of any errors or issues by giving the appropriate feedback.

```
ibrahemjr@ubuntu:~$ ./script.sh
Error: Invalid number of arguments.

Usage: script.sh [extension1 extension2 ... ] [directory]

This script searches for files with the specified extensions (e.g., .txt) in the given directory and its subdirectories.
then it generates a report with file details grouped by owner and sorted (ascending) by total size.

Usage with file filters: script.sh [options] [extension1 extension2 ... ] [directory]

Options:
-h: Display this help section
-s <size>: Filter files by size (in bytes)
-p <permissions>: Filter files by permissions (in octal format)
-t <timestamp>: Filter files by last modified timestamp after the specified date in (YYYY-MM-DD) format
-r: Generate a summary report in a separate file
```

```
ibrahemjr@ubuntu:~$ ./script.sh /home/ibrahemjr/test_directory
Error: Invalid number of arguments.

Usage: script.sh [extension1 extension2 ... ] [directory]
```

```
ibrahemjr@ubuntu:~$ ./script.sh /home/ibrahemjr/test_directory
Error: Invalid number of arguments.

Usage: script.sh [extension1 extension2 ... ] [directory]
```

Here we see the case where the user didn't specify arguments so the script will provide him with the help section.

```
ibrahemjr@ubuntu:~$ ./script.sh -s txt png /home/ibrahemjr/test_directory
Searching for files with extensions 'png' in '/home/ibrahemjr/test_directory'...
find: invalid -size type 't'
```

Here we see that the user did not fulfill the option arguments so it would give him an error also.


```

ibrahemjr@ubuntu:~$ ./script.sh txt png
Error: make sure to enter a valid directory.

Usage: script.sh [extension1 extension2 ... ] [directory]

This script searches for files with the specified extensions (e.g., .txt) in the given directory and its subdirectories.
then it generates a report with file details grouped by owner and sorted (ascending) by total size.

Usage with file filters: script.sh [options] [extension1 extension2 ... ] [directory]

Options:
-h: Display this help section
-s <size>: Filter files by size (in bytes)
-p <permissions>: Filter files by permissions (in octal format)
-t <timestamp>: Filter files by last modified timestamp after the specified date in (YYYY-MM-DD) format
-r: Generate a summary report in a separate file

ibrahemjr@ubuntu:~$ ./script.sh txt png /home/ibrahemjr/doesntExist
Error: make sure to enter a valid directory.

Usage: script.sh [extension1 extension2 ... ] [directory]

```

Here we see the case where the user didn't specify or used an invalid directory so the script will provide him with an error and the help section.

-Reflection

I faced some difficulties in implementing my solution in the shell script syntax, maybe in parsing arguments and getting inputs from the user, it was interesting to search for commands and their usage so I could use them effectively in my approach, one problem was while trying to use multiple file extensions so I had to use an array and loop throw each extension one by one while searching for files using the (find command) trying not to use a nested loop, it was difficult to make it work 100% taking in consideration all the filters that may be specified simultaneously and to get all the file stats for all files of each extension all at once.

If I would improve this script, maybe I would add options or parameters to customize the format, maybe csv or json and content of the generated report or maybe Include more file attributes in the report, and Implement more sorting options for the user to use.