CS 332/532 – 1G- Systems Programming

HW 2

Deadline: 10/06/2024 Sunday 11:59pm

Objectives

To implement a **traverse** program in C using system calls for file and directory manipulation with additional file filtering and sorting functionalities.

Description

A program similar to **find** is commonly used in UNIX systems to explore the file system. In this homework, you will implement a program called **traverse** with extended functionality for sorting and filtering file listings. Your program will begin file traversal from a given directory and support various options for file size filtering, pattern matching, and sorting based on file size. Your program should support the following functionality:

Basic Directory Traversal

- The program should take the directory name from where to start file traversal as a command-line argument and list all files in the directory hierarchy.
- If the program is executed without any arguments, it should start from the current working directory.
- If the program encounters other directories, it should list the directory name first, followed by the files with one-tab indentation for each level of depth.

Symbolic Links

- If a file is a symbolic link, the program should display the symbolic link name and, in parentheses, the file that the link points to.

Command-Line Options

The program should support the following options:

- **-S**
  List all files along with their sizes in parentheses next to their names.

- **-s <file size in bytes>**
  List all files with size greater than or equal to the value provided.

- **-f <string pattern>**
  List all files or directories that contain the substring in their names.

- **-r**
  Sort the output by file size in reverse order (largest to smallest).

Multiple Option Combinations

The program should support combinations of the above options, such as:

-S -s 1024

-f .jpg -S

-S -r

-s 2048 -f homework

*** The order of options should not matter, and all criteria should be applied together.

Graduate Student Requirement

Graduate students should implement an additional option:

- **-t f**
  List regular files only.

- **-t d**
  List directories only.

Guidelines and Hints

1. You can use function pointers similar to Figure 4.22 in the text book to implement the functionality described above. You can use the logic and structure from Figure 4.22 as the starting point to implement this program (make sure to go over the program in Figure 4.22 and understand all the steps performed). However, please note that your final program must compile and execute without any dependencies on the source code provided by the textbook. You can find a simple example on how to use function pointers in the `funcptr.c` file
2. You can use the *`getopt`* function to process the command-line options. See *`man 3 getopt`* for more details and an example on how to use *getopt* function.
3. You should use a **Makefile** to compile and build this project and make sure to submit the Makefile along with the rest of the source code.
4. You should upload all the source code, Makefile, and a README.txt file to Canvas. Please do not upload any object files or executable files.

Program Documentation and Testing

1. Use appropriate names for variables and functions.
2. Use a Makefile to compile your program.
3. Include meaningful comments to indicate various operations performed by the program.
4. Programs must include the following header information within comments:

```
/*
Name:
BlazerId:
Project #:
To compile: <instructions for compiling the program>
To run: <instructions to run the program>
*/
```

5. Test your program with the sample test cases provided as well as your own test cases.
6. You can include any comments you may have about testing in the README.txt file.

Examples

`./traverse`	List all files in the current directory
`./traverse ../files`	List all files in the directory `../files`
`./traverse -S ../files`	List all files with their sizes in `../files`.
`./traverse -s 1024`	List files with size ≥ 1024 bytes in the current directory
`./traverse -f .jpg`	List files with `.jpg` in their names.
`./traverse -S -r`	List all files sorted by size in reverse order.
`./traverse -f report -s 2048`	List files containing "report" with size ≥ 2048 bytes

Sample Input and Output:

Assume the following directory structure:

```
projects:
  file1.c
  file2.c
  dir1:
    file3.txt
  dir2:
    file4.png
```

Then Output for ./traverse -S:

```
projects:
  file1.c (512)
  file2.c (1024)
  dir1:
    file3.txt (256)
  dir2:
    file4.png (4096)
```

It is not necessary that the order of the files are exactly as shown above, but the overall structure should look similar to the output shown above. You can use the following tar file to create the directory structure: `projects.tar`. Download this file and extract the file using the command:

```
$ tar xvf projects.tar
```

Submission Guidelines

• Use best software engineering practices to design and implement this homework.

• Use functions and organize these functions in multiple files if necessary.

• Use a Makefile to compile and build the multiple files.

• Document you code and include instructions for compiling and executing the program in the README.txt file.

• Test your program and describe how you tested this program in the README.txt file.

• To submit this homework :

- Upload all the source files and documentation to Canvas.(Source code, Makefile, and README.txt)
- **Do not forget Independent Completion Form**

CS332 Grading Rubrics

Description	Points
Basic directory traversal and file listing	50
Correct implementation of `-S` and `-r` options	10
Correct implementation of `-s` option	10
Correct implementation of `-f` option	10
Correct combination of multiple options	10
Documentation and Makefile	10

CS532 Grading Rubrics

Description	Points
Basic directory traversal and file listing	30
Correct implementation of `-S` and `-r` options	10
Correct implementation of `-s` option	10
Correct implementation of `-f` option	10
Correct combination of multiple options	10
Documentation and Makefile	10
Correct implementation of `-t f` option	10
Correct implementation of `-t d` option	10