CS 332/532 – 1G- Systems Programming

HW 3

Deadline: 10/23/2022 Sunday 11:59pm

Objectives

To add additional features to the *search* program implemented in Project #2.

Description

The goal of this project is to add additional functionality to the search program implemented in Project #2. In addition to the functionality described in Project #2, the program must support the following command-line options:

1. -e "<unix-command with arguments>"
   For each file that matches the search criteria the UNIX command specified with arguments must be executed.
2. -E "<unix-command with arguments>" [**Graduate Students Only**]
   The list of files that matches the search criteria must be provided as an argument to the UNIX command specified.

Note that with the -e option the UNIX command is executed for each file whereas with the -E option the UNIX command is executed only once but uses all the file names as the argument. You must use fork/exec/wait to create a new process to execute the UNIX command.

The UNIX command and any optional arguments are enclosed within double quotes. The program should support -e or -E options in combination with -s and -f options. You can assume that the -e or -E options appear after the -s and -f options.

Guidelines and Hints

1. 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.
2. 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>
*/
```

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

Examples

Command	***Description***
$./search -s 1024 -e "ls -l"	List all files with size <= 1024 bytes in the current directory, and execute the command "ls -l" on each file (ignore directories)
$./search -f jpg 3 -E "tar cvf jpg.tar"	List all files that have the substring “jpg” in their filename or directory name with depth <=3 relative to the current directory, and creates a tar file named jpg.tar that contains these files
$./search -s 1024 -f jpg 3 -e "wc -l"	List all files that have the substring “jpg” in their filename with depth <=3 relative to the current directory and size <= 1024, and execute the command "wc -l" on each file (ignore directories)

Sample Input and Output:

If you have the following directory structure as shown by the output of "ls -lR" command:

```
$ ls -R projects
projects:
fread.c fwrite.c project1 project2 project3 project4 read.c write.c

projects/project1:
project1.docx README

projects/project2:
project2.docx README

projects/project3:
project3.docx README

projects/project4:
project4.docx README
```

Then the output of find without any argument should look like this:

```
projects
        fread.c
        fwrite.c
        project1
                README
                project1.docx
        project2
                project2.docx
                README
        project3
                project3.docx
                README
        project4
                project4.docx
                README
        read.c
        write.c
```

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

Upload the C source code, PDF version of C source code, and README.txt file to Canvas. Do not include any executable or object files.

Use the following command to create PDFs of your source code.(replace the <Source_code_File_name> and <Output_Source_code_File_name> with your C source code file name):

```
$enscript <Source_code_File_Name>.c -o - | ps2pdf - <Output_Source_code_File_Name>.pdf
```

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

• 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.

Grading Rubrics

Description	**Points**
Implementation of parsing the arguments to the search program correctly and invoking the appropriate function	20 points
Implementation of *search* that executes the UNIX command with optional arguments for each matching file using fork/exec/wait	50 points
Implementation of *search* that lists the files and directories in the specified format when executed with multiple options (combination of -s, -f and -e)	20 points
Use of Makefile, source code documentation (including README.txt file)	10 points