CS 332/532 – 1G- Systems Programming HW 3

Objectives

1. To add additional features to the search program implemented in Homework #2.

Assignment Submission

The assignment submission in Canvas is mandatory. No late submissions or extensions will be accepted.

Submission Checklist:

- Upload a C source file (.c file) to Canvas as part of this assignment submission. Submissions through the Canvas "Comments" will not be accepted. The file should be named in this naming convention: yourblazerid_HW03.c
- Upload a README.md file which should include:
    - Instructions on how to compile your C source file into an executable.
    - How to run the executable program.
    - Any citation documentation.
    - A link to your GitHub repository.
- Upload a Make.mk file which should include:
    - The build instructions for your program
- Upload an Independent Completion Form.

Please do not upload executables or object files.

Rubric

Criteria	332 & 532
Uploaded Independent Completion Form	5 pts
Uploaded .c file Uploaded .c file of code submission.	5 pts
Uploaded .md readme file and .mk MAKE file. Uploaded .md file of code description, usage information, and any coding citations. Uploaded .mk file of code compilation instructions.	5 pts
Provided working (shared) GitHub repository link Provided working (shared) GitHub repository link in the README file.	5 pts
“Basic Functionality” Implementation of parsing the arguments to the search program correctly and invoking the appropriate function Compiles & Does Not Error	30 pts
“Using fork/exec/wait” Implementation of search that executes the UNIX command with optional arguments for each matching file using fork/exec/wait Compiles & Does Not Error	25 pts
“Works with HW2 Functionality” Implementation of search that lists the files and directories in the specified format when executed with multiple options (combination of -L, -s and -e/E) Compiles & Does Not Error	25 pts

Your code needs to be compiled on GitHub codespace, or you need to demo your code to TA.

HW Assignment #3

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:

- **[Undergraduate & Graduate Students]**
    - -e "<unix-command with arguments>"
        - For each file that matches the search criteria the UNIX command specified with arguments must be executed.
- **[Graduate Students Only]**
    - -E "<unix-command with arguments>"
        - 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 arguments. 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 -L and -s options. You can assume that the -e or -E options appear after the -L and -s options.

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

Command	***Description***
$./search -L 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 -s 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
$./serch -L 1024 -s 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
```

It is not necessary that the order of the files be exactly as shown above, but the overall structure should look like 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
```