Assignment #5

Due: March 6, 2017 11:59pm

Overview

This assignment requires interaction with the filesystem (via system calls), traversing a portion of the directory hierarchy, and managing scarce resources. For this assignment you will write a program that acts like the Unix `find` utility. This utility recursively searches a directory hierarchy for files. Once found, the utility can take a number of actions. You will write a greatly simplified version[1].

sfind

Your program, named `sfind`, will be executed as follows (note that the brackets are used to indicate an optional argument, and the `|` is used to denote a choice between two flags):

```
sfind filename [-name str] -print | -exec cmd ;
```

The `sfind` program will recursively process all files in the hierarchy starting with `filename` (if this name is not a directory, then no recursion will take place; files are to be processed in lexicographic order as determined by either `strcmp` or `strcasecmp`). For each processed file, the specified action (`-print` or `-exec`) will be performed.

If the `-name str` switch is present, then only those files with `str` as a *substring* will be processed. Directories are processed in the same manner as files, but they are also always examined as part of the recursive descent (i.e., searched) even if they do not contain the specified substring.

If the `-print` action is specified, then the name (the complete pathname based on the starting directory) of the processed file is printed to `stdout`. For example, the following command will list all files starting with the current directory.

```
% sfind . -print
```

If the `-exec` action is specified, then another process must be created to execute the specified command. The command `cmd` begins immediately following the `-exec` switch and continues until the `;`[2]. Within the `cmd`, any occurrence of {} (without spaces between) must be replaced by the processed filename (the complete pathname based on the starting directory) prior to execution. For example, the following command will act much like the command above, but will instead operate by executing `echo` with the name of each processed file.

```
% sfind . -exec echo {} \;
```

Note: When processing a directory's contents, do not recursively follow the two special files "." and ".." and do not follow symbolic links.

Implementation Requirements

You must write your own directory traversal code using system calls like `opendir` and `readdir`. You may not use any library routine to do the traversal for you (e.g., no use of `scandir` or the fts functions)[3]. You must also write your own process creation and program execution code. You may *not* use the `system` library routine in your solution.

Resource Management

All programs must properly manage the resources they use. In this respect, systems programs differ only in that the resources they must manage are often more scarce than those used by general applications. To address this issue, one will often use more readily available resources (e.g., memory) to avoid exhausting other resources.

If you are not careful in your implementation, then your program will likely run out of file descriptors and be unable to continue. Do not allow this to happen. You must also prevent your program from hitting the process limit.

---

[1] Take careful note of the fact that this is a simplified. You do not need to try to match all functionality of the existing `find` tool.

[2] Note that when running your program, you will need to escape the `;` because most shells will process this as delimiting a sequence of commands to execute. To escape the `;`, use `\;`.

[3] Apply what you have learned here. Then use library functions in later projects.

Submit

- All source code.
- A Makefile that will build your program.
- A **README** file that specifies anything about your program that you feel I should know during grading.

Grading

Feature	Percentage
Final Decomposition	10
Basic Functionality (including print)	25
exec	40
Resource Management (memory, file descriptors, processes)	15
Error Handling	10