myfind

Work in groups of 2 and write a program in C/C++ that enables a user to parallelly find different files in a folder without the usage of the linux command “find” as a child process (or somehow else).

Usage:

/myfind [-R] [-i] searchpath filename1 [filename2] …[filenameN]

The main program accepts the arguments <searchpath> and <filename1 .. N>. Keep in mind that a variable number of arguments (= variable number of filenames to look for) can be used and that the options -R and -i can be placed anywhere in the arguments list.

- -R:
  - should switch myfind in recursive mode and find **all** matching files in and below the searchpath folder
    (else the files should only be searched in the searchpath folder)
- -i
  - case in-sensitive search
- searchpath
  - can be an absolute or a relative path.
- filename
  - only filenames as plain string
  - no support for paths, subpaths, wildcards required.

Example:

./myfind ./ test.txt test.doc test

Here, myfind searches for the 3 files in the current working directory in parallel.

The main program fork()s a child-process for each filename and looks for the file in the defined searchpath. In case a file is found, an entry in the following output format will be printed to stdout (unsorted; but readable in full lines). Describe in the code how you achieved this requirement (in comment above main).

Output-Format:

<pid>: <filename>: <complete-path-to-found-file>\n

- <pid>: process-id of the child-process that finds the entry.
- <filename>: references the filename that was passed into the main program as an argument.
- <complete-path-to-found-file>: absolute path to found file

Hints:

- The parent-process must react on the termination of child-processes (prevent a zombie apocalypse).
- Use getopt() for argument parsing.
- Check the code-samples in moodle if you are not familiar with file system functions in C, but you can also use the C++17 File System library, see for example https://www.geeksforgeeks.org/file-system-library-in-cpp-17/
- Take care of the synchronization of the output of different child processed to stdout.
- Code quality and compliance to the principles of C-programming is part of the grading.
- Comment, structure and indent your code properly.

Deliverables

- Hand-in
  - The commented code of the myfind project
  - Makefile for the targets “all” and “clean”
  - Executables
  - A short protocol (1 page; pdf) describing the key parallelization concepts and needed synchronization of the output on stdout

Marking System (25 points)

- 9: concurrency and synchronized output
- 3: argument management with proper handling of multiple files
- 3: recursive find
- 2: case-insensitive use
- 3: Makefile, structure, error-handling, code quality, indentation, comments
- 5: protocol