March 19th, 2021

CSE344 – System Programming - Homework #1 - v3

Objective

You are expected to write an “advanced” file search program for POSIX compatible operating systems. Your program must be able to search for files satisfying the given criteria, and print out the results in the form of a nicely formatted tree.

How

The search criteria can be **any combination** of the following (at least one of them must be employed):

- `-f` : filename (case insensitive), supporting the following regular expression: +
- `-b` : file size (in bytes)
- `-t` : file type (d: directory, s: socket, b: block device, c: character device f: regular file, p: pipe, l: symbolic link)
- `-p` : permissions, as 9 characters (e.g. ‘rwxr-xr--’)
- `-l`: number of links

It seems that I have overestimated you. Some of you have managed to arrive to the 6th semester without knowing about regular expressions. The symbol + means one or more of the character preceding it. **You are to implement it on your own. Don’t use a library for regexp parsing**. + is the only regexp operator to support, at its simplest form; no braces or brackets (i.e. { or } or [or]) just one or more of the preceding character. Check the output example for more details. Ask questions at the forum if it’s still unclear.

Yes, this means you need to support multiple occurrences of + as regexp; e.g. los+t+file that matches losssssttttfile for instance.

In case the user wants to search for a filename containing the ASCII + character, then the user must escape that character with \; e.g. lost\+file will match lost+file

The program will additionally receive as mandatory parameter:

- `-w`: the path in which to search **recursively** (i.e. across all of its subtrees)

You can use the `getopt()` library method for parsing command-line arguments.

Example

```
./myFind -w targetDirectoryPath -f ‘lost+file‘ -b 100 -t b
```

means it will search in the `targetDirectoryPath` path, for block device files named `lost+file` of size exactly 100 bytes

Output

In the form of nicely formatted tree

```
targetDirectoryPath
|--subDirectory
|----subDirectory2
|------subDirectory3
|--------LOSttttttFile
|--subDirectory6
|----lOstFile
```

If no file satisfying the search criteria has been found, then simple output “No file found”. All error messages are to be printed to stderr. All system calls are to be checked for failure and the user is to be notified accordingly.

Evaluation
You program will be evaluated with searches on our local boxes. This is a simple, all or nothing homework, i.e. strive to satisfy all the requirements. Make sure you have tested and ensured all search criteria are correctly implemented.

Rules:
1) Compilation error: grade set to 1; if the error is resolved during the demo, then evaluation continues.
2) Compilation warning (with respect to the **-Wall** flag); -1 for every warning until 10. -20 points if there are more than 10 warnings; no chance of correction at demo.
3) No makefile: -20
4) No pdf (other formats are inadmissible) report submitted (or submitted but insufficient, e.g. 3 lines of text or no design explanation, etc): -20.
5) A report prepared via latex (pdf and tex source submitted together): +5
6) If the required command line arguments are missing/invalid, your program must print usage information and exit. Otherwise: -10
8) The program crashes and/or doesn’t produce expected output with normal input: -100
9) Presence of memory leak (regardless of amount – checked with valgrind) -30
10) Zombie process (regardless of number) -30
11) Deadlock of any kind due to poor synchronization -50
12) Use of poor synchronization: busy waiting/trylock/trywait/timedwait/sleep or any other form other than those specified at homework: -60
13) Late submissions will not be accepted
14) In case of an arbitrary error, exit by printing to stderr a nicely formatted informative message. Otherwise: -10
15) In case of CTRL-C the program must stop execution, return all resources to the system and exit with an information message.
16) Use system calls for all file I/O purposes, don’t use standard C library functions.

Is my homework submission valid?
If it traverses recursively the entire given subtree (regardless of whether it finds the sought files or not), then yes; if not, then no it will not be considered a valid submission.

Submission rules:
- Your source files, your makefile and a report; place them all in a directory with your student number as its name, and zip the directory.
- Your report must contain: how you solved this problem, your design decisions, which requirements you achieved and which you have failed.
- The report **must be in English**.
- Your makefile must only compile the program, not run it!
- Do not submit any binary executable files. The TAs will compile them on their own boxes.
- Any deviation from the submission rules will results in automatic (without content evaluation) failure of the homework.

Hints:
- Plan/design carefully.
- Check for memory leaks with valgrind before submitting.

- Compile and run your program on more than one POSIX system to ensure portability, if you can.
- Control whether you satisfy each and every one of the requirements prior to submission.

Good luck.