HW-1 / REPORT

Can Duyar - 171044075

Project Summary

MyFind is a POSIX compatible file searching software which parses command line options, searches files with those options and displays needed file path with a nicely formatted tree. The program uses system calls to open a directory and recursively search for files according to the given criteria. If no file satisfying the search criteria has been found, it will display "No file found". All error messages are printed to stderr. All system calls are checked for failure and the user is to be notified accordingly.

Build

Compilation is possible with Makefile. To build the project run the command "make" in the project directory and executable file myFind will be created at the same directory. The GCC command line argument -Wall is used to exclude flaws in the code and any kind of warning during compilation.

Usage

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

- -f: filename (case insensitive)
- -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

The program additionally receives mandatory parameter:
- -w: the path in which to search recursively

The usage information can be checked by running program without any command line option. Besides the program will also write usage information when command line options are invalid or missing.

-How did I solve this problem? + Design Decisions

At first command line arguments are parsed to detect search criteria. Program uses getopt() function to parse command line options and initialize search criteria. If the options are invalid/missing, program will display informational message, usage information and exit.

myFind is able to search files with case insensitive form, to achieve this requirement program is converting file names into lowercase before comparing them to each other. To support the regex operator program is also waiting for + symbol during the name comparisons in the file name and handling it accordingly to support the mentioned regular expression.

Signal handler is used to achieve the requirement about CTRL+C handling, program catches this signal, prints informative message end finishes work. The function sigaction() is used to register signal handler.

The system call functions opendir() and readdir() are used to open and iterate the target directory during search. Search function is recursive and it will follow every sub directory loaded with readdir() function. After finishing the work with current directory, all occupied resources will be cleaned up with closedir() function.

If the file matches with given criteria program will display the full path of the file with a nicely formatted tree. The last printed directory path is used to compare and continue path drawing from last matching directory. Using the last directory path program also calculates last matching directory depth, later this information is used to calculate indentation and print for sub directory.

I checked number of Zombie processes with "top" command on terminal after program runs. There is no Zombie process after my program runs.

I also checked memory leak with valgrind and there is no memory leak after program runs. I used that command on terminal to control memory leaks:

valgrind ./myFind -w targetDirectoryPath .....

I compiled and run my program on more than one POSIX system to ensure portability.

-Are the requirements done?

All requirements are done and program repeatedly tested.Everything works as it should.