CSE344 – System Programming
HW#1 – Report

Berk Pekgöz
171044041

Solving The Problem & Design:

To solve our problem, I divided problem into four parts. First part is parsing the command line arguments and getting search criterias. Second part is checking if a file is the file we looking for. Third part is creating a general tree structure which keeps the directory tree. Fourth and final part is traversing directories and combining past three parts in one place.

First Part: To parse command line arguments getopt() function is used. Rules for each criteria is checked. To keep given criterias a struct named file_opts is created. parse_command_line_opt() function initialize this structure with default values(NULL/-1) then fills with values from optarg (checked for valid values). If user enters invalid input for a criteria (e.g. string for number of links), error occurs and information message will be on stderr. For type of file get_mode_of_opt_type() function created. This file takes input letter for file type and returns related mode_t type value. In the end function checks three things: If there are extra arguments, then error occurs. If there is no targetDirectoryPath given, then error occurs. If user dont give at least one criteria for search, then error occurs.

Second Part: To get information of a file, this function (is_searched_file()) gets stat structure of file as input. Then compares with the information in the file_opt structure which filled in first part.
To achive required regular expression (+) cmpstr_plus() function created. This function compares each char of string but if there is '+' char then another loop goes through same char until different char comes.
File type is already filled in file_opt struct with mode_t value in first part so in this part just compare with other mode_t value.
For permissions, perm string of file is created with get_perm_string_of_file() function then compared with the string which is given by user.

Third Part: To keep directories, a general tree structure is needed. So, i created a node struct named dir_node. And some functions to manage tree are written(simple tree structure with dynamic array). All nodes are initialized with 0 sub-node capacity. At first add capacity becomes 1. Later additions makes capacity doubled. I designed with this way with knowledge from Data Structures course.

Fourth part: traverse_directories() function works on this part. Opendir() function (POSIX Supported) used to open given directory with given path. Also file_opt strucutre (first part) and parent node given as parameters. This function can reach every subfile of given directory with readdir() function (POSIX Supported). Newpath created for every sub-file and lstat function called for every sub-file. Then calls is_searched_file() function (second part). After that if sub-file is a directory, makes recursive call. After the recursive call returns

checks for the result of both is_searched_file() and recursive call. If one of them returns 1 (which means a file founded) then adds current node to parent node.
If there is an error or recursive call returns –1 (which means an error occured) function free all resources and returns as –1.

If interrupt signal is happened then function start to free all resources and returns to main function.

All errors are handled by checking and error messages sent to stderr.

Achieved Requirements:

(All requirements given in pdf file has achieved.)

- 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.

```
berk@berk-VirtualBox:~/Desktop/System$ ./hw1 -w /home/berk/Desktop/Test -b 4096
/home/berk/Desktop/Test
|--tesst
|----tessst
|--tessssssssssssst
|--tessst
|----temp
|------test
```

- The search criteria can be any combination.
- At least one of criteria must be employed.

```
berk@berk-VirtualBox:~/Desktop/System$ ./hw1 -w /home/berk/Desktop/Test
Please enter at least one criteria for correct usage.
```

- filename (case insensitive), supporting the following regular expression: +.

```
berk@berk-VirtualBox:~/Desktop/System$ ./hw1 -w /home/berk/Desktop/Test -f tES+t
/home/berk/Desktop/Test
|--tesst
|----test
|----tessst
|------test
|--tessssssst
|--tessssssssssssst
|--test
|--tessst
|----temp
|------tesssssssssst
|------test
|--------tesst
```

- The program will additionally receive as mandatory parameter -w: the path in which to search recursively.

```
berk@berk-VirtualBox:~/Desktop/System$ ./hw1 -f tES+t
Please enter path(-w) for correct usage.
```

- If no file satisfying the search criteria has been found, then simple output “No file found”.

```
berk@berk-VirtualBox:~/Desktop/System$ ./hw1 -w /home/berk/Desktop/Test -f dnm
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.

```
berk@berk-VirtualBox:~/Desktop/System$ ./hw1 -w /home/berk/Desktop/Test2 -f dnm
ERRNO: 2 - No such file or directory
```

- If the required command line arguments are missing/invalid, your program must print usage information and exit.

```
berk@berk-VirtualBox:~/Desktop/System$ ./hw1 -w /home/berk/Desktop/Test -f
./hw1: option requires an argument -- 'f'
berk@berk-VirtualBox:~/Desktop/System$ ./hw1 -w /home/berk/Desktop/Test -f dnm d
ummy
Unknown arguments given. Please read manual for correct usage.
```

- In case of CTRL-C the program must stop execution, return all resources to the system and exit with an information message. (To test this requirement i used sleep function)

```
berk@berk-VirtualBox:~/Desktop/System$ ./hw1 -w /home/berk/Desktop/Test -f tes+t
^CExecution interrupted by user!
```