March 24, 2021

# GTU Department of Computer Engineering
# CSE344 - Spring 2021
# Homework 1 Report

**Akif Kartal**
**171044098**

# 1 Problem Definition

The problem is to write an "advanced" file search program for POSIX compatible operating systems.

# 2 Solution

In order to write this program we need to **divide problem into modules**. These modules are following;

## 2.1 Argument Handling

In order to manage arguments in a proper way I used **argument struct** to keep given argument data as one block data. My argument struct is following.

```
typedef struct st
{
    int wFlag;
    int fFlag;
    int bFlag;
    int tFlag;
    int pFlag;
    int lFlag;
    char *wArg;
    char *fArg;
    char *bArg;
    char *tArg;
    char *pArg;
    char *lArg;
    int isFound;
    int count;

} args;
```

In this struct **count** represent number of optional arguments and **isFound** represent file is found or not. Also, in order to get argument's value **getopt()** library method was used.

## 2.2 Error Handling

In order to handle any error **stderr** and **exit** system call was used. Following code shows an example of this;

```
void showUsageAndExit()
{
    // All error messages are to be printed to stderr.
    fprintf(stderr, "Usage: ./myFind [FLAGS] and [PARAMETERS]\n"
                    "Optional Flags: in any combinations(at least 1)\n"
                    "-f : filename (case insensitive), supporting the following regular
    expression: + \n"
                    "-b : file size (in bytes) \n"
                    "-t : file type (d: directory, s: socket, b: block device, c: character
    device f: regular file, p: pipe, l: symbolic link) \n"
                    "-p : permissions, as 9 characters (e.g. 'rwxr-xr--') -l: number of
    links \n"
                    "Mandotory Flags:\n"
                    "-w: the path in which to search recursively (i.e. across all of its
    subtrees)\n"
                    "Example\n"
                    "./myFind -w targetDirectoryPath -f 'lost+file' -b 100 -t b\n");
    exit(EXIT_FAILURE);
}
```

## 2.3 Advanced Search Algorithm

### 2.3.1 Regex Handling

The filename argument can contain more than one "+" character. To handle this situation in an easy way somehow we need to keep position data of these regexs. In order to do this I created my own **LinkedList** data structure to keep both position and previous letter information. **My LinkedList** node is following;

```
//regex information node
typedef struct node_s
{
    int position;
    char preChr;
    struct node_s *next;

} node_t;
```

### 2.3.2 Checking Given Parameters

- File name check was made by using regex linkedlist.
- File size, file type, file permissions and file links was checked by using **stat calls**.

### 2.3.3 Recursive Search Algorithm

TBO.

### 2.3.4 Drawing nicely formatted tree

If the searching file is found then directory tree will be drawn by using same recursive search algorithm with just a **minor** difference.

## 2.4 CTRL-C Handling

In order to give a message on CTRL-C interrupt, I used **signal** function from **signal.h** library.