```
#include <stdio.h>
#include <stdlib.h>
#include <sys/stat.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <unistd.h>
#include <dirent.h>
#include <string.h>

struct options
{
	// -f
	char* substring;
	// -s
	int min_size;
	// -e
	int e_command;
	// -E
	int E_command;
	// command
	char* command;
	// options
	char** command_options;
};

void get_options_from_command(char* command, struct options* options);
void execute_command(char* path, char* command, char** options);
void recurse_print_dir(char* path, int count);
```

```
struct options global_options;



char** all_files;
int file_count = 0;



void execute_command(char* file_path, char* command, char** options)
{
	int pid = fork();
	if (pid < 0)
	{
		printf("fork error \n");
		exit(-1);
	}
	else if (pid == 0)
	{
		char* arr[32];
		arr[0] = command;
		int arr_count = 1;
		//create array of options
		for (int o = 0; options[o] != NULL; arr_count++, o++)
		{
			arr[arr_count] = options[o];
		}
		arr[arr_count++] = file_path;
		arr[arr_count] = NULL;
```

```
                //execute command on file
                execvp(command, arr);
        }
        else
        {
                wait(NULL);
        }
}

//get the options from command
void get_options_from_command(char* command, struct options* options)
{
        options->command_options = malloc(20 * sizeof(char*));
        char* temp = strtok(command, " ");
        int i = 0;
        if (temp == NULL)
        {
                printf("No command!\n");
                exit(-1);
        }
        else
        {
                options->command = temp;
                temp = strtok(NULL, " ");

                while (temp != NULL)
                {
                        options->command_options[i] = temp;
                        temp = strtok(NULL, " ");
```

```
                i++;
            }
        }
        options->command_options[i] = NULL;
}

// execute command on all files
void execute_command_on_all_files()
{
        int pid = fork();
        if (pid < 0)
        {
            printf("fork error\n");
            exit(-1);
        }
        else if (pid == 0)
        {
            char* arr[4000];
            arr[0] = global_options.command;
            int arr_count = 1;

            //create array of option
            for (int o = 0; global_options.command_options[o] != NULL; arr_count++, o++)
            {
                arr[arr_count] = global_options.command_options[o];
            }

            // add all_files to array
            for (int k = 0; k < file_count; k++)
```

```
                {
                        arr[arr_count++] = all_files[k];
                }

                arr[arr_count] = NULL;

                execvp(global_options.command, arr);
        }
        else
        {
                wait(NULL);
        }
}

// traverse the directories and print
void recurse_print_dir(char* path, int count) {
        DIR* dir = opendir(path);

        if (dir == NULL)
        {
                printf("Error opening directory '%s'\n", path);
                free(global_options.command_options);
                exit(-1);
        }
        int k = 0;
        while (k++ < count)
                printf("\t");
        printf("%s\n", path);
```

```
	struct dirent* dirent_s;
	char curr_path[1000];

	//get files/directories from dir
	while ((dirent_s = readdir(dir)) != NULL)
	{
		if (strcmp(dirent_s->d_name, ".") == 0 || strcmp(dirent_s->d_name, "..") == 0)
			continue;

		strcpy(curr_path, path);
		strcat(curr_path, "/");
		strcat(curr_path, dirent_s->d_name);

		if (dirent_s->d_type == DT_DIR)
		{
			recurse_print_dir(curr_path, count + 1);
			continue;
		}
		// -f option
		if (global_options.substring != NULL && strstr(dirent_s->d_name,
global_options.substring) == NULL)
			continue;

		// -s option
		if (global_options.min_size)
		{
			struct stat st;
			stat(curr_path, &st);
			if (st.st_size <= global_options.min_size)
```

```
                        continue;
                }

                k = 0;
                while (k++ <= count)
                        printf("\t");
                printf("->%s\n", dirent_s->d_name);

                if (global_options.e_command)
                        // execute command on file
                        execute_command(curr_path, global_options.command,
global_options.command_options);

                else if (global_options.E_command)
                {
                        all_files[file_count] = malloc(strlen(curr_path) + 1);
                        strcpy(all_files[file_count], curr_path);
                        all_files[file_count][strlen(curr_path)] = '\0';
                        file_count++;
                }
        }

        closedir(dir);
}


int main(int argc, char* argv[]) {
        char* first_dir = malloc(256);
        char* command_string;
```

```
strcpy(first_dir, ".\0");

global_options.min_size = 0;
global_options.e_command = 0;
global_options.E_command = 0;
global_options.substring = NULL;

// get options from commandline
for (int i = 1; i < argc; i++)
{
	if (!strcmp(argv[i], "-s"))
	{
		i++;
		if (i < argc)
		{
			global_options.min_size = atoi(argv[i]);
		}
		else
		{
			printf("valid argument not passed for option [-f]\n");
		}
	}
	else if (!strcmp(argv[i], "-f"))
	{
		i++;
		if (i < argc)
		{
			global_options.substring = malloc(strlen(argv[i]) + 1);
			strcpy(global_options.substring, argv[i]);
```

```
                global_options.substring[strlen(argv[i])] = '\0';
        }
        else
        {
                printf("no arg for option -f\n");
        }
}
else if (!strcmp(argv[i], "-e"))
{
        i++;
        if (i < argc)
        {
                global_options.e_command = 1;
                get_options_from_command(argv[i], &global_options);
                command_string = argv[i];
        }
        else
        {
                printf("No arg for option -e \n");
                return -1;
        }
}
else if (!strcmp(argv[i], "-E"))
{
        i++;
        if (i < argc)
        {
                all_files = malloc(4000 * sizeof(char*));
                global_options.E_command = 1;
```

```
                        get_options_from_command(argv[i], &global_options);
                        command_string = argv[i];
                }
                else
                {
                        printf("No arg for option -E \n");
                        return -1;
                }
        }
        else
        {
                strcpy(first_dir, argv[i]);
                first_dir[strlen(argv[i])] = '\0';
        }

}

// traverse from directory and print recursively
recurse_print_dir(first_dir, 0);

// E command
if (global_options.E_command)
{
        printf("files on which command will be executed !: \n");
        for (int k = 0; k < file_count; k++)
                printf("\t %s\n", all_files[k]);
        printf("Executing command [%s] on files !\n", command_string);
        execute_command_on_all_files();
}
```

```
        free(first_dir);
        free(global_options.command_options);
        return 0;
}
```