```c
/*
Name: John Robertson
BlazerId: jprob
Project #: 3
To compile: gcc p3.c
To run: ./a.out [directory name] [flags] #see README.md for details on flags
*/
#include <stdio.h>
#include <stdlib.h>
#include <dirent.h>
#include <unistd.h>
#include <sys/stat.h>
#include <string.h>
#include <limits.h>
#include <sys/wait.h>

char* s_arg = NULL;
char* f_arg = NULL;
char* e_arg = NULL;
int s_arg_set = 0;
int f_arg_set = 0;
int e_arg_set = 0;
int dash_S = 0;
void smallS(char*, char*, int);
void smallF(char*, char*, int);
void smallE(char*, char*, int);
void capS(char*, char*, int);

char* filetype(unsigned char type) {
        char *str;
        switch(type) {
                case DT_BLK: str = "block device"; break;
                case DT_CHR: str = "character device"; break;
                case DT_DIR: str = "directory"; break;
                case DT_FIFO: str = "named pipe (FIFO)"; break;
                case DT_LNK: str = "symbolic link"; break;
                case DT_REG: str = "regular file"; break;
                case DT_SOCK: str = "UNIX domain socket"; break;
                case DT_UNKNOWN: str = "unknown file type"; break;
                default: str = "UNKNOWN";
        }
        return str;
}

int tokenize(char* buf, char** argu_ve, int extra) {
    char* tempo = (char *) malloc(strlen(buf) + 1);
    strcpy(tempo, buf);
    char* head = strtok(tempo, " ");
    int i = 1;
    argu_ve[0] = head;
    char* next;
    while ((next = strtok(NULL, " ")) != NULL) {
        argu_ve[i] = next;
        i++;
    }
    argu_ve[i + extra] = NULL;
    return i;
}

off_t filesize(char* filename) {
    //https://stackoverflow.com/a/238609/9295513
    struct stat st;
    stat(filename, &st);
    return st.st_size;
}

void print_util(char* name, int filesize_S, int indent) {
    int i;
    for (i = 0; i < indent; i++) {printf("\t");}
    if (filesize_S != -1) {
        printf("NAME: %s (SIZE: %d)", name, filesize_S);
    }
    else {
        printf("NAME: %s", name);
    }
    printf("\n");
}

void smallS(char* name, char* type, int indent) {

    if(s_arg_set) {
        int cmpsz = atoi(s_arg);
        if (filesize(name) >= cmpsz) {
            smallF(name, type, indent);
        }
    }
    else {
        smallF(name, type, indent);
    }

}

void smallF(char* name, char* type, int indent) {

    if(f_arg_set) {
        char* sub = f_arg;
        //printf("%s vs %s\n", sub, name);
        if (strstr(name, sub) != NULL) {
            smallE(name, type, indent);
        }
    }
    else {
        smallE(name, type, indent);
    }
}

void smallE(char* name, char* type, int indent) {

    if(e_arg_set) {
        capS(name, type, 0);
        int status;
        if(fork() == 0) {
            char* arguve[strlen(e_arg) + 1];

                                        if (strcmp(type, "directory") == 0) {
                                            tokenize(e_arg, arguve, 0);
                                        }
                                        else {
                                            int fin_arg = tokenize(e_arg, arguve, 1);
                                            arguve[fin_arg] = name;
                                        }

            execvp(arguve[0], &arguve[0]);
                perror("execvp");
                exit(-1);
        }
        else {
            wait(&status);
        }
    }
    else {
        capS(name, type, indent);
    }
}

void capS(char* name, char* type, int indent) {

    if (dash_S) print_util(name, filesize(name), indent);
    else print_util(name, -1, indent);
}

void filetrav(int indent, char* dname) {

    DIR* pd = opendir(dname);
    struct dirent* dnt;
        while ((dnt = readdir(pd)) != NULL) {
        smallS(dnt->d_name, filetype(dnt->d_type), indent);
                if (DT_DIR == dnt->d_type &&
                    strcmp(dnt->d_name, "..") != 0 &&
                    strcmp(dnt->d_name, ".") != 0 &&
                pd != NULL) {
            char curr_dir[PATH_MAX];
            strcpy(curr_dir, dname);
                    char* next_dir = dnt->d_name;
            strcat(curr_dir, "/");
            strcat(curr_dir, next_dir);
                    filetrav(indent + 1, curr_dir);
                }
        else {
            continue;
        }
        }
    //closedir(pd);
}

int main (int argc, char *argv[]) {

        char retval[PATH_MAX];
        getcwd(retval, sizeof(retval));

        int flag;
        while ((flag = getopt(argc, argv, "Ss:f:e:")) != -1) {
            switch (flag) {
                case 's':
                    s_arg = optarg;
                s_arg_set = 1;
                    break;
                case 'f':
                    f_arg = optarg;
                f_arg_set = 1;
                break;
            case 'e':
                e_arg = optarg;
                e_arg_set = 1;
                break;
            case 'S':
                dash_S = 1;
                    break;
            }
        }



    printf("%d vs %d\n", optind, argc);

        if (optind < argc) {
        filetrav(0, argv[optind]);
    }
    else {
        filetrav(0, retval);
    }
        return 0;
}
```