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

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;
}

void sarg_branch(const char* d_name, char* buf, char* s_arg, char* f_arg, int symlink,
int size) {
    if (s_arg == NULL) {
        if (f_arg == NULL) {
            if (symlink) {
                printf("(%s) %s\n", d_name, buf);
            }
            else {
                printf("%s\n", d_name);
            }
        }
        else {
            if (0 == strncmp(f_arg, d_name, strlen(f_arg))) {
                if (symlink) {
                    printf("(%s) %s\n", d_name, buf);
                }
                else {
                    printf("%s\n", d_name);
                }
            }
        }
    }
    else {
        int sarg = atoi(s_arg);
        if (sarg <= size) {
            if (f_arg == NULL) {
                if (symlink) {
                    printf("(%s) %s\n", d_name, buf);
                }
                else {
                    printf("%s\n", d_name);
                }
            }
            else {
                if (0 == strncmp(f_arg, d_name, strlen(f_arg))) {
                    if (symlink) {
                        printf("(%s) %s\n", d_name, buf);
                    }
                    else {
                        printf("%s\n", d_name);
                    }
                }
            }
        }
    }
}

void filetraverse(DIR* pd, int indent, char* dname, int dash_S, char* s_arg, char* f_ar
g) {
    chdir(dname);
    struct dirent* dnt;
        if (dash_S) {
        while ((dnt = readdir(pd)) != NULL) {
                int i;for (i = 0; i < indent; i++) {printf("\t");}//cruft to handle ind
entation, look away

            if (DT_LNK == dnt->d_type) {
                // https://wiki.sei.cmu.edu/confluence/display/c/POS30-C.+Use+the+readl
ink%28%29+function+properly
                // https://stackoverflow.com/questions/238603/how-can-i-get-a-files-siz
e-in-c
                char buf[255];
                ssize_t len = readlink(dname, buf, sizeof(buf));
                buf[len] = '\0';
                struct stat st;
                stat(dnt->d_name, &st);
                int size = (int) st.st_size;
                sarg_branch(dnt->d_name, buf, s_arg, f_arg, 1, size);
            }
            else if (strcmp(dnt->d_name, "..") != 0 && strcmp(dnt->d_name, ".") != 0) {
                char buf[1];
                struct stat st;
                stat(dnt->d_name, &st);
                int size = (int) st.st_size;
                        sarg_branch(dnt->d_name, buf, s_arg, f_arg, 0, size);
            }

                if (DT_DIR == dnt->d_type && strcmp(dnt->d_name, "..") != 0 && strcmp(d
nt->d_name, ".") != 0 && pd != NULL) {
                    DIR* child_dir;
                    char* fullpath = dnt->d_name;
                    child_dir = opendir(fullpath);
                    if (child_dir != NULL) {
                        filetraverse(child_dir, indent + 1, dnt->d_name, dash_S, s_arg,
 f_arg);
                        chdir("..");
                    }
                }
        }
    }
    else {
        while ((dnt = readdir(pd)) != NULL) {
                int i;for (i = 0; i < indent; i++) {printf("\t");}//cruft

            if (DT_DIR == dnt->d_type && strcmp(dnt->d_name, "..") != 0 && strcmp(dnt->
d_name, ".") != 0) {
                char buf[1];
                struct stat st;
                stat(dnt->d_name, &st);
                int size = (int) st.st_size;
                sarg_branch(dnt->d_name, buf, s_arg, f_arg, 0, size);
            }
```

```
                if (DT_DIR == dnt->d_type && strcmp(dnt->d_name, "..") != 0 && strcmp(d
nt->d_name, ".") != 0 && pd != NULL) {
                    DIR* child_dir;
                    char* fullpath = dnt->d_name;
                    child_dir = opendir(fullpath);
                    if (child_dir != NULL) {
                        filetraverse(child_dir, indent + 1, dnt->d_name, dash_S, s_arg,
 f_arg);
                        chdir("..");
                    }
                }

        }
    }
        closedir(pd);
}

int main (int argc, char **argv) {
    DIR* parentDir;
        if (argc < 2) {
                parentDir = opendir("."); // is this right?
        }
        else {
            parentDir = opendir(argv[1]);
        }
        if (parentDir == NULL) {
                printf ("Error opening directory '%s'\n", argv[1]);
                exit (-1);
        }

    int dash_S = 0;
    char* s_arg = NULL;
    char* f_arg = NULL;

        int flag;
        // https://www.gnu.org/software/libc/manual/html_node/Example-of-Getopt.html
        while ((flag = getopt(argc, argv, "Ss:f:")) != -1) {
            switch (flag) {
                case 'S':
                    dash_S = 1;
                    break;
                case 's':
                    s_arg = optarg;
                    break;
                case 'f':
                    f_arg = optarg;
                    break;
            }
        }
        char* retval;
        getcwd(retval, sizeof(retval));
        filetraverse(parentDir, 0, argv[1], dash_S, s_arg, f_arg);
        chdir(retval);
        return 0;
}
```