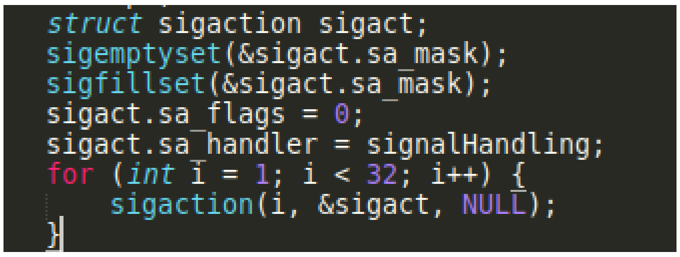
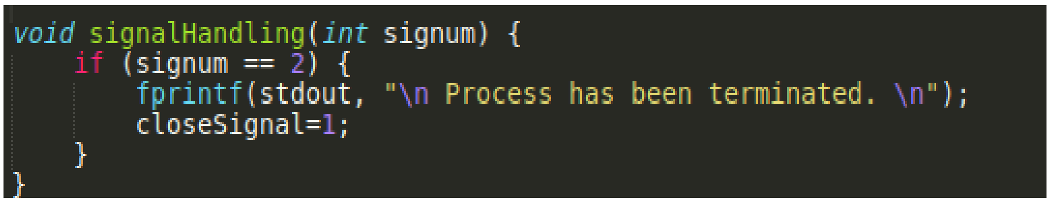
1)Signal Handling:



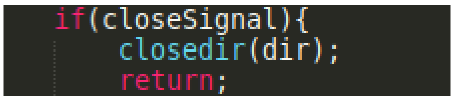
```c
struct sigaction sigact;
sigemptyset(&sigact.sa_mask);
sigfillset(&sigact.sa_mask);
sigact.sa_flags = 0;
sigact.sa_handler = signalHandling;
for (int i = 1; i < 32; i++) {
    sigaction(i, &sigact, NULL);
}
```

In this way,all signals can be catch except sigkill and sigstop which are uncatchable,and process using signalHandling function.



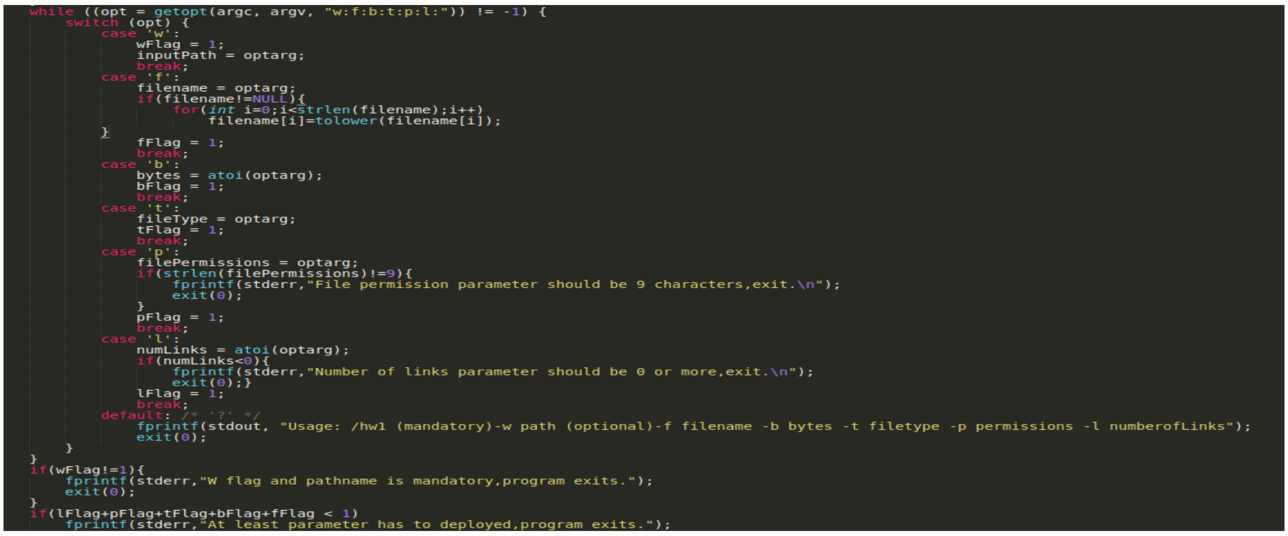
```c
void signalHandling(int signum) {
    if (signum == 2) {
        fprintf(stdout, "\n Process has been terminated. \n");
        closeSignal=1;
    }
}
```

However,i only need SIGINT signal which is ctrl-C.When this signal comes,a global variable closesignal will be 1.CloseSignal always controls on recursive directory search,so if it become 1,this recursion starts to end.



```c
    if(closeSignal){
        closedir(dir);
        return;
```

2)Argument Check



```c
    while ((opt = getopt(argc, argv, "w:f:b:t:p:l:")) != -1) {
        switch (opt) {
            case 'w':
                wFlag = 1;
                inputPath = optarg;
                break;
            case 'f':
                filename = optarg;
                if(filename!=NULL){
                    for(int i=0;i<strlen(filename);i++)
                        filename[i]=tolower(filename[i]);
            }
                fFlag = 1;
                break;
            case 'b':
                bytes = atoi(optarg);
                bFlag = 1;
                break;
            case 't':
                fileType = optarg;
                tFlag = 1;
                break;
            case 'p':
                filePermissions = optarg;
                if(strlen(filePermissions)!=9){
                    fprintf(stderr,"File permission parameter should be 9 characters,exit.\n");
                    exit(0);
                }
                pFlag = 1;
                break;
            case 'l':
                numLinks = atoi(optarg);
                if(numLinks<0){
                    fprintf(stderr,"Number of links parameter should be 0 or more,exit.\n");
                    exit(0);}
                lFlag = 1;
                break;
            default: /* '?' */
                fprintf(stdout, "Usage: /hw1 (mandatory)-w path (optional)-f filename -b bytes -t filetype -p permissions -l numberofLinks");
                exit(0);
        }
    }
    if(wFlag!=1){
        fprintf(stderr,"W flag and pathname is mandatory,program exits.");
        exit(0);
    }
    if(lFlag+pFlag+tFlag+bFlag+fFlag < 1)
        fprintf(stderr,"At least parameter has to deployed,program exits.");
```

Getopt library used for argument check,-wFlag accept as mandatory,filename converts into lower case,file perms are expect as 9 chars,links should be positive number.

3)Recursive search

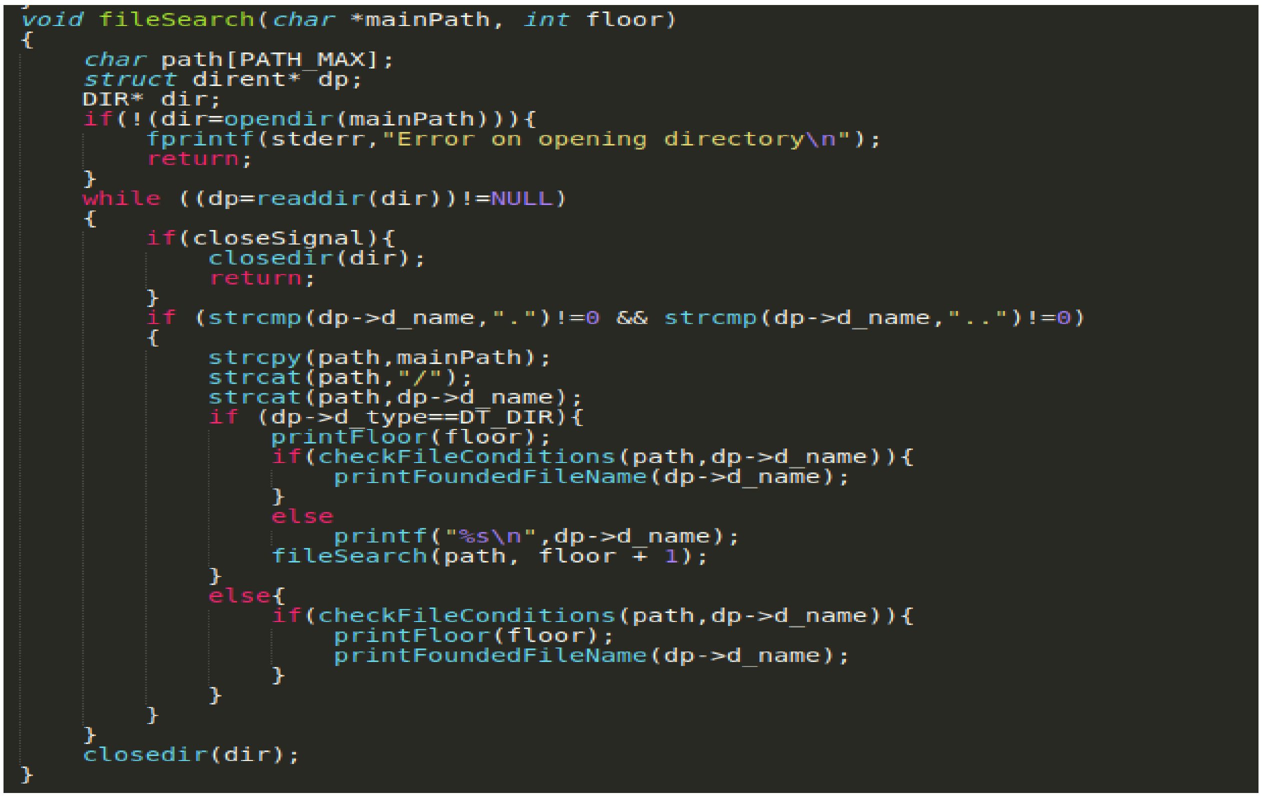
Function takes two arguments,one of them is the path of directory,the other one is for keep track the depth of this directory.PATH_MAX variable defined in Linux,so it can take maximum legal path length while searching.

Firstly open the directory,then inside this directory,get the files one by one.After every file process,close signal is checking,so if ctrl signal is received, it will stop after process the file.

Path will be iterate with everyfile with using strcat.

Checking for if file type is directory,if it is prints its name whether search or not and call recursive function with new directory path.

If its not directory,then check for if its searching file or not,and if its target,prints it.

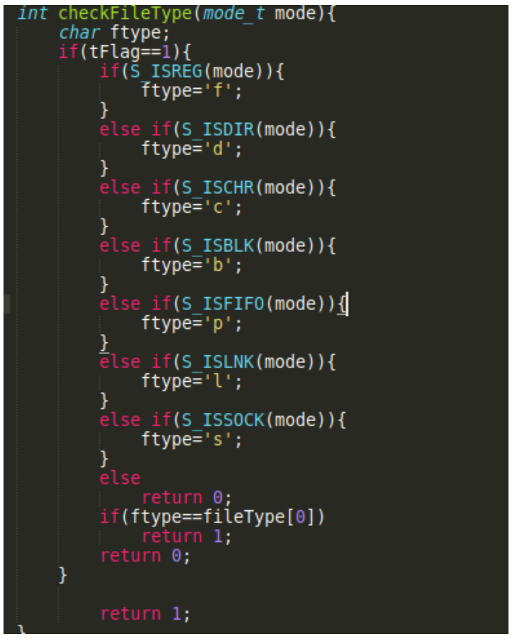


```c
void fileSearch(char *mainPath, int floor)
{
    char path[PATH_MAX];
    struct dirent* dp;
    DIR* dir;
    if(!(dir=opendir(mainPath))){
        fprintf(stderr,"Error on opening directory\n");
        return;
    }
    while ((dp=readdir(dir))!=NULL)
    {
        if(closeSignal){
            closedir(dir);
            return;
        }
        if (strcmp(dp->d_name,".")!=0 && strcmp(dp->d_name,"..")!=0)
        {
            strcpy(path,mainPath);
            strcat(path,"/");
            strcat(path,dp->d_name);
            if (dp->d_type==DT_DIR){
                printFloor(floor);
                if(checkFileConditions(path,dp->d_name)){
                    printFoundedFileName(dp->d_name);
                }
                else
                    printf("%s\n",dp->d_name);
                fileSearch(path, floor + 1);
            }
            else{
                if(checkFileConditions(path,dp->d_name)){
                    printFloor(floor);
                    printFoundedFileName(dp->d_name);
                }
            }
        }
    }
    closedir(dir);
}
```

4)CHECK

For all file conditions,lstat opens the path and its structure parts are checking in their relative functions.

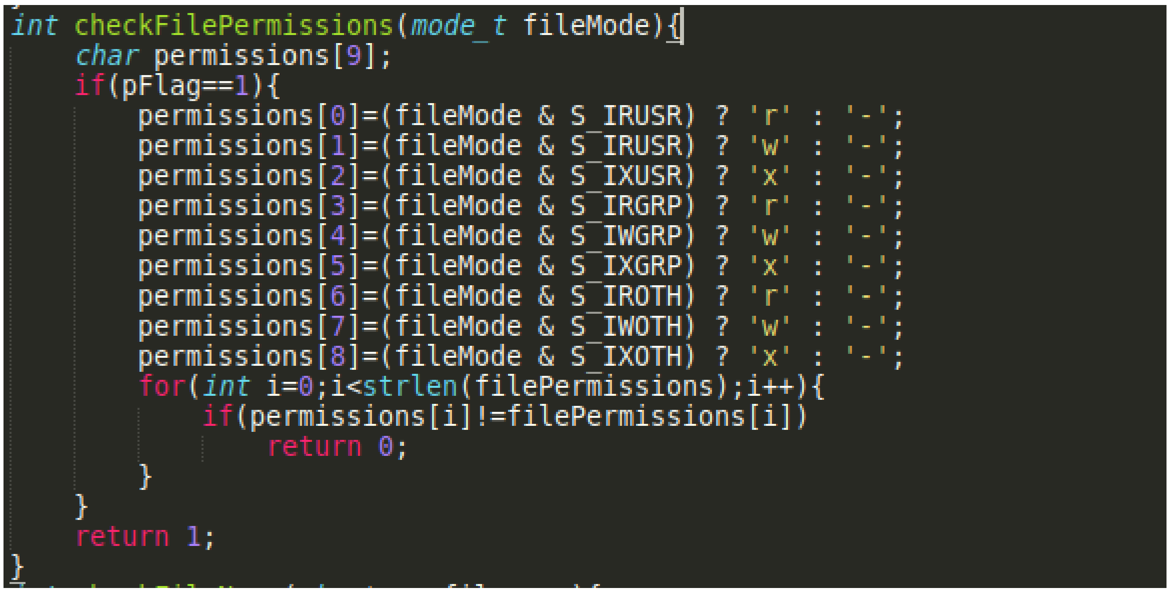
For file type checking using sys/stat.h macros.



```c
int checkFileType(mode_t mode){
    char ftype;
    if(tFlag==1){
        if(S_ISREG(mode)){
            ftype='f';
        }
        else if(S_ISDIR(mode)){
            ftype='d';
        }
        else if(S_ISCHR(mode)){
            ftype='c';
        }
        else if(S_ISBLK(mode)){
            ftype='b';
        }
        else if(S_ISFIFO(mode)){
            ftype='p';
        }
        else if(S_ISLNK(mode)){
            ftype='l';
        }
        else if(S_ISSOCK(mode)){
            ftype='s';
        }
        else
            return 0;
        if(ftype==fileType[0])
            return 1;
        return 0;
    }

        return 1;
}
```

For permission checking,9 permissions put in an char array then compare with input permissions.



```c
int checkFilePermissions(mode_t fileMode){
    char permissions[9];
    if(pFlag==1){
        permissions[0]=(fileMode & S_IRUSR) ? 'r' : '-';
        permissions[1]=(fileMode & S_IRUSR) ? 'w' : '-';
        permissions[2]=(fileMode & S_IXUSR) ? 'x' : '-';
        permissions[3]=(fileMode & S_IRGRP) ? 'r' : '-';
        permissions[4]=(fileMode & S_IWGRP) ? 'w' : '-';
        permissions[5]=(fileMode & S_IXGRP) ? 'x' : '-';
        permissions[6]=(fileMode & S_IROTH) ? 'r' : '-';
        permissions[7]=(fileMode & S_IWOTH) ? 'w' : '-';
        permissions[8]=(fileMode & S_IXOTH) ? 'x' : '-';
        for(int i=0;i<strlen(filePermissions);i++){
            if(permissions[i]!=filePermissions[i])
                return 0;
        }
    }
    return 1;
}
```

Link and filesize check just simply compare two variable with each other.

Filename check firstly check for if current filename is shorter then searched and if that’s true,no need any other because it will simply cant be the file we have searched.Then it compares char by char,when a + character appears,a helper function checks the regex and pass the characters.

5)File Found

If any file satisfy the conditions,its name written by a green text.