201250123 刘晓旭 Linux程序设计第三次作业

一、在一个只有128M内存并且没有交换分区的计算机上，说说两个程序在编译以及运行结果上的差别。

```
#define MEMSIZE 1024*1024
int count = 0;
void *p = NULL;
while(1) {
p = (void *)malloc(MEMSIZE);
if (!p) break;
printf("Current allocation %d MB\n", ++count);
}
#define MEMSIZE 1024*1024
int count = 0;
void *p = NULL;
while(1) {
p = (void *)malloc(MEMSIZE);
if (!p) break;
memset(p, 1, MEMSIZE);
printf("Current allocation %d MB\n", ++count);
}
```

第一段程序不断地分配内存，直到申请失败为止。第二段也是分配内存，但是每次分配内存之后会将内存填充为1。

在编译期间，由于第一段程序只是分配但是从未读取和写入，可能会被编译器将内存分配的步骤优化掉，导致程序陷入循环；但是第二段程序在分配的基础上多了写入过程，在编译过程中不会被优化。

体现在运行结果上，第一段可能陷入死循环或是不能分配内存，第二段可以一直分配内存，直到系统内存不足为止。

二、使用程序设计语言实现ls、wc命令

实现 源码 比较不同并写入文档

1.wc命令

功能

实现对文件的行数、单词数和字节数展示。

用法

```
wc {filepath}
```

代码

```
#include <stdio.h>

void wordCount(FILE *fp, char * fileName){
    if(fp==NULL){
        printf("Can't open file %s.\n",fileName);
        return;
    }
    int line=0,word=0,byte=0;
    char c;
    int state=0;//状态为0为可以接收新单词的状态
    while((c= fgetc(fp))!=EOF){
        byte++;
        if(c=='\n'||c=='\t'||c==' '){
            if(c=='\n'){
                line++;
            }
            state=0;
            continue;
        }else{
            if(state==0){
                word++;
                state=1;
            }
            continue;
        }
    }
    printf("%d %d %d",line,word,byte);
}
int main(int argc,char* argv[]){
    int count=1;
    while(count < argc){//支持多个文件查询
        FILE* fPointer = fopen(argv[count],"r");
        wordCount(fPointer,argv[count]);
        count++;
    }
}
```

源码

1.源码引入了更多的库

```
#include "safe-read.h"
#include "xfreopen.h"
etc
```

2.源码内嵌了更多的struct

```
struct fstatus{
    int failed;
    struct stas st;
}
```

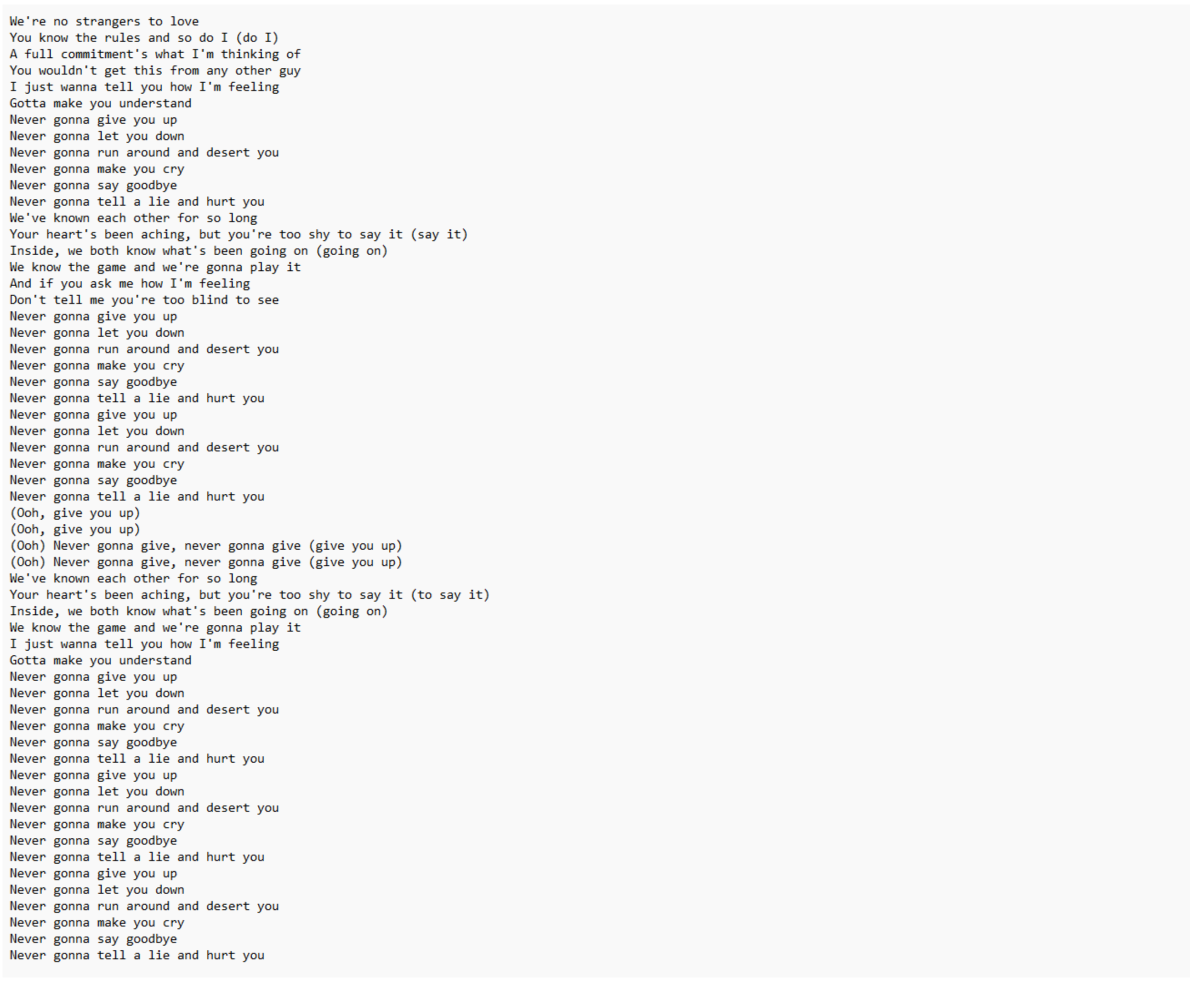
3.源码的组合性更强

```
源码与其它函数的组合性更强，能更好调用其它函数
```

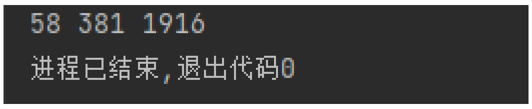
4.源码支持的参数更多

```
除了wc filepath，还支持参数如-l等
```

结果



```
We're no strangers to love
You know the rules and so do I (do I)
A full commitment's what I'm thinking of
You wouldn't get this from any other guy
I just wanna tell you how I'm feeling
Gotta make you understand
Never gonna give you up
Never gonna let you down
Never gonna run around and desert you
Never gonna make you cry
Never gonna say goodbye
Never gonna tell a lie and hurt you
We've known each other for so long
Your heart's been aching, but you're too shy to say it (say it)
Inside, we both know what's been going on (going on)
We know the game and we're gonna play it
And if you ask me how I'm feeling
Don't tell me you're too blind to see
Never gonna give you up
Never gonna let you down
Never gonna run around and desert you
Never gonna make you cry
Never gonna say goodbye
Never gonna tell a lie and hurt you
Never gonna give you up
Never gonna let you down
Never gonna run around and desert you
Never gonna make you cry
Never gonna say goodbye
Never gonna tell a lie and hurt you
(Ooh, give you up)
(Ooh, give you up)
(Ooh) Never gonna give, never gonna give (give you up)
(Ooh) Never gonna give, never gonna give (give you up)
We've known each other for so long
Your heart's been aching, but you're too shy to say it (to say it)
Inside, we both know what's been going on (going on)
We know the game and we're gonna play it
I just wanna tell you how I'm feeling
Gotta make you understand
Never gonna give you up
Never gonna let you down
Never gonna run around and desert you
Never gonna make you cry
Never gonna say goodbye
Never gonna tell a lie and hurt you
Never gonna give you up
Never gonna let you down
Never gonna run around and desert you
Never gonna make you cry
Never gonna say goodbye
Never gonna tell a lie and hurt you
Never gonna give you up
Never gonna let you down
Never gonna run around and desert you
Never gonna make you cry
Never gonna say goodbye
Never gonna tell a lie and hurt you
```



```
58 381 1916
进程已结束,退出代码0
```

2.ls命令

功能

实现显示当前目录下的文件和相关信息。

```
-i 显示文件inode号
-l 列举目录内容的细节
-R 递归
-a 全部文件（包括隐藏文件）
-d 将目录像文件一样显示
```

用法

```
ls {-l -i -R -a -d}
```

代码

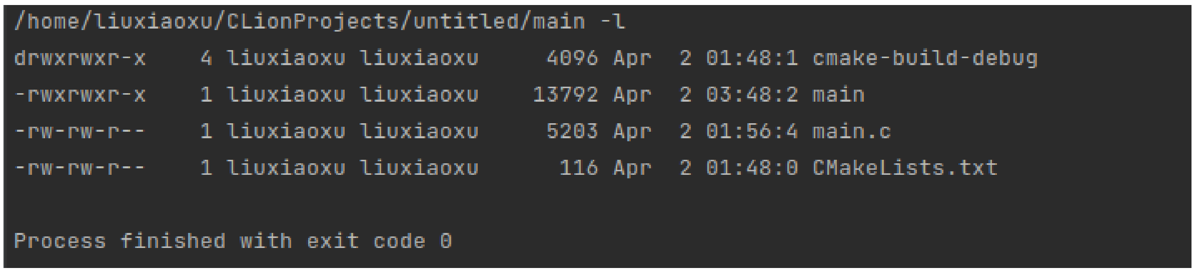
```
#include <stdio.h>
#include <sys/types.h>
#include <dirent.h>
#include <pwd.h>
#include <grp.h>
#include <sys/stat.h>
#include <unistd.h>
#include <string.h>
#include <stdlib.h>

int l_opt=0;
int d_opt=0;
int R_opt=0;
int a_opt=0;
int i_opt=0;

//将文件的访问权限转换为ls的字符形式，如-rwxr-xr--
void mode_to_letters(int mode, char str[]) {
    strcpy(str, "----------");

    if (S_ISDIR(mode)) {
        str[0] = 'd';
    }

    if (S_ISCHR(mode)) {
        str[0] = 'c';
    }

    if (S_ISBLK(mode)) {
        str[0] = 'b';
    }

    if ((mode & S_IRUSR)) {
        str[1] = 'r';
    }

    if ((mode & S_IWUSR)) {
        str[2] = 'w';
    }

    if ((mode & S_IXUSR)) {
        str[3] = 'x';
    }
    if ((mode & S_IRGRP)) {
        str[4] = 'r';
    }

    if ((mode & S_IWGRP)) {
        str[5] = 'w';
    }

    if ((mode & S_IXGRP)) {
        str[6] = 'x';
    }

    if ((mode & S_IROTH)) {
        str[7] = 'r';
    }

    if ((mode & S_IWOTH)) {
        str[8] = 'w';
    }

    if ((mode & S_IXOTH)) {
        str[9] = 'x';
    }
}

//将用户id转换为用户名
char *uid_to_name(uid_t uid) {
    struct passwd *pw_ptr;
    static char num_str[10];

    if ((pw_ptr = getpwuid(uid)) == NULL) {
        sprintf(num_str, "%d", uid);

        return num_str;
    } else {
        return pw_ptr->pw_name;
    }
}

//将组id转换为组名
char *gid_to_name(gid_t gid) {
    struct group *grp_ptr;
    static char num_str[10];

    if ((grp_ptr = getgrgid(gid)) == NULL) {
        sprintf(num_str, "%d", gid);
        return num_str;
    } else {
        return grp_ptr->gr_name;
    }
}

//输出文件信息
void output(struct dirent *dirent_pos, char *path) {
    if (i_opt) {
        printf("%20ld ", dirent_pos->d_ino);
    }
    if (l_opt) {
        struct stat info;
        char mode_str[11];
        char *ctime();
        if (stat(path, &info) == -1) {
            perror(path);
        } else {
            mode_to_letters(info.st_mode, mode_str);
            printf("%s ", mode_str);
            printf("%4d ", (int) info.st_nlink);
            printf("%-8s ", uid_to_name(info.st_uid));
            printf("%-8s ", gid_to_name(info.st_gid));
            printf("%8ld ", (long) info.st_size);
            printf("%.14s ", 4 + ctime(&info.st_mtime));
        }
    }
    printf("%s\n", dirent_pos->d_name);
}

//读取完文件返回上一级目录
void rmPath(char *path, int len) {
    memset(path + len, '\0', strlen(path) - len);
}

//将文件名添加到路径末尾，便于读取下一级目录中的文件
void addPath(char *path, char *add) {
    path[strlen(path)] = '/';
    long len = strlen(path);
    memmove(path + len, add, strlen(add));
}

//读取指定目录的所有文件
void readDir(DIR *dir_ptr, char *path) {
    struct dirent *dirent_pos;
    int len = strlen(path);
    DIR * tmp_pos[100];
    char tmp_path[100][100];
    int cnt = 0;
    if (R_opt) printf("Source directory: %s\n", path);
    while ((dirent_pos = readdir(dir_ptr)) != NULL) {

        addPath(path, dirent_pos->d_name);
        if (d_opt) {
            output(dirent_pos, path);
            return;
        }
        if (dirent_pos->d_name[0] == '.' && !a_opt) {
            rmPath(path, len);
            continue;
        }

        output(dirent_pos, path);
        if (R_opt) {
            if(strlen(dirent_pos->d_name) == 1 && dirent_pos->d_name[0] == '.')
{
                rmPath(path, len);
                continue;
            }
            if(strlen(dirent_pos->d_name) == 2 && dirent_pos->d_name[0] == '.' &&
dirent_pos->d_name[1] == '.') {
                rmPath(path, len);
                continue;
            }
            // d_type：8-文件，4-目录
            if (dirent_pos->d_type == 4) {
                DIR *dir_ptr_next;
                if ((dir_ptr_next = opendir(path)) != NULL) {
                    memcpy(tmp_path[cnt], path, strlen(path));
                    tmp_pos[cnt++] = dir_ptr_next;
                }
            }
        }
        rmPath(path, len);
    }
    if(R_opt){
        for(int i = 0; i < cnt;++i){
            readDir(tmp_pos[i], tmp_path[i]);
        }
    }
    closedir(dir_ptr);
}

//读取参数并查询错误
int main(int argc,char* argv[]) {
    char argument;
    if(argc > 1){
        while((argument = getopt(argc,argv,"ldRai"))!=-1){
            switch(argument){
                case'l':
                    l_opt=1;break;
                case'd':
                    d_opt=1;break;
                case'R':
                    R_opt=1;break;
                case'a':
                    a_opt=1;break;
                case'i':
                    i_opt=1;break;
                default:
                    fprintf(stderr,"parameter error.");return 1;
            }
        }
    }
    DIR * DIR_PTR;
    char * PATH=getenv("PWD");
    if((DIR_PTR = opendir("."))==NULL){
        fprintf(stderr,"folder failed to open.");return 1;
    }else{
        char ROOT[1000];
        memset(ROOT, '\0', sizeof(ROOT));
        strcpy(ROOT, PATH);
        readDir(DIR_PTR, ROOT);
    }
}
```

源码

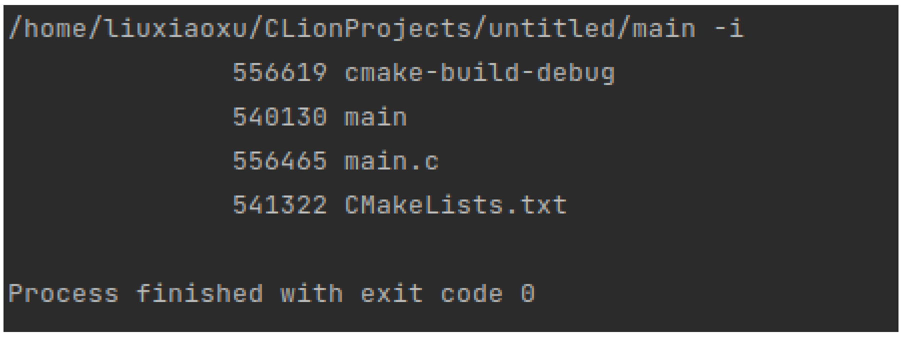
1.源码能支持更多参数

```
-h 将文件大小以易懂方式显示
-t 按照文件修改时间先后顺序排序
-r 将文件以相反顺序显示
-F 在文件、目录的后面添加一个特殊字符，表示其类型
同时，还有-c -d -G等。
```

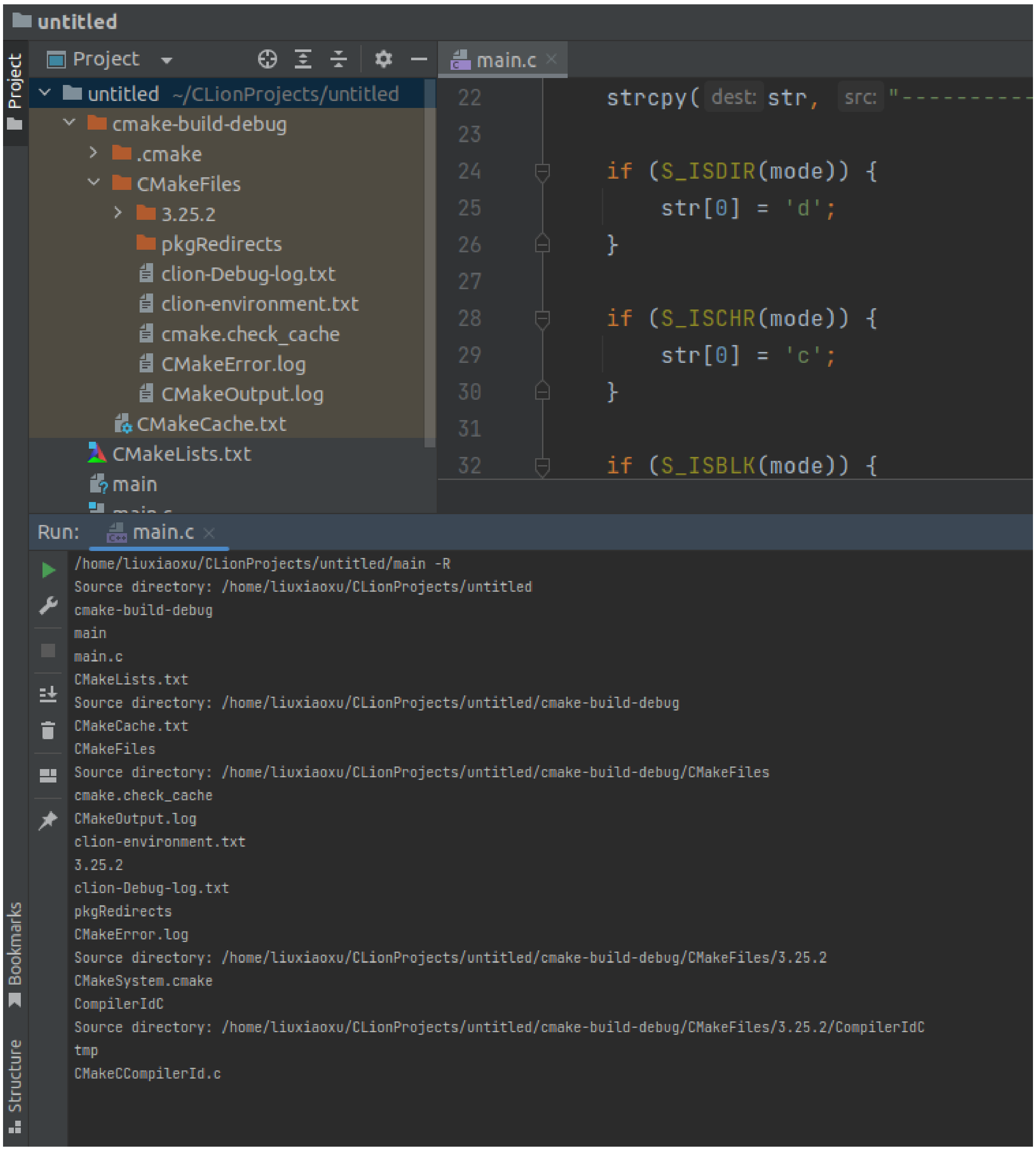
结果



```
/home/liuxiaoxu/CLionProjects/untitled/main -l
drwxrwxr-x    4 liuxiaoxu liuxiaoxu     4096 Apr  2 01:48:1 cmake-build-debug
-rwxrwxr-x    1 liuxiaoxu liuxiaoxu    13792 Apr  2 03:48:2 main
-rw-rw-r--    1 liuxiaoxu liuxiaoxu     5203 Apr  2 01:56:4 main.c
-rw-rw-r--    1 liuxiaoxu liuxiaoxu      116 Apr  2 01:48:0 CMakeLists.txt

Process finished with exit code 0
```

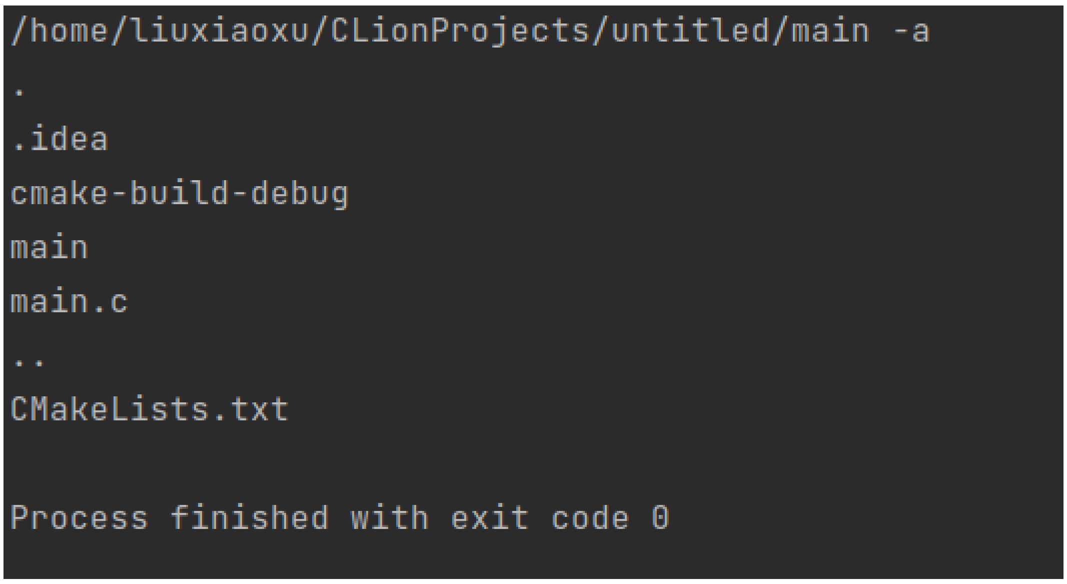


```
/home/liuxiaoxu/CLionProjects/untitled/main -i
              556619 cmake-build-debug
              540130 main
              556465 main.c
              541322 CMakeLists.txt

Process finished with exit code 0
```

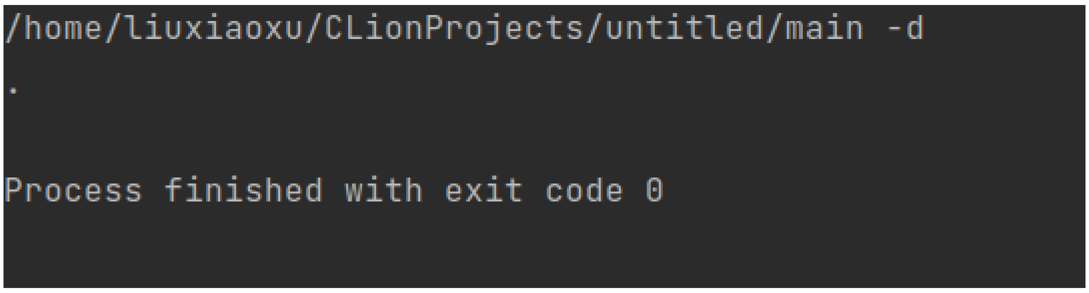


```
    strcpy( dest: str,  src: "---------

    if (S_ISDIR(mode)) {
        str[0] = 'd';
    }

    if (S_ISCHR(mode)) {
        str[0] = 'c';
    }

    if (S_ISBLK(mode)) {
```

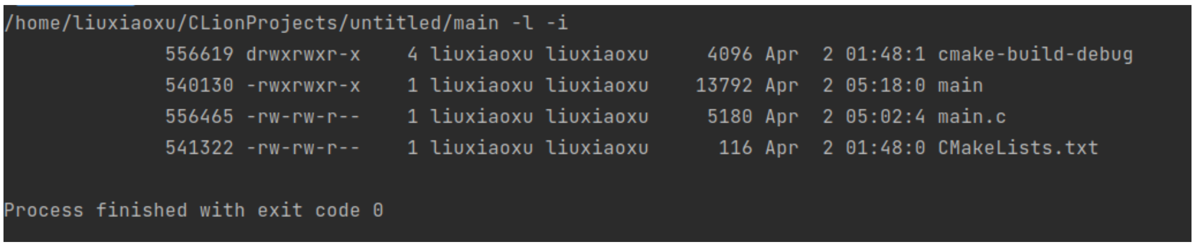
```
/home/liuxiaoxu/CLionProjects/untitled/main -R
Source directory: /home/liuxiaoxu/CLionProjects/untitled
cmake-build-debug
main
main.c
CMakeLists.txt
Source directory: /home/liuxiaoxu/CLionProjects/untitled/cmake-build-debug
CMakeCache.txt
CMakeFiles
Source directory: /home/liuxiaoxu/CLionProjects/untitled/cmake-build-debug/CMakeFiles
cmake.check_cache
CMakeOutput.log
clion-environment.txt
3.25.2
clion-Debug-log.txt
pkgRedirects
CMakeError.log
Source directory: /home/liuxiaoxu/CLionProjects/untitled/cmake-build-debug/CMakeFiles/3.25.2
CMakeSystem.cmake
CompilerIdC
Source directory: /home/liuxiaoxu/CLionProjects/untitled/cmake-build-debug/CMakeFiles/3.25.2/CompilerIdC
tmp
CMakeCCompilerId.c
```



```
/home/liuxiaoxu/CLionProjects/untitled/main -a
.
.idea
cmake-build-debug
main
main.c
..
CMakeLists.txt

Process finished with exit code 0
```



```
/home/liuxiaoxu/CLionProjects/untitled/main -d
.

Process finished with exit code 0
```



```
/home/liuxiaoxu/CLionProjects/untitled/main -l -i
              556619 drwxrwxr-x    4 liuxiaoxu liuxiaoxu     4096 Apr  2 01:48:1 cmake-build-debug
              540130 -rwxrwxr-x    1 liuxiaoxu liuxiaoxu    13792 Apr  2 05:18:0 main
              556465 -rw-rw-r--    1 liuxiaoxu liuxiaoxu     5180 Apr  2 05:02:4 main.c
              541322 -rw-rw-r--    1 liuxiaoxu liuxiaoxu      116 Apr  2 01:48:0 CMakeLists.txt

Process finished with exit code 0
```