```
/*
Name: Hung-Yi Lu
BlazerId: lu0106
Project #:Homework 3
To compile: use command "make"
To run: use command:
./hw3 -e "ls -l -s"
./hw3 -f jpg -E "tar cvf jpg.tar"
./hw3 -f txt -e "ls -l -s"
*/
#include <stdio.h>
#include <stdlib.h>
#include <dirent.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <unistd.h>
#include <string.h>
#include <fcntl.h>

// same as HW2
typedef int FileFunction(char *path, int size, char *str, int S, int d, int t, int nest
ingC, int countspace);
int test(char *path, int size, char *str, int S, int d, int t, int nestingC, int counts
pace);
int test2(char *path, int size, char *str, int S, int d, int t, int nestingC, int count
space, FileFunction *function);
int get_file_size(char *file_name);

char **ppath; // pwd path
int pnumber = 0; // pwd number
int *fta; // file type array
int k = 0; // for loop

// Reference https://www.delftstack.com/zh-tw/howto/c/optind-in-c/
int main(int argc, char *argv[]){

    ppath = (char **)malloc( 1024*sizeof( char *));
    for(k = 0; k < 1024; k++){
        ppath[k] = (char *)malloc( 128*sizeof( char));
    }
    fta = (int *)calloc(128, sizeof(int));

    char addr[999];
    memset(addr, 0, sizeof(addr));
    getcwd(addr, sizeof(addr));
    strcat(addr, "/");

    char addr2[999];
    memset(addr2, 0, sizeof(addr2));
    getcwd(addr2, sizeof(addr2));
    strcat(addr2, "/");

    char str[999];
    memset(str, 0, sizeof(str));

    char *strE = (char *)malloc(128*sizeof(char));
    char *stre = (char *)malloc(128*sizeof(char));

    int S = 0;
    int t = 0;
    int d = 0;
    int E = 0;
    int e = 0;
```

```
    int size = 0;

    pid_t pid;
    int exit_status;

    int nestingC = 1;
    int countspace = 0;

    int result;
    while((result = getopt(argc, argv, "Ss:f:t:E:e:")) != -1){
        switch(result){
            case 'S':
                S = 1;
                break;
            case 's':
                size = atoi(optarg);
                break;
            case 'f':
                strcpy(str, optarg);
                break;
            case 't':
                if(strcmp(optarg, "f") == 0){
                    t = 1;
                }
                else if (strcmp(optarg, "d") == 0){
                    d = 1;
                }
                else{
                    printf("Error\n");
                    exit(EXIT_SUCCESS);
                }
                break;
            case 'E':
                E = 1;
                strcpy(strE, optarg);
                break;
            case 'e':
                e = 1;
                strcpy(stre, optarg);
                break;
        }
    }

    if(argc > optind && argv[optind] != NULL){
        strcat(addr2, argv[argc-1]);

        if(argc == 1){
            test2(addr2, 0, str, 0, 0, 0, nestingC, countspace, test);
        }
        else{
            test2(addr2, size, str, S, d, t, nestingC, countspace, test);
        }
    }

    if(argc == 1){
        test2(addr, 0, str, 0, 0, 0, nestingC, countspace, test);
    }
    else{
        test2(addr, size, str, S, d, t, nestingC, countspace, test);
    }

// Reference http://c.biancheng.net/cpp/html/289.html
// Reference https://wenyuangg.github.io/posts/linux/fork-use.html
```

```
    pid = fork();
    if(pid == 0){  //fork()=0, child process
        if(E == 1){
            char *cppath = (char *)calloc(4096, sizeof(char));
            char put[4096];
            memset(put, 0, sizeof(put));
            strcpy(cppath, ppath[k]);

            for(k = 1; k < pnumber; k++){
                if(ppath[k] == NULL){
                    break;
                }
                strcat(cppath, " ");
                strcat(cppath, ppath[k]);
            }
            sprintf(put, "%s %s", strE, cppath);
            system(put);
        }

        if(e == 1){
            char put[128];
            DIR *Dir;
            struct dirent *dirent;

            memset(put, 0,sizeof(put));

            for(k = 0; k < pnumber; k++){
                if(fta[k] == 1){
                    sprintf(put, "%s %s", stre, ppath[k]);
                    system(put);
                    printf("\n");
                }
            }
        }
        exit(0);
    }
    else if (pid == -1){ // fork()=-1, no child process error
        perror("fork()");   // error message
        exit(-1);
    }

    else{  //fork()>0, parent
        wait(&exit_status);  //wait child process
        printf("[Parent] Child's exit status is [%d]\n", WEXITSTATUS(exit_status));
    }
    return 0;
}

// same as HW2
// get the type of file
// Reference http://www.gnu.org/software/libc/manual/html_node/Directory-Entries.html
char *get_file_type(unsigned char type){

    char *print;
    switch (type){
        case DT_REG:
            print = "regular file";
            break;
        case DT_DIR:
            print = "directory";
            break;
        case DT_FIFO:
```

```
            print = "FIFO";
            break;
        case DT_SOCK:
            print = "local-domain socket";
            break;
        case DT_CHR:
            print = "character device";
            break;
        case DT_BLK:
            print = "block device";
            break;
        case DT_LNK:
            print = "symbolic link";
            break;
        case DT_UNKNOWN:
            print = "unknown file type";
            break;
        default:
            print = "UNKNOWN";
    }
    return print;
}

int test(char *path, int size, char *str, int S, int d, int t, int nestingC, int counts
pace){

    struct dirent *ptr;
    int file_size = 2147483647; // max int

    char directory_name[999];
    memset(directory_name, 0,sizeof(directory_name));

    DIR *Dir = opendir(path);
    if (Dir == NULL){
        printf("Directory Opening Error!\n");
        exit(EXIT_FAILURE);
    }

    while ((ptr = readdir(Dir)) != NULL){
        if ((*ptr).d_type == DT_REG){
            file_size = get_file_size((*ptr).d_name);
        }

        //null 0 0
        if((str == NULL) && (t == 0) && (d == 0)){
            if(file_size >= size){
                for (k = 0; k < countspace; k++){
                    printf("         ");
                }
                printf("[%d] %s (File Type: %s)", countspace, (*ptr).d_name, get_file_t
ype((*ptr).d_type));

                if (!(strcmp((*ptr).d_name, ".") == 0 || strcmp((*ptr).d_name, "..") ==
 0)){
                        sprintf(ppath[pnumber], "%s/%s", path, (*ptr).d_name);
                        if((*ptr).d_type == DT_REG){ // regular file
                        fta[pnumber] = 1;
                        }
                        else if((*ptr).d_type == DT_DIR){ // directory
                        fta[pnumber] = 0;
                        }
                        else{
                        fta[pnumber] = -1;
```

```
                        }
                        pnumber++;
                    }

                if( (S == 1) && ((*ptr).d_type == DT_REG)){  // regular file
                    printf("\t(File Size: %d)", file_size);
                }
                printf("\n");
                if ((*ptr).d_type != DT_DIR){ // directory
                    continue; // run next loop
                }
                else if (!(strcmp((*ptr).d_name, ".") == 0 || strcmp((*ptr).d_name, ".."
") == 0)){
                    sprintf(directory_name, "%s/%s", path, (*ptr).d_name); // directory
_name = path, ptr.d_name
                    test(directory_name, size, str, S, d, t, nestingC, countspace + 1);
 // c+1
                }
                nestingC = nestingC +1;
            }
        }

        //null 0 1
        if((str == NULL) && (t == 0) && (d == 1)){
            if(strcmp(get_file_type((*ptr).d_type), "directory") == 0){
                if(file_size >= size){
                    for (k = 0; k < countspace; k++){
                        printf("         ");
                    }
                    printf("[%d] %s (File Type: %s)", countspace,(*ptr).d_name, get_fil
e_type((*ptr).d_type));

                    if (!(strcmp((*ptr).d_name, ".") == 0 || strcmp((*ptr).d_name, ".."
) == 0)){
                        sprintf(ppath[pnumber], "%s/%s", path, (*ptr).d_name);
                        if((*ptr).d_type == DT_REG){ // regular file
                        fta[pnumber] = 1;
                        }
                        else if((*ptr).d_type == DT_DIR){ // directory
                        fta[pnumber] = 0;
                        }
                        else{
                        fta[pnumber] = -1;
                        }
                        pnumber++;
                    }

                    if( (S == 1) && ((*ptr).d_type == DT_REG)){ // regular file
                        printf("\t(File Size: %d)", file_size);
                    }
                    printf("\n");

                    if ((*ptr).d_type != DT_DIR){ // directory
                        continue; // run next loop
                    }
                    else if (!(strcmp((*ptr).d_name, ".") == 0 || strcmp((*ptr).d_name,
 "..") == 0)){
                        sprintf(directory_name, "%s/%s", path, (*ptr).d_name); // direc
tory_name = path, ptr.d_name
                        test(directory_name, size, str, S, d, t, nestingC, countspace+1
); // c+1
                    }
                    nestingC = nestingC +1;
```

```
                }
            }
        }


        //null 1 0
        if((str == NULL) && (t == 1) && (d == 0)){
            if(strcmp( get_file_type((*ptr).d_type), "regular file") == 0){
                if(file_size >= size){
                    for (k = 0; k < countspace; k++){
                        printf("         ");
                    }
                    printf("[%d] %s (File Type: %s)", countspace,(*ptr).d_name, get_fil
e_type((*ptr).d_type));

                    if (!(strcmp((*ptr).d_name, ".") == 0 || strcmp((*ptr).d_name, ".."
) == 0)){
                        sprintf(ppath[pnumber], "%s/%s", path, (*ptr).d_name);
                        if((*ptr).d_type == DT_REG){ // regular file
                        fta[pnumber] = 1;
                        }
                        else if((*ptr).d_type == DT_DIR){ // directory
                        fta[pnumber] = 0;
                        }
                        else{
                        fta[pnumber] = -1;
                        }
                        pnumber++;
                    }

                    if( (S == 1) && ((*ptr).d_type == DT_REG)){ // regular file
                        printf("\t(File Size: %d)", file_size);
                    }
                    printf("\n");

                    if ((*ptr).d_type != DT_DIR){ // directory
                        continue; // run next loop
                    }
                    else if (!(strcmp((*ptr).d_name, ".") == 0 || strcmp((*ptr).d_name,
 "..") == 0)){
                        sprintf(directory_name, "%s/%s", path, (*ptr).d_name); // direc
tory_name = path, ptr.d_name
                        test(directory_name, size, str, S, d, t, nestingC, countspace+1
); // c+1
                    }
                    nestingC = nestingC +1;
                }
            }
        }

        // 0 0
        if((str != NULL) && (t == 0) && (d == 0)){
            if(strstr((*ptr).d_name, str) != NULL){
                if(file_size >= size){
                    for (k = 0; k < countspace; k++){
                        printf("         ");
                    }
                    printf("[%d] %s (File Type: %s)", countspace,(*ptr).d_name, get_fil
e_type((*ptr).d_type));

                    if (!(strcmp((*ptr).d_name, ".") == 0 || strcmp((*ptr).d_name, ".."
) == 0)){
                        sprintf(ppath[pnumber], "%s/%s", path, (*ptr).d_name);
```

```
                        if((*ptr).d_type == DT_REG){ // regular file
                        fta[pnumber] = 1;
                        }
                        else if((*ptr).d_type == DT_DIR){ // directory
                        fta[pnumber] = 0;
                        }
                        else{
                        fta[pnumber] = -1;
                        }
                        pnumber++;
                    }

                    if( (S == 1) && ((*ptr).d_type == DT_REG)){ // regular file
                        printf("\t(File Size: %d)", file_size);
                    }
                    printf("\n");

                    if ((*ptr).d_type != DT_DIR){ // directory
                        continue; // run next loop
                    }
                    else if (!(strcmp((*ptr).d_name, ".") == 0 || strcmp((*ptr).d_name,
 "..") == 0)){
                        sprintf(directory_name, "%s/%s", path, (*ptr).d_name); // direc
tory_name = path, ptr.d_name
                        test(directory_name, size, str, S, d, t, nestingC, countspace+1
); // c+1
                    }
                    nestingC = nestingC +1;
                }
            }
        }

        // 0 1
        if((str != NULL) && (t == 0) && (d == 1)){
            if((strstr( (*ptr).d_name, str) != NULL) && (strcmp( get_file_type((*ptr).d
_type), "directory") == 0)){
                if(file_size >= size){
                    for (k = 0; k < countspace; k++){
                        printf("         ");
                    }
                    printf("[%d] %s (File Type: %s)", countspace,(*ptr).d_name, get_fil
e_type((*ptr).d_type));

                    if (!(strcmp((*ptr).d_name, ".") == 0 || strcmp((*ptr).d_name, ".."
) == 0)){
                        sprintf(ppath[pnumber], "%s/%s", path, (*ptr).d_name);
                        if((*ptr).d_type == DT_REG){ // regular file
                        fta[pnumber] = 1;
                        }
                        else if((*ptr).d_type == DT_DIR){ // directory
                        fta[pnumber] = 0;
                        }
                        else{
                        fta[pnumber] = -1;
                        }
                        pnumber++;
                    }

                    if( (S == 1) && ((*ptr).d_type == DT_REG)){ // regular file
                        printf("\t(File Size: %d)", file_size);
                    }
                    printf("\n");
```

```
                    if ((*ptr).d_type != DT_DIR){ // directory
                        continue; // run next loop
                    }
                    else if (!(strcmp((*ptr).d_name, ".") == 0 || strcmp((*ptr).d_name,
 "..") == 0)){
                        sprintf(directory_name, "%s/%s", path, (*ptr).d_name); // direc
tory_name = path, ptr.d_name
                        test(directory_name, size, str, S, d, t, nestingC, countspace+1
); // c+1
                    }
                    nestingC = nestingC +1;
                }
            }
        }

        // 1 0
        if((str != NULL) && (t == 1) && (d == 0)){
            if((strstr( (*ptr).d_name, str) != NULL) && (strcmp( get_file_type((*ptr).d
_type), "regular file") == 0)){
                if(file_size >= size){
                    for (k = 0; k < countspace; k++){
                        printf("         ");
                    }
                    printf("[%d] %s (File Type: %s)", countspace, (*ptr).d_name, get_fi
le_type((*ptr).d_type));

                    if (!(strcmp((*ptr).d_name, ".") == 0 || strcmp((*ptr).d_name, ".."
) == 0)){
                        sprintf(ppath[pnumber], "%s/%s", path, (*ptr).d_name);
                        if((*ptr).d_type == DT_REG){ // regular file
                        fta[pnumber] = 1;
                        }
                        else if((*ptr).d_type == DT_DIR){ // directory
                        fta[pnumber] = 0;
                        }
                        else{
                        fta[pnumber] = -1;
                        }
                        pnumber++;
                    }

                    if( (S == 1) && ((*ptr).d_type == DT_REG)){ // regular file
                        printf("\t(File Size: %d)", file_size);
                    }
                    printf("\n");

                    if ((*ptr).d_type != DT_DIR){ // directory
                        continue; // run next loop
                    }
                    else if (!(strcmp((*ptr).d_name, ".") == 0 || strcmp((*ptr).d_name,
 "..") == 0)){
                        sprintf(directory_name, "%s/%s", path, (*ptr).d_name); // direc
tory_name = path, ptr.d_name
                        test(directory_name, size, str, S, d, t, nestingC, countspace+1
); // c+1
                    }
                    nestingC = nestingC +1;
                }
            }
        }

        // Error
        if((t == 1) && (d == 1)){
```

```
            printf("Error t and d Error\n");
            exit(EXIT_FAILURE);
        }
    }
    closedir(Dir);
    return 1;
}

// same as HW2
int test2(char *path, int size, char *str, int S, int d, int t, int nestingC, int count
space, FileFunction *function){
    return function(path, size, str, S, d, t, nestingC, countspace);
}

// same as HW2
// get file size
// Reference https://stackoverflow.com/questions/238603/how-can-i-get-a-files-size-in-c
int get_file_size(char *file_name){

    struct stat st;
    stat(file_name, &st);
    int size = st.st_size;

    return size;
}
```