1. list 程序

1.1. 程序功能

```
-h           显示帮助
-a           显示.开头文件
-r           递归
-l <bytes>   只显示大于l bytes的文件
-s <bytes>   只显示小于s bytes的文件
-m <days>    只显示修改于m天之内的文件
```

1.2. 程序运行效果

```
ridd@Ridds-MacBook-Pro linux % make clean
ridd@Ridds-MacBook-Pro linux % make
ridd@Ridds-MacBook-Pro linux % ./list
            4516  list.c
             260  report.md
              64  makefile
           50576  list
          folder  testDir
ridd@Ridds-MacBook-Pro linux % ./list -a
          folder  .
          folder  ..
            4516  list.c
            6148  .DS_Store
             260  report.md
              64  makefile
           50576  list
          folder  testDir
ridd@Ridds-MacBook-Pro linux % ./list -a -r
          folder  .
          folder  ..
            4516  list.c
            6148  .DS_Store
             260  report.md
              64  makefile
           50576  list
          folder  testDir
          folder  testDir/.
          folder  testDir/..
               0  testDir/testFile
ridd@Ridds-MacBook-Pro linux % ./list -l 1024 -s 10240
            4516  list.c
            1511  report.md
          folder  testDir
```

1.3. 源代码

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

#define SEC_PER_DAY 86400
#define MAX_BUF_SIZE 1024
#define NUM_OPT 6 //number of available options
#define H_VALUE INT32_MAX
#define L_VALUE -1

int showAll = 0, recursively = 0, opt_g = 0, largestSize = H_VALUE,
smallestSize = L_VALUE, modified = H_VALUE;
int *pOpt[NUM_OPT] = {&showAll, &recursively, &modified, &smallestSize,
&largestSize, &opt_g};

void list(char *path, char *prePath, char *filename) {
    DIR *pDir = NULL;
    struct dirent *pDirent;
    struct stat st;
    if (stat(path, &st) < 0) {
        printf("%s: No such file or directory.\n", path);
        return;
    }
    if (S_ISDIR(st.st_mode)) {
        if ((pDir = opendir(path)) == NULL) {
            printf("%s: No such file or directory.\n", path);
            return;
        }
        while ((pDirent = readdir(pDir)) != NULL) {
            if ((showAll == 0) && (pDirent->d_name[0] == '.')) {
                continue;
            }
            char newPath[MAX_BUF_SIZE] = {0}, buf[MAX_BUF_SIZE] =
{0};//initialize sub path
            strcat(newPath, path), strcat(newPath, "/"), strcat(buf, prePath),
strcat(buf, pDirent->d_name);
            if (pDirent->d_type == 4) {// is folder
                printf("%16s  %s\n", "folder", buf);
                if (recursively && (strcmp("..", pDirent->d_name) != 0) &&
(strcmp(".", pDirent->d_name) != 0)) {
                    list(strcat(newPath, pDirent->d_name), strcat(buf, "/"),
pDirent->d_name);
                }
            } else {
                list(strcat(newPath, pDirent->d_name), prePath, pDirent-
>d_name);
            }
        }
    } else {
        char buf[MAX_BUF_SIZE] = {0};
        time_t now = time(NULL);
        strcat(buf, prePath);
        if (st.st_size > smallestSize && st.st_size < largestSize && (now -
st.st_mtime) / SEC_PER_DAY < modified)
            printf("%16lld  %s\n", st.st_size, strcat(buf, filename));
    }
}

void listFileInCwd(char *filename, int *num) {
    char *cwd = getcwd(NULL, MAX_BUF_SIZE);
    strcat(cwd, "/");
    if (filename[0] == '/') {
        list(filename, "", filename);
    } else {
        list(strcat(cwd, filename), "", filename);
    }
    free(cwd);
    *num += 1;
}

void help() {
    printf("LIST 1.0.1 by Ridd, %s %s \
        \nUsage: list [OPTION]... [FILE]..., \
        \nList information about the FILEs (the current directory by
default),\n \
        \n-h\tDisplay help\
        \n-a\tShow files starting with\
        \n-r\tList subdirectories recursively\
        \n-l <bytes>\tOnly show files larger than <bytes>\
        \n-s <bytes>\tOnly show files smaller than <bytes>\
        \n-m <days> \tOnly show files modified within <days>\n",
           __DATE__, __TIME__);
    exit(0);
}

void parse(int argc, char **argv) {
    int state = 0, num = 0;
```

```c
    for (int i = 1; i < argc; i++) {// argv[0] is path to executable
        switch (state) {
            case 0:
                switch (argv[i][1]) {
                    case 'r':
                        recursively = 1;
                        break;
                    case 'a':
                        showAll = 1;
                        break;
                    case 'm':
                        state = 2;
                        break;
                    case 'l':
                        state = 3;
                        break;
                    case 's':
                        state = 4;
                        break;
                    case '-':
                        state = 5;
                        break;
                    case 'h':
                        help();
                        break;
                    default:
                        listFileInCwd(argv[i], &num);
                }
                break;
            case 8:
                listFileInCwd(argv[i], &num);
                break;
            default:
                if (state > 1 && state < 5)
                    *pOpt[state] = atoi(argv[i]);
                else if (state <= 1)
                    *pOpt[state] = 1;
                else if (state == 5) {
                    state = 8;
                    break;
                }
                state = 0;
                break;
        }
    }
    if ((state != 0) && (state != 8)) {
        help();
    } else if (num == 0) {
        listFileInCwd("", &num);
    }

}

int main(int argc, char *argv[]) {
    parse(argc, argv);
    return 0;
}
```