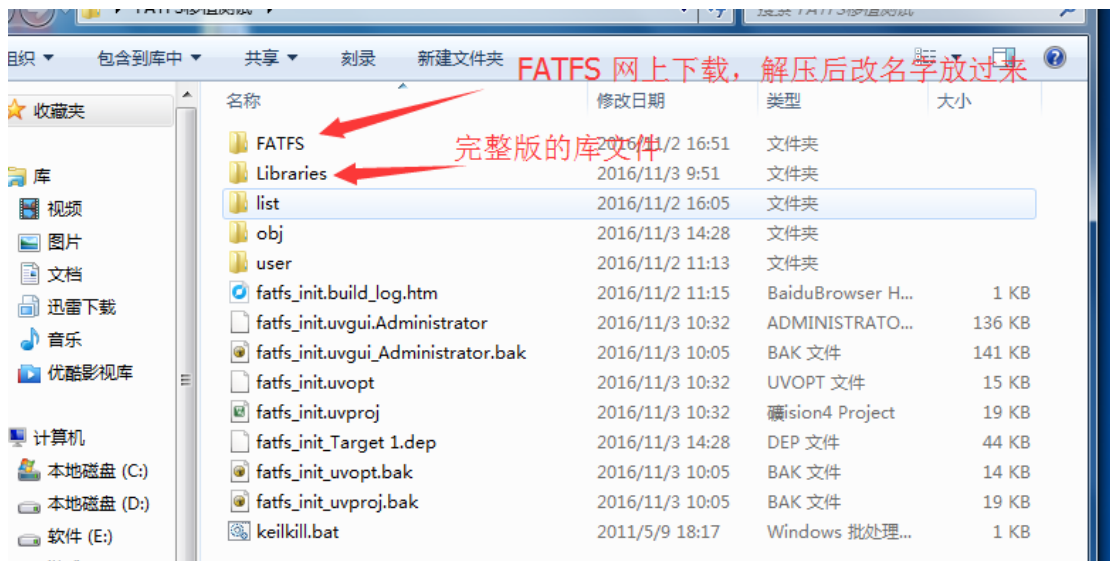


# FATFS 文件系统移植教程 韩工

一步一步慢慢来

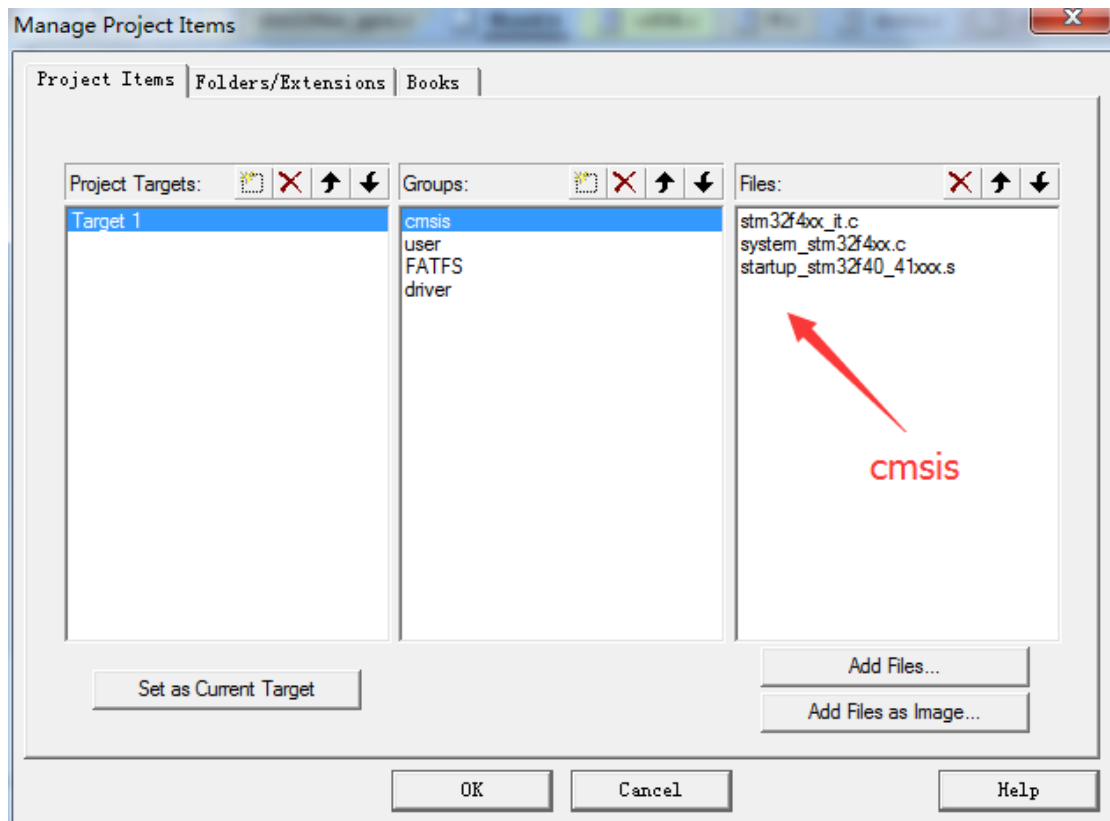
第 1 步:

将 FATFS 文件夹（官网下载解压改名成 FATFS）和完整的库函数文件夹 Libraries 放到工程模板里面

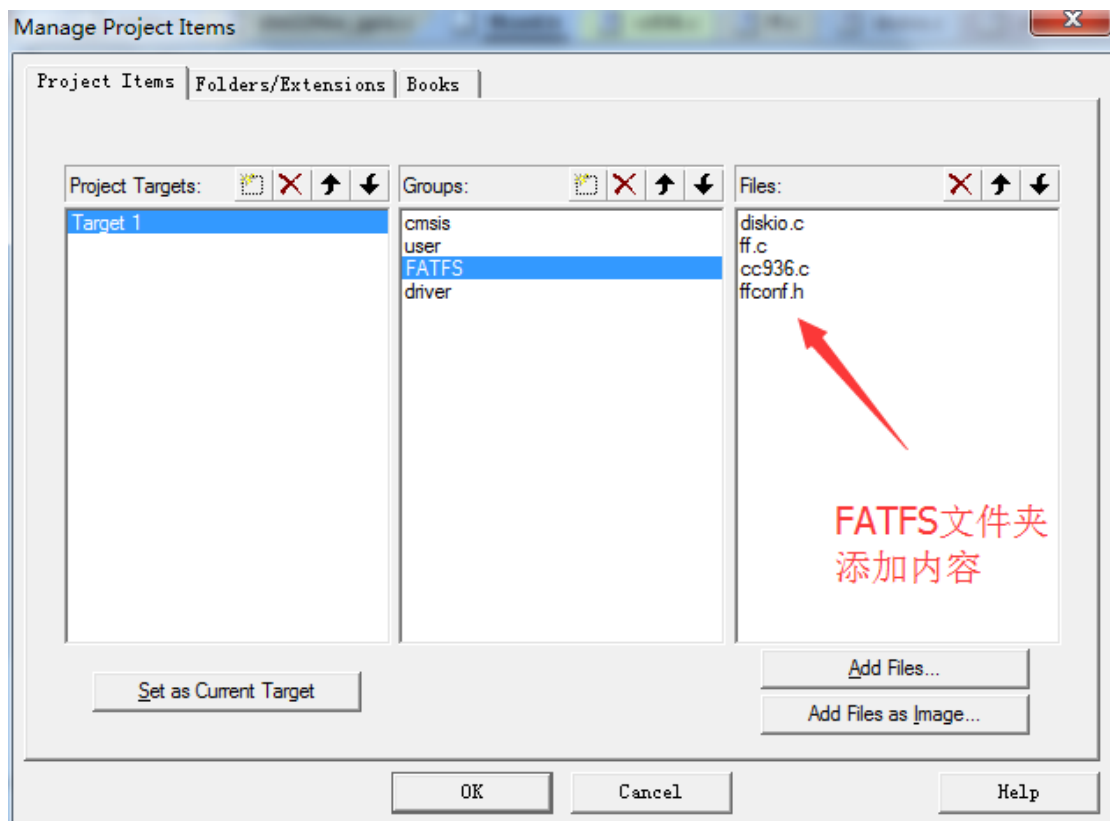


第 2 步，建工程的时候确保不同文件夹内有这些东西

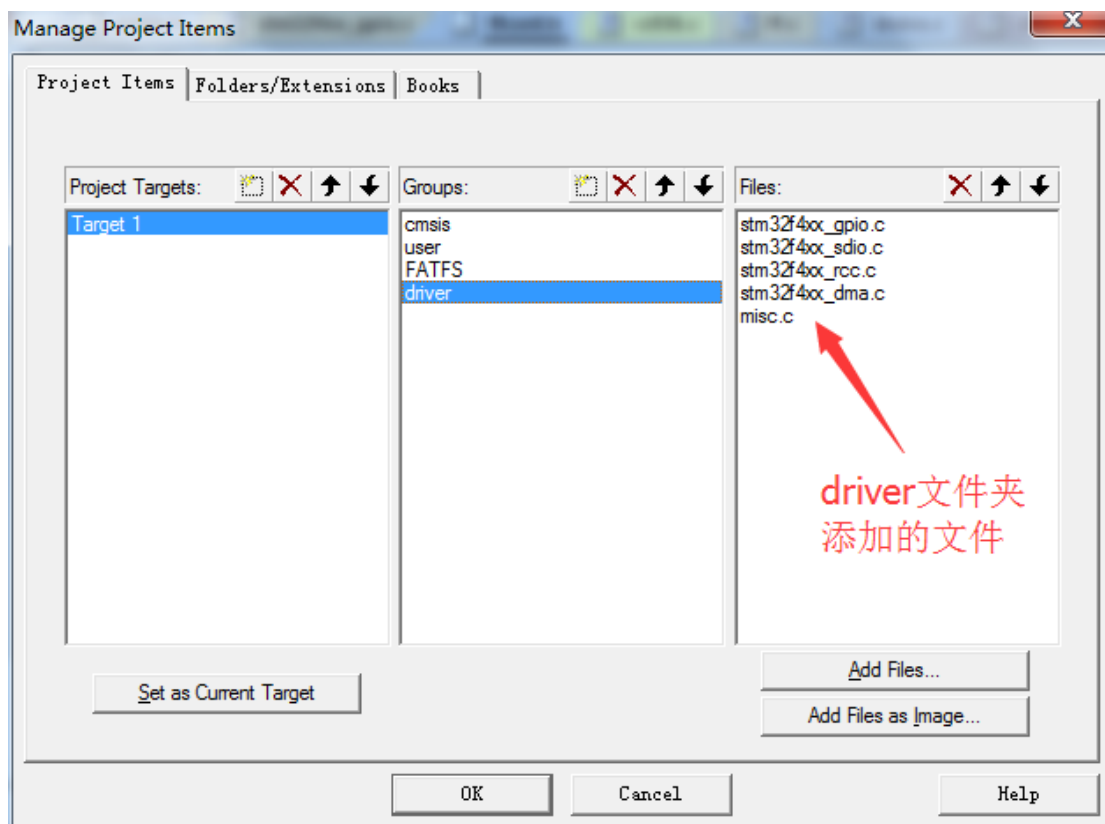
a: CMSIS 文件夹包含文件



b: FATFS 文件夹包含的文件



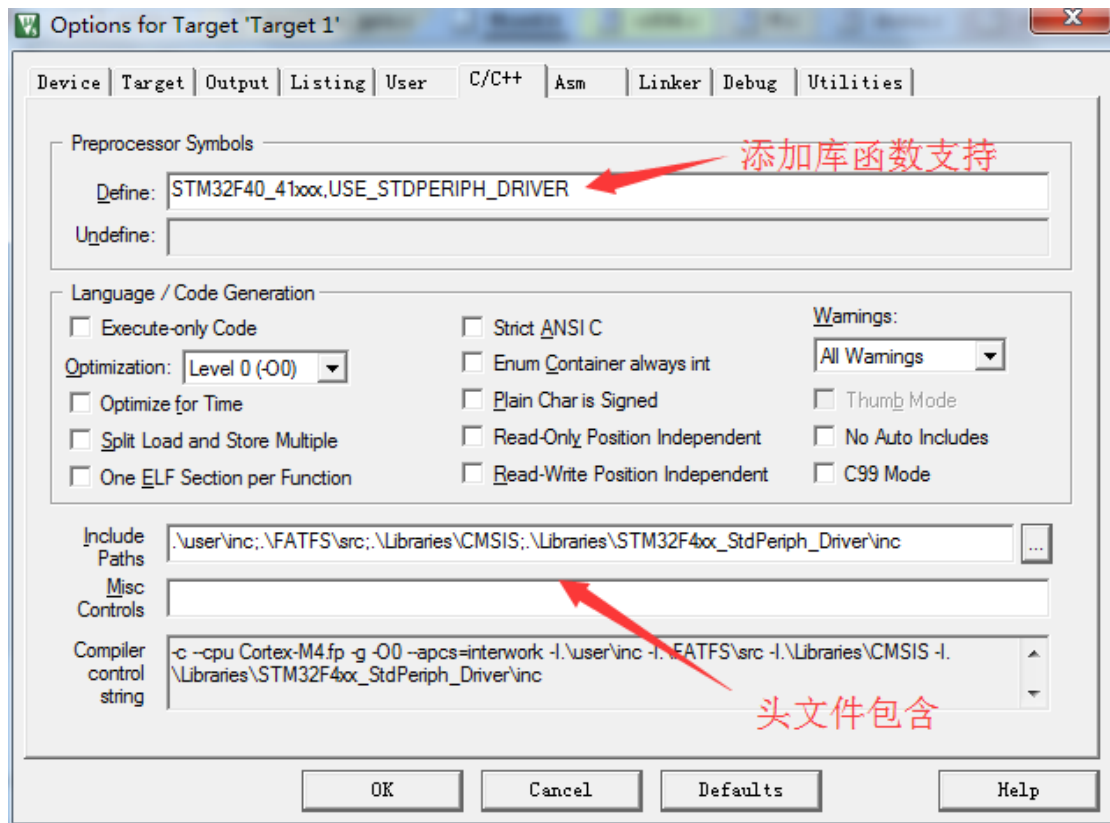
c: driver 文件夹中包含的文件:



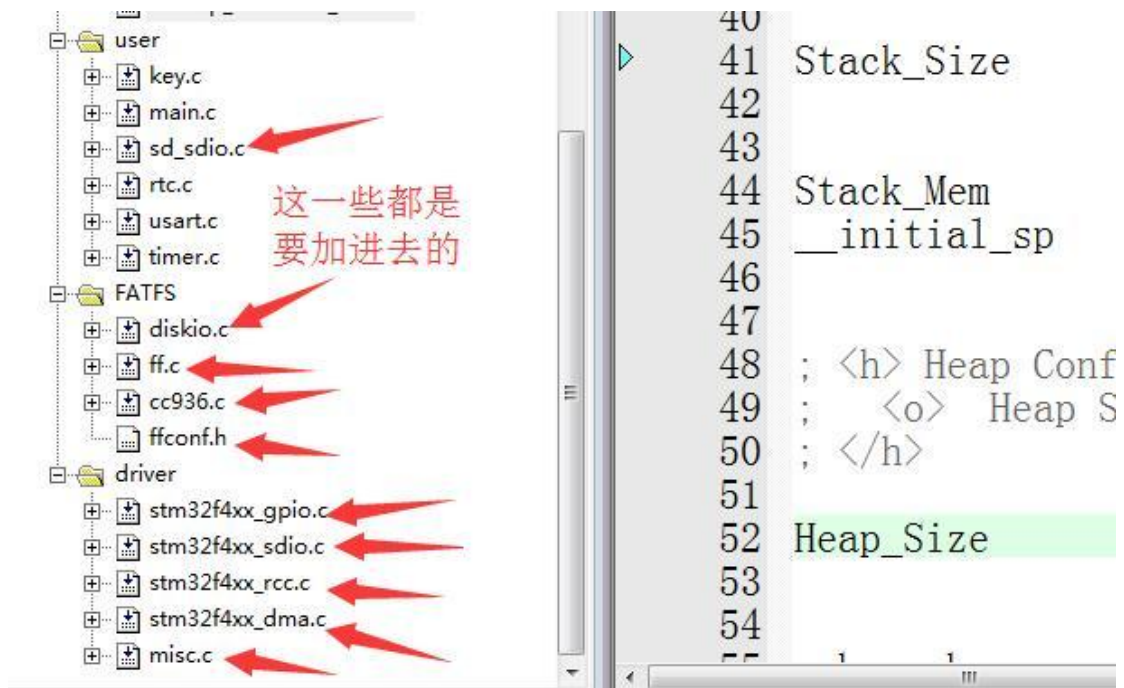
3: 工程树中需要的设置

a: C/C++中需要引入库函数支持

b: include 一栏中将图示的文件夹包含进去



4: 建工程成功后确保主界面左边文件夹中有这些文件, 其中 sd\_sdio.c 放在根目录\user\src, sd\_sdio.h 放在根目录\user\inc



## 5: 修改 ffconf.h 配置文件

### a: 中文支持

```
07
68 /*-----
69 /  Locale and Namespace Configurations
70 /-----
71 |
72 #define _CODE_PAGE 936
73 /* This option specifies the OEM code page to be used.
74 /  Incorrect setting of the code page can cause a variety of
75 /
76 /  1 - ASCII (No extended character. Non-LFN cfg files will not
77 /  437 - U.S.
78 /  720 - Arabic
79 /  737 - Greek
80 /  771 - KBL
81 /  775 - Baltic
82 /  850 - Latin 1
```

932改936支持中文

ffconf.h

### b: 长文件名支持

```
97 /  950 - Traditional Chinese (DBCS)
98 */
99 |
100 |
101 #define _USE_LFN 3
102 #define _MAX_LFN 255
103 /* The _USE_LFN switches the support of long file name (LFN)
104 /
105 /  0: Disable support of LFN. _MAX_LFN has no effect.
106 /  1: Enable LFN with static working buffer on the BSS.
107 /  2: Enable LFN with dynamic working buffer on the STACK.
108 /  3: Enable LFN with dynamic working buffer on the HEAP.
109 /
110 /  To enable the LFN, Unicode handling functions (optionally) must be added
111 /  to the project. The working buffer occupies (_MAX_LFN + 1) bytes.
```

本来是0, 改成3, 支持长文件名

ffconf.h

## 6: 修改 startup\_stm32f40\_41xxx.s 中的堆栈设置, 必须修改, 不然可以初始化 SD 卡, 可以挂载 SD 卡, 但是不能创建文件, 会报 17 错误

```
40
41 Stack_Size EQU 1024 * 4 ← 栈改变
42
43 AREA STACK, NOINIT, READWRITE, ALIGN=3
44 Stack_Mem SPACE Stack_Size
45 __initial_sp
46 startup_stm32f40_41xxx.s
47
48 ; <h> Heap Configuration
49 ; <o> Heap Size (in Bytes) <0x0-0xFFFFFFFF:8>
50 ; </h>
51
52 Heap_Size EQU 1024 * 8 ← 堆改变
53
54 AREA HEAP, NOINIT, READWRITE, ALIGN=3
```

## 7: 修改 diskio.c 文件

a: 添加 sd\_sdio.h 头文件支持，注释掉不需要的设备，将 SD 卡定义为设备 0

```
6 /* This is an example of glue functions to attach various existing
7 /* storage control modules to the FatFs module with a defined API.
8 /*-----
9
10 #include "diskio.h" /* FatFs lower layer API */
11 #include "sd_sdio.h" ← 添加SD卡头文件
12
13 /* Definitions of physical drive number for each drive */
14 // #define DEV_RAM 0 /* Example: Map Ramdisk to physical drive 0 */
15 #define DEV_MMC 1 /* Example: Map MMC/SD card to physical drive
16 // #define DEV_USB 2 /* Example: Map USB MSD to physical drive 2 */
17
18 #define DEV_SD 0 ← 自己定义SD卡设备，驱动器好为0
19
20 /*-----
21 /* Get Drive Status
```

b: 修改 disk\_status() 获取磁盘状态函数

```
22 /*-----
23
24 DSTATUS disk_status (
25     BYTE pdrv /* Physical drive number to identify the drive */
26 )
27 {
28     // DSTATUS stat;
29     int result;
30
31     switch (pdrv) {
32     case DEV_SD :
33         result = 0;
34         return result;
35     }
36     return STA_NOINIT;
37 }
```

获取状态函数删除多余的case, 直接返回0即可, 因为如果初始化未通过的话也不会对后面的操作有任何影响

c: 修改 disk\_initialize()初始化函数

```
44
45 DSTATUS disk_initialize (
46     BYTE pdrv /* Physical drive number to identify
47 )
48 {
49     // DSTATUS stat;
50     int result;
51
52     switch (pdrv) {
53     case DEV_SD :
54         result = SD_Init();
55         return result;
56     }
57     return STA_NOINIT;
58 }
```

初始化函数

d: 修改 disk\_read()磁盘读取函数

```
66 DRESULT disk_read (  
67     BYTE pdrv,      /* Physical drive number to identify the drive */  
68     BYTE *buff,    /* Data buffer to store read data */  
69     DWORD sector,  /* Start sector in LBA */  
70     UINT count     /* Number of sectors to read */  
71 )  
72 {  
73     int result;  
74     switch (pdrv) {  
75     case DEV_SD :  
76         result = SD_ReadDisk(buff, sector, count);  
77         return result;  
78     }  
79     return RES_PARERR;  
80 }
```

读取数据函数

e: 修改 disk\_write()磁盘写入函数

```
88 DRESULT disk_write (  
89     BYTE pdrv,      /* Physical drive number to identify the drive */  
90     const BYTE *buff, /* Data to be written */  
91     DWORD sector,  /* Start sector in LBA */  
92     UINT count     /* Number of sectors to write */  
93 )  
94 {  
95     int result;  
96  
97     switch (pdrv) {  
98     case DEV_SD :  
99         result = SD_WriteDisk((u8 *)buff, sector, count);  
100        return result;  
101    }  
102    return RES_PARERR;  
103 }
```

SD卡写入函数

f: 添加 disk\_ioctl()额外功能函数 (图片可以放大的)



```
110
111 DRESULT disk_ioctl (
112     BYTE pdrv, /* Physical drive number (0..) */
113     BYTE cmd, /* Control code */
114     void *buff /* Buffer to send/receive control data */
115 )
116 {
117     DRESULT res;
118     int result;
119     SD_CardInfo cardinfo;
120
121
122     switch (cmd) {
123         case CTRL_SYNC: //等待写过程
124
125             if (SD_GetState() == SD_CARD_READY) result = RES_ERROR; /*等待卡准备好*/
126             else res = RES_OK; //成功
127
128             break;
129         case GET_SECTOR_SIZE: //获取扇区大小
130             *(DWORD*)buff = 512;
131             res = RES_OK; //成功
132             break;
133         case GET_BLOCK_SIZE: //获取块大小
134             SD_GetCardInfo(&cardinfo); //获取卡信息
135             *(WORD*)buff = cardinfo.CardBlockSize; //块大小(扇区为单位)，一块等于 8 个扇区
136             res = RES_OK;
137             break;
138         case GET_SECTOR_COUNT: //获取总扇区数量
139
140             *(DWORD*)buff = cardinfo.CardCapacity/512;
141             res = RES_OK;
142             break;
143         default: //命令错误
144             res = RES_PARERR;
145             break;
146     }
147     return res;
148 }
```

额外控制函数

### g: 添加 malloc/free 函数

```
149
150
151 #include "stdlib.h"
152
153 void* ff_memalloc (UINT msize) /* Allocate memory block */
154 {
155     return malloc(msize);
156 }
157 void ff_memfree (void* mblock) /* Free memory block */
158 {
159     free(mblock);
160 }
161
162
163 #include "rtc.h"
164 DWORD get_fattime (void)
```

内存申请，释放函数

不要忘记添加头文件

### h: 添加 get\_fattime()获取修改时间函数

```
163 #include "rtc.h"
164 DWORD get_fattime (void)
165 {
166     u32 date;
167
168     date =
169     (
170     ((date_time.year+20) << 25) |
171     (date_time.month << 21) |
172     (date_time.day << 16) |
173     (date_time.hour << 11) |
174     (date_time.min << 5) |
175     (date_time.sec)
176     );
177     return date;
178 }
```

别忘了RTC头文件，此函数自动调用RTC，需要先实现RTC功能

获取修改文件时间函数

另外在 `sd_sdio.c` 文件里面有一部分是汇编语言写的编辑器打红叉，但是编译的时候不会报错，也不影响 FATFS 的使用，谁知道的来解释一下？

```
7
8 //关闭所有中断(但是不包括fault和NMI中断)
9 __asm__ void INTX_DISABLE(void)
10 {
11     CPSID    I
12     BX      LR
13 }
14 //开启所有中断
15 __asm__ void INTX_ENABLE(void)
16 {
17     CPSIE    I
18     BX      LR
19 }
20
21
22
```

`sd_sdio.c`里面有一部分用汇编写的会提示错误，但是不管貌似也可以正常使用

这样就可以了！可以在 `main()` 里面调用 API 函数进行各种文件操作了！