例程二十二 SDCard 的 FAT 系统文件的读写

这个讲例程主要讲解一下如果向 SDCard 写一个系统文件或读一个系统文件。上一讲 的内容主要是讲解向 SDCard 中的 Block 读写数据，而这一讲如果在往卡里面读写系统文件。FAT 公司已经提供一些底层和应用层的函数接口给我们。我们主要在利用他们的函数接口就可以完成这个读写过程了，系统文件的读写对我们来说非常重要的，特别是要在液晶屏在显示图片或者中文的时候就显得特别重要了，在后面的两讲会讲到，下面先来看看怎么往 SDCard 里面读写系统文件。在读写之前我们要到 FAT 官网下载几个文件。下面我们来看看那个几个文件是什么文件的。

1、 integer.h 数据类型的定义
2、 diskio.c 底层磁盘操作函数，这些是空函数，需要用户去完成
3、 ff.c 操作文件的函数库，用 C 来完成的

对于这些函数的操作是要完全了解和理解他的工作原理，很多东西并非我的能力所及，我只是带带大家如果移植到风驰电子的 STM8 开发板上，至于深一层的就有大家去研究研究。我们就把那几个文件添加到我们的工程就行了，跟别的文件添加过程一样。

下面再看看一下 diskio.c 里面的一些重要函数原型

```
/* Inidialize a Drive                                                    */

DSTATUS disk_initialize (
        BYTE drv                                /* Physical drive nmuber (0..) */
)

/*-----------------------------------------------------------------------
/* Return Disk Status

DSTATUS disk_status (
         BYTE drv                   /* Physical drive nmuber (0..) */
)

/*-----------------------------------------------------------------------*/
/* Read Sector(s)                                                        */

DRESULT disk_read (
         BYTE drv,                  /* Physical drive nmuber (0..) */
         BYTE *buff,                /* Data buffer to store read data */
         DWORD sector,    /* Sector address (LBA) */
         BYTE count                 /* Number of sectors to read (1..255) */
)

/*-----------------------------------------------------------------------*/
/* Write Sector(s)                                                       */

#if _READONLY == 0
DRESULT disk_write (
         BYTE drv,                          /* Physical drive nmuber (0..) */
         const BYTE *buff,        /* Data to be written */
         DWORD sector,            /* Sector address (LBA) */
         BYTE count                         /* Number of sectors to write (1..255) */
)
```

```
/*-----------------------------------------------------------------------*/
/* Miscellaneous Functions                                               */

DRESULT disk_ioctl (
        BYTE drv,               /* Physical drive nmuber (0..) */
        BYTE ctrl,              /* Control code */
        void *buff              /* Buffer to send/receive control data */
)
```

上面几个函数是底层操作的介质函数，一般需要用户去定义的，定义好的就交给 FAT 的应用函数去调用来操作 SDCard 了，相应的函数操作大家就看看例程吧。

先看看我的一个测试函数

```
void Sd_fs_test(void)
{

    disk_initialize( 0 );             /* SD 卡硬件初始化 */

    f_mount(0, &myfs[0]);          /*在文件系统注册一个工作区,*/
    myres = f_open( &myfsrc , "0:/My__Demo.TXT" , FA_CREATE_NEW | FA_WRITE);
    /*打开一个文件*/
    if ( myres == FR_OK )
    {
        /* Write buffer to file */
        myres = f_write(&myfsrc, mystring, sizeof(mystring), &mybr);
         /*往一个文件里面写数据*/
        f_close(&myfsrc);
    }
    else if ( myres == FR_EXIST )   //如果文件已经存在
    {
    }

    f_close(&myfsrc);     /* 关闭打开的文件 */

}
```

这个测试函数就是往 SDCard 创建一个文件并写数据。整个过程：先 SD 卡硬件初始化，这里面包括 SPI 的初始话，和 SDCard 的挂起状态，也就是上一讲的初始化→创建并打开文件→写数据→关闭文件。整个流程就是这样。下面再看看那个函数原型吧

```
FRESULT f_mount (
        BYTE vol,               /* Logical drive number to be mounted/unmounted */
        FATFS *fs               /* Pointer to new file system object (NULL for unmount)*/
```

```
/*-----------------------------------------------------------------------*/
/* Open or Create a File                                                 */
/*-----------------------------------------------------------------------*/

FRESULT f_open (
        FIL *fp,                        /* Pointer to the blank file object */
        const XCHAR *path,      /* Pointer to the file name */
        BYTE mode                       /* Access mode and file open mode flags */
)
```

```
/*-----------------------------------------------------------------------*/
/* Write File                                                            */
/*-----------------------------------------------------------------------*/

FRESULT f_write (
        FIL *fp,                        /* Pointer to the file object */
        const void *buff,       /* Pointer to the data to be written */
        UINT btw,                       /* Number of bytes to write */
        UINT *bw                        /* Pointer to number of bytes written */
)
```

```
/*-----------------------------------------------------------------------*/
/* Read File                                                             */
/*-----------------------------------------------------------------------*/

FRESULT f_read (
        FIL *fp,                /* Pointer to the file object */
        void *buff,             /* Pointer to data buffer */
        UINT btr,               /* Number of bytes to read */
        UINT *br                /* Pointer to number of bytes read */
)
/*-----------------------------------------------------------------------*/
/* Close File                                                            */
/*-----------------------------------------------------------------------*/

FRESULT f_close (
        FIL *fp         /* Pointer to the file object to be closed */
)
```

一般最常用的就是这几个函数，其实大家看了它的英文注释也大概了解到这个函数的意思了吧。大家结合例程中的例子进行简单的分析下。

f_mount(0, &myfs[0]);　在文件系统注册一个工作区，这个工作区的设备号为 0，相对于盘符的意思，这个参数开设为 0～9，&myfs[0]是指向工作区的指针。一般定义 myfs 为 FATFS 类型，这个 FATFS 是个结构体类型的。

f_open(&myfsrc , "0:/My__Demo.TXT" , FA_CREATE_NEW | FA_WRITE);这个函数的意思是在刚刚开辟盘符 0 下一个以只写的方式打开文件名为 My__Demo.TXT 的文件，如果该文件不存在，就新建一个名为 My__Demo.TXT。该文件是关联到 myfsrc 这个结构体指针的，在之后的文件读写都是通过该结构体去完成的。

myres = f_write(&myfsrc, mystring, sizeof(mystring), &mybr);这个函数就要把你要写的信息写进刚才创建或打开的 My__Demo.TXT 中。

f_close(&myfsrc); 关闭文件。

f_read(&myfsrc, save_buffer, sizeof(save_buffer), &mybr);这个函数就是从已经打开的文件里读出数据放在 save_buffer。

但是从应用层方面来说，会运用上面所说那几个函数就可以了，会使用这几个函数就会了对系统文件的读写，就可以做更多东西了，接下那 2 讲的教程就是要用到这几个函数对系统文件的读写。我们期待着下 2 讲教程。

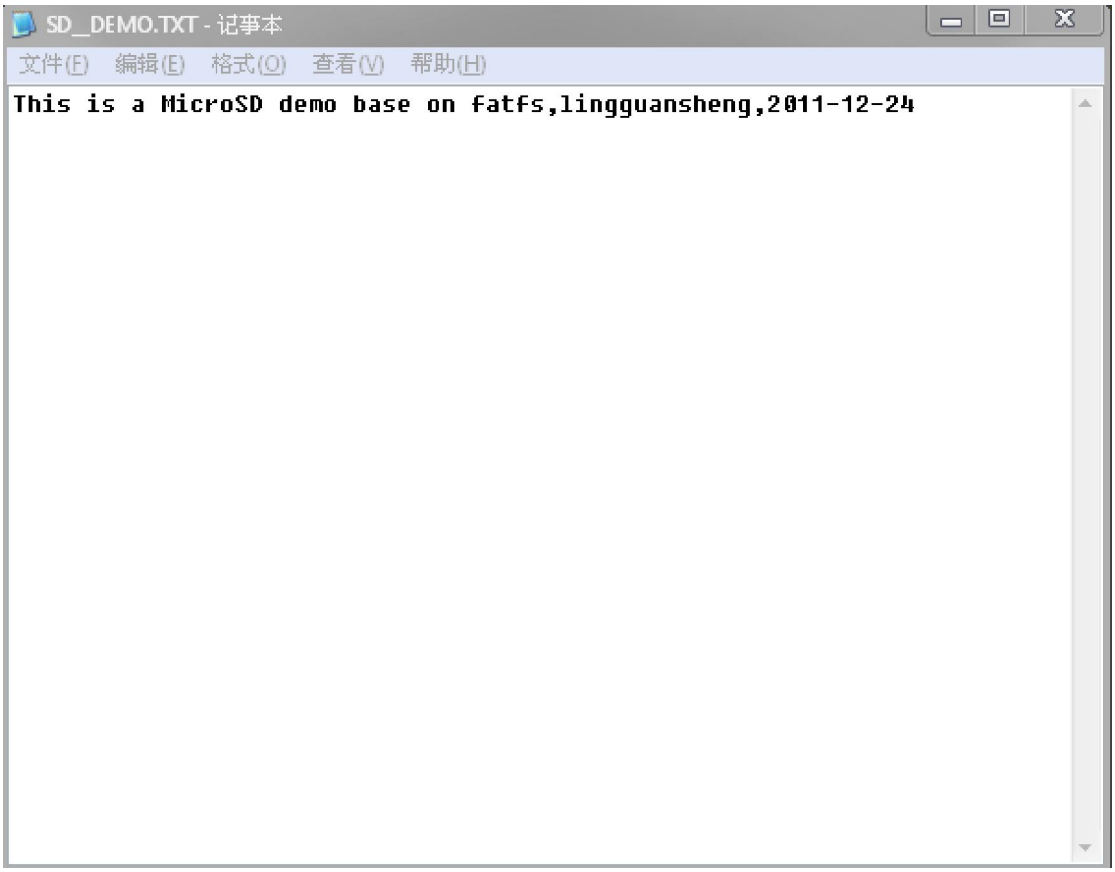
下面就是看看主函数

```
int main(void)
{
  /* Infinite loop */

  /*设置内部时钟16M为主时钟*/

  CLK_HSIPrescalerConfig(CLK_PRESCALER_HSIDIV1);
  /*!<Set High speed internal clock  */
  Uart_Init();

   while(SD_CARD_USER_Init())
  {
    UART1_SendString("SD Card Failed!",sizeof("SD Card Failed!"));
  }
  UART1_SendString("SD Card Checked OK",sizeof("SD Card Checked OK"));
  Sd_fs_test();
  Sdfs_NewCreateFile("0:/SD__Demo.TXT", pData, sizeof(pData));
  while (1)
  {
    /*  添加你的代码  */
  }
}
```

很简单，就是一个测试系统文件函数和一个创建系统文件并往系统文件里面写数据函数，相信大家看完我对上面的那几个函数的分析，相信大家也知道这个 2 个函数里面是怎样的了。测试函数在上面已经看过了，那下面就看看创建文件的函数的原型

```
/*****************************************************************************
* Function Name  : Sdfs_NewCreateFile
* Description    : 新建一个文件并写入数据
* Input          : new_file_name--新建文件的名称
*                  write_buffer--写入文件的数据缓冲区地址
*                  buffer_size--缓冲区大小
* Output         : None
* Return         : 0(success)  1(file existed )  -1(fail)
* Attention              : None
*****************************************************************************/
int Sdfs_NewCreateFile(BYTE *new_file_name, BYTE *write_buffer, u16 buffer_size)
{


    f_mount(0, &myfs[0]);
    myres = f_open( &myfsrc , (char*)new_file_name , FA_CREATE_NEW | FA_WRITE);

    if ( myres == FR_OK )
    {

        myres = f_write(&myfsrc, write_buffer,buffer_size, &mybr); //写入数据
        f_close(&myfsrc);

        return 0;

    }

    else if ( myres == FR_EXIST )  //如果文件已经存在
    {
        return FR_EXIST;
    }

    else
    {
        return -1;
    }
```

下载完我的例程，在 SDCard 套里面放进 SD，运行一遍，用读卡器在 PC 上打开就可以看到 2 个 TXT 文件。其中的一个就是这个样子



风驰电子祝您学习愉快～～～！！！！