

File System

1 简介

文件系统是用于明确存储设备或分区上的文件的方法和数据结构;即在存储设备上组织文件的方法。从系统角度来看,文件系统是对文件存储设备的空间进行组织和分配,负责文件存储并对存入的文件进行保护和检索的系统。具体地说,它负责为用户建立文件,存入、读出、修改,控制文件的存取,当用户不再使用时撤销文件等。

2 接口定义

文件系统接口分为文件操作、文件夹操作和文件系统操作三部分

【文件操作相关】

2.1 FileOpen

@brief Open a file

@param file_name file name // ".\\mp3\\short.mp3",\\mp3\\short.mp3

@param mode open mode // FA_READ|FA_WRITE|FA_CREATE_ALWAYS

@return File Handle

字符串打开文件不支持中文名的文件夹或文件,文件格式只支持 8.3 格式非中文

```
bool FileOpen(FAT_FILE* File, const uint8_t* FileName, const uint8_t Mode);
```

2.2 FileEOF

@brief Check End-of-File indicator.

@param File Pointer to a FILE object that specifies stream.

@return The feof function returns a nonzero value if a read operation has attempted to read past the end of the file; it returns 0 otherwise.

```
int32_t FileEOF(FAT_FILE* File);
```

2.3 FileRewind

返回到文件开头

```
void FileRewind(FAT_FILE* File);
```

2.4 FileOpenByNum

按文件序号打开文件

打开成功：返回 TRUE，目标文件信息填充到 File 所指的结构体。

打开失败：返回 FALSE。

Folder == NULL：打开整个设备上的第 FileNum 个文件。

Folder != NULL：打开 Folder 文件夹中的第 FileNum 个文件夹。

```
bool FileOpenByNum(FAT_FILE* File, FOLDER* Folder, uint16_t FileNum);
```

2.5 FileOpenByName

按文件名称打开指定文件夹中的文件。

打开成功：返回 TRUE，目标文件信息填充到 File 所指的结构体。

打开失败：返回 FALSE。

Folder 指针不能为空指针。

```
bool FileOpenByName(FAT_FILE* File, FOLDER* Folder, uint8_t* FileName);
```

2.6 FileOpenByLfName

按文件长名称打开指定文件夹中的文件。

打开成功：返回 TRUE，目标文件信息填充到 File 所指的结构体。

打开失败：返回 FALSE。

Folder 不能为空，FileName 为 Unicode 16 编码

```
bool FileOpenByLfName(FAT_FILE* File, FOLDER* Folder, uint8_t* FileName, uint8_t Len);
```

2.7 FileSeek

@brief Moves the file pointer to a specified location.

@param File Pointer to a FILE object that specifies stream.

@param Offset Number of bytes from origin.

@param Base Initial position.

@return If successful, the function returns a zero value. Otherwise, it returns nonzero value.

```
int32_t FileSeek(FAT_FILE* File, int32_t Offset, uint8_t Base);
```

2.8 FileRead

@brief Read data from stream
@param buffer Pointer to a block of buffer with a minimum size of (size*count) bytes.
@param size Size in bytes of each element to be read.
@param count Number of elements, each one with a size of size bytes.
@param File Pointer to a FILE object that specifies an input stream.
@return The total number of elements successfully read.

```
uint32_t FileRead(void* buffer, uint32_t size, uint32_t count, FAT_FILE* File);
```

2.9 FileWrite

@brief Write data to stream
@param buffer Pointer to a block of buffer with a minimum size of (size*count) bytes.
@param size Size in bytes of each element to be write.
@param count Number of elements, each one with a size of size bytes.
@param File Pointer to a FILE object that specifies an output stream.
@return The total number of elements successfully write.

```
uint32_t FileWrite(const void* buffer, uint32_t size, uint32_t count, FAT_FILE* File);
```

2.10 FileSof

@brief Get size of the opened file.
@param File Pointer to a FILE object that specifies an output stream.
@return size of the opened file.

```
int32_t FileSof(FAT_FILE* File);
```

2.11 FileTell

@brief Gets the current position of a file pointer.
@param File Pointer to a FILE object that specifies stream.
@return the current position of a file pointer in byte.

```
int32_t FileTell(FAT_FILE* File);
```

2.12 FileSave

@brief File save Fat and Dir

@param File Pointer to a FILE object that specifies an output stream.

@return fclose returns 0 if the stream is successfully closed.

```
int32_t FileSave(FAT_FILE* File);
```

2.13 FileClose

@brief Closes a stream.

@param File Pointer to a FILE object that specifies an output stream.

@return fclose returns 0 if the stream is successfully closed.

```
int32_t FileClose(FAT_FILE* File);
```

2.14 FileGetLongName

获取指定文件的长文件名。

长文件名 LongFileName 不能小于 GetMaxLength

获取到长文件名，返回 TRUE。

无长文件名，返回 FALSE。

```
bool FileGetLongName(FAT_FILE* File, uint8_t* LongFileName, uint8_t GetMaxLength);
```

2.15 FileFlush

文件清空。

成功:返回 TRUE，失败:返回 FALSE。

```
bool FileFlush(FAT_FILE* File);
```

2.16 FileDelete

文件删除。成功:返回 TRUE，失败:返回 FALSE。

```
bool FileDelete(FAT_FILE* File);
```

2.17 FileCreate

按文件名称在指定文件夹中新建一个文件。

目标文件信息填充到 File 所指的结构体。

FileName[]为短文件名，例如："123.TXT", "ABC123.MP3"。

短文件名长度不能超过 8+3 字节，短文件名中不能同时出现大小写字母。

```
bool FileCreate(FAT_FILE* File, FOLDER* Folder, uint8_t* FileName);
```

2.18 FileCreateByLfName

按长文件名称在指定文件夹中新建一个文件。

目标文件信息填充到 File 所指的结构体。

FileName[]为长文件名

Folder 不能为空，FileName 为 Unicode 16 编码

```
bool FileCreateByLfName(FAT_FILE* File, FOLDER* Folder, uint8_t* LongFileName, uint8_t Len);
```

2.19 FSFormat

格式化当前设备

```
bool FSFormat(void);
```

2.20 FileSetTime

修改文件的时间信息。

CreateTime: 创建时间，精度为 1 秒

ModifyTime: 最后修改时间，精度为 2 秒

AccessTime: 最后访问时间，只有日期，忽略时间部分

如果某个时间信息传入的指针为 NULL，则表示不改变该时间信息

成功:返回 TRUE， 失败:返回 FALSE。

```
bool FileSetTime(FAT_FILE* File, FILE_TIME* CreateTime, FILE_TIME* ModifyTime, FILE_TIME* AccessTime);
```

2.21 FileGetTime

获取文件的时间信息。

CreateTime: 创建时间，精度为 1 秒

ModifyTime: 最后修改时间，精度为 2 秒

AccessTime: 最后访问时间，只有日期，忽略时间部分

如果某个时间信息传入的指针为 NULL，则表示不获取该时间信息

成功:返回 TRUE， 失败:返回 FALSE。

```
bool FileGetTime(FAT_FILE* File, FILE_TIME* CreateTime, FILE_TIME* ModifyTime, FILE_TIME*
AccessTime);
```

【文件夹操作相关】

2.22 FolderOpenByNum

按文件夹序号打开文件夹

打开成功：返回 TRUE，目标文件夹信息填充到 Folder 所指的结构体。

打开失败：返回 FALSE。

ParentFolder == NULL: 打开整个设备上的第 FolderNum 个文件夹。

ParentFolder != NULL: 打开 ParentFolder 文件夹中的第 FolderNum 个文件夹。

```
bool FolderOpenByNum(FOLDER* Folder, FOLDER* ParentFolder, uint16_t FolderNum);
```

2.23 FolderOpenByValidNum

按文件夹有效序号（滤除空文件夹后的序号）打开文件夹。

函数功能类似于 FolderOpenByNum()。

```
bool FolderOpenByValidNum(FOLDER* Folder, FOLDER* ParentFolder, uint16_t ValidFolderNum);
```

2.24 FolderOpenByName

按文件夹名称打开文件夹。

打开成功：返回 TRUE，目标文件夹信息填充到 Folder 所指的结构体。

打开失败：返回 FALSE。

ParentFolder == NULL: 打开根目录中的 FolderName 文件夹。

ParentFolder != NULL: 打开 ParentFolder 文件夹中的 FolderName 文件夹。

FolderName[]长度不大于 8 字节，根目录名称为"\"

```
bool FolderOpenByName(FOLDER* Folder, FOLDER* ParentFolder, uint8_t* FolderName);
```

2.25 FolderGetLongName

获取指定文件夹的长文件名。

长文件名最长为 66 个字节，所以 LongFileName[] 数组至少要有 66 个字节，否则将会导致不可预料的错误。

获取到长文件名，返回 TRUE。

无长文件名，返回 FALSE。

```
bool FolderGetLongName(FOLDER* Folder, uint8_t* LongFileName, uint8_t GetMaxLength);  
    //LongFileName[]: 66 Bytes
```

2.26 FolderCreate

新建文件夹。

在 ParentFolder 文件夹中新建一个名称为 FolderName[] 的文件夹。

ParentFolder 父文件夹指针不能为 NULL。

返回 TRUE：创建成功，新创建的文件夹信息填充到 Folder 所指结构体中。

返回 FALSE：创建失败。

```
bool FolderCreate(FOLDER* Folder, FOLDER* ParentFolder, uint8_t* FolderName);
```

【文件系统操作相关】

2.27 FSInit

文件系统初始化，分析分区表，预搜索统计文件系统有关信息

DevicID 设备号： DEV_ID_USB、 DEV_ID_SD

```
bool FSInit(uint8_t DevicID)
```

2.28 FSDeInit

文件系统去初始化

DevicID 设备号： DEV_ID_USB、 DEV_ID_SD

```
bool FSDeInit(uint8_t DevicID)
```

3 典型应用

- a. 检测设备 初始化文件系统

存储设备支持 SD 卡(含 TF 卡、MMC 卡)和 U 盘两种，使用 IsCardLink()、IsUDiskLink()检测设备是否连接(不能同时使用)。调用 FSInit 对文件系统进行初始化成功后就可以正常使用相关 API 接口进行响应操作。

b. 执行应用相关操作

参照上文 API 即可对文件、文件夹进行创建、打开、删除、修改操作。