XR806 VFS 中间件

开发指南

版本号：1.0

发布时间：2021-02-20

版本历史

版本	日期	责任人	版本描述
1.0	2021-02-20	AWA 1680	创建文档。

目录

表格目录

1 前言

1.1 文档简介

本文档介绍了 XR806 平台上 VFS 模块的使用方法。

1.2 目标读者

使用 XR806 SDK 的开发人员。

1.3 适用范围

此文档适用于 XR806 SDK，支持 XR806 系列芯片产品。

1.4 文档约定

1.4.1 标志说明

本文档采用各种醒目的标志来表示在操作过程中应该特别注意的地方，这些标志的含义如下：

标识	说明
警告	该标志后的说明应给予格外关注，如果不遵守，可能会导致人员受伤或死亡。
注意	提醒操作中应注意的事项。不当的操作可能会损坏器件，影响可靠性、降低性能等。
说明	为准确理解文中指令、正确实施操作而提供的补充或强调信息。
窍门	一些容易忽视的小功能、技巧。了解这些功能或技巧能帮助解决特定问题或者节省操作时间。

1.4.2 地址与数据描述方法约定

本文档在描述地址、数据时遵循如下约定：

符号	例子	说明
0x	0x0200，0x79	地址或数据以 16 进制表示。
0b	0b010，0b00 000 111	数据采用二进制表示(寄存器描述除外)。
X	00X，XX1	数据描述中，X 代表 0 或 1。 例如，00X 代表 000 或 001；XX1 代表 001，011，101 或 111。

1.4.3 数值单位约定

本文档在描述数据容量（如 NAND 容量）时，单位词头代表的是 1024 的倍数；描述频率、数据速率等时则代表的是 1000 的倍数。具体如下：

类型	符号	对应数值
数据容量（如 NAND 容量）	1 K	1024
	1 M	1 048 576
	1 G	1 073 741 824
频率，数据速率等	1 k	1000
	1 M	1 000 000
	1 G	1 000 000 000

2 概述

2.1 背景说明

为满足不同用户对不同类型文件系统的使用需求，XR806 SDK 提供了 VFS（Virtual File System）中间件模块，该模块支持与 littleFS、SPIFFS、FatFS 等文件系统对接，方便用户灵活选择。

VFS 中将文件系统分为两大类：Flash 上的文件系统（如 littleFS、SPIFFS）和 SD 卡上的文件系统（如 FatFS），通过不同的 URL（Uniform Resource Locator）前缀来判断用户实际操作的文件系统类型。

littleFS、SPIFFS、FatFS 三者均为嵌入式设备常用的文件系统，其中，由于 littleFS 具有较好的掉电安全性能、支持磨损均衡以及活跃的社区状态，因此 XR806 平台主要以使用 littleFS 文件系统为主。

2.2 规格特性

VFS 中间件模块支持以下功能特点：

- 支持 littleFS、SPIFFS、FatFS 等多种开源文件系统。
- 支持统一的接口访问不同的文件系统。
- 支持通过 URL 前缀来判断实际操作的文件系统。
- 当前不支持同时使用多个同类型文件系统。

2.3 文件位置

以 SDK 包为根目录，本中间件涉及到的主要文件位置如下。

表 2-1 VFS 中间件的文件位置

组件名	文件分类	文件位置
VFS	源码文件	./src/fs/
	头文件	./include/fs/vfs.h
	示例代码	./project/common/cmd/cmd_fs.c

表 2-2 VFS 中间件的文件说明

文件名	文件说明
vfs.c	VFS 模块的主文件
vfs_lfs.c	VFS 模块与 littleFS 接口对接的文件
vfs_spiffs.c	VFS 模块与 SPIFFS 接口对接的文件
vfs_fatfs.c	VFS 模块与 FatFS 接口对接的文件

说明

XR806 SDK 可在以下 GitHub 仓库获取：https://github.com/XradioTech/xr806_sdk.git

3 应用说明

3.1 应用简述

VFS 中间件模块已经内嵌到 XR806 SDK，应用步骤如下：

1. 在 menuconfig 中确认/检查所需要使用的文件系统类型与基本参数配置（文件系统类型选择与配置可参考“3.2 配置说明”章节）。
2. 在用户应用代码中添加 VFS 模块头文件（参见“2.3 文件位置”章节），XR806 平台初始化后，调用 VFS 接口进行使用（接口使用参见“3.3 接口说明”章节）。

3.2 配置说明

打开 VFS 功能，可按照如下配置：执行 make menuconfig，选中并进入 filesystem support，在 FileSystem Type Select 中选择需要使用的文件系统类型，如 littleFS。

图 3-1 打开 XR806 文件系统功能

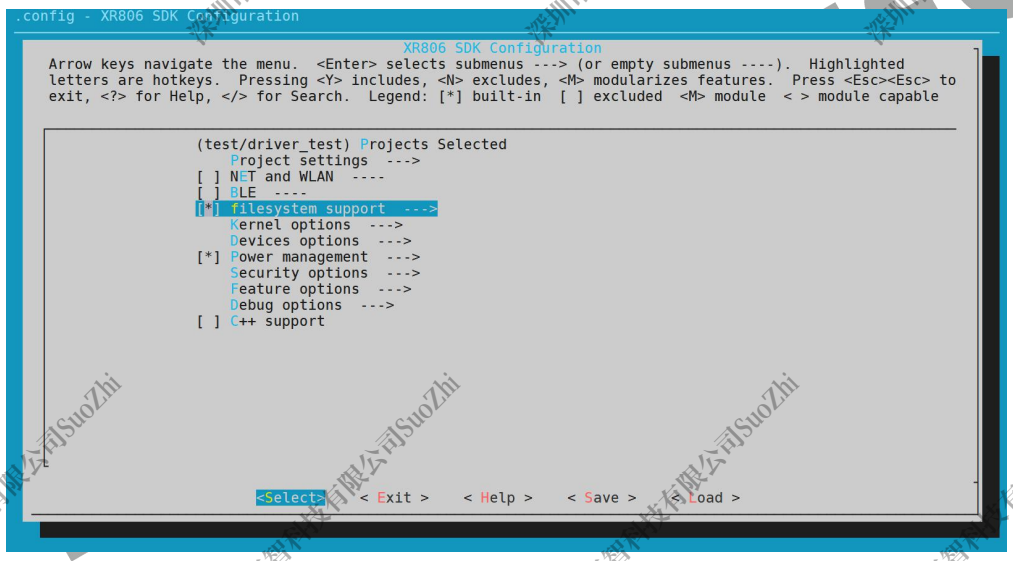
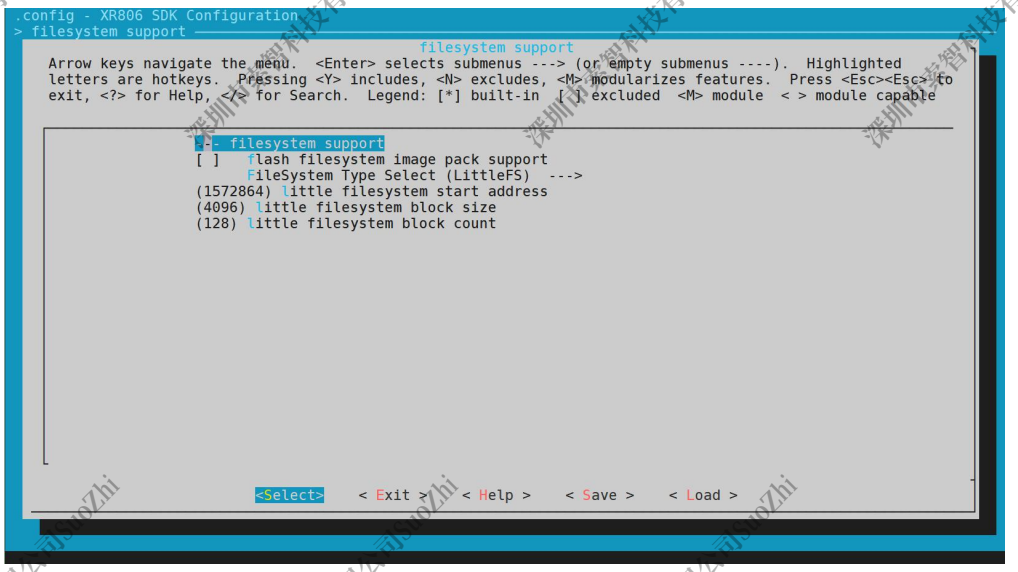


图 3-2 选择文件系统配置



按照 menuconfig 中的提示可选择用户需要的文件系统类型和相关配置，对应的 littleFS 与 SPIFFS 配置具体说明如下：

表 3-1 XR806 VFS-littleFS 配置列表

配置项	配置说明
文件系统起始地址（littleFS）	设置说明： 此项配置 littleFS 在 Flash 上的起始物理地址 设置位置： menuconfig 中 filesystem support 选项下，条目名称为“little filesystem start address” 设置方式： 回车进入此条目，输入十进制值（默认为 1572864，即 Flash 上 1.5M 处）
文件系统所使用的 block 大小（littleFS）	设置说明： 此项配置 littleFS 所使用的 block 大小 设置位置： menuconfig 中 filesystem support 选项下，条目名称为“little filesystem block size” 设置方式： 回车进入此条目，输入十进制值（默认为 4096，即 4KByte）
文件系统所使用的 block 数量（littleFS）	设置说明： 此项配置 littleFS 总共使用的 block 数量 设置位置： menuconfig 中 filesystem support 选项下，条目名称为“little filesystem block

配置项	配置说明
	count” 设置方式： 回车进入此条目，输入十进制值（默认为 128，由于 block size 默认为 4K，因此 littleFS 默认使用 512K 的 Flash 物理空间）

表 3-2 XR806 VFS-SPIFFS 配置列表

配置项	配置说明
文件系统起始地址（SPIFFS）	设置说明： 此项配置 SPIFFS 在 Flash 上的起始物理地址 设置位置： menuconfig 中 filesystem support 选项下，条目名称为“spiffs filesystem start address” 设置方式： 回车进入此条目，输入十进制值（默认为 1572864，即 Flash 上 1.5M 处）
文件系统所使用的 block 大小（SPIFFS）	设置说明： 此项配置 SPIFFS 所使用的 block 大小 设置位置： menuconfig 中 filesystem support 选项下，条目名称为“spiffs filesystem block size” 设置方式： 回车进入此条目，输入十进制值（默认为 4096，即 4KByte）
文件系统占用 Flash 物理大小（SPIFFS）	设置说明： 此项配置 SPIFFS 所占用的 Flash 物理空间总大小 设置位置： menuconfig 中 filesystem support 选项下，条目名称为“spiffs filesystem physical size” 设置方式： 回车进入此条目，输入十进制值（默认为 131072，即 128KB 的 Flash 物理空间）

注意

1) menuconfig 中 SPIFFS 的相关配置只有在“FileSystem Type Select”被选择为“SPIFFS”时才会出现且生效。

2) 条目“flash filesystem image pack support”为空文件系统自动打包功能，默认关闭，详细可参考《XR806_文件系统工具_使用指南》文档中的对应章节。

XR806 VFS 模块也可设置调试开关，如下表所示。

表 3-3 XR806 VFS 模块配置列表

配置项	配置说明
开启或关闭调试宏	设置说明： 此项配置使能/关闭调试信息 设置位置： sdk/include/fs/vfs.h 设置方式： 1.定义宏 VFS_DBG_ON 的值来启动调试，如打开调试信息： #define VFS_DBG_ON 1 2.定义宏 VFS_ASSERT_ON 的值来启动断言，如打开断言信息： #define VFS_ASSERT_ON 1

3.3 接口说明

XR806 VFS 模块提供 1 套对文件系统操作的通用接口，位于 SDK/include/fs/vfs.h 文件中，各接口的简要说明如下表所示。

表 3-4 XR806 VFS 模块接口简介

接口名	简要说明
文件系统控制接口	包括 4 个接口函数，用于 VFS 的初始化、挂载、卸载与格式化，属于系统平台内部接口。
基本操作接口	包括 16 个接口函数，用于用户对文件、文件夹的基本操作，属于外部接口。

上述抽象接口分别于对应的 vfs.c、fs_ctrl.c（SDK/project/common/framework）文件中实现，详细说明如下。

3.3.1 文件系统控制接口

3.3.1.1 fs_ctrl_init

表 3-5 fs_ctrl_init 接口函数说明

信息项	说明
原型	int fs_ctrl_init(void);
功能	初始化虚拟文件系统，内部会对 VFS 模块进行初始化，包括 VFS 列表初始化以及 VFS 注册。

信息项	说明
	若 menuconfig 中文件系统功能打开，则此接口会在 platform_init_level2 中被调用，因此用户代码中若进行了 XR806 平台初始化，则无需自行调用此接口。
参数	无
返回值	0：初始化成功 -1：初始化失败

3.3.1.2 fs_ctrl_mount

表 3-6 fs_ctrl_mount 接口函数说明

信息项	说明
原型	int fs_ctrl_mount(enum fs_mnt_dev_type dev_type, uint32_t dev_id);
功能	配置文件系统并进行挂载。 若 menuconfig 中文件系统功能打开，则此接口会在 platform_init_level2 中被调用，因此用户代码中若进行了 XR806 平台初始化，则系统上电后会自动挂载文件系统。
参数	enum fs_mnt_dev_type dev_type 含义解释：用于区分挂载的设备类型，SD 卡或 Flash 使用说明：fs_mnt_dev_type 为枚举值，具体含义如下 FS_MNT_DEV_TYPE_SDCARD = 0（表示挂载到 SD 卡设备） FS_MNT_DEV_TYPE_FLASH = 1（表示挂载到 Flash 设备） uint32_t dev_id 含义解释：设备 ID 号，用于指定文件系统挂载到该设备类型下的哪一个设备 使用说明：对于 Flash 设备类型，SDK 当前只支持挂载到一块 Flash，其设备号默认选择 0，下同。
返回值	0：挂载成功 -1：挂载失败

注意

XR806 SDK 暂不支持挂载多个同类型（SD 卡类型与 Flash 类型）的文件系统，例如 Flash 上挂载的文件系统只能是 littleFS 和 SPIFFS 二选一。

3.3.1.3 fs_ctrl_unmount

表 3-7 fs_ctrl_unmount 接口函数说明

信息项	说明
原型	int fs_ctrl_unmount(enum fs_mnt_dev_type dev_type, uint32_t dev_id);
功能	卸载文件系统，卸载之前需要保证文件系统已被挂载，否则会卸载失败。

信息项	说明
参数	enum fs_mnt_dev_type dev_type 含义解释：用于区分挂载的设备类型，SD 卡或 Flash 使用说明：fs_mnt_dev_type 为枚举值，具体含义如下 FS_MNT_DEV_TYPE_SDCARD = 0（表示挂载到 SD 卡的设备） FS_MNT_DEV_TYPE_FLASH = 1（表示挂载到 Flash 的设备） uint32_t dev_id 含义解释：设备 ID 号，用于指定需要卸载的文件系统所属的设备号 使用说明：卸载时的设备号与挂载时所用的设备号一致即可
返回值	0：卸载成功 -1：卸载失败

3.3.1.4 fs_ctrl_format

表 3-8 fs_ctrl_format 接口函数说明

信息项	说明
原型	int fs_ctrl_format(enum fs_mnt_dev_type dev_type, uint32_t dev_id);
功能	格式化文件系统，格式化之前需要保证文件系统已被挂载（目的是为了获取当前文件系统的参数，格式化过程中，接口内部会先卸载文件系统再进行内容移除），否则会格式化失败。
参数	enum fs_mnt_dev_type dev_type 含义解释：用于区分挂载的设备类型，SD 卡或 Flash 使用说明：fs_mnt_dev_type 为枚举值，具体含义如下 FS_MNT_DEV_TYPE_SDCARD = 0（表示挂载到 SD 卡的设备） FS_MNT_DEV_TYPE_FLASH = 1（表示挂载到 Flash 的设备） uint32_t dev_id 含义解释：设备 ID 号，用于指定需要格式化的文件系统所属的设备号 使用说明：格式化时的设备号与挂载时所用的设备号一致即可
返回值	0：格式化成功 -1：格式化失败

3.3.2 基本操作接口

3.3.2.1 vfs_open

表 3-9 vfs_open 接口函数说明

信息项	说明
原型	vfs_file_t *vfs_open(const char *path, int flags);
功能	打开或创建文件，返回一个 VFS 文件操作句柄。

信息项	说明
	文件打开一次后，即可通过 VFS 文件句柄对此文件进行各类操作。
参数	const char *path 含义解释：文件的绝对路径 使用说明：对于 SD 卡设备上的文件系统，路径需要加上“sdcard/”前缀；对于 Flash 设备上的文件系统，路径需要加上“data/”前缀，两种前缀均包含根目录信息“/”，“data”与“sdcard”标识则主要用于文件系统类型的识别与区分，下同。 int flags 含义解释：打开文件的模式/标志 使用说明：各模式/标志说明如下 VFS_RDONLY 以只读的方式打开文件 VFS_WRONLY 以只写的方式打开文件 VFS_RDWR 以可读可写的方式打开文件 VFS_CREAT 如果文件不存在则创建文件 VFS_EXCL 如果文件已存在，则打开失败 VFS_TRUNC 如果文件已存在，则将其长度截断为 0 VFS_APPEND 每次写时都会将光标移动至文件末尾
返回值	0（空指针）：打开失败 非 0（VFS 文件操作句柄）：成功，句柄为 vfs_file_t 结构体指针类型，可参见表 3-10 vfs_file_t 结构体的成员说明。

注意

XR806 SDK 目前只支持绝对路径，不支持相对路径，所有涉及路径的操作都需要加上“data/”或“sdcard/”前缀，如 vfs_open、vfs_stat、vfs_unlink、vfs_rename、vfs_opendir、vfs_mkdir、vfs_rmdir 等接口。

说明

VFS 模块中，文件名最大长度为 255：

1) 对于 littleFS，文件名长度从目录结束符“/”后开始计算，如从“data/”前缀后开始，可成功创建一个名称长度为 255 字节的文件。

2) 对于 SPIFFS，由于其内部不支持真正的文件夹，因此文件名长度从“data”后开始计算，后面的“/”也会算作一个字节，且内部支持的文件名称最大总长度为 31 字节。用户使用时需注意。

3) 不论文件名后缀为.txt 还是.xxx，均会被算作文件名称长度的一部分。

表 3-10 vfs_file_t 结构体的成员说明

成员项	说明
const struct vfs_file_ops *ops;	含义解释：用于 VFS 文件操作接口与底层文件系统操作接口对接 使用说明：ops 成员为 vfs_file_ops 结构体类型，可参见表 3-11 struct vfs_file_ops 结构体的成员说明

表 3-11 struct vfs_file_ops 结构体的成员说明

函数指针	说明
(*vfs_open)()	含义解释：VFS 内部 open 操作接口，用于和底层 open 接口对接，供对外的 vfs_open 接口调用 使用说明：N/A
(*vfs_close)()	含义解释：VFS 内部 close 操作接口，用于和底层 close 接口对接，供对外的 vfs_close 接口调用 使用说明：N/A
(*vfs_read)()	含义解释：VFS 内部 read 操作接口，用于和底层 read 接口对接，供对外的 vfs_read 接口调用 使用说明：N/A
(*vfs_write)()	含义解释：VFS 内部 write 操作接口，用于和底层 write 接口对接，供对外的 vfs_write 接口调用 使用说明：N/A
(*vfs_seek)()	含义解释：VFS 内部 seek 操作接口，用于和底层 seek 接口对接，供对外的 vfs_seek 接口调用 使用说明：N/A
(*vfs_sync)()	含义解释：VFS 内部 sync 操作接口，用于和底层 sync 接口对接，供对外的 vfs_sync 接口调用 使用说明：N/A
(*vfs_stat)()	含义解释：VFS 内部 stat 操作接口，用于和底层 stat 接口对接，供对外的 vfs_stat 接口调用 使用说明：N/A
(*vfs_unlink)()	含义解释：VFS 内部 unlink 操作接口，用于和底层 unlink 接口对接，供对外的 vfs_unlink 接口调用 使用说明：N/A
(*vfs_rename)()	含义解释：VFS 内部 rename 操作接口，用于和底层 rename 接口对接，供对外的 vfs_rename 接口调用 使用说明：N/A
(*vfs_tell)()	含义解释：VFS 内部 tell 操作接口，用于和底层 tell 接口对接，供对外的 vfs_tell 接口调用

函数指针	说明
	使用说明：N/A
(*vfs_size)()	含义解释：VFS 内部 size 操作接口，用于和底层 size 接口对接，供对外的 vfs_size 接口调用 使用说明：N/A
(*vfs_opendir)()	含义解释：VFS 内部 opendir 操作接口，用于和底层 opendir 接口对接，供对外的 vfs_opendir 接口调用 使用说明：N/A
(*vfs_closedir)()	含义解释：VFS 内部 closedir 操作接口，用于和底层 closedir 接口对接，供对外的 vfs_closedir 接口调用 使用说明：N/A
(*vfs_readdir)()	含义解释：VFS 内部 readdir 操作接口，用于和底层 readdir 接口对接，供对外的 vfs_readdir 接口调用 使用说明：N/A
(*vfs_mkdir)()	含义解释：VFS 内部 mkdir 操作接口，用于和底层 mkdir 接口对接，供对外的 vfs_mkdir 接口调用 使用说明：N/A
(*vfs_rmdir)()	含义解释：VFS 内部 rmdir 操作接口，用于和底层 rmdir 接口对接，供对外的 vfs_rmdir 接口调用 使用说明：N/A
(*vfs_release)()	含义解释：VFS 内部 release 操作接口，用于和底层 release 接口对接，供 VFS 模块内部调用，以释放对应的文件/文件夹操作句柄 使用说明：N/A

3.3.2.2 vfs_close

表 3-12 vfs_close 接口函数说明

信息项	说明
原型	int vfs_close(vfs_file_t *vfs);
功能	关闭文件，需要先打开文件。 vfs_close 接口一般与 vfs_open 接口成对使用，打开文件，进行操作完毕后，请记得关闭。
参数	vfs_file_t *vfs 含义解释：文件操作句柄 使用说明：文件操作句柄只能通过 vfs_open 接口的返回值获得
返回值	0：关闭成功 负值：关闭失败，负值的绝对值为错误码，详细可参见 SDK/include/libc/errno.h

3.3.2.3 vfs_read

表 3-13 vfs_read 接口函数说明

信息项	说明
原型	int vfs_read(vfs_file_t *vfs, void *buffer, unsigned int len);
功能	从当前光标处读取指定长度的文件内容（需要先打开文件），读取后光标的位置也会跟着向后移动指定长度
参数	vfs_file_t *vfs 含义解释：文件操作的句柄 使用说明：文件操作句柄只能通过 vfs_open 接口的返回值获得 void *buffer 含义解释：读取出数据的存储地址 使用说明：N/A unsigned int len 含义解释：读取长度 使用说明：读取文件内容时，用户需要注意当前光标的位置，一般打开文件后，光标位置从 0 开始，进行读/写操作后光标位置会移动，若文件未关闭，继续次进行读/写时，光标也会从最新的位置开始。
返回值	非负值：读取成功，为读取到的字节数 负值：读取失败，负值的绝对值为错误码，详细可参见 SDK/include/libc/errno.h

3.3.2.4 vfs_write

表 3-14 vfs_write 接口函数说明

信息项	说明
原型	int vfs_write(vfs_file_t *vfs, const void *buffer, unsigned int len);
功能	从当前光标处向文件内写入指定长度内容（需要先打开文件），写入后，光标的位置也会随着向后移动指定长度
参数	vfs_file_t *vfs 含义解释：文件操作的句柄 使用说明：文件操作句柄只能通过 vfs_open 接口的返回值获得 void *buffer 含义解释：需要被写入数据的地址 使用说明：N/A unsigned int len 含义解释：写入长度 使用说明：写入文件内容时，用户需要注意当前光标的位置，进行读/写操作后光标位置会移动，若文件未关闭，继续次进行读/写时，光标也会从最新的位置开始。

信息项	说明
返回值	非负值：写入成功，为写入的字节数 负值：写入失败，负值的绝对值为错误码，详细可参见 SDK/include/libc/errno.h

3.3.2.5 vfs_seek

表 3-15 vfs_seek 接口函数说明

信息项	说明
原型	int vfs_seek(vfs_file_t *vfs, int64_t offset, int whence);
功能	移动当前光标到指定位置（需要先打开文件）
参数	vfs_file_t *vfs 含义解释：文件操作的句柄 使用说明：文件操作句柄只能通过 vfs_open 接口的返回值获得 int64_t offset 含义解释：光标的偏移量 使用说明：此偏移量是相对于参数 whence 而言，光标最终的位置等于 whence + offset int whence 含义解释：光标移动的基准 使用说明：基准标志说明如下 SEEK_SET 表示光标从文件的起始处开始移动，移动长度为 offset（一般为正数） SEEK_CUR 表示光标从当前位置开始移动，移动长度为 offset（可正可负） SEEK_END 表示光标从文件的结束处开始移动，移动长度为 offset（一般为负数）
返回值	0：移动成功 负值：移动失败，负值的绝对值为错误码，详细可参见 SDK/include/libc/errno.h

注意

无论光标怎样移动，建议光标最终的位置不要小于 0 且不要大于文件本身的大小，否则可能出错（例如 SPIFFS 中就不支持光标移动后的位置大于文件自身大小）。

3.3.2.6 vfs_sync

表 3-16 vfs_sync 接口函数说明

信息项	说明
原型	int vfs_sync(vfs_file_t *vfs);
功能	同步文件的内容到存储设备中，即强制将缓存 buffer 中的数据刷新到存储设备中（Flash 或 SD 卡）

信息项	说明
参数	vfs_file_t *vfs 含义解释：文件操作的句柄 使用说明：文件操作句柄只能通过 vfs_open 接口的返回值获得
返回值	0：同步成功 负值：同步失败，负值的绝对值为错误码，详细可参见 SDK/include/libc/errno.h

3.3.2.7 vfs_tell

表 3-17 vfs_tell 接口函数说明

信息项	说明
原型	int vfs_tell(vfs_file_t *vfs);
功能	获取光标当前的位置（需要先打开文件）
参数	vfs_file_t *vfs 含义解释：文件操作的句柄 使用说明：文件操作句柄只能通过 vfs_open 接口的返回值获得
返回值	非负值：获取成功，为光标当前的位置 负值：获取失败，负值的绝对值为错误码，详细可参见 SDK/include/libc/errno.h

3.3.2.8 vfs_size

表 3-18 vfs_size 接口函数说明

信息项	说明
原型	int vfs_size(vfs_file_t *vfs);
功能	获取此文件的大小（需要先打开文件）
参数	vfs_file_t *vfs 含义解释：文件操作的句柄 使用说明：文件操作句柄只能通过 vfs_open 接口的返回值获得
返回值	非负值：获取成功，为文件的大小（单位：字节） 负值：获取失败，负值的绝对值为错误码，详细可参见 SDK/include/libc/errno.h

3.3.2.9 vfs_stat

表 3-19 vfs_stat 接口函数说明

信息项	说明
原型	int vfs_stat(const char *path, struct vfs_info *info);
功能	获取一个路径名的信息，包括路径的类型（文件/文件夹），若为文件，还会获取到文件的大小，存储在 info 中。
参数	const char *path

信息项	说明
	含义解释：文件或文件夹的绝对路径 使用说明：对于 SD 卡设备上的文件系统，路径需要加上“sdcard/”前缀；对于 Flash 设备上的文件系统，路径需要加上“data/”前缀。 struct vfs_info *info 含义解释：用于存储获取到的文件/目录信息 使用说明：struct vfs_info 的成员说明可参见“表 3-20 struct vfs_info 结构体的成员说明”
返回值	0：获取成功 负值：获取失败，负值的绝对值为错误码，详细可参见 SDK/include/libc/errno.h

表 3-20 struct vfs_info 结构体的成员说明

成员项	说明
VFS_FILE_TYPE type;	含义解释：路径类型 使用说明：VFS_FILE_TYPE 为枚举值，具体含义如下 VFS_TYPE_FILE = 0 表示此路径为文件 VFS_TYPE_DIR = 1 表示此路径为文件夹
int size;	含义解释：用于存储文件的大小信息 使用说明：若此路径类型为文件，则会将文件的大小信息存储在成员变量 size 中
char name[VFS_NAME_MAX + 1];	含义解释：用于存放文件名称 使用说明：调用 vfs_stat 接口后，不会向 name 成员中存储文件名，仅仅调用 vfs_readdir 时，会将读取到的文件名存储在 name 成员变量中。

3.3.2.10 vfs_unlink

表 3-21 vfs_unlink 接口函数说明

信息项	说明
原型	int vfs_unlink(const char *path);
功能	从文件系统中删除一个文件
参数	const char *path 含义解释：需要被删除文件的绝对路径 使用说明：对于 SD 卡设备上的文件系统，路径需要加上“sdcard/”前缀；对于 Flash 设备上的文件系统，路径需要加上“data/”前缀。
返回值	0：删除成功

信息项	说明
	负值：删除失败，负值的绝对值为错误码，详细可参见 SDK/include/libc/errno.h

3.3.2.11 vfs_rename

表 3-22 vfs_rename 接口函数说明

信息项	说明
原型	int vfs_rename(const char *old_name, const char *new_name);
功能	对文件/文件夹进行重命名（用户在进行文件重命名之前，尽可能先关闭此文件）
参数	const char *old_name 含义解释：旧文件/文件夹路径（绝对路径） 使用说明：对于 SD 卡设备上的文件系统，路径需要加上“sdcard/”前缀；对于 Flash 设备上的文件系统，路径需要加上“data/”前缀。 const char *new_name 含义解释：新文件/文件夹路径（绝对路径） 使用说明：对于 SD 卡设备上的文件系统，路径需要加上“sdcard/”前缀；对于 Flash 设备上的文件系统，路径需要加上“data/”前缀。
返回值	0：重命名成功 负值：重命名失败，负值的绝对值为错误码，详细可参见 SDK/include/libc/errno.h

说明

对于 littleFS：

1. 若重命名文件为一个已经存在的文件名称，例如将 file1.txt 重命名为 file2.txt 时，若 file2.txt 已经存在，则无论 file2.txt 是否为空，file2.txt 文件都会被删除(unlink)，新的 file2.txt 文件内容为重命名前的 file1.txt 中的内容。
2. 若重命名文件夹时，目标文件夹名称已存在，则 littleFS 内部会进行判断已存在的目标文件夹是否为空，如果为空则重命名成功，如果不为空，则重命名失败。

说明

对于 SPIFFS，不支持将文件重命名为一个已经存在的文件名称，否则会报错。

3.3.2.12 vfs_opendir

表 3-23 vfs_opendir 接口函数说明

信息项	说明
原型	vfs_file_t *vfs_opendir(const char *path);

信息项	说明
功能	打开一个文件夹，返回一个 VFS 文件夹操作句柄
参数	const char *path 含义解释：目录的绝对路径 使用说明：对于 SD 卡设备上的文件系统，路径需要加上“sdcard/”前缀；对于 Flash 设备上的文件系统，路径需要加上“data/”前缀，均表示根目录“/”。
返回值	0（空指针）：打开失败 非 0（文件夹操作句柄）：成功，文件夹操作句柄为 vfs_file_t 结构体指针类型，可参见表 3-10 vfs_file_t 结构体的成员说明。

注意

vfs_opendir 获得的文件夹句柄只能用于文件夹操作（如用于 vfs_readdir、vfs_closedir 等接口），而 vfs_open 获得的文件句柄只能用于文件操作（vfs_close、vfs_read、vfs_write、vfs_seek、vfs_sync、vfs_tell、vfs_size 等接口），两者不能混用。

说明

对于 littleFS 与 FatFS 等支持真实目录的文件系统，在 VFS 中打开目录“data/”表示打开根目录“/”，打开目录“data/book/”表示打开根目录“/”下的“book”目录。

说明

对于 SPIFFS，不支持真实的目录，例如创建文件“data/book/text1.txt”时，实际上是创建了一个名称为“/book/text1.txt”的文件，并不存在根目录“/”与次目录“book”。因此，用户若选择使用 SPIFFS 文件系统，则需要谨慎使用其目录功能。

3.3.2.13 vfs_readdir

表 3-24 vfs_readdir 接口函数说明

信息项	说明
原型	int vfs_readdir(vfs_file_t *vfs, struct vfs_info *info);
功能	读取文件夹中下一个文件的信息，并存储到 info 指向的区域中（需要先打开文件夹，获取到 VFS 文件夹句柄） 读取后，指向文件的指针也会移动到下一个文件处，再次读取文件信息则是从下一个文件开始
参数	vfs_file_t *vfs 含义解释：文件夹操作句柄 使用说明：文件夹操作句柄只能通过 vfs_opendir 接口的返回值获得
返回值	0：读取文件信息成功

信息项	说明
	负值：读取失败，负值的绝对值为错误码，详细可参见 SDK/include/libc/errno.h

3.3.2.14 vfs_closedir

表 3-25 vfs_closedir 接口函数说明

信息项	说明
原型	int vfs_closedir(vfs_file_t *vfs);
功能	关闭文件夹
参数	vfs_file_t *vfs 含义解释：文件夹操作句柄 使用说明：文件夹操作句柄只能通过 vfs_opendir 接口的返回值获得
返回值	0：关闭成功 负值：关闭失败，负值的绝对值为错误码，详细可参见 SDK/include/libc/errno.h

3.3.2.15 vfs_mkdir

表 3-26 vfs_mkdir 接口函数说明

信息项	说明
原型	int vfs_mkdir(const char *path);
功能	创建一个目录/文件夹，创建前需保证其父目录存在
参数	const char *path 含义解释：需要创建目录的绝对路径 使用说明：对于 SD 卡设备上的文件系统，路径需要加上“sdcard/”前缀；对于 Flash 设备上的文件系统，路径需要加上“data/”前缀
返回值	0：创建成功 负值：创建失败，负值的绝对值为错误码，详细可参见 SDK/include/libc/errno.h

说明

littleFS 不支持跨级创建目录，创建当前目录时，需要保证其上一级已存在。

SPIFFS 不支持创建目录。

3.3.2.16 vfs_rmdir

表 3-27 vfs_rmdir 接口函数说明

信息项	说明
原型	int vfs_rmdir(const char *path);
功能	删除一个目录/文件夹

信息项	说明
参数	const char *path 含义解释：需要删除目录的绝对路径 使用说明：对于 SD 卡设备上的文件系统，路径需要加上“sdcard/”前缀；对于 Flash 设备上的文件系统，路径需要加上“data/”前缀
返回值	0：删除成功 负值：删除失败，负值的绝对值为错误码，详细可参见 SDK/include/libc/errno.h

说明

SPIFFS 不支持删除目录。

4 示例说明

VFS 接口用于对底层文件系统进行统一的文件/文件夹操作，下面以文件的读写以及文件夹的读与创建为例，简要说明 VFS 接口的使用。

4.1 示例简介

VFS 接口使用示例演示的目的是简要介绍 VFS 接口的基本使用方法（文件/文件夹的创建、读、写、关闭等操作），通过串口打印，在控制台上输出对文件/文件夹的操作状态信息。

4.1.1 获取方式

VFS 接口使用示例基于 hello_demo 工程，使用 cmd_fs.c 文件中的示例代码进行演示，hello_demo 工程位于 XR806 SDK 的/project/demo/hello_demo 目录，cmd_fs.c 示例代码位于 SDK/project/common/cmd 目录。

说明

XR806 SDK 可在以下 GitHub 仓库获取：https://github.com/XradioTech/xr806_sdk.git

4.1.2 准备工作

本示例的硬件准备有如下。

1. 评估板：运行示例工程代码。
2. 串口线：连接评估板的 UART0 插针，用于 console 控制台的输入输出。
3. PC 机：用于镜像烧录和 console 控制的输入输出。

VFS 示例的软件准备，包括烧写工具、代码编译和烧写操作，请参见《XR806_SDK_快速入门指南》。

基于 hello_demo 工程，需要准备的代码如下：

1. 在 command.c 文件的 g_main_cmds 数组中添加一行代码以支持 fs 命令

```
static const struct cmd_data g_main_cmds[] = {
...
   { "fs",         cmd_fs_exec },
...
};
```

2. 在 main.c 文件的 main 函数中注释掉打印“Hello world”的语句，方便控制台输出信息查看。

```
int main(void)
{
    platform_init();
    return 0;
```

```
}
```

4.1.3 操作步骤

littleFS 支持文件夹操作，因此本示例以 littleFS 为例，进行演示 VFS 接口的使用，其操作步骤如下：

1. 在 menuconfig 中打开 filesystem support 选项，并选择 littleFS 及其默认配置
2. 将生成的固件编译烧写进开发板运行
3. 格式化文件系统

```
$fs format d=1
```

4. 挂载文件系统

```
$fs mount d=1
```

5. 在根目录下创建或打开名称为“test.txt”的文件，如下

```
$fs open data/test.txt
```

6. 写入内容 abcd123456789，如下

```
$fs write abcd123456789
```

7. 移动光标到起始位置，如下

```
$fs seek begin 0
```

8. 查看读取文件内容（14 字节），如下

```
$fs read 14
```

9. 移动光标至位置 3 处，如下

```
$fs seek begin 3
```

10. 查看当前光标位置，如下

```
$fs tell
```

11. 写入内容 test，如下

```
$fs write test
```

12. 移动光标到起始位置，如下

```
$fs seek begin 0
```

13. 查看读取文件内容（14 字节），如下

```
$fs read 14
```

14. 查看文件大小，如下

```
$fs size
```

15. 关闭文件，如下

```
$fs close
```

16. 查看路径“/test.txt”的信息，如下

```
$fs stat data/test.txt
```

17. 重命名文件“/test.txt”为“new_test.txt”，如下

```
$fs rename data/test.txt data/new_test.txt
```

18. 在根目录“/”下创建一个文件夹“book”，如下

```
$fs mkdir data/book
```

19. 打开根目录“/”，如下

```
$fs opendir data/
```

20. 逐个读取该文件夹下的文件信息，如下

```
$fs readdir
```

21. 关闭该文件夹，如下

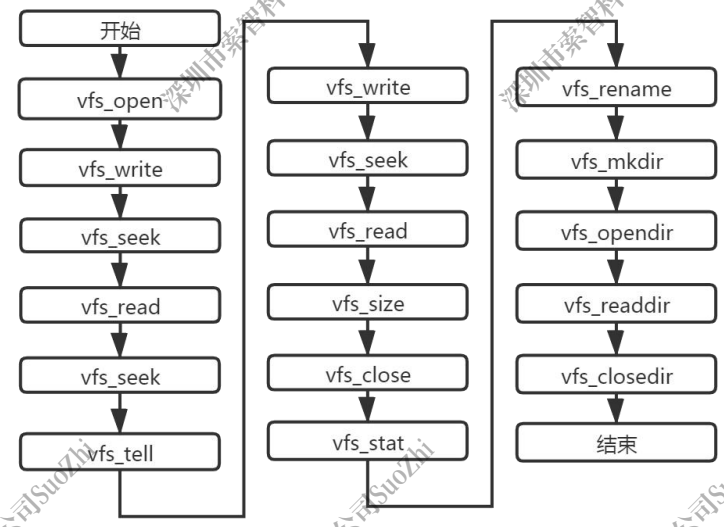
```
$fs closedir
```

4.2 关键实现

上述步骤中，各命令的对应的代码均在 cmd_fs.c 中已实现，用户可进行代码参考。系统上电后，会进行 platform_init()初始化，在初始化工程中会进行 VFS 初始化和挂载文件系统，用户使用时无需自行挂载。

上述步骤中对 VFS 接口的主要调用流程如下：

图 4-1 VFS 接口使用示例流程图



注意

cmd_fs.c 中使用的文件句柄 cmd_file 和文件夹句柄 cmd_dir 为两个不同的变量，cmd_file 句柄只能用于文件相关的操作接口，cmd_dir 句柄只能用于文件夹相关的操作接口，二者不能混用。详细可参考“3.3 接口说明”章节

4.3 效果展示

在评估板中运行各命令后，控制台中打印输如下所示。

```
$ fs format d=1
[cmd] format success
<ACK> 200 OK
$ fs mount d=1
[VFS INF] LittleFS mount success.
[cmd] mount success
<ACK> 200 OK
$ fs open data/test.txt
[cmd] open success
<ACK> 200 OK
$ fs write abcd123456789
<ACK> 200 OK
$ fs seek begin 0
<ACK> 200 OK
$ fs read 14
abcd123456789
<ACK> 200 OK
$ fs seek begin 3
<ACK> 200 OK
```

```
$ fs tell
file position:3
<ACK> 200 OK
$ fs write test
<ACK> 200 OK
$ fs seek begin 0
<ACK> 200 OK
$ fs read 14
abctest456789
<ACK> 200 OK
$ fs size
file size:13
<ACK> 200 OK
$ fs close
<ACK> 200 OK
$ fs stat data/test.txt
data/test.txt, type:file, file size:13
<ACK> 200 OK
$ fs rename data/test.txt data/new_test.txt
file rename success
<ACK> 200 OK
$ fs mkdir data/book
mkdir data/book success
<ACK> 200 OK
$ fs opendir data/
open directory data/ success
<ACK> 200 OK
$ fs readdir
., type:directory
.., type:directory
book, type:directory
new_test.txt, type:file, file size:13
<ACK> 200 OK
$ fs closedir
close directory success
<ACK> 200 OK
```

使用 VFS 接口，创建文件后，写入内容读取符合预期，移动光标后再次写入并读取，内容符合预期，查看文件大小，对文件重命名，创建文件夹，查看文件夹中的文件信息等操作的输出均符合预期。

附录 A： 术语表

表 A-1 术语表

U		
UART	Universal Asynchronous Receiver/Transmitter	通用异步收发传输器
URL	Uniform Resource Locator	统一资源定位系统
V		
VFS	Virtual File System	虚拟文件系统

著作权声明

商标声明

免责声明