# 实验7鸿蒙LiteOS-a内核移植——存储系统移植

1. 实验目的
2. 实验内容

参考资料：vendor\democom\demochip\driver\mtd\spi\_nor\src\common\spinor.c

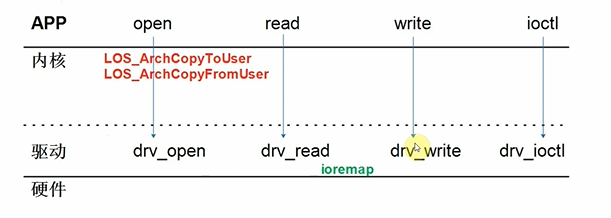
Linux中设备驱动程序分为3类：字符设备、块设备、网络设备。

所谓字符设备就是LED、按键、LCD、触摸屏这些非存储设备，APP可以直接调用驱动函数去操作它们。  
 而块设备就是Flash、磁盘这些存储设备，APP读写普通的文件时，最终会由驱动程序访问硬件。

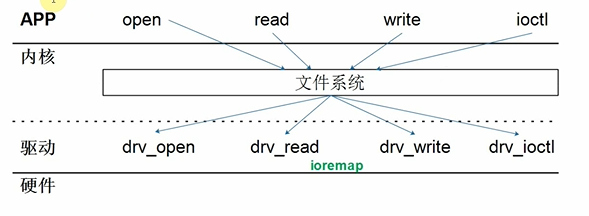
为什么叫**块设备**？以前的磁盘读写时，是以块为单位的：即使只是读写一个字节，也需要读写一个块。 主要差别在于：

* 字符设备驱动程序里，可以读写任意长度的数据
* 块设备驱动程序里，读写数据时以块(扇区)为单位

### 1.1.1 APP与驱动程序的交互



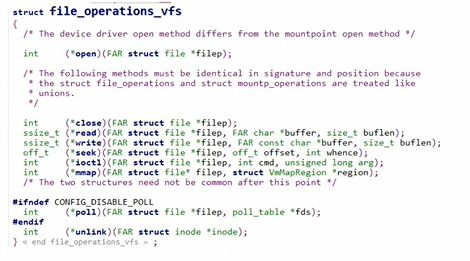
#### *2. 块设备驱动程序教*



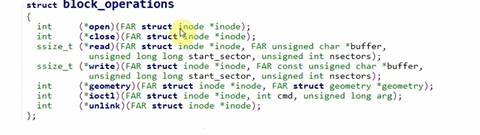
### 1.1.2 驱动程序结构体

从上面的图形可以看到，无论是字符设备还是块设备，都要提供open/read/write/ioctl这些函数。  
 它们的驱动程序核心是类似的：

* 字符设备驱动程序：file\_operations\_vfs



* 块设备驱动程序：block\_operations



### 1.1.3 注册函数

#### *1. 字符设备驱动程序注册函数*

int register\_driver(FAR const char \*path, FAR const struct file\_operations\_vfs \*fops,  
 mode\_t mode, FAR void \*priv);

* 1
* 2

示例：

int ret = register\_driver("/dev/hello", &g\_helloDevOps, 0666, NULL);

* 1

#### *2. 块设备驱动程序注册函数*

int register\_blockdriver(FAR const char \*path,  
 FAR const struct block\_operations \*bops,  
 mode\_t mode, FAR void \*priv);

* 1
* 2
* 3

示例：

int ret = register\_blockdriver("/dev/spinor", &g\_dev\_spinor\_ops, 0755, mtd);

* 1

## 1.2 MTD设备

在各类电子产品中，存储设备类型多种多样，比如Nor Flash、Nand Flash，这些Flash又有不同的接口：比如SPI接口等等。  
 这些不同Flash的访问方法各有不同，但是肯定有这三种操作：

* 读
* 写
* 擦除

那么可以抽象出一个软件层：MTD，含义为\*\*Memory Technology Device\*\*，它封装了不同Flash的操作。主要是抽象出一个结构体：

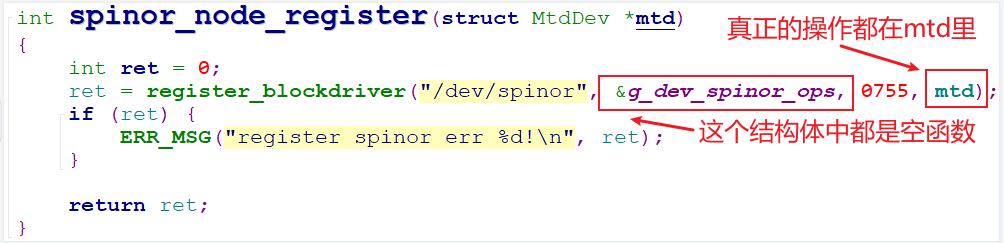
struct MtdDev {  
 VOID \*priv;  
 UINT32 type;  
  
 UINT64 size;  
 UINT32 eraseSize;  
  
 int (\*erase)(struct MtdDev \*mtd, UINT64 start, UINT64 len, UINT64 \*failAddr);  
 int (\*read)(struct MtdDev \*mtd, UINT64 start, UINT64 len, const char \*buf);  
 int (\*write)(struct MtdDev \*mtd, UINT64 start, UINT64 len, const char \*buf);  
};

* 1
* 2
* 3
* 4
* 5
* 6
* 7
* 8
* 9
* 10
* 11

不同的Flash要提供它自己的MtdDev结构体。

## 1.3 块设备驱动程序为MTD开了一个后门

视频里讲解。



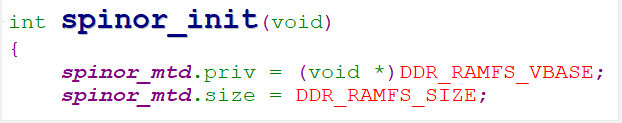
在JFFS2文件系统中，是直接使用MTD的，没有使用block\_operations。  
 比如：third\_party\Linux\_Kernel\fs\jffs2\read.c：



## 1.4 怎么用内存模拟Flash

### 1.4.1 指定要使用的内存地址、大小

源码：vendor\democom\demochip\driver\mtd\spi\_nor\src\common\spinor.c



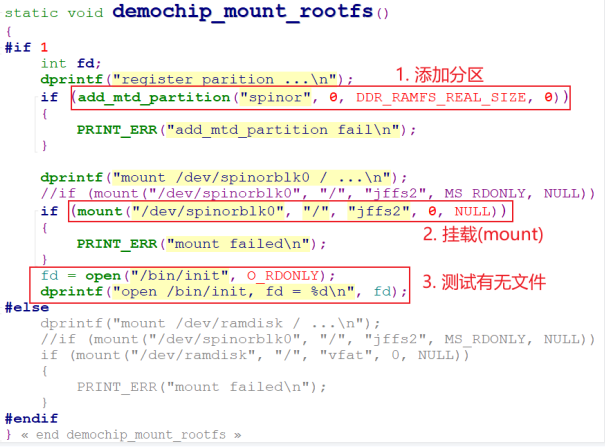
### 1.4.2 实现MtdDev结构体

源码：vendor\democom\demochip\driver\mtd\spi\_nor\src\common\spinor.c



### 1.4.3 怎么使用块设备

* 添加分区
* /dev/spinor表示整个块设备
* /dev/spinorblk0表示里面的第0个分区
* 不添加分区也可以，可以直接挂载/dev/spinor
* mount



# 2. 使用内存模拟Flash

参考资料：vendor\democom\demochip\driver\mtd\spi\_nor\src\common\spinor.c

## 2.1 最终结果

本章节做的修改会制作为补丁文件：

* 04\_openharmony\_ramfs\_imx6ull.patch

假设目录openharmony中是未修改的代码，从没打过补丁；  
 假设补丁文件放在openharmony的同级目录； 打补丁方法如下：

$ cd openharmony  
$ patch -p1 < ../openharmony\_100ask\_v1.2.patch  
$ patch -p1 < ../01\_openharmony\_add\_demo\_board.patch  
$ patch -p1 < ../02\_openharmony\_memmap\_imx6ull.patch   
$ patch -p1 < ../03\_openharmony\_uart\_imx6ull.patch  
$ patch -p1 < ../04\_openharmony\_ramfs\_imx6ull.patch

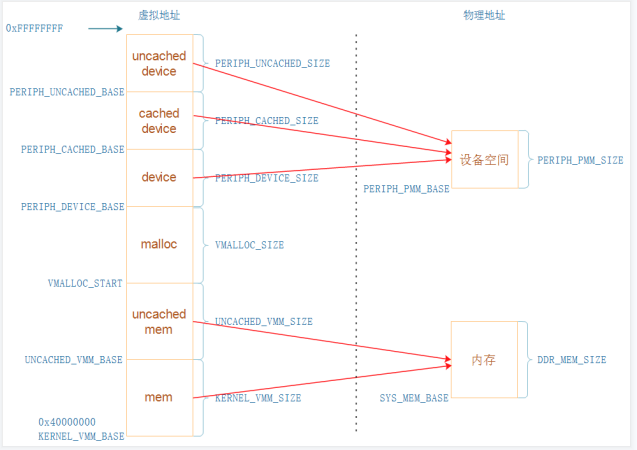
* 1
* 2
* 3
* 4
* 5
* 6

打上补丁后，可以如此编译：

$ cd kernel/liteos\_a  
$ cp tools/build/config/debug/demochip\_clang.config .config  
$ make clean  
$ make

* 1
* 2
* 3
* 4

## 2.2 原来的内存映射



## 2.3 在内存里挖出一块用来模拟Flash

