实验7 鸿蒙 `LiteOS-a` 内核移植——存储系统移植

《实用操作系统》实验报告　22920212204392 黄勖

1　实验环境

Windows10 21H2、Vmware Workstation Pro 16、Ubuntu 18.04

配置了相关的软件。

2　实验目的

了解 LiteOS 中设备驱动程序的分类和特点，学习块设备驱动程序的设计和实现原理

为新增加的单板DemoChip进行存储系统移植，使用内存模拟Flash

3　实验步骤与内容

3.1　存储设备驱动程序分析

Linux中设备驱动程序分为3类：字符设备、块设备、网络设备。

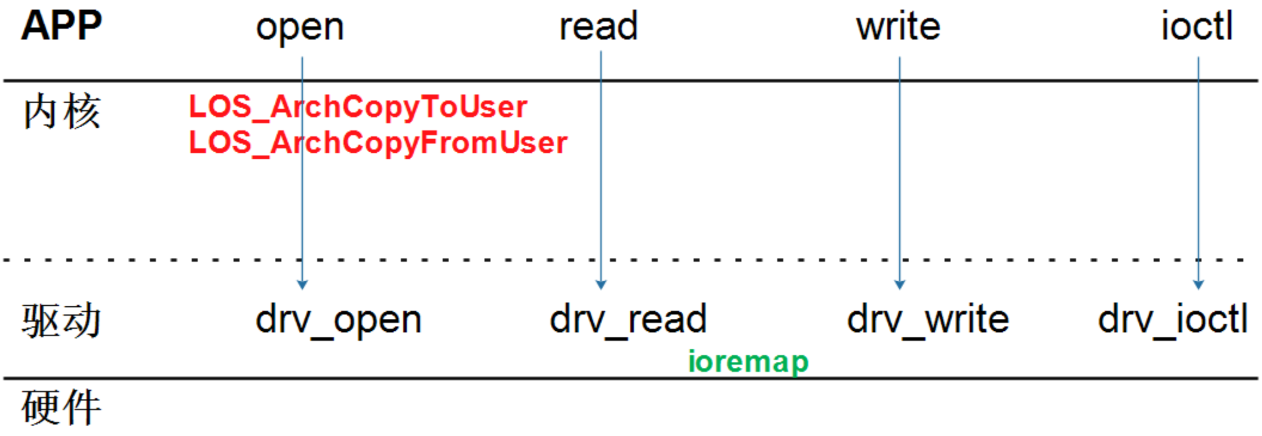
- 字符设备就是LED、按键、LCD、触摸屏这些非存储设备，APP可以直接调用驱动函数去操作它们。
- 块设备就是Flash、磁盘这些存储设备，APP读写普通的文件时，最终会由驱动程序访问硬件。以前的磁盘读写时，是以块为单位的：即使只是读写一个字节，也需要读写一个块。

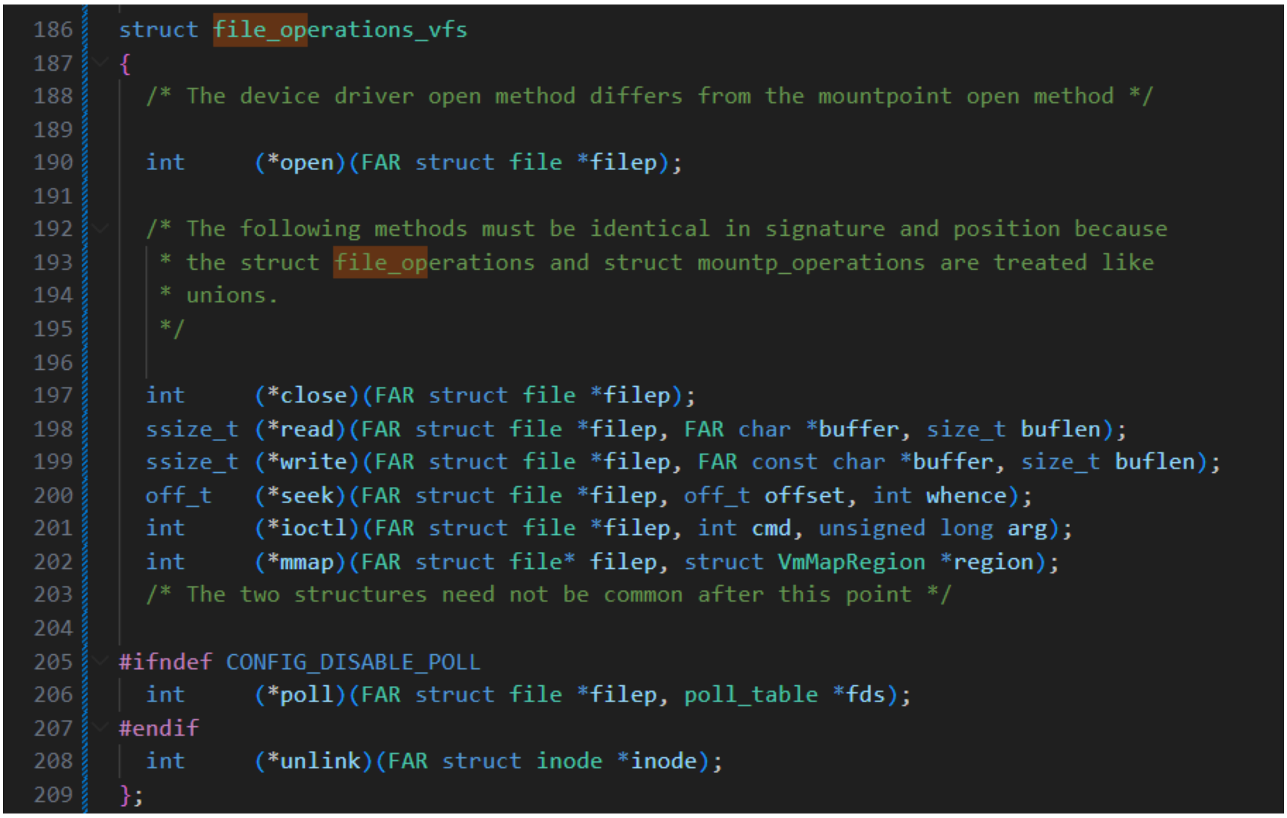
主要差别在于：

- 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

3.1.1　字符设备驱动程序

/kernel/liteos_a/fs/include/fs/fs.h

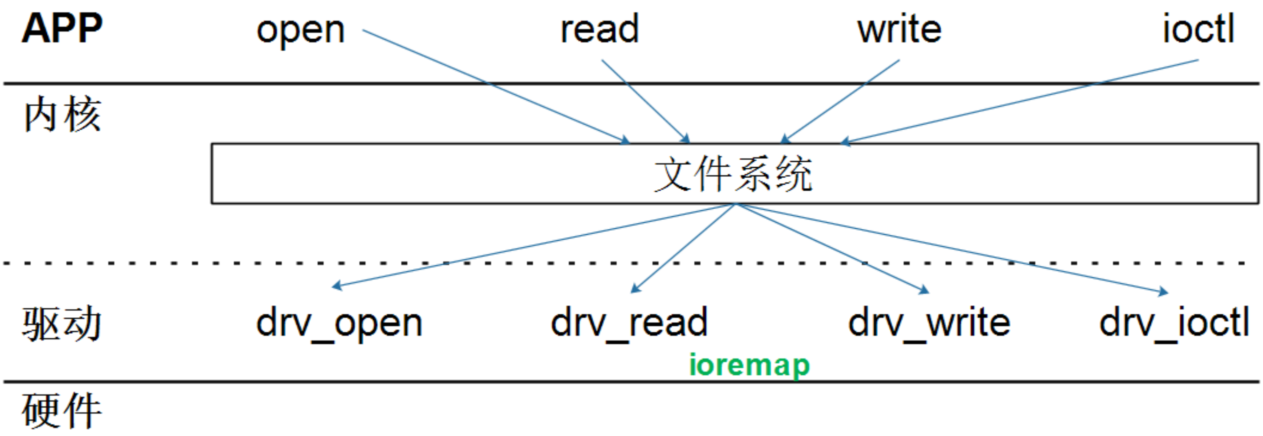


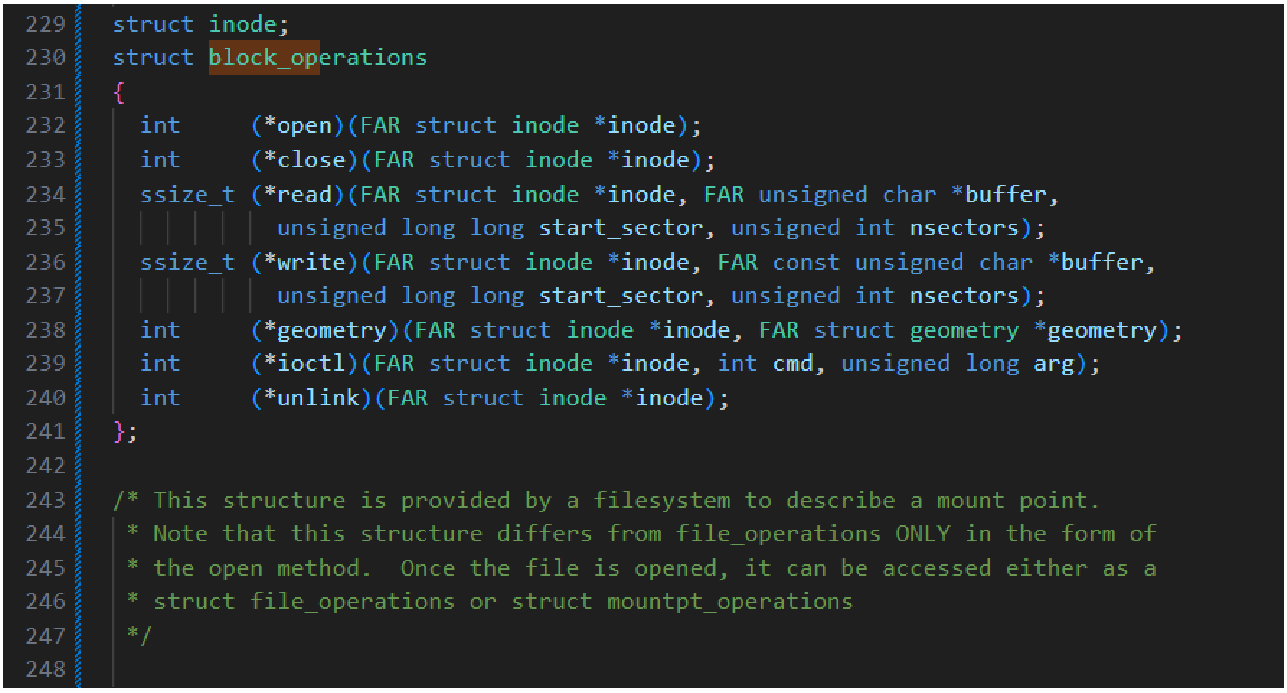


```c
struct file_operations_vfs
{
  /* The device driver open method differs from the mountpoint open method */

  int     (*open)(FAR struct file *filep);

  /* The following methods must be identical in signature and position because
   * the struct file_operations and struct mountp_operations are treated like
   * unions.
   */

  int     (*close)(FAR struct file *filep);
  ssize_t (*read)(FAR struct file *filep, FAR char *buffer, size_t buflen);
  ssize_t (*write)(FAR struct file *filep, FAR const char *buffer, size_t buflen);
  off_t   (*seek)(FAR struct file *filep, off_t offset, int whence);
  int     (*ioctl)(FAR struct file *filep, int cmd, unsigned long arg);
  int     (*mmap)(FAR struct file* filep, struct VmMapRegion *region);
  /* The two structures need not be common after this point */

#ifndef CONFIG_DISABLE_POLL
  int     (*poll)(FAR struct file *filep, poll_table *fds);
#endif
  int     (*unlink)(FAR struct inode *inode);
};
```

3.1.2　块设备驱动程序

/kernel/liteos_a/fs/include/fs/fs.h

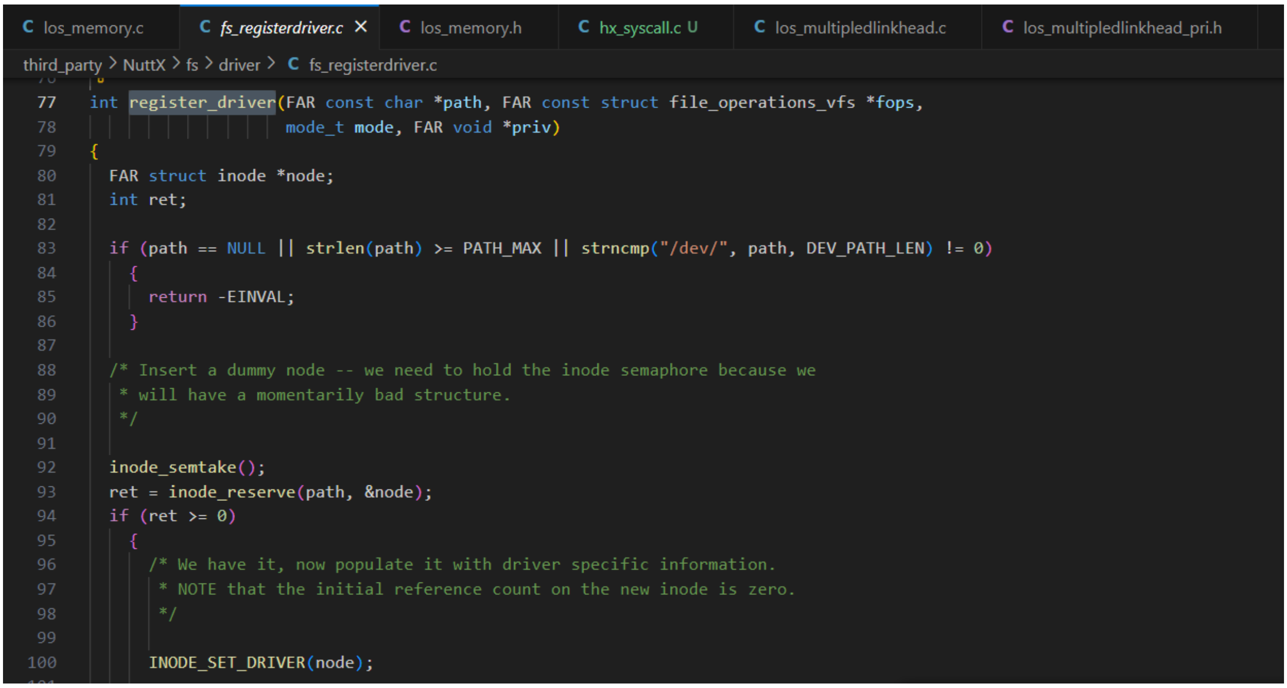




```c
struct inode;
struct block_operations
{
  int     (*open)(FAR struct inode *inode);
  int     (*close)(FAR struct inode *inode);
  ssize_t (*read)(FAR struct inode *inode, FAR unsigned char *buffer,
            unsigned long long start_sector, unsigned int nsectors);
  ssize_t (*write)(FAR struct inode *inode, FAR const unsigned char *buffer,
            unsigned long long start_sector, unsigned int nsectors);
  int     (*geometry)(FAR struct inode *inode, FAR struct geometry *geometry);
  int     (*ioctl)(FAR struct inode *inode, int cmd, unsigned long arg);
  int     (*unlink)(FAR struct inode *inode);
};

/* This structure is provided by a filesystem to describe a mount point.
 * Note that this structure differs from file_operations ONLY in the form of
 * the open method.  Once the file is opened, it can be accessed either as a
 * struct file_operations or struct mountpt_operations
 */

```

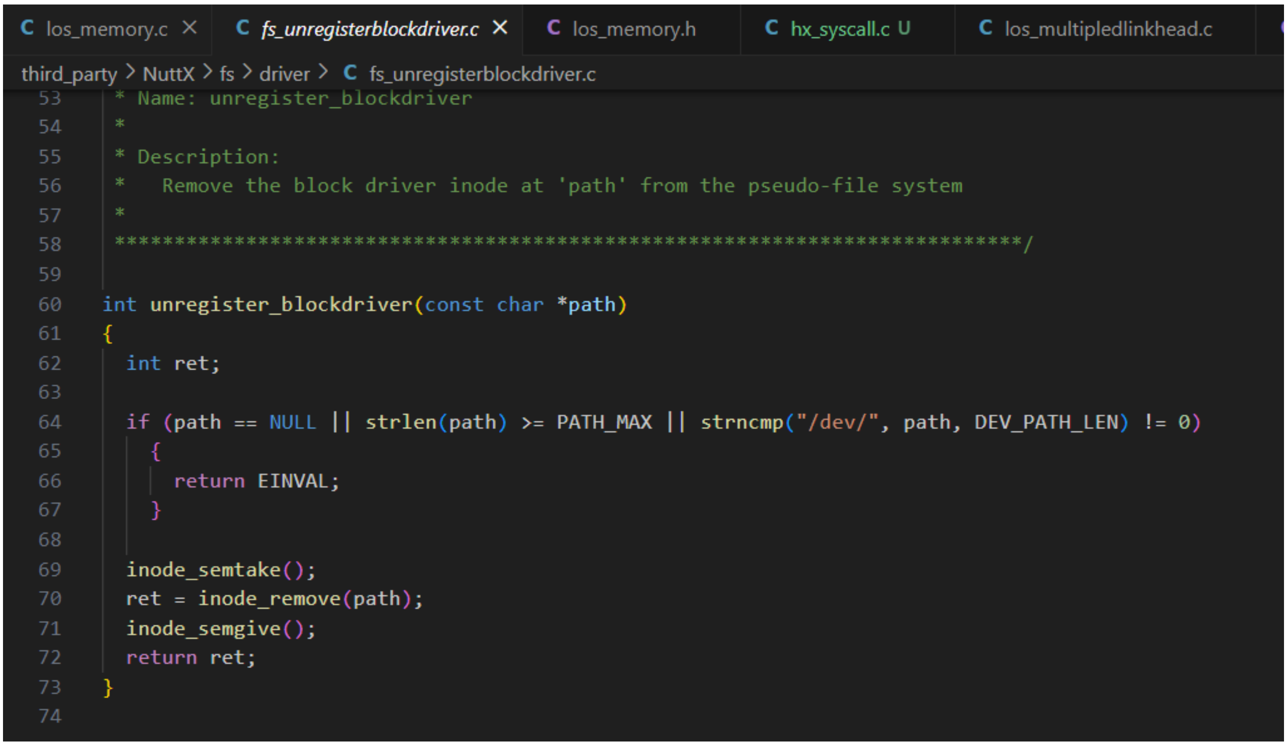
3.1.3　注册函数

字符设备驱动程序注册函数：openharmony\third_party\NuttX\fs\driver\fs_registerdriver.c


C los_memory.c
C fs_registerdriver.c ×
C los_memory.h
C hx_syscall.c U
C los_multipledlinkhead.c
C los_multipledlinkhead_pri.h
third_party > NuttX > fs > driver > C fs_registerdriver.c

```c
int register_driver(FAR const char *path, FAR const struct file_operations_vfs *fops,
                    mode_t mode, FAR void *priv)
{
  FAR struct inode *node;
  int ret;

  if (path == NULL || strlen(path) >= PATH_MAX || strncmp("/dev/", path, DEV_PATH_LEN) != 0)
    {
      return -EINVAL;
    }

  /* Insert a dummy node -- we need to hold the inode semaphore because we
   * will have a momentarily bad structure.
   */

  inode_semtake();
  ret = inode_reserve(path, &node);
  if (ret >= 0)
    {
      /* We have it, now populate it with driver specific information.
       * NOTE that the initial reference count on the new inode is zero.
       */

      INODE_SET_DRIVER(node);
```

块设备驱动程序注册函数：openharmony\third_party\NuttX\fs\driver\fs_registerblockdriver.c


los_memory.c
fs_unregisterblockdriver.c
los_memory.h
hx_syscall.c U
los_multipledlinkhead.c
third_party > NuttX > fs > driver > fs_unregisterblockdriver.c

```c
 * Name: unregister_blockdriver
 *
 * Description:
 *   Remove the block driver inode at 'path' from the pseudo-file system
 *
 ****************************************************************************/

int unregister_blockdriver(const char *path)
{
  int ret;

  if (path == NULL || strlen(path) >= PATH_MAX || strncmp("/dev/", path, DEV_PATH_LEN) != 0)
    {
      return EINVAL;
    }

  inode_semtake();
  ret = inode_remove(path);
  inode_semgive();
  return ret;
}
```

3.1.4 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：

/kernel/liteos_a/fs/vfs/include/driver/mtd_dev.h

```c
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);
};
```

不同的Flash要提供它自己的MtdDev结构体。

3.1.5 块驱动设备与MTD

查看块设备的注册函数，阅读代码发现 `g_dev_spinor_ops` 里面的函数都是空的，而真正的操作定义在 `mtd` 中

/openharmony/vendor/democom/demochip/driver/mtd/spi_nor/src/common/spinor.c

```
int spinor_node_register(struct MtdDev *mtd)
{
    int ret = 0;
    ret = register_blockdriver("/dev/spinor", &g_dev_spinor_ops,
0755, mtd);
    if (ret)
    {
       ERR_MSG("register spinor err %d!\n", ret);
    }
    return ret;
}
```

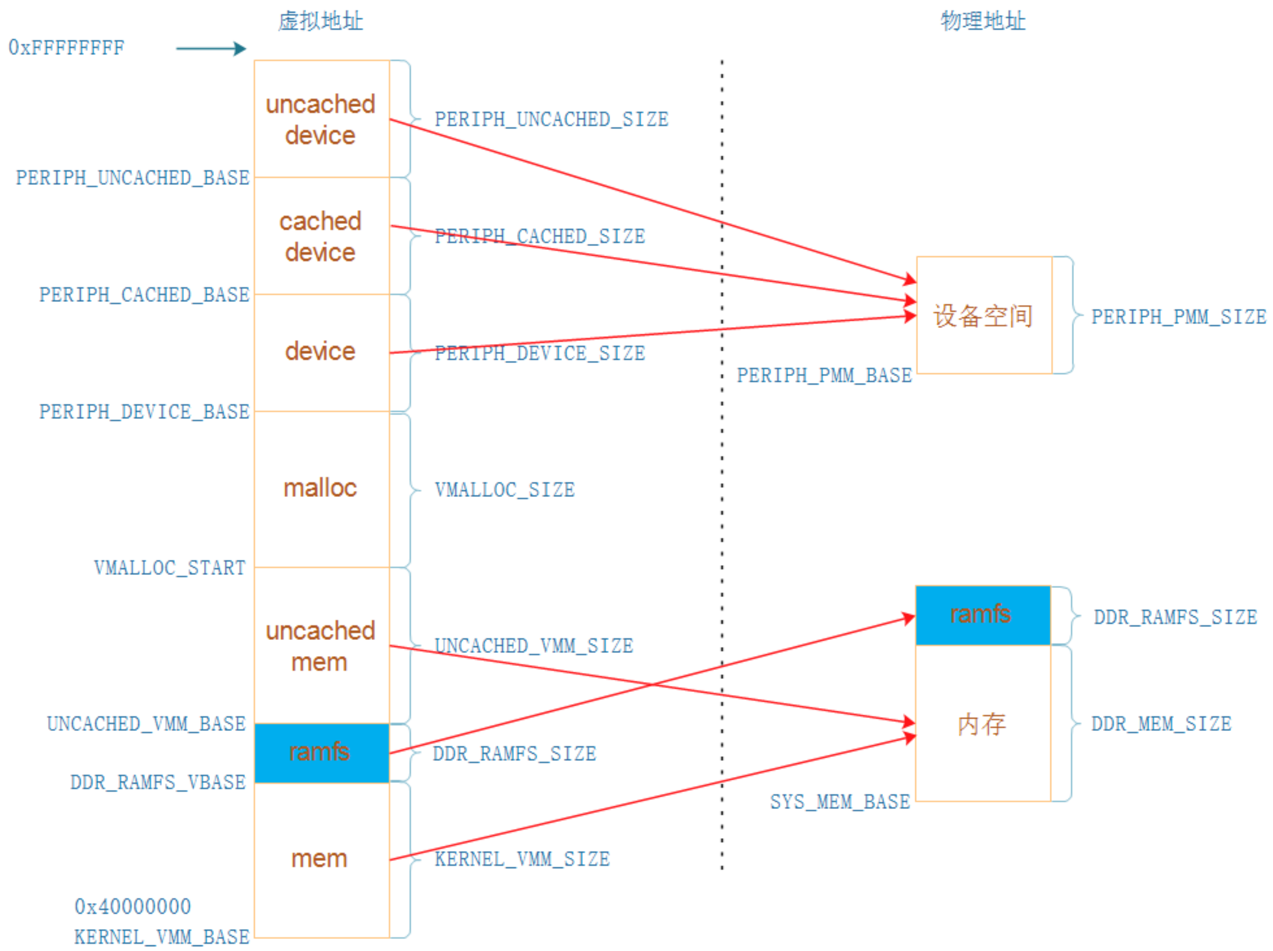
比如在JFFS2文件系统中，就直接使用了MTD，没有使用block_operations

```
int jffs2_flash_direct_read(struct jffs2_sb_info *c, loff_t ofs,
size_t len, size_t *retlen, const char *buf)
{
    int ret;
    ret = c->mtd->read(c->mtd, ofs, len, (char *)buf);
    if (ret >= 0) {
        *retlen = ret;
        return 0;
    }
    *retlen = 0;
    return ret;
}
```

3.2 使用内存模拟Flash

如下图所示，使用内存模拟Flash的思路大概为：

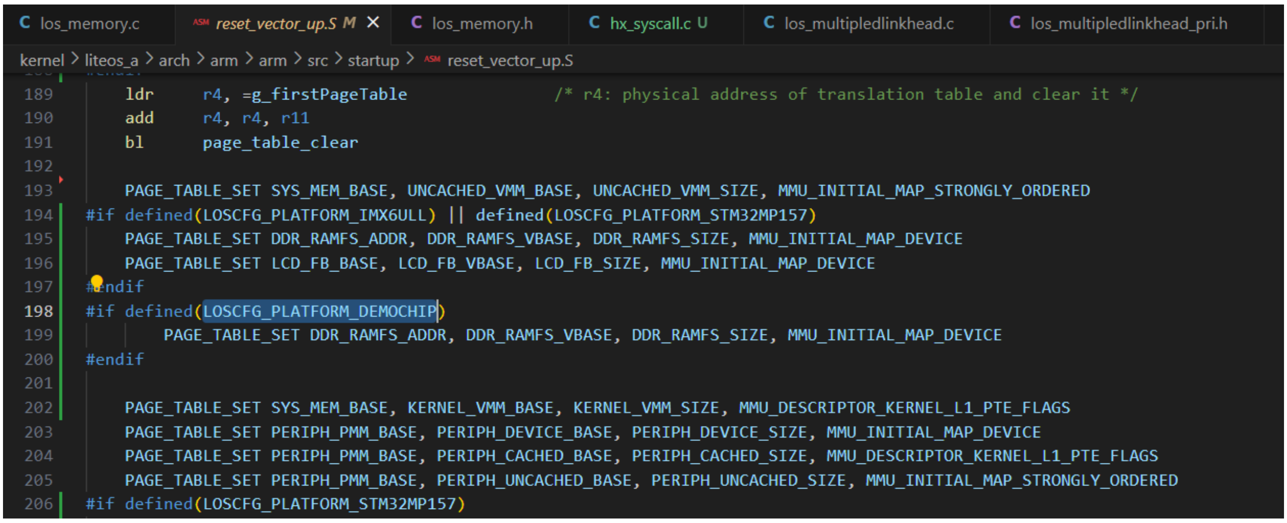
1. 指定要使用的内存地址、大小，减小 DDR_MEM_SIZE ，使系统内核管理的内存空间减少，腾出来的内存空间用于模拟Flash
2. 将虚拟地址的Ramfs部分地址映射到模拟Flash的内存空间，当程序通过虚拟地址访问Flash的时候，就会访问到对应的模拟Flash



3.2.1 添加宏开关，固件初始化

kernel/liteos_a/arch/arm/arm/src/startup/reset_vector_up.S

```
PAGE_TABLE_SET LCD_FB_BASE, LCD_FB_VBASE, LCD_FB_SIZE, MMU_INITIAL_MAP_DEVICE
#endif
+#if defined(LOSCFG_PLATFORM_DEMOCHIP)
+        PAGE_TABLE_SET DDR_RAMFS_ADDR, DDR_RAMFS_VBASE, DDR_RAMFS_SIZE,
MMU_INITIAL_MAP_DEVICE
+#endif
+
PAGE_TABLE_SET SYS_MEM_BASE, KERNEL_VMM_BASE, KERNEL_VMM_SIZE,
MMU_DESCRIPTOR_KERNEL_L1_PTE_FLAGS
```



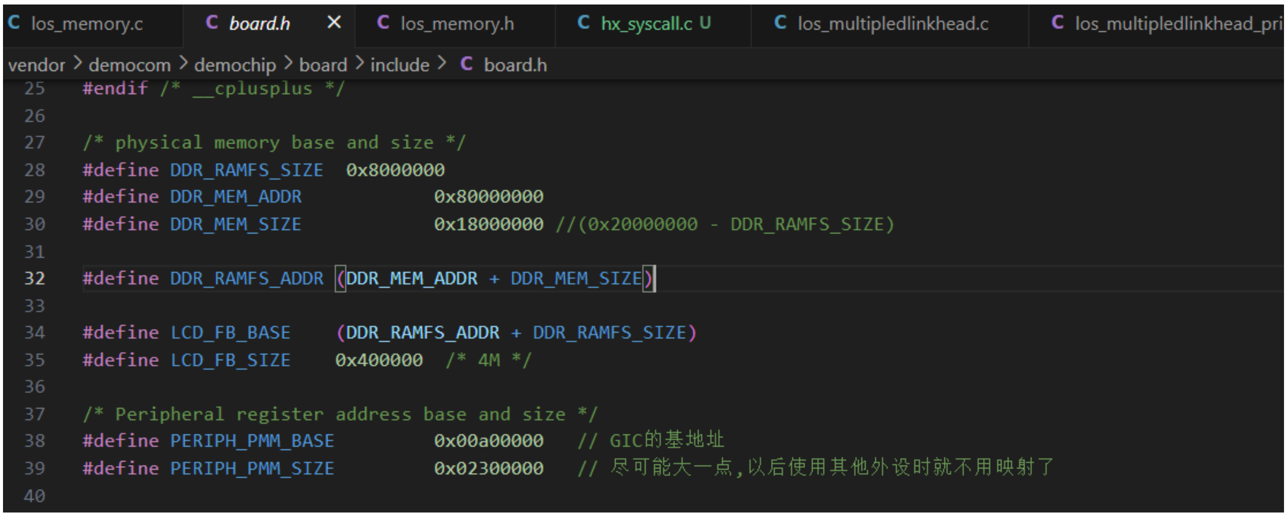
los_memory.c
reset_vector_up.S M
los_memory.h
hx_syscall.c U
los_multipledlinkhead.c
los_multipledlinkhead_pri.h
kernel > liteos_a > arch > arm > arm > src > startup > reset_vector_up.S

```
    ldr     r4, =g_firstPageTable               /* r4: physical address of translation table and clear it */
    add     r4, r4, r11
    bl      page_table_clear

    PAGE_TABLE_SET SYS_MEM_BASE, UNCACHED_VMM_BASE, UNCACHED_VMM_SIZE, MMU_INITIAL_MAP_STRONGLY_ORDERED
#if defined(LOSCFG_PLATFORM_IMX6ULL) || defined(LOSCFG_PLATFORM_STM32MP157)
    PAGE_TABLE_SET DDR_RAMFS_ADDR, DDR_RAMFS_VBASE, DDR_RAMFS_SIZE, MMU_INITIAL_MAP_DEVICE
    PAGE_TABLE_SET LCD_FB_BASE, LCD_FB_VBASE, LCD_FB_SIZE, MMU_INITIAL_MAP_DEVICE
#endif
#if defined(LOSCFG_PLATFORM_DEMOCHIP)
        PAGE_TABLE_SET DDR_RAMFS_ADDR, DDR_RAMFS_VBASE, DDR_RAMFS_SIZE, MMU_INITIAL_MAP_DEVICE
#endif

    PAGE_TABLE_SET SYS_MEM_BASE, KERNEL_VMM_BASE, KERNEL_VMM_SIZE, MMU_DESCRIPTOR_KERNEL_L1_PTE_FLAGS
    PAGE_TABLE_SET PERIPH_PMM_BASE, PERIPH_DEVICE_BASE, PERIPH_DEVICE_SIZE, MMU_INITIAL_MAP_DEVICE
    PAGE_TABLE_SET PERIPH_PMM_BASE, PERIPH_CACHED_BASE, PERIPH_CACHED_SIZE, MMU_DESCRIPTOR_KERNEL_L1_PTE_FLAGS
    PAGE_TABLE_SET PERIPH_PMM_BASE, PERIPH_UNCACHED_BASE, PERIPH_UNCACHED_SIZE, MMU_INITIAL_MAP_STRONGLY_ORDERED
#if defined(LOSCFG_PLATFORM_STM32MP157)
```

3.2.2 修改宏定义，确定由内核管理的内存的基地址、大小等

vendor/democom/demochip/board/include/board.h

```
 #endif /* __cplusplus */

 /* physical memory base and size */
+#define DDR_RAMFS_SIZE  0x8000000
 #define DDR_MEM_ADDR            0x80000000
-#define DDR_MEM_SIZE            0x20000000
+#define DDR_MEM_SIZE            0x18000000 //(0x20000000 - DDR_RAMFS_SIZE)

 #define DDR_RAMFS_ADDR (DDR_MEM_ADDR + DDR_MEM_SIZE)
-#define DDR_RAMFS_SIZE 0x4000000  /* 60M for ramfs, 4M for lcd */

 #define LCD_FB_BASE    (DDR_RAMFS_ADDR + DDR_RAMFS_SIZE)
 #define LCD_FB_SIZE    0x400000  /* 4M */
```



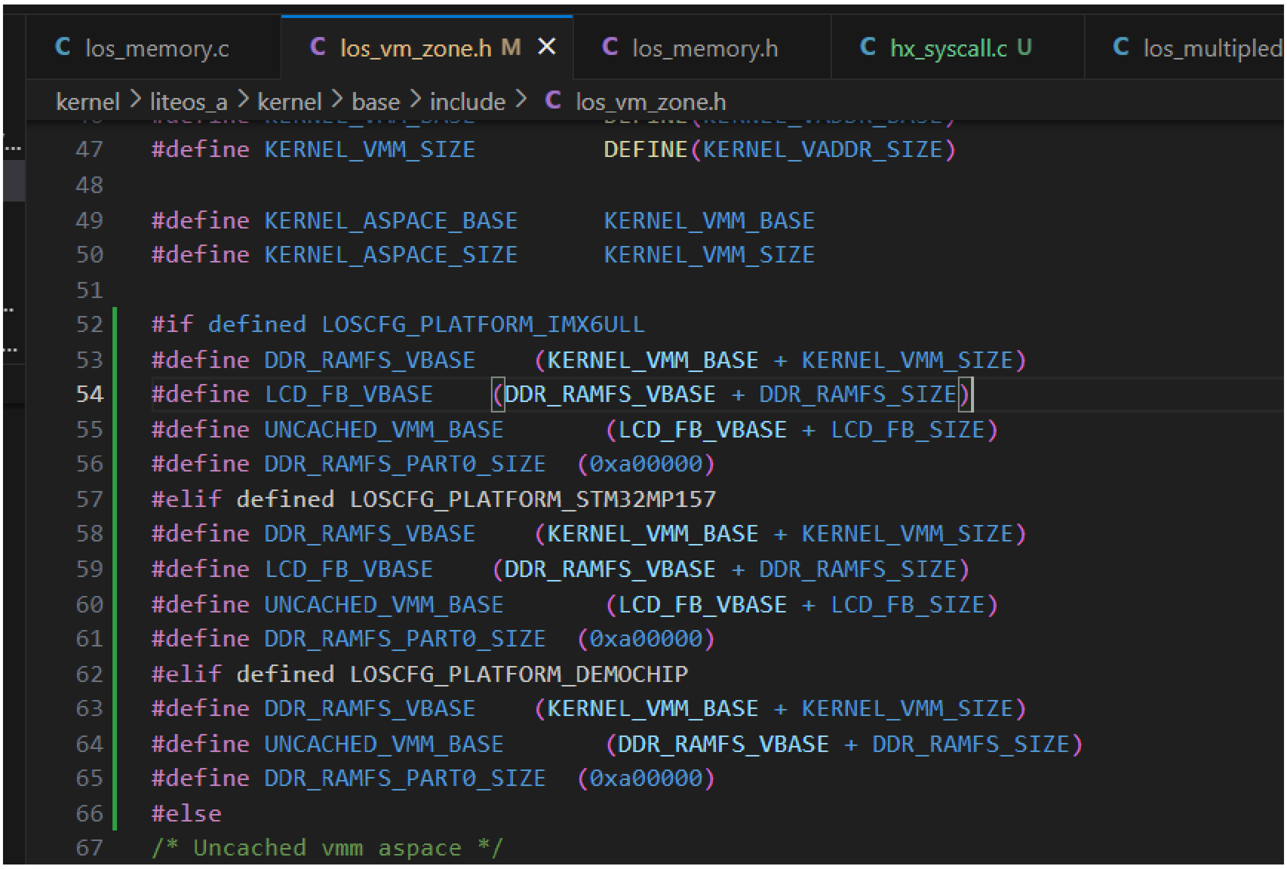
los_memory.c
board.h
los_memory.h
hx_syscall.c U
los_multipledlinkhead.c
los_multipledlinkhead_pri
vendor > democom > demochip > board > include > board.h

```
#endif /* __cplusplus */

/* physical memory base and size */
#define DDR_RAMFS_SIZE  0x8000000
#define DDR_MEM_ADDR            0x80000000
#define DDR_MEM_SIZE            0x18000000 //(0x20000000 - DDR_RAMFS_SIZE)

#define DDR_RAMFS_ADDR (DDR_MEM_ADDR + DDR_MEM_SIZE)

#define LCD_FB_BASE    (DDR_RAMFS_ADDR + DDR_RAMFS_SIZE)
#define LCD_FB_SIZE    0x400000  /* 4M */

/* Peripheral register address base and size */
#define PERIPH_PMM_BASE         0x00a00000   // GIC的基地址
#define PERIPH_PMM_SIZE         0x02300000   // 尽可能大一点,以后使用其他外设时就不用映射了

```

3.2.3　为自己的板子添加宏定义，通过宏的计算来确定模拟 Flash 的地址信息

kernel/liteos_a/kernel/base/include/los_vm_zone.h

```
#define DDR_RAMFS_VBASE     (KERNEL_VMM_BASE + KERNEL_VMM_SIZE)
#define LCD_FB_VBASE     (DDR_RAMFS_VBASE + DDR_RAMFS_SIZE)
#define UNCACHED_VMM_BASE        (LCD_FB_VBASE + LCD_FB_SIZE)
-#define DDR_RAMFS_REAL_SIZE  (0xa00000)
+#define DDR_RAMFS_PART0_SIZE  (0xa00000)
#elif defined LOSCFG_PLATFORM_STM32MP157
#define DDR_RAMFS_VBASE     (KERNEL_VMM_BASE + KERNEL_VMM_SIZE)
#define LCD_FB_VBASE     (DDR_RAMFS_VBASE + DDR_RAMFS_SIZE)
#define UNCACHED_VMM_BASE        (LCD_FB_VBASE + LCD_FB_SIZE)
-#define DDR_RAMFS_REAL_SIZE  (0xa00000)
+#define DDR_RAMFS_PART0_SIZE  (0xa00000)
+#elif defined LOSCFG_PLATFORM_DEMOCHIP
+#define DDR_RAMFS_VBASE     (KERNEL_VMM_BASE + KERNEL_VMM_SIZE)
+#define UNCACHED_VMM_BASE        (DDR_RAMFS_VBASE + DDR_RAMFS_SIZE)
+#define DDR_RAMFS_PART0_SIZE  (0xa00000)
#else
```



```
#define KERNEL_VMM_SIZE          DEFINE(KERNEL_VADDR_SIZE)

#define KERNEL_ASPACE_BASE       KERNEL_VMM_BASE
#define KERNEL_ASPACE_SIZE       KERNEL_VMM_SIZE

#if defined LOSCFG_PLATFORM_IMX6ULL
#define DDR_RAMFS_VBASE     (KERNEL_VMM_BASE + KERNEL_VMM_SIZE)
#define LCD_FB_VBASE     (DDR_RAMFS_VBASE + DDR_RAMFS_SIZE)
#define UNCACHED_VMM_BASE        (LCD_FB_VBASE + LCD_FB_SIZE)
#define DDR_RAMFS_PART0_SIZE  (0xa00000)
#elif defined LOSCFG_PLATFORM_STM32MP157
#define DDR_RAMFS_VBASE     (KERNEL_VMM_BASE + KERNEL_VMM_SIZE)
#define LCD_FB_VBASE     (DDR_RAMFS_VBASE + DDR_RAMFS_SIZE)
#define UNCACHED_VMM_BASE        (LCD_FB_VBASE + LCD_FB_SIZE)
#define DDR_RAMFS_PART0_SIZE  (0xa00000)
#elif defined LOSCFG_PLATFORM_DEMOCHIP
#define DDR_RAMFS_VBASE     (KERNEL_VMM_BASE + KERNEL_VMM_SIZE)
#define UNCACHED_VMM_BASE        (DDR_RAMFS_VBASE + DDR_RAMFS_SIZE)
#define DDR_RAMFS_PART0_SIZE  (0xa00000)
#else
/* Uncached vmm aspace */
```

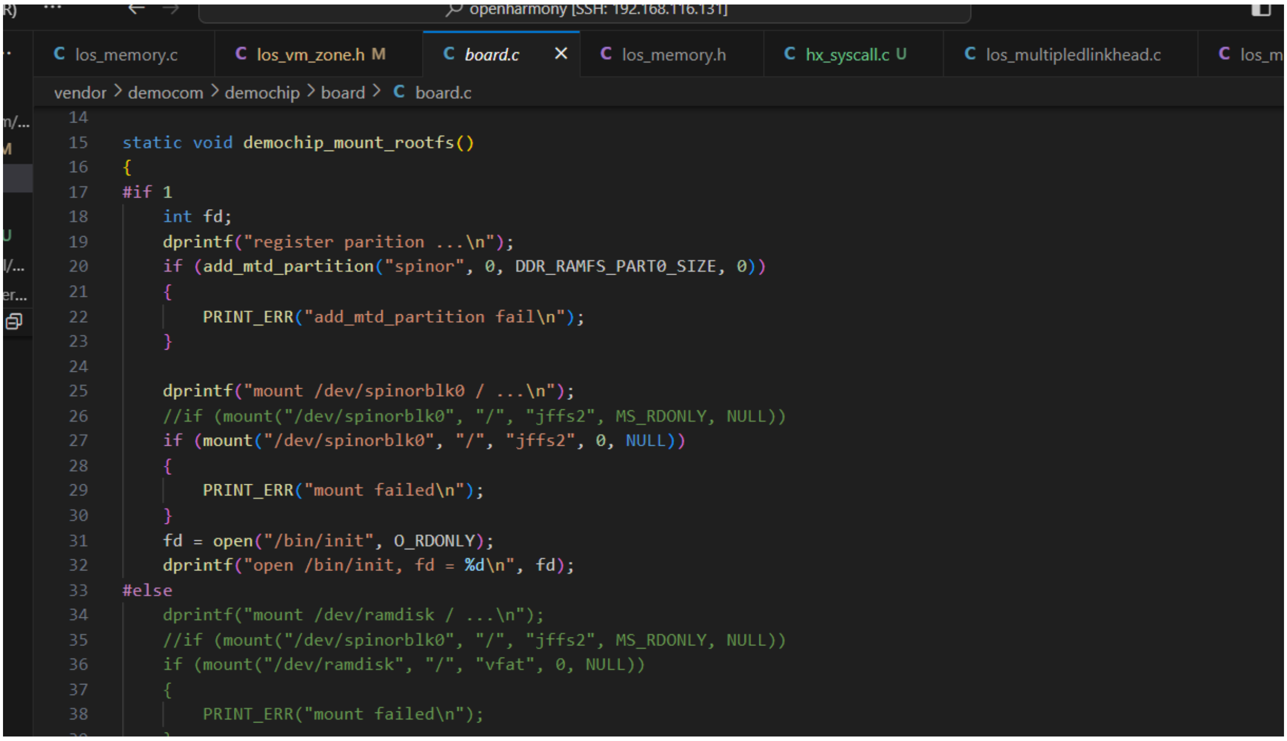
3.2.4　修改驱动初始化函数以及挂载函数的名称

在挂载跟文件系统的函数中，修改添加分区的宏定义。

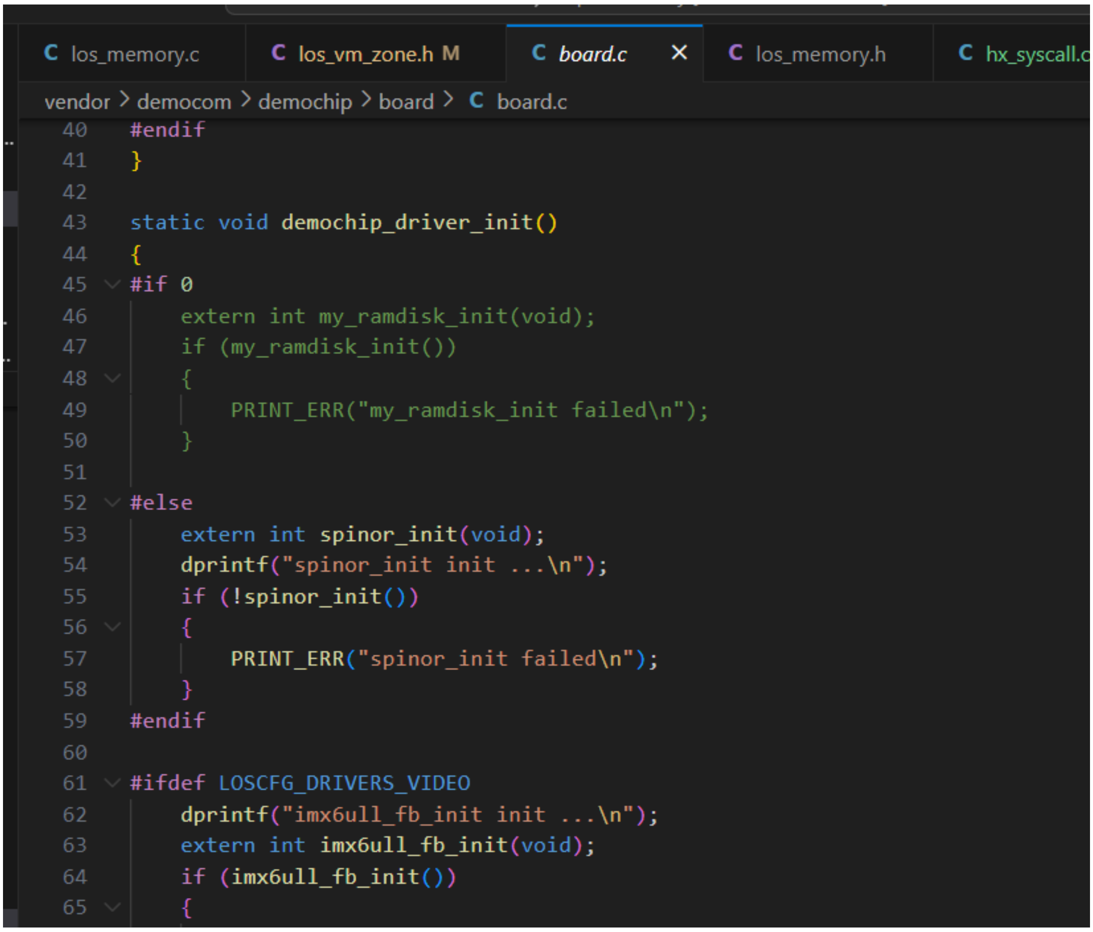
vendor/democom/demcohip/board/board.c

```
    return 0;
```

```
 }

-static void imx6ull_mount_rootfs()
+static void demochip_mount_rootfs()
 {
-#if 0
+#if 1
    int fd;
     dprintf("register parition ...\n");
-    if (add_mtd_partition("spinor", 0, DDR_RAMFS_REAL_SIZE, 0))
+    if (add_mtd_partition("spinor", 0, DDR_RAMFS_PART0_SIZE, 0))
     {
         PRINT_ERR("add_mtd_partition fail\n");
     }
@@ -30,7 +30,7 @@
     }
    fd = open("/bin/init", O_RDONLY);
     dprintf("open /bin/init, fd = %d\n", fd);
-//#else
+#else
     dprintf("mount /dev/ramdisk / ...\n");
     //if (mount("/dev/spinorblk0", "/", "jffs2", MS_RDONLY, NULL))
     if (mount("/dev/ramdisk", "/", "vfat", 0, NULL))
@@ -40,7 +40,7 @@
 #endif
 }

-static void imx6ull_driver_init()
+static void demochip_driver_init()
 {
 #if 0
    extern int my_ramdisk_init(void);
@@ -83,8 +83,8 @@
     extern int mem_dev_register(void);
     mem_dev_register();
 #endif
-    imx6ull_driver_init();
-    imx6ull_mount_rootfs();
+    demochip_driver_init();
+    demochip_mount_rootfs();

 #ifdef LOSCFG_DRIVERS_HDF
```



```
static void demochip_mount_rootfs()
{
#if 1
    int fd;
    dprintf("register parition ...\n");
    if (add_mtd_partition("spinor", 0, DDR_RAMFS_PART0_SIZE, 0))
    {
        PRINT_ERR("add_mtd_partition fail\n");
    }

    dprintf("mount /dev/spinorblk0 / ...\n");
    //if (mount("/dev/spinorblk0", "/", "jffs2", MS_RDONLY, NULL))
    if (mount("/dev/spinorblk0", "/", "jffs2", 0, NULL))
    {
        PRINT_ERR("mount failed\n");
    }
    fd = open("/bin/init", O_RDONLY);
    dprintf("open /bin/init, fd = %d\n", fd);
#else
    dprintf("mount /dev/ramdisk / ...\n");
    //if (mount("/dev/spinorblk0", "/", "jffs2", MS_RDONLY, NULL))
    if (mount("/dev/ramdisk", "/", "vfat", 0, NULL))
    {
        PRINT_ERR("mount failed\n");
```



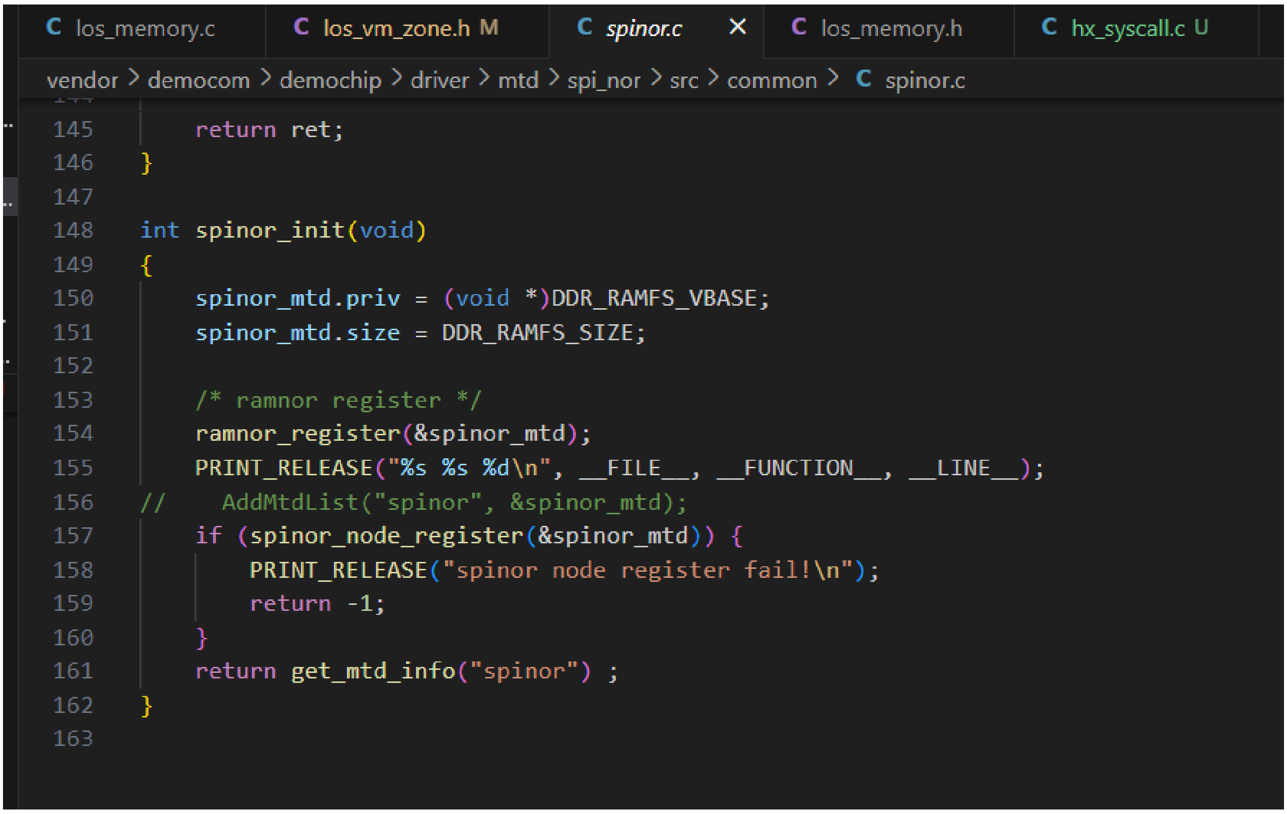
```
#endif
}

static void demochip_driver_init()
{
#if 0
    extern int my_ramdisk_init(void);
    if (my_ramdisk_init())
    {
        PRINT_ERR("my_ramdisk_init failed\n");
    }

#else
    extern int spinor_init(void);
    dprintf("spinor_init init ...\n");
    if (!spinor_init())
    {
        PRINT_ERR("spinor_init failed\n");
    }
#endif

#ifdef LOSCFG_DRIVERS_VIDEO
    dprintf("imx6ull_fb_init init ...\n");
    extern int imx6ull_fb_init(void);
    if (imx6ull_fb_init())
    {
```

3.2.5 初始化驱动

vendor/democom/demochip/driver/mtd/spi_nor/src/common/spinor.c

赋给 spinor_mtd 基地址和大小

```
 int spinor_init(void)
 {
-    spinor_mtd.priv = (void *)0 ; //100ask err, DDR_RAMFS_VBASE;
-   spinor_mtd.size = 0; //100ask err, DDR_RAMFS_SIZE;
+    spinor_mtd.priv = (void *)DDR_RAMFS_VBASE;
+   spinor_mtd.size = DDR_RAMFS_SIZE;

     /* ramnor register */
     ramnor_register(&spinor_mtd);
```

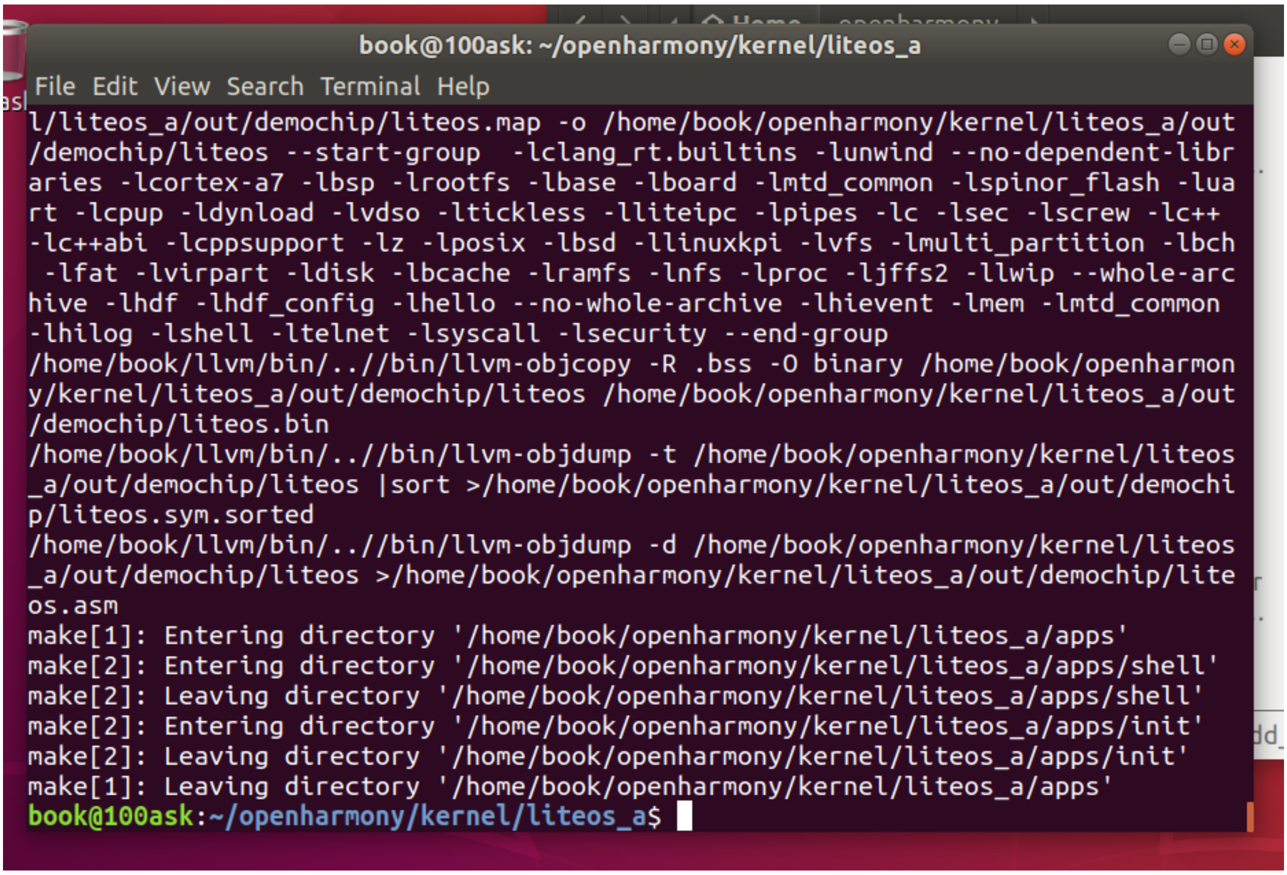

C los_memory.c
C los_vm_zone.h M
C spinor.c ×
C los_memory.h
C hx_syscall.c U
vendor > democom > demochip > driver > mtd > spi_nor > src > common > C spinor.c

```
    return ret;
}

int spinor_init(void)
{
    spinor_mtd.priv = (void *)DDR_RAMFS_VBASE;
    spinor_mtd.size = DDR_RAMFS_SIZE;

    /* ramnor register */
    ramnor_register(&spinor_mtd);
    PRINT_RELEASE("%s %s %d\n", __FILE__, __FUNCTION__, __LINE__);
//    AddMtdList("spinor", &spinor_mtd);
    if (spinor_node_register(&spinor_mtd)) {
        PRINT_RELEASE("spinor node register fail!\n");
        return -1;
    }
    return get_mtd_info("spinor") ;
}
```

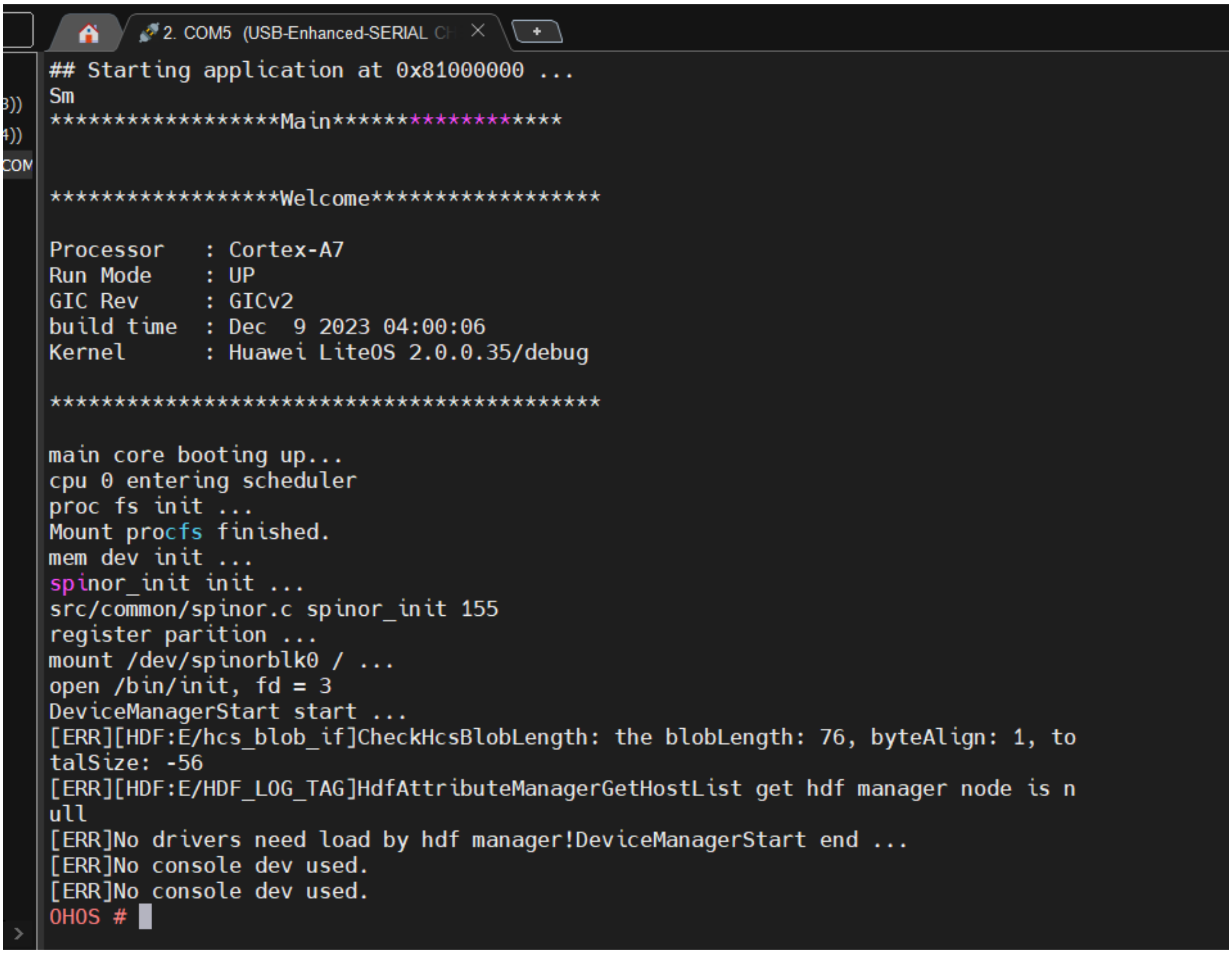
3.2.6 编译烧录

成功进入鸿蒙系统，说明实现了存储系统的移植

```
$ cd kernel/liteos_a
$ cp tools/build/config/debug/demochip_clang.config .config
$ make clean
$ make
```


book@100ask: ~/openharmony/kernel/liteos_a
File Edit View Search Terminal Help

```
l/liteos_a/out/demochip/liteos.map -o /home/book/openharmony/kernel/liteos_a/out
/demochip/liteos --start-group  -lclang_rt.builtins -lunwind --no-dependent-libr
aries -lcortex-a7 -lbsp -lrootfs -lbase -lboard -lmtd_common -lspinor_flash -lua
rt -lcpup -ldynload -lvdso -ltickless -lliteipc -lpipes -lc -lsec -lscrew -lc++
-lc++abi -lcppsupport -lz -lposix -lbsd -llinuxkpi -lvfs -lmulti_partition -lbch
 -lfat -lvirpart -ldisk -lbcache -lramfs -lnfs -lproc -ljffs2 -llwip --whole-arc
hive -lhdf -lhdf_config -lhello --no-whole-archive -lhievent -lmem -lmtd_common
-lhilog -lshell -ltelnet -lsyscall -lsecurity --end-group
/home/book/llvm/bin/..//bin/llvm-objcopy -R .bss -O binary /home/book/openharmon
y/kernel/liteos_a/out/demochip/liteos /home/book/openharmony/kernel/liteos_a/out
/demochip/liteos.bin
/home/book/llvm/bin/..//bin/llvm-objdump -t /home/book/openharmony/kernel/liteos
_a/out/demochip/liteos |sort >/home/book/openharmony/kernel/liteos_a/out/demochi
p/liteos.sym.sorted
/home/book/llvm/bin/..//bin/llvm-objdump -d /home/book/openharmony/kernel/liteos
_a/out/demochip/liteos >/home/book/openharmony/kernel/liteos_a/out/demochip/lite
os.asm
make[1]: Entering directory '/home/book/openharmony/kernel/liteos_a/apps'
make[2]: Entering directory '/home/book/openharmony/kernel/liteos_a/apps/shell'
make[2]: Leaving directory '/home/book/openharmony/kernel/liteos_a/apps/shell'
make[2]: Entering directory '/home/book/openharmony/kernel/liteos_a/apps/init'
make[2]: Leaving directory '/home/book/openharmony/kernel/liteos_a/apps/init'
make[1]: Leaving directory '/home/book/openharmony/kernel/liteos_a/apps'
book@100ask:~/openharmony/kernel/liteos_a$ 
```



```
## Starting application at 0x81000000 ...
Sm
*****************Main*****************


*****************Welcome*****************

Processor   : Cortex-A7
Run Mode    : UP
GIC Rev     : GICv2
build time  : Dec  9 2023 04:00:06
Kernel      : Huawei LiteOS 2.0.0.35/debug

*******************************************

main core booting up...
cpu 0 entering scheduler
proc fs init ...
Mount procfs finished.
mem dev init ...
spinor_init init ...
src/common/spinor.c spinor_init 155
register parition ...
mount /dev/spinorblk0 / ...
open /bin/init, fd = 3
DeviceManagerStart start ...
[ERR][HDF:E/hcs_blob_if]CheckHcsBlobLength: the blobLength: 76, byteAlign: 1, to
talSize: -56
[ERR][HDF:E/HDF_LOG_TAG]HdfAttributeManagerGetHostList get hdf manager node is n
ull
[ERR]No drivers need load by hdf manager!DeviceManagerStart end ...
[ERR]No console dev used.
[ERR]No console dev used.
OHOS # 
```

4　问题和解决方法

本次实验没有遇到什么较大的问题。

5　实验体会

通过本次实验，我深入理解了设备驱动程序的分类和特点，掌握了块设备驱动程序的设计和实现原理。同时，通过移植操作系统内核和文件系统的实践，提升了对嵌入式系统开发的理解和实际操作能力。