实用操作系统课程实验报告

实验名称：	实验七 鸿蒙 LiteOS-a 内核移植——存储系统移植
实验日期：	2023-12-8
实验地点：	文宣楼 B313

学号：	33920212204567
姓名：	任宇
专业年级：	软工 2021 级
学年学期：	2023-2024 学年第一学期

1.实验目的

- 进行鸿蒙 LiteOS-a 内核的存储系统移植，实现：
  - 用内存模拟 Flash

2. 实验内容和步骤

理论知识学习：

对于嵌入式设备，应用程序保存在 Flash 上面，内核如果想读取应用程序，首先得有能力读取存储设备。

在进行实践前，首先分析存储设备驱动程序，路径：openharmony/vendor/democom/demochip/driver/mtd/spi_nor/src/common/spinor.c。

在鸿蒙内核中，存储设备可分为字符设备、块设备和网络设备三类。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，而块设备驱动程序里读写数据以块为单位。无论对于什么设备，都要提供 open/read/write/ioctl 等函数。存储设备的类型是很多的，这些不同的存储设备访问方法各有不同，但肯定有读/写/擦除这三种操作，因此可以抽象出一个软件层：MTD(Memory Technology Device)，它封装了不同 Flash 的操作。查找 LiteOS-a 中相关定义：

```
book@ry-virtual-machine:~$ grep -nr "struct MtdDev {" openharmony
openharmony/kernel/liteos_a/fs/vfs/include/driver/mtd_dev.h:47:struct MtdDev {
```

```
47 struct MtdDev {
48     VOID *priv;
49     UINT32 type;
50
51     UINT64 size;
52     UINT32 eraseSize;
53
54     int (*erase)(struct MtdDev *mtd, UINT64 start, UINT64 len, UINT64 *failAddr);
55     int (*read)(struct MtdDev *mtd, UINT64 start, UINT64 len, char *buf);
56     int (*write)(struct MtdDev *mtd, UINT64 start, UINT64 len, const char *buf);
57 };
```

这里的定义类似于面向对象设计中的抽象类，不同的 Flash 要提供它自己的 MtdDev 结构体，实现其中的函数。JFFS2 文件系统直接使用 MTD 的，没有使用 block_operations，如图所示：

```
267 static int jffs2_fill_scan_buf(struct jffs2_sb_info *c, void *buf,
268                              uint32_t ofs, uint32_t len)
269 {
270   int ret;
271   size_t retlen;
272
273   ret = jffs2_flash_read(c, ofs, len, &retlen, buf);
274   if (ret) {
275     jffs2_dbg(1, "mtd->read(0x%x bytes from 0x%x) returned %d\n",
276               len, ofs, ret);
277     return ret;
278   }
279   if (retlen < len) {
280     jffs2_dbg(1, "Read at 0x%x gave only 0x%zx bytes\n",
```

```
141 #ifdef CONFIG_JFFS2_FS_WRITEBUFFER
142   if (jffs2_cleanmarker_oob(c)) {
143     int ret;
144
145     if (mtd_block_isbad(c->mtd, jeb->offset))
146       return BLK_STATE_BADBLOCK;
147
```

用内存模拟 Flash 需要指定要使用的内存地址和大小、实现 MtdDev 结构体。

实践——用内存模拟 Flash：

（1） 修改 reset_vector_up.S 文件：

文件路径为： openharmony/kernel/liteos_a/arch/arm/arm/src/startup/reset_vector_up.S，为 LOSCFG_PLATFORM_DEMOCHIP 定义新的内存映射配置：

```
194 #if defined(LOSCFG_PLATFORM_IMX6ULL) || defined(LOSCFG_PLATFORM_STM32MP157)
195     PAGE_TABLE_SET DDR_RAMFS_ADDR, DDR_RAMFS_VBASE, DDR_RAMFS_SIZE, MMU_INITIAL_MAP_DEVICE
196     PAGE_TABLE_SET LCD_FB_BASE, LCD_FB_VBASE, LCD_FB_SIZE, MMU_INITIAL_MAP_DEVICE
197 #endif
198 #if defined(LOSCFG_PLATFORM_DEMOCHIP)
199     PAGE_TABLE_SET DDR_RAMFS_ADDR, DDR_RAMFS_VBASE, DDR_RAMFS_SIZE, MMU_INITIAL_MAP_DEVICE
200 #endif
```

（2） 修改 board.h 文件：

文件路径为：openharmony/vendor/democom/demochip/board/include/board.h，定义 DEMOCHIP 平台的物理内存基址和大小以及

RAM 文件系统的地址和大小，如图所示：

```
/* physical memory base and size */
//#define DDR_MEM_ADDR            0x80000000
//#define DDR_MEM_SIZE            0x20000000
#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 DDR_RAMFS_SIZE 0x4000000  /* 60M for ramfs, 4M for lcd */
#define DDR_RAMFS_ADDR (DDR_MEM_ADDR + DDR_MEM_SIZE)
#define DDR_RAMFS_SIZE  0x8000000
```

（3） 修改 los_vm_zone.h 文件：

文件路径为：openharmony/kernel/liteos_a/kernel/base/include/los_vm_zone.h，为 DEMOCHIP 平台添加虚拟内存的宏定义，仿照其他平台：

```
#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)
#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)
```

（4） 修改 board.c 文件：

文件路径为：openharmony/vendor/democom/demochip/board/board.c，需要将原有针对 imx6ull 的函数名称更改为适用于 DEMOCHIP 的名称。将 imx6ull_mount_rootfs 更改为 demochip_mount_rootfs，并将 imx6ull_driver_init 更改为 demochip_driver_init。

```
void SystemInit()
{
#ifdef LOSCFG_FS_PROC
    dprintf("proc fs init ...\n");
    extern void ProcFsInit(void);
    ProcFsInit();
#endif

#ifdef LOSCFG_DRIVERS_MEM
    dprintf("mem dev init ...\n");
    extern int mem_dev_register(void);
    mem_dev_register();
#endif
    //imx6ull_driver_init();
    //imx6ull_mount_rootfs();
    demochip_driver_init();
    demochip_mount_rootfs();
```

在 demochip_mount_rootfs 函数中，根据条件编译 (#if 1) 更改了挂载根文件系统的方式，包括添加 MTD 分区和打开初始化文件。

```
15 static void demochip_mount_rootfs()
16 {
17 #if 1
18   int fd;
19     dprintf("register parition ...\n");
20     if (add_mtd_partition("spinor", 0, DDR_RAMFS_PART0_SIZE, 0))
21     {
22         PRINT_ERR("add_mtd_partition fail\n");
23     }
24
25     dprintf("mount /dev/spinorblk0 / ...\n");
26     //if (mount("/dev/spinorblk0", "/", "jffs2", MS_RDONLY, NULL))
27     if (mount("/dev/spinorblk0", "/", "jffs2", 0, NULL))
28     {
29         PRINT_ERR("mount failed\n");
30     }
31   fd = open("/bin/init", O_RDONLY);
32     dprintf("open /bin/init, fd = %d\n", fd);
33 #else
34     dprintf("mount /dev/ramdisk / ...\n");
35     //if (mount("/dev/spinorblk0", "/", "jffs2", MS_RDONLY, NULL))
36     if (mount("/dev/ramdisk", "/", "vfat", 0, NULL))
37     {
38         PRINT_ERR("mount failed\n");
39     }
40 #endif
41 }
```

（5） 修改 spinor.c 文件:、

文件路径为：openharmony/vendor/democom/demochip/driver/mtd/spi_nor/src/common/spinor.c，在 SPINOR 驱动的初始化函数 spinor_init 中，将 spinor_mtd.priv 和 spinor_mtd.size 设置为 DEMOCHIP 平台特定的值:

```
148 int spinor_init(void)
149 {
150     //spinor_mtd.priv = (void *)0 ; //100ask err, DDR_RAMFS_VBASE;
151   //spinor_mtd.size = 0; //100ask err, DDR_RAMFS_SIZE;
152     spinor_mtd.priv = (void *)DDR_RAMFS_VBASE;
153     spinor_mtd.size = DDR_RAMFS_SIZE;
154     /* ramnor register */
```

（6） 编译并运行:

```
book@ry-virtual-machine:~/openharmony/kernel/liteos_a$ make -j 8
make[1]: 进入目录“/home/book/openharmony/kernel/liteos_a”
/home/book/openharmony/kernel/liteos_a/tools/menuconfig/conf --silentoldconfig /home/book/openharmony/kernel/liteos_a/Kconfig
*
* Restart config...
*
*
* Compiler
*
LiteOS_Compiler_Type
  1. arm-linux-ohoseabi (COMPILER_HIMIX_32) (NEW)
> 2. clang-llvm (COMPILER_CLANG_LLVM)
choice[1-2?]:
```

```
make[1]: 进入目录“/home/book/openharmony/kernel/liteos_a/apps”
make[2]: 进入目录“/home/book/openharmony/kernel/liteos_a/apps/shell”
make[2]: 离开目录“/home/book/openharmony/kernel/liteos_a/apps/shell”
make[2]: 进入目录“/home/book/openharmony/kernel/liteos_a/apps/init”
make[2]: 离开目录“/home/book/openharmony/kernel/liteos_a/apps/init”
make[1]: 离开目录“/home/book/openharmony/kernel/liteos_a/apps”
book@ry-virtual-machine:~/openharmony/kernel/liteos_a$
```

将得到的 liteos.bin 文件放入烧写工具中运行：

```
uuu (Universal Update Utility) for nxp imx chips -- libuuu_1.3.136-0-g1ecc47f

Wait for Known USB Device Appear...
New USB Device Attached at 2:4
2:4>Start Cmd:FB:  download  -f  ../../files/rootfs.jffs2
2:4>Okay (0.051s)
2:4>Start Cmd:FB[-t 60000]: ucmd mw.l 98000000 ffffffff 1000000
2:4>Okay (0.177s)
2:4>Start Cmd:FB[-t 60000]: ucmd cp.b ${fastboot_buffer} 98000000 ${fastboot_byte
s}
2:4>Okay (0.008s)
2:4>Start Cmd:FB:  download  -f  ../../files/liteos.bin
2:4>Okay (0.05s)
2:4>Start Cmd:FB[-t 60000]: ucmd  cp.b  ${fastboot_buffer}  81000000  ${fastboot
_bytes}
2:4>Okay (0.011s)
```

可以正常执行指令，实现使用内存模拟 Flash。

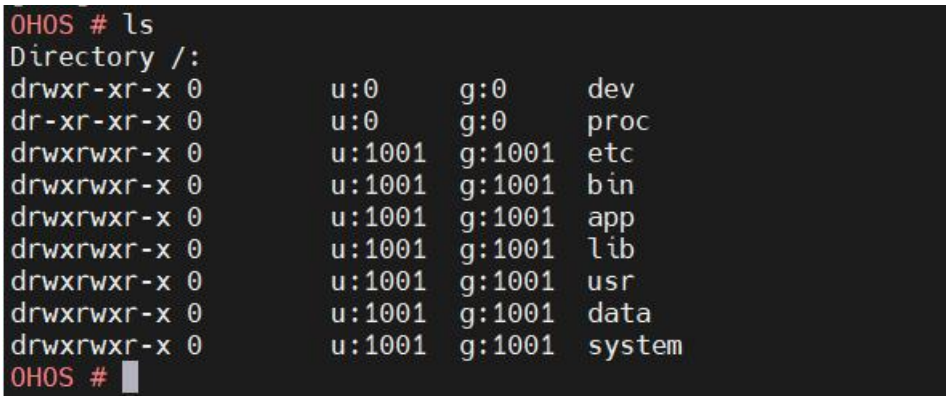
```
......
downloading of 891776 bytes finished
UCmd : UCmd: cp.b  ${fastboot_buffer}  81000000  ${fastboot_bytes}
UCmd : UCmd: go 81000000
## Starting application at 0x81000000 ...
Sm
******************Main******************


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

Processor   : Cortex-A7
Run Mode    : UP
GIC Rev     : GICv2
build time  : Nov 29 2023 21:46:33
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, totalSize: -56
[ERR][HDF:E/HDF_LOG_TAG]HdfAttributeManagerGetHostList get hdf manager node is null
[ERR]No drivers need load by hdf manager!DeviceManagerStart end ...
[ERR]No console dev used.
[ERR]No console dev used.
OHOS #
```



3. 实验总结

在本次实验中,我成功实现了鸿蒙 LiteOS-a 内核的存储系统移植，特别是通过内存模拟 Flash 的功能。这一过程涉及了对鸿蒙内核存储设备驱动程序的深入理解和修改。通过这次实验，我不仅加深了对鸿蒙操作系统内核存储系统的理解,而且提升了在实际嵌入式环境中进行系统移植和驱动程序开发的能力。实验结果表明，我们能够有效地利用内存来模拟 Flash 存储，从而为后续的移植提供强有力的基础。

4. 遇到的困难及解决方法

无