《嵌入式系统》实验报告 6

学号：21013134 姓名：徐昊博 班级：计 213 日期：2024.6.7

成绩：_____ 指导教师：罗飞

实验名称：嵌入式驱动程序的移植	**实验地点：**

实验仪器：ECS 云计算环境（提供 Linux 实验环境）；开发板

一、实验目的：

1．了解 Linux 驱动程序的结构
2．掌握 Linux 驱动程序常用结构体和操作函数的使用方法
3．初步掌握 Linux 驱动程序的移植方法及过程
4．掌握 Linux 驱动程序的加载方法

二、实验内容：

查看自己虚拟机 ip： 172.18.120.215

基于/home/ecust/samba_share/embed/Linux/linux-3.2.tar.bz2 来完成

1. 准备代码

ecust@Ubuntu:~$ cd workplace

ecust@Ubuntu:~/workplace$ tar jxvf /home/ecust/samba_share/embed/Linux/linux-3.2.tar.bz2

（1）拷贝 regs-nand.h 到 arch/arm/mach-s5pv210/include/mach/

ecust@Ubuntu:~/workplace$cd linux-3.2/arch/arm/mach-s5pv210/include/mach

ecust@Ubuntu:~/workplace/linux-3.2/arch/arm/mach-s5pv210/include/mach$ wget -0 regs-nand.h http://172.18.120.26:10005/experiments/experiment/downloadFileResource?fileId=1892

（2）拷贝 s3c_nand.c 到 drivers/mtd/nand/

ecust@Ubuntu:~/workplace/linux-3.2/arch/arm/mach-s5pv210/include/mach$ cd /home/ecust/workplace/linux-3.2/drivers/mtd/nand/

ecust@Ubuntu:~/workplace/linux-3.2/drivers/mtd/nand$ wget -0 s3c_nand.c http://172.18.120.26:10005/experiments/experiment/downloadFileResource?fileId=1893

2. 修改平台代码：

简述主要步骤及关键代码含义。

打开 arch/arm/mach-s5pv210/mach-smdkv210.c

新建 c 文件，按下“i”进入编辑模式，编辑内容如下：

（1） 添加头文件

```
#ifdef CONFIG_MTD_NAND
#include <linux/mtd/mtd.h>
#include <linux/mtd/nand.h>
#include <linux/mtd/nand_ecc.h>
#include <linux/mtd/partitions.h>
#include <plat/nand.h>
#endif
```

（2） 添加平台设备

```
#ifdef CONFIG_MTD_NAND
```

```
static struct mtd_partition s5pv210_partition_info[] = {
[0] = {
.name= "Bootloader",
.offset= 0,
.size= SZ_1M,
},
[1] = {
  .name= "Kernel",
.offset= MTDPART_OFS_APPEND,
.size= SZ_1M * 3,
},
[2] = {
.name= "Rootfs",
.offset= MTDPART_OFS_APPEND,
.size= SZ_8M,
},
[3] = {
.name= "Userfs",
.offset= MTDPART_OFS_APPEND,
.size= MTDPART_SIZ_FULL,
},
};
struct s3c_nand_mtd_info s5pv210_nand_info = {
.chip_nr = 1,
.mtd_part_nr = ARRAY_SIZE(s5pv210_partition_info),
.partition = s5pv210_partition_info
};
struct resource s5pv210_nand_resource[] = {
[0] = {
.start = 0xB0E00000,
.end= 0xB0E00000 + SZ_1M,
.flags= IORESOURCE_MEM,
},
};
struct platform_device s5pv210_device_nand = {
.name= "s5pv210-nand",
.id= -1,
.num_resources = ARRAY_SIZE(s5pv210_nand_resource),
.resource = s5pv210_nand_resource,
.dev= {
.platform_data = &s5pv210_nand_info,
},
};
#endif
```

上述代码的含义：

①检查内核配置

在#ifdef CONFIG_MTD_NAND 宏定义的条件下，包含了代码，这意味着只有在内核配置启用了 MTD NAND 支持的情况下，这些代码才会生效。

②定义 NAND 分区信息

定义了一个数组 s5pv210_partition_info，用于描述 NAND 闪存设备上的分区。这个数组包含四个分区，每个分区有自己的名称、起始地址和大小：

Bootloader: 名称为“Bootloader”，起始地址为 0，大小为 1MB。

Kernel: 名称为“Kernel”，紧跟在 Bootloader 分区之后，大小为 3MB。

Rootfs: 名称为“Rootfs”，紧跟在 Kernel 分区之后，大小为 8MB。

Userfs: 名称为“Userfs”，紧跟在 Rootfs 分区之后，使用剩余的全部空间。

③ 定义 NAND 设备信息

定义了一个结构体 s5pv210_nand_info，用于描述 NAND 设备的具体信息。这个结构体包含以下字段：

chip_nr: 指定 NAND 芯片的数量，这里为 1。

mtd_part_nr: 指定分区的数量，即 s5pv210_partition_info 数组的大小。

partition: 指向定义好的分区信息数组 s5pv210_partition_info。

④ 定义 NAND 设备资源

定义了一个数组 s5pv210_nand_resource，用于描述 NAND 设备的资源。这个数组包含一个资源，资源的起始地址为 0xB0E00000，结束地址为 0xB0E00000 + 1MB，类型为内存资源。

⑤ 定义平台设备

定义了一个结构体 s5pv210_device_nand，将 NAND 设备注册为平台设备。这个结构体包含以下字段：

name: 设备名称为"s5pv210-nand"。

id: 设备 ID 为-1，表示自动分配 ID。

num_resources: 资源数量，即 s5pv210_nand_resource 数组的大小。

resource: 指向定义好的资源数组 s5pv210_nand_resource。

dev.platform_data: 指向设备的私有数据，即 s5pv210_nand_info。

⑥ 结束条件编译块

使用#endif 结束条件编译块，表示在 CONFIG_MTD_NAND 配置启用的情况下，包含上述所有定义。这确保了代码仅在内核启用 MTD NAND 支持时生效。

（3） 在结构体数组smdkv210_devices[]中添加

```
static struct platform_device *smdkv210_devices[] __initdata = {
……
#if defined(CONFIG_MTD_NAND)
&s5pv210_device_nand,
#endif
};
```

上述代码的含义：

定义平台设备数组：

static struct platform_device *smdkv210_devices[] __initdata：定义一个指向 platform_device 结构体的指针数组，并将其标记为__initdata。__initdata 表示该数组中的数据仅在内核初始化期间使用，初始化完成后这些数据可以被释放，从而节省内存。

条件编译：

#if defined(CONFIG_MTD_NAND)：检查内核配置是否启用了 MTD NAND 支持。如果启用了，包含后续的代码。

&s5pv210_device_nand,：将先前定义的 NAND 设备 s5pv210_device_nand 的地址添加到设备数组中。

结束条件编译：

#endif：结束条件编译块，表示在 CONFIG_MTD_NAND 配置启用的情况下，包含前面的代码行。

（4）修改arch/arm/plat-samsung/include/plat/nand.h

简述主要步骤：

打开arch/arm/plat-samsung/include/plat/nand.h

按下“i”进入编辑模式，添加如下内容：

```
struct s3c_nand_mtd_info {
uint chip_nr;
uint mtd_part_nr;
struct mtd_partition *partition;
};
```

（5） 修改include/linux/mtd/partitions.h

简述主要步骤：

打开 include/linux/mtd/partitions.h

按下“i”进入编辑模式，添加如下内容：

```
int add_mtd_partitions(struct mtd_info *, const struct mtd_partition *, int);
```

（6） 修改arch/arm/mach-s5pv210/clock.c中结构体

简述主要步骤：

按下“i”进入编辑模式，在最后添加如下代码(加粗部分)

```
static struct clk init_clocks_off[] = {
{
……
.ctrlbit= (1 << 0),
},{
.name= "nandxl",
.id= -1,
.parent= &clk_hclk_psys.clk,
.enable= s5pv210_clk_ip1_ctrl,
.ctrlbit= (1 << 24),
},
};
```

（7）修改drivers/mtd/nand/Kconfig

简述主要步骤和关键代码含义：

打开drivers/mtd/nand/Kconfig按下“i”进入编辑模式，在213行添加如下代码：

```
config MTD_NAND_S3C
tristate "NAND Flash support for S3C SoC"
depends on MTD_NAND && (ARCH_S5PC1XX || ARCH_S5PC11X || ARCH_S5PV2XX ||
ARCH_S5PV210)
help
This enables the NAND flash controller on the S3C.
No board specfic support is done by this driver, each board
must advertise a platform_device for the driver to attach.
   config MTD_NAND_S3C_DEBUG
```

bool "S3C NAND driver debug"
depends on MTD_NAND_S3C
help
Enable debugging of the S3C NAND driver
config MTD_NAND_S3C_HWECC
bool "S3C NAND Hardware ECC"
depends on MTD_NAND_S3C
help
Enable the use of the S3C's internal ECC generator when
using NAND. Early versions of the chip have had problems with
incorrect ECC generation, and if using these, the default of
software ECC is preferable.
If you lay down a device with the hardware ECC, then you will
currently not be able to switch to software, as there is no
implementation for ECC method used by the S3C

上述代码的含义：

配置 MTD_NAND_S3C

定义一个新的配置选项，允许用户选择是否启用 S3C SoC（系统级芯片）的 NAND Flash 支持。这一选项依赖于 MTD NAND 支持以及特定的架构(如 ARCH_S5PC1XX、ARCH_S5PC11X、ARCH_S5PV2XX、ARCH_S5PV210）。启用该选项后，S3C 上的 NAND Flash 控制器将被启用。不过，这个驱动程序不包含具体板卡的支持，每个板卡需要提供一个平台设备供驱动程序附加。

配置 MTD_NAND_S3C_DEBUG

定义一个布尔配置选项，用于启用或禁用 S3C NAND 驱动程序的调试功能。这一选项依赖于之前定义的 S3C SoC 的 NAND Flash 支持选项。启用该选项后，将允许进行 S3C NAND 驱动程序的调试，方便开发者进行故障排查和优化。

配置 MTD_NAND_S3C_HWECC

定义另一个布尔配置选项，用于启用或禁用 S3C NAND 硬件 ECC（纠错码）功能。这一选项同样依赖于 S3C SoC 的 NAND Flash 支持选项。启用该选项后，将使用 S3C 内部的 ECC 生成器来处理 NAND Flash 的错误校正。需要注意的是，早期版本的芯片在 ECC 生成方面存在问题，建议使用软件 ECC。如果设备使用了硬件 ECC，目前无法切换到软件 ECC，因为 S3C 没有实现这种切换的方法。

7. 修改drivers/mtd/nand/Makefile

简述主要步骤和关键代码含义：

先打开Makefile，按下“i”进入编辑模式在

obj-$(CONFIG_MTD_NAND_S3C2410)+= s3c2410.o

下面添加：

obj-$(CONFIG_MTD_NAND_S3C)+= s3c_nand.o

含义通过在 Makefile 中添加 obj-$(CONFIG_MTD_NAND_S3C)+= s3c_nand.o，确保当配置选项 CONFIG_MTD_NAND_S3C 被启用时，s3c_nand.o 文件会被编译并链接到最终的内核模块中。这一步是为了在编译内核时包括新的 S3C SoC 的 NAND Flash 支持。

（9） 将S3C NAND配置到内核中

简述主要步骤：

运行 make menuconfig 命令。

在菜单中导航到 Device Drivers -> Memory Technology Device (MTD) support -> NAND Flash device support。

找到并启用 NAND Flash support for S3C SoC 以及相关的调试和硬件 ECC 选项。

保存配置并退出。

（10） 编译

执行以下命令开始编译：make

三、思考

1. 在内核中使用 S3C NAND 驱动有几种方式？分别如何操作？

有两种方式：

1. 编译进内核

将 S3C NAND 驱动直接编译进内核，这样驱动在内核启动时立即加载。

进入内核配置菜单。

在配置菜单中找到 NAND 闪存支持选项，进入设备驱动（Device Drivers）->内存技术设备支持（Memory Technology Device (MTD) support）->NAND 闪存设备（NAND Flash Device），选择 三星 S3C NAND 驱动（Samsung S3C NAND driver）并将其设置为内嵌模式（即选择项前显示为[*]）。

保存配置并退出配置菜单。

编译内核，生成的内核映像会包含 S3C NAND 驱动。

将新的内核映像安装到设备中并重启。

2. 编译为模块

将 S3C NAND 驱动编译为内核模块，驱动在需要时通过加载模块来访问 NAND 闪存。

进入内核配置菜单。

在配置菜单中找到 NAND 闪存支持选项，进入设备驱动（Device Drivers）->内存技术设备支持（Memory Technology Device (MTD) support）->NAND 闪存设备（NAND Flash Device），选择 三星 S3C NAND 驱动（Samsung S3C NAND driver）并将其设置为模块模式（即选择项前显示为[M]）。

保存配置并退出配置菜单。

编译内核和模块，生成的模块文件将独立于内核映像。

安装模块文件到系统的模块目录中。

在需要使用 NAND 闪存时，加载模块。

任课教师签名：　　2019 年 11 月 9 日