

Android 的 Linux 内核与驱动程序

Android 的 Linux 内核与驱动程序

- 第一部分 Linux 核心与驱动
- 第二部分 Android 专用驱动
- 第三部分 Android 使用的设备驱动

第一部分 Linux 核心与驱动

Android 使用标准的 Linux2.6 内核，
作为其操作系统。

Android 1.0 (release-1.0)

使用 Linux2.6.25

Android 1.5 (sdk-1.5_r1)

使用 Linux2.6.27

Android 1.6 (sdk-1.6_r1)

使用 Linux2.6.29

第一部分 Linux 核心与驱动

获取通用内核的代码：

```
$ git clone git://android.git.kernel.org/kernel/common.git
```

kernel/common.git 为通用 Kernel 的工程名称。

Android1.5 之前的版本具有 kernel 目录，其中也是参考的 kernel。

第一部分 Linux 核心与驱动

Android Linux 内核的配置和编译：

```
$ make ARCH=arm goldfish_defconfig .config  
$ make ARCH=arm CROSS_COMPILE={path}/arm-none-linux-gnueabi-
```

Android 通用的 Kernel 使用的处理器为 **goldfish**，这是一种 ARM 处理器。这个 Linux 编译生成的结果在 Android 的模拟器中使用。

第一部分 Linux 核心与驱动

Goldfish 处理器的编译结果: **vmlinux** 为内核的 **ELF** 文件, **zImage** 为内核的压缩映像文件。

```
LD    vmlinux
SYSMAP System.map
SYSMAP .tmp_System.map
OBJCOPY arch/arm/boot/Image
Kernel: arch/arm/boot/Image is ready
AS    arch/arm/boot/compressed/head.o
GZIP  arch/arm/boot/compressed/piggy.gz
AS    arch/arm/boot/compressed/piggy.o
CC    arch/arm/boot/compressed/misc.o
LD    arch/arm/boot/compressed/vmlinux
OBJCOPY arch/arm/boot/zImage
Kernel: arch/arm/boot/zImage is ready
```

vmlinux 和 **zImage** 分别对应 **Android** 代码 **prebuilt** 中的预编译的 **arm** 内核。

第一部分 Linux 核心与驱动

Android 模拟器使用的处理器的 goldfish 的配置文件为：
[arch/arm/mach-goldfish](#)

```
#  
# System Type  
#  
CONFIG_ARCH_GOLDFISH=y  
#  
# Android  
#  
CONFIG_ANDROID=y  
CONFIG_ANDROID_BINDER_IPC=y  
CONFIG_ANDROID_LOGGER=y  
# CONFIG_ANDROID_RAM_CONSOLE is not set  
CONFIG_ANDROID_TIMED_OUTPUT=y  
CONFIG_ANDROID_LOW_MEMORY_KILLER=y  
#  
# Networking options  
#  
CONFIG_ANDROID_PARANOID_NETWORK=y  
#  
# UBI - Unsorted block images  
#  
CONFIG_ANDROID_PMEM=y
```

`goldfish` 是一种 ARM 处理器，其核心内容的路径为：

[arch/arm/mach-goldfish](#)

`Android` 还需要在标准的 `Linux` 内核中需要增加必要的驱动，用于对系统用户程序的支持。在 `android` 中的驱动主要分成两种类型：

- ❑ `Android` 专用驱动
- ❑ `Android` 使用的设备驱动

第二部分 Android 专用驱动

2.1 Ashmem

2.2 binder

2.3 logger

第二部分 Android 专用驱动

Ashmem :

匿名共享内存驱动

Logger :

轻量级的 log 驱动

Binder 驱动 (Binder Driver) :

基于 OpenBinder 驱动，为 Android 平台提供 IPC 的支持

能源管理 (Android Power Management) :

轻量级的能源管理，基于 Linux 的能源管理，为嵌入式系统做了优化

Android Power Management (PM) :

定时器驱动，用于唤醒设备

Low Memory Killer :

在缺少内存的情况下，杀死进程

Android PMEM :

物理内存驱动

2.1 Ashmem

Android 的 Ashmem 的含义为：
Anonymous Shared Memory 匿名共享内存，通过内核的机制，为用户空间程序提供分配内存的机制。

Ashmem 设备节点名称：

[/dev/ashmem](#)

主设备号为 **10** (Misc Driver)

次设备号动态生成

2.1 Ashmem

Ashmem 的代码路径:

[kernel/include/linux/ashmem.h](#)

[kernel/mm/ashmem.c](#)

在用户空间 C `libutil` 库对 `Ashmem` 封装并提供接口:

[system/core/include/cutils/ashmem.h](#)

[system/core/libcutils/ashmem-dev.c](#)

[system/core/libcutils/ashmem-host.c](#)

2.1 Binder

Android 的 Binder 驱动程序为用户层程序提供了 IPC（进程间通信）的支持，**Android** 整个系统的运行依赖 Binder 驱动

Binder 设备节点名称：

/dev/binder

主设备号为 **10**（**Misc Driver**）

次设备号动态生成

2.1 Ashmem

binder 的代码路径:

[kernel/include/linux/binder.h](#)

[kernel/drivers/misc/binder.c](#)

在用户空间 **libutil** 工具库和 **Service Manager** 守护进程调用 **Binder** 接口提供对整个系统的支持:

[frameworks/base/cmds/servicemanager/](#)

[frameworks/base/include/utls/](#)

[frameworks/base/libs/utls/](#)

2.1 Logger

Android 的 **Logger** 驱动程序为用户层程序提供 Log 的支持，这个驱动作为一个工具来使用。

Logger 有三个设备节点：

[/dev/log/main](#)

[/dev/log/event](#)

[/dev/log/radio](#)

主设备号为 **10** (Misc Driver)

次设备号动态生成

2.1 Ashmem

Logger 驱动的代码路径:

[kernel/include/linux/logger.h](#)
[kernel/drivers/misc/logger.c](#)

在用户空间 logcat 程序调用 Logger 驱动:

[system/core/logcat/](#)

第三部分 Android 使用的设备驱动

- 3.1 framebuffer 驱动
- 3.2 Event 输入设备驱动
- 3.3 v4l2 摄像头—视频驱动
- 3.4 OSS 音频驱动
- 3.5 ALSA 音频驱动
- 3.6 MTD 驱动
- 3.7 蓝牙驱动
- 3.8 Wlan 驱动

3.1 framebuffer 显示驱动

显示驱动使用 framebuffer 驱动。

framebuffer 驱动的设备节点:

[/dev/fb0](#)

[/dev/graphics/fb0](#)

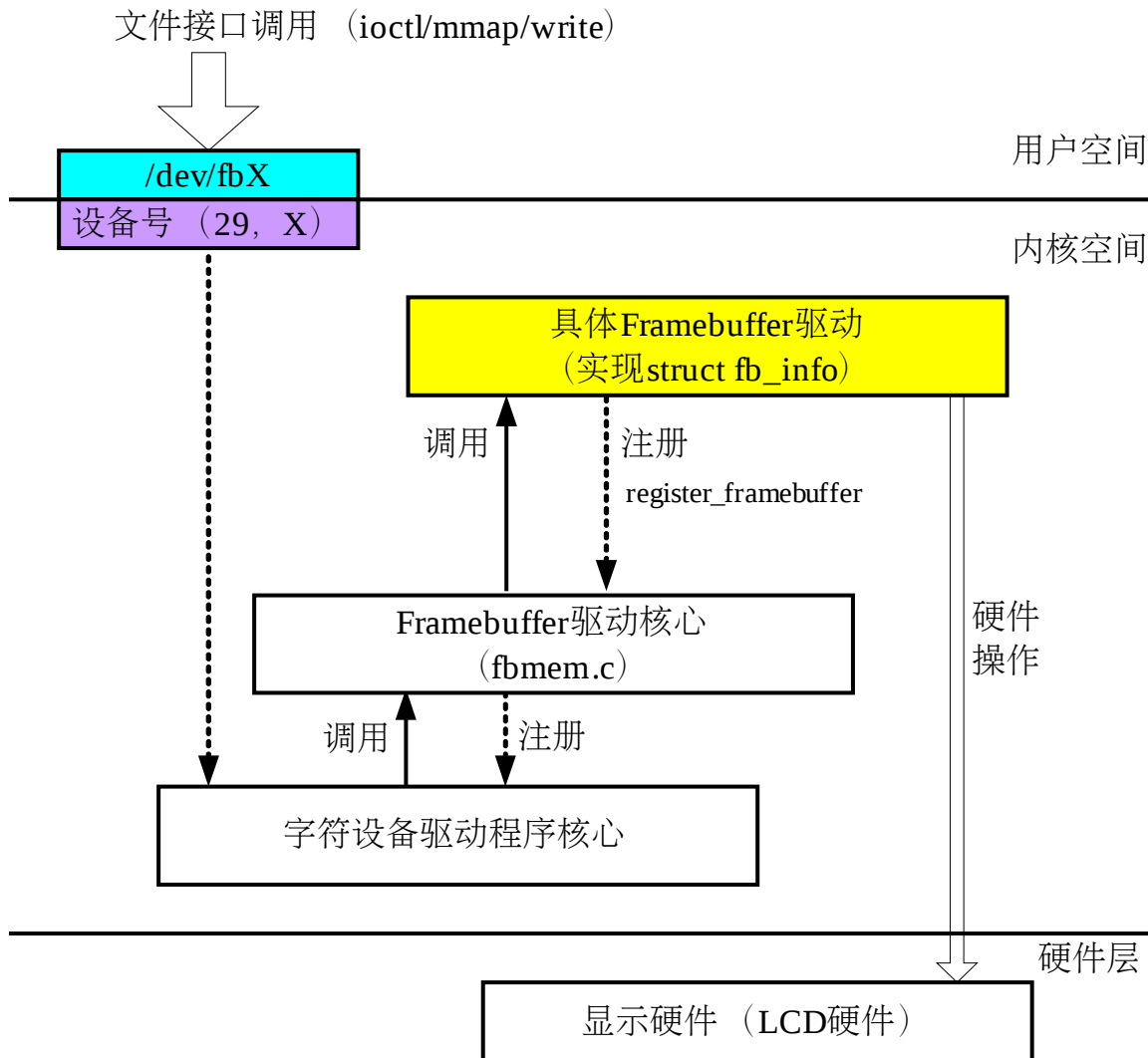
主设备号为 29，次设备号递增生成。

代码路径:

[include/linux/fb.h](#)

[drivers/video/fbmem.c](#)

3.1 framebuffer 显示驱动



3.2 Event 输入设备驱动

输入设备的驱动通常使用 **Input** 设备中的 **Event** 设备。

Event 的字符设备的设备节点:

[/dev/input/eventX](#)

主设备号为 **13**，设备节点为 **64-95**

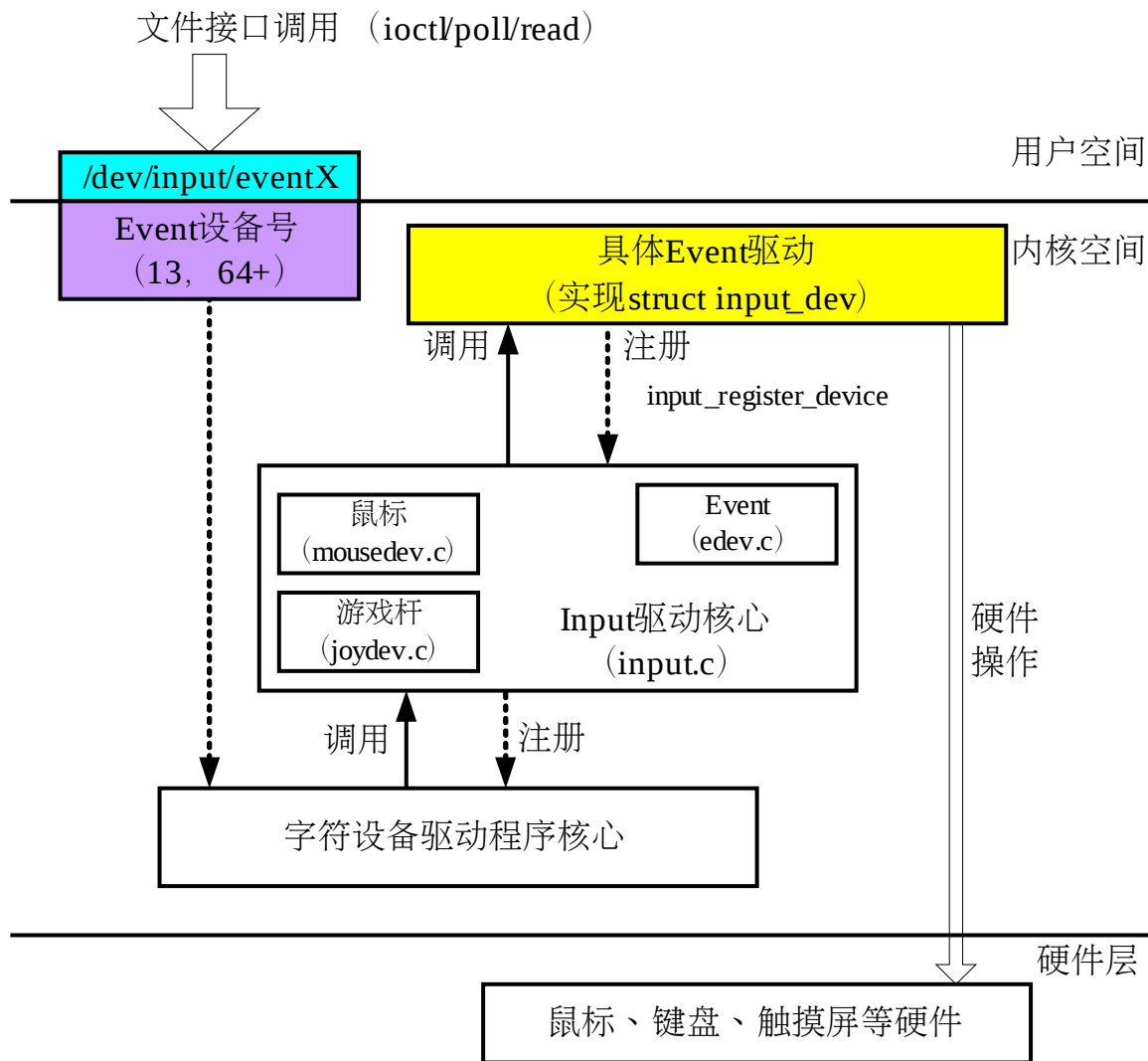
Input 驱动程序的头文件: [include/linux/input.h](#)

代码路径:

[drivers/input/input.c](#): 核心代码

[drivers/input/evdev.c](#): **Event** 部分的实现。

3.2 Event 输入设备驱动



3.3 v4l2 摄像头 — 视频驱动

摄像头（Camera）— 视频驱动通常使用 Video For Linux。

v4l2 驱动的设备节点：

[/dev/video/videoX](#)

主设备号为 **81**，次设备号 **0-63**。

v4l2 驱动主要头文件路径：

[include/linux/videodev.h](#)：v4l 第一版的头文件

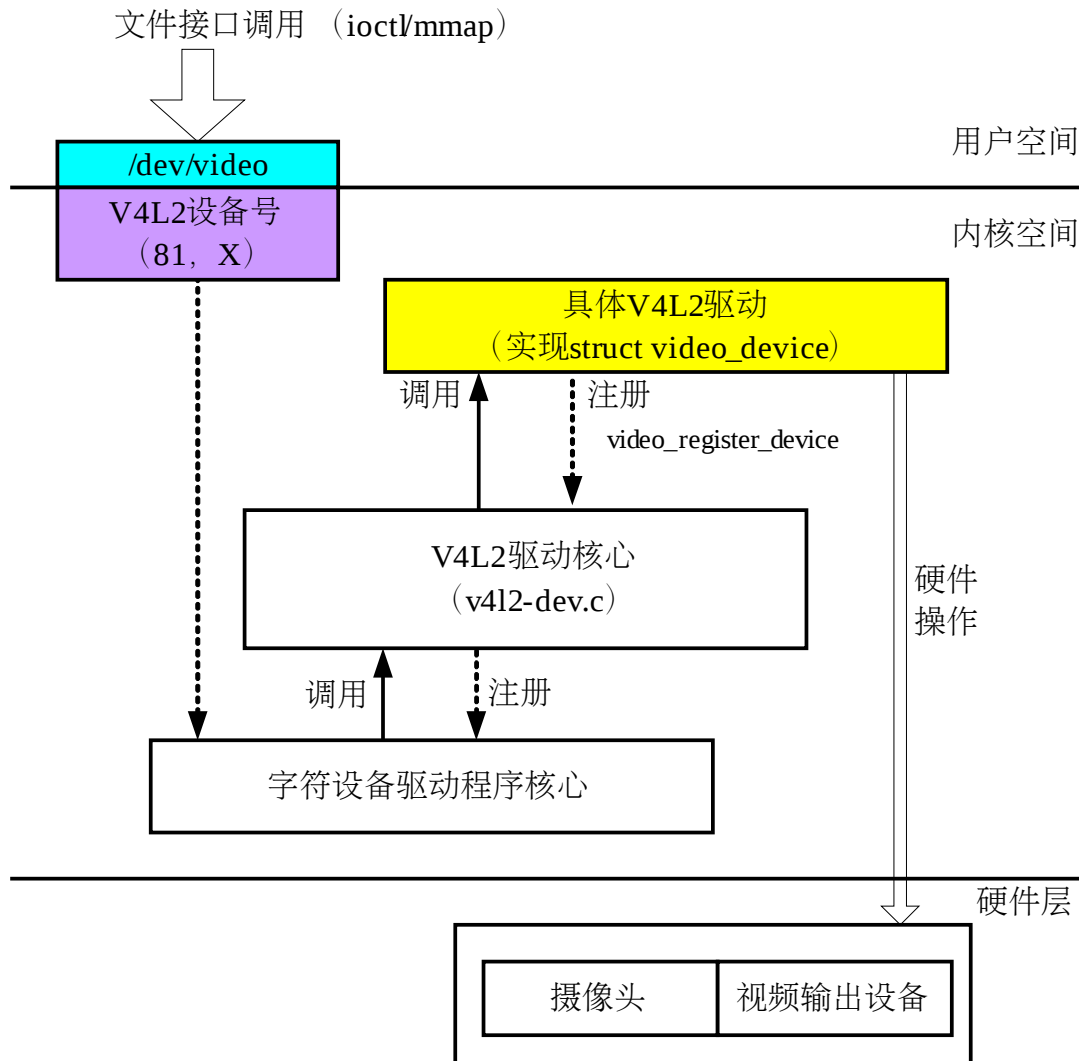
[include/linux/videodev2.h](#)：定义主要的数据接口和常量

[include/media/v4l2-dev.h](#)：设备头文件，具体设备使用其中的接口注册

v4l2 驱动核心实现路径：

[driver/media/video/v4l2-dev.c](#)

3.3 v4l2 摄像头 — 视频驱动



3.4 OSS 音频驱动

OSS (Open Sound System) 开放声音系统。

OSS 驱动的设备节点:

[/dev/mixer](#)

[/dev/sndstat](#)

[/dev/dsp](#)

OSS 主设备号为 **14** , 次设备号为各个设备。

OSS 驱动程序的主要头文件:

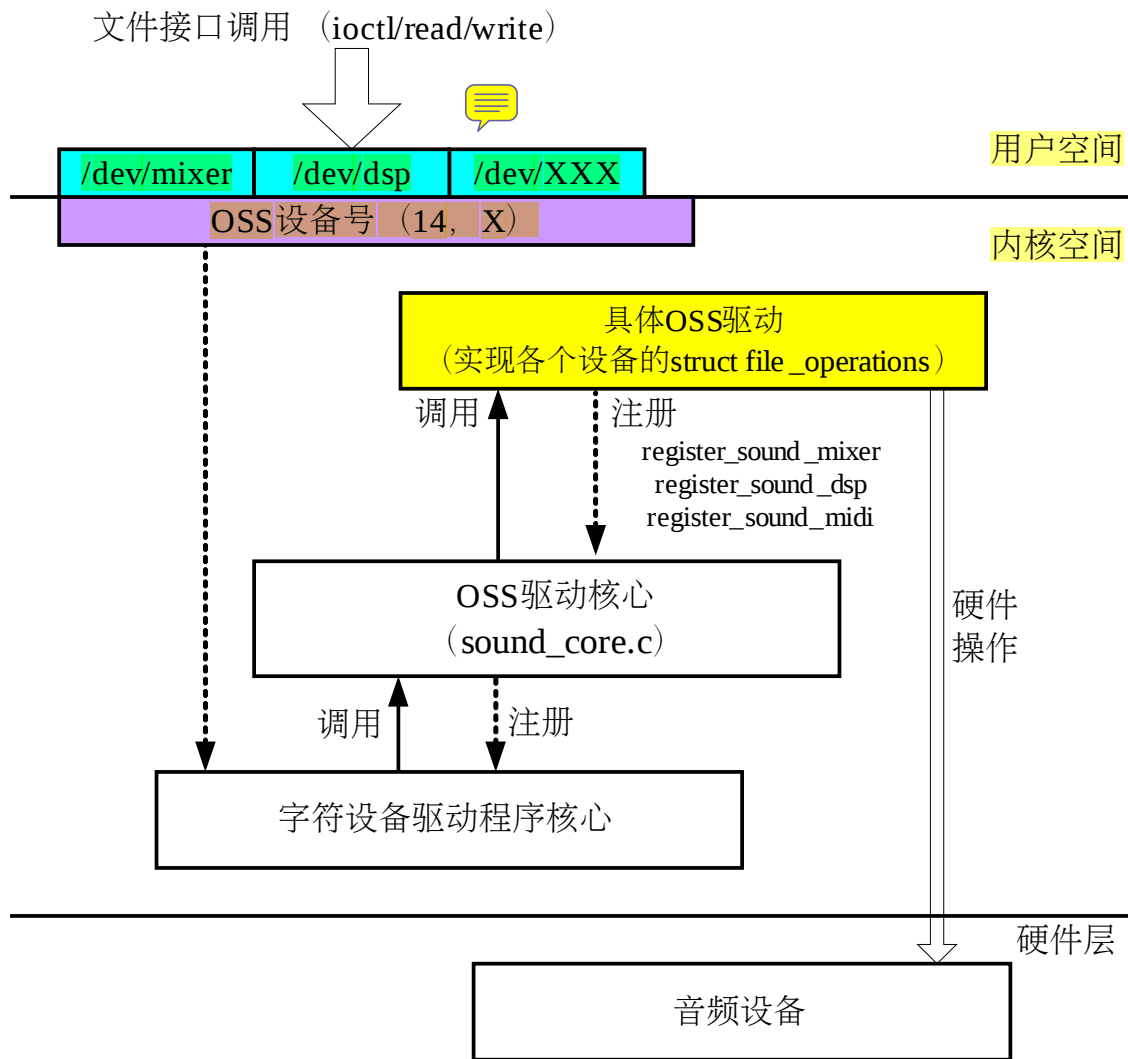
[include/linux/soundcard.h](#) : OSS 驱动的主要头文件

[include/linux/sound.h](#) : 定义 OSS 驱动的次设备号和注册函数

OSS 驱动程序的核心:

[sound/sound_core.c](#)

3.4 OSS 音频驱动



3.5 ALSA 音频驱动

ALSA (Advanced Linux Sound Architecture) 高级 Linux 声音体系。

ALSA 驱动的设备节点:

[/dev/snd/controlCX](#)

[/dev/snd/pcmXXXc](#)

[/dev/snd/pcmXXXp](#)

[/dev/snd/seq](#)

[/dev/snd/timer](#)

主设备号为 **116**，次设备号为各个设备。

ALSA 驱动程序的头文件:

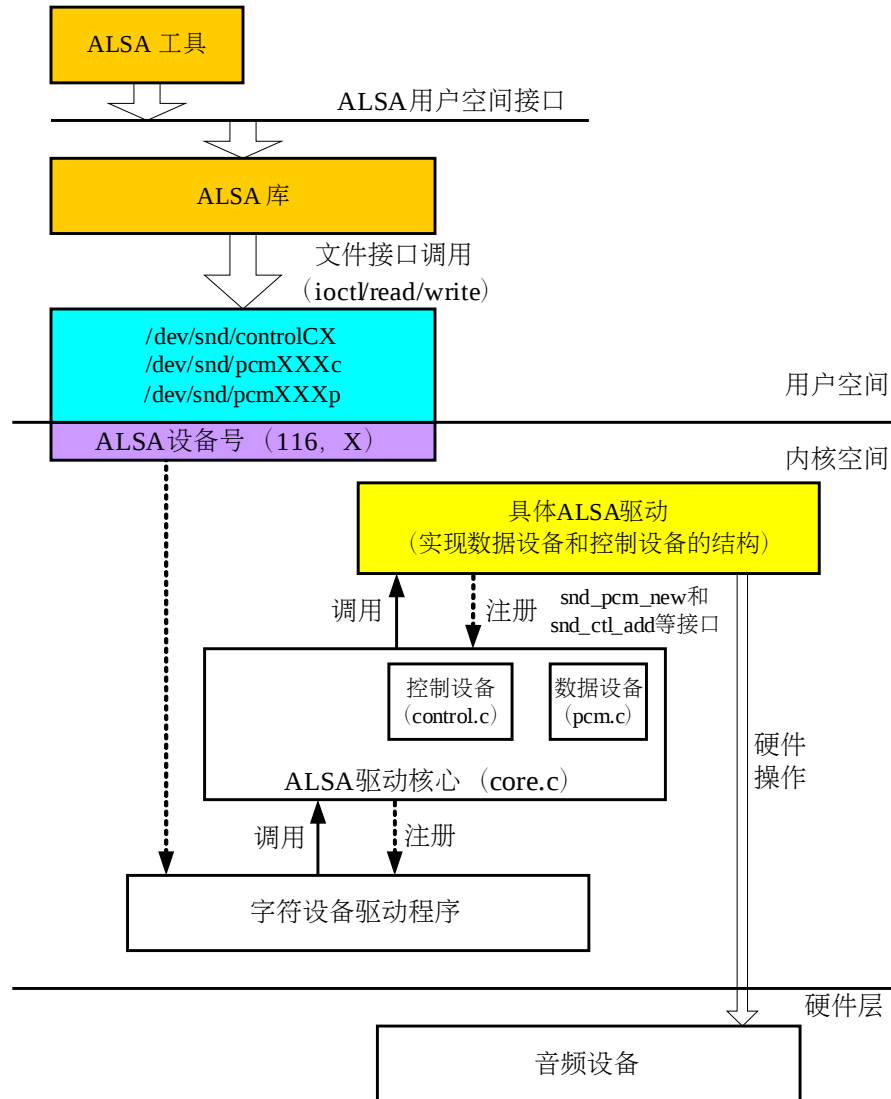
[include/sound/asound.h](#): ALSA 驱动的主要头文件

[include/sound/core.h](#): ALSA 驱动核心数据结构和具体驱动的注册函数

ALSA 驱动程序的核心实现:

[sound/core/sound.c](#)

3.5 ALSA 音频驱动



3.5 MTD 驱动

Flash 驱动通常使用 MTD (memory technology device)，内存技术设备。

MTD 的字符设备:

/dev/mtdX

主设备号为 90。

MTD 的块设备:

/dev/block/mtdblockX

主设备号为 13。

MTD 驱动程序头文件路径: [include/linux/mtd/mtd.h](#)

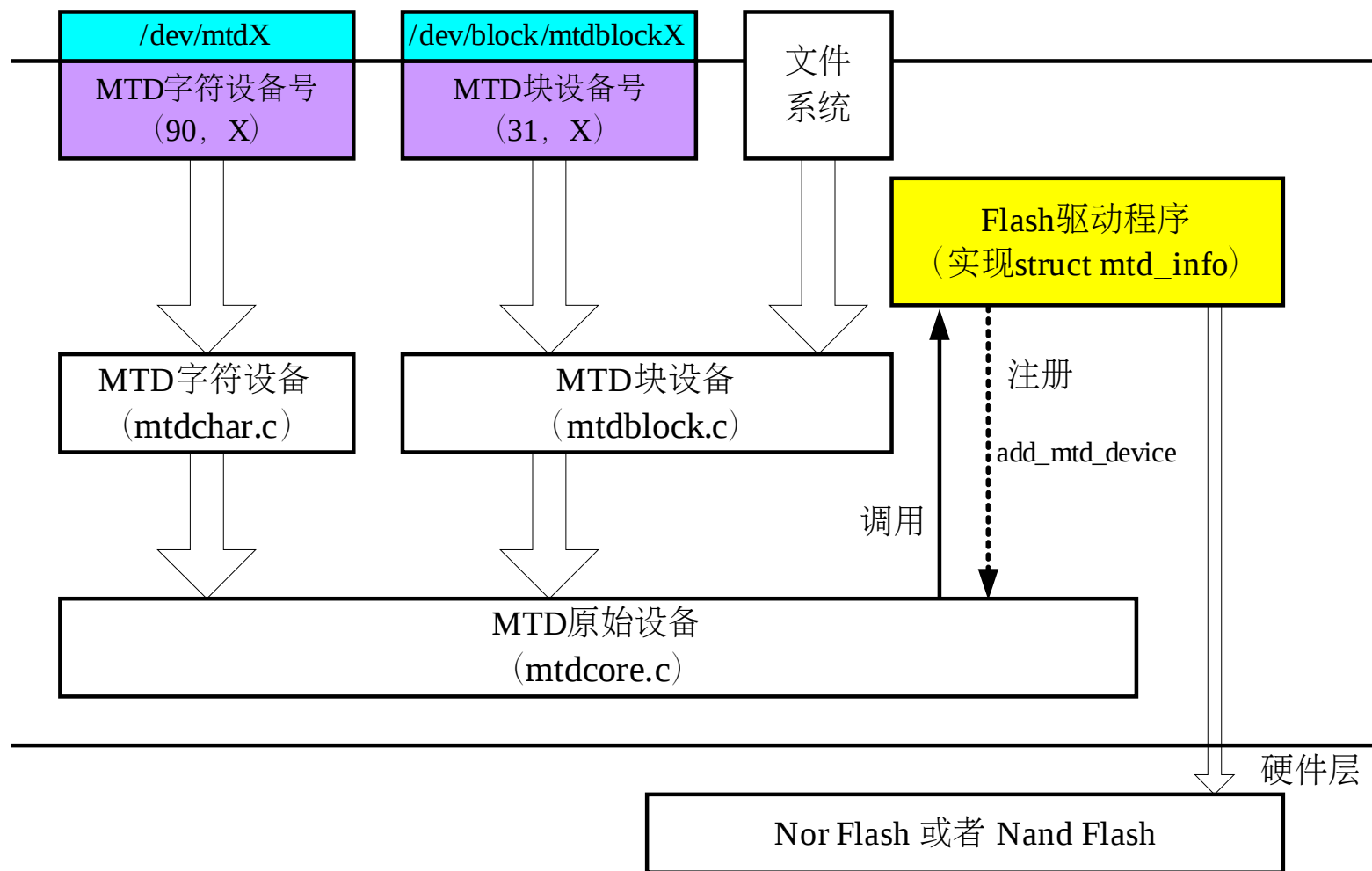
MTD 源代码路径:

[drivers/mtd/mtdcore.c](#): MTD 核心, 定义 MTD 原始设备

[drivers/mtd/mtdchar.c](#): MTD 字符设备

[drivers/mtd/mtdblock.c](#): MTD 块设备

3.5 MTD 驱动



3.6 蓝牙驱动

在 **Linux** 中，蓝牙设备驱动是网络设备，使用网络接口。

蓝牙设备的网络协议：

协议族：**AF_BLUETOOTH** （31）

蓝牙协议部分头文件：

[include/net/bluetooth/hci_core.h](#)

[include/net/bluetooth/bluetooth.h](#)

蓝牙协议源代码文件：

[net/bluetooth/*](#)

蓝牙驱动程序部分的文件：

[drivers/bluetooth/*](#)

3.6 蓝牙驱动

蓝牙的驱动程序一般都通过标准的 HCI 控制实现。但根据硬件接口和初始化流程的不同，又存在一些差别。这类初始化动作一般是一些晶振频率，波特率等基础设置。比如 CSR 的芯片一般通过 BCSP 协议完成最初的初始化配置，再激活标准 HCI 控制流程。对 Linux 来说，一旦 bluez 可以使用 HCI 与芯片建立起通信（一般是 `hciattach + hciconfig`），便可以利用其上的标准协议（SCO，L2CAP 等），与蓝牙通信，使其正常工作了。

3.7 Wlan 驱动

在 Linux 中，Wlan 设备驱动是网络设备，使用网络接口。Wlan 在用户空间使用标准的 socket 接口进行控制。

WIFI 协议部分头文件:

[include/net/wireless.h](#)

WIFI 协议部分源文件:

[net/wireless/*](#)

WIFI 驱动程序部分:

[drivers/net/wireless/*](#)

谢谢！