《Android 系统移植和驱动开发》

作者：华清远见

第 5 章 Android 移植与驱动

本章目标

在第 4 章我们搭建了移植 Android 系统的环境，并且在 FS_S5PC100 开发平台上进行了 Android 系统内核和文件系统的编译和烧写，这只是 Android 系统移植的第一步，在进行 Android 移植的过程中难点不在于编译 Android 内核和文件系统，更不是把 Android 镜像烧写到开发板上，而是对 Android 内核中驱动程序的修改。

5.1 Android 移植和驱动的关系

假设我们出产基于 Android 的代号为“大米”的手机的摄像头是 800 万像素，伴随着硬件的发展，我们打算升级这款手机为“老米”手机，它的摄像头升级为 1000 万像素，那么这时原先“大米”手机的 Android 镜像中的摄像头驱动就要进行修改，然后重新进行编译和烧写，所以在进行 Android 移植中的难点在于驱动程序的编写和修改。

5.2 设备驱动程序

5.2.1 设备驱动概念

任何一个计算机系统的运行都是系统中软硬件协作的结果，没有硬件的软件是空中楼阁，而没有软件的硬件则只是一堆废铁。硬件是底层基础，是所有软件得以运行的平台，代码最终会落实为硬件上的组合逻辑与时序逻辑；软件则实现了具体应用，它按照各种不同的业务需求而设计，满足了用户的需求。硬件较固定，软件则很灵活，可以适应各种复杂多变的应用。可以说，计算机系统的软硬件互相成就了对方。但是，软硬件之间同样存在着悖论，那就是软件和硬件不应该互相渗透到对方的领地。为了尽可能快速地完成设计，应用软件工程师不想也不必关心硬件，而硬件工程师也难有足够的闲暇和能力来顾及软件。

例如，应用软件工程师在调用套接字发送和接收数据包时，不必关心网卡上的中断、寄存器、存储空间、I/O 端口、片选，以及其他任何硬件词汇；在使用 printf()函数输出信息时，不用知道底层究竟是怎样把相应的信息输出到屏幕或串口的。也就是说，应用软件工程师需要看到一个没有硬件的纯粹的软件世界，硬件必须被透明地呈现给他们。谁来实现硬件对应用软件工程师的隐形？这个艰巨的任务就落在了驱动工程师的身上。对设备驱动最通俗的解释就是“驱使硬件设备行动”。设备驱动与底层硬件直接打交道，按照硬件设备的具体工作方式读/写设备寄存器，完成设备的轮询、中断处理、DMA 通信，进行物理内存向虚拟内存的映射，最终使通信设备能够收发数据，使显示设备能够显示文字和画面，使存储设备能够记录文件和数据。由此可见，设备驱动充当了硬件和应用软件之间的纽带，它使得应用软件只需要调用系统软件的应用编程接口（API）就可让硬件去完成要求的工作。在系统中没有操作系统的情况下，工程师可以根据硬件设备的特点自行定义接口，如对串口定义 SerialSend()、SerialRecv()；对 LED 定义 LightOn()、LightOff()；以及对 Flash 定义 FlashWrite()、FlashRead()等。而在有操作系统的情况下，设备驱动的架构则由相应的操作系统定义，驱动工程师必须按照相应的架构设计设备驱动，这样，设备驱动才能良好地整合到操作系统的内核中。驱动程序沟通着硬件和应用软件，而驱动工程师则沟通着硬件工程师和应用软件工程师。

随着通信、电子行业的迅速发展，全世界每天都会有大量的新芯片被生产，大量的新电路板被设计，因此，也会有大量设备驱动需要开发。这些设备驱动，或运行在简单的单任务环境中，或运行在 VxWorks、Linux、Windows 等多任务操作系统环境中，发挥着不可替代的作用。

5.2.2 内核驱动程序

内核是操作系统最基本的部分。它是为众多应用程序提供对计算机硬件的安全访问的一部分软件，这种访问是有限的，并且内核决定一个程序在什么时候对某部分硬件操作多长时间。直接对硬件操作是非常复杂的，所以内核通常提供一种硬件抽象的方法来完成这些操作。硬件抽象隐藏了复杂性，为应用软件和硬件提供了一套简洁、统一的接口，使程序设计更为简单。这就是内核空间实现的驱动程序，它使用内核资源、内核栈，包括可加载的内核驱动模块。

5.2.3 用户空间驱动程序

在用户空间实现驱动程序是把驱动程序的一部分放在用户空间，但不能将驱动程序的任意部分放在用户空间内。而且并非所有设备驱动都在用户空间内实现，因为对于大部分设备驱动程序是在内核空间实现的。只有设备种类众多、驱动程序较复杂、能够分层时，为了使用方便，其中上层或增加的驱动程序部分可在用户空间实现。

5.2.4 Linux 设备驱动概述

1. Linux 设备驱动

操作系统是通过各种驱动程序来驾驭硬件设备的，它为用户屏蔽了各种各样的设备，驱动硬件是操作系统最基本的功能，并且提供统一的操作方式。设备驱动程序是操作系统最基本的组成部分之一，在 Linux 内核源程序中也占有 60%以上。因此，熟悉驱动的编写是很重要的。

Linux 的一个重要特点就是将所有的设备都当做文件进行处理，这类特殊文件就是设备文件（通常在 /dev 目录下），这样，在应用程序看来，硬件设备只是一个设备文件，应用程序可以像操作普通文件一样对硬件设备进行操作，从而大大方便了对设备的处理。

Linux 系统的设备分为 3 类：字符设备、块设备和网络设备。

- 字符设备通常指像普通文件或字节流一样，以字节为单位顺序读/写的设备， 如并口设备、虚拟控制台等。字符设备可以通过设备文件节点访问，它与普通文件之间的区别在于，普通文件可以被随机访问（可以前后移动访问指针），而大多数字符设备只能提供顺序访问，因为对它们的访问不会被系统所缓存。但也有例外，例如，帧缓存（framebuffer）是一个可以被随机访问的字符设备。
- 块设备通常指一些需要以块为单位随机读/写的设备，如 IDE 硬盘、SCSI 硬盘、光驱等。它不仅可以提供随机访问，而且可以容纳文件系统（如硬盘、闪存等）。Linux 可以使用户态程序像访问字符设备一样每次进行任意字节的操作，只是在内核态内部中的管理方式和内核提供的驱动接口上不同。

通过文件属性可以查看它们是哪种设备文件（字符设备文件或块设备文件）。

```
$ ls -l /dev
crw-rw----  1 root  uucp 4,  64 08-30 22:58 ttyS0 /*串口设备， c 表示字符设备*/
brw-r-----  1 root  floppy 2,   0 08-30 22:58 fd0/*软盘设备，b 表示块设备*/
```

网络设备通常是指通过网络能够与其他主机进行数据通信的设备，如网卡等。

内核和网络设备驱动程序之间的通信调用一套数据包处理函数，它们完全不同于内核和字符，以及块设备驱动程序之间的通信（read()、write()等函数）。Linux 网络设备不是面向流的设备，因此不会将网络设备的名称（如 eth0）映射到文件系统中去。

2. Linux 设备驱动程序的特点

Linux 中的设备驱动程序有如下几个特点。

- 内核代码：设备驱动程序是内核的一部分，如果驱动程序出错，则可能导致系统崩溃。
- 内核接口：设备驱动程序必须为内核或者其子系统提供一个标准接口。比如，一个终端驱动程序必须为内核提供一个文件 I/O 接口；一个 SCSI 设备驱动程序应该为 SCSI 子系统提供一个 SCSI 设备接口，同时 SCSI 子系统也必须为内核提供文件的 I/O 接口及缓冲区。
- 内核机制和服务：设备驱动程序使用一些标准的内核服务，如内存分配等。
- 可装载：大多数的 Linux 操作系统设备驱动程序都可以在需要时装载进内核，在不需要时从内核中卸载。
- 可设置：Linux 操作系统设备驱动程序可以集成为内核的一部分，并可以根据需要把其中的某一部分集成到内核中，这只需要在系统编译时进行相应的设置。
- 动态性：在系统启动且各个设备驱动程序初始化后，驱动程序将维护其控制的设备。如果该设备驱动程序控制的设备不存在也不影响系统的运行，那么此时的设备驱动程序只是多占用了一点系统内存。

3．Linux 设备驱动程序与整个软硬件系统的关系

除网络设备外，字符设备与块设备都被映射到 Linux 文件系统的文件和目录，通过文件系统的系统调用接口 open()、write()、read()、close()等函数即可访问字符设备和块设备。所有的字符设备和块设备都被统一地呈现给用户。块设备比字符设备复杂，在它上面会首先建立一个磁盘/Flash 文件系统，如 FAT、Ext3、YAFFS、JFFS2 等。它们规范了文件和目录在存储介质上的组织。

应用程序可以使用 Linux 系统调用接口编程，也可以使用 C 库函数，出于代码可移植性的考虑，后者更值得推荐。C 库函数本身也通过系统调用接口而实现的，如 C 库函数中的 fopen()、fwrite()、fread()、fclose()分别会调用操作系统 API 的 open()、write()、read()、close()函数。

Linux 设备驱动程序与整个软硬件系统的关系如图 5.1 所示。

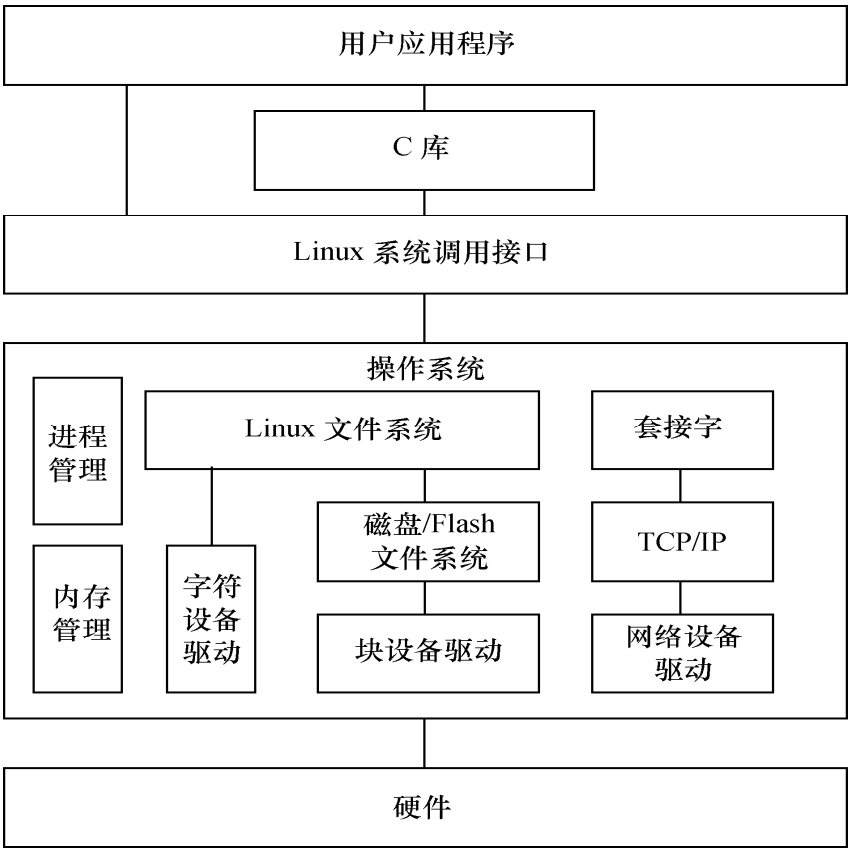

图 5.1　Linux 设备驱动与整个软硬件系统的关系

5.3　Linux 核心与 Android 驱动

Android 中内核的结构和标准的 Linux 2.6 内核是基本相同的，Android 在其基础上增加了私有内容。Android 在 Linux 内核中增加的主要是一些驱动程序，这些驱动程序主要分为两种：Android 专用驱动和 Android 使用的设备驱动。

5.3.1　Android 专用驱动

Android 专用驱动程序不是 Linux 的标准驱动，它们的作用是辅助系统运行，一般不操作实际硬件。

- 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：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。

下面重点介绍 Android 专用驱动中最重要的 3 个驱动。

1．Ashmem 匿名共享内存驱动

Ashmem（Anonymous Shared Memory），即匿名共享内存，通过内核的机制，为用户空间程序提供分配内存的机制。

Ashmem 设备节点名称为/dev/ashmem，主设备号为 10（Misc Driver），次设备号动态生成。

Ashmem 的代码路径如下：

```
kernel/include/linux/ashmem.h
kernel/mm/ashmem.c
```

2．Logger：轻量级的 log 驱动

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 3 个设备节点，分别为/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。

Logger 驱动的代码路径如下：

```
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/可执行程序。
```

3．Binder 驱动（Binder Driver）：

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

Binder 设备节点名称为/dev/binder，主设备号为 10（Misc Driver），次设备号动态生成。

Binder 的代码路径如下：

```
kernel/include/linux/binder.h
kernel/drivers/misc/binder.c
```

Android 系统上层用户空间的 libutil 工具库和 ServiceManager 守护进程都是调用 Binder 接口驱动提供对整个系统进程间通信功能的支持，它们的代码路径如下：

```
frameworks/base/cmds/servicemanager/
frameworks/base/include/utils/
frameworks/base/libs/utils/
```

Binder 是 Android 中主要使用的 IPC 方式，通常需要按照模板定义相关的类，不需要直接调用 Binder 驱动程序的设备节点。

5.3.2 Android 使用的设备驱动

Android 中常使用的设备主要有 Framebuffer 驱动、输入设备驱动、v4l2 摄像头－视频驱动、OSS 音频驱动、ALSA 音频驱动、MTD 驱动、蓝牙驱动、Wlan 驱动。

1．Framebuffe 显示驱动

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。

Framebuffe 显示驱动工作原理如图 5.2 所示。

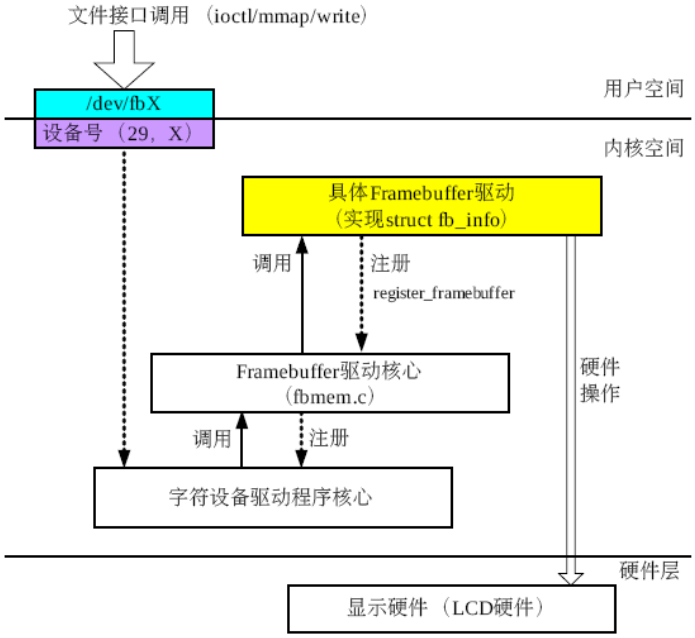

图 5.2　Framebuffe 显示驱动工作原理图

2．输入设备驱动

Android 中的输入设备设备驱动主要包括：游戏杆（Joystick）、鼠标（Mouse）和事件设备（Event）。

输入设备驱动同样也是字符设备，主设备号是 13，次设备号是 64~95 之间自动生成的，这个输入设备驱动程序那是相当相当的复杂。在 Android 内核中主要需要关注以下几个文件：

- include/linux/input.h（驱动头文件）。
- driver/input/input.c（驱动核心实现，包含大量的操作接口）。
- driver/input/event.c（event 机制）。
- driver/input/joydev.c（joystick 驱动）。
- driver/input/mousedev.c（鼠标驱动）。

其实上面这些东西都不需我们去实现，内核已经帮我们实现好了，不过在写硬件驱动时需要和 Inputcore 交互，所以需要用到上面这些函数中的接口，也就是说上面这些函数是透明的。

Event 输入设备驱动工作原理如图 5.3 所示。

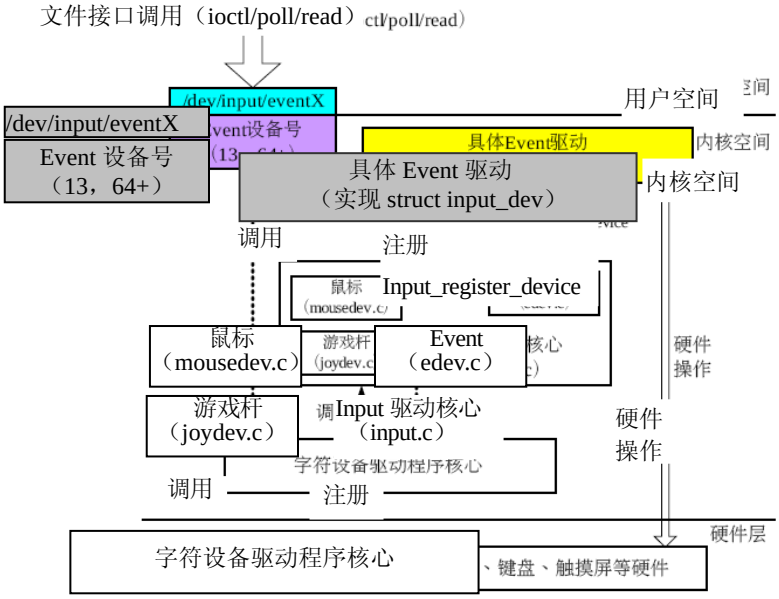

图 5.3　Event 输入设备驱动工作原理图

3．v4l2 摄像头－视频驱动

摄像头（Camera）－视频驱动驱动通常使用 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。
- v412 摄像头-视频驱动工作原理如图 5.4 所示。

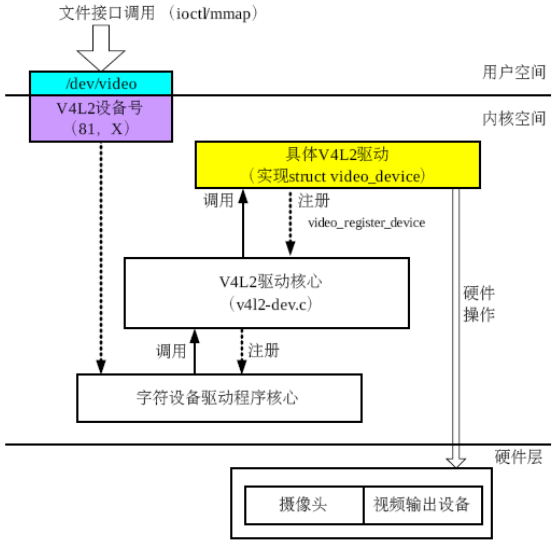

图 5.4　v4l2 摄像头－视频驱动工作原理图

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。

OSS 音频驱动工作原理如图 5.5 所示。

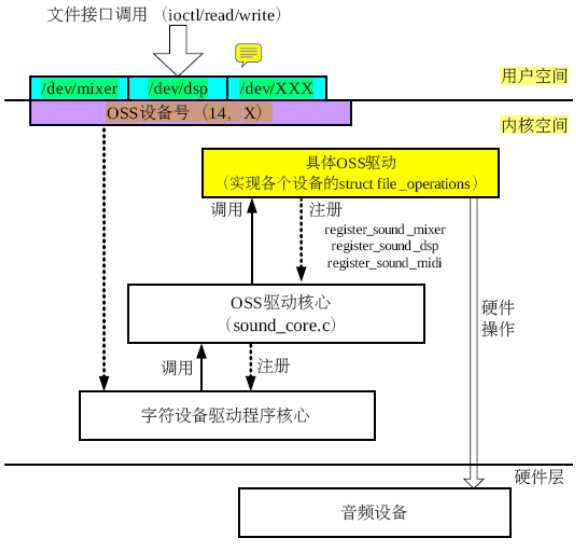

图 5.5　OSS 音频驱动工作原理图

5．ALSA 音频驱动

ALSA（Advanced Linux Sound Architecture）即高级 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。

ALSA 音频驱动工作原理如图 5.6 所示。

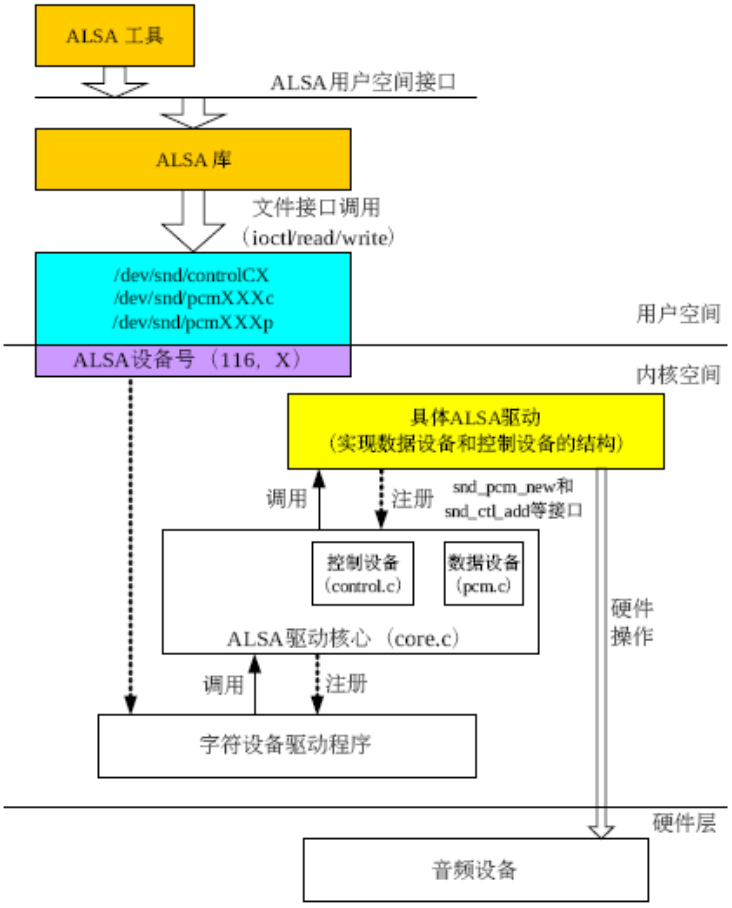

图 5.6　ALSA 音频驱动工作原理图

6．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。

MTD 驱动工作原理如图 5.7 所示。

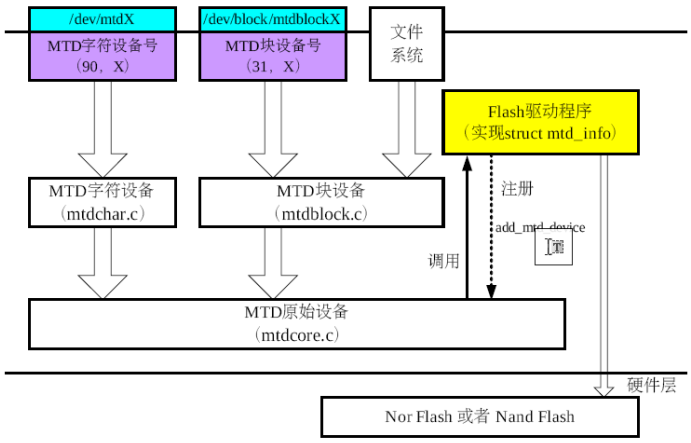

图 5.7　MTD 驱动工作原理图

7．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/*。

8．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/*。

5.4　Android 驱动的 HelloWorld

上述介绍的都是关于驱动理论，下面介绍一个 Android 实例。

代码如下：

```
/*
 * hello.c
 *
 * Simple hello world 2.6 driver module with module_init, module_exit
 *
 * Copyright (C) 2005 Farsight
 *
 * This program is free software; you can redistribute it and/or modify
 * it under the terms of the GNU General Public License as published by
 * the Free Software Foundation; either version 2 of the License, or
 * (at your option) any later version.

 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.

 * You should have received a copy of the GNU General Public License
```

```
 * along with this program; if not, write to the Free Software
 * Foundation, Inc., 51 Franklin St, Fifth Floor, Boston, MA  02110-1301  USA
 *
 */


#include <linux/module.h>
#include <linux/kernel.h>
#include <linux/init.h>

MODULE_LICENSE ("GPL");

static int __init hello_2_init (void)
{
 printk (KERN_INFO "Hello world\n");
 return 0;
}

static void __exit hello_2_exit (void)
{
 printk (KERN_INFO "Goodbye world\n");
}

module_init (hello_2_init);
module_exit (hello_2_exit);

makefile 文件:
ifeq ($(KERNELRELEASE),)

#KERNELDIR ?= /home/lht/kernel2.6/linux-2.6.14

KERNELDIR ?= /lib/modules/$(shell uname -r)/build M=$(PWD) modules
PWD := $(shell pwd)

modules:
 $(MAKE) -C $(KERNELDIR) M=$(PWD) modules

modules_install:
 $(MAKE) -C $(KERNELDIR) M=$(PWD) modules_install

clean:
 rm -rf *.o *~ core .depend .*.cmd *.ko *.mod.c .tmp_versions

.PHONY: modules modules_install clean

else
   obj-m := hello.o
endif
```

其中有些是见过的，第一个 ifeq ($(KERNELRELEASE),)目前并无用处，它的由来是指在 Linux 源码根目录下的 Makefile 编译内核时，KERNELRELEASE 宏会被定义，那么如果是从源码根目录开始的 make 则会将 myhello.o 模块编译进内核。

KERNELDIR ?= /home/linux/linux-2.6.22.6 是对 KERNELDIR 进行赋值，这个变量是后面我们用到的指代内核源码目录用的。

PWD := $(shell pwd)是对 PWD 变量进行赋值，作用是将$(shell pwd)的返回结果即求得当前目录的路径赋值给 PWD，这个变量在后面指代要编译的驱动程序所在的位置。

```
modules:
      $(MAKE) -C $(KERNELDIR) M=$(PWD) modules
```

这句是 Makefile 的规则：这里的$(MAKE)就相当于 make，-C 选项的作用是指将当前工作目录转移到所指定的位置。“M=”选项的作用是当用户需要以某个内核为基础编译一个外部模块时，需要在 make modules 命令中加入“M=dir”，程序会自动到所指定的 dir 目录中查找模块源码将其编译，生成 KO 文件。

```
modules_install:
            $(MAKE) -C $(KERNELDIR) M=$(PWD) modules_install
```

这个命令是模块的安装，在 Makefile 中搜索“lib\/modules”可以看到下面的语句，通过阅读不难找到这个“MODLIB”的用处，它是用来指定安装路径的，而变量“INSTALL_MOD_PATH”往往为空。

```
MODLIB = $(INSTALL_MOD_PATH)/lib/modules/$(KERNELRELEASE)
       Export MODLIB

.PHONY: modules modules_install clean
```

这句话的作用是保证 modules、modules_install,clean 这 3 个命令能正常完成。PHONY 是一个特殊目标名称，还有其他的，如.SUFFIXES、.DEFAULT、.PRECIOUS、.INTERMEDIATE、.SECONDARY、.SECONDEXPANSION、.DELETE_ON_ERROR、.IGNORE .LOW_RESOLUTION_TIME .SILENT .EXPORT_ALL_VARIABLES .NOTPARALLEL。

它们的具体用法可以参考 GNU 手册中的 Special Built-in Target Names 章节。

.PHONY 目标的具体意思是如果在 Makefile 的工作目录中有名如 modules、modules_install、clean 等文件时，命令会出错。

实验：简单模块编写实验

【实验内容】

编写一个最基本的模块，加载模块，观察结果。

【实验步骤】

（1）将文件夹 ex1-hello-world 复制到 Linux 环境中，如/home/linux/test。

```
#su root
#cp ex1-hello-world /home/linux/test –a
```

（2）#cd /home/linux/test/ex1-hello-world。

（3）#make。

（4）通过 insmod 命令将模块加入内核：

```
#insmod insmod hello.ko
```

（5）通过 lsmod 查看内核模块：

```
# lsmod | grep hello
```

（6）通过 rmmod 删除内核中的模块：

```
# rmmod hello
```

5.5 小结

本章讲述了 Android 驱动开发的基本概念、设备驱动的概念、内核驱动程序和用户空间驱动个程序。介绍了什么是嵌入式系统，概述了 Android 平台的概念和特点，以及嵌入式 Android 的系统架构。重点介绍 Linux 核心与 Android 驱动的知识。通过本章的学习，可初步了解驱动程序的概念和 Android 驱动的开发原理。另外，还讲述了 Android 设备驱动的开发环境的搭建，并创建简单的开发示例。

5.6 思考题

1．简述驱动程序和应用程序的区别。

2．简述 Linux 驱动和 Android 驱动的异同。

联系方式

集团官网：www.hqyj.com　　嵌入式学院：www.embedu.org　　移动互联网学院：www.3g-edu.org

企业学院：www.farsight.com.cn　　物联网学院：www.topsight.cn　　研发中心：dev.hqyj.com

集团总部地址：北京市海淀区西三旗悦秀路北京明园大学校内 华清远见教育集团

北京地址：北京市海淀区西三旗悦秀路北京明园大学校区，电话：010-82600386/5

上海地址：上海市徐汇区漕溪路 250 号银海大厦 11 层 B 区，电话：021-54485127

深圳地址：深圳市龙华新区人民北路美丽 AAA 大厦 15 层，电话：0755-22193762

成都地址：成都市武侯区科华北路 99 号科华大厦 6 层，电话：028-85405115

南京地址：南京市白下区汉中路 185 号鸿运大厦 10 层，电话：025-86551900

武汉地址：武汉市工程大学卓刀泉校区科技孵化器大楼 8 层，电话：027-87804688

西安地址：西安市高新区高新一路 12 号创业大厦 D3 楼 5 层，电话：029-68785218

广州地址：广州市天河区中山大道 268 号天河广场 3 层，电话：020-28916067