# ALSA驱动框架

sound/core/sound.c

与应用层接口

static const struct file\_operations snd\_fops = {

.owner = THIS\_MODULE,

.open = snd\_open,

.llseek = noop\_llseek,

};

snd\_open(struct inode \*inode, struct file \*file)//通用的open接口函数

mptr = snd\_minors[minor]; //从snd\_minors数组中取出一个结构体

new\_fops = fops\_get(mptr->f\_ops);//找到f\_ops结构体

replace\_fops(file, new\_fops);//file指针指向新的fops

err = file->f\_op->open(inode, file)//使用新fops里面的open函数打开设备

何处注册了snd\_minors？

int snd\_register\_device(int type, struct snd\_card \*card, int dev,

const struct file\_operations \*f\_ops,

void \*private\_data, struct device \*device)

struct snd\_minor \*preg;

preg->f\_ops = f\_ops;

minor = snd\_find\_free\_minor(type, card, dev);

snd\_minors[minor] = preg;

snd\_minor结构体

struct snd\_minor {

int type; /\* SNDRV\_DEVICE\_TYPE\_XXX \*/

int card; /\* card number \*/

int device; /\* device number \*/

const struct file\_operations \*f\_ops; /\* file operations \*/

void \*private\_data; /\* private data for f\_ops->open \*/

struct device \*dev; /\* device for sysfs \*/

struct snd\_card \*card\_ptr; /\* assigned card instance \*/

};

snd\_register\_device由谁调用？

sound/core/control.c里面有调用，调用函数snd\_ctl\_dev\_register

sound/core/pcm.c里面有调用，调用函数snd\_pcm\_dev\_register

分析sound/core/control.c

static int snd\_ctl\_dev\_register(struct snd\_device \*device)

snd\_register\_device（）

snd\_ctl\_dev\_register由snd\_ctl\_create调用

snd\_ctl\_create

static struct snd\_device\_ops ops = {

.dev\_free = snd\_ctl\_dev\_free,

.dev\_register = snd\_ctl\_dev\_register,

.dev\_disconnect = snd\_ctl\_dev\_disconnect,

};

snd\_ctl\_create由init.c (sound\core)里面的snd\_card\_new调用

int snd\_card\_new(struct device \*parent, int idx, const char \*xid,

struct module \*module, int extra\_size,

struct snd\_card \*\*card\_ret)

snd\_card\_new在特定的声卡文件里面调用

分析sound/core/pcm.c

static int snd\_pcm\_dev\_register(struct snd\_device \*device)

snd\_pcm\_dev\_register由\_snd\_pcm\_new调用

static int \_snd\_pcm\_new(struct snd\_card \*card, const char \*id, int device,

int playback\_count, int capture\_count, bool internal,

struct snd\_pcm \*\*rpcm)

🡪static struct snd\_device\_ops ops = {

.dev\_free = snd\_pcm\_dev\_free,

.dev\_register = snd\_pcm\_dev\_register,

.dev\_disconnect = snd\_pcm\_dev\_disconnect,

};

\_snd\_pcm\_new 由snd\_pcm\_new调用的

int snd\_pcm\_new(struct snd\_card \*card, const char \*id, int device,

int playback\_count, int capture\_count, struct snd\_pcm \*\*rpcm)

{

return \_snd\_pcm\_new(card, id, device, playback\_count, capture\_count,

false, rpcm);

}

snd\_pcm\_new

soc\_new\_pcm

分析wm8978声卡

static struct i2c\_driver wm8978\_i2c\_driver = {

.driver = {

.name = "wm8978",

},

.probe = wm8978\_i2c\_probe,

.remove = wm8978\_i2c\_remove,

.id\_table = wm8978\_i2c\_id,

};

wm8978\_i2c\_probe

ret = snd\_soc\_register\_codec(&i2c->dev,

&soc\_codec\_dev\_wm8978, &wm8978\_dai, 1);

snd\_soc\_register\_dais

ret = snd\_soc\_instantiate\_card(card);

# ALSA-lib 学习