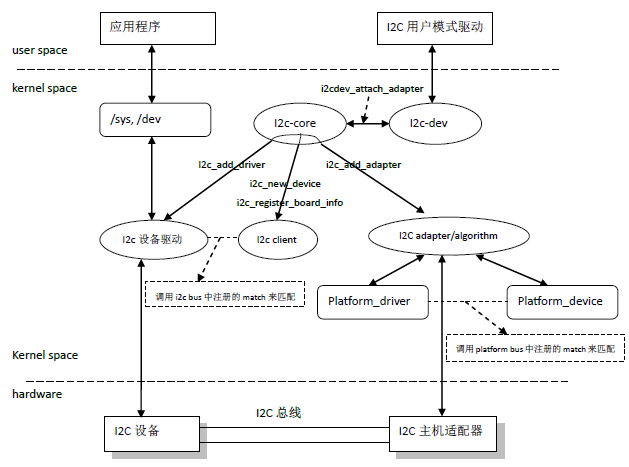
# Linux驱动\_I2C子系统

## 1. I2C子系统架构

### 1.1 系统架构图



### 1.2 三大组成部分

**I2C核心(i2c-core)**  
I2C核心提供了I2C总线驱动和设备驱动的注册、注销方法，I2C通信方法(algorithm)上层的、与具体适配器无关的代码以及探测设备、检测设备地址的上层代码等。

**I2C总线驱动(I2C adapter/Algo driver)**I2C总线驱动是I2C适配器的软件实现，提供I2C适配器与从设备间完成数据通信的能力。I2C总线驱动由i2c\_adapter和i2c\_algorithm来描述。

**I2C客户驱动程序(I2C client driver)**  
I2C客户驱动是对I2C从设备的软件实现，一个具体的I2C客户驱动包括两个部分：一部分是i2c\_driver，用于将设备挂接于i2c总线；另一部分是设备本身的驱动。  
I2C客户驱动程序由i2c\_driver和i2c\_client来描述。

所有的I2C驱动代码位于drivers/i2c目录下：

I2c**-**core**.**c    实现I2C核心的功能

I2c**-**dev**.**c   通用的从设备驱动

Chips       特定的I2C设备驱动

Busses      I2C适配器的驱动

Algos       实现了一些I2C总线适配器的algorithm

从上面的图我们可以看到两种编写驱动方法，一种是利用系统提供的i2c-dev.c来实现一个i2c适配器的设备文件，然后通过在应用层操作I2C适配器来控制I2C设备；另一种是为I2C从设备独立编写一个设备驱动，不需要i2c-dev.c文件。

## 2. I2C设备代码编写

### 2.1 AT24c08的I2C代码

#### 2.1.1 i2c\_client

创建一个i2c设备，该设备名称为at24c08，i2c地址为0x50。

//所支持的i2c设备的列表

static struct i2c\_board\_info at24cxx\_info **=** **{**

//一项代表一个支持的设备，它的名字叫做“at24c08”，器件地址是0x50

I2C\_BOARD\_INFO**(**"at24c08"**,** 0x50**),**

**};**

static struct i2c\_client **\***at24cxx\_client**;**

static int at24cxx\_dev\_init**(**void**)**

**{**

//分配一个适配器的指针

struct i2c\_adapter **\***i2c\_adap**;**

//调用core层的函数，获得一个i2c总线。这里我们已经知道新增的器件挂接在编号为0的i2c总线上

i2c\_adap **=** i2c\_get\_adapter**(**0**);**

//把i2c适配器和新增的I2C器件关联起来，这个用了i2c总线0，地址是0x50。这就组成了一个客户端。

at24cxx\_client **=** i2c\_new\_device**(**i2c\_adap**,** **&**at24cxx\_info**);**

//增加i2c适配器的引用

i2c\_put\_adapter**(**i2c\_adap**);**

**return** 0**;**

**}**

static void at24cxx\_dev\_exit**(**void**)**

**{**

i2c\_unregister\_device**(**at24cxx\_client**);**

**}**

module\_init**(**at24cxx\_dev\_init**);**

module\_exit**(**at24cxx\_dev\_exit**);**

MODULE\_LICENSE**(**"GPL"**);**

#### 2.1.2 i2c\_driver

创建了i2c设备，接下来需要创建设备对应的驱动程序。