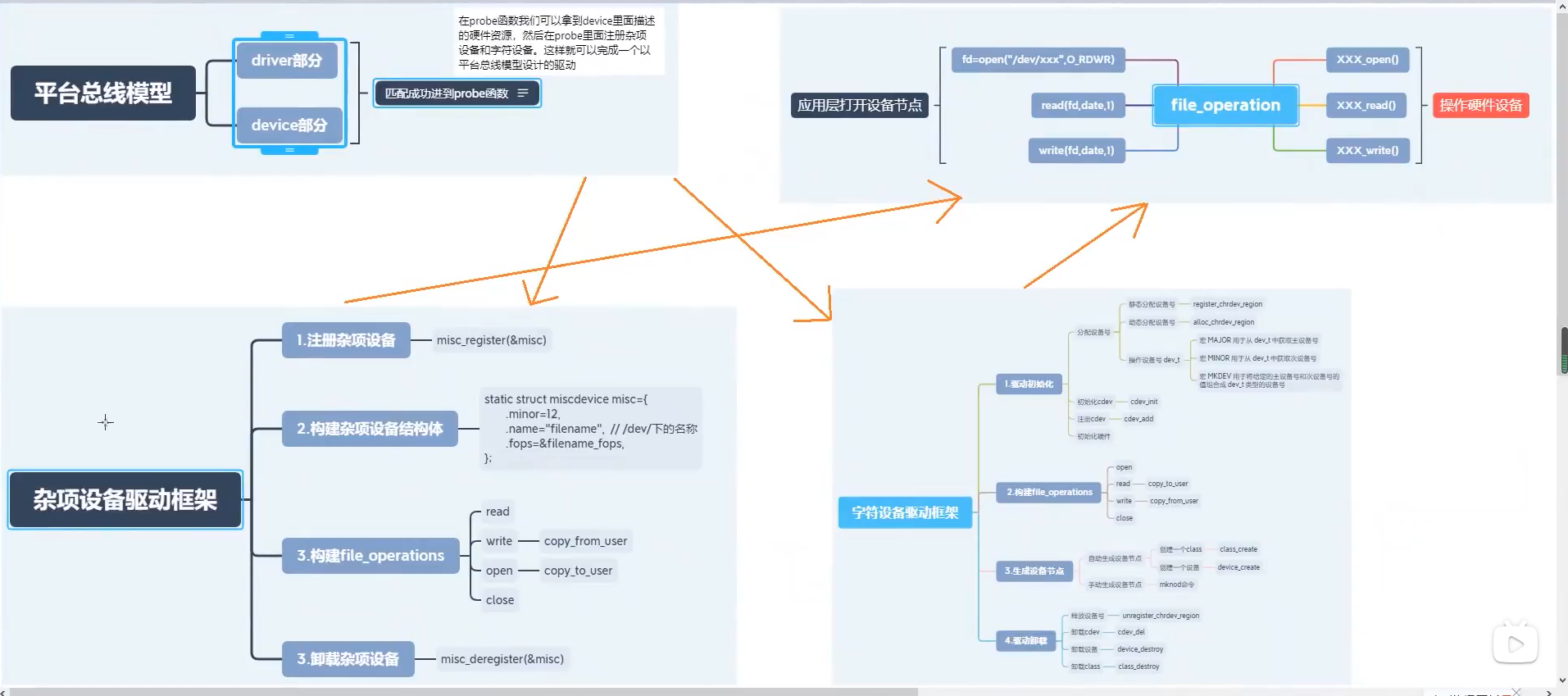
总框图:



左上角：平台总线模型

平台总线模型分为driver（驱动相关的代码）和device部分（与硬件相关的一些描述）。当driver 和device 以名字进行匹配成功后，就会进入probe函数。我们就可以在probe里面通过另外一些函数获得device里面的硬件资源。进而就可以在probe里面注册杂项/字符设备。从而完成一个以平台总线为模型设计的驱动。

下面两个图：杂项设备驱动框架 和 字符设备驱动框架：

也可以说是:我们进入probe函数后可以做的事情。也就是说我们在probe函数里面可以按照杂项设备 或 字符设备的注册框架流程来注册设备。

右上角：应用层打开设备节点：

字符设备和杂项设备 与 应用层 依靠file\_operation进行连接。我们注册完字符设备或杂项设备后，我们就可以通过file\_operation结构体操作硬件。我们在应用层打开设备节点，然后read、write就可以执行对硬件的控制操作。

我们之所以学习平台总线模型，是因为linux里面的驱动都是以平台总线模型进行设计。如果不了解平台总线模型的概念，在之后学习linux源码的时候是看不懂的。所以需要了解平台总线的概念。

平台总线的好处就是：他把driver和device分开了，就可以提高驱动代码的重用性，以及 大量减少重复性代码。并且他把驱动都挂载到了总线上，也方便管理。