Device\_add( )

首先使用get\_device( )函数来得到需要添加的设备，该设备的引用计数 到底还是增加dev->kobj->kref的计数

然后使用device\_private\_init( )函数将数据私有化，没有的话就申请并初始化,这个数据很重要,它是连接所属bus,parent,对应驱动等的重要连接点。

然后使用if 判断当前需要添加的设备名是否存在，存在即使用dev\_set\_name( )函数初始化dev-kobject->name

如果deice的init\_name不存在且dev-kobject->name也不存在,则使用dev\_set\_name( )函数使用bus的dev\_name和dev\_id来设置目录名

如果上面的操作都没有找到可设置的目录名，则报错。

将父设备的引用计数+1

并使用get\_device\_parent( )获取设备的父设备，拿到父节点，并将其获取结果赋值给kobj，如果成功获取到了父节点，就将拿到的父节点赋值给dev->kobj.parent,确定设备父子关系,也确定了sysfs中的目录关系。

然后使用if判断父设备是否存在，存在则使用set\_dev\_node( )函数设置该设备节点为-1，一般未注册前在device\_initialize已经初始化为-1

使用kobject\_add( )函数把内嵌的kobject注册到设备模型中将设备加入到kobject模型中，创建sys相关目录，目录名字为kobj->name

使用device\_create\_file( )函数创建sys目录下设备的uevent属性文件，通过它可以查看设备的uevent事件,主要是在/sys/devices/.../中添加dev的uevent属性文件\*/

使用if判断MAJOR（主设备号）存在，则产生dev属性,并在/dev目录下产生设备节点文件

然后使用device\_create\_file( )函数创建sys目录下设备的设备号属性，即major和minor /主要是在sys/devices/...中添加dev属性文件

device\_create\_sys\_dev\_entry( )函数在/sys/dev/char/或者/sys/dev/block/创建devt的属性的连接文件,形如10:45,由主设备号和次设备号构成,指向/sys/devices/.../的具体设备目录\*/

//该链接文件只具备读属性，显示主设备号：次设备号，如10:45,用户空间udev响应uevent事件时，将根据设备号在/dev下创建节点文件

device\_add\_class\_symlinks( )函数实际创建的kobject都是在device下面,其他class,bus之类的里面的具体设备都是device目录下设备的符号链接,这里是在class下创建符号链接

device\_add\_attrs( )函数创建sys目录下设备其他属性文件(添加设备属性文件)

bus\_add\_device( ) 添加设备的总线属性，将设备加入到管理它的bus总线的设备连表上，创建subsystem链接文件，链接class下的具体的子系统文件夹 将设备添加到其总线的设备列表中。

dpm\_sysfs\_add( )函数把设备增加到sysfs电源管理power目录（组）下,如果该设备设置电源管理相关的内容

device\_pm\_add( )设备添加到电源管理相关的设备列表中

if (dev->bus)//通知客户端,有新设备加入

产生一个内核uevent事件（这里是有设备加入），可以是helper,也可是通过netlink机制和用户空间通信该事件可以被内核以及应用层捕获，属于linux设备模型中热插拔机制

bus\_probe\_device( )函数给设备探测寻找相对应的驱动，在bus上找dev对应的drv,主要执行\_\_device\_attach,主要进行match,sys\_add,执行probe函数和绑定等操作

klist\_add\_tail(&dev->p->knode\_parent添加新设备到父设备的子列表中

if (dev->class) 来判断如果改dev有所属类，则将dev的添加到类的设备列表里面

mutex\_lock(&dev->class->p->mutex);//要使用class的互斥锁

klist\_add\_tail(&dev->knode\_class,//dev添加到class的klist\_device链表（对driver也有klist\_driver链表）

通知有新设备加入,执行该dev的class\_intf->add\_dev(),好处是只有设备匹配注册成功了，才进行其它的注册工作(如字符设备的注册，生成/dev/\*\*\*节点文件)以及部分初始化工作。

device\_private\_init

初始化私有数据，很重要的一个数据,它是连接所属bus、parent和对应驱动等的重要连接桥梁。

bus\_add\_device

将新建设备添加到对应的bus上。

bus\_add\_device(struct device \*dev)

struct bus\_type \*bus = bus\_get(dev->bus)拿到device对应的总线，且把总线引用计数加1

device\_add\_attrs(bus, dev);//创建添加相应属性

sysfs\_create\_link( );在sys/bus/总线类型/devices/目录下,创建名字为dev\_name的符号链接，指向sys/devices/相同设备名字（前面创建了class目录下的,这创建bus目录下的符号链接）

sysfs\_create\_link( )在sys/devices/分类/设备名字/目录下创建目录名字为subsystem的符号链接，并且指向在sys/bus/总线类型

klist\_add\_tail( )设备加到总线链表中

bus\_probe\_device( )

if (!bus)确定总线存在，否则出错返回

if ( ) drivers\_autoprobe是一个bit变量,为l则允许本条总线上的device注册时自动匹配driver,drivers\_autoprobe默认总是为1,除非用户空间修改

device\_attach( )匹配设备和驱动

device\_attach

1.默认注册的device自带有驱动，只需要调用device\_bind\_driver绑定，不需要匹配。

2.如果没有带有驱动，则需要后面遍历总线，调用匹配函数\_\_device\_attach分析匹配对应的驱动

if ( ) driver已经放在device了（初始化device,时,手动添加的driver）

device\_bind\_driver( ) driver放在device里了,但还没真正的绑定 ,则执行这个函数绑定

else {刚注册的device，没有添加对应的driver,需要查找匹配对应的驱动

bus\_for\_each\_drv( )遍历总线上的driver链表,一个一个进行匹配

device\_bind\_driver

device上有driver，只需绑定即可

driver\_sysfs\_add(dev);//把dev和driver链接起来

driver\_bound(dev);//device里面私有的driver节点挂接到driver的设备链表（一个driver可能对应多个device）

driver\_sysfs\_add( )

blocking\_notifier\_call\_chain( )通知其它总线将要绑定driver 到device

kobject\_name( )在driver目录下创建device目录的符号链接,名字为设备的名字

sysfs\_create\_link( )在device目录下创建driver的目录,名字为driver

driver\_bound( )

if (klist\_node\_attached(&dev->p->knode\_driver)) {//再次检查,确定没绑定

绑定!!!!!! 把device私有的p里的knode\_driver,绑定到driver里面的klist\_device链表上

klist\_add\_tail(&dev->p->knode\_driver, &dev->driver->p->klist\_devices);

if (dev->bus)//通知其它子模块以及绑定成功

bus\_for\_each\_drv( )

如果设备没有对应的驱动，则需要遍历总线上的driver链表,一个一个进行匹配。

bus\_for\_each\_drv(dev->bus, NULL, dev, \_\_device\_attach)

如果设备没有对应的驱动，则需要遍历总线上的driver链表,一个一个进行匹配。

 \_\_device\_attach

if (!driver\_match\_device(drv, dev))//确认匹配成功

return driver\_probe\_device(drv, dev);//调用probe函数

driver\_probe\_device( )

if (!device\_is\_registered(dev))//通过查看在sysfs中的状态判断是都已经注册

ret = really\_probe(dev, drv);//真正的probe函数

really\_probe

dev->driver = drv;//匹配好后的驱动信息记录到设备内部

if (driver\_sysfs\_add(dev)) {//driver加入sysfs（其实就是创建各种符号链接,前面device默认绑定有driver那里已经分析过了）

if (dev->bus->probe) {//如果总线上定义probe函数则调用

else if (drv->probe) {//否则调用驱动上的probe函数

driver\_bound(dev);//将设备加入到驱动支持的设备链表中，一个设备需要一个驱动，一个驱动支持多个设备,前面device默认绑定driver那里已经分析过了