in drivers/base/dd.c

```
1.  static atomic_t probe_count = ATOMIC_INIT(0);
2.  static DECLARE_WAIT_QUEUE_HEAD(probe_waitqueue);
```

really_probe()用于调用drv中的probe()来初始化dev device。

```
1.  static int really_probe(struct device *dev, struct device_driver *drv)
2.  {
3.          int ret = 0;
4.          int local_trigger_count = atomic_read(&deferred_trigger_count);
5.
6.          atomic_inc(&probe_count);


                ①
7.          pr_debug("bus: '%s': %s: probing driver %s with device %s\n",
8.                   drv->bus->name, __func__, drv->name, dev_name(dev));
9.          WARN_ON(!list_empty(&dev->devres_head));
10.
11.         dev->driver = drv;
12.
13.         /* If using pinctrl, bind pins now before probing */
14.         ret = pinctrl_bind_pins(dev);
15.         if (ret)
16.                 goto probe_failed;
17.
18.         if (driver_sysfs_add(dev)) {
19.                 printk(KERN_ERR "%s: driver_sysfs_add(%s) failed\n",
20.                         __func__, dev_name(dev));
21.                 goto probe_failed;
22.         }
23.
24.         if (dev->bus->probe) {
25.                 ret = dev->bus->probe(dev);
26.                 if (ret)
27.                         goto probe_failed;
28.         } else if (drv->probe) {
29.                 ret = drv->probe(dev);
30.                 if (ret)
31.                         goto probe_failed;
32.         }
33.
34.         driver_bound(dev);
35.         ret = 1;
36.         pr_debug("bus: '%s': %s: bound device %s to driver %s\n",
37.                  drv->bus->name, __func__, dev_name(dev), drv->name);
38.         goto done;
39.
40.  probe_failed:
41.         devres_release_all(dev);
42.         driver_sysfs_remove(dev);
43.         dev->driver = NULL;
44.         dev_set_drvdata(dev, NULL);
45.
46.         if (ret == -EPROBE_DEFER) {
47.                 /* Driver requested deferred probing */
48.                 dev_info(dev, "Driver %s requests probe deferral\n", drv->name);
49.                 driver_deferred_probe_add(dev);
50.                 /* Did a trigger occur while probing? Need to re-trigger if yes *
    /
```

```
                if (local_trigger_count != atomic_read(&deferred_trigger_count))
                        driver_deferred_probe_trigger();
        } else if (ret != -ENODEV && ret != -ENXIO) {
                /* driver matched but the probe failed */
                printk(KERN_WARNING
                       "%s: probe of %s failed with error %d\n",
                       drv->name, dev_name(dev), ret);
        } else {
                pr_debug("%s: probe of %s rejects match %d\n",
                       drv->name, dev_name(dev), ret);
        }
        /*
         * Ignore errors returned by ->probe so that the next driver can try
         * its luck.
         */
        ret = 0;
done:
        atomic_dec(&probe_count);
                                                                    ②
        wake_up(&probe_waitqueue);
        return ret;
}
```

整个function用probe_count来标示是否当前正在运行某个driver对特定device的probe动作,也就是driver正处于initialization中。

①在进入probe以前，probe_count + 1

②在离开probe以后，probe_count - 1

如果非零，表示忙着probe呢。

```
/**
 * wait_for_device_probe
 * Wait for device probing to be completed.
 */
void wait_for_device_probe(void)
{
        /* wait for the known devices to complete their probing */
        wait_event(probe_waitqueue, atomic_read(&probe_count) == 0);
        async_synchronize_full();
}
```

wait_for_device_probe() function就用于判断是否由device正在被某个driver probe。如果是(probe_count != 0)，则等待在probe_waitqueue里。