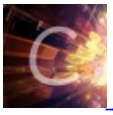
记一次misc_register注册失败

原创

那颗流星 于 2020-04-09 22:04:03 发布 864 收藏 2
分类专栏： DRIVERS

本文链接：https://blog.csdn.net/mike8825/article/details/105420609



在看内核log时，发现有一个misc_register注册失败的warning。一开始以为配置的参数有误，看代码也使用了参数MISC_DYNAMIC_MINOR，不太可能出现注册失败的情况。

```
int misc_register(struct miscdevice * misc)
{
 dev_t dev;
 int err = 0;
 bool is_dynamic = (misc->minor == MISC_DYNAMIC_MINOR);

 if (is_dynamic) {
  struct miscdevice *c;
  int i = find_first_zero_bit(misc_minors, DYNAMIC_MINORS);
  if (i >= DYNAMIC_MINORS) {
   err = -EBUSY;
   goto out;
  }
  list_for_each_entry(c, &misc_list, list) {
   if (c == misc) {
    err = -EEXIST;
    goto out;
   }
  }
  misc->minor = DYNAMIC_MINORS - i - 1;
  set_bit(i, misc_minors);
 } else {

     ...
 }

 ...
}
```

加打印发现is_dynamic为0，但有使用参数MISC_DYNAMIC_MINOR，不应该为0。出问题的代码如下，能看出哪里有问题没

```
static int xxx_img_probe(struct platform_device *pdev)
{
 int ret = 0;

 ret = misc_register(&image_dev);
 if (ret) {
  pr_err("cannot register miscdev on minor=%d (%d).\n",IMAGE_MINOR, ret);
  ret = -EACCES;
  goto exit;
 }
}

static const struct of_device_id  of_match_table_dcam[] = {
 { .compatible = "xxx,dcam"},
};

static struct platform_driver xxx_img_driver = {
 .probe = xxx_img_probe,
 .driver = {
  .of_match_table = of_match_ptr(of_match_table_dcam),
  },
};
```

发现xxx_img_probe跑了两遍，第一次注册成功了，导致misc->minor为固定值(非255)，导致不能动态注册该设备，使用静态注册时，该设备号已经被使用了，从而注册失败。

但xxx_img_probe为什么会跑两遍的，设备只有一个。一开始以为xxx_img_probe返回值为EPROBE_DEFER，导致xxx_img_probe再次运行。但打印发现xxx_img_probe返回值为0。只能加打印进行跟踪了。

```
const struct of_device_id *__of_match_node(const struct of_device_id *matches,
        const struct device_node *node)
{
 const struct of_device_id *best_match = NULL;
 int score, best_score = 0;

 if (!matches)
  return NULL;

 for (; matches->name[0] || matches->type[0] || matches->compatible[0]; matches++) {
  score = __of_device_is_compatible(node, matches->compatible,
        matches->type, matches->name);
  if (score > best_score) {
   best_match = matches;
   best_score = score;
  }
 }

 return best_match;
}
```

发现matches指向了其他的非法数据结构，导致probe函数第二次运行。

```
static const struct of_device_id  of_match_table_dcam[] = {
 { .compatible = "xxx,dcam"},
+       {                         },
};
```

罪魁祸首就是少了一个括号，导致野指针出现，出现了非法probe。这是某cpu厂商代码的bug，已提给他们修复。